МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования «Уральский федеральный университет
имени первого Президента России Б.Н. Ельцина»

ИТОГОВАЯ АТТЕСТАЦИОННАЯ РАБОТА
на тему
«Умные фотоловушки»

по дополнительной профессиональной программе –
программе профессиональной переподготовки
«Разработка приложений интернета вещей и искусственного интеллекта»

Группа в составе:

1. Слушатель Аникина И. В.
2. Слушатель Габдорахманова Е. А.
3. Слушатель Мухаметдинова В. И.

Руководитель

Старший преподаватель КВКТ ИЕНиМ Дунаева А. В.

Допустить к защите
«___» _____ 2023 г. Руководитель программы
«___» _____ 2023 г. Секретарь

Екатеринбург
2023

СОДЕРЖАНИЕ

ПАСПОРТ ПРОЕКТА

О проекте
Название проекта Умные фотоловушки с функцией распознавания животных
Цель проекта Разработать мобильное приложение, в котором будет отображаться, какие животные проходили мимо фотоловушек в режиме реального времени
Результат проекта Умная фотоловушка, которая включает камеру при детектировании движения и посылает кадры с камеры на сервер, пока есть движение. Сервер классифицирует с помощью нейронных сетей тип и количество животных на снимке и сохраняет его в базу данных. В любой момент мобильное приложение может обратиться к серверу и продемонстрировать пользователю данные о том, какие животные мимо какой фотоловушки в какое время прошли.
Критерии приемки • Наличие действующего прототипа устройства • Разработанное мобильное приложение, листинг программы • Разработанный модуль распознавания на основе технологий ИИ, листинг модуля • Настроенный облачный сервис для размещения данных, скриншоты • Примеры обработанных данных (не менее 10), скриншоты • Презентация проекта (не менее 10 слайдов, PowerPoint) • Отчет по проекту (не менее 15 стр, MS Word)
Проблема Заповедники круглогодично отслеживают количество и перемещения животных в своих подвластных территориях. Позже эта информация используется для выдачи лицензий на охоту. Сейчас используются обычные фотоловушки, которые детектируют движение и записывают видео. Лесничие раз в какое-то время проезжаются по угодиям, собирают карты памяти в

фотоловушках и заменяют их на новые. Привезенные карты памяти лесничие отсматривают вручную и делают записи о передвижениях каждого из 4 видов животных: лосей, кабанов, медведей и рысей. Это долго и дорого. Профессиональные лесничие могли потратить часы просмотра на более полезные для заповедника занятия. Для реализации проекта необходимы компетенции в трех сферах: • Интернет вещей, • Системы искусственного интеллекта, в частности компьютерного зрения, • Программирование мобильных приложений.
Тип проекта
Прикладной
Количество команд и участников (студентов)
Количество команд, которые параллельно реализуют проект 1
Количество студентов в одной команде 3
Требования к функциональным ролям студентов в проекте
Роль 1: Программист мобильного приложения Роль 2: Инженер IOT Роль 3: Программист нейронной сети и облачного сервиса
Требования к роли 1. Программист мобильного приложения • Уметь программировать на Java • Уметь выбрать фреймворк для сборки мобильного приложения Требование к роли 2. Инженер IOT • Уметь программировать микроконтроллеры • Уметь выбрать технологию передачи сообщений • Уметь подобрать компоненты в соответствии с поставленной задачей (лес, отсутствие связи, тяжелый доступ к устройству) Требования к роли 3. Программист нейронной сети и облачного сервиса • Уметь выбрать облачный сервис, подходящий под конкретную задачу • Интегрировать облачный сервис при помощи API • Уметь программировать на Python

• Уметь адаптировать готовые предобученные нейронные сети под конкретную задачу
Список формируемых компетенций
ПК-10 Решает задачи искусственного интеллекта (ИИ) *ПК-11 Применяет технологии умного производства и Интернета вещей* *ПК–12 Применяет языки программирования для решения профессиональных задач*
Требуемые от заказчика ресурсы
Кластер для обучения нейронной сети

ОПИСАНИЕ ПРОДУКТА ИЛИ УСЛУГИ

Целью данной работы является проект по построению умной фотоловушки, которая не только записывает видео материалы при детектировании движения, но и с помощью протоколов связи отправляет их на облачный сервис, который классифицирует количество особей и их принадлежность к одному из выбранных видов и сохраняет полученную и обработанную информацию. Также проект содержит мобильное приложение, которое по запросу егеря или лесничего может в любой момент в режиме онлайн предоставить записи с сервера о передвижении животных мимо каждой из фотоловушек. Таким образом, решение будет быстрым, поскольку задержка от фотоловушки до приложения займет не более пары минут, и менее трудозатратным, потому что вместо ручного рассмотрения материалов лесничие видят уже краткую сводку с обработанных фотографий.

Разработка приложения проходила в 8 этапов:

1. Анализ проблемы и поиск потребителей;
2. Обзор существующих решений;
3. Создание обучающего набора данных;
4. Эксперименты по дообучению нейронных сетей;
5. Разработка оболочки облачного сервиса;
6. Конструирование камеры;
7. Программирование камеры;
8. Разработка мобильного приложения.

При обзоре существующих решений было рассмотрено 4 архитектуры нейронных сетей, 7 общедоступных наборов данных. Был создан обучающий набор данных, содержащий в себе 100 тысяч изображений животных 4 видов: европейские лоси (лат. Alces alces), бурые медведи (лат. Ursus arctos), рыси (лат. Lynx lynx), кабаны (лат. Sus scrofa). Было проведено 11 экспериментов по дообучению рассмотренных архитектур. Был разработан облачный сервис с

использованием технологии Docker и языка программирования Python с HTTP интерфейсом и возможностью UDP взаимодействия с камерой.

Схему компонент разрабатываемого приложения можно увидеть на Рисунок *1*.

