**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»

на тему:

«Разработка нейросетевых функций. Операция Convolution Transpose»

Выполнил:

студент группы БВТ2003

Газизов Камил Ильдарович

Москва, 2023 г.

Оглавление

[Цель работы 3](#_Toc153253044)

[Задание 3](#_Toc153253045)

[Теоретические сведения 4](#_Toc153253046)

[Выполнение работы 5](#_Toc153253047)

[Дополнительное задание 7](#_Toc153253048)

[Заключение 10](#_Toc153253049)

# 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;
* дополнительное задание: реализовать алгоритм работы транспонированной свертки, через алгоритм двумерной свертки, реализованный в первой лабораторной. Необходимо перерассчитать входные параметры.

# Теоретические сведения

Операция транспонированной 2D свертки, также известная как деконволюция или транспонированная свертка, часто используется в нейронных сетях для увеличения разрешения данных. Эта операция обратна обычной 2D свертке и используется, например, при апсемплинге (увеличении размера) изображений.

Основные моменты операции:

1) Цель операции:

- Основная цель - увеличить пространственное разрешение данных. В отличие от обычной свертки, транспонированная свертка увеличивает размер изображения, добавляя пиксели и увеличивая его разрешение.

2) Ядро транспонированной свертки (Transpose Kernel):

- Так же, как и в обычной 2D свертке, транспонированная свертка также использует ядро, которое определяет, каким образом значения входных пикселей влияют на выходные пиксели.

- Веса ядра определяются обратным образом по отношению к обычной свертке.

3) Шаг и заполнение:

- Как и в обычной свертке, транспонированная свертка может иметь шаг и заполнение (padding).

- Шаг определяет, насколько далеко сдвигается ядро при каждой операции.

- Заполнение добавляет нулевые значения вокруг входных данных, чтобы сохранить размер.

4) Размер выхода:

- Размер выходного изображения определяется размерами входного изображения, размером ядра, шагом и заполнением.

- Формула для вычисления размера выхода аналогична формуле для обычной свертки.

5) Пример использования:

- Транспонированные свертки могут использоваться, например, в генеративных моделях, таких как Generative Adversarial Networks (GANs), для генерации изображений высокого разрешения.

# Выполнение работы

В моем коде функция conv\_transpose2d представляет собой реализацию операции транспонированной 2D свертки.

Входные параметры:

- input: Трехмерный тензор входных данных, представляющий собой батч объемных данных.

- weight: Трехмерный тензор весов (ядра транспонированной свертки) с размерностью [out\_channels, in\_channels, kernel\_size[0], kernel\_size[1]].

- bias: Массив смещений для каждого канала выходных данных.

- in\_channels: Количество входных каналов.

- out\_channels: Количество выходных каналов.

- kernel\_size: Размер ядра транспонированной свертки в двух измерениях.

- stride: Шаг транспонированной свертки (по умолчанию 1).

- padding: Заполнение вокруг входных данных (по умолчанию 0).

- output\_padding: Заполнение вокруг выходных данных (по умолчанию 0).

- dilation: Разреживание (по умолчанию 1).

- bias\_enabled: Флаг, указывающий, следует ли добавлять смещение (по умолчанию True).

- padding\_mode: Режим заполнения ('zeros' по умолчанию, также может быть 'replicate').

Определение размерности выходных данных:

- Рассчитываются размеры выходных данных (h\_out, w\_out) с учетом размеров входных данных, ядра транспонированной свертки, шага, заполнения, разреживания и заполнения выходных данных.

Страйдированный вход:

- Применяется функция stride\_input, которая уменьшает размер входных данных в соответствии с шагом транспонированной свертки.

Итерация по батчам и выходным каналам:

- Происходит итерация по батчам и каналам выходных данных.

- Создается пустой массив для выходных данных (out).

Итерация по выходным пикселям:

- Происходит итерация по координатам выходных данных (y\_out, x\_out).

- Для каждого выходного пикселя рассчитывается взвешенная сумма с учетом весов ядра, входных данных, шага, заполнения, разреживания и заполнения выходных данных.

Обработка граничных условий:

- Проверяются условия для гарантии корректного доступа к пикселям входных данных.

- Если используется режим 'replicate', то при выходе за границы входных данных используются значения на границах.

Добавление смещения:

- Если флаг bias\_enabled установлен в True, добавляется смещение для каждого канала выходных данных.

Формирование результата:

- Результаты сохраняются в выходной массив.

Возврат результата:

- Возвращается трехмерный тензор выходных данных.

Изображение выглядит как текст, снимок экрана, Шрифт, документ

Автоматически созданное описание

Рисунок 1 – Реализация функции conv\_transpose2d

Метод stride\_input выполняет операцию стратификации (или сокращения размерности) входного тензора с использованием заданного шага (stride).

Входные параметры:

- input: Трехмерный тензор входных данных, представляющий собой батч объемных данных.

- stride: Кортеж шагов для каждой размерности (в данном случае, для высоты и ширины).

Размерность входных данных:

- Получаются размеры каналов, высоты и ширины входного тензора (channels, rows, cols).

Размерность выходных данных:

- Рассчитываются размеры выходного тензора (out\_rows, out\_cols) с учетом шага транспонированной свертки.

