PDF REPORT CHAPTER PYTORCH

Oleh :

Loaeza Septavial/1103204003

PRODI S1 TEKNIK KOMPUTER
FAKULTAS TEKNIK ELEKTRO
UNIVERSITAS TELKOM BANDUNG

2023

00. PyTorch Fundamentals

PyTorch adalah suatu framework atau kerangka kerja (framework) sumber terbuka yang digunakan untuk machine learning dan deep learning. Dengan menggunakan Python, PyTorch memungkinkan pengguna untuk memanipulasi dan memproses data serta menulis algoritma machine learning.

PyTorch dapat digunakan dalam berbagai industri dan penelitian. Banyak perusahaan teknologi terkemuka seperti Meta (Facebook), Tesla, dan Microsoft, serta perusahaan penelitian kecerdasan buatan seperti OpenAI, menggunakan PyTorch untuk penelitian dan mengintegrasikan machine learning ke dalam produk mereka. Sebagai contoh, Andrej Karpathy, kepala AI di Tesla, telah memberikan beberapa presentasi tentang penggunaan PyTorch dalam mengembangkan model computer vision untuk mobil otonom mereka.

PyTorch digunakan tidak hanya di industri teknologi, tetapi juga dalam industri lain seperti pertanian, di mana PyTorch digunakan untuk mengaktifkan computer vision pada traktor.

Keunggulan PyTorch terletak pada kemudahan penggunaannya oleh peneliti machine learning. PyTorch juga menyediakan dukungan untuk akselerasi GPU secara otomatis, meningkatkan kecepatan eksekusi kode. Hal ini memungkinkan pengguna untuk fokus pada manipulasi data dan penulisan algoritma, sementara PyTorch secara otomatis mengoptimalkan eksekusinya.

Modul ini mencakup dasar-dasar PyTorch, yang dimulai dengan pengenalan tentang "tensor", yaitu blok dasar untuk machine learning dan deep learning. Modul ini juga mencakup pembuatan tensor, pengambilan informasi dari tensor, manipulasi tensor, penanganan bentuk tensor, indeks tensor, dan penggunaan tensor PyTorch bersama NumPy. Selain itu, topik lainnya termasuk reproduktibilitas, menjalankan tensor pada GPU untuk meningkatkan kecepatan komputasi, dan cara mendapatkan bantuan melalui dokumentasi PyTorch dan forum pengembang.

01. PyTorch Workflow Fundamentals

Dalam modul ini, kita memulai dengan pemahaman dasar mengenai machine learning dan deep learning. Ide dasarnya adalah menggunakan data masa lalu untuk membangun algoritma, seperti jaringan saraf (neural network), guna menemukan pola di dalamnya. Pola ini kemudian digunakan untuk memprediksi kejadian di masa depan. Kita mulai dengan memahami langkah-langkah dasar dalam sebuah proyek machine learning dengan PyTorch.

Langkah pertama adalah persiapan data, di mana kita menciptakan data sederhana dalam bentuk garis lurus menggunakan regresi linear. Selanjutnya, kita membagi data tersebut menjadi dua set: data latih dan data uji. Data latih digunakan untuk melatih model, sedangkan data uji digunakan untuk menguji sejauh mana model mampu menggeneralisasi pola yang telah dipelajari.

Setelah itu, kita membangun model menggunakan PyTorch. Model yang kita gunakan adalah model regresi linear sederhana, yang mencoba untuk menemukan hubungan antara fitur (X) dan label (y). Selain itu, kita memilih fungsi kerugian (loss function) dan pengoptimal (optimizer) untuk proses pelatihan.

Proses pelatihan model melibatkan langkah-langkah forward pass, perhitungan loss, pembaharuan gradien, dan optimisasi. Dalam contoh ini, kita menggunakan mean absolute error (MAE) sebagai fungsi kerugian dan stochastic gradient descent (SGD) sebagai optimizer.

Setelah melatih model, kita melakukan evaluasi pada data uji dan membuat prediksi. Proses evaluasi ini melibatkan perhitungan loss dan dapat melibatkan juga pengukuran metrik lainnya, tergantung pada jenis masalah yang sedang dihadapi.

