Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н. Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Информатика, искусственный интеллект и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

НА ТЕМУ:

«Метод распознавания надводных объектов с аэрофотоснимков с использованием нейронных сетей»

Студент ИУ7-83Б _____ Миронов Г. А.
(Группа) (Подпись, дата) (И. О. Фамилия)

Руководитель ВКР _____ Тассов К. Л.
(Подпись, дата) (И. О. Фамилия)

Нормоконтролер _____ Мальцева Д. Ю.
(Подпись, дата) (И. О. Фамилия)

2023 г.

РЕФЕРАТ

Расчетно-пояснительная записка 67 с., 38 рис., 1 табл., 34 источн., 3 прил.

Ключевые слова: РАСПОЗНАВАНИЕ, НАДВОДНЫЕ ОБЪЕКТЫ, СВЕРТОЧНЫЕ НЕЙРОННЫЕ СЕТИ

В работе представлена разработка метода распознавания надводных объектов с аэрофотоснимков.

СОДЕРЖАНИЕ

РЕФЕРАТ **5**

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ **9**

ВВЕДЕНИЕ **10**

1 Аналитический раздел **11**

1.1 Предметная область 11

1.2 Нейронные сети 11

1.3 Сверточные нейронные сети 13

1.3.1 Свертка 14

1.3.2 Двухэтапные CNN 16

1.3.3 Одноэтапные CNN 16

1.4 Ансамбли 17

1.4.1 Стекинг 18

1.4.2 Бустинг 18

1.4.3 Бэггинг 20

1.5 Существующие методы 21

1.5.1 R-CNN 21

1.5.2 YOLO 22

1.5.3 Параметры для сравнения 25

1.5.4 Сравнение рассмотренных методов 26

1.6 Метод распознавания надводных объектов с аэрофотоснимков с использованием нейронных сетей 27

1.7 Формализованная постановка задачи 28

1.8 Выбор данных для обучения модели 28

1.8.1 Разрешение изображений 28

1.8.2 Требования к данным 29

1.9 Вывод 30

2 Конструкторский раздел **31**

2.1 Требования 31

2.2 Проектирование метода распознавания 31

2.2.1 Выбор семейства/метода обнаружения 31
2.2.2 IDEF0–диаграмма 32
2.2.3 Объединение результатов работы слабых экспертов . . . 32
2.3 Структура разрабатываемого программного комплекса 35
2.4 Данные для обучения модели 37
2.5 Обогащение данных 40
2.6 Формат хранения разметки данных 41
2.7 Обучение и тестирование слабых экспертов 41
2.8 Вывод 42

3 Технологический раздел 43
3.1 Средства реализации 43
3.2 Реализация программного комплекса 43
3.2.1 Модуль метода распознавания 43
3.2.2 Обучение слабых экспертов 44
3.3 Результаты обучения слабых экспертов 46
3.4 Примеры использования разработанного программного комплекса 49
3.4.1 Пример обучения модели 49
3.4.2 Пример распознавания объектов 50
3.5 Вывод 51

4 Исследовательский раздел 52
4.1 Аппаратное обеспечение 52
4.2 Размер входного изображения 52
4.3 Зернистость входного изображения 52
4.4 Зашумленность входного изображения 52
4.5 Смазанность входного изображения 52
4.6 Вывод 52

ЗАКЛЮЧЕНИЕ 53

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 54

ПРИЛОЖЕНИЕ А Модуль модели YOLOv8n 58

ПРИЛОЖЕНИЕ Б Модуль пользовательского приложения 63

ПРИЛОЖЕНИЕ В Утилитарный модуль **66**

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

В настоящей расчетно-пояснительной записке применяют следующие сокращения и обозначения.

ANN — (англ. Artificial Neural Network) Искусственная нейронная сеть

CNN — (англ. Convolutional Neural Network) Свёрточная нейронная сеть

RPN — (англ. Region Proposal Network) Сеть предсказания регионов

RoI — (англ. Region Of Interest) Область интереса

VHR — (англ. Very High Resolution) Сверхвысокое разрешение

MR — (англ. Medium Resolution) Среднее разрешение

ВВЕДЕНИЕ

Целью настоящей работы является разработка метода распознавания надводных объектов с аэрофотоснимков. Для достижения данной цели требуется решить следующие задачи:

1 Аналитический раздел

1.1 Предметная область

В настояще время все больше внимания привлекает тема автоматизации судоходства, а так же общее повышение уровня безопасности в процессе эксплуатации судов.

Методы распознавания различных надводных объектов имеют особое значение в данном контексте, так как позволяют решить множество проблем: избежание столкновений судов, осуществление автономного плавания и пр. [1].

Не менее важной является задача распознавания малых надводных объектов, поскольку традиционные методы обнаружения, основанные на использовании радара, не подходят для задачи обнаружения близко расположенных и малых объектов [2].

Кроме того, обнаружение активности рыболовецких судов по-прежнему является сложной задачей для многих стран, расположенных на архипелагах, например — Индонезии. В настоящее время для мониторинга огромной морской акватории используется технология, использующая датчики SAR для обнаружения кораблей, разрабатываемая с 1985 года. Однако стоимость использования данной технологии является одним из основных препятствий для дальнейшего развития [3].

1.2 Нейронные сети

ANN определяется как массово параллельный распределенный процессор, состоящий из простых процессорных блоков, который обладает естественной склонностью накапливать эмпирические знания [4].

Их название и структура вдохновлены человеческим мозгом, а алгоритм работы основывается на способе, которым биологические нейроны передают сигналы друг другу.

В общем случае ANN включает в себя входной слой, выходной (или целевой) слой и, между ними, скрытый слой. Слои соединены через узлы (искусственные нейроны). Эти соединения образуют «сеть» — нейронную сеть — из взаимосвязанных узлов. Пример приведен на рисунке 1.1.

В общем случае искусственный нейрон можно представить в виде регрессионной модели [5], состоящей из входных данных, весовых коэффициентов,

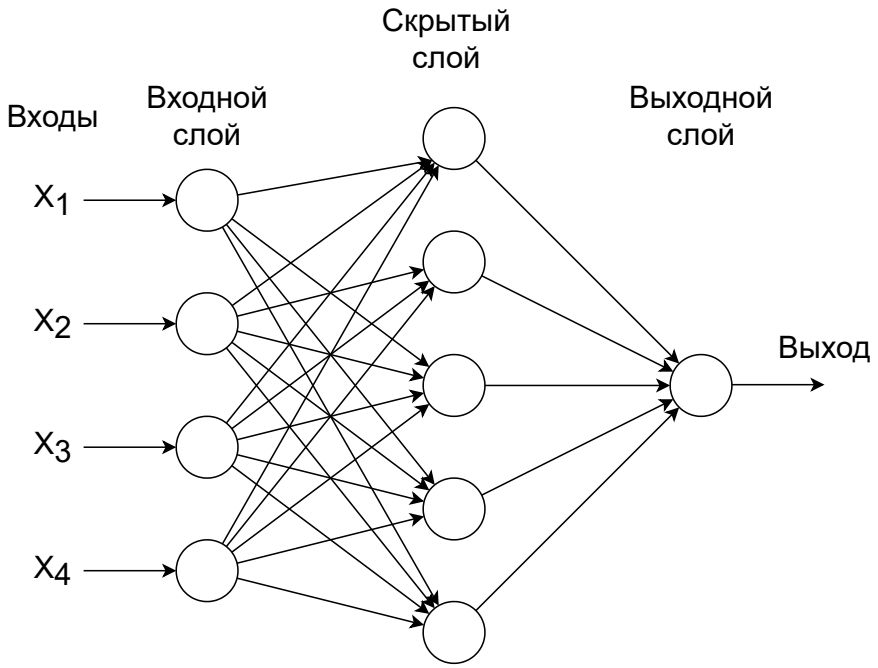

Рисунок 1.1 – Пример схемы ANN

смещения (или порогового значения) и выходных данных. Эту модель можно описать следующей формулой:

$$\hat{y} = \sum_{i=1}^{n} w_i x_i + w_0, \tag{1.1}$$

В качестве функции активации можно использовать: ступенчатую, линейную функции, сигмоиду, ReLu и другие [5].

Модель искусственного нейрона приведена на рисунке 1.2.

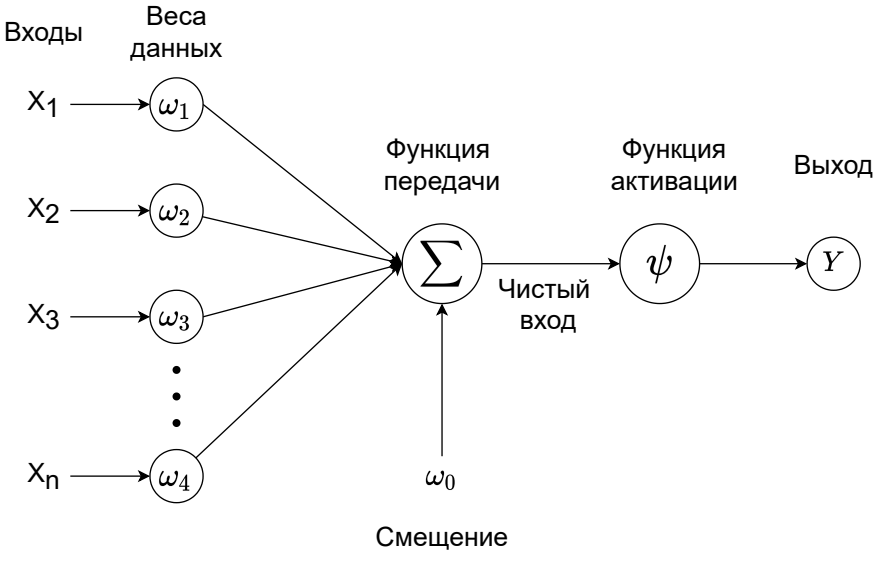

Рисунок 1.2 – Общая схема искусственного нейрона

Данное представление применимо к любому виду нейронных сетей — вне зависимости от типа, нейронные сети реализуются путем упорядочивания нейронов в слои и последующим связыванием соответствующих слоев между собой [4].

В процессе обучения нейронной сети используется так называемая обучающая выборка — заранее подготовленный набор данных, отражающий суть рассматриваемой предметной области [4]. В зависимости от содержимого обучающей выборки результирующие весовые конфигурации нейронной сети (т. е. веса связей между нейронами, а так же смещения отдельно взятых нейронов) могут отличаться [4]. В связи с этим одна и та же структура нейронной сети может переиспользована для работы в различных предметных областях.

1.3 Сверточные нейронные сети

Одним из основных видов нейронных сетей, применяемых для распознавания является CNN [6].

CNN представляет собой тип ANN, которая имеет архитектуру с глубокой обратной связью и выделяется на фоне остальных ANN с полносвязными слоями своей способностью к обобщению. CNN работает с сильно абстрагированными характеристиками объектов, особенно это касается пространственных данных, что позволяет добиться более эффективно идентифицировать объекты в сравнении с другими типами ANN [6]. Одним из отличительных свойств CNN является способность к фильтрации посторонних шумов во входных данных.

Модель CNN состоит из конечного набора уровней обработки, которые могут изучать различные характеристики входных данных (например, изображения) с несколькими уровнями абстракции. Начальные уровни изучают и извлекают высокоуровневые свойства, а более глубокие уровни изучают и извлекают более низкоуровневые свойства. Концептуальная модель CNN представлена на рисунке 1.3.

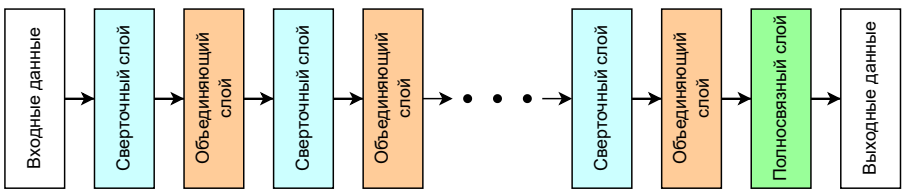

Рисунок 1.3 – Концептуальная модель CNN

Существующие архитектуры CNN для обнаружения объектов на изображениях можно разделить на две категории: одноэтапные (one-stage) и двухэтапные (two-stage) [7].

1.3.1 Свертка

Ядро

Прежде, чем рассматривать процесс свертки, необходимо определить понятие «ядро», используемое при свертке. Ядро представляет собой матрицу из дискретных значений или чисел, где каждое значение известно как вес этого ядра. Пример двухмерного ядра приведен на рисунке 1.4.

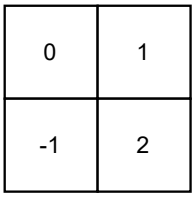

Рисунок 1.4 – Пример двумерного ядра с размерностью 2×2

Ядро инициализируется случайными значениями, которые изменяются в ходе обучения CNN.

Процесс свертки

Свертка — это операция над парой матриц $A(n_x, n_y)$ и $B(m_x, m_y)$, $m_x \leq n_x$, $m_y \leq n_y$, результатом которой является матрица

$$C(n_x - m_x + 1, n_y - m_y + 1) = A * B, \tag{1.2}$$

каждый элемент которой является скалярным произведением матрицы B (ядра свертки) и некоторой подматрицы A такого же размера.

Т. е. элемент матрицы C вычисляется следующим образом:

$$C_{i,j} = \Sigma_{u=0}^{m_x-1} \Sigma_{v=0}^{m_y-1} A_{i+u,j+v} B_{u,v}. \tag{1.3}$$

Пример

Разберем пример свертки для изображения в градациях серого, т.к. такое изображение содержит лишь один канал, передаваемый на вход CNN.

Пусть дано изображение, представленное на рисунке 1.5.

1	0	-2	1
-1	0	1	2
0	2	1	0
1	0	0	1

Рисунок 1.5 – Пример изображения в градациях серого с размерностью 4×4

Далее рассмотрим первые два шага процесса свертки, представленные на рисунках 1.6 и 1.7, соответственно.

