МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ  
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ**

**Н. Г. ЧЕРНЫШЕВСКОГО»**

Кафедра информатики и программирования

**««Разработка мобильного приложения для распознавания маршрутных номеров с использованием нейронных сетей»**

**КУРСОВАЯ РАБОТА**

студента 3 курса 341 группы

Протасова Ивана

направления 02.03.03 Математическое обеспечение и администрирование

информационных систем

факультета компьютерных наук и информационных технологий

Научный руководитель \_\_\_\_\_\_\_\_\_\_ Е.Е. Лапшева

Старший преподаватель

Зав. кафедрой

к.ф.-м.н., доцент \_\_\_\_\_\_\_\_\_\_ М.В. Огнева

**Саратов 2025**

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc198231736)

[1. Обзор существующих решений 5](#_Toc198231737)

[2. Обзор и выбор технологий для реализации приложения 6](#_Toc198231738)

[2.1. Языки программирования 6](#_Toc198231739)

[2.2. Библиотеки и фреймворки 7](#_Toc198231740)

[2.3. Инструменты мобильной разработки 8](#_Toc198231741)

[2.4. Инструменты тестирования 9](#_Toc198231742)

[2.5. Инструмент разметки датасета 10](#_Toc198231743)

[3. Описание предметной области 10](#_Toc198231744)

[3.1. Компьютерное зрение 10](#_Toc198231745)

[3.1.1. Задача распознавания объектов 11](#_Toc198231746)

[3.1.2. Задача распознавания текста 14](#_Toc198231747)

[3.2. Свёрточные нейронные сети 15](#_Toc198231748)

[3.3. Архитектура свёрточной нейронной сети 17](#_Toc198231749)

[3.4. Обучение свёрточной нейронной сети 19](#_Toc198231750)

[3.6. Работа сети после завершения обучения 21](#_Toc198231751)

[4. Практическая часть 21](#_Toc198231752)

[4.1. Подготовка датасета 21](#_Toc198231753)

[5. Глава-комментарий 24](#_Toc198231754)

[ЗАКЛЮЧЕНИЕ 25](#_Toc198231755)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 26](#_Toc198231756)

# ВВЕДЕНИЕ

**Актуальность** данной темы заключается в необходимости создания решений для облегчения доступа слабовидящих людей к общественному транспорту. Существующие методы, такие как звуковые оповещения и дисплеи высокой контрастности, не всегда эффективны. Использование методов компьютерного зрения позволяет предложить современное и персонализированное решение. Несмотря на успехи в данной области, включая алгоритмы OCR (optical character recognition) и технологии обработки изображений, специализированных решений для распознавания номеров автобусов в сложных условиях пока недостаточно. Важным аспектом данной работы является создание собственного набора данных и разработка специализированной нейронной сети, что позволит учесть специфику задачи и улучшить точность и надежность распознавания.

**Цель курсовой работы:** разработать мобильное приложение с интегрированной нейронной сетью, предназначенное для распознавания вида общественного транспорта и его номера в условиях городской среды.

**Задачи научно-исследовательской работы:**

1. Изучить теоретические основы компьютерного зрения и свёрточных нейронных сетей, необходимых для решения задачи распознавания объектов и текста.
2. Проанализировать существующие программные решения и выбрать наиболее подходящие технологии и инструменты для разработки мобильного приложения и обучения нейросети.
3. Подготовить датасет, провести обучение модели нейронной сети и выполнить её оптимизацию для мобильных устройств.
4. Разработать Android-приложение с использованием выбранных инструментов и интегрировать в неё обученную модель.
5. Провести тестирование и оценку эффективности работы разработанной системы в приближённых к реальным условиях.

# ****1. Обзор существующих решений****

На сегодняшний день существует множество решений, направленных на распознавание текста и объектов с помощью компьютерного зрения. Однако большинство из них ориентированы на другие задачи или имеют ограничения, которые делают их недостаточно эффективными для распознавания номеров автобусов в условиях городской среды.

**Системы автоматического распознавания номеров (ANPR)**  
широко используются для распознавания автомобильных номеров. Они основаны на алгоритмах компьютерного зрения, таких как YOLO или SSD для детекции объекта и OCR для извлечения текста. Эти системы демонстрируют высокую точность в контролируемых условиях (фиксированные камеры, стабильное освещение). Однако их применение ограничено, поскольку они не предназначены для мобильных устройств, работы в сложных условиях освещения или распознавания номеров на движущемся транспорте. [1]

**Seeing AI** от Microsoft способен распознавать текст и объекты, предоставляя голосовую обратную связь. Тем не менее, это приложение являются универсальным и не адаптировано для задач, связанных с идентификацией номеров автобусов. Это приводит к снижению эффективности и удобства использования в подобных сценариях. [2]

**Инфраструктурные решения в транспорте некоторых** городов, такие как внедрение систем озвучивания маршрутов или дисплеев высокой контрастности на автобусах, требуют значительных финансовых вложений и модернизации инфраструктуры, что делает их недоступными для большинства регионов. Более того, их эффективность ограничена, так как они не учитывают индивидуальные потребности слабовидящих пользователей.

**Таким образом**, несмотря на наличие аналогов, ни одно из существующих решений не удовлетворяет всем требованиям задачи распознавания номеров автобусов для слабовидящих пользователей.

