BAB 1

Pengantar Visi Komputer dan Pembelajaran Mendalam

Penglihatan merupakan anugerah terbaik dari Tuhan bagi umat manusia.

Sejak lahir, penglihatan memungkinkan kita mengembangkan pikiran yang sadar. Warna, bentuk, objek, dan wajah merupakan blok bangunan bagi dunia kita. Anugerah alam ini sangat penting bagi indra kita.

Visi komputer adalah salah satu kemampuan yang memungkinkan mesin untuk meniru kekuatan ini. Dan dengan menggunakan Pembelajaran Mendalam, kami meningkatkan perintah kami dan membuat kemajuan di bidang ini.

Buku ini akan membahas konsep visi komputer dari sudut pandang Pembelajaran Mendalam. Kita akan mempelajari blok-blok dasar Jaringan Syaraf Tiruan, mengembangkan kasus-kasus penggunaan pragmatis dengan mengambil pendekatan berbasis studi kasus, dan membandingkan serta mengontraskan kinerja berbagai solusi. Kami akan membahas praktik terbaik, berbagi kiat dan wawasan yang diikuti dalam industri, membuat Anda menyadari jebakan umum, dan mengembangkan proses berpikir untuk merancang Jaringan Saraf.

Verdhan, *Visi Komputer Menggunakan Pembelajaran Mendalam,*
https://doi.org/10.1007/978-1-4842-6616-8_1

Sepanjang buku ini, kami memperkenalkan sebuah konsep, mengeksplorasinya secara terperinci, lalu mengembangkan kasus penggunaan dalam Python di sekitarnya. Karena sebuah bab pertama-tama membangun fondasi Deep Learning dan kemudian penggunaan praktisnya, pengetahuan yang lengkap memungkinkan Anda untuk merancang solusi dan kemudian mengembangkan Jaringan Neural untuk pengambilan keputusan yang lebih baik.

Diperlukan pengetahuan tentang Python dan konsep pemrograman berorientasi objek untuk pemahaman yang baik. Pemahaman dasar hingga menengah tentang ilmu data disarankan, meskipun bukan persyaratan yang mutlak.

Dalam bab pengantar ini, kita akan mengembangkan konsep Pemrosesan Gambar menggunakan OpenCV dan Pembelajaran Mendalam. OpenCV adalah pustaka yang hebat dan banyak digunakan dalam robotika, pengenalan wajah, pengenalan gerakan, AR, dan sebagainya. Pembelajaran Mendalam, sebagai tambahan, menawarkan tingkat kompleksitas dan fleksibilitas yang lebih tinggi untuk mengembangkan kasus penggunaan Pemrosesan Gambar. Kita akan membahas topik-topik berikut dalam bab ini:

(1) Pengolahan Citra Menggunakan OpenCV

(2) Dasar-dasar Pembelajaran Mendalam

(3) Cara Kerja Deep Learning

(4) Pustaka Pembelajaran Mendalam yang Populer

1.1 Persyaratan teknis

Kami mengembangkan semua solusi dalam Python di seluruh buku ini; oleh karena itu, instalasi Python versi terbaru diperlukan.

Semua kode, kumpulan data, dan hasil masing-masing diperiksa ke dalam kode repositori di https://github.com/Apress/computer-vision-using-deep-learning/tree/main/Chapter1 . Anda disarankan untuk menjalankan semua kode bersama kami dan meniru hasilnya. Ini akan memperkuat pemahaman Anda terhadap konsep tersebut.

1.2 Pengolahan Gambar menggunakan OpenCV

Gambar juga seperti titik data lainnya. Di komputer dan ponsel, gambar muncul sebagai objek atau ikon dalam format .jpeg, .bmp, dan .png.
Oleh karena itu, manusia akan kesulitan memvisualisasikannya dalam struktur baris-kolom, seperti kita memvisualisasikan basis data lainnya. Oleh karena itu, data ini sering disebut sebagai *data tak terstruktur.*

Agar komputer dan algoritme kita dapat menganalisis gambar dan mengolahnya, kita harus merepresentasikan gambar dalam bentuk bilangan bulat. Oleh karena itu, kita mengolah gambar piksel demi piksel. Secara matematis, salah satu cara merepresentasikan setiap piksel adalah nilai RGB (Red, Green, Blue). Kita menggunakan informasi ini untuk melakukan Pemrosesan Gambar.

Info Cara termudah untuk mendapatkan RGB untuk warna apa pun adalah dengan membukanya di Paint pada sistem operasi Windows. Arahkan kursor ke warna apa pun dan dapatkan nilai RGB yang sesuai. Di Mac OS, Anda dapat menggunakan Digital Colour Meter.

Pembelajaran Mendalam memungkinkan kita mengembangkan kasus penggunaan yang jauh lebih rumit untuk diselesaikan menggunakan teknik Pemrosesan Gambar tradisional. Misalnya, mendeteksi wajah juga dapat dilakukan menggunakan OpenCV, tetapi untuk dapat mengenalinya diperlukan Pembelajaran Mendalam.

Selama proses pengembangan solusi visi komputer menggunakan Pembelajaran Mendalam, kami menyiapkan kumpulan data gambar terlebih dahulu. Selama persiapan, kami mungkin harus melakukan penskalaan abu-abu pada gambar, mendeteksi kontur, memotong gambar, lalu memasukkannya ke Jaringan Syaraf.

OpenCV adalah pustaka yang paling terkenal untuk tugas-tugas semacam itu. Sebagai langkah pertama, mari kita kembangkan beberapa blok penyusun metode Pemrosesan Gambar ini. Kita akan membuat tiga solusi menggunakan OpenCV.

Catatan Kunjungi www.opencv.org dan ikuti petunjuk di sana untuk menginstal OpenCV pada sistem Anda.

Gambar yang digunakan untuk solusi adalah gambar yang umum tersedia. Anda disarankan untuk memeriksa kode dan mengikuti implementasi langkah demi langkah yang dilakukan. Kami akan mendeteksi bentuk, warna, dan wajah dalam gambar.

Mari selami dunia gambar yang menarik!

1.2.1 Deteksi warna menggunakan OpenCV

Ketika kita memikirkan suatu gambar, ia terdiri dari bentuk, ukuran, dan warna. Memiliki kemampuan untuk mendeteksi bentuk, ukuran, atau warna dalam sebuah gambar dapat mengotomatiskan banyak proses dan menghemat banyak tenaga. Pada contoh pertama, kita akan mengembangkan sebuah "sistem deteksi warna".

Deteksi warna memiliki berbagai kegunaan di berbagai domain dan industri seperti manufaktur, otomotif, listrik, utilitas, dan sebagainya. Deteksi warna dapat digunakan untuk mencari gangguan, kegagalan, dan gangguan pada perilaku normal. Kita dapat melatih sensor untuk mengambil keputusan tertentu berdasarkan warna dan membunyikan alarm jika diperlukan.

Suatu gambar direpresentasikan menggunakan piksel, dan setiap piksel terdiri dari RGB nilai berkisar dari 0 hingga 255. Kita akan menggunakan properti ini untuk mengidentifikasi warna biru pada gambar (Gambar 1-1). Anda dapat mengubah nilai masing-masing untuk warna biru dan mendeteksi warna pilihan apa pun.

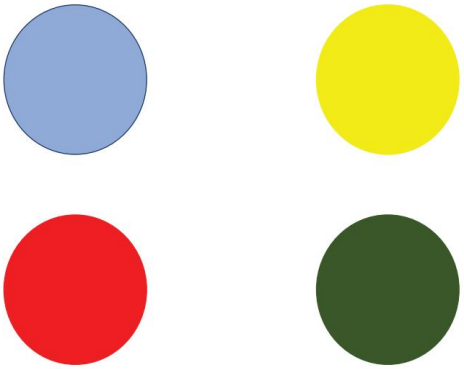
Ikuti langkah-langkah berikut:

1. Buka Python Jupyter Notebook.
2. Pertama muat pustaka yang diperlukan, numpy dan OpenCV.

```
impor numpy sebagai np
impor cv2
```

3. Muat berkas gambar.

```
gambar = cv2.imread('Warna.png')
```



Gambar 1-1. *Gambar mentah yang akan digunakan untuk deteksi warna. Gambar yang ditampilkan memiliki empat warna berbeda, dan solusi OpenCV akan mendeteksinya satu per satu.*

4. Sekarang mari kita ubah gambar mentah kita menjadi HSV (Hue Saturation Format Value). Format ini memungkinkan kita untuk memisahkan saturasi dan pseudo-iluminasi. cv2.cvtColor memungkinkan kita untuk melakukan hal itu.

   hsv_convert = cv2.cvtColor(gambar, cv2.COLOR_BGR2HSV)

5. Tentukan rentang atas dan bawah warna di sini. Kita mendeteksi warna biru. Dari pustaka numpy, kita telah memberikan rentang masing-masing untuk warna biru.

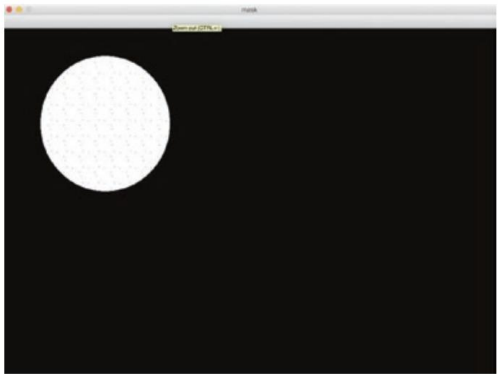
   rentang_bawah = np.array([110,50,50]) rentang_atas =
   np.array([130,255,255])

6. Sekarang, mari kita mendeteksi warna biru dan memisahkannya dari sisa gambar.

   mask_toput = cv2.inRange(hsv_convert, rentang_bawah, rentang_atas)
   cv2.imshow('gambar',
   gambar) cv2.imshow('topeng', mask_toput)

   sementara(Benar):
   k = cv2.waitKey(5)& 0xFF jika k== 27: putus

Output kode ini akan seperti ditunjukkan pada Gambar 1-2.



Gambar 1-2. *Output dari sistem deteksi warna. Kita ingin mendeteksi warna biru yang terdeteksi dan dipisahkan dari sisa gambar.*

Seperti yang terlihat, warna biru disorot dalam warna putih, sedangkan bagian gambar lainnya berwarna hitam. Dengan mengubah rentang pada langkah 5, Anda dapat mendeteksi berbagai warna pilihan Anda.

Setelah warna selesai, saatnya mendeteksi bentuk dalam gambar; ayo kita lakukan!

1.3 Deteksi bentuk menggunakan OpenCV

Seperti kita mendeteksi warna biru di bagian terakhir, kita akan mendeteksi segitiga, persegi, persegi panjang, dan lingkaran dalam sebuah gambar. Deteksi bentuk memungkinkan Anda untuk memisahkan bagian-bagian dalam gambar dan memeriksa polanya. Deteksi warna dan bentuk membuat solusi menjadi sangat konkret. Kegunaannya terletak pada pemantauan keselamatan, jalur produksi, pusat otomotif, dan sebagainya.

Untuk deteksi bentuk, kita mendapatkan kontur setiap bentuk, memeriksa jumlah elemen, lalu mengklasifikasikannya sesuai dengan bentuknya. Misalnya, jika angkanya tiga, maka itu adalah segitiga. Dalam solusi ini, Anda juga akan mengamati cara mengubah gambar menjadi skala abu-abu dan mendeteksi kontur.

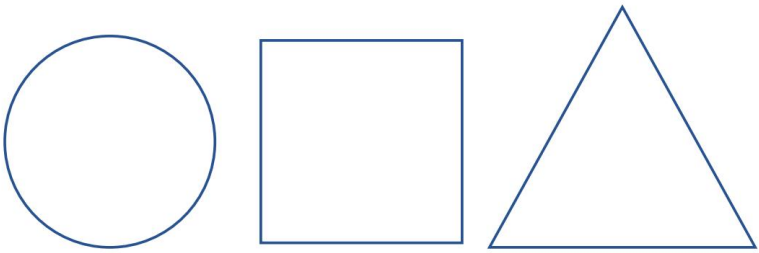
Ikuti langkah-langkah berikut untuk mendeteksi bentuk:

1. Impor pustaka terlebih dahulu.

```
impor numpy sebagai np
impor cv2
```

2. Muat gambar mentah yang sekarang ditunjukkan pada Gambar 1-3.

```
bentuk_gambar = cv2.imread('bentuk.png')
```



Gambar 1-3. *Gambar input mentah untuk mendeteksi tiga bentuk lingkaran, segitiga, dan persegi panjang*

3. Selanjutnya, ubah gambar menjadi skala abu-abu. Skala abu-abu dilakukan untuk menyederhanakan karena RGB bersifat tiga dimensi sedangkan skala abu-abu bersifat dua dimensi, dan mengubah ke skala abu-abu menyederhanakan solusi. Hal ini juga membuat kode menjadi efisien.

```
gray_image = cv2.cvtColor(gambar_bentuk, cv2.COLOR_BGR2GRAY)
ret,thresh = cv2.threshold(gambar_abu-abu,127,255,1)
```

4. Temukan kontur pada gambar.

```
kontur,h = cv2.temukanKontur(ambang,1,2)
```

5. Cobalah untuk memperkirakan setiap kontur menggunakan
approxPolyDP. Metode ini mengembalikan jumlah
elemen dalam kontur terdeteksi. Kemudian kami memutuskan
bentuk berdasarkan jumlah elemen dalam kontur.
Jika nilainya tiga, maka bentuknya segitiga; jika nilainya empat, maka
bentuknya persegi; dan seterusnya.

```
untuk cnt dalam kontur:
    sekitar =
    cv2.approxPolyDP(cnt,0.01*cv2.arcLength(cnt,Benar),Benar)
cetak (len(approx))
jika len(approx)==3:
    cetak ("segitiga")
    cv2.gambarKontur(bentuk_gambar,[cnt],0,(0,255,0),-1)
elif len(kira-kira)==4:
    cetak ("persegi")
    cv2.gambarKontur(bentuk_gambar,[cnt],0,(0,0,255),-1)
elif len(kira-kira) > 15:
    cetak ("lingkaran")
    cv2.gambarKontur(bentuk_gambar,[cnt],0,(0,255,255),-1)
cv2.imshow('bentuk_gambar',bentuk_gambar) cv2.waitKey(0)
cv2.hancurkansemuajendela()
```

6. Output kode sebelumnya ditunjukkan pada Gambar 1-4.

Gambar 1-4. *Output dari sistem deteksi warna. Lingkaran ditunjukkan dengan warna kuning, persegi ditunjukkan dengan warna merah, dan segitiga ditunjukkan dengan warna hijau.*

Kini Anda dapat mendeteksi bentuk dalam gambar apa pun. Kita telah mendeteksi lingkaran, segitiga, dan persegi. Tantangan yang bagus adalah mendeteksi pentagon atau heksagon; apakah Anda tertarik?

Ayo lakukan sesuatu yang lebih menyenangkan sekarang!

1.3.1 Deteksi wajah menggunakan OpenCV

Deteksi wajah bukanlah kemampuan baru. Setiap kali kita melihat sebuah gambar, kita dapat mengenali wajah dengan mudah. Kamera ponsel kita menggambar kotak persegi di sekeliling wajah. Atau di media sosial, kotak persegi dibuat di sekeliling wajah. Ini disebut *deteksi wajah.*

Deteksi wajah mengacu pada pencarian wajah manusia dalam gambar digital. Deteksi wajah berbeda dengan pengenalan wajah. Pada deteksi wajah, kita hanya mendeteksi wajah dalam gambar, sedangkan pada pengenalan wajah, kita juga memberi nama pada wajah tersebut, yaitu, siapa orang dalam foto tersebut.

Sebagian besar kamera dan ponsel modern memiliki kemampuan bawaan untuk mendeteksi wajah. Solusi serupa dapat dikembangkan menggunakan OpenCV. Solusi ini lebih mudah dipahami dan diterapkan serta dibangun menggunakan Haar-cascade. algoritma. Kita akan menyorot wajah dan mata dalam sebuah foto sambil menggunakan algoritma ini dalam Python.

