Фреймворк PyTorch для разработки искусственных нейронных сетей
Урок 1. Основные понятия и структуры PyTorch
Создайте случайный FloatTensor размера 3x4x5
Выведите его форму (shape)
Приведите его к форме 6x10
Создайте случайный IntTensor размера 10
Умножьте созданный FloatTensor на созданный IntTensor поэлементно
Умножьте тензор матрично на себя, чтобы результат был размерности 6x6
Подсказка: .dot может не подойти
Посчитайте производную функции y = x**3 + z - 75t в точке (1, 0.5, 2)
Создайте единичный тензор размера 5x6
Переведите его в формат numpy Write Python code:

Вот код для выполнения заданий:

```
import torch

import numpy as np


# Creating a random FloatTensor of size 3x4x5

x = torch.randn(3, 4, 5)


# Printing the shape of the tensor

print(x.shape)


# Reshaping the tensor to size 6x10

x = x.view(6, 10)


# Creating a random IntTensor of size 10

y = torch.randint(0, 10, (10,))


# Elementwise multiplication of FloatTensor and IntTensor

z = x * y
```

```
# Matrix multiplication of the tensor with itself

w = torch.matmul(x, x.t())


# Computing the derivative of the function y = x**3 + z - 75t at (1, 0.5, 2)

x = torch.tensor([1.0, 0.5, 2.0], requires_grad=True)

t = torch.tensor(75.0)

z = torch.sum(x ** 3) + t

z.backward()

print(x.grad)


# Creating an identity tensor of size 5x6

a = torch.eye(5, 6)


# Converting to numpy format

b = a.numpy()
```

Обратите внимание, что для расчета производной функции мы установили флаг `requires_grad` на `True` для тензора `x`, чтобы PyTorch мог отслеживать его градиенты. Затем мы вычислили значение функции в точке `(1, 0.5, 2)` и вызвали метод `backward()` на этом значении, чтобы PyTorch автоматически вычислил градиенты в этой точке. Результат выводится с помощью `x.grad`.

Вывод в файл Output.txt:

```
torch.Size([3, 4, 5])

tensor([ 3.0000,  0.7500, 12.0000])
```