

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

Направление подготовки: 09.04.01 «Информатика и вычислительная техника»

Магистерская программа: «Прикладной искусственный интеллект»

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

на тему

СИСТЕМА АНАЛИЗА И РАСПОЗНАВАНИЯ
ИЗОБРАЖЕНИЙ

Студент

Мефокос Ярослав Сергеевич

(подпись, дата) (ФИО полностью)

30.05.2025

Руководитель

Карамышева Н.С.

(подпись, дата) 30.05.2025

(фамилия, инициалы)

Нормоконтролёр

30.05.2025

Бычков А.С.

(подпись, дата)

(фамилия, инициалы)

Рецензент

зав. каф. Информатика ПензГТУ

(должность, место работы)

(подпись, дата) (фамилия, инициалы)

30.05.2025

Мартышкин А.И.

Работа допущена к защите (протокол зас. кафедры от 28.05.2025 № 16)

Заведующий кафедрой

М.А. Митрохин

(подпись)(фамилия, инициалы)

Работа защищена с отметкой отлично (протокол заседания ГЭК от

5 июня 2025г.

№

7

Секретарь ГЭК

Е.И. Гурин

(подпись)(фамилия, инициалы)

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
М.А. Митрохин
« 29 » 09 2023 г.

**ЗАДАНИЕ
НА МАГИСТЕРСКУЮ ДИССЕРТАЦИЮ**

1. Студент Мефоков Ярослав Сергеевич гр. 23ВВИМ1 факультета вычислительной техники, направления 09.04.01

2. Тема работы «Система анализа и распознавания изображений».

Тема утверждена приказом ПГУ № 986/р от " 29 " 09 2023 г.

3. Руководитель магистерской диссертации д.т.н доцент Карамышева Н.С.

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

Назначение разработки: Разработка системы анализа и распознавания спутниковых изображений с использованием модели машинного обучения DeepLabV3Plus. Система будет предназначена для семантической сегментации спутниковых снимков, что позволит применять ее в таких областях, как мониторинг окружающей среды, сельское хозяйство и управление земельными ресурсами.

Исходные данные: Наборы данных спутниковых изображений, включая ногоспектральные снимки и соответствующие маски для семантической сегментации.

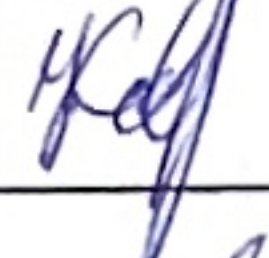
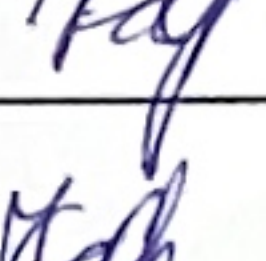
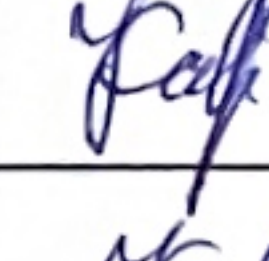
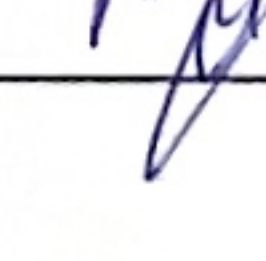
5. Перечень подлежащих разработке вопросов:

1. Какие методы предобработки данных наиболее эффективны для улучшения качества семантической сегментации спутниковых изображений?

2. Какие параметры модели (например, количество слоев, размер фильтров) оказывают наибольшее влияние на точность распознавания объектов на спутниковых изображениях?

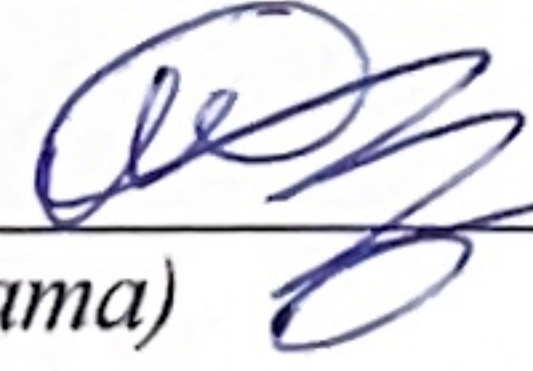
3. Какие сценарии использования являются наиболее актуальными?

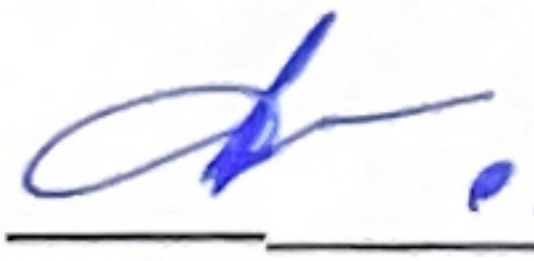
6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и постановка задачи	10%	29.09.2023-21.11.2023	
2	Изучение современных методов анализа изображений и выбор архитектуры модели	10%	22.11.2023-22.12.2023	
3	Сбор и подготовка обучающих данных, создание конвейера обработки	15%	23.12.2023-23.03.2024	
4	Реализация и обучение модели DeepLabV3Plus	20%	24.03.2024-24.06.2024	
5	Оптимизация и тестирование модели	10%	25.06.2024-25.08.2024	
6	Разработка модуля ГИС и интеграция с основной системой	15%	26.08.2024-26.11.2024	
7	Создание пользовательского интерфейса и тестирование системы	10%	27.11.2024-27.03.2025	
8	Оформление диссертации и подготовка к защите	10%	28.03.2025-31.05.2025	

Дата выдачи задания "29" сентября 2023г.

Руководитель магистерской диссертации  29.09.2023 Карамышева Н.С.
(подпись, дата)

Задание к исполнению принял студент  29.09.2023 Мефоков Я.С.
(подпись, дата)

Работу к защите допустить
Декан факультета  07.06.2025 Фионова Л.Р.
(подпись, дата)

РЕФЕРАТ

Пояснительная записка содержит 133 листов, 73 рисунков, 28 источников и 3 приложения

ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ, МЕТОДЫ ПРОЕКТИРОВАНИЯ, НЕЙРОННЫЕ СЕТИ, МАШИННОЕ ОБУЧЕНИЕ, DeepLabV3

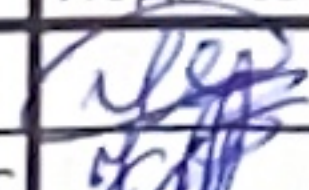
Целью проекта является разработка программного обеспечения, позволяющего применять нейронные сети для анализа и распознавания изображений.

Объектом исследования является процесс обучения нейронных сетей.

Предметом исследования является программное обеспечение для анализа и распознавания изображений, в основу работы которой заложена модель машинного обучения DeepLabV3Plus.

Технологии разработки: DeepLabV3Plus, Python.

Результат работы: спроектированное и разработанное приложение, позволяющее обучать нейронные сети для анализа и распознавания изображений.

					ПГУ 1.090401.22.001 ПЗ						
Изм.	Лист	№ докум.	Подпись	Дата	Система анализа и распознавания изображений Пояснительная записка	Лит.			Лист	Листов	
Разраб.		Мефоков Я.С.		30.05.25					2	133	
Провер.		Карамышева Н.С.		30.05.25							
Н. Контр		Бычков А.С.		30.05.25							
Утверд.		Митрохин М.А.		30.05.25							
						ФВТ, каф. ВТ гр. 23ВВИм1					

Содержание

ВВЕДЕНИЕ	6
1 Предметная область.....	7
1.1 Анализ построения и функционирования искусственных нейронных сетей.....	7
1.1.1 Принципы строения искусственного нейрона	7
1.2 Задача сегментации изображения.....	11
1.3 Методы семантической сегментации изображения основные на CNN.....	12
1.3.1 FCN.....	12
1.3.2 U-Net.....	12
1.3.3 SegNet.....	14
1.3.4 PSPNet	14
1.3.5 RefineNet	15
1.3.6 DeepLab	15
1.3.7 DenseASPP	16
1.4 Оптимизация нейронной сети.....	18
1.4.1 Градиентный спуск	19
1.4.2 SGD.....	19
1.4.3 Метод моментов.....	19
1.4.4 Адаптивный градиент.....	19
1.4.5 RMSProp	19
1.4.6 Adam.....	19
1.5 Сегментируемые данные	20
1.5.1 Спутниковые снимки.....	20
1.5.2 Требования к данным	21
1.5.3 Набор данных	22
1.6 Выбор инструментов.....	26
1.6.1 TensorFlow	26
1.6.2 PyTorch.....	28
1.6.3 Flask.....	29
1.7 Выбор исследуемых методов	30
1.7.1 Модель ResNet101.....	30
1.7.2 Модель DeepLabV3Plus.....	32

1.8 Метрики для оценки качества сегментации	34
2 Требования к программе.....	36
2.1 Функциональные требования.....	36
2.1.1 Требования к подсистеме хранения данных.....	37
2.1.2 Требования к подсистеме визуализации пространственных данных	37
2.1.3 Требования к модулю «ГИС».....	38
2.1.4 Требования к модулю «Интеллектуальная обработка».....	38
2.1.5 Требования к модулю «Пространственные данные»	39
2.1.6 Требования к модулю «Проектирование процессов».....	39
2.2 Не функциональные требования.....	40
2.2.1 Требования к составу, структуре и способам организации данных	40
2.2.2 Требования к программному обеспечению	41
2.2.3 Технические требования	42
2.2.4 Аппаратные требования	45
3 Обучение модели DeepLabV3Plus	46
4 Разработка приложения	51
4.1 Модуль «ГИС».....	51
4.1.1 Карточка атрибутивных данных объекта.....	56
4.2 Модуль «Интеллектуальная обработка».....	62
4.3 Модуль «Пространственные данные».....	74
4.4 Модуль «Проектирование процессов».....	78
5 Тестирование	85
5.1 Тестирование качества обучения моделей	85
5.2 Тестирование системы.....	50
ЗАКЛЮЧЕНИЕ.....	96
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	97
ПРИЛОЖЕНИЕ А (Листинг программы).....	100
ПРИЛОЖЕНИЕ Б (Структура модели DeepLabV3Plus)	107

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ

В данной работе применяют следующие сокращения и обозначения

API (Application programming interface) – набор способов и правил, по которым происходит взаимодействие между программами

Атрибутивные данные – Качественные или количественные характеристики пространственных объектов

Детектирование – Нахождение экземпляров объектов на изображении

Идентификатор сегмента Число от 1 до N, где N – общее количество сегментов

Искусственный интеллект – Способность вычислительной машины моделировать процесс мышления за счет выполнения функций, связанных с человеческим интеллектом

Модуль «ГИС» – Геоинформационная система визуализации результатов интеллектуальной обработки

Модуль «Интеллектуальная обработка» – Подсистема управления процессами обработки фотограмметрических данных

Модуль «Проектирование процессов» – Подсистема проектирования процессов обработки фотограмметрических данных

Модуль «Пространственные данные» Подсистема управления источниками данных, используемые при инициализации процессов обработки фотограмметрических данных

Ортогонализация – Процесс построения по заданному базису линейного пространства некоторого ортогонального базиса, имеющий ту же самую линейную оболочку

Семантическая сегментация – объектов Разбиение изображения на объекты с определением их типов

Слой на карте – Механизм, используемый для отображения географических наборов данных в геопортале Платформы. Слои, лежащие друг на друге, составляют карту.

СУБД – Система управления базами данных

ВВЕДЕНИЕ

Мир вступает в новую информационную эпоху, которая связана с наступлением повсеместной цифровизацией всех сторон жизни современного общества. Сегодня цифровая среда проникает почти во все сферы таких как в области профессиональной и повседневной деятельности человека, изменяя их формы и перестраивая сознание человека. Быстрое развитие цифровых технологий увеличивает интерес ученых к проблемам изменений социальной реальности, появления абсолютно новой цифровой и финансовой культуры, изменения мира человека.

Анализ и обработка изображений – это самое перспективное направление в развитии цифровых технологий. Анализ изображений может быть применен во многих сферах жизни человека от анализа медицинских сканов, до анализа космических снимков. В последнее время стремительно возрос интерес к цифровым, электронным, и оптическим методам обработки изображений, для улучшения их качества. Большое освещение получили работы, связанные с биомедицинскими и космическими исследованиями. Другие применения – это аэрофотосъемка и промышленная радиографию.

Программное зрение начинает использоваться в новых сферах, постоянно расширяется и требует, во-первых, разработки новых алгоритмов и методов обработки изображений, во-вторых, в теоретическом анализе полученных результатов.

Целью данной работы является реализация системы, позволяющей анализировать и распознавать изображения с помощью выбранной модели машинного обучения.

1 Предметная область

1.1 Анализ принципов построения и функционирования искусственных нейронных сетей

Искусственная нейронная сеть (ИНС) – математическая модель, которая имитирует работу биологических нейронных сетей. ИНС состоит из множества элементов взаимосвязанных между собой – нейронов, которые принимают входящую информацию, обрабатывают и передают ее друг другу, для получения некоего результата [1].

ИНС могут быть применены в следующих сферах:

- классификация. Способность классифицировать объекты по определенным признакам. Пример, определение, представлено ли на изображении собака или кошка,
- распознавание образов. Возможность распознавать образы, такие как дорожные знаки, лица людей, или другие объекты,
- прогнозирование. ИНС может быть использована для прогнозирования будущих событий на основе исторических данных. Например, можно предсказать рост цен на акции или изменение погоды по предыдущим данным,
- автоматическое управление. ИНС может быть использована для автоматического управления системами, такими как беспилотные автомобили, промышленные роботы,
- обработка естественного языка. ИНС также может применяться для обработки речи и текстов, позволяя компьютерам понимать и генерировать естественный язык.

1.1.1 Принципы строения искусственного нейрона

Искусственный нейрон – узел искусственной нейронной сети, имитирующий работу биологического нейрона. Искусственные нейроны являются основной и составной частью искусственных нейронных сетей.

Из структуры строения искусственного нейрона выделяют следующие:

– входы. Искусственный нейрон через некоторое количество входов, получает данные на входе. Входные данные могут быть представлены в виде значений или чисел,

– веса. Веса могут быть отрицательными или положительными. Вход нейрона имеет свой вес, который определяет его влияние на выходной сигнал,

– функция активации. Искусственный нейрон использует функцию активации для преобразования данных в выходной сигнал. Функция активации может быть сигмоидальной, линейной, или другой,

– сумматор. Входной сигнал умножается на соответствующий вес и суммируется с остальными входными сигналами. Полученная сумма является аргументом функции активации,

– выход. Выходной сигнал нейрона – это результат функции активации. Он может быть использован другими нейронами в сети или понят как решение задачи,

– порог. Порог может быть настраиваемым или фиксированным. Некоторые искусственные нейронные сети имеют свой порог, который ограничивает выходной сигнал нейрона определенным диапазоном значений.

Искусственная нейронная сеть может адаптироваться к новым данным и условиям путем изменения весов синапсов. Что дает возможность к обучению и решению сложных задач.

Взвешенная сумма его входов определяется, как текущее состояние S нейрона, представленная в формуле 1.

$$S = \sum_{i=1}^n x_i \times w_i, \quad (1)$$

где x_i – значение входа в нейрон;

w_i – вес входа в нейрон;

n – количество входов.

Выход нейрона есть функция Y его состояния, представленная в формуле

$$Y = F(S), \quad (2)$$

На рисунке 1 приведена структура искусственного нейрона.

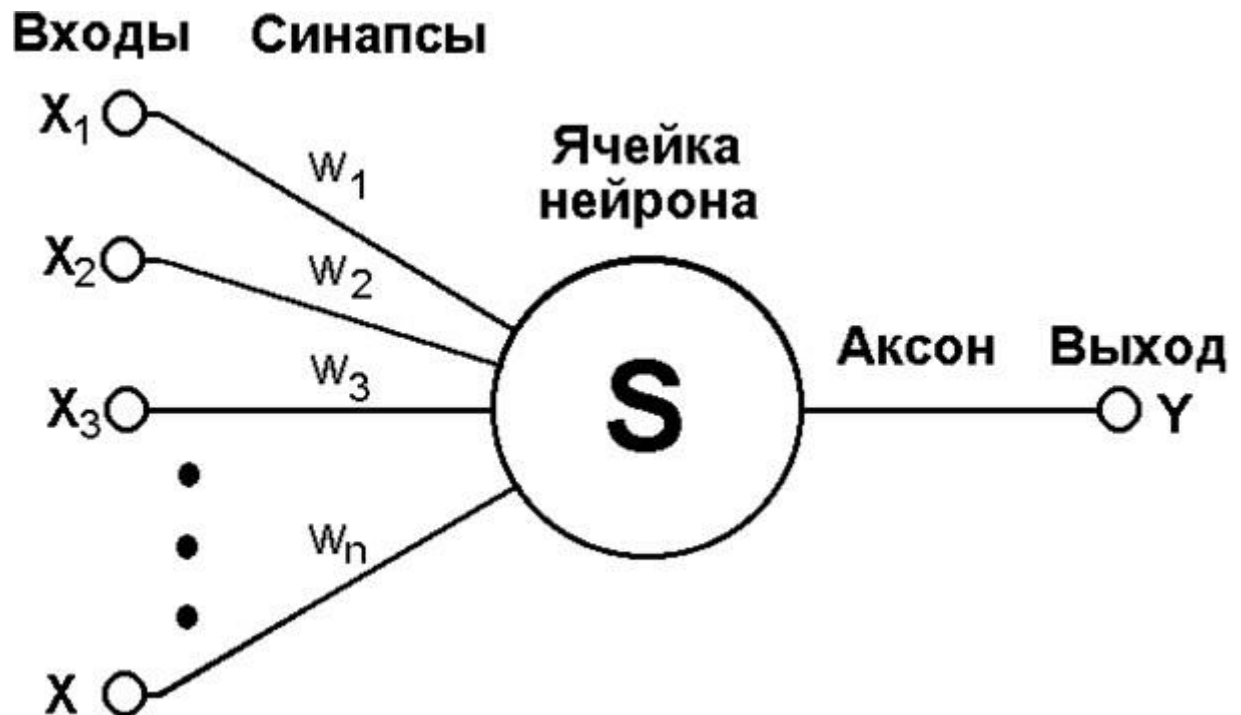


Рисунок 1 – Структура искусственного нейрона

1.1.2 Основы структуры и функционирования нейронной сети ИНС состоит из следующих частей:

- нейроны. Каждый нейрон в сети имеет один выход и несколько входов. Входы получают сигналы от внешних источников или других нейронов, а выход передает сигнал другим нейронам,
- синапсы. Связи между нейронами имеют название синапсы. Синапсы могут быть тормозящими или возбуждающими. Возбуждающие синапсы увеличивают активность нейрона, а тормозящие уменьшают активность,
- веса. Синапсы имеют веса. Веса определяют силу влияния на нейроны. Веса могут быть отрицательными или положительными,

– слои. Нейронные сети могут иметь несколько слоев или один слой. Первый слой, в который поступают входные данные, называется входным слоем. Последний слой, из которого образуются выходные данные, называется выходным слоем. Между ними могут быть скрытые слои, которые обрабатывают данные,

– функция активации. Нейроны используют функцию активации для трансформации сигналов входных в выходные. Функция активации может быть линейной, сигмоидальной или другой.

ИНС обучаются на основе данных с помощью алгоритмов машинного обучения. Чтобы сеть могла правильно обрабатывать входные данные, в процессе обучения подбираются оптимальные веса для синапсов.

Некоторые из нейронных сетей имеют обратную связь, когда выходы нейронов возвращаются на входы. Это позволяет сетям запоминать и воспроизводить последовательности событий.

ИНС могут адаптироваться к новым данным и условиям путем изменения весов синапсов. Это делает их способными к обучению и решению сложных задач.

В отличие от традиционных алгоритмов, ИНС способны к параллельной обработке информации, что значительно ускоряет процесс вычислений. Благодаря нелинейным функциям активации, ИНС могут моделировать сложные зависимости между входными данными и выходными данными.

Выделяют следующие этапы функционирования ИНС:

– обработка данных. Данные проходят через слои нейронов, где они обрабатываются с помощью функций активации. Каждый нейрон получает входные данные от других нейронов и преобразует их в выходные данные. Этот процесс повторяется для всех нейронов в сети,

– обучение. На основе полученных данных ИНС обучается, то есть настраивает свои параметры таким образом, чтобы минимизировать ошибку между ожидаемыми и реальными выходными данными. Обучение может быть выполнено с использованием различных алгоритмов, таких как

метод обратного распространения ошибки, генетических алгоритмов,

– использование. После обучения ИНС может быть использована для решения выбранных задач. Она принимает входные данные задачи и выдает соответствующие выходные данные, которые могут быть интерпретированы в качестве ответа на поставленную задачу.

1.2 Задача сегментации изображения

Сегментация изображений – это задача машинного зрения, включающая разделение изображения на отдельные части, например на группы пикселей, обладающих некими общими атрибутами. То, что выступает в качестве атрибутов, определяется как спецификой предметной области, так и конкретной задачей, но чаще всего основными атрибутами служат интенсивность пикселей, текстура и цвет. Сегментацию изображения можно разделить на три вида [2]:

– сегментация экземпляра. Это вид, при котором для каждого пикселя определяется принадлежность экземпляра объекта. Данный вид обнаруживает каждый отдельный объект интереса на изображении. Например, когда каждый человек на рисунке сегментирован как отдельный объект,

– семантическая сегментация. Это вид, определяющий для каждого пикселя класс принадлежности объекта. Например, когда все люди на рисунке сегментированы как один объект, а фон – как другой объект [3],

– паноптическая сегментация сочетает в себе семантическую и сегментацию экземпляра. Этот подход, определяет для каждого пикселя класс принадлежности. В отличие от семантической сегментации, паноптическая сегментация различает разные экземпляры одного и того же класса.