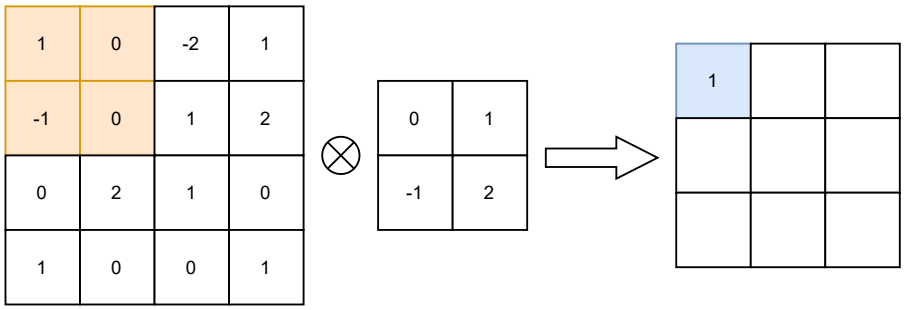

Рисунок 1.6 – Пример свертки изображения в градациях серого с размерностью 4×4

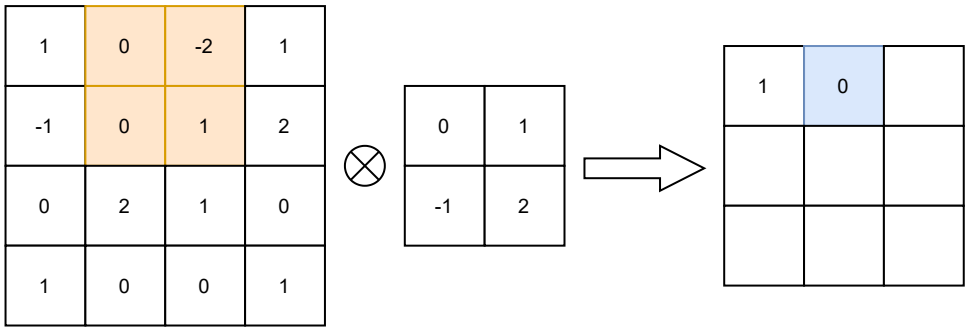

Рисунок 1.7 – Пример свертки изображения в градациях серого с размерностью 4×4

Аналогичным образом свертка продолжается до полного заполнения результирующей матрицы. Стоит отметить, что, в зависимости от размеров окна и перекрытия окон, будет меняться размер результирующей матрицы.

1.3.2 Двухэтапные CNN

В таких нейросетевых алгоритмах выделяют два этапа: поиска RoI (англ. Candidate Region Extraction) на изображении и последующей классификации RoI, найденных на первом этапе. При этом под RoI на изображении подразумеваются зоны, потенциально содержащие искомые объекты [8].

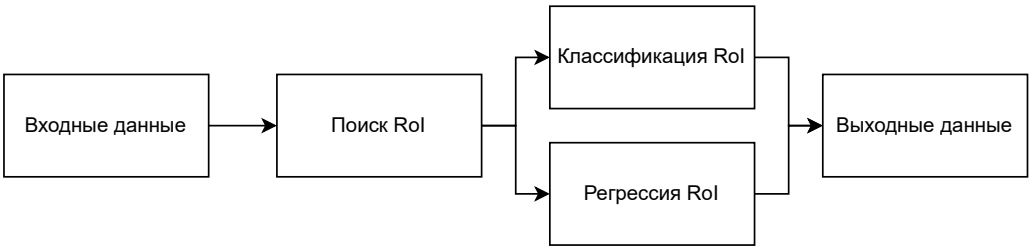

Рисунок 1.8 – Схема работы двухэтапного алгоритма

Стоит отметить, что первый этап может происходить без использования нейронных сетей. Для этого можно использовать информацию о контрасте, ключевые точки или перебор всех возможных положений объекта с помощью процедуры `selective search` [9].

RoI, полученные вышеперечисленными методами, могут обладать серьезными недостатками, например:

— содержать слишком большое количество фона;

— содержать лишь небольшую часть объекта;

— содержать более одного объекта.

В связи с этим на первом этапе более предпочтительным методом является применение CNN, не содержащих полносвязных слоев [9].

На втором этапе CNN применяются к обнаруженным RoI.

Преимуществом данных алгоритмов является высокая точность распознавания объектов, однако, платой за это является время, необходимое для выделения «подозрительных» зон на изображении [8].

1.3.3 Одноэтапные CNN

Данные нейросетевые алгоритмы не включают в себя этап поиска RoI на изображении.

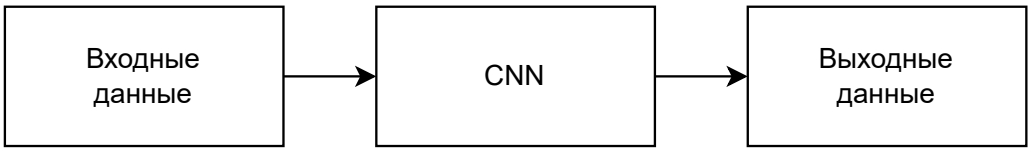

Рисунок 1.9 – Схема работы одноэтапного алгоритма

Преимуществами одноэтапных алгоритмов являются их простота и относительно высокая скорость работы. К недостаткам же можно отнести более низкую точность детектирования объектов по сравнению с двухэтапными алгоритмами, а также меньшую гибкость алгоритма с точки зрения рассматриваемых изображений [7].

В настоящее время в CNN принято выделять несколько частей, использующихся в процессе работы. Общий вид современного одноэтапного детектора представлен на рисунке 1.10.

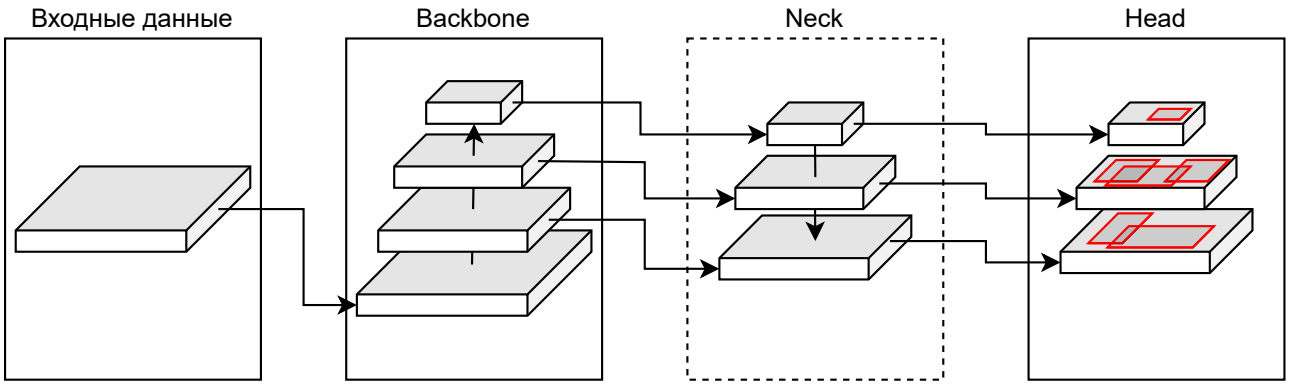

Рисунок 1.10 – Общий вид современного одноэтапного детектора

Основными частями такой CNN являются:

— Backbone;

— Neck;

— Dense Prediction.

1.4 Ансамбли

Ансамбль — это набор слабых экспертов, выполняющих распознавание произвольного объекта $x \in X$, конечный результат которого рассчитывается на основе результатов работы составляющих слабых экспертов. Слабыми экспертами в ансамбле могут выступать в том числе и другие ансамбли.

Тем не менее, ансамбль на основе нескольких моделей, построенных независимо друг от друга, в общем случае будет иметь более низкое качество, чем ансамбль на основе моделей, построенных с использованием специальных алгоритмов [10].

К этим алгоритмам относятся стекинг, бустинг и бэггинг, позволяющие существенно повысить качество классификации [10].

1.4.1 Стекинг

Стекинг подразумевает параллельные обучение и работу всех слабых экспертов, т. е. классификаторы не зависят друг от друга [10].

Схема стекинга представлена на рис. 1.11.

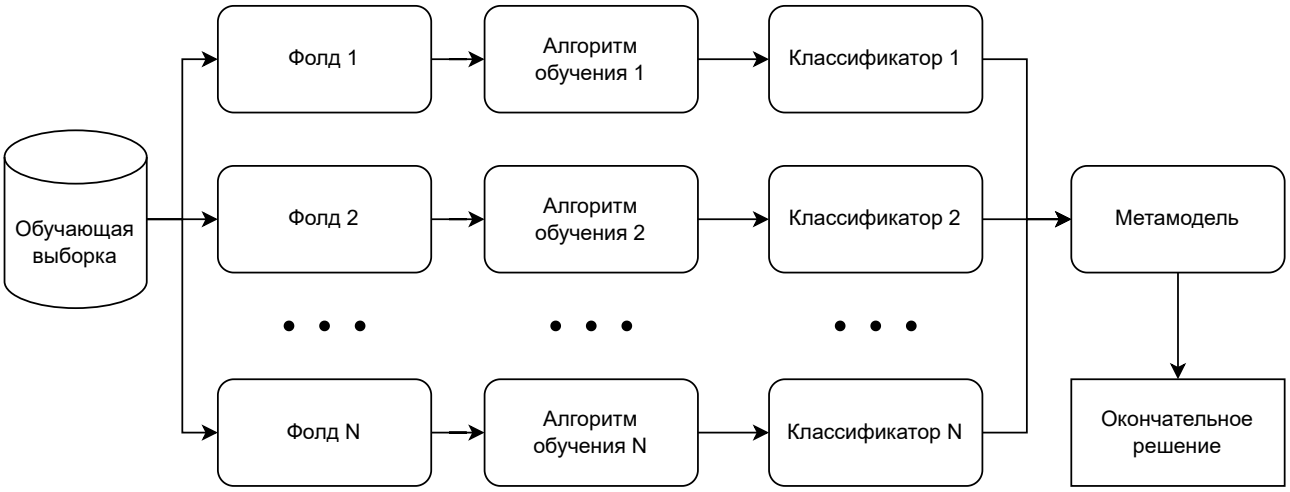

Рисунок 1.11 – Схематическое представление алгоритма стекинга

Обучающая выборка X разделяется на n случайных равновеликих частей (фолдов).

Для объекта из выборки, находящегося в k-ом фолде, производится предсказание слабыми экспертами, обученными на $k-1$ фолдах. Данный процесс итеративен и происходит для каждого фолда.

Таким образом, для каждого объекта обучающей выборки создается набор прогнозов слабых экспертов. Далее, на сформированных наборах прогнозов происходит обучение метамодели [10].

1.4.2 Бустинг

При использовании бустинга, каждый последующий алгоритм, входящий в ансамбль, стремится компенсировать недостатки композиции всех предыду-

щих алгоритмов [10].

Формальная постановка задачи

Пусть $h(a, \overrightarrow{x})$ — слабый эксперт, где $\overrightarrow{x}$ — это вектор параметров. Необходимо найти следующий алгоритм:

$$H_T(a) = \Sigma_{t=1}^{T} b_t h(a, \overrightarrow{x}), \tag{1.4}$$

где $b_i \in \mathbb{R}$ — коэффициенты, при которых

$$Q = \Sigma_i L(H_T(a_i), y_i) \to min, \tag{1.5}$$

где L — функция потерь.

Так как в общем случае процесс вычисления $\{(\overrightarrow{x}_t, b_t)\}_{t=1}^{T}$ нетривиален, решение находят пошагово:

$$H_t(a) = H_{t-1}(a) + b_t h(a, \overrightarrow{x}). \tag{1.6}$$

Схема бустинга представлена на рис. 1.12.

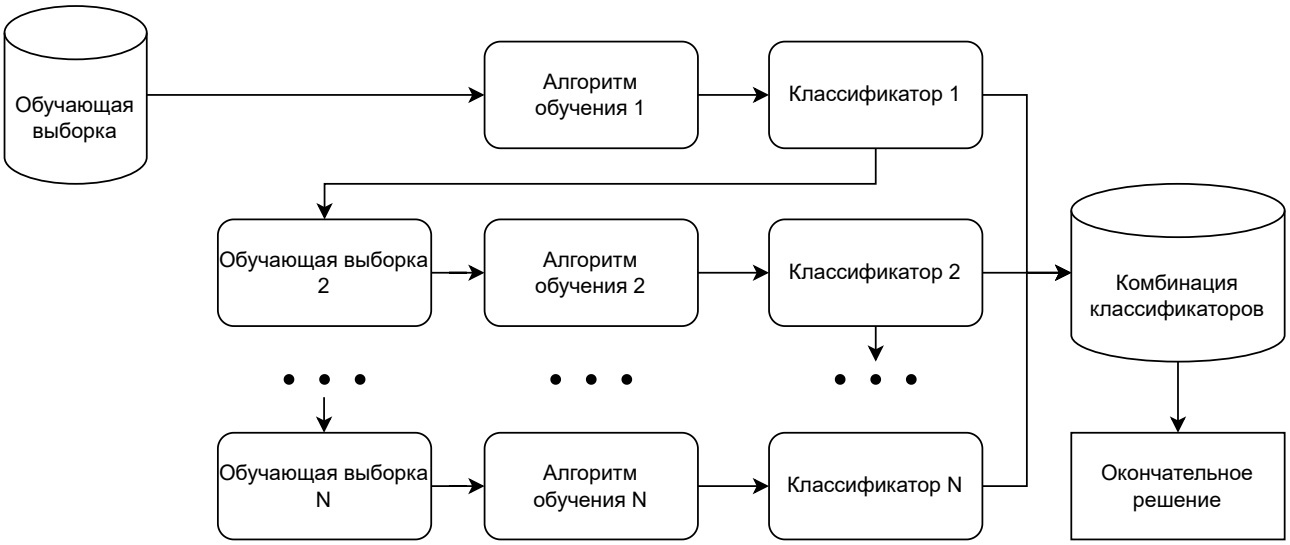

Рисунок 1.12 – Схематическое представление алгоритма бустинга

Алгоритмы

Среди наиболле часто используемых алгоритмов бустинга: AdaBoost и BrownBoost [10].

1.4.3 Бэггинг

В бэггинге все слабые эксперты обучаются и работают параллельно, т. е. независимо друг от друга. При этом обучающая выборка X разделяется на n выборок $X_1, X_2, \ldots, X_n$, причем X_i и X_j могут пересекаться при любых $i, j \in 1 \ldots n$.

Идея данного подхода заключается в том, что в отличие от бустинга классификаторы не исправляют ошибки друг друга, а компенсируют их при голосовании [10].

При этом результат голосования определяется посредством:

— консенсуса — все классификаторы дают одинаковый ответ;

— простого большинства;

— взвешивания — каждому классификатору присваивается вес, учитываемый при принятии решения.

Схема бэггинга представлена на рис. 1.13.