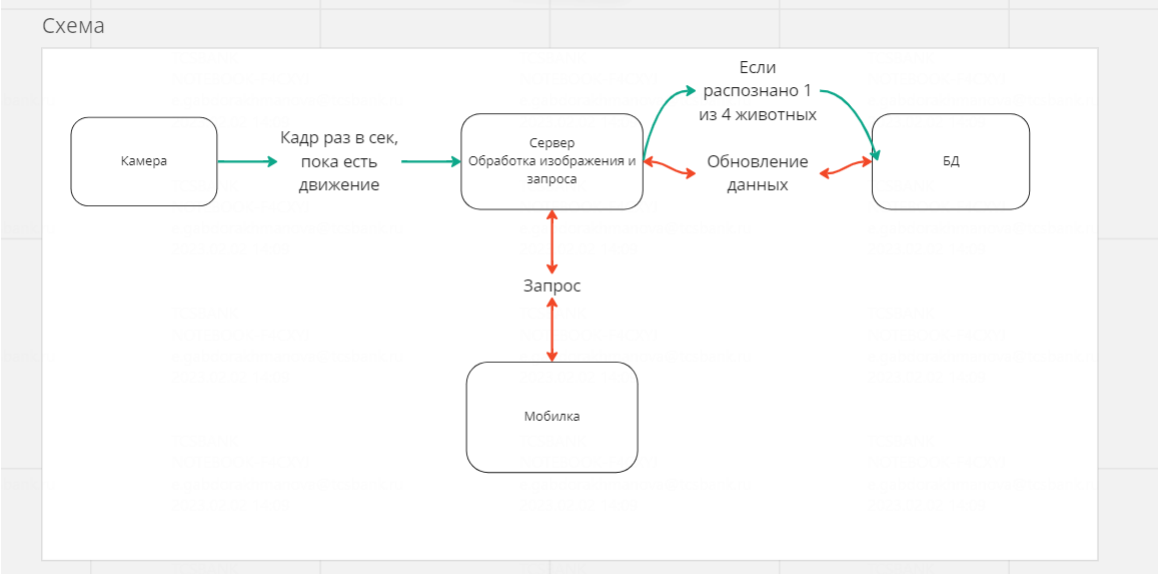

Рисунок 1. Верхнеуровневая архитектура сервера

Для построения самого устройства фотоловушки мы также просмотрели несколько схем и выбрали самую оптимальную: минимальную по комплектации и легко программируемую.

Программирование происходило на в интегрируемой среде разработки Arduino IDE, которая позволяет писать и загружать программный код на плату Arduino, использованную в работе.

Сама фотоловушка оснащена сенсором обнаружения движения, что позволяет ей активироваться только при необходимости, определяя целевые объекты. После обнаружения фотоловушка моментально срабатывает, отправляет фотографию на сервер по протоколу UDP и “засыпает” на 30 секунд.

Кроме того, мы разработали пользовательский интерфейс на базе мобильного приложения, который позволяет просматривать уже

проанализированные изображения и загружать новые изображения для анализа. Это делает наше устройство более удобным и функциональным для конечного пользователя.

АНАЛИТИКА РЫНКА

В данном разделе представлены результаты исследования опубликованных научных работ по теме классификации лесных животных по фотографиям с фотоловушек с 2021 года.

MegaDetector v5.0 2022

Пятая версия нейронной сети от Microsoft. MegaDetector [5] умеет обнаруживать животных, людей и транспортные средства на изображениях с фотоловушек. Он не идентифицирует животных на видовом уровне, он просто находит их.

Сети семейства ResNet

На данный момент сеть считается классикой задачи распознавания мультиклассовых объектов на изображениях. Существуют варианты в 18, 50, 101 и так далее остаточных блоков, отсюда сети имеют названия ResNet18 [3], ResNet50 и ResNet101 соответственно.

MobileNetv3 large

Основная особенность нейронных сетей архитектуры типа MobileNet [4] – их легкость и возможность работы на встроенных системах, в том числе их эффективность в работе на мобильных телефонах с небольшим количеством операционной памяти и большим потоком данных на вход.

EfficientNetv2 large

В настоящее время семейство нейронных сетей EfficientNet [6] являются одной из самых мощных моделей сверточных нейронных сетей (CNN).

Таблица 1 — Сравнение существующих алгоритмов на тестовых данных

	ResNet50	EfficientNetV2-L	MobileNetv3-L
Набор данных	ImageNet	ImageNet	ImageNet
mean acc	0,25	0,31	0,20

mean score	35%	73%	48%
moose acc	0	0	0
moose score	-	-	-
boar acc	0,35	0,48	0,24
boar score	30%	69%	47%
lynx acc	0,0033	0,15	0,0033
lynx score	36%	56%	88%
bear acc	0,84	0,9	0,75
bear score	38%	78%	49%
Время	10 min	4 h 20 min	8 min

Для тестирования эффективности сетей был собран набор данных из картинок-вырезок из видео, полученных с фотоловушек, расположенных в Ревдинском охотхозяйстве.

В результате исследования были получены результаты, которые будут полезны в дальнейших исследованиях по классификации животных по изображению с фотоловушки:

1. Одно из главных замечаний к набору данных ImageNet заключается в отсутствии изображений-представителей лосей (лат. Alces alces), что привело к тому, что нейронные сети, обученные на этом датасете, не смогли справиться с поставленной задачей классификации лосей.

2. В результате исследования было обнаружено, что EfficientNetv2 large лучше остальных справилась с задачей классификации. Однако, следует отметить, что этот алгоритм работает значительно медленнее конкурентов, что может быть недопустимо в некоторых условиях, например, при работе в режиме реального времени или на небольших по ресурсах устройствах: телефонах, фотоловушках и так далее.

3. Стоит отметить, что нейронная сеть MegaDetector без дополнительного обучения не подходит для задач классификации животных с фотоловушек, так как имеет недостаточное количество классов, и без дополнительного дообучения ее использование выглядит затруднительным. Однако, ее с легкостью можно использовать с любой из вышеперечисленных сетей в тандеме.

4. Проблема распознавания лосей на изображениях с фотоловушек является актуальный проблемой сообщества на стыке экологии, биологии и машинного обучения. Перед учеными стоит задача не только собрать большой и полный набор данных для обучения, но и разобраться в области машинного обучения, где текущие инструменты неэфективны или неприменимы.

Также в процессе выбора камеры для фотоловушки в нашем проекте мы обратили внимание на несколько ключевых факторов. Одним из важных аспектов было определение разрешения камеры, учитывая экономию при передаче данных по сети и хранении.

Поскольку фотоловушка будет отправлять фотографии на сервер по протоколу UDP, было важно обеспечить быструю и эффективную передачу данных, а также минимизировать использование ресурсов.

В связи с этим мы решили выбрать камеру с разрешениемв 2МП, которое всё же обеспечивало достаточное качество изображения для наших целей. Это разрешение должно быть оптимальным - достаточным для обнаружения и идентификации объектов, но не избыточным.

Остальные компоненты устройства также являются общедоступными.

ОСНОВНЫЕ ЭТАПЫ РЕАЛИЗАЦИИ ПРОЕКТА

Вся работа разбивалась на три больших блока:

1. Интернет вещей,
2. Системы искусственного интеллекта, в частности компьютерного зрения,
3. Разработка мобильного приложения.

Интернет вещей

Блок «Интернет вещей» подразумевал собой создание устройства для отправки изображения после отметки движения. Этот блок был поручен участнику команды Мухаметдиновой Валерии.

В рамках это задачи было необходимо создать умную фотоловушку, которая бы при обнаружении движения отправляла фотографии на сервер, где и происходило распознавание. То есть было две основных части: собрать устройство и написать код.

Для этого Валерия нашла статью [2], где была представлена готовая схема фото-ловушки и список материалов для нее.

