Nama : Rizka Rahmadina

NIM : 1103204115

00. PyTorch Fundamentals

- Apa itu PyTorch?

PyTorch adalah sebuah framework machine learning open-source yang dikembangkan oleh Facebook's AI Research lab (FAIR). PyTorch dirancang untuk memberikan fleksibilitas tinggi dan interaktivitas yang memudahkan pengembangan model deep learning. Kelebihan utama PyTorch adalah sintaksis yang dinamis, memungkinkan pengguna untuk melakukan komputasi tensor dengan cara yang mirip dengan Python biasa.

- PyTorch bisa digunakan untuk apa?

PyTorch dapat digunakan untuk berbagai tugas dalam machine learning seperti pengenalan gambar, pemrosesan bahasa alami, prediksi, klasifikasi, dan sebagainya. PyTorch juga mendukung pembuatan model neural network dengan arsitektur yang kompleks dan canggih, membuatnya sangat berguna dalam riset dan pengembangan kecerdasan buatan.

- Siapa yang menggunakan PyTorch?

PyTorch digunakan oleh banyak perusahaan dan peneliti di seluruh dunia. Banyak perusahaan terkemuka seperti Facebook, Twitter, dan Tesla menggunakan PyTorch untuk pengembangan kecerdasan buatan. Selain itu, para peneliti dan praktisi di bidang machine learning dan deep learning memilih PyTorch karena fleksibilitas dan kejelasan sintaksisnya.

- Mengapa menggunakan PyTorch?

PyTorch memiliki sintaksis yang jelas dan mudah dipahami, yang membuatnya cocok untuk pemula dan ahli. PyTorch juga mendukung mode komputasi dinamis yang memudahkan eksperimen dan debugging. Kemampuan PyTorch untuk mendukung model dengan struktur yang kompleks, serta kemampuannya yang baik dalam pelatihan model pada GPU, membuatnya menjadi pilihan utama untuk banyak peneliti dan praktisi.

- Apa itu Tensor?

Tensor adalah struktur data utama dalam PyTorch. Secara sederhana, tensor adalah array multidimensional yang mirip dengan matriks atau vektor, tetapi dapat memiliki dimensi yang lebih tinggi. Tensor digunakan untuk merepresentasikan data dalam bentuk yang dapat dioperasikan oleh model machine learning.

- Apa tugas tensor?

Tensor memiliki tugas utama untuk menyimpan dan mengelola data numerik yang digunakan sebagai input dan output dalam model machine learning. Mereka juga berperan penting dalam operasi matematika yang diperlukan selama pelatihan dan inferensi model.

- Membuat Tensor

Membuat tensor dapat dilakukan dengan berbagai cara. Contohnya, dengan menggunakan konstruktor torch.tensor untuk mengonversi list Python menjadi tensor. Ini memungkinkan pembuatan tensor dengan nilai yang sudah ditentukan. Berikut adalah contohnya

```
import torch

# Membuat tensor dari list
tensor_list = torch.tensor([1, 2, 3])

# Membuat tensor berukuran 2x3 dengan nilai acak
tensor_random = torch.rand(2, 3)
```

- Tensor Acak

PyTorch menyediakan fungsi untuk membuat tensor dengan nilai acak. Hal ini sering digunakan, misalnya, saat inisialisasi bobot dalam neural network, dimana nilai-nilai tersebut perlu diinisialisasi secara acak. Berikut adalah contohnya

```
# Membuat tensor acak dengan distribusi normal
tensor_normal = torch.randn(2, 2)

# Membuat tensor acak integer antara 0 dan 5
tensor_randint = torch.randint(0, 5, (3, 3))
```

- Tensor Berisi Nol dan Satu

Inisialisasi tensor dengan nilai nol atau satu adalah langkah umum dalam pembuatan model. Ini memungkinkan kita untuk memulai dengan parameter yang dikenal. Berikut adalah contohnya

```
# Membuat tensor berisi nol dengan ukuran 2x2
tensor_zeros = torch.zeros(2, 2)

# Membuat tensor berisi satu dengan ukuran 3x3
tensor_ones = torch.ones(3, 3)
```

- Membuat Rentang dan Tensor Sejenisnya

Membuat tensor dengan rentang nilai atau dengan menggunakan struktur tensor lainnya dapat membantu dalam pengelolaan data dan mempersiapkannya untuk operasi lebih lanjut. Berikut adalah contohnya

```
# Membuat tensor dengan rentang nilai
tensor_range = torch.arange(0, 10, 2)

# Membuat tensor sejenis dengan tensor lain
tensor_like = torch.ones_like(tensor_range)
```

- Tipe Data Tensor

PyTorch mendukung berbagai tipe data untuk tensor, seperti float, integer, dan lainnya. Pemilihan tipe data yang tepat penting untuk memastikan akurasi dan efisiensi komputasi. Berikut adalah contohnya

```
# Membuat tensor dengan tipe data float
tensor_float = torch.tensor([1.0, 2.0, 3.0], dtype=torch.float32)

# Membuat tensor dengan tipe data integer
tensor_int = torch.tensor([1, 2, 3], dtype=torch.int64)
```

