SDM5013 Assignment 2

(一) Question

1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1

INPUT

1	0	-1
1	0	-1
1	0	-1

Kernel 1

1	0
0	1

Kernel 2

0	0
0	0

Reference

计算由INPUT经过Kernel 1与Kernel 2卷积层后得到的结果:

$$Result = INPUT * Kernel\,1 * Kernel\,2.$$

并且定义Loss为:

$$Loss = ||Result - Reference||_2^2.$$

通过反向传播求得过程中的梯度。

Answer:

1.1 The result after two kernels

(1) After the first kernel:

Size of result is $(5-3+1)\times(5-3+1)=3\times 3$

1*1+1*0+1*(-1) = 0

1*1+1*0 = 1

1*1 = 1

...

1*1+1*0+1*(-1) = 0

So,the first result1 is

0	1	1
-1	0	1
-1	-1	0

(2) After the second kernel:

Size of result is $(3-2+1)\times(3-2+1)=2\times 2$

1*0+1*0 = 0

1*1+1*1 = 2

-1*1+(-1*1) =-2

1*0+1*0 = 0

So,the second result2 is

0	2
-2	0

(3) Code

```
import torch
from torch import nn
from d2l import torch as d2l
# 定义卷积的函数
def corr2d(X, K):
    """Compute 2D cross-correlation"""
    h, w = K.shape
    Y = torch.zeros((X.shape[0] - h + 1, X.shape[1] - w + 1))
    for i in range(Y.shape[0]):
        for j in range(Y.shape[1]):
            Y[i, j] = (X[i:i+h, j:j+w] * K).sum()
    return Y

# 定义输入矩阵和两个卷积核
input_matrix = torch.tensor([[1,0,0,0,0],
                             [0,1,0,0,0],
                             [0,0,1,0,0],
                             [0,0,0,1,0],
                              [0,0,0,0,1]], dtype=torch.float32, requires_grad=True)
kernel1 = torch.tensor([[1, 0, -1],
                        [1, 0, -1],
                        [1, 0, -1]], dtype=torch.float32, requires_grad=True)

kernel2 = torch.tensor([[1, 0],
                        [0, 1]], dtype=torch.float32, requires_grad=True)

result1 = corr2d(input_matrix, kernel1)
result2 = corr2d(result1, kernel2)

# 定义参考结果
reference = torch.tensor([[0, 0],
                          [0, 0]], dtype=torch.float32)

# 计算 L2 范数损失
loss =(result2 - reference).pow(2).sum()

# 反向传播求梯度
loss.backward()

print("First result1:\n",result1)
print("Second result2:\n",result2)
print("loss:",loss.item())
```

图 1 Screenshot of the results

1.2 Calculation of gradients

(1) Formula

$$Result1 = INPUT * Kernel1$$

$$Result2 = Result1 * Kernel2$$

$$loss = ||Result2 - Reference||_2^2$$

$$\frac{\partial loss}{\partial Kernel} = Convolution(Input\ X, Loss\ gradient \frac{\partial Loss}{\partial O})$$

$$\frac{\partial loss}{\partial X} = Full\ Convolution(180^{\circ} ratated\ Filter\ Kernel, Loss\ gradient \frac{\partial Loss}{\partial O})$$

Note：CNN_Backprop_Recitation_5_F21.pdf & process.pdf for derivation and process

(2) Backpropagation

Loss for the gradient of Kernel2(2×2)

$$\frac{\partial loss}{\partial Kernel2} = Convolution(Ressult1, Loss\ gradient \frac{\partial Loss}{\partial O})$$

8	4
4	8

Loss for the gradient of Kernel1(3×3)

0	-8	0
8	0	-8
0	8	0

(3) Code

```
1. import torch
2. from torch import nn
```

```
from d2l import torch as d2l

# 定义卷积的函数
def corr2d(X, K):
    """Compute 2D cross-correlation"""
    h, w = K.shape
    Y = torch.zeros((X.shape[0] - h + 1, X.shape[1] - w + 1))
    for i in range(Y.shape[0]):
        for j in range(Y.shape[1]):
            Y[i, j] = (X[i:i+h, j:j+w] * K).sum()
    return Y

# 定义输入矩阵和两个卷积核
input_matrix = torch.tensor([[1,0,0,0,0],
                             [0,1,0,0,0],
                             [0,0,1,0,0],
                             [0,0,0,1,0],
                              [0,0,0,0,1]], dtype=torch.float32, requires_grad=True)
kernel1 = torch.tensor([[1, 0, -1],
                        [1, 0, -1],
                        [1, 0, -1]], dtype=torch.float32, requires_grad=True)

kernel2 = torch.tensor([[1, 0],
                        [0, 1]], dtype=torch.float32, requires_grad=True)

result1 = corr2d(input_matrix, kernel1)
result2 = corr2d(result1, kernel2)

# 定义参考结果
reference = torch.tensor([[0, 0],
                          [0, 0]], dtype=torch.float32)

# 计算 L2 范数损失
loss =(result2 - reference).pow(2).sum()

# 反向传播求梯度
loss.backward()

# 输出梯度
print("Gradient of input matrix:")
print(input_matrix.grad)
print("\nGradient of kernel1:")
print(kernel1.grad)
```

```
print("\nGradient of kernel2:")
print(kernel2.grad)
```

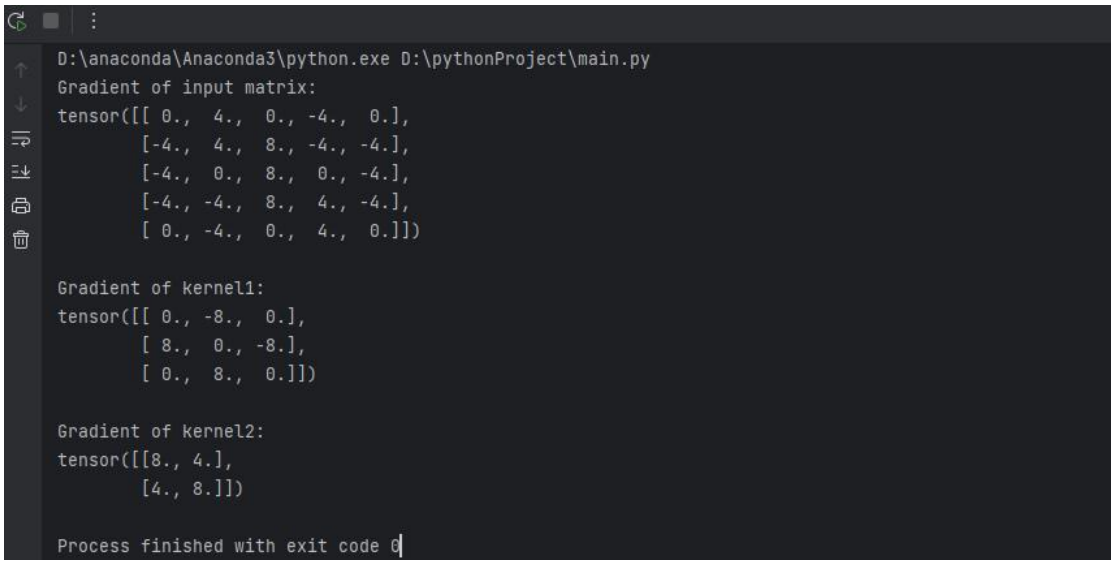

图 2 Screenshot of the results