Санкт-Петербургский государственный университет
Факультет прикладной математики – процессов управления
Кафедра математической теории микропроцессорных систем управления

Бондарев Петр Сергеевич

Выпускная квалификационная работа бакалавра

Бюджетное мобильное приложение для слабовидящих пользователей

Направление 02.03.02
Фундаментальная информатика и информационные технологии

Заведующий кафедрой,
доктор физ.-мат. наук,
профессор Зубов А.В.

Научный руководитель,
ассистент,
Ужегов Н.С.

Рецензент,
кандидат физ.-мат. наук,
доцент Ковшов А.М.

Санкт-Петербург
2018

Содержание

Введение ... 3

Постановка задачи ... 5

Обзор литературы ... 6

Глава 1. Мониторинг современных технологий ... 8

1.1. Современные мобильные операционные системы ... 8

1.2. Роль технологий для людей с нарушениями зрения ... 10

Глава 2. Требования к продукту ... 13

2.1. Обзор имеющихся решений ... 13

2.2. Уточнение требований ... 17

Глава 3. Машинное обучение с использованием TensorFlow ... 18

3.1. Машинное обучение ... 18

3.2. Нейронные сети ... 19

3.3. Глубокое обучение ... 22

3.4. TensorFlow. Обзор и сравнение с другими библиотеками ... 24

Глава 4. Программная реализация ... 29

4.1. Средства разработки платформы Android ... 29

4.2. Архитектура приложения ... 30

4.3. Реализация классификатора ... 33

4.4. Описание функциональности ... 37

Выводы ... 39

Заключение ... 40

Список литературы ... 41

Приложение ... 43

Введение

Во всем мире насчитывается 285 миллионов человек, которые слепы или имеют нарушения зрения [1]. И каждый год это число увеличивается. Решение проблем, с которыми сталкиваются люди с ограниченными возможностями, в том числе и люди с нарушением зрения, является одной из наиболее важных и актуальных задач на сегодняшний день. Доступность компьютеров и мобильных устройств, без которых большинство людей в современном мире уже не могут обойтись, тем не менее порождает и новые трудности перед людьми с ограничениями здоровья. Согласно опросу «Яндекса», 66,5 процентов людей с нарушением зрения активно пользуются смартфонами [2]. Поэтому непосредственной задачей программистов является создание и усовершенствование такого программного обеспечения, которое могло бы упростить использование девайса или помочь в повседневной жизни.

Ранее единственным доступным для слепых средством ориентирования в пространстве было использование белых тростей или собак-поводырей. Однако с развитием инновационных технологий данный список может быть значительно расширен. Это, например, очки Assisted Vision Smart Glasses, с помощью которых пользователь может визуально определять информацию, устройство AI Glasses, которое также выглядит как обычные очки, но содержит в себе стереозвуковые и GPS датчики. Оба устройства в буквальном смысле снабжают слепого всеми необходимыми знаниями, вслух описывая окружающее пространство. Однако подобные гаджеты стоят немалых денег и доступны далеко не всем слабовидящим.

В рамках данной работы будет описано создание бюджетного мобильного приложения, которое сможет определять с помощью тыловой камеры, что находится перед ней, и голосовым сигналом сообщать об обнаруженном объекте, тем самым обеспечивая безопасное передвижение пользователя приложения. В базу объектов будут внесены слова определенной

тематической группы, характеризующие городское пространство, такие как скамейка, урна, столб, дерево, также приложение сможет идентифицировать людей. В первую очередь разрабатываемое приложение должно быть удобно и доступно всем слабовидящим, независимо от материального положения.

Постановка задачи

Целью данной выпускной квалификационной работы бакалавра является разработка программного продукта, основной задачей которого являлось бы помочь слабовидящим и незрячим людям обнаружить препятствие перед собой. Оповещение об обнаруженных препятствиях будет происходить при помощи голосового сигнала, который поможет идентифицировать каждый конкретный объект.

Для достижения нашей цели были сформулированы следующие задачи:

- Выбор мобильной операционной системы.
- Анализ существующих решений, сравнение их достоинств и недостатков.
- Выбор технологий создания приложения, инструментов и средств.
- Разработка доступного интерфейса для нашей целевой аудитории.

Обзор литературы

В ходе написания выпускной квалификационной работы были использованы различные источники. Это научная и учебная литература, научно-исследовательские статьи и интернет-публикации, касающиеся как теоретической составляющей данной работы, так и помогающие разобраться в практической.

Книга П. Дейтела, Х. Дейтела, и А. Уолда «Android для разработчиков. 3-е издание» послужила отличным практическим помощникам при разработке программного продукта под операционную систему Android. В источнике описываются основные возможности платформы, основные технологии и концепции разработки.

Основной литературой, раскрывающей теоретические основы машинного обучения и нейронных сетей, стали лекции К. Воронцова и книга И. Бенджио, Я. Гудфеллоу и А. Курвилля под названием «Глубокое обучение». При изучении глубокого обучения и глубоких сетей использовались учебники Ли Денга и Донга Ю «Глубокое обучение: методы и применение» и С. Николенко, А. Кадурина, и Е. Архангельской «Глубокое обучение. Погружение в мир нейронных сетей». В книге Денга и Ю предоставляется обзор общей методологии глубокого обучения и ее применение для различных задач. Во второй – показываются архитектуры глубоких нейронных сетей, алгоритмы их обучения, и все это подкреплено практическими советами и примерами реализации с использованием современной библиотеки машинного обучения TensorFlow.

Помимо этого, руководство, касающееся реализации проекта с помощью TensorFlow, представлено на официальном сайте tensorflow.org. На сайте содержится документация, демонстрирующая ключевые функции API, все необходимые компоненты, детали и принципы работы.

Данные, относящиеся к выявлению предпочтений людей с нарушениями зрения, основаны на информации, полученной путем опросов и исследований компаний Яндекс и WebAIM.

Глава 1. Мониторинг современных технологий

1.1. Современные мобильные операционные системы

Ушли дни, когда мы использовали мобильные телефоны только для совершения звонков и отправки текстовых сообщений. Теперь функционал современного смартфона значительно расширился, что позволяет его сравнивать даже с компьютером. Лишь размер аппаратов является наиболее значимым отличием, сам процессор, операционная система схожи. И хотя на мировом рынке пользователю предлагается множество операционных систем, нельзя сказать, что каждая из них обладает рядом исключительных особенностей, в целом, все они выполняют одни и те же действия.

На простом языке операционная система (ОС) – это не что иное, как инфраструктура программного обеспечения, созданная программными компонентами, с помощью которых наше оборудование может работать более эффективно. Мобильная операционная система (мобильная ОС) – это специально разработанная ОС для смартфонов, планшетов и других мобильных устройств. Она контролирует все основные операции, которые не только позволяют пользователю устанавливать и запускать новые приложения, но и добавлять дополнительные функции на мобильное устройство.

Ни для кого не секрет, что в настоящее время лидерами на рынке мобильных ОС являются Android, iOS и Windows Phone. По данным компании IDC, на долю системы Android приходится 85 процентов, что составляет 2 млрд. активных устройств. Ниже приведена таблица пользователей различных платформ за последние периоды (см. таблицу 1) [3].

период	Android	IOS	Windows Phone	другие
2016Q1	83,4%	15,4%	0,8%	0,4%
2016Q2	87,6%	11,7%	0,4%	0,3%
2016Q3	86,8%	12,5%	0,3%	0,4%
2016Q4	81,4%	18,2%	0,2%	0,2%
2017Q1	85,0%	14,7%	0,1%	0,1%

Источник: IDC, май 2017 г.

Таблица 1. Рынок мобильных операционных систем

Android – самая широко используемая мобильная операционная система на планете. Главное и значительное преимущество перед другими системами состоит в том, что компания Google предлагает открытый исходный код своей системы, тем самым давая разработчикам неограниченные возможности для написания приложений под эту платформу. Поддерживается на большинстве современных смартфонов, включая Samsung, Sony, LG, HTC, Motorola и, конечно же, Google.

Вторая по популярности – iOS от компании Apple, используемая на всех iPhone, iPad и iPod Touch. iOS – это собственная мобильная операционная система, имеющая закрытый исходный код. Это означает, что ее можно найти только на продуктах Apple, цены которых находятся на высоком уровне.

Windows Phone (WP) – это сравнительно молодая мобильная ОС от Microsoft. Хотя она и не так популярна, как заявленные ранее, WP предлагает надежный и персонализированный пользовательский интерфейс, полностью совместимый с другими продуктами Microsoft. Недостатком является то, что доступно ограниченное количество приложений по сравнению с устройствами на Android и iOS.

