МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ордена Трудового Красного Знамени

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский технический университет связи и информатики»

Кафедра «Математическая кибернетика и информационные технологии»

Отчет по лабораторной работе №3

по дисциплине «Системы машинного зрения»

Выполнил:

студент группы БВТ2002

Митрофанов А.О.

Оглавление

[Цель работы 3](#_Toc153003983)

[Задание 3](#_Toc153003984)

[Требования к выполнению работы 3](#_Toc153003985)

[Ход работы 4](#_Toc153003986)

[Вывод 8](#_Toc153003987)

[Приложение 8](#_Toc153003988)

# Цель работы

Разработать алгоритм, используя язык python, реализующий работу операции транспонированной 2d свертки.

# Задание

При выполнении лабораторной работы необходимо:

* ознакомиться с описанием операции библиотеки PyTorch (https://pytorch.org/docs/stable/generated/torch.nn.ConvTranspose2d);
* используя язык программирования Python написать алгоритм, реализующий алгоритм свертки;
* составить отчет по лабораторной работе;

# Требования к выполнению работы

1. Должна быть реализована функция Convolution Transpose. Входные параметры функции должны соответствовать входным параметрам функции ConvTranspose библиотеки PyTorch;
2. Должны быть разработаны группы тестов, для проверки реализованного функционала;
3. Составить отчет по лабораторной работе, включающий краткое описание выполненной работы, ссылки на место хранения разработанного алгоритма и существенные замечания, возникшие в ходе выполнения работ;
4. Доп. Задание: реализовать алгоритм работы транспонированной свертки, через алгоритм двумерной свертки, реализованный в первой лабораторной. Необходимо перерассчитать входные параметры. Выполнение данного задания засчитывается как выполнение лабораторной № 4.

# Ход работы

В ходе лабораторной работы была реализована работа функции для транспонированной свёртки. За основу была взята функция свёртки conv\_transpose2d библиотеки pytorch. Опираясь на документацию библиотеки, была реализована функция conv\_transpose2d, принимающая те же аргументы, что и оригинальная функция: input, weights, padding, output\_padding dilation, stride, groups. Реализация данной функции представлена ниже:

def custom\_conv\_transpose2d(

input: Tensor,

weights: Tensor,

bias: Tensor | None = None,

stride: tuple | int = 1,

padding: int | str | tuple[int, int] = 0,

) :

if len(input.shape) < 3 or len(input.shape) > 4:

raise ValueError(

f"Expected 3D (unbatched) or 4D (batched) input to conv2d, but got input of size: {input.shape}"

)

if len(input.shape) == 3:

input = input.unsqueeze(0)

print(f"unsqueezed input: {input.shape}")

if len(weights.shape) != 4:

raise ValueError(

f"Expected 4D weights with shape [out\_channels, in\_channels/groups, kH, kW], but got weights with shape: {weights.shape}"

)

if bias is not None and bias.shape[0] != weights.shape[1]:

raise ValueError(

f"Expected bias shape to be [out\_channels], but got bias with shape: {bias.shape}"

)

batch\_size, input\_channels, input\_h, input\_w = input.shape

input\_channels, out\_channels, kernel\_h, kernel\_w = weights.shape

if type(stride) is int:

stride = (stride, stride)

if type(padding) is int:

padding = (padding, padding)

z = tuple(map(lambda x: x-1, stride))

p\_ = tuple(map(lambda t: t[0] - t[1] - 1, zip(weights.shape[2:], padding)))

expanded\_h = input\_h + (input\_h-1) \* (z[0])

expanded\_w = input\_w + (input\_w-1) \* (z[1])

expanded = torch.zeros((batch\_size, input\_channels, expanded\_h, expanded\_w))

expanded[::, ::, ::z[0]+1, ::z[1]+1] = input[::, ::, ::, ::]

expanded\_padded = Tensor(np.pad(expanded, ((0,0), (0,0), (p\_[0], p\_[0]), (p\_[1], p\_[1]))))

result = torch.zeros((batch\_size, out\_channels, expanded\_padded.shape[2]-kernel\_h+1, expanded\_padded.shape[3]-kernel\_w+1))

for img\_num, img in enumerate(result):

for ch\_num, \_ in enumerate(img):

for k\_num, \_ in enumerate(expanded\_padded[img\_num:img\_num+1]):

result[img\_num:img\_num+1, ch\_num:ch\_num+1] += custom\_conv2d(

expanded\_padded[img\_num:img\_num+1, k\_num:k\_num+1],

weights[k\_num:k\_num+1, ch\_num:ch\_num+1, ::, ::].flip(3,2),

stride=1,

bias=bias[ch\_num:ch\_num+1],

)

return result

В данной функции были реализованы: padding – параметр, отвечающий за изменение размеров тензора, bias – нейрон смещения, stride – шаг, с которым ядро будет проходить по входному тензору. Параметры dilation и groups не реализованы в полной мере, поэтому функция неполностью повторяет оригинал, что нужно учитывать при использовании.

После чего с помощью библиотеки pytest были реализованы тесты, отвечающие за проверку корректности выполненной работы.

Далее приведены сами тесты:

%%ipytest

def test\_1():

stride = 1

padding = 0

bias = torch.Tensor([1])

inputs = torch.rand((2, 1, 2, 2)).float()

filters = torch.rand((1, 1, 3, 3)).float()

conv = F.conv\_transpose2d(inputs, filters, stride=stride, padding=padding, bias=bias)

my\_conv = custom\_conv\_transpose2d(inputs, filters, stride=stride, padding=padding, bias=bias)

assert torch.allclose(conv, my\_conv)

def test\_2():

stride = 2

padding = 0

bias = torch.Tensor([2])

inputs = torch.rand((2, 1, 3, 3)).float()

filters = torch.rand((1, 1, 5, 5)).float()

conv = F.conv\_transpose2d(inputs, filters, stride=stride, padding=padding, bias=bias)

my\_conv = custom\_conv\_transpose2d(inputs, filters, stride=stride, padding=padding, bias=bias)

assert torch.allclose(conv, my\_conv)

def test\_3():

stride = 3

padding = 1

bias = torch.Tensor([4])

inputs = torch.rand((2, 1, 8, 8)).float()

filters = torch.rand((1, 1, 4, 4)).float()

conv = F.conv\_transpose2d(inputs, filters, stride=stride, padding=padding, bias=bias)

my\_conv = custom\_conv\_transpose2d(inputs, filters, stride=stride, padding=padding, bias=bias)

assert torch.allclose(conv, my\_conv)

Сравнение результатов изначальной функции и функции библиотеки pytorch было проведено с помощью функционала этой же библиотеки, а именно функции allclose, отвечающей за приблизительное сравнение полученных обоими функциями значений, т.к. полная эквивалентность невозможна.

Все 4 реализованных теста прошли успешно, что можно увидеть на рисунке 1.



Рисунок 1 – Результат работы тестов.

# Вывод

В ходе работы была реализована функция транспонированной свёртки, включающая в себя большую часть возможностей оригинала. Кроме того, были проведены тесты с успешным результатом, что означает корректность работы реализованного функционала.

# Приложение

Ссылка на репозиторий гитхаба с реализованным функционалом: <https://github.com/d1agnozzz/CV_labs>