YAYASAN PRIMA AGUS TEKNIK

CYBERSECURITY
BERBASIS
KECERDASAN BUATAN (AI)

Dr. Joseph Teguh Santoso, S.Kom, M.Kom.

Dr. Joseph Teguh Santoso, S.Kom, M.Kom.

CYBERSECURITY BERBASIS KECERDASAN BUATAN (AI)

YAYASAN PRIMA AGUS TEKNIK

PENERBIT :
YAYASAN PRIMA AGUS TEKNIK
Jl. Majapahit No. 605 Semarang
Telp. (024) 6723456. Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

ISBN 978-634-7227-40-9 (PDF)
9 786347 227409

Cybersecurity berbasis Kecerdasan Buatan (AI)

Penulis :
Dr. Joseph Teguh Santoso, S.Kom., M.Kom.

ISBN : 978-634-7227-40-9

Editor :
Dr. Agus Wibowo, M.Kom, M.Si, MM.

Penyunting :
Dr. Mars Caroline Wibowo. S.T., M.Mm.Tech

Desain Sampul dan Tata Letak :
Irdha Yunianto, S.Ds., M.Kom

Penebit :
Yayasan Prima Agus Teknik Bekerja sama dengan
Universitas Sains & Teknologi Komputer (Universitas STEKOM)

Anggota IKAPI No: 279 / ALB / JTE / 2023

Redaksi :
Jl. Majapahit no 605 Semarang
Telp. 08122925000
Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

Distributor Tunggal :
Universitas STEKOM
Jl. Majapahit no 605 Semarang
Telp. 08122925000
Fax. 024-6710144
Email : info@stekom.ac.id

KATA PENGANTAR

Puji dan syukur penulis panjatkan ke hadirat Tuhan Yang Maha Esa atas rahmat dan karunia-Nya sehingga buku berjudul *Cybersecurity berbasis Kecerdasan Buatan (AI)* ini dapat diselesaikan dengan baik. Buku ini disusun sebagai upaya untuk memberikan pemahaman yang komprehensif mengenai perkembangan teknologi kecerdasan buatan dan penerapannya dalam dunia keamanan siber yang semakin kompleks dan dinamis.

Buku ini mengupas secara mendalam berbagai aspek mulai dari konsep dasar kecerdasan buatan, pembelajaran mesin, jaringan syaraf tiruan, hingga aplikasi khusus seperti visi komputer dan implementasi gambar dimensi yang relevan dengan keamanan siber masa kini. Melalui bab-bab yang tersusun sistematis, pembaca diajak untuk memahami evolusi teknologi keamanan siber dalam ranah AI, teknik-teknik pembelajaran mesin termasuk dengan penggunaan python, serta berbagai aplikasi praktis untuk menghadapi tantangan keamanan jaringan dan data.

Bab Pertama membahas evolusi keamanan siber secara kronologis dan memperkenalkan konsep kecerdasan buatan (AI) pada ranah keamanan siber. Dijelaskan sub-bidang kecerdasan buatan, perkembangan sistem pakar, serta pentingnya data sebagai fondasi kecerdasan buatan di era keemasan AI. Bab selanjutnya, bab 2, menguraikan konsep pembelajaran mesin secara umum, proses dan aspek teoretisnya, termasuk pembelajaran tanpa supervisi dan perceptron. Pembahasan juga mencakup penerapan pembelajaran mesin untuk perlindungan titik akhir dalam sistem keamanan.

Bab ketiga menampilkan aplikasi pembelajaran mesin dalam konteks praktis, seperti pembangunan chatbot dan prediksi harga saham, menggunakan bahasa pemrograman Python sebagai alat utama. Pada bab Empat, Bab ini mengupas teori dasar jaringan syaraf tiruan, termasuk berbagai arsitektur seperti jaringan Hopfield, propagasi tandingan, hingga jaringan backpropagation berulang yang digunakan dalam pemodelan kecerdasan buatan.

Bab ke Lima buku ini memiliki pokok bahasan bab ini lebih mendalam pada jaringan syaraf tiruan untuk pembelajaran mendalam (deep learning), komponen yang terkait pada platform cloud seperti AWS, Microsoft Azure, Google Cloud, serta penerapan untuk pemeliharaan prediktif. Bab Enam menjelaskan sejarah dan teknik pembuatan gambar dalam visi komputer, serta pengembangan sistem Ann 3D optimal yang memanfaatkan kecerdasan buatan.

Bab ketujuh menguraikan teknik perspektif tiga dimensi, mekanika kamera, distorsi lensa, sampai proses pengambilan sampel citra digital 2D dan 3D yang erat kaitannya dengan keamanan pengolahan citra. Bab terakhir buku sekaligus penutup buku ini membahas konsep-konsep penting dalam pemrosesan citra, termasuk pentingnya padding, teknik operasional sistem Ann, serta metode transformasi berbasis geometri dan wavelet pada pengenalan pola dan analisis citra.

Penulis menyadari bahwa pembuatan buku ini tidak lepas dari bantuan dan dukungan berbagai pihak. Oleh karena itu, penulis mengucapkan terima kasih yang sebesar-besarnya kepada semua pihak yang telah turut berkontribusi secara langsung maupun tidak langsung. Semoga buku ini dapat menjadi sumber referensi yang bermanfaat bagi para akademisi,

praktisi, dan pembaca umum yang ingin mendalami teknologi kecerdasan buatan dalam keamanan siber.

Penulis juga membuka ruang bagi kritik dan saran yang membangun demi penyempurnaan dan perkembangan karya tulis ini di masa mendatang. Semoga buku ini dapat memberikan manfaat dan menjadi salah satu langkah dalam memperkuat pertahanan dunia maya melalui teknologi AI.

Semangat Dan Selamat Membaca …

Semarang, Agustus 2025
Penulis

Dr. Joseph Teguh Santoso, S.Kom., M.Kom.

"Melindungi dunia digital dengan kecerdasan yang cerdas, membangun masa depan yang aman dan terpercaya. Untuk para mahasiswa yang bersemangat menapaki dunia keamanan siber dan kecerdasan buatan. Semoga buku ajar ini menjadi panduan yang dapat memperluas wawasan, menumbuhkan rasa ingin tahu, dan mempermudah perjalanan akademik di bidang yang terus berkembang ini. Dengan harapan agar setiap pembaca dapat berkontribusi menciptakan ekosistem digital yang aman dan inovatif di masa depan. Keamanan digital bermula dari kecerdasan yang terus berkembang."

Dr. Joseph Teguh Santoso, S.Kom., M.Kom.

DAFTAR ISI

BAB 1
KECERDASAN BUATAN

Tidak diragukan lagi bahwa dunia saat ini jauh berbeda dari lima puluh atau bahkan tiga puluh tahun yang lalu, dari sudut pandang teknologi. Bayangkan saja ketika kita mendaratkan manusia pertama di bulan pada tahun 1969. Semua komputer yang digunakan di NASA adalah komputer mainframe, yang dikembangkan terutama oleh IBM dan perusahaan komputer terkait lainnya. Komputer-komputer ini sangat besar dan masif—bahkan, bisa memenuhi seluruh ruangan.

Bahkan komputer yang digunakan di roket Saturn V dan di Modul Komando dan Ekskursi Bulan juga merupakan tipe mainframe. Saat itu, bahkan hanya memiliki memori RAM 5 MB dalam komputer kecil adalah hal yang besar. Menurut standar saat ini, iPhone sangat jauh dari jenis teknologi komputasi ini, dan hanya dalam satu perangkat ini, kita mungkin memiliki daya komputasi yang cukup untuk mengirim roket Saturn V yang sama ke bulan dan kembali setidaknya 100 kali.

Tapi coba pikirkan, yang dibutuhkan saat itu hanya memori sebesar ini. Konsep-konsep Cloud, virtualisasi, dll. nyaris tidak pernah terdengar. Komputer yang dirancang saat itu, sebagai contoh, hanya memiliki satu tujuan spesifik: untuk memproses instruksi input dan output (juga dikenal sebagai "I/O") sehingga pesawat ruang angkasa dapat melakukan perjalanan yang aman ke bulan, mendarat di sana, dan kembali dengan selamat ke Bumi sekali lagi.

Karena kebutuhan yang terbatas ini (meskipun dianggap sangat besar pada saat itu), yang dibutuhkan hanyalah memori yang sangat kecil. Namun, dengan standar saat ini, mengingat semua aplikasi yang kita miliki saat ini, kita membutuhkan setidaknya 1.000 kali lipat lebih banyak hanya untuk menjalankan aplikasi berbasis Cloud yang paling sederhana sekalipun. Namun, saat itu pun, ada satu konsep yang bahkan belum pernah terdengar: Keamanan Siber.

Faktanya, bahkan istilah "Siber" pun belum pernah terdengar. Sebagian besar masalah keamanan saat itu berpusat pada keamanan fisik. Ambil contoh, NASA lagi. Kekhawatiran utamanya hanya mengizinkan karyawan yang berwenang dan sah masuk ke Mission Control. Siapa yang akan berpikir bahwa saat itu bahkan ada kemungkinan terkecil bahwa seorang Penyerang Siber secara harfiah dapat mengambil alih kendali komputer dan bahkan berpotensi mengarahkan roket Saturn V menjauh dari lintasan yang direncanakan.

Namun hari ini, mengingat semua kemajuan teknologi terkini, skenario kiamat ini kini menjadi kenyataan. Sebagai contoh, seorang Penyerang Siber dapat dengan sangat mudah mendapatkan akses ke gadget elektronik yang terkait dengan pesawat jet, mobil, atau bahkan kapal modern. Dengan mendapatkan akses ke sini dari pintu belakang tersembunyi, Penyerang Siber berpotensi mengambil alih kendali salah satu moda kapal ini dan secara harfiah membawanya ke tujuan yang tidak dimaksudkan.

Oleh karena itu, konsep Keamanan Siber kini menjadi sorotan utama, terutama mengingat krisis yang dialami dunia akibat Virus Corona, atau COVID-19. Namun, ketika kita memikirkan istilah ini, apa sebenarnya arti sebenarnya? Ketika kita memikirkannya, banyak pikiran dan gambaran yang muncul di benak. Misalnya, yang terbayang adalah server, workstation, dan perangkat nirkabel (termasuk notebook, tablet, dan ponsel pintar seperti perangkat Android dan iOS).

Selain itu, seseorang bahkan mungkin memikirkan Internet dan semua ratusan ribu mil kabel yang telah dipasang sehingga kita dapat mengakses situs web pilihan kita hanya dalam satu detik atau lebih. Namun perlu diingat bahwa ini hanya satu aspek dari Keamanan Siber. Aspek penting lainnya yang sering terlupakan adalah keamanan fisik yang terlibat. Seperti yang dijelaskan sebelumnya dengan contoh NASA kita, ini terutama melibatkan perlindungan premis fisik sebuah bisnis atau perusahaan. Ini termasuk melindungi premis eksterior dan interior. Sebagai contoh, ini tidak hanya bisa mendapatkan akses utama ke premis itu sendiri, tetapi juga bagian interiornya, seperti ruang server dan tempat di mana informasi dan data rahasia perusahaan disimpan. Sangat penting untuk diingat bahwa semua ini, baik fisik maupun digital, berada dalam risiko besar untuk diserang.

Tidak ada satu individu atau badan usaha pun yang bebas dari hal ini, semua pihak berisiko terkena Serangan Siber. Kuncinya adalah bagaimana memitigasi risiko tersebut agar tidak menyebar lebih jauh setelah Anda mengetahui bahwa Anda memang telah menjadi korban. Jadi, setelah kita membahas apa sebenarnya ruang lingkup Keamanan Siber, bagaimana definisi spesifiknya?

Dapat didefinisikan sebagai berikut: *Juga disebut sebagai keamanan informasi, keamanan siber mengacu pada praktik memastikan integritas, kerahasiaan, dan ketersediaan (ICA) informasi. Keamanan siber terdiri dari serangkaian alat, pendekatan manajemen risiko, teknologi, pelatihan, dan praktik terbaik yang terus berkembang, yang dirancang untuk melindungi jaringan, perangkat, program, dan data dari serangan atau akses tidak sah.*

Meskipun ini merupakan definisi yang sangat luas, dalam upaya untuk mempersempitnya lebih lanjut, Keamanan Siber melibatkan komponen-komponen berikut:

- Keamanan jaringan (melindungi seluruh jaringan dan subnet bisnis);
- Keamanan aplikasi (melindungi aplikasi penting, terutama yang berbasis web);
- Keamanan titik akhir (melindungi titik asal dan tujuan koneksi jaringan);
- Keamanan data (melindungi kumpulan data penting, terutama yang berkaitan dengan Informasi Identitas Pribadi (PII))
- Manajemen identitas (memastikan bahwa hanya individu yang sah yang dapat memperoleh akses logis dan/atau fisik);
- Keamanan basis data dan infrastruktur (melindungi server yang menyimpan PII);
- Keamanan cloud (melindungi komponen Infrastruktur sebagai Layanan (IaaS), Perangkat Lunak sebagai Layanan (SaaS), dan Platform sebagai Layanan (PaaS) dari platform berbasis cloud);
- Keamanan seluler (melindungi semua aspek perangkat nirkabel dan ponsel pintar, baik dari segi perangkat keras, sistem operasi, maupun seluler);

- Pemulihan bencana/perencanaan kesinambungan bisnis (menyusun rencana yang tepat agar bisnis dapat meningkatkan aplikasi penting ke tingkat operasional dan agar mereka dapat terus melanjutkannya setelah terjadi pelanggaran keamanan);
- Edukasi pengguna akhir (menjaga karyawan dan individu tetap terlatih tentang cara mengurangi risiko menjadi korban).

Setelah kita membahas pentingnya, definisi, dan komponen Keamanan Siber, kini penting untuk melihat evolusinya, yang diilustrasikan di bagian selanjutnya.

1.1 EVOLUSI KRONOLOGIS KEAMANAN SIBER

Sebagaimana teknologi berevolusi dan berkembang pesat, dunia Keamanan Siber pun demikian. Sebagaimana telah disebutkan, sekitar 50 tahun, di puncak program luar angkasa Apollo, istilah "Siber" mungkin hampir tidak terpikirkan. Namun di masa kini, dan terutama di dekade ini, istilah tersebut kini hampir menjadi bagian dari kehidupan kita sehari-hari.

Di bagian ini, kami akan memberikan gambaran singkat tentang bagaimana Keamanan Siber sebenarnya berevolusi.

Cacing Morris (1988)

- Dibuat oleh Robert Morris, seorang mahasiswa pascasarjana di Cornell.
- Menjatuhkan sekitar 10% dari 70.000 komputer yang terhubung ke Internet secara global.
- Menyebabkan kerugian setidaknya $96 juta secara total.
- Sebenarnya menjadi prototipe untuk serangan Distributed Denial of Service (DDoS) yang kita lihat sekarang.

Virus Melissa (Maret 1999)

- Dinamai dari seorang penari di Florida, dan menginfeksi file .DOC yang dikirimkan ke buku alamat di Microsoft Outlook.
- Virus ini menyebabkan Microsoft, Lockheed Martin, dan Intel menutup seluruh operasi mereka untuk periode waktu tertentu.
- Menyebabkan kerugian sebesar $80 juta, dan menginfeksi lebih dari 1.000.000 komputer di seluruh dunia.
- Pencipta virus ini, David L. Smith, dijatuhi hukuman penjara 20 bulan.

Departemen Pertahanan Amerika Serikat (DoD) (Agustus 1999)

- Jonathan James, seorang hacker berusia 15 tahun, meretas infrastruktur jaringan di Defense Threat Reduction Agency.
- Ia adalah remaja pertama yang dihukum karena kejahatan siber besar.
- NASA harus menutup seluruh basis operasi mereka selama tiga minggu.
- Tidak hanya mencuri kata sandi, tetapi juga mencuri aplikasi perangkat lunak senilai sekitar $1,7 juta yang mendukung Stasiun Luar Angkasa Internasional.

Mafiaboy (Februari 2002)

- Seorang hacker remaja, Michael Calce (alias 'Mafiaboy'), meluncurkan varian ancaman khusus yang dikenal sebagai 'Project Rivolta'.
- Ini adalah serangkaian serangan Denial of Service (DoS) yang menjatuhkan situs web perusahaan besar Amerika Serikat.
- Contohnya termasuk Yahoo, eBay, CNN, E-Trade, dan server berbasis Amazon.
- Hal ini mendorong Gedung Putih untuk mengadakan KTT Keamanan Siber pertama mereka.
- Kerugian finansial melebihi $1,2 miliar.

Target (November 2013)

- Dianggap sebagai salah satu serangan siber ritel terbesar dalam sejarah, dan terjadi tepat selama musim liburan 2013.
- Karena serangan ini, laba bersih Target turun hingga 46%.
- Lebih dari 40 juta nomor kartu kredit dicuri.
- Malware dipasang di terminal Point of Sale (PoS) di semua toko Target.
- Data ini dijual di Dark Web dengan keuntungan besar.
- Serangan ini menjadi model bagi serangan siber ritel berikutnya.

Sony Pictures (November 2014)

- Nomor Jaminan Sosial dan kartu kredit bocor ke publik.
- Informasi dan data penggajian rahasia juga dibocorkan.

- Serangan siber ini mendorong Co-Chair Sony Pictures, Amy Pascal, untuk mengundurkan diri dari posisinya.

Anthem (Januari 2015)

- Dianggap sebagai serangan siber terbesar yang menimpa organisasi kesehatan besar.
- Informasi Identitas Pribadi (PII) dari lebih dari 80.000.000 anggota dicuri termasuk Nomor Jaminan Sosial, alamat email, dan informasi pekerjaan.

Ransomworm Pertama (2017)

- WannaCry dianggap sebagai ransomware varian ancaman pertama, dan menargetkan komputer yang menjalankan sistem operasi Windows.
- Satu-satunya cara korban bisa menggunakan kembali komputernya adalah dengan membayar tebusan kepada peretas dalam bentuk mata uang virtual, seperti Bitcoin.
- Dalam satu hari saja, varian ancaman WannaCry menginfeksi lebih dari 230.000 komputer di lebih dari 50 negara.
- Versi terbaru dari ancaman ini adalah 'NotPetya'. Serangan ini menginfeksi lebih dari 12.500 komputer secara global. Industri yang terdampak termasuk perusahaan energi, bank, dan lembaga pemerintah.

Serangan Siber Kartu Kredit Terbesar (2017)

- Biro kartu kredit, dikenal sebagai Equifax, gagal total dalam menginstal patch perangkat lunak terbaru dan pembaruan pada server Apache Struts mereka.
- Para peretas berhasil mendapatkan akses ke lebih dari 210.000 kartu kredit konsumen, yang berdampak pada lebih dari 143 juta warga Amerika.

Facebook, MyHeritage, Marriott Hotels, dan British Airways (2018)

- Facebook terkena serangan siber besar dengan firma analitik Cambridge Analytica. Informasi Identitas Pribadi (PII) yang dicuri berdampak pada lebih dari 87 juta pengguna.
- Dengan MyHeritage, lebih dari 92 juta pengguna terdampak. Untungnya, tidak ada informasi perbankan atau kartu kredit, tes DNA, atau kata sandi yang dicuri.
- Dengan Marriott Hotels, lebih dari 500 juta pengguna terdampak. Walaupun kebocoran ini terungkap pada 2018, malware sebenarnya telah ditanam sejak 2014. Marriott dijatuhi denda sebesar $123 juta.
- Dengan British Airways, lebih dari 500.000 transaksi kartu kredit terdampak. Informasi Identitas Pribadi (PII) yang dicuri termasuk nama, alamat email, nomor telepon, alamat, dan nomor kartu kredit. Perusahaan menghadapi denda $230 juta sebagaimana diberlakukan oleh GDPR, setara dengan 1,5% dari total pendapatan mereka.

Sektor Kesehatan Singapura (2019):

- Otoritas Ilmu Kesehatan (HSA) Singapura mengalihdayakan beberapa fungsinya kepada vendor pihak ketiga yang dikenal sebagai Secur Solutions Group. Informasi Identifikasi Pribadi (PII) dari 808.000 donor terungkap secara online, dan item-item yang disusupi termasuk nama, nomor kartu identitas, jenis kelamin, tanggal tiga donasi terakhir, dan dalam beberapa kasus, golongan darah, tinggi, dan berat badan para donor.
- Unit Kesehatan Masyarakat Nasional dari Kementerian Kesehatan Singapura terkena dampak ketika status HIV dari 14.200 orang terungkap secara online.

Jadi, seperti yang Anda lihat, ini adalah kronologi dari semua peristiwa besar Keamanan Siber yang telah membawa kita ke titik saat ini. Bahkan di dunia Keamanan Siber, terdapat pula kemajuan teknologi besar yang telah dicapai untuk menggagalkan serangan siber dan mengikuti dinamika Lanskap Ancaman Siber yang terus berubah.

Salah satu area dalam hal ini dikenal sebagai "Kecerdasan Buatan", atau disingkat "AI". Hal ini akan diulas lebih lanjut di bagian selanjutnya, dan merupakan fokus utama dari keseluruhan buku ini.

1.2 PENGANTAR KECERDASAN BUATAN DALAM KEAMANAN SIBER

Konsep Kecerdasan Buatan bukanlah hal baru; melainkan sudah ada sejak lama—bahkan hingga tahun 1960-an. Meskipun terdapat beberapa aplikasi yang dikembangkan pada saat itu, namun belum benar-benar mencapai momentum sebesar sekarang hingga saat ini, terutama yang berkaitan dengan Keamanan Siber. Faktanya, minat terhadap AI bahkan belum

muncul di industri ini hingga akhir 2019. Saat ini, bersama dengan jargon teknologi lainnya yang beredar, AI menjadi salah satu kata kunci paling populer saat ini.

Namun, bukan hanya Keamanan Siber saja AI yang mendapatkan semua perhatian. Ada banyak bidang lain juga, terutama yang berkaitan dengan manufaktur dan rantai pasok, serta industri logistik. Anda mungkin bertanya-tanya, apa istimewanya Kecerdasan Buatan? Kuncinya adalah bidang ini dapat membantu membawa otomatisasi tugas ke tingkat yang jauh lebih optimal dan efisien daripada yang pernah dapat dicapai manusia.

Misalnya, dalam industri-industri yang disebutkan di atas (kecuali Keamanan Siber), berbagai proses robotik dapat dikembangkan dari perangkat AI untuk mempercepat proses tertentu. Ini termasuk melakukan tugas-tugas berulang di lini produksi mobil, atau bahkan di gudang-gudang industri rantai pasok dan logistik. Bidang ini dikenal sebagai "Otomatisasi Proses Robotik" atau disingkat "RPA", dan akan dibahas lebih detail nanti di buku ini.

Namun, terkait Keamanan Siber, salah satu area utama di mana Kecerdasan Buatan memainkan peran kunci adalah otomatisasi tugas, seperti yang baru saja dibahas. Misalnya, Pengujian Penetrasi dan Perburuan Ancaman merupakan tugas yang sangat memakan waktu, melelahkan, dan menguras pikiran. Ada banyak langkah kecil dalam kedua proses ini yang harus dilakukan, dan sekali lagi, banyak di antaranya yang repetitif. Di sinilah peran perangkat AI.

Hasilnya, anggota tim di sisi Pengujian Penetrasi dan Perburuan Ancaman dapat lebih fokus pada tugas yang jauh lebih penting, termasuk menemukan celah dan kelemahan tersembunyi maupun yang tidak tersembunyi dalam Infrastruktur TI dan Jaringan klien mereka, serta menyediakan tindakan yang tepat untuk menutupi celah dan kelemahan ini.

Area hebat lainnya dalam Keamanan Siber di mana perangkat Kecerdasan Buatan digunakan adalah penyaringan positif palsu. Misalnya, tim keamanan TI di berbagai bisnis dan perusahaan, baik besar maupun kecil, dibanjiri peringatan dan imbauan akibat banyaknya alat keamanan yang mereka gunakan, terutama yang berkaitan dengan Firewall, Perangkat Intrusi Jaringan, dan Router. Saat ini, mereka harus menyaring setiap perangkat secara manual agar dapat diprioritaskan dengan tepat.

Namun karena waktu yang dibutuhkan untuk melakukan hal ini, banyak peringatan dan imbauan nyata yang muncul seringkali tidak disadari, sehingga meningkatkan Risiko Siber entitas bisnis tersebut setidaknya 1.000 kali lipat. Namun, dengan menggunakan perangkat yang berkaitan dengan Kecerdasan Buatan, semua positif palsu ini tersaring, sehingga hanya menyisakan yang asli dan sah yang perlu diperiksa dan diprioritaskan. Hasilnya, tim keamanan TI dapat bereaksi terhadap ancaman-ancaman khusus ini dengan jauh lebih cepat, dan yang terpenting, mempertahankan pola pikir proaktif untuk menangkal varian ancaman ini.

Perlu dicatat juga bahwa banyak bisnis dan perusahaan kini mulai menyadari bahwa memiliki terlalu banyak perangkat keamanan untuk memperkuat lini pertahanan mereka masing-masing bukanlah hal yang baik—bahkan, hal itu hanya meningkatkan permukaan serangan bagi penyerang siber. Oleh karena itu, kini banyak entitas bisnis ini mulai menyadari pentingnya penerapan berbagai perangkat analisis risiko untuk melihat di mana semua teknologi keamanan ini dapat ditempatkan secara strategis.

Jadi, alih-alih mengadopsi pola pikir bahwa lebih banyak lebih baik, kini bergeser bahwa kualitas penerapan jauh lebih krusial dan penting. Jadi, alih-alih menerapkan sepuluh Firewall, jauh lebih strategis untuk menerapkan mungkin hanya tiga di tempat yang paling membutuhkannya. Selain itu, dengan mengadopsi pola pikir seperti ini, bisnis atau korporasi akan mencapai Return On Investment (ROI) yang jauh lebih besar, yang berarti CIO dan/atau CISO akan berada di posisi yang jauh lebih baik untuk mendapatkan lebih banyak dana untuk anggaran keamanan mereka.

Namun, Anda mungkin bertanya-tanya, apa sebenarnya Kecerdasan Buatan itu? Definisi formalnya ada di sini:

Kecerdasan buatan (AI) memungkinkan mesin untuk belajar dari pengalaman, menyesuaikan diri dengan masukan baru, dan melakukan tugas-tugas layaknya manusia. Sebagian besar contoh AI yang Anda dengar saat ini—mulai dari komputer untuk bermain catur hingga mobil tanpa pengemudi—sangat bergantung pada pembelajaran mendalam dan pemrosesan bahasa alami. Dengan menggunakan teknologi ini, komputer dapat dilatih untuk menyelesaikan tugas-tugas tertentu dengan memproses data dalam jumlah besar dan mengenali pola dalam data tersebut.

Sebagaimana dapat dilihat dari definisi di atas, tujuan utama Kecerdasan Buatan adalah kemampuan untuk belajar dan memproyeksikan masa depan dengan belajar dari perilaku masa lalu. Dalam hal ini, perilaku masa lalu biasanya berarti memanfaatkan kumpulan data besar yang muncul dan berasal dari berbagai umpan data yang dimasukkan ke dalam berbagai teknologi AI yang sedang digunakan, mempelajari tren tersebut, dan memiliki kemampuan untuk menjalankan tugas yang ada serta melihat ke masa depan.

Dalam hal ini, manfaat besar lain yang dibawa Kecerdasan Buatan bagi Keamanan Siber adalah kemampuannya untuk memprediksi masa depan, dan menilai seperti apa varian ancaman potensial yang lebih baru. Kita akan membahas pentingnya data bagi Kecerdasan Buatan nanti di bab ini. Namun, pada titik ini, sangat penting untuk diingat bahwa Kecerdasan Buatan hanyalah bidang utama, dan ada banyak sub-bidang lain yang berada tepat di bawahnya; yang paling umum adalah sebagai berikut:

- Pembelajaran Mesin;
- Jaringan Saraf Tiruan;
- Visi Komputer.

Definisi formal untuk masing-masing bidang di atas disediakan di bagian selanjutnya.

1.3 SUB-BIDANG KECERDASAN BUATAN

Pembelajaran Mesin

Sub-bidang pertama yang akan kita bahas secara singkat adalah apa yang dikenal sebagai "Pembelajaran Mesin", atau disingkat "ML". Definisi spesifiknya adalah sebagai berikut:

Algoritma pembelajaran mesin menggunakan statistik untuk menemukan pola dalam data dalam jumlah besar. Dan data, di sini, mencakup banyak hal—angka, kata, gambar, klik,

dan sebagainya. Jika dapat disimpan secara digital, data tersebut dapat dimasukkan ke dalam algoritma pembelajaran mesin.

Pembelajaran mesin adalah proses yang menggerakkan banyak layanan yang kita gunakan saat ini—sistem rekomendasi seperti yang ada di Netflix, YouTube, dan Spotify; mesin pencari seperti Google dan Baidu; umpan media sosial seperti Facebook dan Twitter; asisten suara seperti Siri dan Alexa. Daftarnya masih panjang.

Sub-bidang Pembelajaran Mesin sebenarnya sangat luas, beragam, dan bahkan cukup kompleks. Namun, secara umum, sebagaimana dijelaskan dalam definisi di atas, metode ini menggunakan lebih banyak teknik statistik daripada matematika untuk menambang dan menyisir sejumlah besar dataset guna menemukan tren yang tidak tersembunyi. Data ini kemudian dapat dimasukkan ke dalam perangkat Kecerdasan Buatan, misalnya, untuk memprediksi Lanskap Ancaman Siber di masa mendatang. Namun, metode ini juga memiliki banyak aplikasi lain, sebagaimana dicontohkan pada bagian kedua definisi ini.

Jaringan Saraf Tiruan

Sub-bidang kedua yang akan dibahas selanjutnya adalah Jaringan Saraf Tiruan (juga dikenal sebagai Jaringan Saraf Tiruan). Definisi spesifiknya adalah sebagai berikut:

Jaringan saraf tiruan adalah serangkaian algoritma, yang dimodelkan secara longgar berdasarkan otak manusia, yang dirancang untuk mengenali pola. Jaringan ini menginterpretasikan data sensorik melalui semacam persepsi mesin, pelabelan, atau pengelompokan masukan mentah. Pola yang dikenali bersifat numerik, termuat dalam vektor, yang menjadi tempat semua data dunia nyata, baik berupa gambar, suara, teks, maupun deret waktu, harus diterjemahkan.

Jaringan saraf tiruan membantu kita mengelompokkan dan mengklasifikasikan. Anda dapat menganggapnya sebagai lapisan pengelompokan dan klasifikasi di atas data yang Anda simpan dan kelola. Jaringan saraf tiruan membantu mengelompokkan data yang tidak berlabel berdasarkan kesamaan di antara masukan contoh, dan mengklasifikasikan data ketika memiliki set data berlabel untuk pelatihan. (Jaringan saraf tiruan juga dapat mengekstrak fitur yang diumpankan ke algoritma lain untuk pengelompokan dan klasifikasi; sehingga Anda dapat menganggap jaringan saraf tiruan dalam sebagai komponen aplikasi pembelajaran mesin yang lebih besar yang melibatkan algoritma untuk pembelajaran penguatan, klasifikasi, dan regresi).

Dengan cara yang mirip dengan Pembelajaran Mesin, Jaringan Saraf Tiruan juga dirancang untuk mengamati kumpulan data besar guna mengenali pola tersembunyi dan tidak tersembunyi. Namun, perbedaan utamanya adalah Jaringan Saraf Tiruan dirancang untuk mencoba mereplikasi proses berpikir otak manusia, dengan memeriksa aktivitas neuron otak secara saksama.

Otak manusia terdiri dari ratusan juta neuron, dan dihipotesiskan bahwa neuron merupakan katalisator rasional di balik proses pengambilan keputusan yang terjadi di dalam otak. Perbedaan utama lainnya adalah Jaringan Saraf Tiruan juga dapat digunakan untuk mengatur, menyaring, dan menyajikan kumpulan data yang paling relevan. Kembali ke contoh sebelumnya tentang penyaringan positif palsu, ini adalah contoh utama di mana Jaringan Saraf Tiruan digunakan. Konsep neuron akan dibahas lebih rinci nanti dalam buku ini.

Visi Komputer

Sub-bidang ketiga yang akan dibahas adalah Visi Komputer. Definisi spesifiknya adalah sebagai berikut:

Visi komputer adalah proses menggunakan mesin untuk memahami dan menganalisis citra (baik foto maupun video). Meskipun jenis algoritma ini telah ada dalam berbagai bentuk sejak tahun 1960-an, kemajuan terbaru dalam Pembelajaran Mesin, serta kemajuan pesat dalam penyimpanan data, kemampuan komputasi, dan perangkat input berkualitas tinggi yang murah telah mendorong peningkatan besar dalam seberapa baik perangkat lunak kita dapat mengeksplorasi jenis konten ini.

Visi komputer adalah nama induk yang luas untuk semua komputasi yang melibatkan konten visual—artinya gambar, video, ikon, dan apa pun yang melibatkan piksel. Namun, dalam gagasan induk ini, terdapat beberapa tugas spesifik yang merupakan blok pembangun inti:

Dalam klasifikasi objek, Anda melatih model pada kumpulan data objek tertentu, dan model tersebut mengklasifikasikan objek baru sebagai milik satu atau lebih kategori pelatihan Anda. Untuk identifikasi objek, model Anda akan mengenali contoh spesifik suatu objek—misalnya, mengurai dua wajah dalam sebuah gambar dan menandai satu sebagai Tom Cruise dan satu lagi sebagai Katie Holmes.

Sebagaimana dapat dilihat dari definisi di atas, Visi Komputer terutama digunakan untuk memeriksa jenis dan macam dataset visual, menganalisisnya, dan memasukkannya ke dalam perangkat Kecerdasan Buatan. Terkait dengan Keamanan Siber, hal ini paling relevan dalam hal melindungi aset fisik suatu bisnis atau perusahaan, bukan aset digital.

Misalnya, kamera CCTV digunakan untuk membantu mengonfirmasi identitas individu (seperti karyawan) yang mencoba mendapatkan akses masuk utama atau akses sekunder di dalam bisnis atau perusahaan. Pengenalan Wajah sangat sering digunakan di sini, untuk melacak dan menyaring segala jenis perilaku jahat atau anomali.

Ini sering dianggap sebagai lapis kedua setelah kamera CCTV, tetapi selain itu, perangkat Visi Komputer juga dapat digunakan dengan teknologi Pengenalan Wajah untuk menyediakan sampel yang jauh lebih kuat untuk dikumpulkan, dan untuk dapat bereaksi terhadap pelanggaran keamanan dengan cara yang jauh lebih cepat dan efisien.

Sebagaimana telah disebutkan, dan bahkan dapat dilihat dari judul bab pertama ini, seluruh premis buku ini dibangun di seputar Kecerdasan Buatan. Memang, ada banyak buku di luar sana yang berfokus pada pokok bahasan ini, tetapi banyak di antaranya bersifat sangat teoretis, dan mungkin tidak menawarkan nilai sebanyak itu bagi bisnis dan korporasi. Sebaliknya, buku-buku tersebut lebih ditujukan untuk pasar akademis dan pemerintahan, seperti ilmuwan riset, profesor universitas, kontraktor pertahanan, dan sebagainya. Tidak banyak dari mereka yang benar-benar membahas sisi aplikasi Kecerdasan Buatan. Inilah yang membedakan buku ini, secara harfiah, dari buku-buku lain yang ada di luar sana.

Misalnya, terdapat komponen teoretis di setiap bab. Hal ini diperlukan karena untuk memahami sisi aplikasi Kecerdasan Buatan, seseorang juga perlu memiliki latar belakang yang kuat dalam teorinya. Hal ini sebenarnya mencakup sekitar paruh pertama setiap bab. Namun,

paruh kedua setiap bab akan dikhususkan untuk sisi praktis Kecerdasan Buatan—yaitu aplikasinya.

Keunikan buku ini adalah aplikasi-aplikasi yang dibahas dan diulas adalah aplikasi-aplikasi yang telah atau sedang dalam proses penerapan di berbagai jenis dan tipe aplikasi Keamanan Siber. Aplikasi-aplikasi ini ditulis oleh para Pakar Subjek (SME) sendiri. Sepengetahuan kami, belum ada buku lain yang membahas hal ini. Saat Anda membaca bab-bab ini, Anda akan merasa sangat terhibur dengan membaca tentang aplikasi-aplikasi ini.

Pada akhirnya, bab terakhir dikhususkan untuk praktik terbaik Kecerdasan Buatan. Dengan kata lain, kami tidak hanya membahas sudut pandang teoretis dan aplikasi, tetapi juga menawarkan panduan Praktik Terbaik (atau, jika Anda mau, daftar periksa) dalam pembuatan dan penerapan aplikasi Kecerdasan Buatan.

Oleh karena itu, buku ini dapat melayani dua jenis audiens: 1) sektor akademis dan pemerintahan sebagaimana telah dibahas sebelumnya; dan, 2) CIO, CISO, Manajer Keamanan TI, dan bahkan Manajer Proyek yang ingin menerapkan aplikasi Kecerdasan Buatan.

Untuk memulai komponen teoretis dari bab pertama ini, pertama-tama kami akan membahas Kecerdasan Buatan dan bagaimana ia menjadi komponen penting Keamanan Siber saat ini. Kedua, dilanjutkan dengan membahas pentingnya data—bagaimanapun juga, seperti yang telah diulas sebelumnya, data adalah bahan bakar yang secara harfiah menggerakkan mesin aplikasi Kecerdasan Buatan.

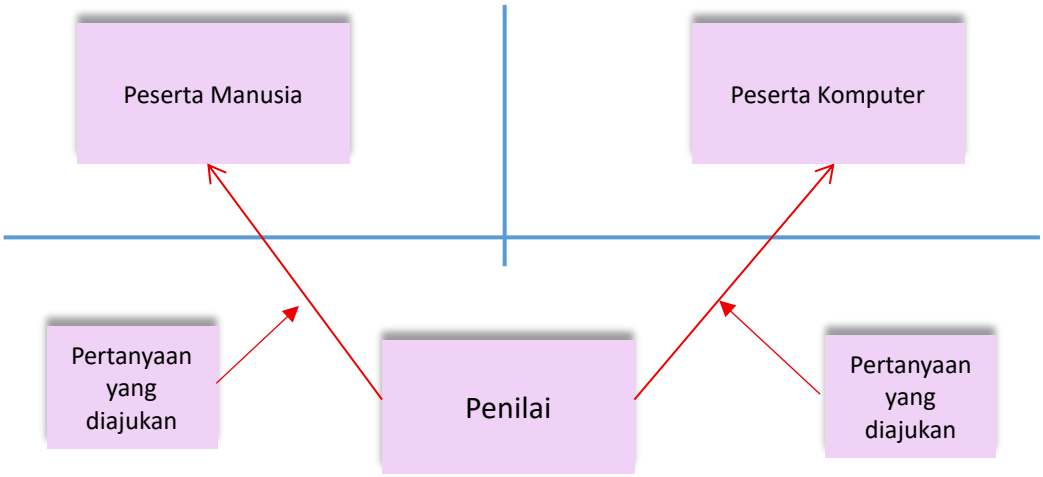
Sejarah Singkat Kecerdasan Buatan

Sebagai permulaan, mungkin tokoh pertama yang terkenal di bidang Kecerdasan Buatan adalah Alan Turing. Ia dianggap sebagai pelopor dalam ilmu komputer, dan bahkan sering disebut sebagai "Bapak Kecerdasan Buatan". Pada tahun 1936, ia menulis sebuah makalah ilmiah penting berjudul "On Computable Numbers". Dalam karya terkenal ini, ia menjabarkan konsep tentang apa itu komputer dan apa tujuan utamanya. Penting untuk diingat bahwa komputer hampir tidak ada pada masa itu, dan faktanya, "jenis" komputer pertama baru muncul jauh di dekade berikutnya.

Ide dasar tentang komputer menurutnya didasarkan pada premis bahwa komputer harus cerdas dalam suatu cara atau corak tertentu. Namun pada saat itu, sangat sulit untuk menemukan ukuran aktual tentang apa sebenarnya "kecerdasan" itu. Oleh karena itu, ia mencetuskan konsep yang akhirnya dikenal sebagai "Tes Turing".

Dalam skenario ini, terdapat sebuah permainan yang melibatkan tiga pemain. Salah satu peserta adalah manusia, dan yang lainnya adalah komputer. Peserta ketiga adalah moderator, atau evaluator. Dalam skenario ini, moderator akan mengajukan serangkaian pertanyaan terbuka kepada keduanya, untuk menentukan siapa di antara kedua peserta tersebut yang sebenarnya manusia. Jika penentuan tidak dapat dilakukan dengan mengajukan pertanyaan terbuka ini, maka komputer akan dianggap sebagai entitas "cerdas".

Uji Turing diilustrasikan di bawah ini:



Dalam model ini, tidaklah perlu bahwa komputer harus benar-benar mengetahui sesuatu yang spesifik, memiliki sejumlah besar informasi dan data, atau bahkan benar dalam jawabannya terhadap pertanyaan terbuka. Namun sebaliknya, harus ada indikasi kuat bahwa komputer dapat, dengan satu atau lain cara, berkomunikasi dengan Evaluator dengan sendirinya, tanpa melibatkan campur tangan manusia.

Percaya atau tidak, Tes Turing telah teruji oleh waktu dan tetap sulit dipecahkan, bahkan di dekade baru abad ke-21 ini. Sebagai contoh, telah banyak kontes dan kompetisi untuk melihat apakah komputer dapat memenuhi Tes Turing, dan beberapa yang paling penting adalah "Hadiah Loebner" dan "Kompetisi Tes Turing".

Titik balik terjadi dalam sebuah kompetisi yang diadakan pada Mei 2018 di Konferensi I/O yang diselenggarakan oleh Google. CEO Google saat itu, Sundar Pichai, memberikan demonstrasi langsung salah satu aplikasi terbaru mereka, yang dikenal sebagai "Asisten Google". Aplikasi ini digunakan untuk menelepon langsung ke penata rambut setempat guna membuat dan menjadwalkan janji temu. Seseorang memang mengangkat telepon di saluran lain, tetapi skenario ini gagal dalam Uji Turing.

Mengapa? Karena pertanyaan yang diajukan bersifat tertutup, bukan pertanyaan terbuka.

Terobosan besar berikutnya setelah Tes Turing adalah terciptanya dan pengembangan makalah ilmiah berjudul "Minds, Brains, and Programs". Makalah ini ditulis oleh ilmuwan bernama John Searle dan diterbitkan pada tahun 1980. Dalam makalah penelitian ini, ia merumuskan model lain yang sangat mirip dengan Tes Turing, yang kemudian dikenal sebagai "Argumen Ruang Bahasa Mandarin".

Berikut premis dasarnya: Misalkan ada seseorang bernama "Tracey". Ia tidak tahu atau bahkan tidak mengerti bahasa Mandarin, tetapi ia memiliki dua buku panduan berisi aturan langkah demi langkah tentang cara menafsirkan dan berkomunikasi dalam bahasa Mandarin. Tepat di luar ruangan ini terdapat seorang individu lain bernama "Suzanne". Suzanne mengerti bahasa Mandarin, dan membantu Tracey dengan membantunya menguraikan banyak karakter.

Setelah beberapa waktu, Suzanne akan mendapatkan terjemahan yang cukup akurat dari Tracey. Dengan demikian, masuk akal untuk berpikir bahwa Suzanne berasumsi dengan yakin bahwa Tracey dapat memahami, hingga tingkat yang berbeda-beda, bahasa Mandarin.

Inti dari argumen ini adalah jika Tracey tidak dapat memahami bahasa Mandarin dengan menerapkan aturan yang tepat untuk memahami bahasa Mandarin terlepas dari semua alat bantu yang dimilikinya (dua buku panduan dan Suzanne, tepat di luar ruangan), maka komputer tidak dapat belajar dengan metodologi ini karena tidak ada satu komputer pun yang memiliki pengetahuan lebih banyak daripada yang dimiliki pria atau wanita lain.

Makalah yang ditulis John Searle juga menguraikan dua jenis Kecerdasan Buatan yang berpotensi ada:

1) AI Kuat:

   Ini terjadi ketika komputer benar-benar memahami dan sepenuhnya menyadari apa yang terjadi di sekitarnya. Ini bahkan dapat melibatkan komputer yang memiliki semacam emosi dan kreativitas yang melekat padanya. Bidang Kecerdasan Buatan ini juga secara teknis dikenal sebagai "Kecerdasan Umum Buatan", atau disingkat "AGI".

2) AI Lemah:

   Ini adalah bentuk Kecerdasan Buatan yang dianggap tidak begitu kuat secara alami, dan diberikan fokus atau serangkaian tugas yang sangat terbatas untuk dikerjakan. Contoh utama dari hal ini termasuk Asisten Pribadi Virtual (VPA) Siri dan Alexa (yang masing-masing dimiliki oleh Apple dan Amazon).

Kemunculan Uji Turing juga mendorong pengembangan beberapa model penting lainnya, yang meliputi:

1) Uji Kurzweil-Kapor:

   Model ini diciptakan dan dikembangkan oleh Ray Kurzweil dan Mitch Kapor. Dalam uji ini, komputer diharuskan melakukan semacam percakapan dengan tiga juri. Jika dua dari mereka menganggap percakapan tersebut "cerdas", maka komputer tersebut juga dianggap cerdas. Namun, permutasi pasti tentang apa yang sebenarnya mendefinisikan "percakapan cerdas" tidak diberikan.

2) Uji Kopi:

   Model ini dikembangkan oleh pendiri Apple, Steve Wozniak, dan sebenarnya cukup sederhana: Robot harus bisa masuk ke dalam rumah, menemukan letak dapur, dan membuat/menyeduh secangkir kopi.

Terobosan besar berikutnya dalam Kecerdasan Buatan adalah sebuah makalah ilmiah berjudul "Kalkulus Logika Ide-Ide yang Imanen dalam Aktivitas Saraf". Makalah ini ditulis bersama oleh Warren McCulloch dan Walter Pitts pada tahun 1943. Premis utama makalah ini adalah bahwa deduksi logis dapat menjelaskan kekuatan otak manusia. Makalah ini kemudian diterbitkan dalam *Bulletin of Mathematical Biophysics*.

Dalam buku ini, McCulloch dan Pitts berpendapat bahwa fungsi inti otak manusia, khususnya neuron dan aktivitas sinaptik yang terjadi, dapat dijelaskan sepenuhnya oleh operator logika matematika (misalnya, Dan, Tidak, dan lainya.).

Dalam upaya untuk membangun hal ini, Norbert Wiener menciptakan dan menerbitkan sebuah buku ilmiah berjudul *Sibernetika: Atau Kontrol dan Komunikasi dalam Hewan dan Mesin.* Buku khusus ini membahas topik-topik seperti Mekanika Newton, Statistik, Termodinamika, dll. Buku ini memperkenalkan jenis teori baru yang disebut "Teori Chaos". Ia juga menyamakan otak manusia dengan otak komputer, yaitu otak seharusnya mampu bermain catur, dan seharusnya mampu belajar pada tingkat yang lebih tinggi seiring dengan semakin banyaknya permainan yang dimainkan.

Periode penting berikutnya bagi Kecerdasan Buatan dikenal sebagai "Kisah Asal Usul", dan akan diulas lebih detail di sub-bagian berikutnya.

Kisah Asal Usul

Langkah besar berikutnya dalam dunia Kecerdasan Buatan datang ketika seorang individu bernama John McCarthy mengorganisir dan menyelenggarakan program penelitian selama sepuluh minggu di Universitas Dartmouth. Program tersebut berjudul "Studi Kecerdasan Buatan", dan ini adalah pertama kalinya istilah ini digunakan. Sifat pasti dari proyek ini adalah sebagai berikut: *Studi ini akan dilanjutkan berdasarkan dugaan bahwa setiap aspek pembelajaran atau fitur kecerdasan lainnya pada prinsipnya dapat dijelaskan secara tepat sehingga mesin dapat disimulasikan. Dengan demikian, akan dilakukan upaya untuk menemukan cara membuat mesin menggunakan bahasa, membentuk abstraksi dan konsep, memecahkan berbagai masalah yang sekarang hanya diperuntukkan bagi manusia, dan meningkatkan kemampuan diri mereka sendiri. Kami berpendapat bahwa kemajuan signifikan dapat dicapai dalam satu atau lebih permasalahan ini jika sekelompok ilmuwan terpilih bekerja sama selama musim panas.*

Selama retret khusus ini, sebuah program komputer bernama "Logic Theorist" didemonstrasikan, yang sebenarnya dikembangkan di RAND Corporation. Fokusnya adalah memecahkan teorema matematika kompleks dari publikasi yang dikenal sebagai "Principia Mathematica". Untuk menciptakan bahasa pemrograman ini, komputer mainframe IBM 701 digunakan, yang utamanya menggunakan bahasa mesin untuk memproses informasi dan data.

Namun, untuk lebih mengoptimalkan kecepatan "Logic Theorist", sebuah bahasa pemrosesan baru digunakan, yang kemudian dikenal sebagai "Information Processing Language" (Bahasa Pemrosesan Informasi), atau disingkat "IPL". Namun, mainframe IBM 701 tidak memiliki memori atau daya pemrosesan yang cukup untuk IPL, sehingga hal ini mendorong terciptanya pengembangan lain: Alokasi Memori Dinamis. Hasilnya, “Logic Theorist” dianggap sebagai bahasa pemrograman Kecerdasan Buatan pertama yang pernah diciptakan.

Setelah itu, John McCarthy melanjutkan dengan menciptakan aspek-aspek lain untuk Kecerdasan Buatan pada tahun 1950-an. Beberapa di antaranya meliputi:

- Bahasa Pemrograman LISP:
  - Bahasa ini memungkinkan penggunaan data nonnumerik (seperti titik data kualitatif);
  - Pengembangan fungsi pemrograman seperti Rekursi, Pengetikan Dinamis, dan Pengumpulan Sampah diciptakan dan diterapkan;

- Komputer mainframe berbagi waktu:
  Komputer-komputer ini diciptakan, yang sebenarnya merupakan cikal bakal Internet pertama, yang disebut "APRANET";
- Mobil yang Dikendalikan Komputer:
  Ini adalah makalah ilmiah yang ia terbitkan yang menjelaskan bagaimana seseorang dapat mengetik petunjuk arah menggunakan papan ketik dan kamera televisi khusus kemudian akan membantu menavigasi kendaraan tersebut. Dalam arti tertentu, ini merupakan versi primitif dari sistem GPS yang tersedia saat ini.

Sejak saat itu, era Kecerdasan Buatan dikenal sebagai "Zaman Keemasan AI", dengan berbagai perkembangan penting yang terjadi. Hal ini akan diulas lebih detail di subbagian berikutnya.

1.4 ZAMAN KEEMASAN KECERDASAN BUATAN

Selama periode ini, banyak inovasi yang terjadi dalam Kecerdasan Buatan berasal dari sektor akademis. Sumber pendanaan utama untuk semua proyek berbasis AI berasal dari Advanced Research Projects Agency, yang juga dikenal sebagai "ARPA". Beberapa perkembangan utama yang terjadi adalah sebagai berikut:

1) Symbolic Automatic INTegrator:
   Juga dikenal sebagai "SAINT", program ini dikembangkan oleh James Slagle, seorang peneliti di MIT, pada tahun 1961. Program ini diciptakan untuk membantu memecahkan masalah dan persamaan kalkulus yang kompleks. Jenis program komputer lain diciptakan dari program ini, yang dikenal sebagai "SIN" dan "MACSYMA", yang memecahkan masalah matematika yang jauh lebih rumit dengan penggunaan aljabar linear dan persamaan diferensial tertentu. SAINT sebenarnya dianggap sebagai apa yang kemudian dikenal sebagai "Sistem Pakar" pertama.
2) ANALOGI:
   Ini adalah program komputer lain yang dikembangkan oleh seorang profesor di MIT bernama Thomas Evans pada tahun 1963. Program ini dirancang khusus untuk memecahkan masalah berbasis analogi yang disajikan dalam tes IQ.
3) MAHASISWA:
   Program komputer jenis ini dikembangkan oleh peneliti lain di MIT, Daniel Bobrow, pada tahun 1964. Program ini adalah yang pertama menggunakan apa yang dikenal sebagai "Pemrosesan Bahasa Alami", dan merupakan topik yang akan dibahas lebih rinci nanti di buku ini.
4) ELIZA:
   Ini juga merupakan program Kecerdasan Buatan lain yang dikembangkan pada tahun 1965 oleh Joseph Weizenbaum, seorang profesor di MIT. Program ini sebenarnya merupakan cikal bakal Chatbot, yang sangat diminati saat ini. Dalam aplikasi khusus ini, pengguna akhir dapat mengetik berbagai pertanyaan, dan komputer akan memberikan semacam respons. Aplikasi di sini ditujukan untuk psikologi—program ini bertindak seperti psikoanalis virtual.

5) Visi Komputer:
   Pada tahun 1966, seorang peneliti di MIT, Marvin Minsky, memimpin perkembangan apa yang dikenal sebagai Visi Komputer, yang merupakan bab selanjutnya dalam buku ini. Ia menghubungkan kamera sederhana ke komputer dan menulis program khusus untuk mendeskripsikan secara detail apa yang dilihatnya. Program tersebut mendeteksi pola visual dasar.
6) Mac Hack:
   Ini juga merupakan program Kecerdasan Buatan lain yang dikembangkan oleh Richard Greenblatt, seorang profesor lain di MIT, pada tahun 1968.
7) Hearsay I:
   Program ini dianggap sebagai salah satu program Kecerdasan Buatan tercanggih pada masanya. Program ini dikembangkan oleh Raj Reddy pada tahun 1968, dan digunakan untuk membuat prototipe pertama Sistem Pengenalan Ucapan.

Selama Periode Keemasan ini, muncul dua teori utama Kecerdasan Buatan, yaitu:

- Kebutuhan akan sistem simbolik: Hal ini akan sangat memanfaatkan logika komputer, seperti pernyataan "Jika-Maka-Lainnya".
- Kebutuhan Sistem Kecerdasan Buatan untuk berperilaku lebih seperti otak manusia: Ini adalah upaya pertama yang diketahui untuk memetakan neuron di otak dan aktivitasnya yang sesuai. Teori ini dikembangkan oleh Frank Rosenblatt, tetapi ia mengganti nama neuron tersebut menjadi "perceptron".

Pada tahun 1957, Rosenblatt menciptakan program Kecerdasan Buatan pertama untuk melakukan hal ini, yang disebut "Mark I Perceptron". Komputer yang menjalankan program khusus ini dilengkapi dengan dua kamera untuk membedakan dua gambar terpisah, yang berskala 20 x 20 piksel. Program ini juga akan menggunakan pembobotan statistik acak untuk menjalani proses iteratif langkah demi langkah ini:

1) Membuat dan memasukkan masukan, tetapi menghasilkan keluaran yang berbasis perceptron.
2) Masukan dan keluaran harus cocok, dan jika tidak, maka langkah-langkah berikut harus diambil:
8) Jika keluaran (perceptron) adalah "I" (bukan 0), bobot statistik untuk "I" harus dikurangi. – Kebalikan dari yang di atas, jika keluaran (perseptron) adalah "0" (bukan I), bobot statistik untuk "I" harus ditingkatkan dengan cara yang sama.
3) Dua langkah pertama harus diulang dalam proses iteratif yang berkelanjutan hingga "I" = 0, atau sebaliknya.

Program ini juga berfungsi sebagai protégé untuk Jaringan Syaraf Tiruan (yang juga merupakan bab selanjutnya dalam buku ini), tetapi meskipun dianggap sukses, program ini juga menuai banyak kritik. Salah satu kelemahan utama yang ditunjukkan adalah hanya memiliki satu lapisan pemrosesan.

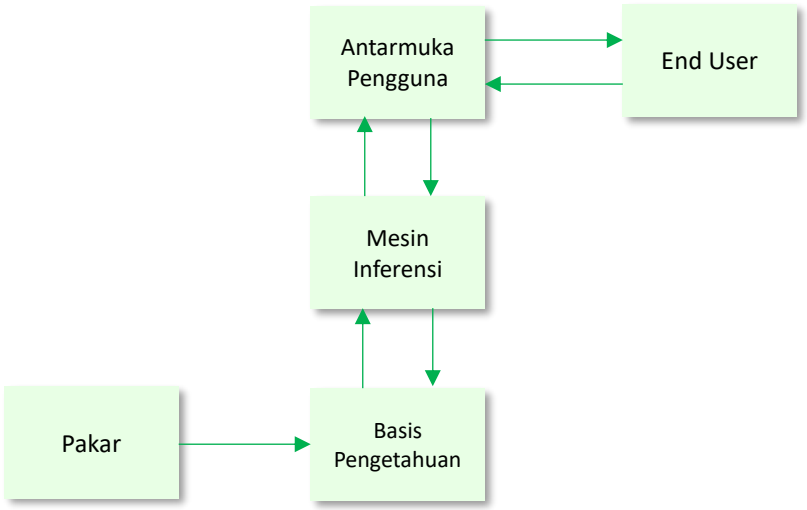
Fase utama berikutnya yang terjadi dalam Kecerdasan Buatan adalah pengembangan Sistem Pakar, yang akan diulas lebih detail di subbagian berikutnya.

1.5 EVOLUSI SISTEM PAKAR

Selama era ini, banyak peristiwa lain yang terjadi di bidang Kecerdasan Buatan. Salah satunya adalah pengembangan teknik propagasi balik. Teknik ini banyak digunakan dalam pembobotan statistik untuk masukan yang masuk ke sistem Jaringan Syaraf Tiruan. Sebagaimana disebutkan sebelumnya, terdapat satu bab dalam buku ini yang membahas topik ini, baik dari sudut pandang teoretis maupun aplikasinya.

Perkembangan penting lainnya adalah terciptanya apa yang dikenal sebagai "Jaringan Syaraf Tiruan Berulang", atau disingkat "RNN". Teknik ini memungkinkan koneksi dalam sistem Kecerdasan Buatan bergerak mulus melalui lapisan masukan dan keluaran. Katalis penting lainnya adalah evolusi Komputer Pribadi dan rekan-rekan minikomputernya, yang kemudian mengarah pada pengembangan apa yang dikenal sebagai "Sistem Pakar", yang banyak menggunakan logika simbolik.

Diagram berikut mengilustrasikan komponen-komponen utama yang terlibat dalam Sistem Pakar:



Dalam hal ini, salah satu contoh terbaik Sistem Pakar adalah "eXpert CONfigurer", yang juga dikenal sebagai "XCON". Sistem ini dikembangkan oleh John McDermott di Universitas Carnegie Mellon. Tujuan utamanya adalah untuk lebih mengoptimalkan pemilihan komponen komputer, dan sistem ini memiliki sekitar 2.500 aturan (matematis dan statistik) yang terintegrasi di dalamnya. Dengan kata lain, sistem ini merupakan cikal bakal Asisten Pribadi Virtual (VPA) Siri dan Cortana, yang memungkinkan Anda membuat pilihan.

Pengembangan XCON semakin mempercepat pertumbuhan Sistem Pakar. Implementasi Sistem Pakar lainnya yang sukses adalah pengembangan "Deep Blue" oleh IBM pada tahun 1996. Bahkan, penerapannya yang paling sukses terjadi ketika ia bermain catur melawan Grandmaster Garry Kasparov. Dalam hal ini, Deep Blue dapat memproses lebih dari 200 juta posisi hanya dalam satu detik.

Namun, terlepas dari semua ini, terdapat sejumlah kekurangan serius pada Sistem Pakar, yaitu sebagai berikut:

- Mereka tidak bisa diterapkan pada aplikasi lain; dengan kata lain, mereka hanya bisa digunakan untuk satu tujuan utama, dan oleh karena itu, mereka memiliki fokus yang sangat sempit.
- Seiring berkembangnya Sistem Pakar, pengelolaannya menjadi jauh lebih sulit dan rumit, tidak hanya untuk mengelolanya tetapi juga untuk terus memberinya data karena semuanya merupakan teknologi berbasis mainframe. Akibatnya, hal ini menyebabkan lebih banyak kesalahan pada keluaran.
- Pengujian Sistem Pakar ini terbukti menjadi proses yang jauh lebih melelahkan dan memakan waktu daripada yang diperkirakan sebelumnya.
- Tidak seperti perangkat Kecerdasan Buatan saat ini, Sistem Pakar tidak dapat belajar sendiri dalam jangka waktu tertentu. Sebaliknya, model logika intinya harus diperbarui secara manual, yang menyebabkan biaya dan tenaga kerja yang jauh lebih besar.

Akhirnya, tahun 1980-an menyaksikan evolusi era baru dalam Kecerdasan Buatan, yang dikenal sebagai "Pembelajaran Mendalam". Secara spesifik, hal ini dapat didefinisikan sebagai berikut:

Pembelajaran mendalam adalah jenis pembelajaran mesin yang melatih komputer untuk melakukan tugas-tugas layaknya manusia, seperti mengenali ucapan, mengidentifikasi gambar, atau membuat prediksi. Alih-alih mengorganisir data untuk dijalankan melalui persamaan yang telah ditentukan sebelumnya, pembelajaran mendalam menetapkan parameter dasar tentang data dan melatih komputer untuk belajar sendiri dengan mengenali pola menggunakan banyak lapisan pemrosesan.

Sederhananya, sistem semacam ini tidak memerlukan algoritma matematika atau statistik yang sudah mapan untuk belajar dari data yang dimasukkan ke dalamnya. Yang dibutuhkan hanyalah permutasi tertentu dan dari sana, sistem tersebut benar-benar dapat belajar sendiri—bahkan membuat proyeksi ke masa depan.

Terdapat juga dua perkembangan besar pada masa ini terkait Pembelajaran Mendalam:

- Pada tahun 1980, Kunihiko Fukushima mengembangkan Kecerdasan Buatan yang disebut "Neocognitron". Ini merupakan cikal bakal lahirnya apa yang dikenal sebagai "Jaringan Saraf Konvolusional", atau singkatnya "CNN". Hal ini didasarkan pada proses yang ditemukan di korteks visual berbagai jenis hewan.
- Pada tahun 1982, John Hopfield mengembangkan sistem Kecerdasan Buatan lain yang disebut "Jaringan Hopfield". Ini meletakkan dasar bagi apa yang dikenal sebagai "Jaringan Saraf Berulang", atau singkatnya "RNN".

Baik CNN maupun RNN akan dibahas dalam bab tentang Jaringan Saraf.

Bagian selanjutnya dari buku ini akan membahas data dan kumpulan data, yang pada dasarnya merupakan bahan bakar yang menggerakkan algoritma dan aplikasi Kecerdasan Buatan dari semua jenis.

1.6 PENTINGNYA DATA DALAM KECERDASAN BUATAN

Sejauh ini dalam bab ini, kita telah mengkaji secara mendalam apa itu Kecerdasan Buatan dan apa saja subkomponennya, serta memberikan landasan yang sangat kuat dalam

hal penerapan teoretis dan praktisnya, yang telah menjadikannya pusat kekuatan dalam Keamanan Siber saat ini. Di bagian bab ini, kita akan berfokus pada unsur utama yang menggerakkan mesin Kecerdasan Buatan saat ini—data yang dimasukkan ke dalamnya, dan sumbernya.

Kita semua tentu pernah mendengar istilah "data". Ini adalah sesuatu yang telah diajarkan kepada kita sejak kita mulai sekolah dasar. Namun, apa sebenarnya data itu? Apa definisi ilmiahnya? Data dapat didefinisikan sebagai berikut: *Dalam komputasi, data adalah informasi yang telah diterjemahkan ke dalam bentuk yang efisien untuk dipindahkan atau diproses. Dibandingkan dengan komputer dan media transmisi saat ini, data adalah informasi yang diubah menjadi bentuk digital biner.*

Jadi, karena ini dapat diterapkan pada Kecerdasan Buatan, alat yang mendasarinya akan mengambil semua data yang dimasukkan ke dalamnya (baik numerik maupun non-numerik), mengubahnya ke dalam format yang dapat dipahami dan diproses, dan dari sana memberikan keluaran yang dibutuhkan. Dalam arti tertentu, ini seperti sampah masuk/sampah keluar, tetapi pada tingkat yang jauh lebih canggih.

Bagian ini akan membahas aspek data dan maknanya bagi Kecerdasan Buatan dari perspektif berikut:

- Fundamental data;
- Jenis data yang tersedia;
- Big Data;
- Memahami penyiapan data;
- Konsep data relevan lainnya yang penting bagi Kecerdasan Buatan.

Fundamental Data

Mari kita akui, ke mana pun kita pergi, kita terpapar data pada tingkat tertentu. Dengan hadirnya ponsel pintar, digitalisasi, teknologi nirkabel, media sosial, Internet of Things (IoT), dan lainya., kita terpapar setiap hari dengan cara yang bahkan tidak kita sadari. Misalnya, ketika kita mengetik pesan teks atau membalas email, hal itu sebenarnya dianggap sebagai data, meskipun lebih bersifat kualitatif. Bahkan video yang dapat Anda akses di YouTube atau podcast juga dapat dianggap sebagai data.

Penting untuk diingat bahwa data tidak harus selalu berupa angka. Jika dipikir-pikir, apa pun yang menghasilkan konten, baik tertulis, dalam bentuk audio atau video, atau bahkan visual, semuanya dianggap sebagai data. Namun dalam dunia Teknologi Informasi, dan bahkan dalam lingkup yang lebih rendah dalam Kecerdasan Buatan, data didefinisikan jauh lebih tepat, dan lebih sering direpresentasikan secara simbolis, terutama ketika kode sumber mengkompilasi kumpulan data yang telah diberikan.

Dalam hal ini, data yang paling sering digunakan oleh komputer adalah data digit biner. Nilainya bisa 0 atau 1, dan faktanya, inilah bagian data terkecil yang dapat diproses komputer. Komputer masa kini dapat memproses data setidaknya 1.000 kali lebih besar, terutama karena besarnya memori yang dimilikinya dan kemampuan pemrosesannya yang sangat canggih.

Dalam hal ini, digit biner sering disebut hanya sebagai "Bit". Data yang lebih besar dari ini disebut sebagai "Byte". Hal ini diilustrasikan dalam tabel di bawah ini:

Unit	*Nilai*
Megabyte	1.000 Kilobyte
Gigabyte	1.000 Megabyte
Terabyte	1.000 Gigabyte
Petabyte	1.000 Terabyte
Exabyte	1.000 Petabyte
Zettabyte	1.000 Exabyte
Yottabyte	1.000 Zettabyte

Jenis-jenis Data yang Tersedia

Secara umum, terdapat empat jenis data yang dapat digunakan oleh sistem Kecerdasan Buatan. Keempatnya adalah sebagai berikut:

1) Data Terstruktur:

   Ini adalah kumpulan data yang memiliki beberapa jenis atau bentuk praformat. Dengan kata lain, kumpulan data tersebut dapat berada dalam bidang tetap di dalam rekaman atau berkas dari dalam basis data yang sedang digunakan. Contoh data terstruktur biasanya mencakup nilai-nilai seperti nama, tanggal, alamat, nomor kartu kredit, harga saham, dll. Beberapa contoh terbaik data terstruktur mungkin adalah berkas Excel, dan data yang disimpan dalam basis data SQL. Biasanya, jenis data ini hanya mencakup 20 persen dari kumpulan data yang digunakan oleh aplikasi atau alat Kecerdasan Buatan. Ini juga disebut sebagai "Data Kuantitatif".

2) Data Tidak Terstruktur:

   Ini adalah kumpulan data yang tidak memiliki format khusus yang telah ditentukan sebelumnya. Dengan kata lain, data tersebut tidak akan dapat dengan mudah dimasukkan ke dalam lembar kerja Excel atau bahkan basis data SQL. Dengan kata lain, ini adalah semua data di luar sana yang memiliki batasan yang tidak didefinisikan secara jelas. Penting untuk diingat bahwa meskipun mungkin tidak memiliki keberadaan eksternal dari kumpulan data yang terorganisir, data tersebut memiliki semacam organisasi internal dan/atau format di dalamnya. Ini juga disebut sebagai "Data Kualitatif", dan contoh-contoh umum dari data ini meliputi:

   - Berkas teks: Pengolah kata, lembar kerja, presentasi, surel, log.
   - Surel: Surel memiliki beberapa struktur internal berkat metadatanya, dan terkadang kita menyebutnya semi-terstruktur. Namun, bidang pesannya tidak terstruktur dan alat analitik tradisional tidak dapat menguraikannya.
   - Media Sosial: Data dari Facebook, Twitter, LinkedIn.
   - Situs web: YouTube, Instagram, situs berbagi foto.
   - Data seluler: Pesan teks, lokasi.
   - Komunikasi: Obrolan, pesan instan, rekaman telepon, perangkat lunak kolaborasi.
   - Media: Berkas MP3, foto digital, audio, dan video.
   - Aplikasi bisnis: dokumen MS Office, aplikasi produktivitas

Jenis dataset ini mencakup sekitar 70 persen data yang digunakan oleh alat Kecerdasan Buatan.

3) Data Semi-Terstruktur:
Sesuai namanya, tidak ada format kaku tentang bagaimana data ini biasanya diorganisasikan, tetapi baik secara eksternal maupun internal, terdapat semacam pengorganisasian di dalamnya. Data ini dapat dimodifikasi lebih lanjut agar sesuai dengan kolom dan bidang basis data, tetapi seringkali, hal ini memerlukan semacam intervensi manusia untuk memastikan bahwa data tersebut diproses dengan cara yang tepat. Beberapa contoh umum dari jenis dataset ini termasuk "Extensible Markup Language", yang juga dikenal sebagai "XML". Sama seperti HTML, XML dianggap sebagai bahasa markup yang terdiri dari berbagai aturan untuk mengidentifikasi dan/atau mengonfirmasi elemen-elemen tertentu dalam sebuah dokumen. Contoh lain dari Data Semi-Terstruktur adalah "JavaScript Object Notation", yang juga dikenal sebagai "JSO". Ini adalah cara di mana informasi dapat ditransfer dari aplikasi web ke sejumlah Antarmuka Protokol Aplikasi (juga dikenal sebagai "API"), dan dari sana, ke server tempat kode sumber aplikasi web berada. Proses ini juga dapat terjadi dalam proses sebaliknya. Jenis dataset ini mencakup sekitar 10 persen data yang dikonsumsi oleh perangkat Kecerdasan Buatan.

4) Data Deret Waktu:
Sesuai namanya, jenis dataset ini terdiri dari titik-titik data yang memiliki nilai waktu tertentu. Terkadang, ini juga dapat disebut sebagai data "Perjalanan", karena selama perjalanan, terdapat titik-titik data yang dapat diakses sepanjang waktu, mulai dari meninggalkan titik asal hingga akhirnya tiba di titik tujuan. Beberapa contoh umum dari hal ini meliputi kisaran harga saham atau komoditas tertentu saat diperdagangkan dalam periode intraday, pertama kali calon pembeli mengunjungi situs web pedagang dan berbagai halaman web yang mereka klik atau materi yang mereka unduh hingga mereka keluar dari situs web, dll.

Setelah kita mendefinisikan empat set data yang paling umum, Anda mungkin bertanya-tanya, apa saja contohnya? Contoh-contoh tersebut antara lain:

Untuk Set Data Terstruktur:

- Basis Data SQL;
- Spreadsheet seperti Excel;
- Sistem OLTP;
- Formulir daring;
- Sensor seperti GPS atau tag RFID;
- Log jaringan dan server web;
- Perangkat medis.

Untuk Set Data Tidak Terstruktur:

- Media sosial;
- Data Lokasi & Geo;
- Generator Mesin & Berbasis Sensor;

- Aliran digital;
- Dokumen teks;
- Log;
  - Transaksi
  - Mikroblog

Untuk Set Data Semi-Terstruktur:

- Email;
- XML dan bahasa markup lainnya;
- Eksekusi Biner;
- Paket TCP/IP;
- Berkas Zip;
- Integrasi data dari berbagai sumber;
- Halaman web (Oracle, n.d.).

Untuk Set Data Deret Waktu:

- Statista;
- Set Data Statistik Data-Planet;
- Euromonitor Passport;
- Statistik OECD;
- Basis Data Statistik Perserikatan Bangsa-Bangsa;
- Data Bank Dunia;
- Biro Sensus AS: Basis Data Internasional;
- Bloomberg;
- Capital IQ;
- Datastream;
- Data Keuangan Global;
- Statistik Keuangan Internasional Daring;
- MarketLine Advantage;
- Morningstar Direct.

Seperti yang telah disebutkan sebelumnya, Set Data Tak Terstrukturlah yang mencakup sebagian besar set data yang dimasukkan ke dalam aplikasi Kecerdasan Buatan, dan ada keunggulan tersendiri. Set Data Tak Terstruktur begitu canggih sehingga dapat mengambil hampir semua jenis set data yang disajikan, mengolahnya menjadi format yang dapat dipahami, memprosesnya, dan memberikan keluaran yang dibutuhkan. Dengan kata lain, tidak ada faktor pembatas dalam hal ini, dan sebagai hasilnya, set Data Tak Terstruktur dapat memberikan hampir semua jenis prediksi atau jawaban yang diminta.

Big Data

Sebagaimana telah diulas sebelumnya, ukuran dan jumlah kumpulan data tumbuh secara eksponensial setiap harinya, mengingat semua kemajuan teknologi yang sedang terjadi. Ada istilah khusus untuk ini, yaitu "Big Data". Definisi teknisnya adalah sebagai berikut: *Big data adalah kumpulan data yang lebih besar dan lebih kompleks, terutama dari sumber data baru. Kumpulan data ini sangat besar sehingga perangkat lunak pemrosesan data tradisional*

tidak dapat mengelolanya. Namun, volume data yang sangat besar ini dapat digunakan untuk mengatasi masalah bisnis yang sebelumnya tidak dapat diatasi.

Dalam arti tertentu, hal ini juga dapat disamakan dengan konsep lain yang dikenal sebagai "Pergudangan Data". Ada tiga karakteristik utama yang terkait dengan "Big Data", yaitu sebagai berikut:

1) Volume:
   Ini mengacu pada ukuran dan skala kumpulan data. Seringkali, data tersebut berupa Data Tidak Terstruktur. Ukuran kumpulan data ini dapat mencapai Terabyte.
2) Variasi:
   Ini menggambarkan keragaman semua kumpulan data yang terdapat dalam Big Data. Ini mencakup Data Terstruktur, Data Tidak Terstruktur, Data Semi-Terstruktur, dan Data Deret Waktu. Ini juga menggambarkan sumber asal semua kumpulan data ini.
3) Kecepatan:
   Ini merujuk pada kecepatan yang sangat cepat di mana dataset dalam Big Data sebenarnya dibuat.
4) Nilai:
   Ini mengacu pada seberapa bermanfaat Big Data. Dengan kata lain, jika dimasukkan ke dalam sistem Kecerdasan Buatan, seberapa dekat data tersebut akan memberikan keluaran yang diinginkan atau diharapkan?
5) Variabilitas:
   Ini menggambarkan seberapa cepat kumpulan data dalam Big Data akan berubah dalam periode waktu tertentu. Misalnya, Data Terstruktur, Data Deret Waktu, dan Data Semi-Terstruktur tidak akan banyak berubah, tetapi Data Tidak Terstruktur akan berubah. Hal ini semata-mata karena sifatnya yang dinamis.
6) Visualisasi:
   Ini adalah cara alat bantu visual digunakan dalam dataset yang ada di Big Data. Sebagai contoh, ini bisa berupa grafik, dasbor, dll.

Memahami Persiapan Data

Seperti yang telah disebutkan sebelumnya, datalah yang mendorong aplikasi Kecerdasan Buatan untuk melakukan tugasnya. Dengan kata lain, data bagaikan bahan bakar yang dibutuhkan aplikasi ini untuk berjalan. Meskipun aplikasi cukup tangguh dalam menyediakan keluaran yang diminta, hal ini masih dipandang sebagai proses "Garbage In and Garbage Out". Artinya, kualitas keluaran yang akan Anda dapatkan hanya akan sebaik data yang dimasukkan ke dalam aplikasi.

Oleh karena itu, Anda harus berupaya keras untuk memastikan bahwa kumpulan data yang Anda masukkan ke dalam sistem Kecerdasan Buatan Anda sangat tangguh dan akan memenuhi kebutuhan yang Anda harapkan dalam hal keluaran yang diinginkan. Langkah pertama dalam proses ini dikenal sebagai "Pemahaman Data":

1) Pemahaman Data:

Dalam hal ini, Anda perlu menilai dengan cermat dari mana sumber data dan umpannya masing-masing berasal. Tergantung pada keadaan dan kebutuhan spesifik Anda, data tersebut biasanya berasal dari sumber-sumber berikut:

- Data Internal:
  Sesuai namanya, ini adalah titik data yang benar-benar masuk ke bisnis atau perusahaan Anda. Misalnya, bisa jadi data yang berasal dari intranet perusahaan Anda, atau bahkan situs web eksternal Anda, saat pelanggan dan calon pelanggan mengunduh materi dari situs Anda atau bahkan mengisi formulir kontak. Selain itu, mungkin saja Anda sudah memiliki kumpulan data di organisasi Anda yang dapat Anda gunakan.
- Data Sumber Terbuka:
  Ini adalah jenis data yang tersedia secara bebas dari Internet, terutama saat Anda menggunakan Google untuk menemukan berbagai sumber data. Misalnya, Pemerintah Federal merupakan sumber daya yang sangat baik untuk hal ini, begitu pula banyak perusahaan swasta (tentu saja, Anda harus membayar langganan untuk ini, tetapi pada awalnya, mereka kemungkinan besar akan menawarkan uji coba gratis untuk menguji dataset mereka masing-masing). Ini akan menjadi peluang bagus untuk melihat apakah yang mereka tawarkan akan kompatibel dengan sistem Kecerdasan Buatan Anda, dan apakah berpotensi menghasilkan keluaran yang diinginkan. Dataset semacam ini kemungkinan besar akan menggunakan Antarmuka Protokol Aplikasi (API) khusus untuk mengunduh data. Selain keuntungan gratis, keuntungan utama lainnya dari penggunaan Data Sumber Terbuka adalah data tersebut sudah tersedia dalam format yang dapat diunggah dan dimasukkan ke dalam sistem Kecerdasan Buatan Anda.
- Data Pihak Ketiga:
  Ini adalah jenis dataset yang tersedia secara eksklusif dari vendor eksternal. Contohnya dapat dilihat di subbagian terakhir bab ini. Keuntungan utama memperoleh data dari sumber-sumber ini adalah Anda dapat dijamin, sampai batas tertentu, bahwa data tersebut telah divalidasi. Namun, kerugiannya adalah biayanya bisa sangat mahal, dan jika Anda perlu memperbaruinya, Anda harus membayar biaya yang lebih tinggi. kumpulan data Anda, Anda harus kembali ke vendor yang sama dan membayar harga premium lagi untuk itu.

Menurut penelitian terbaru, sekitar 70 persen sistem Kecerdasan Buatan yang digunakan saat ini menggunakan Data Internal, 20 persen di antaranya menggunakan Data Sumber Terbuka, dan 10 persen sisanya berasal dari vendor eksternal. Untuk sepenuhnya memahami ketahanan dataset yang akan Anda peroleh, hal-hal berikut harus dijawab terlebih dahulu:

- Apakah dataset tersebut lengkap sesuai kebutuhan dan persyaratan Anda? Apakah ada data yang hilang?
- Bagaimana data tersebut awalnya dikumpulkan?
- Bagaimana data tersebut awalnya diproses?

- Apakah ada perubahan signifikan yang perlu Anda ketahui?
- Apakah ada masalah Kontrol Kualitas (QC) dengan dataset tersebut?

2) Persiapan Data:

Bagian ini sering disebut sebagai "Pembersihan Data", dan memerlukan tindakan berikut yang harus Anda lakukan sebelum Anda dapat memasukkan data ke dalam sistem Kecerdasan Buatan Anda:

- Deduplikasi:
  Sangat penting untuk memastikan bahwa data Anda tidak mengandung set duplikat. Jika hal ini terjadi, dan tidak disadari, hal ini dapat sangat memengaruhi dan mendistorsi output yang dihasilkan.
- Outlier:
  Ini adalah titik data yang terletak di titik ekstrem dari set data lainnya. Mungkin outlier ini berguna untuk suatu tujuan, tetapi Anda perlu memastikan terlebih dahulu bahwa outlier tersebut diperlukan untuk aplikasi spesifik Anda. Jika tidak, maka outlier tersebut harus dihapus.
- Konsistensi:
  Dalam situasi ini, Anda harus memastikan bahwa semua variabel memiliki definisi yang jelas, dan Anda mengetahui artinya. Tidak boleh ada tumpang tindih makna antara outlier dengan variabel lainnya.
- Aturan Validasi:
  Di sinilah Anda mencoba menemukan batasan teknis dari kumpulan data yang ingin Anda gunakan. Melakukannya secara manual bisa sangat memakan waktu dan tenaga, sehingga terdapat banyak aplikasi perangkat lunak yang tersedia yang dapat membantu Anda menentukan batasan-batasan spesifik ini. Tentu saja, pertama-tama Anda perlu memutuskan dan memasukkan permutasi yang relevan, yang dapat disebut sebagai "ambang batas".
- Pengelompokan:
  Saat Anda mendapatkan kumpulan data, mungkin juga Anda tidak memerlukan setiap data untuk dimasukkan ke dalam sistem Kecerdasan Buatan Anda. Oleh karena itu, Anda harus melihat setiap kategori dan memutuskan mana yang paling relevan untuk keluaran yang ingin Anda dapatkan.
- Kedaluwarsa:
  Ini mungkin salah satu faktor terpenting yang perlu dipertimbangkan. Seberapa tepat waktu dan relevankah kumpulan data yang Anda gunakan? Untuk aplikasi Kecerdasan Buatan, sangat penting bagi Anda untuk mendapatkan data yang diperbarui secara real-time jika keluaran yang Anda inginkan adalah memprediksi sesuatu di masa mendatang.
- Penggabungan:
  Bisa jadi dua kolom dalam kumpulan data Anda berisi informasi yang sangat mirip. Jika demikian, Anda mungkin ingin mempertimbangkan untuk menggabungkan kedua kolom ini. Dengan demikian, Anda sebenarnya

menggunakan kemampuan pemrosesan Kecerdasan Buatan Anda dengan jauh lebih efisien.

- Pengodean Cepat:
  Sampai batas tertentu, data kualitatif mungkin dapat direpresentasikan sebagai data kuantitatif, sekali lagi, tergantung pada kebutuhan dan persyaratan Anda.
- Konversi:
  Ini lebih merupakan aspek pemformatan unit untuk menentukan bagaimana Anda ingin keluaran Anda terlihat. Misalnya, jika semua kumpulan data Anda menggunakan sistem desimal, tetapi keluaran Anda mengharuskan nilai-nilai tersebut menggunakan sistem metrik, maka penggunaan teknik ini akan menjadi penting.
- Menemukan Data yang Hilang:
  content − type = "list2 − para" >Saat Anda memeriksa kumpulan data Anda secara saksama, sering kali mungkin ada beberapa bagian yang hilang. Dalam hal ini, terdapat dua jenis data yang hilang:
  - Data yang hilang secara acak: Di sini, Anda dapat menghitung median atau bahkan rata-rata sebagai nilai pengganti. Dengan melakukan ini, output seharusnya hanya sedikit mendistorsinya.
  - Data yang hilang secara berurutan: Ini terjadi ketika data hilang secara berurutan, secara iteratif. Menghitung median atau rata-rata tidak akan berhasil karena terlalu banyak data yang tidak tersedia untuk membentuk estimasi ilmiah. Anda dapat mencoba mengekstrapolasi data sebelumnya dan data selanjutnya untuk membuat hipotesis, tetapi ini merupakan langkah yang lebih berisiko. Atau, Anda dapat menghapus kolom-kolom yang data berurutannya hilang. Namun, dalam kedua kasus tersebut, kemungkinan output akan jauh lebih miring dan tidak terlalu andal jauh lebih besar.
- Memperbaiki Ketidakselarasan Data:
  content − type = "list2 − para" >Penting untuk diperhatikan bahwa sebelum Anda menggabungkan kolom apa pun dalam kumpulan data Anda, pastikan titik data masing-masing "selaras" dengan kumpulan data lain yang Anda miliki. Untuk memperhitungkan dan memperbaikinya, pertimbangkan tindakan berikut yang dapat Anda lakukan:
  - Jika memungkinkan, cobalah untuk menghitung dan memastikan data yang hilang yang mungkin Anda miliki dalam kumpulan data Anda (seperti yang telah diulas sebelumnya);
  - Temukan data lain yang hilang di semua kumpulan data lain yang Anda miliki dan ingin gunakan;
  - Cobalah untuk menggabungkan kumpulan data sehingga Anda memiliki kolom yang dapat menyediakan kolom yang konsisten;

– Jika perlu, modifikasi atau tingkatkan lebih lanjut hasil yang diinginkan dari keluaran tersebut untuk mengakomodasi setiap perubahan yang telah dilakukan untuk memperbaiki ketidakselarasan data.

Konsep Data Relevan Lainnya yang Penting bagi Kecerdasan Buatan

Terakhir, dalam subbagian ini kami mengkaji beberapa konsep data lain yang sangat relevan dengan sistem Kecerdasan Buatan, yaitu sebagai berikut:

1) Analisis Diagnostik:
   Ini adalah pemeriksaan cermat terhadap kumpulan data untuk melihat mengapa tren tertentu terjadi. Contohnya adalah menemukan tren tersembunyi yang mungkin belum diketahui sebelumnya. Hal ini sangat sering dilakukan dalam proyek Pergudangan Data atau Big Data.
2) Ekstraksi, Transformasi, dan Pemuatan (ETL):
   Ini adalah jenis integrasi data khusus, dan biasanya digunakan dalam aplikasi Pergudangan Data.
3) Fitur:
   Ini adalah kolom data.
4) Instansi:
   Ini adalah baris data.
5) Metadata:
   Ini adalah data yang tersedia tentang kumpulan data.
6) Pemrosesan Analitik Daring (OLAP):
   Ini adalah teknik yang memungkinkan Anda memeriksa kumpulan data dari berbagai jenis basis data ke dalam satu tampilan yang terharmonisasi.
7) Data Kategorikal:
   Jenis data ini tidak memiliki nilai numerik, tetapi memiliki makna tekstual yang terkait dengannya.
8) Data Ordinal:
   Ini merupakan campuran dari Data Kategorikal dan Data Numerik.
9) Analisis Prediktif:
   Di sinilah sistem Kecerdasan Buatan mencoba membuat prediksi tertentu tentang masa depan (yang ditampilkan sebagai keluaran), berdasarkan kumpulan data yang dimasukkan ke dalamnya.
10) Analisis Preskriptif:
    Di sinilah konsep Big Data (seperti yang telah dibahas sebelumnya) digunakan untuk membantu membuat keputusan yang lebih baik berdasarkan keluaran yang dihasilkan.
11) Variabel Skalar:
    Ini adalah jenis variabel yang menyimpan dan hanya terdiri dari nilai tunggal.
12) Data Transaksional:
    Ini adalah jenis kumpulan data yang merepresentasikan data untuk transaksi aktual yang telah terjadi dalam kegiatan bisnis sehari-hari.

