Санкт-Петербургский политехнический университет Петра Великого
Институт компьютерных наук и технологий
Высшая школа программной инженерии

Работа допущена к защите
Директор ВШ ПИ
_____ П.Д. Дробинцев
«___» _____ 2019 г.

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

ПРИЛОЖЕНИЕ ПО РАСПОЗНАВАНИЮ ВИДОВ ЦВЕТОВ И ГРИБОВ МЕТОДОМ ГЛУБИННОГО ОБУЧЕНИЯ СВЕРТОЧНЫХ НЕЙРОННЫХ СЕТЕЙ

По направлению 09.03.04 «Программная инженерия» по образовательной программе
09.03.04_01 «Технология разработки и сопровождения качественного программного продукта»

Выполнила
студентка гр. 43504/4 (подпись) А.Ю. Фролова
Руководитель
старший преподаватель (подпись) И.В. Зайцев

Санкт-Петербург
2019 г.

Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Высшая школа программной инженерии

УТВЕРЖДАЮ

Директор ВШ ПИ

_____ П.Д. Дробинцев

«___» _____ 2019 г.

З А Д А Н И Е

на выпускную квалификационную работу бакалавра

Студентке группы 43504/4 Фроловой Алёне Юрьевне

1. Тема проекта (работы) Приложение по распознаванию видов цветов и грибов методом глубинного обучения сверточных нейронных сетей (утверждена распоряжением по ИКНТ от _____№_____)

2. Срок сдачи студентом оконченного проекта (работы)

3. Исходные данные к проекту (работе)

_____

_____

4. Содержание расчетно-пояснительной записки (перечень подлежащих разработке вопросов)

Актуальность темы, Описание и сравнение существующих решений, Описание алгоритма, Реализация алгоритма, Проектирование архитектуры мобильного приложения, Программная реализация мобильного приложения, Тестирование мобильного приложения, Анализ результатов

5. Перечень графического материала (с точным указанием обязательных чертежей)

_____

_____

6. Консультанты по проекту (с указанием относящихся к ним разделов проекта, работы)

_____

7. Дача выдачи задания « » _____ 20___г.

Руководитель _____ *Зайцев И.В.*

(подпись) *(фамилия, инициалы)*

Задание принял к исполнению _____ *Фролова А.Ю.*

(подпись) *(фамилия, инициалы)*

« » _____ 20___г.

(дата)

РЕФЕРАТ

МАШИННОЕ ОБУЧЕНИЕ, ГЛУБОКИЕ НЕЙРОННЫЕ СЕТИ, СВЁРТОЧНАЯ НЕЙРОННАЯ СЕТЬ, РАСПОЗНАВАНИЕ ОБЪЕКТОВ, БИБЛИОТЕКА ТЕНЗОРФЛОУ

Целью данной работы является разработка решения для распознавания цветов и грибов, которое будет использоваться в приложении Android, разработанном параллельно как другой проект. В случае грибов есть аспект безопасности – проводить различие между съедобными и ядовитыми грибами, чтобы уменьшить вероятность отравления. Для этой цели были рассмотрены различные подходы к обучению сверточной нейронной сети и различные существующие библиотеки для распознавания объектов.

Работа включает в себя описание, разработку и тестирование сверточной нейронной сети, предназначенной для распознавания видов цветов и грибов на основе файлов изображений. Финальные испытания показали высокую точность модели.

ABSTRACT

MACHINE LEARNING, DEEP NEURAL NETWORK, CONVOLUTIONAL NEURAL NETWORK, RECOGNITION OF OBJECTS, LIBRARY TENSORFLOW

The purpose of this work is to develop a solution for recognition of flowers and mushroom species to be used in the Android application developed in parallel as another project. In case of mushrooms there is a safety aspect - to discriminate between edible and poisonous mushrooms in order to decrease poisoning accidents. For this purpose different approaches to training of convolutional neural network and different existing libraries for object recognition have been considered to make the best choice.

The work includes a description, development and testing of a convolutional neural network intended for recognizing the types of flowers and mushrooms on the basis of image files. The final tests have showed high accuracy of the created model.

Содержание

Список иллюстраций

Список таблиц

Введение

Во всём мире существует более 250 тысяч видов грибов [1]. И каждый год ученые увеличивают это число. Изучая эту тему, всегда необходимо понимать, что в это число входят не только грибы, которые мы привыкли употреблять в пищу, но и ядовитые грибы, которые могут поражать растения, вызывать заболевания людей, а иногда и смерть. Ранее невероятная доступность грибов в наших лесах побуждала лесников ходить по грибы. Тем самым раньше грибники были более продвинуты в теме распознавания грибов. К сожалению, в современном мире люди утратили интерес к походам по грибы и обеспечили себе полное отсутствие знаний в этой области. Всё больше людей в наши дни из-за незнания правильного распознавания грибов являются жертвами отравления. Вот почему так важно уметь отличать ядовитые грибы от съедобных. Поэтому в качестве задачи было выбрано создание приложения, которое могло бы упростить изучение видов грибов для человека.

Раньше единственным доступным источником информации для людей в изучении грибов было использование справочников и книг. Однако с развитием технологий можно применить много новых методов и подходов, которые смогли бы упростить изучение многих видов грибов. Одним из таких подходов является глубинное обучение нейронных сетей.

Актуальность выбора данного метода можно подтвердить огромным количеством публикаций на эту тему, а также многими разработанными библиотеками для этой задачи.

Глубинное обучение нейронных сетей – это технология которая используется в разных сферах, но её главным назначением является об-

работка изображений. [2] В настоящее время доступно огромное количество изображений различных видов грибов, поэтому применение данной технологии для выбранной задачи обещало стать успешным.

Применение нейронных сетей для распознавания изображений можно разделить на несколько уровней. Главные задачи, решаемые нейронными сетями в приложении к изображениям, рассмотрены ниже:

- Распознавание объектов на изображении;
- Разделение изображения на различные отдельные объекты;
- Выделение объектов внимания (позволяет определять то, на что обратил бы внимание человек на данном изображении);

Решение задачи распознавания изображений является сложным процессом. В качестве объектов исследования являются цветы и грибы. Сложность данных объектов заключается в том, что данные объекты несут в себе ряд уникальных скрытых признаков. Для человека такие признаки легко распознаваемы, но для машины они являются сложными.

В ходе выполнения работы была получена модель нейронной сети, которая позволяет с большой вероятностью определять виды грибов и цветов.

Глава 1 Обзор Литературы

1.1 Нейронные сети

В области обучения нейронных сетей существует множество различных подходов. Быстрое развитие аппаратных и программных средств позволяют появляться новым методам и алгоритмам. Главной причиной роста популярности использования нейронных сетей является их востребованность в любой сфере деятельности. (описать)

Нейронные сети – раздел машинного обучения, который пытается сымитировать то, как учится человеческий мозг. Мозг получает импульсы от внешнего мира, выполняет обработку на входе, а затем генерирует сигналы на выход. Используя искусственную нейронную сеть, мы пытаемся повторить подобное поведение.

Искусственный нейрон представляет собой нелинейную функцию, аргументом которой является линейная комбинация всех входных сигналов. [3] Эта нелинейная функция называется активационной. Затем результат активационной функции посылается на выход нейрона. Объединение многих таких нейронов создает искусственную нейронную сеть.

1.2 Модель нейронной сети

Одной из первых моделей обучения нейронных сетей является метод перцептрон, который был предложен Фрэнком Розенблаттом в 1957 году. [4] Это метод получения на вход сигналов, каждый из которых имеет определенный вес. Каждый из входов суммируется на узле и передается в функцию активации, после вычисления которой получается выходной сигнал нейрона. На рис. 1.1 показано визуальное представление перцептрона.

