Machine Learning Techical Report

"Deep Learning with PyTorch"

Universitas Telkom

Oleh:

Muhammad Haidar Abdul Jabbar

1103202071

Program Studi Teknik Komputer

Fakultas Teknik Elektro

Universitas Telkom

2022/2023

Tensor Basic

PyTorch adalah sebuah library deep learning yang melakukan operasi pada array numerik yang disebut tensor. Kode ini menunjukkan bagaimana membuat tensor dengan dimensi yang berbeda menggunakan torch.empty(), torch.rand(), torch.zeros(), dan torch.tensor(). Kode ini juga menunjukkan bagaimana memeriksa ukuran dan tipe data dari tensor menggunakan x.size() dan x.dtype, masing-masing.

Kode ini kemudian menunjukkan beberapa operasi tensor dasar seperti penjumlahan, pengurangan, perkalian, dan pembagian elemen-wise menggunakan operator +, -, *, dan /. Kode ini juga menunjukkan bagaimana memotong tensor menggunakan notasi indeks dan potongan.

Kode ini kemudian menunjukkan bagaimana mereshape tensor menggunakan torch.view(). Kode ini juga menunjukkan bagaimana mengonversi tensor PyTorch menjadi array NumPy dan sebaliknya menggunakan tensor.numpy() dan torch.from_numpy(), masing-masing.

Terakhir, kode ini menunjukkan beberapa operasi matematika dasar yang dapat dilakukan pada tensor seperti torch.exp(), torch.log(), torch.sin(), torch.abs(), torch.square(), torch.sqrt(), torch.ceil(), torch.round(), dan torch.clip().

Kode ini memberikan pemahaman dasar tentang tensor dan operasi PyTorch dan dapat berfungsi sebagai titik awal untuk tugas deep learning yang lebih canggih. PyTorch adalah alat yang kuat untuk membangun dan melatih neural network, dan kode ini memberikan dasar untuk memahami cara bekerja dengan tensor di PyTorch.

AutoGrad

Kode ini menjelaskan tentang bagaimana PyTorch melakukan automatic differentiation pada tensor. Automatic differentiation adalah teknik yang digunakan untuk menghitung gradien dari suatu fungsi secara otomatis. Kode ini menunjukkan bagaimana membuat tensor dengan requires_grad=True untuk melacak semua operasi pada tensor.

Kode ini juga menunjukkan bagaimana menghitung gradien dari suatu fungsi dengan menggunakan backward() pada tensor. Kode ini menunjukkan bagaimana menghentikan tensor untuk melacak history dengan menggunakan requires_grad_(False), detach(), atau with torch.no_grad():.

Kode ini juga menunjukkan bagaimana mengoptimasi model dengan menghitung gradien dan mengubah bobot model dengan menggunakan optimizer seperti torch.optim.SGD.

BackPropagation

Kode ini adalah contoh sederhana tentang bagaimana melakukan optimasi parameter pada PyTorch. Kode ini menunjukkan bagaimana membuat tensor dengan requires_grad=True

untuk melacak semua operasi pada tensor. Kemudian, kode ini menunjukkan bagaimana melakukan forward pass untuk menghitung loss dan backward pass untuk menghitung gradien.

Kode ini juga menunjukkan bagaimana mengoptimasi parameter dengan mengubah bobot model menggunakan with torch.no_grad(): dan mengatur ulang gradien menggunakan w.grad.zero_().

GradientManually

Kode tersebut adalah contoh sederhana dari regresi linear menggunakan NumPy. Regresi linear adalah teknik analisis data yang digunakan untuk menemukan hubungan antara variabel dependen dan sekumpulan variabel independen. Dalam kode tersebut, variabel X dan Y adalah data latih, variabel w adalah bobot model yang diinisialisasi dengan nilai 0.0, fungsi forward digunakan untuk menghitung output model, fungsi loss digunakan untuk menghitung mean squared error, dan fungsi gradient digunakan untuk menghitung gradien loss terhadap bobot. Kode tersebut melakukan iterasi sebanyak n_iters untuk menghitung output model, loss, dan gradien, dan memperbarui bobot menggunakan gradient descent. Akhirnya, kode tersebut mencetak output model untuk nilai input baru.

