YAYASAN PRIMA AGUS TEKNIK

DEEP LEARNING DENGAN PYTHON

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

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

DEEP LEARNING DENGAN PYTHON

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 978-634-7227-11-9 (PDF)
9 786347 227119

DEEP LEARNING DENGAN PYTHON

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

ISBN : 978-634-7227-11-9

Editor :
Dr. Ir. 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 syukur penulis panjatkan ke hadirat Tuhan Yang Maha Esa atas rahmat dan karunia-Nya sehingga buku berjudul "***Deep Learning Dengan Python***" ini dapat diselesaikan dengan baik. Buku ini hadir sebagai upaya memberikan pemahaman yang komprehensif mengenai konsep, teknik, dan implementasi deep learning menggunakan bahasa pemrograman Python, yang kini menjadi salah satu fondasi utama dalam pengembangan kecerdasan buatan modern.

Penulisan buku ini dilatarbelakangi oleh pesatnya perkembangan teknologi informasi, khususnya di bidang kecerdasan buatan dan machine learning. Deep learning telah menjadi salah satu topik yang sangat diminati baik di dunia akademis maupun industri, karena kemampuannya dalam memecahkan berbagai permasalahan kompleks, mulai dari pengenalan citra, pemrosesan bahasa alami, hingga pengembangan sistem rekomendasi. Python, sebagai bahasa pemrograman yang sederhana namun sangat powerful, telah menjadi pilihan utama para pengembang dan peneliti dalam membangun solusi berbasis deep learning.

Dimulai dari pengenalan konsep dasar TensorFlow, pembaca akan diajak menelusuri berbagai topik penting, seperti pembuatan tensor, grafik komputasional, fungsi aktivasi, hingga fungsi kerugian yang menjadi inti dari proses pembelajaran mesin. Selanjutnya, buku ini mengupas pemodelan mendalam dengan Keras, membangun dan meningkatkan model *neural network*, serta integrasi antara Keras dan TensorFlow untuk menghasilkan solusi yang efisien dan handal.

Tidak hanya membahas teori, buku ini juga mengajak pembaca untuk langsung terjun ke dunia nyata melalui studi kasus dan implementasi berbagai arsitektur jaringan saraf, seperti *Multilayer Perceptron* (MLP), *Convolutional Neural Network* (CNN), hingga *Recurrent Neural Network* (RNN) dan *Long Short-Term Memory* (LSTM). Selain itu, aplikasi AI yang sedang berkembang pesat seperti konversi ucapan ke teks, pengembangan chatbot, serta deteksi dan pengenalan wajah juga dibahas secara komprehensif untuk memperluas wawasan dan keterampilan pembaca.

Penulis berharap buku ini dapat memberikan referensi yang mudah dipahami dan aplikatif bagi mahasiswa, dosen, peneliti, serta praktisi yang ingin memperdalam pengetahuan dan keterampilan dalam bidang deep learning menggunakan Python.

Akhir kata, penulis mengucapkan terima kasih atas kepercayaan pembaca yang telah memilih buku ini sebagai panduan belajar. Kritik, saran, dan masukan sangat penulis harapkan demi perbaikan dan pengembangan buku ini di masa mendatang. Semoga buku ini dapat menjadi bekal berharga dalam perjalanan Anda menaklukkan dunia kecerdasan buatan.

Semarang, Mei 2025
Penulis

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

DAFTAR ISI

BAB 1
DASAR-DASAR TENSORFLOW

Bab ini membahas dasar-dasar TensorFlow, kerangka kerja pembelajaran mendalam. Pembelajaran mendalam bekerja dengan sangat baik dalam pengenalan pola, terutama dalam konteks gambar, suara, ucapan, bahasa, dan data deret waktu. Dengan bantuan pembelajaran mendalam, Anda dapat mengklasifikasikan, memprediksi, mengelompokkan, dan mengekstrak fitur. Untungnya, pada bulan November 2015, Google merilis TensorFlow, yang telah digunakan di sebagian besar produk Google seperti Google Search, deteksi spam, pengenalan ucapan, Google Assistant, Google Now, dan Google Photos. Menjelaskan komponen dasar TensorFlow merupakan tujuan dari bab ini.

TensorFlow memiliki kemampuan unik untuk melakukan komputasi subgraf parsial sehingga memungkinkan pelatihan terdistribusi dengan bantuan partisi jaringan saraf. Dengan kata lain, TensorFlow memungkinkan paralelisme model dan paralelisme data. TensorFlow menyediakan beberapa API.

API tingkat terendah—TensorFlow Core—memberi Anda kontrol pemrograman yang lengkap. Perhatikan poin-poin penting berikut mengenai TensorFlow:

- Grafiknya merupakan deskripsi komputasi.
- Grafiknya memiliki simpul yang merupakan operasi.
- Grafiknya menjalankan komputasi dalam konteks sesi tertentu.
- Grafik harus diluncurkan dalam sesi untuk komputasi apa pun.
- Sesi menempatkan operasi grafik ke perangkat seperti CPU dan GPU.
- Sesi menyediakan metode untuk menjalankan operasi grafik.

Untuk instalasi, silakan kunjungi https://www.tensorflow.org/install/
Saya akan membahas topik-topik berikut:

1.1 TENSOR

Sebelum Anda masuk ke pustaka TensorFlow, mari kita pahami dulu satuan data dasar di TensorFlow. Tensor adalah objek matematika dan generalisasi skalar, vektor, dan matriks. Tensor dapat direpresentasikan sebagai array multidimensi. Tensor dengan peringkat (orde)

nol tidak lain hanyalah skalar. Vektor/array adalah tensor dengan peringkat 1, sedangkan matriks adalah tensor dengan peringkat 2. Singkatnya, tensor dapat dianggap sebagai array n-dimensi. Berikut ini beberapa contoh tensor:

- 5: Ini adalah tensor peringkat 0; ini adalah skalar dengan bentuk [ ].
- [2. ,5. , 3.]: Ini adalah tensor peringkat 1; ini adalah vektor dengan bentuk [3].
- [[1. , 2. , 7.], [3. , 5. , 4.]]: Ini adalah tensor peringkat 2; ini adalah matriks dengan bentuk [2, 3].
- [[[1. , 2. , 3.]], [[7. , 8. , 9.]]]: Ini adalah tensor peringkat 3 dengan bentuk [2, 1, 3].

1.2 GRAFIK KOMPUTASIONAL DAN SESI

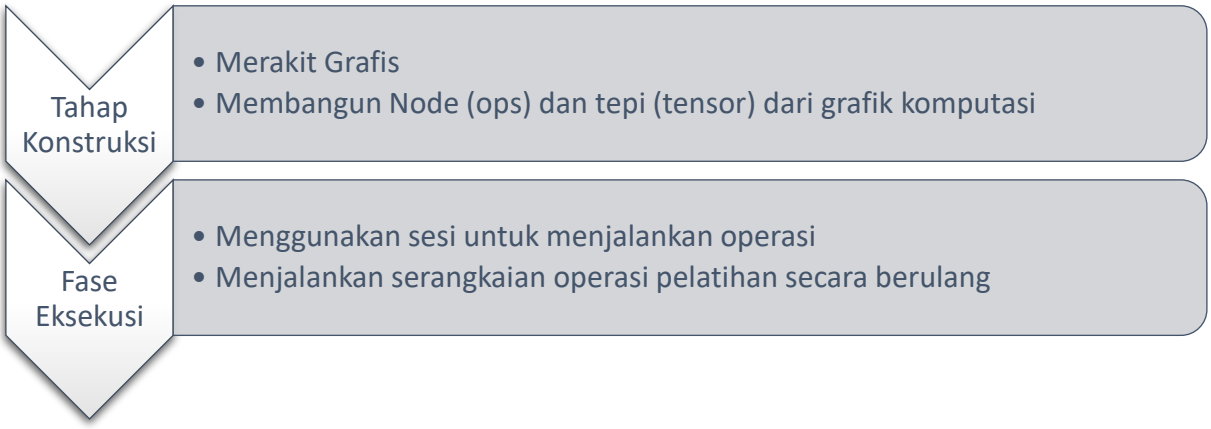
TensorFlow populer dengan program TensorFlow Core-nya yang memiliki dua tindakan utama.

- Membangun grafik komputasional dalam fase konstruksi
- Menjalankan grafik komputasional dalam fase eksekusi

Mari kita pahami cara kerja TensorFlow.

- Program-programnya biasanya terstruktur menjadi fase konstruksi dan fase eksekusi.
- Fase konstruksi menyusun grafik yang memiliki simpul (ops/operasi) dan tepi (tensor).
- Fase eksekusi menggunakan sesi untuk mengeksekusi ops (operasi) dalam grafik.
- Operasi yang paling sederhana adalah konstanta yang tidak mengambil input tetapi meneruskan output ke operasi lain yang melakukan komputasi.
- Contoh operasi adalah perkalian (atau penjumlahan atau pengurangan yang mengambil dua matriks sebagai input dan meneruskan matriks sebagai output).
- Pustaka TensorFlow memiliki grafik default yang konstruktor ops tambahkan node.

Jadi, struktur program TensorFlow memiliki dua fase, yang ditunjukkan di sini:



Grafik komputasional adalah serangkaian operasi TensorFlow yang disusun menjadi grafik simpul.

Mari kita lihat TensorFlow versus Numpy. Di Numpy, jika Anda berencana untuk mengalikan dua matriks, Anda membuat matriks dan mengalikannya. Namun di TensorFlow, Anda menyiapkan grafik (grafik default kecuali Anda membuat grafik lain).

Selanjutnya, Anda perlu membuat variabel, placeholder, dan nilai konstan lalu membuat sesi dan menginisialisasi variabel. Terakhir, Anda memasukkan data tersebut ke

placeholder untuk memanggil tindakan apa pun.

Untuk benar-benar mengevaluasi simpul, Anda harus menjalankan grafik komputasional dalam sesi. *Sesi* merangkum kontrol dan status runtime TensorFlow. Kode berikut membuat objek `Session`:

sess = tf.Session()

Kemudian memanggil metode run untuk menjalankan grafik komputasional yang cukup untuk mengevaluasi node1 dan node2.

Grafik komputasi mendefinisikan komputasi. Grafik tersebut tidak menghitung apa pun atau menyimpan nilai apa pun. Grafik tersebut dimaksudkan untuk mendefinisikan operasi yang disebutkan dalam kode. Grafik default dibuat. Jadi, Anda tidak perlu membuatnya kecuali Anda ingin membuat grafik untuk berbagai keperluan.

Sesi memungkinkan Anda untuk mengeksekusi grafik atau bagian dari grafik. Sesi mengalokasikan sumber daya (pada satu atau lebih CPU atau GPU) untuk eksekusi. Sesi menyimpan nilai aktual dari hasil dan variabel antara.

Nilai variabel yang dibuat di TensorFlow hanya berlaku dalam satu sesi. Jika Anda mencoba untuk mengkueri nilai setelahnya di sesi kedua, TensorFlow akan memunculkan kesalahan karena variabel tidak diinisialisasi di sana.

Untuk menjalankan operasi apa pun, Anda perlu membuat sesi untuk grafik tersebut. Sesi juga akan mengalokasikan memori untuk menyimpan nilai variabel saat ini
Berikut adalah kode untuk menunjukkannya:

```
import tensorflow as tf
sess = tf.Session()
```

```
# Creating a new graph(not default)
myGraph = tf.Graph()
with myGraph.as_default():
    variable = tf.Variable(30, name="navin")
    initialize = tf.global_variables_initializer()
```

```
with tf.Session(graph=myGraph) as sess:
    sess.run(initialize)
    print(sess.run(variable))
```

```
30
```

```
# Tensorboard can be used. It is optionalmy_
# Output graph can be seen on tensorboard
import os
merged = tf.summary.merge_all(key='summaries')
if not os.path.exists('tenosrboard_logs/'):
    os.makedirs('tenosrboard_logs/')

my_writer = tf.summary.FileWriter('/home/manaswi/tenosrboard_logs/', sess.graph)

def TB(cleanup=False):
    import webbrowser
    webbrowser.open('http://127.0.1.1:6006')
    !tensorboard --logdir='/home/manaswi/tenosrboard_logs'

    if cleanup:
        !rm -R tensorboard_logs/

TB(1)        # Launch graph on tensorborad on your browser
```

1.3 KONSTANTA, PLACEHOLDER, DAN VARIABEL

Program TensorFlow menggunakan struktur data tensor untuk merepresentasikan semua data—hanya tensor yang dilewatkan di antara operasi dalam grafik komputasi. Anda dapat menganggap tensor TensorFlow sebagai array atau daftar n-dimensi. Tensor memiliki tipe statis, peringkat, dan bentuk. Di sini grafik menghasilkan hasil yang konstan. Variabel mempertahankan status di seluruh eksekusi grafik.

Umumnya, Anda harus menangani banyak gambar dalam pembelajaran mendalam, jadi Anda harus menempatkan nilai piksel untuk setiap gambar dan terus mengulangi semua gambar.

Untuk melatih model, Anda harus dapat memodifikasi grafik untuk menyetel beberapa objek seperti bobot dan bias. Singkatnya, variabel memungkinkan Anda untuk menambahkan parameter yang dapat dilatih ke grafik. Mereka dibangun dengan tipe dan nilai awal. Mari buat konstanta di TensorFlow dan cetak.

```
import tensorflow as tf
x = tf.constant(12, dtype='float32')
sess = tf.Session()
print(sess.run(x))
```

```
12.0
```

Berikut penjelasan kode sebelumnya secara sederhana:

1. Impor modul `Tensorflow` dan beri nama tf.
2. Buat nilai konstan (x) dan tetapkan nilai numerik 12.
3. Buat sesi untuk menghitung nilai.
4. Jalankan variabel x saja dan cetak nilainya saat ini.

Dua langkah pertama termasuk dalam fase konstruksi, dan dua langkah terakhir termasuk dalam fase eksekusi. Saya akan membahas fase konstruksi dan eksekusi TensorFlow sekarang. Anda dapat menulis ulang kode sebelumnya dengan cara lain, seperti yang ditunjukkan di sini:

```
import tensorflow as tf
x = tf.constant(12, dtype='float32')
with tf.Session() as sess:
    print(sess.run(x))
```

```
12.0
```

Sekarang Anda akan mempelajari cara membuat variabel dan menginisialisasinya. Berikut kode yang melakukannya:

```
import tensorflow as tf
x = tf.constant(12, dtype='float32')
y = tf.Variable(x+11)
model = tf.global_variables_initializer()
with tf.Session() as sess:
    sess.run(model)
    print(sess.run(y))
```

```
23.0
```

Berikut penjelasan kode sebelumnya:

1. Impor modul tensorflow dan beri nama `tf`.
2. Buat nilai konstanta yang disebut x dan berikan nilai numerik 12.
3. Buat variabel yang disebut y dan definisikan sebagai persamaan $12 + 11$.
4. Inisialisasi variabel dengan `tf.global_variables_initializer()`.
5. Buat sesi untuk menghitung nilai.
6. Jalankan model yang dibuat pada langkah 4.
7. Jalankan hanya variabel y dan cetak nilainya saat ini.

Berikut beberapa kode lagi untuk Anda baca:

```
import tensorflow as tf
x = tf.constant([14, 23, 40, 30])
y = tf.Variable(x*2 + 100)
model = tf.global_variables_initializer()
with tf.Session() as sess:
    sess.run(model)
    print(sess.run(y))
```

```
[128 146 180 160]
```

1.4 PLACEHOLDER

Placeholder adalah variabel yang dapat Anda masukkan sesuatu di lain waktu. Placeholder dimaksudkan untuk menerima input eksternal. Placeholder dapat memiliki satu atau beberapa dimensi, yang dimaksudkan untuk menyimpan array n-dimensi.

```
import tensorflow as tf
x = tf.placeholder("float", None)
y = x*10 + 500
with tf.Session() as sess:
    placeX = sess.run(y, feed_dict={x: [0, 5, 15, 25]})
    print(placeX)
```

```
[500. 550. 650. 750.]
```

Berikut penjelasan kode sebelumnya:

1. Impor modul `tensorflow` dan beri nama `tf`.
2. Buat placeholder bernama x, sebutkan tipe float.
3. Buat tensor bernama y yang merupakan operasi perkalian x dengan 10 dan tambahkan 500 ke dalamnya. Perhatikan bahwa nilai awal untuk x tidak ditentukan.
4. Buat sesi untuk menghitung nilai.
5. Tentukan nilai x dalam `feed_dict` untuk menjalankan `y`.
6. Cetak nilainya.

Dalam contoh berikut, Anda membuat matriks 2x4 (array 2D) untuk menyimpan beberapa angka di dalamnya. Anda kemudian menggunakan operasi yang sama seperti sebelumnya untuk melakukan perkalian elemen demi elemen dengan 10 dan menambahkan 1 ke dalamnya. Dimensi pertama placeholder adalah **`None`**, yang berarti sejumlah baris diperbolehkan.

Anda juga dapat mempertimbangkan array 2D sebagai pengganti array 1D. Berikut kodenya:

```
import tensorflow as tf
x = tf.placeholder("float", [None, 4])
y = x*10 + 1
with tf.Session() as sess:
    dataX = [[12, 2, 0, -2],
             [14, 4, 1, 0]]
    placeX = sess.run(y, feed_dict={x: dataX})
    print(placeX)
```

```
[[121.  21.   1. -19.]
 [141.  41.  11.   1.]]
```

Ini adalah matriks 2x4. Jadi, jika Anda mengganti **None** dengan 2, Anda dapat melihat output yang sama.

```
import tensorflow as tf
x = tf.placeholder("float", [2, 4])
y = x*10 + 1
with tf.Session() as sess:
    dataX = [[12, 2, 0, -2],
             [14, 4, 1, 0]]
    placeX = sess.run(y, feed_dict={x: dataX})
    print(placeX)
```

```
[[121.  21.   1. -19.]
 [141.  41.  11.   1.]]
```

Namun jika Anda membuat placeholder berbentuk $[3, 4]$ (perhatikan bahwa Anda akan memasukkan matriks 2×4 nanti), akan terjadi kesalahan, seperti yang ditunjukkan di sini:

```
import tensorflow as tf
x = tf.placeholder("float", [3, 4])
y = x*10 + 1
with tf.Session() as sess:
    dataX = [[12, 2, 0, -2],
             [14, 4, 1, 0]]
    placeX = sess.run(y, feed_dict={x: dataX})
    print(placeX)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-10-c70a14b67e27> in <module>()
      5     dataX = [[12, 2, 0, -2],
      6              [14, 4, 1, 0]]
----> 7     placeX = sess.run(y, feed_dict={x: dataX})
      8     print(placeX)

~\Anaconda3\envs\tensorflow\lib\site-packages\tensorflow\python\client\session.py in run(self, fetches, feed_dict, options, run
_metadata)
    887     try:
    888       result = self._run(None, fetches, feed_dict, options_ptr,
--> 889                          run_metadata_ptr)
    890       if run_metadata:
    891         proto_data = tf_session.TF_GetBuffer(run_metadata_ptr)

~\Anaconda3\envs\tensorflow\lib\site-packages\tensorflow\python\client\session.py in _run(self, handle, fetches, feed_dict, opt
ions, run_metadata)
   1094                 'Cannot feed value of shape %r for Tensor %r, '
   1095                 'which has shape %r'
-> 1096                 % (np_val.shape, subfeed_t.name, str(subfeed_t.get_shape())))
   1097           if not self.graph.is_feedable(subfeed_t):
   1098             raise ValueError('Tensor %s may not be fed.' % subfeed_t)

ValueError: Cannot feed value of shape (2, 4) for Tensor 'Placeholder_5:0', which has shape '(3, 4)'
```

```
################## What happens in a linear model ##########
# Weight and Bias as Variables as they are to be tuned
W = tf.Variable([2], dtype=tf.float32)
b = tf.Variable([3], dtype=tf.float32)
# Training dataset that will be fed while training as Placeholders
x = tf.placeholder(tf.float32)
# Linear Model
y = W * x + b
```

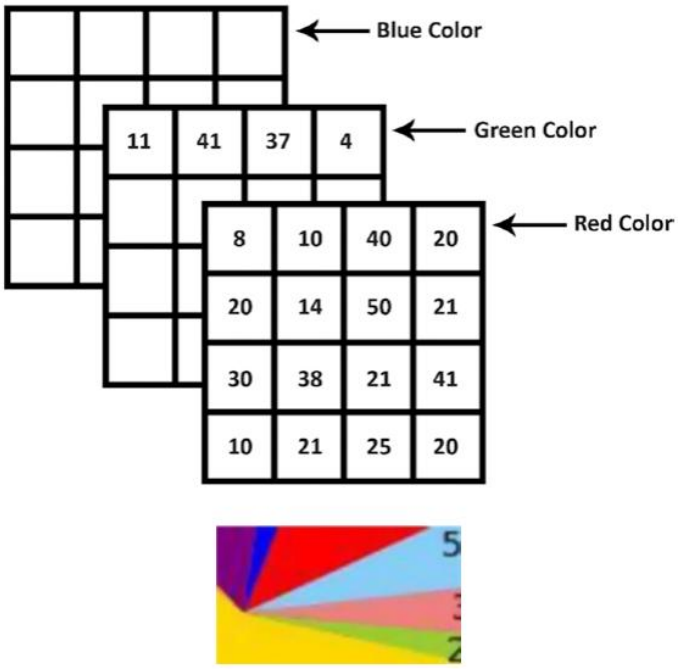
Konstanta diinisialisasi saat Anda memanggil `tf.constant`, dan nilainya tidak akan pernah berubah. Sebaliknya, variabel tidak diinisialisasi saat Anda memanggil `tf.Variable`. Untuk menginisialisasi semua variabel dalam program TensorFlow, Anda harus secara eksplisit memanggil operasi khusus sebagai berikut.

```
sess.run(tf.global_variables_initializer())
```

Penting untuk menyadari bahwa `init` adalah handle untuk subgraf TensorFlow yang menginisialisasi semua variabel global. Hingga Anda memanggil `sess.run`, variabel tidak diinisialisasi.

1.5 MEMBUAT TENSOR

Gambar adalah tensor orde ketiga yang dimensinya adalah tinggi, lebar, dan jumlah saluran (Merah, Biru, dan Hijau). Di sini Anda dapat melihat bagaimana gambar diubah menjadi tensor:



```
image = tf.image.decode_jpeg(tf.read_file("./Desktop/image.jpg"), channels=3)
sess = tf.InteractiveSession()
print(sess.run(tf.shape(image)))
```

```
[218 178   3]
```

```
print(sess.run(image[10:15,0:4,1]))
```

```
[[47 48 48 47]
 [45 45 45 44]
 [43 43 43 42]
 [41 42 42 41]
 [41 41 41 40]]
```

Anda dapat menghasilkan tensor berbagai jenis seperti tensor tetap, tensor acak, dan tensor sekuensial.

Tensor Tetap

Berikut adalah tensor tetap:

```
import tensorflow as tf
sess = tf.Session()
A = tf.zeros([2,3])
print(sess.run(A))
```

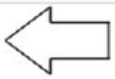
```
[[0. 0. 0.]
 [0. 0. 0.]]
```

```
B = tf.ones([4,3])
print(sess.run(B))
```

```
[[1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]]
```

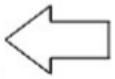
```
import tensorflow as tf
sess = tf.Session()
A = tf.zeros([2,3])
print(sess.run(A))
```

```
[[0. 0. 0.]
 [0. 0. 0.]]
```




```
B = tf.ones([4,3])
print(sess.run(B))
```

```
[[1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]]
```



`tf:.fill` membuat tensor berbentuk (2×3) yang memiliki nomor unik.

```
C = tf.fill([2,3], 13)
print(sess.run(C))
```

```
[[13 13 13]
 [13 13 13]]
```

`tf.diag` membuat matriks diagonal yang memiliki elemen diagonal yang ditentukan.

```
D = tf.diag([4,-3,2])
print(sess.run(D))
```

```
[[ 4  0  0]
 [ 0 -3  0]
 [ 0  0  2]]
```

`tf.constant` membuat tensor konstan.

```
E = tf.constant([5,2,4,2])
print(sess.run(E))
```

```
[5 2 4 2]
```

Tensor Urutan

`tf.range` menciptakan urutan angka yang dimulai dari nilai yang ditentukan dan memiliki kenaikan yang ditentukan.

```
G = tf.range(start=6, limit=45, delta=3)
print(sess.run(G))
```

```
[ 6  9 12 15 18 21 24 27 30 33 36 39 42]
```

`tf.linspace` menciptakan serangkaian nilai yang diberi jarak secara merata.

```
H = tf.linspace(10.0, 92.0, 5)
print(sess.run(H))
```

```
[10.  30.5 51.  71.5 92. ]
```

Tensor Acak

`tf.random_uniform` menghasilkan nilai acak dari distribusi seragam dalam suatu rentang.

```
R1 = tf.random_uniform([2,3], minval=0, maxval=4)
print(sess.run(R1))
```

```
[[0.74450636 1.9570832  3.1126966 ]
 [2.359518   2.101438   2.65689   ]]
```

`tf.random_normal` menghasilkan nilai acak dari distribusi normal yang memiliki rata-rata dan deviasi standar yang ditentukan.

```
R2 = tf.random_normal([2,3], mean=5, stddev=4)
print(sess.run(R2))
```

```
[[-1.8996243  3.2514744  5.9602127]
 [ 8.307009   4.84437    6.8460846]]
```

```
print(sess.run(tf.diag([3,-2,4])))
```

```
[[ 3  0  0]
 [ 0 -2  0]
 [ 0  0  4]]
```

```
R3 = tf.random_shuffle(tf.diag([3,-2,4]))
print(sess.run(R3))
```

```
[[ 3  0  0]
 [ 0  0  4]
 [ 0 -2  0]]
```

```
R4 = tf.random_crop(tf.diag([3,-2,4]), [3,2])
print(sess.run(R4))
```

```
[[ 3  0]
 [ 0 -2]
 [ 0  0]]
```

Bisakah Anda menebak hasilnya?

```
print(sess.run(tf.zeros([2,4])))
```

```
print(sess.run(tf.diag([3,1,5,-2])))
```

```
print(sess.run(tf.range(start=4, limit=16, delta=2)))
```

Jika Anda tidak dapat menemukan hasilnya, silakan baca kembali bagian sebelumnya yang membahas tentang pembuatan tensor.
Berikut ini hasilnya:

```
print(sess.run(tf.zeros([2,4])))
```

```
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]]
```

```
print(sess.run(tf.diag([3,1,5,-2])))
```

```
[[ 3  0  0  0]
 [ 0  1  0  0]
 [ 0  0  5  0]
 [ 0  0  0 -2]]
```

```
print(sess.run(tf.range(start=4, limit=16, delta=2)))
```

```
[ 4  6  8 10 12 14]
```

Bekerja pada Matriks

Setelah Anda merasa nyaman membuat tensor, Anda dapat menikmati bekerja pada matriks (tensor 2D).

```
import tensorflow as tf
import numpy as np
sess = tf.Session()
A = tf.random_uniform([3,2])
B = tf.fill([2,4], 3.5)
C = tf.random_normal([3,4])
```

```
print(sess.run(A))
```

```
[[0.31633115 0.71407604]
 [0.18088198 0.36230946]
 [0.34481096 0.6156665 ]]
```

```
print(sess.run(B))
```

```
[[3.5 3.5 3.5 3.5]
 [3.5 3.5 3.5 3.5]]
```

```
print(sess.run(tf.matmul(A,B)))# Multiplication of Matrices
```

```
[[0.9453191 0.9453191 0.9453191 0.9453191]
 [4.4488316 4.4488316 4.4488316 4.4488316]
 [3.308284  3.308284  3.308284  3.308284 ]]
```

```
print(sess.run(tf.matmul(A,B) + C))# Multiplication & addition
```

```
[[4.6027136 4.5958595 6.9527874 4.413632 ]
 [3.3000264 4.4702578 5.0858393 5.168917 ]
 [3.1176403 4.626109  4.1446424 3.7285264]]
```

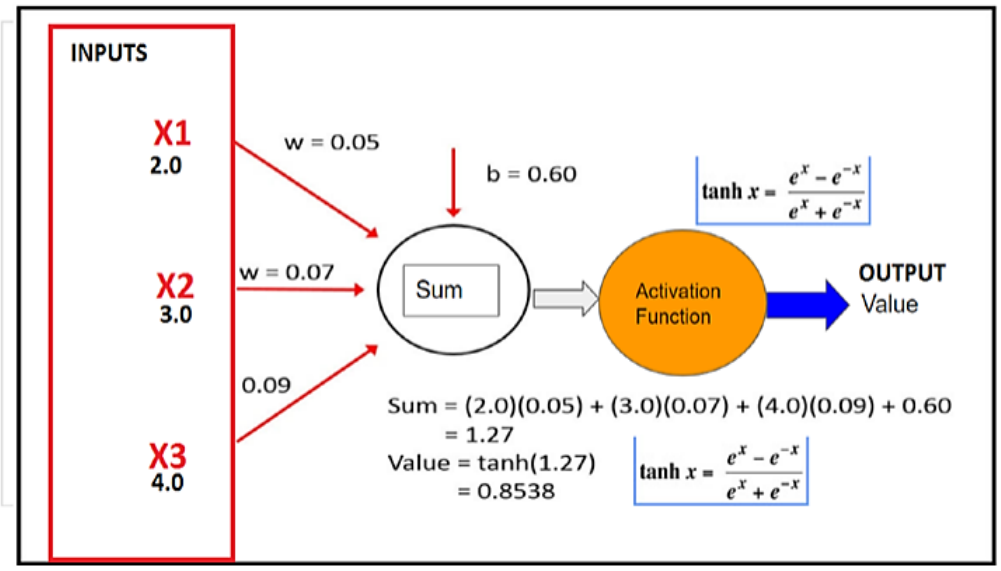
1.6 FUNGSI AKTIVASI

Ide fungsi aktivasi berasal dari analisis tentang cara kerja neuron di otak manusia (lihat Gambar 1-1). Neuron menjadi aktif melampaui ambang tertentu, yang lebih dikenal sebagai *potensial aktivasi.* Dalam kebanyakan kasus, neuron juga mencoba menempatkan output ke dalam rentang kecil.

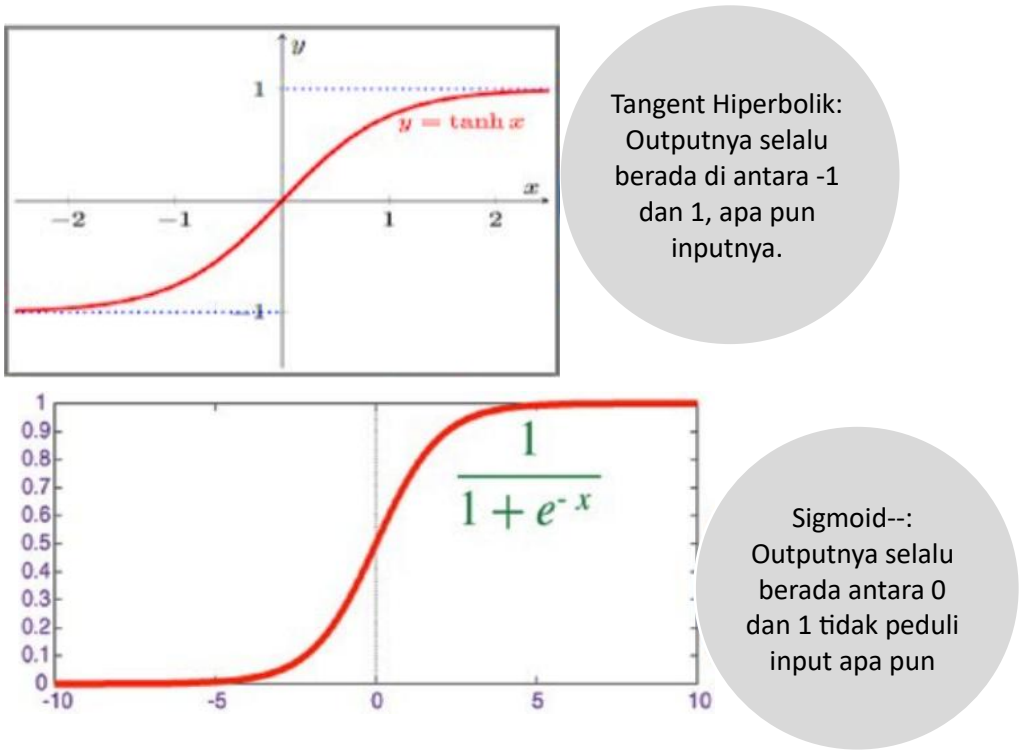
Sigmoid, tangen hiperbolik (tanh), ReLU, dan ELU adalah fungsi aktivasi yang paling populer. Mari kita lihat fungsi aktivasi yang populer.

Tangen Hiperbolik dan Sigmoid

Gambar 1-2 menunjukkan fungsi aktivasi tangen hiperbolik dan sigmoid.



Gambar 1-1. Fungsi aktivasi



Gambar 1-2. Dua fungsi aktivasi yang populer

Berikut adalah kode demo:

```
E = tf.nn.tanh([10,2,1,0.5,0,-0.5,-1.,-2.,-10.])
print(sess.run(E))
```

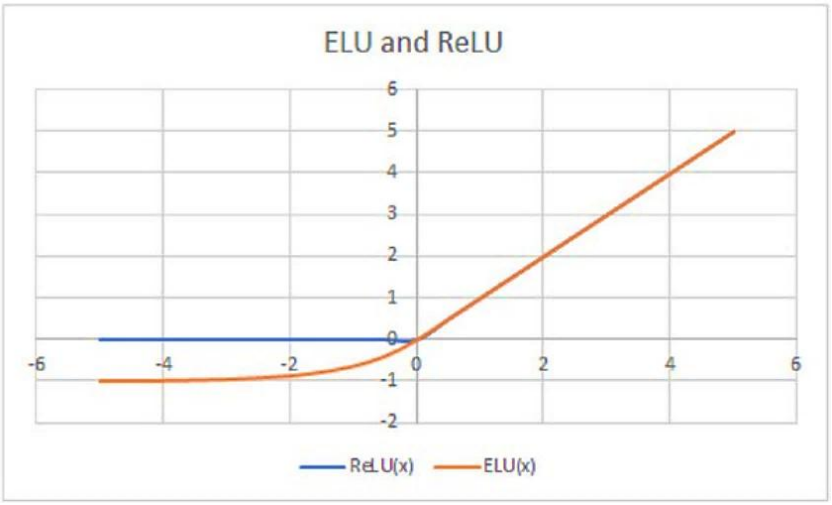
```
[ 1.          0.9640276   0.7615942   0.46211717  0.         -0.46211717
 -0.7615942  -0.9640276  -1.        ]
```

```
J = tf.nn.sigmoid([10,2,1,0.5,0,-0.5,-1.,-2.,-10.])
print(sess.run(J))
```

```
[9.9995458e-01 8.8079703e-01 7.3105860e-01 6.2245935e-01 5.0000000e-01
 3.7754068e-01 2.6894143e-01 1.1920292e-01 4.5397872e-05]
```

ReLU dan ELU

Gambar 1-3 menunjukkan fungsi ReLU dan ELU.



Gambar 1-3. Fungsi ReLU dan ELU

Berikut ini kode untuk menghasilkan fungsi-fungsi tersebut:

```
A = tf.nn.relu([-2,1,-3,13])
print(sess.run(A))
```

```
[ 0  1  0 13]
```

ReLU6

ReLU6 mirip dengan ReLU kecuali outputnya tidak boleh lebih dari enam.

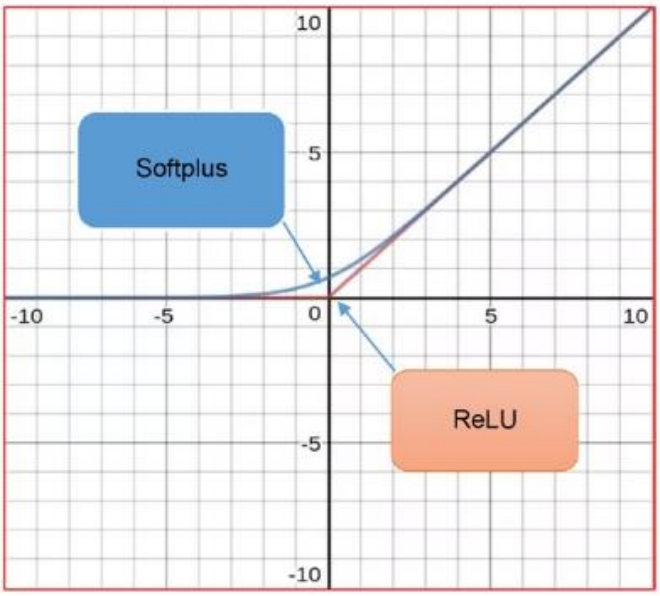
```
B = tf.nn.relu6([-2,1,-3,13])
print(sess.run(B))
```

```
[0 1 0 6]
```

```
C = tf.nn.relu([[-2,1,-3],[10,-16,-5]])
print(sess.run(C))
```

```
[[ 0  1  0]
 [10  0  0]]
```

Perhatikan bahwa tanh adalah fungsi sigmoid logistik yang diskalakan ulang.

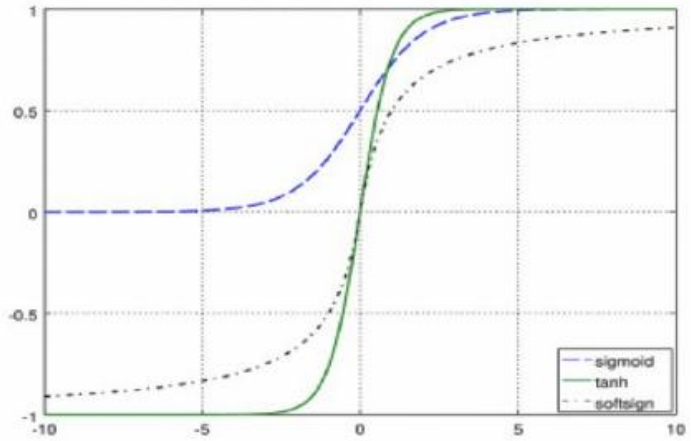



```
K = tf.nn.relu([10,2,1,0.5,0,-0.5,-1.,-2.,-10.])
print(sess.run(K))
```

```
[10.   2.   1.   0.5  0.   0.   0.   0.   0. ]
```

```
M = tf.nn.softplus([10,2,1,0.5,0,-0.5,-1.,-2.,-10.])
print(sess.run(M))
```

```
[1.0000046e+01 2.1269281e+00 1.3132616e+00 9.7407699e-01 6.9314718e-01
 4.7407699e-01 3.1326163e-01 1.2692805e-01 4.5417706e-05]
```




```
H = tf.nn.elu([10,2,1,0.5,0,-0.5,-1.,-2.,-10.])
print(sess.run(H))
```

```
[10.          2.          1.          0.5         0.         -0.39346933
 -0.63212055 -0.86466473 -0.9999546 ]
```

```
I = tf.nn.relu6([10,2,1,0.5,0,-0.5,-1.,-2.,-10.])
print(sess.run(I))
```

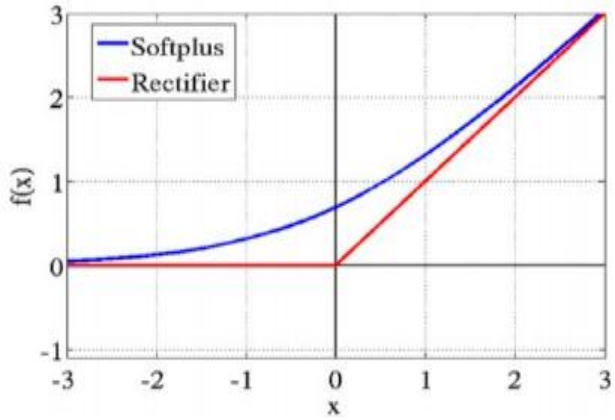
```
[6.  2.  1.  0.5 0.  0.  0.  0.  0. ]
```

```
G = tf.nn.softsign([10,2,1,0.5,0,-0.5,-1.,-2.,-10.])
print(sess.run(G))
```

```
[ 0.90909094  0.6666667   0.5         0.33333334  0.         -0.33333334
 -0.5        -0.6666667  -0.90909094]
```

```
F = tf.nn.softplus([10,2,1,0.5,0,-0.5,-1.,-2.,-10.])
print(sess.run(F))
```

```
[1.0000046e+01 2.1269281e+00 1.3132616e+00 9.7407699e-01 6.9314718e-01
 4.7407699e-01 3.1326163e-01 1.2692805e-01 4.5417706e-05]
```



1.7 FUNGSI KERUGIAN (LOSS FUNCTIONS)

Fungsi kerugian (fungsi biaya) harus diminimalkan untuk mendapatkan nilai terbaik bagi setiap parameter model. Misalnya, Anda perlu mendapatkan nilai terbaik dari bobot (kemiringan) dan bias (intersep-y) untuk menjelaskan target (y) dalam bentuk prediktor (X). Metodenya adalah untuk mencapai nilai terbaik dari kemiringan, dan intersep-y adalah untuk meminimalkan fungsi biaya/fungsi kerugian/jumlah kuadrat. Untuk model apa pun, ada banyak parameter, dan struktur model dalam prediksi atau klasifikasi dinyatakan dalam bentuk nilai parameter.

