Nama : Adifa Syahira

Nim : 1103202067

Kelas : TK 44G4

TECHNICAL REPORT UAS MACHINE LEARNING

A. 00_pytorch_fundamentals.ipynb

Langkah-langkah dari fundamentals pada pytorch dari apa yang telah dicoba atau ditelusuri adalah kita mendapat sebuah pelajaran bahwasannya :

- Pengenalan Tensors dalam PyTorch: Tensors, sebagai struktur data kunci dalam PyTorch, tidak hanya mengacu pada array multidimensi, tetapi juga merupakan fondasi dari semua komputasi dalam framework ini. Mereka secara langsung mirip dengan array NumPy, tetapi kekuatannya terletak pada kemampuan untuk melakukan komputasi pada GPU, yang sangat mempercepat operasi numerik. Ketersediaan operasi diferensiasi otomatis juga membuat tensors sangat vital dalam pembuatan model neural networks. Ketika berinteraksi dengan data dalam bentuk tensor, baik itu data gambar, teks, atau bilangan, penggunaan operasi tensor menjadi inti dari proses pra-pemrosesan, pelatihan, dan inferensi model.
- Operasi pada Tensors: Operasi pada tensors mencakup serangkaian fungsi matematis dasar yang digunakan untuk memanipulasi data. Mulai dari operasi sederhana seperti penjumlahan, pengurangan, hingga operasi yang lebih kompleks seperti perkalian matriks, semua operasi ini memungkinkan pemrosesan dan manipulasi data yang efisien. Dalam konteks machine learning, operasi tensor ini membentuk dasar dari perhitungan yang diperlukan dalam proses pelatihan model.
- Penggunaan Operasi Matrix dan Linear Layer: Penggunaan operasi matriks, seperti torch.matmul, membuka jalan untuk perhitungan tensor yang lebih kompleks, memungkinkan model untuk melakukan operasi matriks besar secara efisien. Linear layer (torch.nn.Linear) memungkinkan representasi yang sangat kuat dalam pembuatan model, karena menyediakan struktur lapisan dengan bobot dan bias yang dapat disesuaikan saat model dilatih. Inilah yang memungkinkan model untuk belajar dari data dan menyesuaikan parameter untuk melakukan tugas tertentu, seperti klasifikasi atau prediksi.
- Transformasi Tensors: Transformasi tensor memungkinkan pengubahan bentuk, dimensi, dan representasi data secara fleksibel. Melalui operasi seperti reshaping, squeezing, unsqueezing, stacking, permuting, dan indexing, kita dapat mengelola dan memanipulasi data dengan lebih efektif, memastikan bahwa tensor memiliki bentuk yang sesuai dengan kebutuhan dan persyaratan model.
- Konversi Antara NumPy Arrays dan Tensors: Kemampuan untuk mengonversi data dari NumPy arrays ke tensors PyTorch, dan sebaliknya, memungkinkan integrasi yang mulus antara

berbagai library dalam lingkungan pengembangan. Hal ini sangat penting karena memungkinkan penggunaan fitur dan kelebihan dari masing-masing framework, memperluas kapabilitas penggunaan data.

- Penggunaan GPU: Kemampuan PyTorch untuk menjalankan operasi tensor di GPU adalah salah satu fitur kunci yang membedakannya. Pemrosesan paralel pada GPU sangat diperlukan untuk komputasi besar dan tugas-tugas komputasi intensif lainnya yang umumnya ditemui dalam machine learning dan komputasi ilmiah.
- Pengelolaan Randomness dan Reproducibility: Pengelolaan randomness dalam pengembangan model sangat penting untuk menjaga reproduktibilitas hasil. Dengan menetapkan seed yang sama, eksperimen dapat direplikasi dengan hasil yang identik, memungkinkan validasi yang kuat terhadap model yang dikembangkan.
- Pemeriksaan Perangkat yang Tersedia: Memahami dan memilih perangkat keras yang tepat untuk melakukan komputasi adalah langkah krusial. Pemilihan antara CPU dan GPU harus didasarkan pada tugas yang akan dijalankan, memastikan penggunaan sumber daya yang efisien dan optimal.
- Perbandingan Tensors: Perbandingan nilai di antara tensors memungkinkan pengecekan validitas dan keakuratan operasi yang dilakukan pada data. Hal ini menjadi langkah penting dalam pengembangan model, karena memvalidasi hasil operasi dan menjamin kebenaran data yang digunakan.
- Mengelola Device Tensors: Kemampuan untuk memindahkan tensors antara perangkat seperti CPU dan GPU memungkinkan penggunaan sumber daya yang efisien. Hal ini sangat penting untuk menangani data yang lebih besar yang memerlukan kecepatan dan kekuatan komputasi yang berbeda di antara perangkat yang tersedia.

B. 01_pytorch_workflow

Mari kita telaah setiap langkah yang dijelaskan dalam kode menggunakan PyTorch dengan lebih mendalam :

