Technical Report UAS Machine Learning
Deep Learning with PyTorch

Disusun oleh :

Giovanni Nathaniel 1103202211

PROGRAM STUDI TEKNIK KOMPUTER
FAKULTAS TEKNIK ELEKTRO
UNIVERSITAS TELKOM

2023

A. Pendahuluan

Deep learning adalah metode dalam kecerdasan buatan (AI) yang mengajarkan komputer untuk memproses data dengan cara yang terinspirasi otak manusia. Model deep learning dapat mengenali pola kompleks dalam gambar, teks, suara, dan data lain untuk menghasilkan wawasan dan prediksi yang akurat. Anda dapat menggunakan metode deep learning untuk mengotomatiskan tugas yang biasanya membutuhkan kecerdasan manusia, seperti mendeskripsikan citra atau menyalin file suara ke dalam teks.

B. Analisa Kode

Tensor adalah struktur data yang digunakan untuk menyimpan dan memanipulasi data dalam bentuk array multidimensi. Kode tersebut menunjukkan berbagai operasi dasar pada Tensor, seperti pembuatan Tensor kosong dengan berbagai ukuran, inisialisasi Tensor dengan nilai acak atau nol, mengubah ukuran Tensor, dan melakukan operasi matematika seperti penjumlahan, pengurangan, perkalian, dan pembagian antara Tensor. Kode juga menunjukkan pengindeksan dan pengambilan nilai elemen dari Tensor. Selain itu, kode juga mencakup konversi antara Tensor dan array NumPy, serta penggunaan Tensor pada perangkat GPU jika tersedia. Dengan menggunakan Tensor dalam PyTorch, kita dapat dengan mudah melakukan komputasi numerik dan pembelajaran mesin secara efisien.

Fitur autograd pada PyTorch. Fitur ini memungkinkan kita untuk menghitung gradien dari tensor secara otomatis. Pada kode tersebut, tensor x dibuat dengan requires_grad=True untuk menandakan bahwa gradien tensor akan dilacak. Tensor y kemudian dibuat sebagai hasil penjumlahan tensor x dengan 2. Kemudian, tensor z dibuat dengan mengalikan y dengan dirinya sendiri dan dikalikan dengan 3. Kemudian, tensor z dijadikan rata-rata dengan menggunakan metode .mean(). Setelah itu, dipanggil metode .backward() pada tensor z untuk menghitung gradien. Gradien tensor x dapat diakses dengan menggunakan atribut .grad dari tensor x. Selanjutnya, tensor x baru dibuat dengan requires_grad=True, kemudian dilakukan pengulangan sebanyak 10 kali untuk mengalikan tensor y dengan 2. Setelah itu, tensor y di-backward dengan menggunakan tensor v sebagai gradien. Gradien tensor x dapat diakses kembali dengan menggunakan atribut .grad dari tensor x.
Selanjutnya, dilakukan percobaan dengan tensor a dan b untuk menunjukkan penggunaan metode .requires_grad_(), .detach(), dan penggunaan torch.no_grad() untuk menghentikan pelacakan gradien. Terakhir, dilakukan iterasi sebanyak 3 kali untuk menghitung gradien tensor weights dan mengupdate nilainya menggunakan gradien dan learning rate. Hasil akhirnya adalah tensor weights dan nilai model_output. Fitur autograd pada PyTorch memudahkan dalam menghitung gradien secara otomatis dan sangat berguna dalam pelatihan model pada pembelajaran mesin. backpropagation pada framework PyTorch. Backpropagation adalah algoritme yang digunakan untuk menghitung gradien melalui jaringan saraf (neural network) dalam rangka melatih model secara efisien. Pada kode tersebut, sebuah tensor x dan y dibuat dengan nilai 1.0 dan 2.0. Selanjutnya, tensor w dibuat dengan nilai 1.0 dan diberi atribut requires_grad=True untuk mengindikasikan bahwa gradien tensor tersebut akan dilacak.

Kemudian, dilakukan perhitungan prediksi y_predicted dengan mengalikan tensor w dengan tensor x. Selanjutnya, dihitung nilai loss yang merupakan selisih kuadrat antara y_predicted dan y. Loss ini merupakan fungsi objektif yang ingin diminimalkan selama pelatihan model.
Selanjutnya, dilakukan langkah backpropagation dengan memanggil metode .backward() pada tensor loss. Hal ini akan menghitung gradien loss terhadap tensor w secara otomatis. Gradien tersebut dapat diakses melalui atribut .grad dari tensor w.
Selanjutnya, dilakukan peng-update nilai tensor w dengan menggunakan metode .grad dan learning rate. Dalam kode tersebut, nilai learning rate diberikan sebagai 0.01, dan dilakukan pengurangan tensor w dengan hasil perkalian antara learning rate dan gradien w.
Setelah meng-update nilai w, gradien tensor w direset menjadi nol menggunakan metode .zero_() untuk digunakan pada iterasi berikutnya.
Proses ini diulang sebanyak yang diperlukan untuk melatih model dengan melakukan iterasi, menghitung prediksi, menghitung loss, melakukan backpropagation, dan meng-update nilai parameter model. Backpropagation memungkinkan model untuk belajar dari kesalahan dan menyesuaikan bobot dan parameter agar mendekati solusi yang lebih baik.

Training yang dilakukan pada model ini dilakukan dengan memperbarui parameter model menggunakan pengoptimal seperti Stochastic Gradient Descent (SGD).