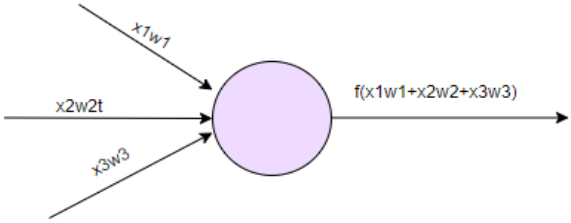

Рисунок 1.1 Модель искусственного нейрона

На рисунке 1.1:

- x_i – входной сигнал;
- w_i – вес входного сигнала;
- $f(s)$ – функция активации, где s – значение, полученное на сумматоре входных сигналов;

Необходимо определить сумму, которая будет передаваться функции активации:

$$\sum_{i=1}^{n} x_i w_i$$

1.3 Функция активации

Одним из важных процессов в искусственных нейронных сетях является функция активации. Именно она решает, активировать на данный момент нейрон или нет, проверить, является ли информация, которую получает нейрон, актуальной или её не следует принимать во внимание.

Функция активации нейрона представляет собой зависимость сигнала на выходе нейрона от суммы сигналов на его входах, то есть это некоторое преобразование, которое мы вычисляем на основе выходных сигналов.

Основными активационными функциями являются:

- Сигмоидальная активационная функция

$$f(s) = \frac{1}{1 + e^{-s}}$$

- Гиперболическая

$$f(s) = \frac{\sinh(s)}{\cosh(s)} = \frac{e^{s} - e^{-s}}{e^{s} + e^{-s}}$$

- Положительная линейная

$$f(s) = \max(0, s)$$

На рисунке 1.2-1.4 показаны графики этих функций.

1.3.1 Сигмоидальная функция активации

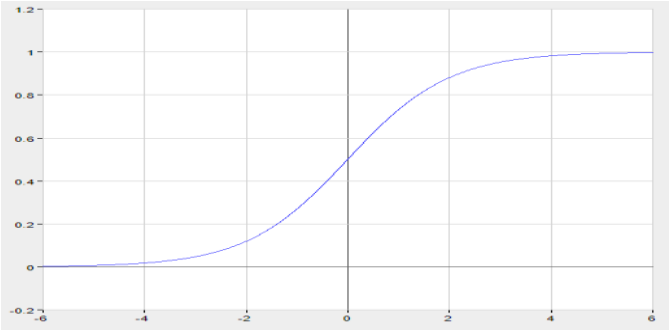

Рисунок 1.2 Сигмоидная(логистическая)функция активации

Сигмоидная функция является одной из самых частых активационных функций для задач классификации в нейронных сетях. Она стремится привести значения к одной из сторон кривой. Такое поведение позволяет находить четкие границы при предсказании. Преимуществом этой функции является фиксирование диапазона значений функции – [0,1], тогда как линейная функция изменяется в пределах – (-inf,inf). Такое свойство сигмоиды очень полезно, так как не приводит к ошибкам в случае больших значений.

1.3.2 Гиперболическая функция активации

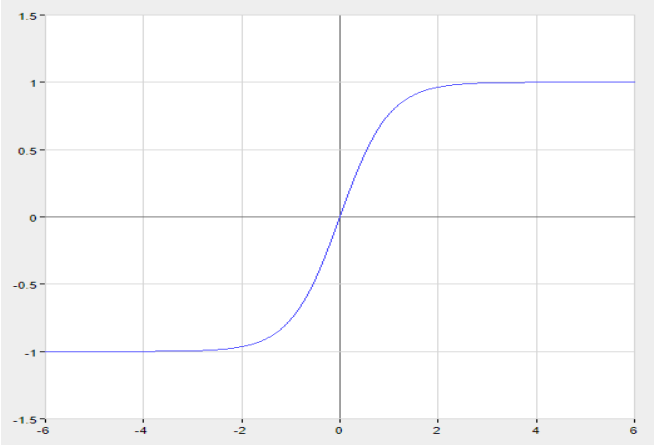

Рисунок 1.3 Гиперболистическая функция активации

Гиперболистическая функция активации похожа на сигмоидную функцию, она имеет те же характеристики. Она хорошо подходит для комбинации слоёв, и диапазон значений - (-1,1). Поэтому не стоит беспокоится, что активационная функция перегрузится от больших значений.

1.3.3 Положительная функция активации

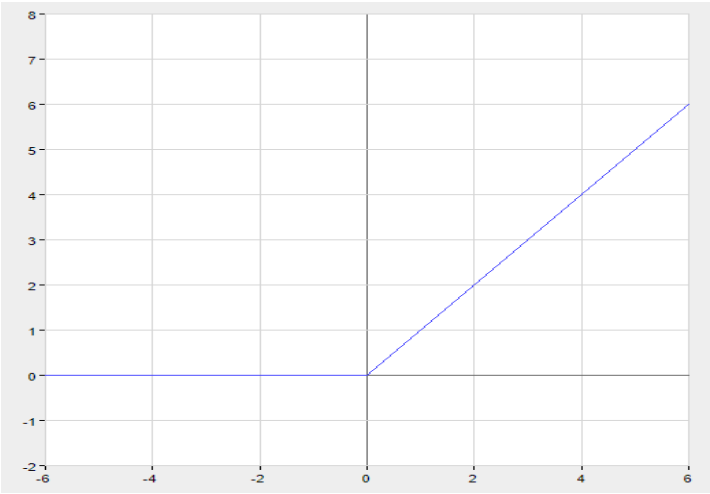

Рисунок 1.4 Положительная линейная функция активации

Положительная линейная функция позволяет некоторым нейронам не активироваться. Имея нейронную сеть со случайно инициализированными весами, в которой примерно 50% активаций равны 0, будет использоваться меньшее количество нейронов, а значит сеть будет становится легче.

Функция активации выбирается в зависимости от поставленной задачи. Для классификации изображений будет удобно использовать сигмоидную функцию для нескольких разных значений n. На выходе нейронной сети мы получим распределение вероятностей принадлежности входного образа одному из непересекающихся n классов.

1.4 Глубокое обучение нейронных сетей

Глубинное обучение нейронных сетей представляет собой методы машинного обучения, которые основаны на обучении представлениям (обучение по признакам), ориентированные на конкретные задачи. [5] Методы глубокого обучения имеют несколько уровней представления. В процессе обучения формируются слои выявления признаков, которые соответствуют различным уровням абстракции. При этом признаки организованы иерархически, признаки более высокого уровня являются производными от признаков более низкого уровня. Это и делает глубокое обучение глубоким. Каждый уровень описывает какой-то вид информации, упорядочивает её и передаёт на следующий уровень. Например, изображение гриба может иметь ряд признаков, таких как красная или белая шапка гриба.

Модель глубинной нейронной сети на рисунке 1.5. Между входным и выходным слоями находится множество скрытых слоёв. Чем их больше, тем глубже наша сеть.

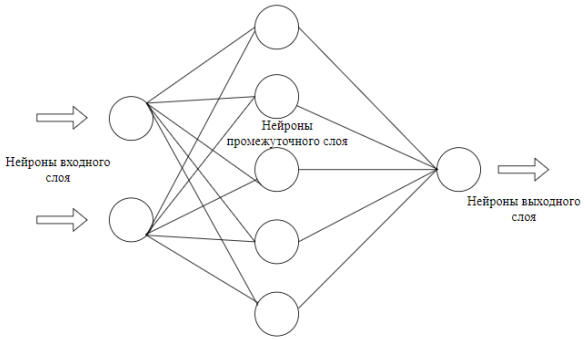

Рисунок 1.5 Модель глубинной нейронной сети

1.5 Сверточные нейронные сети

Сверточные нейронные сети являются классом глубоких нейронных сетей, наиболее часто используемых для анализа распознавания изображений. Сверточные сети выявляют определенные признаки на изображении, от самых незначительных, до более абстрактных деталей. [6] Такие признаки трудны для интерпретации настолько, что в практических системах не рекомендуется пытаться понять эти признаки. Они имеют три компонента:

- Сверточные слои

В качестве применения к изображению, указывают определенное количество сверточных фильтров, на выход получают функцию активации.

