МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»
Кафедра математической кибернетики и компьютерных наук

РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ ДЛЯ РАСПОЗНАВАНИЯ ВИДОВ ГРИБОВ С ИСПОЛЬЗОВАНИЕМ ТЕХНОЛОГИЙ МАШИННОГО ОБУЧЕНИЯ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 451 группы
направления 09.03.04 — Программная инженерия
факультета КНиИТ
Демина Егора Игоревича

Научный руководитель
зав. каф. техн. пр.,
к. ф.-м. н., доцент _____ И. А. Батраева

Заведующий кафедрой
доцент, к. ф.-м. н. _____ С. В. Миронов

Саратов 2025

ВВЕДЕНИЕ

Современный мир характеризуется стремительным ростом объемов визуальной информации и развитием технологий компьютерного зрения и глубокого обучения. Эти достижения позволяют эффективно обрабатывать изображения и автоматизировать сложные задачи распознавания, а мощные мобильные устройства – интегрировать их в приложения.

Одним из практически значимых применений является идентификация природных объектов, в частности, грибов. Ошибка в распознавании вида гриба критична для безопасности человека, поэтому разработка инструмента для точного определения вида и съедобности представляет высокую ценность.

Целью данной бакалаврской работы является разработка мобильного приложения на платформе Android для визуального распознавания видов грибов на основе технологий глубокого обучения.

Для достижения поставленной цели необходимо решить следующие задачи:

- Собрать и подготовить набор изображений грибов для обучения модели;
- Выбрать архитектуру сверточной нейронной сети;
- Обучить и оценить производительность выбранной модели на подготовленном наборе данных;
- Оптимизировать обученную модель для мобильных устройств;
- Реализовать функционал распознавания модели на мобильном устройстве;
- Разработать пользовательский интерфейс мобильного приложения, позволяющий отображать результаты распознавания, включая название гриба, его съедобность и вероятность классификации;
- Предусмотреть функционал для сохранения изображений с результатами распознавания и ведения журнала выполненных классификаций.

Структура и объём работы. Бакалаврская работа состоит из введения, двух разделов, заключения, списка использованных источников и трех приложений. Общий объем работы — 53 страниц, из них 44 страницы — основное содержание, включая 4 рисунка, список использованных источников информации — 20 наименований.

1 Распознавание изображений с использованием методов глубокого обучения

1.1 Ключевые концепции компьютерного зрения

Компьютерное зрение — это область искусственного интеллекта, которая наделяет компьютеры способностью «видеть» и извлекать информацию из изображений и видео. Она автоматизирует процессы, аналогичные человеческой зрительной системе. Классический конвейер компьютерного зрения включает несколько этапов: получение изображений, их обработка, анализ и понимание сцены.

1.1.1 Получение изображений

Получение изображений (Image Acquisition) является первым и фундаментальным этапом в любой системе компьютерного зрения. На этом этапе визуальная информация из реального мира преобразуется в цифровой формат, который может быть обработан алгоритмами. Результатом является цифровое изображение, представляющее собой дискретный массив пикселей, каждый из которых содержит информацию о цвете и яркости определенной точки сцены.

Для систем компьютерного зрения качество и характеристики полученного изображения критически важны, так как они напрямую влияют на успешность последующих этапов — выделения признаков, распознавания объектов и принятия решений. Изображение выступает в роли «сырых данных», из которых система должна извлечь полезную информацию. Для повышения качества применяются методы предварительной обработки, включая нормализацию, регулировку контрастности и пр., что обеспечивает более точный и надёжный анализ.

1.1.2 Обработка изображений

После получения цифрового изображения следующим этапом в конвейере компьютерного зрения часто является его обработка (Image Processing). Обработка изображений представляет собой совокупность операций, выполняемых над изображением с целью его улучшения, изменения или подготовки для дальнейшего анализа. В отличие от высокоуровневого анализа, который стремится понять содержание изображения, обработка работает непосредственно с пиксельными данными или низкоуровневыми признаками.

Цели обработки изображений включают в себя улучшение изображения, его восстановление и преобразование.

1.1.3 Анализ изображения

После этапов получения и предварительной обработки, следующим шагом является анализ изображений. Цель этого этапа — извлечь из изображения более высокоуровневую, осмысленную информацию, выходящую за рамки простого манипулирования пиксельными значениями. Анализ направлен на выявление структур, объектов и их свойств, которые будут использоваться на последующих этапах.