# ****2.**** Обзор и выбор технологий для реализации приложения

## 2.1. Языки программирования

**Python** — это высокоуровневый язык программирования, который широко используется в области машинного обучения и искусственного интеллекта. Он имеет богатую экосистему библиотек для работы с данными, таких как NumPy, Pandas, PyTorch, TensorFlow, а также поддерживает удобные фреймворки для разработки и обучения нейронных сетей.

* **Почему выбран**: Python был выбран для разработки нейронной сети в связи с его высокой продуктивностью в области искусственного интеллекта и машинного обучения. Он обладает простой и понятной синтаксисической структурой, что позволяет быстро прототипировать и разрабатывать модели, а также активно поддерживается научным сообществом.
* **Альтернативы**: C++ может быть более эффективным по времени выполнения и имеет большой контроль над низкоуровневыми операциями, что важно для разработки высокоскоростных систем, однако не имеет таких удобных и мощных библиотек для машинного обучения, что делает процесс разработки более трудозатратным. [3]

**Kotlin** — это современный язык программирования, который используется для создания приложений под Android. Kotlin полностью совместим с Java, но обладает более современным синтаксисом и функциональными возможностями.

* **Почему выбран**: Kotlin выбран для создания Android-приложения, поскольку это официальный язык разработки для Android, поддерживаемый Google. Он предоставляет безопасную работу с нулевыми значениями, удобную работу с функциональными методами и лаконичный синтаксис, что делает разработку более быстрой и менее подверженной ошибкам.
* **Альтернативы**: **Java** является еще одним популярным языком для Android-разработки, однако Kotlin предлагает более простое и выразительное решение для современных приложений. [4]

## 2.2. Библиотеки и фреймворки

**PyTorch** — это популярная библиотека для работы с нейронными сетями, которая позволяет легко строить, обучать и развертывать модели машинного обучения. Она поддерживает динамическое вычисление графов, что делает её удобной для быстрого прототипирования.

* **Почему выбран**: PyTorch был выбран за свою гибкость, удобство в обучении нейронных сетей и активную поддержку мобильных платформ через **PyTorch Mobile**. PyTorch также имеет развитую документацию и сообщество, что значительно облегчает процесс разработки и внедрения.
* **Альтернативы**: **TensorFlow** — конкурент PyTorch, предоставляющий аналогичные возможности, однако для мобильных приложений PyTorch Mobile обеспечивает более простую интеграцию и поддержку. [5]

**TorchScript** — это способ сериализации моделей PyTorch, который позволяет использовать их на мобильных устройствах. С помощью TorchScript модель PyTorch конвертируется в формат, который можно легко загрузить и выполнить на Android.

* **Почему выбран**: TorchScript выбран для конвертации модели в формат, совместимый с Android-приложением. Это позволяет использовать ту же модель, которая была обучена в Python, в мобильном приложении, обеспечивая высокий уровень производительности и совместимости.
* **Альтернативы**: **ONNX** (Open Neural Network Exchange) также является форматом для сериализации моделей и может быть использован для работы с TensorFlow, PyTorch и другими библиотеками. Однако для использования PyTorch в Android приложении TorchScript является более естественным выбором.

## 2.3. Инструменты мобильной разработки

**Android Studio** — это официальная среда разработки для Android, которая предоставляет весь необходимый инструментарий для создания приложений, включая редактор кода, эмуляторы, инструменты для отладки и анализа производительности.

* **Почему выбран**: Android Studio это официальная среда разработки для Android, которая поддерживает все необходимые инструменты для интеграции с PyTorch и OpenCV.
* **Альтернативы**: **IntelliJ IDEA** также поддерживает Android-разработку, но Android Studio предоставляет более тесную интеграцию с платформой Android.

**Gradle** — это инструмент для автоматизации сборки, который используется для управления зависимостями, настройки и сборки проекта. Он широко используется в экосистеме Android.

* **Почему выбран**: Gradle это стандартный инструмент для управления зависимостями и сборки проектов на Android, и он идеально подходит для интеграции библиотек, таких как OpenCV, PyTorch и CameraX.
* **Альтернативы**: **Maven** — еще один инструмент для управления зависимостями, однако Gradle предоставляет больше возможностей для гибкой настройки сборки в Android-приложениях.

**CameraX** — это инструменты для работы с камерами на Android. CameraX упрощает работу с камерой, обеспечивая простоту использования и поддержку различных типов устройств.

* **Почему выбран**: CameraX был выбран для работы с камерой, так как он упрощает захват изображений, что особенно важно для приложения, использующего компьютерное зрение для распознавания номеров автобусов. [6]
* **Альтернативы**: **Camera2 API** предоставляет более низкоуровневый доступ к камере, что дает больше возможностей, но требует более сложной настройки. CameraX является более высокоуровневым API, которое подходит для большинства случаев использования.

**OpenCV (for android)** — это библиотека для обработки изображений и видео в реальном времени. В контексте мобильных приложений, она предоставляет удобный набор инструментов для работы с изображениями и видео.

* **Почему выбран**: OpenCV был выбран для обработки изображений с камеры, так как эта библиотека предоставляет широкий функционал для манипуляции изображениями, а также для предварительной обработки данных перед подачей их в нейронную сеть.
* **Альтернативы**: **TensorFlow Lite** также предоставляет инструменты для обработки изображений, но OpenCV предлагает более широкие возможности и гибкость в области обработки изображений.

