Nama : Andhika Aria Pratama N

NIM : 1103202121

0 PyTorch Fundamentals

Pytorch

PyTorch adalah perpustakaan pembelajaran mesin sumber terbuka yang digunakan luas untuk pengembangan dan pelatihan model deep learning. PyTorch memberikan grafik komputasi yang fleksibel dan dinamis, mendukung desain, pelatihan, dan implementasi jaringan saraf.

PyTorch adalah framework deep learning yang dapat digunakan untuk mengembangkan dan melatih model neural network. Dengan PyTorch, Anda dapat melakukan tugas seperti klasifikasi gambar, deteksi objek, pemrosesan bahasa alami, dan berbagai aplikasi machine learning lainnya. PyTorch dapat digunakan untuk mengembangkan dan melatih model neural network dalam berbagai aplikasi machine learning, termasuk klasifikasi gambar, deteksi objek, pemrosesan bahasa alami, dan pemodelan data kompleks. PyTorch juga sering digunakan untuk riset dan eksperimen dalam pengembangan algoritma deep learning.

Banyak perusahaan teknologi terbesar di dunia seperti Meta (Facebook), Tesla dan Microsoft serta perusahaan riset kecerdasan buatan seperti OpenAI menggunakan PyTorch untuk mendukung penelitian dan menghadirkan pembelajaran mesin pada produk mereka.

PyTorch menjadi pilihan yang populer di kalangan peneliti machine learning karena beberapa alasan. Salah satunya adalah karena banyak digunakan dalam komunitas penelitian, seperti terlihat dari dominasinya di platform Papers With Code yang melacak makalah penelitian dan kode terkait.

Keunggulan utama PyTorch adalah kemampuannya untuk mengoptimalkan penggunaan GPU, sehingga meningkatkan kecepatan eksekusi kode. Dengan begitu, peneliti dan pengembang dapat fokus pada manipulasi data dan desain algoritma, sementara PyTorch secara efisien mengelola akselerasi perangkat keras.

Dukungan dari perusahaan-perusahaan ternama seperti Tesla dan Meta (Facebook) juga menunjukkan kemampuan PyTorch dalam menghadirkan aplikasi skala besar. Perusahaan-perusahaan ini menggunakan PyTorch untuk membangun dan menerapkan model yang digunakan dalam berbagai aplikasi, menggerakkan kendaraan, dan menyajikan konten kepada jutaan pengguna.

Secara keseluruhan, PyTorch populer karena kehadirannya yang kuat dalam penelitian, kemampuan GPU yang efisien, dan keberhasilannya dalam menggerakkan aplikasi penting bagi perusahaan-perusahaan besar.

- Tensor

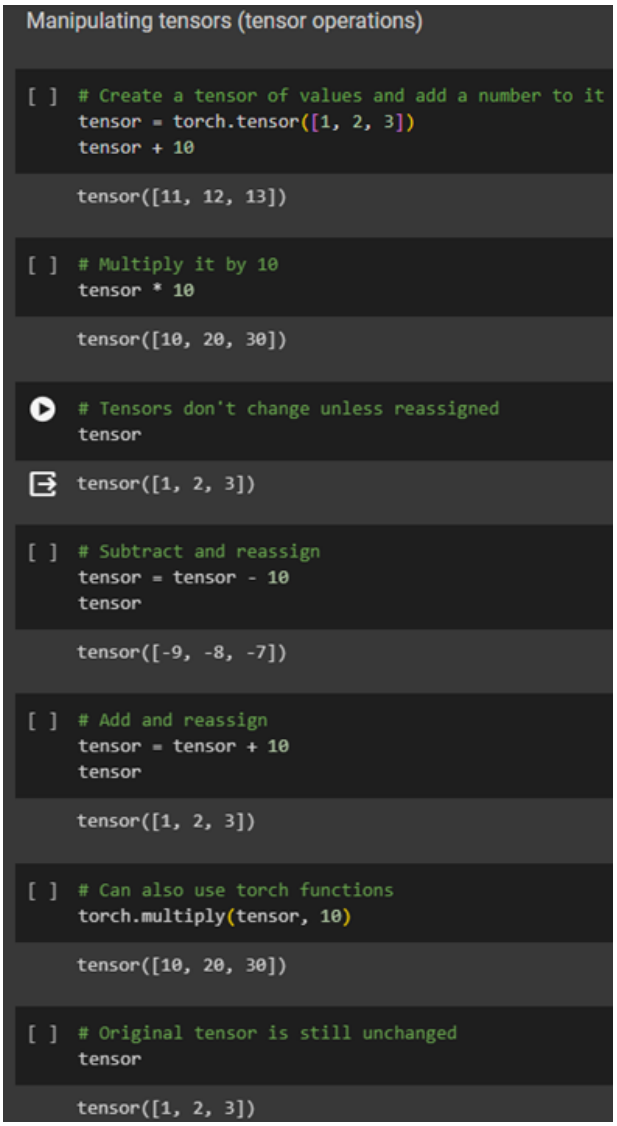
konsep tensor dalam PyTorch, menekankan peran mereka sebagai struktur data fundamental untuk merepresentasikan data numerik dalam machine learning. Tensor dijelaskan sebagai bahan dasar untuk berbagai komputasi, dengan contoh praktis merepresentasikan gambar sebagai tensor dengan dimensi yang sesuai dengan saluran warna, tinggi, dan lebar.