- Pemrosesan Data: Tahap pertama dalam pengembangan model adalah memproses dan menyiapkan data. Di sini, kita membuat data sintetis untuk model regresi linear, yang kemudian dibagi menjadi dua subset: data latih dan data uji. Data latih digunakan untuk melatih model, sementara data uji digunakan untuk menguji kinerja model pada data yang tidak pernah dilihat sebelumnya. Pemisahan ini penting untuk mengukur kemampuan generalisasi model pada data baru. Setelah memastikan dataset siap, langkah selanjutnya adalah melakukan penyesuaian data agar sesuai dengan kebutuhan model yang akan digunakan.
- Pemodelan: Langkah kedua adalah membangun arsitektur model itu sendiri. Dalam kasus ini, kita menggunakan pendekatan regresi linear menggunakan PyTorch's nn.Module. Model ini terdiri dari dua parameter utama: bobot (weights) dan bias. Melalui pendekatan modular PyTorch, kita dapat membangun arsitektur model dengan jelas dan sistematis. nn.Module memungkinkan kita untuk mengatur bagian-bagian model dengan lebih terstruktur dan memberikan kontrol penuh terhadap proses komputasi.
- Pelatihan dan Evaluasi Model: Proses pelatihan model dimulai dengan menyajikan data latih ke dalam model, kemudian menghitung loss function untuk mengevaluasi seberapa baik model memprediksi nilai sebenarnya. Setelah itu, model diperbarui menggunakan optimasi seperti Stochastic Gradient Descent (SGD) untuk mengoptimalkan parameter agar lebih mendekati nilai sebenarnya. Evaluasi dilakukan pada data uji untuk mengevaluasi kinerja model pada data yang belum pernah dilihat sebelumnya, membantu kita memahami seberapa baik model akan bekerja di dunia nyata.
- Penyimpanan dan Pemuatan Model: Langkah berikutnya adalah menyimpan model yang telah dilatih ke dalam file agar dapat digunakan kembali tanpa harus melatih ulang. Ini sangat bermanfaat dalam menyimpan hasil latihan model yang memerlukan waktu dan sumber daya komputasi yang besar. Kemampuan untuk memuat model yang disimpan juga penting karena memungkinkan penggunaan model di berbagai lingkungan atau skenario aplikasi tanpa memerlukan proses pelatihan ulang.
- Penyesuaian pada Perangkat yang Tersedia: Penggunaan perangkat keras seperti CPU atau GPU dapat mempengaruhi kinerja dan kecepatan pemrosesan model. PyTorch memungkinkan kita untuk mengatur perpindahan data dan model antara perangkat yang berbeda. Hal ini memungkinkan kita untuk menggunakan GPU (jika tersedia) untuk meningkatkan kecepatan komputasi, terutama pada dataset yang lebih besar.
- Perbandingan dan Validasi Model: Langkah terakhir adalah membandingkan hasil prediksi sebelum dan sesudah penyimpanan model untuk memastikan konsistensi dan validitasnya.

Validasi semacam ini sangat penting karena memastikan bahwa model yang disimpan dapat mempertahankan kinerjanya tanpa perubahan yang signifikan setelah dimuat kembali. Dengan validasi ini, kita memiliki kepercayaan bahwa model yang disimpan dapat diandalkan dalam berbagai situasi penggunaan. Proses ini membantu memastikan bahwa model yang disimpan tetap konsisten dan dapat diandalkan dalam aplikasi praktis.

C. 02_pytorch_classification

Masalah klasifikasi adalah masalah di mana tujuannya adalah untuk mengelompokkan atau mengkategorikan data ke dalam kelas atau kategori yang berbeda berdasarkan fitur atau atribut tertentu. Tujuan utama dari masalah klasifikasi adalah untuk membuat prediksi atau penentuan terhadap kelas atau kategori yang tepat untuk setiap data baru berdasarkan pola atau informasi yang ditemukan dalam data pelatihan. Masalah klasifikasi sering digunakan dalam berbagai bidang seperti ilmu komputer, statistik, dan machine learning untuk berbagai aplikasi seperti pengenalan pola, analisis data, dan prediksi.

Langkah-langkahnya dengan menggunakan PyTorch:

1) Architecture of a classification neural network

Dalam PyTorch, arsitektur jaringan saraf untuk klasifikasi dapat dibuat menggunakan modul torch.nn. Ini terdiri dari lapisan-lapisan seperti torch.nn.Linear untuk lapisan-lapisan linier, fungsi aktivasi seperti torch.nn.ReLU untuk non-linearitas, dan torch.nn.Softmax untuk output layer dalam kasus klasifikasi.

2) Make classification data and get it ready Input and output shapes

Anda perlu mengetahui dimensi dari fitur-fitur input dan jumlah kelas/output yang ingin Anda prediksi. Misalnya, untuk gambar, input shape bisa (batch_size, channels, height, width), sementara output shape mungkin (batch_size, num_classes).

3) Turn data into tensors and create train and test splits

Data dalam PyTorch diwakili sebagai tensors. Anda dapat menggunakan torch.Tensor untuk mengubah data menjadi format tensor. Selanjutnya, menggunakan torch.utils.data.Dataset dan torch.utils.data.DataLoader, Anda bisa membuat train dan test splits dari dataset Anda.

4) Building a model, Setup loss function and optimizer

Di PyTorch, Anda mendefinisikan loss function dengan torch.nn.CrossEntropyLoss() untuk klasifikasi multi-kelas atau torch.nn.BCELoss() untuk klasifikasi biner.

Optimizer seperti torch.optim.SGD atau torch.optim.Adam digunakan untuk mengoptimalkan bobot-bobot jaringan.

5) Make predictions and evaluate the model

   Setelah melatih model, Anda dapat menggunakan model tersebut untuk membuat prediksi pada data baru. Evaluasi model dilakukan dengan metrik-metrik seperti akurasi, presisi, recall, dan F1-score untuk mengukur performa klasifikasi.

6) Improving a model (from a model perspective)

   Ada berbagai cara untuk meningkatkan performa model. Ini bisa meliputi menambahkan lapisan-lapisan baru, mengubah jumlah neuron dalam lapisan, menggunakan teknik regularisasi seperti dropout, atau melakukan fine-tuning pada hyperparameter.

7) The missing piece: non-linearity

   Non-linearitas adalah elemen penting dalam jaringan saraf yang memungkinkan model untuk mempelajari pola-pola yang kompleks dalam data. Tanpa non-linearitas, jaringan saraf hanya akan bisa melakukan transformasi linear.

8) Replicating non-linear activation functions

   Activation functions seperti ReLU (Rectified Linear Unit), Sigmoid, dan Tanh adalah contoh dari fungsi non-linear yang digunakan untuk memperkenalkan non-linearitas dalam jaringan saraf.

9) Putting things together by building a multi-class PyTorch model

   Dengan menggunakan PyTorch, Anda dapat menggabungkan semua konsep ini untuk membangun model klasifikasi multi-kelas. Ini melibatkan pembuatan arsitektur jaringan, mengkompilasi loss function dan optimizer, serta melatih model menggunakan data.

10) More classification evaluation metrics

    Selain metrik-metrik dasar seperti akurasi, terdapat metrik lain seperti area di bawah kurva ROC (AUC-ROC), area di bawah kurva presisi-recall (AUC-PR), dan matriks kebingungan (confusion matrix) yang dapat memberikan pemahaman yang lebih mendalam tentang performa model klasifikasi.

