Nama : Agung Reynaldi Avizena

NIM : 1103204044

UAS Machine Learning: Learn PyTorch for Deep Learning

## 1. Chapter 00 – PyTorch Fundamentals

### a. Creating Tensors

- Skalar, Vektor, Matriks, dan Tensor

| Kode | Deskripsi |
|---|---|
| scalar = torch.tensor(7) | Membuat tensor skalar dengan nilai 7 |
| vector = torch.tensor([7, 7]) | Membuat tensor 1 dimensi (vektor) dengan dua elemen |
| MATRIX = torch.tensor([[7, 8], [9, 10]]) | Membuat tensor matriks 2x2 |
| TENSOR = torch.tensor([[[1, 2, 3], [3, 6, 9], [2, 4, 5]]]) | Membuat tensor 3 dimensi |
| random_tensor = torch.rand(size=(3, 4)) | Membuat tensor acak dengan ukuran (3, 4) |
| zeros = torch.zeros(size=(3, 4)) | Membuat tensor nol dengan ukuran (3, 4) |
| ones = torch.ones(size=(3, 4)) | Membuat tensor satu dengan ukuran (3, 4) |

- Informasi Tensor

| Kode | Deskripsi |
|---|---|
| tensor.ndim | Mendapatkan jumlah dimensi dari suatu tensor |
| tensor.shape | Mendapatkan bentuk (shape) dari suatu tensor |
| tensor.item() | Mendapatkan nilai Python dalam suatu tensor (hanya untuk tensor satu elemen) |
| tensor.dtype | Mendapatkan tipe data dari suatu tensor |

b. Operasi Tensor

- Operasi Matematika

| Kode | Deskripsi |
|---|---|
| tensor + 10 | Menambahkan 10 pada setiap elemen tensor |
| tensor * 10 | Mengalikan setiap elemen tensor dengan 10 |
| torch.matmul(tensor, tensor) | Perkalian matriks dua tensor |
| tensor * tensor | Perkalian elemen-wise dua tensor |

c. Manipulasi Tensor

- Reshaping, Stacking, Squeezing, dan Unsqueezing

| Kode | Deskripsi |
|---|---|
| x_reshaped = x.reshape(1, 7) | Mereshape tensor menjadi (1, 7) |
| x.view(1, 7) | Mengubah tampilan tensor menjadi (1, 7) |
| torch.stack([x, x, x, x], dim=0) | Mempile tensor satu di atas yang lain sepanjang dimensi 0 |
| x_reshaped.squeeze() | Menghapus dimensi tambahan dari suatu tensor |
| x_squeezed.unsqueeze(dim=0) | Menambah dimensi tambahan pada suatu tensor |

d. Pengindeksan Tensor dan Integrasi dengan NumPy

- Pengindeksan

| Kode | Deskripsi |
|---|---|
| x[:, :, 1] | Mendapatkan semua nilai dimensi 0 dan 1, namun hanya indeks 1 dari dimensi 2 |
| x[0, 0, :] | Mendapatkan indeks 0 dari dimensi 0 dan 1, serta semua nilai dari dimensi 2 |
| tensor.argmax() | Mengembalikan indeks di mana nilai maksimum terjadi |
| tensor.argmin() | Mengembalikan indeks di mana nilai minimum terjadi |

- Integrasi dengan NumPy

| Kode | Deskripsi |
|---|---|
| torch.from_numpy(array) | Mengubah array NumPy menjadi tensor PyTorch |
| tensor.numpy() | Mengubah tensor PyTorch menjadi array NumPy |

e. Reproduktibilitas

- Seed Acak

| Kode | Deskripsi |
|---|---|
| torch.manual_seed(seed=RANDOM_SEED) | Menetapkan seed acak untuk PyTorch |
| torch.random.manual_seed(seed=RANDOM_SEED) | Mereset seed untuk setiap tensor acak baru |

