Санкт-Петербургский политехнический университет Петра Великого

Институт машиностроения, материалов и транспорта

Высшая школа автоматизации и робототехники

Отчёт

по курсовой работе

Дисциплина: Математические методы интеллектуальных технологий

Тема: CNN

Студент гр. 3341506/30401 Вестников Р. Н.

Преподаватель Ананьевский М. С.

Санкт-Петербург

2023

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc99220525)

[Описание алгоритма 3](#_Toc99220526)

[Исследование алгоритма 9](#_Toc99220527)

[Заключение 13](#_Toc99220528)

[Список литературы 14](#_Toc99220529)

[ПРИЛОЖЕНИЕ 15](#_Toc99220530)

# ВВЕДЕНИЕ

Распознавание рукописных цифр имеет фундаментальное значение в области машинного обучения и искусственного интеллекта. Оно находит широкое применение во многих областях, начиная от автоматической почты, которая сортирует почтовые индексы, и заканчивая банковскими системами, которые читают рукописные чеки. Однако одной из самых известных и широко используемых задач в этой области является распознавание рукописных цифр от 0 до 9.

В данном отчёте мы подробно рассмотрим архитектуру свёрточной нейронной сети (CNN), обученной на наборе данных MNIST, который стал краеугольным камнем в исследованиях компьютерного зрения и машинного обучения. Набор данных MNIST был разработан для тестирования и сравнения различных методов машинного обучения и стал эталоном для задачи распознавания рукописных цифр.

**Задача**: обучить модель CNN на MNIST данных, получить точность ≥98%, при числе параметров < 10 000.

**Архитектура модели**:

Модель составлялась при помощи фреймворка PyTorch, и состоит из трёх свёрточных и одного полносвязного слоя. Общее число параметров 8890, что соответствует задаче.

**Обучение модели**:

Модель была обучена на тренировочном наборе данных MNIST, состоящем из 60 000 изображений, и протестирована на валидационном наборе данных, состоящем из 10 000 изображений.

Модель обучалась 20 эпох, с оптимизатором Adam (lr = 0.0001), лосс функцией CrossEntropyLoss, размер батча 512

Конечный лосс при обучении равен 0.004, а точность на тестовых данных 99.04%, что также соответствует задаче.

# Описание модели

Для решения задачи распознавания рукописных цифр на наборе данных MNIST была разработана следующая архитектура CNN:

*Слой 1: Свёрточный слой*:

• Тип: Свёрточный слой (nn.Conv2d)

• Входные данные: Черно-белые изображения с одним каналом (1), размером 28x28 пикселей.

• Количество фильтров: 8

• Размер фильтров: 3x3

• Заполнение (padding): 1 пиксель.

• Функция активации: ReLU

Слой 1 выполняет операцию свёртки на входных изображениях. Его задачей является выделение различных признаков изображений, таких как грани и текстуры, с использованием 8 различных фильтров.

*Слой 2: Пулинг слой*:

• Тип: Пулинг слой (nn.MaxPool2d)

• Тип пулинга: MaxPooling

• Размер пулинга: 2x2

Слой 2 выполняет операцию пулинга (субдискретизации), уменьшая размер изображения в 2 раза. MaxPooling выбирает максимальное значение из каждой 2x2 области изображения.

*Слой 3: Свёрточный слой*:

• Тип: Свёрточный слой (nn.Conv2d)

• Количество фильтров: 16

• Размер фильтров: 3x3

• Заполнение (padding): 1 пиксель.

• Функция активации: ReLU

Слой 3 аналогичен Слой 1, но уже на входе получает данные, уменьшенные после применения Слоя 2. Он выделяет более абстрактные признаки изображений.

*Слой 4: Пулинг слой*:

• Тип: Пулинг слой (nn.MaxPool2d)

• Тип пулинга: MaxPooling

• Размер пулинга: 2x2

Слой 4 аналогичен Слой 2 и уменьшает размерность данных.

*Слой 5: Свёрточный слой*:

• Тип: Свёрточный слой (nn.Conv2d)

• Количество фильтров: 32

• Размер фильтров: 3x3

• Заполнение (padding): 1 пиксель.

• Функция активации: ReLU

Слой 5 продолжает выделять более сложные признаки изображений после применения двух предыдущих свёрточных слоев.

*Слой 6: Пулинг слой*:

• Тип: Пулинг слой (nn.MaxPool2d)

• Тип пулинга: MaxPooling

• Размер пулинга: 2x2

Слой 6 аналогичен Слой 2 и Слой 4, уменьшая размерность данных.

*Слой 7: Полносвязанный слой (Fully Connected Layer)*

• Тип: Полносвязанный слой (nn.Linear)

• Количество выходных нейронов: 10 (это число классов в задаче MNIST).

• Функция активации: Отсутствует.

Слой 7 является полносвязанным слоем, который преобразует выходы предыдущих слоев в оценки для каждого из 10 классов в задаче MNIST.