Anda perlu mengevaluasi model Anda, dan untuk itu Anda perlu menentukan fungsi biaya (fungsi kerugian). Minimalisasi fungsi kerugian dapat menjadi kekuatan pendorong untuk menemukan nilai optimum setiap parameter. Untuk regresi/prediksi numerik, L1 atau L2 dapat menjadi fungsi kerugian yang berguna. Untuk klasifikasi, entropi silang dapat menjadi fungsi kerugian yang berguna. Softmax atau entropi silang sigmoid dapat menjadi fungsi kerugian yang cukup populer.

Contoh Fungsi Kerugian

Berikut adalah kode untuk menunjukkannya:

```
import tensorflow as tf
import numpy as np
sess = tf.Session()
```

```
#Assuming prediction model
pred=np.asarray([0.2,0.3,0.5,10.0,12.0,13.0,3.5,7.4,3.9,2.3])
#convert ndarray into tensor
x_val=tf.convert_to_tensor(pred)
#Assuming actual values
actual=np.asarray([0.1,0.4,0.6,9.0,11.0,12.0,3.4,7.1,3.8,2.0])
```

```
#L2 loss:L1=(pred-actual)^2
l2=tf.square(pred-actual)
l2_out=sess.run(tf.round(l2))
print(l2_out)
```

```
[ 0.  0.  0.  1.  1.  1.  0.  0.  0.  0.]
```

```
#L2 loss:L1=abs(pred-actual)
l1=tf.abs(pred-actual)
l1_out=sess.run(l1)
print(l1_out)
```

```
[ 0.1  0.1  0.1  1.   1.   1.   0.1  0.3  0.1  0.3]
```

```
#cross entropy loss
softmax_xentropy_variable=tf.nn.sigmoid_cross_entropy_with_logits(logits=l1_out,labels=l2_out)
print(sess.run(softmax_xentropy_variable))
```

```
[ 0.74439666  0.74439666  0.74439666  0.31326169  0.31326169  0.31326169
  0.74439666  0.85435524  0.74439666  0.85435524]
```

Fungsi Kerugian Umum

Berikut ini adalah daftar fungsi kerugian yang paling umum:

```
tf.contrib.losses.absolute_difference
tf.contrib.losses.add_loss
tf.contrib.losses.hinge_loss
tf.contrib.losses.compute_weighted_loss
tf.contrib.losses.cosine_distance
tf.contrib.losses.get_losses
tf.contrib.losses.get_regularization_losses
tf.contrib.losses.get_total_loss
tf.contrib.losses.log_loss
tf.contrib.losses.mean_pairwise_squared_error
tf.contrib.losses.mean_squared_error
tf.contrib.losses.sigmoid_cross_entropy
tf.contrib.losses.softmax_cross_entropy
tf.contrib.losses.sparse_softmax_cross_entropy
tf.contrib.losses.log(predictions,labels,weight=2.0)
```

Pengoptimal

Sekarang Anda harus yakin bahwa Anda perlu menggunakan fungsi kerugian untuk mendapatkan nilai terbaik dari setiap parameter model. Bagaimana Anda bisa mendapatkan nilai terbaik?

Awalnya Anda mengasumsikan nilai awal bobot dan bias untuk model (regresi linier, dll.). Sekarang Anda perlu menemukan cara untuk mencapai nilai terbaik dari parameter. Pengoptimal adalah cara untuk mencapai nilai terbaik dari parameter. Dalam setiap iterasi, nilai berubah ke arah yang disarankan oleh pengoptimal. Misalkan Anda memiliki 16 nilai bobot (`w1,w2,w3, …, w16`) dan 4 bias (`b1,b2,b3,b4`). Awalnya Anda dapat mengasumsikan

setiap bobot dan bias menjadi nol (atau satu atau angka apa pun). Pengoptimal menyarankan apakah `w1` (dan parameter lainnya) harus meningkat atau menurun pada iterasi berikutnya sambil tetap mengingat tujuan minimalisasi. Setelah banyak iterasi, `w1` (dan parameter lainnya) akan stabil ke nilai terbaik (atau nilai-nilai) parameter. Dengan kata lain, TensorFlow, dan setiap kerangka kerja pembelajaran mendalam lainnya, menyediakan pengoptimal yang mengubah setiap parameter secara perlahan untuk meminimalkan fungsi kerugian. Tujuan pengoptimal adalah untuk memberikan arah pada bobot dan bias untuk perubahan pada iterasi berikutnya.

Misalkan Anda memiliki 64 bobot dan 16 bias; Anda mencoba mengubah nilai bobot dan bias pada setiap iterasi (selama backpropagation) sehingga Anda mendapatkan nilai bobot dan bias yang benar setelah banyak iterasi sambil mencoba meminimalkan fungsi kerugian.

Memilih pengoptimal terbaik bagi model untuk konvergen dengan cepat dan mempelajari bobot dan bias dengan benar adalah tugas yang sulit.

Teknik adaptif (adadelta, adagrad, dll.) adalah pengoptimal yang baik untuk konvergen lebih cepat untuk jaringan saraf yang kompleks. Adam seharusnya menjadi pengoptimal terbaik untuk sebagian besar kasus. Ia juga mengungguli teknik adaptif lainnya (adadelta, adagrad, dll.), tetapi secara komputasi mahal. Untuk set data yang jarang, metode seperti SGD, NAG, dan momentum bukanlah pilihan terbaik; metode kecepatan pembelajaran adaptif adalah pilihan terbaik. *Manfaat tambahannya* adalah Anda tidak perlu menyesuaikan kecepatan pembelajaran tetapi kemungkinan besar dapat memperoleh hasil terbaik dengan nilai default.

Contoh Fungsi Kerugian

Berikut adalah kode untuk mendemonstrasikannya:

```
# Importing libraries
import tensorflow as tf
```

```
# Assign the value into variable
x = tf.Variable(3, name='x', dtype=tf.float32)
log_x = tf.log(x)
log_x_squared = tf.square(log_x)
```

```
# Apply GradientDescentOptimizer
optimizer = tf.train.GradientDescentOptimizer(0.7)
train = optimizer.minimize(log_x_squared)
```

```
# Initialize Variables
init = tf.global_variables_initializer()
```

```
# Finally running computation
with tf.Session() as session:
    session.run(init)
    print("starting at", "x:", session.run(x), "log(x)^2:", session.run(log_x_squared))
    for step in range(10):
        session.run(train)
        print("step", step, "x:", session.run(x), "log(x)^2:", session.run(log_x_squared))
```

```
starting at x: 3.0 log(x)^2: 1.20695
step 0 x: 2.48731 log(x)^2: 0.830292
step 1 x: 1.97444 log(x)^2: 0.462786
step 2 x: 1.49207 log(x)^2: 0.160134
step 3 x: 1.1166 log(x)^2: 0.0121637
step 4 x: 0.97832 log(x)^2: 0.00048043
step 5 x: 1.00969 log(x)^2: 9.29177e-05
step 6 x: 0.99632 log(x)^2: 1.35901e-05
step 7 x: 1.0015 log(x)^2: 2.24809e-06
step 8 x: 0.999405 log(x)^2: 3.54772e-07
step 9 x: 1.00024 log(x)^2: 5.70574e-08
```

Pengoptimal Umum

Berikut ini adalah daftar pengoptimal umum:

```
tf.train.Optimizer
tf.train.GradientDescentOptimizer
tf.train.AdadeltaOptimizer
tf.train.AdagradOptimizer
tf.train.AdagradDAOptimizer
tf.train.MomentumOptimizer
tf.train.AdamOptimizer
tf.train.FtrlOptimizer
tf.train.ProximalGradientDescentOptimizer
tf.train.ProximalAdagradOptimizer
tf.train.RMSPropOptimizer
```

Metrik

Setelah mempelajari beberapa cara untuk membangun model, sekarang saatnya untuk mengevaluasi model tersebut. Jadi, Anda perlu mengevaluasi regresor atau pengklasifikasi. Ada banyak metrik evaluasi, di antaranya akurasi klasifikasi, kerugian logaritmik, dan area di bawah kurva ROC adalah yang paling populer.

Akurasi klasifikasi adalah rasio jumlah prediksi yang benar terhadap jumlah semua prediksi. Ketika pengamatan untuk setiap kelas tidak terlalu miring, akurasi dapat dianggap sebagai metrik yang baik.

```
tf.contrib.metrics.accuracy(actual_labels, predictions)
```

Ada juga metrik evaluasi lainnya.

Contoh Metrik

Bagian ini menunjukkan kode yang akan didemonstrasikan. Di sini Anda membuat nilai aktual (menyebutnya x) dan nilai prediksi (menyebutnya y). Kemudian Anda memeriksa keakuratannya. Akurasi menunjukkan rasio jumlah kali nilai aktual sama dengan nilai prediksi

dan jumlah total kejadian.

Metrik Umum

Berikut ini adalah daftar metrik umum:

```
tf.contrib.metrics.streaming_root_mean_squared_error
tf.contrib.metrics.streaming_covariance
tf.contrib.metrics.streaming_pearson_correlation
tf.contrib.metrics.streaming_mean_cosine_distance
tf.contrib.metrics.streaming_percentage_less
tf.contrib.metrics.streaming_sensitivity_at_specificity
tf.contrib.metrics.streaming_sparse_average_precision_at_k
tf.contrib.metrics.streaming_sparse_precision_at_k
tf.contrib.metrics.streaming_sparse_precision_at_top_k
tf.contrib.metrics.streaming_specificity_at_sensitivity
tf.contrib.metrics.streaming_concat
tf.contrib.metrics.streaming_false_negatives
tf.contrib.metrics.streaming_false_negatives_at_thresholds
```

```
# Importing libraries
import numpy as np
import tensorflow as tf
```

```
# Placeholders declaration
x=tf. placeholder(tf.int32, [5])
y=tf. placeholder(tf.int32, [5])
```

```
# Metrices declaration
acc, acc_op=tf.metrics.accuracy(labels=x, predictions=y)
```

```
# Session initialization
sess=tf.InteractiveSession()
sess.run(tf.global_variables_initializer())
sess.run(tf.local_variables_initializer())
```

```
# Value assign
val= sess.run([acc,acc_op], feed_dict={x: [1,1,0,1,0], y: [0,1,0,0,1]})
```

```
# Print Accuracy
val_acc=sess.run(acc)
print(val_acc)
# You can see only 2nd and 3rd positions value are same
```

```
0.4
```

BAB 2
MEMAHAMI DAN BEKERJA DENGAN KERAS

Keras adalah pustaka Python tingkat tinggi yang ringkas dan mudah dipelajari untuk pembelajaran mendalam yang dapat berjalan di atas TensorFlow (atau Theano atau CNTK). Pustaka ini memungkinkan pengembang untuk fokus pada konsep utama pembelajaran mendalam, seperti membuat lapisan untuk jaringan saraf, sambil memperhatikan detail tensor, bentuknya, dan detail matematisnya. TensorFlow (atau Theano atau CNTK) harus menjadi back-end untuk Keras. Anda dapat menggunakan Keras untuk aplikasi pembelajaran mendalam tanpa berinteraksi dengan TensorFlow (atau Theano atau CNTK) yang relatif rumit.

Ada dua jenis kerangka kerja utama: *API sekuensial* dan *API fungsional*. API sekuensial didasarkan pada gagasan tentang urutan lapisan; ini adalah penggunaan Keras yang paling umum dan bagian termudah dari Keras. Model sekuensial dapat dianggap sebagai tumpukan lapisan linier.

Singkatnya, Anda membuat model sekuensial tempat Anda dapat dengan mudah menambahkan lapisan, dan setiap lapisan dapat memiliki konvolusi, pengumpulan maksimum, aktivasi, pelepasan, dan normalisasi batch. Mari kita bahas langkah-langkah utama untuk mengembangkan model pembelajaran mendalam di Keras.

2.1 PERMODELAN PEMBELAJARAN MENDALAM

Empat bagian inti dari model pembelajaran mendalam di Keras adalah sebagai berikut:

1. Tentukan model. Di sini Anda membuat model sekuensial dan menambahkan lapisan. Setiap lapisan dapat berisi satu atau lebih konvolusi, penggabungan, normalisasi batch, dan fungsi aktivasi.
2. Kompilasi model. Di sini Anda menerapkan fungsi kerugian dan pengoptimal sebelum memanggil fungsi `compile()` pada model.
3. Sesuaikan model dengan data pelatihan. Di sini Anda melatih model pada data uji dengan memanggil fungsi `fit()` pada model.
4. Buat prediksi. Di sini Anda menggunakan model untuk menghasilkan prediksi pada data baru dengan memanggil fungsi seperti `evaluate()` dan `predict()`.

Ada delapan langkah untuk proses pembelajaran mendalam di Keras:

1. Muat data.
2. Praproses data.
3. Tentukan model.
4. Menyusun model.
5. Sesuaikan model.
6. Evaluasi model.
7. Buat prediksi.
8. Simpan modelnya.

Memuat Data

Berikut ini cara memuat data:

```
# Importing modules
import numpy as np
import os
from keras.datasets import cifar10
from keras.models import Sequential
from keras.layers.core import Dense, Dropout, Activation
from keras.optimizers import adam
from keras.utils import np_utils
```

```
#Load Data
np.random.seed(100) # for reproducibility
(X_train, y_train), (X_test, y_test) = cifar10.load_data()

#cifar-10 has images of airplane, automobile, bird, cat,
# deer, dog, frog, horse, ship and truck ( 10 unique labels)
# For each image. width = 32, height =32, Number of channels(RGB) = 3
```

Praproses Data

Berikut ini cara Anda melakukan praproses data:

```
#Preprocess the data
#Flatten the data, MLP doesn't use the 2D structure of the data. 3072 = 3*32*32
X_train = X_train.reshape(50000, 3072) # 50,000 images for training
X_test = X_test.reshape(10000, 3072) # 10,000 images for test

# Gaussian Normalization( Z- score)
X_train = (X_train- np.mean(X_train))/np.std(X_train)
X_test = (X_test- np.mean(X_test))/np.std(X_test)
```

```
# Convert class vectors to binary class matrices (ie one-hot vectors)
labels = 10 #10 unique labels(0-9)
Y_train = np_utils.to_categorical(y_train, labels)
Y_test = np_utils.to_categorical(y_test, labels)
```

Tentukan Model

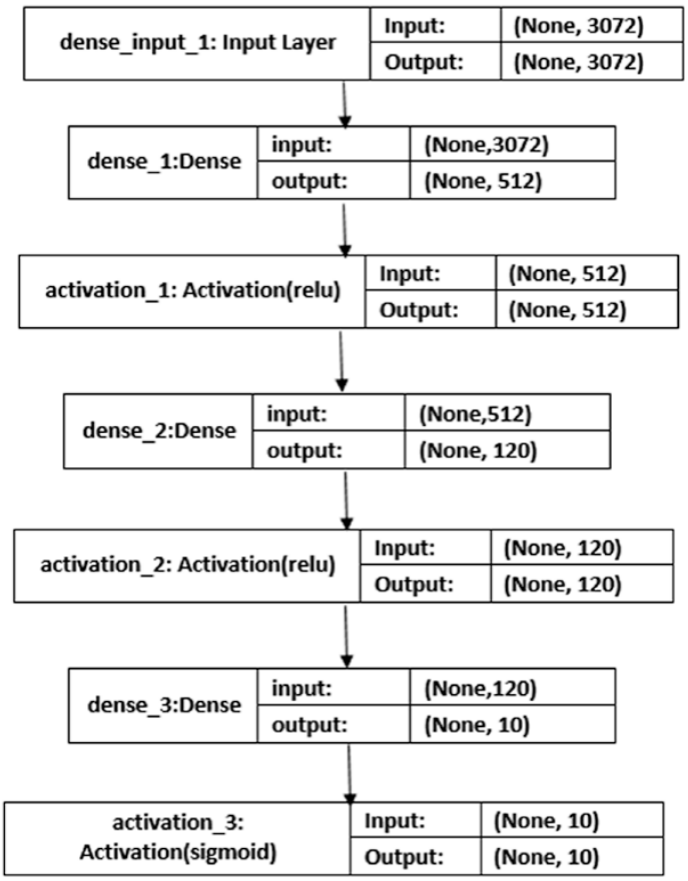
Model sekuensial di Keras didefinisikan sebagai urutan lapisan. Anda membuat model sekuensial lalu menambahkan lapisan. Anda perlu memastikan lapisan input memiliki jumlah input yang tepat. Asumsikan Anda memiliki 3.072 variabel input; maka Anda perlu membuat lapisan tersembunyi pertama dengan 512 node/neuron. Di lapisan tersembunyi kedua, Anda memiliki 120 node/neuron. Terakhir, Anda memiliki sepuluh node di lapisan output. Misalnya, gambar dipetakan ke sepuluh node yang menunjukkan probabilitas label1 (pesawat terbang), label2 (mobil), label3 (kucing), …, label10 (truk). Node dengan probabilitas tertinggi adalah kelas/label yang diprediksi.

```
#Define the model achitecture
model = Sequential()
model.add(Dense(512, input_shape=(3072,))) # 3*32*32 = 3072
model.add(Activation('relu'))
model.add(Dropout(0.4)) # Regularization
model.add(Dense(120))
model.add(Activation('relu'))
model.add(Dropout(0.2))# Regularization
model.add(Dense(labels)) #Last layer with 10 outputs, each output per class
model.add(Activation('sigmoid'))
```

Satu gambar memiliki tiga saluran (RGB), dan di setiap saluran, gambar memiliki $32 \times 32 = 1024$ piksel. Jadi, setiap gambar memiliki $3 \times 1024 = 3072$ piksel (fitur/X/input).

Dengan bantuan 3.072 fitur, Anda perlu memprediksi probabilitas label1 (Digit 0), label2 (Digit 1), dan seterusnya. Ini berarti model memprediksi sepuluh keluaran (Digit 0–9) di mana setiap keluaran mewakili probabilitas label yang sesuai. Fungsi aktivasi terakhir (sigmoid, seperti yang ditunjukkan sebelumnya) memberikan 0 untuk sembilan keluaran dan 1 hanya untuk satu keluaran. Label tersebut adalah kelas yang diprediksi untuk gambar tersebut (Gambar 2-1).

Misalnya, 3.072 fitur ➤ 512 node ➤ 120 node ➤ 10 node.



Gambar 2-1. Menentukan model

Pertanyaan berikutnya adalah, bagaimana Anda mengetahui jumlah lapisan yang akan

digunakan dan jenisnya? Tidak ada yang punya jawaban pasti. Yang terbaik untuk metrik evaluasi adalah Anda memutuskan jumlah lapisan yang optimal dan parameter serta langkah-langkah di setiap lapisan. Pendekatan heuristik juga digunakan. Struktur jaringan terbaik ditemukan melalui proses percobaan coba-coba. Umumnya, Anda memerlukan jaringan yang cukup besar untuk menangkap struktur masalah.

Dalam contoh ini, Anda akan menggunakan struktur jaringan yang terhubung penuh dengan tiga lapisan. Kelas padat mendefinisikan lapisan yang terhubung penuh.

Dalam kasus ini, Anda menginisialisasi bobot jaringan ke angka acak kecil yang dihasilkan dari distribusi seragam (seragam) dalam kasus ini antara 0 dan 0,05 karena itu adalah inisialisasi bobot seragam default di Keras. Alternatif tradisional lainnya adalah normal untuk angka acak kecil yang dihasilkan dari distribusi Gaussian. Anda menggunakan atau snap ke klasifikasi keras dari salah satu kelas dengan ambang batas default 0,5. Anda dapat menyatukan semuanya dengan menambahkan setiap lapisan.

Menyusun Model

Setelah mendefinisikan model dalam bentuk lapisan, Anda perlu mendeklarasikan fungsi kerugian, pengoptimal, dan metrik evaluasi. Ketika model diusulkan, nilai bobot dan bias awal diasumsikan sebagai 0 atau 1, angka acak yang didistribusikan secara normal, atau angka lain yang mudah digunakan. Namun, nilai awal bukanlah nilai terbaik untuk model tersebut. Ini berarti nilai awal bobot dan bias tidak dapat menjelaskan target/label dalam bentuk prediktor (X). Jadi, Anda ingin mendapatkan nilai optimal untuk model tersebut. Perjalanan dari nilai awal ke nilai optimal memerlukan motivasi, yang akan meminimalkan fungsi biaya/fungsi kerugian. Perjalanan tersebut memerlukan jalur (perubahan pada setiap iterasi), yang disarankan oleh pengoptimal. Perjalanan tersebut juga memerlukan pengukuran evaluasi, atau metrik evaluasi.

```
# Compile the model
# Use adam as an optimizer
adam = adam(0.01)
# the cross entropy between the true label and the output(softmax) of the model
model.compile(loss='categorical_crossentropy', optimizer=adam, metrics=["accuracy"])
```

Fungsi kerugian yang populer adalah binary cross entropy, categorical cross entropy, mean_squared_logarithmic_error, dan hinge loss. Pengoptimal yang populer adalah stochastic gradient descent (SGD), RMSProp, adam, adagrad, dan adadelta. Metrik evaluasi yang populer adalah accuracy, recall, dan F1 score.

Singkatnya, langkah ini ditujukan untuk menyetel bobot dan bias berdasarkan fungsi kerugian melalui iterasi berdasarkan pengoptimal yang dievaluasi oleh metrik seperti akurasi.

Sesuaikan Model

Setelah mendefinisikan dan mengompilasi model, Anda perlu membuat prediksi dengan mengeksekusi model pada beberapa data. Di sini Anda perlu menentukan epoch; ini adalah jumlah iterasi untuk proses pelatihan agar berjalan melalui set data dan ukuran batch, yang merupakan jumlah instans yang dievaluasi sebelum pembaruan bobot.

Untuk masalah ini, program akan berjalan selama sejumlah kecil epoch (10), dan di setiap epoch, program akan menyelesaikan 50(=50.000/1.000) iterasi di mana ukuran batch adalah 1.000 dan set data pelatihan memiliki 50.000 instans/gambar. Sekali lagi, tidak ada aturan keras untuk memilih ukuran batch. Namun, ukurannya tidak boleh terlalu kecil, dan harus jauh lebih kecil dari ukuran set data pelatihan untuk menghabiskan lebih sedikit memori.

```
#Make the model learn ( Fit the model)
model.fit(X_train, Y_train,batch_size=1000, nb_epoch=10,validation_data=(X_test, Y_test))
```

```
Train on 50000 samples, validate on 10000 samples
Epoch 1/10
 1000/50000 [..............................] - ETA: 6s - loss: 2.3028 - acc: 0.1060
C:\ProgramData\Anaconda3\lib\site-packages\keras\models.py:848: UserWarning: The `nb_epoch` argument in `fit` has been renamed
`epochs`.
  warnings.warn('The `nb_epoch` argument in `fit` '
50000/50000 [==============================] - 6s - loss: 2.3030 - acc: 0.0974 - val_loss: 2.3027 - val_acc: 0.1000
Epoch 2/10
50000/50000 [==============================] - 7s - loss: 2.3029 - acc: 0.1012 - val_loss: 2.3027 - val_acc: 0.1000
Epoch 3/10
50000/50000 [==============================] - 7s - loss: 2.3028 - acc: 0.0972 - val_loss: 2.3026 - val_acc: 0.1000
Epoch 4/10
50000/50000 [==============================] - 7s - loss: 2.3028 - acc: 0.0997 - val_loss: 2.3027 - val_acc: 0.1000
Epoch 5/10
50000/50000 [==============================] - 7s - loss: 2.3029 - acc: 0.0975 - val_loss: 2.3027 - val_acc: 0.1000
Epoch 6/10
50000/50000 [==============================] - 7s - loss: 2.3029 - acc: 0.0986 - val_loss: 2.3028 - val_acc: 0.1000
Epoch 7/10
50000/50000 [==============================] - 7s - loss: 2.3029 - acc: 0.0995 - val_loss: 2.3028 - val_acc: 0.1000
Epoch 8/10
50000/50000 [==============================] - 7s - loss: 2.3028 - acc: 0.0983 - val_loss: 2.3027 - val_acc: 0.1000
Epoch 9/10
50000/50000 [==============================] - 7s - loss: 2.3029 - acc: 0.0998 - val_loss: 2.3027 - val_acc: 0.1000
Epoch 10/10
50000/50000 [==============================] - 7s - loss: 2.3029 - acc: 0.0972 - val_loss: 2.3027 - val_acc: 0.1000

<keras.callbacks.History at 0x2870136eef0>
```

Mengevaluasi Model

Setelah melatih jaringan saraf pada set data pelatihan, Anda perlu mengevaluasi kinerja jaringan. Perhatikan bahwa ini hanya akan memberi Anda gambaran tentang seberapa baik Anda telah memodelkan set data (misalnya, akurasi pelatihan), tetapi Anda tidak akan tahu seberapa baik kinerja algoritme pada data baru. Ini untuk kesederhanaan, tetapi idealnya, Anda dapat memisahkan data Anda menjadi set data pelatihan dan pengujian untuk pelatihan dan evaluasi model Anda.

```
#Evaluate how the model does on the test set
score = model.evaluate(X_test, Y_test, verbose=0)
#Accuracy Score
print('Test accuracy:', score[1])
```

Anda dapat mengevaluasi model Anda pada set data pelatihan Anda menggunakan fungsi `evaluation()` pada model Anda dan meneruskannya input dan output yang sama yang digunakan untuk melatih model. Ini akan menghasilkan prediksi untuk setiap pasangan input dan output dan mengumpulkan skor, termasuk kerugian rata-rata dan metrik apa pun yang telah Anda konfigurasikan, seperti akurasi.

Prediksi

Setelah Anda membangun dan mengevaluasi model, Anda perlu memprediksi data yang tidak diketahui.

```
#Predict digit(0-9) for test Data
model.predict_classes(X_test)
```

```
 9888/10000 [============================>.] - ETA: 0s
array([3, 8, 8, ..., 3, 4, 7], dtype=int64)
```

Simpan dan Muat Ulang Model

Berikut langkah terakhirnya:

```
#Saving the model
model.save('model.h5')
jsonModel = model.to_json()
model.save_weights('modelWeight.h5')
```

```
#Load weight of the saved model
modelWt = model.load_weights('modelWeight.h5')
```

Opsional: Rangkum Model

Sekarang mari kita lihat cara meringkas model.

```
#Summary of the model
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_7 (Dense)              (None, 512)               1573376
_________________________________________________________________
activation_7 (Activation)    (None, 512)               0
_________________________________________________________________
dropout_5 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_8 (Dense)              (None, 120)               61560
_________________________________________________________________
activation_8 (Activation)    (None, 120)               0
_________________________________________________________________
dropout_6 (Dropout)          (None, 120)               0
_________________________________________________________________
dense_9 (Dense)              (None, 10)                1210
_________________________________________________________________
activation_9 (Activation)    (None, 10)                0
=================================================================
Total params: 1,636,146
Trainable params: 1,636,146
Non-trainable params: 0
_________________________________________________________________
```

2.2 MENINGKATKAN MODEL KERAS

Berikut ini beberapa langkah tambahan untuk meningkatkan model Anda:

1. Terkadang, proses pembuatan model tidak selesai karena gradien yang hilang atau meledak. Jika demikian, Anda harus melakukan hal berikut:

```
from keras.callbacks import EarlyStopping
early_stopping_monitor = EarlyStopping (patience=2)
model.fit (x train, y train, batch_size=1000, epochs=10,
validation data=(x test, y_test),
callbacks=[early_stopping_monitor])
```

2. Modelkan bentuk keluaran.
```
#Shape of the n-dim array (output of the model at the current position)
 model.output_sh
```

3. Modelkan representasi ringkasan.
```
model.summary()
```

4. Modelkan konfigurasinya.
```
model.get_config()
```

5. Buat daftar semua tensor bobot dalam model.
```
model.get_weights()
```

Berikut ini saya bagikan kode lengkap untuk model Keras. Bisakah Anda mencoba menjelaskannya?

```
# A TYPING DEEP LEARNING MODEL WITH KERAS
     import numpy as np
     from keras.models import Sequential
     from keras.layers import Dense

# Loading Data
     data = np.random.random((500,100))
     labels = np.random.randint(2,size=(500,1))

# Create model
     model = Sequential()
     model.add(Dense(12, input_dim=8, activation='relu'))
     model.add(Dense(8, activation='relu'))
     model.add(Dense(1, activation='sigmoid'))

# Compile model
     model.compile(loss='binary_crossentropy',
     optimizer='adam',metrics=['accuracy'])

# Fit the model
     model.fit(X[train], Y[train], epochs=150, batch_size=10, verbose=0)

# Evaluate the model
     scores = model.evaluate(X[test], Y[test], verbose=0)
     print("%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
     cvscores.append(scores[1] * 100) print("%.2f%% (+/-%.2f%%)"%
```

```
    (numpy.mean(cvscores), numpy.std(cvscores)))

# Predict
    predictions = model.predict(data)
```

2.3 KERAS DENGAN TENSORFLOW

Keras menyediakan jaringan neural tingkat tinggi dengan memanfaatkan pustaka pembelajaran mendalam yang canggih dan jelas di atas TensorFlow/Theano. Keras merupakan tambahan yang bagus untuk TensorFlow karena lapisan dan modelnya kompatibel dengan tensor TensorFlow murni. Selain itu, Keras dapat digunakan bersama pustaka TensorFlow lainnya.

Berikut adalah langkah-langkah yang terlibat dalam penggunaan Keras untuk TensorFlow:

1. Mulailah dengan membuat sesi TensorFlow dan mendaftarkannya dengan Keras. Ini berarti Keras akan menggunakan sesi yang Anda daftarkan untuk menginisialisasi semua variabel yang dibuatnya secara internal.

```
import TensorFlow as tf
sess = tf.Session()
from keras import backend as K
K.set_session(sess)
```

2. Modul Keras seperti model, lapisan, dan aktivasi digunakan untuk membangun model. Mesin Keras secara otomatis mengonversi modul-modul ini ke skrip yang setara dengan TensorFlow.
3. Selain TensorFlow, Theano dan CNTK dapat digunakan sebagai backend untuk Keras.
4. Backend TensorFlow memiliki konvensi untuk membuat bentuk input (ke lapisan pertama jaringan Anda) dalam urutan kedalaman, tinggi, lebar, di mana kedalaman dapat berarti jumlah saluran.
5. Anda perlu mengonfigurasi file `keras.json` dengan benar sehingga menggunakan back-end TensorFlow. Tampilannya akan seperti ini:

```
{
        "backend": "theano",
        "epsilon": 1e-07,
        "image_data_format": "channels_first",
        "floatx": "float32"
}
```

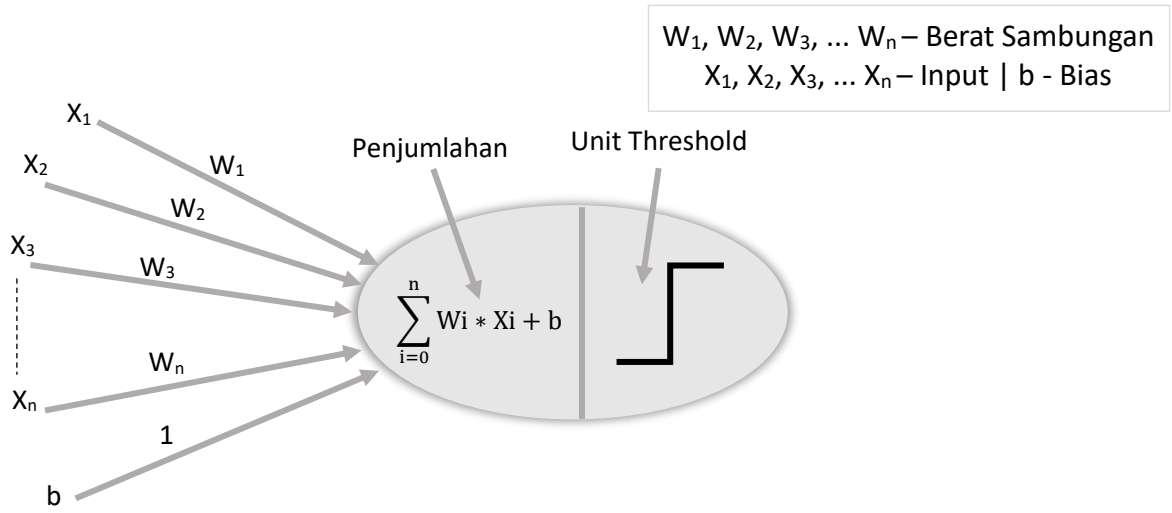
Pada bab berikutnya, Anda akan mempelajari cara memanfaatkan Keras untuk bekerja pada CNN, RNN, LSTM, dan aktivitas pembelajaran mendalam lainnya.

BAB 3
MULTILAYER PERCEPTRON

Sebelum Anda mulai mempelajari tentang multilayered perceptron, Anda perlu mendapatkan gambaran besar tentang jaringan saraf tiruan. Itulah yang akan saya mulai dalam bab ini.

3.1 JARINGAN SARAF TIRUAN

Jaringan saraf tiruan (ANN) adalah jaringan komputasional (sistem simpul dan interkoneksi antara simpul) yang terinspirasi oleh jaringan saraf biologis, yang merupakan jaringan neuron kompleks dalam otak manusia (lihat Gambar 3-1). Simpul yang dibuat dalam ANN seharusnya diprogram untuk berperilaku seperti neuron sebenarnya, dan karenanya mereka adalah neuron buatan. Gambar 3-1 menunjukkan jaringan simpul (neuron buatan) yang membentuk jaringan saraf tiruan.



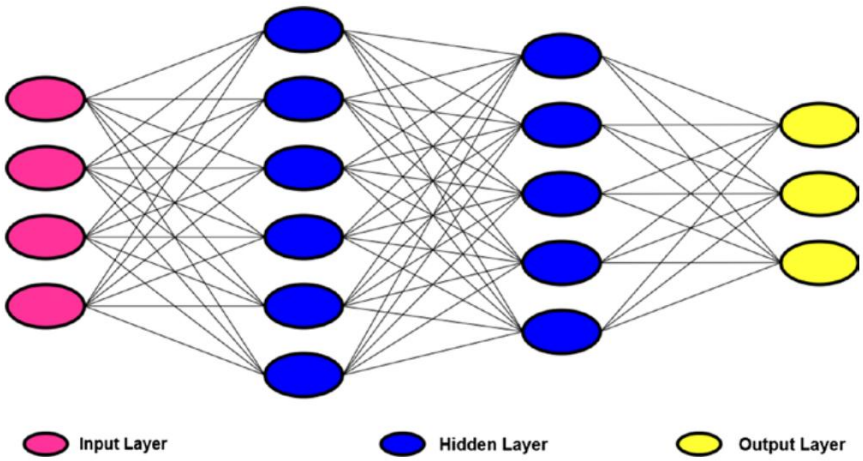
Gambar 3-1. Jaringan saraf tiruan

Jumlah lapisan dan jumlah neuron/simpul per lapisan dapat menjadi komponen struktural utama dari jaringan saraf tiruan. Awalnya, bobot (yang mewakili interkoneksi) dan bias tidak cukup baik untuk membuat keputusan (klasifikasi, dll.). Ini seperti otak bayi yang tidak memiliki pengalaman sebelumnya. Bayi belajar dari pengalaman agar menjadi pembuat keputusan yang baik (pengklasifikasi). Pengalaman/data (berlabel) membantu jaringan saraf otak menyetel bobot dan bias (saraf). Jaringan saraf tiruan melalui proses yang sama. Bobot disetel per iterasi untuk membuat pengklasifikasi yang baik. Karena penyetelan dan dengan demikian mendapatkan bobot yang benar secara manual untuk ribuan neuron sangat memakan waktu, Anda menggunakan algoritme untuk melakukan tugas-tugas ini.

Proses penyetelan bobot itu disebut pembelajaran atau pelatihan. Ini sama dengan apa yang dilakukan manusia setiap hari. Kami mencoba memungkinkan komputer untuk bekerja

seperti manusia.

Mari kita mulai menjelajahi model ANN yang paling sederhana. Jaringan saraf tipikal mengandung sejumlah besar neuron buatan yang disebut unit yang tersusun dalam serangkaian lapisan berbeda: lapisan masukan, lapisan tersembunyi, dan lapisan keluaran (Gambar 3-2).



Gambar 3-2. Jaringan saraf

Jaringan saraf terhubung, yang berarti setiap neuron dalam lapisan tersembunyi terhubung sepenuhnya ke setiap neuron dalam lapisan masukan sebelumnya dan ke lapisan keluaran berikutnya. Jaringan saraf belajar dengan menyesuaikan bobot dan bias di setiap lapisan secara berulang untuk mendapatkan hasil yang optimal.

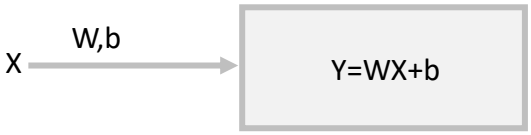
Perceptron Lapisan Tunggal

Perceptron lapisan tunggal adalah pengklasifikasi biner linier sederhana. Ia mengambil masukan dan bobot terkait dan menggabungkannya untuk menghasilkan keluaran yang digunakan untuk klasifikasi. Ia tidak memiliki lapisan tersembunyi. Regresi logistik adalah perceptron lapisan tunggal.

Perceptron Multilapis

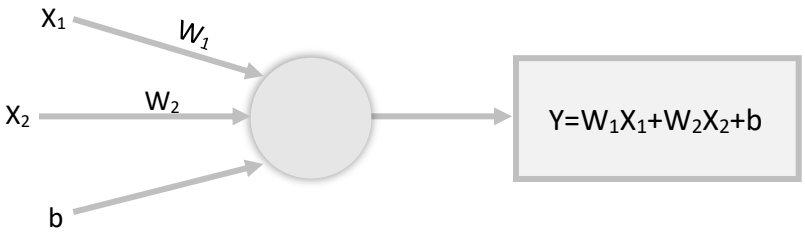
Perceptron multilapis (MLP) adalah contoh sederhana dari jaringan saraf buatan umpan balik. MLP terdiri dari setidaknya satu lapisan tersembunyi dari simpul selain lapisan masukan dan lapisan keluaran. Setiap simpul dari lapisan selain lapisan masukan disebut neuron yang menggunakan fungsi aktivasi nonlinier seperti sigmoid atau ReLU. MLP menggunakan teknik pembelajaran terbimbing yang disebut backpropagation untuk pelatihan, sekaligus meminimalkan fungsi kerugian seperti entropi silang. MLP menggunakan pengoptimal untuk menyetel parameter (bobot dan bias).