Проанализируем данные по распределению различных версий ОС Android (см. рисунок 1), опубликованные компанией Google [4].

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.5%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.5%
4.1.x	Jelly Bean	16	2.2%
4.2.x		17	3.1%
4.3		18	0.9%
4.4	KitKat	19	13.8%
5.0	Lollipop	21	6.4%
5.1		22	20.8%
6.0	Marshmallow	23	30.9%
7.0	Nougat	24	17.6%
7.1		25	3.0%
8.0	Oreo	26	0.3%

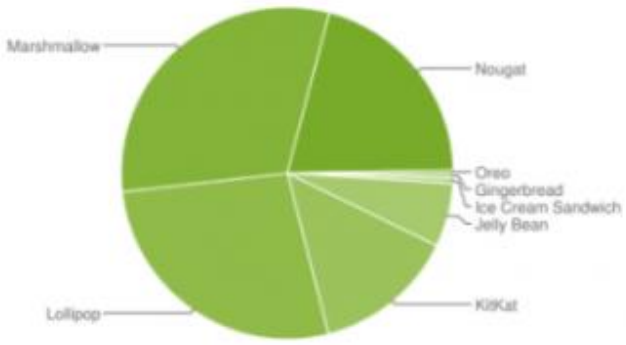


Рисунок 1. Статистика распределения версий ОС Android

Наиболее распространенной из ОС Android является версия 6.0 Marshmallow, установленная на 30,9 процентов всех Android-устройств. И несмотря на появление более новых версий Android, версия 6.0 все еще остается на лидирующих позициях.

1.2. Роль технологий для людей с нарушениями зрения

В 2015 году компания Яндекс провела опрос среди людей с нарушениями зрения, позволяющий определить интересы и предпочтения данной категории пользователей [2]. Обратимся к таблице 2, демонстрирующей распределение респондентов по мобильным ОС.

Ответ	Доля
Android	77,9%
iOS	32,9%
Symbian	20,8%
Другая	0,8%

Таблица 2. Опрос Яндекса

Необходимо отметить, что большинство людей с нарушениями зрения пользуются Android, при этом также заметна тенденция к использованию одновременно нескольких мобильных ОС (сумма долей превышает 100%). Можно предположить, что часть пользователей таким образом компенсирует слабые стороны устаревшего аппарата на Symbian более новым, модернизированным аппаратом на Android или iOS. При таком варианте эксплуатации устройств Symbian используется для совершения элементарных задач, таких как вызов абонента, а устройство на более современной ОС – прочих мультимедийных. Распространенность такого подхода среди незрячих может определяться наличием устойчивой привычки к использованию кнопочных аппаратов как наиболее знакомых и понятных.

Однако подобное исследование, проведенное среди пользователей англоязычного рынка [5], указывает на противоположные предпочтения респондентов: на долю iOS приходится 75 процентов, в то время как на Android – 22. Возможно, такие расхождения в результатах исследований обусловлены социально-экономическими факторами.

Далее рассмотрим, какими вспомогательными технологиями пользуются слабовидящие люди (см. таблицу 3):

Ответ	Доля
Программу экранного доступа с синтезатором речи	94,1%
Программу увеличения экрана и коррекции изображения	10,9%
Программу экранного доступа с брайлевским дисплеем	9,8%
Другую	2%

Таблица 3. Опрос Яндекса

Программные синтезаторы речи являются ключевым инструментом для пользователей, имеющих нарушения зрения. Поэтому опрошенных, работающих с данными технологиями экранного доступа, существенно больше чем других. Однако следует также заметить, что доля людей, которые пользуются средствами увеличения экрана, все же больше чем представлена в таблице, так как они зачастую просто не относят себя к группе лиц с ограниченными возможностями. Поэтому в исследовании участвовал лишь небольшой их процент.

Часть слабовидящих пользователей совмещают различные вспомогательные технологии вместе (сумма долей более 100%). Процент пользователей, прибегающих к использованию брайлевских дисплеев является одним из наименьших. Это объясняется первую очередь тем, что данный вид дисплеев считается одним из наиболее дорогостоящих. К тому же, по данным некоторых исследований, лишь около 15 процентов незрячих умеют пользоваться азбукой Брайля.

Глава 2. Требования к продукту

2.1. Обзор имеющихся решений

Google Play и его востребованность среди пользователей платформы Android

Google Play (в прошлом – Android Market) – это сервис цифровой дистрибуции, управляемый и разработанный компанией Google. Он является официальным магазином приложений для операционной системы Android, позволяя пользователям просматривать и загружать приложения, разработанные с помощью Android SDK и опубликованные через Google. Google Play также предлагает музыку, журналы, книги, фильмы и телевизионные программы.

На данный момент магазин включает 26 различных категорий приложений:

- Бизнес
- Виджеты
- Живые обои
- Здоровье и спорт
- Инструменты
- Книги и справочники
- Комиксы
- Медицина
- Музыка и видео
- Мультимедиа и видео
- Новости и журналы
- Образование
- Персонализация
- Погода
- Покупки
- Путешествия
- Работа
- Развлечения
- Разное
- Связь
- Социальные
- Спорт
- Стиль жизни
- Транспорт
- Финансы
- Фотография

На март 2018 в Google Play содержится более чем 3,7 млрд. активных приложений [6]. Вместе с тем пользователи часто жалуются на приложения низкого качества, находящиеся в магазине. Поэтому прежде всего важно обеспечить разработку качественного продукта. Основным моментом начала

разработки является поиск и анализ уже существующих аналогичных или схожих приложений.

Существующие программные продукты. Преимущества и недостатки.

Приложений на платформе Android, способных помогать слабовидящим и незрячим людям ориентироваться в пространстве, крайне мало. Рассмотрим и проанализируем самые популярные из них, давно зарекомендовавшие себя на рынке. Разберем принципы их работы и сравним достоинства и недостатки.

TapTapSee

Описание: TapTapSee разработан CamFind Inc., чтобы помочь людям с нарушениями зрения идентифицировать объекты, с которыми они сталкиваются в своей повседневной жизни. Приложение работает на основе API распознавания изображений CloudSight. Использует функции камеры для фотографирования объектов и Talkback[1] для озвучивания их пользователю.

Работа с программой (см. рисунок 2): Прежде чем приступить к использованию, нужно на своем смартфоне включить сервис Talkback. Пользователь дважды касается экрана устройства, чтобы сделать снимок. Приложение тщательно анализирует и определяет его в течение нескольких секунд, путем отправки изображения на сервер. После получения ответа, Talkback устройства сообщает о результатах пользователю. Помимо общего определения объектов, пользователь также может узнать информацию с этикетки какого-либо продукта, например, бренд, название и информацию о продукте.

[1] TalkBack – это сервис от Google, который озвучивает текст, элементы интерфейса, а также обеспечивает звуковой и виброотклик.

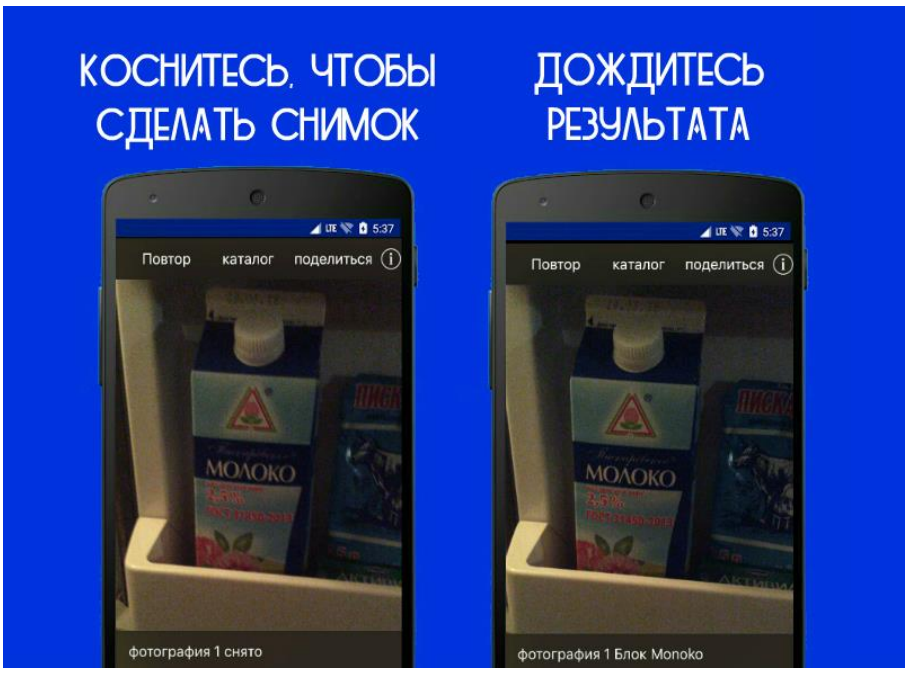


Рисунок 2. Работа с TapTapSee

Be My Eyes

Описание: Приложение с говорящим названием «Будь моими глазами» помогает слепым или слабовидящим людям «видеть» вещи с помощью зрячих добровольцев и видеокамер на своих смартфонах. Посредством онлайн видеозвонка оно дает незрячим возможность попросить помощи волонтеров, готовых помочь им в задачах, требующих нормального зрения, таких как переход дороги или проверка срока годности продуктов.

