Technical Report Using PyTorch Machine Learning
Final Semester Examination

Disusun Oleh :

Ibnutsany Tegar Bhagaskoro (1103204094)

Program Studi S1 Teknik Komputer

Fakultas Teknik Elektro

Telkom University

2023/2024

00. PyTorch Fundamentals

Apa itu PyTorch?

PyTorch adalah sebuah framework machine learning open-source yang digunakan untuk pengembangan dan pelatihan model deep learning. PyTorch dikembangkan oleh Facebook's AI Research lab (FAIR) dan dirilis secara resmi pada tahun 2016. Framework ini memiliki popularitas yang terus meningkat di kalangan peneliti dan praktisi machine learning karena desainnya yang mudah dipahami dan dinamis.

PyTorch dapat digunakan untuk apa?

PyTorch bersifat serba guna dan dapat digunakan untuk berbagai keperluan, termasuk:

- Pengenalan Gambar

  PyTorch banyak digunakan untuk membangun model deep learning yang dapat mengenali objek dalam gambar (object recognition), klasifikasi gambar, deteksi objek, segmentasi semantik, dan tugas-tugas pengolahan gambar lainnya.

- Pengolahan Suara

  Dalam bidang pengolahan suara, PyTorch dapat digunakan untuk tugas seperti pengenalan ucapan, sintesis ucapan, dan pemrosesan sinyal suara.

- Pemodelan dan Simulasi

  PyTorch dapat digunakan dalam pemodelan dan simulasi untuk berbagai aplikasi, termasuk fisika, biologi, dan ilmu lainnya.

- Generative Models

  PyTorch dapat digunakan untuk mengembangkan model generatif seperti Variational Autoencoders (VAEs) dan Generative Adversarial Networks (GANs) untuk generasi gambar baru, pembesaran gambar, dan tugas-tugas lainnya.

- Transfer Learning

  PyTorch mendukung transfer learning, yang memungkinkan pengguna untuk menggunakan model yang sudah dilatih pada dataset besar untuk tugas-tugas yang spesifik atau dataset yang lebih kecil.

Siapa yang menggunakan PyTorch?

PyTorch digunakan oleh berbagai kalangan, termasuk peneliti akademis, ilmuwan data, insinyur machine learning, dan perusahaan teknologi besar. Perusahaan dan institusi riset terkemuka menggunakan PyTorch untuk riset dan pengembangan kecerdasan buatan terdepan.

Mengapa Menggunakan PyTorch?

Fleksibilitas : PyTorch memiliki desain yang lebih fleksibel dan ekspresif. Ini memungkinkan pengembang untuk mengekspresikan ide dengan lebih mudah dan membuat kode lebih mudah dibaca dan dimengerti.
Community dan Dokumentasi yang Kuat : PyTorch memiliki komunitas yang aktif dan dukungan dokumentasi yang baik. Hal ini membuatnya mudah untuk menemukan sumber daya pembelajaran, tutorial, dan solusi masalah yang mungkin dihadapi saat pengembangan.
Dukungan GPU yang Baik : PyTorch mendukung penggunaan GPU untuk percepatan komputasi, memungkinkan pelatihan model dengan ukuran dan kompleksitas yang lebih besar.

Apa yang akan kita bahas dalam modul ini?

Modul ini akan membahas aspek-aspek dasar PyTorch, meliputi:

- Mengimpor PyTorch
- Pengenalan terhadap Tensor
- Membuat Tensor
- Tensor Acak
- Tensor Nol dan Satu
- Membuat Rentang dan Tensor Serupa
- Jenis Data Tensor

- Mendapatkan Informasi dari Tensor
- Memanipulasi Tensor
- Operasi Dasar
- Perkalian Matriks
- Salah satu kesalahan paling umum dalam pembelajaran mendalam (kesalahan bentuk)
- Mencari nilai min, maks, mean, sum, dll. (agregasi)
- Minimum/Maksimum Posisi
- Mengubah tipe data tensor
- Merombak, menggabungkan, mengecilkan, dan mengembangkan tensor
- PyTorch Tensors & NumPy
- Reproduktibilitas (mencoba menghilangkan unsur acak)
- Menjalankan tensor di GPU (dan membuat perhitungan lebih cepat)
- Mendapatkan GPU
- Menjalankan PyTorch di GPU
- Memindahkan tensor kembali ke CPU

Dari mana bisa mendapatkan bantuan?