## 2.4. Инструменты тестирования

**Android Emulator** — это инструмент для тестирования Android-приложений на различных устройствах без необходимости использовать физическое устройство.

* **Почему выбран**: Android Emulator был выбран для тестирования приложения, так как он позволяет имитировать работу различных Android-устройств, что важно для проверки совместимости и функциональности приложения.
* **Альтернативы**: ф**изическое устройство** — может быть использовано для тестирования, однако Android Emulator предоставляет более быстрый и удобный способ тестирования на разных устройствах.

## 2.5. Инструмент разметки датасета

**Computer Vision Annotation Tool (CVAT)** — это веб-инструмент с открытым исходным кодом, предназначенный для разметки изображений и видео в задачах компьютерного зрения. CVAT поддерживает такие типы аннотаций, как прямоугольные рамки (bounding boxes), многоугольники, полилинии, ключевые точки и трассировка объектов во времени.

* **Почему выбран**: CVAT предоставляет удобный и интуитивно понятный интерфейс для аннотирования изображений, позволяет создавать задачи, распределять их между участниками, поддерживает экспорт в различные форматы (COCO, Pascal VOC, CVAT for images), а также имеет встроенную валидацию и отслеживание качества разметки.
* **Альтернативы**: LabelImg — лёгкий инструмент для создания bounding box аннотаций, однако он менее удобен при работе с большими датасетами. Label Studio — более универсален и настраиваем, но требует дополнительной конфигурации и не так хорошо оптимизирован под задачи компьютерного зрения.

# ****3. Описание предметной области****

## 3.1. Компьютерное зрение

Компьютерное зрение — это область искусственного интеллекта, направленная на создание алгоритмов и систем, которые могут интерпретировать и анализировать содержимое цифровых изображений и видео. Основная задача состоит в том, чтобы компьютер «понимал» изображения или видеопотоки так же, как это делает человек. С точки зрения техники, компьютерное зрение включает в себя обработку данных (цифровых изображений) и применение алгоритмов для выполнения задач, таких как распознавание, сегментация, отслеживание и анализ объектов.

Ключевая особенность компьютерного зрения — способность машин анализировать визуальные данные в реальном времени и адаптироваться к динамически изменяющимся условиям. Это качество имеет решающее значение для задач, связанных с городской средой, где изображения часто содержат большое количество объектов и сложные условия освещения.

### 3.1.1. Задача распознавания объектов

Распознавание объектов (Object Detection) — одна из центральных задач в компьютерном зрении. Её цель состоит в определении местоположения объектов определённых классов на изображении. Задача включает два этапа: детекция объектов и классификация объектов. [7]

На рисунке 1 представлена схема работы одного из алгоритмов распознавания объектов – R-CNN:

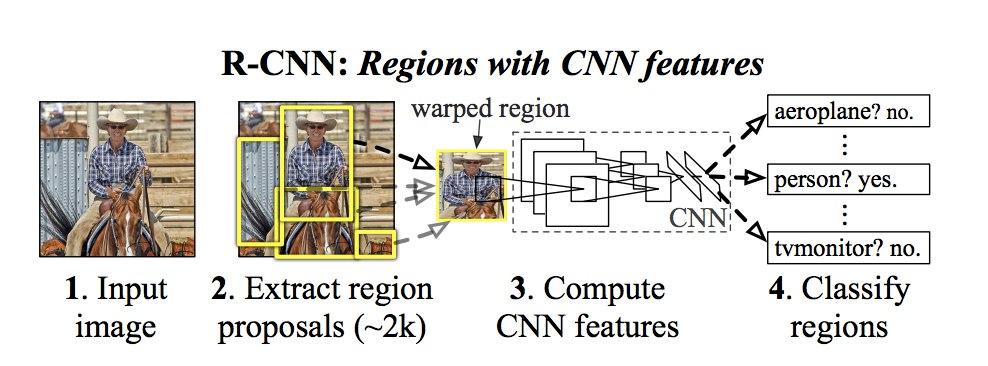


Рисунок 1 — Схема работы алгоритма R-CNN

**Терминология:**

* **Ограничивающая рамка (bounding box)** – координаты, ограничивающие определенную область изображения, – чаще всего в форме прямоугольника. Может быть представлена в виде четырёх координат в двух форматах: центрированный () и обычный ().
* **Гипотеза (Proposal), P** – определенный регион изображения (заданный с помощью ограничивающей рамки), в котором предположительно находится объект.
* **IoU (Intersection-over-Union)** – метрика степени пересечения между двумя гипотезами. Определяется как отношение области пересечения (area of overlap) и области объединения (area of union) их рамок. На рисунке 2 приведена наглядная иллюстрация определения метрики IoU.



Рисунок 2 — Определение метрики IoU

**Детекция объектов** — это процесс нахождения и выделения области, в которой расположен объект интереса, на изображении или видео. Она включает не только идентификацию наличия объекта, но и определение его местоположения. Основные этапы детекции:

1. Локализация объекта**:** определение координат ограничивающей рамки, в которой предположительно находится объект.
2. Анализ контекста**:** использование дополнительных признаков (большего количества гипотез) для подтверждения наличия объекта.