В отличие от обработки изображений, которая фокусируется на модификации или улучшении пиксельных данных, анализ изображений направлен на извлечение из изображения более высокоуровневой, осмысленной информации. Цель анализа — выявление структур, объектов и их свойств для последующей интерпретации.

1.1.4 Понимание сцены

Понимание сцены (Scene Understanding) — это завершающий этап классического конвейера компьютерного зрения, на котором система не просто извлекает признаки или выделяет объекты, а формирует осмысленное представление о содержимом изображения и его контексте.

На этом этапе результаты анализа изображений (например, выделенные объекты, их границы, взаимное расположение, свойства) интерпретируются для ответа на вопросы: что изображено на сцене, где находятся объекты, как они взаимодействуют между собой, какой вывод или действие требуется предпринять на основе увиденного.

В результате этапа понимания сцены (Scene Understanding) система компьютерного зрения получает осмысленное, структурированное представление о содержимом изображения. Это не просто набор пикселей или даже выделенных объектов, а информация, которую можно использовать для принятия решений, автоматизации процессов или взаимодействия с пользователем.

1.2 Основы глубокого обучения и принципы работы искусственных нейронных сетей

Глубокое обучение (Deep Learning) — направление машинного обучения, основанное на использовании искусственных нейронных сетей (ИНС) с большим количеством слоев.

1.2.1 Искусственные нейронные сети: структура и принципы

ИНС — это вычислительная модель, вдохновленная биологическими нейронными сетями. Ее основной элемент — искусственный нейрон, который принимает взвешенные входные сигналы, суммирует их и пропускает через нелинейную функцию активации. Типичная нейронная сеть состоит из трёх видов слоёв: входного, выходного и расположенного между ними одного (или нескольких) скрытых слоев нейронов.

Входной слой выполняет только одну задачу: распределение входных сигналов (например, значения пикселей изображения) остальным нейронам. Нейроны этого слоя не производят никаких вычислений.

Скрытые слои преобразуют входные сигналы в некоторые промежуточные результаты. Они преобразуют входные данные, выявляя всё более сложные признаки.

Выходной слой преобразует данные, полученные от скрытых слоёв, в конечные результаты (например, класс объекта на изображении).

Каждое соединение между нейронами имеет свой вес, который определяет вклад соответствующего входа в итоговое значение нейрона. В процессе обучения веса корректируются для минимизации ошибки сети.

1.2.2 Обучение нейронной сети

Обучение — это процесс настройки весов сети на основе обучающей выборки. Он осуществляется с помощью метода обратного распространения ошибки и алгоритмов оптимизации, таких как градиентный спуск. На каждом шаге сеть делает прогноз, вычисляет ошибку (разницу между прогнозом и правильным ответом) и корректирует веса для минимизации этой ошибки. Для успешного обучения требуются большие объемы данных, методы регуляризации (Dropout, Batch Normalization) для предотвращения переобучения.

1.3 Сверточные нейронные сети. Архитектура и специфика для обработки изображений

Сверточные нейронные сети (CNN) — это специализированный класс глубоких сетей, архитектура которых идеально подходит для обработки двумерных данных, таких как изображения.

Ключевыми компонентами архитектуры CNN являются:

1. Сверточный слой (Convolutional Layer) выполняет операцию свертки с использованием набора обучаемых фильтров. Каждый фильтр предназначен для обнаружения определенного локального признака (ребра, текстуры, цветовые пятна). Благодаря этому слою модель может находить самые важные элементы изображения.
2. Слой активации (Activation Layer) применяет нелинейную функцию (например, ReLU) к выходам сверточного слоя, что позволяет сети моделировать сложные зависимости.
3. Пулинговый слой (Pooling Layer) уменьшает пространственную размерность карт признаков, что снижает вычислительную нагрузку.
4. Полносвязные слои (Fully Connected Layers) располагаются в конце сети. Они принимают на вход одномерный вектор признаков и выполняют финальную классификацию.

Ключевая идея CNN заключается в автоматическом построении иерархии признаков: от простых локальных на начальных слоях до сложных, семантически значимых представлений целых объектов на более глубоких слоях.

1.4 Трансферное обучение в задачах классификации изображений

Трансферное обучение (Transfer Learning) — это подход, при котором знания, полученные моделью при решении одной задачи, переносятся для решения другой, смежной задачи. В компьютерном зрении это позволяет использовать мощные, предварительно обученные на огромных датасетах (например, ImageNet) модели для новых задач, даже при ограниченном объеме данных.

