2
Pengaturan dan Pengenalan Deep Kerangka Pembelajaran

Pada titik ini, Anda sudah familier dengan **machine learning (ML)** dan **deep learning (DL)** - ini hebat! Anda seharusnya merasa siap untuk mulai membuat persiapan untuk menulis dan menjalankan program Anda sendiri. Bab ini membantu Anda dalam proses menyiapkan TensorFlow dan Keras, dan memperkenalkan kegunaan dan tujuan keduanya dalam deep learning.
Dopamin disajikan sebagai kerangka pembelajaran penguatan baru yang akan kita gunakan nanti. Bab ini juga secara singkat memperkenalkan pustaka pembelajaran mendalam lainnya yang penting untuk diketahui.

Topik yang akan dibahas dalam bab ini adalah sebagai berikut:

- Pengantar Kolaborasi
- Pengenalan dan pengaturan TensorFlow
- Pengenalan dan pengaturan Keras
- Pengantar PyTorch
- Pengantar Dopamin
- Pustaka pembelajaran mendalam lainnya

Pengantar Kolaborasi

Apa itu Colaboratory? Colaboratory adalah alat penelitian berbasis web untuk melakukan pembelajaran mesin dan pembelajaran mendalam. Pada dasarnya alat ini seperti Jupyter Notebook. Colaboratory menjadi sangat populer akhir-akhir ini karena tidak memerlukan pengaturan apa pun.

Sepanjang buku ini, kita akan menggunakan Python 3 yang berjalan di Colaboratory yang telah menginstal semua pustaka yang mungkin kita perlukan.

Colaboratory gratis untuk digunakan dan kompatibel dengan sebagian besar browser utama. Perusahaan yang bertanggung jawab atas pengembangan alat Colaboratory adalah Google™ . Berbeda dengan notebook Jupyter, di Colaboratory Anda menjalankan semuanya di cloud dan bukan di komputer Anda sendiri. Inilah kendalanya: Anda memerlukan akun Google karena semua notebook Colaboratory disimpan di ruang Google Drive pribadi Anda. Namun, jika Anda tidak memiliki akun Google, Anda masih dapat terus membaca untuk melihat cara menginstal setiap bagian pustaka Python yang Anda perlukan untuk menjalankan semuanya sendiri. Namun, saya sangat menyarankan Anda membuat akun Google, setidaknya untuk mempelajari pembelajaran mendalam menggunakan notebook Colaboratory dalam buku ini.

Saat Anda menjalankan kode di Colaboratory, kode tersebut berjalan pada mesin virtual khusus, dan inilah bagian yang menyenangkan: Anda dapat mengalokasikan GPU untuk digunakan! Atau Anda juga dapat menggunakan CPU jika Anda mau. Setiap kali Anda tidak menjalankan sesuatu, Colaboratory akan mendealokasikan sumber daya (Anda tahu, karena kita semua ingin bekerja), tetapi Anda dapat menghubungkannya kembali kapan saja.

Jika Anda siap, lanjutkan dan navigasikan ke tautan ini: https://colab.research.google.com/

Jika Anda tertarik dengan informasi lebih lanjut dan pengenalan lebih lanjut tentang Colaboratory, cari *Selamat Datang di Colaboratory!.* Sekarang setelah Anda mengakses tautan sebelumnya, mari kita mulai dengan TensorFlow.

Mulai sekarang, kami akan menyebut **Colaboratory** dengan sebutan **Colab .** Begitulah orang-orang menyebutnya.

Pengenalan dan pengaturan TensorFlow

TensorFlow (TF) memiliki kata Tensor dalam namanya , yang merupakan sinonim dari vektor. Jadi, TF adalah kerangka kerja Python yang dirancang untuk unggul dalam operasi vektor yang berkaitan dengan pemodelan jaringan saraf. Ini adalah pustaka paling populer untuk pembelajaran mesin.

Sebagai ilmuwan data, kami lebih memilih TF karena gratis, sumber terbuka dengan basis pengguna yang kuat, dan menggunakan penelitian terkini tentang eksekusi operasi tensor berbasis grafik.

