|  |  |
| --- | --- |
| Федеральное агентство железнодорожного транспорта  Омский государственный университет путей сообщения  Кафедра «Автоматика и системы управления»  ЗНАКОМСТВО С PYTHON  Лабораторная работа № 8  по теме «Информационные технологии» | |
|  | Студент гр. 23м  Н.С. Карпов, Л.М. Шамардина, А.К. Сагитова, А.О. Орозматов  « » 2024 г.  Руководитель –  старший преподаватель кафедры «АиСУ»  Т.В. Васеева  « » 2024 г. |
| Омск 2024 | |

# Цель работы

Целью данной лабораторной работы является знакомство языком программирования Python, а также с сортировкой фото с помощью нейронной сети.

# Выполнение работы

## Выбор среды

«Jupyter Notebook» – это веб-приложение с открытым исходным кодом, которое широко используется в области «Data Science» для визуализации данных. Оно представляет собой среду разработки, где можно сразу видеть результат выполнения кода и его отдельных фрагментов (рисунок 1).

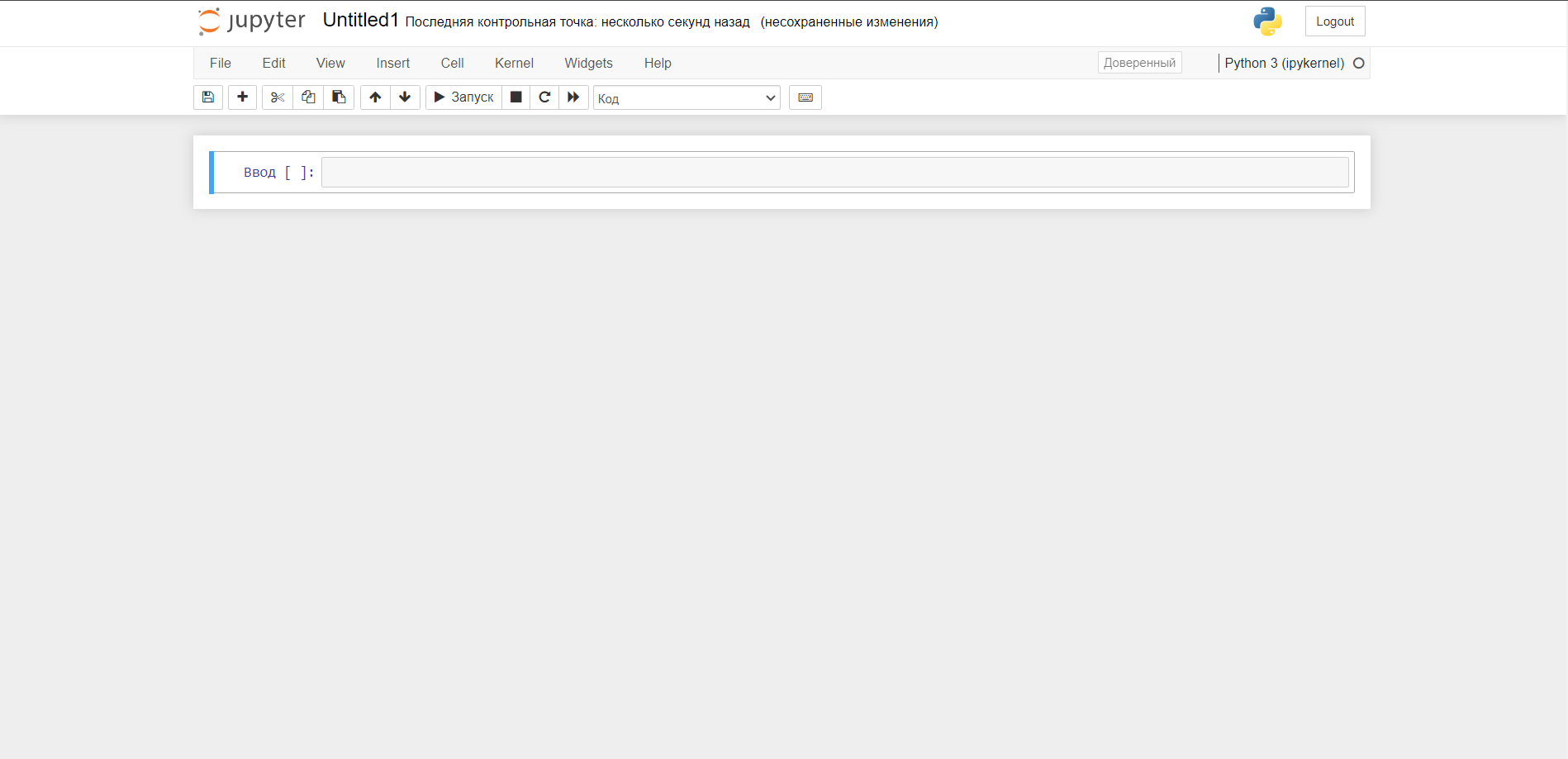


Рисунок 1 – Интерфейс «Jupyter Notebook»

## Установка среды «Anaconda»

Чтобы установить для Windows перейдем по ссылке: [https://www.anaconda.com/download#download-section](https://www.anaconda.com/download%23download-section) (рисунок 2 и 3).

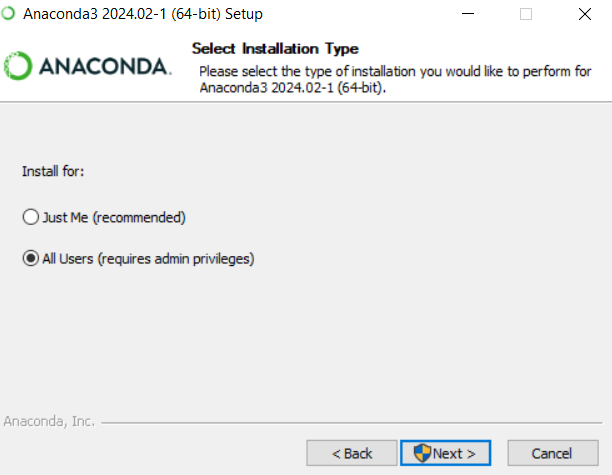


Рисунок 2 – Установка «Anaconda»

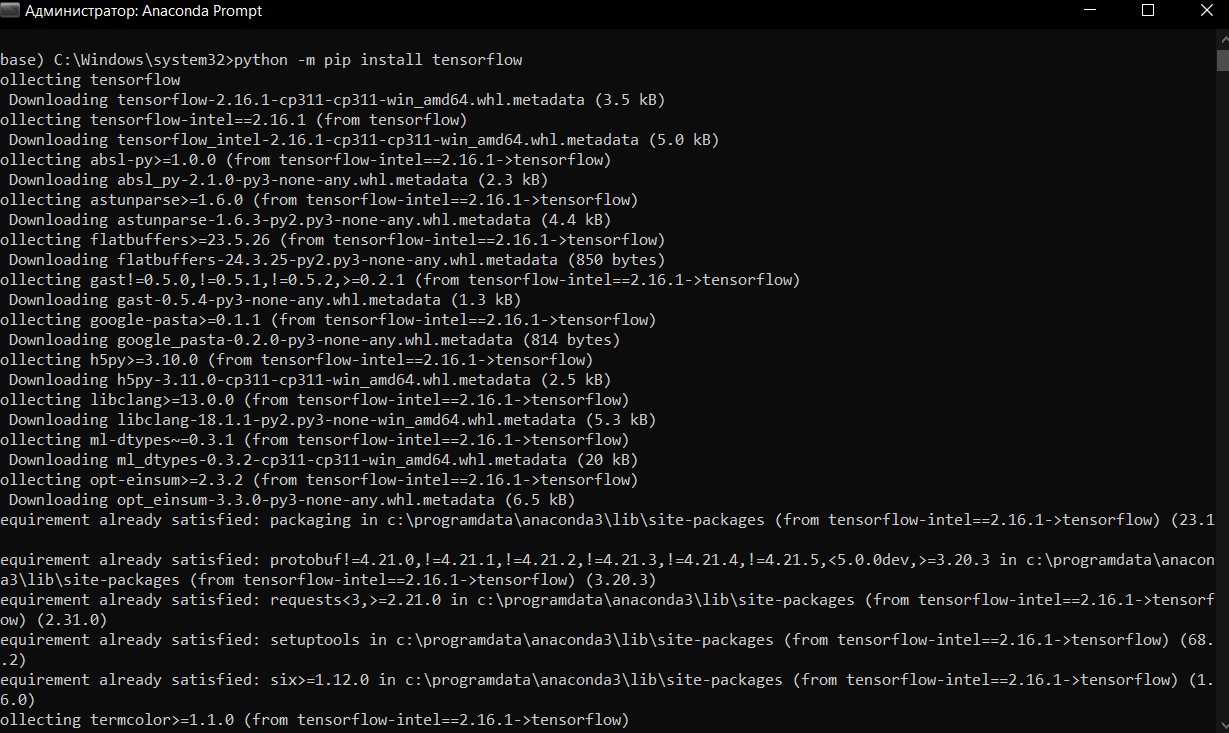


Рисунок 3 – Установка «tensorflow»

## Сортировка фото кошек и собак с помощью нейронной сети

Подгружаем нужные библиотеки, классы и модули (рисунок 4).