- Пулы слоёв

Уменьшают данные изображений, извлеченные сверточными нейронными слоями, уменьшают время обработки. Выбирают явные признаки, отбрасывают ненужные.

- Связанные слои

Выполняют классификацию объектов, извлеченных сверточными слоями. В плотном слое каждый узел в слое связан с каждым узлом в предыдущем слое.

1.6 Существующие решения

1.6.1 TensorFlow

TensorFlow представляет собой мощную библиотеку от Google для выполнения крупномасштабных численных вычислений. Одной из

задач, которую она выполняет, является реализация и обучение глубоких нейронных сетей. TensorFlow предоставляет примитивы для определения функций на тензорах и автоматического вычисления их производных. [7] Эта библиотека стала одним из самых популярных проектов машинного обучения, которая имеет открытый исходный код. В TensorFlow вычисления не выполняются сразу, вместо этого создаются графы, описывающие вычисления. Узлы графа представляют собой 25 математических операций, в то время как ребра графа представляют многомерные массивы данных (тензоры), соединенные этими узлами. TensorFlow можно рассматривать как гибкую модель программирования на основе потока данных для машинного обучения. Затем графы могут быть распределены для выполнения на нескольких устройствах.

Данная библиотека имеет следующие преимущества:

- TensorFlow полностью бесплатен и имеет открытый исходный код;
- Достаточно прост в использовании и поддерживает большинство практических задач;
- Гибкая архитектура и портативность позволяют развертывать вычисления на одном или нескольких процессорах, сервере или даже в составе продукта на смартфоне без переписывания кода с помощью единого API;
- Благодаря возможности работы на CPU (центральном процессоре) и GPU (графическом процессоре) он может быть внедрен в широком спектре продуктов, например, с распознаванием речи или образов;
- TensorFlow можно использовать для обучения и обслуживания моделей в режиме реального времени;

1.6.2 Сравнение с другими библиотеками

Другие доступные библиотеки, конкурирующие и сравнимые с TensorFlow:

1. Theano
2. Torch 26
3. Caffe
4. MXNet
5. Neon
6. CNTK

Рассмотрим основные особенности и характеристики этих инструментов:

- Использование языков программирования

Работа с глубоким обучением нейронных сетей подразумевает использование фреймворка, который поддерживает язык, с которым вы знакомы. Например, использование библиотек Caffe и Torch имеют привязку Python для своей кодовой базы. Но, если мы хотим использовать эти технологии, необходимо отлично знать основные языки этих библиотек – C++ или Lua соответственно. Для сравнения, TensorFlow и MXNet обладают отличной поддержкой нескольких языков, что позволяет использовать эту технологию, даже если вы не владеете C++.

- Руководство и учебные материалы

Библиотеки глубокого обучения в наше время имеют огромное количество, но несмотря на это ресурсы сильно различаются по качеству, количеству учебных пособий и материалов для начала работы. Theano, TensorFlow, Torch и MXNet имеют хорошие документации, которые легко понять и реализовать. Хотя CNTK от Microsoft и Nervana Neon от Intel являются мощными инструментами, найти учебные материалы для начального уровня по ним довольно сложно. Кроме того,

важно отметить активное участие сообщества на GitHub. Необходимо подчеркнуть, что TensorFlow – это лучший фреймворк по количеству руководств, учебных материалов, а также сообществу разработчиков и пользователей.

- Возможность моделирования сверточных нейронных сетей

Сверточные нейронные сети (CHH) используют для распознавания изображений. CHH состоят из набора различных слоев, которые преобразуют исходный объем данных в итоговые оценки предопределенных классов. Мы рассматриваем возможность моделирования технологий CHH для некоторых функций. Эти функции включают в себя пространство возможностей для определения моделей, наличия готовых слоев и инструментов, доступных для подключения этих слоев. Отличные возможности моделирования CNN есть у Torch и MXNet. Тем не менее, легкая способность TensorFlow опираться на модель Inception v3 делает эту технологию лучшей для возможностей моделирования CNN.

- Архитектура

Чтобы создавать и обучать новые модели в определенной библиотеке, важно иметь простой в использовании модульный интерфейс. TensorFlow, Torch и MXNet имеют простую и модульную архитектуру, которая значительно упрощает разработку. Для сравнения, такие фреймворки, как Caffe, требуют значительного объема работы для создания новых слоев. TensorFlow, в частности, легко отлаживать и отслеживать во время и после обучения, поскольку в TensorFlow для визуализации процесса включено графическое приложение TensorBoard.

- Скорость

Torch и Neon имеют лучшую документально подтвержденную производительность для сверточных нейронных сетей. Для большинства тестов скорость TensorFlow также была весьма сопоставима, в то время как Caffe и Theano были немного хуже. [8]

- Совместимость с Keras

Keras – библиотека высокого уровня для быстрого прототипирования глубокого обучения. Это отличный инструмент для того, чтобы использовать все преимущества глубокого обучения. При этом Keras спроектирован так, чтобы быть компактным, модульным и расширяемым фреймворком. До недавнего времени Keras поддерживался всего двумя библиотеками: TensorFlow и Theano, однако с недавнего времени к ним присоединилась и CNTK. [9] Сравнительный анализ представлен в таблице 1.1.

Таблица 1.1

	Theano	Torch	Caffe	**TensorFlow**
Языки	Python C++	Lua Python	C++	Python
Руководство и учебные материалы	среднее количество	малое количество	малое количество	большое количество
Возможность моделирования глубоких нейронных сетей	Имеет	И Не имеет	Имеет	Имеет

Архитектура как простата в использовании	сложная	средняя	сложная	простая
Скорость	Средняя	Быстрая	Низкая	Средняя
Совместимость с keras	Имеет	Не имеет	Не имеет	Имеет
Открытый код	Имеет	Имеет	Не имеет	Имеет

Сравнение существующих решений

Глава 2 Концепция

Концепция данной работы предполагает решение задачи распознавания изображений цветов и грибов. Данную тему выбрали в качестве объекта исследования, так как распознавание изображений является одной из самых популярных среди машинного обучения.

Основная идея реализуемого подхода – обучить модель таким образом, чтобы на вход приложения поступала информация (изображение или явные признаки объектов исследования) определенного вида гриба, или цветка, а на выходе имелась информация с его распознаванием (вид, тип). Соответственно, при проверке модели на тестовой выборке ожидается, что алгоритм сверточной нейронной сети сможет применить выявленные признаки грибов и цветов для получения точного результата распознавания.

2.1 Фреймворк TensorFlow

Любое вычисление в TensorFlow представляет собой граф потока данных (или граф вычислений). Этот граф является моделью, описывающей как, будут выполняться вычисления. Он состоит из плэйсхолдеров (заполнитель), переменных и операций. Внутри графа производится вычисления тензоров (многомерных массивов), которые представляют собой число или вектор. [10]

Все графы выполняются в сессиях, модуль – tf.Session. Существуют два типа сессий: обычные и интерактивные – tf.InteractiveSession, которая больше подходит для выполнения в консоли. Сессия хранит состояние переменных и очередей. Явное создание сессий и графов гарантирует надлежащее освобождение ресурсов. [11]

В графе каждая вершина имеет от нуля и больше входов и от нуля до больше выходов, они представляют собой реализацию операции. Тензоры являются рёбрами графа, имеют произвольный размер, тип массива указывается во время построения графа.

В задаче нейронной сети, параметры модели хранят тензоры в переменных, которые обновляются на каждом шаге обучения.

2.2 Модуль создания свёрточной нейронной сети

TensorFlow позволяет использовать готовые модули для разработки, обучения и тестирования нейронных сетей. В нашем случае распознавания изображений будет выбран модуль tf.layers. Он предоставляет методы, которые облегчают создание сверточных слоев, добавление функций активации и применение регуляризации выпадения.