Также в модели использовались слои BatchNorm2d, который играет важную роль в обучении нейронных сетей, ускоряя сходимость обучения и улучшая стабильность градиентов. Также он вносит регуляризацию, уменьшая вероятность переобучения.

Структура модели с параметрами:

Number of parameters: 8890

conv1.weight 72

conv1.bias 8

bn1.weight 8

bn1.bias 8

conv2.weight 1152

conv2.bias 16

bn2.weight 16

bn2.bias 16

conv3.weight 4608

conv3.bias 32

bn3.weight 32

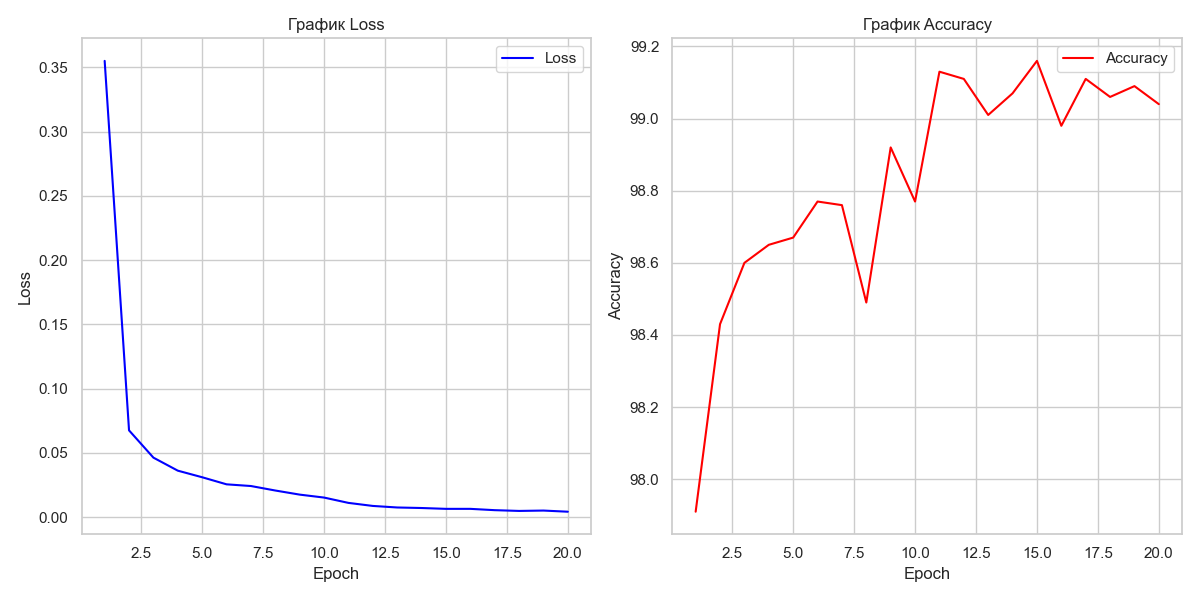
bn3.bias 32

fc.weight 2880

fc.bias 10

# Эксперименты

Были обучены 14 моделей, с разными параметрами и архитектурами. Наилучшим оказался 13 эксперимент:



Лучший результат для тестовой выборки показал значение: 99.13%.

Также модель была протестирована на вручную сделанных данных при помощи Paint. Было создано две выборки – простая и сложная.

Простая:

Изображение выглядит как текст, Шрифт, снимок экрана, типография

Автоматически созданное описание

Сложная:

Изображение выглядит как текст, Шрифт, снимок экрана, типография

Автоматически созданное описание

В первом случае модель не смогла распознать 6, а в остальных всё выполнено верно. Это говорит о правильном составлении модели.

# ПРИЛОЖЕНИЕ

Код модели:

class CNN(nn.Module):  
 def \_\_init\_\_(self):  
 super(CNN, self).\_\_init\_\_()  
 self.relu = nn.ReLU(inplace=True)  
 self.maxpool = nn.MaxPool2d(2, 2)  
  
 self.conv1 = nn.Conv2d(in\_channels=1,  
 out\_channels=8,  
 kernel\_size=3,  
 padding=1)  
 self.bn1 = nn.BatchNorm2d(8)  
  
 self.conv2 = nn.Conv2d(8, 16, 3, padding=1)  
 self.bn2 = nn.BatchNorm2d(16)  
  
 self.conv3 = nn.Conv2d(16, 32, 3, padding=1)  
 self.bn3 = nn.BatchNorm2d(32)  
  
 self.fc = nn.Linear(32 \* 3 \* 3, cfg.DATA.NUM\_CLASSES)  
  
 def forward(self, x):  
 x = self.conv1(x)  
 x = self.bn1(x)  
 x = self.relu(x)  
 x = self.maxpool(x)  
  
 x = self.conv2(x)  
 x = self.bn2(x)  
 x = self.relu(x)  
 x = self.maxpool(x)  
  
 x = self.conv3(x)  
 x = self.bn3(x)  
 x = self.relu(x)  
 x = self.maxpool(x)  
  
 x = x.view(x.size(0), -1)  
 x = self.fc(x)  
  
 return x