Nama : Nabilah Salwa

NIM : 1103204060

UAS Machine Learning

00. PyTorch Fundamentals Exercice

1. Create a random tensor with shape (7, 7).

```
# Import torch
import torch

# Create random tensor
X = torch.rand(size=(7, 7))
X, X.shape
```

Kode tersebut merupakan implementasi menggunakan library PyTorch dalam bahasa pemrograman Python. Pertama, perintah `import torch` digunakan untuk mengimpor library PyTorch ke dalam program. Selanjutnya, dengan menggunakan fungsi `torch.rand(size=(7, 7))`, sebuah tensor acak berukuran 7x7 dibuat dan disimpan dalam variabel `X`. Fungsi `torch.rand()` menghasilkan nilai acak antara 0 dan 1 dengan distribusi seragam.

Pada baris terakhir, kita mencetak tensor `X` dan ukurannya menggunakan `X.shape`. Atribut `.shape` memberikan informasi tentang dimensi atau ukuran tensor. Dalam hal ini, `X.shape` memberikan sebuah tuple yang menyatakan bahwa tensor `X` memiliki dimensi 7x7.

Secara keseluruhan, kode tersebut digunakan untuk menghasilkan tensor acak berukuran 7x7 menggunakan PyTorch, yang dapat digunakan untuk keperluan pengolahan data atau operasi matematika lainnya.

2. Perform a matrix multiplication on the tensor from 2 with another random tensor with shape (1, 7) (hint: you may have to transpose the second tensor).

```
# Create another random tensor
Y = torch.rand(size=(1, 7))
# Z = torch.matmul(X, Y) # will error because of shape issues
Z = torch.matmul(X, Y.T) # no error because of transpose
Z, Z.shape
```

Kode tersebut pertama-tama menciptakan tensor acak baru `Y` dengan ukuran 1x7 menggunakan fungsi `torch.rand()`. Selanjutnya, kode mencoba melakukan operasi perkalian matriks (matmul) antara tensor `X` (ukuran 7x7) dan tensor `Y`. Namun, akan terjadi kesalahan karena dimensi tensor tidak kompatibel untuk operasi tersebut.

Untuk mengatasi masalah dimensi, dilakukan operasi transpose pada tensor `Y` menggunakan `Y.T`. Ini mentransposisi tensor `Y` sehingga ukuran baris dan kolomnya saling bertukar, sehingga menjadi 7x1. Dengan menggunakan tensor transposisi ini, operasi perkalian matriks (`torch.matmul(X, Y.T)`) dapat dilakukan tanpa kesalahan.

Hasil dari operasi ini disimpan dalam tensor `Z`, dan selanjutnya, dilakukan pencetakan tensor `Z` bersama dengan ukurannya menggunakan `Z.shape`. Hal ini memberikan informasi mengenai ukuran tensor hasil perkalian matriks. Keseluruhan, kode tersebut menciptakan tensor acak, menangani masalah dimensi dengan mentransposisi tensor, dan melakukan operasi perkalian matriks antara `X` dan `Y` secara berhasil.

3. Set the random seed to 0 and do 2 & 3 over again.

```
# Set manual seed
torch.manual_seed(0)

# Create two random tensors
X = torch.rand(size=(7, 7))
Y = torch.rand(size=(1, 7))

# Matrix multiply tensors
Z = torch.matmul(X, Y.T)
Z, Z.shape
```

Kode di atas dimulai dengan mengatur seed manual menggunakan `torch.manual_seed(0)`. Ini bertujuan untuk memberikan konsistensi dalam pembangkitan angka acak, sehingga hasil yang dihasilkan dapat direproduksi dengan seed yang sama.

Selanjutnya, dibuat dua tensor acak, `X` dan `Y`, dengan ukuran masing-masing 7x7 dan 1x7 menggunakan fungsi `torch.rand()`. Tensor `X` dan `Y` akan digunakan untuk operasi perkalian matriks.

Setelah itu, dilakukan operasi perkalian matriks antara tensor `X` dan tensor transpose dari `Y` (`Y.T`) menggunakan `torch.matmul(X, Y.T)`. Operasi ini dilakukan untuk memastikan bahwa dimensi tensor sesuai untuk perkalian matriks. Hasil perkalian matriks disimpan dalam tensor `Z`.

