Zero to Mastery Learn PyTorch for Deep Learning Course Summary

00.PyTorch Fundamentals

Nama : Ihsan Ridho Asmoro

NIM : 1103204184

Bagian: Pengenalan PyTorch

Apa itu PyTorch?

PyTorch adalah framework machine learning dan deep learning yang bisa digunakan secara gratis.

Penggunaan PyTorch:

PyTorch memudahkan kita dalam mengelola dan memproses data.

Kita bisa menulis algoritma machine learning dengan menggunakan kode Python.

Penggunaan di Industri dan Penelitian:

Perusahaan besar seperti Meta (Facebook), Tesla, dan Microsoft menggunakan PyTorch.

PyTorch juga digunakan dalam penelitian kecerdasan buatan oleh perusahaan seperti OpenAI.

Contohnya, PyTorch digunakan di Tesla untuk model computer vision pada mobil otonom.

Keunggulan PyTorch:

Disukai oleh peneliti machine learning.

Pada Februari 2022, PyTorch adalah framework deep learning paling banyak digunakan di Papers With Code.

Mempercepat eksekusi kode dengan menggunakan GPU.

Bagian: Pengenalan Tensors

Apa itu Tensor?

Tensor adalah blok dasar dalam machine learning yang digunakan untuk merepresentasikan data secara numerik.

Pengantar Tensors:

Tensor bisa merepresentasikan berbagai jenis data, misalnya, gambar bisa direpresentasikan sebagai tensor dengan bentuk [3, 224, 224] (saluran warna, tinggi, lebar).

Tensors memiliki dimensi, seperti tensor gambar yang memiliki tiga dimensi: saluran warna, tinggi, dan lebar.

Membuat Tensors:

Scalar: Tensors dengan satu angka (dimensi nol).

Vector: Tensors satu dimensi yang bisa berisi banyak angka.

Matrix: Tensors dua dimensi yang fleksibel dalam merepresentasikan data.

Tensor: Tensors n-dimensi yang bisa merepresentasikan hampir segala sesuatu.

Manipulasi Tensors:

Mendapatkan informasi dari tensors menggunakan ndim (jumlah dimensi) dan shape (bentuk tensor).

Menangani dimensi, contohnya membuat tensor 3D.

Tensors Acak:

Membuat tensors dengan angka acak menggunakan torch.rand().

Tensors Berisi Nol dan Satu:

Membuat tensors yang berisi nol atau satu menggunakan torch.zeros() dan torch.ones().

Rangkaian dan Tensors Serupa:

Membuat tensor berdasarkan range angka menggunakan torch.arange().

Membuat tensor yang sama dengan tensor lain menggunakan torch.zeros_like() dan torch.ones_like().

Jenis Data Tensors:

Terdapat berbagai jenis data tensor seperti torch.float32, torch.float16, torch.int8, dll.

Precision datatypes berkaitan dengan tingkat detail yang digunakan untuk menggambarkan sebuah angka.

Masalah yang Umum:

Masalah umum termasuk masalah dimensi, jenis data, dan perangkat (device) ketika berinteraksi dengan tensors.

Perangkat dan Dtype:

Perhatikan masalah perangkat dan dtype (tipe data) ketika bekerja dengan tensors, karena PyTorch menyukai konsistensi antara tensors.

Bagian: Mendapatkan Informasi dari Tensors

Atribut Umum:

Tiga atribut umum yang sering digunakan untuk mendapatkan informasi tentang tensors: shape, dtype, dan device.

Bagian: Manipulasi Tensors (Operasi Tensor)

Operasi Dasar:

Operasi dasar melibatkan penambahan, pengurangan, dan perkalian.

Nilai dalam tensor tidak berubah kecuali jika direassign.

Reassign Tensor:

Tensor dapat diubah nilainya dengan mereassign variabel tensor.

Fungsi Built-in:

PyTorch menyediakan fungsi built-in seperti torch.mul() dan torch.add() untuk operasi dasar pada tensors.

Operasi Element-wise:

Operasi element-wise melibatkan pengalihan operasi pada setiap elemen tensor.

Perkalian Matriks:

Perkalian matriks adalah operasi umum dalam machine learning.

PyTorch menyediakan fungsi torch.matmul() untuk melakukan perkalian matriks.

Catatan Penting:

Pemahaman operasi matriks sangat penting dalam deep learning.

Penggunaan fungsi built-in disarankan karena lebih cepat daripada implementasi manual.

Waktu Komputasi:

Penggunaan torch.matmul() lebih cepat daripada mengimplementasikan operasi matriks secara manual dengan loop.

Bagian: Umpan Balik dan Koreksi (Feedback and Corrections)

Umpan Balik:

Tutorial ini memberikan pemahaman yang baik tentang manipulasi tensors di PyTorch.

Penjelasan tentang operasi matriks, penggunaan torch.matmul(), dan visualisasi di http://matrixmultiplication.xyz/ memberikan tambahan konteks yang berguna.

Pengenalan tentang error bentuk (shape errors) dan cara menanggulanginya dengan mentransposisi matriks juga penting untuk dipahami.

Koreksi/Penjelasan Tambahan:

Pada bagian "Change tensor datatype," penjelasan tentang perbedaan precision dalam komputasi bisa lebih diuraikan secara lebih rinci agar lebih mudah dipahami oleh pembaca yang mungkin tidak memiliki latar belakang matematika/komputasi yang kuat.

