|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное автономное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ Информатика, искусственный интеллект и системы управления

КАФЕДРА Системы обработки информации и управления

**ОТЧЕТ**

***ПО ЛАБОРАТОРНОЙ РАБОТЕ №2***

***НА ТЕМУ:***

**Свёрточные нейронные сети**

Студент \_\_ИУ5-22М\_\_ **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_П.А. Бибиков \_\_**

(Группа) (Подпись, дата) (И.О.Фамилия)

Преподаватель **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_А.И.Канев\_\_**

(Подпись, дата) (И.О.Фамилия)

2025 г.

СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 2](#_Toc169968534)

[ВВЕДЕНИЕ 3](#_Toc169968535)

[Задание 3](#_Toc169968536)

[Часть 1. Обучение сверточной нейронной сети 5](#_Toc169968537)

[Таблица с результатами 10](#_Toc169968538)

[Часть 2. Сравнение свёрточной НС и полносвязной НС 13](#_Toc169968539)

[ЗАКЛЮЧЕНИЕ 15](#_Toc169968540)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 16](#_Toc169968541)

ВВЕДЕНИЕ

Сверточные нейронные сети[1][2] (Convolutional Neural Networks, CNN) – класс алгоритмов машинного обучения. С их помощью удается достичь впечатляющих результатов в области распознавания образов, классификации изображений, а также обработки и анализа видеоданных.

Сверточные нейронные сети разработаны специально для работы с данными, имеющими пространственную или растровую структуру, такими как изображения. Отдельно стоит отметить их способность к автоматическому изучению признаков. Вместо ручного выбора сверточные нейронные сети используют обратное распространение ошибки и градиентный спуск для настройки весовых коэффициентов.

# Задание

По заданию выбрать свои классы и обучить сверточную нейронную сеть из примера, используя GPU, а затем повысить точность модели. Провести три обучения для 3 разных тактик pooling: пуллинг с помощью шага свёртки stride, макс пуллинг, усредняющий пуллинг. Сравнить достигнутое качество, время обучения и степень переобучения. Выбрать лучшую конфигурацию. Сохранить модель. Перезапустить среду выполнения - теряются все текущие переменные.

Загрузить в colab готовую уже обученную на cifar100 модель. Преобразовать в onnx и сохранить локально.

Скачать каталог с html-файлом и встроить в него два файла моделей - обученную на ЛР1 и на ЛР2.

Скачать картинки из интернета согласно варианту и открыть их в html по кнопке. Автоматически в скрипте масштабируется изображение.

Выбрать в js нужные классы для готовой модели. Проверить на устойчивость обе модели, полносвязную и свёрточную, двигая картинку, убедиться в наличии свойства инвариантности сверточного слоя.

Для задания нужно сформировать свою подвыборку CIFAR100 по варианту. Вы должны использовать следующие классы из CIFAR100:

1. Номер группы + 15
2. Номер варианта + 56
3. ИУ5 (Номер варианта + 21); ГУИМЦ (80); Иностранцы (90)

Полученные варианты:

1. 22 + 15 = 37
2. 3 + 56 = 59
3. 3 + 21 = 24

# Часть 1. Обучение сверточной нейронной сети

Начальная конфигурация нейронной сети указана на рис.1

Изображение выглядит как текст, снимок экрана, меню, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 1 – базовая конфигурация СНС.

Проводим для неё обучение, обучаем 500 эпох, батч 128, метод – обучение с импульсом.

Изображение выглядит как График, линия, снимок экрана, диаграмма

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 2 – графики функции ошибки и точности для начальной конфигурации.

Полученные графики свидетельствуют о том, что получили переобучение – при accuracy в 87%, обучающая выборка имеет точность 100%. Метрики базовой нейронной сети приведены в таблице 1.

Таблица 1 – базовые метрики модели на тестовой выборке.

|  |  |  |  |
| --- | --- | --- | --- |
| класс | precision | recall | f1-score |
| 37 | 0.9388 | 0.9200 | 0.9293 |
| 59 | 0.8602 | 0.8000 | 0.8290 |
| 24 | 0.8165 | 0.8900 | 0.8517 |

По графикам можно судить о начале переобучения около 200-й эпохи – ограничим модель на ней. Результаты для 200-й эпохи – рисунок 2 и точность 89%.

Изображение выглядит как линия, График, снимок экрана, диаграмма

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 2 - графики функции ошибки и точности на 200-й эпохе.

