**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 №2

по дисциплине

«Системы машинного зрения»

на тему:

«Разработка нейросетевых функций. Операция Convolution 3D»

Выполнил:

студент группы БВТ2003

Аджиумеров Александр

Москва, 2023 г.

Оглавление

[Цель работы 3](#_Toc153250682)

[Задание 3](#_Toc153250683)

[Теоретические сведения 3](#_Toc153250684)

[Выполнение работы 5](#_Toc153250685)

[Заключение 8](#_Toc153250686)

# Цель работы

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

# Задание

При выполнении лабораторной работы необходимо:

* ознакомиться с описанием операции библиотеки PyTorch (<https://pytorch.org/docs/stable/generated/torch.nn.Conv3d>);
* используя язык программирования Python написать алгоритм, реализующий алгоритм свертки;
* составить отчет по лабораторной работе

# Теоретические сведения

Операция трехмерной свертки применяется к объемам данных трехмерных тензоров, таких как видео, медицинские изображения, или 3D-модели. Эта операция расширяет идею двумерной свертки на третье измерение и позволяет модели обрабатывать объемные пространственные данные.

Основные моменты:

1) Вводные данные (Input):

- Трехмерный тензор представляет собой объем данных с тремя измерениями: высота, ширина и глубина (также называемая каналом или глубиной кадра).

- Например, для видео входной тензор может иметь размерность [количество кадров, высота, ширина, глубина].

2) Ядро свертки (Kernel):

- Трехмерное ядро свертки также представляет собой объем с тремя измерениями.

- Каждый элемент ядра является весом, который умножается на соответствующий элемент в объеме входных данных.

3) Свертка (Convolution):

- Производится перемножение элементов трехмерного ядра свертки с соответствующими элементами объема входных данных.

- Результат каждого умножения суммируется.

4) Шаг перемещения (Stride):

- Определяет, насколько смещается ядро при применении свертки к объему входных данных.

5) Заполнение (Padding):

- Аналогично двумерной свертке, можно применять заполнение для контроля размера выходного объема.

6) Активация (Activation):

- Результат свертки часто подвергается функции активации для введения нелинейности.

7) Выход (Output):

- Результат свертки создает новый объем данных (трехмерный тензор) с выделенными пространственными признаками.

Операция трехмерной свертки активно используется в архитектурах нейронных сетей для обработки трехмерных данных, таких как 3D-свеженные кадры видео, в медицинских 3D-изображениях, а также в других задачах, где трехмерная структура данных имеет смысл.

# Выполнение работы

В моем коде функция conv3d реализует операцию двумерной свертки с учетом заданных параметров.

Входные параметры:

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

- weight: Трехмерный тензор весов (ядра свертки) с размерностью [out\_channels, in\_channels, kernel\_size[0], kernel\_size[1], kernel\_size[2]].

- bias: Массив смещений для каждого канала выходных данных.

- in\_channels: Количество входных каналов.

- out\_channels: Количество выходных каналов.

- kernel\_size: Размер ядра свертки в трех измерениях.

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

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 (z\_out, y\_out, 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.

Возврат результата:

- Возвращается трехмерный тензор выходных данных.

Изображение выглядит как текст, снимок экрана, документ, Шрифт

Автоматически созданное описание

Рисунок 1 – Реализация метода conv3d

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

Входные данные (input):

- Создаются два канала (channel\_1 и channel\_2) с трехмерными массивами 4x4x4.

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

Веса (weight):

- Создаются два трехмерных ядра (weight\_1 и weight\_2) размером 2x2x2x2 с единичными значениями.

- Создается трехмерный тензор весов weight с формой (2, 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: трехмерный входной тензор input, трехмерный 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, документ, Шрифт

Автоматически созданное описание

Рисунок 4 – Реализация метода test (часть 1)

Изображение выглядит как текст, снимок экрана, Шрифт, документ

Автоматически созданное описание

Рисунок 4 – Реализация метода test (часть 2)

Изображение выглядит как текст, 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 трехмерной свертки. Также были созданы тесты для проверки корректности работы алгоритма.

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