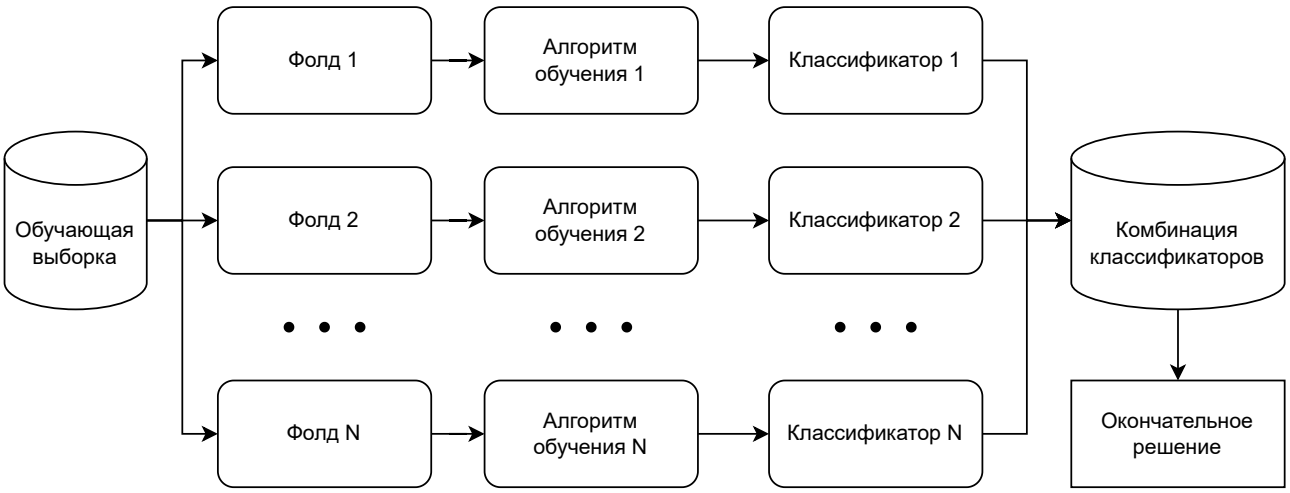

Рисунок 1.13 – Схематическое представление алгоритма бэггинга

Преимуществом данного метода перед стекингом является детерминированность результата: мета-модель в стекинге может переобучаться с течением времени, в то время как результат голосования в беггинге является детерминированным [10].

1.5 Существующие методы

По результатам проведенного исследования рынка, в настоящий момент не представлены общедоступные системы распознавания надводных объектов, в связи с чем невозможно определить методы, использующиеся в оных.

Тем не менее, в последние годы тема распознавания надводных объектов является объектом множества исследований, предлагающих разнообразные подходы к решению поставленных задач. Данные методы и будут рассмотрены далее.

1.5.1 R-CNN

Первой CNN, разработанной для распознавания объектов, является модель Region-based CNN, которая использует подходы на основе скользящего окна (sliding window) [11].

Здесь авторы разделяют всю задачу на три модуля. В первом модуле из каждого входного изображения извлекаются RoI, которые могут содержать какой-либо объект (с помощью процедуры selective search), затем во втором модуле авторы используют аффинное искажение изображения, чтобы сделать все извлеченные RoI фиксированного размера (или фиксированного соотношения сторон), а затем пропускают эти искаженные RoI через AlexNet CNN для извлечения конечных признаков (векторов признаков фиксированного размера). Наконец, третий этап представляет собой набор линейных SVM (support vector machine), которые причисляют каждый вектор какому-либо классу и отдельный регрессор обрамляющих окон [11].

Последующие версии алгоритма — Fast R-CNN [12] и Faster R-CNN [13] — призваны оптимизировать время работы алгоритма, а так же повысить точность распознавания. Так, в Faster R-CNN процедура selective search была заменена на сеть предложений регионов (RPN). RPN — это CNN, используемая для создания высококачественных RoI.

CNN данного семейства активно используются в задачах, допускающих повышение времени работы системы ради повышения точности распознавания, т.е. в задачах, не относящихся к системам реального времени.

Faster R-CNN

Данная CNN является последним опубликованным улучшением алгоритма R-CNN. Основное отличие от предшествующей Fast CNN является замена процедуры генерации претендентов избирательным поиском на нейронную сеть, которая использует имеющуюся карту особенностей [13].

Архитектура работы Faster R–CNN представлена на рисунке

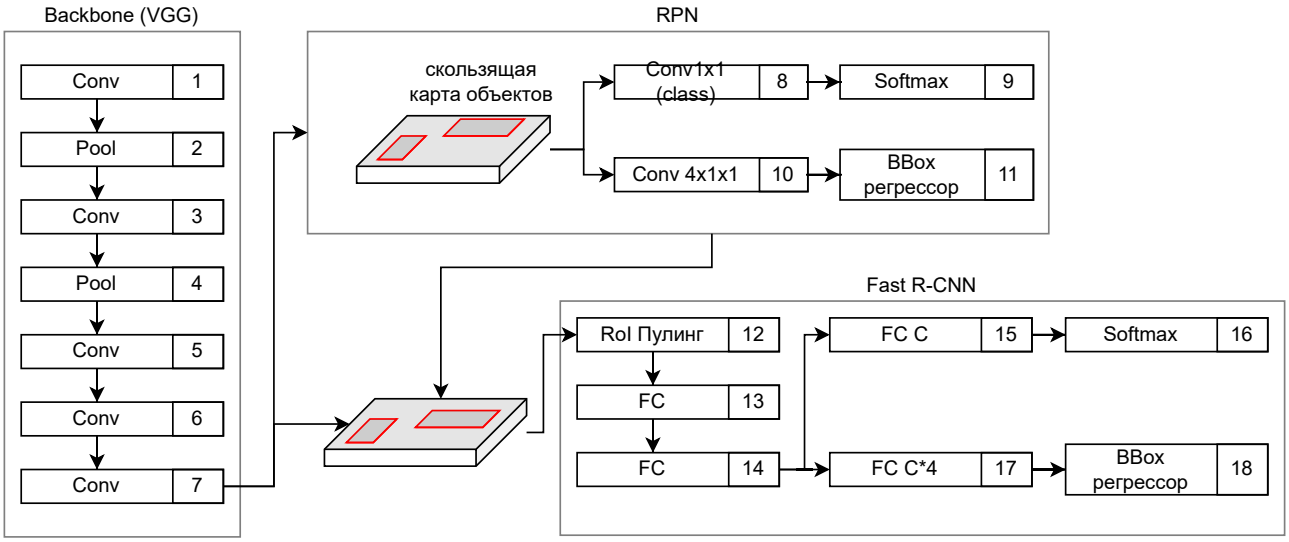

Рисунок 1.14 – Архитектура Faster R–CNN

Внесенные изменения позволяют повысить быстродействие распознавания образов на изображении вплоть до десяти раз в сравнении с предшествующей версией алгоритма [13].

1.5.2 YOLO

Алгоритм YOLO (You Only Look Once) является одноэтапным и может непосредственно распознавать объекты, а также их местоположение с помощью сквозной обученной модели CNN [14].

В исходном алгоритме YOLO входное изображение разбивается на фиксированное число сеток, а затем из каждой сетки предсказывается фиксированное число местоположений обрамляющих окон (bounding boxes) и вероятностей. Затем используется пороговое значение для выбора и определения местоположения объекта на изображении [14].

Основной проблемой YOLO является более низкая точность при распознавании больших и малых объектов, а так же присущая всем одноэтапным

алгоритмам потеря точности распознавания в сравнении с двухэтапными алгоритмами [14].

Данное семейство алгоритмов включает в себя множество оптимизаций оригинального YOLO, причем последней опубликованной является YOLOv8 [15], представленная в 2023 году.

Начиная с YOLOv4, варианты алгоритма имеют малозначительные изменения, призванные улучшить его характеристики в контексте конкретных задач [16]. Например, одним из вариантов развития актуальной на тот момент времени YOLOv3 [17] является YOLOv3 tiny [18], разработанная для решения задачи распознавания судов и кораблей в режиме реального времени.

Данное семейство CNN активно используется в задачах распознавания надводных объектов. В первую очередь, это связано с малыми затратами времени на обработку изображения, в сравнении с другими алгоритмами. Так же, достоинством YOLO является возможность распознавать большое число объектов на одном изображении [19].

YOLOv5

В настоящее время данная CNN, разработанная Ultralytics в 2020 году, пользуется большой популярностью. Данная версия была опубликована вскоре после выхода YOLOv4, однако основным ее отличием стало использование PyTorch [20] вместо Darknet [21] в качестве средства реализации. Поддержка и развитие осуществляется за счет сообщества, в связи с чем до сих пор не опубликована научная работа, описывающая данную CNN [15].

В момент написания данной работы, актуальной является версия v7.0. Кроме того, существуют следующие версии масштабирования CNN: YOLOv5n (nano), YOLOv5s (small), YOLOv5m (medium), YOLOv5l (large), YOLOv5x (extra large).

Отдельно стоит отметить, что перечисленные выше версии данной CNN имеют так же варианты с различным числом выходных слоев для объектов — от P3 до P6, соответственно, и обозначаемые, например YOLOv5s6 для YOLOv5s с P6.

В зависимости от версии масштабирования изменяется точность и время обработки единичного изображения, соответственно [15].

Архитектура YOLOv5 состоит из следующих частей: CSP-Darknet53

в качестве Backbone, SPPF и PANet в качестве Neck и Head аналогичный YOLOv4. В общем виде YOLOv5 представлена на рисунках 1.15 и 1.16.

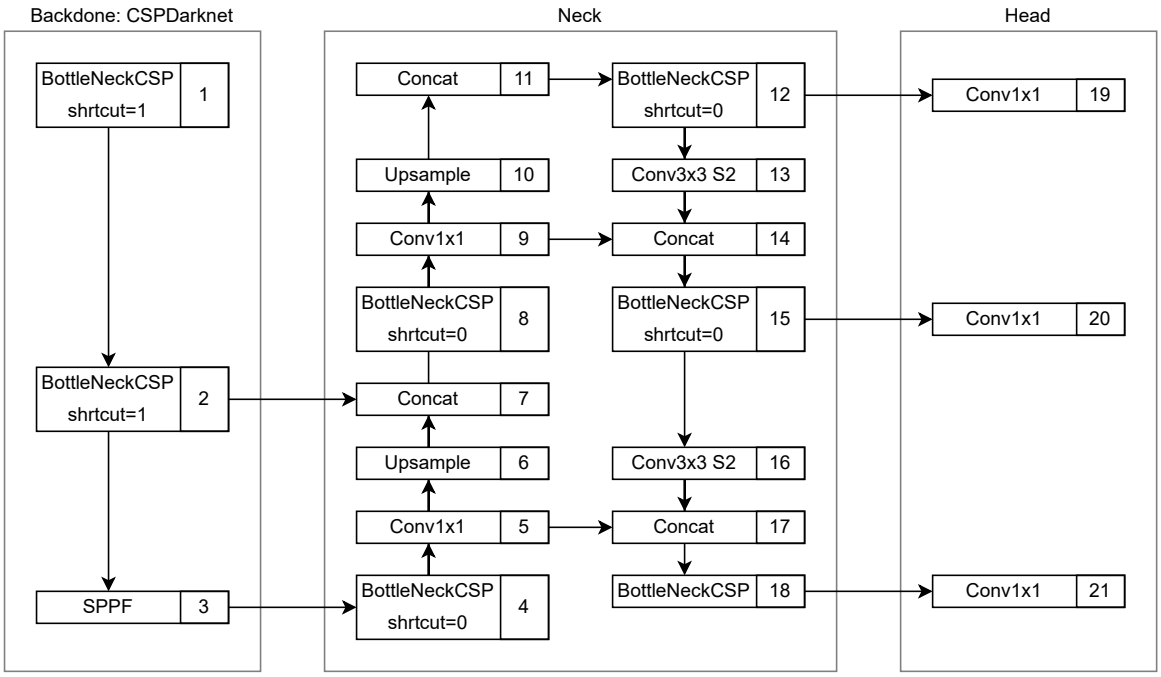

Рисунок 1.15 – Архитектура YOLOv5. Часть 1

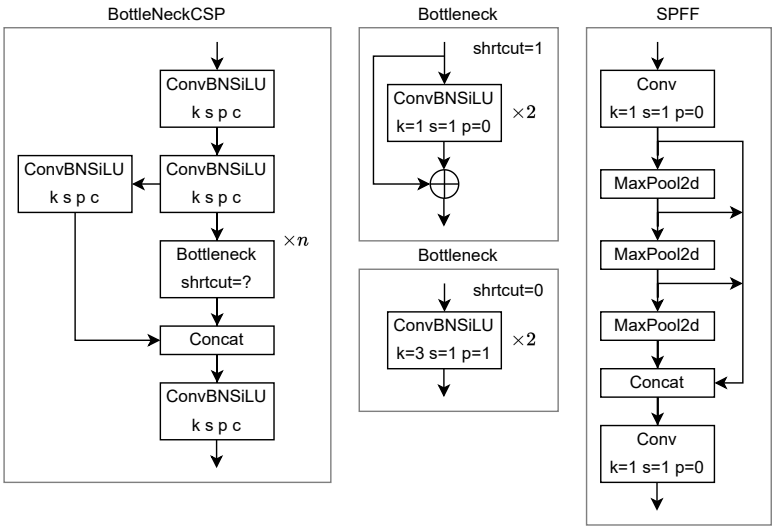

Рисунок 1.16 – Архитектура YOLOv5. Часть 2

YOLOv8

Данная CNN так же разработана Ultralytics. В ней применены несколько новых подходов как к процессу обучения, так и к самой архитектуре сети.

Аналогично YOLOv5, существует 5 масштабных версий от YOLOv8n до YOLOv8x включительно.

К сожалению, из-за новизны данной архитектуры, еще не были опубликованы какие-либо работы, сравнивающие точность работы и производительность новой архитектуры с предыдущими версиями и, в частности, с YOLOv5.

Исходя из представленной разработчиком информации, данная CNN имеет более высокую точность распознавания объектов, однако уступает предыдущей версии в быстродействии.

Общий вид данной архитектуры представлен на рисунках 1.17 и 1.18.