## 2. Chapter 01 - PyTorch Workflow Fundamentals

| Step | Deskripsi |
|---|---|
| Import Library | - Mengimpor library yang dibutuhkan seperti torch, matplotlib, nn (PyTorch's building blocks), pathlib, dan pprint.<br>- Menyiapkan data awal. |
| Data (Preparing and Loading) | - Menentukan parameter yang diketahui (weight dan bias).<br>- Membuat data menggunakan persamaan linear.<br>- Memisahkan data menjadi set pelatihan dan pengujian. |
| Plot Predictions Function | - Fungsi untuk memvisualisasikan data pelatihan, pengujian, dan prediksi. |
| Build Model | - Membuat kelas model regresi linear (LinearRegressionModel).<br>- Mendefinisikan parameter (weights dan bias) sebagai parameter yang dapat diubah.<br>- Mendefinisikan metode forward yang menghitung prediksi linear. |
| Checking Model Contents | - Menggunakan seed manual untuk parameter acak.<br>- Membuat instance model dan mengecek parameter menggunakan parameters() dan state_dict(). |
| Making Predictions with torch.inference_mode() | - Melakukan prediksi dengan model menggunakan torch.inference_mode().<br>- Mengecek hasil prediksi. |
| Train Model | - Menentukan fungsi loss (L1Loss) dan optimizer (SGD). |

| | |
|---|---|
| | - Melakukan loop pelatihan dengan metode Stochastic Gradient Descent.<br>- Mencetak dan memonitor loss pelatihan dan pengujian. |
| Plot Loss Curves | - Memvisualisasikan kurva loss pelatihan dan pengujian. |
| Find Model's Learned Parameters | - Mencetak parameter terpelajari dari model dan membandingkannya dengan nilai awal. |
| Making Predictions with Trained Model | - Menggunakan model yang telah dilatih untuk membuat prediksi pada data pengujian.<br>- Memvisualisasikan prediksi. |
| Saving and Loading a PyTorch Model | - Menyimpan model ke file menggunakan torch.save().<br>- Memuat kembali model dari file. |
| Putting it All Together | - Menyiapkan kode agar dapat dijalankan di perangkat apa pun.<br>- Membuat dan memvisualisasikan data. |
| Building a PyTorch Linear Model (Improved) | - Membangun model regresi linear dengan menggunakan nn.Linear untuk kejelasan.<br>- Memeriksa perangkat dan mengaturnya ke GPU jika tersedia. |
| Training Improved Model | - Melatih model baru selama 1000 epoch.<br>- Mencetak dan memonitor loss pelatihan dan pengujian. |
| Model Evaluation and Predictions | - Mengevaluasi model pada data pengujian.<br>- Membuat prediksi dan memvisualisasikannya. |
| Saving and Loading Improved Model | - Menyimpan dan memuat model yang telah ditingkatkan. |

### 3. Chapter 02 – Neural Network Classification

| Step | Deskripsi |
|---|---|
| Mengimport Pustaka | Mengimpor pustaka seperti NumPy, pandas, Matplotlib, PyTorch, scikit-learn, dan permintaan. Ini memastikan ketersediaan fungsi-fungsi penting untuk komputasi, manipulasi data, visualisasi, dan pembelajaran mendalam. |
| Membuat Data untuk Klasifikasi | Menghasilkan data melingkar menggunakan make_circles untuk klasifikasi. Visualisasikan data menggunakan plot sebar. |