Выделяют два основных типа нейронных сетей используемых для семантической сегментации изображений:

1) сверточные нейронные сети (CNN). Этот тип нейронной сети может быть использован для анализа изображений. CNN состоит из нескольких слоев, каждый слой выполняет определенные операции над данными входными. Для семантической сегментации используются CNN с несколькими выходами, которые соответствуют различным классам распознаваемых объектов [4, 5],

2) рекуррентные нейронные сети (RNN). Другой тип нейронных сетей, который также может быть использован для семантической сегментации. RNN способны обрабатывать последовательности данных, что делает их подходящими для задач анализа временных рядов. Они так же могут быть использованы для семантической сегментации изображений, так как изображения можно рассматривать как последовательность пикселей.

В рамках данной работы будет использоваться первый тип нейронных сетей, так как планируется работать со слоями изображений, каждый из которых будет содержать соответствующую ему информацию.

1.3 Методы семантической сегментации изображения CNN

1.3.1 FCN

Полносвязанная нейронная сеть представляет собой сверточную нейронную сеть, на каждом уровне свертки которой есть связь между входным уровнем светки и выходным уровнем светки. Это позволяет использовать сеть для семантической сегментации. Данная сеть использует последовательность слоев для преобразования карты признаков в карту сегментированных областей.

1.3.2 U-Net

Архитектура нейронной сети, разработанная для семантической сегментации. U-Net состоит из двух частей: сверточной части, которая уменьшает размер изображения, и деконволюционной части, которая восстанавливает исходное разрешение изображения. Данная структура позволяет U-Net обрабатывать изображения разных размеров и получать точные результаты сегментации [6]. Архитектура U-Net представлена на рисунке 2.

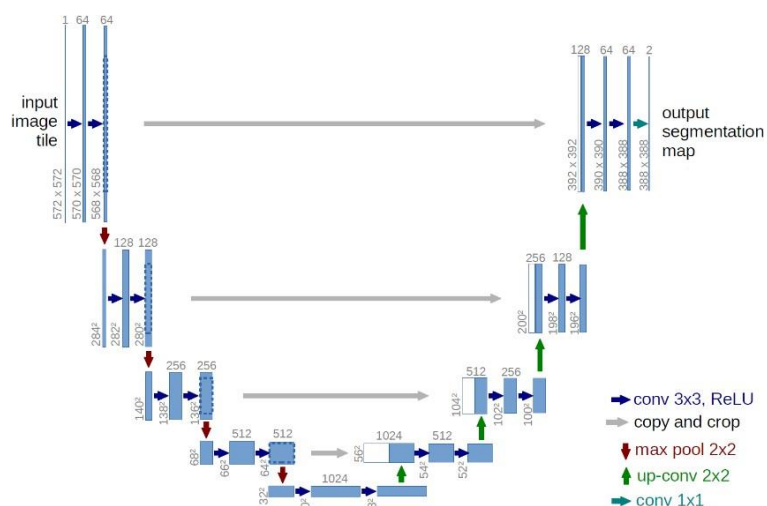


Рисунок 2 – Архитектура U-Net

Сеть U-Net состоит из двух основных частей:

- кодировщик. Часть сети, которая уменьшает размер изображения и извлекает из него признаки. Кодировщик состоит из нескольких слоев свертки и последовательно уменьшающего размер изображения пулинга,

- декодировщик. Часть сети, которая восстанавливает исходное изображение и добавляет детали к нему. Декодировщик также состоит из слоев свертки, но они имеют больший размер ядра и шаг, чтобы восстановить исходное разрешение изображения. В декодере используются транспонированные свертки, для сохранения пространственных отношений между пикселями.

В U-Net сети используются следующие слои:

- пропускные соединения. Соединения между слоями кодировщика и декодировщика, позволяющие передавать информацию о деталях изображения на более высокие уровни. Это в свою очередь помогает улучшить качество сегментации,

- конкатенация. Операция объединения двух тензоров по глубине (третьему измерению). В U-Net конкатенация используется для объединения информации из энкодера с информацией из декодера на каждом уровне,

- функция активации. Функция, которая применяется к каждому нейрону в сети после каждого слоя. В U-Net обычно используется функция ReLU,
- слой пулинга. Слой, который уменьшает размер изображения путем объединения в один соседних пикселей. Пулинг помогает ускорить процесс обучения и уменьшить количество параметров в сети,
- слой свертки. Слой, который применяет фильтр (ядро) к изображению и вычисляет свертку. Слой свертки извлекает из него признаки и уменьшает размерность изображения,
- транспонированная свертка. Перевернутая свертка, которая увеличивает размер изображения и восстанавливает детали. Транспонированные свертки используются для сохранения пространственных отношений между пикселями в декодировщике,
- полносвязный слой. Последний слой в U-Net, преобразует выходные данные в вероятности принадлежности каждого пикселя к определенному классу.

1.3.3 SegNet

Модификация FCN, которая использует кодировщик-декодировщик для повышения эффективности обучения и уменьшения количества параметров. SegNet также использует функцию потерь, основанную на бинарной кросс-энтропии, для обучения модели.

В SegNet используется метод максимального объединения, позволяющий сохранять в процессе уменьшения размера изображения пространственную информацию. Это улучшает качество сегментации по сравнению с другими методами пулинга [7].

1.3.4 PSPNet

Метод семантической сегментации, использует пирамиду пространственных пирамид для обработки изображений разных масштабов. PSPNet объединяет информацию из разных уровней пирамиды для

получения более точных результатов сегментации [8]. Архитектура PSPNet представлена на рисунке 3.

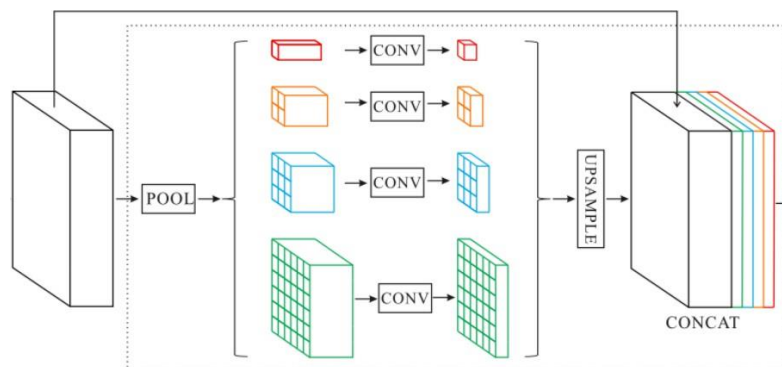


Рисунок 3 – Архитектура PSPNet

Пирамида признаков сети PSPNet состоит из нескольких уровней, каждый из которых содержит информацию о различных масштабах изображения. Это позволяет сети учитывать контекст изображения на разных масштабах и улучшает качество сегментации.

В качестве функции потерь для PSPNet обычно используется функция потерь, которая учитывает как точность сегментации, так и гладкость границ между классами. В качестве оптимизатора, используемого для обновления весов сети в процессе обучения, зачастую применяется оптимизатор Adam.

1.3.5 RefineNet

Метод, который улучшает результаты семантической сегментации путем добавления дополнительных слоев к сверточным нейронным сетям. RefineNet использует последовательность слоев для уточнения результатов сегментации и уменьшения ошибок.

1.3.6 DeepLab

Семейство методов семантической сегментации, основанных на сверточных нейронных сетях. DeepLab для улучшения качества сегментации использует несколько техник [9]:

- atrous свертка. Это метод, который позволяет увеличить поле зрения сверточной нейронной сети без увеличения количества параметров или вычислительных затрат. Atrous свертка использует расширенную матрицу

весов, которая позволяет пропускать пиксели в определенных местах, что приводит к увеличению поля зрения и улучшению качества сегментации,

- использование полносвязанных сверточных слоев для обработки изображений, что позволяет избежать использования полносвязных слоев и уменьшить количество параметров модели. FCN также позволяют выполнять сегментацию изображений произвольного размера,

- пиксельная регрессия. в некоторых версиях DeepLab используется метод пиксельной регрессии для уточнения границ сегментов. Этот метод позволяет улучшить качество сегментации за счет учета информации о соседних пикселях,

- многомасштабная обработка. DeepLab использует многомасштабную обработку для улучшения качества сегментации на изображениях разного разрешения, что достигается путем применения сверток разного размера на разных уровнях сети,

- использование контекстной информации. DeepLab также учитывает контекстную информацию при сегментации изображений, что помогает улучшить результаты на сложных сценах. Контекстная информация может быть получена из предыдущих уровней сети или из других источников,

- обучение с использованием данных аннотирования. Для обучения DeepLab используются данные аннотирования, где каждый пиксель изображения помечен как принадлежащий определенному классу. Это позволяет модели научиться точно сегментировать изображения,

- оптимизация потерь. При обучении DeepLab оптимизируется функция потерь, которая измеряет разницу между предсказанными и истинными метками. Оптимизация функции потерь позволяет модели лучше адаптироваться к данным и улучшать качество сегментации.

1.3.7 DenseASPP

Метод, основанный на использовании сверточных блоков ASPP для извлечения признаков разных масштабов и их объединения с помощью операции конкатенации. DenseASPP позволяет. Архитектура DenseASPP представлена на рисунке 4.

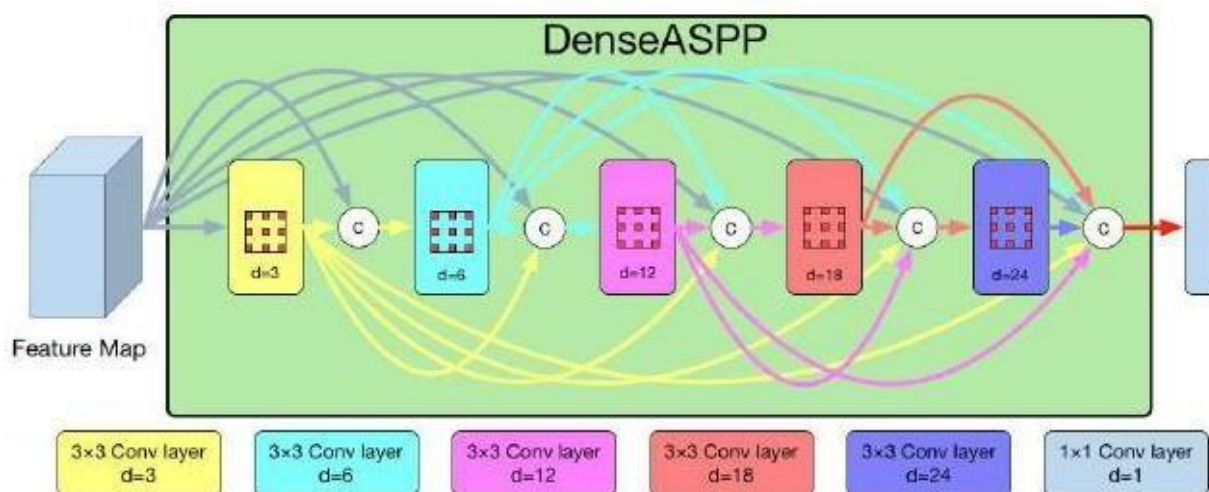


Рисунок 4 – Архитектура DenseASPP

Сверточные блоки ASPP представляют собой набор сверточных слоев, которые принимают на вход объединенные признаки и производят дополнительные признаки, учитывающие пространственные отношения между пикселями [10].

Работа сверточных блоков ASPP состоит из следующих ключевых этапов:

- получение входных данных. На вход сверточным блокам ASPP подаются изображения или тензоры, содержащие информацию о пикселях. Входные данные могут быть предварительно обработаны с помощью других слоев нейронной сети, таких как сверточные слои или слои пулинга,
- свертка с ядром различных размеров. Сверточные блоки ASPP используют ядра разных размеров для свертки входных данных, что позволяет извлекать признаки разных уровней абстракции и масштабов,
- объединение результатов свертки. Результаты свертки из разных ядер объединяются в единый выходной тензор, что позволяет получить представление о пространственных отношениях между пикселями на разных масштабах,

– пирамида признаков. Выходные данные сверточных блоков ASPP формируют пирамиду признаков, где информацию о разных масштабах изображения содержит каждый уровень. Эта пирамида используется для учета контекста изображения на разных уровнях детализации,

– пулинг. После формирования пирамиды признаков происходит объединение информации из разных уровней пирамиды. Пулинг позволяет выделить наиболее важные признаки и уменьшить размерность данных,

– оптимизация. Процесс обучения включает оптимизацию весов нейронной сети с использованием алгоритмов, таких как градиентный спуск, Adam или других. Оптимизация направлена на минимизацию ошибки между предсказанными и истинными классами, границами сегментов,

– классификация или сегментация. Полученные после пулинга данные передаются на последующие слои нейронной сети для классификации или сегментации изображения. Используются полносвязные слои и функции потерь для обучения модели,

– выход. Результатом работы сверточных блоков ASPP является сегментированное изображение или классификация объектов на изображении. От точности извлечения признаков и способности модели учитывать контекст изображения зависит качество сегментации.

Особенностью DenseASPP является использование плотных блоков ASPP, что позволяет улучшить качество сегментации по сравнению с традиционными методами и учитывать пространственные отношения между пикселями.

Таким образом, из всех представленных нейронных сетей типа CNN больший интерес вызывает семейство методов семантической сегментации DeepLab.

1.4 Оптимизация нейронной сети

Оптимизация нейронной сети – процесс настройки параметров модели для достижения максимальной производительности и эффективности. Оптимизация включает в себя изменение весов, смещений, функций активации и других параметров для улучшения точности, скорости или обобщающей способности

модели [17].

1.4.1 Градиентный спуск

Один из наиболее распространенных методов оптимизации, заключается в смещении, в направлении противоположном градиенту функции потерь и последовательном изменении весов. Что позволяет постепенно уменьшать ошибку работы модели.

1.4.2 SGD

Упрощенная версия градиентного спуска, которая обрабатывает данные по одному примеру за раз вместо использования всего набора данных сразу. SGD может быть более эффективным для больших наборов данных.

1.4.3 Метод моментов

Метод, основанный на использовании инерции для ускорения процесса обучения. В методе моментов используется компонент импульса к градиенту, позволяющий запомнить предыдущее направление движения и двигаться быстрее в этом направлении.

1.4.4 Адаптивный градиент

Это метод, адаптирует скорость обучения для каждого параметра отдельно. Когда некоторые параметры изменяются слишком медленно, помогает избежать проблемы затухания градиента.

1.4.5 RMSProp

Метод оптимизации основанный на AdaGrad, использующий экспоненциально взвешенное среднее квадратов градиентов для адаптации скорости обучения.

1.4.6 Adam

Комбинированный метод, объединяющий идеи метода моментов и метода RMSProp. Adam использует адаптивные оценки момента первого и второго порядка для эффективной оптимизации [18].

Псевдоалгоритм функции представлен на рисунке 5

```

for  $t = 1$  to  $\dots$  do
  if maximize :
     $g_t \leftarrow -\nabla_{\theta} f_t(\theta_{t-1})$ 
  else
     $g_t \leftarrow \nabla_{\theta} f_t(\theta_{t-1})$ 
  if  $\lambda \neq 0$ 
     $g_t \leftarrow g_t + \lambda \theta_{t-1}$ 
   $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$ 
   $v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$ 
   $\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ 
   $\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ 
  if amsgrad
     $\widehat{v}_t^{max} \leftarrow \max(\widehat{v}_t^{max}, \widehat{v}_t)$ 
     $\theta_t \leftarrow \theta_{t-1} - \gamma \widehat{m}_t / (\sqrt{\widehat{v}_t^{max}} + \epsilon)$ 
  else
     $\theta_t \leftarrow \theta_{t-1} - \gamma \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ 
return  $\theta_t$ 

```

Рисунок 5 – Псевдоалгоритм функции оптимизации Adam

1.5 Сегментируемые данные

1.5.1 Спутниковые снимки

Наиболее распространенными источниками данных для семантической сегментации являются спутниковые снимки. Они предоставляют широкий спектр информации о поверхности Земли, включая различные типы растительности, здания, дороги, водные объекты и другие элементы ландшафта. Спутниковые данные имеют ряд преимуществ, которые делают их подходящими для семантической сегментации [19, 20]:

- масштабность и доступность. Спутниковые снимки доступны в различных масштабах и разрешениях, что позволяет выбрать оптимальный вариант в зависимости от требований решаемой задачи. Кроме того, они регулярно обновляются, обеспечивая актуальную информацию о состоянии поверхности Земли,

- разнообразие. Спутники могут снимать изображения в разных спектральных диапазонах, что позволяет получать информацию о различных характеристиках объектов, таких как цвет, текстура и отражательная

способность. Это делает спутниковые данные более информативными для семантической сегментации по сравнению с другими источниками данных,

- точность и надежность. Современные спутники оснащены высокоточными сенсорами, что обеспечивает высокую точность и надежность данных. Это важно для семантической сегментации, где требуется точная классификация объектов,

- широкий охват. Спутниковые снимки охватывают большие территории, что позволяет проводить анализ на региональном и на глобальном уровне. Это особенно полезно для мониторинга изменений в землепользовании, климатических условиях, а также и в других процессах, происходящих на больших площадях,

- возможность интеграции с другими данными. Спутниковые снимки могут быть легко интегрированы с другими типами данных, такими как наземные наблюдения, модели климата и прочие. Это позволяет получить более полную картину происходящего на поверхности Земли,

- автоматизация обработки. Существует множество алгоритмов и инструментов для автоматической обработки спутниковых данных. Это упрощает процесс семантической сегментации и снижает затраты на ручной труд.

В целом, выбор спутниковых снимков для семантической сегментации обусловлен их доступностью, разнообразием, точностью, надежностью, широким охватом и возможностью интеграции с другими данными. Эти преимущества делают спутниковые снимки предпочтительными для семантической сегментации.

1.5.2 Требования к данным

К набору спутниковых снимков предъявляются следующие требования:

- изображения формата: jpeg, jpg, png или tif,
- единых масштаб спутниковых снимков,
- максимальный размер изображений: 4000x4000 пикселей.

Требуемые для распознавания классы должны быть размечены на так

называемой маске, каждый класс своим цветом.

1.5.3 Набор данных

Требованиям, предъявленным в предыдущем пункте, соответствует набор данных – «DeepGlobe Land Cover Classification Dataset» [21].

Данный набор содержит:

- 803 пары изображений (спутниковый снимок в формате jpg и маска в формате png), предназначенных для обучения,
- 86 пар изображений, предназначенных для тестирования,
- 171 изображение для визуальной проверки качества работы модели сегментации изображения.

Пример спутникового снимка представлен на рисунке 6.



Рисунок 6 – Спутниковый снимок

Пример маски представлен на рисунке 7.

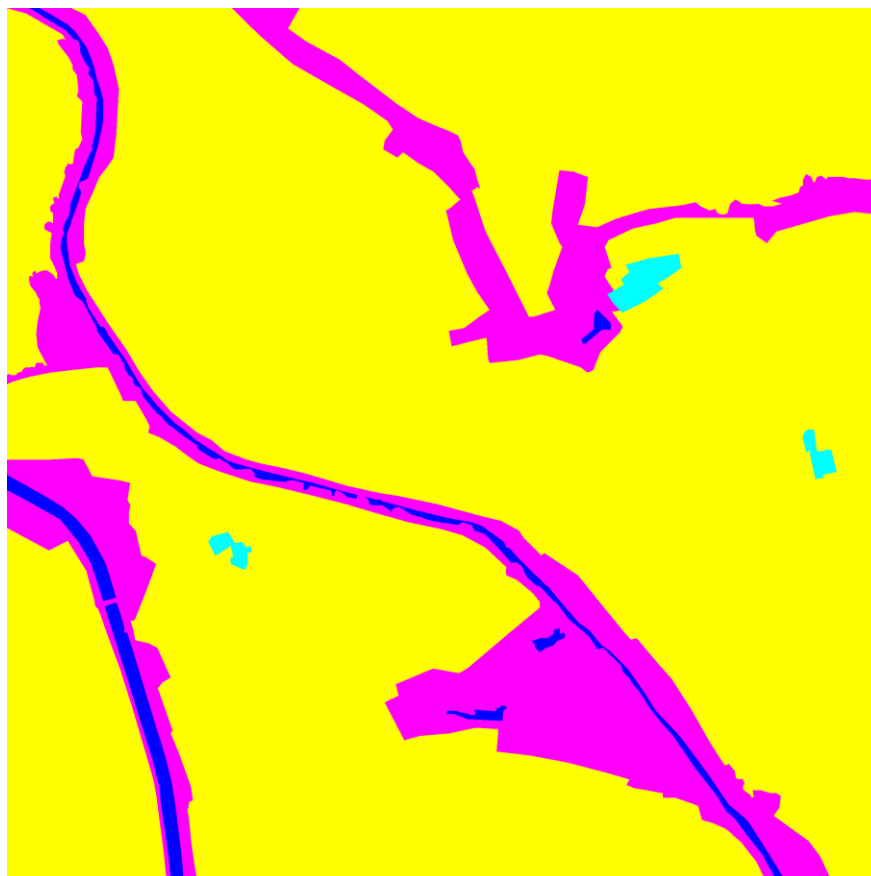


Рисунок 7 – Маска классов

В наборе представлены следующие классы:

- городская земля RGB(0,255,255). Созданные человеком, застроенные территории,
- сельское хозяйство RGB(255 255,0). Фермерские хозяйства, а также запланированные плантации, пахотные земли, сады, виноградники, питомники и зоны декоративного садоводства,
- пастбищные угодья RGB(255,0,255). Любые нелесные, несельскохозяйственные, зеленые земли, трава,
- лесные угодья RGB(0,255,0). Любые земли с кронами деревьев,
- вода RGB(0,0,255). Реки, океаны, озера, водно-болотные угодья, пруды,
- бесплодная земля RGB(255,255,255). Горы, суша, скалы, пустынные места, пляж, без растительности,
- неизвестно RGB(0,0,0). Облака и другое.

Также был выбран набор «Fine land-cover classification set», который состоит из 10 снимков формата tif и размера 7200x6800 пикселей, которые в последствии будут разбиты на изображения меньшего разрешения для увеличения выборки [22]. Пример спутникового снимка приведен на рисунке 8.



Рисунок 8 – Спутниковый снимок

Пример маски представлен на рисунке 9.