D. 03. PyTorch Computer Vision

Mari kita telaah setiap langkah yang dijelaskan dalam kode menggunakan PyTorch dengan lebih mendalam :

1. **Pengantar Penggunaan Library PyTorch dalam Computer Vision:** Di sini, kamu akan mempelajari konsep dasar penggunaan PyTorch dalam konteks computer vision, termasuk cara memanfaatkan modul-modul khusus seperti **torchvision** untuk mempermudah proses pengolahan data gambar dan pengembangan model.

2. **Pencarian Bantuan:** Kamu akan dipandu untuk menemukan sumber daya yang beragam, termasuk dokumentasi resmi PyTorch, forum pengguna, tutorial online, atau sumber bantuan lainnya yang berguna dalam menjawab pertanyaan dan menyelesaikan masalah saat menggunakan PyTorch.

3. **Persiapan Dataset:** Langkah-langkah konkret untuk mengakses, memuat, dan mempersiapkan dataset yang relevan dengan tujuan tertentu dalam bidang computer vision, mungkin termasuk pemrosesan pra-pemrosesan data seperti normalisasi atau augmentasi.

4. **Visualisasi Data:** Penjelasan mendalam tentang teknik-teknik visualisasi data dalam PyTorch, seperti penggunaan matplotlib atau fungsi bawaan dari PyTorch itu sendiri untuk menampilkan gambar dari dataset yang telah diunduh.

5. **Persiapan DataLoader:** Rincian langkah-langkah yang diperlukan untuk membangun DataLoader, sebuah komponen kunci dalam PyTorch yang memungkinkan kamu untuk mengatur dan memasukkan data ke dalam model secara efisien.

6. **Model Dasar (Model 0):** Pembuatan model awal yang mungkin sederhana untuk digunakan sebagai titik awal eksperimen, membantu dalam memahami alur kerja dasar dari PyTorch dalam membangun model.

7. **Konfigurasi Loss, Optimizer, dan Metrik Evaluasi:** Detail tentang pemilihan fungsi loss, optimizer, serta metrik evaluasi yang tepat untuk tugas yang sedang dihadapi, menjadi landasan yang penting dalam pelatihan dan evaluasi model.

8. **Pelatihan Model pada Data Batches:** Langkah-langkah untuk melatih model menggunakan metode pembelajaran mini-batch, sebuah pendekatan efisien yang digunakan dalam Deep Learning untuk melatih model dengan data yang besar.

9. **Prediksi dan Evaluasi Model 0:** Proses untuk membuat prediksi menggunakan Model 0 yang telah dibuat sebelumnya dan mengevaluasi kinerjanya terhadap data yang telah diproses.

10. **Kode Device Agnostic:** Penyesuaian kode agar dapat berjalan baik pada CPU maupun GPU, memanfaatkan GPU jika tersedia untuk mempercepat proses komputasi.

11. **Peningkatan Model (Model 1):** Pengembangan model yang lebih kompleks dan canggih dengan penggunaan teknik non-linearitas guna meningkatkan performa dari Model 0 yang sebelumnya.

12. **Pembangunan Convolutional Neural Network (CNN) (Model 2):** Pembuatan model menggunakan Convolutional Neural Network, arsitektur yang kuat untuk tugas-tugas visual seperti pengolahan gambar.

13. **Penjelasan tentang Layer Convolutional dan MaxPool:** Detil yang lengkap tentang bagaimana layer Conv2d() dan MaxPool2d() bekerja dalam jaringan saraf konvolusi serta peran mereka dalam memproses informasi visual.

14. **Konfigurasi Loss dan Optimizer untuk Model 2:** Pengaturan khusus fungsi loss dan optimizer yang cocok dengan Model 2 yang menggunakan arsitektur CNN.

15. **Pelatihan dan Pengujian Model 2:** Langkah-langkah untuk melatih Model 2 menggunakan data yang telah dipersiapkan sebelumnya dan mengevaluasi performanya terhadap dataset uji.

16. **Perbandingan Hasil dan Waktu Pelatihan Model:** Analisis komparatif dari hasil dan waktu yang dibutuhkan oleh beberapa model yang telah dibangun sebelumnya, membantu dalam memahami kompromi antara performa dan kecepatan.

17. **Tradeoff Performa dan Kecepatan:** Pemahaman mendalam tentang pentingnya tradeoff antara kualitas performa model dan waktu yang diperlukan untuk melatih atau melakukan inferensi.

18. **Prediksi Acak dan Evaluasi dengan Model Terbaik:** Penggunaan model terbaik yang telah dilatih sebelumnya untuk melakukan prediksi pada data acak, dan evaluasi terhadap performa model pada data tersebut.

19. **Pembuatan Matriks Konfusi untuk Evaluasi Prediksi Lebih Lanjut:** Pembuatan matriks konfusi, sebuah alat evaluasi yang kuat untuk memahami sejauh mana model dapat melakukan prediksi dengan benar terhadap kelas-kelas yang berbeda.

20. **Simpan dan Muat Model Terbaik:** Langkah-langkah untuk menyimpan model yang telah memiliki performa terbaik serta cara untuk memuatnya kembali untuk penggunaan di masa depan.

E. 04. PyTorch Custom Datasets

mari kita ulas dengan lebih detail setiap poin yang Anda sebutkan dalam konteks penggunaan PyTorch :

0. Importing PyTorch and setting up device-agnostic code: Di tahap ini, penggunaan PyTorch dimulai dengan mengimpor perpustakaannya. Ini termasuk mengimpor modul-modul yang diperlukan seperti torch dan torchvision. Selain itu, dalam upaya untuk membuat kode yang agnostik terhadap perangkat (device-agnostic), langkah-langkah seperti menentukan apakah perangkat yang tersedia adalah CPU atau GPU (dengan memanfaatkan torch.device) dilakukan. Pengaturan ini memungkinkan penggunaan perangkat keras yang berbeda-beda tanpa perubahan signifikan dalam kode.