Pengaturan

Sekarang mari kita mulai dengan petunjuk untuk menyiapkan atau memverifikasi bahwa Anda memiliki pengaturan yang tepat:

1. Untuk memulai instalasi TF, jalankan perintah berikut di Colaboratory Anda:

```
%tensorflow_versi 2.x !pip instal
tensorflow
```

Ini akan menginstal sekitar 20 pustaka yang dibutuhkan untuk menjalankan TF, termasuk numpy, misalnya.

Perhatikan tanda seru (!) di awal perintah? Beginilah cara Anda menjalankan perintah shell di Colaboratory. Misalnya, katakanlah Anda ingin menghapus file bernama model.h5, maka Anda akan menjalankan perintah !rm model.h5.

2. Jika eksekusi instalasi berjalan dengan baik, Anda akan dapat menjalankan perintah berikut, yang akan mencetak versi TF yang terinstal di komputer Anda Laboratorium kolaboratif:

```
impor tensorflow sebagai tf
cetak(tf.__version__)
```

Ini akan menghasilkan keluaran berikut:

```
2.1.0
```

3. Versi TF ini adalah versi TF terkini pada saat buku ini ditulis. Namun, kita semua tahu bahwa versi TF sering berubah dan kemungkinan akan ada versi TF baru saat Anda membaca buku ini. Jika demikian, Anda dapat menginstal versi TF tertentu sebagai berikut:

```
!pip pasang tensorflow==2.1.0
```

Kami berasumsi bahwa Anda familier dengan Python, oleh karena itu, kami akan mempercayakan tanggung jawab Anda untuk mencocokkan pustaka yang tepat dengan versi yang kami gunakan dalam buku ini. Ini tidak sulit dan dapat dilakukan dengan mudah seperti yang ditunjukkan sebelumnya, misalnya, menggunakan tanda == untuk menentukan versi. Kami akan menunjukkan versi yang digunakan saat kami melanjutkan.

TensorFlow dengan dukungan GPU Colaboratory,

secara default, memiliki dukungan GPU yang diaktifkan secara otomatis untuk TensorFlow. Namun, jika Anda memiliki akses ke sistem Anda sendiri dengan GPU dan ingin menyiapkan TensorFlow dengan dukungan GPU, penginstalannya sangat mudah. Cukup ketik perintah berikut di sistem pribadi Anda:

```
$ pip menginstal tensorflow-gpu
```

Namun, perlu diperhatikan bahwa ini mengasumsikan bahwa Anda telah menyiapkan semua driver yang diperlukan agar sistem Anda dapat memberikan akses ke GPU. Namun, jangan khawatir, ada banyak dokumentasi tentang proses ini yang dapat dicari di internet, misalnya, https://www.tensorflow.org/install/gpu . Jika Anda menemui masalah dan perlu melanjutkan, saya sangat menyarankan Anda kembali dan mengerjakan tugas di Colaboratory, karena ini adalah cara termudah untuk belajar.

Sekarang mari kita bahas cara kerja TensorFlow dan bagaimana paradigma grafiknya membuatnya sangat tangguh.

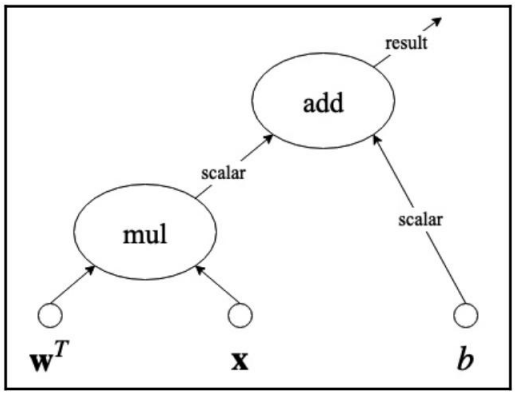
Prinsip di balik TensorFlow

Buku ini ditujukan untuk pemula dalam pembelajaran mendalam. Oleh karena itu, berikut ini adalah hal-hal yang ingin kami sampaikan kepada Anda tentang cara kerja TF. TF membuat grafik yang berisi eksekusi dari tensor inputnya, hingga tingkat abstraksi operasi tertinggi.