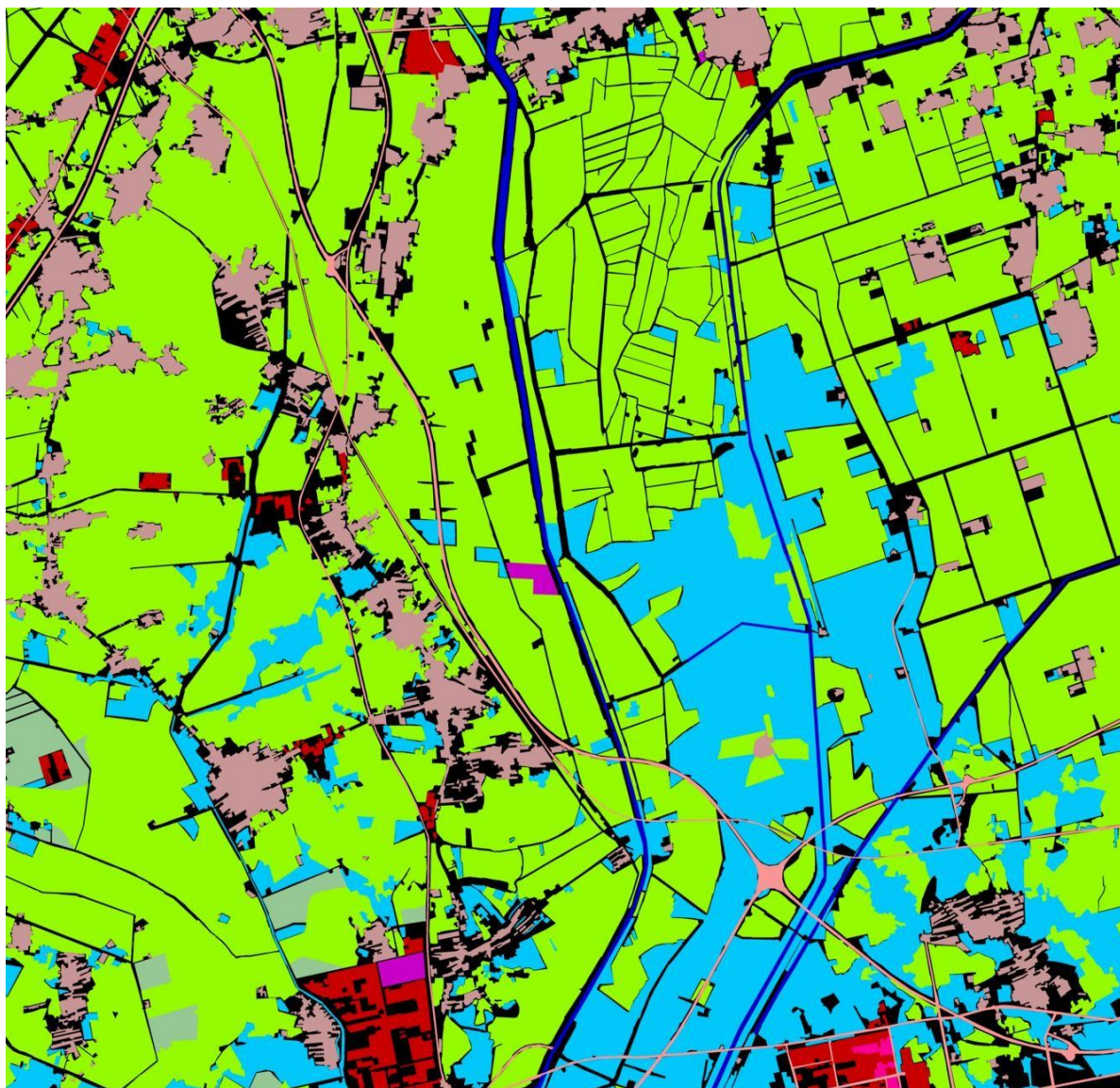


Рисунок 9 – Маска классов В наборе представлены следующие классы:

- неизвестно RGB(0,0,0). Облака и другое,
- производственная площадь RGB(200,0,0). Заводы, предприятия,
- городская жилая площадь RGB(250,0,150). Многоэтажные здания,
- сельская жилая площадь RGB(200,150,150). Дома, низко этажные здания,
- дорожное покрытие RGB(250,150,150). Дороги, трассы, автомагистрали,
- рисовые поля RGB(0,200,0),
- орошаемые земли RGB(150,250,0),

- сухая пахотная земля RGB(150, 200, 150),
- садовая территория RGB(200, 0, 200),
- лесные угодья RGB(150,0,250). Любые земли с кронами деревьев,
- кустарная местность RGB(150, 150, 250),
- естественная поляна RGB(250, 200, 0),
- искусственная поляна RGB(200, 200, 0),
- реки RGB(0, 0, 200),
- озера RGB(0, 150, 200),
- пруды RGB(0, 200, 250).

1.6 Выбор инструментов

На данный момент самым популярным языком для создания и реализации любого вида нейросетей является Python. Огромное множество библиотек и API, написанных на данном языке, позволяют реализовать и игрового агента, управляемого нейросетью. Чтобы разобраться какой же инструмент использовать, рассмотрим основные характеристики самых популярных из них.

1.6.1 TensorFlow

TensorFlow – это фреймворк глубокого обучения с открытым исходным кодом, разработанный командой «Google Brain». Он один из наиболее широко используемых фреймворков для построения и обучения нейронных сетей. TensorFlow предоставляет обширную экосистему инструментов, библиотек и ресурсов для машинного обучения и разработки искусственного интеллекта [23, 24].

Вот некоторые ключевые особенности и понятия TensorFlow:

- граф вычислений. Работа с TensorFlow основана на создании и выполнении графа вычислений, который определяет, как будут проводиться вычисления. В отличие от классического императивного программирования, где код выполняется последовательно, в TensorFlow императивный подход используется лишь для вспомогательных целей. Основная задача

TensorFlow – создание структуры, определяющей порядок вычислений. В TensorFlow программа естественным образом разделяется на две части: построение графа вычислений и выполнение вычислений в данной структуре,

- тензор. Тензор представляет собой многомерный массив. Тензор может быть отдельными числами, векторами признаков, изображениями, а также батчами объектов или массивами изображений,

- сессии. Для выполнения вычислительных графов используются сессии. Контекст выполнения графа, включая необходимые ресурсы, вспомогательные классы и адресные пространства содержит объект сессии. В TensorFlow существуют два типа сессий: интерактивные и обычные сессии. Интерактивная сессия удобнее для работы в консоли – главное различие между ними. Интерактивная сессия является сессией по умолчанию,

- автоматическая дифференциация. TensorFlow предоставляет возможности автоматического дифференцирования. Это очень важно для обучения нейронных сетей с использованием алгоритмов оптимизации на основе градиента. Автоматическая дифференциация позволяет автоматически вычислять градиенты сложных функций, что облегчает реализацию и обучение моделей глубокого обучения,

- распределенные вычисления и ускорение обучения на GPU. TensorFlow поддерживает распределенные вычисления, позволяя выполнять процессы обучения и вывода на нескольких устройствах или машинах. Эта возможность необходима для использования возможностей кластеров или ускорителей GPU и обучения крупномасштабных моделей.

TensorFlow имеет большое и активное сообщество разработчиков, исследователей и энтузиастов, в следствии чего найти какую-либо информацию по данной библиотеке не составляет труда.

TensorFlow совместим с несколькими языками программирования, при этом наиболее часто используемым языком является Python для разработки

TensorFlow благодаря обширной экосистеме машинного обучения и своей простоте.

Помимо основной библиотеки TensorFlow, существуют специализированные версии, такие как TensorFlow Extended для производственных конвейеров, TensorFlow Probability для вероятностного моделирования и TensorFlow Lite для мобильных и встроенных устройств.

1.6.2 PyTorch

PyTorch – это фреймворк глубокого обучения с открытым исходным кодом, разработанный командой «Facebook AI Research». Он завоевал значительную популярность благодаря динамичному вычислительному графу и удобному интерфейсу. PyTorch обеспечивает удобство использования и простоту, при этом обеспечивает контроль над моделями нейронных сетей и гибкость [25].

Вот некоторые ключевые особенности и концепции, связанные с PyTorch:

- динамический вычислительный граф. PyTorch строит граф вычислений подобно TensorFlow, однако в данном фреймворке динамический граф вычислений строится и оценивается на лету во время выполнения программы. Динамическая структура позволяет более гибко отлаживать и разрабатывать модели,

- ускорение обучения на GPU. PyTorch легко интегрируется с графическими процессорами, обеспечивая эффективные вычисления на GPU для ускоренного обучения и выводов. Фреймворк обеспечивает поддержку CUDA, что позволяет пользователям использовать вычислительную мощность графических процессоров NVIDIA для ускорения своих моделей глубокого обучения,

- автоматическая дифференциация. PyTorch подобно TensorFlow обеспечивает автоматическое дифференцирование, что позволяет пользователям вычислять градиенты функций относительно переменных. В PyTorch модуль autograd позволяет отслеживать операции, выполняемые над

тензорами, и автоматически вычисляет и распространяет градиенты через вычислительный граф,

– TorchVision и TorchText. PyTorch включает в себя специфические библиотеки, такие как TorchVision и TorchText. TorchVision предоставляет популярные наборы данных, архитектуры моделей и предварительно обученные модели для задач компьютерного зрения. TorchText предлагает утилиты для обработки текста, включая создание словаря и подготовку набора данных,

– TorchScript. Инструмент, который позволяет пользователям сериализовать модели PyTorch и использовать автономно. Эта функциональность полезна для развертывания моделей и сред, где модель должна быть выполнена эффективно без использования всего фреймворка PyTorch.

1.6.3 Flask

Flask – это микрофреймворк для разработки веб-приложений на языке Python. Он предоставляет разработчикам инструменты и функции, необходимые для создания веб-сервисов, но при этом оставляет большую часть работы по проектированию и реализации приложения на усмотрение разработчика.

Flask позволяет быстро создавать и запускать простые веб-сервисы, а также легко расширять их функциональность с помощью дополнительных библиотек и модулей. Это делает его популярным выбором среди разработчиков, которые хотят быстро создать прототип или небольшой проект [26].

Основные особенности Flask:

– гибкость. Фреймворк позволяет разработчикам выбирать библиотеки и инструменты, которые лучше всего подходят для их проекта. Это дает большую гибкость при разработке,

- простота. Фреймворк имеет минималистичный дизайн и не включает в себя множество функций, что делает его простым в изучении и использовании,
- расширяемость. Фреймворк легко расширяется с помощью сторонних библиотек и модулей, что позволяет добавлять новые функции и возможности в приложение.

Фреймворк часто используется в сочетании с другими библиотеками и инструментами, такими как SQLAlchemy для работы с базами данных, Jinja2 для шаблонизации и т.д. Это позволяет разработчикам создавать сложные и функциональные веб-приложения.

1.7 Выбор исследуемых методов

1.7.1 Модель ResNet101

Архитектура ResNet – это архитектура глубоких нейронных сетей, разработанная для решения проблемы деградации производительности при увеличении глубины сети [2].

Основные принципы архитектуры ResNet:

- использование остаточных блоков вместо традиционных слоев. Остаточные блоки позволяют пропускать входные данные через сеть без изменения масштаба и добавлять их к выходным данным. Это позволяет избежать проблемы деградации производительности,
- применение пакетной нормализации после каждого слоя. Пакетная нормализация помогает стабилизировать процесс обучения и ускорить сходимость,
- использование функций активации ReLU после каждого остаточного блока. Функция ReLU является простой и эффективной функцией активации, которая помогает предотвратить насыщение нейронов.

Остаточный блок состоит из следующих частей:

- вход. Входные данные x ,

- пропускные соединения. Входные данные пропускаются через сеть без изменений,
- слой свертки. Слой свертки выполняет преобразование входных данных,
- функция активации. Функция активации применяется к выходу слоя свертки,
- добавляются выходные данные слоя свертки и входные данные, пропущенные через сеть,
- выход. Результат добавления входных и выходных данных. Пример структуры остаточного блока на языке Python:

```
x = Input(shape=(input_shape))
x = Conv2D(filters=32, kernel_size=(3, 3), padding='same')(x)
x = BatchNormalization()(x)
x = Activation('relu')(x)
shortcut = Conv2D(filters=32, kernel_size=(1, 1), strides=(1, 1))(x)
x = Add()([x, shortcut])
x = Activation('relu')(x)
```

В данном примере остаточный блок состоит из двух слоев свертки, функции активации и пакетной нормализации.

Данная модель решает проблему деградации, вводя структуру глубокого остаточного обучения. Вместо того, чтобы предполагать, что каждые несколько сложенных слоев напрямую соответствуют желаемому базовому отображению, модель явно позволяет этим слоям соответствовать остаточному отображению. Формально, обозначая желаемое базовое отображение как $H(x)$, модель позволяет сложенным нелинейным слоям соответствовать другому отображению $F(x)$ формула 3.

$$F(x) = H(x) - x, \quad (3)$$

где x – входные данные слоя.

Исходное отображение преобразуется в $F(x) + x$. Предполагается, что легче оптимизировать остаточное отображение, чем оптимизировать исходное, неиспользуемое отображение. В крайнем случае, если бы сопоставление идентичности было оптимальным, было бы проще свести остаток к нулю, чем подогнать отображение идентичности с помощью стека нелинейных слоев. Формулировку $F(x) + x$ можно реализовать с помощью нейронных сетей прямого распространения с «короткими соединениями». Короткие соединения – это соединения, пропускающие один или несколько слоев. В данном случае короткие соединения просто выполняют сопоставление идентификаторов, а их выходные данные добавляются к выходам составных слоев. Упрощенные соединения идентификаторов не добавляют ни дополнительных параметров, ни сложности вычислений. Остаточный блок представлен на рисунке 10.

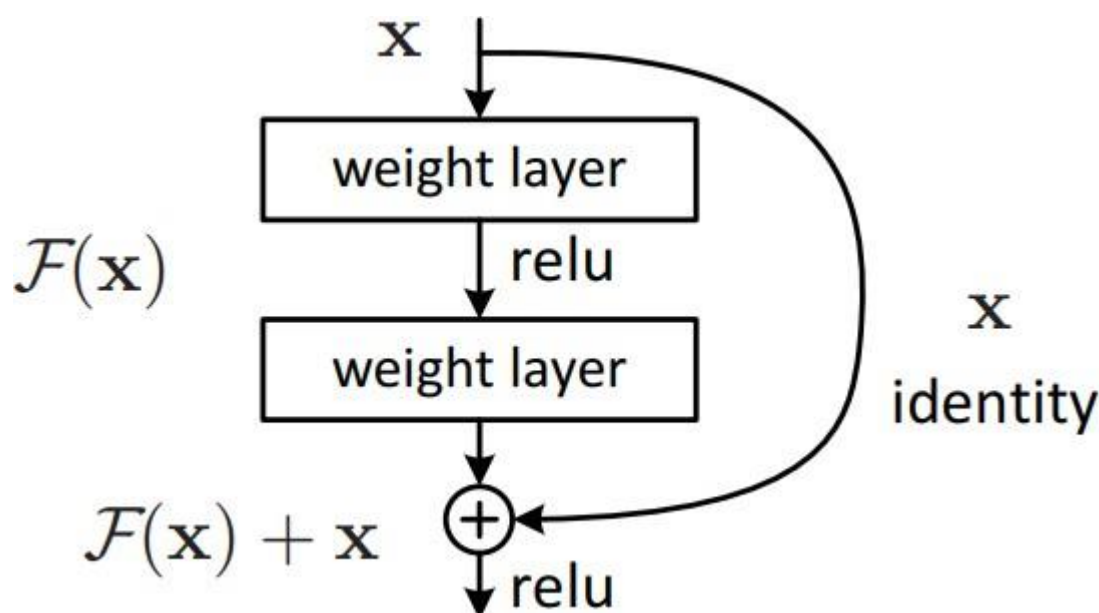


Рисунок 10 – Остаточный блок

1.7.2 Модель DeepLabV3Plus

DeepLabV3Plus – это сверточная нейронная сеть, разработанная для семантической сегментации изображений. DeepLabV3Plus является улучшенной версией DeepLabV3 и использует несколько ключевых технологий для повышения точности и эффективности сегментации.

Архитектура DeepLabV3Plus состоит из нескольких основных блоков:

- входной блок. Принимает изображение в качестве входных данных и выполняет предварительную обработку, такую как масштабирование и нормализация,
- сверточный блок. Содержит несколько сверточных и Atrous сверточных слоев, которые извлекают признаки из изображения,
- асимметричный сверточный блок. Применяет асимметричные сверточные слои для улучшения деталей сегментированных объектов,
- блок объединения масштабов. Объединяет карты признаков, полученные на разных масштабах, для создания многомасштабного представления изображения,
- полностью связанный блок. Использует полностью связанные слои для извлечения высокоуровневых признаков и предсказания классов пикселей,
- функция потерь и оптимизация. Вычисляет функцию потерь на основе предсказанных и истинных меток пикселей и оптимизирует параметры модели с помощью алгоритма оптимизации, такого как стохастический градиентный спуск.

DeepLabV3Plus также включает в себя несколько дополнительных блоков и функций, которые помогают улучшить производительность модели:

- блок нормализации. Используется для нормализации карт признаков перед их объединением в многомасштабном представлении. Это помогает предотвратить переобучение модели и улучшить ее обобщающую способность,
- функция потерь с учетом границ. Учитывает гладкость границ между классами при вычислении функции потерь. Это позволяет модели лучше определять границы объектов и избегать ошибок сегментации,
- оптимизация параметров. Использует алгоритм оптимизации, такой как стохастический градиентный спуск, для настройки параметров модели на основе функции потерь,

– многозадачное обучение. Модель может быть обучена одновременно выполнять несколько задач, таких как семантическая сегментация и классификация объектов. Это может помочь улучшить общую производительность модели,

– регуляризация. Используются методы регуляризации, такие как L2-регуляризация, чтобы предотвратить переобучение и улучшить обобщающую способность модели.

Полная структура модели DeepLabV3Plus представлена в приложении Б.

Исходя из результатов сравнительного анализа, а также выбора DeepLab в качестве нейронной сети типа CNN, оптимальной моделью машинного обучения для системы является DeepLabV3Plus

1.8 Метрики для оценки качества сегментации

Качество сегментации изображений можно оценить с помощью различных метрик [27]:

– пересечение над объединением (IoU). Это метрика, которая измеряет степень совпадения между предсказанными и истинными масками сегментации. Она рассчитывается как отношение площади пересечения предсказания и истины к площади объединения этих двух областей. IoU является одной из самых популярных метрик для задач сегментации и может быть адаптирована для разных типов данных,

– F-мера. Объединяет точность и полноту в одну метрику. F-мера учитывает как количество верно классифицированных объектов, так и количество пропущенных объектов. Это делает ее полезной для оценки моделей, которые могут иметь высокую точность, но низкую полноту или наоборот,

– средняя точность. Эта метрика вычисляет долю правильно классифицированных пикселей относительно общего числа пикселей в изображении. Метрика может быть полезна для сравнения моделей сегментации на разных наборах данных или для анализа производительности модели на отдельных классах объектов,

– коэффициент Дайса. Похож на IoU, но использует другую формулу для

расчета. Коэффициент Дайса также измеряет сходство между предсказанием и истиной, но он более устойчив к несбалансированным данным,

- потеря кросс-энтропии. Используется в качестве функции потерь при обучении моделей сегментации. Кросс-энтропия измеряет разницу между истинным распределением меток и предсказанным распределением. Меньшее значение потери означает более точное предсказание,

- полнота. Показывает, какая доля истинных положительных результатов была правильно предсказана моделью. Для задач, где необходимо минимизировать пропуски важных объектов важна полнота,

- точность. Измеряет долю, верно предсказанных положительных результатов среди всех предсказанных положительных результатов. Точность важна для задач, где важно минимизировать ложные срабатывания,

- площадь под кривой ошибок. Представляет собой интеграл под графиком зависимости количества, верно определенных положительных примеров от количества неверно определенных отрицательных примеров. Метрика часто используется для оценки обобщающей способности модели.

2 Требования к программе

2.1 Функциональные требования

В системе анализа и распознавания изображений должны быть реализованы следующие функции:

- загрузка фотограмметрических данных (входные данные для работы с системой),
- управление процессом обработки фотограмметрических данных,
- создание и настройка конфигурации обработки фотограмметрических данных,
- отображение результатов обработки фотограмметрических данных,
- выгрузка результатов обработки фотограмметрических данных в формате *.geojson (выходные данные),
- просмотр подробной информации о выбранном процессе обработки фотограмметрических данных,
- импорт и удаление пользовательских векторных данных (слоев) в формате *.geojson, *.shp, *.zip в процесс обработки фотограмметрических данных,
- указание системы координат для каждого загружаемого векторного слоя в формате *.geojson, *.shp, *.zip в процесс обработки фотограмметрических данных,
- просмотр импортированных пользовательских векторных данных в процесс обработки фотограмметрических данных,
- просмотр этапов конвейера обработки фотограмметрических данных в процессе обработки фотограмметрических данных,
- управление фотограмметрическими данными, используемые при запуске процесса обработки фотограмметрических данных,
- просмотр подробной информации об источнике данных,
- просмотр подробной информации о конфигурации, используемой при запуске процесса обработки фотограмметрических данных,

- просмотр атрибутивных данных выявленных объектов, входящих в результат обработки фотограмметрических данных,
- управление дополнительными атрибутивными данными выявленных объектов,
- измерение расстояния между объектами в выбранном векторном слое,
- скачивание изображений рабочей области интерфейса, в которой отображены растровые и векторные данные, для последующей печати,

2.1.1 Требования к подсистеме хранения данных

Подсистема хранения данных должна обеспечивать выполнение следующих функций:

- управление доступом на уровне базы данных в целом, таблиц базы данных, полей таблиц базы данных,
- сохранение данных, поддержание их целостности,
- масштабируемость состава и структуры таблиц по мере развития базы данных в процессе эксплуатации.

