**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ФАКУЛЬТЕТ РАДИОФИЗИКИ И КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ**

**Кафедра радиофизики и цифровых медиа технологий**

**ЛАБОРАТОРНЫЙ ПРАКТИКУМ**

**ПО ТЕМЕ: “АНАЛИЗ И ОБРАБОТКА МЕДИАДАННЫХ”**

**Методические указания**

Минск, 2024

1. ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ О МЕДИАДАННЫХ И АЛГОРИТМАХ ИХ ОБРАБОТКИ

## Общее представление об алгоритмах обработки и анализа медиаданных

В эпоху цифровых технологий медиаданные стали неотъемлемой частью информационного пространства. Эти данные, включающие изображения, звук и видео, представляют собой неисчерпаемый источник информации и творчества.

Медиаданные представляют собой информацию, сопровождающую цифровые медиафайлы, такие как изображения, аудио и видео. Эта информация включает в себя технические детали файла (формат, разрешение, длительность).

Рассмотрим основные виды медиаданных: аудио, изображения и видео.

Аудио — это форма медиаконтента, представляющая собой передачу звуковых сигналов или запись звуковой информации. В рамках аудио звуковые данные могут включать в себя музыку, речь, звуковые эффекты и другие акустические элементы.

Изображения представляет собой статические изображения для передачи информации или создания визуальных впечатлений. Изображение может быть представлено в различных форматах, включая фотографии, графику и рисунки.

Видео является формой медиаданных, которая включает в себя последовательность движущихся изображений, представленных с определенной частотой (кадры в секунду), сопровождаемую звуковой информацией или без нее. Видео может быть записанным, стриминговым или воспроизводимым в режиме реального времени.

Классификация медиаданных является важным этапом в обработке и управлении разнообразными типами медийной информации. Этот процесс позволяет системам и специалистам структурировать, организовывать и эффективно работать с медийными ресурсами. В рамках данной работы безусловно важно иметь общие знания и навыки для работы с разнообразными видами медиаданных, а также знать общеизвестные алгоритмы их обработки.

Алгоритмы обработки медиаконтента настолько разнообразны, как и сами медиаданные. Развитие в этой области успело пройти путь от квантования и семплирования до использования нейронных сетей глубокого обучения в анализе изображений или аудио. В современном информационном пространстве, где объем новых медиаданных за последний год сравним с объемом за последние 150 лет, алгоритмы стали неотъемлемой частью обработки данных, обогащая наше восприятие и взаимодействие с медийными форматами [1].

Начав с основополагающих принципов семплирования и квантования, алгоритмы обработки аудио-сигналов превратились в мощные инструменты для улучшения качества звука, распознавания речи и создания инновационных аудио эффектов. Развитие методов шумоподавления и алгоритмов сжатия привнесли в мир звука более чистые и эффективные решения работы с аудиосигналами. Приведенные ниже алгоритмы стали неотъемлемой частью их анализа:

**Шумоподавление:** Очистка аудио от ненужных шумов

В современном мире, где нас постоянно окружает шум, становится все более актуальной задача обеспечения чистого и разборчивого звучания. Алгоритмы шумоподавления выступают в роли умных помощников, способных очистить аудио от ненужных звуков, делая речь более понятной и приятной для восприятия. Ниже приведены наиболее значимые алгоритмы, позволяющие эффективно “чистить” аудио.

Представьте себе фоновый шум как отдельный "слой" на аудиозаписи. Spectral Subtraction работает подобно хитрому уборщику, который бережно снимает этот слой, не трогая саму речь. Алгоритм анализирует спектр сигнала, выделяя компоненты, характерные для шума, и затем вычитает их из оригинальной записи.

Метод Wiener Filtering основан на глубоком понимании природы шума. Алгоритм "знает", как звучит шум в записи, и использует эти знания для его эффективного отделения от речевого сигнала. Подобно тому, как мы учимся отличать голоса друзей от фонового шума в кафе, он умеет "прислушиваться" к речи, игнорируя посторонние звуки.

В отличие от стационарных шумов, динамичный шум постоянно меняется со временем и это становится важной проблемой шумоподавления в реальном времени. Adaptive Noise Cancellation – это алгоритм, который способен подстраиваться под любые изменения шумовой обстановки, постоянно анализируя окружающую среду, "узнавая" шум на лету.

Алгоритмы шумоподавления не просто убирают шум, но и сохраняют все важные детали речевого сигнала, такие как интонация, тембр голоса и эмоциональная окраска речи. Эффективность работы алгоритмов зависит от типа и интенсивности шума, а также от качества исходной записи, в тоже время современные технологии шумоподавления постоянно развиваются, предлагая все более совершенные методы очистки аудио.

Таким образом, алгоритмы шумоподавления играют важную роль в обеспечении чистого и разборчивого звучания, что имеет большое значение для различных областей, таких как образование, медицина, телекоммуникации и индустрия развлечений.

С появления цифровой фотографии как таковой инструментарий работы с изображениями не позволял произвести полный и качественный анализ изображений с высоким разрешением или в принципе большого количества изображений, однако, обработка изображений пережила революцию с появлением методов машинного зрения и глубокого обучения. Алгоритмы классификации изображений, детекции объектов и сегментации создают возможность автоматизированного анализа визуальных данных. Нейронные сети, воспроизводящие сложные механизмы восприятия, поднимают обработку изображений на новый уровень, обеспечивая точность и эффективность в решении различных задач [27].

В области компьютерного зрения, являющейся одним из ключевых направлений развития искусственного интеллекта, выделяются три фундаментальные задачи: классификация, детекция и сегментация изображений. Каждая из них имеет свою специфику и решается с помощью различных алгоритмов, основанных на глубоком обучении.

Классификация изображений, позволяющая определить принадлежность объекта к определенному классу, активно развивается благодаря использованию сверточных нейронных сетей (CNN). Современные архитектуры, такие как EfficientNet и RegNet, демонстрируют высокую точность и эффективность за счет оптимизации структуры сети и использования новых методов обучения. Оба семейства архитектур активно используют современные методы обучения, такие как прогрессивное изменение размера изображений, стохастическая глубина и обучение с учителем, что способствует повышению точности и скорости сходимости моделей.

Детекция объектов, направленная на определение местоположения и типа каждого объекта на изображении, также использует CNN в качестве основы. Однако для решения этой задачи применяются специализированные архитектуры, такие как RetinaNet и CenterNet, которые обеспечивают высокую точность и скорость обнаружения объектов в реальном времени. Важным направлением развития является также обнаружение объектов в условиях ограниченной видимости и сложных сцен [33].

Сегментация изображений, заключающаяся в разделении изображения на сегменты, выделяя объекты и их границы, также активно развивается благодаря использованию CNN. Современные подходы, такие как DeepLabv3+ и HRNet, позволяют достичь высокой точности сегментации даже на сложных изображениях. Особое внимание уделяется разработке алгоритмов, способных работать с изображениями высокого разрешения и обеспечивать сегментацию в реальном времени.

Таким образом, классические методы анализа и обработки изображений остаются актуальными и находят применение во многих областях, обеспечивая основу для сравнения и совмещения современных методов глубокого обучения. Они обогащают набор инструментов для разработки искусственного зрения и обработки визуальных данных. Применение вышеупомянутых алгоритмов находит широкое применение в различных областях, от медицины и автономного транспорта до робототехники и систем безопасности. Дальнейшее развитие компьютерного зрения будет определенно связано с созданием более точных и универсальных алгоритмов, способных решать сложные задачи в реальных условиях.

## Исследование сверточной нейронной сети (CNN)

Как было упомянуто в предыдущем пункте, сверточные нейронные сети являются основой для многих алгоритмов, которые ближе познакомят студентов с обработкой различных типов медиа и что делает их незаменимым инструментом для анализа визуального контента.

Практическая работа с CNN даст студентам возможность не только понять, как эти сети устроены, но и разобраться, как их использовать для решения различных задач. В лабораторных работах можно будет поэкспериментировать с классификацией картинок, поиском объектов на фото, а также с дообучением модели для распознавания дополнительных параметров. Это поможет изучающим набраться опыта и прокачать свои знания в анализе и обработке медиаданных.

Из указанного источника [8] следует:

Сonvolutional neural network (CNN, ConvNet), или Сверточная нейронная сеть — класс глубоких нейронных сетей, часто применяемый в анализе визуальных образов. Сверточные нейронные сети являются разновидностью многослойного персептрона с использованием операций свёртки. Они нашли применение в распознавании изображений и видео, рекомендательных системах, классификации изображений, NLP (natural language processing) и анализе временных рядов.

**Принцип работы операции свертки:**

Операцию свёртки можно представить следующим алгоритмом:

Скользящее окно, называемое фильтром, с размером (n, n) двигается по входному признаку. Количество движений определяется заданным количеством фильтров.

Каждый полученный шаблон имеет форму (n, n, d), где d — глубина входного признака.

Каждый шаблон умножается на своё ядро свёртки, в результате, формируется выходная карта признаков. Полученная выходная карта признаков имеет форму (h, w, N), где h и w — длина и ширина, полученные в результате отсечения, а N — количество фильтров.

Количество фильтров — гиперпараметр, поэтому выбирается самостоятельно. Обычно его подбирают как степень двойки с увеличением количества фильтров по мере увеличения глубины архитектуры. А ядра свёртки являются обучаемыми параметрами.

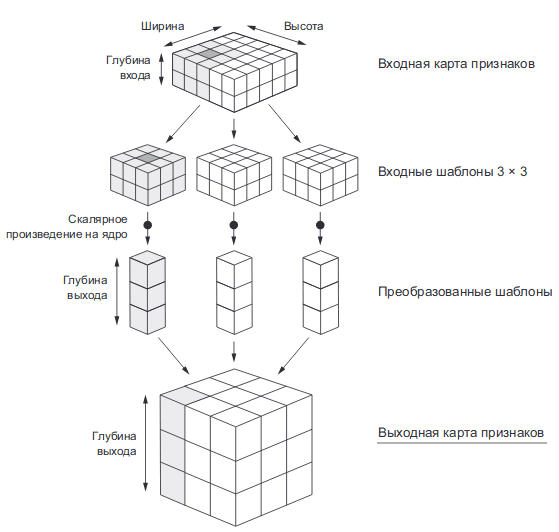


Рисунок 1.1 – Принцип действия операции свёртки

Рассмотрим процесс свёртки на примере изображения в оттенках серого с размером (28,28). Глубина изображения в оттенках серого равна 1, если бы это было RGB, то глубина входа равнялась бы 3. Пусть размер фильтра равняется (3,3), а всего их 32.

На первом этапе сформируются 32 шаблона размером (3,3,1), где 1 — глубина изображения.

Полученные шаблоны умножаются на ядра свертки. Каждый преобразованный в результате умножения шаблон формирует вектор с длиной равной количеству фильтров, т.е. 32.

Все преобразованные шаблоны объединяются в выходную карту признаков. Она имеет размер (26,26,32)

Почему уменьшается размерность после операции свёртки:

В рассмотренном выше примере выходная карта признаков имеет размерность (26,26,32), в то время как исходное изображение имело размерность (28,28,1). Если 32 — количество фильтров, а 1 — глубина входа, тогда почему исходный размер 28 уменьшился до 26?

Рассмотрим матрицу (5,5) и фильтр (3,3). Дело в том, что центр скользящего окна может встать только в 9 клеток матрицы (5,5), как это показано на рисунке ниже. Следовательно, после умножения на ядра свёртки сформируется выходная карта признаков с высотой и шириной (3,3).

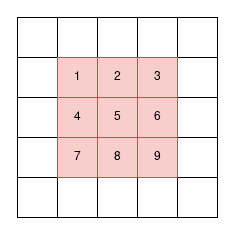


Рисунок 1.2 – Количество положений, куда может поместиться центр фильтра (3,3)

Для фильтра с размером (5,5) исходное изображение (28,28) уменьшилось бы в (24,24). Иногда такого обрезания можно избежать путём эффекта дополнения (padding). Он заключается в добавлении строк и столбцов так, что центр скользящего окна можно поместить в каждую клетку. Для фильтра (3,3) добавляются строки сверху и снизу и столбцы слева и справа. Для фильтра (5,5) добавляются по 2 строки снизу и сверху и 2 столбца слева и справа

**Слой Pooling:**

В сверточных нейронных сетях применяется ещё один слой, называемый слоем Pooling. Суть этого слоя заключается в уменьшении размерности карты признаков.

Pooling имеет две разновидности: max-pooling и average-pooling. В большинстве случаев применяется max-pooling. Операция Pooling схожа с операцией свертки:

Скользящее окно, обычно это окно (2,2), двигается по карте признаков.

Из выбранного шаблона выбирается максимальное (max-pooling) или среднее (average-pooling) значение. Формируется уменьшенная в размере карта признаков.

На рисунке 1.3 показано, как из матрицы (4,4) получается выходная карта (2,2) после операции max-pooling и average-pooling.

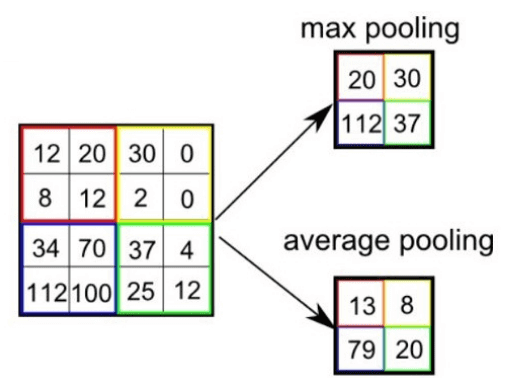


Рисунок 1.3 – Применение Pooling к матрице (4,4) с фильтром (2,2)

Зачем нужно уменьшать размерность с помощью Pooling?

Для поддержания иерархичности. Архитектура сверточных нейронных сетей похожа на воронку, где все начинается с большой картины с последующим углублением в отдельные детали. Человеческий мозг устроен также: сначала он видит на улице кошку, а затем начинает разглядывать ее цвет, пятна, уши, глаза и т.д. Это является основой Deep learning — обучение на представлениях.

Уменьшение размерности приводит к уменьшению количества обучаемых коэффициентов, поэтому это ещё и выигрыш в вычислительных ресурсах [8].

## Исследование алгоритмов сжатия и восстановления аудиосигналов

Владение алгоритмами сжатия и восстановления аудиосигналов становится неотъемлемым навыком для специалистов в области обработки звука. Эти навыки имеют ключевое значение в различных сценариях и предоставляют специалистам ценный инструментарий для эффективного управления аудиоинформацией.

Первая причина безусловно связана с экономией места хранения: Специалисты должны уметь оптимизировать пространство хранения, особенно при работе с большим объемом аудиофайлов. Алгоритмы сжатия позволяют значительно уменьшить размер файлов, сохраняя при этом качество звука, что является важным фактором в условиях ограниченных ресурсов. Вместе с экономией, важнейшей проблемой становится ускорение передачи данных, ведь при передаче аудиоинформации по сети, будь то веб-стриминг, трансляции или обмен данными, эффективные методы сжатия существенно сокращают временные задержки и обеспечивают более быструю передачу данных.

Сохранение качества звука также является важной причиной, почему необходимо уметь работать с аудиосигналами. Восстановление аудиосигналов после сжатия становится важным аспектом для поддержания высокого качества звучания. Специалисты должны быть способными выбирать и настраивать алгоритмы восстановления в зависимости от требований конкретного проекта. В этом вопросе очень важна точность, т.к. эффективно обработанный аудиофайл помогает лучше понять процесс и позволяет сделать более точные выводы.

Одной из важнейших причин начать углублённое изучение методов сжатия и восстановления – это работа с аудио в реальном времени, т.к. в некоторых областях, таких как аудио обработка в реальном времени, знание алгоритмов сжатия и восстановления является критическим. Это может быть актуально в области звукового дизайна, распознавания и синтез речи, биоакустический анализ и других приложениях, где задержка недопустима.

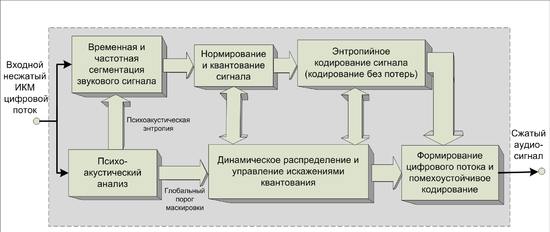
Таким образом, умение эффективно использовать алгоритмы сжатия и восстановления аудиосигналов становится важным навыком, применимым в области обработки звука, обеспечивая оптимальное управление аудиоинформацией в различных сценариях и проектах. На рисунке 1.4 изображено как работает большинство современных кодеров, позволяющих проводить необходимые работы со звуком, в частности сжатие.

Рисунок 1.4 Обобщенная структура кодера с компрессией цифровых аудиоданных

MP3 (MPEG Audio Layer III):

MP3 представляет собой алгоритм сжатия аудио с использованием метода потерь. Он основан на псих-акустической модели, которая учитывает характеристики человеческого слуха. Алгоритм разбивает аудиосигнал на временные окна и применяет преобразование Фурье для перевода их в частотную область. Затем применяется квантование, в результате чего часть данных, считаемых менее важными с точки зрения восприятия слухом, отбрасывается. Псих-акустические модели также используются для идентификации тех частей аудиосигнала, которые могут быть без значительной потери удалены. Такой метод сжатия позволяет значительно уменьшить размер аудиофайлов при воспроизведении, сохраняя воспринимаемое человеческим ухом качество звука [7].

AAC (Advanced Audio Coding):

AAC также является методом сжатия аудио с потерями, представляющим более эффективную альтернативу MP3. Он использует более сложные методы кодирования, такие как преобразование MDCT (Modified Discrete Cosine Transform) для преобразования аудиосигнала. Дополнительно, AAC применяет методы предсказания и кодирования взаимозависимостей между временными окнами, что позволяет дополнительно сжимать данные. Как и MP3, AAC использует псих акустические модели для определения, какие части аудиосигнала могут быть удалены без ущерба для восприятия звука человеком. Этот формат обеспечивает более высокое качество при более низкой битрейте по сравнению с MP3 [7].

FLAC (Free Lossless Audio Codec):

FLAC представляет собой метод сжатия аудио без потерь, сохраняя полную информацию об оригинальном сигнале. Алгоритм FLAC использует представление сигнала в виде прогрессивного преобразования Предиктивного Кодирования Райса, что позволяет эффективно уменьшить размер файла. В отличие от методов с потерями, где определенные части аудиосигнала удаляются, FLAC сохраняет каждый бит оригинального сигнала. Это особенно важно в профессиональных приложениях, где высокое качество звука обязательно [30].

Каждый из этих алгоритмов опирается на различные физические принципы и математические модели для эффективного сжатия аудиосигналов. Психоакустические модели, преобразование Фурье и прогрессивные методы кодирования играют ключевую роль в достижении высокой степени сжатия при сохранении воспринимаемого качества звука.

1. ИЗУЧЕНИЕ ИНСТРУМЕНТОВ ПРАКТИКУМА

В современных реалиях выбор инструментов для проведения практикума является одной из главных проблем, при подготовке к созданию такового. Каждый инструмент, будь то язык разработки или библиотека, должен быть простым в понимании, но при этом иметь полный функционал, для обеспечения оптимальной работы с доступными ресурсами.

## Функции Google Colaboratory и преимущества сервиса

Google Colaboratory, более известная как «Colab», — бесплатная платформа для блокнотов Jupyter.

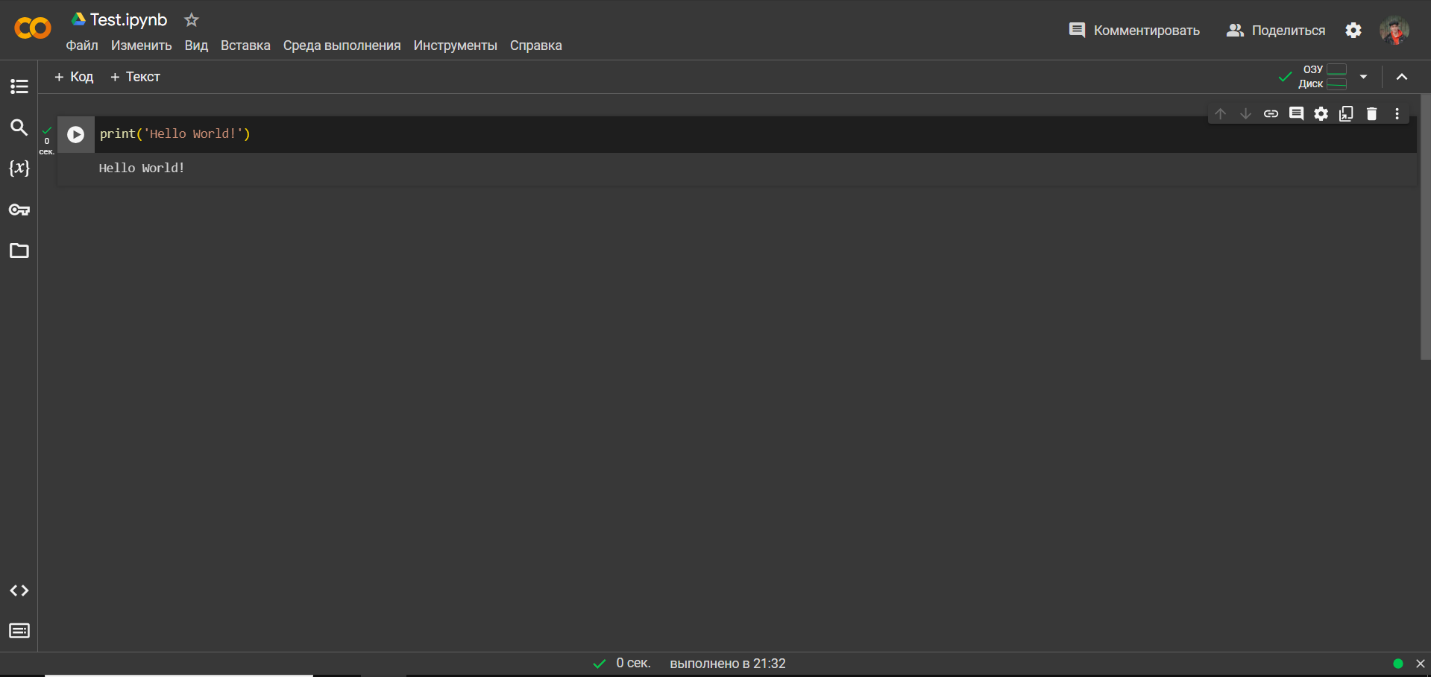
Кроме среды запуска блокнотов Python и R, Colab позволяет совместно использовать свободный доступ к ограниченному количеству GPU и TPU.

Рисунок 2.1 – Среда разработки Google Colaboratory.

Принцип у нее такой же, как у остальных онлайн-офисов компании: она позволяет одновременно с коллегами работать с данными. В Google Colab можно применять различные библиотеки на Python, загружать и запускать файлы, анализировать данные и получать результаты в браузере. Этот сервис особенно полезен для разработчиков и студентов, изучающих программирование на Python. Здесь те же ячейки (cells), которые поддерживают текст, формулы, изображения, разметку HTML и не только. То есть можно заниматься программированием на языке Python и не качать лишние файлы, кучу библиотек, не перегружать машину и не переживать, что место на жестком диске вот-вот закончится. Единственное условие — нужно иметь Google-аккаунт.

* Доступ к Colab: для доступа к Colab необходим только браузер и аккаунт Google.
* Создание и выполнение ячеек кода: Colab организован посредством ячеек, в которых можно выполнять код пошагово. Это удобно для экспериментов и обучения.
* Совместная работа: Colab позволяет совместно редактировать и комментировать документы, что упрощает работу в команде.
* Интеграция с Google Drive: Проекты Colab автоматически сохраняются в Google Drive, обеспечивая удобство хранения и совместного доступа.

Определив основные пункты для начала работы с Google Colab, можно предварительно зарегистрировать аккаунт и протестировать платформу. Также, студенты самостоятельно должны изучить возможности и основы работ с платформой, т.к. дальнейшие работы будут предполагать, что студенты умеют: работать со средами выполнения, подключать собственный Google Drive c использованием собственных выборок данных, а также уметь работать с различными типами ячеек в среде разработки.

## Библиотеки языка программирования Python для работы с медиа данными

При выборе библиотеки для работы с медиа данными необходимо учитывать следующие факторы:

* Поддержка задач практикума: библиотеки должны предоставлять необходимые функции, методы и классы для решения задач, которые будут выполняться в рамках лабораторных.
* Доступность: библиотеки должны быть доступны для использования студентами, а именно быть общедоступными, бесплатными и иметь возможность подключаться в любой проект.
* Сложность: библиотеки должны быть достаточно простыми для освоения студентами.

Именно поэтому были определены следующие библиотеки: Keras, TensorFlow, Matplotlib, Numpy, Librosa.

Keras – простая и удобная в использовании библиотека для создания и обучения нейронных сетей. Она может использоваться для выполнения различных задач анализа медиаданных, таких как обнаружение объектов, распознавание лиц и классификация изображений [15].

TensorFlow является более мощной библиотекой для машинного обучения, чем Keras. Одним из главных преимуществ TensorFlow является его возможность обучать глубокие нейронные сети. Он позволяет создавать сложные модели с использованием различных архитектур. Библиотека также предлагает множество предварительно обученных моделей, которые могут быть использованы для таких более простых задач, как классификация изображений, распознавание речи, обработка естественного языка [25].

Matplotlib – это популярная библиотека для визуализации данных. Она может использоваться для отображения результатов анализа медиаданных, таких как изображения, графики и диаграммы [14].

NumPy считается эффективной библиотекой для работы с многомерными массивами, предоставляя математический аппарат для произведения различных операций свертки и расчета коэффициентов и весов. Она может использоваться для хранения и обработки данных медиаданных, таких как изображения, видео и аудио [19].

Librosa — особая библиотека для обработки аудио, которая ориентирована на анализ музыки. Она может использоваться для выполнения различных задач анализа аудио, таких как распознавание речи, определение тона и обнаружение ритма [17].

## Архитектура нейронной сети EfficientNet B7

EfficientNets – это семейство моделей классификации изображений, которые обеспечивают высочайшую точность, но на порядок меньше и быстрее, чем предыдущие модели. EfficientNet была разработана командой исследователей компании Google и впервые представлена в 2019 году. Основная идея архитектуры заключается в масштабировании модели, при котором увеличивается не только количество слоев (глубина), но также и ширина слоев, и разрешение входных данных. Это позволяет достигать оптимального баланса между точностью и производительностью модели. В частности, сначала была использована AutoML Mnas Mobile framework, представляющая собой платформу с открытым исходным кодом для автоматизированного проектирования и генерации мобильных моделей нейронных сетей и машинного обучения, для разработки базовой сети мобильного размера, названной EfficientNet-b0; затем был использован метод составного масштабирования, чтобы увеличить этот базовый уровень, чтобы получить EfficientNet-b1 до B7 [9]. Как можно заметить из рисунка 2.2, “поколение” моделей EfficientNet значительно превосходит модели-конкуренты, обеспечивая выигрыш как по количеству параметров, так и по точности.

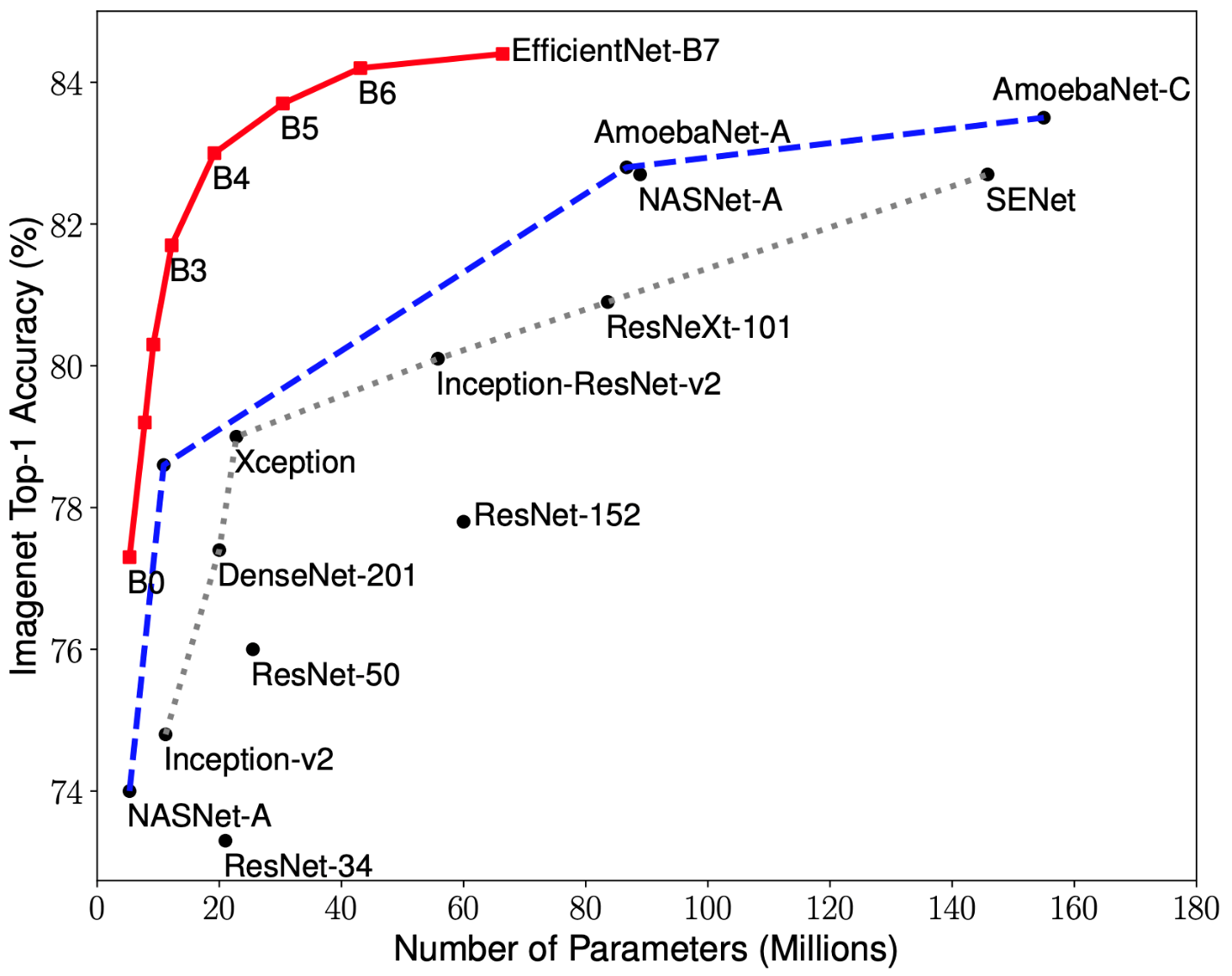
По оси Х: Количество параметров (миллионов), по оси Y: Оценка точности моделей компьютерного зрения (в процентах) [9]

Рисунок 2.2 – Размер модели в зависимости от точности [9]

Эффективные сети значительно превосходят по производительности другие сети. В частности, EfficientNet-B7 обеспечивает новую, самую современную точность - 84,3%, но в 8,4 раза меньше и в 6,1 раза быстрее, чем модель GPipe. EfficientNet-B1 в 7,6 раз меньше и в 5,7 раз быстрее, чем ResNet-152, которая считалась в свое время верхом скорости производительности и точности. Обучение модели EfficientNet в ImageNet [12], использует аналогичные настройки, такие как: оптимизатор RMSProp с затуханием 0,9 и импульсом 0,9; пакетная норма импульса 0,99 с затуханием веса 1e-5; начальная скорость обучения 0,256 с затуханием на 0,97 каждые 2,4 эпохи [18]. Также используется силовую активацию, автоаугментацию и стохастическую глубину с вероятностью выживания 0,8. Поскольку общеизвестно, что более крупные модели нуждаются в большей регуляризации, в архитектуре линейно увеличивается коэффициент отсева с 0,2 для EfficientNet-B0 до 0,5 для B7. Рекомендуемый процесс обучения заключается в следующем: резервируется 25 тысяч случайно выбранных изображений из обучающего набора в качестве минимального набора и выполняется ранняя остановка для этого мини-набора; затем оценивается контрольную точку ранней остановки в исходном наборе проверки как сообщите об окончательной точности проверки. Однако ввиду сложности самого процесса, модель изначально корректируется для прикладного применения в рамках практикума.

Основными строительными блоками EfficientNetB7 являются модули MBConv. Эти модули используют комбинацию сверточных операций с различными размерами ядра, а также операции сжатия и расширения каналов для достижения оптимального баланса между точностью и эффективностью. fficientNetB7 использует метод Compound Scaling для масштабирования модели. Этот метод пропорционально увеличивает разрешение изображения, глубину сети и количество каналов при сохранении баланса между точностью и эффективностью.

Одной из удивительных особенностей этой модели является предложенный проектировщиками новый метод комплексного масштабирования, который использует комплексный коэффициент φ для равномерного масштабирования ширины, глубины и разрешения сети принципиальным образом:

Глубина:

d = αφ

Ширина:

w = βφ

Разрешение:

r = γφ

где α · β2 · γ2 ≈ 2, α ≥ 1, β ≥ 1, γ ≥ 1 и α, β, γ – константы, которые могут быть определены с помощью поиска по небольшой сетке. φ – это заданный пользователем коэффициент

Важно заметить, что φ – пользовательский коэффициент, который определяет, сколько дополнительных ресурсов доступно для масштабирования модели, в то время как α, β, γ определяют, как соотносить эти дополнительные ресурсы с шириной, глубиной и разрешением сети. Примечательно, что количество провалов обычной операции свертки пропорционально d, w2, r2, т.е. удвоение глубины сети приведет к удвоению количества провалов, но удвоение ширины сети или разрешения увеличит количество провалов в четыре раза. Поскольку операции свертки обычно “доминируют” в “стоимости вычислений” в ConvNet, масштабирование ConvNet с помощью вышеуказанного уравнения примерно увеличит общее количество провалов на

(α · β2 · γ2)φ

Резюмируя, EfficientNetB7 является мощной и универсальной архитектурой СНН, которая может быть использована для различных задач классификации и компьютерного зрения, что будет весьма кстати при подготовке лабораторной работы по классификации изображений. В рамках лабораторного практикума студентам будет предложено решить задачу классификации изображений, используя архитектуру EfficientNet. Основная цель — продемонстрировать эффективность и применимость данной модели в реальных сценариях обработки медиаданных, решая задачу распознавания и дальнейшей классификации.

## Описание наборов данных для решения задач классификации в лабораторном практикуме

В рамках лабораторного практикума важное место занимает работа с набором данных, на основе которых строятся модели машинного обучения для решения задач классификации. Правильный выбор и подготовка данных являются ключевыми аспектами, влияющими на качество и точность полученных моделей. Таким образом, важно подобрать такие дата сеты, которые не вызовут излишней перегрузки ресурсов машины, на основе которой и будет проходить обучение, но и не будут недостаточно полными, чтобы избежать переобучения и повысить точность и эффективность.

Для двух лабораторных работ будет использовано три дата сета.

**Набор данных Animal-10**

Представляет собой тщательно подобранную коллекцию изображений десяти различных видов животных. Этот дата сет включает в себя тысячи изображений, которые были классифицированы вручную и предназначены для обучения и тестирования моделей машинного обучения [3]. На основе такого количества данных грамотно спроектированная сеть будет показывать точные результаты, в чем можно будет убедиться, запустив обучение модели всего лишь на 15 эпохах, представляющих собой один полный проход обучения по всему набору данных. Основные характеристики набора данных Animal-10 включают:

* 10 видов животных, т.е. классов (собака, кошка, лошадь, паук, бабочка, курица, овца, корова, белка, слон).
* Более 28,000 изображений высокого качества, размером 300х169 пикселей. Общий размер дата сета составил 614 Мб.
* Основное предназначение – обучение и тестирование моделей для решения задач классификации животных.

По основным характеристикам можно сказать, что такой дата сет хранит в себе изображения, подходящие под задачу дообучения, т.к. набор классов охватывают широкий спектр животных, включая как домашних, так и диких представителей фауны, а сами изображения тщательно отобраны и аннотированы вручную для обеспечения высокого качества и точности, при этом имея высокое разрешение и разнообразные ракурсы, что позволяет моделям лучше обучаться и справляться с вариативностью реальных данных.

На рисунке ниже приведено распределение количества изображений в каждом из десяти классов:

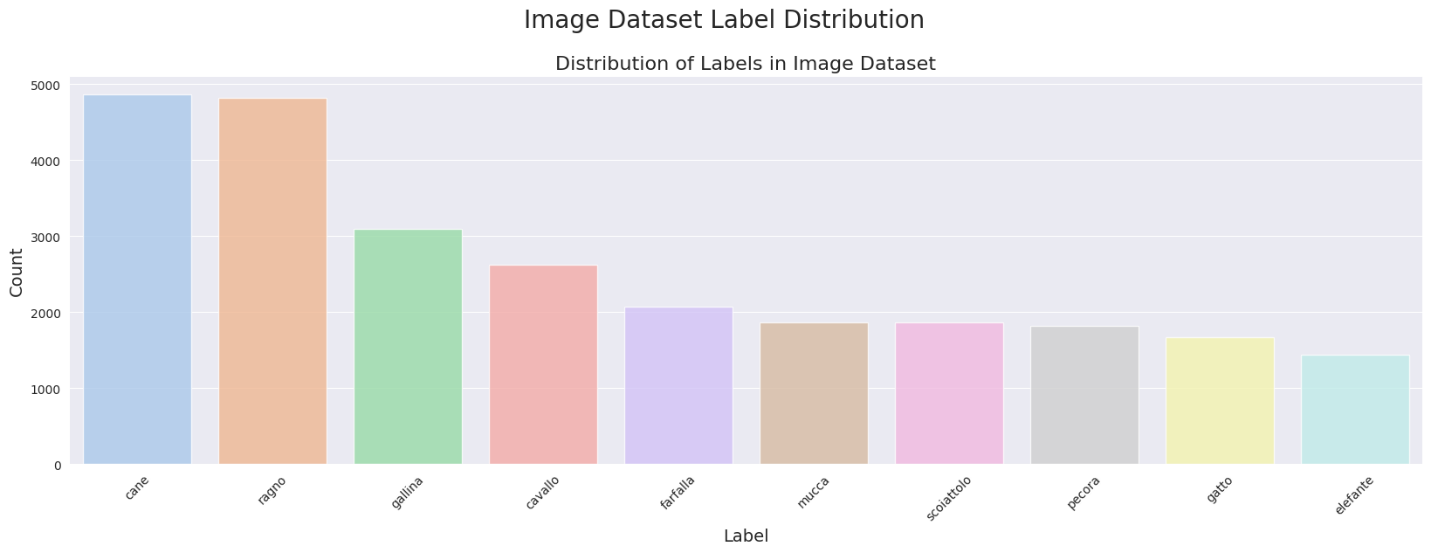
По оси Х слева направо: собака, паук, курица, конь, бабочка, корова, белка, овца, кошка, слон; По оси Y – количество изображений в категории.

Рисунок 2.3 – Распределение данных по категориям в наборе данных Animals-10

**Набор данных Fashion-MNIST**

Fashion-MNIST – это набор данных, представляет собой коллекцию изображений предметов одежды и аксессуаров, и является популярным выбором для обучения и тестирования моделей машинного обучения в задачах классификации изображений, состоящий из изображений размером 28×28 в оттенках серого 70 000 модных товаров из 10 категорий, по 7000 изображений в каждой категории. Обучающий набор содержит 60 000 изображений, а тестовый - 10 000. Fashion-MNIST использует тот же размер изображений, формат данных и структуру разделов для обучения и тестирования, что и оригинальный MNIST [23]. Набор данных Fashion-MNIST был создан в качестве более сложной и актуальной альтернативы классическому набору данных MNIST, содержащему изображения рукописных цифр.

Дата сет используется как стандартный тестовый набор данных для проверки и отладки моделей машинного обучения, например подстройки весов или корректировка слоев. Благодаря своей популярности и доступности, этот набор данных позволяет исследователям и разработчикам сравнивать результаты своих моделей с результатами других моделей и алгоритмов. Fashion-MNIST также часто используется для обучения моделей начального уровня, прежде чем переходить к более сложным задачам и наборам данных, что будет полезно для студентов, которые только знакомятся с нейронными сетями и методами машинного обучения в целом.

На рисунке 2.4 представлены изображения во всех десяти классах одежды.

Рисунок 2.4 – Дата сет Fashion-MNIST в картинках [23]

**Набор данных SoftToy-1**

Пользовательский дата сет SoftToy-1 был специально создан мной для целей данного лабораторного практикума, для подробного изучения процесса дообучения нейронной сети. SoftToy-1 — это дата сет, состоящий из более 2000 изображений плюшевых игрушек животных, разделенных на десять классов животных: дата сет содержит изображения плюшевых слонов, медведей, леопардов, быков, жирафов, свиней, скунсов, щенят, морских львов и овец. Эти классы охватывают широкий спектр мягких игрушек, позволяя моделям освоить разнообразие форм и текстур.

Источником данных послужили различные источники, включая интернет-магазины, личные коллекции и общественные библиотеки изображений. Изображения предварительно были аугментированы, что позволило увеличить количество изображений и добавить реалистичности, вариативности и точности при обучении модели.

Использование подобного набора данных поможет получить опыт в процессе дообучение, а также познакомит студентов с уникальным процессом составления и дополнения, существующего дата сета. Таким образом, составляя для изучающих практикум полную картину работы любого классификатора на базе нейронных сетей и определяя важность подбирать качественные изображения на каждом из этапе работы с нейросетью – тренировка, валидация и тестирование.

1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ЛАБОРАТОРНОГО ПРАКТИКУМА

Лабораторный практикум по анализу и обработке медиаданных ставят перед студентами ряд требований, нацеленных на развитие не только технических, но и широкого спектра навыков. Основная цель практикума – предоставить студентам практический опыт работы с медиа данными, начиная с базовых концепций и заканчивая решением конкретных задач. В рамках выполняемой работы студентам будут предложены работы, основной целью которых является изучение основ работы с изображениями и аудио на базе АМО.

## Формулировка задач практикума

Формулируя задачи практикума важно учитывать навыки, которые получат студенты после его выполнения. Также важным аспектом будет планирование сроков выполнения с учетом теоретической части предмета, т.к. может потребоваться время для понимания основ нейросетей.

Основная цель практикума – предоставить студентам практический опыт работы с медиа данными, начиная с базовых концепций и заканчивая решением конкретных задач.

По итогам лабораторного практикума студенты:

* Освоят основные методы анализа и обработки медиа данных, а также узнают о современных средах разработки и актуальных инструментах для работы с подобными типами данных.
* Приобретут навыки работы в среде разработки Google Colab, изучат тонкости библиотек Python, таких как TensorFlow, Keras, Matplotlib, Pydub и другие; получат базовые представления об устройстве нейронных сетей и алгоритмов сжатия.
* Смогут применять полученные знания для решения реальных задач в различных областях, что может подтолкнуть на создание собственного проекта в той или иной сфере.

Важным аспектом выполнения практикума станет формирование отчетов о проделанной работе, с дальнейшей презентацией, т.к. работы предполагают улучшение существующих моделей путем уникальных преобразований. Несмотря на использование отчетов повсеместно, в рамках практикума студенты могут получить важный навык презентации, задействуя основные метрики оценки эффективности, точности и скорости алгоритмов.

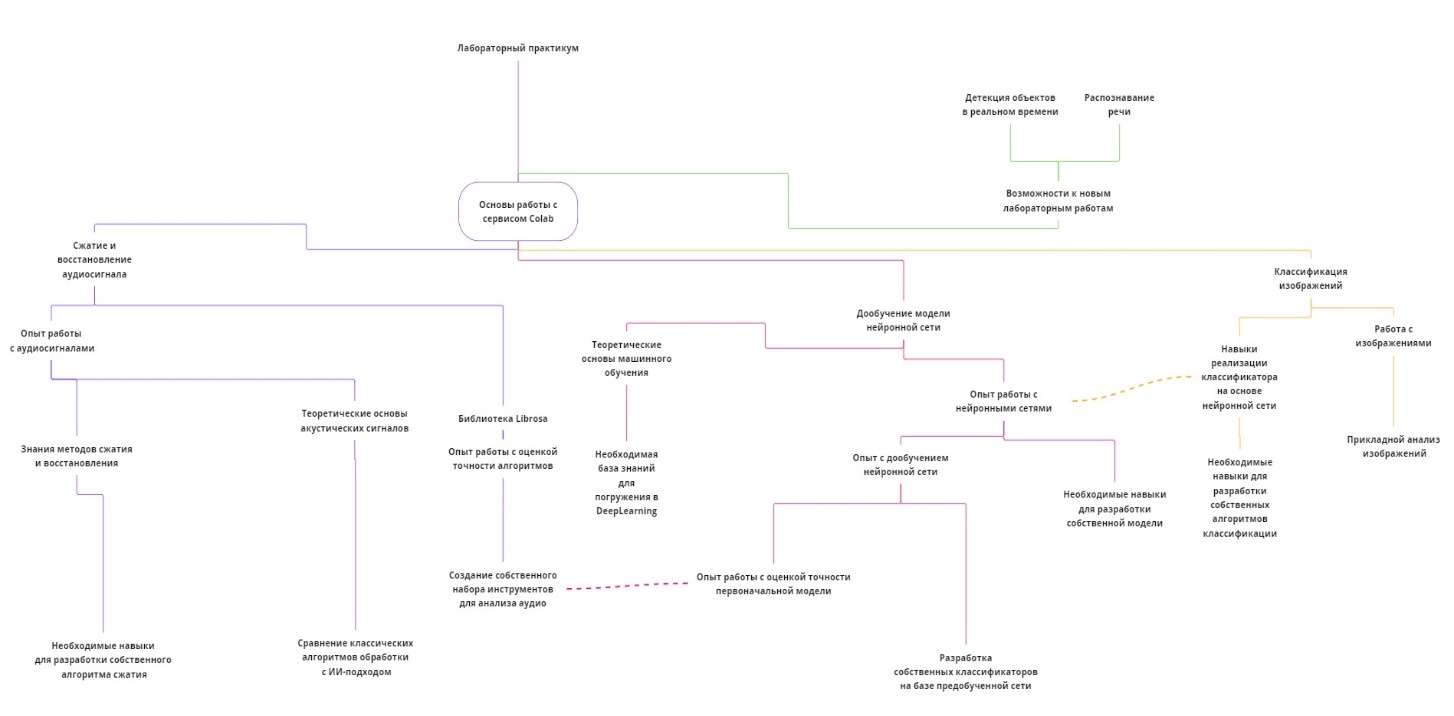
На рисунке 3.1 представлены темы лабораторных работ и навыки, полученные при их выполнении, а также возможное развитие самих работ. Каждая “ветвь дерева” будет изучена подробнее в следующем пункте.

Рисунок 3.1 – Блок-схема полученных навыков студентами

## Темы лабораторных работ

На основе проделанной формулировки задач, проделанной в пункте 3.1, мною были определены темы для четырех лабораторных работ:

1. **Введение в Google Colab и Python для обработки медиа данных.**

В данной работе в рамках теоретической будет предложено ознакомиться с возможностями Google Colab, описаны основные библиотеки для дальнейшей обработки медиа.

Практическая часть будет включать в себя создание собственного аккаунта в Colab и знакомства с основами работы: подключение внешних данных, знакомство с Pandas DataFrame и визуализация данных. Для выполнения поставленных задач требуется базовые знания Python и опыт работы с Jupyter Notebooks.

1. **Классификация изображений на базе предобученной модели библиотеки Keras.**

Теория для второй работы включает описание основ архитектуры СНН, знакомство с библиотеками для машинного обучения TensorFlow и Keras и пример работы классификации на основе одной из предобученных моделей библиотеки Keras.

В рамках практической части мною предложено построить графики тренировочной и валидационной метрик и изучить способности такой модели на изображениях разрешениям в 10 раз больше, чем изображения, на которых модель обучалась, а в случае необходимости скорректировать слои модели.

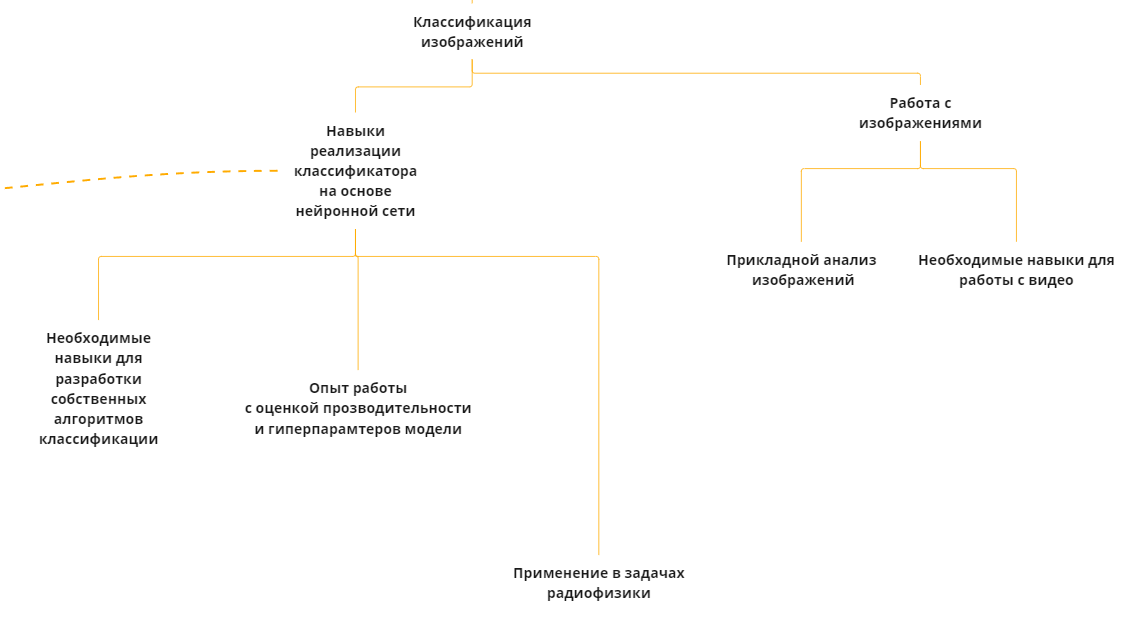
Рисунок 3.2 описывает блок-схему полученных навыков при выполнении работы.

Рисунок 3.2 – Блок-схема навыков лабораторной работы номер 2

1. **Сжатие и восстановление аудиосигналов методами библиотек Librosa и Pydub.**

Предложенная теоретическая часть включает основы цифрового аудио, описание методов сжатия аудио и их реализация в библиотеках Librosa и Pydub, а также пример реализации сжатия с дальнейшей визуализацией результата.

Практика заключается в использовании собственного аудиофайла на алгоритме сжатия из примера, исследование параметров сжатия таких как изменение битрейта и частоты дискретизации с целью улучшения качества сжатого аудиосигнала и применение другого алгоритма сжатия одной из библиотек. Таким образом, практику в этой работе можно разделить на несколько подпунктов или разделить по вариантам, т.к. подбор нужного алгоритма занимает длительное время и, вероятно, не будет возможен студентами самостоятельно в рамках практикума.

Весьма интересным при выполнении этой работы станет отчетность потому, что студенты смогут посоревноваться в улучшении качества сжатия.

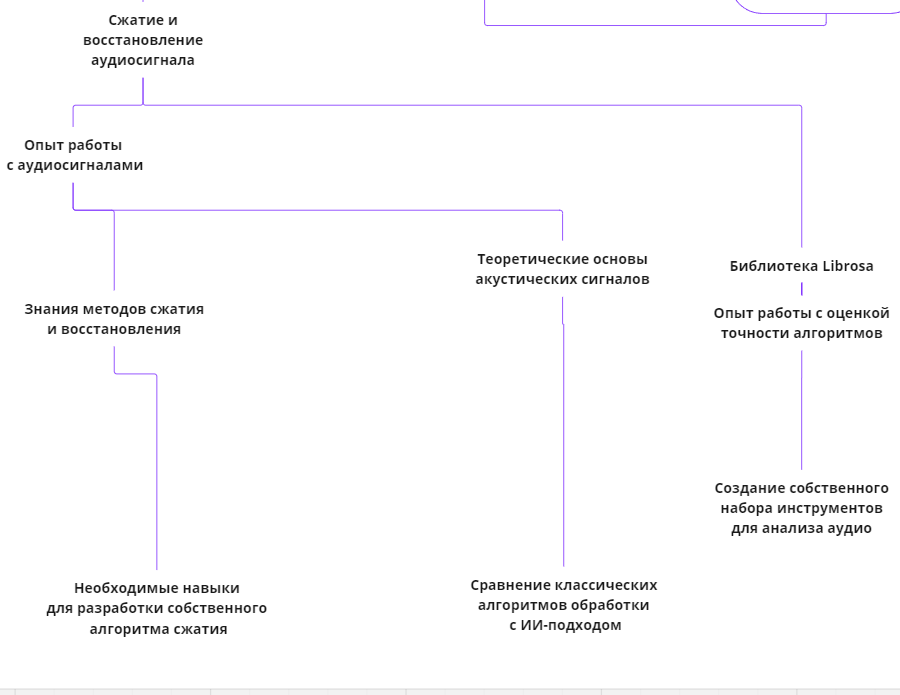
Рисунок 3.3 также содержит блок-схему полученных навыков после выполнения работы.

Рисунок 3.3 – Блок-схема навыков лабораторной работы номер 3

Можно заметить, что работа, посвященная обработке звука, является самой простой для выполнения, однако полученные навыки способствуют дальнейшему изучению в этой области. Замечу также то, что при выполнении данной работы стоит учесть разнообразие форматов и жанров аудиосигналов.

1. **Дообучение модели нейронной сети EfficientNet на базе изображений мягких игрушек.**

Теория содержит необходимую информацию про процесс дообучения, также известного как трансферное обучение. Отличие от работы по классификации будет в более подробном изучении важнейших метрик для оценки качества модели, таких как: точность, полнота, F1-мера и др.; информация о оптимизации модели, часть из которой пойдет в практику. Примером работы подобной системы выступит уже выполненный процесс обучения модели, трансферного обучения и валидации результатов на базе двух дата сетов Animals-10 и SoftToy-1.

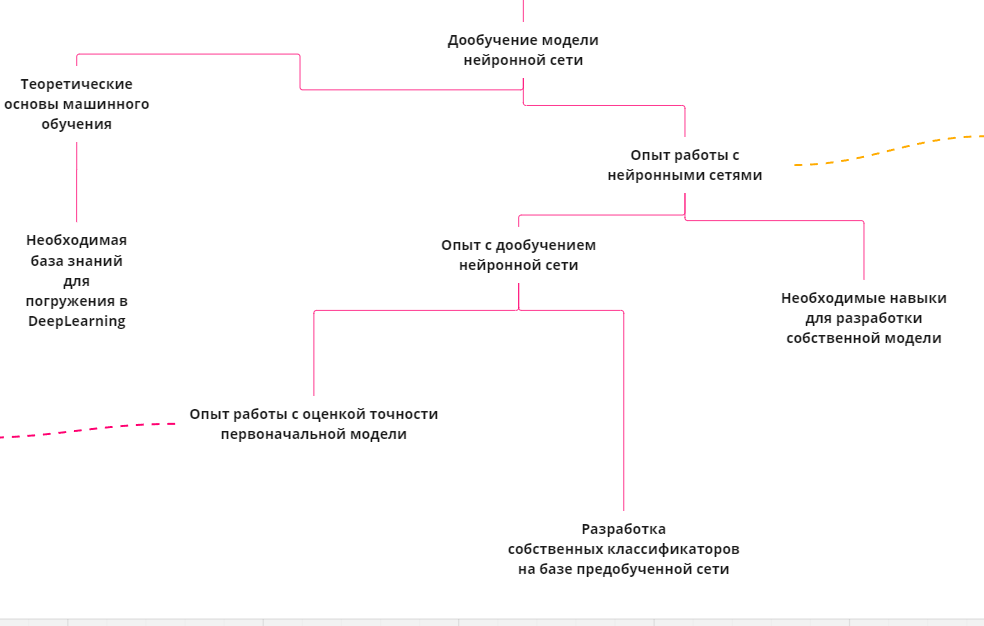
Практическая работа студентов включит в себя подготовку данных, путем добавления новых изображений в набор данных и последующей разбивкой на обучающую, валидационную и тестовую выборки. Изменение параметров выходных слоев станет основной задачей работы, т.к. сам процесс трансферного обучения не обходиться без заморозки весов начальных слоев. Студентам также придется разобраться с оптимизацией модели, что и войдет в отчет по лабораторной работе и позволит оценить уникальность подходов к выполнению практикума. Рисунок 3.4 иллюстрирует что изучат студенты во время выполнения работы.

Рисунок 3.4 – Блок-схема навыков лабораторной работы номер 4

Таким образом, после формулирования задач и тем лабораторных работ можно перейти к созданию методических указаний и определить их необходимое содержание.

1. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛЕНИЮ ЛАБОРАТОРНЫХ РАБОТ

Цель данных указаний – предоставить студентам четкие инструкции и рекомендации по выполнению лабораторных работ, а также помочь им эффективно освоить необходимые навыки и знания, предоставляя возможность изучить подробнее каждую из областей работы с медиа. Опираясь на задания, темы, теоретические сведения из второй главы, методические указания должны включать:

* Теоретическую часть: Краткое изложение теоретических основ, необходимых для выполнения работы. Этот раздел включает в себя ключевые концепции, алгоритмы и методы, которые будут применяться в лабораторных работах.
* Описание задания: Пошаговое описание задачи, которую нужно решить в рамках лабораторной работы в целях освоить определенную область обработки и анализа медиаданных.
* Примеры кода: Примеры, которые помогут студентам понять, как реализовать решение задачи. Весь код реализован на языке программирования Python с использованием сторонних библиотек для определенной задачи.
* Вопросы для самоконтроля: Вопросы, которые помогут студентам проверить свое понимание материала и подготовиться к защите работы. В некоторых работах отчетность предполагает собой составление презентаций с демонстрацией полученных результатов работы алгоритмов, разработанных самими студентами.

Указания также содержат дополнительные рекомендации по использованию внешних ресурсов, таких как различные научные статьи, документация библиотек и прочие источники, которые могут быть полезны для углубленного изучения тематики лабораторных работ и помочь с необходимыми знаниями для выполнения определенных заданий.

Следует упомянуть, что часть указаний реализованы непосредственно в Jupyter Notebooks на платформе Google Colab в ячейках типа Markdown для большей наглядности. Предложенные указания также содержат ссылки на файлы Colab, в которых описывается та же информация, что и в теоретической части, дополненная практическими примерами.

## Введение в Google Colab и Python для обработки медиа данных

Одним из важнейших навыков последних десятилетий стало умение работать с информацией и грамотно работать с ней. Повседневность современного человека в бытовой жизни или на производстве невозможно представить без использования медиа данных – изображений, аудио и видео. Цель данной лабораторной работы – познакомить студентов с основными инструментами и современными методами обработки медиа данных.

Главным инструментом на данный момент является язык программирования Python. Python – это понятный и лаконичный язык с хорошей поддержкой ввода/вывода, цифр, изображений и построения графиков. У этого языка есть некоторые особенности, такие как отступы и компактный синтаксис, к которым нужно привыкнуть [21]. Выполнение дальнейших работ предполагает знание базовых основ языка, которые можно изучить в источнике [16]. Написанные для него библиотеки позволяют решать многие задачи, использовав готовые алгоритмы. Библиотеки необходимо импортировать с помощью ключевого слова import.

Использование методов этих библиотек предполагает обращения к ним через точку с дальнейшим указанием метода, например: plt.show(). Основные методы библиотек визуализации, чтения и анализа данных, работы с изображениями и звуком приведены ниже:

Визуализация данных (Matplotlib) [14]:

* plt.plot(): Построение графиков (линейных, точечных, столбчатых и др.).
* plt.imshow(): Отображение изображений.
* plt.hist(): Построение гистограмм.
* plt.scatter(): Построение диаграмм рассеяния.
* plt.bar(): Построение столбчатых диаграмм.
* plt.pie(): Построение круговых диаграмм.
* plt.xlabel(), plt.ylabel(), plt.title(): Добавление подписей к осям и заголовка графика.

Чтение и анализ данных (Pandas) [22]:

* pd.read\_csv(), pd.read\_excel(): Чтение данных из CSV и Excel файлов.
* df.head(), df.tail(): Просмотр первых и последних строк DataFrame.
* df.describe(): Получение описательной статистики данных.
* df.info(): Получение информации о типах данных и пропущенных значениях.
* df.groupby(): Группировка данных по определенному столбцу.
* df.sort\_values(): Сортировка данных по значениям столбца.
* df.fillna(): Заполнение пропущенных значений.
* df.loc[], df.iloc[]: Индексация и выборка данных.

Машинное обучение (TensorFlow) [25]:

* tf.constant(): Создание тензора (многомерного массива) с постоянными значениями.
* tf.Variable(): Создание тензора, значения которого могут изменяться в процессе вычислений.
* tf.data.Dataset: Создание объекта для представления набора данных, который может быть использован для обучения моделей.
* tf.function: Декоратор для преобразования функции Python в граф вычислений TensorFlow, что может ускорить выполнение.
* tf.GradientTape(): Контекстный менеджер для автоматического вычисления градиентов, что необходимо для обучения моделей с помощью градиентного спуска.

Работа со звуком (Librosa) [17]:

* librosa.load(): Загрузка аудиофайла.
* librosa.output.write\_wav(): Сохранение аудиоданных в WAV-файл.
* librosa.stft(): Вычисление кратковременного преобразования Фурье (STFT).
* librosa.amplitude\_to\_db(): Преобразование амплитуды в децибелы.
* librosa.feature.melspectrogram(): Вычисление мел-спектрограммы.
* librosa.display.specshow(): Отображение спектрограммы.
* librosa.display.waveplot(): Построение графика амплитуды аудиосигнала (осциллограммы).
* librosa.display.specshow(): Отображение спектрограммы (визуализация частотного содержания аудиосигнала).

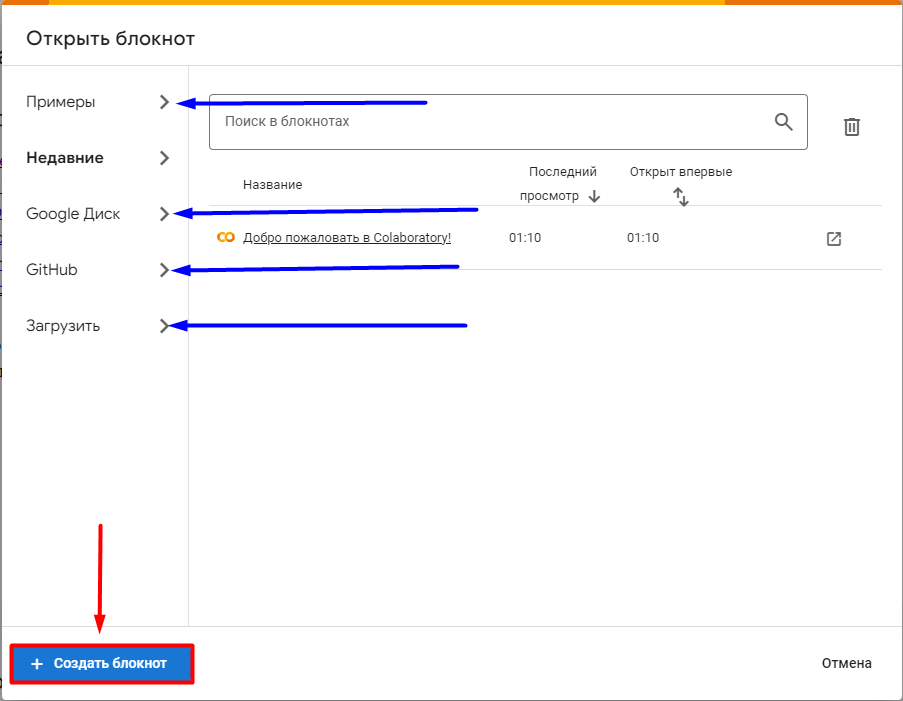
**Создание первого файла в Google Colab**

Для работы с медиа данными также необходима среда разработки. Можно использовать любые среды разработки, сохраняя код в файлы с расширением .py, однако важным моментом является использование ресурсов графического ядра при обучении моделей нейронной сети, поэтому желательно использовать редактор с встроенным аппаратным ускорителем такой как Google Colaboratory.

После перехода по ссылке [10] в качестве дополнительного материала можно ознакомиться с вводным уроком, после чего зарегистрироваться. На рисунке 4.1 показано приветственное окно с вводным уроком.

Стрелка указывает на кнопку “Войти”, перенаправляющую на окно входа в аккаунт Google

Рисунок 4.1 – Приветственное окно редактора Google Colab

После входа в аккаунт Google, происходит перенаправление на вкладку “Открыть блокнот”. На рисунке 4.2 показаны разделы, откуда можно открыть блокнот Jupyter Notebook: Google Диск, GitHub и загрузить с локального хранилища, а также можно ознакомиться с примерами от авторов среды разработки. Необходимо создать первый блокнот нажав на кнопку “Создать блокнот”.

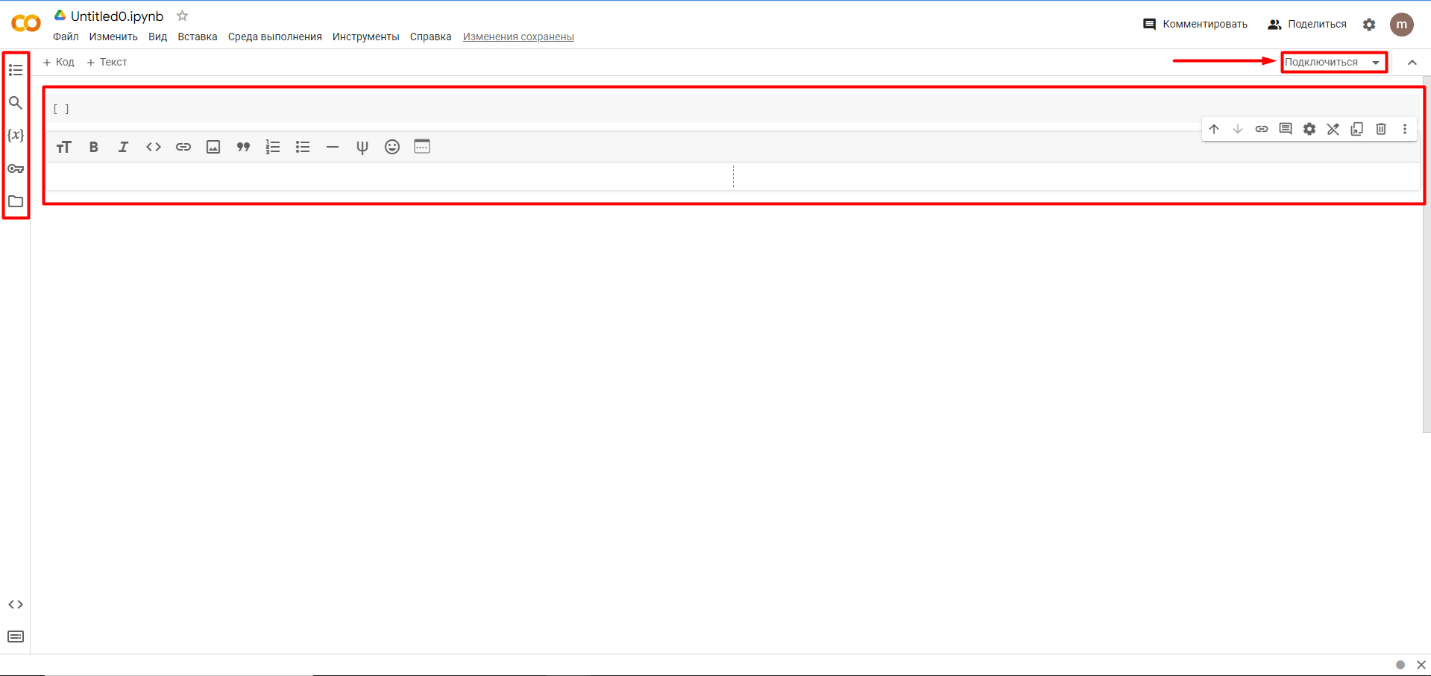
Горизонтальными стрелками обозначены способы открытия блокнота Jupyter, вертикальная стрелка указывает на кнопку “Создать блокнот”

Рисунок 4.2 – Окно “Открыть блокнот”

После создания блокнота открывается чистый файл формата .ipynb. На рисунке 4.3 показаны основные разделы: управление содержанием и файлами, подключение к средам выполнения и рабочее окно.

**Среды выполнения**

Нажав на стрелку расположенную у кнопки “Подключиться”, выпадает окно сред выполнения. Стоит обратить особенное внимание на разделы “Сменить среду выполнения”, содержащую информацию о аппаратных ускорителях и типе среды выполнения, “Показать ресурсы”, позволяющую оценить остаток вычислительных единиц, и “Управление сеансами”, при нажатии которого открывается окно активных сеансов, удобное для мониторинга открытых и запущенных сред выполнения.

В базовой бесплатной подписке предоставляются ресурсы, показанные на рисунке 4.4, такие как, CPU (Центральный процессор), T4 GPU (Графический процессор с тензорными ядрами) и TPU v2 (Тензорный процессор от Google), разницу между которыми можно изучить в источнике [32].

Схематично отображены основные разделы: вертикальный прямоугольник слева – управление содержанием и файлами, прямоугольник по центру – рабочее окно, прямоугольник справа с указывающей стрелкой – подключение к средам выполнения

Рисунок 4.3 – Начальное окно созданного блокнота

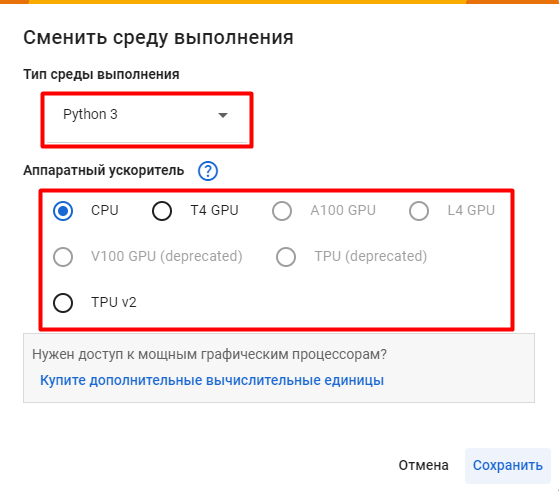


Рисунок 4.4 – Окно смены среды выполнения

**Подключение Google Drive**

Основной особенностью выбранной среды является интеграция с облачным хранилищем Google Drive. Использование хранилище позволяет делать автоматические сохранения, удобно подключать сторонние файлы, например дата сеты или модель нейронной сети, а также предоставляет доступ к данным из любого места.

Подключение Drive в свой блокнот производится выполнением команду в ячейке кода:

from google.colab import drive

drive.mount('/content/drive')

После удачной авторизации возможно обращение к файлам в Google Drive через путь /content/drive/MyDrive/.

**Практический пример**

Перед началом работы с основными библиотеками в рамках практической части, предлагается изучить пример использования основных методов библиотек обработки медиа данных:

Первым делом необходимо установить и импортировать библиотеки, выполнив команду:

!pip install opencv-python # Установка OpenCV

# Импорт библиотек

import cv2 # OpenCV

import matplotlib.pyplot as plt #Основной модуль Matplotlib

import pandas as pd # Pandas

Выполнив следующий код создается датафрейм содержащий информацию о распределении продаж по годам:

data = {'Год': [2020, 2021, 2022, 2023], 'Продажи': [1500, 1800, 2200, 2500]} # создается словарь

df = pd.DataFrame(data) # преобразование в датафрейм

После успешного создания датафрейма производится построение графика распределения:

plt.figure(figsize=(8, 6)) # задание размера графика

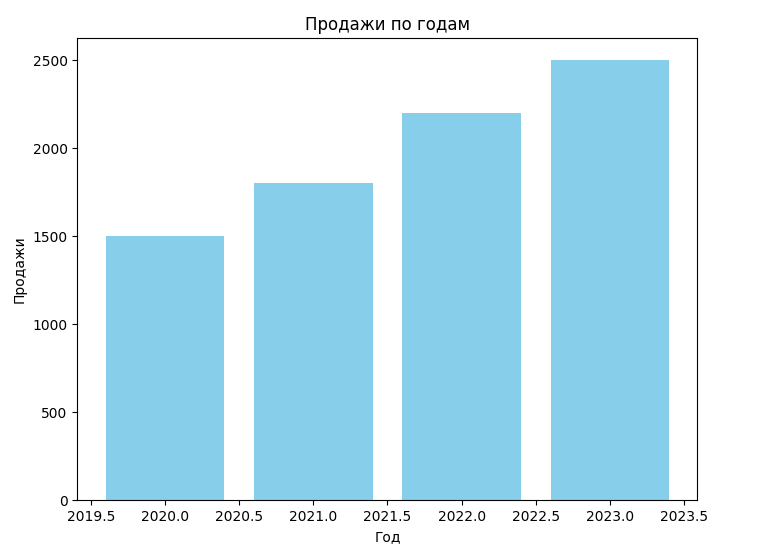
plt.bar(df['Год'], df['Продажи'], color='skyblue') # построение графика

plt.title('Продажи по годам') # задание названия графика

plt.xlabel('Год') # подпись оси Х

plt.ylabel('Продажи') # подпись оси Y

plt.show() # отображение графика

Рисунок 4.5 – Результаты работы программного кода библиотеки Pandas

Разобравшись с основами matplotlib и Pandas, можно приступить к обработке изображений, вызвав методы OpenCV, предварительно загрузив пробное изображение на Google Drive.

# Путь к изображению в Google Drive (или локальный путь)

image\_path = '/content/drive/MyDrive/wallpapers-autumn-019.jpg'

image\_nonrgb = cv2.imread(image\_path) # метод чтения файлов OpenCV для изображений

# Конвертация в RGB

image\_rgb = cv2.cvtColor(image\_nonrgb, cv2.COLOR\_BGR2RGB)

plt.imshow(image\_rgb)

plt.title('Изображение')

plt.show()

Рисунок 4.6 – Результаты работы программного кода библиотеки OpenCV

Теперь проводится преобразование изображения, рассматривая простые операции изменения размера, поворота и отражения по вертикали, а также изменения яркости/контраста и обрезка:

# Изменение размера до 300х200 пикселей

resized\_image = cv2.resize(image, (300, 200))

# Поворот на 90 градусов по часовой стрелке

rotated\_image = cv2.rotate(image, cv2.ROTATE\_90\_CLOCKWISE)

# 0 - отражение по оси X (вертикали)

flipped\_image = cv2.flip(image, 0)

# Обрезка изображения

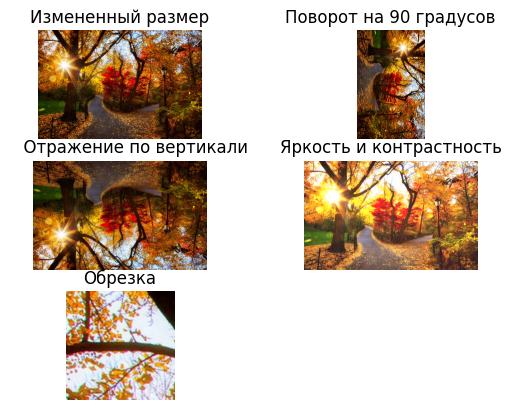
cropped\_image = image[50:250, 100:300]

# Изменение яркости и контраста

alpha = 1.5 # Умножение для яркости (1.0 - без изменений)

beta = 30 # Смещение для контрастности (0 - без изменений)

adjusted\_image = cv2.convertScaleAbs(image, alpha=alpha, beta=beta)

Рисунок 4.7 – Результаты обработки пробного изображения

**Задания**

1. Необходимо загрузить 5 различных изображений на собственный Google Drive и отобразить их в виде сетки при помощи метода plt.subplot()
2. Изменить размер загруженных изображений до размера 450х500 пикселей и обрезать понравившуюся область, создав при этом два дополнительных изображения: обрезанной области и изображения без нее.
3. Применить к исходному изображению методы размытия по Гауссу cv2.GaussianBlur(), меняя при этом радиус размытия, а также обнаружить края изображения методом cv2.Canny(), меняя пороги.
4. Собрать 7 изображений различных сцен (пейзажи, натюрморты и т.д.), перевести изображения в цветовую модель HSV [11] и для каждого изображения вычислить гистограмму методом cv2.calcHist() каналов H, S и V, сохранив значения в DataFrame Pandas, после чего найти среднее, медиану, стандартное отклонение методами Pandas и сравнить распределения цветов для разных типов изображений.

## Классификация изображений на базе предобученной модели библиотеки Keras

Классификация изображений – это важнейшая задача в области компьютерного зрения, направленная на автоматическое распределение изображений по заранее определенным категориям. Этот процесс имеет огромное значение для множества приложений, таких как распознавание объектов, анализ медицинских изображений, автономное управление транспортом и системы безопасности. Классификаторы изображений обучаются на большом объеме размеченных данных, что позволяет моделям научиться распознавать характерные черты объектов и отличать их друг от друга [24]. Реализация подобных классификаторов невозможна без использования предобученных нейронных сетей.

Предобученные модели являются важным инструментом в этой области. Они представляют собой нейронные сети, которые были обучены на огромных наборах данных, таких как ImageNet, содержащем миллионы изображений. В процессе обучения такие модели приобретают способность извлекать общие визуальные признаки, такие как края, текстуры и формы, что делает их крайне полезными для различных задач компьютерного зрения. Использование предобученных моделей значительно ускоряет разработку новых приложений, так как не требует начинать обучение с нуля. Вместо этого модели можно адаптировать к конкретной задаче, дообучив их на меньшем, специфичном для этой задачи, наборе данных [2].

Таким образом, предобученные модели не только сокращают время разработки, но и повышают точность и эффективность классификации изображений. Важным аспектом в понимании задачи классификации является владение представлением о сверточных нейронных сетях, основная информация о которых была изложена в первой главе, однако для дополнительной информации можно обратиться к источнику [8].

**Библиотека Keras как инструмент реализации классификатора**

Keras – это высокоуровневая библиотека и мощный инструмент, упрощающий создание и обучение нейронных сетей. Его понятный интерфейс позволяет сосредоточиться на архитектуре модели и экспериментах, абстрагируясь от низкоуровневых деталей [15].

Существует два основных подхода для определения модели: последовательная модель (Sequential), являющаяся наиболее простым способом построения модели, где слои следуют друг за другом и идеально подходит для задач, где данные проходят через сеть линейно, и функциональный API, который позволяет создавать более сложные модели с разветвлениями и объединениями слоев [15].

После определения модели, необходимо добавить слои. В Keras предлагаются различные виды слоев, такие как сверточные слои (Conv2D), извлекающие признаки из изображений с помощью применения фильтров, слои пулинга (MaxPooling2D), которые уменьшают размерность данных, сохраняя важные признаки, и полносвязные слои (Dense), выполняющие классификацию на основе извлеченных признаков [15]. Также Keras предлагает множество других слоев для различных задач, таких как нормализация, активация и регуляризация.

После добавления слоев, модель необходимо скомпилировать с помощью функции model.compile(), которая включает выбор оптимизатора, корректирующего веса модели для уменьшения ошибки (например, Adam или SGD), функции потерь, оценивающей, насколько хорошо модель предсказывает результаты (например, categorical\_crossentropy для классификации), и метрик, показывающих качество модели (например, accuracy для оценки точности) [15]. Затем можно приступить к обучению модели с использованием функции model.fit(), где подготовленные данные для обучения (изображения и соответствующие метки классов) проходят через сеть в течение нескольких эпох – количество проходов по всему набору данных, а также определяется размер пакета (batch size) – количество образцов, обрабатываемых за один шаг обучения [15].

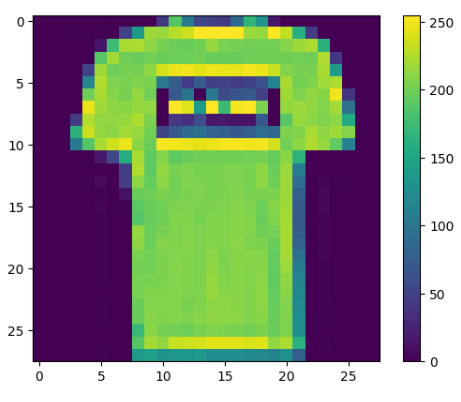
После завершения обучения, модель оценивается с использованием функции model.evaluate() на тестовых данных, которые не использовались для обучения, чтобы оценить реальную производительность модели. Наконец, для получения предсказаний классов на новых данных используется функция model.predict(), применяемая к изображениям, для которых нужно получить предсказания.

**Пример реализации классификатора на базе предобученной модели с использованием набора данных Fashion-MNIST**

Все дальнейшие действия будут сопровождаться текстовым описанием и иллюстрациями. Код работы находится в приложении А.

В первую очередь необходимо импортировать библиотеки, классы для создания слоев и дата сет. Важным этапом является разделение на тренировочную и тестовую выборки, на основе одних которых будет проводится обучение классификатора, а других – валидация и тестирование эффективности обучения. Затем задается список названий классов, классификация по которым и будет происходить.

Этап предподоготовки к обучению также включает визуализацию начальных данных, проведенную в целях проверки работы нормализации, изображенную на рисунке 4.8 и 4.9.

Рисунок 4.8 – Цветовая шкала первого изображения из тренировочного набора

Стоит учесть, что изображения в наборе данных черно-белые, так что шкала описывает степень “затененности” пикселей.

Рисунок 4.9 – Первые 25 изображений из тренировочного набора

Подобная визуализация крайне полезна для проверки качества данных до и после нормализации данных, проделанной в следующем шаге.

После преобразования данных описана архитектура последовательной модели с тремя слоями: выравнивания, плотный и выходной. Также в программном коде указаны функции активации, информацию о которых можно изучить в источнике [13]. Далее идет компиляция модели с указанием оптимизатора, функцию потерь и метрику точности, информацию о которых можно подчерпнуть в источнике [20].

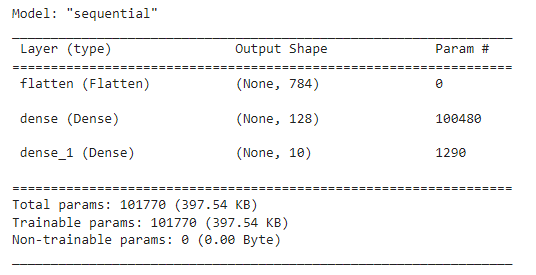
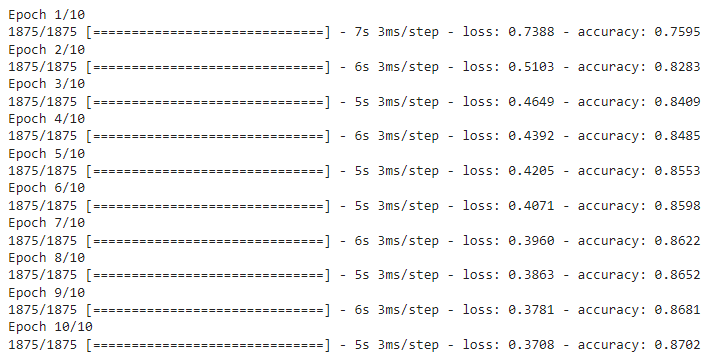
 На рисунке 4.10 описано строение такой модели: количество параметров, размер слоев и другие детали.

Рисунок 4.10 – Строение последовательной модели

Процесс тренировки модели происходит в течении 10 эпох, иллюстрированных на рисунке 4.11, после чего происходит процесс оценки модели на тестовых данных, валидация результатов и возврат потерь. Как можно заметить, каждая из эпох содержала 1875 различных изображений, на которых обучалась модель. Также рисунок 4.11 содержит информацию о значении функции потерь и точности на каждой из эпох, затраченное время на выполнение шага и целой эпохи.

Последним этапом выполняется предсказание для тренировочной выборки, возвращая вероятности для каждого из классов и сравнивая индекс класса с наибольшей вероятностью с реальной меткой класса определенного изображения.

 Рисунок 4.11 – Результаты работы метода model.fit()

**Задания**

1. Запустите программный код, указанный в примере. Постройте кривые функции потерь и точность. Подберите количество нейронов в плотном и выходном слое для достижения повышения точности.
2. Измените функции активации в слоях модели, используйте другой оптимизатор и функцию потерь. Подбирая различные комбинации, добейтесь повышения точности классификации с минимальным увеличением количества эпох.
3. Загрузите 10 изображений одежды, в соответствии с определенными в задании классами на Google Drive. Использовав нормализацию, приведите изображения к разрешению 280х280 пикселей. Попробуйте классифицировать эти изображения и дайте объяснения результатам.
4. Загрузите и подготовьте любой дата сет. Выберите предобученную модель из списка (VGG16, ResNet50, EfficientNet) и дообучите ее использую классы и методы из примера. Оцените точность классификации на тестовой выборке.

## Сжатие и восстановление аудиосигналов методами библиотек Librosa и Pydub

Для работы с аудиосигналами необходимо иметь представление о создании таких сигналов, т.е. создание цифрового аудио.

Из источника [4] следует, что цифровое аудио представляет собой преобразование звуковых волн в цифровую форму для обработки, хранения и передачи с использованием цифровых устройств. Этот процесс включает несколько ключевых этапов:

**Дискретизация (Sampling):**

Звуковой сигнал непрерывно измеряется через равные интервалы времени для создания дискретного набора данных. Частота дискретизации, или количество выборок в секунду, измеряется в герцах (Гц). Стандартная частота дискретизации для CD-аудио составляет 44,1 кГц, что означает 44,100 выборок в секунду. Более высокая частота дискретизации позволяет более точно захватить звуковой сигнал, особенно высокочастотные компоненты.

**Квантование (Quantization):**

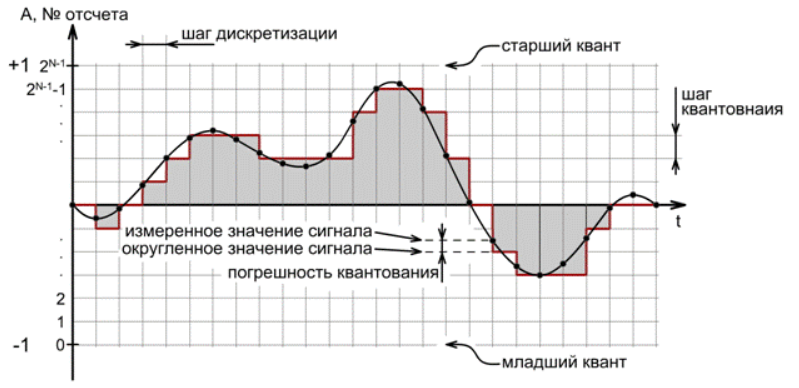
После дискретизации каждая выборка звукового сигнала должна быть представлена числовым значением, которое называется квантованием. Битовая глубина (bit depth) определяет количество битов, используемых для представления каждой выборки. Больше битов позволяет более точно представить амплитуду сигнала, улучшая качество звука. Например, битовая глубина 16 бит используется в CD-аудио, предоставляя 65,536 возможных значений амплитуды. На рисунке 4.12 можно изучить схематичное представление процесса линейного квантования, а в источнике [31] прочитать подробнее.

Рисунок 4.12 – Представление линейного(однородного) квантования [31]

**Кодирование аудио (Encoding):**

Кодирование аудио включает преобразование квантованных выборок звукового сигнала в цифровой формат, пригодный для хранения и передачи. Один из основных методов кодирования — это импульсно-кодовая модуляция (Pulse Code Modulation, PCM), которая используется в большинстве несжатых аудиоформатов, таких как WAV и AIFF. PCM обеспечивает линейное представление амплитуды звукового сигнала, где каждая выборка представлена числовым значением. Это позволяет сохранять высокую точность исходного сигнала. Другие методы, такие как адаптивная дифференциальная импульсно-кодовая модуляция (Adaptive Differential PCM, ADPCM), используют алгоритмы сжатия без потерь, уменьшая размер файла без потери качества звука. Более сложные алгоритмы, такие как MP3 и AAC, применяют сжатие с потерями, что позволяет значительно уменьшить размер файла за счет удаления данных, которые воспринимаются как менее важные для человеческого слуха​.

**Форматы аудио (Audio formats)**

Из источника [6] следует, что форматы аудио можно разделить на две основные категории: несжатые и сжатые. Несжатые форматы, такие как WAV (Waveform Audio File Format) и AIFF (Audio Interchange File Format), сохраняют все детали звукового сигнала, что делает их идеальными для профессионального аудиопроизводства и архивирования. WAV широко используется в системах Windows, тогда как AIFF — на платформах Apple. Сжатые форматы делятся на два типа: сжатие без потерь и сжатие с потерями. Форматы без потерь, такие как FLAC (Free Lossless Audio Codec) и ALAC (Apple Lossless Audio Codec), уменьшают размер файла, сохраняя при этом качество звука, что делает их популярными среди аудиофилов. Форматы с потерями, такие как MP3 и AAC, значительно уменьшают размер файла за счет удаления части данных, что снижает качество звука, но делает эти форматы идеальными для потоковой передачи и хранения большого количества аудиофайлов.

**Методы сжатия аудио в библиотеке Librosa и Pydub**

Рассматриваются основные возможности для сжатия сигнала, реализованные в библиотеках Librosa и Pydub:

1. Изменение частоты дискретизации:

Изменение частоты дискретизации аудиосигнала позволяет уменьшить размер файла, снижая частоту сэмплирования. Это полезно для компромисса между качеством и размером файла.

1. Прореживание данных (Downsampling):

Прореживание данных позволяет снизить количество выборок путем удаления части данных. Этот метод можно использовать в сочетании с изменением частоты дискретизации для дополнительного уменьшения размера файла.

1. Сжатие динамического диапазона (Dynamic Range Compression):

Сжатие динамического диапазона уменьшает разницу между самыми громкими и самыми тихими частями аудиосигнала, что может уменьшить размер файла при сохранении в форматах с потерями.

1. Обработка тишины (Silence Removal):

Удаление участков тишины или почти тишины из аудиофайла может существенно уменьшить его размер.

1. Преобразование в моно (Convert to Mono):

Преобразование стереофонического звука в монофонический может уменьшить размер файла, так как сохраняется только один канал звука вместо двух.

Подробнее про процесс сжатия аудио (audio compression) можно узнать в источнике [5]. Результаты работы сжатия методом изменения частоты дискретизации представлены на рисунке 4.13, предварительно загрузив тестовый аудиофайл на Google Drive. Спектрограмма сигнала построена в аудиоредакторе Audacity, c применением функции окно Хемминга размером 2048, подпись осей дана в линейном масштабе.

Благодаря сжатия удалось получить 26-ти кратный выигрыш в размере файла: размер исходного файла составил 26.2 Мб, сжатого – 1.1Мб, при этом сохранив качество звучания и изменив частоту дискретизации до 22.05 кГц. После оценки результатов работы алгоритма сжатия на спектрограмме, проводится процесс восстановления исходного аудиосигнала, который особенно эффективен при сравнении работ разных алгоритмов сжатия. Восстановленный файл наглядно показывает различия между исходным и сжатым сигналом, помогая визуализировать влияние алгоритмов на качество звука.

Полный программный код примера, включая реализацию работы различных алгоритмов сжатия, практический пример сжатия и восстановления сигнала, а также расчет размеров итоговых файлов, находится в приложении Б.

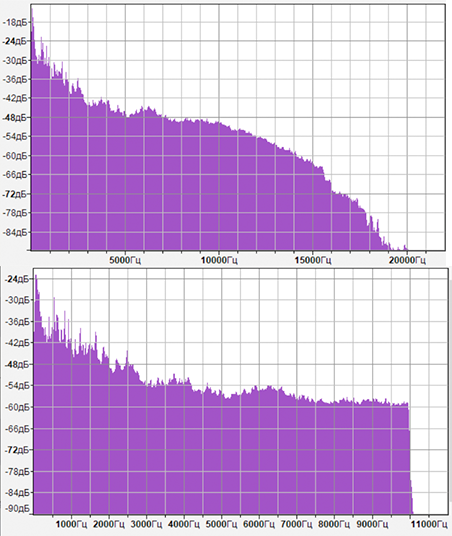
Верхний рисунок – спектр оригинального аудиофайла формата .wav, нижний – спектр сжатого аудиофайла формата .mp3

Рисунок 4.13 – Результаты работы алгоритма сжатия библиотеки Pydub

**Задания**

1. Используя код примера, проверьте работоспособность алгоритмов сжатия и восстановления, применяя собственный аудиофайл.
2. Постройте спектрограммы исходного, сжатого и восстановленного аудиофайлов в Audacity. Объясните результаты работы алгоритмов.
3. Исследуйте влияние частоты дискретизации, путем сжатия с разными частотами дискретизации (44.1 кГц, 22.05 кГц, 16 кГц). Сравните качество звука и размер файлов и подберите оптимальную частоту для собственного файла.
4. Используйте методы библиотеки Librosa для извлечения признаков из музыкальных произведений (темп, ритм, тональность). Постройте хроматическую спектрограмму методом librosa.display.specshow(chroma, y\_axis='chroma', x\_axis='time'), STFT методом librosa.stft() и MFCC методом librosa.feature.mfcc(). Определите тональность методом librosa.pitch\_to\_note(). Все необходимые методы и их параметры можно изучить в источнике [17].

## Дообучение модели нейронной сети EfficientNet на базе изображений мягких игрушек

**Трансферное обучение**

Трансферное обучение (TL) – это метод машинного обучения (МО), при котором модель, предварительно обученная выполнению одной задачи, настраивается для выполнения новой, связанной с предыдущей. Обучение новой модели МО – это трудоемкий и длительный процесс, требующий большого количества данных, достаточной вычислительной мощности и прохождения нескольких итераций, прежде чем модель будет готова к запуску. Вместо этого организации используют метод TL для переобучения существующих моделей, подготавливая их к решению смежных задач с использованием новых данных [34].

**Основные стратегии трансферного обучения**

Трансдуктивное трансферное обучение предполагает перенос знаний из определенной исходной функциональной области в другую, но связанную целевую область, при этом основное внимание уделяется целевой области. Это особенно полезно, когда маркированных данных из целевой функциональной области мало или совсем нет [34]. Такая стратегия предлагает модели делать прогнозы на основе целевых данных, используя ранее полученные знания. Поскольку целевые данные математически аналогичны исходным, модель находит закономерности и работает быстрее.

Индуктивное трансферное обучение – это метод, в котором исходная и целевая области совпадают, но задачи, которые должна решать модель, различаются. Предварительно обученная модель уже знакома с исходными данными и быстрее обучается новым функциям.

Трансферное обучение без наблюдения использует стратегию, аналогичную индуктивному трансферному обучению, для развития новых способностей. Однако эту форму трансферного обучения стоит использовать, когда у вас есть только немаркированные данные как в исходной области, так и в целевой. Модель изучает общие черты немаркированных данных, чтобы их можно было более точно обобщить, когда ее попросят выполнить целевую задачу. Этот метод полезен, если получение маркированных исходных данных является сложным или дорогим процессом.

**Этапы трансферного обучения**

Основная идея трансферного обучения на практике заключается в корректировке количества слоев, путем заморозки базовых слоев и добавления новых слоев, таким образом верхние слои базовой модели, предварительно обученной, например, на изображениях автомобилей, остаются нетронутыми, в то время как на нижних слоях необходимо недопустить изменения их весов путем заморозки, после чего добавляются новые слои для классификации, например, марок автомобилей.

Итоговый алгоритм дообучения выглядит следующим образом:

1. Сбор и подготовка данных с использованием аугментации для дальнейшего увеличения объема обучающего набора.
2. Загрузка предобученной модели с последующей заморозкой базовых слоев.
3. Добавление новых слоев для дальнейшей классификации на новых данных
4. Компиляция модели, выбор оптимизатора и функции потерь.
5. Обучение модели, использую часть тренировочной выборки для валидации результатов.
6. Оценка и тестирование модели, построение графиков метрик
7. “Тонкая настройка” (Fine-Tuning) модели: разморозка слоев, компиляция с уменьешенной скоростью обучения, дообучение модели.

Практическая часть к этой лабораторной работе содержит задания, в рамках которых требуется реализовать каждый из представленных этапов.

**Метрики классификации**

Оценка качества модели машинного обучения — это ключевой этап, позволяющий понять, насколько эффективно модель решает поставленную задачу. В контексте классификации изображений наиболее часто используются приведенные ниже метрики, подробнее описанные в источнике [28]:

Accuracy (точность): Процент правильных предсказаний среди всех сделанных предсказаний. Эта метрика особенно полезна, когда классы в наборе данных сбалансированы.

Precision (точность предсказания положительного класса): Процент истинных положительных предсказаний среди всех предсказанных положительных результатов. Важна в ситуациях, где критично минимизировать количество ложноположительных результатов.

Recall (полнота): Процент истинных положительных результатов среди всех фактических положительных случаев. Эта метрика особенно важна, когда необходимо минимизировать количество ложноотрицательных срабатываний.

F1-мера: Гармоническое среднее между точностью (precision) и полнотой (recall). Полезна, когда необходимо найти баланс между этими двумя метриками.

Confusion Matrix (матрица ошибок): Таблица, показывающая количество истинных положительных, истинных отрицательных, ложноположительных и ложноотрицательных предсказаний. Позволяет детально анализировать, где именно модель ошибается.

Эти метрики дают всестороннее понимание производительности модели, позволяя выявить её сильные и слабые стороны и принять обоснованные решения о дальнейшей настройке и улучшении.

**Оптимизация модели**

Оптимизация модели — это процесс улучшения ее производительности путем корректировки гиперпараметров и архитектуры. В Keras можно использовать различные подходы для оптимизации, каждый из которых имеет свои особенности и применение. Основные оптимизаторы, использующиеся при работе с предобученными моделями Keras:

Adam: Этот оптимизатор адаптирует скорость обучения для каждого параметра. Он хорошо работает на большинстве задач и часто является хорошей отправной точкой.

SGD (Stochastic Gradient Descent): Классический оптимизатор, который обновляет параметры на каждом шаге, основанном на случайном подмножестве данных. Полезен для задач с большим объемом данных и позволяет контролировать шаг обучения.

RMSprop: Оптимизатор, который адаптирует скорость обучения для каждого параметра, предотвращая слишком большие шаги и улучшая сходимость на задачах с шумными градиентами.

Подробнее задачи оптимизации и их решения описаны в источнике [29].

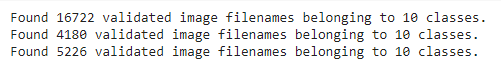
**Пример реализации трансферного обучения**

Примером реализации классификатора с последующим трансферным обучением послужит обучение предобученной модели EfficientNetB7 сперва на изображениях животных из набора данных Animals-10, а затем на изображениях мягких игрушей животных набора данных SoftToy-1.

Необходимый программный код примера реализован в приложении В.

Как и в предыдущей задаче классификации, в первую очередь импортируются библиотеки для работы с данными: анализ и обработка данных, визуализация, методы машинного обучения и библиотека для построения метрик.

Вторым шагом происходит загрузка данных, “сбор” изображений по всем директроиям набора данных и последующее помещение файлов в формат Pandas DataFrame.

Cледующий шаг разбивает изображения на три выборки, результаты которого показаны на рисунке 4.14.

16722 изображения добавлены в тренировочную выборку, 4180 – в валидационную и 5226 – в тестовую

Рисунок 4.14 – Результаты разделения изображений на три выборки

Реализация предобученной модели заключается в ее загрузке и определении входных параметров, таких как входная форма, выбор степень пулинга и определение весов. Основным шагом является проектирование дополнительных слоев модели, реализованное следующим программным кодом из приложения В:

# полносвязный слой

x = Dense(128, activation='relu')(pretrained\_model.output)

# слой нормализации, для ускорения процесса обучения

x = BatchNormalization()(x)

# слой случайным образом "выключает" (устанавливает в ноль) 45% нейронов на каждой итерации обучения.

x = Dropout(0.45)(x)

x = Dense(256, activation='relu')(x)

x = BatchNormalization()(x)

x = Dropout(0.45)(x)

Подстройка количества нейронов, изменение функции активации и сам порядок слоев являются важными параметрами, определяющими точность модели.

На этапе обучения модели было выбрано количество эпох равное 15 с целью уменьшения времени работы модели, т.к. каждая из эпох проходится по 523-ем уникальным изображениям и тратя при этом порядка секунды на каждом шаге. Процесс обучения изображен на рисунке 4.15.

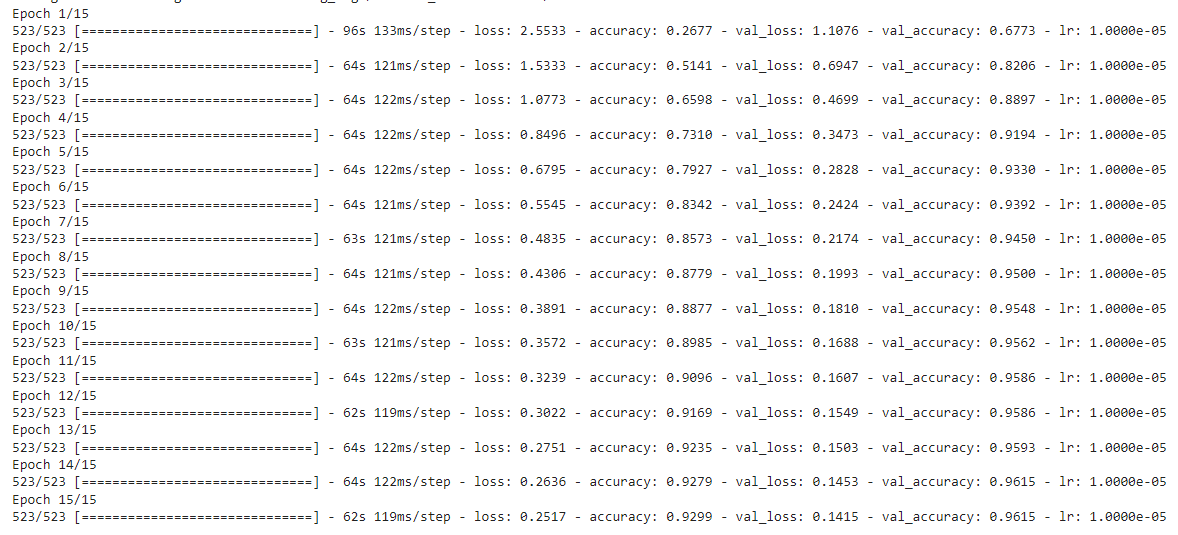
Основные характеристики обучения слева направо: loss – значение функции потерь, accuracy – точность на обучающих данных, val\_loss - значение функции потерь на валидационных данных, важнейший параметр, определяющий эффективность модели val\_accuracy – точность на валидационных данных и lr(learning rate) – скорость обучения или параметр коррекции весов.

Рисунок 4.15 – Процесс обучения модели на изображениях животных

После обучения, проводится расчет точности модели и построение графиков метрик:

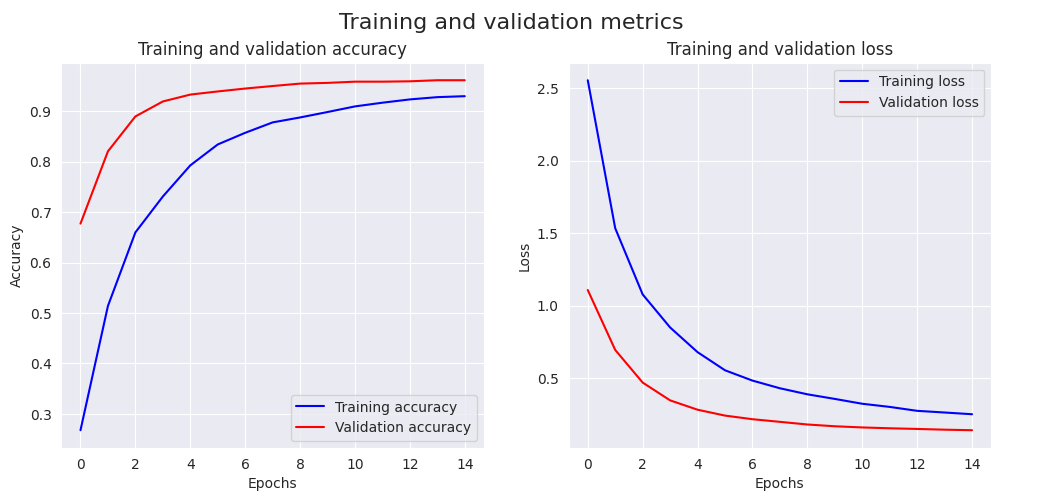
results = model.evaluate(test\_images, verbose=0)

print(" Test Loss: {:.5f}".format(results[0]))

print("Test Accuracy: {:.2f}%".format(results[1] \* 100))

С полученными результатами: Test Loss: 0.12359 и Test Accuracy: 96.48%. Таким образом обученная модель с вероятностью в 96.48 процента сможет классифицировать животное на наборе данных Animals-10.

На рисунке 4.16 видны графики значений точности и потерь на каждой из эпох.

Рисунок 4.16 – Графики тренировочных и валидационных метрик

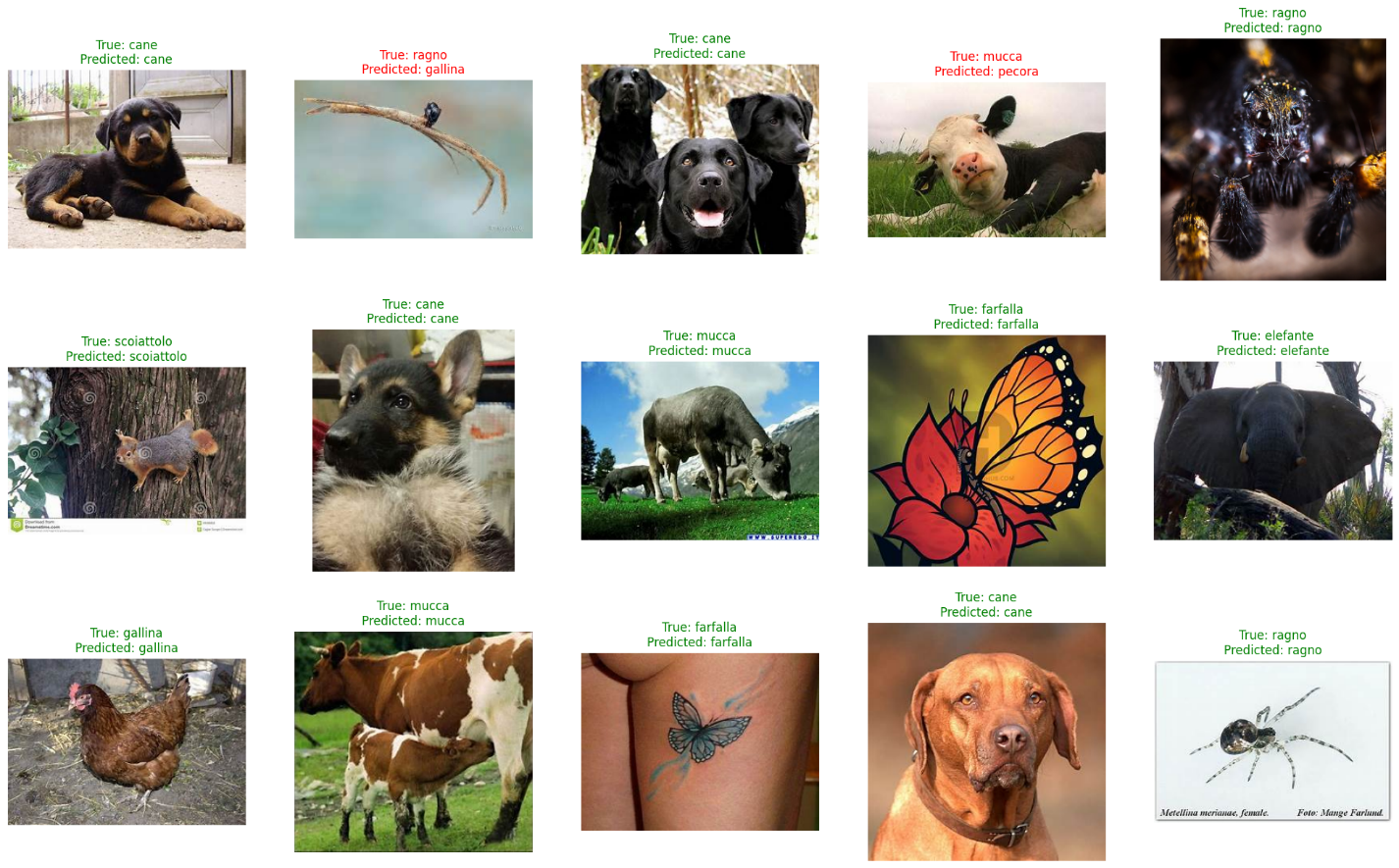
За этапом оценки метрик следует предсказание и визуализация результатов классификации на 25-ти случайных изображениях из тестовой выборки. Результат работы классификатора изображен на рисунке 4.17.

Рисунок 4.17 – Демонстрация классификации изображений

**Обучение модели на изображениях мягких игрушек животных**

Повторив шаги от загрузки данных до разделения на выборки, проектируется новая модель, на основе предыдущей, при этом заморозив базовые слои:

# Слои новой модели

x = Dense(256, activation='relu')(model.output)

x = BatchNormalization()(x)

x = Dropout(0.45)(x)

Далее, задавая количество эпох равное 100, проводится обучение модели, проходя по 56-ти уникальным изображениям каждую эпоху. На рисунке 4.18. представлены результаты классификации.

Рисунок 4.18 – Демонстрация классификации мягких игрушек

**Задания:**

1. Запустите классифкатор животных и загрузите изображения мягкихигрушек. Изучите результаты работы без создания дополнительных слоев.
2. Загрузите 20 изображений мягких игрушек соответствующих классов. Скорректируйте параметры слоев и самой модели, такие как: количество нейронов, функция потерь, оптимизатор и функция активации. Постройте графики метрик для классификатора мягких игрушек по аналогу с примером.
3. Постройте матрицу ошибок, используя метод confusion\_matrix(). Определите классы, в которых модель совершает наибольшее количество ошибок.
4. Добавьте еще один класс в набор данных, разделив один из десяти, например, по цветовому признаку. Скорректируйте модель и изучите результаты.

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. AI Has Already Created as Many Images as Photographers Have Taken in 150 Years. Statistics for 2023 [Электронный ресурс]. -   Режим доступа: https://journal.everypixel.com/ai-image-statistics/. – Дата доступа: 03.04.2024.
2. A Comprehensive Study of Vision Transformers in Image Classification Tasks [Электронный ресурс]. – Режим доступа: https://arxiv.org/abs/2312.01232/. – Дата доступа 25.05.2024.
3. Animals-10 [Электронный ресурс]. – Режим доступа: https://www.kaggle.com/datasets/alessiocorrado99/animals10/data/. – Дата доступа 25.04.2024.
4. An introduction to digital audio [Электронный ресурс]. – Режим доступа: https://www.researchgate.net/publication/3584571\_An\_introduction\_to\_digital\_audio/. – Дата доступа 26.05.2024.
5. Audio Compression Explained [Электронный ресурс]. – Режим доступа: https://www.uaudio.com/blog/audio-compression-basics/. – Дата доступа 26.05.2024.
6. Audio File Formats Explained [Электронный ресурс]. – Режим доступа: https://www.masteringthemix.com/blogs/learn/audio-file-formats-explained/. – Дата доступа 26.05.2024.
7. Audio Signal Processing and Coding [Электронный ресурс]. – Режим доступа: https://www.researchgate.net/publication/243527836\_Audio\_Signal\_Processing\_and\_Coding/. – Дата доступа 16.05.2024.
8. Convolutional neural network [Электронный ресурс]. – Режим доступа: https://python-school.ru/wiki/convolutional-neural-network/. – Дата доступа: 05.04.2024.
9. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks [Электронный ресурс]. – Режим доступа: https://arxiv.org/abs/1905.11946/. – Дата доступа 12.04.2024.
10. Google Colab [Электронный ресурс]. – Режим доступа: https://colab.research.google.com/. – Дата доступа 24.05.2024.
11. How to convert an RGB image to HSV image using OpenCV Python? [Электронный ресурс]. – Режим доступа: https://www.tutorialspoint.com/how-to-convert-an-rgb-image-to-hsv-image-using-opencv-python/. – Дата доступа 24.05.2024.
12. ImageNet classification with deep convolutional neural networks. Advances in neural information processing systems [Электронный ресурс]. – Режим доступа: https://papers.nips.cc/paper\_files/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html/. – Дата доступа 10.05.2024.
13. Introduction to Activation Functions in Neural Networks [Электронный ресурс]. – Режим доступа: https://www.datacamp.com/tutorial/introduction-to-activation-functions-in-neural-networks/. – Дата доступа 25.05.2024.
14. Introduction to Matplotlib [Электронный ресурс]. – Режим доступа: https://www.geeksforgeeks.org/python-introduction-matplotlib/.  – Дата доступа: 20.05.2024.
15. Keras: API высокого уровня для TensorFlow [Электронный ресурс]. – Режим доступа: https://www.tensorflow.org/guide/keras?hl=ru/. – Дата доступа: 20.05.2024.
16. Learning Python, 5th Edition by Mark Lutz [Электронный ресурс]. – Режим доступа: https://www.oreilly.com/library/view/learning-python-5th/9781449355722/. – Дата доступа 23.05.2024.
17. Librosa 0.10.2 documentation [Электронный ресурс]. – Режим доступа: https://librosa.org/doc/latest/index.html/. Дата доступа: 21.05.2024.
18. MnasNet: Platform-aware neural architecture search for mobile. CVPR, 2019 [Электронный ресурс]. – Режим доступа: https://arxiv.org/abs/1807.11626/. – Дата доступа 13.04.2024.
19. NumPy [Электронный ресурс]. – Режим доступа: https://numpy.org/. – Дата доступа: 20.05.2024.
20. Optimization Techniques in Deep Learning [Электронный ресурс]. – Режим доступа: https://medium.com/@datasciencejourney100\_83560/optimization-techniques-in-deep-learning-7d7c8057f740/. – Дата доступа 25.05.2024.
21. Programming Computer Vision with Python [Электронный ресурс]. – Режим доступа: https://dokumen.pub/programming-computer-vision-with-python-tools-and-algorithms-for-analyzing-images-0636920022923-9781449316549-1449316549.html/. – Дата доступа: 23.05.2024.
22. Python for Data Analysis [Электронный ресурс]. – Режим доступа: https://www.oreilly.com/library/view/python-for-data/9781491957653/. – Дата доступа 23.05.2024.
23. Stanislav Pidhorskyi, Ranya Almohsen, Donald A Adjeroh, Gianfranco Doretto. Generative Probabilistic Novelty Detection with Adversarial Autoencoders [Электронный ресурс]. – Режим доступа: https://arxiv.org/abs/1807.02588/. – Дата доступа 26.04.2024.
24. Understanding Image Classification: A Comprehensive Guide [Электронный ресурс]. – Режим доступа: https://pareto.ai/blog/image-classification/. – Дата доступа 25.05.2024.
25. Библиотека для машинного обучения Tensorflow [Электронный ресурс]. – Режим доступа: https://www.nic.ru/help/biblioteka-dlya-mashinnого-obucheniya-tensorflow-kak-ustanovit6tensorflow\_11703.html/. – Дата доступа 10.04.2024.
26. Блокнот с реализацией трансферного обучения [Электронный ресурс]. – Режим доступа: https://colab.research.google.com/drive/1-kNBUU2OQmv-YfIn-FdRL54NchplpuzG?usp=sharing/. – Дата доступа 23.05.2024.
27. Лукашевич М. М. Цифровая обработка изображений и распознавание образов: пособие. – Минск: БГУИР, 2023. – 72 с.
28. Метрики классификации и регрессии [Электронный ресурс]. – Режим доступа: https://education.yandex.ru/handbook/ml/article/metriki-klassifikacii-i-regressii/. – Дата доступа 28.05.2024.
29. Оптимизация в ML [Электронный ресурс]. – Режим доступа: https://education.yandex.ru/handbook/ml/article/optimizaciya-v-ml/. – Дата доступа 28.05.2024.
30. Тропченко А.Ю., Тропченко А.А. Методы сжатия изображений, аудиосигналов и видео: Учебное пособие – СПб: СПбГУ ИТМО, 2009. – 108 c.
31. Цифровые аудио сигналы [Электронный ресурс]. – Режим доступа: https://sergey-tlt.narod.ru/other/Digital\_audio\_signals.htm/. – Дата доступа 26.05.2024.
32. ЦП против ГП против ТПУ [Электронный ресурс]. – Режим доступа: https://www.mechantica.org/myfreedom/3/. – Дата доступа 24.05.2024.
33. Хурсов П. С., Искра Н. А. Алгоритмы детекции объектов для анализа изображений [Электронный ресурс]. – Режим доступа: https://libeldoc.bsuir.by/handle/123456789/37064/. – Дата доступа 04.04.2024.
34. Что такое трансферное обучение? [Электронный ресурс]. – Режим доступа: https://aws.amazon.com/ru/what-is/transfer-learning/. – Дата доступа 28.05.2024.
35. ПРОГРАМНЫЙ КОД ПРИМЕРА КЛАССИФИКАТОРА

# Импорт библиотек

import numpy as np

import tensorflow as tf

import matplotlib.pyplot as plt

from tensorflow import keras

%matplotlib inline

# импорт дата сета

from tensorflow.keras.datasets import fashion\_mnist

# класс для создания линейной модели нейронной сети.

from tensorflow.keras.models import Sequential

# классы для добавления полносвязных слоев и слоев регуляризации в модель нейронной сети.

from tensorflow.keras.layers import Dense, Dropout

# вспомогательные функции

from tensorflow.keras import utils

# разделение на тренировочную и тестовую выборки

(x\_train, y\_train), (x\_test, y\_test) = fashion\_mnist.load\_data()

# список названий классов

class\_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat', 'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']

# отображает первое изображение из тренировочного набора на цветовой шкале

plt.figure()

plt.imshow(x\_train[1])

plt.colorbar()

plt.grid(False)

# Нормализует данные, деля каждое значение пикселя на 255, чтобы привести их в диапазон от 0 до 1.

x\_train = x\_train / 255

x\_test = x\_test / 255

plt.figure()

plt.imshow(x\_train[1])

plt.colorbar()

plt.grid(False)

# первые 25 изображений из тренировочного набора в виде сетки 5x5

plt.figure(figsize=(10,10))

for i in range (25):

plt.subplot(5,5,i+1)

plt.xticks([])

plt.yticks([])

plt.imshow(x\_train[i], cmap=plt.cm.binary)

plt.xlabel(class\_names[y\_train[i]])

# Создает последовательную модель нейронной сети с тремя слоями:

# слой выравнивания (преобразует 2D-изображения в 1D-вектор),

# плотный слой с 128 нейронами и функцией активации ReLU и

# выходной слой с 10 нейронами и функцией активации softmax для предсказания классов.

model = keras.Sequential([

keras.layers.Flatten(input\_shape=(28,28)),

keras.layers.Dense(128, activation="relu"),

keras.layers.Dense(10,activation="softmax")

])

# Компилирует модель, указывая оптимизатор стохастического градиентного спуска (SGD),

# функцию потерь (sparse categorical crossentropy) и метрику точности.

model.compile(optimizer=tf.keras.optimizers.SGD(), loss='sparse\_categorical\_crossentropy', metrics=['accuracy'])

# Выводит сводку модели, показывая структуру модели, количество параметров и другие детали.

model.summary()

# Тренирует модель на тренировочных данных в течение 10 эпох.

model.fit(x\_train, y\_train, epochs=10)

# Оценивает модель на тестовых данных и возвращает потери и точность на тестовом наборе.

test\_loss, test\_acc = model.evaluate(x\_test, y\_test)

# Выводит точность модели на тестовом наборе данных.

print('Test accuracy:', test\_acc)

# Делает предсказания для тренировочных данных, возвращая вероятности для каждого класса.

predictions = model.predict(x\_train)

# Находит индекс класса с наибольшей вероятностью для 125-го изображения в тренировочном наборе.

np.argmax(predictions[124])

# Выводит реальную метку класса для 125-го изображения в тренировочном наборе.

y\_train[124]

plt.figure()

plt.imshow(x\_train[124])

plt.colorbar()

plt.grid(False)

# Выводит название класса с наибольшей вероятностью для 125-го изображения на основе предсказаний модели.

class\_names[np.argmax(predictions[124])]

1. ПРОГРАММНЫЙ КОД РЕАЛИЗАЦИИ СЖАТИЯ АУДИОСИГНАЛА

**# Изменение частоты дискретизации:**

import librosa

# Загрузка аудиофайла с сохранением исходной частоты дискретизации, где y

y, sr = librosa.load('audio.wav', sr=None)

# Снижение частоты дискретизации до 22050 Гц

y\_resampled = librosa.resample(y, orig\_sr=sr, target\_sr=22050)

# Сохранение ресемплированного аудиофайла с новой частотой дискретизации

librosa.output.write\_wav('resampled\_audio.wav',y\_resampled, sr=22050)

# Обработка и фильтрация:

y, sr = librosa.load('audio.wav', sr=None)

# Применение фильтрации

y\_filtered = librosa.effects.preemphasis(y)

librosa.output.write\_wav('filtered\_audio.wav', y\_filtered, sr)

# Прореживание данных (Downsampling)

import numpy as np

y, sr = librosa.load('audio.wav', sr=None)

# Прореживание данных, сохраняя каждую вторую выборку

y\_downsampled = y[::2]

librosa.output.write\_wav('downsampled\_audio.wav', y\_downsampled, sr)

# Сжатие динамического диапазона (Dynamic Range Compression)

y, sr = librosa.load('audio.wav', sr=None)

# Применение эффекта перкуссии для сжатия динамического диапазона

y\_compressed = librosa.effects.percussive(y)

librosa.output.write\_wav('compressed\_dynamic\_range.wav', y\_compressed, sr)

# Обработка тишины (Silence Removal)

y, sr = librosa.load('audio.wav', sr=None)

# Найти все непустые интервалы

non\_silent\_intervals = librosa.effects.split(y, top\_db=20)

y\_non\_silent = np.concatenate([y[start:end] for start, end in non\_silent\_intervals])

librosa.output.write\_wav('non\_silent\_audio.wav', y\_non\_silent, sr)

# Преобразование в моно (Convert to Mono)

y, sr = librosa.load('audio.wav', sr=None, mono=False)

# Преобразование в моно

y\_mono = librosa.to\_mono(y)

librosa.output.write\_wav('mono\_audio.wav', y\_mono, sr)

# Конвертация аудиоформатов в Pydub

from pydub import AudioSegment

sound = AudioSegment.from\_file('audio.wav')

# Конвертация в MP3 с битрейтом 64 кбит/с

sound.export('compressed\_audio.mp3', format='mp3', bitrate='64k')

import librosa

import soundfile as sf

import os

# Загрузка начального аудиофайла

y, sr = librosa.load('audio.wav', sr=None)

# Ресемплирование аудиосигнала до 22050 Гц

y\_resampled = librosa.resample(y, orig\_sr=sr, target\_sr=22050)

# Сохранение ресемплированного аудиофайла

sf.write('resampled\_audio.wav', y\_resampled, 22050)

# Сжатие ресемплированного аудиофайла с помощью pydub

from pydub import AudioSegment

compressed\_audio = AudioSegment.from\_file('resampled\_audio.wav')

compressed\_audio.export('compressed\_audio.mp3', format='mp3', bitrate='64k')

# Восстановление аудиофайла

restored\_audio = AudioSegment.from\_file('compressed\_audio.mp3')

restored\_audio.export('restored\_audio.wav', format='wav')

# Получение размеров файлов

initial\_size = os.path.getsize('audio.wav')

resampled\_size = os.path.getsize('resampled\_audio.wav')

compressed\_size = os.path.getsize('compressed\_audio.mp3')

restored\_size = os.path.getsize('restored\_audio.wav')

print(f'Initial file size: {initial\_size / 1024:.2f} KB')

print(f'Resampled file size: {resampled\_size / 1024:.2f} KB')

print(f'Compressed file size: {compressed\_size / 1024:.2f} KB')

print(f'Restored file size: {restored\_size / 1024:.2f} KB')

1. ПРОГРАММНЫЙ КОД РЕАЛИЗАЦИИ ТРАНСФЕРНОГО ОБУЧЕНИЯ

#Импорт основных библиотек

# Import Data Science Libraries

import numpy as np

import pandas as pd

import tensorflow as tf

from sklearn.model\_selection import train\_test\_split

import itertools

import random

# Import visualization libraries

import matplotlib.pyplot as plt

import matplotlib.cm as cm

import cv2

import seaborn as sns

# Tensorflow Libraries

from tensorflow import keras

from tensorflow.keras import layers,models

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.layers import Dense, Dropout, BatchNormalization

from tensorflow.keras.callbacks import Callback, EarlyStopping,ModelCheckpoint, ReduceLROnPlateau

from tensorflow.keras import Model

from tensorflow.keras.layers.experimental import preprocessing

from tensorflow.keras.optimizers import Adam

# System libraries

from pathlib import Path

import os.path

# Metrics

from sklearn.metrics import classification\_report, confusion\_matrix

sns.set\_style('darkgrid')

#Загрузка данных, определение размера батча

batch\_size = 32

target\_size = (224, 224)

dataset = "/kaggle/input/animals10/raw-img"

walk\_through\_dir(dataset)

#Разделение данных на 3 выборки

train\_df, test\_df = train\_test\_split(image\_df, test\_size=0.2, shuffle=True, random\_state=42)

# Создание генераторов изображений

train\_generator = ImageDataGenerator(

preprocessing\_function=tf.keras.applications.efficientnet.preprocess\_input,

validation\_split=0.2

)

test\_generator = ImageDataGenerator(

preprocessing\_function=tf.keras.applications.efficientnet.preprocess\_input,

)

train\_images = train\_generator.flow\_from\_dataframe(

dataframe=train\_df,

x\_col='Filepath',

y\_col='Label',

target\_size=TARGET\_SIZE,

color\_mode='rgb',

class\_mode='categorical',

batch\_size=BATCH\_SIZE,

shuffle=True,

seed=42,

subset='training'

)

val\_images = train\_generator.flow\_from\_dataframe(

dataframe=train\_df,

x\_col='Filepath',

y\_col='Label',

target\_size=TARGET\_SIZE,

color\_mode='rgb',

class\_mode='categorical',

batch\_size=BATCH\_SIZE,

shuffle=True,

seed=42,

subset='validation'

)

test\_images = test\_generator.flow\_from\_dataframe(

dataframe=test\_df,

x\_col='Filepath',

y\_col='Label',

target\_size=TARGET\_SIZE,

color\_mode='rgb',

class\_mode='categorical',

batch\_size=BATCH\_SIZE,

shuffle=False

)

**Реализация обучения на изображениях животных**

# Загрузка предобученной модели

pretrained\_model = tf.keras.applications.efficientnet.EfficientNetB7(

input\_shape=(224, 224, 3),

include\_top=False,

weights='imagenet',

pooling='max'

)

pretrained\_model.trainable = False

# Создание чек-поинтов ранней остановки обучения

# Create checkpoint callback

checkpoint\_path = "animals\_classification\_model\_checkpoint"

checkpoint\_callback = ModelCheckpoint(checkpoint\_path,

save\_weights\_only=True,

monitor="val\_accuracy",

save\_best\_only=True)

# Setup EarlyStopping callback to stop training if model's val\_loss doesn't improve for 3 epochs

early\_stopping = EarlyStopping(monitor = "val\_loss",

# watch the val loss metric

patience = 5,

restore\_best\_weights = True) # if val loss decreases for 3 epochs in a row, stop training

reduce\_lr = ReduceLROnPlateau(monitor='val\_loss', factor=0.2, patience=3, min\_lr=1e-6)

#Определение слоев модели и ее тренировка

inputs = pretrained\_model.input

x = augment(inputs)

x = Dense(128, activation='relu')(pretrained\_model.output)

x = BatchNormalization()(x)

x = Dropout(0.45)(x)

x = Dense(256, activation='relu')(x)

x = BatchNormalization()(x)

x = Dropout(0.45)(x)

outputs = Dense(10, activation='softmax')(x)

model = Model(inputs=inputs, outputs=outputs)

model.compile(

optimizer=Adam(0.00001),

loss='categorical\_crossentropy',

metrics=['accuracy']

)

history = model.fit(

train\_images,

steps\_per\_epoch=len(train\_images),

validation\_data=val\_images,

validation\_steps=len(val\_images),

epochs=15,

callbacks=[

early\_stopping,

create\_tensorboard\_callback("training\_logs",

"animals\_classification"),

checkpoint\_callback,

reduce\_lr

]

)

# Построение метрик

accuracy = history.history['accuracy']

val\_accuracy = history.history['val\_accuracy']

loss = history.history['loss']

val\_loss = history.history['val\_loss']

epochs = range(len(accuracy))

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 5))

ax1.plot(epochs, accuracy, 'b', label='Training accuracy')

ax1.plot(epochs, val\_accuracy, 'r', label='Validation accuracy')

ax1.set\_title('Training and validation accuracy')

ax1.set\_xlabel('Epochs')

ax1.set\_ylabel('Accuracy')

ax1.legend()

ax2.plot(epochs, loss, 'b', label='Training loss')

ax2.plot(epochs, val\_loss, 'r', label='Validation loss')

ax2.set\_title('Training and validation loss')

ax2.set\_xlabel('Epochs')

ax2.set\_ylabel('Loss')

ax2.legend()

fig.suptitle('Training and validation metrics', fontsize=16)

plt.show()

# Предсказание класса случайных 25-ти изображений

# Predict the label of the test\_images

pred = model.predict(test\_images)

pred = np.argmax(pred,axis=1)

# Map the label

labels = (train\_images.class\_indices)

labels = dict((v,k) for k,v in labels.items())

pred = [labels[k] for k in pred]

# Display the result

print(f'The first 5 predictions: {pred[:5]}')

# Display 25 random pictures from the dataset with their labels

random\_index = np.random.randint(0, len(test\_df) - 1, 15)

fig, axes = plt.subplots(nrows=3, ncols=5, figsize=(25, 15),

subplot\_kw={'xticks': [], 'yticks': []})

for i, ax in enumerate(axes.flat):

ax.imshow(plt.imread(test\_df.Filepath.iloc[random\_index[i]]))

if test\_df.Label.iloc[random\_index[i]] == pred[random\_index[i]]:

color = "green"

else:

color = "red"

ax.set\_title(f"True: {test\_df.Label.iloc[random\_index[i]]}\nPredicted: {pred[random\_index[i]]}", color=color)

plt.show()

plt.tight\_layout()

**Реализация дообучения**

BATCH\_SIZE = 10

TARGET\_SIZE = (224, 224)

image\_df\_toy = convert\_path\_to\_df(dataset\_toy)

train\_df\_toy, test\_df\_toy = train\_test\_split(image\_df\_toy, test\_size=0.2, shuffle=True, random\_state=42)

train\_generator\_toy = ImageDataGenerator(

preprocessing\_function=tf.keras.applications.efficientnet.preprocess\_input,

validation\_split=0.2

)

test\_generator\_toy = ImageDataGenerator(

preprocessing\_function=tf.keras.applications.efficientnet.preprocess\_input,

)

train\_images\_toy = train\_generator\_toy.flow\_from\_dataframe(

dataframe=train\_df\_toy,

x\_col='Filepath',

y\_col='Label',

target\_size=TARGET\_SIZE,

color\_mode='rgb',

class\_mode='categorical',

batch\_size=BATCH\_SIZE,

shuffle=True,

seed=42,

subset='training'

)

val\_images\_toy = train\_generator\_toy.flow\_from\_dataframe(

dataframe=train\_df\_toy,

x\_col='Filepath',

y\_col='Label',

target\_size=TARGET\_SIZE,

color\_mode='rgb',

class\_mode='categorical',

batch\_size=BATCH\_SIZE,

shuffle=True,

seed=42,

subset='validation'

)

test\_images\_toy = test\_generator\_toy.flow\_from\_dataframe(

dataframe=test\_df\_toy,

x\_col='Filepath',

y\_col='Label',

target\_size=TARGET\_SIZE,

color\_mode='rgb',

class\_mode='categorical',

batch\_size=BATCH\_SIZE,

shuffle=False

)

**Пользовательская аугментация**

out\_height, out\_width = 224,224

resize = tf.keras.layers.Resizing(out\_height, out\_width)

height = tf.keras.layers.RandomHeight(0.3)

width = tf.keras.layers.RandomWidth(0.3)

zoom = tf.keras.layers.RandomZoom(0.3)

flip = tf.keras.layers.RandomFlip("horizontal\_and\_vertical")

rotate = tf.keras.layers.RandomRotation(0.2)

crop = tf.keras.layers.RandomCrop(out\_height, out\_width)

translation = tf.keras.layers.RandomTranslation(height\_factor=0.2, width\_factor=0.2)

brightness = tf.keras.layers.RandomBrightness([-0.8,0.8])

contrast = tf.keras.layers.RandomContrast(0.2)

# Компиляция и обучение модели

inputs = model.input

x = augment(inputs)

x = Dense(128, activation='relu')(model.output)

x = BatchNormalization()(x)

x = Dropout(0.45)(x)

#x = Dense(256, activation='relu')(x)

#x = BatchNormalization()(x)

#x = Dropout(0.45)(x)

outputs = Dense(10, activation='softmax')(x)

model\_toy = Model(inputs=inputs, outputs=outputs)

model\_toy.compile(

optimizer=Adam(0.00001),

loss='categorical\_crossentropy',

metrics=['accuracy']

)

history = model\_toy.fit(

train\_images\_toy,

steps\_per\_epoch=len(train\_images\_toy),

validation\_data=val\_images\_toy,

validation\_steps=len(val\_images\_toy),

epochs=100,

callbacks=[

early\_stopping,

create\_tensorboard\_callback("training\_logs",

"animals\_classification"),

checkpoint\_callback,

reduce\_lr

]

)

**Предсказание и отображение результатов классификации**

# Display 25 random pictures from the dataset with their labels

random\_index = np.random.randint(0, len(test\_df\_toy) - 1, 15)

fig, axes = plt.subplots(nrows=3, ncols=5, figsize=(25, 15),

subplot\_kw={'xticks': [], 'yticks': []})

for i, ax in enumerate(axes.flat):

ax.imshow(plt.imread(test\_df\_toy.Filepath.iloc[random\_index[i]]))

if test\_df\_toy.Label.iloc[random\_index[i]] == pred[random\_index[i]]:

color = "green"

else:

color = "red"

ax.set\_title(f"True: {test\_df\_toy.Label.iloc[random\_index[i]]}\nPredicted: {pred[random\_index[i]]}", color=color)

plt.show()

plt.tight\_layout()