Terakhir, kita membahas cara menyimpan dan memuat model yang telah dilatih menggunakan PyTorch. Pada akhirnya, kita menyusun semua langkah di atas menjadi satu kesatuan yang utuh.

Selama proses ini, kita juga menekankan pentingnya visualisasi data untuk memahami pola-pola yang ada. Semua langkah dijelaskan secara rinci dengan kode dalam bahasa Python menggunakan PyTorch.

02. PyTorch Neural Network Classification

Dalam konteks pembelajaran mesin, masalah klasifikasi adalah tugas memprediksi apakah suatu data termasuk ke dalam satu kategori atau kategori lainnya. Ada beberapa jenis masalah klasifikasi, termasuk:

1. Binary Classification: Memiliki dua opsi target, seperti ya atau tidak. Contohnya adalah memprediksi apakah seseorang memiliki penyakit jantung berdasarkan parameter kesehatan mereka.

2. Multi-class Classification: Target dapat menjadi salah satu dari lebih dari dua opsi. Contohnya adalah mengidentifikasi apakah suatu gambar adalah makanan, manusia, atau anjing.

3. Multi-label Classification: Target dapat memiliki lebih dari satu opsi. Misalnya, memprediksi kategori apa yang harus diberikan pada artikel Wikipedia, seperti matematika, sains, atau filsafat.

Masalah klasifikasi, bersama dengan regresi (memprediksi suatu nilai), adalah salah satu tipe masalah pembelajaran mesin yang paling umum.

Dalam notebook ini, disajikan langkah-langkah untuk mengimplementasikan workflow PyTorch pada masalah klasifikasi. Langkah-langkahnya mencakup:

- Arsitektur Jaringan Saraf untuk Klasifikasi: Menjelaskan struktur umum dari jaringan saraf untuk masalah klasifikasi, termasuk lapisan input, lapisan tersembunyi, dan lapisan output.

- Persiapan Data untuk Klasifikasi Binary: Membuat dataset sederhana untuk klasifikasi binary menggunakan metode `make_circles` dari Scikit-Learn.

- Membangun Model Klasifikasi PyTorch: Membuat model untuk mempelajari pola dalam data, memilih fungsi kerugian, pengoptimal, dan membangun loop pelatihan khusus untuk klasifikasi.

- Pelatihan Model dan Evaluasi: Melatih model pada data dan mengevaluasi kinerjanya pada data uji.

- Meningkatkan Model: Menyelidiki cara-cara untuk meningkatkan model, termasuk penambahan lapisan, penyesuaian fungsi aktivasi, dan perubahan hyperparameter lainnya.

- Non-linearity: Menyajikan konsep non-linieritas dalam model dan bagaimana menambahkannya menggunakan fungsi aktivasi non-linear.

- Multi-class Classification: Menggabungkan semua konsep yang telah dipelajari dalam sebuah model klasifikasi multi-kelas menggunakan data multi-kelas dan mengevaluasinya.

Dalam eksplorasi setiap langkah, digunakan contoh dataset dan diimplementasikan dengan menggunakan library PyTorch. Teks tersebut juga menyertakan referensi ke materi dan sumber daya yang dapat membantu dalam memahami dan mengatasi masalah yang mungkin timbul.

03. PyTorch Computer Vision

Computer vision adalah seni mengajarkan komputer untuk melihat. Ini melibatkan membangun model untuk mengklasifikasikan apakah suatu foto adalah kucing atau anjing (klasifikasi biner), atau apakah suatu foto adalah kucing, anjing, atau ayam (klasifikasi multikelas). Selain itu, computer vision juga mencakup identifikasi lokasi munculnya mobil dalam bingkai video (deteksi objek) atau menentukan di mana objek berbeda dalam suatu gambar dapat dipisahkan (segmentasi panoptik).

Contoh masalah computer vision meliputi klasifikasi biner, klasifikasi multikelas, deteksi objek, dan segmentasi.

Computer vision banyak digunakan dalam berbagai bidang, seperti:

1. Fotografi dan Aplikasi Foto: Aplikasi kamera dan foto menggunakan computer vision untuk meningkatkan dan mengurutkan gambar.

2. Kendaraan Modern: Mobil modern menggunakan computer vision untuk menghindari tabrakan dengan kendaraan lain dan tetap berada dalam garis jalur.