Penambahan contoh atau penjelasan lebih lanjut pada bagian "Reshaping, stacking, squeezing and unsqueezing" bisa memberikan tambahan wawasan bagi pembaca.

Bagian: Indexing pada Tensors (Pemilihan Data dari Tensors)

Pada PyTorch, kita dapat menggunakan indeks untuk memilih data tertentu dari tensors. Indeks berjalan dari dimensi luar ke dimensi dalam. Misalnya, untuk tensor x dengan bentuk (1, 3, 3), kita dapat menggunakan indeks seperti x[0][0][0] untuk memilih nilai tertentu. Penggunaan ":" memungkinkan kita untuk memilih semua nilai dalam suatu dimensi, dan koma (",") digunakan untuk menambah dimensi.

Contoh:

x[:, 0] akan mengembalikan semua nilai dalam dimensi pertama dan indeks pertama dari dimensi kedua.

x[:, :, 1] akan mengembalikan semua nilai dari dimensi pertama dan kedua, tetapi hanya indeks kedua dari dimensi ketiga.

Indeks pada tensors memungkinkan kita memanipulasi dan memilih data sesuai kebutuhan.

Bagian: Tensors PyTorch & NumPy

PyTorch memudahkan interaksi dengan NumPy, suatu pustaka komputasi numerik populer dalam Python. Terdapat dua metode utama yang dapat digunakan untuk konversi antara NumPy dan PyTorch:

torch.from_numpy(ndarray): Mengubah array NumPy menjadi tensor PyTorch.

torch.Tensor.numpy(): Mengubah tensor PyTorch menjadi array NumPy.

Contoh penggunaan:

torch.from_numpy(array) mengubah array NumPy menjadi tensor PyTorch.

tensor.numpy() mengubah tensor PyTorch menjadi array NumPy.

Perlu diperhatikan bahwa konversi dari NumPy ke PyTorch atau sebaliknya akan mempertahankan tipe data (dtype) asalnya, kecuali diubah secara eksplisit. Jika ingin mengubah tipe data, bisa menggunakan .type() untuk tensor atau mengatur dtype pada array NumPy.

Reproducibility in PyTorch:

Reproducibility adalah kemampuan untuk mendapatkan hasil yang sama atau sangat mirip ketika menjalankan kode pada dua komputer yang berbeda. Dalam konteks pembelajaran mesin dan jaringan saraf, ketidakpastian dapat disebabkan oleh pseudorandomness, yang merupakan hasil dari angka acak yang dihasilkan oleh komputer.

Dalam PyTorch, kita dapat menggunakan torch.manual_seed(seed) untuk mengontrol randomness dan menciptakan hasil yang dapat diulang. Sebagai contoh, dua tensor acak yang dihasilkan dengan seed yang sama akan memiliki nilai yang identik, memungkinkan eksperimen yang dapat diulang.

Menjalankan Tensor pada GPU (dan Meningkatkan Kecepatan Perhitungan)

Algoritma deep learning memerlukan banyak operasi numerik. Secara default, operasi-operasi ini sering dilakukan pada CPU (unit pemrosesan komputer). Namun, terdapat sebuah perangkat keras umum lainnya yang disebut GPU (unit pemrosesan grafis), yang seringkali jauh lebih cepat dalam melakukan jenis operasi tertentu yang diperlukan oleh jaringan saraf (seperti perkalian matriks) dibandingkan dengan CPU.

Komputer mungkin dilengkapi dengan GPU. Jika ya, sebaiknya menggunakan GPU tersebut setiap kali melatih jaringan saraf karena kemungkinan besar itu akan mempercepat waktu pelatihan secara dramatis.

Ada beberapa cara untuk pertama-tama mendapatkan akses ke GPU dan kedua, membuat PyTorch menggunakan GPU.

Mendapatkan GPU:

Google Colab: Mudah, Gratis digunakan, hampir tidak perlu setup, dapat berbagi pekerjaan dengan orang lain dengan mudah melalui tautan. Kekurangannya, tidak menyimpan output data kita, komputasi terbatas, dan terkena batas waktu.

Gunakan milik kita sendiri: Sedang, Jalankan semuanya secara lokal di mesin kita sendiri. Kelemahannya, GPU tidak gratis, membutuhkan biaya awal.

Cloud computing (AWS, GCP, Azure): Sedang-Sulit, Biaya awal kecil, akses ke komputasi yang hampir tak terbatas. Kekurangannya, bisa mahal jika berjalan terus-menerus, memerlukan waktu untuk setup yang tepat.

Mendapatkan PyTorch Berjalan di GPU:

Pastikan GPU dapat diakses menggunakan torch.cuda.is_available(). Jika bernilai True, PyTorch dapat melihat dan menggunakan GPU.

Tentukan perangkat yang akan digunakan (CPU atau GPU) menggunakan torch.device("cuda" if torch.cuda.is_available() else "cpu").

Meletakkan Tensor (dan Model) pada GPU:

Gunakan to(device) pada tensor (dan model) untuk menempatkannya pada perangkat tertentu (CPU atau GPU).

Saat menggunakan to(device), itu mengembalikan salinan tensor tersebut. Untuk menimpanya, berikan kembali hasilnya ke tensor yang sama (reassign).

Mengembalikan Tensor ke CPU:

Gunakan Tensor.cpu() untuk memindahkan tensor kembali ke CPU. Ini diperlukan jika ingin berinteraksi dengan tensor menggunakan NumPy, karena NumPy tidak menggunakan GPU.