MACHINE LEARNING : Deep Learning with PyTorch

Alvandi Damansyah

1103192191

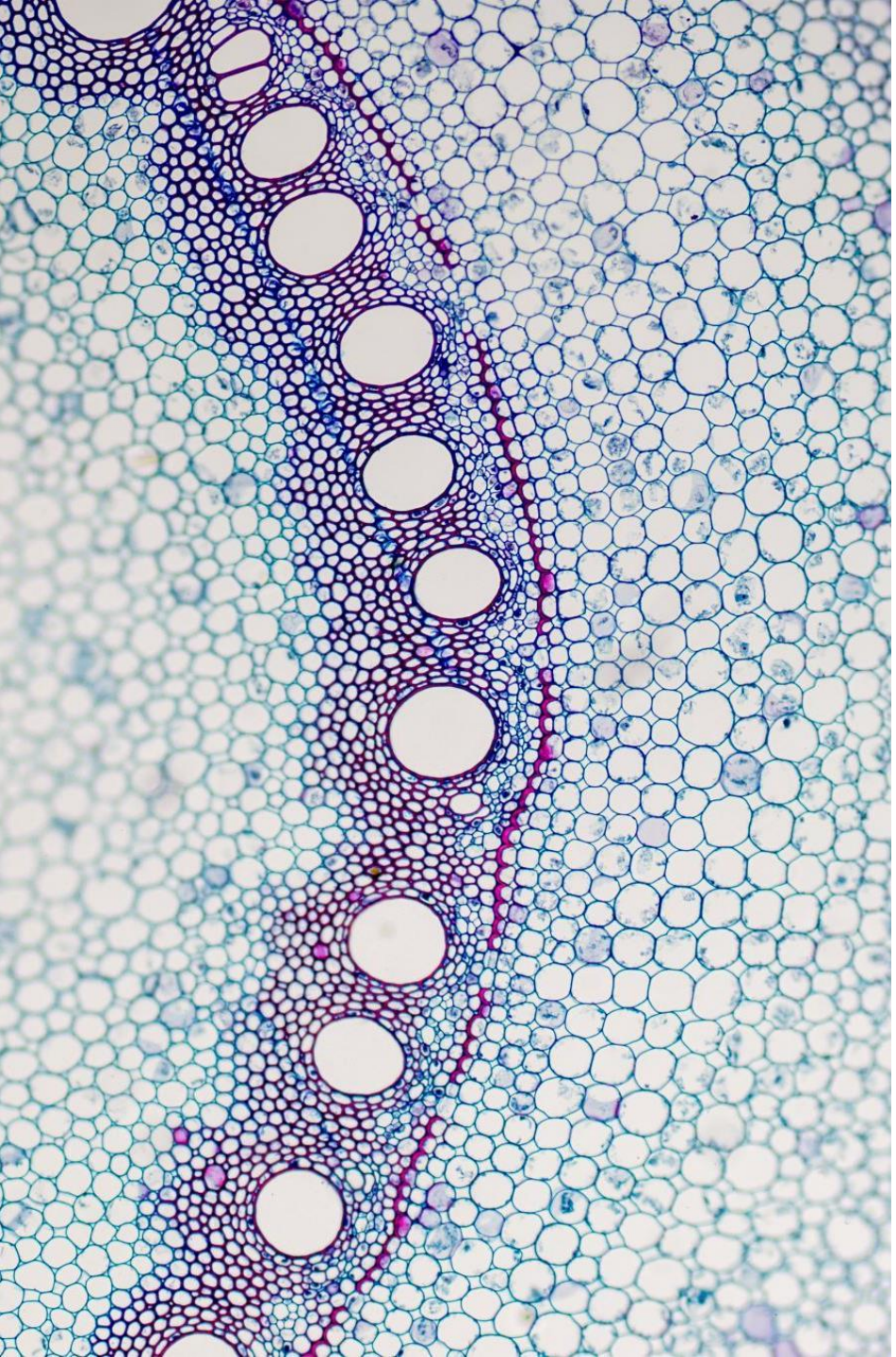
Table Of Content

Table Of Content

Introduction

pytorch

- PyTorch adalah sebuah framework open-source yang digunakan untuk pembelajaran mesin dan pemrosesan data skala besar.
- Dengan fokus pada fleksibilitas dan efisiensi komputasi, PyTorch menyediakan alat dan fungsi yang memudahkan pembuatan, pelatihan, dan evaluasi model jaringan saraf tiruan. Dalam PyTorch, data direpresentasikan dalam bentuk tensor, yang memungkinkan manipulasi dan pengolahan data numerik secara efisien.
- Salah satu keunggulan PyTorch adalah kemampuannya untuk membangun model secara dinamis, yang memfasilitasi eksplorasi dan eksperimen yang lebih mudah dalam pengembangan model.
- PyTorch juga memiliki dukungan yang baik untuk komputasi GPU, yang memungkinkan percepatan pemrosesan data dan pelatihan model. Dengan pustaka dan modul yang berguna serta dukungan aktif dari komunitas, PyTorch menjadi salah satu pilihan utama bagi peneliti dan praktisi dalam bidang kecerdasan buatan.

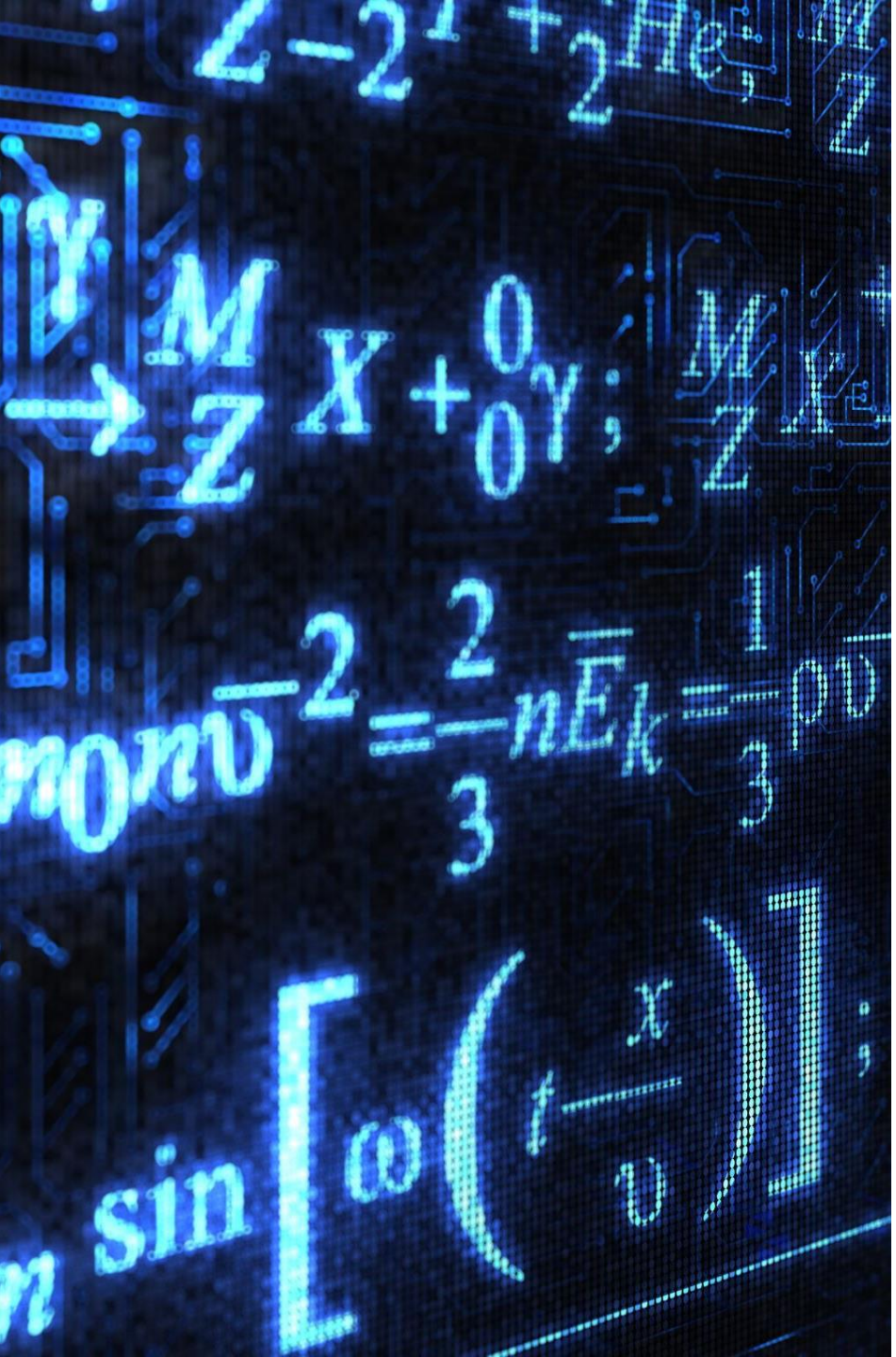


Tensorflow

TensorFlow adalah sebuah framework open-source yang digunakan untuk pembelajaran mesin dan kecerdasan buatan. Dikembangkan oleh Google, TensorFlow menyediakan alat dan infrastruktur yang kuat untuk membangun, melatih, dan menerapkan model pembelajaran mesin yang kompleks.