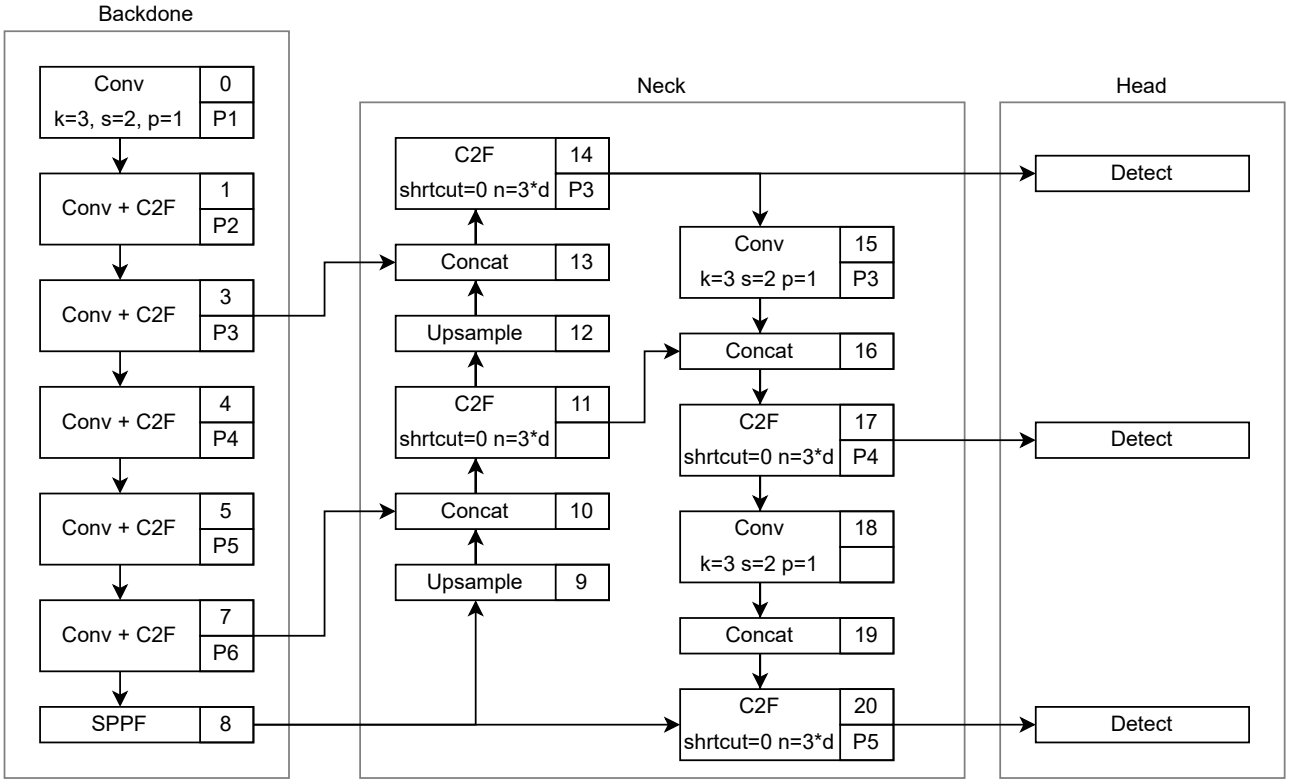

Рисунок 1.17 – Архитектура YOLOv8. Часть 1

1.5.3 Параметры для сравнения

Для оценки точности работы нейронной сети используется величина AP (Average Precision), вычисляемая следующим образом:

$$AP = \frac{\text{кол-во верно распознанных объектов}}{\text{общее кол-во распознанных объектов}}. \tag{1.7}$$

Отметим, что AP вычисляется для объектов одного класса, в связи с чем для классификаторов с несколькими возможными классами объектов используется величина mAP (англ. mean AP) — среднее значение AP.

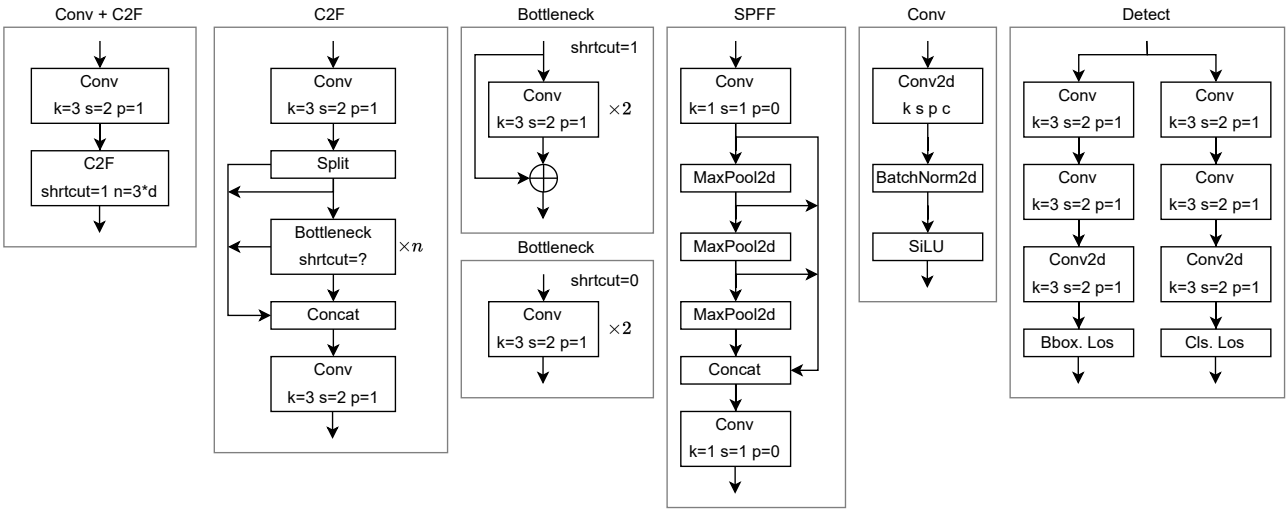

Рисунок 1.18 – Архитектура YOLOv8. Часть 2

Кроме того, данную величину принято оценивать в зависимости от IoU (Intersection over Union), вычисляемую по формуле:

$$IoU = \frac{\text{площадь пересечения областей}}{\text{площадь объединения областей}}. \tag{1.8}$$

IoU описывает то, насколько предсказанные CNN обрамляющие окна близки к «истинным». Данная величина принимает значения в диапазоне $[0; 1]$, соответственно.

1.5.4 Сравнение рассмотренных методов

YOLOv5x

На основании датасета MS COCO test-dev 2017, достигается значение AP равное 50.7% при размере входного изображения в 640 пикселей. При увеличении размера изображения до 1536 пикселей AP достигает 55.8% [15].

При этом, производительность CNN составляет вплоть до 200 FPS при использовании NVIDIA V100 [22].

YOLOv8x

Заявляется, что при использовании того же датасета YOLOv8x достигает AP равного 53.9% при размере входного изображения в 640 пикселей (в сравнении с 50.7% для YOLOv5x) [15].

Производительность данной CNN достигает 280 FPS при использовании NVIDIA A100 [23] и TensorRT [24].

Сравнение методов

Приведенные в таблице результаты получены при размере входного изображения равном 640×640 пикселей. Измерения проводились на основании датасета COCO [25] с использованием GPU NVIDIA GeForce RTX 4090 [26].

Таблица 1.1 – Сравнение рассмотренных методов

CNN	mAP_{IoU}		Параметры, млн. шт.	FLOPs, млрд.	FPS
	$mAP_{0.5}$	$mAP_{0.5:0.95}$			
Faster R-CNN	62.5	—	53	888	< 20
YOLOv5n	45.7	28.0	1.9	4.5	934
YOLOv5x	50.7	68.9	86.7	205.7	252
YOLOv8n	37.3	50.4	3.2	8.7	1163
YOLOv8x	53.9	—	68.2	257.8	236

1.6 Метод распознавания надводных объектов с аэрофотоснимков с использованием нейронных сетей

Распознавание надводного объекта на основе единственного изображения возможно с использованием CNN, однако при этом надводный объект должен быть различим, что накладывает ограничения на условия работы и разрешение входных данных.

В связи с тем, что размеры надводных объектов сильно варьируются (например, размеры морского буя и сухогрузного судна) и могут быть крайне малы при съемке с большого расстояния, метод, разрабатываемый в рамках данной работы должен иметь возможность распознавания лишь относительно крупных объектов. Такие ограничения связаны, в первую очередь, с зашумленностью получаемых снимков из-за погодных условий и контекста изображений.

Кроме того, аэрофотосъемку можно грубо разделить на две категории: съемка с большой высоты (например, с использованием искусственных спутников) и съемка с малой высоты, например, с использованием БПЛА, с различных ракурсов.

Таким образом разрабатываемый метод должен решать две описанные выше задачи — распознавать надводные объекты снятые с различных высот и ракурсов.

1.7 Формализованная постановка задачи

Цель работы — разработка метода распознавания надводных объектов с аэрофотоснимков. Для достижения этой цели необходимо решить следующие задачи:

— разработать соответствующий метод;

— реализовать разработанный метод;

— оценить результаты работы метода в зависимости от различных параметров системы.

Входными данными метода является изображение. Результатом работы являются объекты распознавания.

На метод накладываются следующие ограничения:

— входные данные — фотоснимок в формате PNG, JPG, JPEG;

— размерность входного изображения — не ниже 640×640 и не более 1280×1280 пикселей;

— разрешение входного изображения — от 0.5 до 15 метров/пиксель;

— работа только в дневное время суток (так как метод, разработанный для «ночной» работы может обладать совершенно другими свойствами);

— распознавание только различимых объектов на изображении.

На рисунке 1.19 представлена диаграмма, описывающая общий вид метода распознавания надводных объектов с использованием нейронных сетей.

1.8 Выбор данных для обучения модели

1.8.1 Разрешение изображений

В настоящее время существующие методы можно разделить на две группы по разрешению используемых в процессе обучения и работы снимков:

— VHR — достигает разрешения в 0.5 метра и менее на 1 пиксель;

— MR — достигает разрешения в несколько метров на 1 пиксель.

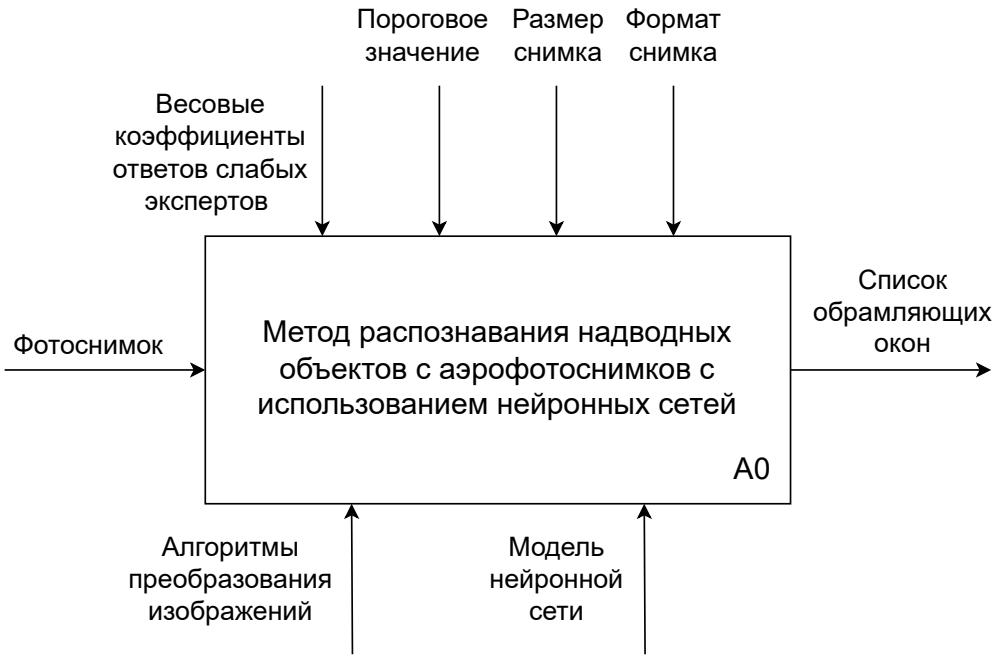

Рисунок 1.19 – Общий вид метода

Одна из проблем методов, основанных на VHR — сбор данных. В настоящее время изображения с таким разрешением могут быть получены лишь в нескольких источниках, использующих спутниковую съемку.

В тоже время, MR изображения могут быть получены, помимо прочего, с помощью съемки с БПЛА, что может значительно удешевить процесс сбора данных в сравнении с VHR.

Тем не менее, выборки снимков надводных объектов, находящиеся в открытом доступе, преимущественно состоят из VHR снимков, что объясняется сложностью проведения съемок посредством БПЛА, связанных с ограничениями в ряде регионов и стран.

1.8.2 Требования к данным

Для того, чтобы сделать вывод о корректности результата обучения модели, данные, использующиеся в процессе обучения должны удовлетворять нескольким требованиям:

— полнота — в выборке должны быть представлены различные расположения надводных объектов. Например судно расположенное в порту и в открытом море;

— количество — снимков в выборке должно быть достаточно для того,

чтобы модель на одном и том же изображении не давала разных результатов;

— единообразие — снимки в выборке должны быть схожего формата для исключения неоднозначных результатов.

1.9 Вывод

2 Конструкторский раздел

2.1 Требования

К разрабатываемому методу

Метод распознавания надводных объектов должен:

— принимать на вход изображения в форматах PNG, JPG, JPEG, BMP с размером от 640×640 до 1280×1280 включительно;

— производить распознавание различимых надводных объектов в дневное время суток.

К разрабатываемому программному комплексу

Программный комплекс, реализующий интерфейс для разработанного метода, должен предоставлять:

— возможность загрузки изображений через графический интерфейс;

— возможность создания итогового изображения с обрамляющими окнами обнаруженных объектов.

2.2 Проектирование метода распознавания

2.2.1 Выбор семейства/метода обнаружения

На основании данных из таблицы 1.1, в качестве метода обнаружения выбрана CNN YOLOv8n.

Данный выбор обусловлен более высокой точностью метода в сравнении с YOLOv5n. Кроме того, YOLOv8n обладает более низкими требованиями как к размеру обучающей выборки, так и к аппаратному обеспечению в сравнении как с YOLOv5x, так и с YOLOv8x.

В дополнение к выше сказанному, стоит отметить, что все масштабные версии YOLOv8 представлены в виде общедоступных пакетов для языка программирования Python 3 [27], что существенно упрощает использование данного метода.

2.2.2 IDEF0–диаграмма

На рисунке 2.1 представлена IDEF0 диаграмма метода уровня A1.

Рисунок 2.1 – IDEF0-диаграмма. Уровень A1

2.2.3 Объединение результатов работы слабых экспертов

Результатом работы слабых экспертов является множество обрамляющих окон, а так же соответствующие меры достоверности результатов.

В общем случае, обрамляющие окна, полученные в результате работы слабых экспертов могут пересекаться или же полностью дублироваться. Кроме того, в разрабатываемом методе предполагается учитывание весовых коэффициентов соответствующих слабых экспертов для корректировки результатов работы в случае, когда контекст снимка заранее известен.

Соответствующая схема алгоритма представлена на рисунке 2.2.