GradientDescent Auto

Kode tersebut adalah contoh sederhana dari regresi linear menggunakan PyTorch. Regresi linear adalah teknik analisis data yang digunakan untuk menemukan hubungan antara variabel dependen dan sekumpulan variabel independen. Dalam kode tersebut, variabel X dan Y adalah training data, variabel w adalah bobot model yang diinisialisasi dengan nilai 0.0 dan diatur agar memerlukan gradien, fungsi forward digunakan untuk menghitung output model, dan fungsi loss digunakan untuk menghitung mean squared error. Kode tersebut melakukan iterasi sebanyak n_iters untuk menghitung output model, loss, dan gradien, dan memperbarui bobot menggunakan stochastic gradient descent.

Loss & Optimizer

Kode tersebut menggunakan loss dan optimizer untuk melatih model linear regression. Pertama,mendefinisikan objek loss menggunakan nn.MSELoss() yang merupakan fungsi kerugian Mean Squared Error (MSE). Fungsi ini mengukur kesalahan antara prediksi model dan target output.

Selanjutnya, mendefinisikan optimizer menggunakan torch.optim.SGD([w], lr=learning_rate), di mana w adalah parameter yang akan dioptimasi dan learning_rate adalah laju pembelajaran yang menentukan seberapa besar perubahan yang dilakukan pada setiap langkah optimisasi. SGD (Stochastic Gradient Descent) adalah algoritme optimisasi yang digunakan di sini.

Selama proses training, dalam setiap iterasi, diperlukan prediksi menggunakan fungsi forward, menghitung loss menggunakan fungsi loss, melakukan backpropagation dengan memanggil l.backward() untuk menghitung gradien loss terhadap parameter w, dan memperbarui parameter w menggunakan optimizer.step(). Selanjutnya, kita mengatur ulang gradien menggunakan optimizer.zero_grad() untuk langkah optimisasi berikutnya.

Dalam setiap epoch kelipatan 10, dicetak epoch, nilai parameter w, dan loss saat itu. Setelah pelatihan selesai, mencetak prediksi setelah pelatihan menggunakan input 5 dengan memanggil fungsi forward(5) dan memperoleh nilai item dari hasil tensor untuk mencetak hanya nilainya.

Model Loss & Optimizer

Kode tersebut adalah implementasi linear regression menggunakan PyTorch dengan model yang dibangun menggunakan modul nn.Linear. Pertama, mendefinisikan data input (X) dan target output (Y) sebagai tensor PyTorch. Jumlah sampel dan fitur dihitung dengan X.shape dan dicetak. Kemudian, kita mendefinisikan data input untuk pengujian (X_test) sebagai tensor PyTorch.

Selanjutnya,mendefinisikan ukuran input dan output model. Model linear regression dibangun dengan menggunakan nn.Linear(input_size, output_size), di mana input_size dan output_size adalah ukuran dimensi input dan output. Model tersebut adalah objek dari kelas nn.Linear yang sudah tersedia di PyTorch.

Setelah itu, dicetak prediksi sebelum pelatihan menggunakan model(X_test). Selanjutnya, mendefinisikan fungsi kerugian (loss function) menggunakan nn.MSELoss(), dan optimizer menggunakan SGD dengan memanggil torch.optim.SGD(model.parameters(), lr=learning_rate). Dalam proses pelatihan, pada setiap iterasi, dilakukan prediksi menggunakan model(X), menghitung loss antara prediksi dan target menggunakan fungsi loss, melakukan backpropagation dengan memanggil l.backward() untuk menghitung gradien loss terhadap parameter model, dan memperbarui parameter model menggunakan optimizer.step(). Setelah itu, gradien diatur ulang dengan optimizer.zero_grad().

Dalam setiap epoch kelipatan 10, dicetak epoch, nilai parameter w, dan loss saat itu. Setelah pelatihan selesai, dicetak prediksi setelah pelatihan menggunakan model(X_test).

Linear Regression

Kode di atas adalah implementasi regresi linier menggunakan PyTorch untuk memprediksi data yang dibuat secara acak. Pertama, kita menggunakan datasets.make_regression dari modul sklearn untuk membuat data acak dengan jumlah sampel (n_samples) sebanyak 100, jumlah fitur (n_features) sebanyak 1, tingkat kebisingan (noise) sebesar 20, dan seed (random_state) sebesar 4. Data tersebut kemudian disimpan dalam variabel X_numpy (fitur) dan y_numpy (target).