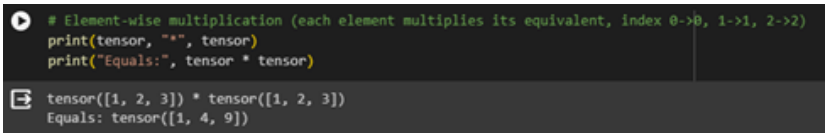
Penjelasan menyoroti pentingnya memahami dimensi tensor dalam konteks PyTorch, karena dimensi tersebut mencerminkan struktur data yang sedang diproses. Tensor dapat menggabungkan beragam informasi numerik, seperti intensitas piksel dalam gambar atau fitur numerik dalam dataset, menjadikannya penting untuk tugas machine learning.

Secara keseluruhan, pemahaman dasar tentang tensor disajikan sebagai krusial untuk memproses dan memanipulasi data dalam kerangka kerja PyTorch, membentuk konsep fundamental bagi praktisi machine learning.

- Manipulasi Tensor
    - Operasi dasar.

Operasi tensor adalah konsep kunci dalam PyTorch, di mana tensor adalah struktur data fundamental yang digunakan untuk menyimpan dan memanipulasi data. Berikut adalah beberapa operasi tensor yang umum dilakukan dalam PyTorch:





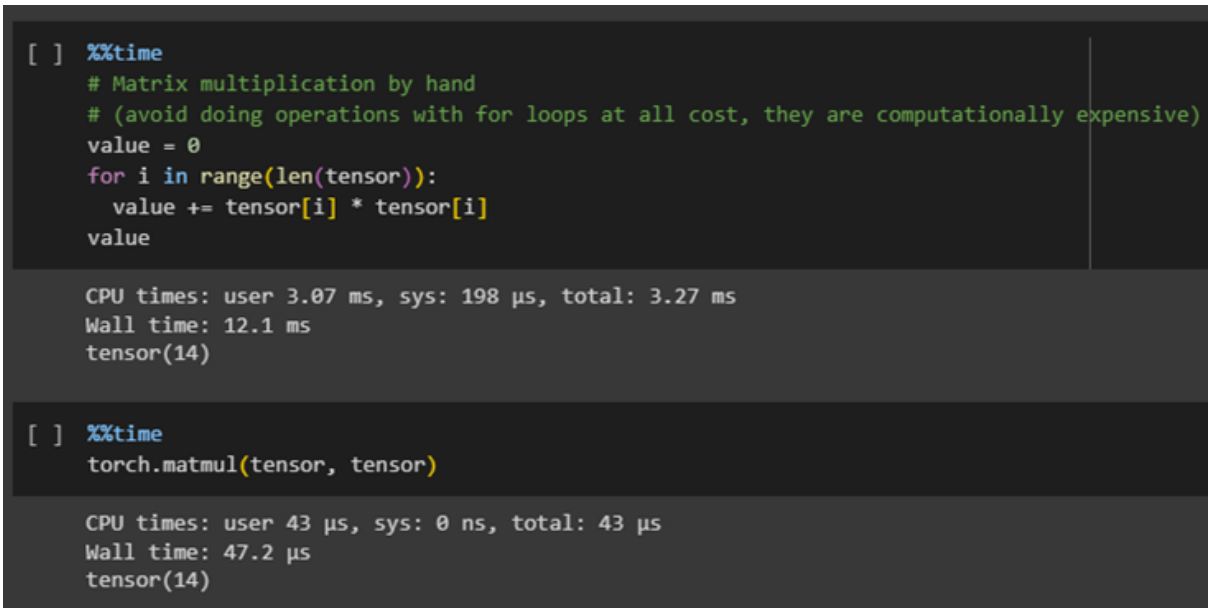
```
# Element-wise multiplication (each element multiplies its equivalent, index 0->0, 1->1, 2->2)
print(tensor, "*", tensor)
print("Equals:", tensor * tensor)
```

```
tensor([1, 2, 3]) * tensor([1, 2, 3])
Equals: tensor([1, 4, 9])
```

Operasi tensor ini memungkinkan pengguna untuk melakukan berbagai manipulasi data dan perhitungan yang diperlukan dalam pengembangan model machine learning menggunakan PyTorch.