Terakhir, tensor `Z` dicetak bersama dengan ukurannya menggunakan `Z.shape`. Ini memberikan informasi tentang ukuran tensor hasil perkalian matriks. Keseluruhan, kode ini mengatur seed manual, membuat dua tensor acak, melakukan operasi perkalian matriks, dan mencetak hasil serta ukurannya. Pengaturan seed manual memastikan konsistensi dalam penghasilan angka acak, sehingga hasilnya dapat direproduksi.

4. Speaking of random seeds, we saw how to set it with torch.manual_seed() but is there a GPU equivalent? (hint: you'll need to look into the documentation for torch.cuda for this one)

```
# Set random seed on the GPU
torch.cuda.manual_seed(1234)
```

Kode di atas menggunakan torch.cuda.manual_seed(1234) untuk mengatur seed acak secara manual pada GPU (Graphics Processing Unit). Ini memiliki tujuan yang serupa dengan torch.manual_seed(0), namun di sini fokusnya adalah pada pengaturan seed untuk operasi-operasi yang dilakukan di GPU.

5. Create two random tensors of shape (2, 3) and send them both to the GPU (you'll need access to a GPU for this). Set torch.manual_seed(1234) when creating the tensors (this doesn't have to be the GPU random seed)

```
# Set random seed
torch.manual_seed(1234)

# Check for access to GPU
device = "cuda" if torch.cuda.is_available() else "cpu"
print(f"Device: {device}")

# Create two random tensors on GPU
tensor_A = torch.rand(size=(2,3)).to(device)
tensor_B = torch.rand(size=(2,3)).to(device)
tensor_A, tensor_B
```

Kode tersebut mengatur seed acak, memeriksa ketersediaan GPU, dan membuat dua tensor acak di perangkat yang sesuai. Seed acak diatur dengan `torch.manual_seed(1234)` untuk memastikan reproduktibilitas hasil acak. Kemudian, dilakukan pengecekan ketersediaan GPU, dan perangkat (`device`) ditetapkan sesuai. Dua tensor acak, `tensor_A` dan `tensor_B`, kemudian dibuat dan dipindahkan ke perangkat yang ditentukan. Informasi perangkat dicetak, serta tensor-tensor yang telah dibuat. Keseluruhan kode ini mempersiapkan lingkungan untuk operasi tensor pada GPU jika tersedia, dengan fokus pada pengaturan seed acak dan penanganan perangkat.

6. Perform a matrix multiplication on the tensors you created in 6 (again, you may have to adjust the shapes of one of the tensors).

```
# Perform matmul on tensor_A and tensor_B
# tensor_C = torch.matmul(tensor_A, tensor_B) # won't work because of shape error
tensor_C = torch.matmul(tensor_A, tensor_B.T)
tensor_C, tensor_C.shape
```

Kode di atas berusaha melakukan operasi perkalian matriks (matmul) antara dua tensor, `tensor_A` dan `tensor_B`. Awalnya, terdapat baris yang di-comment (#) karena operasi matmul tersebut akan menghasilkan kesalahan dimensi yang tidak kompatibel. Hal ini dikarenakan bentuk (shape) tensor `tensor_B` yang tidak sesuai untuk operasi matmul langsung dengan `tensor_A`.

Solusi untuk kesalahan tersebut adalah dengan mentransposisi tensor `tensor_B` menggunakan `.T` sehingga dimensinya menjadi sesuai. Baris yang diaktifkan (`tensor_C = torch.matmul(tensor_A, tensor_B.T)`) adalah solusi tersebut, di mana hasil perkalian matriks disimpan dalam tensor `tensor_C`. Tensor `tensor_C` dan ukurannya kemudian dicetak untuk memberikan informasi tentang hasil operasi perkalian matriks.

Secara keseluruhan, kode tersebut mencoba melakukan operasi matmul antara dua tensor, memperbaiki kesalahan dimensi dengan mentransposisi tensor, dan mencetak hasil perkalian matriks beserta ukurannya.