| | |
|---|---|
| Bentuk Input dan Output | Mengidentifikasi fitur (X) dan label (y) beserta dimensinya, penting untuk desain model. |
| Membangun Model | Membangun model klasifikasi menggunakan multi-layer perceptron (MLP) dengan dua lapisan linier. Model juga direplikasi dengan pendekatan nn.Sequential. |
| Fungsi Kerugian dan Pengoptimal | Memilih nn.BCEWithLogitsLoss sebagai fungsi kerugian dan menggunakan Stochastic Gradient Descent (SGD) sebagai pengoptimal. |
| Loop Pelatihan dan Pengujian | Melakukan pelatihan dan pengujian melalui iterasi epoch dengan langkah maju, perhitungan kerugian, langkah mundur, dan optimalisasi. |
| Evaluasi Model | Mengukur akurasi model untuk mengevaluasi kemampuannya dalam mengklasifikasikan sampel. Visualisasi batas keputusan model. |
| Meningkatkan Model | Eksperimen dengan menambahkan lapisan dan melatih model lebih lama untuk meningkatkan kompleksitas dan kinerja model. |
| Mengatasi Kesalahan Kode | Mengimpor fungsi yang mungkin hilang, seperti plot_decision_boundary. |
| Mempersiapkan Data untuk Regresi | Membuat data linier untuk menjelajahi kemampuan model dalam mempelajari hubungan linier. Membagi data menjadi set pelatihan dan pengujian. |

## 4. Chapter 03 – Computer Vision

| Step | Deskripsi |
|---|---|
| Computer Vision Libraries in PyTorch | - Penerapan alur kerja PyTorch pada masalah computer vision.<br>- Fokus pada penggunaan library PyTorch khusus untuk computer vision. |
| Load Data | - Penggunaan torchvision.datasets.FashionMNIST() untuk mendownload dataset.<br>- Eksplorasi bentuk input dan output gambar serta jumlah sampel.<br>- Visualisasi kelas pakaian dalam dataset menggunakan matplotlib. |

| | |
|---|---|
| Prepare Data | - Penggunaan DataLoader untuk mempermudah proses loading data ke dalam model.<br>- Pembagian dataset menjadi batch atau mini-batch untuk efisiensi komputasi. |
| Model 0: Building a Baseline Model | - Pembangunan model dasar untuk klasifikasi multi-kelas.<br>- Penggunaan nn.Flatten() sebagai lapisan pertama untuk meratakan dimensi tensor.<br>- Persiapan model sebagai dasar untuk model-model berikutnya. |
| Making Predictions and Evaluating Model 0 | - Pembuatan fungsi untuk melakukan prediksi dan evaluasi model.<br>- Kode yang dapat berjalan pada CPU atau GPU untuk fleksibilitas perangkat. |
| Setup Device Agnostic Code for Future Models | - Implementasi best practices untuk penulisan kode yang agnostik perangkat.<br>- Persiapan kode untuk fleksibilitas penggunaan perangkat (CPU/GPU). |
| Model 1: Adding Non-linearity | - Eksperimen dengan menambahkan lapisan non-linear (nn.ReLU()) pada model.<br>- Modifikasi arsitektur untuk menangkap pola kompleks dalam data. |
| Model 2: Convolutional Neural Network (CNN) | - Pengenalan arsitektur Convolutional Neural Network (CNN).<br>- Penerapan CNN untuk tugas computer vision. |
| Comparing Our Models | - Penilaian dan perbandingan tiga model yang telah dibangun.<br>- Evaluasi berdasarkan metrik kinerja. |
| Evaluating Our Best Model | - Penerapan model terbaik pada gambar acak dan evaluasi hasilnya.<br>- Pembuatan matriks kebingungan untuk analisis lebih lanjut. |
| Making a Confusion Matrix | - Penjelasan matriks kebingungan sebagai alat evaluasi.<br>- Demonstrasi pembuatan dan interpretasi matriks kebingungan. |

| | |
|---|---|
| Saving and Loading the Best Performing Model | - Penyimpanan model terlatih untuk penggunaan masa depan.<br>- Demonstrasi memuat kembali model yang disimpan untuk memastikan kebenaran. |

