Nama: Pratyenggo Damar I.P.

NIM: 1103194141

Technical Report: PyTorch Fundamentals

Pendahuluan

ini merangkum langkah-langkah fundamental dalam PyTorch, seperti yang dijelaskan dalam file **00_pytorch_fundamentals.ipynb**. Pendekatan ini memberikan pemahaman mendalam tentang dasar-dasar PyTorch dan bagaimana mengimplementasikannya dalam proses pembelajaran mesin dan deep learning.

00. PyTorch Fundamentals

PyTorch, sebagai kerangka kerja sumber terbuka, memainkan peran kunci dalam pengembangan machine learning dan deep learning. Dengan PyTorch, pengguna dapat memanipulasi data dan mengimplementasikan algoritma pembelajaran mesin dengan bahasa pemrograman Python. Prestasi PyTorch diakui secara global, terutama dalam komunitas penelitian machine learning. Pada Februari 2022, PyTorch menjadi kerangka kerja pembelajaran mendalam yang paling banyak digunakan menurut Papers With Code, sebuah situs yang melacak makalah penelitian dan repositori kode terkait.

PyTorch juga mengatasi aspek teknis seperti akselerasi GPU di belakang layar, meningkatkan kecepatan eksekusi. Dengan demikian, pengguna dapat fokus pada manipulasi data dan algoritma, sementara PyTorch mengoptimalkan proses eksekusi. ika perusahaan-perusahaan seperti Tesla dan Meta (Facebook) mempercayakan PyTorch untuk membangun dan menerapkan model dalam aplikasi mereka yang berjumlah ratusan, mengendalikan ribuan mobil, dan menyampaikan konten kepada miliaran orang, hal ini menunjukkan bahwa PyTorch memainkan peran vital dalam pengembangan perangkat lunak.

- **Tensor**

Tensor, sebagai struktur data dasar dalam komputasi numerik, digunakan untuk merepresentasikan data sebagai array multi-dimensi. Dalam konteks deep learning, tensor berperan penting dalam menyimpan dan memanipulasi data seperti gambar, teks, atau bilangan dalam bentuk array multi-dimensi. Setiap elemen dalam tensor memiliki nilai dan tipe data tertentu, seperti bilangan bulat atau pecahan.

- **Mendapatkan Informasi dari Tensors**

Tensors memiliki tiga atribut umum: shape (bentuk), dtype (tipe data), dan device (perangkat). Shape mendeskripsikan dimensi dari tensor, dtype menentukan tipe data elemen, dan device menunjukkan perangkat di mana tensor disimpan (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 menyediakan 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)**

Dalam 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 seringkali lebih cepat dalam 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)**.

Langkah-langkah dari Fundamentals pada PyTorch

Langkah-langkah dari fundamentals pada PyTorch, seperti yang telah dicoba atau ditelusuri, memberikan pemahaman lebih mendalam. Berikut adalah beberapa pelajaran kunci yang dapat diambil:

1. **Pengenalan Tensors dalam PyTorch:** Tensors bukan hanya array multidimensi, tetapi juga fondasi dari semua komputasi dalam PyTorch. Kemampuan untuk melakukan komputasi pada GPU membuat operasi numerik menjadi sangat cepat.
2. **Operasi pada Tensors:** Operasi tensor mencakup fungsi matematis dasar untuk memanipulasi data. Ini membentuk dasar dari perhitungan yang diperlukan dalam proses pelatihan model machine learning.
3. **Penggunaan Operasi Matrix dan Linear Layer:** Operasi matriks, seperti torch.matmul, dan linear layer (torch.nn.Linear) memberikan kekuatan komputasi yang tinggi untuk model neural networks.
4. **Transformasi Tensors:** Transformasi tensor memungkinkan pengubahan bentuk, dimensi, dan representasi data secara fleksibel, memastikan tensor memiliki bentuk yang sesuai dengan kebutuhan model.
5. **Konversi Antara NumPy Arrays dan Tensors:** Kemampuan untuk mengonversi data antara NumPy arrays dan PyTorch tensors memungkinkan integrasi yang mulus antara berbagai library dalam pengembangan.
6. **Penggunaan GPU:** PyTorch memungkinkan operasi tensor di GPU, meningkatkan kecepatan eksekusi dalam tugas komputasi intensif.
7. **Pengelolaan Randomness dan Reproducibility:** Mengelola randomness penting untuk menjaga reproduktibilitas hasil eksperimen.
8. **Pemeriksaan Perangkat yang Tersedia:** Memahami dan memilih perangkat keras yang tepat untuk komputasi sangat krusial.
9. **Perbandingan Tensors:** Pengecekan validitas dan keakuratan operasi pada data dengan membandingkan nilai di antara tensors menjadi langkah penting dalam pengembangan model.
10. **Mengelola Device Tensors:** Kemampuan untuk memindahkan tensors antara perangkat seperti CPU dan GPU memungkinkan penggunaan sumber daya yang efisien.

Kesimpulan

Penguasaan dasar-dasar PyTorch menjadi kunci dalam pengembangan model machine learning yang efisien dan dapat diandalkan. Melalui langkah-langkah yang telah dijelaskan, pengguna dapat memahami cara memanipulasi data, melakukan operasi tensor, dan mengoptimalkan penggunaan sumber daya seperti GPU. Semua ini merupakan fondasi untuk pembelajaran mesin yang sukses dan pengembangan model deep learning yang canggih.