Sejauh ini, kami telah memberikan gambaran umum yang luas tentang betapa pentingnya data dan kumpulan data bagi sistem Kecerdasan Buatan. Sisa buku ini akan membahas Pembelajaran Mesin, Jaringan Saraf Tiruan, dan Visi Komputer secara lebih mendalam.

BAB 2
PEMBELAJARAN MESIN

Pada bab terakhir (Bab 1), kami mengulas apa itu Kecerdasan Buatan dengan memberikan gambaran umum. Secara spesifik, topik-topik berikut dibahas:

- Pengantar Keamanan Siber;
- Berbagai aspek Keamanan Siber;
- Linimasa kronologis evolusi Keamanan Siber;
- Pengantar Kecerdasan Buatan;
- Definisi Kecerdasan Buatan;
- Berbagai komponen Kecerdasan Buatan dan definisi teknisnya (termasuk Pembelajaran Mesin, Visi Komputer, dan Jaringan Syaraf Tiruan);
- Tinjauan umum buku ini;
- Sejarah Kecerdasan Buatan;
- Pentingnya data dan perannya dalam sistem dan aplikasi Kecerdasan Buatan;
- Aplikasi Kecerdasan Buatan.

Dalam bab ini, kami mengkaji subkomponen pertama Kecerdasan Buatan, yaitu Pembelajaran Mesin, yang juga dikenal sebagai "ML". Kita akan mendalami aspek teoretis Pembelajaran Mesin terlebih dahulu, kemudian dilanjutkan dengan berbagai aplikasinya, seperti pada bab sebelumnya. Namun, sebelum kita membahas semua aspek teoretis Pembelajaran Mesin, pertama-tama kita akan memberikan gambaran umum tentang apa itu Pembelajaran Mesin.

2.1 IKHTISAR TINGKAT TINGGI

Meskipun Pembelajaran Mesin telah ada sejak lama (beberapa perkiraan menyebutkan hingga beberapa dekade), terdapat sejumlah aplikasi utama yang menggunakan Pembelajaran Mesin. Beberapa contohnya adalah sebagai berikut:

1) Pemeliharaan Prediktif:

   Aplikasi semacam ini biasanya digunakan dalam sektor rantai pasok, manufaktur, distribusi, dan logistik. Misalnya, di sinilah konsep Pengendalian Kualitas berperan penting. Dalam manufaktur, Anda ingin dapat memprediksi berapa banyak batch produk yang akan diproduksi yang berpotensi mengalami cacat. Tentu saja, Anda ingin angka ini serendah mungkin. Secara teoritis, Anda tidak ingin jenis atau jenis produk apa pun mengalami cacat, tetapi dalam dunia nyata, hal ini hampir mustahil untuk dicapai. Dengan Pembelajaran Mesin, Anda dapat mengatur berbagai permutasi dalam algoritma matematika dan statistik dengan permutasi yang berbeda mengenai apa yang dianggap sebagai produk cacat atau tidak.

2) Rekrutmen Karyawan:

Ada satu kesamaan dalam industri rekrutmen, yaitu banyaknya resume yang diterima perekrut dari berbagai industri. Pertimbangkan beberapa statistik berikut:

- Baru-baru ini, Career Builder, salah satu portal pencarian kerja yang paling banyak digunakan, melaporkan:
  * 2,3 juta lowongan pekerjaan diposting;
  * 680 profil unik pencari kerja dikumpulkan;
  * 310 juta resume dikumpulkan;
  * 2,5 juta pemeriksaan latar belakang dilakukan dengan platform Career Builder.

Bayangkan berapa lama waktu yang dibutuhkan tim perekrut untuk memeriksa semua hal di atas. Namun dengan Pembelajaran Mesin, semua itu dapat dilakukan dalam hitungan menit, dengan memeriksanya berdasarkan kata kunci tertentu untuk menemukan kandidat yang diinginkan. Selain itu, alih-alih membiarkan perekrut memposting setiap lowongan pekerjaan secara manual ke Career Builder, alat Pembelajaran Mesin yang tepat dapat digunakan untuk mengotomatiskan proses ini sepenuhnya, sehingga menghemat waktu perekrut untuk mewawancarai kandidat yang tepat untuk pekerjaan tersebut.

3) Pengalaman Pelanggan:

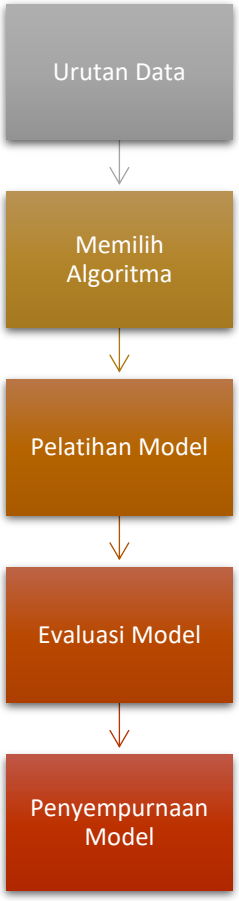
Dalam masyarakat Amerika saat ini, kita ingin semuanya ada di sini dan saat ini juga, dalam sekejap. Tidak hanya itu, tetapi di atas semua ini, kita juga mengharapkan layanan pelanggan yang sempurna. Dan ketika semua ini tidak terjadi, kita memiliki kemewahan untuk bertanya kepada pesaing dan melihat apakah mereka dapat melakukan yang lebih baik. Dalam hal ini, banyak bisnis dan perusahaan telah mulai menggunakan Agen Virtual. Ini adalah kotak obrolan kecil yang biasanya terdapat di bagian kanan bawah peramban web Anda. Dengan ini, Anda benar-benar dapat berkomunikasi dengan seseorang untuk mendapatkan jawaban atas pertanyaan Anda atau menyelesaikan masalah belanja. Hal yang menyenangkan tentang ini adalah mereka juga tersedia sesuai permintaan, 24/7/365. Namun, untuk memberikan pengalaman yang lancar kepada pelanggan atau calon pelanggan, banyak entitas bisnis kini memanfaatkan apa yang dikenal sebagai "Chat Bot". Ini adalah versi Agen Virtual yang jauh lebih canggih karena menggunakan algoritma Pembelajaran Mesin. Dengan demikian, Chat Bot dapat menemukan jawaban yang jauh lebih spesifik atas pertanyaan Anda dengan melakukan pencarian yang lebih "cerdas" di repositori informasi bisnis atau perusahaan. Banyak pusat panggilan juga memanfaatkan Pembelajaran Mesin. Dengan cara ini, ketika pelanggan menelepon, riwayat panggilan, profil, dan seluruh percakapan mereka akan ditampilkan dalam hitungan detik untuk agen pusat panggilan, sehingga mereka dapat lebih mudah mengantisipasi pertanyaan Anda dan memberikan layanan terbaik.

4) Keuangan:

Di segmen pasar ini, ada satu hal yang diinginkan semua orang, terutama para pedagang, yaitu kemampuan untuk memprediksi pasar keuangan, serta apa yang akan mereka lakukan di masa mendatang, sehingga mereka dapat melakukan lindung nilai

dan menghasilkan perdagangan yang menguntungkan. Bahasa Indonesia: Meskipun ini dapat dilakukan melalui proses manual, itu bisa menjadi proses yang sangat melelahkan dan memakan waktu untuk dicapai. Tentu saja, kita semua tahu bahwa pasar dapat bergerak dalam hitungan detik dengan volatilitas yang tidak pasti, seperti yang telah kita lihat baru-baru ini dengan Coronavirus. Faktanya, waktu yang tepat dan memprediksi pasar keuangan dengan akurasi 100 persen adalah hal yang hampir mustahil untuk dicapai. Tetapi di sinilah peran Pembelajaran Mesin dapat dimainkan. Misalnya, ia dapat mengambil semua data yang dimasukkan ke dalamnya, dan dalam hitungan detik membuat prediksi yang lebih akurat tentang apa yang berpotensi dilakukan pasar, memberi para pedagang waktu yang berharga untuk membuat keputusan sepersekian detik yang diperlukan untuk menghasilkan perdagangan yang berkualitas. Ini sangat berguna untuk apa yang dikenal sebagai "Perdagangan Intra Day," di mana para pedagang keuangan mencoba mengatur waktu pasar saat mereka buka setiap menit.

2.2 PROSES PEMBELAJARAN MESIN



Ketika Anda menerapkan Pembelajaran Mesin pada pertanyaan tertentu yang ingin Anda jawab atau untuk memprediksi hasil tertentu, sangat penting untuk mengikuti proses yang spesifik agar dapat menyelesaikan tugas-tugas ini. Dengan kata lain, Anda ingin membangun model yang efektif yang dapat berfungsi dengan baik untuk tujuan dan sasaran lain di kemudian hari. Dengan kata lain, Anda ingin melatih model ini dengan cara tertentu, sehingga dapat memberikan tingkat akurasi dan keandalan yang sangat tinggi. Proses ini digambarkan di samping:

Urutan Data

Pada langkah ini, Anda ingin memastikan bahwa data tidak terorganisir dan tidak terurut sebisa mungkin. Meskipun terdengar agak bertentangan, jika kumpulan data diurutkan atau diatur dalam bentuk apa pun, Algoritma Pembelajaran Mesin yang digunakan dapat mendeteksi hal ini sebagai sebuah pola, yang tentu tidak Anda inginkan terjadi dalam kasus khusus ini.

Memilih Algoritma

Pada fase ini, Anda perlu memilih algoritma Pembelajaran Mesin yang tepat untuk model Anda. Hal ini akan dibahas secara mendalam di bagian bab ini.

Melatih Model

Kumpulan data yang Anda miliki akan dimasukkan ke dalam sistem Pembelajaran Mesin agar dapat dipelajari terlebih dahulu. Dengan kata lain, berbagai asosiasi dan hubungan akan dibuat dan diperiksa sehingga keluaran yang diinginkan dapat diformulasikan. Misalnya, salah satu algoritma paling sederhana yang dapat

digunakan dalam Pembelajaran Mesin adalah Regresi Linier, yang direpresentasikan secara matematis sebagai berikut:

$$Y = M^{*}X + B$$

Di mana:
M = kemiringan pada grafik;
B = intersep Y pada grafik.

Evaluasi Model

Pada langkah ini, Anda akan menggunakan sampel data representatif dari kumpulan data, yang secara teknis dikenal sebagai "Data Uji". Dengan memasukkan data ini terlebih dahulu ke dalam sistem Pembelajaran Mesin, Anda dapat mengukur seberapa akurat keluaran yang Anda inginkan dalam lingkungan pengujian sebelum merilis kumpulan data Anda ke lingkungan produksi.

Penyempurnaan Model

Pada fase terakhir ini, Anda akan menyesuaikan permutasi yang telah Anda buat dalam sistem Pembelajaran Mesin agar dapat menghasilkan keluaran yang Anda inginkan secara wajar. Pada subbagian berikutnya, kita akan membahas klasifikasi utama dan jenis Algoritma Pembelajaran Mesin yang umum digunakan saat ini.

Klasifikasi Algoritma Pembelajaran Mesin

Ada empat kategorisasi utama Algoritma Pembelajaran Mesin, yaitu sebagai berikut:

1) Pembelajaran Terbimbing:

   Jenis algoritma ini menggunakan apa yang dikenal sebagai "data berlabel". Ini berarti bahwa setiap kumpulan data memiliki label tertentu yang terkait dengannya. Dalam hal ini, salah satu hal penting yang perlu diingat adalah Anda perlu memiliki sejumlah besar dataset untuk menghasilkan dataset yang Anda cari ketika menggunakan algoritma berdasarkan kategori ini. Namun, jika dataset tersebut belum diberi label, akan sangat memakan waktu untuk membuat dan menetapkan label untuk setiap dataset. Inilah kelemahan utama penggunaan algoritma Pembelajaran Mesin dari kategori khusus ini.

2) Pembelajaran Tanpa Pengawasan:

   Algoritma jenis ini bekerja dengan data yang biasanya tidak diberi label. Karena keterbatasan waktu yang diperlukan untuk membuat dan menetapkan label untuk setiap kategori (seperti yang telah disebutkan sebelumnya), Anda harus menggunakan apa yang dikenal sebagai "Algoritma Pembelajaran Mendalam" untuk mendeteksi tren data yang tidak terlihat yang terdapat dalam semua dataset Anda. Dalam hal ini, salah satu pendekatan yang paling umum digunakan dalam kategori ini adalah "Pengelompokan". Dengan ini, Anda hanya mengambil semua kumpulan data yang tidak berlabel dan menggunakan berbagai algoritma yang tersedia dalam kategori khusus ini untuk mengelompokkan kumpulan data tersebut ke dalam berbagai

kelompok yang memiliki kesamaan penyebut atau afiliasi. Untuk membantu hal ini, ada beberapa cara untuk melakukannya, yaitu sebagai berikut:

- Metrik Euclidean:
  Ini adalah garis lurus antara dua kumpulan data independen.
- Metrik Kesamaan Kosinus:
  Dalam hal ini, fungsi trigonometri yang dikenal sebagai "Kosinus" digunakan untuk mengukur sudut-sudut tertentu di antara kumpulan data. Tujuannya adalah untuk menemukan kedekatan atau kesamaan antara setidaknya dua atau lebih kumpulan data independen berdasarkan orientasi geometrisnya.
- Metrik Manhattan:
  Teknik ini melibatkan penjumlahan setidaknya dua atau lebih jarak nilai absolut dari kumpulan data yang Anda miliki.
- Asosiasi:
  Intinya di sini adalah jika suatu kejadian spesifik terjadi di salah satu set data Anda, maka kemungkinan besar kejadian tersebut juga akan terjadi di set data yang memiliki hubungan dengan set data awal yang telah digunakan.
- Deteksi Anomali:
  Dengan metodologi ini, Anda mengidentifikasi secara statistik outlier atau pola anomali lain yang mungkin ada dalam set data Anda. Teknik ini telah banyak digunakan dalam Keamanan Siber, terutama dalam hal penyaringan positif palsu dari berkas log yang dikumpulkan dari Firewall, Perangkat Intrusi Jaringan, dan Router, serta perilaku apa pun yang mungkin dianggap mencurigakan atau berbahaya.
- Autoencoder:
  Dengan teknik khusus ini, set data yang Anda miliki akan diformat dan dimasukkan ke dalam format terkompresi, dan dari sana, akan direkonstruksi sekali lagi. Ide di balik ini adalah untuk mendeteksi dan menemukan segala jenis pola baru atau tren tersembunyi yang mungkin ada di dalam set data Anda.

3) Pembelajaran Penguatan:
   Dalam hal ini, Anda mempelajari dan memanfaatkan kekuatan kumpulan data Anda melalui proses coba-coba, sesuai dengan nama kategori ini.
4) Pembelajaran Semi-Supervised:
   Metodologi ini sebenarnya merupakan gabungan antara Pembelajaran Supervised dan Pembelajaran Tanpa Supervised. Namun, teknik ini hanya digunakan ketika Anda memiliki sejumlah kecil kumpulan data yang diberi label. Di dalamnya, terdapat sub-teknik yang disebut "Pseudo-Labeling". Dalam hal ini, Anda secara harfiah menerjemahkan semua kumpulan data tanpa supervisi ke dalam keadaan alami yang tersupervisi.

Algoritma Pembelajaran Mesin

Ada banyak jenis dan ragam algoritma matematika dan statistik yang digunakan dalam Pembelajaran Mesin. Dalam subbagian ini, kita akan membahas beberapa algoritma yang lebih

umum, dan kita akan membahasnya lebih lanjut nanti di bab ini. Berikut algoritma-algoritma tersebut:

1) Klasifikasi Bayes Naif:

   Alasan mengapa algoritma khusus ini disebut "naif" adalah karena asumsi yang mendasarinya adalah bahwa variabel-variabel dalam setiap dataset yang Anda miliki sebenarnya semuanya independen satu sama lain. Dengan kata lain, kemunculan statistik dari satu variabel dalam satu dataset tidak akan ada hubungannya sama sekali dengan variabel-variabel dalam dataset yang tersisa. Namun, ada argumen tandingan yang menyatakan bahwa asosiasi ini akan terbukti salah secara statistik jika ada dataset yang benar-benar berubah dalam hal nilai-nilai yang sesuai.Perlu dicatat bahwa ada juga perubahan atau variasi spesifik pada algoritma khusus ini, dan variasinya adalah sebagai berikut:

   - Bermoulli:
     Ini hanya digunakan jika Anda memiliki nilai biner dalam dataset Anda.
   - Multinomial:
     Teknik ini hanya digunakan jika nilai-nilai dalam kumpulan data Anda bersifat diskrit, dengan kata lain, jika nilai-nilai tersebut mengandung nilai absolut matematis.
   - Gaussian:
     Metodologi ini hanya digunakan jika kumpulan data Anda berdistribusi normal secara statistik.

     Perlu dicatat bahwa teknik umum ini banyak digunakan untuk menganalisis secara detail data-data yang memiliki nilai teks. Dalam Keamanan Siber, teknik ini terbukti sangat berguna dalam mengidentifikasi dan mengonfirmasi email phishing dengan memeriksa fitur dan pola utama dalam isi pesan email, alamat pengirim, dan konten di baris subjek.

2) K-Nearest Neighbor:

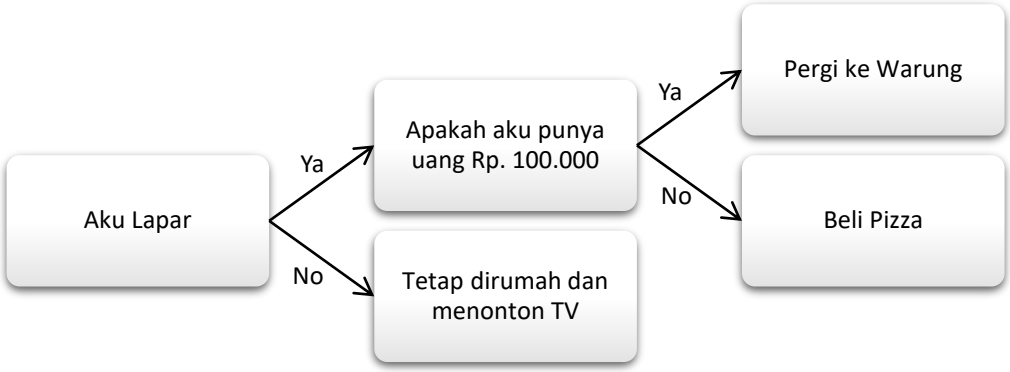
   Metodologi khusus ini digunakan untuk mengklasifikasikan setiap kumpulan data yang Anda miliki. Konstruksi teoretis dasar dari nilai-nilai yang terkait erat atau terkait satu sama lain dalam kumpulan data Anda secara statistik akan menjadi prediktor yang baik untuk model Pembelajaran Mesin. Untuk menggunakan model ini, pertama-tama Anda perlu menghitung jarak numerik antara nilai-nilai terdekat. Jika nilai-nilai ini kuantitatif, Anda dapat menggunakan rumus Jarak Euclidean. Namun, jika dataset Anda memiliki nilai kualitatif, Anda dapat menggunakan apa yang dikenal sebagai "Metrik Tumpang Tindih". Selanjutnya, Anda harus memastikan jumlah total nilai yang saling berkaitan erat. Meskipun memiliki lebih banyak nilai seperti ini dalam dataset Anda dapat menghasilkan Model Pembelajaran Mesin yang jauh lebih efisien dan tangguh, hal ini juga berarti penggunaan sumber daya pemrosesan Sistem Pembelajaran Mesin Anda yang jauh lebih banyak. Untuk mengakomodasi hal ini, Anda selalu dapat menetapkan bobot statistik dengan nilai yang lebih tinggi pada nilai-nilai tertentu yang saling berkaitan erat.

3) Regresi Linier:

Metodologi semacam ini sepenuhnya bersifat statistik. Artinya, metodologi ini mencoba memeriksa dan memastikan hubungan antara variabel-variabel yang telah ditentukan sebelumnya yang terdapat dalam dataset Anda. Dengan metode ini, sebuah garis biasanya diplot, dan dapat dihaluskan lebih lanjut menggunakan teknik yang disebut "Kuadrat Terkecil".

4) Pohon Keputusan:

Metodologi ini sebenarnya menyediakan alternatif untuk teknik-teknik lain yang telah dijelaskan sejauh ini. Faktanya, Pohon Keputusan bekerja jauh lebih baik dan jauh lebih efisien dengan data non-numerik, seperti data yang berkaitan dengan nilai teks. Titik awal utama keputusan terletak pada simpul, yang biasanya dimulai di bagian atas setiap bagan. Dari titik ini, akan ada serangkaian cabang keputusan yang muncul, sehingga dinamakan Pohon Keputusan. Contoh berikut menggambarkan contoh Pohon Keputusan yang sangat sederhana:



Gambar di atas tentu saja merupakan Pohon Keputusan yang sangat sederhana untuk menggambarkan poin tersebut. Namun, dalam Pembelajaran Mesin, Pohon Keputusan bisa menjadi sangat panjang, detail, dan jauh lebih kompleks. Salah satu keuntungan utama menggunakan Pohon Keputusan adalah pohon ini dapat bekerja dengan sangat baik dengan kumpulan data yang sangat besar dan memberikan tingkat transparansi selama proses pembangunan Model Pembelajaran Mesin.

Namun, di sisi lain, Pohon Keputusan juga memiliki kekurangan yang serius. Misalnya, jika hanya satu cabang yang gagal, hal itu akan berdampak negatif dan berjenjang pada cabang-cabang lain dari Pohon Keputusan.

5) Model Ensemble:

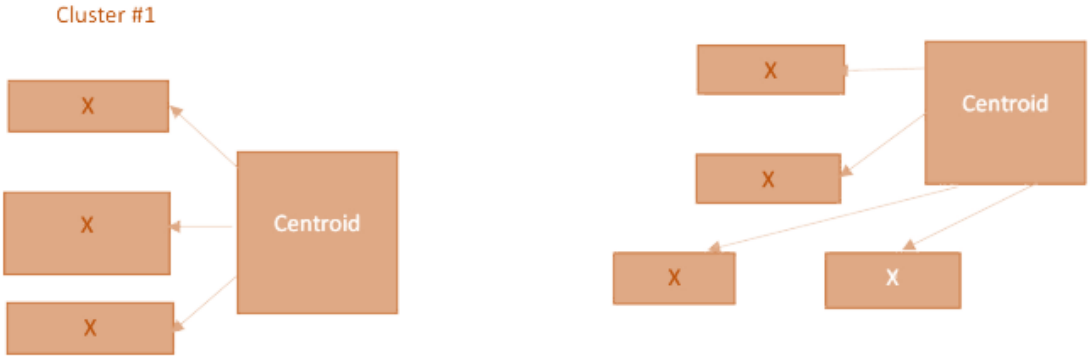
Sesuai namanya, teknik khusus ini berarti menggunakan lebih dari satu model, melainkan menggunakan kombinasi dari apa yang telah diulas sejauh ini.

6) Pengelompokan K-Means:

Metodologi ini sangat berguna untuk kumpulan data yang sangat besar—metodologi ini mengelompokkan kumpulan data yang tidak berlabel ke dalam berbagai jenis kelompok lainnya. Langkah pertama dalam proses ini adalah memilih sekelompok

klaster, yang dilambangkan dengan nilai "k". Sebagai ilustrasi, diagram di bawah ini merepresentasikan dua klaster yang berbeda:

Setelah Anda menentukan klaster-klaster ini, langkah selanjutnya adalah menghitung apa yang disebut "Centroid". Secara teknis, ini adalah titik tengah dari kedua klaster, yang diilustrasikan di bawah ini:



Akhirnya, algoritma khusus ini akan menghitung jarak rata-rata kedua Centroid, dan akan terus melakukannya secara iteratif hingga kedua Centroid mencapai titik konvergensi—yaitu, ketika batas kedua klaster benar-benar bertemu. Perlu dicatat bahwa teknik ini memiliki dua kelemahan:

- Teknik ini tidak bekerja dengan baik dengan dataset non-spherical;
- Mungkin terdapat beberapa klaster dengan banyak titik data di dalamnya, dan beberapa klaster dengan hampir tidak ada titik data sama sekali. Dalam kasus khusus ini, teknik ini tidak akan mendeteksi titik data yang terakhir.

Konsep Statistik Utama

Selain sisi matematika dari algoritma, Pembelajaran Mesin juga banyak menggunakan prinsip-prinsip statistik, dan beberapa prinsip terpenting yang digunakan dijelaskan dalam subbagian ini:

1) Deviasi Standar:
   Ini mengukur jarak rata-rata dari aspek statistik suatu dataset.
2) Distribusi Normal:

Ini adalah "kurva berbentuk lonceng" yang sudah sering kita dengar. Dalam istilah yang lebih teknis, ini merepresentasikan jumlah properti statistik dalam variabel dari semua dataset yang akan Anda gunakan untuk sistem Pembelajaran Mesin.

3) Teorema Bayes:
   Teorema ini memberikan informasi statistik yang detail tentang dataset Anda.
4) Korelasi:
   Di sinilah korelasi statistik atau kesamaan (atau bahkan asosiasi) ditemukan di antara semua dataset. Berikut prinsip-prinsip panduan di baliknya:
   - Lebih besar dari 0:
     Ini terjadi ketika suatu variabel meningkat dengan selisih satu. Akibatnya, variabel lain juga akan meningkat setidaknya sebesar satu nilai.
   - 0:
     Tidak ada korelasi statistik antara variabel mana pun dalam dataset.
   - Kurang dari 0:
     Ini terjadi ketika suatu variabel meningkat dengan selisih satu. Akibatnya, variabel lain juga akan menurun setidaknya sebesar satu nilai.

Sejauh ini, kami telah memberikan gambaran umum tingkat tinggi tentang aspek teoretis Pembelajaran Mesin. Di bagian selanjutnya buku ini, kita akan melakukan "Pendalaman".

2.3 PENDALAMAN ASPEK TEORETIS PEMBELAJARAN MESIN

Memahami Probabilitas

Jika Anda belum menyadarinya, salah satu pendorong utama di balik Pembelajaran Mesin adalah kualitas dan ketahanan set data yang Anda miliki untuk sistem yang Anda gunakan. Bahkan, bisa dibilang bahwa data tersebut merupakan sekitar 80 persen dari perjuangan untuk menjalankan sistem Pembelajaran Mesin Anda dan menghasilkan keluaran yang Anda butuhkan untuk proyek Anda. Jadi dalam hal ini, Anda mungkin akan lebih mengandalkan konsep statistika daripada matematika murni dan diskrit, karena set data Anda akan sangat bergantung pada hal ini.

Dalam bidang statistika, konsep probabilitas cukup sering digunakan. Probabilitas, dalam istilah yang lebih spesifik, adalah ilmu yang mencoba mengonfirmasi ketidakpastian suatu peristiwa, atau bahkan serangkaian peristiwa. Nilai "E" paling umum digunakan untuk merepresentasikan peristiwa tertentu, dan nilai P€ akan merepresentasikan tingkat probabilitas terjadinya peristiwa tersebut. Jika hal ini tidak benar-benar terjadi, maka disebut "Trail". Faktanya, banyak algoritma yang digunakan untuk Pembelajaran Mesin berasal dari prinsip-prinsip probabilitas dan model Bayesian naif.

Perlu dicatat bahwa terdapat tiga kategori spesifik untuk mendefinisikan probabilitas lebih lanjut, yaitu sebagai berikut:

1) Probabilitas Teoretis:
   Ini dapat didefinisikan sebagai jumlah cara terjadinya suatu peristiwa tertentu, yang secara matematis dibagi dengan jumlah total kemungkinan hasil yang sebenarnya

dapat terjadi. Konsep ini sangat sering digunakan untuk sistem Pembelajaran Mesin guna membuat prediksi yang lebih baik untuk masa depan, seperti memprediksi seperti apa Lanskap Ancaman Siber di masa mendatang.

2) Probabilitas Empiris:
   Ini menggambarkan jumlah spesifik berapa kali suatu peristiwa akan terjadi, yang kemudian dibagi secara matematis dengan jumlah total insiden yang juga mungkin terjadi.
3) Keanggotaan Kelas:
   Dalam hal ini, ketika suatu dataset tertentu ditetapkan dan diberi label, hal ini secara teknis dikenal sebagai "Pemodelan Prediktif Klasifikasi." Dalam hal ini, probabilitas bahwa suatu observasi tertentu akan benar-benar terjadi, seperti menetapkan dataset tertentu ke setiap kelas, dapat diprediksi. Hal ini memudahkan untuk menetapkan tujuan aktual dari apa yang akan dicapai oleh sistem Pembelajaran Mesin sebelum Anda memilih algoritma yang Anda perlukan.

Perlu dicatat bahwa klasifikasi probabilitas yang disebutkan di atas juga dapat dikonversi menjadi apa yang dikenal sebagai "Label Kelas Crisp." Untuk melakukan prosedur khusus ini, Anda perlu memilih dataset yang memiliki tingkat probabilitas tertinggi, serta yang dapat diskalakan melalui proses kalibrasi tertentu. Perlu diingat bahwa setidaknya 90 persen model Pembelajaran Mesin sebenarnya diformulasikan dengan menggunakan urutan spesifik dari berbagai algoritma iteratif.

Salah satu teknik yang paling umum digunakan untuk menyelesaikan tugas ini adalah yang dikenal sebagai "Algoritma Maksimalisasi Ekspektasi" yang paling cocok untuk mengelompokkan set data tanpa pengawasan. Dengan kata lain, teknik ini secara khusus meminimalkan perbedaan antara distribusi probabilitas yang diprediksi dan distribusi probabilitas yang diprediksi.

Seperti yang akan dibahas lebih lanjut di subbagian berikutnya, Optimasi Bayesian digunakan untuk apa yang dikenal sebagai "Optimasi Hiperparameter". Teknik ini membantu menemukan jumlah total kemungkinan hasil yang dapat terjadi untuk semua set data yang Anda gunakan dalam sistem Pembelajaran Mesin Anda. Selain itu, ukuran probabilistik dapat digunakan untuk mengevaluasi ketahanan algoritma ini. Salah satu teknik lain yang dapat digunakan dalam kasus ini dikenal sebagai "Kurva Karakteristik Operasi Penerima", atau disingkat "ROC".

Sebagai contoh, kurva ini dapat digunakan untuk memeriksa lebih lanjut tradeoff dari algoritma spesifik ini.

Teorema Bayesian

Inti dari perumusan semua jenis algoritma Pembelajaran Mesin adalah apa yang dikenal sebagai "Teori Probabilitas Bayesian". Dalam hal ini, tingkat ketidakpastian, atau risiko, dalam pengumpulan data sebelum memulai proses optimasi dikenal sebagai "Probabilitas Prior", dan pemeriksaan tingkat risiko ini setelah proses optimasi data selesai dikenal sebagai "Probabilitas Posterior". Hal ini juga dikenal secara lebih longgar sebagai "Teorema Bayes".

Ini secara sederhana menyatakan bahwa hubungan antara probabilitas suatu hipotesis sebelum mendapatkan bukti statistik apa pun (yang direpresentasikan sebagai $P[H]$) dan sesudahnya dapat dimasukkan ke dalam sistem Pembelajaran Mesin dengan menggunakan perhitungan matematika berikut:

$$\Pr(H|E) = \Pr(E|H) * \Pr(H)/\Pr(E)$$

Dalam dunia Pembelajaran Mesin, terdapat dua bidang statistika yang paling relevan, yaitu sebagai berikut:

1) Statistik Deskriptif:
   Ini adalah cabang statistika yang selanjutnya menghitung properti apa pun yang berguna dari kumpulan data Anda yang diperlukan untuk sistem Pembelajaran Mesin Anda. Ini sebenarnya melibatkan serangkaian perhitungan sederhana, seperti mencari nilai rata-rata, median, dan modus di antara semua kumpulan data Anda. Berikut ini:
   - Rata-rata: Ini adalah nilai rata-rata dari kumpulan data;
   - Modus: Ini adalah nilai yang paling sering muncul dalam kumpulan data Anda;
   - Median: Ini adalah nilai tengah yang secara fisik memisahkan separuh nilai yang lebih tinggi dalam kumpulan data Anda dari separuh nilai yang lebih rendah dalam kumpulan data Anda.
2) Statistik Inferensial:
   Pengelompokan statistik ini diimplementasikan ke dalam berbagai metode yang mendukung berbagai properti kuantifikasi dari kumpulan data yang Anda gunakan untuk sistem Pembelajaran Mesin Anda. Teknik-teknik spesifik ini digunakan untuk membantu mengkuantifikasi kemungkinan statistik dari setiap kumpulan data yang digunakan dalam membuat asumsi untuk proses formulasi model Pembelajaran Mesin.

Distribusi Probabilitas untuk Pembelajaran Mesin

Dalam Pembelajaran Mesin, hubungan statistik antara berbagai peristiwa dari apa yang dikenal sebagai "Variabel Acak Kontinu" dan probabilitas terkaitnya dikenal sebagai "Distribusi Probabilitas Kontinu." Kumpulan distribusi spesifik ini sebenarnya merupakan komponen kunci dari operasi yang dilakukan oleh model Pembelajaran Mesin dalam hal mengoptimalkan variabel masukan dan keluaran numerik.

Selain itu, probabilitas statistik suatu kejadian yang sama dengan atau kurang dari nilai tertentu yang telah ditentukan secara teknis dikenal sebagai "Fungsi Distribusi Kumulatif", atau disingkat "CDF". Kebalikan dari fungsi ini disebut "Fungsi Titik Persentase", atau disingkat "PPF". Dengan kata lain, Fungsi Kepadatan Probabilitas menghitung probabilitas statistik suatu hasil tertentu yang berkelanjutan, dan Fungsi Kepadatan Kumulatif menghitung probabilitas statistik bahwa suatu nilai yang kurang dari atau sama dengan hasil tertentu akan benar-benar terjadi dalam kumpulan data yang Anda gunakan dalam sistem Pembelajaran Mesin Anda.

Distribusi Normal

Distribusi Normal juga dikenal sebagai "Distribusi Gaussian". Premisnya adalah terdapat probabilitas statistik terjadinya suatu peristiwa waktu nyata dalam sistem

Pembelajaran Mesin Anda dari kumpulan data yang Anda miliki. Distribusi ini juga terdiri dari apa yang dikenal sebagai "Variabel Acak Kontinu", dan memiliki Distribusi Normal yang terbagi rata di antara kumpulan data Anda.

Lebih lanjut, Distribusi Normal didefinisikan dengan menggunakan dua parameter yang berbeda dan telah ditetapkan, yaitu Rata-rata (dilambangkan sebagai "mu") dan Varians (yang dilambangkan sebagai Sigma ^2). Selain itu, Deviasi Standar biasanya merupakan sebaran rata-rata dari rata-rata dan dilambangkan juga sebagai "Sigma". Distribusi Normal dapat direpresentasikan secara matematis sebagai berikut:

$$\mathrm{F(X)\ =\ 1/aSQRT\ PI\ \hat{}e\ -\ (u\ -\ x)\hat{}2/2O\hat{}2.}$$

Perlu dicatat juga bahwa rumus matematika ini dapat digunakan dalam berbagai Algoritma Pembelajaran Mesin untuk menghitung ukuran jarak dan penurunan gradien, yang juga mencakup "$\mathrm{K-Means}$" dan "$\mathrm{K-Nearest\ Neighbors}$". Terkadang, rumus yang disebutkan di atas perlu diskalakan ulang hingga distribusi statistik yang sesuai tercapai. Untuk melakukan proses penskalaan ulang, "$\mathrm{Normalisasi\ Z-Score}$" dan "$\mathrm{Transformasi\ Min-Max}$" digunakan.

Terakhir, dalam Algoritma Pembelajaran Mesin, variabel independen yang digunakan dalam kumpulan data Anda juga dikenal sebagai "Fitur". Variabel dependen juga dikenal sebagai "Keluaran".

Pembelajaran Terbimbing

Sebelumnya di bab ini, Pembelajaran Terbimbing telah diulas. Meskipun hanya memberikan gambaran umum tingkat tinggi, di subbagian ini, kita akan membahasnya lebih dalam. Perlu dicatat bahwa banyak algoritma Pembelajaran Mesin sebenarnya termasuk dalam kategori khusus ini. Secara umum, Pembelajaran Terbimbing bekerja dengan menggunakan variabel independen yang ditargetkan (bahkan bisa berupa serangkaian variabel dependen). Dari titik ini, fungsi matematika spesifik dapat dibuat yang dapat mengaitkan, atau memetakan, masukan dari kumpulan data dengan keluaran yang diinginkan atau diharapkan.

Ini adalah proses iteratif yang terus berlanjut hingga tingkat akurasi optimal tercapai, dan keluaran yang diinginkan juga memiliki hasil yang diharapkan. Berikut ini adalah contoh umum beberapa teknik statistik yang digunakan dalam proses iteratif ini:

1) <u>Regresi Linier:</u>

   Ini mungkin merupakan pendekatan terbaik untuk memperkirakan secara statistik nilai riil atau absolut apa pun yang didasarkan pada variabel kontinu yang ada dalam model Pembelajaran Mesin. Dengan teknik ini, hubungan linier (sesuai namanya) sebenarnya dibangun dan ditempatkan di antara variabel independen dan variabel dependen yang ada dalam model Pembelajaran Mesin. Secara teknis, ini dikenal sebagai "Garis Regresi", dan rumus matematikanya adalah sebagai berikut:

$$\mathrm{Y = a^{*}X + b.}$$

Dengan teknik pemodelan semacam ini, hubungan statistik sebenarnya dibuat dan difilter melalui berbagai Fungsi Prediktor Linier. Dari sini, parameter fungsi-fungsi tersebut kemudian diestimasi dari kumpulan data yang digunakan dalam sistem Pembelajaran Mesin. Meskipun Regresi Linier banyak digunakan dalam Pembelajaran Mesin, terdapat juga sejumlah kegunaan spesifik lainnya, yaitu sebagai berikut:

- Menentukan kekuatan prediktor, yang dapat menjadi tugas yang sangat subjektif untuk dicapai;
- Peramalan Tren, yang dapat digunakan untuk memperkirakan tingkat dampak dari setiap perubahan yang mungkin terjadi dalam kumpulan data;
- Memprediksi atau meramalkan peristiwa tertentu di masa mendatang. Misalnya, terkait Keamanan Siber, ini dapat digunakan untuk membantu memprediksi seperti apa varian vektor ancaman baru.

  Jika terdapat beberapa variabel independen yang digunakan (biasanya hanya ada satu, sebagaimana dilambangkan dengan nilai "Y" dalam persamaan di atas), maka teknik lain juga harus digunakan, termasuk Seleksi Maju, Eliminasi Bertahap, dan Eliminasi Mundur.

2) Regresi Logistik:

Teknik statistik ini digunakan untuk menentukan tingkat probabilitas keberhasilan dan kegagalan suatu hasil. Dengan demikian, variabel dependen yang ada harus dalam format biner, yaitu 0 atau 1. Teknik semacam ini dapat direpresentasikan secara matematis sebagai berikut:

$$\begin{aligned} \text{Odds} &= \text{p}/(1-\text{p}) \\ \text{Ln}(\text{odds}) &= \ln[\text{p}/(1-\text{p})] \\ \text{Logit}(\text{p}) &= \ln\ln[\text{p}/(1-\text{p})]. \end{aligned}$$

Perlu dicatat juga bahwa teknik ini juga menggunakan apa yang dikenal sebagai "Distribusi Binomial". Dengan kata lain, Fungsi Tautan harus dipilih untuk distribusi spesifik yang ada. Tidak seperti teknik yang disebutkan sebelumnya, tidak ada hubungan linier yang diperlukan. Lebih lanjut, teknik semacam ini sebagian besar digunakan untuk tujuan klasifikasi masalah pada sistem Pembelajaran Mesin.

3) Regresi Bertahap:

Seperti yang telah disebutkan sebelumnya, teknik semacam ini bekerja paling baik ketika terdapat beberapa variabel independen. Dalam hal ini, variabel-variabel independen ini dapat dioptimalkan lebih lanjut dengan alat-alat berikut:

- Metrik AIC;
- Uji-T;
- R Kuadrat, serta R Kuadrat yang Disesuaikan.

Salah satu manfaat utama teknik ini adalah Variabel Kovarian dapat ditambahkan satu per satu, tetapi permutasi untuk melakukannya harus ditentukan terlebih dahulu. Salah satu perbedaan utama antara Regresi Bertahap dan Regresi Maju adalah Regresi

Bertahap dapat menghilangkan semua jenis prediktor statistik, sementara Regresi Maju, "Prediktor Signifikan" dapat menambahkan variabel statistik tambahan apa pun yang diperlukan dalam pengembangan model Pembelajaran Mesin. Selain itu, Eliminasi Mundur memulai proses ini dengan semua prediktor statistik yang ada dalam model Pembelajaran Mesin, dan dari sana menghilangkan setiap variabel paling tidak signifikan yang muncul di seluruh siklus iteratif ini.

4) Regresi Polinomial:

Jika pangkat suatu variabel independen lebih besar dari satu (ini dapat direpresentasikan secara matematis sebagai "Y^1 > 1"), maka ini akan menjadi apa yang dikenal sebagai "Persamaan Regresi Polinomial". Hal ini dapat direpresentasikan secara matematis sebagai berikut:

$$Y = a + b^{*}Y\text{^}2.$$

5) Regresi Ridge:

Teknik ini secara khusus digunakan ketika kumpulan data yang digunakan untuk sistem Pembelajaran Mesin mengalami transformasi yang dikenal sebagai "Multikolinearitas". Hal ini biasanya terjadi ketika variabel independen sangat berkorelasi, atau berasosiasi, satu sama lain, dan dari sana, perhitungan Kuadrat Terkecil tetap berada pada titik netral atau tidak berubah.

Untuk mengatasi efek Multikolinearitas, sejumlah bias statistik ditambahkan untuk membantu mengurangi Kesalahan Standar atau jenis deviasi statistik lainnya yang mungkin terjadi dalam model Pembelajaran Mesin. Efek Multikolinearitas dapat direpresentasikan secara matematis sebagai berikut:

$$Y = a + y\,a + b1x1 + b2x2 +, b3x3, \text{dan seterusnya.}$$

Dalam teknik ini juga, "Metode Regularisasi" dapat digunakan untuk memastikan bahwa nilai koefisien yang terdapat dalam rumus di atas tidak akan pernah mencapai nol selama sistem Pembelajaran Mesin digunakan.

6) Penyusutan Absolut Terkecil & Regresi Operator Selektor (alias "Regresi Laso"):

Teknik khusus ini memiliki kemampuan untuk mengurangi variabilitas statistik apa pun yang ada dalam model Pembelajaran Mesin, dengan mengurangi jumlah variabilitas yang ada. Teknik ini juga dapat dianggap sebagai teknik optimasi atau "regularisasi" di mana hanya satu opsi statistik yang dipilih dari kumpulan prediktor agregat. Teknik ini juga dapat membuat prediksi masa depan jauh lebih akurat.

Pertanyaan mendasar yang sering diajukan pada titik ini adalah jenis Teknik Regresi apa yang harus digunakan untuk model Pembelajaran Mesin? Aturan praktisnya adalah jika keluarannya harus kontinu (atau linier), maka Regresi Linier harus digunakan. Namun, jika keluarannya bersifat multi-opsi, misalnya biner, maka model Regresi Biner atau Regresi Logistik harus digunakan. Namun, ada faktor-faktor lain yang perlu dipertimbangkan, yang meliputi:

- Jenis variabel independen dan dependen yang digunakan;
- Karakteristik dataset yang digunakan serta dimensionalitas matematisnya.

Pohon Keputusan

Tinjauan umum tentang Pohon Keputusan telah diberikan sebelumnya dalam bab ini, dan di subbagian ini, kita akan membahasnya lebih lanjut. Teknik ini sebenarnya dianggap sebagai bagian dari Pembelajaran Terbimbing. Tujuan akhir dari Pohon Keputusan adalah untuk menciptakan model Pembelajaran Mesin yang berpotensi memprediksi nilai tertentu dari variabel target dengan mempelajari aturan keputusan, atau permutasi, yang telah diterapkan sebelumnya ke dalam dataset, untuk menciptakan lingkungan pembelajaran yang lebih efektif bagi sistem Pembelajaran Mesin.

Perlu dicatat bahwa Pohon Keputusan juga dapat disebut "Pohon Klasifikasi dan Regresi," atau disingkat "CART". Dalam situasi khusus ini, kemampuan untuk memprediksi nilai variabel target diciptakan oleh apa yang dikenal sebagai "Pernyataan Jika/Maka." Beberapa atribut Pohon Keputusan meliputi:

1) Atribut:
   Ini adalah kuantitas numerik yang menggambarkan nilai suatu instans.
2) Instans:
   Ini adalah atribut yang selanjutnya mendefinisikan ruang masukan dan juga disebut sebagai "Vektor Fitur".
3) Sampel:
   Ini adalah himpunan masukan yang dikaitkan dengan atau dikombinasikan dengan label tertentu. Ini kemudian dikenal sebagai "Kumpulan Pelatihan".
4) Konsep:
   Ini adalah fungsi matematika yang mengaitkan atau memetakan masukan tertentu ke keluaran tertentu.
5) Konsep Target:
   Ini dapat dianggap sebagai keluaran yang telah memberikan hasil atau keluaran yang diinginkan.
6) Kelas Hipotesis:
   Ini adalah himpunan atau kategori kemungkinan hasil.
7) Himpunan Pengujian:
   Ini adalah sub-teknik yang digunakan untuk lebih mengoptimalkan kinerja "Konsep Kandidat".
8) Konsep Kandidat:
   Ini juga disebut sebagai "Konsep Target".

Contoh grafis Pohon Keputusan telah diberikan sebelumnya dalam bab ini. Perlu dicatat bahwa teknik ini juga memanfaatkan fungsi Boolean, operator matematika AND OR XOR, serta gerbang Boolean.

Langkah-langkah spesifik untuk membuat Pohon Keputusan berbasis Pembelajaran Mesin adalah sebagai berikut:

Dapatkan kumpulan data yang akan dibutuhkan dan dari sana hitung ketidakpastian statistik untuk masing-masing kumpulan data tersebut;

- Buat daftar pertanyaan yang harus ditanyakan pada setiap simpul tertentu dari Pohon Keputusan;
- Setelah pertanyaan dirumuskan, buat baris "Benar" dan "Salah" yang diperlukan;
- Hitung informasi yang telah diperoleh dari partisi yang terjadi pada langkah sebelumnya;
- Selanjutnya, perbarui pertanyaan yang diajukan dari hasil proses yang telah dikumpulkan pada langkah terakhir;
- Terakhir, bagi, dan jika perlu, bagi lagi simpul-simpulnya dan terus ulangi proses iteratif ini hingga Anda menyelesaikan tujuan Pohon Keputusan dan dapat digunakan untuk sistem Pembelajaran Mesin.

Perlu dicatat juga bahwa dalam Pembelajaran Mesin, Bahasa Pemrograman Python digunakan secara luas. Hal ini akan dibahas lebih lanjut, tetapi di bawah ini memberikan contoh bagaimana bahasa pemrograman Python dapat digunakan dalam pembuatan Pohon Keputusan juga:

```
Import numpy as np
Import pandas as pd
From sklearn.metrics import confusion_matrix
From sklearn.cross -validation import train_test_split
From sklearn.tree import DecisionTreeClassifier
From sklearn.metrics import accuracy_score
From sklearn.metrics import classification_report

# Function importing data set
Def importdata ();
       Balance_data = pd.read_csv(
#Printing the dataswet shape
Print (“data set Length:”,len(balance_data)
Print (“data set Shape: ”, balance_data.shape)

#Printing the data set observations
Print “[data set: ”, balance _data.head()]
Return balance_data
#Function to split the data set
Def splitdata set(balance_data):

#Separating the target variable

X = balance_data.values [:, 1:5]
Y = balance_data.values[:, 0]
#Splitting the dataset into train and test
X_train, X-test, y_train, y_test, = train_test_split(
X, Y, test_size = 0.3, random_state = 100)

Return X, Y, X _train; X_test; y_train, Y_test
#Function to perform training with giniIndex.
Def train_using_gini(X_train. X_test, y_train);

#Creating the classifier object
```

```
Of_gini = DecisionTreeClassifer(criterion = “gin”,

Random_state = 100,max_depth=3,
Min_samples_leaf=5)
#Performing training
Cif_gini.fit(X_train, y_train)
Retrn cif_gini
#Function to perform training with entropy.
Def tarin_using_entropy(X_train, X_test, y_train);

#Decision tree with entropy
Clf_entropy_= DecisionTreeClassifier(
        Criterion = “entropy”, random_state#

100,
Max_depth= 3, min _samples_leaf =5)
#Performing training
Clf_enrtropy.fit(X_train, y_train)
Return clf_entropy

#Function to make predictions
Def prediction(X_test, clf_object):
#Prediction on test with giniIndex
Y_pred = clf_object.predict(X_test)

Print(“Predicted values:”)
Return y_pred

#Function to compute accuracy
Def cal_accuracy(y_test, y_pred):
    Print(“Confusion Matrix: ”;
              Confusion_matrix(y_test, y_pred)
Print (“Accuracy :”
Accuracy_score(y_test, y_pred)*100)
Print (“Report : ”,
Classification_report(y_test, y_pred)

#Driver code
Def():

#Building Phase
Data = importdata()
X, Y, X_train, X_test, y_train, y_test = splitdata set(data)
Clf_gini = train_using_gini(X_train, X_test, y_train)
Clf_entropy = train_using_entropy(X_train, X_test, y_train)

#Operational Phase
Print(“Results Using Gini Index:”)
#Prediction using gini
Y_pred_gini = prediction(X_test, clf_gini)
Cal_accuracy(y_test, y_pred_gini)

Print(“Results Using Entropy:”)
     #Prediction using entropy
Y_pred_entropy = prediction(X_test, clf_entropy)
    Cal_accuracy(y_test, y_pred_entropy)

#Calling amin function
If_name_==”_main_:”
```

```
       Main()

(Sharma, n.d.)
(SOURCE: 2).
```

Masalah Overfitting pada Pohon Keputusan

Setelah Pohon Keputusan selesai, salah satu kelemahan utamanya adalah sangat rentan terhadap apa yang dikenal sebagai "Overfitting". Ini berarti terdapat lebih banyak dataset daripada yang dibutuhkan untuk sistem Pembelajaran Mesin; oleh karena itu, optimasi lebih lanjut diperlukan untuk mendapatkan hasil yang diinginkan. Untuk mencegah fenomena ini terjadi, Anda perlu mempelajari dengan saksama cabang-cabang pada Pohon Keputusan yang dianggap kurang penting.

Dalam hal ini, cabang-cabang, atau simpul-simpul spesifik tersebut, perlu dihilangkan. Proses ini juga disebut "Post Pruning", atau singkatnya "Pruning". Dalam hal ini, terdapat dua teknik yang lebih spesifik, yaitu sebagai berikut:

1) Kesalahan Minimum:
   Dalam hal ini, Pohon Keputusan dipangkas kembali ke titik di mana Kesalahan Validasi Silang berada pada titik minimumnya.
2) Pohon Terkecil:
   Dalam hal ini, Pohon Keputusan dikurangi bahkan lebih dari nilai Kesalahan Minimum yang telah ditetapkan. Akibatnya, proses ini akan menghasilkan Pohon Keputusan dengan Kesalahan Validasi Silang yang berjarak setidaknya satu Standar Deviasi dari Kesalahan Minimum.

Namun, sangat penting untuk memeriksa Overfitting saat Anda membangun Pohon Keputusan. Dalam hal ini, Anda dapat menggunakan apa yang dikenal sebagai "Heuristik Penghentian Awal".

Hutan Acak

Hutan Acak adalah kombinasi dari banyak Pohon Keputusan, bahkan mungkin dalam kisaran minimal ratusan atau bahkan ribuan. Setiap pohon dilatih dan disimulasikan dengan cara yang sedikit berbeda satu sama lain. Setelah Hutan Acak selesai dan dioptimalkan, keluaran akhir dihitung oleh sistem Pembelajaran Mesin dalam proses yang dikenal sebagai "Rata-Rata Prediktif".

Dengan Hutan Acak, kumpulan data dibagi menjadi subset yang jauh lebih kecil yang didasarkan pada fitur spesifiknya, dan yang juga hanya berada di bawah satu Jenis Label tertentu. Hutan Acak juga memiliki pemisahan statistik tertentu dengan ukuran statistik yang dihitung pada masing-masing subset dari dalam Pohon Keputusan.

Bagging

Bagging yang juga dikenal sebagai "Agregasi Bootstrap" Ini adalah pendekatan khusus yang digunakan untuk menggabungkan prediksi dari berbagai sistem Pembelajaran Mesin yang Anda gunakan dan menyatukannya dengan tujuan tunggal untuk mencapai Prediksi Mode yang lebih akurat daripada sistem lain yang sedang digunakan. Karena itu, Pohon Keputusan

dapat sangat sensitif secara statistik terhadap kumpulan data spesifik tempat mereka dilatih dan dioptimalkan.

Bagging juga dapat dianggap sebagai subset lebih lanjut dalam arti bahwa hal ini biasanya diterapkan pada algoritma Pembelajaran Mesin yang dianggap memiliki "Varians Tinggi". Pohon Keputusan yang dibuat dari Agregasi Bootstrap juga dapat sangat sensitif terhadap kumpulan data yang digunakan untuk tugas-tugas yang telah dilatih. Alasan utamanya adalah bahwa setiap perubahan kecil atau inkremental dapat secara drastis mengubah komposisi dan susunan struktur Pohon Keputusan.

Dengan teknik Bagging, kumpulan data sebenarnya tidak dibagi lagi; Sebaliknya, setiap simpul Pohon Keputusan dikaitkan dengan sampel spesifik dari kumpulan data yang dimaksud. Ukuran acak biasanya ditetapkan. Hal ini sangat berbeda dengan Pohon Keputusan yang lebih ternormalisasi, di mana keacakan biasanya terjadi ketika simpul spesifik tersebut dibagi lagi, dan dari sana, tingkat pemisahan statistik yang lebih tinggi dapat dicapai.

Pertanyaan yang biasanya muncul pada titik ini adalah, mana yang lebih baik: Hutan Acak, atau memanfaatkan beberapa Pohon Keputusan yang tidak saling terkait atau terhubung satu sama lain? Dalam kebanyakan kasus, pilihan yang pertama jauh lebih baik, karena Teknik Penggabungan yang lebih baik, serta berbagai jenis algoritma Pembelajaran Mesin lainnya, juga dapat digunakan, dan digabungkan menjadi satu unit yang kohesif.

Metode Naïve Bayes

Ini adalah teknik umum yang biasanya digunakan untuk skenario Pemodelan Prediktif oleh sistem Pembelajaran Mesin. Perlu dicatat bahwa dengan Pembelajaran Mesin, komputasi dilakukan pada kumpulan data tertentu di mana hipotesis statistik terbaik harus ditentukan untuk menghasilkan keluaran yang diinginkan. Metode Naïve Bayes dapat direpresentasikan secara matematis sebagai berikut:

$$P(h|d) = [P(d|h) * P(h))/P(d)]$$

Di mana:

$P(h|d)$ = adalah probabilitas statistik dari suatu hipotesis tertentu (dikenal sebagai "h") yang dihitung pada kumpulan data tertentu (dikenal sebagai "d").

$P(d|h)$ = adalah probabilitas kumpulan data "d", dengan asumsi hipotesis "h" benar secara statistik.

$P(d)$ = adalah probabilitas kumpulan data yang tidak memiliki hipotesis apa pun ("h") atau kumpulan data "d" apa pun.

Dalam hal ini, jika semua hal di atas juga benar, maka dapat disimpulkan bahwa hipotesis "h" juga benar. Apa yang dikenal sebagai "Probabilitas Posterior" juga terkait dengan konsep ini.

Metodologi di atas juga dapat digunakan untuk menghitung "Probabilitas Posterior" untuk sejumlah hipotesis statistik tertentu. Tentu saja, yang memiliki tingkat probabilitas tertinggi akan dipilih untuk Sistem Pembelajaran Mesin karena dianggap paling berhasil dan

paling robust. Namun, jika muncul situasi di mana semua tingkat hipotesis statistik memiliki nilai yang sama, maka hal ini dapat direpresentasikan secara matematis sebagai berikut:

$$\mathrm{MAP(h)} = \mathrm{maks[(P(d|h)).}$$

Perlu juga disebutkan bahwa metodologi ini terdiri dari algoritma lain yang dikenal sebagai "Klasifikasi Naif Bayes". Teknik ini biasanya digunakan untuk menentukan dan memastikan apakah suatu nilai statistik tertentu bersifat Kategorikal atau Biner berdasarkan rancangannya. Probabilitas Kelas dan himpunan kondisional terkaitnya juga dikenal sebagai "representasi" dari Model Naïve Bayes. Probabilitas Kelas juga merupakan peluang statistik setiap kelas yang ada dalam himpunan data; Probabilitas Kondisional ditentukan dari nilai input yang diberikan untuk setiap Kelas Nilai dari himpunan data yang digunakan dalam sistem Pembelajaran Mesin.

Pertanyaan umum lain yang biasanya ditanyakan pada tahap ini adalah, bagaimana Teorema Naïve Bayes sebenarnya bekerja, setidaknya pada tingkat tinggi? Nah, pertama-tama kita perlu menghitung Probabilitas Posterior (yang dilambangkan sebagai $\mathrm{P(c|x)}$ dari $\mathrm{P©}$, $\mathrm{P(X)}$, dan $\mathrm{PX|C)}$. Dengan kata lain, fondasi algoritma ini dapat direpresentasikan secara matematis sebagai berikut:

$$\mathrm{P(c|x)} = \mathrm{P(x|c)P©/P(x)}$$

Di mana:

$\mathrm{P(c|x)}$ = Probabilitas Posterior;
$\mathrm{P(x|c)}$ = Kemungkinan Statistik;
$\mathrm{P©}$ = Probabilitas Prior Kelas;
$\mathrm{P(x)}$ = Probabilitas Prior Prediktor.

Dengan representasi matematis di atas, kelas spesifik yang memiliki tingkat Probabilitas Posterior statistik tertinggi kemungkinan besar akan menjadi kandidat yang akan digunakan dalam menghitung keluaran akhir dari sistem Pembelajaran Mesin.

Keunggulan Metode Naïve Bayes adalah sebagai berikut:

- Merupakan salah satu algoritma yang paling banyak digunakan dalam Pembelajaran Mesin hingga saat ini;
- Memberikan hasil yang sangat robust untuk semua jenis prediksi Multi-Kelas;
- Membutuhkan pelatihan yang jauh lebih sedikit dibandingkan beberapa metode lain yang baru saja diulas;
- Paling cocok untuk tujuan Prediksi Real-Time, terutama untuk tujuan Keamanan Siber dalam hal penyaringan positif palsu;
- Dapat memprediksi probabilitas statistik berbagai Kelas Ganda dari variabel yang ditargetkan;
- Dapat digunakan untuk tujuan klasifikasi teks (biasanya ketika dataset tidak bersifat kuantitatif, melainkan kualitatif;

- Dengan pendekatan penyaringan yang dimilikinya, metode ini dapat dengan mudah menemukan tren tersembunyi jauh lebih cepat daripada metode lain yang telah diulas sebelumnya.

Kelemahan Metode Naïve Bayes adalah sebagai berikut:

- Tidak efisien untuk memprediksi kelas dataset uji;
- Jika ada Metode Transformasi yang digunakan, metode ini tidak dapat mengonversi dataset menjadi kurva Distribusi Normal Standar;
- Tidak dapat menangani Fitur Berkorelasi tertentu karena dianggap sebagai overhead dalam hal daya pemrosesan pada sistem Pembelajaran Mesin;
- Tidak ada Teknik Minimisasi Varians yang digunakan, sehingga tidak dapat memanfaatkan "Teknik Bagging";
- Memiliki kumpulan yang sangat terbatas untuk Penyetelan Parameter;
- Asumsi lebih lanjut adalah bahwa setiap fitur unik dalam setiap dataset yang ada dan digunakan untuk sistem Pembelajaran Mesin tidak terkait satu sama lain, sehingga tidak akan memiliki dampak positif apa pun pada fitur lain yang mungkin ada dalam kumpulan data.

Algoritma KNN

Algoritma ini juga dikenal sebagai algoritma "K Nearest Neighbors". Algoritma ini juga dianggap sebagai algoritma Pembelajaran Mesin Terawasi. Algoritma ini biasanya digunakan oleh sistem Pembelajaran Mesin untuk menyelesaikan skenario Klasifikasi dan Regresi secara spesifik. Algoritma ini sangat banyak digunakan karena memiliki dua sifat yang berbeda, tidak seperti yang telah dibahas sebelumnya. Sifat-sifat tersebut adalah sebagai berikut:

1) Algoritma "Malas":
   Algoritma ini malas dalam artian algoritma ini tidak memiliki segmen pelatihan khusus yang terkait dengannya, sehingga memanfaatkan semua kumpulan data yang tersedia saat pelatihan dalam Fase Klasifikasi.
2) Algoritma ini bersifat Non-Parametrik:
   Ini berarti bahwa algoritma spesifik ini tidak pernah membuat asumsi apa pun tentang kumpulan data yang mendasarinya.

Untuk mengimplementasikan Algoritma KNN sepenuhnya pada semua jenis sistem Pembelajaran Mesin, langkah-langkah berikut harus diambil:

1) Deploy dataset, dan inisialisasi nilai "K" ke himpunan yang telah ditentukan sebelumnya dari total jumlah tetangga terdekat yang ada. Selain itu, dataset pelatihan dan dataset pengujian lainnya juga harus di-deploy.
2) Penting juga untuk menghitung nilai "K" serta jarak antara dataset pelatihan dan dataset uji.
3) Dari setiap titik dalam dataset uji, Anda juga perlu menghitung jarak antara dataset uji serta setiap baris untuk setiap dataset pelatihan.
4) Setelah langkah di atas selesai, urutkan nilai "K" dalam format urutan menaik berdasarkan nilai jarak yang telah dihitung sebelumnya. Dari titik ini, pilih baris "K" teratas, dan tetapkan kelas tertentu untuknya.

5) Terakhir, dapatkan Label yang telah ditetapkan untuk entri "K" yang baru saja Anda pilih.

Keunggulan utama lain dari Algoritma KNN adalah tidak adanya proses pembelajaran yang biasanya diperlukan, sehingga sangat mudah untuk diperbarui ketika dataset baru tersedia. Algoritma ini juga dapat menyimpan dataset dalam bentuk lain dengan mengambil struktur dataset yang kompleks dan mencocokkan pola pembelajaran baru saat mencoba memprediksi nilai dari berbagai keluaran. Dengan demikian, jika ada jenis prediksi baru yang harus dibuat untuk keluaran tersebut, Algoritma ini dapat menggunakan dataset pelatihan yang sudah ada.

Seperti yang telah disinggung sebelumnya, berbagai jarak harus dihitung untuk Algoritma KNN. Yang paling umum digunakan adalah apa yang dikenal sebagai "Jarak Euclidean", yang direpresentasikan oleh rumus matematika berikut:

$$\text{Euclidean Distance}(X, Xi) = \text{SQRT}[(\text{jumlah}((X) - (\text{xij}\text{^}2)]$$

Perlu dicatat juga bahwa rumus jarak lainnya juga dapat digunakan, terutama Jarak Kosinus. Selain itu, kompleksitas komputasi Algoritma KNN juga dapat meningkat seiring dengan ukuran dataset pelatihan. Ini berarti terdapat hubungan statistik positif: seiring bertambahnya ukuran, kompleksitasnya juga akan meningkat.

Seperti yang telah disebutkan, Python sangat sering digunakan untuk Pembelajaran Mesin, dan kode berikut dapat digunakan untuk memprediksi keluaran yang akan dihasilkan oleh Algoritma KNN:

```
Knn_predict <-function(test, train, k_value){
Pred <-c()
#LOOP – 1
For(I in c(1:row(test))){
       Dist = c()
       Char = c()
       Setosa = 0
       Versicolor = 0
       Virginica = 0

}
#LOOP – 2 – looping over trained data
For (j in c(1:row(train))){}
Dist <-c(dist, ED(test[I,], train [j,]))
Char <-c(char, as.character(train[j,][[5]]))
Df <-data.frame(char, dist$SepallLength)
Df <-df[order(df$dist.SepallLength),]
#sorting dataframe
     Df<-df[1:k_value,]

#Loop3: loops over df and counts classes of all neighbors
    For(k in c(1:nrow(df))){
If(as.character(df[k, “char”]) = = “setoasa”){
Setosa = setosa + 1
}else if(as.character(df[k,, “char]) = =
“versicolor”){
        Versicolor = versicolor + 1
```

```
        }else
        Virginica = virginica +1
        }
N<-table(df$char)
Pred = names(n)[which(n==max(n))]
Return(pred) #return prediction vector
}
#Predicting the value for K=1
K=1
Predictions <-knn_predict(test, train, K)
Output:
For K=1

[1]”Iris-virginica
(SOURCE: 2).
```

2.4 PEMBELAJARAN TANPA SUPERVISI

Dalam Pembelajaran Mesin, Algoritma Tanpa Supervisi dapat digunakan untuk membuat inferensi dari kumpulan data yang terdiri dari data masukan jika tidak memiliki Respons Berlabel yang terkait dengannya. Dalam kategori ini, berbagai model yang digunakan (dan yang akan dibahas lebih detail) menggunakan tipe data masukan [X], dan lebih lanjut, tidak memiliki asosiasi apa pun dengan nilai keluaran yang dihitung.

Hal ini membentuk dasar untuk Pembelajaran Tanpa Supervisi, terutama karena tujuan model adalah untuk menemukan dan merepresentasikan tren tersembunyi tanpa siklus pembelajaran sebelumnya. Dalam hal ini, terdapat dua kategori utama: Pengelompokan dan Asosiasi.

1) Pengelompokan:

   Hal ini biasanya terjadi ketika kelompok inheren harus ditemukan dalam kumpulan data. Namun dalam kategori ini, sistem Pembelajaran Mesin harus menangani sejumlah besar kumpulan data besar, yang sering disebut sebagai "Data Besar". Dengan Pengelompokan, tujuannya adalah untuk menemukan semua asosiasi (yang tersembunyi dan tidak tersembunyi) dalam kumpulan data besar ini. Berikut ini adalah jenis-jenis utama Properti Pengelompokan yang sangat sering digunakan saat ini dalam Pembelajaran Mesin:

   - Pengelompokan Probabilistik:
     Ini melibatkan pengelompokan berbagai dataset ke dalam klasternya masing-masing berdasarkan skala probabilistik yang telah ditentukan sebelumnya.
   - Pengelompokan K-Means:
     Ini melibatkan pengelompokan semua dataset ke dalam klaster yang saling eksklusif secara statistik sebanyak "K".
   - Pengelompokan Hirarkis:
     Ini mengklasifikasikan dan mengategorikan titik-titik data spesifik di semua dataset ke dalam apa yang dikenal sebagai "Kluster Induk-Anak".
   - Model Campuran Gaussian:
     Ini terdiri dari Komponen Kepadatan Multivariat dan Normal.

- Model Markov Tersembunyi:
  Teknik ini digunakan untuk menganalisis semua dataset yang digunakan oleh sistem Pembelajaran Mesin, serta untuk menemukan status sekuensial yang mungkin ada di antara dataset tersebut.
- Peta Pengorganisasian Mandiri:
  Ini memetakan berbagai struktur Jaringan Saraf Tiruan yang dapat mempelajari Distribusi Statistik serta Topologi kumpulan data.

Model Generatif

Model-model ini merupakan bagian terbesar dari Model Pembelajaran Tanpa Pengawasan (Unsupervised Learning Models). Alasan utamanya adalah karena model-model ini dapat menghasilkan sampel data baru dari distribusi yang sama dari setiap set data pelatihan yang telah ada. Model-model ini dibuat dan diimplementasikan untuk mempelajari data tentang set data tersebut. Hal ini sering disebut sebagai "Metadata".

Kompresi Data

Ini mengacu pada proses untuk menjaga set data sekecil mungkin. Ini murni upaya untuk menjaganya tetap semulus dan seefisien mungkin agar tidak menguras daya pemrosesan sistem Pembelajaran Mesin. Hal ini sering dilakukan melalui apa yang dikenal sebagai "Proses Reduksi Dimensionalitas". Teknik lain yang dapat digunakan dalam hal ini termasuk "Dekomposisi Nilai Singular" dan "Analisis Komponen Utama".

Dekomposisi Nilai Singular secara matematis memfaktorkan set data menjadi produk dari tiga set data lainnya, menggunakan konsep Aljabar Matriks. Dengan Analisis Komponen Utama, berbagai Kombinasi Linear digunakan untuk menemukan varians statistik spesifik di antara semua set data.

Asosiasi

Sesuai namanya, ini sebenarnya adalah metodologi Pembelajaran Mesin Berbasis Aturan yang dapat digunakan untuk menemukan hubungan tersembunyi maupun tidak tersembunyi di semua set data. Untuk mencapai hal ini,"Aturan Asosiasi" biasanya diterapkan. Aturan ini terdiri dari konsekuen dan anteseden. Contoh penerapannya ditunjukkan pada matriks di bawah ini:

Jumlah Frekuensi	***Barang yang Ada***
1	Roti, Susu
2	Roti, Biskuit, Minuman, Telur
3	Susu, Biskuit, Minuman, Diet Coke
4	Roti, Susu, Biskuit, Diet Coke
5	Roti, Susu, Diet Coke, dan Coke

Ada dua properti yang sangat penting untuk diperhatikan di sini:

- Jumlah Dukungan:

Ini adalah jumlah aktual untuk frekuensi kemunculan dalam setiap himpunan yang ada dalam matriks di atas. Misalnya, $[(\text{Susu}, \text{Roti}, \text{Biskuit})] = 2$. Di sini, representasi matematisnya dapat diberikan sebagai berikut:

$X -> Y$, di mana nilai X dan Y dapat berupa dua himpunan apa pun dalam matriks di atas. Misalnya, $(\text{Susu}, \text{Biskuit}) -> (\text{Minuman})$.

- Item yang Sering Muncul:

Ini adalah himpunan statistik yang ada ketika nilainya sama dengan atau bahkan lebih besar dari ambang batas minimum himpunan data. Dalam hal ini, ada tiga metrik utama yang perlu diperhatikan:

1) Dukungan:

Metrik spesifik ini menjelaskan seberapa sering suatu Himpunan Item benar-benar muncul dalam semua transaksi pemrosesan data. Rumus matematika untuk menghitung tingkat kemunculan ini adalah sebagai berikut:

$\text{Dukungan}[(X) \rightarrow (Y)] = \text{transaksi yang berisi X dan Y/Jumlah total transaksi.}$

2) Keyakinan:

Metrik ini digunakan untuk mengukur kemungkinan statistik suatu kejadian yang memiliki efek konsekuensial berikutnya. Rumus matematika untuk menghitungnya adalah sebagai berikut:

$\text{Keyakinan}[(X) \rightarrow (Y)] = \text{total transaksi yang mengandung X dan Y/}$
$\text{Transaksi yang mengandung X.}$

3) Lift:

Metrik ini digunakan untuk mendukung secara statistik frekuensi aktual suatu konsekuen yang darinya sifat kondisional kejadian (Y) mengingat keadaan (X) dapat dihitung. Lebih spesifik, ini dapat didefinisikan sebagai kenaikan statistik dalam tingkat probabilitas pengaruh (Y) terhadap (X). Rumus matematika untuk menghitungnya adalah sebagai berikut:

$$\text{Lift}[(X) \rightarrow (Y)] = (\text{Total transaksi yang mengandung X dan Y})$$
$$*)\text{Transaksi yang mengandung X})/\text{Total fraksi transaksi yang mengandung Y.}$$

Perlu dicatat bahwa Aturan Asosiasi sangat bergantung pada penggunaan pola data serta ko-kemunculan statistik. Seringkali dalam situasi ini, pernyataan "Jika/Maka" digunakan. Terdapat tiga algoritma Pembelajaran Mesin lainnya yang termasuk dalam kategori ini, yaitu:

1) Algoritma AIS:

Dengan ini, sistem Pembelajaran Mesin dapat memindai dan memberikan jumlah total dataset yang dimasukkan ke dalam sistem Pembelajaran Mesin.

2) Algoritma SETM:

Ini digunakan untuk lebih mengoptimalkan transaksi yang terjadi dalam dataset saat diproses oleh sistem Pembelajaran Mesin.

3) Algoritma Apriori:

Ini memungkinkan Item Kandidat ditetapkan sebagai variabel spesifik yang dikenal sebagai "S" untuk menghasilkan hanya jumlah dukungan yang diperlukan untuk Item Besar yang berada dalam dataset.

Estimasi Kepadatan

Ini dianggap sebagai hubungan statistik antara jumlah total observasi dan tingkat probabilitasnya. Perlu dicatat di sini bahwa dalam hal keluaran yang diperoleh dari sistem Pembelajaran Mesin, probabilitas kepadatan dapat bervariasi dari tinggi hingga rendah, dan apa pun di antaranya.

Namun, untuk memastikan hal ini sepenuhnya, kita juga perlu menentukan apakah suatu observasi statistik akan benar-benar terjadi atau tidak.

Fungsi Kepadatan Kernel

Fungsi matematika ini digunakan untuk mengestimasi lebih lanjut probabilitas statistik Variabel Kontinu yang benar-benar muncul dalam kumpulan data. Dalam hal ini, semua Fungsi Kernel yang ada dibagi secara matematis dengan total Fungsi Kernel, baik yang benar-benar ada maupun tidak. Hal ini dimaksudkan untuk memberikan jaminan bahwa Fungsi Kepadatan Probabilitas tetap bernilai non-negatif, dan untuk memastikan bahwa fungsi tersebut akan tetap menjadi integral matematis atas kumpulan data yang digunakan oleh sistem Pembelajaran Mesin.

Kode sumber Python untuk fungsi ini adalah sebagai berikut:

```
For I = 1 to n:
For all X;
Dens(X)
+ = (1/n) * (1/w) *K[(x-Xi)/w]
```

Di mana:

- Input = Fungsi Kernel $\mathrm{K(x)}$, dengan Lebar Kernel W, yang terdiri dari Instansi Data $\mathrm{x1}$ dan xN.
- Output = Fungsi Kepadatan Probabilitas yang diestimasi yang mendasari dataset pelatihan.
- Proses: Ini menginisialisasi $\mathrm{Dens(X)} = 0$ di semua titik "X" yang muncul dalam dataset.

Variabel Laten

Variabel-variabel ini dianggap sebagai variabel yang disimpulkan secara statistik dari variabel lain dalam dataset yang tidak memiliki korelasi langsung satu sama lain. Variabel-variabel semacam ini tidak digunakan dalam set pelatihan, dan pada dasarnya tidak kuantitatif. Sebaliknya, bersifat kualitatif.

Model Campuran Gaussian

Model ini juga dianggap sebagai model Variabel Laten. Model ini banyak digunakan dalam aplikasi Pembelajaran Mesin karena dapat menghitung jumlah total data dalam dataset, termasuk yang berisi Klaster. Masing-masing yang terakhir dapat direpresentasikan lebih lanjut sebagai $\mathrm{N1}, \ldots \mathrm{NK}$, tetapi distribusi statistik yang terdapat di dalamnya dianggap sebagai Campuran Gaussian secara alami.

2.5 PERCEPTRON

Seperti yang mungkin telah Anda simpulkan, mungkin salah satu tujuan terbesar Kecerdasan Buatan, Pembelajaran Mesin, dan Jaringan Syaraf Tiruan adalah untuk memodelkan proses otak manusia. Tentu saja, kita tahu bahwa otak manusia sangat rumit, dan kita mungkin baru memahami 1 persennya. Sejujurnya, kita tidak akan pernah sepenuhnya memahami otak manusia, dan jika kita sampai pada titik itu, dapat dikatakan bahwa hal itu masih akan terjadi berabad-abad lagi.

Seperti yang kita ketahui, Unit Pemrosesan Pusat (CPU) adalah komponen pemrosesan utama komputer. Namun, jika disamakan dengan tingkat otak, maka padanannya adalah apa yang disebut "Neuron". Hal ini akan dibahas lebih detail pada bab yang membahas Jaringan Syaraf Tiruan, tetapi kami akan memberikan sedikit gambaran umum di sini, di bagian bab ini.

Otak manusia terdiri dari miliaran dan miliaran neuron—menurut beberapa studi ilmiah, jumlahnya mencapai hampir 90 miliar. Penelitian juga menunjukkan bahwa Neuron biasanya jauh lebih lambat daripada CPU di komputer, tetapi ia mengimbanginya dengan jumlah neuron yang sangat banyak, serta konektivitas yang sangat luas.

Koneksi ini dikenal sebagai "Sinaps", dan menariknya, mereka bekerja secara paralel satu sama lain, mirip dengan pemrosesan paralel di komputer. Perlu dicatat bahwa di komputer, CPU selalu aktif dan memori (seperti RAM) merupakan entitas terpisah. Namun, di otak manusia, semua Sinaps terdistribusi secara merata di jaringannya sendiri. Untuk sekali lagi menyamakan otak dengan komputer, pemrosesan sebenarnya terjadi di Neuron, sedangkan memori terletak di Sinaps otak manusia.

Di dalam infrastruktur Neuron terdapat apa yang dikenal sebagai "Perceptron". Layaknya sistem Pembelajaran Mesin, Perceptron juga dapat memproses masukan dan menghasilkan keluaran dengan caranya sendiri. Hal ini dapat direpresentasikan secara matematis sebagai berikut:

$$Xj = E, R, j = 1, \ldots d$$

Di mana:

D = bobot koneksi (juga dikenal sebagai "Bobot Sinaptik");

Wj = R adalah keluaran spesifik;

Y = jumlah bobot masukan.

Jumlah masukan yang terbobot dapat direpresentasikan secara matematis sebagai berikut:

$$Y = d \sum j = 1 WjXj + Wo$$

Di mana:

Wo = nilai intersep untuk mengoptimalkan Model Perceptron lebih lanjut.

Keluaran aktual Perceptron direpresentasikan secara matematis sebagai berikut:

$$Y = W\text{^}t * X.$$

Dalam situasi ini:

$$W = [W0, W1, \ldots Wd]\text{^}T$$
$$X = [1, x1, \ldots Xd\text{^}T.$$

Nilai-nilai yang disebutkan di atas juga dikenal sebagai "Vektor Tertambah", yang mencakup "Bobot Bias", yang berorientasi statistik, serta nilai-nilai spesifik untuk masukan. Ketika Model Perseptron menjalani fase pengujiannya, bobot statistik (dilambangkan sebagai "W1") dan masukan (dilambangkan sebagai "X") akan dihitung untuk menghasilkan keluaran yang diinginkan, yang dilambangkan sebagai "y".

Namun, sistem Pembelajaran Mesin perlu mempelajari bobot statistik khusus yang telah ditetapkan, serta parameternya, agar dapat menghasilkan keluaran yang dibutuhkan. Proses spesifik ini dapat direpresentasikan secara matematis sebagai berikut:

$$Y = Wx + w0.$$

Hal di atas hanya merepresentasikan satu masukan dan satu keluaran. Ini juga menjadi garis linier solid ketika disematkan pada Bidang Geometri Kartesius. Namun, jika terdapat lebih dari satu masukan, maka garis linier ini menjadi apa yang dikenal sebagai "Hiperbidang". Dalam contoh khusus ini, masukan-masukan ini dapat digunakan untuk mengimplementasikan apa yang dikenal sebagai "Kesesuaian Linier Multivariat". Dari sini, masukan dalam Model Perceptron dapat dibagi menjadi dua, di mana salah satu ruang masukan berisi nilai positif, dan ruang masukan lainnya berisi nilai negatif.

Pembagian ini dapat dilakukan menggunakan teknik yang dikenal sebagai "Fungsi Diskriminan Linear", dan operasi yang dijalankannya dikenal sebagai "Fungsi Ambang". Hal ini dapat direpresentasikan secara matematis sebagai berikut:

$$S(a) = \{1 \text{ jika } a > 0;\ 0 \text{ otherwise}\}$$
$$\text{Pilih } \{C1 \text{ jika } s(w\text{^}tx) > 0, C2 \text{ otherwise}].$$

Perlu dicatat bahwa setiap Perceptron sebenarnya merupakan fungsi berbasis lokal dari berbagai masukan dan bobot sinaptiknya. Namun, penerapan Model Perceptron ke dalam sistem Pembelajaran Mesin merupakan proses dua langkah. Hal ini dapat direpresentasikan secara matematis sebagai berikut:

$$Oi = W\text{^}TI * X$$
$$Yi = \exp 0i / \sum k \exp 0k.$$

Melatih Perceptron

Karena Perceptron sebenarnya mendefinisikan Hyperplane, teknik yang dikenal sebagai "Pembelajaran Daring" digunakan. Dalam skenario khusus ini, seluruh dataset tidak

dimasukkan ke dalam Model Perceptron, melainkan diberikan sampel representatif. Ada dua keuntungan dari pendekatan ini, yaitu sebagai berikut:

- Pendekatan ini memanfaatkan daya pemrosesan dan sumber daya Model Perceptron secara efisien;
- Model Perceptron dapat menguraikan dengan cepat dataset lama dan dataset baru dalam data pelatihan.

Dengan teknik "Pembelajaran Daring", Fungsionalitas Kesalahan yang terkait dengan dataset tidak ditimpa sama sekali. Sebaliknya, bobot statistik pertama yang ditetapkan digunakan untuk menyempurnakan parameter guna meminimalkan kesalahan yang ditemukan di masa mendatang dalam dataset. Teknik ini dapat direpresentasikan secara matematis sebagai berikut:

$$E\hat{}T(w|x\hat{}1, r\hat{}1 = ½ (r\hat{}t - y\hat{}t)\hat{}2 = ½ \{r\hat{}2 - (w\hat{}T X\hat{}1)\}\hat{}2.$$

Pembaruan Daring dapat direpresentasikan sebagai berikut:

$$Delta\ W\hat{}tj = n(r\hat{}1 - y\hat{}t)\ x\hat{}t * j$$

Di mana:

N = faktor pembelajaran.

Faktor pembelajaran diturunkan secara perlahan selama periode waktu yang telah ditentukan agar Faktor Konvergensi dapat terjadi. Namun, jika set pelatihan bersifat tetap dan tidak cukup dinamis, bobot statistik kemudian ditetapkan secara acak. Dalam istilah teknis, ini dikenal sebagai "Penurunan Gradien Stokastik". Dalam kondisi normal, biasanya merupakan ide yang sangat baik untuk mengoptimalkan dan/atau menormalkan berbagai masukan sehingga semuanya dapat dipusatkan di sekitar nilai 0, namun tetap mempertahankan jenis properti skalar yang sama.

Dengan cara yang sama, Aturan Pembaruan juga dapat direpresentasikan secara matematis diturunkan untuk semua jenis skenario Klasifikasi, yang menggunakan teknik khusus yang disebut "Diskriminasi Logistik". Dalam hal ini, Pembaruan sebenarnya dilakukan setelah setiap Varians Pola, alih-alih menunggu hingga akhir dan kemudian mendapatkan penjumlahan matematisnya.

Misalnya, ketika terdapat dua jenis Kelas yang terlibat dalam sistem Pembelajaran Mesin, Instansi Tunggal dapat direpresentasikan sebagai berikut:

$$(x\hat{}2, r\hat{}2)$$

Di mana:

$$R\hat{}I = X\hat{}t\ E\ C1 \text{ dan } R\hat{}1 = 0 \text{ jika } X\hat{}1\ E\ C2.$$

Dari sini, keluaran tunggal dapat dihitung sebagai berikut:

$$Y\hat{}1 = \text{sigmoid}\,(w\hat{}T, x\hat{}t).$$

Dari sini, Entropi Silang kemudian dihitung dari rumus matematika ini:

$$E\hat{}t\,(w|x\hat{}t, r\hat{}t) = -r\hat{}t \log y\hat{}t - (1 - r\hat{}t) \log (1 - y\hat{}t).$$

Semua hal di atas dapat direpresentasikan oleh kode sumber Python berikut:

```
For i = 1, … K
For j = 0, … d
Wij ← rand (-0.01, 0.01)
Repeat
For all (x^t, r^t) E X in random order
For I = 1, … K
Oi = 0
For j = 0, … d
Oi ← Oi + Wijx^1
For I = 1, … K
Y1 ← exp(oi)/∑k exp(0k)
For I = 1, … K
For j = 0, … d
Wij ← Wij + n(n^ti -y) * x^1j.
```

Fungsi Boolean

Dari dalam Fungsi Boolean, masukan dianggap biner, dan nilai keluaran biasanya 1 jika Nilai Fungsi terkait dianggap "Benar", dan memiliki nilai 0 dalam wujud materi lainnya. Dengan demikian, hal ini juga dapat dikarakterisasikan sebagai masalah klasifikasi dua tingkat, dan diskriminan matematis dapat dihitung dari rumus berikut:

$$Y = s(X1 + X2 - 1{,}5)$$

Di mana:

$$X = [1, x1, x2]\hat{}T$$
$$W = [-1{,}5, 1, 1]\hat{}T.$$

Penting untuk dicatat pada titik ini bahwa Fungsi Boolean biasanya terdiri dari Operator statistik AND dan OR. Keduanya dapat dipisahkan secara linear jika konsep Perceptron digunakan. Namun, operator XOR tidak tersedia dalam Fungsi Boolean. Dengan demikian, keduanya juga dapat diselesaikan oleh Perceptron. Untuk contoh ini, masukan dan keluaran yang diperlukan diberikan oleh matriks di bawah ini:

X1	X2	R
0	0	0

0	1	1
1	0	1
1	1	0

Perceptron Berlapis Ganda

Biasanya, Perceptron biasanya hanya terdiri dari satu lapis bobot statistik, sehingga hanya dapat beroperasi secara linear. Perceptron tidak dapat menangani operator statistik XOR, yang mana diskriminan matematisnya diasumsikan bersifat nonlinier. Namun, jika konsep "Jaringan Umpan Maju" digunakan, maka "Lapisan Tersembunyi" sebenarnya terdapat di dalam Perceptron, yang dapat ditempatkan di antara lapis masukan dan keluaran.