Работа с программой (см. рисунок 3): После открытия приложения пользователь должен коснуться середины экрана, чтобы подключиться к первому доступному помощнику. Затем приложение скажет, что создало запрос на подключение к серверам. Далее пользователь получает голосовое сообщение, в котором говорится: «Ожидание другой стороны». Это продолжается, пока кто-то не отвечает на звонок и не приветствует вас. Это может занять минуту или две. После подключения незрячий «заимствует» глаза помощника через свой смартфон. Зрячий помощник может видеть и описывать, что ему показывает слепой человек, который снимает все происходящее с помощью своей видеокамеры в смартфоне. Таким образом,

работая вместе, они могут решить проблемы, с которыми сталкиваются незрячие люди.

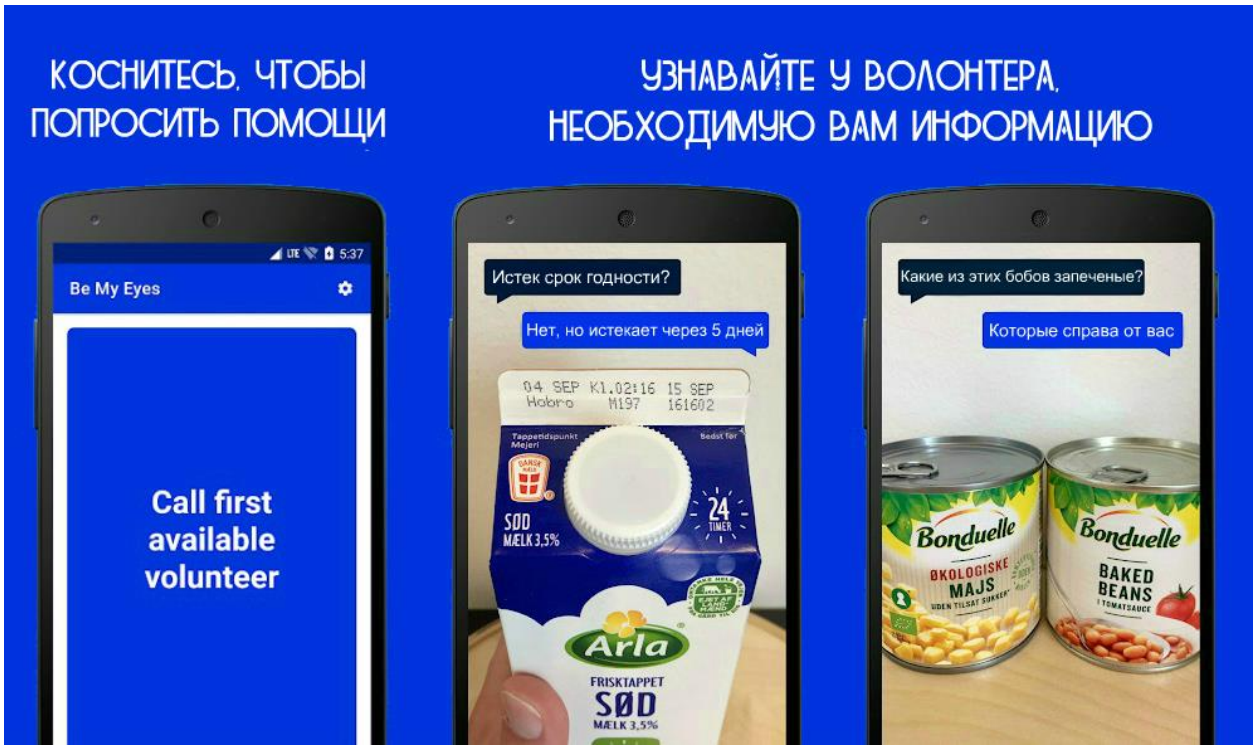


Рисунок 3. Работа в «Be My Eyes»

И TapTapSee, и Be My Eyes крайне полезны для нашей целевой аудитории, однако можно выделить и недостатки, при работе с которыми могут столкнуться пользователи. Основными минусами обоих приложений является невозможность работы в офлайн, при выключенном или недоступном интернет-соединении. TapTapSee получает информацию со своего сервера, а в Be My Eyes видеосвязь определенно требует наличие сети. TapTapSee также не может похвастаться возможностью работы в режиме «реального времени». Как было сказано выше, в этом приложении определение информации происходит через фотосъемку. Допустим, вам нужно перейти улицу. Пока пользователь сделает фотографию и дождется полученного ответа с сервера о идентифицированном объекте, время на ответ, к слову, составляет 5-7 секунд, пройдет довольно большой промежуток времени. Поэтому это приложение будет неактуально использовать при перемещении по пространству и определению препятствий. Также, как замечают в отзывах на Google Play про

TapTapSee, продукт не совсем бесплатен, и после определенного лимита фотографий, начинает требовать с вас плату за загрузку. Помимо этого, можно сказать и о доступном для незрячих интерфейсе, который не требовал бы от них никаких касаний или было бы достаточно минимального взаимодействия с дисплеем смартфона.

2.2. Уточнение требований

Мощные вычислительные возможности современных мобильных устройств и использование математических моделей позволяют создавать приложения, способные помогать незрячим людям справляться с повседневными трудностями. Однако до сих пор не существует продукта на базе ОС Android, позволяющего идентифицировать информацию, работая при этом не только в режиме реального времени, но и без подключения к сети Интернет.

Основываясь на проведенном анализе, были сформулированы следующие требования:

1. Определение образов;
2. Озвучивание результата;
3. Работа в режиме «реального времени»;
4. Возможность работы в офлайн;
5. Доступный интерфейс.

Глава 3. Машинное обучение с использованием TensorFlow

3.1. Машинное обучение

В 1959 году Артур Сэмюэль ввел термин «машинное обучение», который определил, как процесс, в результате которого машина (компьютер) способна показывать поведение, которое в нее не было явно заложено (запрограммировано) [7]. Сэмюэль разработал программу, которая научилась играть в шашки лучше, чем он. Его определение – отличное определение, но, возможно, слишком расплывчатое. Том Митчелл, еще один хорошо известный исследователь по компьютерному обучению, предложил более точное определение в 1998 году: «Говорят, что компьютерная программа обучается на основе опыта E по отношению к некоторому классу задач T и меры качества P, если качество решения задач из T, измеренное на основе P, улучшается с приобретением опыта E» [8].

Вышеприведенные определения устанавливают четкую цель для машинного обучения, однако они не говорят нам, как достичь этой цели. Мы должны уточнить наше определение. Машинное обучение – это практика использования алгоритмов для анализа данных, их изучения, а затем определения или прогнозирования чего-либо в мире.

Рассмотрим основные алгоритмы машинного обучения:

- Обучение с учителем: Алгоритм получает обучающиеся данные, которые содержат «правильный ответ» для каждого примера. Например, алгоритм «обучения с учителем» для обнаружения мошенничества с кредитными картами будет принимать в качестве входных данных набор зарегистрированных операций. Для каждой транзакции данные обучения будут содержать флажок, указывающий, является ли она мошеннической или нет.

Таким образом, мы заранее обучаем нашу систему с помощью учителя.

- Обучение без учителя: Алгоритм ищет не пары объект-ответ, а структуру (связи) в обучающихся данных, например, какие примеры похожи друг на друга, и группирует их в кластеры.

  Система в данном случае обучается спонтанно, без вмешательства учителя.

3.2. Нейронные сети

Нейронная сеть представляет собой мощный механизм машинного обучения, который в основном имитирует 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.

Впервые, понятие искусственной нейронной сети было рассмотрено в 1943 году биофизиками У. Питтсом и У. Маккалаком в статье [9]. В частности, ими была предложена модель искусственного нейрона. В 1960-х годах Ф. Розенблатт используя полученную модель Маккалака-Питтса, продолжил данную разработку и впоследствии дал миру первый персептрон – метод обучения для этой модели.