Misalnya, anggaplah kita memiliki tensor ***x*** dan ***w*** yang merupakan vektor input yang diketahui, dan kita memiliki konstanta b yang diketahui, dan katakanlah Anda ingin melakukan operasi ini:

$$\mathbf{w}^T\mathbf{x} + b$$

Jika kita membuat operasi ini dengan mendeklarasikan dan menetapkan tensor, grafiknya akan terlihat seperti pada *Gambar 2.1:*



Gambar 2.1 - Contoh operasi perkalian dan penjumlahan tensor

Dalam gambar ini, terdapat operasi perkalian tensor, *mul,* yang hasilnya adalah skalar dan perlu dijumlahkan, *add,* dengan skalar lain, b. Perhatikan bahwa ini mungkin merupakan hasil antara dan, dalam grafik komputasi nyata, hasil ini naik lebih tinggi di pohon eksekusi. Untuk informasi lebih rinci tentang bagaimana TF menggunakan grafik, silakan lihat makalah ini (Abadi, M., et.al., 2016).

Singkatnya, TF menemukan cara terbaik untuk menjalankan operasi tensor dengan mendelegasikan bagian-bagian tertentu ke GPU jika tersedia, atau dengan cara lain memparalelkan operasi pada inti CPU jika tersedia. TF bersifat sumber terbuka dengan komunitas pengguna yang terus berkembang di seluruh dunia. Sebagian besar profesional pembelajaran mendalam mengetahui tentang TF.

Sekarang mari kita bahas cara menyiapkan Keras dan cara mengabstraksikan fungsionalitas TensorFlow.

Pengenalan dan pengaturan Keras

Jika Anda mencari contoh kode TensorFlow di internet, Anda akan menemukan bahwa kode tersebut mungkin tidak terlalu mudah dipahami atau diikuti. Anda dapat menemukan tutorial untuk pemula, tetapi pada kenyataannya, semuanya dapat menjadi rumit dengan sangat mudah dan mengedit kode orang lain bisa jadi sangat sulit.
Keras hadir sebagai solusi API untuk mengembangkan prototipe model Tensorflow deep learning dengan relatif mudah. Faktanya, Keras mendukung berjalan tidak hanya di atas TensorFlow, tetapi juga di atas CNTK dan Theano.

Kita dapat menganggap Keras sebagai abstraksi dari model dan metode TensorFlow yang sebenarnya. Hubungan simbiosis ini telah menjadi sangat populer sehingga TensorFlow sekarang secara tidak resmi mendorong penggunaannya bagi mereka yang baru mulai menggunakan TensorFlow. Keras sangat ramah pengguna, mudah diikuti dengan Python, dan mudah dipelajari secara umum.

Pengaturan

Untuk menyiapkan Keras di Colab Anda, lakukan hal berikut:

1. Jalankan perintah berikut:

```
!pip install keras
```

2. Sistem akan melanjutkan untuk menginstal pustaka dan dependensi yang diperlukan. Setelah itu selesai, ketik dan jalankan potongan kode berikut:

```
impor keras
cetak(keras.__version__)
```

Ini menampilkan pesan konfirmasi penggunaan TensorFlow sebagai backend serta versi terbaru Keras, yang pada saat buku ini ditulis adalah 2.2.4.
Jadi, outputnya terlihat seperti ini:

```
Menggunakan backend TensorFlow. 2.2.4
```

Prinsip di balik Keras

Ada dua cara utama Keras menyediakan fungsionalitas kepada penggunanya: model sekuensial dan API Fungsional.

Hal ini dapat diringkas sebagai berikut:

- **Model sekuensial:** Ini merujuk pada cara penggunaan Keras yang memungkinkan Anda menumpuk instans lapisan secara linier (atau berurutan). Instans lapisan, dalam hal ini, memiliki makna yang sama seperti dalam pembahasan kita sebelumnya di Bab 1, *Pendahuluan tentang Pembelajaran Mesin.* Artinya, lapisan memiliki beberapa jenis masukan, beberapa jenis perilaku atau operasi model utama, dan beberapa jenis keluaran.
- **API Fungsional:** Ini adalah cara terbaik untuk lebih mendalami pendefinisian model yang lebih kompleks, seperti model gabungan, model dengan beberapa keluaran, model dengan beberapa lapisan bersama, dan banyak kemungkinan lainnya. Jangan khawatir, ini adalah topik lanjutan yang akan dijelaskan di bab selanjutnya. Paradigma API Fungsional memberi koder lebih banyak kebebasan untuk melakukan berbagai hal inovatif.