Dengan demikian, Perceptron Berlapis Ganda ini dapat digunakan untuk menerapkan model non-diskriminan ke dalam sistem Pembelajaran Mesin, dan oleh karena itu, seseorang dapat dengan mudah menghitung Fungsionalitas Nonlinier dari berbagai masukan. Dalam contoh ini, masukan "x" dimasukkan ke dalam lapisan masukan, dan proses aktivasi ini akan berlanjut, dan Nilai Tersembunyi (dilambangkan dengan "Zh") kemudian dihitung dengan rumus matematika berikut:

$$Zh = \text{sigmoid}(w\text{^}Th * X) = 1/1 + \exp[-(\sum d\,j = 1\,WhjXj + Who)], \text{^}h = 1, \ldots H$$

Keluaran dari sistem Pembelajaran Mesin (yang dilambangkan sebagai "Yi") dihitung dengan rumus matematika berikut:

$$Yi = V\text{^}Ti * Z = \sum H\,h = 1 * VihZh + Vi0$$

Matriks berikut menunjukkan berbagai masukan yang digunakan dengan Operator XOR statistik. Penting untuk dicatat bahwa dalam contoh ini, terdapat dua unit tersembunyi yang sebenarnya menerapkan dua "AND", dan keluarannya menerima kondisi "OR" statistik apa pun dari keduanya:

X1	X2	Z1	Z2	Y
0	0	0	0	0
0	1	0	1	1
1	0	1	0	1
1	1	0	0	0

Multi-Layer Perceptron (MLP): Sebuah Aproksimator Statistik

Dari titik ini, Fungsi Boolean juga dapat mengoperasikan disunion dari kumpulan data yang terunion. Ekspresi statistik ini dapat dengan mudah diimplementasikan ke dalam MLP dengan memanfaatkan Hidden Layer. Dengan demikian, setiap union dapat ditingkatkan dengan nilai satu Hidden Unit, dan demikian pula, setiap disunion dapat dikurangi dengan satu nilai Output Unit.

Hal ini direpresentasikan secara statistik sebagai berikut:

$$\text{X1 XOR X2} \ = \ (\text{X1 AND} \ - \ \text{X2}) \ \text{OR} \ (\sim \text{X1 AND X2}).$$

Dalam Pemrosesan Paralel dalam sistem Pembelajaran Mesin, dua MLP dapat beroperasi bersamaan dengan dua "AND", dan Perceptron lain kemudian dapat secara statistik "OR" keduanya, sehingga membentuk kumpulan yang terunion. Hal ini direpresentasikan secara statistik sebagai berikut:

$$Z1 \ = \ S(x1 \ - \ x2 - 0{,}5)$$
$$Z2 \ = \ S(-X1 \ + \ X2 - 0{,}5)$$
$$Y \ = \ s(Z1 \ + \ Z2 - 0{,}5).$$

Dengan demikian, pembuktian Aproksimasi Statistik mudah ditunjukkan dengan dua Lapisan Tersembunyi. Misalnya, untuk setiap masukan, wilayah statistiknya dapat dibatasi oleh serangkaian "Hiperbidang", dengan memanfaatkan Unit Tersembunyi pada Lapisan Tersembunyi. Dengan demikian, Unit Tersembunyi yang ada di lapisan kedua kemudian secara statistik "AND" keduanya, yang kemudian mengikatnya ke wilayah spesifik tersebut dalam sistem Pembelajaran Mesin.

Dari sana, bobot koneksi dari Unit Tersembunyi ke Unit Keluaran akan sama nilainya dengan nilai yang diprediksi. Proses ini terkadang juga dikenal sebagai "Aproksimasi Konstan Sepotong-sepotong".

Algoritma Backpropagation

Perlu dicatat juga bahwa melatih sistem MLP pada dasarnya sama dengan melatih satu Perceptron. Namun dalam situasi ini, keluaran yang dihasilkan pada dasarnya adalah fungsi nonlinier yang berkorelasi kuat dengan masukannya. Proses ini secara teknis juga dikenal sebagai "Gradien", dan secara matematis direpresentasikan sebagai berikut:

$$\text{VE/VWhj} = (\text{VE/VYi})^{*}(\text{VYi/VZh})^{*}(\text{VZh/VWhj}).$$

Menarik untuk dicatat bahwa kesalahan statistik sebenarnya "berpropagasi balik" dari pita terluar nilai "Y" kembali ke berbagai nilai, sehingga dinamakan "Algoritma Propagasi Balik".

Regresi Nonlinier

Dalam sistem Pembelajaran Mesin, Regresi Nonlinier dapat direpresentasikan secara statistik sebagai berikut:

$$\text{Y\^{}Z} \ = \ \text{H}\sum\text{h} \ = \ 1 \ \text{Vh} \ ^{*} \ \text{Z\^{}th} \ + \ \text{V0}.$$

Lapisan kedua Perceptron dikaitkan dengan Unit Tersembunyi dan masukan yang berkorelasi; Dengan demikian, "Aturan Kuadrat Terkecil" dapat digunakan untuk memperbarui bobot statistik Lapisan Kedua secara harfiah dengan rumus berikut:

$$\Delta Vh = n\sum r * (r\hat{}1 - Z\hat{}t) * Z\hat{}th$$

Namun, untuk memperbarui bobot statistik Lapisan Pertama, "Aturan Rantai" kini diterapkan, yaitu sebagai berikut:

$$\Delta Whj = -n^{*}(VE)/VWhj$$
$$N\sum t\,(VE\hat{}1/Vy\hat{}t) * (Vy\hat{}t/VzTh) * (VZ\hat{}th/VWhj)$$
$$N\sum t - [(r\hat{}2 - y\hat{}t)/VE\hat{}t] * [Vh/Vy\hat{}T|VZ\hat{}tn] * [Z\hat{}tH\,(1 - 1Z\hat{}th)x\hat{}tJ/Vz\hat{}th|VWhj]$$
$$N\sum [(r\hat{}t - y\hat{}t)] * [VhZ\hat{}th] * [(1 - Zth) * x\hat{}tj].$$

Dengan Aturan Rantai yang kini telah ditetapkan secara kokoh melalui urutan persamaan di atas, pola statistik setiap arah dapat dihitung untuk menentukan parameter mana yang perlu diubah dalam sistem Pembelajaran Mesin, serta besaran spesifik dari perubahan tersebut.

Namun, dalam "Proses Pembelajaran Batch", setiap perubahan besaran diakumulasikan selama deret waktu tertentu, dan perubahan tersebut hanya dapat dilakukan setelah seluruh Aturan Rantai dilalui. Dalam Aturan Rantai, dimungkinkan juga untuk memiliki Beberapa Unit Keluaran, yang dalam hal ini sistem Pembelajaran Mesin harus mempelajarinya dengan rumus matematika berikut:

$$Y\hat{}Ti = H\sum h + 1 * VihZ\hat{}th + Vi0$$

Namun, rumus di atas hanya merepresentasikan pembaruan satu kali. Agar sistem Pembelajaran Mesin dapat memperbarui secara real-time 24/7/365, rangkaian persamaan statistik berikut diperlukan, yang secara teknis disebut "Aturan Pembaruan Batch":

$$\Delta Vih = n\sum t * (r\hat{}zi - y\hat{}ti) * Z\hat{}rh$$
$$\Delta Whj = n\sum [\sum t * (r\hat{}ti - y\hat{}ti)Vjh] * [Z\hat{}th\,(1 - Z\hat{}th) * X\hat{}tj]$$

2.6 DESKRIPSI KELAS STATISTIK DALAM PEMBELAJARAN MESIN

Dalam dunia Pembelajaran Mesin, terdapat sejumlah jenis Diskriminasi ini, dan akan dibahas lebih lanjut dalam subbagian ini.

Diskriminasi Statistik Dua Kelas

Jika terdapat dua kelas masukan yang digunakan untuk sistem Pembelajaran Mesin, maka hanya satu keluaran yang akan dihasilkan. Hal ini direpresentasikan secara matematis sebagai berikut:

$$Y\hat{}t = \text{sigmoid}\,(H\sum h = 1 * VhZ\hat{}th + v0.$$

Hal ini selanjutnya mengaproksimasi nilai potensial keluaran yang direpresentasikan sebagai berikut:

$$P(C1|X\hat{}t)$$
$$P(C2|X\hat{}T) = 1 - y\hat{}t.$$

Distribusi Multikelas

Jika terdapat jumlah keluaran yang tidak terbatas yang akan dihitung oleh sistem Pembelajaran Mesin, nilai "K" (yang merepresentasikan jumlah keluaran yang tidak pasti) dapat direpresentasikan secara matematis sebagai berikut:

$$O\hat{}ti = H\sum h = 1 * VihZ\hat{}th + Vi0.$$

Penting juga untuk dicatat bahwa dalam jenis Diskriminasi Kelas ini, keluaran yang diturunkan dari sistem Pembelajaran Mesin dapat saling eksklusif atau inklusif satu sama lain. Hal ini dapat direpresentasikan secara statistik sebagai:

$$X\hat{}ti = (Exp0\hat{}t1) / (\sum k\ ep\ 0\hat{}tk)$$

Diskriminasi Multilabel

Bisa juga terjadi bahwa Jika terdapat beberapa Label yang digunakan dalam sistem Pembelajaran Mesin untuk suatu masukan, dan jika terdapat jumlah tak tentu (juga direpresentasikan sebagai "K"), dan jika keduanya secara statistik saling eksklusif, maka hal ini direpresentasikan sebagai berikut:

$$R\hat{}ti = \{1 \text{ if } x\hat{}r \text{ has a label of “I”}\}, \{0 \text{ otherwise}\}.$$

Perlu dicatat bahwa dalam situasi Diskriminasi seperti ini, pendekatan tradisional adalah mengevaluasi "K" sebagai dua masalah Klasifikasi yang terpisah dan berbeda. Skenario seperti ini biasanya ditemukan dalam model linear, terutama ketika Perseptron digunakan. Di sini, berpotensi terdapat jumlah tak tentu model berbasis nilai "K" yang ada, dengan keluaran berbasis nilai Sigmoid tertentu.

Dengan demikian, Lapisan Tersembunyi kini diperlukan dalam sistem Pembelajaran Mesin, sehingga nilai untuk "K" dapat dilatih secara terpisah satu sama lain, terutama jika Perseptron Berlapis-lapis digunakan. Kasus ini juga dapat terjadi jika terdapat Lapisan Tersembunyi yang umum untuk semua Perseptron yang bahkan menggunakan Perseptron yang sama. dataset. Jika hal ini benar-benar terjadi, maka ukuran dataset juga dapat meningkat, sehingga sangat mengurangi daya pemrosesan sistem Pembelajaran Mesin.

Fenomena ini dapat direpresentasikan secara matematis sebagai berikut:

$$Y\hat{}ti = sigmoid\ (H\sum h = 1 * VihZ\hat{}th + Vi0)$$

Di mana:

$Y\hat{}ti, I = 1, \ldots K$ terhubung ke $Z\hat{}Th, h = 1, \ldots H$ yang sama.

Pelatihan Berlebih

Jika terdapat Multilevel Perceptron yang digunakan, kemungkinan besar akan terdapat jumlah Unit Tersembunyi (dilambangkan dengan "H") dan jumlah keluaran (dilambangkan dengan "K") yang tak tentu, dan keduanya juga akan memiliki nilai bobot statistik $H(d + 1)$. Semua ini akan berada di dalam lapisan pertama Multilevel Perceptron (MLP), dan juga, akan ada bobot statistik tambahan yang akan ditetapkan ke lapisan kedua (dilambangkan sebagai "$K(H + 1)$").

Namun, dalam situasi di mana nilai "d" dan "K" telah ditentukan sebelumnya, optimasi lebih lanjut terhadap Multilevel Perceptron perlu dilakukan sebelum dapat diimplementasikan ke dalam sistem Pembelajaran Mesin. Penting juga untuk diingat bahwa jika model MLP dibuat terlalu kompleks, maka sistem Pembelajaran Mesin akan memperhitungkan semua "noise" tambahan yang telah dihasilkan, sehingga tidak akan dapat menghasilkan set keluaran yang dioptimalkan sesuai kebutuhan aplikasi yang bersangkutan.

Hal ini terutama berlaku ketika model statistik yang dikenal sebagai "Regresi Polinomial" digunakan. Praktik umum, terkait dengan Pembelajaran Mesin, adalah meningkatkan orde besaran statistik yang sudah dimilikinya. Selain itu, jika jumlah total Unit Tersembunyi cukup besar, output juga akan menurun secara signifikan, sehingga semakin memperburuk Bias/Varians dalam sistem Pembelajaran Mesin. Fenomena semacam ini juga biasanya terjadi ketika sistem Pembelajaran Mesin menghabiskan terlalu banyak waktu untuk mempelajari dataset, setidaknya pada awalnya. Khususnya, Kesalahan Validasi akan meningkat secara drastis, dan hal ini harus dihindari dengan segala cara.

Misalnya, ketika dataset pertama kali dimasukkan ke dalam sistem Pembelajaran Mesin, semuanya memiliki faktor bobot statistik awal yang hampir 0. Namun, jika pelatihan berlangsung dalam periode waktu yang lebih lama, bobot ini kemudian akan menjauh dari 0, dan ukurannya akan membesar dengan cepat. Hasil akhirnya adalah hal ini dapat menurunkan kualitas kinerja sistem Pembelajaran Mesin secara signifikan. Alasan utamanya adalah karena hal ini justru akan meningkatkan jumlah total parameter dalam sistem Pembelajaran Mesin, sehingga bahkan mengesampingkan parameter yang telah diterapkan.

Pada akhirnya, sistem Pembelajaran Mesin menjadi terlalu rumit sejak awal, dan intinya adalah sistem ini tidak akan menghasilkan serangkaian keluaran yang diinginkan untuk menyelesaikan proyek tepat waktu.

Akibatnya, proses ini harus dihentikan cukup dini agar fenomena yang dikenal sebagai "Overtraining" tidak terjadi. Dengan demikian, "titik sempurna" di mana tingkat pelatihan awal untuk sistem Pembelajaran Mesin harus dihentikan adalah pada titik di mana jumlah optimal

Lapisan Tersembunyi dalam Multilevel Perceptron akhirnya tercapai. Namun, hal ini hanya dapat dipastikan dengan menggunakan teknik statistik yang dikenal sebagai "Validasi Silang".

Pembelajaran Sistem Pembelajaran Mesin Melalui Representasi Tersembunyi dan Statistik.

Bagaimana Sistem Pembelajaran Mesin Dapat Berlatih dari Representasi Statistik Tersembunyi Sebagaimana telah diulas sebelumnya dalam bab ini, Regresor Dasar atau Pengklasifikasi Data dalam sistem Pembelajaran Mesin dapat direpresentasikan secara statistik sebagai berikut:

$$Y = h\sum j = 1 * (VjXj + V0).$$

Jika Klasifikasi Linear digunakan dalam sistem Pembelajaran Mesin, maka seseorang cukup melihat tanda matematika "y" untuk memilih salah satu dari dua kelas. Pendekatan ini dianggap bersifat Linear, tetapi Anda dapat melangkah lebih jauh dengan menggunakan teknik lain yang dikenal sebagai "Fungsi Basis Nonlinier". Ini dapat direpresentasikan secara statistik sebagai berikut:

$$Y = H\sum h = 1 * [VjO/h(x)]$$

Di mana:

$O/(x) =$ Fungsi Basis Nonlinier.

Selain itu, dengan cara yang sangat mirip, teknik statistik jenis ini juga dapat digunakan untuk Multilevel Perceptron, dan secara matematis digambarkan sebagai berikut:

$$Y = H\sum h = 1 * Vh0/(X|Wh)$$

Di mana:

$0/(X|Wh) =$ sigmoid (W^ThX).

Ada juga sejumlah konsep kunci yang penting ketika sistem Pembelajaran Mesin menggunakan representasi statistik tersembunyi. Konsep-konsep tersebut adalah sebagai berikut:

1) <u>Penanaman:</u>
   Ini adalah representasi statistik dari sebuah instans statistik yang ditemukan dalam ruang tersembunyi di Multi-Layer Perceptron. Hal ini biasanya terjadi ketika lapisan pertama (dilambangkan sebagai H < d) mengimplementasikan properti Reduksi Dimensi ke dalam sistem Pembelajaran Mesin. Lebih lanjut, unit tersembunyi yang berada di sini dapat dianalisis lebih lanjut dengan memeriksa secara kritis faktor bobot statistik yang masuk ke sistem Pembelajaran Mesin. Selain itu, jika masukan dianggap ternormalisasi secara statistik, maka hal ini memberikan indikasi yang cukup baik tentang kepentingan relatif dan tingkat prioritasnya dalam sistem Pembelajaran Mesin.
2) <u>Pembelajaran Transfer:</u>

Hal ini terjadi ketika sistem Pembelajaran Mesin terdiri dari dua tugas yang berbeda namun saling terkait yang sedang dikerjakan. Misalnya, jika sistem mencoba menyelesaikan keluaran yang dibutuhkan untuk Masalah X, dan jika tidak terdapat cukup dataset untuk itu, maka secara teoritis Anda dapat melatih sistem Pembelajaran Mesin untuk belajar dari dataset yang digunakan untuk menyelesaikan keluaran untuk Masalah Y. Dengan kata lain, Anda secara harfiah mentransfer Lapisan Tersembunyi dari Masalah Y dan menanamkannya ke dalam Masalah X.

3) Pembelajaran Semi-supervised:

Skenario ini muncul ketika sistem Pembelajaran Mesin memiliki satu dataset berlabel kecil serta dataset tak berlabel yang jauh lebih besar. Dari sini, dataset tak berlabel tersebut dapat digunakan untuk mempelajari lebih lanjut tentang ruang tersembunyi dari dataset berlabel. Hasil akhirnya adalah dataset ini kemudian dapat digunakan untuk tujuan pelatihan awal.

Autoencoder

Komponen unik lain dari Multi-Layer Perceptrons dikenal sebagai "Autoencoder". Dalam arsitektur semacam ini, jumlah total masukan yang masuk ke sistem Pembelajaran Mesin akan sama dengan jumlah total keluaran yang akan dihasilkan. Dalam hal ini, nilai kuantitatif keluaran juga akan sama dengan nilai kuantitatif masukan. Namun, jika jumlah total Unit Tersembunyi sebenarnya lebih kecil daripada nilai total masukan, maka fenomena yang dikenal sebagai "Reduksi Dimensionalitas" akan terjadi.

Perlu dicatat bahwa lapisan pertama dalam Multi-Layer Perceptron adalah "Encoder", dan nilai-nilai dari Unit Tersembunyi inilah yang membentuk "Kode" yang mendasarinya. Oleh karena itu, Multi-Layer Perceptron diperlukan untuk memastikan perkiraan terbaik masukan dalam Lapisan Tersembunyi, sehingga dapat diduplikasi di kemudian hari.

Representasi matematis dari Encoder adalah sebagai berikut:

$$Z\hat{}t = Enc(X\hat{}t|W)$$

Di mana:

$W = \text{parameter Encoder.}$

Dari rumus di atas, lapisan kedua dari Unit Tersembunyi di Multi-Layer Perceptron kini bertindak sebagai apa yang dikenal sebagai "Decoder". Hal ini direpresentasikan secara matematis sebagai berikut:

$$X\hat{}t = Dec(Z\hat{}t|V)$$

Di mana:

$V = \text{parameter Decoder.}$

Dari sini, Backpropagation mencoba menentukan parameter Encoder dan Decoder terbaik dalam upaya terpadu untuk menemukan "Kesalahan Rekonstruksi". Hal ini dapat dihitung sebagai berikut:

$$E(W, V|X) = \sum t\,||x\hat{}t - x\hat{}t||\hat{}2 = \sum t||X\hat{}2 - Dec[Enc(X\hat{}2|W) * (V)]||\hat{}2.$$

Ketika Encoder berada dalam fase desain, sepotong kode sumber yang dianggap berdimensi kecil, serta Encoder yang memiliki nilai Kapasitas Rendah, disertakan untuk menjaga integritas dataset yang digunakan oleh sistem Pembelajaran Mesin. Namun, mungkin ada bagian dari dataset yang dapat dibuang karena adanya noise atau varians. Hal ini telah dibahas panjang lebar di bagian sebelumnya dalam bab ini.

Namun, jika Encoder dan Decoder tidak hanya berada dalam satu lapisan, tetapi berada dalam beberapa lapisan, Encoder akan menerapkan apa yang dikenal sebagai "Pengurangan Dimensi Nonlinier". Penting juga untuk dicatat bahwa ekstensi dari Autoencoder dikenal sebagai "Denoising Autoencoder". Ini adalah situasi di mana derau atau tingkat varians ekstra ditambahkan untuk menciptakan apa yang disebut "Contoh Virtual" agar sistem Pembelajaran Mesin dapat mempelajarinya juga, termasuk derau atau varians dari kumpulan data.

Penambahan derau atau varians ekstra ini dilakukan secara sengaja untuk memperkirakan kesalahan yang mungkin terjadi ketika keluaran akhir dihasilkan oleh sistem Pembelajaran Mesin. Ada ekstensi lain lagi, yang dikenal sebagai "Sparse Encoder". Tujuan dari ekstensi ekstra ini adalah untuk menanamkan derau atau varians ekstra ini, sehingga tidak berdimensi panjang, melainkan bersifat "renggang".

Terakhir, cara lain bagi Multi-Layer Perceptron untuk menerapkan Reduksi Dimensionalitas ke dalam sistem Pembelajaran Mesin adalah melalui proses yang dikenal sebagai "Penskalaan Multidimensi". Hal ini dapat direpresentasikan secara matematis sebagai berikut:

$$E(O|X) = \sum rt\,[||[g(x\hat{}r|O) - g(x\hat{}t|0)]|| - ||x\hat{}t - x\hat{}s| / ||x\hat{}r - x\hat{}s||.$$

Arsitektur Word2vec

Jika Autoencoder dianggap cukup "berisik", ia akan dipaksa untuk membuat dan menghasilkan potongan kode yang serupa karena semua keluaran yang dihasilkan oleh sistem Pembelajaran Mesin kurang lebih harus sama nilainya. Hal ini, tentu saja, sangat bergantung pada jenis aplikasi spesifik yang digunakannya. Misalnya, secara sederhana, jika terdapat masukan yang berbeda, maka keluaran yang sama harus dihasilkan oleh sistem Pembelajaran Mesin.

Premis di balik contoh yang disebutkan di atas dikenal secara khusus sebagai "arsitektur word2vec". Di sini, keluarannya bersifat kualitatif, seperti sebuah kata; masukan ke dalam sistem Pembelajaran Mesin adalah konteks dari kata spesifik tersebut. Selain itu, jika dua kata terpisah cukup sering muncul dalam konteks yang sama, maka keduanya juga seharusnya serupa. Dengan demikian, tujuan keseluruhan teknik ini adalah untuk memastikan dan

menemukan representasi statistik yang berkelanjutan untuk kata-kata yang dapat digunakan dalam apa yang dikenal sebagai "Pemrosesan Bahasa Alami", atau disingkat "NLP".

Dari titik ini, sebenarnya terdapat dua jenis model yang digunakan untuk arsitektur word2vec, yaitu "CBOW" dan "Skip-Gram". Fitur umum di antara keduanya adalah sebagai berikut:

- Masukan d-dimensi;
- Keluaran d-dimensi;
- Jika H < d memiliki jumlah Unit Tersembunyi "X" dan jumlah Lapisan Tersembunyi "X", maka ini akan sangat mirip dengan Autoencoder (sebagaimana diulas di subbagian sebelumnya).

Namun, kedua model ini juga berbeda satu sama lain dalam cara kita mendefinisikan istilah konteks untuk kata tersebut. Dalam model CBOW, semua kata spesifik yang digunakan dirata-ratakan bersama untuk membentuk representasi binernya. Dengan demikian, ini juga dapat menjadi input bagi sistem Pembelajaran Mesin. Namun, dalam model Skip-Gram, semua kata yang digunakan dirata-ratakan satu per satu, dan sebagai hasilnya, kata-kata tersebut membentuk pasangan set data pelatihan yang berbeda yang juga dapat digunakan oleh sistem Pembelajaran Mesin.

Pada akhirnya, telah ditentukan bahwa model Skip-Gram bekerja jauh lebih baik daripada model CBOW untuk sistem Pembelajaran Mesin. Namun, ada juga sejumlah cara untuk meningkatkan teknik word2vec, yaitu sebagai berikut:

- Kata-kata seperti "the" atau "with", yang cukup sering digunakan, dapat digunakan lebih sedikit agar sistem Pembelajaran Mesin lebih efisien;
- Waktu komputasi dan pemrosesan sistem Pembelajaran Mesin dapat dioptimalkan lebih lanjut dan dibuat jauh lebih efisien dengan terlebih dahulu mengambil sampel statistik dari keluarannya. Hal ini secara teknis juga dikenal sebagai "Sampling Negatif".

2.7 PENERAPAN PEMBELAJARAN MESIN UNTUK PERLINDUNGAN TITIK AKHIR

Dunia semakin bergantung pada infrastruktur siber yang terus berkembang. Tidak hanya komputer dan perangkat pintar kita yang menghubungkan kita dengan keluarga, teman, pemberi kerja, pemerintah, dan perusahaan tempat kita membeli barang dan jasa, tetapi juga infrastruktur kita menjadi lebih otomatis dan terkomputerisasi, termasuk semua moda transportasi, pembangkitan dan distribusi listrik, manufaktur, logistik rantai pasokan, dan lain-lain. Mengamankan semua infrastruktur siber ini menjadi penting untuk memiliki keberadaan yang efisien dan aman. Banyak vektor serangan ada, sehingga setiap strategi keamanan siber harus luas dan mendalam. Ini umumnya disebut sebagai "pertahanan mendalam." Karena perangkat titik akhir seperti komputer pribadi dan telepon pintar sangat banyak dan di bawah kendali banyak pengguna individu, mereka adalah salah satu mata rantai terlemah dalam rantai keamanan infrastruktur. Melindungi titik akhir agar tidak terinfeksi oleh malware adalah panah penting dalam tabung panah keamanan siber. Di bagian ini, kita akan mengeksplorasi bagaimana pembelajaran mesin dapat digunakan untuk membantu mendeteksi dan mencegah infeksi malware pada perangkat titik akhir.

Perlu dicatat bahwa perangkat lunak anti-malware sering disebut sebagai perangkat lunak "anti-virus", meskipun malware hadir dalam berbagai bentuk selain virus—misalnya: worm, Trojan horse, adware, spyware, ransomware, dan lain-lain. Malware juga memiliki banyak tujuan. Meskipun beberapa di antaranya hanyalah lelucon berbahaya dan mengganggu, sebagian besar memiliki motivasi ekonomi atau keamanan nasional. Malware tersebut mungkin mencoba menghasilkan pendapatan iklan atau mengalihkan pembelian ke situs web tertentu, mencuri informasi berharga tentang individu untuk dijual kembali, mencuri uang dari lembaga keuangan atau kartu kredit, mencuri kekayaan intelektual atau informasi kompetitif, mengenkripsi data berharga untuk tebusan, atau bahkan mencuri siklus komputasi untuk menambang mata uang kripto. Negara-negara dan kelompok teroris menggunakan malware untuk mendapatkan akses ke informasi intelijen penting atau untuk merusak infrastruktur musuh, seperti worm Stuxnet, yang menyebabkan sentrifus Iran hancur saat memperkaya uranium.

Sejak "cacing Morris" menginfeksi 10 persen internet pada tahun 1988, persaingan antara pembuat malware dan perangkat lunak anti-malware telah menjadi persaingan yang terus-menerus, dengan peningkatan deteksi dan pencegahan yang diikuti oleh serangan-serangan baru yang mengakali perlindungan tersebut.Dimulai pada awal 1990-an, mekanisme utama untuk mendeteksi malware mengandalkan beberapa bentuk deteksi tanda tangan. Misalnya, pendekatan deteksi paling awal hanya memeriksa kode biner untuk mendeteksi modifikasi yang menyebabkan eksekusi kode melompat ke akhir berkas untuk menjalankan perangkat lunak berbahaya, sebuah pola yang tidak digunakan oleh perangkat lunak yang aman. Persaingan semakin sengit dari sana.

Saat ini, perusahaan perangkat lunak anti-malware menerapkan banyak sistem "honeypot" yang memiliki kerentanan keamanan yang diketahui dan tampaknya menjadi sistem yang menarik bagi penyerang untuk menarik dan mendapatkan sampel malware baru. Setelah malware baru ditangkap dalam honeypot, malware tersebut dianalisis oleh analis ancaman untuk mengembangkan "tanda tangan" dari contoh-contoh malware baru ini. Tanda tangan baru ini kemudian diterapkan ke titik akhir yang berisi produk mereka untuk mendeteksi dan memblokir varian-varian baru ini. Bentuk tanda tangan yang paling sederhana dan umum adalah menghitung hash kriptografi (misalnya MD5, SHA-1, SHA-2, SHA256, dan lain-lain.) dari berkas biner dan mendistribusikannya ke titik akhir untuk memblokir berkas apa pun dengan hash yang sama. Hash kriptografi dirancang sedemikian rupa sehingga sangat kecil kemungkinan perangkat lunak yang aman memiliki hash yang sama, sehingga tidak akan terblokir secara tidak sengaja. Namun, mengubah satu bit saja dalam biner malware akan menghasilkan hash yang sama sekali berbeda, sehingga sangat mudah bagi pembuat malware untuk membuat beberapa salinan malware yang sama dengan tanda tangan hash yang berbeda (dikenal sebagai malware "polimorfik"). Malware yang membuat salinannya sendiri dengan sedikit perbedaan dikenal sebagai "metamorfik".

Teknik hash "Fuzzy" dapat digunakan untuk menggagalkan beberapa modifikasi sederhana pada biner malware dan tetap mendeteksi versi metamorfik ini. Context Triggered Piecewise Hash (CTPH) [Sumber a] adalah contoh pendekatan ini. Alih-alih menghitung satu

hash di seluruh berkas, sebuah hash dihasilkan untuk banyak segmen berkas. Dalam hal ini, perubahan satu bit hanya akan memengaruhi satu hash, sehingga hash yang tersisa dapat mengidentifikasi sampel malware. Bahkan dalam kasus ini, beberapa perubahan kecil di seluruh berkas dapat menghasilkan hash yang berbeda untuk setiap segmen berkas.

Untuk mengakali strategi pengubahan tanda tangan sederhana semacam ini, perangkat lunak anti-malware menghasilkan tanda tangan yang lebih canggih berdasarkan struktur dan fitur di dalam berkas yang lebih sulit diubah oleh pembuat malware, namun tetap tidak menandai perangkat lunak yang tidak berbahaya sebagai malware. Contohnya adalah "import hash" (atau imphash – sumber b). Tabel impor dihasilkan untuk setiap file yang dieksekusi untuk setiap fungsi yang dipanggil oleh file tersebut dari berkas lain. Cara tabel impor ini dibuat memungkinkan komputasi hash yang dapat mengidentifikasi keluarga malware terkait, meskipun hash berkas atau CTPH-nya berbeda.

Bentuk pembuatan tanda tangan yang lebih kompleks pun dimungkinkan, tetapi proses ini bergantung pada analis ancaman manusia, sehingga memakan waktu dan rawan kesalahan. Selama waktu yang dibutuhkan untuk mendapatkan dan mendistribusikan tanda tangan untuk malware baru (juga dikenal sebagai "malware zero-day"), semua titik akhir tanda tangan rentan terhadap serangan. Tergantung pada kebaruan sampel malware baru, jendela paparan dapat berlangsung dari beberapa hari hingga beberapa minggu (seperti mengembangkan vaksin baru untuk COVID-19 yang membutuhkan waktu lebih lama daripada mengembangkan vaksin untuk flu musiman musim gugur mendatang). Model Pembelajaran Mesin yang dapat mendeteksi malware zero-day tanpa campur tangan manusia dapat menghilangkan jendela kerentanan terhadap malware zero-day ini.

Jendela kerentanan ini bukan satu-satunya masalah dengan pendekatan berbasis tanda tangan yang diidentifikasi manusia untuk mendeteksi malware. Kerentanan lainnya adalah ketergantungan pada kemampuan untuk terus memperbarui setiap titik akhir dengan daftar tanda tangan terbaru. Untuk titik akhir yang hampir selalu terhubung ke internet, ini bukanlah risiko tambahan. Namun, jika titik akhir memiliki koneksi internet yang sporadis atau berada dalam jaringan yang sangat aman dan tidak pernah terhubung ke internet, hal ini secara drastis meningkatkan kerentanan terhadap malware zero-day. Sekali lagi, model Pembelajaran Mesin yang dapat mendeteksi malware zero-day tanpa campur tangan manusia dapat mengatasi masalah ini karena tidak memerlukan pembaruan tanda tangan berkala agar efektif dalam mendeteksi dan memblokir malware.

Sebelum sebuah malware meledak, deteksi merupakan masalah klasifikasi biner (misalnya, apakah berkas ini bersih atau berbahaya?) dengan jumlah sampel berlabel yang sangat besar sehingga menjadikannya masalah ideal untuk Pembelajaran Mesin. Setelah sebuah berkas diklasifikasikan sebagai malware, analis ancaman perlu menentukan tindakan apa yang harus diambil. Hal ini sebagian akan ditentukan oleh jenis malware yang diwakili oleh berkas tersebut (misalnya, worm, virus, trojan, ransomware, adware, dan lain-lain.). Klasifikasi multikelas dari sampel malware juga cocok untuk pendekatan Pembelajaran Mesin. Secara teori, salah satu pendekatan klasifikasi Pembelajaran Mesin yang umum digunakan dapat

digunakan. Opsi yang paling umum adalah sebagai berikut dan dijelaskan lebih rinci sebelumnya dalam bab ini:

- Random Forest;
- Pohon yang Didorong Gradien;
- Mesin Vektor Pendukung;
- Jaringan Bayesian;
- Tetangga Terdekat K;
- Regresi Logistik;
- Jaringan Syaraf Tiruan.

Pemilihan pendekatan Pembelajaran Mesin yang tepat sangat dipengaruhi oleh kendala dari permasalahan khusus ini. Misalnya, meskipun beberapa karakteristik berkas yang relevan dalam mengklasifikasikan malware adalah nilai numerik (misalnya, ukuran berkas, entropi, dan lain-lain.), Anda akan melihat dalam diskusi pemilihan fitur bahwa lebih banyak lagi yang bersifat kategoris (misalnya, panggilan API yang digunakan, kunci Registri yang dimodifikasi, dan lain-lain.) daripada numerik. Pendekatan Pembelajaran Mesin tidak hanya perlu menangani fitur kategoris, tetapi juga harus tangguh terhadap fitur yang tidak selalu ada dan tidak memiliki cara yang berarti untuk diimputasi.

Pelatihan model Pembelajaran Mesin dilakukan secara luring (offline), sehingga jumlah sumber daya komputer yang digunakan biasanya bukan kendala yang relevan. Namun, setelah model dilatih, pembuatan prediksi pada berkas sangat dibatasi oleh perangkat titik akhir yang harus menjalankan prediksi tersebut. Algoritme prediksi harus membatasi konsumsi CPU, Memori, dan Daya Baterai karena perangkat titik akhir memiliki aplikasi lain yang harus dapat berjalan secara bersamaan. Lebih lanjut, jika prediksi harus selesai sebelum file baru dapat mulai dieksekusi, latensi prediksi harus sangat rendah agar tidak memengaruhi produktivitas pengguna titik akhir. Pemilihan model Pembelajaran Mesin yang menggunakan sumber daya komputasi secara sangat efisien merupakan keputusan desain yang krusial.

Algoritma pohon keputusan (misalnya, Random Forest dan Gradient-Boosted Trees) memiliki beberapa keunggulan yang nyata dalam klasifikasi malware karena secara alami menangani kategori dalam struktur setiap simpul keputusan dan banyak fitur malware bersifat kategoris. Lebih lanjut, setelah dilatih, Pohon Keputusan relatif ringan dalam penggunaan komputasi dan memori dibandingkan dengan Jaringan Bayesian dan Jaringan Syaraf Tiruan, dan umumnya menghasilkan prediksi yang lebih baik daripada opsi lainnya.

Pemilihan Fitur dan Rekayasa Fitur untuk Mendeteksi Malware

Karena model Pembelajaran Mesin belajar dari data pelatihannya, pemilihan data yang tepat sangat penting untuk membangun model yang efektif. Model tersebut mencoba mempelajari "sinyal malware" yang tersembunyi dalam noise dari sisa konten file, sehingga data pelatihan perlu menyertakan sinyal apa pun yang biasanya ada dalam malware. Proses pemilihan data ini disebut pemilihan fitur dan rekayasa fitur. Memilih dan merekayasa fitur yang berbeda antara malware dan berkas bersih sangatlah penting. Untungnya, proses ini dapat dipandu oleh hal-hal yang digunakan oleh analis ancaman manusia untuk

mengidentifikasi malware. Bagian ini akan menjelaskan contoh jenis fitur yang sering digunakan untuk membangun model klasifikasi malware Pembelajaran Mesin.

Kerentanan dan Paparan Umum (CVE)

Setiap kali eksploitasi baru pada sistem operasi, peramban, atau aplikasi ditemukan, detailnya akan diserahkan kepada MITRE Corporation, yang didanai oleh Divisi Keamanan Siber Nasional dari Departemen Keamanan Dalam Negeri Amerika Serikat untuk memelihara daftar paparan keamanan yang diketahui dan tersedia untuk umum. MITRE memberikan masing-masing "nomor CVE" yang unik. Sistem Penilaian Kerentanan Umum (CVSS) adalah standar industri yang bebas dan terbuka untuk menilai tingkat keparahan CVE ini. CVE ini dimanfaatkan oleh malware untuk mendapatkan akses ke sistem komputer guna menyelesaikan tugas utamanya. Karena itu, CVE seringkali tidak diungkapkan kepada publik hingga vendor yang bertanggung jawab atas CVE tersebut memiliki kesempatan untuk merilis patch atau perbaikan yang menghilangkan kerentanan atau paparan tersebut.

Pada masa-masa awal deteksi anti-malware, mengidentifikasi kode yang mengeksploitasi CVE merupakan salah satu cara paling efektif untuk mendeteksi malware. Kini, puluhan ribu CVE dilaporkan setiap tahun. Sekalipun analis ancaman mampu mengimbangi serangan gencar ini, pada saat mereka merilis tanda tangan untuk CVE ini, vendor yang bertanggung jawab atas paparan ini pasti sudah merilis perbaikannya sehingga sistem yang telah ditambal dengan baik tidak lagi rentan. Hal ini juga membuat CVE menjadi fitur yang sangat buruk untuk melatih model Pembelajaran Mesin. Model tersebut dapat mempelajari semua CVE yang diketahui, tetapi peluangnya untuk memprediksi CVE di masa mendatang sangat kecil. Untungnya, CVE hanyalah "kunci" yang digunakan malware untuk membuka kunci sistem komputer. Meskipun semua "kunci" ini berbeda, begitu berada di dalam sistem, malware akan menjalankan tugasnya yang lebih mudah dideteksi daripada CVE yang digunakan untuk mendapatkan akses. Petunjuk untuk aktivitas ini dapat dideteksi dengan jenis fitur berikut (Sumber c).

String Teks

Meskipun malware utamanya terdiri dari kode yang dapat dieksekusi, kode tersebut juga berisi bidang data yang telah ditentukan sebelumnya dan data teks lainnya yang dapat membantu mengungkap malware. String teks ini dapat mencakup nama pembuat, nama berkas, nama sumber daya sistem yang digunakan, dan lain-lain. Untuk malware yang lebih canggih yang mencoba mengaburkan petunjuk ini, histogram karakter non-alfanumerik dan panjang string (baik yang sangat pendek maupun panjang) dapat membantu mendeteksi teknik ini. Meskipun string dapat menghasilkan fitur penting untuk melatih model malware, mengekstraknya dari kode yang dapat dieksekusi dapat memakan banyak komputasi, terutama untuk berkas yang lebih besar.

Urutan Byte

Rangkaian fitur efektif lainnya untuk mendeteksi malware adalah menganalisis berkas yang dapat dieksekusi pada tingkat byte. Pendekatan yang populer adalah menghitung histogram n-gram. N-gram adalah urutan dengan panjang n byte. Misalnya, trigram (3-gram) dapat berupa urutan byte "04 A7 3C". Karena kompleksitas komputasi untuk menghitung n-

gram bersifat eksponensial terhadap n, perhitungan fitur ini biasanya terbatas pada bigram dan trigram. Pendekatan sederhana ini ternyata efektif dalam membedakan beberapa bentuk malware dari eksekusi yang aman.

Opcode

Instruksi biner yang dieksekusi CPU disebut opcode (kode operasi). Penguraian bagian kode dari berkas eksekusi dengan cara yang sama seperti yang dilakukan CPU memungkinkan penghitungan frekuensi penggunaan setiap opcode. Demikian pula, histogram n-gram opcode dapat dihitung berdasarkan urutan opcode. Malware seringkali lebih sering menggunakan opcode tertentu daripada aplikasi yang aman pada umumnya. Contoh terbaru adalah malware yang mengeksploitasi serangan saluran samping cache yang dimungkinkan oleh cacat desain dalam eksekusi kode spekulatif di CPU modern seperti Meltdown dan Spectre (sumber e dan f). Serangan ini memanfaatkan opcode manipulasi cache khusus yang tidak digunakan oleh sebagian besar aplikasi. Namun, mengekstrak opcode dari sebuah executable memerlukan decompiler yang membutuhkan komputasi yang tinggi, sehingga lebih cocok untuk deteksi malware offline.

API, Panggilan Sistem, dan DAN LAIN-LAIN

Malware juga dapat menimbulkan kecurigaan dengan memeriksa perangkat lunak dan sumber daya sistem lain yang digunakannya. Penggunaan API/Panggilan Sistem tertentu atau cara API/Panggilan Sistem digunakan merupakan petunjuk penting. Demikian pula, daftar Dynamic-Linked Libraries (DAN LAIN-LAIN) yang digunakan oleh executable dan imphash yang dibahas sebelumnya dapat digunakan untuk merangkum sebagian informasi ini dan memberikan tanda-tanda yang jelas. Misalnya, contoh Spectre/Meltdown yang disebutkan sebelumnya sangat bergantung pada informasi waktu eksekusi yang akurat dan akan lebih sering melakukan panggilan ke panggilan sistem pengatur waktu daripada perangkat lunak yang aman.

Entropi

Malware canggih sering kali menggunakan enkripsi atau kompresi untuk menyembunyikan fitur-fitur yang dapat mengungkapnya. Enkripsi dan kompresi meningkatkan keacakan data biner. Entropi informasi adalah ukuran ketidakpastian dalam kemungkinan hasil suatu variabel atau keacakannya. Dengan menghitung entropi bagian-bagian kode, bagian-bagian yang telah dienkripsi atau dikompresi dapat dideteksi. Berkas yang dapat dieksekusi dengan entropi yang signifikan merupakan indikasi kuat bahwa berkas tersebut adalah malware. Namun, kompresi juga digunakan oleh aplikasi yang aman seperti pengemas yang dapat dieksekusi seperti UPX, sehingga entropi yang tinggi harus dikombinasikan dengan fitur-fitur lain untuk menghasilkan prediksi yang akurat.

Proses Seleksi Fitur untuk Deteksi Malware

Seperti kebanyakan aplikasi Pembelajaran Mesin, proses seleksi fitur merupakan hal terpenting dalam mengembangkan model yang baik. Hal ini memerlukan serangkaian eksperimen yang cermat, di mana serangkaian fitur dihasilkan dan digunakan untuk melatih model menggunakan serangkaian berkas yang dipilih untuk pelatihan. Model tersebut kemudian diuji terhadap berkas yang tidak termasuk dalam set pelatihan untuk menentukan

efikasinya terhadap berkas di luar sampel. Berbagai teknik dapat digunakan untuk menentukan tingkat kepentingan setiap fitur terhadap prediksi yang dibuat oleh model. Hal ini dikenal sebagai "kepentingan fitur". Fitur dengan tingkat kepentingan mendekati nol dapat dihapus dan model baru dilatih untuk memastikan bahwa efikasinya tidak akan terpengaruh. Fitur yang sangat berkorelasi satu sama lain juga dapat disederhanakan dengan cara yang sama untuk mencapai serangkaian fitur optimal yang tetap mencapai efikasi yang diinginkan. Hal ini khususnya penting untuk fitur yang membutuhkan komputasi yang mahal untuk dihitung. Misalnya, 4-gram umumnya lebih baik dalam membedakan malware dari berkas yang tidak berbahaya, tetapi lebih kompleks untuk dihitung daripada trigram. Eksperimen akan menentukan apakah efikasi tambahan sepadan dengan kompleksitas komputasinya. Pembuatan jenis fitur baru dan eksperimen merupakan kunci peningkatan berkelanjutan model deteksi malware.

Proses Pemilihan Fitur untuk Klasifikasi Malware

Setelah sebuah berkas diprediksi sebagai malware, analis keamanan/ancaman akan ingin mengetahui jenis malware yang terdeteksi karena respons yang diperlukan untuk adware sangat berbeda dengan ransomware, misalnya. Ini adalah masalah klasifikasi multikelas yang dapat menggunakan sebagian besar teknik Pembelajaran Mesin yang sama dengan deteksi malware, tetapi dengan serangkaian kendala yang berbeda.

Deteksi malware memerlukan prediksi yang sangat cepat agar pengalaman pengguna tidak terpengaruh saat model menentukan apakah aman untuk meluncurkan aplikasi. Ini berarti prediksi malware perlu dilakukan dalam ratusan milidetik. Setelah deteksi malware ditentukan, eksekusi berkas tersebut akan diblokir dan analis keamanan/ancaman tidak perlu mengetahui jenis malware selama beberapa detik atau menit. Bahkan, algoritma klasifikasi bahkan tidak perlu dieksekusi di titik akhir, tetapi dapat dikirim ke server terpisah untuk klasifikasi.

Tidak hanya batasan komputasi dan memori yang sangat berbeda untuk klasifikasi malware, tetapi pemilihan fitur optimal juga sangat mungkin berbeda. Pertama, beberapa fitur yang membedakan malware dari aplikasi jinak bersifat umum di antara berbagai jenis malware dan tidak akan membantu. Fitur lain mungkin umum pada perangkat lunak jinak tetapi berguna untuk membedakan berbagai jenis malware. Terakhir, mengingat batasan yang lebih longgar untuk model klasifikasi malware, fitur yang terlalu mahal untuk dihitung pada titik akhir kini dapat digunakan dalam model klasifikasi malware. Sekali lagi, pembuatan jenis fitur baru dan eksperimen merupakan kunci untuk peningkatan berkelanjutan model klasifikasi malware.

Data Pelatihan

Seperti halnya pengembangan model Pembelajaran Mesin, model yang dihasilkan hanya sebaik data yang digunakan untuk pelatihan. Untungnya, sampel malware tersedia dalam jutaan, tetapi itu tidak cukup. Melatih pengklasifikasi biner yang baik membutuhkan sampel representatif dari perangkat lunak jinak. Sampel jinak dari penyedia perangkat lunak besar seperti Apple, Google, dan Microsoft relatif mudah diperoleh. Beberapa penyedia perangkat lunak yang lebih kecil hanya menyediakan salinan untuk pelanggan berbayar.

Aplikasi yang dikembangkan untuk penggunaan internal di perusahaan sangat sulit diperoleh. Hal ini bahkan lebih buruk untuk berkas dokumen. Sebagian besar berkas dokumen jinak dihasilkan oleh bisnis atau konsumen dan tidak tersedia untuk umum.

Lebih lanjut, deteksi malware merupakan masalah klasifikasi biner yang sangat tidak seimbang. Rasio berkas jinak terhadap berkas ganas adalah $>> 1\mathrm{M}:1$. Jika set pelatihan juga tidak seimbang, model akan bias untuk memprediksi jinak karena itu adalah jawaban yang benar $> 99{,}9$ persen dari waktu. Jadi, set pelatihan harus lebih seimbang antara berkas ganas dan jinak daripada yang ditemukan dalam kehidupan nyata. Perhatikan bahwa keseimbangan tidak boleh terlalu jauh ke arah lain atau model akan bias untuk memprediksi ganas daripada jinak. Dalam set pelatihan yang relatif seimbang ini, sampel sampel jinak dan ganas harus beragam mungkin untuk menghasilkan model yang akan memprediksi dengan baik dalam penerapan. Perawatan dan peningkatan set pelatihan yang konstan dengan menggabungkan kelas berkas yang salah diprediksi sangat penting untuk meningkatkan efikasi model deteksi malware.

Penyetelan Model Klasifikasi Malware Menggunakan Kurva Karakteristik Operasi Penerima

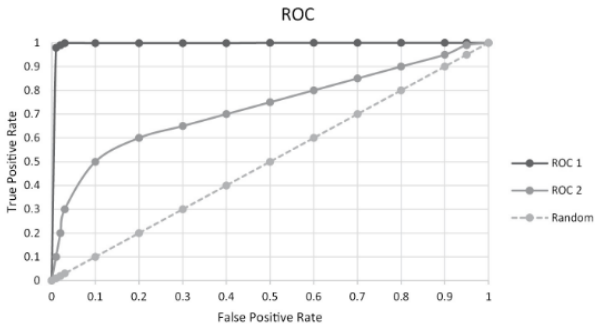
Model klasifikasi malware yang ideal akan selalu memblokir malware dan tidak pernah memblokir perangkat lunak yang tidak berbahaya. Namun, bahkan model terbaik sekalipun akan memiliki Negatif Palsu (kegagalan mendeteksi malware) dan Positif Palsu (mendeteksi perangkat lunak yang tidak berbahaya sebagai berbahaya). Jika rasio Positif Palsu terlalu tinggi, pengguna akan frustrasi karena pekerjaan mereka yang sah terganggu oleh model yang terlalu sering "menangis". Di sisi lain, model dengan rasio Negatif Palsu yang tinggi tidak jauh lebih baik daripada tidak memiliki model sama sekali. Sebagian besar teknik pemodelan memiliki tingkat keyakinan yang terkait dengan prediksinya. Tingkat keyakinan ini dapat digunakan untuk menetapkan ambang batas kapan prediksi malware menghasilkan berkas yang benar-benar diblokir (misalnya, hanya memblokir perangkat lunak ketika model memiliki keyakinan > 90 persen terhadap prediksi malware-nya).

Teknik umum yang digunakan untuk menetapkan ambang batas ini adalah dengan menggunakan kurva Karakteristik Operasi Penerima (ROC). Kurva ini dibuat dengan memplot rasio Positif Benar (malware diprediksi dengan benar) versus rasio Positif Palsu (perangkat lunak yang tidak berbahaya diprediksi sebagai berbahaya) untuk setiap ambang batas tingkat keyakinan. Gambar 1 adalah contoh dua kurva ROC yang berbeda untuk dua model yang berbeda. Sudut kanan atas mencerminkan pengaturan ambang batas sehingga semua perangkat lunak diprediksi sebagai malware (yaitu, ambang batas keyakinan malware = 0). Pada pengaturan ini, 100 persen dari semua malware akan terdeteksi, tetapi 100 persen perangkat lunak yang tidak berbahaya akan salah diidentifikasi sebagai malware. Sudut kiri bawah adalah ekstrem yang berlawanan di mana tidak ada yang terdeteksi sebagai malware (yaitu, ambang batas keyakinan = 100 persen). Sisa kurva mencerminkan dampak penyesuaian ambang batas keyakinan antara kedua ekstrem ini.

Kurva ROC memberikan indikasi visual yang kuat tentang daya prediksi suatu model. Model yang sempurna akan memiliki "kurva" ROC berupa garis vertikal pada sumbu y yang terhubung ke garis horizontal di sepanjang bagian atas plot. Tebakan acak akan menghasilkan

garis diagonal dari titik asal ke sudut kanan atas. Semakin dekat kurva ROC ke sudut kiri atas, semakin prediktif model tersebut. Faktanya, Area di Bawah Kurva (AUC) sering digunakan sebagai metrik untuk membandingkan efektivitas suatu model. Pada Gambar 1, model dengan kurva ROC 1 jauh lebih prediktif daripada model yang diwakili oleh kurva ROC 2.

Algoritma deteksi malware yang efektif perlu mencapai tingkat efikasi yang sangat tinggi. Model yang menandai perangkat lunak jinak sebagai malware 10 persen dari waktu (rasio Positif Palsu 0,1) akan sangat mengganggu bagi sebagian besar pengguna. Model ROC 2 hanya mencapai rasio Positif Benar sebesar 50 persen jika ambang batas ditetapkan untuk menghasilkan rasio Positif Palsu sebesar 10 persen. Sebaliknya, ambang batas untuk model yang diwakili oleh ROC 1 dapat ditetapkan pada rasio Positif Palsu 1 persen dan tetap mencapai rasio Positif Benar di kisaran 90 persen yang tinggi.



Pelanggan yang berbeda mungkin memiliki sensitivitas yang berbeda terhadap rasio Positif Palsu vs. Positif Benar. Misalnya, pelanggan yang canggih dan berkeamanan tinggi mungkin lebih menyukai rasio Positif Palsu yang lebih tinggi daripada rasio Positif Benar yang lebih tinggi, sehingga penyesuaian ambang batas dapat diekspos kepada pelanggan untuk dipilih. Produk perlindungan titik akhir juga dapat menawarkan tindakan yang berbeda berdasarkan ambang batas keyakinan. Jika model sangat yakin bahwa suatu sampel adalah malware, berkas tersebut dapat langsung dikarantina. Jika model sedikit kurang yakin, model tersebut dapat membiarkan berkas tersebut dan memberi tahu Analis Keamanan untuk menentukan tindakan yang harus dilakukan.

Mendeteksi Malware setelah Detonasi

Waktu teraman untuk mendeteksi malware adalah sebelum malware diizinkan untuk dieksekusi. Namun, bahkan model terbaik pun akan melewatkan deteksi beberapa malware sebelum eksekusi. Lapisan perlindungan berikutnya adalah mendeteksi malware yang sudah aktif di titik akhir. Banyak fitur dan teknik yang sama yang digunakan untuk analisis berkas statis dapat digunakan untuk menganalisis jejak dalam memori dari setiap proses aktif. Ketika malware aktif di memori, malware kemungkinan telah mendekripsi bagian terenkripsi dari kodenya yang coba disembunyikan. Hal ini membuat entropi sebagai fitur kurang bermanfaat, tetapi banyak fitur lainnya mungkin menjadi lebih efektif karena lebih banyak kode dan data malware yang sebenarnya kini terekspos secara jelas. Namun, malware apa pun yang lolos dari

deteksi dalam model analisis berkas statis memiliki peluang yang cukup besar untuk tidak terdeteksi hanya oleh fitur-fitur yang sama ini.

Setelah malware aktif, model yang mendeteksi perilaku anomali dapat lebih efektif dalam mendeteksi keberadaan malware. Dengan mengamati hal-hal seperti aktivitas CPU, memori, jaringan, berkas, dan pembaruan registri untuk aktivitas yang tidak biasa, malware dapat mengungkapkan dirinya sendiri. Beberapa aktivitas malware yang tidak biasa dapat diantisipasi seperti mengunggah sejumlah besar data, memodifikasi kunci registri sensitif, atau memperbarui area penyimpanan penting (misalnya, boot record), yang dapat dicari oleh model secara eksplisit.

Untuk aktivitas lain yang lebih halus, diperlukan model deteksi anomali. Tidak seperti model terawasi yang dijelaskan untuk deteksi dan klasifikasi malware, deteksi anomali memerlukan pendekatan pemodelan tanpa pengawasan karena sampel berlabel dari perilaku berbahaya kemungkinan besar tidak tersedia. Beberapa algoritma tanpa pengawasan meliputi:

- Algoritma pengelompokan (k-means, DBSCAN, HDBSCAN, dan lain-lain.);
- Deteksi anomali (Faktor outlier lokal, Hutan Isolasi);
- Pemodelan perilaku normal (Berbagai algoritma autoencoder jaringan saraf tiruan, ...).

Dalam semua pendekatan ini, kuncinya adalah model telah dilatih dengan data "normal" yang cukup untuk dapat mendeteksi sesuatu yang "tidak normal". Semua model ini mempelajari beberapa representasi tentang hubungan normal antara semua fitur di masa lalu. Setelah hubungan baru terdeteksi, model membuat prediksi "abnormal". Apakah kondisi abnormal ini disebabkan oleh malware atau sekadar perilaku "normal baru" yang belum pernah terlihat sebelumnya, bergantung pada Analis Keamanan untuk mengetahuinya.

Untuk titik akhir yang relatif terkunci dan biasanya melakukan hal yang sama berulang kali (misalnya pengontrol proses tertanam), deteksi anomali semacam ini bisa sangat efektif. Untuk titik akhir yang lebih umum (misalnya PC individu), tempat aplikasi baru diinstal dan situs web baru diakses, risiko Positif Palsu meningkat cukup signifikan dan dapat mengakibatkan sindrom "serangga pencemburu" yang menakutkan.

Ringkasan

Dengan tersedianya kumpulan data pelatihan yang sangat besar dan berlabel baik serta fitur-fitur relevan yang diekstraksi, pengembangan model deteksi malware menggunakan pembelajaran mesin sangat memungkinkan. Seiring dengan semakin lazimnya model-model ini pada titik akhir yang diterapkan, kita dapat yakin bahwa pembuat malware akan menemukan cara untuk menghindari deteksi dan menciptakan gelombang eskalasi berikutnya dalam perang yang tak pernah berakhir ini.

BAB 3
PEMBELAJARAN MESIN MENGGUNAKAN PYTHON

Seperti yang telah Anda lihat di seluruh buku ini, inti dari Kecerdasan Buatan dan semua subset yang berada di dalamnya (Pembelajaran Mesin, Jaringan Saraf Tiruan, dan Visi Komputer) adalah data dan himpunan tempat mereka "berada". Kecerdasan Buatan hanya sebaik himpunan data yang digunakannya untuk menghasilkan keluaran yang diinginkan.

Di Bab 1, kami mengkaji secara mendalam pentingnya data, dan kehati-hatian yang harus digunakan dalam memilih bagian data yang tepat yang dibutuhkan untuk aplikasi Kecerdasan Buatan Anda. Di paruh pertama Bab 2, kami juga mengkaji secara mendalam jenis-jenis himpunan data yang dapat digunakan, dari sudut pandang statistik komputasional. Kami juga meninjau jenis-jenis konsep statistik serta matematika yang diperlukan untuk lebih mengoptimalkan himpunan data semacam ini.

Optimalisasi dalam hal ini merupakan langkah yang sangat penting sebelum himpunan data dimasukkan ke dalam sistem Pembelajaran Mesin. Misalnya, seperti yang telah kita pelajari, jika Anda memiliki terlalu banyak data, sistem akan mengalami overtraining karena kelebihan data yang ada, dan kemungkinan besar akan menghasilkan serangkaian keluaran yang sangat miring, jauh melampaui apa yang Anda antisipasi atau harapkan.

Set data dengan terlalu banyak data juga dapat membebani pemrosesan serta daya komputasi sistem Pembelajaran Mesin. Penting untuk diingat bahwa kondisi ideal agar sistem Pembelajaran Mesin Anda menghasilkan keluaran yang diinginkan adalah dengan terus-menerus menerima set data 24/7/365. Hal ini cukup membebani sistem itu sendiri. Oleh karena itu, proses pembersihan dan pengoptimalan set data untuk membuang data yang berlebih atau tidak diperlukan merupakan keharusan mutlak.

Dalam bab-bab dalam buku ini sejauh ini, setiap kali sistem Kecerdasan Buatan atau sistem Pembelajaran Mesin digunakan sebagai titik referensi, diasumsikan bahwa suatu teknologi sudah ada, bukan dikembangkan dari awal. Dalam hal ini, dapat diasumsikan lebih lanjut bahwa sistem tersebut telah tersedia secara mudah karena siap untuk diterapkan dari platform berbasis Cloud.

Dengan kata lain, aplikasi Kecerdasan Buatan dan Pembelajaran Mesin ini tersedia sebagai penawaran "Perangkat Lunak sebagai Layanan" atau "SaaS". Namun, perlu diingat juga bahwa banyak aplikasi Pembelajaran Mesin juga dapat dibangun dari awal untuk memenuhi kebutuhan Anda yang sangat spesifik. Untuk melakukan hal ini, bahasa pemrograman Python cukup sering digunakan.

Selagi kita terus membahas berbagai jenis aplikasi, kami akan memberikan dua contoh terpisah tentang bagaimana Anda dapat membangun aplikasi Pembelajaran Mesin yang sangat sederhana menggunakan Python di dua sektor pasar yang berbeda:

- Sektor Kesehatan;
- Sektor Jasa Keuangan.

Penggunaan Pemrograman Python di Sektor Kesehatan

Mengingat dunia yang kita tinggali saat ini dengan COVID-19 dan dampaknya terhadap manusia dan ekonomi di seluruh dunia, banyak orang kehilangan pekerjaan, dan banyak lainnya dirumahkan, tanpa jaminan bahwa pekerjaan mereka akan tetap tersedia setelah situasi mulai membaik. Dalam hal ini, penggunaan chatbot kini menjadi penting, tidak hanya untuk aplikasi e-commerce dan toko online, tetapi juga untuk industri kesehatan.

Bahkan, perangkat teknologi ini kini digunakan untuk membantu dokter dan perawat di UGD, pada tingkat tertentu, mendiagnosis pasien dan bahkan memberikan beberapa rekomendasi perawatan. Di bagian ini, kita akan membahas lebih lanjut cara membangun chatbot yang sangat sederhana menggunakan bahasa Python. Namun, pertama-tama, penting untuk memberikan gambaran umum tentang bagaimana Pembelajaran Mesin dan chatbot digunakan bersama-sama saat ini.

3.1 PEMBELAJARAN MESIN DALAM CHATBOT

Penting untuk diingat bahwa dengan chatbot, Anda dapat berinteraksi dengannya dengan salah satu dari dua cara, dan bahkan mungkin keduanya:

- Obrolan teks;
- Perintah suara.

Untuk mengakomodasi kedua skenario ini, chatbot memanfaatkan konsep Pembelajaran Mesin (ML) dan Pemrosesan Bahasa Alami (NLP). Pembelajaran Mesin digunakan untuk menciptakan jawaban dan respons cerdas atas pertanyaan Anda saat Anda terlibat dalam percakapan dengannya.

Salah satu keuntungan utama menggunakan Pembelajaran Mesin dalam aspek ini adalah ia benar-benar dapat mempelajari Anda seiring Anda terus berinteraksi dengannya selama periode waktu tertentu.

Misalnya, ia membangun profil Anda dan melacak semua percakapan Anda sehingga dapat diakses dalam hitungan detik untuk sesi obrolan berikutnya, dan kemudian, bahkan dapat mengantisipasi pertanyaan yang mungkin Anda ajukan sehingga dapat memberikan jawaban terbaik yang sesuai dengan kebutuhan Anda.

Dengan cara ini, Anda tidak perlu lagi mengetik informasi yang sama berulang kali.

NLP adalah cabang lain dari AI, dan ini adalah alat yang terutama digunakan jika Anda terlibat dalam percakapan vokal yang sebenarnya dengan chatbot. NLP dapat dengan mudah mereplikasi berbagai pola bicara manusia untuk menghasilkan nada suara yang realistis saat merespons Anda. Baik Anda menggunakan salah satu atau kedua metode komunikasi ini, penting untuk dicatat bahwa chatbot semakin canggih hampir setiap hari.

Alasan utamanya adalah karena chatbot menggunakan kombinasi algoritma statistik yang sangat canggih dan teknik pemodelan tingkat tinggi, serta konsep penambangan data. Karena itu, chatbot sekarang dapat berinteraksi secara sangat proaktif dengan Anda, alih-alih Anda harus memimpin percakapan, sehingga percakapan mengalir hampir mulus.

Sebagai hasil dari penggunaan Pembelajaran Mesin dan NLP, chatbot kini digunakan dalam berbagai jenis aplikasi, beberapa di antaranya meliputi:

- Situs web pedagang yang memanfaatkan toko online;
- Aplikasi seluler yang dapat digunakan di perangkat Android atau iOS Anda;
- Platform perpesanan;
- Riset pasar terkait peluncuran produk dan layanan baru;
- Pembuatan prospek;
- Kesadaran merek;
- Jenis skenario e-commerce lainnya;
- Layanan pelanggan (ini mungkin penggunaan terbesarnya sejauh ini);
- Layanan kesehatan (terutama untuk membuat janji temu dengan dokter Anda);
- Pengiriman konten.

Keunggulan Strategis Pembelajaran Mesin dalam Chatbot

Seperti yang dapat disimpulkan, ada banyak keuntungan menggunakan pendekatan semacam ini untuk bisnis Anda. Beberapa di antaranya adalah sebagai berikut:

1) Anda memiliki perwakilan penjualan yang siaga 24/7/365:
   Seperti yang telah disebutkan sebelumnya, tidak diperlukan keterlibatan manusia jika Anda memiliki chatbot berbasis AI. Oleh karena itu, Anda memiliki agen yang dapat bekerja kapan saja, siang dan malam, untuk membantu menjual produk dan layanan Anda secara real-time. Dengan kata lain, chatbot tidak akan pernah lelah dan akan selalu bersemangat melayani!
2) Mengurangi biaya:
   Dengan menggunakan chatbot, Anda bahkan mungkin tidak perlu mempekerjakan staf layanan pelanggan purnawaktu. Dengan demikian, Anda akan dapat menghemat laba dengan tidak perlu membayar gaji dan tunjangan. Namun perlu diingat, Anda tidak boleh menggunakan chatbot sebagai pengganti penuh tim layanan pelanggan Anda. Pada suatu saat, Anda akan membutuhkan beberapa dari mereka untuk membantu menyelesaikan masalah atau pertanyaan kompleks jika chatbot tidak dapat melakukannya.
3) Tingkat kepuasan pelanggan yang lebih tinggi:
   Mari kita akui, dalam masyarakat kita, kita ingin memiliki segalanya sekarang dan di sini. Kita tidak sabar ketika harus menunggu, bahkan beberapa menit saja, untuk berbicara dengan perwakilan layanan pelanggan di saluran lain. Namun, dengan menggunakan chatbot, waktu tunggu ini berkurang menjadi hanya beberapa detik, sehingga menghasilkan pelanggan yang jauh lebih puas dan lebih banyak pelanggan tetap.
4) Retensi pelanggan yang lebih baik:
   Ketika Anda mampu memberikan jawaban atau solusi yang sangat dibutuhkan kepada pelanggan dan calon pelanggan yang putus asa, ada kemungkinan yang jauh lebih tinggi bahwa Anda akan dapat mempertahankan mereka untuk jangka panjang. Di sinilah chatbot berperan. Ingat, Anda mungkin memiliki merek yang kuat, tetapi itu akan segera hilang dengan cepat jika Anda tidak dapat memenuhi kebutuhan hanya dalam hitungan menit.

5) Anda dapat menjangkau batas internasional:

Di dunia e-commerce saat ini, tidak ada batasan internasional. Pelanggan atau calon pelanggan adalah mereka yang dapat membeli produk dan layanan Anda dari lokasi geografis mana pun mereka berada. Jika Anda mencoba melakukan ini dengan model perwakilan layanan pelanggan tradisional, bukan hanya akan mahal, tetapi perwakilan tersebut juga harus dilatih dalam bahasa lain. Faktor gangguan juga dapat muncul dengan cepat jika perwakilan pelanggan tidak dapat berbicara dalam bahasa yang diinginkan dengan nada dan format yang konsisten. Namun, chatbot berbasis AI meringankan semua masalah ini karena sudah dilengkapi dengan fungsi pemrosesan bahasa asing.

6) Dapat membantu memilah kasus:

Jika bisnis Anda cukup besar sehingga membutuhkan pusat panggilan khusus untuk mendukungnya sepenuhnya, kemungkinan besar perwakilan layanan pelanggan Anda akan dibombardir dengan panggilan telepon dan kesulitan untuk meresponsnya. Jika Anda menggunakan chatbot di sini, Anda dapat menggunakannya untuk membantu menyelesaikan berbagai masalah dan pertanyaan, mulai dari yang sederhana hingga yang lebih rumit. Namun, jika ada sesuatu yang jauh lebih rumit dan chatbot tidak dapat menyelesaikannya, chatbot juga memiliki fungsi untuk mengarahkan percakapan tersebut ke perwakilan yang tepat yang dapat menanganinya. Dengan kata lain, chatbot juga dapat memilah percakapan dengan pelanggan dan calon pelanggan jika diperlukan.

Matrix Pembelajaran Mesin dan Chatbot

Matriks berikut menggambarkan keunggulan penggunaan chatbot berbasis AI dibandingkan asisten virtual tradisional:

Fungsionalitas	***AI-driven Chatbot***	***Asisten Virtual***
FAQ mudah dijawab	Ya	Ya
Dapat memahami pertanyaan yang rumit	Ya	Tidak
Dapat membuat respons yang disesuaikan dan personal	Ya	Tidak
Dapat mempelajari lebih banyak tentang Anda dari percakapan sebelumnya	Ya	Tidak
Dapat meningkatkan percakapan di masa mendatang dengan Anda	Ya	Tidak

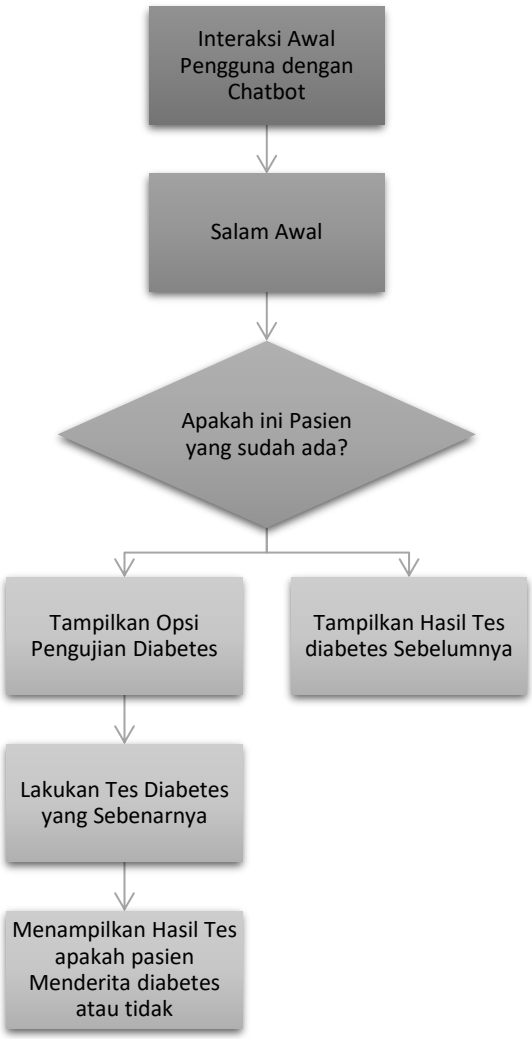
3.2 MEMBANGUN CHATBOT—KONTEKS PENGUJIAN DIABETES

Dalam contoh khusus ini, peran utama chatbot adalah untuk membantu menyapa pasien tertentu dan memandu mereka melalui serangkaian pertanyaan agar mereka dapat mengirimkan tes darah untuk menentukan apakah individu ini menderita diabetes atau tidak. Penting untuk diingat bahwa bukan chatbot yang akan benar-benar melakukan tes semacam ini, melainkan pasien yang harus duduk terpisah di depan mesin pengujian otomatis agar tes darah dapat dilakukan.

Sebagaimana telah dijelaskan secara panjang lebar dalam dua bab pertama ini, sangat penting untuk membuat Pohon Keputusan terlebih dahulu, karena ini akan memandu tim pengembangan perangkat lunak dalam membuat modul perangkat lunak, serta kode sumber

yang ada di dalamnya. Karena contoh yang kami berikan di subbagian ini cukup sederhana, Pohon Keputusan yang dihasilkan pun relatif mudah.

Berikut ini menggambarkan Pohon Keputusan ini:



Modul Inisialisasi

Kode Python inisialisasinya adalah:

Instal menggunakan perintah berikut:

```
Import nltk
Nltk.download (‘wordnet’)
[nltk_data] Downloading package wordnet to
[nltk_data] C:\Users\PMAUTHOR\Appdata\Roaming\nltk_data …
[nltk_data] Unzipping corpora\wordnet.zip
Out[4]: True
Import nltk
Nltk.download (‘punkt’)
```

Catatan: Kode sumber Python di atas akan menampilkan pustaka Antarmuka Pengguna Grafis (GUI) dan berbagai gambar agar pasien dapat berinteraksi dengan lancar dengan chatbot. Juga disertakan fungsi "dormant" khusus yang akan menidurkan chatbot jika tidak digunakan secara aktif dalam waktu lama.

Modul Antarmuka Pengguna Grafis (GUI)

Paket berikutnya adalah kode sumber yang akan membantu membuat GUI yang disebutkan di atas untuk membantu pasien:

```
# -*- coding: utf-8 -*-
“””
@author: RaviDas
“””
#Loading tkinter libraries which will be used to in the GUI of the
medial chatbot
Import tkinter
From tkinter import *
From tkinter.scrolledtext import *
From tkinter import ttk
Import time
From PIL import Image Tk, Image
Import tkinter
#Loading random choices in our Chatbot program
Import random
```

Modul Layar Percikan

Setelah GUI awal ditampilkan kepada pasien melalui kode sumber Python yang dijelaskan di atas, langkah selanjutnya adalah membuat "Layar Percikan" yang akan menyambut pasien di rumah sakit medis tersebut. Kode sumber Python untuk melakukannya adalah sebagai berikut:

```
#Splash Screen
Splash = tkinter.Tk ()
Splash.title (“Welcome to this Diabetes Testing Portal, brought to you by Hospital XYZ”)
Splash.geometry (“1000 X 1000”)
Splash.configure (background = ‘green’)
W = Label(splash, text = “Hospital XYZ Diabetes Testing Portal\nloading …, font = “Helvetica”,
  26), fg = “white”, bg = “green”
w.pack ()
splash.update ()
time.sleep (6)
splash.deiconify ()
splash.destroy ().
```

Modul Ucapan Selamat Datang Pasien

Setelah GUI sambutan keseluruhan ditampilkan kepada pasien, langkah selanjutnya adalah membuat jendela khusus yang secara khusus menyapa pasien, menggunakan nama depan dan belakang mereka. Kode sumber Python untuk melakukan ini adalah sebagai berikut:

```
#Initializing tkinter library for GUI Window show up window =
tkinter.Tk ()
S = tkinter.Scrollbar (window)
Chatmsg.focus_set ()
s.pack (side = tkinter.RIGHT, fill = tkinter.Y)
chatmsg.pack (side = tkinter.TOP, fill = tkinter.Y)
s.config (command = chatmsg.yview)
chat.config (yscrollcommand = s.set)
input_user = String Var ()
input_field = Entry (window, text = input_user_
input_field.pack (side = tkinter.BOTTOM, fill = tkinter.X)
bot_text = “Welcome to the Hospital XYZ Diabetes Testing Portal\n”
chatmsg.insert (INSERT, ‘Bot:%s\n’ % bot_text)
bot_text = “Press enter to continue ”
chatmsg.insert (INSERT, ‘Bot:%s\n’ % bot_text)
chat.msg.focus ().
```

Modul Korpus Diabetes

Dalam skenario dan aplikasi dunia nyata, terutama yang berkaitan dengan Pemrosesan Bahasa Alami (NLP), terdapat konsep yang dikenal sebagai "Korpus". Sederhananya, ini hanyalah kumpulan jargon dan leksikon terkait lainnya yang digunakan oleh industri tertentu. Jadi, dalam contoh chatbot kami yang menggunakan pemrograman Python, akan terdapat banyak istilah medis yang digunakan jika chatbot ini benar-benar diterapkan di dunia nyata, seperti di ruang praktik dokter, pusat rawat jalan, atau bahkan di rumah sakit.

Membahas setiap jenis terminologi medis yang dapat digunakan dengan chatbot medis berada di luar cakupan buku ini, tetapi untuk memberikan contoh bagaimana chatbot tersebut dapat dibuat dalam bahasa pemrograman Python, berikut ini menunjukkan cara membuat apa yang dikenal sebagai "Korpus Diagnostik" untuk memeriksa pasien yang berpotensi menderita diabetes:

```
#Diagnostics Corpus for medical chatbot
Greet = [‘Hello, welcome to the Hospital XYZ Diabetes Testing Portal’, ‘Hi, welcome to the
    Hospital XYZ Diabetes Testing Portal’, ‘Hey, welcome to the Hospital XYZ Diabetes Testing
    Portal’, ‘Good Day, welcome to the Hospital XYZ Diabetes Testing Portal’]
Confirm = [‘Yes’, ‘Yay’, ‘Yeah’, ‘Yo’]
Membered = [‘12345’, 12346’, 12347’, 12348’, ‘12349’]
Customer = [‘Hello’, ‘Hi’, ‘Hey’]
Answer = [‘Please select one of the options so that I can help you’, ‘I truly understand and
    sympathize with your anxieties, but please input an appropriate response’]
```

```
Greetings = ['Hola, welcome to the Hospital XYZ Diabetes Testing Portal again', 'Hello, welcome
    to the Hospital XYZ Diabetes Testing Portal again', 'Hey, welcome to the Hospital XYZ
    Diabetes Testing Portal', 'Hi, welcome to the Hospital XYZ Diabetes Testing Portal']
Question = ['How are you?', 'How are you doing?']
Responses = ['I am OK', 'I could be doing better', 'I feelsick', 'I feel anxious', 'I am fine']
Another = ["Do you want another Diabetes Test?"]
Diabetes tests = ['Type 1 for the hbAic Test', 'Type 2 for the Blood Viscosity Test', 'Type 3 for the
    Heart Rate Test', 'Type 4 for the Blood Oxygen Test', 'Type 5 for the Blood Pressure Test']
Testresponse = ['1', '2', '3', '4', '5', '6']
```

CATATAN: Seperti yang Anda lihat dari kode sumber pemrograman Python sederhana di atas, terdapat berbagai jenis respons yang diberikan kepada pasien, tergantung pada kedalaman kemampuan berbahasa mereka, dan kosakata yang mereka gunakan dalam percakapan sehari-hari dengan orang lain. Sebagaimana telah disebutkan, ini hanyalah contoh sederhana, dan jika ini benar-benar diterapkan dalam lingkungan medis dunia nyata, banyak respons lain yang juga harus dimasukkan ke dalam kode sumber Python. Selain itu, bahasa asing lainnya juga harus diprogram, terutama bahasa Spanyol.

Modul Chatbot

Dalam modul khusus ini, kami akan mengkaji lebih lanjut konstruksi dasar dari sumber yang kini membentuk aplikasi Chatbot Diabetes yang sebenarnya:

```
#Global variable to check first time greeting
Firstswitch = 1
Newid = '12310'
Memid = 0
Def chat (event):
    Import time
    Import random
    Global memid
    Condition= ""
    #Greet for first time
    Global firstswitch
If (firstswitch==1):
       Bot_text = random.choice (greet0
       Chatmsg.insert (INSERT, 'Bot:%s\n' %bot_text)
       Bot_text = "If you are an existing patient of Hospital XYZ, please enter in your Patient ID: or
enter no if you are a new patient"
      Chatmsg.insert (INSERT, 'Bot:%s\n' %bot_text)
      Firstswitch = 2
If (firstswitch = 1):
       Input_get = input_field.get().lower()
       If any (srchstr in input_get for srchstr in membered):
            Memid = input_get
            Bot_text = "Thank you for being a loyal and dedicated patient of Hospital XYZ\n Please
               choose the type of service that is most suited for your visit this time from the following
               menu in order to continue with the diagnostics procedure\ nType 1 for the hbAic Test\
               nType 2 for the Blood Viscosity Test\ nType 3 for the Heart Rate Test'\ nType 4 for
               the Blood Oxygen Test\ nType 5 for the Blood Pressure Test'\ nType 6 to Exit the
               Diabetes Testing Portal\n\n"
Elif (input_get=="no"):
     Memid = newid
```

```
      Bot_text = “Your new Member Identification Number is: ” + newid + “Please remember this
         for future reference since you are a new patient. \n Please choose the type of service that
         is most suited for your visit this time from the following menu in order to continue with the
         diagnostics procedure\ nType 1 for the hbAic Test\ nType 2 for the Blood Viscosity Test\
         nType 3 for the Heart Rate Test’\ nType 4 for the Blood Oxygen Test\ nType 5 for the
         Blood Pressure Test’\ nType 6 to Exit the Diabetes Testing Portal\n\n”
Elif any (srchstr in input_get for srchstr in testresponse):
      Bot-text = “Please place any of your fingers up on the Fingerprint Panel  as indicated above
         in order to proceed with your Diabetes Test”
      Chatmsg.insert (INSERT, ‘Bot:%s\n’ % bot_text)
      Delaycounter = 0
      For delaycounter in range (0,10):
           Bot_text = str (delaycounter)
           Time.sleep (1)
           Chatmsg.insert (INSERT, ‘Bot:%s\n’ % bot_text)
      Bot_text = “Please wait for a few minutes, we are analyzing your Diabetes Test, and will
         present you with the results shortly\n”
      Chatmsg.insert (INSERT, ‘Bot:%s\n’ % bot_text)
          Time.sleep(2)
          If (input_get==”1):
                Hba1c = random.randit (4, 10)
                Bot_text = “Member Identification Number:” +  str(memidnum) + “Your hbaa1c
                Test resultis: ”+ str(hba1c)
                If (hba1c>=4 and hbaic<=5.6):
                      Condition= “You do not have Diabetes”
                Elif (hba1c>5.7 and hba1c<=6.4):
                       Condition = “You are prediabetic, please consult your  Primary Care
                          Physician as soon as possible”
                Elif (hba1c>6.5):
                      Condition = “You are diabetic, please consult your  Primary Care Physician as
                        soon as possible”
                Bot_text = bot_text +” Your condition is: “+condition
                Chatmsg.insert (INSERT, ‘Bot:%s\n % bot_text)
                Elif (input_get==2):
                       Viscosity=random.randit (20,60)
                       Bot_text = “Member Identification Number: “ +  str(memidnum) + “Your
                          Blood Viscosity Level test result is “ +str(viscosity)
               Elif (input_get==3):
                       Viscosity=random.randit (20,60)
                       Bot_text = “Member Identification Number: “ +  str(memidnum) + “Your
                          Heart Rate Level test result is “ +str(heatrate)
            Elif (input_get==4):
                      Viscosity=random.randit (20,60)
                      Bot_text = “Member Identification Number: “ +  str(memidnum) + “Your
                         Blood Oxygen test result is “ +str(oxygen)
            Elif (input_get==5):
                  Systolic = random.randit (80,200)
                  Diastolic = random.randit (80,110)
                  Bot_text = “Member Identification Number: “ +  str(memidnum) + “Your
                     Blood Pressure Level test result is: Systolic: “ +str(systolic)”
                  “Diastolic:“ + str(diastolic)
            Elif (input_get==6):
                  Import sys
                  Window.deiconfy ()
                  Window.destroy ()
                  Sys.exit (0)
                  Else:
```

```
        From nltk.stem import WordNetLemmatizer
Import nltk
If ((not input_get) or (int(input_get)<=0)):
      Print (“Did you just press Enter?”) #print some info
Else:
      Lemmatizer = WordNetLemmatizer()
      Input_get = input_field.get().lower()
      Lemvalue = lemmatizer.lemmatize(input_get)
      Whatsentiment = getSentiment(lemvalue)
      If (whatsentiment==”pos”):
            Bot_text = answer[0]
            #print (“Positive Sentiment”)
      Elif (whatsentiment==”neg”):
            Bot_text = answer[1]
      #print (“Negative Sentiment”)
      Chatmsg.insert (INSERT, ‘%s\n’ % lemvalue)
      #bot_text = “I do not understand what you mean!”
 Chatmsg.insert (INSERT, ‘%s\n’ % lemvalue)
 #label = Label(window, text = input_get)
 Input_user.set(“)
 #label.pack()
 Return “break”
```

Modul Analisis Sentimen

Perlu dicatat bahwa komponen kunci chatbot yang memanfaatkan Pembelajaran Mesin adalah apa yang dikenal sebagai "Analisis Sentimen". Secara teknis, hal ini dapat didefinisikan sebagai berikut: *Analisis sentimen adalah penambangan kontekstual teks yang mengidentifikasi dan mengekstrak informasi subjektif dalam materi sumber, dan membantu bisnis memahami sentimen sosial merek, produk, atau layanan mereka sambil memantau percakapan daring. Namun, analisis aliran media sosial biasanya terbatas pada analisis sentimen dasar dan metrik berbasis hitungan.*

Seperti yang dapat dilihat dari definisi di atas, pada tingkat yang paling sederhana, tujuan penggunaan Analisis Sentimen adalah untuk mencoba mengukur, dengan kepastian ilmiah, suasana hati calon pelanggan atau pelanggan. Dalam contoh kita, premis dasarnya adalah untuk mengukur secara tepat (hingga tingkat kepastian tertentu) bagaimana perasaan pasien yang ada dan/atau pasien baru. Penting untuk diingat bahwa Analisis Sentimen bisa sangat rumit, dan menerjemahkan semua itu ke dalam chatbot mode produksi akan membutuhkan banyak baris kode sumber Python.