Pusat dari TensorFlow adalah representasi data menggunakan tensor, yang merupakan struktur data multidimensi yang mirip dengan array. Framework ini memungkinkan pengguna untuk membuat model jaringan saraf tiruan yang dalam (deep neural networks) dengan mudah, serta menyediakan fitur-fitur seperti komputasi GPU, distribusi, dan skala besar.

TensorFlow juga menyediakan antarmuka yang mudah digunakan untuk membangun model, serta dukungan untuk berbagai tugas pembelajaran mesin seperti pengenalan wajah, terjemahan mesin, dan pengenalan suara. Selain itu, TensorFlow memiliki ekosistem yang kaya dengan pustaka dan alat tambahan yang dikembangkan oleh komunitas yang aktif, membuatnya menjadi salah satu framework terkemuka dalam pembelajaran mesin dan kecerdasan buatan.



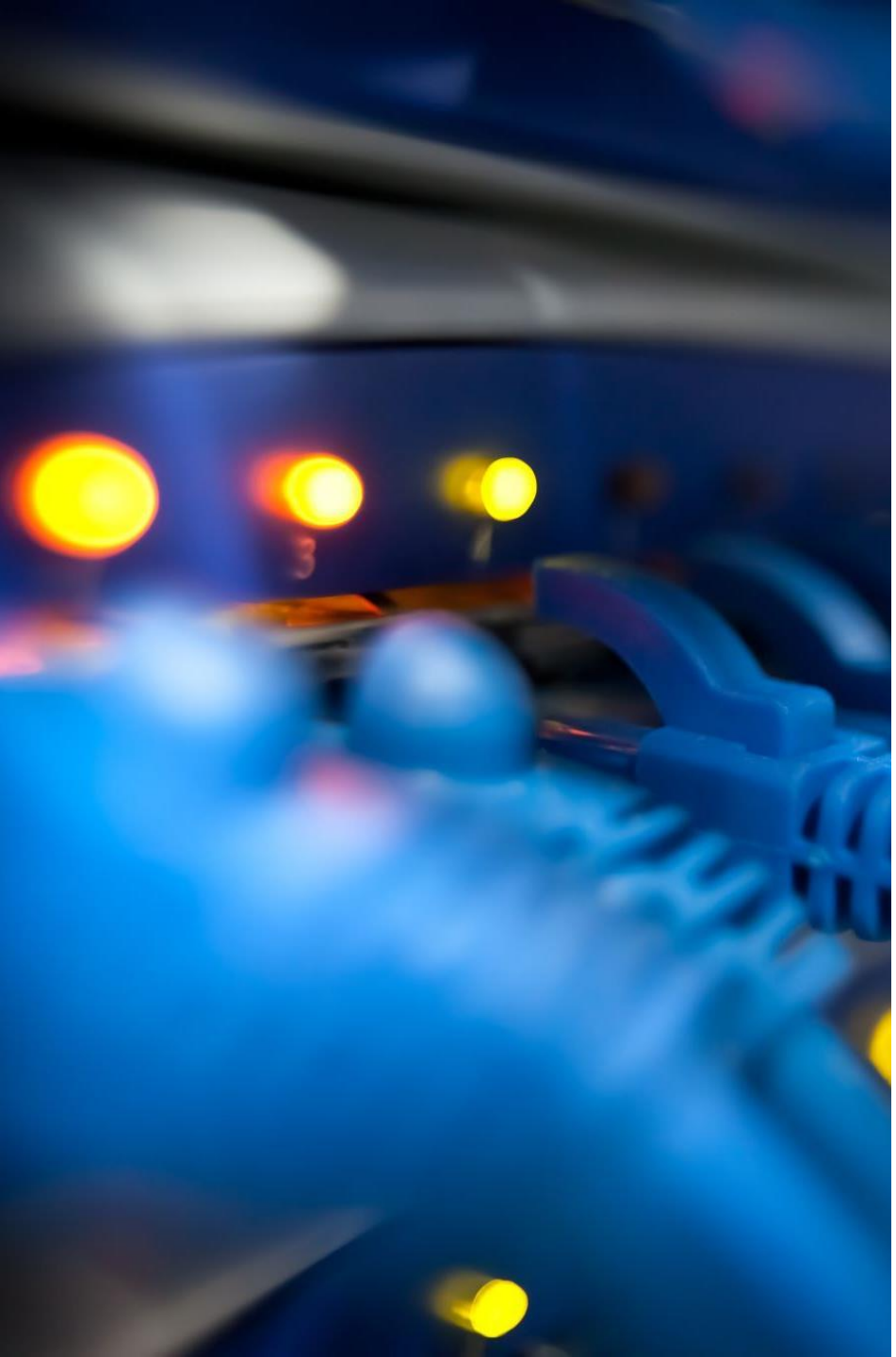
Deep Learning

Deep learning adalah cabang dari pembelajaran mesin yang berfokus pada pengembangan dan penerapan model jaringan saraf tiruan yang dalam (deep neural networks).

Deep learning bertujuan untuk mengeksplorasi, memodelkan, dan mempelajari representasi yang lebih abstrak dari data, dengan menggunakan jaringan saraf tiruan yang terdiri dari banyak lapisan (layer) yang saling terhubung.

Dalam deep learning, model jaringan saraf tiruan ini belajar secara otomatis melalui algoritma pembelajaran yang mendalam (deep learning algorithms) untuk mengidentifikasi pola dan memahami struktur data yang kompleks.

Keunggulan deep learning terletak pada kemampuannya untuk memproses data yang berukuran besar dan kompleks, seperti gambar, suara, teks, atau data waktu berkelanjutan, serta kemampuannya untuk melakukan tugas-tugas seperti pengenalan objek, deteksi wajah, terjemahan mesin, dan pengenalan ucapan, di antara banyak aplikasi lainnya.



Commonly Used Deep Learning Models

Berikut adalah beberapa contoh model deep learning yang sering digunakan:

- Convolutional Neural Networks (CNN): CNN adalah jenis model deep learning yang paling umum digunakan dalam bidang pengolahan citra dan pengenalan pola. CNN dirancang khusus untuk memproses data berupa gambar dan mengenali pola-pola visual dalam gambar. Model ini terdiri dari beberapa lapisan konvolusi dan lapisan pooling yang berguna untuk mengekstraksi fitur-fitur penting dari gambar.
- Recurrent Neural Networks (RNN): RNN adalah model deep learning yang dirancang untuk mengolah data berkelanjutan, seperti teks atau urutan data waktu. RNN memiliki keunggulan dalam memperhitungkan konteks sebelumnya melalui penggunaan unit memori yang disebut sel memori (memory cell), yang memungkinkan mereka untuk menjaga dan memanfaatkan informasi kontekstual dalam data urutan.
- Long Short-Term Memory (LSTM): LSTM adalah jenis RNN yang lebih kompleks dan efektif dalam memproses urutan data yang panjang. LSTM menggunakan gate (pintu) khusus untuk mengontrol aliran informasi dalam sel memori, yang membantu dalam mempelajari dan mengingat pola jangka panjang dalam urutan data.

Work
With Pytoch Tensor

Tensor : Basic

Tensor adalah struktur data fundamental dalam pembelajaran mesin dan pengolahan data numerik. Secara sederhana, tensor dapat dianggap sebagai generalisasi dari konsep matriks dalam aljabar linear. Tensor dapat memiliki dimensi yang lebih tinggi daripada matriks, artinya tensor dapat memiliki beberapa dimensi, seperti satu dimensi (vektor), dua dimensi (matriks), atau lebih.