Untuk bantuan dan diskusi, komunitas PyTorch aktif di forum seperti Stack Overflow, Reddit, dan forum diskusi resmi PyTorch. Selain itu, dokumentasi yang komprehensif tersedia untuk referensi.

Mengimpor PyTorch

Untuk mulai menggunakan PyTorch, impor perpustakaan ke dalam lingkungan Python dengan perintah berikut:

- python
- Copy code
- import torch

Pengenalan ke Tensor

Tensor adalah blok bangunan dasar dalam PyTorch, merepresentasikan larik multidimensi yang digunakan untuk perhitungan.

Membuat Tensor

Pelajari cara membuat tensor, termasuk spesifikasi nilai secara manual dan inisialisasi acak.

Tensor Acak

Jelajahi pembuatan tensor dengan nilai acak menggunakan fungsi acak PyTorch.

Zeros and ones

Buat tensor yang diinisialisasi dengan nol atau satu menggunakan fungsi PyTorch.

Membuat Rentang dan Tensor Serupa

Buat tensor menggunakan fungsi yang membuat urutan, serupa dengan fungsi range dalam Python.

Jenis Data Tensor

Pahami berbagai jenis data yang dapat dimiliki oleh tensor dan cara menentukannya.

Mendapatkan Informasi dari Tensor

Pelajari metode untuk mengekstrak informasi dari tensor, seperti ukuran, bentuk, dan dimensionalitas.

Memanipulasi Tensor

Jelajahi teknik dasar manipulasi tensor, termasuk pemotongan, indeks, dan penggabungan.

Operasi Dasar

Lakukan operasi tensor fundamental seperti penambahan, pengurangan, perkalian, dan pembagian.

Perkalian Matriks

Pahami cara melakukan perkalian matriks menggunakan PyTorch.

Kesalahan Paling Umum dalam Pembelajaran Mendalam (Kesalahan Bentuk)

Atasi kesalahan umum yang terkait dengan bentuk tensor dalam aplikasi pembelajaran mendalam.

Mencari Min, Max, Mean, Sum, dll. (Agregasi)

Pelajari cara mengumpulkan informasi dari tensor, termasuk menemukan nilai minimum, maksimum, mean, dan sum.

Posisi *Min/Max*

Identifikasi posisi nilai minimum dan maksimum dalam tensor.

Mengubah Tipe Data Tensor

Konversi tensor ke berbagai jenis data sesuai kebutuhan.

Merombak, Menggabungkan, Mengecilkan, dan Mengembangkan

Ubah bentuk tensor menggunakan operasi seperti merombak, menggabungkan, mengecilkan, dan mengembangkan.

PyTorch Tensors & NumPy

Pahami interoperabilitas antara tensor PyTorch dan array NumPy.

Reproduktibilitas (Mencoba Menghilangkan Unsur Acak)

Jelajahi teknik untuk meningkatkan reproduktibilitas dalam eksperimen yang melibatkan operasi acak.

Menjalankan Tensor di GPU (dan Membuat Perhitungan Lebih Cepat)

Kenalkan akselerasi GPU untuk meningkatkan kecepatan dan efisiensi perhitungan.

1. **Mendapatkan GPU**
   Pelajari cara mendapatkan dan mengatur GPU untuk perhitungan PyTorch.
2. **Menjalankan PyTorch di GPU**
   Konfigurasi PyTorch untuk menggunakan GPU dalam melatih model dan inferensi yang dipercepat.
3. **Memindahkan Tensor Kembali ke CPU**
   Pahami proses mentransfer tensor dari GPU kembali ke CPU untuk analisis atau visualisasi lebih

01. PyTorch Workflow Fundamentals

Apa yang Akan Dibahas:

Laporan ini membahas dasar-dasar alur kerja menggunakan PyTorch untuk pengembangan model. Mulai dari persiapan data hingga penggunaan model yang telah dilatih, setiap langkah akan dijelaskan secara rinci.