Kita dapat menganggap model sekuensial sebagai cara mudah untuk memulai dengan Keras, dan API Fungsional sebagai cara untuk mengatasi masalah yang lebih kompleks.

Ingat jaringan saraf dangkal dari Bab 1, *Pengantar Pembelajaran Mesin?*
Nah, beginilah cara Anda membuat model tersebut menggunakan paradigma model sekuensial di Keras:

```
dari keras.models impor Sequential dari keras.layers impor
Dense, Aktivasi

model = Berurut([
        Padat(10, bentuk_input=(10,)),
        Aktivasi('relu'),
        Padat(8),
        Aktivasi('relu'),
        Padat(4),
        Aktivasi('softmax'),
])
```

Dua baris kode pertama mengimpor model Sequential dan lapisan Dense dan Activation , masing-masing. Lapisan Dense adalah jaringan neural yang terhubung sepenuhnya, sedangkan lapisan Activation adalah cara yang sangat spesifik untuk memanggil serangkaian fungsi aktivasi yang lengkap, seperti ReLU dan SoftMax, seperti pada contoh sebelumnya (ini akan dijelaskan secara rinci nanti).

Atau, Anda dapat melakukan model yang sama, tetapi menggunakan metode add() :

```
dari keras.models impor Sequential dari keras.layers impor
Dense, Aktivasi

model = Sequential()
model.tambahkan(Dense(10, input_dim=10))
model.tambahkan(Aktivasi('relu'))
model.tambahkan(Dense(8))
model.tambahkan(Aktivasi('relu'))
model.tambahkan(Padat(4))
model.tambahkan(Aktivasi('softmax'))
```

Cara kedua penulisan kode untuk model neural ini terlihat lebih linier, sedangkan cara pertama lebih mirip cara Pythonic untuk melakukannya dengan daftar item. Sebenarnya keduanya sama saja dan Anda mungkin akan lebih memilih salah satu cara. Namun, perlu diingat, kedua contoh sebelumnya menggunakan model sekuensial Keras.

Nah, hanya untuk tujuan perbandingan, beginilah cara Anda membuat kode arsitektur jaringan saraf yang sama persis, tetapi menggunakan paradigma Keras Functional API:

```
dari keras.layers impor Input, Padat dari keras.models impor
Model

input = Input(bentuk=(10,))

x = Padat(10, aktivasi='relu')(input) x = Padat(8, aktivasi='relu')(x)
y = Padat(4, aktivasi='softmax')(x)


model = Model(input=input, output=y)
```

Jika Anda seorang programmer berpengalaman, Anda akan melihat bahwa gaya API Fungsional memungkinkan fleksibilitas yang lebih tinggi. Gaya ini memungkinkan Anda untuk menentukan tensor input guna menggunakannya sebagai input ke berbagai bagian model, jika diperlukan. Namun, penggunaan API Fungsional mengasumsikan bahwa Anda sudah familier dengan model sekuensial. Oleh karena itu, dalam buku ini, kita akan mulai dengan model sekuensial dan beralih ke paradigma API Fungsional seiring kemajuan kita menuju model neural yang lebih kompleks.

Sama seperti Keras, ada pustaka dan kerangka kerja Python lain yang memungkinkan kita melakukan pembelajaran mesin dengan tingkat kesulitan yang relatif rendah. Pada saat buku ini ditulis, yang paling populer adalah Keras dan yang kedua paling populer adalah PyTorch.