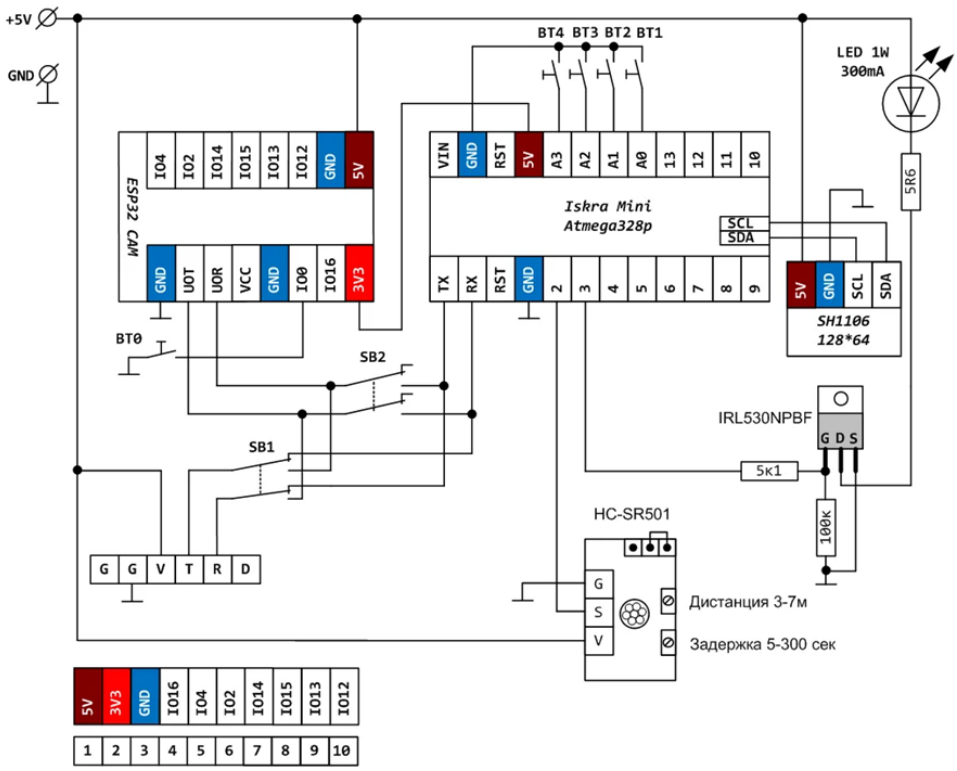

Рисунок 2. Первоначальная схема устройства

В результате работы схема была упрощена:

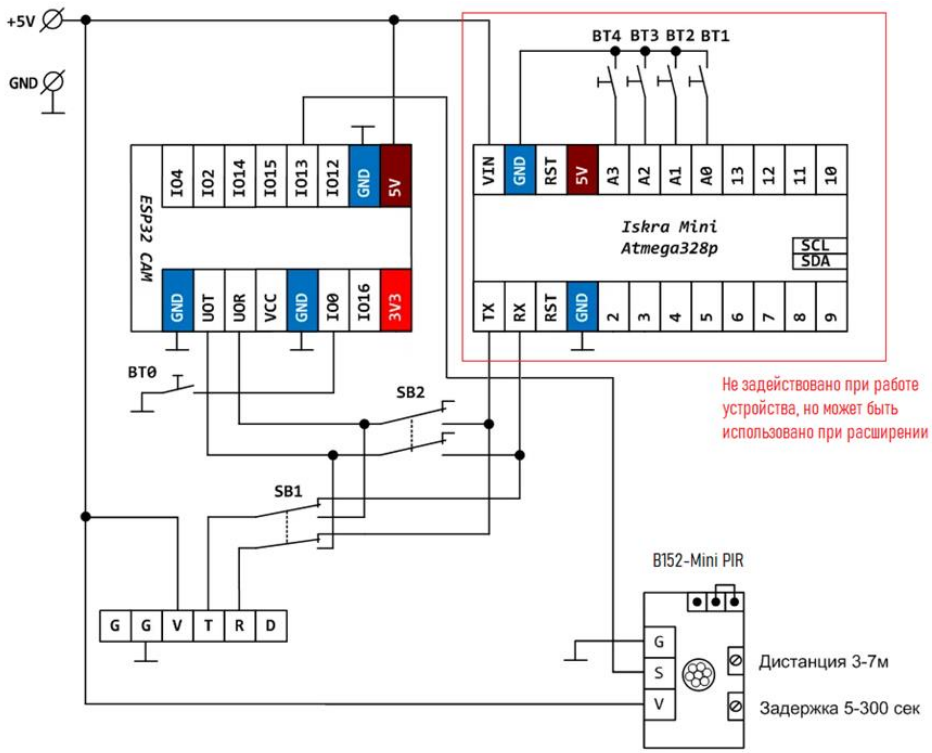

Рисунок 2. Итоговая схема устройства

Камеру ESP32 потребовалось заземлить в двух местах, поскольку работоспособность устройства без этого была бы невозможной. Светодиод и датчик температуры и влажности также были удалены из схемы, поскольку они не были заявлены в задаче. Датчик движения был подключен напрямую к ESP32-CAM, поскольку была нарушена связь ESP32-CAM ↔ Iskra Mini ↔ B152-Mini PIR, однако источник проблемы найти не удалось. В итоге, хотя место для Iskra Mini есть на плате, само устройство работает без него.

Таким образом, итоговый список компонентов и модулей представлен ниже:

- Платформа(плата) ESP32-CAM,
- камера OV2640, 2МП, угол обзора 66 градусов,
- датчик движения B152-Mini PIR,
- переключатели, 3 шт. : BT0, SB1, SB2 на схеме,
- макетная печатная плата под пайку,
- USB-UART преобразователь для прошивки платы,
- источник питания с напряжением 5V и выходным током 1-2A,
- провода.

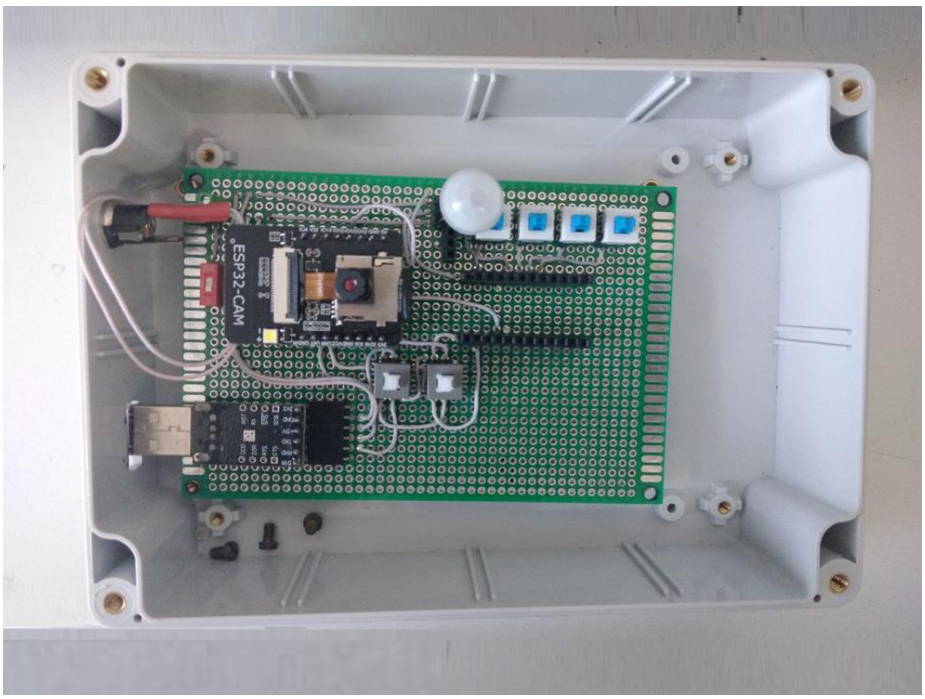

Рисунок 3. Итоговое устройство

Итоговое устройство имеет три режима работы в зависимости от 3-х переключателей:

Таблица 2. Режимы работы устройства

Режимы работы	BT0	SB1	SB2
Программирование ESP32-CAM	ON	Нажата	Отжата
Программирование Iskra Mini	OFF	Отжата	Отжата
Работа устройства	OFF	Нажата	Отжата

Весь код был написан в среде Arduino IDE, отправка изображений на сервер осуществляется через AsyncUDP по IP-адресу. Код можно посмотреть на GitHub [1].

Системы искусственного интеллекта

За данный блок проекта отвечала Габдорахманова Елена.

Создание набора данных

Тестовый набор данных представляет собой собрание видеоматериалов, которые были предоставлены Ревдинским охотхозяйством. Набор данных включает в себя 6 классов: дикий кабан, бурый медведь, лось, рысь, другие животные, пустые изображения. В список наблюдаемых животных были добавлены пустые изображения и изображения других животных, чтобы охватить все возможные случаи картинок. Общий размер тестового набора данных ~ 2 Гб или 5 061 изображение.

Итоговый набор обучающих данных для обучения включает в себя датасеты NACTI и Idaho, фотографии из Inaturalist, пользовательские наборы данных с Roboflow, а также датасет AwA2. Кроме того, в наборе данных присутствуют видеоматериалы, предоставленные Ревдинским охотхозяйством, которые не пересекаются с данными, используемыми в тестовом наборе данных. Из собранного набора данных были выделены два поднабора: набор данных для обучения нейронной сети (или train) и набор для валидации сети (он же val или valid) в соотношении 5 изображений к 1. Итоговый размер валидационного набора составляет 7,6 Гб (15 108 изображений), а размер обучающего датасета — 38 Гб (75 276 изображения).

Обучение существующих решений

Для начала экспериментов были выбраны рассмотренные ранее архитектуры: ResNet50, MobileNetv3 L и EfficientNetv2.

Проведенные эксперименты представлены списком ниже. Для каждого эксперимента бралась лучшая модель по итогам предыдущих экспериментов, то есть в эксперименте №2 архитектура ResNet50 использовалась с коэффициентом скорости обучения равным 0,0003. Для проверки эффективности полученных моделей был использован тестовый набор данных.

1. Изменение коэффициента скорости обучения (learning rate) до 0,0003 для архитектуры ResNet50. Точность на тестовом наборе данных поднялась до 0,23.

2. Удаление шедулера для каждой из архитектур. Точность на тестовом наборе данных для MobileNetv3 L поднялась до 0,45, для ResNet50 — до 0,4156.
3. Использование оригинальных параметров MobileNetv3 L для обучения из статьи о разработке сети [4]. Точность на тестовом наборе данных упала до 0,32, гипотезу пришлось отвергнуть.
4. Использование параметров обучения из статьи про трюки для распознавания животных на изображениях с фотоловушек [7]. Наибольшее значение точности на тестовом наборе данных во время обучения для MobileNetv3 L было равно 0,4576, для ResNet50 — до 0,4191. Было принято решение продолжить эксперимент и увеличить количество эпох.
5. Увеличение количества эпох до 100 и наблюдение за провалами и подъемами графиков, изучение максимально возможного качества ранее полученных моделей. ResNet50 + AdamW с lr = 0,00001 на ~35 эпохе показала на тестовом наборе данных 0,5144 точности, MobileNetv3 L на ~23 эпохи показала на тестовом наборе 0,4603 точности.
6. Перебор коэффициента скорости обучения для MobileNetv3 L в 5 случаях: от 0,1 до 0,00001. Хуже всех справилась сеть с learning rate = 0,1 и показала 0,3185 точности на тестовом наборе данных, далее качество сетей на тестовом наборе данных возрастает с уменьшением скорости обучения.
7. Изменение размера архитектуры ResNet. Проведены эксперименты с вариантами из 18 и 152 слоями. За 100 эпох ResNet152 показала точность 0,4168 на тестовом наборе данных, ResNet18 — 0,4492.
8. Смена аугментаций: удаление отражения по вертикали и добавление поворота для архитектуры ResNet50. Данная конфигурация сети показала в пике точность 0,5188 на тестовом наборе данных и, в

отличие от других экспериментов с использованием архитектуры ResNet50, достигла пика точности за 7 эпох, в сравнении с 30+ эпохами.

9. Поиск оптимального угла поворота при аугментации. Было проведено 9 запусков архитектуры ResNet50 с шагом в 10 градусов: с вероятностью 50% изображение было повернуто от 10 до 90 градусов равновероятно по часовой стрелки или против нее. Каждый из запусков имел пик точности на тестовом наборе на 6-7 эпохе, затем точность падала и стабилизировалась в районе 0,41. Лучше всех справилась нейронная сеть с поворотом на 90 градусов и получила точность на тестовом наборе данных равную в пике 0,5188, худшие показатели были у сети с поворотом на 10 градусов — 0,46 точности.
10. Добавление цветовой аугментации путем рандомизированного изменения значений hue, saturation и value исходного изображения, представленного в цветовой модели HSV. Были исследованы вариации с изменением hue в пределах 20 пунктов, saturation — в пределах 30 пунктов, value — в пределах 20 пунктов, а также вариант, объединяющий все три изменения. Между вариациями разницы в точности на тестовом наборе найдено не было. Общая точность на тестовом наборе упала до 0.47, принято решение не использовать данную аугментацию для итогового решения.
11. Применение Test-Time Data Augmentation (в сокращении TTA) [8]. TTA — метод, который может повысить производительность модели за счет применения аугментаций во время вывода. Предсказание делается на основе нескольких измененных версий одного и того же изображения, а затем берется мода прогноза для получения более высокой общей точности. В отличие от других экспериментов, нет необходимости вносить никаких изменений в модель, поэтому ее можно применить к уже обученной модели.

В результате проведенных экспериментов была получена итоговая модель, которая будет помещена на сервис классификации животных по изображениях с фотоловушек. Сеть архитектуры ResNet50 с оптимизатором AdamW и коэффициентом скорости обучения равным 0,00001, а также шедулером CosineAnnealingLR и аугментациями отражения по горизонтали и поворота до 90 градусов. Для предсказания используется TTA с аналогичными аугментациями отражениями и поворота. Тепловая карта работы полученной нейронной сети на тестовом наборе данных, а также графики обучения представлены ниже.