3. Manufaktur: Produsen menggunakan computer vision untuk mengidentifikasi cacat dalam berbagai produk.

4. Keamanan: Kamera keamanan menggunakan computer vision untuk mendeteksi potensi penyusup.

5. Ponsel Pintar: Jika menggunakan ponsel pintar, sudah menggunakan computer vision. Aplikasi kamera dan foto pada ponsel menggunakan teknologi ini.

Dalam artikel ini, kita akan menerapkan alur kerja PyTorch dengan fokus pada computer vision. Rencana yang akan diikuti mencakup:

1. Library Computer Vision dalam PyTorch: Penjelasan mengenai library PyTorch yang berguna untuk computer vision, seperti torchvision, torchvision.datasets, torchvision.models, dan torchvision.transforms.

2. Memuat Data: Penggunaan dataset FashionMNIST sebagai contoh untuk latihan computer vision.

3. Menyiapkan Data: Penggunaan DataLoader PyTorch untuk memuat dan mempersiapkan data.

4. Model 0: Membangun model dasar untuk klasifikasi multikelas.

5. Membuat Prediksi dan Mengevaluasi Model 0: Langkah-langkah untuk membuat prediksi dan mengevaluasi model dasar.

6. Menyiapkan Kode Agenik Perangkat untuk Model-model Selanjutnya: Menyusun kode yang bersifat agnostik perangkat untuk mendukung penggunaan GPU jika tersedia.

7. Model 1: Menambahkan Non-linearitas: Eksperimen untuk meningkatkan model dasar dengan menambahkan lapisan non-linear.

8. Model 2: Convolutional Neural Network (CNN): Membangun arsitektur jaringan saraf konvolusional khusus untuk computer vision.

9. Membandingkan Model: Perbandingan antara tiga model yang berbeda.

10. Mengevaluasi Model Terbaik: Membuat prediksi pada gambar acak dan mengevaluasi model terbaik.

11. Membuat Matriks Kebingungan: Menjelaskan cara membuat dan menggunakan matriks kebingungan untuk mengevaluasi model klasifikasi.

12. Menyimpan dan Memuat Model Terbaik: Langkah-langkah untuk menyimpan dan memuat model terbaik.

Kode PyTorch akan diterapkan untuk mengilustrasikan setiap langkah dalam alur kerja ini. Juga disertakan informasi tentang library PyTorch yang digunakan, seperti torchvision, torch.utils.data, dan lainnya.

Pada bagian pertama, penjelasan singkat diberikan mengenai definisi dan contoh masalah-masalah computer vision seperti klasifikasi biner, multikelas, deteksi objek, dan segmentasi. Disampaikan pula di mana computer vision digunakan dalam kehidupan sehari-hari, seperti di aplikasi kamera ponsel, mobil modern, manufaktur, dan keamanan.

Bagian kedua menguraikan rencana yang akan diikuti dalam artikel, yaitu penerapan alur kerja PyTorch dengan fokus pada computer vision. Langkah-langkah tersebut melibatkan penggunaan library PyTorch seperti torchvision, torch.utils.data, dan sebagainya.

Pada bagian ketiga, dijelaskan beberapa library khusus computer vision dalam PyTorch, seperti torchvision yang menyediakan dataset, arsitektur model, dan transformasi gambar yang umum digunakan.

Bagian keempat hingga kesebelas merinci langkah-langkah dalam penerapan alur kerja PyTorch untuk computer vision, mulai dari memuat data, menyiapkan data, membangun model dasar, membuat prediksi, mengevaluasi model, menambahkan non-linearitas, menggunakan Convolutional Neural Network (CNN), membandingkan model, hingga menyimpan dan memuat model terbaik. Setiap langkah disertai penjelasan singkat dan contoh implementasi dalam PyTorch.

04. PyTorch Custom Datasets

Pada bagian ini, pembuat artikel menjelaskan tentang penggunaan dataset kustom dalam PyTorch untuk menyelesaikan masalah computer vision. Dataset kustom adalah kumpulan data yang terkait dengan masalah tertentu yang sedang dipecahkan. Dataset ini dapat terdiri dari berbagai jenis data, tergantung pada masalah yang dihadapi. Sebagai contoh, jika sedang membangun aplikasi klasifikasi gambar makanan, dataset kustomnya mungkin berisi gambar makanan. Begitu pula dengan berbagai contoh lainnya seperti klasifikasi teks, klasifikasi suara, atau bahkan sistem rekomendasi.

