Nama : Arfara Yema Samgusdian
NIM : 1103202004
Kelas : TK-44-04

Zero to Mastery Learn PyTorch for Deep Learning

00. PyTorch Fundamentals

PyTorch adalah suatu kerangka kerja (framework) sumber terbuka untuk pembelajaran mesin dan deep learning. Dengan menggunakan PyTorch, pengguna dapat memanipulasi dan memproses data serta mengimplementasikan algoritma pembelajaran mesin menggunakan bahasa pemrograman Python.

PyTorch dapat digunakan untuk berbagai tujuan dalam pengembangan model pembelajaran mesin dan deep learning. Beberapa kegunaan utama PyTorch meliputi pemrosesan dan analisis data, pengembangan model pembelajaran mesin, serta implementasi algoritma deep learning.

Banyak perusahaan teknologi terkemuka di dunia, seperti Meta (Facebook), Tesla, dan Microsoft, serta perusahaan penelitian kecerdasan buatan seperti OpenAI, menggunakan PyTorch untuk mendukung penelitian dan mengimplementasikan pembelajaran mesin dalam produk-produk mereka.

Penggunaan PyTorch dihargai oleh para peneliti dalam bidang pembelajaran mesin. Pada Februari 2022, PyTorch menjadi kerangka kerja pembelajaran mendalam yang paling banyak digunakan di Papers With Code, sebuah situs web yang melacak makalah penelitian pembelajaran mesin dan repositori kode yang terkait dengan mereka.

PyTorch juga membantu mengelola banyak hal, seperti akselerasi GPU (membuat kode berjalan lebih cepat) di balik layar. Ini memungkinkan Anda fokus pada manipulasi data dan penulisan algoritma, sementara PyTorch memastikan agar eksekusi berjalan dengan cepat.

Jika perusahaan-perusahaan seperti Tesla dan Meta (Facebook) menggunakan PyTorch untuk membangun model yang mereka terapkan dalam ratusan aplikasi, menggerakkan ribuan mobil, dan menyampaikan konten kepada miliaran orang, maka jelas bahwa PyTorch mampu dalam pengembangan perangkat lunak.

Tensor

- Tensor adalah struktur data dasar dalam komputasi numerik dan representasi data dalam bentuk multi-dimensional array.
- Dalam konteks deep learning, tensor digunakan untuk menyimpan dan memanipulasi data, seperti gambar, teks, atau bilangan, yang dapat diinterpretasikan sebagai array multi-dimensi.

- Tensors dapat memiliki berbagai dimensi, seperti skalar (0D), vektor (1D), matriks (2D), dan lebih tinggi. Sebagai contoh, matriks 2D dapat merepresentasikan citra piksel atau data tabular.
- Setiap elemen dalam tensor memiliki nilai dan dapat memiliki tipe data tertentu, seperti bilangan bulat atau pecahan.
-

Mendapatkan Informasi dari Tensors
Tensors memiliki tiga atribut umum: shape, dtype, dan device.
- shape: Mendeskripsikan dimensi dari tensor.
- dtype: Menentukan tipe data dari elemen-elemen dalam tensor.
- device: Menunjukkan perangkat di mana tensor disimpan (misalnya, CPU atau GPU).

Manipulasi Tensors (Operasi Tensor):
- Operasi tensor dasar melibatkan penambahan, pengurangan, perkalian element-wise, pembagian, dan perkalian matriks.
- Tensors dapat dibuat dan dimanipulasi menggunakan operator aritmatika standar (+, -, *).
- Operasi pada tensors tidak mengubah tensor asli kecuali secara eksplisit diassign ulang.
- PyTorch menyediakan fungsi bawaan seperti torch.mul() dan torch.add() untuk operasi dasar.

Perkalian Matriks
- Perkalian matriks adalah operasi dasar dalam machine learning dan deep learning.
- PyTorch mengimplementasikan perkalian matriks menggunakan torch.matmul() atau simbol "@" dalam Python.
- Dua aturan kunci untuk perkalian matriks: dimensi dalam harus sesuai, dan hasilnya memiliki bentuk dari dimensi luar.
- Perkalian matriks lebih efisien dilakukan menggunakan fungsi bawaan PyTorch daripada implementasi manual dengan loop for.

Indexing (Seleksi Data dari Tensors)
- Indexing pada PyTorch tensors mirip dengan indexing pada Python lists atau NumPy arrays.
- Menggunakan kurung siku untuk mengakses nilai tensors, dengan urutan dimensi dari luar ke dalam.
- Menggunakan ":" untuk menyatakan "semua nilai dalam dimensi ini" dan koma (,) untuk menambah dimensi lain.
- Contoh: x[:, 0] untuk mendapatkan semua nilai dari dimensi ke-0 dan indeks ke-0 dari dimensi ke-1.
-

PyTorch Tensors & NumPy
- PyTorch memiliki fungsi-fungsi (torch.from_numpy() dan torch.Tensor.numpy()) untuk berinteraksi dengan NumPy arrays. NumPy array dapat dikonversi ke PyTorch

tensor, dan sebaliknya. Penting untuk memperhatikan tipe data saat mengonversi, dan dapat menggunakan type() untuk mengubah tipe data.

Reproducibility (Mengurangi Keseimbangan dari Keacakan)

- Pada deep learning, keacakan sering dimanfaatkan untuk inisialisasi parameter yang akan dioptimalkan.
- Reproducibility penting agar eksperimen dapat diulang dengan hasil yang sama.
- Menggunakan torch.manual_seed(seed) untuk memberikan "rasa" pada keacakan, sehingga hasil dapat direproduksi.

Running Tensors on GPUs (Pemanfaatan GPU untuk Komputasi Cepat)

- GPU (Graphics Processing Unit) seringkali lebih cepat dalam melakukan operasi numerik yang dibutuhkan oleh neural networks daripada CPU.
- Beberapa cara untuk mendapatkan akses ke GPU: Google Colab, penggunaan perangkat sendiri, atau cloud computing (AWS, GCP, Azure).
- torch.cuda.is_available() digunakan untuk memeriksa apakah GPU tersedia.
- Code device agnostic direkomendasikan agar dapat berjalan pada CPU atau GPU.

Putting Tensors on the GPU

- Tensors (dan model) dapat ditempatkan pada GPU dengan menggunakan .to(device), di mana device adalah perangkat yang diinginkan (CPU atau GPU).
- torch.cuda.device_count() dapat digunakan untuk menghitung jumlah GPU yang tersedia.Penting untuk menetapkan tensor kembali jika ingin menyimpan hasilnya: tensor = tensor.to(device).