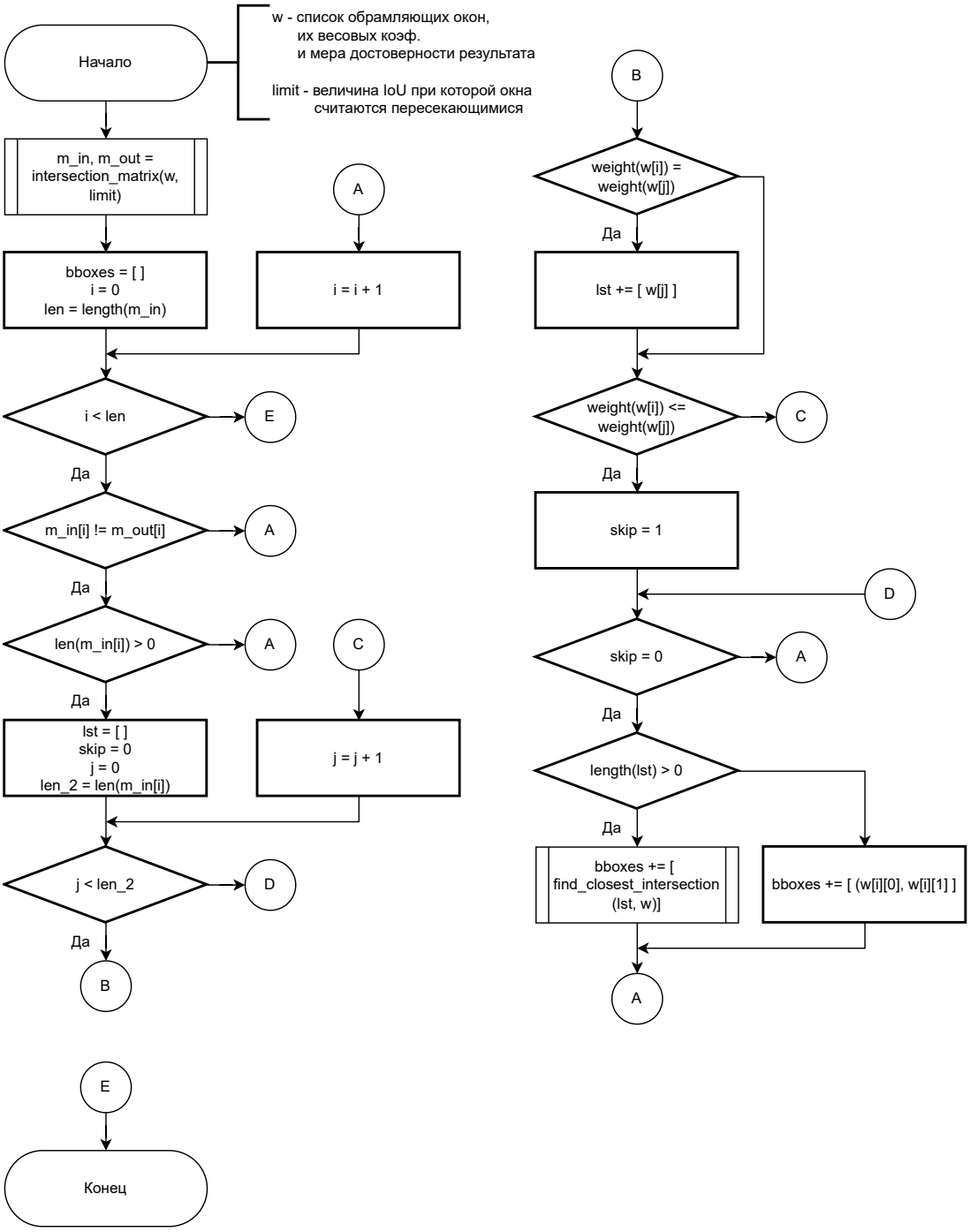

Рисунок 2.2 – Схема алгоритма объединения результатов работы слабых экспертов

Для того, чтобы обнаружить пересечение обрамляющих окон, необходимо установить порог значения IoU, при превышении которого окна считаются пересекающимися. Такое решение обусловлено контекстом решаемой задачи — при съемке с большого расстояния нгадводные объекты могут частично накладываться, что должно быть учтено в разрабатываемом методе.

Схема алгоритма обнаружения пересечений обрамляющих окон представлена на рисунке 2.3

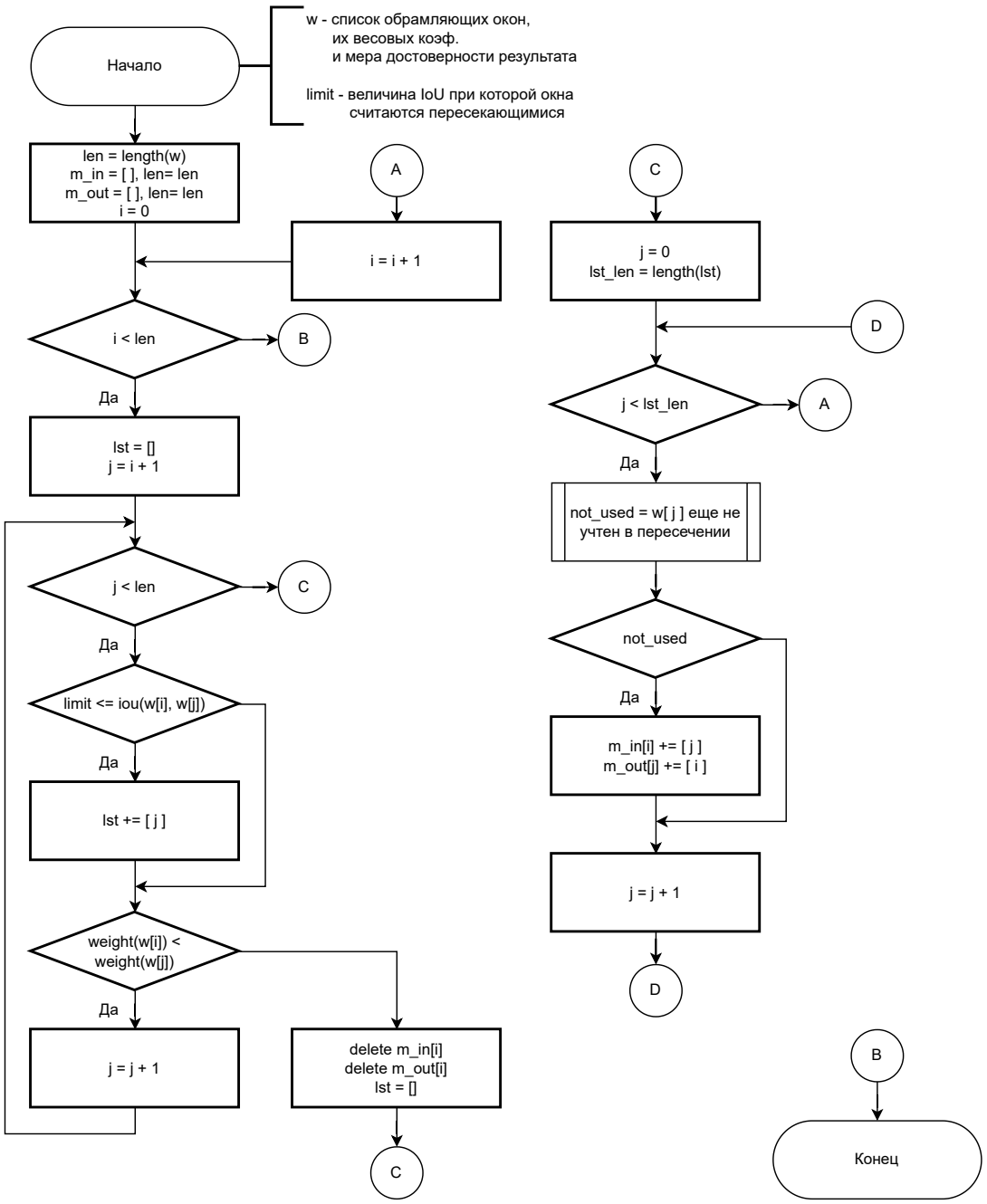

Рисунок 2.3 – Схема алгоритма обнаружения пересечений обрамляющих окон

В процессе обработки результатов работы слабых экспертов возможна ситуация, при которой пересекающиеся окна имеют одинаковый вес и, соответственно, становится невозможным выбор лишь одного из окон.

В качестве алгоритма разрешения подобных конфликтов предолагается использовать выбор наиболее близкого к центру масс пересечения какого-либо из подмножеств пересекающихся окон. При этом под наиболее близким понимаеется окно с наименьшим Евклидовым расстоянием между центром

окна и центром масс, соотвественно.

Соответствующая схема алгоритма представлена на рисунке 2.4

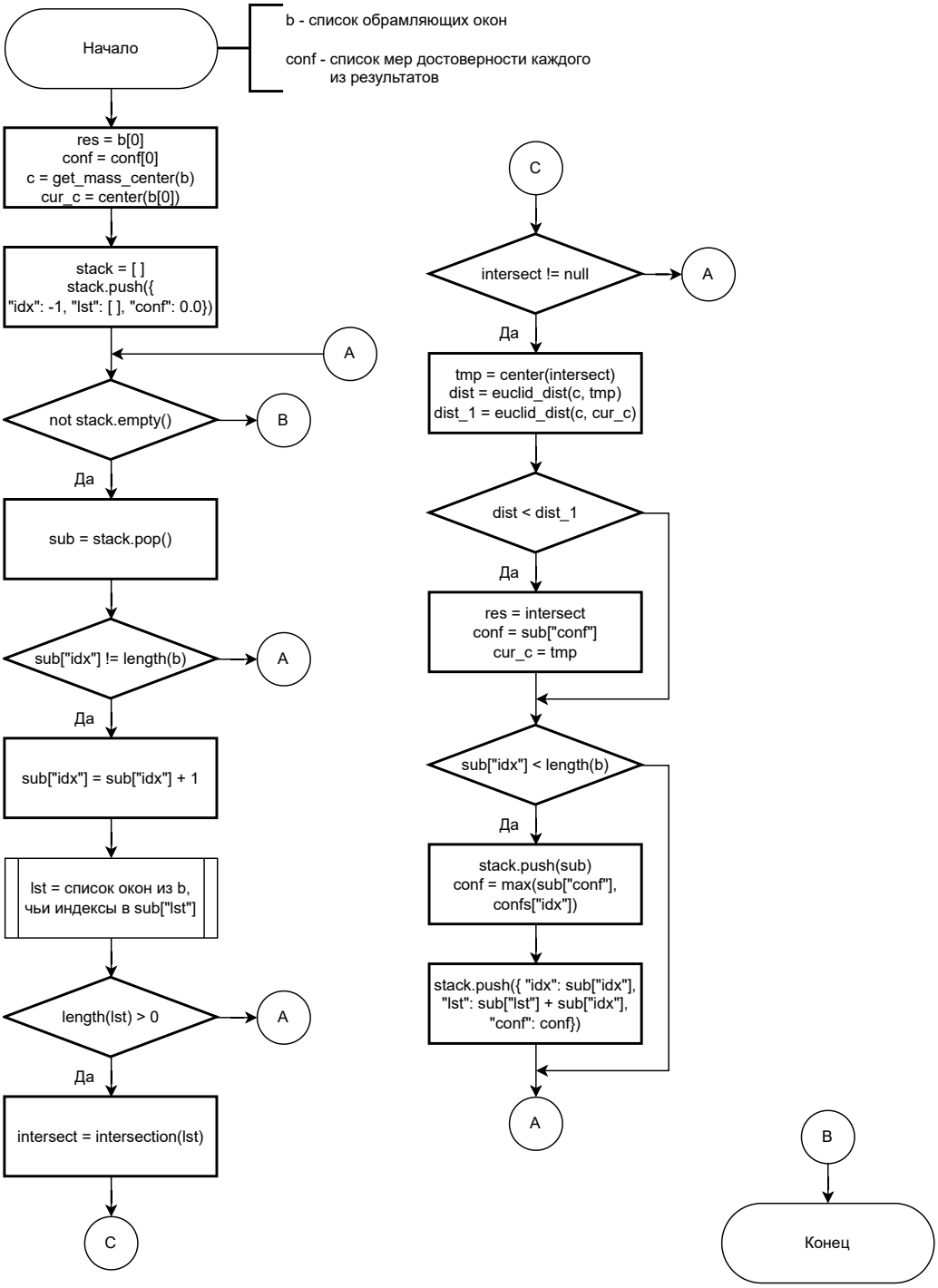

Рисунок 2.4 – Схема алгоритма пересечения обрамляющих окон

2.3 Структура разрабатываемого программного комплекса

Разрабатываемый программный комплекс состоит из двух модулей:

— модуль, реализующий метод распознавания объектов с использованием модели YOLOv8;

— пользовательское приложение, производящее распознавание объектов на основе разработанного метода, а так же предоставляющее возможность обучения модели, используемой в предыдущем модуле.

Модуль YOLOv8 модели

В данном модуле происходит только обучение и сохранение модели. Модуль должен быть использован либо при первоначальном обучении модели, либо при последующем дообучении в связи с вносящимися изменениями.

Результатом работы модуля является файл, содержит в себе обученную модель, т.е. ее структуру и весовые коэффициенты соответствующих связей. Это необходимо для того, чтобы не тратить время на обучение модели при повторном использовании, так как процесс обучения может занимать продолжительный период времени в зависисмотси от некоторых факторов.

На рисунке 2.5 представлена схема работы с модулем YOLOv8 модели.

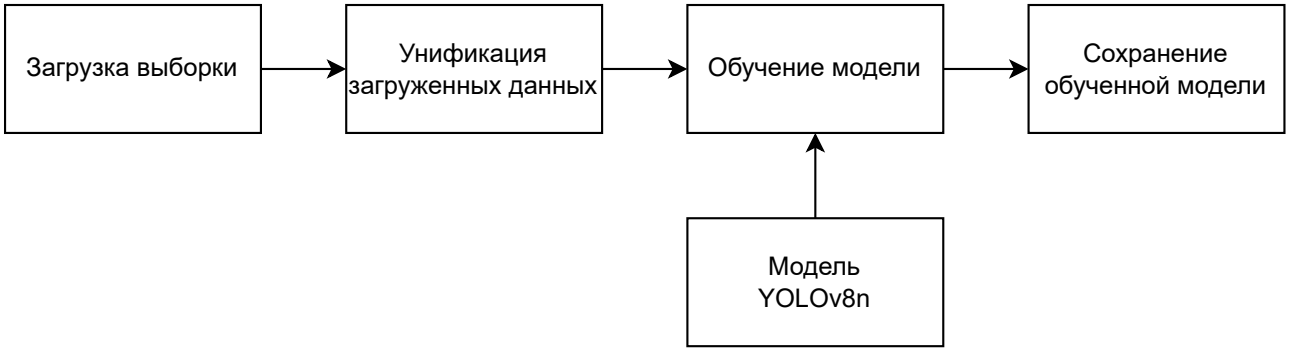

Рисунок 2.5 – Схема работы с модулем YOLOv8 модели

Модуль пользовательского приложения

Данный модуль позволяет пользователю загружать изображение для последующего анализа и выбора параметров настройки метода (весовых коэффициентов слабых экспертов). Результатом работы модуля является изображение с выделенными надводными объектами.