Методы для создания каждого из трех типов слоёв:

- Conv2d() – создаёт двумерный сверточный слой. Принимает количество фильтров, размер ядра фильтра и функцию активации в качестве аргументов.
- Max_pooling2d() – создает двумерный слой пула, используя алгоритм максимального пула. Принимает размер пула и шаг в качестве аргументов.
- Dense() - строит плотный слой. Принимает количество нейронов и функцию активации в качестве аргументов.

Каждый из этих методов принимает тензор в качестве входных данных и возвращает преобразованный тензор в качестве выходных данных.

2.3 Процесс переобучения модели

На начальном этапе анализируются все изображения и рассчитываются bottleneck (узкое место) для каждого из них. Узкое место является неофициальным термином которое часто используется для слоя непосредственно перед окончательным выходным слоем , который фактически делает классификацию. Модели TensorFlow состоят из множества слоёв, которые наложены друг на друга. Эти слои предварительно обучены и имеют информацию, которая поможет классифицировать изображения. Поэтому мы будем тренировать только последний слой, в то время как предыдущие слои сохранят своё уже обученное состояние. На модель переобучения можно посмотреть на рисунке

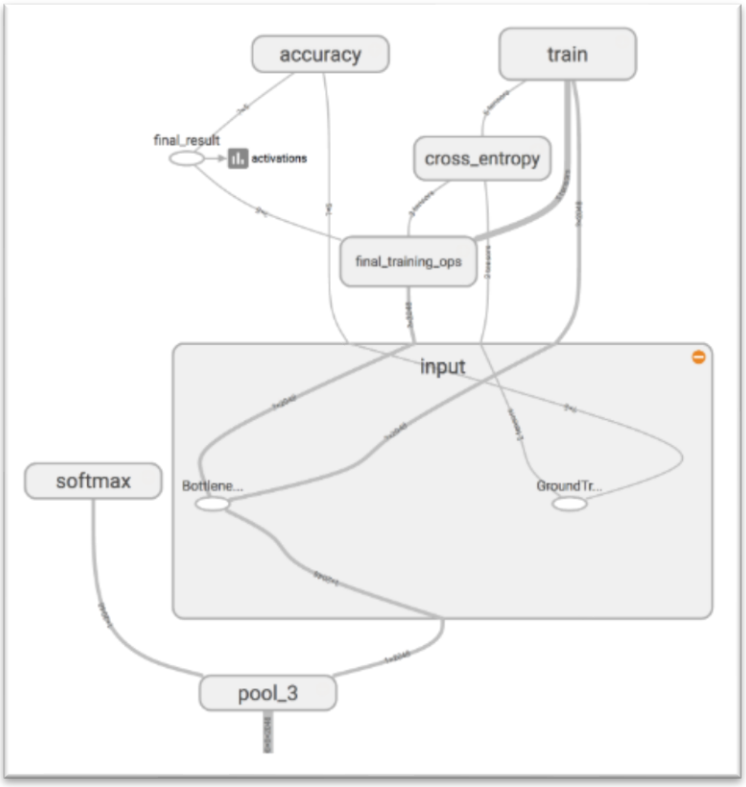

Рисунок 2.1 Структура модели переобучения

На приведенном выше рисунке узел, обозначенный “softmax”, с левой стороны является выходным слоем исходной модели. В то время как все узлы справа от “softmax” были добавлены скриптом переподготовки. Узкое место не было использовано в данной схеме, потому что лишний уровень замедляет работу сети. Мы используем термин “узкое место”, потому что вблизи выхода представление гораздо компактнее, чем в основной части сети.

Каждое изображение многократно используется во время тренировки. Расчет слоев за узким местом для каждого изображения занимает значительное количество времени. Поскольку эти нижние уровни сети не модифицируются, их выходы могут быть кэшированы и использованы повторно.

Таким образом, скрипт запускает постоянную часть сети, все, что находится ниже узла, помеченного bottlenecks, и кэширует результаты. Таким образом, если мы повторно запустим сценарий, он будет использован повторно, но не придется ждать много времени снова.

2.4 Обучение

Как только сценарий завершает генерацию всех файлов с узкими местами, начинается фактическое обучение на последнем уровне сети.

По умолчанию этот скрипт выполняет 4000 шагов обучения. Каждый шаг выбирает 10 изображений случайным образом из обучающего набора, находит их узкие места в кэше и передает их в последний слой для получения прогнозов. Затем эти прогнозы сравниваются с фактическими метками, и результаты этого сравнения используются для обновления весов.

Во время тренировки можно увидеть серию пошаговых выходов, каждый из которых показывает точность обучения:

- Точность обучения показывает процент изображений, используемых в текущей обучающей выборке, которые были помечены с правильным классом.

- Точность проверки — это точность (процент правильно помеченных изображений) для случайно выбранной группы изображений из другого набора.
- Кросс-энтропия — это функция потерь, которая дает представление о том, насколько хорошо прогрессирует процесс обучения. (Чем ниже цифры, тем лучше.)

На рисунках ниже показан пример прогресса точности модели и перекрестной энтропии в процессе обучения. Когда наша модель завершит создание файлов узких мест, мы сможем проверить прогресс нашей модели в специальном приложении TensorBoard.

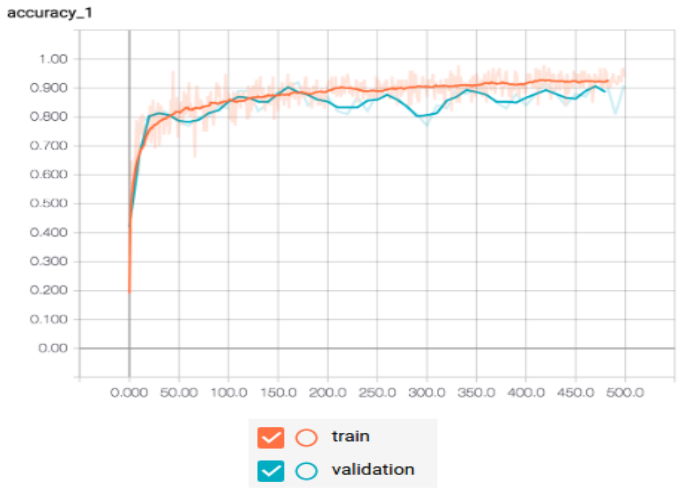

Рисунок 2 2 Диаграмма модели обучения

На рисунке показана точность (ось Y) как функция прогресса тренировки (ось X)

Показаны две линии. Оранжевая линия показывает точность модели на тренировочных данных. При этом синяя линия показывает точность на тестовом наборе (который не использовался для тренировок). Это гораздо лучший показатель истинной производительности сети. Если точность обучения продолжает расти, в то время как точность проверки уменьшается, то модель называется «переоснащением». Переоснащение — это когда модель начинает запоминать тренировочный набор вместо понимания общих закономерностей в данных.

По мере того, как процесс продолжается, мы должны увидеть улучшение точности. После завершения всех этапов обучения сценарий выполняет окончательную оценку точности теста для набора изображений, которые хранятся отдельно от изображений для обучения и проверки. Эта тестовая оценка дает наилучшую оценку того, как обученная модель будет выполнять задачу классификации.

Мы должны увидеть значение точности от 85% до 99%, хотя точное значение будет варьироваться каждый раз, так как в процессе обучения есть случайность. Это числовое значение указывает процентное соотношение изображений в тестовом наборе, которым присваивается правильная метка после того, как модель полностью обучена.

Глава 3 Реализация

Реализация поставленной задачи распознавания видов цветов и грибов предполагает использование определенных программных средств. Одним из самых популярных языков программирования в области создания нейронных сетей является Python. Этот язык был выбран для решения поставленной задачи, так как он позволяет не только разработать необходимый алгоритм распознавания цветов и грибов, но и обучить модели, в конечном итоге протестировать.