Pengklasifikasi Haar-cascade digunakan untuk mendeteksi wajah dan fitur wajah lainnya. atribut seperti mata dalam gambar. Ini adalah solusi Pembelajaran Mesin di mana pelatihan dilakukan pada banyak gambar yang memiliki wajah dan yang tidak memiliki wajah di dalamnya. Pengklasifikasi mempelajari fitur masing-masing. Kemudian kami menggunakan pengklasifikasi yang sama untuk mendeteksi wajah bagi kami. Kami tidak perlu melakukan pelatihan apa pun di sini karena pengklasifikasi sudah dilatih dan siap digunakan. Menghemat waktu dan tenaga juga!

Deteksi Objek Info menggunakan pengklasifikasi kaskade berbasis Haar diusulkan oleh Paul Viola dan Michael Jones dalam makalah mereka "Rapid Object Detection using a Boosted Cascade of Simple Features" pada tahun 2001. Anda disarankan untuk membaca makalah inovatif ini.

Ikuti langkah-langkah berikut untuk mendeteksi wajah:

1. Impor pustaka terlebih dahulu.

```
impor numpy sebagai np
impor cv2
```

2. Muat file xml pengklasifikasi. File .xml dirancang oleh OpenCV dan dibuat dengan melatih rangkaian wajah negatif yang ditumpangkan pada gambar positif dan karenanya dapat mendeteksi fitur wajah.

```
muka_bertingkat =
cv2.PengklasifikasiBertingkat('muka_bertingkat_default.xml')
mata_cascade =
cv2.CascadeClassifier( 'haarcascade_eye.xml')
```

3. Berikutnya, muat gambar (Gambar 1-5).

```
img = cv2.imread('Vaibhav.jpg')
```

Gambar 1-5. *Citra masukan mentah wajah yang digunakan untuk deteksi wajah menggunakan solusi Haar-cascade*

4. Ubah gambar menjadi skala abu-abu.

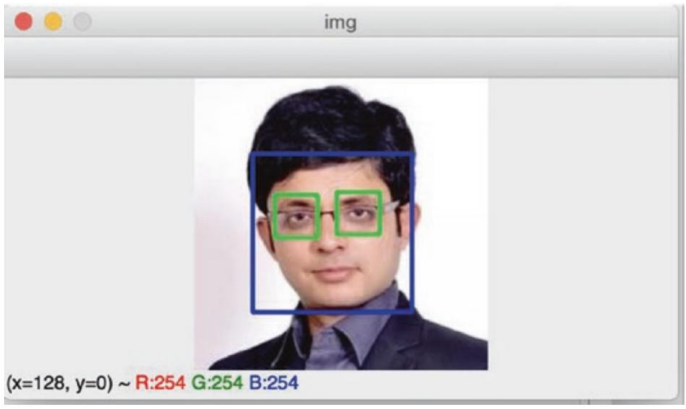
```
abu-abu = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

5. Jalankan kode berikut untuk mendeteksi wajah di gambar. Jika ada wajah yang ditemukan, kami mengembalikan posisi wajah yang terdeteksi sebagai Rect(x,y,w,h). Selanjutnya, mata dideteksi pada wajah.

```
wajah = face_cascade.detectMultiScale(abu-abu, 1.3, 5) untuk
(x,y,w,h) di wajah:
gambar = cv2.persegi panjang(gambar,(x,y),(x+w,y+h),
(255,0,0),2) roi_gr = gray[y:y+h, x:x+w] roi_clr =
img[y:y+h, x:x+w]
mata_mata = riam_mata.deteksiMultiSkala(roi_gr)
untuk (mis.,ey,ew,eh) di mata:
    cv2.rectangle(roi_clr,(mis.,ey),(mis.+ew,ey+eh), (0,255,0),2)

