

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

Профиль: «Программное обеспечение средств вычислительной
техники и автоматизированных систем»

БАКАЛАВРСКАЯ РАБОТА

на тему

**НЕЙРОСЕТЕВОЙ ДЕТЕКТОР ЛЮДЕЙ НА
ИЗОБРАЖЕНИЯХ, СНЯТЫХ С БПЛА**

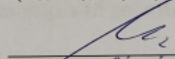
Студент

 06.06.24

Мирясов Никита Андреевич

(подпись, дата) (ФИО полностью)

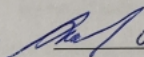
Руководитель

 06.06.24

М.А. Митрохин

(подпись, дата) (фамилия, инициалы)

Нормоконтролёр

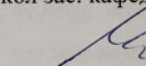
 06.06.24

А.С. Бычков

(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 27.05.24 № 11)

Заведующий кафедрой

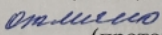


М.А. Митрохин

(подпись)

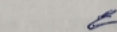
(фамилия, инициалы)

Работа защищена с отметкой



(протокол заседания ГЭК от 10.06.24 № 19)

Секретарь ГЭК



Е.И.Гурин

(подпись)

(фамилия, инициалы)

Пенза, 2024

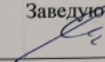
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»

Заведующий кафедрой

 М.А. Митрохин

«19» 09 2024г.

**ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ**

1. Студент Мирясов Никита Андреевич гр. 20ВВП2 факультета вычислительной техники.

2. Тема работы «Нейросетевой детектор людей на изображениях, снятых с БПЛА».

Тема утверждена приказом ПГУ № 405/0 от "19" 09.2024 г.

3. Руководитель бакалаврской работы к.т.н., доцент кафедры ВТ ПГУ Митрохин М.А.

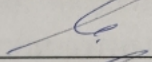
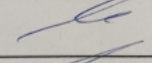
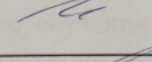
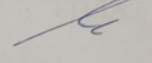
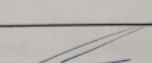
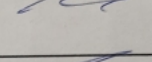
4. Задание на работу (назначение разработки, исходные данные и т.п.)

Исследовать возможность использования различных архитектур искусственных нейронных сетей в задаче обнаружения объектов на изображениях, полученных с беспилотных летательных аппаратов. Разработать приложение, которое будет включать в себя предварительную обработку данных, обучение модели искусственной нейронной сети и запуска ее на целевом устройстве. Язык программирования Python, среда разработки PyCharm. Необходимо выбрать наиболее подходящую архитектуру для решения поставленной задачи и исследовать ее в целях достижения лучших результатов в поставленной задаче.

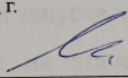
5. Перечень подлежащих разработке вопросов.

Требуется произвести анализ предметной области, архитектур моделей машинного обучения и фреймворков для работы с ними. Выбрать технологии разработки.

6. Календарный график выполнения работы

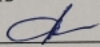
№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ архитектур моделей машинного обучения	32	19.04.2024- 22.04.2024	
2	Анализ инструментов разработки	32	23.04.2024- 27.04.2024	
3	Подготовка набора данных	65	28.04.2024- 02.05.2024	
4	Разработка программного обеспечения для проведения экспериментов	94	03.05.2024- 23.05.2024	
5	Проведение экспериментов с целью исследования параметров и гиперпараметров модели	49	24.04.2024- 27.05.2024	
6	Оформление выпускной работы	32	28.05.2024- 31.05.2024	

Дата выдачи задания "10" 04.2024 г.

Руководитель бакалаврской работы  М.А. Митрохин
(подпись, дата)

Задание к исполнению принял студент  19.04.2024 Н.А. Мирясов
(подпись, дата)

Работу к защите допустить

Декан факультета  06.06.2024 Л.Р.Фионова
(подпись, дата)

Реферат

Бакалаврская работа содержит 66 л., 15 рис., 4 табл., 51 источн., 3 прил.
НЕЙРОННЫЕ СЕТИ, ДЕТЕКТОР, ОБНАРУЖЕНИЕ ОБЪЕКТОВ, МОДЕЛИ
МАШИННОГО ОБУЧЕНИЯ, PYTHON, YOLOV8, БЕСПИЛОТНЫЕ
ЛЕТАТЕЛЬНЫЕ ОБЪЕКТЫ

Цель работы – исследование архитектур моделей машинного обучения в области обнаружения объекта на изображениях, полученных с БПЛА, и разработка нейросетевого детектора на основе одной из архитектур.

Объектом исследования является информация, которая позволит эффективно применять модели YOLOv8 в поставленной задаче.

В результате проведенной работы исследованы параметры и гиперпараметры архитектуры моделей YOLOv8, а также разработано приложение для предварительной обработки данных и запуска модели.

Результаты данной работы были на этапе разработки и изготовления опытного образца в составе работ АО НПП «Цифровые решения».

				ПГУ 1.090301.33.001 ПЗ			
Изм	Лист	№ докум.	Подпись	Дата			
Разраб.		Мирясов Н.А.		06.06.24	Литера	Лист	Листов
Пров.		Митрохин М. А.		06.06.24		2	
Нейросетевой детектор людей на изображениях, снятых с БПЛА. Пояснительная записка					ФВТ, каф. ВТ гр. 20ВВП2		
Н. Контр.		Бычков А.С.		06.06.24			
Утв.		Митрохин М.А.		06.06.24			

Содержание

Реферат	4
Перечень сокращений	Ошибка! Закладка не определена.
Введение.....	7
1 Обзор предметной области.....	9
1.1 Анализ темы.....	9
1.2 Анализ решений	12
1.3 Анализ существующих наборов данных	17
1.4 Постановка задачи	19
2 Выбор технологий разработки.....	21
2.1 Выбор программной библиотеки	21
2.1.1 TensorFlow	21
2.1.2 Keras	21
2.1.3 PyTorch.....	21
2.2 Выбор IDE.....	22
2.2.1 Visual Studio Code	22
2.2.2 Jupyter Notebook.....	22
2.2.3 PyCharm.....	22
2.3 Выбор инструмента аннотирования данных.....	23
2.3.1 LabelStudio	23
2.3.2 CVAT.....	23
2.4 Выбор инструмента версионирования данных	24
2.4.1 Amazon S3.....	24
2.4.2 Pachyderm.....	25
2.4.3 Delta Lake	25
2.4.4 LakeFS	25
2.5 Выбор инструмента версионирования моделей	26
2.5.1 Weights & Biases.....	26
2.5.2 Git.....	27
2.5.3 Git LFS.....	28

2.5.4 DVC	28
2.5.5 MLflow	28
2.6 Выбор инструмента управления виртуальным окружением.....	29
2.6.1 Venv	29
2.6.2 Pipenv.....	30
2.6.3 Conda	30
3 Набор данных	31
3.1 Предварительная обработка данных.....	32
3.2 Описание набора данных	40
4 Программная реализация.....	42
4.1 Описание архитектуры YOLOv8.....	42
4.2 Описание функций YOLOv8.....	44
4.2.1 Predict	44
4.2.2 Val.....	45
4.2.3 Train	45
4.3 Проектирование ПО.....	46
4.4 Архитектура разрабатываемой системы.....	47
5 Эксперименты.....	49
5.1 Настройка виртуального окружения.....	49
5.2 Метрики.....	49
5.3 Сравнение моделей семейства архитектур YOLOv8	52
5.4 Оборудование	52
5.5 Выбор оптимизатора.....	53
5.6 Аугментация входных данных	55
5.7 Результаты на конечном устройстве.....	56
6 Безопасность жизнедеятельности.....	58
6.1 Вред здоровью при работе с компьютером.....	58
6.2 Снижение вредного воздействия на здоровье при работе с ПК.....	58
6.3 Техника безопасности при работе с компьютером	59
Заключение	60

Список использованных источников	61
Приложение А – UML-диаграммы	67
Приложение Б – Листинг программы	72
Приложение В – Презентация.....	77

					ПГУ 1.090301.33.001 ПЗ	Лист
						5
Изм.	Лист	№ докум.	Подпись	Дата		

Перечень сокращений и обозначений

В настоящей работе применяют следующие сокращения и обозначения

БПЛА – беспилотный летательный аппарат

ПО – программное обеспечение

AP (Average Precision) – Средняя точность

CE (Cross-Entropy Loss) – Функция потерь перекрестной энтропии

CSP (Cross-Stage Partial Connections) – Частичные связи между этапами

DVC (Data Version Control) – Управление версиями данных

FL (Focal Loss) – Очаговая потеря

FLOPS (Floating Point Operations Per Second) – Операции с плавающей запятой в секунду

FN (False Negative) – Ложно отрицательный

FP (False Positive) – Ложно положительный

FPN (Feature Pyramid Network) – Пирамида признаков

FPS (Frames Per Second) – Кадры в секунду

IDE (Integrated Development Environment) – Интегрированная среда разработки

IoU (Intersection over Union) – Пересечение по объединению

lr (learning rate) – Коэффициент скорости обучения

LFS (Large File Storage) – Хранилище больших файлов

mAP (mean Average Precision) – Средняя точность по среднему значению

PAN (Path Aggregation Network) – Сеть агрегации путей

SAHI (Slicing Aided Hyper Inference) – Гипер-вывод с поддержкой среза

SGD (Stochastic Gradient Descent) – Стохастический градиентный спуск

SOTA (State of the Art) – Последние достижения

SSD (Single Shot MultiBox Detector) – Одноэтапный детектор с множеством ограничивающих рамок

TN (True Negative) – Положительно отрицательный

TP (True Positive) – Истинно положительный

YOLO (You Only Look Once) – Стоит только раз взглянуть

Введение

Обнаружение объектов играет ключевую роль в задачах компьютерного зрения и обработки изображений, целью которых является обнаружение и определение местоположения интересующих объектов на изображении или видео. Задача включает определение положения и границ объектов на изображении, а также классификацию объектов по разным категориям. Нейронные сети, предназначенные для обнаружения объектов на изображениях, далее детекторы, уже сейчас активно применяются во многих сферах деятельности. Одним из наиболее популярных направлений применения детекторов стали БПЛА.

В настоящее время БПЛА становятся все более распространенным и востребованным средством для визуального наблюдения, а также контроля и мониторинга различных территорий. Точность обнаружения становится особенно важной при использовании БПЛА в военных или поисково-спасательных операциях. Однако, их использование в упомянутых задачах, связано с рядом проблем:

- малый размер обнаруживаемых объектов: искомые объекты могут занимать на изображении лишь несколько пикселей, что делает их почти не находимыми при использовании традиционных методов обработки изображений;
- сложный фон: изображения, снятые с БПЛА, как правило, содержат большое количество различных фоновых объектов, таких как деревья, дорожные знаки, здания и пр., что создаёт помехи при обнаружении целевых объектов;
- изменчивость условий съемки: погодные условия, угол съемки, освещенность, высота полета и другие факторы могут сильно повлиять на качество съемки и усложнить задачу поиска объекта.

Искусственные нейронные сети благодаря своей архитектуре являются перспективным подходом для решения задачи обнаружения небольших объектов на изображениях, снятых с дрона. Благодаря своей способности извлекать признаки и эффективно обрабатывать большие объемы данных они способны эффективно решить все вышеупомянутые проблемы поставленной задачи.

					ПГУ 1.090301.33.001 ПЗ	Лист
						7
Изм.	Лист	№ докум.	Подпись	Дата		

В рамках данной работы рассматривается применение искусственных нейронных сетей для обнаружения небольших объектов на изображения, полученных с БПЛА. Будут исследованы различные архитектуры нейронных сетей, а также методы предобработки данных. Будут проведена серия экспериментов с моделью наилучшей архитектуры, которая будет выбрана после сравнительного анализа всех подходящих решений, с целью добиться наилучших показателей в поставленной задаче.

					ПГУ 1.090301.33.001 ПЗ	Лист
						8
Изм.	Лист	№ докум.	Подпись	Дата		

1 Обзор предметной области

Обнаружение людей на кадрах, полученных с БПЛА, с применением нейронных сетей, объединяет в себе сразу несколько ключевых направлений в задачах компьютерного зрения, машинного обучения и обработки изображений. В данном разделе будет проведен обзор существующих методов, технологий и подходов к решению задачи, а также обсуждение актуальности и важности данной темы.

1.1 Анализ темы

В настоящее время дроны используются в различных отраслях жизни, кроме того, их популярность продолжает расти, а вместе с ней возрастает потребность в более точном и производительном алгоритме обнаружения объектов на изображениях, снятых с дронов. Такие методы имеют важное практическое значение и активно разрабатываются, и применяются в системах безопасности, оптимизации дорожного трафика, медицинских и научных исследованиях, контроле территорий и объектов, поисково-спасательных операциях и военной сфере.

1.1.1 Применение алгоритма в области безопасности

Алгоритмы обнаружения людей могут быть использованы для мониторинга общественных мест, таких как парки, площади, торговые центры, а также критически важных объектов, таких как аэропорты, железнодорожные станции, энергетические установки и промышленные комплексы. Дроны, оснащенные камерами и оборудованные алгоритмами обнаружения, могут обеспечивать непрерывное наблюдение за этими объектами, выявлять подозрительные действия и предотвращать возможные инциденты [1, 2].

Важным аспектом безопасности является контроль за границами и предотвращение нелегальной миграции. Алгоритмы обнаружения людей могут быть применены для мониторинга границ и обнаружения нелегальных пересечений границы. Это позволит правоохранительным органам оперативно реагировать на инциденты и предотвращать нелегальную деятельность [3, 4].

					ПГУ 1.090301.33.001 ПЗ	Лист
						9
Изм.	Лист	№ докум.	Подпись	Дата		

Дроны могут быть использованы для наблюдения за толпой во время проведения масштабных мероприятий (концертов, митингов, спортивных событий или других массовых мероприятий), выявления подозрительных предметов или поведения [3, 4].

1.1.2 Применение алгоритма в области оптимизации дорожного трафика

Алгоритмы обнаружения людей могут быть использованы для мониторинга дорожного движения и обнаружения нарушений правил безопасности, таких как переход через дорогу в неположенном месте или движение по дороге в неправильном направлении [5, 6].

Проведение непрерывного мониторинга дорог с помощью алгоритмов обнаружения людей позволит оперативно выявлять потенциально опасные ситуации и предотвращать аварии. Так, например, система могла бы сообщать о наличии препятствий на проезжей части и других факторов, которые могут привести к авариям [7, 8].

Алгоритм может быть использован для контроля скорости транспортных средств путем обнаружения людей на пешеходных переходах или сложных участках пути [9, 10].

1.1.3 Применение алгоритма в области медицинских и научных исследований

Алгоритмы обнаружения людей могут использоваться для анализа социального взаимодействия людей в различных ситуациях. Например, исследователи могут изучить групповое поведение в общественных местах, таких как парки, площади или торговые центры [11, 12].

Алгоритмы обнаружения могут отслеживать физическую активность людей. Например, исследователи могут изучить активность в парках или на спортивных площадках, чтобы проанализировать уровень физической активности населения и его влияние на здоровье [13, 14].

1.1.4 Применение алгоритма в области контроля территорий и объектов

Промышленные комплексы, заводы, энергетические установки и другие объекты инфраструктуры требуют постоянного контроля и обеспечения

безопасности. Беспилотные летательные аппараты могут использоваться для мониторинга и обнаружения присутствия людей на территории этих объектов. Алгоритмы обнаружения могут помочь выявить незаконные проникновения, несанкционированные действия и другие потенциально опасные ситуации с участием людей [10, 15].

БПЛА оснащенные алгоритмами обнаружения людей могут использоваться для контроля государственных, региональных или любых других границ [16, 17].

Для охраны природных ресурсов и заповедников можно использовать дроны с алгоритмами обнаружения людей. Такие БПЛА смогут отслеживать присутствие людей в запретных зонах, предотвращать браконьерство и незаконную добычу ресурсов, а также обеспечивать безопасность посетителей и персонала [10, 15].

1.1.5 Применение алгоритма в области поисково-спасательных операций

В поисково-спасательных операциях главной целью является быстрый поиск и локализация людей, которые оказались в опасности в результате стихийного бедствия или аварии. Такие ситуации регулярно возникают в труднопроходимых регионах, где поиск с земли неэффективен. БПЛА, оснащенные алгоритмами обнаружения людей, способны значительно сократить время необходимое на поиск пострадавших [18, 19].