Beberapa lapisan dan aktivasi nonliniernya membedakan MLP dari perceptron linier. Perceptron multilapis adalah bentuk dasar dari jaringan saraf dalam. Sebelum mempelajari MLP, mari kita lihat model linier dan model logistik. Anda dapat memahami perbedaan halus antara model linier, logistik, dan MLP dalam hal kompleksitas. Gambar 3-3 menunjukkan model linier dengan satu masukan (X) dan satu keluaran (Y).



Gambar 3-3. Vektor masukan tunggal

Model masukan tunggal memiliki vektor X dengan bobot W dan bias b. Keluarannya, Y, adalah WX + b, yang merupakan model linier.
Gambar 3-4 menunjukkan beberapa masukan (X_1 dan X_2) dan satu keluaran (Y).

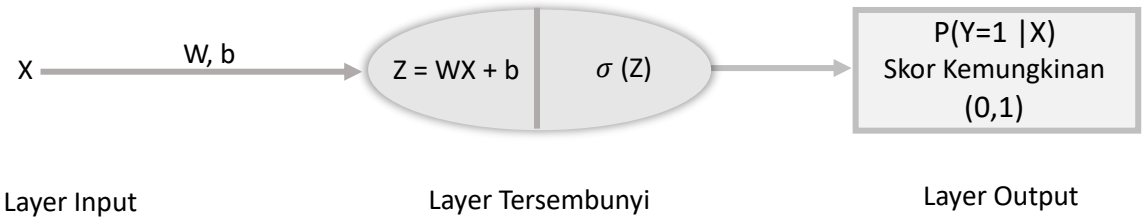


Gambar 3-4. Model linier

Model linier ini memiliki dua fitur masukan: X_1 dan X_2 dengan bobot yang sesuai untuk setiap fitur masukan yaitu W_1, W_2, dan bias b. Keluarannya, Y, adalah $W_1X_1 + W_2X_2 + b$.

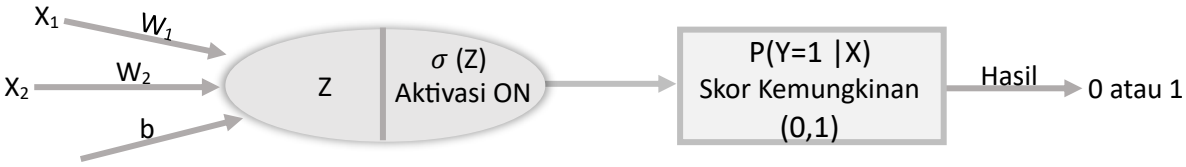
3.2 MODEL REGRESI LOGISTIK

Gambar 3-5 menunjukkan algoritme pembelajaran yang Anda gunakan saat label keluaran Y adalah 0 atau 1 untuk masalah klasifikasi biner. Diberikan vektor fitur masukan X, Anda menginginkan probabilitas bahwa Y = 1 berdasarkan fitur masukan X. Ini juga disebut sebagai jaringan saraf dangkal atau jaringan saraf satu lapis (tanpa lapisan tersembunyi; hanya satu lapisan keluaran). Lapisan keluaran, Y, adalah σ(Z), di mana Z adalah WX + b dan σ adalah fungsi sigmoid.



Gambar 3-5. Satu masukan (X) dan satu keluaran (Y)

Gambar 3-6 menunjukkan algoritma pembelajaran yang Anda gunakan saat label keluaran Y adalah 0 atau 1 untuk masalah klasifikasi biner.

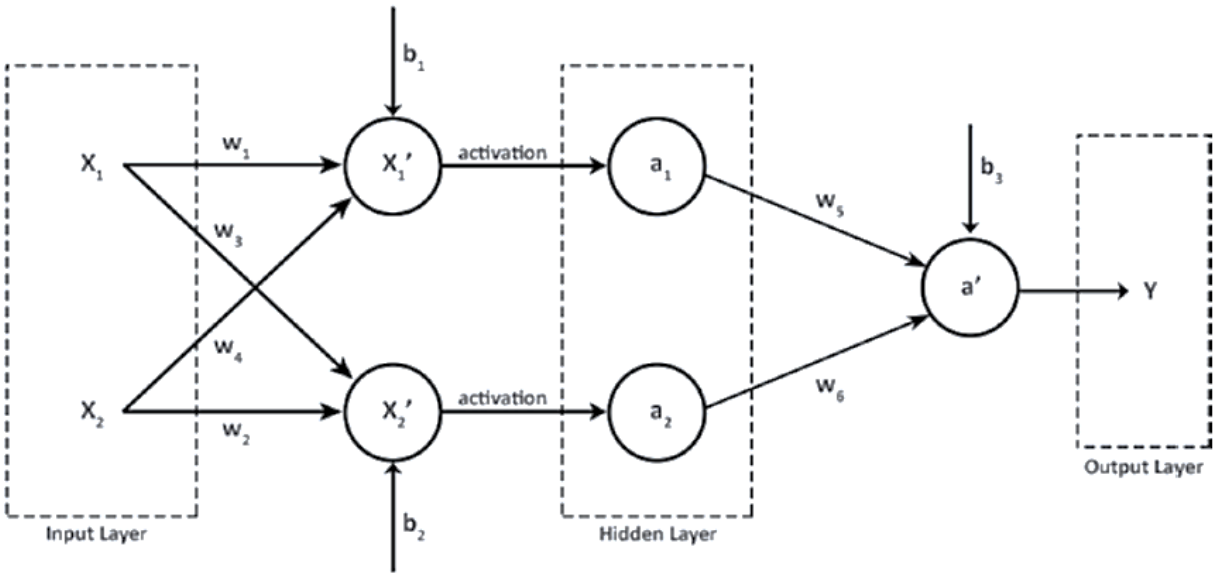


Gambar 3-6. Beberapa masukan (X1 dan X1) dan satu keluaran (Y)

Dengan vektor fitur input X1 dan X2, Anda menginginkan probabilitas bahwa Y = 1 berdasarkan fitur input. Ini juga disebut perceptron. Lapisan output, Y, adalah σ(Z), di mana Z adalah WX + b.

$$\begin{bmatrix} X1 \\ X2 \end{bmatrix} \rightarrow \begin{bmatrix} W1 & W2 \\ W3 & W4 \end{bmatrix} \begin{bmatrix} X1 \\ X2 \end{bmatrix} + \begin{bmatrix} b1 \\ b2 \end{bmatrix} \rightarrow \sigma\left(\begin{bmatrix} W1 * X1 + W2 * X2 + b1 \\ W3 * X1 + W4 * X2 + b2 \end{bmatrix}\right)$$

Gambar 3-7 menunjukkan jaringan saraf dua lapis, dengan lapisan tersembunyi dan lapisan keluaran. Anggaplah Anda memiliki dua vektor fitur masukan X1 dan X2 yang terhubung ke dua neuron, X1' dan X2.' Parameter (bobot) yang dikaitkan dari lapisan masukan ke lapisan tersembunyi adalah w1, w2, w3, w4, b1, b2.

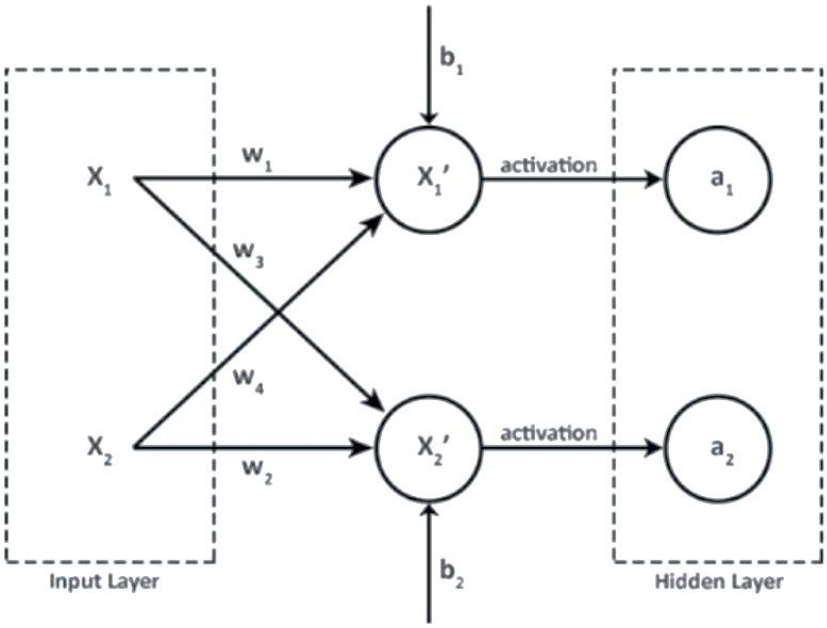


Gambar 3-7. Jaringan saraf dua lapis

X1' dan X2' menghitung kombinasi linier (Gambar 3-8).

$$\begin{bmatrix} X1' \\ X2' \end{bmatrix} = \begin{bmatrix} w1 & w2 \\ w3 & w4 \end{bmatrix} \begin{bmatrix} X1 \\ X2 \end{bmatrix} + \begin{bmatrix} b1 \\ b2 \end{bmatrix}$$

$(2 \times 1)(2 \times 2)(2 \times 1)(2 \times 1)$ adalah dimensi lapisan input dan tersembunyi.



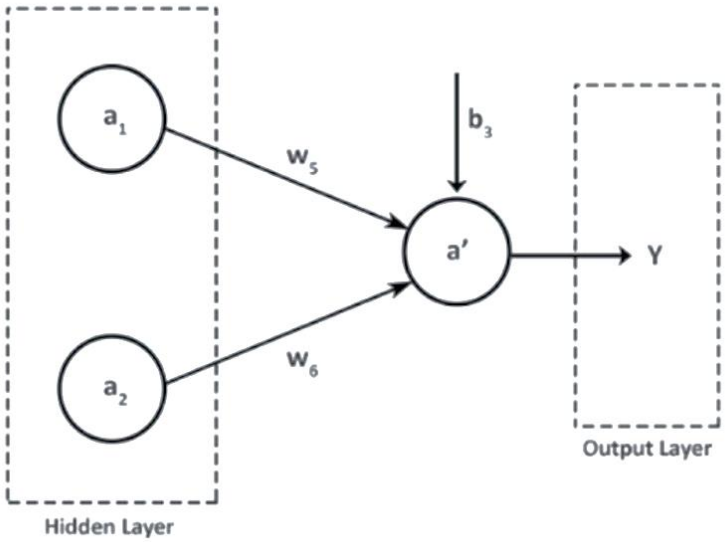
Gambar 3-8. Perhitungan dalam jaringan saraf

Input linier X1’ dan X2’ melewati unit aktivasi a1 dan a2 di lapisan tersembunyi.

a1 adalah σ(X1’) dan a2 adalah σ(X2’), jadi Anda juga dapat menuliskan persamaan sebagai berikut:

$$\begin{bmatrix} a1 \\ a2 \end{bmatrix} = \sigma \begin{bmatrix} X1' \\ X2' \end{bmatrix}$$

Nilai maju disebarkan dari lapisan tersembunyi ke lapisan keluaran. Input a1 dan a2 serta parameter w5, w6, dan b3 melewati lapisan keluaran a’ (Gambar 3-9).



Gambar 3-9. Perambatan maju

$\mathrm{a}' = [w5 \quad w6] \begin{bmatrix} a1 \\ a2 \end{bmatrix} + [b3]$ menciptakan kombinasi linear dari $(\mathrm{w5} * \mathrm{a1} + \mathrm{w6} *$

$\mathrm{a2}) + \mathrm{b3}$, yang akan melewati fungsi sigmoid nonlinier ke lapisan keluaran akhir, Y.

$$y = \sigma(\mathrm{a}')$$

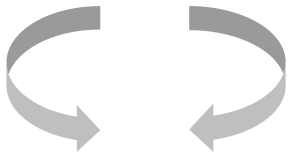
Misalkan struktur model awal dalam satu dimensi adalah $\mathrm{Y} = \mathrm{w} * \mathrm{X} + \mathrm{b}$, di mana parameter w dan b adalah bobot dan bias.

Pertimbangkan fungsi kerugian $\mathrm{L}(\mathrm{w}, \mathrm{b}) = 0.9$ untuk nilai awal parameter $\mathrm{w} = 1$ dan $\mathrm{b} = 1$. Anda mendapatkan keluaran ini: $\mathrm{y} = 1 * \mathrm{X} + 1 \& \mathrm{L}(\mathrm{w}, \mathrm{b}) = 0.9$.

Tujuannya adalah meminimalkan kerugian dengan menyesuaikan parameter w dan b. Kesalahan akan dipropagasi balik dari lapisan keluaran ke lapisan tersembunyi ke lapisan masukan untuk menyesuaikan parameter melalui laju pembelajaran dan pengoptimal. Terakhir, kami ingin membangun model (regresor) yang dapat menjelaskan Y dalam bentuk X.

Untuk memulai proses membangun model, kami menginisialisasi bobot dan bias. Untuk kenyamanan, $\mathrm{w} = 1, \mathrm{b} = 1$ (Nilai awal), (pengoptimal) penurunan gradien stokastik dengan laju pembelajaran $(\alpha = 0{,}01)$.

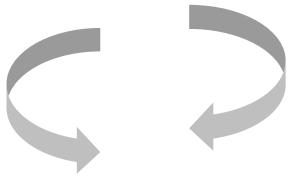
Berikut ***langkah 1****:* $\mathrm{Y} = 1 * \mathrm{X} + 1$.



1.20 0.35

Parameter disesuaikan menjadi $\mathrm{w} = 1{,}20$ dan $\mathrm{b} = 0{,}35$.

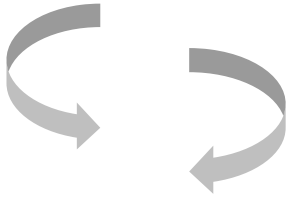
Berikut ***langkah 2****:* $\mathrm{Y1} = 1{,}20 * \mathrm{X} + 0{,}35$.



1.24 0.31

Parameter disesuaikan menjadi $\mathrm{w} = 1{,}24$ dan $\mathrm{b} = 0{,}31$.

Berikut ***langkah 3***: $\mathrm{Y1} = 1{,}24 * \mathrm{X} + 0{,}31$.



1.25 0.30

Setelah beberapa iterasi, bobot dan bias menjadi stabil. Seperti yang Anda lihat, perubahan awal tinggi saat penyetelan. Setelah beberapa iterasi, perubahannya tidak signifikan.

$\mathrm{L}(\mathrm{w},\mathrm{b})$ diminimalkan untuk $\mathrm{w} = 1{,}26$ dan $\mathrm{b} = 0{,}29$; oleh karena itu, model akhir menjadi sebagai berikut:

$$\mathrm{Y} = 1.26 * \mathrm{X} + 0.29$$

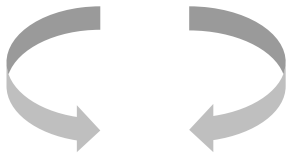
Demikian pula, dalam dua dimensi, Anda dapat mempertimbangkan parameter, matriks bobot, dan vektor bias.

Mari kita asumsikan bahwa matriks bobot awal dan vektor bias sebagai $\mathrm{W} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$ dan $\mathrm{B} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

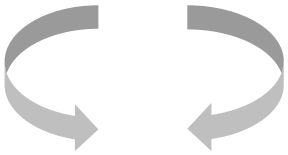
Anda mengulangi dan menyebarkan kembali kesalahan untuk menyesuaikan w dan b.

$\mathrm{Y} = \mathrm{W} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} * [\mathrm{X}] + \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ adalah model awal. Matriks bobot $(2\mathrm{x}2)$ dan matriks bias $(2\mathrm{x}1)$ disetel pada setiap iterasi. Jadi, kita dapat melihat perubahan pada matriks bobot dan bias

Berikut ***langkah 1:***



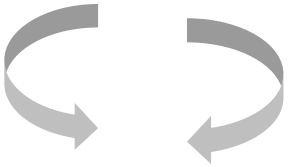
$$W = \begin{bmatrix} 0.7 & 0.8 \\ 0.6 & 1.2 \end{bmatrix}, B = \begin{bmatrix} 2.4 \\ 3.2 \end{bmatrix}$$



$$\begin{bmatrix} 0.7 & 0.8 \\ 0.6 & 1.2 \end{bmatrix} \quad \begin{bmatrix} 2.4 \\ 3.2 \end{bmatrix}$$

Berikut ***langkah 2:***

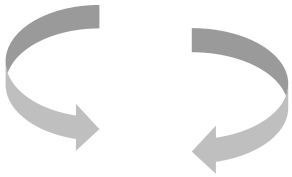
$$\begin{bmatrix} 0.7 & 0.8 \\ 0.6 & 1.2 \end{bmatrix} \quad \begin{bmatrix} 2.4 \\ 3.2 \end{bmatrix}$$



$$W = \begin{bmatrix} 0.6 & 0.7 \\ 0.4 & 1.3 \end{bmatrix}, B = \begin{bmatrix} 2.8 \\ 3.8 \end{bmatrix}$$

Berikut ***langkah 3:***

$$\begin{bmatrix}0.6 & 0.7\\ 0.4 & 1.3\end{bmatrix} \qquad \begin{bmatrix}2.8\\ 3.8\end{bmatrix}$$



Anda dapat melihat perubahan pada matriks bobot $(2x2)$ dan matriks bias $(2x1)$ dalam iterasi.

$$W = \begin{bmatrix}0.5 & 0.6\\ 0.3 & 1.3\end{bmatrix}, B = \begin{bmatrix}2.9\\ 4.0\end{bmatrix}$$

Model akhir setelah w dan b disesuaikan adalah sebagai berikut:

$$\mathrm{Y} = \begin{bmatrix}0.4 & 0.5\\ 0.2 & 1.3\end{bmatrix} * [\mathrm{X}] + \begin{bmatrix}3.0\\ 4.0\end{bmatrix}$$

Dalam bab ini, Anda mempelajari bagaimana bobot dan bias disetel dalam setiap iterasi sambil tetap menjaga tujuan meminimalkan fungsi kerugian. Itu dilakukan dengan bantuan pengoptimal seperti penurunan gradien stokastik.

Dalam bab ini, kita telah memahami ANN dan MLP sebagai model pembelajaran mendalam dasar. Di sini, kita dapat melihat MLP sebagai perkembangan alami dari regresi linier dan logistik. Kita telah melihat bagaimana bobot dan bias disetel dalam setiap iterasi yang terjadi dalam backpropagation. Tanpa membahas detail backpropagation, kita telah melihat tindakan/hasil backpropagation. Dalam dua bab berikutnya, kita dapat mempelajari cara membangun model MLP di TensorFlow dan di keras.

BAB 4
REGRESI KE MLP DI TENSORFLOW

Orang-orang telah menggunakan regresi dan pengklasifikasi untuk waktu yang lama. Sekarang saatnya beralih ke topik jaringan saraf. Multilayered perseptrona (MLP) adalah model jaringan saraf sederhana tempat Anda dapat menambahkan satu atau beberapa lapisan tersembunyi di antara lapisan masukan dan keluaran.

Dalam bab ini, Anda akan melihat bagaimana TensorFlow dapat membantu Anda membangun model. Anda akan memulai dengan model paling dasar, yaitu model linier. Model logistik dan MLP juga dibahas dalam bab ini.

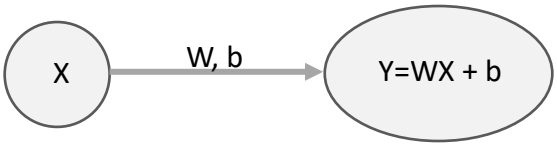
4.1 LANGKAH-LANGKAH TENSORFLOW UNTUK MEMBANGUN MODEL

Di bagian ini, penulis akan membahas langkah-langkah untuk membangun model di TensorFlow. Saya akan memandu Anda melalui langkah-langkah di sini, lalu Anda akan melihat kode di seluruh bab ini:

1. Muat data.
2. Pisahkan data menjadi train dan test.
3. Normalisasikan jika diperlukan.
4. Inisialisasi placeholder yang akan berisi prediktor dan target.
5. Buat variabel (bobot dan bias) yang akan disetel.
6. Nyatakan operasi model.
7. Nyatakan fungsi kerugian dan pengoptimal.
8. Inisialisasi variabel dan sesi.
9. Sesuaikan model dengan menggunakan loop pelatihan.
10. Periksa dan tampilkan hasilnya dengan data uji.

Regresi Linier dalam TensorFlow

Pertama-tama Anda perlu memahami kode untuk regresi linier dalam TensorFlow. Gambar 4-1 menunjukkan model linier dasar.

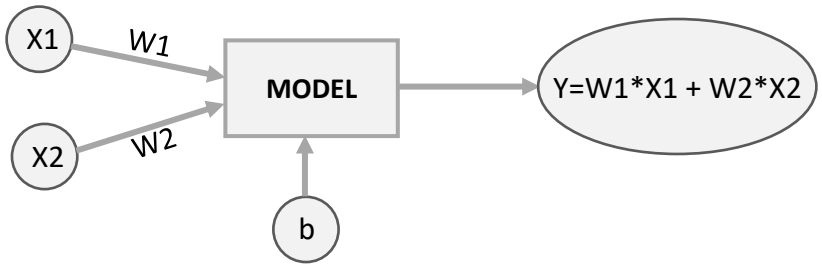


Gambar 4-1. Model linier dasar

Seperti yang ditunjukkan pada Gambar 4-1, bobot (W) dan bias (b) harus disetel untuk mendapatkan nilai bobot dan bias yang tepat. Jadi, bobot (W) dan bias (b) adalah variabel dalam kode TensorFlow; Anda akan menyetel/memodifikasinya dalam setiap iterasi hingga Anda mendapatkan nilai yang stabil (benar).

Anda perlu membuat placeholder untuk X. Placeholder memiliki bentuk tertentu dan berisi jenis tertentu.

Jika Anda memiliki lebih dari satu fitur, Anda akan memiliki model kerja yang mirip dengan Gambar 4-2.



Gambar 4-2. Model linier dengan beberapa masukan

Dalam kode berikut, Anda akan menggunakan kumpulan data Iris dari Seaborn, yang memiliki lima atribut. Anda akan mempertimbangkan panjang sepal sebagai masukan dan panjang petal sebagai nilai keluaran. Tujuan utama dari model regresi ini adalah untuk memprediksi panjang petal saat Anda diberi nilai panjang sepal. X adalah panjang sepal, dan Y adalah panjang petal.

Regresi linier menggunakan TensorFlow pada data Iris

```
############ Linear Regression: TensorFlow Way ##################
import matplotlib.pyplot as plt
import tensorflow as tf
from sklearn import datasets
import numpy as np
from sklearn.cross_validation import train_test_split
from matplotlib import pyplot
```

```
# 1. Load the data
# iris.data = [(Sepal Length, Sepal Width, Petal Length, Petal Width)]
iris = datasets.load_iris()
# X is Sepal.Length and Y is Petal Length
predictors_vals = np.array([predictors[0] for predictors in iris.data])
target_vals = np.array([predictors[2] for predictors in iris.data])
```

```
# 2.Split Data into train and test 80%-20%
x_trn, x_tst, y_trn, y_tst = train_test_split(predictors_vals, target_vals, test_size=0.2, random_state=12)
#training_idx = np.random.randint(x_vals.shape[0], size=80)
#training, test = x_vals[training_idx,:], x_vals[-training_idx,:]
```

```
# 3. Normalize if needed
# 4. Initialize placeholders that will contain predictors and target
predictor = tf.placeholder(shape=[None, 1], dtype=tf.float32)
target = tf.placeholder(shape=[None, 1], dtype=tf.float32)
```

```
#5. Create variables (Weight and Bias) that will be tuned up
A = tf.Variable(tf.zeros(shape=[1,1]))
b = tf.Variable(tf.ones(shape=[1,1]))
```

```
# 6. Declare model operations: Ax+b
model_output = tf.add(tf.matmul(predictor, A), b)
```

```
#7. Declare loss function and optimizer
#Declare loss function (L1 loss)
loss = tf.reduce_mean(tf.abs(target - model_output))
# Declare optimizer
my_opt = tf.train.GradientDescentOptimizer(0.01)
#my_opt = tftrain.AdamOptimizer(0.01)
train_step = my_opt.minimize(loss)
```

```
#8. Initialize variables and session
sess = tf.Session()
init = tf.global_variables_initializer()
sess.run(init)
```

```
#9. Fit Model by using Training Loops
# Training loop
lossArray = []
batch_size = 40
for i in range(200):
    rand_rows = np.random.randint(0, len(x_trn)-1, size=batch_size)
    batchX = np.transpose([x_trn[rand_rows]])
    batchY = np.transpose([y_trn[rand_rows]])
    sess.run(train_step, feed_dict={predictor: batchX, target: batchY})
    batchLoss = sess.run(loss, feed_dict={predictor: batchX, target: batchY})
    lossArray.append(batchLoss)
    if (i+1)%50==0:
        print('Step Number' + str(i+1) + ' A = ' + str(sess.run(A)) + ' b = ' + str(sess.run(b)))
        print('L1 Loss = ' + str(batchLoss))

[slope] = sess.run(A)
[y_intercept] = sess.run(b)
```

```
# 10. Check  and Display the result on test data
lossArray = []
batch_size = 30
for i in range(100):
    rand_rows = np.random.randint(0, len(x_tst)-1, size=batch_size)
    batchX = np.transpose([x_tst[rand_rows]])
    batchY = np.transpose([y_tst[rand_rows]])
    sess.run(train_step, feed_dict={predictor: batchX, target: batchY})
    batchLoss = sess.run(loss, feed_dict={predictor: batchX, target: batchY})
    lossArray.append(batchLoss)
    if (i+1)%20==0:
        print('Step Number: ' + str(i+1) + ' A = ' + str(sess.run(A)) + ' b = ' + str(sess.run(b)))
        print('L1 Loss = ' + str(batchLoss))
# Get the optimal coefficients
[slope] = sess.run(A)
[y_intercept] = sess.run(b)
```

```
# Original Data and Plot
plt.plot(x_tst, y_tst, 'o', label='Actual Data')
test_fit = []
for i in x_tst:
  test_fit.append(slope*i+y_intercept)
# predicted values and Plot
plt.plot(x_tst, test_fit, 'r-', label='Predicted line', linewidth=3)
plt.legend(loc='lower right')
plt.title('Petal Length vs Sepal Length')
plt.ylabel('Petal Length')
plt.xlabel('Sepal Length')
plt.show()
```

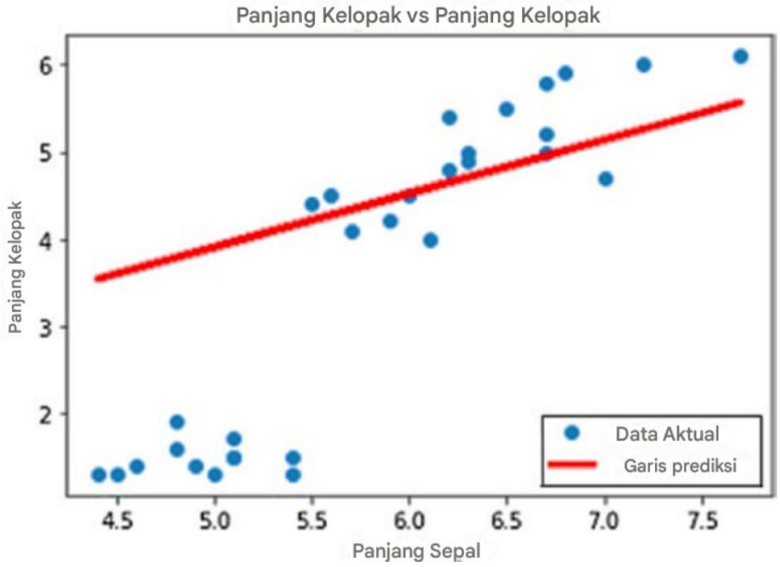
```
# Plot loss over time
plt.plot(lossArray, 'r-')
plt.title('L1 Loss per loop')
plt.xlabel('Loop')
plt.ylabel('L1 Loss')
plt.show()
```

Jika Anda menjalankan kode tersebut, Anda akan melihat output yang ditunjukkan pada Gambar 4-3.

```
Step Number50 A = [[ 0.57060003]] b = [[ 1.05275011]]
L1 Loss = 0.965698
Step Number100 A = [[ 0.56647497]] b = [[ 1.00924981]]
L1 Loss = 1.124
Step Number150 A = [[ 0.56645012]] b = [[ 0.96424991]]
L1 Loss = 1.18043
Step Number200 A = [[ 0.58122498]] b = [[ 0.92174983]]
L1 Loss = 1.20376
Step Number: 20 A = [[ 0.58945829]] b = [[ 0.90308326]]
L1 Loss = 1.18207
Step Number: 40 A = [[ 0.62599164]] b = [[ 0.88975]]
L1 Loss = 0.826957
Step Number: 60 A = [[ 0.63695836]] b = [[ 0.87108338]]
L1 Loss = 0.838114
Step Number: 80 A = [[ 0.60072505]] b = [[ 0.8450833]]
L1 Loss = 1.52654
Step Number: 100 A = [[ 0.6150251]] b = [[ 0.8290832]]
L1 Loss = 1.25477
```

Gambar 4-3. Bobot, bias, dan kerugian pada setiap Langkah

Gambar 4-4 menunjukkan plot untuk nilai prediksi panjang kelopak.

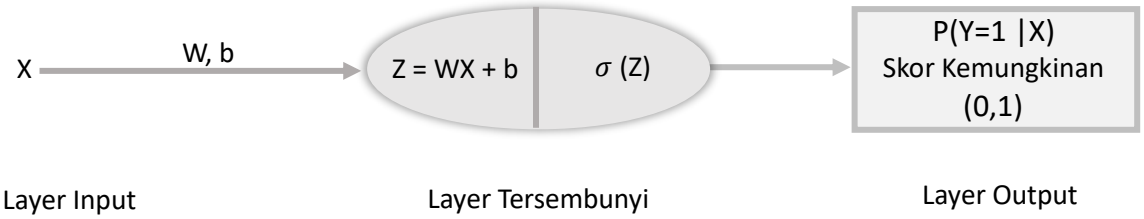


Gambar 4-4. Panjang kelopak versus panjang sepal

Model Regresi Logistik

Untuk klasifikasi, pendekatan yang paling sederhana adalah regresi logistik. Di bagian ini, Anda akan mempelajari cara melakukan regresi logistik di TensorFlow. Di sini Anda membuat bobot dan bias sebagai variabel sehingga ada ruang lingkup penyetelan/pengubahannya per iterasi. Placeholder dibuat untuk memuat X. Anda perlu membuat placeholder untuk X. Placeholder memiliki bentuk tertentu dan memuat jenis

tertentu, seperti yang ditunjukkan pada Gambar 4-5.



Gambar 4-5. Bagan model regresi logistik

Dalam kode berikut, Anda akan menggunakan kumpulan data Iris, yang memiliki lima atribut. Yang kelima adalah kelas target. Anda akan mempertimbangkan panjang sepal dan lebar sepal sebagai atribut prediktor dan spesies bunga sebagai nilai target. Tujuan utama dari model regresi logistik ini adalah untuk memprediksi jenis spesies saat Anda diberi nilai panjang sepal dan lebar sepal.

Buat file Python dan impor semua pustaka yang diperlukan.

```
######Logistic Regression in Tensorflow##########################
import numpy as np
import tensorflow as tf
from sklearn import datasets
import pandas as pd
from sklearn.cross_validation import train_test_split
from matplotlib import pyplot
# 1. Loading Data
iris = datasets.load_iris()
# Predictors Two columns : Sepal Length and Sepal Width
predictors_vals = np.array([predictor[0:2] for predictor in iris.data])
# For setosa Species, target is 0.
target_vals = np.array([1. if predictor==0 else 0. for predictor in iris.target])
```

```
# 2. Split data into train/test = 75%/25%
predictors_vals_train, predictors_vals_test, target_vals_train, target_vals_test = train_test_split(predictors_vals, target_val
s,
                                                    train_size=0.75,
                                                    random_state=0)
```

```
# 3. Normalize if needed
#4.Initialize placeholders that will contain predictors and target
x_data = tf.placeholder(shape=[None, 2], dtype=tf.float32)
y_target = tf.placeholder(shape=[None, 1], dtype=tf.float32)
```

```
#5. Create variables (Weight and Bias) that will be tuned up
W = tf.Variable(tf.ones(shape=[2,1]))
b = tf.Variable(tf.ones(shape=[1,1]))
```

```
# 6. Declare model operations : y = xW +b
model = tf.add(tf.matmul(x_data, W), b)
```

```
#7. Declare loss function and Optimizer
loss = tf.reduce_mean(tf.nn.sigmoid_cross_entropy_with_logits(logits=model, labels=y_target))
my_opt = tf.train.AdamOptimizer(0.02) #Learning rate =0.02
train_step = my_opt.minimize(loss)
```

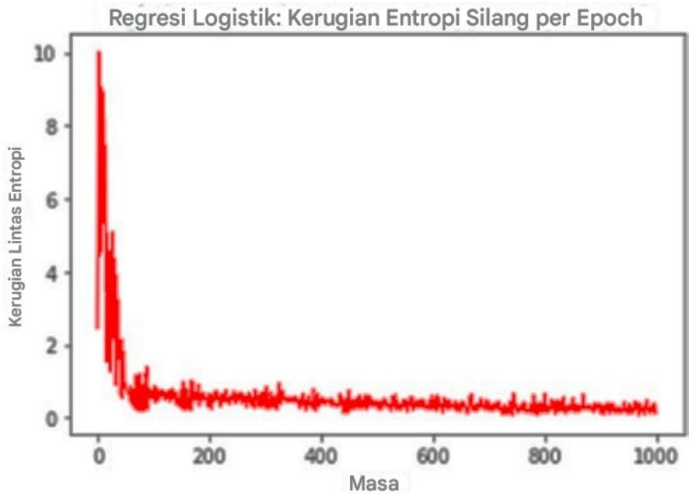
```
#8. Initialize variables and session
init = tf.global_variables_initializer()
sess=tf.Session()
sess.run(init)
```

```
#9. Actual Prediction:
prediction = tf.round(tf.sigmoid(model))
predictions_correct = tf.cast(tf.equal(prediction, y_target), tf.float32)
accuracy = tf.reduce_mean(predictions_correct)
```

```
#10. Training loop
lossArray = []
trainAccuracy = []
testAccuracy = []
for i in range(1000):
    #Random instances for Batch size
    batch_size = 4 #Declare batch size
    batchIndex = np.random.choice(len(predictors_vals_train), size=batch_size)
    batchX = predictors_vals_train[batchIndex]
    batchY = np.transpose([target_vals_train[batchIndex]])
    # Tuning weight and bias while minimizing loss function through optimizer
    sess.run(train_step, feed_dict={x_data: batchX, y_target: batchY})
    #Loss function per epoch/generation
    batchLoss = sess.run(loss, feed_dict={x_data: batchX, y_target: batchY})
    lossArray.append(batchLoss) # adding it to loss_vec
    # accuracy for each epoch for train
    batchAccuracyTrain = sess.run(accuracy, feed_dict={x_data: predictors_vals_train, y_target: np.transpose([target_vals_trai
n])})
    trainAccuracy.append(batchAccuracyTrain) # adding it to train_acc
    # accuracy for each epoch for test
    batchAccuracyTest = sess.run(accuracy, feed_dict={x_data: predictors_vals_test, y_target: np.transpose([target_vals_test])})
    testAccuracy.append(batchAccuracyTest)
    # printing loss after 10 epochs/generations to avoid verbosity
    if (i+1)%50==0:
        print('Loss = ' + str(batchLoss)+ ' and Accuracy = '    + str(batchAccuracyTrain))
```

```
# 11. Check model performance
pyplot.plot(lossArray, 'r-' )
pyplot.title('Logistic Regression: Cross Entropy Loss per Epoch')
pyplot.xlabel('Epoch')
pyplot.ylabel('Cross Entropy Loss')
pyplot.show()
```

Jika Anda menjalankan kode sebelumnya, plot kerugian entropi silang terhadap setiap zaman tampak seperti Gambar 4-6.



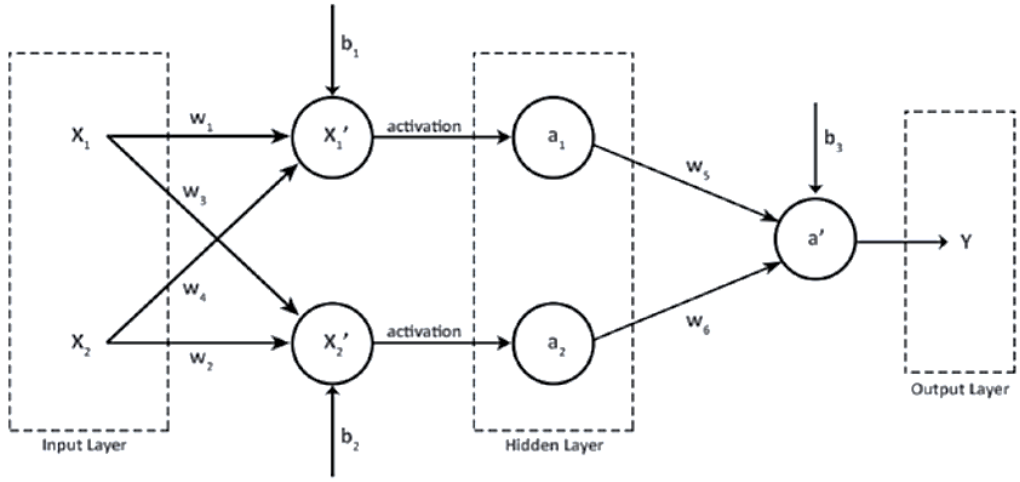
Gambar 4-6. Plot untuk kerugian entropi silang per epoch

4.2 MULTILAYER PERCEPTRON DI TENSORFLOW

Multilayer perceptron (MLP) adalah contoh sederhana dari jaringan saraf tiruan umpan balik. MLP terdiri dari setidaknya satu lapisan tersembunyi dari simpul selain lapisan masukan

dan lapisan keluaran. Setiap simpul dari lapisan selain lapisan masukan disebut neuron yang menggunakan fungsi aktivasi nonlinier seperti sigmoid atau ReLU. MLP menggunakan teknik pembelajaran terbimbing yang disebut backpropagation untuk pelatihan sambil meminimalkan fungsi kerugian seperti entropi silang dan menggunakan pengoptimal untuk menyetel parameter (bobot dan bias). Beberapa lapisan dan aktivasi nonliniernya membedakan MLP dari linear perceptron.

TensorFlow sangat cocok untuk membangun model MLP. Dalam MLP, Anda perlu menyetel bobot dan bias per iterasi. Ini berarti bobot dan bias terus berubah hingga menjadi stabil sambil meminimalkan fungsi kerugian. Jadi, Anda dapat membuat bobot dan bias sebagai variabel di TensorFlow. Saya cenderung memberi mereka nilai awal (semua 0 atau semua 1 atau beberapa nilai acak yang terdistribusi normal). Placeholder harus memiliki jenis nilai tertentu dan bentuk yang ditentukan, seperti yang ditunjukkan pada Gambar 4-7.