Pengantar PyTorch

Pada saat buku ini ditulis, PyTorch merupakan kerangka kerja pembelajaran mendalam ketiga yang paling populer secara keseluruhan. Popularitasnya terus meningkat meskipun tergolong baru di dunia dibandingkan dengan TensorFlow. Salah satu hal menarik tentang PyTorch adalah ia memungkinkan beberapa penyesuaian yang tidak dapat dilakukan oleh TensorFlow. Lebih jauh lagi, PyTorch didukung oleh Facebook™.

Meskipun buku ini membahas TensorFlow dan Keras, saya rasa penting bagi kita semua untuk mengingat bahwa PyTorch adalah alternatif yang bagus dan tampilannya sangat mirip dengan Keras. Sebagai referensi, berikut adalah tampilan jaringan saraf dangkal yang sama persis yang kami tunjukkan sebelumnya jika dikodekan dalam PyTorch:

```
impor obor

perangkat = torch.device('cpu')

model = obor.nn.Berurutan(
```

```
torch.nn.Linear(10, 10), torch.nn.ReLU(),
torch.nn.Linear(10, 8),
torch.nn.ReLU(), torch.nn.Linear(8, 2),

torch.nn.Softmax(2)).ke(perangkat)
```

Ada banyak kesamaan. Selain itu, transisi dari Keras ke PyTorch seharusnya tidak terlalu sulit bagi pembaca yang termotivasi, dan ini bisa menjadi keterampilan yang bagus untuk dimiliki di masa mendatang.
Namun, untuk saat ini, sebagian besar minat komunitas tertuju pada TensorFlow dan semua turunannya, terutama Keras. Jika Anda ingin mengetahui lebih lanjut tentang asal usul dan prinsip dasar PyTorch, Anda mungkin menganggap bacaan ini bermanfaat (Paszke, A., et.al., 2017).

Pengantar Dopamin

Perkembangan terbaru yang menarik dalam dunia deep reinforcement learning adalah Dopamine. Dopamine adalah kerangka kerja untuk pembuatan prototipe cepat algoritma deep reinforcement learning. Buku ini akan membahas secara singkat tentang reinforcement learning, tetapi Anda perlu mengetahui cara menginstalnya.

Dopamin dikenal mudah digunakan bagi pengguna baru di dunia pembelajaran penguatan. Selain itu, meskipun bukan produk resmi Google, sebagian besar pengembangnya adalah karyawan Google. Dalam kondisi saat ini, saat buku ini ditulis, kerangka kerja ini sangat ringkas dan menyediakan algoritme yang siap pakai.

Untuk menginstal Dopamin, Anda dapat menjalankan perintah berikut:

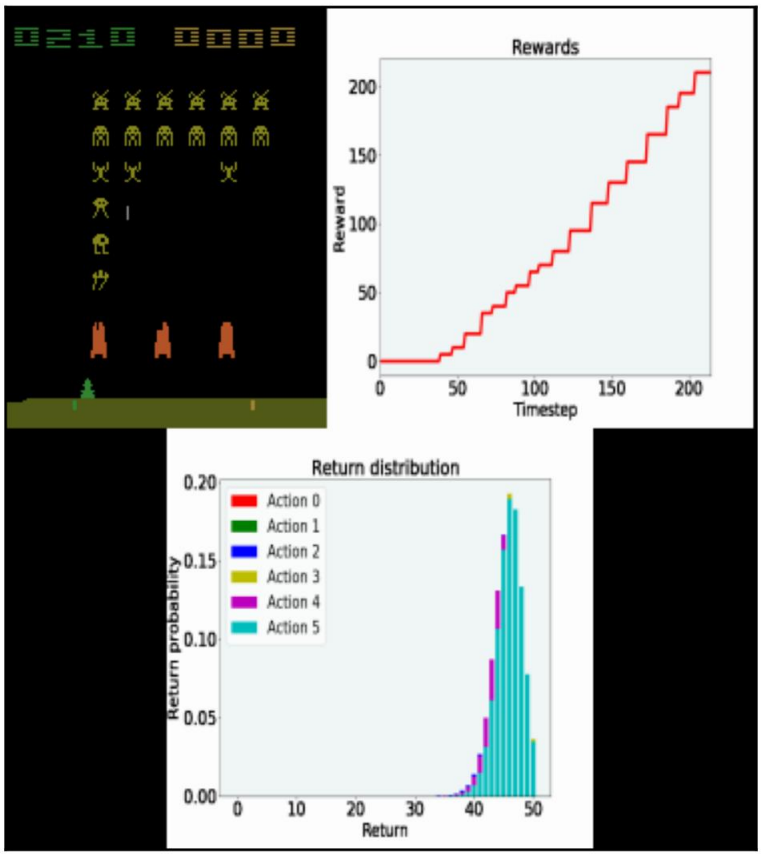
```
!pip instal dopamin-rl
```

Anda dapat menguji instalasi Dopamine yang benar hanya dengan menjalankan perintah berikut:

```
impor dopamin
```

Ini tidak memberikan output, kecuali jika terjadi kesalahan. Biasanya, Dopamine akan menggunakan banyak pustaka di luarnya untuk memungkinkan melakukan banyak hal yang lebih menarik. Saat ini, beberapa hal paling menarik yang dapat dilakukan dengan pembelajaran penguatan adalah melatih agen dengan kebijakan hadiah, yang memiliki aplikasi langsung dalam permainan.

Sebagai contoh, lihat *Gambar 2.2,* yang menampilkan cuplikan waktu permainan video saat permainan tersebut belajar, menggunakan kebijakan yang memperkuat perilaku yang diinginkan tergantung pada tindakan yang diambil oleh agen:



Gambar 2.2 - Contoh visualisasi agen Dopamin dalam masalah pembelajaran penguatan dalam permainan

Agen dalam pembelajaran penguatan adalah bagian yang memutuskan tindakan apa yang akan diambil selanjutnya. Agen melakukannya dengan mengamati dunia dan aturan-aturan dunia. Semakin jelas aturannya, semakin terbatas pula hasilnya. Jika aturannya terlalu longgar, agen mungkin tidak dapat membuat keputusan yang baik tentang tindakan apa yang akan diambil.

Meskipun buku ini tidak membahas secara mendalam tentang pembelajaran penguatan, kami akan membahas aplikasi permainan yang menarik di bab terakhir buku ini. Untuk saat ini, Anda dapat membaca white paper berikut untuk informasi lebih lanjut tentang Dopamin (Castro, PS, et.al., 2018).

Pustaka pembelajaran mendalam lainnya

Selain dua pesaing besar, TensorFlow dan Keras, ada pesaing lain yang merambah dunia pembelajaran mendalam. Kami telah membahas PyTorch, tetapi masih banyak lagi.
Di sini kita akan membahasnya secara singkat.

Kafe

Caffe juga merupakan framework populer yang dikembangkan di UC Berkeley (Jia, Y., et.al. 2014). Framework ini menjadi sangat populer pada tahun 2015-2016. Beberapa pemberi kerja masih membutuhkan keahlian ini dan artikel ilmiah masih menyebutkan penggunaannya. Akan tetapi, penggunaannya menurun sebagian karena keberhasilan besar TF dan aksesibilitas Keras.

Untuk informasi lebih lanjut tentang Caffe, kunjungi: https://caffe.berkeleyvision.
organisasi.

Perhatikan juga keberadaan Caffe2, yang dikembangkan oleh Facebook dan bersifat open source. Aplikasi ini dibangun berdasarkan Caffe, tetapi sekarang Facebook memiliki jagoan barunya, PyTorch.

Theano

Theano dikembangkan oleh kelompok Yoshua Bengio di Universitas Montreal pada tahun 2007 (Al-Rfou, R., *et.al.* 2016). Theano memiliki basis pengguna yang relatif lama yang mungkin menjadi saksi kebangkitan TF.
Rilisan utama terakhir dibuat pada akhir tahun 2017 dan, meskipun belum ada rencana yang jelas mengenai rilisan utama baru, pembaruan masih dilakukan oleh komunitas.

