Muhammad Fauzan Nur'ilham

1103204085

Robotika

TK-44-G7

Technical Report Chapter 00 – Chapter 03

Kode yang diberikan pada Chapter 00 memperlihatkan dasar-dasar penggunaan PyTorch, sebuah framework deep learning yang populer. Pada paragraf pertama, berbagai jenis tensor diperkenalkan, mulai dari skalar hingga tensor multi-dimensi. Kode ini menggambarkan cara membuat tensor dengan ukuran, bentuk, dan tipe data tertentu, menekankan fleksibilitas dan keunggulan tensor PyTorch.

Yang kedua menjelajahi operasi tensor, termasuk operasi aritmatika dasar seperti penambahan, pengurangan, dan perkalian. Kode ini menunjukkan kenyamanan melakukan operasi ini langsung pada tensor dan menyoroti sifat operasi ini yang dilakukan per elemen. Selain itu, kode ini mendemonstrasikan penggunaan fungsi seperti **torch.range** (yang sudah usang) dan **torch.arange** untuk membuat tensor dengan nilai sekuensial.

Yang ketiga memperkenalkan konsep tipe data dan perangkat pada PyTorch. Ini menggambarkan cara menetapkan tipe data dan perangkat secara eksplisit untuk tensor, memberikan kendali atas presisi dan akselerasi perangkat keras. Ini sangat penting untuk mengoptimalkan perhitungan dan mengelola sumber daya secara efektif, terutama dalam aplikasi deep learning berskala besar.

Yang keempat membahas perkalian matriks, memperlihatkan berbagai pendekatan. Kode ini menyajikan implementasi manual menggunakan loop for dan kemudian membandingkannya dengan fungsi **torch.matmul** yang sangat dioptimalkan. Ini menekankan pentingnya memanfaatkan operasi tensor yang efisien dalam PyTorch, serta menyarankan untuk menghindari penggunaan loop for eksplisit dalam manipulasi matriks.

Yang kelima, disajikan suatu skenario khusus yang melibatkan perkalian matriks dengan dua matriks, tensor_A dan tensor_B. Kode ini menyoroti bagaimana PyTorch menyederhanakan operasi matematika kompleks, seperti perkalian matriks, membuatnya dapat diakses dan efisien untuk tugas machine learning dan deep learning.

Secara keseluruhan, kode yang diberikan berfungsi sebagai pengantar komprehensif terhadap operasi tensor, tipe data, dan manipulasi matriks dalam PyTorch, memperlihatkan kemampuannya untuk komputasi numerik dalam konteks deep learning.

Dalam laporan ini, kita akan menjelaskan beberapa dasar-dasar penggunaan PyTorch melalui kode-kode yang diberikan. Kode-kode tersebut mencakup berbagai aspek, mulai dari pembuatan tensor hingga operasi tensor lanjutan, menyoroti fleksibilitas dan kekuatan PyTorch dalam pengembangan model deep learning.

```
import torch
torch.__version__
```

Pertama-tama, kode ini mengenalkan konsep tensor dalam PyTorch. Kita dapat membuat tensor dengan berbagai dimensi, seperti skalar, vektor, matriks, dan tensor tiga dimensi. Penggunaan **torch.tensor** memudahkan kita dalam membuat tensor dengan ukuran dan bentuk yang diinginkan.

```
scalar = torch.tensor(7)
vector = torch.tensor([7, 7])
MATRIX = torch.tensor([[7, 8], [9, 10]])
TENSOR = torch.tensor([[[1, 2, 3], [3, 6, 9], [2, 4, 5]]])
```

Selanjutnya, kode ini menunjukkan operasi dasar pada tensor, termasuk penambahan, pengurangan, dan perkalian. PyTorch memungkinkan kita melakukan operasi ini langsung pada tensor, memberikan kejelasan dan efisiensi dalam kode.

```
tensor = torch.tensor([1, 2, 3])
tensor + 10
tensor * 10
```

