Вопрос 1

Верно

Баллов: 1,00 из 1,00

В данном шаге вам **требуется реализовать функцию батч-нормализации** без использования стандартной функции со следующими упрощениями:

- Параметр Бета принимается равным 0.
- Параметр Гамма принимается равным 1.
- Функция должна корректно работать только на этапе обучения.
- Вход имеет размерность число элементов в батче * длина каждого инстанса.

Очень внимательно посмотрите на определение функции, вычисляющей std.

Ответ: (штрафной режим: 0 %)

Сброс ответа

```
import numpy as np
import torch
import torch.nn as nn

def custom_batch_norm1d(input_tensor, eps):
    mean = input_tensor.mean(dim=0, keepdim=True)
    variance = input_tensor.var(dim=0, keepdim=True, unbiased=False)

    # Normalize the tensor
    normed_tensor = (input_tensor - mean) / torch.sqrt(variance + eps)
    return normed_tensor


input_tensor = torch.Tensor([[0.0, 0, 1, 0, 2], [0, 1, 1, 0, 10]])
batch_norm = nn.BatchNorm1d(input_tensor.shape[1], affine=False)

# Проверка происходит автоматически вызовом следующего кода
# (раскомментируйте для самостоятельной проверки,
#  в коде для сдачи задания должно быть закомментировано):
# import numpy as np
# all_correct = True
# for eps_power in range(10):
#     eps = np.power(10., -eps_power)
#     batch_norm.eps = eps
#     batch_norm_out = batch_norm(input_tensor)
#     custom_batch_norm_out = custom_batch_norm1d(input_tensor, eps)

#     all_correct &= torch.allclose(batch_norm_out, custom_batch_norm_out)
#     all_correct &= batch_norm_out.shape == custom_batch_norm_out.shape
# print(all_correct)

```

Прошли все тесты! ✔

Верно

Баллы за эту попытку: 1,00/1,00.

Вопрос 2

Верно

Баллов: 1,00 из 1,00

Немного обобщим функцию с предыдущего шага - добавим возможность задавать параметры Бета и Гамма.

На данном шаге вам **требуется реализовать функцию батч-нормализации** без использования стандартной функции со следующими упрощениями:

- Функция должна корректно работать только на этапе обучения.
- Вход имеет размерность число элементов в батче * длина каждого инстанса.

Ответ: (штрафной режим: 0 %)

Сброс ответа

```
import torch
import torch.nn as nn

input_size = 7
batch_size = 5
input_tensor = torch.randn(batch_size, input_size, dtype=torch.float)

eps = 1e-3

def custom_batch_norm1d(input_tensor, weight, bias, eps):
    # Calculate the mean and variance of the input_tensor along the batch dimension
    batch_mean = torch.mean(input_tensor, dim=0)
    batch_var = torch.var(input_tensor, dim=0, unbiased=False)

    # Normalize the input_tensor
    normed_tensor = (input_tensor - batch_mean) / torch.sqrt(batch_var + eps)

    # Reshape weight and bias to have the same shape as the features of input_tensor
    weight = weight.view(1, -1).expand_as(normed_tensor)
    bias = bias.view(1, -1).expand_as(normed_tensor)

    # Scale and shift the normalized tensor using weight (gamma) and bias (beta)
    normed_tensor = weight * normed_tensor + bias
    return normed_tensor

# Проверка происходит автоматически вызовом следующего кода
# (раскомментируйте для самостоятельной проверки,
#  в коде для сдачи задания должно быть закомментировано):
# batch_norm = nn.BatchNorm1d(input_size, eps=eps)
# batch_norm.bias.data = torch.randn(input_size, dtype=torch.float)
# batch_norm.weight.data = torch.randn(input_size, dtype=torch.float)
# batch_norm_out = batch_norm(input_tensor)
# custom_batch_norm_out = custom_batch_norm1d(input_tensor, batch_norm.weight.data, batch_norm.bias.data, eps)
# print(torch.allclose(batch_norm_out, custom_batch_norm_out, 1e-3) \
#       and batch_norm_out.shape == custom_batch_norm_out.shape)
```

Прошли все тесты! ✔

Верно

Баллы за эту попытку: 1,00/1,00.

Вопрос 3

Верно

Баллов: 1,00 из 1,00

Избавимся еще от одного упрощения - реализуем работу слоя батч-нормализации на этапе предсказания.

На этом этапе вместо статистик по батчу будем использовать экспоненциально сглаженные статистики из истории обучения слоя.