Преимущества языка Python:

- Прост в использовании;

Python обеспечивает большую легкость в отношении написания и выполнения кода.

- Поддержка готовых библиотек;

Python содержит множество библиотек, в зависимости от проекта.

- Кроссплатформенность;

Кроссплатформенность позволяет экономить время разработки для тестирования на разных платформах и переноса кода.

- Популярность;

Постановка задачи предполагает не только распознавание видов цветов и грибов, но и съедобность или ядовитость гриба. Таким образом была сделана схема реализации проекта, она представлена на рисунке 3.1.

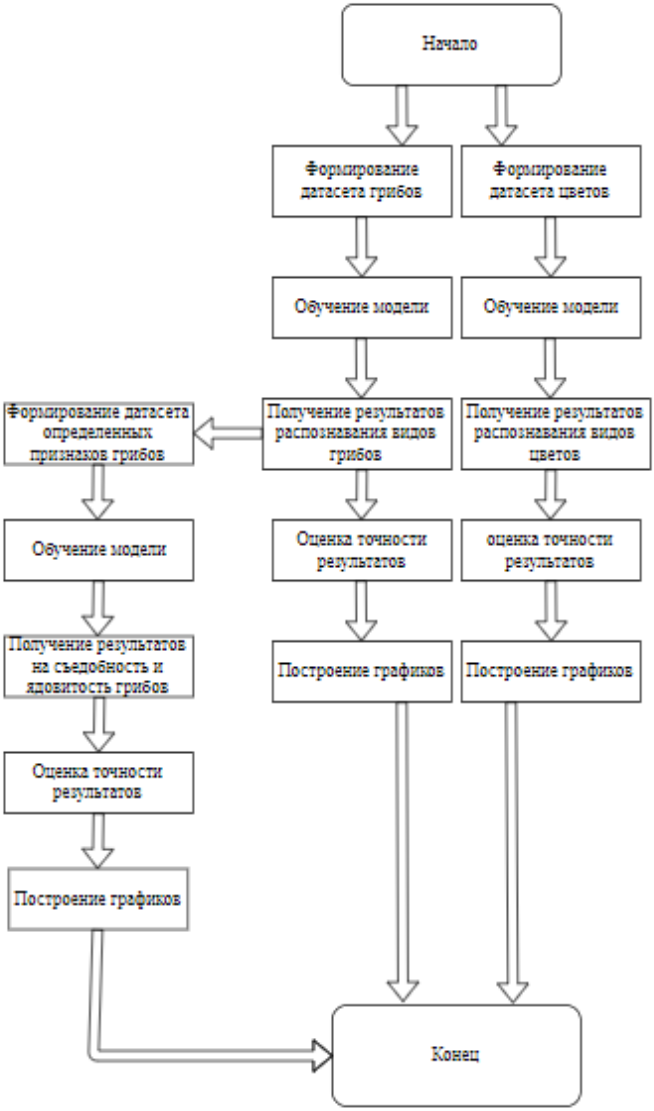

Рисунок 3.1 Схема реализации проекта

Рисунок 3.1 Схема реализации проекта

Для работы над проектом было выбрано несколько библиотек:

- TensorFlow
- Numpy
- Pandas

3.1 Формирование данных грибов и цветов

До начала работы, необходимо подготовить обучающие входные данные. Для распознавания видов цветов и грибов было создано несколько каталогов, которые включают изображения, описывающие каждые из них.

Набор данных для распознавания цветов содержит 5 каталогов с видами:

- ромашка
- одуванчик
- роза
- подсолнух
- тюльпан

Набор данных для распознавания грибов содержит 8 каталогов с видами:

- белый гриб
- подберезовик
- лисичка
- подосиновик
- бледная поганка
- мухомор белый

- мухомор красный
- рядовка ядовитая

Наборы данных содержат по 1000 изображений.

3.2 Обучение моделей цветов и грибов

3.2.1 Реализация классификатора

Для создания классификатора была использована библиотека TensorFlow. Одной из возможностей фреймворка является использование концепции трансферного обучения для сокращения времени и сложности обучения путем перепрофилирования предварительно подготовленной модели.

Для обучения воспользуемся файлом retrain, который предоставляет библиотека TensorFlow. Этот скрипт загружает предварительно подготовленную модель, удаляет старый верхний слой и обучает новый на загруженных нами изображениях. Введем необходимые атрибуты для выполнения метода.

```
python -m scripts.retrain \
  --how_many_training_steps = 500 \
  --output_graph = bkr/retrained_graph.pb \
  --output_labels = bkr/retrained_labels.txt \
  --image_dir = bkr/flower_photos
```

Рисунок 3.2 Запуск обучения распознавания цветов

```
python -m scripts.retrain \
  --how_many_training_steps = 500 \
  --output_graph = bkr/retrained_graph.pb \
  --output_labels = bkr/retrained_labels.txt \
  --image_dir = bkr/mushroom_photos
```

Рисунок 3.3 Запуск обучения распознавания грибов

где

- how_many_training_steps – количество тренировочных шагов;
- output_graph – полученная модель;
- output_labels - метки объектов модели;
- image_dir – путь к директории с подкаталогами входных данных;

Трансферное обучение заключается в том, что более низкие уровни, которые были обучены различать некоторые объекты, могут быть повторно использованы для многих задач распознавания без каких-либо изменений. Так же по этой причине для выбора распознавания изображений было выбрано два объекта исследований.

В качестве функции активации последнего слоя глубинной нейронной сети используется Softmax. Функцией потерь выбирается перекрестная энтропия.

После обучения на выход получаем 4 файла:

- retrained_flowers_graph.pb – это граф, который содержит версию нейронной сети с последним уровнем, прошедшим обучение по пяти категориям цветов;

- retrained_mushroom_graph.pb – это граф, который содержит версию нейронной сети с последним уровнем, прошедшим обучение по восьми категориям грибов (содержат как съедобные, так и ядовитые);
- retrained_flowers_labels.txt – файл, который содержит названия видов цветов;
- retrained_mushroom_labels.txt – файл, который содержит названия видов грибов;

3.2.2 Получение результатов распознавания видов цветов и грибов

После 500 тренировочных шагов можно протестировать полученные модели на изображениях, взятых не из обучающего набора.

Для тестирования полученных нейронных сетей, воспользуемся файлом label_image, который предоставляется библиотекой TensorFlow. Для проверки результатов были выбраны изображения, не находившиеся в обучающей выборке.

Рисунок 3.4 Тестовое изображение цветка

Назвав файл daisy.jpg, прогоним его по нашему графу и получим результат:

```
python -m scripts.label_image \
    --graph = tf_files/retrained_flowers_graph.pb \
    --image = tf_files/test/daisy.jpg
```

Рисунок 3.5 Запуск теста цветов

```
daisy (score = 0.99071)
```

Рисунок 3.6 Результат теста с цветком

Для сравнения результатов с другими видами цветов, используем файл retrained_flowers_labels.txt, получим:

```
daisy (score = 0.99071)
sunflowers (score = 0.00595)
dandelion (score = 0.00252)
roses (score = 0.00049)
tulips (score = 0.00032)
```

Рисунок 3.7

Как видно из результатов обучение является эффективным.