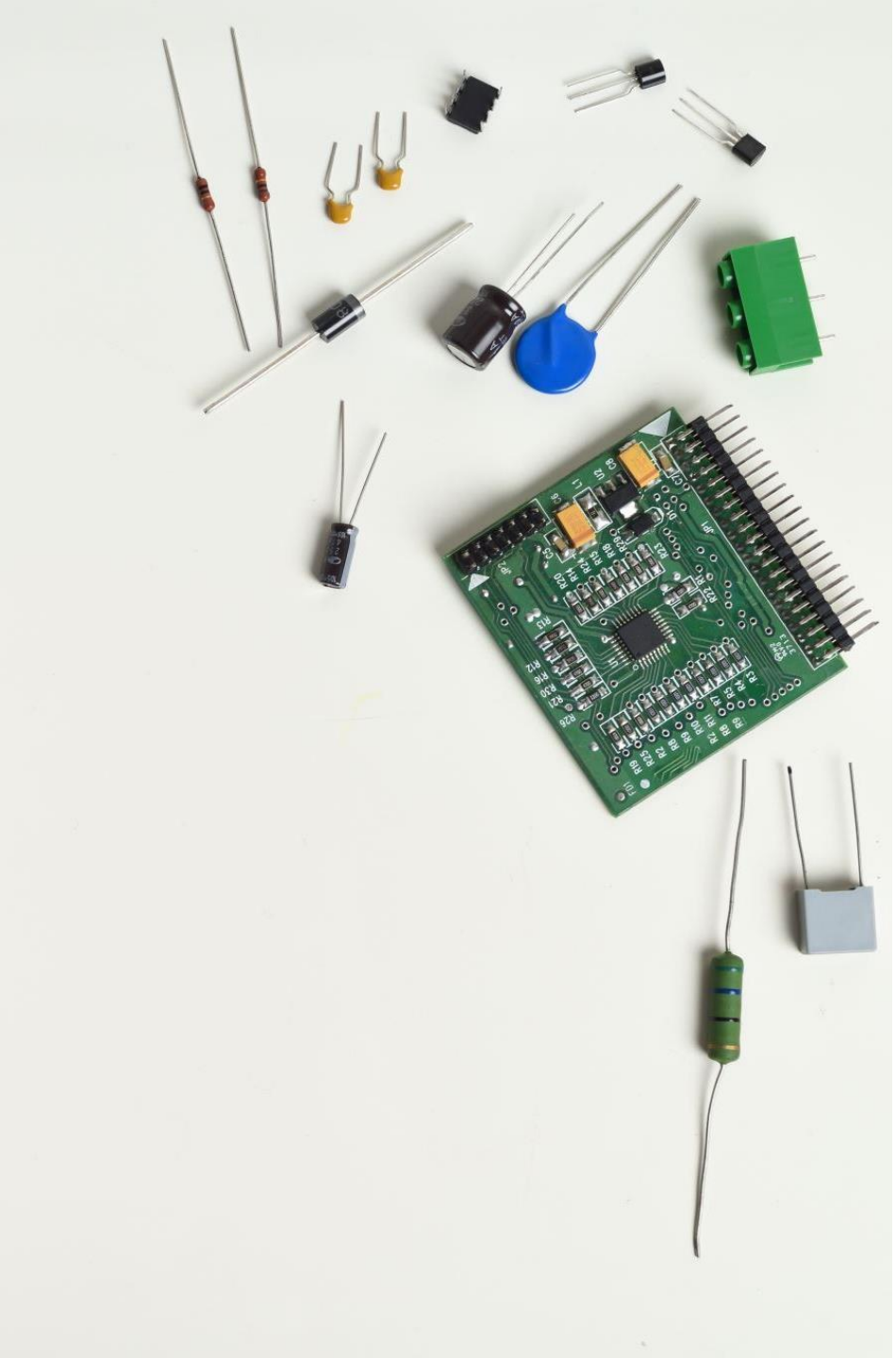
Secara lebih spesifik, tensor adalah array multidimensi yang terdiri dari elemen-elemen numerik. Setiap elemen di dalam tensor diberi indeks berdasarkan dimensi yang sesuai. Misalnya, tensor 2D memiliki dua indeks: baris dan kolom. Tensor 3D memiliki tiga indeks, dan seterusnya.

Autograd

PyTorch autograd adalah modul dalam framework PyTorch yang bertanggung jawab untuk mengotomatisasi perhitungan gradien (gradient) dalam jaringan saraf tiruan.

Dengan menggunakan autograd, PyTorch secara otomatis menghitung gradien dari parameter yang dibutuhkan dalam proses pelatihan model. Ketika melakukan operasi matematika pada tensor yang diberi tanda bahwa perlu dilakukan pelacakan gradien, autograd akan merekam operasi dan membangun grafik komputasi yang berisi hubungan antara tensor input, operasi, dan tensor output. Selama proses mundur (backpropagation), autograd menggunakan grafik ini untuk menghitung gradien secara otomatis menggunakan aturan rantai (chain rule) dalam kalkulus.

Hal ini mempermudah pelatihan model dengan gradient descent dan optimisasi parameter lainnya, karena kita tidak perlu secara manual mengimplementasikan perhitungan gradien yang rumit.



Backpropagation

PyTorch backpropagation (mundur) adalah metode yang digunakan dalam pembelajaran mesin untuk menghitung gradien dari parameter-model yang diperlukan dalam proses pelatihan.

Proses backpropagation melibatkan pemrosesan mundur melalui jaringan saraf tiruan, dimulai dari output ke input, dengan menghitung gradien setiap lapisan berdasarkan aturan rantai (chain rule) dalam kalkulus. Dalam PyTorch, dengan bantuan modul autograd, backpropagation secara otomatis menghitung gradien dari fungsi rugi (loss function) terhadap parameter-model, dan mengupdate parameter-model menggunakan algoritma optimisasi seperti stochastic gradient descent (SGD).

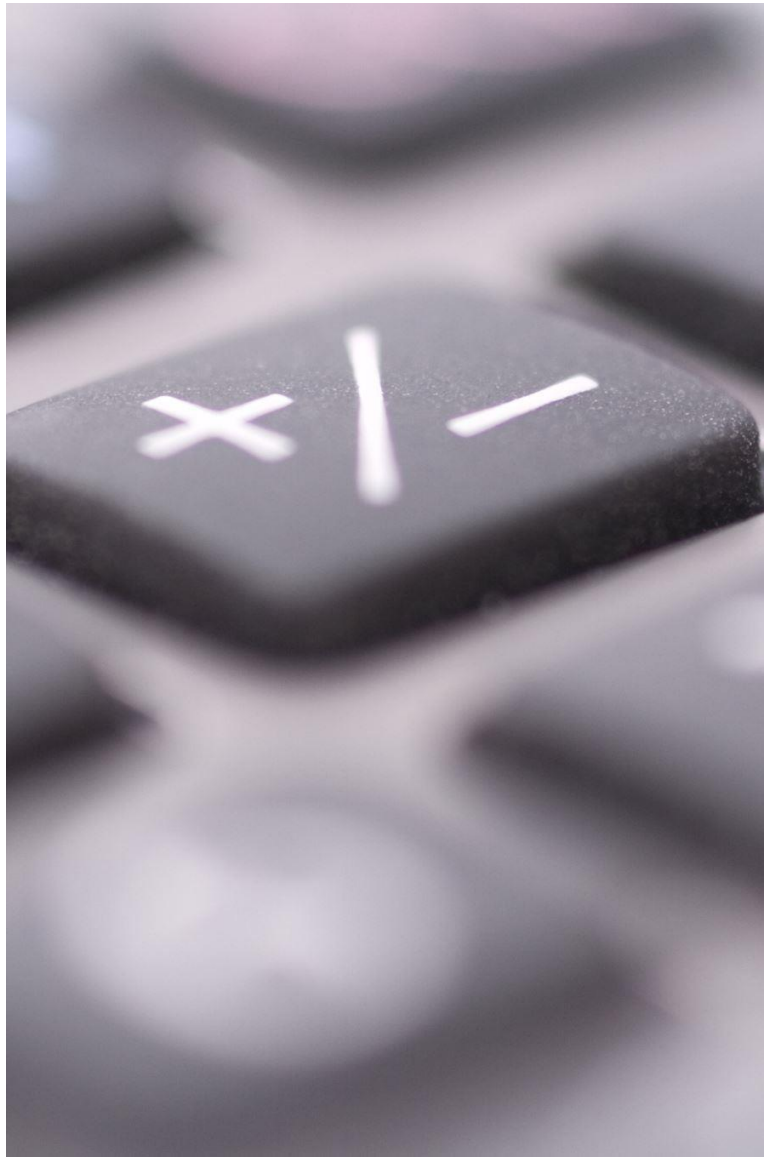
Dengan demikian, backpropagation memungkinkan model untuk mengoptimalkan parameter-modelnya secara iteratif, meningkatkan kinerja model selama pelatihan.