- Mendapatkan Informasi dari Tensor

Mengetahui informasi tentang tensor seperti ukuran atau tipe datanya dapat membantu dalam debugging dan pemrosesan data. Berikut adalah contohnya

```
# Mendapatkan ukuran tensor
size_tensor = tensor_random.size()

# Mendapatkan tipe data tensor
dtype_tensor = tensor_zeros.dtype
```

- Memanipulasi Tensor (Operasi Tensor)

Operasi tensor seperti penjumlahan, pengurangan, dan perkalian memungkinkan kita untuk melakukan perhitungan matematis pada data. Ini merupakan langkah penting dalam pelatihan model. Berikut adalah contohnya

```
# Penjumlahan dua tensor
result_addition = tensor_ones + tensor_zeros

# Perkalian dua tensor
```

```
result_multiply = torch.matmul(tensor_range, tensor_range)
```

- Operasi Dasar

Operasi dasar seperti transpose, reshape, dan slicing memungkinkan kita untuk mengubah bentuk dan struktur tensor sesuai kebutuhan. Berikut adalah contohnya

```
# Transpose tensor
tensor_transpose = tensor_random.t()

# Reshape tensor
tensor_reshape = tensor_range.view(2, 5)

# Slicing tensor
tensor_slice = tensor_random[:, 1:]
```

- Menemukan Min, Maks, Mean, Jumlah, dll. (Agregasi)

Operasi agregasi seperti menemukan nilai minimum, maksimum, mean, atau jumlah dari tensor membantu kita memahami distribusi dan statistik dari data. Berikut adalah contohnya

```
# Menemukan nilai minimum dan maksimum dalam tensor
tensor_min = torch.min(tensor_random)
tensor_max = torch.max(tensor_random)

# Menghitung mean dan jumlah nilai dalam tensor
tensor_mean = torch.mean(tensor_random)
tensor_sum = torch.sum(tensor_random)
```

- Ubah Tipe Data Tensor

Pengubahan tipe data tensor diperlukan terkadang untuk memastikan konsistensi dan kecocokan dalam perhitungan. Berikut adalah contohnya

```
# Mengubah tipe data tensor
tensor_int = tensor_float.to(torch.int32)
```

- Membentuk Ulang atau Mengubah Dimensi Tensor

Mengubah dimensi tensor dapat diperlukan saat kita ingin menyusun data untuk model atau mengubah bentuk tensor untuk keperluan tertentu. Berikut adalah contohnya

```
# Mengubah dimensi tensor
tensor_reshape = tensor_range.view(2, 5)
```

- Pengindeksan (Memilih Data dari Tensor)

Pengindeksan memungkinkan kita untuk mengakses atau memilih elemen atau bagian tertentu dari tensor. Berikut adalah contohnya

```
# Memilih elemen dari tensor berdasarkan indeks
selected_element = tensor_random[1, 2]

# Memilih baris atau kolom tertentu
selected_row = tensor_random[0, :]
```

- Tensor PyTorch & NumPy

Integrasi dengan NumPy memungkinkan kita untuk menggunakan fungsionalitas NumPy di dalam PyTorch dan sebaliknya. Hal ini mempermudah perpindahan data antara kedua framework. Berikut adalah contohnya

```
import numpy as np

# Konversi tensor PyTorch ke array NumPy
numpy_array = tensor_random.numpy()

# Konversi array NumPy ke tensor PyTorch
pytorch_tensor = torch.from_numpy(numpy_array)
```

- Reproducibilitas (Mencoba Mengambil yang Acak dari yang Acak)

Menetapkan seed memastikan bahwa operasi acak yang dilakukan oleh model dapat direproduksi, membantu dalam pemecahan bug dan perbandingan hasil eksperimen. Berikut adalah contohnya

```
# Mengatur seed untuk reproduktibilitas
torch.manual_seed(42)
```

- Menjalankan Tensor pada GPU (dan Membuat Komputasi Lebih Cepat)

Menggunakan GPU mempercepat perhitungan, terutama saat melibatkan operasi matematis yang intensif seperti dalam pelatihan model deep learning. Berikut adalah contohnya

```
# Menjalankan tensor pada GPU
cuda_tensor = tensor_random.cuda()
```

- Mendapatkan GPU

Mendapatkan perangkat GPU yang tersedia membantu kita mengetahui sumber daya yang dapat digunakan untuk akselerasi komputasi. Berikut adalah contohnya

```
# Mendapatkan perangkat GPU pertama
gpu_device = torch.cuda.current_device()
```

- Menjalankan PyTorch di GPU

Menjalankan operasi pada GPU dapat dilakukan dengan mentransfer tensor ke perangkat GPU yang sesuai. Berikut adalah contohnya

```
# Menjalankan operasi pada GPU
result_gpu = tensor_random.cuda() + tensor_ones.cuda()
```

- Menempatkan Tensor (dan Model) pada GPU

Memindahkan tensor atau model ke GPU diperlukan untuk memastikan bahwa komputasi dilakukan pada perangkat keras yang paling efisien. Berikut adalah contohnya

```
# Menempatkan tensor pada GPU
tensor_on_gpu = tensor_random.to('cuda')

# Menempatkan model pada GPU
model_on_gpu = model.to('cuda')
```

- Memindahkan Tensor Kembali ke CPU

Memindahkan tensor dari GPU ke CPU mungkin diperlukan setelah selesai komputasi, terutama jika hasilnya perlu digunakan bersama dengan operasi yang hanya dapat dilakukan di CPU. Berikut adalah contohnya

```
# Memindahkan tensor dari GPU ke CPU
tensor_on_cpu = tensor_on_gpu.cpu()
```

Dengan rangkaian fitur ini, PyTorch memberikan kekuatan dan fleksibilitas yang besar dalam pengembangan model machine learning, dari pemrosesan data hingga pelatihan model, serta implementasi di perangkat keras seperti GPU.