Рисунок 3.8 Тестовое изображение гриба

Назвав файл white_mushroom.jpg, прогоним его по нашему графу и получим результат:

```
python -m scripts.label_image \
    --graph=tf_files/retrained_mushroom_graph.pb  \
    --image=tf_files/tets/white_mushroom.jpg
```

Рисунок 3.9 Запуск теста грибов

```
white_mushroom (score = 0.99143)
```

Рисунок 3 10 Результат теста с грибом

Для сравнения результатов с другими видами цветов, используем файл retrained_mushroom_labels.txt, получим:

```
white_mushroom (score = 0.99143)
birch_bolete (score = 0.00423)
chanterelle (score = 0.00011)
orange-cap boletus (score = 0.00039)
death cap (score = 0.00059)
amanita_white (score = 0.00024)
amanita_red (score = 0.00022)
poisonous (score = 0.00031)
```

Рисунок 3.11

3.2.3 Создание датасета грибов

Используя готовый файл retrained_mushroom_labels.txt (хранит в себе названия грибов) соберём новый файл data.csv, который будет определен по специальным признакам:

Таблица 2.1

1. cap-shape: bell=b, conical=c, convex=x, flat=f, knobbed=k, sunken=s
2. cap-surface: fibrous=f, grooves=g, scaly=y, smooth=s
3. cap-color: brown=n, buff=b, cinnamon=c, green=r, pink=p, purple=u, red=e, white=w, yellow=y
4. bruises: bruises=t, no=f

5. odor: almond=a ,anise=l, creosote=c, fishy=y, foul=f, musty=m, none=n, pungent=p, spicy=s
6. gill-attachment: attached=a, descending=d, free=f, notched=n
7. gill-spacing: close=c, crowded=w, distant=d
8. gill-size: broad=b, narrow=n
9. gill-color: black=k, brown=n, buff=b, chocolate=h, gray=g, green=r, orange=o, pink=p, purple=u, red=e, white=w, yellow=y
10. stalk-shape: enlarging=e, tapering=t
11. stalk-root: bulbous=b, club=c, cup=u, equal=e, rhizomorphs=z, rooted=r, missing=?
12. stalk-surface-above-ring: fibrous=f, scaly=y, silky=k, smooth=s
13. stalk-surface-below-ring: fibrous=f, scaly=y, silky=k, smooth=s
14. stalk-color-above-ring: brown=n, buff=b, cinnamon=c, gray=g, orange=o, pink=p, red=e, white=w, yellow=y
15. stalk-color-below-ring: brown=n, buff=b, cinnamon=c, gray=g, orange=o, pink=p, red=e, white=w, yellow=y
16. veil-type: partial=p, universal=u
17. veil-color: brown=n, orange=o, white=w, yellow=y
18. ring-number: none=n, one=o, two=t
19. ring-type: cobwebby=c, evanescent=e, flaring=f, large=l, none=n, pendant=p, sheathing=s, zone=z
20. spore-print-color: black=k, brown=n, buff=b, chocolate=h, green=r, orange=o, purple=u, white=w, yellow=y
21. population: abundant=a, clustered=c, numerous=n, scattered=s, several=v, solitary=y
22. habitat: grasses=g, leaves=l, meadows=m, paths=p, urban=u, waste=w, woods=d

Признаки грибов

Файл data.csv был собран специальным методом TensorFlow, при распознавании изображений грибов на виды.

3.3 Определение ядовитости и съедобности грибов

Главная задача проекта, это определить, съедобность или ядовитость грибов. Основной проблемой является то, что параметры грибов являются категорийными, а не числовыми. Например, параметр “форма ножек” может иметь значение клубневидное(округлое), булавовидное (с резким сужением внизу) или усечённое(полушаровидное). Таким образом, был собран набор данных грибов из полученных меток. Каждый вид грибов идентифицируется как съедобный, ядовитый, или гриб неизвестного вида, который выбирать не рекомендуется (класс был объединён ядовитым).

3.3.1 Подготовка данных

Для обучения и тестирования нейронной сети загружать данные будем из файла csv (файл, разделенный запятыми). В файл dataset.csv помещаем 8124 строки и 22 колонки. Так как некоторые фотографии не имели явных нужных признаков, то некоторые фотографии были не включены в сбор файла data.csv. [12] Каждая строка представляет собой один гриб, а в каждой колонке один из 22-х параметров гриба в виде специального символа, который является сокращением параметра от целого слова.

Далее необходимо использовать файл dataset.csv в нашей программе. Для этого была подключена и использована библиотека Pandas.

Чтобы считать файл с параметрами грибов, будем использовать функцию read_csv. Для этого использовали: df = pd.read_csv(‘dataset.csv’). Затем для правильного использования передачи параметров, был сформирован массив с элементами названий параметров:
attribute = [‘class’, ‘cap_shape’, ‘cap_surface’, ‘cap_color’, ‘bruises’, ‘odor’, ‘gill_attachment’, ‘gill_spacing’, ‘gill_size’, ‘gill_color’, ‘stalk_shape’, ‘stalk_root’, ‘stalk_surface_above_ring’, ‘stalk_surface_belonw_ring’, ‘stalk_color_above_ring’, ‘stalk_color_below_ring’, ‘veil_type’, ‘veil_color’, ‘ring_number’, ‘ring_type’, ‘spore_print_color’, ‘population’, ‘habitat’].
Для удобства использования алгоритма, в нашем наборе данных необходимо заменить символьный параметр съедобности и ядовитости на цифровой, то есть ‘p’ на 0, и ‘e’ на 1. Была использована функция Python – replace (): df = df[‘class’].replace(‘p’, 0), df = df[‘class’].replace(‘e’, 1).
Так как изначально параметры грибов обозначаются в наборе данных в символьном виде, необходимо конвертировать их в цифры, потому что TensorFlow умеет работать только с цифрами. Библиотека Pandas имеет функцию – get_dummies, которая сконвертирует данные в цифры:
df = pd.get_dummies(df, columns = attribute[1:])
Чтобы выполнить обучение модели, необходимо разделить набор данных на два набора: тренировочный и набор для тестирования нейронной сети. Разделив изначальный набор данных, записали тренировочный и калибровочный наборы в два новых файла csv: train.csv, test.csv. [13]

3.3.2 Обучение модели

Для начала необходимо сформировать данные в TensorFlow, то есть отдать их на обучение.

Это делается с помощью функции load_csv_with_header(), которую предоставляет выбранный фреймворк. Эта функция принимает в качестве аргументов: target_dtype, котораый является типом предсказываемых в итоге данных, features_dtype – параметр, где задается тип принимаемых на обучение параметров, target_column -–индекс колонки с параметром, который нейронная сеть должна предсказать.

Далее создаем объект классификатора TensorFlow, объект класса, который занимается предсказанием результата.

```
Classifier = tf.contrib.learn.DNNClassifier(
feature_columns = feature_columns,
hidden_units = [10, 20, 10],
n_classes = 2,
model_dir = “”
)
```

Первым параметром являются параметры грибов. Вторым параметром является количество нейронов в каждом слое нейронной сети. Очень сложно подобрать правильный подбор, данные цифры были выбраны спустя многих тренировок. Третий параметр отвечает за количество классов для предсказания, у нас их два, съедобный и ядовитый. Последний параметр – это путь, в которой будет сохранена натренированная модель нейронной сети. В будущем будет использоваться для предсказания результатов, чтобы не обучать каждый раз.

Чтобы тренировать нейронную сеть, создали две функции. Каждая функция выдает свой набор входных данных – для тренировки и для калибровки.

Тренируем сеть с помощью функции fit():

```
Classifier.fit(input_fn=get_train_inputs, steps=2000)
```

Первый параметр принимает на вход сформированные данные, второй – количество шагов тренировки. [13]

Далее калибруем натренированную сеть. Это делается с помощью сформированного калибровочного набора данных.

```
accuracy_score = classifier.evaluate
(input_fn=get_test_inputs, steps=1)["accuracy"]
print("\n prediction accuracy: {0:f}\n".format(accuracy_score)
```

Рисунок 3.12

Результатом работы будет точность будущих предсказаний нейронной сети.

3.3.3 Получение результатов предсказания съедобности грибов.