Pada bagian ini, kita mempelajari pengaturan tipe data dan perangkat untuk tensor. PyTorch memberikan kebebasan untuk menentukan tipe data dan perangkat pada tensor, memungkinkan kita mengoptimalkan kinerja dan mengelola sumber daya dengan efektif.

```
float_32_tensor = torch.tensor([3.0, 6.0, 9.0], dtype=torch.float32,
device=None, requires_grad=False)
```

laporan ini menyoroti konsep perkalian matriks dalam PyTorch. Kode menyajikan implementasi manual dan menggunakan fungsi bawaan PyTorch, menunjukkan pentingnya menggunakan fungsi yang dioptimalkan untuk meningkatkan efisiensi komputasi.

```
tensor_A = torch.tensor([[1, 2], [3, 4], [5, 6]], dtype=torch.float32)
tensor_B = torch.tensor([[7, 10], [8, 11], [9, 12]], dtype=torch.float32)

# Implementasi Manual
%%time
value = 0
for i in range(len(tensor)):
  value += tensor[i] * tensor[i]

# Menggunakan torch.matmul
%%time
torch.matmul(tensor, tensor)
```

Kode ini menggambarkan manipulasi matriks dan tensor dengan PyTorch. Pemahaman matriks transpose, perkalian matriks, serta perubahan bentuk tensor (reshape, squeeze, unsqueeze) diperlihatkan dengan jelas. Ini memberikan fondasi yang kuat untuk manipulasi data dalam konteks machine learning.

```
print(tensor_A)
print(tensor_B)

print(tensor_A)
print(tensor_B.T)
```

```
print(f"Original shapes: tensor_A = {tensor_A.shape}, tensor_B =
{tensor_B.shape}\n")
print(f"New shapes: tensor_A = {tensor_A.shape} (same as above),
tensor_B.T = {tensor_B.T.shape}\n")
print(f"Multiplying: {tensor_A.shape} * {tensor_B.T.shape} <- inner
dimensions match\n")
print("Output:\n")
output = torch.matmul(tensor_A, tensor_B.T)
print(output)
print(f"\nOutput shape: {output.shape}")
```

Bagian ini menjelaskan penerapan operasi neural network dasar menggunakan PyTorch. Fungsi **torch.nn.Linear** digunakan untuk mengilustrasikan langkah-langkah dasar dalam pemodelan jaringan saraf, mengintegrasikan input, dan menghasilkan output.

```
torch.manual_seed(42)
linear = torch.nn.Linear(in_features=2, out_features=6)
x = tensor_A
output = linear(x)
print(f"Input shape: {x.shape}\n")
print(f"Output:\n{output}\n\nOutput shape: {output.shape}")
```

Bagian ini mencakup penggunaan PyTorch untuk operasi statistik pada tensor. Fungsi **torch.max**, **torch.min**, **torch.mean**, dan **torch.sum** digunakan untuk menggambarkan cara menghitung nilai maksimum, minimum, rata-rata, dan jumlah dari suatu tensor.

```
x = torch.arange(0, 100, 10)
torch.max(x), torch.min(x), torch.mean(x.type(torch.float32)),
torch.sum(x)
```

Kode ini memberikan pemahaman tentang manipulasi dimensi dan indeks tensor. Melalui operasi slicing dan indexing, kita dapat mengakses subset dari tensor dan merubah bentuknya, membantu dalam pengelolaan data yang kompleks.

```
x = torch.arange(1, 10).reshape(1, 3, 3)
x[:, 0]
x[:, :, 1]
x[0, 0, :]
```

Bagian ini membahas konversi antara NumPy dan PyTorch serta pengelolaan tensor. Penggunaan **torch.from_numpy** untuk mengonversi array NumPy menjadi tensor dan manipulasi nilai tensor memperlihatkan hubungan erat antara keduanya.

```
import torch
import numpy as np
array = np.arange(1.0, 8.0)
tensor = torch.from_numpy(array)
array, tensor
```