В данном шаге вам **требуется реализовать полноценный класс батч-нормализации** без использования стандартной функции, принимающий на вход двумерный тензор. Осторожно, *расчёт дисперсии ведётся по смещенной выборке, а расчет скользящего среднего по несмещенной.*

Ответ: (штрафной режим: 0 %)

Сброс ответа

```
import torch
import torch.nn as nn


input_size = 3
batch_size = 5
eps = 1e-1


class CustomBatchNorm1d:
    def __init__(self, weight, bias, eps, momentum):
        self.weight = weight
        self.bias = bias
        self.eps = eps
        self.momentum = momentum
        self.var = 1
        self.mean = 0
        self.training=True

    def __call__(self, input_tensor):
        if self.training == True:
          var_t=torch.var(input_tensor,dim=0,unbiased=False)
          mean_t=torch.mean(input_tensor,dim=0)

          self.mean=(1-self.momentum)*mean_t + self.momentum*self.mean
          self.var=(1-self.momentum)*var_t*(batch_size/(batch_size-1)) + self.momentum*self.var

          normed_tensor = ((input_tensor - mean_t)/torch.sqrt(var_t+self.eps))*self.weight + self.bias
        else:


          normed_tensor = ((input_tensor - self.mean)/torch.sqrt(self.var+self.eps))*self.weight + self.bias

        return normed_tensor

    def eval(self):
        self.training = False


batch_norm = nn.BatchNorm1d(input_size, eps=eps)
batch_norm.bias.data = torch.randn(input_size, dtype=torch.float)
batch_norm.weight.data = torch.randn(input_size, dtype=torch.float)
batch_norm.momentum = 0.5

custom_batch_norm1d = CustomBatchNorm1d(batch_norm.weight.data,
                                        batch_norm.bias.data, eps, batch_norm.momentum)

# Проверка происходит автоматически вызовом следующего кода
# (раскомментируйте для самостоятельной проверки,
#  в коде для сдачи задания должно быть закомментировано):
# all_correct = True

```

Прошли все тесты! ✔

Верно

Баллы за эту попытку: 1,00/1,00.

Вопрос 4

Верно

Баллов: 1,00 из 1,00

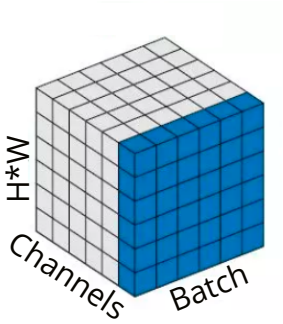
Как вы могли убедиться, реализовать батч-норм слой на этапе предсказания не так просто, поэтому в дальнейших шагах этого урока мы больше не будем требовать реализовать эту часть.

Слой батч-нормализации существует для входа любой размерности.

В данном шаге рассмотрим его для входа из многоканальных двумерных тензоров, например, изображений.

Если вытянуть каждый канал картинки в вектор, то вход будет трехмерным:

- количество картинок в батче
- число каналов в каждой картинке
- число пикселей в картинке



Процесс нормализации:

- Вход разбивается на срезы, параллельные синей части. То есть, каждый срез - это все пиксели всех изображений по одному из каналов.
- Для каждого среза считаются мат. ожидание и дисперсия.
- Каждый срез нормализуется независимо.

На данном шаге вам предлагается **реализовать батч-норм слой для четырехмерного входа** (например, батч из многоканальных двумерных картинок) без использования стандартной реализации со следующими упрощениями:

- Параметр Бета = 0.
- Параметр Гамма = 1.
- Функция должна корректно работать только на этапе обучения.

Ответ: (штрафной режим: 0 %)

Сброс ответа

```
import torch
import torch.nn as nn

eps = 1e-3

input_channels = 3
batch_size = 3
height = 10
width = 10

batch_norm_2d = nn.BatchNorm2d(input_channels, affine=False, eps=eps)

input_tensor = torch.randn(batch_size, input_channels, height, width, dtype=torch.float)


def custom_batch_norm2d(input_tensor, eps):
    mean = input_tensor.mean(dim=(0, 2, 3), keepdim=True)
    var = input_tensor.var(dim=(0, 2, 3), keepdim=True, unbiased=False)

    # Normalize the tensor
    normed_tensor = (input_tensor - mean) / (var + eps).sqrt()
    return normed_tensor


# Проверка происходит автоматически вызовом следующего кода
# (раскомментируйте для самостоятельной проверки,
#  в коде для сдачи задания должно быть закомментировано):
# norm_output = batch_norm_2d(input_tensor)
# custom_output = custom_batch_norm2d(input_tensor, eps)
# print(torch.allclose(norm_output, custom_output) and norm_output.shape == custom_output.shape)