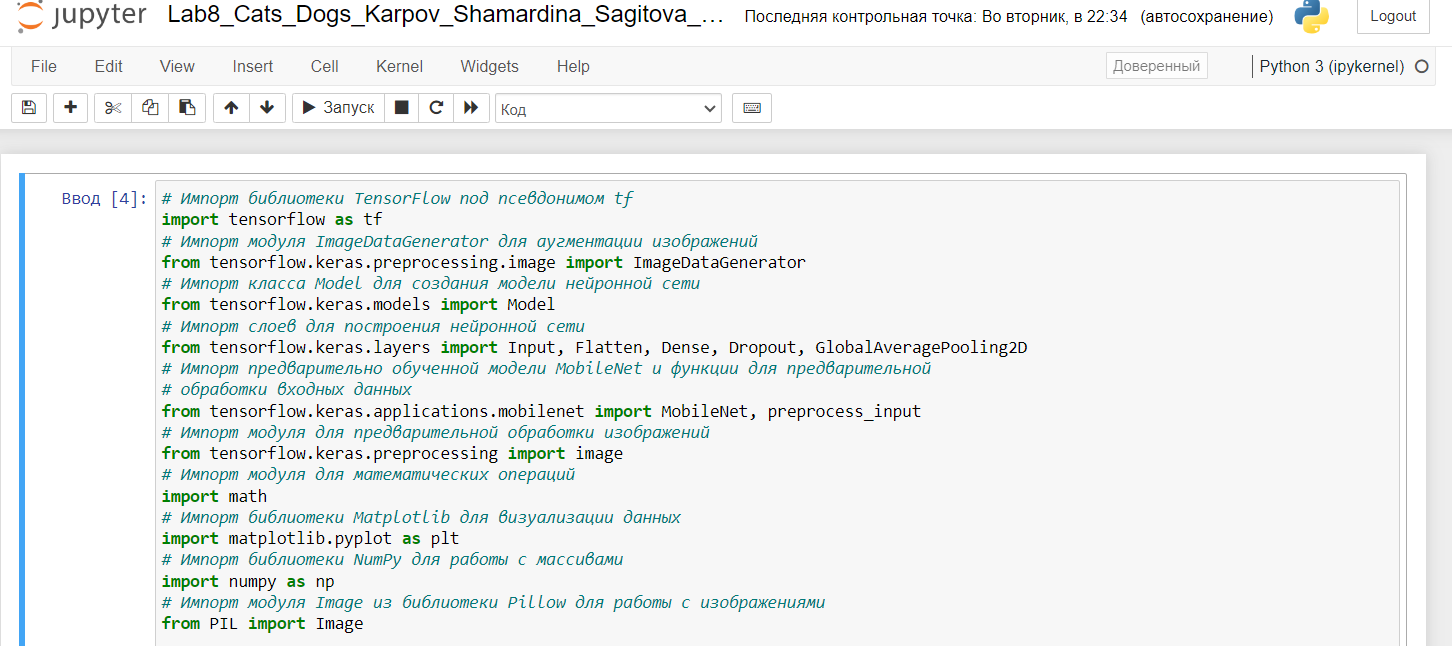


Рисунок 4 – Подгрузка нужных библиотек класс и модулей

Для чего нужны константы (рисунок 5)?

Эти константы используются при обучении модели для классификации изображений кошек и собак.

«TRAIN\_DATA\_DIR» и «VALIDATION\_DATA\_DIR» – указывают пути к каталогам с обучающими данными и данными для валидации соответственно. Наша модель будет обучаться на данных из каталога «CatsDogs/train» и валидироваться на данных из каталога «CatsDogs/test».

«TRAIN\_SAMPLES» и «VALIDATION\_SAMPLES» – эти константы определяют общее количество обучающих и валидационных примеров соответственно. Например, если «TRAIN\_SAMPLES» установлен в 1000, это означает, что у нас есть 1000 обучающих изображений автобусов и миниавтобусов в каталоге обучающих данных. Количество «TRAIN\_SAMPLES» может быть начальным выбором, который мы можем изменить в зависимости от результатов ваших экспериментов. Такое значения стоит брать, чтобы улучшить точность модели и её обобщающую способность.

«NUM\_CLASSES» – это количество классов, которые модель должна предсказывать. В нашем случае два класса: «кошки» и «собаки».

«IMG\_WIDTH» и «IMG\_HEIGHT» – эти константы определяют размеры изображений. Обычно перед обучением изображения изменяются до этих размеров. В нашем случае, изображения будут изменены до размера 224x224 пикселя.

«BATCH\_SIZE» – это количество образцов, которые обрабатываются моделью за один шаг обучения. Например, если «BATCH\_SIZE» установлен в 5, то модель будет обновлять свои веса после каждых 5 образцов.

Эти константы помогают управлять процессом обучения модели, устанавливая параметры, такие как размеры изображений, размер пакета, количество классов и т. д.

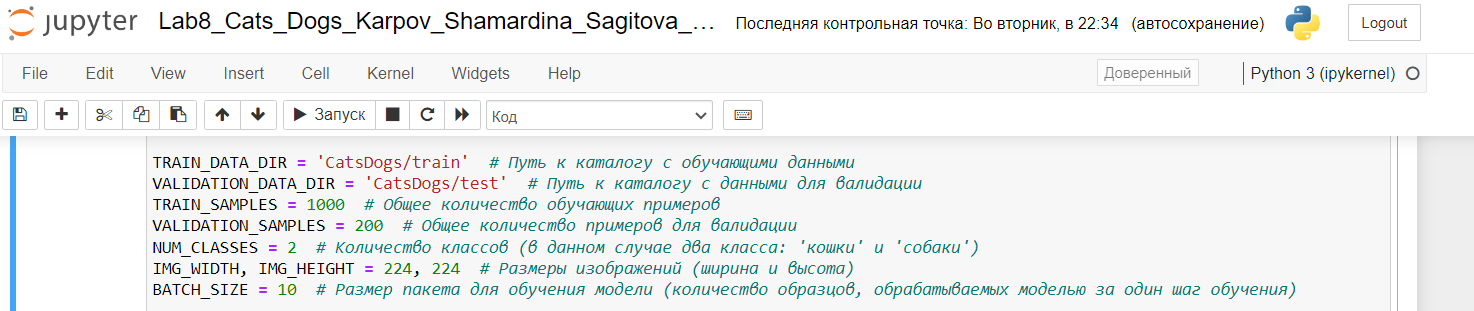


Рисунок 5 – Константы

Создаются два генератора данных (рисунок 6). Для обучения нейросети недостаточно пары сотен фотографий. Поэтому будет использоваться такой приём как аргументация.

Техника аргументации (или аугментации) данных — это процесс создания новых образцов данных путем применения различных случайных преобразований к существующим данным. В контексте обучения нейронной сети на изображениях это может включать в себя вращение, сдвиг, изменение масштаба и другие преобразования, которые могут помочь модели обучаться на более разнообразных и разносторонних данных.

В нашем коде используется «ImageDataGenerator» из библиотеки «Keras», который предоставляет мощные средства для аргументации изображений. Вот некоторые из параметров, которые мы установили:

«rotation\_range»: устанавливает диапазон углов для случайного вращения изображений.

«width\_shift\_range» и «height\_shift\_range»: определяют диапазон для случайного сдвига изображений по горизонтали и вертикали соответственно.

«zoom\_range»: устанавливает диапазон масштабирования изображений.

Эти преобразования помогают создать разнообразие изображений из ограниченного набора данных, что позволяет модели обучаться на более широком спектре условий и сценариев.

Генераторы данных («train\_generator» и «validation\_generator») используются для создания батчей данных из изображений в соответствующих каталогах, которые затем будут использоваться для обучения и валидации модели.