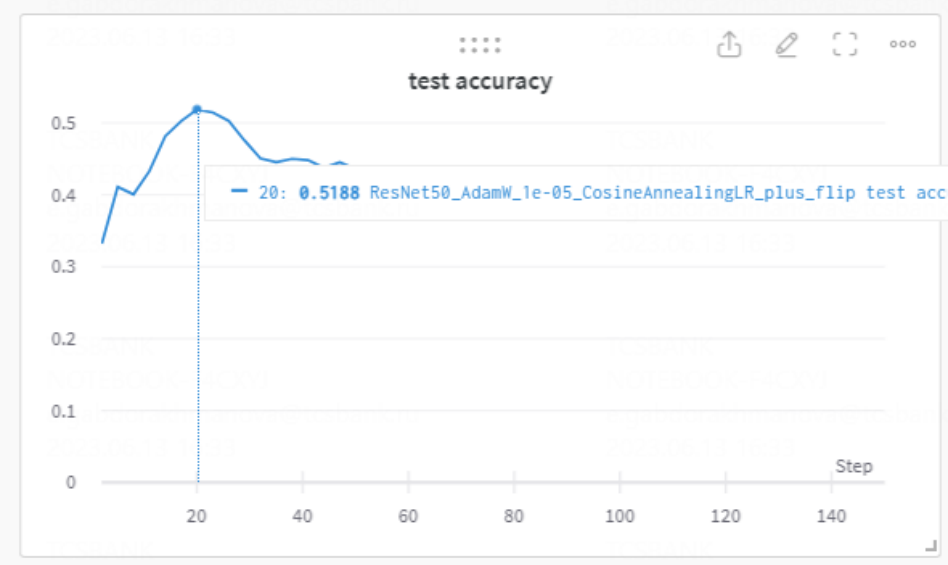

Рисунок 3. График точности итоговой нейронной сети на тестовом наборе данных

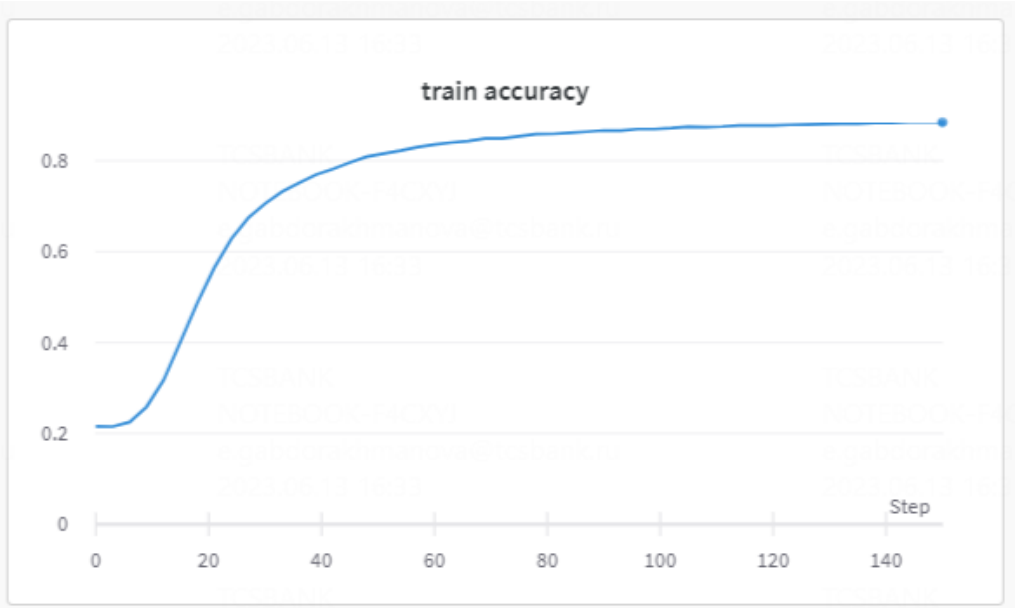

Рисунок 4. График точности итоговой нейронной сети на обучающем наборе данных

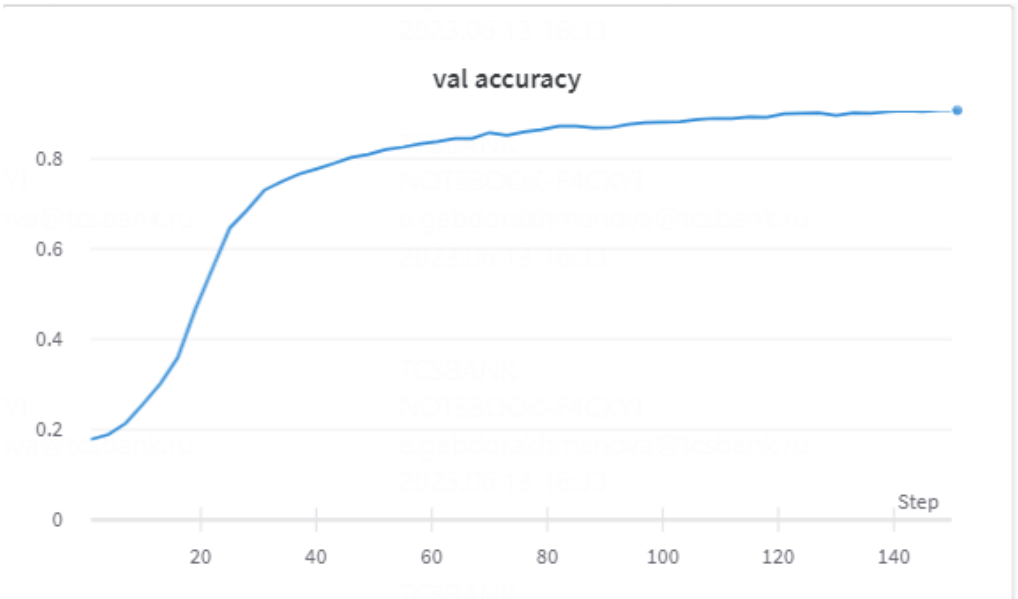

Рисунок 5. График точности итоговой нейронной сети на валидационном наборе данных