Основной персептрон представлен следующим образом (см. рисунок 4):

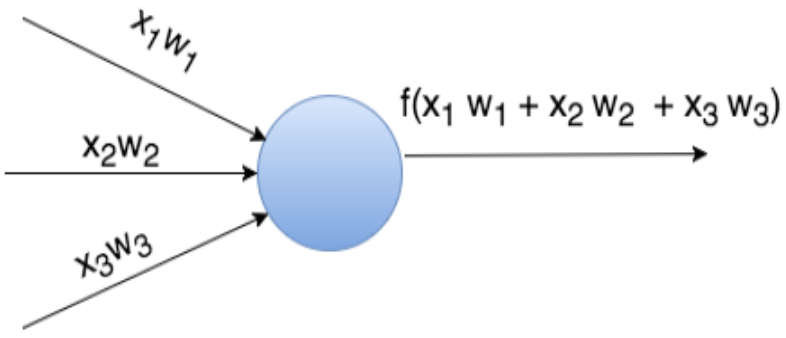


Рисунок 4. Модель искусственного нейрона

- x_i – входной сигнал;
- w_i – вес входного сигнала;
- $f(s)$ – функция активации.

  s – значение, полученное на сумматоре входных сигналов.

Нейрон через входные каналы получает сигналы (в нашем случае от x1, x2 и x3). Каждый из сигналов имеет определенный вес (w1, w2, w3 соответственно). Каждый из входов взвешивается и суммируется на узле:

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

Затем эта сумма передается функции активации, после вычисления которой получается выходной сигнал нейрона.

Функции активации

Активационные функции являются чрезвычайно важной особенностью искусственных нейронных сетей. Они решают, должен ли нейрон быть активирован или нет, является ли информация, которую получает нейрон, актуальной или ее следует игнорировать.

Функция активации – это преобразование, которое мы вычисляем над входным сигналом.

Опишем наиболее используемые функции активации:

- Сигмоидная (логистическая): $f(s) = \frac{1}{1+e^{-s}}$;
- Гиперболистическая: $f(s) = \frac{\sinh(s)}{\cosh(s)} = \frac{e^{s}-e^{-s}}{e^{s}+e^{-s}}$;
- Положительная линейная (ReLU): $f(s) = \max(0, s)$;

На рисунке 5 проиллюстрированы графики этих функций.

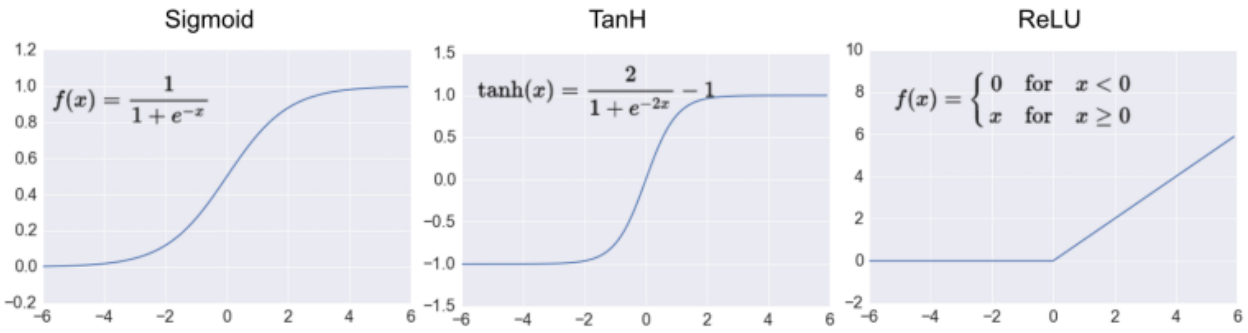


Рисунок 5. Основные функции активации

Функция активации выбирается в зависимости от поставленной задачи. Для классификации будет удобно использовать функцию мягкого максимума (softmax). Представим, что у нас имеется n разных значений. В этом случае на выходе нейронной сети мы получим распределение вероятностей принадлежности входного образа одному из непересекающихся n классов.

Softmax представляет собой обобщение сигмоидной функции для многомерного случая [10] и определяется следующим образом:

$$f(s)_j = \frac{e^{s_j}}{\sum_{k=1}^{K} e^{s_k}} \text{ для } j = 1, \ldots, K.$$

Общая сумма выходных значений будет равна единице.

Функция потерь

На выходе нейронной сети каждой обучающей выборки мы будем иметь нужное нам распределение вероятностей. Для сравнения двух распределений требуется установить корректную меру.

Пусть X – множество меток объектов, Y – множество допустимых ответов. Существует зависимость $y^*: X \to Y$, значения которой известны только на объектах обучающей выборки $X^n = \{(x_1, y_1), \ldots, (x_n, y_n)\}$.

Введем функцию потерь $\mathcal{L}(y, y')$, которая характеризует величину отклонения предсказанных ответов y от истинных y', где $y' = y^*(x)$ на всяком $x \in X$.

Функция потерь отображает наше представление о том, насколько неточен нейрон в принятии решений при текущем значении параметров. Если мы сталкиваемся с задачей классификации и распределением вероятностей, то в качестве функции потерь предпочтительнее использовать перекрестную энтропию (cross entropy). Формула кросс-энтропии выглядит следующим образом:

$\mathcal{L}(y_i, y^*(x_i)) = -\sum_{j=1}^{K} y_{ij}^* \log y_{ij}$, где K – количество меток классов.

Задача классического метода обучения сводится к тому, чтобы минимизировать эмпирический риск путем настраивания весов W. Эмпирический риск (или функционал качества) – это мера, которая характеризует среднюю ошибку на обучающей выборке или доля неправильных ответов [11]. Функционал качества вычисляется формулой:

$$Q(y, X) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(y_i, y^*(x_i)).$$

3.3. Глубокое обучение

Традиционные методы машинного обучения были ограничены в своей способности обрабатывать естественные данные в их исходной форме. На протяжении десятилетий построение системы распознавания образов или машинного обучения требовало тщательной разработки и значительного опыта в области проектирования механизма извлечения признаков, который преобразовывал необработанные данные (например, значения пикселей изображения) в подходящее внутреннее представление, из которого подсистема обучения могла обнаруживать или классифицировать шаблоны на входе.

Глубокое обучение (или глубинное обучение) является частью более широкого семейства методов машинного обучения, основанных на обучении

представлениям, в отличие от алгоритмов, ориентированных на конкретные задачи [12].

Обучение представлению – это набор методов, который позволяет машине подавать необработанные данные и автоматически обнаруживать представления, необходимые для идентификации или классификации. Методы глубокого обучения – это методы обучения представлению с несколькими уровнями представления, полученные путем составления простых, но нелинейных модулей, каждый из которых преобразует представление на одном уровне (начиная с исходного ввода) в представление на более высоком, немного более абстрактном уровне. Это и делает глубокое обучение глубоким. Каждый уровень описывает какой-то вид информации, упорядочивает ее и передает на следующий уровень. Например, изображение имеет форму массива значений пикселей, изученные объекты в первом слое представления обычно представляют наличие или отсутствие ребер в определенных ориентациях и местоположениях изображения. Второй слой обычно определяет модели путем выделения определенных расположений краев, независимо от небольших изменений в положениях краев. Третий слой может собирать модели в более крупные комбинации, которые соответствуют частям знакомых объектов, а последующие слои будут обнаруживать объекты как комбинации этих частей.

Модель глубокой сети можем рассмотреть на примере рисунка 6. Между входным и выходным слоями находится множество скрытых слоев. Чем их больше, тем глубже наша сеть.