Untuk informasi lebih lanjut tentang Theano, silakan kunjungi:
http://deeplearning.net/software/theano/

Penghargaan terhormat

Ada alternatif lain di luar sana yang mungkin tidak sepopuler itu, karena berbagai alasan, tetapi perlu disebutkan di sini jika masa depan mereka berubah. Berikut ini adalah beberapa di antaranya:

Nama	**Dikembangkan oleh**	**Informasi lebih lanjut**
Bahasa Indonesia: MXNET	Bahasa Apache	https://mxnet.apache.org/
Bahasa Indonesia: CNTK	Microsoft	https://cntk.ai
Pembelajaran Mendalam4J	Skymind	https://deeplearning4j.org/
Rantai	Jaringan Pilihan	https://chainer.org/
AI cepat	Jeremy Howard	https://www.fast.ai/

Ringkasan

Bab pengantar ini menunjukkan cara menyiapkan pustaka yang diperlukan untuk menjalankan TensorFlow, Keras, dan Dopamin. Semoga Anda akan menggunakan Colabs untuk mempermudah Anda belajar. Anda juga mempelajari pola pikir dasar dan konsep desain di balik kerangka kerja ini. Meskipun kerangka kerja seperti itu adalah yang paling populer pada saat buku ini ditulis, ada pesaing lain di luar sana, yang juga kami perkenalkan secara singkat.

Pada titik ini, Anda siap untuk memulai perjalanan untuk menguasai pembelajaran mendalam. tonggak sejarah adalah mengetahui cara menyiapkan data untuk aplikasi pembelajaran mendalam. Item ini adalah penting bagi keberhasilan model. Tidak peduli seberapa bagus modelnya dan seberapa dalam mereka adalah, jika data tidak diformat atau diolah dengan benar, hal ini dapat menyebabkan bencana hasil kinerja. Oleh karena itu, sekarang kita akan masuk ke Bab 3, *Mempersiapkan Data.* Dalam hal itu bab ini, Anda akan mempelajari cara mengambil kumpulan data dan mempersiapkannya untuk tugas spesifik yang Anda lakukan mencoba memecahkan masalah dengan jenis model pembelajaran mendalam tertentu. Namun, sebelum Anda melakukannya, Silakan coba uji diri Anda dengan pertanyaan-pertanyaan berikut.

Pertanyaan dan jawaban

1. **Apakah Colab berjalan di komputer pribadi saya?**

   Tidak, ini berjalan di cloud, tetapi dengan beberapa keterampilan dan pengaturan, Anda dapat menghubungkannya ke awan pribadi Anda.

2. **Apakah Keras menggunakan GPU?**

   Ya. Karena Keras berjalan pada TensorFlow (dalam pengaturan buku ini) dan TensorFlow menggunakan GPU, maka Keras juga demikian.

3. **Apa dua paradigma pengkodean utama di Keras?**

   (A) Model sekuensial; (B) API fungsional.

4. **Mengapa kita peduli dengan Dopamin?**

   Karena hanya ada beberapa kerangka kerja pembelajaran penguatan yang dapat Anda percaya di luar sana, dan Dopamin adalah salah satunya.

Referensi

- Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., dan Kudlur, M. (2016). *Tensorflow: Sistem untuk pembelajaran mesin berskala besar.* Dalam *Simposium {USENIX} ke-12 tentang Desain dan Implementasi Sistem Operasi* ({OSDI} 16) (hlm. 265-283).
- Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L. dan Lerer, A. (2017). *Diferensiasi otomatis di pytorch.*
- Castro, PS, Moitra, S., Gelada, C., Kumar, S., dan Bellemare, MG (2018). *Dopamin: Kerangka kerja penelitian untuk pembelajaran penguatan mendalam.* Pracetak arXiv arXiv:1812.06110.
- Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., dan Darrell, T. (2014, November). *Caffe: Arsitektur konvolusional untuk penyisipan fitur yang cepat.* Dalam *Prosiding konferensi internasional ACM ke-22 tentang Multimedia* (hlm. 675-678). *ACM.*
- Al-Rfou, R., Alain, G., Almahairi, A., Angermueller, C., Bahdanau, D., Ballas, N., Bastien, F., Bayer, J., Belikov, A., Belopolsky, A. dan Bengio, Y. (2016). *Theano: Kerangka kerja Python untuk perhitungan cepat ekspresi matematika.* arXiv pracetak arXiv:1605.02688.