Для проверки результатов выберем два совершенно новых гриба, которые не присутствовали ни в тренировочном, ни в калибровочном наборах. В функцию test_mushroom(), был добавлен массив с новыми данными:

```
def new_samples():
    return np.array([[0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
                      1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0,
                      0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0,
                      0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0,
                      0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
                      0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1,
                      0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                      1, 0, 1, 0, 0, 0, 0],
                     [0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0,
                      0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1,
                      0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0,
                      0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0,
                      0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
                      0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0,
                      0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1,
                      0, 0, 0, 0, 0, 0, 1]], dtype=np.int)
```

Рисунок 3.13 Тест на проверку ядовитости грибов

Далее используя функцию predict (), определим предсказание съедобности грибов:

```
predictions = list(classifier.predict(input_fn=new_samples))
print("Predictions edible test mushrooms:    {}\n"
      "Result evaluation:    {}\n"
      .format(predictions, score))
```

Рисунок 3.14

Результатом будет следующее:

```
Predictions edible test mushrooms: [0,1]
Predictions edible test mushrooms: [0.8971,0.9014]
```

Рисунок 3.15 Результат теста ядовитости грибов

Это означает, что первый гриб ядовитый, с точностью 0.8971, а второй гриб съедобный, с точностью 0.9014, то есть 89% и 90% соответственно. Таким образом можно делать предсказания на основе любых данных. [13]

3.4 Распознавание изображений в приложении

Перед тем как использовать полученные обученные модели цветов и грибов в приложении, необходимо удалить из них неподдерживаемые слои декодирования JPEG. Для этого выполним для каждого графа скрипт библиотеки TensorFlow strip_unused.py:

```
bazel build tensorflow/python/tools:strip_unused
bazel-bin/tensorflow/python/tools/strip_unused
    --input_graph=retrainde_flowers_graph.pb
    --output_graph=/tmp/flowers_graph.pb
    --input_node_names="Mul"
    --output_node_names="final_result"
    --input_binary=true
```

Рисунок 3.16 Готовый граф цветов

```
bazel build tensorflow/python/tools:strip_unused
bazel-bin/tensorflow/python/tools/strip_unused
    --input_graph=retrainde_mushroom_graph.pb
    --output_graph=/tmp/mushroom_graph.pb
    --input_node_names="Mul"
    --output_node_names="final_result"
    --input_binary=true
```

Рисунок 3.17 Готовый граф грибов

Были получены два файла: flowers_graph.pb и mushroom_graph.pb, в расширении pb, метки retrained_labels, в текстовом формате. Эти файлы помещаются в каталоге assets для дальнейшей работы с ними. В классах, приложения на Android Studio, ClassifierFlowersActivity и ClassifierMushroomActivity объявляем и передаём их пути. Там же указываем настройки для моделей:

```
INPUT_SIZE = 300;
IMAGE_MEAN = 128;
IMAGE_STD = 128;
INPUT_NAME = “Mul”
OUTPUT_NAME = “final_result”;
MODEL_FILE = “/android_asset/flower_graph.pb”;
LABEL_FILE = "/android_asset/retrained_labels.txt";
```

Рисунок 3.18

```
INPUT_SIZE = 300;
IMAGE_MEAN = 128;
IMAGE_STD = 128;
INPUT_NAME = “Mul”
OUTPUT_NAME = “final_result”;
MODEL_FILE = “/android_asset/mushroom_graph.pb”;
LABEL_FILE = "/android_asset/retrained_labels.txt";
```

Рисунок 3.19

- input_size – размер входного изображения;
- image_mean – среднее значение;
- image_std – стандартное отклонение;
- input_name – метки входных узлов;

- output_name – метки выходных узлов;

Кадры, сделанные с камеры телефона, преобразуются ко входному размеру и указанные параметры передаются в класс TensorFlowImageClasssifier для классификации.

Фреймворк TensorFlow обладает рядом уникальных встроенных библиотек, для взаимодействия с моделями TensorFlow на Java. Необходимо включить в grade-файл зависимость:

org.tensorflow:tensorflow-android и использовать специальный класс: TensorFlowInferenceInterface в коде проекта.

c.inferenceInterface = new TensorFlowInferenceInterface(assetManager, modelFilename);

Основные моменты метода recognizeImage:

```
@Override
public List<Recognition> recognizeImage(final float[] pixels) {
 // Копируем входные данные в TensorFlow.
 inferenceInterface.feed(inputName, pixels, new long[]{inputSiz
     * inputSize});
 // Запустим вызов вывода
 inferenceInterface.run(outputNames);
 // Скопируем выходной тензор обратно в выходной массив.
 inferenceInterface.fetch(outputName, outputs);
 // Найдем лучшую классификацию.
 for (int i = 0; i < outputs.length; ++i) {
 <…>
 }
 return recognitions;
}
```

Рисунок 3.20

Подаём данные пикселей, запускаем классификатор, затем извлекаем выходные данные. После этого выходы сортируются для получения большей уверенности и предоставляются пользователю. [14]

Глава 4 Анализ результатов

Реализация модели для распознавания видов цветов и грибов позволила получить численные результаты. Во время обучения, мы проследили за тем, что система предсказывала на каждом шаге обучения. Для этого было необходимо запустить:

tensorboard --logdir=log_simple_stats, затем на странице localhost:6006/#events, вывелся интерактивный график, рисунок 4.1

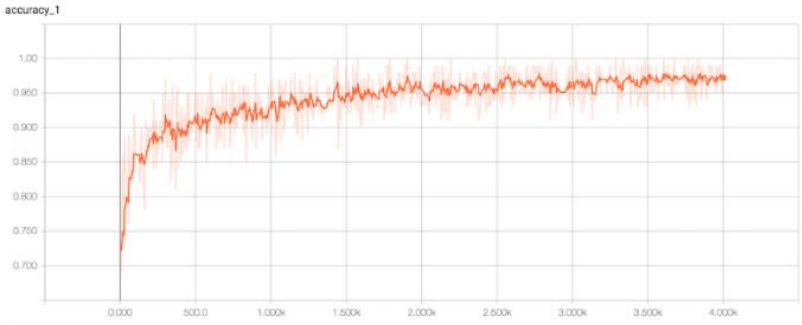

Рисунок 4.1 Диаграмма обучения

На рисунке 4.1 по оси x – количество тренировочных шагов, по оси y – точность.

Чем полученное значение ближе к единице, тем более достоверными и точными считается результат. Как видно из графика, нейронная сеть быстро адаптируется к новым входным данным и продолжает обучение. Средняя абсолютная процентная погрешность прогноза на обучаемом наборе равняется 70% , что считается хорошим результатом.

Обе модели проверены на новых данных, которые были собраны на различных интернет ресурсах.

Тестирование включает в себя использование двух наборов данных: цветов и грибов. В ходе исследования получили два соответствующих

набора результатов: распознавание вида цветов и грибов, и показатель точности для каждого вида распознавания.

При предсказании распознавания видов показатель точности, полученный с помощью метода score варьируется от 0.8 до 0.99, в зависимости от выбранной модели. В таблице 4.1 приведены значения распознавания для каждого видов цветов. В таблице 4.2 приведены значения распознавания для каждого видов грибов.

Таблица 4.1

Изображение на входе	Вид цветка	Ожидаемый результат	Полученный результат
	ромашка	0.99999	0.99072
	одуван-чик	0.99999	0.99102
	роза	0.99999	0.97023

	подсол-нух	0.99999	0.99080
	тюльпан	0.99999	0.97145

Результаты обучения цветов

Таблица 4.2

Изображение на входе	Вид гриба	Ожидаемы результат	Получен-ный резуль-тат
	белый гриб	0.99999	0.99012
	подберезовик	0.99999	0.97171

	лисичка	0.99999	0.98010
	подосиновик	0.99999	0.97001
	бледная поганка	0.99999	0.98176
	мухомор белый	0.99999	0.96781
	мухомор красный	0.99999	0.99999

	рядовка ядовитая	0.99999	0.99019

Результаты обучения грибов

Таблица 4.3

Вид гриба	Ожидаемы результат определения ядовитости	Определение ядовитости	Процент съедобности
белый гриб	съедобен	съедобен	0.97121
подберезовик	съедобен	съедобен	0.97145
лисичка	съедобен	съедобен	0.98899
подосиновик	съедобен	съедобен	0.97214
бледная поганка	ядовит	ядовит	0.98019
мухомор белый	ядовит	ядовит	0.98178
мухомор красный	ядовит	ядовит	0.99789
рядовка ядовитая	ядовит	ядовит	0.98907

Результаты съедобности и ядовитости грибов

Распознавание имеет высокий процент точности, что говорит о качестве выбранной библиотеки и её методах.