01. PyTorch Workflow Fundamentals

- Dasar-Dasar Alur Kerja PyTorch

Alur kerja PyTorch melibatkan beberapa langkah penting dalam pembelajaran mesin, seperti mengubah data, membuat model, menyesuaikan model dengan data, membuat prediksi, mengevaluasi model, menyimpan dan muat model, dan menyatukan semua hasil.

Dasar-Dasar Alur Kerja PyTorch melibatkan konsep fundamental dalam pengembangan dan pelatihan model deep learning. PyTorch, sebuah framework machine learning yang sangat populer, menggunakan struktur data yang disebut Tensor untuk merepresentasikan data numerik. Tensor ini serupa dengan array multidimensi dan memungkinkan pengguna untuk melakukan operasi matematika secara efisien. Selain itu, PyTorch menyediakan modul Autograd yang memungkinkan perhitungan gradien otomatis, yang merupakan kunci dalam proses pelatihan model.

- Mempersiapkan Data

Data yang digunakan dalam pembelajaran mesin bisa berupa apa saja, seperti tabel angka, gambar, video, file audio, struktur protein, teks, dan banyak lagi. Dalam contoh ini, kita akan membuat garis lurus sederhana untuk memahami cara bekerja PyTorch.

- Membangun Model

Kita akan membuat model yang menggunakan fungsi kerugian, pengoptimal, dan loop pelatihan untuk mempelajari pola dalam data. Model yang kita buat akan mencoba dan mempelajari hubungan antara fitur (X) dan label (y).

- Menyesuaikan Model dengan Data (pelatihan)

Kita punya data dan model, sekarang biarkan model (mencoba) menemukan pola dalam data (pelatihan). Kita akan membagi data menjadi set pelatihan dan pengujian, kemudian model akan menemukan pola dalam data pelatihan.

- Membuat Prediksi dan Mengevaluasi Model (inferensi)

Model kita menemukan pola dalam data, mari kita bandingkan temuannya dengan data aktual (pengujian). Kita akan mengevaluasi model apakah cocok atau tidak dengan metrik seperti ketataui, ke.

- Menyimpan dan Memuat Model

Model kita menemukan pola dalam data, mari kita bandingkan temuannya dengan data aktual (pengujian). Kita akan mengevaluasi model apakah cocok atau tidak dengan metrik seperti ketataui, ke.

- Menyatukan Semua Hasil

Semua materi untuk kursus ini tersedia di GitHub. Dan jika Anda mengalami kesulitan, Anda juga dapat mengajukan pertanyaan di halaman Diskusi di sana. Ada juga forum pengembang PyTorch tempat yang sangat membantu untuk segala hal tentang PyTorch.

- Mengenai Teknologi PyTorch

PyTorch adalah pustaka alat machine learning yang open-source yang dikelola oleh PyTorch Foundation. PyTorch dikenal sebagai framework machine learning yang sangat efisien dan memiliki beberapa fitur penting, seperti kemampuan untuk melakukan pelatihan deep learning, dukungan ekosistem CUDA untuk GPU dan CPU, dan kompatibilitas dengan berbagai sistem operasi.

- Mengenai Tujuan Kursus

Tujuan dari kursus ini adalah untuk mempelajari cara menggunakan PyTorch dalam pembelajaran mesin. Kita akan memulai dari nolak, memahami alur kerja PyTorch, membuat model, menyesuaikan model dengan data, membuat prediksi, mengevaluasi model, dan menyimpan model. Kita akan menggunakan garis lurus sederhana sebagai contoh untuk menjelaskan setiap langkah.

- Mengenai Sumber Daya Tambahan

Selain menggunakan PyTorch, kita juga akan menggunakan beberapa library tambahan seperti torch.nn (nn singkatan dari jaringan saraf dan paket ini berisi blok penyusun untuk membuat jaringan saraf di PyTorch) dan matplotlib.pyplot untuk membuat visualisasi.

- Mengenai Versi PyTorch

Versi PyTorch yang digunakan dalam contoh ini adalah 1.12.1+cu113. PyTorch adalah framework yang berkembang dengan cepat dan mendukung pengguna melalui versi yang terbaru. Selalu pastikan untuk memeriksa dokumentasi PyTorch untuk informasi terbaru dan perubahan.

- Alur Kerja PyTorch

Langkah-langkah utama dalam alur kerja PyTorch meliputi mengubah data menjadi angka, membangun model, menyesuaikan model dengan data, membuat prediksi, mengevaluasi model, menyimpan dan muat model, dan menyatukan semua hasil.

