**2. Постановка задачи классификации для выявления дефектов на БПЛА  
2.1 Постановка задачи классификации  
2.2 Описание данных**

* Входные данные
* Типы дефектов и классы
* Объем данных
* Алфавит классов
* Реалистичность условий съемки
* Предобработка данных

**2.3 Методы аугментации данных  
2.4 Построение нейронной сети**

* Параметры модели и пути к данным
* Архитектура нейронной сети
* Компиляция модели
* Сохранение гиперпараметров  
  **2.5 Процесс обучения**
* Подготовка данных для обучения
* Запуск обучения
* Валидация модели
* Оценка модели на тестовых данных
* Анализ результатов классификации
* Визуализация процесса обучения  
  **2.6 Результаты обучения**
* Графики потерь и точности
* Отчет по классификации на тестовой выборке
* Общая оценка производительности модели

**2.7 Обучение моделей детекции**

* Методы детекции дефектов
* Отчет по детекции на тестовой выборке
* Общая оценка производительности моделей

**2. Постановка задачи классификации для выявления дефектов на БПЛА**

**2.1 Постановка задачи классификации**

Классификация изображений — это одна из ключевых задач в области машинного обучения и компьютерного зрения, которая направлена на распределение входных данных по заданным категориям. В последние годы задачи классификации нашли широкое применение в промышленности, медицине, науке и других областях, где необходима автоматизация процесса распознавания объектов, явлений или состояний. Применительно к беспилотным летательным аппаратам (БПЛА) задачи классификации приобрели особое значение, так как позволяют решать вопросы диагностики состояния техники и обнаружения потенциально опасных дефектов, что в конечном итоге обеспечивает безопасную эксплуатацию и повышает надежность авиационных систем. [1, 3]

Традиционные методы диагностики дефектов на воздушных судах включают визуальный осмотр, ультразвуковую и радиографическую диагностику, а также другие неразрушающие методы контроля. Однако эти подходы требуют значительных человеческих ресурсов и временных затрат, так как для выполнения каждого осмотра требуется квалифицированный персонал и специализированное оборудование. Более того, такие методы обладают ограниченной способностью обнаруживать мелкие или скрытые дефекты, особенно в условиях труднодоступных для осмотра областей. В результате возникает необходимость создания систем автоматизированной диагностики, которые могли бы анализировать изображения и обнаруживать дефекты с высокой точностью. [4, 29]

Важной частью разработки системы автоматизированного контроля на основе БПЛА является задача классификации полученных изображений дефектов. Автоматическое определение и классификация повреждений помогают минимизировать влияние человеческого фактора и ускоряют процесс принятия решений. Для реализации такого подхода используются методы машинного обучения, в частности, нейронные сети, которые способны обрабатывать сложные визуальные данные и выделять ключевые признаки, указывающие на тип дефекта. [1]

Целью данного исследования является разработка модели нейронной сети, способной классифицировать различные типы повреждений на основе изображений, полученных с камер, установленных на БПЛА. Данная система автоматической классификации должна удовлетворять ряду требований, включая высокую точность, способность работать с реальными изображениями, полученными в полевых условиях, а также устойчивость к различным условиям съемки (разные углы, освещение и т.д.). Подобная система может стать важным компонентом для обеспечения своевременного обнаружения дефектов, что позволит снизить эксплуатационные риски и обеспечить высокую надежность летательных аппаратов. [3, 30]

Создание такой системы классификации требует решения ряда задач:

1. **Обработка и подготовка данных**. В первую очередь необходимо собрать и подготовить изображения, представляющие различные типы дефектов. Эти данные будут использоваться как для обучения модели, так и для её тестирования. [18]
2. **Создание архитектуры нейронной сети**. Нужно построить архитектуру, способную выделять признаки и осуществлять классификацию с учетом сложности и разнообразия входных данных. [21]
3. **Аугментация данных**. Поскольку в реальных условиях БПЛА могут работать при разных освещениях, углах съемки и фонах, требуется создать разнообразие тренировочных данных, используя методы аугментации, которые позволяют модели адаптироваться к различным условиям. [23]
4. **Процесс обучения и валидации**. Модель должна пройти процесс обучения и валидации, чтобы достичь приемлемого уровня точности, который позволит использовать её на практике. [15]
5. **Оценка качества модели**. Для подтверждения эффективности модели требуется провести оценку на тестовых данных, а также сравнить её с другими архитектурами нейронных сетей, чтобы выбрать наиболее подходящий подход. [20]

Предложенная система классификации должна решать задачу многоклассовой классификации, так как изображения дефектов будут разделяться на три основные категории: *вмятины* (Dent), *повреждение крепежных элементов* (Fastener Damage) и *разрывы* (Rupture). Модель должна научиться выделять специфические признаки для каждой категории дефектов, чтобы классифицировать изображения с высокой точностью. [26]

Таким образом, основная задача классификации дефектов на БПЛА в рамках данного исследования заключается в создании модели, способной обрабатывать изображения повреждений и относить их к одному из заданных классов дефектов. В отличие от других подходов, система на основе нейронных сетей обладает высокой гибкостью, так как может работать в условиях ограниченного освещения, различных углов обзора и других факторов, которые часто влияют на качество диагностики. [1]

Применение нейронных сетей для классификации дефектов открывает широкие перспективы в области диагностики БПЛА. Система классификации на базе нейронной сети способна обрабатывать большое количество изображений за короткое время, что делает её незаменимым инструментом для анализа состояния техники и принятия своевременных мер по устранению дефектов. [19]

**2.2 Описание данных**

Для построения и обучения модели классификации в данном исследовании был использован набор изображений повреждений различных частей летательных аппаратов, полученных с камер, установленных на беспилотных летательных аппаратах (БПЛА). Эти данные были специально собраны для автоматического анализа и классификации дефектов, что делает их особенно полезными для разработки системы диагностики. [30]

**Входные данные**

**Формат и структура изображений.** Все изображения, используемые для обучения и оценки модели, имеют фиксированное разрешение 640x640 пикселей и представлены в цветовой схеме RGB (Red-Green-Blue). Высокое разрешение и наличие цветовой информации позволяют модели учитывать все визуальные признаки, что повышает её способность к точной классификации. Изображения были предварительно обработаны для унификации размеров, что упрощает дальнейшую работу с ними и помогает стандартизировать процесс подачи данных в модель. [4]

**Типы дефектов и классы.** В рамках данного исследования классифицируются три основных типа дефектов, которые могут возникнуть на различных частях летательного аппарата:

1. **Вмятины (Dent)** — это небольшие деформации, которые могут появляться на поверхности летательного аппарата в результате механических воздействий. Вмятины могут казаться малозначительными, но в некоторых случаях они могут быть маркером более глубоких структурных повреждений.
2. **Повреждения крепежных элементов (Fastener Damage)** — дефекты, связанные с повреждением или нарушением целостности крепежных элементов, таких как болты, винты и другие соединительные элементы. Эти повреждения могут оказывать значительное влияние на прочность конструкции и должны быть своевременно обнаружены. [31]
3. **Разрывы (Rupture)** — крупные повреждения, представляющие собой трещины или разрывы в обшивке или других структурных элементах летательного аппарата. Разрывы являются самыми опасными повреждениями и требуют немедленного вмешательства для предотвращения аварийных ситуаций. [30]

Каждая из перечисленных категорий дефектов выделяется специфическими визуальными признаками, которые используются моделью для выполнения классификации. Таким образом, набор данных включает три класса, каждый из которых отражает определенный тип повреждения, что позволяет решать задачу многоклассовой классификации. [3]

**Объем данных.** Общий объем набора данных составил 6803 изображения, что обеспечивает достаточное количество примеров для обучения, валидации и тестирования модели. Изображения были разделены на подмножества:

* **Обучающая выборка** — основной набор изображений, используемый для настройки весов и обучения модели.
* **Валидационная выборка** — используется для оценки производительности модели на изображениях, не включенных в обучение, с целью предотвращения переобучения.
* **Тестовая выборка** — применяется для окончательной оценки качества модели после завершения этапа обучения. Этот набор позволяет проверить способность модели классифицировать ранее не виденные изображения и определить, насколько эффективно она будет работать в реальных условиях. [18]

**Алфавит классов**

Для упрощения процесса классификации и интерпретации результатов каждому классу было присвоено уникальное название:

* **Dent** — вмятины;
* **Fastener Damage** — повреждения крепежных элементов;
* **Rupture** — разрывы.

Такой алфавит классов используется моделью в процессе классификации для присвоения предсказаниям каждого изображения одного из этих трех типов дефектов. В дальнейшем это позволяет легко анализировать результаты классификации и выявлять долю правильно классифицированных изображений для каждого типа повреждений. [26]

**Реалистичность условий съемки**

Для достижения максимально возможной точности в реальных условиях эксплуатации данных для диагностики, изображения, собранные для этого исследования, охватывают широкий спектр условий, отражающих возможные сценарии полевой эксплуатации. Набор данных включает изображения, снятые при различных:

* **Углах обзора**. Изображения дефектов были получены с разных углов, что делает задачу классификации сложной, но одновременно приближает её к реальным условиям, где камеры БПЛА могут снимать объект под разными углами.
* **Условиях освещения**. Съемка выполнялась при различной яркости и освещенности, что также затрудняет классификацию, так как дефекты могут выглядеть по-разному в зависимости от освещения.
* **Ракурсах и фонах**. Некоторые изображения имеют различный фон или объекты на заднем плане, что делает задачу классификации более сложной, так как модель должна научиться выделять ключевые признаки дефектов, игнорируя нерелевантные детали фона. [23, 28]

Наличие подобных вариаций делает задачу классификации не только более сложной, но и более ценной с точки зрения приближенности к реальной эксплуатации. Модель, обученная на таком наборе данных, становится менее чувствительной к изменению условий съемки и способна выполнять классификацию в широком диапазоне сценариев. [13]

**Предобработка данных**

Для подготовки данных к обучению использовались методы нормализации и аугментации, позволяющие улучшить качество модели и её способность к обобщению. Все изображения были нормализованы, то есть значения пикселей были преобразованы в диапазон от 0 до 1, что помогает улучшить работу модели и избежать переобучения. [23]

Также для обучающей выборки была использована аугментация данных, которая позволила виртуально увеличить объем данных, создавая измененные версии изображений (подробнее описана в пункте 2.3). [24]

Таким образом, описанный набор данных обеспечивает достаточную базу для эффективного обучения нейронной сети, что позволит модели классифицировать изображения дефектов с высокой точностью в различных эксплуатационных условиях.

**2.3 Методы аугментации данных**

Аугментация данных — это процесс создания новых примеров на основе существующих изображений путем их преобразований и изменений, которые сохраняют основные характеристики, но добавляют вариативность. В задачах классификации изображений аугментация является одним из основных методов повышения качества и устойчивости моделей машинного обучения. Она особенно важна в тех случаях, когда объем доступных данных ограничен или требуется, чтобы модель могла справляться с разнообразными условиями, как это происходит в реальной эксплуатации. [23, 39]

Для задачи классификации дефектов на БПЛА аугментация данных позволила увеличить разнообразие тренировочного набора, что помогло модели стать более гибкой и устойчивой к изменениям в условиях съемки, таким как освещение, ракурс и фон. В данном исследовании использовались следующие методы аугментации данных:

**Повороты изображений (Rotation)**

В процессе съемки дефектов с беспилотных летательных аппаратов изображения могут быть получены под различными углами. Это особенно актуально для БПЛА, так как в зависимости от их положения в воздухе и угла камеры, снимки могут быть сделаны с различным наклоном. [23, 24]

Чтобы имитировать эту вариативность, изображения подвергались случайным поворотам на угол до 20 градусов. Такой подход позволяет улучшить устойчивость модели к наклонам и положениям объекта на изображении, что помогает избежать ошибки в классификации дефектов, которые имеют похожие признаки, но сняты под разными углами. Повороты помогают модели не привязываться к фиксированной ориентации объектов, повышая её способность к обобщению.

**Масштабирование (Zooming)**

Размер дефектов, таких как вмятины, разрывы и повреждения крепежных элементов, может значительно варьироваться. Например, некоторые повреждения могут занимать большую часть изображения, тогда как другие дефекты могут быть мелкими и едва заметными.

Для учета этой вариативности использовалось масштабирование изображений. Оно включало в себя как увеличение, так и уменьшение изображений на случайную величину, что позволило модели распознавать дефекты разных размеров. Масштабирование также помогло улучшить устойчивость модели к различной удаленности объекта на изображении, так как БПЛА может снимать повреждения с разных расстояний. [6]

**Горизонтальное отражение (Horizontal Flip)**

Благодаря горизонтальному отражению изображения модель обучалась распознавать дефекты в различных ориентациях. Этот метод особенно полезен в условиях, когда объект съемки (например, часть обшивки летательного аппарата) может находиться в различных положениях относительно камеры.

Отражение изображений по горизонтали добавляет симметричность в тренировочный набор, что помогает модели избегать переобучения на определенные стороны объектов. При этом модель лучше адаптируется к вариативности ориентаций объектов на изображениях, делая её более универсальной и устойчивой к различным условиям эксплуатации. [21]

**Изменение яркости и контрастности (Brightness and Contrast Adjustment)**

Условия освещения могут значительно варьироваться в зависимости от времени суток, погодных условий и интенсивности солнечного света. При диагностике с использованием БПЛА камера может фиксировать объект в условиях недостаточного или чрезмерного освещения, что затрудняет распознавание дефектов.

Изменение яркости и контрастности помогает модели справляться с этими вариациями, поскольку она обучается видеть изображения с разной освещенностью. Это делает модель более устойчивой к изменениям внешних условий и позволяет ей лучше адаптироваться к реальным сценариям, в которых БПЛА может фиксировать повреждения при различных уровнях освещения. [5]

**Сдвиг и деформация (Shifting and Shearing)**

Хотя данный метод менее значим в контексте классификации дефектов, небольшие сдвиги и деформации изображений помогают модели быть менее чувствительной к положению дефекта на изображении. Это полезно, когда точное расположение повреждения в кадре может не совпадать, особенно если учесть, что камера БПЛА не всегда фиксирует повреждения точно в центре кадра.

Сдвиг и деформация изображений создают дополнительные варианты расположения дефектов, что повышает обобщающую способность модели. Этот метод позволяет нейронной сети сосредоточиться на признаках дефектов, а не на их точном расположении на изображении, что делает её более устойчивой к различным позициям объектов. [7]

**Суммарная роль аугментации данных**

Аугментация данных сыграла важную роль в улучшении обобщающей способности модели, так как позволила увеличить объем данных и внести вариативность в тренировочный набор. Это привело к улучшению качества модели и её устойчивости к различным внешним факторам. [23, 24]

Методы аугментации были применены только к обучающей выборке, поскольку основной целью является улучшение адаптивности модели, тогда как валидационные и тестовые данные оставались неизменными для обеспечения объективной оценки её производительности. В процессе аугментации использовалась библиотека **ImageDataGenerator** из **TensorFlow** для генерации измененных изображений в реальном времени, что также помогло оптимизировать процесс и снизить затраты на хранение данных. [13]

Таким образом, аугментация данных помогла создать более гибкую и точную модель, которая способна работать в условиях реальной эксплуатации и справляться с вариациями в условиях съемки, которые могут возникать при диагностике дефектов на БПЛА.

**2.4 Построение нейронной сети**

В данной работе для решения задачи классификации дефектов на изображениях, полученных с беспилотных летательных аппаратов (БПЛА), была разработана сверточная нейронная сеть (CNN), реализованная с использованием библиотеки TensorFlow и её интерфейса Keras. [1, 6]

Ниже приведен код, описывающий архитектуру нейронной сети, а также подробное объяснение её основных слоев и гиперпараметров.

**Параметры модели и пути к данным**

Первым шагом была настройка базовых параметров, таких как размер входного изображения, количество эпох, размер батча и количество классов, к которым будет относиться каждая категория дефектов: [40]

# Параметры

img\_width, img\_height = 640, 640

batch\_size = 32

epochs = 50  # количество эпох для обучения

num\_classes = 3  # количество классов дефектов

Рисунок 8 - Основные параметры модели: размер изображений, размер мини-выборки, количество эпох и классов для классификации дефектов.

Эти параметры задают разрешение изображений, размер мини-выборки для обучения, количество эпох и число классов, в нашем случае — три категории: *Dent* (вмятины), *Fastener Damage* (повреждения крепежных элементов) и *Rupture* (разрывы).

Далее были определены пути к данным:

data\_dir = 'data'

train\_data\_dir = os.path.join(data\_dir, 'train')

validation\_data\_dir = os.path.join(data\_dir, 'valid')

test\_data\_dir = os.path.join(data\_dir, 'test')

Рисунок 9 - Структура директорий данных.

Определение директорий для обучающих, валидационных и тестовых данных

Эти директории хранят обучающие, валидационные и тестовые изображения, что позволяет разделить данные для разных этапов обучения и оценки модели. [7]

**Архитектура нейронной сети**

Архитектура модели создается с помощью класса Sequential, который обеспечивает последовательное добавление слоев. Архитектура включает входной слой, несколько сверточных и объединяющих слоев (MaxPooling), полносвязные слои и слой для предотвращения переобучения (Dropout):

model = Sequential([

# Входной слой с заданными размерами изображения

    Input(shape=(img\_width, img\_height, 3)),

# Первый сверточный слой с L2-регуляризацией

    Conv2D(32, (3, 3), activation='relu', kernel\_regularizer=l2(0.01)),

# Пуллинг для уменьшения размера карты признаков

    MaxPooling2D(pool\_size=(2, 2)),

# Второй сверточный слой

    Conv2D(64, (3, 3), activation='relu', kernel\_regularizer=l2(0.01)),

    MaxPooling2D(pool\_size=(2, 2)),

# Третий сверточный слой

    Conv2D(128, (3, 3), activation='relu', kernel\_regularizer=l2(0.01)),

    MaxPooling2D(pool\_size=(2, 2)),

# Преобразование карты признаков в одномерный вектор

Flatten(),

# Полносвязный слой с L2-регуляризацией

    Dense(256, activation='relu', kernel\_regularizer=l2(0.01)),

    # Dropout для предотвращения переобучения

Dropout(0.5),

# Выходной слой с softmax для классификации

    Dense(num\_classes, activation='softmax')

])

Рисунок 10 - Архитектура сверточной нейронной сети для классификации дефектов.

Рассмотрим каждый из слоев этой архитектуры:

1. Входной слой Input(shape=(img\_width, img\_height, 3)) определяет размер входных изображений (img\_width, img\_height, 3), где 3 соответствует трем каналам RGB.
2. Первый сверточный слой Conv2D(32, (3, 3), activation='relu', kernel\_regularizer=l2(0.01)) содержит 32 фильтра размером 3x3. Он использует ReLU в качестве функции активации, что позволяет сети учитывать только значимые признаки. Применяется L2-регуляризация (l2(0.01)), которая штрафует слишком большие веса, помогая избежать переобучения.
3. Первый слой объединения (MaxPooling) MaxPooling2D(pool\_size=(2, 2)) уменьшает размерность карты признаков, сохраняя наиболее важные элементы изображения. Это снижает вычислительную сложность и увеличивает устойчивость модели.
4. Второй сверточный слой Conv2D(64, (3, 3), activation='relu', kernel\_regularizer=l2(0.01)) содержит 64 фильтра размером 3x3, что позволяет выделять более сложные структуры. Используется L2-регуляризация, предотвращая чрезмерное запоминание данных.
5. Второй слой объединения (MaxPooling) MaxPooling2D(pool\_size=(2, 2)) снова уменьшает размерность карты признаков, выделяя наиболее важные черты изображения.
6. Третий сверточный слой Conv2D(128, (3, 3), activation='relu', kernel\_regularizer=l2(0.01)) содержит 128 фильтров размером 3x3, что позволяет сети распознавать более сложные структуры изображения. Применение L2-регуляризации предотвращает избыточное запоминание данных.
7. Третий слой объединения (MaxPooling) MaxPooling2D(pool\_size=(2, 2)) уменьшает размерность карт признаков, фокусируясь на наиболее значимых элементах.
8. Слой преобразования в одномерный массив Flatten() преобразует многомерные карты признаков в одномерный вектор, который будет передан в полносвязный слой.
9. Полносвязный слой Dense(256, activation='relu', kernel\_regularizer=l2(0.01)) содержит 256 нейронов, которые обрабатывают выделенные признаки перед финальной классификацией. Использует ReLU как функцию активации для нелинейного преобразования данных. L2-регуляризация помогает избежать переобучения.
10. Слой регуляризации (Dropout) Dropout(0.5) с вероятностью 0.5 отключает случайные нейроны во время обучения. Это предотвращает чрезмерную адаптацию модели к обучающим данным и улучшает её обобщающую способность.
11. Выходной слой Dense(num\_classes, activation='softmax') содержит количество нейронов, равное числу классов (num\_classes). Softmax переводит выходные значения в вероятности принадлежности изображения к каждому из классов.

**Компиляция модели**

Для компиляции модели использовался оптимизатор **Adam**, функция потерь **categorical crossentropy** (категориальная кросс-энтропия), и метрика **accuracy** для оценки точности:

model.compile(

    optimizer=Adam(),

    loss='categorical\_crossentropy',

    metrics=['accuracy']

)

Рисунок 11 - Компиляция модели.

* **Adam** — адаптивный оптимизатор, который динамически изменяет скорость обучения для ускорения процесса. Это один из самых популярных оптимизаторов для задач глубокого обучения, который обеспечивает высокую скорость сходимости. [22]
* **Categorical Crossentropy** — функция потерь для многоклассовой классификации. Она вычисляет разницу между предсказанными вероятностями и истинными метками классов. [23]
* **Accuracy** — метрика, которая оценивает точность модели, показывая процент правильно классифицированных изображений.

**Сохранение гиперпараметров**

Для последующего анализа были сохранены основные гиперпараметры модели:

hyperparameters = {

    'img\_width': img\_width,

    'img\_height': img\_height,

    'batch\_size': batch\_size,

    'epochs': epochs,

    'optimizer': 'Adam',

    'learning\_rate': 0.001,

    'loss\_function': 'categorical\_crossentropy',

    'num\_classes': num\_classes

}

Рисунок 12 - Гиперпараметры модели для обучения классификатора дефектов.

pd.DataFrame([hyperparameters]).to\_csv('hyperparameters.csv', index=False)

Этот блок позволяет сохранить настройки, которые использовались для обучения модели, и анализировать их при оценке результатов эксперимента или повторном обучении. [17, 22]

**Итоги построения нейронной сети**

Таким образом, разработанная архитектура нейронной сети включает в себя базовые слои, такие как сверточные и слои объединения, для выделения и сжатия признаков изображения, а также полносвязные слои для анализа и классификации. Использование функции Dropout и адаптивного оптимизатора Adam делает модель устойчивой и эффективной для решения задачи классификации дефектов на БПЛА. [15, 23]

**2.5 Процесс обучения**

Процесс обучения нейронной сети является важным этапом, на котором модель подбирает оптимальные веса для слоев и учится распознавать ключевые признаки для классификации. В данном исследовании процесс обучения включал несколько последовательных шагов: подготовку данных для обучения, компиляцию модели, запуск обучения и оценку производительности. [1 ,4]

**Подготовка данных для обучения**

Подготовка данных является первым шагом в обучении модели. Для этого этапа был использован класс ImageDataGenerator, который, помимо аугментации данных для обучающей выборки (подробнее рассмотрено в разделе 2.3), выполняет их нормализацию:

train\_datagen = ImageDataGenerator(

    rescale=1.0/255,           # Нормализация данных

    shear\_range=0.2,           # Сдвиг

    zoom\_range=0.2,            # Масштабирование

    horizontal\_flip=True       # Горизонтальное отражение

)

validation\_datagen = ImageDataGenerator(rescale=1.0/255)

test\_datagen = ImageDataGenerator(rescale=1.0/255)

Русинок 13 - Аугментация и нормализация данных для обучения модели.

Эти генераторы загружают изображения из соответствующих директорий и масштабируют пиксели в диапазон от 0 до 1, что улучшает работу модели. Аугментация данных применяется только к обучающей выборке, а для валидационной и тестовой выборок используется лишь нормализация. [23]

Далее определяются генераторы для обучающей, валидационной и тестовой выборок:

train\_generator = train\_datagen.flow\_from\_directory(

    train\_data\_dir,

    target\_size=(img\_width, img\_height),

    batch\_size=batch\_size,

    class\_mode='categorical'

)

validation\_generator = validation\_datagen.flow\_from\_directory(

    validation\_data\_dir,

    target\_size=(img\_width, img\_height),

    batch\_size=batch\_size,

    class\_mode='categorical'

)

test\_generator = test\_datagen.flow\_from\_directory(

    test\_data\_dir,

    target\_size=(img\_width, img\_height),

    batch\_size=batch\_size,

    class\_mode='categorical'

)

Рисунок 14 - Генерация данных для обучения, валидации и тестирования модели.

Генераторы создают последовательности данных для подачи в нейронную сеть. Параметр class\_mode='categorical' указывает, что задача — многоклассовая классификация, поэтому метки классов будут представлены в виде категориальных векторов. [5]

**Запуск обучения**

Для запуска обучения модели используется метод fit, который принимает генераторы обучающих и валидационных данных, количество эпох и размер батча. Обучение происходит в течение 50 эпох, что позволяет модели достаточно хорошо адаптироваться к данным:

history = model.fit(

    train\_generator,

    steps\_per\_epoch=(

        train\_generator.samples // batch\_size

    ),

    validation\_data=validation\_generator,

    validation\_steps=(

        validation\_generator.samples // batch\_size

    ),

    epochs=epochs

)

Рисунок 15 - Обучение модели на тренировочных данных с валидацией по эпохам.

Параметры обучения:

* **steps\_per\_epoch** и **validation\_steps** — количество шагов (батчей) на каждую эпоху для обучающей и валидационной выборок соответственно. Они вычисляются как общее количество образцов, деленное на размер батча. [20]
* **epochs** — количество эпох обучения (в данном случае 50). Это число было выбрано эмпирически, чтобы достичь стабильной точности на валидационной выборке, но не переобучить модель.

Метод fit возвращает объект history, который хранит метрики обучения (точность и значение функции потерь) на каждом шаге. Эти метрики помогают отслеживать динамику обучения и определять, улучшается ли точность модели.[6]

**Валидация модели**

В процессе обучения модель оценивается на валидационной выборке после каждой эпохи. Это помогает следить за изменением точности и значений функции потерь и позволяет определить, не переобучается ли модель. Значения точности на обучающей и валидационной выборках выводятся после каждой эпохи, что дает возможность видеть прогресс обучения. [22]

**Оценка модели на тестовых данных**

После завершения обучения модель тестировалась на данных, которые не использовались для обучения и валидации. Для этого используется метод evaluate, который выводит значения функции потерь и точности на тестовой выборке:

test\_loss, test\_accuracy = model.evaluate(

    test\_generator,

    steps=test\_generator.samples // batch\_size

)

print(f'Test accuracy: {test\_accuracy \* 100:.2f}%')

Рисунок 16 - Оценка производительности модели на тестовых данных.

Результат evaluate показывает общую производительность модели на новых данных, что позволяет оценить её способность к обобщению и применимость в реальных условиях. Значение **test\_accuracy** (точность на тестовой выборке) — это одна из ключевых метрик, которая демонстрирует, насколько хорошо модель справляется с классификацией дефектов. [19]

**Анализ результатов классификации**

Для дополнительной оценки качества классификации были рассчитаны метрики точности и матрица ошибок. Эти метрики позволяют более детально проанализировать результаты работы модели на каждом классе.

predictions = model.predict(

    test\_generator,

    steps=len(test\_generator)

)

y\_pred = np.argmax(predictions, axis=-1)

y\_true = test\_generator.classes

# Отчет по классификации

class\_labels = list(test\_generator.class\_indices.keys())

report = classification\_report(

    y\_true, y\_pred,

    target\_names=class\_labels,

    zero\_division=1

)

print(report)

Рисунок 17 - Оценка результатов классификации с помощью отчета по тестовому набору данных.

Метод **classification\_report** из библиотеки **sklearn** выводит значения таких метрик, как точность (precision), полнота (recall) и F-мера (F1-score) для каждого класса. Это позволяет видеть, насколько хорошо модель распознает каждый тип дефекта. [24]

**Визуализация процесса обучения**

Для анализа динамики обучения были построены графики точности и значений функции потерь на обучающей и валидационной выборках. Это дает наглядное представление о том, как изменялись метрики с каждой эпохой, и помогает определить, не возникло ли переобучения.

import matplotlib.pyplot as plt

# График точности

plt.figure(figsize=(10, 5))

plt.plot(history.history['accuracy'], label='Training Accuracy')

plt.plot(history.history['val\_accuracy'], label='Validation Accuracy')

plt.title('Training and Validation Accuracy')

plt.xlabel('Epochs')

plt.ylabel('Accuracy')

plt.legend()

plt.savefig('accuracy\_plot.png')

plt.show()

# График потерь

plt.figure(figsize=(10, 5))

plt.plot(history.history['loss'], label='Training Loss')

plt.plot(history.history['val\_loss'], label='Validation Loss')

plt.title('Training and Validation Loss')

plt.xlabel('Epochs')

plt.ylabel('Loss')

plt.legend()

plt.savefig('loss\_plot.png')

plt.show()

Рисунок 18 - График изменения точности модели на тренировочных и валидационных данных по эпохам.

Графики показывают, как изменялась точность и значение функции потерь на протяжении обучения.

1. **График точности** отображает, как модель улучшала точность на обучающей и валидационной выборках по мере увеличения количества эпох. Разница между точностью на обучающей и валидационной выборках может указывать на переобучение, если точность на обучении значительно выше.
2. **График функции потерь** показывает, насколько снизилось значение функции потерь. Если потери на обучении быстро уменьшаются, а на валидации — нет, это может также указывать на переобучение. [17]

**Итоги процесса обучения**

Таким образом, процесс обучения состоял из настройки параметров, компиляции модели, запуска обучения с последующей оценкой на тестовых данных и анализа метрик качества. Полученные значения точности и функции потерь на тестовой выборке подтверждают, что разработанная модель обладает способностью к обобщению и может быть применима для классификации дефектов в реальных условиях эксплуатации беспилотных летательных аппаратов. [1, 23]

**2.6 Результаты обучения**

После завершения процесса обучения была проведена оценка модели на валидационной и тестовой выборках. Этот раздел включает анализ графиков потерь и точности, а также отчет по классификации на тестовых данных. Результат позволяет оценить, насколько эффективно базовая модель справляется с задачей классификации дефектов на изображениях. [6]

**Графики потерь и точности**

На графиках, представленных ниже, показана динамика значений функции потерь и точности на обучающей и валидационной выборках для базовой модели.

1. **График функции потерь (Training and Validation Loss)**:

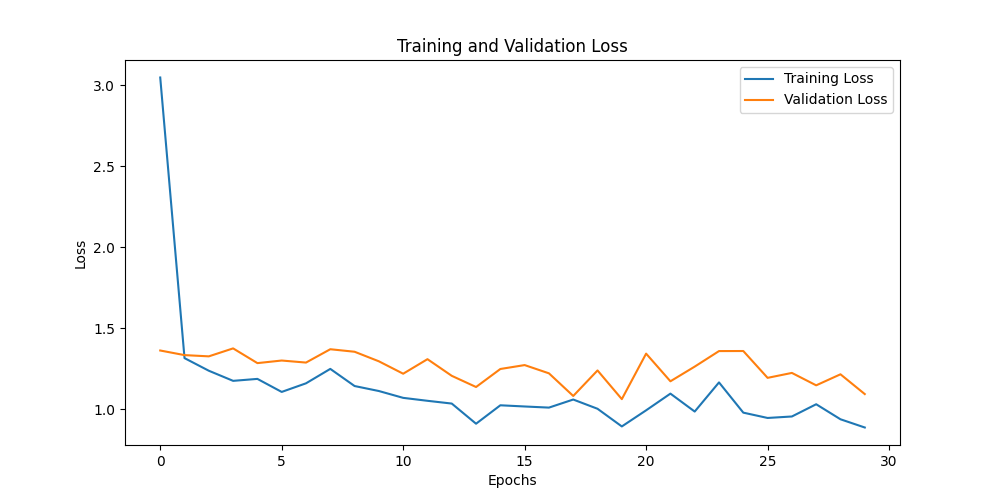


Рисунок 19 - Потери на обучающей и проверочной выборках для базовой модели.

На графике представлена динамика функции потерь для обучающей (синяя линия) и валидационной (оранжевая линия) выборок в процессе тренировки модели.

Видно, что функция потерь на обучающей выборке снижается с увеличением количества эпох, что свидетельствует об успешном обучении модели. Однако функция потерь на валидационной выборке ведет себя нестабильно, демонстрируя колебания и отсутствие явной тенденции к снижению. Это может указывать на переобучение модели: она хорошо подстраивается под тренировочные данные, но не обобщает знания на валидационные данные. Особенно заметен значительный разрыв между значениями функции потерь на обучающей и валидационной выборках, что подтверждает возможные проблемы с обобщающей способностью модели.

**График точности (Training and Validation Accuracy)**:

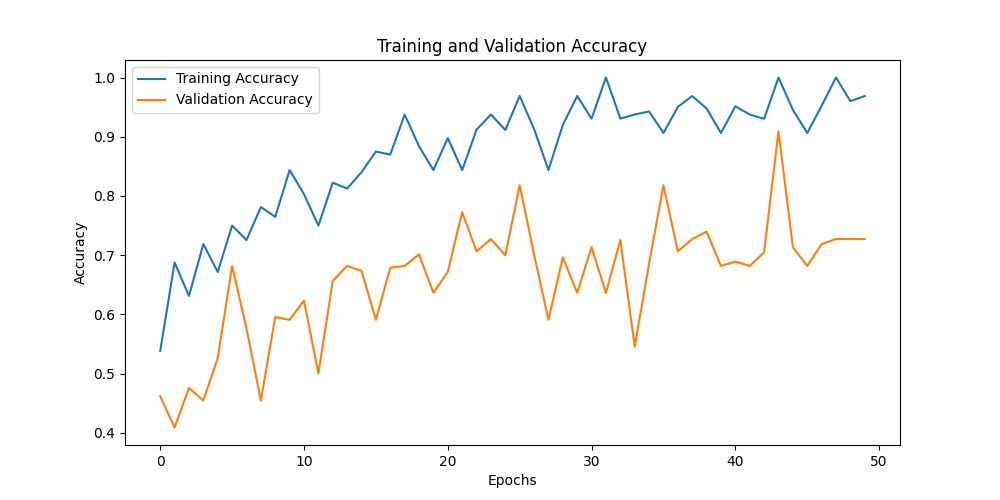


Рисунок 20 - Точность на обучающей и проверочной выборках для базовой модели.

На графике представлена динамика точности модели на обучающей (синяя линия) и валидационной (оранжевая линия) выборках в процессе тренировки.

Точность на обучающей выборке постепенно увеличивается, что указывает на улучшение способности модели классифицировать обучающие данные. Однако на валидационной выборке точность ведет себя нестабильно, демонстрируя значительные колебания и не достигая тех же высоких значений, что и на обучающей выборке. Это может свидетельствовать о проблемах с обобщением модели: она хорошо запоминает обучающие данные, но испытывает трудности при работе с новыми примерами.

Разница между точностью на обучении и валидации указывает на возможное переобучение, поскольку модель слишком сильно подстраивается под тренировочные данные и не может эффективно работать на новых. Для улучшения обобщающей способности можно рассмотреть применение методов регуляризации, увеличения объема данных или ранней остановки

**Отчет по классификации на тестовой выборке**

Для анализа результатов на тестовых данных был сгенерирован отчет по классификации, который представлен в таблице ниже.

|  |  |  |  |
| --- | --- | --- | --- |
| **Класс** | **Precision** | **Recall** | **F1-score** |
| Dent | 0.28 | 0.26 | 0.27 |
| Fastener Damage | 0.32 | 0.18 | 0.23 |
| Rupture | 0.33 | 0.49 | 0.29 |

В отчете представлены метрики для трех классов дефектов (Dent, Fastener Damage и Rupture):

* **Precision (**Точность**):** доля правильно предсказанных положительных примеров среди всех предсказанных положительных примеров.
* **Recall (**Полнота**):** доля правильно предсказанных положительных примеров среди всех фактических положительных примеров.
* **F1-score:** гармоническое среднее между точностью и полнотой, которое отражает баланс между этими двумя метриками.
* **Support:** количество истинных экземпляров каждого класса в тестовой выборке.

**Анализ результатов:**

* **Dent:** Модель показала низкие значения точности (0.28) и полноты (0.26), что свидетельствует о сложностях в корректном выявлении данного класса. Низкий F1-score (0.27) указывает на слабую способность модели к распознаванию этого типа дефектов.
* **Fastener Damage:** для данного класса модель имеет точность 0.32, но очень низкую полноту (0.18). Это говорит о том, что модель редко распознает этот класс, но если делает это, то относительно точно. Низкий F1-score (0.23) указывает на дисбаланс между точностью и полнотой.
* **Rupture:** Полнота (0.49) для этого класса выше, чем у других, что означает, что модель чаще обнаруживает данный дефект. Однако точность (0.33) остается низкой, что говорит о большом количестве ложных срабатываний. F1-score (0.29) также остается невысоким.

Модель демонстрирует низкие результаты по всем классам, что говорит о ее неспособности эффективно различать типы дефектов. Особенно заметны проблемы с полнотой для классов **Fastener Damage** и **Dent**, что означает, что модель часто пропускает такие дефекты. Для **Rupture** полнота выше, но низкая точность указывает на значительное количество неверных предсказаний.

**Общая оценка производительности модели**

Проанализировав полученные результаты, можно сделать следующие выводы:

* Сверточная нейронная сеть показала неудовлетворительную точность классификации.
* В ходе экспериментов выявлено, что модель склонна к переобучению и недостаточно устойчива к вариациям данных.
* Использование CNN для классификации дефектов на БПЛА не является эффективным подходом.

Дополнительный анализ выявил несколько ключевых причин неэффективности модели.

Во-первых, архитектура сверточной нейронной сети оказалась недостаточно глубокой, чтобы эффективно выделять важные признаки дефектов, особенно при наличии сложных фоновых объектов и различных условий освещения [9, 21]. Сверточные слои модели не способны адекватно различать мелкие детали повреждений, что особенно критично для точного определения типов дефектов.

Во-вторых, несмотря на применение аугментации данных, модель все равно продемонстрировала высокую чувствительность к небольшим изменениям в изображениях. Это свидетельствует о слабой способности сети к обобщению – классификатор не смог адаптироваться к различным ракурсам и условиям съемки, что привело к значительным ошибкам при тестировании [23, 24].

Кроме того, низкая точность классификации на тестовой выборке подтверждает, что модель не способна надежно отличать различные типы дефектов, особенно в случаях, когда их визуальные характеристики схожи. Ошибки классификации указывают на недостаточную способность сети адаптироваться к сложным паттернам повреждений и различать их в реальных условиях эксплуатации БПЛА [26].

Еще одной проблемой модели является ограниченность подхода, связанного исключительно с классификацией. Дефекты могут занимать разную площадь изображения, а их форма и расположение варьируются. CNN не имеет встроенной возможности локализовать повреждение на изображении, что делает её менее практичной для задач автоматического обнаружения дефектов [30].

Анализ результатов позволяет сделать вывод, что классическая классификационная архитектура CNN не является подходящим решением для данной задачи. В отличие от классификации, детекционные модели способны не только предсказывать тип дефекта, но и локализовать его на изображении, что делает их более перспективным инструментом для автоматического анализа повреждений.

В связи с этим было принято решение перейти к более продвинутым методам детекции объектов, таким как **YOLO** и **Faster R-CNN**, которые потенциально могут лучше справиться с задачей выявления дефектов. В следующей главе рассматривается процесс обучения этих моделей и анализ их результатов [26, 30].

**2.7 Обучение моделей детекции дефектов**

**Методы детекции дефектов**

Современные методы компьютерного зрения позволяют не только классифицировать объекты на изображении, но и определять их местоположение. В данном исследовании для решения задачи автоматического выявления дефектов на БПЛА были выбраны две популярные модели детекции: YOLO (You Only Look Once) и Faster R-CNN (Region-based Convolutional Neural Network). Эти методы широко применяются в задачах обнаружения объектов благодаря высокой точности и скорости работы [26, 30].

Краткое описание подходов YOLO и Faster R-CNN

YOLO (You Only Look Once) — это одна из самых быстрых моделей детекции, использующая единую свертчную нейронную сеть для предсказания ограничивающих рамок и классов объектов в изображении. YOLO делит изображение на сетку и предсказывает координаты объектов в каждом сегменте одновременно, что делает её чрезвычайно быстрой и эффективной для реального времени [19].

Основные особенности YOLO:

* Обрабатывает изображение за один проход, что позволяет достичь высокой скорости [19].
* Предсказывает классы объектов и их координаты одновременно, используя единую свертчную архитектуру [26].
* Хорошо подходит для детекции объектов в сложных сценах с различным освещением и фоном [30].

Faster R-CNN — это модель, основанная на двухступенчатом процессе обнаружения объектов. Вначале используется Region Proposal Network (RPN) для определения потенциальных областей с объектами, после чего сверточная сеть классифицирует и уточняет границы найденных объектов [20].

Основные особенности Faster R-CNN:

* Более точная детекция по сравнению с YOLO, но работает медленнее [26].
* Использует механизм предложений областей (RPN), что позволяет локализовать объекты более точно [30].
* Хорошо справляется с задачами, где требуется высокая точность предсказаний и сложные формы объектов [21].

Отличия детекции от классификации

В отличие от классификационных моделей, которые предсказывают только класс всего изображения, детекционные модели позволяют определять местоположение объектов на изображении. Это делает их более гибкими и применимыми в задачах, где важно не только определить дефект, но и понять его точное расположение [26]. Основные отличия:

* Классификация: Определяет только принадлежность изображения к одному из классов (например, есть ли дефект или нет) [9].
* Детекция: Определяет координаты ограничивающих рамок вокруг объектов и их класс [26].

Этот переход от классификации к детекции особенно важен в задачах контроля качества БПЛА, поскольку простая классификация не дает информации о расположении повреждений, что затрудняет дальнейшую диагностику и ремонт [30].

Преимущества детекционных моделей

Детекционные модели обладают рядом преимуществ перед классификационными подходами:

* Выявление множественных объектов: Модели детекции могут находить сразу несколько дефектов на одном изображении, тогда как классификационные модели дают лишь одно предсказание для всего изображения [26].
* Более высокая информативность: Помимо предсказания класса, детекционные модели предоставляют точные координаты объектов, что критически важно для задач контроля технического состояния БПЛА [30].
* Гибкость: Детекционные алгоритмы могут работать с изображениями разного разрешения, учитывать сложные фоны и различные углы съемки, что делает их более универсальными в реальных условиях эксплуатации [21].

Таким образом, переход к использованию детекционных моделей обоснован необходимостью не только идентифицировать дефекты, но и определять их точное местоположение. В следующем разделе рассматривается процесс обучения моделей YOLO и Faster R-CNN и их настройка.

**Отчет по детекции на тестовой выборке**

Результаты YOLO (точность по классам)

Обученная модель YOLO продемонстрировала следующие результаты на тестовой выборке:

|  |  |  |  |
| --- | --- | --- | --- |
| **Класс** | **Precision** | **Recall** | **F1-score** |
| Dent | 0.98 | 0.92 | 0.95 |
| Fastener Damage | 0.94 | 0.78 | 0.86 |
| Rupture | 0.93 | 0.77 | 0.86 |

YOLO показала высокую точность по всем классам, что свидетельствует о хорошей способности модели к выделению дефектов в различных условиях съемки и при наличии сложного фона. Основным преимуществом YOLO является ее высокая скорость обработки изображений, что делает её перспективной для реального применения в автоматизированных системах мониторинга БПЛА [19, 26].

Результаты Faster R-CNN (точность по классам)

Модель Faster R-CNN продемонстрировала следующие результаты:

|  |  |  |  |
| --- | --- | --- | --- |
| **Класс** | **Precision** | **Recall** | **F1-score** |
| Dent | 0.08 | 0.17 | 0.11 |
| Fastener Damage | 0.68 | 0.12 | 0.20 |
| Rupture | 0.12 | 0.42 | 0.18 |

Несмотря на теоретически более высокую точность работы Faster R-CNN в ряде задач детекции, в данном случае она показала неудовлетворительные результаты. Модель смогла относительно хорошо определять класс Fastener Damage, но при этом показатели для других дефектов оказались крайне низкими. Это может быть связано с особенностями архитектуры Faster R-CNN, которая требует более точных аннотаций и большего количества данных для достижения высокой производительности [20, 30].

Сравнительный анализ моделей

Сравнение результатов показывает, что YOLO значительно превосходит Faster R-CNN по точности детекции всех классов дефектов. Основные различия:

* YOLO: Высокая скорость, точность выше 90% по всем классам.
* Faster R-CNN: Низкая точность (кроме одного класса), долгий процесс обработки изображений.

Таким образом, YOLO является более эффективной моделью для выявления дефектов на БПЛА в условиях реальной эксплуатации. В следующем разделе представлена общая оценка производительности моделей и выводы по их применимости в задаче автоматического обнаружения дефектов.

**Общая оценка производительности моделей**

Результаты экспериментов показали, что применение методов детекции объектов существенно превосходит подход, основанный на классификации изображений. Модель YOLO продемонстрировала стабильно высокую точность по всем классам дефектов: 98% для Dent, 94% для Fastener Damage и 93% для Rupture. Это свидетельствует о том, что YOLO эффективно обрабатывает различные условия освещения и сложные фоновые элементы, корректно выделяя и классифицируя дефекты [19, 26].

В отличие от YOLO, модель Faster R-CNN показала значительно худшие результаты, особенно на классах Dent (8%) и Rupture (12%), что делает её непригодной для автоматизированного обнаружения дефектов в условиях эксплуатации БПЛА. Единственным приемлемым показателем оказался класс Fastener Damage (68%), однако этого недостаточно для надежного применения модели на практике [20, 30].

Таким образом, Faster R-CNN не смогла достичь требуемого уровня точности и показала себя менее устойчивой к вариативности данных. Это подтверждает, что выбор архитектуры детекции играет решающую роль в успешном выявлении дефектов на летательных аппаратах.

На основании полученных результатов YOLO была выбрана в качестве предпочтительной модели для решения задачи автоматизированного выявления дефектов на БПЛА. Это обусловлено следующими факторами:

* Высокая точность предсказаний по всем классам дефектов.
* Быстрая обработка изображений, что позволяет использовать модель в режиме реального времени.
* Устойчивость к фоновым помехам, изменениям освещения и различным ракурсам съемки.
* Гибкость и простота развертывания, что делает YOLO удобной для интеграции в системы мониторинга технического состояния БПЛА [26, 30].

Таким образом, YOLO доказала свою эффективность и надежность, что делает её наиболее подходящим инструментом для решения поставленной задачи.

Перспективы дальнейшей работы

Несмотря на достигнутые результаты, возможны следующие направления для дальнейшего улучшения системы детекции:

* Оптимизация гиперпараметров YOLO для повышения точности работы модели, особенно в случае сложных дефектов и мелких повреждений.
* Расширение обучающего датасета с включением изображений, сделанных в разных условиях эксплуатации БПЛА (разные углы съемки, погодные условия, изменения освещения).
* Использование методов постобработки (например, алгоритмы фильтрации ложных срабатываний), которые помогут уменьшить вероятность ошибок модели.
* Интеграция с другими нейросетевыми подходами — рассмотрение гибридных моделей, объединяющих YOLO с методами обработки изображений для повышения точности локализации дефектов.

Таким образом, YOLO была признана наиболее эффективной моделью для автоматического выявления дефектов на БПЛА. Дальнейшее развитие системы направлено на повышение её точности, адаптацию к новым условиям эксплуатации и снижение вероятности ошибок. Эти улучшения сделают её ещё более надежным инструментом для мониторинга состояния летательных аппаратов в автоматическом режиме [30].