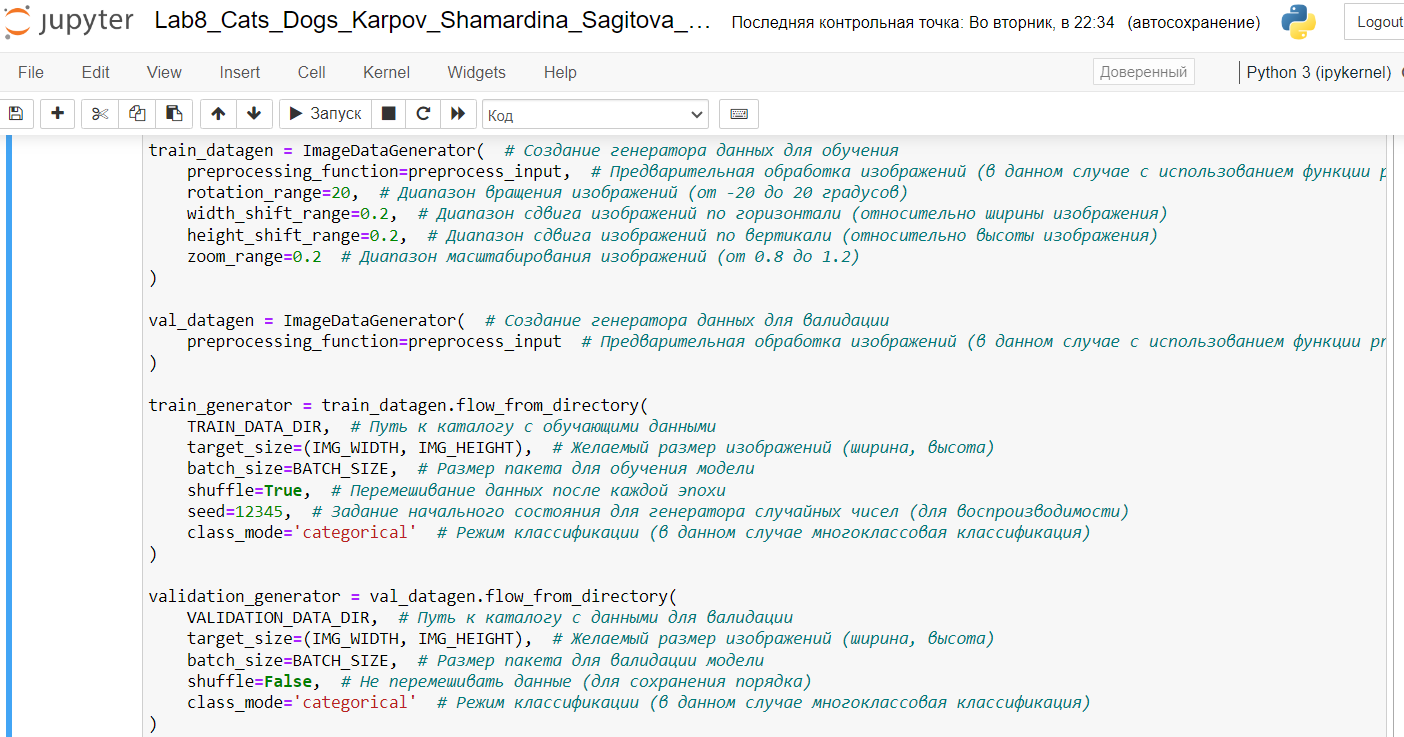


Рисунок 6 – Генераторы данных

## Создание модели на основе нейронной сети, которая основана на готовой нейронной сети «MobileNet» (она называется базовой моделью)

В базовой модели нужно отключить обучение (для всех слоев устанавливается trainable = False).

Входное изображение поступает на вход базовой модели.

После базовой модели добавляется несколько своих слоев: слой пулинга «GlobalAveragePooling2D»; полносвязный слой из 16 нейронов «Dense»; используется механизм «Дропаут» (случайное отключение нейронных связей во время обучения); последний слой должен выдавать два значения (вероятности того, что на фото кошка или собака – два класса).

Таким образом мы решаем задачу двухклассовой классификации (рисунок 7).

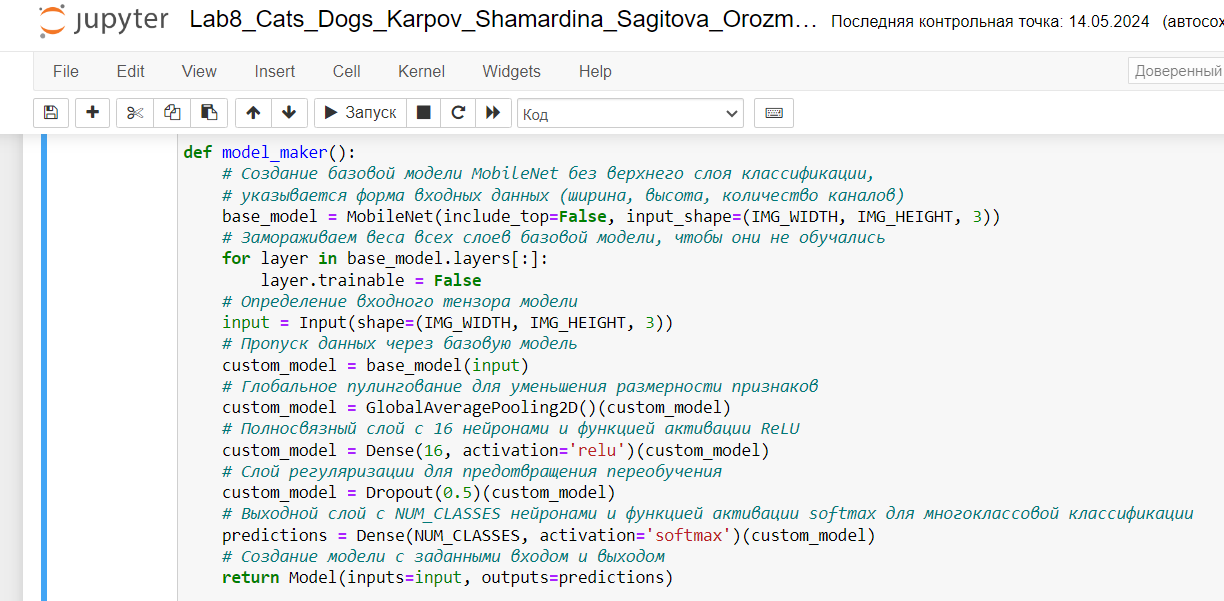


Рисунок 7 – Создание модели

Компилируем и обучаем модель (рисунки 8-9).

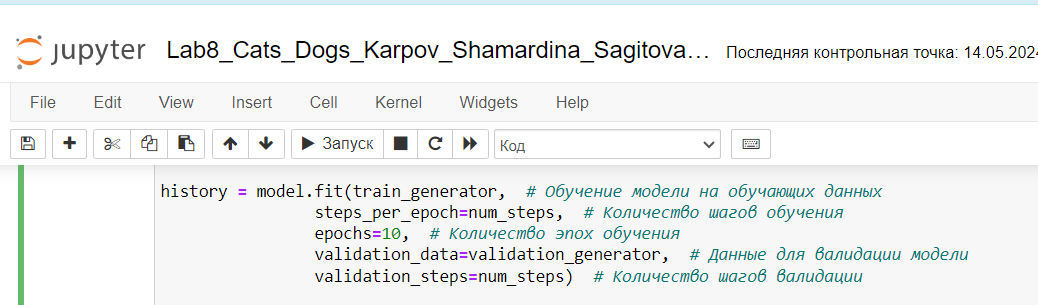


Рисунок 8 – Обучения модели (1)

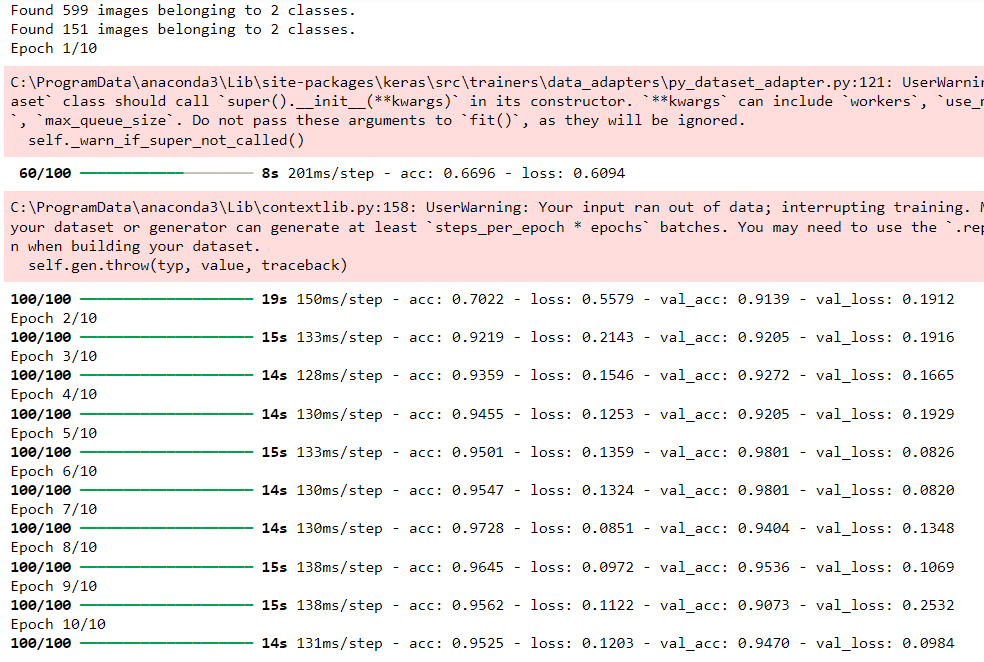


Рисунок 9 – Обучения модели (2)

Построения график (рисунки 12-14).

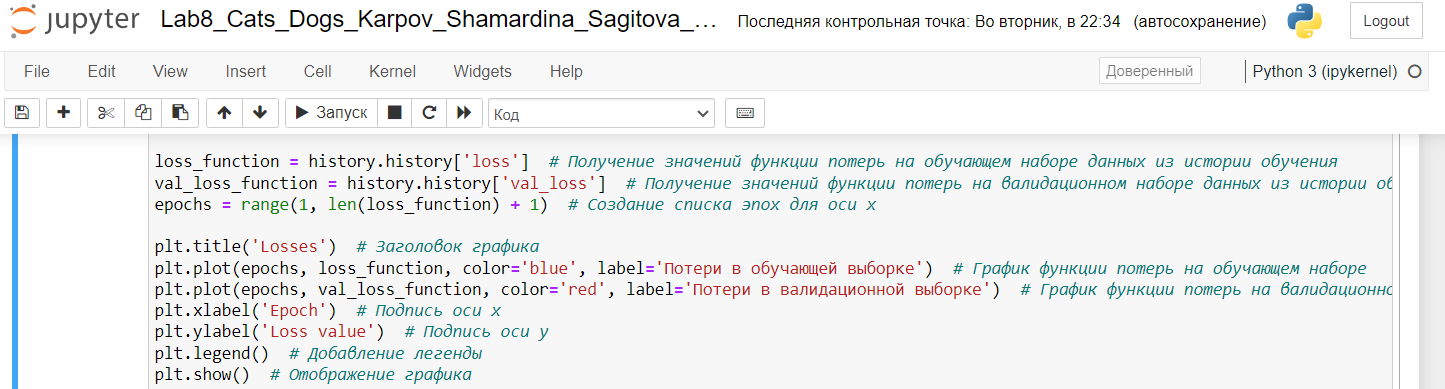


Рисунок 12 – Построения графика

Визуализация функции потерь позволяет оценить эффективность обучения модели: мы стремимся к тому, чтобы функция потерь на обучающими валидационном наборах данных уменьшалась с каждой эпохой, что свидетельствует о том, что модель обучается эффективно и не переобучается. Теперь модель можно использовать и проверить, как она работает.