Selanjutnya, data tersebut dikonversi menjadi tensor PyTorch menggunakan torch.from_numpy dan ditentukan tipe datanya sebagai float32. Selain itu, bentuk tensor target y diubah menggunakan y.view(y.shape[0], 1) untuk memastikan memiliki dimensi yang sesuai dengan output model.

Kemudian, diinisialisasi ukuran input (input_size) sesuai dengan jumlah fitur dan ukuran output (output_size) sebesar 1. Model regresi linier dibangun menggunakan nn.Linear(input_size, output_size).

Selanjutnya, ditentukan learning_rate untuk optimasi dan kriteria (criterion) yang merupakan fungsi kerugian Mean Squared Error (MSE). Optimizer yang digunakan adalah SGD (Stochastic Gradient Descent) dengan memanggil torch.optim.SGD(model.parameters(), lr=learning_rate).

Selama proses pelatihan, pada setiap iterasi epoch, dilakukan prediksi menggunakan model dengan model(X), menghitung loss antara prediksi dan target menggunakan fungsi criterion dengan criterion(y_predicted, y), melakukan backpropagation dengan memanggil loss.backward() untuk menghitung gradien loss terhadap parameter model, dan memperbarui parameter model menggunakan optimizer.step(). Setelah itu, gradien diatur ulang dengan optimizer.zero_grad().

Dalam setiap epoch kelipatan 10, dicetak epoch dan nilai loss saat itu. Setelah pelatihan selesai, dilakukan prediksi menggunakan model dan hasilnya diubah menjadi numpy array dengan model(X).detach().numpy(). Data asli dan hasil prediksi kemudian diplot menggunakan matplotlib untuk memvisualisasikan hasil regresi linier.

Logistic Regression

Kode tersebut adalah contoh penggunaan PyTorch untuk melakukan klasifikasi pada dataset kanker payudara. Kode tersebut menggunakan dataset kanker payudara dari library scikit-learn dan membaginya menjadi data latih dan data uji. Kemudian, kode tersebut melakukan normalisasi data menggunakan StandardScaler dan mengubah data menjadi tensor PyTorch. Kode tersebut mendefinisikan model menggunakan modul nn dan mengoptimasi model menggunakan stochastic gradient descent. Kode tersebut mencetak akurasi model pada data uji setelah pelatihan.

Data Loader

Kode tersebut adalah contoh penggunaan PyTorch untuk memuat dan memproses dataset wine menggunakan modul Dataset dan DataLoader. Kode tersebut mendefinisikan kelas WineDataset yang memuat data wine dari file CSV dan mengubahnya menjadi tensor PyTorch. Kemudian, kode tersebut menggunakan DataLoader untuk membagi data menjadi batch dan memuat data secara paralel. Kode tersebut juga mencetak fitur dan label dari data pertama dan batch pertama.

Transformers

Kode tersebut merupakan implementasi dataset kustom (WineDataset) dengan menggunakan kelas turunan dari torch.utils.data.Dataset pada PyTorch. Dataset ini digunakan untuk membaca dan memuat data dari file CSV (wine.csv). Dataset tersebut berisi data terkait angka-angka yang berkaitan dengan klasifikasi jenis-jenis anggur.

Dalam kelas WineDataset, metode __init__ dijalankan saat objek dataset dibuat dan menginisialisasi atribut x_data dan y_data yang merepresentasikan atribut dan label dari dataset, serta atribut transform yang digunakan untuk melakukan transformasi pada dataset. Metode __getitem__ digunakan untuk mengakses elemen dataset berdasarkan indeks, mengambil sampel (input dan target) dari data, dan menerapkan transformasi jika diberikan. Metode __len__ digunakan untuk mengembalikan jumlah sampel dalam dataset.

Selain itu, terdapat kelas transformasi yaitu ToTensor dan MulTransform. Kelas ToTensor mengkonversi sampel menjadi tensor PyTorch, sedangkan kelas MulTransform mengalikan input dengan faktor tertentu.

Pada bagian utama kode, dilakukan pengujian dataset dengan dan tanpa transformasi. Dataset pertama kali dibuat tanpa transformasi (transform=None) dan kemudian dilakukan akses ke elemen pertama dataset. Hasilnya, fitur dan label yang diperoleh adalah dalam bentuk numpy array. Selanjutnya, dilakukan pengujian dataset dengan transformasi tensor (transform=ToTensor()) dan transformasi tensor dengan perkalian (transform=composed). Hasilnya, fitur dan label yang diperoleh sudah dalam bentuk tensor PyTorch dan sudah mengalami transformasi sesuai yang ditentukan.