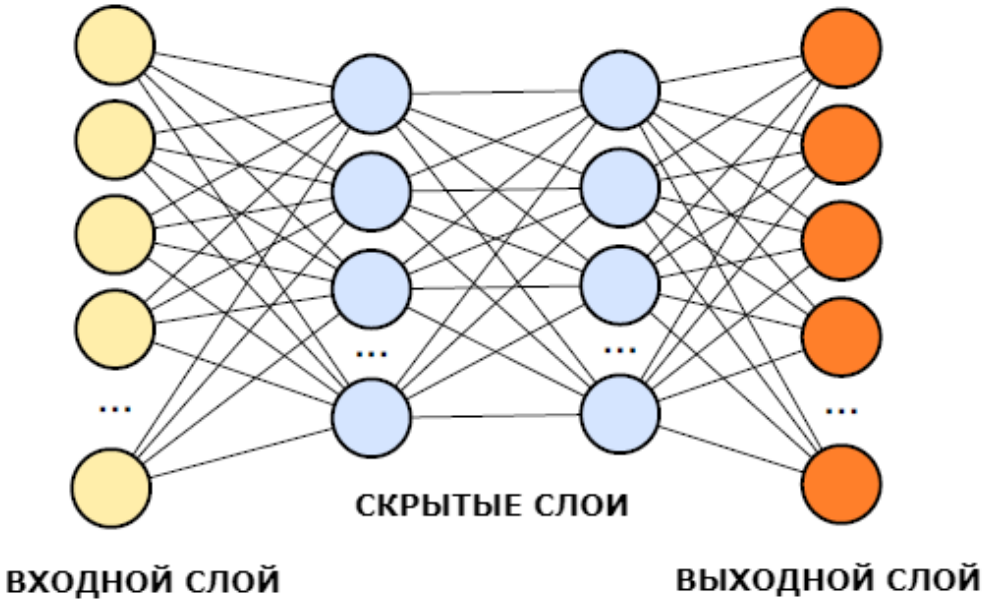


Рисунок 6. Модель глубокой сети

Глубокое обучение делает большие успехи в решении проблем, которые на протяжении многих лет противостояли лучшим попыткам искусственного интеллекта. Оно оказалось очень хорошо в обнаружении сложных структур в многомерных данных и поэтому применимо ко многим областям науки. Методы глубокого обучения очень сильно повлияли на положение дел в распознавании речи и зрительных объектов.

3.4. TensorFlow. Обзор и сравнение с другими библиотеками

TensorFlow – это мощная библиотека 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 [13]. Хотя TensorFlow доступен чуть больше двух лет, он быстро стал одним из самых популярных проектов машинного обучения с открытым исходным кодом на GitHub.

В TensorFlow вычисления не выполняются сразу, вместо этого создаются графы, описывающие вычисления. Узлы графа представляют собой

математические операции, 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 [14].

Данная библиотека содержит следующие преимущества:

1. TensorFlow полностью бесплатен и имеет открытый исходный код. Поддерживается компанией Google.
2. Фреймворк достаточно прост в использовании и поддерживает большинство практических задач.
3. 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.
4. 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.
5. TensorFlow можно использовать для обучения и обслуживания моделей в режиме реального времени. Очевидно, что коды не нуждаются в переписывании, и разработчики могут применять свои идеи к продуктам намного быстрее.
6. TensorFlow позволяет максимально использовать доступное оборудование, поскольку имеет расширенную поддержку потоков, асинхронных вычислений и очередей.

Другие доступные библиотеки, конкурирующие и сравнимые с TensorFlow:

- Theano
- Torch

- Caffe
- MXNet
- Neon
- CNTK

Рассмотрим основные особенности и характеристики этих инструментов. Сравнительный анализ представлен в таблице 4.

	Языки	Руководство и учебные материалы	Возможность моделирования CNN	Архитектура: простота в использовании и модульный фронтенд	Скорость	Совместимость с Keras
Theano	Python, C++	++	++	+	++	+
Tensor-Flow	Python	+++	+++	+++	++	+
Torch	Lua, Python (new)	+	+++	++	+++	
Caffe	C++	+	++	+	+	
MXNet	R, Python, Julia, Scala	++	++	++	++	
Neon	Python	+	++	+	+++	
CNTK	C++	+	+	+	++	+

Таблица 4. Сравнительный анализ библиотек машинного обучения

Языки. При начале работы с глубоким обучением лучше всего использовать фреймворк, который поддерживает язык, с которым вы знакомы. Например, Caffe (C++) и Torch (Lua) имеют привязку Python для своей кодовой базы. Однако если вы хотите использовать эти технологии, вы должны 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 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. Это не только является сильным индикатором развития инструмента, но и показывает, насколько быстро можно решить возникающие проблемы путем поиска на форуме StackOverflow или GitHub репозитории библиотеки. Необходимо подчеркнуть, что TensorFlow – это лучший фреймворк по количеству руководств, учебных материалов, а также сообществу разработчиков и пользователей.

Возможность моделирования CNN. Сверточные нейронные сети (CNN) использую для распознавания изображений, рекомендательных механизмов и обработки естественного языка. CNN состоят из набора различных слоев, которые преобразуют исходный объем данных в итоговые оценки предопределенных классов. Мы рассматриваем возможность моделирования технологий CNN 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 [15]. Для большинства тестов скорость TensorFlow [16] также была весьма сопоставима, в то время как Caffe и Theano были немного хуже.

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.

Глава 4. Программная реализация

4.1. Средства разработки платформы Android

Ввиду преимуществ платформы Android, указанных в параграфе 1.1, для разработки мобильного приложения было решено использовать Android. Вместе с тем необходимо также отметить то, что большинство людей с нарушениями зрения пользуются именно Android. (параграф 1.2)

Важной составляющей в ходе разработки приложения является выбор подходящей IDE (интегрированная среда разработки), которая зависит не только от платформы, но и собственных навыков и степени подготовленности. IDE – это комплекс программных средств, объединяющий основные инструменты, необходимые разработчикам для написания и тестирования программного обеспечения. Язык программирования Java был преобладающим при разработке приложений на Android.

Перед тем как начать программировать Java, нам необходимо установить специальное программное обеспечение — JRE и JDK, последние версии которых находятся на сайте разработчика. JRE (Java Runtime Environment) представляет собой среду выполнения и позволяет запускать приложения, написанные на Java. JDK (Java Development Kit) — это набор библиотек и инструментов для создания, компилирования и отладки программ. JDK входит в состав JRE.

Разработку приложений можно вести в среде Android Studio, IntelliJ IDEA или в Eclipse, используя при этом плагин Android Development Tools (ADT). Однако в настоящее время Google прекратила поддержку ADT для Eclipse. И теперь Android Studio является основной средой создания приложений под Android. Но некоторые разработчики могут не нуждаться в Java или просто не ладят с языками стиля C, другие разработчики хотели бы иметь единую кодовую базу для поддержки других платформ, что называется

кроссплатформенной разработкой. Xamarin считается лучшим инструментом для таких решений. В качестве языка используется C#.

Теперь перейдем к работе с Android Studio и узнаем, что это такое. Android Studio – официальная интегрированная среда разработки для операционной системы Android от Google, построенная на программном обеспечении IntelliJ IDEA от JetBrains и предназначенная специально для разработки под платформу Android [17]. IDE абсолютно бесплатна и доступна для загрузки в операционных системах на базе Windows, macOS и Linux, соответственно являясь мультиплатформенной. Так что, если вы работаете над проектом на Windows, вы легко можете перейти на Mac без каких-либо трудностей. Скачать можно с официального сайта Android Studio. Установщик включает в себя все необходимое — сама IDE, Android Software Development Kit (SDK), а также эмулятор. SDK – это набор средств, используемых для разработки приложений. Предоставляется поставщиками аппаратного и программного обеспечения. Пакеты SDK обычно состоят из интерфейсов прикладного программирования (API), примера кода, документации и т. д. В свою очередь, API – это интерфейс, который позволяет программному обеспечению взаимодействовать со внешней системой.

4.2. Архитектура приложения

Для программной реализации была выбрана среда разработки для Android приложений – Android Studio. В качестве языка программирования выступает Java.

Схема работы приложения показана на рисунке 7. Сначала пользователь запускает приложение на своем устройстве. После чего на дисплей выводится главное окно, основным объектом которого является активная тыловая камера. Следующим шагом пользователь уже может наводить на находящиеся перед собой препятствия и предметы. Приложение ищет в своей базе данный объект и далее выводит полученную информацию на экран. После

идентифицирования текст преобразуется в речь, чтобы голосовым сигналом сообщить об обнаруженном объекте. Приложение будет возвращаться к первому шагу до тех пор, пока его работа не будет прервана самим пользователем или системой.

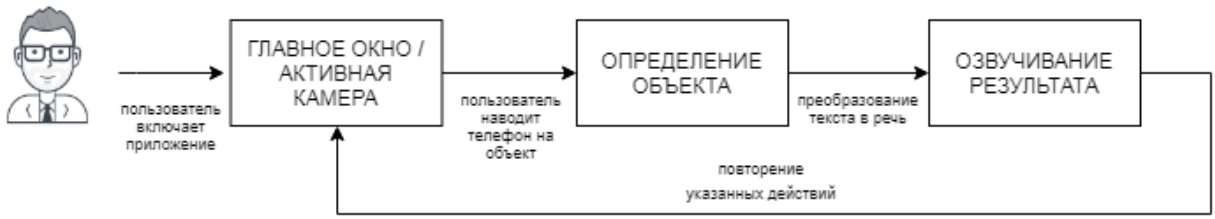


Рисунок 7. Схема работы приложения

Тем самым будет обеспечиваться безопасное передвижение пользователя в окружающем пространстве. В базу объектов будут внесены слова определенной тематической группы, характеризующие городское пространство, такие как скамейка, урна, столб, дерево, также приложение сможет идентифицировать людей.