Alur Kerja PyTorch umumnya terdiri dari beberapa langkah. Pertama, pengguna mendefinisikan arsitektur model dengan membuat kelas yang mewarisi dari kelas dasar PyTorch, seperti torch.nn.Module. Dalam kelas ini, struktur model dan operasi yang akan dilakukan oleh setiap layer didefinisikan. Selanjutnya, pengguna menginisialisasi model dan memilih fungsi loss serta optimizer yang sesuai untuk tugas yang dihadapi. Data pelatihan kemudian dimuat dan disesuaikan dengan model menggunakan metode pengoptimalan, diikuti dengan langkah perhitungan gradien menggunakan Autograd. Setelah itu, optimizer diterapkan untuk mengupdate parameter model berdasarkan gradien yang dihitung. Proses ini diulang secara iteratif untuk meningkatkan performa model.

- Mengenali Kesalahan

Kesalahan bisa terjadi selama pelatihan atau inferensi. Kita perlu mengenali kesalahan tersebut untuk menyesuaikan model dan meningkatkan hasil.

- Menggunakan API PyTorch

Kita akan menggunakan modul PyTorch yang sesuai untuk membangun model, mengoptimalkan model, dan melatih model. Kita akan membuat model yang menggunakan fungsi kerugian, pengoptimal, dan loop pelatihan untuk mempelajari pola dalam data.

Cara Menggunakan API PyTorch dimulai dengan impor modul dan kelas yang diperlukan, seperti torch dan torch.nn. Kemudian, pengguna dapat membuat instance dari model yang telah didefinisikan sebelumnya dan menentukan konfigurasi lainnya. Data diolah dengan menggunakan tensor, dan operasi-operasi matematika dapat diterapkan menggunakan fungsi-fungsi PyTorch. Proses pelatihan dan evaluasi model melibatkan pemanggilan fungsi loss, perhitungan gradien, dan pengoptimalan parameter. Untuk inferensi, model yang telah dilatih dapat digunakan untuk membuat prediksi pada data baru. Seluruh alur kerja ini mendemonstrasikan fleksibilitas dan kekuatan PyTorch dalam membangun, melatih, dan menggunakan model machine learning.

- Menggunakan Data

Kita akan menggunakan data yang telah ada sebelumnya untuk melatih model. Data ini bisa berupa apa saja, seperti tabel angka, gambar, video, file audio, struktur protein, teks, dan banyak lagi.

01_pytorch_workflow menjelaskan tentang alur kerja PyTorch dalam pembelajaran mesin. Alur kerja ini melibatkan beberapa langkah, seperti mengubah data menjadi angka, membangun model, menyesuaikan model dengan data, membuat prediksi, mengevaluasi model, menyimpan dan memuat model, dan menyatukan semua hasil. Dalam contoh ini, kita akan membuat garis lurus sederhana untuk memahami cara bekerja PyTorch. Langkah pertama adalah mempersiapkan data dengan membuat garis lurus sederhana. Selanjutnya, kita akan

membuat model yang menggunakan fungsi kerugian, pengoptimal, dan loop pelatihan untuk mempelajari pola dalam data. Model yang kita buat akan mencoba dan mempelajari hubungan antara fitur (X) dan label (y). Setelah itu, kita akan membagi data menjadi set pelatihan dan pengujian, kemudian model akan menemukan pola dalam data pelatihan. Model yang telah belajar dapat digunakan untuk membuat prediksi pada data pengujian. Kita juga akan membahas cara menyimpan dan memuat model PyTorch, seperti menyimpan model ke dalam file dan memuatnya kembali ke dalam kelas model. Terakhir, kita akan menggabungkan semua hal yang telah kita pelajari sebelumnya, mulai dari persiapan dan pemuatan data hingga menyimpan dan memuat model. Dalam contoh ini, kita akan menggunakan PyTorch untuk membuat model yang mempelajari garis lurus sederhana dan mencocokkan model tersebut dengan data yang ada. Selain itu, pada 01_pytorch_workflow juga memberikan sumber daya tambahan seperti halaman Wikipedia untuk penurunan gradien dan propagasi mundur, serta forum pengembang PyTorch yang dapat membantu dalam memahami PyTorch.

02. PyTorch Neural Network Classification

- Masalah Klasifikasi

Meliputi klasifikasi biner, klasifikasi jamak, dan klasifikasi multi-label. Contohnya, klasifikasi biner dapat digunakan untuk memprediksi apakah seseorang mengidap penyakit jantung berdasarkan parameter kesehatannya. Klasifikasi jamak dapat digunakan untuk memutuskan apakah suatu foto berupa makanan, orang, atau anjing. Sedangkan klasifikasi multi-label dapat digunakan untuk memprediksi kategori apa yang harus ditetapkan pada artikel Wikipedia, misalnya matematika, sains, atau filsafat

- Arsitektur Jaringan Saraf Klasifikasi

Membahas lapisan masukan, lapisan tersembunyi, lapisan keluaran, serta hiperparameter klasifikasi biner dan klasifikasi multikelas. Buku catatan ini menjelaskan arsitektur umum jaringan saraf klasifikasi