Gambar 4-7. Diagram alir untuk MLP

Impor semua pustaka yang diperlukan. Implementasi MLP di TensorFlow.

```
# Implementing a one-hidden layer Neural Network (MLP)

import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf
from sklearn import datasets
from sklearn.cross_validation import train_test_split
from matplotlib import pyplot
```

```
#2. Split data into train/test = 80%/20%

predictors_vals_train, predictors_vals_test, target_vals_train, target_vals_test= train_test_split(predictors_vals, target_val
s, test_size=0.2, random_state=12)
# 3. Normalize if needed
```

```
# 4.Initialize placeholders that will contain predictors and target
x_data = tf.placeholder(shape=[None, 3], dtype=tf.float32)
y_target = tf.placeholder(shape=[None, 1], dtype=tf.float32)
```

```
#5. Create variables (Weight and Bias) that will be tuned up
hidden_layer_nodes = 10
#For first layer
A1 = tf.Variable(tf.ones(shape=[3,hidden_layer_nodes])) # inputs -> hidden nodes
b1 = tf.Variable(tf.ones(shape=[hidden_layer_nodes]))   # one biases for each hidden node
#For second layer
A2 = tf.Variable(tf.ones(shape=[hidden_layer_nodes,1])) # hidden inputs -> 1 output
b2 = tf.Variable(tf.ones(shape=[1]))   # 1 bias for the output
```

```
# 6. Define Model Structure
hidden_output = tf.nn.relu(tf.add(tf.matmul(x_data, A1), b1))
final_output = tf.nn.relu(tf.add(tf.matmul(hidden_output, A2), b2))
```

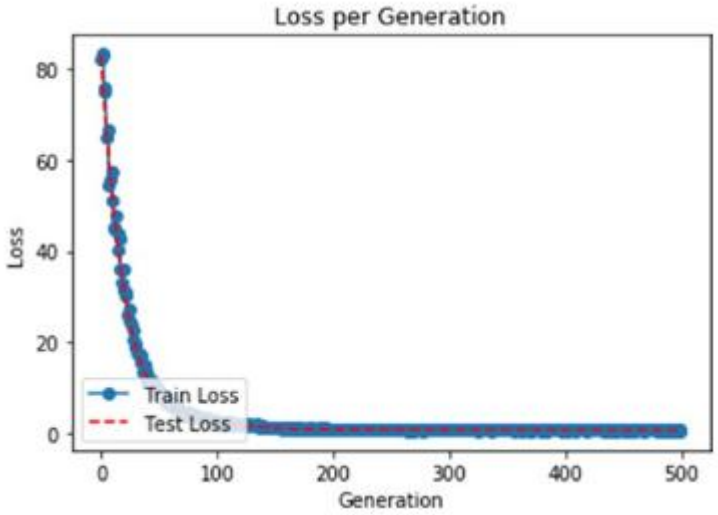
```
# 7. Declare loss function (MSE) and optimizer
loss = tf.reduce_mean(tf.square(y_target - final_output))
my_opt = tf.train.AdamOptimizer(0.02) # Learning rate = 0.02
train_step = my_opt.minimize(loss)
```

```
# 8.Initialize variables and session
init = tf.global_variables_initializer()
# Create graph session
sess = tf.Session()
sess.run(init)
```

```
# 9. Training Loop
lossArray = []
test_loss = []
batch_size =20
for i in range(500):
    batchIndex = np.random.choice(len(predictors_vals_train), size=batch_size)
    batchX = predictors_vals_train[batchIndex]
    batchY = np.transpose([target_vals_train[batchIndex]])
    sess.run(train_step, feed_dict={x_data: batchX, y_target: batchY})
    #
    batchLoss = sess.run(loss, feed_dict={x_data: batchX, y_target: batchY})
    lossArray.append(np.sqrt(batchLoss))

    test_temp_loss = sess.run(loss, feed_dict={x_data: predictors_vals_test, y_target: np.transpose([target_vals_test])})
    test_loss.append(np.sqrt(test_temp_loss))
    if (i+1)%50==0:
        print('Loss = ' + str(batchLoss))
```

```
# 10. Check model performance
#Plot loss(mean squared error) over time
pyplot.plot(lossArray, 'o-', label='Train Loss')
pyplot.plot(test_loss, 'r--', label='Test Loss')
pyplot.title('Loss per Generation')
pyplot.legend(loc='lower left')
pyplot.xlabel('Generation')
pyplot.ylabel('Loss')
pyplot.show()
```

Jika Anda menjalankan kode ini, Anda akan mendapatkan plot yang ditunjukkan pada Gambar 4-8.



Gambar 4-8. Plot untuk kerugian selama pelatihan dan pengujian

Dalam bab ini, saya membahas cara membangun model linear, logistik, dan MLP di TensorFlow secara sistemik.

BAB 5
REGRESI KE MLP DI KERAS

Anda telah mengerjakan regresi saat memecahkan aplikasi pembelajaran mesin. Regresi linier dan regresi nonlinier digunakan untuk memprediksi target numerik, sementara regresi logistik dan pengklasifikasi lainnya digunakan untuk memprediksi variabel target non-numerik. Dalam bab ini, saya akan membahas evolusi multilayer perceptron.

Secara khusus, Anda akan membandingkan akurasi yang dihasilkan oleh berbagai model dengan dan tanpa menggunakan Keras.

5.1 MODEL LOG-LINEAR

Buat file Python baru dan impor paket berikut. Pastikan Anda telah menginstal Keras di sistem Anda.

```
####### Log-Linear Model ########################################
from sklearn.datasets import load_iris
from sklearn.cross_validation import train_test_split
from sklearn.linear_model import LogisticRegressionCV
from sklearn.linear_model import LinearRegression
import numpy as np
import matplotlib.pyplot as plt
from keras.models import Sequential
from keras.layers import Dense, Activation
```

Anda akan menggunakan kumpulan data Iris sebagai sumber data. Jadi, muat kumpulan data dari Seaborn.

```
# Load the iris dataset from seaborn.
iris = load_iris()
```

Kumpulan data Iris memiliki lima atribut. Anda akan menggunakan empat atribut pertama untuk memprediksi spesies, yang kelasnya ditetapkan dalam atribut kelima dari kumpulan data tersebut.

```
# Use the first 4 variables to predict the species.
X, y = iris.data[:, :4], iris.target
```

Menggunakan fungsi `scikit-learn`, pisahkan set data pengujian dan pelatihan.

```
# Split both independent and dependent variables in half
# for cross-validation
train_X, test_X, train_y, test_y = train_test_split(X, y, train_size=0.5, random_state=0)
#print(type(train_X),len(train_y),len(test_X),len(test_y))
```

```
###################################
# scikit Learn for (Log) Linear Regression #
###################################
```

Gunakan fungsi `model.fit` untuk melatih model dengan set data pelatihan.

```
# Train a scikit-learn log-regression model
# lr =LogisticRegressionCV
# Train a scikit-learn linear-regression model
lr = LinearRegression()
lr.fit(train_X, train_y)
```

Saat model dilatih, Anda dapat memprediksi output dari set pengujian.

```
# Test the model. Print the accuracy on the test data
pred_y = lr.predict(test_X)
#print("Accuracy is {:.2f}".format(lr.score(test_X, test_y)))
```

Jaringan Syaraf Keras Untuk Regresi Linier

Sekarang, mari kita buat model jaringan syaraf Keras untuk regresi linier.

```
# Build the keras model
model = Sequential()
# 4 features in the input layer (the four flower measurements)
# 16 hidden units
model.add(Dense(16, input_shape=(4,)))
model.add(Activation('sigmoid'))
# 3 classes in the ouput layer (corresponding to the 3 species)
model.add(Dense(3))
model.add(Activation('softmax'))
```

```
# Compile the model
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Gunakan fungsi model.fit untuk melatih model dengan set data pelatihan.

```
# Fit/Train the keras model
model.fit(train_X, train_y, verbose=1, batch_size=1, nb_epoch=100)
```

Saat model dilatih, Anda dapat memprediksi output dari set pengujian.

```
# Test the model. Print the accuracy on the test data
loss, accuracy = model.evaluate(test_X, test_y, verbose=0)
print("\nAccuracy is using keras prediction  {:.2f}".format(accuracy))
```

Cetak akurasi yang diperoleh oleh kedua model.

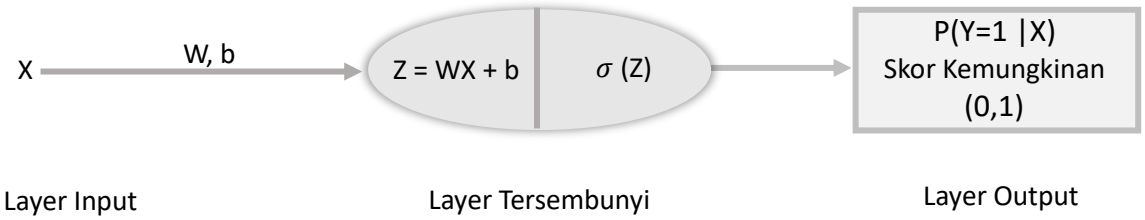
```
print("\nAccuracy is using keras prediction  {:.2f}".format(accuracy))

print("Accuracy is using regression  {:.2f}".format(lr.score(test_X, test_y)))
```

Jika Anda menjalankan kode tersebut, Anda akan melihat keluaran berikut:

```
Using TensorFlow backend.

Epoch 1/100

75/75 [==============================] - 0s - loss: 1.2947 - acc: 0.4533

Epoch 2/100

75/75 [==============================] - 0s - loss: 1.0353 - acc: 0.6400

Epoch 3/100

75/75 [==============================] - 0s - loss: 0.8930 - acc: 0.6533

...                              ...                              ...

...                              ...                              ...

...                              ...                              ...

...                              ...                              ...

Epoch 99/100

75/75 [==============================] - 0s - loss: 0.1186 - acc: 0.9733

Epoch 100/100

75/75 [==============================] - 0s - loss: 0.1167 - acc: 0.9867


Accuracy is using keras prediction  0.99

Accuracy is using regression  0.89
```

5.2 REGRESI LOGISTIK

Pada bagian ini, Penulis akan membagikan contoh regresi logistik sehingga Anda dapat membandingkan kode di scikit-learn dengan kode di Keras (lihat Gambar 5-1).



Gambar 5-1. Regresi logistik yang digunakan untuk klasifikasi

Buat file Python baru dan impor paket berikut. Pastikan Anda telah menginstal Keras di sistem Anda.

```
from sklearn.datasets import load_iris
import numpy as np
from sklearn.cross_validation import train_test_split
from sklearn.linear_model import LogisticRegressionCV
from keras.models import Sequential
from keras.layers.core import Dense, Activation
from keras.utils import np_utils
```

Anda akan menggunakan set data Iris sebagai sumber data. Jadi, muat set data dari scikit-learn.

```
# Load Data and Prepare data
iris = load_iris()
X, y = iris.data[:, :4], iris.target
```

Dengan menggunakan fungsi scikit-learn, pisahkan set data pengujian dan pelatihan.

```
# Load Data and Prepare data
iris = load_iris()
X, y = iris.data[:, :4], iris.target
```

scikit-learn untuk Regresi Logistik

Gunakan fungsi model.fit untuk melatih model dengan set data pelatihan. Setelah model dilatih, Anda dapat memprediksi keluaran set pengujian.

```
################################
# scikit Learn for  Logistic Regression
################################
lr = LogisticRegressionCV()
lr.fit(train_X, train_y)
pred_y = lr.predict(test_X)
print("Test fraction correct (LR-Accuracy) = {:.2f}".format(lr.score(test_X, test_y)))
```

Jaringan Neural Keras untuk Regresi Logistik

Pengodean one-hot mengubah fitur ke format yang berfungsi lebih baik dengan algoritme klasifikasi dan regresi.

```
################################
# Keras Neural Network for Logistic Regression
################################

# Make ONE-HOT enconding for converting into categorical variable
def one_hot_encode_object_array(arr):
    uniques, ids = np.unique(arr, return_inverse=True)
    return np_utils.to_categorical(ids, len(uniques))
```

```
# Dividing data into train and test data
train_y_ohe = one_hot_encode_object_array(train_y)
test_y_ohe = one_hot_encode_object_array(test_y)
#Creating a model
model = Sequential()
model.add(Dense(16, input_shape=(4,)))
model.add(Activation('sigmoid'))
model.add(Dense(3))
model.add(Activation('softmax'))
```

```
# Compiling the model
model.compile(loss='categorical_crossentropy', metrics=['accuracy'], optimizer='adam')
```

Gunakan fungsi model.fit untuk melatih model dengan set data pelatihan.

```
# Actual modelling
model.fit(train_X, train_y_ohe, verbose=0, batch_size=1, nb_epoch=100)
```

Gunakan fungsi model.evaluate untuk mengevaluasi kinerja model.

```
score, accuracy = model.evaluate(test_X, test_y_ohe, batch_size=16, verbose=0)
```

Cetak akurasi yang diperoleh oleh kedua model.
Akurasi untuk model berbasis scikit-learn

```
print("\n Test fraction correct (LR-Accuracy) logistic regression = {:.2f}".format(lr.score(test_X, test_y)))
```

Akurasinya adalah 0,83.

Akurasi untuk model keras

```
print("Test fraction correct (NN-Accuracy) keras  = {:.2f}".format(accuracy))
```

Akurasinya adalah 0,99.

Jika Anda menjalankan kode tersebut, Anda akan melihat keluaran berikut:

```
Using TensorFlow backend.
Test fraction correct (LR-Accuracy) logistic regression =
0.83
Test fraction correct (NN-Accuracy) keras  = 0.99
Epoch 1/100
75/75 [==============================] - 0s - loss: 1.2947 -
acc: 0.4533
Epoch 2/100
75/75 [==============================] - 0s - loss: 1.0353 -
acc: 0.6400
Epoch 3/100
75/75 [==============================] - 0s - loss: 0.8930 -
acc: 0.6533
...                                ...                   ...
...                                ...                   ...
...                                ...                   ...
...                                ...                   ...
Epoch 99/100
75/75 [==============================] - 0s - loss: 0.1186 -
acc: 0.9733
Epoch 100/100
75/75 [==============================] - 0s - loss: 0.1167 -
acc: 0.9867

Accuracy is using keras prediction  0.99
Accuracy is using regression  0.89
```

Untuk memberikan contoh nyata, saya akan membahas beberapa kode yang menggunakan set data Fashion MNIST, yang merupakan set data gambar Zalando.com yang terdiri dari set

pelatihan yang terdiri dari 60.000 contoh dan set pengujian yang terdiri dari 10.000 contoh. Setiap contoh adalah gambar skala abu-abu berukuran 28x28 yang dikaitkan dengan label dari sepuluh kelas.

Data MNIST Fashion: Regresi Logistik di Keras

Buat file Python baru dan impor paket berikut. Pastikan Anda telah menginstal Keras di sistem Anda.

```
from __future__ import print_function
from keras.models import load_model
import keras
import fashion_mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout
import numpy as np

batch_size = 128
num_classes = 10
epochs = 2
```

Seperti yang disebutkan, Anda akan menggunakan set data Fashion MNIST. Simpan data dan label dalam dua variabel yang berbeda.

```
# the data, shuffled and split between train and test sets
(x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()

x_train = x_train.reshape(60000, 784)
x_test = x_test.reshape(10000, 784)
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
```

Normalisasikan set data, seperti yang ditunjukkan di sini:

```
#Gaussian Normalization of the dataset
x_train = (x_train-np.mean(x_train))/np.std(x_train)
x_test = (x_test-np.mean(x_test))/np.std(x_test)

# convert class vectors to binary class matrices
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)
```

Tentukan modelnya, seperti yang ditunjukkan di sini:

```
#Building a model architecture
model = Sequential()
model.add(Dense(256, activation='elu', input_shape=(784,)))
model.add(Dropout(0.4))
model.add(Dense(512, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))

model.summary()

model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

model.fit(x_train, y_train,
                   batch_size=batch_size,
                   epochs=epochs,
                   validation_data=(x_test, y_test))
```

Simpan model dalam file .h5 (sehingga Anda dapat menggunakannya nanti secara langsung dengan fungsi load_model() dari keras.models) dan cetak akurasi model dalam set pengujian,

seperti yang ditunjukkan di sini:

```
#saving the model using the 'model.save' function
model.save('my_model.h5')
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

Jika Anda menjalankan kode sebelumnya, Anda akan melihat output berikut:

```
('train-images-idx3-ubyte.gz', <http.client.HTTPMessage object at 0x00000171338E2B38>)
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_59 (Dense)             (None, 256)               200960
_________________________________________________________________
dropout_10 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_60 (Dense)             (None, 512)               131584
_________________________________________________________________
dense_61 (Dense)             (None, 10)                5130
=================================================================
Total params: 337,674
Trainable params: 337,674
Non-trainable params: 0
_________________________________________________________________
Train on 60000 samples, validate on 10000 samples
Epoch 1/2
60000/60000 [==============================]
- loss: 0.5188 - acc: 0.8127 - val_loss: 0.4133 - val_acc: 0.8454
Epoch 2/2
60000/60000 [==============================]
- loss: 0.3976 - acc: 0.8545 - val_loss: 0.4010 - val_acc: 0.8513
Test loss: 0.400989927697
Test accuracy: 0.8513
```

5.3 MLP PADA DATA IRIS

Perceptron multilayer adalah model jaringan neural minimal. Di bagian ini, saya akan menunjukkan kodenya.

Tulis Kode

Buat file Python baru dan impor paket berikut. Pastikan Anda telah menginstal Keras di sistem Anda.

```
##########MLP on iris data ####################

import pandas as pd
import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Activation
from keras.utils import np_utils
```

Muat set data dengan membaca file CSV menggunakan Pandas.

```
#load and Prepare Data
datatrain = pd.read_csv('./Datasets/iris/iris_train.csv')
```

Tetapkan nilai numerik ke kelas set data.

```
#change string value to numeric
datatrain.set_value(datatrain['species']=='Iris-setosa',['species'],0)
datatrain.set_value(datatrain['species']=='Iris-versicolor',['species'],1)
datatrain.set_value(datatrain['species']=='Iris-virginica',['species'],2)
datatrain = datatrain.apply(pd.to_numeric)
```

Ubah bingkai data menjadi array.

```
#change dataframe to array
datatrain_array = datatrain.as_matrix()
```

Pisahkan data dan target, lalu simpan dalam dua variabel berbeda.

```
# split x and y (feature and target)
xtrain = datatrain_array[:,:4]
ytrain = datatrain_array[:,4]
```

Ubah format target menggunakan Numpy.

```
#change target format
ytrain = np_utils.to_categorical(ytrain)
```

Bangun Model Keras Berurutan

Di sini Anda akan membangun model perceptron multilapis dengan satu lapisan tersembunyi.

- Lapisan masukan: Lapisan masukan berisi empat neuron, yang mewakili fitur iris (panjang sepal, dll.).
- Lapisan tersembunyi: Lapisan tersembunyi berisi sepuluh neuron, dan aktivasi menggunakan ReLU.
- Lapisan keluaran: Lapisan keluaran berisi tiga neuron, yang mewakili kelas lapisan softmax Iris.

```
#Build Keras model
#Multilayer perceptron model, with one hidden layer.
#Input layer : 4 neuron, represents the feature of Iris(Sepal Length etc)
#Hidden layer : 10 neuron, activation using ReLU
#Output layer : 3 neuron, represents the class of Iris, Softmax Layer
model = Sequential()
model.add(Dense(output_dim=10, input_dim=4))
model.add(Activation("relu"))
model.add(Dense(output_dim=3))
model.add(Activation("softmax"))
```

Kompilasi model dan pilih pengoptimal dan fungsi kerugian untuk melatih dan mengoptimalkan data Anda, seperti yang ditunjukkan di sini:

```
#Compile model :choose optimizer and loss function
#optimizer = stochastic gradient descent with no batch-size
#loss function = categorical cross entropy
#learning rate = default from keras.optimizer.SGD, 0.01
model.compile(loss='categorical_crossentropy', optimizer='sgd', metrics=['accuracy'])
```

Latih model menggunakan fungsi model.fit, seperti yang ditunjukkan di sini:

```
#train
model.fit(xtrain, ytrain, nb_epoch=100, batch_size=120)
```

Muat dan persiapkan data pengujian, seperti yang ditunjukkan di sini:

```
## Evaluate on test data
#load and Prepare Data
datatest = pd.read_csv('./Datasets/iris/iris_test.csv')
```

Ubah nilai string menjadi nilai numerik, seperti yang ditunjukkan di sini:

```
#change string value to numeric
datatest.set_value(datatest['species']=='Iris-setosa',['species'],0)
datatest.set_value(datatest['species']=='Iris-versicolor',['species'],1)
datatest.set_value(datatest['species']=='Iris-virginica',['species'],2)
datatest = datatest.apply(pd.to_numeric)
```

Ubah kerangka data menjadi array, seperti yang ditunjukkan di sini:

```
#change dataframe to array
datatest_array = datatest.as_matrix()
```

Pisahkan x dan y, dengan kata lain, kumpulan fitur dan kumpulan target, seperti yang ditunjukkan di sini:

```
#split x and y (feature and target)
xtest= datatest_array[:,:4]
ytest = datatest_array[:,4]
```

Buat prediksi pada model yang telah dilatih, seperti yang ditunjukkan di sini:

```
#get prediction
classes = model.predict_classes(xtest, batch_size=120)
```

Hitung akurasinya, seperti yang ditunjukkan di sini:

```
#get accuration
accuration = np.sum(classes == ytest)/30.0 * 100
```

Cetak akurasi yang dihasilkan oleh model, seperti yang ditunjukkan di sini:

```
print("Test Accuration : " + str(accuration) + '%')
print("Prediction :")
print(classes)
print("Target :")
print(np.asarray(ytest,dtype="int32"))
```

Jika Anda menjalankan kode tersebut, Anda akan melihat output berikut:

```
Epoch 1/100 120/120 [==============================]
- 0s - loss: 2.7240 - acc: 0.3667
Epoch 2/100 120/120 [==============================]
- 0s - loss: 2.4166 - acc: 0.3667
Epoch 3/100 120/120 [==============================]
- 0s - loss: 2.1622 - acc: 0.4083
Epoch 4/100 120/120 [==============================]
- 0s - loss: 1.9456 - acc: 0.6583
Epoch 98/100 120/120 [==============================]
- 0s - loss: 0.5571 - acc: 0.9250
Epoch 99/100 120/120 [==============================]
- 0s - loss: 0.5554 - acc: 0.9250
Epoch 100/100 120/120 [==============================]
- 0s - loss: 0.5537 - acc: 0.9250
```

MLP pada Data MNIST (Klasifikasi Digit)

MNIST adalah kumpulan data standar untuk memprediksi digit tulisan tangan. Di bagian ini, Anda akan melihat cara menerapkan konsep multilayer perceptron dan membuat sistem pengenalan digit tulisan tangan.

Buat file Python baru dan impor paket berikut. Pastikan Anda telah menginstal Keras di sistem Anda.

```
##########MLP : MNIST Data (Digit Classification) ########################

import numpy as np
import os
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers.core import Dense, Dropout, Activation
from keras.optimizers import RMSprop
from keras.utils import np_utils
```

Beberapa variabel penting telah didefinisikan.

```
np.random.seed(100) # for reproducibility
batch_size = 128 #Number of images used in each optimization step
nb_classes = 10 #One class per digit
nb_epoch = 20 #Number of times the whole data is used to learn
```

Muat set data menggunakan fungsi `mnist.load_data()`.

```
(X_train, y_train), (X_test, y_test) = mnist.load_data()

#Flatten the data, MLP doesn't use the 2D structure of the data. 784 = 28*28
X_train = X_train.reshape(60000, 784) # 60,000 digit images
X_test = X_test.reshape(10000, 784)
```

Jenis set pelatihan dan set pengujian diubah menjadi `float32`.

```
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
```

Set data dinormalisasi; dengan kata lain, ditetapkan ke skor Z.

```
# Gaussian Normalization( Z- score)
X_train = (X_train- np.mean(X_train))/np.std(X_train)
X_test = (X_test- np.mean(X_test))/np.std(X_test)
```

Tampilkan jumlah sampel pelatihan yang ada dalam set data dan juga jumlah set pengujian yang tersedia.

```
#Display number of training and test instances
print(X_train.shape[0], 'train samples')
print(X_test.shape[0], 'test samples')
```

Ubah vektor kelas menjadi matriks kelas biner.

```
# convert class vectors to binary class matrices (ie one-hot vectors)
Y_train = np_utils.to_categorical(y_train, nb_classes)
Y_test = np_utils.to_categorical(y_test, nb_classes)
```

Tentukan model sekuensial dari multilayer perceptron.

```
#Define the model achitecture
model = Sequential()
model.add(Dense(512, input_shape=(784,)))
model.add(Activation('relu'))
model.add(Dropout(0.2)) # Regularization
model.add(Dense(120))
model.add(Activation('relu'))
model.add(Dropout(0.2))
model.add(Dense(10)) #Last layer with one output per class
model.add(Activation('softmax')) #We want a score simlar to a probability for each class
```

Gunakan pengoptimal.

```
#Use rmsprop as an optimizer
rms = RMSprop()
```

Fungsi yang dioptimalkan adalah entropi silang antara label sebenarnya dan keluaran

(softmax) model.

```
#The function to optimize is the cross entropy between the true label and the output (softmax) of the model
model.compile(loss='categorical_crossentropy', optimizer=rms, metrics=["accuracy"])
```

Gunakan fungsi model.fit untuk melatih model.

```
#Make the model learn
model.fit(X_train, Y_train,
batch_size=batch_size, nb_epoch=nb_epoch,
verbose=2,
validation_data=(X_test, Y_test))
```

Dengan menggunakan model, evaluasi fungsi untuk mengevaluasi kinerja model.

```
#Evaluate how the model does on the test set
score = model.evaluate(X_test, Y_test, verbose=0)
```

Cetak akurasi yang dihasilkan dalam model.

```
print('Test score:', score[0])
print('Test accuracy:', score[1])
```

Jika Anda menjalankan kode, Anda akan mendapatkan keluaran berikut: 60000 sampel pelatihan

10000 test samples
Train on 60000 samples, validate on 10000 samples
Epoch 1/20
13s - loss: 0.2849 - acc: 0.9132 - val_loss: 0.1149 - val_acc: 0.9652
Epoch 2/20
11s - loss: 0.1299 - acc: 0.9611 - val_loss: 0.0880 - val_acc: 0.9741
Epoch 3/20
11s - loss: 0.0998 - acc: 0.9712 - val_loss: 0.1121 - val_acc: 0.9671
Epoch 4/20
Epoch 18/20
14s - loss: 0.0538 - acc: 0.9886 - val_loss: 0.1241 - val_acc: 0.9814
Epoch 19/20
12s - loss: 0.0522 - acc: 0.9888 - val_loss: 0.1154 - val_acc: 0.9829
Epoch 20/20
13s - loss: 0.0521 - acc: 0.9891 - val_loss: 0.1183 - val_acc: 0.9824
Test score: 0.118255248802
Test accuracy: 0.9824

Sekarang, saatnya membuat kumpulan data dan menggunakan multilayer perceptron.
Di sini Anda akan membuat kumpulan data Anda sendiri menggunakan fungsi acak dan menjalankan model multilayer perceptron pada data yang dihasilkan.

MLP pada Data yang Dihasilkan Secara Acak

Buat file Python baru dan impor paket berikut. Pastikan Anda telah menginstal Keras di sistem Anda.

```
#####MLP on randomly generated Data ##############
import keras
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation
from keras.optimizers import SGD
import numpy as np
```

Hasilkan data menggunakan fungsi acak.

```
# Generate dummy data
x_train = np.random.random((1000, 20))
# Y having 10 possible categories
y_train = keras.utils.to_categorical(np.random.randint(10, size=(1000, 1)), num_classes=10)
x_test = np.random.random((100, 20))
y_test = keras.utils.to_categorical(np.random.randint(10, size=(100, 1)), num_classes=10)
```

Buat model sekuensial.

```
#Create a model
model = Sequential()
# Dense(64) is a fully-connected layer with 64 hidden units.
# In the first layer, you must specify the expected input data shape:
# here, 20-dimensional vectors.
model.add(Dense(64, activation='relu', input_dim=20))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(10, activation='softmax'))
```

Kompilasi model.

```
#Compile the model
sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='categorical_crossentropy',optimizer=sgd,metrics=['accuracy'])
```

Gunakan fungsi model.fit untuk melatih model.

```
# Fit the model
model.fit(x_train, y_train,epochs=20,batch_size=128)
```

Evaluasi kinerja model menggunakan fungsi model.evaluate.

```
# Evaluate the model
score = model.evaluate(x_test, y_test, batch_size=128)
```

Jika Anda menjalankan kode tersebut, Anda akan mendapatkan output berikut:

```
Epoch 1/20
1000/1000 [==============================] - 0s - loss: 2.4432 - acc: 0.0970
Epoch 2/20
1000/1000 [==============================] - 0s - loss: 2.3927 - acc: 0.0850
Epoch 3/20
```

```
1000/1000 [==============================] - 0s - loss: 2.3361 - acc: 0.1190
Epoch 4/20
1000/1000 [==============================] - 0s - loss: 2.3354 - acc: 0.1000
Epoch 19/20
1000/1000 [==============================] - 0s - loss: 2.3034 - acc: 0.1160
Epoch 20/20
1000/1000 [==============================] - 0s - loss: 2.3055 - acc: 0.0980
100/100 [==============================] - 0s
```

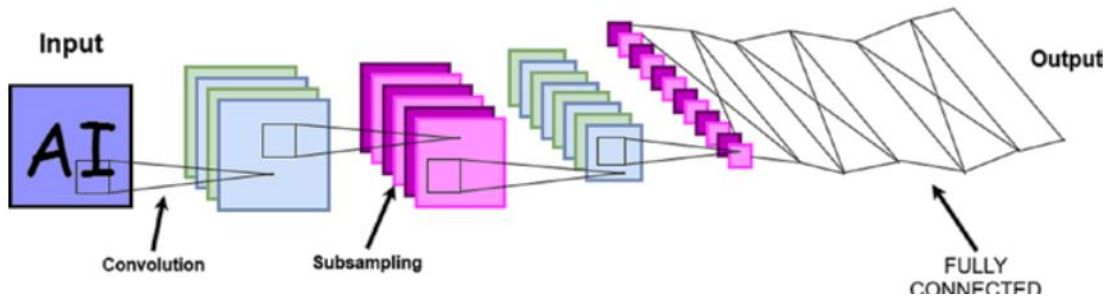
Dalam bab ini, saya membahas cara membangun model linear, logistik, dan MLP di Keras secara sistemik.

BAB 6
JARINGAN SYARAF TIRUAN KONVOLUSIONAL

Jaringan syaraf tiruan konvolusional (CNN) adalah jaringan syaraf tiruan umpan-maju yang mendalam di mana jaringan syaraf mempertahankan struktur hierarkis dengan mempelajari representasi fitur internal dan menggeneralisasi fitur dalam masalah gambar umum seperti pengenalan objek dan masalah visi komputer lainnya. Jaringan ini tidak terbatas pada gambar; jaringan ini juga mencapai hasil mutakhir dalam masalah pemrosesan bahasa alami dan pengenalan suara.

6.1 BERBAGAI LAPISAN DALAM CNN

CNN terdiri dari beberapa lapisan, seperti yang ditunjukkan pada Gambar 6-1.

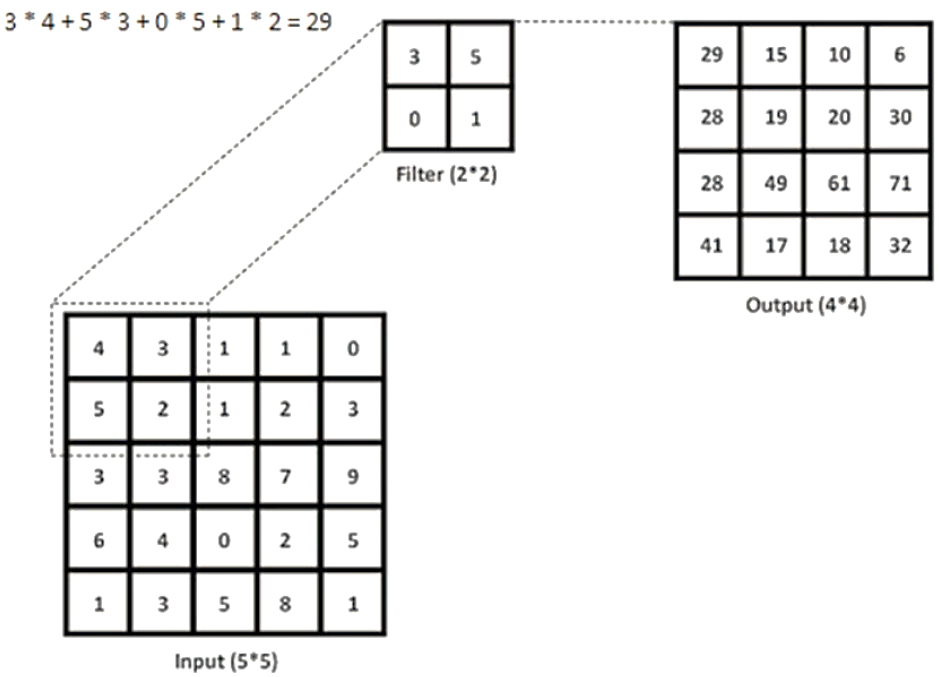


Gambar 6-1. Lapisan dalam jaringan syaraf tiruan konvolusional

Lapisan konvolusi terdiri dari filter dan peta gambar. Anggap gambar masukan skala abu-abu memiliki ukuran 5×5, yang merupakan matriks dengan 25 nilai piksel. Data gambar dinyatakan sebagai matriks tiga dimensi dengan lebar × tinggi × saluran. Peta gambar adalah daftar koordinat yang berkaitan dengan gambar tertentu.

Konvolusi bertujuan untuk mengekstraksi fitur dari gambar masukan, dan karenanya mempertahankan hubungan spasial antara piksel dengan mempelajari fitur gambar menggunakan kotak kecil data masukan. Invariansi rotasi, invariansi translasi, dan invariansi skala dapat diharapkan. Misalnya, gambar kucing yang diputar atau gambar kucing yang diskalakan ulang dapat dengan mudah diidentifikasi oleh CNN karena langkah konvolusi.

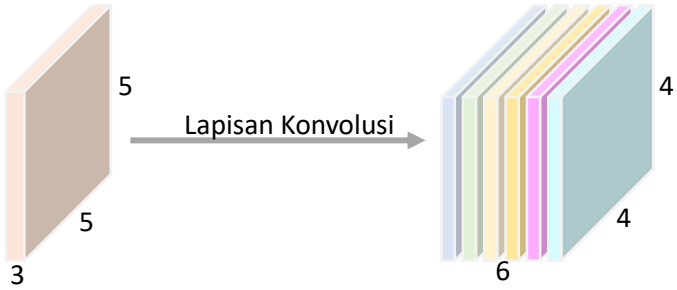
Anda menggeser filter (matriks persegi) di atas gambar asli Anda (di sini, 1 piksel), dan pada setiap posisi yang diberikan, Anda menghitung perkalian per elemen (antara matriks filter dan gambar asli) dan menambahkan keluaran perkalian untuk mendapatkan bilangan bulat akhir yang membentuk elemen matriks keluaran. Subsampling hanyalah pengumpulan rata-rata dengan bobot yang dapat dipelajari per peta fitur, seperti yang ditunjukkan pada Gambar 6-2.



Gambar 6-2. Subsampling

Seperti yang ditunjukkan pada Gambar 6-2, filter memiliki bobot input dan menghasilkan neuron output. Katakanlah Anda mendefinisikan lapisan konvolusional dengan enam filter dan bidang reseptif yang lebarnya 2 piksel dan tingginya 2 piksel dan menggunakan lebar langkah default 1, dan padding default ditetapkan ke 0. Setiap filter menerima input dari bagian gambar berukuran 2x2 piksel. Dengan kata lain, itu berarti 4 piksel pada satu waktu. Oleh karena itu, Anda dapat mengatakan bahwa diperlukan bobot input 4 + 1 (bias).

Volume input adalah 5x5x3 (lebar × tinggi × jumlah saluran), ada enam filter berukuran 2x2 dengan langkah 1 dan pad 0. Oleh karena itu, jumlah parameter dalam lapisan ini untuk setiap filter adalah 2x2x3 + 1 = 13 parameter (ditambahkan +1 untuk bias). Karena ada enam filter, Anda memperoleh 13x6 = 78 parameter.



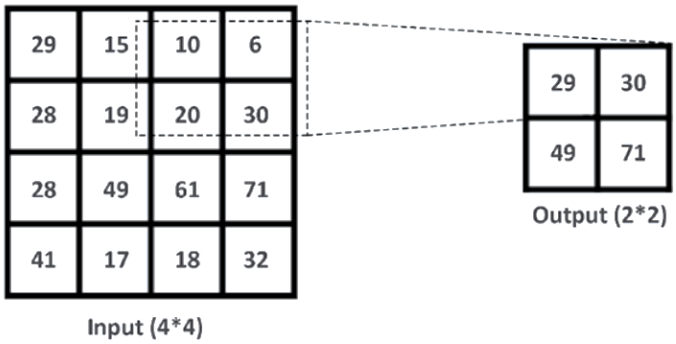
Gambar 6-3. Volume masukan

Berikut ringkasannya:

- Volume masukan berukuran W1 × H1 × D1.
- Model memerlukan hiperparameter: jumlah filter (f), langkah (S), jumlah zero padding (P).

- Ini menghasilkan volume berukuran W2 × H2 × D2.
- W2 = (W1-f+ 2P) /S + 1 = 4.
- H2 = (H1-f+2P)/S +1 = 4.
- D2 = Jumlah filter = f = 6.

Lapisan penggabungan mengurangi peta aktivasi lapisan sebelumnya. Lapisan ini diikuti oleh satu atau beberapa lapisan konvolusional dan menggabungkan semua fitur yang dipelajari dalam peta aktivasi lapisan sebelumnya. Ini mengurangi overfitting data pelatihan dan menggeneralisasi fitur yang direpresentasikan oleh jaringan. Ukuran bidang reseptif hampir selalu ditetapkan ke 2×2 dan menggunakan langkah 1 atau 2 (atau lebih tinggi) untuk memastikan tidak ada tumpang tindih. Anda akan menggunakan operasi maks untuk setiap bidang reseptif sehingga aktivasi adalah nilai masukan maksimum. Di sini, setiap empat angka dipetakan ke hanya satu angka. Jadi, jumlah piksel turun menjadi seperempat dari aslinya dalam langkah ini (Gambar 6-4).

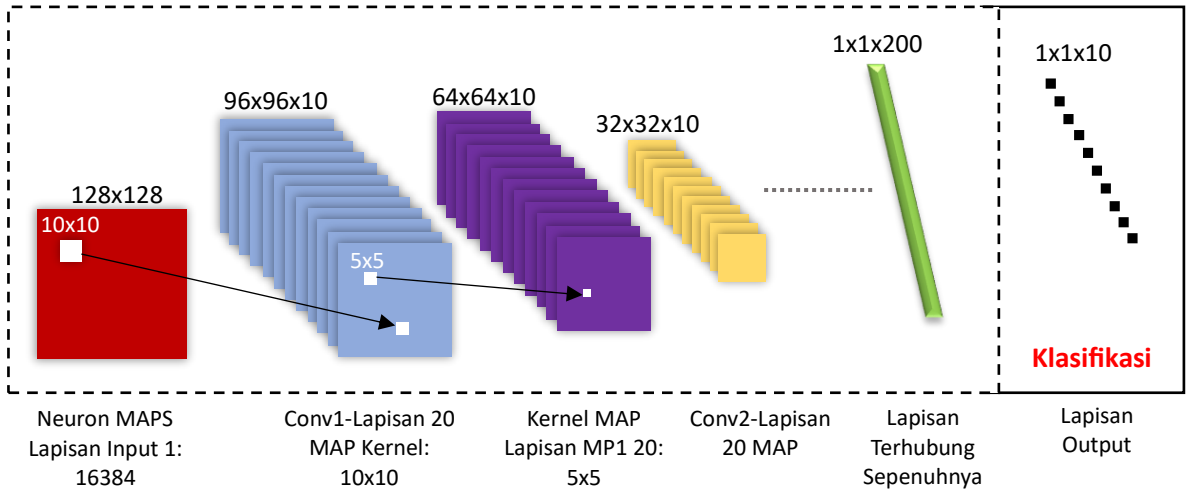


Gambar 6-4. Maxpooling-mengurangi jumlah piksel

Lapisan yang terhubung penuh adalah lapisan jaringan saraf tiruan umpan-maju. Lapisan-lapisan ini memiliki fungsi aktivasi nonlinier untuk menghasilkan probabilitas prediksi kelas. Lapisan-lapisan ini digunakan menjelang akhir setelah semua fitur diidentifikasi dan diekstraksi oleh lapisan konvolusional dan telah dikonsolidasikan oleh lapisan pengumpulan dalam jaringan. Di sini, lapisan tersembunyi dan lapisan keluaran adalah lapisan yang terhubung penuh.