Для обнаружения объекта изображение делится на фрагменты в виде прямоугольных рамок, что позволяет модели выявлять ключевые признаки и определять расположение объекта. Эффективность модели оценивается с помощью метрики IoU, которая измеряет совпадение между истинной рамкой объекта и предполагаемой моделью. Этот метод оценки иллюстрируется на рисунке 3.

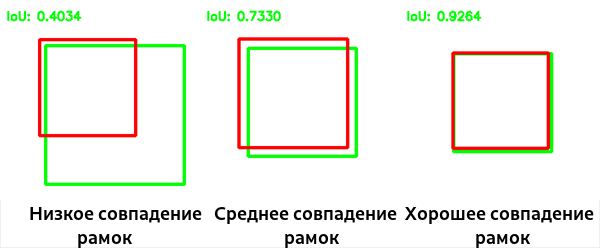


Рисунок 3 — Степень совпадения рамок

**Классификация объектов** — это процесс присвоения объекта к одному из предопределённых классов. Основные этапы детекции **классификации:**

1. **Подготовка данных:**
   * Удаление шумов и улучшение качества изображения (например, выравнивание освещения).
   * Преобразование выделенной области в формат, пригодный для подачи в нейронную сеть.
2. **Извлечение признаков:**
   * Использование алгоритмов глубокого обучения, чтобы извлечь особенности объекта, такие как контуры и текстуры.
3. **Присвоение класса:**
   * На основе извлечённых признаков объект классифицируется как определённый номер или символ.

В задаче распознавания номеров автобусов детекция помогает выделить область табло автобуса, после чего происходит дальнейший анализ для считывания номера. Это особенно важно в условиях городской среды, где объекты могут частично перекрываться или находиться в движении.

### 3.1.2. Задача распознавания текста

Распознавание текста (Optical Character Recognition, OCR) направлено на извлечение текстовой информации из изображений. Основная сложность OCR состоит в том, что текст может быть написан разными шрифтами, находиться под углом или быть частично затенённым. [8]

На рисунке 4 представлена общая схема работы алгоритмов распознавания текста:

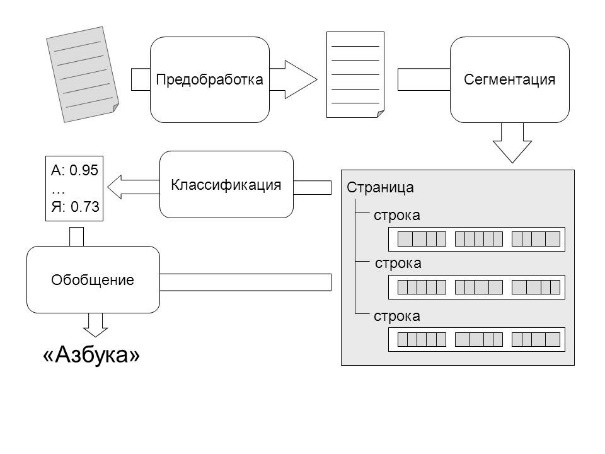


Рисунок 4 — Схема работы алгоритма, решающий задачи распознавания текста

1. Предобработка изображения:
   * Градации серого: преобразование изображения в оттенки серого для уменьшения размерности данных.
   * Бинаризация: преобразование изображения в двоичный формат (черное-белое) для акцентирования текста.
   * Удаление шумов: фильтрация мелких дефектов, вызванных внешними условиями (например, дождем или пылью).
   * Выравнивание: коррекция перспективных искажений текста из-за углов съемки.
2. Детекция текста:
   * На данном этапе выделяются области, содержащие текст
3. Распознавание текста (OCR):
   * После выделения текстовой области используется Optical Character Recognition - технология преобразования текстовой информации в цифровой, машиночитаемый текст.
4. Постобработка результатов
   * Исправление ошибок: использование словарей или статистических моделей для устранения ошибок распознавания.
   * Устранение ложных срабатываний: фильтрация неверных областей, например, случайных шумов, распознанных как текст.

В контексте задачи распознавания номеров автобусов OCR используется для извлечения числовой информации с табло автобуса, что требует адаптации алгоритмов к условиям городской среды, например, к низкому качеству изображения или яркому солнечному свету.

## 3.2. Свёрточные нейронные сети

Свёрточные нейронные сети (Convolutional Neural Networks, CNN) — это класс глубоких нейронных сетей, которые особенно эффективны в задачах обработки изображений. CNN автоматически извлекают признаки из изображений, такие как контуры, текстуры и формы, что делает их идеальными для задач компьютерного зрения. [9]

CNN состоят из нескольких основных компонентов:

* **Свертка** — основной строительный блок CNN. Она заключается в применении фильтров к входным данным для выделения различных признаков, таких как края, текстуры и формы. Фильтр скользит по изображению и вычисляет свертку, создавая карту признаков. Этот процесс позволяет выделить важные детали изображения, которые затем используются для дальнейшей обработки. С математической точки зрения результатом операции свёртки над матрицами *A* (размером ) и *B* (размером ) является матрица *C* (размером ), элементы который вычисляются следующей формулой: . Работа операции свертки для двух матриц представлена на рисунке 5.