Arsitektur jaringan saraf klasifikasi adalah struktur umum dari jaringan saraf yang digunakan untuk memecahkan masalah klasifikasi. Arsitektur ini terdiri dari beberapa lapisan, termasuk lapisan masukan, lapisan tersembunyi, dan lapisan keluaran. Lapisan masukan menerima data masukan, lapisan tersembunyi memproses data masukan, dan lapisan keluaran menghasilkan prediksi kelas. Selain itu, arsitektur jaringan saraf klasifikasi juga memiliki hiperparameter seperti bentuk lapisan masukan, lapisan tersembunyi, dan lapisan keluaran, serta fungsi aktivasi dan fungsi kerugian yang digunakan untuk melatih model. Arsitektur jaringan saraf klasifikasi dapat disesuaikan dengan masalah klasifikasi yang dihadapi, seperti klasifikasi biner atau klasifikasi multikelas. Buku catatan "Klasifikasi Jaringan Syaraf PyTorch" membahas arsitektur jaringan saraf klasifikasi secara rinci, termasuk hiperparameter klasifikasi biner dan klasifikasi multikelas

- Input dan Output Shapes

Menjelaskan pentingnya menghindari kesalahan bentuk (shape mismatch) dalam pembelajaran deep learning, serta mengetahui bentuk data masukan dan keluaran dalam konteks klasifikasi. Buku catatan ini menjelaskan bagaimana menghindari kesalahan bentuk dan mengetahui bentuk data masukan dan keluaran dalam konteks klasifikasi

- Alur Kerja PyTorch

Mulai dari persiapan data, pembangunan model, penyesuaian model dengan data (pelatihan), membuat prediksi, evaluasi model, hingga perbaikan model. Buku catatan ini menjelaskan alur kerja PyTorch untuk masalah klasifikasi

Alur Kerja PyTorch umumnya terdiri dari beberapa langkah. Pertama, pengguna mendefinisikan arsitektur model dengan membuat kelas yang mewarisi dari kelas dasar PyTorch, seperti torch.nn.Module. Dalam kelas ini, struktur model dan operasi yang akan dilakukan oleh setiap layer didefinisikan. Selanjutnya, pengguna menginisialisasi model dan memilih fungsi loss serta optimizer yang sesuai untuk tugas yang dihadapi. Data pelatihan kemudian dimuat dan disesuaikan dengan model menggunakan metode pengoptimalan, diikuti dengan langkah perhitungan gradien menggunakan Autograd. Setelah itu, optimizer diterapkan untuk mengupdate parameter model berdasarkan gradien yang dihitung. Proses ini diulang secara iteratif untuk meningkatkan performa model

- Contoh Model Klasifikasi Biner dan Jamak

Memberikan contoh model klasifikasi biner dan jamak sederhana, serta penggunaan fungsi kerugian dan pengoptimal yang sesuai. Buku catatan ini memberikan contoh model klasifikasi biner dan jamak sederhana, serta penggunaan fungsi kerugian dan pengoptimal yang sesuai.

Dalam konteks pembelajaran mesin, model klasifikasi biner merujuk pada model yang digunakan untuk memprediksi apakah suatu contoh data termasuk ke dalam salah satu dari dua kelas yang mungkin. Sebagai contoh, model klasifikasi biner dapat digunakan untuk memprediksi apakah seorang pelanggan akan membeli suatu produk atau tidak berdasarkan riwayat pembelian mereka. Di sisi lain, model klasifikasi jamak digunakan untuk memprediksi kelas dari suatu contoh data yang dapat termasuk ke dalam salah satu dari lebih dari dua kelas yang mungkin. Sebagai contoh, model klasifikasi jamak dapat digunakan untuk memprediksi jenis bunga berdasarkan sejumlah fitur tertentu, seperti panjang dan lebar kelopak. Dalam kedua kasus ini, model klasifikasi digunakan untuk memisahkan contoh data ke dalam kelas-kelas yang berbeda berdasarkan fitur-fitur yang diamati.

Klasifikasi biner dan klasifikasi jamak adalah dua jenis masalah klasifikasi yang berbeda dalam pembelajaran mesin. Klasifikasi biner melibatkan prediksi apakah suatu contoh data termasuk ke dalam salah satu dari dua kelas yang mungkin, seperti memprediksi apakah seseorang mengidap penyakit jantung atau tidak berdasarkan parameter kesehatannya. Di sisi lain, klasifikasi jamak melibatkan prediksi kelas dari suatu contoh data yang dapat termasuk ke dalam salah satu dari lebih dari dua kelas yang mungkin, seperti memprediksi jenis bunga berdasarkan sejumlah fitur tertentu, seperti panjang dan lebar kelopak. Dalam klasifikasi biner, target dapat berupa salah satu dari dua pilihan, sedangkan dalam klasifikasi jamak, target dapat berupa salah satu dari lebih dari dua pilihan. Selain itu, klasifikasi multi-label adalah jenis masalah klasifikasi lainnya di mana target dapat diberikan lebih dari satu opsi. Buku catatan "Klasifikasi Jaringan Syaraf PyTorch" membahas klasifikasi biner dan klasifikasi jamak secara rinci, serta memberikan contoh model klasifikasi biner dan jamak sederhana