В состав подсистемы хранения данных должны входить следующие компоненты:

- оперативная база данных,
- файловое хранилище (ftp-сервер),
- брокер сообщений (RabbitMQ),
- модуль управления данными,
- сервис доступа к файлам,
- процессная база данных,
- сервис модели данных,
- сервис ETL.

2.1.2 Требования к подсистеме визуализации пространственных данных

В состав подсистемы ввода и визуализации пространственных данных должны входить следующие модули:

- ГИС (Геоинформационная система визуализации результатов

интеллектуальной обработки данных),

- интеллектуальная обработка (управление процессами обработки данных),
- пространственные данные (управление источниками данных),
- проектирование процессов (проектирование процессов обработки данных).

2.1.3 Требования к модулю «ГИС»

В модуле «ГИС» должна осуществляться визуализация результатов интеллектуальной обработки данных. При нажатии на объект должно открываться модально окно карточки объекта, в котором должна отображаться информация об интеллектуальной обработке объекта, а также должна быть реализована возможность скачивания объекта в формате GeoJSON и добавления дополнительных данных по объекту

Модуль «ГИС» должен обеспечивать возможность визуализации пространственных данных на картографической основе «OpenStreetMap».

Модуль «ГИС» должен обеспечивать возможность визуализации пространственных данных по административному делению: федеральные округа, субъекты РФ, районы. Выбор административного деления должен осуществляться с помощью элемента «чек-бокс».

Модуль «ГИС» должен обеспечивать возможность скачивания и печати изображения рабочей области карты, измерения расстояния области карты, масштабирования и переход в полноэкранный режим отображения карты.

2.1.4 Требования к модулю «Интеллектуальная обработка»

В модуле «Интеллектуальная обработка данных» должно осуществляться управление процессами обработки данных, в частности должны быть реализованы следующие функции:

- скачивание данных объекта в формате GeoJSON,
- переход в модуль «ГИС» для просмотра объекта интеллектуальной обработки на картографической основе,
- импорт и удаление пользовательских векторных данных (доступные

форматы загрузки: GeoJSON, *.shp, *.zip),

- визуализация объекта на мини-карте с возможностью масштабирования карты и перехода в полноэкранный режим просмотра объекта,
- визуализация прогресса процессов обработки данных по каждому из этапов.

При создании нового процесса обработки данных должна открываться соответствующая экранная форма, которая включает в себя следующие функции:

- ввод названия процесса интеллектуальной обработки данных,
- выбор конфигурации интеллектуальной обработки данных с помощью поля типа «справочник»,
- выбор источника данных с помощью поля типа «справочник»,
- отображение мини-карты с возможностью масштабирования и перехода в полноэкранный режим просмотра,
- выбор зоны обработки на мини-карте,
- запуск процесса интеллектуальной обработки данных,
- указание дополнительных данных в формате запроса.

После запуска процесса созданный процесс должен отображаться в перечне элементов.

2.1.5 Требования к модулю «Пространственные данные»

В модуле «Пространственные данные» должно осуществляться управление источниками данных системы.

Должна быть реализована возможность добавления источника данных с помощью соответствующей кнопки. При нажатии на кнопку «Загрузить источник» должна открываться экранная форма «Создание источника данных». При создании источника данных должны использоваться следующие допустимые данные: расширения файлов (*.tiff, *.tif), системы координат (EPSG:4326, МСК). После загрузки источник данных должен отображаться в перечне источников.

2.1.6 Требования к модулю «Проектирование процессов (проектирование

процессов обработки данных)»

Модуль проектирования процессов должен состоять из веб-приложения для управления бизнес-процессами (Camunda Cockpit). При открытии модуля должна отображаться экранная форма авторизации, включающая в себя поле для ввода логина и пароля, а также кнопки «Log In». В случае неверной авторизации должно отображаться соответствующее сообщение об ошибке.

После авторизации в модуле должен отображаться интерфейс Camunda Cockpit, в котором будет осуществляться работа по проектированию, запуску, мониторингу и анализу процессов обработки данных.

2.2 Не функциональные требования

2.2.1 Требования к составу, структуре и способам организации данных

В качестве входных данных для работы с системой используются web-карты спецификации TMS/WMS/WMTS, аэрофотоснимки и спутниковые снимки формата *.geotiff со следующими характеристиками:

- web-карты,
- спецификация: TMS/WMS/WMTS,
- аэрофотоснимки и спутниковые снимки формата *.geotiff,
- система координат: WGS-84 (EPSG:4326) или МСК (Местная система координат),
- каналы: 3 или 4 канала типа Byte (RGB или RGBA), спектральные данные,
- качество изображения: разрешающая способность от 4 см до 10 м на 1 px,
- аэрофотоснимки с разрешающей способностью 4-10 см на 1 px,
- аэрофото и спутниковые снимки с разрешающей способностью 10-30 см на 1 px,
- спутниковые снимки с разрешающей способностью 30-70 см на 1 px,
- спутниковые снимки с разрешающей способностью 10 м на 1 px,
- геосервисы способные возвращать изображения с разрешением 256 px, 512 px или 1024 px.

В качестве выходных данных в программе «Система мониторинга,

семантического и пространственного анализа на основе технологии искусственного интеллекта и алгоритмов машинного обучения» формируются:

- результаты обработки фотограмметрических данных в формате *.geojson (информация о детектированных объектах), и отображение их на геопортале Системы,

- цифровые реестры (аналитические отчеты) в формате *.xlsx, *.geojson, и отображение их на тематических дашборд-панелях в информационно-аналитической подсистеме Системы,

- характеристики выходных данных, формируемые программой,

- результаты обработки фотограмметрических данных (результаты детектирования), полностью или по классам (типам) детектированных объектов,

- файлы в формате *.geojson, *.xlsx.

2.2.2 Требования к программному обеспечению

Специализированное программное обеспечение системы должно состоять из следующих элементов: СУБД, серверы приложений, веб-серверы.

Графический пользовательский интерфейс системы должен быть реализован как веб-приложение, используемое пользователями с помощью веб-браузеров.

Корректная работа веб-приложений системы должна обеспечиваться для официально поддерживаемых производителями последних версий браузеров Chromium-gost, Mozilla Firefox и Яндекс Браузер.

2.2.3 Технические требования

Таблица 1 – Технические требования

Наименование docker-контейнера	Соотношение docker-контейнера к составляющей Системы
PostgreSQL 12	Реляционная СУБД PostgreSQL 12

Продолжение таблицы 1

Наименование docker-контейнера	Соотношение docker-контейнера к составляющей Системы
Mongo	NoSQL СУБД MongoDB
Nginx	Компонент «Web-сервер Nginx»
Portainer	Графическая оболочка управления docker-контейнерами
Core	Компоненты ядра Системы
Core-versioning	Сервис обеспечения автоматического сохранения версий сущностей системы при их изменениях
Config	Сервис обеспечения хранения и оперативного доступа к конфигурационным параметрам docker-контейнеров Системы
Discovery	Сервис оркестровки
R	Сервис агрегации и маршрутизации API
F	Сервис хранения основных файлов системы
Rabbit-MQ	Утилита «Брокер сообщений (событий)»
Mongo	NoSQL СУБД MongoDB
Att Data Asignation	Микросервис присвоения атрибутивных данных
Environment Setup	Микросервис настройки рабочего окружения процесса обработки
File Uploading	Микросервис выгрузки результатов во внешнюю БД

Продолжение таблицы 1

Наименование docker-контейнера	Соотношение docker-контейнера к составляющей Системы
Product Preparing	Микросервис загрузки тайлов из продактов
Tile Downloader	Микросервис загрузки тайлов
Gdaltransform	Сервис преобразований систем координат в WGS 84
Aip-parser	Сервис обеспечения парсинга файлов табличного вида
Masks Blending	Микросервис смешивания масок
Masks Binarization	Микросервис бинаризации масок
Masks Splitting	Микросервис разделения масок
Segmentation CPU	Микросервис сегментации (Mask R-CNN, CPU-версия)
Segmentation GPU	Микросервис сегментации (Mask R-CNN, GPU-версия)
Segmentation Multiband CPU	Микросервис сегментации для работы с многоканальными изображениями (Mask R-CNN, CPU-версия)
Segmentation Multiband GPU	Микросервис сегментации для работы с многоканальными изображениями (Mask R-CNN, GPU-версия)
Segmentation Deeplab CPU	Микросервис сегментации (Deeplab- V3Plus, CPU-версия)
Segmentation Deeplab GPU	Микросервис сегментации (Deeplab- V3Plus, GPU-версия)

Продолжение таблицы 1

Наименование docker-контейнера	Соотношение docker-контейнера к составляющей Системы
Segmentation- Deeplab Multiband GPU	Микросервис сегментации для работы с многоканальными изображениями (DeerLab- V3Plus, GPU-версия)
Segmentation Unet CPU	Микросервис сегментации и классификации (U-Net, CPU-версия)
Segmentation Unet GPU	Микросервис сегментации и классификации (U-Net, GPU-версия)
Segmentators Manager	Сервис управления микросервисами сегментации
IC Displacing	Микросервис генерации тайлов, смещенных по сторонам света
IC Filtering	Микросервис межкардинальной фильтрации сегментированных объектов
IC Redisplacing	Микросервис обратного смещения объектов, отфильтрованных алгоритмом межкардинальной фильтрации
QGIS Executor	Микросервис выполнения QGIS-алгоритмов
Orthogonalization	Микросервис расчета ортогонализованных полигонов
Vectorization	Микросервис векторизации тайловых масок
Pentaho Carte	Сервис запуска и мониторинга преобразований и заданий Pentaho

Продолжение таблицы 1

Наименование docker-контейнера	Соотношение docker-контейнера к составляющей Системы
Pentaho Server	Сервис Pentaho Data Integration
Analytical Service	Микросервис подготовки данных для аналитики
PostgreSQL 12 with PostGIS 3	Реляционная СУБД PostgreSQL 12 с расширением PostGIS 3
Geoserver 2.17	Сервер распространения геоданных Geoserver
Mapproxy	Сервер проксирования внешних геоданных Mapproxy
Product Preparer	Микросервис подготовки продуктов
Redis	СУБД Redis
RabbitMQ	Брокер сообщения RabbitMQ
Result Uploading	Микросервис выгрузки результатов во внутреннюю БД
PKK Data Provider	Сервис доступа к данным публичной кадастровой карты

2.2.4 Аппаратные требования

Рекомендуемые требования к аппаратной части, на которой планируется развёртывание и функционирование системы:

- процессор: 16x CPU 3.5 GHz,
- оперативная память: 64 Gb,
- видео карта: NVIDIA GeForce RTX 3080, 10 Gb,
- SSD накопитель: 1 Tb SSD,
- жесткий диск: 18 Tb HDD,
- сетевой адаптер: Gigabit Ethernet 1000 Mbit.

3 Обучение модели DeepLabV3Plus

Была разработана и обучена модель DeepLabV3Plus, в качестве функции потерь использована DiceLoss, в качестве функции оптимизации Adam.

Было пройдено 7 эпох обучения, время обучения составило 2 часа 40 минут. Результаты обучения представлены на рисунках 11, 12.

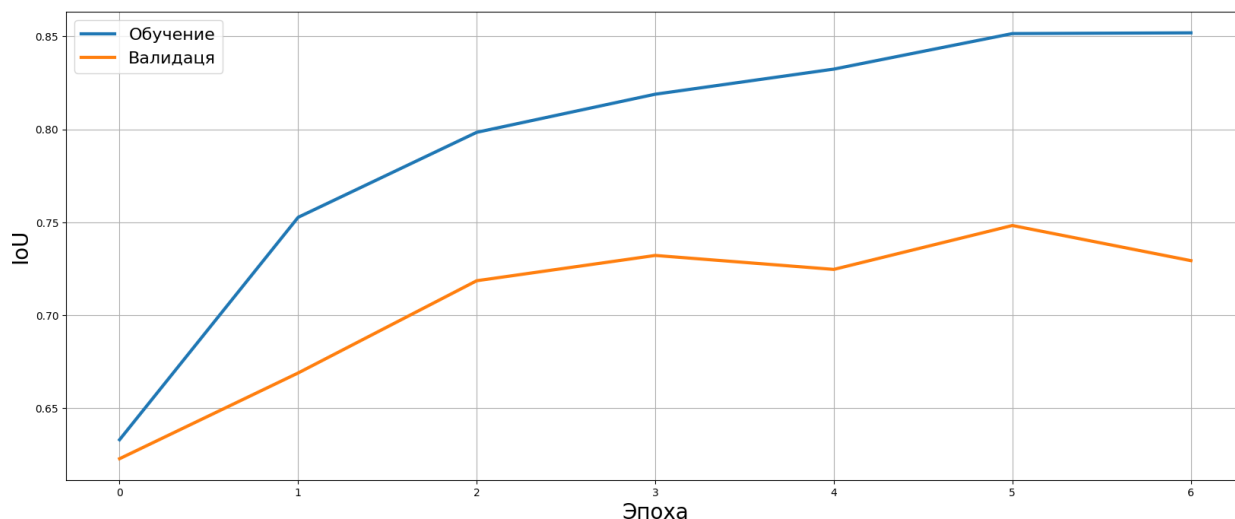


Рисунок 11 – Результат обучения модели

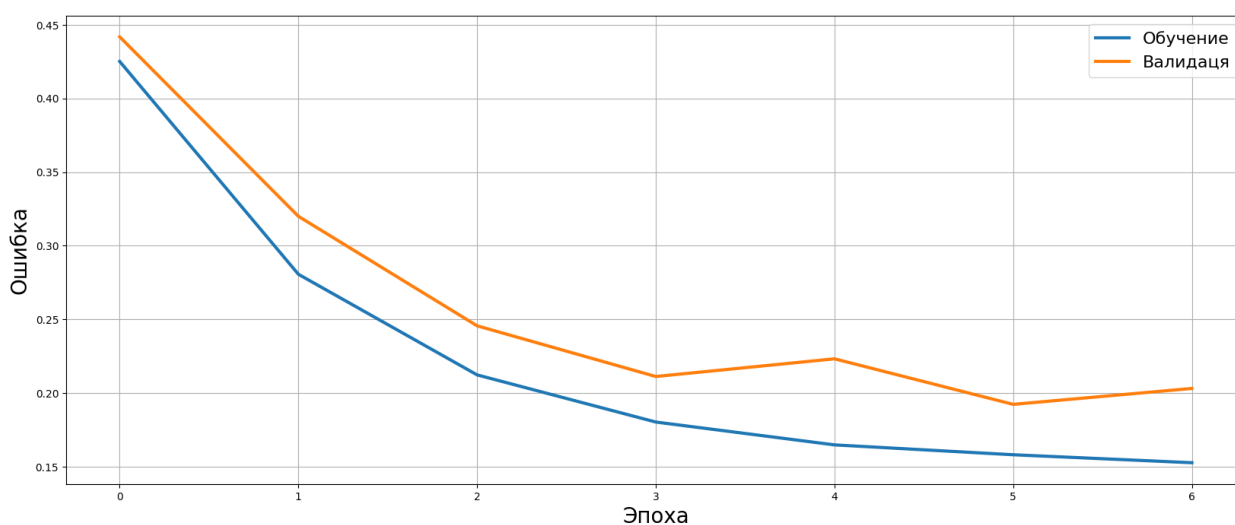


Рисунок 12 – Результат обучения модели

После настройки структуры модели DeepLabV3Plus, которая представлена в приложении Б, можно приступить к ее обучению.

Параметры обучения, такие как базовая модель, количество эпох, скорость обучения и т. д., настраиваются с помощью класса TrainingConfig

```
@dataclass(frozen=True)
class TrainingConfig:
```

```

MODEL: str = "resnet50_v2_imagenet"
EPOCHS: int = 35
LEARNING_RATE: float = 1e-4
CKPT_DIR: str=os.path.join("checkpoints_"+"_".join(MODEL.split
("_")[:2])),
    "deeplabv3_plus_"+"_".join(MODEL.split("_")[:2])+".h5")
LOGS_DIR:      str = "logs_"+"_".join(MODEL.split("_")[:2])

```

Далее требуется подготовить набор данных для семантической сегментации, на которых будет происходить обучение.

Для работы с данными мы будем использовать TensorFlow tf.data API - мощный инструмент, который предоставляет гибкие возможности для создания эффективных конвейеров обработки данных. Этот API позволяет:

- работать с большими наборами данных,
- выполнять сложные преобразования данных,
- поддерживать различные форматы данных,
- создавать модульный код с возможностью повторного использования.

Процесс подготовки данных начинается с определения путей к файлам изображений и соответствующим маскам. Важным этапом является случайное перемешивание (shuffle) этих путей, что помогает избежать систематической ошибки в обучении модели и обеспечивает более равномерное представление данных при обучении.

```

data_images = glob.glob(os.path.join(DATASET_DIR, "Images",
"*.*jpg"))
data_masks = glob.glob(os.path.join(DATASET_DIR, "Masks",
"*.*jpg"))
# Shuffle the data paths before data preparation.
zipped_data = list(zip(data_images, data_masks))
random.shuffle(zipped_data)
data_images, data_masks = zip(*zipped_data)
data_images = list(data_images)

```

```
data_masks = list(data_masks)
```

Для сокращения пути к файлам изображений и масок, требуется объединить их в набор данных TensorFlow, используя

```
tf.data.Dataset.from_tensor_slices
```

```
org_data = tf.data.Dataset.from_tensor_slices((data_images, data_masks))
```

После выполнения вышеописанных операций можно приступить к обучению модели.

В качестве порогового значения для пикселей масок установлен диапазон от 200 до 255.

```
{"images": image, "segmentation_masks": mask}
```

Загрузим изображения и маски из набора данных, созданного с использованием путей к файлам. Во время загрузки необходимо поддерживать разрешение изображения (256, 256) для изменения размера изображений и масок. Последним шагом будет создание сущности, в которой входные изображения сопоставляются с масками

```
def read_image_mask(image_path, mask=False, size = data_config.IMAGE_SIZE):  
    image = tf.io.read_file(image_path)  
    if mask:  
        image = tf.io.decode_image(image, channels=1)  
        image.set_shape([None, None, 1])  
        image = tf.image.resize(images=image, size=size, method = "bicubic")  
        image_mask = tf.zeros_like(image)  
        cond = image >= 200  
        updates = tf.ones_like(image[cond])  
        image_mask = tf.tensor_scatter_nd_update(image_mask, tf.where(cond), updates)  
        image = tf.cast(image_mask, tf.uint8)  
    else:  
        image = tf.io.decode_image(image, channels=3)  
        image.set_shape([None, None, 3])
```

```

        image = tf.image.resize(images=image, size=size,
method = "bicubic")
        image = tf.cast(tf.clip_by_value(image, 0., 255.),
tf.float32)
    return image

```

Далее создадим обучающие и проверочные наборы данных

```

train_ds = train_data.map(load_data,
num_parallel_calls=tf.data.AUTOTUNE)
valid_ds = valid_data.map(load_data,
num_parallel_calls=tf.data.AUTOTUNE)
def unpack_inputs(inputs):
    images = inputs["images"]
    segmentation_masks = inputs["segmentation_masks"]
    return images, segmentation_masks

```

Визуализация процесса обучения модели DeepLabV3Plus представлена на рисунке 13

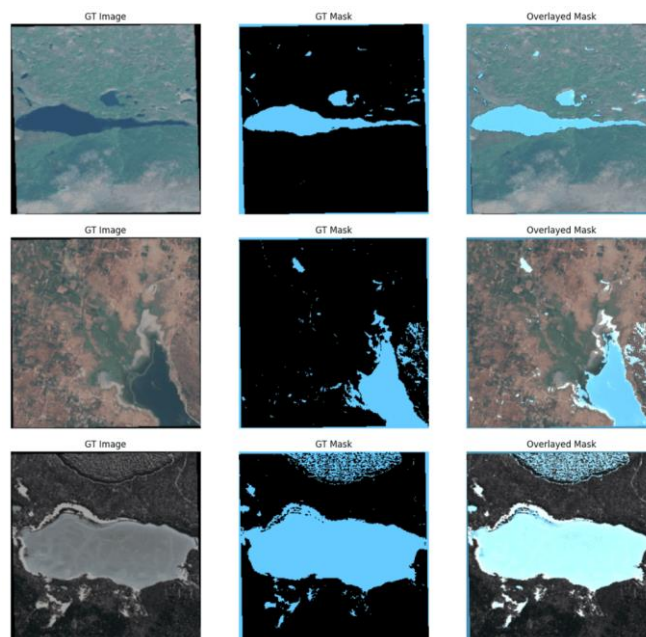


Рисунок 13 – Процесс обучения модели DeepLabV3Plus

Тестирование системы

Тест на допустимые значения представлен на рисунке 14.