Namun, untuk keperluan chatbot yang sedang kita bangun di subbagian ini, kita akan menunjukkan secara sederhana seperti apa kode sumber Python-nya:

```
#Sentiment Analyzer using NLP
Def getSentiment(text):
    Import nltk
    From nltk.tokenize import word_tokenize
    #nltk,download (‘punkit’)
    #Step1 – Training data building from the Diabetes Corpus Module
```

```
    Train = [(thanks for an outstanding diabetes report", "pos"),
    ("Your service is very efficient and seamless", "pos"),
    ("As a patient, I am overall pleased with the services that have been provided", "pos")
    ("I did not know that I actually had Diabetes until after I took this series of tests", "neg",)
    ("The service could have been a little bit quicker—perhaps too much to be processed", "neg"),
    ("Hospital XYZ was not easy for me to find", "neg"),
    ("Hospital XYZ was very easy for me to find", "pos"),
    ("I do not quite believe the results of the tests that were conducted—I will seek a second
       medical opinion", "neg"),
    ("I wish there was more human contact at Hospital XYZ, everything seems to be too
  automated", "neg"),
    ("Can I actually talk to a human medical expert here?!", "neg"),
    ("The test results from the Diabetes tests are good", "pos"),
    ("Hospital XYZ has a good level of medicalservice", "pos"),
    ("Hospital XYZ has a great level of medicalservice", "pos"),
    ("Hospital XYZ has a superior level of medicalservice", "pos"),
    ("Hospital XYZ has an amazing array of medical technology", "pos"),
    ("This Diabetes Report cannot be true by any means", "neg"),
    ("This testing procedure will be very expensive for me—I am not sure if my medical insurance
       will even cover this", "neg"),
    ("I cannot believe that I have Diabetes based upon this report", "neg"),
    ("Does this mean I have to take special Diabetic medication and prescriptions?", "neg"),
    ("Will I have to take either injections or oral medication on a daily basis?", "neg"),
    ("My lipids are getting much worse than expected—should I see my Primary Care
        Physician?", "neg"),
    ("Hospital XYZ has very poor level of service", "neg"),
    ("Hospital XYZ has a poor level of service", "neg"),
    ("Hospital XYZ has a bad level of service", "neg"),
    ("Hospital XYZ is extremely slow with service and medical report processing", "neg"),
    ("Hospital XYZ is very slow with service and medical report processing", "neg"),
    ("Hospital XYZ is slow with service and medical report processing", "neg"),
    ("My Diabetes actually got worst with these tests than with previous ones", "neg"),
    ("I don't believe this Diabetes Report", "neg"),
    ("I don't like the sound of this Diabetes Report", "neg"),
    ("I am in Diabetes Limbo here", "neg"),
#Step 2 Tokenize the words to the dictionary
    Dictionary = set(word.lower() for passage in train for word in word_
    Tokenize (passage[0]))
#Step 3 Locate the word in training data
    T = [({word: (word in word_tokenize(x[0])) for word in dictionary),
    X[1]) for x in train]
```

```
#Step 4 – the classifier is trained with sample data
    Classifier = nltk.NaiveBayesClassifer.train(t)
    Test_data = “oh my gosh what is this???”
    Test_data_features = {word.lower(): (word in word_tokenize (test_data.
    Lower ()))) for word in dictionary}
    Print (classifer.classify(test_data_features))
    Return classifier.classify(test_data_features)
#Start the program chat and put in loop
Input_field.bind (“<Return>”, chat)
Tkinter.mainloop()
```

Secara keseluruhan, bagian ini telah mengkaji penggunaan kode sumber Python untuk membangun, pada dasarnya, prototipe chatbot yang sangat primitif yang memanfaatkan Pembelajaran Mesin dalam lingkungan medis. Penting untuk diingat bahwa chatbot yang digunakan di dunia nyata dalam mode produksi sebenarnya membutuhkan jutaan baris kode sumber Python, mengingat kedalaman dan kompleksitas aplikasi yang dimaksud.

3.3 MEMPREDIKSI PERGERAKAN HARGA SAHAM

Mungkin salah satu penggunaan terbesar Kecerdasan Buatan adalah dalam Industri Keuangan. Dalam hal ini, Kecerdasan Buatan paling sering digunakan untuk mencoba memprediksi pergerakan harga saham sehingga pedagang keuangan, manajer dana lindung nilai, manajer reksa dana, dan lain-lain. dapat melakukan perdagangan yang menguntungkan, tidak hanya agar mereka dapat menghasilkan lebih banyak uang dalam portofolio yang mereka kelola, tetapi juga untuk memastikan bahwa klien mereka melakukan hal yang sama.

Ini sebenarnya adalah bidang yang sebaiknya diserahkan kepada Jaringan Saraf Tiruan, tetapi Pembelajaran Mesin juga dapat digunakan dengan baik. Di bagian ini, kami membangun model berbasis Python yang sangat sederhana untuk membantu memprediksi potensi pergerakan harga saham di masa mendatang. Penting untuk diingat bahwa tidak ada sistem yang pernah atau akan pernah memprediksi pergerakan seperti ini dengan tingkat akurasi 100 persen.

Hal terbaik yang dapat dilakukan seseorang adalah mencoba memperkirakan kisaran penurunan harga saham di masa mendatang, dan dari sana, membuat ekstrapolasi yang paling akurat. Oleh karena itu, dalam hal ini, konsep statistik sering kali digunakan, seperti Analisis Rata-Rata Bergerak dan Regresi Berganda.

Modul Akuisisi Harga S&P 500

Sebelum Anda dapat mulai menulis kode sumber Python, Anda perlu mengakses Umpan Harga Pasar Saham terlebih dahulu. Dalam hal ini, Anda memerlukan API yang dapat terhubung langsung dan terintegrasi dengan kode sumber Python. Untuk membangun rangkaian modul ini, Anda perlu mendapatkan Pandas Data Reader, yang tersedia di tautan ini:

pandas-datareader.readthedocs.io/en/latest/remote_data.html

Kode sumber Python di bawah ini menunjukkan cara memuat data S&P 500, dan memberikan harga saham relevan yang Anda butuhkan:

```
#-“ -coding: utf-8 -*-
AUTHOR: RaviDas
====
Input numpy as np
Import pandas as pd
#import pandas.io.data as web
From pandas_datareader import data, wb
Sp500 = data.DataReader (‘^GSPC, data_source= ‘yahoo’, start=’5/18/2020’
End = ‘7/1/2020’)
#sp500 = data.DataReader (‘^GSPC, data_source= ‘yahoo’)
Sp500.ix [‘5/18/2020’]
Sp500.info()
Print(sp500)
Print(sp500.columns)
Print(sp500.shape)
Import matplotlib.pyplot as plt
Plt.plot(sp500[‘Close’])
#now calculating the 42nd day as well as the 252 day trend for the index
Sp500[‘42d’ = np.round(pd.rolling_mean(sp500[‘Close’], window=42),2)
Sp500[‘252d’ = np.round(pd.rolling_mean(sp500[‘Close’], window=252),2)
#Look at the Data
Sp500[[‘Close’, ‘42d’, ‘252d’]].tail()
Plt.plot(sp500[[‘Close’, ‘42d’, 252d’]])
```

Memuat Data dari API

Modul berikut menggambarkan kode sumber Python untuk memuat lebih banyak data keuangan dari API yang ditentukan, seperti yang dijelaskan di subbagian terakhir:

Pip install Quandi

```
#-*-coding: utf-8 -*-
@author: RaviDas
Import quandl
Quandl.ApiConfig.api_key = ‘INSERT YOUR API KEY HERE’
# get the table for daily stock prices and,
# filter the table for the selected tickers, columns within a time range
# set paginate to True because Quandl limits the tables from the API to 10,000
per call
Data = quandl.get_table (‘WICKI/PRICES’, ticker = [‘AAPL’, ‘MFST’, ‘WMT’]
    Qopts = {‘colums’: [‘ticker’, ‘date’, ‘adj_close’]},
    Date = {‘gte’: ‘5-18-2020’, ‘lte’: ‘5-18-2020”],
         Paginate=True)
Data.head()
# create a new dataframe with ‘date’ column as index
Now = data.set_index(‘date’)
```

```
#use pandas pivot function to sort aj_close by tickers
Clean_data = new.pivot (columns='ticker')
#check the head of the output
Clean_data.head()
Import Quandl
Quandl.ApiConfig.api_key = 'z1bx8q275VanEKSOLJwa'
Quandl.AiConfig.api_version = '5-18-2020'
Import Quandl
Data = qunadl.get ('NYSE/MSFT')
Data.head()
Data.columns
Data.shape
#This stores the stock price data in a flat file
Data.to_csv("NYSE_MSFT.csv")
#A basic statistical plot of the MSFT price data over the certain timespan
Data['Close'].plot()
```

Modul Prediksi Harga Saham Hari Berikutnya Berdasarkan Harga Penutupan Hari Ini

Sesuai dengan judul subbagian ini, Anda menggunakan informasi saham keuangan yang telah Anda masukkan di modul sebelumnya untuk mencoba memperkirakan harga saham tertentu saat NYSE dibuka keesokan paginya berdasarkan harga penutupan hari sebelumnya:

```
Import numpy as np
Import pandas as pd
Import os
#Change your directory to wherever the actual financial dataset is stored at
Os.chdir ("E:\\") # Change this to your directory path or wherever you downloaded the
financial
   stock price information from the API based dataset.
#Loading the dataset of the particular company for which the prediction is replaced
Df=pd.read_csv ("StockPriceSP500DDataset.csv", parse_dates=['Date'])
Print(df.head(1))
Print(df.columns)
Out[*]
Unamed: 0 Date Opening Price High Low Closing Price Total Number of Shares Traded
Index ([u'Unamed: 0', u'Date', u'Opening Price' u'High', u 'Low', u'Closing Price',
u'Total
    Number of Shares Traded'
Dtype = 'object'
Df.shape
```

Modul Optimasi Data Keuangan (Pembersihan)

Dalam modul ini, data keuangan yang telah dikumpulkan dari API SP500 (sebagaimana telah diulas sebelumnya) kini "dibersihkan" untuk memberikan pembacaan harga saham keuangan di masa mendatang yang lebih akurat:

```
#Checking to see if any financial data optimization, or clean-up is further
required
Df.isnull().any()
#df=df.dropna()
#df=df.replace("NA", 0)
Df.types
Out[96]:
Date datetime64[ns]
Open float64
Close float 64
Dtype: object
```

Plotting Data Keuangan SP500 Tahun Sebelumnya + Satu Bulan

Sesuai dengan judulnya, modul ini memplot data keuangan SP500 yang ditentukan dari tahun sebelumnya, dengan jeda waktu satu bulan:

```
#Now plot the SP500 financial data for just the entire previous year and one month
Df['Date'].dt.year==2019
Mask=(df['Date'] > 1-1-2019 & (df['Date']] <= '12/31/2018')
Print(df.loc[mask])
Df2018=df.loc[mask]
Print(df2018.head(5)]
Plt.plot(df2018['Date'], df2018['Close'])
```

Plotting Data Keuangan SP500 Selama Satu Bulan

Modul ini memplot data keuangan SP500 yang ditentukan hanya untuk rentang waktu satu bulan:

```
#Plotting the last 1 month data from the SP500
Mask = (df['Date'] > '12-13-2017') & (df['Date'] <= '12-24-2018')
Print(df.loc[mask])
Dfdec2017=df.loc[mask]
Print(dfdec2018.head(S))
Pt.plot(dfdec2018['Date'], dfdec2018['Close'])
```

Menghitung Rata-Rata Bergerak Saham SP500

Seperti yang telah disebutkan sebelumnya di bagian ini, salah satu alat statistik yang digunakan untuk membantu memprediksi harga saham di masa mendatang adalah Rata-Rata Bergerak. Kode sumber Python berikut menunjukkan cara melakukannya:

```
#Now calculating the Moving Average of A Stock In The SP500
#Simple Moving Average Of Just One Year
Df2019['SMA'] = df2018['Close'].rolling(window=20).mean()
Df2019.head(2S)
Df2018[['SMA', 'Close']].plot().
```

Menghitung Rata-Rata Pergerakan Saham SP500 Hanya untuk Jangka Waktu Satu Bulan

Kode sumber Python di bawah ini hampir sama dengan modul sebelumnya, tetapi hanya untuk satu bulan:

```
# Now calculating the Moving Average of A Stock In The SP500 for just a one month time span
Dfdec2019['SMA'] = dfdec2019['Close'].rolling(window=2).mean()
Dfdec2019.head(25)
Dfdec2019[['SMA', 'Close']].plot()
```

Pembuatan Kolom NextDayOpen untuk Prediksi Harga Keuangan SP500

Meskipun semua modul sebelumnya penting, modul ini lebih krusial karena merupakan langkah selanjutnya sebelum prediksi harga keuangan SP500 yang sebenarnya dapat dilakukan:

```
#Now creating the NextDayOpen Column for the SP500 stock price prediction
Ln=len(df)
Lnop=len(df['Open'])
Print(lnop)
Ii=o
Df['NextDayOpen']=df['Open']
Df['NextDayOpen]=0
For I in range(o,ln-1):
  Print("Open Price: ", df['Open'][i]
  If i!=0
     Ii=i-1
Df['NextDayOpen'] [ii]=df['Open] [i]
Print(df['NextDayOpen'][ii])
```

Memeriksa Korelasi Statistik yang Ada di Kolom NextDayOpen untuk Prediksi Harga Saham SP500

Penting untuk dicatat bahwa sebelum informasi harga saham SP500 dapat diprediksi, sangat penting untuk memeriksa apakah terdapat korelasi statistik dengan harga yang telah dikumpulkan pada modul sebelumnya. Alasan utamanya adalah jika ada korelasi, hal tersebut dapat sangat mendistorsi prediksi harga untuk saham tertentu. Oleh karena itu, hal ini harus diperiksa dengan cermat, seperti yang ditunjukkan oleh kode sumber Python berikut:

```
#Checking to determine if there is any statistical correlation from the financial information collected
  by the last module
Dfnew=df[['Close', 'NextDayOpen']]
Print(dfnew.head(5))
Dfnew.corr()
Out[110];
In [111];
```

Pembuatan Model Regresi Linier untuk Memprediksi Data Harga SP500 di Masa Mendatang

Dalam modul kode sumber Python terakhir ini, kita akan sampai pada langkah terakhir: pembuatan model Regresi Linier statistik yang berpotensi digunakan untuk memprediksi pergerakan harga finansial SP500:

```
#The creation of the Linear Regression Model for predicting price movements in the SP500
#Importing the variables
From sklearn import_cross validation
From sklearn.utils import shuffle
From sklearn import linear_model
From sklearn.netrics import mean_squared_error_Y2_score
#Creating the features and target dataframes
Pricedfnew[‘Close’]
Print(price)
Print(dfnew.columns)
Features = dfnew[[‘NextDatOpen’]]
#Shuffling the data
Price = shuffle (price, random_state=0)
Features = shuffle (features, random_stated=0)
#Dividing the SP financial data into Training Mode and Test Mode
X_train,, X_test, y_train, y_test= cross_validation.train_test_
Split(features, price, test_size=0.2, random_state=0)
#Linear Regression Model on SP500 financial price information
Reg= linear_model.LinearRegression()
X_train.shape
Reg.fit(X_train, y_train)
redDT.fit(X_train, y_train)
y_pred= reg.predict(X_test)
y_pred= regDT.predict(X_test)
print (“Coefficients: ”, reg.coef_)
#Calculating the Mean Squared Error
Print(“mean squared error: ”,mean_squared_error(y_test, y_pred))
#Calculating the Variance Score
Print (“mean squared error: ”,r2_score(y_test, y_prod))
#Calculating the Standard Deviation
Standarddev=price.std()
#Predict the Opening Price of the SP500 and the Opening Volume
#In the predict function, please enter the first parameter for the Opening Price of the SP500 and the
    2nd Volume in US Dollars
SP500ClosePredict=reg.predict ([[269.05]])
#180 is the Standard Deviation of the difference between the Opening Price and the Closing Price
   of the SP500
So this range
Print(“Stock Likely To Open at: ”,SP500ClosePredict, “(+-11)”)
Print(“Stock Open between: ”, SP500ClosePredict+standarddev,” & “
SP500ClosePredict-standarddev)
Name: Close, Length: 5911, dtype: float64
Index([u’Close’, u’NextDayOpen’], dtype=’object’)
(‘Coefficients: ”, array([0.98986882]))
(‘mean squared error: ‘, 313.02619408516466
(‘Variance Score: ‘, 0.994126802384695)
(‘SP500 Stock likely to open at: ‘, array([269.34940985]), ‘(+-11)’)
(‘SP500 Stock Open between: ‘, array([500.67339591]), ‘ & ‘
Array([38.02542379]))
```

Secara keseluruhan, modul-modul kode sumber Python yang terpisah ini, ketika terintegrasi bersama, akan membentuk dasar suatu mode untuk membantu memprediksi pergerakan harga SP500, dan dari sana, membuat keputusan perdagangan yang relevan dan menguntungkan. Sekali lagi, seperti halnya Model Chatbot Portal Diabetes, kode sumber Python di sini hanyalah contoh dasar.

Jutaan baris pemrograman Python lagi akan dibutuhkan untuk menerapkannya ke mode produksi di dunia nyata. Selain itu, model tersebut harus disempurnakan dan dioptimalkan secara real-time agar tetap prima.

BAB 4
JARINGAN SYARAF TIRUAN

Sejauh ini dalam buku ini, dua bab pertama telah memberikan wawasan yang sangat mendalam tentang apa sebenarnya Kecerdasan Buatan (Bab 1), dan bagaimana Pembelajaran Mesin (Bab 2) mulai memberikan dampak yang besar dalam Keamanan Siber saat ini. Di bab terakhir, kita membahas secara mendalam aspek teoretis dan terapan dari Pembelajaran Mesin. Di paruh kedua bab kedua, dua contoh spesifik dieksplorasi lebih lanjut tentang bagaimana Pembelajaran Mesin dapat digunakan, dengan memanfaatkan bahasa pemrograman Python.

Contoh-contoh yang dibahas meliputi pembuatan Portal Pengujian Diabetes untuk klinik rawat jalan (atau bahkan rumah sakit besar), dan pembuatan alat untuk membantu memprediksi harga saham tertentu di S&P 500, salah satu lembaga perdagangan keuangan terbesar di Amerika Serikat, pada hari berikutnya. Namun, ada subkomponen lain dari Kecerdasan Buatan yang juga mendapatkan perhatian dengan sangat cepat, yaitu Jaringan Syaraf Tiruan.

Pada Bab 1, kami memberikan ikhtisar dan definisi teknis tentang apa itu Jaringan Saraf Tiruan, tetapi kami mendedikasikan seluruh bab ini untuk Jaringan Saraf Tiruan. Bab ini akan membahas topik ini dari sudut pandang teoretis dan terapan, sama seperti bab sebelumnya. Singkat cerita, Jaringan Saraf Tiruan adalah bagian dari Kecerdasan Buatan yang mencoba "meniru" proses berpikir dan penalaran otak manusia.

Namun sebelum kita membahasnya lebih dalam, pertama-tama sangat penting untuk memberikan ikhtisar tingkat tinggi.

4.1 JARINGAN SARAF TIRUAN

Neuron

Seperti yang baru saja dijelaskan, mungkin tujuan terbesar Jaringan Saraf Tiruan adalah untuk meniru proses berpikir dan penalaran otak manusia. Penting untuk diingat bahwa ini tidak hanya melibatkan pemeriksaan struktur otak pada tingkat makro, tetapi tujuannya adalah untuk mendalami hingga ke lapisan neuron, yang dianggap sebagai blok pembangun paling dasar dari otak manusia.

Dalam banyak hal, otak manusia dapat dianggap seperti Infrastruktur Jaringan, yang terdiri dari berbagai jenis koneksi jaringan. Dan dalam setiap jalur komunikasi jaringan, paket data selalu menjadi inti dari proses ini. Dengan cara yang sangat mirip dengan otak manusia, paket data inilah yang bertindak sebagai neuron. Layaknya paket data, neuron juga terdiri dari badan pusat, yang dikenal sebagai "nukleus". Faktanya, hal ini sangat mirip dengan header, paket informasi/data, dan trailer yang membentuk keseluruhan paket data.

Di tingkat nukleus inilah semua proses komputasi berlangsung. Namun, penting untuk diingat bahwa bukan hanya satu neuron tunggal yang menghasilkan semua daya ini. Sebaliknya, otak manusia terdiri dari miliaran neuron ini, yang secara harfiah menghasilkan

semua penalaran dan pemikiran logis yang dapat dilakukannya untuk satu manusia. Dengan kata lain, kumpulan miliaran neuron inilah yang membentuk otak manusia.

Ambil contoh, sekali lagi, paket data. Bukan hanya satu paket data yang memungkinkan kita berkomunikasi melalui internet, melainkan kekuatan kolektif dari ratusan atau bahkan ribuan paket data yang memungkinkan kita berinteraksi tidak hanya dengan situs web lain, tetapi juga dengan individu lain, terutama saat kita mengirim email, pesan teks, dan pesan obrolan (misalnya, saat Anda menggunakan chatbot di situs e-commerce).

Jadi, pertanyaan yang tersisa sekarang adalah, bagaimana miliaran neuron ini terhubung satu sama lain, sehingga proses berpikir, logika, dan penalaran kita tampak begitu mulus? Jawabannya berasal dari berbagai pemicu listrik, yang dikirim dari satu neuron ke neuron berikutnya secara berurutan. Dalam istilah yang lebih fisiologis, pemicu listrik ini pada dasarnya adalah proses elektrokimia, yang biasanya terdiri dari pertukaran ion dan transmisi yang terjadi di antara miliaran neuron ini.

Hal ini dicapai dengan melewatkan pemicu listrik ini di sepanjang bidang geometri aksonomis serta melalui difusi molekul neurotransmiter melalui apa yang dikenal sebagai "Celah Sinaptik". Namun, penting untuk diingat bahwa komunikasi yang terjadi antar neuron bukanlah konduksi listrik langsung, melainkan melalui muatan ionik ini, seperti yang baru saja dijelaskan. Jadi, dengan kata lain, pada tingkat yang sangat sederhana, ketika satu neuron berkomunikasi dengan neuron lain, jalur komunikasi pertama-tama berasal dari nukleus neuron.

Dari sana, muatan bergerak keluar ke akson, dan kemudian dari sana ke sambungan sinaptik yang terletak di titik ujung akson. Jalur komunikasi (dari satu neuron ke neuron lain) menuju ke tingkat yang jauh lebih dalam, yang dikenal sebagai "Dendrit", yang juga disebut sebagai "Soma". Komunikasi yang terjadi dari satu neuron ke neuron lain telah tercatat pada kecepatan yang mencengangkan, tiga meter per detik.

Sekarang, mari kita ambil contoh bagaimana satu neuron berkomunikasi dengan neuron lainnya, lalu kalikan dengan faktor 1.000.000.000 kali. Inilah yang membentuk keseluruhan proses berpikir, logika, dan penalaran otak manusia, dan oleh karena itu, disebut sebagai "Jaringan Saraf Biologis".

Terkait hal ini, dalam dua bab sebelumnya, kita telah membahas bagaimana masukan yang berbeda untuk sistem Kecerdasan Buatan memiliki nilai bobot statistik yang berbeda pula. Namun, jika membahas fisiologi dan anatomi otak manusia, semua bobot ini memiliki nilai statistik yang sama yang ditetapkan untuk setiap miliaran neuron yang ada di dalamnya. Namun, inilah masukan yang masuk ke otak manusia, karena secara teknis ini adalah stimulus yang kita lihat di dunia luar, sebagaimana ditangkap oleh mata manusia. Penting untuk dicatat bahwa interkoneksi antar neuron (seperti yang baru saja dijelaskan sebelumnya) tidak memiliki bobot statistik yang sama. Sebaliknya, keduanya memiliki nilai yang berbeda, yang secara teknis dianggap bersifat "Eksitori" atau "Inhibitori". Misalnya, yang pertama akan mempercepat komunikasi yang terjadi dari satu neuron ke neuron berikutnya, sedangkan yang kedua justru dapat memblokir komunikasi tersebut, sesuai namanya. Namun, jelas bahwa bobot statistik yang bervariasi ini tidak dapat ditetapkan secara manual, melainkan ditentukan

oleh varians, atau perbedaan, dalam pemancar kimia serta zat modulasi yang ada di dalam neuron itu sendiri, dan juga dalam akson yang ada di sambungan sinaptik.

Pembobotan spesifik dengan tingkat yang bervariasi seperti yang baru saja dijelaskan inilah yang membentuk dasar bagi apa yang dikenal sebagai "Jaringan Syaraf Tiruan", yang juga dikenal sebagai "ANN". Meskipun kecepatan rata-rata komunikasi antara satu neuron ke neuron berikutnya dianggap tiga meter per detik, kecepatan ini kini dapat bervariasi mengingat efek dari keadaan "Eksitori" dan "Inhibisi" neuron. Perbedaan ini dapat berkisar antara 1,5 meter per detik hingga lima meter per detik.

Dasar-Dasar Jaringan Syaraf Tiruan (ANN)

Meskipun Jaringan Syaraf Tiruan mungkin terdengar seperti jargon teknologi baru di dunia Keamanan Siber, kenyataannya asal-usulnya sebenarnya sudah ada sejak tahun 1940-an, lebih tepatnya, tahun 1943. Selama kurun waktu tersebut, banyak ilmuwan menemukan beberapa fondasi yang berfungsi untuk

Jaringan Syaraf Tiruan, dan pada akhirnya, enam di antaranya masih ada hingga saat ini. Fondasinya adalah sebagai berikut:

1) Aktivitas spesifik Neuron dalam ANN menggunakan pendekatan "semua atau tidak sama sekali". Ini berarti aktivitas tersebut digunakan sepenuhnya untuk membantu memprediksi hasil keluaran, atau tidak digunakan sama sekali.
2) Dalam ANN, jika terdapat sejumlah sinapsis saraf yang memiliki bobot statistik lebih besar dari satu, sinapsis tersebut harus "diaktifkan" dalam jangka waktu yang telah ditentukan (konsep ini telah diulas lebih lanjut di subbagian sebelumnya).
3) Satu-satunya penundaan yang dapat diterima dalam sistem ANN adalah penundaan Sinaptik.
4) Jika Sinaps dianggap bersifat "inhibitor" (hal ini juga telah diulas secara rinci di subbagian sebelumnya), maka satu-satunya tindakan pencegahan yang dapat dilakukan dari dalam ANN adalah menghentikan aksi satu Neuron pada satu waktu dalam sistem.
5) Interkoneksi yang terdapat dalam ANN tidak, dan tidak boleh berubah seiring waktu.
6) Neuron sebenarnya tersusun dari format biner dalam sistem ANN.

Teorema kunci lain yang berkaitan dengan ANN yang masih banyak digunakan hingga saat ini dikenal sebagai "Hukum Pembelajaran Hebbian". Sistem ini juga didirikan pada tahun 1949, dan secara spesifik dinyatakan sebagai berikut: *Ketika akson Sel A cukup dekat untuk merangsang tingkat Sel B, dan ketika Sel A berperan aktif dalam transmisi Sel B, maka terjadi proses pertumbuhan atau perubahan metabolisme seiring dengan peningkatan tingkat Sel A yang meningkatkan tingkat efisiensinya.*

Dengan kata lain, terdapat hubungan matematis langsung satu-satu (1:1) antara Sel A dan Sel B. Semakin aktif Sel B dalam sistem ANN, maka hal tersebut akan berdampak langsung dan positif terhadap beban kerja dan produktivitas Sel A, yang akan meningkatkan keseluruhan proses sistem ANN untuk menghasilkan keluaran yang diinginkan.

Kemudian, pada tahun 1960-an dan 1980-an, dua konstruksi teoretis lainnya juga dirumuskan, yang bahkan diterapkan pada sistem ANN yang digunakan saat ini. Konstruksi-konstruksi tersebut adalah sebagai berikut:

1) Prinsip Memori Asosiatif, juga dikenal sebagai "AM" (1968):
   Prinsip ini menyatakan bahwa jika sebuah Vektor Informasi (yang utamanya terdiri dari kode sumber dan berbagai pola lainnya [seperti pola pada kumpulan data kualitatif]) digunakan dalam sistem ANN, maka hal tersebut juga dapat dianggap sebagai masukan untuk memodifikasi bobot statistik yang telah ditetapkan agar dapat lebih berkorelasi dengan kumpulan data yang telah dikaitkan dengannya.
2) Prinsip Pemenang Mengambil Semua, juga dikenal sebagai "WTA" (1984):
   Konstruk prinsip ini menyatakan bahwa jika terdapat pengelompokan Neuron tertentu (dilambangkan sebagai "N"), dan jika semuanya menerima jenis Vektor Input yang sama, maka hanya satu Neuron yang perlu diaktifkan untuk mengoptimalkan lebih lanjut kemampuan komputasi dan pemrosesan sistem ANN. Neuron ini kemudian akan ditetapkan lebih lanjut sebagai neuron yang bobot input statistiknya paling sesuai dengan sistem ANN sehingga output yang diinginkan dapat tercapai. Dengan kata lain, jika hanya dibutuhkan satu Neuron tertentu untuk menyelesaikan suatu fungsi tertentu, maka tidak ada kebutuhan praktis untuk memiliki beberapa Neuron untuk menjalankan jenis fungsi yang sama dari dalam sistem ANN.

Penting untuk dicatat bahwa kedua teorema di atas sebagaimana baru saja dijelaskan sebenarnya telah terbukti secara ilmiah keberadaannya dalam proses otak manusia, atau "Jaringan Saraf Biologis".

Setelah enam prinsip dan dua teorema di atas dikembangkan, struktur dasar untuk sistem ANN kemudian dirumuskan. Model-model ini juga digunakan dalam sistem ANN saat ini. Model-model tersebut adalah sebagai berikut:

1) Perceptron:
   Model ini telah diulas secara rinci dalam komponen teoretis Bab 2.
2) Artron:
   Model ini juga disebut sebagai "Model Neuron Berbasis Peralihan Statistik", dan dikembangkan pada akhir tahun 1950-an. Artron dianggap sebagai subset dari Neuron, karena hanya digunakan untuk membantu mengotomatiskan proses lebih lanjut dari dalam sistem ANN. Artron tidak memiliki arsitektur berbasis Neuron sendiri.
3) Adaline:
   Model ini juga disebut sebagai "Neuron Linear Adaptif", dan dikembangkan pada awal tahun 1960-an. Ini sebenarnya adalah Neuron berbasis buatan. Perlu dicatat bahwa ini hanya merujuk pada satu Neuron, dan bukan serangkaian Neuron yang membentuk jaringan yang lebih kohesif.
4) Madaline:
   Model ini dikembangkan pada tahun 1988, dan sebenarnya didasarkan pada Adaline, seperti yang baru saja diulas. Namun, Madaline terdiri dari banyak Neuron, bukan hanya satu. Ini juga disebut "Banyak Adaline".

Akhirnya, keempat komponen di atas mengarah pada terciptanya fondasi bagi sistem ANN yang digunakan saat ini. Komponen-komponen tersebut adalah sebagai berikut:

1) Jaringan Backpropagation:
   Ini adalah ANN berlapis ganda, dengan Perceptron sebagai wahana utama yang digunakan untuk menghitung keluaran yang diinginkan dari sistem ANN. Sistem ini menggunakan berbagai "Lapisan Tersembunyi", dan inti matematis untuk sistem ANN jenis ini adalah "Teori Pemrograman Dinamis Richard Bellman".
2) Jaringan Hopfield:
   Ini dikembangkan oleh seorang ilmuwan yang dikenal sebagai John Hopfield pada tahun 1982. Sistem ANN jenis ini juga memiliki banyak lapisan, tetapi yang membedakannya dari Jaringan Backpropagation adalah "umpan balik" dari Neuron yang digunakan dalam sistem ANN juga digunakan untuk menghitung nilai keluaran yang diinginkan. Bobot statistik yang diberikan pada masukan didasarkan pada Prinsip Memori Asosiatif, sebagaimana telah dijelaskan.
3) Jaringan Kontra Propagasi:
   Sistem ANN ini diciptakan pada tahun 1987, dan fondasi matematikanya terletak pada apa yang dikenal sebagai "Kohonen Self-Organizing Mapping", yang juga dikenal sebagai "SOM". Sistem ini memanfaatkan lebih lanjut Prinsip Pemenang Ambil Semua, yang juga telah dijelaskan sebelumnya. Sistem ini juga memanfaatkan apa yang dikenal sebagai "Pembelajaran Tanpa Pengawasan", dan sangat sering digunakan ketika hasil yang cepat dibutuhkan dari keluaran yang dihitung.
4) LAMSTAR:
   Ini adalah akronim yang merupakan singkatan dari "Large Memory Storage and Retrieval Network". Sistem ini juga dikenal sebagai sistem ANN tipe "Hebbian", yang menggunakan berbagai lapisan SOM dan komponen WTM. Untuk menetapkan bobot statistik pada masukan dalam sistem ANN, konsep yang dikenal sebagai "Bobot Tautan Berbasis Kantian" digunakan. Konsep ini terutama digunakan untuk menghubungkan beberapa lapisan Neuron, yang kemudian memungkinkan sistem ANN untuk mengintegrasikan masukan dari berbagai jenis dan dimensi. Fitur unik LAMSTAR adalah ia juga memanfaatkan apa yang dikenal sebagai "Peta Fitur" yang menampilkan aktivitas Neuron yang aktif dari dalam sistem ANN. Sistem ini juga memanfaatkan "Graduated Forgetting". Ini berarti bahwa sistem ANN jenis ini tetap dapat berjalan dengan lancar meskipun terdapat potongan data besar yang hilang dalam masing-masing set data.

4.2 ASPEK TEORETIS JARINGAN SYARAF TIRUAN

Adaline

Sebagaimana telah diulas di subbagian sebelumnya, Adaline (yang sebenarnya merupakan akronim untuk **AD**aptive ***LI***near ***NE***uron) bukan hanya salah satu aspek terpenting dari sistem ANN, tetapi juga merupakan salah satu blok pembangun utama untuk

apa yang dikenal sebagai "Bipolar Perceptron". Secara matematis, adaline dapat direpresentasikan sebagai berikut:

$$Z = Wo + n \sum t = 1 WiXi$$

Di mana:

Wo = Istilah yang bias secara statistik untuk fungsionalitas pelatihan sistem ANN.

Ketika Adaline benar-benar diterapkan pada sistem ANN, keluaran yang diinginkan dapat dihitung sebagai berikut:

$$Z = \sum I WiXi.$$

Pelatihan Adaline

Perlu dicatat bahwa pelatihan spesifik untuk setiap sistem ANN, pada tingkat yang sangat sederhana, hanya melibatkan proses pemberian berbagai bobot statistik untuk semua masukan yang digunakan untuk mendapatkan keluaran yang dibutuhkan. Secara teknis, ini sebenarnya dikenal sebagai "Adaptive Linear Combiner", atau disingkat "ALC". Dengan kata lain, ini hanyalah penjumlahan berbasis linear yang umum di antara semua elemen dalam "Bipolar Perceptrons". Pelatihan semacam ini dapat direpresentasikan secara matematis sebagai berikut:

Diberikan sejumlah "X" set pelatihan di mana X1 ... Xl; d1 ... dL Di mana:

$$Xi = (X1 \ldots Xn)T * I; I = 1,2, \ldots L$$

Di mana:

I = set numerik ke − I;
N = jumlah total masukan;
Di = keluaran yang diinginkan dari Neuron spesifik yang dimaksud.

Hal ini kemudian menghasilkan algoritma pelatihan ANN akhir, yang secara matematis direpresentasikan sebagai berikut:

$$J(w) = E(e\text{^}2*k) = 1/L*L \sum k = 1*c\text{^}2*k$$

Di mana:

E = ekspektasi statistik;
Ek = galat pelatihan statistik;
K = himpunan numerik iteratif yang digunakan oleh sistem ANN.

Penting untuk dicatat bahwa untuk mengoptimalkan bobot statistik yang ditetapkan pada sistem ANN, konsep yang dikenal sebagai "Kuadrat Rata-Rata Terkecil" digunakan. Dari sudut pandang statistik, konsep ini dapat direpresentasikan sebagai berikut:

$$VJ = 0J/0W = 0$$

Lebih lanjut, bobot statistik yang ditetapkan pada masukan sistem ANN dalam skenario spesifik ini juga dapat direpresentasikan secara statistik sebagai berikut:

$$W\text{^}LMS = R\text{^} - 1 * p.$$

Pelatihan Penurunan Tercuram

Teknik statistik lain yang digunakan oleh sistem ANN saat ini disebut "Pelatihan Penurunan Tercuram". Teknik ini mencoba menggunakan bobot statistik yang telah ditetapkan pada masukan dalam satu set data tertentu dan mengaproksimasi, atau memperkirakan, bobot tersebut untuk set data berikutnya yang dimaksud. Prosedur untuk melakukan hal ini adalah sebagai berikut:

$$L > n + 1$$

Di mana:

$N =$ jumlah total masukan yang digunakan oleh sistem ANN.

Dari uraian di atas, sebuah "Prosedur Pencarian Gradien" kemudian dibentuk, yang secara matematis direpresentasikan sebagai berikut:

$$w(m+1) = w(m) + Vw*(m)$$

Di mana:

Vw = perubahan, atau variasi statistik.

Variasi ini dapat dihitung secara matematis sebagai berikut:

$$Vw(m) = uVJw(m)$$

Di mana:

$U =$ parameter laju statistik.

Akhirnya, pelatihan yang digunakan oleh sistem ANN direpresentasikan secara matematis sebagai berikut:

$$VJ = [0J/0W1 / 0J/0Wn]^T.$$

Madaline

Sebagaimana telah dijelaskan sebelumnya dalam bab ini, "Madaline" sebenarnya merupakan perluasan lebih lanjut dari "Adaline", karena terdapat beberapa lapisan dalam infrastrukturnya. Struktur Madaline yang sebenarnya berbeda dari Adaline dalam artian bahwa keluaran yang dihasilkan dari Madaline tidaklah lengkap. Dengan kata lain, hanya keluaran yang lengkap yang dapat dihasilkan oleh sistem ANN. Untuk melatih Madaline, prosedur khusus yang dikenal sebagai "Aturan Madaline II" sangat sering digunakan saat ini.

Teknik ini didasarkan pada teorema statistik yang dikenal sebagai "Prinsip Gangguan Minimum". Teorema ini terdiri dari beberapa fase berbeda, yaitu:

1) Semua bobot statistik yang ditetapkan untuk masukan sistem ANN awalnya memiliki nilai acak yang sangat rendah yang terkait dengannya. Dengan kata lain, set data pelatihan tertentu—seperti di mana $Xi(i = 1,2 \ldots)$—hanya diterapkan secara matematis pada satu vektor pada satu waktu ke masukan sistem ANN yang dimaksud.
2) Sejumlah nilai bipolar statistik yang salah pada lapisan keluaran Madaline dihitung satu per satu, dan juga dicatat oleh Kesalahan "E" pada setiap vektor yang juga bertindak sebagai masukan.
3) Untuk setiap Neuron yang mungkin ada pada lapisan keluaran Madaline, sub-prosedur berikut juga digunakan:
    a. Ambang fungsi aktivasi dilambangkan sebagai "Th." Dengan kata lain, untuk setiap masukan yang ada dalam sistem ANN, Neuron pertama yang belum disetel sebenarnya dipilih, dan juga dilambangkan sebagai "$ABS[z - th]$." Ini berarti bahwa nilai-nilai Neuron ini haruslah absolut dari sudut pandang matematis. Jadi, misalnya, jika terdapat sejumlah "L" masukan berbasis vektor, maka proses seleksi ini dapat direpresentasikan secara matematis sebagai berikut: $n * L$ nilai Z. Ini adalah simpul spesifik yang dapat membalikkan polaritasnya bahkan dengan varians sekecil apa pun, sehingga nama teknisnya adalah "Neuron Jarak Minimum". Ia dipilih dari nilai "$ABS[z - th]$" yang sesuai.
    b. Selanjutnya, bobot statistik setiap Neuron dalam sistem ANN diubah sehingga keluaran bipolar (dilambangkan sebagai "Y") juga berubah dalam format linear yang sama.
    c. Masukan yang berbasis vektor secara matematis dipropagasikan kembali ke keluaran sistem ANN.
    d. Jika terdapat perubahan atau varians dalam bobot statistik yang ditetapkan, maka bobot statistik sebelumnya dikembalikan ke Neuron, dan selanjutnya akan diteruskan ke vektor matematis berikutnya yang terkait dengan gangguan atau varians kecil berikutnya, ke Neuron berikutnya.
    e. Langkah-langkah dengan a – d hingga semua jumlah total kesalahan keluaran berkurang sepenuhnya ke tingkat serendah mungkin.
4) Langkah 3 diulang untuk semua lapisan Neuron yang ada dalam sistem ANN.

5) Jika terdapat Neuron yang ada di Lapisan Keluaran, maka langkah 3 dan 4 diterapkan secara berurutan untuk pasangan Neuron yang keluaran simpul berbasis analognya mendekati nilai "0".
6) Untuk Neuron lain yang ada di Lapisan Keluaran, langkah 3 dan 4 juga diterapkan untuk "Neuron Triplet" yang keluaran simpul berbasis analognya mendekati nilai "0".
7) Setelah langkah terakhir tercapai, vektor matematika berikutnya ditetapkan ke "tingkat ke-L" dalam sistem ANN.
8) Langkah 7 diulang untuk setiap kombinasi vektor matematika berbasis "L" tersebut hingga pelatihan sistem ANN dianggap berada pada tingkat optimal dan memuaskan.

Perlu dicatat pada titik ini bahwa prosedur-prosedur ini (Langkah 1–8) dapat diulang untuk pengurutan Neuron, misalnya bahkan "Neuron Quadruple". Sekali lagi, dalam hal ini, semua bobot statistik yang ditetapkan pada Neuron ditetapkan pada nilai ambang batas yang sangat rendah. Misalnya, nilai-nilai spesifik ini dapat positif atau negatif, dalam rentang -1 hingga $+1$. Untuk tujuan pengujian dan pelatihan yang optimal, jumlah total Lapisan Tersembunyi dalam Neuron harus minimal tiga, dan lebih disukai lagi lebih tinggi.

Berdasarkan deskripsi Madaline yang terperinci ini, metode ini sebenarnya dikenal sebagai "Metode Intuitif Heuristik". Dengan kata lain, nilai keluaran yang dihasilkan oleh sistem ANN tidak diharapkan memenuhi apa yang sebenarnya diinginkan. Sistem ini juga sangat rentan terhadap degradasi jika ada kumpulan data yang tidak dioptimalkan dan dibersihkan—proses ini juga telah diulas di Bab 1. Namun pada akhirnya, Adaline dan Madaline-lah yang telah menciptakan fondasi bagi banyak sistem ANN yang digunakan saat ini.

Contoh Madaline: Pengenalan Karakter

Dalam subbagian ini, kami mengkaji studi kasus aktual yang menggunakan Madaline dalam skenario Pengenalan Karakter. Dalam contoh ini, terdapat tiga karakter berbeda, yaitu 0, C, dan F. Karakter-karakter ini telah dikonversi ke dalam format biner matematika, dalam Bidang Geometris Kartesius berukuran enam kali enam. Dalam contoh khusus ini, Madaline dilatih dan dioptimalkan lebih lanjut dengan berbagai teknik, dan Total Error Rate (TLR) serta Konvergensi statistik juga dicatat dan direkam. Pelatihan Madaline menggunakan prosedur berikut:

1) Sebuah set data pelatihan dibuat dengan masing-masing lima set 0s, Cs, dan Fs.
2) Data ini kemudian dimasukkan ke dalam Madaline.
3) Bobot statistik untuk input Madaline kemudian ditetapkan secara acak ke rentang numerik -1 hingga $+1$.
4) Fungsi transfer batas keras berbasis matematika kemudian diterapkan pada Madaline untuk setiap Neuron di dalamnya, yang direpresentasikan sebagai berikut:

$$\mathrm{Y(n)} = \{1, \mathrm{if\,X} > 0;\ -1, \mathrm{jika\,X} < 0\}.$$

5) Setelah langkah di atas, setiap keluaran yang telah dihitung kemudian diteruskan ke masukan berikutnya di lapisan berikutnya.

6) Keluaran akhir kemudian dibandingkan dengan keluaran yang diinginkan, dan Kesalahan Kumulatif untuk 15 karakter berbeda (seperti yang dijelaskan pada Langkah 1) kemudian dihitung.
7) Jika Kesalahan Kumulatif di atas 15 persen, maka bobot statistik untuk Neuron spesifik yang nilai keluarannya paling mendekati nol kemudian dikoreksi menggunakan rumus matematika berikut:

$$\text{WEIGHTnew} = \text{WEIGHTold} + 2 * \text{constant} * \text{output of the previouslayer} * \text{error.}$$

8) Bobot statistik untuk masukan kemudian diperbarui dan Kesalahan Kumulatif baru kemudian dihitung.
9) Langkah 1–8 diulang hingga tidak ada lagi Kesalahan Kumulatif, atau hingga dianggap sebagai ambang batas yang wajar atau diinginkan.
10) Dataset uji yang dimasukkan ke dalam Madaline terus diperbarui dengan bobot statistik baru (untuk input) dan dari sana, output kemudian dihitung dengan menentukan optimasi keseluruhan Madaline.

Propagasi Balik

Algoritma Propagasi Balik (alias "BP") sebenarnya telah dikembangkan sejak tahun 1986. Tujuan algoritma ini adalah untuk menerapkan bobot statistik dengan berbagai tingkat ke dalam dataset dan menggunakannya untuk melatih Perceptron Multi-Lapisan. Hal ini kemudian mengarah pada pengembangan ANN Multi-Lapisan. Namun, tidak seperti Adaline atau Madaline yang baru saja diulas secara ekstensif di dua subbagian terakhir, lapisan tersembunyi tidak memiliki output yang mudah diakses.

Jadi, premis dasar Algoritma BP adalah menetapkan metodologi komprehensif tertentu yang dapat digunakan untuk mengatur dan mengimplementasikan bobot statistik antara pada masukan yang digunakan oleh sistem ANN, guna melatih Lapisan Tersembunyi yang berada di dalamnya. Algoritma BP secara matematis diturunkan melalui proses berikut:

1) Lapisan Keluaran awal dihitung terlebih dahulu, di mana lapisan antara sistem ANN tidak dapat diakses. Hal ini direpresentasikan secara matematis sebagai berikut;

$$E = V\,½ \sum k * (Dk - Yk)\text{^}2 = ½ \sum k * e\text{^}2k$$

Di mana:
$K = 1 \ldots N$;
N = Jumlah total Neuron yang berada di Lapisan Keluaran.

2) Gradien Tercuram kemudian dihitung sebagai berikut:

$$Wkj\,(m+1) = Wkj(m) + VWkj(m).$$

3) Selanjutnya, "Minimum Arah Turun Bukit" berbasis statistik dihitung secara matematis sebagai berikut:

$$Zk = \sum j\ Wkj\hat{}xj.$$

4) Keluaran Perceptron dihitung sebagai berikut:

$$Yk = Fx(Zk).$$

5) Dengan menggunakan prinsip substitusi, Fungsi Nonlinier untuk sistem ANN didefinisikan secara matematis sebagai berikut:

$$Oe/OWkj = (0e/0zk) * (0zk/0Wkj).$$

6) Terakhir, Lapisan Keluaran akhir sistem ANN direpresentasikan secara matematis sebagai berikut:

$$0z/0Wkj = 0e/0Zk * X2(p) = 0z/0ZrYj(p-1).$$

Algoritma Backpropagation (BP) yang Dimodifikasi

Sesuai dengan judul subbagian ini, tujuannya adalah untuk memasukkan beberapa tingkat risiko atau bias ke dalam Algoritma BP. Idenya adalah untuk membantu membuat set data pelatihan lebih bervariasi, sehingga sistem ANN dapat menghitung keluaran robust yang dianggap dapat diterima. Dengan kata lain, tujuannya adalah untuk menjaga sistem ANN tetap optimal pada tingkat makro dengan memasukkan beberapa varians ke dalamnya, sehingga dapat belajar lebih baik dengan set data mendatang yang dimasukkan ke dalamnya.

Untuk mencapai tugas spesifik ini, tingkat bias dimasukkan ke dalam input, dengan semacam konstanta matematika yang terkait dengannya, seperti +1 atau +B. Tingkat ini dihitung sebagai berikut:

$$Bi = Woi * B$$

Di mana:

Woi = bobot statistik yang diberikan pada masukan Neuron terkait.

Seperti yang telah disebutkan sebelumnya, tingkat varians ini dapat memiliki nilai matematika positif atau negatif.

Namun, untuk memastikan tidak ada terlalu banyak varians yang dapat mendistorsi keluaran sistem ANN secara drastis, dua teknik dapat digunakan: Momentum dan Pemulusan.

Teknik Momentum

Dengan ini, Suku Momentum cukup ditambahkan ke sistem ANN, yaitu sebagai berikut:

$$VWij\ \hat{}(m) = N0\ (r)Yj\ ^{*}(r-1) + aVwij\ \hat{}\ (m-1)$$
$$Wij\hat{}(m+1) = Wij\hat{}(m) + Vwj\hat{}(m).$$

Metode Pemulusan

Ini direpresentasikan secara matematis sebagai berikut:

$$\text{Vwij\^{}(m)} = \text{aVWij\^{}(m} - 1) + (1 - \text{a})0\text{i (r)Yj}^{*}(\text{r} - 1)$$
$$\text{Wij\^{}(m} + 1) = \text{Wij\^{}(m)} + \text{NAWij\^{}(m).}$$

Ada juga teknik lain seperti dua teknik di atas yang baru saja dijelaskan, yaitu sebagai berikut:

1) Meningkatkan rentang matematis Fungsi Sigmoid dari 0 hingga $+1$ menjadi rentang $-0{,}5$ hingga $+0{,}5$.
2) Meningkatkan ukuran langkah sistem ANN lebih lanjut agar tidak "macet" dalam loop pemrosesan, yang dapat menyebabkan "Learning Paralysis".
3) Menggunakan alat konvergensi dan menerapkannya pada "Minima Lokal" sistem ANN. Ini hanya boleh digunakan ketika terdapat probabilitas statistik bahwa pemindahan sistem ANN akan menyebabkan aplikasi mengalami penurunan kinerja selama periode waktu tertentu.
4) Memanfaatkan Algoritma BP yang dimodifikasi atau "disempurnakan". Algoritma ini dapat digunakan untuk mempercepat Konvergensi dan mengurangi varians apa pun. Teknik ini hanya memperhitungkan tanda matematis dari Turunan Parsial untuk menghitung bobot statistik, alih-alih menetapkan Nilai Absolut.

Studi Kasus Backpropagation: Pengenalan Karakter

Kami mengulas kembali Pengenalan Karakter, tetapi kali ini dengan Jaringan Syaraf Tiruan. Dalam contoh khusus ini, model utamanya terdiri dari tiga lapisan berbeda, dengan dua Neuron untuk setiap lapisan. Terdapat juga dua lapisan tersembunyi dengan 36 masukan berbeda yang ditetapkan ke sistem ANN. Fungsi Sigmoid untuk hal ini dapat direpresentasikan sebagai berikut:

$$\text{Y} = 1/1 + \exp(-\text{z}).$$

Representasi matematis di atas juga dapat dianggap sebagai "Fungsi Aktivasi Neuron." Bobot masukan statistik juga telah ditetapkan, dengan beberapa varians yang diizinkan (seperti yang telah diulas sebelumnya), ke sistem ANN. Algoritma BP telah dilatih lebih lanjut untuk mengenali karakter-karakter unik berikut: "A," "B," dan "C." Namun, untuk mengoptimalkan sistem ANN sepenuhnya, karakter-karakter tambahan juga telah diperkenalkan, termasuk: "D," "E," "F," "G," "H," dan "I." Terakhir, untuk memastikan apakah ada kesalahan statistik yang dapat ditangkap, tiga karakter tambahan juga telah ditetapkan, termasuk: "X," "Y," dan "Z."

Algoritma BP digunakan untuk mengeksplorasi lebih lanjut studi ini. Tujuan akhir dari Algoritma BP adalah untuk secara fundamental mengurangi jumlah gangguan, atau kesalahan, yang telah dikaitkan dengan Lapisan Keluaran. Dari sini, serangkaian masukan vektor berbasis matematika telah diterapkan ke sistem ANN melalui Algoritma BP, dan masukan tersebut telah

ditetapkan ke semua nilai masukan. Nilai-nilai ini kemudian dipropagasi maju ke Lapisan Keluaran.

Bobot statistik yang telah ditetapkan juga telah disesuaikan oleh Algoritma BP. Sepanjang siklus hidup pemrosesan sistem ANN, langkah-langkah ini telah digunakan berulang kali, dengan iterasi matematis berikut:

$$(m + 2).$$

Keseluruhan proses berakhir ketika Konvergensi tertentu telah tercapai.

Studi Kasus Backpropagation: Menghitung Suhu Tertinggi dan Terendah Bulanan

Meskipun Jaringan Syaraf Tiruan dan Algoritma BP dapat digunakan hampir di semua jenis industri, Algoritma ini telah menemukan kegunaan khusus di bidang meteorologi. Misalnya, model-model semacam ini dapat membantu menentukan pola cuaca di masa mendatang, terutama terkait tornado, badai petir hebat, hujan deras, siklon, topan, badai, dan bahkan titik-titik panas pemanasan global di planet ini.

Algoritma ini juga dapat digunakan untuk meteorologi pertanian, terutama dalam memprediksi dampak suhu pada tanaman, terutama untuk biji-bijian seperti gandum, jagung, dan kedelai. Algoritma ini juga dapat digunakan untuk memprediksi tingkat kejenuhan atau kekeringan suatu wilayah produksi pertanian di seluruh dunia. Sesuai dengan judul subbagian ini, studi kasus selanjutnya akan mengkaji lebih lanjut bagaimana sistem ANN dengan Algoritma BP dapat digunakan untuk memprediksi suhu rendah dan tinggi setiap hari. Dalam kasus khusus ini, beberapa variabel lain juga dipertimbangkan, yang meliputi:

- Laju penguapan air;
- Kelembapan relatif;
- Kecepatan angin;
- Arah angin;
- Pola presipitasi;
- Jenis presipitasi.

Untuk studi kasus ini, sistem ANN berlapis-lapis telah dibuat dan diimplementasikan dengan Algoritma BP. Dengan Algoritma BP, sistem ini terdiri dari tiga komponen: 1) Lapisan Masukan; 2) Lapisan Tersembunyi; dan 3) Lapisan Keluaran. Perlu dicatat bahwa terdapat Neuron yang terletak di Lapisan Tersembunyi maupun Lapisan Keluaran. Secara kolektif, neuron-neuron tersebut secara matematis merepresentasikan penjumlahan hasil perkalian masukan yang masuk ke sistem ANN, serta bobot statistik terkait.

Algoritma BP telah diformulasikan secara matematis berdasarkan prinsip Metode Kuadrat Terkecil, yang juga dikenal sebagai "LSM". Perlu dicatat bahwa kinerja keseluruhan dan optimasi sistem ANN yang digabungkan dengan Algoritma BP dihitung dengan metodologi Mean Square Error. Hal ini direpresentasikan secara statistik sebagai berikut:

$$F(x) = E(e\text{^}2) = E[(t - a)\text{^}2]$$

Di mana:

F(x) = kinerja sistem keseluruhan;
E = galat statistik yang terdapat di antara keluaran target, atau yang diinginkan, yang dilambangkan dengan "t" dan "a".

Dalam studi kasus khusus ini, Algoritma BP sebenarnya sangat bergantung pada matriks bobot masukan statistik pertama yang telah ditetapkan ke semua lapisan sistem ANN, seperti yang telah dijelaskan sebelumnya. Matriks-matriks ini telah ditetapkan sebelumnya dengan nilai numerik kecil dan rentang yang dilambangkan sebagai "[a, b]". Perlu dicatat bahwa matriks bobot ini dioptimalkan lebih lanjut dengan rumus matematika berikut:

$$W^{*}(k+1) = W(k) + W(k)$$

Di mana:

VW(k) = hasil kali galat statistik yang terdapat pada iterasi tertentu dalam sistem ANN.

Pada titik ini, Algoritma BP kemudian ditransposisikan secara matematis ke dalam wilayah Lapisan Tersembunyi sistem ANN. Ini digunakan untuk menghitung tingkat sensitivitas, atau variasi, matriks bobot yang dioptimalkan untuk setiap Lapisan Tersembunyi yang terdapat dalam sistem ANN. Dalam hal ini, tingkat varians, atau sensitivitas, dilambangkan sebagai "m + 1", dan dihitung secara matematis sebagai berikut:

$$S^{*}(m+1) = -2^{*}F'(n) * e$$

Di mana:

E = galat statistik;
F'(n) = garis diagonal pada Bidang Geometri Kartesius.

Model matematika yang lebih optimal untuk menghitung tingkat varians, atau sensitivitas, diberikan sebagai berikut:

$$Sm = Fm(nm) = W^{*}(m+1)'^{*}S^{*}(m+1)$$

Di mana:

Fm(nm) = turunan matematika sepanjang lapisan "m" pada Bidang Geometri Kartesius.

Namun, untuk memperbarui matriks bobot secara iteratif, rumus matematika berikut digunakan:

$$Wm + (k+1) = Wm(k) - a^{*}Sm^{*}(am-1)'$$

Di mana:

A = laju pembelajaran sistem ANN saat ini.

Sebagai gantinya, data dari berbagai set data yang telah dimasukkan ke dalam sistem ANN akan ditempatkan di Lapisan Keluaran, yang dikaitkan dengan Fungsi Log Sigmoid atau Fungsi Linear Murni.

Secara keseluruhan, dalam model khusus ini, Algoritma BP terdiri dari 252 masukan, yang disusun sebagai berikut, sesuai skema berikut:

- Satu Lapisan Masukan dengan 200 Neuron;
- Tiga Lapisan Tersembunyi yang masing-masing terdiri dari 150, 100, dan 50 Neuron;
- Satu Lapisan Keluaran yang memiliki 12 Neuron untuk secara matematis menghasilkan 12 keluaran target yang berbeda.

Awalnya, kumpulan data yang digunakan oleh sistem ANN harus dioptimalkan. Untuk mencapai tujuan ini, kumpulan data tersebut dikategorikan sebagai Suhu Bulanan Rata-rata Tinggi, atau Suhu Bulanan Rendah. Kumpulan data tersebut juga dikategorikan berdasarkan tahun tahunannya masing-masing, yang merupakan cara keluaran yang dihitung oleh sistem ANN menampilkan hasilnya.

Setelah langkah di atas selesai, data tersebut kemudian dimasukkan ke dalam sistem ANN. Dua jenis Algoritma BP yang berbeda digunakan, yang masing-masing mewakili Suhu Tinggi dan Suhu Rendah. Dari sini, kumpulan data kemudian ditransmisikan ke Lapisan Input dan tiga Lapisan Tersembunyi yang ada dalam sistem ANN, semuanya terkait dengan Fungsi Log Sigmoid. Perlu dicatat bahwa Fungsi Linear Murni dipilih daripada Fungsi Log Sigmoid karena model ini tidak memiliki karakter spesifik yang terdapat dalam kumpulan data. Hanya Fungsi Log Sigmoid yang dapat menangani jenis data kualitatif ini.

4.3 JARINGAN HOPFIELD

Dalam semua konfigurasi sistem ANN yang telah kita bahas sejauh ini dalam buku ini, hanya konsep "Aliran Maju" yang diperkenalkan. Ini berarti bahwa hanya aliran unimodal yang diamati, khususnya yang hanya bergerak dari input ke output. Dalam istilah yang lebih teknis, ini dikenal sebagai "Interkoneksi Nonrekursif". Salah satu keuntungan utama dari hal ini adalah, sampai batas tertentu, dapat menawarkan stabilitas jaringan. Namun, dalam upaya untuk mereplikasi proses berpikir, logika, dan penalaran otak manusia secara lebih akurat, mekanisme yang disebut "Umpan Balik" juga perlu disertakan.

Oleh karena itu, fitur ini juga perlu diintegrasikan ke dalam sistem ANN. Di sinilah peran Jaringan Saraf Hopfield berperan, karena terdiri dari mekanisme "Aliran Maju" dan "Umpan Balik". Namun, kelemahan utamanya adalah stabilitas jaringan dalam sistem ANN tidak dapat dijamin sama sekali. Oleh karena itu, suatu mekanisme perlu diimplementasikan untuk mengatasi efek-efek ini.

Oleh karena itu, penting untuk dicatat bahwa meskipun Jaringan Saraf Hopfield secara tradisional hanya terdiri dari satu Lapisan, mekanisme "Umpan Balik" pada akhirnya justru menjadikannya Jaringan Saraf Berlapis-lapis. Selain itu, Jaringan Saraf Hopfield telah diakui sebagai salah satu yang pertama kali memecahkan apa yang dikenal sebagai "Keputusan Berbasis Non-Konveks".

Dalam Jaringan Saraf Hopfield, mekanisme yang telah dirancang dan diimplementasikan untuk mengatasi efek stabilitas adalah fitur yang tertunda. Dalam arti tertentu, penundaan semacam ini juga terdapat di otak manusia. Hal ini sebenarnya ditunjukkan dalam penundaan waktu di Celah Sinaptik dan aktivasi aktivitas Neuron selanjutnya yang berasal darinya.

Karena pendekatan Multi-Lapisan yang digunakan dalam keluaran Jaringan Saraf Hopfield, jaringan ini juga dapat dianggap bersifat biner. Representasi matematisnya adalah sebagai berikut:

$$\mathrm{Zj} = \sum \mathrm{i} = -\mathrm{WijYz(n)} + \mathrm{Ij};\ \mathrm{n} = 0, 1, 2 \ldots$$
$$\mathrm{Vj(n+1)} = \{1\ \mathrm{V\ Zj} > \mathrm{Thj};\ 0\ \mathrm{V\ Zj} < \mathrm{Thj}$$
$$\mathrm{OR}$$
$$1\ \mathrm{V\ Zj(n)} > \mathrm{Thj}$$
$$\mathrm{Yj(n)\ V\ Zj} = \mathrm{Thj}$$
$$0\ \mathrm{VZj} < \mathrm{Thj}$$

Dengan demikian, dalam hal ini, Jaringan Saraf Tiruan Hopfield Biner dapat dianggap sebagai sistem keadaan "T", yang keluarannya secara teknis termasuk dalam himpunan empat keadaan, yang direpresentasikan sebagai berikut:

$$\{00, 01, 10, 11\}$$

Akibatnya, ketika Jaringan Saraf Tiruan Hopfield memiliki vektor yang dimasukkan ke dalamnya, stabilisasi jaringan akan terjadi pada salah satu dari empat keadaan di atas, dengan keadaan yang tepat pada akhirnya ditentukan oleh bobot statistik yang ditetapkan untuk setiap masukan. Hal ini dijelaskan lebih lanjut di subbagian berikutnya.

Penetapan, atau Pengaturan Bobot dalam Jaringan Saraf Tiruan Hopfield

Jaringan Saraf Tiruan Hopfield menggunakan prinsip-prinsip yang dikenal sebagai "Memori Asosiatif" (alias "AM"), dan "Memori Asosiatif Dua Arah" (alias "BAM"). Secara matematis, keduanya dapat direpresentasikan sebagai berikut:

$$\mathrm{XiER\hat{}m;\ YiEr\hat{}n;\ i} = 1, 2, \ldots \mathrm{L}$$
$$\mathrm{W} = \sum \mathrm{I\ YiXi\hat{}t}$$

Di mana:

W = hubungan bobot antara elemen "x" dan "y" dari vektor masukan.

Persamaan di atas juga dapat dianggap sebagai "Jaringan Asosiatif", yang secara matematis direpresentasikan sebagai berikut:

$$W = L\sum i = 1 XiXi\text{^}Ti \text{ atas sejumlah "X" vektor masukan.}$$

Persamaan di atas juga dikenal sebagai "BAM", seperti yang baru saja dibahas sebelumnya, karena semua nilai Xi berkorelasi erat dengan vektor masukan yang dilambangkan sebagai "W".

Sebelumnya di subbagian terakhir, telah disebutkan bahwa Jaringan Saraf Tiruan Hopfield awalnya merupakan Lapisan Tunggal pada tahap masukan, dan ini dapat direpresentasikan secara matematis sebagai berikut:

$$W = L \sum i = 1 XiXi\text{^}T$$

Di mana:

$$Wij = WJi \, V \, I, j.$$

Namun, untuk sepenuhnya memenuhi tuntutan stabilitas jaringan dengan masukan satu Lapisan dalam Jaringan Saraf Hopfield, persamaan berikut perlu digunakan:

$$Wii = 0 \, V \, i.$$

Namun, jika Jaringan Saraf Hopfield perlu dikonversi agar masukan biner—dilambangkan sebagai "$x(o, 1)$"—dapat menghasilkan nilai matematika dalam rentang numerik -1 hingga $+1$, maka rumus matematika berikut harus digunakan:

$$W = \sum I \, ^{*} \, (2Xi - 1) \, (2Xi - 1)\text{^}T.$$

Menghitung Tingkat Stabilitas Jaringan Spesifik dalam Jaringan Saraf Hopfield

Konsep pengenalan Stabilitas Jaringan telah diperkenalkan secara rinci di subbagian sebelumnya. Di subbagian ini, kami akan membahasnya lebih detail, terutama cara menghitungnya untuk sistem ANN yang menggunakan Jaringan Saraf Hopfield. Penelitian sebelumnya telah menunjukkan bahwa Stabilitas Jaringan dapat dijamin pada tingkat yang lebih tinggi jika matriks "W" dari bobot masukan statistik bersifat simetris geometris, dan jika garis diagonal yang melintasinya sedekat mungkin dengan "0". Hal ini direpresentasikan secara matematis sebagai berikut:

$$Wij = Wji \, Vi, j$$

Di mana:

$$Wu = V \, i.$$

Teori fundamental untuk kedua persamaan di atas berasal dari apa yang dikenal sebagai "Teorema Stabilitas Lyapunov", yang menyatakan bahwa jika Stabilitas Jaringan digunakan dalam fungsi energi matematis dalam sistem ANN, dan jika dapat disempurnakan lebih lanjut sehingga akan menurun seiring waktu, Stabilitas Jaringan dapat dianggap sebagai model prima untuk sistem ANN yang dimaksud.

Namun, agar hal ini terjadi, kondisi berikut harus dipenuhi terlebih dahulu:

- Kondisi 1:

  Setiap perubahan terbatas yang terjadi dalam Sistem Jaringan yang dilambangkan sebagai "Y" akan menghasilkan peningkatan terbatas dalam "E", dengan laju korelasi positif.
- Kondisi 2:

  "E" dibatasi oleh persamaan matematika di bawah ini:

$$E = \sum I\ THjYj - \sum j\ I2yj - \frac{1}{2} \sum I \sum j =/1\ WijYjYi$$

Di mana:

I = Neuron "i"; J = Neuron "j";
Ij = masukan eksternal ke Neuron "J";
THj = ambang batas statistik untuk Neuron "J."

Sekarang, bagaimana "Teorema Stabilitas Lyapunov" dapat membuktikan Stabilitas Jaringan tertentu dari sistem ANN adalah sebagai berikut:

Pada langkah pertama, nilai "W" dibuktikan simetris secara geometris dengan semua elemen diagonal pada Bidang Geometris Kartesius berada pada nilai "0," seperti yang dijelaskan sebelumnya. Keduanya dicapai dengan dua persamaan matematika berikut:

$$W = W"t$$
$$Wii = o\ Vi$$

Di mana:

Nilai Mutlak [Wij] dibatasi untuk himpunan numerik yang ada dalam himpunan "I, J."

Pada langkah kedua, nilai "E" secara matematis memenuhi kondisi "A" dengan mempertimbangkan perubahan, atau varians, yang terjadi hanya di satu wilayah Lapisan Keluaran, yang secara matematis direpresentasikan sebagai berikut:

$$Yk(n+1).$$

Varians selanjutnya dihitung sebagai berikut:

$$Ven = E(n+1) - E(n)$$
$$= [Yk(n) - Yk(n+1)] * [\sum i =/k\ WikYi(n) + Ik - Thx]$$

Namun, dengan asumsi bahwa Jaringan Saraf Tiruan Hopfield berbasis biner digunakan, tiga kondisi statistik berikut juga harus dipenuhi:

$$\mathrm{Yk(n+1)} = \{1 \ldots \mathrm{VZk(n)} > \mathrm{Thk};\ \mathrm{Yk(n)} \ldots \mathrm{VZk(n)} = \mathrm{Thk};\ 0 \ldots \mathrm{VZk(n)} < \mathrm{Thk}$$

Di mana:

$$\mathrm{Zk} = \sum \mathrm{WikYi} + \mathrm{Ik}.$$

Akhirnya, hanya ada dua jenis varians. dapat terjadi dalam sistem ANN, yang secara statistik direpresentasikan sebagai berikut:

$$\mathrm{If\ Yk(n)} = 1, \mathrm{then\ Yk(n+1)} = 0;$$
$$\mathrm{If\ Yk(n)} = 0, \mathrm{then\ Yk(n+1)} = 1.$$

Bagaimana Jaringan Saraf Hopfield Dapat Diimplementasikan

Pada subbagian ini, kami akan memberikan ringkasan tentang bagaimana Jaringan Saraf Hopfield dapat diterapkan ke dalam sistem ANN.

Secara keseluruhan, bobot statistik dari masukan yang ditetapkan harus memenuhi rumus matematika berikut:

$$\mathrm{W} = \mathrm{W}\sum \mathrm{i} = 1\ ^{*}\ (2\mathrm{Xi} - 1)\ ^{*}\ (2\mathrm{Xi} - \mathrm{I})\text{^T}.$$

Sekarang, komputasi Jaringan Saraf Tiruan Hopfield dapat diselesaikan, dengan asumsi komponen "BAM" berada di dalamnya, dengan menggunakan metodologi berikut:

1) Bobot statistik Wij ditetapkan ke matriks matematika yang dilambangkan sebagai "W", di mana $\mathrm{Wii} = \mathrm{o\ Vi}$ dan Xi adalah vektor pelatihan aktual yang sedang digunakan.
2) Pola masukan berbobot yang tidak diketahui, dilambangkan sebagai "X", ditetapkan sebagai:

$$\mathrm{Yi(0)} = \mathrm{Xi}$$

Di mana:

$\mathrm{Xi} =$ elemen "i" dari vektor matematika "X".

3) Langkah #2 dapat direpresentasikan secara statistik sebagai berikut:

$$\mathrm{Yi(n+1)} = \mathrm{Fn[Zi(n)]}$$

Di mana:

Fn = Fungsi Aktivasi yang direpresentasikan sebagai berikut:

$Fn(z) = \{1 \ldots Vz > Th;\ Tidak\ Berubah \ldots Vz = Th;\ -1 \ldots Vz < Th\}$

$Zi(n) = \sum i = 1\ WijYi(n)$

Di mana:

N = rentang bilangan bulat numerik yang mungkin ditemukan dalam iterasi yang dilambangkan sebagai ($n = 0, 1, 2 \ldots$).

Catatan: iterasi di atas terus berulang hingga mencapai Konvergensi tertentu, di mana perubahan $Yi(n+1)$ dapat berkorelasi erat dengan $Yi(n)$ di bawah nilai ambang batas yang telah ditetapkan sebelumnya.

Langkah 1–3 diulang untuk semua elemen dari setiap vektor matematika yang tidak diketahui

4) Hal ini dilakukan hingga elemen berikutnya dari vektor matematika yang tidak diketahui mencapai 100 persen dalam sistem ANN.
5) Namun setelah semua ini, jika ada vektor matematika lain yang tidak diketahui kemudian ditemukan, maka seluruh proses ini, yang mencakup Langkah 1–4, akan diulang lagi.

Model Hopfield Kontinu

Perlu dicatat bahwa semua konsep yang terkait dengan Jaringan Saraf Tiruan Hopfield bersifat diskrit. Namun, konsep-konsep tersebut juga dapat diubah menjadi keadaan kontinu dengan menggunakan model matematika berikut:

$$Yi = f1(AZi) = 1/2 * [1 + \tanh(AZi)].$$

Dalam model di atas, persamaan diferensial dapat digunakan untuk menunda waktu yang terjadi antara Lapisan Masukan dan Lapisan Keluaran sistem ANN. Hal ini dapat dilakukan dengan persamaan matematika berikut:

$$\sum j = /I\ Tij - Zi/Ri + Ii = 0$$
$$C * Dzi/Dt) = \sum j = /1 * (TijYj) - (Zi/Ri) + (Li)$$

Di mana:

$$Yi = Fn(Zi).$$

Studi Kasus Menggunakan Jaringan Saraf Tiruan Hopfield: Deteksi Sel Molekuler

Dalam dunia ilmu biologi, konsep yang dikenal sebagai "Injeksi Mikro Intraseluler" merupakan prosedur yang sangat umum digunakan untuk memanipulasi berbagai jenis kultur sel. Dalam hal ini, segala jenis proses "Mikromanipulasi" untuk satu struktur selulit sangat penting dalam bidang Toksikologi In-Vitro, Kanker, serta penelitian berbasis HIV. Namun, untuk benar-benar menstimulasi sel, salah satu kendala terpenting yang harus diatasi adalah menentukan bentuk geometris sel yang akurat.

Dalam hal Ekstraksi Kontur, sejumlah bidang lain telah diteliti secara mendalam, seperti Pemrosesan Citra. Penentuan struktur tepi sel telah memanfaatkan teknik-teknik seperti detektor berbasis Gradien, salah satunya dikenal secara khusus sebagai konsep "Prewitt, Sobel, dan Laplace". Teknik struktur tepi lainnya juga telah diusulkan, seperti Detektor Penyeberangan Nol Derivatif ke-2 berbasis matematika atau bahkan beberapa metode komputasi lainnya, seperti "Kriteria Canny".

Namun, mengingat kendala lain, seperti tekstur sel, derau sel, keburaman gambar, iluminasi pemandangan, dan lain-lain., teknik-teknik yang baru saja dijelaskan ini tidak dapat menghasilkan hasil dengan tingkat keyakinan statistik yang tinggi. Selain itu, citra sumber sel yang dimaksud dapat direpresentasikan sebagai fragmen tepi yang rusak yang mungkin tidak dapat dideteksi sama sekali. Bahkan data yang ditemukan oleh tepi sel dapat dimiringkan oleh piksel yang diekstraksi dari citra sel yang telah ditangkap.

Selain itu, semua teknik yang baru saja dijelaskan ini biasanya juga memerlukan semacam optimasi "pasca-pemrosesan". Dengan kata lain, kontur aktif sel perlu ditangkap, dan sebagai hasilnya, teknik baru yang dikenal sebagai "Ular: Model Kontur Aktif" diusulkan pada tahun 1988, dan faktanya, teknik ini telah digunakan secara luas. Beberapa hal yang dapat dilakukannya antara lain:

- Deteksi tepi sel;
- Pemodelan bentuk sel;
- Segmentasi sel;
- Pengenalan pola/Pelacakan objek sel.

Teknik "Snake" dengan demikian mampu menghasilkan gambar membran sel yang tertutup dan aktif, dan bahkan dapat disegmentasi dan dibagi lebih lanjut untuk pemeriksaan yang lebih teliti. Pada titik ini, teknik "Snake" juga dapat diintegrasikan ke dalam Jaringan Saraf Hopfield, dan representasi matematis dari penggabungan ini dapat direpresentasikan sebagai berikut:

$$\text{Esnake} = |\text{S2}\,[\text{AEcount(v)} + \text{BEcurv(v)} + \text{TEimage}] * \text{ds}$$

Di mana:

A, B, T = pengaruh relatif suku energi dalam sistem ANN;
Ecurve = suku kehalusan statistik;
Econt = suku kontinuitas statistik;
Eimage = tingkat energi yang berkaitan dengan gaya eksternal untuk menarik sifat – sifat konsep "Ular" ke kontur citra membran sel yang dibutuhkan.

Seperti yang dapat dilihat, Komponen Energi merupakan komponen yang sangat penting dalam contoh spesifik Jaringan Saraf Hopfield ini, dan hal ini dapat direpresentasikan secara matematis sebagai berikut:

$$\text{Esnake} = \text{N}\sum\text{i} = 1\,\{\text{A}[(\text{Xi} - \text{Xi} - 1)\text{^}2 + (\text{Yi} - \text{Yi} - 1)\text{^}2 + \text{B}[(\text{Xi} - 1 - 2\text{Xi} + \text{Xi} + 1)\text{^}2 + (\text{Yi} - 1 - 2\text{Yi} + \text{Yi} + 1)\text{^}2 - \text{Tgi}\}$$

Di mana:

N = jumlah total simpul yang berada dalam "Ular";

Gi = nilai gradien citra pada titik Xi, Yi.

Dalam studi kasus ini, Jaringan Saraf Tiruan Biner Hopfield dua dimensi (2D) digunakan, dan dari sana, Neuron diperbarui pada interval waktu yang telah ditentukan menggunakan rumus matematika berikut:

$$Wik = N\sum i = 1\ M\sum j = 1\ Tikjt\text{^}Vjt + Iik\ Vi = g(Uih)$$
$$G(Utk) = \{1, \text{if } Utk = \max(Uth;\ h = 1,2\ \ldots, M\ 0, \text{otherwise})$$

Di mana:

N = jumlah total simpul "Ular";

M = jumlah total titik tetangga yang harus dipertimbangkan untuk setiap simpul yang berada dalam setiap Neuron yang digunakan oleh sistem ANN.

Metode "Ular" dapat digunakan untuk mengurangi tingkat Energi, dan ini dapat dihitung sebagai berikut:

$$E = -1/2\ N\sum i = 1\ M\sum k = 1\ N\sum j = 1\ M\sum l = 1 * Tikjt\ \text{^}\ Vjk\ \text{^}\ Vjt - N\sum i = 1\ M\sum k = 1\ IikVih.$$

Hal di atas kemudian dapat dipetakan ke Jaringan Saraf Hopfield sebagai berikut:

$$Tikjt - [(4A + 12B)\ \text{^}\ 0ij - (2A + 8B)\ ^{*}\ Oi + 1j - (2A + 8B)\ ^{*}\ 2Bbi + 2j + 2Bb - 1)\ ^{*}\ [XikKjt + YikYjt]]$$
$$Iik = TGik.$$

Perlu dicatat bahwa dalam model ini, koneksi umpan balik dapat menjadi sangat tidak stabil (seperti yang dibahas di subbagian sebelumnya), dan untuk meminimalkan risiko ini, setiap Keluaran Neuron yang dapat berkontribusi pada minimalisasi energi total sistem ANN diterima. Akhirnya, sistem ANN terdiri dari hal-hal berikut:

- 16 Node (dilambangkan sebagai "N=16");
- Garis radial 50 titik pada Bidang Geometri Kartesius (dilambangkan sebagai "M = 50");
- Jumlah total Neuron dalam sistem ANN adalah 800 (dilambangkan sebagai "N X M").

4.4 PROPAGASI TANDINGAN

Jaringan Syaraf Tiruan Propagasi Tandingan (CP) pertama kali diteliti dan ditemukan pada tahun 1987. Dibandingkan dengan jaringan Backpropagation sebagaimana diulas di bagian sebelumnya, jaringan ini sangat cepat, bahkan lebih dari 100 kali lebih cepat. Namun, kelemahannya adalah tidak dapat digunakan untuk berbagai aplikasi; hanya dapat digunakan

untuk sejumlah aplikasi tertentu. Alasan utamanya adalah kecepatan yang lebih tinggi tentu saja membutuhkan daya pemrosesan yang jauh lebih besar pada sebagian sistem ANN.

CP sebenarnya merupakan kombinasi dari jaringan "Self-Organizing" dan "Outstar". Salah satu keuntungan utama menggunakan Jaringan Saraf CP adalah cukup berguna untuk tujuan generalisasi, dalam arti sangat baik dalam mencoba memprediksi seperti apa keluaran dari sistem ANN. Dalam hal ini, CP sangat baik untuk vektor masukan matematika yang dianggap sebagian lengkap atau bahkan sebagian salah secara alami.

Dua konsep utama yang mendasari Jaringan Saraf CP adalah Kohonen Self-Organizing Map Layer (juga dikenal sebagai "SOM"), dan Grossberg Layer, yang akan dibahas lebih rinci dalam dua subbagian berikutnya.

Lapisan Peta Pengorganisasian Mandiri Kohonen

Lapisan ini juga dikenal sebagai lapisan "Pemenang Ambil Semua" dari sistem ANN. Dengan kata lain, hanya untuk satu vektor masukan matematis, keluarannya hanya "1", sementara semua vektor lainnya dianggap bernilai "0". Lebih lanjut, penting untuk dicatat bahwa tidak ada vektor pelatihan lain yang diperlukan untuk Kohonen SOM. Keluaran untuk lapisan ini direpresentasikan oleh rumus matematika berikut:

$$\mathrm{Kj} = \mathrm{m}\sum \mathrm{i} = 1\mathrm{WijXi} = \mathrm{W^{T}iX};\ \mathrm{WjV}[\mathrm{W1j} \ldots \mathrm{Wmj})^{\mathrm{T}}$$

$$\mathrm{XV}\ [\mathrm{X1}\ \ldots\ \mathrm{Xm}]\text{^}\mathrm{T}$$

Di mana:

$\mathrm{J} = \mathrm{q}, 2, \ldots\ \mathrm{p}, \mathrm{p};$

$\mathrm{M} =$ dimensi statistik dari vektor masukan.

Untuk menentukan secara lengkap seperti apa Neuron berikutnya setelah yang pertama (dilambangkan sebagai "$\mathrm{j} = \mathrm{h}$"), hal ini dapat direpresentasikan secara matematis sebagai berikut:

$$\mathrm{Kh} > \mathrm{Kj} = /\mathrm{h}.$$

Namun, jika Neuron didefinisikan sebagai iterasi spesifik, maka persamaan berikut digunakan untuk menghitung deret spesifik ini:

$$\mathrm{Kh} = \mathrm{m}\sum \mathrm{i} = 1\ \mathrm{WihXi} = 1 = \mathrm{W\text{^}ThX}.$$

Lapisan Grossberg

Lapisan ini sebenarnya dianggap sebagai lapisan keluaran terbobot secara statistik dari Lapisan SOM. Namun, pada Lapisan spesifik ini, jumlah total Neuron harus setidaknya setengah dari nilai kelas-kelas berbeda yang digunakan oleh sistem ANN, dan selanjutnya, representasi ini harus bersifat biner. Hal ini dapat dicapai dengan rumus matematika berikut:

$$\mathrm{Gq} = \sum \mathrm{IKiViq} = \mathrm{K\text{^}TVq};\ \mathrm{k\ V}\ [\mathrm{k1}\ \ldots\ \mathrm{kp}]\text{^}\mathrm{T}$$

$$\text{Vq V [V1q ldots Vpq]\^{}T}$$

Di mana:
$Q = 1\,1, 2, \ldots, r, r.$ Ini adalah representasi biner yang sebenarnya, seperti yang telah disebutkan sebelumnya.

Seperti yang telah disinggung sebelumnya, Lapisan SOM menggunakan pendekatan "Pemenang Ambil Semua". Hal ini direpresentasikan secara matematis sebagai berikut:

$\{Kh = 1;\ ki = /0]$ Jika salah satu dari kedua kondisi ini terpenuhi, maka "Winner Take All" dapat dihitung secara matematis sebagai berikut:

$$Gq = p\sum I = 1KijVjq = khUhq = Vhq.$$

Bagaimana Lapisan Input Kohonen Diproses Awal

Langkah-langkah berikut diperlukan untuk menyelesaikan proses ini:
Normalisasi statistik Input Lapisan Kohonen dihitung sebagai berikut:

$$\text{X\^{}ri} = \text{Xi/SQUAREROOT } \sum \text{jX\^{}2j.}$$

Sekarang, pelatihan Lapisan Kohonen terjadi dalam proses berikut:

1) Normalisasi vektor input "X" dilakukan untuk mendapatkan vektor input X';
2) Nilai Neuron pada level Lapisan Kohonen dihitung sebagai berikut:

$$\text{(X')\^{}T} * \text{Wh} = \text{K'h.}$$

3) Akhirnya, semua bobot statistik vektor input pada Lapisan Kohonen dihitung sebagai berikut:

$$\text{K'h} = \sum \text{I X\^{}iWih} = \text{X'iWih} = \text{X\^{}ji} + \text{X\^{}i2} + \ldots \text{X'm} * \text{Whm} = \text{(X')\^{}T} * \text{Wh.}$$

Bagaimana Bobot Statistik Diinisialisasi di Lapisan Kohonen

Setelah fase Prapemrosesan selesai sebagaimana dijelaskan di subbagian sebelumnya, proses inisialisasi, sesuai dengan judul subbagian ini, dihitung secara matematis di bawah ini. Semua bobot statistik ditetapkan ke nilai yang sama, dihitung sebagai berikut:

$$\text{N }^{*}(1/\text{SQUAREROOTN})^{2} = 1.$$

Untuk menambahkan varians tertentu (seperti yang dibahas sebelumnya dalam bab ini), rumus matematika berikut digunakan:

$$\text{X\^{} }^{*}\text{ I} = \text{TXi} + (1 - \text{T}) \ ^{*}\ (1/\text{SQUAREROOT N}).$$

Namun, terdapat juga metode untuk menambahkan derau tambahan, yaitu sebagai berikut:

1) Menambahkan lebih banyak derau ke Vektor Input;
2) Memanfaatkan Bobot Ternormalisasi Acak Statistik;
3) Pemilihan representasi terbaik dari Vektor Input, dan menggunakannya sebagai bobot awal. Hasil akhirnya adalah setiap Neuron akan diinisialisasi satu vektor matematika pada satu waktu.

Lapisan Mode Interpolatif

Perlu dicatat bahwa lapisan Kohonen pada umumnya hanya akan menahan apa yang disebut "Neuron Pemenang". Namun, Lapisan Mode Interpolatif akan menahan sekelompok Neuron berbasis Kohonen tertentu dalam kelas vektor masukan tertentu. Dalam hal ini, keluaran sistem ANN akan dinormalisasi secara statistik ke bobot yang telah ditetapkan sebelumnya; semua keluaran lainnya akan kembali ke nol.

Pelatihan Lapisan Grossberg

Keluaran Lapisan Grossberg dihitung secara matematis sebagai berikut:

$$\mathrm{Gi} = \sum\mathrm{j}\ \mathrm{VijKj} = \mathrm{VshKh} = \mathrm{Vih}$$
$$\mathrm{Gi} = \sum\mathrm{j}\ \mathrm{VijKj} = \mathrm{VshKh} = \mathrm{Vih}.$$

Penyesuaian bobot statistik lebih lanjut dilakukan sebagai berikut:

$$\mathrm{Vij(n+1)} = \mathrm{Vij(n)} + \mathrm{B[Ti - Vij(n)kj]}$$

Di mana:

Ti = keluaran yang diinginkan dari sistem ANN;

$\mathrm{N+1}$ = Neuron yang ditetapkan bernilai "1";

Vij = vektor masukan acak yang ditetapkan bernilai "1" untuk setiap Neuron dalam sistem ANN.

Jaringan Propagasi Kontra Gabungan

Telah dikaji bahwa Lapisan Grossberg dapat digunakan untuk melatih berbagai keluaran sistem ANN agar "berkonvergensi" satu sama lain, sedangkan Lapisan Kohonen pada dasarnya dikenal sebagai "Pra-Klasifikasi" dalam artian juga memperhitungkan apa yang dikenal sebagai "Masukan Tidak Sempurna". Dengan kata lain, yang terakhir tetap tidak tersupervisi, sementara yang pertama tetap dalam status tersupervisi dari dalam sistem ANN.

Selain itu, Neuron yang terletak di dalam Lapisan Grossberg akan secara harfiah bertemu dengan input target yang sesuai, dan ini akan diterapkan secara bersamaan ke Lapisan Kohonen. Faktanya, inilah asal mula istilah "Propagasi Tandingan". Hal ini terutama disebabkan oleh penerapan input target yang diterapkan ke Lapisan Kohonen secara bersamaan.