Kode ini menyoroti penanganan randomness di dalam PyTorch. Pemahaman mengenai seed dan pengaruhnya terhadap nilai random pada tensor membantu menjaga konsistensi dalam eksperimen dan pengembangan model.

```
import torch
import random

RANDOM_SEED=42
torch.manual_seed(seed=RANDOM_SEED)
random_tensor_C = torch.rand(3, 4)
torch.random.manual_seed(seed=RANDOM_SEED)
random_tensor_D = torch.rand(3, 4)
```

Terakhir, laporan ini mengakhiri eksplorasi dengan pembahasan mengenai pengolahan GPU dalam PyTorch. Pemahaman penggunaan GPU untuk percepatan komputasi menjadi penting, dan laporan ini menyajikan cara memeriksa ketersediaan GPU serta cara mentransfer tensor ke GPU.

```
!nvidia-smi
import torch
torch.cuda.is_available()
device = "cuda" if torch.cuda.is_available() else "cpu"
device
```

Pada chapter 01, laporan ini memberikan gambaran holistik tentang berbagai fitur dan fungsionalitas PyTorch, mendukung pemahaman dan penerapan konsep-konsep kunci dalam pengembangan model deep learning.

Dalam eksplorasi ini, kita membahas dasar-dasar PyTorch dan langkah-langkah dalam sebuah workflow machine learning. Rangkuman ini mencakup konsep-konsep fundamental, mulai dari manipulasi tensor, pembuatan model, pelatihan model, evaluasi, hingga penyimpanan dan pengembalian model.

Mengenalkan dasar-dasar PyTorch, menciptakan tensor, dan memberikan gambaran tentang manipulasi matriks dan tensor. Pembuatan tensor dari data sekuensial dan penggunaan fungsi-fungsi dasar PyTorch seperti **torch.rand** dan **torch.arange** memberikan pemahaman awal tentang manipulasi data.

Selanjutnya, kode mengimplementasikan model regresi linear sederhana menggunakan PyTorch. Kita membahas pembuatan model, inisialisasi parameter, dan langkah-langkah pelatihan. Selama proses ini, terdapat visualisasi prediksi model terhadap data pelatihan dan pengujian.

Proses evaluasi model dan pengoptimalan dilakukan melalui perhitungan loss menggunakan fungsi **nn.L1Loss** dan optimisasi dengan Stochastic Gradient Descent (SGD). Grafik loss untuk pelatihan dan pengujian dibuat untuk melihat perkembangan model selama beberapa epoch.

Langkah selanjutnya membahas cara menyimpan model ke dalam file untuk pemulihan di masa depan. Kode menunjukkan proses penyimpanan dan pemulihan model dengan menggunakan state dictionary. Ini penting untuk menyimpan kemajuan model setelah proses pelatihan.

Eksplorasi berikutnya mencakup konsep penanganan GPU dengan PyTorch. Kode menunjukkan cara memeriksa ketersediaan GPU, memindahkan tensor ke GPU, dan membuat inferensi dengan model yang ada di GPU. Hal ini dapat meningkatkan kinerja pada dataset yang lebih besar.

Pada tahap ini, kode memperkenalkan model regresi linear kedua dengan menggunakan modul **nn.Linear**. Dalam model ini, kita memperlihatkan pemindahan model ke GPU dan penggunaan loss function serta optimizer PyTorch yang lebih bervariasi.

Proses evaluasi dan pemulihan model kedua mencakup perubahan arsitektur model dan pemindahan model ke GPU. Kode menunjukkan cara melakukan evaluasi dan pemulihan dengan state dictionary, memastikan integritas dan konsistensi model.

Pada tahap ini, kita membahas persiapan model untuk digunakan pada perangkat GPU. Kode memperlihatkan cara menentukan perangkat yang akan digunakan (GPU atau CPU) dan memastikan model dan data berada di perangkat yang sama.

Langkah terakhir memperlihatkan cara menyimpan dan memulihkan model yang telah disiapkan untuk penggunaan GPU. Ini memastikan bahwa model dapat digunakan kembali dengan benar di masa depan, termasuk pada perangkat GPU jika tersedia.