4.2 Оценка точности результатов

В качестве обучающей выборки для каждого вида было выбрано 1000 фотографий. В качестве тестовой выборки было выбрано 500 фотографий. Распознавание видов цветов и грибов при использовании библиотеки TensorFlow даёт высокий показатель точности. В таблице 4.4 приведён анализ распознавания других решений, конкурирующих с TensorFlow.

Таблица 4.4

	Caffe	Theano	Torch	TensorFlow
точность	99.1%	99.16%	99.4%	99.70%

Сравнение с другими библиотеками

Несмотря на небольшие отличия в значениях точности, которые обусловлены различиями в настройках начальных весов сетей и параметрах методов оптимизации, применяемых в процессе обучения [15], точность выбранной библиотеки выше, что говорит о том, что библиотека TensorFlow является наилучшим решением поставленной задачи.

4.3 Что было реализовано в приложении

Полученные обученные графы цветов и грибов были реализованы для приложения Android. Библиотека TensorFlow позволяет работать с методами и классами для реализации классификатора изображений в приложении Java.

4.3.1 Добавление графов и меток в проект

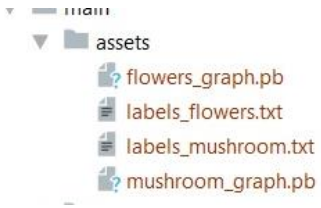

Рисунок 4.2 Добавление графов в проект Android Studio

Чтобы использовать полученные модели необходимо добавить графы и метки в корень проекта, показано на рисунке 4.2

Также необходимо добавить атрибуты для графов, показано на рисунке 4.3:

```
package com.lely.florofil.classifier

const val GRAPH_FLOWERS_PATH = "file:///android_asset/flower_graph.pb"
const val LABELS_FLOWERS_PATH = "file:///android_asset/labels_flowers.txt"
const val GRAPH_MUSHROOMS_PATH = "file:///android_asset/mushroom_graph.pb"
const val LABELS_MUSHROOMS_PATH = "file:///android_asset/labels_mushroom.txt"

const val GRAPH_INPUT_NAME = "Mul"
const val GRAPH_OUTPUT_NAME = "final_result"

const val IMAGE_SIZE = 300
const val IMAGE_MEAN = 128
const val IMAGE_STD = 128

const val COLOR_CHANNELS = 10

const val ASSETS_PATH = "file:///android_asset/"
```

Рисунок 4.3

4.3.2 Добавление классификатора для распознавания видов цветов и грибов

Метод recognizeImage копирует данные в TensorFlow и находит лучшую классификацию. (Рисунок 4.4)

```
class ImageClassifier (
    private val inputName: String,
    private val outputName: String,
    private val imageSize: Long,
    private val labels: List<String>,
    private val imageBitmapPixels: IntArray,
    private val imageNormalizedPixels: FloatArray,
    private val results: FloatArray,
    private val tensorFlowInference: TensorFlowInferenceInterface
) : Classifier {

    override fun recognizeImage(bitmap: Bitmap): Result {
        preprocessImageToNormalizedFloats(bitmap)
        classifyImageToOutputs()
        val outputQueue = getResults()
        return outputQueue.poll()
    }
```

Рисунок 4.4 Классификатор для распознавания

```
private fun classifyImageToOutputs() {
    tensorFlowInference.feed(inputName, imageNormalizedPixels,
        1L, imageSize, imageSize, COLOR_CHANNELS.toLong())
    tensorFlowInference.run(arrayOf(outputName), ENABLE_LOG_STATS)
    tensorFlowInference.fetch(outputName, results)
}

private fun getResults(): PriorityQueue<Result> {
    val outputQueue = createOutputQueue()
    results.indices.mapTo(outputQueue) { Result(labels[it], results[it]) }
    return outputQueue
}

private fun createOutputQueue(): PriorityQueue<Result> {
    return PriorityQueue(
        labels.size,
        Comparator { (_, rConfidence), (_, lConfidence) ->
            Float.compare(lConfidence, rConfidence) })
}
```

Рисунок 4.5 Методы результата распознавания

4.5 Возможные пути развития

Так как выбранный алгоритм показывает высокую точность результатов, не только распознавания видов цветов и грибов, но и определяет съедобность или ядовитость грибов, необходимо добавить новые виды цветов и грибов для обучения.

Заключение

В ходе данной работы были поставлены задачи распознавания видов цветов и грибов, распознавание съедобности и ядовитости грибов с использованием методов глубинного обучения сверточных нейронных сетей. Проанализировали несколько методов, которые уже применяются в исследуемой области. Перед постановкой задачи и выбора средств для её реализации были описаны сильные и слабые стороны конкурентов.

Основной идеей реализуемого подхода является использование изображений, которые поступают на вход нейронной сети, и использование явных признаков грибов для определения съедобности и ядовитости.

Проект реализован с помощью языка программирования Python и библиотекой TensorFlow. Выбранная библиотека отлично работает с Python и имеет много методов и классов для реализации классификатора изображений в приложении Java. Выбранная библиотека помогает получить знания и опыт в работе с данными и алгоритмами машинного обучения, она также используется в профессиональных рабочих проектах.

После создания и обучения модели получили процент предсказания распознаваемого изображения. Получив данные с явными признаками грибов, также обучив модель, получили определение съедобности гриба.

Список использованных источников

[1] Л. Гарибова и С. Лекомцева, Основы микологии: Морфология и систематика грибов и грибоподобных организмов., 2005..

[2] A. S. I. &. H. G. E. Krizhevsky, Imagenet classification with deep convolu-tional neural networks., (2012)..

[3] «Искусственный нейрон,» [В Интернете]. Available: https://ru.wikipedia.org/wiki/Искусственный_нейрон .

[4] M. T.M., Machine Learning. McGraw-Hill, 1997.

[5] Y. D. Deng L., Deep Learning: Methods and Applications, 2014. Vol. 7, No. 3–4, P. 197-387..

[6] «Информация по нейронным сетям,» [В Интернете]. Available: http://mechanoid.kiev.ua/neural-net-backprop.html.

[7] К. А. А. Е. Николенко С., Глубокое обучение. Погружение в мир нейрон-ных сетей., СПб.: Питер, 2018.

[8] «Информация по использованию библиотек машинного обучения,» [В Интернете]. Available: http://habrahabr.net/thread/12437.

[9] «tfbenchmarks,» [В Интернете]. Available: https://github.com/soumith/convnet-benchmarks/blob/master/README.md.

[10] «Информация по сравнению программ глубокого обучения,» [В Интернете]. Available: http://ai-news.ru/2018/05/ai_prakticheskij_kurs_sravnenie_po_glubokogo_obucheniya.html.

[11] «Обучающий материал TensorFlow,» [В Интернете]. Available: https://neurohive.io/ru/tutorial/tensorflow-tutorial-tenzory-i-vektory/.

[12] «Mushroom Data Set,» [В Интернете]. Available: http://archive.ics.uci.edu/ml/datasets/Mushroom.

[13] «Нейросети: реализация задачи про грибы,» [В Интернете]. Available: https://habr.com/ru/post/419917/.

[14] «Using a Pre-Trained TensorFlow Model on Android,» [В Интернете]. Available: https://medium.com/capital-one-tech/using-a-pre-trained-tensorflow-model-on-android-e747831a3d6.

[15] «Сравнение библиотек глубокого обучения,» [В Интернете]. Available: https://habr.com/ru/company/intel/blog/254747/.