    cv2.imshow('img',image) cv2.waitKey(0)
    cv2.destroyAllWindows()
```

6. Hasilnya ditunjukkan pada Gambar 1-6. Perhatikan bagaimana kotak biru digambar di sekitar wajah dan dua kotak kecil berwarna hijau di sekitar mata.



Gambar 1-6. *Wajah dan mata terdeteksi dalam gambar; kotak hijau berada di sekitar mata dan kotak biru berada di sekitar wajah*

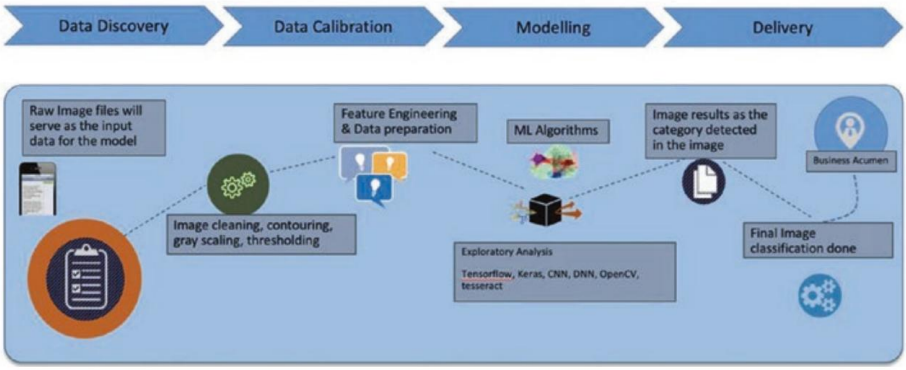
Deteksi wajah memungkinkan kita menemukan wajah dalam gambar dan video. Ini adalah langkah pertama untuk pengenalan wajah. Ini digunakan secara luas untuk aplikasi keamanan, pemantauan kehadiran, dan sebagainya. Kami akan mengembangkan deteksi dan pengenalan wajah menggunakan Deep Learning di bab-bab berikutnya.

Kita telah mempelajari beberapa konsep Pengolahan Citra. Sekarang saatnya untuk pelajari dan pelajari konsep Deep Learning. Ini adalah fondasi perjalanan yang telah Anda mulai.

1.4 Dasar-dasar Pembelajaran Mendalam

Pembelajaran Mendalam merupakan subbidang dari Pembelajaran Mesin. "Pendalaman" dalam Pembelajaran Mendalam memiliki lapisan representasi yang berurutan; oleh karena itu, kedalaman model mengacu pada jumlah lapisan dalam model Jaringan Syaraf Tiruan (JST) dan pada dasarnya disebut sebagai Pembelajaran Mendalam.

Ini adalah pendekatan baru untuk menganalisis data historis dan belajar darinya lapisan-lapisan representasi yang semakin bermakna. Proses umum dalam proyek Pembelajaran Mendalam mirip dengan proyek Pembelajaran Mesin seperti yang dijelaskan berikut ini dan ditunjukkan pada Gambar 1-7.



Gambar 1-7. *Proses pembelajaran mesin menyeluruh dari penemuan data hingga pengembangan akhir solusi. Semua langkah dibahas secara rinci di sini dan dibahas kembali di Bab 8 buku ini*

1. Penyerapan data: File data mentah/gambar/teks dan sebagainya diserap ke dalam sistem. Mereka berfungsi sebagai input data untuk melatih dan menguji jaringan.

2. Pembersihan data: Pada langkah ini, kami membersihkan data. Sering kali, terdapat terlalu banyak gangguan seperti nilai sampah, duplikat, NULL, dan outlier yang ada dalam kumpulan data terstruktur. Semua titik data tersebut harus ditangani pada tahap ini. Untuk gambar, kita mungkin harus menghilangkan noise yang tidak perlu pada gambar.

3. Persiapan data: Kami menyiapkan data kami untuk pelatihan. Pada langkah ini, variabel turunan baru mungkin diperlukan, atau kita mungkin perlu memutar/memotong gambar jika kita bekerja pada kumpulan data gambar.

4. Analisis data eksploratif: Kami melakukan analisis awal untuk menghasilkan wawasan cepat tentang kumpulan data kami.

5. Desain jaringan dan pelatihan model: Kami merancang Jaringan Saraf kami di sini dan tentukan jumlahnya lapisan tersembunyi, simpul, fungsi aktivasi, fungsi kerugian, dan sebagainya. Jaringan kemudian dilatih.

6. Periksa akurasi dan ulangi: Kami mengukur keakuratan jaringan. Kami menggunakan Confusion Matrix, nilai AUC, presisi, recall, dan sebagainya untuk mengukurnya. Lalu kami menyetel hiperparameter dan menyetelnya lebih lanjut.

7. Model akhir dipresentasikan ke bisnis dan kami dapatkan umpan baliknya.

8. Kami mengulangi model dan memperbaikinya berdasarkan masukan yang diterima dan solusi akhir dibuat.

9. Model tersebut digunakan dalam produksi. Kemudian, model tersebut dirawat dan diperbarui secara berkala.

Langkah-langkah ini biasanya diikuti selama proyek Pembelajaran Mesin. Kita akan membahas semua langkah ini secara terperinci di bab terakhir buku ini. Sekaranglah saatnya untuk memahami Jaringan Neural.

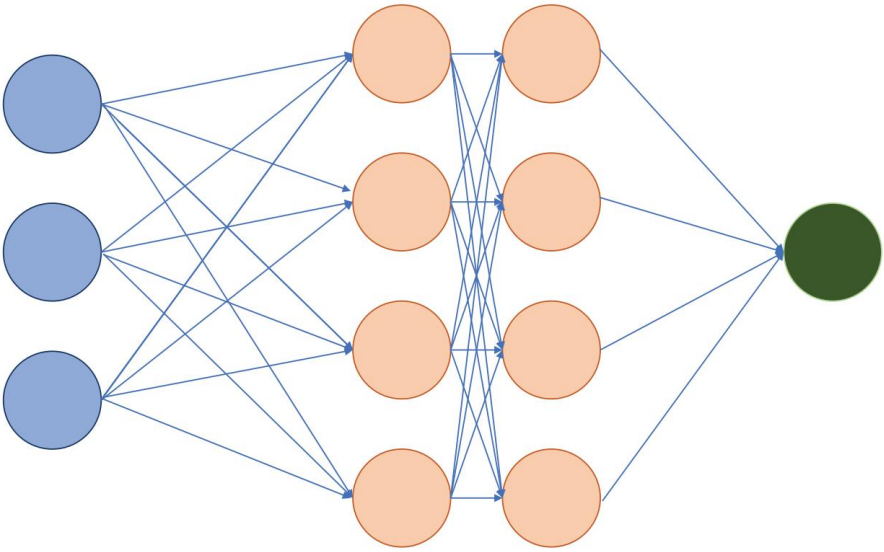
1.4.1 Motivasi di balik Jaringan Saraf

Jaringan Syaraf Tiruan (JST) konon terinspirasi dari cara kerja otak manusia. Saat kita melihat sebuah gambar, kita mengasosiasikan suatu label dengan gambar tersebut. Kita melatih otak dan indra kita untuk mengenali gambar saat kita melihatnya lagi dan memberinya label yang benar.

ANN belajar melakukan tugas serupa dengan belajar atau mendapatkan pelatihan. Hal ini dilakukan dengan melihat berbagai contoh titik data historis seperti data transaksional atau gambar dan sebagian besar waktu tanpa diprogram untuk aturan tertentu. Misalnya, untuk membedakan antara mobil dan manusia, ANN akan mulai tanpa pemahaman dan pengetahuan sebelumnya tentang atribut masing-masing kelas. Kemudian, ANN menghasilkan atribut dan karakteristik identifikasi dari data pelatihan. ANN kemudian mempelajari atribut tersebut dan menggunakannya nanti untuk membuat prediksi.

Secara formal, "belajar" dalam konteks Jaringan Syaraf Tiruan mengacu pada penyesuaian bobot dan bias di dalam jaringan untuk meningkatkan akurasi jaringan selanjutnya. Dan cara yang jelas untuk melakukan ini adalah dengan mengurangi istilah kesalahan yang tidak lain adalah perbedaan antara nilai aktual dan nilai prediksi. Untuk mengukur tingkat kesalahan, kita memiliki fungsi biaya yang ditetapkan yang dievaluasi secara ketat selama fase pembelajaran jaringan. Kita akan memeriksa semua istilah ini secara terperinci di bagian berikutnya.

Jaringan Syaraf Tiruan pada umumnya tampak seperti Gambar 1-8.



Gambar 1-8. *Jaringan Neural yang umum dengan lapisan input, lapisan tersembunyi, dan lapisan output. Setiap lapisan memiliki beberapa neuron di dalamnya. Lapisan tersembunyi bertindak sebagai jantung dan jiwa jaringan. Lapisan input menerima data input, dan lapisan output bertanggung jawab untuk menghasilkan hasil akhir.*

Jaringan Neural yang ditunjukkan sebelumnya memiliki tiga unit input, dua unit tersembunyi lapisan dengan empat neuron masing-masing, dan satu lapisan keluaran akhir.

Sekarang mari kita bahas berbagai komponen Jaringan Syaraf di bagian selanjutnya.

1.4.2 Lapisan dalam Jaringan Saraf

Arsitektur Jaringan Syaraf dasar terutama terdiri dari tiga lapisan:

- Lapisan masukan: Sesuai namanya, lapisan ini menerima data masukan. Pertimbangkan untuk memasukkan gambar mentah/gambar yang telah diproses ke lapisan masukan. Ini adalah langkah pertama dari Neural Jaringan.

- Lapisan tersembunyi: Lapisan ini merupakan jantung dan jiwa jaringan. Semua pemrosesan, ekstraksi fitur, pembelajaran, dan pelatihan dilakukan di lapisan ini. Lapisan tersembunyi memecah data mentah menjadi atribut dan fitur serta mempelajari nuansa data. Pembelajaran ini digunakan kemudian di lapisan keluaran untuk membuat keputusan.
- Lapisan keluaran: Lapisan keputusan dan bagian akhir dalam jaringan. Lapisan ini menerima keluaran dari lapisan tersembunyi sebelumnya dan kemudian membuat penilaian pada klasifikasi akhir.

Blok pembangun jaringan yang paling rinci adalah neuron. Neuron adalah di mana seluruh keajaiban terjadi, yang akan kita bahas selanjutnya.

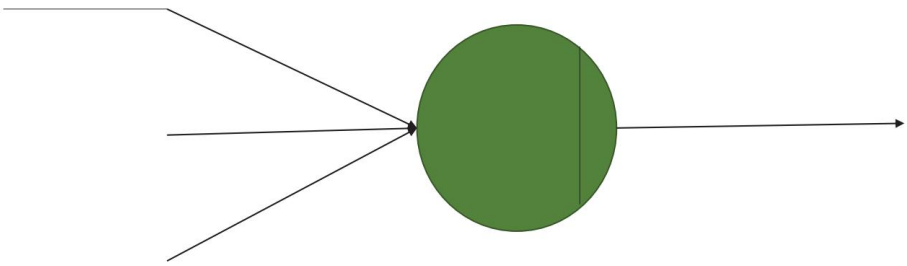
1.4.3 Sel Saraf

Neuron atau neuron buatan merupakan fondasi dari Jaringan Saraf.

Seluruh perhitungan kompleks terjadi hanya pada satu neuron. Lapisan di jaringan dapat berisi lebih dari satu neuron.

Neuron menerima masukan dari lapisan sebelumnya atau lapisan masukan, lalu memproses informasi tersebut dan membagikan keluaran. Data masukan dapat berupa data mentah atau informasi yang telah diproses dari neuron sebelumnya. Neuron kemudian menggabungkan masukan tersebut dengan status internalnya sendiri dan mencapai nilai menggunakan fungsi aktivasi (kita akan membahas fungsi aktivasi nanti). Selanjutnya, keluaran dihasilkan menggunakan fungsi keluaran.

Sebuah neuron dapat dianggap sebagai Gambar 1-9 di mana ia menerima masukan masing-masing dan menghitung keluarannya.



Gambar 1-9. *Representasi neuron yang menerima input dari lapisan sebelumnya dan menggunakan fungsi aktivasi untuk memproses dan memberikan output. Ini adalah blok penyusun Jaringan Neural*

Input ke neuron diterima dari output pendahulunya dan koneksinya masing-masing. Input yang diterima dihitung sebagai jumlah tertimbang, dan suku bias umumnya ditambahkan juga. Ini adalah fungsi dari *fungsi propagasi.* Seperti yang ditunjukkan pada Gambar 1-9, f adalah fungsi aktivasi, w adalah suku bobot, dan b adalah suku bias. Setelah perhitungan selesai, kita menerima output.

Misalnya, data pelatihan input akan memiliki gambar mentah atau gambar yang telah diproses. Gambar-gambar ini akan dimasukkan ke lapisan input. Data sekarang akan masuk ke lapisan tersembunyi tempat semua perhitungan dilakukan. Perhitungan ini dilakukan oleh neuron di setiap lapisan.

Output adalah tugas yang perlu diselesaikan, misalnya identifikasi suatu objek, atau jika kita ingin mengklasifikasikan suatu gambar, dan seterusnya.

Seperti yang telah kita bahas, Jaringan Neural pada umumnya mampu mengekstrak informasi itu sendiri, tetapi kita masih harus memulai beberapa parameter untuk proses pelatihan jaringan. Parameter-parameter ini disebut sebagai *hiperparameter* yang akan kita bahas selanjutnya.

1.4.4 Hiperparameter

Selama pelatihan jaringan, algoritma terus-menerus mempelajari atribut data mentah. Namun ada beberapa parameter yang harus diperhatikan oleh jaringan.

tidak dapat mempelajari dirinya sendiri dan memerlukan pengaturan awal. Hiperparameter adalah variabel dan atribut yang tidak dapat dipelajari oleh Jaringan Syaraf Tiruan dengan sendirinya. Ini adalah variabel yang menentukan struktur Jaringan Syaraf Tiruan dan variabel terkait yang berguna untuk melatih jaringan.

Hiperparameter ditetapkan sebelum pelatihan jaringan yang sebenarnya. Laju pembelajaran, jumlah lapisan tersembunyi dalam jaringan, jumlah neuron di setiap lapisan, fungsi aktivasi, jumlah epoch, ukuran batch, dropout, dan inisialisasi bobot jaringan adalah contoh hiperparameter.

Penyetelan hiperparameter adalah proses memilih nilai terbaik untuk hiperparameter berdasarkan kinerjanya. Kami mengukur kinerja jaringan pada set validasi, lalu mengubah hiperparameter, lalu mengevaluasi ulang dan mengubah lagi, dan proses ini berlanjut. Kami akan memeriksa berbagai hiperparameter di bagian berikutnya.

1.4.5 Koneksi dan bobot ANN

ANN terdiri dari berbagai koneksi. Setiap koneksi bertujuan untuk menerima masukan dan memberikan keluaran yang dihitung. Keluaran ini berfungsi sebagai masukan ke neuron berikutnya.

Selain itu, setiap koneksi diberi bobot yang mewakili kepentingannya masing-masing. Penting untuk dicatat bahwa neuron dapat memiliki beberapa koneksi input dan output yang berarti neuron dapat menerima input dan mengirimkan beberapa sinyal.

Istilah berikutnya lagi-lagi merupakan komponen penting – istilah bias.

1.4.6 Istilah Bias

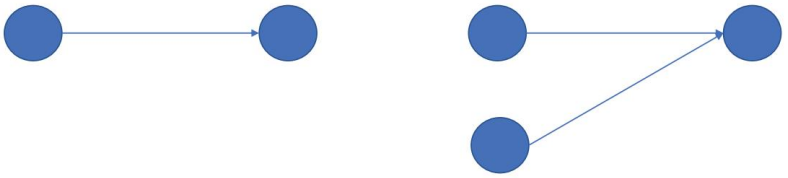
Bias sama seperti menambahkan nilai intersep ke persamaan linear. Bias merupakan parameter ekstra atau tambahan dalam jaringan.

Cara paling sederhana untuk memahami bias adalah seperti persamaan berikut:

persamaan y = mx + c

Jika kita tidak memiliki suku konstan c, persamaan akan melewati (0,0). Jika kita memiliki suku konstan c, kita dapat mengharapkan model pembelajaran mesin yang lebih sesuai.

Seperti yang dapat kita amati pada Gambar 1-10, di sebelah kiri, kita memiliki neuron tanpa istilah bias, sementara di sebelah kanan kami telah menambahkan istilah bias. Jadi bias memungkinkan kita untuk menyesuaikan output beserta jumlah bobot input.