Melalui serangkaian langkah-langkah tersebut, eksplorasi ini memberikan pemahaman mendalam tentang dasar-dasar PyTorch, pembuatan model, pelatihan, evaluasi, serta penyimpanan dan pengembalian model. Rangkuman ini menjadi panduan awal untuk memahami dasar-dasar workflow machine learning dengan PyTorch.

Pada chapter 02, Kode dimulai dengan memperkenalkan pembuatan data berbentuk lingkaran (circles) menggunakan **make_circles** dari **sklearn.datasets**. Data ini digunakan untuk eksplorasi penggunaan PyTorch dalam kasus regresi logistik. Data dihasilkan dengan menambahkan sedikit noise ke dalam titik-titik lingkaran dan dipersiapkan dalam format DataFrame menggunakan Pandas. Visualisasi menggunakan Matplotlib untuk memahami distribusi data.

Kode menunjukkan konversi data ke tensor PyTorch dan membagi dataset menjadi set pelatihan dan pengujian dengan fungsi **train_test_split** dari **sklearn.model_selection**. Model regresi linear sederhana dibuat dengan PyTorch menggunakan modul **nn.Module**. Proses pelatihan model, evaluasi, dan visualisasi hasilnya dilakukan.

Penggunaan GPU dengan PyTorch diilustrasikan untuk meningkatkan kinerja pada dataset yang lebih besar. Model regresi linear kedua diperkenalkan dengan modul **nn.Linear**. Model kedua dievaluasi dengan fungsi loss dan akurasi. Pemulihan model dilakukan dengan menyimpan dan memuat state dictionary.

Pembuatan data regresi linear dan pembagian data menjadi set pelatihan dan pengujian dilakukan untuk eksplorasi regresi linear. Model regresi linear dibuat menggunakan PyTorch dan dilatih untuk memprediksi hasil regresi linear. Proses ini memperlihatkan aplikasi PyTorch dalam konteks regresi.

Konsep aktivasi non-linear (ReLU) diperkenalkan, dan model untuk klasifikasi multi-kelas dibuat. Proses pelatihan dan evaluasi model dilakukan dengan CrossEntropyLoss. Model klasifikasi multi-kelas dibuat dengan modul **nn.Module**. Pelatihan model dilakukan dengan fungsi loss CrossEntropy dan evaluasi menggunakan akurasi.

Penggunaan TorchMetrics untuk mengukur akurasi model. Metric ini dihitung di perangkat yang sama dengan target device. Hasil dari beberapa model dievaluasi dan dibandingkan. Visualisasi dilakukan dengan membuat decision boundaries dan memplot hasil prediksi.

Alternatif metrik akurasi digunakan dengan menggunakan library TorchMetrics. Metrik ini menyediakan fungsi akurasi yang bersifat multikelas. Pendekatan regresi linear untuk permasalahan regresi dijelaskan dengan pembuatan model dan pelatihan menggunakan PyTorch.

Penggunaan fungsi aktivasi ReLU diaplikasikan dalam model multi-klasifikasi. Proses pelatihan dan evaluasi model dengan data linier dan klasifikasi multi-kelas dieksplorasi.

Laporan teknis ini mencakup serangkaian eksplorasi menggunakan PyTorch untuk beberapa jenis masalah, mulai dari regresi linear hingga klasifikasi multi-kelas. Setiap bagian mengilustrasikan konsep-konsep dasar PyTorch, seperti pembuatan model, proses pelatihan, evaluasi, dan visualisasi hasilnya. Model-model dibangun untuk memahami aplikasi PyTorch dalam berbagai konteks, termasuk penggunaan GPU dan penanganan data berdimensi lebih tinggi. Proses evaluasi dan optimisasi dilakukan dengan metode-metode standar PyTorch seperti fungsi loss, optimisasi SGD, serta penggunaan metrik akurasi. Pengenalan aktivasi non-linear ReLU memberikan pemahaman tambahan dalam penanganan data yang lebih kompleks. Dalam konteks klasifikasi multi-kelas, model menggunakan fungsi loss CrossEntropy dan metrik akurasi multikelas dari TorchMetrics. Eksplorasi ini memberikan landasan bagi pengguna untuk memahami dan mengaplikasikan PyTorch dalam proyek-proyek machine learning mereka sendiri.