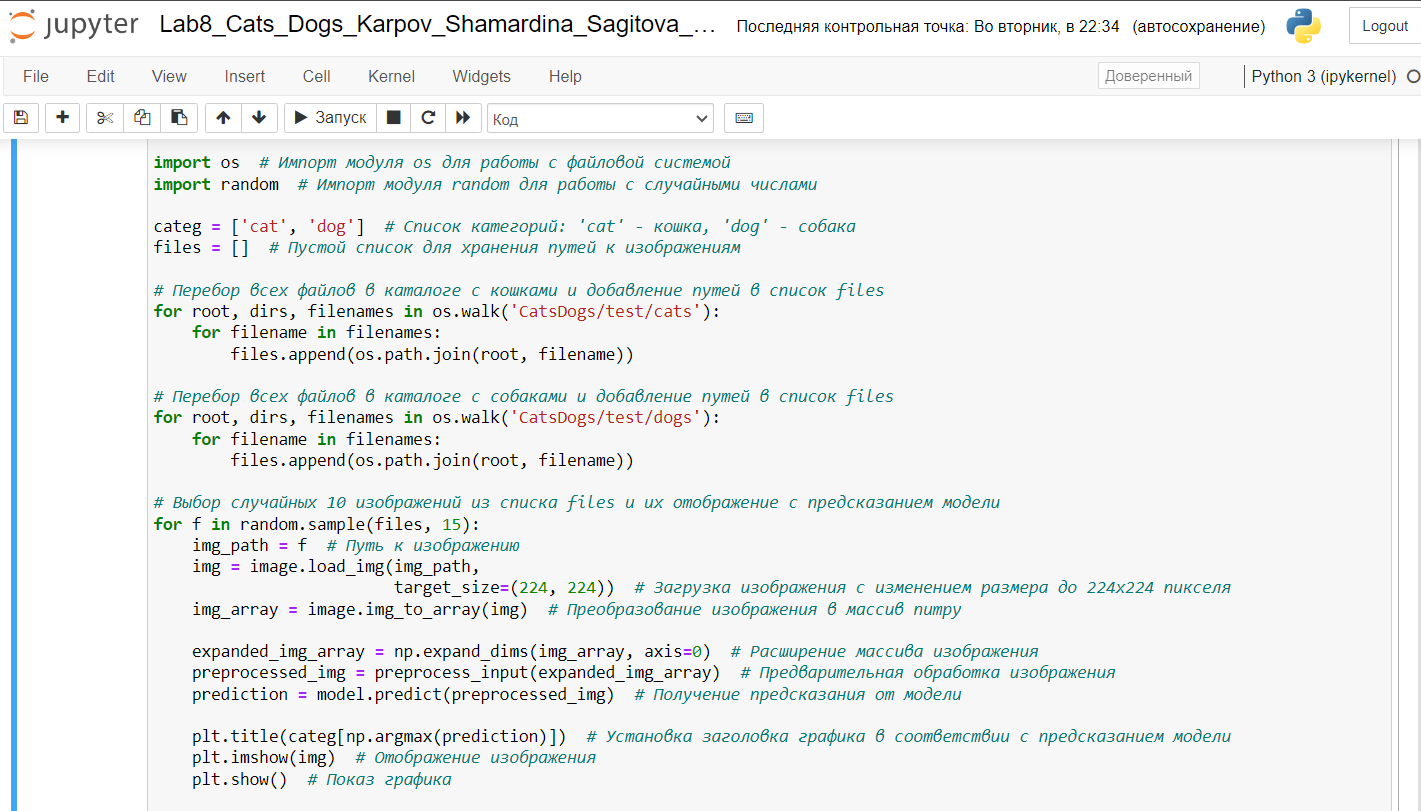


Рисунок 13 – Работа с изображениями

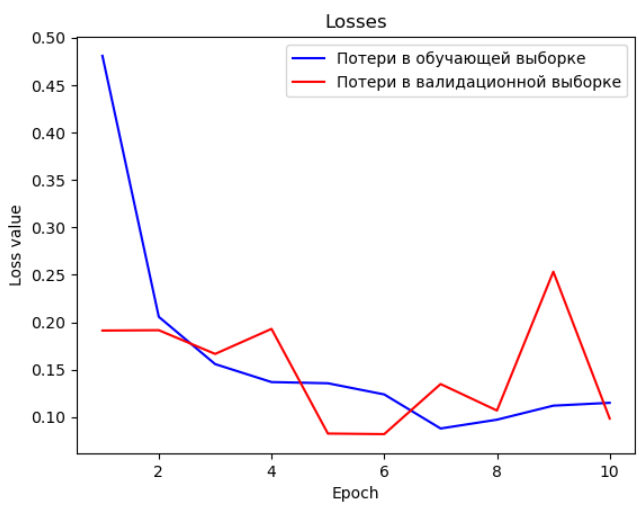


Рисунок 14 – График потерь

Как можно увидеть график потерь нестабилен, и некоторые фотографии сортируются неверно (рисунок 15).

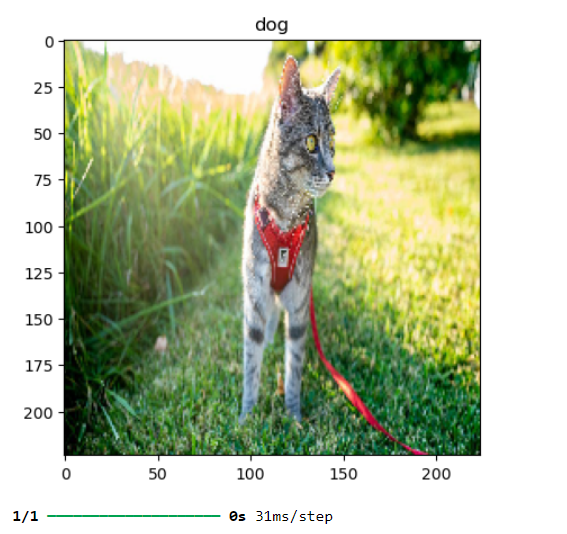


Рисунок 15 – Неверно отсортированное фото

Чтобы это исправить изменим полносвязный слой «Dance» с 16 нейронов на 60 нейронов, слой регуляризации с 0,5 до 0,65 и количество эпох 7 (рисунок 16).