Сейчас в модели используется ядро 5x5 с шагом 4. Пуллинг происходит только после 2 свёртки, а размер картинки уменьшается за счёт того, что шаг 4 берёт каждый 4й пиксель. Следовательно, сравнивать будем по accuracy в конфигурациях, описанных в таблице 2. Для шага 2, первый слой pooling делает пуллинг 2x2.

Таблица 2 – Конфигурации и сравнение результатов обучения

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Шаг | Слой pooling 1 | Слой pooling 2 | Количество эпох до переобучения | Точность, % |
| 4 | Нет | avg | 200 | 89 |
| 2 | avg | avg | 180 | 88 |
| 2 | avg | max | 40 | 83.7 |
| 2 | max | avg | 200 | 91.3 |
| 2 | max | max | 23 | 87.7 |
| 1 | avg | avg | 200 |  |
| 1 | avg | max | 80 |  |
| 1 | max | avg | 60 |  |
| 1 | max | max | 35 | 85 |

По таб. 2 видно, что наилучшие результаты дают модели max-avg и max-max для шага 2 и avg-avg, max-max для шага 1. Лучшая модель – шаг 1, оба pooling – усредняющие.

Итоговые графики точности и ФП для модели 1-max-max представлены на рис.3 и рис. 4.

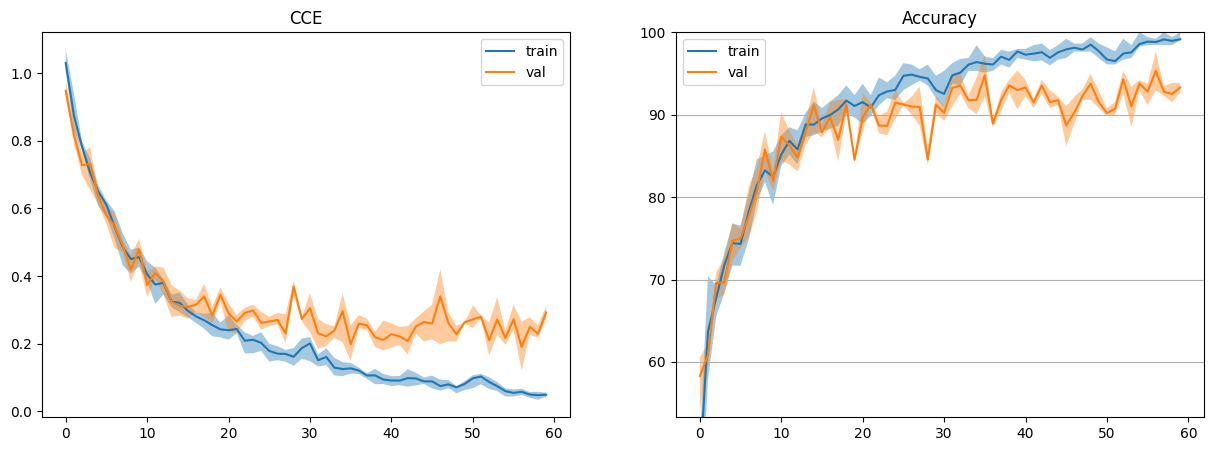


Рис. 3 – график ФП и точности для 60-й эпохи 1-max-max.