Dimana Anda Bisa Mendapatkan Bantuan?

Bantuan untuk penggunaan PyTorch dapat ditemukan melalui berbagai sumber. Dokumentasi resmi PyTorch, forum seperti Stack Overflow, dan komunitas pengguna PyTorch yang aktif adalah beberapa tempat yang baik untuk mencari solusi dan mendapatkan bantuan.

1. Data (Persiapan dan Pengisian):

- Membagi Data menjadi Set Pelatihan dan Pengujian:
  Langkah awal adalah membagi dataset menjadi dua bagian: set pelatihan untuk melatih model dan set pengujian untuk menguji performa model pada data yang belum pernah dilihat sebelumnya.

2. Membangun Model:

- Dasar-dasar Pembangunan Model PyTorch:
  Pada tahap ini, akan dibahas dasar-dasar dalam membangun model menggunakan PyTorch, termasuk definisi lapisan dan fungsi aktivasi.
- Memeriksa Isi Model PyTorch:
  Melakukan pemeriksaan terhadap struktur dan parameter dalam model PyTorch untuk memastikan konfigurasi yang benar.
- Membuat Prediksi menggunakan torch.inference_mode():
  Demonstrasi cara membuat prediksi menggunakan mode inferensi PyTorch, yang berguna ketika kita hanya ingin menggunakan model untuk inferensi tanpa memperbarui parameter.

3. Melatih Model:

- Membuat Loop Optimisasi di PyTorch:
  Membuat loop optimisasi untuk menyesuaikan parameter model berdasarkan nilai loss yang dihitung.
- Loop Pelatihan dan Pengujian di PyTorch:
  Membangun loop pelatihan dan pengujian yang menyeluruh untuk melatih dan menguji model pada setiap epok pelatihan.

4. Membuat Prediksi dengan Model PyTorch yang Telah Dilatih (Inferensi):

- Menyimpan dan Memuat state_dict() Model PyTorch:
  Menjelaskan proses menyimpan dan memuat parameter model menggunakan state_dict() untuk memastikan kelangsungan model.
- Menggabungkan Semuanya:

Menunjukkan bagaimana menyusun semua langkah sebelumnya: persiapan data, pembangunan model, pelatihan, dan inferensi, untuk mendapatkan alur kerja yang lengkap.

Dengan memahami langkah-langkah ini, kita dapat membangun dan mengelola alur kerja PyTorch secara efektif untuk pengembangan model machine learning.

02. PyTorch Neural Network Classification

Apa itu Masalah Klasifikasi?

Masalah klasifikasi adalah jenis tugas dalam pembelajaran mesin di mana tujuan utamanya adalah untuk mengelompokkan suatu data ke dalam kategori atau kelas tertentu. Dalam konteks ini, kita mencoba untuk memprediksi kelas atau label yang tepat untuk setiap data input.

Apa yang Akan Dibahas:

Dalam laporan ini, kita akan membahas dasar-dasar penggunaan PyTorch untuk menyelesaikan masalah klasifikasi dengan jaringan saraf. Poin-poin utama melibatkan pengantar masalah klasifikasi, lingkup materi, sumber bantuan yang tersedia, arsitektur dasar jaringan saraf untuk klasifikasi, dan langkah-langkah rinci untuk membangun, melatih, dan mengevaluasi model klasifikasi.

Dimana Anda Bisa Mendapatkan Bantuan?

Bantuan dapat ditemukan melalui berbagai sumber, termasuk dokumentasi resmi PyTorch, forum diskusi seperti Stack Overflow, dan komunitas pengguna PyTorch yang aktif.

0. Architecture of a classification neural network

Sebelum memulai, kita perlu memahami arsitektur dasar dari jaringan saraf untuk masalah klasifikasi. Ini mencakup lapisan input, lapisan tersembunyi, dan lapisan output. Lapisan tersembunyi biasanya mengandung fungsi aktivasi untuk mempelajari pola dalam data.