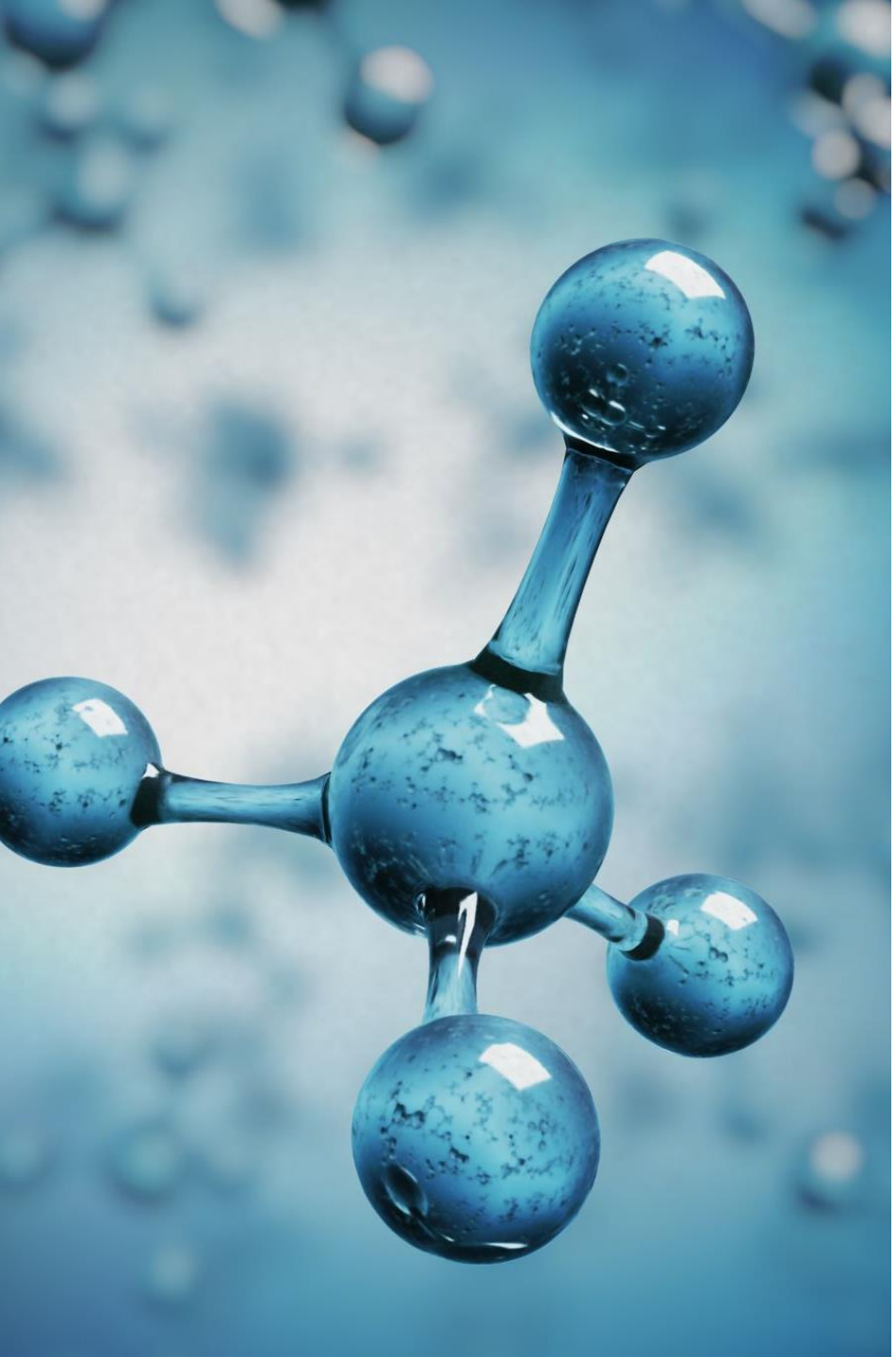
Gradient Descent

PyTorch gradient descent (penurunan gradien) adalah algoritma optimisasi yang digunakan dalam pembelajaran mesin untuk mencari nilai-nilai parameter-model yang menghasilkan hasil yang optimal.

Dalam PyTorch, gradient descent melibatkan menghitung gradien fungsi rugi (loss function) terhadap parameter-model menggunakan metode backpropagation. Setelah gradien dihitung, parameter-model diperbarui dengan mengurangi gradien dengan faktor kecepatan pembelajaran (learning rate). Proses ini diulang secara iteratif, dengan tujuan mengurangi nilai fungsi rugi dan memperbarui parameter-model untuk mendekati hasil yang lebih baik.

Gradient descent dalam PyTorch dapat dilakukan dalam berbagai variasi, seperti stochastic gradient descent (SGD), mini-batch gradient descent, atau metode optimisasi yang lebih canggih seperti Adam atau RMSprop. Gradient descent merupakan algoritma penting dalam pelatihan model, yang memungkinkan model untuk belajar dari data dan menyesuaikan parameter-modelnya secara efisien untuk menghasilkan prediksi yang lebih baik.





Pipeline

Dalam PyTorch, training pipeline (pipa pelatihan) untuk model-model terdiri dari beberapa komponen utama: model, fungsi rugi (loss function), dan optimizer. Pertama, model merupakan arsitektur jaringan saraf tiruan yang terdiri dari lapisan-lapisan yang saling terhubung. Selanjutnya, fungsi rugi digunakan untuk mengukur kesalahan prediksi model terhadap data pelatihan.

Tujuan utama dalam pelatihan adalah untuk mengoptimalkan model sehingga fungsi rugi dapat diminimalkan. Inilah peran optimizer, yang bertanggung jawab untuk memperbarui parameter-model menggunakan algoritma optimisasi, seperti stochastic gradient descent (SGD) atau Adam, berdasarkan gradien yang dihitung menggunakan metode backpropagation.