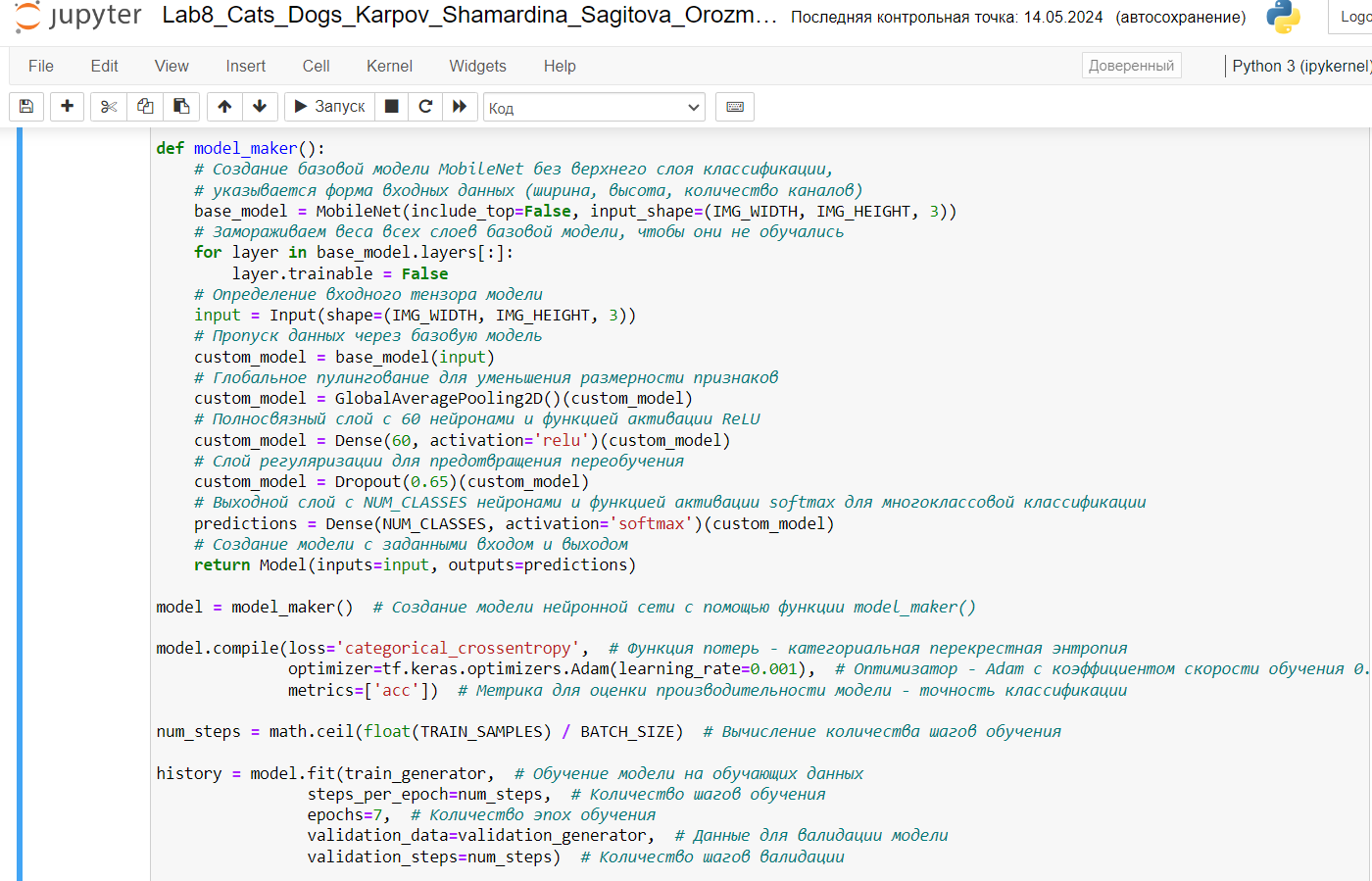


Рисунок 16 – Изменения параметров обработки

После изменений график стал более стабильным, а значит ошибок в обучении быть не должно (рисунки 17-18).

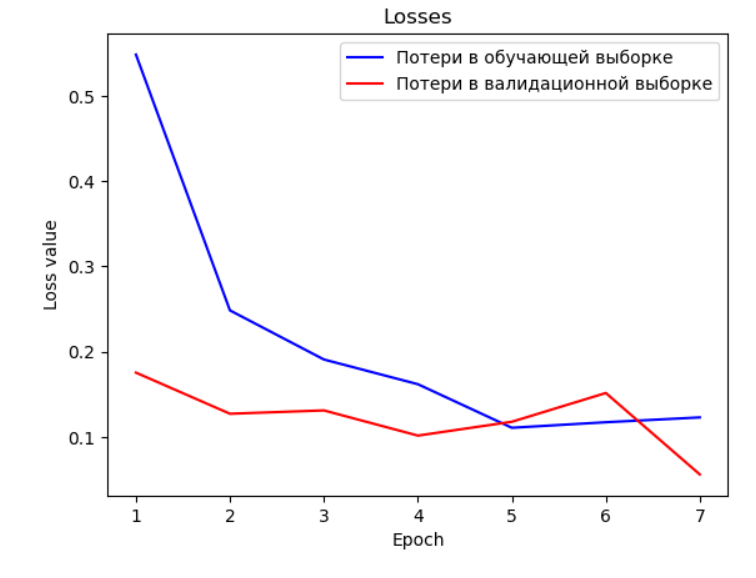


Рисунок 17 – Стабильный график потерь

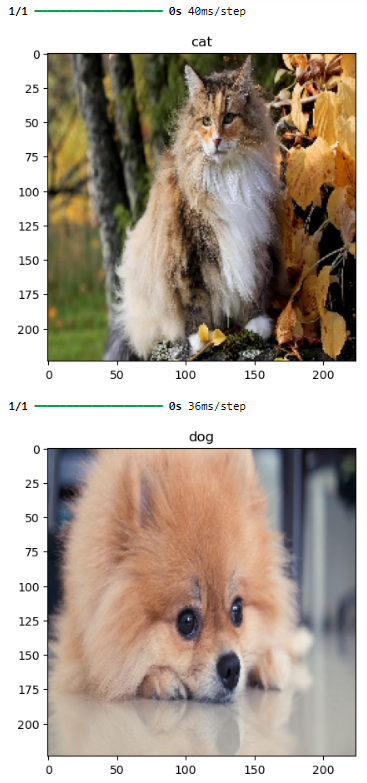


Рисунок 18 – Правильно обработанные фотографии

Для эффективного обучения нейронной сети можно изменять несколько ключевых параметров и методов. Рассмотрим, какие параметры можно регулировать и как это сделать:

Размер пакета («Batch Size») – Количество примеров, обрабатываемых моделью за один шаг градиентного спуска:

Увеличение размера пакета может уменьшить шум в градиентных оценках и сделать обучение более стабильным, но требует больше памяти. Уменьшение размера пакета может помочь при ограниченной памяти и иногда улучшить обобщающую способность модели.

Скорость обучения («Learning Rate») – Коэффициент, определяющий величину шага, который модель делает в направлении уменьшения функции потерь при каждом обновлении весов:

Можно уменьшить скорость обучения, если модель колеблется и не сходится. Это поможет сделать шаги градиентного спуска менее резкими. Использование методов адаптивной регулировки скорости обучения, такие как «ReduceLROnPlateau» или циклическую скорость обучения.

Архитектура модели – Структура и компоненты модели, включая количество и типы слоев, количество нейронов в каждом слое и их соединения:

Изменение архитектуры модели может существенно повлиять на ее производительность. Мы можем добавить дополнительные слои, изменить количество нейронов в полносвязных слоях или использовать другие типы слоев. Замена базовой модели «MobileNet» на «DenseNet», «ResNet» или другую предобученную архитектуру может улучшить результаты.

Регуляризация – Техника для предотвращения переобучения путем добавления штрафа за сложность модели в функцию потерь:

Добавление или изменение параметров регуляризации («Dropout», «L2-регуляризация») может предотвратить переобучение. Экспериментируйте с разными значениями коэффициентов регуляризации.

Аугментация данных – Метод искусственного увеличения размера обучающего набора данных путем создания модифицированных версий существующих данных:

Можно расширить набор методов аугментации данных, таких как горизонтальное и вертикальное отражение, изменение яркости, контраста, добавление шума и т.д.

Количество эпох – Количество полных проходов через весь обучающий набор данных, которые модель делает во время обучения:

Увеличение количества эпох может позволить модели лучше обучиться, особенно если данные достаточно сложные. Использование EarlyStopping поможет прекратить обучение, когда улучшения в валидационных потерях прекращаются.

Оптимизаторы – Алгоритмы для обновления весов модели с целью минимизации функции потерь. Примеры включают «SGD», «Adam», «RMSprop»:

Экспериментирование с разными оптимизаторами, такими как «SGD» с импульсом, «RMSprop» или «Adam» с различными параметрами.

Инициализация весов – Метод задания начальных значений весов модели перед началом обучения:

Изменение схемы инициализации весов может иногда повлиять на сходимость модели.

# Ответы на контрольные вопросы

## Какие есть еще готовые нейронные сети, которые умеют выделять осмысленно информацию, кроме «MobileNet»?

Помимо «MobileNet», существует множество других предварительно обученных нейронных сетей, способных выделять осмысленную информацию из изображений. Вот некоторые из них:

«ResNet» («Residual Network»): эта сеть использует блоки с остаточным соединением, которые позволяют преодолевать проблему затухающего градиента и улучшают производительность на глубоких сетях.

«InceptionV3»: эта сеть использует различные размеры фильтров в одном слое и объединяет их результаты, чтобы извлечь информацию с различных уровней детализации.

«VGG» («Visual Geometry Group»): сеть, состоящая из множества сверточных слоев, которые существенно увеличивают количество параметров, но показывают высокую производительность в задачах классификации изображений.

«Xception»: эта сеть является вариантом «InceptionV3», где используются кросс-канальные корреляции для дальнейшего улучшения производительности.

«DenseNet»: сеть, в которой каждый слой связан с каждым другим слоем внутри блока. Это позволяет увеличить связность и использовать больше признаков из предыдущих слоев.

«EfficientNet»: это семейство сетей, оптимизированных по коэффициенту масштабирования, который балансирует глубину, ширину и разрешение сети для лучшего сочетания производительности и эффективности.

«NASNet» («Neural Architecture Search Network»): Сеть, созданная с использованием алгоритма поиска архитектуры нейронной сети, который самостоятельно находит наилучшую архитектуру для данной задачи.

## Как влияет количество эпох обучения на результат нейронной сети «MobileNet»?

Количество эпох обучения влияет на результат нейронной сети «MobileNet» и, в целом, на любую нейронную сеть. Вот как это происходит:

Недостаточное количество эпох. Если количество эпох недостаточно, модель может не успеть достаточно выучить закономерности в данных, что приведет к недообучению. В результате модель может быть недостаточно точной на тестовых данных.

Слишком большое количество эпох. Слишком много эпох обучения может привести к переобучению модели, когда она начинает выучивать шум в данных или особенности, специфичные для обучающего набора, и не может обобщить их на новые данные.

Оптимальное количество эпох. Идеальное количество эпох зависит от конкретной задачи, размера данных и сложности модели. Обычно для достижения оптимального результата требуется настройка гиперпараметров, таких как скорость обучения, регуляризация и другие.

Чтобы найти оптимальное количество эпох, рекомендуется провести эксперименты с разными значениями и оценить производительность модели на отложенной выборке или с помощью кросс-валидации. Обычно начинают с небольшого числа эпох и постепенно увеличивают до тех пор, пока производительность модели на валидационном наборе данных не начнет ухудшаться или стабилизироваться.