Namun, salah satu kelemahan utama Propagasi Tandingan adalah ia mengharuskan semua pola input memiliki dimensi yang sama pada Bidang Geometris Kartesius. Oleh karena itu, Propagasi Tandingan tidak dapat digunakan untuk aplikasi yang lebih luas, baik pada tingkat makro maupun umum.

Studi Kasus Propagasi Tandingan: Pengenalan Karakter

Dalam studi kasus ini, tujuan utamanya adalah mengenali tiga nilai numerik berikut: "0," "1," "2," dan "4." Sesuai dengan judul subbagian ini, subbagian ini juga menggunakan teknik Propagasi Tandingan. Dalam hal pelatihan, digunakan dataset berdimensi delapan kali delapan, dengan Kesalahan Bit dalam rentang nilai $1, 5, 10, 20, 30,$ dan 40.

Dalam hal pengaturan bobot statistik, prosedur berikut ditetapkan:

1) Dapatkan semua vektor dataset pelatihan yang relevan yang berada dalam permutasi matematika ini: $\mathrm{Xi, I} = 1, 2, \ldots \mathrm{L}$
2) Untuk setiap vektor relevan yang termasuk dalam permutasi yang ditetapkan pada langkah terakhir, sub-prosedur berikut juga digunakan:
   - Normalisasikan setiap $\mathrm{Xi, I} = 1, 2, \ldots \mathrm{L}$ dengan permutasi matematika berikut: Xi^t/SQUAREROOT ($\sum$X^2j);
   - Hitung vektor rata-rata sebagai $\mathrm{X} =$ ($\sum$Xj^1)/N;
   - Normalisasikan vektor rata-rata sehingga $\mathrm{X, X'} =$ X/SQUAREROOT ($\sum$X^2j);
   - Tetapkan bobot Neuron Kohonen menjadi $\mathrm{Wk} =$ X;
   - Atur bobot Neuron Grossberg ke (W1kW1k … W1k) sehingga sepenuhnya disesuaikan dengan vektor keluaran yang dilambangkan sebagai "Y".
3) Langkah 1–2 terus berulang dalam proses iteratif hingga semua set data pelatihan disebarkan secara keseluruhan ke seluruh sistem ANN.

Terakhir, set data uji dihasilkan melalui prosedur acak, dengan rumus berikut:

$$\text{testingData} = \text{getCPTTesting (trainingData, numberOfBitError, numberPerTrainingSet)}$$

Di mana:

numberOfBitError = jumlah Kesalahan Bit yang diharapkan;

numberPerTrainingSet: digunakan untuk menentukan ukuran set data pengujian yang diharapkan;

testingData: digunakan untuk mendapatkan parameter keluaran lainnya, serta set data uji.

4.5 TEORI RESONANSI ADAPTIF

Teori Resonansi Adaptif dikembangkan pada tahun 1987, dan dikenal sebagai "ART" singkatnya. Tujuan utama teori ini adalah untuk menciptakan, mengembangkan, dan menerapkan sistem ANN terkait Pengenalan Pola atau Perilaku Klasifikasi yang sangat mirip dengan Jaringan Syaraf Biologis (BNN). Dengan kata lain, tujuan utama ART adalah mengembangkan sistem ANN dengan apa yang dikenal sebagai "Plastisitas". Setiap kali sistem ANN mempelajari pola baru, ia tidak akan menggunakannya untuk menggantikan pola lain

yang telah dipelajari sebelumnya. Intinya, sistem ANN menjadi repositori pusat dari semua yang telah dipelajarinya dan akan terus dipelajari di masa mendatang.

Jaringan ART terdiri dari komponen-komponen berikut:

- Lapisan Perbandingan;
- Lapisan Pengenalan;
- Elemen Penguatan yang menyalurkan outputnya ke "g1";
- Elemen Penguatan yang menyalurkan outputnya ke "g2";
- Elemen Reset (di sinilah Lapisan Perbandingan dievaluasi dan dibandingkan dengan "Nilai Kewaspadaan", yang tidak lain adalah tingkat toleransi yang dirancang khusus untuk sistem ANN.

Masing-masing komponen di atas akan diulas lebih detail di beberapa subbagian berikutnya.

Lapisan Perbandingan

Pada lapisan spesifik ini, sebuah Elemen Biner dimasukkan ke dalam Neuron pada Lapisan Perbandingan, dengan permutasi matematis berikut:

$$(j = 1 \ldots m;\ m = \dim(X)).$$

Bobot statistik juga diberikan kepada Neuron ini dengan rumus statistik berikut:

$$Pj = m\sum I = 1\ TijTi$$

Di mana:

Ri = iterasi "i" dari vektor keluaran berdimensi "m" dari "r";
n = jumlah total Kategori yang perlu dikenali dalam sistem ANN.

Perlu dicatat juga bahwa semua Neuron pada Lapisan Perbandingan akan menerima Keluaran Skalar matematis yang sama, dilambangkan sebagai "Gi," berdasarkan permutasi berikut:

$$Cj(0) = Xj(0).$$

Lapisan Pengenalan

Ini sebenarnya berfungsi sebagai varian lain dari "Lapisan Klasifikasi." Berbagai masukan yang diterimanya secara matematis diturunkan dari vektor bobot dimensi "n" "d." Hal ini dihitung secara matematis sebagai berikut:

$$Dj = m\sum I = 1\ BjiCi = bj\text{^}T\ C;\ Bj\ V\ [Bj1 \ldots Bjm]$$

Di mana:

I = 1, 2, … m;
J = 1, 2, … n;

$M = \dim(x);$
$N = \text{jumlah Kategori.}$

Pada Lapisan Pengenalan, terdapat properti yang dikenal sebagai "Koneksi Inhibisi Lateral". Di sinilah keluaran setiap Neuron (dilambangkan sebagai "I") terhubung melalui matriks berbobot koneksi "inhibisi", yang dilambangkan sebagai berikut:

$$L = \{Lij\}, I = /j$$

Di mana:

$Lij < 0$ untuk setiap Neuron lain dalam sistem ANN (dilambangkan sebagai "j"). Hasil akhirnya adalah bahwa Neuron dengan keluaran matematis yang besar akan menggantikan semua Neuron lain dengan nilai ambang batas matematis yang lebih rendah.

Konsep kunci lain yang perlu diperhatikan adalah "Penguatan Positif". Di sinilah loop umpan balik positif dalam sistem ANN (dilambangkan sebagai "$Ijj > 0$") digunakan sedemikian rupa sehingga setiap keluaran matematis Neuron (dilambangkan sebagai "Rj") secara harfiah diumpan balik dengan bobot statistik bernilai positif untuk memperkuat keluaran lebih lanjut (seperti yang baru saja dijelaskan) jika Neuron tersebut akan mengaktifkan Neuron lain secara berurutan dan berulang.

Elemen Penguatan dan Penyetelan Ulang

Elemen-elemen ini menggunakan jenis Keluaran Skalar yang sama dengan semua Neuron dalam sistem ANN. Hal ini direpresentasikan secara statistik sebagai berikut:

$$G2 = OR(x) = OR(x1 \ldots Xn)$$
$$G2 = OR(\text{atau}) \text{ U } OR(x)$$
$$= OR(r1 \ldots rN) \text{ U } OR(x1 \ldots Xn)$$
$$= g2 \text{ U } OR(r).$$

Dengan kata lain, jika terdapat setidaknya satu elemen masukan "X" yang nilainya sama dengan 1, maka $g2 = 1$. Atau, jika terdapat elemen lain dari $g2 = 1$, tetapi tidak ada elemen dari "r" maka $g1 = 1$, atau $g1 = 0$. Batang di atas adalah faktor negasi berbasis statistik, sedangkan "U" juga merepresentasikan interseksi logis dan statistik. Demikian pula, jika $OR(x)$ maka $OR(r)$ juga akan selalu sama dengan nol.

Selain itu, "Elemen Reset" akan mengevaluasi secara cermat tingkat korelasi yang ada antara masukan vektor "X" dan keluaran vektor "C", dengan permutasi berikut:

$$N < N0$$

Di mana:

$N0$ = nilai toleransi awal yang telah ditetapkan sebelumnya, secara teknis juga dikenal sebagai "Nilai Kewaspadaan".

Pembentukan Jaringan Saraf Tiruan ART

Langkah pertama dalam proses pembuatan Jaringan Saraf Tiruan berbasis ART adalah inisialisasi bobot statistik. Dalam matriks ini, Lapisan Perbandingan (CL) pertama kali diinisialisasi, dan ini dilambangkan dengan "B". Untuk memulai bagian ini, rumus matematika berikut digunakan:

$$\mathrm{Bij} = < \mathrm{E}/\mathrm{E} + 1 - 1\,\mathrm{Vij}.$$

Ini harus memenuhi permutasi berikut:

$$\mathrm{M} = \dim(\mathrm{x});$$
$$\mathrm{E} > 1\ (\text{biasanya } \mathrm{E} = 2).$$

Matriks terbobot RL, dilambangkan sebagai "T", kemudian diinisialisasi sehingga:

$$\mathrm{Tij} = 1 - \mathrm{Vi,j}.$$

Dari sini, tingkat toleransi (dilambangkan sebagai "Rjo") ditentukan dengan rumus berikut:

$$0 < \mathrm{N0} < 1.$$

Penting untuk dicatat bahwa N0 yang tinggi akan menghasilkan diskriminasi statistik yang spesifik, tetapi sebaliknya, ambang batas N0 yang lebih rendah memungkinkan pengelompokan pola yang lebih kolektif dalam sistem ANN yang sifatnya tidak serupa. Dengan demikian, sistem ANN sebenarnya dapat dimulai dengan nilai N0 yang jauh lebih rendah, dan dari sana menaikkannya sesuai kebutuhan dan/atau yang dipersyaratkan.

Pelatihan Jaringan Syaraf Tiruan ART

Pelatihan pertama-tama dimulai dengan pembentukan matriks terbobot "B", yang merepresentasikan sisi RL, dan "T," yang mewakili sisi Lapisan Perbandingan (CL). Lebih lanjut, Jaringan Saraf Tiruan ART dapat terdampak oleh beberapa iterasi vektor masukan, yang mana tidak ada waktu untuk mencocokkan vektor masukan tertentu dengan nilai lain yang sesuai dan memiliki rata-rata, dilambangkan sebagai "X."

Parameter untuk menyiapkan pelatihan Jaringan Saraf Tiruan ART diatur sebagai berikut:

$$\mathrm{Bij} = \mathrm{Eci}/\mathrm{E} + 1 + \mathrm{k}\sum \mathrm{Ck}$$

Di mana:

$\mathrm{E} > 1;$

Ci = komponen ke-i dari vektor masukan "C," di mana nilai "j" kemudian akan dikaitkan dengan Neuron Pemenang, yang dilambangkan sebagai "Rj."

Selain itu, parameter yang dilambangkan sebagai "Tij" dari "T" ditentukan oleh rumus matematika berikut:

$$\mathrm{Tij} \;=\; \mathrm{Ci\ Vi} \;=\; 1 \;\ldots\; \mathrm{m}, \mathrm{m} \;=\; \dim(\mathrm{X}), \mathrm{j} \;=\; 1, \ldots\; \mathrm{n}.$$

Dalam contoh khusus ini, "j" mewakili Winning Neuron.

Operasi Jaringan Jaringan Saraf Tiruan ART

Setelah pelatihan Jaringan Saraf Tiruan ART selesai, fase selanjutnya adalah meluncurkan kompatibilitas jaringan, atau operasi sistem. Untuk memulai proses khusus ini (yang merupakan langkah pertama), iterasi "0" (di mana $\mathrm{X} \;=\; 0$), direpresentasikan oleh persamaan matematika berikut:

$$\mathrm{G2}(0) \;=\; 0 \text{ dan}$$
$$\mathrm{G1}(0) \;=\; 0.$$

Selanjutnya (pada langkah kedua), jika vektor masukan di mana $\mathrm{X} \;= /0$ maka vektor keluaran yang dilambangkan sebagai "r" pada Lapisan Perbandingan adalah vektor yang akan mengatur semua lapisan dalam sistem ANN. Ini dilambangkan sebagai "$\mathrm{r}(0) \;=\; 0$." Selanjutnya, jika vektor masukan di mana $\mathrm{X} \;= /0$, maka tidak akan ada Neuron spesifik yang tidak memiliki keunggulan lebih dari Neuron lain dalam sistem ANN.

Pada langkah ketiga, hanya Neuron yang terkait dengan RL yang akan aktif. Dengan demikian, dalam hal ini, jika $\mathrm{Rj} \;=\; 1$, dan $\mathrm{Ri} \;= /\mathrm{j} \;=\; 0$ akan menentukan vektor masukan mana (dilambangkan sebagai "r") yang akan menjadi keluaran sisi RL sistem ANN. Namun, jika beberapa Neuron memiliki nilai "d" yang sama, maka Neuron pertama yang memiliki nilai serendah mungkin akan dipilih, yang dilambangkan sebagai "j."

Jika bobot masukan statistik multidimensi digunakan, masukan yang spesifik untuk Lapisan Perbandingan akan ditentukan oleh rumus matematika berikut:

$$\mathrm{Pj} \;=\; \mathrm{Tj};\ \mathrm{Tj}, \text{dari vektor masukan "T."}$$

Neuron yang menang akan dilambangkan sebagai "$\mathrm{Pj} \;=\; 0.$"

Pada langkah keempat, klasifikasi statistik dipandang sebagai "Elemen Reset" dari sistem ANN. Akibatnya, semua proses Klasifikasi akan terhenti. Oleh karena itu, akan terdapat varians yang sangat besar antara vektor masukan "p" dan "x". Hal ini akan menghasilkan nilai "N" yang sangat rendah, yang selanjutnya dikenal sebagai "Elemen Reset" sistem ANN. Hal ini dilakukan sedemikian rupa sehingga $\mathrm{N} \;<\; \mathrm{N0}$. Oleh karena itu, jika semua Neuron diberi bobot dengan jenis masukan statistik yang sama, Neuron yang berbeda dalam komponen RL secara teknis akan menang. Namun, jika tidak ada Neuron yang benar-benar sesuai dengan vektor masukan dalam sistem ANN dalam tingkat varians yang ditentukan, maka langkah berikutnya akan segera diikuti.

Pada langkah kelima, Neuron yang sebelumnya tidak diketahui akan diberi vektor bobot statistik yang dilambangkan sebagai "Tj" dan "Bj" untuk mengasosiasikannya dengan vektor masukan "X." Keuntungan utama di sini adalah bahwa sistem ANN secara keseluruhan dan jaringan pembelajaran yang dimilikinya tidak akan "kehilangan," atau "melupakan," pola apa pun yang telah dipelajari sebelumnya. Tidak hanya akan dipertahankan, tetapi pola-pola baru lainnya yang dipelajari akan ditambahkan ke dalam sistem ANN. Proses ini sangat mirip dengan Jaringan Saraf Biologis (BNN).

Terakhir, pada langkah terakhir dan keenam, prosedur yang baru saja dijelaskan sebelumnya akan mengkategorikan lebih lanjut secara statistik semua kelas dan pola yang telah dilatih sejauh ini dalam sistem ANN.

Sifat-Sifat Jaringan Saraf ART

Daftar berikut merangkum beberapa fitur terbaik Jaringan Saraf ART, serta apa yang membedakannya dari Jaringan Saraf lainnya sebagaimana dijelaskan sejauh ini dalam bab ini:

1) Setelah jaringan spesifik ini stabil, sifat yang dikenal sebagai "Akses Langsung" akan menjadi sangat mirip dengan fungsi "Pengambilan Cepat" yang terdapat dalam Jaringan Saraf Biologis (BNN).
2) Proses Pencarian akan membantu menormalkan secara statistik Neuron Pemenang
3) Set data pelatihan Jaringan Saraf Tiruan ART dianggap stabil sehingga tidak akan terjadi perpindahan data setelah Neuron Pemenang dipastikan.
4) Pelatihan kemudian akan stabil hingga mencapai jumlah iterasi statistik yang terbatas.

Namun, terdapat beberapa kelemahan Jaringan Saraf Tiruan ART, yaitu sebagai berikut:

- Jaringan Saraf Tiruan ART menggunakan elemen Gain dan Reset, yang secara harfiah tidak relevan dengan Jaringan Saraf Tiruan Biologis.
- Sangat mungkin jika Neuron yang hilang, seluruh proses pembelajaran yang telah diperoleh oleh sistem ANN dapat dihilangkan.

Komentar Lebih Lanjut tentang Jaringan Saraf Tiruan ART 1 & ART 2

Perlu dicatat lebih lanjut bahwa Jaringan Saraf Tiruan ART sebenarnya dibagi lagi menjadi jenis Jaringan Saraf Tiruan ART 1 dan ART 2. Berikut ringkasan fitur-fiturnya:

1) Jaringan Saraf Tiruan ART 1:
   - Menggunakan struktur berlapis;
   - Menggunakan mekanisme umpan balik, tetapi berbeda dengan yang digunakan dalam Jaringan Saraf Tiruan Hopfield;
   - Menggunakan set data pelatihan BAM;
   - Menggunakan konsep "Winner Take All";
   - Menggunakan Inhibisi;
   - Menggunakan Fungsi Reset;
   - Memiliki Fitur Plastisitas;
   - Tidak berkinerja optimal ketika setidaknya satu atau lebih Neuron hilang atau bahkan mengalami malfungsi dalam sistem ANN;
   - Tidak transparan, dengan kata lain, masih rentan dianggap sebagai "Kotak Hitam".

2) Jaringan Saraf Tiruan ART 2:

- Dirancang khusus untuk memanfaatkan masukan Pelatihan Analog atau Berkelanjutan;
- Tidak memerlukan pengaturan atau penerapan sebelumnya;
- Pola (seperti pola pada set data kualitatif) dapat ditambahkan saat sistem ANN masih beroperasi;
- Selain itu, pola-pola yang disebutkan di atas dapat dikategorikan dan diklasifikasikan sebelum disalurkan ke sistem ANN;
- Matriks matematika "B" dan "T" juga cukup skalabel sehingga dapat diperluas lebih lanjut ke dalam sistem ANN jika diperlukan.

Studi Kasus ART 1: Memanfaatkan Pengenalan Ucapan

Dalam studi kasus khusus ini, konsep Pengenalan Ucapan digunakan untuk membedakan kata-kata berikut:

- Lima;
- Enam;
- Tujuh.

Dengan menggunakan desain Jaringan Saraf Tiruan (ANN) terkini, kata-kata ini diteruskan ke dalam larik matematis yang dikenal sebagai "Five Band Pass Filters". Energi yang diperoleh lebih lanjut dari keluaran sistem ANN kemudian dirata-ratakan secara statistik ke dalam interval 20 milidetik selama lima iterasi, yang berpuncak pada total 100 milidetik. Selain itu, matriks lima kali lima diimplementasikan ke dalam Bidang Geometri Kartesius, yang terdiri dari nilai biner 0 dan 1, yang dikaitkan dengan kata-kata yang diucapkan yang dijelaskan di atas.

Matriks masukan referensi juga disusun berdasarkan pengulangan kata-kata yang diucapkan oleh pengguna akhir, yang masing-masing diucapkan sebanyak 20 kali. Nilai-nilai ini kemudian dirata-ratakan selama 20 iterasi milidetik.

Aplikasi ini menggunakan bahasa pemrograman C, yaitu sebagai berikut:

Tampilan

```
“5”, “6”, or “7” (zero random noise) – choose input pattern (patterns are in three groups:
      5 patterns which represent the word “5” when it is used in different types of pronunciations:
      “6” similar to “5”
      “7” similar to “6”
Pattern # (0-random) -There are ten different input patterns that strongly correlate from the spoken
words of “5”, “6” and “7”, thus choose one
Create new pattern for: - specify how many patterns need to be assigned
```

Variabel-variabel berikut juga digunakan dalam kode sumber C:

```
PATT = the stored patterns;
PPATT = the previous inputs that are correlated with the speech patterns in the Comparison Layer
   of the ANN system;
T = the statistical weights that are assigned to the Neurons that are in the Comparison Layer;
TO = the statistical weights of a Neuron that is in the Comparison Layer and also correlated with
```

```
   the Winning Neuron that is found at the Recognition Layer;
TS = the status of the Recognition Layer Neurons;
BO = the statistical input to the Neurons in the Recognition Layer;
C = the outputs that are generated from the Recognition Layer in the ANN system;
INP = the input vector;
NR = the total number of patterns that are stored in the weights of both the Comparison Layer and
   the Recognition Layer;
GAIN = a stored pattern that correlates with 1 input and 2 inputs when there are no stored patterns
   in the ANN system
SINP = the total number of "1" values that are present in the input vector;
SC = the total number of "1" values that are present in the Output Layer of the ANN system;
STO = the total number of "1" values that are chosen for the speech patterns of the chosen words;
MAXB = the mathematical pointer which is used to best associate all of the input vectors that are
   present in the ANN system.
```

Versi modifikasi dari Jaringan Syaraf Tiruan ART 1 diberikan sebagai berikut:

$$\mathrm{D\ (modified)\ =\ min(D, D1)}$$

Di mana:

$\mathrm{D}\ =\ \mathrm{D}$ reguler dari ART 1'

$\mathrm{D1}\ =\ \mathrm{c/p}$; di mana juga $\mathrm{p}\ =$ jumlah nilai 1 dalam ujaran yang dipilih dari tiga angka, seperti yang dijelaskan sebelumnya.

Contohnya meliputi:

$$\mathrm{Input\ Vector\ 1111000000;\ x\ =\ 4}$$
$$\mathrm{Chosen\ pattern\ 1111001111;\ p\ =\ 8}$$
$$\mathrm{Comparison\ Layer\ 11110000000\ =\ 4}$$

Ini akan menghasilkan produk berikut, yang dihitung sebagai berikut:

$$\mathrm{D\ =\ c/x\ =\ 4/4\ =\ 1.0\ in\ regular\ ART-1}$$
$$\mathrm{D1\ =\ c/p\ =\ 4/8\ =\ 0.5}$$
$$\mathrm{D\ (modified)\ =\ min(D, D1)\ =\ 0.5}$$

4.6 COGNITRON DAN NEOCOGNITRON

Cognitron adalah jenis Jaringan Saraf Tiruan khusus yang telah diciptakan dan dirancang untuk penerapan Pola Pengenalan. Untuk mencapai tugas spesifik ini, Jaringan Saraf Tiruan berbasis Cognitron memanfaatkan sepenuhnya Neuron Inhibitor dan Eksitori. Konsep ini pertama kali dicetuskan pada tahun 1975, dengan memanfaatkan Jaringan Saraf Tiruan Tanpa Pengawasan. Dalam hal ini, model ini dimaksudkan untuk meniru proses inisiasi retina (yang terletak di bagian belakang mata). Eksperimen ini dianggap sebagai jenis "Pembelajaran Mendalam", dan konsep ini akan dibahas lebih lanjut secara lebih rinci nanti di bab ini.

Neocognitron juga dikembangkan pada awal 1980-an. Hal ini dilakukan untuk memperluas cakupan Cognitron, baik dari segi fungsionalitas maupun optimasi. Hal ini

meletakkan dasar bagi terciptanya apa yang dikenal sebagai Jaringan Saraf Tiruan jenis "Pembelajaran Mendalam Konvolusional", yang terjadi pada tahun 1989.

Dalam hal komposisi Cognitron, jaringan ini terutama terdiri dari banyak lapisan dan bahkan sub-lapisan, baik Neuron Inhibitor maupun Neuron Eksitori. Koneksi antara kedua jenis Neuron ini hanya dibangun pada koneksi yang telah dibangun pada lapisan di bawahnya dalam sistem ANN. Istilah teknis untuk ini dikenal sebagai "Kompetisi Koneksi" Neuron. Dengan kata lain, koneksi dibangun dari pendekatan bottom-up, bukan pendekatan top-down tradisional.

Untuk mengoptimalkan pelatihan sistem ANN, tidak semua Neuron digunakan atau diaktifkan; melainkan, pelatihan dicadangkan secara khusus untuk kelas Neuron yang dikenal sebagai "Grup Elit". Ini adalah Neuron yang dikhususkan untuk tugas tertentu dan untuk menghasilkan jenis keluaran tertentu dari sistem ANN. Perlu dicatat juga bahwa Neuron dalam "Grup Elit" adalah neuron yang telah dilatih sebelumnya. Dalam pendekatan bottom-up terkait konektivitas Neuron, sering kali terjadi tumpang tindih. Di sinilah Neuron juga dapat dikaitkan dengan interkoneksi lainnya.

Tumpang tindih semacam ini dapat menyebabkan penurunan kinerja dari dalam sistem ANN; oleh karena itu, konsep "Kompetisi" digunakan untuk mengatasi tumpang tindih ini. Pada titik ini, koneksi antar Neuron yang dianggap "lemah" akan terputus secara otomatis. Dengan "Kompetisi", terdapat pula redundansi, sehingga pemutusan koneksi ini tidak akan menghambat proses lain yang sedang berlangsung di dalam sistem ANN.

Struktur Cognitron telah dirancang berdasarkan prinsip arsitektur Multilevel, dan Neuron yang berada di antara dua lapisan spesifik selanjutnya disebut $\mathrm{L-I}$ dan $\mathrm{L-II}$, secara iteratif, dilambangkan sebagai "$2\mathrm{n}$." Iterasi ini dapat direpresentasikan sebagai berikut:

- L-I1;
- L-II1;
- L-I2;
- L-II2.

Operasi Jaringan Neuron Eksitasi dan Inhibitor

Keluaran spesifik Neuron Eksitasi dihitung secara matematis sebagai berikut:

Untuk Masukan Neuron Eksitasi:

$$\mathrm{Xi} = \sum\mathrm{k}\ \mathrm{AikYk};$$

Untuk Masukan Neuron Inhibitor:

$$\mathrm{Zi} = \sum\mathrm{k}\ \mathrm{BikVk}$$

Di mana:

Yk = keluaran dari lapisan sebelumnya dalam sistem ANN;

Vj = keluaran Neuron Inhibitor dari lapisan sebelumnya dalam sistem ANN;

$\mathrm{Aik\ dan\ Bik}$ = bobot statistik yang sesuai yang telah ditetapkan, dan juga disesuaikan lebih lanjut ketika Neuron tertentu dianggap lebih "aktif" daripada yang lain.

Ketika kedua rumus matematika di atas digabungkan, total, atau keseluruhan, keluaran agregat dari sistem ANN dihitung sebagai berikut:

$$Yi = f(Ni)$$

Di mana:

$$Ni = (1 + Xi)/(1 + Zj) - 1 = (Xi - Zi)/(1 + Zi)$$
$$f(Ni) = \{Ni \ldots \text{for } Ni > 0;\ 0 \ldots \text{for } Ni < 0\}.$$

Untuk Masukan Neuron Inhibitor

Keluaran Neuron-neuron ini dihitung secara matematis sebagai berikut:

$$V = \sum I\ CiYi;$$
$$\sum iCi = 1.$$

Pelatihan Awal Neuron Eksitorik

Set data awal yang digunakan pertama-tama ditetapkan ke Neuron Eksitorik dalam serangkaian iterasi statistik berdasarkan rumus berikut:

$$Obi = (q\sum j\ AjiY\text{^}2j)/(2v*);\ Obi = \text{the change in } Bi$$

Di mana:

Bi = bobot statistik koneksi yang terbentuk antara Neuron Inhibitor yang terletak di lapisan "L1" dan Neuron Eksitori "i" yang terletak di lapisan "L2." Perlu dicatat di sini bahwa "∑j" sebenarnya merupakan penjumlahan matematis bobot dari setiap Neuron Eksitori "L1" hingga Neuron "i" di lapisan L2.

Persamaan di atas dikembangkan dengan asumsi bahwa akan selalu ada Neuron aktif dalam sistem ANN. Namun, jika tidak ada aktivitas sama sekali, maka dua persamaan berikut secara otomatis menggantikan:

$$Oaji = q\text{^}rCjYj$$
$$Obi = q'Vi$$

Di mana:

$$Q\text{^}r < q.$$

Singkatnya, terdapat korelasi positif antara keluaran Inhibisi dan bobot statistiknya; ketika salah satu meningkat, yang lain juga akan meningkat dengan tingkat atau jumlah yang sama.

Inhibisi Lateral

Konsep kunci lainnya di sini adalah "Inhibisi Lateral". Di sinilah Neuron tertentu berada di setiap Lapisan Kompetisi sistem ANN. Dalam hal ini, Neuron Inhibitor sebenarnya

memperoleh masukan statistiknya dari Neuron Eksitoris di satu lapisan spesifik dengan bobot yang baru saja ditetapkan, dan dilambangkan sebagai "Gi". Hal ini direpresentasikan sebagai berikut:

$$V = \sum iGiYi$$

Di mana:

Yi = keluaran Neuron Eksitoris.

Dari sini, keluaran V dari Neuron Inhibitor $L2$ dihitung sebagai berikut:

$$O/I = f[1 + Yi/1 + V] - 1.$$

Neocognitron

Setelah kita mengulas Cognitron secara mendalam, penting untuk membahas lebih detail tentang apa itu Neocognitron. Sebagaimana telah disebutkan sebelumnya, Neocognitron dianggap sebagai versi Cognitron yang jauh lebih canggih. Neocognitron memiliki struktur hierarki dan secara khusus diarahkan untuk memahami bagaimana penglihatan manusia diproses.

Dalam struktur hierarki tersebut, terdapat dua kelompok lapisan, yang terdiri dari Sel Sederhana dan Sel Berlapis Banyak. Terdapat pula lapisan tebal yang berada di antara kedua struktur berbasis Seluler ini. Dalam pendekatan tiga tingkat ini, jumlah total Neuron justru berkurang secara top-down. Hal ini telah dirancang khusus agar Neocognitron dapat mengatasi berbagai masalah pengenalan yang dialami oleh Cognitron, dan bahkan berhasil di saat yang gagal. Ini termasuk gambar yang berada di posisi yang salah atau yang memiliki distorsi sudut.

4.7 JARINGAN BACKPROPAGATION BERULANG

Jaringan Saraf Tiruan Backpropagation telah diperkenalkan dan diulas secara mendalam sebelumnya dalam bab ini. Kini, fungsionalitas berulang dapat ditambahkan ke dalamnya, dan dengan itu, keluaran spesifik dari sistem ANN dapat secara otomatis diumpan balik ke masukan sistem ANN. Perlu dicatat bahwa hal ini hanya dapat dicapai dalam iterasi kecil. Konsep ini sebenarnya diperkenalkan pada tahun 1986 dan 1988, dan akhirnya diimplementasikan sepenuhnya ke dalam Jaringan Saraf Tiruan Backpropagation pada tahun 1991.

Dengan penerapan semacam ini, jumlah Lapisan Tersembunyi (Hidden Layer) dalam sistem ANN juga sangat minimal. Dalam konfigurasi ini, mekanisme penundaan diperkenalkan sehingga berbagai Loop Umpan Balik akan sepenuhnya independen satu sama lain di antara setiap iterasi, yang secara teknis juga dikenal sebagai "epoch". Jadi, setelah interval waktu pertama selesai, keluaran kemudian diumpan balik ke masukan yang terkait dengannya. Menariknya, setiap kesalahan yang berkorelasi dengan keluaran dari sistem ANN juga dapat berputar kembali sebagai masukan langsung untuk iterasi berikutnya dalam sistem ANN.

Misalnya, jika sistem ANN menerima masukan yang masing-masing dilambangkan sebagai "X1" dan "X2", ini akan dihitung sebagai iterasi pertama.

Setelah ini, bobot statistik untuk masukan juga dihitung dalam Jaringan Syaraf Tiruan Backpropagation, dan dari sini, semuanya dijumlahkan tanpa penyesuaian lebih lanjut hingga iterasi pertama benar-benar menyelesaikan siklusnya.

Kemudian, keluaran, yang masing-masing dilambangkan sebagai "Y1" dan "Y2", berputar kembali ke dalam sistem ANN untuk digunakan sebagai masukan sekali lagi pada iterasi kedua. Proses ini terus berulang hingga sistem ANN telah belajar dari kumpulan data baru yang telah dimasukkan ke dalamnya.

Jaringan Rekursif Penuh

Jaringan ini sebenarnya sangat mirip dengan mekanisme Rekursif yang baru saja dibahas. Namun, ada satu perbedaan utama. Alih-alih keluaran sistem ANN diumpan balik sebagai masukan, keluaran tersebut diumpan balik sebagai Lapisan. Jadi, pada akhir iterasi pertama, Lapisan Keluaran akan diumpan balik sebagai Lapisan Masukan ke dalam sistem ANN. Dengan demikian, Neuron Rekursif juga ditransposisi dengan cara yang sama.

Jaringan Backpropagation Rekursif Kontinu

Dalam situasi khusus ini, Mekanisme Rekursif yang ada dalam Jaringan Backpropagation terus berlanjut selamanya, tetapi setiap kali iterasi dasarnya menjadi lebih pendek. Secara matematis, hal ini dapat direpresentasikan sebagai berikut:

$$T(DYi)/(Dt) = -Yi + g\,(Xi + \sum j\ WijVj)$$

Di mana:

T = koefisien konstanta waktu;
Xi = masukan eksternal;
G = fungsi aktivasi Neuron;
Yi = keluaran dari sistem ANN;
Vj = keluaran Lapisan Tersembunyi Neuron dari sistem ANN.

Stabilitas juga diperkenalkan di sini, dan secara matematis direpresentasikan sebagai berikut:

$$Yi = g(Xi + \sum j\ WijVj).$$

BAB 5
JARINGAN SYARAF TIRUAN PEMBELAJARAN MENDALAM

Sesuai namanya, Jaringan Syaraf Tiruan Pembelajaran Mendalam, juga dikenal sebagai "DLNN", adalah Jaringan Syaraf Tiruan khusus yang mencapai tingkat pembelajaran mendalam tertentu. Secara spesifik, Pembelajaran Mendalam dapat didefinisikan secara teknis sebagai berikut: *Pembelajaran mendalam adalah bagian dari pembelajaran mesin di mana jaringan saraf tiruan, algoritma yang terinspirasi oleh otak manusia, belajar dari data dalam jumlah besar. … Pembelajaran mendalam memungkinkan mesin untuk memecahkan masalah kompleks bahkan ketika menggunakan kumpulan data yang sangat beragam, tidak terstruktur, dan saling terhubung.*

Misalnya, Pembelajaran Mendalam mengkaji kumpulan data yang jauh lebih kompleks daripada jenis sistem Jaringan Syaraf Tiruan lain yang telah diulas sejauh ini dalam buku ini. Pembelajaran Mendalam (Deep Learning) dapat menyelidiki lebih dalam kumpulan data yang sangat kompleks, baik yang bersifat kualitatif maupun kuantitatif. Pembelajaran Mendalam juga dapat menyelidiki dan menemukan tren tersembunyi dalam kumpulan data yang akan membantu mengoptimalkan keluaran yang dihasilkan oleh sistem ANN, dalam upaya untuk mendapatkan hasil yang diinginkan.

Pembelajaran Mendalam juga dapat mengurai, menyaring, dan menganalisis kumpulan data tersebut dengan cara yang jauh lebih canggih, memanfaatkan berbagai jenis algoritma matematika yang biasanya mencakup hal-hal berikut:

Metode komputasi logis lainnya;

- Metode linear;
- Metode nonlinier;
- Metode analitis lainnya;
- Metode heuristik;
- Teknik deterministik;
- Teknik stokastik.

Berdasarkan hal ini, definisi teknis lain dari Pembelajaran Mendalam (Deep Learning) dapat ditawarkan sebagai berikut: *DLNN adalah kelas khusus teknik Pembelajaran Mesin yang memanfaatkan banyak lapisan informasi berbasis nonlinier untuk pemrosesan ekstraksi fitur terawasi dan tak terawasi, serta untuk analisis dan klasifikasi pola.*

Sebagian besar Jaringan Saraf Tiruan yang telah dibahas dalam buku ini, meskipun rumit dari segi desain, masih digunakan untuk aplikasi yang dianggap "sederhana". Penting untuk dicatat bahwa kata "sederhana" yang digunakan sangat subjektif, dan apa yang tampak mudah bagi satu entitas mungkin sebenarnya tampak rumit bagi entitas lain. Dengan demikian, Pembelajaran Mendalam biasanya digunakan dalam aplikasi Jaringan Saraf Tiruan yang "berat" di mana dibutuhkan dataset berukuran Terabita atau bahkan Petabita untuk memasukkan masukan yang bermakna ke dalam sistem ANN yang dimaksud.

Mengingat besarnya dataset, salah satu komponen kunci yang sangat penting bagi sistem ANN adalah mempertahankan tingkat integrasi yang tinggi. Artinya, mengingat luas, beragam, dan cakupan kumpulan data yang sangat besar ini, semuanya harus bekerja sama secara mulus agar sistem ANN dapat "mencerna" semuanya secara efisien dan terpadu, sehingga keluaran yang dihasilkan tidak akan miring atau bias dalam bentuk apa pun.

Demikian pula dengan Pembelajaran Mendalam (Deep Learning), sistem ANN semacam ini harus mampu belajar dengan cepat, terlepas dari ukuran kumpulan data yang sangat besar. Sistem ini harus mampu menerima berbagai jenis kumpulan data ini secara konstan, tergantung pada persyaratan yang telah ditetapkan. Pembelajaran Mendalam juga mencoba meniru, atau mereplikasi, otak manusia semaksimal mungkin.

Sebenarnya, konsep Pembelajaran Mendalam bukanlah hal baru. Minat terhadap hal ini tumbuh seiring para ilmuwan mulai mengeksplorasi konsep Pembelajaran Mesin (yang menjadi fokus utama dalam Bab 2) dan bagaimana pembelajaran mesin dapat digunakan untuk memproses data dalam jumlah besar. Konsep Pembelajaran Mendalam (Deep Learning) juga pertama kali diimplementasikan ke dalam sistem ANN yang menggunakan prinsip Backpropagation. Hal ini pertama kali diperkenalkan pada tahun 1986.

Jaringan Saraf Tiruan Konvolusional (juga dikenal sebagai "CNN")-lah yang pertama kali mengadopsi dan menerapkan konsep Pembelajaran Mendalam. Motivasi untuk menciptakan CNN adalah upaya untuk memodelkan korteks visual otak manusia. Oleh karena itu, CNN yang diterapkan terutama terbatas pada aplikasi komersial yang banyak menggunakan pencitraan.

Perlu dicatat bahwa CNN pertama yang menggunakan Pembelajaran Mendalam membutuhkan waktu tiga hari penuh untuk memproses semua set data yang dimasukkan. Meskipun proses ini tampak sangat lambat menurut standar saat ini, pada tahun 1989, proses tersebut merupakan proses yang sangat cepat untuk sistem ANN tersebut. Setelah terbukti bahwa Pembelajaran Mendalam dapat diterapkan pada aplikasi visual dan pencitraan, langkah selanjutnya adalah penggunaannya untuk aplikasi Pemrosesan Ucapan dan Pengenalan Ucapan. Mereka memanfaatkan teknik lain yang secara teknis dikenal sebagai algoritma matematika berbasis "Support Vector Machine", atau disingkat "SVM".

Terobosan besar berikutnya untuk prinsip-prinsip Pembelajaran Mendalam muncul pada tahun 1996, ketika konsep yang dikenal sebagai "Jaringan Saraf Tiruan Penyimpanan dan Pengambilan Memori Besar" (juga dikenal sebagai "LAMSTAR" atau disingkat "LNN") diciptakan. Dalam situasi ini, jenis konfigurasi ini dibuat untuk membuat prediksi, investigasi, dan deteksi tertentu, serta keputusan berbasis operasional dari beragam kumpulan data besar. Konfigurasi ini mencakup karakteristik berikut:

- Deterministik;
- Stokastik;
- Spasial;
- Temporal;
- Logis;
- Deret waktu;
- Kuantitatif/Kualitatif.

Perlu dicatat bahwa konstruksi teoretis untuk LAMSTAR berawal dari tahun 1969, ketika sebuah alat Pembelajaran Mesin pertama kali diperkenalkan. Alat ini melakukan berbagai upaya untuk mereplikasi interkoneksi Neuron yang terdapat di antara berbagai lapisan dan korteks otak manusia. Untuk mencapai tujuan besar ini, LAMSTAR memanfaatkan teknik pemodelan yang sangat canggih, yang meliputi:
Integrasi dan pemeringkatan parameter;

- Koprosesing;
- Stokastik;
- Analitik;
- Entropi;
- Wavelet.

Kekuatan pemrosesan dan komputasi untuk sistem ANN semacam ini berasal dari konstruksi teoretis berikut:

- Prinsip Hebbian-Pavlovian;
- Pendekatan Kohonen Winner Take All;
- Komputasi Paralel.

Versi lain dari LAMSTAR dirilis pada tahun 2008, dan secara tepat disebut "LAMSTAR-2" atau disingkat "LNN-2". Versi ini dikembangkan untuk mengatasi beberapa kekurangan LAMSTAR, dan versi ini menawarkan daya komputasi dan pemrosesan yang jauh lebih besar.

Dua Jenis Jaringan Syaraf Tiruan Pembelajaran Mendalam

Selain konfigurasi Jaringan Syaraf Tiruan lain yang telah dibahas dalam bab ini, terdapat dua konfigurasi khusus lain yang juga dianggap sebagai Jaringan Syaraf Tiruan Pembelajaran Mendalam, yaitu sebagai berikut:

1) Mesin Boltzmann Dalam (DBM):
   Ini dianggap sebagai jenis Jaringan Syaraf Tiruan stokastik. DBM pertama kali diperkenalkan dan diterapkan pada tahun 2009, dan pada dasarnya bersifat tanpa pengawasan. Agar sistem ANN dapat belajar dari kumpulan data yang dimasukkan ke dalamnya, sebuah konsep yang dikenal sebagai "Keseimbangan Termodinamika" digunakan, yang didasarkan pada distribusi statistik Gibbs-Boltzmann. Proses pembelajaran yang sebenarnya dilakukan melalui teknik khusus yang disebut "Log-Likelihood", berdasarkan maksimisasi gradien. Dengan kata lain, kesalahan statistik antara kumpulan data dan model sistem ANN dianalisis dengan sangat cermat. Kelemahan utama DBM adalah membutuhkan daya komputasi dan pemrosesan yang sangat besar, sehingga cakupannya sangat terbatas dalam hal penerapan aplikasi.
2) Jaringan Syaraf Tiruan Pembelajaran Berulang Dalam (DRN):
   Jenis Jaringan Syaraf Tiruan Pembelajaran Dalam ini secara khusus memanfaatkan teknik Backpropagation (sebagaimana telah diulas secara mendalam sebelumnya dalam bab ini). Jaringan ini ditumpuk dalam pola linear pada interval waktu yang bervariasi, dan juga dimasukkan ke dalam input sistem ANN. Jaringan ini juga terlalu lambat untuk penerapan aplikasi skala luas, karena memerlukan penggabungan

algoritma matematika lain ke dalam komponen pembelajaran sistem ANN. Namun, perlu dicatat bahwa DRN telah sangat berhasil dalam memodelkan berbagai bahasa.

5.1 JARINGAN SYARAF TIRUAN LAMSTAR

Jaringan Syaraf Tiruan LAMSTAR sebenarnya telah diulas di subbagian sebelumnya. Pada dasarnya, terdapat dua jenis jaringan saraf tiruan, yang masing-masing dikenal sebagai "LAMSTAR-1" dan "LAMSTAR-2". Jenis Jaringan Syaraf Tiruan ini dirancang khusus untuk aplikasi yang ditujukan untuk pengambilan data, analisis, klasifikasi, prediksi, dan pengambilan keputusan. Mereka juga dirancang untuk digunakan dengan kumpulan data yang sangat besar, yang tidak dapat diproses semudah konfigurasi Jaringan Saraf Tiruan lain yang telah dibahas sejauh ini dalam bab ini. Oleh karena itu, dalam hal ini, alat tambahan yang paling disukai untuk Jaringan Saraf Tiruan LAMSTAR ini adalah Kohonen Self-Organizing Map (SOM).

Selain itu, Jaringan Saraf Tiruan LAMSTAR dirancang untuk menangani data kuantitatif dan kualitatif, ketika data tersebut bersifat multidimensi, dan bahkan tidak lengkap di banyak area. Selain itu, Jaringan Saraf Tiruan jenis ini dianggap sebagai apa yang dikenal sebagai "sistem cerdas pakar", di mana kumpulan data terus-menerus disempurnakan dan dioptimalkan untuk mendapatkan keluaran yang diinginkan. Jaringan Saraf Tiruan LAMSTAR dapat digunakan untuk membantu memperkirakan semua jenis data yang hilang dalam kumpulan data melalui teknik interpolasi dan ekstrapolasi.

Jenis Jaringan Saraf Tiruan ini dianggap sangat transparan, sehingga membantu meringankan gagasan fenomena "kotak hitam" yang sering dikaitkan dengan semua jenis Jaringan Saraf Tiruan. Alasan utamanya adalah Jaringan Saraf Tiruan LAMSTAR memiliki metode unik dalam menetapkan bobot statistik untuk masing-masing masukannya. Dengan kata lain, jenis Jaringan Saraf Tiruan ini telah terbukti sangat berhasil dalam aplikasi yang umumnya berkaitan dengan pengambilan keputusan dan pengenalan.

Dalam Jaringan Saraf Tiruan LAMSTAR, keluaran Neuron biasanya dihitung berdasarkan rumus matematika berikut:

$$Y = f[p\sum I = 1 WijXij]$$

Di mana:
$F(x)$ = fungsi nonlinier;
Wij = bobot Memori Asosiatif yang telah ditetapkan secara statistik untuk masukan.

Perlu dicatat juga bahwa dalam situasi ini, aktivasi spesifik Neuron menggunakan pendekatan semua atau tidak sama sekali. Dengan memanfaatkan penetapan bobot statistik yang unik untuk masukan, Jaringan Saraf Tiruan LAMSTAR tidak hanya memperhitungkan nilai-nilai yang tersimpan dalam memori sistem ANN, tetapi juga berbagai korelasi yang terjadi di antara keduanya. Selain itu, ketika Neuron aktif pada suatu titik waktu ketika iterasi deret waktu berikutnya akan terjadi dalam sistem ANN, bobot statistik korelasi ini juga meningkat secara proporsional.

Koneksi inilah yang juga mendukung kemampuan Jaringan Saraf Tiruan LAMSTAR untuk melakukan interpolasi dan ekstrapolasi, sebagaimana telah dibahas sebelumnya, tanpa harus memprogram ulang sistem ANN secara keseluruhan.

Elemen Struktural Jaringan Syaraf Tiruan LAMSTAR

Dalam hal penyimpanan dataset dan inputnya, Jaringan Syaraf Tiruan LAMSTAR memanfaatkan modul Kohonen SOM, yang semakin diperkuat dengan prinsip Memori Asosiatif. Sebagaimana telah disebutkan, alasan mengapa Jaringan Syaraf Tiruan LAMSTAR dapat menangani dataset sebesar itu adalah karena penggunaan algoritma komputasi matematis sederhana yang tersebar lebih jauh pada tautan-tautan ini. Hal ini berarti lebih sedikit pemrosesan dan daya komputasi yang dibutuhkan. Tautan-tautan ini, atau koneksi, juga dianggap sebagai penggerak utama dalam keseluruhan sistem Jaringan Syaraf Tiruan, dengan menghubungkan modul-modul SOM lebih lanjut.

Karena berbagai tautan dan koneksi yang diterapkan dalam sistem Jaringan Syaraf Tiruan ini, sistem ini kini hingga taraf tertentu menyerupai Sistem Saraf Pusat (SSP) otak manusia. Lebih lanjut, di sebagian besar sistem berbasis SOM, setiap Neuron diperiksa secara cermat untuk mengetahui kedekatannya dengan rentang numerik vektor input yang saat ini ada di seluruh sistem ANN. Namun, dalam LAMSTAR NN, hanya pengelompokan Neuron yang lebih kecil (dilambangkan sebagai "q") yang dapat diperiksa, yang tentu saja merupakan kerugian besar dalam hal ini. Penentuan himpunan khusus ini diatur oleh tautan, atau koneksi, yang ada dalam sistem ANN.

Perlu dicatat juga bahwa mesin utama LAMSTAR NN adalah penjumlahan matematis aktual dari semua tautan, atau titik, koneksi yang baru saja diulas sejauh ini. Selain itu, bobot statistik yang diberikan kepada mereka diperbarui secara waktu nyata (real-time) dengan jumlah lalu lintas yang ada pada simpul tautan dan koneksi ini dalam sistem ANN.

Algoritma Matematika yang Digunakan untuk Menetapkan Bobot Statistik untuk Input dan Tautan dalam Modul SOM dalam Sistem ANN

Setiap kali input baru ditambahkan ke dalam sistem ANN, terutama input dari set data pelatihan, LAMSTAR NN akan memeriksa dengan cermat semua vektor bobot penyimpanan untuk setiap modul (dilambangkan sebagai "i"), dan membandingkannya dengan bobot statistik yang berpotensi ditetapkan untuk input dari set data tersebut. Dari pemeriksaan cermat ini, "Neuron Pemenang" (sebagaimana dibahas sebelumnya di bab ini) kemudian dihitung dengan rumus matematika berikut:

$$\mathrm{D(j,j)} \;=\; ||\mathrm{Xj} - \mathrm{Wj}|| \;<\; ||\mathrm{Xj} - \mathrm{Wk} \;= /\mathrm{j}|| \;=\; \mathrm{d(j,k)}.$$

Selain itu, bobot statistik yang baru saja dijelaskan dapat disesuaikan lebih lanjut jika diperlukan, untuk mendapatkan optimalisasi dan keandalan semaksimal mungkin. Hal ini dilakukan dengan teknik matematika khusus lainnya, yang secara teknis dikenal sebagai "Fungsi Jarak Hamming" (dilambangkan sebagai "Dmax"), dan dapat direpresentasikan sebagai berikut:

$$\mathrm{Dmax} \;=\; \max[\mathrm{d(xiWi)}].$$

Selain itu, sebagaimana telah disebutkan sebelumnya di subbagian terakhir, Jaringan Saraf Tiruan (ANN) LAMSTAR mengandung banyak interkoneksi, atau tautan, antara lapisan masukan dan lapisan keluaran sistem ANN. Meskipun tautan-tautan ini dapat dianggap "dinamis", tautan-tautan ini juga perlu diperbarui untuk optimasi dan keandalan. Sekali lagi, hal ini dilakukan dengan menetapkan nilai bobot statistik yang berbeda untuk berbagai interkoneksi ini, sehingga dapat dihnitung dan ditetapkan sesuai dengan rumus berikut:

$$\mathrm{Li,j/k,m\ ^{*}(t+1)\ =\ Li,j/k,m\hat{}(t)\ +\ VL;}$$
$$\mathrm{Li,j/k,m\ *\ (t+1)\ =\ Li,j/k,m\hat{}(t+1)\ VM,s=/1;}$$
$$\mathrm{L(0)\ =\ 0}$$

Di mana:
$\mathrm{Li,j/km}\ =$ merepresentasikan tautan Neuron Pemenang (dilambangkan sebagai "I") dalam modul keluaran (dilambangkan sebagai "j").

Bobot statistik sebagaimana dijelaskan dalam persamaan di atas juga dapat membantu mengatur aliran masukan dari kumpulan data dalam sistem ANN sehingga hanya daya pemrosesan dan komputasi yang dibutuhkan yang digunakan, dan tidak lebih. Dalam banyak aplikasi yang benar-benar menggunakan ANN LAMSTAR, satu-satunya interkoneksi atau tautan yang dipertimbangkan untuk diperbarui adalah yang berada di antara lapisan SOM dan keluaran sistem ANN. Namun, interkoneksi, atau tautan, antara berbagai modul SOM tidak diperbarui sama sekali.

Seperti yang telah disebutkan sebelumnya, salah satu komponen kunci ANN LAMSTAR adalah "Melupakan" dan "Inhibisi". Dalam hal yang pertama, komponen ini, yang digabungkan dengan apa yang dikenal sebagai "Faktor Melupakan", dilambangkan sebagai "F", dapat diatur ulang pada berbagai interval yang telah ditentukan sebelumnya. Hal ini dapat dilambangkan sebagai $\mathrm{k\ =\ sK,s\ =\ 0,1,2,3,}$ dan seterusnya., di mana K merupakan nilai konstanta numerik yang telah ditentukan. Secara matematis, hal ini direpresentasikan sebagai:

$$\mathrm{L\ ^{*}(k+1)\ =\ FL(k)}$$

Di mana:
$0\ >\ \mathrm{F}\ >\ 1\ =$ Faktor Lupa yang telah ditentukan sebelumnya.

Penting untuk dicatat pada titik ini bahwa algoritma matematika lain juga dapat menggantikan persamaan di atas, yang dikenal sebagai "Algoritma Lupa", di mana nilai $\mathrm{L(k)}$ diatur ulang pada setiap $\mathrm{k\ =\ sK,s\ =\ 0,1,2,3,}$ dan seterusnya. Algoritma ini dapat direpresentasikan sebagai berikut:

$$\mathrm{F(i)\ =\ (1-z)\hat{}1\ L(k),0\ <\ z\ <<<<}$$
$$\mathrm{1\ I\ =\ (k-sK)}$$

Di mana:

Z = nilai numerik tertinggi untuk mencapai "$Ks < k$", sehingga "i" dimulai dari awal pada nilai 0, dan selanjutnya meningkat nilainya pada setiap iterasi dalam sistem ANN.

Sehubungan dengan "Penghambatan", ini harus tertanam dan diprogram ke dalam sistem ANN sebelum dapat dieksekusi di lingkungan produksi. Sehubungan dengan NN LAMSTAR, ini biasanya disertakan dengan menetapkan Neuron yang dipilih sebelumnya di lapisan masukan.

Tinjauan Umum Prosesor dalam Jaringan Syaraf Tiruan LAMSTAR

Sebagaimana telah diulas sebelumnya di subbagian sebelumnya, Jaringan Syaraf Tiruan LAMSTAR memanfaatkan apa yang dikenal sebagai "Pembelajaran Mendalam". Dengan fungsionalitas tambahan ini, jaringan ini dapat menghitung keluaran dengan menggunakan prosesor khusus agar sistem Jaringan Syaraf Tiruan dapat digunakan dalam aplikasi yang jauh lebih besar dan kompleks. Selain itu, untuk memfasilitasi daya pemrosesan dan kecepatan komputasi, sistem Jaringan Syaraf Tiruan dapat memanfaatkan konsep pemrosesan paralel. Prosesor Jaringan Syaraf Tiruan LAMSTAR sering ditemukan pada masukan lapisan SOM sistem Jaringan Syaraf Tiruan.

Iterasi Pelatihan versus Iterasi Operasional

Salah satu keunggulan terbesar sistem Jaringan Syaraf Tiruan pada umumnya adalah kemampuannya untuk terus berlatih tanpa henti 24/7/365, selama dataset yang bersih dan robust terus-menerus diberikan. Namun, seperti yang telah dijelaskan sebelumnya di subbagian sebelumnya, hal ini tidak berlaku untuk Jaringan Syaraf Tiruan LAMSTAR. Ini hanya dapat beroperasi dalam mode siklus iteratif. Dengan kata lain, Jaringan Saraf Tiruan LAMSTAR hanya dapat berjalan dan beroperasi dalam pengujian dan operasional, dengan mode mulai-berhenti.

Namun, untuk lebih mengoptimalkan kinerja jaringan sistem Jaringan Saraf Tiruan (ANN), sejumlah uji coba perlu diimplementasikan agar Jaringan Saraf Tiruan LAMSTAR dapat mengaktifkan Neuron sehingga kumpulan data aktual dapat mulai dimasukkan ke dalamnya.

Masalah Data yang Hilang dalam Jaringan Saraf Tiruan LAMSTAR

Sebagaimana telah disebutkan sebelumnya, Jaringan Saraf Tiruan LAMSTAR dapat berjalan tanpa adanya data yang hilang yang mungkin ada dalam kumpulan data. Hal ini dapat dicapai dengan menjumlahkan secara statistik nilai-nilai keseluruhan "k" yang ada.

Proses Pengambilan Keputusan Jaringan Syaraf Tiruan LAMSTAR

Secara keseluruhan, struktur jaringan Jaringan Syaraf Tiruan LAMSTAR-1 dan Jaringan Syaraf Tiruan LAMSTAR-2 sangat mirip. Kedua jenis Jaringan Syaraf Tiruan ini bahkan memiliki proses pengambilan keputusan yang sama dalam hal bagaimana masukan dan set data terkait akan digunakan untuk menghitung keluaran dari sistem Jaringan Syaraf Tiruan. Algoritma pengambilan keputusan dapat direpresentasikan secara matematis sebagai berikut:

$$M\sum k(w)\ I, nL\ k(w)\ >\ M\sum k(w)\ I, jL\ k(w)\ Vi, j, k, n, j = /n$$

Di mana:

I = modul keluaran ke-I;

N = Neuron Pemenang;

$K(w)$ = modul keluaran;

M = bobot tautan yang telah ditetapkan antara Neuron Pemenang dalam modul masukan (dilambangkan sebagai "k"), dan Neuron (dilambangkan sebagai "j") pada lapisan keluaran ke-I.

Fungsionalitas Analisis Data dalam Jaringan Saraf Tiruan LAMSTAR

Sebagaimana telah disebutkan di bab pertama buku ini, data dan kumpulan data terkaitnya merupakan "bahan bakar" yang menggerakkan sistem ANN, dan yang membuatnya menghasilkan keluaran yang diinginkan. Namun, satu aspek kuncinya adalah data harus selalu dibersihkan dan dioptimalkan. Hal ini bahkan berlaku untuk Jaringan Saraf Tiruan LAMSTAR. Faktanya, sebagian besar informasi dan data yang terdapat dalam Jaringan Saraf Tiruan semacam ini sebenarnya berada dalam bobot statistik yang telah ditetapkan untuk berbagai tautan, atau interkoneksi, sebagaimana telah dijelaskan secara luas sejauh ini.

Oleh karena itu, LAMSTAR bahkan dapat digunakan sebagai Analisis Data untuk sistem ANN. Dalam hal ini, data masukanlah yang dapat dianalisis lebih lanjut, dalam hal analisis berbagai lapisan masukan dan kumpulan data terkait yang digunakan. Selain itu, tingkat korelasi statistik antar kumpulan data ini juga dapat diperiksa. Dalam kebanyakan kasus, analisis yang dapat dilakukan oleh LAMSTAR NN merupakan proses dua langkah, yaitu sebagai berikut:

1) Penetapan konfigurasi analisis yang akan dilakukan;
2) Setelah hal di atas tercapai, analisis lebih lanjut dapat menggantikan bobot statistik dan kumpulan data yang berkorelasi dengan tautan, atau interkoneksi, dalam LAMSTAR NN.

Istilah "analisis" dapat bersifat luas, tergantung pada jenis aplikasi yang digunakan sistem ANN, dan keluaran yang diinginkan yang ingin dicapai darinya. Untuk keperluan LAMSTAR NN, analisis berarti memberikan wawasan lebih lanjut tentang masalah aktual yang coba dipecahkan oleh aplikasi tersebut. Perlu dicatat juga bahwa informasi/data apa pun yang diperoleh lebih lanjut dari fase analisis ini dapat dioptimalkan lebih lanjut dalam hal kinerja dan kecepatan jika di kemudian hari diputuskan bahwa Neuron atau Lapisan Input dan/atau Output tambahan perlu ditambahkan atau dihapus.

Dari sini, klaster statistik yang terkait dengan tautan yang memiliki nilai numerik tertinggi akan menentukan lebih lanjut tren yang diantisipasi dari kumpulan data yang digunakan sebagai input ke dalam sistem ANN. Dari sini, klaster-klaster tersebut kemudian akan "berkolaborasi" untuk menghasilkan keluaran yang diinginkan yang akan dihitung oleh sistem ANN.

Perlu dicatat bahwa analisis, yang dapat dilakukan oleh LAMSTAR NN, dapat dilakukan kapan saja ketika sistem ANN yang sebenarnya sedang mempelajari input dan kumpulan data yang telah dimasukkan ke dalamnya, dan menghasilkan keluaran yang diinginkan. Khususnya selama fase pelatihan sistem ANN, LAMSTAR NN akan menemukan tautan, atau interkoneksi, dengan nilai statistik tertinggi yang telah ditetapkan, dan dari sana, mengambil informasi/data

relevan apa pun dari modul SOM yang memiliki asosiasi lebih lanjut dengan tautan atau koneksi tersebut, sebagaimana dijelaskan sebelumnya.

Proses yang disebutkan di atas dapat dicapai melalui dua pendekatan terpisah dan berbeda, yaitu sebagai berikut:

1) Memilih dan menyebarkan tautan yang memiliki nilai numerik yang jauh melebihi ambang batas yang telah ditentukan sebelumnya;
2) Memilih dan menyebarkan sejumlah tautan atau interkoneksi yang telah ditentukan sebelumnya yang memiliki nilai statistik tertinggi yang terkait dengannya.

Komponen kunci lain dari komponen analisis LAMSTAR NN adalah kemampuannya untuk mengekstrak fitur unik yang terdapat dalam sistem ANN. Fitur-fitur ini juga dapat dihapus jika dianggap perlu. Terdapat beberapa properti untuk hal ini, yaitu sebagai berikut:

1) Lapisan memori dan/atau input/output paling signifikan:
   Hal ini sebenarnya dapat diekstraksi menggunakan matriks matematika yang dilambangkan sebagai

$$\text{"A(I, j)".}$$

   Di mana:
   I = Neuron Pemenang dalam model penyimpanan SOM yang terdapat dalam Jaringan Saraf LAMSTAR.

2) Lapisan memori dan/atau input/output paling tidak signifikan:
   Dalam kasus khusus ini, Neuron Pemenang ditentukan dengan rumus matematika berikut:

$$[i*, s*/dk]: L(I, s/dk) > L(j, p/dk)$$

Di mana:

P tidak sama dengan "S";

$L(I, s/dk)$ = bobot statistik hubungan antara Neuron Pemenang (dilambangkan sebagai “j”), di lapisan mana pun (dilambangkan sebagai “p”), serta lapisan keluaran Neuron yang dilambangkan sebagai “dk”.

3) Modul SOM Paling Signifikan:
   Ini dihitung dengan persamaan matematika berikut:

$$S^{**}(dk): \sum i(\{L(I, s/dk)\} > \sum j(\{L(j, p/dk)\}$$

4) Modul SOM Paling Signifikan:
   Ini dihitung dengan persamaan matematika berikut:

$$L(I, s/dk) > L(j, s/dk)$$

Catatan: Persamaan di atas dapat diterapkan untuk setiap Neuron (dilambangkan sebagai "j") untuk Modul SOM yang sama yang terdapat dalam Jaringan Saraf Terstruktur (NN) LAMSTAR.

5) Redundansi:

Ini dapat diekstrapolasi lebih lanjut sebagai berikut:

Setiap kali Neuron tertentu (dilambangkan sebagai "I" dalam kasus khusus ini) di lapisan masukan SOM dianggap sebagai yang unggul, Neuron tersebut juga dianggap memiliki masukan unggul yang seharusnya digunakan juga di lapisan masukan SOM tersebut. Ini dikenal sebagai "Redundansi."

6) Redundansi Informasi Nol:

Dalam kasus khusus ini, jika hanya ada satu Neuron yang selalu dianggap sebagai pemenang dalam lapisan SOM tertentu (dilambangkan sebagai "k" untuk tujuan ini), maka lapisan tertentu ini sama sekali tidak akan berisi informasi/data yang relevan.

Seperti yang telah disebutkan sebelumnya, Jaringan Saraf Tiruan LAMSTAR memiliki dua properti yang berbeda, yaitu sebagai berikut:

1) Fitur Korelasi:

Di sinilah lapisan SOM yang paling signifikan (baik berbasis masukan maupun keluaran) berisi Neuron yang paling signifikan secara statistik untuk berbagai faktor yang terkait dengannya (dilambangkan sebagai "m"), dengan asumsi bahwa lapisan-lapisan tersebut berkorelasi dengan keluaran yang sama yang telah dihitung oleh sistem Jaringan Saraf Tiruan. Semua ini dicapai dengan apa yang secara teknis dikenal sebagai "Aturan Pengaturan Lapisan Korelasi", dan secara matematis direpresentasikan sebagai berikut:

$$m - 1 \sum i = 1 \, i(\text{per output decision “DK”}).$$

Selain itu, pada titik khusus ini, konsep statistik Korelasi Otomatis dan Korelasi Silang dapat digunakan pada setiap iterasi berbasis waktu di LAMSTAR NN sebagaimana dianggap perlu.

2) Fitur Interpolasi/Ekstrapolasi:

Dalam kasus ini, Neuron tertentu [(dilambangkan sebagai “N(I, p)”] dianggap “diinterpolasi” atau “diekstrapolasi” jika memenuhi kondisi yang ditetapkan oleh persamaan matematika ini:

$$\sum q \{L(I, p/w, q - dk)\} > \sum \{L(v, p/w, q - dk)\}$$

Di mana:

I = berbagai Neuron dalam Modul SOM tertentu;

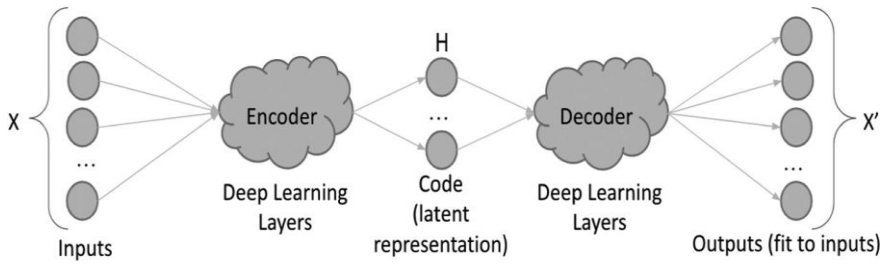
$\{L(v, p/w, q - dk)\}$ = menyatakan tautan, atau interkoneksi, yang berada di dalam Lapisan Korelasi dalam Jaringan Saraf Tiruan (NN) LAMSTAR (yang dilambangkan sebagai “$V\ (p/q)$”).

Penting juga untuk dicatat bahwa hanya ada satu Neuron Pemenang untuk setiap masukan yang digunakan (yang dilambangkan sebagai "N(w, q)").

Sejauh ini dalam bab ini, kita telah meninjau secara ekstensif konsep-konsep teoretis yang terkait dengan Jaringan Saraf Tiruan Jaringan. Sisa bab ini sekarang dikhususkan untuk penerapan teori ini.

5.2 AUTOENCODER JARINGAN SYARAF TIRUAN PEMBELAJARAN MENDALAM

Autoencoder adalah jenis jaringan saraf tiruan pembelajaran mendalam yang digunakan untuk mempelajari penyandian yang efisien untuk sekumpulan data secara tanpa pengawasan. Pada dasarnya, autoencoder mencoba menyalin Input-nya ke Output-nya melalui lapisan penyandian yang dibatasi, sehingga menghasilkan penyandian yang diinginkan. Autoencoder telah efektif digunakan untuk memecahkan berbagai masalah seperti makna semantik kata, pengenalan wajah, dan pemeliharaan prediktif (yang akan dijelaskan di bagian aplikasi bab ini).



Arsitektur dasar autoencoder ditunjukkan pada gambar di atas. Lapisan masukan dan keluaran memiliki jumlah simpul (x) yang sama dengan kumpulan data yang akan dikodekan. Di tengah terdapat lapisan tersembunyi dengan jumlah simpul kurang dari x, tempat representasi terkode (atau laten) (H) akan dipelajari. Jaringan saraf tiruan pembelajaran mendalam di sebelah kiri belajar mengodekan X menjadi H, sementara jaringan saraf tiruan pembelajaran mendalam di sebelah kanan belajar mendekodekan H menjadi X' dengan tujuan meminimalkan selisih antara X dan X' (dikenal sebagai kesalahan rekonstruksi). Karena autoencoder belajar menghasilkan X' dari X, data itu sendiri menyediakan label bagi model untuk dilatih, sehingga menjadikannya pendekatan pembelajaran tanpa pengawasan (belajar dari kumpulan data apa pun tanpa harus memberi label pada keluaran yang diinginkan).

Teknik pembelajaran mendalam yang umum seperti propagasi balik digunakan untuk mengurangi kesalahan rekonstruksi dengan mengoptimalkan encoder untuk menghasilkan kode yang lebih baik yang dapat digunakan dekoder untuk merekonstruksi X. Dengan kesalahan rekonstruksi yang kecil, lapisan tengah merepresentasikan informasi esensial (atau laten) dalam X dengan semua noise dan redundansi dihilangkan. Hal ini serupa dengan mengompresi berkas komputer menjadi representasi yang lebih kecil menggunakan sesuatu seperti Zip. Salah satu perbedaannya adalah Zip merupakan pengodean tanpa kehilangan

(lossless) sehingga dekoder Zip dapat merekonstruksi berkas asli dengan sempurna, sementara autoencoder merekonstruksi X dengan beberapa kesalahan intrinsik.

Dengan memilih nilai h terkecil dengan kesalahan rekonstruksi yang dapat diterima, autoencoder dapat digunakan untuk mengurangi dimensi tanggal input dari x menjadi h tanpa kehilangan informasi signifikansi dari X asli (dikenal sebagai reduksi dimensionalitas). Lapisan kode juga dapat digunakan untuk menentukan hubungan dalam data input. Misalnya, nilai yang dikodekan untuk kata seperti "London" harus mendekati kata "Inggris" dan "Paris". Atau, pengodean citra wajah Anda yang baru harus mendekati pengodean wajah Anda sebelumnya.

Kegunaan lain dari autoencoder adalah deteksi anomali. Dengan autoencoder yang mencoba merekonstruksi data baru yang tidak digunakan dalam pelatihan, set masukan yang direkonstruksi dengan buruk merupakan indikator bahwa data baru ini berbeda dari data asli atau anomali. Masukan yang direkonstruksi paling buruk adalah yang paling berbeda dari data pelatihan dibandingkan dengan masukan lainnya. Kesalahan rekonstruksi masukan individual ini memberikan informasi yang dapat digunakan untuk menjelaskan anomali pada data baru. Contoh penggunaan autoencoder untuk pemeliharaan prediktif diberikan di bagian aplikasi bab ini.

Aplikasi Jaringan Saraf Tiruan

Secara keseluruhan, bab ini telah mengkaji konsep Jaringan Saraf Tiruan, terutama dari perspektif teoretis. Penting untuk dicatat bahwa semua teori yang baru saja dijabarkan memiliki satu tujuan utama: meletakkan fondasi bagi aplikasi modern yang kita lihat dan gunakan sehari-hari. Misalnya, dalam dunia Keamanan Siber, banyak aplikasi berbasis Jaringan Saraf Tiruan kini digunakan untuk triase ancaman Siber, terutama untuk menyaring positif palsu, sehingga tim Keamanan TI dapat dengan cepat mengidentifikasi dan menindaklanjuti vektor ancaman yang nyata.

Namun, Jaringan Saraf Tiruan juga digunakan dalam dunia yang dikenal sebagai "Internet of Things" atau disingkat "IoT". Di sinilah semua objek yang berinteraksi dengan kita sehari-hari, baik di dunia virtual maupun fisik, saling terhubung melalui berbagai jalur komunikasi berbasis jaringan.

Karena Jaringan Saraf Tiruan kini mulai diterapkan di berbagai jenis aplikasi, muncul anggapan bahwa banyak dari aplikasi ini sangat mahal untuk diperoleh dan rumit untuk diterapkan. Namun kenyataannya tidak demikian. Sebagai contoh, banyak aplikasi berbasis Jaringan Saraf Tiruan kini tersedia melalui banyak Penyedia Cloud terbesar. Hal ini tentu saja memberikan banyak keuntungan, seperti harga bulanan yang tetap dan terjangkau, dan yang terpenting, skalabilitas, sehingga Anda dapat meningkatkan atau menurunkan kebutuhan hanya dalam hitungan detik. Di subbagian selanjutnya dari bab ini, kami akan merinci beberapa Penyedia Cloud utama ini.

Penyedia Cloud Utama untuk Jaringan Saraf Tiruan

Dalam hal ini, beberapa raksasa di bidang ini adalah Amazon Web Services (AWS) dan Microsoft Azure. Ada juga penyedia lain, dan mereka juga akan dibahas.

1) Amazon Web Services (AWS):

Perlu dicatat bahwa AWS adalah Penyedia berbasis Cloud tertua, pertama kali diluncurkan pada tahun 2006. Sejak saat itu, mereka secara konsisten menduduki peringkat sebagai salah satu Platform Cloud teratas yang digunakan, menurut "Gartner's Magic Quadrant." Namun, mereka dikenal lebih mahal, dan menawarkan solusi yang lebih kompleks yang kurang cocok untuk UKM yang mencoba menerapkan aplikasi berbasis Jaringan Neural.

2) Microsoft Azure:

Microsoft Azure (alias "Azure") telah bertahan di posisi kedua yang sangat stabil, tepat setelah AWS, juga menurut "Gartner's Magic Quadrant." Azure sangat menarik bagi bisnis yang memiliki beban kerja berbasis lama dan bagi mereka yang ingin menerapkan dan mengimplementasikan aplikasi Jaringan Neural baru pada Platform berbasis Cloud. Lebih penting lagi, mereka juga menawarkan platform yang sangat terspesialisasi untuk apa yang dikenal sebagai aplikasi "Platform as a Service" (alias "PaaS"), Penyimpanan Data, Pembelajaran Mesin (yang merupakan topik utama Bab 2), serta Internet of Things (IoT), dengan layanan yang berbasis pada semua layanan khusus ini. Selain itu, pengembang perangkat lunak yang tertarik untuk menerapkan aplikasi berbasis NET di Cloud kemungkinan besar akan menganggap Azure sebagai platform terbaik untuk digunakan dalam hal ini. Microsoft juga telah mengadopsi penggunaan platform berbasis perangkat lunak lain ke Azure, terutama Linux dan bahkan Oracle. Faktanya, 50 persen beban kerja berbasis Cloud di Azure berbasis Linux. Bahkan, seperti yang juga dicatat oleh Gartner:

"Microsoft memiliki visi unik untuk masa depan yang melibatkan pengintegrasian mitra teknologi melalui penawaran asli pihak pertama seperti yang ditawarkan oleh VMware, NetApp, Red Hat, Cray, dan Databricks"

3) Google Cloud Platform (GCP):

Google Cloud Platform, alias "GCP", pertama kali diluncurkan pada tahun 2018. Dibandingkan dengan AWS dan Azure, GCP berada di peringkat ketiga dalam hal Penyedia Berbasis Cloud. GCP terutama dikenal dengan penawaran berbasis Big Data Cloud-nya, dan akan segera memanfaatkan platform mereka untuk melayani sistem berbasis SAP dan CRM. GCP juga dikenal dengan Otomasi, Kontainer, Kubernetes, dan bahkan Tensor Flow. GCP terutama berfokus pada pemanfaatan Platform Open Source, seperti Linux.

4) Alibaba Cloud:

GCP pertama kali diluncurkan pada tahun 2017, dan mereka terutama melayani pasar Tiongkok, baik dari sudut pandang sektor swasta maupun pemerintah, terutama untuk membangun platform Hybrid Cloud.

5) Oracle Cloud Infrastructure (OSI):

Oracle Cloud Infrastructure, juga dikenal sebagai "OCI", pertama kali diluncurkan pada tahun 2017. Mereka terutama menawarkan Mesin Virtual (alias "VM") yang terutama mendukung Beban Kerja Basis Data Oracle dan layanan berbasis Cloud Infrastruktur sebagai Layanan (alias "IaaS") dasar lainnya.

6) IBM Cloud:

Secara tradisional, IBM dikenal karena dominasinya yang luar biasa di pasar, baik untuk segmen Mainframe maupun Komputasi Pribadi. Namun, ketika mereka mulai tergerus, mereka mencoba merangkul Platform berbasis Cloud dengan cara yang serupa dengan AWS dan Axure. Dalam hal ini, penawaran berbasis Cloud mereka mencakup Platform Kontainer dan bentuk penawaran PaaS lainnya. IBM Cloud terutama ditujukan untuk pasar yang masih menggunakan mainframe IBM serta beban kerja IBM tradisional lainnya. IBM juga terkenal dengan paket AI-nya yang dikenal sebagai "Watson".

5.3 KOMPONEN JARINGAN NEURAL AMAZON WEB SERVICES & MICROSOFT AZURE

Di bagian bab ini, kita akan berfokus pada berbagai komponen yang menghubungkan Kecerdasan Buatan dengan platform utama AWS dan Azure.

Amazon Web Services (AWS)

Sebagaimana telah disebutkan, AWS memiliki banyak komponen yang dapat dimanfaatkan oleh pengguna akhir, tidak hanya untuk Kecerdasan Buatan. Namun, dalam hal penerapan Kecerdasan Buatan, berikut beberapa komponen yang dapat digunakan oleh bisnis apa pun:

Amazon SageMaker

Paket ini awalnya diluncurkan pada tahun 2017. Ini adalah jenis platform Kecerdasan Buatan khusus yang memungkinkan pengembang perangkat lunak dan ilmuwan data untuk membuat, melatih, dan mengimplementasikan model AI pada Infrastruktur berbasis Cloud. Dalam hal ini, subset yang sangat penting dari Amazon SageMaker dikenal sebagai "Jupyter Notebook". Notebook ini menggunakan jenis kode sumber tertentu, yaitu Python, dan algoritma AI dapat diintegrasikan ke dalam infrastrukturnya. Penting untuk dicatat bahwa dengan "Jupyter Notebook", file .EXE dapat dikompilasi dengan sangat mudah dan cepat ke hampir semua jenis perangkat nirkabel, terutama perangkat iOS dan Android. Selain itu, Amazon SageMaker memiliki keunggulan berikut:

- Layanan ini terkelola sepenuhnya, sehingga tidak perlu khawatir tentang keamanan atau penerapan patch atau pemutakhiran perangkat lunak apa pun;
- Beberapa alat AI yang paling umum digunakan secara otomatis disertakan dengan Amazon SageMaker, dan alat-alat ini telah dioptimalkan secara maksimal sehingga semua jenis aplikasi yang Anda buat akan berjalan sepuluh kali lebih cepat daripada jenis penerapan AI lainnya. Anda bahkan dapat menerapkan algoritma AI kustom Anda sendiri ke dalam Amazon SageMaker;
- Amazon SageMaker menyediakan tingkat optimasi yang tepat untuk semua jenis beban kerja yang dibutuhkan aplikasi AI Anda. Dalam hal ini, Anda dapat menggunakan mesin

virtual "ml.t2.medium" versi yang lebih rendah, atau mesin virtual "ml.p3dn.24xlarge" yang sangat canggih.

Selain itu, Amazon SageMaker memungkinkan ilmuwan data dan tim pengembangan perangkat lunak mana pun untuk beroperasi dengan lancar dan cepat dengan layanan AWS lainnya yang mencakup hal-hal berikut:

Dari Sudut Pandang Persiapan Data

- S3;
- RDS;
- DynamoDB;
- Lambda.

Dari Sudut Pandang Pemilihan, Optimasi, dan Pelatihan Algoritma

Sebagaimana telah disebutkan, Amazon SageMaker memiliki sejumlah algoritma matematika yang sangat canggih, sangat cepat, dan sangat akurat. Algoritma-algoritma ini dapat menangani dataset berukuran petabyte, dan bahkan meningkatkan kinerja hingga sepuluh kali lipat dibandingkan algoritma matematika AI tradisional lainnya. Berikut adalah contoh algoritma AI yang tersedia saat ini terkait dengan Amazon SageMaker:

- Teks yang Membara;
- Peramalan DeepAR;
- Mesin Faktorisasi;
- K-Means;
- Hutan Potongan Acak;
- Deteksi Objek;
- Klasifikasi Citra;
- Model Topik Neural (NTM);
- Wawasan IP;
- Tetangga K-Terdekat (alias "k-NN");
- Alokasi Dirichlet Laten;
- Pembelajar Linier;
- Object2Vec;
- Analisis Komponen Utama;
- Segmentasi Semantik;
- Urutan ke Urutan;
- XGBoost.

Dari Sudut Pandang Algoritma Matematika AI dan Optimalisasi

Amazon SageMaker juga dilengkapi dengan Penyetelan Model AI otomatis, yang dalam istilah teknis dikenal sebagai "Penyetelan Hiperparameter". Dengan proses ini, pola statistik terbaik untuk aplikasi AI spesifik Anda dijalankan melalui serangkaian iterasi matematika yang memanfaatkan kumpulan data yang akan digunakan aplikasi AI Anda. Dalam hal metrik pelatihan, "kartu skor" juga disimpan untuk algoritma AI yang dianggap berjalan paling baik, sehingga Anda dapat melihat algoritma mana yang paling efektif untuk aplikasi AI Anda.

Untuk mengilustrasikan hal ini lebih lanjut, bayangkan Anda mencoba mengimplementasikan aplikasi jenis Klasifikasi Biner. Dalam istilah matematika, di semua tingkatan yang memungkinkan, Anda ingin memaksimalkan apa yang dikenal sebagai "Area di Bawah Kurva", atau disingkat "AUC". Ini akan dilakukan dengan melatih model matematika khusus yang dikenal sebagai "XGBoost". Berikut adalah ketentuan yang akan digunakan:

- Alfa;
- ETA;
- Min_Child_Weight;
- Max_Depth.

Dari sini, Anda dapat mengatur rentang permutasi tertentu untuk "Penyetelan Hiperparameter".

Dari Sudut Pandang Penerapan Algoritma

Dari perspektif tim pengembangan perangkat lunak dan ilmuwan data, penerapan model berbasis AI sebenarnya merupakan pendekatan dua tahap yang sangat mudah, yaitu sebagai berikut:

1) Anda perlu mengonfigurasi titik akhir spesifik aplikasi AI berbasis Cloud Anda terlebih dahulu agar beberapa instans dapat digunakan dalam Mesin Virtual (VM) yang sama;
2) Dari sini, Anda dapat meluncurkan lebih banyak instans berbasis AI dari aplikasi Anda agar berbagai prediksi dapat dibuat mengenai keluaran yang diinginkan. Penting juga untuk dicatat bahwa API Amazon SageMaker juga dapat bekerja dengan lancar dengan jenis instans AI lainnya, dan karena itu, Anda dapat membuat aplikasi AI Anda lebih tangguh.

Selain itu, Amazon SageMaker dapat bekerja dengan jenis prediksi yang dianggap "batch" dan "one-off". Untuk prediksi batch, prediksi semacam ini dapat dibuat pada kumpulan data yang terdapat dan tersimpan di Amazon S3.

Dari Sudut Pandang Integrasi dan Pemanggilan

Amazon SageMaker menyediakan berbagai jenis alat berikut:

1) <u>API Berbasis Web:</u>
   Jenis API khusus ini dapat digunakan untuk mengontrol lebih lanjut dan secara harfiah "memanggil" instans Server Virtual dari Amazon SageMaker.
2) <u>API SageMaker:</u>
   Jenis API khusus ini dapat menggunakan bahasa kode sumber berikut:
   - Go;
   - C++;
   - Java;
   - Java Script;
   - Python;
   - PHP;
   - Ruby;
   - Ruby On Rails.
3) <u>Antarmuka Web:</u>

Ini adalah antarmuka langsung ke Jupyter Notebook.

4) AWS CLI:

Ini adalah Antarmuka Baris Perintah (CLI) untuk AWS.

Amazon Comprehend

Salah satu komponen kunci dari setiap aplikasi Kecerdasan Buatan adalah Pemrosesan Bahasa Alami, yang juga dikenal sebagai "NLP". Pemrosesan bahasa alami (NLP) dapat didefinisikan secara spesifik sebagai berikut: *Pemrosesan bahasa alami (NLP) adalah cabang kecerdasan buatan yang membantu komputer memahami, menafsirkan, dan memanipulasi bahasa manusia. NLP memanfaatkan berbagai disiplin ilmu, termasuk ilmu komputer dan linguistik komputasional, dalam upayanya untuk menjembatani kesenjangan antara komunikasi manusia dan pemahaman komputer.*

Dalam hal ini, AWS memudahkan Anda untuk mengimplementasikan Pemrosesan Bahasa Alami untuk aplikasi AI, terutama dalam hal bahasa manusia, dan dari sana, AWS dapat memastikan segala jenis konten implisit maupun eksplisit dalam bahasa manusia yang digunakan. Dalam hal ini, NLP juga dapat dianggap sebagai "Big Data", tetapi pada tingkat kualitatif. Misalnya, ini dapat mencakup email dukungan pelanggan, segala bentuk umpan balik yang diberikan oleh pelanggan, khususnya dalam hal ulasan produk/layanan, segala jenis percakapan di pusat panggilan, serta semua yang terjadi di berbagai situs media sosial, khususnya Facebook, LinkedIn, dan Twitter.

Nama alat Pemrosesan Bahasa Alami (NLP) yang digunakan oleh AWS disebut "Amazon Comprehend". Alat ini memiliki fungsi-fungsi berikut:

1) Menganalisis Kasus Penggunaan:

Alat ini dapat dengan cepat dan mudah memindai hampir semua jenis dokumen, untuk menemukan korelasi statistik atau pola tersembunyi yang terdapat di dalamnya. Ini mencakup hal-hal seperti Analisis Sentimen, Ekstraksi Entitas, dan bahkan pengorganisasian Dokumen, tergantung pada jenis kategori spesifiknya.

2) Akses Konsol:

Amazon Comprehend dapat diakses dengan sangat cepat dan mudah dari dalam Konsol Manajemen AWS. Jika Anda memiliki data kuantitatif dalam jumlah besar yang tersimpan di S3, Anda dapat dengan mudah mengintegrasikannya dengan Amazon Comprehend. Dari sini, Anda dapat menggunakan API khusus untuk menemukan korelasi atau tren tersembunyi yang tidak terlihat pada awalnya. Keuntungan utama di sini adalah Anda bahkan dapat mengumpulkan berbagai set data dari S3 agar dapat diproses lebih lanjut oleh Amazon Comprehend. Amazon Comprehend juga memiliki enam API berbeda yang dapat Anda gunakan, yaitu sebagai berikut:

- API Ekstraksi Frasa Kunci: API ini dapat digunakan untuk mengidentifikasi frasa dan/atau istilah tertentu dari dalam kumpulan data kualitatif yang disediakan;
- API Analisis Sentimen: API ini akan menghitung tingkat perasaan keseluruhan teks yang diketik dan/atau dimasukkan oleh individu dan memeringkatnya sebagai positif, negatif, atau netral.