Dataset dan DataLoader dalam framework PyTorch. Dataset dan DataLoader adalah komponen penting dalam pemrosesan data pada PyTorch. Berikut adalah penjelasan mengenai masing-masing bagian kode:
Pertama, didefinisikan kelas WineDataset yang merupakan turunan dari kelas Dataset. Kelas ini digunakan untuk memuat dan mengolah data set anggur dari file CSV. Data CSV tersebut diubah menjadi tensor menggunakan NumPy dan diinisialisasi dalam atribut x_data dan y_data dalam bentuk tensor. Metode getitem() digunakan untuk mengambil sampel data berdasarkan indeks, sementara metode len() mengembalikan jumlah total sampel dalam dataset.
Setelah mendefinisikan kelas WineDataset, instance dataset dibuat dengan menggunakan WineDataset(). Kemudian, dilakukan akses terhadap data pertama dalam dataset menggunakan indexing dataset[0], dimana features dan labels didefinisikan sebagai hasil dari akses tersebut.
Selanjutnya, dilakukan inisialisasi train_loader dengan menggunakan DataLoader. DataLoader memungkinkan untuk membagi dataset menjadi batch-batch kecil saat melatih model. Dalam kode tersebut, dataset=dataset mengacu pada instance dataset yang telah dibuat sebelumnya. Parameter batch_size=4 menunjukkan bahwa setiap batch akan terdiri dari 4 sampel data. shuffle=True mengindikasikan bahwa data akan diacak sebelum diproses oleh DataLoader. num_workers=2 menunjukkan jumlah pekerja yang akan digunakan untuk memuat data dalam background.
Kemudian, dilakukan iterasi melalui train_loader untuk melatih model. Dalam contoh ini, dilakukan iterasi selama dua epoch (num_epochs=2) dan menggunakan enumerate(train_loader) untuk mendapatkan setiap batch data dengan inputs dan

labels. Dalam loop tersebut, dicetak informasi mengenai jumlah langkah dan dimensi inputs dan labels setiap lima langkah menggunakan print().
Selanjutnya, digunakan kelas torchvision.datasets.MNIST untuk mendownload dataset MNIST yang berisi gambar-gambar digit tulisan tangan. Data tersebut kemudian dimuat menggunakan DataLoader dengan batch_size=3 dan shuffle=True. DataLoader memungkinkan kita untuk dengan mudah membagi dataset menjadi batch-batch kecil saat melatih model. Dengan menggunakan DataLoader, kita dapat melatih model secara efisien dengan memproses data dalam batch-batch kecil, mengurangi kebutuhan memori dan meningkatkan kecepatan pelatihan model. Dengan menggunakan Dataset dan DataLoader dalam PyTorch, kita dapat dengan mudah mengelola dan memuat data secara efisien dalam proses pelatihan model.

Sebelum melakukan Transfer Learning saya membuat code untuk mengakses Google Drive lewat Google Collab

Transfer learning menggunakan model ResNet-18 pada dataset hymenoptera_data. Transfer learning adalah teknik yang memanfaatkan pengetahuan yang sudah dipelajari pada model yang sudah dilatih sebelumnya untuk mengatasi masalah baru.
Pertama, dilakukan pengaturan transformasi data menggunakan data_transforms. Transformasi ini mencakup operasi seperti pemotongan, flipping, normalisasi, dan lainnya untuk mempersiapkan data sebelum dimasukkan ke dalam model.
Selanjutnya, dataset dan dataloader dibuat berdasarkan transformasi yang telah ditentukan. Dataset diambil dari folder "train" dan "val" pada direktori hymenoptera_data. Dalam dataset ini, gambar-gambar diubah menjadi tensor dan dinormalisasi menggunakan mean dan std yang telah ditentukan.
Kemudian, dilakukan pengaturan perangkat (device) untuk menjalankan model pada CPU atau GPU. Jika tersedia, akan menggunakan GPU untuk kecepatan yang lebih tinggi.
Dilakukan fungsi imshow() untuk menampilkan beberapa contoh gambar dari dataset dengan label yang sesuai.
Selanjutnya, didefinisikan fungsi train_model() yang akan melatih model. Fungsi ini akan melalui beberapa epoch dan melatih model dengan menggunakan data dari dataloader. Dalam setiap epoch, akan dihitung loss, akurasi, dan dilakukan backpropagation untuk memperbarui parameter model.
Setelah itu, model ResNet-18 diinisialisasi dengan pre-trained weights dan dilakukan fine-tuning dengan mengganti fully connected layer pada akhirnya. Model ini akan melatih hanya lapisan terakhir (fc) sementara lapisan sebelumnya tidak akan dilatih lagi.
Selanjutnya, dilakukan pelatihan model_conv dengan menggunakan model yang sudah ditentukan sebelumnya. Proses ini melibatkan pelatihan hanya pada fully connected layer (fc) dan tidak melibatkan lapisan sebelumnya.
Pada akhirnya, model yang telah dilatih akan digunakan untuk tugas yang relevan, seperti klasifikasi gambar pada dataset hymenoptera_data.
Transfer learning merupakan teknik yang berguna ketika kita memiliki dataset terbatas atau ketika ingin menghemat waktu dan sumber daya untuk melatih model dari awal. Dengan memanfaatkan pengetahuan yang telah dipelajari pada model yang

sudah dilatih sebelumnya, kita dapat mencapai performa yang baik dalam tugas baru dengan menggunakan waktu yang lebih sedikit.