

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ  
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИМЕНИ И.С. ТУРГЕНЕВА»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

по направлению подготовки

09.03.04 Программная инженерия

Направленность (профиль) Промышленная разработка программного обеспечения

Студента Шалимова Александра Сергеевича                      шифр 150349/п

Институт приборостроения, автоматизации и информационных технологий

Тема выпускной квалификационной работы

«Программная система поиска рецептов на основе фотографий продуктов»

Студент

21.06.19 Шалимов А. С. Шалимов

Руководитель

Prof. 21.06.19 О. В. Конюхова

Нормоконтроль

21.06.19 А.Ю. Ужаринский

Зав. кафедрой  
программной инженерии

21.06.19 А.И. Фролов

Орёл 2019

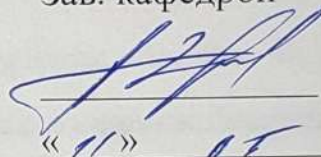
МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ  
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИМЕНИ И.С. ТУРГЕНЕВА»

Институт приборостроения, автоматизации и информационных технологий  
Кафедра программной инженерии  
Направление 09.03.04 Программная инженерия  
Направленность (профиль) Промышленная разработка программного обеспечения

УТВЕРЖДАЮ:

Зав. кафедрой



А.И. Фролов

« 11 » 11 2019 г.

ЗАДАНИЕ

на выполнение выпускной квалификационной работы

студента Шалимова Александра Сергеевича

шифр 150349/п

1 Тема ВКР «Программная система поиска рецептов на основе фотографий продуктов»

Утверждена приказом по университету от «20» ноября 2018г. № 2-3635

2 Срок сдачи студентом законченной работы «20» июня 2019г.

3 Исходные данные к работе

Теоретический материал; информация о предметной области;

4 Содержание пояснительной записки (перечень подлежащих разработке вопросов)  
Исследование предметной области системы поиска рецептов на основе фотографий продуктов;

Проектирование системы поиска рецептов на основе фотографий продуктов;

Разработка системы поиска рецептов на основе фотографий продуктов;

Реализация системы поиска рецептов на основе фотографий продуктов.



## 5 Перечень демонстрационного материала

Презентация, отображающая основные этапы и результаты выполнения ВКР

Дата выдачи задания

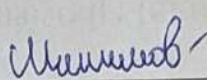
« 11 » мая 2019 г.

Руководитель

  
(подпись)

О. В. Конюхова

Задание принял к исполнению

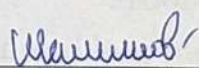
  
(подпись)

А. С. Шалимов

### КАЛЕНДАРНЫЙ ПЛАН

Наименование этапов работы	Сроки выполнения этапов работы	Примечание
Исследование предметной области системы поиска рецептов на основе фотографий продуктов	11.05.19 – 15.05.19	
Проектирование системы поиска рецептов на основе фотографий продуктов	16.05.19 – 21.05.19	
Разработка системы поиска рецептов на основе фотографий продуктов	22.05.19 – 30.05.19	
Реализация системы поиска рецептов на основе фотографий продуктов	31.05.19 – 9.06.19	
Оформление ВКР	10.06.19 – 20.06.19	

Студент

  
(подпись)

А. С. Шалимов

Руководитель

  
(подпись)

О. В. Конюхова

## АННОТАЦИЯ

ВКР 89 с., 42 рис., 1 табл., 10 источников, 2 прил.

МАШИННОЕ ОБУЧЕНИЕ, КОМПЬЮТЕРНОЕ ЗРЕНИЕ, СИСТЕМА ДЕТЕКТИРОВАНИЯ ПРОДУКТОВ ПИТАНИЯ, ДООБУЧЕНИЕ МОДЕЛЕЙ РАСПОЗНАВАНИЯ ОБЪЕКТОВ, ANDROID-ПРИЛОЖЕНИЕ, ПОДБОР РЕЦЕПТОВ.

Выпускная квалификационная работа на тему «Программная система поиска рецептов на основе фотографий продуктов».

В первой главе ВКР проводится анализ методов распознавания объектов на изображении, рассматриваются имеющиеся аналоги на рынке мобильных приложений, выявляются функциональные и нефункциональные требования к разрабатываемой программе.

Во второй главе происходит проектирование системы поиска рецептов на основе фотографий продуктов. Разрабатывается функционал системы.

В третьей главе описан процесс разработки мобильного приложения и процесс дообучения предобученной модели распознавания объектов для решения задачи распознавания продуктов питания.

В четвертой главе описывается процесс реализации мобильного приложения подбора рецептов по списку ингредиентов, процесс подготовки изображений продуктов и процесс дообучения модели распознавания объектов для задачи нахождения множества продуктов питания на изображении, процесс интеграции дообученной модели в мобильное приложение. Приводятся результаты ошибок классификации и локализации модели распознавания продуктов питания, а также пользовательский интерфейс системы поиска рецептов на основе фотографий продуктов.

Графическая часть выпускной квалификационной работы содержит иллюстрации, таблицы, которые объединены в презентации PowerPoint.

Библиографическая часть выпускной квалификационной работы включает в себя 10 источников.

## СОДЕРЖАНИЕ

ВВЕДЕНИЕ	6
1 ИССЛЕДОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ СИСТЕМЫ ПОИСКА РЕЦЕПТОВ НА ОСНОВЕ ФОТОГРАФИЙ ПРОДУКТОВ	8
1.1 Описание предметной области	8
1.2 Анализ существующих методов решения задачи распознавания объектов на сцене	10
1.3 Функциональные и нефункциональные требования системы поиска рецептов на основе фотографий продуктов	18
1.4 Обзор аналогов	18
2 ПРОЕКТИРОВАНИЕ СИСТЕМЫ ПОИСКА РЕЦЕПТОВ НА ОСНОВЕ ФОТОГРАФИЙ ПРОДУКТОВ	25
2.1 Архитектура системы поиска рецептов на основе фотографий продуктов	25
2.2 Концептуальная модель системы поиска рецептов на основе фотографий продуктов	26
2.3 Поведение системы поиска рецептов на основе фотографий продуктов	30
2.4 Проектирование базы данных	32
3 РАЗРАБОТКА СИСТЕМЫ ПОИСКА РЕЦЕПТОВ НА ОСНОВЕ ФОТОГРАФИЙ ПРОДУКТОВ	34
3.1 Проектирование структуры ПО	34
3.2 Проектирование алгоритмов приложения	34
3.3 Проектирование дообучения модели	39
4 РЕАЛИЗАЦИЯ СИСТЕМЫ ПОИСКА РЕЦЕПТОВ НА ОСНОВЕ ФОТОГРАФИЙ ПРОДУКТОВ	43
4.1 Выбор мобильной платформы	43
4.2 Выбор среды разработки	43
4.3 Выбор модели нейронной сети	44
4.4 Подготовка данных для обучения	45

4.5 Дообучение модели	49
4.6 Импорт дообученной модели в Android Studio	51
4.7 Демонстрация интерфейса реализованного приложения	52
ЗАКЛЮЧЕНИЕ	58
СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ	59
ПРИЛОЖЕНИЕ А – ЛИСТИНГ ПРОГРАММЫ	61
ПРИЛОЖЕНИЕ Б – ЛИСТИНГ ФАЙЛОВ ДЛЯ ДООБУЧЕНИЯ МОДЕЛИ	77
УДОСТОВЕРЯЮЩИЙ ЛИСТ № 150349 НА ДЕМОНСТРАЦИОННЫЙ МАТЕРИАЛ	86
ИНФОРМАЦИОННО-ПОИСКОВАЯ ХАРАКТЕРИСТИКА ДОКУМЕНТА НА ЭЛЕКТРОННОМ НОСИТЕЛЕ	87

## ВВЕДЕНИЕ

Области машинного обучения, связанные с распознаванием изображений, сильно развились за последние десять лет. В настоящее время проблема классификации объекта на изображении является хорошо изученной. Разработчики предложили множество архитектур свёрточных нейронных сетей, решающих эту проблему.

Наиболее актуальными задачами компьютерного зрения, изучаемыми сейчас являются:

- определение границ;
- сегментация;
- детектирование.

Проблема современных архитектур, решающих вышеописанные задачи в их масштабах. Обладая большой точностью и относительно быстрой скоростью работы, данные модели невозможно использовать на устройствах, обладающими слабой производительностью и маленькими объемами оперативной памяти.

Мобильные устройства подходят для решения большого количества прикладных задач компьютерного зрения. Они позволяют пользователю быстро получить изображение с помощью встроенной камеры, при этом они обладают хорошей автономностью.

Аппаратная составляющая смартфонов отстаёт от персональных компьютеров, поэтому основной упор в выборе модели идёт на скорость работы и объём, занимаемый моделью в памяти.

Для пользователя приложения по поиску рецептов, использующего распознавание изображения с камеры устройства, основным требованием будет возможность распознавания нескольких объектов на одной фотографии. Это позволит сократить время на поиск рецептов, так как не будет необходимости делать несколько фотографий отдельных ингредиентов, для их дальнейшей классификации.

Целью выпускной квалификационной работы является снижение временных затрат на подбор рецептов по фотографии за счёт распознавания множества объектов на сцене.

Для достижения поставленной цели необходимо решить следующие задачи:

1. выполнить анализ задач распознавания объекта и выбрать модель, распознающую несколько объектов на сцене;
2. подготовить изображения продуктов и на их основе дообучить модель распознавания объектов;
3. выполнить преобразование полученной модели распознавания объектов для обеспечения возможности её работы на мобильных устройствах;
4. спроектировать и реализовать приложение, использующее преобразованную модель, для автоматизированного поиска рецептов по фотографии продуктов.



# 1 ИССЛЕДОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ СИСТЕМЫ ПОИСКА РЕЦЕПТОВ НА ОСНОВЕ ФОТОГРАФИЙ ПРОДУКТОВ

## 1.1 Описание предметной области

Идея автоматизированного поиска объектов и их классификации на изображении является актуальной на протяжении нескольких десятков лет. В последнее время наблюдается тенденция использования алгоритмов глубокого машинного обучения. Это обусловлено несколькими факторами: появление больших вычислительных мощностей, накопление огромного набора данных, находящегося в свободном доступе (изображения, видео).

Все современные смартфоны оснащены камерой, с помощью которой можно делать снимки. Поэтому, разработка мобильного приложения является наиболее актуальной. При этом, в отличие от стационарного компьютера, мобильное устройство сильнее ограничено в вычислительных ресурсах и памяти, требуемой для работы приложения. Данная особенность мобильных устройств, создаёт ограничения по выбору моделей машинного обучения, применяемых для разработки требуемого приложения.

Продукты питания представляют собой множество ингредиентов. Большое количество кулинарных сайтов содержит фотографии необходимых продуктов до приготовления. Зачастую это одна фотография со всеми ингредиентами.

При решении задачи классификации изображения, ответом является метка класса, которому принадлежит изображение. Данный метод можно использовать для фотографии одного продукта. Для случая, когда на изображении находится несколько объектов, задача преобразуется в двойную: поместить объект в прямоугольную рамку (локализовать), определить – какому классу принадлежит найденный объект.

Классификация баклажана на изображении представлена на рисунке 1.1.

Детектирование продуктов питания представлено на рисунке 1.2.



**На изображении есть баклажан?**



**Да**

Рисунок 1.1 – Классификация продукта на изображении



Рисунок 1.2 – Локализация и классификация продуктов на изображении

Вторая задача представляет большой интерес, так как нет необходимости применять алгоритм распознавания к нескольким изображениям по отдельности.

С использованием современных «глубоких» нейронных сетей, ориентированных на скорость работы, становится возможным распознавать объекты в реальном времени. При этом суммарная ошибка, состоящая из ошибки локализации и ошибки классификации довольно мала.

## **1.2 Анализ существующих методов решения задачи распознавания объектов на сцене**

Для выделения прямоугольных областей используют два класса методов:

- нейронные сети, выделяющие прямоугольные области, каждая из которых подаётся на вход готовому классификатору (R-CNN, Fast-R-CNN, Faster-R-CNN);
- алгоритмы, позволяющие выделить прямоугольные области за один «просмотр» изображения (Single Shot Detector, YOLO).

Первая группа обладает большей точностью распознавания, при этом время на анализ изображения составляет до 1 секунды. Так как для каждой границы на изображении, происходит один проход через всю модель. Для мобильных устройств, ограниченных в ресурсах, такой подход не подходит.

Наиболее популярным представителем второй группы является Single Shot MultiBox Detector, он считается одним из наиболее быстрых, и позволяет локализовать объекты в реальном времени (60-кадров в секунду, для входного изображения 300×300 пикселей).

Структура SSD представлена на рисунке 1.3.



Рисунок 1.3 – Структура SSD object detector

Выделителем признаков является свёрточная нейронная сеть. В процессе применения операций свёрток к изображению можно извлекать различные признаки. В начале это могут быть грани, присущие конкретному объекту. Далее признаки становятся больше, сеть может выявлять различные детали и компоненты. Последняя операция выявляет самые большие особенности объекта (очертания лица, форму продуктов).

Для выявления объектов разного масштаба в SSD используют 6 сетей различного размера:  $19 \times 19$ ,  $10 \times 10$ ,  $5 \times 5$ ,  $3 \times 3$ ,  $2 \times 2$ ,  $1 \times 1$ . Чем больше размер сети, тем меньше объекты будет искать алгоритм. Для сети  $5 \times 5$  получается 25 областей. Задача алгоритма – определить внутри какой области находится объект, скорректировать результат и определить класс объекта. Пример представлен на рисунке 1.4.



Рисунок 1.4 – Сеть размером  $5 \times 5$ , наложенная на изображение

Так как положение объектов на сцене разное, в каждой из областей заранее определяют набор прямоугольников. На рисунке 1.5 представлены стандартные прямоугольники для области с координатой (3,3).

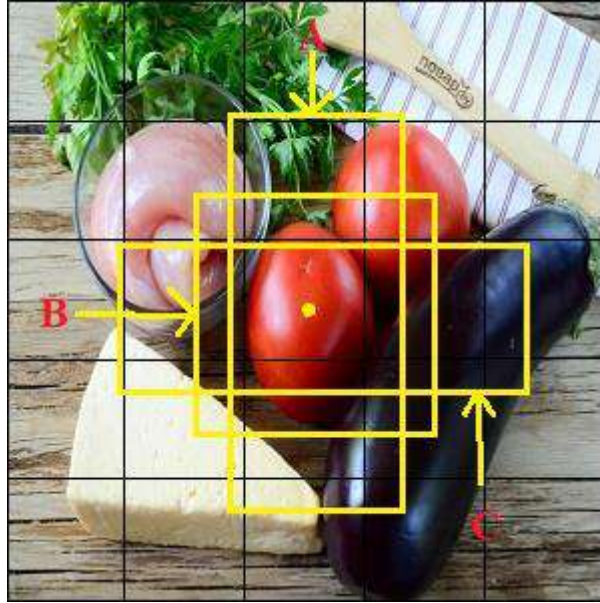


Рисунок 1.5 – Стандартные прямоугольники для области с координатой (3,3)

У каждого прямоугольника есть 4 параметра: координата центра по  $x$ , координата центра по  $y$ , ширина прямоугольника, высота прямоугольника.

Пусть  $s_{min}$  и  $s_{max}$  – минимальный и максимальный масштаб, который задается константами,  $m$  – количество сетей, равное 6. Сторона прямоугольника находится по формуле (1):

$$s_k = s_{min} + \frac{s_{max} - s_{min}}{m - 1} * (k - 1), \quad k \in [1, m] \quad (1)$$

Для стороны  $s_k$  существуют 5 коэффициентов сжатия  $a_r$ , необходимых для определения высоты и ширины прямоугольника.  $a_r \in \{1, 2, \frac{1}{2}, 3, \frac{1}{3}\}$ .

Ширина находится по формуле (2):

$$w_k^a = s_k * \sqrt{a_r} \quad (2)$$

Высота находится по формуле (3):

$$h_k^a = \frac{s_k}{\sqrt{a_r}} \quad (3)$$



А также создаётся один прямоугольник с одинаковой высотой и шириной, формула (4):

$$w_k = h_k = \sqrt{s_k * s_{k+1}} \quad (4)$$

На рисунке 1.5 используются коэффициенты сжатия  $a_r = 2$  и  $\frac{1}{2}$ , и один квадрат. Таким образом всего прямоугольников на изображении  $5 \times 5 \times 3 = 75$ . Для каждого прямоугольника определяется корректировка по 4 параметрам. Для данного прямоугольника определяется вероятность принадлежности объекта к одному из возможных классов.

Для SSD используется понятие фонового класса (background), когда ни один из объектов не был отнесен к заданным классам, он относится к background классу.

Далее отбрасываются те прямоугольники, у которых вероятность принадлежности к классам кроме фонового, ниже порогового значения. В результате получают 6 ответов для одного изображения. Применяя алгоритм Fast NMS, находящего лучший из полученных границ объекта, на выходе имеем границы прямоугольника, обрамляющего объект и вероятность принадлежности данного объекта к заданным классам.

Функция ошибки состоит из двух частей:

Ошибка локализации – несоответствие между правильными границами объекта и предсказанными. SSD использует только положительные распознавания ( $\text{IoU} > 0.5$ ).

$\text{IoU}$  – пересечение над объединением (intersection over the union). Соотношение между пересеченной областью и объединением областей (предсказанной и реальной). На рисунке 1.6 представлено пересечение областей.