- API Sintaksis: API ini memungkinkan Anda membedakan kata-kata yang diucapkan, seperti kata benda, kata kerja, kata sifat, kata ganti, dan lain-lain.
- API Pengenalan Entitas: API ini dapat digunakan untuk mengidentifikasi lebih lanjut entitas aktual dalam teks, seperti entitas tempat, orang, dan lain-lain.
- API Deteksi Bahasa: API ini dapat digunakan untuk mengidentifikasi secara spesifik bahasa yang digunakan dalam penyampaian teks.
- API Klasifikasi Kustom: Dengan API canggih ini, Anda bahkan dapat membuat dan menerapkan model klasifikasi khusus untuk aplikasi AI Anda.

Amazon Rekognition

Amazon Rekognition adalah alat di AWS yang dirancang khusus untuk memproses segala jenis gambar dan/atau video yang mungkin Anda gunakan untuk aplikasi AI Anda. Alat ini sangat canggih karena telah dilatih sebelumnya dengan miliaran gambar yang dapat dikenali dengan mudah. Meskipun terdengar sangat rumit, alat ini cukup mudah digunakan karena memanfaatkan algoritma matematika Pembelajaran Mendalam yang sudah tersimpan di AWS hanya melalui satu API.

Berikut ini hanyalah contoh bagaimana alat ini dapat digunakan untuk aplikasi AI:

- Deteksi Objek dan Pemandangan;
- Pengenalan Gender;
- Pengenalan Wajah: Di sinilah identitas individu tertentu dikonfirmasi oleh fitur unik yang terdapat pada wajah mereka. Saat digunakan dengan AWS, alat ini memanfaatkan teknik dan algoritma Pembelajaran Mendalam.

Amazon Translate

Sesuai namanya, alat di AWS ini benar-benar dapat menerjemahkan segala bentuk teks tertulis dengan cepat dan mudah ke bahasa lain. Bahasa asing yang didukung Amazon Translate ditunjukkan dalam matriks berikut, beserta kode spesifiknya yang digunakan untuk mengidentifikasi AWS dan Amazon Translate di dalamnya:

Bahasa	***Kode Bahasa AWS***
Arab	ar
China (sederhana)	zh
China (tradisional)	zh-TW
Ceko	cs
Denmark	da
Belanda	nl
Inggris	en
Finlandia	fi
Prancis	fr
Jerman	de
Yunani	el
Ibrani	he
Hindi	hi
Hungaria	hu

Indonesia	id
Italia	it
Jepang	ja
Korea	ko
Melayu	ms
Norwegia	no
Persia	fa
Polandia	pl
Portugis	pt
Rumania	ro
Rusia	ru
Spanyol	es
Swedia	sv
Thai	th
Turki	tr
Ukraina	uk

Amazon Translate dapat diakses melalui tiga metode berbeda:

- Dari AWS Management Console;
- Menggunakan API AWS yang dirancang khusus;
  Bahasa kode sumber yang didukung meliputi:

  *Go;
  *C++;
  *Java;
  *Java Script;
  *Python;
  *PHP;
  *Ruby;
  *Ruby On Rails.

- Dari AWS CLI.

Amazon Transcribe

Alat di AWS ini memanfaatkan apa yang dikenal sebagai "Pengenalan Ucapan Otomatis", atau disingkat "ASR". Dengan ini, tim pengembangan perangkat lunak Anda dapat dengan mudah dan cepat mengintegrasikan fungsi ucapan ke teks ke aplikasi AI Anda. Alat ini juga dapat menganalisis dan mentranskripsi berkas audio MP3, yang juga dapat digunakan secara waktu nyata (real-time). Misalnya, alat ini dapat mengambil aliran audio langsung dan menyediakan teks secara waktu nyata (real-time). Alat ini bahkan dapat memberikan stempel waktu untuk setiap kata yang telah ditranskripsi.

Amazon Textract

Salah satu kendala tersulit untuk semua jenis aplikasi AI adalah mengenali tulisan tangan spesifik dari individu tertentu. Dengan kata lain, alat ini dapat mengambil tulisan tangan yang tidak jelas, mengubahnya menjadi gambar, dan kemudian mengekstraknya ke dalam format berbasis teks. Amazon Textract bahkan dapat memastikan tata letak setiap bentuk dokumen dan elemen-elemen yang terkait dengannya. Alat ini bahkan dapat mengekstrak data yang ada dalam formulir dan/atau tabel yang disematkan.

Microsoft Azure

Di dunia Microsoft Azure, inilah "Azure Machine Learning Studio" yang terdiri dari semua alat yang pernah Anda impikan untuk membuat dan membangun aplikasi Kecerdasan Buatan (AI). Studio ini menggunakan pendekatan berbasis GUI untuk melakukan hal ini, dan bahkan dapat terintegrasi dengan alat Microsoft lainnya, terutama Power BI.

Ruang Kerja Interaktif Azure Machine Learning Studio

Sesuai namanya, ini adalah semacam ruang kerja interaktif tempat Anda dapat memasukkan kumpulan data raksasa ke dalam aplikasi AI Anda, memanipulasinya, lalu menyelesaikan analisis mendalam dengan berbagai fungsi dan rumus statistik yang sangat canggih, dan bahkan mendapatkan gambaran sekilas tentang seperti apa keluaran dari sistem AI yang baru saja Anda bangun. Seluruh proses ini secara teknis juga disebut sebagai "Machine Learning Pipeline". Keuntungan utamanya adalah semua yang ada dalam proses ini ditampilkan secara visual.

Perlu dicatat bahwa proses di atas dapat diulang berkali-kali sebagai apa yang disebut "Eksperimen Pelatihan" hingga hasil yang Anda inginkan tercapai. Setelah selesai, latihan ini dapat dikonversi ke lingkungan produksi, yang dikenal sebagai "Eksperimen Prediktif".

Machine Learning Studio terdiri dari fungsi-fungsi berikut:

- Proyek:
  Ini adalah kumpulan Eksperimen Pelatihan dan Eksperimen Prediktif.
- Eksperimen:
  Di sinilah eksperimen spesifik benar-benar dibuat, direvisi, diluncurkan, dan dieksekusi.
- Layanan Web:
  Eksperimen berbasis produksi Anda juga dapat dikonversi ke layanan berbasis web tertentu.
- Buku Catatan:
  Machine Learning Studio juga mendukung Jaringan Jupyter, yang merupakan layanan eksklusif dari AWS.
- Set Data:
  Di sinilah Anda mengunggah dan menyimpan set data masing-masing yang akan dimasukkan ke dalam aplikasi AI Anda.
- Model Terlatih:
  Ini adalah model AI spesifik yang telah Anda buat dan dengan demikian telah dilatih dalam Eksperimen Pelatihan atau Eksperimen Prediktif.

Perlu dicatat bahwa ada beberapa kondisi yang harus dipenuhi terlebih dahulu sebelum Anda dapat mulai membuat dan meluncurkan model dan aplikasi AI. Kondisi-kondisi tersebut adalah sebagai berikut:

- Anda harus memiliki setidaknya satu set data dan satu modul yang sudah ada;
- Set data yang Anda rencanakan untuk dimasukkan ke dalam model/aplikasi AI Anda hanya dapat dihubungkan ke modulnya masing-masing;
- Modul dapat dihubungkan dengan cepat dan mudah ke model lain;
- Harus ada setidaknya satu koneksi ke set data yang Anda rencanakan untuk dimasukkan ke dalam model/aplikasi AI;
- Anda harus telah menetapkan permutasi yang diperlukan sebelum Anda dapat memulai pekerjaan apa pun.

Perlu dicatat bahwa modul hanyalah sebuah algoritma yang dapat digunakan untuk menganalisis lebih lanjut kumpulan data Anda. Beberapa modul yang sudah termasuk dalam Machine Learning Studio antara lain:

1) Modul Konversi ARFF:
   Modul ini mengonversi kumpulan data .NET menjadi Format Berkas Hubungan Atribut (alias "ARFF").
2) Modul Komputasi Statistik Dasar:
   Modul ini menghitung statistik dasar, seperti R^2, R^2 yang Disesuaikan, Rata-rata, Modus, Median, Simpangan Baku, dan lain-lain.
3) Berbagai Model Regresi Berganda:
   Anda memiliki beragam model statistik yang dapat Anda pilih, tanpa perlu membuat model baru.
4) Model Penilaian:
   Modul ini dapat menilai Model Regresi Berganda yang akan Anda gunakan untuk aplikasi AI Anda secara kuantitatif.

Layanan Azure Machine Learning

Ini adalah platform besar Azure lainnya yang memungkinkan aplikasi AI Anda menjadi jauh lebih skalabel. Platform ini mendukung kode sumber Python, yang merupakan bahasa pemrograman pilihan untuk sebagian besar aplikasi AI pada umumnya. Layanan ini juga menggunakan Docker Container. Layanan ini dapat diakses melalui dua cara berbeda, yaitu:

- Software Development Kit (SDK);
- Jenis antarmuka berbasis visual lainnya, terutama Microsoft Visual Studio.

Perbedaan utama antara Azure Machine Learning Services dan Azure Machine Learning Studio diuraikan dalam matriks berikut:

Layanan Pembelajaran Mesin Azure	*Azure Machine Learning Studio*
Mendukung Lingkungan Hibrida Cloud dan Lokal	Hanya eksperimen standar yang dapat dibuat, diluncurkan, dan dijalankan.
Anda dapat menggunakan berbagai kerangka kerja dan instans Mesin Virtual	Ini sepenuhnya dikelola oleh Azure.

Mendukung Penyetelan Hiperparameter Otomatis	Ini hanya tersedia di Cloud, bukan sebagai solusi Lokal.

Layanan Kognitif Azure

Layanan khusus ini memiliki komponen-komponen berikut:

1) Layanan Keputusan:
   Saat Anda menerapkan berbagai aplikasi AI, sistem ini akan memberikan rekomendasi tertentu agar Anda dapat membuat keputusan yang lebih baik tentang cara meningkatkan efisiensi dan optimalisasi aplikasi AI Anda.
2) Layanan Visi:
   Layanan ini dapat mengaktifkan aplikasi AI Anda secara otomatis sehingga dapat menganalisis dan memanipulasi gambar dan video.
3) Layanan Pencarian:
   Anda dapat mengintegrasikan Mesin Pencari Bing ke dalam aplikasi AI Anda.
4) Layanan Ucapan:
   Layanan ini dapat mengonversi kata-kata yang diucapkan menjadi format teks. Layanan ini juga sepenuhnya mendukung modalitas Biometrik Pengenalan Ucapan.
5) Layanan Bahasa:
   Ini adalah komponen Pemrosesan Bahasa Alami (NLP) Azure, dan dapat dengan cepat dan mudah menganalisis sentimen apa pun yang telah dikomunikasikan, terutama yang digunakan dalam chatbot.

5.4 GOOGLE CLOUD PLATFORM

Dibandingkan dengan AWS dan Azure, Google Cloud Platform berada di posisi ketiga yang cukup jauh. Komponen terbesar GCP adalah apa yang dikenal sebagai "AI Hub". Ini adalah antarmuka besar yang terdiri dari komponen plug-and-play, algoritma AI canggih, fitur kolaborasi instan, serta kemampuan untuk mengimpor sejumlah besar dataset yang telah disimpan di Penyedia Cloud lainnya. Berikut adalah beberapa fitur utama AI Hub:

1) Penemuan Komponen dan Kode:
   Melalui ini, Anda dapat mengakses komponen-komponen berikut:
   - Google AI;
   - Google Cloud AI;
   - Google Cloud Partners.
2) Kolaborasi:
   Komponen ini membantu menghindari duplikasi, terutama jika Anda sedang membangun proyek AI skala besar sebagai bagian dari upaya tim yang besar. Komponen ini memiliki jenis kontrol yang sangat terperinci, dan bahkan dilengkapi dengan serangkaian algoritma AI yang dapat Anda gunakan langsung.
3) Penerapan:
   Fungsi khusus ini memungkinkan modifikasi dan kustomisasi penuh algoritma AI yang Anda rencanakan untuk digunakan atau sedang dalam proses penggunaan untuk

aplikasi AI Anda. Setelah Anda membangun aplikasi Anda, Anda bahkan dapat menghostingnya di platform Penyedia Cloud lainnya juga.

Komponen Dasar AI Google Cloud

Google Cloud Platform (GCP) juga dilengkapi dengan banyak alat lain, yaitu:

1) Model Kustom Google Cloud AutoML:
   AutoML memanfaatkan Arsitektur Jaringan Neural dan Pembelajaran yang sangat canggih sehingga Anda dapat membuat aplikasi AI yang sangat spesifik dalam subdomain Kecerdasan Buatan tertentu.
2) API Pra-Terlatih Google Cloud:
   Dengan ini, Anda benar-benar dapat menggunakan API yang dilatih secara khusus tanpa harus terlebih dahulu mempelajari seluruh proses pelatihan aplikasi AI Anda. Keunggulannya adalah API spesifik ini terus ditingkatkan agar tetap optimal dan disempurnakan untuk tingkat pemrosesan dan kecepatan yang tinggi.
3) Vision AI dan AutoML Vision:
   Dengan layanan semacam ini, Anda dapat memperoleh wawasan tepat waktu dari model AutoML Vision atau Vision API, yang semuanya telah dilatih sebelumnya. Layanan ini bahkan dapat digunakan untuk mendeteksi emosi seseorang, terutama jika Anda menggunakan aplikasi AI untuk alat chatbot yang canggih. Lebih lanjut, dengan Google Vision API, Anda bahkan dapat menggunakan panggilan "RESTful" dan "RPC API". Dengan masing-masing API ini, Anda dapat dengan cepat dan mudah mengklasifikasikan segala jenis gambar yang Anda unggah ke aplikasi AI Anda. Ini sebenarnya adalah layanan yang telah dilatih sebelumnya, dan terdiri dari lebih dari satu juta jenis kategori. Layanan ini dapat digunakan untuk mengubah ucapan menjadi teks, dan untuk menggabungkan teknologi Pengenalan Wajah ke dalam sistem AI Anda.
4) AutoML Intelligence dan Video Intelligence API:
   Ini adalah layanan yang dapat Anda gunakan untuk melacak dan mengklasifikasikan objek dalam video, menggunakan berbagai jenis model AI. Anda juga dapat menggunakan layanan ini untuk melacak objek dalam video streaming.
5) AutoML Natural Language dan Natural Language API:
   Melalui API yang mudah digunakan, Anda dapat menentukan semua jenis "sentimen", yang meliputi:
   - Analisis Entitas;
     Analisis Sentimen;
   - Klasifikasi Konten;
   - Analisis Sentimen Entitas;
   - Analisis Sintaksis.

   Anda bahkan dapat memasukkan kumpulan data ke dalamnya untuk menentukan mana yang paling sesuai untuk aplikasi AI Anda.
6) Dialogflow:
   Ini sebenarnya adalah layanan pengembangan perangkat lunak, di mana tim pengembangan perangkat lunak dapat membuat agen yang dapat terlibat dalam

percakapan dengan orang sungguhan, seperti, sekali lagi, chatbot. Setelah ini selesai, Anda dapat langsung meluncurkan chatbot Anda di berbagai platform berikut:

- Asisten Google;
- Facebook Messenger;
- Slack;
- Layanan Suara Alexa.

7) Teks ke Ucapan:

Dengan ini, Anda dapat dengan cepat dan mudah mengonversi ucapan manusia apa pun ke lebih dari 30 bahasa asing dan dialeknya. Untuk melakukannya, alat ini menggunakan alat Sintesis Ucapan yang disebut "WaveNet" untuk menghasilkan berkas audio MP3 berkualitas perusahaan.

8) Ucapan ke Teks:

Ini kebalikan dari yang di atas. Dengan ini, Anda juga dapat dengan cepat dan mudah mengonversi berkas audio menjadi teks menggunakan algoritma Jaringan Neural yang sudah terintegrasi di Google Cloud Platform. Meskipun algoritma ini cukup kompleks, algoritma ini dapat dipanggil dengan mudah dan cepat melalui penggunaan API khusus. Lebih dari 120 bahasa dan dialeknya didukung. Ucapan ke Teks dapat digunakan untuk tujuan berikut:

- Dapat mengaktifkan segala jenis perintah suara di aplikasi apa pun;
- Dapat mentranskripsikan percakapan pusat panggilan;
- Dapat dengan mudah digabungkan dengan layanan non-Google lain yang terkait dengan AI;
- Dapat memproses audio secara real-time dan mengonversi ucapan menjadi teks dari percakapan yang telah direkam sebelumnya.

9) Tabel AutoML:

Dengan fungsionalitas ini, Anda dapat menerapkan model AI pada set data yang sepenuhnya terstruktur. Meskipun tidak memerlukan pengkodean khusus, jika perlu, Anda dapat menggunakan "Colab Notebooks". Cara kerjanya sangat mirip dengan Jupyter di AWS.

10) AI Rekomendasi:

Ini adalah layanan unik yang dapat memberikan semua jenis rekomendasi produk untuk aplikasi AI yang berhubungan dengan pelanggan, sekali lagi, seperti halnya chatbot.

Kami telah meninjau apa yang ditawarkan oleh Penyedia Cloud utama, Amazon Web Services, Microsoft Azure, dan Google dalam hal layanan terkait AI. Kami sekarang akan membahas beberapa aplikasi AI yang dapat Anda bangun, dengan memanfaatkan kode sumber Python.

Membangun Aplikasi yang Dapat Membuat Berbagai Kelas Pendapatan

Dalam contoh ini, kita akan melihat cara menggunakan sumber Python untuk membuat aplikasi yang dapat membuat berbagai klasifikasi untuk berbagai tingkat pendapatan bagi

seluruh populasi. Hal ini dapat bekerja dengan sangat baik untuk Pemerintah Federal, terutama dalam hal perpajakan dan/atau pemberian hak dan tunjangan.

Contoh ini menggunakan dataset yang terdiri dari populasi 25.000 orang:

```
# Input file containing
Data input_file = 'income_data.txt'
# Read the data
X=[]
Y = []
Count_Class1 = 0
Count_Class2 = 0
Max_datapoints = 25000
With open (input_file, 'r') as f:
     For line in f. readiness ():
          If count_Class >=max_datapoints and Count_Class2 >=max_datapoints
     Break
If '?' in line:
     Continue
     Data = line [:-1].split (',')
     If data [-1] ==”<=50K” and Count_Class1 < max_datapoints;
           X.append(data)
           Count_Class1 ==1
     If data [-1] ====”>50K” and Count_Class2 < max_datapoints;
           X.append(data)
           Count_Class2 +-1
     # Convert to numpy array
     X = np.array
     #Convert string data to numerical data
     Label_encoder = []
     X_encoded = np.empty (X.shape)
     For I, item in enumerate (X[0]);
          If item.isdigit ();
                X_Encoded [;, i] = X [:, 1]
          Else:
                Label_encoder.append(preprocessing.LabelEncoder ())
                X_Encoded [:, i] = label_enocder [-1].fit.tranform (X[:, 1])
          X = X_encoded [:, :-1], astype (int)
          Y = X_encoded [:, :-1], astype (int)
#Create SVM classifier
Classifier = OneVaOneClassifier (LinearSVC (random_state=0);
#Train the Classifier
Classifier.fit (X, y)
#Cross Validation
X_train, X_test, y_train, y_test = train_test_split.train_test_
Split (X, y, test_size=0.2, random_state=5)
Classifier = OneVaOneClassifier (LinearSVC (random_state=0);
Classifier.fit (X_train, y_train)
Y_test_pred = classifier.predict (X_test)
#Compute the F1 score of the SVM Classifier
F1 = train_test_split.cross_val_score (classifier, X, y, scoring='f1_weighted', cv=3)
Print ("F1 score: + str(round(100*f1.mean(), 2)} + "%"
#Predict output for a test datapoint
Input_data = ['32', 'Public' or 'Private', '34456', 'College Graduate', 'Married', 'Physician' 'Has
   Family', 'Caucasian', 'Female', 23', 'United States']
#Encode test datapoint
Input_Data_Encoded = [-1] * len(input_data)
Count = 0
For I, item in enumerate (input_data);
     If item.isdigit ():
        Input_data_encoded [i] = int (input_data [i])
    Else:
         Input_data_encoded[i] = int (label_encoder[count].
```

```
    Transform(input_data[i]))
    Input_data_encoded = np.array(input_data_encoded)
#Run classifier on encoded datapoint and print output
Predicted_class = classifier.predict (input_data_encoded)
Print (label_encoder [-1].inverse_transform (predicted_class) [0])
```

Membangun Aplikasi yang Dapat Memprediksi Harga Rumah

Di masa ekonomi yang baik, salah satu pasar yang cenderung sangat menarik perhatian dan menjadi "panas" adalah pasar properti. Hal ini terutama berlaku jika Anda ingin "membalik" rumah dengan nilai yang lebih tinggi, atau hanya ingin menjual rumah Anda yang sudah ada. Aplikasi ini bahkan dapat digunakan untuk memprediksi nilai pasar rumah yang ingin Anda beli. Hal yang sebaliknya juga berlaku. Model yang dikembangkan di sini juga dapat digunakan dengan model berbasis keuangan lainnya jika terjadi penurunan ekonomi, di mana harga properti dapat berfluktuasi secara signifikan.

Berikut adalah kode sumber Python untuk membuat aplikasi semacam ini:

```
#Load housing data
Data = datasets.load_boston()
#Shuffle the data
X, y = Shuffle(data.data, data.target, random_state=32)
#Split the data into training and testing datasets
Num_training = int (0.8 * len (X))
X_trai, Y_train = X[:num_training], y[:num_training]
X_test, Y_test = X(num_training;), y[num_training:]
#CrSv_regressor = SVR(kernel = ‘linear’, C=1.0 epsilon=0.1)
#Train Support Vector Regressor
Sv_regressor.fit (X_train, Y_train)
#Evaluate performance of Support Vector Regressor
Y_test_pred = sv_regressor.predict (X_test)
MSE=mean_squared_error (y_test, y_test_pred)
EVS = explained variance score (y_test, y_test_pred)
Print (“\n### Performance ###”)
Print (“Mean squared error =”, round (mse, 2))
Print (“Explained variance score =”, round (evs, 2))
#Test the regressor on test datapoint
Test_data = (Iterations of housing pricing datasets)
Print (“\nPredicted Proce:”, sv_regressor.predict ([test_data]) [0])eate Support Vector
Regression model
```

Membangun Aplikasi yang Dapat Memprediksi Pola Lalu Lintas Kendaraan di Kota-Kota Besar

Meskipun banyak orang bekerja dari rumah karena pandemi COVID-19, kemacetan lalu lintas masih terjadi. Kemacetan lalu lintas mungkin tidak terlalu parah di daerah pedesaan, tetapi di wilayah metropolitan yang jauh lebih besar, kemacetan lalu lintas masih sangat tinggi. Mengingat hal ini dan fakta bahwa hampir semua hal dapat mengganggu kelancaran arus lalu lintas kendaraan, baik karena cuaca, kecelakaan besar, atau bahkan serangan siber, pejabat pemerintah perlu memiliki cara untuk memprediksi kondisi lalu lintas berdasarkan hasil tertentu, misalnya dengan menggunakan permutasi yang baru saja dijelaskan. Selain itu, pengemudi kendaraan perlu terus mendapatkan informasi terbaru melalui aplikasi seluler

mereka (terutama Google Maps) jika ada rute baru yang harus diambil, jika terjadi kemacetan lalu lintas skala besar.

Berikut adalah kode sumber Python untuk membantu pembuatan aplikasi semacam itu:

```
#Load input data
Input_file = 'traffic data.txt'
Data = []
With open (input_file, 'r') as f:
     For line in f.readiness ();
          Items = line [:-1], split (',')
          Data.append (items)
Data=np.array(data)
#Convert string data to numerical data
Label_Encoder = []
X_encoded = np.empty (data.shape)
For I, item in enumerate (data[0]):
     If item.isdigit ():
     X_encoded (:, i) = data [:, i]
     Else:
          Label_encoder.append (preprocessing.LabelEncoder(;)
          X_encoded [;, i] = label_encoder [-1].fit_transform(data[I, 1])
     X = X_encoded [:, :-1].astype(int)
     Y = X_encoded [:, -1].astype(int)
#Split data into training and testing datasets
X_train, X_test, y_train, y_test = train_test_split(
     X, y, test_size=0.25, random_state=5)
#Extremely Random Forests Regressor
Params = {'n_estimators': 100, 'max_depth': 4, 'random_state':0)
Regressor = ExtraTreesRegressor (**params)
Regressor.fit (X_train, y_train)
#Compute the regressor performance on test data
Y_pred = regressor.predict (X_test)
Print ("Mean absolute error:", round (mean_absolute_error (y_test, y_pred), 2))
#Testing encoding on single data instance
Test_datapoint = ['Friday', '6 PM CST', 'Chicago', 'no']
Test_datapoint_encoded = [-1] * len(test_datapoint)
#Predict the output for the test datapoint
Print ('Predicted Traffic:", int (regressor.predict ([test_datapoint_encoded]) [0]))
```

Membangun Aplikasi yang Dapat Memprediksi Pola Pembelian E-Commerce

Karena pandemi COVID-19 diperkirakan akan berlangsung cukup lama, banyak konsumen kini memilih untuk berbelanja online langsung dari perangkat nirkabel mereka untuk mendapatkan produk dan layanan yang mereka butuhkan, daripada mengunjungi toko fisik untuk aktivitas semacam ini. Oleh karena itu, akan menjadi sangat penting bagi para pedagang E-Commerce untuk memiliki aplikasi yang dapat membantu memprediksi pola pembelian secara real-time, dan bahkan memperkirakan seperti apa pola pembelian di masa mendatang, sehingga mereka dapat menyimpan tingkat inventaris yang dibutuhkan dengan tepat. Berikut adalah kode sumber Python untuk membantu membuat aplikasi tersebut:

```
#Load data from input file
Input_file = 'sales.csv"
File_reader = csv.reader (open(inout_file, 'r'), delimeters=', '
X = []
For count, row in enumerate (file_reader):
```

```
If not count:
     Names = row[1:]
     Cont
X.append ([float(x) for x in row [1:]])
#Convert to numpy array
X= np.array(X)
#Estimating the bandwidth of input data
Bandwidth = estimate_bandwidth (X, quantile=0.8, n_samples=len(x))
#Compute clustering with MeanShift
Meanshift_model = Meanshift (bandwidth=bandwidth, bin_seeding=True)
Meanshift_model.fit (X)
Labels = meanshift_model.labels_
Cluster_centers = meanshift_model.cluster_centers_
Num_clusters = len (np.unique(labels))
Print ("\nNumber of clusters in input data =" num_clusters)
Print ("\nCenters of clusters:"
Print ('\t.join([name[:3] for in names]))
For cluster_center in cluster_centers:
     Print('\t'.join([str(int(X)} for X in cluster_center]))
#Extract two features for visualization
Cluster_centers_2d = cluster_centers[:, 1:3]
#Plot the cluster centers
Plt.figure()
Plt.scatter (cluster_centers_2d{:, 0], cluster_centers_2d[:1,1],
     S=120, edgecolors='blue', facecolors='none']
Offset=0.25
Plt.xlim (cluster_centers_2d[:, 0].max() + offset * cluster_
Centers_2d[:, 0].ptp,
     Cluster_centers_2d[:,0], max() + offset *cluster
Centers_2d[:, 0].ptp(),
Plt.ylim (cluster_centers_2d[:, 1].max() + offset * cluster_
Centers_2d[:, 1].ptp(),
     Cluster_centers_2d[:,1], max() + offset *cluster_
Centers_2d[:, 1].ptp())
Plt.title ('Centersof 2D Clusters')
Plt.show()
```

Membangun Aplikasi yang Dapat Merekomendasikan Pilihan Film Terbaik

Sebagaimana telah dijelaskan di seluruh buku ini, penggunaan chatbot mungkin merupakan salah satu aplikasi terbesar, tidak hanya Kecerdasan Buatan, tetapi juga Jaringan Saraf Tiruan. Ide di balik semua ini adalah bahwa percakapan dengan calon pelanggan atau pelanggan haruslah lancar, di mana mereka merasa sedang berinteraksi dengan manusia sungguhan. Salah satu tujuan dasar dari hal ini adalah untuk mencoba memprediksi terlebih dahulu pertanyaan, kekhawatiran, atau permintaan apa yang mungkin muncul berdasarkan percakapan dan interaksi sebelumnya dengan chatbot. Dalam aplikasi ini, kami mengkaji cara menanamkan percakapan semacam itu ketika merekomendasikan film untuk seseorang. Dalam arti tertentu, ini adalah versi primitif dari apa yang juga dapat dilakukan oleh Asisten Pribadi Virtual (VPA) seperti Siri dan Cortana.

Berikut adalah kode sumber Python:

```
Import argparse
Import json
Import numpy as np
From compute_scores import pearson_score
From collaborative_filtering import find_similar_users
Def build_arg_parser ():
```

```
    Parser = argparse.ArgumentParser (description=’Find recommendations
For the given user’)
    Parser.add_argument (‘–user’, dest=’user’, required=True,
        Help=’Input
    Return parser
#Get movie recommendations for the input user
Def get_recommendations (dataset, input_user):
    If input_user no in dataset
          Raise TypeError (‘Cannot find ‘ = input_user + ‘ in the
    Dataset’)
    Overall_scores = {}
    Similarity_scores = {}
    For user in [x for x in dataset if x [= input_user]:
               Similarity_score = pearson_score (dataset, input_user, user)
               If similarity_score <=0:
                     Continue
         Filtered_list = [x for x in dataset[user] if x not in \
               Dataset[input_user] or dataset [input_user] [x] ==0]
         For item in filtered_list:
              Overall_scores.update ({item: dataset[user] [item] *
         Similarity_score})
                    Similarity_scores.update ({item:similarity_score})
         If len (overall_scores) == 0:
                     Return [‘No movie recommendations are possible’}
#Generate movie selection rankings by normalization
Movie_scores = np.array {[(score/similarity_scores(item), item]
   For item, score in overall_scores.items())]}
#Sort in decreasing order
Movie_scores = movie_scores [np.argsort (movie_scores [:, 0]) [::-1]]
#Extract the movie selection recommendations
Movie_recommendations = [movie for_, movie in movie_scores]
Return movie_recommendations
If___name___ = = ____main____’:
     Args = build_arg_parser().parse_args()
     User = args.user
     Ratings_file = ‘ratings.json’
     With open (ratings_file, ‘r’) as f:
          Data = json.loads (f.read())
     Print (“\nMovie recommendations for” + user +“:”)
     Movies = get_recommendations (data,user)
     For I, movie in enumerate (movies):
          For I, movie in enumerate (movies):
```

Membangun Aplikasi Penganalisis Sentimen

Sejauh ini dalam bab ini, salah satu subjek yang telah dibahas adalah apa yang disebut "Analisis Sentimen". Dengan ini, aplikasi AI mencoba mengukur suasana hati pengguna akhir secara harfiah ketika komunikasi apa pun diterima dalam format teks tertulis. Bahkan ketika pesan diucapkan, mengingat tingkat kecanggihan AWS dan Azure, modalitas Biometrik Pengenalan Suara juga dapat digunakan untuk mengukur suasana hati individu tertentu. Konsep semacam ini biasanya diterapkan dalam riset pasar waktu nyata, terutama dalam hal pengujian pemasaran produk atau layanan baru sebelum diluncurkan ke publik. Dalam aplikasi ini, kami menggunakan file ulasan film hipotetis yang diilustrasikan dalam aplikasi sebelumnya.

Berikut adalah kode sumber Python:

```
From nltk.classify import NaiveBayesClassifier
From nltk.classify.util import accuracy as nltk_accuracy
#Extract features from the input list of words
```

```
Def extract_features (words):
    Return dict([word, True) for word in words])
If____name____==’____main____’:
    #Load the data from the corpus
    Fields_pos = movie_reviews.fields (‘pos’)
    Fields_neg = movie_reviews.fields (‘neg’)
#Extract the features form the movie reviews
Features_pos = [(extract_features (movie_reviews.words(
    Fileside=[f])), ‘Positive’) for f in fields_pos]
#Define the train and test split (80% and 20%)
Threshold = 0.8
Num_pos = int (threshold = len (features_pos))
Num_neg = int (threshold = len (features_neg))
#Create training and training datasets
Features_train = features_pos [:num_pos] + features_neg [:num_neg]
Features_test = features_pos [:num_pos] + features_neg [:num_neg]
#Print the number of datapoints that are used
Print (‘\nNumber of training datapoints:’, len (features_train))
Print (‘Number of test datapoints: ‘, len (features_test))
#Train a Naïve Bayes classifier
      Classifier = NaiveBayesClassifier.train (features_train)
Print (‘\nAccuracy of the classifier:’, nltk_accuracy(
Classifer, features_test))
N=15
Print (‘\nTop ‘ + str(N) + most informative words:’)
For I, item in enumerate (classifier.most_informative_features()]:
    Print (str (i+1) + ‘, ‘ + item[0])
    If I =
        Break
    #Test input movie reviews
    Input_reviews = [
         Movie was great’,
         ‘Movie was good’,
         ‘Movie was OK’,
         ‘Movie was bad’,
         ‘Movie was horrible’,
         ‘I would not recommend this movie’,
         ‘I would recommend this movie’,
              ]
    Print(“\nMovie review predictions:”)
    For review in input_reviews:
          Print(“\nReview:”, review)
    #Compute the statistical probabilities
    Probabilities = classifier.prob_classify (extract_
Features (review.split()))
#Pick the maximum value
Predicted_sentiment = probabilities.max ()
#Print outputs
Print (“Predicted sentiment:”, predicted_sentiment)
Print (“Probability:”, round (probabilities.prob (predicted_sentiment),))
```

5.5 PENERAPAN JARINGAN SARAF TIRUAN UNTUK PEMELIHARAAN PREDIKTIF

Mencegah kegagalan dan kecelakaan peralatan sangat penting bagi perusahaan dan pemerintah. Waktu henti yang tidak perlu dapat mengurangi pendapatan dan meningkatkan biaya secara signifikan, yang berdampak negatif pada profitabilitas. Dalam militer dan pertahanan, hal ini tidak hanya mahal, tetapi misi-misi penting dapat terdampak atau dibatalkan. Hal ini juga dapat mengakibatkan cedera atau kematian manusia yang signifikan. Oleh karena itu, memprediksi dan menghindari kegagalan dan kecelakaan ini sangatlah penting. Pemeliharaan prediktif dapat menjadi kunci untuk menghindari kejadian semacam itu.

Model berbasis fisika biasanya digunakan untuk mengidentifikasi kapan suatu mesin atau proses kompleks cenderung menuju kegagalan. Pemodelan fisika yang sepenuhnya akurat dari semua interaksi kompleks antar subsistem saat ini belum memungkinkan. Lebih lanjut, seiring bertambahnya usia aset, menjalani pemeliharaan, dan penggantian komponen, perilaku sistem mulai menyimpang dari model fisika aslinya. Yang dibutuhkan adalah model yang dapat mempelajari bagaimana sistem berubah seiring waktu. Model Pembelajaran Mesin yang menggunakan Jaringan Saraf Tiruan mampu melakukan hal tersebut.

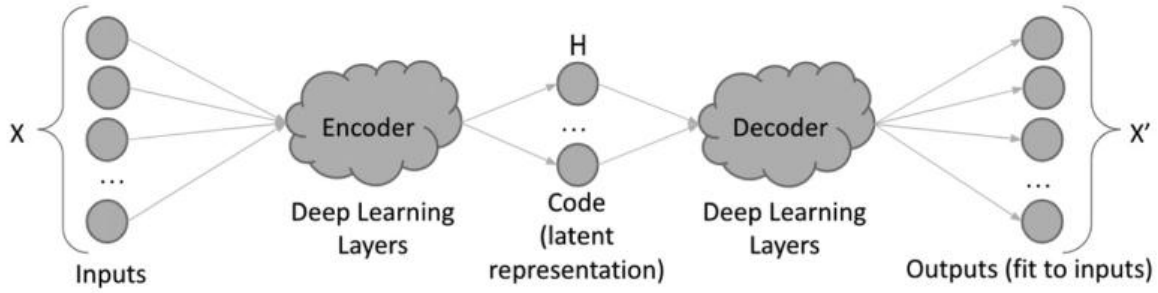
Sebagaimana telah ditekankan sebelumnya, model Pembelajaran Mesin membutuhkan banyak data pelatihan, dan hal ini bahkan lebih berlaku untuk Jaringan Saraf Tiruan. Untungnya, mesin dan proses modern memiliki sejumlah besar sensor yang mengukur suhu, tekanan, getaran, aliran fluida, dll., yang dikumpulkan dan disimpan dalam data historis. Jadi, umumnya tersedia lebih dari cukup data untuk melatih model Jaringan Saraf Tiruan.

Namun, sebagaimana dijelaskan dalam bab-bab sebelumnya, teknik Pembelajaran Mesin menggunakan pembelajaran terawasi yang mengharuskan data pelatihan diberi label dengan hasil yang diharapkan. Dalam hal ini, ini berarti contoh kegagalan peralatan atau proses yang diberi label. Data pelatihan berlabel jenis ini dapat dihasilkan dengan menjalankan sekumpulan aset industri ini hingga mengalami kegagalan dalam semua kemungkinan mode kegagalan. Tentu saja, hal ini tidak praktis mengingat kompleksitas dan biaya sistem industri ini. Lebih lanjut, sistem ini secara inheren sangat andal, yang semakin mempersulit pengumpulan data kegagalan aktual. Dengan demikian, yang tersedia adalah data historis dalam jumlah besar dengan subset mode kegagalan masa lalu yang sangat terbatas. Keterbatasan data pelatihan berlabel ini biasanya membuat teknik Pembelajaran Mesin terawasi menjadi tidak efektif.

Model Perilaku Normal Menggunakan Autoencoder

Salah satu pendekatan untuk mengatasi masalah ini adalah membuat model perilaku normal aset dengan melatih model hanya menggunakan data sensor historis dari semua mode operasi normal aset. Jika suatu aset tidak pernah gagal, ini akan mencakup semua data masa lalu. Data apa pun dari periode kejadian abnormal atau kegagalan perlu dikecualikan. Model yang telah mempelajari operasi normal suatu aset akan dapat menunjukkan kapan ia mulai berperilaku abnormal, yang seringkali merupakan tanda kegagalan yang akan datang atau operasi yang kurang optimal.

Autoencoder Jaringan Saraf Tiruan yang dijelaskan pada halaman xx-yy sangat cocok untuk mempelajari perilaku normal suatu aset dari nilai sensor historisnya. Perlu diingat bahwa autoencoder mencoba menyalin Input-nya ke Output-nya melalui pengkodean terbatas atau lapisan laten yang menciptakan pengkodean yang diinginkan. Diagram autoencoder diulang di bawah ini. Karena autoencoder mempelajari X' dari X, data pelatihan diberi label sendiri. Yang diperlukan hanyalah menghapus data abnormal apa pun dari set pelatihan.



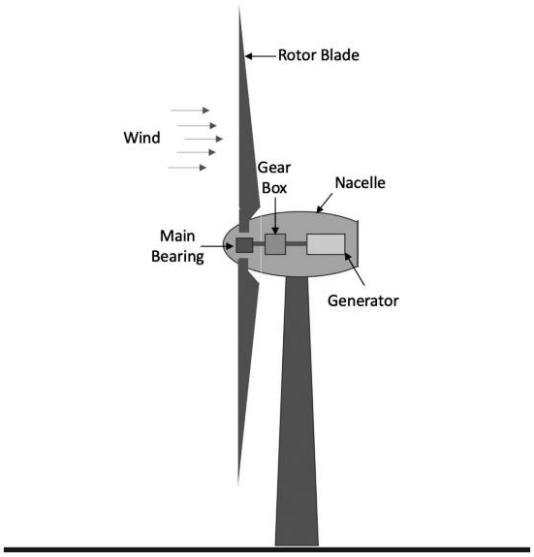
Sensor yang relevan untuk aset tersebut adalah input X. Encoder mempelajari cara mengompresi data input ke dalam status operasi normal yang dikodekan dalam ruang laten H. Decoder juga mempelajari cara mendekode ruang laten H untuk merekonstruksi input sebagai X'. Ruang laten, H, harus sekecil mungkin, tetapi tetap cukup besar untuk merepresentasikan semua status operasi normal yang penting. Analisis statistik data (misalnya, Analisis Komponen Utama atau PCA) dapat menentukan nilai yang tepat untuk H. Model kemudian dilatih untuk meminimalkan perbedaan antara X dan X' untuk semua data pelatihan.

Setelah dilatih, data operasional langsung dapat dimasukkan ke model untuk memprediksi X' baru. Jika kesalahan antara X' yang diprediksi dan X kecil, aset kemungkinan besar beroperasi dalam status operasi normal yang mendekati salah satu status dalam data pelatihan. Seiring meningkatnya kesalahan prediksi ini, kemungkinan aset beroperasi dalam status yang tidak terlihat selama data pelatihan meningkat karena model mengalami kesulitan dalam merekonstruksi data input. Nilai X' dengan kesalahan prediksi terbesar juga memberikan petunjuk penting bagi operator manusia sebagai indikator ketidaknormalan pada kondisi operasi saat ini, yang sangat penting untuk kemudahan penjelasan dan identifikasi tindakan yang perlu diambil untuk memperbaiki ketidaknormalan tersebut.

Contoh Turbin Angin

Turbin angin telah menjadi sumber energi terbarukan yang penting dan dapat dilihat di masa depan di banyak tempat di seluruh dunia. Turbin angin juga memberikan contoh yang relatif mudah untuk penerapan Autoencoder Jaringan Saraf Tiruan (Neural Network Autoencoder) dalam memprediksi peristiwa kegagalan yang tertunda. Ketika Turbin Angin gagal, dibutuhkan waktu berminggu-minggu untuk menjadwalkan derek dan peralatan lain yang diperlukan untuk melakukan perbaikan. Selama waktu tersebut, semua listrik (dan pendapatan) yang seharusnya dihasilkan Turbin Angin akan hilang selamanya. Oleh karena itu, memprediksi kegagalan yang tertunda dengan peringatan yang memadai sangat penting untuk memaksimalkan pendapatan dari ladang Turbin Angin.

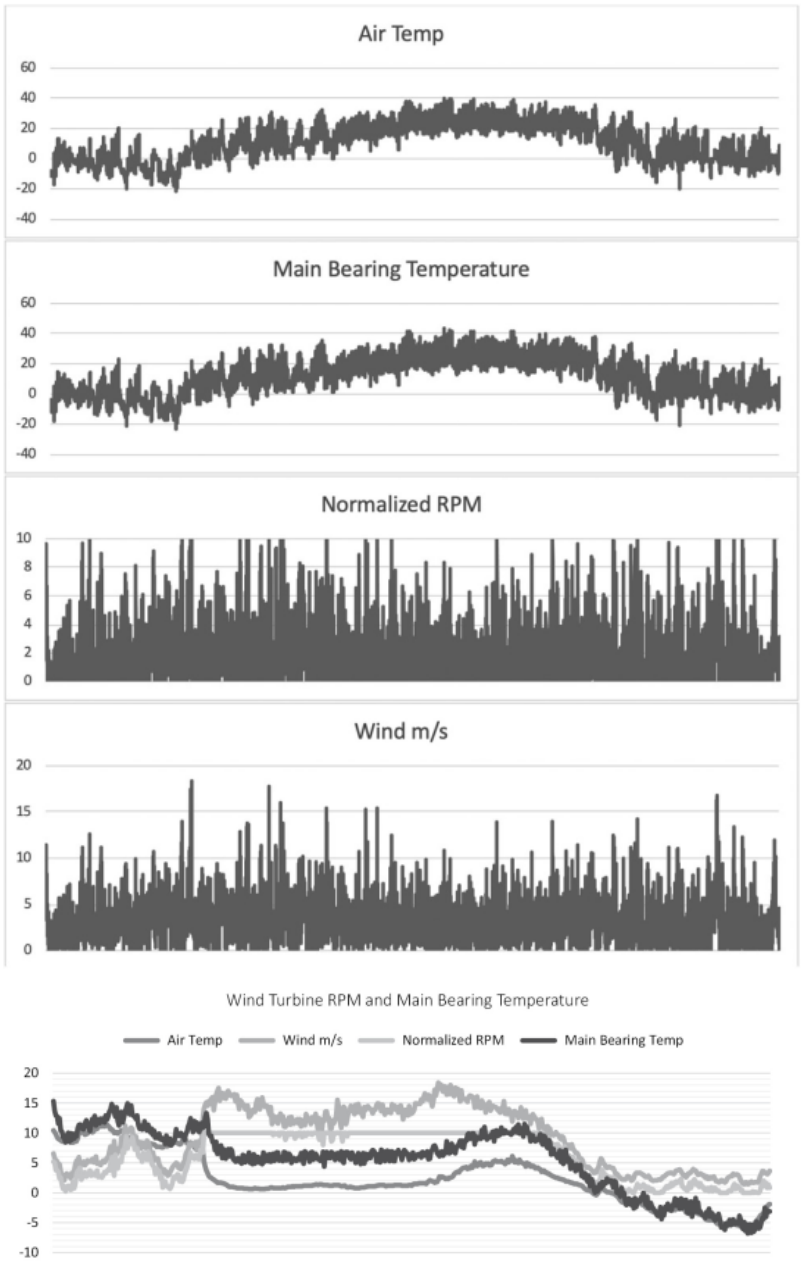
Diagram sederhana Turbin Angin ditunjukkan di bawah ini. Turbin angin biasanya terdiri dari tiga bilah rotor besar yang diarahkan ke arah angin yang bertiup. Rotor memiliki airfoil yang mirip dengan sayap pada pesawat terbang. Efek Bernoulli yang melintasi rotor menariknya berputar. Hal ini memutar poros di dalam Bantalan Utama. Gearbox mengubah putaran rotor yang lebih lambat (RPM) menjadi RPM yang lebih tinggi yang diperlukan untuk pembangkitan listrik yang efisien pada generator. Masing-masing komponen di dalam Turbin Angin ini dapat menjadi sumber kegagalan dan perlu dimodelkan untuk memprediksi kegagalan yang akan datang. Pemodelan perilaku normal bantalan utama akan digunakan sebagai contoh.



Gambar 3.1 Diagram Generator Turbin Angin.

Untuk contoh ini, sensor suhu bantalan utama akan menjadi sensor utama yang digunakan untuk menunjukkan adanya masalah yang tertunda pada bantalan utama. Di bawah ini adalah grafik suhu udara dan kecepatan angin di dekat Oakley, Kansas, untuk tahun 2019 dari data cuaca NOAA yang tersedia untuk umum dan bukan dari ladang angin yang sebenarnya (meskipun ladang angin banyak terdapat di Kansas barat). Plot suhu udara menunjukkan tren musiman tahunan dari musim dingin di bulan Januari, hingga musim panas, dan kemudian kembali ke musim dingin di bulan Desember. Siklus suhu harian dari yang lebih dingin di pagi hari hingga lebih hangat di sore hari juga terlihat dalam plot ini.

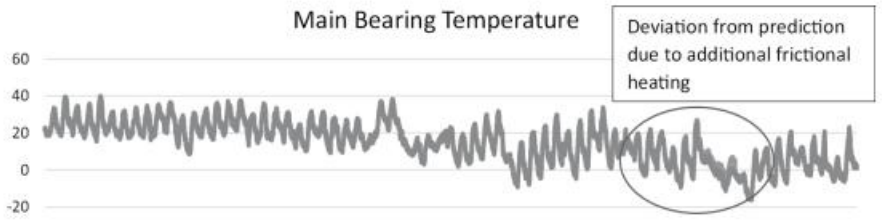
Kecepatan angin bervariasi, tetapi tidak secara jelas bersifat musiman. Simulasi spreadsheet sederhana dari turbin angin menunjukkan bahwa kecepatan putaran turbin (RPM) mengikuti kecepatan angin kecuali ketika kecepatan angin melebihi batas atas kemampuan putaran turbin. RPM dinormalisasi antara 0 dan 10 untuk grafik ini. Suhu bantalan utama mengikuti suhu udara tetapi umumnya lebih tinggi karena pemanasan gesekan saat turbin berputar.



Grafik ini memplot keempat masukan mulai akhir Oktober 2019. Garis hijau dan kuning menunjukkan bagaimana RPM melacak kecepatan angin hingga kemampuan RPM maksimum Turbin tercapai. Suhu bantalan utama berwarna merah melacak suhu udara, tetapi bergerak lebih tinggi ketika RPM rotor meningkatkan suhu bantalan utama akibat pemanasan gesekan. Ketika angin kencang dari front dingin datang, suhu bantalan utama tetap jauh di atas suhu udara hingga setelah front berlalu dan kecepatan angin kembali ke kisaran yang lebih normal. Dari sana, RPM menurun dan suhu bantalan utama kembali melacak suhu udara.

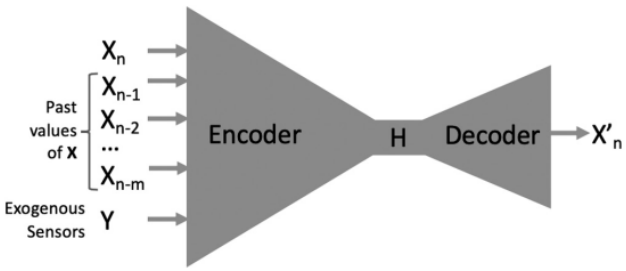
Autoencoder Jaringan Saraf Tiruan dapat dilatih untuk mempelajari semua hubungan ini hanya dengan mempelajari cara merekonstruksi keempat masukan ini ditambah sensor relevan lainnya pada turbin angin seperti sudut bilah, suhu nacelle, sensor getaran, dll. Setelah autoencoder dilatih, ia dapat digunakan untuk memprediksi masukan ini menggunakan data

langsung dari turbin angin. Jika bantalan utama mulai mengalami kerusakan mekanis, yang meningkatkan panas akibat gesekan, model akan terus memprediksi garis biru di bawah, tetapi suhu aktual akan mulai menyimpang ke nilai oranye, yang menunjukkan perlunya aktivitas pemeliharaan. Setelah diperbaiki, suhu bantalan utama akan kembali sesuai dengan nilai yang diprediksi.



Autoencoder, menurut definisi, memiliki masukan X yang sama dengan keluaran X'. Namun, untuk model perilaku normal aset fisik, beberapa modifikasi seringkali berguna dalam aplikasi industri. Misalnya, dalam kasus turbin angin ini, prediksi suhu udara dan kecepatan angin yang akurat tidak relevan untuk mendeteksi masalah yang tertunda pada turbin karena operator tidak memiliki kendali atas angin atau suhu. Masukan penting namun eksogen ini dapat diberikan sebagai serangkaian masukan Y ke encoder yang tidak termasuk dalam keluaran X' yang coba direkonstruksi oleh dekoder.

Demikian pula, versi tunda waktu dari beberapa masukan X dapat disertakan dalam Y, yang memungkinkan model untuk mempelajari ketergantungan waktu dalam data. Misalnya, perubahan RPM tidak terjadi secara instan dengan perubahan kecepatan angin karena momentum rotor yang besar. Demikian pula, perubahan suhu akibat gesekan juga tertinggal dari perubahan RPM atau perubahan suhu udara dan pada laju yang berbeda. Dengan demikian, encoder jaringan saraf tiruan mungkin memiliki beberapa subset $X_n, X_{n1}, X_{n-2}, \ldots X_{n-m}$, serta Y, yang semuanya diumpankan ke encoder dan kemudian digunakan oleh dekoder untuk memprediksi X'. Diagram ini mengilustrasikan konsep tersebut.



Contoh Turbin Angin jauh lebih sederhana daripada kebanyakan model perilaku normal yang dibuat untuk pemeliharaan prediktif. Aset yang lebih umum akan memiliki puluhan sensor di X'. Dalam kasus ini, sinyal perilaku abnormal mungkin terkandung dalam kesalahan rekonstruksi lebih dari satu sensor. Dengan demikian, beberapa bentuk skor agregat

menggunakan sesuatu seperti Mean Squared Error (MSE) atau skor Hoteling digunakan untuk membuat skor "abnormalitas" tunggal. Dalam semua kasus, kesalahan rekonstruksi untuk setiap sensor umumnya merupakan indikator yang baik bagi operator tentang tindakan apa yang harus diambil (misalnya, getaran atau suhu terlalu tinggi).

Dengan kumpulan data pelatihan normal yang wajar, model perilaku normal yang dibangun dari autoencoder jaringan saraf dapat sangat baik dalam mendeteksi ketika suatu aset berperilaku berbeda dari sebelumnya. Namun, model-model ini tidak dapat membedakan antara perilaku abnormal yang memerlukan pemeliharaan dan aset yang sekarang beroperasi dalam keadaan "normal baru". Yang terakhir dapat terjadi setelah perbaikan atau pemeliharaan di mana komponen baru atau pelumasan telah mengubah hubungan antara input. Jika operator menentukan bahwa model mendeteksi "normal baru", model tersebut perlu dilatih ulang dengan sampel data baru ini sebelum dapat kembali efektif. Pelatihan ulang berkala juga berguna untuk mengatasi penyimpangan yang tak terelakkan seiring keausan dan penuaan komponen mekanis.

Contoh Turbin Angin ini telah menunjukkan bagaimana model perilaku normal dapat dikembangkan untuk aset industri menggunakan data sensor historis dan Neural Network Autoencoder (atau variannya). Model ini dapat digunakan dengan data sensor langsung untuk mengidentifikasi kapan aset menyimpang dari operasi normal sebelumnya dan memberikan petunjuk penting tentang sensor mana yang menyimpang dari normal. Informasi ini dapat digunakan untuk mendiagnosis dan mengambil tindakan terhadap aset yang berada dalam kondisi suboptimal atau cenderung mengalami kegagalan sebelum kegagalan terjadi. Jenis model perilaku normal ini merupakan bagian penting dari sistem pemeliharaan preventif.

BAB 6
APLIKASI KHUSUS UNTUK VISI KOMPUTER

Sejauh ini dalam buku ini, kita telah membahas tiga topik utama: Kecerdasan Buatan, Pembelajaran Mesin, dan Jaringan Saraf Tiruan. Masih ada satu bidang lagi dalam Kecerdasan Buatan yang sedang mendapatkan perhatian serius—yaitu bidang Visi Komputer. Bidang ini akan menjadi fokus bab ini. Sesuai namanya, dengan Visi Komputer, kita mencoba mereplikasi cara kerja penglihatan manusia, tetapi pada tingkat komputer atau mesin. Dalam arti tertentu, hal ini sangat mirip dengan Kecerdasan Buatan, yang tujuan utamanya adalah mereplikasi proses berpikir, perilaku, dan pengambilan keputusan otak manusia.

Dalam bab ini, kita akan mulai dengan memberikan gambaran umum tingkat tinggi tentang Visi Komputer, dan dari sana, kita akan mendalami lebih dalam teori dan aplikasi yang mendorong bidang Kecerdasan Buatan yang sedang berkembang ini. Namun, sebelum kita mulai mempelajari lebih lanjut subjek ini, pertama-tama sangat penting untuk memberikan definisi teknis tentang apa itu Visi Komputer. Ini dia: *Visi komputer (CV) adalah subkategori dari Ilmu Komputer & Kecerdasan Buatan. Ini adalah serangkaian metode dan teknologi yang memungkinkan otomatisasi tugas tertentu dari sebuah gambar. Faktanya, sebuah mesin mampu mendeteksi, menganalisis, dan menginterpretasikan satu atau lebih elemen gambar untuk membuat keputusan dan melakukan suatu tindakan.*

Sederhananya, bidang Visi Komputer dari dalam konstruksi Kecerdasan Buatan memeriksa jenis gambar tertentu yang dimasukkan ke dalam sistem, dan dari sana, berdasarkan jenis algoritma matematika dan statistik yang digunakan, keluaran dihasilkan dari proses pengambilan keputusan yang berlangsung. Dalam hal ini, terdapat dua jenis Pengenalan Gambar yang sangat luas, yaitu sebagai berikut:

1) Deteksi Objek:
   Dalam istilah matematika, ini secara teknis dikenal sebagai "Segmentasi Poligon." Dalam hal ini, sistem ANN secara khusus mencari elemen dari dalam gambar tertentu dengan mengisolasinya ke dalam kotak tertentu. Hal ini dianggap jauh lebih unggul dan canggih daripada pendekatan piksel, yang masih paling banyak digunakan.
2) Klasifikasi Gambar:
   Ini adalah proses yang menentukan kategori suatu gambar berdasarkan komposisinya, yang utamanya digunakan untuk mengidentifikasi subjek utama dalam gambar.

Aplikasi untuk Visi Komputer

Meskipun Visi Komputer masih dalam tahap awal, ketika digunakan dengan sistem ANN, seperti yang telah disebutkan, visi ini digunakan dalam berbagai macam aplikasi, beberapa di antaranya adalah sebagai berikut:

- Pengenalan Karakter Optik: Ini adalah analisis, misalnya, berbagai tulisan tangan, dan bahkan pengenalan pelat otomatis (alias ANPR);

- Inspeksi Mesin: Ini terutama digunakan untuk Tujuan Pengujian Jaminan Kualitas, di mana cahaya khusus dapat disinari ke berbagai jenis proses manufaktur, seperti proses produksi komponen terpisah untuk pesawat terbang dan bahkan memeriksanya untuk menemukan cacat apa pun yang sulit dideteksi oleh mata manusia. Dalam kasus khusus ini, penglihatan Sinar-X (yang sebenarnya merupakan subkomponen dari sistem ANN) juga dapat digunakan;
- Pembuatan Model 3-D: Ini juga dikenal sebagai "Fotogrammetri", dan merupakan proses di mana Model 3-Dimensi dari foto survei udara, atau bahkan citra yang ditangkap oleh satelit, dapat dibuat ulang secara otomatis oleh sistem ANN;
- Pencitraan Medis: Visi Komputer dalam hal ini dapat digunakan untuk membuat citra pra-operasi serta pasca-operasi pasien tepat sebelum dan sesudah operasi;
- Pencocokan Gerakan: Proses ini memanfaatkan apa yang dikenal sebagai "Computer Generated Imager" (alias "CGI"), di mana berbagai titik fitur dapat dilacak dalam video berbasis sumber. Ini juga dapat digunakan untuk memperkirakan lebih lanjut tingkat gerakan Kamera 3-Dimensi, serta bentuk-bentuk lain yang dapat dipastikan dari video sumber;
- Penangkapan Gerak: Konsep-konsep di sini terutama digunakan untuk Animasi Komputer, di mana berbagai Penanda Retro-Reflektif dapat ditangkap;
- Pengawasan: Ini mungkin salah satu aspek Visi Komputer yang paling banyak digunakan. Dalam hal ini, Visi Komputer dapat digunakan bersama dengan teknologi CCTV serta teknologi Pengenalan Wajah untuk memberikan bukti positif bagi tersangka yang tertangkap.

Penting untuk dicatat bahwa Visi Komputer juga dapat digunakan dengan sangat baik untuk foto dan gambar diam, berbeda dengan foto dan gambar dinamis yang baru saja dijelaskan sebelumnya. Dengan demikian, dalam hal ini, beberapa aplikasi tipikal meliputi:

- Penggabungan: Teknik ini dapat digunakan untuk mengubah jenis gambar yang tumpang tindih menjadi satu "panorama gabungan" yang tampak hampir mulus;
- Bracketing Eksposur: Teknik ini dapat mengambil beberapa eksposur dari kamera canggih dalam kondisi pencahayaan yang sangat sulit dengan menggabungkan semuanya;
- Morphing: Dengan menggunakan matematika "Morphing", Anda dapat mengubah satu gambar menjadi gambar lain dengan jenis yang sama;
- Pencocokan Video/Penstabilan: Dengan proses khusus ini, seseorang dapat mengambil gambar 2-Dimensi dan 3-Dimensi dan secara harfiah memasukkannya ke dalam video untuk secara otomatis menemukan titik referensi berbasis matematika terdekat;
- Analisis Berbasis Foto: Dengan teknik khusus ini, Anda dapat menelusuri serangkaian gambar yang sangat berbeda untuk menentukan lokasi fitur-fitur utama;
- Autentikasi Visual: Ini juga dapat digunakan sebagai bentuk autentikasi, sama seperti kata sandi atau sidik jari Anda yang dapat 100 persen mengonfirmasi identitas, misalnya, saat Anda mengakses sumber daya bersama.

6.1 SEJARAH VISI KOMPUTER

Dibandingkan dengan Kecerdasan Buatan, Pembelajaran Mesin, dan Jaringan Saraf Tiruan, Visi Komputer belum ada selama itu, hanya karena kemajuan yang dicapai di dalamnya membutuhkan waktu lebih lama daripada yang lain. Namun, visi komputer juga memiliki sejarah yang cukup kaya, dan di bagian ini, kita akan mengulas beberapa poin pentingnya.

- Tahun 1970-an:

  Ini dianggap sebagai titik awal pertama Visi Komputer. Gagasan utamanya adalah bahwa Pembelajaran Mesin hanya akan meniru komponen dan aspek visual otak manusia. Namun, saat itu belum disadari betapa rumitnya proses ini. Jadi, fokus utamanya adalah membangun sistem Visi Komputer (CV) sebagai bagian dari keseluruhan sistem ANN yang dapat menganalisis hampir semua jenis masukan visual, dan menggunakannya untuk membantu menghasilkan keluaran yang diinginkan. Faktanya, upaya besar pertama yang diketahui dalam CV terjadi ketika seorang peneliti MIT ternama bernama Marvin Minsky meminta salah satu rekan penelitinya untuk menghubungkan kamera ke komputer dan membuatnya menghasilkan output sesuai dengan apa yang dilihatnya secara harfiah. Pada saat ini, terdapat perbedaan yang jelas antara CV dan bidang Pemrosesan Citra Digital. Dalam hal ini, berbagai citra 3 Dimensi diekstrapolasi dari citra 2 Dimensi itu sendiri. Terobosan penting lainnya yang terjadi pada periode ini juga mencakup hal-hal berikut:

  - Pengembangan Algoritma Pelabelan Garis;
  - Pengembangan rumus Deteksi Tepi untuk digunakan pada citra statis;
  - Implementasi pemodelan 3 Dimensi Objek non-Polihedral, memanfaatkan Silinder Umum;
  - Penciptaan Pola Elastis untuk membuat Struktur Gambar otomatis;
  - Pendekatan kualitatif pertama untuk Visi Komputer dimulai dengan penggunaan Citra Intrinsik;
  - Pendekatan yang lebih kuantitatif terhadap Visi Komputer telah diciptakan, seperti Algoritma Korespondensi Stereo dan Algoritma Aliran Optik Berbasis Intensitas.
  - Tiga teori kunci Visi Komputer juga dirumuskan, yaitu:

    1. Teori Komputasi:

       Teori ini mempertanyakan tujuan dari tugas Visi Komputer tertentu, dan dari sana, memastikan permutasi matematis apa yang diperlukan untuk mendapatkan keluaran yang diinginkan dari sistem ANN.
    2. Teori Representasi Citra dan Algoritma Korespondensi:

    Teori ini bertujuan untuk menjawab pertanyaan mendasar tentang bagaimana masukan, keluaran, dan kumpulan data antara digunakan untuk menghitung keluaran yang diinginkan.

    3. Teori Implementasi Perangkat Keras:

       Teori khusus ini mencoba untuk menentukan bagaimana perangkat keras sistem Visi Komputer dapat dikaitkan dengan perangkat keras

sistem ANN untuk menghitung keluaran yang diinginkan. Kebalikannya juga berlaku, yaitu mencoba untuk menentukan bagaimana perangkat keras dapat dikaitkan dengan algoritma CV dengan cara yang paling efisien.

- Tahun 1980-an:

  Dalam rentang waktu spesifik ini, lebih banyak pekerjaan dilakukan untuk menyempurnakan dan memajukan aspek matematika Visi Komputer, yang fondasinya telah dibangun pada tahun 1970-an. Perkembangan utama di era ini meliputi:

  - Pengembangan Piramida Citra untuk digunakan dalam apa yang dikenal sebagai "Pencampuran Citra";
  - Pengembangan Pemrosesan Skala Ruang, di mana piramida yang telah dibuat dapat dipindahkan ke aplikasi CV selain yang awalnya ditujukan untuknya;
  - Penciptaan Isyarat Bentuk Kuantitatif berbasis stereo untuk digunakan dalam berbagai jenis aplikasi Sinar-X;
  - Penyempurnaan algoritma matematika berbasis Deteksi Tepi dan Kontur (ini juga mengarah pada penciptaan "Pelacak Kontur");
  - Pengembangan berbagai jenis Model Fisik berbasis 3 Dimensi;
  - Pengembangan Model Medan Acak Markov diskret, yang memungkinkan algoritma matematika deteksi stereo, aliran, dan tepi disatukan dan dioptimalkan sebagai satu set kohesif untuk digunakan oleh sistem JST;
  - Penyempurnaan lebih lanjut juga dilakukan pada Model Medan Acak Markov, yang meliputi:
  - Pemetaan "Filter Visi Kalman";
  - Pemetaan otomatis Model Medan Acak Markov sehingga dapat digunakan sebagai pendahulu pemrosesan paralel yang akan dilakukan dari dalam sistem JST;
  - Pengembangan teknik Pemrosesan Data Rentang 3-Dimensi, yang akan digunakan untuk akuisisi, penggabungan, pemodelan matematika, dan pengenalan berbagai gambar yang akan dimasukkan ke dalam sistem JST.

- Tahun 1990-an:

  Era ini dalam Visi Komputer juga menyaksikan perkembangan penting berikut:

  - Pengembangan algoritma yang dikenal sebagai "Rekonstruksi Proyektif" yang utamanya digunakan untuk kalibrasi kamera secara tepat agar dapat mengambil gambar yang diperlukan untuk digunakan oleh sistem ANN;
  - Penciptaan dan implementasi "Teknik Faktorisasi" untuk menghitung secara akurat perkiraan yang dibutuhkan untuk kamera berbasis Ortografi;
  - Pengembangan "Teknik Penyesuaian Bundel" yang akan digunakan di hampir semua jenis teknik Fotogrametri;
  - Pengembangan penggunaan warna dan intensitas pada gambar spesifik, yang memanfaatkan apa yang dikenal sebagai "Transportasi Radiasi" dan "Pembentukan

Citra Berwarna" yang dapat langsung diterapkan pada subset baru Visi Komputer yang pada saat itu dikenal sebagai "Visi Berbasis Fisika";

- Penyempurnaan berkelanjutan sebagian besar Metode Aliran Optik yang digunakan oleh komponen Visi Komputer yang berasal dari dalam sistem ANN;
- Penyempurnaan Algoritma Korespondensi Stereo Padat;
- Penelitian yang jauh lebih aktif dan dinamis mulai dilakukan dalam implementasi Algoritma Stereo Multi-Tampilan yang dapat diterapkan untuk mereplikasi dan menghasilkan gambar 3 Dimensi dengan mudah;
- Pengembangan algoritma matematika yang dapat digunakan untuk merekam dan menghasilkan berbagai Deskripsi Volumetrik 3 Dimensi dari berbagai siluet tipe Biner;
- Teknik juga dikembangkan untuk pengembangan konstruksi yang dikenal sebagai "Kontur Oklusi Halus";
- Algoritma Pelacakan Gambar mengalami peningkatan pesat, terutama dalam hal berbagai algoritma Pelacakan Kontur seperti "Ular", "Filter Partikel", dan "Kumpulan Level";
- Penelitian yang jauh lebih aktif juga mulai memicu bidang subset Visi Komputer yang dikenal sebagai "Segmentasi Gambar". Teknik-teknik yang dikembangkan di bidang ini meliputi Energi Minimum, Panjang Deskripsi Minimum, Potongan Ternormalisasi, dan Pergeseran Rata-rata yang dapat diterapkan pada analisis gambar dari dalam sistem ANN;
- Periode waktu khusus ini juga menandai lahirnya algoritma berbasis statistik pertama yang digunakan dalam Visi Komputer. Ini pertama kali diterapkan pada aplikasi sistem ANN seperti Analisis Komponen Utama (alias "PCA"), yang bergantung pada penggunaan Eigenfaces yang intensif, dan pengembangan Sistem Dinamis Berbasis Linear, yang digunakan dalam Pelacakan Kurva;
- Perkembangan paling berkelanjutan dalam Visi Komputer yang terjadi selama periode ini mungkin adalah meningkatnya interaksi dengan Grafik Komputer, yang juga dapat digunakan dalam subbidang Pemodelan Berbasis Gambar dan bahkan Rendering;
- Berbagai jenis algoritma Image Morphing juga diciptakan untuk membuat animasi komputer dari gambar statis dan dinamis. Algoritma spesifik ini juga dapat diterapkan pada Image Photo Stitching, dan Full Light Field Rendering;
- Jenis algoritma matematika dan statistik lainnya dikembangkan sehingga Model Gambar 3 Dimensi dapat dibuat secara otomatis dari serangkaian gambar statis.

▪ Tahun 2000-an dan Seterusnya:

Periode ini mungkin menjadi saksi interaksi terbesar antara Visi Komputer dan Grafik Komputer. Berikut perkembangannya sejauh ini:

- Subbidang Visi Komputer, yang meliputi Penggabungan Gambar, Pengambilan Medan Cahaya, dan Rendering, serta teknik Rentang Dinamis Tinggi (alias HDR) digabungkan menjadi satu bidang Visi Komputer yang spesifik, yang kemudian

dikenal sebagai "Fotografi Komputasional". Sejak kemunculannya, berbagai jenis algoritma "Pemetaan Nada" dikembangkan;

- Berbagai jenis algoritma berbasis statistik dan matematika lainnya juga diciptakan agar Gambar berbasis Flash dapat dengan mudah digabungkan dengan Gambar Non-Flash, serta untuk memisahkan segmen yang tumpang tindih pada gambar statis dan dinamis menjadi entitas uniknya sendiri;
- Teknik Sintesis Tekstur dan Inpainting dikembangkan untuk menciptakan gambar baru dari gambar sampel;
- Berbagai prinsip, yang kemudian dikenal sebagai "Teknik Berbasis Fitur" juga berkembang, yang dapat digunakan untuk Pengenalan Objek oleh sistem ANN. Ini mencakup pengembangan Model Konstelasi dan Teknik Struktur Pictorial, serta Teknik Berbasis Titik Minat, yang memanfaatkan kontur dan segmentasi wilayah pada citra statis dan dinamis;
- Teori "Looping Belief Propagation" juga dikembangkan, yang memungkinkan citra statis dan dinamis untuk disematkan dan dianalisis lebih lanjut pada Bidang Geometri Kartesius dan bidang grafik kompleks lainnya;
- Terakhir, periode ini juga menyaksikan kombinasi teknik Pembelajaran Mesin ke dalam Visi Komputer yang dapat digunakan oleh sistem ANN untuk menghasilkan keluaran yang dihasilkan.

Sejauh ini dalam bab ini, kami telah memberikan definisi teknis untuk Visi Komputer dan beberapa aplikasi yang dilayaninya, serta memberikan latar belakang historis tentang bagaimana visi ini menjadi bidang seperti sekarang ini, dan mengeksplorasi dominasinya dalam bidang Kecerdasan Buatan. Sisa bab ini sekarang dikhususkan untuk mendalami lebih dalam konstruksi teoretis Visi Komputer.

6.2 PEMBUATAN GAMBAR DALAM VISI KOMPUTER (PEMBUATAN GAMBAR)

Setelah kita membahas secara mendalam apa itu Visi Komputer, terdapat banyak konstruksi, proses, dan prosedur teoretis yang menyertainya. Langkah pertama untuk memulai dalam hal ini adalah Pembuatan Gambar, baik yang bersifat statis maupun dinamis. Subbagian berikut akan membahasnya lebih detail.

Konstruksi Geometris—Fase 2 Dimensi

Setiap jenis gambar, baik statis maupun dinamis, pada dasarnya dibuat menggunakan prinsip-prinsip Geometri. Konsep-konsep di sini banyak digunakan untuk membuat gambar 3 Dimensi yang kuat. Komponen penyusunnya adalah garis, titik, dan bidang sederhana. Namun perlu diingat bahwa hal-hal ini juga dapat menjadi sangat kompleks, tergantung pada seberapa kaya gambar 3 Dimensi tersebut.

Kita mulai dengan apa yang dikenal sebagai Titik 2 Dimensi. Hal ini dapat direpresentasikan secara matematis sebagai berikut:

$$X = (x, y) \text{ E } R\text{^}2.$$

Jika Titik-titik 2 Dimensi ini menggunakan apa yang dikenal sebagai "Koordinat Heterogen", maka titik-titik ini dapat diimplementasikan kembali ke bidang geometrisnya, yang secara teknis dikenal sebagai "Ruang Proyektif 2 Dimensi". Berbagai macam Vektor Homogen kemudian digunakan, dan dapat direpresentasikan secara matematis sebagai berikut:

$$X = (\underline{X}, \underline{Y}, \underline{W}) = \underline{W}(X, y, 1) = W\underline{X}i$$

Di mana:
$\underline{W}(X, y, 1)$ = Vektor Tertambah.

Dengan Titik-titik 2 Dimensi, ikuti Garis 2 Dimensi. Sebuah garis tunggal dalam hal ini dapat direpresentasikan secara matematis sebagai berikut:

$$\underline{X}^{*}\underline{I} = ax + by + c = 0.$$

Perpotongan dua Garis 2 Dimensi direpresentasikan secara matematis sebagai berikut:

$$\underline{X} = \underline{I}1 \text{ X } \underline{I}2.$$

Nah, jika kedua Garis 2 Dimensi ini dapat digabungkan, secara matematis juga direpresentasikan sebagai berikut:

$$\underline{I} = \underline{I}1 \text{ X } \underline{I}2.$$

Setelah kita memiliki Garis 2 Dimensi dan Titik 2 Dimensi, hal berikutnya yang dapat diciptakan dalam gambar statis atau dinamis adalah apa yang dikenal sebagai "Kerucut", atau singkatnya, Kerucut. Kerucut ini menggunakan persamaan Polinomial Homogen, dan ini dapat direpresentasikan dengan rumus semi-kuadrat sebagai berikut:

$$\underline{X}\text{^T X } Q\underline{x} = 0.$$

Faktanya, persamaan Kuadrat memainkan peran besar dalam kalibrasi kamera tempat gambar diambil.

Konstruksi Geometri—Fase 3 Dimensi

Sekarang kita beralih untuk membahas fitur 3 Dimensi penting untuk gambar statis maupun dinamis. Misalnya, sebuah Titik 3 Dimensi direpresentasikan secara matematis sebagai berikut:

$$X = (\underline{X}, \underline{Y}, \underline{Z}, \underline{W}) \text{ E P^3}.$$

Dari Titik 3 Dimensi muncullah Bidang 3 Dimensi. Persamaan matematika yang digunakan untuk merepresentasikan hal ini lebih lanjut adalah sebagai berikut:

$$\underline{X}^{*}\underline{M} = ax + by + cz + d = 0.$$

Berbagai sudut pada bidang geometri semacam ini dapat dilihat sebagai berikut:

$$N = (COS\ 0, COS\ 0/, SIN\ 0/\ COS0/, SIN\ 0/).$$

Perlu dicatat lebih lanjut bahwa pada bidang geometris ini, koordinat bola digunakan, tetapi penggunaan koordinat polar jauh lebih umum untuk aplikasi Visi Komputer saat ini dalam sistem ANN.

Mungkin blok bangunan paling dasar dalam pembentukan sudut 3 Dimensi adalah garis 3 Dimensi. Pada tingkat paling primitifnya, dua titik linear pada satu garis tunggal dapat direpresentasikan secara matematis sebagai berikut:

$$(p, q).$$

Dalam matematika berbasis linear, kombinasi kedua titik ini dapat dilihat sebagai berikut:

$$R = (1 - Y)p + Yq.$$

Jika koordinat homogen digunakan, Garis 3 Dimensi dapat direpresentasikan secara matematis sebagai berikut:

$$\underline{R} = u\underline{p} + Y\underline{q}.$$

Perlu dicatat pada titik ini bahwa kelemahan utama Garis 3 Dimensi adalah terlalu banyak derajat kebebasan statistik pada titik-titik ujung garis semacam ini. Dalam contoh umum ini, terdapat tiga derajat kebebasan untuk kedua titik ujung dari satu Garis 3 Dimensi. Untuk mengatasi kekurangan ini, dengan hasil akhir berupa Garis 3 Dimensi yang dapat dibentuk sudutnya pada hampir semua orientasi, konsep "Teorema Koordinat Plucker" digunakan. Teorema ini direpresentasikan secara matematis sebagai berikut:

$$L = p\_q\text{^}T - \underline{qp}\text{^}T$$

Di mana:

$\underline{P}, \underline{Q}$ = Dua titik linear yang terletak di sepanjang Garis 3 Dimensi.

Seperti halnya Kerucut 2 Dimensi, Kerucut 3 Dimensi dapat dibuat, juga menggunakan persamaan semi-kuadrat. Teorema ini direpresentasikan secara matematis sebagai berikut:

$$\underline{X}\text{^}TQ\underline{x} = 0.$$

Konstruksi Geometri—Transformasi 2 Dimensi

Penting untuk diingat bahwa setiap garis, titik, atau kerucut (baik 2 Dimensi maupun 3 Dimensi) dapat dimanipulasi sedemikian rupa sehingga bayangan yang dibentuknya dapat diubah menjadi bayangan terkait. Konstruksi matematika untuk melakukan hal ini dikenal sebagai "Transformasi". Dari sudut pandang dua transformasi 2 Dimensi, hal ini dapat direpresentasikan secara matematis sebagai berikut:

$$X' = x + t$$

Di mana:
$\mathbf{I}$ = Matriks Berbasis Identitas 2 X 2.

Matriks jenis ini juga direpresentasikan secara matematis sebagai berikut:

$$X' [1\ t] * [0\text{^}T\ 1] * X.$$

Setelah hal di atas sepenuhnya ditetapkan, transformasi dapat diputar dalam berbagai derajat sebagaimana disyaratkan oleh sistem ANN. Ini secara teknis dikenal sebagai "Gerakan Benda Tegar 2 Dimensi", atau juga dikenal sebagai "Transformasi Euklides 2 Dimensi". Terdapat dua cara matematis yang terpisah dan berbeda untuk merepresentasikan hal ini, yaitu sebagai berikut:

$$\text{Representasi \#1: } X' = [R\ t] * \underline{x}$$
$$\text{Representasi \#2: } X' = Rx + t$$

Di mana:
$R = [\text{COS } 0/, \text{SIN } 0/] * [-\text{SIN } 0/, \text{COS } 0/].$

Penting untuk dicatat bahwa kedua kasus representasi di atas menggunakan apa yang dikenal sebagai "Matriks Rotasi Ortonormal".

Namun, ini bukan satu-satunya transformasi yang ada untuk gambar 2 Dimensi yang statis maupun dinamis. Ada juga yang lain, yaitu sebagai berikut:

- Rotasi Berskala:
  Ini juga dikenal secara teknis sebagai "Transformasi Kesamaan", dan secara matematis dapat direpresentasikan sebagai berikut:

$$X' = [sR\ t] * \underline{x} = [a, b] * [-b, a] * [Tx, Ty] * \underline{X}1.$$

- Transformasi Afin:
  Ini direpresentasikan secara matematis sebagai berikut:

$$X' = [a00, a10] * [a01, a1I] * [a02, a12]] * \underline{X.}$$

- Transformasi Proyektif:
  Transformasi 2 Dimensi jenis ini menggunakan Koordinat Homogen (sebagaimana telah diulas sebelumnya dalam bab ini), dan ini direpresentasikan secara matematis sebagai berikut:

$$X' = [h00 + h01y + h02] * [h20x + h21y + h22]$$
$$Y' = [h10x + h11y + h12] * [h20x + h21y + h22].$$

  Penting juga untuk dicatat bahwa Garis 2 Dimensi dalam transformasi jenis ini juga dapat ditransformasikan satu per satu, dan bukan sebagai satu kesatuan yang kohesif. Hal ini dapat dicapai dengan rumus matematika berikut:

$$\underline{L} * \underline{x} = \underline{l}\text{^T} * H\underline{x} = (\underline{H}\text{^Ti})\text{^T}\underline{x} = \underline{l} * \underline{x} = 0.$$

- Transformasi Peregangan dan Peregangan:
  Transformasi semacam ini secara harfiah dapat mengubah rasio matematika gambar. Hal ini dapat direpresentasikan sebagai berikut:

$$X' = SxX + tX$$
$$Y' = syY + Ty1.$$

- Transformasi Aliran Permukaan Planar:
  Teknik transformasi jenis ini digunakan dalam kasus-kasus tertentu di mana gambar, baik statis maupun dinamis, mengalami serangkaian rotasi tertentu, tetapi hanya pada tingkat inkremental kecil sehingga perubahan ini dapat ditangkap oleh sistem ANN. Teknik ini dapat dicapai dengan dua persamaan matematika berikut:

$$X' = a0 + a1x + a2y + a6x\text{^}2 + a7xy$$
$$Y' = a3 + a4x + a5y + a7x\text{^}2 + a6xy.$$

- Transformasi Interpolasi Bilinear:
  Teknik khusus ini dapat digunakan untuk mengoreksi deformitas apa pun pada citra, baik statis maupun dinamis, jika citra tersebut berbentuk persegi atau tidak. Persamaan matematika berikut dapat digunakan untuk menyelesaikan tugas khusus ini:

$$X' = a0 + a1x + a2y + a6xy$$
$$Y' = a3 + a4x + a5y + a7xy.$$

Konstruksi Geometris—Transformasi 3 Dimensi

Jumlah total teknik transformasi yang tersedia untuk gambar 3 Dimensi yang statis dan/atau dinamis tidak sebanyak teknik transformasi untuk gambar 2 Dimensi. Namun, teknik-teknik ini tetap penting dalam penggunaan dan fungsi spesifiknya, yaitu sebagai berikut:

- Teknik Transformasi Dasar:
- Persamaan matematika yang digunakan untuk contoh ini direpresentasikan sebagai berikut:

$$X' = [I\ t] * x$$

Di mana:
I = A 3X3 Matriks Identitas berbasis matematika .

- Transformasi Rotasi dan Translasi:
  Ini adalah jenis teknik transformasi khusus yang eksklusif untuk gambar 3 Dimensi yang bersifat statis maupun dinamis. Teknik ini juga disebut secara teknis sebagai "Gerakan Benda Tegar 3 Dimensi", dan rumus matematika berikut dapat digunakan untuk menyelesaikan tugas khusus ini:

$$X' = R(x - c) = Rx - Rc.$$

- Transformasi Rotasi Berskala:
  Teknik semacam ini dapat direpresentasikan secara matematis sebagai berikut:

$$X' = [sR\ t] * \underline{x.}$$

- Transformasi Afin:
  Teknik ini digunakan ketika gambar statis maupun dinamis diasumsikan sebagai matriks matematis tiga kali empat. Teknik ini direpresentasikan sebagai berikut:

$$X'\ [a00, a10, a20] * [a01, a11, a21] * [a02, a12, a22] * [a03, a13, a23] * \underline{x.}$$

- Transformasi Proyektif:
  Teknik ini juga menggunakan Koordinat Homogen, dan dalam istilah yang lebih teknis, dikenal juga sebagai "Transformasi Perspektif 3 Dimensi." Teknik ini direpresentasikan secara matematis sebagai berikut:

$$\underline{X} = \underline{Hx}.$$

Konstruksi Geometris—Rotasi 3 Dimensi

Tidak seperti gambar 2 Dimensi, gambar 3 Dimensi (baik statis maupun dinamis) dapat diputar dengan besaran yang bervariasi ke berbagai arah. Rotasi ini bisa sekecil beberapa derajat, atau jauh lebih besar dari itu, di sisi ekstrem lainnya. Ada sejumlah teknik matematika yang dapat digunakan untuk menyelesaikan tugas semacam ini agar dapat diproses oleh sistem ANN, yaitu sebagai berikut:

- Sudut Euler:
  Di sinilah derajat rotasi tertentu dicapai ketika produk matematika dari tiga gerakan independen terjadi di sekitar titik sumbu gambar, baik statis maupun dinamis. Namun, teknik ini tidak banyak digunakan saat ini karena tidak ada himpunan permutasi yang baku untuk diikuti dalam memutar gambar 3 Dimensi yang dimaksud.
- Teknik Putaran Eksponensial:
  Teknik ini digunakan ketika gambar 3 Dimensi (baik statis maupun dinamis) diputar dalam berbagai derajat oleh vektor matematika 3 Dimensi. Rotasi semacam ini dihitung dengan rumus matematika berikut:

$$V|| = n\hat{}(n * v) = (n\hat{}n\hat{}) * v.$$

  Agar rotasi citra 3-Dimensi seoptimal mungkin, berbagai macam vektor matematika digunakan. Salah satu teknik vektor populer direpresentasikan sebagai berikut:

$$U = uT + v|| = (I + SIN\,0/[n\hat{}]x + (1 - COS0)[n\hat{}]2/x) * v.$$

- Teknik Kuaternion Satuan:
  Teknik khusus ini menggunakan matriks matematika empat vektor. Ini dapat direpresentasikan secara matematis sebagai berikut:

$$Q = (qx, qy, qz, qw).$$

Penting untuk dicatat bahwa teknik ini mengasumsikan bahwa sifat rotasi citra 3-Dimensi selalu kontinu, dan tidak akan dihentikan oleh sistem ANN hingga permutasi spesifik dimasukkan ke dalamnya. Teknik ini banyak digunakan untuk jenis aplikasi yang memanfaatkan pose. Perlu dicatat bahwa "Quaternion" dapat dihitung dengan rumus matematika berikut:

$$Q = (v, w) = (SIN0/2\ n\hat{}, COS0/2).$$

Kebalikan dari Quaternion dikenal sebagai Quaternion "Antipodal", dan dihitung dengan rumus matematika berikut:

$$Q2 = q0/q1 = q0q1\hat{} - 1 = (v0\ x\ v1 + w0v1 - w1v0 - w0w1 - v0 * v1).$$

Rotasi inkremental dalam hal ini juga secara teknis dikenal sebagai "Interpolasi Linear Sferis", dan dihitung dengan dua rumus matematika berikut:

$$Q2 = q\text{^}ar * q0$$
$$Q2 = [SIN(1 - A)\text{^}0/SIN0] * q0 + [SINA0/SIN0] * q1.$$

6.3 MENENTUKAN TEKNIK 3 DIMENSI YANG OPTIMAL UNTUK SISTEM ANN

Dalam hal rotasi spesifik gambar 3 Dimensi, teknik mana yang akan digunakan (sebagaimana diulas di subbagian terakhir) terutama bergantung pada aplikasi yang dimaksud dan keluaran yang diinginkan dari sistem ANN. Perlu dicatat lebih lanjut bahwa representasi matematis dari berbagai sudut atau sumbu dalam gambar 3 Dimensi (baik statis maupun dinamis) tidak memerlukan daya pemrosesan atau overhead tambahan dari sistem ANN.

