ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

***«*САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО»**

Институт компьютерных наук и технологий

**Высшая школа программной инженерии**

Изображение выглядит как текст

Автоматически созданное описание

**ЛАБОРАТОРНАЯ РАБОТА №3**

по дисциплине «Проектирование интеллектуальных систем управления»

Студент А. М. Потапова

гр. 3530202/90202

Руководитель Ю. Н. Кожубаев

Санкт-Петербург

2022 г

**Ход работы**

В качестве примера мною было выбрано создание простой сети глубокого обучения для классификации.

В этом примере показано, как создать и обучить простую сврточную нейронную сеть для классификации глубокого обучения. Сверточные нейронные сети являются важными инструментами для глубокого обучения и особенно подходят для распознавания изображений.

* **Загрузка и изучение данных изображения.**

Шаг 1

Загрузим данные образца цифры как хранилище данных изображения. ImageDatastore автоматически помечает изображения на основе имен папок и сохраняет данные как объект ImageDatastore. Хранилище данных изображений позволяет хранить большие данные изображений, в том числе данные, которые не помещаются в памяти, и эффективно считывать пакеты изображений во время обучения сверточной нейронной сети.

Изображение выглядит как текст

Автоматически созданное описание

Отобразим некоторые изображения в хранилище данных.

Изображение выглядит как текст

Автоматически созданное описание

*Результат*

*Изображение выглядит как текст

Автоматически созданное описание*

Шаг 2

Подсчитаем количество изображений в каждой категории. labelCount — это таблица, содержащая метки и количество изображений с каждой меткой. Хранилище данных содержит 1000 изображений для каждой из цифр 0–9, всего 10000 изображений. Мы можем указать количество классов в последнем полносвязном слое нашей сети в качестве аргумента OutputSize.



*Результат*

Изображение выглядит как стол

Автоматически созданное описание

Шаг 3

Укажем размер изображений во входном слое сети и проверим размер первого изображения в digitData.

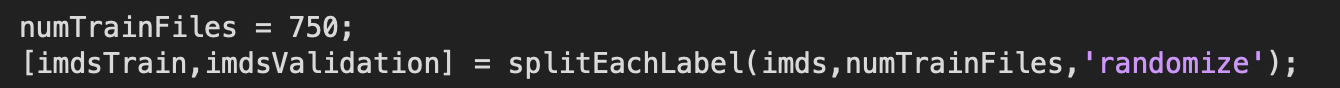
Изображение выглядит как текст

Автоматически созданное описание

Получили, что каждое изображение имеет размер 28 на 28 на 1 пиксель.

* **Определение обучающих и проверочных наборов**

Разделим данные на обучающие и проверочные наборы данных, чтобы каждая категория в обучающем наборе содержала 750 изображений, а проверочный набор содержал оставшиеся изображения с каждой метки. splitEachLabel разделяет хранилище данных digitData на два новых хранилища данных, trainDigitData и valDigitData.



* **Определение сетевой архитектуры**

Изображение выглядит как текст

Автоматически созданное описание

**Определение параметров обучения**

После определения структуры сети укажем параметры обучения. Обучим сеть, используя стохастический градиентный спуск с импульсом (SGDM) с начальной скоростью обучения 0,01. Установим максимальное количество эпох равным 4. Эпоха — это полный цикл обучения на всем наборе обучающих данных. Проконтролируем точность сети во время обучения, указав данные проверки и частоту проверки. Перетасуем данные каждую эпоху. Включим график прогресса обучения и отключим вывод командного окна.

Изображение выглядит как текст

Автоматически созданное описание

**Обучение сети используя обучающие данные**

Обучим сеть, используя архитектуру, определенную слоями, обучающими данными и вариантами обучения. По умолчанию trainNetwork использует графический процессор, если он доступен, в противном случае он использует ЦП. Мы также можем указать среду выполнения, используя аргумент пары "имя-значение" ExecutionEnvironment для trainingOptions. На графике хода обучения показаны потери и точность мини-пакетов, а также потери и точность проверки. Потеря — это кросс-энтропийная потеря. Точность — это процент изображений, которые сеть правильно классифицирует.

**

*РезультатИзображение выглядит как стол

Автоматически созданное описание*

**Классифицирование проверочных изображений и вычисление точности**

Предскажем метки данных проверки, используя обученную сеть, и рассчитаем окончательную точность проверки. Точность — это доля меток, которые сеть предсказывает правильно.

Изображение выглядит как текст

Автоматически созданное описание

*Результат*



**Вывод**

В ходе данной работы мне удалось создать простую сети глубокого обучения для классификации. В моем случае более 99% предсказанных меток совпадают с истинными метками проверочного набора.