- Hyperparameter:
  Hyperparameter adalah parameter yang nilainya tidak dapat dipelajari oleh model dan harus diatur sebelum pelatihan. Contoh hyperparameter termasuk tingkat pembelajaran, jumlah epok, dan jumlah neuron dalam lapisan tersembunyi.
- Binary Classification dan Multiclass Classification:
  Klasifikasi biner melibatkan pemisahan data ke dalam dua kelas atau kategori, sedangkan klasifikasi multikelas melibatkan pemisahan data ke dalam lebih dari dua kelas.

1. Make Classification Data and Get it Ready

Input and Output Shapes

Penting untuk memahami bentuk input dan output data untuk membangun model yang sesuai.

Turn Data into Tensors and Create Train and Test Splits

Mengubah data ke dalam format tensor dan membuat pembagian antara data pelatihan dan pengujian.

2. Building a Model

Menyiapkan Fungsi Kerugian dan Optimizer

Loss Function/Optimizer, Problem Type, PyTorch Code

Menyiapkan fungsi kerugian dan optimizer sesuai dengan jenis masalah dan menggunakan kode PyTorch.

3. Train Model

Going from Raw Model Outputs to Predicted Labels

Mengubah output model menjadi label prediksi melalui proses logit, probabilitas prediksi, dan label prediksi.

Building a Training and Testing Loop

Membangun loop pelatihan dan pengujian untuk melatih dan menguji model.

4. Membuat Prediksi dan Mengevaluasi Model

Setelah model jaringan saraf kita dilatih menggunakan data pelatihan, langkah selanjutnya adalah membuat prediksi menggunakan data uji atau data yang belum pernah dilihat sebelumnya. Proses ini melibatkan beberapa tahapan:

Membuat Prediksi:

Model yang telah dilatih digunakan untuk membuat prediksi terhadap data uji. Prediksi ini dapat berupa nilai kelas (untuk masalah klasifikasi) atau nilai kontinu (untuk masalah regresi).

Evaluasi Model:

Evaluasi model penting untuk mengukur sejauh mana model mampu menggeneralisasi pada data yang belum pernah dilihat. Terdapat beberapa metrik evaluasi yang umum digunakan, tergantung pada jenis masalah yang dihadapi:

Untuk Masalah Klasifikasi:

- Akurasi (Accuracy): Persentase total prediksi yang benar.
- Presisi (Precision): Kemampuan model untuk tidak memberikan prediksi positif ketika seharusnya negatif.
- Recall (Sensitivity): Kemampuan model untuk menemukan semua instance yang seharusnya positif.
- F1-Score: Penggabungan presisi dan recall untuk memberikan metrik yang seimbang.
  Untuk Masalah Regresi:
- Mean Absolute Error (MAE): Rata-rata dari selisih absolut antara prediksi dan nilai sebenarnya.
- Mean Squared Error (MSE): Rata-rata dari kuadrat selisih antara prediksi dan nilai sebenarnya.
- Root Mean Squared Error (RMSE): Akar kuadrat dari MSE.
- Confusion Matrix (Untuk Masalah Klasifikasi):
  Matriks yang menyajikan jumlah prediksi benar dan salah berdasarkan kelas target. Berguna untuk mendapatkan wawasan lebih dalam tentang performa model.

5. Meningkatkan Model (dari Perspektif Model)

Meningkatkan model dari perspektif model melibatkan serangkaian langkah untuk membuat model lebih baik dalam menangani pola-pola yang kompleks dalam data. Berikut adalah beberapa tahapan umum yang dapat diambil untuk meningkatkan model:

- Persiapan Data untuk Melihat Apakah Model Kita Dapat Memodelkan Garis Lurus:
  Sebelum melakukan perbaikan pada model, pertama-tama kita perlu memastikan bahwa model dapat menangkap pola yang paling sederhana, seperti garis lurus. Ini melibatkan persiapan data sederhana dan pengujian model untuk melihat sejauh mana model dapat mempelajari hubungan yang sederhana.
- Menyesuaikan Model untuk Memfitting Garis Lurus:
  Jika model awal tidak mampu memodelkan pola yang sederhana, mungkin perlu dilakukan penyesuaian. Ini bisa melibatkan perubahan arsitektur model, penyesuaian hyperparameter seperti tingkat pembelajaran, atau penambahan lapisan dan neuron.