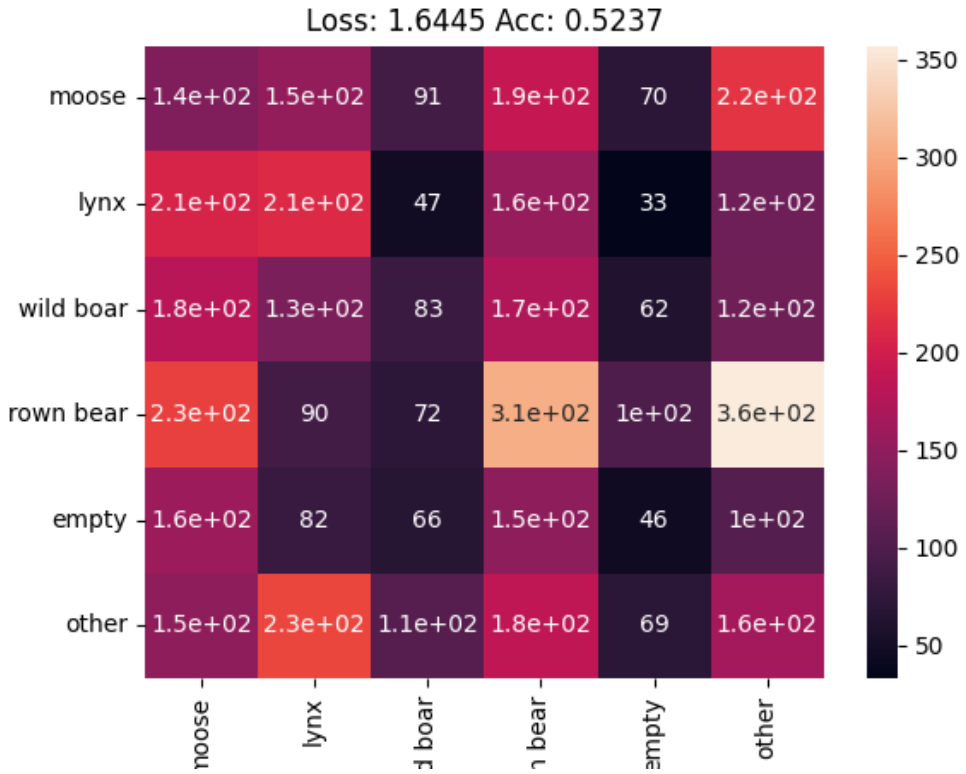

Рисунок 6. Тепловая карта классификации итоговой нейронной сети

Для более детальной работы над данной задачей предлагается объединить классическую сеть, которая определяет класс животного, с сегментацией животного на изображении через сеть MegaDetector. Такой подход позволит более точно определить класс животного на изображении. Изображение сначала пройдет сегментацию через MegaDetector, после чего будут получены максимальные габаритные прямоугольники, которые определяют животное.

Алгоритм поиска максимального габаритного прямоугольника животного проходит по всем найденным с помощью MegaDetector габаритным прямоугольникам. Если площадь пересечения двух прямоугольников больше 60% площади наименьшего из них, то считается, что это прямоугольники одного животного. По двум пересекающимся прямоугольникам строится третий — описанный вокруг пересекающихся. После прохождения по всем сочетаниям прямоугольников составляется результирующий список габаритных треугольников, а также определяется максимально возможное количество животных на изображении.

Если при обработке потока с камеры или видеоряда количество обнаруженных животных по габаритным прямоугольникам меньше текущего максимума животных на изображении, то текущий кадр отбрасывается. Это возможно благодаря повышению порога детектирования MegaDetector до 0.4, что уменьшает количество ложноположительных срабатываний, которые критичны в представленном процессе распознавания, поскольку невозможно в процессе обработки наверняка определить: это ложноположительное срабатывание или особь ненадолго мелькнула в кадре. С другой стороны, ложноотрицательные срабатывания для текущего процесса не являются критичными, поскольку на вход поступает поток изображений или видеоряда, и у детектора есть до 30 кадров на обнаружение животного. Кроме того, рассматриваемые животные крайне редко пересекаются в дикой природе, поэтому допустимо пренебречь появлением нового вида в кадре и считать, что животные одного вида заходят в кадр и выходят из него.

Затем классическая сеть определит класс полученного животного на максимальных габаритных прямоугольниках. Таким образом, комбинируя две сети, мы сможем точнее определять класс животного на изображении. Примеры работы алгоритма представлен ниже.

Рисунок 7. Корректное предсказание каскадной нейронной сети на изображении лося

Рисунок 8. Корректное предсказание каскадной нейронной сети на изображении рыси

После проведенных экспериментов итоговое качество каскадной нейронной сети составило 73,65% precision при сегментации и 52,37% accuracy при классификации. Помимо этого, было выявлено, что классификация проходит лучше и качественнее на полном изображении или его фрагменте, содержащем окружающую среду, нежели на прицельных габаритных прямоугольниках.

Размещение нейронной сети в облачном хранилище

Создано приложение с двумя серверами: REST API сервер для взаимодействия с мобильным приложением и UDP сервер для взаимодействия с камерой.

REST API сервер с двумя методами: POST (записать картинку) и GET (отдать данные). По первому методу сервер получает картинку, дает предсказание, сохраняет в базу. По второму методу сервер получает из базы, применяет фильтры и возвращает информацию. Реализован Swagger для удобства разработки мобильного приложения.

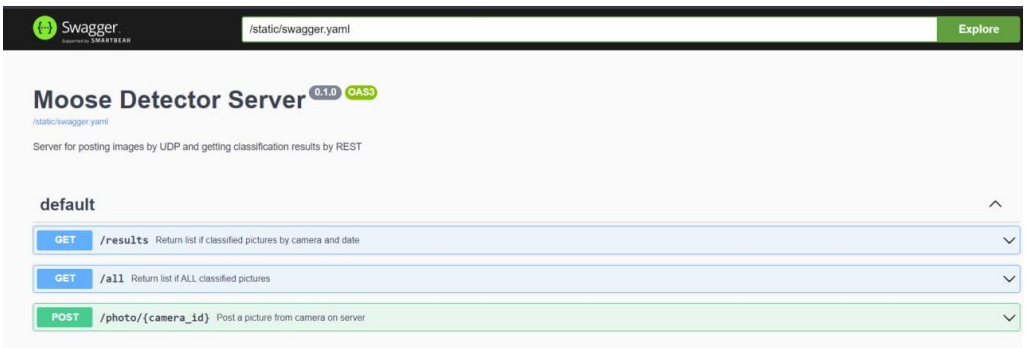

Рисунок 9. Swagger сервиса

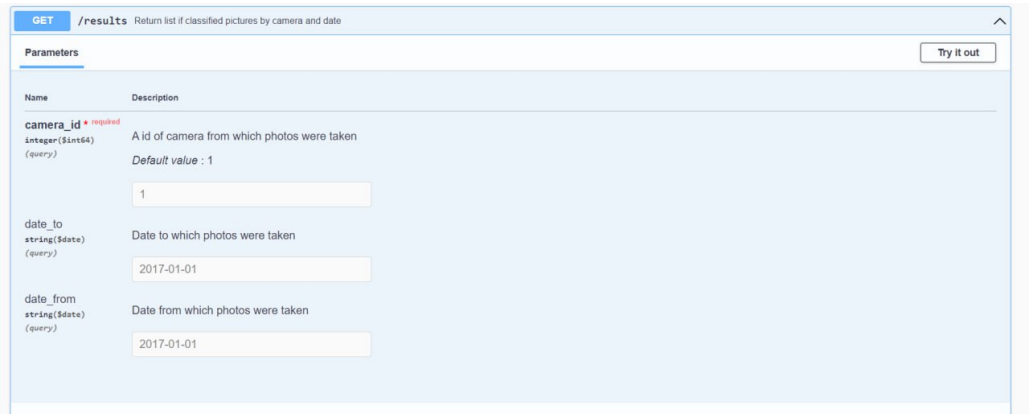

Рисунок 10. Пример описания HTTP метода в Swagger

Также для удобства пользователя реализована страница администратора, с которой можно загружать изображения и видео с помощью веб-сервиса. Это создано для использования сервера отдельно от самостоятельно сконструированной камеры.