1. Get data: Langkah awal dalam membangun model machine learning adalah memperoleh data. Di sini, mungkin akan dilakukan proses pengunduhan atau pemuatan data yang akan digunakan dalam pelatihan atau pengujian model. Dataset dapat berasal dari berbagai sumber, bisa berupa dataset bawaan PyTorch atau dataset yang diimpor dari sumber eksternal.

2. Become one with the data (data preparation): Persiapan data adalah tahap kunci. Ini melibatkan pemahaman mendalam terhadap struktur dan karakteristik data yang telah diperoleh. Proses ini mencakup pembersihan data, penyesuaian skala, penanganan nilai yang hilang, atau pemformatan data agar sesuai dengan kebutuhan model yang akan dibangun. Mempersiapkan data dengan benar membantu mencegah masalah dan meningkatkan kualitas hasil akhir dari model.

2.1 Visualize an image: Visualisasi gambar dari dataset membantu pemahaman visual terhadap data. Dengan menggunakan fungsi-fungsi dari modul visualisasi PyTorch atau matplotlib, contohnya, kita bisa menampilkan sebuah gambar dari dataset. Langkah ini memungkinkan kita untuk melihat bagaimana data sebenarnya terstruktur dan memvalidasi apakah proses pengunduhan atau pemuatan data telah berjalan dengan baik.

3. Transforming data: Proses transformasi data melibatkan perubahan data mentah menjadi bentuk yang lebih sesuai untuk pembelajaran mesin. Ini bisa termasuk normalisasi, rotasi, pemangkasan, atau perubahan lainnya yang diperlukan untuk mempersiapkan data sebelum dimasukkan ke dalam model. Dalam konteks pengolahan gambar, transformasi data dapat mengubah ukuran gambar, meningkatkan kontras, atau melakukan augmentasi data.

4. Option 1: Loading Image Data Using ImageFolder: Penggunaan ImageFolder dari PyTorch memungkinkan penggunaan struktur folder yang sesuai dengan kelas atau label gambar. Dengan mengorganisir gambar ke dalam folder berdasarkan kelasnya, kita dapat memanfaatkan kemudahan PyTorch dalam memuat dataset dengan memanggil ImageFolder dan menentukan transformasi apa yang diterapkan pada setiap gambar.

5. Option 2: Loading Image Data with a Custom Dataset: Memuat data gambar dengan menggunakan dataset kustom memberikan kontrol yang lebih besar terhadap cara data dimuat dan diproses. Dalam langkah ini, pengguna dapat membuat dataset sendiri yang mengikuti struktur yang dibutuhkan oleh model yang akan digunakan. Ini memungkinkan fleksibilitas dan pengolahan yang lebih spesifik sesuai kebutuhan.

6. Other forms of transforms (data augmentation): Augmentasi data merupakan teknik yang penting dalam pengolahan gambar untuk meningkatkan variasi data pelatihan dengan membuat variasi baru dari gambar yang ada, seperti rotasi, pemangkasan, atau pergeseran. Dengan menggunakan transformasi data yang berbeda, kita dapat meningkatkan jumlah sampel pelatihan dan meningkatkan kegeneralisasian model terhadap data baru.

7. Model 0: TinyVGG without data augmentation: Pada tahap ini, kita akan membangun model TinyVGG tanpa menggunakan augmentasi data. Model ini mungkin merupakan versi ringan dari arsitektur VGG (Visual Geometry Group) yang telah terkenal. Pembuatan model ini merupakan langkah awal untuk melihat kinerja dasar dari arsitektur yang dipilih sebelum mempertimbangkan augmentasi data.

8. What should an ideal loss curve look like?: Kurva kerugian (loss curve) yang ideal menunjukkan penurunan yang stabil dari nilai kerugian selama pelatihan model. Pada grafik, kita ingin melihat nilai kerugian berkurang seiring dengan berjalannya iterasi atau epoch, menandakan bahwa model semakin mampu menggeneralisasi data yang belum pernah dilihat sebelumnya.

9. Model 1: TinyVGG with Data Augmentation: Setelah memahami kinerja dasar model, langkah selanjutnya adalah membangun versi model yang sama dengan menggunakan augmentasi data. Dengan menerapkan teknik augmentasi data pada data pelatihan, tujuannya adalah meningkatkan kinerja dan generalisasi model terhadap data baru yang mungkin berbeda dengan data pelatihan.

10. Compare model results: Langkah ini melibatkan perbandingan hasil atau kinerja antara dua atau lebih model yang telah dilatih. Ini dapat dilakukan dengan melihat metrik evaluasi seperti akurasi, presisi, recall, atau F1-score dari setiap model. Dengan membandingkan kinerja, kita dapat menentukan model mana yang lebih baik atau lebih cocok untuk kasus yang sedang dihadapi.

11. Make a prediction on a custom image: Langkah terakhir adalah menguji model yang telah dilatih pada gambar kustom yang tidak termasuk dalam data pelatihan. Dengan menggunakan model yang telah terlatih, kita dapat melakukan prediksi atau klasifikasi terhadap gambar baru tersebut untuk melihat bagaimana model menangani data yang belum pernah dilihat sebelumnya.

Setiap langkah ini merupakan bagian penting dalam proses pengembangan model menggunakan PyTorch. Dengan memahami setiap tahap secara mendalam, Anda dapat membangun, melatih, dan mengevaluasi model machine learning dengan lebih baik.

F. 05. PyTorch Going Modular

Mari kita bahas lebih mendalam setiap langkah yang telah Anda lakukan dalam konteks pembelajaran mesin menggunakan PyTorch :

1. Mengunduh dan Mempersiapkan Dataset:

Anda mengambil langkah awal dengan mendownload dataset "pizza_steak_sushi" yang akan digunakan sebagai sumber data untuk melatih dan menguji model. Setelah berhasil diunduh, Anda memastikan struktur folder yang diperlukan untuk data tersebut ada dengan membuat direktori data latih dan data uji jika belum ada. Proses ini penting karena memastikan data terstruktur dengan benar sehingga dapat dimuat dengan tepat saat digunakan dalam model. Setelah itu, Anda mengekstrak file zip yang berisi dataset dan menyiapkan struktur folder yang sesuai agar dapat dengan mudah dimuat menggunakan ImageFolder dari PyTorch.

2. Transformasi Data:

Dalam proses ini, transformasi data dilakukan menggunakan torchvision.transforms. Anda telah mengatur transformasi yang diperlukan untuk mempersiapkan gambar agar sesuai dengan kebutuhan model. Dengan melakukan resize gambar ke ukuran tertentu dan mengonversi gambar menjadi tensor, Anda memastikan konsistensi format dan struktur data yang diperlukan untuk memproses gambar dalam model. Transformasi data adalah langkah penting untuk memastikan konsistensi dan kecocokan antara format data asli dengan format yang diterima oleh model.

3. Membuat Dataset:

Penggunaan datasets.ImageFolder dari PyTorch adalah langkah selanjutnya yang penting dalam membangun dataset. Dengan menggunakan struktur folder yang telah disiapkan sebelumnya, Anda membuat dataset dari gambar-gambar tersebut. Ini memungkinkan PyTorch untuk dengan mudah memuat dataset dengan memanggil ImageFolder dan menerapkan transformasi yang telah ditentukan pada setiap gambar. Ini adalah langkah krusial karena menyediakan akses yang mudah dan terstruktur ke data, yang sangat penting dalam pelatihan model.

4. Meload Data dalam DataLoader:

Setelah dataset dibuat, langkah berikutnya adalah mengorganisir data ke dalam DataLoader. Dalam hal ini, Anda menggunakan torch.utils.data.DataLoader untuk membuat data loader untuk data latih dan data uji. Data loader ini penting karena memfasilitasi pengelolaan dan pemuatan data secara efisien saat proses pelatihan dan evaluasi model. Dengan menentukan berapa banyak sampel per batch yang akan digunakan, jumlah pekerja yang digunakan untuk memuat data, dan apakah data perlu diacak atau tidak, Anda mengatur cara bagaimana data akan diproses dalam model.

5. Membangun Arsitektur Model:

Dalam langkah ini, Anda mendefinisikan arsitektur model, dalam hal ini TinyVGG, sebagai sebuah kelas dalam PyTorch. Dengan mendefinisikan lapisan-lapisan konvolusi, fungsi aktivasi, dan lapisan linear, Anda menentukan bagaimana data akan mengalir melalui model. Langkah ini menciptakan kerangka kerja struktural yang akan digunakan untuk mempelajari pola dari data yang ada. Pengaturan ini adalah fondasi dari model yang akan digunakan dalam proses pembelajaran selanjutnya.

6. Melatih dan Menguji Model:

Pada tahap ini, model mulai dilatih dan diuji. Proses pelatihan melibatkan perhitungan loss, optimisasi, dan evaluasi terhadap data latih untuk menyesuaikan parameter-model agar cocok dengan data. Pengujian dilakukan untuk mengevaluasi kinerja model terhadap data uji yang belum pernah dilihat sebelumnya. Pengukuran loss dan akurasi menjadi kunci dalam memahami sejauh mana model dapat memahami data dan melakukan prediksi yang akurat. Proses ini adalah inti dari pembelajaran mesin karena di sinilah model belajar dari data dan meningkatkan kinerjanya seiring berjalannya waktu.

7. Mengukur Kinerja Model:

Anda mengukur kinerja model dengan mencatat loss dan akurasi pada setiap epoch selama proses pelatihan dan pengujian. Mengukur kinerja model berarti mengevaluasi seberapa baik model bekerja. Ini dilakukan dengan mencatat tingkat kesalahan (loss) dan tingkat keakuratan (akurasi) pada setiap tahap proses pelatihan dan pengujian model.

8. Menyimpan Model:

Setelah pelatihan selesai, Anda menyimpan model yang telah dilatih dalam format .pth untuk penggunaan atau pengujian lebih lanjut.

Ini merupakan langkah-langkah esensial dalam penggunaan PyTorch untuk mempersiapkan data, membangun model, melatihnya, dan menyimpannya untuk digunakan nanti. Dengan langkah-langkah ini, Anda telah membuat fondasi yang kuat dalam mempelajari dan mengembangkan model machine learning dengan PyTorch.

G. 06. PyTorch Transfer Learning

- Penyiapan Data:

  Pada tahap ini, terdapat proses yang komprehensif untuk mempersiapkan dataset gambar yang akan digunakan. Ini mencakup langkah-langkah seperti unduhan dataset yang bersangkutan, dalam kasus ini, gambar-gambar yang berkaitan dengan kategori pizza, steak, dan sushi. Setelah dataset diunduh, langkah berikutnya adalah memastikan bahwa data tersebut siap digunakan dalam pelatihan model. Untuk ini, dilakukan transformasi data, termasuk operasi seperti Resize, ToTensor, dan Normalize. Misalnya, Resize dilakukan agar semua gambar memiliki ukuran yang konsisten, ToTensor untuk mengubah gambar menjadi tensor yang bisa diproses oleh model, dan Normalize untuk standarisasi nilai piksel agar model bisa belajar dengan lebih baik. Proses transformasi ini mempersiapkan data latih dan data uji sehingga siap untuk digunakan dalam pembelajaran.

- Penggunaan Pretrained Model:

  Penggunaan model pre-trained, seperti EfficientNet dalam framework torchvision, adalah langkah yang cerdas dalam pembuatan model jaringan saraf. Dalam konteks ini, model pre-trained digunakan untuk mendapatkan keuntungan dari pengetahuan yang telah dipelajari model tersebut dari dataset yang sangat besar (biasanya ImageNet). Langkah selanjutnya yang sangat penting adalah mengunci beberapa bagian dari model, khususnya bagian ekstraktor fitur (feature extractor). Ini dilakukan dengan mengatur requires_grad menjadi False pada parameter-parameter tertentu, sehingga bagian-bagian tersebut tidak akan terkena pembelajaran ulang saat pelatihan model dilakukan. Hal ini membantu menjaga informasi yang telah dipelajari oleh model pre-trained dan mencegahnya untuk terlalu terpengaruh oleh dataset spesifik yang lebih kecil yang digunakan untuk pelatihan.

- Pelatihan Model:

  Proses pelatihan model melibatkan beberapa langkah kunci. Pertama, pemilihan fungsi loss yang tepat untuk tugas klasifikasi, dalam hal ini menggunakan CrossEntropyLoss yang cocok untuk klasifikasi multi-kelas. Selanjutnya, optimisasi model dilakukan melalui penggunaan optimizer seperti Adam yang membantu dalam menyesuaikan bobot-bobot model berdasarkan nilai gradien dari loss function. Proses pelatihan diimplementasikan pada dataset yang telah dipersiapkan sebelumnya, dan hasil performa model dievaluasi pada data uji. Melalui beberapa iterasi (epoch), model secara bertahap belajar untuk memetakan gambar-gambar tersebut ke dalam kategori yang tepat. Langkah terakhir dari pelatihan adalah plot kurva loss, yang memberikan pandangan visual tentang bagaimana performa model berkembang selama proses pembelajaran.

- Prediksi pada Gambar:

Bagian ini membahas proses prediksi pada gambar-gambar dengan menggunakan model yang telah dilatih. Ini melibatkan beberapa langkah, termasuk pengubahan gambar menjadi format yang bisa diproses oleh model (seperti resize dan normalisasi), dilanjutkan dengan proses inferensi menggunakan model yang sudah dilatih untuk menghasilkan prediksi kelas. Dalam kasus ini, model tersebut dipergunakan untuk memprediksi kelas gambar-gambar di dalam dataset uji dan juga pada gambar-gambar di luar dataset yang diunduh dari internet. Hasil prediksi kemudian divisualisasikan dengan menampilkan gambar bersama dengan label kelas yang diprediksi dan probabilitas prediksi yang diperoleh dari model.

H. 07. PyTorch Experiment Tracking

Setiap langkah dalam pelacakan eksperimen dalam pengembangan model Machine Learning menggunakan PyTorch :

Persiapan (Getting setup): Membuat fungsi bantu untuk menetapkan biji acak adalah langkah awal yang penting dalam memastikan konsistensi dan reproduktibilitas eksperimen. Dalam dunia Machine Learning, kekonsistenan dalam pengaturan biji acak (seeds) memungkinkan hasil yang dapat direproduksi dari satu percobaan ke percobaan lainnya. Hal ini penting karena memungkinkan peneliti untuk menguji dan memvalidasi model mereka dengan kondisi yang sama secara konsisten, memastikan bahwa hasil yang diperoleh adalah akurat dan dapat dipercaya. Dengan menetapkan biji acak secara tepat, para praktisi ML dapat memperoleh wawasan yang lebih mendalam tentang cara kerja model mereka dan memastikan keandalan serta stabilitas performa model di berbagai percobaan.

Mendapatkan Data (Get data): Langkah ini merupakan langkah awal dalam siklus pembangunan model. Pengumpulan data yang relevan, bersih, dan representatif sangat krusial dalam memastikan model dapat belajar dari pola yang signifikan dan mampu melakukan generalisasi pada data baru. Identifikasi sumber data yang tepat, pembersihan data, serta transformasi data merupakan bagian integral dari proses ini. Menyediakan dataset yang baik adalah kunci keberhasilan dalam pengembangan model yang andal dan akurat.

Membuat Dataset dan DataLoader (Create Datasets and DataLoaders): Setelah data terkumpul, langkah selanjutnya adalah menyusun dataset dan DataLoaders. Mempersiapkan dataset dengan benar adalah langkah penting dalam mengatur data untuk pelatihan model. Pembuatan DataLoaders yang efisien mempermudah proses pelatihan dengan memuat dan mengatur data secara otomatis ke dalam model. Ini menjadi sangat penting ketika bekerja dengan dataset yang besar, karena memungkinkan proses pelatihan untuk berjalan lebih efisien dan lebih mudah dielusidasi saat melakukan debug atau peningkatan performa.

Mendapatkan Model Pretrained, Membekukan Layer Dasar, dan Mengubah Classifier Head (Getting a pretrained model, freezing the base layers and changing the classifier head): Seringkali, menggunakan model yang sudah dilatih sebelumnya (pretrained) bisa menjadi keuntungan besar dalam menghemat waktu dan sumber daya. Melatih model dari awal membutuhkan waktu dan komputasi yang besar. Dalam banyak kasus, Anda dapat menggunakan model yang sudah dilatih pada dataset besar dan kemudian menyesuaikannya dengan tugas khusus yang Anda kerjakan. Mengganti classifier head atau bagian tertentu dari model, serta membekukan layer dasar untuk mencegah pembelajaran ulang dari awal, dapat membantu model beradaptasi dengan tugas spesifik tanpa kehilangan pengetahuan yang sudah terakumulasi pada bagian yang lain.

Melatih Model dan Melacak Hasil (Train model and track results): Proses pelatihan model merupakan langkah kunci dalam pengembangan model ML. Selama pelatihan, melacak hasil adalah penting untuk memahami perkembangan model dan mengevaluasi kinerjanya. Menggunakan alat seperti SummaryWriter() dari PyTorch memungkinkan pencatatan metrik kunci, seperti akurasi, loss function, atau gradient, sehingga memudahkan analisis terhadap performa model selama proses pelatihan.

Melihat Hasil Model pada TensorBoard (View our model's results in TensorBoard): TensorBoard adalah alat visualisasi yang kuat yang digunakan untuk menganalisis dan memvisualisasikan performa model. Dengan memvisualisasikan hasil-hasil eksperimen, peneliti dapat memperoleh wawasan yang lebih dalam tentang bagaimana model bereaksi terhadap data latihan dan validasi, serta mengidentifikasi area yang memerlukan perbaikan atau perubahan.

Membuat Fungsi Bantu untuk Membangun Instance SummaryWriter() (Create a helper function to build SummaryWriter() instances): Membuat fungsi bantu untuk membangun objek SummaryWriter() membantu dalam mempermudah proses pencatatan hasil eksperimen. Dengan demikian, peneliti atau praktisi dapat dengan mudah membuat instance SummaryWriter() yang sesuai dengan percobaan yang sedang dilakukan, dan melakukan pencatatan metrik yang relevan dengan mudah dan konsisten.

Menyiapkan Serangkaian Eksperimen Modelling (Setting up a series of modelling experiments): Mempersiapkan serangkaian eksperimen dengan variasi parameter, arsitektur model, atau konfigurasi lainnya memungkinkan pembandingan yang mendalam antara berbagai pendekatan. Dengan melakukan ini, peneliti atau praktisi ML dapat mengeksplorasi berbagai opsi untuk mendapatkan pemahaman yang lebih baik tentang model mana yang paling efektif atau cocok untuk masalah yang dihadapi.

Melihat Eksperimen pada TensorBoard (View experiments in TensorBoard): TensorBoard memfasilitasi perbandingan dan analisis yang lebih mendalam antara berbagai hasil eksperimen. Dengan menyajikan hasil eksperimen dalam tata letak visual yang intuitif, TensorBoard memungkinkan pembandingan metrik kunci dari berbagai model atau konfigurasi, memudahkan identifikasi model terbaik atau area yang memerlukan peningkatan.

Memuat Model Terbaik dan Melakukan Prediksi (Load in the best model and make predictions with it): Setelah mengevaluasi berbagai eksperimen, langkah selanjutnya adalah memilih model terbaik berdasarkan metrik performa yang telah ditetapkan. Memuat model dengan performa terbaik dan melakukan prediksi pada data baru memungkinkan peneliti untuk melihat bagaimana model akan berperilaku pada situasi praktis dan memastikan bahwa model yang dihasilkan dapat memberikan hasil yang memuaskan di luar lingkungan eksperimen.

Sebagai hasil dari langkah-langkah tersebut, peneliti atau praktisi ML dapat memperoleh wawasan yang lebih dalam tentang kinerja model, mengidentifikasi area untuk per baikan atau penyesuaian, dan mengembangkan model yang lebih efektif dan andal untuk tugas tertentu dalam Machine Learning. Kesimpulan utama yang dapat ditarik adalah bahwa dengan melakukan pelacakan eksperimen yang cermat dan sistematis, proses pengembangan model dapat ditingkatkan secara signifikan.

I. 08. PyTorch Paper Replicating

Materi yang dapat diambil setelah apa yang telah dikerjakan adalah :

1. **Mendapatkan Data:** Langkah pertama dalam proses pengembangan model machine learning adalah memperoleh data yang akan digunakan. Proses ini bisa melibatkan berbagai sumber, seperti dataset publik, data internal, atau pengumpulan data dari sumber lain. Dalam konteks penggunaan PyTorch, pengambilan data umumnya melibatkan penggunaan fungsi-fungsi bawaan PyTorch atau pustaka pendukung lainnya untuk membaca dan memuat data ke dalam format yang sesuai dengan kebutuhan model. Data ini kemudian akan dibagi menjadi subset untuk pelatihan, validasi, dan pengujian.

2. **Membuat Dataset dan DataLoader:** Setelah data diperoleh, langkah selanjutnya adalah mempersiapkan data tersebut untuk digunakan dalam model. Dalam PyTorch, hal ini dilakukan dengan membuat kelas-kelas Dataset yang mengatur akses ke data dan transformasi yang diperlukan, seperti pengubahan ukuran gambar atau normalisasi. Setelah Dataset dibuat, DataLoader digunakan untuk memuat data secara efisien ke dalam model saat pelatihan. DataLoader memungkinkan pengaturan berbagai parameter, seperti ukuran batch dan pengacakan data, untuk meningkatkan efisiensi dan konsistensi dalam proses pelatihan.

3. **Replikasi Paper ViT: Gambaran Umum:** Proses replikasi sebuah paper, dalam hal ini Vision Transformer (ViT), dimulai dengan memahami secara menyeluruh apa yang ingin direplikasi dari paper tersebut. Ini melibatkan pembacaan ulang terhadap paper, memahami arsitektur model, bagaimana data diproses, dan komponen-komponen kunci yang membuat model tersebut unik. Dengan pemahaman yang kokoh terhadap tujuan replikasi, langkah-langkah berikutnya dapat diarahkan dengan lebih tepat untuk mereproduksi model tersebut.

4. **Persamaan 1: Membagi Data menjadi Patches dan Membuat Embedding:** Langkah ini terfokus pada pemahaman terhadap persamaan matematis yang menggambarkan bagaimana data gambar dibagi menjadi "patch" yang kemudian diubah menjadi embedding untuk kelas, posisi, dan patch itu sendiri. Ini melibatkan proses pemetaan data ke dalam representasi yang dapat digunakan oleh model, yang menjadi fondasi utama dari Vision Transformer.

5. **Persamaan 2: Multi-Head Attention (MSA):** Persamaan Multi-Head Attention menjelaskan bagaimana mekanisme penting ini diimplementasikan dalam model ViT. Ini adalah bagian yang sangat krusial dalam arsitektur Transformer yang memungkinkan model untuk fokus pada bagian-bagian penting dari inputnya.

6. **Persamaan 3: Multilayer Perceptron (MLP):** Komponen berikutnya dari model ViT adalah Multilayer Perceptron (MLP). Persamaan ini menjelaskan bagaimana MLP digunakan dalam

konteks spesifik ViT, yang umumnya berperan dalam memproses informasi lokal dari masing-masing patch.

7. **Membuat Encoder Transformer:** Proses pembuatan Encoder Transformer adalah langkah penting dalam membangun arsitektur lengkap dari ViT. Ini melibatkan pengaturan berbagai blok Transformer yang akan menyusun struktur utama dari model.

8. **Menggabungkan Semua Komponen untuk Membuat ViT:** Tahap ini menerapkan integrasi dari semua bagian-bagian yang telah dibuat sebelumnya. Hal ini melibatkan penggabungan komponen-komponen yang sudah terdefinisi sebelumnya, seperti encoder Transformer, Multi-Head Attention, dan MLP, untuk membentuk arsitektur lengkap dari Vision Transformer.

9. **Menyiapkan Kode Pelatihan untuk Model ViT:** Setelah arsitektur ViT dibangun, langkah selanjutnya adalah menyiapkan kode pelatihan. Ini melibatkan penyesuaian parameter pelatihan, pilihan optimizer, dan pengaturan lainnya untuk melatih model ViT.

10. **Menggunakan ViT yang Sudah Dilatih dari torchvision.models pada Dataset yang Sama:** Sebuah pendekatan lain dalam memanfaatkan ViT adalah dengan menggunakan model yang sudah dilatih sebelumnya. PyTorch menyediakan pustaka torchvision.models yang mencakup berbagai arsitektur model termasuk ViT, memungkinkan pengguna untuk mengakses model-model yang sudah dilatih dan menggunakan mereka pada dataset yang sama.

11. **Membuat Prediksi pada Gambar Kustom:** Langkah terakhir adalah menggunakan model ViT yang sudah dilatih untuk membuat prediksi pada gambar-gambar yang tidak terdapat dalam dataset pelatihan. Ini melibatkan penggunaan model untuk mengklasifikasikan atau melakukan tugas lainnya pada data baru untuk mengevaluasi kinerja model.

Dengan memahami setiap langkah secara mendalam, pengguna dapat mengimplementasikan Vision Transformer (ViT) dengan lebih baik menggunakan PyTorch dan memperoleh pemahaman yang lebih kuat tentang kerja dan kekuatan dari model tersebut.

J. 09. PyTorch Model Deployment

Ini adalah beberapa poin penting dalam pembangunan model machine learning yang memerlukan pengumpulan data yang tepat, eksperimen dalam penyusunan model yang optimal, serta pembuatan dan penggunaan ekstraktor fitur untuk memproses informasi yang diperlukan dalam prediksi. Langkah-langkah ini merupakan fondasi penting dalam membangun model yang akurat dan efisien:

1. **Pengumpulan Data:** Tahap awal dalam pengembangan model machine learning adalah pengumpulan data yang relevan. Ini mencakup identifikasi sumber data yang valid, proses pengumpulan, dan seringkali membersihkan data dari noise atau kekacauan. Setelah data terkumpul, dilakukan proses pemisahan antara data latih, validasi, dan uji. Langkah ini krusial karena kualitas data yang digunakan sangat mempengaruhi performa akhir model. Dataset yang baik dan representatif akan membantu model belajar pola yang benar dan membuat prediksi yang lebih akurat.

2. **Eksperimen Penyusunan Model FoodVision Mini:** Pada tahap ini, dilakukan eksperimen untuk membuat model klasifikasi gambar yang disebut FoodVision Mini. Langkah awal melibatkan pembuatan ekstraktor fitur menggunakan model EffNetB2 dan ViT. Proses ini mungkin melibatkan percobaan dengan berbagai arsitektur model dan teknik untuk menentukan kombinasi terbaik yang menghasilkan performa yang optimal.

3. **Penggunaan Ekstraktor Fitur:** Langkah penting lainnya adalah implementasi ekstraktor fitur menggunakan model EffNetB2 dan ViT. Ekstraktor fitur ini bertanggung jawab dalam mengekstrak representasi penting dari gambar yang akan digunakan untuk membuat prediksi. Proses ini termasuk menginisialisasi model, ekstraksi fitur dari gambar, dan penyimpanan informasi yang diekstraksi. Kualitas ekstraksi fitur ini sangat mempengaruhi kemampuan model dalam memahami dan menginterpretasi informasi yang ada pada gambar.

4. **Pembuatan Prediksi dengan Model Terlatih dan Pengukuran Waktu:** Setelah model dilatih, langkah selanjutnya adalah menggunakannya untuk membuat prediksi pada data yang belum dilihat sebelumnya. Proses ini melibatkan menguji model pada dataset validasi atau uji, mencatat waktu yang diperlukan untuk membuat prediksi, dan mengevaluasi performa model berdasarkan metrik yang relevan seperti akurasi, presisi, dan recall.

5. **Perbandingan Hasil Model:** Setelah prediksi dibuat, dilakukan perbandingan antara hasil dari berbagai model yang telah dikembangkan. Ini melibatkan analisis terhadap metrik performa masing-masing model seperti akurasi, serta membandingkan waktu yang dibutuhkan dan ukuran model.

6. **Membuat Demo Gradio FoodVision Mini:** Tahap ini merupakan langkah praktis untuk menampilkan model kepada pengguna. Proses ini termasuk dalam membuat demo

menggunakan Gradio, yang memungkinkan pengguna untuk melakukan inferensi pada model dengan antarmuka yang user-friendly.

7. **Mengubah Demo Gradio FoodVision Mini Menjadi Aplikasi Yang Dapat Dideploy:** Setelah membuat demo, langkah selanjutnya adalah mengubahnya menjadi aplikasi yang dapat di-deploy. Hal ini dapat dilakukan dengan menggunakan berbagai platform deployment seperti AWS, Heroku, atau HuggingFace Spaces.

8. **Deployment Aplikasi FoodVision Mini ke HuggingFace Spaces:** Tahap ini mengimplikasikan proses mengambil model yang sudah dibuat sebelumnya dan menjalankannya pada platform HuggingFace Spaces sehingga model dapat diakses secara online dan digunakan oleh pengguna lain.

9. **Pembuatan Model FoodVision Big:** Langkah ini mencakup pengembangan model yang lebih besar dari FoodVision Mini. Ini bisa termasuk peningkatan jumlah data yang digunakan, kompleksitas model, atau fitur-fitur tambahan yang diperkenalkan.

10. **Mengubah Model FoodVision Big Menjadi Aplikasi Yang Dapat Dideploy:** Sama seperti langkah sebelumnya, setelah pembuatan model FoodVision Big, langkah berikutnya adalah mengubahnya menjadi aplikasi yang dapat di-deploy, memungkinkan akses bagi pengguna untuk menggunakan model tersebut.

Dalam keseluruhan proses ini, setiap langkah merupakan bagian penting dari siklus pengembangan model machine learning yang komprehensif, dimulai dari pengumpulan data hingga deployment aplikasi yang dapat digunakan oleh pengguna akhir.

.