6.2 ARSITEKTUR CNN

CNN adalah arsitektur jaringan saraf dalam umpan-maju yang terdiri dari beberapa lapisan konvolusional, masing-masing diikuti oleh lapisan pengumpulan, fungsi aktivasi, dan normalisasi batch opsional. CNN juga terdiri dari lapisan-lapisan yang terhubung penuh. Saat gambar bergerak melalui jaringan, gambar tersebut menjadi lebih kecil, sebagian besar karena pengumpulan maksimum. Lapisan terakhir menghasilkan prediksi probabilitas kelas.



Gambar 6-5. Arsitektur CNN untuk Klasifikasi

Beberapa tahun terakhir telah banyak arsitektur yang dikembangkan yang telah membuat kemajuan luar biasa dalam bidang klasifikasi gambar.

Jaringan pra-latihan pemenang penghargaan (VGG16, VGG19, ResNet50, Inception V3, dan Xception) telah digunakan untuk berbagai tantangan klasifikasi gambar termasuk pencitraan medis. Pembelajaran transfer adalah jenis praktik di mana Anda menggunakan model pra-latihan sebagai tambahan beberapa lapisan. Ini dapat digunakan untuk memecahkan tantangan klasifikasi gambar di setiap bidang.

BAB 7
CNN DALAM TENSORFLOW

Bab ini akan menunjukkan cara menggunakan TensorFlow untuk membangun model CNN. Model CNN dapat membantu Anda membangun pengklasifikasi gambar yang dapat memprediksi/mengklasifikasi gambar. Secara umum, Anda membuat beberapa lapisan dalam arsitektur model dengan nilai awal bobot dan bias. Kemudian Anda menyetel bobot dan bias dengan bantuan set data pelatihan.

Ada pendekatan lain yang melibatkan penggunaan model yang telah dilatih sebelumnya seperti InceptionV3 untuk mengklasifikasikan gambar. Anda dapat menggunakan pendekatan pembelajaran transfer ini di mana Anda menambahkan beberapa lapisan (yang parameternya telah dilatih) di atas lapisan model yang telah dilatih sebelumnya (dengan nilai parameter utuh) untuk membuat pengklasifikasi yang sangat kuat.

Dalam bab ini, penulis akan menggunakan TensorFlow untuk menunjukkan cara mengembangkan jaringan konvolusi untuk berbagai aplikasi visi komputer. Lebih mudah untuk mengekspresikan arsitektur CNN sebagai grafik aliran data.

7.1 TENSORFLOW PADA MODEL CNN

Dalam TensorFlow, gambar dapat direpresentasikan sebagai array tiga dimensi atau tensor bentuk (tinggi, lebar, dan saluran). TensorFlow memberikan fleksibilitas untuk melakukan iterasi dengan cepat, memungkinkan Anda melatih model dengan lebih cepat, dan memungkinkan Anda menjalankan lebih banyak eksperimen. Saat membawa model TensorFlow ke tahap produksi, Anda dapat menjalankannya pada GPU dan TPU berskala besar.

Kode TensorFlow untuk Membangun Pengklasifikasi Gambar untuk Data MNIST

Di bagian ini, penulis akan memberikan contoh untuk memahami cara mengimplementasikan CNN di TensorFlow. Kode berikut mengimpor set data MNIST dengan gambar skala abu-abu 28×28 digit dari paket kontribusi TensorFlow dan memuat semua pustaka yang diperlukan. Di sini, tujuannya adalah untuk membangun pengklasifikasi guna memprediksi digit yang diberikan dalam gambar.

```
from tensorflow.contrib.learn.python.learn.datasets.mnist
import read_data_sets
from tensorflow.python.framework import ops
import tensorflow as tf
import numpy as np
```

Anda kemudian memulai sesi grafik.

```
# Start a graph session
sess = tf.Session()
```

Anda memuat data MNIST dan membuat set pelatihan dan pengujian.

```
# Load data
```

```
from keras.datasets import mnist
(X_train, y_train), (X_test, y_test) = mnist.load_data()
```

Anda kemudian menormalkan fitur rangkaian kereta dan pengujian.

```
# Z- score or Gaussian Normalization
X_train = X_train - np.mean(X_train) / X_train.std()
X_test = X_test - np.mean(X_test) / X_test.std()
```

Karena ini adalah masalah klasifikasi multikelas, maka akan selalu lebih baik untuk menggunakan pengodean one-hot dari nilai kelas keluaran.

```
# Convert labels into one-hot encoded vectors
num_class = 10
train_labels = tf.one_hot(y_train, num_class)
test_labels = tf.one_hot(y_test, num_class)
```

Mari kita tetapkan parameter model sekarang karena gambar-gambar ini berskala abu-abu. Oleh karena itu, kedalaman gambar (saluran) adalah 1.

```
# Set model parameters
batch_size = 784
samples =500
learning_rate = 0.03
img_width = X_train[0].shape[0]
img_height = X_train[0].shape[1]
target_size = max(train_labels) + 1
num_channels = 1 # greyscale = 1 channel
epoch = 200
no_channels = 1
conv1_features = 30
filt1_features = 5
conv2_features = 15
filt2_features = 3
max_pool_size1 = 2 # NxN window for 1st max pool layer
max_pool_size2 = 2 # NxN window for 2nd max pool layer
fully_connected_size1 = 150
```

Mari kita nyatakan placeholder untuk model tersebut. Fitur data input, variabel target, dan ukuran batch dapat diubah untuk set pelatihan dan evaluasi.

```
# Declare model placeholders
x_input_shape = (batch_size, img_width, img_height, num_channels)
x_input = tf.placeholder(tf.float32, shape=x_input_shape)
y_target = tf.placeholder(tf.int32, shape=(batch_size))
eval_input_shape = (samples, img_width, img_height, num_channels)
eval_input = tf.placeholder(tf.float32, shape=eval_input_shape)
eval_target = tf.placeholder(tf.int32, shape=(samples))
```

Mari kita nyatakan nilai bobot dan bias variabel model untuk neuron input dan lapisan tersembunyi.

```
# Declare model variables
W1 = tf.Variable(tf.random_normal([filt1_features, filt1_features,
```

```
no_channels, conv1_features]))
b1 = tf.Variable(tf.ones([conv1_features]))
W2 = tf.Variable(tf.random_normal([filt2_features, filt2_features,
conv1_features, conv2_features]))
b2 = tf.Variable(tf.ones([conv2_features]))
```

Mari kita nyatakan variabel model untuk lapisan yang terhubung sepenuhnya dan tentukan bobot dan bias untuk 2 lapisan terakhir ini.

```
# Declare model variables for fully connected layers
resulting_width = img_width // (max_pool_size1 * max_pool_size2)
resulting_height = img_height // (max_pool_size1 * max_pool_size2)
full1_input_size = resulting_width * resulting_height * conv2_features
W3 = tf.Variable(tf.truncated_normal([full1_input_size,
fully_connected_size1], stddev=0.1, dtype=tf.float32))
b3 = tf.Variable(tf.truncated_normal([fully_connected_size1], stddev=0.1,
dtype=tf.float32))
W_out = tf.Variable(tf.truncated_normal([fully_connected_size1,
target_size], stddev=0.1, dtype=tf.float32))
b_out = tf.Variable(tf.truncated_normal([target_size], stddev=0.1,
dtype=tf.float32))
```

Mari membuat fungsi pembantu untuk menentukan lapisan konvolusional dan pengumpulan maksimal.

```
# Define helper functions for the convolution and maxpool layers:
def conv_layer(x, W, b):
    conv = tf.nn.conv2d(x, W, strides=[1, 1, 1, 1], padding='SAME')
    conv_with_b = tf.nn.bias_add(conv, b)
    conv_out = tf.nn.relu(conv_with_b)
    return conv_out
def maxpool_layer(conv, k=2):
    return tf.nn.max_pool(conv, ksize=[1, k, k, 1], strides=[1, k, k, 1],
    padding='SAME')
```

Model jaringan saraf didefinisikan dengan dua lapisan tersembunyi dan dua lapisan yang terhubung penuh. Unit linier yang diperbaiki digunakan sebagai fungsi aktivasi untuk lapisan tersembunyi dan lapisan keluaran akhir.

```
# Initialize Model Operations
def my_conv_net(input_data):
     # First Conv-ReLU-MaxPool Layer
     conv_out1 = conv_layer(input_data, W1, b1)
     maxpool_out1 = max
# Second Conv-ReLU-MaxPool Layer
 conv_out2 = conv_layer(maxpool_out1, W2, b2)
 maxpool_out2 = maxpool_layer(conv_out2)
 # Transform Output into a 1xN layer for next fully connected layer
 final_conv_shape = maxpool_out2.get_shape().as_list()
 final_shape=final_conv_shape[1]*final_conv_shape[2]*final_conv_shape[3]
 flat_output = tf.reshape(maxpool_out2, [final_conv_shape[0],final_shape])
 # First Fully Connected Layer
```

```
    fully_connected1 = tf.nn.relu(tf.add(tf.matmul(flat_output, W3), b3))
    # Second Fully Connected Layer
    final_model_output = tf.add(tf.matmul(fully_connected1, W_out), b_out)
    return(final_model_output)
  model_output = my_conv_net(x_input)
  test_model_output = my_conv_net(eval_input)
```

Anda akan menggunakan fungsi entropi silang softmax (cenderung bekerja lebih baik untuk klasifikasi multikelas) untuk menentukan kerugian yang beroperasi pada logit.

```
  # Declare Loss Function (softmax cross entropy)
  loss = tf.reduce_mean(tf.nn.sparse_softmax_cross_entropy_with_
  logits(logits=model_output, labels=y_target))
```

Mari kita definisikan fungsi prediksi himpunan kereta dan himpunan pengujian.

```
  # Create a prediction function
  prediction = tf.nn.softmax(model_output)
  test_prediction = tf.nn.softmax(test_model_output)
```

Untuk menentukan akurasi model pada setiap batch, mari kita definisikan fungsi akurasi.

```
  # Create accuracy function
  def get_accuracy(logits, targets):
     batch_predictions = np.argmax(logits, axis=1)
     num_correct = np.sum(np.equal(batch_predictions, targets))
     return(100. * num_correct/batch_predictions.shape[0])
```

Mari kita nyatakan langkah pelatihan dan tentukan fungsi pengoptimal.

```
  # Create an optimizer
  my_optimizer = tf.train.AdamOptimizer(learning_rate, 0.9)
  train_step = my_optimizer.minimize(loss)
```

Mari kita inisialisasi semua variabel model yang dideklarasikan sebelumnya.

```
  # Initialize Variables
  varInit = tf.global_variables_initializer()
  sess.run(varInit)
```

Mari kita mulai melatih model dan melakukan pengulangan secara acak melalui kumpulan data. Anda ingin mengevaluasi model pada kumpulan set pelatihan dan pengujian serta mencatat kerugian dan akurasi.

```
  # Start training loop
  train_loss = []
  train_acc = []
  test_acc = []
  for i in range(epoch):
      random_index = np.random.choice(len(X_train), size=batch_size)
      random_x = X_train[random_index]
      random_x = np.expand_dims(random_x, 3)
      random_y = train_labels[random_index]
      train_dict = {x_input: random_x, y_target: random_y}
  sess.run(train_step, feed_dict=train_dict)
```

```
 temp_train_loss, temp_train_preds = sess.run([loss,
prediction], feed_dict=train_dict)
   temp_train_acc = get_accuracy(temp_train_preds, random_y)
   eval_index = np.random.choice(len(X_test), size=evaluation_size)
   eval_x = X_test[eval_index]
   eval_x = np.expand_dims(eval_x, 3)
   eval_y = test_labels[eval_index]
   test_dict = {eval_input: eval_x, eval_target: eval_y}
   test_preds = sess.run(test_prediction, feed_dict=test_dict)
   temp_test_acc = get_accuracy(test_preds, eval_y)
```

Hasil model dicatat dalam format berikut dan dicetak dalam output:

```
# Record and print results
train_loss.append(temp_train_loss)
train_acc.append(temp_train_acc)
test_acc.append(temp_test_acc)
print('Epoch # {}. Train Loss: {:.2f}. Train Acc : {:.2f} .
temp_test_acc : {:.2f}'.format(i+1,temp_train_loss,
temp_train_acc,temp_test_acc))
```

7.2 MENGGUNAKAN API DALAM MEMBANGUN MODEL CNN

TFLearn, TensorLayer, tflayers, TF-Slim, tf.contrib.learn, Pretty Tensor, keras, dan Sonnet adalah API TensorFlow tingkat tinggi. Jika Anda menggunakan salah satu API tingkat tinggi ini, Anda dapat membangun model CNN dalam beberapa baris kode. Jadi, Anda dapat menjelajahi salah satu API ini untuk bekerja dengan cerdas.

BAB 8
CNN DI KERAS

Bab ini akan menunjukkan cara menggunakan Keras untuk membangun model CNN. Model CNN dapat membantu Anda membangun pengklasifikasi gambar yang dapat memprediksi dan mengklasifikasikan gambar. Secara umum, Anda membuat beberapa lapisan dalam arsitektur model dengan nilai awal bobot dan bias. Kemudian Anda menyetel variabel bobot dan bias dengan bantuan set data pelatihan. Anda akan mempelajari cara membuat kode di Keras dalam konteks ini. Ada pendekatan lain yang melibatkan penggunaan model yang telah dilatih sebelumnya seperti InceptionV3 dan ResNet50 yang dapat mengklasifikasikan gambar.

Mari kita definisikan model CNN dan mengevaluasi seberapa baik kinerjanya. Anda akan menggunakan struktur dengan lapisan konvolusional; kemudian Anda akan menggunakan pengumpulan maksimum dan meratakan jaringan untuk menghubungkan lapisan sepenuhnya dan membuat prediksi.

8.1 MEMBANGUN PENGKLASIFIKASI GAMBAR UNTUK DATA MNIST DI KERAS

Di sini saya akan menunjukkan proses membangun pengklasifikasi untuk digit tulisan tangan menggunakan set data MNIST yang populer.

Tugas ini merupakan tantangan besar untuk bermain dengan jaringan saraf, tetapi dapat dikelola pada satu komputer. Basis data MNIST berisi 60.000 gambar pelatihan dan 10.000 gambar pengujian.

Mulailah dengan mengimpor Numpy dan menetapkan benih untuk generator angka pseudorandom komputer. Ini memungkinkan Anda untuk mereproduksi hasil dari skrip Anda.

```
import numpy as np
# random seed for reproducibility
np.random.seed(123)
```

Selanjutnya, Anda mengimpor tipe model sekuensial dari Keras. Ini hanyalah tumpukan linier dari lapisan jaringan saraf.

```
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import Flatten
from keras.layers import Conv2D
from keras.layers import MaxPooling2d
#Now we will import some utilities
from keras.utils import np_utils
#Fixed dimension ordering issue
from keras import backend as K
K.set_image_dim_ordering('th')
```

```
#Load image data from MNIST
#Load pre-shuffled MNIST data into train and test sets
(X_train,y_train),(X_test, y_test)=mnist.load_data()

#Preprocess imput data for Keras
# Reshape input data.
# reshape to be [samples][channels][width][height]
X_train=X_train.reshape(X_train.shape[0],1,28,28)
X_test=X_test.reshape(X_test.shape[0],1,28,28)

# to convert our data type to float32 and normalize our database
X_train=X_train.astype('float32')
X_test=X_test.astype('float32')
print(X_train.shape)

# Z-scoring or Gaussian Normalization
X_train=X_train - np.mean(X_train) / X_train.std()
X_test=X_test - np.mean(X_test) / X_test.std()
#(60000, 1, 28, 28)

# convert 1-dim class arrays to 10 dim class metrices
#one hot encoding outputs
y_train=np_utils.to_categorical(y_train)
y_test-np_utils.to_categorical(y_test)
num_classes=y_test.shape[1]
print(num_classes)
#10

#Define a simple CNN model
print(X_train.shape)
#(60000,1,28,28)
```

Tentukan Struktur Jaringan

Struktur jaringan adalah sebagai berikut:

- Jaringan memiliki lapisan masukan konvolusional, dengan 32 peta fitur berukuran 5×5. Fungsi aktivasi adalah unit linier yang diperbaiki.
- Lapisan kumpulan maksimum berukuran 2×2.
- Dropout ditetapkan menjadi 30 persen.
- Anda dapat meratakan lapisan tersebut.
- Jaringan memiliki lapisan yang terhubung penuh dengan 240 unit, dan fungsi aktivasi adalah unit linier eksponensial.
- Lapisan terakhir jaringan adalah lapisan keluaran yang terhubung penuh dengan sepuluh unit, dan fungsi aktivasi adalah softmax.

Kemudian Anda mengompilasi model dengan menggunakan entropi silang biner sebagai fungsi kerugian dan adagrad sebagai pengoptimal.

Menentukan Arsitektur Model

Arsitektur terdiri dari kombinasi lapisan konvolusional dan lapisan pengumpulan maksimum serta lapisan padat di bagian akhir.

```
# create a model
    model=Sequential()
    model.add(Conv2D(32, (5,5), input_shape=(1,28,28), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2,2)))
    model.add(Dropout(0.3)) # Dropout, one form of regularization
    model.add(Flatten())
    model.add(Dense(240,activation='elu'))
    model.add(Dense(num_classes, activation='softmax'))
    print(model.output_shape)
    (None, 10)

# Compile the model
model.compile(loss='binary_crossentropy', optimizer='adagrad',
matrices=['accuracy'])
```

Kemudian Anda menyesuaikan model dengan menggunakan set data pelatihan dengan mengambil ukuran batch sebesar 200. Model mengambil 200 contoh/baris pertama (dari yang ke-1 hingga ke-200) dari set data pelatihan dan melatih jaringan. Kemudian model mengambil 200 contoh kedua (dari yang ke-201 hingga ke-400) untuk jaringan pelatihan lagi. Dengan cara ini, Anda menyebarkan semua contoh melalui jaringan. Model membutuhkan lebih sedikit memori saat Anda melatih jaringan dengan lebih sedikit contoh setiap kali. Namun, ukuran batch yang kecil tidak menawarkan estimasi gradien yang baik, dan karenanya penyetelan bobot dan bias dapat menjadi tantangan.

Satu epoch berarti satu lintasan maju dan satu lintasan mundur dari semua contoh pelatihan. Diperlukan beberapa iterasi untuk menyelesaikan satu epoch. Di sini, Anda memiliki 60.000 contoh pelatihan, dan ukuran batch Anda adalah 200, jadi diperlukan 300 iterasi untuk menyelesaikan 1 epoch.

```
# Fit the model
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=1,
batch_size=200)

# Evaluate model on test data
    # Final evaluation of the model
    scores =model.evaluate(X_test, y_test, verbose=0)
    print("CNN error: % .2f%%" % (100-scores[1]*100))
    # CNN Error: 17.98%
    # Save the model
    # save model
    model_json= model.to_join()
    with open("model_json", "w") as json_file: json_file.write(model_json)
    # serialize weights to HDFS
    model.save_weights("model.h5")
```

8.2 MEMBANGUN PENGKLASIFIKASI GAMBAR DENGAN DATA CIFAR-10

Bagian ini menjelaskan cara membangun pengklasifikasi yang dapat mengklasifikasikan sepuluh label set data CIFAR-10 menggunakan model Keras CNN.

Catatan Set data CIFAR-10 terdiri dari 60.000 gambar berwarna 32×32 dalam 10 kelas, dengan 6.000 gambar per kelas. Ada 50.000 gambar pelatihan dan 10.000 gambar uji.

```
############Building CNN Model with CIFAR10 data####################
# plot cifar10 instances
   from keras.datasets import cifar10
   from matplotlib import pyplot
   from scipy.misc import toimage
   import numpy
   from keras.models import Sequential
   from keras.layers import Dense
   from keras.layers import Dropout
   from keras.layers import Flatten
   from keras.layers import Conv2D
   from keras.layers import MaxPooling2d
   #Now we will import some utilities
   from keras.utils import np_utils
   from keras.layers.normalization import BatchNormalization
   #Fixed dimension ordering issue
   from keras import backend as K
   K.set_image_dim_ordering('th')
   # fix random seed for reproducibility
   seed=12
   numpy.random.seed(seed)
   # Preprocess imput data for Keras
   # Reshape input data.
   # reshape to be [samples][channels][width][height]
   X_train=X_train.reshape(X_train.shape[0],3,32,32). astype('float32')
   X_test=X_test.reshape(X_test.shape[0],3,32,32). astype('float32')
   # Z-scoring or Gaussian Normalization
   X_train=X_train - np.mean(X_train) / X_train.std()
   X_test=X_test - np.mean(X_test) / X_test.std()
   # convert 1-dim class arrays to 10 dim class metrices
   # one hot encoding outputs
   y_train=np_utils.to_categorical(y_train)
   y_test-np_utils.to_categorical(y_test)
   num_classes=y_test.shape[1]
   print(num_classes)
   #10

  Define a simple CNN model
   print(X_train.shape)
   #(50000,3,32,32)
```

Tentukan Struktur Jaringan

Struktur jaringan adalah sebagai berikut:

- Lapisan masukan konvolusional memiliki 32 peta fitur dengan ukuran 5×5, dan fungsi aktivasi adalah unit linier yang diperbaiki.
- Lapisan kumpulan maksimum memiliki ukuran 2×2.

- Lapisan konvolusional memiliki 32 peta fitur dengan ukuran 5×5, dan fungsi aktivasi adalah unit linier yang diperbaiki.
- Jaringan memiliki normalisasi batch.
- Lapisan kumpulan maksimum memiliki ukuran 2×2.
- Dropout ditetapkan menjadi 30 persen.
- Anda dapat meratakan lapisan.
- Lapisan yang terhubung penuh memiliki 240 unit, dan fungsi aktivasi adalah unit linier eksponensial.
- Lapisan keluaran yang terhubung penuh memiliki sepuluh unit, dan fungsi aktivasi adalah softmax.

Kemudian Anda menyesuaikan model dengan menggunakan set data pelatihan dengan mengambil ukuran batch sebesar 200. Anda mengambil 200 instans/baris pertama (dari yang ke-1 hingga ke-200) dari set data pelatihan dan melatih jaringan. Kemudian Anda mengambil 200 instans kedua (dari yang ke-201 hingga ke-400) untuk melatih jaringan lagi. Dengan cara ini, Anda menyebarkan semua instans melalui jaringan. Satu epoch berarti satu lintasan maju dan satu lintasan mundur dari semua contoh pelatihan. Diperlukan beberapa iterasi untuk menyelesaikan satu epoch.

Di sini, Anda memiliki 50.000 contoh pelatihan, dan ukuran batch Anda adalah 200, jadi diperlukan 250 iterasi untuk menyelesaikan 1 epoch.

Menentukan Arsitektur Model

Model sekuensial dibuat dengan kombinasi lapisan konvolusional dan max pooling. Kemudian lapisan padat yang terhubung penuh dipasang.

```
# create a model
 model=Sequential()
 model.add(Conv2D(32, (5,5), input_shape=(3,32,32), activation='relu'))
 model.add(MaxPooling2D(pool_size=(2,2)))
 model.add(Conv2D(32, (5,5), activation='relu', padding='same'))
 model.add(BatchNormalization())
 model.add(MaxPooling2D(pool_size=(2,2)))
 model.add(Dropout(0.3)) # Dropout, one form of regularization
 model.add(Flatten())
 model.add(Dense(240,activation='elu'))
 model.add(Dense(num_classes, activation='softmax'))
 print(model.output_shape)
 model.compile(loss='binary_crossentropy', optimizer='adagrad')
 # fit model model.fit(X_train,y_train,validation_data=(X_test,y_test),epochs=1,
batch_size=200)
 # Final evaluation of the model
 scores=model.evaluate(X_test, y_test, verbose=0)
 print("CNN error: % .2f%%" % (100-score
```

8.3 MODEL YANG DILATIH SEBELUMNYA

Pada bagian ini, saya akan menunjukkan cara menggunakan model yang dilatih sebelumnya seperti VGG dan inception untuk membangun pengklasifikasi.

```
from keras import applications
from keras, models import Sequential, Model
from keras.applications.vgg16 import VGG16
from keras.applications.vgg16 import preprocess_input, decode predictions
from keras.models import Model
model = VGG16(weights='imagenet', include_top=True)
model. summary ()

#predicting for any new image based on the pre-trained model
# Loading Image
img = image.load_img('('./Data/horse.jpg', target_size=(224, 224)
img = image.img_to_array (img)
img = np.expand_dims (img, axis=0)
img=preprocess_input (img)

# Predict the output
preds = model.predict (img)

# decode the predictions
pred_class = decode_predictions (preds, top=3) [0] [0]
print ("Predicted Class: %s" %pred_class [1] )
print ("Confidence ("Confidance: %s"% pred_class [2])
#Predicted Class: hartebeest
#Confidence: 0.964784
ResNet50 and InceptionV3 models can be easily utilized for
prediction/classification of new images.
from keras.applications import ResNet50
model = ResNet50 (weights='imagenet' , include_top=True)
model. summary ()

# create the base pre-trained model
from keras.applications import InceptionV3
model = InceptionV3(weights='imagenet')
model. summary
```

Model pra-latihan Inception-V3 dapat mendeteksi/mengklasifikasikan objek dari 22.000 kategori. Model ini dapat mendeteksi/mengklasifikasikan baki, obor, payung, dan lainnya.

Dalam banyak skenario, kita perlu membuat pengklasifikasi sesuai kebutuhan kita. Untuk itu, pembelajaran transfer digunakan di mana kita menggunakan model pra-latihan (digunakan untuk ekstraksi fitur) dan beberapa neural.

BAB 9
RNN DAN LSTM

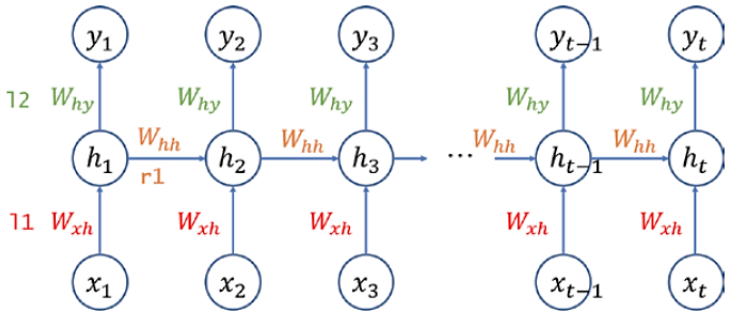
Bab ini akan membahas konsep jaringan saraf berulang (*Recurrent Neural Network*/RNN) dan versi modifikasinya, memori jangka pendek panjang (*Long Short-Term Memory*/LSTM). LSTM terutama digunakan untuk prediksi sekuens. Anda akan mempelajari tentang berbagai prediksi sekuens dan kemudian mempelajari cara melakukan peramalan deret waktu dengan bantuan model LSTM.

9.1 KONSEP RNN

Jaringan saraf berulang adalah jenis jaringan saraf buatan yang paling cocok untuk mengenali pola dalam sekuens data, seperti teks, video, ucapan, bahasa, genom, dan data deret waktu. RNN adalah algoritme yang sangat canggih yang dapat mengklasifikasikan, mengelompokkan, dan membuat prediksi tentang data, khususnya deret waktu dan teks.

RNN dapat dilihat sebagai jaringan MLP dengan penambahan loop ke arsitekturnya. Pada Gambar 9-1, Anda dapat melihat bahwa ada lapisan input (dengan node seperti x_1, x_2, dan seterusnya), lapisan tersembunyi (dengan node seperti h_1, h_2, dan seterusnya), dan lapisan output (dengan node seperti y_1, y_2, dan seterusnya).

Hal ini mirip dengan arsitektur MLP. Perbedaannya adalah bahwa node dari lapisan tersembunyi saling terhubung. Dalam RNN/LSTM vanilla (dasar), node terhubung dalam satu arah. Ini berarti bahwa h_2 bergantung pada h_1 (dan x_2), dan h_3 bergantung pada h_2 (dan x_3). Node dalam lapisan tersembunyi ditentukan oleh node sebelumnya dalam lapisan tersembunyi.



$$h_t = \tanh(\, l1(x_t) + r1(h_{t-1})\,)$$
$$y_t = l2(h_t)$$

Gambar 9-1. RNN

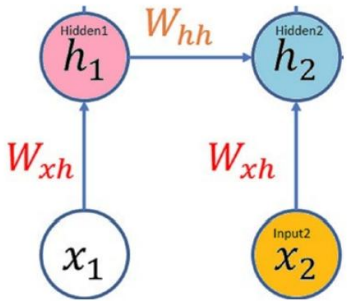
Arsitektur semacam ini memastikan bahwa output pada $t = n$ bergantung pada input pada $t = n$, $t = n - 1$, …, dan $t = 1$. Dengan kata lain, output bergantung pada urutan data, bukan pada satu bagian data (Gambar 9-2).

```
(Input1) → Output1

(Input2, Input1) → Output2

(Input3, Input2, Input1) →  Output3

(Input4, Input3, Input2, Input1) → Output4
```

Gambar 9-2. Urutan

Gambar 9-3 memperlihatkan bagaimana simpul-simpul lapisan tersembunyi dihubungkan ke simpul-simpul lapisan masukan.

```
(Input1) → Hidden1

(Input2, Hidden1) → Hidden2

(Input3, Hidden2) → Hidden3

(Input4, Hidden3) → Hidden4
```



Gambar 9-3. Koneksi

Dalam RNN, jika urutannya cukup panjang, gradien (yang penting untuk menyetel bobot dan bias) dihitung selama pelatihannya (backpropagation). Gradien tersebut akan hilang (perkalian banyak nilai kecil kurang dari 1) atau meledak (perkalian banyak nilai besar lebih dari 1), yang menyebabkan model dilatih dengan sangat lambat.

9.2 KONSEP LSTM

Memori jangka panjang dan pendek adalah arsitektur RNN yang dimodifikasi yang mengatasi masalah gradien yang menghilang dan meledak serta mengatasi masalah pelatihan melalui rangkaian panjang dan mempertahankan memori. Semua RNN memiliki loop umpan balik di lapisan berulang. Loop umpan balik membantu menyimpan informasi dalam "memori" dari waktu ke waktu. Namun, mungkin sulit untuk melatih RNN standar guna memecahkan masalah yang memerlukan pembelajaran ketergantungan temporal jangka panjang. Karena gradien fungsi kerugian menurun secara eksponensial seiring waktu (fenomena yang dikenal sebagai masalah gradien yang menghilang), sulit untuk melatih RNN yang umum. Itulah sebabnya RNN dimodifikasi sedemikian rupa sehingga mencakup sel memori yang dapat menyimpan informasi dalam memori untuk jangka waktu yang lama.

RNN yang dimodifikasi lebih dikenal sebagai LSTM. Dalam LSTM, serangkaian gerbang digunakan untuk mengontrol kapan informasi memasuki memori, yang memecahkan masalah gradien yang menghilang atau meledak. Koneksi berulang menambahkan status atau memori ke jaringan dan memungkinkannya mempelajari dan memanfaatkan sifat pengamatan yang terurut dalam urutan masukan. Memori internal berarti keluaran jaringan bergantung pada konteks terkini dalam urutan masukan, bukan apa yang baru saja disajikan sebagai masukan ke jaringan.

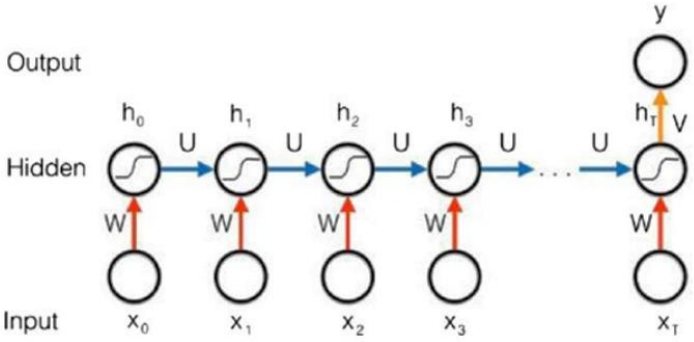
Mode LSTM

LSTM dapat memiliki salah satu mode berikut:

- Model satu-ke-satu
- Model satu-ke-banyak
- Model banyak-ke-satu
- Model banyak-ke-banyak

Selain mode-mode ini, model banyak-ke-banyak yang disinkronkan juga digunakan, terutama untuk klasifikasi video.

Gambar 9-4 menunjukkan LSTM banyak-ke-satu. Ini menyiratkan bahwa banyak masukan menghasilkan satu keluaran dalam model ini.



Gambar 9-4. LSTM Banyak-ke-satu

Prediksi Urutan

LSTM paling cocok untuk data urutan. LSTM dapat memprediksi, mengklasifikasikan, dan menghasilkan data urutan. Urutan berarti urutan pengamatan, bukan sekumpulan pengamatan. Contoh urutan adalah rangkaian uji yang stempel waktu dan nilainya berada dalam urutan (kronologis) urutan. Contoh lain adalah video, yang dapat dianggap sebagai urutan gambar atau urutan klip audio.

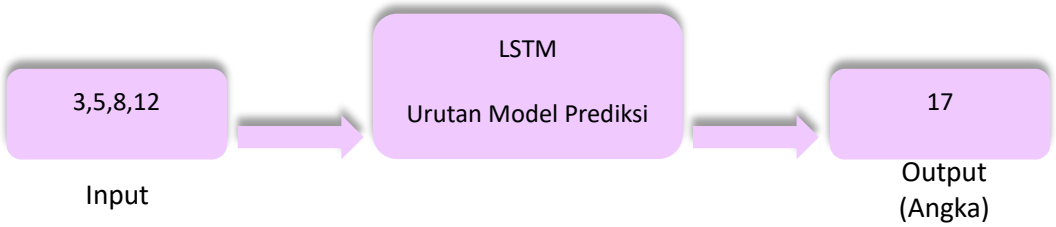
Prediksi berdasarkan urutan data disebut prediksi urutan. Prediksi urutan dikatakan memiliki empat jenis.

- Prediksi numerik urutan
- Klasifikasi urutan
- Pembuatan urutan
- Prediksi urutan-ke-urutan

Prediksi Numerik Urutan

Prediksi numerik urutan adalah memprediksi nilai berikutnya untuk urutan tertentu. Kasus penggunaannya adalah peramalan pasar saham dan peramalan cuaca. Berikut contohnya:

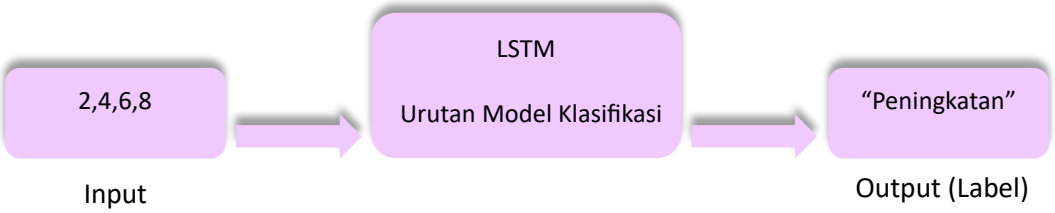
- Urutan input: 3,5,8,12
- Output: 17



Klasifikasi Urutan

Klasifikasi urutan memprediksi label kelas untuk urutan tertentu. Kasus penggunaannya adalah deteksi penipuan (yang menggunakan urutan transaksi sebagai input untuk mengklasifikasikan/memprediksi apakah akun telah diretas atau tidak) dan klasifikasi siswa berdasarkan kinerja (urutan nilai ujian selama enam bulan terakhir secara kronologis). Berikut contohnya:

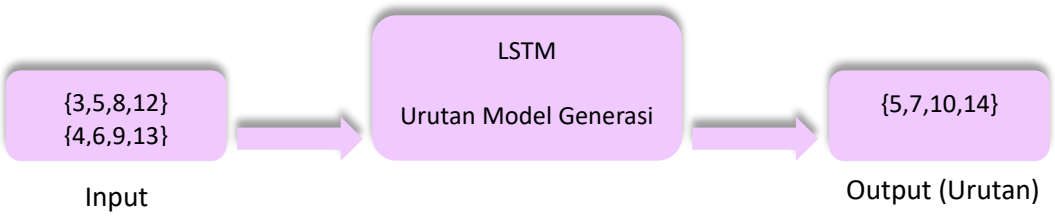
- Urutan input: 2,4,6,8
- Output: “Peningkatan”



Pembuatan Urutan

Pembuatan urutan adalah saat Anda membuat urutan keluaran baru yang memiliki properti yang sama dengan urutan masukan dalam korpus masukan. Kasus penggunaannya adalah pembuatan teks (diberikan 100 baris blog, buat baris blog berikutnya) dan pembuatan musik (diberikan contoh musik, buat karya musik baru). Berikut contohnya:

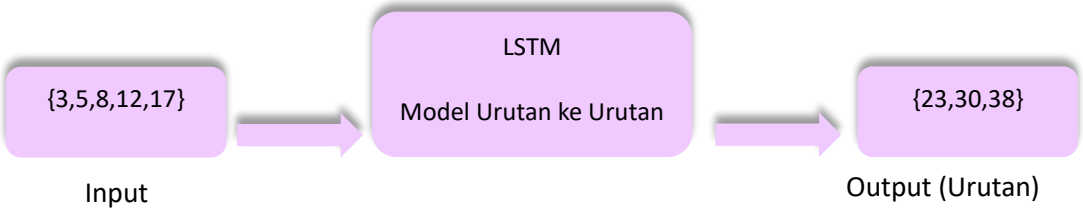
- Urutan masukan: [3, 5,8,12], [4,6,9,13]
- Keluaran: [5,7,10,14]



Prediksi Urutan-ke-Urutan

Prediksi urutan-ke-urutan adalah saat Anda memprediksi urutan berikutnya untuk urutan tertentu. Kasus penggunaannya adalah peringkasan dokumen dan peramalan deret waktu multilangkah (memprediksi urutan angka). Berikut contohnya:

- Urutan masukan: [3, 5,8,12,17]
- Keluaran: [23,30,38]



Seperti yang disebutkan, LSTM digunakan untuk peramalan deret waktu dalam bisnis.

Mari kita bahas model LSTM. Asumsikan bahwa file CSV diberikan di mana kolom pertama adalah stempel waktu dan kolom kedua adalah nilai. File ini dapat mewakili data sensor (IoT). Dengan data deret waktu, Anda harus memprediksi nilai untuk masa mendatang.

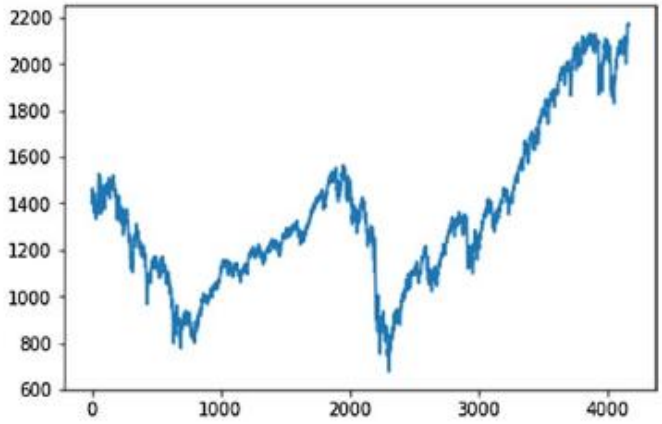
9.3 PERAMALAN DERET WAKTU DENGAN MODEL LSTM

Berikut adalah contoh lengkap peramalan deret waktu dengan LSTM:

```
# Simple LSTM for a time series data
import numpy as np
import matplotlib.pyplot as plt
from pandas import read_csv
import math
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error
import pylab

# convert an array of values into a timeseries data
def create_timeseries(series, ts_lag=1):
   dataX = []
   dataY = []
   n_rows = len(series)-ts_lag
   for i in range(n_rows-1):
      a = series[i:(i+ts_lag), 0]
      dataX.append(a)
      dataY.append(series[i + ts_lag, 0])
    X, Y = np.array(dataX), np.array(dataY)
    return X, Y
# fix random seed for reproducibility
np.random.seed(230)
# load dataset
dataframe = read_csv('sp500.csv', usecols=[0])
plt.plot(dataframe)
```

```
plt.show()
```

Gambar 9-5 memperlihatkan plot data.



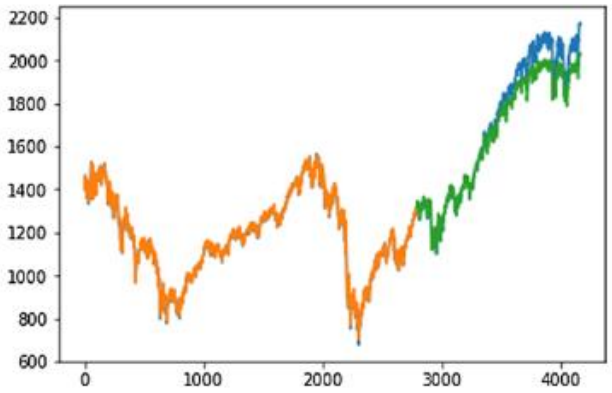
Gambar 9-5. Plot data

Berikut beberapa kode lainnya:

```
# Changing datatype to float32 type
series = dataframe.values.astype('float32')

# Normalize the dataset
scaler = StandardScaler()
series = scaler.fit_transform(series)

# split the datasets into train and test sets
train_size = int(len(series) * 0.75)
test_size = len(series) - train_size
train, test = series[0:train_size,:], series[train_size:len(series),:]

# reshape the train and test dataset into X=t and Y=t+1
ts_lag = 1
trainX, trainY = create_timeseries(train, ts_lag)
testX, testY = create_timeseries(test, ts_lag)

# reshape input data to be [samples, time steps, features]
trainX = np.reshape(trainX, (trainX.shape[0], 1, trainX. shape[1]))
testX = np.reshape(testX, (testX.shape[0], 1, testX.shape[1]))

# Define the LSTM model
model = Sequential()
model.add(LSTM(10, input_shape=(1, ts_lag)))
model.add(Dense(1))
model.compile(loss='mean_squared_logarithmic_error', optimizer='adagrad')

# fit the model
model.fit(trainX, trainY, epochs=500, batch_size=30)
# make predictions
trainPredict = model.predict(trainX)
```

```
testPredict = model.predict(testX)

# rescale predicted values
trainPredict = scaler.inverse_transform(trainPredict)
trainY = scaler.inverse_transform([trainY])
testPredict = scaler.inverse_transform(testPredict)
testY = scaler.inverse_transform([testY])

# calculate root mean squared error
trainScore = math.sqrt(mean_squared_error(trainY[0], trainPredict[:,0]))
print('Train Score: %.2f RMSE' % (trainScore))
testScore = math.sqrt(mean_squared_error(testY[0], testPredict[:,0]))
print('Test Score: %.2f RMSE' % (testScore))

# plot baseline and predictions
pylab.plot(trainPredictPlot)
pylab.plot(testPredictPlot)
pylab.show()
```

Pada Gambar 9-6, Anda dapat melihat plot deret waktu aktual versus deret waktu yang diprediksi. Bagian berwarna oranye adalah data pelatihan, bagian berwarna biru adalah data uji, dan bagian berwarna hijau adalah output yang diprediksi.



Gambar 9-6. Plot deret waktu aktual versus prediksi

Sejauh ini, kita telah mempelajari konsep RNN, LSTM, dan peramalan deret waktu dengan model LSTM.

LSTM telah digunakan dalam klasifikasi teks. Kita menggunakan LSTM (vanilla LSTM atau bi-directional LSTM) untuk membangun pengklasifikasi teks. Pertama, korpus teks diubah menjadi angka dengan menggunakan penyisipan kata (semantik) seperti word2vec atau glove. Kemudian, klasifikasi sekuens dilakukan melalui LSTM.

Pendekatan ini menawarkan akurasi yang jauh lebih tinggi daripada bag of words atau tf-idf yang diikuti oleh pengklasifikasi ML seperti SVM, Random Forest. Dalam bab 11, kita dapat melihat bagaimana LSTM dapat digunakan untuk pengklasifikasi.

BAB 10
KONVERSI UCAPAN KE TEKS DAN SEBALIKNYA

Dalam bab ini, Anda akan mempelajari tentang pentingnya konversi ucapan ke teks dan teks ke ucapan. Anda juga akan mempelajari tentang fungsi dan komponen yang diperlukan untuk melakukan jenis konversi ini.

Secara khusus, penulis akan membahas hal-hal berikut:

- Mengapa Anda ingin mengonversi ucapan ke teks
- Ucapan sebagai data
- Fitur ucapan yang memetakan ucapan ke matriks
- Spektrogram, yang memetakan ucapan ke gambar
- Membangun pengklasifikasi untuk pengenalan ucapan melalui fitur koefisien cepstral frekuensi-mel (MFCC)
- Membangun pengklasifikasi untuk pengenalan ucapan melalui spektrogram
- Pendekatan sumber terbuka untuk pengenalan ucapan
- Penyedia layanan kognitif populer
- Masa depan ucapan teks

10.1 KONVERSI UCAPAN KE TEKS

Konversi ucapan ke teks, dalam istilah awam, berarti bahwa sebuah aplikasi mengenali kata-kata yang diucapkan oleh seseorang dan mengubah suara menjadi teks tertulis. Ada banyak alasan mengapa Anda ingin menggunakan konversi Ucapan ke Teks.

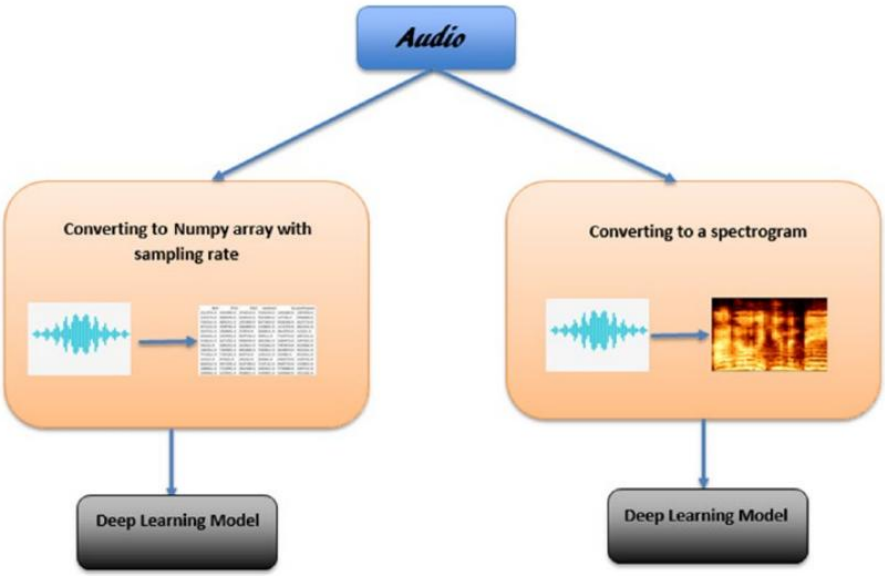
- Orang yang tuna netra atau memiliki keterbatasan fisik dapat mengendalikan berbagai perangkat hanya dengan menggunakan suara.
- Anda dapat menyimpan rekaman rapat dan acara lainnya dengan mengubah percakapan lisan menjadi transkrip teks.
- Anda dapat mengubah audio dalam file video dan audio untuk mendapatkan subtitle dari kata-kata yang diucapkan.
- Anda dapat menerjemahkan kata-kata ke bahasa lain dengan berbicara ke perangkat dalam satu bahasa dan mengubah teks menjadi ucapan dalam bahasa lain.

Ucapan sebagai Data

Langkah pertama dalam membuat sistem pengenalan ucapan otomatis adalah mendapatkan fitur-fiturnya. Dengan kata lain, Anda mengidentifikasi komponen gelombang audio yang berguna untuk mengenali konten linguistik dan menghapus semua fitur tidak berguna lainnya yang hanya berupa suara latar. Ucapan setiap orang disaring oleh bentuk saluran vokal mereka dan juga oleh lidah dan gigi.

Suara yang keluar bergantung pada bentuk ini. Untuk mengidentifikasi fonem yang dihasilkan secara akurat, Anda perlu menentukan bentuk ini secara akurat. Anda dapat mengatakan bahwa bentuk saluran vokal memanifestasikan dirinya untuk membentuk amplop spektrum daya jangka pendek. Tugas MFCC adalah merepresentasikan amplop ini secara

akurat. Ucapan juga dapat direpresentasikan sebagai data dengan mengubahnya menjadi spektrogram (Gambar 10-1).



Gambar 10-1. Ucapan sebagai data

Fitur Ucapan: Memetakan Ucapan ke Matriks

MFCC banyak digunakan dalam ucapan otomatis dan pengenalan pembicara. Skala mel menghubungkan frekuensi yang dirasakan, atau nada, dari nada murni dengan frekuensi terukurnya yang sebenarnya.

Anda dapat mengonversi audio dalam skala frekuensi ke skala mel menggunakan rumus berikut:

$$M(f) = 1125\, ln(\, 1 + f/700)$$

Untuk mengubahnya kembali menjadi frekuensi, gunakan rumus berikut:

$$M^{-1} = (m) = 700(\exp(m/1125) - 1$$

Berikut adalah fungsi untuk mengekstrak fitur MFCC dalam Python:

```
def mfcc(signal,samplerate=16000,winlen=0.025,winstep=0.01,
  numcep=13, nfilt=26,nfft=512,lowfreq=0,highfreq=None,
  preemph=0.97, ceplifter=22,appendEnergy=True)
```

Berikut ini adalah parameter yang digunakan:

- `signal`: Ini adalah sinyal yang Anda perlukan untuk menghitung fitur MFCC. Sinyal ini harus berupa array $N * 1$ (baca file WAV).
- `samplerate`: Ini adalah laju sampel sinyal yang sedang Anda gunakan.
- `winlen`: Ini adalah panjang jendela analisis dalam detik. Secara default, ini adalah

0,025 detik.

- `winstep`: Ini adalah langkah jendela yang berurutan. Secara default, ini adalah 0,01 detik.
- `numcep`: Ini adalah jumlah ceptrum yang harus dikembalikan oleh fungsi. Secara default, ini adalah 13.
- `nfilt`: Ini adalah jumlah filter di bank filter. Secara default, ini adalah 26.
- `nfft`: Ini adalah ukuran transformasi Fourier cepat (FFT). Secara default, ini adalah 512.
- `lowfreq`: Ini adalah tepi pita terendah, dalam hertz. Secara default nilainya adalah 0.
- `highfreq`: Ini adalah tepi pita tertinggi, dalam hertz. Secara default nilainya adalah laju sampel dibagi 2.
- `preemph`: Ini menerapkan filter preemphasis dengan preemph
- sebagai koefisien. 0 berarti tidak ada filter. Secara default adalah 0,97.
- `ceplifter`: Ini menerapkan lifter ke koefisien cepstral akhir. 0 berarti tidak ada lifter. Secara default adalah 22.
- `appendEnergy`: Koefisien cepstral ke nol diganti dengan log dari total energi bingkai, jika diatur ke true.

Fungsi ini mengembalikan array Numpy yang berisi fitur. Setiap baris berisi satu vektor fitur.

10.2 SPEKTROGRAM: MEMETAKAN UCAPAN KE GAMBAR

Spektrogram adalah representasi fotografis atau elektronik dari suatu spektrum. Idenya adalah mengonversi berkas audio menjadi gambar dan meneruskan gambar tersebut ke model pembelajaran mendalam seperti CNN dan LSTM untuk analisis dan klasifikasi.

Spektrogram dihitung sebagai urutan FFT dari segmen data berjendela. Format umum adalah grafik dengan dua dimensi geometris; satu sumbu mewakili waktu, dan sumbu lainnya mewakili frekuensi. Dimensi ketiga menggunakan warna atau ukuran titik untuk menunjukkan amplitudo frekuensi tertentu pada waktu tertentu. Spektrogram biasanya dibuat dengan salah satu dari dua cara. Spektrogram dapat didekati sebagai bank filter yang dihasilkan dari serangkaian filter band-pass. Atau, dalam Python, ada fungsi langsung yang memetakan audio ke spektrogram.

Membangun Pengklasifikasi untuk Pengenalan Ucapan Melalui Fitur MFCC

Untuk membangun pengklasifikasi untuk pengenalan ucapan, Anda perlu menginstal paket Python python_speech_features. Anda dapat menggunakan perintah pip install python_speech_features untuk menginstal paket ini.

Fungsi mfcc membuat matriks fitur untuk file audio. Untuk membangun pengklasifikasi yang mengenali suara orang yang berbeda, Anda perlu mengumpulkan data ucapan mereka dalam format WAV. Kemudian Anda mengonversi semua file audio menjadi matriks menggunakan fungsi mfcc. Kode untuk mengekstrak fitur dari file WAV ditampilkan di sini:

```
from python_speech_features import mfcc
from python_speech_features import delta
from python_speech_features import logfbank
import scipy.io.wavfile as wav

(samplerate,signal) = wav.read("audio.wav")
mfccfeatures = mfcc(signal,samplerate)
dmfccfeature = delta(mfccfeatures, 2)
fbankfeature = logfbank(signal,samplerate)

print(fbankfeature)
```

Jika Anda menjalankan kode sebelumnya, Anda akan mendapatkan output dalam bentuk berikut:

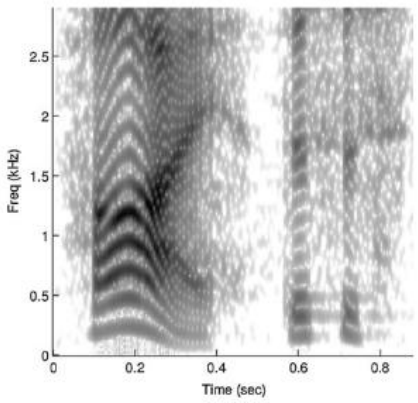
```
[[ 7.66608682  7.04137131  7.30715423 ..., 9.43362359  9.11932984  9.93454603]
 [ 4.9474559   4.97057377  6.90352236 ..., 8.6771281   8.86454547  9.7975147 ]
 [ 7.4795622   6.63821063  5.98854983 ..., 8.78622734  8.805521    9.83712966]
 ...,
 [ 7.8886269   6.57456605  6.47895433 ..., 8.62870034  8.79965464  9.67997298]
 [ 5.73028657  4.87985847  6.64977329 ..., 8.64089442  8.62887745  9.90470194]
 [ 8.8449656   6.67098127  7.09752316 ..., 8.84914694  8.97807983  9.45123015]]
```

- Di sini, setiap baris mewakili satu vektor fitur.
- Kumpulkan rekaman suara seseorang sebanyak mungkin dan tambahkan matriks fitur setiap berkas audio dalam matriks ini.
- Ini akan bertindak sebagai set data pelatihan Anda.
- Ulangi langkah yang sama dengan semua kelas lainnya.

Setelah set data disiapkan, Anda dapat memasukkan data ini ke dalam model pembelajaran mendalam (yang digunakan untuk klasifikasi) untuk mengklasifikasikan suara orang yang berbeda.

Membangun Pengklasifikasi untuk Pengenalan Ucapan Melalui Spektogram

Menggunakan pendekatan spektogram mengubah semua berkas audio menjadi gambar (Gambar 10-2), jadi yang harus Anda lakukan adalah mengubah semua berkas suara dalam data pelatihan menjadi gambar dan memasukkan gambar tersebut ke model pembelajaran mendalam seperti yang Anda lakukan di CNN.



Gambar 10-2. Spektrogram sampel ucapan

Berikut adalah kode Python untuk mengonversi berkas audio menjadi spektrogram:

```
import matplotlib.pyplot as plt
from scipy import signal
from scipy.io import wavfile

sample_rate, samples = wavfile.read('monoAudioFile.wav')
frequencies, times, spectogram = signal.spectrogram(samples, sample_rate)

plt.imshow(spectogram)
plt.ylabel('Freq(kHz)')
plt.xlabel('Time (sec)')
plt.show()
```

10.3 PENDEKATAN OPEN SOURCE

Tersedia beberapa paket open source untuk Python yang melakukan konversi ucapan ke teks dan teks ke ucapan.

Berikut ini adalah beberapa API konversi ucapan ke teks open source:

- PocketSphinx
- Google Speech
- Google Cloud Speech
- Wit.ai
- Houndify
- IBM Speech to Text API
- Microsoft Bing Speech

Setelah menggunakan semua ini, saya dapat mengatakan bahwa semuanya bekerja dengan cukup baik; aksen Amerika sangat jelas.

Jika Anda tertarik untuk mengevaluasi keakuratan konversi, Anda memerlukan satu metrik: rasio kesalahan kata (WER). Di bagian berikutnya, saya akan membahas setiap API yang disebutkan sebelumnya.

Penggunaan PocketSphinx

PocketSphinx adalah API open source yang digunakan untuk konversi ucapan ke teks. Ini adalah mesin pengenalan ucapan yang ringan, yang secara khusus disesuaikan untuk perangkat genggam dan seluler, meskipun berfungsi sama baiknya di desktop. Cukup gunakan perintah pip install PocketSphinx untuk menginstal paket tersebut.

```
import speech_recognition as sr
from os import path
AUDIO_FILE = "MyAudioFile.wav"

r = sr.Recognizer()
with sr.AudioFile(AUDIO_FILE) as source: audio = r.record(source)

try:
 print("Sphinx thinks you said " + r.recognize_sphinx(audio))
except sr.UnknownValueError:
 print("Sphinx could not understand audio")
except sr.RequestError as e:
 print("Sphinx error; {0}".format(e))
==================================================================
```

Menggunakan Google Speech API

Google menyediakan Speech API miliknya sendiri yang dapat diimplementasikan dalam kode Python dan dapat digunakan untuk membuat berbagai aplikasi.

```
# recognize speech using Google Speech Recognition
try:
 print("Google Speech Recognition thinks you said " + r.recognize_google(audio))
except sr.UnknownValueError:
 print("Google Speech Recognition could not understand audio")
except sr.RequestError as e:
 print("Could not request results from Google Speech
 Recognition service;{0}".format(e))
```

Menggunakan Google Cloud Speech API

Anda juga dapat menggunakan Google Cloud Speech API untuk konversi. Buat akun di Google Cloud dan salin kredensialnya.

```
GOOGLE_CLOUD_SPEECH_CREDENTIALS = r"INSERT THE CONTENTS OF THE
GOOGLE CLOUD SPEECH JSON CREDENTIALS FILE HERE" try:
   print("Google Cloud Speech thinks you said " +
   r.recognize_google_cloud(audio,
   credentials_json=GOOGLE_CLOUD_SPEECH_CREDENTIALS))
except sr.UnknownValueError:
   print("Google Cloud Speech could not understand audio")
except sr.RequestError as e:
   print("Could not request results from Google Cloud Speech
   service; {0}".format(e))
```

Menggunakan Wit.ai API

Wit.ai API memungkinkan Anda membuat konverter ucapan ke teks. Anda perlu membuat akun, lalu membuat proyek. Salin kunci Wit.ai Anda dan mulai membuat kode.

```
#recognize speech using Wit.ai
WIT_AI_KEY = "INSERT WIT.AI API KEY HERE" # Wit.ai keys are 32-character
uppercase alphanumeric strings
try:
 print("Wit.ai thinks you said " + r.recognize_wit(audio, key=WIT_AI_KEY))
except sr.UnknownValueError:
 print("Wit.ai could not understand audio")
except sr.RequestError as e:
 print("Could not request results from Wit.ai service; {0}". format(e))
```

Menggunakan API Houndify

Mirip dengan API sebelumnya, Anda perlu membuat akun di Houndify dan mendapatkan ID dan kunci klien. Ini memungkinkan Anda membuat aplikasi yang merespons suara.

```
# recognize speech using Houndify
HOUNDIFY_CLIENT_ID = "INSERT HOUNDIFY CLIENT ID HERE"
# Houndify client IDs are Base64-encoded strings
HOUNDIFY_CLIENT_KEY = "INSERT HOUNDIFY CLIENT KEY HERE"
# Houndify client keys are Base64-encoded strings
try:
  print("Houndify thinks you said " + r.recognize_houndify(audio,
  client_id=HOUNDIFY_CLIENT_ID, client_
  key=HOUNDIFY_CLIENT_KEY))
except sr.UnknownValueError:
   print("Houndify could not understand audio")
except sr.RequestError as e:
   print("Could not request results from Houndify service; {0}".format(e))
```

Menggunakan IBM Speech to Text API

IBM Speech to Text API memungkinkan Anda menambahkan kemampuan pengenalan suara IBM ke aplikasi Anda. Masuk ke cloud IBM dan mulai proyek Anda untuk mendapatkan nama pengguna dan kata sandi IBM.

```
  # IBM Speech to Text
  # recognize speech using IBM Speech to Text
  IBM_USERNAME = "INSERT IBM SPEECH TO TEXT USERNAME HERE" # IBM
  Speech to Text usernames are strings of the form XXXXXXXX-XXXXXXXX-XXXX-
  XXXXXXXXXXXX
  IBM_PASSWORD = "INSERT IBM SPEECH TO TEXT PASSWORD HERE" # IBM
  Speech to Text passwords are mixed-case alphanumeric strings
  try:
    print("IBM Speech to Text thinks you said " + r.recognize_ ibm(audio,
    username=IBM_USERNAME, password=IBM_PASSWORD))
  except sr.UnknownValueError:
    print("IBM Speech to Text could not understand audio")
  except sr.RequestError as e:
    print("Could not request results from IBM Speech to Text service;
    {0}".format(e))
```

Menggunakan Bing Voice Recognition API

API ini mengenali audio yang keluar dari mikrofon secara real time. Buat akun di Bing.com dan dapatkan kunci Bing Voice Recognition API.

```
  # recognize speech using Microsoft Bing Voice Recognition
  BING_KEY = "INSERT BING API KEY HERE" # Microsoft Bing Voice
  Recognition API key is 32-character lowercase hexadecimal strings
  try:
    print("Microsoft Bing Voice Recognition thinks you said " +
    r.recognize_bing(audio, key=BING_KEY))
  except sr.UnknownValueError:
    print("Microsoft Bing Voice Recognition could not understand audio")
  except sr.RequestError as e:
    print("Could not request results from Microsoft Bing Voice Recognition
```

```
    service; {0}".format(e))
```

Setelah Anda mengubah ucapan menjadi teks, Anda tidak dapat mengharapkan keakuratan 100 persen. Untuk mengukur keakuratan, Anda dapat menggunakan WER.

10.4 KONVERSI TEKS KE UCAPAN

Bagian bab ini berfokus pada konversi teks tertulis ke berkas audio.

Menggunakan pyttsx

Dengan menggunakan paket Python yang disebut pyttsx, Anda dapat mengonversi teks ke audio.

```
Do a pip install pyttsx. If you are using python 3.6 then do
pip3 install pyttsx3.
import pyttsx
engine = pyttsx.init()
engine.say("Your Message")
engine.runAndWait()
```

Menggunakan SAPI

Anda juga dapat menggunakan SAPI untuk melakukan konversi teks ke ucapan dalam Python.

```
from win32com.client import constants, Dispatch
Msg = "Hi this is a test"
speaker = Dispatch("SAPI.SpVoice")      #Create SAPI SpVoice Object
speaker.Speak(Msg)                      #Process TTS
del speaker
```

Menggunakan SpeechLib

Anda dapat mengambil input dari file teks dan mengubahnya menjadi audio menggunakan SpeechLib, seperti yang ditunjukkan di sini:

```
from comtypes.client import CreateObject engine =
CreateObject("SAPI.SpVoice")
stream = CreateObject("SAPI.SpFileStream")
from comtypes.gen import SpeechLib
infile = "SHIVA.txt"
outfile = "SHIVA-audio.wav"
stream.Open(outfile, SpeechLib.SSFMCreateForWrite)
engine.AudioOutputStream = stream
f = open(infile, 'r')
theText = f.read()
f.close()
engine.speak(theText)
stream.Close()
```

Sering kali, Anda harus mengedit audio agar dapat menghapus suara dari berkas audio. Bagian

berikutnya menunjukkan caranya.

Kode Pemotongan Audio

Buat berkas CSV audio yang berisi nilai-nilai yang dipisahkan koma dari detail audio dan lakukan hal berikut menggunakan Python:

```
import wave
import sys
import os
import csv
origAudio = wave.open('Howard.wav', 'r')      #change path
frameRate = origAudio.getframerate()
nChannels = origAudio.getnchannels()
sampWidth = origAudio.getsampwidth()
nFrames = origAudio.getnframes()

filename = 'result1.csv' #change path
exampleFile = open(filename)
exampleReader = csv.reader(exampleFile)
exampleData = list(exampleReader)

count = 0

for data in exampleData:
  #for selections in data:
    print('Selections ', data[4], data[5])
    count += 1
    if data[4] == 'startTime' and data[5] == 'endTime':
      print('Start time')
    else:
      start = float(data[4])
      end = float(data[5])
      origAudio.setpos(start*frameRate)
      chunkData = origAudio.readframes(int((endstart)*frameRate))
      outputFilePath = 'C:/Users/Navin/outputFile{0}.wav'.format(count)
      # change path
      chunkAudio = wave.open(outputFilePath, 'w')
      chunkAudio.setnchannels(nChannels)
      chunkAudio.setsampwidth(sampWidth)
      chunkAudio.setframerate(frameRate)
      chunkAudio.writeframes(chunkData)
      chunkAudio.close()
```

10.5 PENYEDIA LAYANAN KOGNITIF

Mari kita lihat beberapa penyedia layanan kognitif yang membantu pemrosesan ucapan.

Microsoft Azure

Microsoft Azure menyediakan hal-hal berikut:

- Layanan Ucapan Kustom: Layanan ini mengatasi kendala pengenalan ucapan seperti gaya bicara, kosakata, dan kebisingan latar belakang.

- API Ucapan Penerjemah: Layanan ini memungkinkan penerjemahan ucapan secara real-time.
- API Identifikasi Pembicara: Layanan ini dapat mengidentifikasi pembicara berdasarkan sampel ucapan setiap pembicara dalam data audio yang diberikan.
- API Ucapan Bing: Layanan ini mengubah audio menjadi teks, memahami maksud, dan mengubah teks kembali menjadi ucapan untuk respons yang alami.

Amazon Cognitive Services

Amazon Cognitive Services menyediakan Amazon Polly, layanan yang mengubah teks menjadi ucapan. Amazon Polly memungkinkan Anda membuat aplikasi yang berbicara, sehingga Anda dapat membangun kategori produk yang sepenuhnya baru yang mendukung ucapan.

- 47 suara dan 24 bahasa dapat digunakan, dan tersedia opsi Bahasa Inggris India.
- Nada seperti berbisik, marah, dan sebagainya, dapat ditambahkan ke bagian tertentu dari ucapan menggunakan efek Amazon.
- Anda dapat memberi tahu sistem cara mengucapkan frasa atau kata tertentu dengan cara yang berbeda. Misalnya, “W3C” diucapkan sebagai World Wide Web Consortium, tetapi Anda dapat mengubahnya untuk mengucapkan akronimnya saja. Anda juga dapat memberikan teks input dalam format SSML.

Layanan IBM Watson

Ada dua layanan dari IBM Watson.

- Ucapan ke teks: Bahasa Inggris AS, Bahasa Spanyol, dan Bahasa Jepang
- Teks ke ucapan: Bahasa Inggris AS, Bahasa Inggris Inggris, Bahasa Spanyol, Bahasa Prancis, Bahasa Italia, dan Bahasa Jerman

Masa Depan Analisis Ucapan

Teknologi pengenalan ucapan telah mengalami kemajuan yang pesat. Setiap tahun, teknologi ini menjadi sekitar 10 hingga 15 persen lebih akurat daripada tahun sebelumnya. Di masa mendatang, teknologi ini akan menyediakan antarmuka paling interaktif untuk komputer.

Ada banyak aplikasi yang akan segera Anda saksikan di pasaran, termasuk buku interaktif, kontrol robotik, dan antarmuka mobil tanpa pengemudi. Data ucapan menawarkan beberapa kemungkinan baru yang menarik karena merupakan masa depan industri. Kecerdasan bicara memungkinkan orang untuk mengirim pesan, menerima atau memberi perintah, menyampaikan keluhan, dan melakukan pekerjaan apa pun yang biasanya dilakukan dengan mengetik secara manual.

Kecerdasan bicara menawarkan pengalaman pelanggan yang luar biasa dan mungkin itulah sebabnya semua departemen dan bisnis yang berhadapan langsung dengan pelanggan cenderung menggunakan aplikasi bicara secara intensif. Saya dapat melihat masa depan yang cerah bagi pengembang aplikasi bicara.

BAB 11
MENGEMBANGKAN CHATBOT

Sistem kecerdasan buatan yang bertindak sebagai antarmuka untuk interaksi manusia dan mesin melalui teks atau suara disebut chatbot. Interaksi dengan chatbot bisa langsung atau rumit. Contoh interaksi langsung adalah menanyakan berita terkini. Interaksi bisa menjadi lebih rumit jika menyangkut pemecahan masalah, misalnya, pada ponsel Android Anda. Istilah chatbot telah memperoleh popularitas luar biasa dalam setahun terakhir dan telah berkembang menjadi platform yang paling disukai untuk interaksi dan keterlibatan pengguna. Bot, bentuk chatbot tingkat lanjut, membantu mengotomatiskan tugas yang "dilakukan pengguna".

Bab tentang chatbot ini akan menjadi panduan menyeluruh tentang apa, bagaimana, di mana, kapan, dan mengapa chatbot digunakan! Secara khusus, saya akan membahas hal-hal berikut:

- Mengapa Anda ingin menggunakan chatbot
- Desain dan fungsi chatbot
- Langkah-langkah untuk membangun chatbot
- Pengembangan chatbot menggunakan API
- Praktik terbaik chatbot

11.1 MENGAPA CHATBOT?

Penting bagi chatbot untuk memahami informasi apa yang dicari pengguna, yang disebut maksud. Misalkan pengguna ingin tahu restoran vegetarian terdekat; pengguna dapat mengajukan pertanyaan itu dengan berbagai cara. Chatbot (khususnya pengklasifikasi maksud di dalam chatbot) harus dapat memahami maksud karena pengguna ingin mendapatkan jawaban yang tepat.

Faktanya, untuk memberikan jawaban yang tepat, chatbot harus dapat memahami konteks, maksud, entitas, dan sentimen. Chatbot bahkan harus memperhitungkan apa pun yang dibahas dalam sesi tersebut. Misalnya, pengguna mungkin mengajukan pertanyaan "Berapa harga biryani ayam di sana?" Meskipun pengguna telah menanyakan harga, mesin obrolan dapat salah paham dan menganggap pengguna sedang mencari restoran. Jadi, sebagai tanggapan, chatbot dapat memberikan nama restoran tersebut.

Desain dan Fungsi Chatbot

Chatbot merangsang percakapan cerdas dengan manusia melalui penerapan AI. Antarmuka tempat percakapan berlangsung difasilitasi melalui teks lisan atau tertulis. Facebook Messenger, Slack, dan Telegram menggunakan platform pengiriman pesan chatbot. Platform ini melayani banyak keperluan, termasuk memesan produk secara daring, berinvestasi, dan mengelola keuangan, dan sebagainya. Aspek penting chatbot adalah memungkinkan terjadinya percakapan kontekstual. Chatbot berkomunikasi dengan pengguna dengan cara yang mirip dengan cara manusia berkomunikasi dalam kehidupan sehari-hari.

Meskipun chatbot dapat berkomunikasi secara kontekstual, mereka masih perlu melakukan banyak hal dalam hal berkomunikasi secara kontekstual dengan segala hal.

Namun, antarmuka obrolan memanfaatkan bahasa untuk menghubungkan mesin dengan manusia, membantu orang menyelesaikan berbagai hal dengan mudah dengan memberikan informasi secara kontekstual.

Selain itu, chatbot mengubah cara bisnis dijalankan. Dari menjangkau konsumen hingga menyambut mereka di ekosistem bisnis hingga memberikan informasi kepada konsumen tentang berbagai produk dan fiturnya, chatbot membantu semuanya. Chatbot muncul sebagai cara paling nyaman untuk berurusan dengan konsumen secara tepat waktu dan memuaskan.

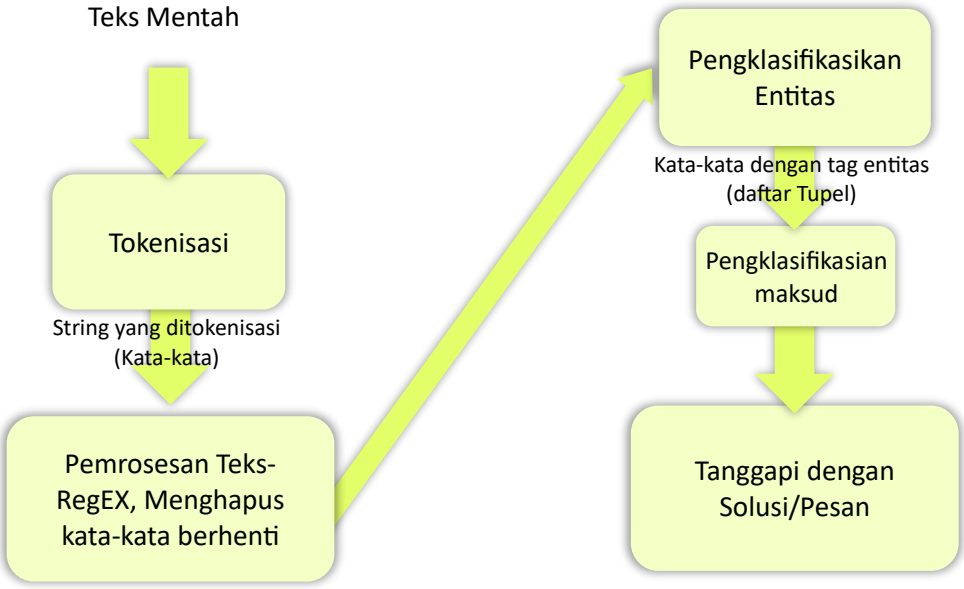
Langkah-langkah untuk Membangun Chatbot

Chatbot dibangun untuk berkomunikasi dengan pengguna dan memberi mereka perasaan bahwa mereka berkomunikasi dengan manusia dan bukan bot. Namun, saat pengguna memberikan masukan, biasanya mereka tidak memberikan masukan dengan cara yang tepat.

Dengan kata lain, mereka mungkin memasukkan tanda baca yang tidak perlu, atau mungkin ada cara berbeda untuk mengajukan pertanyaan yang sama.

Misalnya, untuk "Restoran di dekat saya?" pengguna dapat memasukkan "Restoran di samping saya?" atau "Temukan restoran terdekat."

Oleh karena itu, Anda perlu memproses data terlebih dahulu sehingga mesin chatbot dapat dengan mudah memahaminya. Gambar 11-1 menunjukkan prosesnya, yang dirinci di bagian berikut.



Gambar 11-1. Diagram alir untuk menunjukkan bagaimana mesin chatbot memproses rangkaian masukan dan memberikan balasan yang valid.

Prapemrosesan Teks dan Pesan

Prapemrosesan teks dan pesan mencakup beberapa langkah, yang akan dibahas

selanjutnya.

Tokenisasi

Memotong kalimat menjadi kata-kata tunggal (disebut token) disebut tokenisasi. Dalam Python, umumnya string ditokenisasi dan disimpan dalam daftar.

Misalnya, kalimat "Kecerdasan buatan adalah tentang penerapan matematika" menjadi berikut:

["Artificial", "intelligence", "is", "all", "about", "applying", "mathematics"]

Berikut adalah contoh kodenya:

```
from nltk.tokenize import TreebankWordTokenizer
l = "Artificial intelligence is all about applying mathematics"
token = TreebankWordTokenizer().tokenize(l)
print(token)
```

Menghapus Tanda Baca

Anda juga dapat menghapus tanda baca yang tidak diperlukan dalam kalimat.

Misalnya, kalimat "*Dapatkah saya memperoleh daftar restoran yang menyediakan layanan pesan antar ke rumah*." menjadi seperti berikut:

"*Dapatkah saya memperoleh daftar restoran yang menyediakan layanan pesan antar ke rumah*."

Berikut adalah contoh kodenya:

```
from nltk.tokenize import TreebankWordTokenizer
from nltk.corpus import stopwords
l = "Artificial intelligence is all about applying mathematics!"
token = TreebankWordTokenizer().tokenize(l)
output = []
output = [k for k in token if k.isalpha()]
print(output)
```

Menghapus Kata Henti

Kata henti adalah kata-kata yang ada dalam kalimat yang tidak terlalu berpengaruh jika dihilangkan. Meskipun format kalimat berubah, ini sangat membantu dalam pemahaman bahasa alami (NLU).

Misalnya, kalimat "*Artificial intelligence can change the lifestyle of the people*." menjadi berikut setelah menghapus kata henti:

"*Artificial intelligence change lifestyle people*."

Berikut adalah contoh kodenya:

```
from nltk.tokenize import TreebankWordTokenizer
from nltk.corpus import stopwords
l = "Artificial intelligence is all about applying mathematics"
token = TreebankWordTokenizer().tokenize(l)
stop_words = set(stopwords.words('english'))
output= []
for k in token:
  if k not in stop_words:
    output.append(k)
print(output)
```

Kata-kata yang dianggap sebagai kata henti dapat bervariasi. Ada beberapa set kata henti yang telah ditetapkan sebelumnya yang disediakan oleh Natural Language Toolkit (NLTK), Google, dan lainnya.

Pengenalan Entitas Bernama

Pengenalan entitas bernama (NER), juga dikenal sebagai identifikasi entitas, adalah tugas mengklasifikasikan entitas dalam teks ke dalam kelas-kelas yang telah ditetapkan sebelumnya seperti nama negara, nama orang, dan sebagainya. Anda juga dapat menentukan kelas Anda sendiri.

Misalnya, menerapkan NER pada kalimat "Pertandingan kriket India vs Australia hari ini fantastis." memberikan Anda keluaran berikut:

[Hari ini] Waktu [India] Negara vs [Australia] Negara [kriket] Pertandingan pertandingan fantastis.

Untuk menjalankan kode untuk NER, Anda perlu mengunduh dan mengimpor paket yang diperlukan, seperti yang disebutkan dalam kode berikut.

Menggunakan Stanford NER

Untuk menjalankan kode, unduh file `english.all.3class.distsim.crf.ser.gz` dan `stanford-ner.jar`.

```
from nltk.tag import StanfordNERTagger
from nltk.tokenize import word_tokenize

StanfordNERTagger("stanford-
  ner/classifiers/english.all.3class.distsim.crf.ser.gz","stanford-
  ner/stanford-ner.jar")
text = "Ron was the founder of Ron Institute at New york"
text = word_tokenize(text)
ner_tags = ner_tagger.tag(text)

print(ner_tags)
```

Menggunakan MITIE NER (Pretrained)

Unduh file `ner_model.dat` dari MITIE untuk menjalankan kode.

```
from mitie.mitie import *
from nltk.tokenize import word_tokenize

print("loading NER model...")
ner = named_entity_extractor("mitie/MITIE-
  models/english/ner_model.dat".encode("utf8"))

text = "Ron was the founder of Ron Institute at New york".
encode("utf-8")
text = word_tokenize(text)

ner_tags = ner.extract_entities(text)
print("\nEntities found:", ner_tags)
for e in ner_tags:
  range = e[0]
  tag = e[1]
  entity_text = " ".join(text[i].decode() for i in range)
  print( str(tag) + " : " + entity_text)
```

Menggunakan MITIE NER (Self-Trained)

Unduh file `total_word_feature_extractor.dat` dari MITIE (https://github.com/mit-nlp/MITIE) untuk menjalankan kode.

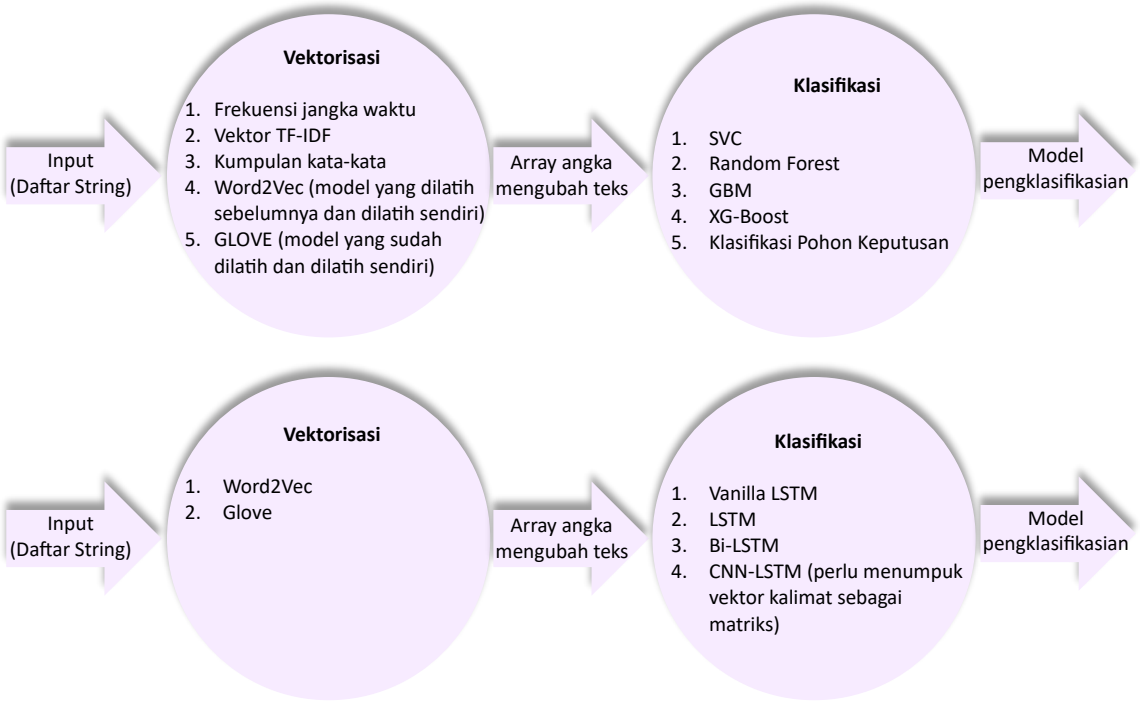
```
from mitie.mitie import *
sample = ner_training_instance([b"Ron", b"was", b"the", b"founder", b"of",
  b"Ron", b"Institute", b"at", b"New", b"York", b"."])

sample.add_entity(range(0, 1), "person".encode("utf-8"))
sample.add_entity(range(5, 7), "organization".encode("utf-8"))
sample.add_entity(range(8, 10), "Location".encode("utf-8"))

trainer = ner_trainer("mitie/MITIE-models/english/total_word_
  feature_extractor.dat".encode("utf-8"))

trainer.add(sample)

ner = trainer.train()

tokens = [b"John", b"was", b"the", b"founder", b"of", b"John",
b"University", b"."]
entities = ner.extract_entities(tokens)
print ("\nEntities found:", entities)
for e in entities:
  range = e[0]
  tag = e[1]
  entity_text = " ".join(str(tokens[i]) for i in range)
  print (" " + str(tag) + ": " + entity_text)
```

Klasifikasi Maksud

Klasifikasi maksud adalah langkah dalam NLU di mana Anda mencoba memahami apa yang diinginkan pengguna. Berikut adalah dua contoh masukan ke chatbot untuk menemukan tempat terdekat:

- "Saya perlu membeli bahan makanan.": Maksudnya adalah mencari toko bahan makanan terdekat.
- "Saya ingin makan makanan vegetarian.": Maksudnya adalah mencari restoran terdekat, idealnya restoran vegetarian.

Pada dasarnya, Anda perlu memahami apa yang dicari pengguna dan mengklasifikasikan permintaan tersebut ke dalam kategori maksud tertentu (Gambar 11-2).



Gambar 11-2. Alur umum klasifikasi maksud, dari kalimat ke vektor ke model

Untuk melakukan ini, Anda perlu melatih model untuk mengklasifikasikan permintaan ke dalam maksud menggunakan algoritme, mulai dari kalimat ke vektor ke model.

Penyisipan Kata

Penyisipan kata adalah teknik mengubah teks menjadi angka. Sulit untuk menerapkan algoritme apa pun dalam teks. Oleh karena itu, Anda harus mengubahnya menjadi angka. Berikut ini adalah berbagai jenis teknik penyematan kata.

Hitung Vektor

Misalkan Anda memiliki tiga dokumen (D1, D2, dan D3) dan ada N kata unik dalam kelompok dokumen tersebut. Anda membuat matriks (D×N), yang disebut C, yang dikenal sebagai vektor hitungan. Setiap entri matriks adalah frekuensi kata unik dalam dokumen tersebut.

Mari kita lihat ini menggunakan sebuah contoh.
D1: *Pooja is very lazy.*
D2: *But she is intelligent.*
D3: *She hardly comes to class..*
Di sini, D=3 dan N=12.

Kata-kata uniknya adalah barely, lazy, But, to, Pooja, she, intelligent, comes, very, class, dan is.

Oleh karena itu, vektor hitungan, C, akan menjadi sebagai berikut:

	hardly	***laziest***	***but***	***to***	***pooja***	***she***	***intelligent***	***comes***	***very***	***class***	***is***
D1	0	1	0	0	1	0	0	0	1	0	1
D2	0	0	1	0	0	1	1	0	0	0	1
D3	1	0	0	1	0	1	0	1	0	1	0

11.2 TERM FREQUENCY-INVERSE DOCUMENT FREQUENCY (TF-IDF)

Untuk teknik ini, Anda memberi setiap kata dalam kalimat sebuah angka tergantung pada seberapa sering kata itu muncul dalam kalimat itu dan juga tergantung pada dokumennya. Kata-kata yang muncul berkali-kali dalam sebuah kalimat dan tidak terlalu sering muncul dalam sebuah dokumen akan memiliki nilai yang tinggi.
Misalnya, perhatikan serangkaian kalimat:

- *"I am a boy."*
- *"I am a girl."*
- *"Where do you live?"*

TF-IDF mengubah rangkaian fitur untuk kalimat sebelumnya, seperti yang ditunjukkan di sini:

	Am	**Boy**	**Girl**	**Where**	**do**	**you**	**Live**
1	0.60	0.80	0	0	0	0	0
2	0.60	0	0.80	0	0	0	0
3	0	0	0	0.5	0.5	0.5	0.5

Anda dapat mengimpor paket TFIDF dan menggunakannya untuk membuat tabel ini.

Sekarang mari kita lihat beberapa contoh kode. Anda dapat menggunakan pengklasifikasi vektor pendukung pada fitur-fitur yang ditransformasikan TF-IDF dari string permintaan.

```
#import required packages
import pandas as pd
from random import sample
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.svm import SVC
```

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score, accuracy_score
# read csv file
data = pd.read_csv("intent1.csv")
print(data.sample(6))
```

Sebelum melanjutkan dengan kode, berikut contoh kumpulan datanya:

Deskripsi (Pesan)	**Intent_label (target)**
Restoran Non-Vegetarian yang Bagus di Dekat Saya	0
Saya mencari rumah sakit	1
Rumah sakit yang bagus untuk operasi Jantung	1
Sekolah internasional untuk anak-anak	2
Restoran non-vegetarian di sekitar saya	0
Sekolah untuk Anak kecil	2

Dalam contoh ini, berikut nilai yang akan digunakan:

- 0 berarti mencari restoran.
- 1 berarti mencari rumah sakit.
- 2 berarti mencari sekolah. Sekarang mari kita bahas kumpulan data tersebut.

```
# split dataset into train and test.
X_train, X_test, Y_train, Y_test = train_test_split(data
["Description"], data["intent_label"], test_size=3)
print(X_train.shape, X_test.shape, Y_train.shape, Y_test.shape)
# vectorize the input using tfidf values.
tfidf = TfidfVectorizer()
tfidf = tfidf.fit(X_train)
X_train = tfidf.transform(X_train)
X_test = tfidf.transform(X_test)
# label encoding for different categories of intents
le = LabelEncoder().fit(Y_train)
Y_train = le.transform(Y_train)
Y_test = le.transform(Y_test)
# other models like GBM, Random Forest may also be used
model = SVC()
model = model.fit(X_train, Y_train)
p = model.predict(X_test)
# calculate the f1_score. average="micro" since we want to calculate score
for multiclass.
# Each instance(rather than class(search for macro average))
contribute equally towards the scoring.
print("f1_score:", f1_score( Y_test, p, average="micro"))
print("accuracy_score:",accuracy_score(Y_test, p))
```

11.3 WORD2VEC

Ada berbagai metode untuk mendapatkan vektor kata untuk sebuah kalimat, tetapi teori utama di balik semua teknik tersebut adalah untuk memberikan kata-kata yang serupa representasi vektor yang serupa. Jadi, kata-kata seperti man dan boy dan girl akan memiliki vektor yang serupa. Panjang setiap vektor dapat diatur. Contoh teknik Word2vec meliputi GloVe dan CBOW (n-gram dengan atau tanpa skip gram).

Anda dapat menggunakan Word2vec dengan melatihnya untuk set data Anda sendiri (jika Anda memiliki cukup data untuk masalah tersebut), atau Anda dapat menggunakan data yang telah dilatih sebelumnya. Word2vec tersedia di Internet. Model yang telah dilatih sebelumnya telah dilatih pada dokumen besar seperti data Wikipedia, tweet, dan sebagainya, dan model tersebut hampir selalu bagus untuk masalah tersebut.

Contoh beberapa teknik yang dapat Anda gunakan untuk melatih pengklasifikasi maksud Anda adalah dengan menggunakan 1D-CNN pada vektor kata dari kata-kata dalam sebuah kalimat, yang ditambahkan dalam daftar untuk setiap kalimat.

```
# import required packages
from gensim.models import Word2Vec
import pandas as pd
import numpy as np
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.utils.np_utils import to_categorical
from keras.layers import Dense, Input, Flatten
from keras.layers import Conv1D, MaxPooling1D, Embedding, Dropout
from keras.models import Model
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import f1_score, accuracy_score

# read data
data = pd.read_csv("intent1.csv")

# split data into test and train
X_train, X_test, Y_train, Y_test = train_test_split(data ["Description"],
data["intent_label"], test_size=6)

# label encoding for different categories of intents
le = LabelEncoder().fit(Y_train)
Y_train = le.transform(Y_train)
Y_test = le.transform(Y_test)

# get word_vectors for words in training set
X_train = [sent for sent in X_train]
X_test = [sent for sent in X_test]

# by default genism.Word2Vec uses CBOW, to train word vecs.
We can also use skipgram with it
# by setting the "sg" attribute to number of skips we want.
```

```
# CBOW and Skip gram for the sentence "Hi Ron how was your day?" becomes:
# Continuos bag of words: 3-grams {"Hi Ron how", "Ron how was", "how was
your" ...}
# Skip-gram 1-skip 3-grams: {"Hi Ron how", "Hi Ron was", "Hi how was", "Ron
how your", ...}

# See how: "Hi Ron was" skips over "how".
# Skip-gram 2-skip 3-grams: {"Hi Ron how", "Hi Ron was", "Hi
Ron your", "Hi was your", ...}
# See how: "Hi Ron your" skips over "how was".
# Those are the general meaning of CBOW and skip gram.
word_vecs = Word2Vec(X_train)
print("Word vectors trained")
# prune each sentence to maximum of 20 words.
max_sent_len = 20

# tokenize input strings
tokenizer = Tokenizer()
tokenizer.fit_on_texts(X_train)
sequences = tokenizer.texts_to_sequences(X_train)
sequences_test = tokenizer.texts_to_sequences(X_test)
word_index = tokenizer.word_index
vocab_size = len(word_index)

# sentences with less than 20 words, will be padded with zeroes
to make it of length 20
# sentences with more than 20 words, will be pruned to 20.
x = pad_sequences(sequences, maxlen=max_sent_len)
X_test = pad_sequences(sequences_test, maxlen=max_sent_len)
# 100 is the size of wordvec.
embedding_matrix = np.zeros((vocab_size + 1, 100))
# make matrix of each word with its word_vectors for the CNN model.
# so each row of a matrix will represent one word. There will be a row for
each word in the training set
for word, i in word_index.items():
    try:
      embedding_vector = word_vecs[word]
    except:
      embedding_vector = None
      if embedding_vector is not None:
        embedding_matrix[i] = embedding_vector
print("Embeddings done")
vocab_size = len(embedding_matrix)

# CNN model requires multiclass labels to be converted into one hot ecoding.
# i.e. each column represents a label, and will be marked one for
corresponding label.
y = to_categorical(np.asarray(Y_train))

embedding_layer = Embedding(vocab_size, 100, weights=[embedding_matrix],
     input_length=max_sent_len, trainable=True)
sequence_input = Input(shape=(max_sent_len,), dtype='int32')
```

```
# stack each word of a sentence in a matrix. So each matrix represents a
sentence.
# Each row in a matrix is a word(Word Vector) of a sentence.
embedded_sequences = embedding_layer(sequence_input)

# build the Convolutional model.
l_cov1 = Conv1D(128, 4, activation='relu')(embedded_sequences)
l_pool1 = MaxPooling1D(4)(l_cov1)
l_flat = Flatten()(l_pool1)
hidden = Dense(100, activation='relu')(l_flat)
preds = Dense(len(y[0]), activation='softmax')(hidden)
model = Model(sequence_input, preds)
model.compile(loss='binary_crossentropy',optimizer='Adam')

print("model fitting - simplified convolutional neural network")
model.summary()

# train the model
model.fit(x, y, epochs=10, batch_size=128)

#get scores and predictions.
p = model.predict(X_test)
p = [np.argmax(i) for i in p]
score_cnn = f1_score(Y_test, p, average="micro")
print("accuracy_score:",accuracy_score(Y_test, p))
print("f1_score:", score_cnn)
```

Model yang digunakan adalah jaringan saraf konvolusional yang disederhanakan, seperti yang ditunjukkan di sini:

Layer (Type)	Bentuk Output	Parameter #
input_20 (InputLayer)	(None, 20)	0
embedding_20 (Embedding)	(None, 20, 100)	2800
conv1d_19 (Conv1D)	(None, 17, 128)	51328
max_pooling1d_19 (MaxPooling)	(None, 4, 128)	0
flatten_19 (Flatten)	(None, 512)	0
dense_35 (Dense)	(None, 100)	51300
dense_36 (Dense)	(None, 3)	303

Berikut jumlah parameternya:

- Total parameter: 105.731
- Parameter yang dapat dilatih: 105.731
- Parameter yang tidak dapat dilatih: 0

Berikut beberapa fungsi penting Word2vec menggunakan paket Gensim:

- Beginilah cara mengimpor Gensim dan memuat model yang telah dilatih:
  `import genism`

```
#loading the pre-trained model
model = gensim.models.KeyedVectors.
load_word2vec_format('GoogleNews-vectors-negative300.bin', binary=True)
```

- Ini adalah model yang telah dilatih sebelumnya dari Google untuk bahasa Inggris, dan memiliki 300 dimensi.
- Berikut cara menemukan vektor kata dari sebuah kata dari model yang telah dilatih sebelumnya:

```
# getting word vectors of a word
lion = model['lion']
print(len(lion))
```

- Berikut cara mencari indeks kesamaan antara dua kata:

```
#Calculating similarity index
print(model.similarity('King', 'Queen'))
```

- Berikut cara menemukan kata ganjil dari sekumpulan kata:

```
#Choose odd one out
print(model.doesnt_match("Mango Grape Tiger Banana Strawberry".split()))
```

- Berikut cara menemukan kata-kata yang paling mirip:

```
print(model.most_similar(positive=[Prince, Girl], negative=[Boy]))
```

Fitur unik Word2vec adalah Anda bisa mendapatkan vektor dari vektor lain menggunakan operasi vektor. Misalnya, vektor "*Prince*" dikurangi vektor "*Boy*" ditambah vektor "*Girl*" akan hampir sama dengan vektor "*Princess*". Oleh karena itu, saat Anda menghitungnya, Anda akan mendapatkan vektor "*Princess*".

```
Vec ("Prince") – Vec("boy") + Vec("girl") ≈ Vec("Princess")
```

Ini hanyalah sebuah contoh. Kasus ini berlaku dalam banyak kasus lainnya. Ini adalah spesialisasi Word2vec dan berguna dalam memperkirakan kata-kata yang mirip, kata-kata berikutnya, pembangkitan bahasa alami (NLG), dan sebagainya. Tabel 11-1 menunjukkan model yang telah dilatih sebelumnya dengan parameter lainnya.

Tabel 11-1. Berbagai Model Pra-Pelatihan dengan Parameter Lain

Model File	Jumlah	Corpus Size	Ukuran Vocabulary	Arsitektur	Ukuran Konteks Window	Penemu
Google News	300	100B	3M	Word2Vec	BoW, ~5	Google
Freebase IDs	1000	100B	1.4M	Word2Vec, Skip-gram	BoW, ~10	Google
Freebase names	1000	100B	1.4M	Word2Vec, Skip-gram	BoW, ~10	Google
Wikipedia + Gigaword 5	50	6B	400,000	GloVe	10+10	GloVe

Wikipedia + Gigaword 5	100	6B	400,000	GloVe	10+10	GloVe
Wikipedia + Gigaword 5	200	6B	400,000	GloVe	10+10	GloVe
Wikipedia + Gigaword 5	300	6B	400,000	GloVe	10+10	GloVe
Common Crawl 42B	300	42B	1.9M	GloVe	AdaGrad	GloVe
Common Crawl 840B	300	840B	2.2M	GloVe	AdaGrad	GloVe
Wikipedia dependency	300	-	174,000	Word2Vec	Syntactic Dependencies	Levy & Goldberg
DBPedia vectors (wiki2vec)	1000	-	-	Word2Vec	BoW, 10	Idio

11.4 MEMBANGUN RESPONS

Respons merupakan bagian penting lain dari chatbot. Berdasarkan cara chatbot membalas, pengguna mungkin tertarik padanya. Setiap kali chatbot dibuat, satu hal yang harus diingat adalah penggunanya. Anda perlu tahu siapa yang akan menggunakannya dan untuk tujuan apa chatbot akan digunakan. Misalnya, chatbot untuk situs web restoran hanya akan ditanya tentang restoran dan makanan. Jadi, Anda kurang lebih tahu pertanyaan apa yang akan ditanyakan. Oleh karena itu, untuk setiap maksud, Anda menyimpan beberapa jawaban yang dapat digunakan setelah mengidentifikasi maksud tersebut sehingga pengguna tidak akan mendapatkan jawaban yang sama berulang kali. Anda juga dapat memiliki satu maksud untuk pertanyaan di luar konteks; maksud tersebut dapat memiliki beberapa jawaban, dan dengan memilih secara acak, chatbot dapat membalas.

Misalnya, jika maksudnya adalah "halo," Anda dapat memiliki beberapa balasan seperti "Halo! Apa kabar?" dan "Halo! Apa kabar?" dan "Hai! Ada yang bisa saya bantu?"

Chatbot dapat memilih salah satu secara acak untuk balasan.

Dalam contoh kode berikut, Anda mengambil masukan dari pengguna, tetapi dalam chatbot asli, maksudnya ditentukan oleh chatbot itu sendiri berdasarkan pertanyaan yang diajukan oleh pengguna.

```
import random
intent = input()
output = ["Hello! How are you","Hello! How are you doing","Hii!
How can I help you","Hey! There","Hiiii","Hello! How can I
assist you?","Hey! What's up?"]
if(intent == "Hii"):
print(random.choice(output))
```

11.5 PENGEMBANGAN CHATBOT MENGGUNAKAN API

Membuat chatbot bukanlah tugas yang mudah. Anda memerlukan ketelitian dan

ketajaman pikiran untuk membangun chatbot yang dapat digunakan dengan baik. Ada dua pendekatan untuk membangun chatbot.

- Pendekatan berbasis aturan
- Pendekatan pembelajaran mesin yang membuat sistem belajar sendiri dengan menyederhanakan data

Beberapa chatbot bersifat dasar, sementara yang lain lebih canggih dengan otak AI. Chatbot yang dapat memahami bahasa alami dan menanggapinya menggunakan otak AI, dan penggemar teknologi memanfaatkan berbagai sumber seperti Api.ai untuk membuat chatbot yang kaya AI tersebut. Programmer memanfaatkan layanan berikut untuk membuat bot:

- Kerangka kerja bot Microsoft
- Wit.ai
- Api.ai
- Watson milik IBM

Penggemar pembuatan bot lainnya dengan keterampilan pemrograman terbatas atau tidak sama sekali memanfaatkan platform pengembangan bot seperti berikut untuk membuat chatbot:

- Chatfuel
- Texit.in
- Octane AI
- Motion.ai

Ada berbagai API yang menganalisis teks. Tiga raksasa utamanya adalah sebagai berikut:

- Cognitive Services milik Microsoft Azure
- Amazon Lex
- IBM Watson

Cognitive Services milik Microsoft Azure

Mari kita mulai dengan Microsoft Azure.

- *Language Understanding Intelligent Service (LUIS):* Layanan ini menyediakan alat sederhana yang memungkinkan Anda membuat model bahasa (maksud/entitas) Anda sendiri yang memungkinkan aplikasi/bot apa pun memahami perintah Anda dan bertindak sesuai dengannya.
- *Text Analytics API*: Layanan ini mengevaluasi sentimen dan topik untuk memahami apa yang diinginkan pengguna.
- *Translator Text API:* Ini mengidentifikasi bahasa secara otomatis dan kemudian menerjemahkannya ke bahasa lain secara real time.
- *Web Language Model API:* Ini memasukkan spasi ke dalam rangkaian kata yang tidak memiliki spasi secara otomatis.
- *Bing Spell Check API:* Ini memungkinkan pengguna untuk mengoreksi kesalahan ejaan; mengenali perbedaan antara nama, nama merek, dan bahasa gaul; dan memahami homofon saat mereka mengetik.
- *Linguistic Analysis API:* Ini memungkinkan Anda untuk mengidentifikasi konsep dan tindakan dalam teks Anda dengan penandaan part-of-speech dan menemukan frasa

dan konsep menggunakan pengurai bahasa alami. Ini sangat berguna untuk menambang umpan balik pelanggan.

Amazon Lex

Amazon Lex adalah layanan untuk membangun antarmuka percakapan ke dalam aplikasi apa pun menggunakan suara dan teks. Sayangnya, tidak ada opsi sinonim, dan tidak ada ekstraksi entitas dan klasifikasi maksud yang tepat.

Berikut ini adalah beberapa manfaat penting menggunakan Amazon Lex:

- Sederhana. Memandu Anda dalam membuat chatbot.
- Memiliki algoritme pembelajaran mendalam. Algoritme seperti NLU dan NLP diimplementasikan untuk chatbot.

Amazon telah memusatkan fungsionalitas ini sehingga dapat digunakan dengan mudah.

- Memiliki fitur penyebaran dan penskalaan yang mudah.
- Memiliki integrasi bawaan dengan platform AWS.
- Hemat biaya.

IBM Watson

IBM menyediakan API IBM Watson untuk membangun chatbot Anda sendiri dengan cepat. Dalam implementasi, pendekatan terhadap perjalanan sama pentingnya dengan perjalanan itu sendiri. Mempelajari AI Percakapan Watson untuk dasar-dasar desain percakapan perusahaan, dan dampaknya pada bisnis Anda, sangat penting dalam merumuskan rencana tindakan yang sukses. Persiapan ini akan memungkinkan Anda untuk berkomunikasi, belajar, dan memantau terhadap suatu standar, yang memungkinkan bisnis Anda untuk membangun proyek yang siap untuk pelanggan dan sukses.

Desain percakapan adalah bagian terpenting dalam membangun chatbot. Hal pertama yang harus dipahami adalah siapa pengguna dan apa yang ingin mereka capai. IBM Watson memiliki banyak teknologi yang dapat Anda integrasikan dengan mudah dalam chatbot Anda; beberapa di antaranya adalah Watson Conversation, Watson Tone Analyzer, speech to text, dan masih banyak lagi.

11.6 PRAKTIK TERBAIK PENGEMBANGAN CHATBOT

Saat membangun chatbot, penting untuk memahami bahwa ada praktik terbaik tertentu yang dapat dimanfaatkan. Ini akan membantu dalam menciptakan bot yang ramah pengguna dan berhasil yang dapat memenuhi tujuannya untuk melakukan percakapan yang lancar dengan pengguna.

Salah satu hal terpenting dalam hubungan ini adalah mengetahui target audiens dengan baik. Berikutnya adalah hal-hal lain seperti mengidentifikasi skenario kasus penggunaan, mengatur nada obrolan, dan mengidentifikasi platform pengiriman pesan.

Dengan mematuhi praktik terbaik berikut, keinginan untuk memastikan percakapan yang lancar dengan pengguna dapat menjadi kenyataan.

Kenali Calon Pengguna

Pemahaman menyeluruh tentang target audiens adalah langkah pertama dalam membangun bot yang sukses. Tahap selanjutnya adalah mengetahui tujuan bot tersebut

dibuat.

Berikut adalah beberapa hal yang perlu diingat:

- Ketahui apa tujuan bot tertentu. Bot tersebut dapat berupa bot untuk menghibur audiens, memfasilitasi pengguna untuk bertransaksi, menyediakan berita, atau berfungsi sebagai saluran layanan pelanggan.
- Jadikan bot lebih ramah pelanggan dengan mempelajari produk pelanggan.

Baca Sentimen Pengguna dan Jadikan Bot Lebih Efektif

Chatbot harus hangat dan ramah seperti manusia agar percakapan menjadi pengalaman yang menyenangkan. Chatbot harus membaca dan memahami sentimen pengguna dengan cerdas untuk mendorong pengguna melanjutkan percakapan. Pengguna akan terdorong untuk kembali lagi jika pengalaman pertama mereka menyenangkan.

Berikut beberapa hal yang perlu diingat:

- Promosikan produk Anda atau jadikan pengguna sebagai duta merek dengan memanfaatkan sentimen positif.
- Segera tanggapi komentar negatif agar percakapan tetap menarik.
- Jika memungkinkan, gunakan bahasa yang ramah agar pengguna merasa berinteraksi dengan manusia yang dikenal.
- Buat pengguna merasa nyaman dengan mengulang masukan dan pastikan mereka memahami semua hal yang dibahas.

BAB 12
DETEKSI DAN PENGENALAN WAJAH

Deteksi wajah adalah proses mendeteksi wajah dalam gambar atau video. Pengenalan wajah adalah proses mendeteksi wajah dalam gambar dan kemudian menggunakan algoritme untuk mengidentifikasi siapa pemilik wajah tersebut. Dengan demikian, pengenalan wajah merupakan bentuk identifikasi orang.

Pertama-tama Anda perlu mengekstraksi fitur dari gambar untuk melatih pengklasifikasi pembelajaran mesin guna mengidentifikasi wajah dalam gambar. Sistem ini tidak hanya bersifat nonsubjektif, tetapi juga otomatis—tidak diperlukan pelabelan fitur wajah secara manual. Anda cukup mengekstraksi fitur dari wajah, melatih pengklasifikasi, lalu menggunakannya untuk mengidentifikasi wajah berikutnya.

Karena untuk pengenalan wajah, pertama-tama Anda perlu mendeteksi wajah dari gambar, Anda dapat menganggap pengenalan wajah sebagai tahap dua fase.

- ***Tahap 1***: Mendeteksi keberadaan wajah dalam aliran gambar atau video menggunakan metode seperti kaskade Haar, HOG + SVM Linear, pembelajaran mendalam, atau algoritme lain yang dapat melokalisasi wajah.
- ***Tahap 2***: Ambil setiap wajah yang terdeteksi selama fase lokalisasi dan pelajari siapa pemilik wajah tersebut—di sinilah Anda benar-benar menetapkan nama pada wajah tersebut.

12.1 DETEKSI WAJAH, PENGENALAN WAJAH, DAN ANALISIS WAJAH

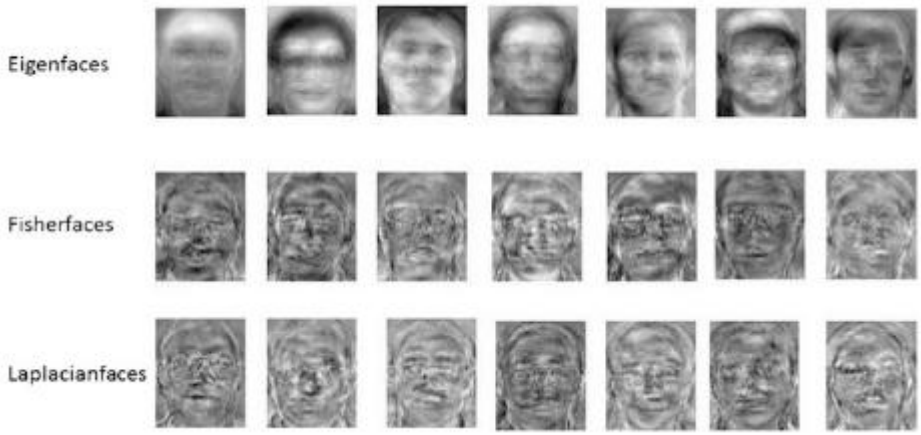
Ada perbedaan antara deteksi wajah, pengenalan wajah, dan analisis wajah.

- *Deteksi wajah:* Ini adalah teknik untuk menemukan semua wajah manusia dalam sebuah gambar.
- *Pengenalan wajah*: Ini adalah langkah berikutnya setelah deteksi wajah. Dalam pengenalan wajah, Anda mengidentifikasi wajah mana yang dimiliki orang mana menggunakan repositori gambar yang ada.
- *Analisis wajah*: Wajah diperiksa, dan beberapa kesimpulan diambil seperti usia, warna kulit, dan sebagainya.

OpenCV

OpenCV menyediakan tiga metode untuk pengenalan wajah (lihat Gambar 12-1):

- Eigenfaces
- Histogram pola biner lokal (LBPH)
- Fisherfaces

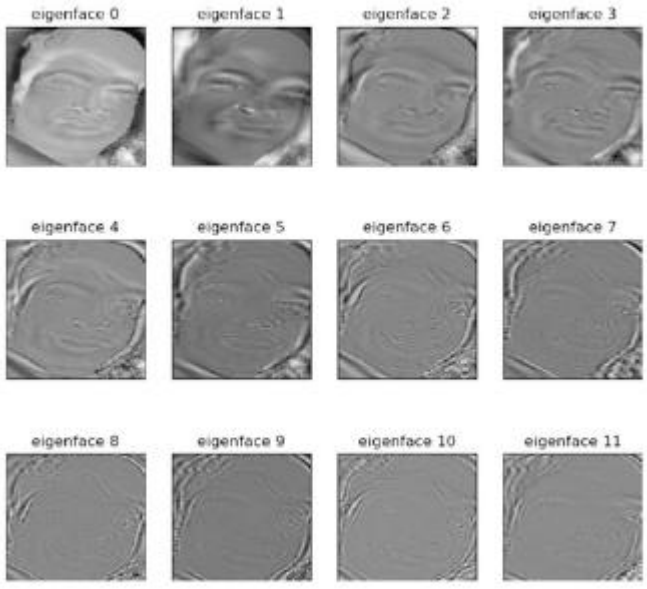


Gambar 12-1. Menerapkan metode OpenCV pada wajah

Ketiga metode mengenali wajah dengan membandingkan wajah tersebut dengan beberapa set pelatihan wajah yang dikenal. Untuk pelatihan, Anda menyediakan wajah pada algoritme dan melabelinya dengan orang yang menjadi pemiliknya. Saat Anda menggunakan algoritme untuk mengenali wajah yang tidak dikenal, algoritme tersebut menggunakan model yang dilatih pada set pelatihan untuk melakukan pengenalan. Masing-masing dari tiga metode yang disebutkan di atas menggunakan set pelatihan dengan cara yang sedikit berbeda.
Wajah Laplacian dapat menjadi cara lain untuk mengenali wajah.

Eigenfaces

Algoritme eigenfaces menggunakan analisis komponen utama untuk membangun representasi gambar wajah berdimensi rendah, yang akan Anda gunakan sebagai fitur untuk gambar wajah yang sesuai (Gambar 12-2).



Gambar 12-2. Menerapkan dekomposisi nilai eigen dan mengekstraksi 11 eigenface dengan magnitudo terbesar

Untuk ini, Anda mengumpulkan kumpulan data wajah dengan beberapa gambar wajah dari setiap orang yang ingin Anda kenali—ini seperti memiliki beberapa contoh pelatihan dari kelas gambar yang ingin Anda beri label dalam klasifikasi gambar. Dengan kumpulan data gambar wajah ini, yang diasumsikan memiliki lebar dan tinggi yang sama dan idealnya dengan mata dan struktur wajah yang sejajar pada koordinat (x, y) yang sama, Anda menerapkan dekomposisi nilai eigen dari kumpulan data tersebut, dengan tetap mempertahankan vektor eigen dengan nilai eigen terkait yang terbesar.

Dengan vektor eigen ini, wajah kemudian dapat direpresentasikan sebagai kombinasi linier dari apa yang disebut Kirby dan Sirovich sebagai eigenface. Algoritme eigenface melihat seluruh kumpulan data.

LBPH

Anda dapat menganalisis setiap gambar secara independen di LBPH. Metode LBPH agak lebih sederhana, dalam artian Anda mengkarakterisasi setiap gambar dalam set data secara lokal; ketika gambar baru yang tidak dikenal diberikan, Anda melakukan analisis yang sama pada gambar tersebut dan membandingkan hasilnya dengan setiap gambar dalam set data. Cara Anda menganalisis gambar adalah dengan mengkarakterisasi pola lokal di setiap lokasi dalam gambar.

Sementara algoritma eigenfaces bergantung pada PCA untuk membangun representasi gambar wajah berdimensi rendah, metode pola biner lokal (LBP) bergantung pada, seperti namanya, ekstraksi fitur.

Pertama kali diperkenalkan oleh Ahonen dkk. dalam makalah tahun 2006 “Pengenalan Wajah dengan Pola Biner Lokal,” metode ini menyarankan untuk membagi gambar wajah menjadi kisi 7x7 dengan sel berukuran sama (Gambar 12-3).

Gambar 12-3. Menerapkan LBPH untuk pengenalan wajah dimulai dengan membagi gambar wajah menjadi kisi 7x7 dengan sel berukuran sama

Anda kemudian mengekstrak histogram pola biner lokal dari masing-masing 49 sel. Dengan membagi gambar menjadi sel, Anda memperkenalkan lokalitas ke dalam vektor fitur akhir. Lebih jauh, sel-sel di bagian tengah memiliki bobot lebih besar sehingga berkontribusi lebih banyak pada representasi keseluruhan. Sel-sel di sudut membawa lebih sedikit informasi wajah yang dapat diidentifikasi dibandingkan dengan sel-sel di bagian tengah kisi (yang berisi

struktur mata, hidung, dan bibir). Terakhir, Anda menggabungkan histogram LBP berbobot ini dari 49 sel untuk membentuk vektor fitur akhir Anda.

Fisherfaces

Principal Component Analysis (PCA), yang merupakan inti dari metode Eigenfaces, menemukan kombinasi linear fitur yang memaksimalkan varians total dalam data. Meskipun ini jelas merupakan cara yang ampuh untuk merepresentasikan data, metode ini tidak mempertimbangkan kelas apa pun sehingga banyak informasi diskriminatif yang mungkin hilang saat membuang komponen.

Bayangkan situasi di mana varians dalam data Anda dihasilkan oleh sumber eksternal, biarkan itu menjadi cahaya. Komponen yang diidentifikasi oleh PCA tidak selalu mengandung informasi diskriminatif sama sekali, sehingga sampel yang diproyeksikan tercampur dan klasifikasi menjadi tidak mungkin.

Analisis Diskriminan Linier melakukan reduksi dimensionalitas khusus kelas dan ditemukan oleh ahli statistika hebat Sir R. A. Fisher. Penggunaan beberapa pengukuran dalam masalah taksonomi. Untuk menemukan kombinasi fitur yang paling baik memisahkan antara kelas, Analisis Diskriminan Linier memaksimalkan rasio sebaran antarkelas dengan dalam kelas, alih-alih memaksimalkan sebaran keseluruhan. Idenya sederhana: kelas yang sama harus mengelompok rapat, sementara kelas yang berbeda berada sejauh mungkin dari satu sama lain dalam representasi dimensi yang lebih rendah.

12.2 MENDETEKSI WAJAH

Fitur pertama yang Anda perlukan untuk melakukan pengenalan wajah adalah mendeteksi di mana wajah berada dalam gambar saat ini. Dalam Python, Anda dapat menggunakan filter Haar cascade dari pustaka OpenCV untuk melakukannya secara efisien.

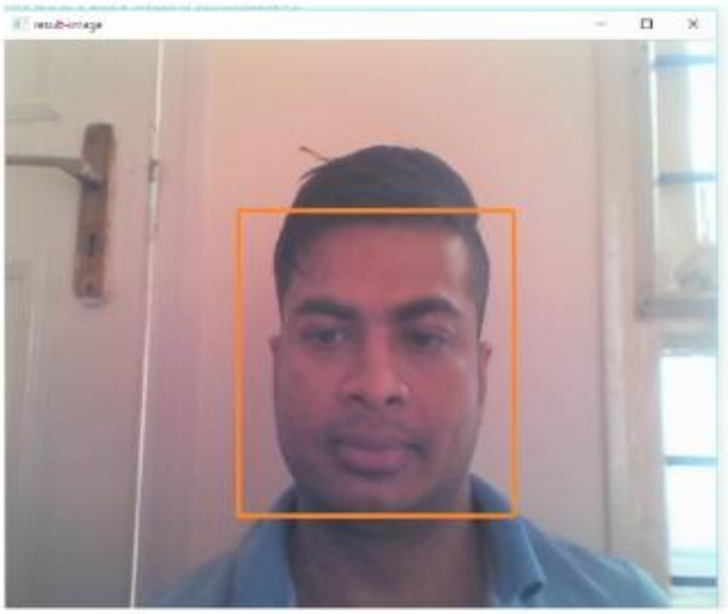
Untuk implementasi yang ditunjukkan di sini, saya menggunakan Anaconda dengan Python 3.5, OpenCV 3.1.0, dan dlib 19.1.0. Untuk menggunakan kode berikut, pastikan Anda memiliki versi ini (atau yang lebih baru).

Untuk melakukan deteksi wajah, beberapa inisialisasi harus dilakukan, seperti yang ditunjukkan di sini:

```
# Import the OpenCV library
import cv2
# Initialize a face cascade using the frontal face haar cascade provided
# with the OpenCV2 library. This will be required for face detection in an
# image.
faceCascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
# The desired output width and height, can be modified according to the needs.
OUTPUT_SIZE_WIDTH = 700
OUTPUT_SIZE_HEIGHT = 600

# Open the first webcam device
capture = cv2.VideoCapture(0)

# Create two opencv named windows for shoving the input, output images.
cv2.namedWindow("base-image", cv2.WINDOW_AUTOSIZE)
cv2.namedWindow("result-image", cv2.WINDOW_AUTOSIZE)
```

```
# Position the windows next to each other
cv2.moveWindow("base-image", 20, 200)
cv2.moveWindow("result-image", 640, 200)

# Start the window thread for the two windows we are using
cv2.startWindowThread()

rectangleColor = (0, 100, 255)
```

Sisa kode akan menjadi loop tak terbatas yang terus-menerus mengambil gambar terbaru dari webcam, mendeteksi semua wajah dalam gambar yang diambil, menggambar persegi panjang di sekitar wajah terbesar yang terdeteksi, dan kemudian akhirnya menunjukkan gambar input dan output dalam sebuah jendela (Gambar 12-4).



Gambar 12-4. Contoh keluaran yang menunjukkan wajah yang terdeteksi

Anda dapat melakukannya dengan kode berikut dalam loop tak terbatas:

```
# Retrieve the latest image from the webcam
rc,fullSizeBaseImage = capture.read()
# Resize the image to 520x420
baseImage= cv2.resize(fullSizeBaseImage, (520, 420))

# Check if a key was pressed and if it was Q or q, then destroy all
# opencv windows and exit the application, stopping the infinite loop.
pressedKey = cv2.waitKey(2)
if (pressedKey == ord('Q')) | (pressedKey == ord('q')):
cv2.destroyAllWindows()
exit(0)
# Result image is the image we will show the user, which is a
# combination of the original image captured from the webcam with the
# overlayed rectangle detecting the largest face
resultImage = baseImage.copy()

# We will be using gray colored image for face detection.
# So we need to convert the baseImage captured by webcam to a gray-based image
gray_image = cv2.cvtColor(baseImage, cv2.COLOR_BGR2GRAY)
# Now use the haar cascade detector to find all faces in the
# image
faces = faceCascade.detectMultiScale(gray_image, 1.3, 5)
```

```
# As we are only interested in the 'largest' face, we need to
# calculate the largest area of the found rectangle.
# For this, first initialize the required variables to 0.
maxArea = 0
x = 0
y = 0
w = 0
h = 0

# Loop over all faces found in the image and check if the area for this face is
# the largest so far
for(_x, _y, _w, _h) in faces:
if _w * _h > maxArea:
        x = _x
        y = _y
        w = _w
        h = _h
    maxArea = w * h

# If any face is found, draw a rectangle around the
#  largest face present in the picture
If maxArea > 0:
cv2.rectangle(resultImage, (x-10, y-20),
(x + w+10, y + h+20), rectangleColor, 2)
# Since we want to show something larger on the screen than the
# original 520x420, we resize the image again

# Note that it would also be possible to keep the large version
# of the baseimage and make the result image a copy of this large
# base image and use the scaling factor to draw the rectangle
# at the right coordinates.
largeResult = cv2.resize(resultImage,
(OUTPUT_SIZE_WIDTH, OUTPUT_SIZE_HEIGHT))
# Finally, we show the images on the screen
cv2.imshow("base-image", baseImage)
cv2.imshow("result-image", largeResult)
```

Melacak Wajah

Kode sebelumnya untuk deteksi wajah memiliki beberapa kekurangan.

- Kode tersebut mungkin memerlukan komputasi yang mahal.
- Jika orang yang terdeteksi menoleh sedikit, kaskade Haar mungkin tidak mendeteksi wajah.
- Sulit untuk melacak wajah di antara bingkai.

Pendekatan yang lebih baik untuk ini adalah dengan melakukan deteksi wajah sekali dan kemudian memanfaatkan pelacak korelasi dari pustaka dlib yang luar biasa untuk melacak wajah dari bingkai ke bingkai. Agar ini berfungsi, Anda perlu mengimpor pustaka lain dan menginisialisasi variabel tambahan.