Regression

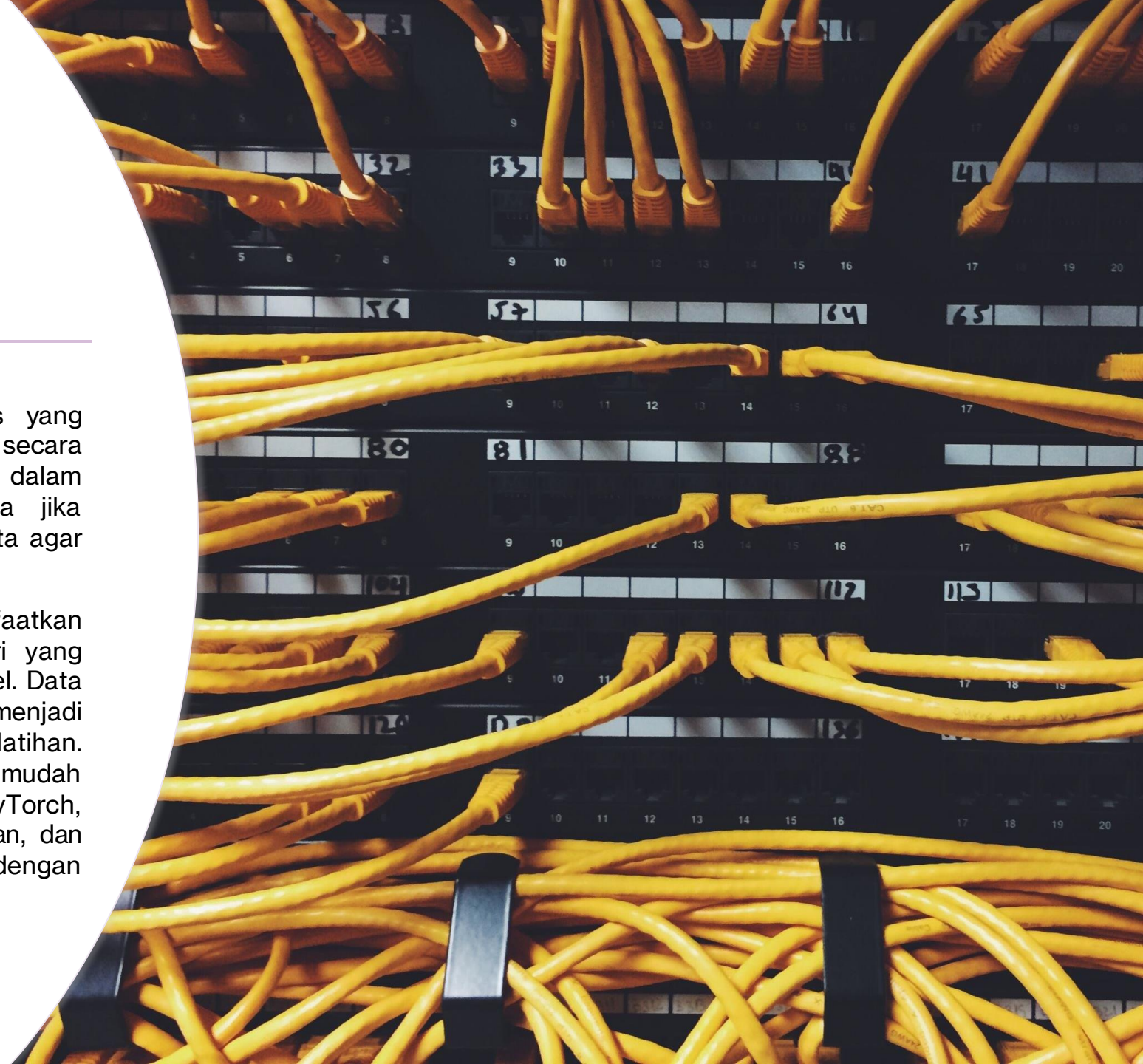
Dalam konteks PyTorch, regresi adalah tugas pembelajaran mesin yang bertujuan untuk memprediksi nilai kontinu berdasarkan input data. Dalam regresi, kita mencoba untuk membangun model yang dapat menemukan hubungan atau pola matematika antara fitur-fitur input dan target output yang berupa nilai numerik.

Dalam PyTorch, regresi dapat diimplementasikan dengan menggunakan model jaringan saraf tiruan, di mana input data akan mengalir melalui lapisan-lapisan jaringan untuk menghasilkan prediksi kontinu. Fungsi rugi seperti Mean Squared Error (MSE) atau Mean Absolute Error (MAE) digunakan untuk mengukur kesalahan antara prediksi dan target output.

Dataloader

Dalam PyTorch, data loader adalah sebuah utilitas yang membantu dalam memuat data pelatihan atau pengujian secara efisien. Data loader berfungsi untuk memuat dataset ke dalam model secara batch, melakukan transformasi data jika diperlukan, serta melakukan pengacakan (shuffling) data agar pelatihan lebih acak.

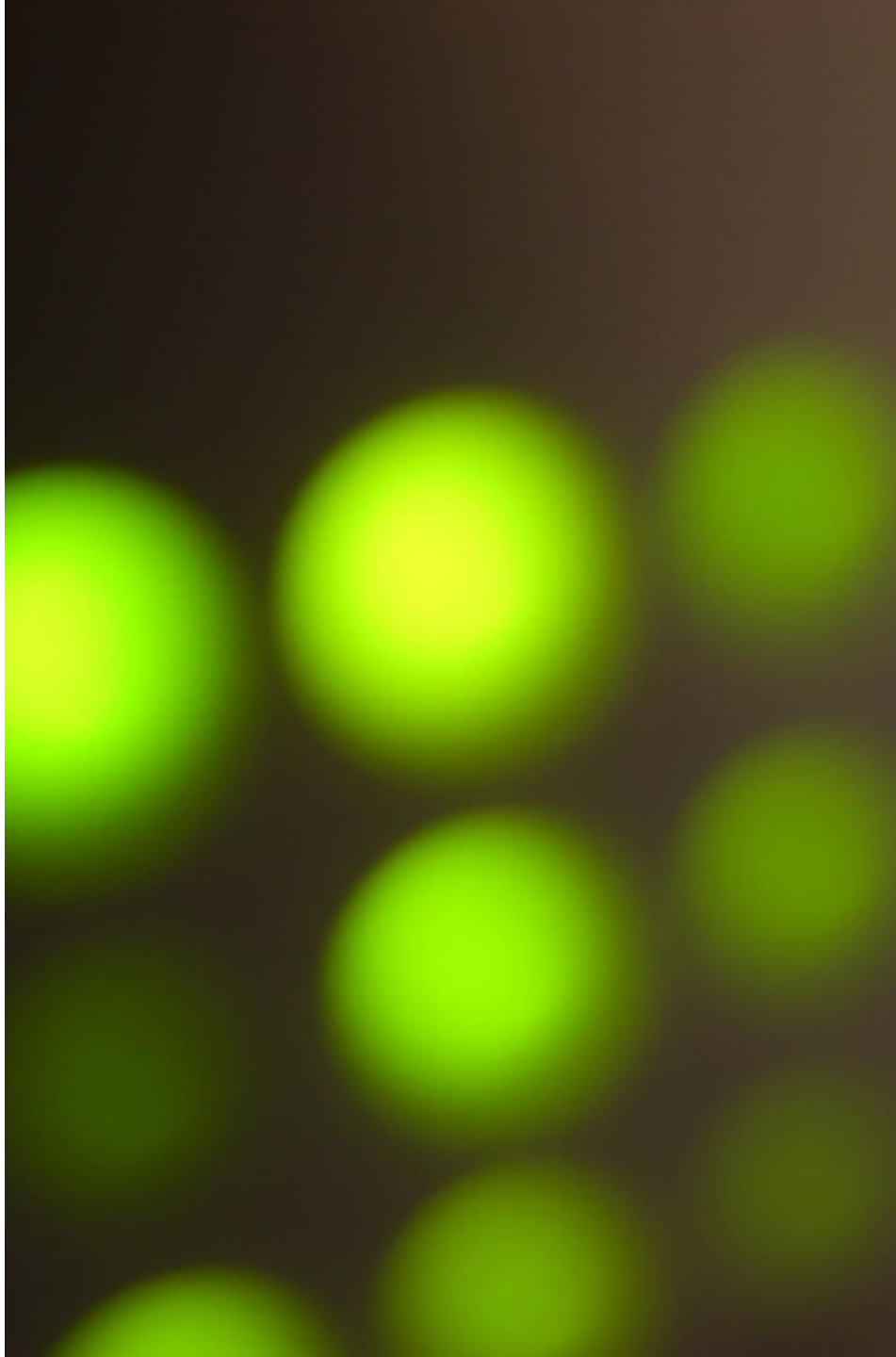
Dengan menggunakan data loader, kita dapat memanfaatkan fitur-fitur seperti paralelisasi dan penggunaan memori yang efisien dalam memuat dan mengolah data secara paralel. Data loader juga memudahkan dalam membagi dataset menjadi batch-batch kecil yang dapat digunakan dalam pelatihan. Dengan adanya data loader, kita dapat dengan mudah mengintegrasikan dataset kita dengan model PyTorch, mengatur pengaturan seperti ukuran batch, pengacakan, dan lainnya, serta mempercepat proses pelatihan dengan memanfaatkan potensi perangkat keras yang tersedia.



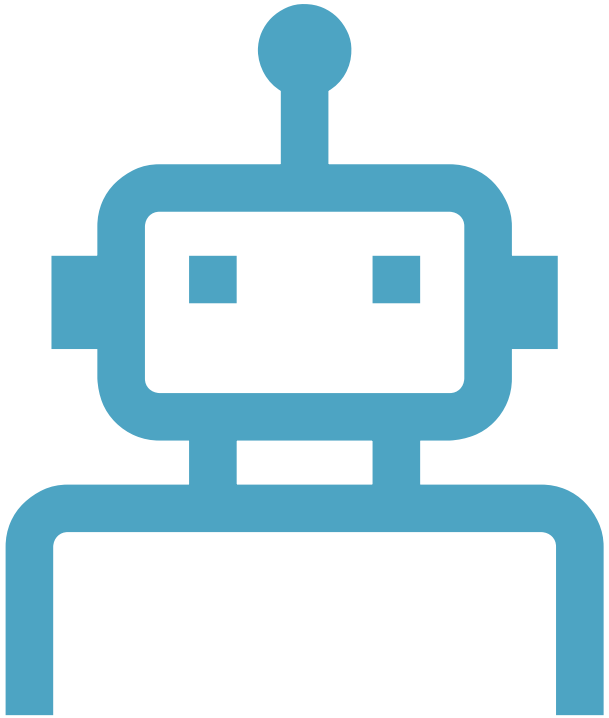
Transform

Dalam PyTorch, transforms adalah modul yang digunakan untuk melakukan transformasi pada data. Transformasi ini dapat diterapkan pada dataset sebelum dimuat ke dalam model, seperti melakukan prapemrosesan atau augmentasi data. Contoh transformasi umum termasuk normalisasi data, perubahan ukuran gambar, pemotongan gambar, flipping horizontal atau vertikal, rotasi, dan transformasi lainnya.