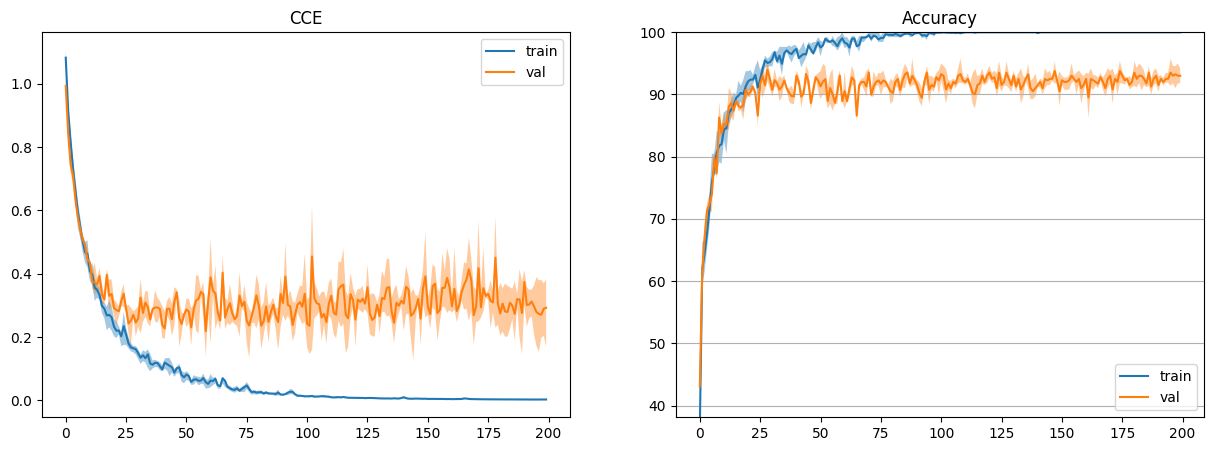


Рис. 4 – график ФП и точности для 100-й эпохи 1-max-max.

Сохраним модель, обученную на 60 эпохе. Попробуем увеличить точность за счёт уменьшения learning rate и увеличения числа эпох. Для lr = 1e-3, точность – 90.3. График ФП изображён на рис. 5.

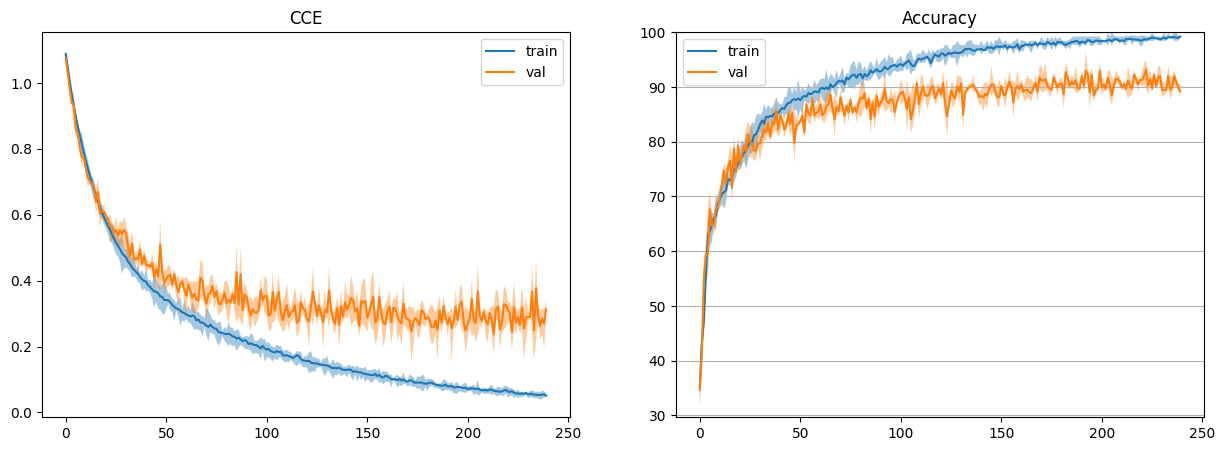


Рис. 5 - график ФП и точности для 50-й эпохи 1-max-max, при lr = 1e-3

Проведём аналогичное исследование для модели с ядром 3x3. Как и в прошлом случае, для шага 2 слой pooling 1 имеет размерность 2x2.

Таблица 3 – Конфигурации и сравнение результатов обучения для ядра 3x3.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Шаг | Слой pooling 1 | Слой pooling 2 | Количество эпох до переобучения | Точность, % |
| 2 | avg | avg | 160 | 87.4 |
| 2 | avg | max | 150 | 90.3 |
| 2 | max | avg | 50 | 88.2 |
| 2 | max | max | 90 | 91.9 |
| 1 | avg | avg | 50 | 91.3 |
| 1 | avg | max | 160 | 91.6 |
| 1 | max | avg | 80 | 91,7 |
| 1 | max | max | 60 | 92 |

Лучшей моделью вновь оказалась 1-max-max, график ФП и точность на рис. 6.

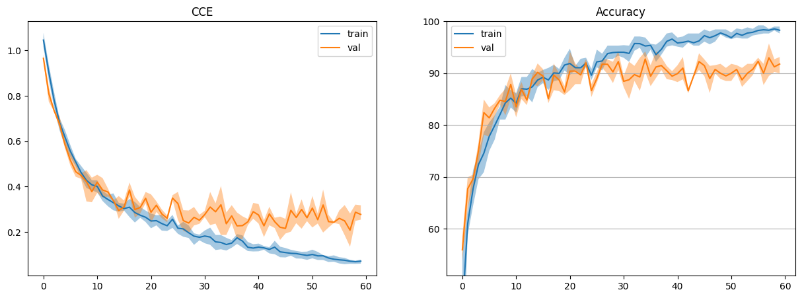


Рис. 6 – график ФП и точности для 40-й эпохи 1-max-max с ядром 3x3

Итого самой точной оказалась модель 5x5-1-max-max, с точностью 92.3 %. Визуализация её функции потерь представлена на рис. 7 и рис. 8.