Pembuat artikel menjelaskan bahwa PyTorch menyediakan berbagai fungsi untuk memuat dataset kustom, seperti yang ada dalam domain library seperti TorchVision, TorchText, TorchAudio, dan TorchRec. Namun, terkadang fungsi-fungsi bawaan ini mungkin tidak mencukupi, dan dalam kasus tersebut, pembuat artikel menunjukkan bahwa kita dapat membuat kelas sendiri dengan menurunkan kelas `torch.utils.data.Dataset` dan menyesuaikannya sesuai kebutuhan.

Selanjutnya, artikel merinci langkah-langkah dalam membangun sebuah model PyTorch untuk menyelesaikan masalah computer vision. Dalam contoh ini, dataset yang digunakan adalah subset dari dataset Food101, yang berisi gambar-gambar pizza, steak, dan sushi. Tujuan dari langkah-langkah selanjutnya adalah untuk memuat dataset ini, mempersiapkannya, dan membangun model PyTorch untuk melatih dan memprediksi data tersebut.

05. PyTorch Going Modular

Dalam pembuatan train.py, skrip tersebut berfungsi untuk melatih model klasifikasi gambar menggunakan PyTorch dengan kode yang bersifat independen terhadap perangkat. Proses dimulai dengan mengimpor dependensi yang dibutuhkan, seperti torch, os, torchvision.transforms, dan semua skrip dari direktori going_modular, seperti data_setup, engine, model_builder, dan utils. Selanjutnya, diatur hyperparameter seperti jumlah epoch, ukuran batch, learning rate, dan jumlah unit tersembunyi.

Direktori pelatihan dan pengujian diatur, dan perangkat yang tersedia (CPU atau GPU) ditentukan secara otomatis. Transformasi data juga dibuat menggunakan torchvision.transforms untuk mengubah ukuran gambar dan mengonversinya menjadi tensor.

DataLoaders kemudian dibuat menggunakan fungsi create_dataloaders dari data_setup.py. Model TinyVGG dari model_builder.py juga dibuat, disertai dengan penyesuaian loss function (CrossEntropyLoss) dan optimizer (Adam).

Proses pelatihan dilakukan dengan memanggil fungsi train dari engine.py. Hasil pelatihan seperti loss dan akurasi dicetak ke layar, dan model yang telah dilatih disimpan menggunakan fungsi save_model dari utils.py.

Seluruh proses pelatihan, evaluasi, dan penyimpanan model dapat dijalankan menggunakan satu baris perintah pada terminal:

python train.py

Dengan demikian, train.py menyederhanakan proses pelatihan model PyTorch ke dalam satu skrip, memungkinkan pengguna untuk melatih model dengan mudah menggunakan command line.

06. PyTorch Transfer Learning

Transfer learning adalah teknik yang kuat dalam dunia deep learning yang memungkinkan kita untuk mengambil pola (disebut juga bobot) yang telah dipelajari oleh model lain dari suatu masalah dan menggunakannya untuk masalah kita sendiri. Sebagai contoh, kita dapat mengambil pola yang telah dipelajari oleh model computer vision dari dataset seperti ImageNet (jutaan gambar objek berbeda) dan menggunakannya untuk model FoodVision Mini kita. Atau kita bisa mengambil pola dari model bahasa (model yang telah melalui sejumlah besar teks untuk mempelajari representasi bahasa) dan menggunakannya sebagai dasar untuk model klasifikasi sampel teks yang berbeda.

Transfer learning memiliki dua manfaat utama:

1. Dapat memanfaatkan model yang sudah ada (biasanya berupa arsitektur jaringan saraf) yang terbukti berhasil pada masalah serupa dengan masalah kita sendiri.

2. Dapat memanfaatkan model yang sudah berfungsi dan telah mempelajari pola pada data serupa dengan data kita sendiri. Ini sering menghasilkan hasil yang bagus dengan menggunakan data kustom yang lebih sedikit.