## Сортировка фото Автобуса и Миниавтобуса с помощью нейронной сети «DenseNet»

«DenseNet» и «MobileNet» — это два разных типа свёрточных нейронных сетей, предназначенных для различных задач и целей. Вот основные отличия между ними:

Архитектура:

«DenseNet» («Densely Connected Convolutional Networks»):

Связи между слоями: В «DenseNet» каждый слой подключен ко всем последующим слоям. Это означает, что выход каждого слоя становится входом для всех последующих слоев. Такая архитектура помогает в улучшении градиентного потока и использует повторно информацию.

Блоки «DenseBlock»: «DenseNet» состоит из нескольких «DenseBlock», между которыми находятся переходные слои («Transition Layers»). Каждый «DenseBlock» содержит несколько слоев, и каждый слой внутри блока соединён с каждым другим слоем.

Преимущества: обеспечивает лучшую передачу градиента через сеть, уменьшает проблему исчезающего градиента, уменьшает количество параметров и увеличивает вариативность в обучении.

«MobileNet»:

Свёртки с глубинным разбиением («Depthwise Separable Convolutions»): «MobileNet» использует глубинные свёртки и точечные свёртки («depthwise and pointwise convolutions») вместо стандартных свёрток, что значительно уменьшает количество параметров и вычислительные затраты.

Параметры α и ρ: «MobileNet» имеет два параметра для настройки модели — коэффициент ширины («width multiplier α») и коэффициент разрешения («resolution multiplier ρ»). Эти параметры позволяют изменять количество каналов в свёртках и размер входных изображений соответственно.

Преимущества: очень эффективна для мобильных и встраиваемых приложений, где вычислительные ресурсы и память ограничены. MobileNet оптимизирована для меньшего количества параметров и быстрого выполнения.

Использование и Применение:

«DenseNet»:

Применение: используется в задачах, требующих высокой точности классификации и других задач компьютерного зрения, таких как детекция объектов и сегментация изображений.

Область: подходит для задач, где наличие ресурсов (памяти и вычислительных мощностей) не является критичным ограничением.

«MobileNet»:

Применение: разработана для мобильных устройств и встраиваемых систем, где важно минимизировать время выполнения и потребление памяти.

Область: широко используется в приложениях реального времени, таких как распознавание лиц на смартфонах, мобильные приложения для обработки изображений и видео.

Параметры и Производительность

«DenseNet»:

Параметры: несмотря на обширные связи, «DenseNet» имеет меньше параметров, чем традиционные свёрточные сети той же глубины благодаря повторному использованию признаков.

Производительность: показала высокую точность на многих задачах и наборах данных, таких как «CIFAR» и «ImageNet».

«MobileNet»:

Параметры: значительно уменьшено количество параметров благодаря глубинным разбиениям, что делает её лёгкой и быстрой.

Производительность: хотя точность может немного уступать более крупным моделям, «MobileNet» всё ещё обеспечивает отличную производительность с учётом её небольшой вычислительной стоимости.

«DenseNet» и «MobileNet» имеют разные подходы к построению сетей и решению задач, что делает их подходящими для различных применений. «DenseNet» ориентирован на высокую точность и эффективность обучения, тогда как «MobileNet» сфокусирован на производительности и эффективности использования ресурсов, особенно на мобильных и встраиваемых устройствах.

Подгружаем нужные библиотеки (рисунок 19).

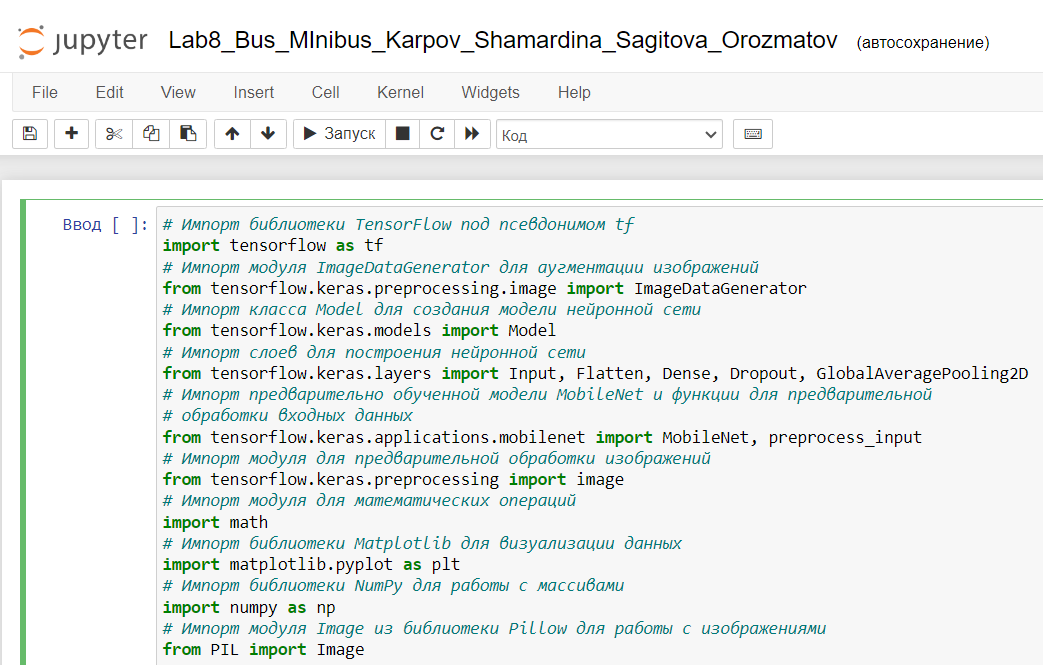


Рисунок 19 – Подгрузка нужных библиотек класс и модулей

Скачиваем 284 фотографий миниавтобусов и 316 фотографий автобусов для папки «train» и 156 фотографий для папки «test».

Константы используются при обучении модели для классификации изображений автобусов и миниавтобусов (рисунок 20).

«TRAIN\_DATA\_DIR» и «VALIDATION\_DATA\_DIR» – указывают пути к каталогам с обучающими данными и данными для валидации соответственно. Наша модель будет обучаться на данных из каталога Cars/train и валидироваться на данных из каталога Cars/test.

«TRAIN\_SAMPLES» и «VALIDATION\_SAMPLES» – эти константы определяют общее количество обучающих и валидационных примеров соответственно. Например, если «TRAIN\_SAMPLES» установлен в 1000, это означает, что у нас есть 1000 обучающих изображений автобусов и миниавтобусов в каталоге обучающих данных. Количество «TRAIN\_SAMPLES» может быть начальным выбором, который мы можем изменить в зависимости от результатов ваших экспериментов. Такое значения стоит брать, чтобы улучшить точность модели и её обобщающую способность.

«NUM\_CLASSES» – это количество классов, которые модель должна предсказывать. В нашем случае два класса: «автобусы» и «миниавтобусы».

«IMG\_WIDTH» и «IMG\_HEIGHT» – эти константы определяют размеры изображений. Обычно перед обучением изображения изменяются до этих размеров. В нашем случае, изображения будут изменены до размера 224x224 пикселя.

«BATCH\_SIZE» – это количество образцов, которые обрабатываются моделью за один шаг обучения. Например, если «BATCH\_SIZE» установлен в 5, то модель будет обновлять свои веса после каждых 5 образцов.

Эти константы помогают управлять процессом обучения модели, устанавливая параметры, такие как размеры изображений, размер пакета, количество классов и т. д.

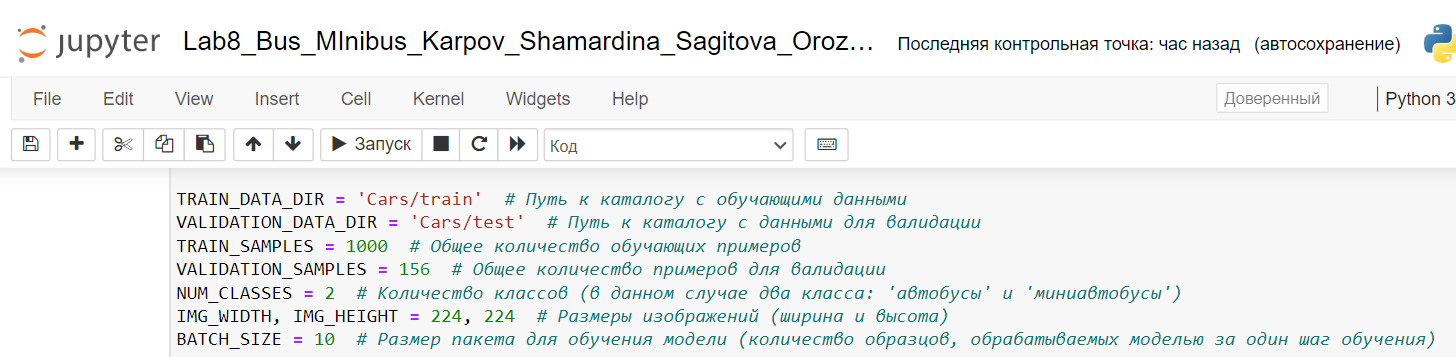


Рисунок 20 – Константы

Создаются два генератора данных (рисунок 21). Для обучения нейросети недостаточно пары сотен фотографий. Поэтому будет использоваться такой приём как аргументация.

Создаются два генератора данных (рисунок 22). Для обучения нейросети недостаточно пары сотен фотографий. Поэтому будет использоваться такой приём как аргументация.