Gambar 1-10. *Istilah bias membantu dalam penyesuaian model yang lebih baik. Di sisi kiri, tidak ada istilah bias, dan di sisi kanan kita memiliki istilah bias. Perhatikan bahwa istilah bias memiliki bobot yang terkait dengannya*

Oleh karena itu, istilah bias bertindak seperti istilah konstan dalam persamaan linier, dan ini membantu dalam penyesuaian data dengan lebih baik.

Sekarang kita akan mempelajari salah satu atribut terpenting dalam Neural Jaringan – fungsi aktivasi – di bagian berikutnya.

1.4.7 Fungsi aktivasi

Peran utama fungsi aktivasi adalah untuk memutuskan apakah neuron/ perceptron harus diaktifkan atau tidak. Mereka memainkan peran utama dalam menyesuaikan gradien selama pelatihan jaringan pada tahap selanjutnya. Fungsi aktivasi ditunjukkan pada Gambar 1-9. Kadang-kadang mereka disebut sebagai *fungsi transfer.*

Perilaku nonlinier dari fungsi aktivasi memungkinkan jaringan Deep Learning mempelajari perilaku yang kompleks. Anda akan mempelajari apa yang dimaksud dengan perilaku nonlinier di Bab 2. Sekarang, kita akan mempelajari beberapa fungsi yang umum digunakan.

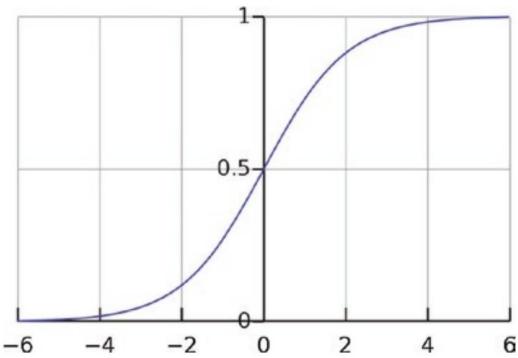
1.4.7.1 Fungsi sigmoid

Fungsi sigmoid adalah fungsi matematika monotonik terbatas. Fungsi ini merupakan fungsi diferensial dengan kurva berbentuk S, dan fungsi turunan pertamanya berbentuk lonceng. Fungsi ini memiliki fungsi turunan nonnegatif dan didefinisikan untuk semua nilai masukan riil.
Fungsi sigmoid digunakan jika nilai keluaran neuron berada di antara 0 dan 1.

Secara matematis, fungsi sigmoid seperti yang ditunjukkan pada Persamaan 1-1.

$$S(x) \quad \frac{1}{1 \quad _{\_}^{x}} \quad \frac{_{\_}^{x}}{_{\_}^{x} \quad 1} \qquad \text{(Persamaan 1-1)}$$

Grafik fungsi sigmoid dapat dilihat pada Gambar 1-11. Perhatikan bentuk fungsi dan nilai maksimum dan minimumnya.



Gambar 1-11. *Fungsi sigmoid; perhatikan bahwa fungsi ini tidak berpusat pada nol, dan nilainya berada di antara 0 dan 1. Fungsi sigmoid memiliki masalah gradien yang menghilang*

Fungsi Sigmoid menemukan penerapannya dalam sistem pembelajaran yang kompleks. Seringkali Anda akan menemukan fungsi Sigmoid digunakan ketika model matematika tertentu tidak sesuai. Fungsi ini biasanya digunakan untuk klasifikasi biner dan pada lapisan keluaran akhir jaringan. Fungsi Sigmoid mengalami masalah gradien yang menghilang yang akan kita bahas di bagian selanjutnya.
bagian.

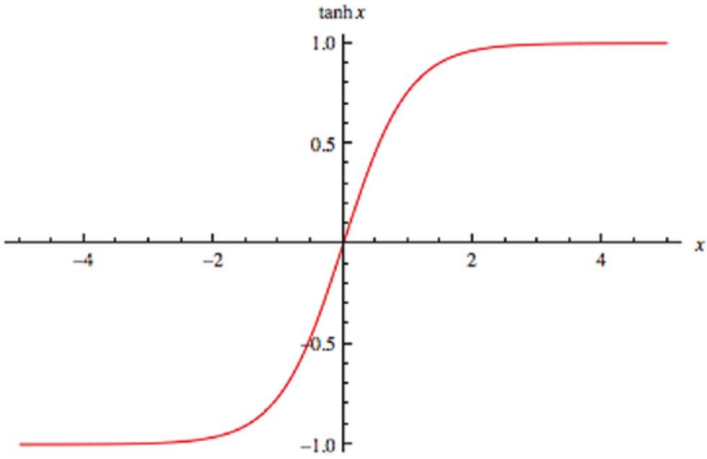
1.4.7.2 fungsi tanh

Dalam matematika, fungsi hiperbolik tangen atau tanh adalah fungsi hiperbolik yang dapat dibedakan. Fungsi ini merupakan versi skala dari fungsi Sigmoid. Fungsi ini merupakan fungsi halus, dan nilai masukannya berada dalam rentang –1 hingga +1.

Dengan gradien yang lebih stabil, maka akan lebih sedikit masalah gradien yang hilang daripada fungsi Sigmoid. Fungsi tanh dapat direpresentasikan seperti Gambar 1-12 dan dapat dilihat pada Persamaan 1-2.

$$Tanh\ x \qquad \frac{aku^{x} \quad ^{x}}{aku^{x} \quad ^{x}} \qquad \text{(Persamaan 1-2)}$$

Representasi grafis tanh juga ditampilkan. Perhatikan perbedaannya antara fungsi Sigmoid dan tanh. Amati bagaimana tanh merupakan versi skala dari fungsi Sigmoid.



Gambar 1-12. *Fungsi tanh; perhatikan bahwa fungsi ini melewati nol dan merupakan versi skala dari fungsi sigmoid. Nilainya berada di antara –1 dan +1. Mirip dengan sigmoid, tanh juga mengalami masalah gradien yang menghilang.*

Fungsi tanh umumnya digunakan dalam lapisan tersembunyi. Fungsi ini membuat nilai rata-rata mendekati nol yang memudahkan pelatihan untuk lapisan berikutnya dalam jaringan. Fungsi ini juga disebut sebagai *pemusatan data.* Fungsi tanh dapat diturunkan dari fungsi sigmoid dan sebaliknya. Mirip dengan sigmoid, fungsi tanh mengalami masalah gradien yang menghilang yang akan kita bahas di bagian selanjutnya.

Sekarang mari kita periksa fungsi aktivasi yang paling populer – ReLU.

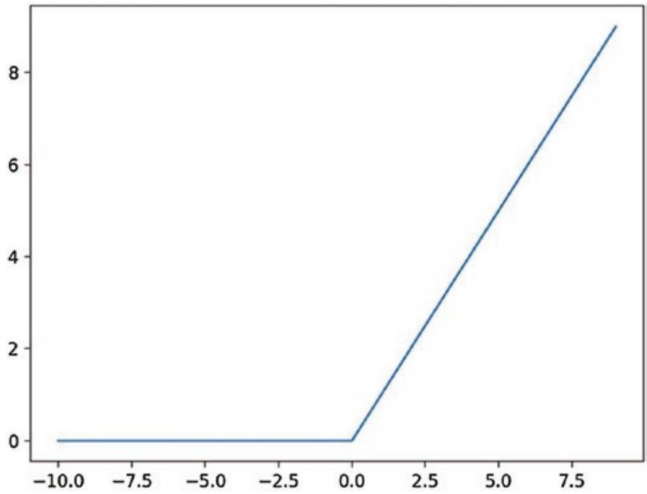
1.4.7.3 Unit Linier Terarah atau ReLU

Rectified Linear Unit atau ReLU adalah fungsi aktivasi yang mendefinisikan positif dari suatu argumen.

ReLU adalah fungsi sederhana, paling murah untuk dihitung, dan dapat dilatih lebih cepat. Tidak terbatas dan tidak berpusat pada nol. dapat dibedakan di semua tempat kecuali nol. Karena fungsi ReLU dapat dilatih lebih cepat, Anda akan mendapati fungsi ini lebih sering digunakan.

Kita dapat memeriksa fungsi ReLU dan grafik pada Gambar 1-13. Persamaan tersebut dapat dilihat pada Persamaan 1-3. Perhatikan bahwa nilainya adalah 0 bahkan untuk nilai negatif, dan dari 0 nilainya mulai menurun.

F(x) = max (0, x) yaitu akan memberikan output sebagai x jika positif, jika tidak 0 (Persamaan 1-3)



Gambar 1-13. *Fungsi ReLU; perlu dicatat bahwa fungsi ini cukup mudah dihitung dan karenanya lebih cepat dalam pelatihan. Fungsi ini digunakan dalam lapisan tersembunyi jaringan. ReLU lebih cepat dilatih daripada sigmoid dan tanh*

Karena fungsi ReLU kurang kompleks, komputasinya lebih murah, dan karenanya banyak digunakan di lapisan tersembunyi untuk melatih jaringan lebih cepat, kami juga akan menggunakan ReLU saat mendesain jaringan. Sekarang kami mempelajari fungsi softmax yang digunakan di lapisan akhir jaringan.

1.4.7.4 Fungsi Softmax

Fungsi softmax digunakan di lapisan akhir Jaringan Neural untuk menghasilkan output dari jaringan. Output dapat berupa klasifikasi akhir gambar untuk kategori yang berbeda.

Fungsi softmax menghitung probabilitas untuk setiap kelas target atas semua kemungkinan. Fungsi ini merupakan fungsi aktivasi yang berguna untuk masalah klasifikasi multikelas dan memaksa Jaringan Neural untuk menghasilkan jumlah 1.

Sebagai contoh, jika inputnya adalah [1,2,3,4,4,3,2,1] dan kita mengambil softmax, maka output yang sesuai adalah [0.024, 0.064, 0.175, 0.475, 0.024, 0.064, 0.175]. Output ini mengalokasikan bobot tertinggi ke yang tertinggi

nilai yang dalam kasus ini adalah 4. Dan karenanya dapat digunakan untuk menyorot nilai tertinggi. Contoh yang lebih praktis adalah jika jumlah kelas yang berbeda untuk suatu gambar adalah mobil, sepeda, atau truk, fungsi softmax akan menghasilkan tiga probabilitas untuk setiap kategori. Kategori yang telah menerima probabilitas tertinggi akan menjadi kategori yang diprediksi.

Kita telah mempelajari fungsi aktivasi yang penting. Namun, masih ada fungsi aktivasi lain seperti Leaky ReLU, ELU, dan sebagainya. Kita akan menjumpai fungsi aktivasi ini di seluruh buku ini. Tabel 1-1 menunjukkan ringkasan fungsi aktivasi untuk referensi cepat.

Tabel 1-1. *Fungsi aktivasi utama dan rinciannya masing-masing*

Pengaktifan Fungsi	Nilai	Positif	Tantangan
Sigmoid	[0,1]	(1) Nonlinier (2) Mudah untuk dikerjakan (3) Berkelanjutan dapat dibedakan (4) Monotonik dan tidak meledak aktivasi	(1) Output tidak berpusat pada nol (2) Masalah gradien yang hilang (3) Lambat dalam pelatihan
tanh	[-1,1]	(1) Mirip dengan fungsi sigmoid (2) Gradien lebih kuat tetapi lebih disukai daripada sigmoid	(1) Masalah gradien menghilang

(lanjutan)

Tabel 1-1. *(lanjutan)*

Pengaktifan Fungsi	Nilai	Positif	Tantangan
Ulang LU	[0,inf]	(1) Tidak linier (2) Mudah dihitung dan karenanya cepat untuk kereta (3) Menyelesaikan masalah gradien yang hilang	(1) Hanya digunakan di lapisan tersembunyi (2) Dapat meledakkan aktivasi (3) Untuk daerah x<0, gradien akan menjadi nol. Oleh karena itu, bobot tidak diperbarui (masalah ReLU yang sekarat)
bocor Ulang LU	maks(0,x)	(1) Varian dari ReLU (2) Memperbaiki ReLU yang sekarat masalah	(1) Tidak dapat digunakan untuk klasifikasi yang kompleks
ELU	[0,inf]	(1) Alternatif untuk ReLU (2) Outputnya lebih halus	(1) Dapat meledakkan aktivasi
Softmax	Menghitung kemungkinan	Umumnya digunakan pada lapisan keluaran	

Fungsi aktivasi membentuk blok penyusun inti suatu jaringan. Di bagian berikutnya, kita membahas laju pembelajaran yang memandu bagaimana jaringan akan mempelajari dan mengoptimalkan pelatihan.

1.4.8 Kecepatan belajar

Untuk Jaringan Neural, *laju pembelajaran* akan menentukan ukuran langkah korektif yang diambil model untuk mengurangi kesalahan. Laju pembelajaran yang lebih tinggi memiliki akurasi yang lebih rendah tetapi waktu pelatihan yang lebih singkat, sedangkan laju pembelajaran yang lebih rendah akan membutuhkan waktu lama untuk pelatihan tetapi memiliki akurasi yang lebih tinggi. Anda harus mendapatkan nilai yang paling optimal.

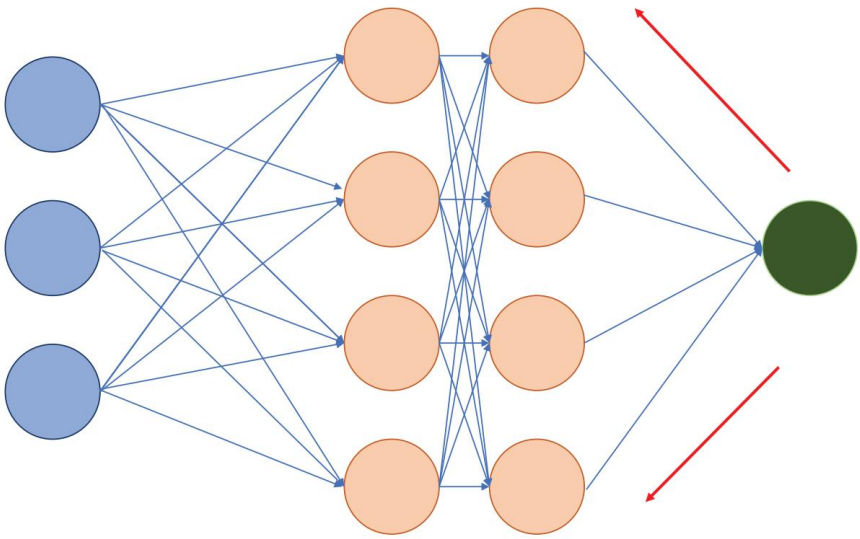
Secara khusus, tingkat pembelajaran akan mengatur penyesuaian yang akan dilakukan terhadap bobot selama pelatihan jaringan. Laju pembelajaran akan secara langsung memengaruhi jumlah waktu yang dibutuhkan jaringan untuk konvergen dan mencapai nilai minimum global. Dalam sebagian besar kasus, memiliki laju pembelajaran 0,01 dapat diterima.

Sekarang kita akan menjelajahi proses yang mungkin paling penting dalam proses pelatihan – algoritma backpropagation.

1.4.9 Backpropagation

Kita telah mempelajari tentang laju pembelajaran di bagian terakhir. Tujuan dari proses pelatihan adalah untuk mengurangi kesalahan dalam prediksi. Sementara laju pembelajaran menentukan ukuran langkah korektif untuk mengurangi kesalahan, backpropagation digunakan untuk menyesuaikan bobot koneksi. Bobot ini diperbarui secara mundur berdasarkan kesalahan. Setelah itu, kesalahan dihitung ulang, penurunan gradien dihitung, dan bobot masing-masing disesuaikan.

Gambar 1-14 menunjukkan proses backpropagation di mana informasi mengalir dari lapisan keluaran kembali ke lapisan tersembunyi.
Perhatikan bahwa aliran informasi bersifat mundur sedangkan perambatan maju di mana informasi mengalir dari kiri ke kanan.



Gambar 1-14. *Backpropagation dalam Jaringan Syaraf Tiruan. Berdasarkan kesalahan, informasi mengalir dari keluaran secara mundur, dan selanjutnya bobot dihitung ulang.*

Mari kita telusuri prosesnya lebih rinci.

Setelah jaringan membuat prediksi, kita dapat menghitung kesalahan yang adalah perbedaan antara nilai yang diharapkan dan nilai yang diprediksi. Ini disebut sebagai fungsi biaya. Berdasarkan nilai biaya, Jaringan Syaraf kemudian menyesuaikan bobot dan biasnya untuk mendapatkan nilai yang paling mendekati nilai sebenarnya atau, dengan kata lain, meminimalkan kesalahan. Ini dilakukan selama backpropagatio

---

Catatan Anda disarankan untuk menyegarkan kalkulus diferensial untuk memahami algoritma backpropagation lebih baik.

Selama backpropagation, parameter koneksi diperbarui dan disesuaikan secara berulang dan berulang. Tingkat penyesuaian ditentukan oleh gradien fungsi biaya sehubungan dengan parameter tersebut. Gradien akan memberi tahu kita ke arah mana kita harus menyesuaikan

bobot untuk meminimalkan biaya. Dan gradien ini dihitung menggunakan aturan rantai. Dengan menggunakan aturan rantai, gradien dihitung untuk satu lapisan pada satu waktu, berulang mundur dari lapisan terakhir ke lapisan pertama. Hal ini dilakukan untuk hindari perhitungan yang berlebihan pada istilah antara dalam aturan rantai.