6. The Missing Piece: Non-linearity

Meningkatkan kemampuannya untuk menangkap pola-pola yang lebih kompleks dalam data. Fungsi aktivasi seperti ReLU atau Sigmoid sering digunakan untuk memperkenalkan non-linieritas.

- Recreating Non-linear Data (Red and Blue Circles):
  Untuk menguji model pada data yang lebih kompleks, kita dapat membuat ulang dataset yang melibatkan pola non-linier, seperti lingkaran merah dan biru.
- Building a Model with Non-linearity:
  Membangun kembali model dengan menambahkan fungsi aktivasi non-linier pada lapisan tersembunyi.

7. Replicating Non-linear Activation Functions

Pada tahap ini, kita akan mengeksplorasi fungsi aktivasi non-linier yang umum digunakan dalam jaringan saraf. Fungsi aktivasi non-linier, seperti ReLU (Rectified Linear Unit), Sigmoid, dan Tanh, memungkinkan model untuk memahami pola-pola yang lebih kompleks dalam data. Peniruan atau replikasi fungsi aktivasi ini melibatkan penerapan dan pemahaman bagaimana setiap fungsi berperilaku.

8. Menyatukan Semua dengan Membangun Model PyTorch Multi-kelas

- Membuat Data Klasifikasi Multi-kelas
  Pada tahap ini, kita akan membuat dataset yang melibatkan lebih dari dua kelas atau kategori. Data klasifikasi multi-kelas sering kali melibatkan lebih dari dua label yang mungkin, dan model yang dibangun harus dapat memprediksi kelas mana yang paling sesuai.
- Membangun Model Klasifikasi Multi-kelas di PyTorch
  Kami akan merancang dan mengimplementasikan arsitektur model jaringan saraf untuk menangani masalah klasifikasi multi-kelas. Ini melibatkan penentuan jumlah lapisan dan neuron, serta pemilihan fungsi aktivasi yang sesuai.
  Membuat Fungsi Kerugian dan Optimizer untuk Model PyTorch Multi-kelas
  Penentuan fungsi kerugian dan optimizer kritis untuk pelatihan model. Fungsi kerugian seperti CrossEntropy sering digunakan dalam skenario klasifikasi multi-kelas.
- Mendapatkan Probabilitas Prediksi untuk Model PyTorch Multi-kelas

Mendapatkan probabilitas prediksi dari model yang dilatih. Hal ini penting untuk memahami sejauh mana model yakin dalam menetapkan kelas tertentu kepada suatu input.

- Membuat Loop Pelatihan dan Pengujian untuk Model PyTorch Multi-kelas
  Pembangunan loop pelatihan dan pengujian khusus untuk model klasifikasi multi-kelas. Proses pelatihan melibatkan melalui setiap batch data, menghitung loss, dan memperbarui parameter model, sementara pengujian mengevaluasi performa model pada dataset yang tidak dilihat sebelumnya.

Dengan menyatukan semua langkah ini, kita dapat mengimplementasikan jaringan saraf yang mampu menangani tugas klasifikasi multi-kelas menggunakan PyTorch.

03. PyTorch Computer Vision

Dimana Computer Vision Digunakan?

Computer vision digunakan dalam berbagai bidang seperti:

- Pengenalan Objek: Identifikasi dan klasifikasi objek dalam gambar atau video.
- Deteksi Wajah: Pengenalan dan lokalisasi wajah manusia dalam gambar atau video.
- Segmentasi Gambar: Memisahkan dan mengelompokkan piksel-piksel dalam gambar.
- Rekonstruksi 3D: Membangun model tiga dimensi dari objek atau lingkungan berdasarkan gambar dua dimensi.

Apa yang Akan Dibahas:

Khususnya, laporan ini akan membahas:

Pustaka Computer Vision dalam PyTorch:

Pada bagian ini, kita menjelajahi beberapa pustaka PyTorch khusus yang digunakan untuk pengembangan computer vision.

1. Mendapatkan Dataset:

- Bentuk Input dan Output Model Computer Vision:
  Memahami bentuk umum input dan output dalam model computer vision, seperti (batch_size, channels, height, width).
- Visualisasi Data:
  Menunjukkan cara visualisasi data membantu pemahaman karakteristik dataset.