Техника аргументации (или аугментации) данных — это процесс создания новых образцов данных путем применения различных случайных преобразований к существующим данным. В контексте обучения нейронной сети на изображениях это может включать в себя вращение, сдвиг, изменение масштаба и другие преобразования, которые могут помочь модели обучаться на более разнообразных и разносторонних данных.

В нашем коде используется «ImageDataGenerator» из библиотеки «Keras», который предоставляет мощные средства для аргументации изображений. Вот некоторые из параметров, которые мы установили:

«rotation\_range»: устанавливает диапазон углов для случайного вращения изображений.

«width\_shift\_range» и «height\_shift\_range»: определяют диапазон для случайного сдвига изображений по горизонтали и вертикали соответственно.

«zoom\_range»: устанавливает диапазон масштабирования изображений.

Эти преобразования помогают создать разнообразие изображений из ограниченного набора данных, что позволяет модели обучаться на более широком спектре условий и сценариев.

Генераторы данных («train\_generator» и «validation\_generator») используются для создания батчей данных из изображений в соответствующих каталогах, которые затем будут использоваться для обучения и валидации модели.

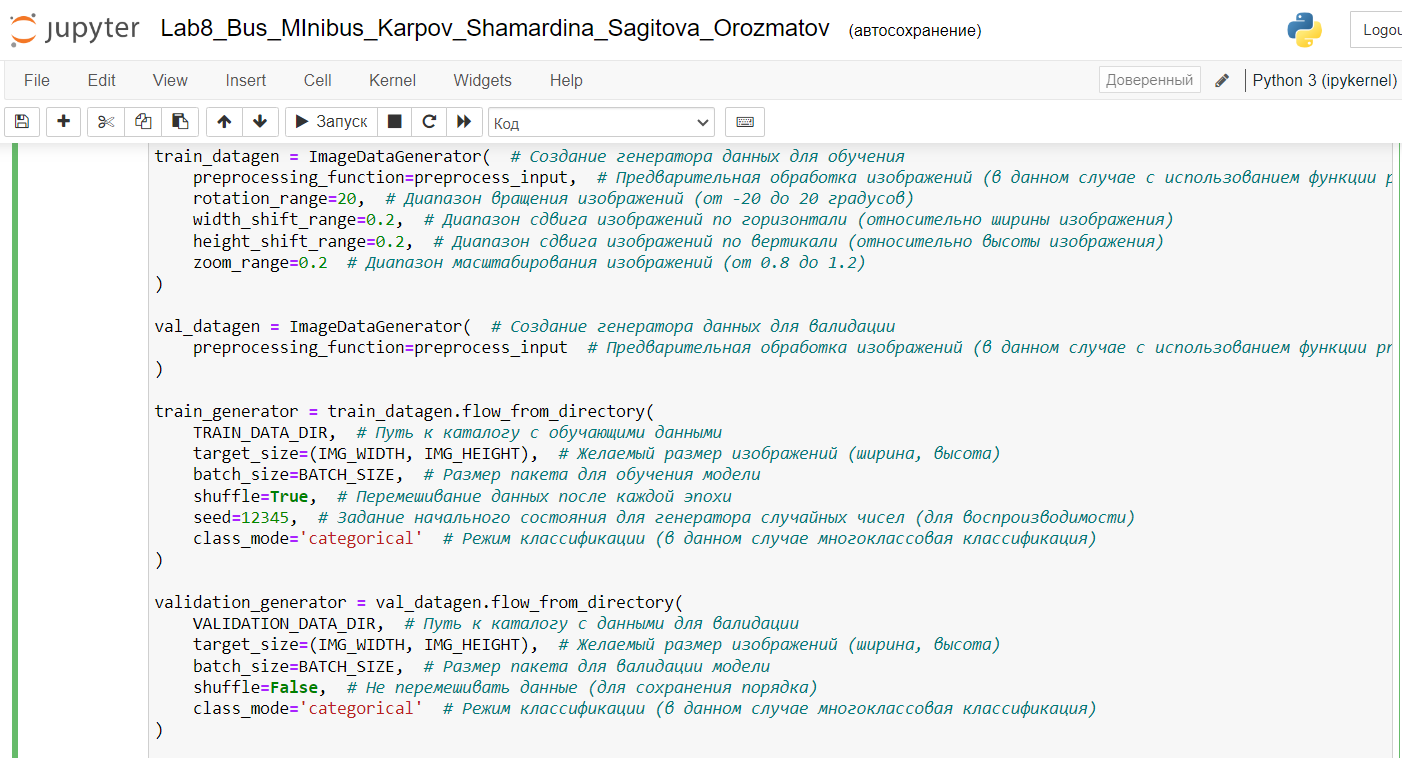


Рисунок 21 – Генераторы данных

Создание модели на основе нейронной сети, которая основана на готовой нейронной сети «DenseNet» (она называется базовой моделью)

В базовой модели нужно отключить обучение (для всех слоев устанавливается trainable = False).

Входное изображение поступает на вход базовой модели.

После базовой модели добавляется несколько своих слоев: слой пулинга «GlobalAveragePooling2D»; полносвязный слой из 60 нейронов «Dense»; используется механизм «Дропаут» (случайное отключение нейронных связей во время обучения, установим 0,5); последний слой должен выдавать два значения (вероятности того, что на фото автобус или миниавтобус – два класса).

Таким образом мы решаем задачу двухклассовой классификации (рисунок 22).

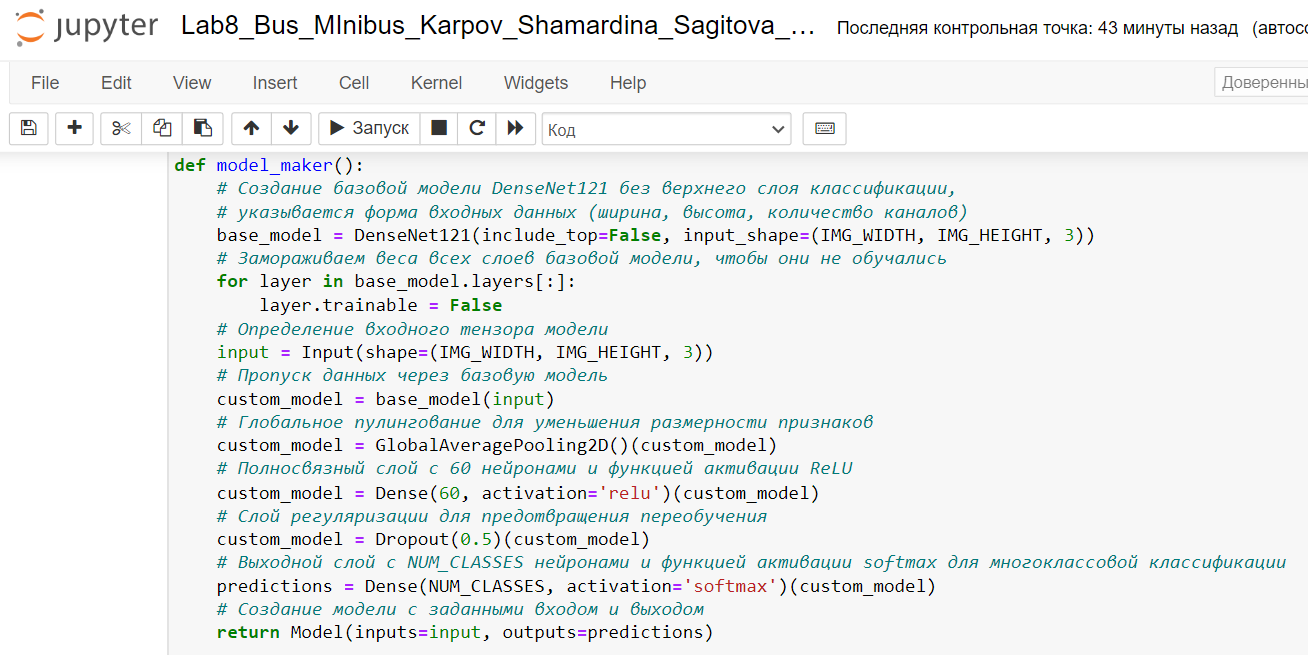


Рисунок 22 – Создание модели

Компилируем и обучаем модель, установим количество эпох 5 чтобы модель не переобучалась (рисунки 23-24).