- Evaluasi Model

Menjelaskan cara menggunakan metrik evaluasi yang tepat, tergantung pada konteks masalah klasifikasi yang dihadapi. Buku catatan ini menjelaskan cara menggunakan metrik evaluasi yang tepat dalam konteks masalah klasifikasi

- Perbaikan Model

Jika model tidak berhasil, menjelaskan langkah-langkah yang dapat diambil untuk memperbaikinya, seperti menyesuaikan fungsi kerugian, menambahkan lebih banyak lapisan, atau menggunakan dropout. Buku catatan ini menjelaskan langkah-langkah yang dapat diambil untuk memperbaiki model yang tidak berhasil

Pada PyTorch Neural Network Classification ini memberikan pemahaman mendalam tentang masalah klasifikasi dan cara menggunakan PyTorch untuk mengembangkan model klasifikasi yang efektif. Selain itu, PyTorch Neural Network Classification ini juga memberikan contoh dan praktik yang praktis untuk membantu memahami lebih baik tentang cara mengimplementasikan model klasifikasi dalam PyTorch

03. PyTorch Computer Vision

- Visi Komputer

Visi komputer adalah seni mengajarkan komputer untuk melihat. Ini melibatkan pembuatan model untuk mengklasifikasikan apakah suatu foto adalah kucing atau anjing (klasifikasi biner), atau apakah foto itu berupa kucing, anjing, atau ayam (klasifikasi kelas jamak), atau mengidentifikasi di mana mobil muncul dalam bingkai video (deteksi objek), atau mencari tahu di mana objek yang berbeda dalam suatu gambar dapat dipisahkan (segmentasi panoptik).

- Aplikasi Visi Komputer

Visi komputer digunakan dalam berbagai bidang, seperti dalam pengujian otomatis, deteksi objek, dan segmentasi. Misalnya, kamera keamanan menggunakan visi komputer untuk mendeteksi calon penyusup, produsen menggunakan visi komputer untuk mengidentifikasi cacat pada berbagai produk, dan mobil modern menggunakan visi komputer untuk menghindari mobil lain dan tetap berada dalam garis jalur.

- Perpustakaan Visi Komputer di PyTorch

PyTorch memiliki banyak perpustakaan visi komputer bawaan yang bermanfaat, seperti torchvision.datasets yang mengandung dataset kelasik seperti FashionMNIST, dan torchvision.models yang mengandung model architektur yang sering digunakan untuk masalah visi computer.

- Model dasar

Membuat model klasifikasi kelas jamak untuk mempelajari pola dalam data, menggunakan fungsi kerugian, pengoptimal, dan loop pelatihan. Model dasar kita akan digunakan untuk memahami pola dalam data FashionMNIST, yang mencakup 10 kelas berbeda gaya pakaian.

- Membuat prediksi dan mengevaluasi model

Buat beberapa prediksi dengan model dasar kita dan evaluasi. Model dasar kita akan digunakan untuk memahami pola dalam data FashionMNIST, yang mencakup 10 kelas berbeda gaya pakaian.

- Model 0 Membangun model dasar

Kita akan membuat model klasifikasi kelas jamak untuk mempelajari pola dalam data. Kita akan menggunakan fungsi kerugian, pengoptimal, dan membangun loop pelatihan.

- Model 0 Membangun model dasar

Kita akan membuat model klasifikasi kelas jamak untuk mempelajari pola dalam data. Kita akan menggunakan layer linear (nn.Linear()) sebagai dasar model. Model dasar ini akan digunakan untuk memahami pola dalam data FashionMNIST, yang mencakup 10 kelas berbeda gaya pakaian.

- Model 1 Menambahkan non-linearitas

Kita akan mencoba menambahkan non-linearitas ke model dasar kita dengan menambahkan layer yang berbeda. Model dasar kita akan digunakan untuk memahami pola dalam data FashionMNIST, yang mencakup 10 kelas berbeda gaya pakaian.

- Model 2 Jaringan Neural Konvolusional (CNN)

Jaringan Neural Konvolusional (CNN) adalah model yang sering digunakan dalam visi komputer. CNN menggunakan layer konvolusi untuk melacakkan fitur lokal dalam data, seperti pixel di sekitar suatu objek.

Kita akan membahas jaringan neural konvolusional (CNN) sebagai model yang sering digunakan dalam visi komputer. Jaringan neural konvolusional menggunakan layer konvolusi untuk melacakkan fitur lokal dalam data, seperti pixel di sekitar suatu objek.

- Model 3 Menggabungkan jaringan Neural Konvolusional dan Max Pooling

Layer Max Pooling digunakan untuk mengurangi dimensi spasial dari data, yang membantu mengurangi overfitting dan meningkatkan peluang melalui gradien yang lebih stabil.

Kita akan membahas cara menggabungkan jaringan Neural Konvolusional (CNN) dan layer Max Pooling untuk menciptakan model yang lebih efisien dalam mengidentifikasi fitur lokal dan mengurangi dimensi spasial dari data.