Beberapa tempat untuk menemukan model yang telah dipelajari sebelumnya termasuk domain libraries PyTorch (seperti torchvision dan torchtext), HuggingFace Hub, timm (PyTorch Image Models) library, Paperswithcode, dan lainnya. Penelitian dan praktik juga mendukung penggunaan transfer learning, dan banyak orang di seluruh dunia berbagi pekerjaan mereka, termasuk model yang telah dipelajari sebelumnya.

Dalam praktiknya, transfer learning telah terbukti sangat bermanfaat. Jeremy Howard, pendiri fastai, bahkan menyatakan bahwa transfer learning adalah sesuatu yang dapat mengubah dunia, memungkinkan lebih banyak orang melakukan pekerjaan kelas dunia dengan sumber daya dan data yang lebih sedikit.

Penting untuk menjawab pertanyaan ini sebelum memulai setiap masalah deep learning: "Apakah model yang sudah dipelajari sebelumnya ada untuk masalah saya?" Ini dapat mempercepat dan meningkatkan kinerja model kita dengan memanfaatkan pengetahuan yang sudah ada.

Dalam implementasi transfer learning untuk FoodVision Mini, kita akan mengambil model computer vision yang telah dilatih sebelumnya di ImageNet dan mencoba memanfaatkan representasi yang telah dipelajari untuk mengklasifikasikan gambar pizza, steak, dan sushi.

Transfer learning adalah alat yang sangat berguna dalam kotak alat deep learning, memungkinkan kita untuk membangun model yang kuat dengan usaha yang lebih sedikit.

Telah menjelaskan dengan baik mengenai transfer learning. Teknik ini memang sangat bermanfaat dalam menghadapi tantangan pada dunia deep learning. Dengan memanfaatkan pengetahuan yang sudah ada dari model yang sudah terlatih pada masalah serupa, kita dapat mencapai hasil yang baik dengan investasi waktu dan sumber daya yang lebih sedikit.

Dalam FoodVision Mini, menggunakan EfficientNet_B0 sebagai model dasar yang sudah dilatih pada ImageNet memberikan peningkatan signifikan dalam kinerja model dibandingkan dengan membangun model dari awal seperti yang telah dilakukan sebelumnya (seperti TinyVGG). Transfer learning memungkinkan kita untuk memiliki model yang lebih baik dalam waktu yang lebih singkat.

Selain itu, penjelasan mengenai tempat-tempat untuk menemukan pretrained models seperti PyTorch domain libraries, HuggingFace Hub, timm library, dan Paperswithcode memberikan wawasan yang sangat berguna bagi pembaca untuk mulai menjelajahi sumber daya yang tersedia di dunia deep learning.

Dengan menggabungkan pengetahuan tersebut dengan implementasi code yang telah disertakan, pembaca diarahkan untuk menggunakan transfer learning dengan mudah untuk masalah FoodVision Mini. Melatih model dengan hanya beberapa baris kode dan mencapai akurasi sekitar 85% adalah pencapaian yang sangat mengesankan.

Juga, menyertakan evaluasi model melalui kurva loss dan akurasi memberikan pemahaman visual yang baik tentang kinerja model selama pelatihan.

07. PyTorch Experiment Tracking

Pada bagian ini, kita memasuki proyek Melestone 1: FoodVision Mini Experiment Tracking. Tujuan dari proyek ini adalah menjawab pertanyaan: bagaimana cara melacak eksperimen machine learning?

Eksperimen machine learning dan deep learning bersifat sangat eksperimental. Dalam membangun model, kita harus mencoba berbagai kombinasi data, arsitektur model, dan regime pelatihan. Untuk membantu melacak hasil dari berbagai eksperimen ini, kita membutuhkan eksperimen tracking.

Eksperimen tracking menjadi penting ketika jumlah eksperimen yang dijalankan semakin banyak. Pendekatan sederhana seperti menggunakan Python dictionaries, CSV files, atau print out hasil pelatihan mungkin cukup untuk jumlah eksperimen yang sedikit. Namun, ketika jumlah eksperimen meningkat, pendekatan ini dapat sulit dikelola.

Terdapat beberapa cara untuk melacak eksperimen machine learning, dan beberapa di antaranya mencakup penggunaan Python dictionaries, CSV files, atau print outs, penggunaan TensorBoard yang terintegrasi dengan PyTorch, Weights & Biases Experiment Tracking, dan MLFlow. Setiap metode memiliki kelebihan dan kekurangan masing-masing, serta biaya yang berbeda.