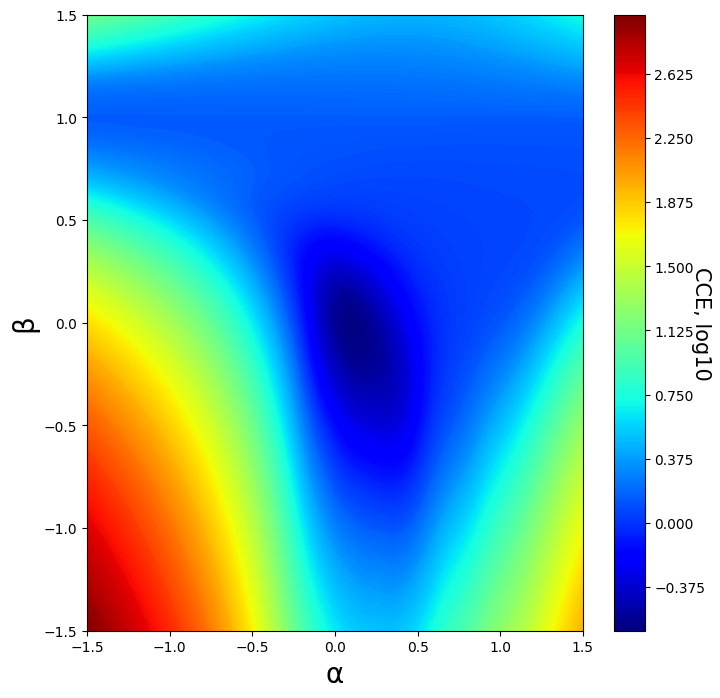


Рис. 8 – 2d визуализация ФП.