Трансферное обучение значительно сокращает время и вычислительные ресурсы для обучения, позволяет достигать высокой точности при ограниченных данных и делает передовые архитектуры доступными для широкого круга разработчиков.

1.5 Технологии реализации мобильного приложения на платформе Android: Kotlin и TensorFlow Lite

Для реализации мобильных приложений с функциями машинного обучения требуется выбор подходящего технологического стека.

1.5.1 Язык программирования Kotlin

Kotlin — современный, статически типизированный язык, официально поддерживаемый Google для Android-разработки. Основные преимущества Kotlin для разработки мобильных приложений включают лаконичность (код на Kotlin компактный и читаемый), безопасность (встроенные в Kotlin механизмы помогают избежать распространенных ошибок, связанных с нулевыми ссылками) и асинхронность (язык поддерживает Coroutines, что упрощает асинхронное программирование и выполнение фоновых задач).

1.5.2 Адаптация модели для мобильных устройств с помощью TensorFlow Lite

Обученные модели глубокого обучения (.h5) имеют большой размер и требуют значительных ресурсов, что делает их непригодными для прямого использования на мобильных устройствах. TensorFlow Lite (TFLite) — это фреймворк, разработанный для решения этой проблемы.

Ключевые особенности TFLite включают в себя высокопроизводительный движок, удобный API и оптимизацию моделей (TFLite позволяет конвертировать модели TensorFlow в компактный формат, в процессе конвертации могут применяться оптимизации, что значительно уменьшает размер модели и ускоряет вычисления).

Сочетание Kotlin и TensorFlow Lite создает прочную технологическую основу для разработки производительных мобильных приложений с функциями компьютерного зрения.

2 Реализация приложения по распознаванию изображений грибов

При реализации практической части работы был использован следующий набор технологий: TensorFlow для обучения модели, TensorFlow Lite для ее развертывания на мобильном устройстве, Kotlin и Android SDK для разработки приложения, а также библиотека CameraX для работы с камерой.

2.1 Подготовка набора данных для обучения модели

2.1.1 Сбор и структурирование данных

Исходные изображения были собраны из открытых источников с упором на разнообразие условий съемки (освещение, ракурс, фон). Было сформировано 28 классов грибов, а также добавлен дополнительный класс «Other» для изображений, не содержащих целевых объектов.

2.1.2 Разделение набора данных

Собранный набор данных был разделен на три независимых подмножества в соотношении 70% / 15% / 15%: обучающая выборка для непосредственного обучения модели, валидационная выборка для мониторинга производительности в процессе обучения и предотвращения переобучения и тестовая выборка для финальной, объективной оценки качества.

2.1.3 Предварительная обработка и аугментация данных

Для подготовки данных к подаче в модель использовался ImageDataGenerator из библиотеки Keras.

Был проведён препроцессинг, т.е. все изображения приводились к единому размеру (224x224 пикселя), требуемому архитектурой ResNet50. К ним применялась специфическая функция preprocess_input, выполняющая нормализацию и перестановку цветовых каналов (RGB в BGR) в соответствии с форматом, на котором обучалась исходная модель. Также проведена аугментация — к изображениям обучающей выборки применялись случайные преобразования: горизонтальное и вертикальное отражение, повороты (до 45 градусов), изменение яркости (в диапазоне [0.6, 1.4]), масштабирование и сдвиги (до 20%). Это позволило искусственно увеличить разнообразие обучающих данных.

2.2 Конфигурирование и обучение модели распознавания

Для решения задачи классификации была выбрана архитектура CNN с применением трансферного обучения, что является оптимальным подходом при ограниченном размере датасета.

2.2.1 Выбор и конфигурирование модели

В качестве базовой модели была использована ResNet50, предобученная на датасете ImageNet. Ее исходный классификационный слой был удален. Поверх базовой модели были добавлены новые слои, адаптированные под целевую задачу:

1. GlobalAveragePooling2D: Слой для усреднения карт признаков, что значительно уменьшает количество параметров.
2. BatchNormalization: Слой для стабилизации процесса обучения.
3. Dense(256, activation='relu'): Полносвязный слой для комбинации высокоуровневых признаков.
4. Dropout(0.5): Слой регуляризации для предотвращения переобучения путем случайного «отключения» 50% нейронов на этапе обучения.
5. Dense(29, activation='softmax'): Выходной полносвязный слой с 29 нейронами (по числу классов), который преобразует выходы в вероятностное распределение.

2.2.2 Стратегия и процесс обучения

На первом этапе обучения веса базовой модели ResNet50 были «заморожены», и обучались только веса новых, добавленных слоев.

Модель была скомпилирована с использованием оптимизатора Adam с начальной скоростью обучения 1e-3 и функцией потерь CategoricalCrossentropy с техникой регуляризации label_smoothing=0.05 для улучшения обобщающей способности.

Процесс обучения контролировался с помощью набора колбэков:

— ModelCheckpoint — для сохранения лучшей версии модели;
— ReduceLROnPlateau — для автоматического снижения скорости обучения при выходе на плато производительности;
— EarlyStopping — для автоматической остановки обучения, если производительность на валидационной выборке перестает улучшаться.

2.3 Адаптация модели для мобильного развертывания

После обучения лучшая модель в формате .h5 (размер 96.7 Мб) была сконвертирована в оптимизированный формат .tflite. Полученный файл (размер 91.6 Мб) был подготовлен для интеграции в Android-приложение.

2.4 Разработка мобильного приложения на платформе Android

2.4.1 Общая архитектура и пользовательский интерфейс

Приложение построено на основе компонентов Activity (MainActivity, LogsActivit) Пользовательский интерфейс использует ConstraintLayout для гибкого размещения элементов. Ключевые элементы UI включают PreviewView для отображения видеопотока с камеры, ImageView для статических изображений, TextView для вывода результата, а также набор кнопок (MaterialButton, ImageButton), видимость которых динамически изменяется в зависимости от режима работы. Предусмотрен баннер с предупреждением о возможных ошибках распознавания.

2.4.2 Работа с источниками изображений

Приложение поддерживает два способа получения изображений: камера и галерея. Камера. Используется библиотека CameraX, которая предоставляет use-кейсы Preview (отображение видеопотока), ImageCapture (создание снимка) и ImageAnalysis (анализ кадров в реальном времени); Галерея. Используется стандартный системный Intent для выбора изображения. Результат обрабатывается через ActivityResultContracts.

2.4.3 Интеграция и выполнение модели TensorFlow Lite

Процесс распознавания на устройстве включает следующие шаги: загрузка модели (файл .tflite помещается в папку assets и загружается в память). Далее следует предобработка входных данных: изображение масштабируется до размера 224x224 пикселя. Затем его пиксельные данные преобразуются в ByteBuffer формата float32. После чего происходит выполнение модели (инференс), то есть подготовленный тензор подается в метод, который выполняет вычисления и возвращает выходной тензор. И в конце — постобработка результата: выходной массив содержит вероятности для каждого из 29 классов. Находится индекс с максимальной вероятностью, который сопоставляется со списком названий грибов для получения итогового результата.

2.4.4 Реализация Live Recognition и других функций

Были реализованы следующие функции:

- Распознавание в реальном времени: в режиме камеры приложение использует ImageAnalysis для анализа кадров из видеопотока с определенной частотой. Каждый кадр обрабатывается моделью, и результат (название и вероятность) отображается поверх превью камеры.
- Сохранение изображения с результатом: пользователь может сохранить снимок, на который программно наложен текст с результатом распознавания, в системную галерею.
- Система логирования: каждое распознавание фиксируется. Записи логов (время, результат) сохраняются локально и отображаются на отдельном экране.
- Вспомогательные компоненты: реализовано управление разрешениями, воспроизведение звука затвора, индикация прогресса и базовая обработка ошибок.

2.5 Тестирование и оценка

Оценка качества модели проводилась на отложенной тестовой выборке, которая не участвовала в обучении и валидации.

2.5.1 Результаты тестирования

На тестовой выборке модель продемонстрировала следующие результаты: точность (Accuracy) — 87.07%, потери (Loss) — 0.6455.

Эти показатели свидетельствуют о высокой обобщающей способности модели и ее способности успешно классифицировать новые, ранее невиденные изображения грибов.

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной работы была реализована и протестирована система распознавания грибов по фотографиям с применением современных методов глубокого обучения и мобильной разработки.

Модель уверенно обобщает знания на новых изображениях: итоговая точность на тестовой выборке составила 87.07%, а динамика обучения демонстрирует устойчивое снижение потерь и достижение высокой точности на валидационном наборе.

Разработанное мобильное приложение на платформе Android обеспечивает удобный пользовательский интерфейс для загрузки и анализа изображений, а также интеграцию обученной модели для распознавания грибов в реальном времени.

Полученные результаты подтверждают эффективность выбранной архитектуры и методов, а также демонстрируют перспективность применения глубокого и трансферного обучения для решения прикладных задач классификации изображений.