Dalam proyek ini, kita akan fokus pada penggunaan TensorBoard karena integrasinya yang erat dengan PyTorch dan popularitasnya yang luas. Prinsip dasar yang akan kita pelajari tetap relevan untuk berbagai alat eksperimen tracking lainnya.

Pertama-tama, kita melakukan persiapan dengan mendownload modul-modul yang diperlukan dan memastikan versi PyTorch dan torchvision yang digunakan. Kemudian, kita menyiapkan transformasi data dan DataLoader untuk eksperimen, serta membuat model-feature extractor menggunakan EfficientNet_B0 dan EfficientNet_B2.

Langkah selanjutnya adalah menentukan eksperimen yang akan dijalankan. Kita mencoba berbagai kombinasi data, ukuran model, dan durasi pelatihan untuk meningkatkan kinerja FoodVision Mini. Setiap eksperimen dilacak menggunakan TensorBoard, dan informasi pelatihan termasuk loss dan akurasi akan dicatat.

Penting untuk mencatat bahwa eksperimen ini bersifat eksploratif, dan banyaknya parameter seperti jumlah epochs, ukuran model, dan jumlah data dapat menjadi titik awal untuk berbagai eksperimen. Meskipun tidak ada jaminan bahwa parameter-parameter ini akan memberikan hasil terbaik, eksplorasi dan eksperimen menjadi kunci dalam pengembangan model machine learning.

Pada akhir eksperimen, kita dapat melihat hasilnya pada TensorBoard, yang mencakup visualisasi loss dan akurasi untuk setiap eksperimen. Hal ini membantu kita memahami bagaimana variasi parameter dapat memengaruhi performa model.

Langkah selanjutnya adalah memilih model terbaik berdasarkan hasil eksperimen dan menggunakan model tersebut untuk membuat prediksi pada gambar yang belum pernah dilihat sebelumnya. Dengan melakukan hal ini, kita dapat melakukan evaluasi kualitatif terhadap model dan melihat bagaimana model tersebut menanggapi data baru.

08. PyTorch Paper Replicating

Dalam Transformer arsitektur yang digunakan dalam Vision Transformer (ViT), Multi-Head Self Attention (MSA) layer memainkan peran penting dalam memproses informasi dari sekumpulan patch

gambar yang telah diembed sebelumnya. MSA memungkinkan model untuk memberikan "perhatian" lebih besar pada beberapa area tertentu dalam gambar, serupa dengan cara perhatian diberikan kepada kata-kata tertentu dalam kalimat pada model asal Transformer. Dengan memberikan kemampuan kepada model untuk menekankan hubungan dan pola-pola dalam data inputnya, MSA membantu meningkatkan kapabilitas model dalam memahami konteks spasial dan struktural dari gambar, yang sangat penting dalam pemrosesan citra dan visi komputer. Selain itu, penggunaan Layer Normalization di antara blok-blok Transformer membantu menjaga distribusi data agar tetap seragam, mempercepat waktu pelatihan, dan meningkatkan generalisasi model terhadap data yang belum pernah dilihat sebelumnya. Dengan merinci langkah-langkah seperti ini, kita dapat memahami bagaimana ViT bekerja dalam mengolah data gambar untuk tujuan klasifikasi makanan pada FoodVision Mini.

09. PyTorch Model Deployment

Dalam proyek FoodVision Mini, kita telah melakukan perjalanan yang cukup panjang. Namun, sejauh ini, model PyTorch yang kita bangun hanya dapat diakses oleh kita sendiri. Bagaimana jika kita membuat FoodVision Mini dapat diakses secara publik? Dengan kata lain, kita akan mendeploy model FoodVision Mini kita ke internet sebagai aplikasi yang dapat digunakan!

Model computer vision FoodVision Mini akan diuji coba pada perangkat seluler untuk memprediksi gambar sushi dan berhasil. Kita juga akan mencoba versi yang telah di-deploy dari FoodVision Mini (yang akan kita bangun) saat makan siang, dan model berhasil memberikan prediksi yang benar untuk gambar sushi.