```

Прошли все тесты! ✔

Верно

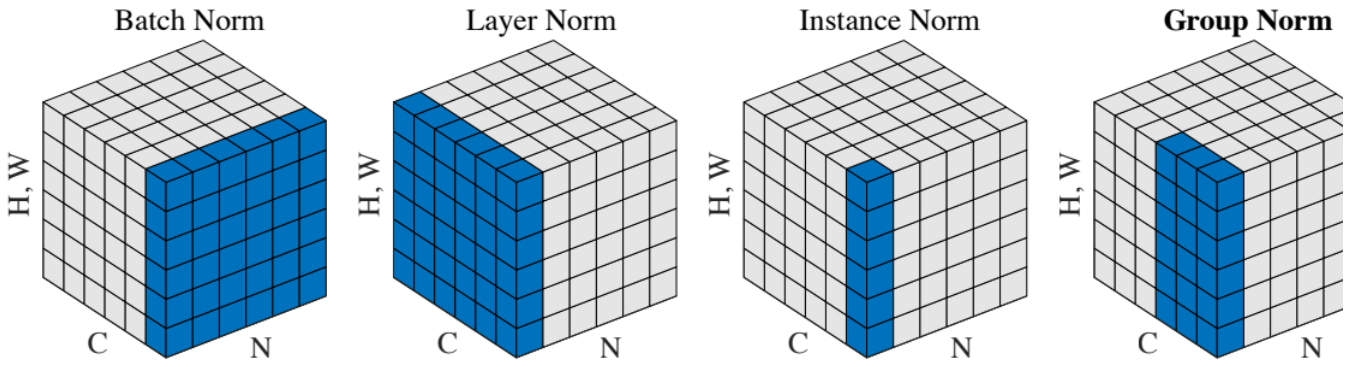
Баллы за эту попытку: 1,00/1,00.

Вопрос Инфо

Мы познакомились поближе с нормализацией "по батчу". Для упрощения дальнейшего изложения остановимся на случае трехмерного тензора на входе слоя, если же размерность входа больше трех, то вытянем все размерности кроме первых двух в одну размерность.

Бывает нормировка не только по батчу, но и по другим измерениям.

Обратите внимание на изображения ниже.



Где:

- C - число каналов на входе.
- N - размер батча.
- H, W - размерность по последней (третьей) размерности входа.

На изображении можно увидеть следующие виды нормализации:

- По батчу.
- По каналу.
- По инстансу.
- По группе.

Кроме указанных видов, также существует множество других, выходящих за рамки нашего урока.

Указанные виды нормализации мы рассмотрим в дальнейших шагах.

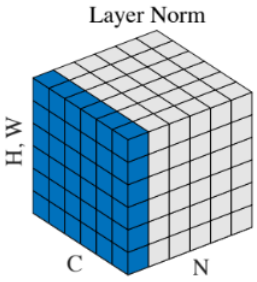
Вопрос 5

Верно

Баллов: 1,00 из 1,00

Идея, лежащая в основе слоя нормализации "по каналу", что сеть должна быть независимой от контраста исходного изображения.

Нормализация "по каналу" работает независимо по каждому изображению батча.



На этом шаге вам предлагается **реализовать нормализацию "по каналу"** без использования стандартного слоя со следующими упрощениями:

- Параметр Бета = 0.
- Параметр Гамма = 1.
- Требуется реализация только этапа обучения.
- Нормализация делается по всем размерностям входа, кроме нулевой.

Обратите внимание, что **размерность входа на данном шаге не фиксирована.**

Уточним, что в слое нормализации "по каналу" **статистики считаются по всем размерностям, кроме нулевой.**

Ответ: (штрафной режим: 0 %)

Сброс ответа

```python
import torch
import torch.nn as nn


eps = 1e-10


def custom_layer_norm(input_tensor, eps):
    # Calculate the mean and variance for each sample in the batch over all dimensions except the first
    mean = input_tensor.mean(dim=tuple(range(1, input_tensor.dim())), keepdim=True)
    var = input_tensor.var(dim=tuple(range(1, input_tensor.dim())), keepdim=True, unbiased=False)

    # Normalize the input tensor
    normed_tensor = (input_tensor - mean) / torch.sqrt(var + eps)

    return normed_tensor


# Проверка происходит автоматически вызовом следующего кода
# (раскомментируйте для самостоятельной проверки,
#  в коде для сдачи задания должно быть закомментировано):
# all_correct = True
# for dim_count in range(3, 9):
#     input_tensor = torch.randn(*list(range(3, dim_count + 2)), dtype=torch.float)
#     layer_norm = nn.LayerNorm(input_tensor.size()[1:], elementwise_affine=False, eps=eps)
#
#     norm_output = layer_norm(input_tensor)
#     custom_output = custom_layer_norm(input_tensor, eps)

#     all_correct &= torch.allclose(norm_output, custom_output, 1e-2)
#     all_correct &= norm_output.shape == custom_output.shape
# print(all_correct)

```

Прошли все тесты! ✔

Верно

Баллы за эту попытку: 1,00/1,00.