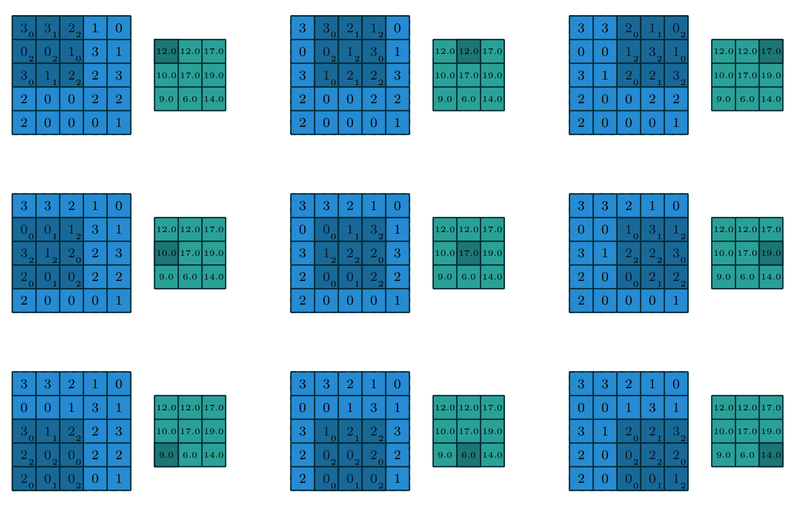


Рисунок 5 — Пример свертки двух матриц

* После свертки применяется **функция активации**, которая добовляет нелинейность в модель. Это помогает ей лучше справляться с сложными задачами, так как нелинейные функции позволяют нейронной сети моделировать более сложные зависимости.
* **Пулинг** (субдискретизация) уменьшает размер карты признаков, сохраняя наиболее значимые признаки. Это помогает уменьшить количество параметров и вычислительных затрат, сохраняя при этом важные характеристики изображения. На рисунке 6 представлена работа операции пулинга с функцией максимума.



Рисунок 6 — Пример операции пулинга с функцией максимума

Архитектуры CNN предоставляют гибкость в настройке модели для работы на устройствах с ограниченными вычислительными ресурсами, например, на смартфонах. В задаче распознавания номеров автобусов CNN используются для детекции табло и распознавания текста. [10]

## 3.3. Архитектура свёрточной нейронной сети

В общем виде CNN модели представляют собой входной слой и выходной слой, а также расположенный между ними набор слоёв свёртки, слоёв активации, слоёв пулинга и полносвязных слоёв. Общую схему работы CNN моделей с классической архитектурой можно увидеть на рисунке 7.

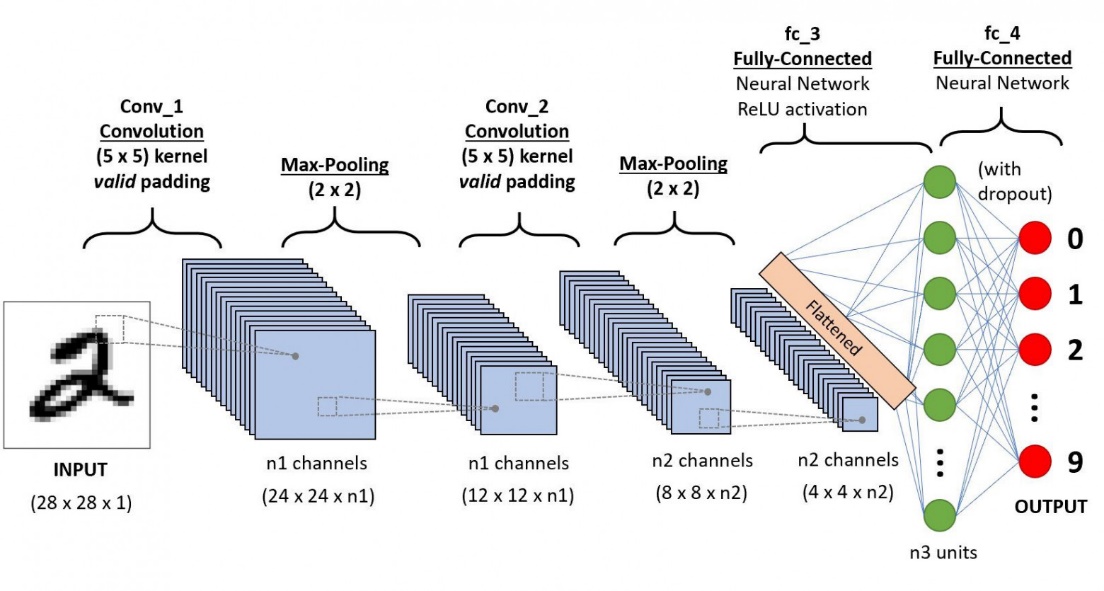


Рисунок 7 — Схема работы CNN моделей

Перед подачей в сеть изображения проходят предобработку: изменение размера (например, до 128x128 пикселей), нормализация значений пикселей и преобразование формата данных в тензор. **Входной слой** принимает данный тензор (то есть массив чисел) для дальнейшей обработки нейронной сетью.

**Сверточные слои** выполняют свертку с различными фильтрами для выделения признаков. Каждый сверточный слой создает несколько карт признаков, которые затем передаются на следующий слой. Эти слои являются основными компонентами CNN и выполняют основную работу по выделению признаков из изображения. Каждый сверточный слой может использовать несколько фильтров, каждый из которых выделяет разные признаки. Для распознавания текста на автобусах свёрточные слои помогают выделять ключевые элементы, такие как формы цифр. Глубокие слои могут изучать более сложные абстракции, например, стиль шрифта или расположение текста.