Dengan menggunakan transforms, kita dapat mengubah dan memodifikasi data dengan mudah sesuai kebutuhan eksperimen atau pelatihan model. Transformasi ini dapat diterapkan baik pada data pelatihan maupun data pengujian, dan memungkinkan untuk meningkatkan keberagaman dan jumlah sampel dalam dataset, serta meningkatkan kemampuan model untuk mengeneralisasi dan menghadapi variasi dalam data yang diberikan.



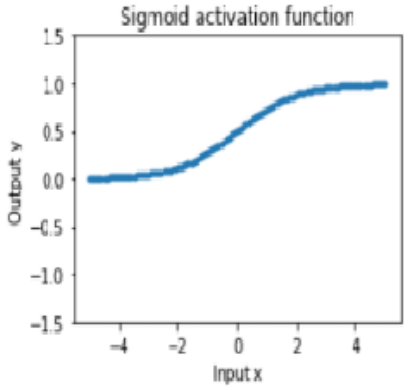
Softmax and Crossentropy

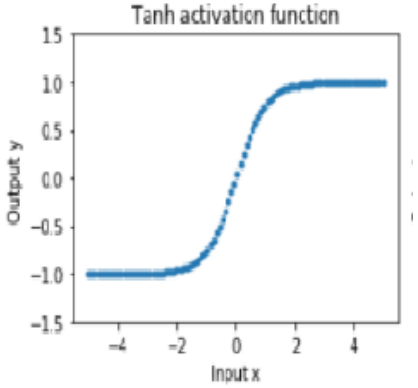


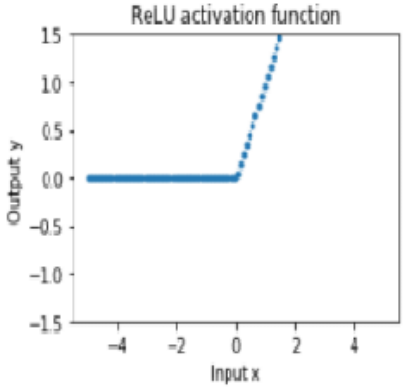
Dalam PyTorch, softmax dan cross-entropy adalah dua fungsi yang sering digunakan dalam pembelajaran mesin untuk tugas klasifikasi. Softmax adalah fungsi aktivasi yang mengambil vektor nilai input dan mengubahnya menjadi probabilitas yang berjumlah satu.

Fungsi ini memetakan input ke dalam distribusi probabilitas yang digunakan untuk memprediksi kelas yang paling mungkin. Cross-entropy, di sisi lain, adalah fungsi rugi yang digunakan untuk mengukur perbedaan antara distribusi probabilitas yang diprediksi oleh model dan distribusi probabilitas yang sebenarnya dari label yang benar.

Tujuan dalam pelatihan adalah untuk meminimalkan nilai cross-entropy, yang berarti membuat prediksi model semakin mendekati label yang benar. Dalam PyTorch, softmax dan cross-entropy sering digunakan bersama-sama dalam proses klasifikasi, di mana softmax digunakan sebagai fungsi aktivasi pada lapisan output model, sementara cross-entropy digunakan sebagai fungsi rugi untuk menghitung kesalahan dalam prediksi model.







Activation Functions

Dalam PyTorch, fungsi aktivasi digunakan dalam jaringan saraf tiruan untuk memperkenalkan non-linearitas ke output lapisan-lapisan jaringan. Fungsi aktivasi berperan dalam memperbaiki kemampuan model untuk mempelajari hubungan yang kompleks antara input dan output.

Beberapa fungsi aktivasi yang umum digunakan termasuk ReLU (Rectified Linear Unit), sigmoid, dan tanh. ReLU mengaktifkan output dengan mengabaikan nilai negatif dan mempertahankan nilai positif. Sigmoid memetakan input ke dalam rentang 0 hingga 1, dan sering digunakan dalam tugas klasifikasi biner. Tanh memetakan input ke dalam rentang -1 hingga 1 dan dapat digunakan dalam tugas klasifikasi multikelas.

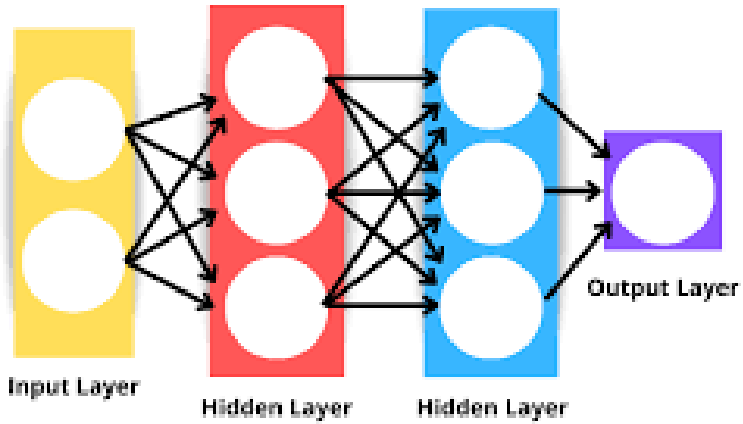
Fungsi aktivasi membantu dalam mengatasi masalah linieritas, meningkatkan kemampuan jaringan untuk menggeneralisasi pola yang kompleks, dan mencegah vanishing gradient problem.

Feedforward

Dalam PyTorch, feedforward (pengumpanan maju) mengacu pada aliran data melalui jaringan saraf tiruan secara sekuensial dari lapisan input ke lapisan output tanpa adanya siklus atau koneksi mundur.

Dalam arsitektur jaringan saraf feedforward, informasi mengalir ke depan melalui lapisan-lapisan jaringan, dan setiap lapisan menerapkan transformasi linier dan non-linear pada inputnya. Proses ini melibatkan komputasi dari lapisan input hingga lapisan output, di mana setiap lapisan menerapkan fungsi aktivasi pada hasil komputasi linier dari lapisan sebelumnya.

Feedforward jaringan saraf umumnya digunakan dalam tugas klasifikasi, di mana input data melewati serangkaian lapisan tersembunyi (hidden layers) untuk menghasilkan prediksi output.