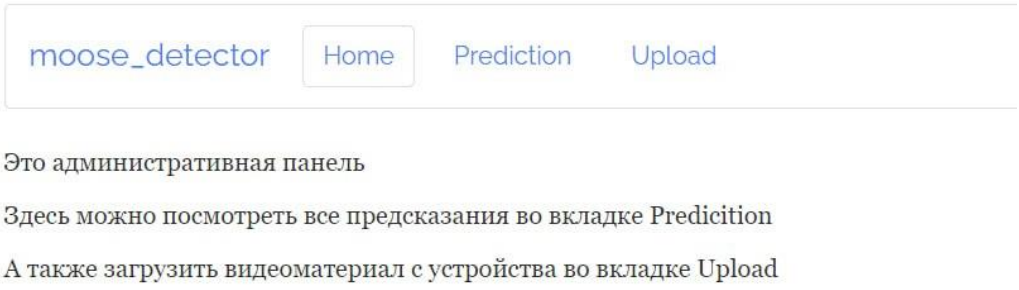

Рисунок 11. Главная страница административной панели сервиса

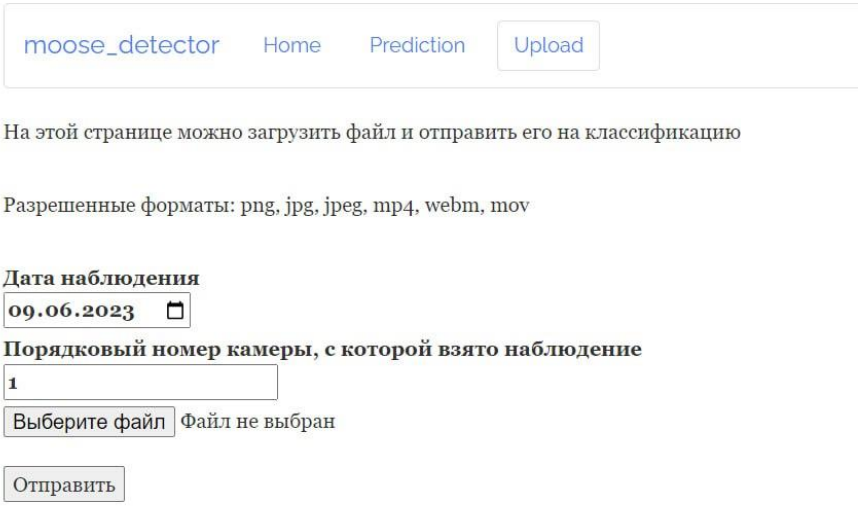

Рисунок 12. Страница загрузки видео- и фотоматериалов на сервис

UDP сервер ждет подключения с камеры, записывает время начала стриминга по каждой камере и принимает изображения. Когда камера не шлет изображения больше 30 секунд, сервер берет фото с наибольшим количеством животных на нем и сохраняет в базу вместе со временем начала и конца стриминга.

Используется база данных MySQL, Docker для развертывания и язык программирования Python, так как на данном языке существует большое количество вспомогательных библиотек для машинного обучения.

Разработка мобильного приложения

Данным блоком занималась Аникина Инна.

В ходе создания андроид-приложения были выполнены следующие задачи:

- Разработка экрана входа и проверка учетных данных пользователя;
- Создание главной страницы с доступом к страницам «Загрузка фото на сервер», «Изображения» и «Поиск изображений», а также с боковым меню со вкладкой «Выйти»;

- Разработка страницы «Загрузка фото на сервер», позволяющей пользователю загружать изображения на сервер;
- Разработка страницы «Изображения», на которой запрашивается и выводится список всех изображений и данных о них с сервера;
- Разработка страницы «Поиск изображений», на которой запрашивается с сервера и выводится список изображений и данных о них, отфильтрованный по номеру камеры и дате снимков;
- В приложении был решен вопрос обмена данными между сервером и клиентом при помощи библиотеки OkHttp. Для этого были созданы классы, которые описывают модель данных и сервисы, которые предоставляют доступ к API Flask сервера;
- Также была решена задача навигации приложения. Для этого было создано несколько Activity, включая экран входа, главную страницу, страницу загрузки и страницу с результатами. Кроме того, была создана логика перехода между страницами и передача данных между ними;
- Задача создания UI приложения была решена при помощи XML разметки и стилей. Были созданы макеты для каждой страницы, кнопки, текстовые поля и другие элементы интерфейса. Также были добавлены обработчики событий для различных элементов, чтобы сделать приложение интерактивным.

Экраны приложения:

Рисунок 13. Вход

Рисунок 14. Главный экран

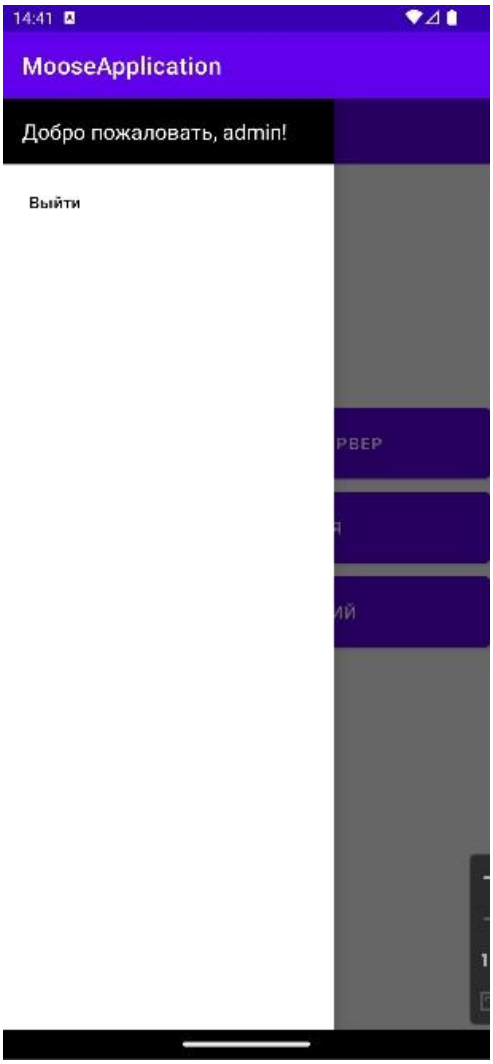

Рисунок 15. Боковое меню

Рисунок 16. Загрузка на сервер

Рисунок 17. Поиск изображений

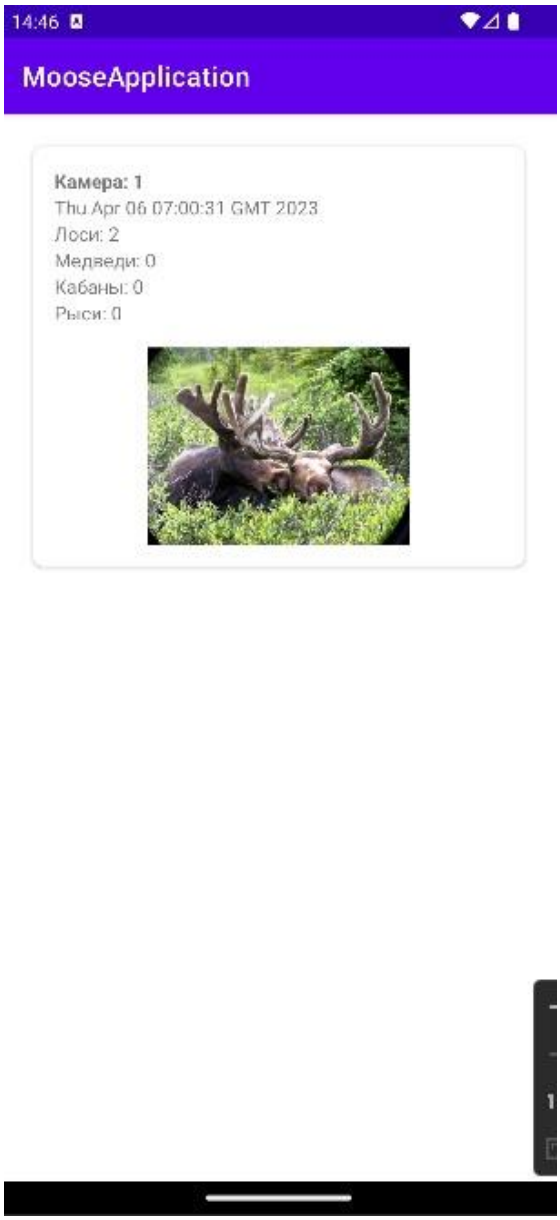

Рисунок 18. Результаты поиска

Рисунок 19. Все изображения

ОРГАНИЗАЦИЯ РАБОТ В ПРОЕКТЕ

Роли в нашем проекте распределены следующим образом:

- Габдорахманова Елена – разработка модуля распознавания на основе технологий ИИ, разработка сервера для взаимодействия камеры, нейросетевого модуля и мобильного приложения.
- Мухаметдинова Валерия – разработка прототипа умной фотоловушки: сборка устройства, написание кода.
- Аникина Инна – разработка мобильного приложения.

В ходе работы мы асинхронно выполняли задачи, регулярно созванивались и обсуждали прогресс внутри команды или с руководителем и заказчиком.

РАЗДЕЛ ПО СПЕЦИФИКЕ ПРОГРАММЫ

Наша программа – это умная фотоловушка с функцией распознавания животных. В целом проект состоит из четырёх основных частей:

- IoT
- Модуль искусственного интеллекта
- Android-приложение
- Сервер

Для того, чтобы понять специфику программы, дадим краткое описание каждой из частей.

1. IoT. Код для платы был написан в среде Arduino IDE на Си подобном языке. Arduino IDE - это интегрированная среда разработки, которая облегчает процесс написания кода, компиляции и загрузки его на различные платы, включая плату ESP32-CAM. В данном случае, было необходимо запрограммировать плату на:

- инициализацию камеры при включении,
- подключение к сети через встроенный модуль Wi-Fi на плате ESP32-CAM,
- принятие сигнала с датчика движения и последующей отправки фотографии на сервер по протоколу UDP.

Таким образом, общение с сервером происходит посредством отправки изображений по протоколу UDP через wi-fi модуль после обнаружения движения.

2. ИИ. В качестве итоговой модели выбрана нейросеть на основе архитектуры ResNet50 в совокупности с сетью для сегментации MegaDetector. Изображение сначала обрабатывается сетью MegaDetector, что позволяет получить максимальные габаритные прямоугольники, определяющие животное, после чего уже сеть для классификации определит непосредственно класс животного.

3. Android-приложение. Мобильное приложение написано на языке Java для ОС Android. Оно содержит интерфейс для загрузки изображений на сервер для определения и классификации имеющихся на нем животных, просмотра информации об уже классифицированных изображениях. Приложение обменивается данными с сервером с помощью HTTP-запросов.
4. Сервер. Приложение написано на языке Python с использованием фреймворка Flask. Сервер является связующим звеном между вышеперечисленными модулями.

Заключение

В результате работы над проектом:

1. Собран новый набор для обучения нейросетей размером около 100 тысяч изображений 4 видов животных: европейские лоси, кабаны, бурые медведи и европейские рыси.
2. Создан облачный сервис, имеющий два режима работы: через подключение фотоловушки с функцией отправки по сети и мобильного приложения или через загрузку собранного видеоряда с компьютера или ноутбука. Точность определения животных на данном сервисе составляет 73,6% распознавания и 52,4% классификации.
3. Дообучены нейронные сети для классификации объектов: ResNet50, Efficientv2-L, Mobilev3-L. Добавлен новый класс — европейский лось. Точность классификации поднята с 0.297 до 0,5237.
4. Собрана фотоловушка, включающая в себя камеру с разрешением 2МП, а также датчик движения, необходимый для активации устройства при обнаружении объектов. Фотоловушка была сконфигурирована для отправки фотографий на сервер по протоколу UDP, обеспечивая быструю и надежную передачу данных.
5. Было разработано мобильное приложение для Android для использования сервиса. Приложение позволяет загружать изображения на сервер, а также просматривать результаты классификации.

Библиографический список

1. Репозиторий кода работы камеры. URL: https://github.com/ValeriaIM/ESP32-Arduino-UDP/tree/main
2. Статья, взятая за основу для камеры. URL: https://dzen.ru/a/YNtYnXQDryKvyJoz
3. Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun (2015). Deep Residual Learning for Image Recognition.
4. Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, Hartwig Adam (2019). Searching for MobileNetV3.
5. Описание инструмента MegaDetectorv5. URL: https://github.com/microsoft/CameraTraps/blob/main/megadetector.md#megadetector-v50-20220615 (дата обращения: 01.05.2023)
6. Mingxing Tan, Quoc V. Le (2021). EfficientNetV2: Smaller Models and Faster Training.
7. Fagner Cunha, Eulanda M. dos Santos, Juan G. Colonna. (2023). Bag of Tricks for Long-Tail Visual Recognition of Animal Species in Camera-Trap Images.
8. Murat Seckin Ayhan and Philipp Berens. (2018). Test-time data augmentation for estimation of heteroscedastic aleatoric uncertainty in deep neural networks.
9. Developer guides. URL: https://developer.android.com/guide (дата обращения: 29.04.2023)
10. OkHttp Android Example Tutorial. URL: https://www.digitalocean.com/community/tutorials/okhttp-android-example-tutorial (дата обращения: 02.05.2023)