Terkadang, kita menghadapi masalah hilangnya gradien selama pelatihan Jaringan Syaraf Tiruan. Masalah gradien menghilang adalah fenomena ketika lapisan awal jaringan berhenti belajar karena gradien mendekati nol. Hal ini membuat jaringan tidak stabil, dan lapisan awal jaringan tidak akan dapat mempelajari apa pun. Kita akan kembali membahas gradien menghilang di Bab 6 dan 8.

Sekarang kita akan membahas masalah overfitting – salah satu masalah yang paling umum masalah yang dihadapi selama pelatihan.

1.4.10 Penyesuaian berlebihan

Anda tahu bahwa data pelatihan digunakan oleh jaringan untuk mempelajari atribut dan pola. Namun, kami ingin model Machine Learning kami bekerja dengan baik pada data yang tidak terlihat sehingga kami dapat menggunakannya untuk membuat prediksi.

Untuk mengukur akurasi Machine Learning, kita harus mengevaluasi kinerja dataset pelatihan dan pengujian. Sering kali, jaringan meniru data pelatihan dengan baik dan memperoleh akurasi pelatihan yang baik, sedangkan pada dataset pengujian/validasi, akurasinya menurun. Ini disebut overfitting .

Secara sederhana, apabila jaringan bekerja dengan baik pada dataset pelatihan tetapi tidak begitu baik pada dataset yang tidak terlihat, maka hal tersebut disebut overfitting.

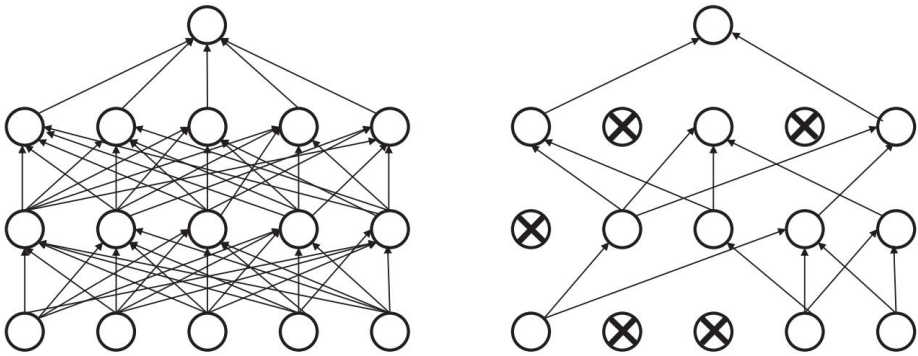
Overfitting itu menyebalkan, dan kita harus melawannya, bukan? Untuk mengatasi overfitting, Anda dapat melatih jaringan dengan lebih banyak data pelatihan. Atau mengurangi kompleksitas jaringan. Dengan mengurangi kompleksitas, kami sarankan untuk mengurangi jumlah bobot, nilai bobot, atau struktur jaringan itu sendiri.

Normalisasi batch dan *Dropout* adalah dua teknik lain untuk memitigasi masalah overfitting.

Normalisasi batch adalah salah satu jenis metode regularisasi. Ini adalah proses normalisasi output dari lapisan dengan mean nol dan

deviasi standar sebesar satu. Ini mengurangi penekanan pada inisialisasi bobot dan dengan demikian mengurangi overfitting.

Dropout adalah teknik lain untuk mengatasi masalah overfitting. Ini adalah metode regularisasi. Selama pelatihan, output dari beberapa layer secara acak dihilangkan atau diabaikan. Efeknya adalah kita mendapatkan jaringan neural yang berbeda untuk setiap kombinasi. Ini juga membuat proses pelatihan menjadi bising. Gambar 1-15 menunjukkan dampak Dropout. Gambar pertama di sebelah kiri (Gambar 1-15(i)) adalah Jaringan Neural standar. Gambar di sebelah kanan (Gambar 1-15(ii)) adalah hasil setelah Dropout.



Gambar 1-15. *Sebelum putus, jaringan mungkin mengalami overfitting. Setelah putus, koneksi acak dan neuron dihilangkan, dan karenanya jaringan tidak akan mengalami overfitting.*

Dengan dropout, kami dapat mengatasi overfitting dalam jaringan kami dan mencapai solusi yang jauh lebih tangguh.

Berikutnya, kita akan mempelajari proses optimasi menggunakan penurunan gradien.

1.4.11 Penurunan gradien

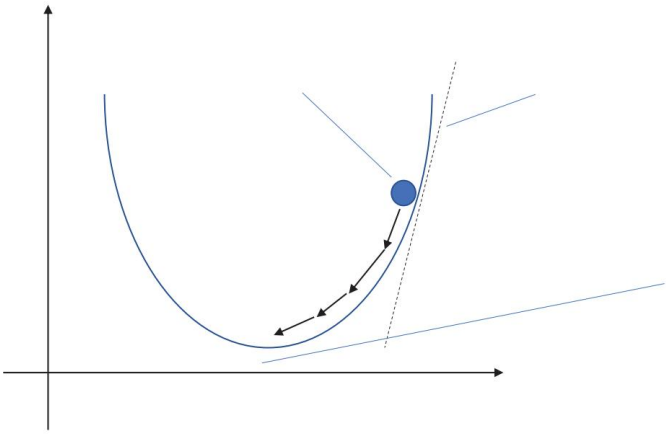
Tujuan dari solusi Machine Learning adalah untuk menemukan nilai yang paling optimal bagi kita. Kita ingin mengurangi kerugian selama fase pelatihan atau memaksimalkan akurasi. Penurunan gradien dapat membantu mencapai tujuan ini.

Penurunan gradien digunakan untuk menemukan nilai minimum global atau nilai maksimum global suatu fungsi. Ini adalah teknik optimasi yang banyak digunakan. Kami melanjutkan ke arah penurunan paling curam secara berulang yang didefinisikan oleh negatif gradien.

Namun, penurunan gradien dapat berjalan lambat pada kumpulan data yang sangat besar. Hal ini disebabkan oleh fakta bahwa satu iterasi algoritma penurunan gradien memprediksi untuk setiap contoh dalam kumpulan data pelatihan. Oleh karena itu, jelas bahwa akan memakan banyak waktu jika kita memiliki ribuan catatan. Untuk situasi seperti itu, kita memiliki *Penurunan Gradien Stokastik.*

Dalam Stochastic Gradient Descent, daripada di akhir batch, koefisien diperbarui untuk setiap contoh pelatihan, sehingga membutuhkan waktu lebih sedikit.

Gambar 1-16 menunjukkan cara kerja penurunan gradien. Perhatikan bagaimana kita dapat bergerak turun ke arah minimum global.



Gambar 1-16. *Konsep penurunan gradien; perhatikan bagaimana tujuannya untuk mencapai minimum global. Tujuan pelatihan jaringan adalah untuk meminimalkan kesalahan yang ditemui*

Kami meneliti cara kami mengoptimalkan fungsi menggunakan penurunan gradien.
Cara untuk memeriksa kemanjuran model pembelajaran mesin adalah dengan mengukur seberapa jauh atau seberapa dekat prediksi dengan nilai sebenarnya. Dan itu didefinisikan menggunakan kerugian yang akan kita bahas sekarang.

1.4.12 Fungsi kerugian

Kerugian adalah ukuran keakuratan model kita. Secara sederhana, kerugian adalah selisih antara nilai aktual dan nilai prediksi. Fungsi yang digunakan untuk menghitung kerugian ini disebut *fungsi kerugian.*

Fungsi kerugian yang berbeda memberikan nilai yang berbeda untuk kerugian yang sama. Dan karena kerugiannya berbeda, kinerja masing-masing model akan berbeda.
juga.

Kami memiliki fungsi kerugian yang berbeda untuk regresi dan klasifikasi masalah. Entropi silang digunakan untuk menentukan fungsi kerugian untuk pengoptimalan. Untuk mengukur kesalahan antara keluaran aktual dan keluaran yang diinginkan, umumnya digunakan kesalahan kuadrat rata-rata. Beberapa peneliti menyarankan untuk menggunakan kesalahan entropi silang alih-alih kesalahan kuadrat rata-rata. Tabel 1-2 memberikan ringkasan cepat untuk berbagai fungsi kerugian.

Tabel 1-2. *Berbagai fungsi kerugian yang dapat digunakan dengan persamaan masing-masing dan penggunaannya*

Fungsi Kerugian	Persamaan untuk Kerugian	Digunakan untuk
Entropi silang	-y(log(p) + (1-y) log(1-p))	Klasifikasi
Kehilangan engsel	maks(0, 1- y *f(x))	Klasifikasi
Kesalahan mutlak	Tentukan nilai x dari persamaan y dan f(x)	Regresi
Kesalahan kuadrat	(y - f(x))2	Regresi
Kerugian Huber	L = ½ (y - f(x))2 , jika |yf(x) <= jika tidak |yf(x)| - ½ 2	Regresi

Kita sekarang telah mempelajari konsep utama Pembelajaran Mendalam. Sekarang mari kita pelajari cara kerja Jaringan Syaraf Tiruan. Kita akan memahami bagaimana berbagai lapisan berinteraksi satu sama lain dan bagaimana informasi diteruskan dari satu lapisan ke lapisan lainnya.

Mari kita mulai!

1.5 Bagaimana Pembelajaran Mendalam bekerja?

Kini Anda tahu bahwa jaringan Deep Learning memiliki berbagai lapisan. Dan Anda juga telah mempelajari konsep-konsep Deep Learning. Anda mungkin bertanya-tanya bagaimana bagian-bagian ini bersatu dan mengatur seluruh pembelajaran. Seluruh proses dapat diperiksa sebagai berikut:

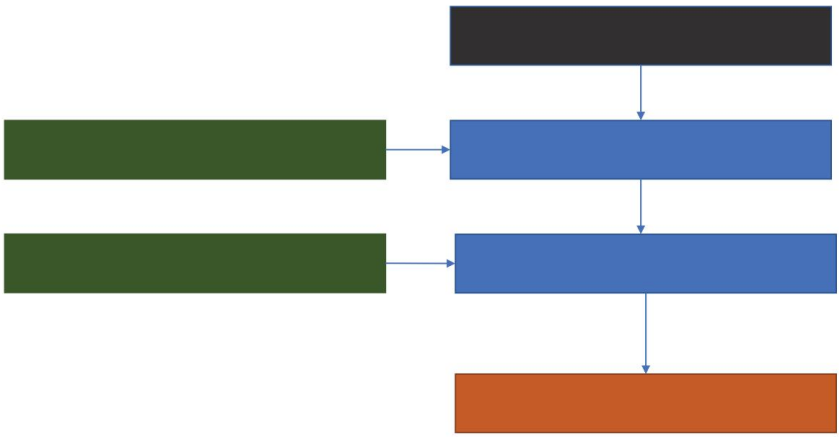
Langkah 1

Anda mungkin bertanya-tanya apa yang sebenarnya dilakukan oleh suatu lapisan, apa yang dicapai oleh suatu lapisan, dan apa yang disimpan dalam bobot masing-masing lapisan. Itu tidak lain hanyalah sekumpulan angka. Secara teknis, transformasi yang diterapkan oleh suatu lapisan harus diparameterisasi oleh bobotnya yang juga disebut sebagai parameter suatu lapisan.

Dan yang dimaksud dengan pembelajaran adalah pertanyaan berikutnya. Pembelajaran untuk Jaringan Neural adalah menemukan kombinasi dan nilai terbaik untuk bobot untuk semua lapisan jaringan sehingga kita dapat mencapai akurasi terbaik. Karena Jaringan Neural dalam dapat memiliki banyak nilai parameter tersebut, kita harus menemukan nilai yang paling optimal untuk semua parameter. Ini tampaknya seperti tugas yang sangat berat mengingat bahwa mengubah satu nilai memengaruhi yang lain.

Mari kita tunjukkan proses dalam Jaringan Syaraf melalui diagram (Gambar 1-17). Periksa lapisan data masukan kita. Dua lapisan transformasi data yang masing-masing memiliki bobot terkait. Lalu kita memiliki prediksi akhir untuk variabel target Y.

Gambar dimasukkan ke dalam lapisan input dan kemudian diubah dalam lapisan transformasi data.



Gambar 1-17. *Data masukan ditransformasikan dalam lapisan transformasi data, bobot ditentukan, dan prediksi awal dibuat*

Langkah 2

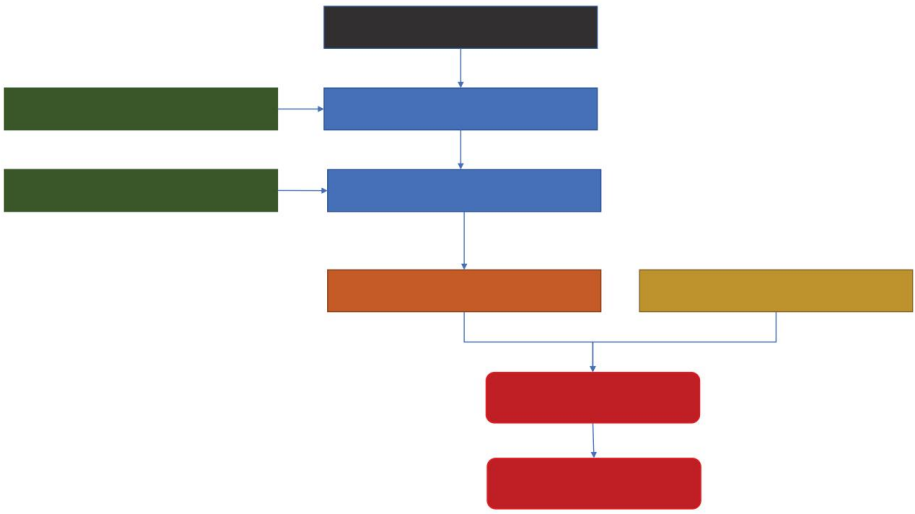
Kami telah membuat kerangka dasar pada langkah 1. Sekarang kita harus mengukur keakuratan jaringan ini.

Kami ingin mengontrol output dari Jaringan Saraf; kita harus membandingkan dan membedakan keakuratan keluaran.

Akurasi Info akan mengacu pada seberapa jauh prediksi kita dari nilai sebenarnya. Sederhananya, seberapa baik atau buruk prediksi kita dari nilai sebenarnya adalah ukuran akurasi.

Pengukuran akurasi dilakukan oleh fungsi kerugian jaringan, yang juga disebut *fungsi objektif.* Fungsi kerugian mengambil prediksi jaringan dan nilai target yang sebenarnya atau aktual. Nilai aktual ini adalah apa yang kami harapkan akan dikeluarkan oleh jaringan. Fungsi kerugian menghitung skor jarak, yang menunjukkan seberapa baik kinerja jaringan.

Mari perbarui diagram yang kita buat pada langkah 1 dengan menambahkan fungsi kerugian dan skor kerugian yang sesuai (Gambar 1-18). Ini membantu mengukur akurasi jaringan.



Gambar 1-18. *Tambahkan fungsi kerugian untuk mengukur akurasi; kerugian adalah perbedaan antara nilai aktual dan nilai prediksi. Pada tahap ini, kita mengetahui kinerja jaringan berdasarkan istilah kesalahan.*

Langkah 3

Seperti yang telah kita bahas sebelumnya, kita harus memaksimalkan akurasi atau menurunkan kerugian. Ini akan membuat solusinya tangguh dan akurat dalam prediksi.

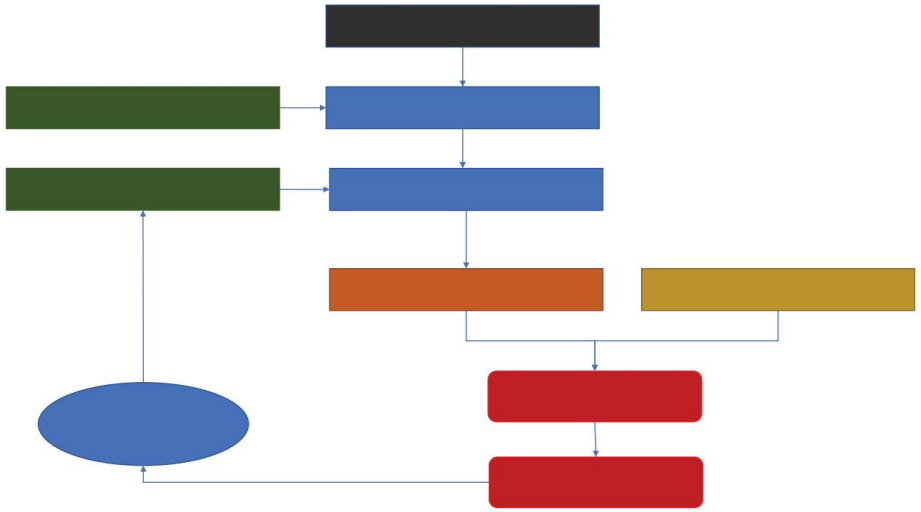
Untuk terus menurunkan kerugian, skor (prediksi – aktual) kemudian digunakan sebagai sinyal umpan balik untuk sedikit menyesuaikan nilai bobot yang dilakukan oleh pengoptimal. Tugas ini dilakukan oleh *algoritma backpropagation* yang terkadang disebut algoritma sentral dalam Deep Learning.

Pada awalnya, beberapa nilai acak diberikan pada bobot, jadi Jaringan menerapkan serangkaian transformasi acak. Dan secara logis, outputnya sangat berbeda dari yang kita harapkan, dan karenanya skor kerugiannya sangat tinggi. Bagaimanapun, ini adalah percobaan pertama!

Namun, ini akan membaik. Saat melatih Jaringan Neural, kami terus-menerus menemukan contoh pelatihan baru. Dan dengan setiap contoh baru, bobot disesuaikan sedikit ke arah yang benar, dan selanjutnya skor kerugian menurun. Kami mengulangi putaran pelatihan ini berkali-kali, dan hasilnya adalah nilai bobot paling optimal yang meminimalkan fungsi kerugian.

Kita kemudian mencapai tujuan kita, yaitu jaringan terlatih dengan kerugian minimal. Artinya, nilai aktual dan prediksi sangat dekat satu sama lain.
Untuk mencapai solusi lengkap, kami meningkatkan mekanisme ini yang terlihat pada Gambar 1-19.

Perhatikan pengoptimal yang memberikan umpan balik yang teratur dan berkelanjutan untuk mencapai solusi terbaik.



Gambar 1-19. *Pengoptimal untuk memberikan umpan balik dan mengoptimalkan bobot; ini adalah proses backpropagation. Ini memastikan bahwa kesalahan dikurangi secara berulang-ulang*

Setelah kita mencapai nilai terbaik untuk jaringan kita, kita menyebutnya

Jaringan kami kini telah terlatih. Kami kini dapat menggunakannya untuk membuat prediksi pada kumpulan data yang belum pernah dilihat sebelumnya.

Sekarang Anda telah memahami berbagai komponen Deep Learning dan bagaimana komponen-komponen tersebut bekerja bersama. Kini saatnya bagi Anda untuk mempelajari semua alat dan pustaka untuk Deep Learning.

1.5.1 Pustaka Pembelajaran Mendalam yang Populer

Ada cukup banyak pustaka Deep Learning yang tersedia. Paket-paket ini memungkinkan kita mengembangkan solusi lebih cepat dan dengan upaya minimum karena sebagian besar pekerjaan berat dilakukan oleh pustaka-pustaka ini.

Di sini kami membahas pustaka yang paling populer.

TensorFlow: TensorFlow (TF) yang dikembangkan oleh Google bisa dibilang merupakan salah satu framework Deep Learning yang paling populer dan banyak digunakan. Framework ini diluncurkan pada tahun 2015 dan sejak saat itu telah digunakan oleh sejumlah bisnis dan merek di seluruh dunia.

Python paling banyak digunakan untuk TF, tetapi C++, Java, C#, JavaScript, dan Julia juga dapat digunakan. Anda harus menginstal pustaka TF pada sistem Anda dan mengimpor pustaka tersebut. Dan pustaka tersebut siap digunakan!

Catatan Kunjungi www.tensorflow.org/install dan ikuti petunjuk untuk menginstal TensorFlow.

Model TF harus dilatih ulang jika terjadi modifikasi pada arsitektur model. Model ini beroperasi dengan grafik komputasi statis, artinya kita mendefinisikan grafik terlebih dahulu, lalu kalkulasi dijalankan.

Aplikasi ini cukup populer karena dikembangkan oleh Google. Aplikasi ini dapat digunakan di perangkat seluler. perangkat seperti iOS dan Android juga.

Keras: Ini adalah salah satu kerangka kerja Deep Learning termudah bagi pemula dan fantastis untuk memahami dan membuat prototipe konsep sederhana. Keras awalnya dirilis pada tahun 2015 dan merupakan salah satu pustaka yang paling direkomendasikan untuk memahami seluk-beluk Jaringan Neural.

Catatan Kunjungi https://keras.io dan ikuti petunjuk untuk menginstal Keras. Tf.keras dapat digunakan sebagai API dan akan sering digunakan dalam buku ini.

Ini adalah solusi yang digerakkan oleh API yang matang. Pembuatan prototipe di Keras difasilitasi hingga batas maksimal. API serialisasi/deserialisasi, panggilan balik, dan streaming data menggunakan generator Python sudah sangat matang. Model-model besar di Keras direduksi menjadi fungsi-fungsi baris tunggal yang membuatnya kurang dapat dikonfigurasi. lingkungan.

PyTorch: Gagasan Facebook PyTorch dirilis pada tahun 2016 dan adalah salah satu pustaka Pembelajaran Mendalam yang populer. Kita dapat menggunakan debugger di PyTorch, misalnya, pdb atau PyCharm. PyTorch beroperasi dengan grafik yang diperbarui secara dinamis dan memungkinkan paralelisme data dan model pembelajaran terdistribusi. Untuk proyek kecil dan pembuatan prototipe, PyTorch harus menjadi pilihan Anda; namun, untuk solusi lintas platform, TensorFlow dikenal lebih baik.

Sonnet: Sonnet DeepMind dikembangkan menggunakan dan di atas TF. Sonnet dirancang untuk aplikasi dan arsitektur Jaringan Neural yang kompleks.

Sonnet membuat objek Python utama yang sesuai dengan bagian tertentu dari Jaringan Syaraf Tiruan (NN). Setelah ini, objek Python ini dihubungkan secara independen ke grafik TensorFlow komputasional.
Ini menyederhanakan desain yang disebabkan oleh pemisahan proses pembuatan objek dan mengaitkannya dengan grafik. Lebih jauh, kemampuan untuk memiliki pustaka berorientasi objek tingkat tinggi menguntungkan karena membantu dalam abstraksi saat kita mengembangkan algoritma Pembelajaran Mesin.

MXNet: MXNet Apache adalah alat Pembelajaran Mendalam yang sangat scalable, mudah digunakan, dan memiliki dokumentasi terperinci. Sejumlah besar bahasa seperti C++, Python, R, Julia, JavaScript, Scala, Go, dan Perl didukung oleh
MXNet.

Ada juga framework lain seperti Swift, Gluon, Chainer, DL4J, dan sebagainya; namun, kami hanya membahas framework yang populer dalam buku ini. Tabel 1-3 memberikan gambaran umum semua kerangka kerja.

Tabel 1-3. *Kerangka kerja Pembelajaran Mendalam Utama dan atributnya masing-masing*

Sumber Kerangka Kerja	Atribut
TensorFlow Open source	Paling populer, dapat bekerja di ponsel juga, TensorBoard menyediakan visualisasi
keras	Solusi matang berbasis API sumber terbuka, sangat mudah digunakan
Obor Py	Sumber terbuka Memungkinkan paralelisme data dan sangat baik untuk pembuatan produk yang cepat
Sonet	Desain Sederhana sumber terbuka, menciptakan objek tingkat tinggi
Jaringan MX	Sumber terbuka Sangat scalable, mudah digunakan
Bahasa Indonesia: MATLAB	Berlisensi Sangat dapat dikonfigurasi, menyediakan kemampuan penerapan

1.6 Ringkasan

Pembelajaran Mendalam merupakan pengalaman belajar berkelanjutan dan membutuhkan disiplin, ketelitian, dan komitmen. Anda telah mengambil langkah pertama dalam perjalanan belajar Anda. Dalam bab pertama ini, kita mempelajari konsep Pemrosesan Gambar dan Pembelajaran Mendalam. Konsep-konsep tersebut merupakan dasar untuk keseluruhan buku dan jalan Anda ke depannya. Kami mengembangkan tiga solusi menggunakan OpenCV.

Di bab berikutnya, Anda akan menyelami TensorFlow dan Keras secara mendalam. Dan Anda akan mengembangkan solusi pertama Anda menggunakan Jaringan Syaraf Konvolusional. Mulai dari merancang jaringan, melatihnya, hingga menerapkannya. Jadi, tetaplah fokus!

LATIHAN ULASAN

1. Apa saja langkah-langkah dalam Pengolahan Gambar?
2. Mengembangkan solusi deteksi objek menggunakan OpenCV.
3. Bagaimana proses pelatihan jaringan Deep Learning?
4. Apa itu overfitting dan bagaimana cara mengatasinya?
5. Apa sajakah fungsi aktivasi?

1.6.1 Bacaan lebih lanjut

1. Pengenalan singkat tentang OpenCV: https://ieeexplore.ieee.org/document/6240859.
2. OpenCV untuk aplikasi visi komputer: www.researchgate.net/publication/301590571_OpenCV_untuk_Aplikasi_Visi_Komputer.
3. Dokumentasi OpenCV dapat diakses di https://docs.opencv.org/.
4. Bacalah makalah berikut ini:
   a . www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf
   b. https://arxiv.org/pdf/1404.7828.pdf

BAB 2

Dasar-dasar Pembelajaran Mendalam untuk Visi Kom

Pikiran bukanlah bejana yang harus diisi, melainkan api yang harus dinyalakan.

—Plutarch

Kita manusia dikaruniai kekuatan pikiran yang luar biasa. Kekuatan ini memungkinkan kita untuk membedakan dan memilah, mengembangkan keterampilan baru, mempelajari seni baru, dan membuat keputusan yang rasional. Kekuatan visual kita tidak terbatas. Kita dapat mengenali wajah tanpa memandang pose dan latar belakang. Kita dapat membedakan objek seperti mobil, anjing, meja, telepon, dan sebagainya tanpa memandang merek dan jenisnya. Kita dapat mengenali warna dan bentuk serta membedakannya dengan jelas dan mudah. Kekuatan ini dikembangkan secara berkala dan sistematis. Di usia muda, kita terus mempelajari atribut objek dan mengembangkan pengetahuan kita. Informasi itu tersimpan dengan aman dalam ingatan kita. Seiring berjalannya waktu, pengetahuan dan pembelajaran ini meningkat. Ini adalah proses yang sangat menakjubkan yang melatih mata dan pikiran kita secara berulang. Sering dikatakan bahwa Pembelajaran Mendalam berasal sebagai mekanisme untuk meniru kekuatan luar biasa ini. Dalam visi komputer, Pembelajaran Mendalam membantu kita mengungkap kemampuan yang dapat digunakan untuk membantu organ

Verdhan, *Visi Komputer Menggunakan Pembelajaran Mendalam,*
https://doi.org/10.1007/978-1-4842-6616-8_2

visi komputer untuk tujuan produktif. Pembelajaran Mendalam telah berkembang pesat dan masih memiliki banyak ruang untuk kemajuan lebih lanjut.

Pada bab pertama, kita mulai dengan dasar-dasar Pembelajaran Mendalam. Pada bab kedua ini, kita akan membangun dasar-dasar tersebut, menyelami lebih dalam, memahami berbagai lapisan Jaringan Syaraf, dan membuat solusi Pembelajaran Mendalam menggunakan Keras dan Python.

Kami akan membahas topik-topik berikut dalam bab ini:

(1) Apa itu Tensor dan bagaimana cara menggunakan TensorFlow

(2) Mengungkap Rahasia Jaringan Syaraf Tiruan Konvolusional

(3) Komponen Jaringan Syaraf Tiruan Konvolusional

(4) Mengembangkan jaringan CNN untuk klasifikasi gambar

2.1 Persyaratan teknis

Kode dan kumpulan data untuk bab ini diunggah di tautan GitHub https://github.com/Apress/computer-vision-using-deep-learning/pohon/utama/Bab2 untuk buku ini. Kami akan menggunakan Jupyter Notebook. Untuk bab ini, CPU sudah cukup baik untuk menjalankan kode, tetapi jika diperlukan Anda dapat menggunakan Google Colaboratory. Anda dapat merujuk ke referensi di akhir buku, jika Anda tidak dapat menyiapkan Google Colab sendiri.

2.2 Pembelajaran Mendalam menggunakan TensorFlow dan Keras

Sekarang mari kita bahas TensorFlow (TF) dan Keras secara singkat. Keduanya bisa dibilang merupakan pustaka sumber terbuka yang paling umum.

TensorFlow (TF) adalah platform untuk Machine Learning buatan Google. *Keras* adalah kerangka kerja yang dikembangkan di atas perangkat DL lain seperti TF, Theano, CNTK, dan sebagainya. Kerangka kerja ini memiliki dukungan bawaan untuk Jaringan Neural konvolusional dan berulang.

Tip Keras adalah solusi berbasis API; sebagian besar pekerjaan berat sudah dilakukan di Keras. Lebih mudah digunakan dan karenanya direkomendasikan untuk pemula.

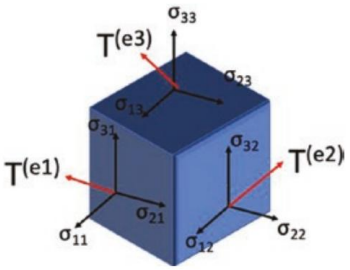
Perhitungan dalam TF dilakukan dengan menggunakan grafik aliran data dimana Data direpresentasikan oleh tepi (yang tidak lain adalah tensor atau array data multidimensi) dan simpul yang merepresentasikan operasi matematika. Jadi, apa sebenarnya tensor itu?

2.3 Apa itu tensor?

Ingat *skalar* dan *vektor* dari pelajaran matematika SMA Anda. Vektor dapat divisualisasikan sebagai skalar dengan arah. Misalnya, kecepatan 50 km/jam adalah skalar, sedangkan 50 km/jam ke arah utara adalah vektor. Ini berarti bahwa vektor adalah besaran skalar dalam arah tertentu. Di sisi lain, *tensor* akan berada dalam beberapa arah, yaitu besaran skalar dalam beberapa arah.

Dalam definisi matematika, tensor adalah objek yang dapat memberikan pemetaan linear antara dua objek aljabar. Objek-objek ini sendiri dapat berupa skalar atau vektor atau bahkan tensor.

Tensor dapat divisualisasikan dalam diagram ruang vektor seperti yang ditunjukkan pada Gambar 2-1.



Gambar 2-1. *Tensor yang direpresentasikan dalam diagram ruang vektor. Tensor adalah besaran skalar dalam berbagai arah dan digunakan untuk menyediakan pemetaan linier antara dua objek aljabar.*

Seperti yang dapat Anda lihat pada Gambar 2-1, tensor memiliki proyeksi di beberapa arah. Tensor dapat dianggap sebagai entitas matematika, yang dijelaskan menggunakan komponen. Komponen dijelaskan dengan mengacu pada basis, dan jika basis terkait ini berubah, tensor harus mengubah dirinya sendiri. Contohnya adalah perubahan koordinat; jika transformasi dilakukan pada basis, nilai numerik tensor juga akan berubah. TensorFlow menggunakan tensor ini untuk melakukan perhitungan yang rumit.

Sekarang mari kita kembangkan pemeriksaan dasar untuk melihat apakah Anda telah menginstal TF dengan benar. Kita akan mengalikan dua konstanta untuk memeriksa apakah instalasinya benar.

Info Lihat Bab 1 jika Anda ingin tahu cara menginstal TensorFlow dan Keras.

---

1. Mari mengimpor TensorFlow:

```
impor tensorflow sebagai tf
```

2. Inisialisasi dua konstanta:

```
a = tf.konstan([1,1,1,1])
b = tf.konstan([2,2,2,2])
```

3. Kalikan kedua konstanta:

```
hasil_produk = tf.kalikan(a, b)
```

4. Cetak hasil akhir:

```
cetak(hasil_produk)
```

Jika Anda memperoleh hasilnya, selamat, Anda siap berangkat!

Sekarang mari kita pelajari Convolutional Neural Network secara detail. Setelah itu, Anda akan siap membuat model klasifikasi gambar pertama Anda.

Menarik, bukan?

2.3.1 Apa itu Jaringan Saraf Konvolusional?

Saat kita sebagai manusia melihat gambar atau wajah, kita dapat langsung mengenalinya. Ini adalah salah satu keterampilan dasar yang kita miliki. Proses identifikasi ini merupakan gabungan dari sejumlah besar proses kecil dan koordinasi antara berbagai komponen vital sistem visual kita.

Jaringan Syaraf Konvolusional atau CNN mampu mereplikasi hal ini kemampuan luar biasa menggunakan Pembelajaran Mendalam.

Pertimbangkan hal ini. Kita harus membuat solusi untuk membedakan antara kucing dan anjing. Atribut yang membedakannya bisa berupa telinga, kumis, hidung, dan sebagainya. CNN berguna untuk mengekstrak atribut gambar yang penting bagi gambar tersebut. Atau dengan kata lain, CNN akan mengekstrak fitur yang membedakan antara kucing dan anjing.

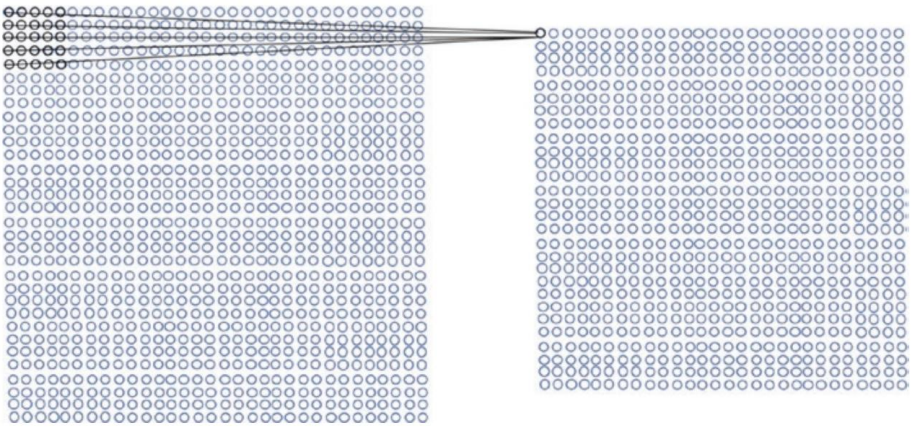
CNN sangat canggih dalam klasifikasi gambar, deteksi objek, pelacakan objek, pemberian keterangan pada gambar, pengenalan wajah, dan sebagainya.

Mari kita menyelami konsep CNN. Kita akan membahas konvolusi terlebih dahulu.

2.3.2 Apa itu konvolusi?

Tujuan utama proses konvolusi adalah untuk mengekstrak fitur-fitur yang penting untuk klasifikasi gambar, deteksi objek, dan sebagainya.
Fitur-fiturnya akan berupa tepi, lengkungan, tetesan warna, garis, dan sebagainya. Setelah proses dilatih dengan baik, proses akan mempelajari atribut-atribut ini pada titik penting dalam gambar. Kemudian, proses dapat mendeteksinya nanti di bagian mana pun dari gambar.

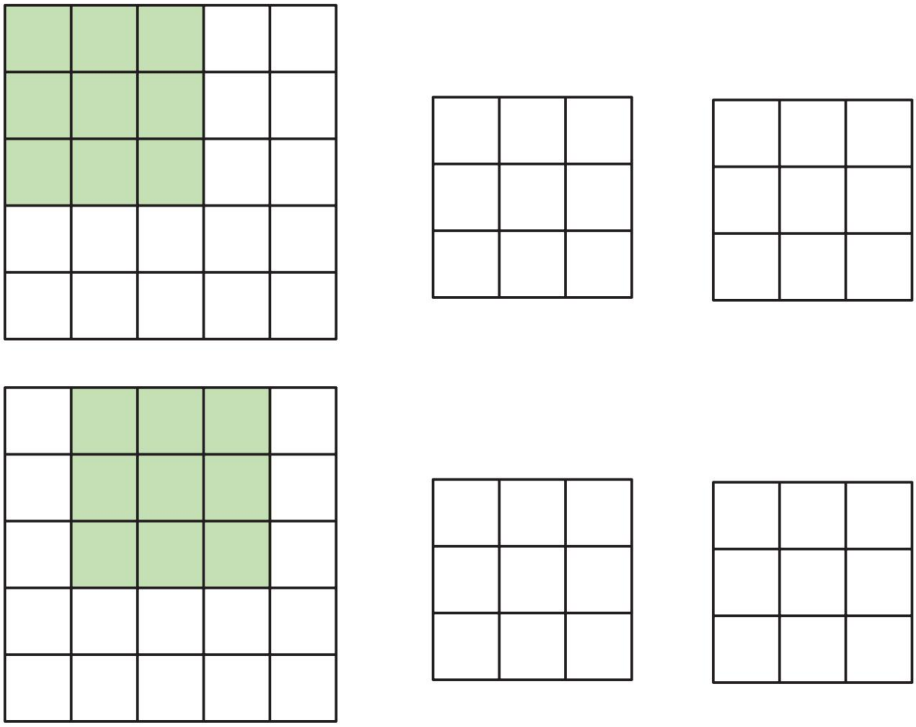
Bayangkan Anda memiliki gambar berukuran 32x32. Itu berarti gambar tersebut dapat direpresentasikan sebagai 32x32x3 piksel jika gambar tersebut berwarna (ingat RGB). Sekarang mari kita pindahkan (atau *buat konvolusi)* area 5x5 di atas gambar ini dan tutupi seluruh gambar. Proses ini disebut *konvolusi.* Dimulai dari kiri atas, area ini dilewatkan ke seluruh gambar. Anda dapat merujuk ke Gambar 2-2 untuk melihat bagaimana gambar 32x32 dikonvolusi oleh filter 5x5.



Gambar 2-2. *Proses konvolusi: lapisan input berada di sebelah kiri, lapisan output berada di sebelah kanan. Gambar 32x32 dikonvolusi oleh filter berukuran 5x5*

Area 5x5 yang dilewatkan pada seluruh gambar disebut *filter* yang terkadang disebut *kernel* atau *detektor fitur.* Wilayah yang disorot dalam Gambar 2-2 disebut *bidang reseptif* filter . Oleh karena itu, kita dapat mengatakan bahwa filter hanyalah matriks dengan nilai yang disebut bobot. Bobot ini dilatih dan diperbarui selama proses pelatihan model. Filter ini bergerak di setiap bagian gambar.

Kita dapat memahami proses konvolusional melalui sebuah contoh dari keseluruhan proses seperti yang ditunjukkan pada Gambar 2-3. Gambar asli berukuran 5x5, dan filter berukuran 3x3. Filter bergerak di atas seluruh gambar dan terus menghasilkan output.



Gambar 2-3. *Konvolusi adalah proses di mana produk berdasarkan elemen dan penjumlahan dilakukan. Pada gambar pertama, outputnya adalah 3, dan pada gambar kedua, filter telah bergeser satu tempat ke kanan dan outputnya adalah 2*

Pada Gambar 2-3, filter 3x3 mengerucut pada seluruh gambar. Filter memeriksa apakah fitur yang ingin dideteksinya ada atau tidak. Filter melakukan proses konvolusi, yang merupakan produk dan jumlah elemen-bijaksana antara dua metrik. Jika suatu fitur ada, keluaran konvolusi dari filter dan bagian gambar akan menghasilkan angka yang tinggi. Jika fitur tidak ada

saat ini, outputnya akan rendah. Oleh karena itu, nilai output ini menunjukkan seberapa yakin filter bahwa fitur tertentu ada dalam gambar.

Kami memindahkan filter ini ke seluruh gambar, sehingga menghasilkan matriks keluaran disebut peta fitur atau peta aktivasi. Peta fitur ini akan memiliki konvolusi filter pada seluruh gambar.

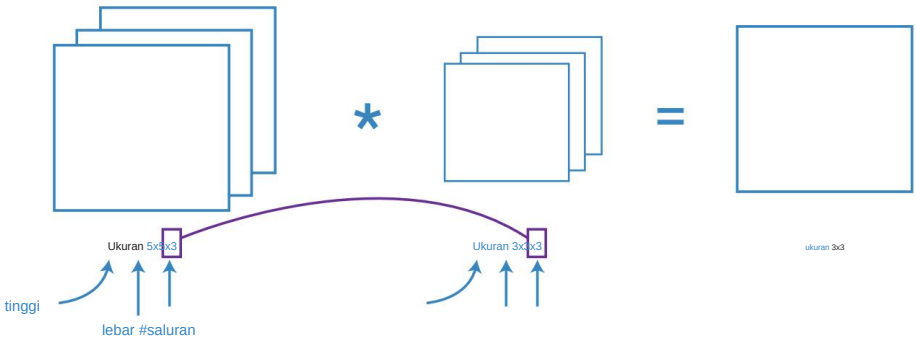
Misalkan dimensi gambar input adalah (n,n) dan dimensi filter adalah (x,x).

Jadi, keluaran setelah lapisan CNN adalah **((n-x+1), (n-x+1)).**

Oleh karena itu, pada contoh di Gambar 2-3, outputnya adalah (5-3+1, 5-3+1) = (3,3).

Ada satu komponen lagi yang disebut *saluran* yang sangat menarik. Saluran adalah kedalaman matriks yang terlibat dalam proses konvolusi. Filter akan diterapkan ke setiap saluran dalam gambar masukan. Kami kembali merepresentasikan keluaran proses pada Gambar 2-4.

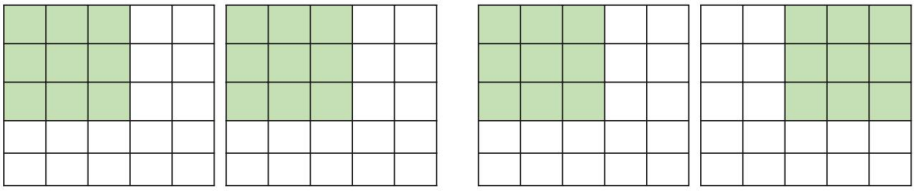
Citra masukan berukuran 5x5x5, dan kami memiliki filter berukuran 3x3x3. Oleh karena itu, keluarannya menjadi citra berukuran (3x3). Perlu dicatat bahwa filter harus memiliki jumlah saluran yang sama persis dengan citra masukan. Pada Gambar 2-4, jumlahnya adalah 3. Hal ini memungkinkan perkalian elemen demi elemen di antara metrik.



Gambar 2-4. *Filter memiliki jumlah saluran yang sama dengan gambar input*

Ada satu komponen lagi yang harus kita ketahui. Filter dapat meluncur di atas gambar input pada interval yang berbeda-beda. Ini disebut sebagai *nilai langkah,* dan ini menunjukkan seberapa banyak filter harus bergerak di setiap langkah.

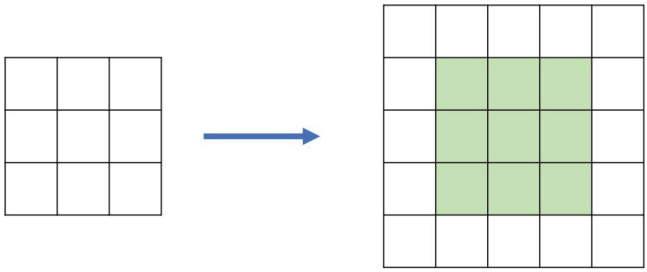
Prosesnya ditunjukkan pada Gambar 2-5. Pada gambar pertama di sebelah kiri, kita melihat gerakan satu langkah, sedangkan di sebelah kanan, gerakan dua langkah telah ditunjukkan.



Gambar 2-5. *Langkah menunjukkan seberapa banyak filter harus bergerak pada setiap langkah. Gambar tersebut menunjukkan dampak langkah pada konvolusi. Pada gambar pertama, kita memiliki langkah 1, sedangkan pada gambar kedua, kita memiliki langkah 2*

Anda pasti setuju bahwa dengan proses konvolusi, kita akan cepat kehilangan piksel di sepanjang tepian. Seperti yang telah kita lihat pada Gambar 2-3, gambar 5x5 menjadi gambar 3x3, dan kehilangan ini akan meningkat dengan jumlah lapisan yang lebih banyak. Untuk mengatasi masalah ini, kita memiliki konsep padding.

Dalam padding, kita menambahkan beberapa piksel ke gambar yang sedang diproses. Misalnya, kita dapat menambahkan angka nol pada gambar seperti yang ditunjukkan pada Gambar 2-6. Penggunaan penambahan angka nol menghasilkan analisis yang lebih baik dan hasil yang lebih baik dari Jaringan Syaraf Tiruan konvolusional.



Gambar 2-6. *Nol padding telah ditambahkan ke gambar masukan. Konvolusi sebagai proses mengurangi jumlah piksel, padding memungkinkan kita untuk mengatasinya*

Sekarang kita telah memahami komponen utama CNN. Mari kita gabungkan konsep-konsep tersebut dan buat sebuah proses kecil. Jika kita memiliki gambar berukuran (nxn) dan kita menerapkan filter berukuran “f,” dengan langkah “s” dan padding “s,” maka output dari proses tersebut akan menjadi

$$((n+2p - f)/s +1), (n+2p - f)/s +1)) \qquad \text{(Persamaan 2-1)}$$

Kita dapat memahami prosesnya seperti yang ditunjukkan pada Gambar 2-7. Kita memiliki gambar input berukuran 37x37 dan filter berukuran 3x3, dan jumlah filter adalah 10, langkahnya adalah 1, dan paddingnya adalah 0. Berdasarkan Persamaan 2-1, outputnya akan berukuran 35x35x10.

Gambar 2-7. *Proses konvolusi di mana kita memiliki filter berukuran 3x3, langkah 1, padding 0, dan jumlah filter 10*

Konvolusi membantu kita dalam mengekstraksi atribut penting dari gambar. Lapisan jaringan yang lebih dekat ke titik asal (gambar input) mempelajari fitur tingkat rendah, sedangkan lapisan terakhir mempelajari fitur tingkat tinggi.
Pada lapisan awal jaringan, fitur seperti tepian, lengkungan, dan lain sebagainya diekstraksi, sedangkan lapisan yang lebih dalam akan mempelajari bentuk yang dihasilkan dari fitur tingkat rendah seperti wajah, objek, dan lain sebagainya.

Namun, perhitungan ini terlihat rumit, bukan? Dan seiring jaringan semakin dalam, kerumitan ini akan meningkat. Jadi, bagaimana kita mengatasinya? Pooling layer adalah jawabannya. Mari kita pahami.

2.3.3 Apa itu Pooling Layer?

Lapisan CNN yang kami pelajari menghasilkan peta fitur input.
Namun seiring dengan semakin dalamnya jaringan, perhitungan ini menjadi rumit.
Hal ini disebabkan oleh fakta bahwa dengan setiap lapisan dan neuron, jumlah dimensi dalam jaringan meningkat. Dan dengan demikian, kompleksitas jaringan secara keseluruhan meningkat.

Dan, ada satu tantangan lagi: setiap penambahan gambar akan berubah peta fitur. Misalnya, rotasi akan mengubah posisi fitur dalam gambar, dan karenanya peta fitur terkait juga akan berubah.

Catatan Sering kali, Anda akan menghadapi ketidaktersediaan data mentah. Augmentasi gambar adalah salah satu metode yang direkomendasikan untuk membuat gambar baru bagi Anda yang dapat berfungsi sebagai data pelatihan.

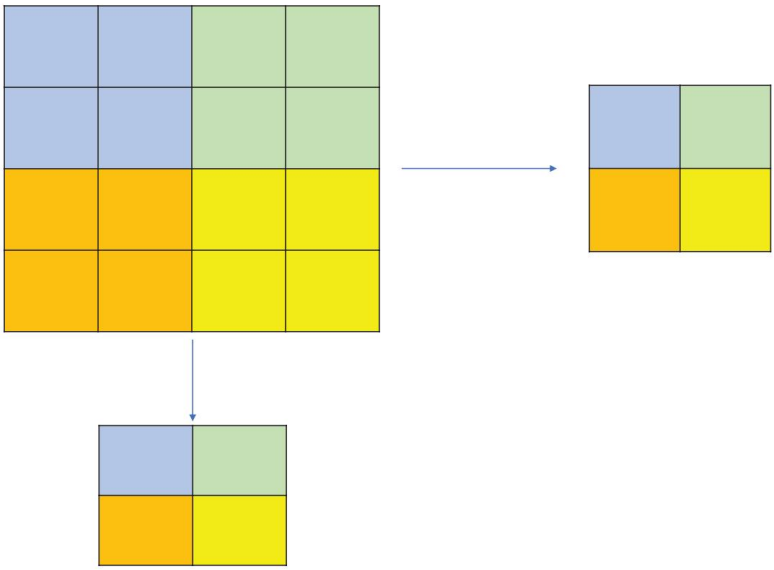
Perubahan pada peta fitur ini dapat diatasi dengan melakukan *downsampling.* downsampling, resolusi gambar input yang lebih rendah dibuat, dan *Pooling Layer* membantu kita dalam hal ini.

Lapisan Penggabungan ditambahkan setelah Lapisan Konvolusional. Setiap peta fitur dioperasikan secara individual, dan kita mendapatkan satu set peta fitur gabungan yang baru. Ukuran filter operasi ini lebih kecil daripada ukuran peta fitur.

Lapisan penggabungan umumnya diterapkan setelah lapisan konvolusional. Lapisan penggabungan dengan piksel 3x3 dan langkah mengurangi ukuran peta fitur dengan faktor 2 yang berarti bahwa setiap dimensi dibagi dua. Misalnya, jika kita menerapkan lapisan penggabungan ke peta fitur berukuran 8x8 (64 piksel), output akan menjadi peta fitur berukuran 4x4 (16 piksel).

Ada dua jenis Pooling Layer.

Pengumpulan Rata-rata dan Pengumpulan Maksimum. Pengumpulan rata-rata menghitung rata-rata setiap nilai peta fitur, sedangkan pengumpulan maksimum mendapatkan nilai maksimum untuk setiap petak peta fitur. Mari kita periksa keduanya seperti yang ditunjukkan pada Gambar 2-8.



Gambar 2-8. *Gambar di sebelah kanan adalah pooling maksimum, sedangkan gambar di bawah adalah pooling rata-rata.*

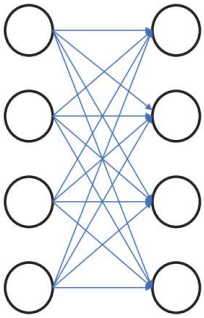
Seperti yang dapat Anda lihat pada Gambar 2-8, lapisan pengumpulan rata-rata melakukan rata-rata dari empat angka, sedangkan pengumpulan maksimum memilih angka maksimum dari empat angka. empat angka.

Ada satu konsep penting lagi tentang lapisan Terhubung Penuh yang harus Anda ketahui sebelum Anda siap membuat model CNN.
Mari kita periksa dan Anda siap melakukannya.

2.3.4 Apa itu Lapisan yang Terhubung Sepenuhnya?

Lapisan Fully Connected mengambil input dari output lapisan sebelumnya (peta aktivasi fitur tingkat tinggi) dan menghasilkan vektor n-dimensi. Di sini, n adalah jumlah kelas yang berbeda.

Misalnya, jika tujuannya adalah untuk memastikan apakah suatu gambar adalah gambar kuda, lapisan yang terhubung penuh akan memiliki fitur tingkat tinggi seperti ekor, kaki, dan sebagainya dalam peta aktivasi. Lapisan yang terhubung penuh tampak seperti Gambar 2-9.



Gambar 2-9. *Lapisan yang terhubung penuh digambarkan di sini*

Lapisan Fully Connected akan melihat fitur-fitur yang paling sesuai dengan kelas tertentu dan memiliki bobot tertentu. Hal ini dilakukan untuk mendapatkan probabilitas yang benar untuk kelas-kelas yang berbeda saat kita mendapatkan hasil perkalian antara bobot dan lapisan sebelumnya.

Sekarang Anda telah memahami CNN dan komponen-komponennya. Sekarang saatnya untuk memukul kode. Anda akan membuat solusi Pembelajaran Mendalam pertama Anda untuk mengklasifikasikan antara kucing dan anjing. Semoga berhasil!

2.4 Mengembangkan solusi DL menggunakan CNN

Sekarang kita akan membuat solusi Deep Learning menggunakan CNN. “Hello World” dari Deep Learning secara umum disebut sebagai kumpulan data MNIST. Kumpulan ini merupakan kumpulan angka numerik yang ditulis tangan seperti yang digambarkan pada Gambar 2-10.

Gambar 2-10. *Dataset MNIST: “Hello World” untuk pengenalan gambar*

Ada sebuah makalah terkenal tentang pengenalan gambar MNIST (deskripsi diberikan di akhir bab). Untuk menghindari pengulangan, kami mengunggah seluruh kode di GitHub. Anda disarankan untuk memeriksa kode tersebut.

Sekarang kita akan mulai membuat model klasifikasi gambar!

Dalam solusi Deep Learning pertama ini, kami ingin membedakan antara kucing dan anjing berdasarkan gambarnya. Kumpulan data tersedia di www.kaggle.com/c/anjing-vs-kucing.

Langkah-langkah yang harus diikuti tercantum di sini:

1. Pertama, mari kita buat datasetnya:

    a. Unduh kumpulan data dari Kaggle. Ekstrak zip kumpulan data.

    b. Anda akan menemukan dua folder: test dan train. Hapus folder tersebut folder pengujian karena kita akan membuat folder pengujian kita sendiri.

    c. Di dalam folder train dan test, buat dua subfolder – kucing dan anjing – dan letakkan gambar di folder masing-masing.

d. Ambil beberapa gambar (saya mengambil 2000) dari folder “train>cats” dan taruh di folder “test>cats”.

e. Ambil beberapa gambar (saya mengambil 2000) dari folder “train>dogs” dan taruh di folder “test>dogs”.

f. Dataset Anda siap digunakan.

2. Impor pustaka yang dibutuhkan sekarang. Kita akan mengimpor sekuensial, pooling, activation, dan flatten lapisan dari keras. Impor numpy juga.

Catatan Dalam Referensi buku ini, kami memberikan deskripsi masing-masing lapisan dan fungsinya masing-masing.

```
dari keras.models impor Sequential
dari keras.layers impor Conv2D, Aktivasi, MaxPooling2D,
Padat, Ratakan, Dropout
impor numpy sebagai np
```

3. Inisialisasi model, variabel pengklasifikasi catDogImage
Di Sini:

```
catDogImageclassifier = Berurutan()
```

4. Sekarang, kita akan menambahkan lapisan ke jaringan kita. Conv2D akan menambahkan lapisan konvolusional dua dimensi yang akan memiliki 32 filter. 3,3 mewakili ukuran filter (3 baris, 3 kolom). Bentuk gambar masukan berikut adalah 64*64*3 – tinggi*lebar*RGB. Setiap angka mewakili intensitas piksel (0–255).

```
catDogImageclassifier.tambahkan(Conv2D(32,(3,3),
bentuk_input=(64,64,3)))
```

5. Output dari lapisan terakhir akan berupa peta fitur.
   Data pelatihan akan bekerja di sana dan mendapatkan
   beberapa peta fitur.

6. Mari tambahkan fungsi aktivasi. Kami menggunakan ReLU
   (Rectified Linear Unit) untuk contoh ini. Dalam keluaran
   peta fitur dari lapisan sebelumnya, fungsi aktivasi akan
   mengganti semua piksel negatif dengan nol.

Catatan Ingat kembali definisi ReLU; yaitu max(0,x). ReLU mengizinkan nilai positif sementara mengganti nilai negatif dengan 0. Secara umum, ReLU
hanya digunakan pada lapisan tersembunyi.

```
catDogImageclassifier.tambahkan(Aktivasi('relu'))
```

7. Sekarang kita tambahkan lapisan Max Pooling karena
   kita tidak ingin jaringan kita menjadi terlalu rumit
   secara komputasi.

```
catDogImageclassifier.tambahkan(MaxPooling2D
(ukuran_pool =(2,2)))
```

8. Selanjutnya, kita tambahkan ketiga blok konvolusional. Setiap
   blok memiliki Cov2D, ReLU, dan Max Pooling Layer.

```
catDogImageclassifier.tambahkan(Conv2D(32,(3,3)))
catDogImageclassifier.tambahkan(Aktivasi('relu'))
catDogImageclassifier.tambahkan(MaxPooling2D(ukuran_pool =(2,2)))
catDogImageclassifier.tambahkan(Conv2D(32,(3,3)))
catDogImageclassifier.tambahkan(Aktivasi('relu'))
catDogImageclassifier.tambahkan(MaxPooling2D(ukuran_pool =(2,2)))
```

```
catDogImageclassifier.tambahkan(Conv2D(32,(3,3
catDogImageclassifier.tambahkan(Aktivasi('relu'))
catDogImageclassifier.tambahkan(MaxPooling2D(ukuran_pool =(2,2)))
```

9. Sekarang, mari kita ratakan kumpulan data yang akan mengubah matriks peta fitur gabungan menjadi satu kolom.

```
catDogImageclassifier.tambahkan(Ratakan())
```

10. Tambahkan fungsi padat sekarang diikuti oleh ReLU pengaktifan:

```
catDogImageclassifier.tambahkan(Padat(64))
catDogImageclassifier.tambahkan(Aktivasi('relu'))
```

Info Menurut Anda mengapa kita memerlukan fungsi nonlinier seperti tanh, ReLU, dan sebagainya? Jika Anda hanya menggunakan fungsi linier, outputnya juga akan linier. Oleh karena itu, kita menggunakan fungsi nonlinier di lapisan tersembunyi.

---

11. Overfitting merupakan suatu gangguan. Kita akan menambahkan lapisan Dropout untuk mengatasi overfitting selanjutnya:

```
catDogImageclassifier.tambahkan(Dropout(0.5))
```

12. Tambahkan satu lapisan lagi yang terhubung penuh untuk mendapatkan keluaran dalam kelas n-dimensi (keluarannya adalah vektor).

```
catDogImageclassifier.tambahkan(Padat(1))
```

13. Tambahkan fungsi Sigmoid untuk mengonversi ke probabilitas:

```
catDogImageclassifier.tambahkan(Aktivasi('sigmoid'))
```

14. Mari kita cetak ringkasan jaringan.

```
catDogImageclassifier.summary()
```

Kita melihat keseluruhan jaringan pada gambar berikut:

```
1 catDogImageclassifier.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_27 (Conv2D)           (None, 62, 62, 32)        896
_________________________________________________________________
activation_22 (Activation)   (None, 62, 62, 32)        0
_________________________________________________________________
max_pooling2d_20 (MaxPooling (None, 31, 31, 32)        0
_________________________________________________________________
conv2d_28 (Conv2D)           (None, 29, 29, 32)        9248
_________________________________________________________________
activation_23 (Activation)   (None, 29, 29, 32)        0
_________________________________________________________________
max_pooling2d_21 (MaxPooling (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_29 (Conv2D)           (None, 12, 12, 32)        9248
_________________________________________________________________
activation_24 (Activation)   (None, 12, 12, 32)        0
_________________________________________________________________
max_pooling2d_22 (MaxPooling (None, 6, 6, 32)          0
_________________________________________________________________
conv2d_30 (Conv2D)           (None, 4, 4, 32)          9248
_________________________________________________________________
activation_25 (Activation)   (None, 4, 4, 32)          0
_________________________________________________________________
max_pooling2d_23 (MaxPooling (None, 2, 2, 32)          0
_________________________________________________________________
flatten_2 (Flatten)          (None, 128)               0
_________________________________________________________________
dense_3 (Dense)              (None, 64)                8256
_________________________________________________________________
activation_26 (Activation)   (None, 64)                0
_________________________________________________________________
dropout_2 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 65
_________________________________________________________________
activation_27 (Activation)   (None, 1)                 0
=================================================================
Total params: 36,961
Trainable params: 36,961
Non-trainable params: 0
```

Kita dapat melihat jumlah parameter dalam jaringan kita adalah 36.961. Anda disarankan untuk bereksperimen dengan berbagai struktur jaringan dan mengukur dampaknya.

15. Sekarang mari kita kompilasi jaringannya. Kita menggunakan pengoptimal rmsprop menggunakan penurunan gradien, lalu kita menambahkan kerugian atau fungsi biaya.

```
catDogImageclassifier.compile(pengoptimal = 'rmsprop',
 kerugian = 'binary_crossentropy',
metrik =['akurasi'])
```

16. Sekarang kita melakukan augmentasi data di sini (zoom, scale, dll.). Ini juga akan membantu mengatasi masalah overfitting. Kita menggunakan ImageDataGenerator fungsi untuk melakukan hal ini:

```
dari keras.preprocessing.image impor
ImageDataGenerator
train_datagen = ImageDataGenerator(skala_ulang =
1./255, rentang_geser = 0.25, rentang_zoom = 0.25,
pembalikan_horizontal = Benar)
test_datagen = ImageDataGenerator(skala ulang = 1./255)
```

17. Muat data pelatihan:

```
training_set = train_datagen.flow_from_directory ('/Pengguna/
 AnjingKucing/latih',ukuran_target=(64,6 4), ukuran_batch=
32,mode_kelas='biner')
```

18. Muat data pengujian:

```
test_set = test_datagen.flow_dari_direktori ('/Pengguna/
KucingAnjing/uji', ukuran_target = (64,64),
ukuran_batch = 32,
class_mode = 'biner')
```

19. Mari kita mulai pelatihannya sekarang.

```
dari IPython.display impor tampilan
dari PIL impor Gambar catDogImageclassifier.fit_
generator(set_pelatihan, langkah_per_epoch =625,
epoch = 10, data_validasi = set_uji, langkah_validasi =
1000)
```

Jumlah langkah per epoch adalah 625, dan jumlah epoch adalah 10. Jika kita memiliki 1000 gambar dan ukuran batch 10, jumlah langkah yang diperlukan adalah 100 (1000/10).

Info Jumlah epoch berarti jumlah lintasan lengkap melalui set data pelatihan lengkap. Batch adalah jumlah contoh pelatihan dalam satu batch, sedangkan iterasi adalah jumlah batch yang diperlukan untuk menyelesaikan satu epoch.

Bergantung pada kompleksitas jaringan, jumlah periode yang diberikan, dan sebagainya, kompilasi akan memerlukan waktu. Kumpulan data pengujian dilewatkan sebagai validation_data di sini.

Outputnya ditunjukkan pada Gambar 2-11.

```
from IPython.display import display
from PIL import Image
catDogImageclassifier.fit_generator(training_set,
                         steps_per_epoch =625,
                         epochs = 10,
                         validation_data =test_set,
                         validation_steps = 1000)
```

```
Epoch 1/10
625/625 [==============================] - 185s 296ms/step - loss: 0.6721 - acc: 0.5822 - val_loss: 0.6069 - val_acc:
0.6610
Epoch 2/10
625/625 [==============================] - 152s 243ms/step - loss: 0.5960 - acc: 0.6831 - val_loss: 0.5151 - val_acc:
0.7543
Epoch 3/10
625/625 [==============================] - 151s 242ms/step - loss: 0.5452 - acc: 0.7217 - val_loss: 0.4891 - val_acc:
0.7545
Epoch 4/10
625/625 [==============================] - 150s 239ms/step - loss: 0.5069 - acc: 0.7568 - val_loss: 0.4657 - val_acc:
0.7743
Epoch 5/10
625/625 [==============================] - 150s 240ms/step - loss: 0.4813 - acc: 0.7713 - val_loss: 0.4407 - val_acc:
0.7925
Epoch 6/10
625/625 [==============================] - 152s 243ms/step - loss: 0.4526 - acc: 0.7866 - val_loss: 0.4374 - val_acc:
0.7924
Epoch 7/10
625/625 [==============================] - 151s 241ms/step - loss: 0.4458 - acc: 0.7953 - val_loss: 0.3891 - val_acc:
0.8324
Epoch 8/10
625/625 [==============================] - 151s 242ms/step - loss: 0.4177 - acc: 0.8123 - val_loss: 0.3917 - val_acc:
0.8221
Epoch 9/10
625/625 [==============================] - 155s 248ms/step - loss: 0.4158 - acc: 0.8158 - val_loss: 0.3947 - val_acc:
0.8176
Epoch 10/10
625/625 [==============================] - 151s 241ms/step - loss: 0.4021 - acc: 0.8201 - val_loss: 0.3783 - val_acc:
0.8221

<keras.callbacks.History at 0x1a376b5160>
```

Gambar 2-11. *Output hasil pelatihan selama 10 epoch*

Seperti yang terlihat pada hasil, pada epoch terakhir, kami memperoleh akurasi validasi sebesar 82,21%. Kita juga dapat melihat bahwa pada Epoch 7 kami memperoleh akurasi sebesar 83,24% yang lebih baik daripada akurasi akhir.

Selanjutnya, kita ingin menggunakan model yang dibuat di Epoch 7 karena akurasinya paling baik. Kita dapat mencapainya dengan menyediakan titik pemeriksaan antara pelatihan dan penyimpanan versi tersebut. Kita akan melihat proses pembuatan dan penyimpanan titik pemeriksaan di bab-bab berikutnya.

Kami telah menyimpan model akhir sebagai file di sini. Model tersebut kemudian dapat dimuat lagi saat diperlukan. Model akan disimpan sebagai HDF5 berkas, dan dapat digunakan kembali nanti.

```
catDogImageclassifier.simpan('catdog_cnn_model.h5')
```

20. Muat model yang disimpan menggunakan load_model:

```
dari keras.models impor load_model
catDogImageclassifier = muat_model('catdog_cnn_model.h5')
```

21. Periksa bagaimana model memprediksi gambar yang tidak terlihat.

Berikut gambar (Gambar 2-12) yang kami gunakan untuk membuat prediksi. Rasakan bebas menguji solusi dengan gambar yang berbeda.

Gambar 2-12. *Gambar anjing untuk menguji keakuratan model*

Dalam blok kode berikut, kami membuat prediksi pada gambar sebelumnya menggunakan model yang telah kami latih.

22. Muat pustaka dan gambar dari folder. Anda harus mengubah lokasi file dalam cuplikan kode di bawah ini.

```
impor numpy sebagai np
dari keras.preprocessing impor gambar
an_image =gambar.load_img('/Pengguna/vaibhavverdhan/Buku
Menulis/2.jpg',target_size=(64,64))
```

Gambar sekarang diubah menjadi serangkaian angka:

```
sebuah_gambar = gambar.img_ke_array(sebuah_gambar)
```

Sekarang mari kita perluas dimensi gambar. Ini akan meningkatkan daya prediksi model. Digunakan untuk memperluas bentuk array dan menyisipkan sumbu baru yang muncul pada posisi sumbu dalam bentuk array yang diperluas.

```
sebuah_gambar =np.expand_dims(sebuah_gambar, sumbu =0)
```

Sekarang saatnya memanggil metode prediksi. Kami menetapkan ambang batas probabilitas pada 0,5. Anda disarankan untuk menguji pada beberapa ambang batas dan memeriksa keakuratannya.

```
putusan = catDogImageclassifier.predict(sebuah_gambar) jika putusan[0]
 [0] >= 0,5:
prediksi = 'anjing'
kalau tidak:
prediksi = 'kucing'
```

23. Mari kita cetak prediksi akhir kita.

```
cetak(prediksi)
```

Model tersebut memprediksi bahwa gambar tersebut adalah seekor “anjing”.

Di sini, dalam contoh ini, kami merancang Jaringan Neural menggunakan Keras. Kami melatih gambar menggunakan gambar kucing dan anjing dan mengujinya. Dimungkinkan juga untuk melatih sistem multiklasifikasi jika kami bisa mendapatkan gambar untuk setiap kelas.

Selamat! Anda baru saja menyelesaikan kasus penggunaan klasifikasi gambar kedua menggunakan Deep Learning. Gunakan untuk melatih kumpulan data gambar Anda sendiri. Bahkan dimungkinkan untuk membuat model klasifikasi multikelas.

Sekarang Anda mungkin berpikir bagaimana Anda akan menggunakan model ini untuk membuat prediksi secara real time. File model yang dikompilasi (misalnya, 'catdog_cnn_model.h5') akan disebarkan ke server untuk membuat prediksi. Kami akan membahas penyebaran model secara terperinci di bab terakhir buku ini.

Dengan ini, kita sampai pada akhir bab kedua. Anda dapat melanjutkan ke ringkasan sekarang.

2.5 Ringkasan

Gambar merupakan sumber informasi dan pengetahuan yang kaya. Kita dapat memecahkan banyak masalah bisnis dengan menganalisis kumpulan data gambar. CNN memimpin revolusi AI khususnya untuk gambar dan video. CNN digunakan dalam industri medis, manufaktur, ritel, BFSI, dan sebagainya. Cukup banyak penelitian yang dilakukan menggunakan CNN.

Solusi berbasis CNN cukup inovatif dan unik. Ada banyak tantangan yang harus diatasi saat merancang solusi berbasis CNN yang inovatif. Pilihan jumlah lapisan, jumlah neuron di setiap lapisan, fungsi aktivasi yang akan digunakan, fungsi kerugian, pengoptimal, dan sebagainya bukanlah pilihan yang mudah. Hal itu bergantung pada kompleksitas masalah bisnis, kumpulan data yang ada, dan daya komputasi yang tersedia. Kemanjuran solusi sangat bergantung pada kumpulan data yang tersedia. Jika kita memiliki tujuan bisnis yang ditetapkan dengan jelas yang dapat diukur, tepat, dan dapat dicapai, jika kita memiliki kumpulan data yang representatif dan lengkap, dan jika kita memiliki daya komputasi yang cukup, banyak masalah bisnis dapat diatasi menggunakan Pembelajaran Mendalam.

Pada bab pertama buku ini, kami memperkenalkan visi komputer dan Pembelajaran Mendalam. Pada bab kedua ini, kami mempelajari konsep lapisan konvolusional, penggabungan, dan lapisan yang terhubung sepenuhnya. Anda akan menggunakan konsep-konsep ini di sepanjang perjalanan Anda. Dan Anda juga mengembangkan model klasifikasi gambar menggunakan Pembelajaran Mendalam.

Tingkat kesulitan akan meningkat dari bab berikutnya dan seterusnya ketika kita mulai dengan arsitektur jaringan. Arsitektur jaringan menggunakan blok-blok penyusun yang telah kita pelajari di dua bab pertama. Blok-blok tersebut dikembangkan oleh para ilmuwan dan peneliti di berbagai organisasi dan universitas untuk memecahkan berbagai masalah yang rumit. Kita akan mempelajari jaringan-jaringan tersebut dan mengembangkan solusi Python juga. Jadi, jangan lewatkan kesempatan ini!

Sekarang, Anda seharusnya dapat menjawab pertanyaan dalam latihan ini!

PERTANYAAN ULASAN

Anda disarankan untuk menjawab pertanyaan berikut:

1. Apa proses konvolusi di CNN dan bagaimana outputnya? dihitung?
2. Mengapa kita membutuhkan fungsi nonlinier dalam lapisan tersembunyi?
3. Apa perbedaan antara pengumpulan maksimal dan rata-rata?
4. Apa yang dimaksud dengan putus sekolah?
5. Unduh data gambar pemandangan alam di seluruh dunia dari www.kaggle.com/puneet6060/intel-image- classification dan mengembangkan model klasifikasi gambar menggunakan CNN.
6. Unduh dataset Fashion MNIST dari https://github. com/zalandoresearch/fashion-mnist dan mengembangkan model klasifikasi gambar.

2.5.1 Bacaan lebih lanjut

Anda disarankan untuk membaca dokumen-dokumen berikut ini:

1. Baca “Menilai Empat Jaringan Saraf pada
   “Dataset Pengenalan Angka Tulis Tangan (MNIST)” di https://
   arxiv.org/pdf/1811.08278.pdf.
2. Pelajari makalah penelitian “A Survey of the Recent
   Architectures of Deep Convolutional Neural
   Networks” di https://arxiv.org/pdf/
   1901.06032.pdf .
3. Bacalah makalah penelitian “Memahami
   Jaringan Syaraf Konvolusional dengan
   “Model Matematika” di https://arxiv.org/
   pdf/1609.04112.pdf.
4. Lakukan Evaluasi Operasi Penggabungan di
   Arsitektur Konvolusional untuk Pengenalan Objek” di
   http://ais.uni-bonn.de/papers/icann2010_
   maxpool.pdf.