**Слои активации** применяют функцию активации (например, ReLU) к картам признаков, добавляя нелинейность в модель. Это позволяет модели лучше справляться с сложными задачами, так как нелинейные функции позволяют нейронной сети моделировать более сложные зависимости.

**Пулинговые слои** уменьшают размер карт признаков, сохраняя важные признаки и уменьшая вычислительные затраты. Это помогает модели быть более эффективной и быстрой, особенно при обработке больших изображений.

**Полносвязные слои** соединяют все нейроны предыдущего слоя с каждым нейроном текущего слоя. Они используются для окончательной классификации или регрессии. Эти слои являются завершающими компонентами CNN и выполняют основную работу по принятию решений на основе выделенных признаков. В задаче распознавания номера автобуса выход полносвязных слоев может быть интерпретирован как вероятности принадлежности изображения к различным номерам.

**Выходной слой** генерирует окончательный результат, для задачи распознавания текста это может быть один из заранее заданных классов (например, цифры от 0 до 9 и буквы).

## 3.4. Обучение свёрточной нейронной сети

**Датасет**: для обучения модели необходимо собрать и разметить набор данных. В данному случае это изображения автобусов с чётко видимыми номерами. Данные проходят предобработку: кадрирование для выделения области с номером, изменение размера, нормализация. Для улучшения качества обучения применяется аугментация данных — добавление шумов, повороты, изменение яркости и контраста, чтобы сеть могла адаптироваться к реальным условиям.

**Гиперпараметры сети**:

* **Размер свёрток** определяет область изображения, которую анализирует фильтр за один проход. В выбранной архитектуре используется размер 3x3. Это значение позволяет эффективно выделять локальные признаки, такие как линии, границы и мелкие текстовые элементы, не увеличивая избыточность параметров.
* **Количество фильтров** в каждом слое определяет, сколько различных признаков будет извлечено на каждом уровне. В начальных слоях используется 32 фильтра для захвата базовых признаков (например, текстур и краёв). В средних слоях это число увеличивается до 64, чтобы извлечь более сложные паттерны. В глубоких слоях 128 фильтров используются для формирования высокоуровневых признаков, таких как формы букв или цифр.
* **Шаг свёртки** определяет, насколько фильтр смещается по изображению при обработке. Выбранный шаг равен 1, что позволяет сохранить высокую пространственную детализацию изображения.
* **Размер batch** определяет количество примеров, обрабатываемых моделью за один проход. В данном случае выбрано значение 32, которое позволяет эффективно использовать ресурсы графического процессора, обеспечивая стабильность вычислений и плавную сходимость модели.
* ReLU (Rectified Linear Unit) используется как **функция активации**. Её простота и эффективность обусловлены тем, что она устраняет проблему затухания градиента и ускоряет обучение модели. ReLU позволяет сохранять значимую информацию для последующих слоёв, оставляя только положительные значения.
* Алгоритм Adam (Adaptive Moment Estimation) выбран в качестве **оптимизатора**. Он автоматически корректирует шаг градиента для каждого параметра, что делает его подходящим для задач с шумными данными.
* **Функция потерь:** Cross-Entropy Loss используется для измерения расхождения между предсказанными вероятностями и истинными метками. Эта функция оптимальна для задач классификации, таких как определение номера автобуса. Она минимизирует ошибку, способствуя повышению точности модели.
* **Пуллинг:** MaxPooling (размер 2x2, шаг 2) используется для уменьшения размерности карты признаков, сохраняя наиболее важные из них. Этот подход снижает вычислительные затраты и уменьшает риск переобучения, сохраняя при этом ключевые характеристики текста.

Процесс обучения заключается в подаче размеченных изображений на вход сети, вычислении функции потерь, обновлении весов с помощью градиентного спуска и проверки результатов на валидационной выборке. Обучение продолжается до достижения оптимального значения точности. [11, 12]

## 3.6. Работа сети после завершения обучения

После обучения модель экспортируется в формат, подходящий для мобильного приложения (TorchScript). Это позволяет запускать модель на устройстве без серверной обработки. Интеграция осуществляется с использованием PyTorch Mobile.

Изображения с камеры смартфона поступают в приложение, где проходят предобработку, аналогичную той, что использовалась при обучении. Это обеспечивает соответствие формата данных и корректность работы сети.

Сеть предсказывает номер автобуса на изображении, преобразует результат в текстовую или голосовую форму. Для удобства пользователей, особенно слабовидящих, информация отображается крупным шрифтом или озвучивается через аудиоинтерфейс.

# 4. Практическая часть

## 4.1. Подготовка датасета

Для реализации задачи распознавания типа общественного транспорта и его маршрутного номера был сформирован собственный датасет. Изображения получались в реальных условиях городской среды с использованием камер мобильных телефонов.

В сборе данных принимали участие несколько человек, что позволило расширить вариативность условий съёмки. Из-за использования различных моделей смартфонов в датасет попали фотографии с разными фокусными расстояниями, разрешением и цветопередачей. Это обеспечило дополнительную устойчивость нейросети к реальным условиям эксплуатации.