7. Find the maximum and minimum values of the output of 7.

```
# Find max
max = torch.max(tensor_C)

# Find min
min = torch.min(tensor_C)
max, min
```

Kode di atas bertujuan untuk menemukan nilai maksimum dan minimum dalam tensor `tensor_C`, yang merupakan hasil dari operasi perkalian matriks sebelumnya antara `tensor_A` dan `tensor_B.T`. Pertama, dengan menggunakan `torch.max(tensor_C)`, nilai maksimum dari seluruh elemen dalam tensor diidentifikasi dan disimpan dalam variabel `max`. Selanjutnya, menggunakan `torch.min(tensor_C)`, nilai minimum dari seluruh elemen dalam tensor diidentifikasi dan disimpan dalam variabel `min`.

Hasilnya, nilai maksimum dan minimum tersebut dicetak. Dengan demikian, kode tersebut memberikan informasi mengenai nilai ekstrem (maksimum dan minimum) dalam tensor hasil perkalian matriks `tensor_C`.

8. Find the maximum and minimum index values of the output of 7.

```
# Find arg max
arg_max = torch.argmax(tensor_C)

# Find arg min
arg_min = torch.argmin(tensor_C)
arg_max, arg_min
```

Kode di atas bertujuan untuk menemukan indeks dari nilai maksimum (arg max) dan nilai minimum (arg min) dalam tensor `tensor_C`, yang merupakan hasil dari operasi perkalian matriks antara `tensor_A` dan `tensor_B.T`. Menggunakan `torch.argmax(tensor_C)`, indeks dari elemen dengan nilai maksimum diidentifikasi dan disimpan dalam variabel `arg_max`. Sebaliknya, dengan `torch.argmin(tensor_C)`, indeks dari elemen dengan nilai minimum diidentifikasi dan disimpan dalam variabel `arg_min`.

Hasilnya, kedua variabel ini dicetak, memberikan informasi tentang posisi (indeks) di mana nilai maksimum dan minimum ditemukan dalam tensor. Dengan demikian, kode tersebut memberikan informasi tentang lokasi elemen ekstrem dalam tensor hasil perkalian matriks `tensor_C`.

9. Make a random tensor with shape (1, 1, 1, 10) and then create a new tensor with all the 1 dimensions removed to be left with a tensor of shape (10). Set the seed to 7 when you create it and print out the first tensor and it's shape as well as the second tensor and it's shape.

```
# Set seed
torch.manual_seed(7)

# Create random tensor
tensor_D = torch.rand(size=(1, 1, 1, 10))

# Remove single dimensions
tensor_E = tensor_D.squeeze()

# Print out tensors
print(tensor_D, tensor_D.shape)
print(tensor_E, tensor_E.shape)
```

Kode di atas dimulai dengan menetapkan seed acak menggunakan `torch.manual_seed(7)`. Selanjutnya, sebuah tensor acak `tensor_D` dibuat dengan ukuran (1, 1, 1, 10) menggunakan `torch.rand()`. Tensor ini memiliki empat dimensi, dengan dimensi terakhir memiliki panjang 10.

Kemudian, dilakukan operasi `squeeze` pada tensor `tensor_D` dengan `tensor_E = tensor_D.squeeze()`. Operasi ini bertujuan untuk menghapus dimensi yang memiliki panjang 1, sehingga menghasilkan tensor `tensor_E` dengan dimensi yang lebih sederhana.

Kode kemudian mencetak kedua tensor, `tensor_D` dan `tensor_E`, bersama dengan ukurannya menggunakan `print(tensor_D, tensor_D.shape)` dan `print(tensor_E, tensor_E.shape)`. Ini memberikan informasi tentang bentuk (shape) awal tensor `tensor_D` dan bentuk hasil setelah operasi `squeeze` pada `tensor_E`.

Secara keseluruhan, kode tersebut mengilustrasikan pengaturan seed acak, pembuatan tensor acak dengan dimensi tertentu, penggunaan operasi `squeeze` untuk menghapus dimensi yang tidak diperlukan, dan mencetak informasi tentang bentuk tensor sebelum dan setelah operasi tersebut.