МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное
учреждение высшего образования
«Южно-Уральский государственный университет»
(национальный исследовательский университет)
Высшая школа экономики и управления
Кафедра «Информационные технологии в экономике»

РАБОТА ПРОВЕРЕНА
Рецензент директор
ООО «Энерготехсервис»

_____ В.Ю.Кийко

«____»_____ 2021 г.

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой «Информационные технологии в экономике», д.т.н., с.н.с.
_____ Б.М. Суховилов

«____»_____ 2021 г.

ПРИМЕНЕНИЕ ТЕХНОЛОГИЙ МАШИННОГО ОБУЧЕНИЯ И СВЁРТОЧНЫХ НЕЙРОННЫХ СЕТЕЙ ДЛЯ ОПРЕДЕЛЕНИЯ СОСТОЯНИЯ ИЗОЛЯЦИИ ЭЛЕКТРООБОРУДОВАНИЯ

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ–09.03.03.2021.301–310.ВКР

Руководитель, к.т.н., доцент
_____/Е.М. Сартасов/
«____»_____ 2021 г.

Автор проекта
студентка группы ЭУ-582
_____ /П.А. Бондаренко/
«____»_____ 2021 г.

Нормоконтролёр, доцент
_____/Е.А. Конова/
«____»_____ 2021 г.

Челябинск 2021

АННОТАЦИЯ

Бондаренко П.А. Применение технологий машинного обучения и свёрточных нейронных сетей для определения состояния изоляции электрооборудования. – Челябинск: ЮУрГУ, ЭУ-582, 51 с., 23 ил., 3 табл., библиогр. список – 20 наим., 2 прил.

Дипломная работа посвящена созданию инновационного продукта в области диагностики электрооборудования.

Цель дипломной работы – разработка системы определения механических и электрических повреждений подвесных изоляторов для контроля технического состояния воздушных линий электропередач и электрооборудования подстанций с возможностью применения беспилотных летательных аппаратов.

В первой главе выбрана актуальная тема, исследована предметная область и рассмотрены системы, использующие технологию распознавания образов.

Во второй главе смоделирована система распознавания состояния изоляции электрооборудования, сформулированы функциональные и нефункциональные требования к ней, а также спроектирован прототип интерфейса приложения.

В третьей главе выполнена подготовка обучающего набора, реализован алгоритм анализа изображений и обучена нейронная сеть.

В четвёртой главе проведено исследование инвестиционной эффективности проекта и рассчитана его экономическая окупаемость.

В результате работы был сформулирован новый подход к развитию системы диагностики с помощью БПЛА, направленный на предупреждение возникновения аварийных ситуаций.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Воздушные линии (ВЛ) электропередачи напряжением 35 кВ и выше являются основными в системах передачи электроэнергии. И поэтому дефекты и неисправности элементов ВЛ требуют немедленной локализации и устранения. При внезапном выходе из строя одного из таких элементов, для возобновления его работоспособности применяется аварийно-восстановительное обслуживание. При этом в межремонтный период проведение диагностических мероприятий не предусмотрено. Такой способ обслуживания характеризуется крайней непредсказуемостью возникновения аварийных процессов, а также значительным ущербом потребителям вследствие внезапного перерыва электроснабжения.

Существующие методы для диагностики таких элементов ВЛ, как изоляторы, предохранители, муфты и т.д., как правило, являются трудоемкими, обладают повышенной опасностью и требуют отключения оборудования от напряжения. Кроме того, результаты осмотров не могут считаться исчерпывающей диагностической информацией, так как наряду с видимыми дефектами конструкции зачастую имеют скрытые дефекты.

В настоящее время существует запрос на обслуживание электрических сетей по фактическому техническому состоянию [1]. Важнейшее условие функционирования такой системы – возможность проведения достаточно частых диагностических работ. На основании информации о фактическом техническом состоянии электрооборудования решаются такие вопросы, как:

– выявление деталей, которые требуют немедленной замены;

– определение характера и места нахождения дефекта на ранней стадии его развития;

– возможность достаточно точного прогнозирования дальнейшего состояния электрооборудования.

Главная цель технического диагностирования – распознавание состояния оборудования в условиях ограниченной информации, а также оценка остаточного

ресурса оборудования и повышение уровня его надежности. Это позволяет избежать надвигающихся аварийных ситуаций, научно обосновать и минимизировать выбор необходимого объёма ремонтных мероприятий.

Во избежание несчастных случаев, связанных с электротравматизмом обслуживающего персонала, диагностика технического состояния электрических сетей происходит при выводе их из работы, что является неудобным для потребителей, поскольку они теряют электроснабжение на некоторое время. В современных условиях техническая диагностика электрических сетей должна проводиться преимущественно под рабочим напряжением без вывода их из работы. Для этих целей в настоящее время применяют автоматизированные системы мониторинга и технического диагностирования (АСМТД) электрических сетей, которые производят диагностику электрооборудования без вмешательства человека. Такие системы функционируют в составе «Умных сетей» (Smart Grid) [2]. Основной недостаток АСМТД – большие финансовые затраты на их приобретение. Поэтому чаще всего они применяются для достаточно мощного и ответственного электрооборудования. Например, для силовых трансформаторов, автотрансформаторов, реакторов и т.д.

Проблема оценки технического состояния и надежности опорно-стрежневой и подвесной изоляции актуальна и востребована практикой, что подтверждается ежедневным опытом и статистикой эксплуатации различных типов изоляторов. Анализ повреждаемости по видам и типам изоляторов показал, что основная часть повреждений приходится на подвесную фарфоровую (более 50%) и подвесную стеклянную (20-25%) изоляцию [3]. Применение дорогостоящих облётов и осмотров линий электропередач (ЛЭП) на вертолётах увеличивает затраты на техническое обслуживание высоковольтных ЛЭП и далеко не всегда является эффективным.

Перспективным направлением диагностики ВЛ является мониторинг технического состояния с использованием беспилотных летательных аппаратов (БПЛА). На сегодняшний день тестовое применение БПЛА ограничивается аэро-

фотосъёмкой, что позволяет только констатировать факт уже повреждённых элементов ВЛ и снизить время обследования ВЛ (по сравнению с пешим осмотром), но не выявляет дефекты, приводящие к повреждению элементов ВЛ и возникновению аварийных ситуаций.

Целью работы является разработка системы определения механических и электрических повреждений подвесных изоляторов для контроля технического состояния воздушных линий электропередач и электрооборудования подстанций с возможностью применения БПЛА.

Мониторинг состояния гирлянд изоляторов предусматривает их распознавание на основе фотографий. Система принимает на вход фотографию и производит анализ. Таким образом определяется наличие и характер неисправностей. В случае обнаружения повреждения, система уведомит пользователя об этом. Применение разработанного метода предполагается в комплексе с БПЛА, что позволит проводить диагностику в труднодоступной местности, уменьшить время и затраты ресурсов на неё, а также не отключать ЛЭП.

Для реализации поставленной цели потребуется выполнить следующие задачи:

– проанализировать предметную область и рассмотреть аналогичные системы;

– спроектировать систему распознавания состояния изоляции электрооборудования;

– выполнить сбор и подготовку данных для обучения;

– реализовать алгоритм анализа изображений и обучить нейронную сеть;

– реализовать и протестировать приложение для определения состояния изоляции электрооборудования по фотографии.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Обзор предметной области

Линейный изолятор – устройство, предназначенное для крепления проводов на опорах воздушной линии электропередачи (ВЛ) с целью изоляции проводника и безопасной подачи электроэнергии.

В настоящее время наиболее распространённый тип подвесных изоляторов – стеклянные или фарфоровые тарельчатые изоляторы с шапкой, стержнем и цементной связкой, с помощью которой соединяются детали тарельчатых изоляторов. Структура подвесного изолятора представлена на рисунке 1.1.

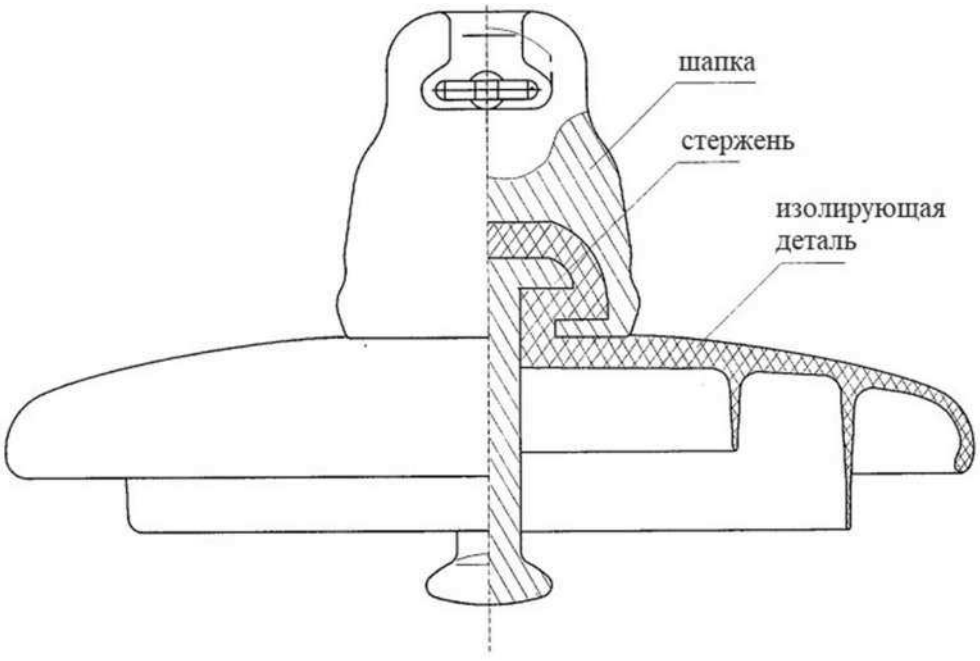


Рисунок 1.1 – Структура подвесного изолятора

Факторы возникновения неисправностей:

– атмосферные перенапряжения из-за интенсивности грозовой деятельности в районе прохождения трассы ВЛ;

– резкие перепады температуры наружного воздуха;

– среда, в которой эксплуатируются изоляторы;

– поражение молнией.

Возможные дефекты подвесных изоляторов:

– наличие мелких трещин и сколов;

– внутренние дефекты;

– «остаток» изолятора, состоящий из оставшейся части растрескавшейся стеклодетали, скрепленной армирующим составом с шапкой и стержнем изолятора;

– наличие поверхностных электрических перекрытий как единичного изолятора, так и всей сцепки гирлянды;

– электрическое разрушение стеклодетали с образованием мелкозернистой поверхностной структуры;

– загрязнение изоляторов и наличие несмываемого налета.

При появлении в гирлянде неисправных дефектных изоляторов возрастает риск пробоя изоляторов и последующее их разрушение вследствие неравномерного распределения напряженности на изоляторы. Протекание тока дуги через пробитые изоляторы приводит к разрушению тарелок из-за выделяющейся в канале внутреннего пробоя энергии и создаёт условия для образования трещин в шапке изолятора, выпадения силового узла и полного разрушения изолятора.

Для предотвращения подобных инцидентов требуется своевременная диагностика неисправностей и замена повреждённых изоляторов. Согласно Инструкции по эксплуатации ВЛ напряжением 35–800 кВ (РД34.20.504-94) [4], замена должна производиться в срочном порядке при наличии в гирлянде изолирующей подвески свыше 30% неисправных изоляторов; в течение ближайших трех месяцев – при наличии в гирлянде изолирующей подвески от 20 до 30% неисправных изоляторов и при очередном капитальном ремонте – при наличии в гирлянде изолирующей подвески до 20% неисправных изоляторов.

На сегодняшний день для определения дефектных изоляторов применяются такие способы контроля состояния ВЛЭП, как:

– наземный осмотр (пешие обходы линий), частота проведения – ежегодно по всей длине линии;

– верховой осмотр (пешие обходы линий, включая подъём электромонтёра на опору, с отключением ВЛ), частота проведения – 1 раз в 5 лет;

– облёты линий на вертолёте.

Наземный и верховой осмотры линий являются наиболее трудоёмкими, травмоопасными и распространенными в России. В труднодоступной местности такой метод диагностики представляет угрозу безопасности рабочего персонала, а для некоторых районов применение данного метода невозможно из-за заболоченности местности, отсутствия подъездных путей и т.д.

Облёты ВЛ проводятся реже в связи с существенными эксплуатационными и временными затратам. Во время облётов проводится осмотр элементов ВЛ с высоты бреющего полета и заполняются листы осмотра. Сделать это крайне сложно, так как минимальная скорость полета самолёта или вертолёта составляет 130-150 км/ч: медленнее летать опасно и экономически невыгодно эксплуатанту воздушного судна. Кроме того, облёты ВЛ проводятся без использования специального оборудования, поэтому эффективность данного метода ограничивается опытом и остротой зрения специалиста.

1.2 Обзор существующих аналогов

Во время изучения предметной области, в качестве аналогов создаваемой системы не были найдены продукты, работающие с выбранной предметной областью и конкретно с задачей распознавания. Приведённая выше система диагностики АСМТД базируется на сравнении заложенных параметров для нормального функционирования оборудования с фактическими параметрами, при этом датчики сравнения устанавливаются непосредственно на оборудование. Так как АСМТД не связана с задачей распознавания, далее будут рассмотрены различные системы, использующие технологию распознавания образов с помощью свёрточных нейронных сетей, но не связанные с выбранной предметной областью.

Распознавание медицинских изображений

Распознавание образов применяется к медицинским визуализациям (например, МРТ, рентген) [5]. С помощью семантической сегментации, представленной на рисунке 1.2, определяется точное местоположение и форма структур тела, а также наличие и локализация возможных аномалий и отклонений. Задачу семантической сегментации способны решить глубокие свёрточные нейронные сети. Но для достижения хороших результатов при их обучении необходима большая выборка размеченных данных, что требует значительных временных затрат.

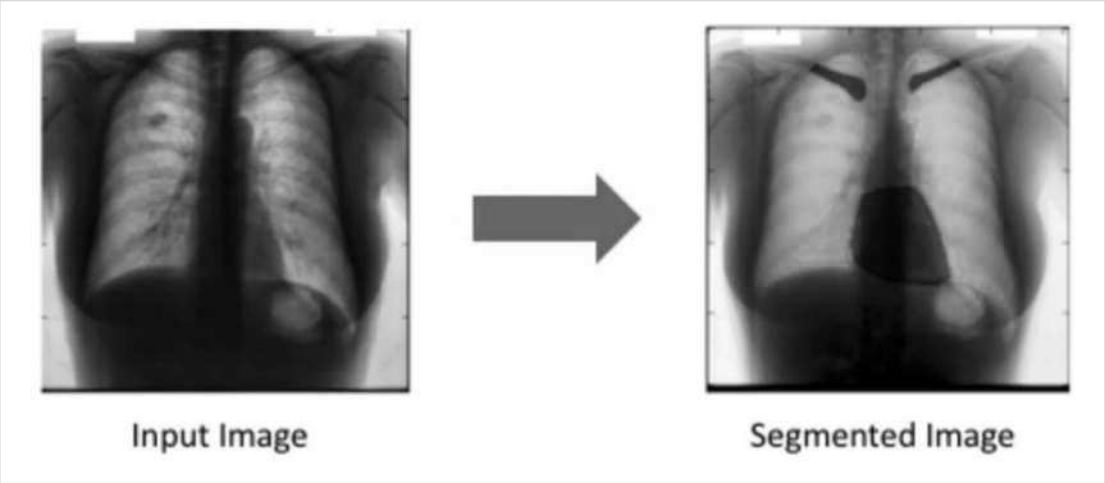


Рисунок 1.2 – Иллюстрация распознавания медицинских изображений

Автономные транспортные средства

Автономное вождение – сложная робототехническая задача, которая требует восприятия, планирования и выполнения в быстро меняющихся условиях [6]. Распознавание образов с помощью семантической сегментации предоставляет информацию о предоставляет информацию о свободном пространстве на дорогах, а также обнаруживает разметку полос, дорожные знаки, парковочные места, находящихся поблизости людей и т.д. Пример сегментации объектов на дороге представлен на рисунке 1.3. Результаты семантической сегментации затем используются для принятия решений по правильному управлению транспортного средства.

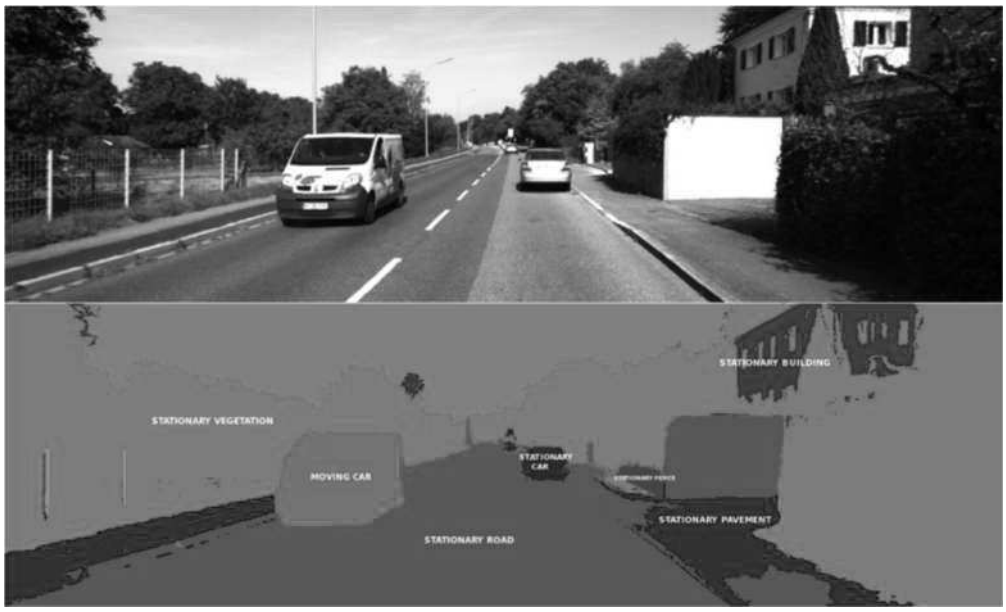


Рисунок 1.3 – Семантическая сегментация в беспилотном автомобиле

Обработка изображений со спутников

Распознавание образов также используется для определения типов земной поверхности по спутниковым снимкам. Пример сегментации спутниковых снимков представлен на рисунке 1.4. Один из вариантов использования технологии – определение контуров водоёмов для предоставления более точной картографической информации. Продвинутые алгоритмы используются для нанесения на карту дорог, определения типов сельскохозяйственных культур и т.д.

Рисунок 1.4 – Семантическая сегментация спутникового снимка

Умные парковки («Интерсвязь»)

Данная система позволяет определить наличие свободных мест на парковках городов Челябинской области [7]. Сервис показывает ближайшие незанятые парковочные места, используя GPS устройство пользователя. Для распознавания свободных мест применяется свёрточная нейронная сеть. Обновление изображения происходит каждые тридцать секунд. Результат работы приложения представлен на рисунке 1.5.

Рисунок 1.5 – Иллюстрация работы приложения

1.3 Обзор решений для реализации проекта

TensorFlow – это программная библиотека для машинного обучения с открытым исходным кодом, разработанная для решения задач построения и тренировки нейронной сети с целью автоматического нахождения и классификации образов [8]. Платформа предоставляет интуитивно понятные высокоуровневые API-интерфейсы (например, Keras) с быстрым выполнением, что обеспечивает немедленную итерацию модели и простую отладку.

Keras – это высокоуровневый программный интерфейс, нацеленный на оперативную работу с сетями глубокого обучения [9]. Работает на платформе ма-

шинного обучения TensorFlow. Обладает такими характеристиками, как минимальность, модульность и расширяемость.

OpenCV – это библиотека компьютерного зрения и машинного обучения с открытым исходным кодом, предназначенная для анализа, классификации и обработки изображений [10]. Содержит алгоритмы для интерпретации изображений, определения формы и сегментации объектов, калибровки камеры и т.д.

PyTorch – это библиотека глубокого обучения на языке Python с открытым исходным кодом, обеспечивающая тензорные вычисления с GPU-ускорением [11]. В PyTorch имеется большое количество предварительно обученных моделей.

Выводы по первой главе

На сегодняшний день отсутствуют приложения, которые могли бы упростить задачу диагностики состояния изоляции электрооборудования без отключения комплекса подстанций и ВЛ.

Стоит отметить существование большого количества различных решений для реализации проекта, а также широкого ряда библиотек для создания и обучения нейронных сетей. Использование подобных библиотек может значительно облегчить задачу распознавания состояния изоляции электрооборудования.

Принимая это во внимание, а также описанные выше факторы, можно сделать вывод, что исследования в данной области являются необходимыми, а задача разработать данное приложение – актуальной.

2 ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ

2.1 Функциональные и нефункциональные требования к системе

Для решения задачи распознавания состояния изоляции электрооборудования необходимо создать систему, с помощью которой можно обработать входное изображение, запустить поиск неисправностей и вывести результат на экран, а также в виде готового отчёта.

Функциональные требования

Функциональные требования к разрабатываемому приложению описывают, как работает система. После разработки она должна удовлетворять следующим требованиям:

- принимать на вход изображение в формате .jpg и .png;
- распознавать детали подвесного изолятора;
- распознавать следующие неисправности: микротрещины, сколы, повреждения электрическим током и оплавления;
- после обработки изображения выводить результат распознавания на экран приложения и, если есть запрос от пользователя, выводить результат в документ формата .doc.

Нефункциональные требования

Нефункциональные требования включают в себя те свойства, которые имеются у приложения. Они также определяют критерии, которые могут использоваться для оценки работы системы. Список нефункциональных требований к системе:

- система должна быть реализована на языке Python 3.7;
- система должна использовать фреймворк PyTorch и предварительно обученную модель в качестве основы для создания глубокой свёрточной нейронной сети;

– приложение должно обладать простым и дружественным пользователю интерфейсом, не требующим от него действий, выходящих за рамки имеющегося функционала;

– пользовательский интерфейс должен быть реализован на языке C# с использованием Windows Forms.

Использование языка Python обусловлено необходимостью задействования фреймворка машинного обучения для этого языка – PyTorch. В свою очередь, выбор именно этого фреймворка обусловлен простотой создания и обучения моделей, а также наличием большого количества справочной литературы, что немало важно. Фреймворк PyTorch имеет большое количество предварительно обученных моделей и готовых модульных частей, которые легко комбинируются между собой.

Для разработки интерфейса была выбрана платформа пользовательского интерфейса для создания классических приложений Windows – Windows Forms. Она поддерживает широкий набор функций для разработки приложений (элементы управления, графику, привязку данных и т.д.), имеет удобный визуальный конструктор для создания приложений, а также предоставляет возможность разработки кроссплатформенного графического пользовательского интерфейса.

2.2 Варианты взаимодействия с системой

Для проектирования системы использовался UML (united modeling language) – универсальный язык моделирования для разработки в области программного обеспечения, предназначенный для стандартного способа визуализации проектирования системы [12].

В построенной диаграмме прецедентов отражены варианты использования системы распознавания состояния изоляции электрооборудования. Диаграмма прецедентов представлена на рисунке 2.1.

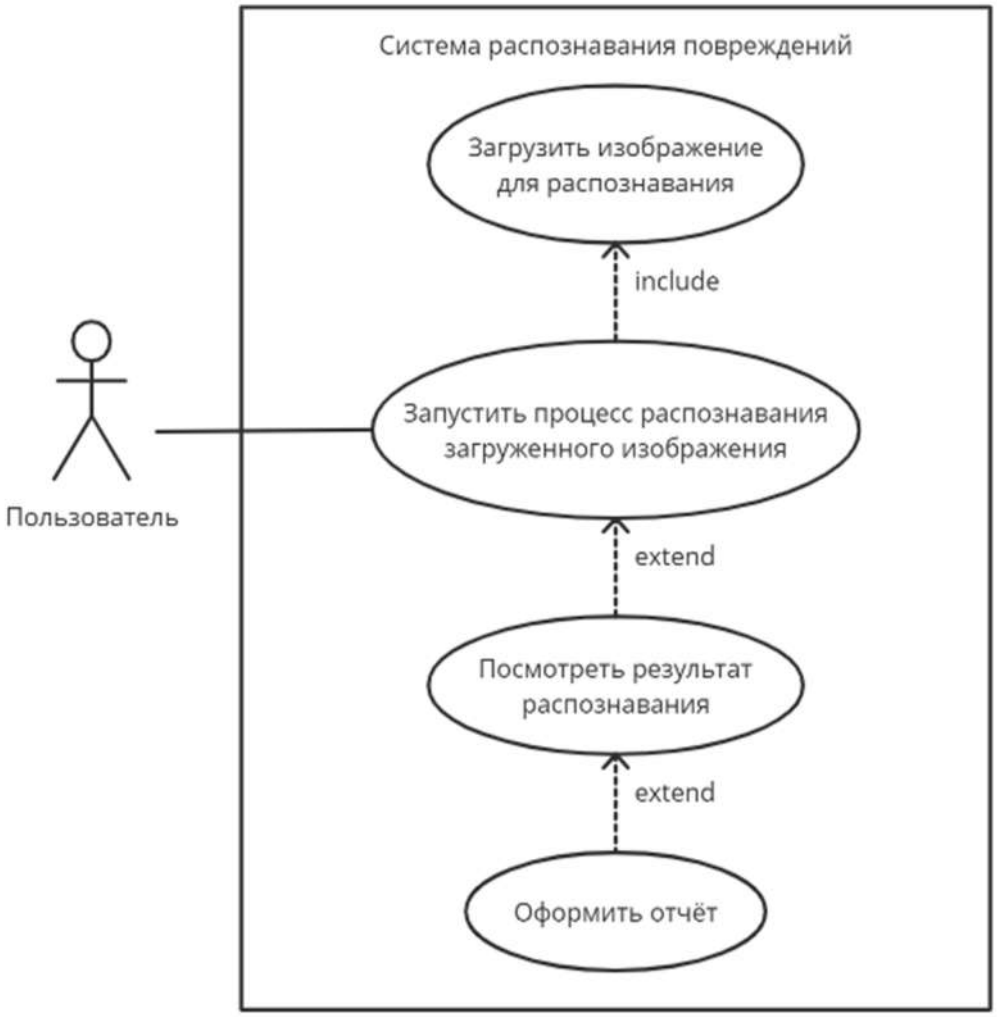


Рисунок 2.1 – Диаграмма прецедентов

Актёры, участвующие во взаимодействии с приложением: пользователь – человек, задействующий функционал системы.

Описание вариантов взаимодействия:

– загрузить изображение для распознавания (выбрать расположение изображения для распознавания и загрузить его);

– запустить процесс распознавания загруженного изображения;

– посмотреть результат распознавания;

– оформить отчёт.

2.3 Диаграмма деятельности приложения

Исходя из функциональных и нефункциональных требований выведена диаграмма деятельности системы, представленная на рисунке 2.2.

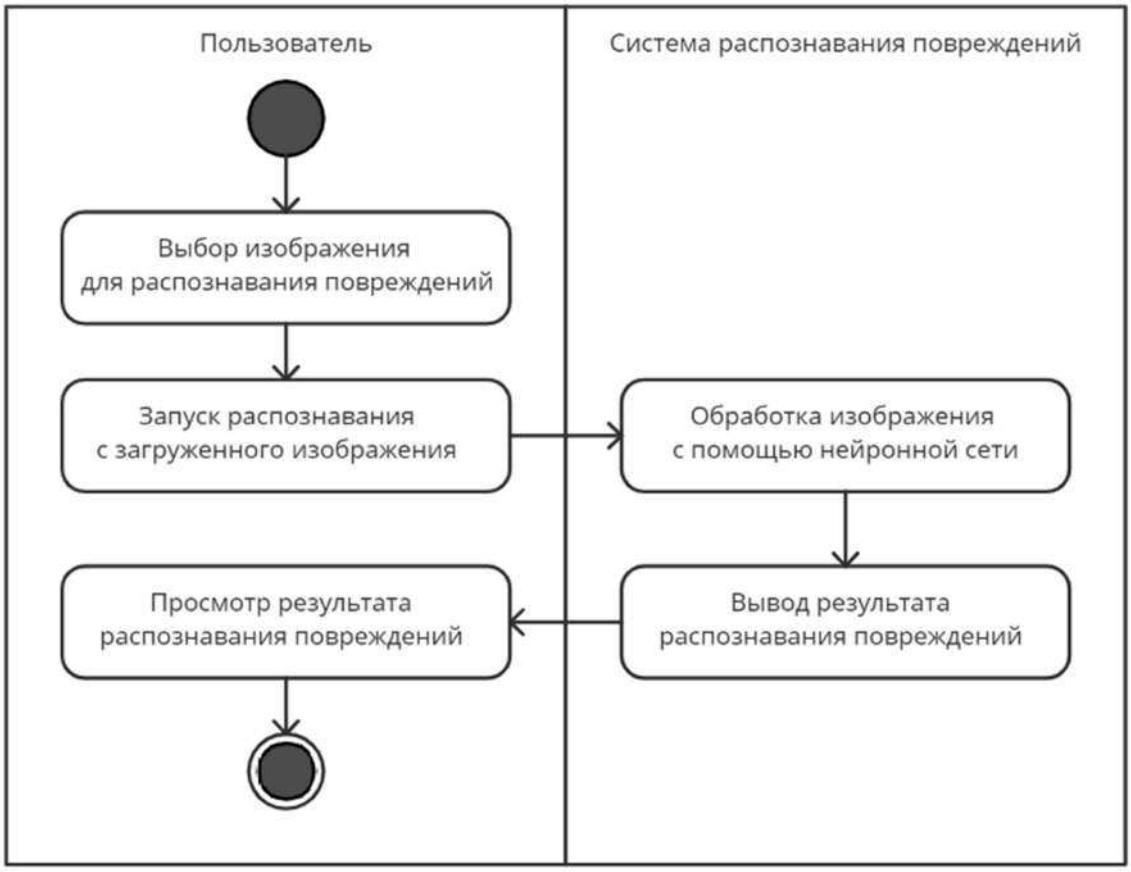


Рисунок 2.2 – Диаграмма деятельности приложения

Диаграмма деятельности отражает процесс взаимодействия пользователя с системой распознавания повреждений.

2.4 Диаграмма последовательности

Диаграмма последовательности системы распознавания повреждений представлена на рисунке 2.3. Рассматриваемый вид диаграмм демонстрирует более детальное описание логики сценариев использования.

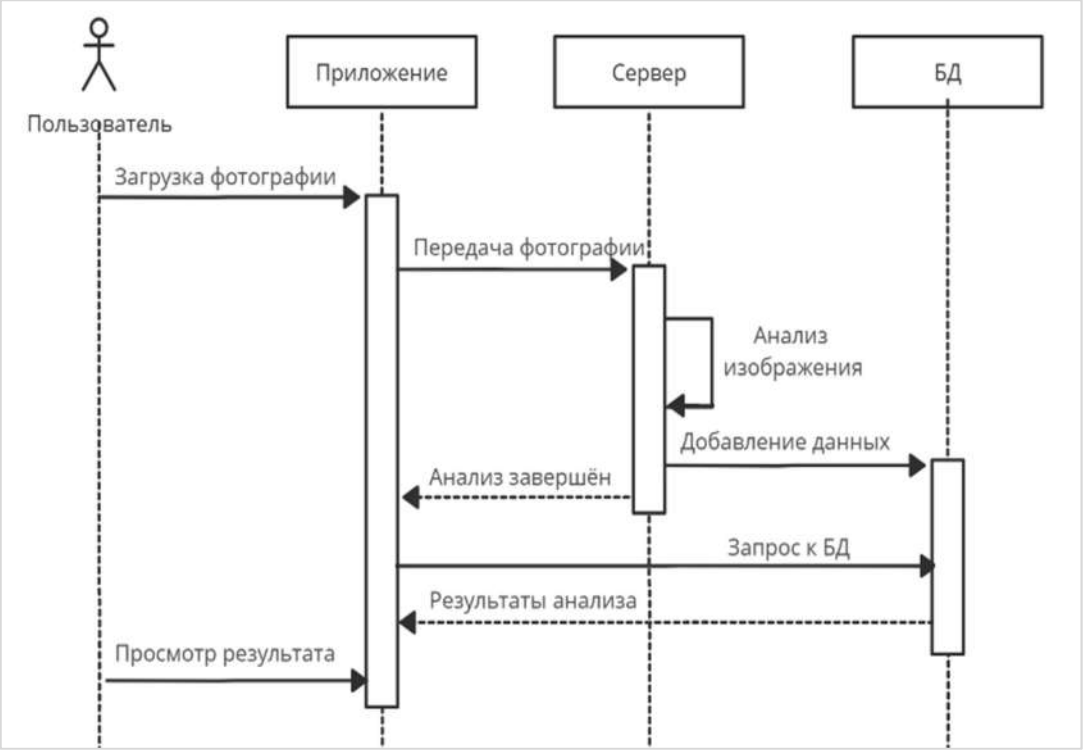


Рисунок 2.3 – Диаграмма последовательности

2.5 Проектирование интерфейса приложения

Прототип интерфейса представлен на рисунке 2.4.

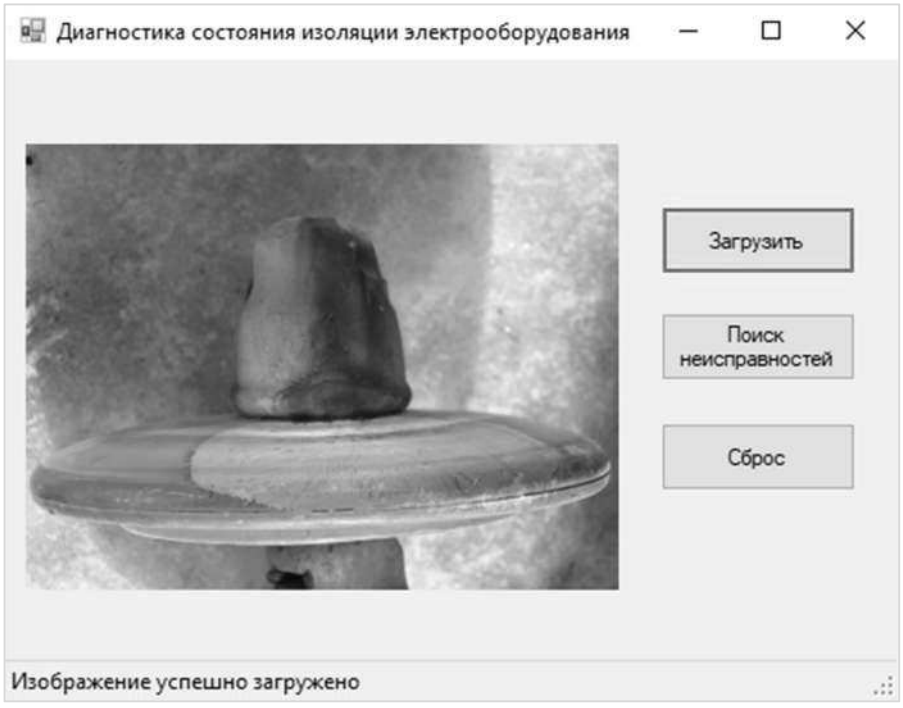
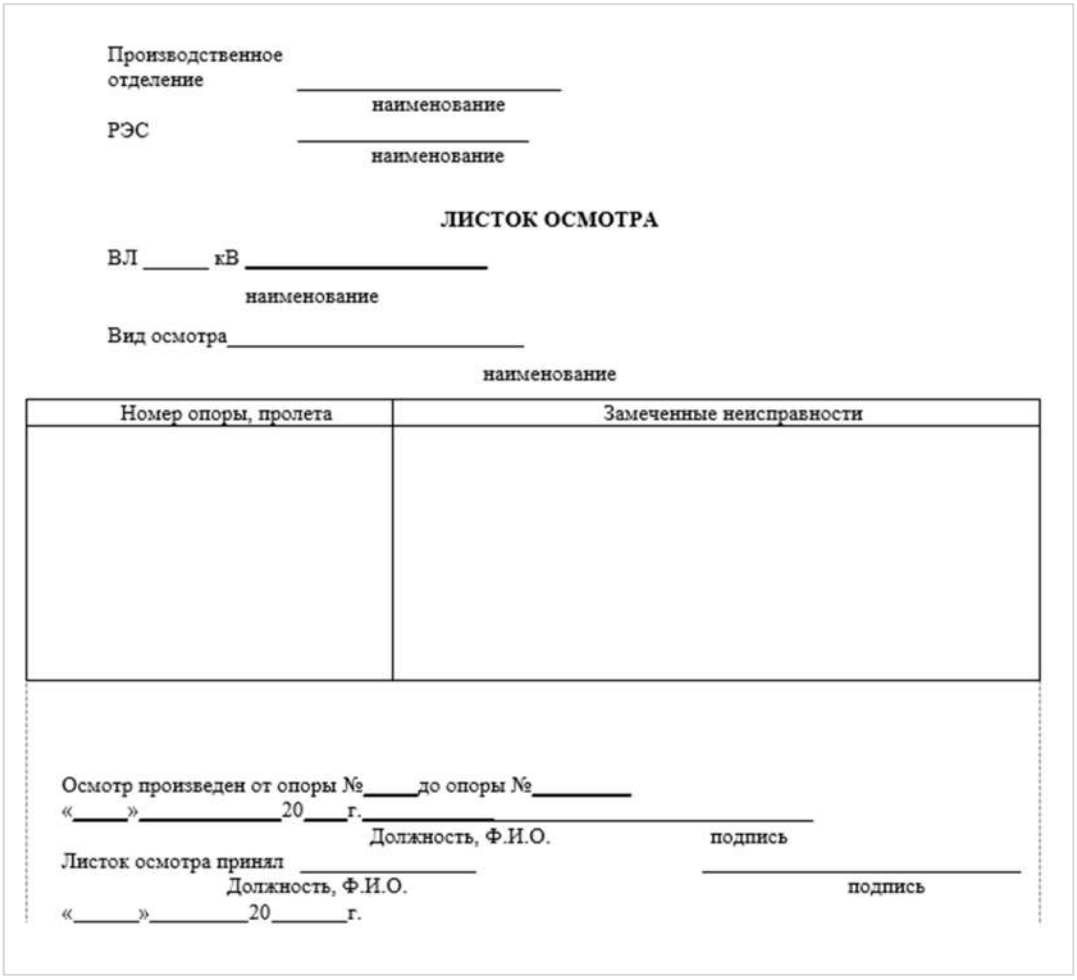


Рисунок 2.4 – Внешний вид интерфейса

После запуска приложения пользователю предоставляется возможность выбрать изображение и загрузить его для диагностики неисправностей. По нажатию кнопки «Поиск неисправностей» происходит запуск процесса оценивания состояния подвесного изолятора и поиск неисправностей.

После окончания процесса распознавания пользователь может ознакомиться с результатами распознавания, а также по нажатию кнопки «Создать отчет» сформировать отчёт в документ формата .doc. Пример отчёта представлен на рисунке 2.5.



Производственное отделение _____
наименование
РЭС _____
наименование

ЛИСТОК ОСМОТРА

ВЛ _____ кВ _____
наименование

Вид осмотра_____
наименование

Номер опоры, пролета	Замеченные неисправности

Осмотр произведен от опоры №____до опоры №_____
«____»_____20___г._____
Должность, Ф.И.О. подпись

Листок осмотра принял _____ _____
Должность, Ф.И.О. подпись
«_____»_____20_____г.

Рисунок 2.5 – Пример сформированного отчёта

3 РЕАЛИЗАЦИЯ РАСПОЗНАВАНИЯ ИЗОБРАЖЕНИЙ

Процесс анализа изображения в разрабатываемом приложении заключается в поиске и сегментации различных повреждений подвесных изоляторов и их локализации. Для решения данной задачи применены свёрточные нейронные сети и распознавание образов.

3.1 Свёрточные нейронные сети

Свёрточная нейронная сеть (convolutional neural network, CNN) – специальная архитектура искусственных нейронных сетей, которая нацелена на эффективное распознавание образов и входит в состав технологий глубокого обучения [13]. Свёрточные нейронные сети обеспечивают частичную устойчивость к изменениям масштаба, смещениям, поворотам, смене ракурса и другим искажениям.

Свёрточная нейронная сеть состоит из слоёв нескольких видов:

- свёрточные слои (convolutional layer);
- субдискретизирующие слои (subsampling, pooling);
- слои полносвязной нейронной сети – персептрона.

На рисунке 3.1 представлена топология свёрточной нейронной сети.

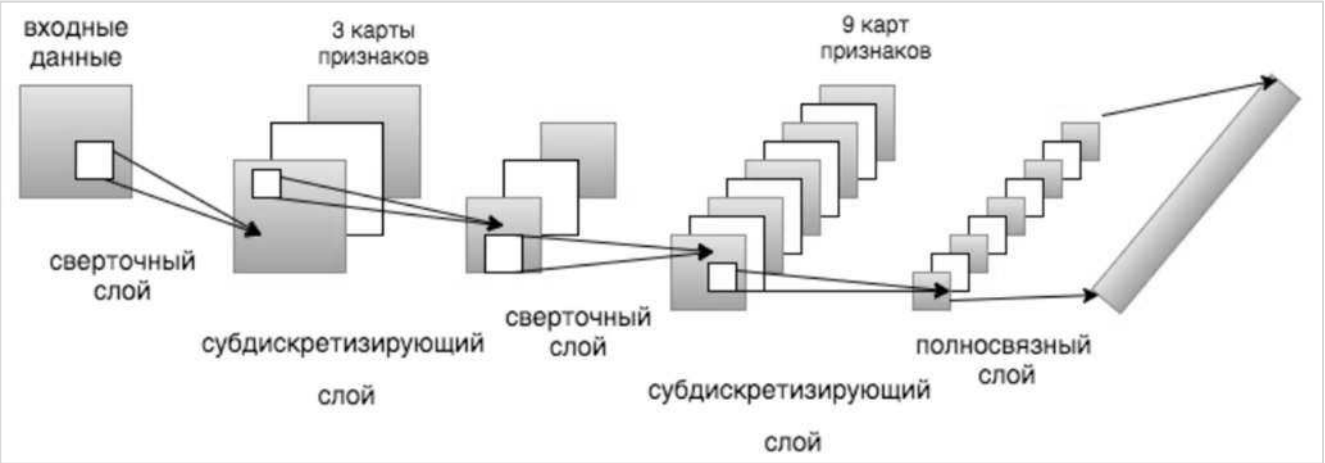


Рисунок 3.1 – Топология свёрточной нейронной сети

В полносвязной нейронной сети каждый нейрон связан со всеми нейронами предыдущего слоя, и каждая связь имеет свой персональный весовой коэффициент. В свёрточной нейронной сети в операции свёртки используется лишь ограниченная матрица весов небольшого размера, которая передвигается по обрабатываемому слою и формирует после каждого сдвига сигнал активации для нейрона следующего слоя с аналогичной позицией. Таким образом, для различных нейронов выходного слоя используется одна и та же матрица весов – ядро свёртки. Она интерпретируется как графическое кодирование какого-либо признака. В результате операции ядра свёртки получается следующий слой, который показывает наличие данного признака в обрабатываемом слое и его координаты, формируя «карту признаков».

Свёрточный слой

Основной блок свёрточной нейронной сети – слой свёртки, представленный на рисунке 3.2.

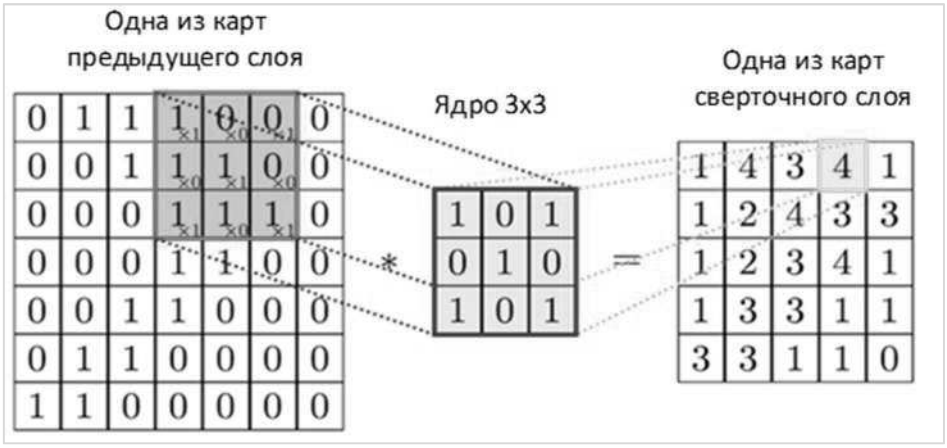


Рисунок 3.2 – Процесс свёртки в свёрточной нейронной сети

Свёрточный слой включает в себя матрицу весов (ядро свёртки), которая обрабатывает предыдущий слой по фрагментам. Весовые коэффициенты ядра свёртки устанавливаются в процессе обучения.

Подвыборочный слой

Подвыборочный слой представлен на рисунке 3.3.

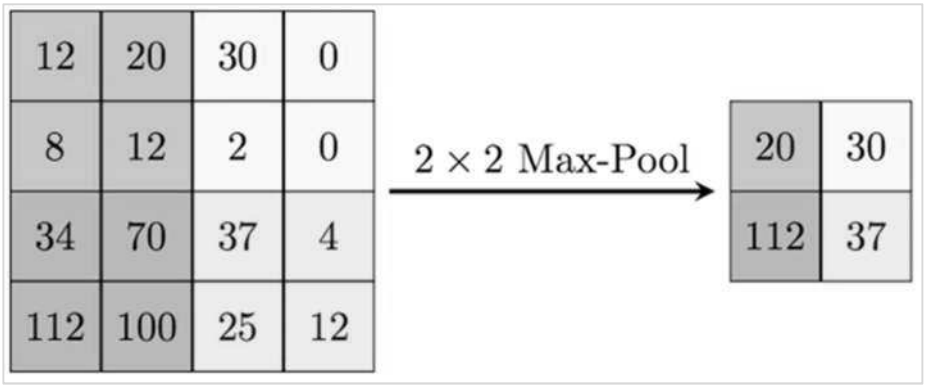


Рисунок 3.3 – Формирование новой карты подвыборочного слоя

На этапе субдискретизации (операция подвыборки, пулинг) выполняется уменьшение размерности сформированных карт признаков: из нескольких соседних нейронов карты признаков выбирается максимальный и принимается за один нейрон уплотнённой карты признаков меньшей размерности. Использование этого слоя позволяет ускорить дальнейшие вычисления, а также улучшить распознавание образов с изменённым масштабом.

Полносвязный слой

Последний из типов слоёв – слой обычного многослойного персептрона, представленный на рисунке 3.4.

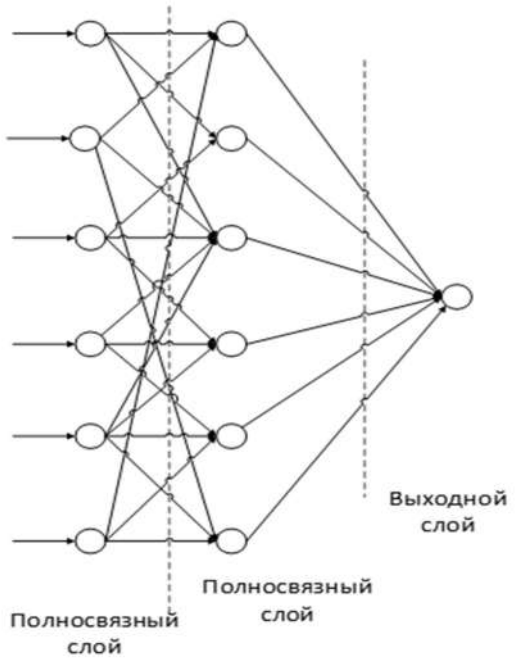


Рисунок 3.4 – Схема полносвязного слоя

Слои, расположенные до полносвязного слоя, являются средствами предобработки изображения и используются для выделения различных признаков, а дальнейшая классификация выполняется персептроном (сетью прямого распространения).

Задачей полносвязного слоя – смоделировать сложную нелинейную функцию, которая используется для классификации. Эта функция оптимизируется в процессе обучения, благодаря чему улучшается качество распознавания.

3.2 Распознавание образов

Любая классификация всегда основывается на прецедентах. Прецедент – это ранее классифицированный объект, принимаемый как образец при решении задач классификации [14]. Все объекты или явления разбиты на конечное число классов. Для каждого класса известно конечное число прецедентов. Задача распознавания образов заключается в отнесении нового объекта к определенному классу с помощью выделения существенных признаков, характеризующих этот объект.

Существует четыре задачи распознавания образов, которые иллюстрирует рисунок 3.5.

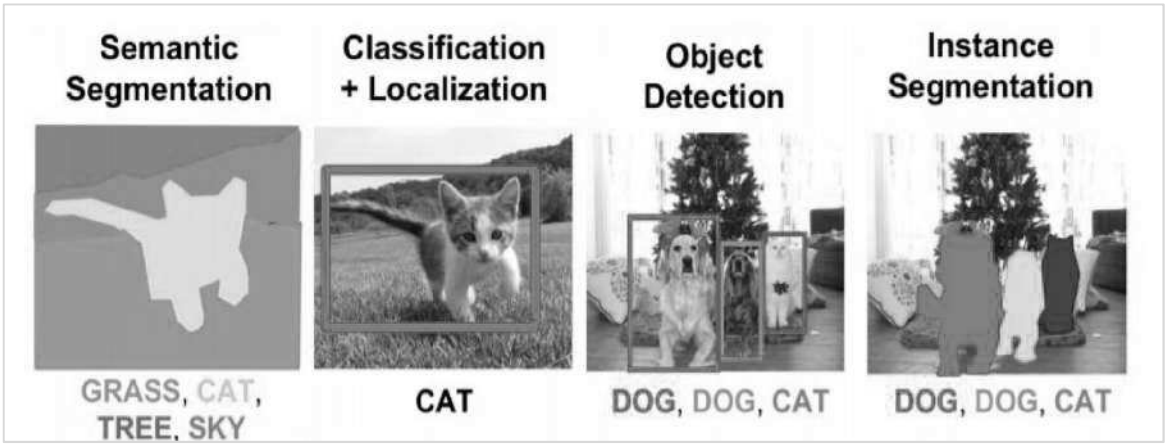


Рисунок 3.5 – Задачи распознавания образов

1. Семантическая сегментация: классификация каждого пикселя по определенному классу.

2. Классификация + локализация: классификация и локализация одного объекта изображения.

3. Обнаружение объекта: классификация и локализация всех объектов изображения.

4. Сегментация экземпляра: вместе с классификацией на уровне пикселей происходит классификация каждого экземпляра класса отдельно.

Для решения задачи распознавания частей изолятора использовалась семантическая сегментация, так как необходимо не только определять класс объектов, но и выявлять их структуры, правильно выделять части объектов на изображении.

3.3 Подготовка данных

Важный этап на пути обучения моделей – подготовка данных, в результате которой создаётся обработанный набор очищенных данных и их масок, пригодных для обработки алгоритмами нейронных сетей. Такая выборка (dataset) нужна для тренировки модели, чтобы обучить систему и затем использовать её для решения реальных задач.

Для решения поставленной задачи сегментации частей подвесных изоляторов и поиска неисправностей на них были проведены сбор и подготовка данных, так как необходимых готовых датасетов в открытом доступе нет. Подготовка данных состояла из трёх этапов.

1. Поиск или создание изображений.

2. Разметка изображений для обучения.

3. Разделение данных на выборки: обучающую, тестовую и валидационную.

Для реализации первого этапа с помощью фотоаппарата были сделаны снимки частей подвесного изолятора, различных видов повреждений, а также гирлянд изоляторов.

Разметка данных для обучения моделей является ресурсоемких процессом, что характерно для задач компьютерного зрения. Именно поэтому для разметки датасета был выбран сервис Supervisely [15], который значительно облегчает ручную разметку данных.

Для задачи сегментации частей подвесного изолятора было размечено 106 изображений, которые в общем содержали восемь классов:

1) тарелка (plate);
2) шапка (header);
3) основание (kernel);
4) скол (chip);
5) трещина (crack);
6) оплавление (reflow);
7) перекрытие электрическим током (overlap);
8) разбитое стекло (broken glass).

Список классов, где отражён тип каждого объекта и его цвет, приведён на рисунке 3.6.

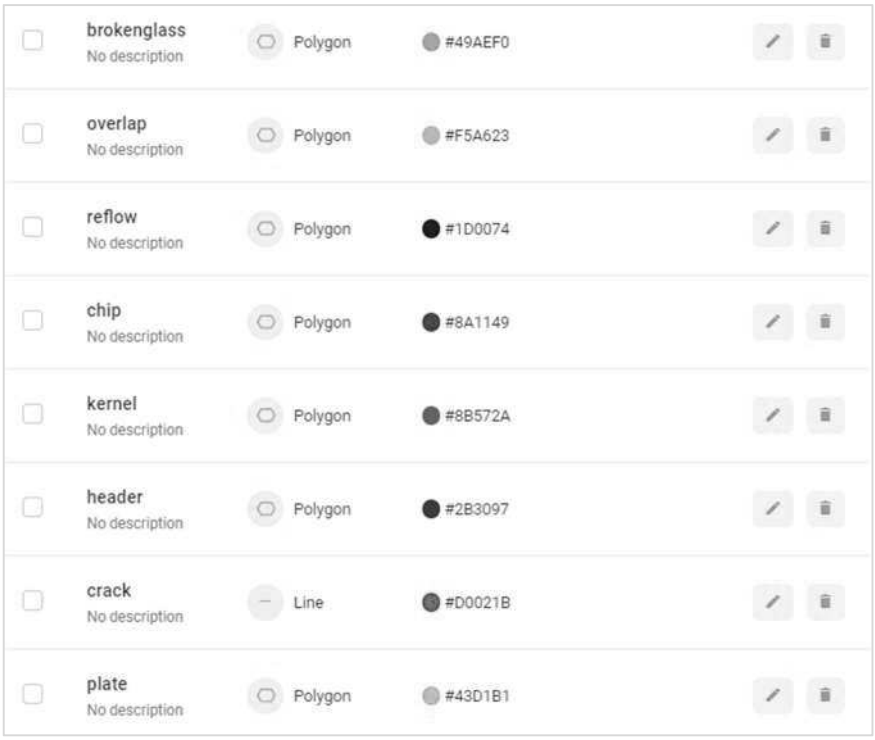


Рисунок 3.6 – Список классов

Для разметки использовались два типа объектов:

1) линии (line);
2) полигоны (polygone).

Пример размеченных изображений представлен на рисунке 3.7.

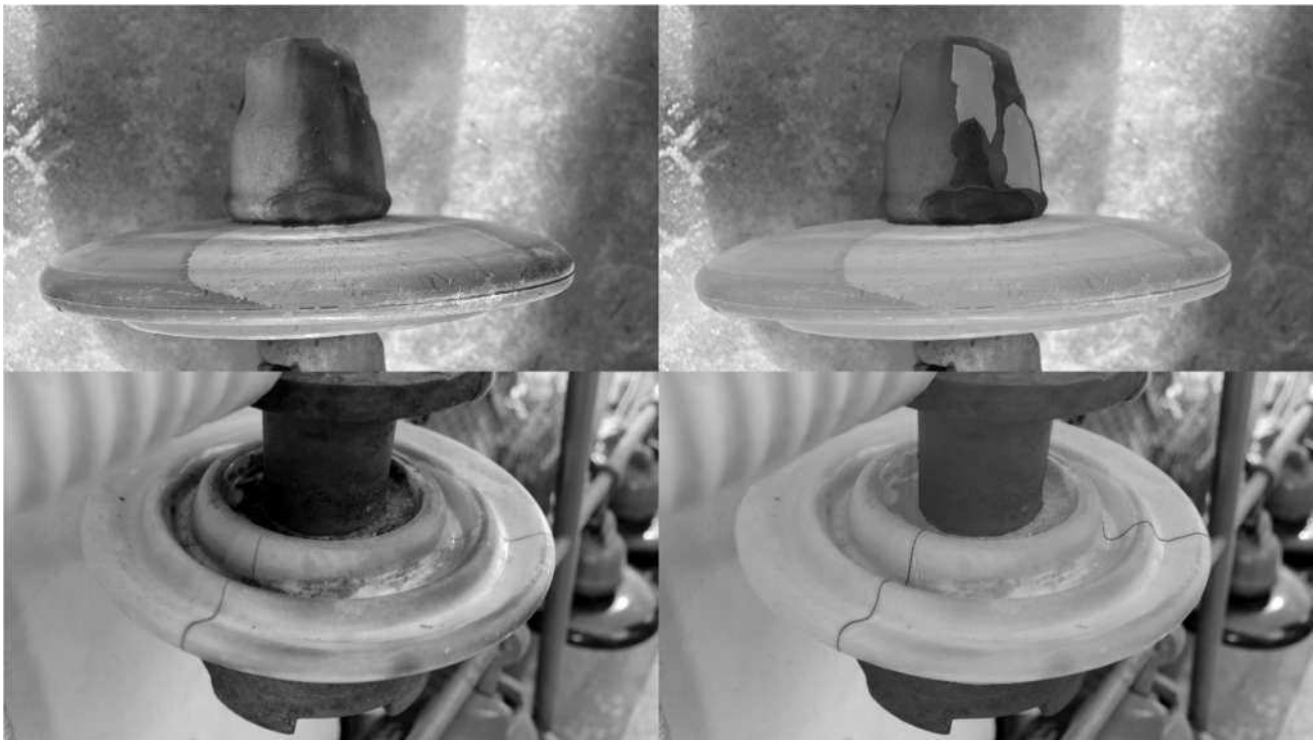


Рисунок 3.7 – Размеченные изображения

Статистика сегментации представлена на рисунке 3.8.

	Первая выборка (р-р 1366x768)	Total
plate	54 (96.43%)	54 (96.43%)
crack	14 (25.00%)	14 (25.00%)
header	50 (89.29%)	50 (89.29%)
kernel	33 (58.93%)	33 (58.93%)
chip	17 (30.36%)	17 (30.36%)
reflow	7 (12.50%)	7 (12.50%)
overlap	12 (21.43%)	12 (21.43%)
brokenglass	2 (3.57%)	2 (3.57%)
Partially / completely marked	56 (100.00%)	56 (100.00%)
Image not marked	0 (0.00%)	0 (0.00%)
Images in dataset		56

Рисунок 3.8 – Статистика сегментации

После разметки изображений на выходе получены файлы в текстовом формате JSON, содержащие координаты размеченных классов. Из этих данных необходимо сделать маски, которые впоследствии будут использоваться для обучения модели. Файл в формате JSON с координатами размеченных классов представлен на рисунке 3.9.

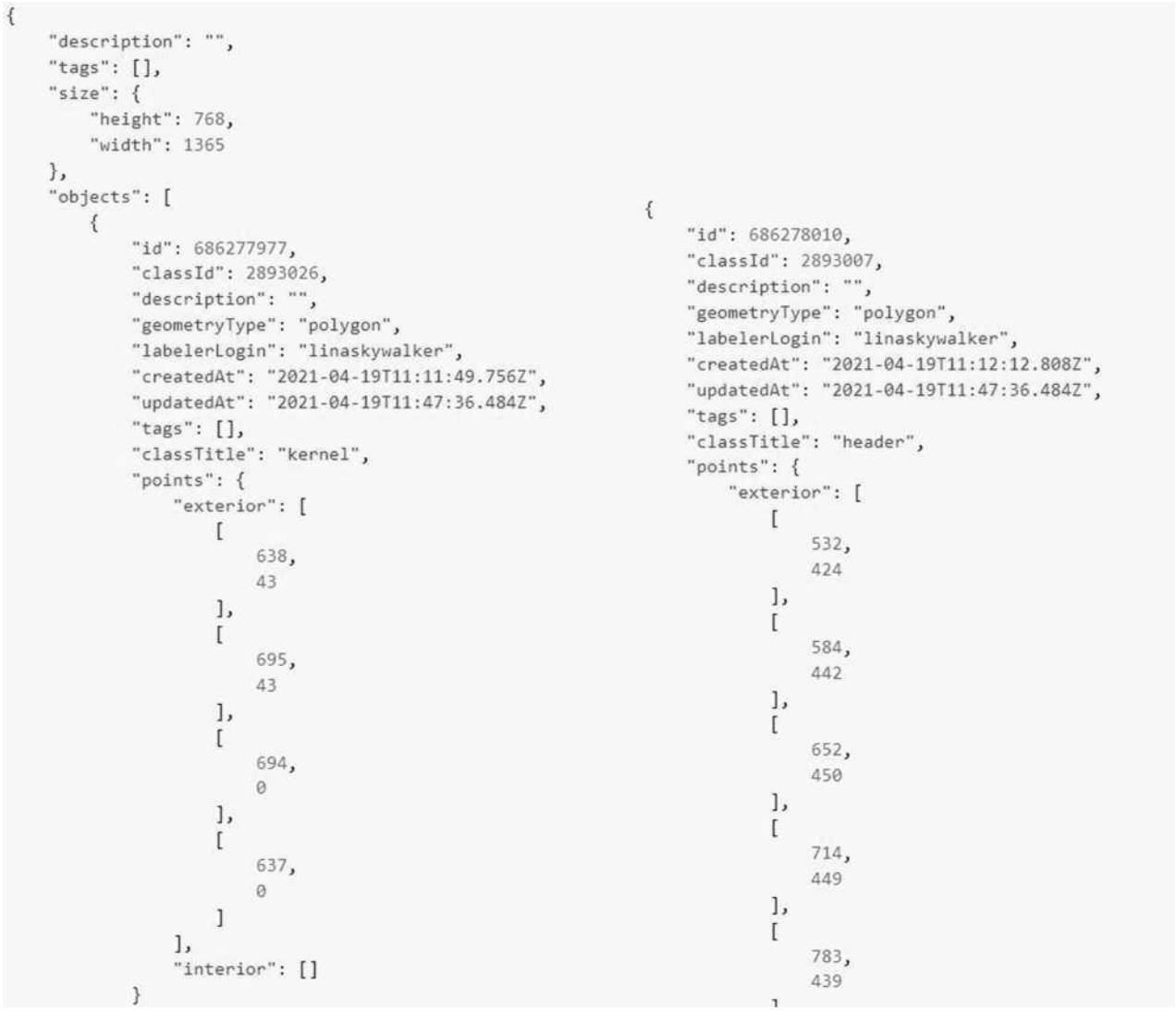


Рисунок 3.9 – Файл в JSON формате с координатами размеченных классов

Для трансформации JSON файлов в маски, представленные на рисунке 3.10, реализован алгоритм, приведённый в Приложении А.

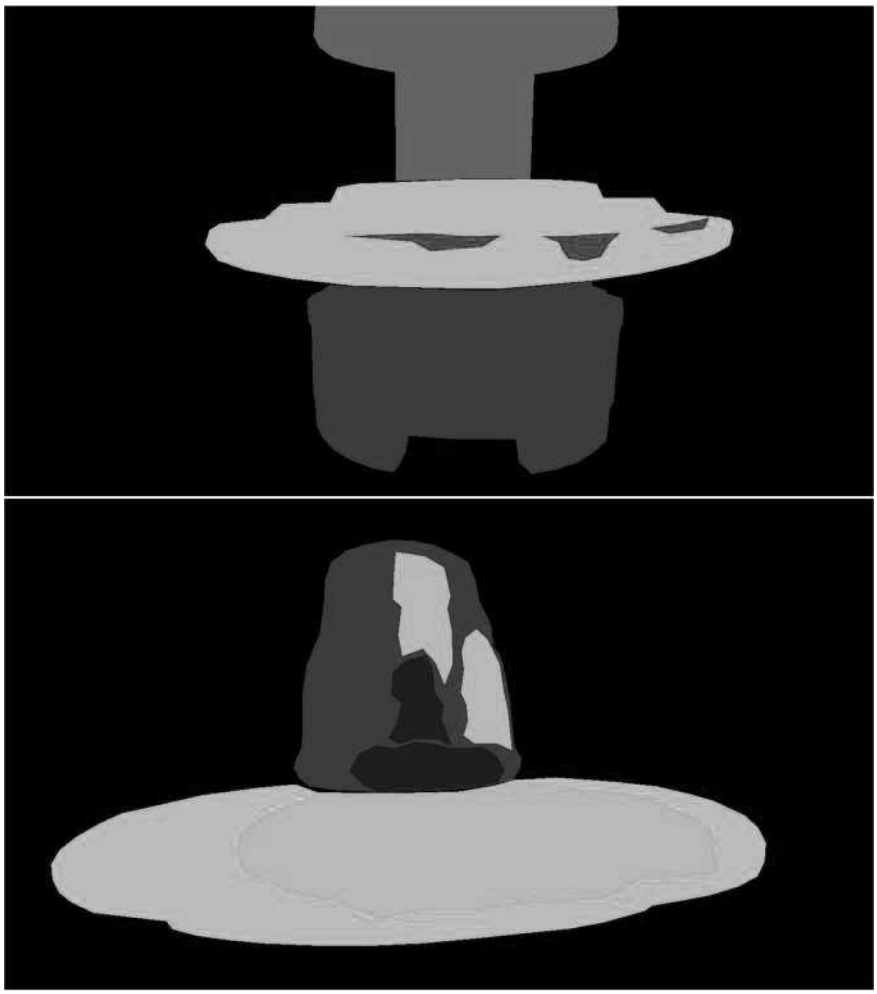


Рисунок 3.10 – Маски для обучения модели

Третий этап подготовки данных рассмотрен в пункте 3.5.

3.4 U-Net

Для решения задачи семантической сегментации была выбрана архитектура U-Net. U-Net – это свёрточная нейронная сеть, которая изначально была разработана для сегментации биомедицинских изображений [16]. Несмотря на это, сеть универсальна и может использоваться для решения задач сегментации изображений разного рода.

Для U-Net характерно использование небольшого количества данных для достижения хороших результатов, что стало решающим фактором при выборе архитектуры, так как размер имеющейся выборки сравнительно небольшой.

Архитектура сети приведена на рисунке 3.11. Она представляет собой полносвязную свёрточную нейронную сеть, модифицированную таким образом, чтобы работать с меньшим количеством обучающих образов и при этом делать более точную сегментацию.

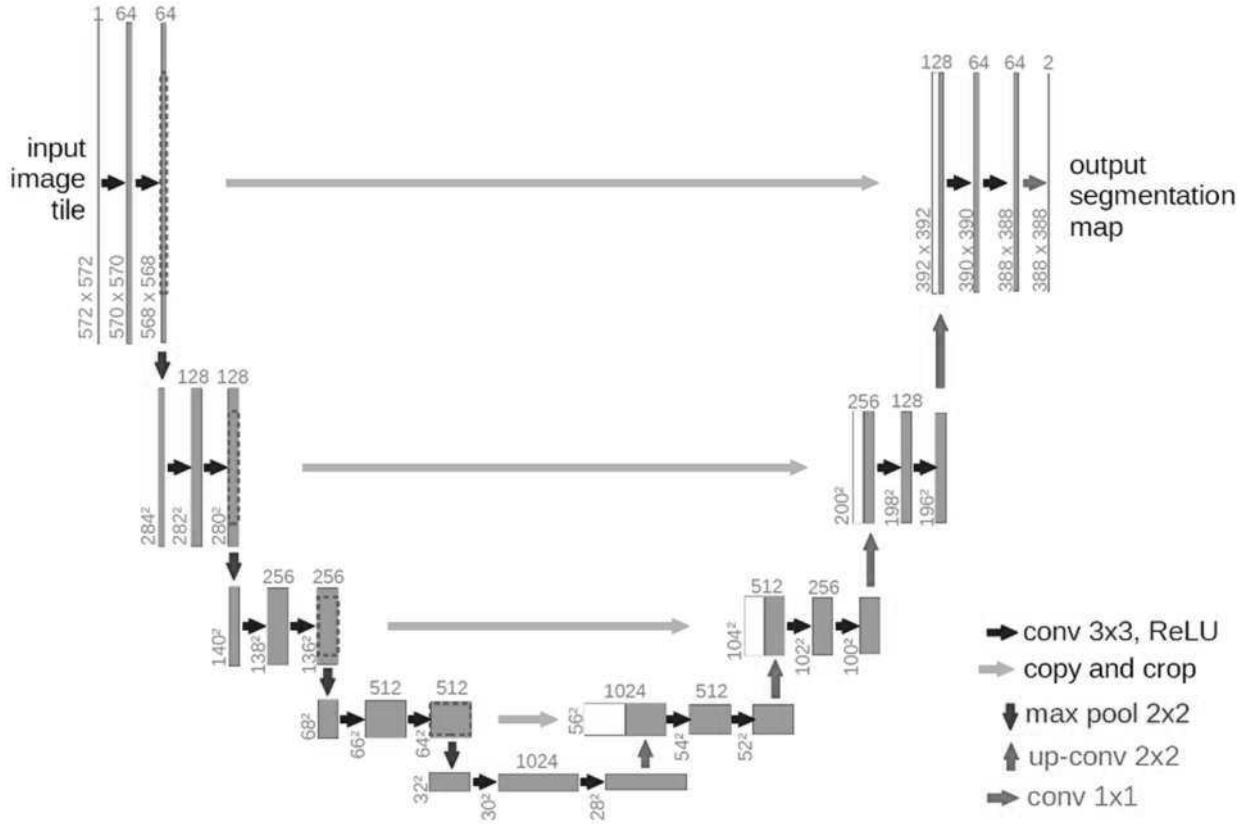


Рисунок 3.11 – Архитектура сети U-Net

Архитектура U-Net состоит из кодера и декодера. Кодер принимает изображение, выполняет различные свертки и операции максимального объединения изображения и создает его скрытое представление. Декодер получает это представление, увеличивает выборку изображения с помощью пропускаемых соединений и генерирует маску сегментации.

3.5 Сегментация изображений

На вход поступил подготовленный датасет, содержащий 106 изображений, которые были разделены на три выборки (тестовую, тренировочную и валидационную) с помощью кода, представленного в листинге 3.1.

Листинг 3.1 – Фрагмент процедуры разделения выборок

```
list_folders = ['content']
for folder in list_folders:
  list_images = os.listdir(os.path.join('/content/IsolatorSegmentation/',
folder, 'img'))
  random.shuffle(list_images)
  length = len(list_images)
  train = list_images[:int(0.8 * length)]
  val = list_images[int(0.8 * length):int(0.9 * length)]
  test = list_images[int(0.9 * length):]
  for im in train:
    shutil.copyfile(os.path.join('/content/IsolatorSegmentation', folder,
'img', im), os.path.join('/content/dataset/train/images', folder+im))
    shutil.copyfile(os.path.join('/content/IsolatorSegmentation', folder,
'masks', im[:-3] +'png'), os.path.join('/content/dataset/train/masks', fold-
er+im[:-3] +'png'))
  for im in val:
    shutil.copyfile(os.path.join('/content/IsolatorSegmentation', folder,
'img', im), os.path.join('/content/dataset/val/images', folder+im))
    shutil.copyfile(os.path.join('/content/IsolatorSegmentation', folder,
'masks', im[:-3] +'png'), os.path.join('/content/dataset/val/masks', fold-
er+im[:-3] +'png'))
  for im in test:
    shutil.copyfile(os.path.join('/content/IsolatorSegmentation', folder,
'img', im), os.path.join('/content/dataset/test/images', folder+im))
    shutil.copyfile(os.path.join('/content/IsolatorSegmentation', folder,
'masks', im[:-3] +'png'), os.path.join('/content/dataset/test/masks', fold-
er+im[:-3] +'png'))
```

Использование предварительно обученной модели значительно ускоряет обучение, поэтому в данном проекте использовалась остаточная сеть ResNet34 [17], обученная на данных ImageNet [18], а также предварительно подготовленные веса для инициализации модели. Функция активации и количество выходных классов модели изменены в соответствии с задачей.

Инициализация модели с предварительно обученным кодером (encoder) представлена в листинге 3.2. Алгоритм анализа изображений приведён в Приложении Б.

Листинг 3.2 – Инициализация модели

```
ENCODER = 'resnet34'
ENCODER_WEIGHTS = 'imagenet'
model = smp.Unet(
  encoder_name=ENCODER,
  encoder_weights=ENCODER_WEIGHTS,
  classes=8,
  activation='sigmoid'
)
model.to(device)
```

В связи с тем, что имеющаяся выборка небольшая, перед её применением использована аугментация данных – техника дополнения/расширения данных за счёт генерации похожих данных на основе уже имеющихся. Применение библиотеки Albumentation для аугментации данных представлено листинге 3.3.

Листинг 3.3 – Аугментация данных

```
def get_training_augmentation():
  train_transform = [
    albu.Resize(256, 256),  #изменение размера
    albu.HorizontalFlip(p=0.5),  #разворот по горизонтали вокруг оси Y с  вероятностью 50%
    albu.ShiftScaleRotate(scale_limit=0.1, rotate_limit=0, shift_limit=0.03, p=0, border_mode=0),  #аффинные преобразования
      #scale_limit - диапазон коэффициентов масштабирования
      #rotate_limit - диапазон вращения
      #shift_limit - диапазон коэффициентов сдвига для высоты и ширины
      #p - вероятность применения преобразований
      #border_mode - метод экстраполяции пикселей
    albu.Normalize(),  #деление значений пикселя на 255
  ]
  return albu.Compose(train_transform)
def get_validation_augmentation():
  test_transform = [
    albu.Resize(256, 256),
    albu.Normalize()
  ]
  return albu.Compose(test_transform)
```

С помощью параметров Resize, HorizontalFlip, ShiftScaleRotate и Normalize происходит изменение размера и положения входного изображения, а также произвольное применение аффинных преобразований.

В качестве метрики качества модели использовалось IoU (Intersection over Union) – число от 0 до 1, показывающее, насколько у двух объектов – эталонного и текущего – совпадает внутренний «объём». В качестве оптимизатора выбрана адаптивная оценка момента – Adam (Adaptive moment estimation). Она отлично подходит для обучения глубоких нейронных сетей, так как быстро сходится и находит верное направление, в котором должно происходить обновление параметров. Применение метрик и оптимизатора отражено в листинге 3.4.

Листинг 3.4 – Оптимизатор, метрики

```
loss = smp.utils.losses.DiceLoss()
metrics = [
  smp.utils.metrics.IoU(threshold=0.5),
]
optimizer = torch.optim.Adam([
  dict(params=model.parameters(), lr=0.0001),
])
```

Далее была обучена модель на 30 эпохах и сохранена лучшая модель. Запуск обучения модели представлен в листинге 3.5.

Листинг 3.5 – Запуск обучения модели на 30 эпохах

```
max_score = 0
for i in range(0, 30):
  print('\nEpoch: {}'.format(i))
  train_logs = train_epoch.run(train_loader)
  valid_logs = valid_epoch.run(valid_loader)
  if max_score < valid_logs['iou_score']:
    max_score = valid_logs['iou_score']
    torch.save(model, './drive/MyDrive/best_model_test31.pth')
    print('Model saved!')
```

Результаты обучения на тестовой выборке представлены на рисунке 3.12.

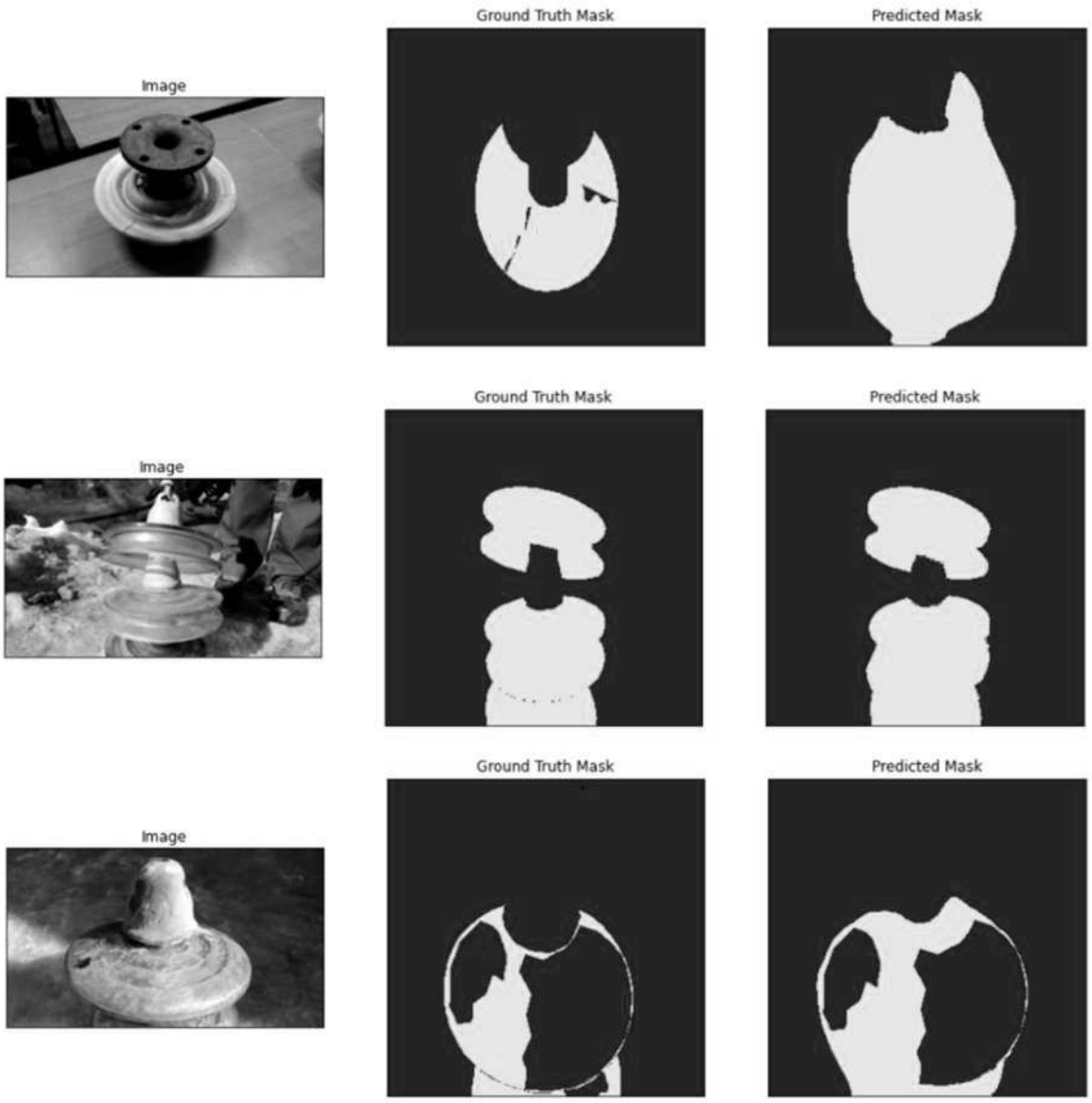


Рисунок 3.12 – Результаты обучения на тестовой выборке

Точность модели приведена в таблице 3.1. Результаты обучения на тестовой выборке представлены на рисунке 3.13.

Таблица 3.1 – Точность модели после обучения

	Train	Valid
IoU score	86,5%	69%
Dice loss	12,9%	25,9%

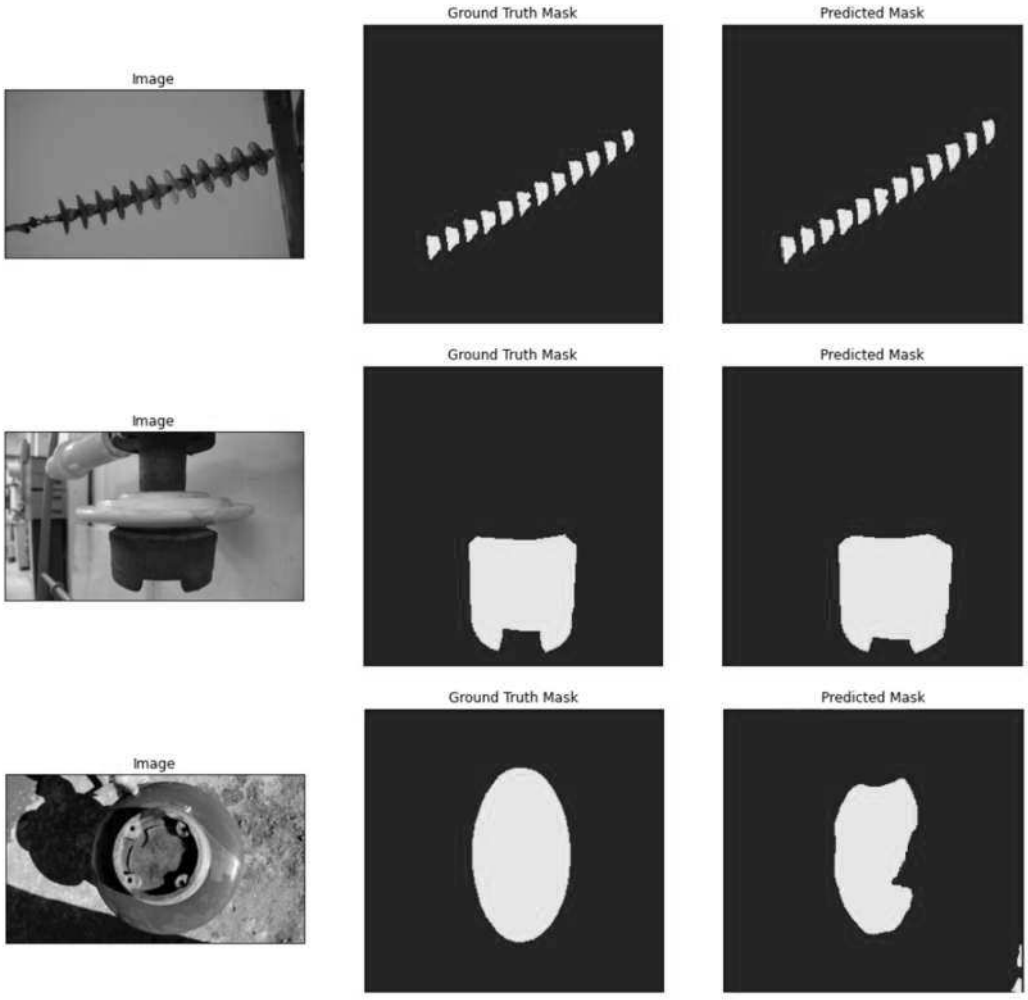


Рисунок 3.13 – Результаты обучения на тестовой выборке

После обучения модели точность составила 69%, что не является высоким показателем. В данном случае небольшая точность сегментации объясняется нехваткой обучающего набора и отсутствием большого количества времени на предварительную обработку обучающей выборки, а также нехваткой времени и ресурсов на само обучение модели, которое требует иметь в распоряжении более мощный компьютер.

Тем не менее, этого достаточно на текущем этапе для дальнейшего развития продукта. В дальнейшем планируется работа над предварительной обработкой изображений в обучающей выборке, её расширение, а также работа над распознаванием не только по фотографиям, но и по видео, что позволит проводить анализ состояния изоляции ЛЭП ещё быстрее.

4 ЭКОНОМИЧЕСКАЯ ЦЕЛЕСООБРАЗНОСТЬ ПРОЕКТА

Для расчета инвестиционной эффективности проведён сравнительный технико-экономический анализ двух методов диагностики состояния воздушных линий электропередач (ВЛЭП):

1) пешие обходы с верховым осмотром;

2) обследование линий методом распознавания с применением беспилотных летательных аппаратов (БПЛА).

В таблице 4.1 представлено экономическое сравнение двух способов контроля состояния ВЛЭП.

Таблица 4.1 – Экономическое сравнение

Параметр	**Пешая группа**	**БПЛА «Геоскан-401»**
Численность рабочей группы, чел.	3	2
Заработная плата, руб., в т.ч.:		
1. Человека в месяц [19]	41 490	41 490
2. Группы в день	5 927	3 951
Число рабочих часов в день, ч.	8	5
Скорость осмотра, км, в т.ч.:		
1. В час	2	24
2. В день *	16	72
Стоимость БПЛА, руб. [20]		1 650 000
Стоимость эксплуатации БПЛА в течении 1 дня, руб.		2 200
Стоимость эксплуатации БПЛА при осмотре 1 км, руб.		31
Общая стоимость осмотра 1 км, руб.	370	86
Время осмотра 1000 км, дней	63	14
Стоимость осмотра 1000 км, руб.	370 000	86 000

* Дальность полета указана для полета в обе стороны.

В таблице 4.2 представлена окупаемость проекта. Окупаемость проекта применительна к общей протяженности ВЛ-110 кВ филиала ОАО «МРСК Урала» – Челябэнерго.

Таблица 4.2 – Окупаемость проекта

Параметр	**Пешая группа**	**БПЛА «Геоскан-401»**
Протяженность ВЛ-110 кВ, км.	5 400	5 400
Время осмотра, дней	340	76
Стоимость осмотра, руб.	1 998 000	464 400
Окупаемость, лет		2,1

Рассчитав окупаемость проекта, можно сделать вывод, что использование беспилотных летательных аппаратов в несколько раз экономически эффективнее обследования наземной группой. Использование БПЛА рациональнее в труднодоступных районах с большой протяженностью линий, что характерно для Челябинской области.

Беспилотные летательные аппараты способны преодолевать расстояния до нескольких десятков километров в день, что значительно сокращает время на поиск повреждения. К тому же, результаты диагностики ВЛ при помощи БПЛА надёжнее, поскольку производится высококачественная запись фото и видео, и в случае необходимости их результаты можно воспроизвести в любое время.

ЗАКЛЮЧЕНИЕ

Рассмотренные в работе современные методы наземного профилактического контроля состояния ВЛ обладают значительной трудоемкостью, часто экономически невыгодны, небезопасны и в ряде случаев практически невозможны. Подобные способы оценки технического состояния ВЛ устарели.

Перспективным направлением обновления методов контроля состояния ВЛ является создание «летающей лаборатории», способной реализовать рассмотренный в работе вид диагностики: контроль состояния электрооборудования методом распознавания с помощью фото и видеосъёмки с применением беспилотных летательных аппаратов и роботизированных комплексов.

Создание подобных комплексов и систем позволяет автоматизировать процессы, которые ранее требовали человеческого участия. Беспилотные летательные аппараты способны работать в круглосуточном режиме практически в любую погоду. Беспилотные технологии помогут повысить качество мониторинга в энергосетевом комплексе, сократить количество ошибок и оперативнее реагировать на внештатные ситуации.

В ходе работы была спроектирована система распознавания состояния изоляции электрооборудования и определены функциональные и нефункциональные требования к ней, а также смоделирован прототип интерфейса и обозначен его базовый функционал.

Были изучены свёрточные нейронные сети и семантическая сегментация. Была создана и размечена небольшая выборка, с помощью которой проводилось обучение и тестирование нейронной сети. Задача сегментации частей изолятора и возможных неисправностей решалась с помощью архитектуры U-Net с кодером ResNet34. Точность сегментации достигла 69%. Небольшая точность сегментации – следствие нехватки обучающего набора и времени на предварительную обработку обучающей выборки, а также времени и ресурсов на само обучение модели, которое требует иметь в распоряжении более мощный компьютер.

На основе проведенного экономического сравнения вариантов можно сказать, что использование программного комплекса вкупе с БПЛА экономически эффективнее обследования наземной группой. Себестоимость обследования 1000 километров ВЛ при помощи беспилотного летательного аппарата по сравнению с обследованием бригадами ниже на 75%, что ещё раз подтверждает выгодность использования беспилотного летательного аппарата.

Анализ погодных условий не выявил серьезных ограничений для применения беспилотного аппарата.

Существенное ограничение применения БПЛА – недостаточно развитая законодательная база и многоэтапная процедура получения разрешений на полёт. Возможно, в скором процедура получения разрешения на использование воздушного пространства беспилотными летательными аппаратами будет значительно упрощена, так как законодательство в данной области постепенно меняется.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Приказ Министерства энергетики РФ от 25 октября 2017 г. №1013. [Электронный ресурс] URL: https://rulaws.ru/acts/Prikaz-Minenergo-Rossii-ot-25.10.2017-N-1013/ (дата обращения: 10.04.2021).
2. Интеллектуальные сети Smart Grid. [Электронный ресурс] URL: http://www.sicon.ru/about/articles/?base=&news=16/ (дата обращения: 11.04.2021).
3. Статистика повреждаемости линейной изоляции в сетях России. [Электронный ресурс] URL: http://np-esi.ru/wp-content/uploads/2017/04/Primenenie-dugootvodjashhih-ustrojstv.pdf (дата обращения: 11.04.2021).
4. РД34.20.504-94. Типовая инструкция по эксплуатации воздушных линий электропередачи напряжением 35-800 кВ. [Электронный ресурс] URL: https://znaytovar.ru/gost/2/RD_342050494_Tipovaya_instrukc.html (дата обращения: 16.04.2021).
5. Fully Convolutional Architectures for Multi-Class Segmentation in Chest Radiographs. [Электронный ресурс] URL: https://arxiv.org/abs/1701.08816 (дата обращения: 20.04.2021).
6. Автономный транспорт. [Электронный ресурс] URL: https://ru.wikipedia.org/wiki/Автономный_транспорт (дата обращения: 20.04.2021).
7. Умные парковки Интерсвязи. [Электронный ресурс] URL: https://uralpress.ru/news/obshchestvo/kompaniya-intersvyaz-rasshiryaet-virtualnyy-servis-umnye-parkovki (дата обращения: 21.04.2021).
8. TensorFlow – An end-to-end open-source platform for machine learning. [Электронный ресурс] URL: https://www.tensorflow.org/ (дата обращения: 24.04.2021).
9. Keras – A deep learning API written in Python. [Электронный ресурс] URL: https://github.com/keras-team/keras (дата обращения: 24.04.2021).
10. OpenCV – A real-time optimized Computer Vision library. [Электронный ресурс] URL: https://opencv.org/ (дата обращения: 26.04.2021).

11. PyTorch – A library for deep learning. [Электронный ресурс] URL: https://pytorch.org/ (дата обращения: 27.04.2021).

12. UML для бизнес-моделирования: зачем нужны диаграммы процессов. [Электронный ресурс] URL: https://evergreens.com.ua/ru/articles/uml-diagrams.html (дата обращения: 2.05.2021).

13. Свёрточная нейронная сеть. [Электронный ресурс] URL: https://ru.wikipedia.org/wiki/Свёрточная_нейронная_сеть (дата обращения: 7.05.2021).

14. Худайберганов, Т. Р. Математические методы распознавания образов / Т. Р. Худайберганов, Х. С. Адинаев, М. А. Артикбаев. // Техника. Технологии. Инженерия. – 2017. – № 2.1 (4.1). – С. 45-47. [Электронный ресурс] URL: https://moluch.ru/th/8/archive/57/2318/ (дата обращения: 9.05.2021).

15. Supervisely – Web platform for computer vision, annotation, training and deploy. [Электронный ресурс] URL: https://supervise.ly/ (дата обращения: 10.05.2021).

16. U-Net: Convolutional Networks for Biomedical Image Segmentation. [Электронный ресурс] URL: https://arxiv.org/pdf/1505.04597.pdf (дата обращения: 15.05.2021).

17. Литвинов С. ResNet (34, 50, 101): «остаточные» CNN для классификации изображений / С. Литвинов // Neurohive. [Электронный ресурс] URL: https://neurohive.io/ru/vidy-nejrosetej/resnet-34-50-101/ (дата обращения: 15.05.2021).

18. ImageNet – An image database. [Электронный ресурс] URL: https://image-net.org/ (дата обращения: 15.05.2021).

19. Федеральная служба государственной статистики [Электронный ресурс] URL: https://rosstat.gov.ru/ (дата обращения: 29.05.2021).

20. ГК Геоскан. Беспилотные технологии для профессионалов Geoscfn производитель беспилотников в России. [Электронный ресурс] URL: https://www.geoscan.aero/ru/products/geoscan401 (дата обращения: 29.05.2021).

ПРИЛОЖЕНИЕ А

Трансформация файлов в текстовом формате JSON в маски

```
import PIL
from PIL import Image, ImageDraw
import json
import cv2
import os
import zipfile
zf = zipfile.ZipFile("D:\\IsolatorSegmentation600\\content\\ann.zi
p", "r")
zf.extractall()
zf = zipfile.ZipFile("D:\\IsolatorSegmentation600\\content\\img.zi
p", "r")
zf.extractall()
root = "D:\\IsolatorSegmentation600\\content\\ann"
img_root = "D:\\IsolatorSegmentation600\\content\\img"
col-
ors = {"header": "#2B3097", "plate": "#43D1B1", "kernel": "#8B572A
", "crack": "#D0021B", "chip": "#8A1149", "reflow": "#1D0074", "ov
erlap": "#F5A623", "brokenglass": "#49AEF0"}
rootlist = os.listdir(root)
savefolder = "D:\\IsolatorSegmentation600\\content\\masks"
for filename in rootlist:
    with open(os.path.join(root, filename), 'r') as f:
        data = json.load(f)
    objects = data['objects']
    image = Image.open(os.path.join(img_root, filename[:-5]))
    mask = Image.new('RGB', size=image.size)
    draw = ImageDraw.ImageDraw(mask)
    for obj in objects:
        XY = obj['points']['exterior']
        XY = [tuple(i) for i in XY]
        draw.polygon(XY, fill=colors[obj['classTitle']])
    mask.save(os.path.join(savefolder, filename[:-8] + 'png'))
```

ПРИЛОЖЕНИЕ Б

Алгоритм анализа изображений с помощью нейронной сети

```
!pip install pretrainedmodels==0.7.4
!pip install segmentation_models_pytorch
!pip install -U git+https://github.com/albu/albumentations
import os
import random
import zipfile
import shutil
import segmentation_models_pytorch as smp
import numpy as np
import matplotlib.pyplot as plt
import PIL
import cv2
import albumentations as albu
import torch
import torch.nn as nn
from PIL import Image, ImageDraw
from albumentations.pytorch.transforms import ToTensor
from segmentation_models_pytorch.encoders import get_preprocessing
_fn
from torch.utils.data import DataLoader
from torch.utils.data import Dataset as BaseDataset
from google.colab import drive
drive.mount('/content/drive', force_remount=True)

with zipfile.ZipFile('/content/drive/My Drive/IsolatorSegmentation
.zip', 'r') as zip_ref: zip_ref.extractall('/content/')
!mkdir /content/dataset/
!mkdir /content/dataset/train
!mkdir /content/dataset/val
!mkdir /content/dataset/test
!mkdir /content/dataset/train/images
!mkdir /content/dataset/val/images
!mkdir /content/dataset/test/images
!mkdir /content/dataset/train/masks
!mkdir /content/dataset/val/masks
!mkdir /content/dataset/test/masks
list_folders = ['content']
for folder in list_folders:
  list_images = os.listdir(os.path.join('/content/IsolatorS
egmentation/', folder, 'img'))
```

```
  random.shuffle(list_images)
  length = len(list_images)
  train = list_images[:int(0.8 * length)]
  val = list_images[int(0.8 * length):int(0.9 * length)]
  test = list_images[int(0.9 * length):]
  for im in train:
    shutil.copyfile(os.path.join('/content/IsolatorSegmentation',
fold-
er, 'img', im), os.path.join('/content/dataset/train/images', fold
er+im))
    shutil.copyfile(os.path.join('/content/IsolatorSegmentation',
folder, 'masks', im[:-
3] +'png'), os.path.join('/content/dataset/train/masks', folder+im
[:-3] +'png'))
  for im in val:
    shutil.copyfile(os.path.join('/content/IsolatorSegmentation',
fold-
er, 'img', im), os.path.join('/content/dataset/val/images', folder
+im))
    shutil.copyfile(os.path.join('/content/IsolatorSegmentation',
folder, 'masks', im[:-
3] +'png'), os.path.join('/content/dataset/val/masks', folder+im[:
-3] +'png'))
  for im in test:
    shutil.copyfile(os.path.join('/content/IsolatorSegmentation',
fold-
er, 'img', im), os.path.join('/content/dataset/test/images', folde
r+im))
    shutil.copyfile(os.path.join('/content/IsolatorSegmentation',
folder, 'masks', im[:-
3] +'png'), os.path.join('/content/dataset/test/masks', folder+im[
:-3] +'png'))
def get_training_augmentation():
  train_transform = [
    albu.Resize(256, 256),  #изменение размера
    albu.HorizontalFlip(p=0.5),  #разворот по горизонтали вокруг о
си Y с вероятностью 50%
    al-
bu.ShiftScaleRotate(scale_limit=0.1, rotate_limit=0, shift_limit=0
.03, p=0, border_mode=0),  #аффинные преобразования
      #scale_limit - диапазон коэффициентов масштабирования
      #rotate_limit - диапазон вращения
```

```
      #shift_limit –
 диапазон коэффициентов сдвига для высоты и ширины
      #p – вероятность применения преобразований
      #border_mode – метод экстраполяции пикселей
   albu.Normalize(),  #деление значений пикселя на 255
  ]
  return albu.Compose(train_transform)

def get_validation_augmentation():
  test_transform = [
    albu.Resize(256, 256),
    albu.Normalize()
  ]
  return albu.Compose(test_transform)

def to_tensor(x, **kwargs):
  print(x.shape)
  return x.transpose(2, 0, 1).astype('float32')

def get_preprocessing(preprocessing_fn):
  _transform = [
    albu.Lambda(image=preprocessing_fn),
    albu.Lambda(image=to_tensor, mask=to_tensor),
  ]
  return albu.Compose(_transform)

col-
ors = {"kernel": "#8B572A", "chip": "#8A1149", "reflow": "#1D0074"
, "overlap": "#F5A623", "brokenglass": "#49AEF0", "plate": "#43D1B
1", "crack": "#D0021B", "header": "#2B3097"}
class Dataset(BaseDataset):
  def __init__(self, images_dir, masks_dir, augmentation=None, pre
processing=None,):
    self.names = os.listdir(images_dir)
     self.images_paths = [os.path.join(images_dir, name) for name i
n self.names]
    self.masks_paths = [os.path.join(masks_dir, name[:-
3]+'png') for name in self.names]
    self.augmentation = augmentation
    self.preprocessing = preprocessing
    self.class_values = set(colors.values())
    self.class_values = [tuple(int(color.lstrip('#')[i:i+2], 16) f
or i in (0, 2, 4)) for color in self.class_values]
```

```
   def __getitem__(self, i):
    image = cv2.imread(self.images_paths[i])
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    mask = cv2.imread(self.masks_paths[i])
    mask = cv2.cvtColor(mask, cv2.COLOR_BGR2RGB)
     masks = np.zeros((mask.shape[0], mask.shape[1], 8))
    for j in range(mask.shape[0]):
      for p in range(mask.shape[1]):
        for r in range(len(self.class_values)):
          if mask[j, p, 0]== self.class_values[r][0] and mask[j, p
, 1] == self.class_values[r][1] and mask[j, p, 2] == self.class_va
lues[r][2]:
            masks[j, p, r] = 1
            break
    mask = masks
    if self.augmentation:
      sample = self.augmentation(image=image, mask=mask)
      image, mask = sample['image'], sample['mask']
    if self.preprocessing:
      sample = self.preprocessing(image=image, mask=mask)
      image, mask = sample['image'], sample['mask']
    return image, mask

  def __len__(self):
    return len(self.names)

x_train_dir = '/content/dataset/train/images'
y_train_dir = '/content/dataset/train/masks'

x_valid_dir = '/content/dataset/val/images'
y_valid_dir = '/content/dataset/val/masks'

x_test_dir = '/content/dataset/test/images'
y_test_dir = '/content/dataset/test/masks'

data-
ta-
set = Dataset(x_train_dir, y_train_dir, augmentation=get_training_
augmentation())

def visualize(class_num = 0, **images):
  n = len(images)
  plt.figure(figsize=(16, 5))
```

```
  for i, (name, image) in enumerate(images.items()):
    plt.subplot(1, n, i + 1)
    plt.xticks([])
    plt.yticks([])
    plt.title(' '.join(name.split('_')).title())
    if i == 0:
      plt.imshow(image)
    else:
      plt.imshow(image[..., class_num])
  plt.show()

image, mask = dataset[1]

visualize(
  class_num = 1,
  image=image,
  cars_mask=mask,
)

de-
vice = torch.device("cuda:0" if torch.cuda.is_available() else "cp
u")

ENCODER = 'resnet34'
ENCODER_WEIGHTS = 'imagenet'

preprocess-
cess-
ing_fn = smp.encoders.get_preprocessing_fn(ENCODER, ENCODER_WEIGHT
S)

model = smp.Unet(
  encoder_name=ENCODER,
  encoder_weights=ENCODER_WEIGHTS,
  classes=8,
  activation='sigmoid'
)
model.to(device)
optimiz-
er = torch.optim.Adam([dict(params=model.parameters(), lr=0.0001),
])
loss = smp.utils.losses.DiceLoss()
metrics=[smp.utils.metrics.IoU(threshold=0.5)]
```

```
train_dataset = Dataset(
  x_train_dir,
  y_train_dir,
  augmentation=get_training_augmentation(),
  preprocessing=get_preprocessing(preprocessing_fn),
)
valid_dataset = Dataset(
  x_valid_dir,
  y_valid_dir,
  augmentation=get_validation_augmentation(),
  preprocessing=get_preprocessing(preprocessing_fn),
)

train_loader = DataLoader(train_dataset, batch_size=1, shuffle=Tru
e)
val-
id_loader = DataLoader(valid_dataset, batch_size=1, shuffle=False)

train_epoch = smp.utils.train.TrainEpoch(
  model,
  loss=loss,
  metrics=metrics,
  optimizer=optimizer,
  device=device,
  verbose=True,
)

valid_epoch = smp.utils.train.ValidEpoch(
  model,
  loss=loss,
  metrics=metrics,
  device=device,
  verbose=True,
)

max_score = 0
for i in range(0, 50):
  print('\nEpoch: {}'.format(i))
  train_logs = train_epoch.run(train_loader)
  valid_logs = valid_epoch.run(valid_loader)

  if max_score < valid_logs['iou_score']:
    max_score = valid_logs['iou_score']
    torch.save(model, './drive/MyDrive/my_model_50epoch.h5')
```

```
    print('Model saved!')

test_dataset = Dataset(
  x_test_dir,
  y_test_dir,
  augmentation=get_validation_augmentation(),
  preprocessing=get_preprocessing(preprocessing_fn),
)
test_loader = DataLoader(test_dataset, batch_size=1, shuffle=False
)
test_epoch = smp.utils.train.ValidEpoch(
  model=model,
  loss=loss,
  metrics=metrics,
  device=device,
)
logs = test_epoch.run(test_loader)
import torch.nn.functional as F
img, mask = next(iter(test_loader))
img = img.to(device)
output = model(img)
output = output.squeeze()
print(output.shape)
plt.imshow(output.detach().cpu().numpy().transpose(1,2,0)[..., 3])
plt.show()

test_dataset_vis = Dataset(
  x_test_dir, y_test_dir
)
channel = 3
for i in range(10):
  n = np.random.choice(len(test_dataset))
  image_vis = test_dataset_vis[n][0].astype('uint8')
  image, gt_mask = test_dataset[n]
  gt_mask = gt_mask.squeeze().transpose(1,2,0)
  x_tensor = torch.from_numpy(image).to(device).unsqueeze(0)
  mask = model.predict(x_tensor)
  mask = (mask.squeeze().cpu().numpy().round().transpose(1,2,0))
  visualize(
    class_num = 4,
    image=image_vis,
    ground_truth_mask=gt_mask,
    predicted_mask=mask
  )
```