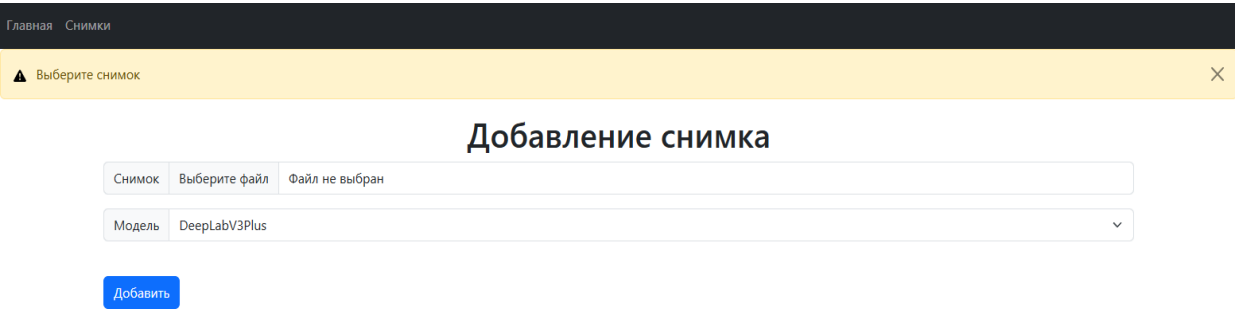


Рисунок 14 – Ввод недопустимых данных Тест добавления нового снимка представлен на рисунке 15

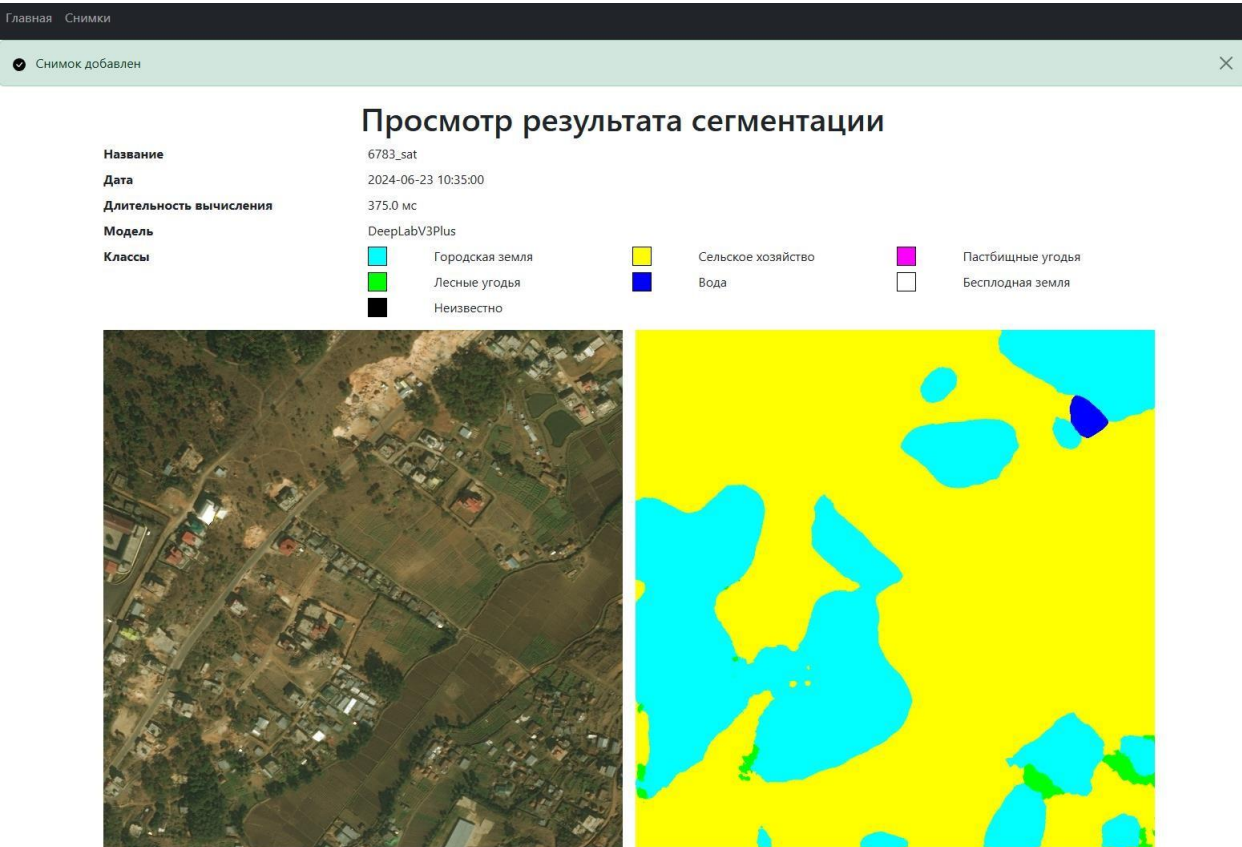


Рисунок 15 – Добавлен новый снимок

Исходя из тестов выше можно сделать вывод, что система работает исправно.

4 Разработка приложения

4.1 Модуль «ГИС»

Модуль «ГИС» предназначен для просмотра и работы с пространственными данными векторных слоев, полученных в процессе обработки фотограмметрических данных.

Основными элементами интерфейса в модуле «ГИС» являются (рисунок 16)

- боковое меню с разделами модуля (пункт 1),
- раздел «Пространственные данные» ,
- элементы,
- картографическая основа и векторные слои,
- блок «Тип карты» (пункт 2),
- блок «Административное деление» (пункт 3),
- блок «Пространственные данные» (пункт 4),
- карта (пункт 5),
- панель инструментов карты (пункт 6),
- кнопка возврата на главную страницу (пункт 7).

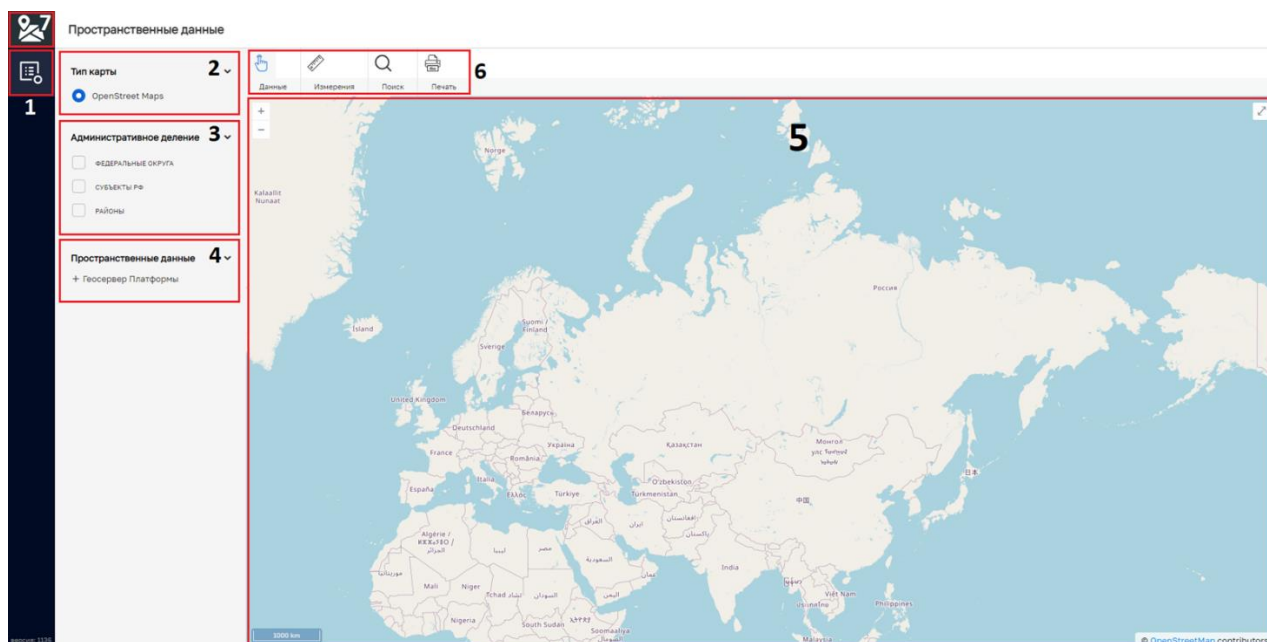
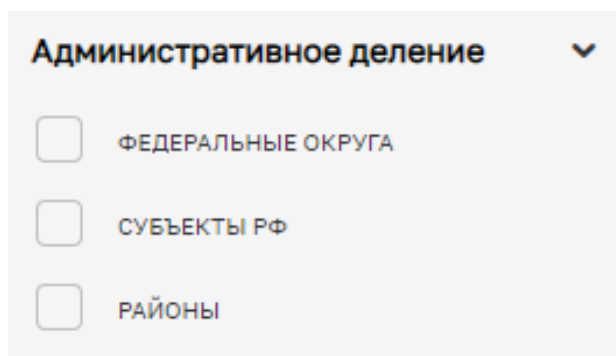


Рисунок 16 – Элементы интерфейса модуля «ГИС»

В качестве картографической основы установлен тип карты «OpenStreet Maps».

В системе реализована возможность выбора одного или нескольких из векторных слоев для отображения административно-территориального деления РФ (рисунок 17).



■ Рисунок 17 – Блок «Административное деление»

Блок «Административное деление» содержит следующие векторные слои:

- «Федеральные округа» (рисунок 18),
- «Субъекты РФ» (рисунок 19),
- «Районы» (рисунок 20).

Чтобы отобразить векторный слой административного деления на карте, следует активировать чек-бокс, расположенный рядом с наименованием необходимого векторного слоя.

Чтобы убрать отображение активированного векторного слоя административного деления, следует деактивировать чек-бокс, расположенный рядом с наименованием необходимого векторного слоя.

По умолчанию на карте слои административного деления не отображены (деактивированы все чек-боксы).

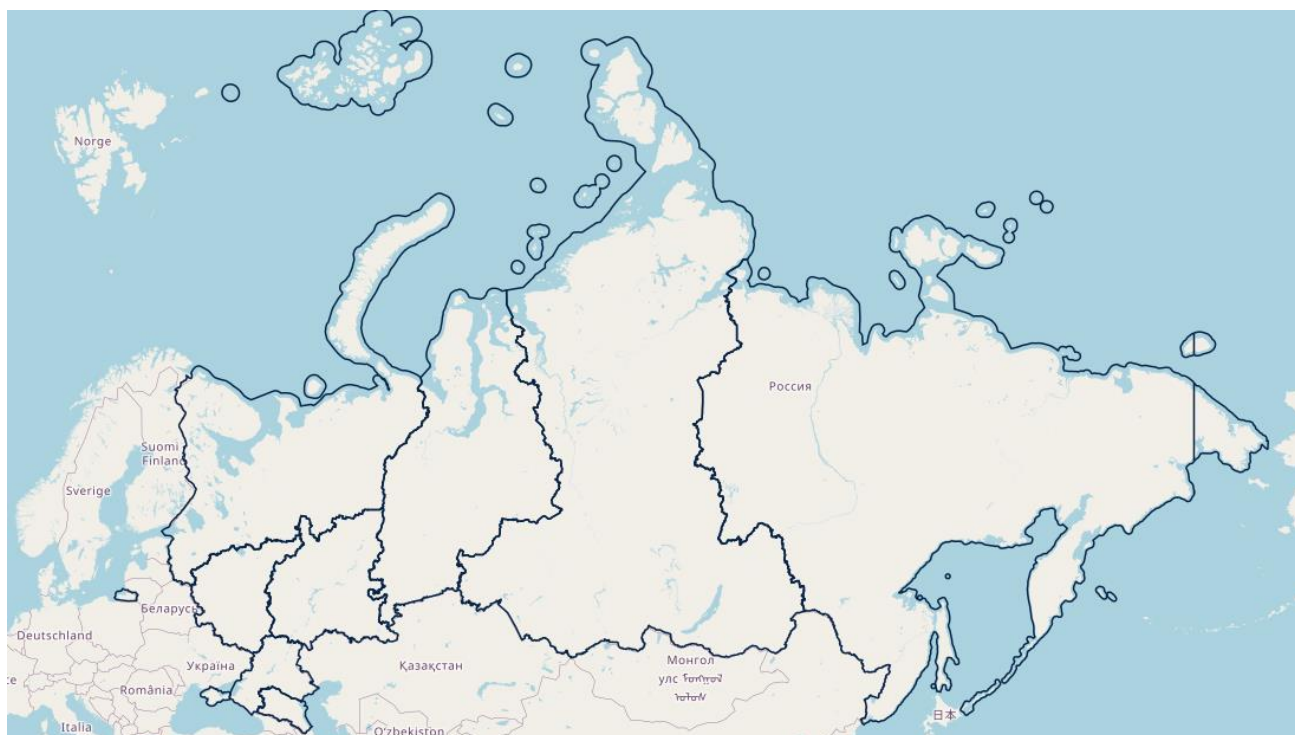


Рисунок 18 – Векторный слой «Федеральные округа»

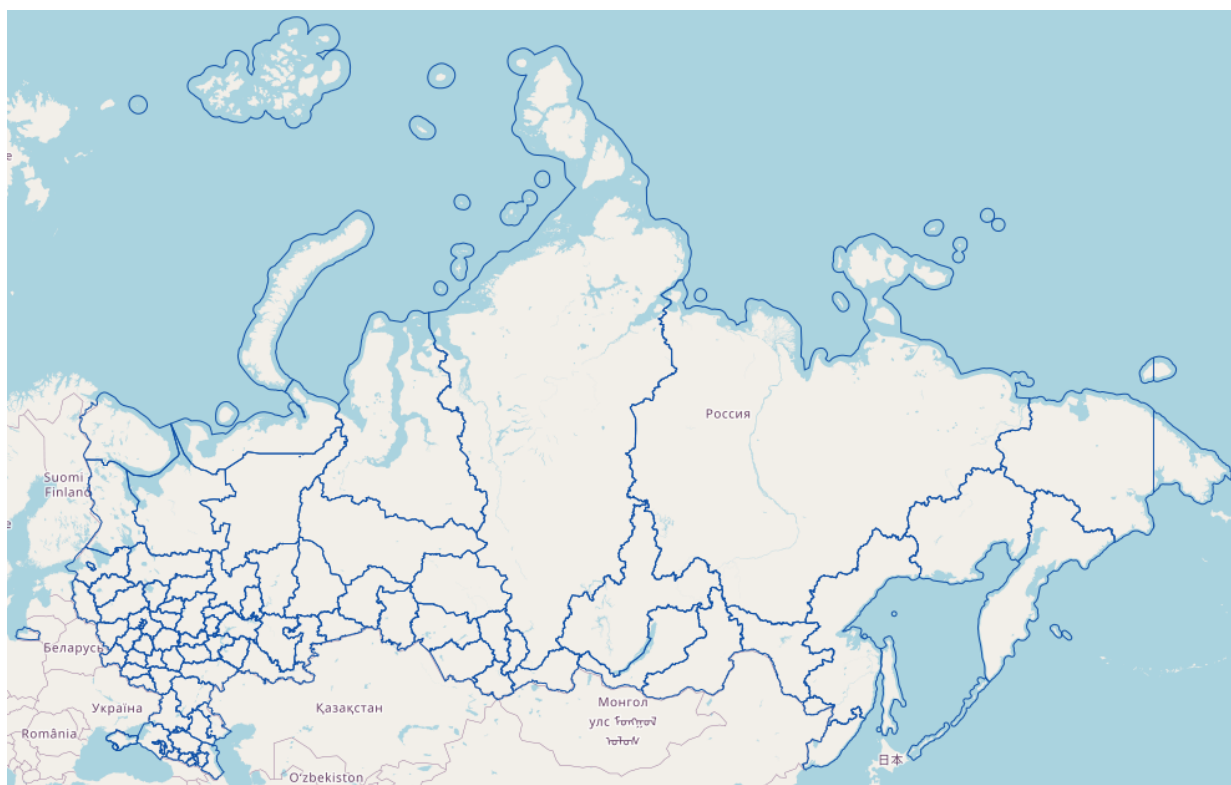


Рисунок 19 – Векторный слой «Субъекты РФ»

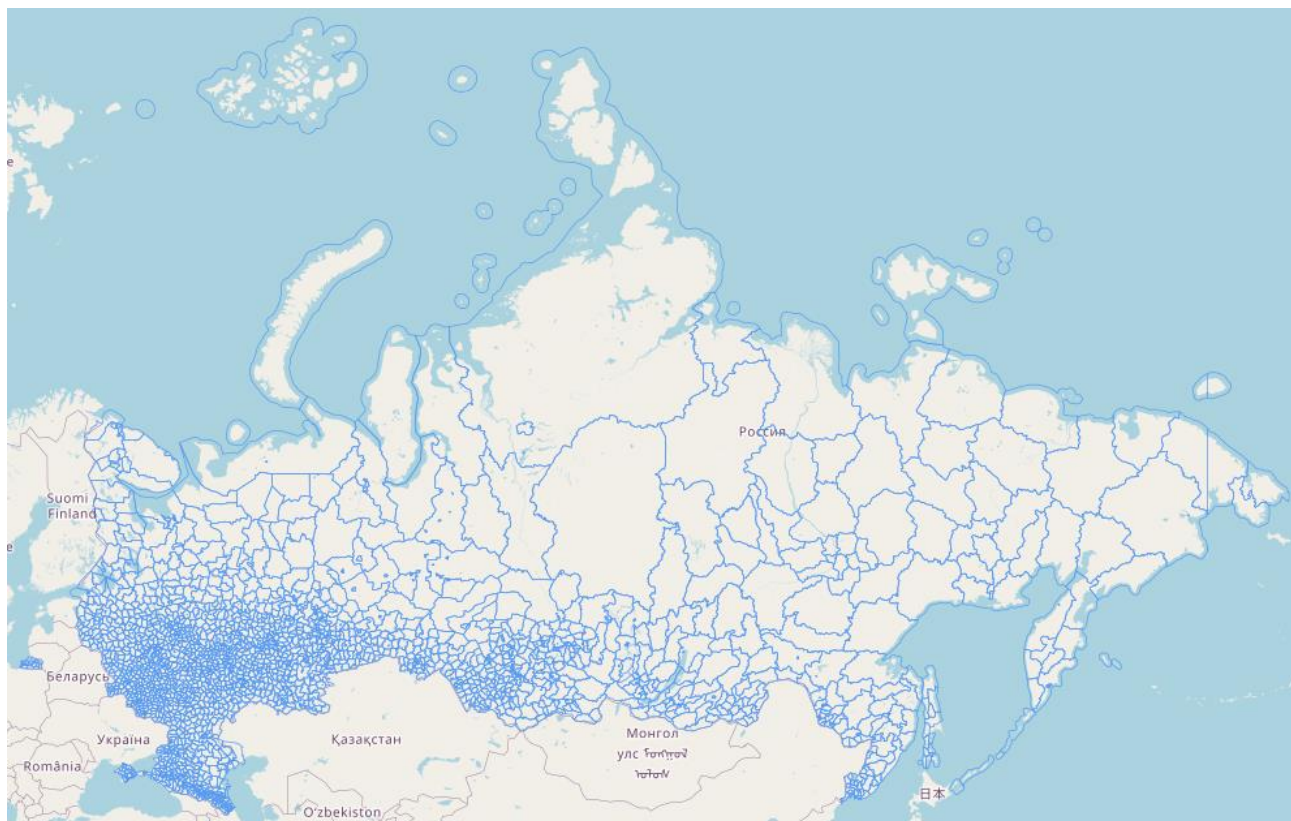


Рисунок 20 – Векторный слой «Районы»

Для просмотра результата процесса интеллектуальной обработки фотограмметрических данных используется блок «Пространственные данные», который включает в себя фотограмметрические данные (растровые данные) из источников системы («Геосервер Платформы»), которые были обработаны, и соответственно, результаты их обработки.

Данные в блоке «Пространственные данные» представлены в древовидной структуре, верхний уровень каталогов которого соответствует источникам фотограмметрических данных, следующий уровень – субъектам РФ, территории которых географически покрывают обработанные фотограмметрические данные из соответствующего источника в рамках процессов.

Древовидная структура представлена в виде иерархии каталогов, открывающаяся в виде раскрывающегося списка (рисунок 21).

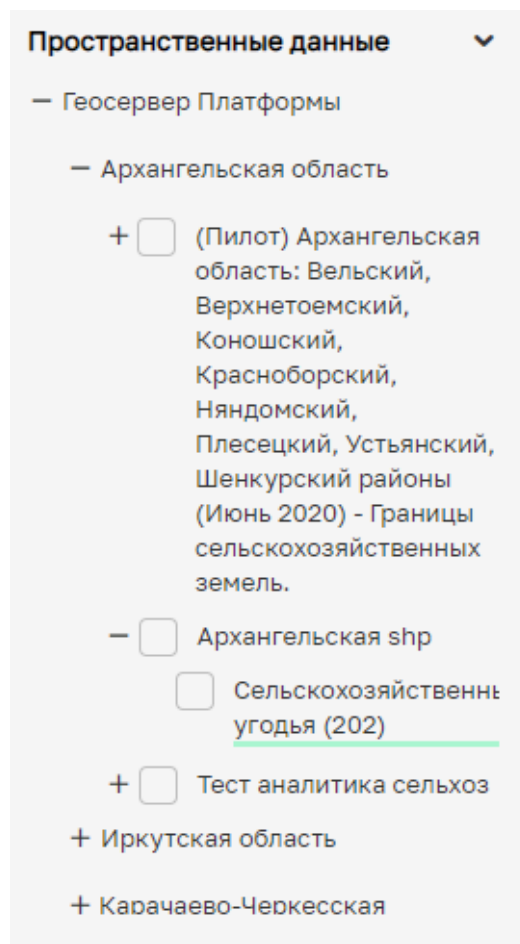


Рисунок 21 – Структура данных в блоке «Пространственные данные»

Для отображения результатов процесса интеллектуальной обработки фотограмметрических данных необходимо выбрать интересующий слой или несколько слоёв, активировав чек-бокс, расположенный рядом с их наименованиями.

После этого на выбранной картографической основе отобразятся следующие данные с их географической привязкой (рисунок 22):

- растровые данные – источник фотограмметрических данных, который был обработан в системе в рамках определенного процесса,
- выбранные векторные данные (слои) с их послойным нанесением.



Рисунок 22 – Отображение растровых и векторных данных по определенному процессу интеллектуальной обработки фотограмметрических данных

По умолчанию отображение пространственных данных отключено.

4.1.1 Карточка атрибутивных данных объекта

Для просмотра подробной информации (атрибутивных данных) о распознанном объекте в рамках конкретного векторного слоя необходимо нажать на его полигон (пункт 1, рисунок 23). Отобразится модальное окно с карточкой выбранного объекта, содержащая атрибутивную информацию (пункт 2, рисунок 23). Для просмотра подробной информации (атрибутивных данных) о распознанном объекте в рамках конкретного векторного слоя необходимо нажать на его полигон. Отобразится модальное окно с карточкой выбранного объекта, содержащая атрибутивную информацию.

Для просмотра подробной информации (атрибутивных данных) о распознанном объекте в рамках конкретного векторного слоя необходимо нажать на его полигон. Отобразится модальное окно с карточкой выбранного объекта, содержащая атрибутивную информацию.