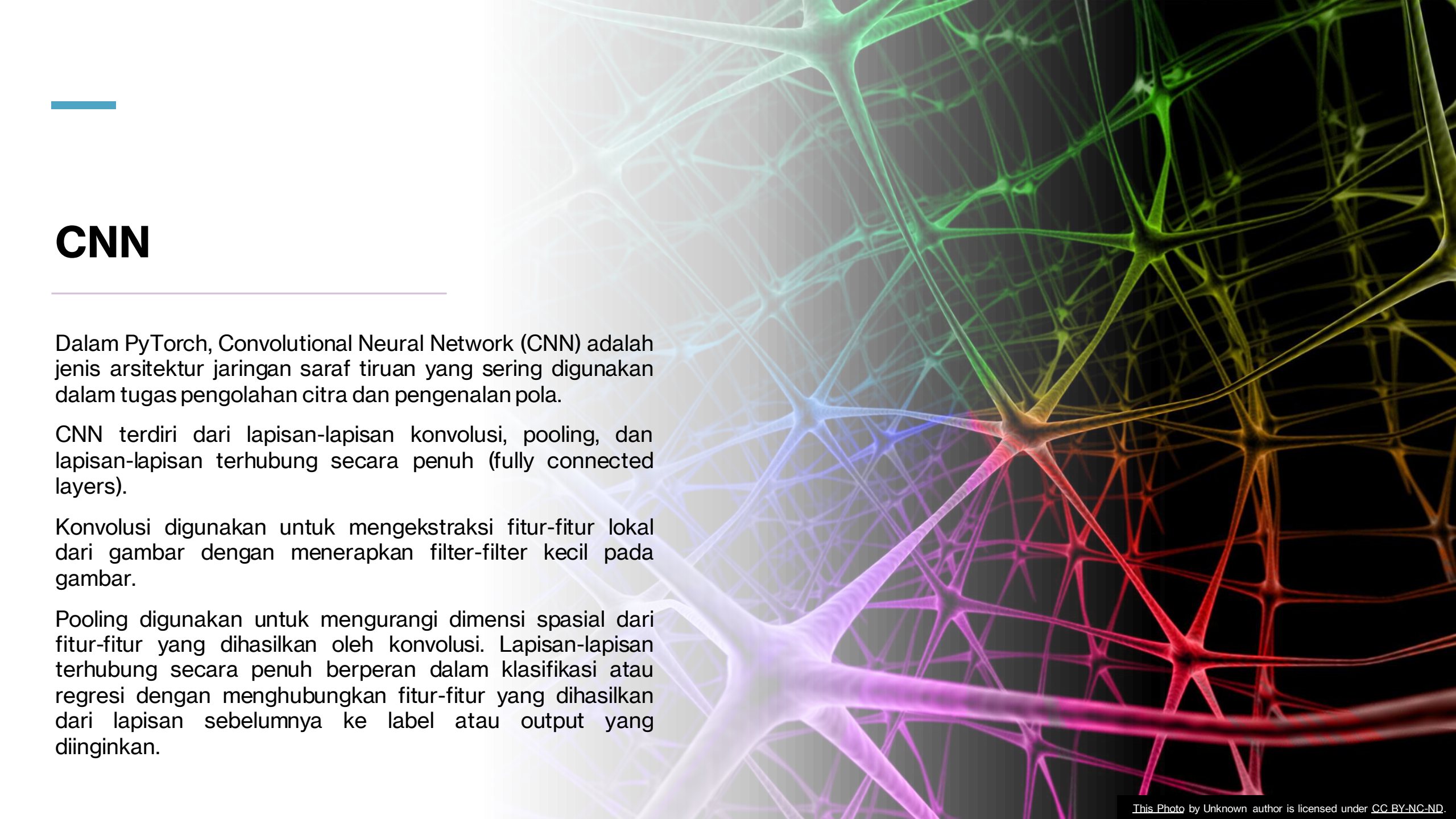
CNN

Dalam PyTorch, Convolutional Neural Network (CNN) adalah jenis arsitektur jaringan saraf tiruan yang sering digunakan dalam tugas pengolahan citra dan pengenalan pola.

CNN terdiri dari lapisan-lapisan konvolusi, pooling, dan lapisan-lapisan terhubung secara penuh (fully connected layers).

Konvolusi digunakan untuk mengekstraksi fitur-fitur lokal dari gambar dengan menerapkan filter-filter kecil pada gambar.

Pooling digunakan untuk mengurangi dimensi spasial dari fitur-fitur yang dihasilkan oleh konvolusi. Lapisan-lapisan terhubung secara penuh berperan dalam klasifikasi atau regresi dengan menghubungkan fitur-fitur yang dihasilkan dari lapisan sebelumnya ke label atau output yang diinginkan.

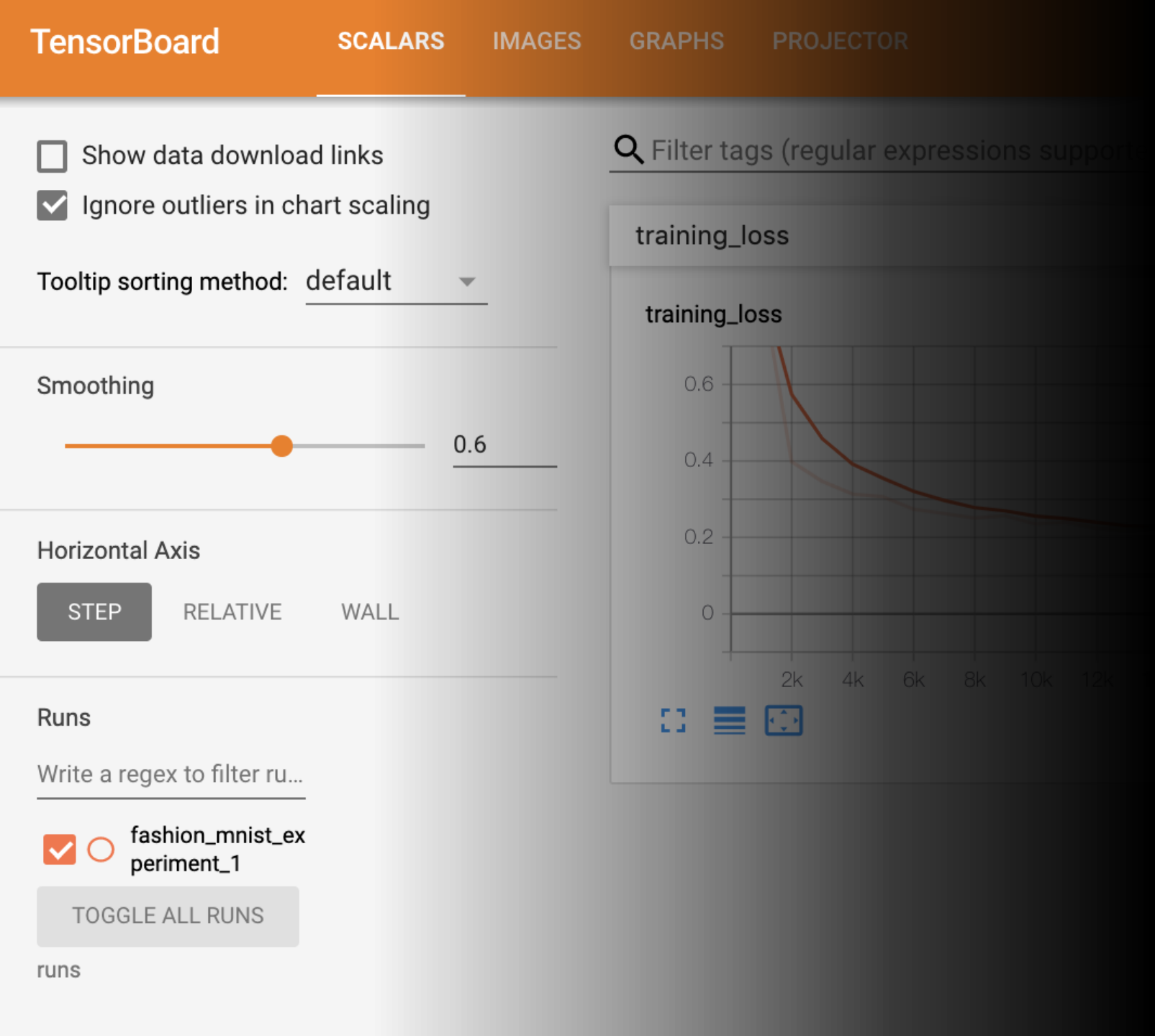


Transfer Learning

Dalam PyTorch, transfer learning adalah teknik yang digunakan untuk memanfaatkan pengetahuan yang telah dipelajari oleh model yang sudah dilatih sebelumnya pada tugas yang berbeda untuk meningkatkan kinerja model pada tugas baru yang serupa.

Dalam transfer learning, biasanya model yang sudah dilatih pada dataset besar dan kompleks, seperti ImageNet, digunakan sebagai "model basis" atau "pre-trained model". Kemudian, lapisan-lapisan terakhir model basis dihapus atau disesuaikan (fine-tuning) untuk menyesuaikan dengan tugas baru.

Dengan menggunakan transfer learning, model dapat mengambil manfaat dari representasi fitur yang telah dipelajari oleh model basis pada tugas yang serupa, sehingga mempercepat dan meningkatkan pembelajaran pada tugas baru dengan dataset yang lebih kecil.



Tensorboard

PyTorch TensorBoard adalah sebuah alat visualisasi yang digunakan untuk melacak, memantau, dan menganalisis proses pelatihan model dalam PyTorch. TensorBoard memungkinkan pengguna untuk memvisualisasikan metrik-metrik penting seperti loss, akurasi, dan gradien selama pelatihan dalam bentuk grafik, plot, dan tabel yang interaktif.

