Вопрос 1

Выполнен

Баллов: 1,00 из 1,00

В последнем утверждении есть некоторая недосказанность.

Мы сказали, что в оригинальной архитектуре LeNet последняя (4-ая) свертка имеет размер 16x5x5 и, соответственно, каждый ее нейрон (которых 120 – по количеству выходов) будет иметь **400 весов**. Что равно количеству весов для нейрона полносвязного слоя в том случае, если мы сперва "растянем" тензор 16x5x5 во входной вектор.

Весов действительно 400, **а сколько всего обучаемых параметров в нейроне предложенной свертки**?

Ответ: 401

Ваш ответ верный.

Вопрос 2

Выполнен

Баллов: 1,00 из 1,00

Сколько чисел содержит тензор, полученный после первого сверточного слоя? Введите одно число

Ответ: 4704

Ваш ответ верный.

Вопрос Инфо

Вопрос 3

Выполнен

Баллов: 1,00 из 1,00

Предположим, мы хотим сконструировать сеть из последовательных блоков вида "свертка + макс-пулинг". Пусть все свертки сохраняют пространственную размерность изображения, т.е. их выход имеет такую же ширину и высоту, как и вход, а в макс-пулинге максимум берется по области размером 2x2.
Известно, что входные изображения имеют разрешение 32x32. Из скольких блоков может состоять сетка? Введите наибольшее возможное число.

Ответ: 5

Ваш ответ верный.

Вопрос Инфо

Вопрос 4

Выполнен

Баллов: 1,00 из 1,00

Пусть C - количество каналов (для RGB - 3), N - количество картинок в батче, W и H - ширина и высота картинки.

Какая размерность должна быть у тензора, который подается на вход конволюционного слоя?

Выберите один ответ:

- a. (H, W, N, C)
- b. (H, W, C, N)
- c. (N, C, H, W)
- d. (C, W, H, N)
- e. (N, C, W, H)
- f. (W, C, H, N)

Ваш ответ верный.

Вопрос 5

Верно

Баллов: 1,00 из 1,00

Подберите размеры паддингов:

Ответ: (штрафной режим: 0 %)

Сброс ответа

```python
import torch

N = 4
C = 3
C_out = 10
H = 8
W = 16

x = torch.ones((N, C, H, W))

# torch.Size([4, 10, 8, 16])
out1 = torch.nn.Conv2d(C, C_out, kernel_size=(3, 3), padding=(1, 1))(x)
#print(out1.shape) # для самопроверки

# torch.Size([4, 10, 8, 16])
out2 = torch.nn.Conv2d(C, C_out, kernel_size=(5, 5), padding=(2, 2))(x)
#print(out2.shape) # для самопроверки

# torch.Size([4, 10, 8, 16])
out3 = torch.nn.Conv2d(C, C_out, kernel_size=(7, 7), padding=(3, 3))(x)
#print(out3.shape) # для самопроверки

# torch.Size([4, 10, 8, 16])
out4 = torch.nn.Conv2d(C, C_out, kernel_size=(9, 9), padding=(4, 4))(x)
#print(out4.shape) # для самопроверки

# torch.Size([4, 10, 8, 16])
out5 = torch.nn.Conv2d(C, C_out, kernel_size=(3, 5), padding=(1, 2))(x)
#print(out5.shape) # для самопроверки

# torch.Size([4, 10, 22, 30])
out6 = torch.nn.Conv2d(C, C_out, kernel_size=(3, 3), padding=(8, 8))(x)
#print(out6.shape) # для самопроверки

#torch.Size([4, 10, 7, 15])
out7 = torch.nn.Conv2d(C, C_out, kernel_size=(4, 4), padding=(1, 1))(x)
#print(out7.shape) # для самопроверки

#torch.Size([4, 10, 9, 17])
out8 = torch.nn.Conv2d(C, C_out, kernel_size=(2, 2), padding=(1, 1))(x)
#print(out8.shape) # для самопроверки
```

Прошли все тесты! ✔

Верно

Баллы за эту попытку: 1,00/1,00.

Вопрос Инфо

Вопрос 6

Выполнен

Баллов: 1,00 из 1,00

В прошлом примере мы делали что-то вроде:

```
test_loss_history.append(loss(test_preds, y_test).cpu())
```

На этот раз код выглядит так:

```
test_loss_history.append(loss(test_preds, y_test).data.cpu())
```

С учётом того, что всё работало, как вы думаете, что было не так?

Выберите один ответ:

- a. мы перепутали loss и accuracy
- b. мы брали с cpu данные, которые на самом деле были на GPU
- c. list заканчивался -- нужно было класть в list, например, каждый 100-й объект
- d. объект, который мы клали в list, хранил весь граф вычислений

Ваш ответ верный.