В случае, если пострадавшие передвигаются, изменяя свое местоположение, алгоритмы обнаружения могут использоваться для непрерывного мониторинга и отслеживания их перемещений. Это позволит спасателям эффективнее координировать спасательную операцию.

1.1.6 Применение алгоритма в военной области

Военная область является основополагающей областью, в которой могут быть применены все вышеприведённые примеры.

Нейронные сети являются основой для многих современных алгоритмов компьютерного зрения. Это обусловлено тем, что алгоритмы, базирующиеся на нейронных сетях, способны извлекать признаки из изображений и обучаться на

больших объемах данных, что позволяет достичь высокой точности и обобщаемости.

Помимо ограничений, накладываемых самим БПЛА, как конечным устройством, существует также ряд проблем, обусловленных внешней средой. Среди них: сложные условия съемки (погодные условия, изменчивость освещения), разнообразие возможных углов съемки и поз искомого объекта, масштаб объектов, а также необходимость обработки большого объема данных в реальном времени.

1.2 Анализ решений

В данном разделе рассмотрены существующие решения и исследования по теме обнаружения людей на изображениях, снятых с беспилотных летательных аппаратов, с использованием нейронных сетей.

В исследовании будет рассматриваться использование только одноэтапных детекторов объектов, так как они являются наиболее подходящими для решения поставленной задачи [20].

Одноэтапные детекторы (one-stage detectors) – это искусственные нейронные сети для обнаружения объектов, которые выполняют задачу локализации и классификации объектов в один этап.

Как правило, современные архитектуры следуют следующим принципам:

- в качестве первого слоя выступает сверточная нейронная сеть для извлечения признаков из входного изображения;
- сеть предсказывает набор ограничивающих обнаружаемые объекты рамок (bounding boxes) и вероятности классов для каждой рамки;
- non-max suppression (NMS) – используется для устранения дублирующих ограничивающих рамок и определения наиболее верных из них.

Одним из важнейших компонентов одноэтапных детекторов является Feature Pyramid Network.

Feature Pyramid Network (FPN)

FPN – это архитектура сверточной нейронной сети (CNN), используемая для обнаружения объектов разных масштабов. FPN создает пирамиду карт объектов, где каждый уровень используется для обнаружения объектов разного размера. Более подробно рассмотрим работу архитектуры:

– bottom-up pathway – этап обработки изображений. Изображение проходит через последовательности сверточных и пулинговых (pooling) слоев, создавая карту признаков с разным уровнем детализации. Нижние уровни содержат данные о низкоуровневых признаках, а верхние – о высокоуровневых семантических признаках, таких как форма объекта.

– top-down pathway и lateral connections – второй этап, на котором создается пирамида признаков следующим образом:

1) происходит последовательное увеличение разрешения карт признаков, путем искусственного повышения частоты дискретизации;

2) на каждом уровне пирамиды, полученные ранее высокоуровневые признаки объединяются с соответствующими низкоуровневыми признаками. Таким образом, семантически богатые признаки верхних уровней дополняются общей информацией из нижних уровней, что особенно полезно при обнаружении небольших объектов;

– predictions – заключительный этап, на котором head слой, отвечающий за прогнозы нейронной сети, применяется на каждом уровне построенной ранее пирамиды. Благодаря этому, детектор использует информация с разных уровней детализации, что повышает точность обнаружения объектов разных размеров.

Минусом компонента можно считать то, что усложнение архитектуры нейронных сетей неминуемо приводит к необходимости увеличения времени обучения и точной настройки моделей.

Главным преимуществом одноэтапных детекторов над другими типами архитектур является высокая скорость работы, что позволяет использовать их в задачах реального времени. Алгоритм при этом жертвует точностью и становится крайне чувствителен к гиперпараметрам [20].

1.2.1 RetinaNet

RetinaNet – одноэтапный детектор объектов, который сочетает в себе две ключевые идеи: FeaturePyramidNetwork (FPN) для эффективного признаков разных масштабов и FocalLoss для решения проблемы дисбаланса классов. В качестве магистральной нейронной сети используется архитектура ResNet. На следующем этапе FPN создает пирамиду признаков с разными уровнями детализации. На каждом уровне пирамиды признаков применяются две подсети:

- подсеть, которая определяет вероятность принадлежности объекта к каждому классу;
- подсеть, которая более точно определяет местоположения ограничивающей рамки объекта.

На заключительном этапе работы, RetinaNet использует в качестве функции потерь FocalLoss. Эта функция значительно уменьшает вклад легко классифицируемых классов (фон) и фокусирует обучение модели на более сложных для обнаружения классах (небольшие объекты).

FocalLoss (FL) – это расширение функции потерь перекрестной энтропии (Cross-EntropyLoss или CE), которое предназначено для решения проблемы дисбаланса классов в моделях обнаружения объектов.

Функция потерь перекрестной энтропии – это метрика, позволяющая оценить модели классификации. Как и в большинстве случаев, значение метрики может находиться в диапазоне от 0 до 1, где 0 – идеальный показатель. Перекрестная энтропия измеряет среднее количество битов между двумя распределениями вероятностей. Другими словами, перекрестная энтропия измеряет разницу между обнаруженным распределением вероятностей модели и прогнозируемыми значениями [21, 22].

1.2.2 Single Shot MultiBox Detector (SSD)

SSD – архитектура искусственной нейронной сети, которая подразумевает под собой использование усеченной, предварительно обученной модели VGG-16. VGG-16, в данном случае, будет выступать в качестве магистрального компонента, целью которого является извлечение признаков из изображения. Из

магистральной сети отсекаются выходные классифицирующие слои, а на их место добавляются специальные сверточные слои, которые и представляют собой SSD. Каждый слой позволяет обрабатывать признаки разных масштабов [23].

1.2.3 MobileSSD

MobileSSD – это одноэтапный детектор объектов, оптимизированный для работы на мобильных устройствах. Он базируется на архитектуре SSD детектора, однако использует ряд модификация увеличивающих скорость и эффективность работы моделей на устройствах с ограниченными ресурсами.

Ключевые особенности MobileSSD:

- в качестве магистральной сети выступает легковесная сеть, например, MobileNetV2 или ShuffleNet. Это значительно снижает количество параметров и вычислительную сложность модели в сравнении с классическим вариантом SSD – VGG-16;

- в рассматриваемой архитектуре вместо обычных сверточных слоев используются depthwise separable convolutions слои. Идея этого типа слоев заключается в том, что стандартная операция свертки делится на свертку по каналам и 1x1 свертку, где порядок выполнения операций не влияет на конечный результат;

- архитектура следует идеи global average pooling, что означает полный отказ от полносвязных слоев. Такая сеть больше не требует на вход определенного размера изображения и состоит только из сверточных/пулинговых слоев необходимых для уменьшения карт признаков.

Модели архитектуры MobileSSD отличаются от моделей базовой SSD высокой скоростью работы, низким энергопотреблением и малым размером, жертвуя при этом точностью прогнозов [23].

1.2.4 CenterNet

CenterNet – архитектура, одноэтапного детектора, которая определяет центр объекта, не выделяя его ограничивающей рамкой.

В качестве магистральной сети используется стандартная сверточная нейронная сеть, такая как ResNet. Главным отличием CenterNet от ранее

рассмотренных детекторов является то, что он не разделяет изображения на сетку, после чего работает с каждой отдельной ячейкой на изображении. Модели данной архитектуры проверяют каждый пиксель на вероятность соответствия центру объекта в этой позиции. Одновременно с этим, CenterNet определяет смещения для каждого обнаруженного центра, которые корректируют позицию центра. Кроме того, модель прогнозирует размер объекта (ширину и высоту) для каждого обнаруженного центра.

Главным недостатком рассматриваемой архитектуры является то, что модели этой искусственной нейронной сети испытывают значительные трудности при работе с изображениями вытянутой формы, так как CenterNet предполагает, что все объекты имеют форму приближенную к квадратной [22].

1.2.5 Fully Convolutional One-Stage Object Detection или FCOS

FCOS – одноэтапный детектор, который следует идеям, заложенным в CenterNet. Архитектура полностью отказывается от работы с изображением как с ячейками сетки и при это полностью полагается на сверточные слои.

В качестве магистральной сети данная нейронная сеть использует ResNet или любую другую сверточную нейронную сеть для извлечения признаков из изображения. После чего FPN создает пирамиду признаков с разными уровнями детализации. На каждом уровне пирамиды для каждого предполагаемого центра объекта применяются следующие действия:

- осуществляется прогноз вероятности наличия объекта в данном местоположении;
- посредством регрессии вычисляются 4 значения, определяющие расстояние от данного местоположения до каждой стороны объекта, т.е. до границ его мнимой ограничивающей рамки;
- прогнозируется величина, которая определяет, насколько данное местоположение близко к центру объекта.

Главным преимуществом FCOS над CenterNet является то, что первый способен обрабатывать случаи с амодальной ограничивающей рамкой, т.е. случаи, в которых центр объекта может не быть центром ограничивающей рамки [23].

1.2.6 YOLO

Семейство архитектур YOLO (YouOnlyLookOnce) – на момент написания данной работы является одной из самых популярных разработок в области одноэтапных детекторов. Модели YOLO делят изображение на сетку и предсказывают ограничивающие рамки и вероятности классов в них для каждой ячейки сетки, что позволяет этим алгоритмам достигать скорости обработки изображений достаточной для того, чтобы их можно было использовать в системах, работающих в режиме реального времени. Точность одноэтапных детекторов, как правило, хуже, чем у двухэтапных детекторов, однако за счет ряда усовершенствований YOLOv8 способна показывать конкурентно способные результаты. Модели этой архитектуры также достаточно гибкие и простые для обучения и запуска, что позволяет интегрировать их в платформы БПЛА, на которых присутствуют определенные ограничения вычислительных ресурсов. Модель данной архитектуры смогла показать достойные результаты и оказалась достаточно эффективна для отслеживания людей на тепловизионных видео, снятых с БПЛА [24].

Исходя из вышесказанного по совокупности своих преимуществ: скорости работы, точности обнаружения, гибкости и простоты запуска модели на БПЛА было принято решение о проведении экспериментов с использованием модели архитектуры YOLOv8.

1.3 Анализ существующих наборов данных

Для обучения и оценки нейронных сетей, необходим качественный и репрезентативный набор данных. В этом подразделе проводится детальный анализ свободно распространяемых наборов данных, предназначенных для решения задачи обнаружения людей на изображениях, полученных с БПЛА.

Оцениваться наборы данных будут по следующим критериям: размер набора данных, разрешение изображений, количество изображений, количество сценариев (зависит от погодных условий, угла съемки и уровня освещения; уровни градации разнообразия условий: мало – до 3, нормально – до 7, много – 7

и больше) и качество разметки (плохое – необходимо переделать разметку полностью или почти полностью, среднее – необходимо незначительно дополнить или исправить ограничивающие рамки, хорошее – в наборе данных нет необходимости дополнять или редактировать разметку). В качестве инструмента для проверки аннотаций будет использована программа CVAT.

Таблица 1.1 – Анализ наборов данных

Название набора данных	NTUT 4K Drone Photo Dataset for Human Detection [25]	DroneCrowd [26]	Aerial Person Detection Object Detection Dataset [27]	MOBDrone [28]	Detect White Clothed People With Drones [29]
Размер набора данных	30 ГБ	10 ГБ	1.5 ГБ	243 ГБ	52 ГБ
Разрешение изображений	3840×2160	1920×1080	1360×765 960×540	1920×1080	960×540
Количество изображений	4100	33600	7015	126170	~167000
Количество сценариев	Мало	Много	Мало	Нормальн о	Много
Качество разметки	Среднее	Хорошее	Среднее	Хорошее	Плохое

Из приведённой выше таблицы становится понятно, что использовать готовые наборы данных, которые подошли бы к текущей задаче и при этом никак их не дополнить или отредактировать не представляется возможным, что является еще одной проблемой при решении данной задачи.

Существующие исследования демонстрируют эффективность нейронных сетей для обнаружения людей на изображениях с БПЛА. Однако, остаются актуальными проблемы, связанные с малыми размерами обнаружаемых объектов,

сложным фоном и широким списком возможных условий съемки. Для решения этих проблем необходимы дальнейшие исследования в области разработки нейронных сетей и методов аугментации данных [16, 26, 30].

1.4 Постановка задачи

В рамках настоящей работы необходимо исследовать алгоритм поиска объектов на изображения, снятых с БПЛА, основанный на искусственной нейронной сети архитектуры.

В процессе исследования необходимо:

- изучить особенности архитектуры YOLOv8;
- обучить модель семейства YOLOv8 на наборе данных, который будет состоять из изображений, полученных с БПЛА;
- оценить производительность модели. В качестве метрик для оценки использовать Precision, Recall, mAP50, mAP50-95;
- изучить методы оптимизации модели для повышения ее точности и эффективности в поставленной задаче.

Необходимо учитывать, что БПЛА имеют ограничения, связанные с аппаратной платформой. Большинство устройств, применяемых в беспилотных летательных аппаратах, разработаны специально под задачи, связанные с ними (управление полетом, навигация, обработки изображений). При этом, устройства ограничиваются размером и весом летательного аппарата.

Рассматриваемая модель искусственной нейронной сети должна запускаться на устройстве, которое соответствует следующим требованиям: вес устройства $\leq 0,2$ Кг, энергопотребление $\leq 10W$, FLOPS (количество операций с плавающей запятой в секунду) ≥ 472 GFLOPs (FP16). Модель должна быть способна обрабатывать изображения на данном устройстве со скоростью не менее 40 кадров в секунду (FPS).

Данное исследование направлено на изучение применения и оптимизацию модели архитектуры YOLOv8 для решения задачи обнаружения объектов на изображениях, снятых с БПЛА. Результаты исследования могут быть

использованы для разработки более эффективных систем управления БПЛА и расширения сферы их применения.

					ПГУ 1.090301.33.001 ПЗ	Лист
						20
Изм.	Лист	№ докум.	Подпись	Дата		

2 Выбор технологий разработки

В разделе будут рассмотрены технологии и среды разработки, которые были выбраны для реализации алгоритма нахождения объектов на изображениях, полученных с БПЛА.

2.1 Выбор программной библиотеки

2.1.1 TensorFlow

TensorFlow – платформа машинного обучения, разработанная компанией Google, чтобы решить задачу построения и обучения нейронных сетей, которые автоматически ищут и классифицируют изображения для достижения качества человеческого восприятия. Фреймворк включает в себя инструменты для работы с данными, их загрузке и предобработке, а также анализа и валидации данных [31].

2.1.2 Keras

Keras – фреймворк, который предоставляет высокоуровневый, интуитивно понятный набор инструкций, который делает простым формирование нейронных сетей, независимо от используемой в качестве вычислительного бэкенда библиотеки научных вычислений. Фреймворк также предоставляет некоторые инструменты для работы с данными, встроенные функции загрузки данных и их предобработки. На данный момент Keras поддерживает только TensorFlow [32].

2.1.3 PyTorch

PyTorch – открытая библиотека программного обеспечения для машинного обучения, созданная на основе Torch, с открытым исходным кодом. Подходит для выполнения задач компьютерного зрения, обработки естественного языка и многих других. Наибольший вклад в разработку библиотеки внесла AI команда Facebook. PyTorch предоставляет широкий набор инструментов для работы с данными, загрузки и предобработки изображений, для работы с текстовыми данными и аудиоданными. Фреймворк отличается гибкостью и простотой использования, а также активным сообществом, которое поддерживает работоспособность даже кода с высоким уровнем технического долга. Кроме того,

PyTorch предоставляет большой выбор предобученных моделей нейронных сетей и библиотек, таких как TorchVision, что значительно ускоряет разработку и процесс обучения [33].

Исходя из вышесказанного по совокупности своих преимуществ: простоты, гибкости и активной поддержки со стороны сообщества, основным фреймворком был выбран PyTorch.

2.2 Выбор IDE

2.2.1 Visual Studio Code

Visual Studio Code (VS Code) – редактор исходного кода, разработанный Microsoft для Windows, Linux и macOS. Позиционируется как «легкий» редактор кода для кроссплатформенной разработки веб и облачных приложений. Большое количество плагинов и расширений позволяет настроить среду разработки под текущие нужды [34].

2.2.2 Jupyter Notebook

Jupyter – интерактивный блокнот, первоначально являвшийся веб-приложением и развитием IPython, превратился в отдельный проект, ориентированный на работу с Python и несколькими средами выполнения, включая R, Julia и Scala. Фреймворк позволяет исполнять код в специальных ячейках, сохраняя полученный результат выполнения кода прямо в документе, что значительно упрощает проведение экспериментов и исследовательских работ. Jupyter может использоваться как плагин или расширение практически во всех современных IDE [35].

2.2.3 PyCharm

PyCharm – это кроссплатформенная IDE для языка программирования Python, разработанная JetBrains на основе IntelliJ IDEA. Предоставляет набор инструментов, позволяющий пользователям писать код, и визуальный отладчик. Среда разработки оснащена набором инструментов для удобной и быстрой разработки на языке Python. IDE поддерживает множество функций, таких как

отладка, автодополнение кода, интеграция с системами контроля версий и др. [36].

В качестве среды разработки была выбрана IDE от JetBrains – PyCharm, так как она предоставляет весь необходимый функционал для комфортной разработки Python-проектов.

2.3 Выбор инструмента аннотирования данных

Аннотирование изображений – это трудоемкий процесс, который напрямую влияет на качество обучаемых моделей. В связи с тем, что наборы данных, находящиеся в открытом доступе, не отвечают всем необходимым требованиям поставленной задачи, потребуется дополнительная разметка имеющихся данных. Выбор правильного инструмента для аннотирования может значительно ускорить и упростить этот процесс.

Требования к инструменту:

- инструмент должен поддерживать возможность аннотирования посредством ограничивающих рамок;
- интерфейс должен быть интуитивно понятен, с возможностью настройки горячих клавиш и быстрой навигации по изображениям;
- инструмент должен поддерживать форматы изображений jpg и png, а также формат аннотации YOLOv8.

2.3.1 LabelStudio

LabelStudio – это платформа с открытым исходным кодом для аннотирования изображений. LabelStudio предлагает гибкий настраиваемый интерфейс, возможность создания пользовательских шаблонов аннотирования и интеграцию с различными инструментами машинного обучения [37].

2.3.2 CVAT

CVAT (ComputerVisionAnnotationTool) – это современный веб-инструмент с открытым исходным кодом. CVAT поддерживает разметку изображений ограничивающими рамками, а также содержит все популярные типы аннотаций в том числе и аннотации в формате YOLOv8 [38].

Исходя из вышеперечисленного можно сделать вывод, что оба инструмента подойдут для решения поставленной задачи. В данной работе будет использоваться CVAT в связи с наличием опыта работы с ним.

2.4 Выбор инструмента версионирования данных

Версионирование данных – это процесс отслеживания и управления изменениями в данных с течением времени. Он позволяет сохранять различные версии набора данных, отслеживать изменения, восстанавливать предыдущие версии и сравнивать их, подобно тому, как системы контроля версий, такие как Git, взаимодействуют с кодом.

Версионирование данных имеет ряд полезных преимуществ, которые упрощают проведение экспериментов:

- воспроизводимость. Позволяет вернуться к предыдущим версиям данных, что полезно, для повторения экспериментов;
- откат изменений. Предоставляет возможность отменить нежелательные изменения и восстановить предыдущие состояние данных;
- обеспечивает возможность отслеживания изменений.

Существует множество инструментов для версионирования данных. Разберем наиболее известные из них и выберем наиболее подходящий для текущей задачи.

2.4.1 Amazon S3

Amazon Simple Storage Service (AmazonS3) – онлайн-веб-сервис, предоставляемый Amazon Web Services, который позволяет хранить и извлекать данные в любое время и в любом объеме из любой точки сети, называется файловым хостингом.

Сервис является одним из наиболее популярных и широко используемых хранилищ данных в облаке, обеспечивая высокую доступность, надежность и масштабируемость. Кроме того, AmazonS3 AWS легко интегрируется с другими сервисами AWS, получая ещё больше возможностей для взаимодействия, анализа, обработки и управления данными [39].

2.4.2 Pachyderm

Pachyderm – это платформа с открытым исходным кодом, предназначенная для управления данными, версионирования данных и создания оркестров конвейеров машинного обучения. Инструмент является легко масштабируемым, кроме того, он содержит большое количество полезных функций как для работы с данными, так и для работы с оркестрами конвейеров. Pachyderm – мощный инструмент, настройка и управление которым требует много времени, что также означает высокие требования к вычислительным ресурсам [40].

2.4.3 Delta Lake

DeltaLake – проект с открытым исходным кодом на базе ApacheSpark, который предоставляет возможность использовать ACID-транзакции и управлять версиями данных. DeltaLake обеспечивает атомарность, согласованность, изоляцию и долговечность ACID-транзакция, что гарантирует согласованность данных даже при одновременном доступе множества пользователей и приложений. Это устраняет проблемы, связанные с несогласованностью данных.

Главным недостатком этого инструмента является ограниченная гибкость версионирования данных, не свойственная Git-подобным системам. Проект автоматически отслеживает изменения данных и сохраняет их как серию версий. DeltaLake может выступать в качестве дополнительного ПО, расширяющего возможности других проектов, например LakeFS [41].

2.4.4 LakeFS

LakeFS - это инструмент с открытым исходным кодом, который преобразует объектное S3 хранилище в репозиторий, подобный Git. Это позволяет управлять данными так же, как кодом. LakeFS обладает рядом достоинств, при работе с большими объемами данных:

- масштабируемость. Инструмент предназначен для работы с огромными объемами данных и легко масштабируется в случае такой необходимости;
- поддержка различных типов данных. LakeFS поддерживает работу с различными типами данных, как структурированных, так и не

структурированных, что делает его пригодным для использования в любых задачах;

– версионирование данных. LakeFS оснащен средствами версионирования данных, что позволяет эффективно проводить анализ и обработку данных, сохранять историю изменения данных и возвращаться к предыдущим итерациям наборов данных [42].

В данном исследовании будет использоваться инструмент LakeFS, так как он предоставляет достаточно гибкие и функциональные инструменты для версионирования данных и является достаточно простым в освоении.

В качестве S3-совместного хранилища для LakeFS был выбран Minio. Использование облачных сервисов для хранения данных из LakeFS, таких как AmazonS3, предполагает аренду или покупку дисковых пространств на удаленных серверах, что в данном исследовании не целесообразно. Minio – проект с открытым исходным кодом, который предоставляет большую гибкость в настройке. Главным преимуществом, из-за которого было принято решение использовать его в исследовании является его быстрая и удобная развертка в локальной среде. В качестве преимуществ Minio так же можно выделить его масштабируемость, посредством добавления новых узлов, увеличивающих емкость и производительность инструмента [43].

2.5 Выбор инструмента версионирования моделей

Версионирование моделей нейронных сетей – один из важнейших аспектов при разработке, развертывании и управлении моделями машинного обучения. Возможность отслеживать изменения в обученных моделях, проводить эксперименты с различными версиями и воспроизводить полученные ранее результаты является ключевым аспектом любого научного исследования.

2.5.1 Weights & Biases

Weights&Biases (WandB) – облачная платформа для управления экспериментами машинного обучения. Она предоставляет инструменты для

					ПГУ 1.090301.33.001 ПЗ	Лист
						26
Изм.	Лист	№ докум.	Подпись	Дата		

оптимизации, управления и отслеживания экспериментов, а также визуализации и анализа их результатов.

Платформа во многом направлена на обеспечение более эффективного сотрудничества группы людей. Инструмент поддерживает различные языки программирования, включая Python, R и JavaScript, а также многие популярные фреймворки языка Python, например TensorFlow, PyTorch, Keras и др.

Для отслеживания экспериментов WandB был оснащен рядом полезных разработчику функций: логирование гиперпараметров, метрик, кода, системных показателей в реальном времени; сравнительный анализ запусков модели и влияния гиперпараметров на метрики и производительность модели; создание отчетов и презентаций с возможностью визуализации полученных результатов обучения [44].

2.5.2 Git

Git – система для работы с кодом, т.е. для работы с текстовыми файлами. В связи с этим, утилита эффективно показывает себя при слиянии тестовых файлов, однако она не способна проводить слияния бинарных файлов моделей.

Работа с ним как с инструментом версионирования данных затрудняется и рядом других ограничений:

- модели искусственных нейронных сетей представляют собой бинарные файлы большого размера. Большой размер репозитория неминуемо повлечет за собой замедление операций в нем, вызвав проблемы с производительностью;
- git не способен отслеживать взаимосвязь данных, кода и модели.

Для устранения упомянутых недостатков было разработано несколько решений:

- различные расширения для Git, такие как GitLFS, которые позволяют хранить большие файлы во внешнем хранилище, а в репозиториях сохранять только файлы с ссылками на них;
- инструменты, которые интегрируются с Git и предоставляют весь необходимый функционал для версионирования данных, например, DVC.

Git можно использовать для версионирования моделей машинного обучения, однако использование специализированных инструментов является предпочтительным вариантом. Далее рассмотрим использование расширения GitLFS и платформу DVC[45].

2.5.3 Git LFS

GitLFS (Large File Storage) – расширение Git, которое позволяет эффективно работать с большими файлами в Git-репозиториях. Расширение автоматически помещает файлы больших размеров во внешнее хранилище, оставляя в репозитории только текстовый указатель (pointer) на изначальный файл. В качестве внешнего хранилища надстройка может использовать различные S3 хранилища, такие как AmazonS3, Google Cloud Storage и др [46].

2.5.4 DVC

DVC (DataVersionControl) – инструмент, расширяющий возможности Git для работы с наборами данных и моделями машинного обучения. Платформа упрощает взаимодействие с версиями данных и моделей, а также обеспечивает воспроизводимость экспериментов.

DVC отслеживает изменения в данных подобно тому, как Git отслеживает изменения в коде. Проект позволяет связывать данные с кодом и моделями нейронных сетей, создавая snapshot проекта. Таким образом, обеспечивается удобная и быстрая воспроизводимость экспериментов, без необходимости возвращения к коду, данным и модели определенных версий по-отдельности.

Платформа содержит инструментарий для создания конвейеров автоматизации задач машинного обучения, а также позволяет отслеживать метрики моделей в процессе обучения и сравнивать их различные версии [47].

2.5.5 MLflow

MLflow – это открытая платформа, которая позволяет создавать конвейеры машинного обучения и управлять всеми этапами жизненного цикла моделей искусственных нейронных сетей.

MLflow состоит из 4 модулей:

– tracking. Отвечает за логирование гиперпараметров, метрик, кода и артефактов, в качестве которых обычно выступают графики и изображения, во время экспериментов;

– projects. Создает воспроизводимые проекты машинного обучения, которые можно запускать в различных средах;

– models. Упаковывает модели искусственных нейронных сетей в стандартный формат, который может быть использован различными инструментами развертывания;

– registry. Предоставляет централизованное хранилище для управления этапами жизненного цикла моделей машинного обучения.

MLflow совместим с различными языками программирования (Python, R, Java) и различными фреймворками машинного обучения (scikit-learn, TensorFlow, PyTorch) [48].

DVC действительно является удобным и функциональным проектом для разработки и исследования архитектур искусственных нейронных сетей. Однако, в данном исследовании в качестве инструмента версионирования данных был выбран MLflow, так как фреймворк Ultralytics, на котором реализована архитектура рассматриваемого в исследовании семейства моделей YOLOv8, легко интегрируется с MLflow и автоматически сохраняет все необходимые данные каждого эксперимента.

2.6 Выбор инструмента управления виртуальным окружением

2.6.1 Venv

Venv – это базовый инструмент в Python, который позволяет создавать изолированное окружение Python-проектов без необходимости установки дополнительных пакетов. Стандартизированный формат виртуальных окружений делает их переносимыми между различными системами. Ограниченный функционал в сравнении с более продвинутыми аналогами, а также отсутствие автоматического управления зависимостями приводит к тому, что использовать

venv стоит лишь при работе с небольшими проектами, без специфических требований [49].

2.6.2 Pipenv

Pipenv – менеджер зависимостей для Python-проектов. Главной особенностью инструмента является то, что помимо создания виртуального окружения, он также способен автоматически управлять зависимостями при создании этого окружения. Одним из основных достоинств менеджера можно считать его интеграцию с pip, что позволяет ему работать с любыми существующими PyPI-пакетами [50].

2.6.3 Conda

Conda позволяет создавать и управлять изолированными окружениями и пакетами в них. Использование Conda означает, что для работы будут использоваться версии Python Anaconda Repository, а интеграция с Anaconda Distribution обеспечит доступ к большому количеству пакетов для научных вычислений и анализа данных. Без доступа к экосистеме Anaconda данный инструмент становится значительно менее гибким, функциональным и удобным [51].

Из всех вышеперечисленных вариантов наиболее подходящим для решения поставленной задачи следует считать Conda, так как инструмент идеально подходит для проведения научных исследований и анализа данных.

3 Набор данных

Для обучения и оценки нейронных сетей, применяемых для обнаружения людей на изображениях, снятых с БПЛА, необходим качественный и репрезентативный набор данных. В данном разделе рассматриваются критерии, которым должен соответствовать итоговый набор данных, этапы его подготовки, а также методы и инструменты, которые будут при этом использованы.

В связи с отсутствием подходящих для решения задачи наборов данных, находящихся в открытом доступе, был собран датасет, объединяющий в себе два набора, находящихся в открытом доступе – NTUT 4K Drone Photo Dataset for Human Detection и DroneCrowd, а также еще один набор данных, представленный на технологическом конкурсе НТИ Up Greate «Экстренный поиск» – Сателлит №1.

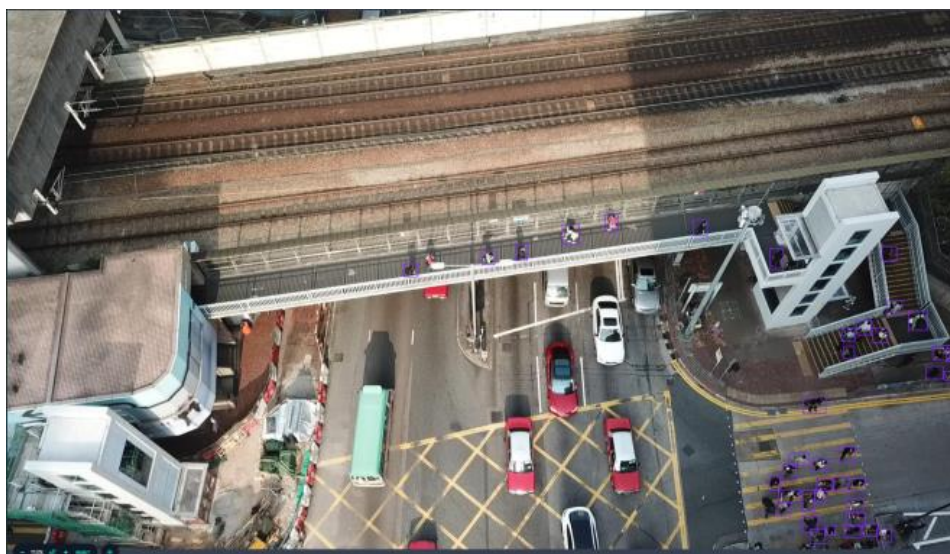
DroneCrowd и NTUT 4K в сумме содержат 38063 изображения. Изображения представляют собой снимки сделанные с БПЛА в условиях городской среды. Съемка производилась в разнообразных частях нескольких городов, с различных углов, на разной высоте полета дрона и в разное время суток.

Большая часть изображений являются частью DroneCrowd. Однако, эти изображения представляют собой кадры видео, последовательно идущие друг за другом. Так как многие изображения являются практически полностью идентичными, в итоговый набор данных будет добавлено только 300 изображений из разных частей видео. Таким образом, из 38063 изображений останется 4403.

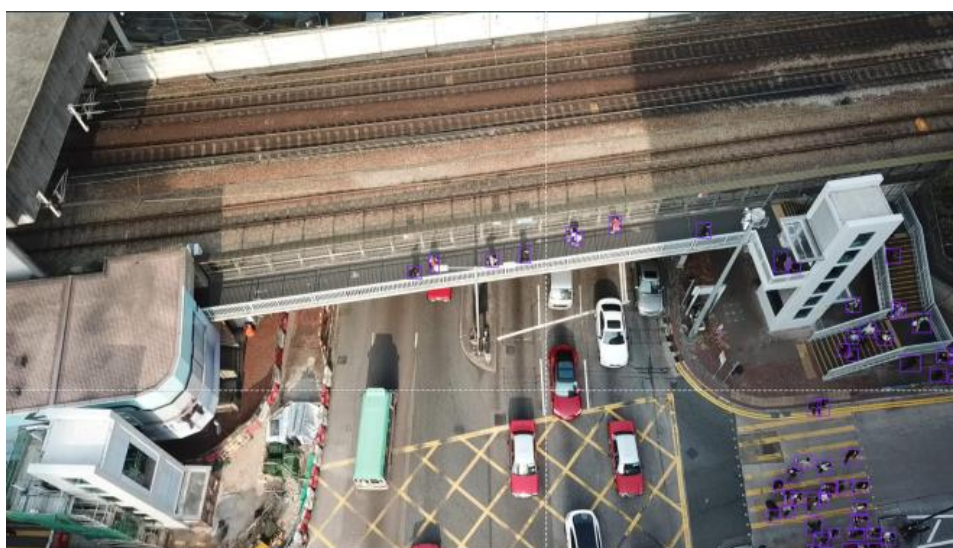
Конкурсный набор данных состоит из 8494 изображений, снятых с БПЛА в пересеченной местности, покрытой снегом. Набор данных практически не нуждается в дополнительных операциях, связанных с предварительной обработкой изображений и улучшением качества разметки.

3.1 Предварительная обработка данных

Разметка наборов данных, полученных из открытых источников, была неподходящей для решения поставленной задачи. В данных присутствовали не размеченные объекты, дубликаты изображений, ограничивающие рамки, покрывающие области без объектов. Для визуального анализа разметки, а также ее корректировки, использовался CVAT. Изображения с начальной разметкой и итоговой представлены на рисунках 3.1, 3.2, 3.3 и 3.4.



а)



б)

Рисунок 3.1 – Стандартная (а) и итоговая (б) разметка изображения

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.33.001 ПЗ

Лист

32



а)



б)

Рисунок 3.2 – Стандартная (а) и итоговая (б) разметка изображения



а)



б)

Рисунок 3.3 – Стандартная (а) и итоговая (б) разметка изображения



а)



б)

Рисунок 3.4 – Стандартная (а) и итоговая (б) разметка изображения

После того, как данные были повторно размечены и очищены от дубликатов и артефактов необходимо привести разметку к единому формату YOLOv8. CVAT

позволяет загружать и выгружать файлы разметки в различных форматах, таким образом все аннотации были сохранены в формате YOLOv8.

Как можно заметить, часть изображений содержит разноцветные ограничивающие рамки, что означает, что эти объекты принадлежат разным классам. Во всех наборах данных не размечены никакие объекты кроме людей, многообразие классов представлено многообразием деятельности людей, например «идет», «стоит», «сидит». Для того, чтобы привести все объекты к одному классу заменим в файле аннотации первое значение каждой строки на цифру «0», которая будет представлять класс «человек»:

```
def change_class_to_zero(annotation_dir):
    for filename in os.listdir(annotation_dir):
        if filename.endswith(".txt"):
            filepath = os.path.join(annotation_dir, filename)
            with open(filepath, "r") as f:
                lines = f.readlines()
            with open(filepath, "w") as f:
                for line in lines:
                    parts = line.strip().split(" ")
            parts[0] = "0" # Заменяем класс на 0
            f.write(" ".join(parts) + "\n")
```

При обучении параметр, отвечающий за размер принимаемого на вход изображения, будет равен 640 пикселям. Это необходимо для того, чтобы модель лучших результатов при обучении, извлекая признаки из изображений того же размера, который будет использоваться при запуске модели на БПЛА устройстве. Фреймворк Ultralytics имеет встроенную предварительную обработку изображений, которая включает в себя приведение размера входного изображения к установленному значению. Если размер изображения меньше заданного, то по краям изображения будут добавлены черные полосы, если же изображение будет больше, то оно будет сжато до нужного значения. При работе с данными, полученными с дрона, еще большее уменьшение размера объектов на них приведет к тому, что найти их будет невозможно. В связи с этим, один из этапов предварительной обработки данных включает нарезку изображений на фрагменты, в которых ширина полученного фрагмента будет равняться 640

					ПГУ 1.090301.33.001 ПЗ	Лист
						36
Изм.	Лист	№ докум.	Подпись	Дата		

пикселей, а высота будет вычисляться исходя из соотношения сторон изначального изображения.

Получаем список файлов изображений в указанной директории:

```
image_files = [f for f in os.listdir(input_folder) if
f.endswith(('.jpg', 'jpeg', 'png', 'bmp'))]
```

Циклом перебираем каждый файл изображения и аннотации:

```
for image_file in image_files:
    input_image_path = os.path.join(input_folder, image_file)
    input_label_path = os.path.join(input_folder,
os.path.splitext(image_file)[0] + ".txt")
```

В каждом файле изображения вычисляем количество столбцов и строк, которыми будет поделено изображение. Для этого используем заданную ширину изображения равную 640, так как набор данных содержит изображения, на которых ширина всегда больше высоты:

```
with Image.open(input_image_path) as img:
    width, height = img.size
    target_height = int(height * (640 / width))
    cols = width // 640
    rows = height // target_height
```

Вычисляем координаты фрагмента изображения, после чего вырезаем его и сохраняем в отдельную переменную для дальнейшей работы с ним:

```
left = col * target_width
top = row * target_height
right = left + target_width
bottom = top + target_height
cropped_img = img.crop((left, top, right, bottom))
```

Далее, необходимо проверить, находятся ли в полученном фрагменте объекты:

```
class_id = parts[0]
x_center = float(parts[1])
y_center = float(parts[2])
box_width = float(parts[3])
box_height = float(parts[4])
```

```
if x_center * width >= left and x_center * width <= right and
y_center * height >= top and y_center * height <= bottom:
```

```
new_x_center = (x_center * width - left) / 640
new_y_center = (y_center * height - top) / target_height
new_box_width = box_width * width / target_width
new_box_height = box_height * height / target_height
```

Фрагмент изображения сохраняется, как и относящийся к нему файл аннотации:

```
output_image_path = os.path.join(output_folder,
f"{os.path.splitext(image_file)[0]}_{row}_{col}.jpg")
new_label_file.write(f"{class_id} {new_x_center} {new_y_center}
{new_box_width} {new_box_height}\n")
```

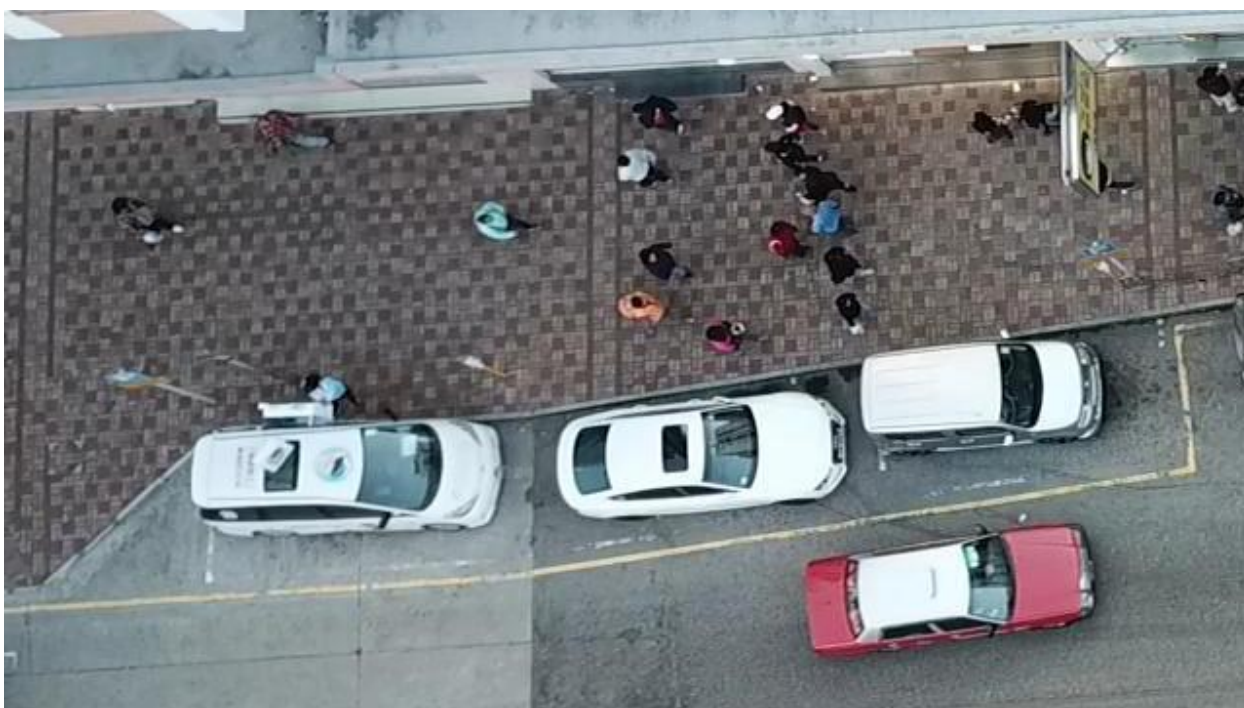
Примеры полученных изображений представлены на рисунках 3.5 и 3.6



Рисунок 3.5 – Фрагмент изображения, полученный из конкурсного набора данных



а)



б)

Рисунок 3.6 – Фрагменты изображений, полученные из NTUT 4К (а) и DroneCrowd (б)

Часть фрагментов, не содержавших объекты, была удалена. Однако, 100 фрагментов были сохранены и использовались для улучшения качества модели искусственной нейронной сети как фоновые изображения.

В указанной директории создается список всех файлов, после чего файл аннотации проверяется на отсутствие в нем содержимого. Если он пустой, то файл разметки и связанное с ним изображение удаляется:

```
files = os.listdir(folder_path)

for file_name in files:
    file_path = os.path.join(folder_path, file_name)

    if file_name.endswith('.txt') and
os.path.getsize(file_path) == 0:
        os.remove(file_path)
        os.remove(os.path.join(folder_path, file_name.replace('.txt',
'.jpg')))
```

Для улучшения качества полученного набора данных также необходимо использовать аугментацию данных. Аугментация данных – процесс изменения изображений, добавления шума на них. Она необходима для того, чтобы модель искусственной нейронной сети лучше научилась извлекать важные признаки из обучающей выборки, что в свою очередь увеличит количество верных прогнозов и уменьшит количество неверных, связанных с фоном на изображениях.

3.2 Описание набора данных

Итоговый набор данных состоит из 8037 изображений, поделенных на 2 выборки: обучающую, которая используется при обучении модели и проверочную, данные которой никогда не используются при обучении модели. Проверочная выборка представляет собой ~20% от общего количества изображений или 1600 изображений и используется только при подсчете метрик. Изображения между выборками были распределены случайным образом, кроме 100 фоновых изображений, которые были определены в обучающую выборку. Для хранения полученных выборок использовалась структура набора данных, представленная на рисунке 3.7.

					ПГУ 1.090301.33.001 ПЗ	Лист
						40
Изм.	Лист	№ докум.	Подпись	Дата		

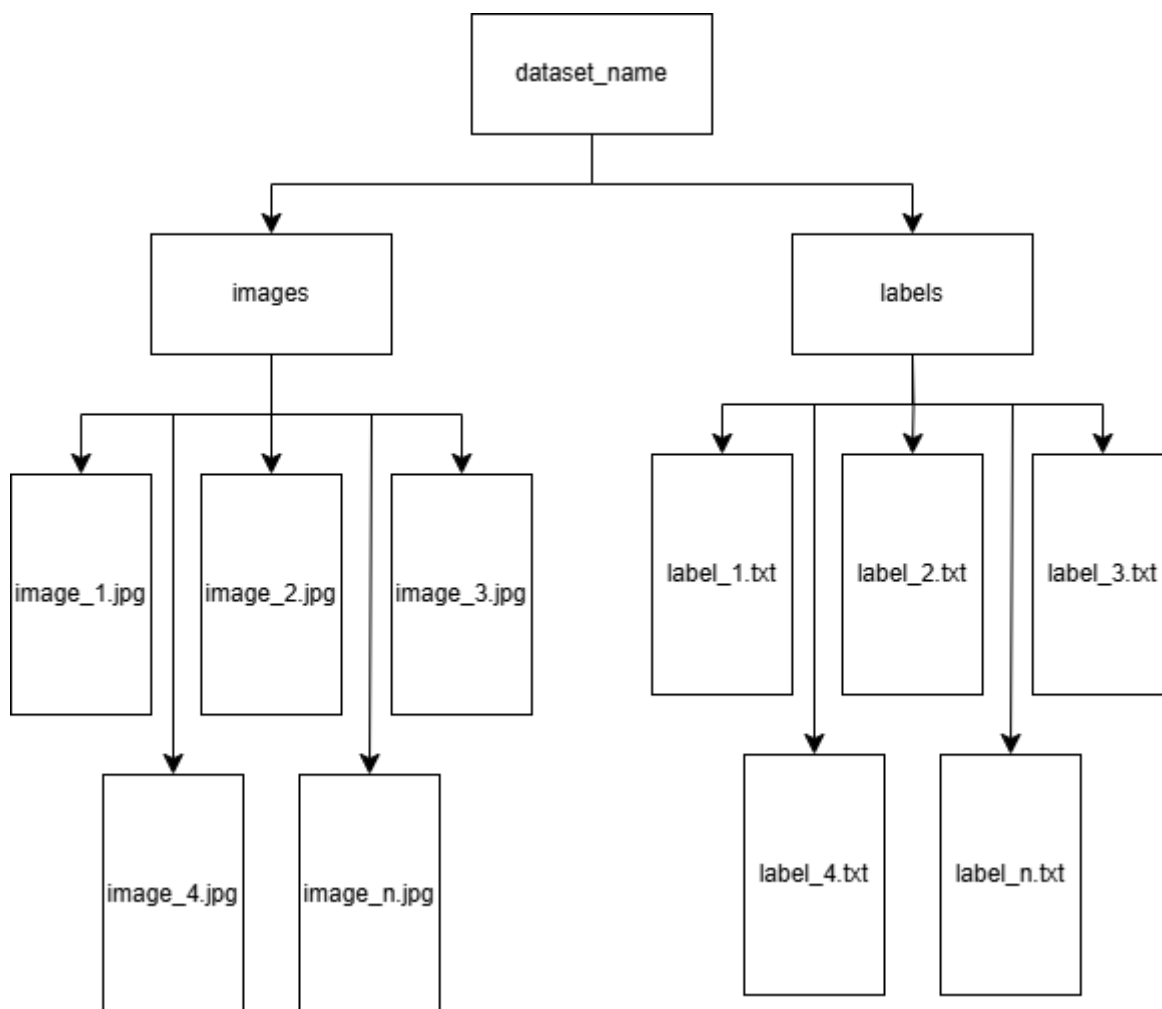


Рисунок 3.7 – Структура хранения набора данных

Выбранная структура хранения данных поддерживается фреймворком Ultralytics.

Полученный набор данных содержит 24029 объектов, представляющих класс «человек». Обучающая выборка содержит 22213 объектов, проверочная – 1816. Около 55% искомых объектов находятся в движении, в момент съемки, 37% стоят на месте, оставшиеся объекты представляют собой людей, которые лежат на земле или лавочках, сидят на лавочках, стульях и т.д. или передвигаются в транспортных средствах. В обучающей и проверочной выборках эти показатели приблизительно равняются 50%, 40% и 10%, а также 10%, 60%, 30% соответственно. Конкурсные изображения, полученные зимой, представляют 70% проверочной выборки и 20% обучающей.

4 Программная реализация

В данной главе рассматривается архитектура YOLOv8, а также основные функции модели и их параметры. Представлено описание разработанного программного модуля для обнаружения людей на изображениях, полученных с беспилотных летательных аппаратов.

4.1 Описание архитектуры YOLOv8

YOLOv8 – SOTA архитектура модели искусственной нейронной сети групп семейства архитектур YOLO. YOLOv8 спроектирована быть быстрой и точной, модель также оснащена новыми функциями, которые не были реализованы в предыдущих версиях YOLO. Повышенные возможности YOLOv8 сделали ее более гибкой и простой в использовании по сравнению с более ранними версиями. Архитектура семейства моделей представлена на рисунке 4.1.

CSP – это архитектурный блок, который широко используется в версиях YOLO, начиная с 3. Архитектуры с CSP разделяют входные данные на 2 пути: один путь – стандартные сверточные блоки, второй путь – часть слоев подключается посредством skip-connection к более поздним слоям, минуя промежуточные слои. Таким образом, информация из ранних слоев напрямую попадает в более поздние слои, что делает модель сети более устойчивой к градиентным исчезновениям, увеличивая точность модели, за счет более эффективного использования информации из ранних слоев, и уменьшает потребление вычислительных ресурсов для достижения тех же результатов, каких способны достичь другие искусственные нейронные сети при большем потреблении ресурсов.

PAN – это архитектурный блок, цель которого схожа с FPN. Через использование skip-connections и агрегации функций позволяет модели эффективно обрабатывать объекты на картах с разным масштабом, что значительно улучшает точность модели, особенно при работе с небольшими объектами.

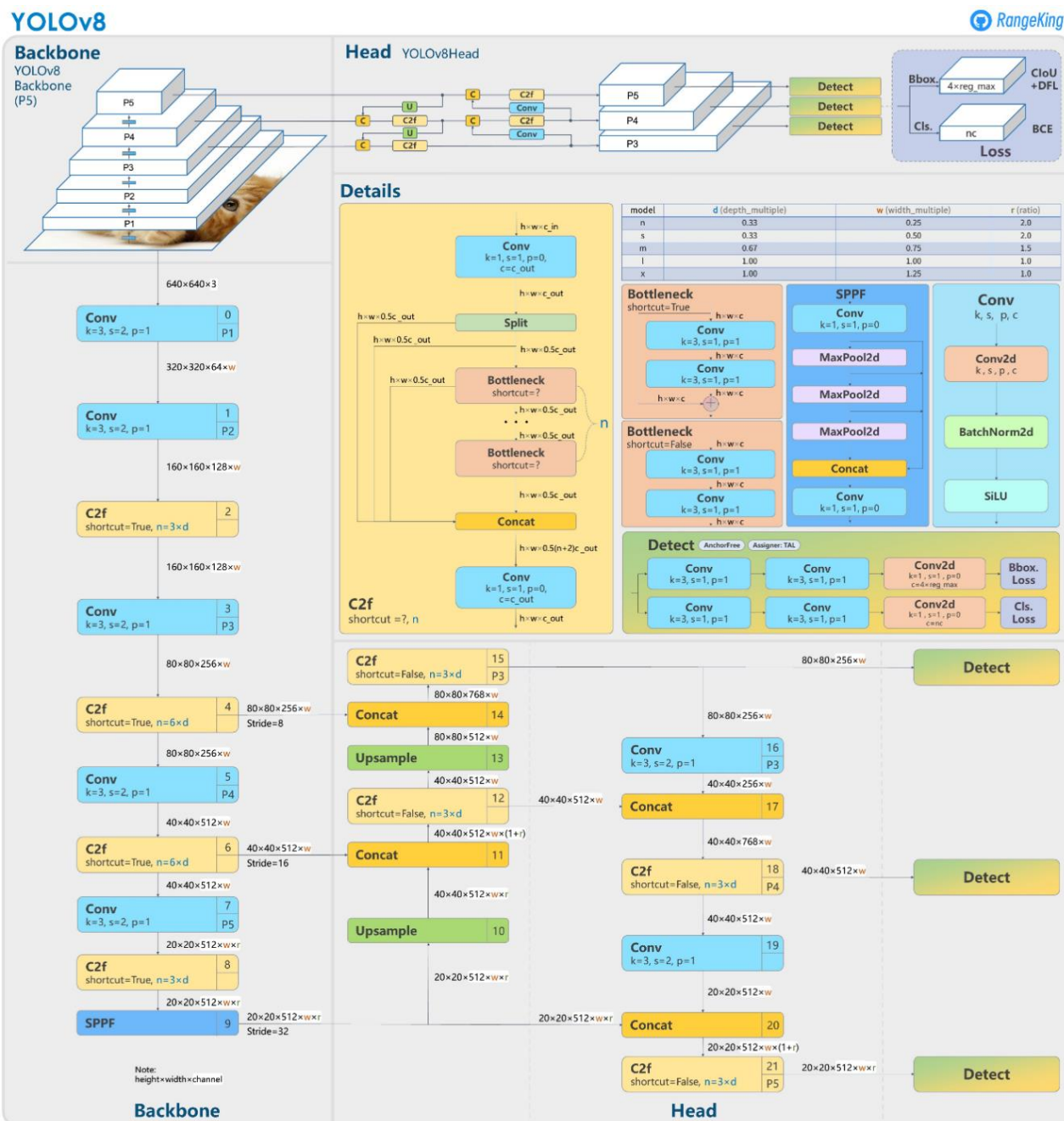


Рисунок 4.1 – Архитектура YOLOv8

Архитектура YOLOv8 состоит из 3 основных компонентов:

- backbone (магистральная сеть). CSPDarkNet53 используется в качестве магистральной сети. Модели, основанные на архитектуре CSP (Cross-Stage Partial Connections), способны эффективнее извлекать признаки из изображений, поступающих на вход, затрачивая меньше вычислительных ресурсов.
- neck. Является связующим звеном между магистральной сетью, которая извлекает признаки из входных данных, и прогнозирующими слоями. Основными

особенностями Neck в YOLOv8 можно назвать использование PAN (Path Aggregation Network), CSP и их различных комбинаций для достижения оптимальной точности модели.

– head. Компонент, который на вход получает данные, переданные компонентом Neck. Отвечающий за прогнозирование результатов обнаружения объектов. Одной из главных особенностей компонента является его многоуровневая архитектура, которая позволяет эффективнее обнаруживать объекты различных размеров.

Head также был значительно оптимизирован по сравнению с предыдущими версиями YOLO. Архитектура была оптимизирована за счет добавления CSP и skip-connections, как и в предыдущих компонентах. В YOLOv8 используются более быстрые алгоритмы свертки (Depthwise Separable Convolutions), потребляющие меньшее количество вычислительных ресурсов, а также алгоритмы пост-обработки, такие как NMS (Non-Maximum Suppression), что также способствует уменьшению времени необходимого для обработки выходных данных.

4.2 Описание функций YOLOv8

Для понимания дальнейших экспериментов следует рассмотреть функции, которые будут использоваться при обучении и проверке модели, а также их параметры.

4.2.1 Predict

Прогнозирование или инференс – процесс осмысления визуальных данных в области компьютерного зрения. Функция Predict от Ultralytics YOLOv8 предназначена для быстрой и точной обработки изображений в режиме реального времени. Она использует обученную модель YOLOv8 для обнаружения объектов на изображениях, видеофайлах или видеопотоках. Функция выводит результаты работы модели в виде изображений или видео с ограничивающими рамками и вероятностями объектов в них. Одним из параметров функции можно также добавить возможность сохранения полученных результатов в файлах

					ПГУ 1.090301.33.001 ПЗ	Лист
						44
Изм.	Лист	№ докум.	Подпись	Дата		

изображений и текстовых файлов с предполагаемыми координатами объектов на них.

4.2.2 Val

Валидация позволяет оценить качество обученных моделей. Данная функция предоставляет весь необходимый функционал для оценки модели YOLOv8 на валидационном наборе данных. Она может загружать веса модели из указанного файла и валидационные данные из конфигурационного файла, после чего подавать изображения на вход модели, сравнивать полученные результаты с аннотациями и рассчитывать по ним метрики. Полученные результаты выводятся в консоль в реальном времени, а также сохраняются по указанному пути.

4.2.3 Train

Функция, предназначенная для обучения модели. Функция не нуждается в загрузке данных, чем позволяет уменьшить объем необходимого кода, содержит встроенную поддержку нескольких графических процессорах, обеспечивая быструю и удобную масштабируемость обучения. Параметры и гиперпараметры могут быть изменены через специальный конфигурационный файл или через командную строку. Функция отображает текущее состояние метрик и результатов обучения в реальном времени.

Ключевые параметры функции:

- data – путь к файлу конфигурации данных;
- epochs – количество эпох, которые будет обучаться модель;
- batch – размер пакета данных, который определяет сколько будет обработано изображений перед обновлением внутренних параметров модели. Чем больше значение, тем быстрее выполняются расчеты;
- imgsz – размер всех изображений, поступающих на вход, будет приводиться к указанному значению;
- pretrained – функция поддерживает запуск предварительно обученных моделей;
- name – название эксперимента;

- optimizer – алгоритм оптимизации, используемый на этапе обучения модели. Влияет на скорость и стабильность сходимости;
- lr0 – начальная скорость обучения. Этот параметр оказывает решающее значение для процесса оптимизации весов модели, путем влияния на скорость ее обучения;
- cos_lr – регулирует коэффициент скорости обучения с помощью планировщика во время обучения;
- lrf – значение, которое регулирует коэффициент скорости обучения в процессе обучения $lr = lr0 * lrf$. Параметр используется совместно с планировщиками скорости обучения для изменения скорости обучения с течением времени.

4.3 Проектирование ПО

Для проведения серии экспериментов с моделью необходимо подготовить Python-модуль, который будет состоять из предобработки данных, загрузки весов модели YOLOv8, обучения модели на обучающей выборке, валидации модели на проверочных данных, а также получении результатов обучения и валидации модели в виде весов модели, графика обучения, примеров данных, поступивших на вход и полученных, после прогноза модели по ним. Структура работы Python-модуля представлена на рисунке 4.2.



Рисунок 4.2 – Структура работы Python-модуля

4.4 Архитектура разрабатываемой системы

В данном подразделе описывается архитектура приложения, взаимосвязь и функционал его компонентов.

Для исследования всех необходимых показателей модели и ее обучения будет использоваться два основных компонента: предварительная обработка данных и обучение модели.

Этап предварительной обработки включает в себя деление входных изображений на фрагменты заданной размерности, удаление фоновых изображений, а также приведение всех объектов к одному классу. На этом этапе пользователь должен указать размерность фрагментов, на которые будет делиться изображение, путь к директории с изображениями, а также к директории, куда будут сохранены полученные фрагменты и файлы аннотаций к ним.

На втором этапе пользователю необходимо выбрать модель для обучения, после чего настроить все интересующие его параметры и гиперпараметры.

Взаимодействие всех модулей системы представлено на рисунке 4.3.

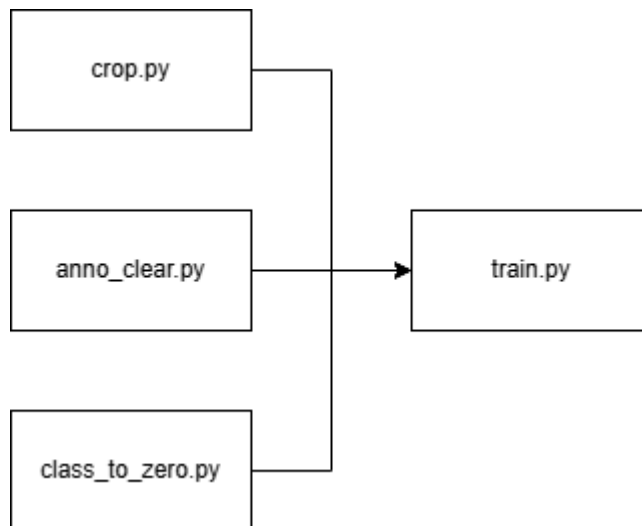


Рисунок 4.3 – Диаграмма компонентов

Из представленной выше диаграммы компонентов следует, что взаимодействие с приложением происходит через основной файл «train.py», в котором задаются входные аргументы для управления работой ПО.

Файлы «crop.py», «anno_clear.py» и «class_to_zero.py» представляют собой предварительную обработку данных, а именно разделение изображений на фрагменты, удаление фоновых изображений и файлов аннотации к ним и приведение всех объектов к одному общему классу соответственно.

					ПГУ 1.090301.33.001 ПЗ	Лист
						48
Изм.	Лист	№ докум.	Подпись	Дата		

5 Эксперименты

В данном разделе будут рассмотрены и проанализированы обучающие, валидационные и тестовые запуски модели YOLOv8. Будут рассмотрены используемые для анализа результатов моделей метрики, а также будет приведена инструкция по настройке виртуального окружения, в котором будут проведены все запуски.

5.1 Настройка виртуального окружения

В исследовании будет использоваться Python версии 3.9.18. После установки Anaconda запускаем терминал Anaconda Prompt и вводим следующие команды:

– создание окружения;

```
conda create -n <название окружения> python=3.9.18
```

– выбор созданного окружения;

```
conda activate <название окружения>
```

– установка фреймворка Ultralytics и всех необходимых зависимостей

```
pip install ultralytics sahi scikit-image imagecodecs imageio-ffmpeg.
```

После установки всех необходимых зависимостей, проект будет полностью готов к работе. Взаимодействовать с проектом можно как в открытом терминале Anaconda Prompt, так и в IDE с поддержкой Conda.

5.2 Метрики

Для оценки точности модели будут использованы метриками Precision, Recall, mAP50, mAP50-95. Перед их непосредственным рассмотрением необходимо разобрать, что представляет из себя такой показатель как IoU.

IoU – (Intersection over Union или пересечение объединения) — это число, которое количественно определяет степень перекрытия между двумя блоками. В случае обнаружения и сегментации объектов IoU оценивает перекрытие реальной

области и спрогнозированной. Графическое представление метрики изображено на рисунке 5.1.

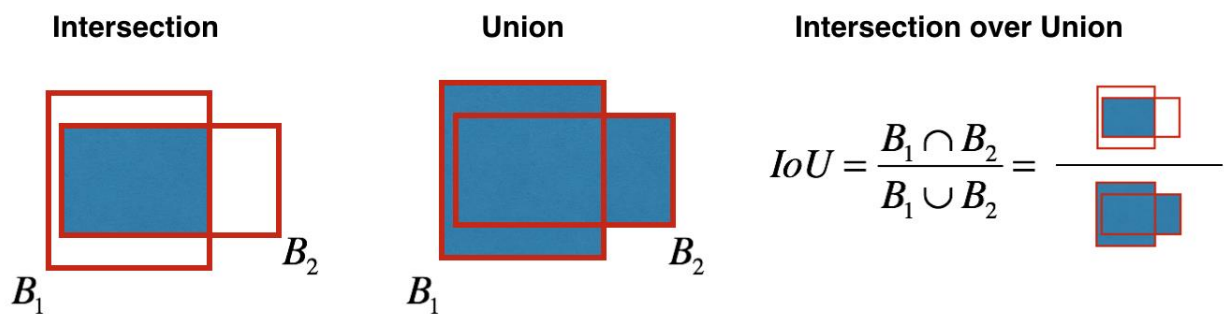


Рисунок 5.1 – Графическое представление IoU

С помощью порога IoU определяется, является ли прогноз истинно положительным (*TP*), ложно положительным (*FP*) или ложно отрицательным (*FN*).

Истинно положительный – количество правильно классифицированных объектов.

Ложно положительный – количество ложно спрогнозированных или классифицированных объектов.

Ложно отрицательный – количество необнаруженных объектов.

Рисунок 5.2 содержит 3 изображения, содержащих графическое представление TP, FP и FN.



Рисунок 5.2 – Графическое представление TP, FP, FN

IoU — это отношение площади пересечения к объединенной площади предсказания и достоверности. Поскольку значения TP, FP и FN представляют

собой не что иное, как площади или количество пикселей, IoU может быть представлен с помощью формулы 5.1.

$$IoU = \frac{TP}{(TP+FP+FN)}, \quad (5.1)$$

Precision (точность) – доля правильно классифицированных объектов относительно общего количества обнаруженных объектов, формула расчета представлена в формуле 5.2.

$$Precision = \frac{TP}{TP+FP}, \quad (5.2)$$

Recall (полнота) – доля правильно классифицированных объектов относительно общего количества объектов в наборе данных, формула расчета представлена в формуле 5.3.

$$Recall = \frac{TP}{TP+FN}, \quad (5.3)$$

Для понимания двух оставшихся метрик необходимо разобрать, что представляет собой Average Precision (AP) или средняя точность.

AP – метрика, которая оценивает точность модели по одному конкретному классу. Она рассчитывается на основе кривой точности-полноты (Precision-Recall curve) и на практике представляет собой точность, усредненную по всем значениям полноты (когда показатели точности и полноты высокие, AP тоже высокий, но если любой из показателей будет низок, то в независимости от второго показателя, AP также будет низок).

mAP50 – средняя точность по всем классам объектов при значении порога IoU = 50% или 0.5.

mAP50–95 – средняя точность по всем классам объектов при значении порога IoU от 50% до 95% или от 0.5 до 0.95 с определённым шагом (наиболее часто используемое значение, которое также будет использоваться в данном исследовании – 0.05).

Так как исследование будет проводится на одном классе, то метрики mAP50 и mAP50-95 так же можно интерпретировать как AP50 и AP50-95 соответственно.

5.3 Сравнение моделей семейства архитектур YOLOv8

Существует множество предварительно обученных моделей YOLOv8, находящихся в открытом доступе, в том числе модели. В их число входят 5 моделей, которые были предварительно обучены и помещены в репозиторий Ultralytics на GitHub. Вне зависимости от того, на чем обучалась предварительно обученная модель и для какой задачи и на каких данных ее планируется обучать в дальнейшем, если есть возможность использовать ее веса, вместо случайно инициализированных весов, то это необходимо делать, так как это снизит время, затрачиваемое на достижение лучших результатов.

Таблица 5.1 – Предварительно обученные модели YOLOv8

Модель	Размер изображения	mAP ^{val} ₅₀₋₉₅	Количество параметров, млн.	FLOPs, млрд.	FPS
YOLOv8n	640	37.3	3.2	8.7	53
YOLOv8s	640	44.9	11.2	28.6	27
YOLOv8m	640	50.2	25.9	78.9	9
YOLOv8l	640	52.9	43.7	165.2	2.3
YOLOv8x	640	53.9	68.2	257.8	-

Исходя из таблицы, можно сделать вывод, что минимальным требованиям по FPS соответствует только модель YOLOv8n, поэтому все дальнейшие эксперименты будут направлены на улучшение ее точности в поставленной задаче.

5.4 Оборудование

При обучении модели YOLOv8n использовался шести ядерный процессор Intel Core i5-9400F OEM, 32 Гб оперативной памяти, представленной 4 DIMM и видеокарта NVIDIA GeForce RTX 3060 с объемом оперативной памяти 12 Гб,

3584 ядрами NVIDIACUDA и показателем тактовой частоты с ускорением равным 1.78.

На описанном оборудовании время обучения модели YOLOv8n на обучающей выборке составляет 46 минут при значениях параметров batch=32 и imgsz=640 с валидацией после каждой эпохи.

5.5 Выбор оптимизатора

Одним из наиболее важных параметров, оказывающим большое влияние на обучаемость моделей искусственных нейронных сетей, является оптимизатор.

Оптимизаторы – это алгоритмы, используемые для минимизации функции потерь в процессе обучения моделей машинного обучения. Путем поиска оптимальных значений параметров моделей они минимизируют ошибку прогноза.

Его выбор, чаще всего, сводится к выбору между Adam и SGD. Это связано с тем, что они не являются узко специализированными в отличие от многих других оптимизаторов и эффективно справляются с решением большинства известных задач.

Стохастический градиентный спуск (SGD) – классический оптимизатор, который использует градиент только одной случайной выборки для обновления параметров.

Adam – модифицированный SGD, использующий экспоненциально взвешенное среднее градиентов и их квадратов, чтобы адаптировать скорость обучения для каждого параметра.

Для любых задач прототипирования (меньше 10000-30000 эпох обучения) рекомендуется использовать Adam. SGD может превосходить Adam для сверточных слоев нейронных сетей, однако подобрать оптимальный коэффициент скорости обучения для него намного тяжелее.

Для выбора оптимизатора будет проведено несколько экспериментов. Каждый из двух оптимизаторов будет использоваться при обучении модели в условиях статичного коэффициента скорости обучения и динамичного, который будет регулироваться с помощью адаптивного планировщика.

					ПГУ 1.090301.33.001 ПЗ	Лист
						53
Изм.	Лист	№ докум.	Подпись	Дата		

Адаптивный планировщик (adaptive scheduler) – это система, которая динамически регулирует процесс обучения модели машинного обучения, оптимизируя скорость обучения и количество эпох, для получения лучшего результата за минимальное время. Он использует информацию, полученную в процессе обучения, для принятия решений, делая его более эффективным.

В ходе экспериментов были отключены все гиперпараметры модели, которые добавляли аугментацию в данные. Полученные результаты представлены в таблице 5.2.

Таблица 5.2. Сравнение оптимизаторов

Оптимизатор	N	L	X	Adam без AS	Adam с AS	SGD без AS	SGD с AS
Precision	0.326	0.49	0.506	0.469	0.429	0.651	0.518
Recall	0.206	0.267	0.282	0.251	0.261	0.482	0.308
mAP50	0.169	0.295	0.31	0.257	0.236	0.517	0.318
mAP50-95	0.054	0.0937	0.104	0.103	0.0911	0.216	0.121

Для визуализации изменений модели в процессе обучения будет использоваться график метрики mAP50-95, как наиболее полно отображающий изменения модели. Результаты представлены на рисунке 5.3.

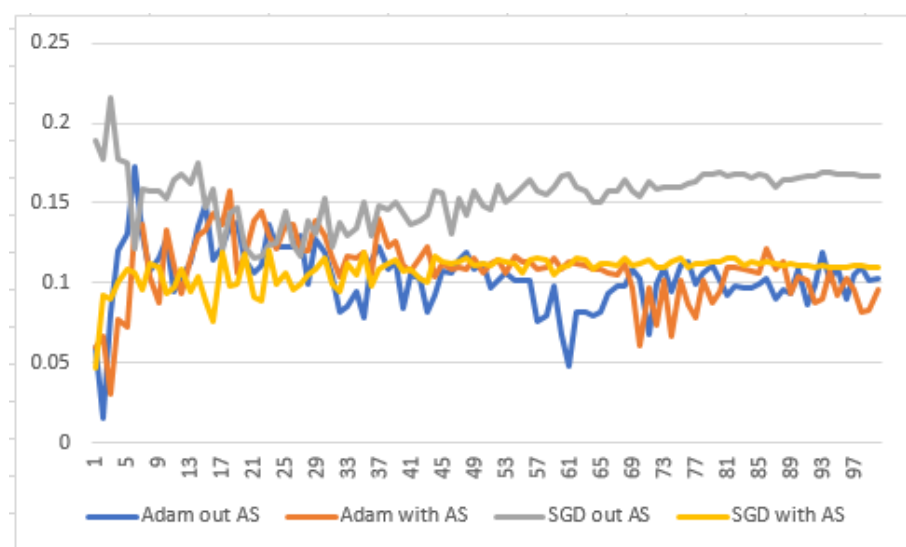


Рисунок 5.3 – Сравнение показателей метрики mAP50-95

Из графика и таблицы следует, что на 100 эпохах обучения без использования адаптивного планировщика с оптимизатором SGD модель показывает наилучший результат. Однако, для долгого обучения модели планировщик рекомендуется применять. Кроме того, можно сделать вывод, что для решения поставленной задачи верным решением будет использование алгоритма оптимизации SGD. Как показано на графике алгоритм также ведет себя более стабильно, что делает результат следующих эпох более прогнозируемым.

5.6 Аугментация входных данных

Следующим шагом, после выбора оптимизатора, будет добавление аугментации в данные. Необходимо изучить влияние различных методов аугментации и их сочетаний на работу обоих оптимизаторов.

В ходе исследований различных методов аугментации были изучены:

- стандартные настройки аугментации функции train – добавлена небольшая изменчивость цвета для улучшения способности работать в разных погодных условиях (hsv_h=0.015), изменена насыщенность цветов (hsv_s=0.7), изменена яркость изображения для улучшения работы при разном уровне освещения (hsv_v=0.4), изображение перемещалось по горизонтали и вертикали, создавая объекты, лишь частично видимые на изображении (translate=0.1), масштабируемость изображений, которая позволяла имитировать различное расстояние от камеры до объекта (scale=0.5), переворот изображения слева направо (fliplr=0.5), объединение четырех обучающих изображений в одно, что способствует лучшему пониманию сложных сцен (mosaic=1.0), случайное удаление части изображения на этапе классификации (erasing=0.4), обрезание изображения на этапе классификации для уменьшения количества фоновых объектов (crop_fraction=1.0);

- из стандартного набора были убраны изменения, связанные с объединением изображений, удалением части изображений и их обрезанием;

- изменения, сделанные на предыдущем шаге, негативно сказались на метриках, поэтому к стандартным настройкам были добавлены повороты

изображения с углом до 180 градусов (degrees=+180), а также перевороты изображения с углом в 360 градусов (flipud=0.3), были убраны перемещения изображений.

Полученные результаты представлены в таблице 5.3.

Таблица 5.3 – Результаты, полученные с применением аугментации

Аугментация	Стандартная аугментация YOLOv8	Стандартная без использования объединения и удаления частей изображения	Стандартная без использования объединения и удаления частей изображения с изменением угла изображения
Precision	0.723	0.63	0.32
Recall	0.254	0.224	0.235
mAP50	0.277	0.233	0.172
mAP50-95	0.191	0.159	0.0812

На основе данных, представленных в таблице 5.3, можно предположить, что базовый набор методов аугментации данных функции train является наиболее подходящим для решения поставленной задачи. Для утверждения предположения был проведен ряд экспериментов с различными комбинациями упомянутых ранее гиперпараметров модели и их значениями, однако все полученные модели демонстрировали более плохие метрики. Таким образом, можно сделать вывод, что выдвинутое ранее предположение оказалось верным и использование параметров аугментации с значениями по умолчанию является наиболее эффективным решением.

5.7 Результаты на конечном устройстве

При проверке полученной модели на конечном устройстве будем предполагать, что видеопоток, подаваемый на вход искусственной нейронной сети, может в разрешении отличном от того, на котором обучалась модель. Для

того, чтобы нейронная сеть не тратила время на сжатие кадров, которое значительно ухудшит ее точность, воспользуемся фреймворком SAHI (Slicing Aided Hyper Inference).

SAHI – библиотека машинного зрения, предназначенная для оптимизации алгоритмов обнаружения объектов на изображениях высокого разрешения. Она разбивает изображения на фрагменты, которые передаются модели для выполнения прогнозирования, после чего полученные фрагменты объединяются в исходное изображение. Фреймворк совместим с рядом современных детекторов объектов, в том числе серией YOLO.

При использовании SAHI на видеофайлах, которые не требуют деления на фрагменты, модель показывает FPS схожие с предварительно обученной YOLOv8n – 53 обрабатываемых кадра в секунду.

Кадры видеофайлов, стороны которых больше заданных значений, разделяются на фрагменты размером 640x640, после чего, поступают на вход нейронной сети. В случае необходимости применения предварительной обработки изображений со стороны SAHI значение FPS равняется 27 кадра в секунду.

Полученные результаты прогнозирования модели с использованием фреймворка SAHI на изображениях со сторонами больше заданных представлены на рисунке 5.4.



Рисунок 5.4 – Результат прогнозирования модели нейронной сети

6 Безопасность жизнедеятельности

Пренебрежение правилами безопасности при работе с компьютером может привести к негативным последствиям для пользователя в виде ущерба для здоровья и имущества. Даже соблюдая технику безопасности как с работой со сложным бытовым устройством, вы можете получить не прямой вред своему здоровью, если не будете соблюдать рекомендации врачей, специально разработанных для пользователей ПК.

6.1 Вред здоровью при работе с компьютером

Продолжительная эксплуатация персонального компьютера может привести к накопительному отрицательному воздействию на здоровье человека, т.е. ущерб долгое время может не проявляться вовсе, а незаметно и постепенно накапливаться. Одними из самых опасных воздействий на здоровье пользователя ПК являются:

- высокая нагрузка на органы зрения, которая вызывает такие синдромы как «красный глаз» или «сухой глаз»;
- болезнь суставов, вызванная неправильной позой при продолжительной работе;
- нагрузка на психику и нервную систему, возникающая при долгой концентрации внимания на большом потоке информации длительное время.

6.2 Снижение вредного воздействия на здоровье при работе с ПК

Для того, чтобы максимально эффективно убрать отрицательное влияние на здоровье пользователя, необходимо соблюдать перечень простых правил, перечисленных ниже:

- соблюдать безопасное расстояние от пользователя до монитора. Безопасным расстоянием считается 40 – 75 сантиметров;
- соблюдать правильный угол наклона экрана монитора в 25 – 30 градусов;

					ПГУ 1.090301.33.001 ПЗ	Лист
						58
Изм.	Лист	№ докум.	Подпись	Дата		

- держать прямую осанку при работе с компьютером. Для этого необходимо правильно отрегулировать рабочее кресло или стул;
- при работе с периферией ввода/вывода, а именно клавиатурой и мышью, стараться держать руки немного согнутыми, а локти должны располагаться на столе или подлокотниках. Это предотвратит напряжение в кистях;
- при работе с компьютером, ноги не должны быть согнуты под стул или кресло, а должны быть выпрямлены вперед с упором в твердую поверхность;
- каждый час необходимо делать перерыв от работы с компьютером и выполнять небольшую разминку для снятия напряжения в суставах и мышцах.

6.3 Техника безопасности при работе с компьютером

Как и для электрического устройства, персональный компьютер также имеет технику безопасности при эксплуатации. Ниже приведен список основных правил при работе с ПК:

- не рекомендуется включать компьютер в розетки без заземления. Розетки и вилки должны быть цельными и без повреждений;
- нельзя работать с компьютером при открытом корпусе системного блока;
- не включать компьютер в помещении с высокой влажностью;
- нельзя оставлять без присмотра работающий ПК на длительное время;
- запрещается работать с компьютером, который имеет внешние повреждения корпуса или изоляцию силовых кабелей. Необходимо провести замену соответствующих кабелей или обратиться в сервисный центр.

Соблюдая технику безопасности при работе с компьютером и вышеперечисленные рекомендации, можно свести риск получить травму и нанести ущерб здоровью пользователя к минимуму.

Заключение

В ходе выполнения данной ВКР были рассмотрены различные архитектуры искусственных нейронных сетей для обнаружения объектов на изображениях, полученных с БПЛА. В ходе их анализа была выбрана наиболее подходящая из них для решения поставленной задачи. В ходе дальнейшего исследования на основе заявленных требований к модели было принято решение о необходимости проведения экспериментов с моделью выбранной архитектуры YOLOv8n. В ходе экспериментов были исследованы наиболее известные алгоритмы оптимизации — Adam и SGD, влияние адаптивного планировщика на их эффективность, а также влияние различных методов аугментации изображений на итоговый результат модели с применением этих оптимизаторов.

В процессе исследования было установлено, что при обучении моделей YOLOv8n для достижения наилучших результатов при решении поставленной задачи следует использовать алгоритм оптимизации SGD, а также следующие методы аугментации изображений: изменение цветовой палитры на изображении, изменение насыщенности цветовой палитры, изменение яркости изображения, незначительные перемещения изображения в случайную сторону, изменение масштаба изображения, переворот изображения, объединение нескольких изображений в одно и удаление случайной части изображения. Несмотря на то, что при использовании адаптивного планировщика метрики модели показывали более плохие значения, чем при обучении без него, на большом количестве эпох адаптивный планировщик будет предпочтительнее.

Для запуска модели, полученной в ходе исследования, на конечном устройстве использовался Python-фреймворк SAHI, который, в случае превышения входными данными допустимого размера сторон изображения, разделял изображения на несколько фрагментов заданного размера, благодаря чему точность модели значительно возросла. Использование дополнительного этапа предварительной обработки данных привело к снижению количества обрабатываемых кадров в секунду на конечном устройстве с 53 до 27.

					ПГУ 1.090301.33.001 ПЗ	Лист
						60
Изм.	Лист	№ докум.	Подпись	Дата		

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ASTRIAL GmbH. Using Unmanned Aerial Vehicles (UAVs) to detect abnormal behavior exhibited by individuals and vehicles within APPRAISE. [Электронный ресурс]. URL: <https://appraise-h2020.eu/node/127>. (дата обращения: 19.04.2024).
2. Волкова К. Тактика работы дронов во время поисково-спасательных мероприятий. [Электронный ресурс]. URL: <https://skymec.ru/blog/drone-use-cases/bezopasnost/rabota-dronov-vo-vremya-psm>. (дата обращения: 19.04.2024).
3. R. Koslowski. Drones Along Borders: Border Security UAVs in the United States and the European Union. [Электронный ресурс]. URL: https://www.researchgate.net/publication/322909294_Drones_Along_Borders_Border_Security_UAVs_in_the_United_States_and_the_European_Union. (дата обращения: 19.04.2024).
4. Kenk, V.S., Križaj, J., Štruc, J. & Dobrišek S., Smart Surveillance Technologies in Border Control, in European Journal of Law and Technology, Vol 4., No. 2., 2013. (дата обращения: 19.04.2024).
5. Zhang, M., Tang, F., Ren, Y., & Yan, C. (2017). A Review of Image-based Traffic Flow Estimation and Vehicle Detection and Tracking Techniques. Journal of Traffic and Transportation Engineering (English Edition), 4(4), 346-361. [Электронный ресурс]. URL: <https://doi.org/10.1016/j.jtte.2017.06.001>. (дата обращения: 19.04.2024).
6. Qureshi, S., & Shah, M. (2008). Real-time detection and tracking of pedestrians with non-stationary cameras. In Proceedings of the 2008 IEEE Conference on Computer Vision and Pattern Recognition (pp. 1-8). [Электронный ресурс]. URL: <https://doi.org/10.1109/cvpr.2008.4587441>. (дата обращения: 19.04.2024).
7. Xu, L., Ren, C., Liu, C., & Jia, J. (2018). Vehicle detection on drones: A benchmark dataset. In 2018 International Joint Conference on Neural Networks (IJCNN) (pp. 1-8). [Электронный ресурс]. URL: <https://doi.org/10.1109/ijcnn.2018.8489108>. (дата обращения: 19.04.2024).

					ПГУ 1.090301.33.001 ПЗ	Лист
						61
Изм.	Лист	№ докум.	Подпись	Дата		

8. Duan, Y., Chen, C., & Yuan, Y. (2019). A lightweight convolutional neural network for pedestrian detection in aerial images. Remote Sensing, 11(19), 2229. [Электронныйресурс]. URL: <https://doi.org/10.3390/rs11192229>. (дата обращения: 19.04.2024).

9. Chen, L. C., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. L. (2017). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE transactions on pattern analysis and machine intelligence, 40(4), 834-848. [Электронныйресурс]. URL: <https://doi.org/10.1109/tpami.2017.2699184>. (дата обращения: 20.04.2024).

10. Zhang, S., Benenson, R., Omran, M., Hosang, J., & Schiele, B. (2018). Towards reaching human performance in pedestrian detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 5275-5284). [Электронныйресурс]. URL: <https://doi.org/10.1109/cvpr.2018.00554>. (дата обращения: 20.04.2024).

11. Leal-Taixé, L., Fenzi, M., Kuznetsova, A., Rosenhahn, B., & Savarese, S. (2017). Learning an image-based motion context for multiple people tracking. In Proceedings of the IEEE International Conference on Computer Vision (pp. 4107-4116). [Электронныйресурс]. URL: <https://doi.org/10.1109/iccv.2017.441>. (дата обращения: 20.04.2024).

12. Li, B., Zhang, D., Huang, K., & Tan, T. (2014). Multi-camera activity correlation analysis for crowd behavior understanding. Pattern Recognition, 47(3), 1136-1147. [Электронныйресурс]. URL: <https://doi.org/10.1016/j.patcog.2013.09.019>. (дата обращения: 20.04.2024).

13. Gade, R., & Moeslund, T. B. (2014). Thermal cameras and applications: A survey. Machine Vision and Applications, 25(1), 245-262. [Электронныйресурс]. URL: <https://doi.org/10.1007/s00138-013-0539-3>. (дата обращения: 20.04.2024).

14. Khan, S., & Yeasin, M. (2019). Understanding group activities from still images: An unsupervised approach. IEEE Transactions on Image Processing, 28(2), 888-901. [Электронныйресурс]. URL: <https://doi.org/10.1109/tip.2018.2873786>. (дата обращения: 20.04.2024).

15. Liu, B., Fan, X., & Zhang, Y. (2019). Deep convolutional neural network based pedestrian detection in aerial images. In 2019 IEEE International Conference on Image Processing (ICIP) (pp. 1805-1809). [Электронный ресурс]. URL: <https://doi.org/10.1109/icip.2019.8803218>. (дата обращения: 20.04.2024).

16. Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In Advances in neural information processing systems (pp. 91-99). (дата обращения: 21.04.2024).

17. Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 779-788). (дата обращения: 21.04.2024).

18. Alahi, A., Goel, K., Ramanathan, V., Robicquet, A., Fei-Fei, L., & Savarese, S. (2016). Social LSTM: Human trajectory prediction in crowded spaces. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 961-971). [Электронный ресурс]. URL: <https://doi.org/10.1109/cvpr.2016.106>. (дата обращения: 21.04.2024).

19. Sadeghian, A., Alahi, A., & Savarese, S. (2019). Sophie: An attentive GAN for predicting paths compliant to social and physical constraints. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1349-1358). [Электронный ресурс]. URL: <https://doi.org/10.1109/cvpr.2019.00145>. (дата обращения: 21.04.2024).

20. Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 779-788). [Электронный ресурс]. URL: <https://doi.org/10.1109/CVPR.2016.91>. (дата обращения: 21.04.2024).

21. Lin, T. Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal Loss for Dense Object Detection. In Proceedings of the IEEE International Conference on Computer Vision (ICCV) (pp. 2980-2988). [Электронный ресурс]. URL: <https://doi.org/10.1109/iccv.2017.324>. (дата обращения: 21.04.2024).

22. Lin, T. Y., Dollár, P., Girshick, R., He, K., Hariharan, B., &Belongie, S. (2017). Feature Pyramid Networks for Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 2117-2125). [Электронныйресурс]. URL: <https://doi.org/10.1109/CVPR.2017.106>. (дата обращения: 21.04.2024).
23. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. In European Conference on Computer Vision (ECCV) (pp. 21-37). [Электронныйресурс]. URL: https://doi.org/10.1007/978-3-319-46448-0_2. (дата обращения: 21.04.2024).
24. SeokwonYeom (2024). Thermal Image Tracking for Search and Rescue Missions with a Drone. [Электронныйресурс]. URL: <https://www.mdpi.com/2504-446X/8/2/53>. (дата обращения: 21.04.2024).
25. K.T. (2023). NTUT 4K Drone Photo Dataset for Human Detection. [Электронный ресурс]. URL: <https://www.kaggle.com/datasets/kuantinglai/ntut-4k-drone-photo-dataset-for-human-detection>. (дата обращения: 22.04.2024).
26. VisDrone (2021). DroneCrowd. [Электронный ресурс]. URL: <https://github.com/VisDrone/DroneCrowd>. (дата обращения: 22.04.2024).
27. Roboflow (2022). Aerial Person Detection Computer Vision Project. [Электронный ресурс]. URL: <https://universe.roboflow.com/aerial-person-detection/aerial-person-detection>. (дата обращения: 22.04.2024).
28. D. Cafarelli, L. Ciampi, L. Vadicamo, C. Gennaro, A. Berton, M. Paterni, C. Benvenuti, M. Passera, F. Falchi (2021). MOBDrone: a Drone Video Dataset for Man OverBoard Rescue. [Электронныйресурс]. URL: <https://aimh.isti.cnr.it/dataset/mobdrone/>. (дата обращения: 22.04.2024).
29. RICARDO DE DEIJN (2022). Detect White Clothed People With Drones. [Электронный ресурс]. URL: <https://www.kaggle.com/datasets/ricardodedeijsn/drone-detection>. (дата обращения: 22.04.2024).
30. Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... & Adam, H. (2017). MobileNets: Efficient Convolutional Neural Networks for

Mobile Vision Applications. [Электронный ресурс]. URL: <https://arxiv.org/abs/1704.04861>. (дата обращения: 22.04.2024).

31. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... & Kudlur, M. (2016). TensorFlow: A system for large-scale machine learning. In 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16) (pp. 265-283). (дата обращения: 23.04.2024).

32. Chollet, F. (2015). Keras: Deep Learning library for Theano and TensorFlow. GitHubRepository. [Электронный ресурс]. URL: <https://github.com/keras-team/keras>. (дата обращения: 23.04.2024).

33. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... & Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Advances in Neural Information Processing Systems (NeurIPS). [Электронный ресурс]. URL: <https://papers.nips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library>. (дата обращения: 23.04.2024).

34. Visual Studio Code Official Website. [Электронный ресурс]. URL: <https://code.visualstudio.com/>. (дата обращения: 24.04.2024).

35. Project Jupyter: Computational Narratives as the Engine of Collaborative Data Science. [Электронный ресурс]. URL: <https://blog.jupyter.org/project-jupyter-computational-narratives-as-the-engine-of-collaborative-data-science-2b5fb94c3c58>. (дата обращения: 24.04.2024).

36. PyCharm Official Website. [Электронный ресурс]. URL: <https://www.jetbrains.com/pycharm/>. (дата обращения: 24.04.2024).

37. Label Studio Official Website. [Электронный ресурс]. URL: <https://labelstud.io/>. (дата обращения: 25.04.2024).

38. CVAT Official Website. [Электронный ресурс]. URL: <https://cvat.org/>. (дата обращения: 25.04.2024).

39. Amazon S3 Official Website. [Электронный ресурс]. URL: <https://aws.amazon.com/s3/>. (дата обращения: 25.04.2024).

40. Introduction to Pachyderm. [Электронный ресурс]. URL: <https://medium.com/better-programming/introduction-to-pachyderm-89b4f866b6fc>. (дата обращения: 25.04.2024).

41. Delta Lake: An Open Source Storage Layer for Big Data Analytics. [Электронный ресурс]. URL: <https://databricks.com/blog/2019/10/07/delta-lake-open-source-storage-layer-big-data-analytics.html>. (дата обращения: 26.04.2024).

42. Introducing lakeFS: Version control for your data lakes. [Электронный ресурс]. URL: <https://treeverse.io/blog/introducing-lakefs-version-control-for-your-data-lakes/>. (дата обращения: 26.04.2024).

43. How to Set Up an S3-Compatible Object Storage Server with Minio on Ubuntu 20.04. [Электронный ресурс]. URL: <https://www.digitalocean.com/community/tutorials/how-to-set-up-an-s3-compatible-object-storage-server-with-minio-on-ubuntu-20-04>. (дата обращения: 26.04.2024).

44. Weights & Biases (WandB) Official Website. [Электронный ресурс]. URL: <https://wandb.ai/>. (дата обращения: 26.04.2024).

45. Version Control for Machine Learning Projects. [Электронный ресурс]. URL: <https://towardsdatascience.com/version-control-for-machine-learning-projects-7c5c0d50f90e>. (дата обращения: 27.04.2024).

46. Git Large File Storage (Git LFS). [Электронный ресурс]. URL: <https://git-lfs.github.com/>. (дата обращения: 27.04.2024).

47. DVC (Data Version Control) Official Website. [Электронный ресурс]. URL: <https://dvc.org/>. (дата обращения: 27.04.2024).

48. MLflow на платформе GitHubPages. [Электронный ресурс]. URL: <https://mlflow.org/docs/latest/index.html>. (дата обращения: 27.04.2024).

49. Документация по модулю venv в Python. [Электронный ресурс]. URL: <https://docs.python.org/3/library/venv.html>. (дата обращения: 27.04.2024).

50. PipenvGitHubRepository. [Электронный ресурс]. URL: <https://github.com/ryur/pipenv>. (дата обращения: 27.04.2024).

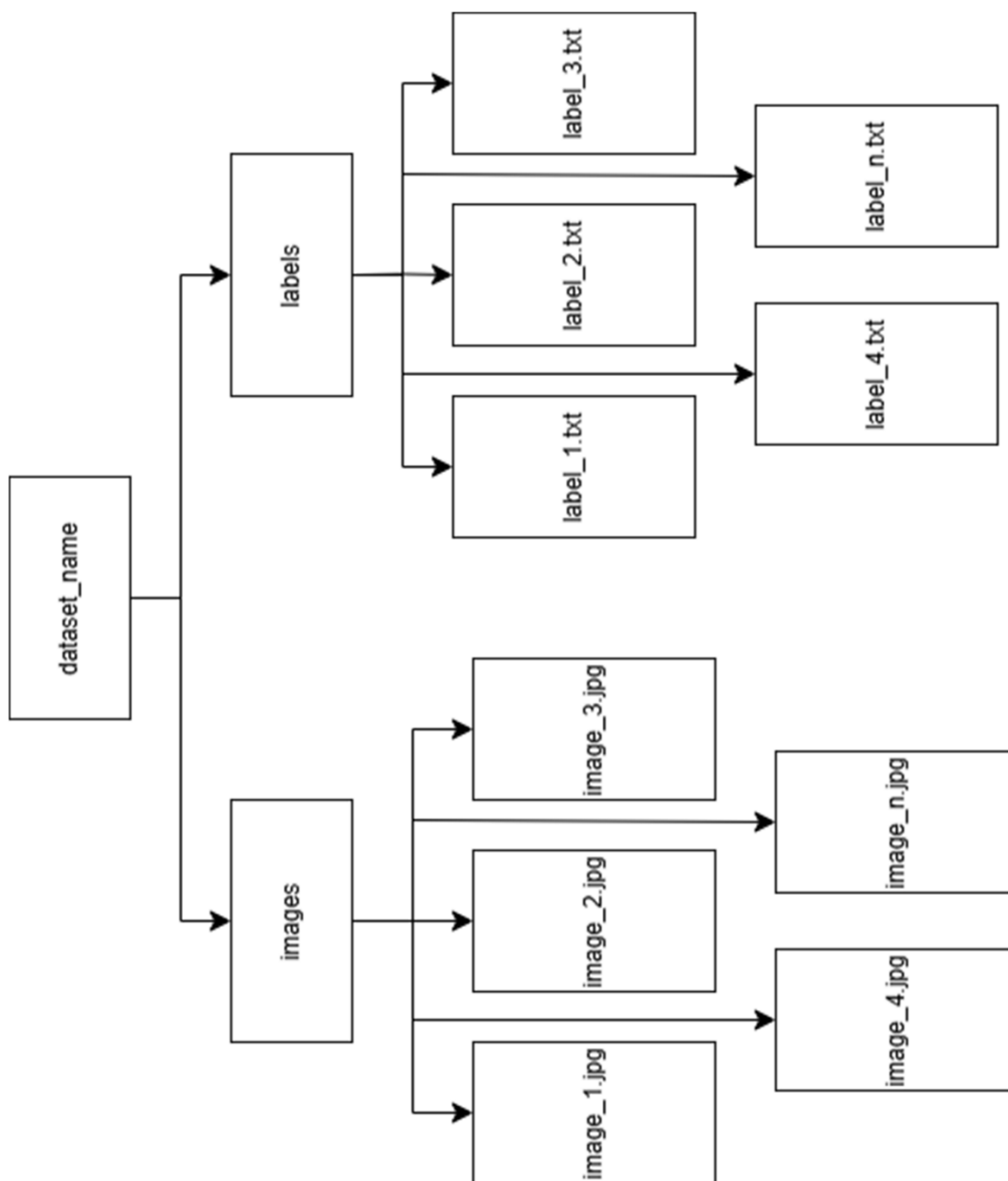
51. Conda Official Website. [Электронный ресурс]. URL: <https://www.anaconda.com/>. (дата обращения: 27.04.2024).

Приложение А

(Обязательное)

Нейросетевой детектор людей на изображениях, снятых с БПЛА
Диаграммы

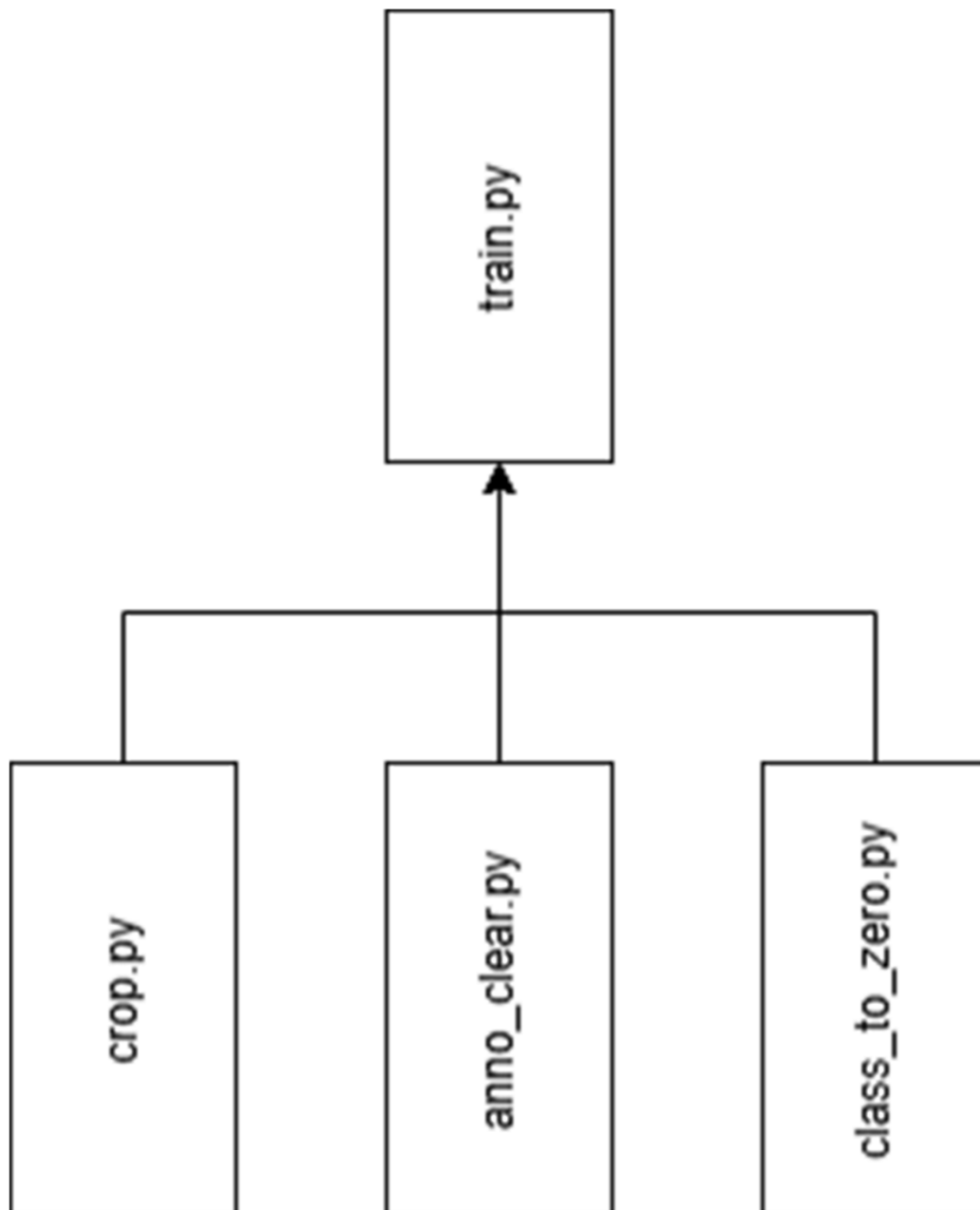
Приложение А.1 – Диаграмма хранения набора данных

[illegible]

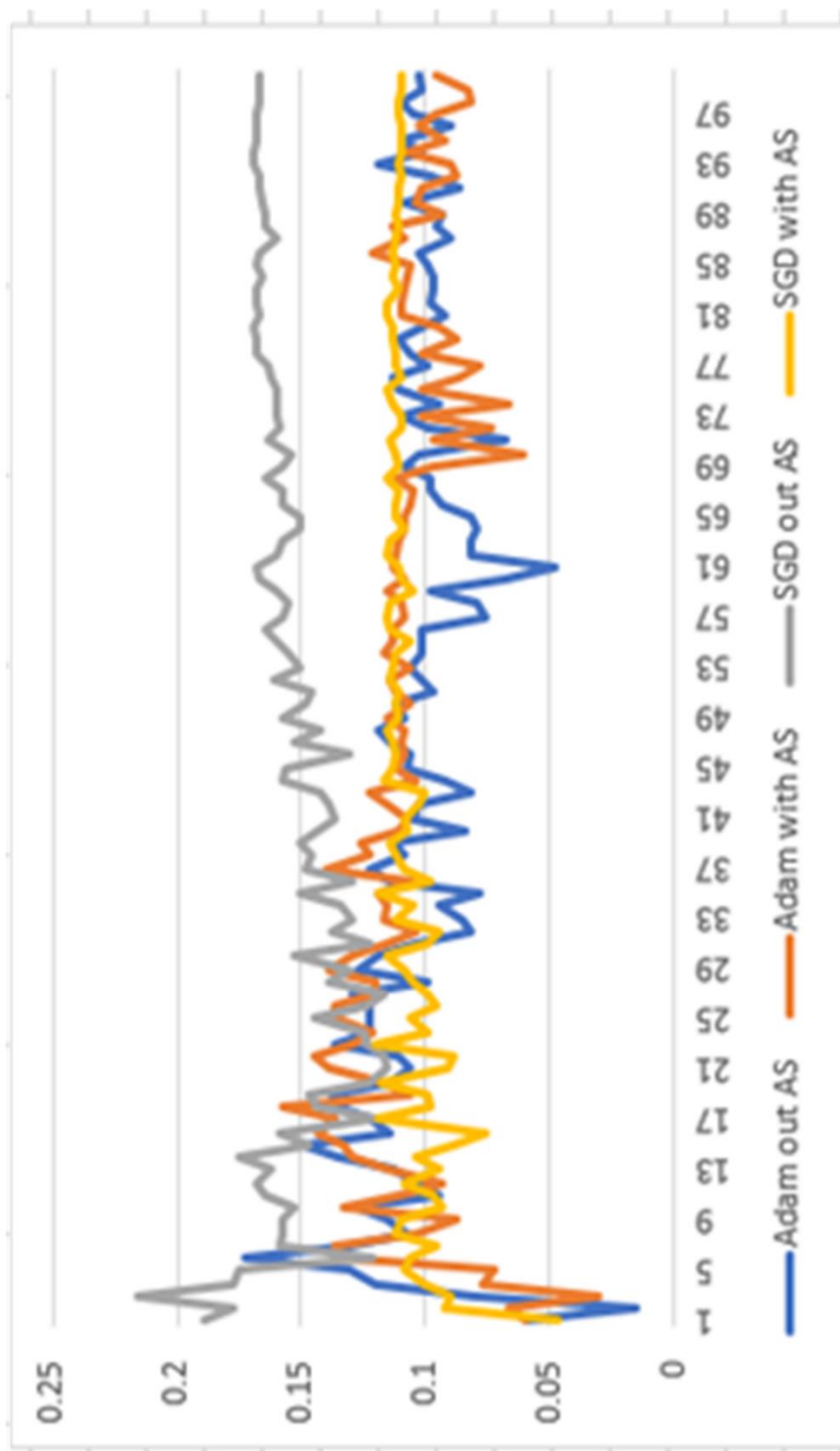
Приложение А.2 – Структурная диаграмма

[illegible]

Приложение А.3 – Диаграмма компонентов

[illegible]

Приложение А.4 – Сравнение показателей метрики tar50-95

[illegible]

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.33.001 ПЗ

Приложение Б

(Обязательное)

Нейросетевой детектор людей на изображениях, снятых с БПЛА

Листинг программы

					ПГУ 1.090301.33.001 ПЗ	Лист
						72
Изм.	Лист	№ докум.	Подпись	Дата		

Файл crop.py

```
import os
from PIL import Image

def crop_images_and_labels(input_folder, output_folder,
target_width):

    if not os.path.exists(output_folder):
        os.makedirs(output_folder)

    image_files = [f for f in os.listdir(input_folder) if
f.endswith(('jpg', 'jpeg', 'png', 'bmp'))]

    for image_file in image_files:
        input_image_path = os.path.join(input_folder, image_file)
        input_label_path = os.path.join(input_folder,
os.path.splitext(image_file)[0] + ".txt")

        # Открываем изображение
        with Image.open(input_image_path) as img:
            width, height = img.size

        target_height = int(height * (target_width / width))

        cols = width // target_width
        rows = height // target_height

        objects_found = False

        for row in range(rows):
            for col in range(cols):

                left = col * target_width
                top = row * target_height
                right = left + target_width
                bottom = top + target_height

                # Вырезаем область изображения
                cropped_img = img.crop((left, top, right, bottom))
                with open(input_label_path, "r") as
label_file:

                    lines = label_file.readlines()

                    for line in lines:
                        parts = line.strip().split()
                        if len(parts) >= 5:

                            objects_found = True
                            break
```

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.33.001 ПЗ

Лист

73

```

        if objects_found:
            output_image_path = os.path.join(output_folder,
f"{os.path.splitext(image_file)[0]}_{row}_{col}.jpg")
            cropped_img.save(output_image_path)

            with
open(os.path.splitext(output_image_path)[0] + ".txt", "w") as
new_label_file:
                for line in lines:
                    parts = line.strip().split()
                    if len(parts) >= 5:

                        class_id = parts[0]
                        x_center = float(parts[1])
                        y_center = float(parts[2])
                        box_width = float(parts[3])
                        box_height = float(parts[4])

                                if x_center * width >= left
and x_center * width <= right and y_center * height >= top and
y_center * height <= bottom:
                                    # Пересчитываем
координаты bounding box'a
                            new_x_center = (x_center * width - left) / target_width
                            new_y_center = (y_center * height - top) / target_height
                            new_box_width = box_width * width / target_width
                            new_box_height = box_height * height / target_height

                            new_label_file.write(f"{class_id} {new_x_center} {new_y_center}
{new_box_width} {new_box_height}\n")

objects_found = False

```

Файл anno_clear.py

```

import os

def remove_empty_txt_files(folder_path):
    files = os.listdir(folder_path)

    for file_name in files:
        file_path = os.path.join(folder_path, file_name)

            if file_name.endswith('.txt') and
os.path.getsize(file_path) == 0:

```

```

os.remove(file_path)
os.remove(os.path.join(folder_path, file_name.replace('.txt',
'.jpg'))))
print(f"Удален пустой файл: {file_name}")

```

Файл class_to_zero.py

```

import os

def change_class_to_zero(annotation_dir):
    for filename in os.listdir(annotation_dir):
        if filename.endswith(".txt"):
            filepath = os.path.join(annotation_dir, filename)
            with open(filepath, "r") as f:
                lines = f.readlines()
            with open(filepath, "w") as f:
                for line in lines:
                    parts = line.strip().split(" ")
            parts[0] = "0"
            f.write(" ".join(parts) + "\n")

```

Файл main.py

```

from ultralytics import YOLO
from crop import crop_images_and_labels
from anno_clear import remove_empty_txt_files
from class_to_zero import change_class_to_zero

target_width = 640
input_folder = "./datasets/v2/dataset/origin_train"
output_folder = "./datasets/v2/dataset/train"
annotation_dir =
"./datasets/v2/NTUT_4K.v1i.yolov8/train/labels"
model_path = './models/yolov8n.pt'
yaml_path = './datasets/v2/data.yaml'
path_to_video = '"./datasets/DroneVideos/Stockflue
Flyaround.mp4'

crop_images_and_labels(input_folder, output_folder,
target_width)
remove_empty_txt_files(output_folder)
change_class_to_zero(annotation_dir)

yolo_model = YOLO(model_path)
yolo_model.train(data=yaml_path, epochs=100, batch=32,
imgsz=640, save_period=1, optimizer='SGD', val=True, cos_lr=True,
hsv_h=0.015, hsv_s=0.7, hsv_v=0.4, translate=0.1, scale=0.5,
fliplr=0.5, mosaic=1.0, erasing=0.4, crop_fraction=1.0)

```

```

result = predict(
    source="./datasets/DroneVideos/Stockflue Flyaround.mp4",
    detection_model=yolo_model,
    slice_height=640,
    slice_width=640,
    overlap_height_ratio=0.2,
    overlap_width_ratio=0.2,
)

```

					ПГУ 1.090301.33.001 ПЗ	Лист
						76
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение В

(Обязательное)

Нейросетевой детектор людей на изображениях, снятых с БПЛА

Презентация

					ПГУ 1.090301.33.001 ПЗ	Лист
						77
Изм.	Лист	№ докум.	Подпись	Дата		

НЕЙРОСЕТЕВОЙ ДЕТЕКТОР ЛЮДЕЙ НА ИЗОБРАЖЕНИЯХ, СНЯТЫХ С БПЛА

АВТОР РАБОТЫ: СТУДЕНТ ГРУППЫ 20ВВП2 МИРЯСОВ Н.А.

НАУЧНЫЙ РУКОВОДИТЕЛЬ: Д.Т.Н., ДОЦЕНТ МИТРОХИН М.А.

Цель и объект работы

Цель работы: исследование архитектур моделей машинного обучения в области обнаружения объекта на изображениях, полученных с БПЛА.

Объектом разработки является программное обеспечение для предварительной обработки данных, обучения, валидации и прогнозирования модели нейросетевого детектора.

					ПГУ 1.090301.33.001 ПЗ	Лист
						78
Изм.	Лист	№ докум.	Подпись	Дата		

Стек технологий

- PyCharm + Jupyter Notebooks
- CVAT
- LakeFS
- Mlflow
- Conda

Данные полученные из открытых источников



До аннотирования



После аннотирования

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.33.001 ПЗ

Лист

79



До аннотирования



После аннотирования

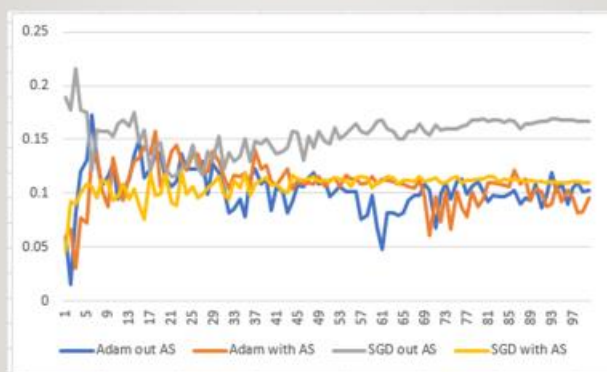
Фрагменты изображений из финального набора данных



Метрики

- Precision (точность) – доля правильно классифицированных объектов относительно общего количества обнаруженных объектов.
- Recall (полнота) – доля правильно классифицированных объектов относительно общего количества объектов в наборе данных.
- AP – метрика, которая оценивает точность модели по одному конкретному классу. Рассчитывается на основе кривой точности-полноты (Precision-Recall curve) и на практике представляет собой точность, усредненную по всем значениям полноты.
- mAP – средняя точность по всем классам.

Алгоритмы оптимизации и адаптивный планировщик



Применение аугментации

Аугментация	Стандартная аугментация YOLOv8	Стандартная аугментация без использования объединения и удаления частей изображения	Стандартная аугментация без использования объединения и удаления частей изображения с изменением угла изображения
Precision	0.723	0.63	0.32
Recall	0.254	0.224	0.235
mAP50	0.277	0.233	0.172
mAP50-95	0.191	0.159	0.0812

Результат работы модели



Заключение

В ходе выполнения выпускной квалификационной работы, были исследованы различные архитектуры нейросетевых детекторов людей на изображениях, снятых с дрона, а также разработано программное обеспечение для исследования возможности применения и получения наилучших результатов при решении поставленной задачи с помощью семейства архитектур искусственных нейронных сетей YOLOv8.

Полученные исследования могут быть использованы в различных сферах жизнедеятельности:

- Патрулирование границ
- Наблюдение за толпой во время проведения масштабных мероприятий, поиск подозрительных предметов или потенциальных злоумышленников
- Исследование социального взаимодействия людей в рамках определённого района
- Поисково-спасательные операции
- Оптимизация дорожного трафика

Спасибо за внимание!

					ПГУ 1.090301.33.001 ПЗ	Лист
						83
Изм.	Лист	№ докум.	Подпись	Дата		