- Model 4 Menggunakan Dropout

Dropout adalah teknik di mana beberapa neuron secara acak selalu di nonaktifkan saat pelatihan, yang mengurangi overfitting dan meningkatkan kehilangan model.

Kita akan membahas cara menggunakan Dropout untuk mengurangi overfitting dalam model yang menggunakan CNN. Dropout adalah teknik di mana beberapa neuron secara acak selalu di nonaktifkan saat pelatihan, yang mengurangi overfitting dan meningkatkan kehilangan model.

- Model 5 Menggunakan Batch Normalization

Batch Normalization (BatchNorm) normalisasi data dengan menghitung statistik median dan ketiga kuartil data setiap fitur, yang membantu melacakkan model dan meningkatkan peluang melalui gradien yang lebih stabil.

Kita akan membahas cara menggunakan Batch Normalization (BatchNorm) dalam model CNN. BatchNorm normalisasi data dengan menghitung statistik median dan ketiga kuartil data setiap fitur, yang membantu melacakkan model dan meningkatkan peluang melalui gradien yang lebih stabil.

- Model 6 Menggunakan Activation ReLU

Fungsi aktif ReLU (Rectified Linear Unit) menghasilkan nilai positif jika input lebih besar daripada nol, dan nol jika tidak, yang membantu melatih model dengan lebih efisien.

Kita akan membahas cara menggunakan fungsi aktif ReLU (Rectified Linear Unit) dalam model CNN. Fungsi aktif ReLU menghasilkan nilai positif jika input lebih besar daripada nol, dan nol jika tidak, yang membantu melatih model dengan lebih efisien.

- Model 7 Menggunakan MaxOut

MaxOut adalah metode yang bertujuan untuk mengatasi masalah dengan gradient vanishing, yang terjadi ketika gradien menjadi sangat kecil di layer yang menggunakan fungsi aktif non-linear seperti ReLU.

Kita akan membahas cara menggunakan MaxOut dalam model CNN. MaxOut adalah metode yang membantu mengatasi masalah dengan gradient vanishing, yang terjadi ketika gradien menjadi sangat kecil di layer yang menggunakan fungsi aktif non-linear seperti ReLU.

- Model 8 Menggunakan Activation ReLU dan MaxOut bersamaan dengan Dropout

Kita akan membahas cara menggabungkan Dropout, ReLU, dan MaxOut dalam model CNN untuk menciptakan model yang lebih efisien dan mempengaruhi hasil. Kita akan menggunakan model ini untuk melakukan klasifikasi pada dataset FashionMNIST.

- Model 9 Melakukan pelatihan

Kita akan melatih model yang kita buat dengan menggunakan dataset FashionMNIST untuk melakukan klasifikasi pada gambar pakaian. Kita akan menggunakan loss fungsi categorical cross entropy untuk mengukur kesalahan dalam prediksi klas.

- Model 10 Memvalidasi model

Kita akan menggunakan metode k-fold cross-validation untuk mengvalidasi model yang kita buat. Kita akan mengelompokkan dataset FashionMNIST menjadi k-fold dan mengulai pelatihan model sebanyak kali kita inginkan, pengguna k-fold.

Metode k-fold cross-validation adalah teknik pelatihan machine learning yang membagi data menjadi beberapa bagian (k-fold), di mana setiap bagian akan digunakan sebagai data pelatihan, validasi, dan tes setiap kali. Ini membantu mengurangi overfitting dan meningkatkan kevaliditas model.

- Model 11 Menggunakan Transfer Learning

Kita akan membahas cara menggunakan transfer learning pada model yang sudah ada dalam PyTorch, seperti model pre-trained model ImageNet. Kita akan menggunakan transfer learning untuk menghasilkan model yang sudah terpre-trained dan menunjukkan model apa yang sudah dapat dikembangkan dengan baik di berbagai dataset.

Transfer learning adalah teknik dalam machine learning di mana model yang telah latih sebelumnya digunakan untuk mengatasi masalah baru. Model yang telah latih sebelumnya, seperti yang digunakan dalam buku "PyTorch Computer Vision", dapat Anda digunakan untuk melatih model baru dengan mengganti layer output dengan layer yang baru.

- Comparing our models

Kita telah membangun tiga model yang berbeda, harap kita membandingkan mereka. Kita akan menggunakan model dasar, model 1 (model linier), dan model 2 (model non-linier).

- Evaluating our best model

Kita akan membuat beberapa prediksi dengan model terbaik kita dan mengevaluasinya. Kita akan menggunakan matriks kesalahan (confusion matrix) untuk mengevaluasi model klasifikasi, termasuk model dasar, model 1, dan model 2.

- Menyimpan dan membaca model

Kita akan menyimpan model yang berhasil dan memastikan model tersebut dapat dimuat kembali dengan benar. Kita akan menggunakan PyTorch untuk menyimpan dan membaca model.