## 5. Chapter 04 – Custom Dataset

| Step | Deskripsi |
|---|---|
| Mengimpor PyTorch dan Menyiapkan Kode yang Agnostik terhadap Perangkat | - Mengimpor PyTorch untuk pembuatan model.<br>- Menyiapkan kode agar dapat dijalankan pada perangkat manapun, mendeteksi ketersediaan GPU (cuda) atau menggunakan CPU. |
| Mendapatkan Data | - Mengunduh dataset kustom, sebuah subset dari Food101.<br>- Dataset mencakup gambar pizza, steak, dan sushi.<br>- Langkah awal untuk membangun dan melatih model sebelum memperluas dataset atau model. |
| Menjadi Satu dengan Data (Persiapan Data) | - Penting untuk memahami dan menjelajahi struktur dataset.<br>- Mendapatkan wawasan untuk pelatihan model dan perluasan potensial. |
| Transformasi Data | - Melakukan transformasi yang diperlukan pada gambar agar siap digunakan oleh model.<br>- Menyesuaikan dengan format atau struktur data awal. |
| Memuat Data dengan ImageFolder (Opsi 1) | - Menggunakan fungsi ImageFolder dari PyTorch untuk memuat data.<br>- Cocok untuk gambar dalam format klasifikasi standar. |
| Memuat Data Gambar dengan Dataset Kustom | - Membuat kelas Dataset kustom jika PyTorch tidak memiliki fungsi bawaan.<br>- Menawarkan fleksibilitas untuk menyesuaikan dataset sesuai kebutuhan. |
| Bentuk Lain dari Transformasi (Augmentasi Data) | - Mengeksplorasi augmentasi data menggunakan fungsi transformasi bawaan PyTorch. |

| | |
|---|---|
| | - Augmentasi membantu meningkatkan keragaman data pelatihan. |
| Model 0: TinyVGG tanpa Augmentasi Data | - Membangun model dasar (TinyVGG) tanpa menggunakan augmentasi data.<br>- Mengembangkan fungsi pelatihan dan evaluasi untuk model. |
| Menjelajahi Kurva Loss | - Memeriksa kurva loss untuk memahami kemajuan pembelajaran model.<br>- Berguna untuk mengidentifikasi kecenderungan underfitting atau overfitting. |
| Model 1: TinyVGG dengan Augmentasi Data | - Membangun model serupa dengan Model 0, namun kali ini dengan augmentasi data.<br>- Meningkatkan performa model dengan data yang telah di-augmentasi. |
| Membandingkan Hasil Model | - Membandingkan hasil dan kurva loss dari kedua model untuk evaluasi.<br>- Memberikan opsi dan diskusi untuk meningkatkan performa model. |
| Melakukan Prediksi pada Gambar Kustom | - Menggunakan model yang telah dilatih untuk membuat prediksi pada gambar di luar dataset pelatihan.<br>- Mendemonstrasikan cara menggunakan model untuk keperluan prediksi. |

## 6. Chapter 05 – Going Modular

| Step | Dekripsi |
|---|---|
| Persiapan Awal | - Import semua pustaka yang diperlukan seperti PyTorch, os, dan lainnya.<br>- Pilih mode sel sebagai notebook atau script. |
| Pengambilan Data | - Unduh dataset gambar kustom dari GitHub yang berisi kategori pizza, steak, dan sushi. |
| Persiapan Data | - Eksplorasi struktur dataset untuk memastikan data telah diunduh dan terorganisir dengan baik. |

