**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 №1

по дисциплине

«Системы машинного зрения»

на тему:

«Разработка нейросетевых функций. Операция Convolution 2D»

Выполнил:

студент группы БВТ2003

Аджиумеров Александр

Москва, 2023 г.

Оглавление

[Цель работы 3](#_Toc153249758)

[Задание 3](#_Toc153249759)

[Теоретические сведения 3](#_Toc153249760)

[Выполнение работы 5](#_Toc153249761)

[Заключение 7](#_Toc153249762)

# Цель работы

Разработать алгоритм, используя язык python, реализующий работу операции двумерной свертки.

# Задание

При выполнении лабораторной работы необходимо:

* ознакомиться с описанием операции библиотеки PyTorch (<https://pytorch.org/docs/stable/generated/torch.nn.Conv2d>);
* используя язык программирования Python написать алгоритм, реализующий алгоритм свертки;
* составить отчет по лабораторной работе

# Теоретические сведения

Операция двумерной свертки играет важную роль в обработке изображений. Она используется для выделения различных признаков, таких как грани, текстуры и другие локальные паттерны, что позволяет сети автоматически извлекать иерархические признаки из входных данных.

Шаги двумерной свертки:

1. Веса (Weights):

- Каждое ядро свертки имеет связанные с ними веса (или параметры), которые обучаются в процессе обучения сети.

- Веса ядра представляют собой фильтры, которые определяют, какие признаки извлекаются из входного изображения.

1. Свертка (Convolution):

- Для каждого фрагмента входного изображения, называемого рецептивным полем, применяется свертка с соответствующим ядром.

- В процессе свертки вычисляется взвешенная сумма значений пикселей в рецептивном поле с использованием весов ядра.

1. Активация (Activation):

- Результат свертки проходит через функцию активации (например, ReLU - Rectified Linear Unit), чтобы внести нелинейность в модель и позволяет сети лучше моделировать сложные зависимости в данных.

1. Шаг перемещения (Stride):

- Определяет, насколько смещается ядро при применении свертки к входному изображению.

- Шаг контролирует размер выходного слоя: меньший шаг приводит к уменьшению размера, что может быть полезно для снижения вычислительной сложности.

1. Заполнение (Padding):

- Добавление нулей вокруг входного изображения перед сверткой.

- Заполнение может использоваться для того, чтобы сохранить размер выходного изображения и предотвратить уменьшение размерности.

6) Многоканальные входы и выходы:

- Входные и выходные тензоры могут иметь несколько каналов. Например, для цветных изображений это могут быть каналы RGB.

- Каждый канал имеет свои собственные веса и свертка применяется отдельно для каждого канала.

Эти операции повторяются для каждого ядра в слое свертки, создавая выходной тензор с различными картами признаков. После слоя свертки часто следует слой пулинга (Pooling), который уменьшает размерность выходных карт признаков, снижая количество параметров и вычислительную сложность. Операция свертки позволяет выделять различные характеристики в изображении, такие как грани, углы и текстуры.

# Выполнение работы

В моем коде функция conv2d реализует операцию двумерной свертки с учетом заданных параметров.

Входные параметры:

- input: Тензор входных данных, представляющий собой батч изображений.

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

- bias: Массив смещений для каждого канала выходных данных.

- in\_channels: Количество каналов во входных данных.

- out\_channels: Количество каналов в выходных данных.

- kernel\_size: Размер ядра свертки.

- stride: Шаг свертки (по умолчанию 1).

- padding: Заполнение вокруг входных 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 и w\_out) с учетом размеров входных данных, ядра свертки, шага, заполнения и разреживания.

Итерация по батчам и выходным каналам:

- Происходит итерация по батчам (b) и каналам выходных данных (c\_out).

- Создается пустой массив для выходных данных (out).

Итерация по выходным пикселям:

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

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

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.

Изображение выглядит как текст, снимок экрана, Шрифт, документ

Автоматически созданное описание

Рисунок 1 – Реализация функции conv2d

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, число

Автоматически созданное описание

Рисунок 2 – 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 3 – 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, Шрифт, документ

Автоматически созданное описание

Рисунок 4 – Реализация метода test (часть 1)

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 5 – Реализация метода test (часть 2)

Изображение выглядит как текст, число, Шрифт, снимок экрана

Автоматически созданное описание

Рисунок 6 – 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.

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