- Perkalian matriks

Perkalian matriks pada tensor mengacu pada operasi matematika yang diterapkan pada dua matriks atau tensor untuk menghasilkan matriks atau tensor baru. Dalam dunia tensor PyTorch, perkalian matriks dilakukan menggunakan fungsi torch.mm() atau metode .matmul(). Seperti kode ini :



```
[ ] %%time
    # Matrix multiplication by hand
    # (avoid doing operations with for loops at all cost, they are computationally expensive)
    value = 0
    for i in range(len(tensor)):
      value += tensor[i] * tensor[i]
    value
```

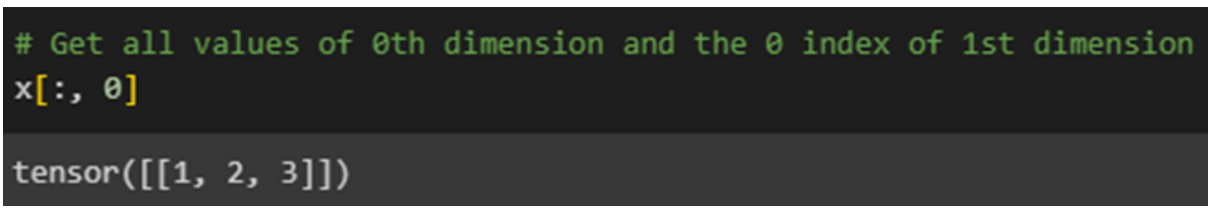
```
CPU times: user 3.07 ms, sys: 198 µs, total: 3.27 ms
Wall time: 12.1 ms
tensor(14)
```

```
[ ] %%time
    torch.matmul(tensor, tensor)
```

```
CPU times: user 43 µs, sys: 0 ns, total: 43 µs
Wall time: 47.2 µs
tensor(14)
```

- Indexing (seleksi data dari tensor)

Indexing pada tensor merujuk pada proses mengakses dan memanipulasi elemen-elemen individu di dalam tensor. Dalam PyTorch, indexing tensor dilakukan dengan sintaksis yang mirip dengan indexing pada list di python. Contohnya :



```
# Get all values of 0th dimension and the 0 index of 1st dimension
x[:, 0]
```

```
tensor([[1, 2, 3]])
```

Gambar diatas merupakan kode untuk mengetahui nilai dari dimensi 0 dalam sebuah index.

```
# Get all values of 0th & 1st dimensions but only index 1 of 2nd dimension
x[:, :, 1]
```

```
tensor([[2, 5, 8]])
```

Gambar diatas memiliki syntax yang berbeda dengan yang lain. Hal ini akan menghasilkan bentuk index dari dimensi ke 0 dan 1. Didapatkan 2 dimensi dari penggunaan tanda ‘:’, tanda ‘:’ diulang 2 kali memiliki arti yaitu 2 dimensi. Setelah terdapat ‘, 1’ yang artinya adalah nilai yang terdapat pada variabel tersebut.

- Tensor dan Numpy

`torch.from_numpy(ndarray)` digunakan untuk mengubah NumPy array menjadi PyTorch tensor. Sebaliknya, `torch.Tensor.numpy()` digunakan untuk mengonversi PyTorch tensor menjadi NumPy array. Kedua fungsi ini memungkinkan perpindahan data antara NumPy dan PyTorch, berguna dalam situasi di mana integrasi atau operasi tertentu lebih optimal dalam salah satu framework tersebut.

- Menjalankan tensor pada GPU

Tensor dapat dijanlankan pada beberap compiler seperti google collab, device, dan cloud computing.

Google Colab: Metode ini mudah diatur dan gratis untuk digunakan. Hampir tidak memerlukan persiapan dan memungkinkan pengguna menjalankan kode pada perangkat keras milik Google. Namun, satu downside utamanya adalah bahwa tidak menyimpan output dan memiliki sumber daya komputasi yang terbatas. Selain itu, bisa terjadi timeout selama komputasi.

Menggunakan GPU sendiri secara lokal: Metode ini melibatkan menjalankan komputasi pada GPU yang Anda miliki atau akses secara lokal. Meskipun memberikan lebih banyak kontrol dan fleksibilitas, GPU tidak gratis dan memerlukan biaya awal. Selain itu, mengatur perangkat lunak yang diperlukan seperti PyTorch mungkin melibatkan prosedur instalasi.

Komputasi awan (AWS, GCP, Azure): Metode ini menawarkan tingkat kesulitan menengah hingga sulit dalam penyiapan, tetapi memungkinkan akses ke sumber daya komputasi yang hampir tak terbatas setelah biaya awal yang kecil. Namun, ini bisa menjadi mahal jika Anda membutuhkan penggunaan yang terus-menerus dan mungkin memerlukan waktu untuk diatur dengan benar. Demikian pula, menginstal framework seperti PyTorch bisa memerlukan mengikuti panduan tertentu.

Tensor pada GPU dijalankan dengan syntax :

Check for GPU

import torch

torch.cuda.is_available()