МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра «Систем обработки информации и управления»

ОТЧЕТ

**Домашняя работа №\_\_1\_\_**

по дисциплине«Разработка нейросетевых систем»

Тема: «Создание web-приложения для классификации изображений»

ИСПОЛНИТЕЛЬ: \_\_Журавлев Н.В.\_\_\_

ФИО

группа ИУ5-24М \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись

"23"\_\_03\_\_\_\_\_2024 г.

ПРЕПОДАВАТЕЛЬ: \_\_\_Канев А.И.\_\_\_\_\_\_

ФИО

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись

"\_\_"\_\_\_\_\_\_\_\_\_202\_ г.

Москва - 2024

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Здание

Требуется разработать экспертную систему на основе SSR веб-приложения для классификации изображений 3 близких классов. Для этого нужно собрать собственный набор данных, минимум по 100 изображений на класс. По каждому классу приложение должно выводить описание и 2-3 ключевых параметра.

# 1. Сохранение модели, обученной в Google Colab

## 1.1 Загрузка изображений

В поисковой системе выполнен поиск картинок для классов ”Хрущёвка”, ”Таунхаус”, ”Коттедж”.

Затем все картинки были выбраны и скачены через разширение Image downloader – Imageye в формате jpg.

Среди скаченных фото были выбранны фото, которые подходят для обучения и являются представителям класса.

После успешной загрузки изображений требуется очистить лишние объекты в ручном режиме на предмет дубликааатов. Это производится вручную.

## 1.1 Импортирование изображений в Google Colab

Выбранные изображения были загружены на google диск, после с помощью команды были загружены google colab:

drive.mount('/content/drive')

# 2. Модернизация исходной модели в Colab

В изначальном варианте нейросети были изменена структура нейросети, а именно используется предобученная модель.

Затем была выполнена аугментация (с коэффициентом из лабораторной работы 4), регуляризация и перенос обучения.

После модель была сохранена в среде colab в формате ONNX, после чего была скачена на локальный компьютер, для дальнейшего использования.

Точность модели после добавления аугментации, регуляризации и переноса обучения составляет 0.8667, что является удовлетворительным результатом. График же, представлен на рис. 1.

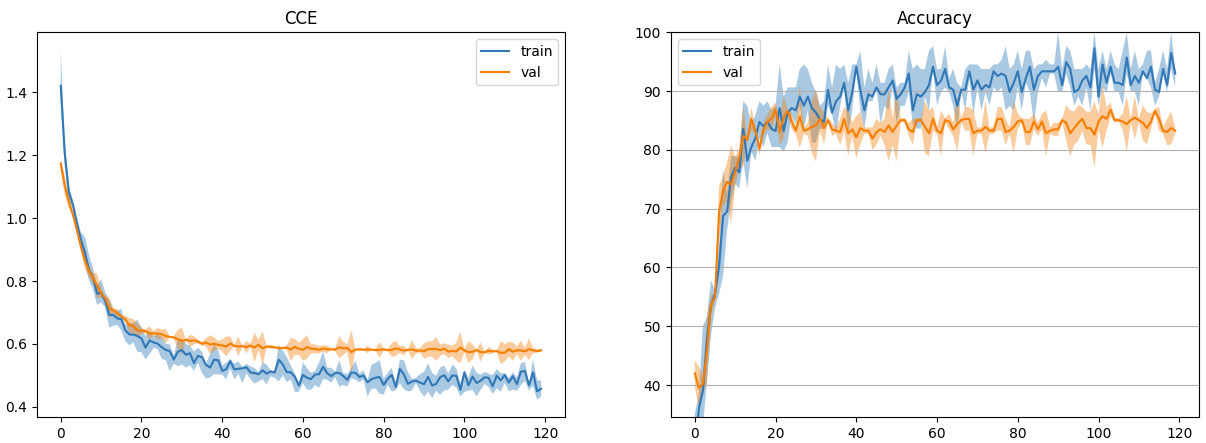


Рисунок 1 – Результат обучения модели базовой версии после добавления аугментации, регуляризации и переноса обучения

Распределение точности по классам, представлена в табл. 1.