В первую очередь, чтобы приложение могло работать с камерой устройства, нужно запросить доступ на ее использование. В манифесте AndroidManifest.xml заполняются все необходимые системе разрешения. Помимо этого, файл содержит всю основную информацию о приложении: название, минимальную версию SDK, классы Activity. Для того чтобы наше приложение научилось распознавать объекты, нам понадобится организовать классификатор изображений на основе обученной нейронной сети. Озвучивание будет происходить при помощи синтезатора речи от Google – TextToSpeech, встроенного в Android SDK и, соответственно, Android Studio. Достоинством данного синтезатора является поддержка многих языков, в том числе и русского. Как правило, TextToSpeech установлен по умолчанию на всех современных девайсах системы Android.

Для описания классов и определения связей между ними будем использовать плагин от команды JetBrains для Android Studio, называемый

simpleUMLCE. Плагин позволяет не только рисовать UML-диаграммы, но и преобразовывать проект в подробную UML-диаграмму с отображением всех связей между классами, методами и т. д. UML (Unified Modeling Language) – универсальный язык моделирования, используемый для графического описания системы [18]. Одним из наиболее популярных типов визуализации в UML является диаграмма классов. Популярность таких диаграмм среди разработчиков для документирования архитектуры программного обеспечения вызвана тем, что они достаточно хорошо описывают то, что должно присутствовать в моделируемой системе. Именно к такому типу и относится плагин simpleUMLCE. UML-диаграмма нашего приложения представлена на рисунке 8.

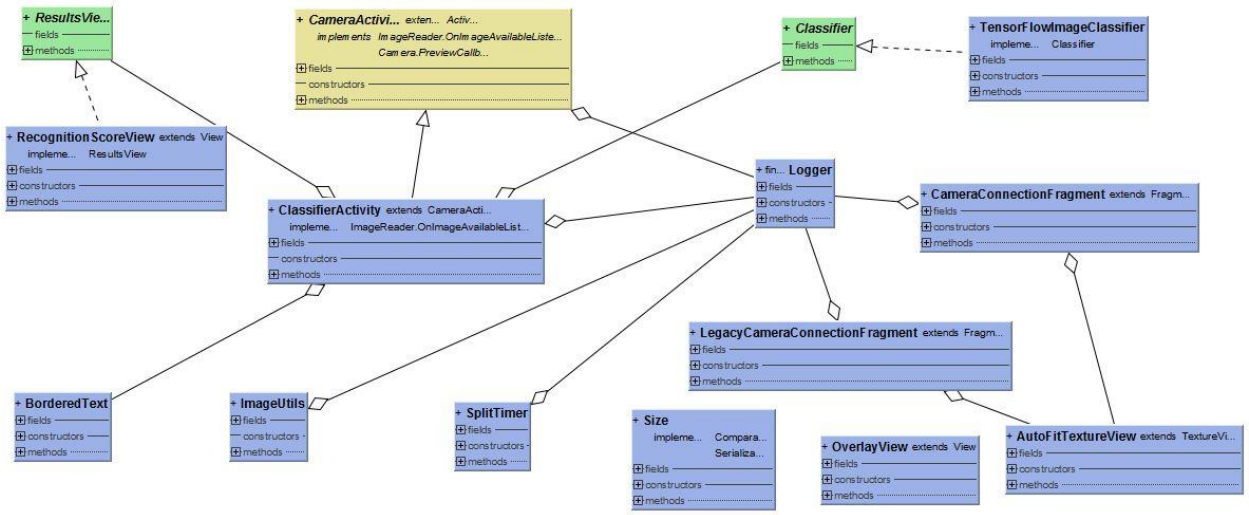


Рисунок 8. UML-диаграмма приложения

Классы в диаграмме представлены блоками, где верхний раздел содержит имя класса, ниже находятся списки полей, конструкторов и методов. Определим взаимосвязи, которые представлены на данной диаграмме. Связь ——▷, называемая ассоциацией, показывает нам, что объекты одного класса связаны с объектами другого класса. Когда класс формируется как совокупность других классов, связь называется агрегирующей между этими классами. Изображается как ◇—— . А - - - - ▷ обозначает реализацию функциональности, определенной в одном классе другим классом.

4.3. Реализация классификатора

При создании классификатора для приложения воспользуемся библиотекой машинного обучения TensorFlow. Одной из возможностей фреймворка является использование концепции трансфертного обучения для сокращения времени и сложности обучения путем перепрофилирования предварительно подготовленной модели. Для работы с TensorFlow в ОС Windows будем использовать терминал cmd с использованием языка Python.

Перед тем, как начать работу, необходимо подготовить обучающие входные данные. Для этого было создано несколько каталогов, включающих набор изображений, описывающих данный предмет. Папка должна иметь имя, которое будет у метки объекта модели. Набор данных содержит по 50 изображений каждого объекта (см. Приложение). Используемого набора будет достаточно, чтобы идентифицировать представленные объекты.

Для обучения введем следующий скрипт:

```
python retrain.py \
  --how_many_training_steps=500 \
  --output_graph=retrained_graph.pb \
  --output_labels=retrained_labels.txt \
  --image_dir=C:\Users\user\Desktop\gorod
```

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.

Файл retrain.py находится в свободном доступе в официальном репозитории TensorFlow на GitHub [19]. Этот скрипт загружает предварительно подготовленную модель Inception v3[2], удаляет старый верхний слой и обучает новый на загруженных нами изображениях.

[2] Inception v3 – предварительно подготовленная модель, построенная с помощью ImageNet

Большинство объектов было в исходных классах ImageNet[3], на которых обучалась полная сеть. Достоинство трансферного обучения 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. Это может быть не так эффективно, как полное обучение с нуля, но удивительно полезно для многих приложений, так как позволяет создавать модели со значительно сокращенными данными и временем обучения. Так как мы работаем над задачей классификации, в качестве функции активации последнего слоя нашей глубокой нейронной сети используется Softmax. Функцией потерь в данном случае выбирается перекрестная энтропия.

После 500 тренировочных шагов точность предсказания составила 96,2 процента. Протестируем полученную модель как на изображении из обучающего набора, так и на изображении взятого из сети Интернет, не из каталога. Для тестирования нам также понадобится скрипт label_image.py из репозитория.

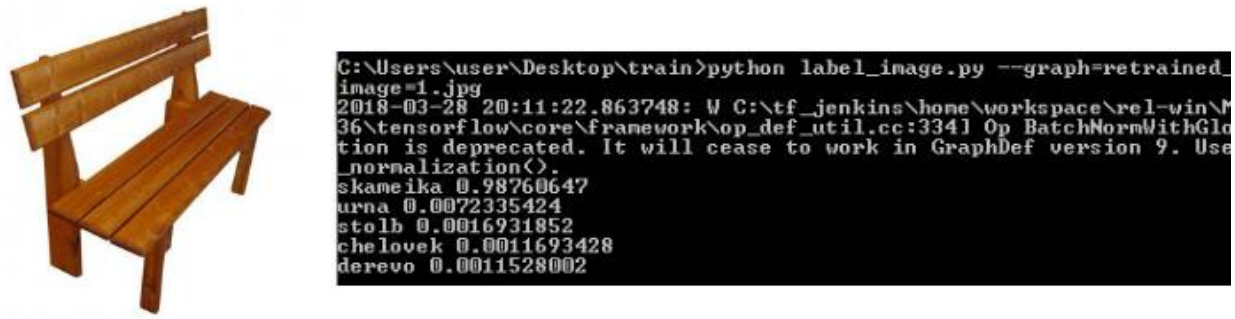


Рисунок 9. Тестирование на изображении из каталога скамеек

[3] ImageNet – база данных изображений от Google

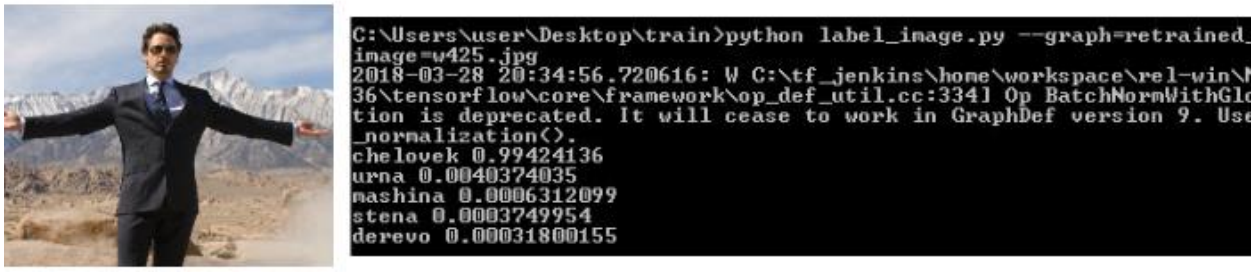


Рисунок 10. Тестирование на случайном изображении с человеком

Как показано на скриншотах (см. рисунок 9 и рисунок 10), при тестировании изображений скамейки и человека были получены числа 0,9876 и 0,9942. Это значит, что классификатор утверждает, что точность нахождения на изображениях данных объектов равна 98,8 и 99,4 процентов соответственно. И как мы видим, классификатор на самом деле прав.