Untuk menentukan suatu teknik sebagai yang paling efektif, sangat penting untuk menyatakannya sebagai kondisi derajat geometri. Ini juga dapat dinyatakan sebagai fungsi dari apa yang dikenal sebagai "Radian". Dalam hal ini, sistem ANN juga dapat menggunakan Kuaternion (juga dibahas sebelumnya dalam bab ini). Namun, teknik ini, dari sudut pandang optimasi, hanya boleh digunakan ketika kamera yang mengambil gambar sedang bergerak, baik yang bersifat linear maupun kurvilinear.

BAB 7
IMPLEMENTASI GAMBAR DIMENSI KE BIDANG GEOMETRIS

Setelah kita menetapkan fondasi yang kokoh mengenai prinsip-prinsip yang berkaitan dengan gambar 2 Dimensi maupun 3 Dimensi (baik statis maupun dinamis), langkah selanjutnya dalam proses ini adalah menentukan cara memproyeksikan gambar 3 Dimensi agar sistem ANN dapat memprosesnya. Secara matematis, tugas ini dapat diselesaikan secara spesifik dengan memanfaatkan matriks proyeksi 3 Dimensi maupun 2 Dimensi. Dalam contoh khusus ini, mungkin matriks matematika yang paling efisien dan paling sederhana untuk digunakan adalah "Matriks Ortografis".

Matriks ini dapat direpresentasikan secara matematis sebagai berikut:

$$X = [I2x2|0] * p.$$

Namun, jika Koordinat Homogen digunakan dalam contoh ini, maka algoritma di atas dapat dinyatakan sebagai berikut:

$$\underline{X} = [1,0,0] * [0,1,0] * [0,0,0] * [0,0,1] * \underline{P}1.$$

Matriks matematika semacam ini dapat diterapkan secara khusus pada kamera yang menggunakan lensa yang bersifat "Telesentrik", misalnya, jika lensa ini menggunakan titik fokus yang sangat panjang dan titik referensi gambar yang akan diambil dangkal relatif terhadap keseluruhan latar depannya. Penskalaan merupakan konsep yang sangat penting di sini, dan oleh karena itu, "Ortografi Berskala" banyak digunakan dalam hal ini. Secara matematis, hal ini dapat direpresentasikan sebagai berikut:

$$X = [sI2x2|0] * P.$$

Penskalaan semacam ini biasanya dapat digunakan dalam berbagai bingkai gambar, secara cepat dan berurutan. Ini juga disebut sebagai "Struktur dalam Gerak". Perlu dicatat juga bahwa teknik ini banyak digunakan untuk menciptakan kembali citra 3 Dimensi yang diambil dari jarak yang sangat jauh. Variabel "Pose" sangat penting di sini, dan secara statistik, dapat direpresentasikan ke bidang geometri sebagai Jumlah Kuadrat Terkecil. Sifat matematika "Faktorisasi" juga dapat digunakan sebagai pengganti.

Teknik lain yang digunakan untuk menyebarkan citra 3 Dimensi (baik statis maupun dinamis) ke bidang geometri dikenal sebagai konsep "Para Perspektif". Dalam hal ini, semua titik referensi dalam citra 3 Dimensi pertama-tama diproyeksikan ke subset bidang geometri aktual yang akan digunakan. Namun, setelah subset khusus ini siap, ia tidak akan diproyeksikan

ke bidang geometri secara ortogonal, melainkan secara paralel. Secara matematis, proyeksi paralel pada bidang geometri dapat direpresentasikan sebagai berikut:

$$X = [a00, a10, 0] * [a01, a11, 00 * [a02, a12, 0] * [a03, a13, 1] * P.$$

7.1 TEKNIK PERSPEKTIF 3 DIMENSI

Sesuai dengan judul subbagian ini, gambar 3 Dimensi yang dimaksud diproyeksikan ke bidang geometri dengan membagi titik-titik referensi yang terdapat pada gambar 3 Dimensi itu sendiri. Hal ini banyak menggunakan koordinat homogen, dan secara matematis direpresentasikan sebagai berikut:

$$X = Pz\,(P) = [X/Z] * [Y/Z] * [1].$$

Representasi koordinat homogen diberikan oleh matriks matematika berikut:

$$X = [1, 0, 0] * [0, 1, 0] * [0, 0, 1] * [0, 0, 0] * \underline{P}1.$$

Subset dari teknik ini sebenarnya menggunakan pendekatan dua fase:

1) Koordinat dari gambar 3 Dimensi dikonversi menjadi apa yang dikenal sebagai "Koordinat Perangkat Ternormalisasi", yang secara matematis direpresentasikan sebagai berikut:

$$(x, y, z)\ E\ [-1, -1]\ X\ [-1, 1]\ X\ [0, 1].$$

2) Koordinat ini kemudian diskalakan ulang dan bahkan diproyeksikan ulang ke bidang geometris dengan menggunakan teknik lain yang disebut "Transformasi Viewport". Ini direpresentasikan sebagai berikut:

$$\underline{X} = [1, 0, 0, 0] * [0, 1, 0, 0] * [0, 0, -zFAR/zRANGE, 1] * [0, 0, zNEARzFAR/zRANGE, 0] * Pr$$

Di mana:

$$zNEAR\ dan\ zFAR = Bidang\ Kliping\ Z.$$

Gambar 2 Dimensi juga dapat diproyeksikan ke bidang geometris, tetapi tidak seumum gambar 3 Dimensi, hanya karena kurangnya algoritma matematika. Namun, jika aplikasi membutuhkan gambar 2 Dimensi, teknik "Sensor Jarak" digunakan, yang menggunakan matriks matematika empat kali empat.

7.2 MEKANIKA KAMERA

Setelah langkah-langkah di atas selesai, titik referensi pada gambar 3 Dimensi yang dimaksud (baik statis maupun dinamis) masih harus dipikselkan ke bidang geometris relatif

terhadap titik asalnya (jika kuadran digunakan, ini akan direpresentasikan sebagai [0,0]). Untuk menyelesaikan tugas khusus ini, algoritma matematika berikut paling umum digunakan:

$$P = [Rs|Cs] * [Sx, 0, 0, 0] * [0, Sy, 0, 0,] * [0, 0, 0, 1] * [Xs, Ys, 1] = MsXs.$$

Sekarang, hubungan spesifik antara titik referensi dari gambar 3 Dimensi dan proyeksinya ke bidang geometri dapat didefinisikan secara matematis sebagai berikut:

$$Xs = aM^{-}1s * Pc = KPc.$$

Hasil proyeksi ini menjadi matriks matematika tiga-kali-tiga, yang dilambangkan sebagai "K". Matriks ini juga disebut "Matriks Kalibrasi", dan memberikan gambaran umum tentang mekanisme kamera yang mengambil gambar 3-Dimensi dalam kaitannya dengan orientasi vektornya pada bidang geometris. Yang terakhir ini dikenal sebagai "Ekstrinsik" kamera.

Setelah gambar 3-Dimensi tertanam ke dalam bidang geometris menggunakan konsep Pikselasi, kamera perlu dikalibrasi agar gambar yang mulus dapat diambil dan diproses dengan cepat dan efisien oleh sistem ANN untuk mendapatkan keluaran yang diinginkan. Kalibrasi spesifik ini dapat dicapai dengan algoritma matematika berikut:

$$Xs = K [R|t] * Pw = Pps^t$$

Di mana:
Pw = "Koordinat Dunia" 3-Dimensi;
$K [R|t]$ = matriks matematika yang digunakan oleh kamera yang dimaksud.

Menentukan Panjang Fokus Kamera

Salah satu tantangan terbesar yang masih belum teratasi di bidang Visi Komputer adalah menentukan dan memastikan bagaimana panjang fokus dari kamera ke gambar 3-Dimensi perlu direpresentasikan. Alasan utama kurangnya pemahaman ini adalah kenyataan bahwa panjang fokus sangat bergantung pada satuan spesifik yang digunakan untuk mengukur ukuran piksel. Salah satu metode untuk mengatasi dilema ini adalah dengan menentukan hubungan matematis antara Panjang Fokus (dilambangkan sebagai "f") kamera dan lebar numerik sensor (dilambangkan sebagai "W") yang telah ditanamkan ke dalam kamera dengan keseluruhan bidang tangkap fotografisnya (yang dilambangkan sebagai "0/").

Hal ini direpresentasikan secara matematis sebagai berikut, dalam dua format berbeda:

$$\text{Format 1: TAN } 0/2 = W/2f;$$
$$\text{Format 2: } f = W/2 [\tan)/2] ^ -1.$$

Jika kamera biasa digunakan untuk mengambil gambar 3 Dimensi, maka satuan metrik standar milimeter seringkali merupakan pilihan terbaik untuk tujuan optimasi. Metrik umum lain yang dapat menggantikan milimeter ini adalah piksel. Solusi lain untuk dilema di atas adalah dengan menyatakan koordinat piksel sebagai serangkaian rentang matematis, yang dapat berkisar dari -1 hingga 1. Ini juga dikenal sebagai "Penskalaan Naik".

Namun, jika rentang yang lebih panjang harus digunakan, hal ini dapat direpresentasikan secara matematis sebagai berikut:

$$[-a\hat{} - 1, a\hat{} - 1].$$

Ini juga dikenal sebagai "Rumus Rasio Aspek Gambar", dan ini dapat direpresentasikan secara matematis sebagai berikut:

$$X = (2Xs - L)/S;$$
$$Y = (2Ys - T)/S$$

Di mana:
$S = \text{maks}(W, H).$

Teknik "Scaling Up" memiliki sejumlah keunggulan utama, yaitu sebagai berikut:

- Panjang Fokus (dilambangkan sebagai "f") dan Pusat Optik (dilambangkan sebagai "Cx, Cy") menjadi independen satu sama lain. Oleh karena itu, gambar seperti kerucut dan piramida dapat dengan mudah ditangkap oleh kamera, dan karenanya, gambar tersebut dapat dimanipulasi lebih lanjut sehingga dapat diproses dengan cepat oleh sistem ANN untuk mendapatkan hasil yang diinginkan;
- Panjang fokus juga dapat digunakan dalam pengaturan lanskap atau potret dengan cepat dan efisien;
- Konversi antar satuan ukuran fokus yang berbeda dapat dilakukan dengan cepat.

Menentukan Matriks Matematika Kamera

Untuk aplikasi Visi Komputer saat ini, banyak jenis matriks matematika yang dapat digunakan, tetapi yang paling umum digunakan oleh sistem ANN adalah "Matriks Kamera 3 x 4", dan ini direpresentasikan secara matematis sebagai berikut:

$$P = K\,[R|t].$$

Matriks matematika empat x empat juga dapat digunakan, dan ini dapat direpresentasikan secara matematis sebagai berikut:

$$P = [K, 0\hat{}T;\ 0\ 1] * [R, 0\hat{}T;\ t, 1] = \underline{K}e1$$

Di mana:

E = transformasi Geometri Euclidean 3 Dimensi;

K = "Matriks Kalibrasi".

Jika matriks matematika empat-kali-empat ini benar-benar digunakan, matriks tersebut dapat secara otomatis memetakan, secara langsung, "Koordinat Dunia" 3-Dimensi (yang dilambangkan sebagai $Pw = [Xw, Yw, Zw, 1]$) ke Koordinat "Dunia Nyata" (yang dilambangkan sebagai $Xs = [Xs, Ys, 1, d]$).

Menentukan Kedalaman Proyektif Kamera

Dalam contoh khusus ini, jika matriks matematika empat-kali-empat digunakan, baris terakhir (dan bahkan kolom) matriks tersebut dapat dipetakan ulang secara otomatis agar sesuai dengan apa yang dikenal sebagai "Kedalaman Proyektif" kamera yang dimaksud. Baris dan kolom terakhir dari matriks matematika empat-kali-empat dapat ditransformasikan dalam hal ini menggunakan rumus matematika berikut:

$$D = S3/z\,(n0 * Pw + c0)$$

Di mana:

Z = jarak numerik dari pusat kamera (dilambangkan sebagai "**C**") terhadap Sumbu Optiknya (dilambangkan sebagai "Z");

Pw = Bidang Referensi.

Perlu dicatat juga bahwa istilah "Kedalaman Proyektif" juga dapat disebut sebagai "Paralaks" atau bahkan "Bidang Ditambah Paralaks."

Invers dari pemetaan yang disebutkan di atas juga dapat terjadi jika diperlukan, dan ini dapat direpresentasikan secara matematis sebagai berikut:

$$\underline{P}w = \underline{P}\text{^} - 1 * Xs.$$

Teknik invers yang disebutkan di atas jarang digunakan untuk aplikasi dalam sistem ANN. Alasan utamanya adalah karena lebih dari satu bidang geometri harus digunakan dalam hal ini, sehingga menghabiskan lebih banyak daya pemrosesan dari dalam sistem ANN.

Bagaimana Gambar 3 Dimensi Dapat Ditransposisikan antara Dua Kamera atau Lebih

Salah satu pertanyaan kunci yang telah dibahas dalam bidang Visi Komputer adalah apakah gambar 3 Dimensi yang diambil dari posisi tertentu dalam satu kamera dapat ditransposisikan ke kamera lain (atau bahkan lebih dari dua) tanpa kehilangan integritas penuh gambar 3 Dimensi tersebut (tidak masalah apakah gambar tersebut statis atau dinamis). Hal ini kurang lebih telah dibahas dengan menggunakan, sekali lagi, matriks matematika empat kali empat, yang dalam kasus khusus ini dilambangkan sebagai "$\underline{P} = \underline{K}\,E$."

Transposisi ini dapat dilakukan dari satu kamera ke kamera berikutnya dengan cukup mudah menggunakan algoritma matematika berikut:

$$\underline{X}0 = \underline{k}0E0p = \underline{P}0p.$$

Selain itu, jika beberapa gambar 3-Dimensi harus ditransposisikan ke dua kamera atau lebih secara paralel, maka algoritma matematika berikut harus digunakan:

$$\underline{X}1 \ = \ \underline{k}1E1p \ = \ \underline{K}1E1E0\text{^} - 1K0\text{^} - 1\underline{x}0 \ = \ \underline{P}1\underline{P}0\text{^} - 1\underline{x}0 \ = \ M10\underline{x}0.$$

Dalam banyak kasus, variabel "Kedalaman Persepsi" tidak perlu dipastikan oleh sistem ANN untuk menghasilkan hasil yang diinginkan. Dengan demikian, metode lain untuk memindahkan gambar 3-Dimensi dari satu kamera ke kamera lain adalah dengan menggunakan algoritma matematika berikut:

$$\underline{X}1 \ = \ K1R1R0\text{^} - 1K0\text{^} - 1\underline{x}0 \ = \ K1R10K0\text{^} - 1\underline{x}0.$$

Dalam contoh khusus ini, gambar 3-Dimensi dapat dengan mudah ditransposisikan antara dua kamera atau lebih dengan menggunakan matriks matematika tiga-kali-tiga yang "Homografis". Namun, untuk menyelesaikan tugas spesifik ini, variabel-variabel berikut harus dipastikan:

- "Rasio Aspek" yang diketahui;
- Pusat Proyeksi;
- Derajat atau Level Rotasi;
- Properti Parameterisasi matriks matematika tiga kali tiga.

Bagaimana Gambar 3 Dimensi Dapat Diproyeksikan ke dalam Format yang Berpusat pada Objek

Kamera yang digunakan untuk menangkap gambar 3 Dimensi mungkin sering kali menggunakan lensa dengan panjang fokus yang sangat panjang. Meskipun hal ini tentu menguntungkan bagi sistem ANN, dalam hal statistik, memperkirakan panjang fokus spesifik ini dengan tepat dapat menjadi sangat rumit. Alasan utamanya adalah panjang fokus kamera yang dimaksud dan jarak numerik aktual gambar yang ditangkap sangat berkorelasi satu sama lain, sehingga sulit untuk membedakan keduanya.

Namun, hal ini dapat diselesaikan hingga tingkat tertentu dengan menggunakan algoritma matematika, dan dua di antaranya telah terbukti bermanfaat dalam penelitian ilmiah:

$$Xs \ = \ f\,[Rx \ ^{*} \ p \ + \ Tz]/[Rz \ ^{*} \ p \ + \ Tz] \ + \ Cz;$$
$$Ys \ = \ f\,[Ry \ ^{*} \ p \ + \ Ty]/[Rz \ ^{*} \ p \ + \ Tz] \ + \ Cy1.$$

Kedua algoritma di atas juga dapat dioptimalkan lebih lanjut sehingga diformulasikan sebagai berikut:

$$Xs \ = \ f\,[Rx \ ^{*} \ p \ + \ Tz]/[1 \ + \ N2R2 \ ^{*} \ P] \ + \ Cz;$$
$$Ys \ = \ f\,[Ry \ ^{*} \ p \ + \ Ty]/\,[1 \ + \ N2R2 \ ^{*} \ P] \ + \ Cy.$$

Kedua persamaan di atas memungkinkan pengukuran panjang fokus proyeksi jauh lebih akurat daripada sebelumnya. Dalam istilah teknis, hal ini juga dikenal secara khusus sebagai "Foreshortening".

7.3 MENGHITUNG DISTORSI PADA LENSA KAMERA

Perlu dicatat bahwa semua teori dan algoritma matematika yang disajikan hingga saat ini dalam bab ini pada dasarnya mengasumsikan bahwa pendekatan linear telah diambil untuk menangkap snapshot gambar yang dimaksud. Dengan kata lain, terdapat satu garis lurus yang dapat divisualisasikan dari lensa kamera ke gambar yang dimaksud, baik statis maupun dinamis. Namun, banyak kamera canggih saat ini yang digunakan oleh sistem ANN sering kali akan mengambil snapshot gambar melalui pendekatan "Kurvilinear". Pendekatan ini juga secara teknis dikenal sebagai "Distorsi Radial".

Akibatnya, proyeksi, seperti yang dijelaskan sebelumnya di subbagian terakhir, menjadi melengkung. Akibatnya, distorsi dapat terjadi pada snapshot gambar tertentu yang ditangkap. Hal ini dapat menyebabkan keburaman, dan karenanya, keluaran yang dihasilkan oleh sistem ANN dapat menjadi sangat miring. Namun sekali lagi, penggunaan matematika, terutama dalam persamaan semi-kuadrat, dapat digunakan untuk membantu mengurangi kesalahan ini. Kedua algoritma ini dapat direpresentasikan sebagai berikut:

$$Xc = [Rx * p + Tx]/[Rz * P + Tz];$$
$$Yc = [Ry * p + Ty]/[Rz * P + Tz].$$

Kedua algoritma di atas secara teknis juga dapat disebut sebagai "Model Distorsi Radial". Postulat dasarnya menyatakan bahwa citra yang akan ditangkap oleh sistem ANN secara teknis "dipindahkan" menjauh (dikenal sebagai "Efek Distorsi Barrel") atau mendekat (dikenal sebagai "Efek Distorsi Pincushion"). Kedua efek ini berkorelasi secara statistik dengan jumlah yang sama dari apa yang disebut "Jarak Radial". Untuk mempertimbangkan kedua efek ini lebih lanjut, Persamaan Polinomial dapat digunakan, yaitu sebagai berikut:

$$Xc = Xc * (1 + k1r\text{^}2c + k2r\text{^}4c);$$
$$Yc = Yc * (1 + k1r\text{^}2c + k2r\text{^}4c)$$

Di mana:
K1 dan K2 = Parameter Jarak Radial.

Setelah distorsi ini diatasi (terutama distorsi kabur, seperti yang baru saja diulas), koordinat geometris akhir piksel citra dapat dihitung sebagai berikut:

$$Xs = fX\text{^}rc + Cx;$$
$$Ys = fY\text{^}rc + Cy.$$

Namun terkadang, bergantung pada bagaimana sistem ANN menangkap snapshot gambar yang dimaksud, kedua algoritma matematika ini mungkin tidak cukup tepat untuk diterapkan. Oleh karena itu, teori analisis yang jauh lebih canggih, yang dikenal sebagai "Distorsi Tangensial" dan "Distorsi Decentering", dapat digunakan hingga taraf tertentu.

Selain itu, penggunaan lensa khusus yang disebut "Lensa Fisheye" juga dapat digunakan untuk mengatasi efek distorsi yang telah disebutkan sebelumnya. Untuk mencapai tugas khusus ini, proyektor "Equi-Distance" dapat digunakan dari jarak tertentu dari Sumbu Optik snapshot gambar yang akan diambil. Untuk melakukan ini, persamaan kuadrat lengkap harus digunakan. Namun, semua gambar 3-Dimensi ini (baik statis maupun dinamis) sebenarnya dianggap berukuran agak kecil. Alasan utamanya adalah gambar tersebut harus dapat diproses dengan mudah dan cepat oleh sistem ANN, secara berurutan.

Namun, gambar yang lebih besar pun dapat digunakan, meskipun hal ini dapat memperlambat waktu pemrosesan untuk menghasilkan hasil yang diinginkan dari sistem ANN. Untuk jenis gambar 3-Dimensi ini, penggunaan "Model Distorsi Parametrik" dan "Spline" akan diperlukan. Dalam kasus khusus ini, akan cukup sulit untuk menemukan titik pusat proyeksi yang tepat di sepanjang bidang geometris yang digunakan. Kita mungkin harus membangun secara matematis apa yang dikenal sebagai "Garis 3-Dimensi" yang harus berkorelasi secara statistik dengan setiap titik piksel yang direpresentasikan dalam gambar 3-Dimensi tersebut.

7.4 MEMBUAT CITRA FOTOMETRIK 3 DIMENSI

Pada titik ini di bab ini, kita telah mengasumsikan bahwa citra 2 Dimensi dan 3 Dimensi (baik statis maupun dinamis) hanya terdiri dari satu pita nilai matematika. Dengan kata lain, kita juga telah mengasumsikan bahwa citra 2 Dimensi dan 3 Dimensi ini biasanya berwarna hitam putih. Namun, penting untuk diingat bahwa meskipun warna-warna ini sangat cocok untuk sistem ANN karena tidak memerlukan daya pemrosesan yang besar, citra 2 Dimensi dan 3 Dimensi berwarna penuh juga dapat diterapkan dan digunakan.

Dengan demikian, keduanya akan memiliki apa yang dikenal sebagai "Nilai Intensitas" yang berbeda. Namun, penting juga untuk memastikan bahwa berbagai "Nilai Intensitas" ini berkorelasi secara statistik satu sama lain dalam beberapa cara. Di bagian ini, kita akan mengkaji beberapa variabel utama yang dapat memengaruhi korelasi statistik dari berbagai jenis "Nilai Intensitas" ini.

Variabel Pencahayaan

Sejujurnya, dan ini cukup jelas, kecuali terdapat pencahayaan yang cukup dari lingkungan eksternal, gambar 2-Dimensi atau 3-Dimensi berkualitas baik tidak dapat dihasilkan. Oleh karena itu, harus ada cahaya yang dapat menyinari gambar dari setidaknya dua sumber, bahkan lebih. Dengan demikian, sumber pencahayaan dapat dibagi lagi menjadi Sumber Cahaya Titik dan Sumber Cahaya Area, yang akan dibahas lebih rinci di sini.

1) Sumber Cahaya Titik:

   Jenis pencahayaan ini biasanya berasal dari satu sumber pada satu waktu. Jenis sumber pencahayaan ini juga memiliki tingkat intensitas tertentu dan memanfaatkan spektrum

warna yang dapat didistribusikan pada panjang gelombang yang berbeda. Ini dapat dilambangkan secara khusus sebagai "L(Y)."

2) Sumber Cahaya Area:

Dalam lingkungan seperti ini, intensitas cahaya yang berasal dari sumber tertentu sebenarnya berkurang seiring waktu ketika kuadrat jarak dari sumber cahaya tertentu untuk gambar yang dimaksud mulai diterangi. Alasan utamanya adalah cahaya yang diproyeksikan dari titik sumber sebenarnya terdistribusi secara parabola di atas permukaan gambar 2-Dimensi atau 3-Dimensi, baik ke atas maupun ke bawah. Hal ini dapat direpresentasikan secara matematis sebagai $Y = X\text{^}2$ atau $Y = -X\text{^}2$. Meskipun "Sumber Cahaya Titik" mungkin terdengar mudah dipahami secara teori, sebenarnya hal ini sulit dicapai di dunia nyata, biasanya ketika sistem ANN digunakan. Contoh umum dari hal ini dikenal secara spesifik sebagai "Penerangan Insiden", dan dapat direpresentasikan oleh persamaan matematika berikut:

$$L\ ^{*}\ (0/Y).$$

Algoritma di atas membuat asumsi ilmiah bahwa cahaya yang berasal dari titik sumbernya dapat merambat tanpa batas.

Efek Pantulan dan Bayangan Cahaya

Kita biasanya jarang memikirkan hal ini, tetapi ketika seberkas cahaya tertentu mengenai gambar 2-Dimensi atau 3-Dimensi, berkas cahaya tersebut sebenarnya tersebar di alam, dan seringkali dipantulkan kembali ke angkasa. Ada banyak teori yang telah ditetapkan untuk menjelaskan fenomena khusus ini, dan teori-teori tersebut akan diulas lebih rinci dalam subbagian ini.

1) Teorema Distribusi Pantulan Dua Arah:

Ini sebenarnya merupakan teori cahaya yang paling diterima secara luas saat ini. Pada dasarnya, teori ini menyatakan bahwa Fungsi Matematika 4-Dimensi dapat secara statistik menggambarkan intensitas setiap panjang gelombang yang masuk ke dalam apa yang dikenal sebagai "Arah Datang" (dilambangkan sebagai "V") sebenarnya dipantulkan kembali ke dalam apa yang dikenal sebagai "Arah Cahaya Pantulan" (dilambangkan sebagai "Vr"). Fungsi semacam ini dapat direpresentasikan secara matematis sebagai berikut:

$$fR\ (0/z1, 0/r, 0/r, Y).$$

Sangat menarik untuk dicatat bahwa teorema ini sebenarnya dapat dianggap sebagai resiprokal matematis, di mana peran spesifik "Vi" dan "Vr" dapat dipertukarkan satu sama lain. Yang juga sama pentingnya adalah fakta bahwa permukaan, baik dalam gambar 2-Dimensi maupun 3-Dimensi (baik statis maupun dinamis), dianggap bersifat "Isotropik". Dengan kata lain, tidak ada arah tertentu dari mana cahaya harus

ditransmisikan. Sifat "Isotopik" ini dapat direpresentasikan secara matematis sebagai berikut:

$$\mathrm{Fr}\,(\mathrm{o/I}, 0/\mathrm{r}\,|0/\mathrm{r}\; -\; 0/\mathrm{I}; \mathrm{Y});$$

Atau juga sebagai:

$$\mathrm{Fr}\,(\underline{\mathrm{V}}1,\ \underline{\mathrm{V}}\mathrm{r},\ \mathrm{N},\ \underline{\mathrm{Y}}).$$

Akhirnya, untuk menghitung secara spesifik jumlah cahaya yang dipantulkan dari gambar 2-Dimensi atau 3-Dimensi, algoritma matematika berikut digunakan:

$$\mathrm{Lr}\,(\underline{\mathrm{V}}\mathrm{r};\ \mathrm{Y})\ =\ \mathrm{F}\,(\mathrm{Li}(\underline{\mathrm{V}}\mathrm{i};\ \mathrm{Y})\ \mathrm{Fr}(\underline{\mathrm{V}}\mathrm{i}, \underline{\mathrm{V}}\mathrm{r}, \underline{\mathrm{N}}, \mathrm{Y})\ \mathrm{COS}\hat{}\ +\ 0/\mathrm{I}, \mathrm{d}\underline{\mathrm{V}}\mathrm{i}.$$

2) <u>Komponen Difusi dari Teorema Distribusi Reflektansi Dua Arah:</u>
Ini sebenarnya merupakan subkomponen spesifik dari teorema yang disebutkan di atas, dan dapat juga disebut sebagai Sifat Refleksi "Lambertian" atau "Matte". Komponen ini sebenarnya mengasumsikan bahwa sumber cahaya dan cahaya yang dipancarkannya terdistribusi secara statistik dalam pola yang seragam di seluruh gambar 2-Dimensi atau gambar 3-Dimensi yang dimaksud. Komponen inilah yang mengarah pada apa yang dikenal sebagai "Shading". Pada dasarnya, ini adalah cahaya tak berkilau yang ditransmisikan ke objek (baik berupa gambar 2-Dimensi maupun gambar 3-Dimensi). Dalam hal ini, cahaya tersebut diserap dan dipantulkan kembali. Penting untuk diingat bahwa ketika cahaya yang berasal dari titik sumbernya disebarkan secara seragam, teorema yang disebutkan di atas menjadi konstan, dan dapat direpresentasikan oleh algoritma matematika berikut:

$$\mathrm{Fd}\,(\underline{\mathrm{V}}\mathrm{i}, \underline{\mathrm{V}}\mathrm{r}, \underline{\mathrm{N}}, \mathrm{Y})\ =\ \mathrm{Fd}(\mathrm{Y}).$$

Untuk memperhitungkan "Efek Bayangan" seperti yang baru saja dijelaskan, algoritma matematika berikut juga digunakan:

$$\mathrm{Ld}\,(\mathrm{Vr};\ \mathrm{Y})\ =\ \sum \mathrm{Li}(\mathrm{Y})\mathrm{Fd}(\mathrm{Y})\ \mathrm{COS}\hat{}\ +\ 0/\mathrm{I}\ =\ \sum \mathrm{Li}(\mathrm{Y})\mathrm{Fd}(\mathrm{Y})\ ^{*}\ [\underline{\mathrm{V}}\mathrm{i}\ ^{*}\ \underline{\mathrm{n}}]\hat{}\ +.$$

3) <u>Komponen Difusi dari Teorema Distribusi Reflektansi Dua Arah:</u>
Ini dianggap sebagai komponen utama kedua dari teori yang disebutkan di atas, dan sebenarnya memperhitungkan secara spesifik pantulan cahaya yang bersifat "Spekuler". Dengan kata lain, cahaya tersebut tampak "Berkilau" ketika ditransmisikan ke gambar 2-Dimensi atau 3-Dimensi yang dimaksud. Ini secara teknis dikenal sebagai "Pencahayaan Insiden", dan dapat diputar 180 derajat pada objek yang dimaksud. Ini dihitung secara matematis sebagai berikut:

$$\underline{Si} = v||\,0\,vT = (2\underline{nn}\hat{}T - I) * Vi.$$

Dengan demikian, jumlah cahaya yang ditransmisikan dalam hal ini terutama bergantung pada variabel-variabel berikut:

- Sudut Datang (dilambangkan sebagai $0/ = COS\hat{} - 1 * (Vr * Si)$;
- Arah Pandang (dilambangkan sebagai "$\underline{V}r$");
- Arah Spekular (dilambangkan sebagai "$\underline{S}i$").

4) Teori Bayangan Phong:

Teori khusus ini menyatakan bahwa aspek Difusi dan Spekular dari cahaya yang dipantulkan secara teknis dapat disebut sebagai "Iluminasi Ambien". Ini mengacu pada fakta bahwa cahaya yang disinari, baik pada gambar 2 Dimensi maupun gambar 3 Dimensi, dapat tersebar merata, tetapi sifatnya "terdifusi". Dalam teori ini, warna cahaya menjadi faktor yang sangat penting, yang selanjutnya memperhitungkan derajat spesifik dari apa yang dikenal sebagai "Iluminasi Ambien". Hal ini dapat direpresentasikan secara matematis sebagai berikut:

$$Fa(Y) = Ka * [(Y)\,La(Y)].$$

Teori Phong juga dapat dinyatakan secara matematis sebagai berikut:

$$Lr(Vr; Y) = Ka(Y)La(Y) + Kd(Y)\sum Li(Y) *[\underline{V}i * \underline{n}]\hat{} + + Kz(Y)\sum Li(Y) * (\underline{V}r * Si)\hat{}k.$$

Penting untuk dicatat bahwa baik Warna Ambien maupun Warna Difusi, yang terdistribusi di seluruh gambar 2-Dimensi atau 3-Dimensi (dilambangkan sebagai "Ka(Y)" dan "Kd(Y)") dianggap sama secara harfiah dalam desain fitur. Selain itu, Iluminasi Ambien yang umum terdapat memiliki jenis bayangan warna yang berbeda dari sumber cahaya tempat ia diproyeksikan. Selain itu, "Komponen Difusi" dari teori khusus ini sangat bergantung pada Sudut Datang pita sinar cahaya yang masuk (yang secara khusus dilambangkan sebagai "Vi"). Namun, ini bukan satu-satunya teori khusus yang digunakan dalam hal ini. Faktanya, model-model canggih lain yang saat ini digunakan dalam Grafika Komputer biasanya menggantikan teori ini.

5) Model Refleksi Dikromatik:

Model ini juga dikenal sebagai "Model Refleksi Torrance dan Sparrow". Teori ini hanya menyatakan bahwa semua pencahayaan berwarna yang digunakan untuk menerangi gambar 2-Dimensi maupun 3-Dimensi (yang dapat berupa statis maupun dinamis) tersebar secara merata, dan biasanya berasal dari satu sumber cahaya saja, dan terdiri dari dua algoritma matematika, yaitu sebagai berikut:

$$Lr(Vr;\ Y) = Li\,(\underline{V}r, \underline{V}i, \underline{N}, Y) + Lb\,(\underline{V}r, \underline{V}i, \underline{N}, Y) = Ci(Y)m1\,(\underline{V}r, \underline{V}i, \underline{N},) + Cb(Y)Mb\,(\underline{V}r, \underline{V}i, \underline{N},)$$

Perlu dicatat bahwa teori khusus ini telah digunakan dalam Visi Komputer untuk memisahkan objek berwarna yang terletak di gambar 2- Dimensi maupun 3-Dimensi di mana terdapat variasi matematika yang sangat besar dalam jumlah bayangan yang disinari ke objek tersebut.

6) Teori Iluminasi Global:

Sebagai tinjauan, teori-teori di atas berasumsi bahwa aliran cahaya diproyeksikan dari titik sumber aslinya, dan akan memantul dari gambar 2-Dimensi atau 3-Dimensi dengan intensitas yang berubah-ubah, sehingga akan diproyeksikan kembali ke kamera dalam lintasan matematis yang terbalik. Namun, teori-teori yang diulas ini berasumsi bahwa hal ini hanya terjadi sekali. Kenyataannya, urutan ini dapat terjadi berkali-kali, melalui banyak iterasi, dalam sebuah siklus berurutan. Dalam hal ini, terdapat dua metodologi khusus yang telah mencoba mengatasi fenomena unik ini. Metodologi-metodologi tersebut adalah sebagai berikut:

- Ray Tracing:

  Ini juga secara teknis dikenal sebagai "Path Tracing." Metodologi ini berasumsi bahwa sinar-sinar terpisah dari kamera akan memantul kembali berkali-kali dari gambar 2-Dimensi atau 3-Dimensi ke sumber cahaya. Lebih lanjut, algoritma yang menyusun metodologi khusus ini berasumsi bahwa "Kontribusi Primer" dapat dihitung secara matematis dengan menggunakan berbagai bentuk persamaan Light Shading. Sinar cahaya tambahan yang dianggap bersifat suplementer juga dapat digunakan di sini.

- Radiositas:

  Prinsip yang sama juga berlaku di sini, tetapi alih-alih menggunakan lampu berwarna, digunakan jenis pencahayaan khusus lain, yang disebut "Uniform Albedo Simple Geometry Illuminator". Selain itu, nilai matematika yang terkait dengan gambar 2 Dimensi atau 3 Dimensi berkorelasi secara statistik satu sama lain. Dengan demikian, di antara cahaya yang ditangkap secara fisik terdapat apa yang dikenal sebagai "Faktor Bentuk", yang merupakan fungsi dari orientasi vektor dan berbagai jenis properti pantulan lainnya. Dalam metodologi ini, hal ini dapat dilambangkan sebagai "$1/r^2$".

Namun, salah satu kelemahan utama dari metodologi khusus ini adalah tidak memperhitungkan apa yang dikenal sebagai "Efek Medan Dekat", seperti kurangnya cahaya yang masuk ke dalam bayangan kecil di dalam gambar 2 Dimensi atau 3 Dimensi, atau bahkan kurangnya pencahayaan sekitar.

Faktanya, berbagai upaya telah dilakukan untuk menggabungkan metodologi yang disebutkan di atas menjadi satu kesatuan yang kohesif. Keuntungan utamanya adalah dapat digunakannya jenis sumber pencahayaan tambahan.

7.5 PENTINGNYA OPTIK

Salah satu aspek kunci dalam Visi Komputer sebagaimana digunakan oleh sistem ANN adalah apa yang dikenal sebagai "Optik". Apa sebenarnya Optik itu? Secara teknis, optik dapat didefinisikan sebagai berikut: *Optik klasik dibagi menjadi dua cabang utama: optik geometris (atau sinar) dan optik fisik (atau gelombang). Dalam optik geometris, cahaya dianggap merambat dalam garis lurus, sedangkan dalam optik fisik, cahaya dianggap sebagai gelombang elektromagnetik.*

Sebagaimana dinyatakan dalam definisi di atas, terdapat dua jenis utama optik yang dapat digunakan dalam Visi Komputer, yaitu sebagai berikut:

- Optik Geometris;
- Optik Fisik.

Secara lebih sederhana, untuk keperluan bab ini, Optik dapat dianggap sebagai cahaya yang harus melewati lensa kamera sebelum mencapai sensor kamera. Atau lebih sederhana lagi, Optik dapat dianggap sebagai lubang jarum kecil yang akan memproyeksikan semua sinar cahaya dari semua sumber cahaya ke satu pusat utama, yang kemudian dapat disinari ke gambar 2 Dimensi atau 3 Dimensi (yang bersifat statis maupun dinamis).

Namun tentu saja, skenario di atas seperti yang baru saja digambarkan bisa menjadi jauh lebih kompleks; banyak hal bergantung pada persyaratan yang ditetapkan oleh sistem ANN. Misalnya, beberapa variabel tambahan yang perlu dipertimbangkan adalah sebagai berikut:

- Properti fokus kamera;
- Laju pencahayaan kamera;
- Vignetting;
- Aberasi.

Dalam hal ini, pengaturan umum untuk penggunaan Optik akan memastikan adanya apa yang dikenal sebagai "Lensa Tipis" yang pada dasarnya hanya terdiri dari satu lembar kaca yang memiliki fitur parabola sangat rendah di kedua sisinya. Terdapat teorema khusus untuk hal ini, yang secara teknis dikenal sebagai "Hukum Lensa". Teorema ini secara khusus menetapkan hubungan matematis antara jarak gambar 2 Dimensi atau 3 Dimensi (yang dapat dilambangkan sebagai "Zo"), serta jarak spesifik dari belakang lensa tempat gambar 2 Dimensi atau 3 Dimensi tersebut ditangkap. Hal ini dapat direpresentasikan secara matematis sebagai berikut:

$$(1/z0) + (1/Zt) = (1/f)$$

Di mana:

$F =$ Panjang Fokus.

Ada pula konsep penting lain yang berkaitan dengan Optik, dan konsep ini secara khusus dikenal sebagai "Kedalaman Bidang" (Depth of Field). Ini adalah fungsi matematis dari

Jarak Fokus yang terdapat pada "Diameter Bukaan", yang dilambangkan sebagai "d". Hal ini dapat direpresentasikan secara matematis sebagai berikut:

$$f/\# \ = \ N \ = \ f/d$$

Di mana:

$f \ =$ Panjang Fokus;

$d \ =$ Diameter Geometris Bukaan kamera.

Perlu dicatat bahwa nilai "f" yang disebutkan di atas direpresentasikan sebagai serangkaian bilangan bulat, seperti berikut:

$$f/1.0, f/2.0/, f/3.2, f/4.8, \ldots$$

Representasi numerik yang dijelaskan di atas sebenarnya merupakan rangkaian iterasi, yang didasarkan pada "Full Stop". Misalnya, karena $f/1.0$ diproses sepenuhnya oleh sistem ANN, sistem akan berhenti sejenak selama satu atau dua detik agar dapat memproses nilai "f" berikutnya, yang dalam hal ini adalah $f/2.0$. Namun, salah satu kelemahan utama penggunaan optik dalam hal ini adalah lensa biasanya sangat tipis, dan hal ini dapat menyebabkan fenomena yang dikenal sebagai "Aberasi Kromatik", yang akan dibahas lebih detail di bagian selanjutnya.

Efek Aberasi Kromatik

Aberasi Kromatik berkaitan dengan apa yang dikenal sebagai "Indeks Refraksi". Ini terjadi ketika cahaya berwarna yang berasal dari berbagai sumbernya akhirnya terfokus pada jarak yang sedikit berbeda dari nilai target yang dituju. Varians ini dapat diukur dengan metrik yang dikenal sebagai "Aberasi Kromatik Transversal", dan ini dapat dimodelkan berdasarkan per warna, tergantung warna mana yang ditransmisikan untuk menerangi gambar 2-Dimensi atau 3-Dimensi.

Keburaman apa pun yang dapat tercipta dalam iluminasi ini secara teknis dikenal sebagai "Aberasi Kromatik Longitudinal". Aberasi ini menimbulkan kerugian besar karena keburaman jenis ini biasanya tidak dapat dihilangkan setelah diproyeksikan ke gambar 2-Dimensi maupun 3-Dimensi. Untuk mengurangi efek semacam ini semaksimal mungkin, lensa kamera menggunakan teknologi yang dikenal sebagai "Lensa Majemuk". Lensa-lensa ini terbuat dari berbagai elemen berbasis kaca.

Alih-alih hanya memiliki apa yang dikenal sebagai "Titik Nodal Tunggal" (yang dapat dilambangkan sebagai "P"), lensa jenis ini memanfaatkan apa yang dikenal sebagai "Panel Nodal Depan". Di sinilah semua berkas cahaya yang digunakan untuk menerangi gambar 2-Dimensi atau 3-Dimensi masuk ke satu lokasi sentral dari dalam kamera, lalu keluar melalui "Titik Nodal Belakang" dalam perjalanannya menuju sensor. Perlu dicatat bahwa ketika mencoba mengkalibrasi kamera, hanya Titik spesifik inilah yang menjadi perhatian utama.

Namun, tidak semua lensa kamera memiliki "Titik Nodal" khusus seperti ini. Contoh tipikal dari hal ini adalah Lensa Fisheye, seperti yang telah diulas sebelumnya dalam bab ini. Untuk mengatasi hambatan semacam ini, fungsi matematika khusus sering kali dibuat agar berbagai koordinat piksel dan efek 3-Dimensi dapat dikorelasikan secara statistik satu sama lain.

Sifat-Sifat Vignetting

Sifat lain dari Aberasi Kromatik adalah "Vignetting". Berdasarkan prinsip ilmiahnya, ini adalah ketika kecerahan sinar cahaya yang disinari, baik pada gambar 2-Dimensi maupun 3-Dimensi, entah karena alasan apa, menuju ujung luar gambar yang dimaksud. Dalam hal ini, terdapat dua jenis Vignetting, dan keduanya diulas sebagai berikut:

1) Vignetting Alami:

   Ini terjadi ketika "Foreshortening" terjadi pada permukaan gambar 2-Dimensi maupun 3-Dimensi, atau piksel apa pun yang terdapat di dalamnya. Hal ini dapat direpresentasikan secara matematis sebagai berikut:

$$00COSY/r\text{^}20\ TT\ *\ (d/2)\text{^}2\ COS\ A\ =\ 00\ *\ (TT/4)\ *\ (d\text{^}2)/z\text{^}2COS\text{^}4\ A.$$

   Cahaya apa pun yang ditransmisikan ke gambar yang dimaksud juga dapat direpresentasikan secara matematis sebagai berikut:

$$00/0i\ =\ (z\text{^}2/z\text{^}2i).$$

   Akhirnya, hubungan matematis antara jumlah cahaya yang ditransmisikan ke piksel gambar 2-Dimensi atau 3-Dimensi (dilambangkan sebagai "i"), diameter geometris Aperture kamera (dilambangkan sebagai "d"), jarak fokus (dilambangkan sebagai Zi~f), dan setiap sudut yang menyimpang (dilambangkan sebagai "A") dapat direpresentasikan secara matematis sebagai berikut:

$$Oo\ *\ (TT/4)\ *\ (d\text{^}2/z\text{^}2o)COS\text{^}4A\ =\ Oo\ *\ (TT/4)\ *\ (d\text{^}2/z\text{^}2o)COS\text{^}4A\ =\ (diTT/4)\ *\ (d/f)\text{^}2\ COS\text{^}4A.$$

   Selain itu, "Hubungan Radiometrik Fundamental" yang terbentuk dari "Cahaya Radiansi" (dilambangkan sebagai "L") dan "Cahaya Iradiansi" (dilambangkan sebagai "E") juga dapat direpresentasikan secara matematis sebagai berikut:

$$E\ =\ L(TT/4)\ *\ (d/f)\ COS\text{^}4\ A1.$$

2) Vignetting Mekanis:

   Hal ini secara teknis juga disebut sebagai "Oklusi Internal", dan terjadi ketika elemen-elemen lensa kamera tidak dapat menyerap semua sinar cahaya yang ditransmisikan

dari sumber cahaya. Namun, hal ini dapat dengan mudah diperbaiki karena panjang Apertur Kamera dapat dikurangi.

7.6 PROPERTI KAMERA DIGITAL

Bagian ini menjelaskan konstruksi dasar bagaimana kamera digital dapat digunakan bersama sistem ANN untuk menghasilkan hasil yang diinginkan. Pertama, cahaya yang dipicu dari berbagai sumber cahaya biasanya dikumpulkan oleh apa yang dikenal sebagai "Area Penginderaan Aktif", yang dapat bertahan selama periode waktu pemaparan gambar 2-Dimensi atau 3-Dimensi. Hal ini biasanya dilakukan dalam sepersekian detik, dan kemudian, cahaya tersebut ditransmisikan ke apa yang dikenal sebagai "Penguat Penginderaan". Teknologi di balik ini adalah "Charged Couple Device" (juga dikenal sebagai CCD), dan oksida logam yang ada di dalamnya, yang seringkali berbasis silikon (juga dikenal sebagai "CMOS").

Dari titik ini, foton-foton tersebut kemudian ditumpuk satu sama lain selama periode pencahayaan gambar 2-Dimensi atau 3-Dimensi yang dimaksud. Kemudian, dalam apa yang dikenal sebagai "Fase Transfer", muatan fotonik ini ditransfer lagi ke apa yang dikenal sebagai "Penguat Sensor". Sesuai namanya, sinyal-sinyal ini diperkuat dan, dari sana, dikirim ke apa yang dikenal sebagai "Konverter Analog ke Digital", yang juga dikenal sebagai "ADC".

Perlu dicatat di sini bahwa pada CCD generasi lama, gambar seringkali mengalami fenomena yang disebut "Blooming". Hal ini terjadi ketika piksel dalam gambar 2-Dimensi atau 3-Dimensi ditransfer ke piksel lain yang berdekatan atau sejajar dengannya. Namun, pada CCD versi terbaru, fenomena ini sangat diringankan dengan penggunaan "Palung". Di sinilah muatan fotonik tambahan dapat ditransfer dengan aman ke area lain pada kamera digital yang sedang digunakan oleh sistem ANN.

Ada juga faktor-faktor lain yang dapat sangat memengaruhi daya pemrosesan dan kinerja CCD, yaitu:

- Kecepatan rana;
- Jarak pengambilan sampel;
- Faktor pengisian;
- Ukuran Unit Pemrosesan Pusat (CPU) di dalam kamera digital;
- Resolusi dari konverter analog ke digital;
- Penguatan analog;
- Derau sensor.

Semua hal di atas akan dibahas di subbagian berikutnya.

Kecepatan Rana

Fungsi khusus kamera digital ini memiliki kendali langsung atas jumlah cahaya yang masuk ke dalam kamera digital, dan juga berdampak langsung pada apakah gambar 2 Dimensi atau 3 Dimensi akan kurang terekspos atau bahkan terlalu terekspos. Untuk gambar 2 Dimensi atau 3 Dimensi yang dinamis, kecepatan rana juga dapat menjadi faktor penting dalam menentukan seberapa banyak "Motion Blur" yang akan dihasilkan pada gambar. Aturan umum di sini adalah bahwa kecepatan rana yang lebih tinggi secara proporsional dapat memungkinkan analisis forensik selanjutnya, baik untuk gambar 2 Dimensi maupun 3 Dimensi.

Sampling Pitch

Metrik ini dianggap sebagai jarak fisik aktual antara sel sensor dan chip pencitraan yang terletak di dalam kamera digital itu sendiri. Aturan praktis yang baik di sini adalah bahwa tingkat sampling pitch yang lebih tinggi biasanya akan menghasilkan resolusi gambar 2 Dimensi maupun 3 Dimensi yang jauh lebih baik. Kebalikannya juga berlaku, yaitu laju pitch yang lebih kecil berarti hanya area gambar yang lebih kecil yang akan ditangkap, sehingga mungkin terdapat objek asing di dalamnya.

Faktor Isi

Ini dapat dianggap sebagai "Area Penginderaan" kamera digital yang sebenarnya. Metrik ini direpresentasikan sebagai pecahan numerik, dan semakin tinggi laju pengisian, akan ada lebih banyak cahaya yang terpancar, sehingga hasil akhirnya adalah snapshot gambar 2-Dimensi atau 3-Dimensi yang jauh lebih kuat akan ditangkap.

Ukuran Unit Pemrosesan Pusat (CPU)

Ada banyak CPU berukuran mini yang tersedia untuk kamera digital yang digunakan oleh sistem ANN, mulai dari sepersekian inci. Namun, untuk hasil yang paling kuat, sangat disarankan untuk menggunakan CPU berukuran lebih besar. Kerugian utama dari hal ini adalah semakin besar CPU, semakin besar pula probabilitas statistik bahwa CPU tersebut merupakan chip yang lebih rusak.

Penguatan Analog

Pada kamera digital lama, penguatan analog diperkuat oleh apa yang dikenal sebagai "Penguat Sense". Namun, pada kamera digital masa kini yang menggunakan sistem ANN, "Penguat Sense" telah digantikan oleh "Pengaturan ISO". Ini merupakan proses otomatis, di mana tingkat penguatan analog yang lebih tinggi akan memungkinkan kamera digital menghasilkan foto 2 Dimensi atau 3 Dimensi dengan kualitas yang jauh lebih baik dalam kondisi pencahayaan yang sangat buruk atau di bawah standar yang mungkin ada di lingkungan eksternal.

Noise Sensor

Selama seluruh siklus hidup kamera digital yang mengambil foto 2 Dimensi atau 3 Dimensi, mungkin terdapat banyak noise "asing" yang dapat muncul selama keseluruhan proses ini. Jenis-jenis "noise" ini dapat dibagi lagi ke dalam kategori berikut:

- Noise pola tetap;
- Noise arus gelap;
- Noise bidikan;
- Noise amplifier;
- Noise kuantisasi.

Penting untuk dicatat pada titik ini bahwa dengan kelima faktor di atas, sumber pencahayaan yang digunakan biasanya dapat memengaruhi gambar 2 Dimensi atau 3 Dimensi yang saat ini digunakan oleh sistem ANN. Namun, masalah "noise" ini dapat diatasi dengan memanfaatkan Model Distribusi Poisson yang berbasis statistik.

Resolusi ADC

Ini adalah akronim yang merupakan singkatan dari "Konversi Analog ke Digital". Ini dapat dianggap sebagai salah satu langkah terakhir dalam pemrosesan gambar 2 Dimensi atau 3 Dimensi sebelum ditransmisikan ke sistem ANN untuk menghitung keluaran yang diinginkan. Ada dua faktor lain yang menjadi perhatian utama di sini, yaitu sebagai berikut:

- Resolusi: Ini adalah metrik yang mencerminkan ukuran byte total gambar 2 Dimensi atau 3 Dimensi;
- Tingkat "Noise" keseluruhan dari gambar-gambar ini, seperti yang baru saja diulas di subbagian sebelumnya.

Untuk yang pertama, disarankan agar gambar 2-Dimensi atau 3-Dimensi tidak lebih dari 16 bit agar daya pemrosesan sistem ANN dapat dioptimalkan dan tidak terbebani melebihi batas desainnya.

Pemrosesan Pasca-Digital

Setelah semua langkah di subbagian terakhir selesai, kamera digital dapat mengambil snapshot gambar 2-Dimensi atau 3-Dimensi, menyempurnakannya lebih lanjut, dan mengompresnya lebih lanjut agar gambar dapat digunakan dengan mudah oleh sistem ANN. Beberapa teknik yang dapat digunakan di sini meliputi:

- Demosaicing Array Filter Warna (juga dikenal sebagai "CFA");
- Pengaturan berbagai Titik Putih;
- Menghitung Fungsi Gamma dari gambar 2-Dimensi atau 3-Dimensi yang hanya bersifat dinamis.

7.7 PENGAMBILAN SAMPEL CITRA 2 DIMENSI ATAU 3 DIMENSI

Sesuai dengan bagian ini, citra 2 Dimensi atau 3 Dimensi yang akan diproses oleh sistem ANN harus terlebih dahulu diambil sampelnya untuk melihat snapshot mana yang paling efektif dalam menghitung keluaran yang diinginkan. Konsep ini juga dapat disebut sebagai "Aliasing". Terdapat algoritma matematika langsung untuk membantu proses ini, yang dapat disebut sebagai "Teorema Pengambilan Sampel Shannon". Teori ini menghitung jumlah pengambilan sampel minimum yang diperlukan untuk menyusun kembali sinyal cahaya yang cukup robust. Istilah "robust" dapat didefinisikan sebagai setidaknya dua kali lebih tinggi (2X) dari frekuensi tertinggi yang sebenarnya dihasilkan oleh kamera digital.

Hal ini dapat direpresentasikan secara matematis sebagai berikut:

$$Fs > 2Fmax.$$

Dengan demikian, dalam hal ini, tingkat frekuensi tertinggi juga dapat disebut sebagai "Frekuensi Nyquist". “Laju Nyquist” juga dapat didefinisikan sebagai minimum dari kebalikan frekuensi yang dimaksud, dan dapat direpresentasikan secara matematis sebagai berikut:

$$Rs = 1/Fn.$$

Pada titik ini, kita bisa saja bertanya, apa gunanya terlibat dalam proses pengambilan sampel sejak awal? Tujuan utamanya adalah mengurangi jumlah tingkat frekuensi yang ditransmisikan ke gambar 2-Dimensi atau 3-Dimensi, sehingga dapat diproses lebih mudah oleh sistem ANN. Dalam hal ini, metrik kunci lain yang dapat digunakan adalah yang dikenal sebagai "Fungsi Sebaran Titik". Hal ini mengasumsikan bahwa tingkat respons piksel yang tertanam dalam gambar snapshot 2-Dimensi atau 3-Dimensi sebenarnya dapat digunakan untuk menunjukkan sumber cahaya optimal yang seharusnya digunakan.

"Fungsi Sebaran Titik" (juga dikenal sebagai "PSF") adalah penjumlahan matematis dari keburaman yang ada, dan "Area Integrasi" yang sebenarnya dapat diciptakan oleh sensor chip kamera digital yang digunakan untuk sistem ANN. Dengan kata lain, jika faktor pengisian diketahui (seperti yang dijelaskan sebelumnya), PSF juga dapat dihitung. Selain itu, "Fungsi Transfer Modular" dapat dihitung untuk memastikan secara statistik berapa banyak pengambilan sampel yang benar-benar dibutuhkan sebelum cuplikan gambar 2-Dimensi atau 3-Dimensi dimasukkan ke dalam sistem ANN.

Perlu dicatat bahwa teknik pengambilan sampel yang baru saja dijelaskan dapat digunakan untuk tujuan lain selain menentukan gambar 2-Dimensi atau 3-Dimensi mana yang paling sesuai untuk sistem ANN. Tujuan-tujuan ini meliputi:

- Pengambilan sampel ulang;
- Pengambilan sampel ulang;
- Penurunan sampel;
- Jenis aplikasi Pemrosesan Gambar lainnya.

Pentingnya Warna Dalam Gambar 2-Dimensi Atau 3-Dimensi

Sejauh ini dalam bab ini, konsep tentang bagaimana berbagai fungsi pencahayaan dan permukaan yang digunakan untuk mengambil cuplikan gambar 2-Dimensi dan 3-Dimensi telah diulas secara mendalam. Misalnya, ketika cahaya datang dari berbagai titik sumber proyeksinya, sinar-sinar ini sebenarnya dipecah menjadi berbagai warna spektrum: merah, hijau, dan biru, yang juga dikenal sebagai "RGB". Ada juga warna-warna lain, seperti cyan, magenta, dan kuning, atau "CYMK". Warna-warna ini juga dikenal sebagai "Warna Subtraktif". Warna-warna lain yang telah dijelaskan sebelumnya dikenal sebagai "Warna Primer Aditif". Warna-warna ini sebenarnya ditambahkan bersama untuk menghasilkan rezim warna CYMK. Selain itu, berbagai warna ini dapat dikombinasikan untuk menghasilkan jenis warna lain juga. Namun, penting untuk diingat bahwa warna-warna ini tidak bercampur atau tergabung secara otomatis. Sebaliknya, warna-warna ini tampak tercampur karena cara Korteks Visual di otak manusia diciptakan.

Semua ini adalah hasil dari apa yang dikenal sebagai sifat "Tri-Stimulus" dari sistem penglihatan kita, seperti yang baru saja dijelaskan. Namun, bila semua ini diterapkan pada bidang Visi Komputer, Anda akan ingin menggunakan sebanyak mungkin warna panjang gelombang yang berbeda dan beraneka ragam agar dapat menghasilkan potret paling kuat baik pada gambar 2-Dimensi maupun gambar 3-Dimensi.

Teorema CIE, RGB, dan XYZ

Ketiga akronim terpisah ini secara teknis juga dikenal sebagai "Teori Trikromatik Persepsi". Dalam hal ini, upaya dilakukan untuk menghasilkan semua warna monokromatik sebagai tiga warna primer agar sistem ANN dapat digunakan secara efisien dan optimal. Teori spesifik ini dapat direpresentasikan secara matematis sebagai berikut:

$$[\mathrm{X,Y,Z}] = 1/0{,}17697\,[(0{,}49, 0{,}17697, 0{,}000)\ ^{*}\ (0{,}31, 0{,}81240, 0{,}01)\ ^{*}\ (0{,}20, 0{,}01063, 0{,}99)]\ ^{*}\ [\mathrm{R,G,B}].$$

Koordinat warna spesifik dari teorema ini dapat direpresentasikan secara matematis sebagai berikut:

$$\mathrm{X} = (\mathrm{X}/\mathrm{X}+\mathrm{Y}+\mathrm{Z}), \mathrm{y} = (\mathrm{Y}/\mathrm{X}+\mathrm{Y}+\mathrm{Z}), \mathrm{z} = (\mathrm{Z}/\mathrm{X}+\mathrm{Y}+\mathrm{Z})$$

Semua ini menghasilkan nilai 1.

Pentingnya Rezim Warna L*a*b untuk Citra 2 Dimensi dan 3 Dimensi

Meskipun subbagian terakhir dari bab ini menekankan pentingnya bagaimana korteks visual manusia dapat secara harfiah memisahkan warna berbasis luminansi dari warna berbasis kromatik, teori-teori yang baru saja diuraikan biasanya tidak mencakup pertanyaan mendasar tentang bagaimana korteks visual dapat benar-benar memeriksa perbedaan halus dan kecil dalam berbagai rezim warna yang baru saja dibahas di subbagian terakhir bab ini.

Untuk mengatasi efek ini (karena Visi Komputer mencoba mereplikasi seluruh sistem visual manusia), sebuah konsep yang dikenal sebagai "Rezim Warna L*a*b" telah dirumuskan. Konsep ini juga disebut sebagai "CIELAB". Hal ini dapat direpresentasikan secara matematis sebagai berikut:

$$\mathrm{L}^{*} = 116\mathrm{f}\ ^{*}\ (\mathrm{Y}/\mathrm{Yn}).$$

Persamaan di atas menghitung komponen "L^*". Algoritma matematika berikut menghitung komponen "a^*" dan "b^*":

$$A^{*} = 500\ [\mathrm{f}(\mathrm{X}/\mathrm{Xn}) - \mathrm{f}(\mathrm{Y}/\mathrm{Yn})];\ b^{*} = 200[\mathrm{f}(\mathrm{Y}/\mathrm{Yn} - \mathrm{f}(\mathrm{Z}/\mathrm{Zn}].$$

7.8 PENTINGNYA KAMERA BERBASIS WARNA DALAM VISI KOMPUTER

Sejauh ini, kita telah mengulas dalam bab ini, khususnya di beberapa subbagian terakhir, bagaimana berbagai warna dapat diterapkan. Namun, terlepas dari semua ini, masih ada satu rezim warna yang belum diteliti—"RGB." Warna-warna spesifik ini adalah merah, biru, dan hijau. Representasi matematis untuk masing-masing warna spektrum ini dapat didefinisikan lebih lanjut sebagai berikut:

$$\mathrm{R\ (Red)} = \sum \mathrm{L(Y)Sr(Y)dYr};$$
$$\mathrm{G\ (Green)} = \sum \mathrm{L(Y)Sg(Y)dYr};$$

$$B\ (Blue) = \sum L(Y)\ Sb(Y)dYr.$$

Di mana:

$L(Y)$ = Spektrum masuk dari salah satu warna yang disebutkan di atas pada setiap lokasi spesifik dari 2-Dimensi atau 3-Dimensi gambar;

$\{Sr(Y), Sg(Y), Sb(Y)\}$ = "Sensitivitas Spektral" merah, biru, dan hijau dari sensor berkorelasi kamera digital yang digunakan oleh sistem ANN.

Meskipun kita sekarang mengetahui warna yang akan digunakan, satu hal yang belum dapat dipastikan adalah sensitivitas ketiga warna cahaya ini. Namun, yang dibutuhkan oleh sistem ANN hanyalah apa yang dikenal sebagai "Nilai Tri Stimulus".

Penggunaan Rangkaian Filter Warna

Kamera digital yang mengumpulkan dan menggunakan spektrum warna RGB juga memiliki chip penginderaan khusus, yang dikenal sebagai "Rangkaian Filter Warna", yang juga disebut "CFA". Dalam hal ini, dan khususnya terkait dengan jenis struktur chip ini, kita memiliki apa yang dikenal sebagai "Pola Bayer". Dalam contoh spesifik ini, terdapat setidaknya dua kali lipat (2X) jenis filter hijau dibandingkan filter merah dan biru. Alasan utamanya adalah karena terdapat berbagai sinyal luminansi yang menuju ke kamera digital, dan dari sana, gambar 2-Dimensi atau 3-Dimensi dianggap jauh lebih sensitif terhadap nilai frekuensi yang lebih tinggi daripada rezim warna dan kromatik lainnya.

Perlu dicatat juga bahwa rezim warna hijau juga jauh lebih rentan terhadap apa yang dikenal sebagai "Interpolasi" atau "Demosaicing". Selain itu, bukan hanya kamera digital yang digunakan oleh sistem ANN yang umumnya menggunakan rezim warna RGB, Monitor LCD standar juga menggunakannya. Keunggulan utama rezim warna RGB dibandingkan yang lain adalah dapat difilter terlebih dahulu secara digital untuk meningkatkan ketahanan pada snapshot yang diambil dari gambar 2-Dimensi atau 3-Dimensi yang dimaksud.

Pentingnya Keseimbangan Warna

Penting untuk dicatat bahwa dalam rezim warna RGB, apa yang dikenal sebagai "Keseimbangan Warna" digunakan untuk memindahkan rezim warna kromatik apa pun (biasanya warna putih) ke dalam gradasi warna yang sesuai yang berada di dalam rezim 2-Dimensi atau Citra 3 Dimensi. Untuk melakukan prosedur semacam ini, "Koreksi Warna" khusus dilakukan, di mana setiap pangkat perkalian dari nilai RGB tertentu dikalikan dengan faktor numerik yang berbeda. Dalam hal ini, transformasi matriks diagonal dapat dilakukan.

Teknik yang jauh lebih canggih juga dapat diterapkan di sini, seperti "Color Twist", yang menggunakan matriks transformasi matematika tiga kali tiga.

Peran Gamma dalam Rezim Warna RGB

Dalam rezim warna RGB, yang digunakan oleh kamera digital untuk sistem ANN, hubungan matematis spesifik antara tegangan kamera digital dan tegangan terkaitnya terkadang dapat disebut sebagai "Gamma", dan dapat direpresentasikan dalam salah satu dari dua cara, yaitu sebagai berikut:

$$Representasi\ 1: B = V\text{^}1;$$

$$\text{Representasi 2: Y'} = \text{Y\^1/z.}$$

Ini sebenarnya merupakan pendekatan nonlinier, tetapi perlu dicatat bahwa pendekatan ini memiliki satu keuntungan utama: segala jenis "derau" yang muncul dari pengambilan gambar, baik gambar 2-Dimensi maupun 3-Dimensi yang akan diproses oleh sistem ANN, dapat dikurangi secara otomatis di area yang terpapar warna. Selain itu, untuk memberikan optimalisasi lebih lanjut pada sistem ANN yang akan memproses berbagai gambar, gambar-gambar tersebut juga dikompresi lebih lanjut dengan memanfaatkan teknik yang dikenal sebagai "Gamma Terbalik".

Namun, kelemahan spesifik lain dari teknik yang disebutkan di atas adalah keberadaan fitur Gamma dalam snapshot yang diambil, baik pada gambar 2-Dimensi maupun 3-Dimensi, dapat menyebabkan shading lebih lanjut. Hal ini dapat diatasi jika nilai Gamma yang sesuai dapat dihitung, tetapi banyak kamera digital yang digunakan oleh sistem ANN saat ini tidak mampu melakukannya. Terdapat pula masalah lain dalam hal ini, seperti menentukan permukaan normal pada gambar 2-Dimensi maupun 3-Dimensi. Untuk membantu mengatasi tingkat ketidakpastian ini, teknik canggih lain juga digunakan, yaitu menggunakan apa yang dikenal sebagai "Stereo Fotometrik". Teknik ini akan membantu membalikkan komputasi berbasis Gamma yang telah dilakukan dan bahkan menyeimbangkan kembali warna "bercak" yang mungkin ada pada gambar 2-Dimensi maupun 3-Dimensi.

Jika teknik "Gamma Terbalik" akan digunakan langsung oleh sistem ANN, "Teknik Linearisasi" juga sangat sering dibutuhkan.

Peran Rezim Warna Lain dalam Citra 2 Dimensi dan 3 Dimensi

Sebagaimana telah disebutkan sebelumnya, meskipun rezim warna RGB dan XYZ yang paling banyak digunakan dalam kamera digital saat ini, terdapat beberapa jenis rezim warna lain yang juga telah dikembangkan, dan rezim-rezim ini juga dapat digunakan oleh sistem ANN untuk menghasilkan keluaran yang diinginkan. Dua rezim warna tersebut dikenal sebagai "YIQ" dan "YUV". Menarik untuk dicatat bahwa keduanya, masing-masing, juga memanfaatkan apa yang dikenal sebagai "Kanal Y". Hal ini sebenarnya dapat direpresentasikan secara matematis sebagai berikut:

$$\text{Y'601} = 0{,}299\text{R'} + 0{,}587\text{G'} + 0{,}144\text{B'}$$

Di mana:

R', G', dan B' sebenarnya adalah rezim warna terkompresi Merah, Hijau, dan Biru yang tertanam secara detail dalam dua rezim warna lain yang baru saja dijelaskan sebelumnya. Dari sini, bagian Ultraviolet dapat disaring dengan menggunakan algoritma matematika berikut:

$$\text{U} = 0{,}42111 * (\text{B'} - \text{Y'});$$
$$\text{V} = 0{,}877283 * (\text{R'} - \text{Y'}).$$

Dengan menggunakan algoritma matematika ini, "Kompatibilitas Mundur" bahkan dapat mendeteksi sinyal berbasis "Kroma Frekuensi Tinggi" yang masih dapat bertahan pada kamera digital yang digunakan oleh sistem ANN.

Dengan rezim warna "YIQ" dan "YUV", gambar .JPEG juga dapat dibuat. Namun, penting untuk diingat bahwa ini bukanlah ekstensi berkas format .JPEG standar, melainkan ekstensi yang harus dibuat khusus agar sistem ANN dapat memprosesnya untuk menghitung keluaran yang diinginkan. Hal ini dapat dihitung dengan algoritma matematika berikut:

$$[\mathrm{Y'}, \mathrm{Cb}, \mathrm{Ct}] = [0{,}299, -0{,}168736, 0{,}5] * [0{,}587, -0{,}331264, -0{,}418688] * [0{,}144, 0{,}5, -0{,}081312] * [\mathrm{R'}, \mathrm{G'}, \mathrm{B'}] + [0, 128, 128]$$

Di mana:
R', G', B' = komponen warna Gamma 8 bit pada gambar 2-Dimensi atau 3-Dimensi yang telah dikompresi lebih lanjut.

Algoritma matematika di atas juga dapat digunakan untuk tujuan "Deblocking" lainnya.

Ada pula rezim warna lain yang telah muncul, dan juga dapat digunakan oleh sistem ANN. Rezim ini dikenal secara khusus sebagai rezim warna "Hue, Saturation, Value", dan disingkat "HSV". Rezim ini juga merupakan subset dari rezim warna RGB. Rezim warna HSV juga memiliki properti berikut:

- Nilai Warna Maksimum;
- Saturasi:
  Jarak berskala dari piksel pada gambar 2-Dimensi atau 3-Dimensi.
- Hue:
  Orientasi vektor skema warna spesifik ini pada gambar 2-Dimensi atau 3-Dimensi.

Properti-properti yang disebutkan di atas dapat direpresentasikan secara matematis sebagai berikut:

$$\mathrm{R} = (\mathrm{R}/\mathrm{R}+\mathrm{G}+\mathrm{B});\ \mathrm{G} = (\mathrm{R}/\mathrm{R}+\mathrm{G}+\mathrm{B});\ \mathrm{B} = (\mathrm{R}/\mathrm{R}+\mathrm{G}+\mathrm{B}).$$

Peran Kompresi dalam Citra 2 Dimensi dan 3 Dimensi

Fase khusus ini, yang dikenal sebagai "Kompresi", adalah langkah terakhir dalam pemrosesan snapshot citra 2 Dimensi atau 3 Dimensi yang diambil oleh kamera digital. Terdapat algoritma matematika khusus untuk menyelesaikan tugas ini, tetapi secara umum, "Sinyal Luminansi" dikompresi lebih lanjut dengan frekuensi/sinyal fidelitas yang jauh lebih tinggi. Setelah fase pertama ini selesai, tahap selanjutnya dikenal sebagai "Transformasi Blok". Di sinilah algoritma matematika khusus, yang disebut sebagai "Transformasi Kosinus Diskrit", merupakan hasil perkalian faktor dari "Transformasi Fourier".

Setelah ini selesai, pada langkah ketiga dalam proses ini, nilai-nilai koefisien yang telah dihitung dikonversi menjadi kumpulan nilai berbasis integer yang lebih kecil. Penting untuk

diingat bahwa bukan hanya citra 2 Dimensi atau 3 Dimensi yang digunakan oleh sistem ANN untuk menghitung keluaran yang diperlukan. Aliran video juga dapat digunakan, tetapi tentu saja, ini akan membutuhkan pemrosesan dan daya komputasi yang jauh lebih besar dari sistem ANN.

Jika video digunakan sebagai pengganti gambar 2-Dimensi atau 3-Dimensi, maka pendekatan matematika lain yang disebut "Kompensasi Gerak" digunakan. Pendekatan ini secara khusus digunakan untuk mengodekan varians yang ada di antara setiap blok video, dan untuk menghasilkan matriks statistik dari blok-blok yang telah dikodekan dalam iterasi sebelumnya. Variasi yang lebih modern dari algoritma matematika khusus ini dapat secara otomatis mengukur blok-blok yang digunakan dalam segmen video oleh sistem ANN, membuat koordinat sub-piksel, dan menciptakan mekanisme yang diperlukan bagi sistem ANN untuk benar-benar menandai blok-blok sebelumnya dalam aliran video yang telah dikompresi.

Terakhir, efektivitas dan ketahanan rumus dan persamaan matematika yang digunakan dalam urutan kompresi ini sebagaimana dijelaskan dalam subbagian ini dapat diukur dengan menggunakan apa yang dikenal sebagai "Rasio Sinyal Puncak terhadap Derau" (Peak Signal to Noise Ratio), atau disingkat "PSNR". Ini adalah derivasi berbasis statistik dari "Average Mean Square Error", yang secara matematis direpresentasikan sebagai berikut:

$$\mathrm{MSE} = 1/\mathrm{n}\sum x * [\mathrm{I}(\mathrm{x}) - \mathrm{I}(\mathrm{x})]\hat{}2$$

Di mana:
$\mathrm{I}(\mathrm{x})$ = Citra Asli yang Tidak Terkompresi;
$\mathrm{I}(\mathrm{x})$ = Citra Padanan yang Terkompresi.

Dari sini, PSNR dapat dihitung secara matematis sebagai berikut:

$$\mathrm{PSNR} = 10\mathrm{LOG}10 * (\mathrm{I}\hat{}2\mathrm{max}/\mathrm{MSE}) = 20\mathrm{LOG}10 * (\mathrm{Imax}/\mathrm{RMS})$$

Di mana:
Imax = Tingkat absolut sinyal yang dapat ditransmisikan dari kamera digital ke gambar 2 Dimensi dan 3 Dimensi.

7.9 TEKNIK PEMROSESAN CITRA

Setelah kita mengulas secara detail bagaimana gambar 2 Dimensi dan 3 Dimensi dapat dibuat, kita akan membahas bagaimana gambar-gambar tersebut dapat diproses lebih lanjut sehingga sistem ANN dapat menggunakan gambar-gambar ini (baik statis maupun dinamis) dengan cara yang paling efisien dan optimal sehingga menghasilkan keluaran yang diinginkan. Di bagian bab ini, kita akan mengulas teknik pemrosesan tersebut, yang juga disebut sebagai "Transformasi". Dalam hal ini, "Operator Titik" dapat digunakan, serta "Operator Ketetanggaan".

Kedua konsep ini menggunakan teknik khusus yang dikenal sebagai "Transformasi Fourier". Lebih lanjut, teknik berbasis "Operator" yang baru saja dijelaskan juga dapat digunakan untuk membuat apa yang dikenal sebagai "Piramida Citra" dan "Wavelet" yang dapat digunakan oleh sistem ANN untuk menganalisis lebih lanjut citra 2-Dimensi atau 3-Dimensi. Terakhir, "Transformasi Geometris" juga dapat digunakan untuk membuat aspek rotasi tertentu pada citra 2-Dimensi dan 3-Dimensi.

Pentingnya Operator Titik

Operator titik dianggap sebagai teknik transformasi yang paling sederhana, dan mungkin paling mudah. Misalnya, untuk menunjukkan tingkat kesederhanaannya, setiap nilai berbasis piksel yang dihitung bergantung pada nilai sebelumnya yang telah dicapai dari piksel sebelumnya. Operator titik jenis ini dapat digunakan untuk karakteristik citra 2-Dimensi dan 3-Dimensi berikut:

- Tingkat kecerahan;
- Tingkat kontras;
- Tingkat koreksi warna;
- Tingkat transformasi geometris.

Dalam hal algoritma matematika yang terlibat dengan "Transformasi Piksel", beberapa yang terpenting adalah sebagai berikut:

$$\mathrm{G(X)} = \mathrm{h[f(x)]}, \text{which can also be represented as } \mathrm{g(x)} = h^{*}\,\mathrm{[Fo(x)]}, \ldots\ \mathrm{Fn(x)}.$$

Hal di atas merupakan teknik transformasi piksel paling dasar yang paling banyak digunakan oleh sistem ANN saat ini. Dalam hal ini:

X = Domain Dimensi;
F, g = rentang statistik spesifik dari piksel yang sedang diperiksa.

Namun, jika gambar 2-Dimensi atau 3-Dimensi diskrit atau statis akan digunakan oleh sistem ANN, dapat dipertimbangkan apa yang dikenal sebagai himpunan lokasi berbasis piksel yang pasti atau terbatas dengan jenis gambar ini. Hal ini dapat direpresentasikan secara matematis sebagai berikut:

$$\mathrm{G(I, j)} = \mathrm{h[f(I, j0)]}.$$

Selain itu, dua jenis operator titik lainnya yang digunakan dengan transformasi berbasis piksel menggunakan sifat matematika perkalian dan penjumlahan, dan dapat direpresentasikan sebagai berikut:

$$\mathrm{G(x)} = \mathrm{a\,f(X)} + \mathrm{b}.$$

Dua variabel matematika yang penting dalam transformasi titik (atau piksel) adalah "Gain" dan "Brightness". Keduanya direpresentasikan sebagai berikut:

$$G(x) = a(x) * f(x) + b(x).$$

Perlu dicatat bahwa sifat perkalian yang baru saja dijelaskan juga dapat digunakan sebagai pendekatan linear, dan ini direpresentasikan secara matematis sebagai berikut:

$$H(f0 + f1) = h(f0) + h(f1).$$

Operator titik ganda juga dapat digunakan dengan cukup mudah dan efektif, dan dapat direpresentasikan secara matematis sebagai berikut:

$$G(x) = (1 - A) * f0(x) + Af1(x).$$

Perlu dicatat juga bahwa algoritma matematika spesifik ini dapat digunakan untuk melakukan apa yang dikenal sebagai "Cross Dissolvation" antara dua atau lebih gambar dan/atau segmen video 2-Dimensi atau 3-Dimensi. Selain itu, teknik yang dikenal sebagai "Koreksi Gamma" juga dapat dilakukan, di mana semua jenis hubungan linear antara koordinat piksel dapat dihilangkan sebagaimana dianggap perlu oleh sistem ANN. Hal ini dapat dicapai dengan algoritma matematika di bawah ini:

$$G(x) = [f(x)]\hat{} - 1/A.$$

Pentingnya Transformasi Warna

Kita telah membahas berbagai rezim warna yang tersedia untuk gambar 2 Dimensi dan/atau 3 Dimensi secara cukup detail di subbagian sebelumnya dalam bab ini. Namun, dalam Pemrosesan Citra, jenis rezim warna yang berbeda ini harus dianggap sebagai sinyal ultra-berkorelasi yang dapat dikaitkan dengan piksel-piksel yang terdapat dalam gambar tersebut. Sistem ANN dapat lebih meningkatkan nilai matematika dari sinyal-sinyal ini hanya dengan menambahkan iterasi numerik yang sama berulang kali, secara iteratif. Namun, kekurangannya di sini adalah tingkat Rona dan Saturasi juga dapat ditingkatkan secara signifikan.

Sekarang muncul pertanyaan, bagaimana semua hal di atas dapat diselesaikan? Dalam hal ini, sangat penting untuk menggunakan konsep "Penyeimbangan Warna" (seperti yang telah diulas sebelumnya dalam bab ini) untuk mengalikan dan menemukan produk numerik yang paling tepat sehingga koordinat piksel, baik dalam gambar 2-Dimensi maupun 3-Dimensi (baik statis maupun dinamis), dapat dikaitkan satu sama lain dalam format berbasis linier.

Dampak Matting Gambar

Aspek kunci lain yang sangat penting dalam Visi Komputer yang berkaitan dengan gambar 2 Dimensi dan 3 Dimensi dikenal sebagai "Matting". Ini adalah teknik khusus di mana objek di "Latar Depan" salah satu gambar ini dapat ditempatkan di latar belakang gambar 2

Dimensi atau 3 Dimensi yang sama sekali berbeda secara mulus. Perlu dicatat bahwa proses terakhir, di mana objek ditempatkan ke dalam gambar yang sama sekali baru, dikenal sebagai "Komposisi". Langkah-langkah yang diperlukan di tengah proses agar semua ini terjadi dikenal sebagai "Gambar Berwarna Matted Alfa".

Namun, dalam proses yang disebutkan di atas, ada saluran lain yang dibuat, dan ini dikenal sebagai "Saluran Alfa". Metrik ini mencerminkan tingkat relatif "Cakupan Fraksional" cahaya yang dipancarkan pada setiap koordinat piksel gambar 2 Dimensi atau 3 Dimensi. Perlu dicatat pada titik ini bahwa setiap koordinat piksel yang ditempatkan dari dalam objek yang dipindahkan pada gambar 2-Dimensi atau 3-Dimensi yang lebih baru berwarna buram, sedangkan setiap koordinat piksel yang terletak di luar objek spesifik ini bersifat transparan.

Untuk mencapai teknik "Komposisi" lebih lanjut seperti yang baru saja dijelaskan, algoritma matematika berikut biasanya digunakan:

$$\mathrm{C} = (1 - \mathrm{A})\text{^}\mathrm{B} + \mathrm{aF}.$$

Akhirnya, ketika sumber cahaya dipantulkan kembali dari latar belakang yang sangat polos ke gambar 2-Dimensi atau 3-Dimensi, nilai matematika cahaya yang melewatinya dijumlahkan secara numerik. Ini juga dikenal sebagai "Gerakan Transparan".

Dampak Ekualisasi Histogram

Salah satu pertanyaan kunci yang sering diajukan dalam Visi Komputer saat ini adalah, bagaimana nilai matematika untuk karakteristik kecerahan dan penguatan gambar 2-Dimensi dan 3-Dimensi dapat dipastikan? Atau dengan kata lain, bagaimana mereka dapat dioptimalkan lebih lanjut agar paling sesuai untuk sistem ANN? Metodologi yang lebih sederhana untuk menjawab pertanyaan-pertanyaan kunci ini adalah dengan menemukan koordinat piksel tergelap dan terterang pada gambar-gambar spesifik ini, dan membandingkannya dengan Bidang Geometri Kartesius hitam putih.

Tentu saja, pendekatan yang jauh lebih statistik untuk hal ini adalah dengan menemukan nilai rata-rata dari semua koordinat piksel ini, dan dari sana memperluas rentang matematis tempat mereka berada saat ini. Dalam contoh spesifik ini, seseorang perlu membuat "Histogram" dari semua rezim warna yang ada dalam gambar 2-Dimensi atau 3-Dimensi, dan dari sana, sekali lagi menggunakan statistik untuk menghitung properti berikut:

- Nilai Minimum;
- Nilai Maksimum;
- Nilai Intensitas Rata-rata.

Teknik yang dikenal sebagai "Equalization Histogram" dapat digunakan di sini. Dengan teknik ini, tujuannya adalah untuk mencapai keseimbangan tertentu antara koordinat piksel yang lebih gelap dan lebih terang yang terdapat pada citra 2-Dimensi atau citra 3-Dimensi. Dari sini, sistem ANN dapat mengambil sampel acak dari koordinat piksel ini untuk menentukan mana yang paling efektif dalam menghitung keluaran yang diinginkan. Hal ini dapat dilakukan melalui "Fungsi Kepadatan Probabilitas", yang terkadang juga disebut sebagai "Fungsi Distribusi Kumulatif", dan secara matematis dapat direpresentasikan sebagai berikut:

$$c(I) = 1/NI\sum i = 0h(i) = c(I-1) + 1/Ng(T)$$

Di mana:
N = jumlah total piksel pada citra 2-Dimensi atau 3-Dimensi.

Namun, terlepas dari keuntungan yang ditawarkan "Ekualisasi Histogram", salah satu kelemahan utamanya adalah pada koordinat piksel yang lebih gelap pada gambar 2 Dimensi atau 3 Dimensi, objek ekstra yang sangat kecil dapat diperbesar secara signifikan, sehingga mendistorsi kualitas keseluruhan gambar.

Memanfaatkan Ekualisasi Histogram Berbasis Lokal

Perlu dicatat bahwa teknik yang baru saja diulas di subbagian terakhir dianggap bersifat "Global". Ini berarti seluruh gambar 2 Dimensi atau 3 Dimensi dianalisis secara keseluruhan. Namun, terkadang, hal ini mungkin tidak diperlukan. Dengan kata lain, mungkin cukup hanya menganalisis lebih lanjut segmen atau wilayah tertentu saja dari gambar 2 Dimensi atau 3 Dimensi. Dengan demikian, dalam hal ini, matriks matematika (dilambangkan sebagai "MxM") dapat digunakan untuk menerapkan algoritma matematika untuk "Ekualisasi Histogram", dan hanya dapat digunakan untuk koordinat piksel tertentu dalam gambar.

Namun, proses ini sebenarnya dapat diotomatisasi dalam artian bahwa "Jendela Bergerak" berbasis statistik dapat diterapkan ke semua koordinat piksel pada gambar 2-Dimensi atau 3-Dimensi yang dimaksud. Namun, ini memerlukan beberapa pengkodean, dan ini sebenarnya dapat dilakukan dengan Kode Sumber Python.

Terdapat juga metodologi lain yang dapat digunakan dalam hal ini, dan secara teknis dapat disebut sebagai "Ekualisasi Histogram Adaptif". Dengan ini, nilai matematika dari koordinat piksel yang tidak tumpang tindih dalam gambar 2-Dimensi atau 3-Dimensi dapat dihitung. Hal ini direpresentasikan secara matematis sebagai berikut:

$$Fs,x(I) = (1-s)\ ^*\ (1-t)\ f00(I) + s(1-t)f10(I) + (1-s)\text{^}t\ f01(I) + stf11(I).$$

Namun, versi yang jauh lebih sederhana adalah melakukan pencarian berbasis statistik pada masing-masing dari empat sudut matriks MxM generik berbasis matematika. Dari sini, koordinat piksel dari keempat sudut ini dapat digabungkan menjadi satu penjumlahan utuh, yang selanjutnya dapat didistribusikan secara statistik; persamaan matematika untuk melakukan hal ini adalah sebagai berikut:

$$Hk,j[I(I,j0] += w(I,j,k)$$

Di mana:
$w(I,j,k)$ = Fungsi Pembobotan Bilinear antara berbagai koordinat piksel.

BAB 8
KONSEP PENYARINGAN LINEAR

Teknik yang diulas di subbagian sebelumnya, yang disebut "Equalization Histogram Adaptif Lokal", juga merupakan replikasi sempurna dari apa yang dikenal dalam statistik sebagai "Operator Ketetanggaan". Dalam contoh khusus ini, teknik khusus ini dapat digunakan untuk memastikan penjumlahan matematis dari nilai-nilai koordinat piksel hanya berdasarkan salah satu nilai yang dianggap berdekatan, baik pada gambar 2-Dimensi maupun 3-Dimensi yang dimaksud.

Teknik ini juga dapat digunakan untuk subset koordinat piksel tertentu, untuk menghitung nilai akhirnya. Teknik ini juga dapat digunakan untuk meningkatkan karakteristik gambar berikut:

- Mengatur tonalitas Rezim Warna;
- Menambahkan objek buram halus untuk tujuan penyempurnaan;
- Menambahkan lebih banyak detail;
- Membuat tepian lebih jelas;
- Menghilangkan objek yang tidak diperlukan atau asing.

Untuk mencapai hal di atas, "Filter Linear" khusus digunakan, dan algoritma matematikanya adalah sebagai berikut:

$$\mathrm{G(I,j)} \;=\; \sum \mathrm{k,l\ f(i+k,j+1)} \;^{*}\; \mathrm{h(k,L)}$$

Di mana:
$\mathrm{h(k,L)} \;=\;$ Koefisien Filter.

Versi yang lebih terfilter dari algoritma matematika di atas direpresentasikan sebagai:

$$\mathrm{G} \;=\; \mathrm{f0/\backslash h.}$$

Namun, penting untuk diingat bahwa algoritma matematika di atas hanya cocok untuk gambar 2-Dimensi atau 3-Dimensi yang dirancang cukup sederhana. Namun, perlu diingat bahwa sistem ANN saat ini, dirancang, telah diprogram untuk memproses gambar yang sangat kompleks, dan hal itu tidak membebani sumber daya pemrosesan atau komputasinya hingga batas maksimal. Namun, agar hal ini terjadi, persamaan matematika khusus lainnya harus digunakan, yaitu sebagai berikut:

$$\mathrm{G(I,j)} \;=\; \sum \mathrm{k,I\ f([-k,j-I)} \;*\; \mathrm{h(k,L)} \;=\; \sum \mathrm{k,I\ f([-k,j-I).}$$

8.1 PENTINGNYA PADDING PADA CITRA 2 DIMENSI ATAU 3 DIMENSI

Namun, kelemahan utama lain dari teknik matematika yang diulas di subbagian sebelumnya adalah apa yang dikenal sebagai "Efek Batas". Efek ini hanyalah penggelapan super pada semua koordinat piksel yang terletak di keempat sudut citra 2 Dimensi atau 3 Dimensi. Untuk mengatasi efek negatif ini, konsep yang dikenal sebagai "Padding" dapat digunakan, dan beberapa yang lebih penting (yang berkaitan dengan sistem ANN) adalah sebagai berikut:

- Penolan:
  Ini menetapkan semua koordinat piksel ke nilai matematika "0" pada citra 2 Dimensi atau 3 Dimensi.
- Konstanta:
  Ini terjadi ketika semua koordinat piksel dihitung dan dikaitkan dengan nilai matematika yang telah ditetapkan.
- Penjepitan:
  Kedua proses di atas dapat diulang secara iteratif, secara otomatis.
- Pembungkusan Siklik:
  Ini menciptakan berbagai putaran di sekitar koordinat piksel pada gambar 2 Dimensi atau 3 Dimensi.
- Pencerminan:
  Ini adalah properti matematika khusus yang digunakan untuk lebih mencerminkan koordinat piksel pada gambar 2 Dimensi atau 3 Dimensi.
- Ekstensi:
  Ini adalah ekstensi matematis dari koordinat piksel pada gambar 2 Dimensi atau 3 Dimensi ketika dibandingkan dengan sinyal yang ditransmisikan dari koordinat piksel di tepi gambar 2 Dimensi atau 3 Dimensi.

Dalam dunia Grafik Komputer, teknik "Padding" ini juga dikenal sebagai "Mode Pembungkusan atau Pengalamatan Tekstur". Ini membantu menjaga koordinat piksel di keempat tepi gambar 2 Dimensi atau 3 Dimensi dari efek penggelapan lebih lanjut. Namun, jika hal ini memang terjadi, rezim warna "RBGA" (sebagaimana telah diulas secara rinci sebelumnya dalam bab ini) dapat dihitung nilai "Alfa"-nya secara statistik sehingga efek ini dapat segera dihentikan.

Efek Penyaringan Terpisah

Ada juga proses yang dikenal sebagai "Konvolusi" dalam dunia Visi Komputer. Proses ini memanfaatkan apa yang dikenal sebagai operator matematika "K2" (yang merupakan perkalian dan penjumlahan) di setiap koordinat piksel gambar 2-Dimensi atau 3-Dimensi. Dalam kasus ini, nilai "K" hanya mewakili tinggi dan lebar total gambar yang dimaksud. Teknik "Konvolusi" ini juga dapat diterapkan secara terpisah pada tinggi dan lebar masing-masing.

Jika hal di atas dilakukan, maka nilai berbasis "K" dianggap sebagai apa yang dikenal sebagai "Terpisah". Lebih lanjut, hal ini dapat direpresentasikan secara matematis sebagai berikut:

$$K = vh\text{^}T.$$

Namun, untuk memastikan apakah fungsi "Terpisah" telah benar-benar dilakukan pada gambar 2-Dimensi atau 3-Dimensi, algoritma matematika berikut harus digunakan:

$$K = \sum iOiu2v\text{^}Ti.$$

Penting untuk diingat bahwa algoritma matematika di atas hanya dapat digunakan jika seluruh gambar 2-Dimensi atau 3-Dimensi dianalisis lebih lanjut. Untuk memastikan apakah fungsi "Pemisahan" telah benar-benar dilakukan untuk tinggi dan lebar secara individual, maka dua perhitungan Akar Kuadrat berikut harus dilakukan:

For the Height: SQUAREROOT o0u0;
For the Width: SQUAREROOT O0v^Tu.

Apa Itu Filter Band Pass dan Filter Steerable

Perlu dicatat bahwa sejauh ini, terdapat jenis "Operator" khusus lainnya, selain yang baru saja diulas di subbagian sebelumnya. Misalnya, terdapat Operator "Sobel" dan "Corner", yang utamanya digunakan untuk menghaluskan kurva yang diperlukan, baik pada gambar 2 Dimensi maupun 3 Dimensi. Hal ini dapat direpresentasikan secara matematis sebagai berikut, ketika menggunakan alat statistik canggih yang dikenal sebagai "Filter Gaussian":

$$G(x, y, O) = (1/2TT0\text{^}2) - (x\text{^}2 + y\text{^}2/2n\text{^}2).$$

Filter-filter di atas juga disebut secara teknis sebagai "Filter Band Pass". Filter ini digunakan untuk menyaring frekuensi-frekuensi yang biasanya tidak diperlukan dari sumber proyeksi cahaya. Terdapat juga "Operator" khusus lainnya yang dikenal sebagai "Operator Laplacian". Teknik khusus ini dapat digunakan untuk mengurangi keburaman halus pada gambar 2-Dimensi maupun 3-Dimensi. Hal ini direpresentasikan melalui matematika sebagai berikut:

$$\text{^}2G(x, y, z) = (x\text{^}2 + y\text{^}2/0\text{^}4) - (2/O2) * G(x, y, z).$$

Lebih spesifik lagi, "Operator Sobel" dapat digunakan untuk mengekstrapolasi orientasi vektor secara statistik (baik dari perspektif Terarah maupun Berorientasi). Namun, algoritma matematika yang disebutkan di atas juga dapat digunakan untuk menyelesaikan tugas ini. Dari sini, "Arahan Terarah" dapat digunakan, yang secara statistik direpresentasikan sebagai berikut:

$$V\underline{u} = o/0\underline{u}.$$

Ada satu filter khusus lagi yang perlu diulas dalam subbagian ini, yaitu "Filter Pengarah". Algoritma matematika yang mendorong konsep ini diilustrasikan di bawah ini:

$$\underline{Guu} = u\text{^}2Gxx + 2uvGxy + v\text{^}2Gyy.$$

Teknik ini paling umum digunakan untuk membuat apa yang dikenal sebagai "Deskripsi Fitur" di sekitar koordinat piksel, baik pada gambar 2-Dimensi maupun 3-Dimensi. Dalam kasus ini, matriks matematika dua-kali-dua digunakan.

Pentingnya Filter Citra Integral

Jika citra 2 Dimensi atau 3 Dimensi akan digunakan secara berurutan, berulang-ulang, dan berulang kali dalam sistem ANN, maka Filter Citra Integral perlu digunakan. Dalam hal ini, sangat penting bagi sistem ANN untuk terlebih dahulu menetapkan apa yang dalam istilah matematika dikenal sebagai "Tabel Luas Terjumlah". Hal ini juga ditunjukkan secara matematis sebagai berikut:

$$S(I,j) = I \sum k = 0 \ ^* f(k,l)$$

Di mana:

$S(I,j) =$ Citra Integral.

Sekarang, "Tabel Luas Terjumlah" diidentifikasi sebagai berikut:

$$[i0,i1] \ X \ [j0,j1].$$

Dari sini, keempat sudut terpisah dari citra 2 Dimensi atau citra 3 Dimensi dijumlahkan untuk mempercepat siklus efisiensi iterasi yang terjadi (seperti yang baru saja dijelaskan sebelumnya). Hal ini dicapai dengan algoritma matematika berikut:

$$S(i0 \ \ldots \ i1, j0 \ \ldots \ j1) = i1 \sum i = i0 \ j1 \sum j = j0 \ ^*$$
$$S(i1,j1) - s(i1,j0-1) - s(i0-1,j1) + s(i0-1,jo-1).$$

Perlu dicatat bahwa salah satu kelemahan terbesar penggunaan teknik khusus semacam ini adalah dianggap sebagai pendekatan logaritmik (dilambangkan sebagai $M + \log N$). Setiap pengumpulan gambar 2 Dimensi atau 3 Dimensi dalam jumlah besar akan menghasilkan ukuran bit yang sangat besar, yang selanjutnya akan membutuhkan daya pemrosesan dan komputasi yang sangat besar dari sistem ANN yang digunakan.

Teknik ini juga telah digunakan dalam versi-versi sebelumnya dari Teknologi Pengenalan Wajah untuk aplikasi tingkat rendah. Gambar yang diambil dari sini sering dimodelkan sebagai "Eigenfaces," yang terdiri dari banyak sekali persegi panjang berbasis geometri. Secara teknis, ini dikenal sebagai "Boxets". Jika statistik tingkat tinggi digunakan di sini, "Summation of the Squared Differences" (juga dikenal sebagai "SSD") juga dapat digunakan, dalam upaya

menghitung nilai matematis total koordinat piksel dalam berbagai Eigenface yang telah digunakan oleh sistem Pengenalan Wajah.

Rincian Teknik Penyaringan Rekursif

Teknik penyaringan semacam ini terutama digunakan untuk pemrosesan sinyal. Di sinilah berbagai rezim warna yang ditransmisikan ke gambar 2 Dimensi atau 3 Dimensi berkumpul dan terakumulasi menjadi satu area gambar, sehingga dapat menyebabkan lebih banyak keburaman atau munculnya objek penghalang lainnya, sehingga semakin menurunkan kualitas gambar. Secara teknis, ini juga dikenal sebagai "Respons Impuls Tak Terbatas", atau singkatnya "IRR". Alasan utama mengapa teknik ini diberi nama ini adalah karena beberapa rezim warna ini dapat diproyeksikan hingga tak terhingga, jika tidak ada penghalang lain di jalurnya. Metode "IRR" biasanya digunakan untuk menghitung kernel masif yang telah dihaluskan secara statistik, baik pada citra 2-Dimensi maupun 3-Dimensi. Namun, sebagaimana telah dibahas secara detail sebelumnya dalam bab ini, pendekatan "Piramida" juga dapat digunakan untuk mencapai tingkat tujuan yang sama.

8.2 TEKNIK OPERASIONAL LAIN YANG DIGUNAKAN OLEH SISTEM ANN

Meskipun kita telah membahas cukup banyak "Operator" dalam bab ini, masih banyak lagi yang tersisa dan juga dapat digunakan oleh sistem ANN. Secara umum, teknik-teknik tersebut dapat dikategorikan sebagai berikut:

- Filter Median Pemelihara Tepi;
- Filter Bilateral;
- Filter Morfologi;
- Filter Semi-Global.

Perlu diingat juga bahwa "Operator" dan teknik penyaringannya secara tradisional berbasis linear. Di bagian bab ini, kita akan membahas teknik penyaringan yang pendekatannya berbasis non-linear. Sebenarnya, pendekatan berbasis linear adalah yang paling mudah dihitung, dalam arti bahwa nilai matematis setiap koordinat piksel, baik dalam gambar 2-Dimensi maupun 3-Dimensi, dapat dianggap sebagai penjumlahan matematis yang lengkap dari koordinat piksel di sekitarnya.

Teknik penyaringan berbasis linear seperti ini paling disukai untuk digunakan dalam sistem ANN, karena membutuhkan overhead yang lebih sedikit, dan mudah diterapkan untuk menghitung keluaran yang diinginkan. Namun, perlu diingat juga bahwa sistem ANN saat ini sangat canggih dan canggih; sehingga dapat memperhitungkan teknik penyaringan berbasis nonlinear apa pun. Oleh karena itu, tujuan dari bagian ini dalam bab ini adalah untuk mengkaji teknik-teknik semacam ini secara lebih rinci.

Teknik Penyaringan Median

Dengan teknik khusus ini, nilai median statistik untuk setiap koordinat piksel terdekat yang mengelilingi koordinat pusat pada citra 2 Dimensi atau citra 3 Dimensi dihitung. Dengan menggunakan pendekatan ini, setiap koordinat piksel yang tidak berkontribusi lebih lanjut pada citra 2 Dimensi atau citra 3 Dimensi secara otomatis dihapus dan dibersihkan. Namun,

terlepas dari keuntungan utama ini, salah satu kelemahan utama dari pendekatan ini adalah hanya dapat melihat satu koordinat piksel pada satu waktu.

Dengan kata lain, sistem tidak dapat melihat penjumlahan median keseluruhan dari kelompok koordinat piksel secara bersamaan. Hal ini tentu saja membutuhkan banyak waktu untuk diproses, sehingga semakin memperlama waktu yang telah dialokasikan bagi sistem ANN untuk menghitung keluaran yang diinginkan. Oleh karena itu, alternatif lain untuk pendekatan khusus ini adalah menggunakan apa yang dikenal sebagai "Indeks Median Tertimbang", yang dapat memanfaatkan fungsi pengelompokan ini.

Hal ini direpresentasikan secara matematis sebagai berikut:

$$\sum k, j \ ^{*} \ w(k, l) \ ^{*} \ f(I \ + \ k1j \ + \ 1) \ - \ g(I, j)|\text{^}P$$

Di mana:

- $g(I, j)$ = keluaran yang diinginkan yang akan dihitung oleh sistem ANN;
- p = nilai numerik "1" untuk Indeks Median Tertimbang.

Selain itu, keuntungan utama lain dari penggunaan "Indeks Median Tertimbang" adalah dapat digunakan untuk "Edge Preserving" baik pada citra 2-Dimensi maupun citra 3-Dimensi. Hal ini memungkinkan tepi citra tersebut tampak lebih halus daripada aslinya.

Penyaringan Bilateral

Sebagaimana disebutkan di subbagian terakhir bab ini, "Indeks Median Tertimbang" tidak dapat digunakan secara otomatis. Namun, dalam teknik khusus ini, yang dikenal sebagai konsep "Penyaringan Bilateral", proses ini tidak hanya diotomatisasi, tetapi juga menggunakan prinsip yang sama, yaitu nilai median statistik dari setiap koordinat piksel terdekat yang mengelilingi koordinat pusat pada citra 2-Dimensi atau citra 3-Dimensi kemudian dihitung.

Rumus matematika untuk teknik ini adalah sebagai berikut:

$$G(I, j) \ = \ [\sum k, I \ ^{*} \ f(k, l) \ ^{*} \ w(I, j, l)]/[\sum w(I, j, l)].$$

Terakhir, sebuah konsep yang dikenal dalam matematika sebagai "Vector Distancing" juga digunakan untuk membantu tidak hanya mengotomatiskan proses yang baru saja dijelaskan, tetapi juga mempercepatnya.

Teknik Iterated Adaptive Smoothing/Anisotropic Diffusion Filtering

Dengan bentuk teknik khusus ini, Filter Bilateral (sebagaimana diulas sebelumnya dalam bab ini) juga dapat digunakan berulang kali secara iteratif. Namun dalam keadaan ini, hanya pengelompokan koordinat piksel yang sangat kecil yang sebenarnya diperlukan. Pengelompokan ini dapat digambarkan sebagai berikut, dalam istilah matematika:

$$D(I, j, k, l) \ = \ EXP \ [(i - k)\text{^}2 \ + \ (j - i)\text{^}2)]/2O\text{^}2d \ = \ \{1, V \ = \ e\text{^} - 1/2o\text{^}2d, |k - i| \ + \ |t - j| \ = \ 0, |k - i| \ + \ |t - j| \ = \ 1$$

Di mana:
$R = \sum(k,j)\text{^}r(I,j,k,l),(k,l)$ adalah koordinat piksel terdekat pada citra 2 Dimensi atau citra 3 Dimensi;
$(I,j) =$ proses iteratif yang baru saja dijelaskan sebelumnya.

Algoritma matematika di atas juga dapat disebut sebagai "Teknik Penyaringan Difusi Anisotropik", dan keuntungan utamanya adalah dapat diterapkan pada hampir semua jenis permasalahan Visi Komputer yang memerlukan sistem ANN.

Namun, perlu dicatat lebih lanjut bahwa teknik matematika khusus ini juga dapat digunakan untuk mengonversi citra 2 Dimensi atau 3 Dimensi statis menjadi citra dinamis. Akan tetapi, sebaiknya setiap masalah penghalusan yang akan diselesaikan dalam hal ini dilakukan dengan pendekatan statistik gabungan.

Teknik Morfologi

Pada titik ini, perlu ditegaskan kembali bahwa teknik penyaringan berbasis nonlinier sangat sering digunakan untuk memproses citra skala abu-abu 2-Dimensi atau 3-Dimensi oleh sistem JST. Namun, hal ini hanya dapat terjadi setelah "Operasi Ambang Batas" tertentu telah dilakukan, dan ini dilakukan menggunakan teknik statistik berikut:

$$0/(f,t) = \{1 \text{ if } f > t;\ 0 \text{ Else}).$$

Teknik biner sering digunakan dalam hal ini, dan secara teknis disebut sebagai "Operasi Morfologi". Alasan utama disebut demikian adalah karena teknik ini secara harfiah dapat mengubah bentuk geometris objek yang dianggap biner, baik dalam citra 2 Dimensi maupun citra 3 Dimensi. Untuk menjalankan prosedur semacam ini, objek-objek spesifik ini digabungkan secara statistik dengan apa yang dikenal sebagai "Elemen Penataan".

Dari sini, "Nilai Keluaran Biner" kemudian dipilih, yang merupakan fungsi langsung dari permutasi yang telah ditetapkan dalam Proses Penggabungan. Penting untuk dicatat bahwa ini dapat mengambil semua jenis bentuk geometris, dan juga dapat diterapkan pada matriks matematika tiga-kali-tiga apa pun. Rumus statistik untuk melakukan komputasi semacam ini adalah:

$$C = f\,0/\backslash S.$$

Ini adalah pendekatan berbasis bilangan bulat yang akan digunakan. Berikut ini adalah beberapa properti terpenting dari "Teknik Morfologi":

- Dilasi:
  Direpresentasikan sebagai:

$$\text{Dilate}(f,s) = 0\backslash(c.1).$$

- Erosi:

Direpresentasikan sebagai:

$$\mathrm{ERODE}(f, s) = 0\backslash(c, S).$$

- Mayoritas:
  Direpresentasikan sebagai:

$$\mathrm{MAJ}(f, s) = 0(c, S/2).$$

- Pembukaan:
  Direpresentasikan sebagai:

$$\mathrm{OPEN}(f, s) = \mathrm{DILATE}[\mathrm{ERODE}(f, s), s)].$$

- Penutupan:
  Direpresentasikan sebagai:

$$\mathrm{CLOSE}(f, s) = \mathrm{ERODE}[\mathrm{DILATE}(f, s), s)].$$

Pada properti spesifik ini, Dilasi justru memperdalam, atau menebalkan, koordinat piksel pada gambar 2-Dimensi atau 3-Dimensi, dan Erosilah yang justru mengecilkannya dalam hal nilai matematis. Selain itu, Penutupan dan Pembukaan tidak memengaruhi area besar berbasis satu entitas yang terdapat pada gambar 2-Dimensi atau 3-Dimensi.

Dampak Teknik Transformasi Jarak

Ini adalah konsep yang digunakan untuk menghitung secara matematis jarak yang telah ditetapkan pada kurva parabola dengan setidaknya dua titik atau lebih yang telah ditetapkan secara pasti. Teknik ini dapat melakukan hal-hal berikut:

- Perhitungan Set Level;
- Melakukan Pencocokan Transfer Cepat;
- Penggunaan Feathering dan Image Stitching pada citra 2 Dimensi atau 3 Dimensi.

Algoritma matematika yang digunakan untuk menghitung adalah sebagai berikut:

$$\mathrm{D1}(k, l) = |k| + |I|.$$

Namun, algoritma matematika di atas hanya bersifat "generik". Ada dua teknik spesifik lain yang dapat digunakan, yaitu:

- Jarak Manhattan;
- Jarak Euclidean.

Jarak Euclidean direpresentasikan secara matematis sebagai berikut:

$$D1(k,l) = SQUAREROOT\ k\text{^}2 + l\text{^}2.$$

Jarak Manhattan direpresentasikan secara matematis sebagai berikut:

$$D(I,j) = MIN\ k,j\ b(k,l) = 0\ ^{*}\ d(i-k,j-1).$$

Karena kedua algoritma matematika yang disebutkan di atas dianggap cukup efisien, sebenarnya tidak perlu menggunakan rumus "Jarak Euclidean" untuk jenis aplikasi ini. Sebagai gantinya, rumus matematika "Vector Valued Distancing" juga dapat digunakan. Di sinilah nilai "x" dan "y" yang sesuai dari koordinat piksel pada gambar 2-Dimensi atau 3-Dimensi digunakan untuk menghitung Luas Persegi, atau "Hipotenusa" dari gambar yang dimaksud.

Terdapat pula Rumus Jarak lain yang dikenal secara khusus sebagai teknik "Transformasi Jarak Bertanda". Ini secara khusus menghitung jarak matematis untuk semua koordinat piksel dalam gambar 2-Dimensi atau 3-Dimensi, dan ini dilakukan dengan menggunakannya secara paralel dengan teknik jarak lain yang baru saja dijelaskan di atas. Intinya, semua teknik jarak tertentu dapat cukup efisien dalam hal penyelarasan dan penggabungan objek 2-Dimensi yang bersifat lengkung dengan permukaan 3-Dimensi yang telah dirancang sedemikian rupa.

Efek Komponen Terhubung

Teknik semacam ini dianggap bersifat semi-global. Dalam teorema ini, wilayah geometris yang dekat, atau bersebelahan, dengan koordinat piksel pada citra 2-Dimensi atau 3-Dimensi sebenarnya memiliki tingkat nilai input yang sama. Penggunaan teorema "Komponen Terhubung" dapat digunakan untuk berbagai aplikasi berikut oleh sistem ANN:

- Menemukan dan menentukan lokasi objek tertentu dalam semua jenis citra berbasis 2-Dimensi atau 3-Dimensi;
- Menemukan dan menentukan lokasi semua jenis "Objek Ambang" dalam citra 2-Dimensi atau 3-Dimensi, dan dari sana menghitung statistik yang diperlukan untuk digunakan oleh sistem ANN.

Untuk menggunakan teknik khusus ini, citra 2-Dimensi atau citra 3-Dimensi harus dipisahkan secara horizontal. Setelah tugas spesifik ini selesai, fase selanjutnya adalah menggabungkan berbagai rezim warna (sebagaimana diulas sebelumnya dalam bab ini) menjadi satu kesatuan yang kohesif, atau struktur.

Statistik luas yang dapat dihitung untuk citra 2 Dimensi atau 3 Dimensi dengan menggunakan teorema "Komponen Terhubung" adalah sebagai berikut:

- Luas geometris (yang merupakan penjumlahan matematis dari semua koordinat piksel);
- Keliling (yang merupakan penjumlahan matematis dari semua koordinat piksel pada tingkat batas);
- Sentroid citra 2 Dimensi atau 3 Dimensi (yang tidak lain adalah rata-rata statistik dari nilai "x" dan "y" dari koordinat piksel);
- Menghitung "Momen Kedua" yang dilakukan sebagai berikut:

$$M = \sum(x,y)Er\,[x-x] * [y-y] * [x-x,y-y].$$

Setelah statistik yang disebutkan di atas dihitung, statistik tersebut dapat digunakan untuk mengurutkan secara otomatis berbagai wilayah dalam gambar 2-Dimensi atau 3-Dimensi.

Teknik Transformasi Fourier

Transformasi Fourier adalah teknik statistik khusus yang dapat digunakan secara spesifik untuk menganalisis lebih lanjut berbagai rezim warna dan berbagai jenis filter yang dapat digunakan dengannya. Selain itu, "Analisis Fourier" dapat digunakan untuk mendeskripsikan dan menganalisis lebih lanjut "konten berbasis kualitatif" yang terkait dengan gambar 2-Dimensi atau 3-Dimensi yang dimaksud. Jika gambar-gambar spesifik ini cukup besar untuk diproses, maka pendekatan lain yang lebih modern adalah menggunakan apa yang dikenal sebagai "Teknik Transformasi Fourier Cepat", yang juga dikenal sebagai "FTT". Selain itu, frekuensi sumber cahaya yang terkait dengan gambar 2-Dimensi atau 3-Dimensi juga dapat dipelajari, dengan memanfaatkan teknik FTT, seperti yang baru saja dijelaskan.

Algoritma matematika untuk melakukan semua hal di atas dijelaskan di bawah ini:

$$S(x) = SIN\ ^{*}(2TTFx + 0\backslash) = SIN\ ^{*}(Wx + 0\backslash i)$$

Di mana:
F = Tingkat Frekuensi;
W = 2TTf = Frekuensi Sudut spesifik; 0\i = Fase spesifik;
X = Koordinat Spasial dari citra 2 Dimensi atau citra 3 Dimensi yang dimaksud.

Alasan utama penggunaan "X" untuk menyatakan hal di atas adalah karena ia juga dapat dianggap sebagai apa yang dikenal sebagai "Bilangan Imajiner". Dengan menggunakan rezim numerik semacam ini, akan jauh lebih mudah untuk membedakan antara berbasis horizontal (dilambangkan sebagai "x") dan berbasis vertikal (dilambangkan sebagai "y") dalam ruang frekuensi citra 2 Dimensi atau citra 3 Dimensi. Jika koordinat sumbu ini digunakan, maka bidang bilangan imajiner juga dapat direpresentasikan sebagai "j" dalam hal ini.

Selain itu, "Sinyal Sinusodial" tambahan (dilambangkan sebagai "s[x]") juga dapat dimasukkan ke dalam algoritma matematika di atas, dan persamaan yang dihasilkan akan terlihat seperti ini:

$$O(x) = h(x) * s(x) = Asin * (wX + o\backslash O)t.$$

Akhirnya, teknik "FTT" dapat direpresentasikan secara matematis sebagai berikut:

$$H(w) = F\{h(x)\} = Ae\text{^}j0\backslash$$

Di mana:
W = respons statistik terhadap Frekuensi Sinusoid berbasis kompleks;
$H(x)$ = frekuensi khusus yang dilalui oleh susunan filter cahaya.

Untuk kemudahan pemrosesan dan optimasi oleh sistem ANN, teknik FTT juga dapat direpresentasikan secara matematis sebagai berikut:

$$H(x)\ U\ \Diamond\ F\ U\ \Diamond\ H(w).$$

Namun, perlu diingat bahwa algoritma matematika di atas tidak dapat digunakan untuk semua aplikasi sistem ANN. Dengan kata lain, filter dan Fungsi Sinusodial mengikuti iterasi tertentu, yaitu sebagai berikut: "Fase, Pergeseran, Ulangi." Proses iteratif ini dapat berlangsung sebanyak yang dibutuhkan oleh sistem ANN hingga keluaran yang diinginkan telah dihitung. Kelemahan utama dari hal ini adalah bahwa melakukan hal ini untuk jumlah loop yang tak terbatas dapat benar-benar menguras sumber daya pemrosesan dan komputasi sistem ANN. Dengan demikian, algoritma matematika lain dapat digunakan untuk menghitung terlebih dahulu jumlah total iterasi yang diperlukan untuk sistem ANN, dan ini dapat direpresentasikan secara matematis sebagai berikut:

$$H(w)\ =\ S\ (+\ \mathrm{INFINITE})\ (-\ \mathrm{INFINITE})\ h(x)e\hat{}-\mathrm{Jwedx1}.$$

Perlu dicatat bahwa algoritma matematika di atas hanya berada dalam "Domain Kontinu". Jika Anda ingin menggunakannya untuk "Domain Diskrit" pada sistem JST, maka algoritma matematika berikut harus digunakan:

$$H(k)\ =\ 1/N\ *\ [N-1\sum x=0\ *\ h(x)e\hat{}-j2TTke/N]$$

Di mana:
N = panjang matematika total sinyal Sinusodial yang ditransmisikan ke area atau wilayah tertentu pada gambar 2-Dimensi atau 3-Dimensi yang sedang dipelajari atau dianalisis oleh sistem JST.

Perlu dicatat bahwa algoritma matematika yang baru saja dijelaskan juga secara teknis disebut sebagai "Transformasi Fourier Diskrit", atau disingkat "DFT". Namun, satu kelemahan penggunaan ini adalah algoritma ini biasanya hanya dapat digunakan dalam rentang matematika seperti yang dilambangkan di bawah ini:

$$K\ =\ [-N/2, +N/2].$$

Alasannya adalah karena nilai-nilai matematika dalam rentang numerik yang lebih tinggi sebenarnya memberikan informasi dan detail lebih lanjut tentang berbagai frekuensi

yang dipantulkan kembali dari gambar 2 Dimensi atau gambar 3 Dimensi ketika berbagai rezim warna disinarikan ke dalamnya.

Setelah teknik FTT dikaji lebih detail, penting pada tahap ini untuk meninjau beberapa propertinya yang lebih penting, yang dapat dijelaskan sebagai berikut:

1) Superposisi:
   Properti ini merepresentasikan penjumlahan matematis dari semua nilai FTT yang dihasilkan oleh gambar 2 Dimensi dan gambar 3 Dimensi.
2) Pergeseran:
   Perlu dicatat bahwa FTT sebenarnya dianggap sebagai "sinyal yang bergeser" dari transformasi yang dihasilkan oleh sumber pencahayaan asli yang telah digunakan. Ini kemudian dikalikan lebih lanjut untuk mendapatkan produk yang dikenal sebagai "Pergeseran Fase Linear". Secara teknis, ini juga disebut sebagai "Sinusoid Kompleks".
3) Pembalikan:
   Ini terjadi ketika FTT benar-benar menjadi "sinyal terbalik", dan dengan demikian menjadi penjumlahan matematis kompleks (atau "Konjugat") dari transformasi berbagai sinyal yang dihasilkan oleh rezim warna yang berbeda.
4) Konvolusi:
   Ini adalah FTT yang telah ditransformasikan melalui sepasang "Sinyal Konvolusional" yang merupakan hasil perkalian seperti yang dijelaskan dalam "Pergeseran".
5) Korelasi:
   Ini adalah FTT yang merupakan korelasi berbasis statistik dari hasil perkalian transformasi pertama yang dilakukan oleh sistem ANN yang kemudian dikalikan lagi dengan "Konjugat Kompleks" kedua.
6) Perkalian:
   Ini adalah FTT yang sebenarnya merupakan transformasi dari dua sinyal terpisah yang ditransmisikan oleh rezim warna berbeda yang telah berevolusi dalam "Konvolusi" proses transformasi.
7) Diferensiasi:
   Ini adalah transformasi FTT ketika turunan matematis sinyal dari rezim warna tertentu menjadi "tertransformasi" ketika dikalikan secara individual dengan tingkat frekuensinya sendiri.
8) Penskalaan Domain:
   Ini adalah transformasi FTT di mana sinyal "Memanjang" atau "Meregang" secara matematis setara dengan sinyal "Terkompresi" atau "Terskala" dari turunan aslinya, dan sebaliknya juga berlaku.
9) Citra Nyata:
   Ini adalah transformasi FTT di mana nilai absolut berbasis matematis dari sinyal dihasilkan dari rezim warna dan juga simetris secara geometris terhadap titik asalnya dari koordinat piksel baik pada citra 2 Dimensi maupun citra 3 Dimensi yang dimaksud. Salah satu keuntungan utama dari properti ini adalah dapat membantu menyediakan

lebih banyak ruang penyimpanan untuk data kuantitatif maupun kualitatif yang digunakan oleh sistem ANN.

10) Teorema Parseval:

Ini melibatkan tingkat energi spesifik yang dihasilkan dari rezim warna yang disinari pada gambar 2-Dimensi atau gambar 3-Dimensi. Ini direpresentasikan sebagai penjumlahan matematis dari nilai kuadrat berbasis statistik.

Pentingnya Pasangan Berbasis Transformasi Fourier

Dalam subbagian ini, kami akan membahas secara lebih rinci apa yang disebut sebagai "Pasangan Transformasi Fourier" dan bagaimana pasangan ini dapat diimplementasikan ke dalam sistem ANN. Lebih spesifiknya, pasangan ini diturunkan dari properti-properti berikut:

1) Impuls:

Ini terdiri dari sebuah konstanta matematika yang merupakan penjumlahan dari semua transformasi FTT, sebagaimana telah diulas pada subbagian sebelumnya.

2) Impuls Bergeser:

Properti spesifik ini memiliki tingkat impuls yang bergeser, baik ke kanan, kiri, atas, atau bawah, jika ditempatkan pada kuadran matematika. Properti ini juga memanfaatkan berbagai jenis fase berbasis linear.

3) Filter Kotak:

Ini sebenarnya adalah Rata-Rata Bergerak berbasis statistik dari semua filter yang telah digunakan, dan secara matematis direpresentasikan sebagai berikut:

$$\mathrm{Box(x)} \ = \ \{1 \ \mathrm{if} \ |\mathrm{x}| \ < \ 1, 0 \ \mathrm{ELSE}\}.$$

Transformasi FTT-nya didasarkan pada algoritma matematika berikut:

$$\mathrm{SINC(w)} \ = \ \mathrm{SINw/W}.$$

Perlu dicatat bahwa kedua algoritma matematika di atas sebenarnya dapat memiliki jumlah tak terhingga jika iterasinya dikenal sebagai "Side Lobes" berbasis statistik. Selain itu, komponen SINC, sebagaimana direpresentasikan dalam algoritma matematika kedua, sebenarnya juga merupakan filter berbasis statistik, tetapi kelemahan utamanya adalah hanya dapat digunakan untuk filter yang memiliki nilai matematika yang jauh lebih rendah.

Penting untuk dicatat bahwa Pasangan Transformasi Fourier juga terdiri dari properti-properti berikut:

1) Properti Tent:

Ini adalah fungsi linear berbasis matematika sepotong-sepotong, dan direpresentasikan sebagai berikut:

$$\mathrm{Tent(x)} \ = \ \max(0, 1 \ - \ |\mathrm{X}|).$$

2) Properti Gaussian:

Ini adalah properti geometri, dan direpresentasikan secara matematis sebagai berikut:

$$G(x, 0) = (1/\text{AKAR KUADRAT } 2TT0 * c) * (e\text{^}x\text{^}2/2TT\text{^}2).$$

3) Properti Laplacian:
   Ini sebenarnya didasarkan pada properti matematika yang dikenal sebagai "Teorema Wavelet Gabor". Ini adalah hasil perkalian dari kosinus frekuensi tertentu (yang dilambangkan sebagai "w|0") dan fungsi matematika Gaussian (yang dilambangkan sebagai "0\"). Perlu dicatat bahwa properti spesifik ini memiliki sub-properti berikut:
   - Lebar Gaussian, yang juga dilambangkan sebagai "0\";
   - Penjumlahan dari dua lebar Gaussian terpisah, yang juga dilambangkan sebagai "0^ − 1". Properti ini sebenarnya berpusat secara statistik pada pusat koordinat piksel gambar 2-Dimensi atau gambar 3-Dimensi, dan ini dilambangkan sebagai "w = + − w0".
4) Properti Topeng Tak Tajam:
   Ini sebenarnya merupakan transformasi berbasis FTT lainnya, dan dapat digunakan secara optimal oleh sistem ANN pada tingkat frekuensi rezim warna yang jauh lebih tinggi.
5) Properti Sinc Berjendela:
   Properti ini paling ideal untuk sistem ANN yang menggunakan "Fungsi Respons" spesifik, yang mencoba memperkirakan filter lolos rendah apa pun yang dihasilkan oleh rezim warna. Properti spesifik ini direpresentasikan secara matematis sebagai berikut:

$$RCOS(x) = ½ * (1 + COS\ TT\ x) * BOX(x).$$

Pentingnya Transformasi Fourier 2-Dimensi

Perlu dicatat bahwa teknik FTT yang telah diulas sejauh ini dalam bab ini sebenarnya hanya dapat digunakan untuk sinyal yang secara matematis bersifat 1-Dimensi, yang kemudian dapat diterjemahkan lebih lanjut menjadi gambar 2-Dimensi, baik statis maupun dinamis. Dengan teknik semacam ini, bukan hanya tinggi atau lebar yang dipertimbangkan. Sebaliknya, semua orientasi vektor dapat dipertimbangkan. Hal ini dapat direpresentasikan secara matematis sebagai berikut:

$$S(x, y) = SIN * (WxX + WyY).$$

Versi konvolusionalnya secara matematis adalah sebagai berikut:

$$H(W2, Wy) = h(x, y) * e\text{^} - j(WzX + WzY) * (DxDy).$$

Versi diskritnya direpresentasikan secara matematis sebagai:

$$H(Kx, Ky) = 1/MN * (M - 1 \sum z = 0 * N - 1 \sum y = 0) * [h(x, y)e\text{^} - j2TT * Kx + KyZ/M, N]$$

Di mana:
M = lebar gambar 2 Dimensi;
N = tinggi gambar 2 Dimensi.

Dampak Teknik Penyaringan Weiner

Perlu dicatat bahwa teknik FTT tidak hanya sangat menguntungkan untuk mempelajari lebih lanjut karakteristik frekuensi berbagai rezim warna, tetapi juga dapat digunakan untuk membantu menganalisis pengelompokan keseluruhannya. Di sinilah konsep yang dikenal sebagai "Filter Wiener" berperan, dan dapat direpresentasikan secara matematis sebagai berikut:

$$\{|S(Wz, Wy)]\text{^}2]\} = Ps^{*}(Wz, Wy).$$

Untuk mengelompokkan semua rezim warna ke dalam satu kategori luas (bahkan satu subkategori), "Gaussian Noise Image" digunakan, dan secara matematis direpresentasikan sebagai berikut:

$$S^{*}(Wz, Wy).$$

Namun, terdapat pula algoritma matematika khusus untuk mengelompokkan subkategori, dan ini juga direpresentasikan sebagai berikut:

$$O(x, y) = s(x, y) + (n, y)$$

Di mana:
$S(x, y)$ = berbagai rezim warna yang akan dipecah menjadi berbagai subkategori;
$N(x, y)$ = Sinyal Aditif;
$o(x, y)$ = rezim warna utama yang telah dikelompokkan ke dalam satu kategori tertentu.

Meskipun teknik FTT pada dasarnya bersifat linear, teknik ini juga dapat diterapkan pada rezim warna yang juga kurvilinear ketika disinari ke gambar 2-Dimensi atau 3-Dimensi, baik yang bersifat statis maupun dinamis. Untuk mengakomodasi ketentuan khusus ini, algoritma matematika berikut juga harus digunakan:

$$0(Wx, Wy) = S(Wz, Wy) + N(Wz, Wy).$$

Akhirnya, satu kelompok rezim warna juga dapat ditumpangkan ke kelompok lain dengan teknik FTT dengan menggunakan persamaan matematika berikut:

$$0(Wx, Wy) = b(x, y) + s(x, y) + n(x, y).$$

Fungsi Transformasi Kosinus Diskrit

Transformasi Kosinus Diskrit, atau disingkat "DCT", sebenarnya juga dianggap sebagai bagian dari teknik FTT. Dalam hal ini, koordinat piksel gambar 2-Dimensi atau gambar 3-Dimensi dapat diperkecil menjadi beberapa "Blok" yang lebih kecil sehingga sistem ANN dapat memproses gambar-gambar ini dengan mudah dan cepat. Terdapat dua versi DCT yang berbeda, tergantung versi mana yang paling sesuai untuk keluaran yang dihitung oleh sistem ANN. Secara matematis, hal ini direpresentasikan sebagai berikut:

$$\text{For} 1-\text{DimensionalUses: } F(k) = N-1 \sum i = 0 \text{COS}[(TT/N(I+½)k)] \ ^* f(i).$$

Algoritma matematika di atas sebenarnya mengompresi lebih lanjut, atau mengodekan koordinat piksel gambar 2-Dimensi atau gambar 3-Dimensi menjadi mode berbasis linear.

$$\text{For } 2-\text{Dimensional Uses: } F(k,l) = N-1 \sum i = 0 \ N-1 \sum j = 0 \ \text{COS}[\text{TTN}(I+½)k)] + \text{COS}[\text{TTN}(I+½)l)] \ ^* f(I,j).$$

Perlu dicatat bahwa kedua algoritma matematika di atas sebagaimana yang baru saja dijelaskan juga dapat diterapkan pada rezim warna yang terpisah, tetapi tidak harus diterapkan sebagai satu kelompok utuh. Selain itu, kedua algoritma matematika ini semakin disempurnakan dengan menerapkan prinsip-prinsip matematika Gabor Wavelet, sebagaimana dijelaskan sebelumnya dalam bab ini. Faktanya, jenis optimasi baru ini membantu mengurangi jumlah total "Artefak Pemblokiran" yang dapat muncul pada gambar 2-Dimensi maupun 3-Dimensi yang dimaksud.

8.3 KONSEP PIRAMIDA

Sejauh ini, semua algoritma matematika dalam bab ini yang telah diulas secara mendalam hanya dapat bekerja bersama sistem ANN untuk menghitung masukan yang dimasukkan ke dalam masukan yang memiliki jenis nilai matematika yang sama. Namun, tergantung pada aplikasi spesifik yang digunakan untuk sistem ANN, seharusnya juga dimungkinkan untuk mengubah ukuran resolusi gambar 2 Dimensi atau gambar 3 Dimensi sebelum pemrosesan lebih lanjut dapat dilakukan oleh sistem ANN untuk menghitung keluaran yang diinginkan.

Misalnya, mungkin Anda ingin mengurangi lebih lanjut ukuran kumpulan data yang dimasukkan ke dalam sistem ANN (tidak masalah apakah berbasis kuantitatif atau kualitatif) sehingga keluaran yang diinginkan yang dihitung oleh sistem ANN akan tepat pada percobaan pertama, alih-alih harus terus-menerus mengubah jenis kumpulan data yang sama untuk akhirnya mencapai keluaran optimal yang dibutuhkan. Selain itu, sangat mungkin ukuran keseluruhan gambar 2 Dimensi atau 3 Dimensi harus diperkecil lebih lanjut (dalam hal ini, Anda akan melihat pemotongan lebih lanjut pada tinggi dan lebarnya) untuk lebih mengoptimalkan

kecepatan dan efisiensi sistem ANN, atau bahkan sekadar memberi lebih banyak ruang penyimpanan pada gambar 2 Dimensi atau gambar 3 Dimensi.

Selain itu, dalam hal Teknologi Biometrik, terutama dalam penggunaan Pengenalan Wajah, gambar berbasis Wajah tertentu harus ditemukan. Dalam kasus khusus ini, Eigenfaces biasanya paling banyak digunakan, tetapi diagram geometris berbasis "Piramida" juga dapat digunakan. Pada kenyataannya, penggunaan diagram semacam ini bisa lebih efektif, karena desainnya jauh lebih sederhana, sehingga basis data Sistem Pengenalan Wajah dapat dipindai jauh lebih cepat. Jika Eigenfaces digunakan, proses ini bisa memakan waktu lebih lama, karena sifatnya jauh lebih canggih daripada diagram berbasis piramida.

Keuntungan utama lain dari penggunaan diagram berbasis piramida adalah kemampuannya untuk mengintegrasikan gambar 2-Dimensi atau 3-Dimensi yang terpisah secara cepat dan mulus menjadi satu kesatuan yang utuh dan kohesif. Perlu dicatat juga bahwa sebagian dari diagram berbasis piramida ini juga dikenal sebagai "Wavelet", dan sesuai namanya, diagram ini juga didasarkan pada fondasi matematika Teori Wavelet Gabor. Penggunaan diagram berbasis piramida juga dapat digunakan untuk memisahkan gambar 2-Dimensi atau 3-Dimensi (secara teknis dikenal sebagai "Interpolasi"), atau untuk mengompresnya lebih lanjut untuk sistem ANN setelah dipisahkan (dikenal sebagai "Desimasi").

Terakhir, konsep yang dikenal sebagai "Piramida Multi-Resolusi" juga dapat digunakan, dan di sinilah hierarki yang spesifik dan mapan dapat diformulasikan dan dibuat dengan memanfaatkan berbagai jenis diagram berbasis piramida. Karena, sekali lagi, ini cenderung kurang rumit sifatnya, ini juga dapat digunakan oleh hampir semua aplikasi untuk sistem ANN untuk menghitung keluaran yang diinginkan, atau dibutuhkan.

Pentingnya Interpolasi

Pada subbagian sebelumnya, kita telah membahas apa itu "Interpolasi". Kita akan membahasnya lebih lanjut di subbagian ini. Pertama, algoritma matematika untuk merepresentasikannya adalah sebagai berikut:

$$G(I,j) = \sum k,I\, f(k,l) * h(I - rk, j - rl).$$

Algoritma ini sebenarnya dapat diterapkan secara langsung pada apa yang dikenal sebagai "Rumus Konvolusi Diskrit", yang juga telah diulas sebelumnya dalam bab ini. Untuk beberapa aplikasi yang lebih umum, rumus ini juga dapat disebut sebagai "Bentuk Filter Polifase". Dalam contoh khusus ini, bentuk khusus nilai matematika yang dikenal sebagai "Kernel" juga digunakan. Pertanyaan umum yang sering diajukan pada tahap ini adalah apa yang membuat sebuah "Kernel" yang baik, dari sudut pandang ilmiah? Banyak hal ini sangat bergantung pada aplikasi spesifik yang digunakan oleh sistem ANN, serta waktu pemrosesan dan komputasi yang juga disertakan. Berikut adalah beberapa karakteristik penting yang perlu dipertimbangkan lebih lanjut:

1) Interpolator Linier:

Ini digunakan untuk menghasilkan kurva berbentuk parabola, baik yang positif maupun negatif. Secara matematis, ini dapat direpresentasikan sebagai berikut:

$$\text{A Positive Parabola: Y} = \text{X\^2;}$$
$$\text{A Negative Parabola: Y} = \text{X\^} - 2.$$

Namun, kelemahan utamanya adalah dapat menciptakan "Kerutan" yang tidak diinginkan, baik pada gambar 2 Dimensi maupun gambar 3 Dimensi, baik yang statis maupun dinamis.

2) Kernel Aproksimasi:
Ini juga secara teknis disebut sebagai "Cubic B Spline". Algoritma ini sebenarnya dapat menghasilkan gambar 2-Dimensi atau 3-Dimensi yang "lebih lembut", di mana tingkat frekuensi tinggi yang dihasilkan oleh berbagai rezim warna dikurangi secara statistik sebelum dimasukkan ke dalam sistem ANN untuk menghitung tingkat keluaran yang diinginkan.

3) Kernel Bikubik:
Ini sebenarnya adalah jenis algoritma matematika yang sangat terspesialisasi, yang secara khusus diciptakan untuk gambar 2-Dimensi atau 3-Dimensi yang memiliki intensitas sangat tinggi dalam hal rezim warna yang digunakannya. Algoritma jenis khusus ini direpresentasikan sebagai berikut:

$$\text{H(x)} = \{1 - (\text{a} + 3)\text{x\^2} + (\text{a} + 2)\ |\text{x\^3}, \text{a}(|\text{x}| - 1) * (|\text{x}| - 2)\text{\^2}); 0 \text{ if } |\text{x}| < 1,$$
$$\text{if} < |\text{x}| < 2, \text{otherwise}\}$$

Di mana:
$\text{A} = \text{turunan dari di mana x} = 1.$

Namun, perlu dicatat bahwa dalam sistem ANN, nilai a = -0,5 sering digunakan, karena dianggap paling optimal. Nilai ini juga dapat disebut sebagai "Spline Reproduksi Kuadrat", yang di dalamnya fungsionalitas kuadrat dan linear dapat digabungkan. Seperti yang mungkin telah Anda perhatikan, istilah "Spline" cukup sering digunakan dalam bab ini. Untuk memberikan wawasan teknis lebih lanjut, ini adalah fungsi matematika spesifik yang terutama digunakan untuk menghitung "Interpolasi Nilai" berbasis fungsional dan data karena fungsi tersebut juga dapat menghitung turunan berbasis matematika. Fungsi ini juga banyak digunakan untuk membantu membuat piramida berbasis geometri. Namun, khususnya terkait dengan aplikasi Visi Komputer untuk sistem ANN, Spline biasanya digunakan untuk jenis operasi berikut:

- Pembuatan gambar 2-Dimensi atau 3-Dimensi Elastis;
- Pembuatan Estimasi Gerak (ini khususnya digunakan untuk kumpulan data berbasis video yang dimasukkan ke dalam sistem ANN;
- Pembuatan interpolasi berbasis permukaan.

Pentingnya Desimasi

Perlu dicatat pada titik ini bahwa teknik dan konsep interpolasi dapat digunakan untuk meningkatkan ukuran resolusi tertentu dari gambar 2 Dimensi atau gambar 3 Dimensi, yang keduanya digunakan oleh sistem ANN. Namun, kebalikan matematis dari hal ini dikenal sebagai "Desimasi", di mana ukuran resolusi gambar 2 Dimensi atau gambar 3 Dimensi sebenarnya berkurang baik dalam ukuran maupun cakupannya. Ada dua komponen matematis terpisah yang terkait dengan "Desimasi", yaitu sebagai berikut:

$$\text{The First Component: } G(I,j) = \sum k,I = \sum k,l * (k,l) * h(Ri - k,rj - l);$$
$$\text{The Second Component: } G(I,j) = \sum k,I = \sum k,l * (k,l) * h(i - k/r,j - l/r).$$

Berbagai jenis "Desimasi" juga dikenal sebagai "Filter", dan jenis-jenisnya adalah sebagai berikut:

1) Filter Berbasis Linear:
   Sesuai namanya, filter ini bersifat linear, berdasarkan rentang matematis dari [1, 2, 1].
2) Filter Binomial:
   Filter ini beroperasi pada rentang matematis dari [1, 4, 6, 4, 1]. Filter ini terutama digunakan untuk mengurangi frekuensi tambahan yang dihasilkan dari rezim warna yang disinari ke gambar 2-Dimensi atau 3-Dimensi, dan bahkan ketika frekuensi tersebut juga disinari ke diagram berbasis piramida, seperti yang dibahas panjang lebar sebelumnya dalam bab ini.
3) Filter Kubik:
   Filter ini beroperasi pada rentang matematis dari [-1 hingga -0,5].
4) Filter QMF:
   Tidak ada rentang matematis spesifik yang diberikan untuk filter ini, tetapi filter ini cukup banyak digunakan untuk apa yang secara khusus dikenal sebagai "Wavelet Denoising" baik untuk gambar 2-Dimensi maupun gambar 3-Dimensi yang dimaksud.

Pentingnya Representasi Multi-Level

Setelah kita secara mendalam menjabarkan fondasi teoretis untuk piramida berbasis geometri yang digunakan oleh sistem ANN dan Visi Komputer saat ini, kita akan mengulas lebih detail bagaimana piramida berbasis geometri ini dapat dibangun di subbagian ini. Dalam hal ini, mungkin salah satu fondasi yang paling terkenal dan paling dihormati untuk membangun piramida berbasis geometri tersebut adalah "Piramida Adelson dan Laplacian".

Untuk memulai konstruksi dengan metodologi khusus ini, gambar 2-Dimensi atau 3-Dimensi terlebih dahulu "dikaburkan" dengan faktor eksponensial matematis hampir dua. Ini disimpan dan juga digunakan untuk membentuk dan menciptakan fondasi. Penting untuk dicatat di sini bahwa ini adalah proses yang sepenuhnya otomatis dan berulang, dan hanya akan berhenti hingga puncak piramida berbasis geometri tercapai. Proses ini juga disebut sebagai "Piramida Oktaf".

Hal ini dapat direpresentasikan secara diagram sebagai berikut:

$$|C\ |B|A|B|C|$$

Di mana:

$B = ¼$

$C = ¼ - a/2.$

Namun, perlu dicatat pada titik ini bahwa "A" sebenarnya ditetapkan pada nilai matematika 3/8, yang merupakan titik optimal untuk sistem JST.

Hal ini dapat direpresentasikan secara diagram sebagai berikut:

$$1/16\ |1|4|6|4|1|.$$

Menarik untuk dicatat bahwa diagram matematika kedua sebenarnya jauh lebih mudah diimplementasikan ke dalam sistem JST daripada yang pertama, seperti yang baru saja dijelaskan. Kedua diagram ini juga dikenal sebagai "Piramida Gaussian", karena keduanya, pada suatu titik waktu dalam siklus iteratif, konvergen.

Ada juga teknik piramida diagram geometri lainnya, yaitu "Piramida Laplacian". Ketika teori spesifik ini pertama kali dirumuskan, dasar piramida pertama kali dibangun menggunakan versi yang disederhanakan dari piramida geometri pertama yang awalnya dibuat. Tingkat yang lebih rendah ini kemudian dikurangi secara matematis untuk menciptakan apa yang dikenal sebagai "Citra Laplacian Band Pass". Keuntungan utama dari hal ini adalah sistem ANN yang menggunakannya menyimpannya secara permanen dan menghapusnya jika diperlukan.

Bahkan, deskripsi di atas secara teknis dikenal sebagai "Rekonstruksi Sempurna", tergantung pada aplikasi spesifik yang digunakan. Terdapat juga variasi lain dari jenis piramida berbasis geometri ini, yang bahkan dapat dibuat dari snapshot awal yang telah diambil dari citra 2-Dimensi atau citra 3-Dimensi, baik statis maupun dinamis. Piramida Geometri Laplacian pertama dapat direpresentasikan secara matematis sebagai berikut:

$$DoG\{I;\ 01, 02\} = G01\ ^*\ I - G02\ ^*\ I = (G01 - G02)\ ^*\ I.$$

Varian tersebut, seperti yang baru saja dijelaskan sebelumnya, juga dapat direpresentasikan secara matematis sebagai berikut:

$$V2 = (02/0x\text{^}2) + (02/0y\text{^}2).$$

Teknik Piramida Geometri Laplacian sebenarnya merupakan teknik yang paling disukai untuk digunakan di sebagian besar jenis sistem ANN. Misalnya, teknik ini dapat digunakan untuk menganalisis lebih lanjut secara lebih rinci tepi gambar 2-Dimensi maupun gambar 3-Dimensi. Ada turunan matematika lain dari teknik ini, yang dikenal sebagai "Piramida Setengah Oktaf". Teknik ini sebenarnya pertama kali diciptakan pada tahun 1984, dan saat itu dikenal

secara khusus sebagai "Transformasi Selisih Low Pass", atau disingkat "DOLP". Namun, teknik khusus ini tidak banyak digunakan dalam aplikasi sistem ANN saat ini.

Namun, ketika teknik yang disebutkan di atas dikombinasikan lebih lanjut dengan teknik statistik yang dikenal sebagai "Pengambilan Sampel Papan Catur", keluaran dari sistem ANN (yang menggunakan teknik gabungan ini) dikenal sebagai pengambilan sampel statistik berbasis "Quincux".

8.4 DASAR-DASAR WAVELET

Perlu dicatat bahwa meskipun piramida berbasis geometri sebenarnya paling disukai untuk digunakan oleh sistem ANN saat ini, terdapat alternatif lain. Alternatif ini dikenal secara khusus sebagai "Wavelet", dan fondasi teoretisnya berasal dari Matematika Wavelet Gabor. Ini adalah filter yang sangat terspesialisasi yang dapat melokalisasi rezim warna yang disinari pada gambar 2-Dimensi atau gambar 3-Dimensi (serta frekuensinya masing-masing). Wavelet juga dapat didefinisikan lebih lanjut secara matematis sebagai hierarki spesifik dari berbagai skala yang telah dirancang dengan permutasi tertentu untuk lebih menghaluskan frekuensi tersebut menjadi berbagai jenis subkomponen, yang dapat berkorelasi sangat erat secara statistik dengan piramida berbasis geometri, yang telah dijelaskan secara rinci pada subbagian terakhir bab ini. Penggunaan Filter Gabor, sebenarnya, telah ada sejak akhir 1980-an, dan bahkan hingga awal 1990-an.

Perlu dicatat bahwa penggunaan Wavelet sangat umum di bidang Grafika Komputer.Di bidang ini, Wavelet dapat digunakan untuk melakukan berbagai macam "Pemrosesan Geometris Multi-Resolusi" baik untuk gambar 2 Dimensi maupun 3 Dimensi, yang akan digunakan oleh sistem ANN. Pertanyaan yang sering muncul di kalangan pakar Visi Komputer yang secara khusus menangani sistem ANN adalah, apa perbedaan utama antara piramida berbasis geometri dan Wavelet, seperti yang baru saja dijelaskan? Pada piramida berbasis geometri, koordinat piksel yang digunakan seringkali lebih banyak daripada yang biasanya dianggap perlu, tetapi pada piramida berbasis geometri, hanya jumlah minimum koordinat piksel yang digunakan. Manfaat utama dari hal ini adalah integritas gambar 2 Dimensi atau 3 Dimensi tetap terjaga, terlepas dari semua proses yang dilaluinya dengan Sistem ANN.

Faktanya, untuk mencapai tugas spesifik ini, Wavelet menggunakan apa yang dikenal sebagai "Bingkai Ketat". Mereka juga lebih banyak menggunakan orientasi vektor berbasis matematika untuk membantu mengoptimalkan prosedur spesifik ini. Lebih lanjut, saat ini, hanya Wavelet 2-Dimensi yang digunakan oleh sistem ANN saat ini, tidak lebih dari itu, meskipun prospek penggunaan Wavelet 3-Dimensi saat ini masih dalam tahap pengembangan.

Proses pembuatan Wavelet 2-Dimensi yang spesifik adalah sebagai berikut:

- "High Pass Filter" pertama kali dibuat, di mana ruang ¾ inci ditempatkan ke dalam gambar 2-Dimensi;
- "Low Pass Filter" selanjutnya dibuat, di mana filter yang lebih rendah dibuat, hanya menggunakan ruang ¼ inci untuk memisahkannya lebih lanjut;

- Filter yang dihasilkan dari dua langkah di atas kemudian dibagi lagi menjadi dua sub-tahap yang terpisah dan berbeda;
- Dua sub-tahap di atas kemudian disebut "High-High" (juga dikenal sebagai "HH"), dan "High-Low" (juga dikenal sebagai "HL");
- Setelah langkah terakhir selesai, sub-tahap baru kemudian dibuat, dan ini dikenal secara khusus sebagai frekuensi "High-Low" (juga dikenal sebagai "HL");
- Frekuensi "HL" dan "LH" inilah yang kemudian ditransmisikan ke sumbu horizontal dan vertikal gambar 2-Dimensi;
- Dari sudut pandang matematis, tingkat frekuensi "HH" inilah yang dapat mengambil kedua frekuensi di atas (seperti yang baru saja dijelaskan), dan menjumlahkannya hanya dengan mengambil turunan timbal baliknya dan menjumlahkannya.

Pertanyaan yang sering ditanyakan lagi adalah, bagaimana ketiga frekuensi ini dihitung secara matematis satu sama lain? Ini telah menjadi isu yang dibahas bahkan selama dua puluh tahun terakhir di bidang Visi Komputer. Jawaban utama untuk pertanyaan ini sebenarnya bergantung pada jenis aplikasi yang digunakan oleh sistem ANN, dengan mempertimbangkan lebih lanjut permutasi berbasis statistik utama berikut:

- Apakah Wavelet akan dirancang untuk Kompresi gambar 2-Dimensi;
- Analisis Gambar seperti apa yang akan dilakukan pada gambar 2-Dimensi yang dimaksud;
- Apakah akan dilakukan "Denoising" pada gambar 2-Dimensi yang dimaksud?

Faktanya, bahkan bagi sebagian profesional Visi Komputer, gagasan untuk membuat dan menerapkan Wavelet tertentu ke dalam sistem ANN dapat dianggap sebagai seni yang sangat "rumit". Dengan kata lain, tidak ada pendekatan berbasis kuantitatif untuk melaksanakan tugas ini; semuanya bergantung pada permutasi yang harus diputuskan. Namun tentu saja, pada akhirnya, hal ini akan sangat bergantung lagi pada persyaratan yang dibutuhkan untuk menghasilkan keluaran yang diinginkan oleh sistem ANN. Namun, jika seseorang mencari aturan praktis yang cepat untuk menyelesaikan tugas ini, maka sangat penting untuk mengambil langkah-langkah berikut:

- Membagi tiga tingkat frekuensi yang berbeda (seperti yang baru saja dijelaskan sebelumnya) menjadi nilai berbasis matematika genap dan ganjil;
- Kemudian, gunakan nilai-nilai yang disebutkan di atas untuk membalik urutan ketiga tingkat frekuensi ini secara spesifik;
- Setelah kedua langkah di atas tercapai, langkah-langkah ini kemudian dikenal sebagai "Wavelet Terangkat". Namun, prosedur yang baru saja dijelaskan ini juga dikenal sebagai "Skema Pengangkatan untuk Wavelet Generasi Kedua".

Alasan utama mengapa nama terakhir ini dipilih adalah karena teknik umum yang disebutkan di atas juga dapat diterapkan secara mulus ke berbagai jenis "Topologi Sampling" berbasis statistik lainnya, yang juga dapat dimasukkan ke dalam sistem ANN. Hal ini sebenarnya bekerja cukup baik untuk jenis aplikasi spesifik tersebut (yang akan digunakan oleh sistem ANN) untuk apa yang secara teknis dikenal sebagai "Manipulasi Permukaan Resolusi Berlapis-lapis". Bahkan, turunannya ditetapkan sebagai "Wavelet Tertimbang Terangkat",

karena koefisien berbasis statistik yang dimanfaatkan darinya dapat digunakan untuk sebagian besar jenis aplikasi yang hanya menggunakan gambar 2-Dimensi.

Namun, perlu dicatat bahwa jika metodologi tiga langkah (seperti yang baru saja dijelaskan sebelumnya) tidak dapat digunakan karena alasan apa pun, maka ada teorema lain yang dapat diterapkan untuk menyelesaikan situasi ini. Teori ini secara khusus dikenal sebagai "Implementasi Frekuensi Radial Piramida", tetapi versi singkat dari teori ini adalah "Piramida yang Dapat Dikendalikan". Ia memiliki karakteristik berikut:

- Perhitungan matematis yang diturunkan dari penggunaan teorema spesifik ini dianggap "Overcomplete" secara statistik, baik secara alami maupun berdasarkan desain;
- Ia memiliki berbagai jenis orientasi berbasis matematis dan berbasis vektor yang dapat dengan mudah dan otomatis diambil oleh sistem ANN untuk menghitung keluaran yang diinginkan;
- Ia juga memiliki apa yang dikenal sebagai fungsi matematika berbasis "Sintesis", yang secara teknis dapat dibalik atau dibalik, tergantung pada jenis aplikasi yang digunakan sistem ANN secara spesifik;
- Hasil akhirnya menjadi apa yang dikenal sebagai "Piramida yang Dapat Dikendalikan", dan ini sebenarnya cukup umum digunakan dalam melakukan analisis berbasis struktural.

Terakhir, "Piramida yang Dapat Dikendalikan" paling cocok untuk jenis analisis dan studi berikut yang dapat dilakukan pada citra 2 Dimensi:

- Analisis berbasis tekstur;
- Analisis berbasis sintesis;
- “Pengurangan Noise pada Gambar” (konsep ini baru saja disebutkan di beberapa subbagian terakhir).

8.5 PENTINGNYA TRANSFORMASI BERBASIS GEOMETRI

Transformasi semacam ini, ketika dilakukan pada bidang geometris (seperti bidang berbasis Kartesius), juga dapat digunakan untuk meningkatkan atau bahkan mengoptimalkan resolusi keseluruhan gambar 2-Dimensi atau gambar 3-Dimensi, mana pun yang direncanakan untuk diterapkan ke dalam sistem ANN guna menghitung keluaran yang diinginkan. Transformasi ini secara teknis juga dapat disebut sebagai "Rotasi Gambar" spesifik, atau bahkan "Lengkungan Umum". Namun, tidak seperti berbagai teknik Pemrosesan Titik yang telah dibahas secara mendetail di beberapa subbagian terakhir, prosedur semacam ini biasanya dapat digunakan untuk seluruh rentang koordinat piksel, di seluruh gambar 2-Dimensi atau 3-Dimensi. Proses khusus ini dapat direpresentasikan secara matematis sebagai berikut:

$$G(x) = h[f(x)].$$

Algoritma matematika di atas juga dapat digunakan untuk rentang nilai numerik berbasis matematika. Jika sistem ANN ingin berfokus pada "Domain" tertentu atau pengelompokan koordinat piksel yang berada dalam citra 2-Dimensi atau citra 3-Dimensi, maka algoritma matematika berikut harus digunakan:

$$G(X) = f[H(X)].$$

Metode utama yang paling banyak digunakan dalam hal ini adalah "Transformasi Parametrik", dan hal ini akan dibahas lebih lanjut di subbagian berikutnya.

Dampak Transformasi Parametrik

Teknik spesifik ini secara harfiah dapat diterapkan pada seluruh rentang koordinat piksel, baik pada citra 2-Dimensi maupun 3-Dimensi yang dimaksud. Salah satu keuntungan terbesar dari penggunaan teknik "Transformasi Parametrik" adalah hanya membutuhkan sekumpulan permutasi berbasis matematika yang sangat kecil dan terbatas yang perlu diformulasikan dan diimplementasikan ke dalam sistem ANN. Salah satu pertanyaan kunci yang muncul pada titik ini adalah, bagaimana koordinat piksel baru dapat dihitung dari citra 2 Dimensi atau citra 3 Dimensi asli (keduanya dapat dilambangkan sebagai "f(x)"), dan menggunakannya untuk membuat citra 2 Dimensi atau 3 Dimensi yang sama sekali baru (keduanya juga dapat dilambangkan sebagai "$g(x)$") hanya dengan menggunakan model transformasi parametrik umum? Perlu diingat bahwa algoritma matematika umum untuk teknik transformasi parametrik direpresentasikan sebagai berikut:

$$X' = h(x).$$

Hal di atas secara teknis juga disebut sebagai "Forward Warping", tetapi memiliki sejumlah kelemahan serius, yaitu sebagai berikut:

- Anda tidak dapat begitu saja "Menyalin dan Menempel" koordinat piksel (direpresentasikan sebagai "f(x)") ke lokasi yang lebih baru (dilambangkan sebagai "g") ke dalam gambar 2 Dimensi atau gambar 3 Dimensi yang baru diturunkan;
- Tidak terdapat cukup nilai matematika berbasis non-integer yang terdefinisi dengan baik.

Ada beberapa solusi untuk hal ini, yang biasanya meliputi hal-hal berikut:

1) Nilai matematika "x'" dapat dibulatkan ke atas untuk menyalin dan menempelkan koordinat piksel asli ke dalam gambar 2 Dimensi dan/atau gambar 3 Dimensi yang baru diturunkan;
2) Hal di atas juga dapat didistribusikan secara statistik ke gambar piksel berbasis kuadran terdekat pada gambar 2 Dimensi atau gambar 3 Dimensi yang baru.

Perlu dicatat bahwa langkah terakhir ini juga disebut sebagai "Splatting". Namun, terkadang hal ini dapat menyebabkan "Keburaman" yang cukup signifikan, baik pada gambar 2 Dimensi maupun 3 Dimensi. Ada masalah besar lain yang dapat terjadi, yaitu berbagai jenis "Retakan" juga dapat muncul pada gambar 2 Dimensi atau 3 Dimensi yang baru dibuat.

Namun, ada solusi lain untuk mengatasi kedua masalah ini, yang secara teknis dikenal sebagai "Sampling Terbalik". Sesuai namanya, teknik ini sebenarnyakebalikan dari "Forward Sampling", dan dalam teknik khusus ini, koordinat piksel pada citra 2-Dimensi atau citra 3-Dimensi yang baru diturunkan dapat "Di-Reverse Sampled" kembali ke citra 2-Dimensi atau citra 3-Dimensi asli yang dimaksud. Algoritma matematika untuk teknik "Inverse Sampling" adalah sebagai berikut:

$$G(x, y) = \sum f(k, l) * h(x - k, y - l)$$

Di mana:
(x, y) = koordinat subpiksel;
$H(x, y)$ = nilai matematika Interpolasi atau Smoothing.

Selain itu, Analisis Fourier dapat diterapkan pada algoritma matematika di atas untuk optimasi dan penyempurnaan lebih lanjut. Algoritma ini dapat direpresentasikan secara matematis sebagai berikut:

$$G(Ax) \text{ U } \Diamond \ |A|\text{^} - 1 * G(A\text{^} - Tf).$$

Tujuan utama buku ini adalah memberikan panduan kepada CIO dan/atau CISO yang dapat mereka gunakan dalam proses pengambilan keputusan spesifik mereka dalam hal pengadaan dan penerapan Sistem Kecerdasan Buatan. Tujuan utama di balik sistem Kecerdasan Buatan (AI) adalah untuk meniru otak manusia dan mencoba mereplikasi proses berpikir dan penalarannya ke aplikasi dunia nyata. Dengan demikian, AI dapat digunakan oleh hampir semua industri, tetapi buku ini diarahkan untuk Keamanan Siber.

Dalam hal ini, AI masih menemukan tempat permanennya di industri ini. Meskipun AI memang menjanjikan untuk berbagai aplikasi di masa depan, aplikasi utamanya terbagi dalam dua cabang spesifik Kecerdasan Buatan:

- Memfilter dan memilah semua peringatan dan imbauan yang diterima Tim Keamanan Siber. Proses ini sebagian besar merupakan proses otomatis, tetapi keuntungan utamanya adalah dapat digunakan untuk menyaring Positif Palsu yang muncul berkali-kali setiap hari, sehingga mengurangi masalah yang dikenal sebagai "Kelelahan Peringatan". Dalam hal ini, Tim Keamanan TI dapat berfokus untuk merespons dengan cepat hanya peringatan dan ancaman yang nyata dan sah.
- Proses ini juga dapat digunakan untuk mengotomatiskan proses yang terlibat dalam latihan Perburuan Ancaman dan Pengujian Penetrasi. Manfaat utamanya adalah Tim Merah, Biru, dan Ungu dapat berfokus pada gambaran besar klien mereka, sementara proses yang lebih rutin dan umum dapat sepenuhnya diotomatisasi.
- Kecerdasan Buatan juga dapat digunakan untuk membantu memodelkan Lanskap Ancaman Keamanan Siber. Dalam hal ini, terdapat banyak sekali data yang harus digunakan jika hal ini dimodelkan secara manual. Tidak hanya itu, tim Keamanan TI

akan membutuhkan waktu berjam-jam, bahkan berhari-hari, untuk memprediksi secara akurat dan pasti apa yang mungkin terjadi di masa mendatang. Dengan jeda waktu yang sangat lama ini, prediksi yang telah dibuat akan segera menjadi usang, dan upaya yang lebih besar harus dicurahkan kembali untuk menghasilkan prediksi yang memadai tentang Lanskap Ancaman Keamanan Siber. Selama waktu yang hilang ini, semakin banyak varian ancaman akan muncul, sehingga waktu yang tersita untuk memeranginya akan semakin berkurang, sehingga menempatkan bisnis, pelanggan, serta aset digital mereka pada risiko yang sangat besar, terutama dalam hal pencurian dan/atau pembajakan dataset Informasi Identitas Pribadi (PII). Namun, dengan menggunakan konsep dan prinsip Kecerdasan Buatan, pemodelan ini dapat dilakukan secara real-time, dengan akurasi yang jauh lebih tinggi daripada yang dapat dicapai manusia. Hasilnya, tim Keamanan TI tidak hanya dapat berfokus pada penanggulangan varian ancaman harian secara real-time, tetapi juga dapat mengembangkan lini pertahanan baru untuk memerangi varian ancaman tersebut serta ancaman apa pun yang akan datang. Keuntungan utama lain dari penggunaan Kecerdasan Buatan dalam hal ini adalah ia dapat mengonsumsi sejumlah besar data dan informasi, dan menganalisisnya hanya dalam hitungan beberapa detik, paling lama.

DAFTAR PUSTAKA

Abd Alkareem, M. H. B., Nasif, F. Q., Ahmed, S. R., Miran, L. D., Algburi, S., & ALmashhadany, M. T. (2023, November). Linguistics for crimes in the world by AI-based cyber security. In 2023 7th International Symposium on Innovative Approaches in Smart Technologies (ISAS) (pp. 1-5). IEEE.

Abdullahi, M., Baashar, Y., Alhussian, H., Alwadain, A., Aziz, N., Capretz, L. F., & Abdulkadir, S. J. (2022). Detecting cybersecurity attacks in internet of things using artificial intelligence methods: A systematic literature review. Electronics, 11(2), 198.

Adewusi, A. O., Okoli, U. I., Olorunsogo, T., Adaga, E., Daraojimba, D. O., & Obi, O. C. (2024). Artificial intelligence in cybersecurity: Protecting national infrastructure: A USA. World Journal of Advanced Research and Reviews, 21(1), 2263-2275.

Aloqaily, M., Kanhere, S., Bellavista, P., & Nogueira, M. (2022). Special issue on cybersecurity management in the era of AI. Journal of Network and Systems Management, 30(3), 39.

Anandita Iyer, A., & Umadevi, K. S. (2023). Role of AI and its impact on the development of cyber security applications. In Artificial Intelligence and Cyber Security in Industry 4.0 (pp. 23-46). Singapore: Springer Nature Singapore.

Ansari, M. F., Dash, B., Sharma, P., & Yathiraju, N. (2022). The impact and limitations of artificial intelligence in cybersecurity: a literature review. International Journal of Advanced Research in Computer and Communication Engineering.

Awadallah, A., Eledlebi, K., Zemerly, M. J., Puthal, D., Damiani, E., Taha, K., ... & Yeun, C. Y. (2024). Artificial intelligence-based cybersecurity for the metaverse: Research challenges and opportunities. IEEE Communications Surveys & Tutorials, 27(2), 1008-1052.

Camacho, N. G. (2024). The role of AI in cybersecurity: Addressing threats in the digital age. Journal of Artificial Intelligence General science (JAIGS) ISSN: 3006-4023, 3(1), 143-154.

Capuano, N., Fenza, G., Loia, V., & Stanzione, C. (2022). Explainable artificial intelligence in cybersecurity: A survey. Ieee Access, 10, 93575-93600.

Chakraborty, A., Biswas, A., & Khan, A. K. (2023). Artificial intelligence for cybersecurity: Threats, attacks and mitigation. In Artificial intelligence for societal issues (pp. 3-25). Cham: Springer International Publishing.

Chakraborty, C., Nagarajan, S. M., Devarajan, G. G., Ramana, T. V., & Mohanty, R. (2023). Intelligent AI-based healthcare cyber security system using multi-source transfer learning method. ACM Transactions on Sensor Networks.

Charmet, F., Tanuwidjaja, H. C., Ayoubi, S., Gimenez, P. F., Han, Y., Jmila, H., ... & Zhang, Z. (2022). Explainable artificial intelligence for cybersecurity: a literature survey. Annals of Telecommunications, 77(11), 789-812.

Dash, B., Ansari, M. F., Sharma, P., & Ali, A. (2022). Threats and opportunities with AI-based cyber security intrusion detection: a review. International Journal of Software Engineering & Applications (IJSEA), 13(5).

De Azambuja, A. J. G., Plesker, C., Schützer, K., Anderl, R., Schleich, B., & Almeida, V. R. (2023). Artificial intelligence-based cyber security in the context of industry 4.0—a survey. Electronics, 12(8), 1920.

Faraji, M. R., Shikder, F., Hasan, M. H., Islam, M. M., & Akter, U. K. (2024). Examining the role of artificial intelligence in cyber security (CS): a systematic review for preventing prospective solutions in financial transactions. International Journal, 5(10), 4766-4782.

Garcia, A. B., Babiceanu, R. F., & Seker, R. (2021, April). Artificial intelligence and machine learning approaches for aviation cybersecurity: An overview. In 2021 integrated communications navigation and surveillance conference (ICNS) (pp. 1-8). IEEE.

Ghillani, D. (2022). Deep learning and artificial intelligence framework to improve the cyber security. Authorea Preprints.

Goswami, M. (2024). AI-based anomaly detection for real-time cybersecurity. International journal of research and review techniques, 3(1), 45-53.

Guembe, B., Azeta, A., Misra, S., Osamor, V. C., Fernandez-Sanz, L., & Pospelova, V. (2022). The emerging threat of ai-driven cyber attacks: A review. Applied Artificial Intelligence, 36(1), 2037254.

Jada, I., & Mayayise, T. O. (2024). The impact of artificial intelligence on organisational cyber security: An outcome of a systematic literature review. Data and Information Management, 8(2), 100063.

Das, R., & Sandhane, R. (2021, July). Artificial intelligence in cyber security. In Journal of Physics: Conference Series (Vol. 1964, No. 4, p. 042072). IOP Publishing.

Jimmy, F. (2021). Emerging threats: The latest cybersecurity risks and the role of artificial intelligence in enhancing cybersecurity defenses. Valley International Journal Digital Library, 1, 564-74.

Kaur, R., Gabrijelčič, D., & Klobučar, T. (2023). Artificial intelligence for cybersecurity: Literature review and future research directions. Information Fusion, 97, 101804.

Khan, M. I., Arif, A., & Khan, A. R. A. (2024). AI-Driven Threat Detection: A Brief Overview of AI Techniques in Cybersecurity. BIN: Bulletin of Informatics, 2(2), 248-61.

Khan, M. I., Arif, A., & Khan, A. R. A. (2024). The most recent advances and uses of AI in cybersecurity. BULLET: Jurnal Multidisiplin Ilmu, 3(4), 566-578.

Kolluri, V. (2024). An extensive investigation into guardians of the digital realm: AI-driven antivirus and cyber threat intelligence. International Journal of Advanced Research and Interdisciplinary Scientific Endeavours, 1(2), 71-77.

Kuzlu, M., Fair, C., & Guler, O. (2021). Role of artificial intelligence in the Internet of Things (IoT) cybersecurity. Discover Internet of things, 1(1), 7.

Lysenko, S., Bobro, N., Korsunova, K., Vasylchyshyn, O., & Tatarchenko, Y. (2024). The role of artificial intelligence in cybersecurity: Automation of protection and detection of threats. Economic Affairs, 69, 43-51.

Malatji, M., & Tolah, A. (2025). Artificial intelligence (AI) cybersecurity dimensions: a comprehensive framework for understanding adversarial and offensive AI. AI and Ethics, 5(2), 883-910.

Manoharan, A., & Sarker, M. (2023). Revolutionizing cybersecurity: Unleashing the power of artificial intelligence and machine learning for next-generation threat detection. DOI: https://www. doi. org/10.56726/IRJMETS32644, 1.

Markevych, M., & Dawson, M. (2023, June). A review of enhancing intrusion detection systems for cybersecurity using artificial intelligence (ai). In International conference knowledge-based organization (Vol. 29, No. 3, pp. 30-37).

Mishra, S. (2023). Exploring the impact of AI-based cyber security financial sector management. Applied Sciences, 13(10), 5875.

Ozkan-Okay, M., Akin, E., Aslan, Ö., Kosunalp, S., Iliev, T., Stoyanov, I., & Beloev, I. (2024). A comprehensive survey: Evaluating the efficiency of artificial intelligence and machine learning techniques on cyber security solutions. IEEe Access, 12, 12229-12256.

Perumal, A. P., Chintale, P., Molleti, R., & Desaboyina, G. (2024). Risk Assessment of Artificial Intelligence Systems in Cybersecurity. American Journal of Science and Learning for Development, 3(7), 49-60.

Piplai, A., Kotal, A., Mohseni, S., Gaur, M., Mittal, S., & Joshi, A. (2023). Knowledge-enhanced neurosymbolic artificial intelligence for cybersecurity and privacy. IEEE Internet Computing, 27(5), 43-48.

Polamarasetti, A., Vadisetty, R., Velaga, V., Routhu, K., Sadaram, G., Boppana, S. B., & Vangala, S. R. (2023). Enhancing Cybersecurity Architectures with Artificial Intelligence (AI): A Framework for Automated Threat Intelligence Detection System. Universal Library of Engineering Technology, (Issue).

Pooyandeh, M., Han, K. J., & Sohn, I. (2022). Cybersecurity in the AI-Based metaverse: A survey. Applied Sciences, 12(24), 12993.

Prasad, S. G., Sharmila, V. C., & Badrinarayanan, M. K. (2023, May). Role of artificial intelligence based chat generative pre-trained transformer (chatgpt) in cyber security. In 2023 2nd International Conference on Applied Artificial Intelligence and Computing (ICAAIC) (pp. 107-114). IEEE.

Rizvi, M. (2023). Enhancing cybersecurity: The power of artificial intelligence in threat detection and prevention. International Journal of Advanced Engineering Research and Science, 10(5), 055-060.

Salem, A. H., Azzam, S. M., Emam, O. E., & Abohany, A. A. (2024). Advancing cybersecurity: a comprehensive review of AI-driven detection techniques. Journal of Big Data, 11(1), 105.

Sarker, I. H. (2023). Multi-aspects AI-based modeling and adversarial learning for cybersecurity intelligence and robustness: A comprehensive overview. Security and Privacy, 6(5), e295.

Sarker, I. H., Furhad, M. H., & Nowrozy, R. (2021). Ai-driven cybersecurity: an overview, security intelligence modeling and research directions. SN Computer Science, 2(3), 173.

Schmitt, M. (2023). Securing the digital world: Protecting smart infrastructures and digital industries with artificial intelligence (AI)-enabled malware and intrusion detection. Journal of Industrial Information Integration, 36, 100520.

Sontan, A. D., & Samuel, S. V. (2024). The intersection of Artificial Intelligence and cybersecurity: Challenges and opportunities. World Journal of Advanced Research and Reviews, 21(2), 1720-1736.

Tao, F., Akhtar, M. S., & Jiayuan, Z. (2021). The future of artificial intelligence in cybersecurity: A comprehensive survey. EAI Endorsed Transactions on Creative Technologies, 8(28).

Yaseen, A. (2023). AI-driven threat detection and response: A paradigm shift in cybersecurity. International Journal of Information and Cybersecurity, 7(12), 25-43.

Zhang, Z., Al Hamadi, H., Damiani, E., Yeun, C. Y., & Taher, F. (2022). Explainable artificial intelligence applications in cyber security: State-of-the-art in research. IEEe Access, 10, 93104-93139.

Zhang, Z., Ning, H., Shi, F., Farha, F., Xu, Y., Xu, J., ... & Choo, K. K. R. (2022). Artificial intelligence in cyber security: research advances, challenges, and opportunities. Artificial Intelligence Review, 55(2), 1029-1053.

CYBERSECURITY BERBASIS KECERDASAN BUATAN (AI)

Dr. Joseph Teguh Santoso, S.Kom, M.Kom.

BIODATA PENULIS

Dr. Joseph Teguh Santoso, M.Kom memiliki Jabatan Akademik Lektor Kepala dan praktisi industri yang berpengalaman. Saat ini menjabat sebagai Rektor Universitas Sains dan Teknologi Komputer (Universitas STEKOM), salah satu universitas terkemuka di Jawa Tengah, Indonesia. Dengan pengalaman lebih dari 13 tahun di dunia bisnis dan praktisi industri di China, beliau membawa perspektif global dan inovasi yang signifikan ke dalam dunia akademis.Sebagai seorang entrepreneur, penulis adalah pencipta TopLoker.com, sebuah platform inovatif yang merevolusi cara mencari dan menawarkan pekerjaan. TopLoker.com adalah portal lowongan bursa kerja terbesar di Indonesia, khusus untuk pendidikan SMA/SMK sederajat. TopLoker.com telah mendapatkan penghargaan sebagai juara 1 Startup4industry 2022 oleh Kementerian Perindustrian Republik Indonesia. Kontribusi Dr. Joseph dalam menyediakan akses pekerjaan yang luas bagi lulusan SMA/SMK telah membantu banyak individu menemukan peluang kerja yang sesuai dengan keahlian mereka. Selain itu, Dr. Joseph Teguh Santoso, M.Kom adalah pendiri dari dua organisasi yaitu (1) organisasi guru/pendidik PTIC (Perkumpulan Teacherpreneur Indonesia Cerdas) yang bertujuan untuk meningkatkan kualitas pendidikan dan kesejahteraan guru/pendidik dengan wawasan entrepreneurship, serta (2) organisasi industri PERKIVI (Perkumpulan Komunitas Industri dan Vokasi Indonesia) yang berfokus pada pengembangan link and match antara industri dan dunia pendidikan. Sebagai Rektor, Dr. Joseph Teguh Santoso, M.Kom memiliki kepemimpinan yang berorientasi pada hasil, dan berkomitmen untuk mendorong kemajuan Universitas Sains dan Teknologi Komputer (Universitas STEKOM). Saat ini Universitas STEKOM telah mengalami transformasi positif dalam peningkatan kualitas pendidikan, perluasan fasilitas, serta penguatan kemitraan Perguruan Tinggi Nasional dan Internasional. Beliau memprioritaskan pengembangan sumber daya manusia dan penelitian, serta memastikan bahwa universitas berada di garis depan dalam inovasi dan teknologi untuk mencapai tujuan akhir, yaitu lulusan yang mampu bekerja dan sukses setelah lulus. Dr. Joseph Teguh Santoso, M.Kom sering diundang sebagai pembicara di berbagai konferensi nasional maupun internasional dan telah menerima berbagai penghargaan atas dedikasinya dalam bidang pendidikan, industri, dan kewirausahaan.

YAYASAN PRIMA AGUS TEKNIK

PENERBIT :
YAYASAN PRIMA AGUS TEKNIK
Jl. Majapahit No. 605 Semarang
Telp. (024) 6723456. Fax. 024-6710144
Email : penerbit_ypat@stekom.ac.id

ISBN 978-634-7227-40-9 (PDF)
9 786347 227409