Таблица 1 – Точность для каждого классы при обучении модели после добавления аугментации, регуляризации и переноса обучения

|  |  |  |  |
| --- | --- | --- | --- |
| **Train** |  | | |
| **Классы** | **Precision** | **Recall** | **F1-score** |
| Коттедж | 0.9756 | 1 | 0.9877 |
| Таунхаус | 1 | 0.9750 | 0.9873 |
| Хрущёвка | 1 | 1 | 1 |
| **Accuracy** |  | | 0.9917 |
| **Train** |  | | |
| **Классы** | **Precision** | **Recall** | **F1-score** |
| Коттедж | 0.8182 | 0.9 | 0.8571 |
| Таунхаус | 0.8947 | 0.85 | 0.8718 |
| Хрущёвка | 1 | 0.95 | 0.9744 |
| **Accuracy** |  | | 0.9 |

После увеличения коэффициента аугментации точность составляет 0.90. График представлен на рис. 2.

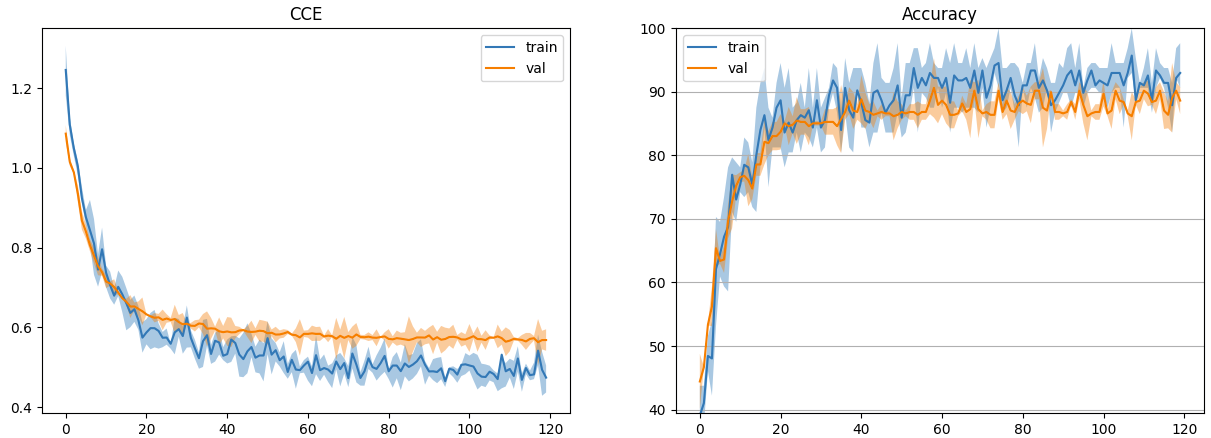
**

Рисунок 2 - Результат обучения модели после увеличения коэффициента аугментации

Распределение точности по классам, представлена в табл. 2.

Таблица 2 – Точность для каждого классы при обучении модели после увеличения коэффициента аугментации

|  |  |  |  |
| --- | --- | --- | --- |
| **Train** |  | | |
| **Классы** | **Precision** | **Recall** | **F1-score** |
| Коттедж | 0.9634 | 0.9875 | 0.9753 |
| Таунхаус | 0.9747 | 0.925 | 0.9686 |
| Хрущёвка | 1 | 0.9875 | 0.9937 |
| **Accuracy** |  | | 0.9792 |
| **Train** |  | | |
| **Классы** | **Precision** | **Recall** | **F1-score** |
| Коттедж | 0.8571 | 0.9 | 0.8780 |
| Таунхаус | 0.9 | 0.9 | 0.9 |
| Хрущёвка | 0.9474 | 0.9 | 0.9231 |
| **Accuracy** |  | | 0.9 |

Увеличение точности объясняется тем, что для обучения было подано больше данных.

Точность получившейся модели после изменения weight\_decay составляет 0.85. График представлен на рис. 3.

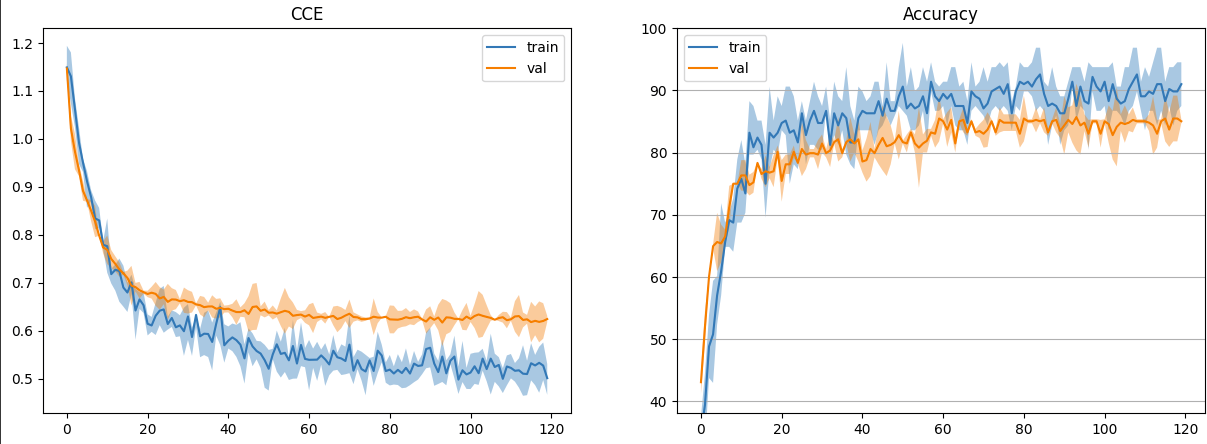


Рисунок 3 - Результат обучения модели после изменения weight\_decay

Распределение точности по классам, представлена в табл. 3.

Таблица 3 – Точность для каждого классы при обучении модели после изменения weight\_decay

|  |  |  |  |
| --- | --- | --- | --- |
| **Train** |  | | |
| **Классы** | **Precision** | **Recall** | **F1-score** |
| Коттедж | 0.9506 | 0.9625 | 0.9565 |
| Таунхаус | 0.9625 | 0.9625 | 0.9625 |
| Хрущёвка | 1 | 0.9875 | 0.9937 |
| **Accuracy** |  | | 0.9708 |
| **Train** |  | | |
| **Классы** | **Precision** | **Recall** | **F1-score** |
| Коттедж | 0.8235 | 0.7 | 0.7568 |
| Таунхаус | 0.7391 | 0.85 | 0.7907 |
| Хрущёвка | 1 | 1 | 1 |
| **Accuracy** |  | | 0.85 |

Судя по графику, точность улучшилась, что связано с тем, что умеьшино влияние на обучение.

Точность модели после применения заморозки составляет 0.7833. График представлен на рис. 4.

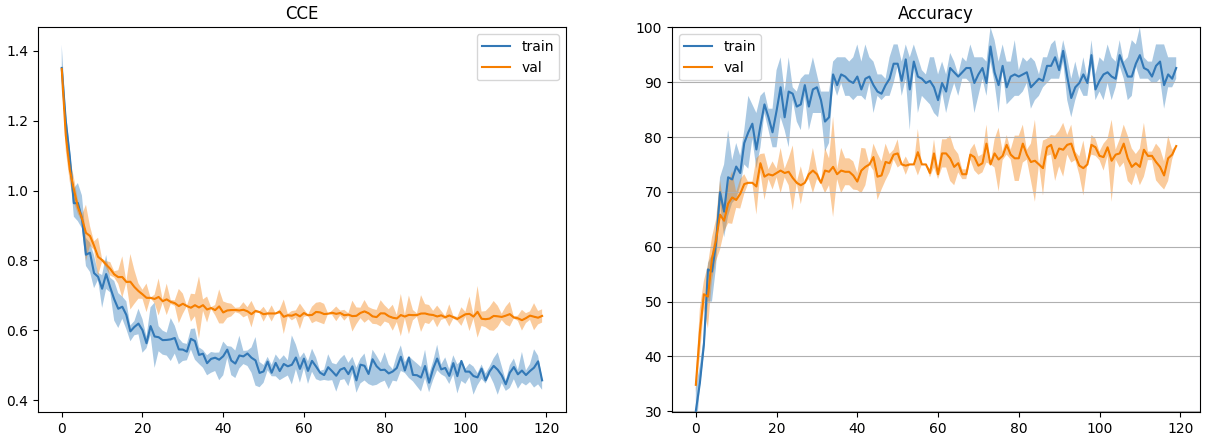


Рисунок 4 - Результат обучения модели после применение заморозки

Распределение точности по классам, представлена в табл. 4.

Таблица 4 – Точность для каждого классы при обучении модели после применение заморозки

|  |  |  |  |
| --- | --- | --- | --- |
| **Train** |  | | |
| **Классы** | **Precision** | **Recall** | **F1-score** |
| Коттедж | 0.9877 | 1 | 0.9938 |
| Таунхаус | 1 | 1 | 1 |
| Хрущёвка | 1 | 0.9875 | 0.9937 |
| **Accuracy** |  | | 0.9958 |
| **Train** |  | | |
| **Классы** | **Precision** | **Recall** | **F1-score** |
| Коттедж | 0.6538 | 0.85 | 0.7391 |
| Таунхаус | 0.7857 | 0.55 | 0.6471 |
| Хрущёвка | 0.95 | 0.95 | 0.95 |
| **Accuracy** |  | | 0.7833 |

Данный результат получился, т.к. предобученная модель изначально не предусматривалась для данной задачи

Точность модели после подбора гиперпараметров составляет 0.90, что является удовлетворительным результатом. График представлен на рис. 5.

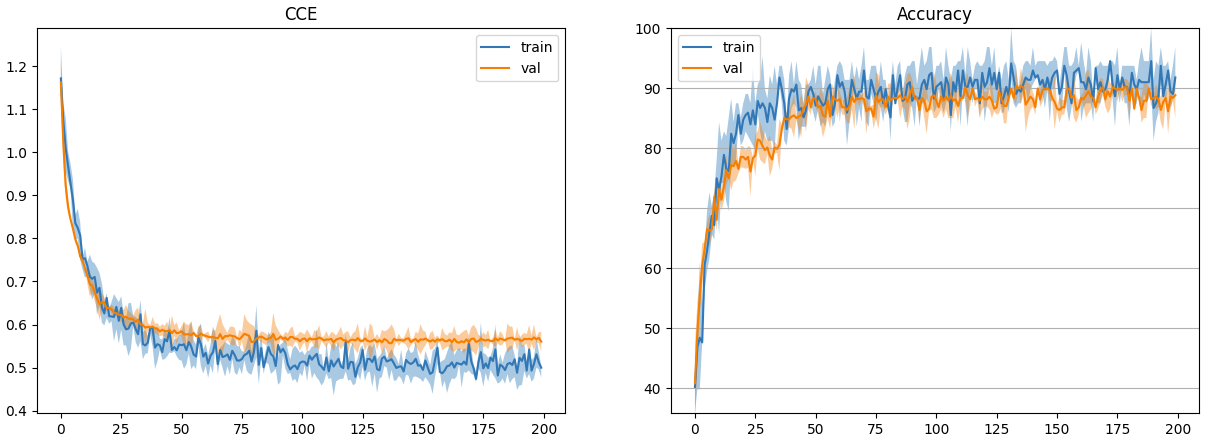


Рисунок 5 - Результат обучения модели после подбора гиперпараметров

Распределение точности по классам, представлена в табл. 5.

Таблица 5 – Точность для каждого классы при обучении модели после подбора гиперпараметров

|  |  |  |  |
| --- | --- | --- | --- |
| **Train** |  | | |
| **Классы** | **Precision** | **Recall** | **F1-score** |
| Коттедж | 0.9750 | 0.9750 | 0.9750 |
| Таунхаус | 0.9630 | 0.9750 | 0.9689 |
| Хрущёвка | 1 | 0.9875 | 0.9937 |
| **Accuracy** |  | | 0.9792 |
| **Train** |  | | |
| **Классы** | **Precision** | **Recall** | **F1-score** |
| Коттедж | 0.8947 | 0.85 | 0.8718 |
| Таунхаус | 0.8571 | 0.9 | 0.8780 |
| Хрущёвка | 0.95 | 0.95 | 0.95 |
| **Accuracy** |  | | 0.9 |

Лучше всего определяется хрущёвка, т.к. она значительно отличается от остальных зданий по строению.

Хуже определяется таунхаус, т.к. относительно коттеджа имеет повторяющиеся элементы и характерную форму.

Хуже всего определяется коттедж, т.к. имеет разнообразные формы.

# 3. Web-приложение классификации изображений

Был создан стандартный проект django. Его содержание представлено на рис. 2.

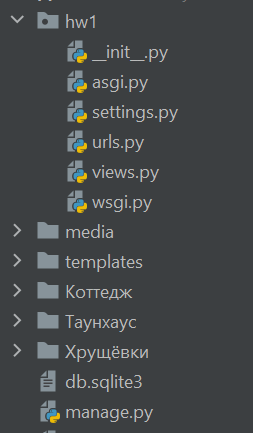


Рисунок 6 – Структура проекта Django

После создания проекта требуется создать в корне проекта папку media для последующего сохранения изображений и файлов формата ONNX. Внутри папки media необходимо создать папки "images" и "models".

В файл setting.py требуется добавить пути к ранее созданной папке media.

Затем были добавлены функции обработчики в файле views.py

файле urls.py были сделаны url приложения на которые возможно переходы и функции-обработчики к ним.

В папку templates добавить файл scorepage.html, который будет использоваться для отображения сайта.

После запуска было полученно следующее приложение, представленное на рис. 7.

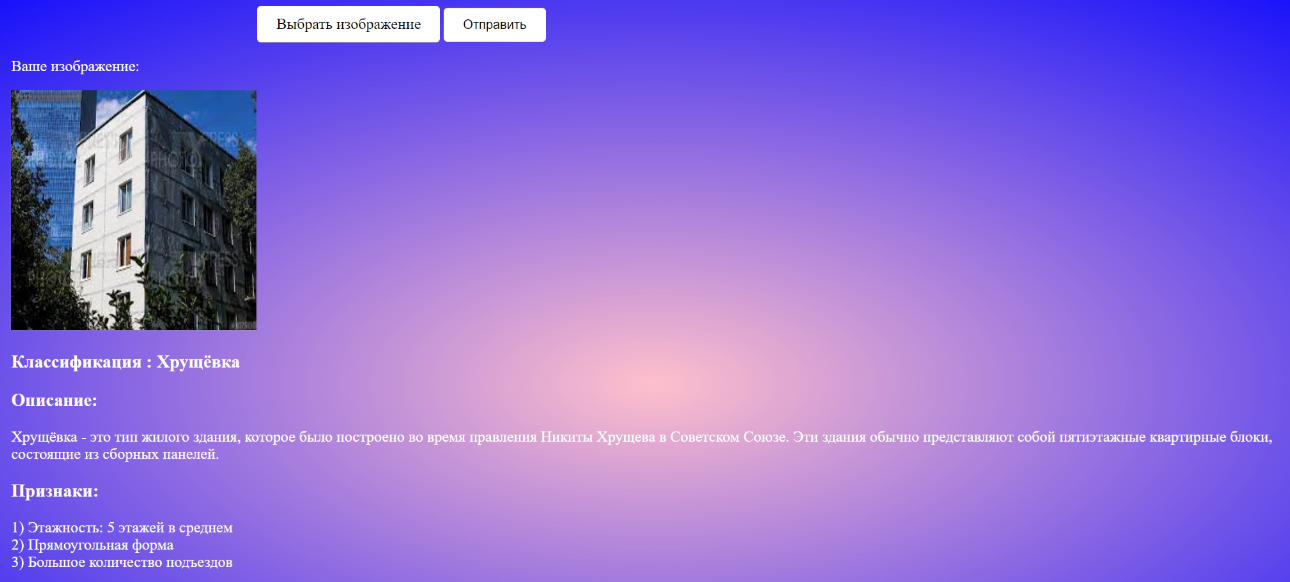


Рисунок 7 – Интерфейс приложения

# Итоговая таблица с результатами для всех вариантов обучения

На табл. 6 представлены результаты обучения.

Таблица 6 - Итоговая таблица с результатами для всех вариантов обучения

|  |  |  |  |
| --- | --- | --- | --- |
| **Конфигурация нейросети** | **Гиперпараметры** | **Точность** | **Комментарий** |
| Полносвязанная нейронная сеть | lr=0.003,  batch\_size = 32,  epoch = 120,  p=0.5,  weight\_decay=1e-5 | train = 100%  test = 48.33% | Базовый вариант |
| Resnet20 (обучаем 272,019) | lr=0.003,  batch\_size = 32,  epoch = 120,  p=0.5,  weight\_decay=1e-5 | train = 93.75%  test = 86.67% | Добавление аугментации, регуляризации и переноса обучения |
| Resnet20 (обучаем 272,019) | lr=0.003,  batch\_size = 32,  epoch = 120,  p=0.7,  weight\_decay=1e-5 | train = 97.92%  test = 90% | Увеличение коэффициента аугментации |
| Resnet20 (обучаем 272,019) | lr=0.003,  batch\_size = 32,  epoch = 120,  p=0.7,  weight\_decay=1e-4 | train = 97.08%  test = 85% | Изменение weight\_decay |
| Resnet20 (обучаем 262,211 из 272,019) | lr=0.003,  batch\_size = 32,  epoch = 120,  p=0.7,  weight\_decay=1e-4 | train = 99.58%  test = 78.33% | Применение заморозки для заранее обученной модели |
| Resnet20 (обучаем 272,019) | lr=0.003,  batch\_size = 32,  epoch = 120, p=0.7, weight\_decay=1e-4 | train = 97,92%  test = 90% | Оптимизация параметров |

# Вывод

В рамках выполнения домашнего задания, были применены навыки обучения нейронных сетей на основе ранее выполненых лабораторных работ.

В результате домашней работы было получено веб-приложение способное отличать три класса домов, а именно ”Хрущёвка”, ”Таунхаус”, ”Коттедж”. Точность результирующей нейронной сети составляет 90%, что является удовлетворительным результатом.