Прежде чем использовать полученную модель в приложении, нужно удалить из нее неподдерживаемые слои декодирования JPEG. Для этого выполним на графе скрипт strip_unused.py из репозитория.

На выходе были получены два файла: stripped_graph – обученная модель, представленная в расширении pb, и метки retrained_labels в текстовом формате. Эти файлы размещаются в каталоге assets для дальнейшей с ними работы. В классе ClassifierActivity объявляем их путь. Там же указываем настройки для модели исходя из того, на какой модели мы обучались. В нашем случае это Inception v3.

```
INPUT_SIZE = 299;
IMAGE_MEAN = 128;
IMAGE_STD = 128;
INPUT_NAME = “Mul”
OUTPUT_NAME = “final_result”;
MODEL_FILE = “file:///android_asset/stripped_graph.pb”;
LABEL_FILE = "file:///android_asset/retrained_labels.txt";
```

- INPUT_SIZE – это размер входного изображения. Предполагается квадратное (299x299);
- IMAGE_MEAN – среднее значение;

- IMAGE STD – стандартное отклонение;
- INPUT и OUTPUT NAME – метки входного и выходного узлов.

Кадры, сделанные с камеры телефона, преобразуются ко входному размеру. И далее указанные параметры передаются в класс TensorFlowImageClassifier для классификации.

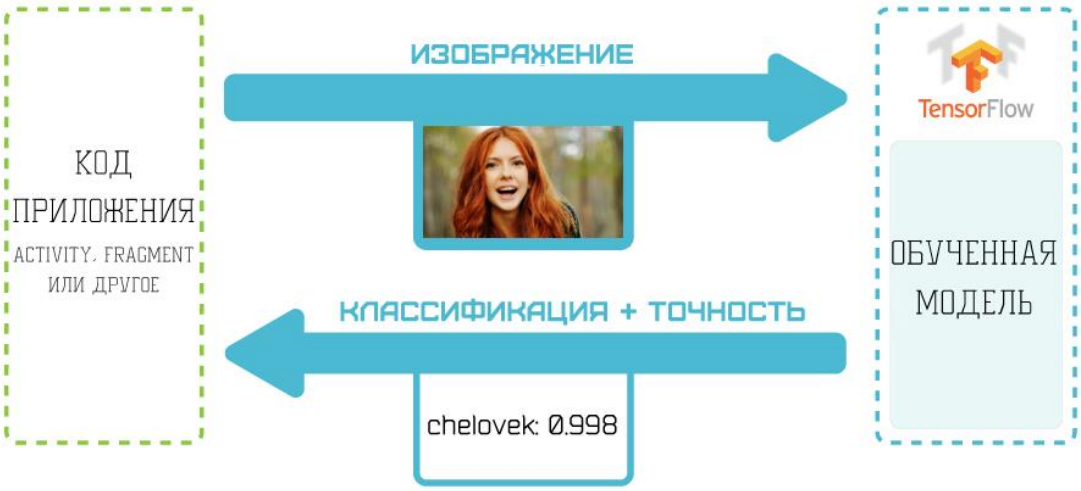


Рисунок 11. Пошаговая структура работы классификатора

В настоящее время в TensorFlow встроены специальные библиотеки для взаимодействия с моделями TensorFlow на Java. Все что нужно сделать, это включить в gradle-файл зависимость «org.tensorflow:tensorflow-android» и использовать класс TensorFlowInferenceInterface в коде проекта.

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

Рассмотрим основные моменты метода recognizeImage из TensorFlowImageClassifier:

```
@Override
public List<Recognition> recognizeImage(final float[] pixels) {
    // Копируем входные данные в TensorFlow.
    inferenceInterface.feed(inputName, pixels, new long[]{inputSize * 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;
}
```

Мы подаем данные пикселей, запускаем классификатор, а затем извлекаем выходные данные. После эти выходы сортируются для получения наивысшей уверенности и предоставляются пользователю (см. рисунок 11).

4.4. Описание функциональности

При проектировании доступного интерфейса было решено использовать один экран, состоящий из трех элементов. В первом воспроизводится видео с тыловой камеры телефона. Во втором находится полученный от классификатора результат с описанием объектов и точностью их нахождения в кадре. Дополнительно вы также можете получить статистику загрузки процессора, время вывода и посмотреть на предварительное обрезанное изображение, нажав на кнопку «Home» устройства. Простой интерфейс достигается в первую очередь тем, что от пользователя не требуется никаких тактильных действий, все происходит автоматически и без вмешательств с его стороны.

При запуске приложения включается основное окно, на котором находится активная камера. Если этот запуск происходит впервые, то пользователю нужно будет дать доступ к своей камере.

Затем пользователю предстоит расположить свой девайс перед собой. Кадры с камеры моментально передаются классификатору, который впоследствии идентифицирует содержимое. Определение занимает буквально пару сотен миллисекунд. На экране устройства отображаются возможные варианты и вероятность их нахождения от нуля до единицы.

Как только точность обнаружения классификатором какого-либо из представленных нами объектов достигает 0.9 (90 процентов), информация непременно озвучивается пользователю.

Далее указанные действия повторяются. Функционирование приложения будет продолжаться до тех пор, пока не будет прервано самим пользователем или системой.

На рисунке 12 представлен снимок, показывающий работу с приложением. Демонстрационное видео в действии можно посмотреть по ссылке [20].