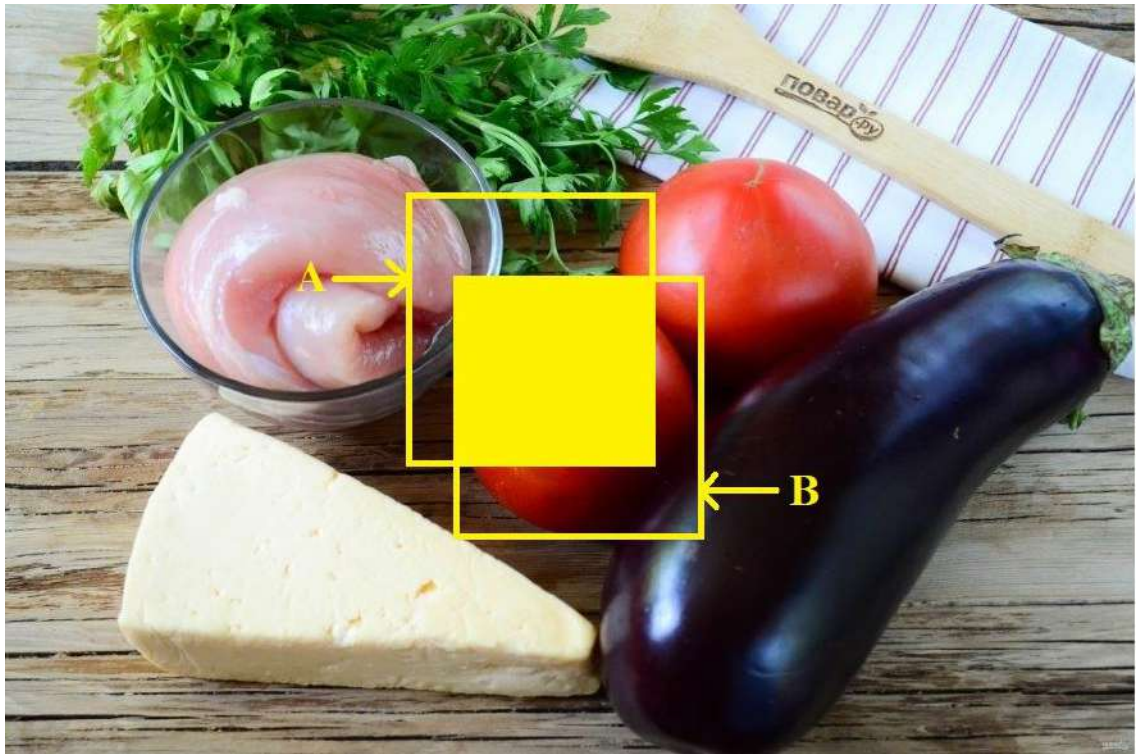


Рисунок 1.6 – Пересечение предсказанной области А с реальной областью В

На рисунке 1.7 представлено объединение областей.

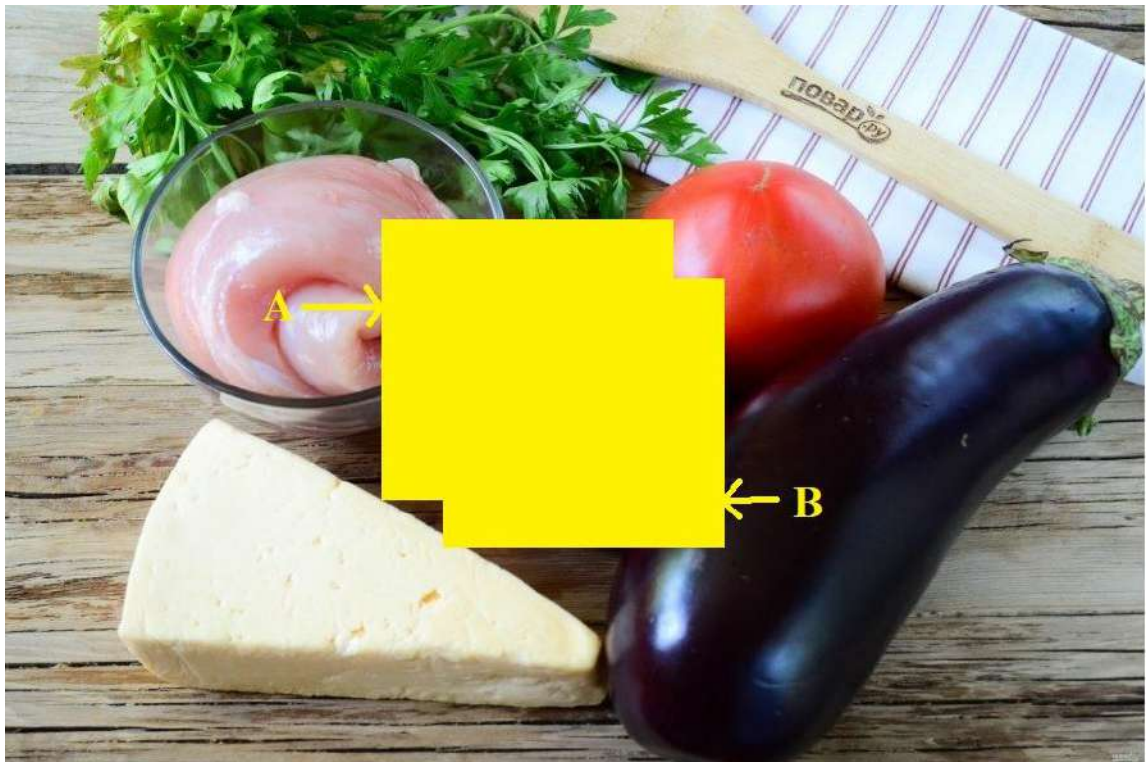


Рисунок 1.7 – Объединение предсказанной области А с реальной областью В

Функция ошибок локализации представлена на формуле (5):

$$L_{loc}(x, l, y) = \sum_{i \in Pos} \sum_{m \in (cx, cy, w, h)} x_{ij}^k * smooth_{L1}(l_i^m - \hat{y}_j^m), \quad (5)$$

где  $\hat{y}_j^{cx} = \frac{y_j^{cx} - d_i^{cx}}{d_i^w}$  – смещение положения центра по x;

$\hat{y}_j^{cy} = \frac{y_j^{cy} - d_i^{cy}}{d_i^h}$  – смещение положения центра по y;

$\hat{y}_j^w = \log\left(\frac{y_j^w}{d_i^w}\right)$  – новая ширина прямоугольника;

$\hat{y}_j^h = \log\left(\frac{y_j^h}{d_i^h}\right)$  – новая высота прямоугольника;

$l_i$  – 4 базовых параметра, предсказанных моделью;

$x_{ij}^k = \begin{cases} 1 & IoU > 0.5 \\ 0 & \text{иначе} \end{cases}$ ;

$smooth_{L1}(x) = \begin{cases} 0.5 * x^2 & \text{если } |x| < 1 \\ |x| - 0.5 & \text{иначе} \end{cases}$ .

Второй составляющей является ошибка классификации. Она представлена на формуле (6):

$$L_{conf}(x, c) = - \sum_{i \in Pos} x_{ij}^k * \log(\hat{c}_i^k) - \sum_{i \in Neg} \log(\hat{c}_i^0), \quad (6)$$

$\hat{c}_i^k = \frac{\exp(c_i^k)}{\sum_p \exp(c_i^k)}$  – softmax ошибка для много классовых коэффициентов.

Результирующая ошибка, представленная на формуле (7):

$$L(x, c, l, y) = \frac{1}{N} \left( L_{conf}(x, c) + \alpha * L_{loc}(x, l, y) \right), \quad (7)$$

где N – количество положительных ответов;

$\alpha$  – вес для ошибки локализации.

Для выделения признаков подходит любая быстрая свёрточная нейросеть. Для решения поставленной задачи необходимо хранить модель в памяти смартфона. Для таких задач используется нейронная сеть для мобильных – mobilenet v1. Основные операции, которые используются в

данной архитектуре: обычная свёртка, разделяемая по глубине свёртка (depthwise separable convolutions).

Разделяемая по глубине свёртка является комбинацией двух операций: свёртка по глубине (depthwise convolution) и точечная свёртка (pointwise convolution). Две данные операции выполняют почти ту же работу, что и обычная свёртка, но за гораздо меньшее число вычислительных действий. Поэтому данную сеть можно использовать на мобильных устройствах.

Учитывая большое число границ, генерируемых во время прямого прохода SSD, важно обрезать большую часть блоков, применяя метод Non-Maximum Suppression: блоки с коэффициентом ниже порога  $k$  и IoU меньше порога  $l$  отбрасываются, только  $N$  лучших границ остаётся. Таким образом, только наиболее вероятные прогнозы сохраняются сетью, более шумные удаляются.

Структура архитектуры SSD mobilenet представлена на рисунке 1.8.

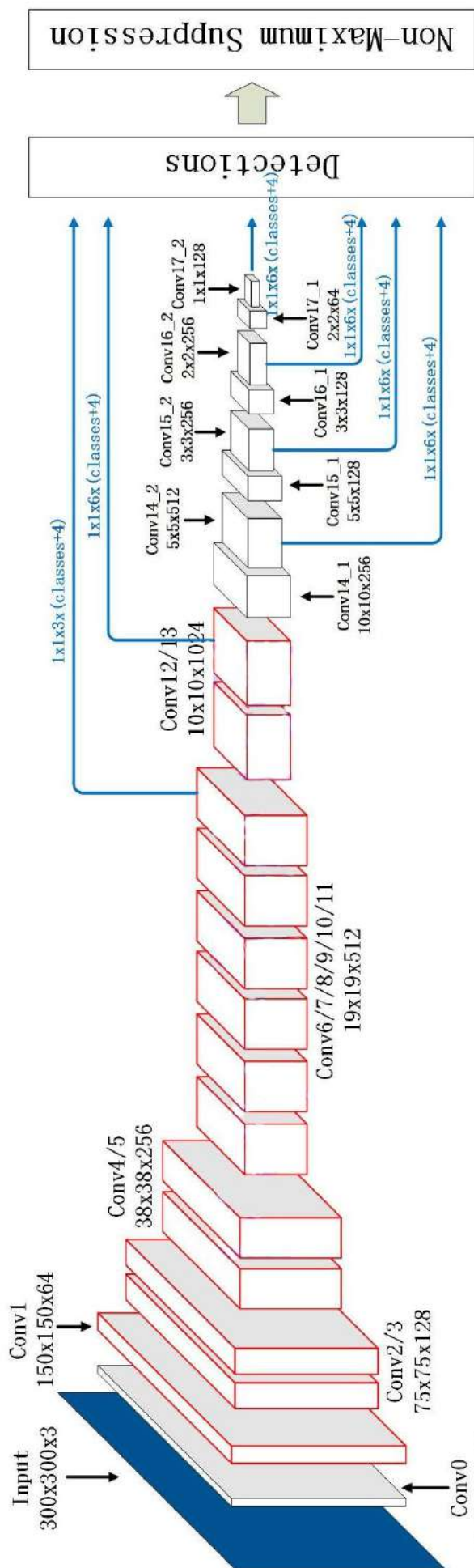


Рисунок 1.8 – Архитектура mobilenet ssd



### **1.3 Функциональные и нефункциональные требования системы поиска рецептов на основе фотографий продуктов**

Разрабатываемая система должна обеспечивать быстрое и точное распознавание продуктов на фото, а также предоставлять удобный интерфейс для поиска рецептов с заданными ингредиентами.

Для разрабатываемой системы были сформулированы следующие функциональные требования:

- распознавание множества объектов на изображении;
- поиск рецептов по заданному множеству ингредиентов;
- возможность вручную добавить новые продукты для поиска;
- возможность убрать ненужные продукты из поискового запроса;
- у найденных рецептов можно просматривать необходимые ингредиенты и инструкцию по приготовлению.

Система поиска рецептов по фотографии продуктов должна обладать рядом нефункциональных требований:

- наличие камеры;
- платформа Android с версией sdk не ниже 21;
- наличие около 50 мб памяти на устройстве, для хранения обученной модели распознавания объектов и базы данных рецептов.

### **1.4 Обзор аналогов**

В области мобильных приложений существует много приложений для кулинарии, а также для подсчёта калорий. Каждое из приложений обладает базой данных с рецептами еды, предлагает добавить новый рецепт или сохранить рецепт в «понравившееся».

Рынок приложений, решающих задачи распознавания еды по фотографии очень мал. В основном приложения, реализованные на данный момент, распознают один объект на сцене.

В ходе анализа были выбраны те продукты, которые наиболее близки к решаемой задаче:

- CalorieMama AI;
- Foodies;
- Простые рецепты.

Приложение CalorieMama AI является мобильным счетчиком калорий. Используемая платформа – Android. Данное приложение позволяет распознавать объект на изображении. При этом отсутствует возможность распознать несколько объектов одновременно.

CalorieMama AI обладает богатым функционалом, а также удобным интерфейсом. С его помощью можно добавлять продукты в свой рацион питания, следить за калориями конкретного продукта, следить за расходом калорий в течении дня.

На рисунках 1.9-1.10 приведены примеры работы с приложением CalorieMama AI.

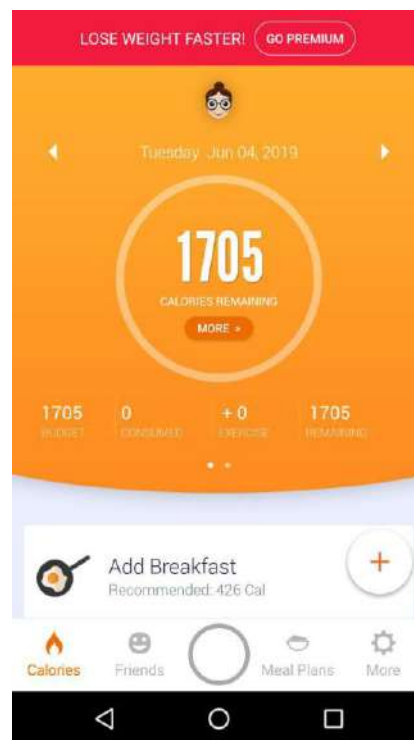


Рисунок 1.9 – Пример основного меню с счетчиком калорий в приложении CalorieMama AI

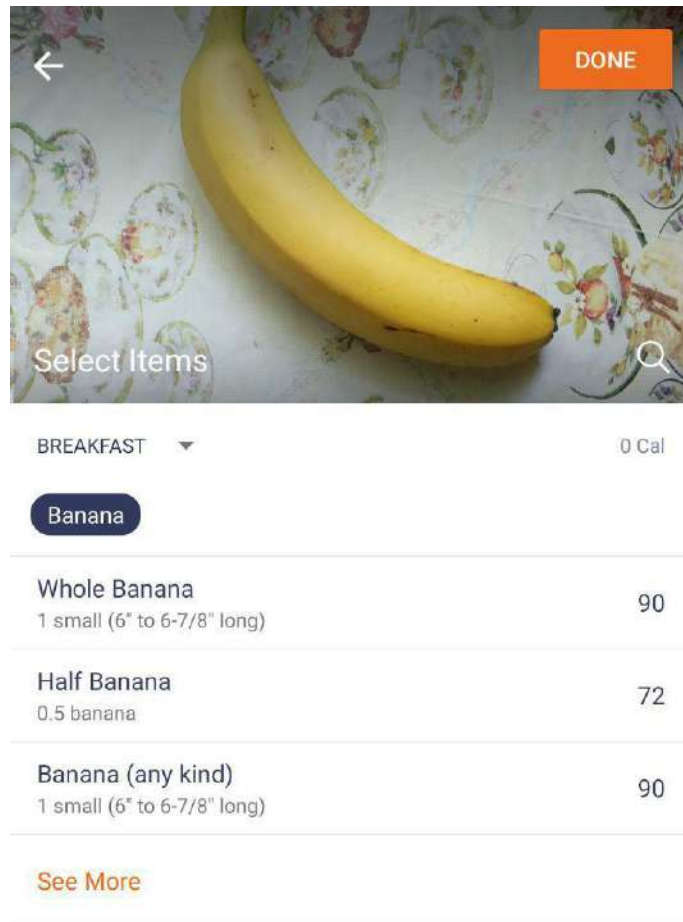


Рисунок 1.10 – Пример отображения результата распознавания банана по фотографии в приложении CalorieMama AI

Foodies – мобильное приложение для Android, позволяющее подобрать рецепты по набору ингредиентов. Существует возможность поиска по нескольким ингредиентам сразу.

Для поиска рецептов необходимо подключение к сети Интернет. При просмотре конкретного рецепта доступно изображение готового рецепта и список необходимых ингредиентов. Для просмотра шагов приготовления необходим переход на сайт приложения.

На рисунках 1.11-1.12 приведены примеры работы с приложением Foodies.

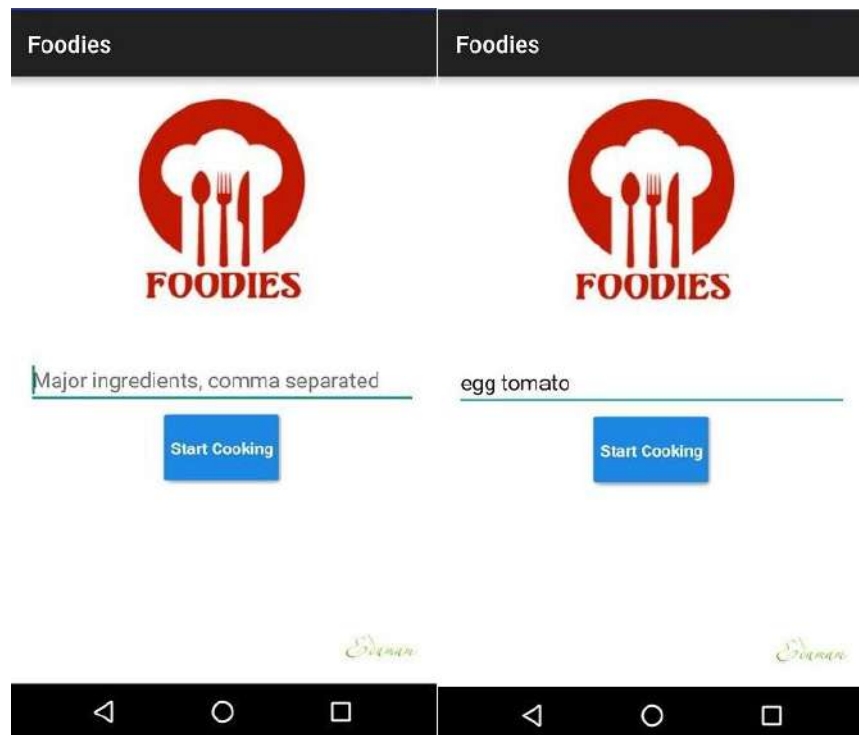


Рисунок 1.11 – Пример стартового экрана и множественного запроса в приложении Foodies

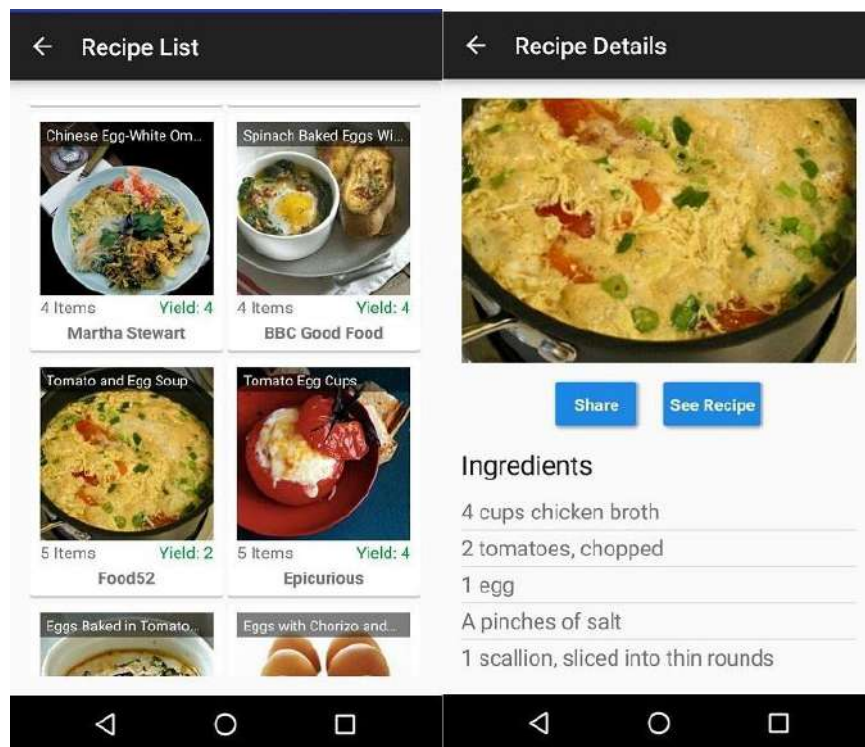


Рисунок 1.12 – Пример отображения рецептов и детального отображения рецепта в приложении Foodies

Приложение «Простые рецепты» является сервисом для поиска различных рецептов. Оно обладает рядом особенностей:

- поиск рецептов по одному ингредиенту;
- возможность добавления рецепта в избранное;
- возможность добавлять отзывы к рецептам;
- возможность ставить оценку рецептам;
- отслеживание процесса приготовления рецепта;
- наличие корзины продуктов.

На рисунках 1.13-1.15 приведены примеры работы с приложением «Простые рецепты».

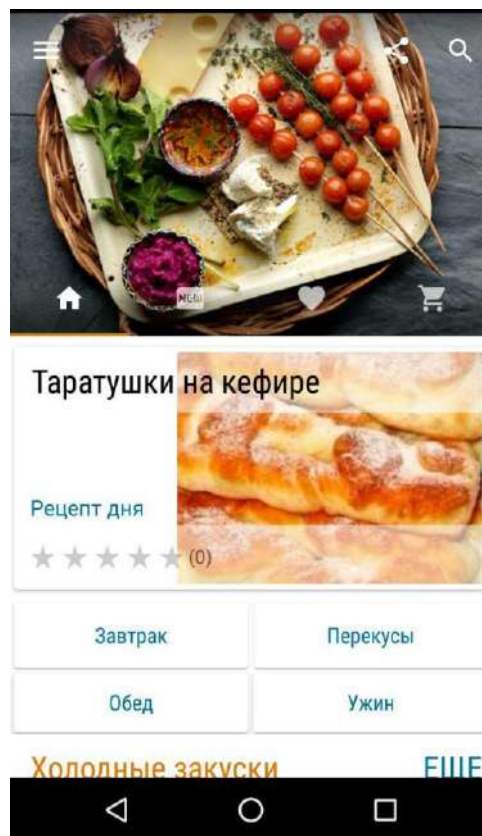


Рисунок 1.13 – Пример запуска приложения «Простые рецепты»



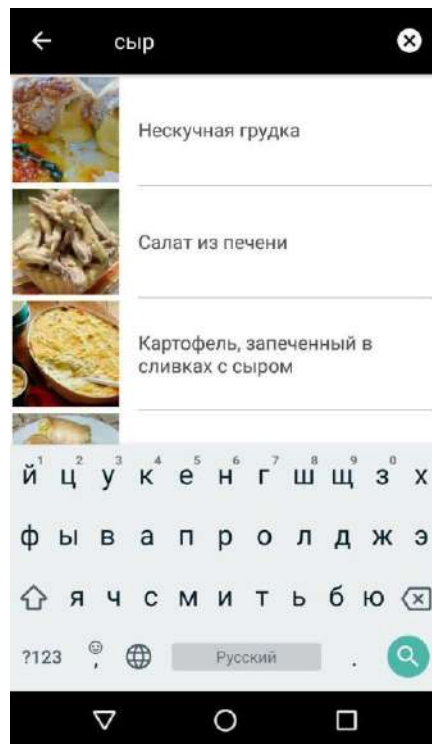


Рисунок 1.14 – Пример отображения поиска по ингредиенту в приложении «Простые рецепты»

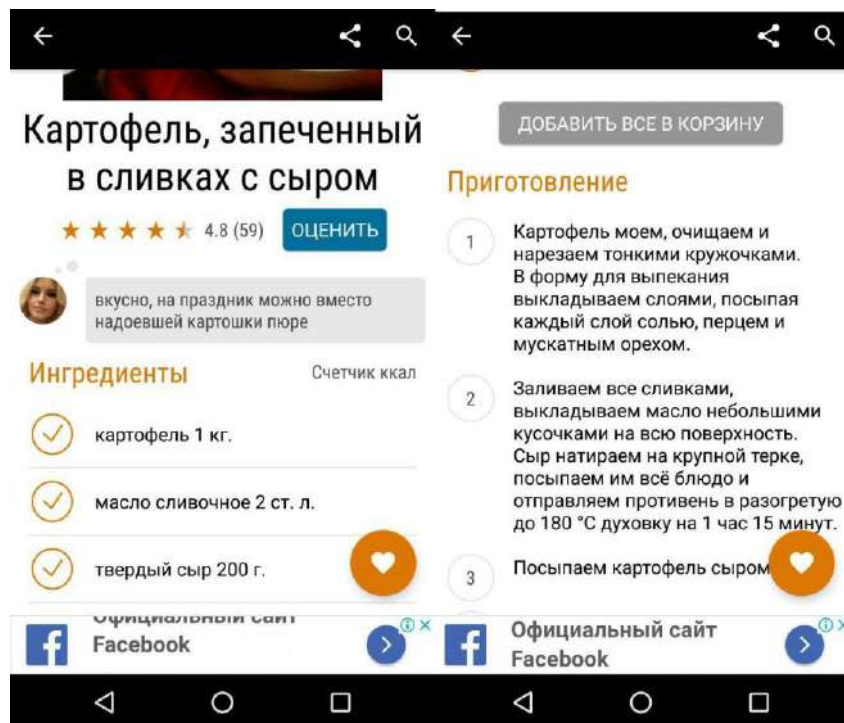


Рисунок 1.15 – Пример отображения рецепта в приложении «Простые рецепты»

Исходя из рассмотренных приложений можно сделать вывод об отсутствии на рынке мобильных приложений с возможностью распознавать ингредиенты по одной фотографии и подбирать рецепты по ним.

Сравнения особенностей рассмотренных аналогов представлен в таблице 1.1.

Таблица 1.1 – Особенности приложений аналогов.

Приложения Критерии	Foodies	Простые рецепты	CalorieMama AI
Требование подключения к сети интернет	+	+	+
Возможность поиска рецептов по нескольким ингредиентам	+	-	-
Наличие пошаговых рецептов	-	+	-
Распознавание продуктов по фотографии	-	-	+/-
Удобный интерфейс	-	+	+

## 2 ПРОЕКТИРОВАНИЕ СИСТЕМЫ ПОИСКА РЕЦЕПТОВ НА ОСНОВЕ ФОТОГРАФИЙ ПРОДУКТОВ

### 2.1 Архитектура системы поиска рецептов на основе фотографий продуктов

Реализуемая система будет хранить в памяти устройства обученную нейронную сеть, для хранения рецептов будет использоваться локальная база данных. Так как все данные уже имеются на устройстве, нет необходимости использовать подключение к сети интернет.

Архитектура системы поиска рецептов на основе фотографий продуктов – модульная. Основными компонентами такой системы являются отдельные модули.

При использовании данной модели упростится обновление приложения, при этом будет возможность реализовывать и тестировать модули не зависимо друг от друга.

Общая схема модулей системы представлена на рисунке 2.1.



Рисунок 2.1 – Модули разрабатываемой системы

## 2.2 Концептуальная модель системы поиска рецептов на основе фотографий продуктов

Для наглядного определения функций программного обеспечения используются UML диаграммы. Они позволяют просто и информативно показать функциональные спецификации приложения.

Для начала построим диаграмму вариантов использования приложения. Это позволит выделить основные действия, которые может выполнять пользователь. Они представлены на рисунке 2.2.

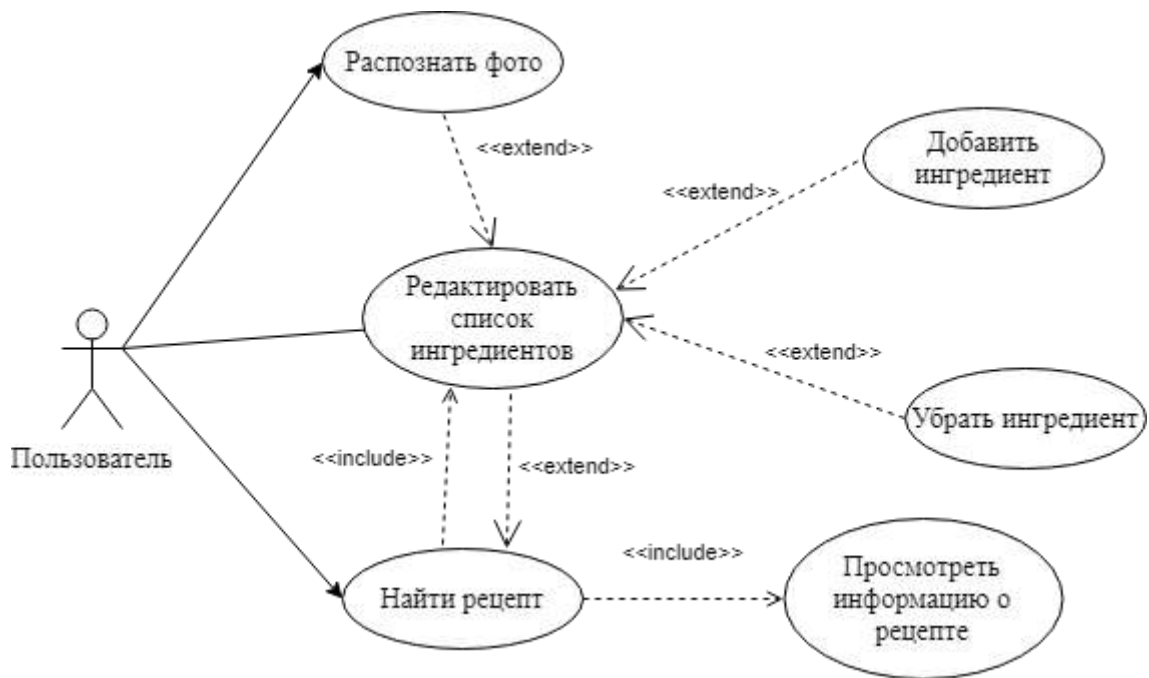


Рисунок 2.2 – Возможные действия пользователя

Диаграмма вариантов является самым общим примером функционирования системы, который в дальнейшем усложняется с использованием других диаграмм.

На данной диаграмме пользователь приложения обозначен как Пользователь. Основными функциями являются: распознать фото – выявление ингредиентов на изображении, установить фильтр – изменение количества ингредиентов, найти рецепт – поиск рецептов используя заданный фильтр.

Основная функция распознать фото расширяет вариант использования установить фильтр и устанавливает его автоматически. Функция установить

фильтр включает в себя два варианта: добавить ингредиент и убрать ингредиент.

Вариант использования найти рецепт включает в себя вариант просмотра информации о рецепте.

Функция «Распознать фото» является главной. Для её подробного проектирования построим диаграмму деятельности, представленную на рисунке 2.3.

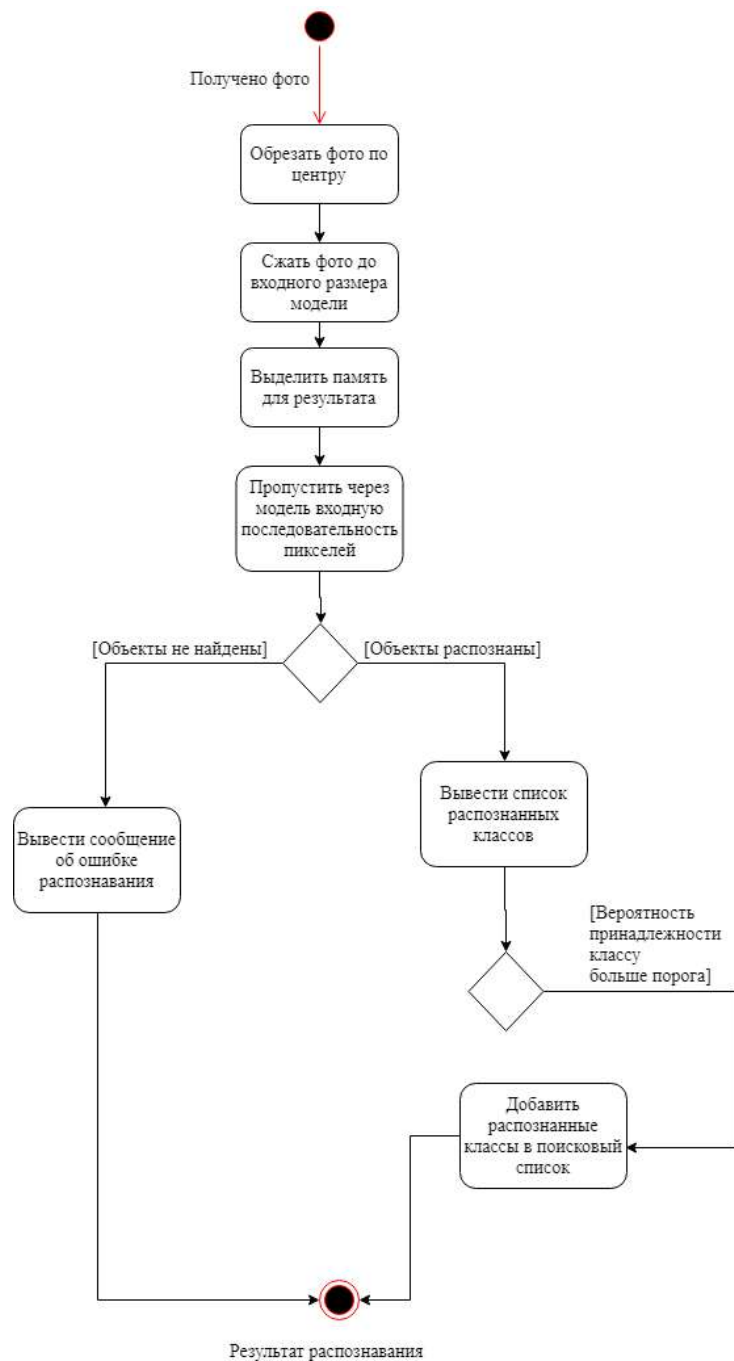


Рисунок 2.3 – Детальное описание функции распознавания по фото



Стартом диаграммы деятельности является момент, когда пользователь делает фотографию. Затем, в полученном фото необходимо сохранить пропорции 1:1. Так как большинство используемых моделей используют именно такое соотношение сторон изображения. Это поможет избежать попадания лишних фрагментов на входное изображение при сжатии.

После, необходимо изменить размер изображения до необходимого на входе модели нейронной сети.

Затем, необходимо подготовить массивы для возвращаемого моделью результата. Это 4 массива: класс, вероятность принадлежности, количество распознанных объектов, границы обрамляющего прямоугольника.

Далее выполняется запуск модели для распознавания множества объектов на сцене. В случае, если не один из объектов не был распознан, на экран выводится оповещение. При удачном распознавании, если вероятность классификации больше порогового значения, происходит добавление ингредиента в поисковый список.

Основными элементами готовой системы будут классы. Для наглядного отображения изобразим основные классы приложения на диаграмме классов, представленной на рисунке 2.4.

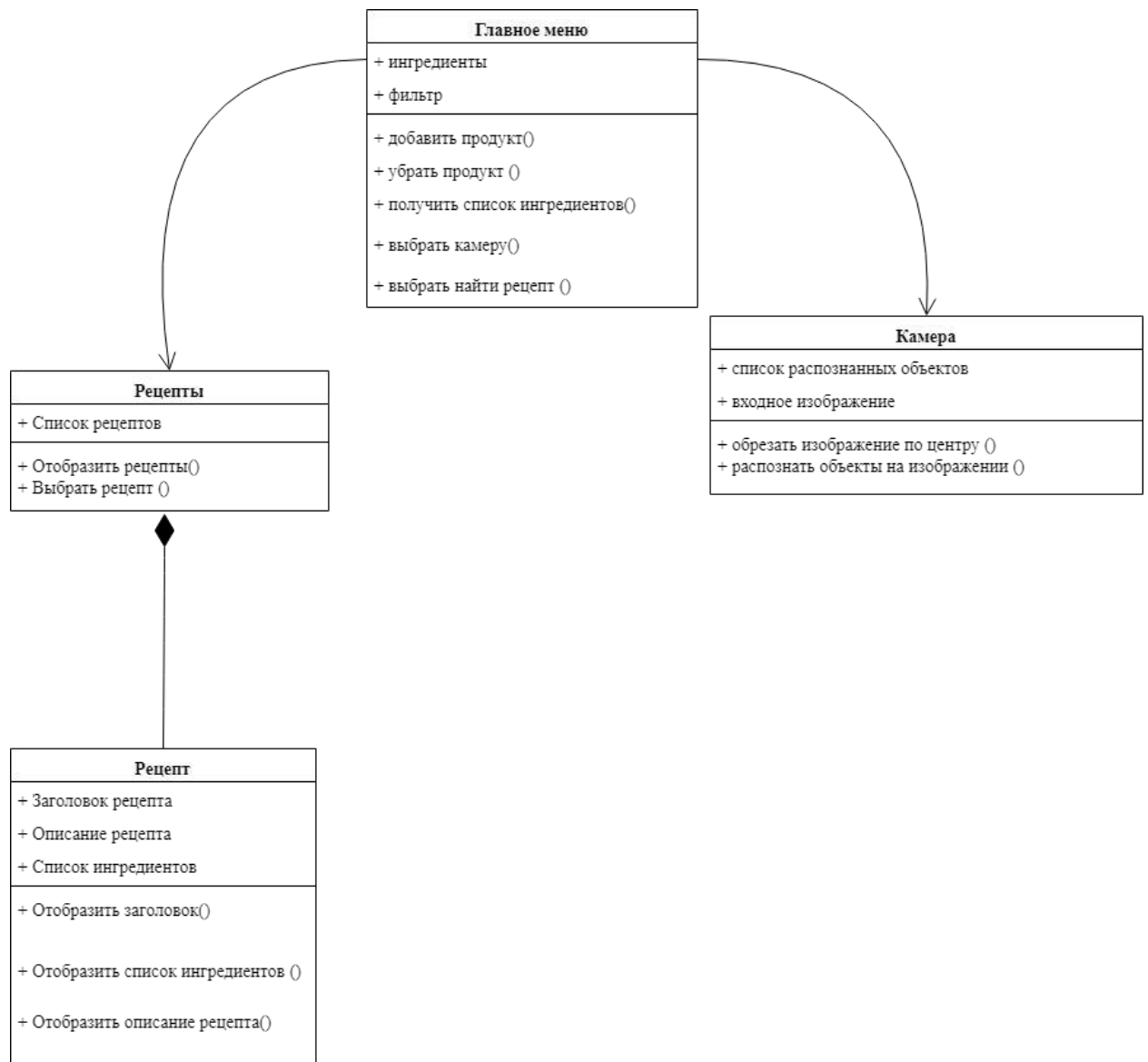


Рисунок 2.4 – Основные классы приложения подбора рецептов по фотографии

Было выделено 4 основных класса приложения. Для класса Главное меню было выделено поле ингредиенты, отвечающее за изначальный список продуктов, для которых можно найти рецепты, а также поле фильтр, которое является списком, поступающим на вход базе данных, реализованной в дальнейшем. В класс Главное меню включены методы добавления и удаления продуктов из фильтра, метод получения списка ингредиентов, заполняющий поле ингредиенты, метод выбора камеры, метод найти рецепт.

Для класса Камера были выделены два поля: список распознанных объектов, который передаётся в Главное меню и поле входное изображение, служащее входом для нейронной сети.

Класс Рецепт хранит в себе информацию об одном рецепте.

Класс Рецепты содержит поле список рецептов и два метода: отобразить рецепты, выбрать рецепт.

Класс Рецепты и Рецепт находятся в состоянии агрегации.

Класс Главное меню находится в состоянии ассоциации с классами Камера и Рецепты.

### **2.3 Поведение системы поиска рецептов на основе фотографий продуктов**

Для отображения изменения поведения приложения в различные моменты времени построим диаграмму переходов состояний. Каждый переход выполняется при совершении определенных действий. В системе можно выделить несколько основных состояний:

- ожидание действий пользователя;
- открыта камера;
- изменён поисковый список;
- показан список рецептов;
- показан один рецепт.

Отобразив данные состояния на диаграмме и обозначив действия по которым они происходят, получим поведение системы поиска рецептов на основе фотографий продуктов. Так как система не обладает развернутым интерфейсом, представленная схема отображает все возможные действия пользователя в разрабатываемой системе.

На рисунке 2.5 представлено поведение системы поиска рецептов на основе фотографий продуктов.

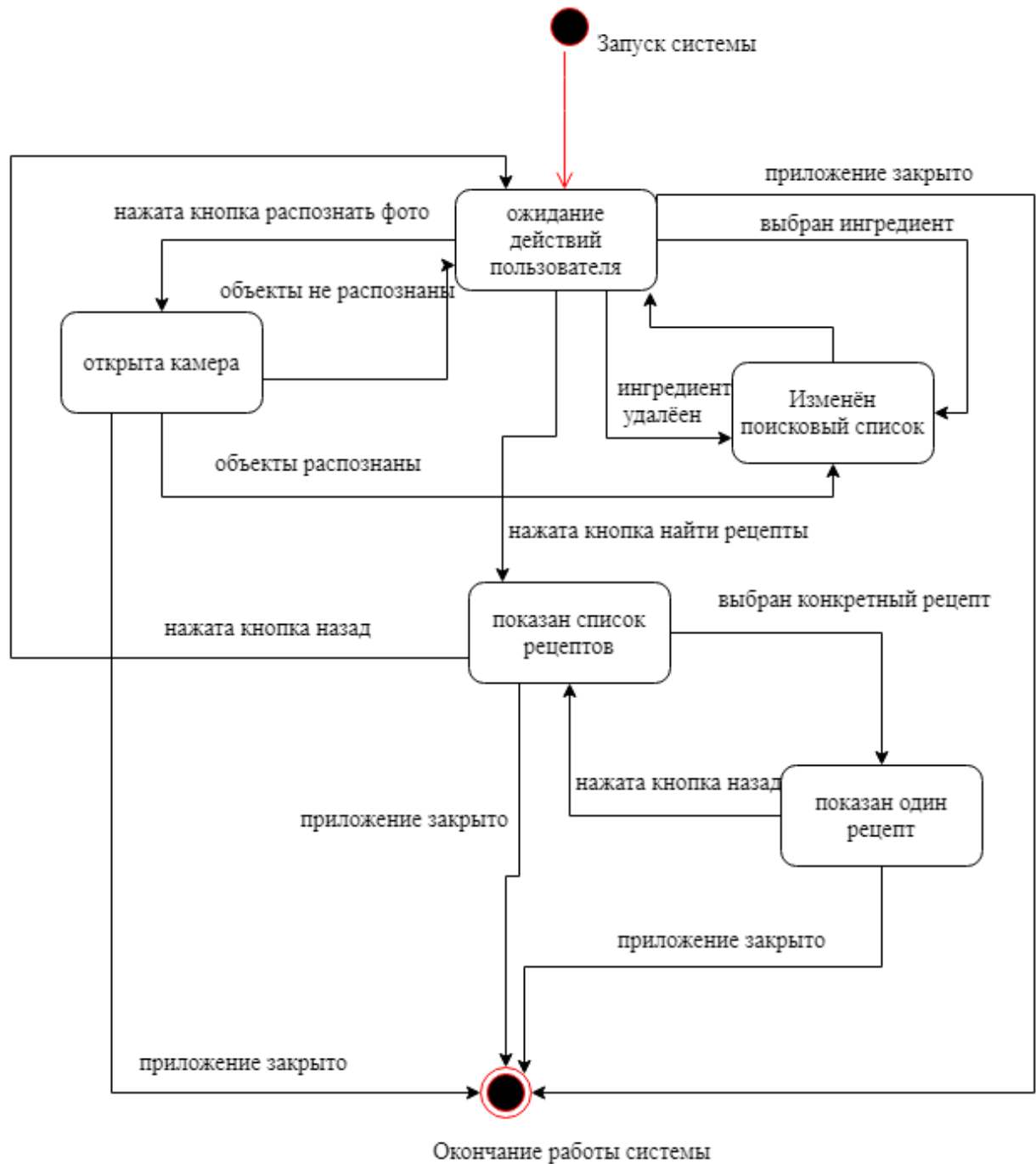


Рисунок 2.5 – Поведение системы поиска рецептов на основе фотографий продуктов

Из начального состояния выполняется переход в состояние ожидания действий пользователя. Если пользователь выберет распознавание изображения, состояние изменится на открытую камеру. Если пользователь закроет приложение состояние изменится на окончание работы системы. Если пользователь добавит ингредиент, он перейдет в состояние добавлены

ингредиенты. Если пользователь нажимает кнопку найти рецепты, список имеющихся ингредиентов передается в базу данных, состояние изменяется на показан список рецептов

Если пользователь находится в состоянии открытой камеры и объекты распознаны, то они добавятся в список ингредиентов для поиска, состояние изменится на добавлены ингредиенты. Если ингредиенты небыли найдены, состояние изменится на ожидание действий пользователя.

Если пользователь находится в состоянии показан список рецептов и выбирает конкретный рецепт, он переходит в состояние показан один рецепт.

Если пользователь находится в состоянии показан один рецепт и нажимает кнопку назад, он переходит в состояние показан список рецептов.

При переходе из состояния показан список рецептов и нажатой кнопке назад состояние изменяется на ожидание действий пользователя, список ингредиентов для поиска очищается.

## 2.4 Проектирование базы данных

Для хранения рецептов в системе будет использоваться база данных. Это позволит хранить данные локально, а значит доступ в интернет будет необязателен. Используется реляционная база данных.

Сущности в проектируемой БД:

- ингредиент;
- рецепт;
- ингредиенты\_в\_рецепте.

Схема БД представлена на рисунке 2.6.

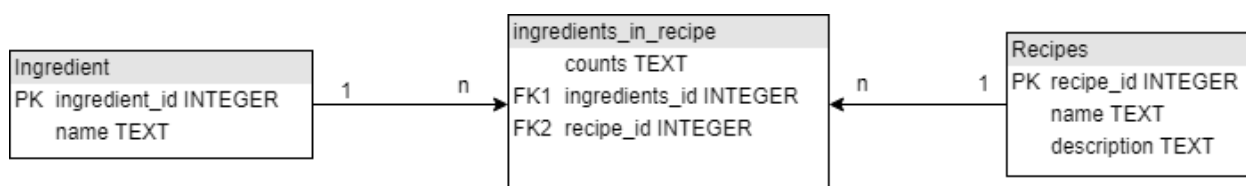


Рисунок 2.6 – Схема базы данных



Таблица Ingredient содержит поля: ingredient\_id (первичный ключ), name.

Таблица Recipes содержит поля: recipe\_id (первичный ключ), name, description.

Таблица ingredients\_in\_recipe была введена, так как отношение между сущностями ингредиент и рецепты многие-ко-многим. Это промежуточная таблица. Дополнительным полем в таблице является поле counts, необходимое для определения количества каждого ингредиента в конкретном рецепте.

Для обеспечения целостности базы выберем стратегию обновления и удаления CASCADE. Стратегия CASCADE предполагает преобразование записей в связанной таблице таким образом, что база данных оставалось корректной при удалении некоторых данных. Стратегия CASCADE при модификации записей материнской таблицы изменяются соответствующие значения в дочерней таблице на новые значения. При удалении записей материнской таблицы подразумевает удаление связанных записей зависимой таблицы.

## **3 РАЗРАБОТКА СИСТЕМЫ ПОИСКА РЕЦЕПТОВ НА ОСНОВЕ ФОТОГРАФИЙ ПРОДУКТОВ**

### **3.1 Проектирование структуры ПО**

Чтобы использовать систему поиска рецептов по фотографиям продуктов на мобильных устройствах, необходимо выполнить два этапа:

- обучить нейронную сеть, распознающую продукты питания;
- создать мобильное приложение, использующее обученную модель.

Мобильное приложение было решено сделать клиентским приложением. Это позволит использовать программу без соединения с интернетом. Данный вариант разработки называется нативным.

### **3.2 Проектирование алгоритмов приложения**

Для реализации полного функционала приложения необходимо реализовать следующие алгоритмы:

- добавление ингредиентов в поисковый запрос;
- удаление ингредиентов из поискового запроса;
- распознавание продуктов на фото;
- просмотр рецептов.

В начале алгоритма проверяется список ингредиентов. Если он пуст, происходит извлечение из памяти всех возможных продуктов для подбора рецептов.

Затем, когда элементы в списке есть, будет найден элемент, по которому было произведено нажатие и он поместится в список для поиска рецептов.

После этого данный элемент извлекается из основного списка ингредиентов.

Вышеописанный алгоритм представлен на рисунке 3.1.

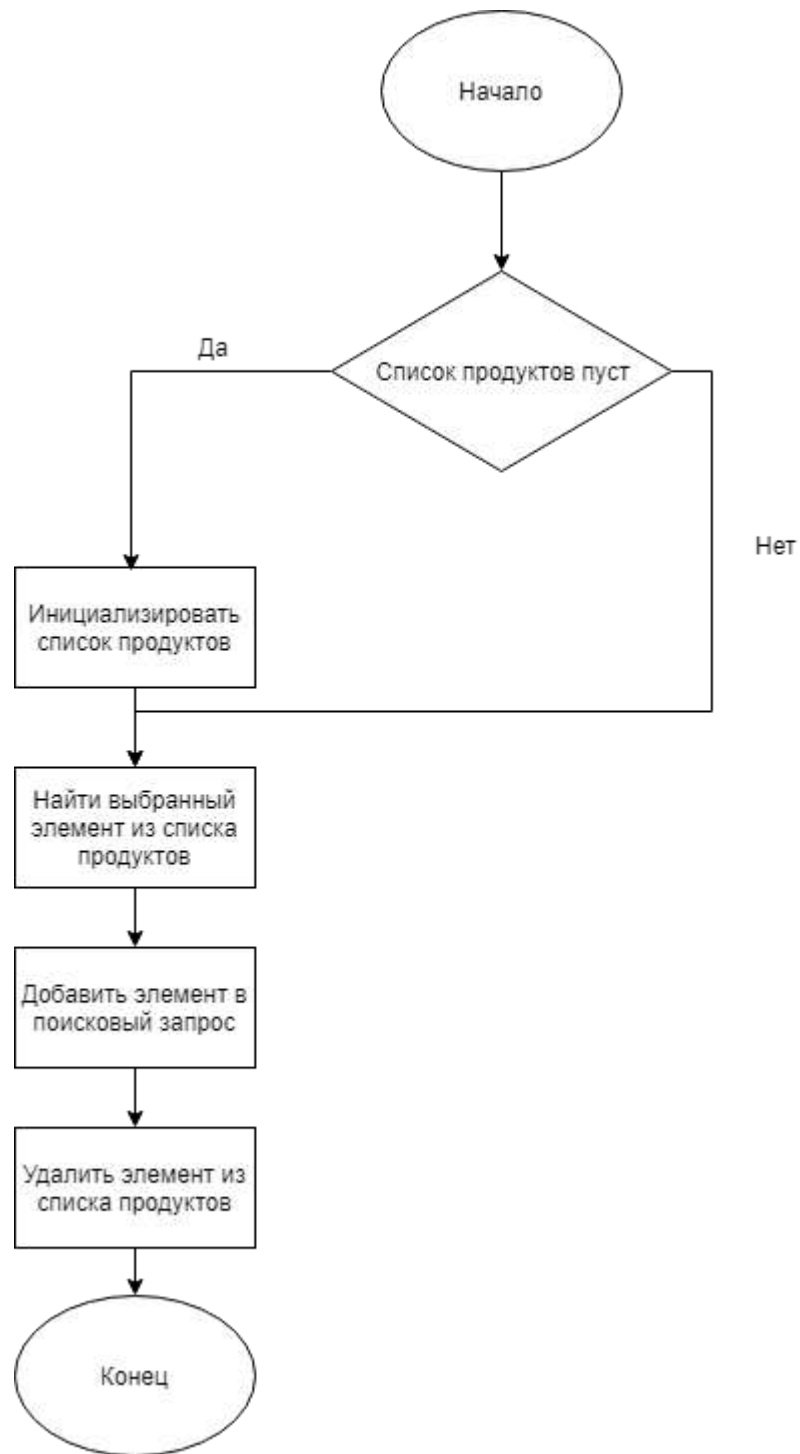


Рисунок 3.1 – Алгоритм добавления ингредиентов

В начале алгоритма проверяется список, содержащий ингредиенты для поиска рецептов. Если он пуст, алгоритм заканчивает свою работу.

Иначе, происходит поиск нажатого элемента и выполняется его удаление из поискового списка.

После происходит добавление данного элемента в список ингредиентов и обновление самого списка элементов. Вышеописанный алгоритм представлен на рисунке 3.2.

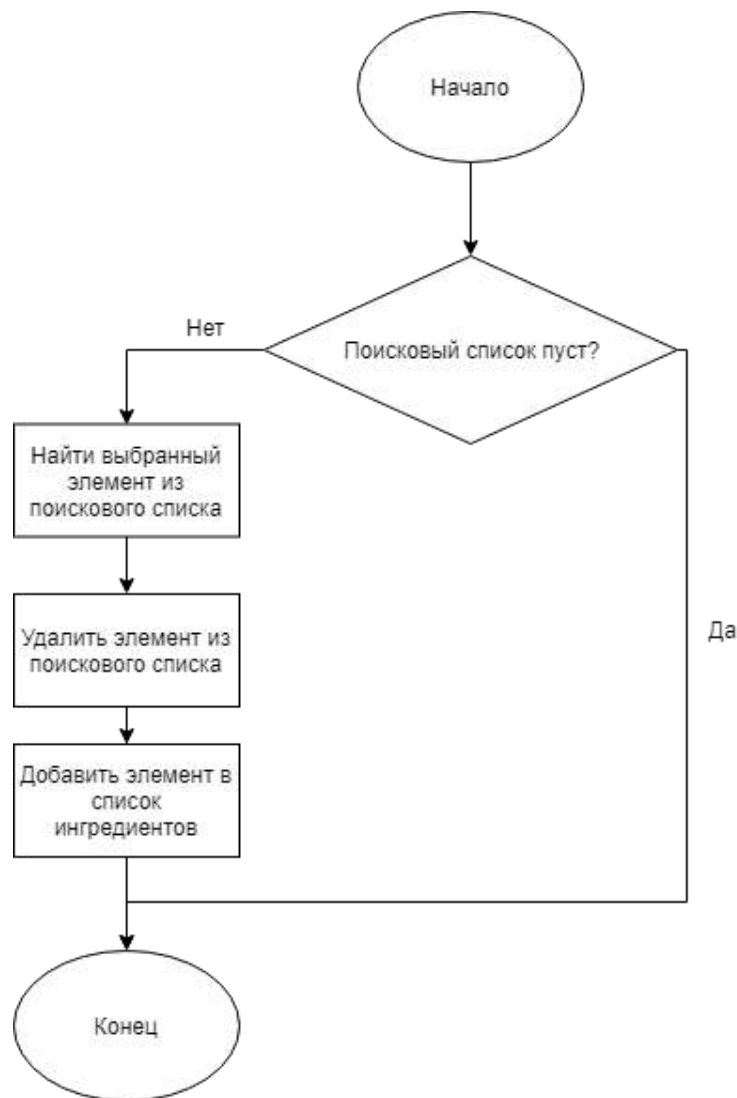


Рисунок 3.2 – Алгоритм удаления ингредиентов

В начале алгоритма из исходного изображения выделяется квадратная часть, соответствующая области, в которой находится продукты.

Затем, изображение сжимается до входных размеров обученной модели. По средствам сторонних функций происходит распознавание объектов на изображении. Результатом выполнения данных функций является структура со следующими элементами:

- прямоугольник, внутри которого находится распознанный объект;

- метка класса, которому принадлежит распознанный объект;
- вероятность принадлежности классу;
- число распознаваний.

Для отображения рецептов необходимо использовать один показатель – метку класса.

Если вероятность принадлежности класса меньше порогового значения, то объект отсеивается

Пройдя в цикле по всем найденным элементам, добавить в поисковый список ингредиент, если он еще не добавлен.

Вышеописанный алгоритм представлен на рисунке 3.3.



Рисунок 3.3 – Алгоритм распознавания продуктов на фото, лист 1



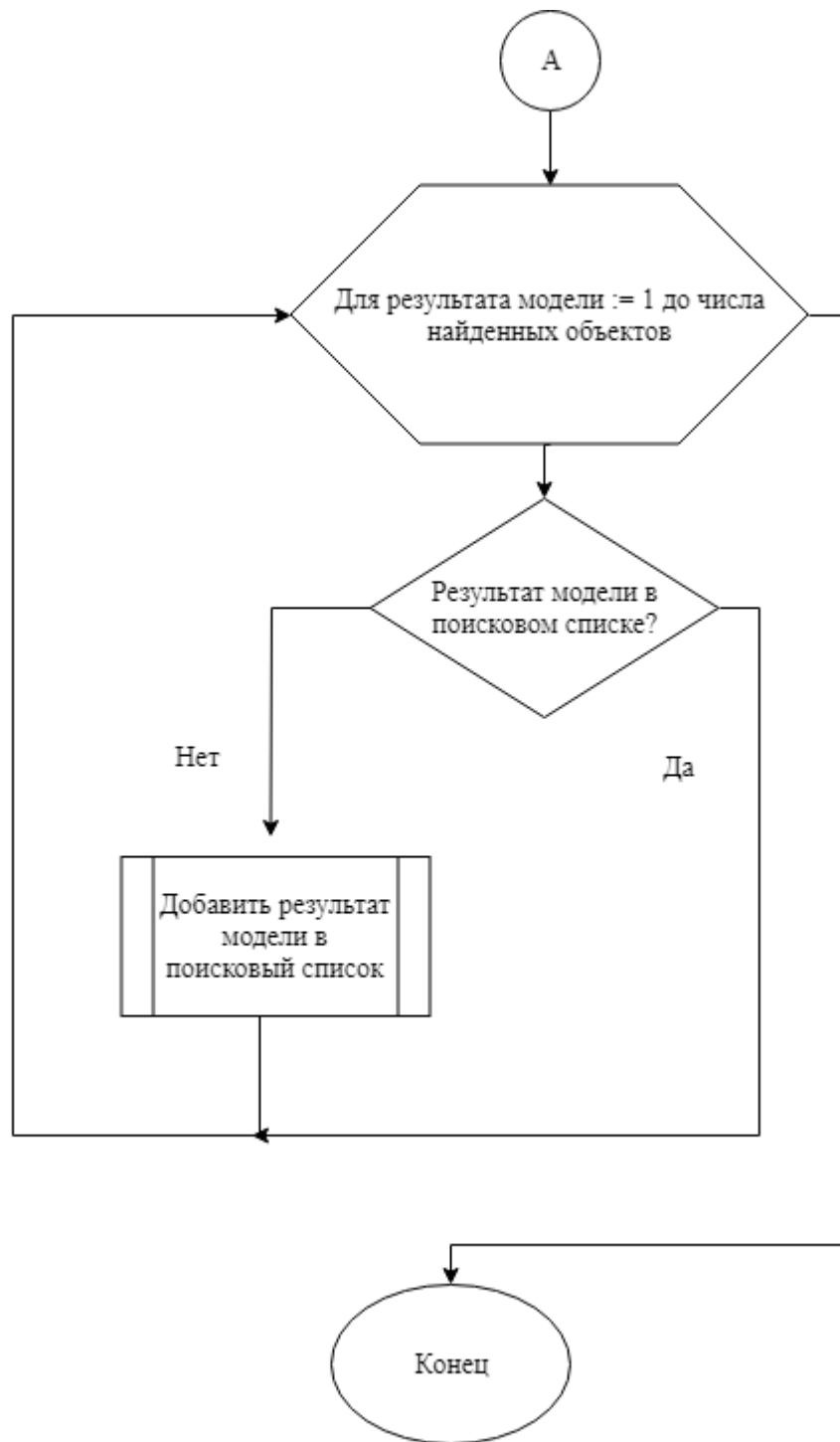


Рисунок 3.3 – Алгоритм распознавания продуктов на фото, лист 2

Алгоритм начинает работу с формирования запроса к базе данных, для получения списка рецептов, которые можно приготовить, используя ингредиенты из поискового списка.

Далее, результаты извлекаются и формируется список рецептов.

Вышеописанный алгоритм представлен на рисунке 3.4.



Рисунок 3.4 – Алгоритм просмотра рецептов

### 3.3 Проектирование дообучения модели

Для готовой модели, находящейся в памяти устройства, необходимо выполнение двух условий:

- её размер не слишком большой (не больше 30 мб);
- скорость позволяет пользователю получить мгновенный результат (менее 0.1 с).

Чтобы получить готовую модель, распознающую необходимые классы, можно воспользоваться уже обученной моделью. Это сэкономит время на выделение базовых признаков и позволит завершить обучение сети в короткие

сроки. Структура модели выбирается исходя из необходимых условий: точность, скорость, объём.

Для любой модели объектного распознавания, используемой библиотекой глубокого машинного обучения tensorflow алгоритм дообучения сводится к следующим шагам.

Во-первых, необходимо определить тип детектора и нейронной сети для извлечения признаков. Готовые модели можно скачать из репозитория tensorflow/models. Любая модель содержит конфигурационный файл, в котором указываются все её характеристики. Настройка данного файла является частью дообучения модели.

Во-вторых, необходимо подготовить наборы данных для обучения и для проверки. Нейронные сети обучаются за счёт метода обратного распространения ошибки. На каждой итерации считается ошибка. Цель обучения – минимизировать ошибку.

Набор данных, используемый для обучения необходимо разметить. Это необходимо для сбора информации о координатах прямоугольных рамок и класса, который находится внутри конкретной рамки.

После, размеченный набор необходимо преобразовать в формат необходимый для дообучения. В файле конфигурации необходимо указать путь до обучающей выборки и до проверочной выборки. Формат записи – tfrecord.

Для дообучения модели необходимо создать файл с метками класса, содержащим два поля: id, name. Разрешение файла – ptxt.

После можно приступить к дообучению модели. Необходимо убедиться, что установлены все необходимые для этого пакеты из Tensorflow Object Detection API.

После дообучения модель необходимо конвертировать в формат, использующийся для мобильных устройств. Tensorflow Lite – это набор утилит, которые помогают разработчикам запускать модели TensorFlow на мобильном. TensorFlow Lite включает в себя следующие компоненты:

- TensorFlow Lite интерпретатор, который запускает специальным образом оптимизированную модель;
- TensorFlow Lite конвертер, который конвертирует TensorFlow модель в эффективную для использования интерпретатором.

Конвертированная модель в формате tflite может быть помещена в память устройства и использоваться интерпретатором TensorFlow. Конвертированные модели обладают меньшим размером за счёт способа представления данных, при этом скорость вычислений увеличивается. За счёт оптимизации удаётся оставить прежнюю точность модели.

Общий алгоритм по дообучению существующей модели представлен на рисунке 3.5.

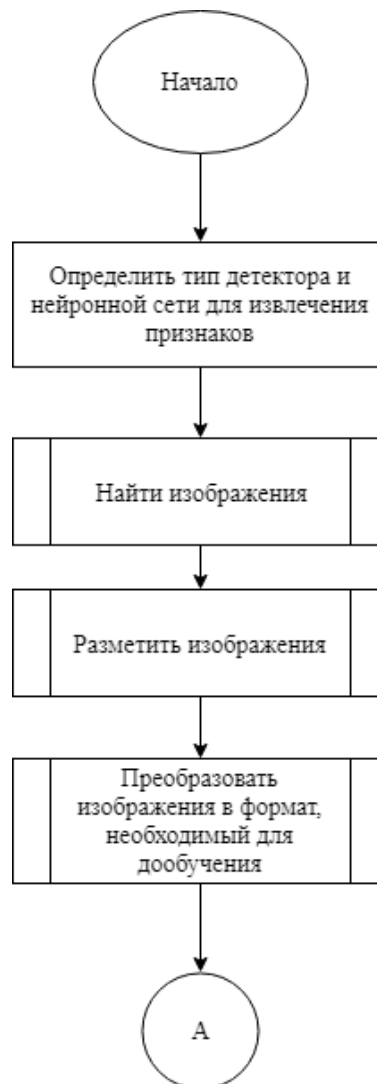


Рисунок 3.5 – Алгоритм дообучения модели, лист 1

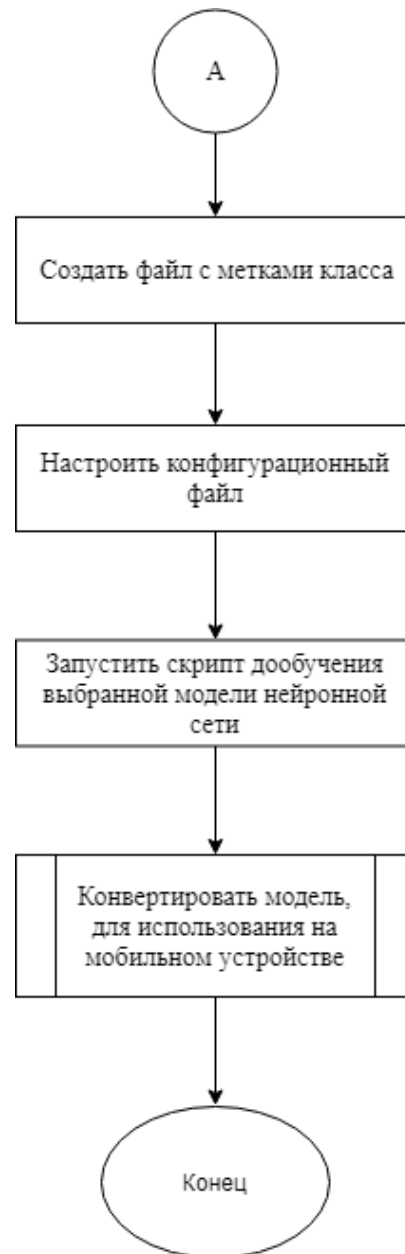


Рисунок 3.5 – Алгоритм дообучения модели, лист 2



## **4 РЕАЛИЗАЦИЯ СИСТЕМЫ ПОИСКА РЕЦЕПТОВ НА ОСНОВЕ ФОТОГРАФИЙ ПРОДУКТОВ**

### **4.1 Выбор мобильной платформы**

Было принято решение использовать для разработки Android – это платформа, разработкой которой занимается компания Google. В настоящий момент является одной из популярных платформ, выпуск первых телефонов под управлением Android начался в октябре 2008 года.

Преимущества:

- Android является открытой платформой. Она была создана на основе открытого исходного кода и находится в свободном распространении, такой способ предоставления данных дает разработчикам множество возможностей, например, наличие доступа к изучению принципов работы неизвестных функций. Каждый пользователь может внести вклад в развитие платформы, для этого в интернете существуют специальные форумы и сообщества;
- Object Detection Lite – библиотека от компании Google, позволяющая запускать обученные на компьютере нейронные сети с мобильных устройств. Протестирована и отлажена в основном на Android смартфонах;
- Для тестирования приложения, написанного под Android, подойдет большое количество смартфонов.

### **4.2 Выбор среды разработки**

Для реализации разрабатываемой среды предполагается использование языков программирования Java, Python, SQLITE.

Java является одним из самых популярных языков программирования на данный момент, он является официальным языком разработки приложений под Android.

Python – высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и

читаемости кода. Стандартная библиотека включает большой объём полезных функций. Большинство модулей, связанных с машинным обучением написаны на Python.

Самый крупный Фреймворк, для создания глубоких нейронных сетей – Tensorflow также имеет модуль, написанный на Python. Несмотря на то, что реализация данной библиотеки написана на быстром C++.

SQLite является простой базой данных, легко интегрируемой в среду разработки Android Studio. Размер базы данных рецептов не слишком большой, а количество обращений не превышает 2 обращений за секунду. При этом SQLite занимает минимальный размер в памяти смартфона.

### 4.3 Выбор модели нейронной сети

Tensorflow имеет набор моделей, обученных на больших наборах данных (COCO, Kitti, Open Images, AVA, iNaturalist).

Модель состоит из выделителя признаков и детектора. Модели, обученные на датасете COCO из 90 классов представлены на рисунке 4.1.

**COCO-trained models**

Model name	Speed (ms)	COCO mAP[^1]	Outputs
<a href="#">ssd_mobilenet_v1_coco</a>	30	21	Boxes
<a href="#">ssd_mobilenet_v1_0.75_depth_coco</a> ☆	26	18	Boxes
<a href="#">ssd_mobilenet_v1_quantized_coco</a> ☆	29	18	Boxes
<a href="#">ssd_mobilenet_v1_0.75_depth_quantized_coco</a> ☆	29	16	Boxes
<a href="#">ssd_mobilenet_v1_ppn_coco</a> ☆	26	20	Boxes
<a href="#">ssd_mobilenet_v1_fpn_coco</a> ☆	56	32	Boxes
<a href="#">ssd_resnet_50_fpn_coco</a> ☆	76	35	Boxes
<a href="#">ssd_mobilenet_v2_coco</a>	31	22	Boxes
<a href="#">ssd_mobilenet_v2_quantized_coco</a>	29	22	Boxes
<a href="#">ssdlite_mobilenet_v2_coco</a>	27	22	Boxes
<a href="#">ssd_inception_v2_coco</a>	42	24	Boxes

Рисунок 4.1 – Список предобученных моделей Tensorflow Object Detection

На данный момент для мобильных приложений поддерживается только Single Shot Detector. Скорость работы данных моделей очень быстрая, это означает, что её можно использовать для распознавания объектов в реальном времени.

Для реализуемой системы будем использовать модель под названием `ssd_mobilenet_v1_coco`.

Каждая модель из «зоопарка» Tensorflow имеет свой конфигурационный файл, в котором указаны все параметры, необходимые для обучения модели. Настроив данный конфиг, и указав в нём новые данные можно дообучить модель.

Дообучение модели происходит намного быстрее, так как базовые выделители признаков уже сформированы.

#### **4.4 Подготовка данных для обучения**

Для дообучения модели создадим набор данных, состоящий из 18 классов продуктов:

- банан;
- говядина;
- свёкла;
- капуста;
- морковь;
- сыр;
- курица;
- огурец;
- творог;
- яйцо;
- баклажан;
- чеснок;
- лимон;

- молоко;
- лук;
- картофель;
- сладкий перец;
- помидор.

Для каждого класса подготовим фотографии. Будем использовать Bing Image Search API, возвращающим ответы в формате JSON. Для загрузки изображения в жесткий диск необходимо знать его url. Bing API хранит эту информацию в «value»: «contentUrl». Реализация скрипта для извлечения изображений представлена в файле `search_bing_api.py`, находящемся в приложении Б.

Для каждого класса найдем 100 фотографий для обучения и 20 фотографий для проверки. Фотографии могут включать как один объект на сцене, так и несколько разных. Пример собранных фотографий представлен на рисунке 4.2.



Рисунок 4.2 – Набор данных для дообучения модели

Далее необходимо локализовать объекты на каждом изображении. Для этого будет использована программа labelImg. Пример разметки изображения представлен на рисунке 4.3.

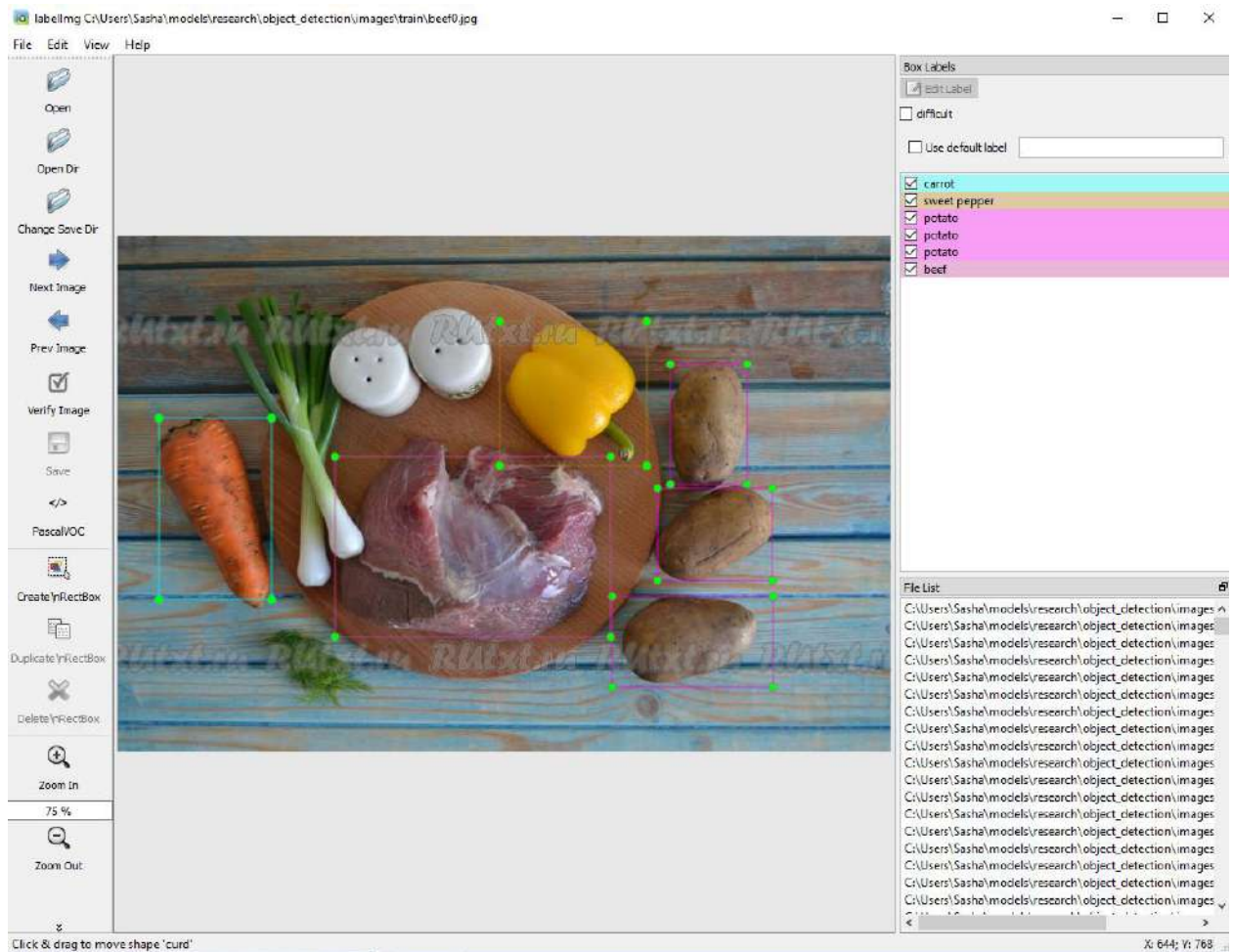


Рисунок 4.3 – Разметка изображений в программе labelImg

Данная программа создает xml документ для каждого размеченного файла, в котором хранится информация о принадлежности к классу, названии исходного изображения, координатах прямоугольника, внутри которого находится объект.

Пример такого xml файла представлен на рисунке 4.4:

```

<?xml version="1.0" encoding="UTF-8"?>
<annotation>
  <folder>milk</folder>
  <filename>milk42.jpg</filename>
  <path>/home/sasha/MyProjects/tensorfow_from_sourse/dataset/milk/milk42.jpg</path>
  <source>
    <database>Unknown</database>
  </source>
  <size>
    <width>0</width>
    <height>0</height>
    <depth>3</depth>
  </size>
  <segmented>0</segmented>
  <object>
    <name>milk</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <bndbox>
      <xmin>487</xmin>
      <ymin>9</ymin>
      <xmax>1124</xmax>
      <ymax>1575</ymax>
    </bndbox>
  </object>
</annotation>

```

Рисунок 4.4 – Информация о размеченном изображении в xml формате

Для обучения подготовленной модели данные о классе и границах прямоугольника должны быть представлены в формате record. Для преобразования был реализован скрипт `generate_tfrecord.py`, который представлен в приложении Б. После того, как подготовлены данные, необходимо настроить `ssd_mobilenet_v1.config` файл.

Указать в нём новое количество классов, `batch_size`, число тестовых изображений и прописать пути до двух файлов `train.record`, в котором находится обучающая выборка и `test.record`, в котором находится тестовая выборка, а также путь до файла с настроенными весами `model.ckpt`. Используемый для дообучения конфигурационный файл представлен в приложении Б.



#### 4.5 Дообучение модели

Для обучения модели необходимо запустить скрипт train.py. Дообучение модели проводилось со следующими настройками конфигурационного файла:

- количество шагов обучения – 160000;
- число тренировочных объектов (batch size) – 4;
- число классов – 18;
- число тестовых объектов – 382;
- aspect ratio: 1, 2, 0.5, 3, 0.33;
- коэффициент ошибки локализации – 0.00004;
- коэффициент ошибки классификации – 0.00004.

Дообучение производилось на видеокарте NVIDIA GEFORCE GTX 1060 3GB.

Размер ошибки локализации представлен на рисунке 4.5

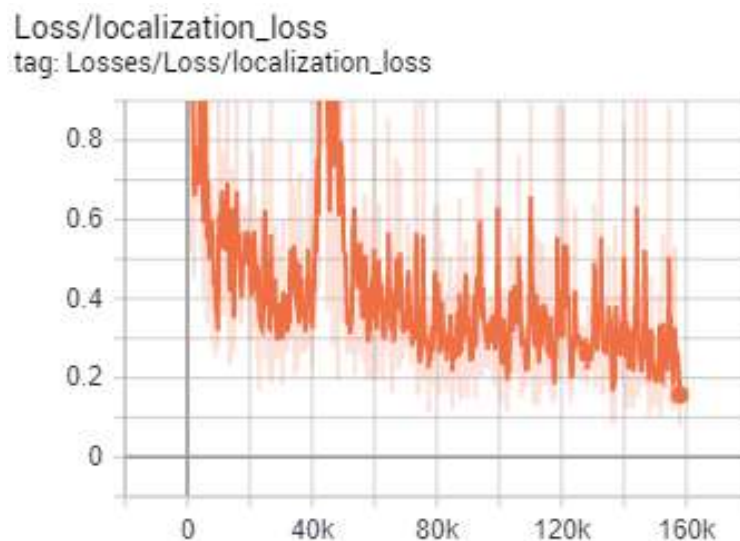


Рисунок 4.5 – Размер ошибки локализации

Размер ошибки классификации представлен на рисунке 4.6.

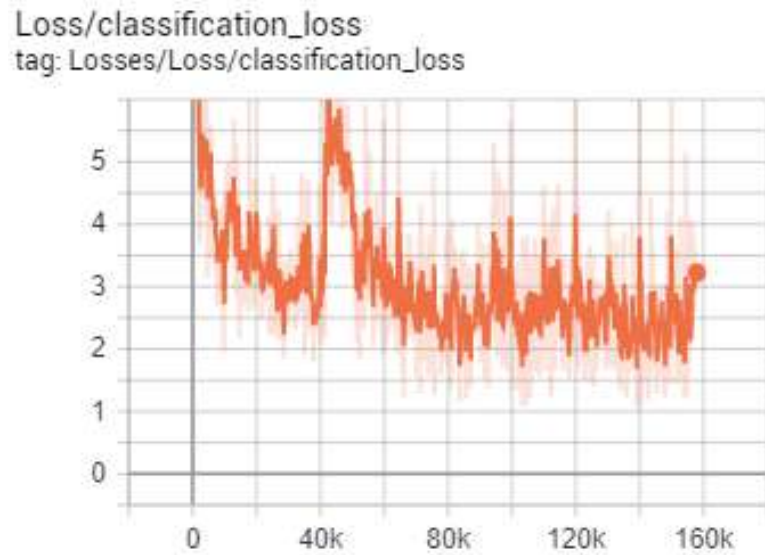


Рисунок 4.6 – Размер ошибки классификации

Размер суммарной ошибки представлен на рисунке 4.7

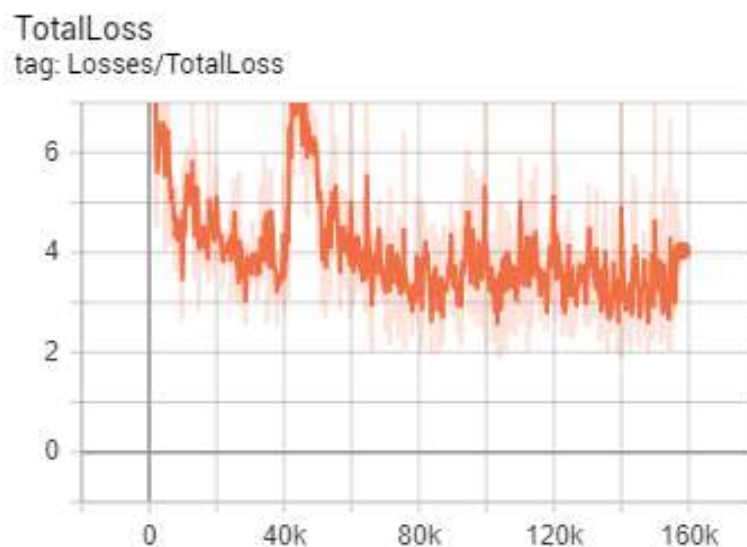


Рисунок 4.7 – Размер суммарной ошибки

Получившаяся модель имеет mAP (mean Average Precision) – среднюю точность 20.7%. Скорость срабатывания модели 0.032 секунды. Данные значения показывают результат, как и у предобученной модели. Для увеличения точности распознавания необходимо использовать модели с наибольшим mAP, но их обучение требует больших объемов вычислительных мощностей.

#### 4.6 Импорт дообученной модели в Android Studio

После дообучения модели необходимо конвертировать её для использования в мобильном приложении. Для этого используем команду, представленную на рисунке 4.8:

```
bazel run --config=opt tensorflow/lite/toco:toco --
--input_file=/home/sasha/MyProjects/tensorfow_from_sourse/
tflite/tflite_graph.pb
--output_file=/home/sasha/MyProjects/tensorfow_from_sourse/
tflite/detect.tflite
--input_shapes=1,300,300,3
--input_arrays=normalized_input_image_tensor
--output_arrays='TFLite_Detection_PostProcess','TFLite_Detection_PostProcess:1',
'TFLite_Detection_PostProcess:2','TFLite_Detection_PostProcess:3'
--inference_type=FLOAT
--allow_custom_ops
```

Рисунок 4.8 – Команда для преобразования модели распознавания объектов в формат, используемый Tensorflow Lite

Данная команда преобразует модель, находящуюся в `input_file`, конвертируемая модель будет находится в `output_file`, `input_shapes` задают размеры первого слоя модели, `input_arrays` указывают тип входных данных для модели, `output_arrays` соответствует 4 массивам: класс объекта, точность, координаты положения, количество, `inference_type = FLOAT` означает, что модель использует 32 битные веса. Можно указать параметр `QUANT`, тогда все веса будут занимать 4 бита. При этом точность распознавания ухудшается. Было принято решение оставить полноразмерные веса для модели.

Полученная модель `detect.tflite` может быть использована для мобильной разработки. Необходимо добавить её в папку `assets` вместе с файлом, содержащим название классов. В первой строке данного файла будет находится «???», отвечающая за фоновый класс, это является особенность SSD.

После этого необходимо добавить модуль `org.tensorflow:tensorflow-lite` в раздел `dependencies` файла `build.gradle`. Это позволит использовать метод

Interpreter библиотеки Tensorflow Lite. При создании экземпляра интерпретатора, в него передаётся путь до модели и файла с названием классов.

Чтобы выполнить прямой проход через имеющуюся модель, необходимо обратиться к методу `runForMultipleInputsOutputs` и передать в него преобразованное изображение в виде массива размером  $1 \times 300 \times 300 \times 3$ .

Результат возвращается в 4 массива. Массив с координатами – трехмерный массив, массив классов – двумерный массив, массив оценок – двумерный массив. Массив количества распознаваний – одномерный массив.

Код основных классов для Android приложения, позволяющего распознавать множество продуктов питания на изображении и подбирать рецепты представлен в Приложении А.

Необходимые для дообучения модели файлы представлены в Приложении Б.

#### 4.7 Демонстрация интерфейса реализованного приложения

На рисунках 4.9-4.14 представлен интерфейс реализованного мобильного приложения.



Рисунок 4.9 – Главное меню разработанного приложения



Рисунок 4.10 – Демонстрация работы фильтра при поиске



Рисунок 4.11 – Экран распознавания по фотографии

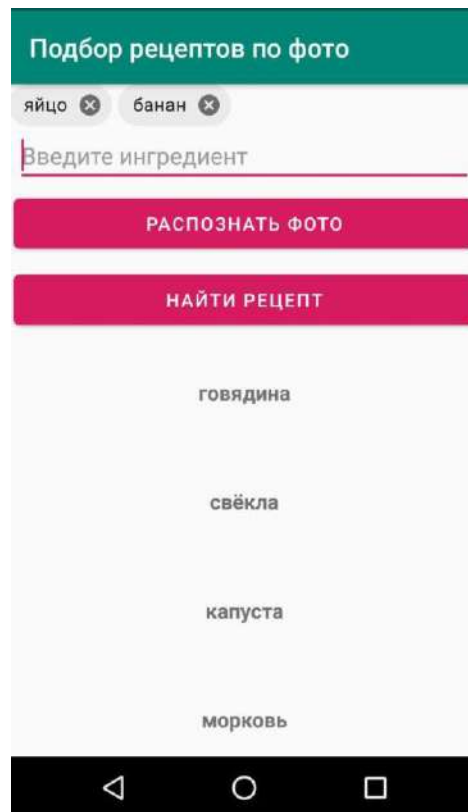


Рисунок 4.12 – Автоматическое заполнение ингредиентов, на основе результата детектирования на изображении



Рисунок 4.13 – Список всех рецептов по заданным ингредиентам



Рисунок 4.14 – Информация о конкретном рецепте

В случае использования свёрточных нейронных сетей, решающих задачу классификации, возникла бы необходимость делать фотографию каждого продукта по отдельности. И каждую фотографию прогонять через используемую сеть.

Реализованное приложение позволяет распознавать множество продуктов по одному изображению. Благодаря этому сокращается время, которое пользователь тратит на фотографии продуктов.

Пример работы приложения аналога, решающего задачу классификации для множества объектов на сцене представлен на рисунке 4.15.



Рисунок 4.15 – Принцип работы системы распознавания образов в приложении аналоге

Пример работы реализованного приложения, использующего SSD для выявления областей объектов и mobilenet v1 для классификации, позволяющего определить объекты на сцене за один прогон модели, представлен на рисунке 4.16.



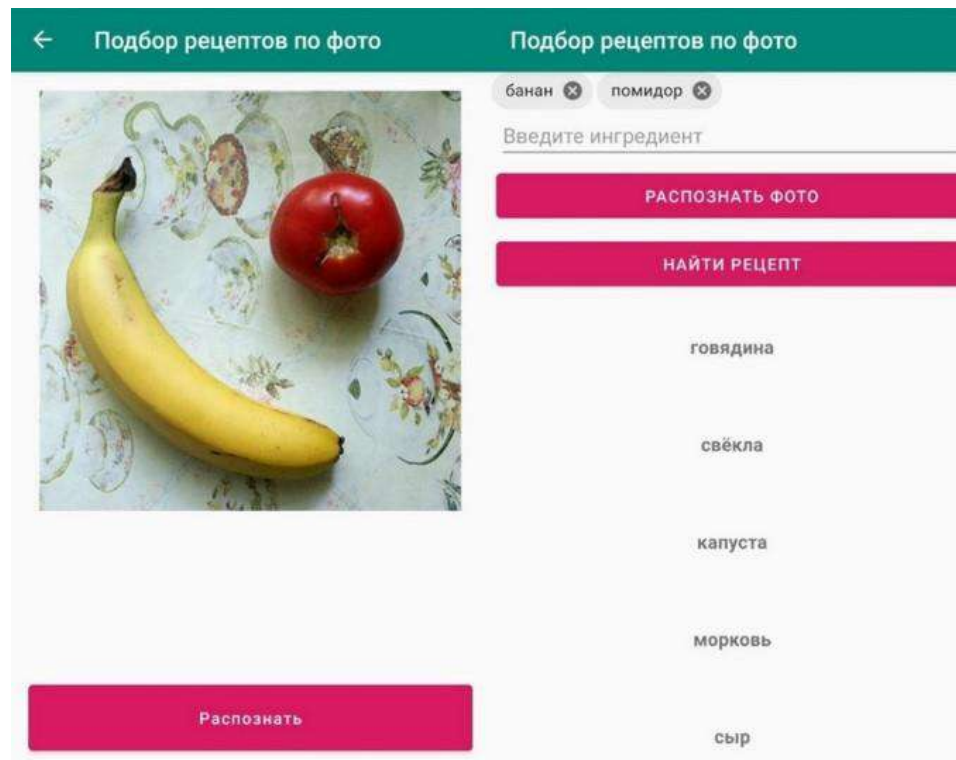


Рисунок 4.16 – Принцип работы системы распознавания образов в реализованном приложении

## ЗАКЛЮЧЕНИЕ

В результате выполнения выпускной квалификационной работы была выбрана модель нейронной сети `ssd_mobilenet_v1`, были найдены и подготовлены изображения для выбранной модели, на подготовленных изображениях была дообучена используемая модель, а также произведена конвертация модели, для работы с мобильными устройствами. Было спроектировано и реализовано приложение, использующее преобразованную модель, для автоматизированного поиска рецептов по фотографии продуктов. Оно позволяет распознать несколько ингредиентов за один проход модели.

Таким образом, поставленные задачи были решены, полученное решение позволяет сократить время на задачу классификации  $N$  объектов почти в  $N$  раз, следовательно, цель выпускной квалификационной работы считается достигнутой.

## СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Основные концепции нейронных сетей [Текст] / Каллан Р.: Пер. с англ. – М.: Изд-кий дом Вильямс, 2001. – 291с.
2. Обработка и анализ изображений в задачах машинного зрения: курс лекций и практических занятий [Текст] / Ю.В. Визильтер, С.Ю. Желтов, А.В. Бонларенко, М.В. Ососков, А.В. Моржин. – М.: Изд-во Физматкнига., 2010. – 689с.
3. Нейронные сети. Полный курс. Второе издание [Текст] / Саймон Хайкин.: Пер. с англ. – М.: Изд-кий дом Вильямс, 2006. – 1104с.
4. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications [Текст] / Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam, Google Inc. 2017. – 9 с.
5. SSD: Single Shot MultiBox Detector [Текст]/ Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, Alexander C. Berg. UNC Chapel Hill; Zoox Inc; Google Inc; University of Michigan, Ann-Arbor. 2015 – 17 с.
6. SSD object detection: Single Shot MultiBox Detector for real-time processing [Электронный ресурс]. – Режим доступа: [https://medium.com/@jonathan\\_hui/ssd-object-detection-single-shot-multibox-detector-for-real-time-processing-9bd8deac0e06](https://medium.com/@jonathan_hui/ssd-object-detection-single-shot-multibox-detector-for-real-time-processing-9bd8deac0e06). – Дата обращения 13.05.2019
7. Архитектура MobileNet-SSD [Электронный ресурс]. – Режим доступа: <https://hey-yahei.cn/2018/08/08/MobileNets-SSD/index.html>. – Дата обращения 17.05.2019
8. Tensorflow Lite [Электронный ресурс]. – Режим доступа: <https://www.tensorflow.org/lite/guide>. – Дата обращения 19.05.2019
9. Tensorflow Object Detection API [Электронный ресурс]. – Режим доступа: <https://github.com/tensorflow/models/tree/master/research/>

object\_detection/ g3doc. – Дата обращения 21.05.2019

10. Tensorflow Object Detection models [Электронный ресурс]. – Режим доступа: [https://www.tensorflow.org/lite/models/object\\_detection/overview](https://www.tensorflow.org/lite/models/object_detection/overview). – Дата обращения 21.05.2019

## ПРИЛОЖЕНИЕ А

### (обязательное)

### ЛИСТИНГ ПРОГРАММЫ

#### MainActivity.class

```

import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.content.Intent;
import android.os.Bundle;
import android.preference.PreferenceManager;
import android.text.Editable;
import android.text.TextWatcher;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

import com.google.android.material.chip.Chip;
import com.google.android.material.chip.ChipGroup;

import java.lang.reflect.Array;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class MainActivity extends AppCompatActivity implements
RecyclerViewItemSelectedListener, View.OnClickListener {

    public static final String INGREDIENTS = "my_ingr";
    public static final String FINDRES = "result";

    private RecyclerView recyclerView;
    private RecyclerViewAdapter recyclerViewAdapter;
    private List<Ingredients> ingredients = new ArrayList<>();
    private EditText userInput;
    private ChipGroup chipGroup;
    private Button findRecipesb, takeImageb;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        recyclerView = findViewById(R.id.recyclerView);
        userInput = findViewById(R.id.txt_name);
        chipGroup = findViewById(R.id.chipGroup);
        findRecipesb = findViewById(R.id.find_recipes);
        takeImageb = findViewById(R.id.take_photo);

        takeImageb.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(MainActivity.this,

```

```

CameraActivity.class);
        startActivityForResult(intent, 1);
    }
});

findRecipesb.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        findRecipes();
    }
});

recyclerView.setLayoutManager(new LinearLayoutManager(this));
recyclerView.setHasFixedSize(true);

getIngredients();

recyclerAdapter = new RecyclerViewAdapter(this, ingredients);
recyclerView.setAdapter(recyclerAdapter);

userInput.addTextChangedListener(new TextWatcher() {
    @Override
    public void beforeTextChanged(CharSequence s, int start, int
count, int after) {

    }

    @Override
    public void onTextChanged(CharSequence s, int start, int before,
int count) {
        String inputText = s.toString();
        List<Ingredients> newIngredients = new ArrayList<>();

        for (Ingredients ingredient : ingredients) {
            if (ingredient.getName().contains(inputText)) {
                newIngredients.add(ingredient);
            }
        }
        recyclerAdapter = new RecyclerViewAdapter(MainActivity.this,
newIngredients);
        recyclerView.setAdapter(recyclerAdapter);

    }

    @Override
    public void afterTextChanged(Editable s) {

    }
});

}

@Override
protected void onActivityResult(int requestCode, int resultCode,
@Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == 1) {
        if (resultCode == RESULT_OK) {
            ArrayList<String> res =
data.getStringArrayListExtra(MainActivity.FINDRES);
            for (String result : res) {

```

```

        addChip(result);
    }
}

private void findRecipes() {
    ArrayList<String> recipes = new ArrayList<>();
    for (int i = 0; i < chipGroup.getChildCount(); i++) {
        Chip chip = (Chip) chipGroup.getChildAt(i);
        recipes.add(chip.getText().toString());
    }
    Intent intent = new Intent(this, RecipeActivity.class);
    intent.putStringArrayListExtra(INGREDIENTS, recipes);

    startActivity(intent);
}

private void addChip(String ingredient) {
    for (int i = 0; i < chipGroup.getChildCount(); i++) {
        Chip chip = (Chip) chipGroup.getChildAt(i);
        if (chip.getText().equals(ingredient)) return;
    }

    Chip chip = new Chip(this);
    chip.setText(ingredient);
    chip.setCloseIconVisible(true);
    chip.setCheckable(false);
    chip.setClickable(false);
    chip.setOnCloseIconClickListener(this);
    chipGroup.addView(chip);
    chipGroup.setVisibility(View.VISIBLE);

    for (Ingredients ing : ingredients) {
        if (ing.getName().equals(ingredient)) {
            ingredients.remove(ing);
            recyclerAdapter = new RecyclerView.Adapter(MainActivity.this,
ingredients);
            recyclerView.setAdapter(recyclerAdapter);
            return;
        }
    }
}

@Override
public void onItemSelected(Ingredients ingredient) {
    addChip(ingredient.getName());
}

private void getIngredients() {
    List<String> names =
Arrays.asList(getResources().getStringArray(R.array.names));
    for (String name : names) {
        ingredients.add(new Ingredients(name));
    }
}

```

```

@Override
public void onClick(View v) {
    Chip chip = (Chip) v;
    chipGroup.removeView(chip);

    String str = (String) chip.getText();
    Ingredients ingredient = new Ingredients(str);
    ingredients.add(ingredient);

    recyclerAdapter = new RecyclerViewAdapter(this, ingredients);
    recyclerView.setAdapter(recyclerAdapter);
}
}

```

## CameraActivity.class

```

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.RectF;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

import com.wonderkiln.camerakit.CameraKitError;
import com.wonderkiln.camerakit.CameraKitEvent;
import com.wonderkiln.camerakit.CameraKitEventListener;
import com.wonderkiln.camerakit.CameraKitImage;
import com.wonderkiln.camerakit.CameraKitVideo;
import com.wonderkiln.camerakit.CameraView;

import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.Executor;
import java.util.concurrent.Executors;

public class CameraActivity extends AppCompatActivity {

    private static final String MODEL_PATH = "detect.tflite";
    private static final String LABEL_PATH = "labelmap.txt";
    private static final int INPUT_SIZE = 300;
    private static final float THRESHOLD = 0.4f;

    Classifier classifier;

    private Executor executor = Executors.newSingleThreadExecutor();

    private Button btnDetectObject;

    private CameraView cameraView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_camera);
        cameraView = findViewById(R.id.cameraView);

        btnDetectObject = findViewById(R.id.btnDetectObject);

        cameraView.addCameraKitListener(new CameraKitEventListener() {

```



```

@Override
public void onEvent(CameraKitEvent cameraKitEvent) {

}

@Override
public void onError(CameraKitError cameraKitError) {

}

@Override
public void onImage(CameraKitImage cameraKitImage) {
    Bitmap bitmap = cameraKitImage.getBitmap();

    bitmap = cropBitmapCenter(bitmap, cameraView.getWidth(),
cameraView.getHeight());

    bitmap = Bitmap.createScaledBitmap(bitmap, INPUT_SIZE,
INPUT_SIZE, false);

    final List<Classifier.Recognition> results =
classifier.recognizeImage(bitmap);
    ArrayList<String> res = new ArrayList<>();
    for (final Classifier.Recognition result : results) {
        final RectF location = result.getLocation();
        if (location != null && result.getConf() >= THRESHOLD) {
            res.add(result.toString());
        }
    }
    Intent resIntent = new Intent();
    resIntent.putStringArrayListExtra(MainActivity.FINDRES, res);
    setResult(RESULT_OK, resIntent);
    finish();
}

@Override
public void onVideo(CameraKitVideo cameraKitVideo) {

}

});

btnDetectObject.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        cameraView.captureImage();
    }
});

initTensorFlowAndLoadModel();
}

@Override
protected void onResume() {
    super.onResume();
    cameraView.start();
}

@Override
protected void onPause() {
    cameraView.stop();
}

```

```

        super.onPause();
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();
        executor.execute(new Runnable() {
            @Override
            public void run() {
                classifier.close();
            }
        });
    }

    private Bitmap cropBitmapCenter(Bitmap bitmap, int cropWidth, int
cropHeight) {
        int bitmapWidth = bitmap.getWidth();
        int bitmapheight = bitmap.getHeight();

        cropWidth = (cropWidth > bitmapWidth) ? bitmapWidth : cropWidth;
        cropHeight = (cropHeight > bitmapheight) ? bitmapheight : cropHeight;

        int newX = bitmapWidth / 2 - cropWidth / 2;
        int newY = bitmapheight / 2 - cropHeight / 2;

        return Bitmap.createBitmap(bitmap, newX, newY, cropWidth,
cropHeight);
    }

    private void initTensorFlowAndLoadModel() {
        executor.execute(new Runnable() {
            @Override
            public void run() {
                try {
                    classifier = TensorFlowImageClassifier.create(
                        getAssets(),
                        MODEL_PATH,
                        LABEL_PATH,
                        INPUT_SIZE
                    );
                    makeButtonVisible();
                } catch (final Exception e) {
                    throw new RuntimeException("Error initializing
TensorFlow!", e);
                }
            }
        });
    }

    private void makeButtonVisible() {
        runOnUiThread(new Runnable() {
            @Override
            public void run() {
                btnDetectObject.setVisibility(View.VISIBLE);
            }
        });
    }
}

```

## RecipeActivity.class

```

import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;

```

```

import androidx.recyclerview.widget.RecyclerView;

import android.content.Intent;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;

import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.PriorityQueue;

public class RecipeActivity extends AppCompatActivity implements
RecyclerViewItemTwoSelectedListener{

    private DatabaseHelper databaseHelper;
    private SQLiteDatabase database;
    private RecyclerView recyclerViewTwo;
    private RecyclerViewAdapterTwo recyclerViewAdapterTwo;
    private List<Recipes> recipes = new ArrayList<>();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_recipe);

        databaseHelper = new DatabaseHelper(this);
        try {
            databaseHelper.updateDataBase();
        } catch (IOException mIOException) {
            throw new Error("UnableToUpdateDatabase");
        }

        recyclerViewTwo = findViewById(R.id.recyclerView1);

        database = databaseHelper.getWritableDatabase();

        recyclerViewTwo.setLayoutManager(new LinearLayoutManager(this));
        recyclerViewTwo.setHasFixedSize(true);

        getRecipes();

        recyclerViewAdapterTwo = new RecyclerViewAdapterTwo(this, recipes);
        recyclerViewTwo.setAdapter(recyclerViewAdapterTwo);
    }

    private void getRecipes() {
        Intent intent = getIntent();
        ArrayList<String> ingredients =
intent.getStringArrayListExtra(MainActivity.INGREDIENTS);
        String[] ingr = ingredients.toArray(new String[ingredients.size()]);

        String query = "SELECT DISTINCT r.name, r.description, r.recipe_id
\n" +
            "FROM Recipes AS r \n" +
            "INNER JOIN ingredients_in_recipe AS i \n" +
            "ON i.recipe_id = r.recipe_id\n" +

```

```

        "INNER JOIN Ingredients AS ing\n" +
        "ON ing.ingredient_id = i.ingredient_id\n" +
        "WHERE ing.name IN (" + makePlaceholders(ingr.length) + ")";

Cursor cursor = database.rawQuery(query, ingr);

cursor.moveToFirst();
while (!cursor.isAfterLast()) {
    String listIngredients = getListIngredients(cursor.getString(2));
    recipes.add(new Recipes(cursor.getString(0), cursor.getString(1),
listIngredients));
    cursor.moveToNext();
}
cursor.close();
}

private String makePlaceholders(int len) {
    if (len < 1) {
        throw new RuntimeException("No placeholders");
    } else {
        StringBuilder sb = new StringBuilder(len * 2 - 1);
        sb.append("?");
        for (int i = 1; i < len; i++) {
            sb.append(",?");
        }
        return sb.toString();
    }
}

private String getListIngredients(String recipeId) {

    String query = "SELECT ing.name, i.counts\n" +
        "FROM Recipes AS r \n" +
        "INNER JOIN ingredients_in_recipe AS i \n" +
        "ON i.recipe_id = r.recipe_id AND i.recipe_id = ?\n" +
        "INNER JOIN Ingredients AS ing\n" +
        "ON i.ingredient_id = ing.ingredient_id";

    Cursor cursor = database.rawQuery(query, new String[] {recipeId});
    cursor.moveToFirst();
    String res = "";
    while (!cursor.isAfterLast()) {
        res += cursor.getString(0) + " - " + cursor.getString(1) + ", ";
        cursor.moveToNext();
    }

    res = res.substring(0, res.length()-2);
    cursor.close();

    return res;
}

@Override
public void onItemSelected(Recipes recipe) {
    Intent intent = new Intent(RecipeActivity.this, RecipeInfo.class);
    String[] recipeToSend = {recipe.getName(), recipe.getDescription(),
recipe.getListIngredients()};
    intent.putExtra("recipe", recipeToSend);
    startActivity(intent);
}
}

```

## TensorflowImageClassifier.class

```

import android.content.res.AssetFileDescriptor;
import android.content.res.AssetManager;
import android.graphics.Bitmap;
import android.graphics.RectF;

import org.tensorflow.lite.Interpreter;

import java.io.BufferedReader;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.nio.ByteBuffer;
import java.nio.ByteOrder;
import java.nio.MappedByteBuffer;
import java.nio.channels.FileChannel;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Vector;

public class TensorFlowImageClassifier implements Classifier {

    private static final int MAX_RESULTS = 10;
    private static final int MAX = 10;

    private static final int BATCH_SIZE = 1;

    private static final int PIXEL_SIZE = 3;

    private static final int IMAGE_MEAN = 128;
    private static final float IMAGE_STD = 128.0f;

    private int inputSize;

    private Vector<String> labels = new Vector<String>();
    private int[] intValues;
    private float[][][] outputLocations;
    private float[][] outputClasses;
    private float[][] outputScores;
    private float[] numDetections;

    private ByteBuffer imgData;
    private Interpreter interpreter;

    private TensorFlowImageClassifier() {
    }

    private MappedByteBuffer loadModelFile(AssetManager assetManager, String
modelPath) throws
        IOException {
        AssetFileDescriptor fileDescriptor = assetManager.openFd(modelPath);
        FileInputStream inputStream = new
FileInputStream(fileDescriptor.getFileDescriptor());
        FileChannel fileChannel = inputStream.getChannel();
        long startOffset = fileDescriptor.getStartOffset();
        long declaredLength = fileDescriptor.getDeclaredLength();
        return fileChannel.map(FileChannel.MapMode.READ_ONLY, startOffset,

```

```

declaredLength);
    }

    static Classifier create(AssetManager assetManager, String modelPath,
String labelPath, int inputSize)
        throws IOException {
        TensorFlowImageClassifier classifier = new
TensorFlowImageClassifier();
        classifier.interpreter = new
Interpreter(classifier.loadModelFile(assetManager, modelPath));
        InputStream labelsInput = assetManager.open(labelPath);
        BufferedReader br = new BufferedReader(new
InputStreamReader(labelsInput));
        String line;
        while ((line = br.readLine()) != null) {
            classifier.labels.add(line);
        }
        br.close();
        classifier.inputSize = inputSize;

        classifier.imgData = ByteBuffer.allocateDirect(4 * BATCH_SIZE *
inputSize * inputSize * PIXEL_SIZE);
        classifier.imgData.order(ByteOrder.nativeOrder());
        classifier.intValues = new int[classifier.inputSize *
classifier.inputSize];

        classifier.outputLocations = new float[1][MAX_RESULTS][4];
        classifier.outputClasses = new float[1][MAX_RESULTS];
        classifier.outputScores = new float[1][MAX_RESULTS];
        classifier.numDetections = new float[1];
        return classifier;
    }

    @Override
    public List<Recognition> recognizeImage(Bitmap bitmap) {
        bitmap.getPixels(intValues, 0, bitmap.getWidth(), 0, 0,
bitmap.getWidth(), bitmap.getHeight());

        imgData.rewind();
        for (int i = 0; i < inputSize; ++i) {
            for (int j = 0; j < inputSize; ++j) {
                int pixelValue = intValues[i * inputSize + j];
                imgData.putFloat((((pixelValue >> 16) & 0xFF) - IMAGE_MEAN) /
IMAGE_STD);
                imgData.putFloat((((pixelValue >> 8) & 0xFF) - IMAGE_MEAN) /
IMAGE_STD);
                imgData.putFloat(((pixelValue & 0xFF) - IMAGE_MEAN) /
IMAGE_STD);
            }
        }

        Object[] inputArray = {imgData};
        Map<Integer, Object> outputMap = new HashMap<>();
        outputMap.put(0, outputLocations);
        outputMap.put(1, outputClasses);
        outputMap.put(2, outputScores);
        outputMap.put(3, numDetections);

        interpreter.runForMultipleInputsOutputs(inputArray, outputMap);

        final ArrayList<Recognition> recognitions = new ArrayList<>(MAX);
        for (int i = 0; i < MAX; ++i) {
            final RectF detection =

```

```

        new RectF(
            outputLocations[0][i][1] * inputSize,
            outputLocations[0][i][0] * inputSize,
            outputLocations[0][i][3] * inputSize,
            outputLocations[0][i][2] * inputSize
        );
        int labelOffset = 1;
        if (outputClasses[0][i] < 0) outputClasses[0][i] = -1;
        if ((outputScores[0][i] > 1) || (outputScores[0][i] < 0))
outputScores[0][i] = 0.0f;

        recognitions.add(
            new Recognition(
                "" + i,
                labels.get((int) outputClasses[0][i] +
labelOffset),
                outputScores[0][i],
                detection));
    }
    return recognitions;
}

@Override
public void close() {
    interpreter.close();
    interpreter = null;
}
}

```

## Ingredients.class

```

public class Ingredients {
    private String name;

    public Ingredients(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }
}

```

## Classifier interface

```

import android.graphics.Bitmap;
import android.graphics.RectF;

import java.util.List;

public interface Classifier {

    class Recognition {

        private final String id;
        private final String title;
        private final Float conf;
        private RectF location;

        public Recognition(final String id, final String title, final Float

```

```

conf, final RectF location) {
    this.id = id;
    this.title = title;
    this.conf = conf;
    this.location = location;
}

public String getId() {
    return id;
}

public String getTitle() {
    return title;
}

public Float getConf() {
    return conf;
}

@Override
public String toString() {
    String res = "";
    if (title != null) {
        res += title;
    }
    return res.trim();
}

public RectF getLocation() {
    return new RectF(location);
}

public void setLocation(RectF location) {
    this.location = location;
}
}

List<Recognition> recognizeImage(Bitmap bitmap);

void close();
}

```

## RecipeInfo.class

```

import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.text.method.ScrollingMovementMethod;
import android.widget.LinearLayout;
import android.widget.TextView;

public class RecipeInfo extends AppCompatActivity {

    TextView name;
    TextView listIngredients;
    TextView description;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }
}

```



```

setContentView(R.layout.activity_recipe_info);

name = findViewById(R.id.info1);
listIngredients = findViewById(R.id.info2);
description = findViewById(R.id.info3);

description.setMovementMethod(new ScrollingMovementMethod());

Intent intent = getIntent();
String[] recipe = intent.getStringArrayExtra("recipe");

name.setText(recipe[0]);
listIngredients.setText(recipe[2]);
description.setText(recipe[1]);

    }
}

```

## Recipes.class

```

public class Recipes {
    private String name;
    private String description;
    private String listIngredients;

    public Recipes(String name, String description, String listIngredients) {
        this.name = name;
        this.description = description;
        this.listIngredients = listIngredients;
    }

    public String getName() {
        return name;
    }

    public String getDescription() {
        return description;
    }

    public String getListIngredients() {
        return listIngredients;
    }
}

```

## RecyclerViewAdapter.class

```

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.LinearLayout;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import java.util.List;

public class RecyclerViewAdapter extends

```

```

RecyclerView.Adapter<RecyclerView.ViewHolder> {

    private Context context;
    private List<Ingredients> ingredients;
    private RecyclerView.OnItemSelectedListener itemSelectedListener;

    public RecyclerView.Adapter(Context context, List<Ingredients> ingredients) {
        this.context = context;
        this.ingredients = ingredients;
        this.itemSelectedListener = (MainActivity) context;
    }

    @NonNull
    @Override
    public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int
viewType) {
        View view =
LayoutInflater.from(parent.getContext()).inflate(R.layout.item_layout,
parent, false);
        return new ViewHolder(view);
    }

    @Override
    public void onBindViewHolder(@NonNull ViewHolder holder, int position)
{
        holder.ingredientName.setText(ingredients.get(position).getName());
    }

    @Override
    public int getItemCount() {
        return ingredients.size();
    }

    class ViewHolder extends RecyclerView.ViewHolder implements
View.OnClickListener {

        TextView ingredientName;
        LinearLayout rootView;

        public ViewHolder(@NonNull View itemView) {
            super(itemView);
            ingredientName = itemView.findViewById(R.id.ingredient_name);
            rootView = itemView.findViewById(R.id.rootView);
            rootView.setOnClickListener(this);
        }

        @Override
        public void onClick(View v) {
            itemSelectedListener.onItemSelected(ingredients.get(getAdapterPosition()));
        }
    }
}