2. Menyiapkan DataLoader:

Pada tahap ini, kita menyiapkan DataLoader untuk mempermudah pemuatan data dalam bentuk batch.

3. Model 0: Membangun Model Dasar:

- Setup Loss, Optimizer, dan Metrik Evaluasi:
  Konfigurasi loss function, optimizer, dan metrik evaluasi untuk model baseline.
- Membuat Fungsi Waktu Eksperimen:
  Implementasi fungsi untuk mengukur waktu eksperimen dan pelatihan model.
- Membuat Loop Pelatihan dan Melatih Model pada Batch Data:
  Implementasi loop pelatihan untuk melatih model pada batch data.

4. Membuat Prediksi dan Mendapatkan Hasil Model 0:

Pada tahap ini, model baseline dilatih dan digunakan untuk membuat prediksi. Hasilnya dievaluasi untuk mendapatkan pemahaman awal tentang kinerja model.

5. Menyiapkan Kode Agnostik Perangkat:

Menunjukkan cara menulis kode yang dapat berjalan baik pada CPU maupun GPU jika tersedia.

6. Model 1: Membangun Model Lebih Baik dengan Non-linieritas:

- Setup Loss, Optimizer, dan Metrik Evaluasi:
  Konfigurasi loss function, optimizer, dan metrik evaluasi untuk model yang diperbaiki.
- Fungsi Training dan Testing:
  Mengubah loop pelatihan menjadi fungsi agar dapat digunakan kembali.

7. Model 2: Membangun Convolutional Neural Network (CNN):

- Meninjau nn.Conv2d():
  Pemahaman tentang lapisan konvolusi dalam PyTorch.
- Meninjau nn.MaxPool2d():
  Pemahaman tentang lapisan pooling dalam PyTorch.
- Setup Loss Function dan Optimizer untuk Model_2:
  Konfigurasi loss function dan optimizer untuk model CNN.
- Melatih dan Menguji Model_2:
  Melatih dan menguji model CNN menggunakan fungsi pelatihan dan pengujian yang sudah dibuat.

8. Membandingkan Hasil Model dan Waktu Pelatihan:

Hasil dan waktu pelatihan dari kedua model dibandingkan untuk mengevaluasi kinerja relatifnya.

9. Membuat dan Mengevaluasi Prediksi Acak dengan Model Terbaik:

Prediksi acak dibuat menggunakan model terbaik dan dievaluasi untuk memahami cara model bertindak pada data yang tidak terlihat sebelumnya.

10. Membuat Matriks Confusion untuk Evaluasi Prediksi Lanjutan:

Membuat matriks kebingungan untuk mendapatkan wawasan lebih lanjut tentang performa model dalam mengklasifikasikan setiap kelas.

11. Menyimpan dan Memuat Model Terbaik:

Model terbaik disimpan untuk penggunaan mendatang dan dapat dimuat kembali saat diperlukan.

Dengan memahami dan mengikuti langkah-langkah ini, pengembang dapat mengembangkan dan mengoptimalkan model computer vision menggunakan PyTorch.Menyimpan model terbaik untuk penggunaan mendatang.

04. PyTorch Custom Datasets

Apa itu Dataset Kustom

Dataset kustom adalah kumpulan data yang dibuat secara khusus untuk memenuhi kebutuhan dan karakteristik tertentu. Pada PyTorch, kita dapat membuat dataset kustom untuk menangani data yang tidak terstruktur atau memiliki format khusus.

Dimana Bisa Mendapatkan Bantuan?

Memberikan informasi tentang sumber bantuan yang dapat diakses.
Apa yang Akan Dibahas?

0. Import PyTorch dan Menyiapkan Kode yang Bersifat Perangkat Agnostik

Menunjukkan cara mengimpor PyTorch dan menulis kode yang dapat berjalan di berbagai perangkat.

1. Mendapatkan Data:

Menjelaskan langkah-langkah untuk mendapatkan data.

2. Menjadi Satu dengan Data (Persiapan Data):

- Visualisasi Sebuah Gambar:
  Menunjukkan cara visualisasi gambar dapat membantu memahami dataset.