На рисунке 2.6 представлена схема работы с модулем пользовательского приложения.

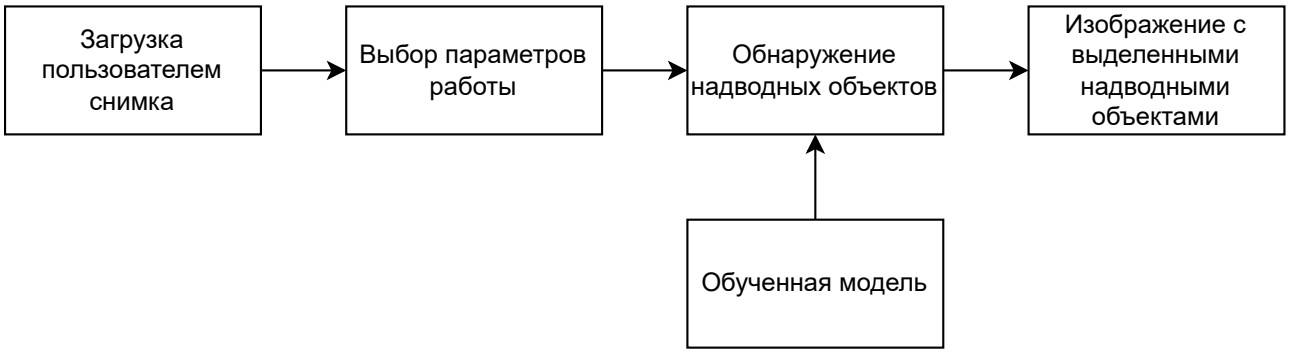

Рисунок 2.6 – Схема работы с модулем пользовательского приложения

2.4 Данные для обучения модели

В качестве данных для обучения моделей были выбраны три общедоступных набора данных: Kaggle Ships in Google Earth [28], ShipRSImageNet [29] и Huawei Ship [30].

Далее приводится описание каждого из использованных наборов данных.

Kaggle Ships In Google Earth

Данный набор данных содержит 1658 снимков, содержащих более двух тысяч надводных объектов в различных контекстах (например, в портовой зоне и в открытом море).

В виду малого числа изображений, набор даных будет разделятся в следующих пропорциях: 85:5:10 для обучающей, тестовой и валидационной частей, соответственно.

На рисунках 2.7 и 2.8 представлены примеры снимков из набора данных.

Рисунок 2.7 – Пример снимка из датасета (портовая зона)

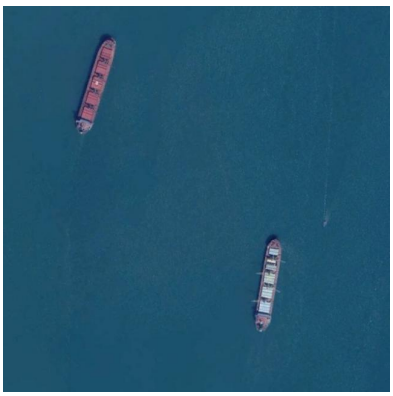

Рисунок 2.8 – Пример снимка из датасета (открытое море)

ShipRSImageNet

В связи с относительно малым числом снимков в описанном выше наборе данных, в дополнение к последнему, будет использован еще один, специально собранный для обучения нейронных сетей в задачах распознавания.

ShipRSImageNet состоит из 3435 снимков, содержащих в общей сложности более 17.5 тысяч объектов, размеченных как с использованием как горизонтальных, так и ориентированныхx обрамляющих окон.

Кроме того, экспертами была проведена разметка объектов по их принадлежности к одному из 50 классов кораблей и судов, что может быть использовано в будущем в рамках одного из направлений развития разрабатываемого метода.

Примеры снимков из этого набора данных приведены на рисунках 2.9 и 2.10.

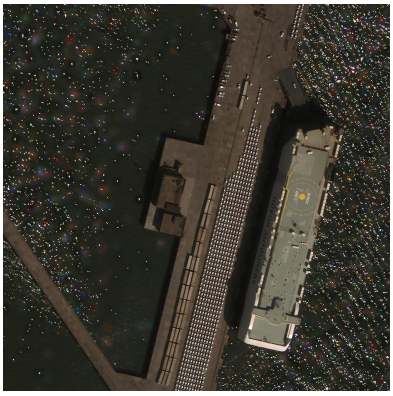

Рисунок 2.9 – Пример снимка из датасета (портовая зона)

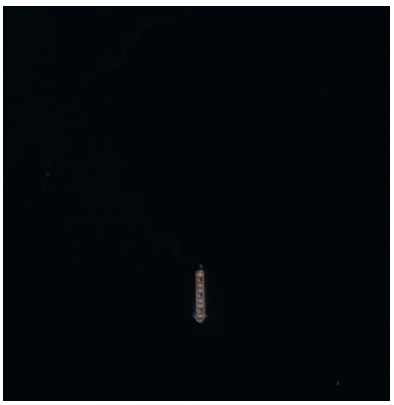

Рисунок 2.10 – Пример снимка из датасета (открытое море)

Huawei Ship

Состоит из 5538 снимков различных надводных объектов (преимущественно судов и кораблей) с различных ракурсов и расстояний, а так же в разных погодных условиях. В общей сложности набор содержит 8132 размеченных надводных объекта, относящихся к одному из восьми представленных классов.

В связи с особенностями работы сверточных нейронных сетей, данный набор данных необъодимо обогатить, чтобы повысить точность работы метода обнаружения. Данная необходимость вызвана большим числом обрабатываемых признаков объектов, в сравнении с предыдущими наборами данных.

Примеры снимков из этого набора данных приведены на рисунках 2.11, 2.12 и 2.13.

Рисунок 2.11 – Пример снимка из датасета (корабль)

Рисунок 2.12 – Пример снимка из датасета (судно в тумане)

Рисунок 2.13 – Пример снимка из датасета (буй)

2.5 Обогащение данных

В связи с малым числом изображений в обучающей выборке Huawei Ship, для каждого из представленных снимков по отдельности применены следующие трансформации (с различными параметрами):

— горизонтальное отражение;

— поворот (до 15 градусов);

— усредняющее размытие;

— Гауссово размытие;

— выравнивание гистограммы;

— исключение (до 20%);

— Гауссов шум;

— повышение цветового тона;

— изменение резкости.

2.6 Формат хранения разметки данных

Формат хранения данных обусловлен соглашением о формате в семействе YOLO. Данные описываются YAML–файлом, содержащим всю информацию о выборке. Пример приведен в листинге 2.1

Листинг 2.1 – Описание выборки в формате YOLO

```
train: ../train/images
val: ../valid/images
test: ../test/images

nc: 1
names: ['boat']
```

В данном примере описываются три подвыборки (тренировочная, тестовая и валидационная), а так же классы объектов, представленных в выборке, использующиеся при решении задачи классификации объектов: общее число классов, а так же соответствующие названия, используемые при разметке обработанных изображений.

Каждая подвыборка содержит две поддиректории: изображения и аннотации в формате YOLO. Пример аннотации представлен в листинге 2.2.

Листинг 2.2 – Пример аннотации в формате YOLO

```
0 0.83203125 0.33359375 0.03046875 0.153125
```

В данном формате аннотации содержат следующую информацию: порядковый номер класса объекта, нормализованные координаты левого верхнего угла обрамляющего окна, нормализованные ширину и высоту обрамляющего окна. Такой формат позволяет работать с изображениями даже при масштабировании последних.

2.7 Обучение и тестирование слабых экспертов

Так как в данной методе используется бэггинг, то каждый из слабых экспертов будет обучаться независимо от остальных. В связи с этим, выборка должна быть разделена на n непересекающихся выборок, где n — число слабых экспертов.

В общем случае для обучения и тестирования слабых экспертов полученные после разделения выборки примерно разделяются в соотношении

80 : 15 : 5 на обучающую, тестовую и валидационную подвыборки, соответственно.

С учетом приведенного выше описания наборов данных, в методе распознавания будет использовано 3 слабых эксперта, обучаемых на соответствующем наборе данных. Тестирование каждого из обученных слабых экспертов будет производится на основании данных, представленных в соотвтествующем наборе данных, в связи со спецификой каждого из последних.

2.8 Вывод

3 Технологический раздел

3.1 Средства реализации

Выбор языка программирования

Для реализации программного комплекса будет использоваться язык программирования Python 3 [27]. Данный выбор обусловлен следующими факторами:

— широкий набор библиотек для работы с нейронными сетями;

— возможность обучать нейронную сеть на графическом процессоре с использованием технологии CUDA [31].

Кроме того, как указывалось ранее, для выбранного языка программирования уже существуют общедоступные пакеты для работы с выбранным методом обнаружения, включая методы для обучения, тестирования и валидации работы метода.

Выбор библиотеки глубокого обучения

Для создания и обучения модели нейронной сети была выбрана библиотека PyTorch [20] версии 2.0.0. Выбор данной версии обусловлен поддержкой CUDA 11.8, предоставляемой GPU NVIDIA GeForce RTX 2060 [32], на котором будет производиться обучение нейронной сети.

3.2 Реализация программного комплекса

3.2.1 Модуль метода распознавания

Реализация модуля построена на YOLOv8n модели. Модуль состоит из трех слабых экспертов, обученных с использованием различных наборов данных, как было описано ранее.

В связи с тем, что данные слабые эксперты не зависят от результатов работы друг друга, присутствует возможность распараллеливания процесса обработки изображения слабыми экспертами с целью снижения времени отклика системы.

Для организации параллельной обработки будет использован Ray — фреймворк с открытым исходным кодом, предоставляющий возможность

параллельного выполнения, а так же кластеризации различного рода вычислений, в том числе при использовании нейронных сетей [33].

В дополнение к вышесказанному, Ray предоставляет возможность управления ресурсами, использующимися в процессе вычислений в кластере, а так же отслеживает доступные на каждом из узлов кластера ресурсы, с целью оптимищзации процесса вычислений.

В листингах А.1 и А.2 приведена реализация модуля модели, отвечающего за обнаружение надводных объектов.

3.2.2 Обучение слабых экспертов

В предоставляемом Ultralytics пакете для работы с YOLOv8 предусмотрены методы для обучения моделей с различными параметрами и возможностью применения таких оптимизаций, как батчинг изображений, мозаичная аугментация на ранних стадиях обучения, отслеживание результатов процесса обучения с прерыванием в случае простоя и прочие [34].

На рисунках 3.1 и 3.2 представлены примеры использования описанных выше оптимизаций в процессе обучения.

Рисунок 3.1 – Батч обучающей выборки с применением мозаичной аугментации. Часть 1

Помимо этого, указанный выше пакет предоставляет средства для автоматической предобработки изображения (его масштабирования) при превышении размера, использованного при обучении модели. В совокупности с использованием нормализованных координат при формаировании результатов

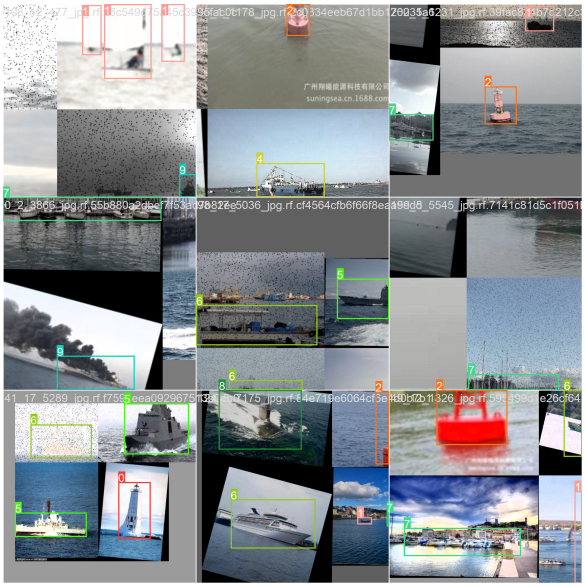

Рисунок 3.2 – Батч обучающей выборки с применением мозаичной аугментации. Часть 2

распознавания, это позволяет полностью переложить процесс предобработки на описанный выше пакет.

На рисунке 3.3 представлен пример использования описанных выше оптимизаций в процессе валидации результатов работы обученной модели.

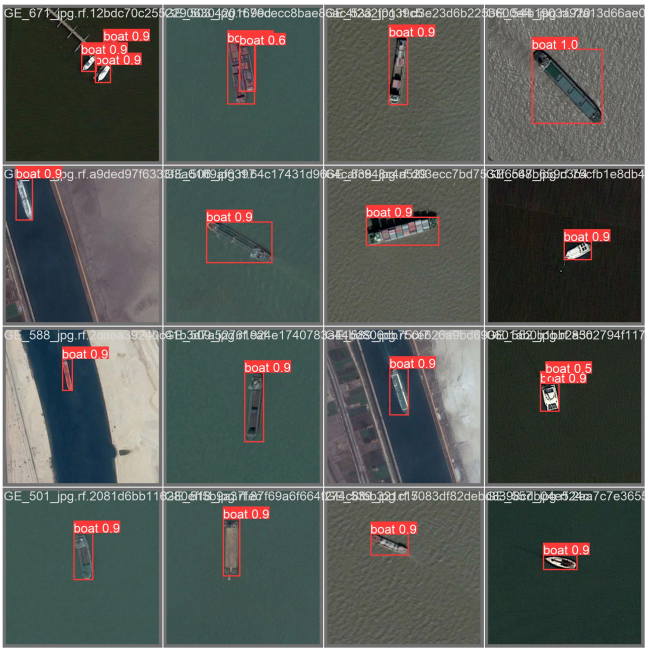

Рисунок 3.3 – Батч валидационной выборки

3.3 Результаты обучения слабых экспертов Kaggle Ships In Google Earth

Процесс обучения состоял из 100 эпох, при этому можно отметить, что точность распознавания возрастала на протяжении всего процесса обучения и достигла 92 %. Отдельно стоит отметить, что начиная с 30 эпохи точности составляла более 80 %.

Листинг 3.1 – Результат обучения слабого эксперта

```
100 epochs completed in 0.465 hours.
Optimizer stripped from runs/detect/train/weights/last.pt, 6.2MB
Optimizer stripped from runs/detect/train/weights/best.pt, 6.2MB
Validating runs/detect/train/weights/best.pt...
Ultralytics YOLOv8.0.90  Python-3.10.11 torch-2.0.0 CUDA:0
   (NVIDIA GeForce RTX 2060, 6144MiB)
Model summary (fused): 168 layers, 3005843 parameters, 0
   gradients, 8.1 GFLOPs
Class   Images   Instances   Box(P      R    mAP50   mAP50-95):
   100%|----------| 9/9 [00:03<00:00,  2.97it/s]
  all      159         357   0.914   0.894   0.946       0.692
Speed: 1.2ms preprocess, 4.3ms inference, 0.0ms loss, 1.8ms
   postprocess per image
Results saved to runs/detect/train
```

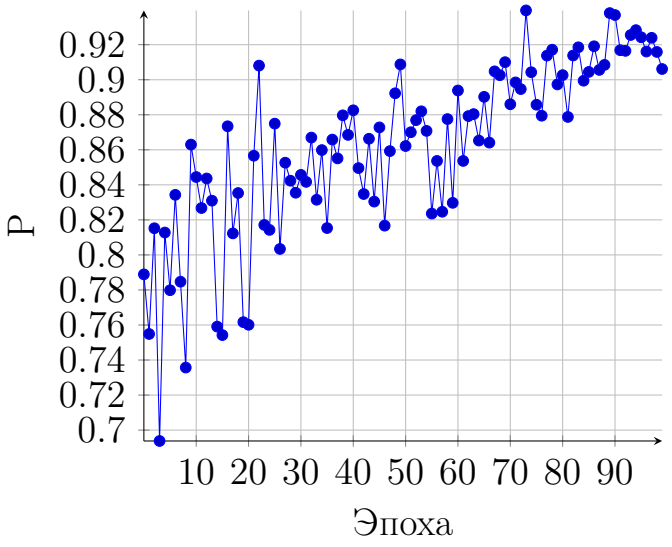

Рисунок 3.4 – Точность слабых экспертов (P)

ShipRSImageNet

Аналогично предыдущему слабому эксперту, обучение производилось в течение 100 эпох. В процессе обучения точность достигла 85 %. При этом, точность в 60 % была достигнута лишь к 40 эпохе.

Различия в кривых обучения в сравнении с предыдущим набором данных можно объяснить различиями в точности разметки данных в различных наборах.

Листинг 3.2 – Результат обучения слабого эксперта

```
100 epochs completed in 0.674 hours.
Optimizer stripped from runs/detect/train2/weights/last.pt, 6.3MB
Optimizer stripped from runs/detect/train2/weights/best.pt, 6.3MB
Validating runs/detect/train2/weights/best.pt...
Ultralytics YOLOv8.0.92  Python-3.10.11 torch-2.0.0 CUDA:0
   (NVIDIA GeForce RTX 2060, 6144MiB)
Model summary (fused): 168 layers, 3015398 parameters, 0
   gradients, 8.1 GFLOPs
Class   Images   Instances   Box(P       R   mAP50   mAP50-95):
   100%|----------| 16/16 [00:04<00:00,  3.46it/s]
  all      278        1240   0.818   0.784   0.865       0.765
Speed: 0.9ms preprocess, 3.4ms inference, 0.0ms loss, 1.8ms
   postprocess per image
Results saved to runs/detect/train2
```

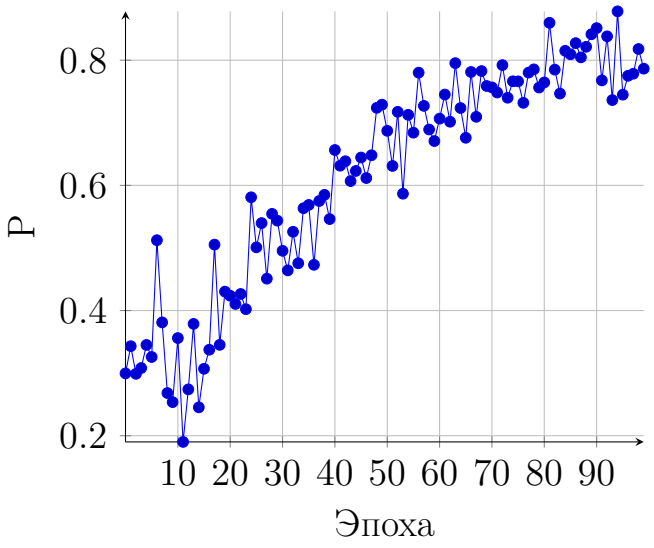

Рисунок 3.5 – Точность слабых экспертов (P)

Huawei Ship

В процессе обучения с использованием данного набора данных, модель завершила обучение на 15ой эпохе. Данный факт был установлен экспериментально — в течение 50-ти эпох после этого не было отмечено значительных изменений в точности обнаружения обхъектов.

По итогам обучения, точности достигла 83 %, при этом уже с первой эпохи точность работы метода составляла не менее 75 %.

Листинг 3.3 – Результат обучения слабого эксперта

```
15 epochs completed in 5.457 hours.
Optimizer stripped from runs/detect/train3/weights/last.pt, 6.2MB
Optimizer stripped from runs/detect/train3/weights/best.pt, 6.2MB
Validating runs/detect/train3/weights/best.pt...
Ultralytics YOLOv8.0.92  Python-3.10.11 torch-2.0.0 CUDA:0
   (NVIDIA GeForce RTX 2060, 6144MiB)
Model summary (fused): 168 layers, 3007598 parameters, 0
   gradients, 8.1 GFLOPs
Class  Images  Instances  Box(P      R  mAP50  mAP50-95):
   100%|----------| 348/348 [01:11<00:00,  4.86it/s]
  all    6256      11638  0.831  0.827  0.845      0.584
Speed: 0.2ms preprocess, 2.9ms inference, 0.0ms loss, 1.7ms
   postprocess per image
Results saved to runs/detect/train3
```

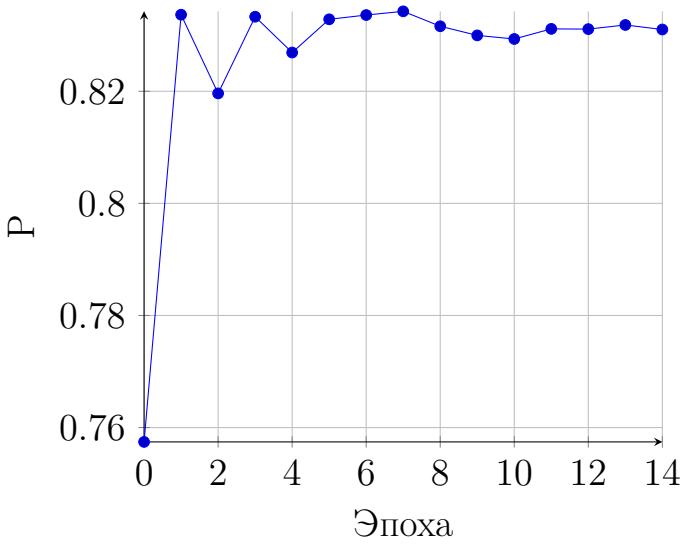

Рисунок 3.6 – Точность слабых экспертов (P)

3.4 Примеры использования разработанного программного комплекса

Модуль, реализующий метод распознавания объектов с использованием модели YOLOv8 выполнен в виде пакета на языке Python 3 и не предоставляет пользовательского интерфейса для работы с ним.

В то же время, модуль пользовательского приложения представляет собой консольное приложение без графического интерфейса, конфигурируемые при помощи аргументов командной строки.

3.4.1 Пример обучения модели

Листинг 3.4 – Взаимодействие с пользовательским приложением обучения модели

```
$ python3 src/train.py --help
usage: train.py [-h] [-d DATA] [-e EPOCHS]

options:
  -h, --help            show this help message and exit
  -d DATA, --data DATA  Пусть к набору данных
  -e EPOCHS, --epochs EPOCHS
                        Число эпох для обучения
```

Листинг 3.5 – Запуск обучения модели

```
$ python3 src/train.py -d dataset
New https://pypi.org/project/ultralytics/8.0.104 available
   Update with 'pip install -U ultralytics'
Ultralytics YOLOv8.0.99  Python-3.10.4 torch-2.0.1 CPU
yolo/engine/trainer: task=detect, mode=train, model=yolov8n.pt,
   data=dataset/data.yaml, epochs=100, patience=50, batch=9,
   imgsz=640, save=True, save_period=-1, cache=ram, device=None,
   workers=8, project=None, name=None, exist_ok=False,
   pretrained=False, optimizer=SGD, verbose=True, seed=0,
   deterministic=False, single_cls=False, rect=False,
   cos_lr=False, close_mosaic=0, resume=False, amp=True,
   overlap_mask=True, mask_ratio=4, dropout=0.0, val=True,
   split=val, save_json=False, save_hybrid=False, conf=None,
   iou=0.7, max_det=300, half=False, dnn=False, plots=True,
   source=None, show=False, save_txt=False, save_conf=False,
   save_crop=False, show_labels=True, show_conf=True,
```

```
   vid_stride=1, line_width=None, visualize=False,
   augment=False, agnostic_nms=False, classes=None,
   retina_masks=False, boxes=True, format=torchscript,
   keras=False, optimize=False, int8=False, dynamic=False,
   simplify=False, opset=None, workspace=4, nms=False, lr0=0.01,
   lrf=0.01, momentum=0.937, weight_decay=0.0005,
   warmup_epochs=3.0, warmup_momentum=0.8, warmup_bias_lr=0.1,
   box=7.5, cls=0.5, dfl=1.5, pose=12.0, kobj=1.0,
   label_smoothing=0.0, nbs=64, hsv_h=0.015, hsv_s=0.7,
   hsv_v=0.4, degrees=0.0, translate=0.1, scale=0.5, shear=0.0,
   perspective=0.0, flipud=0.0, fliplr=0.5, mosaic=1.0,
   mixup=0.0, copy_paste=0.0, cfg=None, v5loader=False,
   tracker=botsort.yaml, save_dir=runs/detect/train
Overriding model.yaml nc=80 with nc=1
...
Model summary: 225 layers, 3011043 parameters, 3011027
   gradients, 8.2 GFLOPs

Transferred 319/355 items from pretrained weights
optimizer: SGD(lr=0.01) with parameter groups 57
   weight(decay=0.0), 64 weight(decay=0.0004921875), 63 bias
train: Scanning datasets/dataset/train/labels.cache... 1420
   images, 0 backgrounds, 0 corrupt
train: Caching images (1.6GB ram): 100%|----------| 1420/1420
   [00:01<00:00, 1204.14it/s]
val: Scanning datasets/dataset/valid/labels.cache... 159 images,
   0 backgrounds, 0 corrupt: 1
val: Caching images (0.2GB ram): 100%|----------| 159/159
   [00:00<00:00, 1428.06it/s]
Plotting labels to runs/detect/train/labels.jpg...
Image sizes 640 train, 640 val
Using 0 dataloader workers
Logging results to runs/detect/train
Starting training for 100 epochs...
```

3.4.2 Пример распознавания объектов

Листинг 3.6 – Взаимодействие с пользовательским приложением распознавания объектов

```
$ python3 cli.py --help
usage: cli.py [-h] [-m MODEL] [-i IMAGE] [-o OUTPUT]
```

```
options:
  -h, --help            show this help message and exit
  -m MODEL, --model MODEL
                        Путь к файлу/директории с обученной
                           моделью
  -i IMAGE, --image IMAGE
                        Пусть к файлу снимка
  -o OUTPUT, --output OUTPUT
                        Директория для сохранения результатов
```

Листинг 3.7 – Запуск приложения

```
$ python3 cli.py -m ./experts -i ./multiple.jpg
2023-05-17 15:29:53,529 INFO worker.py:1625 -- Started a local
   Ray instance.
```

3.5 Вывод

4 Исследовательский раздел

4.1 Аппаратное обеспечение

4.2 Размер входного изображения

4.3 Зернистость входного изображения

4.4 Зашумленность входного изображения

4.5 Смазанность входного изображения

4.6 Вывод

ЗАКЛЮЧЕНИЕ

Таким образом, поставленная цель работы — разработать метод распознавания надводных объектов с аэрофотоснимков, была достигнута.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Lee S.-J.*, *Roh M.-I.*, *Oh M.-j.* Image-based ship detection using deep learning // Ocean Systems Engineering. — 2020. — Т. 10. — С. 415—434. — DOI: 10.12989/ose.2020.10.4.415.
2. *Z. Chen* Deep learning for autonomous ship-oriented small ship detection / Z. Chen [и др.] // Safety Science. — 2020. — Т. 130. — С. 104812.
3. *M. Marzuki* Fishing boat detection using Sentinel-1 validated with VIIRS Data / M. Marzuki [и др.] // IOP Conference Series: Earth and Environmental Science. — 2021. — Т. 925. — С. 012058. — DOI: 10.1088/1755-1315/925/1/012058.
4. *Sharkawy A.-.-N.* Principle of Neural Network and Its Main Types: Review // Journal of Advances in Applied & Computational Mathematics. — 2020. — Т. 7. — С. 8—19. — DOI: 10.15377/2409-5761.2020.07.2.
5. *Sharma S.*, *Sharma S.*, *Athaiya A.* ACTIVATION FUNCTIONS IN NEURAL NETWORKS // *S. Sharma* International Journal of Engineering Applied Sciences and Technology. — 2020. — Т. 4. — С. 310—316. — DOI: 10.33564/IJEAST.2020.v04i12.054.
6. *A. Ghosh* Fundamental Concepts of Convolutional Neural Network / A. Ghosh [и др.]. — 2020. — DOI: 10.1007/978-3-030-32644-9_36.
7. *Zhang H.*, *Cloutier R.* Review on One-Stage Object Detection Based on Deep Learning // EAI Endorsed Transactions on e-Learning. — 2022. — Т. 7. — С. 174—181. — DOI: 10.4108/eai.9-6-2022.174181.
8. *Du L.*, *Zhang R.*, *Wang X.* Overview of two-stage object detection algorithms // Journal of Physics: Conference Series. — 2020. — Т. 1544. — С. 012033. — DOI: 10.1088/1742-6596/1544/1/012033.
9. *А. В. Бондаренко* АЛГОРИТМ НЕЙРОСЕТЕВОГО РАСПОЗНАВАНИЯ НАДВОДНЫХ ОБЪЕКТОВ В РЕАЛЬНОМ ВРЕМЕНИ / А. В. Бондаренко [и др.] // Известия Тульского государственного университета. Технические науки. — 2021. — Т. 1. — С. 19—33.
10. *M. Ganaie* Ensemble deep learning: A review / M. Ganaie [и др.]. — 2021. — DOI: 10.48550/arXiv.2104.02395.

11. *R. Girshick* Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation / R. Girshick [и др.] // Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. — 2013. — DOI: 10.1109/CVPR.2014.81.

12. *Girshick R.* Fast R-CNN. — 2015. — DOI: 10.1109/ICCV.2015.169.

13. *S. Ren* Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks / S. Ren [и др.] // IEEE Transactions on Pattern Analysis and Machine Intelligence. — 2015. — Т. 39. — DOI: 10.1109/TPAMI.2016.2577031.

14. *P. Jiang* A Review of Yolo Algorithm Developments / P. Jiang [и др.] // Procedia Computer Science. — 2022. — Т. 199. — С. 1066—1073. — ISSN 1877-0509. — DOI: https://doi.org/10.1016/j.procs.2022.01.135.

15. *Terven J.*, *Cordova-Esparza D.-M.* A Comprehensive Review of YOLO: From YOLOv1 to YOLOv8 and Beyond. — 2023. — Апр.

16. *L. Hao* Enhanced YOLO v3 Tiny Network for Real-time Ship Detection from Visual Image / L. Hao [и др.] // IEEE Access. — 2021. — Т. PP. — С. 1—1. — DOI: 10.1109/ACCESS.2021.3053956.

17. *Redmon J.*, *Farhadi A.* YOLOv3: An Incremental Improvement. — 2018.

18. *D. Li* Yolo-tiny-MSC: A tiny neural network for object detection / D. Li [и др.] // Journal of Physics: Conference Series. — 2021. — Т. 1873. — С. 012073. — DOI: 10.1088/1742-6596/1873/1/012073.

19. *Redmon J.*, *Farhadi A.* YOLO9000: Better, Faster, Stronger // 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). — 2017. — С. 6517—6525. — DOI: 10.1109/CVPR.2017.690.

20. PyTorch [Электронный ресурс]. — Режим доступа: https://pytorch.org/ (Дата обращения: 22.04.2023).

21. Darknet [Электронный ресурс]. — Режим доступа: https://github.com/pjreddie/darknet (Дата обращения: 22.04.2023).

22. NVIDIA V100 TENSOR CORE GPU [Электронный ресурс]. — Режим доступа: https://www.nvidia.com/en-us/data-center/v100/ (Дата обращения: 22.04.2023).

23. NVIDIA A100 Tensor Core GPU [Электронный ресурс]. — Режим доступа: https://www.nvidia.com/en-us/data-center/a100/ (Дата обращения: 22.04.2023).

24. NVIDIA TensorRT [Электронный ресурс]. — Режим доступа: https://developer.nvidia.com/tensorrt (Дата обращения: 22.04.2023).

25. COCO - Common Objects in Context [Электронный ресурс]. — Режим доступа: https://cocodataset.org/ (Дата обращения: 20.11.2022).

26. GeForce RTX 4090 [Электронный ресурс]. — Режим доступа: https://www.nvidia.com/en-us/geforce/graphics-cards/40-series/rtx-4090/ (Дата обращения: 23.04.2023).

27. Welcome to Python.org [Электронный ресурс]. — Режим доступа: https://www.python.org/ (Дата обращения: 23.04.2023).

28. kaggle-ships-in-google-earth Dataset [Электронный ресурс]. — Режим доступа: https://universe.roboflow.com/robin-public/kaggle-ships-in-google-earth-dfqwt (Дата обращения: 19.04.2023).

29. *Z. Zhang* ShipRSImageNet: A Large-scale Fine-Grained Dataset for Ship Detection in High-Resolution Optical Remote Sensing Images / Z. Zhang [и др.] // IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing. — 2021. — DOI: 10.1109/JSTARS.2021.3104230.

30. Huawei_ship Dataset [Электронный ресурс]. — Режим доступа: https://universe.roboflow.com/huaweiship/huawei_ship (Дата обращения: 07.05.2023).

31. What is CUDA | NVIDIA official blog [Электронный ресурс]. — Режим доступа: https://blogs.nvidia.com/blog/2012/09/10/what-is-cuda-2/ (Дата обращения: 23.04.2023).

32. Introducing The GeForce RTX 2060: Turing For Every Gamer [Электронный ресурс]. — Режим доступа: https://www.nvidia.com/en-us/geforce/news/gfecnt/nvidia-geforce-rtx-2060/ (Дата обращения: 23.04.2023).

33. Productionizing and scaling Python ML workloads simply | Ray [Электронный ресурс]. — Режим доступа: https://www.ray.io/ (Дата обращения: 16.05.2023).

34. ultralytics - PyPI [Электронный ресурс]. — Режим доступа: https://pypi.org/project/ultralytics/ (Дата обращения: 09.05.2023).

ПРИЛОЖЕНИЕ А
Модуль модели YOLOv8n

Листинг А.1 – Модуль YOLOv8 модели. Часть 1

```
import os

import cv2
import numpy as np
from ultralytics import YOLO
import ray

from method.deduplicate import deduplicate_wbboxes


@ray.remote(num_gpus=0.25)
def apply_expert(image: np.ndarray, expert: YOLO):
    r = expert.predict(image)
    return (
        r[0].boxes.xyxyn.detach().cpu().numpy(),
        r[0].boxes.conf.detach().cpu().numpy(),
    )


class Detect:
    experts: list[YOLO] = []
    weights: np.ndarray

    def __init__(self, path: str, ext: tuple[str, ...] =
       tuple(".pt")):
        if os.path.isfile(path):
            self.experts += [YOLO(path)]
            return

        fnames = [os.path.join(path, f) for f in
           os.listdir(path) if f.endswith(ext)]

        self.experts += [YOLO(f) for f in fnames if
           os.path.isfile(f)]
        self.weights = np.ones(len(self.experts))
```

Листинг А.2 – Модуль YOLOv8 модели. Часть 2

```
    def predict(self, input: cv2.Mat, w: np.ndarray = None) ->
       None | np.ndarray:
        w = w or self.weights

        if len(w) != len(self.experts):
            return None

        input_id = ray.put(input)
        ray_ids = [apply_expert.remote(input_id, model) for
           model in self.experts]
        r = ray.get(ray_ids)

        wbboxes = [
            [w[i], np.array(b), np.float32(c)]
            for i, p in enumerate(r)
            for b, c in zip(*p)
        ]

        return deduplicate_wbboxes(wbboxes)
```

Листинг А.3 – Модуль YOLOv8 модели. Часть 3

```
import collections

import numpy as np

from method.utils import (
    get_mass_center,
    get_intersection_of_n_bboxes,
    get_bbox_center,
    get_bbox_weight as weight,
    get_iou as iou,
)


def find_closest_intersection(
    bboxes: np.ndarray, confs: np.ndarray
) -> tuple[np.ndarray, np.float32]:
    res, conf = bboxes[0], confs[0]
    cur_c, c = get_bbox_center(res), get_mass_center(bboxes)
    stack = collections.deque()
```

Листинг А.4 – Модуль YOLOv8 модели. Часть 4

```
    stack.append({"idx": -1, "lst": [], "conf": conf})

    while stack:
        sub = stack.pop()

        if sub["idx"] == len(bboxes):
            continue

        sub["idx"] += 1

        lst = np.array([bboxes[i] for i in sub["lst"]])
        if len(lst) > 0:
            intersection = get_intersection_of_n_bboxes(lst)
            if intersection is None:
                continue

            tmp = get_bbox_center(intersection)
            if np.linalg.norm(c - tmp) < np.linalg.norm(c -
               cur_c):
                res, conf, cur_c = intersection, sub["conf"], tmp

        if sub["idx"] < len(bboxes):
            stack.append(sub)
            stack.append(
                {
                    "idx": sub["idx"],
                    "lst": sub["lst"] + [sub["idx"]],
                    "conf": max(sub["conf"], confs["idx"]),
                }
            )

    return res, conf


def get_wbboxes_intersection_matrix(wbboxes: list, limit:
   np.float32 = 0.75) -> dict:
    m = {i: [[], []] for i in range(len(wbboxes))}
    for i in range(len(wbboxes)):
        cur = wbboxes[i]
```

Листинг А.5 – Модуль YOLOv8 модели. Часть 5

```
        lst = []
        for j in range(i + 1, len(wbboxes)):
            if limit <= iou(cur[1], wbboxes[j][1]):
                lst += [j]

            if weight(cur) < weight(wbboxes[j]):
                m.pop(i, None)
                lst = []
                break

        for j in lst:
            if next((True for k in m[j][1] if j in m[k][0]),
               False):
                continue

            m[i], m[j] = [m[i][0] + [j], m[i][1]], [m[j][0],
               m[j][1] + [i]]

    return m


def deduplicate_wbboxes(wbboxes: list, limit: np.float32 = 0.75)
   -> np.ndarray:
    m = get_wbboxes_intersection_matrix(wbboxes, limit)

    bboxes = []
    for i in m.keys():
        cur = wbboxes[i]
        if len(m[i][0]) == 0 and len(m[i][1]) == 0:
            bboxes += [(cur[1], cur[2])]
            continue

        if len(m[i][0]) == 0:
            continue

        lst, skip = [], False
        for j in m[i][0]:
            if weight(cur) == weight(wbboxes[j]):
                lst += [j]
```

Листинг А.6 – Модуль YOLOv8 модели. Часть 6

```
            if weight(cur) < weight(wbboxes[j]):
                skip = False
                break

        if skip:
            continue

        if len(lst) == 0:
            bboxes += [(cur[1], cur[2])]
            continue

        lst = np.array([cur[1]] + [wbboxes[j][1] for j in lst])
        conf = np.array([cur[2]] + [wbboxes[j][2] for j in lst])
        bboxes += [(*find_closest_intersection(lst, conf),)]

    return bboxes
```

ПРИЛОЖЕНИЕ Б

Модуль пользовательского приложения

Листинг Б.1 – Модуль пользовательского приложения. Часть 1

```
import argparse
import os
from pathlib import Path
import time

import cv2
import ray

from method.detect import Detect
from utils import apply_bboxes


def process(
    path: str,
    input: str,
    output: str,
    line_width: int = None,
    color=(200, 128, 128),
    txt_color=(255, 255, 255),
):
    method = Detect(path)

    im = cv2.imread(input)

    bboxes = method.predict(im)

    im = apply_bboxes(im, bboxes, line_width, color, txt_color)

    Path(output).mkdir(parents=True, exist_ok=True)
    cv2.imwrite(os.path.join(output,
       f"detect-{int(time.time())}.jpeg"), im)
```

Листинг Б.2 – Модуль пользовательского приложения. Часть 2

```
if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "-m", "--model", default="./experts/", help="Путь к
           файлу/директории с обученной моделью"
    )
    parser.add_argument(
        "-i", "--image", default="data/file.jpeg", help="Пусть к
           файлу снимка"
    )
    parser.add_argument(
        "-o",
        "--output",
        default="predicted",
        help="Директория для сохранения результатов",
    )

    args = parser.parse_args()

    ray.init(log_to_driver=False, num_gpus=1)

    process(args.model, args.image, args.output)

    ray.shutdown()
```

Листинг Б.3 – Модуль пользовательского приложения. Часть 3

```
import cv2



def apply_bboxes(
    im: cv2.Mat,
    bboxes,
    line_width: int = None,
    color=(200, 128, 128),
    txt_color=(255, 255, 255),
) -> cv2.Mat:
    lw = line_width or max(round(sum(im.shape) / 2 * 0.003), 2)
    x, y = im.shape[1], im.shape[0]

    for b, c in bboxes:
```

Листинг Б.4 – Модуль пользовательского приложения. Часть 4

```
        p1 = (int(x * b[0]), int(y * b[1]))
        p2 = (int(x * b[2]), int(y * b[3]))
        cv2.rectangle(im, p1, p2, color, thickness=lw,
           lineType=cv2.LINE_AA)

        label = f"{c:.2f}"
        tf = max(lw - 1, 1)  # font thickness
        w, h = cv2.getTextSize(label, 0, fontScale=lw / 3,
           thickness=tf)[
            0
        ]  # text width, height
        outside = p1[1] - h >= 3
        p2 = p1[0] + w, p1[1] - h - 3 if outside else p1[1] + h
           + 3
        cv2.rectangle(im, p1, p2, color, -1, cv2.LINE_AA)  #
           filled
        cv2.putText(
            im,
            label,
            (p1[0], p1[1] - 2 if outside else p1[1] + h + 2),
            0,
            lw / 3,
            txt_color,
            thickness=tf,
            lineType=cv2.LINE_AA,
        )

    return im
```

ПРИЛОЖЕНИЕ В
Утилитарный модуль

Листинг В.1 – Утилитарный модуль. Часть 1

```
import functools

import numpy as np


def get_bbox_center(bbox):
    return 0.5 * np.array([bbox[0] + bbox[2], bbox[1] +
       bbox[3]], dtype=np.float32)


def get_bbox_area(bbox) -> np.float32:
    return (bbox[2] - bbox[0]) * (bbox[3] - bbox[1])


def get_intersection(bbox_1: np.ndarray, bbox_2: np.ndarray) ->
   np.ndarray:
    if (
        (bbox_1 is None or bbox_2 is None)
        or (bbox_1[0] > bbox_2[2] or bbox_2[0] > bbox_1[2])
        or (bbox_1[3] < bbox_2[1] or bbox_2[3] < bbox_1[1])
    ):
        return None

    x_l, y_t = max(bbox_1[0], bbox_2[0]), min(bbox_1[1],
       bbox_2[1])
    x_r, y_b = min(bbox_1[2], bbox_2[2]), max(bbox_1[3],
       bbox_2[3])
    if x_r < x_l or y_b < y_t:
        return None

    return np.array([x_l, y_t, x_r, y_b])


def get_intersection_of_n_bboxes(bboxes: np.ndarray) ->
   np.ndarray:
    return functools.reduce(get_intersection, bboxes)
```

Листинг В.2 – Утилитарный модуль. Часть 2

```
def get_mass_center(bboxes: np.ndarray) -> np.ndarray:
    return np.array([get_bbox_center(bbox) for bbox in
       bboxes]).mean(axis=0)


def get_iou(bbox_1: np.ndarray, bbox_2: np.ndarray) ->
   np.float32:
    intersection = get_intersection(bbox_1, bbox_2)
    if intersection is None:
        return np.float32(0.0)

    intersection_area = get_bbox_area(intersection)
    bbx1_area, bbx2_area = get_bbox_area(bbox_1),
       get_bbox_area(bbox_2)

    return np.float32(
        intersection_area / float(bbx1_area + bbx2_area -
           intersection_area)
    )


def get_bbox_weight(bbox: np.ndarray) -> np.float32:
    return bbox[0] * bbox[2]
```