```
import dlib

# Create the tracker we will use to recognize face in different frames
# we get from the webcam
tracker = dlib.correlation_tracker()

# The Boolean variable we use to keep track whether we are
# using dlib tracker, or not.
trackingFace = 0
```

Dalam loop for tak terbatas, Anda sekarang akan menentukan apakah pelacak korelasi dlib saat ini melacak suatu wilayah dalam gambar. Jika tidak demikian, Anda akan menggunakan kode yang sama seperti sebelumnya untuk menemukan sisi terbesar, tetapi alih-alih menggambar persegi panjang, Anda menggunakan koordinat yang ditemukan untuk menginisialisasi pelacak korelasi.

```
# If we are not tracking a face, then try to detect one using the above code itself.
if not trackingFace:

# We will be using gray colored image for face detection.
# So we need to convert the baseImage captured by webcam to a gray-based image
gray = cv2.cvtColor(baseImage, cv2.COLOR_BGR2GRAY)
# Now use the haar cascade detector to find all faces
# in the image
faces=faceCascade.detectMultiScale(gray,1.3,5)

# In the console we can show our this case of using the
# detector for a face, when we are detecting it for first time.
print("Using the cascade detector to detect face")


# As we are only interested in the 'largest' face, we need to
# calculate the largest area of the found rectangle.
# For this, first initialize the required variables to 0.
maxArea = 0
x = 0
y = 0
w = 0
h =0
```

```
# Loop over all faces and check if the area for this
# face is the largest so far
# We need to convert it to int here because dlib tracker
# needs an int as its argument. If we omit the cast to
# int here, you will get cast errors since the detector
# returns numpy.int32 and the tracker requires an int
for(_x, _y, _w, _h) in faces:
if _w * _h > maxArea:
            x = int(_x)
            y = int(_y)
            w = int(_w)
            h = int(_h)
    maxArea = w * h

# If any face is found, draw a rectangle around the
# largest face present in the picture
if maxArea > 0:

#Initialize the tracker
tracker.start_track(baseImage,
dlib.rectangle(x-10, y-20, x+w+10, y+h+20))

# Set the indicator variable such that we know the
# tracker is tracking a face in the image
trackingFace = 1
```

Sekarang bagian terakhir dalam loop tak terbatas adalah untuk memeriksa lagi apakah pelacak korelasi secara aktif melacak wajah (misalnya, apakah pelacak tersebut baru saja mendeteksi wajah dengan kode sebelumnya, trankingFace=1?). Jika pelacak secara aktif melacak wajah dalam gambar, Anda akan memperbarui pelacak tersebut. Bergantung pada kualitas pembaruan (misalnya, seberapa yakin pelacak tersebut tentang apakah pelacak tersebut masih melacak wajah yang sama), Anda dapat menggambar persegi panjang di sekitar wilayah yang ditunjukkan oleh pelacak atau menunjukkan bahwa Anda tidak lagi melacak wajah.

```
# Check if the tracker is actively tracking a face in the image
if trackingFace:

# Update the tracker and request information about the
# quality of the tracking update
trackingQuality = tracker.update(baseImage)

# If the tracking quality is good enough, determine the
# updated position of the tracked region and draw the
# rectangle
If trackingQuality >= 9.0:
tracked_position = tracker.get_position()
```

```
t_x = int(tracked_position.left())
t_y = int(tracked_position.top())
t_w = int(tracked_position.width())
t_h = int(tracked_position.height())
cv2.rectangle(resultImage, (t_x,t_y),
(t_x+t_w, t_y+t_h),
rectangleColor, 2)

else:
# If the quality of the tracking update is not good enough
# for us  (e.g. the face being tracked moved out of the
# screen) we stop the tracking of the face and in the
# next loop we will find the largest face in the image
# again
trackingFace = 0
```

Seperti yang dapat Anda lihat dalam kode, Anda mencetak pesan ke konsol setiap kali Anda menggunakan detektor lagi. Jika Anda melihat output konsol saat menjalankan aplikasi ini, Anda akan melihat bahwa meskipun Anda bergerak cukup banyak di layar, pelacak cukup baik dalam mengikuti wajah setelah terdeteksi.

12.3 PENGENALAN WAJAH

Sistem pengenalan wajah mengidentifikasi nama orang yang ada dalam bingkai video dengan mencocokkan wajah di setiap bingkai video dengan gambar yang dilatih dan mengembalikan (dan menulis dalam file CSV) label jika wajah dalam bingkai berhasil dicocokkan. Sekarang Anda akan melihat cara membuat sistem pengenalan wajah langkah demi langkah.

Pertama, Anda mengimpor semua pustaka yang diperlukan. face_recognition adalah pustaka sederhana yang dibuat menggunakan pengenalan wajah canggih dlib yang juga dibuat dengan pembelajaran mendalam.

```
import os
import re
import warnings
import scipy.misc
import cv2
import face_recognition
from PIL import Image
import argparse
import csv
import os
```

Argparse adalah pustaka Python yang memungkinkan Anda menambahkan argumen Anda sendiri ke file; kemudian dapat digunakan untuk memasukkan direktori gambar atau jalur file apa pun pada saat eksekusi.

```
parser = argparse.ArgumentParser()
parser.add_argument("-i", "--images-dir", help="image dir")
parser.add_argument("-v", "--video", help="video to recognize faces on")
parser.add_argument("-o", "--output-csv", help="Ouput csv file [Optional]")
parser.add_argument("-u", "--upsample-rate", help="How many times to upsample the image looking for faces. Higher numbers
                    find smaller faces. [Optional]")
args = vars(parser.parse_args())
```

Dalam kode sebelumnya, saat menjalankan berkas Python ini, Anda harus menentukan hal berikut: direktori gambar masukan pelatihan, berkas video yang akan digunakan sebagai kumpulan data, dan berkas CSV keluaran untuk menulis keluaran pada setiap kerangka waktu.

```
#Check if argument values are valid
if args.get("images_dir", None) is None and os.path.exists(args.get("images_dir", None)):
    print("Please check the path to images folder")
    exit()
if args.get("video", None) is None and os.path.isfile(args.get("video", None)):
    print("Please check the path to video")
    exit()
if args.get("output_csv", None) is None:
    print("You haven't specified an output csv file. Nothing will be written.")
# By default upsample rate = 1
upsample_rate = args.get("upsample_rate", None)
if upsample_rate is None:
    upsample_rate = 1
```

```
# Helper functions
def image_files_in_folder(folder):
    return [os.path.join(folder, f) for f in os.listdir(folder) if re.match(r'.*\.(jpg|jpeg|png)', f, flags=re.I)]
```

Dengan menggunakan fungsi sebelumnya, semua berkas gambar dari folder yang ditentukan dapat dibaca.
Fungsi berikut menguji bingkai input dengan gambar pelatihan yang diketahui:

```
def test_image(image_to_check, known_names, known_face_encodings, number_of_times_to_upsample=1):
    """
    Test if any face is recognized in unknown image by checking known images
    :paramimage_to_check: Numpy array of the image
    :paramknown_names: List containing known labels
    :paramknown_face_encodings: List containing training image labels
    :paramnumber_of_times_to_upsample: How many times to upsample the image looking for
    faces. Higher numbers find smaller faces.
    :return: A list of labels of known names
    """
    # unknown_image = face_recognition.load_image_file(image_to_check)
    unknown_image = image_to_check
    # Scale down the image to make it run faster
    if unknown_image.shape[1] > 1600:
        scale_factor = 1600 / unknown_image.shape[1]
        with warnings.catch_warnings():
            warnings.simplefilter("ignore")
        unknown_image = scipy.misc.imresize(unknown_image, scale_factor)
    face_locations = face_recognition.face_locations(unknown_image,  number_of_times_to_upsample)
    unknown_encodings = face_recognition.face_encodings(unknown_image, face_locations)
    result = []
    for unknown_encoding in unknown_encodings:
        result = face_recognition.compare_faces(known_face_encodings, unknown_encoding)
    result_encoding = []
    for nameIndex, is_match in enumerate(result):
        if is_match:
            result_encoding.append(known_names[nameIndex])

    return result_encoding
```

Sekarang Anda mendefinisikan fungsi untuk mengekstrak label untuk gambar yang cocok dan diketahui.

```
def map_file_pattern_to_label(labels_with_pattern, labels_list):
    """
    Map file name pattern to full label
    :paramlabels_with_pattern: dict : { "file_name_pattern": "full_label" }
    :paramlabels_list: list : list of labels of file names got from test_image()
    :return: list of full labels
    """
    result_list = []
    for key, label in labels_with_pattern.items():
        for img_labels in labels_list:
            if str(key).lower() in str(img_labels).lower():
                if str(label) not in result_list:
                    result_list.append(str(label))
                # continue
    # result_list = [label for key, label in labels_with_pattern if str(key).lower() in labels_list]
    return result_list
```

Baca video masukan untuk mengekstrak bingkai uji.

```
cap = cv2.VideoCapture(args["video"])

#get the training images
training_encodings = []
training_labels = []
for file in image_files_in_folder(args['images_dir']):
    basename = os.path.splitext(os.path.basename(file))[0]
    img = face_recognition.load_image_file(file)
    encodings = face_recognition.face_encodings(img)

    if len(encodings) > 1:
        print("WARNING: More than one face found in {}. Only considering the first face.".format(file))

    if len(encodings) == 0:
        print("WARNING: No faces found in {}. Ignoring file.".format(file))
    if len(encodings):
        training_labels.append(basename)
        training_encodings.append(encodings[0])


csvfile = None
csvwriter = None
if args.get("output_csv", None) is not None:
    csvfile = open(args.get("output_csv"), 'w')
    csvwriter = csv.writer(csvfile, delimiter=',', quotechar='|', quoting=csv.QUOTE_MINIMAL)

ret, firstFrame = cap.read()
frameRate = cap.get(cv2.CAP_PROP_FPS)
```

Sekarang tentukan label set pelatihan Anda. Lalu cocokkan frame yang diekstrak dari video input yang diberikan untuk mendapatkan hasil yang diinginkan.

```
# Labels with file pattern, edit this
label_pattern = {
    "shah": "Shahrukh Khan",
    "amir": "Amir Khan"
            }

# match each frame in video with our trained set of labeled images
while ret:
    curr_frame = cap.get(1)

    ret, frame = cap.read()

    result = test_image(frame, training_labels, training_encodings, upsample_rate)
    labels = map_file_pattern_to_label(label_pattern, result)
    curr_time = curr_frame / frameRate
    print("Time: {} faces: {}".format(curr_time, labels))
    if csvwriter:
        csvwriter.writerow([curr_time, labels])
    cv2.imshow('frame', frame)

    key = cv2.waitKey(1) & 0xFF
    if key == ord('q'):
        break
if csvfile:
    csvfile.close()
cap.release()
cv2.destroyAllWindows()
```

Pengenalan Wajah Berbasis Pembelajaran Mendalam

Impor paket yang diperlukan.

```
import cv2                  # working with, mainly resizing, images
import numpy as np          # dealing with arrays
import os                   # dealing with directories
from random import shuffle # mixing up or currently ordered data that might lead our network astray in training.
from tqdm import tqdm
from scipy import misc
import tflearn
from tflearn.layers.conv import conv_2d, max_pool_2d
from tflearn.layers.core import input_data, dropout, fully_connected
from tflearn.layers.estimator import regression
import tensorflow as tf
import glob
import matplotlib.pyplot as plt
import dlib
```

Inisialisasi variabel.

```
from skimage import io
tf.reset_default_graph()
TRAIN_DIR ='resize_a/train'
TEST_DIR ='resize_a/test'
IMG_SIZE = 200
boxScale=1
LR = 1e-3
MODEL_NAME = 'quickest.model'.format(LR, '2conv-basic')
```

Fungsi `label_img()` digunakan untuk membuat array label, dan fungsi `detect_faces()` mendeteksi bagian wajah dalam gambar.

```
def label_img(img):
    word = img.split('(')[-2]
    word_label = word[0]
    if word_label == 'R': return [1,0]

    elif word_label == 'A': return [0,1]

def detect_faces(image):

    # Create a face detector
    face_detector = dlib.get_frontal_face_detector()

    # Run detector and get bounding boxes of the faces on image.
    detected_faces = face_detector(image, 1)
    face_frames = [(x.left(), x.top(),
                    x.right(), x.bottom()) for x in detected_faces]

    return face_frames
```

Fungsi `create_train_data()` digunakan untuk praproses data pelatihan.

```
def create_train_data():
    training_data = []
    for img in tqdm(os.listdir(TRAIN_DIR)):
        label = label_img(img)
        path = os.path.join(TRAIN_DIR,img)
        img= misc.imread(path)
        img = cv2.imread(path,cv2.IMREAD_GRAYSCALE)
        img = cv2.resize(img, (IMG_SIZE,IMG_SIZE))
        detected_faces = detect_faces(img)
        for n, face_rect in enumerate(detected_faces):
            img = Image.fromarray(img).crop(face_rect)
            img = np.array(img)
        img = cv2.resize(img, (IMG_SIZE,IMG_SIZE))
# If any face is found, draw a rectangle around the
#  largest face present in the picture

        training_data.append([np.array(img),np.array(label)])
    shuffle(training_data)
    np.save('train_data.npy', training_data)
    return training_data
```

Fungsi process_test_data() digunakan untuk memproses awal data pengujian.

```
def create_train_data():
    training_data = []
    for img in tqdm(os.listdir(TRAIN_DIR)):
        label = label_img(img)
        path = os.path.join(TRAIN_DIR,img)
        img= misc.imread(path)
        img = cv2.imread(path,cv2.IMREAD_GRAYSCALE)
        img = cv2.resize(img, (IMG_SIZE,IMG_SIZE))
        detected_faces = detect_faces(img)
        for n, face_rect in enumerate(detected_faces):
            img = Image.fromarray(img).crop(face_rect)
            img = np.array(img)
        img = cv2.resize(img, (IMG_SIZE,IMG_SIZE))
# If any face is found, draw a rectangle around the
#  largest face present in the picture

        training_data.append([np.array(img),np.array(label)])
    shuffle(training_data)
    np.save('train_data.npy', training_data)
    return training_data
```

Lalu Anda membuat model dan memasukkan data pelatihan ke dalam model.

```
def process_test_data():
    testing_data = []
    for img in tqdm(os.listdir(TEST_DIR)):
        path = os.path.join(TEST_DIR,img)
        imgnum = img.split('.')[-2]
        img_num=get_num(imgnum)
        img= misc.imread(path)
        img = cv2.imread(path,cv2.IMREAD_GRAYSCALE)
        img = cv2.resize(img, (IMG_SIZE,IMG_SIZE))
        detected_faces = detect_faces(img)
        for n, face_rect in enumerate(detected_faces):
            img = Image.fromarray(img).crop(face_rect)
            img = np.array(img)
        img = cv2.resize(img, (IMG_SIZE,IMG_SIZE))
# If any face is found, draw a rectangle around the
#  largest face present in the picture

        testing_data.append([np.array(img), img_num])
```

Terakhir, Anda menyiapkan data uji dan memprediksi output.

```
test_data = process_test_data()

fig=plt.figure()

for num,data in enumerate(test_data[:12]):


    img_num = data[1]
    img_data = data[0]
    y = fig.add_subplot(3,4,num+1)
    orig = img_data
    data = img_data.reshape(IMG_SIZE,IMG_SIZE,1)
    #model_out = model.predict([data])[0]
    model_out = model.predict([data])[0]

    if np.argmax(model_out) == 0: str_label='Ronaldo'
    elif np.argmax(model_out) == 1: str_label='amitabh'

    y.imshow(orig,cmap='gray')
    plt.title(str_label)
    y.axes.get_xaxis().set_visible(False)
    y.axes.get_yaxis().set_visible(False)
plt.show()
```

12.4 TRANSFER LEARNING

Pembelajaran transfer memanfaatkan pengetahuan yang diperoleh saat memecahkan satu masalah dan menerapkannya pada masalah lain yang terkait. Di sini Anda akan melihat cara menggunakan jaringan saraf dalam yang telah dilatih sebelumnya yang disebut model Inception v3 untuk mengklasifikasikan gambar.

Model Inception cukup mampu mengekstrak informasi yang berguna dari suatu gambar.

Mengapa Transfer Learning?

Sudah diketahui bahwa jaringan konvolusional memerlukan sejumlah besar data dan sumber daya untuk dilatih. Sudah menjadi norma bagi para peneliti dan praktisi untuk menggunakan pembelajaran transfer dan penyempurnaan (yaitu, mentransfer bobot jaringan yang dilatih pada proyek sebelumnya seperti ImageNet ke tugas baru).

Anda dapat mengambil dua pendekatan.

- *Pembelajaran transfer:* Anda dapat mengambil CNN yang telah dilatih sebelumnya di ImageNet, menghapus lapisan terakhir yang terhubung sepenuhnya, lalu memperlakukan sisa CNN sebagai ekstraktor fitur untuk set data baru. Setelah Anda mengekstrak fitur untuk semua gambar, Anda melatih pengklasifikasi untuk set data baru.
- *Penyetelan halus:* Anda dapat mengganti dan melatih ulang pengklasifikasi di atas CNN dan juga menyempurnakan bobot jaringan yang telah dilatih sebelumnya melalui backpropagation.

Contoh Pembelajaran Transfer

Dalam contoh ini, pertama-tama Anda akan mencoba mengklasifikasikan gambar dengan langsung memuat model Inception v3.

Impor semua pustaka yang diperlukan.

```
%matplotlib inline
import matplotlib.pyplot as plt
import tensorflow as tf
import numpy as np
import os

# Functions and classes for loading and using the Inception model.
import inception
```

Sekarang tentukan direktori penyimpanan untuk model tersebut lalu unduh model Inception v3.

```
inception.data_dir = 'D:/'
```

```
inception.maybe_download()
```

Muat model yang telah dilatih sebelumnya dan tentukan fungsi untuk mengklasifikasikan gambar yang diberikan.

```
model = inception.Inception()
```

```
def classify(image_path):
    # Display the image.
    p = Image.open(image_path)
    p.show()

    # Use the Inception model to classify the image.
    pred = model.classify(image_path=image_path)

    # Print the scores and names for the top-10 predictions.
    model.print_scores(pred=pred, k=10, only_first_name=True)
```

Sekarang modelnya sudah didefinisikan, mari kita periksa beberapa gambar.

Ini memberikan hasil yang benar sebesar 91,11 persen, tetapi sekarang jika Anda memeriksa orang tertentu, berikut hasilnya:

Ini 48,50 persen bola tenis!

Sayangnya, model Inception tampaknya tidak dapat mengklasifikasikan gambar orang. Alasannya adalah kumpulan data yang digunakan untuk melatih model Inception, yang memiliki beberapa label teks yang membingungkan untuk kelas.

Anda dapat menggunakan kembali model Inception yang telah dilatih sebelumnya dan cukup mengganti lapisan yang melakukan klasifikasi akhir. Ini disebut pembelajaran transfer.

Pertama-tama Anda memasukkan dan memproses gambar dengan model Inception. Tepat sebelum lapisan klasifikasi akhir model Inception, Anda menyimpan apa yang disebut nilai transfer ke file cache.

Alasan menggunakan file cache adalah karena butuh waktu lama untuk memproses gambar dengan model Inception. Ketika semua gambar dalam kumpulan data baru telah diproses melalui model Inception dan nilai transfer yang dihasilkan disimpan ke file cache, maka Anda dapat menggunakan nilai transfer tersebut sebagai input ke jaringan saraf lainnya. Anda kemudian akan melatih jaringan saraf kedua menggunakan kelas dari kumpulan data baru, sehingga jaringan mempelajari cara mengklasifikasikan gambar berdasarkan nilai transfer dari model Inception. Dengan cara ini, model Inception digunakan untuk mengekstrak informasi yang berguna dari gambar, dan jaringan saraf lainnya kemudian digunakan untuk klasifikasi yang sebenarnya.

Hitung Nilai Transfer

Impor fungsi transfer_value_cache dari file Inception.

```
from inception import transfer_values_cache
```

```
file_path_cache_train = os.path.join(cifar10.data_path, 'inception_cifar10_train.pkl')
file_path_cache_test = os.path.join(cifar10.data_path, 'inception_cifar10_test.pkl')
```

```
print("Processing Inception transfer-values for training-images ...")

# Scale images because Inception needs pixels to be between 0 and 255,
# while the CIFAR-10 functions return pixels between 0.0 and 1.0
images_scaled = images_train * 255.0

# If transfer-values have already been calculated then reload them,
# otherwise calculate them and save them to a cache-file.
transfer_values_train = transfer_values_cache(cache_path=file_path_cache_train,
                                              images=images_scaled,
                                              model=model)
```

```
Processing Inception transfer-values for training-images ...
- Processing image:   1021 / 50000
```

```
print("Processing Inception transfer-values for test-images ...")

# Scale images because Inception needs pixels to be between 0 and 255,
# while the CIFAR-10 functions return pixels between 0.0 and 1.0
images_scaled = images_test * 255.0

# If transfer-values have already been calculated then reload them,
# otherwise calculate them and save them to a cache-file.
transfer_values_test = transfer_values_cache(cache_path=file_path_cache_test,
                                             images=images_scaled,
                                             model=model)
```

Saat ini, nilai transfer disimpan dalam berkas cache. Sekarang Anda akan membuat jaringan saraf baru.

Tentukan jaringannya.

```
# Wrap the transfer-values as a Pretty Tensor object.
x_pretty = pt.wrap(x)

with pt.defaults_scope(activation_fn=tf.nn.relu):
    y_pred, loss = x_pretty.\
        fully_connected(size=1024, name='layer_fc1').\
        softmax_classifier(num_classes=num_classes, labels=y_true)
```

Berikut adalah metode optimasinya:

```
global_step = tf.Variable(initial_value=0,
                          name='global_step', trainable=False)
optimizer = tf.train.AdamOptimizer(learning_rate=1e-4).minimize(loss, global_step)
```

Berikut akurasi klasifikasinya:

```
y_pred_cls = tf.argmax(y_pred, dimension=1)
correct_prediction = tf.equal(y_pred_cls, y_true_cls)
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
```

Berikut hasil TensorFlow:

```
session = tf.Session()
session.run(tf.global_variables_initializer())
```

Berikut adalah fungsi pembantu untuk melakukan pelatihan batch:

```
def random_batch():
    # Number of images (transfer-values) in the training-set.
    num_images = len(transfer_values_train)

    # Create a random index.
    idx = np.random.choice(num_images,
                           size=train_batch_size,
                           replace=False)

    # Use the random index to select random x and y-values.
    # We use the transfer-values instead of images as x-values.
    x_batch = transfer_values_train[idx]
    y_batch = labels_train[idx]

    return x_batch, y_batch
```

Untuk mengoptimalkan, berikut adalah kodenya:

```
def optimize(num_iterations):
    # Number of images (transfer-values) in the training-set.


    start_time = time.time()

    for i in range(num_iterations):
        # Get a batch of training examples.
        # x_batch now holds a batch of images (transfer-values) and
        # y_true_batch are the true labels for those images.
        x_batch, y_true_batch = random_batch()

        # Put the batch into a dict with the proper names
        # for placeholder variables in the TensorFlow graph.
        feed_dict_train = {x: x_batch,
                           y_true: y_true_batch}

        # Run the optimizer using this batch of training data.
        # TensorFlow assigns the variables in feed_dict_train
        # to the placeholder variables and then runs the optimizer.
        # We also want to retrieve the global_step counter.
        i_global, _ = session.run([global_step, optimizer],
                                  feed_dict=feed_dict_train)

        # Print status to screen every 100 iterations (and last).
        if (i_global % 100 == 0) or (i == num_iterations - 1):
            # Calculate the accuracy on the training-batch.
            batch_acc = session.run(accuracy,
                                    feed_dict=feed_dict_train)
            # Print status.
            msg = "Global Step: {0:>6}, Training Batch Accuracy: {1:>6.1%}"
            print(msg.format(i_global, batch_acc))

    # Ending time.
    end_time = time.time()

    # Difference between start and end-times.
    time_dif = end_time - start_time

    # Print the time-usage.
    print("Time usage: " + str(timedelta(seconds=int(round(time_dif)))))

    # Use the random index to select random x and y-values.
    # We use the transfer-values instead of images as x-values.
    x_batch = transfer_values_train[idx]
    y_batch = labels_train[idx]

    return x_batch, y_batch
```

Untuk memplot matriks kebingungan, berikut kodenya:

```
from sklearn.metrics import confusion_matrix

def plot_confusion_matrix(cls_pred):
    # This is called from print_test_accuracy() below.

    # cls_pred is an array of the predicted class-number for
    # all images in the test-set.

    # Get the confusion matrix using sklearn.
    cm = confusion_matrix(y_true=cls_test,  # True class for test-set.
                          y_pred=cls_pred)  # Predicted class.

    # Print the confusion matrix as text.
    for i in range(num_classes):
        # Append the class-name to each line.
        class_name = "({}) {}".format(i, class_names[i])
        print(cm[i, :], class_name)

    # Print the class-numbers for easy reference.
    class_numbers = [" ({0})".format(i) for i in range(num_classes)]
    print("".join(class_numbers))
```

Berikut adalah fungsi pembantu untuk menghitung klasifikasi:

```
# Split the data-set in batches of this size to limit RAM usage.
batch_size = 256

def predict_cls(transfer_values, labels, cls_true):
    # Number of images.
    num_images = len(transfer_values)

    # Allocate an array for the predicted classes which
    # will be calculated in batches and filled into this array.
    cls_pred = np.zeros(shape=num_images, dtype=np.int)

    # Now calculate the predicted classes for the batches.
    # We will just iterate through all the batches.
    # There might be a more clever and Pythonic way of doing this.

    # The starting index for the next batch is denoted i.
    i = 0

    while i < num_images:
        # The ending index for the next batch is denoted j.
        j = min(i + batch_size, num_images)

        # Create a feed-dict with the images and labels
        # between index i and j.
        feed_dict = {x: transfer_values[i:j],
                     y true: labels[i:j]}
```

```
        # Calculate the predicted class using TensorFlow.
        cls_pred[i:j] = session.run(y_pred_cls, feed_dict=feed_dict)

        # Set the start-index for the next batch to the
        # end-index of the current batch.
        i = j

    # Create a boolean array whether each image is correctly classified.
    correct = (cls_true == cls_pred)

    return correct, cls_pred
```

```
def classification_accuracy(correct):
    # When averaging a boolean array, False means 0 and True means 1.
    # So we are calculating: number of True / len(correct) which is
    # the same as the classification accuracy.

    # Return the classification accuracy
    # and the number of correct classifications.
    return correct.mean(), correct.sum()
```

```
def predict_cls_test():
    return predict_cls(transfer_values = transfer_values_test,
                       labels = labels_test,
                       cls_true = cls_test)
```

```
def print_test_accuracy(show_example_errors=False,
                        show_confusion_matrix=False):

    # For all the images in the test-set,
    # calculate the predicted classes and whether they are correct.
    correct, cls_pred = predict_cls_test()

    # Classification accuracy and the number of correct classifications.
    acc, num_correct = classification_accuracy(correct)

    # Number of images being classified.
    num_images = len(correct)

    # Print the accuracy.
    msg = "Accuracy on Test-Set: {0:.1%} ({1} / {2})"
    print(msg.format(acc, num_correct, num_images))

    # Plot some examples of mis-classifications, if desired.
    if show_example_errors:
        print("Example errors:")
        plot_example_errors(cls_pred=cls_pred, correct=correct)

    # Plot the confusion matrix, if desired.
    if show_confusion_matrix:
        print("Confusion Matrix:")
        plot_confusion_matrix(cls_pred=cls_pred)
```

Mari kita jalankan.

```
from datetime import timedelta

optimize(num_iterations=1000)

Global Step: 13100, Training Batch Accuracy: 100.0%
Global Step: 13200, Training Batch Accuracy: 100.0%
Global Step: 13300, Training Batch Accuracy: 100.0%
Global Step: 13400, Training Batch Accuracy: 100.0%
Global Step: 13500, Training Batch Accuracy: 100.0%
Global Step: 13600, Training Batch Accuracy: 100.0%
Global Step: 13700, Training Batch Accuracy: 100.0%
Global Step: 13800, Training Batch Accuracy: 100.0%
Global Step: 13900, Training Batch Accuracy: 100.0%
Global Step: 14000, Training Batch Accuracy: 100.0%
Time usage: 0:00:36

print_test_accuracy(show_example_errors=True, show_confusion_matrix=True)

Accuracy on Test-Set: 83.2% (277 / 333)
Example errors:
Confusion Matrix:
[108 3 5] (0) Aamir Khan
[0 83 22] (1) Salman Khan
[4 22 86] (2) Shahrukh Khan
(0) (1) (2)
```

API

Banyak API yang mudah digunakan juga tersedia untuk tugas deteksi wajah dan pengenalan wajah.

Berikut ini beberapa contoh API deteksi wajah:

- PixLab

- Trueface.ai
- Kairos
- Microsoft Computer Vision

Berikut ini beberapa contoh API pengenalan wajah:

- Face++
- LambdaLabs
- KeyLemon
- PixLab

Jika Anda menginginkan deteksi wajah, pengenalan wajah, dan analisis wajah dari satu penyedia, saat ini ada tiga raksasa utama yang memimpin di sini.

- Amazon's Amazon Recognition API
- Microsoft Azure's Face API
- IBM Watson's Visual Recognition API

API Pengenalan Amazon dapat melakukan empat jenis pengenalan.

- *Deteksi objek dan pemandangan:* Pengenalan mengidentifikasi berbagai objek menarik seperti kendaraan, hewan peliharaan, atau furnitur, dan memberikan skor keyakinan.
- *Analisis wajah:* Anda dapat menemukan wajah dalam gambar dan menganalisis atribut wajah, seperti apakah wajah tersebut tersenyum atau mata terbuka, dengan skor keyakinan tertentu.
- *Perbandingan wajah:* API Pengenalan Amazon memungkinkan Anda mengukur kemungkinan wajah dalam dua gambar adalah orang yang sama. Sayangnya, ukuran kesamaan dua wajah orang yang sama bergantung pada usia pada saat foto diambil. Selain itu, peningkatan lokal dalam pencahayaan wajah mengubah hasil perbandingan wajah.
- *Pengenalan wajah:* API mengidentifikasi orang dalam gambar tertentu menggunakan repositori pribadi. Cepat dan akurat.

API Wajah Microsoft Azure akan mengembalikan skor keyakinan untuk seberapa besar kemungkinan kedua wajah tersebut milik satu orang. Microsoft juga memiliki API lain seperti berikut:

- *Computer Vision API:* Fitur ini menampilkan informasi tentang konten visual yang ditemukan dalam gambar. Fitur ini dapat menggunakan penandaan, deskripsi, dan model khusus domain untuk mengidentifikasi konten dan melabelinya dengan yakin.
- *Content Moderation API:* Fitur ini mendeteksi gambar, teks dalam berbagai bahasa, dan konten video yang berpotensi menyinggung atau tidak diinginkan.
- *Emotion API:* Menganalisis wajah untuk mendeteksi berbagai perasaan dan mempersonalisasi respons aplikasi Anda.
- *Video API:* Menghasilkan keluaran video yang stabil, mendeteksi gerakan, membuat gambar mini yang cerdas, serta mendeteksi dan melacak wajah.
- *Video Indexer:* Menemukan wawasan dalam video seperti entitas ucapan, polaritas sentimen ucapan, dan alur waktu audio.
- *Custom Vision Service*: Menandai gambar baru berdasarkan model bawaan atau model

yang dibuat melalui set data pelatihan yang Anda berikan.

Visual Recognition API IBM Watson dapat melakukan beberapa deteksi khusus seperti berikut:

- Menentukan usia orang tersebut.
- Menentukan jenis kelamin orang tersebut.
- Menentukan lokasi kotak pembatas di sekitar wajah.
- Dapat mengembalikan informasi tentang selebritas yang terdeteksi dalam gambar. (Ini tidak dikembalikan saat selebritas tidak terdeteksi.)

DAFTAR PUSTAKA

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... & Zheng, X. (2016). TensorFlow: A system for large-scale machine learning. *In Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI)* (pp. 265-283).

Bishop, C. M. (2006). Pattern recognition and machine learning. Springer.

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet allocation. Journal of Machine Learning Research, 3, 993-1022.

Brownlee, J. (2019). Deep learning for computer vision with Python: Starter bundle. Machine Learning Mastery.

Chen, J., & Wang, Y. (2018). A survey on deep learning for speech recognition. Journal of Computer Science and Technology, 33(1), 1-20.

Chollet, F. (2018). Deep learning with Python. Manning Publications.

Dosovitskiy, A., & Brox, T. (2016). Inverting visual representations with convolutional networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 38(3), 462-473.

Géron, A. (2019). Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems (2nd ed.). O'Reilly Media.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT Press.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 770-778).

Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. Science, 313(5786), 504-507.

Hutter, F., Kotthoff, L., & Vanschoren, J. (2019). Automated machine learning: Methods, systems, challenges. Springer.

Karpathy, A. (2015). *CS231n: Convolutional Neural Networks for Visual Recognition*. Stanford University.

Kelleher, J. D., & Tierney, B. (2018). Data science: A practical introduction to data science. The MIT Press.

Keras Team. (2021). Keras: The Python deep learning library. *GitHub*.

Khoshgoftaar, T. M., & Van Hulse, J. (2018). A survey of deep learning for big data. Journal of Big Data, 5(1), 1-20.

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. *In Proceedings of the 3rd International Conference on Learning Representations (ICLR)*.

Koller, D., & Friedman, N. (2009). Probabilistic graphical models: Principles and techniques. MIT Press.

LeCun, Y., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal loss for dense object detection. In Proceedings of the IEEE International Conference on Computer Vision (ICCV) (pp. 2980-2988). https://doi.org/10.1109/ICCV.2017.324

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., & Reed, S. (2016). SSD: Single shot multibox detector. *In European Conference on Computer Vision (ECCV)* (pp. 21-37).

Liu, Y., & Wang, Y. (2019). A survey on deep learning for image segmentation. Journal of Computer Science and Technology, 34(1), 1-20.

Rojas, R. (2013). Neural networks: A systematic introduction. Springer.

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. In Proceedings of the International Conference on Learning Representations (ICLR).

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In Advances in Neural Information Processing Systems (NIPS) (pp. 3104-3112).

Szegedy, C., Vanhoucke, V., Vinyals, O., & Google, L. (2015). Going deeper with convolutions. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 1-9).

Wang, Y., & Zhang, Y. (2019). A survey on deep learning for anomaly detection. Journal of Computer Science and Technology, 34(1), 1-20.

Wu, Y., & He, K. (2018). Group normalization. In Proceedings of the European Conference on Computer Vision (ECCV) (pp. 3-19).

Xu, B., Wang, N., & Chen, T. (2015). Empirical evaluation of rectified activations in convolutional network. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV) (pp. 1026-1034).

Yao, Y., & Wang, Y. (2019). A survey on deep learning for image captioning. Journal of Visual Communication and Image Representation, 58, 1-12.

Yang, Q. (2018). A survey on multi-task learning. IEEE Transactions on Knowledge and Data Engineering, 30(9), 1623-1639.

Zhang, H., & Zhang, Y. (2019). A survey on deep learning for time series forecasting. Journal of Forecasting, 38(5), 1-15.

Zhang, K., Zhang, Z., Li, Z., & Qiao, Y. (2016). Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, 23(10), 1499-1503.

Zhang, Y., & Wang, Y. (2018). A survey on deep learning for image classification. Journal of Computer Science and Technology, 33(1), 1-20.

Zhang, Y., & Wang, Y. (2019). A survey on deep learning for natural language processing. Journal of Computer Science and Technology, 34(1), 1-20.

DEEP LEARNING DENGAN PYTHON

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

BIODATA PENULIS

Dr. Joseph Teguh Santoso, M.Kom adalah pemimpin yang visioner dan praktisi industri berpengalaman, yang 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 978-634-7227-11-9 (PDF)