Dengan menggunakan TensorBoard, pengguna dapat dengan mudah memantau kinerja dan perkembangan model mereka seiring waktu, membandingkan eksperimen yang berbeda, serta mengidentifikasi masalah atau penyimpangan yang mungkin terjadi. TensorBoard juga menyediakan fitur-fitur seperti visualisasi distribusi nilai tensor, visualisasi lapisan-lapisan model, dan penjelajahan visual melalui representasi terlatih (embedding) untuk pemahaman yang lebih mendalam tentang model.

attachment

```python
# Operations
y = torch.rand(2, 2)
x = torch.rand(2, 2)
# addition
z = x + y
print(z)
```

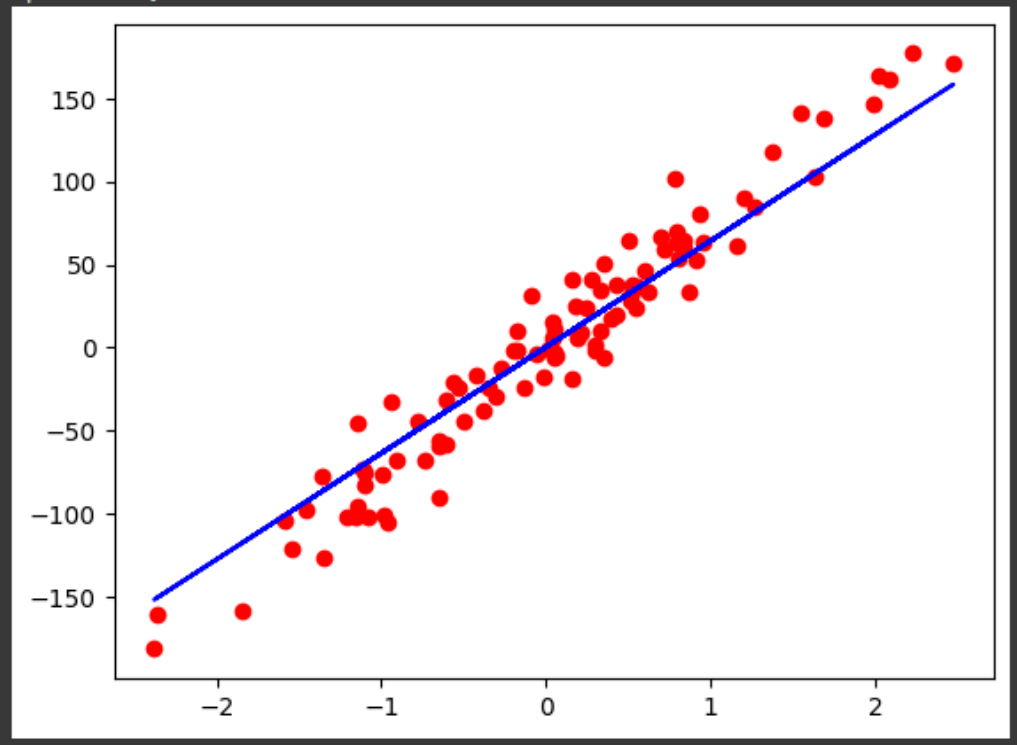
```
tensor([[0.4367, 1.7095],
        [1.5262, 1.0576]])
```

```python
x = torch.empty(2,2,3) # tensor, 3 dimensions
print(x)
```

```
tensor([[[0.0000e+00, 0.0000e+00, 0.0000e+00],
         [0.0000e+00, 2.3822e-44, 1.4586e-39]],

        [[1.0227e+04, 0.0000e+00, 1.0226e+04],
         [0.0000e+00, 0.0000e+00, 0.0000e+00]]])
```

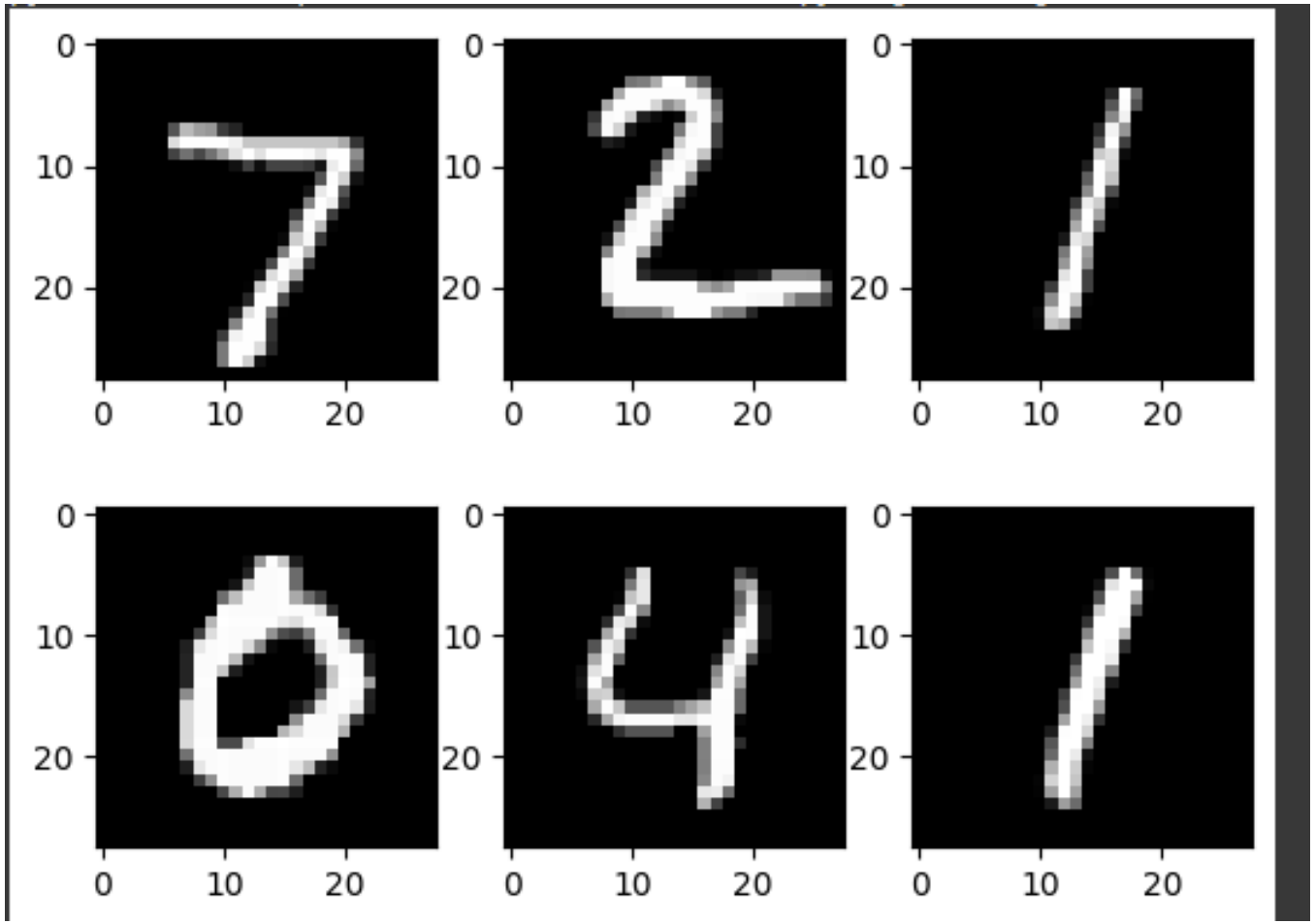
```
epoch: 10, loss = 4059.3955
epoch: 20, loss = 2859.6453
epoch: 30, loss = 2042.1500
epoch: 40, loss = 1484.9943
epoch: 50, loss = 1105.1877
epoch: 60, loss = 846.2231
epoch: 70, loss = 669.6155
epoch: 80, loss = 549.1484
epoch: 90, loss = 466.9595
epoch: 100, loss = 410.8750
```



```
Downloading http://yann.lecun.com/exdb/mni
Downloading http://yann.lecun.com/exdb/mni
100%|██████████| 1648877/1648877 [00:00<00
Extracting /data/MNIST/raw/t10k-images-idx

Downloading http://yann.lecun.com/exdb/mni
Downloading http://yann.lecun.com/exdb/mni
100%|██████████| 4542/4542 [00:00<00:00, 2


torch.Size([3, 1, 28, 28]) torch.Size([3])
```




```
Downloading https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz to
100%|██████████| 170498071/170498071 [00:01<00:00, 104597940.19it/s]
Extracting ./data/cifar-10-python.tar.gz to ./data
Files already downloaded and verified
```