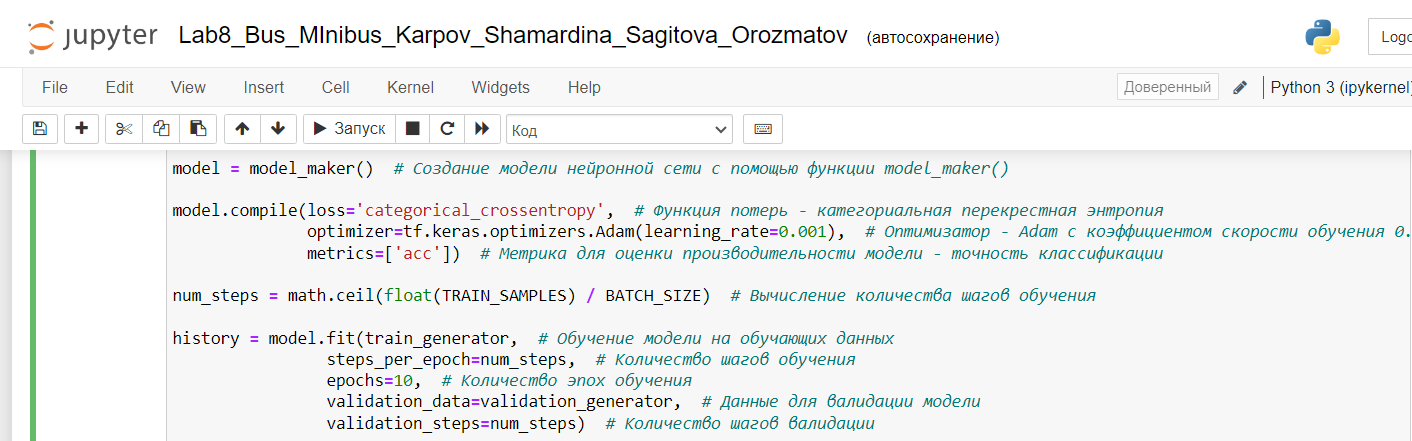


Рисунок 23 – Обучения модели (1)

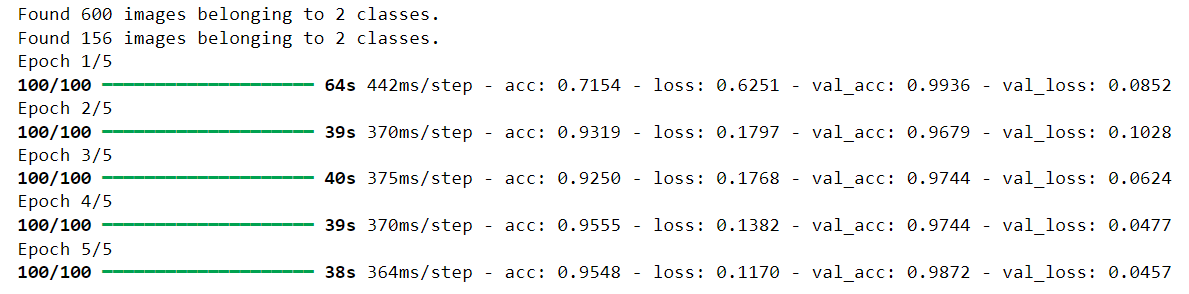


Рисунок 24 – Обучения модели (2)

Построения график (рисунок 25-27).

Визуализация функции потерь позволяет оценить эффективность обучения модели: мы стремимся к тому, чтобы функция потерь на обучающими валидационном наборах данных уменьшалась с каждой эпохой, что свидетельствует о том, что модель обучается эффективно и не переобучается. Теперь модель можно использовать и проверить, как она работает.

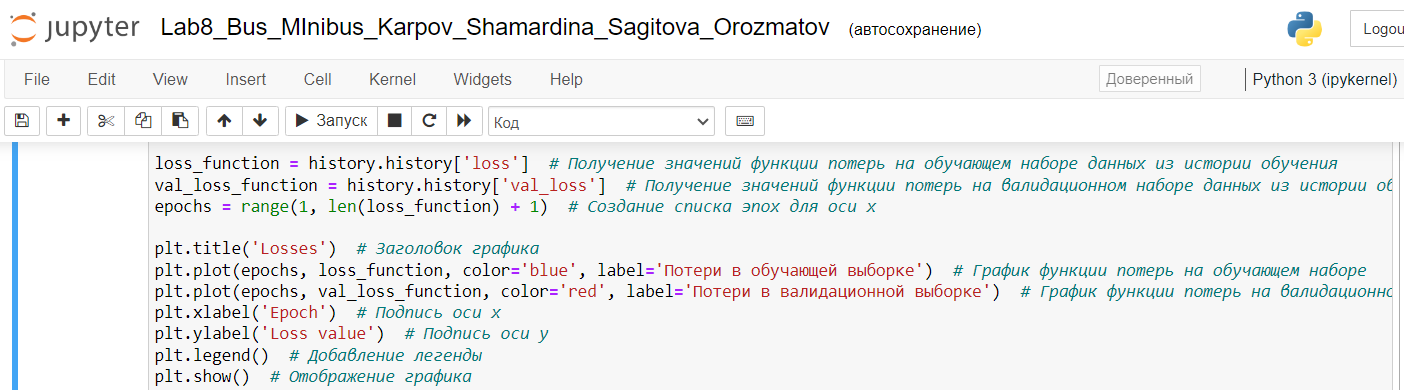


Рисунок 25 – Построения графика

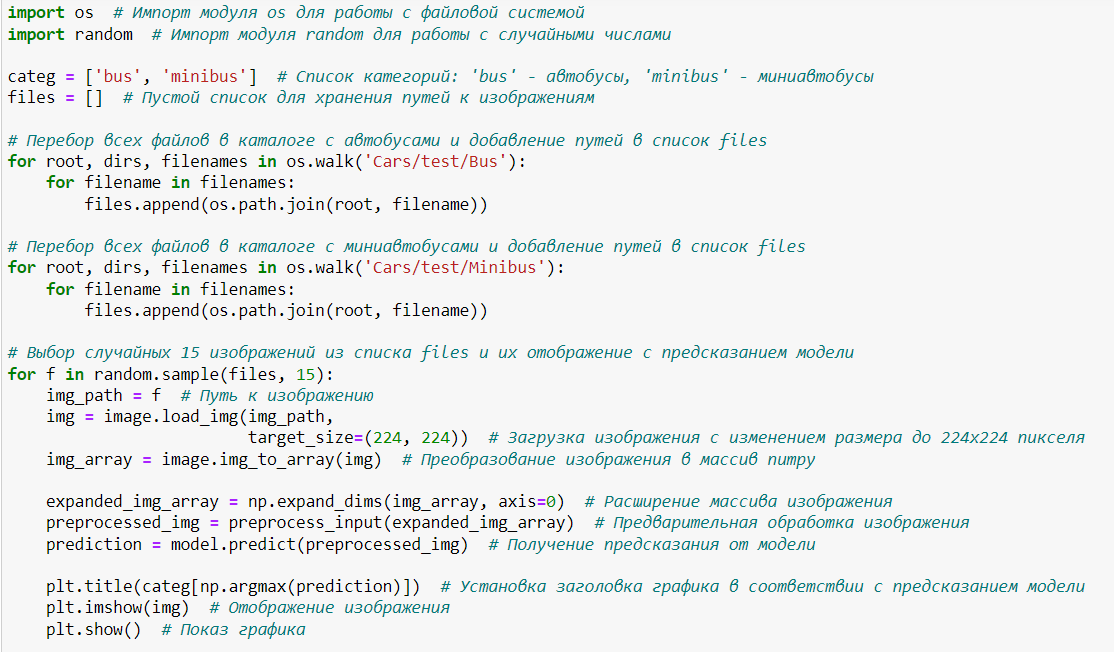


Рисунок 26 – Работа с изображениями

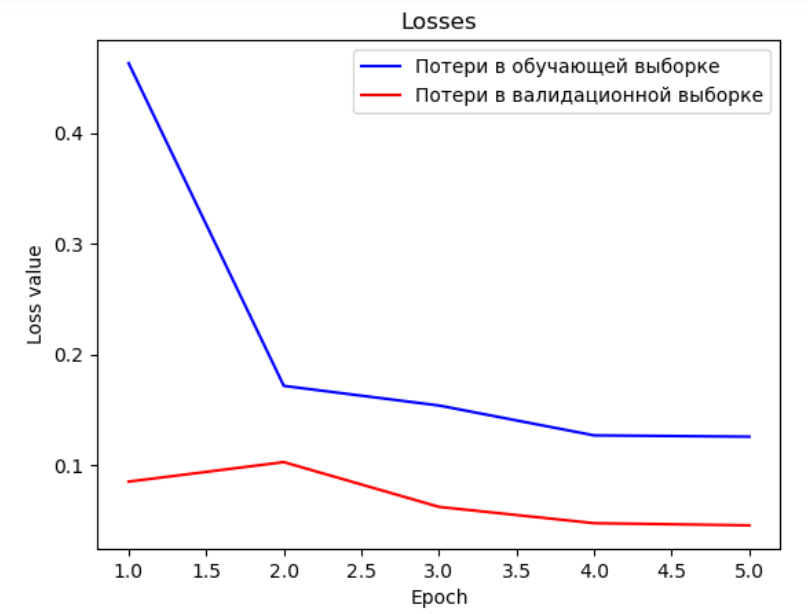


Рисунок 27 – График потерь

Получившиеся фотографии (рисунок 28).

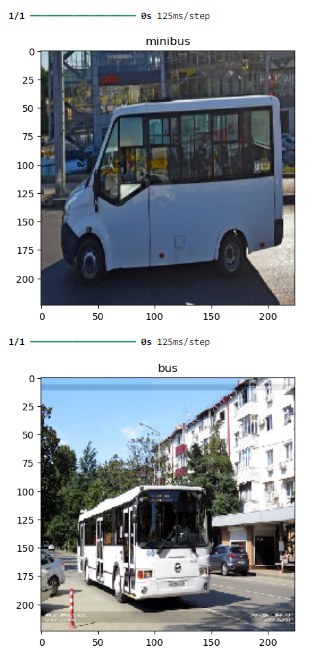


Рисунок 28 – Пример полученных фотографий

Вывод

В данной лабораторной работе мы познакомились с нейронными сетями написанных на языке программирования Python. Изучили различные тонкости и подходы в этих нейронных сетях, а также обучили нейросеть для сортировки фотографий.