Съёмка проводилась в разное время суток и при различных погодных условиях, что способствовало формированию датасета, отражающего реальные сценарии использования.

Целевым ориентиром при сборе данных было достижение объёма не менее 500 качественных изображений с чётко различимыми объектами и маршрутными номерами. Изображения с плохим качеством (размытость, засветка, неполное попадание транспорта в кадр) были удалены из датасета.

**Разметка и предобработка датасета:**

В рамках данной работы использовался формат **CVAT for images 1.1**, который представляет собой набор XML-файлов и изображений, сопровождаемых информацией о разметке.

Каждое изображение размечалось вручную с применением прямоугольных ограничивающих рамок (**bounding boxes**) по двум категориям:

* **Тип транспорта**: каждому транспортному средству на изображении соответствовала рамка с категорией из следующих четырёх классов: **автобус**, **троллейбус**, **трамвай**, **маршрутка** (а также “ **unknown**”).
* **Маршрутный номер**: отдельной рамкой выделялась область, где находится маршрутный номер (обычно на передней части транспорта, в виде таблички или светодиодного дисплея). Эта аннотация использовалась на этапе OCR для извлечения текста. Пример разметки можно увидеть на рисунке 8.

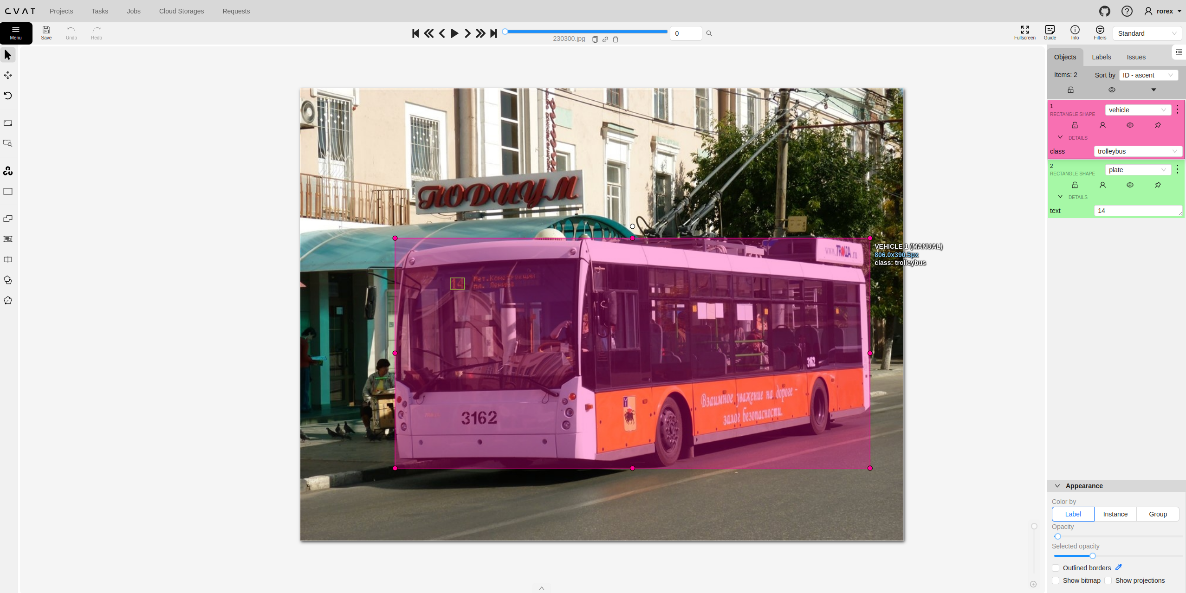


Рисунок 8 — Пример разметки данных в CVAT

CVAT позволяет организовать удобный процесс разметки: изображения подгружаются в проект, каждому объекту на изображении присваивается класс, а все аннотации экспортируются в нужном формате. CVAT for images 1.1 сохраняет аннотации в виде XML-файлов, по одному на каждое изображение. В каждом таком файле содержится информация о позициях объектов, их классах и других атрибутах:

<annotations>

<version>1.1</version>

<meta>

<task>

<labels>

<label>

<name>bus</name>

</label>

<label>

<name>route\_number</name>

</label>

</labels>

</task>

</meta>

<image id="0" name="IMG\_001.jpg" width="1920" height="1080">

<box label="bus" xtl="230.5" ytl="112.7" xbr="870.1" ybr="654.3" />

<box label="route\_number" xtl="310.2" ytl="140.6" xbr="410.8" ybr="180.2" />

</image>

</annotations>

Однако данный формат не является напрямую пригодным для использования в модели, поэтому была реализована процедура конвертации аннотаций в упрощённый формат CSV-подобного файла через небольшой python-скрипт. Он извлекает из разметки координаты маршрутного номера (в нормализованном виде), категорию транспортного средства и текст самого номера, если он указан. Аннотации преобразуются в строки вида: **<имя файла>,<ID\_класса>,<x>,<y>,<w>,<h>,<маршрутный номер>,**  где x, y, w, h — нормализованные координаты центра и размеры рамки с номером, а ID\_класса соответствует одному из типов транспорта. Также предусмотрено значение unknown для изображений без явной метки класса. Это решение позволило упростить последующую загрузку и использование аннотированных данных в модели.