PyTorch Computer Vision membahas topik tentang visi komputer dan aplikasinya dalam berbagai bidang seperti smartphone, mobil modern, produsen, kamera keamanan, dan sistem pemantauan. PyTorch memiliki banyak perpustakaan visi komputer bawaan yang bermanfaat, seperti torchvision.datasets, torchvision.models, dan torch.utils.data.Dataset. Buku ini juga membahas tentang pembuatan model klasifikasi kelas jamak untuk mempelajari pola dalam data, menggunakan fungsi kerugian, pengoptimal, dan loop pelatihan. Selain itu, buku ini juga

membahas tentang jaringan neural konvolusional (CNN) yang sering digunakan dalam visi komputer. Buku ini juga membahas tentang evaluasi model dan pembuatan matriks kesalahan (confusion matrix) untuk mengevaluasi model klasifikasi. Terakhir, buku ini membahas tentang cara menyimpan dan membaca model yang berhasil dan sumber daya yang dapat digunakan untuk mendapatkan bantuan seperti dokumentasi PyTorch dan forum pengembang PyTorch. Selain itu, buku ini juga memberikan contoh kode dan visualisasi untuk membantu memahami topik yang dibahas.

04. PyTorch Custom Datasets

- Persiapan Data

Sebelum memulai proyek, perlu mempersiapkan data dengan menggunakan ImageFolder atau custom subclass dari torch.utils.data.Dataset. Beberapa langkah penting yang perlu diperhatikan meliputi mengimpor PyTorch dan menyiapkan kode agnostik perangkat, melakukan transformasi data, seperti resizing dan konversi menjadi tensor, dan memuat data dengan ImageFolder atau custom subclass dari torch.utils.data.Dataset.

- Augmentasi Data

Untuk meningkatkan kemampuan model, dapat melakukan augmentasi data. Beberapa teknik augmentasi data yang dapat digunakan meliputi flipping, cropping, dan adjusting brightness.

- Membangun Model

Setelah mempersiapkan data, dapat membangun model TinyVGG tanpa augmentasi data. Selanjutnya, dapat mengeksplorasi kurva loss dan membandingkan hasil model. Untuk meningkatkan kemampuan model, dapat membangun model TinyVGG dengan augmentasi data.

- Prediksi pada Gambar Makanan Buatan Sendiri

Setelah membangun model, dapat melakukan prediksi pada gambar makanan buatan sendiri. Dalam proses ini, penting untuk memahami data yang akan digunakan, seperti mengambil beberapa langkah untuk mengetahui data apa yang kita miliki.

- Kurikulum Tambahan

PyTorch Dataset adalah kelas yang diwarisi oleh torch.utils.data.Dataset, yang merupakan dasar kelas utama untuk setiap dataset dalam PyTorch. Setiap kelas dataset Anda akan menwarisi beberapa metode yang akan digunakan untuk mengelola data Anda, seperti __getitem__(), __len__(), dan __iter__()

DataLoader: DataLoader adalah kelas yang diwarisi oleh torch.utils.data.DataLoader, yang memungkinkan Anda untuk memodifikasi dataset Anda dengan cara yang efisien, seperti mengatur ukuran batch, mengatur jumlah thread, dan mengatur shuffle data

pathlib.Path.glob(): Fungsi ini digunakan untuk mengambil semua file yang cocok dengan pola yang diberikan dalam argumen. Dalam konteks ini, Anda dapat menggunakannya untuk mengambil semua file gambar dalam direktori yang Anda berikan

random.choice(): Fungsi ini digunakan untuk memilih satu gambar secara acak dari dataset yang Anda berikan. Anda dapat menggunakannya dalam loop pelatihan training dan testing untuk mengambil satu sampel data secara acak

pathlib.Path.parent.stem: Atribut ini digunakan untuk mendapatkan nama file tanpa ketikangan, yang dapat Anda gunakan untuk mengidentifikasi gambar yang berbeda dengan hanya menggunakan nama file

PIL.Image.open(): Fungsi ini digunakan untuk membuka file gambar menggunakan library PIL (Python Imaging Library)

torch.utils.data.Dataset: Anda dapat menggunakan kelas ini untuk membuat dataset custom Anda, seperti yang dijelaskan dalam contoh ImageFolderCustom

torch.utils.data.DataLoader: Anda dapat menggunakan kelas ini untuk membuat DataLoader yang akan digunakan dalam loop pelatihan training dan testing

model TinyVGG: Model TinyVGG adalah model VGG yang telah diadaptasi untuk beroperasi dengan jumlah data yang lebih kecil. Anda dapat menggunakannya untuk mengelola data image dan melatih model deep learning

Materi tersebut juga memberikan kurikulum tambahan untuk melatih pengetahuan tentang PyTorch Dataset dan DataLoader melalui kumpulan data PyTorch dan buku catatan tutorial pemuat data. Beberapa langkah yang dijelaskan dalam kurikulum tersebut meliputi mengambil semua path gambar menggunakan pathlib.Path.glob() untuk menemukan semua file yang berakhir dengan .jpg, memilih path gambar acak menggunakan random.choice(), mendapatkan nama kelas gambar menggunakan pathlib.Path.parent.stem, membuka gambar acak menggunakan PIL.Image.open(), menampilkan gambar dan mencetak beberapa metadata, dan membuat custom dataset class dengan meng-overwrite getitem method dari torch.utils.data.Dataset untuk mengembalikan satu sampel dari Dataset. Kemudian, dapat mengubah gambar menjadi DataLoader menggunakan torch.utils.data.DataLoader.

-