| | |
|---|---|
| | - Periksa ketersediaan folder penyimpanan dataset.<br>- Unduh dan ekstraksi data jika belum ada. |
| Transformasi Data | - Resize gambar untuk keseragaman.<br>- Normalisasi nilai pixel dan konversi gambar menjadi tensor. |
| Memuat Data dengan ImageFolder (Opsi 1) | - Gunakan ImageFolder jika dataset mengikuti struktur standar. |
| Memuat Data Gambar dengan Dataset Kustom | - Buat kelas dataset kustom jika struktur dataset tidak standar.<br>- Berikan fleksibilitas dalam mengelola dataset kompleks. |
| Transformasi Lainnya (Augmentasi Data) | - Terapkan augmentasi data untuk variasi dataset pelatihan.<br>- Eksplorasi transformasi tambahan seperti rotasi, pemutaran, dll. |
| Model 0: TinyVGG tanpa Augmentasi Data | - Implementasikan TinyVGG, model konvolusional sederhana.<br>- Tentukan fungsi pelatihan dan evaluasi model. |
| Eksplorasi Kurva Loss | - Monitor kurva loss selama pelatihan.<br>- Analisis apakah model cenderung underfitting atau overfitting. |
| Model 1: TinyVGG dengan Augmentasi Data | - Buat versi TinyVGG dengan augmentasi data. |
| Membandingkan Hasil Model | - Bandingkan hasil dan performa kedua model.<br>- Diskusikan opsi untuk meningkatkan performa model. |
| Membuat Prediksi pada Gambar Kustom | - Terapkan model pada gambar kustom untuk membuat prediksi. |
| Penyimpanan Model | - Buat modul utilitas untuk menyimpan model selama pelatihan. |
| Train.py - Penggabungan Semua Komponen | - Gabungkan semua komponen ke dalam satu skrip (train.py).<br>- Tetapkan hyperparameter dan inisiasi serta pelatihan model. |

## 7. Chapter 06 – Transfer Learning

| Step | Deskripsi |
|---|---|
| Persiapan | - Menyiapkan dan mengunduh modul yang diperlukan.<br>- Mengambil direktori going_modular dari repositori pytorch-deep-learning.<br>- Memasang paket torchinfo jika belum terpasang. |
| Mengambil Data | - Men-download dataset pizza_steak_sushi.zip dari GitHub kursus.<br>- Memastikan dataset tidak diunduh ulang jika sudah ada. |
| Membuat Dataset dan DataLoaders | - Menerapkan skrip data_setup.py dari direktori going_modular untuk menyiapkan DataLoaders.<br>- Menyiapkan transformasi khusus untuk model torchvision.models. |
| Transformasi Manual untuk torchvision.models | - Menegaskan pentingnya konsistensi persiapan data dengan data pelatihan asli.<br>- Sebelum torchvision v0.13+, transformasi untuk model pretrained dilakukan secara manual. |
| Transformasi Otomatis untuk torchvision.models | - Mulai dari torchvision v0.13+, fitur pembuatan transformasi otomatis diperkenalkan.<br>- Memilih berat pretrained dan arsitektur model menggunakan torchvision.models.EfficientNet_B0_Weights.DEFAULT. |
| Mendapatkan Model Pretrained | - Menerapkan transfer learning untuk meningkatkan performa model dengan mengadopsi model yang sudah ada dan menyesuaikannya dengan kasus penggunaan spesifik.<br>- Mencari model klasifikasi pretrained di torchvision.models. |
| Melatih Model | - Membuat fungsi kerugian dan pengoptimal.<br>- Menggunakan nn.CrossEntropyLoss() sebagai fungsi kerugian.<br>- Menggunakan torch.optim.Adam() sebagai pengoptimal dengan lr=0.001.<br>- Melatih model selama 5 epoch menggunakan fungsi train() dari skrip engine.py. |

| | |
|---|---|
| Evaluasi Model dengan Plot Kurva Kerugian | - Membuat plot kurva kerugian menggunakan fungsi plot_loss_curves.<br>- Fungsi diambil dari helper_functions.py dan diunduh jika belum tersedia. |
| Prediksi pada Gambar dari Set Uji | - Melakukan prediksi pada gambar dari set uji dan menampilkan hasilnya.<br>- Membuat fungsi pred_and_plot_image() untuk melakukan prediksi dan plotting. |
| Prediksi pada Gambar Kustom | - Menguji model pada gambar kustom (pizza-dad.jpeg) dan membuat prediksi menggunakan fungsi pred_and_plot_image. |