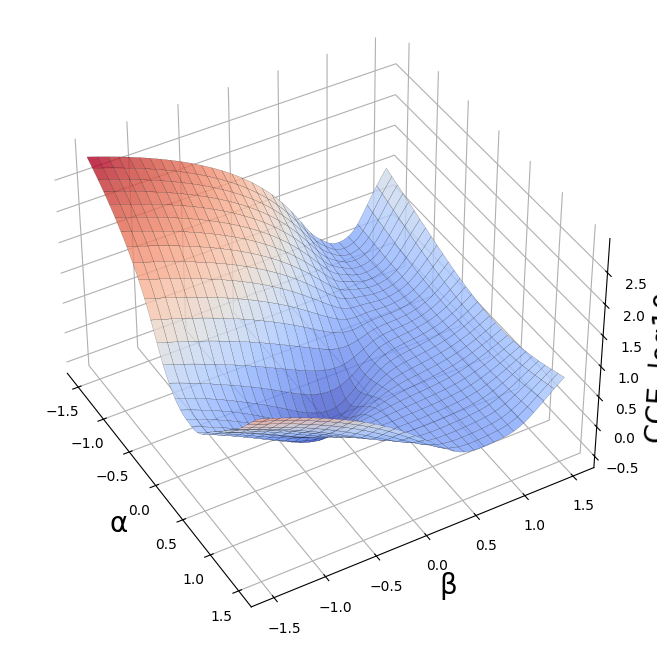


Рис. 9 – 3d визуализация ФП

# Таблица с результатами

| **Конфигурация сети** | **Гиперпараметры** | **Точность** | **Комментарии** |
| --- | --- | --- | --- |
| Conv2d(3, 32, kernel\_size=(5, 5), stride=(4, 4), padding=(2, 2))  Conv2d(32, 64, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1))  AvgPool2d(kernel\_size=4, stride=4, padding=0) | lr = 0.005, batch\_size = 128, epochs = 500 | test = 90 %, train = 100% | Базовый вариант |
| Conv2d(3, 32, kernel\_size=(5, 5), stride=(4, 4), padding=(2, 2))  Conv2d(32, 64, (3, 3), (1, 1), (1, 1))  AvgPool2d(kernel\_size=4, stride=4, padding=0) | lr = 0.005, batch\_size = 128, epochs = 200 | test = 89,1%, train = 97% | В базовой модели происходит переобучение, поэтому уменьшаем количество эпох до 200. |
| Conv2d(3, 32, kernel\_size=(5, 5), stride=(1, 1), padding=(2, 2))  AvgPool2d(kernel\_size=4, stride=4, padding=0)  Conv2d(32, 64, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1))  AvgPool2d(kernel\_size=4, stride=4, padding=0) | lr = 0.005, batch\_size = 128, epochs = 200 | test = 91%, train = 92% | Добавили слой усредняющего pooling после первого сверточного слоя. |
| Conv2d(3, 32, kernel\_size=(5, 5), stride=(1, 1), padding=(2, 2))  MaxPool2d(kernel\_size=4, stride=4, padding=0, dilation=1, ceil\_mode=False)  Conv2d(32, 64, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1))  AvgPool2d(kernel\_size=4, stride=4, padding=0) | lr = 0.005, batch\_size = 128, epochs = 200 | test = 92.3%, train = 91% | Заменяем усредняющий на макс пуллинг после первого сверточного слоя. |
| Conv2d(3, 32, kernel\_size=(5, 5), stride=(1, 1), padding=(2, 2))  MaxPool2d(kernel\_size=4, stride=4, padding=0, dilation=1, ceil\_mode=False)  Conv2d(32, 64, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1))  AvgPool2d(kernel\_size=4, stride=4, padding=0) | lr = 0.001, batch\_size = 128, epochs = 1000 | test = 92,4%, train = 94% | Уменьшаем шаг обучения в 5 раз, увеличиваем количество эпох в столько же. |
| Conv2d(3, 32, kernel\_size=(5, 5), stride=(1, 1), padding=(2, 2))  MaxPool2d(kernel\_size=4, stride=4, padding=0, dilation=1, ceil\_mode=False)  Conv2d(32, 64, kernel\_size=(3, 3), stride=(1, 1), padding=(1, 1))  AvgPool2d(kernel\_size=4, stride=4, padding=0) | lr = 0.005, batch\_size = 256, epochs = 400 | test = 91,2%, train = 94% | Увеличим размер батча в 2 раза и во столько же количество эпох. |
| FC(8), FC(8), FC(3) | lr = 0.001, batch\_size = 128, epochs =450 | test = 77%, train = 91% | Наилучший вариант полносвязной сети из ЛР-1 |

# Часть 2. Сравнение свёрточной НС и полносвязной НС

Для проведения сравнения, используем веб-пакет. У нас имеется 3 класса: сосны, тараканы и дома. Будем сравнивать тараканов.

Исходное изображение представлено на рис. 10. Обе НС уверены, что это – пикап. Используем сдвиг вниз, отклонения начнутся на 10 пикселей вниз. Для сверточной НС сдвиг вниз может быть до 16 - изображено на рис. 12.



Рис. 10 – исходное изображение

Изображение выглядит как снимок экрана, зеленый, диаграмма, линия

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 11 – предел сдвига вправо для полносвязной НС

**Изображение выглядит как снимок экрана, Прямоугольник, диаграмма, линия

Контент, сгенерированный ИИ, может содержать ошибки.**

Рис. 12 – предел сдвига вправо для свёрточной НС.

ЗАКЛЮЧЕНИЕ

Свёртка в нейронных сетях позволяет перейти от прямого изучения входящих данных к поиску отдельных признаков и закономерность в этих данных с помощью нейронов свёрточных слоёв, составляющих карты признаков. Пуллинг предназначен для уменьшения размера входных данных, при этом, не теряя информацию о признаках. Полносвязная нейронная сеть на конце сверточной позволяет проводить классификацию входных данных.

Для свёрточных НС всё так же эффективны обычные способы улучшения обучения – уменьшение learning rate и увеличение batch size. Обучение свёрточных НС обучается быстрее полносвязной, однако для того, чтобы такая сеть была качественной, её размер, и, соответственно, число признаков, должно быть большим. Сверточная нейронная сеть более устойчива к сдвигам, в отличие от полносвязной – это продемонстрировано экспериментально.

Наилучший вариант свёрточной нейронной сети получился при шаге обучения, равным 0,001, размере батча 128, количестве эпох – 1000, первом – максимизирующем pooling, а втором – усредняющем.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Методические указания к лабораторным работам. Источник: <https://github.com/iu5git/Deep-learning>
2. Материал видео-лекций по предмету “Разработка нейросетевых систем”. Источник: [Deep learning - YouTube](https://www.youtube.com/playlist?list=PLLELLTvDgUQ_d9eUj_3XVpAdGByuU37kT)