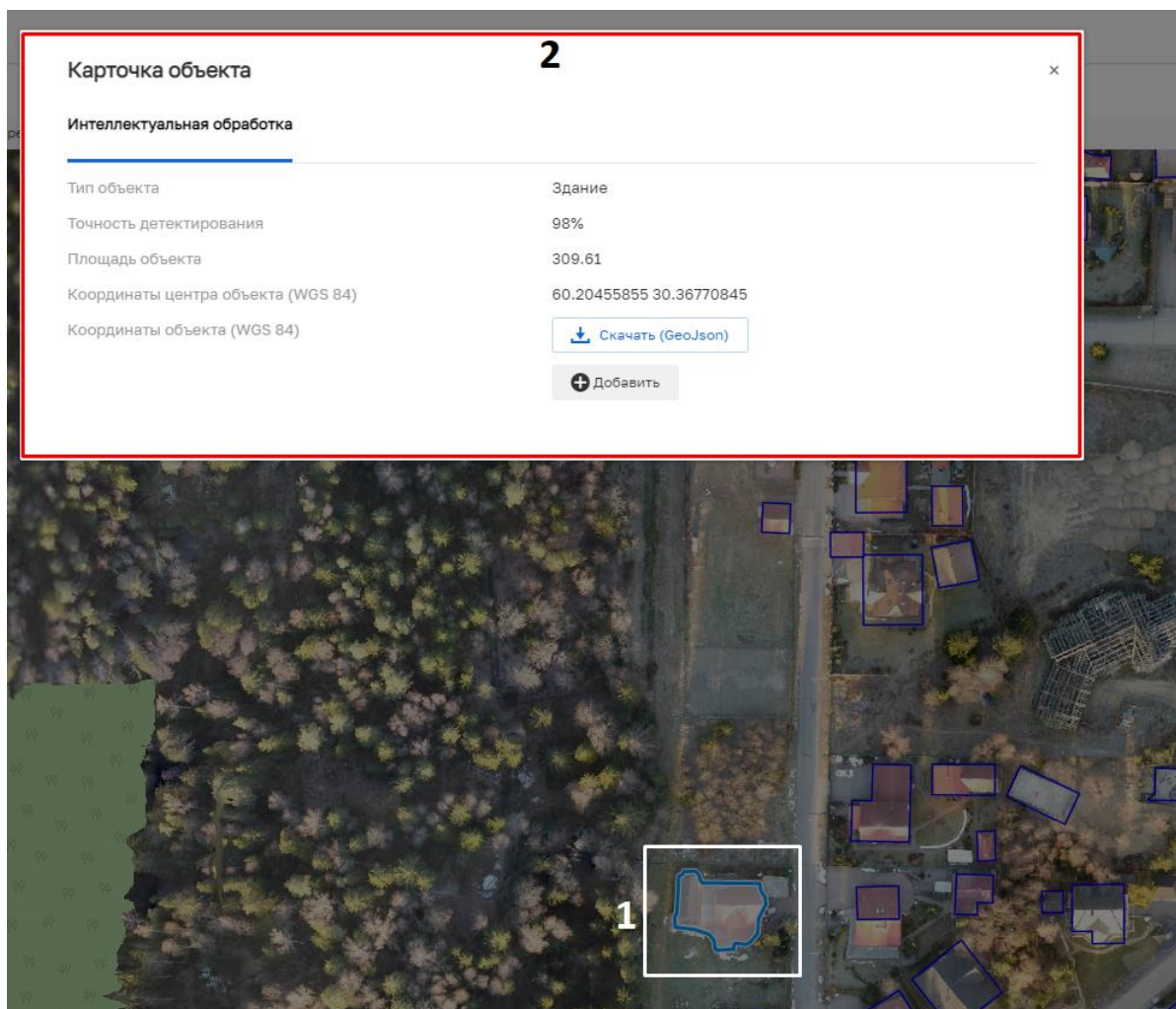


Рисунок 23 – Карточка атрибутивных данных выбранного объекта

Карточка атрибутивных данных объекта может содержать в себе набор информации, в зависимости от типа источника данных, из которого была обогащена информация об объекте.

При необходимости имеется возможность добавить дополнительные атрибутивные данные в карточку объекта. Для этого следует нажать на кнопку «Добавить» на вкладке «Интеллектуальная обработка», после чего заполнить поля «Название» и «Значение» и нажать на кнопку «Сохранить» (рисунок 24).

При нажатии на кнопку «Добавить» в карточке выбранного объекта появятся дополнительные поля для добавления дополнительных атрибутивных данных.

Для редактирования добавленных дополнительных атрибутивных данных необходимо в карточке выбранного объекта нажать на кнопку «Редактировать», расположенную под необходимым атрибутом.

Для удаления добавленных дополнительных атрибутивных данных необходимо в карточке выбранного объекта нажать на кнопку «Редактировать», расположенную под необходимым атрибутом, а затем на иконку «Корзина».

Карточка объекта

Интеллектуальная обработка

Тип объекта	Здание
Точность детектирования	98%
Площадь объекта	309.61
Координаты центра объекта (WGS 84)	60.20455855 30.36770845
Координаты объекта (WGS 84)	Скачать (GeoJson)

Дополнительные данные по объекту

Название	Значение
----------	----------

[+](#) Добавить [Сохранить](#) ☐ Применить ко всем элементам слоя

Рисунок 24 – Добавление атрибутивных данных по объекту в его карточку

В карточке атрибутивных данных объекта имеется возможность скачать координаты выбранного объекта в проекции WGS-84. Для этого следует нажать на полигон необходимого объекта на выбранном векторном слое и в карточке объекта нажать на кнопку «Скачать (GeoJson)» (рисунок 25). Файл автоматически загрузится на компьютер и будет иметь формат *.geojson.

Карточка объекта

Интеллектуальная обработка

Тип объекта	Здание
Точность детектирования	98%
Площадь объекта	309.61
Координаты центра объекта (WGS 84)	60.20455855 30.36770845
Координаты объекта (WGS 84)	Скачать (GeoJson)

[+](#) Добавить

Рисунок 25 – Кнопка «Скачать (GeoJson)»

4.1.2 Панель инструментов карты

Панель инструментов карты включает в себя следующие инструменты для работы:

- перемещение по карте и выбрать объектов на ней для просмотра атрибутивных данных,
- измерение расстояния между объектами на карте,
- поиск объектов в рамках выбранного (активированного) векторного слоя,
- скачивание изображения рабочей области интерфейса, в которой могут быть отображены растровые и векторные данные для последующей печати,
- изменение масштаба карты (увеличение/уменьшение),
- переход в полноэкранный режим.

Пользователь имеет возможность измерить расстояние между двумя точками (или между несколькими точками) на карте. Для этого необходимо выполнить следующие четыре действия (рисунок 26):

- 1) нажать на иконку «Линейка»,
- 2) выбрать место на карте (точку), с которого начнется измерение (пункт 1),
- 3) нажать на левую кнопку мыши,
- 4) вести курсор мыши к месту на карте (к точке), на котором закончится измерение (пункт 2).

Над курсором мыши отображается расстояние между точками на карте.



Рисунок 26 – Измерение расстояния между точками

Для прекращения использования линейки требуется повторно нажать на иконку «Линейка» или на иконку «Рука».

Так же реализована возможность поиска объектов, распознанных на изображении в процессе интеллектуальной обработки фотограмметрических данных в рамках выбранного (активированного) векторного слоя.

В системе пользователю предложена печать изображения рабочей области карты. Для этого требуется активировать необходимые векторные слои и (или) переместиться на карте на область, изображение которой необходимо получить, и нажать на иконку «Принтер». После этого изображение рабочей области карты автоматически загрузится на компьютер в виде файла формата *.jpeg.

Чтобы распечатать изображение, необходимо открыть скаченный файл и нажать на кнопку «Печать» (рисунок 27).

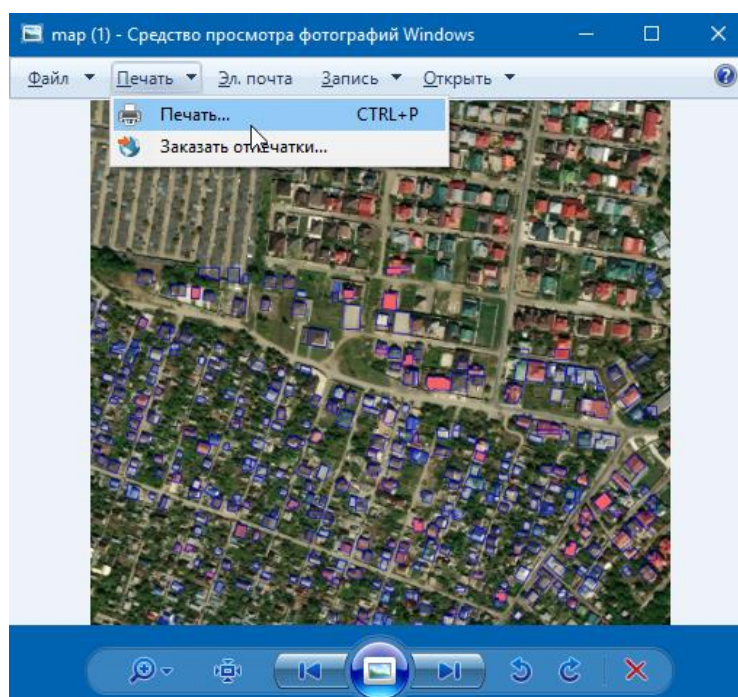


Рисунок 27 – Пример скаченного файла, изображение рабочей области

Для изменения масштаба карты используются иконки «Плюс» и «Минус». Также можно воспользоваться колесиком мыши.

При масштабировании карты изменяется визуальное представление данных на картографической основе.

Например, при приближении карты отображается геометрия полигонов распознанных объектов в виде наложенного векторного слоя на источник

фотограмметрических данных. При отдалении карты распознанные объекты класстерируются в зависимости от уровня отдаления, отображается информация о количестве объектов на слое внутри круга (рисунок 28). При масштабировании карты, число внутри круга будет увеличиваться, суммируя соседние круги (при отдалении) или уменьшаться, вычитая из себя соседние круги (при приближении).

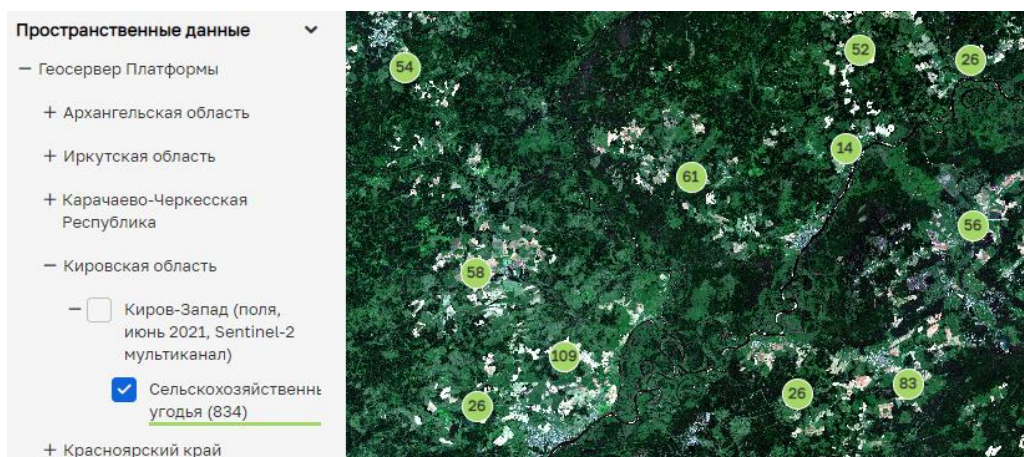


Рисунок 28 – Информация о количестве объектов на слое внутри круга

При нажатии на круг класстеризованных объектов автоматически произойдет приближение карты на 1 значение (zoom), а сам круг будет разделен на несколько кругов, которые так же содержат в себе информацию о количестве входящих в него объектов. Нажимать на круги можно до тех пор, пока не отобразятся полигоны объектов (рисунок 29).

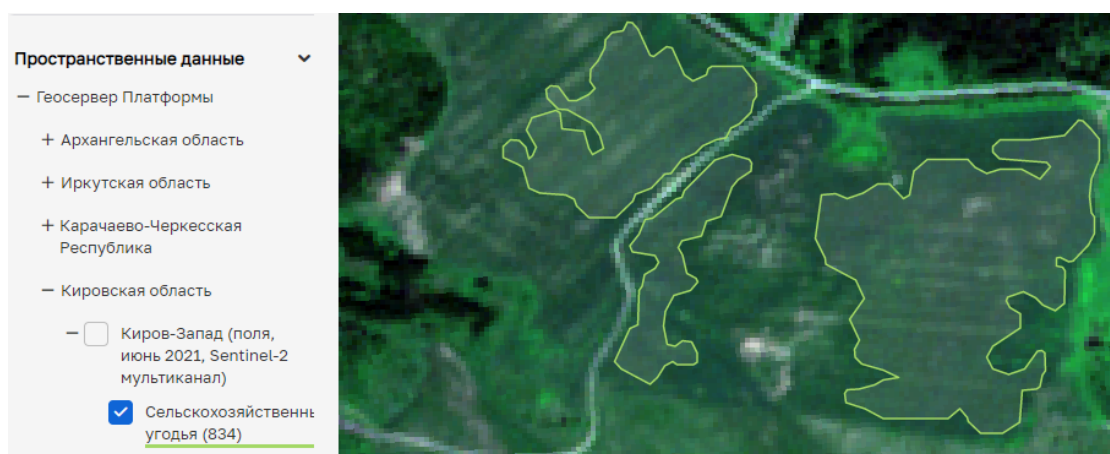


Рисунок 29 – Отображение объектов, после нескольких нажатий на круг

Для просмотра карты в полноэкранном режиме необходимо нажать на иконку «Две стрелочки». Чтобы выйти из полноэкранного режима, требуется нажать на иконку «Крестик» или на клавишу «Esc».

4.2 Модуль «Интеллектуальная обработка»

В модуле «Интеллектуальная обработка» осуществляется работа с процессами обработки фотограмметрических данных.

Основными элементами интерфейса в модуле «Интеллектуальная обработка» являются (рисунок 31):

- боковое меню с разделами модуля (пункт 1),
- раздел «Процессы»,
- элементы,
- панель с информацией об отрасли (пункт 2),
- панель фильтрации элементов (пункт 3),
- панель перечня элементов (пункт 4),
- панель просмотра элемента (пункт 5),
- кнопка возврата на главную страницу (пункт 6),
- кнопка создания элемента (пункт 7).

В панели перечня элементов напротив наименования субъекта РФ отображается иконка с количеством процессов обработки фотограмметрических данных, входящих в него (рисунок 30).

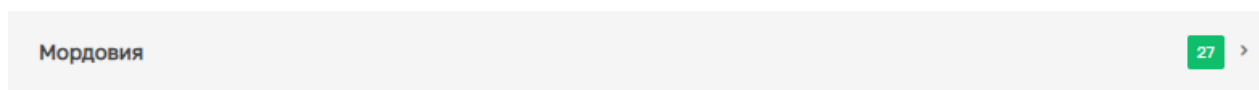


Рисунок 30 – Количество процессов, содержащих данный субъект РФ

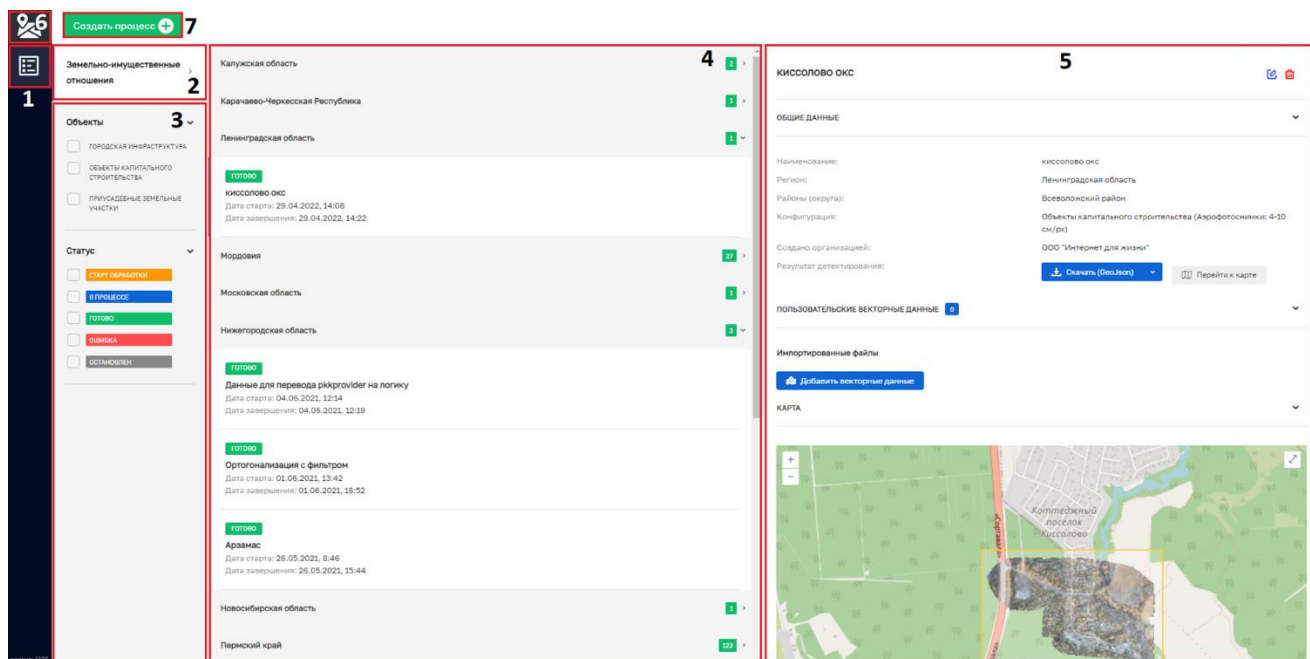


Рисунок 31 – Элементы интерфейса модуля «Интеллектуальная обработка»

4.2.1 Просмотр информации о процессе

Для просмотра информации о процессе обработки фотограмметрических данных, необходимо нажать на наименование субъекта РФ, в котором находится необходимый процесс, затем нажать на наименование процесса из появившегося списка.

В случае, когда не был выбран процесс обработки фотограмметрических данных, в панели просмотра элемента отображается соответствующее сообщение.

В панели просмотра элемента отображаются следующие информационные блоки выбранного процесса обработки фотограмметрических данных:

- «Общие данные» – содержит общую информацию о выбранном процессе обработки фотограмметрических данных,
- «Пользовательские векторные данные» – содержит информацию о загруженных пользователями системы данных в выбранный процесс,
- «Карта» – содержит фотограмметрические данные территории, в которой выполняется или выполнялся определенный процесс их обработки,
- «Процессы» – список процессов (технологических процессов), которые выполняются или выполнялись при обработке фотограмметрических данных, с обозначением текущего статуса каждого из процессов, времени их выполнения.

Чтобы развернуть/свернуть определенный информационный блок, требуется нажать на иконку «Стрелка».

Для скачивания результатов процесса обработки фотограмметрических данных необходимо нажать на кнопку «Скачать (GeoJson)» (Файл загрузится на компьютер автоматически и будет иметь расширение *.geojson).

Если в процессе обработки фотограмметрических данных были детектированы объекты, относящиеся к разным классам объектов (например, частные дома, теплицы, автомобили и другие), то имеется возможность скачать как общий файл, так и отдельный файл под каждый класс детектированных объектов.

Чтобы получить общий файл с результатами обработки фотограмметрических данных, необходимо нажать на кнопку «Скачать (GeoJson)».

Чтобы получить отдельный файл необходимо нажать на иконку «Стрелочка», расположенную на кнопку «Скачать (GeoJson)», и выбрать необходимый класс детектированных объектов из предложенного списка (рисунок 32). Файл загрузится на компьютер автоматически и будет иметь расширение *.geojson.

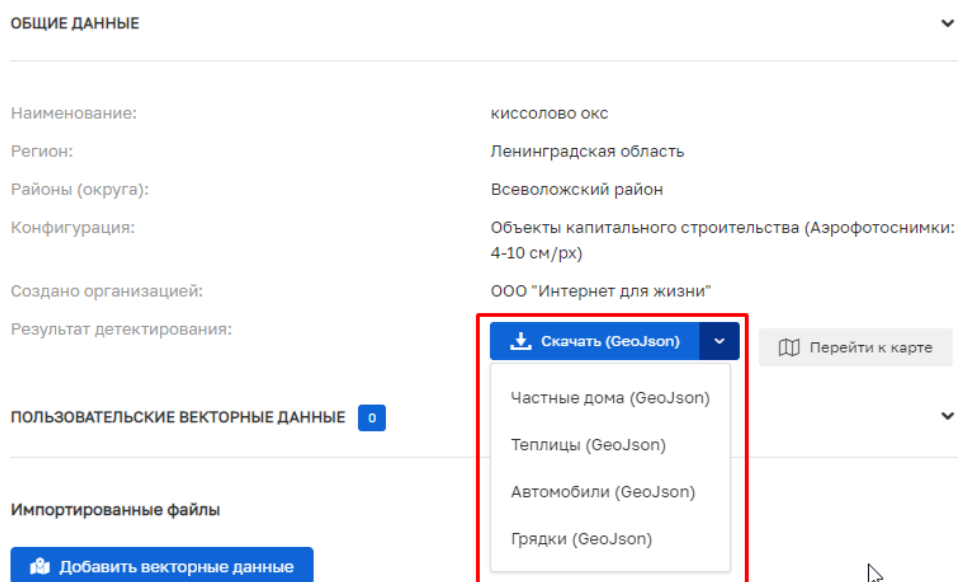


Рисунок 32 – Отображение списка объектов для скачивания

Для просмотра результатов процесса обработки фотограмметрических данных и дальнейшей работы с ними, необходимо нажать на кнопку «Перейти к карте».

После этого произойдет переход в модуль «ГИС» раздел «Пространственные данные» с отображением источника фотограмметрических данных, используемый в

процессе обработки, наложенного на определенную картографическую основу, на котором отобразятся результаты в виде полигонов детектируемых объектов в векторном формате (рисунок 33).