# 5. Глава-комментарий

Приложение находится на последнем этапе разработки, но есть некоторые проблемы с обучением модели: видеокарта не справляется, необходимы некоторые архитектурные изменения в модели, или в гиперпараметрах, или в процессе обучения (нужно проверить экспериментально). Для скорости поиска багов все тесты и обкатку проводил на 25 изображениях и казалось, что памяти карты хватает с запасом, но уже 100 фото вызывают утечку памяти. Тем не менее, сама скорость обучения не плохая, так что обучить модель на всём датасете успеваю. На этих выходных (до 18 мая включительно) все недоработки подправлю и добавлю соответствующие главы (архитектура модели, процесс обучения, разработка приложения и т.д.), а также дополню список источников.

# 

# ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы была реализована полноценная система, включающая в себя обучение нейронной сети и разработку Android-приложения, предназначенного для распознавания вида общественного транспорта и его маршрутного номера в условиях городской среды.

В рамках теоретической части были изучены методы компьютерного зрения и свёрточных нейронных сетей, рассмотрены существующие решения в области распознавания объектов и текста, а также проведён анализ инструментов, применимых для мобильной разработки. На основании этого анализа были выбраны оптимальные технологии: PyTorch для обучения модели, Android Studio и Gradle — для создания и сборки мобильного приложения, OpenCV — для обработки изображений.

Практическая часть включала сбор и разметку собственного датасета из 500 изображений различных видов транспорта. Разметка выполнялась в CVAT с указанием типа транспорта и области маршрутного номера. Для подготовки аннотаций к обучению был разработан специализированный скрипт конвертации.

Обученная модель успешно интегрирована в Android-приложение с использованием PyTorch Mobile. В приложении реализована захват и обработка видео с камеры, отображение предсказанного типа транспорта и распознанного номера маршрута. Интерфейс адаптирован для использования слабовидящими пользователями — с крупными элементами и озвучкой результатов.

Проведённое тестирование в реальных условиях показало стабильную работу системы и удовлетворительное качество распознавания. Разработанное решение демонстрирует практическую применимость методов глубокого обучения в задачах социальной направленности и может быть использовано как прототип полноценного навигационного помощника для слабовидящих людей.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. **Twana Mustafa, Murat Karabatak. Challenges in Automatic License Plate Recognition System Review** [Электронный ресурс] : статья / IEEE Xplore. - URL: <https://ieeexplore.ieee.org/document/10131688> (дата обращения: 14.12.2024).
2. **Bengisu Dost.** How Can We Improve Microsoft's Seeing AI Application? [Электронный ресурс]: статья / Medium. - URL: <https://bengisudost.medium.com/how-can-we-improve-microsofts-seeing-ai-application-4cb4b074e0d6> (дата обращения: 14.12.2024).
3. **Timo Rohner. Python vs. C++ for Machine Learning** [Электронный ресурс]: статья / Netguru. - URL: <https://www.netguru.com/blog/cpp-vs-python> (дата обращения: 14.12.2024).
4. **Gabe Giro. Kotlin vs. Java: All-purpose Uses and Android Apps** [Электронный ресурс] : статья / Toptal. - URL: <https://www.toptal.com/kotlin/kotlin-vs-java> (дата обращения: 14.12.2024).
5. **Kurtis Pykes. PyTorch vs TensorFlow vs Keras** [Электронный ресурс]: статья / DataCamp. - URL: <https://www.datacamp.com/tutorial/pytorch-vs-tensorflow-vs-keras> (дата обращения: 14.12.2024).
6. **Getting Started with CameraX** [Электронный ресурс] : документация / Android Developers. - URL: <https://developer.android.com/codelabs/camerax-getting-started?hl=en#2> (дата обращения: 14.12.2024).
7. **Nadeem Yousuf Khanday, Shabir Ahmad Sofi. Object Recognition** [Электронный ресурс] : статья / ScienceDirect. - URL: <https://www.sciencedirect.com/topics/computer-science/object-recognition> (дата обращения: 18.12.2024).
8. Petru Potrimba. What is Optical Character Recognition (OCR)? [Электронный ресурс] : статья / Roboflow Blog. - URL: <https://blog.roboflow.com/what-is-optical-character-recognition-ocr/> (дата обращения: 18.12.2024).
9. **Howard, Gugger. Deep Learning for Coders with Fastai and PyTorch: AI Applications Without a PhD [Электронный ресурс] : книга** / Amazon. - URL: <https://www.amazon.com/Deep-Learning-Coders-fastai-PyTorch/dp/1492045527>
10. Alexander S. Gillis. What is a convolutional neural network (CNN)? [Электронный ресурс]: статья / TechTarget. - URL: <https://www.techtarget.com/searchenterpriseai/definition/convolutional-neural-network> (дата обращения: 18.12.2024).
11. **Zhang, Liptom. Dive into Deep Learning** [Электронный ресурс]: книга / Dive Into Deep Learning. - URL: <https://d2l.ai/> (дата обращения: 18.12.2024).
12. **Stevens, Antiga, Viehmann. Deep Learning with PyTorch** [Электронный ресурс]: книга/ ISIP. - URL: <https://isip.piconepress.com/courses/temple/ece_4822/resources/books/Deep-Learning-with-PyTorch.pdf> (дата обращения: 18.12.2024).