3. Transformasi Data:

- Transformasi Data dengan torchvision.transforms:
  Mengilustrasikan cara melakukan transformasi pada data menggunakan torchvision.transforms.

4. Opsi 1: Memuat Data Gambar Menggunakan ImageFolder:

- Mengubah gambar yang dimuat menjadi DataLoader:
  Menjelaskan cara mengonversi gambar yang dimuat ke dalam DataLoader.

5. Opsi 2: Memuat Data Gambar dengan Dataset Kustom:

- Membuat fungsi pembantu untuk mendapatkan nama kelas:
  Menunjukkan cara membuat fungsi untuk mendapatkan nama kelas.
- Membuat Dataset Kustom untuk Replikasi ImageFolder:
  Menjelaskan langkah-langkah membuat dataset kustom untuk mereplikasi ImageFolder.
- Membuat fungsi untuk menampilkan gambar secara acak:
  Menunjukkan cara membuat fungsi untuk menampilkan gambar secara acak.
- Mengonversi gambar yang dimuat kustom ke dalam DataLoader:

Menunjukkan langkah-langkah mengonversi gambar yang dimuat kustom menjadi DataLoader.

6. Bentuk-Bentuk Lain dari Transformasi (Augmentasi Data):

Menjelaskan berbagai bentuk augmentasi data yang dapat diterapkan.

7. Model 0: TinyVGG tanpa Augmentasi Data:

- Membuat transformasi dan memuat data untuk Model 0:
  Menunjukkan cara membuat transformasi dan memuat data untuk Model 0.
- Membuat kelas model TinyVGG:
  Menjelaskan langkah-langkah membuat kelas model TinyVGG.
- Mencoba forward pass pada satu gambar (untuk menguji model):
  Menunjukkan cara melakukan forward pass pada satu gambar untuk menguji model.
- Menggunakan torchinfo untuk mendapatkan gambaran bentuk yang melewati model:
  Menunjukkan cara menggunakan torchinfo untuk mendapatkan gambaran bentuk yang melewati model.
- Membuat fungsi loop pelatihan dan pengujian:
  Menunjukkan cara membuat fungsi untuk melatih dan menguji Model 0.
- Membuat fungsi train() untuk menggabungkan train_step() dan test_step():
  Menunjukkan cara membuat fungsi train() yang menggabungkan train_step() dan test_step().
- Melatih dan mengevaluasi Model 0:
  Menjelaskan langkah-langkah melatih dan mengevaluasi Model 0.
- Plot kurva kehilangan Model 0:
  Menunjukkan cara membuat plot kurva kehilangan untuk Model 0.

8. Bagaimana Kurva Kehilangan Ideal Seharusnya?

- Bagaimana Mengatasi Overfitting:
  Menjelaskan cara mengatasi overfitting.
- Bagaimana Mengatasi Underfitting:
  Menjelaskan cara mengatasi underfitting.
- Keseimbangan antara Overfitting dan Underfitting:
  Menjelaskan pentingnya mencari keseimbangan antara overfitting dan underfitting.

9. Model 1: TinyVGG dengan Augmentasi Data:

- Membuat transformasi dengan augmentasi data:
  Menunjukkan cara membuat transformasi dengan augmentasi data.
- Membuat Dataset dan DataLoader untuk Model 1:
  Menjelaskan cara membuat Dataset dan DataLoader untuk Model 1.
- Membangun dan melatih Model 1:
  Menunjukkan langkah-langkah membangun dan melatih Model 1.
- Plot kurva kehilangan Model 1:
  Menunjukkan cara membuat plot kurva kehilangan untuk Model 1.

10. Membandingkan Hasil Model:

Menunjukkan cara membandingkan hasil dari dua model.

11. Membuat Prediksi pada Gambar Kustom:

- Memuat gambar kustom dengan PyTorch:
  Menjelaskan cara memuat gambar kustom menggunakan PyTorch.
- Memprediksi gambar kustom dengan model PyTorch yang terlatih:
  Menunjukkan cara memprediksi gambar kustom dengan model PyTorch yang telah dilatih.
- Menggabungkan prediksi gambar kustom: membangun fungsi:
  Menjelaskan langkah-langkah menggabungkan prediksi gambar kustom untuk membangun sebuah fungsi.