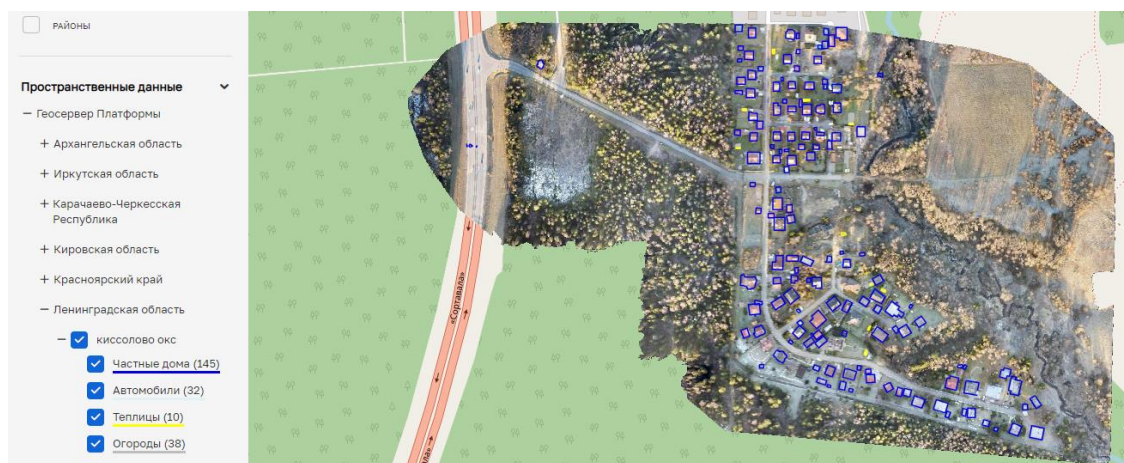


Рисунок 33 – Отображение результатов выполнения процесса обработки фотограмметрических данных на геопортале Платформы

В информационном блоке «Пользовательские векторные данные» отображается информация о загруженных пользователями системы векторных данных в выбранный процесс обработки фотограмметрических данных:

- наименование – наименование пользовательских векторных данных,
- количество слоев – количество слоев, содержащихся в загруженном файле,
- дата и время импорта – данная информация отображается при наведении курсора мыши на наименование файла.

В данном блоке предусмотрена возможность загрузки пользовательских векторных данных в выбранный процесс обработки фотограмметрических данных, для последующего их послойного нанесения и отображения. Для этого необходимо выполнить следующие четыре действия:

- 1) нажать на кнопку «Добавить векторные данные»,
- 2) в появившемся окне нажать на выделенную пунктиром область, в которой написано «Выбрать файл», и выбрать файл в появившемся окне, или переместить из папки на компьютере необходимый файл(ы) в данную область (рисунок 34),
- 3) нажать на кнопку «Загрузить»,
- 4) дождаться окончания загрузки (рисунок 35).

В случае, если был выбран неверный файл, его можно удалить, нажав на иконку «Корзина» (если не была нажата кнопка «Загрузить») или «Остановить импорт» (если файл начал загружаться).

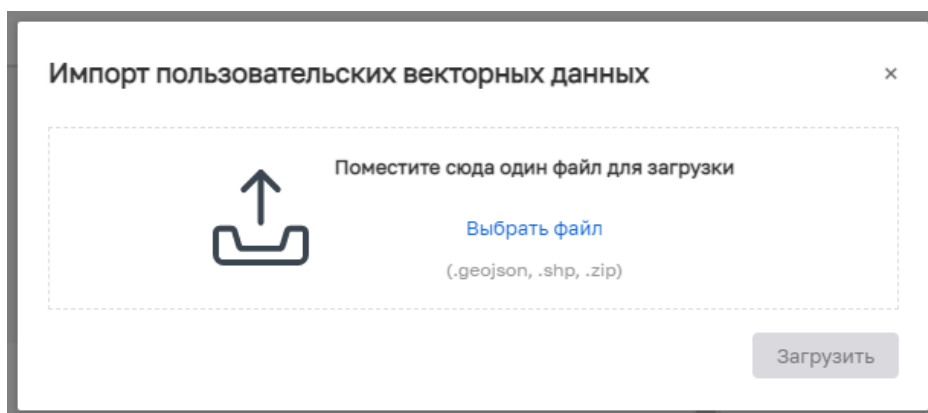


Рисунок 34 – Область для загрузки файла в Платформу

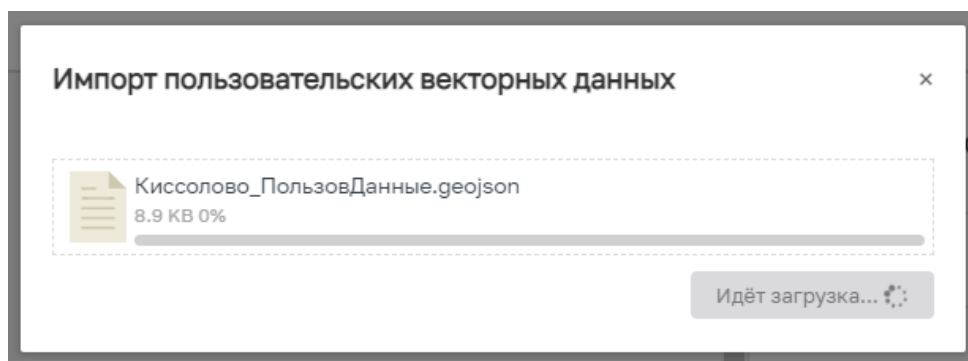


Рисунок 35 – Отображение прогресса загрузки выбранного файла

Загружаемые файлы (пользовательские векторные данные) должны быть одного из следующих форматов:

- *.geojson,
- *.shp,
- *.zip.

При загрузке файла некорректного формата, в интерфейсе отобразится ошибка: «Неверный формат файла».

После начала импорта файла, необходимо указать систему координат загружаемых векторных данных в определенную экранную форму. Для этого необходимо нажать на соответствующее поле и выбрать один из предложенных вариантов или ввести значение с клавиатуры самостоятельно в следующем виде: 101303, где: 13 – код региона; 03 – номер зоны.

После выбора или ввода системы координат следует нажать на кнопку «Применить» и дождаться окончания загрузки файла (рисунок 36).

Чтобы прервать импорт файла, требуется нажать на кнопку «Остановить импорт» и подтвердить действие. Процесс будет прерван, файл автоматически будет удален из системы.

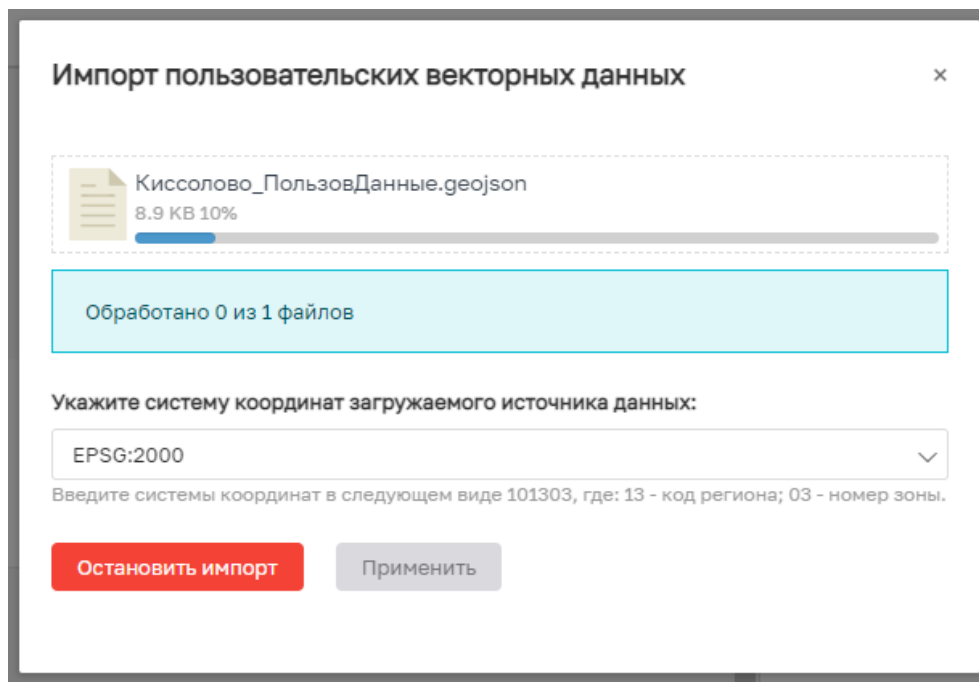


Рисунок 36 – Процесс импорта файла

После успешного импорта файла, в интерфейсе отобразится сообщение об успешно загруженном файле (рисунок 37). В информационном блоке «Пользовательские векторные данные» будет отображена информация о загруженном файле.

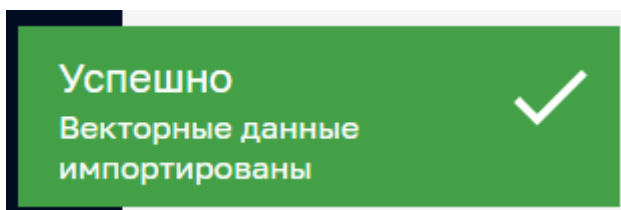


Рисунок 37 – Сообщение об успешном импорте файла

Для удаления импортированных файлов из информационного блока «Пользовательские векторные данные» выбранного процесса необходимо нажать на иконку «Корзина». Файлы будут удалены из системы без возможности их восстановления.

Для просмотра загруженных пользовательских векторных данных на карте следует нажать на кнопку «Перейти к карте». После этого произойдет переход в модуль «ГИС» с развернутым процессом интеллектуальной обработки с соответствующими ему данными. В каталоге векторных слоев пользовательские векторные данные будут иметь имя импортированного файла (рисунок 38).

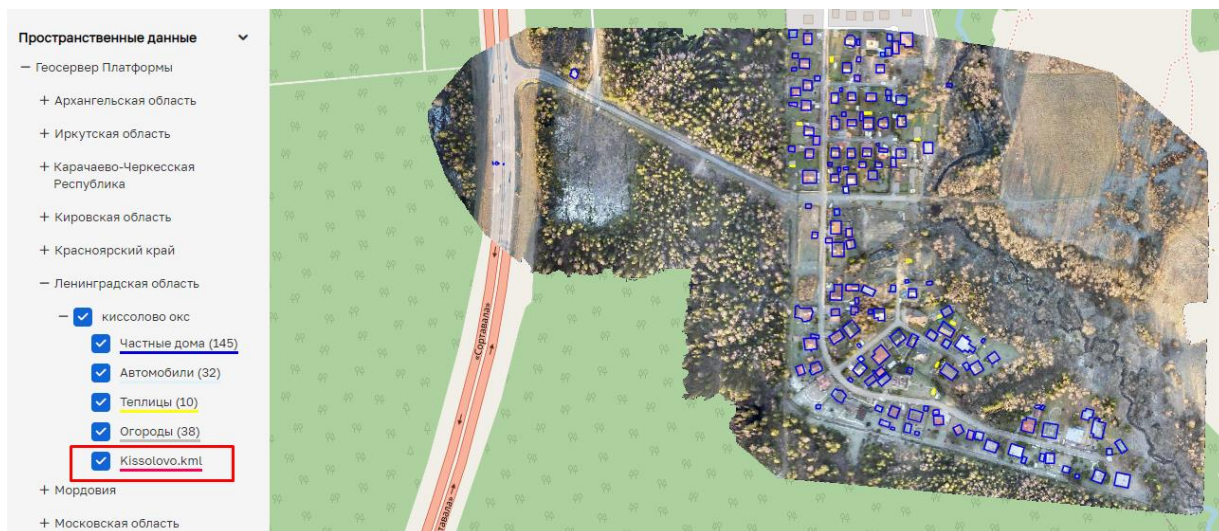


Рисунок 38 – Отображение на карте пользовательских данных

В информационном блоке «Карта» отображена карта территории (зона обработки на картографической основе, или зона обработки на источнике фотограмметрических данных), в выделенной области которой обрабатываются фотограмметрические данные в рамках выбранного процесса

В информационном блоке «Процессы» представлен перечень процессов, выполняемые в рамках обработки фотограмметрических данных (рисунок 39).

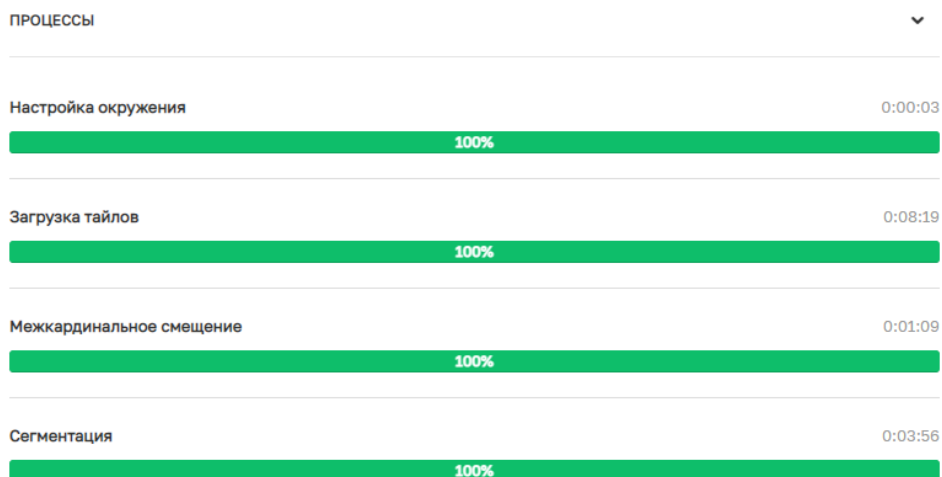


Рисунок 39 – Информационный блок «Процессы»

Каждый процесс имеет полосу загрузки с процентом его завершенности выполнения. После успешного выполнения процесса напротив его наименования отображается время его выполнения.

4.2.2 Создание, редактирование, удаление и управление выполнением процесса обработки данных

Для создания процесса обработки фотограмметрических данных следует нажать на кнопку «Создать процесс» в интерфейсе раздела «Процессы», после этого отобразится форма «Создание процесса» (рисунок 40).

Создание процесса

Название процесса*

Конфигурация*

Тип источника данных:

☒ Геосервер Платформы

Источник данных*

Необходимо выбрать зону обработки на карте или выбрать весь снимок.

Зоны обработки: 0

Запустить процесс

Дополнительные параметры

powered by ace

Выбор зоны обработки

Рисунок 40 – Форма создания обработки фотограмметрических данных

Для запуска (инициализации) процесса обработки фотограмметрических данных в форме для создания процесса требуется указать следующие данные:

- «Наименование процесса» – поле для ввода наименования создаваемого процесса,

- «Конфигурация» – поле для выбора конфигурации из выпадающего списка.

Конфигурация – это один из конвейеров обработки фотограмметрических данных, предустановленный и настроенный в системе. Каждое наименование конфигурации из выпадающего списка отображает информацию о классе распознаваемых объектов на определенном типе фотограмметрических данных,

- «Тип источника данных: «Геосервер Платформы» – тип источника данных, в котором требуется указать (выбрать) сам источник фотограмметрических данных

(аэрофото-/спутниковый снимок). Для этого требуется нажать на пустое поле «Источник данных» и выбрать один из источников из выпадающего списка. После выбора источника фотограмметрических данных он отобразится в правой части интерфейса «Создание процесса» в блоке «Выбор зоны обработки» на картографической подложке «OpenStreetMap».

Далее требуется определить зону(ы) интеллектуальной обработки. Сделать это можно двумя способами:

- указать зону(ы) интеллектуальной обработки на источнике фотограмметрических данных или на картографической подложке,
- обработать весь снимок.

В первом случае (указать зону(ы) интеллектуальной обработки на источнике фотограмметрических данных или на картографической подложке), необходимо нажать на кнопку «Добавление зоны» и нарисовать границы области. Выделенная зона интеллектуальной обработки на источнике фотограмметрических данных отобразится в виде полигона с желтыми границами (рисунок 41).



Рисунок 41 – Отображение выделенной зоны интеллектуальной обработки при выборе типа источника данных «Геосервер Платформы»

Во втором случае (обработать весь снимок), необходимо нажать на чек-бокс «Обработать весь снимок». Тогда будет выбрана вся область источника фотограмметрических данных для интеллектуальной обработки.

После указания зон(ы) интеллектуальной обработки Платформа автоматически определит входящие в нее(них) регион(ы) и район(ы).

Если платформа обнаружила пересечение двух и более регионов, пользователю необходимо указать приоритетный регион, нажав на кнопку «Разрешить конфликт регионов» (рисунок 42). В появившемся окне следует выбрать регион, приоритетный для каталогизации (рисунок 43).

Зоны обработки: **1**

Регион: Рязанская область

Платформа обнаружила пересечение двух и более регионов, примите решение для каталогизации

Разрешить конфликт регионов

Районы: Спасский район, Касимовский район, Пителинский район, городской округ Сасово, Сасовский район, Кадомский район, Ермишинский район, Шацкий район, городской округ Выкса, Дивеевский район, Вознесенский район, городской округ Первомайск, Теньгушевский район, Zubovo-Polyanskiy район, Torbееvskiy район, Atyurevskiy район, Kovylkinskiy район, Temnikovskiy район, Elnikovskiy район, Krasnooslobodskiy район, ZATO Sarov

Рисунок 42 – Конфликт регионов

Разрешение конфликта регионов

Платформа обнаружила следующие регионы, выберите приоритетный для каталогизации

☐ Рязанская область

☐ Пензенская область

☐ Нижегородская область

☐ Республика Мордовия

Подтвердить

Рисунок 43 – Выбор приоритетного региона

Чтобы указать наименование региона самостоятельно, требуется нажать на пустое поле под назначенным автоматически регионом и выбрать значение из списка или ввести его самостоятельно (рисунок 44).

Если регион определен неверно выберите его самостоятельно из списка

Назначенный регион для каталогизации: Пермский край

Республика Адыгея

Республика Башкортостан

Республика Бурятия

Рисунок 44 – Поле для изменения региона зоны обработки

Чтобы указать наименования районов самостоятельно, следует нажать на иконку «Шестеренка», (доступна только при обработки выделенной зоны на карте,. В открывшемся окне необходимо изменить наименование района. Если требуется указать несколько районов, следует перечислить их наименования через запятую (рисунок 45).

Изменение района

×

Район

Пермский район, Свердловский район

Вы можете изменить названия районов в текстовом поле (Пример: Район 1,Район 2)

Подтвердить

Закреть

Рисунок 45 – Окно изменения района зоны обработки

После заполнения всех полей формы создания процесса требуется нажать на кнопку «Запустить процесс», чтобы создать (инициализировать) процесс обработки фотограмметрических данных. Статус выполнения процесса обработки фотограмметрических данных примет значение «Старт обработки».

В панели просмотра элемента раздела «Процессы» имеется возможность редактирования, удаления и управления выполнением выбранного процесса.

Редактировать данные (регион, районы, конфигурация, источник данных) уже созданного процесса обработки не является возможным. Можно изменить только наименование процесса интеллектуальной обработки.

Также реализована возможность удаления процесса, нажав на иконку «Корзина». После этого процесс обработки фотограмметрических данных будет удален из системы без возможности его восстановления.

Также предусмотрена возможность изменения статуса активного процесса обработки фотограмметрических данных.

Чтобы остановить активный процесс обработки данных, требуется выбрать активный процесс обработки и нажать на иконку «Пауза». Статус выполнения процесса обработки фотограмметрических данных примет значение «Остановлен».

Чтобы запустить остановленный процесс обработки данных, требуется выбрать остановленный процесс обработки и нажать на иконку «Возобновить». Статус выполнения процесса обработки фотограмметрических данных примет значение «В процессе».

4.3 Модуль «Пространственные данные»

В модуле «Пространственные данные» осуществляется управление источниками данных геосервера системы для обеспечения возможности их использования в процессах обработки фотограмметрических данных.

Основными элементами интерфейса модуля «Пространственные данные» являются (рисунок 47):

- боковое меню с разделами и подразделами модуля (пункт 1),
- раздел «Источники данных» с подразделами (пункт 2),
- «Геосервер Платформы»,
- элементы,
- панель перечня элементов (пункт 3),
- панель просмотра элемента (пункт 4),
- кнопка возврата на главную страницу (пункт 5),
- кнопка создания элемента (пункт 6),
- панель поиска элементов (пункт 7).

В панели перечня элементов напротив наименования субъекта РФ отображается иконка с количеством источников фотограмметрических данных, входящих в него (рисунок 46).

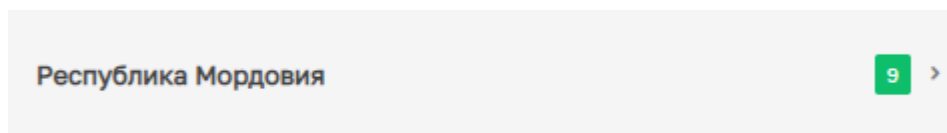


Рисунок 46 – Количество источников данных данного субъекта РФ

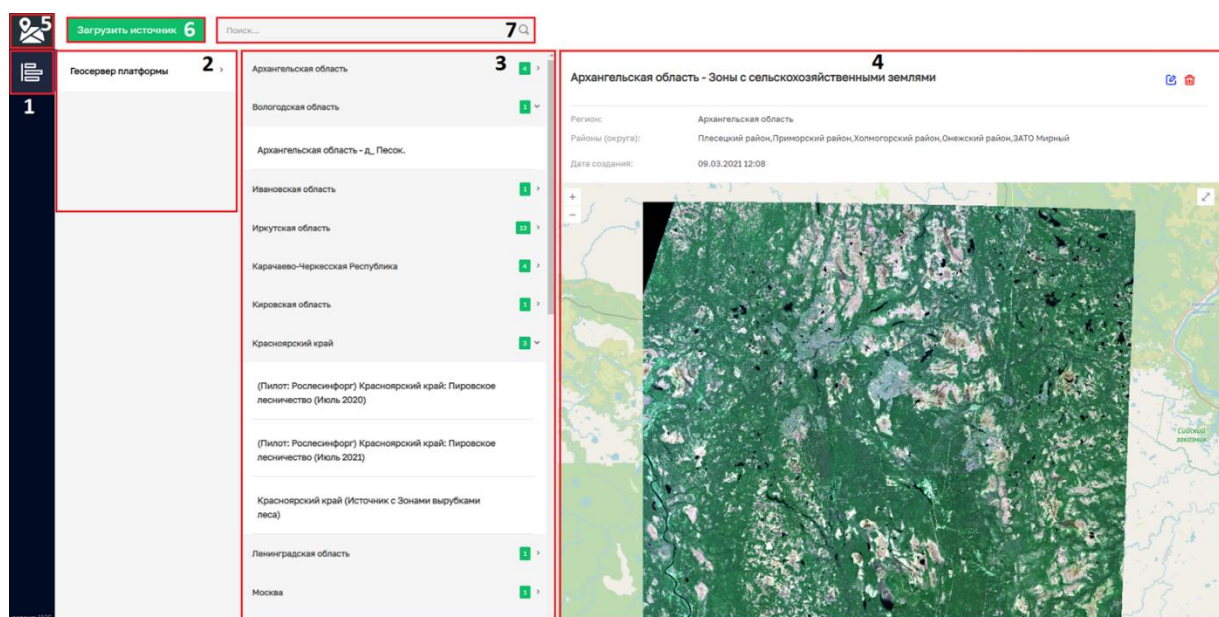


Рисунок 47 – Элементы интерфейса модуля «Пространственные данные» для просмотра информации об источнике данных требуется нажать на наименование субъекта РФ, в котором находится необходимый источник данных, затем нажать на наименование источника данных из появившегося списка.