Softmax And Crossentropy

Kode tersebut adalah contoh implementasi softmax dan cross-entropy loss function menggunakan NumPy dan PyTorch. Softmax digunakan untuk menghitung probabilitas kelas, sedangkan cross-entropy loss digunakan untuk menghitung perbedaan antara dua distribusi probabilitas untuk serangkaian kejadian atau variabel acak yang disediakan. Kode tersebut juga menunjukkan bagaimana menggunakan fungsi loss cross-entropy pada PyTorch, baik untuk klasifikasi biner maupun multikelas. Selain itu, kode tersebut juga menunjukkan contoh penggunaan DataLoader pada PyTorch untuk memuat data dalam batch dan melakukan transformasi data menggunakan PyTorch Transform. Terakhir, kode tersebut juga menunjukkan contoh pembuatan model MLP (Multilayer Perceptron) menggunakan PyTorch.

Activation Functions

Kode tersebut adalah contoh penggunaan beberapa fungsi aktivasi pada PyTorch, yaitu softmax, sigmoid, tanh, ReLU, dan Leaky ReLU. Fungsi-fungsi aktivasi ini digunakan untuk mengaktifkan neuron dalam jaringan saraf dan memperkenalkan non-linearitas pada model.

Kode tersebut juga menunjukkan contoh pembuatan model MLP (Multilayer Perceptron) menggunakan PyTorch dan penggunaan fungsi loss BCELoss dan CrossEntropyLoss pada PyTorch. Fungsi-fungsi loss ini digunakan untuk mengukur perbedaan antara nilai prediksi dan nilai sebenarnya pada tugas klasifikasi biner dan multi class.

Plot Activations

Kode tersebut mendefinisikan dan memplot beberapa fungsi aktivasi yang umum digunakan dalam neural network. Bagian pertama kode mendefinisikan dan memplot fungsi sigmoid, yang merupakan fungsi aktivasi populer dalam jaringan saraf. Bagian selanjutnya mendefinisikan dan memplot fungsi aktivasi TanH, ReLU, Leaky ReLU, dan fungsi tangga. Fungsi-fungsi aktivasi tersebut digunakan untuk mengaktifkan neuron dalam jaringan saraf. Fungsi sigmoid adalah fungsi aktivasi dengan kurva berbentuk seperti huruf s. Fungsi biner sigmoid adalah fungsi sigmoid yang mempunyai unit output angka biner.

FeedForward

Kode tersebut merupakan implementasi dari sebuah neural network menggunakan PyTorch untuk melakukan klasifikasi pada dataset MNIST. Pertama-tama, dilakukan import library yang dibutuhkan seperti PyTorch, Torchvision, dan Matplotlib. Kemudian, dilakukan inisialisasi beberapa parameter seperti input size, hidden size, dan num_classes. Selanjutnya, dilakukan pengambilan data dari dataset MNIST dan dilakukan preprocessing pada data tersebut. Setelah itu, dilakukan pembuatan model neural network dengan menggunakan class NeuralNet yang memiliki 2 layer linear dan 1 layer ReLU. Kemudian, dilakukan training pada model tersebut dengan menggunakan Cross Entropy Loss dan optimizer Adam. Setelah training selesai, dilakukan pengujian pada model dengan menggunakan data test dan dihitung akurasi dari model tersebut.

Cnn

Kode tersebut merupakan implementasi Convolutional Neural Network (CNN) menggunakan PyTorch untuk image classification. CNN adalah salah satu jenis neural network yang digunakan untuk memproses data gambar. Kode tersebut terdiri dari beberapa bagian, yaitu: mengatur device yang digunakan, mengatur hyperparameter seperti jumlah epoch, batch size, dan learning rate, memuat dataset CIFAR-10, membuat model CNN, mengatur loss function dan optimizer, melakukan pelatihan model, dan menghitung akurasi model pada data uji. Selain itu, kode tersebut juga menampilkan beberapa gambar dari dataset CIFAR-10 menggunakan matplotlib. Implementasi CNN pada kode tersebut menggunakan library PyTorch yang merupakan pustaka tensor deep learning yang dioptimalkan berdasarkan Python dan Torch.