Инициализация выходного массива:

- Создается массив нулей для выходных данных.

Итерация по выходным пикселям:

- Происходит итерация по координатам выходных данных (y\_out, x\_out) с шагом stride.

- Для каждого выходного пикселя определяются соответствующие координаты входного тензора (y, x), и значение копируется из входного тензора в выходной.

Формирование результата:

- Результаты сохраняются в выходной массив.

Возврат результата:

- Возвращается трехмерный тензор выходных данных.

Этот метод полезен для уменьшения размерности входных данных, применяемого перед операцией транспонированной свертки.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 2 – Реализация метода stride\_input

Метод default\_data предназначен для создания некоторых тестовых данных для операции свертки.

Входные данные (input):

- Создаются два канала (channel\_1 и channel\_2) с матрицами 4x4.

- Создается входной тензор input с формой (1, 2, 4, 4), представляющий собой батч из одного изображения с двумя каналами.

Веса (weight):

- Создаются два ядра (weight\_1 и weight\_2) размером 2x2 с единичными значениями.

- Создается тензор весов weight с формой (2, 2, 2, 2), представляющий собой два ядра для двух каналов входных данных.

Смещение (bias):

- Создается массив смещений bias с двумя значениями (0, 0) по умолчанию. Если передан параметр bias, используется переданное значение.

Возвращаемые значения:

- Возвращаются три объекта: входной тензор input, тензор весов weight и массив смещений bias.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 3 – Реализация метода default\_data

Метод assert\_equal\_own\_and\_torch предназначен для сравнения результатов собственной реализации операции свертки с результатами, полученными с использованием библиотеки PyTorch.

Преобразование весов и смещения PyTorch:

- Преобразуются веса и смещения PyTorch, используя данные, переданные в метод, и приводя их к типу float.

Вычисление результатов:

- Вычисляются результаты свертки собственной реализации и для PyTorch.

- Результаты округляются вниз с использованием np.floor

Вывод результатов:

- Выводятся результаты свертки собственной реализации и PyTorch.

Проверка равенства результатов:

- Используется assert\_array\_equal из библиотеки NumPy для проверки того, что результаты свертки вашей собственной реализации и PyTorch равны.

- Если результаты не равны, будет вызвано исключение AssertionError.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 4 – Реализация метода assert\_equal\_own\_and\_torch

Метод test предназначен для сравнения результатов нашей реализации с PyTorch при различных данных.

Изображение выглядит как текст, снимок экрана, документ, Шрифт

Автоматически созданное описание

Рисунок 5 – Реализация метода test (часть 1)

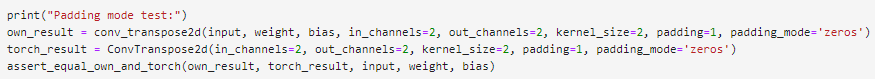


Рисунок 6 – Реализация метода test (часть 2)

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 7 – Вывод результатов сравнения

# Выполнение дополнительного задания

В методе conv2d\_extra реализуется операция транспонированной свертки с использованием алгоритма двумерной свертки.

Входные параметры:

- batch\_size, in\_channels, h\_in, w\_in: Параметры, описывающие размеры входных данных.

- out\_channels, kernel\_size, stride, padding, dilation: Параметры, определяющие транспонированную свертку.

- bias\_enabled: Флаг, указывающий, включено ли использование смещения (bias).

- padding\_mode: Режим заполнения при транспонированной свертке.

Вычисление размеров выходного тензора:

- out\_height и out\_width: Размеры выходного тензора вычисляются на основе размеров входного тензора, ядра, шага, заполнения и диляции.

Инициализация выходного тензора:

- output: Инициализируется нулевым тензором с размерами batch\_size, out\_channels, out\_height, out\_width.

Инверсия ядра:

- weight\_flipped: Ядро инвертируется в обеих размерностях.

Цикл по батчам, выходным каналам и входным каналам:

- Внутренний цикл выполняется по каждому батчу, выходному каналу и входному каналу.

- Вызывается функция conv2d для выполнения свертки с соответствующими входными данными и ядром.

Обновление выходного тензора:

- Результат свертки добавляется к выходному тензору, применяется диляция и шаг.

Добавление смещения (bias):

- Если предоставлено смещение, оно добавляется к выходному тензору.

Возвращение результата:

- Возвращается полученный выходной тензор

Изображение выглядит как текст, снимок экрана, Шрифт, документ

Автоматически созданное описание

Рисунок 8 – Реализация функции и проверка результата

Изображение выглядит как текст, снимок экрана, Шрифт, документ

Автоматически созданное описание

Рисунок 11 – Функция conv2d с небольшими изменениями для корректной работы новой функции

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 10 – Вывод результатов сравнения

# Заключение

В ходе выполнения данной лабораторной работы был разработан алгоритм на языке Python, который реализует операцию двумерной свертки. Также были созданы тесты для проверки корректности работы алгоритма. Также был разработан метод, реализующий операцию транспонированной свертки с использованием алгоритма двумерной свертки. Как можем видеть, результаты абсолютно идентичны.

Ссылка на git-репозиторий: https://github.com/ogkama/SMZ