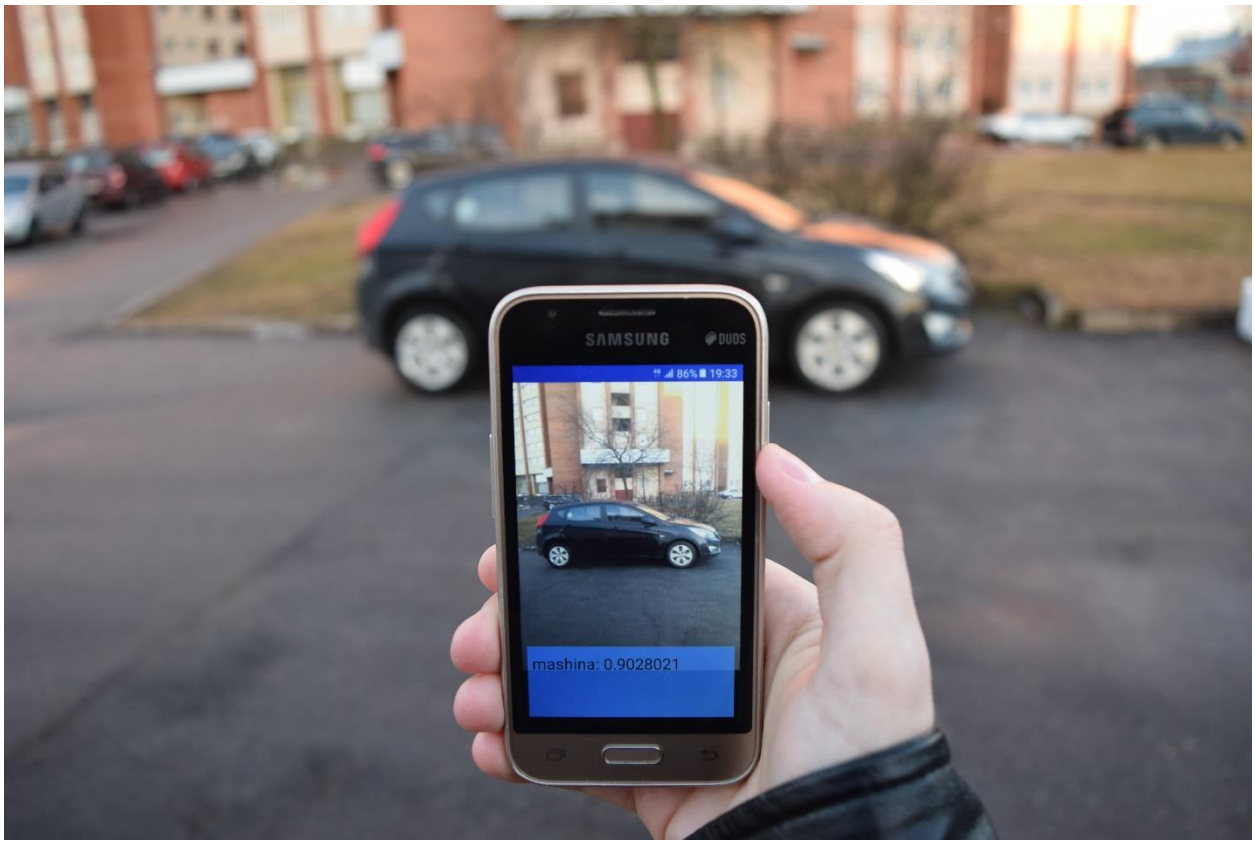


Рисунок 12. Пример работы приложения

Выводы

Мы достигли довольно точных результатов предсказывания на предложенных объектах. В категориях объектов были использованы и обнаружены: «человек», «дерево», «машина», «скамейка», «стена», «столб» и «урна». Точность определения близка к 100 процентам для неподвижного состояния и при хорошем освещении. Однако при изменении одного из факторов, таких как свет, скорость или вообще объекта из другой категорий, точность может уменьшиться. Поэтому в дальнейшем посредством сбора информации необходимо определить объекты-препятствия, в которых будут нуждаться пользователи приложения, а также реализовать функционирование при недостаточном освещении и быстром движении объекта для более безопасного ориентирования.

Достигается работа в режиме «реального времени», потому как камера устройства постоянно активна, и классификатор приложения непрерывно принимает на вход видеопоток с этой камеры. Количество сделанных кадров в секунду и время определения зависит лишь от процессора, установленном на вашем девайсе.

Поскольку приложению не требуется обращаться к посторонним серверам для получения информации, доступна возможность работы в полностью автономном режиме без подключения к сети Интернет. Информация получается напрямую из встроенной в приложение обученной модели.

В перспективе в приложение планируется добавить и поддержку других языков, выявив наиболее необходимые, исходя из опросов пользователей зарубежных стран.

После чего разработанное приложение будет опубликовано в магазине приложений компании Google – Google Play на бесплатной основе.

Заключение

Поднимая вопрос о социальной адаптации слабовидящих, важной частью которого является ориентировка и передвижение в пространстве, необходимо сказать об актуальности разработки программного продукта, соответствующего тематике данной проблемы и реализованного в соответствии с новейшими технологиями.

В рамках работы были выполнены следующие задачи:

1. Выбрана мобильная операционная система и подходящие под нее средства разработки.
2. Проведен анализ существующих приложений из сервиса «Google Play», соответствующих тематике.
3. Разработано приложение, подходящее нашим требованиям:
    - Определение объекта;
    - Голосовое озвучивание;
    - Работа в режиме «реального времени»;
    - Возможность работы офлайн;
    - Простой интерфейс;
    - Ценовая доступность.
4. Приложение было протестировано слабовидящими пользователями и показало свою работоспособность на реальных устройствах системы Android.

Список литературы

[1] Pascolini D., Mariotti S.P., Global estimates of visual impairment: 2010 // British Journal Ophthalmology, 2012. Vol. 96 P. 614-618.

[2] Как используют интернет и современные технологии люди с нарушением зрения. Исследование Яндекса. https://habrahabr.ru/company/yandex/blog/270775/

[3] Smartphone OS. https://www.idc.com/promo/smartphone-market-share/os

[4] Android Dashboards. https://developer.android.com/about/dashboards

[5] Screen Reader User Survey #7 Results. https://webaim.org/projects/screenreadersurvey7

[6] Number of Android applications. https://www.appbrain.com/stats/number-of-android-apps

[7] Arthur L. Samuel. Some Studies in Machine Learning Using the Game of Checkers // IMB Journal, July 1959. P. 210-229.

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

[9] McCulloch W.S, Pitts W. A Logical Calculus of the Ideas Immanent in Nervous Activity // Bulletin of Mathematical Biophysics, 1943. V. 5, P. 115-133.

[10] Бенджио И., Гудфеллоу Я., Курвилль А. Глубокое обучение. М.: ДМК-Пресс, 2018

[11] Воронцов К. Математические методы обучения по прецедентам (теория обучения машин), 2018 http://www.machinelearning.ru/wiki/index.php?title=Машинное_обучение_%28курс_лекций%2C_К.В.Воронцов%29

[12] Deng L., Yu, D. Deep Learning: Methods and Applications // Foundations and Trends in Signal Processing, 2014. Vol. 7, No. 3–4, P. 197-387.

[13] Николенко С., Кадурин А., Архангельская Е. Глубокое обучение. Погружение в мир нейронных сетей. СПб.: Питер, 2018

[14] TensorFlow. https://www.tensorflow.org/

[15] convnet-benchmarks. https://github.com/soumith/convnet-benchmarks/blob/master/README.md

[16] tf benchmarks. https://github.com/tobigithub/tensorflow-deep-learning/wiki/tf-benchmarks

[17] П. Дейтел, Х. Дейтел, А. Уолд. Android для разработчиков. 3-е издание. СПб.: Питер, 2016

[18] Booch G., Rumbaugh J., Jacobson I. Unified Modeling Language User Guide, (2nd Edition). Addison-Wesley. 2005.

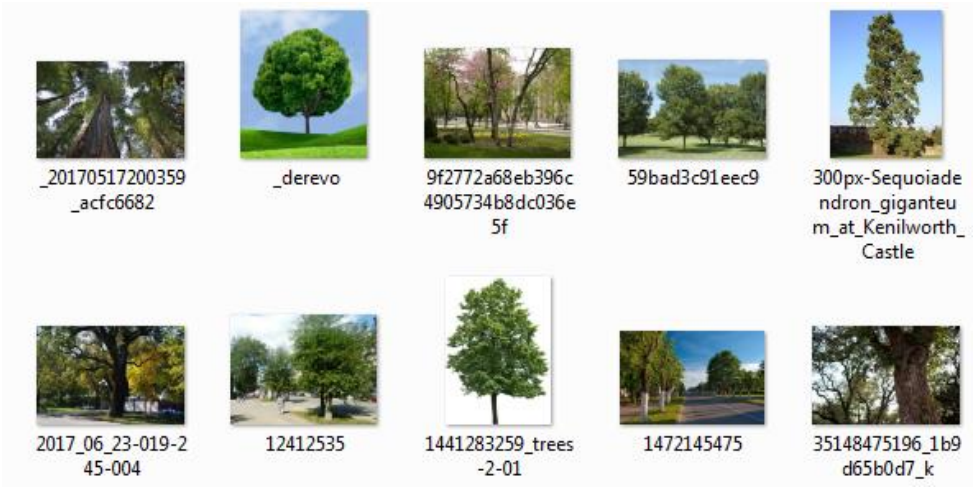
[19] TensorFlow GitHub. https://github.com/tensorflow/tensorflow

[20] BUDDY – Demo Video. https://youtu.be/_CIeIy37xQs

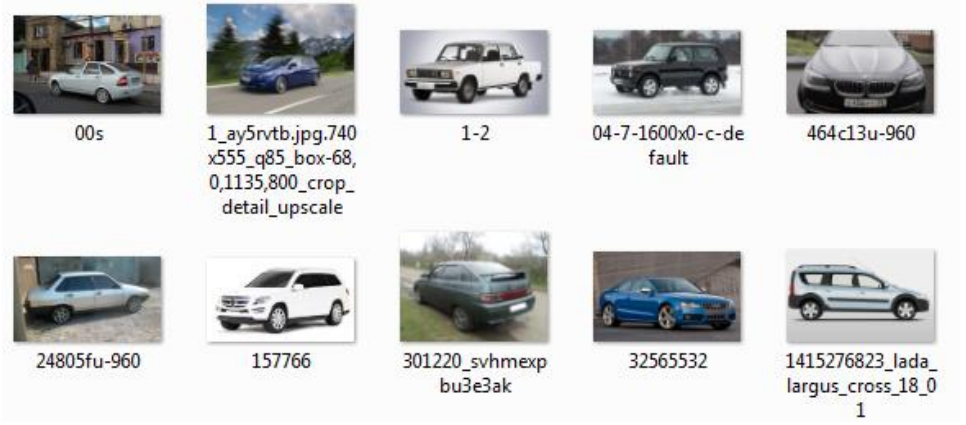
Приложение

Пример содержимого каталогов

Дерево:



Машина:



Скамейка:

Стена:

Столб:

Урна:

Человек: