МИНОБРНАУКИ РОССИИ

РГУ нефти и газа (НИУ) имени И.М. Губкина

|  |  |
| --- | --- |
| Факультет: | **Автоматика и вычислительная техника** |
| Кафедра: | **Автоматизация систем управления** |
| Направление (специальность): | **09.03.01 Информатика и вычислительная техника** |
| Программа (специализация): | **Автоматизированные системы обработки информации и управления** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Оценка: |  | | | Рейтинг: |  |
| Подпись секретаря ГЭК: | | | | | |
|  | |  |  | | |
| (подпись) | |  | (фамилия, имя, отчество) | | |
|  | | | | | |
| (дата) | | | | | |

**ДИПЛОМНАЯ РАБОТА**

Применение методов машинного обучения для обнаружения разливов нефти на аэрофотоснимках

|  |  |
| --- | --- |
| на тему |  |
|  | |
|  | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| РУКОВОДИТЕЛЬ: | |  | | | ВЫПОЛНИЛ: |  | |
|  | |  | | | Студент группы | **АС-20-04** | |
|  | |  | | |  | (номер группы) | |
| Тупысев Антон Михайлович | |  | | | Зайцева Анна Андреевна | | |
| (фамилия, имя, отчество) | |  | | | (фамилия, имя, отчество) | | |
|  | |  | | |  | | |
| (подпись) | |  | | | (подпись) | | |
|  | |  | | |  | | |
| (дата) | |  | | | (дата) | | |
| КОНСУЛЬТАНТ ПО РАЗДЕЛУ: | |  | | |  | | |
|  | |  | | |  | | |
| (наименование раздела) | |  | | |  | | |
|  | |  | | |  | | |
| (должность, степень) | |  | | |  | | |
|  | |  | | |  | | |
| (фамилия, имя, отчество) | |  | | |  | | |
|  | |  | | |  | | |
| (подпись) | |  | | |  | | |
|  | Москва, 20 | | 24 |  | | |

МИНОБРНАУКИ РОССИИ

РГУ нефти и газа (НИУ) имени И.М. Губкина

|  |  |
| --- | --- |
| Факультет | **Автоматика и вычислительная техника** |
| Кафедра | **Автоматизация систем управления** |

**ЗАДАНИЕ НА ВЫПОЛНЕНИЕ**

**ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ДАНО студенту | | **Зайцевой Анне Андреевне** | группы | **АС-20-04** | |
| Применение методов машинного обучения для обнаружения разливов нефти на аэрофотоснимках | | (фамилия, имя, отчество в дательном падеже) |  | (номер группы) | |
| Тема ВКР: |  | | | |
|  | | | | |
|  | | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Руководитель: | | | | | Тупысев А.М., старший преподаватель | | | | | | | |
| (фамилия И.О., должность, ученая степень) | | | | | | | | | | | | |
| Тема ВКР, руководитель и консультант(ы) (при наличии) закреплены приказом | | | | | | | | | | | | | |
| № | | 3889-у | От « | | | 14 | » | Ноября | 20 | 23 | года | | |
| Закрепление изменено приказом (при наличии): | | | | | | | | | | | | | |
| № | |  | от « | | |  | » |  | 20 |  | года | | |
| Анализ существующих методов компьютерного зрения для обнаружения разливов нефти на аэрофотоснимках  **Раздел ВКР:** | | | |  | | | | | | | | |
|  |  | | | | | | | | | | | |
| (наименование раздела)  Задание и исходные данные по разделу:  Анализ существующих методов компьютерного зрения для обнаружения разливов нефти на аэрофотоснимках, анализ существующих нейросетевых архитектур, применяемых для детекции объектов на изображениях и выбор подходящей нейронные сети для решения поставленной задачи. | | | | | | | | | | | | |
|  |  | | | | | | | | | | |
|  |  | | | | | | | | | | |
|  |  | | | | | | | | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Раздел ВКР:** | | Предобработка данных аэрофотоснимков с разметкой и без нее | |
|  | (наименование раздела) | | |
| Задание и исходные данные по разделу:  Предобработка набора данных аэрофотоснимков с разметкой разливов нефти и без нее. | | | |
|  |  | |
|  |  | |
|  |  | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Раздел ВКР:** | | Разработка программного модуля для обнаружения разливов | | | | |
|  | нефти на аэрофотоснимках | | | | | |
| (наименование раздела)  Задание и исходные данные по разделу: | | | | | | |
| |  | | --- | |  | |  | |  | | | | | | | |
| **Раздел ВКР:** | |  | | | | |
|  | (наименование раздела) | | | | | |
|  |  | | | | | |
| Консультант (при наличии): | | |  |  |  | |
|  | | | (фамилия И.О., должность, ученая степень) |  | подпись | |
| Задание и исходные данные по разделу: | | | | | | |
|  |  | | | | |
|  |  | | | | |
|  |  | | | | |

1. Построение модели для обнаружения разливов нефти на

аэрофотоснимках.

2. Анализ полученных результатов и оценка качества построенной модели.

3. Разработка графического интерфейса.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Раздел ВКР:** | |  | | | | |
|  | (наименование раздела) | | | | | |
|  |  | | | | | |
| Консультант (при наличии): | | |  |  |  | |
|  | | | (фамилия И.О., должность, ученая степень) |  | подпись | |
| Задание и исходные данные по разделу: | | | | | | |
|  |  | | | | |
|  |  | | | | |
|  |  | | | | |

**Рекомендуемая литература:**

Глубокое обучение // Гудфеллоу Я., Бенджио И., Курвилль А. Г пер. с анг. А. А. Слинкина. – 2-е изд., испр. – М.: ДМК Пресс, 2018. – 652 с.: цв. ил.

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Задание выдал: | |  | |  | |  | | Тупысев А.М. | |
|  | |  | | (подпись руководителя) | |  | | (фамилия, имя, отчество) | |
| Задание принял к исполнению: |  | |  | |  | |  | | Зайцева А.А |
|  | | |  | | (подпись студента) | |  | | (фамилия, имя, отчество) |

**АННОТАЦИЯ**

Зайцева А.А. Применение методов машинного обучения для обнаружения разливов нефти на аэрофотоснимках, дипломная работа, 2024 – 65 с., 4 таб., 39 рис. Руководитель Тупысев А.М., старший преподаватель. Кафедра автоматизированных систем управления.

Проведен анализ существующих методов компьютерного зрения для обнаружения разливов нефти на аэрофотоснимках. Проведена предобработка данных аэрофотоснимков с разметкой и без нее. Проведена разработка программного обеспечения для автоматизации процесса обнаружения разливов нефти на аэрофотоснимках

.

**Оглавление**

[ВВЕДЕНИЕ 6](#_Toc168677145)

[ГЛАВА 1. Анализ существующих методов компьютерного зрения для обнаружения разливов нефти на аэрофотоснимках. 8](#_Toc168677146)

[1.1 Анализ разливов нефти 8](#_Toc168677147)

[1.2 Аналоговые методы обнаружения разливов нефти 10](#_Toc168677148)

[1.3 Современные нейронные сети для детекции объектов на изображениях 12](#_Toc168677149)

[1.4 YOLOv8. 16](#_Toc168677150)

[ГЛАВА 2. Предобработка набора данных аэрофотоснимков с разметкой разливов нефти и без нее 24](#_Toc168677151)

[2.1 Анализ данных 24](#_Toc168677152)

[2.2 Библиотеки для реализации 26](#_Toc168677153)

[2.3 Предобработка данных 28](#_Toc168677154)

[2.4 Подбор гиперпараметров 30](#_Toc168677155)

[ГЛАВА 3. Разработка программного модуля для обнаружения разливов нефти на аэрофотоснимках 36](#_Toc168677156)

[3.1 Обучение модели 36](#_Toc168677157)

[3.2 Визуализация результатов 37](#_Toc168677158)

[3.3 Метрики качества 39](#_Toc168677159)

[3.4 Разработка графического интерфейса 48](#_Toc168677160)

[ЗАКЛЮЧЕНИЕ 62](#_Toc168677161)

[Список литературы 63](#_Toc168677162)

# ВВЕДЕНИЕ

В последние десятилетия вопросы экологической безопасности приобрели особую значимость, так как антропогенная деятельность с каждым годом оказывает негативное воздействие на окружающую среду. Одной из серьезных экологических катастроф, связанных с деятельностью человека, является разлив нефти.

Разлив нефти — это случайное попадание нефти или нефтепродуктов в окружающую среду, которое происходит в результате аварий или утечек при добыче, транспортировке, хранении или использовании нефти. Это чрезвычайное событие может привести к масштабному загрязнению экосистем, нарушению жизнедеятельности морских и наземных организмов, а также оказывать негативное влияние на здоровье человека.

Своевременное обнаружение и оперативная ликвидация разливов нефти играют важную роль не только в сохранении экологии, но и в минимизации экономических потерь. Эффективные меры борьбы с разливами нефти позволяют сократить затраты на ликвидацию утечек, уменьшить потери нефти и обеспечить быструю реакцию на чрезвычайные ситуации.

Традиционные методы мониторинга разливов нефти, такие как визуальный осмотр и использование специализированного оборудования, часто оказываются недостаточно эффективными из-за их высокой трудоемкости и больших затрат времени и ресурсов. В связи с этим на первый план выходят современные технологии обработки данных, включая методы глубокого обучения, которые могут существенно повысить эффективность и скорость обнаружения участков разливов нефти. Особый интерес представляет применение нейронных сетей для анализа аэрофотоснимков, полученных с летательных аппаратов, спутников, а также камер, расставленных по периметру судов. Они могут быть интегрированы в системы раннего предупреждения и оперативного реагирования. Такой подход позволяет проводить непрерывный мониторинг обширных территорий и выявлять признаки разлива нефти на ранних стадиях.

Таким образом, целью данной дипломной работы является разработка программного обеспечения для автоматизации процесса обнаружения разливов нефти на аэрофотоснимках. Для достижения цели дипломной работы необходимо выполнить следующие задачи:

1. Анализ существующих методов компьютерного зрения для обнаружения разливов нефти на аэрофотоснимках.
2. Анализ существующих нейросетевых архитектур, применяемых для детекции объектов на изображениях и выбор подходящей нейронные сети для решения поставленной задачи.
3. Предобработка набора данных аэрофотоснимков с разметкой разливов нефти и без нее.
4. Построение модели для обнаружения разливов нефти на аэрофотоснимках.
5. Анализ полученных результатов и оценка качества построенной модели.
6. Разработка графического интерфейса.

# ГЛАВА 1. Анализ существующих методов компьютерного зрения для обнаружения разливов нефти на аэрофотоснимках.

## 1.1 Анализ разливов нефти

Разлив нефти — это серьезная угроза для окружающей среды, возникающая по различным причинам. Несвоевременное обнаружение таких разливов может иметь катастрофические последствия, усугубляя их воздействие на экосистемы и здоровье человека.

Один из наиболее опасных источников — это танкеры, перевозящие большие объемы нефти. В случае аварии на одном из таких судов, большое количество нефти может разлиться в море. Часто такие аварии происходят вдали от берега, что значительно затрудняет работу по ликвидации разлива. Примером является авария танкера "Exxon Valdez" в 1989 году, которая привела к обширному загрязнению вод и береговой линии.

Источником разлива нефти также являются буровые установки и добывающие платформы, которые могут привести к крупным разливам нефти. Катастрофа на платформе "Deepwater Horizon" в 2010 году в Мексиканском заливе стала одним из самых масштабных примеров такого рода.

Трубопроводы, проложенные под водой или на суше, также могут стать источником разливов нефти вследствие повреждений их целостности, возникших из-за коррозии, природных катастроф или человеческой деятельности.

Кроме того, разливы нефтепродуктов, таких как бензин, дизельное топливо и моторные масла, также представляют угрозу окружающей среде. Аварии на автомобилях, железнодорожных транспортных средствах и других механизмах, использующих эти виды топлива и смазочные материалы, могут привести к их разливу на землю и в водные источники. Помимо этого, несчастные случаи на складах и предприятиях, где производят, хранят или используются такие продукты, также могут привести к серьезным разливам, загрязняющим окружающую среду и угрожающим здоровью людей и животных [23].

Разлив нефтепродуктов может происходить по нескольким причинам:

1. Превышение объемов добычи нефти;
2. 2.Устаревшее или некачественное оборудование для добычи, транспортировки и переработки нефти, такое как трубопроводы и танкеры;
3. Диверсионные акты, совершаемые экстремистскими группировками;
4. Природные катастрофы, такие как землетрясения, ураганы и наводнения, могут повредить инфраструктуру для добычи и транспортировки нефти, что приводит к утечкам;
5. Ошибки операторов, неправильная эксплуатация оборудования и недостаточное техническое обслуживание могут стать причиной аварийных разливов нефти.

Согласно текущей классификации чрезвычайных ситуаций, вызванных разливами нефти и нефтепродуктов, все возможные разливы делятся на пять уровней:

1. Локального значения: объем разлива нефтепродуктов не превышает 100 тонн;
2. Муниципального значения: объем разлива от 100 до 500 тонн;
3. Территориального значения: объем разлива от 500 до 1000 тонн;
4. Регионального значения: объем разлива от 1000 до 5000 тонн;
5. Федерального значения: разливы, объем которых превышает 5000 тонн или происходят на территории сопредельных стран [22].

Разливы нефти можно классифицировать по типу разлившейся нефти. Разливы сырой нефти являются одними из самых тяжелых и плотных. Они могут оставаться на поверхности воды длительное время, что приводит к серьёзному ущербу окружающей среде. А разливы переработанных нефтепродуктов, таких как дизельное топливо, бензин, мазут, имеют различные уровни токсичности и воздействия на окружающую среду. Разливы, содержащие смесь нефти и химикатов, представляют собой еще более сложную задачу для ликвидации последствий.

Химические вещества, смешанные с нефтью, могут быть более трудно ликвидируемыми и требуют дополнительных технологий и ресурсов для их удаления. На суше такие разливы могут стать серьезными опасностями для сельскохозяйственных угодий, водоемов и здоровья людей, живущих вблизи загрязненных территорий.

Разливы на поверхности воды образуют пленку, вызывающую проблемы с проникновением воздуха в среду обитания морских животных. Обнаружение и контроль таких разливов возможны с поверхности, но их ликвидация сложна. Разливы под водой требуют специализированных технологий из-за сложности их обнаружения и ликвидации. Эмульсии "нефть в воде" представляют особую сложность для очистки. Взаимодействие нефти с водой влияет на методы ликвидации разливов [22].

## **1.2 Аналоговые методы обнаружения разливов нефти**

Раннее обнаружение разливов нефти осуществлялось с использованием стандартных методов, включающих в себя оптические, химические и радиолокационные подходы.

**1.2.1 Оптические методы**

Оптические методы обнаружения разливов нефти основываются на использовании световых волн для обнаружения и анализа нефтяных пятен. Они включают в себя использование спутниковых съемок и аэрофотосъемок, инфракрасной и ультрафиолетовой съемки и лазерной флюоресценции. Поляризационные фильтры помогали уменьшить блики и выделять нефтяные пятна, а инфракрасная фотография использовалась для изучения специфического поглощения нефтью инфракрасного излучения. Спектральный анализ позволял различать нефть и воду по их спектральным характеристикам [19]. Недостатками данного метода являются его зависимость от погодных условий и сложность интерпретации данных. Облачность, туман, дождь, плохие погодные условия могут снизить эффективность оптических методов. Также оптические методы могут давать ложные сигналы из-за наличия других веществ, таких как водоросли, пена и другие природные материалы [20].

**1.2.2 Химические методы**

Химические методы обнаружения разливов нефти основываются на использовании химических реагентов, таких как индикаторы и флуоресцентные красители, которые меняют цвет или свечение при контакте с нефтью. Это позволяет визуально обнаруживать нефтяные разливы на поверхности воды. Химические методы имеют свои ограничения и недостатки. Окружающая среда может существенно влиять на результаты химических методов: низкая температура или высокая соленость воды могут снизить их эффективность в определенных условиях. Еще одним важным аспектом является возможность ложных срабатываний, вызванных реакцией химических индикаторов с другими веществами. Это создает потенциальный риск неправильной интерпретации результатов и требует тщательного контроля и анализа для исключения подобных ошибок [12].

**1.2.3 Радиолокационные системы**

Радиолокационные системы и микроволновая радиометрия также применяются для обнаружения разливов нефти. Они работают на основе изменений радиосигналов, отраженных от поверхности воды с нефтяным пятном, что позволяло идентифицировать такие разливы даже при плохой видимости или в условиях ночи. Недостатком радиолокационных систем является необходимость использования сложного и дорогостоящего оборудования. Такие системы требуют специализированных приборов и компонентов, что повышает их стоимость и делает их недоступными для некоторых случаев из-за финансовых ограничений. Кроме того, сложность оборудования может потребовать специальной квалификации для его обслуживания и настройки, что увеличивает затраты на обучение персонала и обслуживание системы [22].

Рассмотренные методы широко используются, несмотря на их ограничения в точности, скорости обработки и способности работать в различных условиях окружающей среды. Однако, с развитием цифровых технологий процесс обнаружения разливов нефти на изображениях может быть более автоматизирован за счет использования нейронных сетей, которые способны извлекать признаки, соответствующие разливам, из изображений.

## **1.3** Современные нейронные сети для детекции объектов на изображениях

В настоящее время существует множество архитектур нейронных сетей, которые способны решать задачи детекции. Рассмотрим наиболее популярные из них для выбора предпочтительной архитектуры для решения задачи обнаружения разливов нефти на аэрофотоснимках.

**1.3.1 R-CNN (Region Based Convolution Neural Network)**

Алгоритм R-CNN разделяет входное изображение на 2000 регионов и применяет к каждому из них отдельно сверточную нейронную сеть. Сначала происходит начальная подсегментация, отбираются регионы, которые могут содержать части объектов. Далее похожие области объединяют в более крупные с использованием жадных алгоритмов. После чего предполагается, что полученные области должны содержать искомые объекты [3].

Недостатками данной архитектуры можно считать длительное обучение сети из-за необходимости обработки большого количества регионов, невозможность работать в режиме реального времени из-за большого времени обработки каждого тестового изображения, отсутствие обучения при использовании алгоритма выборочного поиска, что может привести к неточным и неправильным результатам [11].

**1.3.2 SSD (Single Shot Detector)**

Данная модель глубокого обучения способна выполнять классификацию объектов на изображении за один проход по архитектуре нейронной сети в режиме реального времени. Признаки изображения извлекаются путем его прохождения через несколько сверточных слоев. В результате получается несколько ограничительных рамок, для каждой из которых вычисляется оценка класса и четыре координаты.

Архитектура SSD имеет более высокую точность обнаружения объектов по сравнению с R-CNN, но при этом имеет проблему с маленькими объектами. Для достижения высокой эффективности необходим большой объем данных для обучения [9].

**1.3.3 YOLO (You Only Look Once)**

Главное отличие алгоритма YOLO от других алгоритмов обнаружения объектов заключается в том, что он способен намного быстрее опознавать объекты в режиме реального времени. Поданное на вход изображение проходит через сверточную нейронную сеть только один раз.

Поданное на вход изображение разбивается на сетку. Для каждой получившейся ячейки создаётся вектор, который будет хранить ее параметры: уверенность в нахождении в ячейке объекта, точность предсказания.

Архитектура нейронной сети состоит из 24 сверточных слоев, после которых идут два полносвязных слоя.

В результате работы сети также, как и при работе SSD, сгенерируется множество ограничительных рамок, которые подвергаются операции non-maximum suppression, которая выбирает наиболее подходящие гипотезы.

YOLO также имеет трудности с обнаружением мелких объектов, и для достижения высокой точности необходимы большие наборы данных. Но при этом имеет высокую скорость обнаружения объектов –45 кадров в секунду и отсутствие необходимости использования больших вычислительных мощностей [8].

**1.3.4 Retina-Net**

Архитектуру Retina-Net предложила компания Facebook Al Research для решения проблемы детекции объектов на изображениях, имеющих разные масштабы. Эта нейронная сеть является двухэтапным детектором, как и нейронная сеть R-CNN. Сначала изображение разбивается на регионы, а после происходит оценка ограничивающих рамок и классов объектов в каждом регионе.

Retina-Net состоит из множества уровней признаков, каждый из которых соответствует разным масштабам объектов на изображении.

Для решения дисбаланса классов используется функция "Focal\_Loss()", которая фокусируется на классификационной потере для трудно классифицируемых параметров за счет уменьшения веса легко классифицируемых параметров. Это помогает повысить общую производительность сети.

Использование Feature Pyramid Network для многомасштабного анализа изображений позволяет эффективно обрабатывать объекты различных размеров и обеспечивает хорошую точность детекции.

Нейронная сеть имеет медленную скорость обучения, проблемы с обнаружением мелких объектов и вычислительную сложность [7].

**1.3.5 Анализ нейронных сетей**

Для решения задачи детекции разливов нефти на аэрофотоснимках следует использовать одноступенчатую архитектуру, способную точно и быстро обнаруживать разливы в режиме реального времени.

Таблица 1 приставляет сравнение рассмотренных архитектур методом непосредственной оценки, каждая из которых оценивается по нескольким критериям. Всем нейронным сетям по каждому из критериев присвоено значение от 0 до 10.

Также для каждого критерия присвоен весовой коэффициент в диапазоне от 0 до 1, отражающий значимость данного критерия при выборе архитектуры.

Для каждой модели суммируются критерии, умноженные на соответствующие им весовые коэффициенты. Общий результат вычисляется по формуле 1.

Таблица – Сравнение архитектур нейронных сетей

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Критерий оценки | R-CNN | | YOLO | | SSD | | RetinalNet | |  |
| Оценка | N | Оценка | N | Оценка | N | Оценка | N |
| Архитектура | Двухступенчатая | 2 | Одноступенчатая | 10 | Одноступенчатая | 10 | Двухступенчатая | 2 | 1 |
| Скорость обнаружения | Ниже среднего | 4 | Очень высокая | 10 | Высокая | 8 | Высокая | 8 | 1 |
| Точность | Высокая | 8 | Средняя до высокой | 7 | Средняя до высокой | 7 | Высокая | 8 | 1 |
| Обработка объектов разных масштабов | Эффективно | 10 | Менее эффективно | 7 | Эффективно | 10 | Эффективно | 10 | 0,8 |
| Устойчивость к дисбалансу классов | Более устойчива | 10 | Менее устойчива | 8 | Менее устойчива | 7 | Более устойчива | 10 | 0,5 |
| Оптимизация вычислений | Требовательно к вычислениям | 3 | Эффективное использование GPU | 10 | Эффективное использование GPU | 10 | Эффективное использование GPU | 10 | 0,6 |

Продолжение таблицы 1

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Сложность реализации | Высокая | 3 | Низкая | 10 | Средняя | 6 | Средняя | 6 | 0,4 |
|  | 0,612 | | 0,951 | | 0,946 | | 0,804 | |  |

Учитывая результаты оценки в таблице, где YOLO получила наивысший балл (0,951), можно сделать вывод о том, что эта нейронная сеть является более предпочтительным инструментом для успешного решения задачи детекции разливов нефти на аэрофотоснимках, поэтому для достижения поставленных задач будет использоваться эта архитектура.

## 1.4 YOLOv8.

В этом разделе будет подробно рассмотрена нейронная сеть YOLO восьмой версии. YOLO – это алгоритм обнаружения объектов, который использует глубокое обучение и нейронные сети для быстрого и точного обнаружения объектов на изображениях или видео в режиме реального времени. Он был представлен в 2015 году Джозефом Редмоном, Сантошем Диввала, Россом Гиршиком и Али Фархади в их научной статье "You Only Look Once: Unified, Real-Time Object Detection".

**1.4.1 Основные понятия в теории сверточных нейронных сетей**

В данном разделе будут рассмотрены основные операции сверточных нейронных сетей, так как именно они являются основной составляющей архитектуры YOLOv8.

Convolution (свертка). В данной операции применяется фильтр (ядро свертки) к входному изображению или к предыдущему слою для выделения признаков определяемых объектов. Фильтр перемещается по имеющимся данным и выполняет операции элементарного умножения и суммирования. Результатом является карта признаков (feature map), которая подчеркивает определенные характеристики, такие как края, текстуры и т.д.

Pooling (пулинг) необходим для уменьшения размерности карты признаков с сохранением важной информации. Чаще всего применяются Max Pooling (максимальное значение из области) и Average Pooling (среднее значение из области). Данная операция помогает снизить вычислительные затраты и предотвратить переобучение.

Fully Connected Layers (полносвязные слои) служат для окончательной классификации или предсказания. Обычно они расположены в конце CNN. Каждый нейрон этого слоя связан со всеми нейронами предыдущего слоя. Выходной вектор обрабатывается с использованием функции активации (например, softmax для задач классификации).

Нормализация "batch normalization" применяется для нормализации входных данных для каждого мини-пакета, стабилизируя и ускоряя обучение.

Приводит значение активаций к заданному распределению, улучшая общую производительность сети.

Регуляризация методом отсечения "Dropout", где случайные нейроны "отключаются" во время обучения, помогает предотвратить переобучение и улучшает способность сети [18].

**1.4.2 Архитектура YOLOv8**

Архитектура YOLOv8 состоит из двух основных частей – Backbone и Hesd. YOLOv8 использует собственный backbone CSPDarknet53. На рисунке 1 изображена его архитектура, которая представляет пирамиду из нескольких сверточных слоев и блоков остаточной связи [1]. Это главная особенность нейронной сети YOLO. Сверточные слои отвечают за извлечение признаков из входного изображения.

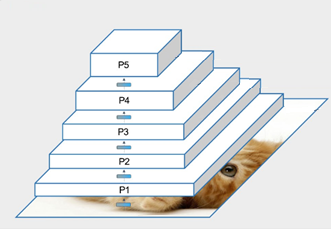


Рисунок – Архитектура Backbone

Такой метод позволяет модели обнаруживать объекты различных размеров на изображении. Сверточный блок состоит из двухмерного сверточного слоя и слоя нормализации, а также функции активации. Пример сверточного блока представлен на рисунке 2 [1].

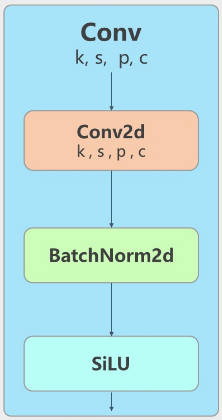


Рисунок 2 – Сверточный блок

Еще одним значительным улучшением в YOLOv8 является использование функции активации Exponential Linear Unit (ELU). ELU помогает ускорить обучение в глубоких нейронных сетях за счет решения проблемы исчезающего градиента, что приводит к более быстрой конвергенции.На разных стадиях этапа backbone извлекает признаки и объединяет их в карту признаков для дальнейшей возможности определения объектов разных размеров.

На следующем этапе используется Spatial Pyramid Pooling Fast (SPPF). Его архитектура изображена на рисунке 3 [1]. Этот слой позволяет сети обрабатывать входные данные разного размера и извлекать более информативные признаки с более высокой точностью. Он использует два сверточных слоя, между которыми находятся pulling-операции, работающие на разных уровнях, такие как max pulling.

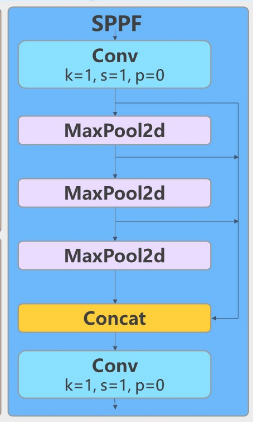


Рисунок 3 – Архитектура Spatial Pyramid Pooling Fast

Далее идет новый модуль C2F, представленный на рисунке 4 [1]. Заменивший традиционную сети Feature Pyramid Network (FPN). Этот модуль объединяет семантические признаки высокого уровня с пространственной информацией низкого уровня, что приводит к улучшению точности обнаружения маленьких объектов. C2F состоит из сверточного слоя, который впоследствии разделяет карты признаков. Один из них сразу попадает в блок объединения, а другой сначала попадает в место сужения, которых может быть несколько. После чего следует сверточный слой.

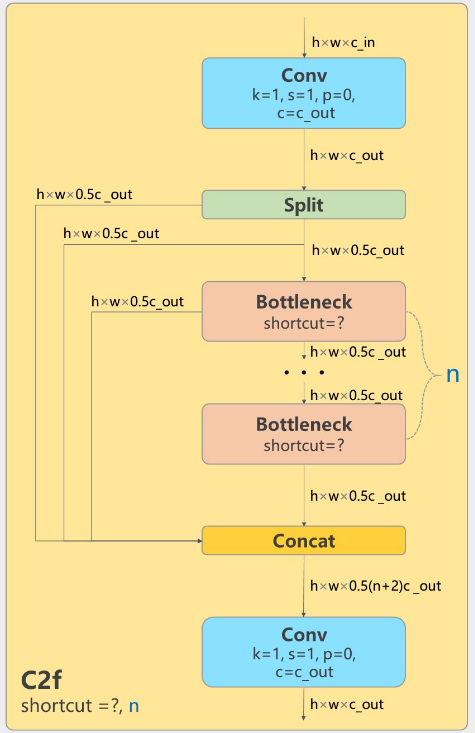


Рисунок 4 – C2f

Место сужения представляет собой последовательность сверточных слоев. Его архитектура показана на рисунке 5 [1].

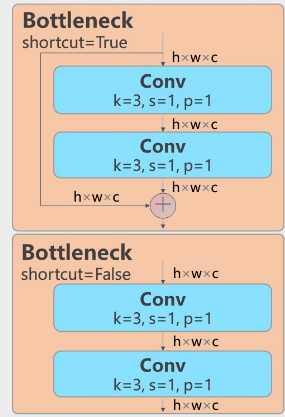


Рисунок 5– Архитектура места сужения

Блок Dense, представленный на рисунке 6, отвечает за прогнозирование и имеет две подсети [1]. Одна из них предсказывает ограничивающую рамку, другая предсказывает класс объекта. Каждая состоит из последовательно расположенных двух одномерных, одного двухмерного сверточных слоев и функции ошибки.

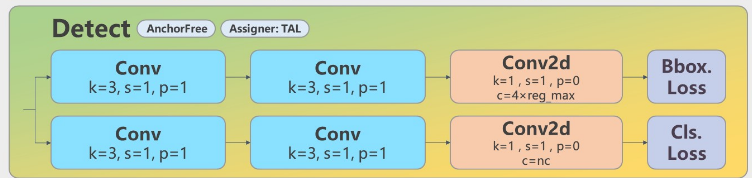


Рисунок 6 – архитектура Dense блока

Подсети Dense используют признаки, извлеченные предыдущими блоками, для обнаружения объектов и их классификации.

**1.4.3 Версии YOLOv8**

У YOLOv8 есть пять версий для обнаружения объектов, каждая из которых оптимизирована для разных сценариев использования. В целом, различия между этими моделями связаны с их архитектурой, количеством параметров, производительностью и точностью:

– YOLOv8n (Nano) – самая компактная версия. Подходит для задач обнаружения объектов, сегментации экземпляров, оценки положения и классификации. Оптимизирована для скорости и хорошей точности. Использует анкерно-свободную разделенную голову Ultralytics;

– YOLOv8s (Small) – оптимизирована для высокой скорости. Хорошо подходит для реального времени обнаружения объектов. Поддерживает задачи обнаружения, сегментации и классификации;

– YOLOv8m (Medium) – средний баланс между точностью и скоростью. Подходит для различных задач, включая обнаружение, сегментацию и классификацию;

– YOLOv8l (Large) – предлагает высокую точность. Используется для задач, где точность является приоритетом;

– YOLOv8x (Extra Large) – самая мощная версия YOLOv8 подходит для сложных задач, требующих высокой точности.

Количество параметров в моделях YOLOv8 зависит от их архитектуры, размера и глубины. Эти параметры играют важную роль в определении структуры и производительности сети. Они могут быть настроены в зависимости от конкретной задачи и доступных вычислительных ресурсов.

В контексте архитектуры сети параметры D, W и Mc обычно обозначают следующее:

– D (Depth Multiplier) – этот параметр указывает на количество узких мест или блоков в сети. Узкое место – это часть сети, где количество каналов временно уменьшается, а затем восстанавливается. Это помогает уменьшить количество параметров и вычислений в сети;

– W (Width Multiplier) – этот параметр указывает на количество каналов в сверточных слоях сети. Увеличение ширины сети обычно приводит к увеличению ее емкости и, следовательно, к улучшению ее производительности, хотя это также увеличивает количество параметров и вычислений;

– Mc (Number of Channelsx) – этот параметр указывает на количество каналов в выходных данных сверточного слоя. Каждый канал обычно соответствует определенному признаку, который был извлечен из входных данных.

В таблице 2 представлено сравнение версий YOLOv8 по рассмотренным параметрам.

Таблица 2 – Значение параметров Depth Multiplier, Width Multiplier и Number of Channels версий VOLOv8

|  |  |  |  |
| --- | --- | --- | --- |
| Версия модели (Model) | D  Глубина  (Depth Multiplier) | W  Ширина  (Width Multiplier) | Mc  Number of Channelsx |
| N | 0,33 | 0,25 | 1024 |
| S | 0,33 | 0,50 | 1024 |
| M | 0,67 | 0,75 | 768 |
| L | 1,00 | 1,00 | 512 |
| XL | 1,00 | 1,25 | 512 |

**1.4.4 Ключевые инновации в YOLOv8**

Несколько ключевых инноваций способствуют впечатляющей производительности YOLOv8:

– YOLOv8 использует потерю GIoU (Generalized Intersection over Union). Обобщенное пересечение по объединению представляет собой более продвинутую версию метрики IoU, которая учитывает форму и размер ограничивающих рамок, повышая точность локализации объекта.

– YOLOv8 увеличила среднюю точность (AP) на 1,2% по сравнению с YOLOv7, что является значительным улучшением. Это было достигнуто за счет уменьшения размера файла weight модели на 80,6 М, что сделало модель более эффективной и простой в развертывании в средах с ограниченными ресурсами.

– YOLOv8 включает механизм пространственного внимания, который фокусируется на соответствующих частях изображения, что приводит к более точной локализации объекта.

– Модуль C2f эффективно сочетает высокоуровневые семантические функции с низкоуровневой пространственной информацией, повышая точность обнаружения небольших объектов.

– Узкие места в магистрали CSPDarknet53 снижают вычислительную сложность при сохранении точности. Кроме того, уровень Spatial Pyramid Pooling Fast (SPPF) фиксирует объекты в нескольких масштабах, что еще больше повышает производительность обнаружения.

– В YOLOv8 используются различные методы увеличения объема данных для улучшения обобщения и сокращения переобучения. Смешанная тренировка точности еще больше повышает скорость и эффективность обучения [10].

Таким образом, были рассмотрены различные методы обнаружения разливов, включая как традиционные, так и современные подходы. Среди традиционных аналоговых методов – оптические, химические и радиолокационные методы. Также был проведен анализ современных нейронных сете таких как R-CNN, YOLO, SSD и Retina-Net. В результате которого была выбрана подходящая для решения поставленной задачи нейронная сеть YOLOv8. После чего были рассмотрены особенности и архитектура выбранной нейронной сети, подчеркнув ее ключевые преимущества и уникальные возможности.

# ГЛАВА 2. Предобработка набора данных аэрофотоснимков с разметкой разливов нефти и без нее

## 2.1 Анализ данных

Для решения задачи обнаружения разливов нефти на аэрофотоснимках был использован специализированный набор данных, который был загружен с платформы Roboflow [5].

Roboflow представляет собой удобный инструмент для аннотирования и обработки изображений, создания различных форматов наборов данных и их масштабирования. Он обеспечивает возможность быстро и эффективно подготовить данные для обучения моделей машинного обучения.

Набор данных включает в себя изображения и соответствующие метаданные, необходимые для обучения и тестирования моделей, содержит 2433 тестовых, 90 валидационных и 20 тестовых пар изображений и аннотаций, что обеспечивает достаточное количество примеров для эффективного обучения и тестирования моделей.

**2.1.1 Анализ изображений**

Изображения размером 640x640 представляли собой аэрофотоснимки, сделанные с различных высот и в разных условиях освещенности. Они содержат разнообразные природные и искусственные объекты, среди которых присутствовали области, загрязненные нефтью. Пример изображения представлен на рисунке 7(a). В набор включены снимки как с высокими, так и с низкими разрешениями, чтобы модели могли эффективно работать в условиях ограниченных ресурсов. Все изображения представлены в формате JPG.

Часть тренировочного набора включала изображения с аугментацией, (рисунок 7(b)), которая заключалась в различных преобразованиях, увеличивающих разнообразие данных без изменения их сути. Каждое изображение состояло из четырёх снимков, которые поворачивались на 90, 180 и 270 градусов. Кроме того, применялась горизонтальная и вертикальная зеркальная симметрия. Масштабирование изображений в различных пропорциях позволило моделям лучше справляться с изменениями размеров объектов. Сдвиги по горизонтали и вертикали добавляли вариативность в положение объектов, что улучшало их способность к обобщению. Регулировка яркости и контрастности помогала моделям адаптироваться к разным условиям освещенности. Наложение шума необходимо для устойчивости моделей к реальным помехам.

Тестовые и валидационные данные не подвергались аугментации для обеспечения объективной оценки производительности моделей на неизменённых исходных изображениях. Это позволило гарантировать, что результаты тестирования отражают реальную способность моделей справляться с новыми, незнакомыми данными.

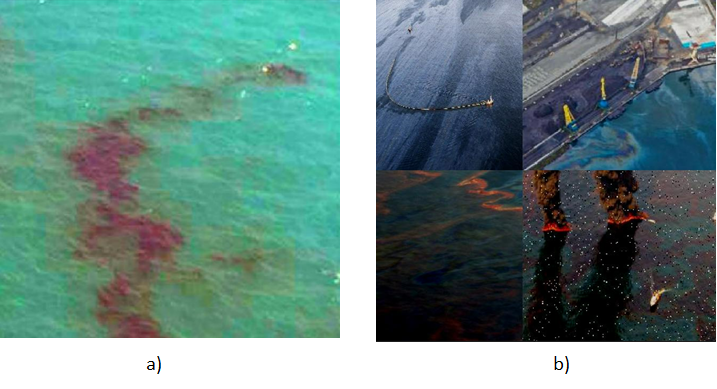
****

Рисунок 7 – Изображения набора данных

а) – без аугментации b) – бс аугментацией

**2.1.2 Аннотационный файл**

Каждое изображение в наборе данных сопровождается файлом с аннотациями, который содержит информацию об координатах ограничивающих рамках каждого разлива нефти, представляющие собой набор чисел, указывающих центр изображения, также ширину и высоту рамки в формате (cx, cy, w, h) и соответствующий класс – метка, указывающая на категорию объекта, ограниченного рамкой. Эти файлы представлены в формате XML. На рисунке 8 приведен пример файла с аннотациями.

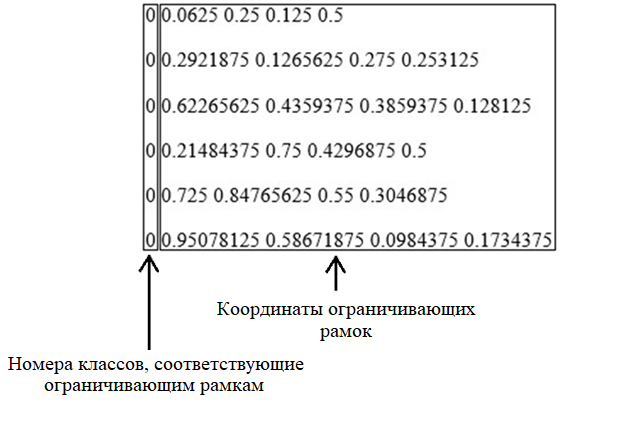


Рисунок 8 – Аннотационный файл

## 2.2 Библиотеки для реализации

Для реализации задачи обнаружения разливов нефти на аэрофотоснимках были рассмотрены библиотеки PyTorch, Keras и TensorFlow.

Несмотря на то, что PyTorch достаточно простой и гибкий в разработке, имеет удобные и привычные инструменты для отладки кода, поддерживает декларативные параллелизм данных, было решено не использовать данный фреймворк в разработке программы. Так как он представляет мало интерфейсов для мониторинга и визуализации результатов работы модели, использующей компьютерное зрение. Поэтому основными инструментами, используемыми при реализации, стали библиотеки Keras и TensorFlow. Выбор данных библиотек обоснован их простотой, мощностью, поддержкой сообщества и гибкостью.

**2.2.1 Библиотека Keras**

Keras – Python-библиотека, специализированная на глубоком машинном обучении. Keras не выполняет сложные математические вычисления самостоятельно, а служит своеобразной оболочкой над другими библиотеками. С версии 2.3 Keras стал надстройкой над библиотекой TensorFlow, основным движком для машинного обучения. Ранее он поддерживал библиотеки Theano и CNTK, но в новых версиях поддержка этих библиотек прекратилась. Keras разработан как гибкая и модульная библиотека с открытым исходным кодом. Он облегчает процесс создания и модификации моделей машинного обучения.

Основное применение данной библиотеки – создание, настройка и тестирование нейронных сетей. Она предоставляет удобные инструменты для построения и обучения моделей, обработки входных данных, визуализации моделей и многое другое. Keras упрощает процесс разработки и обучения моделей машинного обучения, делая его доступным для широкого круга специалистов [14].

**2.2.2 Библиотека TensorFlow**

TensorFlow – библиотека для глубокого машинного обучения, созданная Google и предназначенная для работы с различными направлениями машинного обучения, включая нейронные сети. Название произошло от терминов "тензор" и "поток данных", которые являются основными концепциями библиотеки.

Библиотека предоставляет высокоуровневые абстракции для создания моделей машинного обучения, что делает процесс разработки более простым и удобным. TensorFlow использует графы для представления моделей и вычислений, что позволяет ему эффективно управлять сложными вычислениями.

Основными применениями TensorFlow являются задачи классификации, распознавания образов и анализа естественного языка. Благодаря своей гибкости и мощным возможностям, TensorFlow используется широким кругом специалистов по машинному обучению, исследователей и разработчиков.

Особенности TensorFlow включают: высокий уровень абстракции, интерактивную разработку моделей, гибкость, кроссплатформенность и большое сообщество пользователей, которое активно развивает и поддерживает эту библиотеку.

Хотя TensorFlow предоставляет множество преимуществ, включая высокий уровень абстракции и широкие возможности для создания моделей машинного обучения, он также имеет свои недостатки, такие как собственные стандарты, высокое потребление памяти и сложность в изучении для новичков. Однако, при правильном использовании и понимании этих недостатков, TensorFlow остается одним из самых мощных инструментов в области машинного обучения [16].

## 2.3 Предобработка данных

**2.3.1 Подготовка данных к обучению**

Для точного определения координат ограничивающих рамок необходимо применить их преобразования из диапазона от 0 до 1 в диапазон от 0 до 640. После чего к ним применяется функция "convert\_cyr\_to\_xywh()", которая приведена в листинге 1. Она преобразует координаты ограничивающих рамок из формы cxcywh (cx, cy, w, h), где (cx, cy) – координаты центра ограничивающей рамки в формат xywh (x, y, w, h), где (x, y) – координаты верхнего левого угла рамки. (w, h) – определяют ширину и высоту рамки.

def convert\_cxcywh\_to\_xywh(bbox\_cxcywh):

    bbox\_cx, bbox\_cy, bbox\_w, bbox\_h = tf.split(bbox\_cxcywh, 4, axis=-1)

    bbox\_xmin = bbox\_cx - bbox\_w / 2.0

    bbox\_ymin = bbox\_cy - bbox\_h / 2.0

    bbox\_xmax = bbox\_cx + bbox\_w / 2.0

    bbox\_ymax = bbox\_cy + bbox\_h / 2.0

    bbox\_xywh = tf.concat([bbox\_xmin, bbox\_ymin, bbox\_w, bbox\_h], axis=-1)

        "boxes": bbox\_cxcywh,

    }

    return {"images": tf.cast(image, tf.float32), "bounding\_boxes": bounding\_boxes}

Листинг 1 – Функция "convert\_cxcywh\_to\_xywh()"

Функция "load\_image()", представленная в листинге 2, загружает изображение по заданному пути. Изображение читается с диска с помощью "tf.io.read\_file()" и декодируется в формат JPEG с использованием "tf.image.decode\_jpeg()". Далее изменяется размер изображения до целевого (640x640) с использованием "tf.image.resize()".

def load\_image(image\_path, target\_width, target\_height):

    image = tf.io.read\_file(image\_path)

    image = tf.image.decode\_jpeg(image, channels=3)

    image = tf.image.resize(image, [target\_height, target\_width])

    return image

Листинг – Функция "load\_image()"

Функция "load\_dataset()" загружает изображение и аннотации (ограничивающие рамки и классы) по заданному пути. Она описана в листинге 3. Сначала загружается изображение с помощью функции "load\_image()". Затем ограничивающие рамки преобразуются в требуемый формат с помощью функции "convert\_cxcywh\_to\_xywh()". Создается словарь "bounding\_boxes", содержащий классы и рамки. Все данные (изображения и ограничивающие рамки) объединяются в один словарь и возвращаются.

def load\_dataset(image\_path, classes, bbox):

    image = load\_image(image\_path, target\_width, target\_height)

    bbox\_cxcywh = convert\_cxcywh\_to\_xywh(bbox.to\_tensor())

    bounding\_boxes = {

        "classes": classes,

        "boxes": bbox\_cxcywh,

    }

    return {"images": tf.cast(image, tf.float32), "bounding\_boxes": bounding\_boxes}

Листинг – Функция "load\_dataset()"

Как и в большинстве последних версий YOLO, для обучения любой из моделей YOLOv8 необходимо создать YAML файл, который включает пути к наборам изображений, количество классов и их наименования. Необходимо определить пути к преобразованным изображениям и аннотациям для тренировочных и валидационных данных. После этого создаются списки XML файлов с аннотациями для обучающего и валидационного наборов данных. Для этого используются функции "os.listdir()" и "os.path.join()" для получения полных путей к XML файлам в соответствующих папках. Данные обучающего и валидационного набора обрабатываются с помощью функции "parse\_annotation()", описанной в листинге 4.

Функция "parse\_annotation()", обрабатывает каждый xml-файл, извлекает информацию об изображении и его аннотации, а затем возвращает список словарей, каждый из которых содержит путь к изображению, описание ограничивающих рамок и классы объектов на изображении.

def parse\_annotation(xml\_files, path\_images, path\_annot):

    data = []

    for xml\_file in xml\_files:

        image\_name = os.path.splitext(os.path.basename(xml\_file))[0] + '.jpg'

        image\_path = os.path.join(path\_images, image\_name)

        with open(xml\_file, 'r') as f:

            lines = f.readlines()

        boxes = []

        classes = []

        for line in lines:

            line\_data = line.strip().split(' ')

            class\_id = int(line\_data[0])

            box\_data = list(map(float, line\_data[1:]))

            boxes.append(box\_data)

            classes.append(class\_id)

        data.append({"image\_path": image\_path, "bounding\_boxes": boxes, "classes": classes})

    return data

)

Листинг – Функция "parse\_annotation()"

## 2.4 Подбор гиперпараметров

Подбор гиперпараметров — это важный итеративный процесс, который требует внимательного анализа результатов на каждом этапе. Правильная настройка гиперпараметров может значительно улучшить производительность будущей модели. Библиотека Keras предоставляет мощные инструменты для разработки, обучения, настройки и развертывания нейронных сетей, что делает процесс подбора гиперпараметров более удобным и эффективным.

Функция "keras\_cv.models.YOLOV8Detector()" используется для создания экземпляра модели YOLOv8 для обнаружения объектов на изображениях. Она позволяет настраивать различные аспекты модели, такие как:

– "Backbone" – это базовая сеть для извлечения признаков из изображений. Она выполняет начальную обработку изображений, выделяя ключевые особенности, которые будут использоваться для детекции объектов;

– "num\_classes " задает количество классов объектов, которые модель должна распознавать;

– "bounding\_box\_format" указывает на формат представления ограничивающих рамок (bounding-boxes). Это может быть формат "xywh" (x, y, ширина, высота) или "corners" (координаты углов ограничивающей рамки);

– "fpn\_depth " устанавливает глубину сети Feature Pyramid Network (FPN), которая будет использована для улучшения детекции объектов разных размеров. Чем больше значение этого параметра, тем лучше результат распознавания объектов различных масштабов.

Функция "tf.keras.optimizers.Adam()" создает оптимизатора "Adam" в TensorFlow, который необходим для обновления параметров модели. Он является одним из наиболее популярных и эффективных методов обучения нейронных сетей. Он сочетает в себе лучшие свойства двух других методов — "AdaGrad" и "RMSProp" — и находит индивидуальные темпы обучения для каждого параметра.

Основные параметры, используемые оптимизатором "Adam":

– "learning rate " задает темп обучения. Правильно подобранный шаг может сильно повлиять на сходимость модели. Слишком большой шаг может привести к нестабильному обучению, тогда как слишком маленький замедлит процесс обучения;

– "beta\_1" и "beta\_2" – экспоненциальные коэффициенты скорости затухания первого и второго моментов градиента соответственно. Они определяют, как быстро модель забывает прошлые градиенты при обновлении весов;

– Параметр "epsilon" – небольшое число, добавленное к знаменателю при вычислении обновлений параметров, чтобы избежать деления на ноль. Это предотвращает ошибки и бесконечные значения в процессе обучения;

– "weight decay" или регулятор весов используется для предотвращения переобучения модели путем добавления штрафа за большие значения весов. Он помогает сделать модель более устойчивой и объединяющей, уменьшая переобучение;

– Параметр "global\_clipnorm" в оптимизаторе "Adam" позволяет ограничить общую норму всех градиентов в вашей модели. Это означает, что если суммарная "длина" всех градиентов в вашей модели превышает заданное значение, то они будут масштабированы таким образом, чтобы их общая норма не превышала это значение.

Далее рассматривается функция "compile()", которая используется для конфигурации процесса обучения модели, а также оценки ее производительности.

Параметры, используемые функцией "Compile()":

– "Optimizer" определяет оптимизатор, который будет использоваться для обучения модели;

– Функция потерь "loss" используется для оценки ошибки модели на каждом шаге обучения;

– "loss\_weights" – веса применяемые к различным компонентам функции потерь, если они имеются;

– "metrics" позволяет определить список метрик, которые будут использоваться для оценки производительности модели в процессе обучения. Например, это может быть точность (accuracy), точность классификации (precision), полнота (recall) и т. д;

– Параметр "weighted\_decay" штрафует функцию потерь для избегания переобучения;

– "steps\_per\_execution" представляет целое число, определяющее, сколько шагов выполнить за одну итерацию. Это позволяет управлять скоростью обучения и эффективностью использования ресурсов.

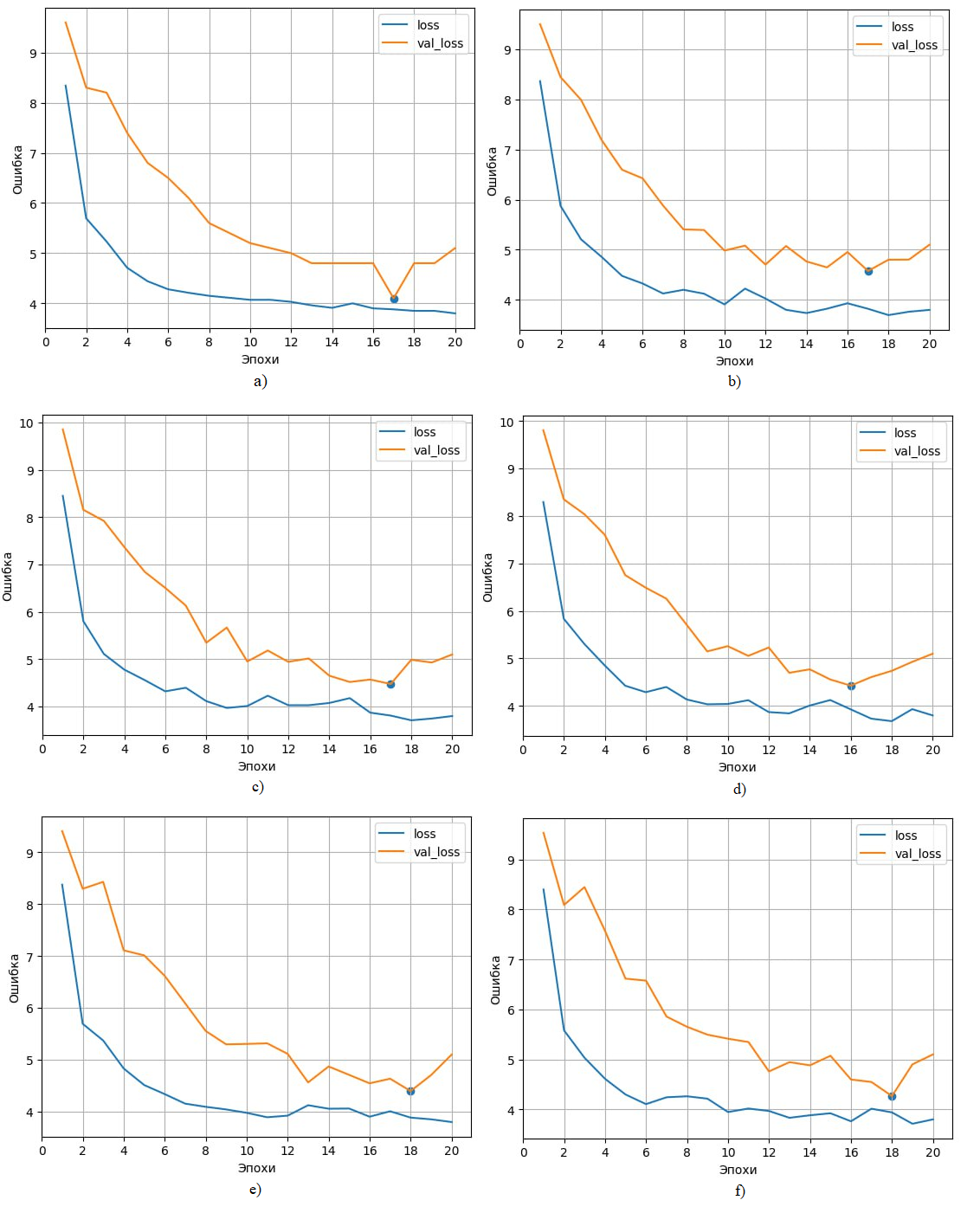
Для более точного описания параметров рассмотренных функций можно обратится к документации библиотеки Keras [14].

Для нахождения набора подходящих гиперпараметров модели версии YOLOv8s было проведено шесть обучений. После каждого обучения модель тестировалось и принималось решение о изменении какого-либо гиперпараметра. Выбор версий и количество итераций обучения обусловлено недостаточным количеством времени, доступного для его проведения. Возможно, при наличии дополнительных ресурсов можно было бы провести более глубокий и тщательный анализ, дополнительно отточить модель или подобрать более подходящие параметры. Наборы гиперпараметров для сравнения представлены в таблице 3.

Таблица 3 – Наборы гиперпараметров для сравнения

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 |
| fpn\_depth | 3 | 4 | 3 | 2 | 5 | 3 |
| learning\_rate | 0.001 | 0.0005 | 0.001 | 0.002 | 0.0001 | 0.001 |
| global\_clipnorm | 1.0 | 0.5 | 1.0 | 1.0 | 0.1 | 0.8 |
| epsilon | 1e-7 | 1e-8 | 1e-7 | 1e-7 | 1e-8 | 1e-7 |
| weight\_decay | 0.0005 | 0.0001 | 0.0001 | 0.001 | 0.0001 | 0.001 |
| box\_loss | smooth\_l1 | giou\_loss | smooth\_l1 | smooth\_l1 | smooth\_l1 | giou\_loss |
| steps\_per\_execution | 1 | 5 | 10 | 1 | 5 | 7 |
| batch size | 32 | 64 | 16 | 128 | 32 | 64 |
| shuffle | True | True | True | True | True | True |

Для выбора лучшего набора гиперпараметров были построены графики зависимостей ошибки, оцененной на тестовых данных "loss" и ошибки, оцененной на валидационных данных "val\_loss" от эпохи обучения. Данные графики представлены на рисунке 9.

Рисунок 9 – a) Зависимости "loss " и " val\_loss" от эпохи обучения для набора 1, b) Зависимости "loss " и " val\_loss" от эпохи обучения для набора 2,

c) Зависимости "loss " и " val\_loss" от эпохи обучения для набора 3,

d) Зависимости "loss " и " val\_loss" от эпохи обучения для набора 4,

e) Зависимости "loss " и " val\_loss" от эпохи обучения для набора 5,

f) Зависимости "loss " и " val\_loss" от эпохи обучения для набора 6

Далее на этих графиках определяется точка минимума графика ошибки работы модели на валидационных данных "val\_loss", набор гиперпараметров, значение "val\_loss" которого в этой точке окажется наименьшим, будет оптимальным для дальнейшего использования.

На рисунке 10 представлена визуализация минимальных значений параметра "val\_loss " у разных комбинаций гиперпараметров.

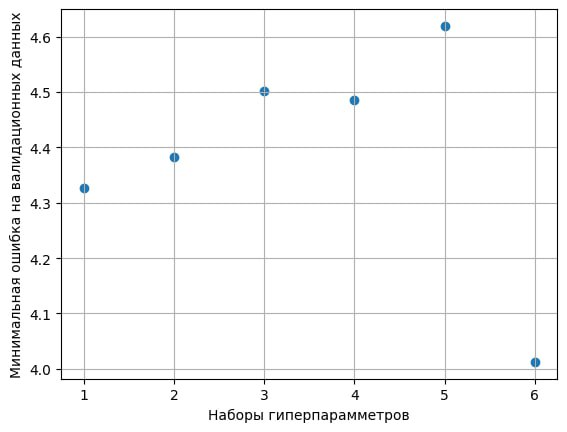


Рисунок 10 – Сравнение минимальных значений параметра "val\_loss"

Как можно заметить на рисунке 10 шестой набор имеет наименьшее значение "val\_loss", это говорит, что данный набор лучше подходит для обучения модели по сравнению с другими, поэтому далее будем пользоваться им.

Таким образом, были рассмотрены исходные данные, используемые в работе, описаны процессы их преобразования в требуемый формат. Проведен обзор гиперпараметров, используемых в обучении модели и выполнен подбор их наиболее подходящих значений для модели YOLOv8s.

# ГЛАВА 3. РАЗРАБОТКА ПРОГРАММНОГО МОДУЛЯ ДЛЯ ОБНАРУЖЕНИЯ РАЗЛИВОВ НЕФТИ НА АЭРОФОТОСНИМКАХ

## 3.1 Обучение модели

Для подбора оптимальной версии модели было проведено обучение трех версий – "S", "M" и "L". Для каждой из которых были рассчитаны метрики качества.

Для обучения модели используется функция "fit()", описанная в листинге 5, для которого необходимо указать:

– Данные обучения (**x** и **y** - входные признаки и целевые переменные), передаваемые как Numpy-массивы или тензоры TensorFlow;

– Размер пакета "batch size" устанавливает размер пакеты входных данных. Чем больше размер пакета, тем более стабильное обновление градиентов, но требуются большие вычислительные ресурсы. Поэтому значение стоит подбирать в зависимости от объема доступной памяти GPU;

– Количество эпох, которое определяет сколько раз весь тренировочный набор данных будет пропущен через модель. От данного параметра зависит степень обученности модели. При недостаточном количестве эпох модель может не успеть изучить все важные особенности данных. Это может привести к недообучению и модель не успеет достичь хороших результатов ни на тренировочном, ни на валидационном наборах данных. Слишком большое количество эпох приводит к переобучению, когда модель начинает слишком хорошо подстраиваться под тренировочные данные;

– "сallbacks" – список обратных вызовов, используемых для мониторинга процесса обучения и принятия решений на его основе. В нем указываются метрики, которые необходимы для расчета качества модели.

Для достижения лучшей производительности в определенных сценариях может потребоваться настройка оставшихся параметров модели [14]. Если эти параметры не указаны, то будут использованы значения по умолчанию, определенные в функции "fit()". Обычно эти значения по умолчанию выбираются таким образом, чтобы обеспечить общее удобство использования и неплохие результаты для большинства случаев.

Обучение модели для обнаружения разливов нефти на аэрофотоснимках начинается с вызова "yolo.fit()", где в качестве аргументов передаются тренировочный набор данных "train\_ds" и валидационный набор "val\_ds". Обучение проходило в течение 20 эпох, что означает, что модель тренировалась на всем тренировочном наборе данных 20 раз. Уровень вывода информации установлен на 1, что обеспечивает отображение прогресса обучения после каждой эпохи.

Валидационный набор данных передан в аргументе "validation\_data". В качестве обратного вызова используется "EvaluateCOCOMetricsCallback", который выполняет оценку метрик модели на валидационном наборе данных и сохраняет обученную модель в файл по указанному пути. В результате, каждая эпоха обучения завершается оценкой текущей версии модели на валидационном наборе, что позволяет отслеживать её производительность и сохранять промежуточные результаты.

yolo.fit(

    train\_ds,

    val\_ds,

    epochs=20,

Shuffle= 1

    verbose=1,

    callbacks=[EvaluateCOCOMetricsCallback(val\_ds, "C:/Users/zaits/OneDrive/ /Диплом/my\_model\_s\_20\_1.keras")],)

Листинг – Функция "fit()"

## 3.2 Визуализация результатов

Для визуализации результатов на валидационной выборке для проверки работы модели была использована функция "visualize\_detections()", (листинг 6). Здесь происходит получение предсказанных детекций "detections" с помощью метода "predict" модели на валидационном наборе данных. Далее эти детекции объединяются в словарь "y\_pred\_dict", содержащий координаты ограничивающих рамок "boxes", классы объектов "classes", количество обнаруженных объектов "num\_detections" и уверенность модели в обнаружении каждого объекта "confidence".

Затем происходит извлечение истинных ограничивающих рамок "true\_boxes" и классов "true\_classes" из валидационного набора данных. Предсказанные ограничивающие рамки "pred\_boxes" и классы "pred\_classes" преобразуются в тензоры для визуализации.

def visualize\_detections(model, dataset, bounding\_box\_format, class\_mapping):

images, y\_true = next(iter(dataset.take(1)))

detections = model.Predict(images)

y\_pred\_dict = {

'boxes': detections['boxes'],

'classes': detections['classes'],

'num\_detections': detections['num\_detections'],

'confidence': detections['confidence']

}

print(y\_true.keys())

y\_true\_dict = {

'boxes': tf.zeros((0, 4), dtype=tf.float32),

'classes': tf.zeros((0,), dtype=tf.int32)

}

pred\_boxes = tf.convert\_to\_tensor(y\_pred\_dict['boxes'])

pred\_classes = y\_pred\_dict['classes']

y\_pred\_confidence = y\_pred\_dict['confidence']

plot\_bounding\_box\_gallery(

images,

value\_range=(0, 255),

bounding\_box\_format=bounding\_box\_format,

y\_true=y\_true\_dict,

y\_pred=y\_pred\_dict,

scale=5,

rows=1,

cols=1,

show=True,

font\_scale=0.8,

class\_mapping=class\_mapping,

)

return pred\_boxes, pred\_classes, y\_pred\_confidence

Листинг – Функция "visualize\_detections()"

Функция "plot\_bounding\_box\_gallery()" используется для отображения изображений и соответствующих им ограничивающих рамок как для истинных из "y\_true\_dict", так и для предсказанных из "y\_pred\_dict" детекций. Эта визуализация позволяет наглядно сравнить результаты детекции модели с истинными данными. Результаты визуализации с прогнозом разливов нефти представлены на рисунке 11.

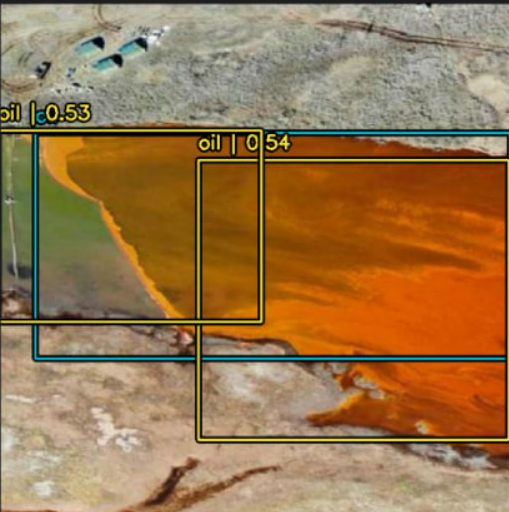


Рисунок 11 – Визуализация результатов прогнозирования

## 3.3 Метрики качества

**3.3.1 Пересечение через объединение (Iou)**

Intersection over Union (IoU) – один из способов количественной оценки правильности работы модели. IoU рассчитывается по формуле 2 как отношение площади пересечения истинной и предсказанной областей к общей их площади. Схематичное изображение представлено на рисунке 12.

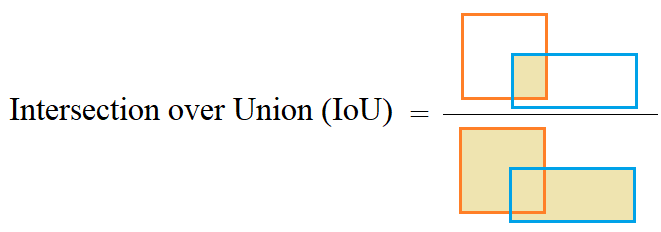


Рисунок 1 – IoU

Если площадь прогнозируемого прямоугольника будет больше, то увеличится знаменатель, что уменьшит значение пересечения через объединение (IoU). Высокое значение IoU говорит о хорошей работе модели, низкое о плохой.

Все возможные значения пересечения через объединение (IoU) находятся в диапазоне от 0 до 1, где 0 означает отсутствие перекрытия, а 1 говорит об абсолютно точном определении расположения объекта [4].

**3.3.2** **Точность (Precision)**

Еще один критерий оценки работы модели – precision, показывающий, насколько точна созданная модель. В отличие от площади пересечения (IOU), точность (precision) оценивает модель более глобально, учитывая количество истинных областей, которые удалось предсказать правильно, и количество неправильно предсказанных областей. Оценка правильности предсказания ведется по ранее рассмотренному параметру – пересечение через объединение (IoU), которому задается пороговое значение. Если вычисленное значение пересечения через объединение (IoU) окажется выше заданного, то считается, что область предсказана правильно, если меньше или равно, то ложно.

Точность (precision) рассчитывается по формуле 3 как отношение количества истинных областей, которые удалось предсказать, к общему числу предсказанных областей.

где – количество правильно предсказанных областей, (False Positives) – количество неправильно предсказанных областей, то есть областей, у которых IoU пересечения с истинными областями меньше порогового значения.

Когда точность (precision) равна 1, все предсказанные области совпали с истинными, то есть не было предсказано лишних или не точных рамок. Если точность (precision) равна нулю, это означает, что все предсказанные области предсказаны неправильно и лежат вдали от истинных.

**3.3.3** **Полнота (Recall)**

Recall **–** метрика, необходимая для оценки полноты модели. Рассчитывается по формуле 4, как отношение правильно предсказанных областей к общему количеству истинных областей.

где – количество правильно предсказанных областей, FN (FalseNegative) – количество истинных областей, которые не удалось предсказать, то есть которые не пересеклись достаточным образом с предсказанными областями.

Если полнота (recall) равна 1, то это будет говорить о том, что все истинные области удалось предсказать, а если полнота (recall) будет равна нулю, то это будет значить, что ни одна из истинных областей не была предсказана.

Полнота (recall) и точность (precision) неотъемлемо дополняют друг друга. При низком пороге уверенности в предсказаниях, сгенерированных рамок будет слишком много, они будут разбросаны в случайном порядке и, скорее всего, закроют все истинные области. Тогда полнота будет равна единице, а precision близок к нулю.

С повышением порога уверенности количество предсказанных областей будет уменьшаться, и все они будут лежать близко к истинным. Тогда точность будет стремиться к единице, а полнота к нулю.

Таким образом, получается, что нельзя оценивать модель по точности или полноте в отдельности. Необходимо рассматривать их как комплексный критерий оценки, что подводит нас к новой метрике – average precision [6].

**3.3.4 Средняя точность (Average Precision)**

Average precision (AP) – более сложная метрика, которая учитывает изменения точности (precision) и полноты (recall) при варьировании порога уверенности. Average precision рассчитывается следующим образом:

1. Устанавливается значение порога уверенности модели в предсказании.
2. Выбирается пороговое значение, рассчитывается площадь пересечения (IoU) предсказаний и определяется их корректность – все области разделяются на предсказанные и не предсказанные (рассчитывается ТР, FP, FN).
3. Рассчитываются точность и полнота. В соответствии с полученными значениями ставится точка на кривой recall - precision.
4. Проводится несколько итераций по пунктам 1 – 3, в процессе изменяется значение порога уверенности.
5. В результате получается график изменения точности и полноты в зависимости друг от друга, изображенный на рисунке 13.
6. С помощью численных методов считается интеграл полученной таблично заданной функции, который и является значением средней точности (average precision).

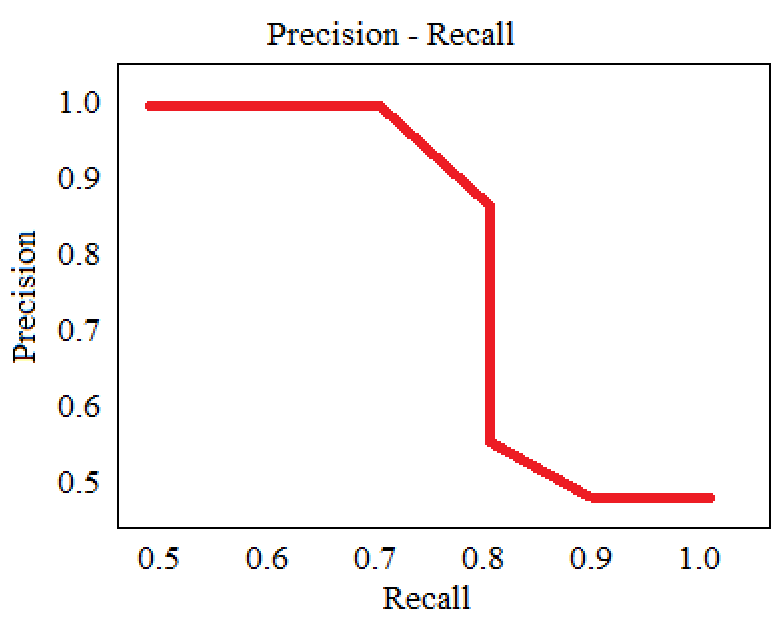


Рисунок 1 – Средняя точность (average precision)

**3.3.5 Сравнение моделей**

Визуализации сравнения моделей YOLOv8 по метрикам точности (precision), полноте (recall), площади пересечения (IoU) и средней точности (AP) представлены на рисунках 14 – 16.

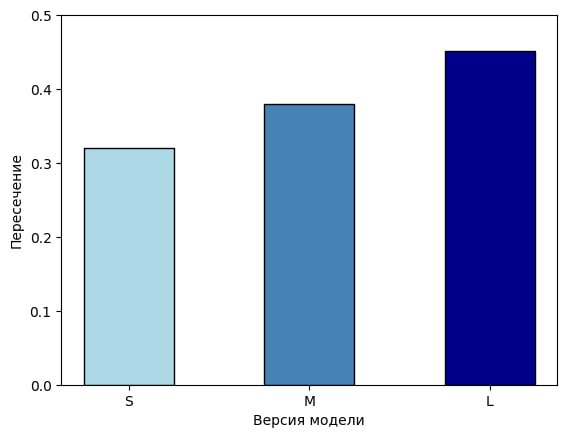


Рисунок 14 – Результаты версий моделей YOLOv8 по метрике площадь пересечение по объединению (IOU)

Из рисунка 14 можно сделать вывод о том, что модель "L" показывает лучший результат (0,452).

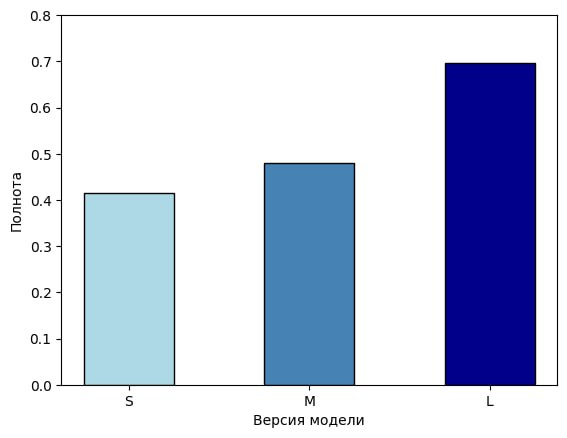


Рисунок 15 – Результаты версий моделей YOLOv8 по полноте (recall)

Анализируя рисунок 15, можно сказать, что по точности модель "L" является ведущей, поскольку имеет наибольшее значение (0,697)

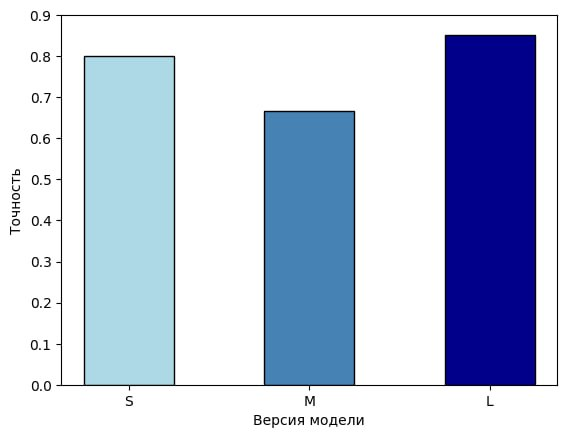


Рисунок 16 – Результаты версий моделей YOLOv8 по точности (precision)

Как видно на рисунке Х модель "L" выигрывает у других по параметру precision.

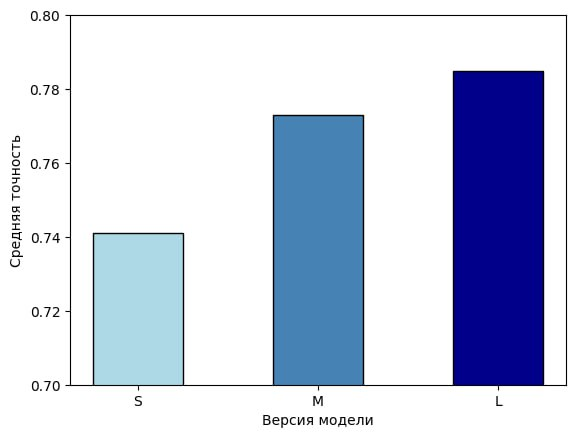


Рисунок 17 – Визуализация сравнения моделей разных версий YOLOv8 по параметру средней точности (AP)

На рисунке 17 нельзя не заметить, что модель "L" показывает наилучшее значение параметра средней точности (АР).

Графики precision – recall, по которым считались значения AP представлены на рисунках 18-20.

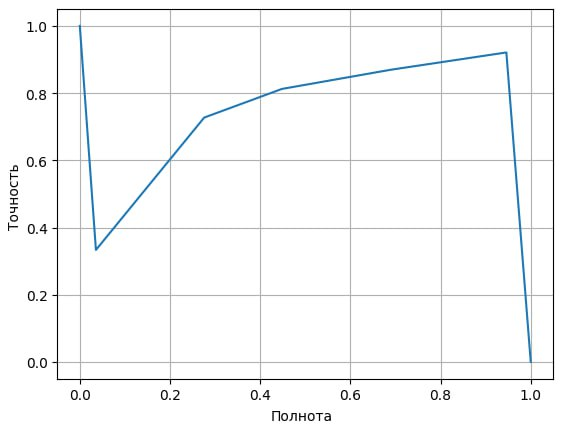


Рисунок 18 – График precision – recall для модели "S"

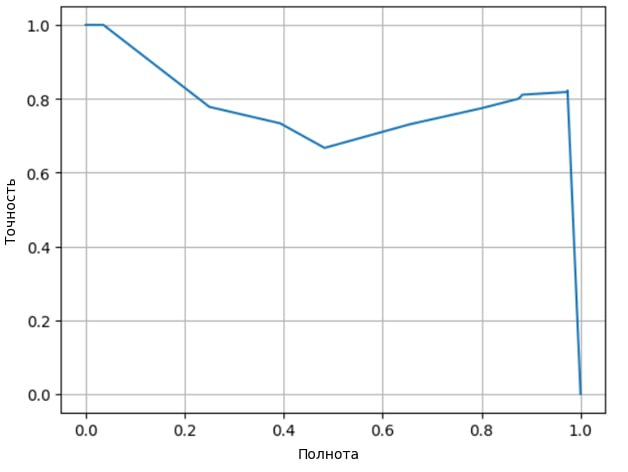


Рисунок 19 – График precision – recall для модели M

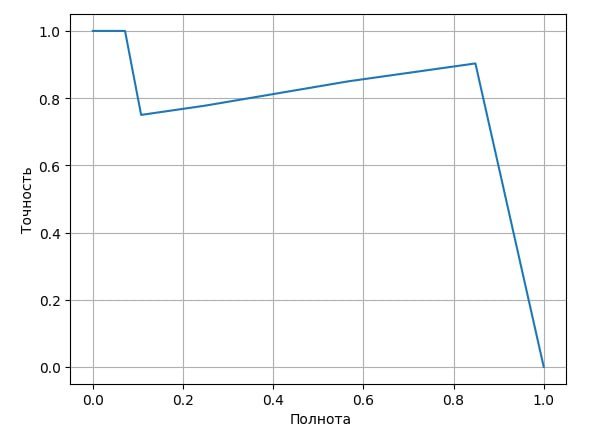


Рисунок 20 – График precision – recall для модели "L"

В таблице 4 представлено сравнение трех модели ("S", "M" и "L") по рассмотренным метрикам качества, для оценки их эффективности. Это позволит выбрать наиболее подходящую модель для решения задачи анализа разливов нефти на аэрофотоснимках.

Таблица 4 – Сравнение YOLOv8s, YOLOv8m, YOLOv8l по метрикам precision, recall, IOU и AP

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Точность (Precision) | Полнота (Recall) | Площадь пересечения (IOU) | Средняя точность (AP) |
| Модель "S" | 0,800 | 0,414 | 0,320 | 0,741 |
| Модель "M" | 0,667 | 0,480 | 0,380 | 0,773 |
| Модель "L" | 0,852 | 0,697 | 0,452 | 0,785 |

Из таблицы 4 видно, что модель "L" превосходит другие модели по всем критериям оценивания. Это объясняется сложной и тщательной тренировкой модели, путем использования большего набора параметров. Модель YOLOv8l имеет более глубокую и широкую архитектуру по сравнению с YOLOv8s и YOLOv8m. Увеличение числа параметров позволяет модели запоминать и различать более тонкие и детализированные параметры в данных. Это особенно важно для сложных задач распознавания объектов, где модели с меньшим количеством параметров могут быть недостаточно мощными для хорошего обобщения. Это позволяет модели лучше захватывать сложные зависимости и особенности в данных.

Однако, такая производительность требует значительно больше вычислительных ресурсов. Обработка изображения моделью "L" занимает примерно в два раза больше времени, чем моделью M, и в три раза больше, чем моделью "S".

Разница же между результатами моделей "S" и "M" оказалась не такой большой и при выборе между ними, учитывая разницу в требуемых вычислительных мощностях, лучше использовать модель "S".

Для обучения моделей было использовано устройство с дискретной видеокартой Nvidia GeForce MX450 и процессором Intel(R) Core(TM) i5-1135G7.

## 3.4 Разработка графического интерфейса

**3.4.1 Анализ библиотек для разработки графического интерфейса**

При разработке графического интерфейса важно выбрать наиболее подходящую библиотеку для решения поставленной задачи. Так как это напрямую влияет на производительность, удобство использования и внешний вид приложения. При выборе библиотеки для реализации графического интерфейса были рассмотрены различные варианты, включающие Tkinter, PyQt, WxPython, Kivy, FLTK и PySide. Проведено сравнение этих библиотек.

Tkinter является стандартной библиотекой GUI для Python, которая поставляется вместе с Python. Представляет большой выбор как встроенных, так и внешних модулей создания интерактивных элементов. Tkinter способна работать на всех основных операционных системах благодаря тому, что она слабо привязана к техническим аспектам какой-либо операционной системы. Также она легка в использовании для создания простых приложении, имеет широкое комьюнити и понятную документацию. Однако, приложения, созданные с помощью Tkinter, выглядят устаревшими. При создании крупного проекта можно столкнуться с проблемами ограниченной функциональности и производимости, и получить ошибки при переносимости на другие платформы [15].

WxPython – это обертка Python для wxWidgets, (который написан на C++), что делает его довольно мощным. Он имеет вид, схожий с нативным, обширный набор инструментов для создания сложных графических интерфейсов. Поддерживается на основных платформах и взаимодействует с другими python-библиотеками, легок в использовании. Но ограничен в создании современных анимаций и мультимедиа. Может иметь большой объем из-за множества зависимостей. [17].

PyQt – это набор библиотек Python для фреймворка Qt. Он поддерживает Qt v6. PyQt является одним из наиболее мощных и широко используемых GUI-фреймворков и часто используется в индустрии за счет большого разнообразия элементов. В PyQt можно использовать встроенные макеты или создавать свои, настраивать их внешний вид с помощью стилей. Он является событийно-ориентированной моделью, это дает возможность создавать интерактивные приложения. Как и Tkinte, PyQt является кроссплатформенной, многофункциональной и простой в использовании.

PyQt6 имеет несколько недостатков, включая сложность обучения из-за обширных библиотек в применяемых проектах и создание крупных исполняемых файлов из-за множества зависимостей, что является проблемой при распространении на устройства с недостаточным объемом памяти. Лицензионные ограничения могут потребовать приобретения коммерческой лицензии для выполнения корпоративных проектов. Справочная литература и сообщество программистов, использующих PyQt5, менее обширны, чем у других библиотек, что затрудняет поиск решений. Кроме того, PyQt5 может быть несовместим с некоторыми сторонними инструментами и библиотеками Python, создавая сложности при интеграции.

FLTK – кроссплатформенная библиотека C++ GUI, являющаяся оберткой Python для FLTK. Преимуществами FLTK являются высокая производительность, малый размер приложений, простота использования и интеграции. Однако у FLTK ограниченный набор виджетов и функций по сравнению с другими библиотеками, меньше ресурсов и примеров для обучения, а также меньшее сообщество пользователей. [2].

PySide – еще одна библиотека GUI для Python. Также, как и PyQt, PySide является набором библиотек Python для фреймворка Qt.

PySide и PyQt поддерживают фреймворк Qt, они очень мощные и широко используются для создания сложных приложений с графическим интерфейсом на Python. PySide также обладает широкой функциональностью, включая поддержку множества платформ и современных технологий и стандартов, таких как Qt Quick и QML. Имеет событийную модель, хорошую документацию и развивающееся сообщество разработчиков. Однако PySide менее популярен в сравнении с PyQt, что может затруднить поиск поддержки и решений для сложных задач и не всегда имеет совместимость с более старыми версиями Qt и некоторыми его функциями. Также возможны ограничения производительности из-за использования обертки над библиотекой на C++ [21].

В результате анализа возможных библиотек для разработки графического интерфейса было принято решение в пользу использования библиотеки PySide.

**3.4.2 Разработка графического интерфейса**

Для удобного взаимодействия с моделями был разработан графический интерфейс, состоящий из нескольких окон с использованием библиотеки PySide6.

Окно авторизации, представленное на рисунке 21, выполнено с полями для ввода логина и пароля, а также с кнопкой для входа.

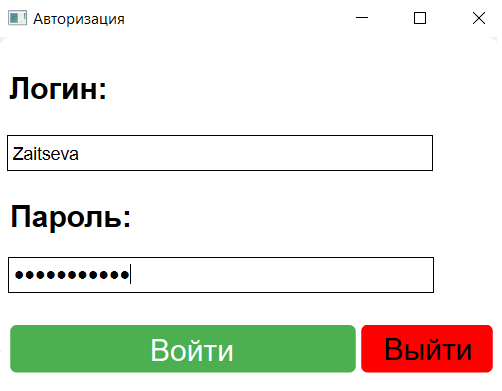


Рисунок 21 – Окно "Авторизация"

Пользователь должен ввести свои учетные данные, и после нажатия кнопки "Войти" приложение проверяет их на корректность. При авторизации происходит взаимодействие с базой данных, в качестве которой использован файл JSON. В нем хранятся: id, логин, пароль, роль в системе, электронная почта каждого пользователя.

Алгоритм взаимодействия с базой данных:

1. Пользователь вводит свои учетные данные.
2. Приложение получает эти учетные данные и отправляет запрос на проверку аутентификации к базе данных.
3. База данных проверяет предоставленные учетные данные на соответствие с данными, хранящимися в своей системе.
4. В случае совпадения учетных данных приложение определяет роль пользователя и предоставляет доступ к соответствующей функциональности или ресурсам.
5. В случае несоответствия учетных данных приложение отображает сообщение об ошибке и предлагает пользователю повторить попытку ввода или проверить корректность учетных данных.

Если учетные данные верны, открывается соответствующее окно для выбора действия (рисунок 22).

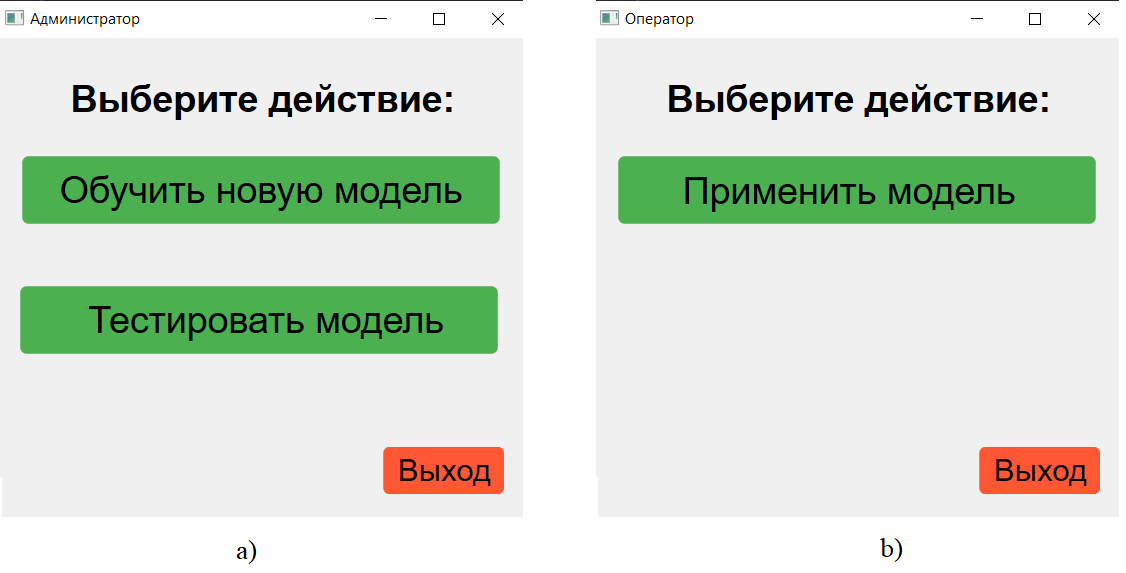


Рисунок 22 – Окно выбора действия: a) для администратора, b) для оператора

По введённым данным, система сама определяет, администратор или оператор выполняет вход в систему. Из окон "Администратор" и "Оператор" можно выйти, нажав кнопку "Выход", расположенную в правом нижнем углу каждого окна. Данная кнопка требует подтверждения действий, поэтому после ее нажатия открывается окно "Подтверждение", изображенное на рисунке 23.

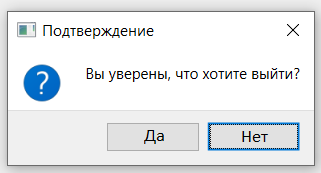


Рисунок 23 – Подтверждение действия "Выход"

Если данные в окне авторизации введены некорректно, то отображается одно из сообщений об ошибке. Рисунок 24.

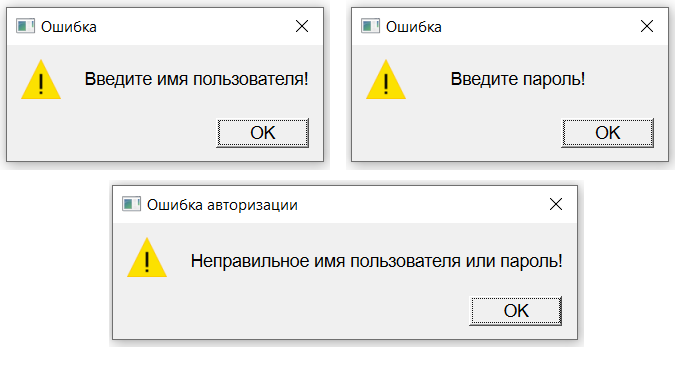


Рисунок 2 – Ошибки при авторизации

После открытия главного окна "Администратор", изображенного на рисунке 22(а), можно выбрать один из двух сценариев. Первый – "Обучить новую модель, второй– "Тестировать модель".

После перехода по кнопке "Обучить новую модель" открывается окно "Обучение модели", представленное на рисунке 25, с помощью которого можно настроить параметры будущей модели.

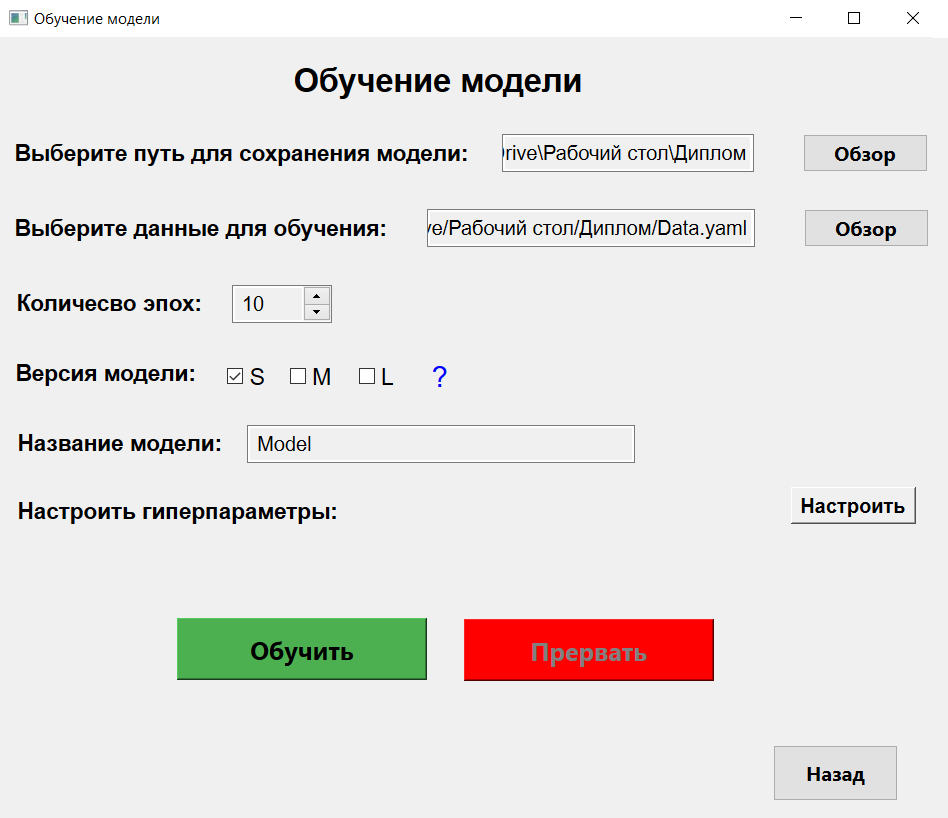


Рисунок 2 – Окно "Обучение модели"

В нем необходимо указать путь к каталогу, где после обучения будет сохранена созданная модель, и путь к данным, на которых она будет обучена. При выборе файлов можно выбрать только те, у к которых формат совпадает с заранее указанным. Ввести название, выбрать одну из трех версий: "S", "M", "L", указать количество эпох для обучения модели. Кнопка "Обучить модель" становится активной только после корректного заполнения всех полей окна. При необходимости можно прервать процесс обучения, нажав соответствующую кнопку. Тогда кнопка "Прервать обучение" — станет неактивной, и снова активируется кнопка "Обучить модель". С помощью элемента "Назад" в левом нижнем углу можно вернуться к окну для выбора действия. При нажатии на кнопку "Настроить" открывается окно, изображенное на рисунке 26 "Настройка гиперпараметра", в котором необходимо настроить гиперпараметры для будущей модели.

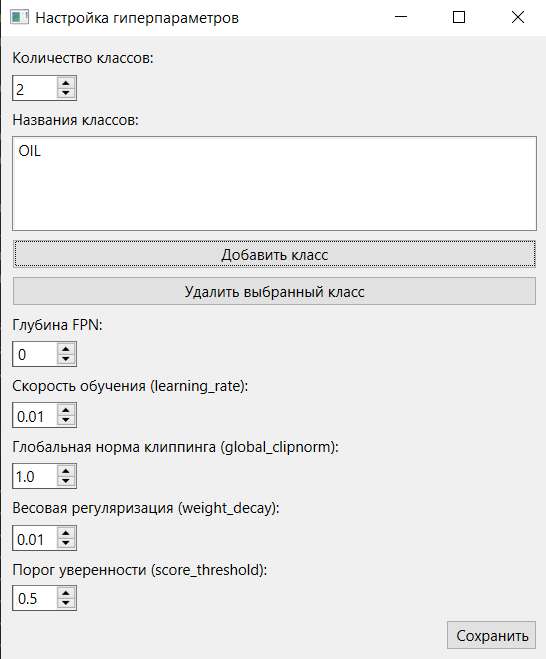


Рисунок 2 – Окно "Настройка параметров"

Для добавления класса необходимо нажать на кнопку "Добавить класс", после чего откроется окно "Добавить класс", представленное на рисунке 27. Для удаления класса необходимо нажать на кнопку "Удалить выбранный класс".

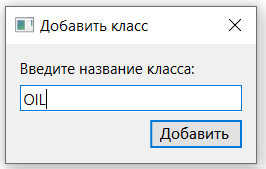


Рисунок 2 – Окно "Добавить класс"

Наведя на синий вопросительный знак, расположенный в окне "Обучить модель", рядом с возможными версиями модели открывается справочное окно "Инструкция для размеров", изображенное на рисунке 28, в котором можно получить дополнительную информацию о возможных вариантах будущей модели.

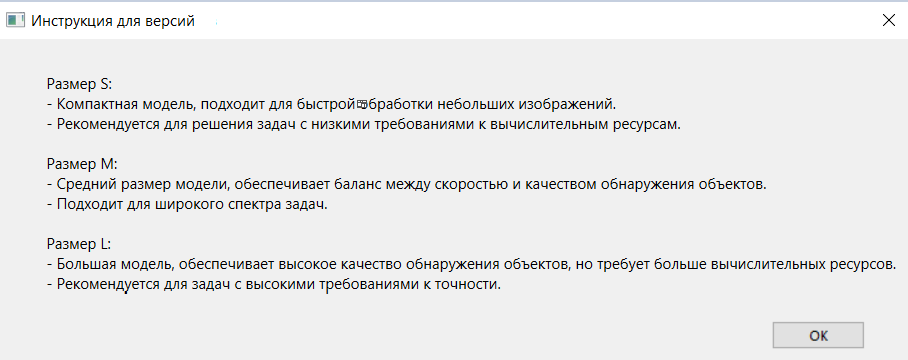
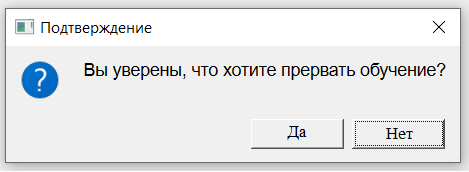


Рисунок 2 – Окно "Инструкция для размеров"

При нажатии кнопки прервать необходимо подтвердить свои действия в окне "Подтверждение прерывания обучения" (рисунок 29).

****Рисунок 29 – Окно "Подтверждение прерывания обучения"

Далее представлены сценарии завершения обучения модели. При успешном обучении появляется окно "Обучение завершено успешно!" (рисунок 30(a)), при сбое, ошибке или прерывании открывается окно "Обучение прервано! " (рисунок 30(b)).

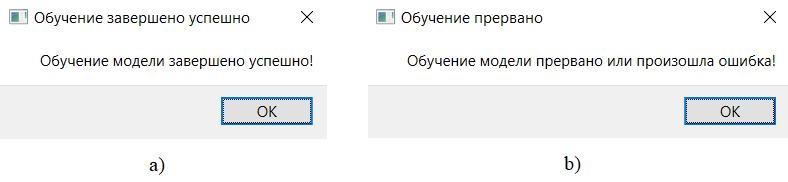


Рисунок 30 – Окно завершение обучения: a) при успешном завершении, b) при сбое, прерывании или ошибке

При выборе сценария для тестирования модели открывается окно "Тестирование модели", изображенное на рисунке 31.

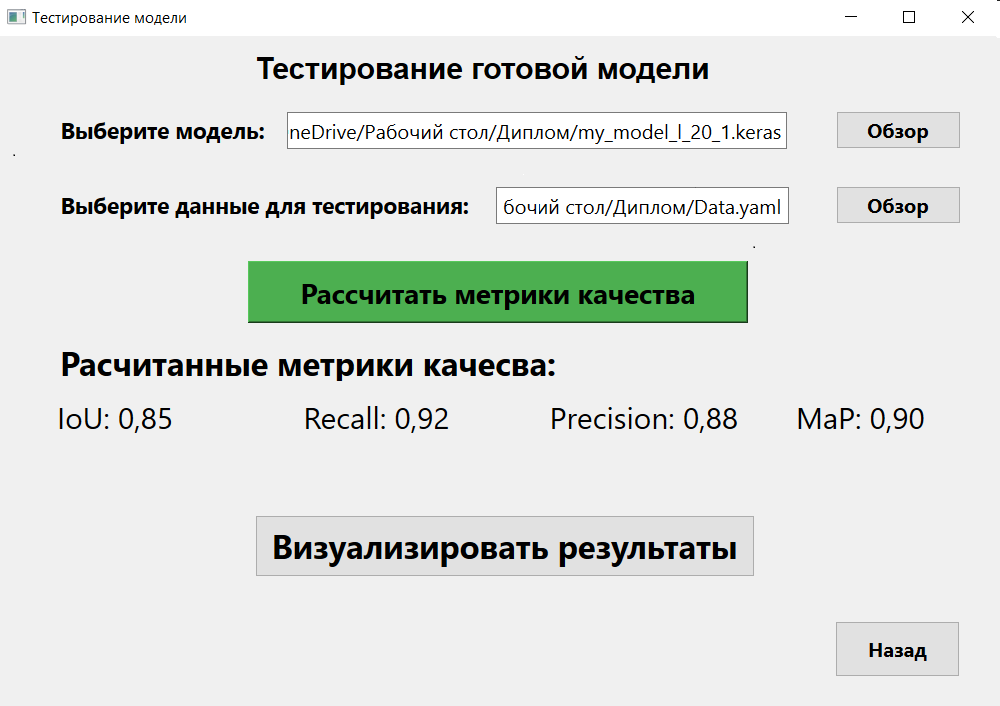


Рисунок 31 – Окно "Тестировать модель"

Администратору следует указать путь к тестируемой модели и каталогу, где собраны данные для ее тестирования. При выборе файлов можно выбрать только те, у к которых формат совпадает с заранее указанным. При нажатии кнопки "Рассчитать метрики качества" происходит расчет и отображение значения метрик пересечения через объединение (IOU), полноты (recall), точности (precision), средней точности (Ap). При выборе кнопки "Визуализировать результаты" открывается окно "Визуализация результатов", представленное на рисунке 32, в котором отображается одно из валидационных изображений с истинными (обозначены голубым цветом) и предсказанными (обозначены желтым цветом) ограничивающими рамками, у которых указан класс детектируемого объекта. Дополнительно, у каждой предсказанной рамки указана вероятность, с которой модель уверена в своем предсказании. Кнопки "Предыдущее изображение" и "Следующее изображение" используются для перехода к другим фотографиям валидационного набора.

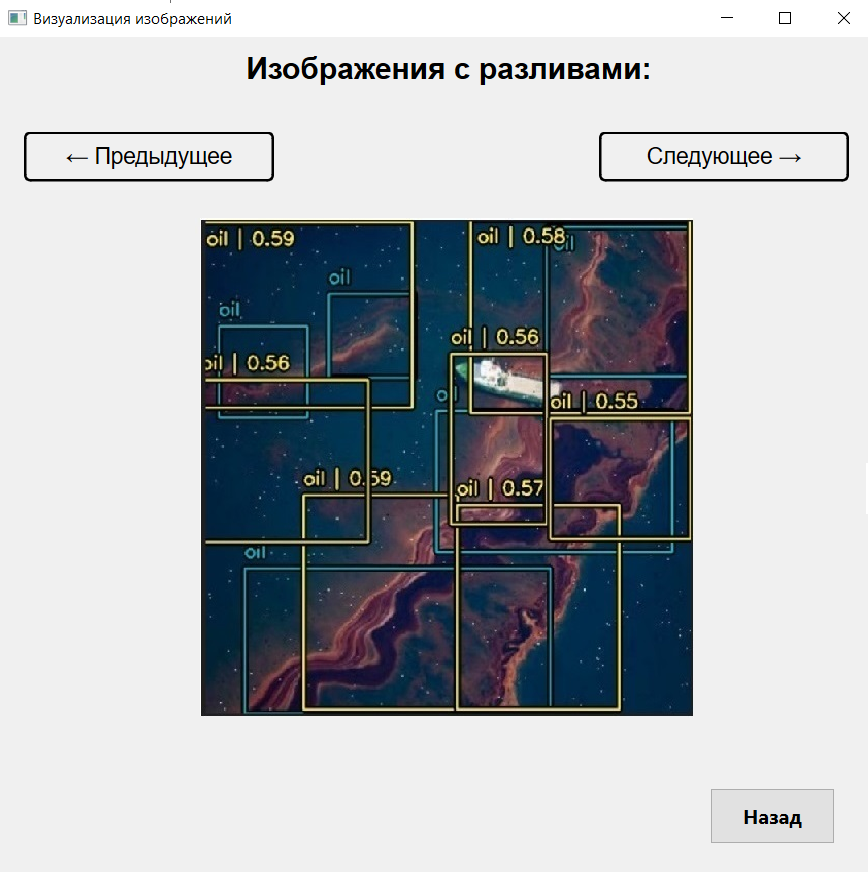


Рисунок 32– Окно "Визуализация результатов"

В окне "Оператор", представленном на рисунке 22(b), присутствует сценарий – "Применить готовую модель". Он активируется после перехода по кнопке "Применить модель", которая предназначена для применения готовой модели к полученным изображениям. Окно "Применение готовой модели" изображено на рисунке 33. Кнопка "Назад" будет неактивна до тех пор, пока не будет нажата кнопка "Сохранить результаты", она в свою очередь не будет активна, пока не будет задан корректный путь для сохранения результатов.

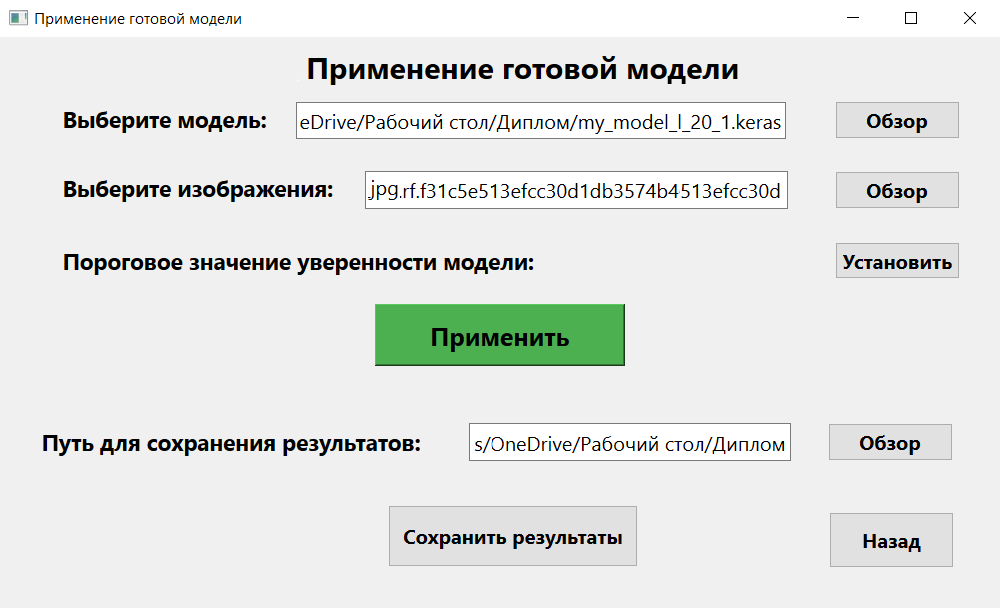


Рисунок 33 – Окно "Применение готовой модели"

В нем необходимо указать путь к готовой модели и каталогу, где собраны изображения для подачи в модель. При выборе файлов можно выбрать только те, у к которых формат совпадает с заранее указанным. Нажав на кнопку "Установить", пользователь открывает окно, показанное на рисунке 34, в котором можно задать значение порога уверенности модели. Порог уверенности представляет собой минимальный уровень достоверности (вероятности) модели в правильности обнаружения объектов. Модель будет отображать предсказания, у которых порог уверенности превышает установленный.

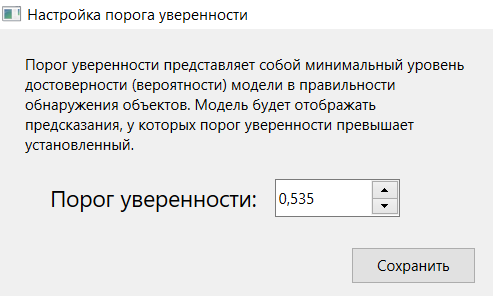


Рисунок 34 – Окно "Настройка порога уверенности"

При нажатии кнопки "Сохранить" появляется окно, требующее подтверждения заданного уровня точности, нажатием на кнопку "Ок". Данное окно изображено на рисунке 35.

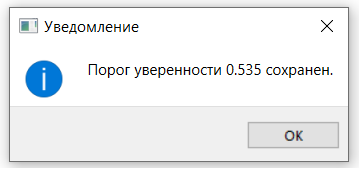


Рисунок 35 – Окно для подтверждения порога уверенности

Кнопка "Применить" становится активной только после выбора применяемой модели и анализируемых изображений. При наличии обнаруженных разливов, при нажатии на кнопку "Применить" открывается окно "Результаты обнаружения разливов", представленное на рисунке 36. В нем отображается таблица с названием файлов изображений, на которых были обнаружены разливы. Нажав на кнопку "копировать", в буфер обмена сохраняется путь изображения с разливом.

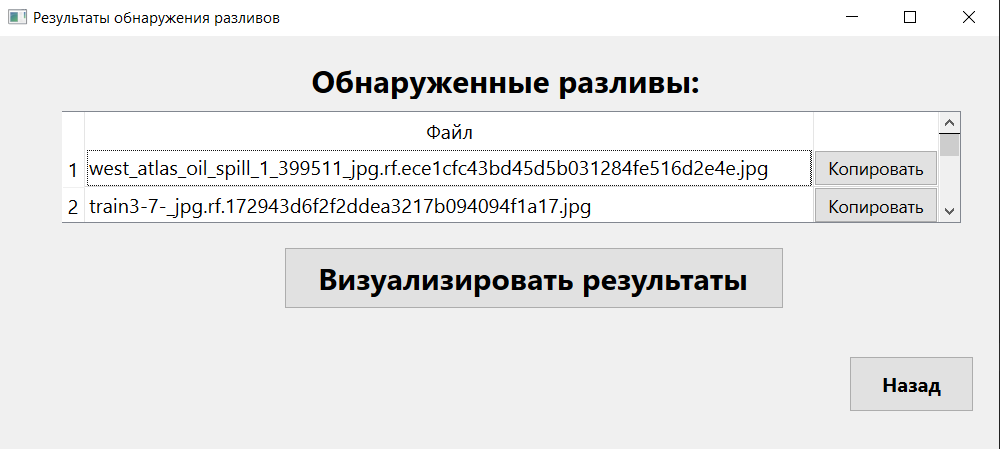


Рисунок 36 – Окно "Результаты обнаружения разливов"

Кнопка "Визуализировать результаты" открывает окно "Визуализация результатов", изображенное на рисунке 37, где происходит отображение одного из тестовых изображение с предсказанными ограничивающими рамками, с соответствующими уверенностями модели в своем прогнозе.



Рисунок 37 – Окно "Визуализация результатов" при обнаружении разливов системой

Присутствует поле с указанным адресом каталога отображаемого изображения, который можно копировать или вставить с помощью специально предусмотренных кнопок. Результатом является изображение с спрогнозированной рамкой, ограничивающей разлив. Кнопки "Предыдущее изображение" и "Следующее изображение" используются для перехода к другим фотографиям валидационного набора. Как и в других окнах, присутствует возможность вернуться к главному окну, нажав кнопку "Назад".

Внизу окна расположена кнопка "Обнаружены разливы", которую оператор обязан нажать для начала процесса ликвидации утечки нефти. Для подтверждения нажатия выскакивает окно "Предупреждение" (рисунок 38).

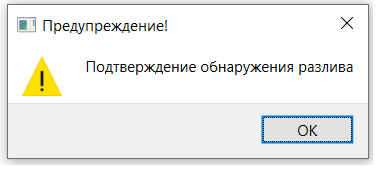


Рисунок 38 – Окно для подтверждения обнаружения разливов

Если после нажатия кнопки "Применить", расположенной в окне "Применение готовой модели", представленное на рисунке 33, используемая модель не обнаружила разливов, то открывается окно с сообщением "Разливы не обнаружены", в котором оператор также может проверить все входные изображения на наличие разливов и в случае обнаружения сообщить его наличии (рисунок 39).

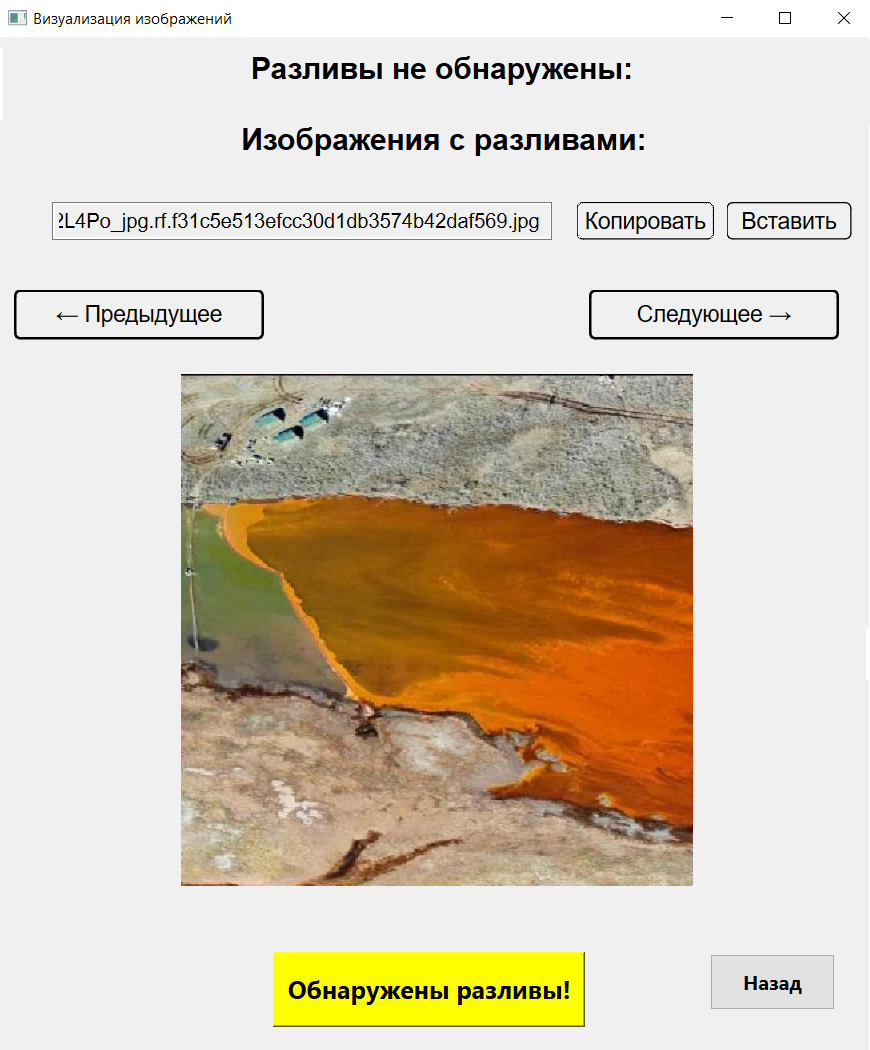


Рисунок 39 – Окно "Результаты обнаружения разливов" при не обнаружении разливов системой

В данной главе были рассмотрены подбор параметров для обучения и само обучение модели. Разработан графический интерфейс, включающий функции обучения и тестирования новых моделей для администратора и функцию использования готовых модели для оператора.

**ЗАКЛЮЧЕНИЕ**

В рамках данной работы была проведена разработка программного обеспечения для автоматизации процесса обнаружения разливов нефти на аэрофотоснимках. При этом были выполнены следующие задачи:

1. Проведен анализ существующих методов компьютерного зрения для обнаружения разливов нефти на аэрофотоснимках.
2. Проведен анализ существующих нейросетевых архитектур, применяемых для детекции объектов на изображениях и выбор подходящей нейронные сети для решения поставленной задачи.
3. Выполнена предобработка набора данных аэрофотоснимков с разметкой разливов нефти и без нее.
4. Выполнено построение модели для обнаружения разливов нефти на аэрофотоснимках.
5. Проведен анализ полученных результатов и оценка качества построенной модели.
6. Разработан графический интерфейс.

# Список литературы

1. Brief summary of YOLOv8 model structure / [электронный ресурс] // Github : [сайт]. — URL: https://github.com/ultralytics/ultralytics/issues/189 (дата обращения: 10.12.2023).
2. Fast Light Toolkit / Иван Голиков [электронный ресурс] // FLTK : [сайт]. — URL: https://www.fltk.org/ (дата обращения: 10.05.2024).
3. Fine-tuning Faster R-CNN on Sea Rescue Dataset / Jayku Maran [Электронный ресурс]. — URL: https://learnopencv.com/fine-tuning-faster-r-cnn. (дата обращения: 29.10.2023).
4. Intersection over Union (IoU) in Object Detection & Segmentation / Kukil [электронный ресурс] // Learn OpenCV : [сайт]. — URL: https://learnopencv.com/intersection-over-union-iou-in-object-detection-and-segmentation/ (дата обращения: 28.04.2024).
5. oilspill\_v2 Computer Vision Project / [электронный ресурс] // Roboflow : [сайт]. — URL: https://universe.roboflow.com/oilspill-dataset/oilspill\_v2 (дата обращения: 20.09.2023).
6. Precision и recall. Как они соотносятся с порогом принятия решений? / Никита Осовский [электронный ресурс] // Хабр : [сайт]. — URL: https://habr.com/ru/articles/661119/ (дата обращения: 30.04.2024).
7. RetinaNet Model Architecture / [электронный ресурс] // OpenGenus : [сайт]. — URL: https://iq.opengenus.org/ (дата обращения: 29.10.2023).
8. Rich feature hierarchies for accurate object detection and semantic segmentation / Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik [электронная статья]. — URL: https://openaccess.thecvf.com/content\_cvpr\_2014/papers/Girshick\_Rich\_Feature\_Hierarchies\_2014\_CVPR\_paper.pdf:(дата обращения: 19.05.2024).
9. SSD: Single Shot MultiBox Detector / Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, Alexander C. Berg [электронный ресурс] // SpringerLink : [сайт]. — URL: https://link.springer.com/chapter/10.1007/978-3-319-46448-0\_2.(дата обращения: 29.10.2023).
10. YOLOv8 Architecture: A Deep Dive into its Architecture / Jane Torres [электронный ресурс] // Yolo8.org : [сайт]. — URL: https://yolov8.org/yolov8-architecture/ (дата обращения: 01.05.2024).
11. You Only Look Once: Unified, Real-Time Object Detection / Joseph Redmon, Santosh Divvala, Ross Girshick , Ali Farhadi [электронная статья] // — URL:https://www.cvfoundation.org/openaccess/content\_cvpr\_2016/papers/Redmon\_You\_Only\_Look\_CVPR\_2016\_paper.pdf. (дата обращения: 01.05.2024).
12. Аварийные разливы нефти: средства локализации и методы ликвидации SECURITY SYSTEMS SECURITY AND SAFETY: [сайт]. — URL: http://secuteck.ru/articles2/prom\_sec/avariynie\_razlivi (Дата обращения: 17.05.2024).
13. Глубокое обучение // Гудфеллоу Я., Бенджио И., Курвилль А. Г пер. с анг. А. А. Слинкина. – 2-е изд., испр. – М.: ДМК Пресс, 2018. – 652 с.: цв. ил.
14. Документация Keras / [электронный ресурс] // Keras : [сайт]. — URL: https://keras.io/ (дата обращения: 19.02.2024).
15. Документация PyQt / [электронный ресурс] // Python : [сайт]. — URL: https://wiki.python.org/moin/PyQt (дата обращения: 17.04.2024).
16. Документация Tensorflow / [электронный ресурс] // Tensorflow : [сайт]. — URL: https://www.tensorflow.org/?hl=ru (дата обращения: 19.02.2024).
17. Документация wxPython / [электронный ресурс] // wxPython : [сайт]. — URL: https://wxpython.org/ (дата обращения: 17.05.2024).
18. Иван Голиков Сверточная нейронная сеть, часть 1 : структура, топология, функции активации и обучающее множество / Иван Голиков [электронный ресурс] // Хабр: [сайт]. — URL: https://habr.com/ru/articles/348000/ (дата обращения: 04.05.2024).
19. Метод и аппаратура дистанционного обнаружения, распознавания и количественного анализа разливов нефти на морской поверхности / Самков В.М., Товбин Б.С. [электронный ресурс] // ResearchGate : [сайт]. — URL: https://goo.su/5tCpwwc (Дата обращения: 02.05.2024).
20. Обоснование оптических методов контроля нефтяных загрязнений водной поверхности / Мельников Г.С., Самков В.М. [электронный ресурс] // Научная электронная библиотека «КиберЛенинка» : [сайт]. — URL: https://cyberleninka.ru/article/n/obosnovanie-opticheskih-metodov-kontrolya-neftyanyh-zagryazneniy-vodnoy-poverhnosti (Дата обращения: 03.05.2024).
21. Работа с PySide / [электронный ресурс] // Хабр : [сайт]. — URL: https://habr.com/ru/articles/134457/ (дата обращения: 17.05.2024).
22. Радиолокационная система обнаружения нефтяных пятен / Генкин В.А., Лошаков В.Р. Переслегин С.В. Тимофеев В.П. [электронная статья]. – URL: https://patents.s3.yandex.net/RU53445U1\_20060510.pdf (дата обращения: 01.05.2024).
23. Разлив нефти / [электронный ресурс] // Горная промышленность: [сайт]. — URL: http://mining-prom.ru/toplivodob/neft/razliv-nefti/dpage5/ (дата обращения: 02.05.2024).