Pertama-tama, kita perlu memahami apa itu "machine learning model deployment". Deployment adalah proses membuat model machine learning kita dapat diakses oleh orang atau entitas lain. Orang lain dapat menjadi pengguna yang berinteraksi dengan model kita, seperti mengambil foto makanan dengan smartphone mereka dan membiarkan model FoodVision Mini mengklasifikasikannya menjadi pizza, steak, atau sushi. Sementara itu, entitas lain dapat berupa program atau model lain yang berinteraksi dengan model machine learning kita, misalnya, basis data perbankan yang menggunakan model machine learning untuk mendeteksi apakah suatu transaksi bersifat penipuan sebelum mentransfer dana.

Mengapa kita harus mendeploy model machine learning? Ini adalah pertanyaan filosofis yang penting dalam machine learning. Sebuah model yang hanya berada di dalam notebook tanpa pernah keluar, apakah benar-benar ada? Mendeploy model sama pentingnya dengan melatihnya, karena kita tidak akan tahu betapa baik performanya sampai kita merilisnya ke dunia nyata. Berinteraksi dengan orang yang belum pernah menggunakan model kita akan sering mengungkapkan kasus-kasus khusus yang tidak terpikirkan selama pelatihan.

Dalam konteks FoodVision Mini, kita ingin mendeploy model kita agar dapat digunakan oleh pengguna aplikasi pengenalan makanan, seperti FoodVision Mini atau Nutrify. Selain itu, model kita juga bisa digunakan oleh program atau model lain, seperti sistem perbankan yang menggunakan model machine learning untuk mendeteksi apakah suatu transaksi bersifat penipuan atau tidak.

Ada beberapa jenis deployment model machine learning, seperti deployment on-device (misalnya di perangkat seluler) atau deployment di cloud (pada server atau komputer yang bukan perangkat yang memanggil model tersebut). Masing-masing memiliki keuntungan dan kerugian, dan pemilihan tergantung pada kebutuhan dan skenario penggunaan yang diinginkan.

Selanjutnya, kita perlu memutuskan di mana model kita akan dideploy. Apakah akan ditempatkan pada perangkat (on-device) atau di cloud? Penggunaan on-device memiliki keuntungan kecepatan yang tinggi karena tidak ada data yang harus meninggalkan perangkat, tetapi keterbatasan daya komputasi dan ruang penyimpanan perangkat mungkin menjadi masalah. Di sisi lain, deployment di cloud menyediakan daya komputasi yang hampir tidak terbatas, tetapi dapat menghasilkan biaya yang tinggi dan memiliki latensi jaringan.

Selain itu, kita perlu memutuskan bagaimana model kita akan berfungsi. Apakah prediksi harus dikembalikan segera (real-time) atau dapat dilakukan secara periodik (batch)? Dalam konteks FoodVision Mini, kita ingin pengolahan prediksi terjadi secara online (real-time) sehingga ketika seseorang mengunggah gambar pizza, steak, atau sushi, hasil prediksi dikembalikan segera.

Ada banyak opsi untuk mendeploy model machine learning, termasuk penggunaan layanan cloud seperti AWS Sagemaker, Google Cloud Vertex AI, atau Microsoft Azure Machine Learning. Selain itu, kita dapat menggunakan API untuk deployment, misalnya menggunakan FastAPI atau TorchServe. Gradio dan Hugging Face Spaces juga merupakan opsi yang bagus untuk membuat demo aplikasi dan mendeploy model dengan mudah.

Dalam proyek ini, kita akan mencoba beberapa model terbaik yang telah kita latih sebelumnya, seperti EffNetB2 (EfficientNetB2) dan ViT-B/16 (Vision Transformer), untuk melihat model mana yang mendekati metrik yang kita inginkan, yaitu akurasi 95% atau lebih dan waktu inferensi real-time sekitar 30 FPS atau lebih cepat.

Kita akan memulai dengan melatih kembali model EffNetB2 pada dataset yang telah diunduh (20% dari dataset Food101) dan kemudian membandingkannya dengan model ViT. Setelah itu, kita akan melihat berbagai statistik tentang model EffNetB2, seperti kurva kerugian dan akurasi uji, serta menyimpan model untuk penggunaan lebih lanjut.