```

## RecyclerViewAdapterTwo.class

```

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.LinearLayout;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import java.util.List;

public class RecyclerViewAdapterTwo extends
RecyclerView.Adapter<RecyclerViewAdapterTwo.MyViewHolderTwo> {

    private Context context;
    private List<Recipes> recipes;
    private RecyclerViewItemTwoSelectedListener itemSelectedListener;

    public RecyclerViewAdapterTwo(Context context, List<Recipes> recipes) {
        this.context = context;
        this.recipes = recipes;
        this.itemSelectedListener = (RecipeActivity) context;
    }

    @NonNull
    @Override
    public MyViewHolderTwo onCreateViewHolder(@NonNull ViewGroup parent, int
viewType) {
        View view =
LayoutInflater.from(parent.getContext()).inflate(R.layout.item_two_layout,
parent, false);
        return new MyViewHolderTwo(view);
    }

    @Override
    public void onBindViewHolder(@NonNull MyViewHolderTwo holder, int
position) {
        holder.recipeName.setText(recipes.get(position).getName());

holder.ingredientsList.setText(recipes.get(position).getListIngredients());
    }

    @Override
    public int getItemCount() {
        return recipes.size();
    }

    class MyViewHolderTwo extends RecyclerView.ViewHolder implements
View.OnClickListener {

        TextView recipeName;
        TextView ingredientsList;
        LinearLayout rootViewTwo;

        public MyViewHolderTwo(@NonNull View itemView) {
            super(itemView);
            recipeName = itemView.findViewById(R.id.recipe_name);
            ingredientsList = itemView.findViewById(R.id.list_ingredients);
            rootViewTwo = itemView.findViewById(R.id.rootViewTwo);
            rootViewTwo.setOnClickListener(this);
        }
    }
}

```

```

    }

    @Override
    public void onClick(View v) {
        itemSelectedListener.onItemSelected(recipes.get(getAdapterPosition()));
    }
}

```

## RecyclerItemSelectedListener interface

```

public interface RecyclerItemSelectedListener {

    public void onItemSelected(Ingredients ingredient);
}

```

## RecyclerItemTwoSelectedListener interface

```

public interface RecyclerItemTwoSelectedListener {
    public void onItemSelected(Recipes recipe);
}

```

## ПРИЛОЖЕНИЕ Б

### (обязательное)

### ЛИСТИНГ ФАЙЛОВ ДЛЯ ДООБУЧЕНИЯ МОДЕЛИ

#### search\_bing\_api.py

```

from requests import exceptions
import argparse
import requests
import cv2
import os

ap = argparse.ArgumentParser()
ap.add_argument("-q", "--query", required=True)
ap.add_argument("-o", "--output", required=True)
args = vars(ap.parse_args())

API_KEY = "84bf9bf243fa44c39e292211fd583bd3"
MAX_RESULTS = 250
GROUP_SIZE = 50

URL = "https://api.cognitive.microsoft.com/bing/v7.0/images/search"

EXCEPTIONS = set([IOError, FileNotFoundError,
                  exceptions.RequestException, exceptions.HTTPError,
                  exceptions.ConnectionError, exceptions.Timeout])

term = args["query"]
headers = {"Ocp-Apim-Subscription-Key": API_KEY}
params = {"q": term, "offset": 0, "count": GROUP_SIZE, "imageType": "photo"}

search = requests.get(URL, headers=headers, params=params)
search.raise_for_status()

res = search.json()
num_res = min(res["totalEstimatedMatches"], MAX_RESULTS)

i = 0
for offset in range(0, num_res, GROUP_SIZE):
    params["offset"] = offset
    search = requests.get(URL, headers=headers, params=params)
    search.raise_for_status()
    res = search.json()
    for v in res["value"]:
        try:
            r = requests.get(v["contentUrl"], timeout=30)
            ext = v["contentUrl"][v["contentUrl"].rfind("."):]
```

```

        path_file = os.path.sep.join(
            [args["output"], "{}{}{}".format(term, str(i).zfill(7),
ext))]

        file = open(path_file, "wb")
        file.write(r.content)
        file.close()
    except Exception as e:
        if type(e) in EXCEPTIONS:
            continue
    image = cv2.imread(path_file)
    if image is None:
        os.remove(path_file)
        continue
    i += 1

```

### ssd\_mobilenet\_v1\_coco.config

```

model {
  ssd {
    num_classes: 18
    box_coder {
      faster_rcnn_box_coder {
        y_scale: 10.0
        x_scale: 10.0
        height_scale: 5.0
        width_scale: 5.0
      }
    }
    matcher {
      argmax_matcher {
        matched_threshold: 0.5
        unmatched_threshold: 0.5
        ignore_thresholds: false
        negatives_lower_than_unmatched: true
        force_match_for_each_row: true
      }
    }
    similarity_calculator {
      iou_similarity {
      }
    }
    anchor_generator {
      ssd_anchor_generator {
        num_layers: 6
        min_scale: 0.2
        max_scale: 0.95
        aspect_ratios: 1.0
        aspect_ratios: 2.0
        aspect_ratios: 0.5
        aspect_ratios: 3.0
        aspect_ratios: 0.3333
      }
    }
    image_resizer {
      fixed_shape_resizer {
        height: 300
        width: 300
      }
    }
  }
}

```

```

    }
  }
  box_predictor {
    convolutional_box_predictor {
      min_depth: 0
      max_depth: 0
      num_layers_before_predictor: 0
      use_dropout: false
      dropout_keep_probability: 0.8
      kernel_size: 1
      box_code_size: 4
      apply_sigmoid_to_scores: false
      conv_hyperparams {
        activation: RELU_6,
        regularizer {
          l2_regularizer {
            weight: 0.00004
          }
        }
      }
      initializer {
        truncated_normal_initializer {
          stddev: 0.03
          mean: 0.0
        }
      }
      batch_norm {
        train: true,
        scale: true,
        center: true,
        decay: 0.9997,
        epsilon: 0.001,
      }
    }
  }
}
feature_extractor {
  type: 'ssd_mobilenet_v1'
  min_depth: 16
  depth_multiplier: 1.0
  conv_hyperparams {
    activation: RELU_6,
    regularizer {
      l2_regularizer {
        weight: 0.00004
      }
    }
  }
  initializer {
    truncated_normal_initializer {
      stddev: 0.03
      mean: 0.0
    }
  }
  batch_norm {
    train: true,
    scale: true,
    center: true,
    decay: 0.9997,
    epsilon: 0.001,
  }
}
loss {
  classification_loss {

```

```

        weighted_sigmoid {
        }
    }
    localization_loss {
        weighted_smooth_l1 {
        }
    }
}
hard_example_miner {
    num_hard_examples: 3000
    iou_threshold: 0.99
    loss_type: CLASSIFICATION
    max_negatives_per_positive: 3
    min_negatives_per_image: 0
}
classification_weight: 1.0
localization_weight: 1.0
}
normalize_loss_by_num_matches: true
post_processing {
    batch_non_max_suppression {
        score_threshold: 1e-8
        iou_threshold: 0.6
        max_detections_per_class: 100
        max_total_detections: 100
    }
    score_converter: SIGMOID
}
}
}

train_config: {
    batch_size: 4
    optimizer {
        rms_prop_optimizer: {
            learning_rate: {
                exponential_decay_learning_rate {
                    initial_learning_rate: 0.004
                    decay_steps: 800720
                    decay_factor: 0.95
                }
            }
        }
        momentum_optimizer_value: 0.9
        decay: 0.9
        epsilon: 1.0
    }
}
fine_tune_checkpoint:
"C:/Users/Sasha/models/research/object_detection/training/model.ckpt-157711"
from_detection_checkpoint: true
num_steps: 160000
data_augmentation_options {
    random_horizontal_flip {
    }
}
data_augmentation_options {
    ssd_random_crop {
    }
}
}

train_input_reader: {
    tf_record_input_reader {
        input_path: "C:/Users/Sasha/models/research/object_detection/train.record"
    }
}

```



```

    }
    label_map_path:
"C:/Users/Sasha/models/research/object_detection/training/labelmap.pbtxt"
}

eval_config: {
  num_examples: 382
  # Note: The below line limits the evaluation process to 10 evaluations.
  # Remove the below line to evaluate indefinitely.
  max_evals: 10
}

eval_input_reader: {
  tf_record_input_reader {
    input_path: "C:/Users/Sasha/models/research/object_detection/test.record"
  }
  label_map_path:
"C:/Users/Sasha/models/research/object_detection/training/labelmap.pbtxt"
  shuffle: false
  num_readers: 1
}

```

## Labelmap.pbtxt

```

item {
  id: 1
  name: 'banana'
}

item {
  id: 2
  name: 'beef'
}

item {
  id: 3
  name: 'beet'
}

item {
  id: 4
  name: 'cabbage'
}

item {
  id: 5
  name: 'carrot'
}

item {
  id: 6
  name: 'cheese'
}

item {
  id: 7
  name: 'chicken'
}

item {
  id: 8
  name: 'cucumber'
}

```

```

}

item {
  id: 9
  name: 'curd'
}

item {
  id: 10
  name: 'egg'
}

item {
  id: 11
  name: 'eggplant'
}

item {
  id: 12
  name: 'garlic'
}

item {
  id: 13
  name: 'lemon'
}

item {
  id: 14
  name: 'milk'
}

item {
  id: 15
  name: 'onion'
}

item {
  id: 16
  name: 'potato'
}

item {
  id: 17
  name: 'sweet pepper'
}

item {
  id: 18
  name: 'tomato'
}

```

### **Generate\_tfrecords.py**

```

import os
import io
import pandas as pd
import tensorflow as tf

from PIL import Image
from object_detection.utils import dataset_util

```

```

from collections import namedtuple, OrderedDict

flags = tf.app.flags
flags.DEFINE_string('csv_input', '', 'Путь до CSV')
flags.DEFINE_string('image_dir', '', 'Путь до изображения')
flags.DEFINE_string('output_path', '', 'Путь к TFRecords')
FLAGS = flags.FLAGS

def class_text_to_int(row_label):
    if row_label == 'banana':
        return 1
    elif row_label == 'beef':
        return 2
    elif row_label == 'beet':
        return 3
    elif row_label == 'cabbage':
        return 4
    elif row_label == 'carrot':
        return 5
    elif row_label == 'cheese':
        return 6
    elif row_label == 'chicken':
        return 7
    elif row_label == 'cucumber':
        return 8
    elif row_label == 'curd':
        return 9
    elif row_label == 'egg':
        return 10
    elif row_label == 'eggplant':
        return 11
    elif row_label == 'garlic':
        return 12
    elif row_label == 'lemon':
        return 13
    elif row_label == 'milk':
        return 14
    elif row_label == 'onion':
        return 15
    elif row_label == 'potato':
        return 16
    elif row_label == 'sweet pepper':
        return 17
    elif row_label == 'tomato':
        return 18
    else:
        None

```

```

def split(df, group):
    data = namedtuple('data', ['filename', 'object'])
    gb = df.groupby(group)
    return [data(filename, gb.get_group(x)) for filename, x in
            zip(gb.groups.keys(), gb.groups)]

def create_tf_example(group, path):
    with tf.gfile.GFile(os.path.join(path, '{}'.format(group.filename)),
                        'rb') as fid:
        encoded_jpg = fid.read()
        encoded_jpg_io = io.BytesIO(encoded_jpg)
        image = Image.open(encoded_jpg_io)
        width, height = image.size

        filename = group.filename.encode('utf8')
        image_format = b'jpg'
        xmins = []
        xmaxs = []
        ymins = []
        ymaxs = []
        classes_text = []
        classes = []

    for index, row in group.object.iterrows():
        xmins.append(row['xmin'] / width)
        xmaxs.append(row['xmax'] / width)
        ymins.append(row['ymin'] / height)
        ymaxs.append(row['ymax'] / height)
        classes_text.append(row['class'].encode('utf8'))
        classes.append(class_text_to_int(row['class']))

    tf_example = tf.train.Example(features=tf.train.Features(feature={
        'image/height': dataset_util.int64_feature(height),
        'image/width': dataset_util.int64_feature(width),
        'image/filename': dataset_util.bytes_feature(filename),
        'image/source_id': dataset_util.bytes_feature(filename),
        'image/encoded': dataset_util.bytes_feature(encoded_jpg),
        'image/format': dataset_util.bytes_feature(image_format),
        'image/object/bbox/xmin': dataset_util.float_list_feature(xmins),
        'image/object/bbox/xmax': dataset_util.float_list_feature(xmaxs),
        'image/object/bbox/ymin': dataset_util.float_list_feature(ymins),
        'image/object/bbox/ymax': dataset_util.float_list_feature(ymaxs),
        'image/object/class/text':
            dataset_util.bytes_list_feature(classes_text),
        'image/object/class/label': dataset_util.int64_list_feature(classes),
    }))
    return tf_example

```

```
def main(_):
    writer = tf.python_io.TFRecordWriter(FLAGS.output_path)
    path = os.path.join(os.getcwd(), FLAGS.image_dir)
    examples = pd.read_csv(FLAGS.csv_input)
    grouped = split(examples, 'filename')
    for group in grouped:
        tf_example = create_tf_example(group, path)
        writer.write(tf_example.SerializeToString())

    writer.close()
    output_path = os.path.join(os.getcwd(), FLAGS.output_path)

if __name__ == '__main__':
    tf.app.run()
```

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ  
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИМЕНИ И.С. ТУРГЕНЕВА»

УДОСТОВЕРЯЮЩИЙ ЛИСТ № 150349/п

К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

на демонстрационный материал, представленный в электронном виде

Студента Шалимова Александра Сергеевича                      шифр 150349/п  
Институт приборостроения, автоматизации и информационных технологий  
Кафедра программной инженерии  
Направление 09.03.04 Программная инженерия  
Направленность (профиль) Промышленная разработка программного обеспечения

Наименование документа: Демонстрационные плакаты к выпускной  
квалификационной работе

Документ разработал:

Студент

Шалимов А. С.

Шалимов 21.06.19

Документ согласован:

Руководитель

Конюхова О. В.

Конюхова 21.06.19

Нормоконтроль

Ужаринский А.Ю.

Ужаринский 21.06.19

Документ утвержден:

Зав. кафедрой

Фролов А.И.

Фролов 21.06.19

ИНФОРМАЦИОННО-ПОИСКОВАЯ ХАРАКТЕРИСТИКА  
ДОКУМЕНТА НА ЭЛЕКТРОННОМ НОСИТЕЛЕ

Наименование		Характеристики документа на электронном носителе
группы атрибутов	атрибута	
1. Описание документа	Обозначение документа (идентификатор(ы) файла(ов))	\\Презентация Шалимов.ppt
	Наименование документа	Демонстрационные плакаты к выпускной квалификационной работе
	Класс документа	ЕСКД
	Вид документа	Оригинал документа на электронном носителе
	Аннотация	Демонстрационный материал, отображающий основные этапы выполнения выпускной квалификационной работы
	Использование документа	Операционная система Windows 10, Microsoft PowerPoint 2010
2. Даты и время	Дата и время копирования документа	20.06.19
	Дата создания документа	14.06.19
	Дата утверждения документа	19.06.19
3. Создатели	Автор	Шалимов А. С.
	Изготовитель	Шалимов А. С.
4. Внешние ссылки	Ссылки на другие документы	Удостоверяющий лист № 150349/п
5. Защита	Санкционирование	ОГУ имени И.С. Тургенева
	Классификация защиты	По законодательству РФ
6. Характеристики содержания	Объем информации документа	1 841 299 Б

7. Структура документа(ов)	Наименование плаката (слайда) №1	Титульный лист
	Наименование плаката (слайда) №2	Цель и задачи работы
	Наименование плаката (слайда) №3	Задача классификации и распознавания объектов
	Наименование плаката (слайда) №4	Используемая модель нейронной сети
	Наименование плаката (слайда) №5	Аналогичные системы и функциональные требования
	Наименование плаката (слайда) №6	Архитектура системы поиска рецептов на основе фотографий продуктов
	Наименование плаката (слайда) №7	Возможные действия пользователя системы поиска рецептов на основе фотографий продуктов
	Наименование плаката (слайда) №8	Основные классы системы поиска рецептов на основе фотографий продуктов
	Наименование плаката (слайда) №9	Алгоритм переобучения модели нейронной сети для системы поиска рецептов на основе фотографий продуктов
	Наименование плаката (слайда) №10	Алгоритм распознавания продуктов на изображении
	Наименование плаката (слайда) №11	Сравнение реализованной системы поиска рецептов на основе фотографий продуктов с аналогами
	Наименование плаката (слайда) №12	Пример работы системы поиска рецептов на основе фотографий продуктов





**АНТИПЛАГИАТ**  
ТВОРИТЕ СОБСТВЕННЫМ УМОМ

Орловский государственный  
университет имени И.С. Тургенева

## СПРАВКА

о результатах проверки текстового документа  
на наличие заимствований

Проверка выполнена в системе  
Антиплагиат.ВУЗ

Автор работы

Шалимов Александр Сергеевич

Подразделение

ИПАИТ, кафедра программной инженерии

Тип работы

Выпускная квалификационная работа

Название работы

вкр\_шалимов\_a\_c.docx

Название файла

вкр\_шалимов\_a\_c.docx

Процент заимствования

10,66%

Процент цитирования

3,76%

Процент оригинальности

85,58%

Дата проверки

12:45:39 21 июня 2019г.

Модули поиска

Модуль поиска ИПС "Адилет"; Модуль выделения библиографических записей; Сводная коллекция ЭБС; Коллекция РГБ; Цитирование; Модуль поиска переводных заимствований; Коллекция eLIBRARY.RU; Коллекция ГАРАНТ; Модуль поиска Интернет; Коллекция Медицина; Модуль поиска перефразирований eLIBRARY.RU; Модуль поиска перефразирований Интернет; Коллекция Патенты; Модуль поиска общеупотребительных выражений; Модуль поиска "ФГБОУ ВО ОГУ им. И.С.Тургенева"; Кольцо вузов

Работу проверил

Ужаринский Антон Юрьевич

ФИО проверяющего

Дата подписи

21.06.19

Подпись проверяющего

Чтобы убедиться  
в подлинности справки,  
используйте QR-код, который  
содержит ссылку на отчет.



Ответ на вопрос, является ли обнаруженное заимствование  
корректным, система оставляет на усмотрение проверяющего.  
Предоставленная информация не подлежит использованию  
в коммерческих целях.