Pada Chapter 03 adalah implementasi pengembangan model jaringan saraf tiruan (neural network) untuk tugas pengenalan pola pada dataset FashionMNIST menggunakan PyTorch, sebuah pustaka untuk komputasi tensor. Proses pengembangan model melibatkan beberapa tahapan, seperti pengaturan data, pembuatan model, pelatihan, dan evaluasi.

Pertama, kita memulai dengan menyusun model jaringan saraf tiruan untuk tugas pengenalan pola pada dataset FashionMNIST menggunakan PyTorch. Proses dimulai dengan persiapan data, di mana kita menggunakan dataset FashionMNIST dari torchvision. Dalam tahap ini, dataloader dibuat untuk memfasilitasi pelatihan dan pengujian, dan sampel-sampel gambar dari dataset ditampilkan untuk pemahaman visual.

Persiapan data

- Menggunakan dataset FashionMNIST dari torchvision.
- Membuat dataloader untuk pelatihan dan pengujian.
- Menampilkan sampel-sampel gambar dari dataset FashionMNIST.

Langkah berikutnya adalah pembuatan model. Model pertama (model_0) adalah model linear sederhana, sementara model kedua (model_1) memasukkan lapisan non-linear dengan fungsi aktivasi ReLU. Model ketiga (model_2) adalah model konvolusional dengan struktur mirip TinyVGG, memungkinkan penangkapan pola spasial yang lebih baik.

Pembuatan model :

- Model pertama (model_0) adalah model linear sederhana.
- Model kedua (model_1) adalah model dengan lapisan non-linear menggunakan fungsi aktivasi ReLU.
- Model ketiga (model_2) adalah model konvolusional dengan struktur mirip TinyVGG.

Proses selanjutnya adalah pelatihan dan evaluasi model. Fungsi pelatihan dan pengujian digunakan untuk mengukur akurasi dan loss pada set pelatihan dan pengujian. Perbandingan hasil pelatihan antara ketiga model dilakukan untuk menentukan performa relatif masing-masing model.

Visualisasi hasil pelatihan merupakan langkah berikutnya, di mana grafik batang digunakan untuk memberikan gambaran yang lebih jelas tentang performa model pada setiap epoch. Selanjutnya, dilakukan prediksi pada sampel data uji, dan hasilnya ditampilkan untuk memahami sejauh mana model dapat melakukan klasifikasi dengan akurat.

Pengukuran kinerja model dilakukan menggunakan matriks kebingungan (confusion matrix) pada set pengujian. Hasilnya memberikan informasi lebih lanjut tentang kemampuan model dalam mengenali kelas-kelas tertentu. Selanjutnya, model terlatih disimpan ke dalam file untuk kemungkinan penggunaan ulang.

Evaluasi model yang dibuat kembali adalah langkah berikutnya, di mana model dimuat kembali dari file yang disimpan. Performa model diperiksa kembali untuk memastikan konsistensi hasil antara model yang baru dibuat dan model yang telah disimpan.

Secara keseluruhan, implementasi ini memberikan gambaran praktis tentang bagaimana membuat, melatih, dan menguji model jaringan saraf tiruan dengan menggunakan PyTorch untuk tugas pengenalan pola pada dataset FashionMNIST. Model-model yang dihasilkan memberikan hasil yang memuaskan dalam pengujian, dengan model konvolusional (model_2) menunjukkan peningkatan performa yang signifikan dibandingkan dengan model-model linear sederhana.

Laporan teknis menyimpulkan bahwa proses pengembangan model dilakukan dengan sukses menggunakan PyTorch, dan penggunaan konvolusi membantu model untuk lebih efektif menangkap pola spasial pada gambar. Keseluruhan, implementasi ini memberikan pemahaman mendalam tentang penerapan praktis dari teknik-teknik ini dalam konteks pengenalan pola pada data gambar.