Transfer Learning

Kode tersebut merupakan implementasi pelatihan model jaringan saraf tiruan menggunakan kerangka kerja PyTorch untuk klasifikasi gambar pada dataset "hymenoptera_data".

Pertama, dilakukan konfigurasi transformasi data menggunakan transforms.Compose dari torchvision. Transformasi ini termasuk pemangkasan acak, pembalikan horizontal, konversi gambar menjadi tensor, dan normalisasi menggunakan mean dan std yang telah ditentukan.

Selanjutnya, dataset gambar diatur menggunakan datasets.ImageFolder, dan dataloader digunakan untuk memuat data dalam batch menggunakan torch.utils.data.DataLoader. Selain itu, dilakukan pengaturan ukuran dataset dan mendefinisikan kelas-kelas yang ada dalam dataset.

Dalam kode tersebut juga ditentukan perangkat yang digunakan (cuda atau cpu) dan fungsi imshow untuk menampilkan gambar dalam tensor.

Setelah itu, dilakukan pelatihan model menggunakan fungsi train_model. Di dalam fungsi ini, dilakukan pelatihan model menggunakan metode mini-batch gradient descent. Pada setiap epoch, model dievaluasi pada data pelatihan dan data validasi, dihitung loss dan akurasi, serta dilakukan pembaharuan bobot model menggunakan optimizer. Selama pelatihan, dilakukan penjadwalan learning rate menggunakan lr_scheduler.

Terakhir, dilakukan pemanggilan train_model untuk dua skenario. Pertama, dengan model resnet18 yang telah diinisialisasi dengan bobot pre-trained pada dataset ImageNet. Kemudian, dilakukan pelatihan pada model hanya dengan mengubah layer fully connected pada model tersebut.

Tensor board

Kode tersebut adalah implementasi pelatihan jaringan saraf tiruan menggunakan PyTorch untuk pengenalan angka pada dataset MNIST.

Pertama, dilakukan import modulyang diperlukan, seperti torch, torch.nn, torchvision, matplotlib.pyplot, sys, torch.nn.functional, dan SummaryWriter dari torch.utils.tensorboard.

Selanjutnya, dilakukan pengaturan perangkat yang digunakan, yaitu GPU jika tersedia atau CPU. Kemudian, ditentukan ukuran input, ukuran tersembunyi, jumlah kelas, jumlah epoch, ukuran batch, dan learning rate.

Dataset MNIST dibagi menjadi dataset pelatihan dan dataset pengujian. Dataset tersebut kemudian dimuat menggunakan torchvision.datasets.MNIST dan dibagi menjadi batch menggunakan torch.utils.data.DataLoader.

Contoh data dari dataset pengujian ditampilkan menggunakan matplotlib.pyplot dan ditambahkan ke tensorboard menggunakan SummaryWriter.

Selanjutnya, didefinisikan model jaringan saraf tiruan menggunakan kelas NeuralNet. Model ini memiliki tiga lapisan: lapisan linier pertama, fungsi aktivasi ReLU, dan lapisan linier kedua.

Setelah itu, ditentukan fungsi loss (CrossEntropyLoss) dan optimizer (Adam) untuk melatih model. Model dan data pelatihan dijalankan melalui beberapa epoch, dan pada setiap langkah, dilakukan perhitungan loss, pembaruan bobot, dan penghitungan akurasi. Hasil pelatihan juga ditambahkan ke tensorboard.

Setelah selesai pelatihan, dilakukan evaluasi akurasi pada dataset pengujian. Hasil prediksi dan labelnya digunakan untuk menghitung akurasi dan menghasilkan kurva presisi-recall (PR curve) untuk setiap kelas menggunakan torch.nn.functional.softmax dan SummaryWriter.

Save Load

Kode di atas merupakan contoh implementasi PyTorch untuk membuat, menyimpan, dan meload model serta parameter terkait. Kode ini terdiri dari definisi sebuah model linear sederhana, penyimpanan model ke file, pengambilan model dari file yang telah disimpan, penyimpanan state_dict model, pengambilan state_dict model yang telah disimpan, pengaturan optimizer, pembuatan checkpoint, penyimpanan checkpoint ke file, pengambilan checkpoint dari file, pengaturan parameter model dan optimizer dari checkpoint, dan evaluasi model.