В панели просмотра элемента отобразится информация о выбранном источнике данных (пункт 4, рисунок 47).

В случае, когда в панели перечня элементов не был выбран источник данных, в панели просмотра элемента отображается соответствующее сообщение.

При выборе в панели перечня элементов источника данных, в панели просмотра элемента отображается следующая информация о выбранном источнике:

- «Наименование» – наименование выбранного источника данных. Задается при создании источника данных,

- «Регион» – регион РФ, который покрывает выбранный источник фотограмметрических данных. Автоматически определяется системой после загрузки фотограмметрических данных,

- «Районы (округа)» – районы (округа) региона РФ, который покрывает выбранный источник фотограмметрических данных. Автоматически определяются системой после загрузки фотограмметрических данных,

- «Дата создания» – дата, когда был создан в системе выбранный источник данных. Формируется автоматически после создания источника данных,

– визуализация источника данных на карте – фотограмметрические данные (аэрофото-/спутниковые снимки), наложенные с географической привязкой на картографическую основу «OpenStreetMap». Формируется после создания источника данных.

4.3.1 Создание, редактирование, удаление источника данных

Для создания нового источника данных необходимо нажать на кнопку «Загрузить источник». После этого откроется форма создания нового источника (рисунок 48). Необходимо заполнить все поля формы и загрузить файлы в формате *.tiff, *.tif в качестве источника данных.

Рисунок 48 – Форма создания нового источника данных

Чтобы загрузить файлы в качестве источника данных, требуется нажать на выделенную пунктиром область, в которой написано «Выбрать файл» или переместить необходимый файл из папки на компьютере в эту область.

Загружаемый файл должен соответствовать одному из следующих форматов:

- *.tiff,
- *.tif.

При загрузке файла некорректного формата (отличных от предусмотренных форматов) в интерфейсе формы создания источника данных отобразится ошибка: «Ошибка загрузки! Некорректный формат файла».

При нажатии на кнопку «Загрузить» без прикрепления файла и/или указания наименования в интерфейсе формы создания источника данных отобразится ошибка: «Ошибка отправки! Не заполнено имя. Не выбраны файлы».

После выбора файла, появляется возможность загрузить его в систему или удалить, при необходимости (рисунок 49).

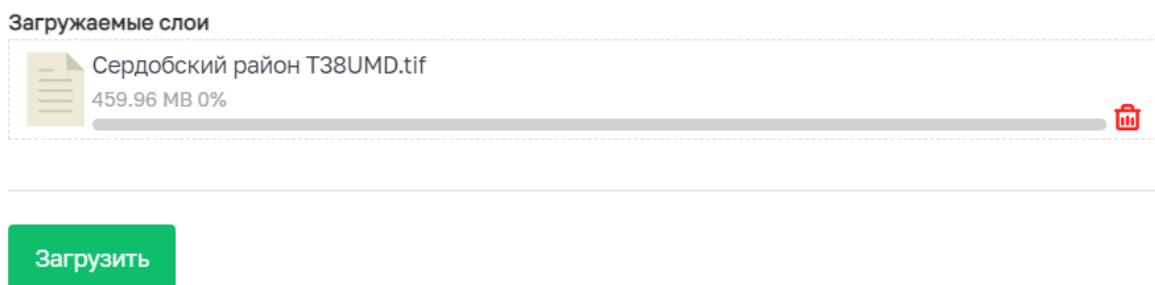


Рисунок 49 – Отображение загружаемого файла

Для удаления выбранного файла следует нажать на иконку «Корзина».

Для загрузки выбранного файла в качестве источника данных требуется нажать на кнопку «Загрузить».

При загрузке файлов в качестве источника данных без содержания в них информации о системе координат, система потребует указать систему координат загружаемых фотограмметрических данных в определенную экранную форму. Для этого необходимо нажать на соответствующее поле и выбрать один из предложенных вариантов или ввести значение с клавиатуры самостоятельно в следующем виде: 101303, где: 13 – код региона; 03 – номер зоны.

После выбора или ввода системы координат следует нажать на кнопку «Применить» для присвоения указанной системы координат загружаемым фотограмметрическим данным. Для каждого файла требуется указать соответствующую ему систему координат.

При загрузке файлов в качестве источника данных возможен случай, когда системой автоматически определяется более одного региона РФ, который покрывает выбранный источник фотограмметрических данных. В данном случае требуется указать приоритетный регион (рисунок 50).

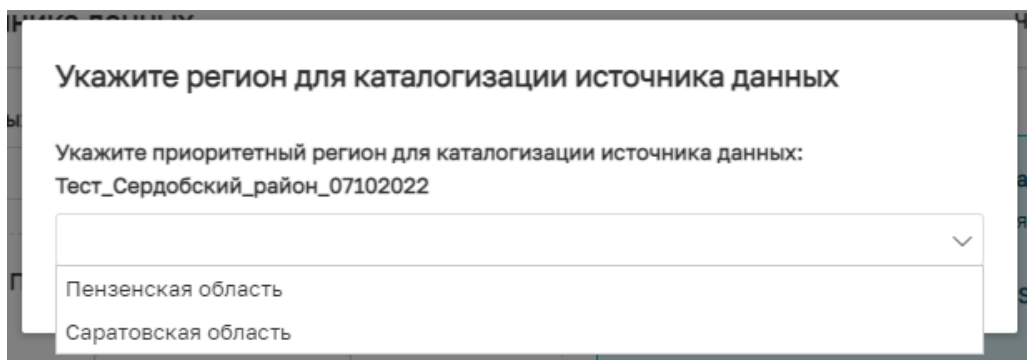


Рисунок 50 – Выбор приоритетного региона

После загрузки файла(ов) в интерфейсе формы создания источника данных отобразится сообщение об успешной загрузке и система предложит перейти к просмотру информации о созданном источнике (рисунок 51).

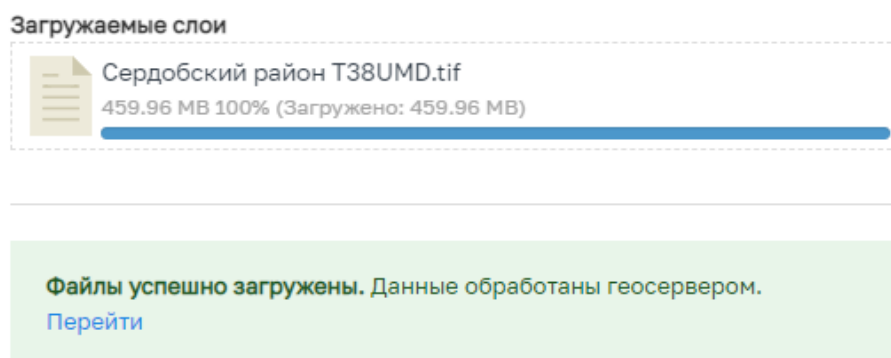


Рисунок 51 – Сообщение об успешной загрузке файла

Редактировать данные созданного источника данных не является возможным. Имеется возможность изменить только наименование созданного источника данных.

Чтобы удалить источник данных, требуется выбрать источник данных и нажать на иконку «Корзина». После этого источник данных будет удален из системы без возможности его восстановления.

4.4 Модуль «Проектирование процессов»

При входе в модуль «Проектирование процессов» открывается раздел «Проектирование процессов», в котором пользователю необходимо пройти процедуру авторизации для входа в систему «Camunda» (рисунок 52).

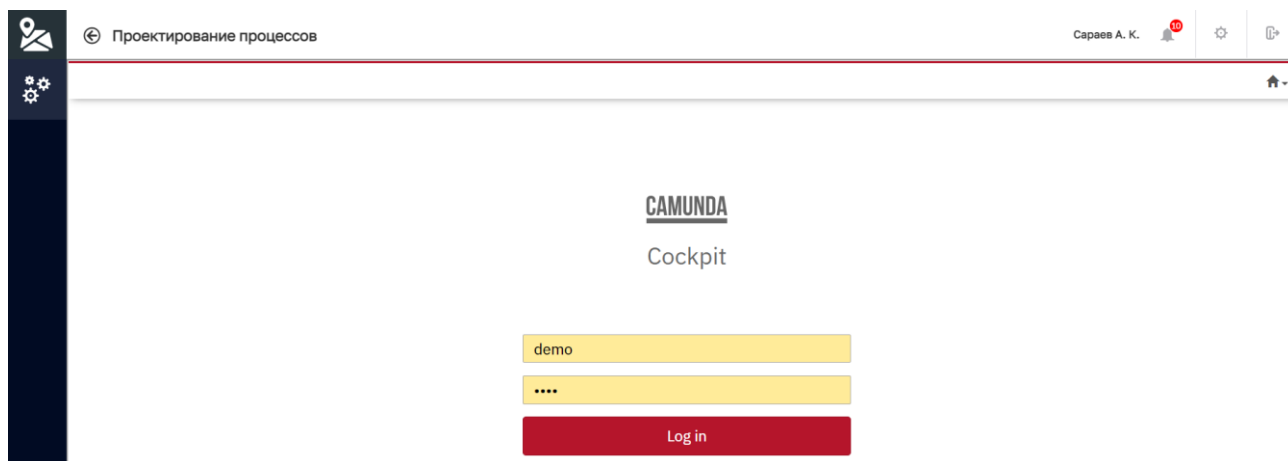


Рисунок 52 – Вход в систему «Camunda»

После успешного входа в систему «Camunda» отобразится интерфейс «Camunda Cockpit», в котором происходит работа с процессами системы (рисунок 53).

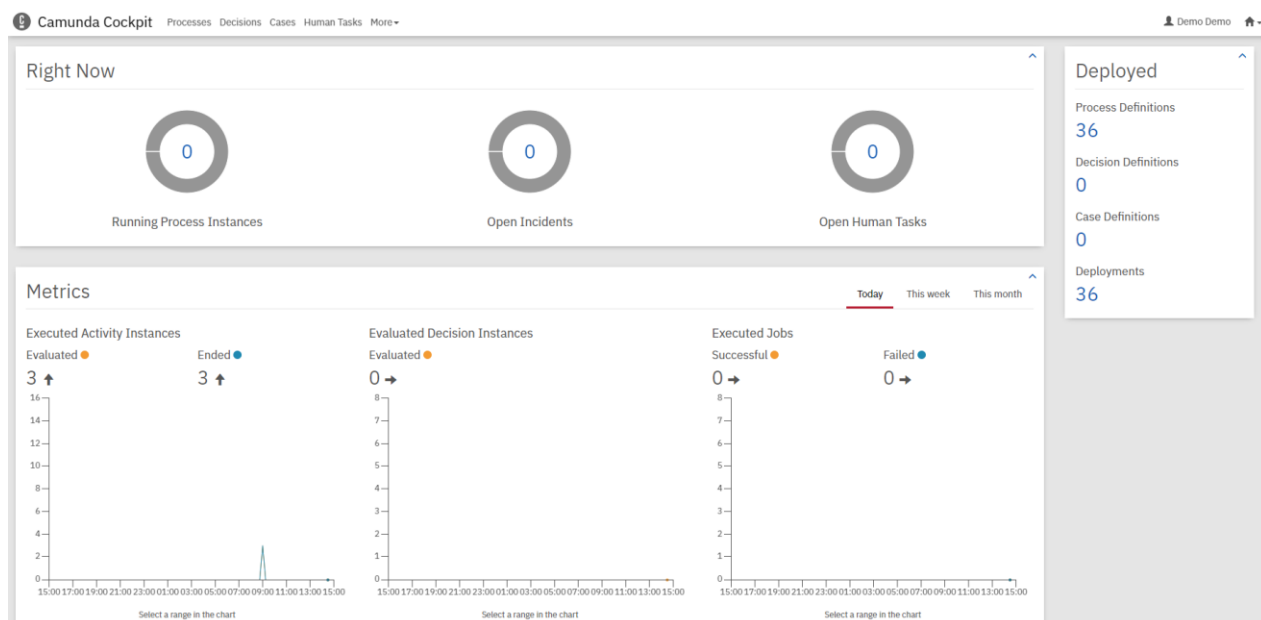


Рисунок 53 – Панель «Camunda Cockpit»

Количество работающих процессов на данный момент времени отображается в блоке «Right Now»:

- «Running Process Instances» – отображает количество запущенных экземпляров процессов,
- «Open Incidents» – отображает количество открытых инцидентов (происшествий), связанных с процессами,
- «Open Human Tasks» – отображает количество открытых задач.

В блоке «Metrics» приведены графики, отражающие состояние задач на определенный момент времени.

Блок «Deployed» содержит информацию о количестве процессов, прецедентов и т.д.

Работа с процессом обработки данных

Процесс обработки фотограмметрических данных зависит от конфигурации, определенной на этапе его создания. Конфигурация – это один из конвейеров обработки фотограмметрических данных, предустановленный и настроенный в системе, обеспечивающий выполнение функций исполнения бизнес-процессов на базе импортированных BPMN-схем.

Для просмотра информации об этапах, которые выполняются при выполнении процесса обработки данных необходимо перейти на вкладку «Processes» и найти необходимую конфигурацию в списке (рисунок 54).

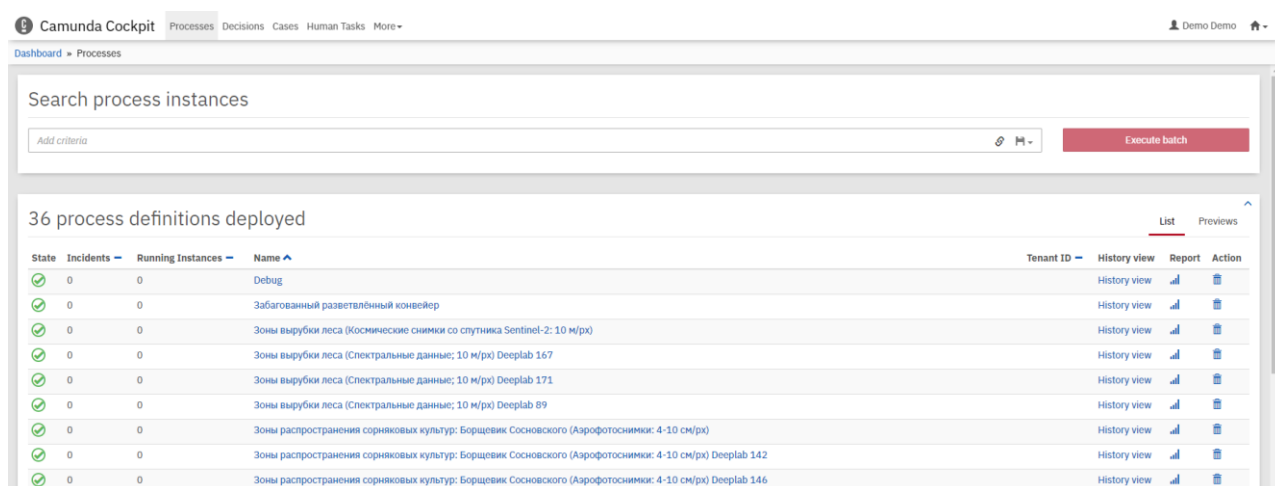


Рисунок 54 – Вкладка «Processes», отображение списка процессов Платформы

Для поиска конфигурации по списку можно воспользоваться поисковой строкой, выбрав критерий из выпадающего списка и введя значение для него.

Нажав на наименование конфигурации, отобразится окно, содержащее информацию об этапах, которые выполняются при выполнении процесса обработки фотограмметрических данных (рисунок 55).

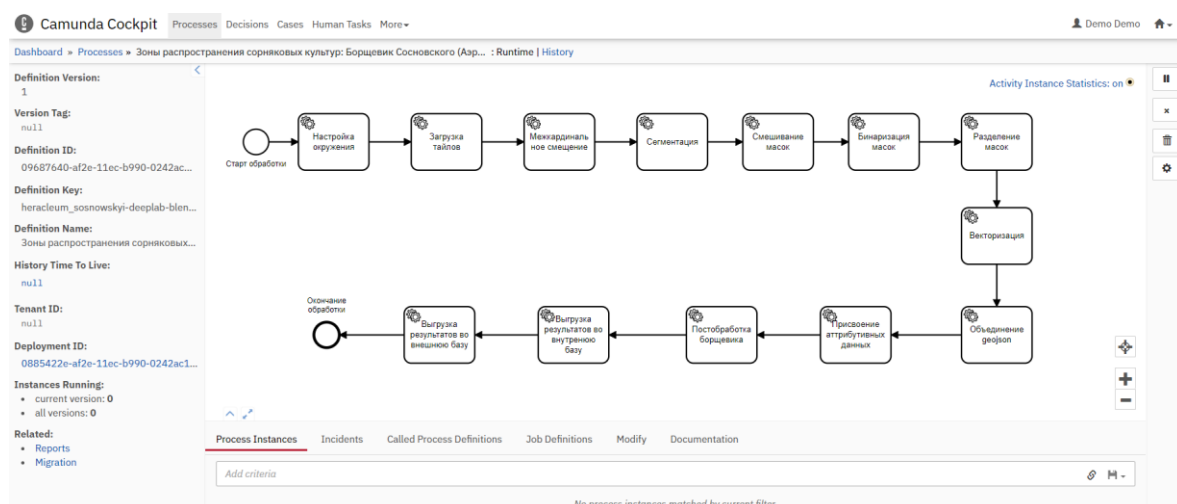


Рисунок 55 – Окно с информацией о выбранной конфигурации

В данном окне отображается схема в нотации BPMN v.2.0, содержащая этапы интеллектуальной обработки. Список всех этапов и их функционал приведен в таблице ниже

Таблица 2 – Этапы интеллектуальной обработки

Наименование	Функционал
Настройка окружения	Настраивает рабочее окружение в горячем хранилище (создает необходимые директории, подгружает модели машинного обучения и т.п.)
Загрузка тайлов	Скачивает тайлы с переданного источника (WMS, WMTS или TMS сервер)
Межкардинальное смещение	Первая стадия алгоритма фильтрации объектов методом межкардинального смещения. Из тайла и прилегающих к нему тайлов формируется мозаика и из нее вырезается 4 смещенных на 25% по сторонам света (С-З, Ю-З, Ю-В, С-В) тайла, которые в последствии будут использоваться в интеллектуальной обработке.

Продолжение таблицы 2

Наименование	Функционал
Сегментация	Алгоритм, используя модели машинного обучения, ищет объекты на снимках и возвращает маски объектов и json-файлы с доп. информацией
Межкардинальная фильтрация	Вторая стадия алгоритма фильтрации объектов методом межкардинального смещения. Алгоритм находит маски объекта, распознанного на 4 смещенных тайлах и отбирает версию объекта, расположенную ближе к центру.
Векторизация	Алгоритм векторизует маски объектов в вектор формата geojson. На каждую маску генерируется отдельный geojson.
Обратное межкардинальное смещение	Третья стадия алгоритма фильтрации объектов методом межкардинального смещения. Алгоритм смещает векторизованные объекты в обратную сторону.
Объединение geojson	Алгоритм объединяет отдельные вектора объектов в один результирующий файл
Выгрузка результатов во внутреннюю базу	Алгоритм заливает результирующий geojson в СУБД MongoDB
Выгрузка результатов во внешнюю базу	Алгоритм заливает результирующий geojson в переданную в конвейер СУБД PostgreSQL (Postgis)

Продолжение таблицы 2

Наименование	Функционал
Постобработка	<p>Стадия применяет последовательность различных геопространственных алгоритмов (simplify, dissolve, orthogonalize и др.) для оптимизации полигональности и улучшения визуального результата. Для каждого конвейера следует настраивать стадию отдельно. В Платформе есть следующие реализации постобработок:</p> <ul style="list-style-type: none"> – постобработка сельскохозяйственных земель, – постобработка лесных вырубок, – ортогонализация – постобработка ОКС, – постобработка борщевика, – постобработка земельных участков.

Чтобы активировать/деактивировать сбор статистики по выбранной конфигурации, необходимо нажать на радиокнопку «Activity Instance Statistics».

Чтобы удалить конфигурацию, требуется нажать на иконку «Корзина». Восстановить конфигурацию после удаления является невозможным.

Чтобы остановить выполнение процесса обработки по выбранной конфигурации, требуется нажать на иконку «Пауза», в появившемся окне активировать/деактивировать функции и подтвердить остановку (рисунок 55):

- «Include Instances» – все существующие этапы будут остановлены,
- «Execute» – остановить выполнение этапов немедленно или указать дату их остановки,
- «Immediately» – остановить немедленно,
- «Delayed» – остановить на определенную дату,

– «Schedule at» – ввод даты и времени, когда этапы будут остановлены.

Suspend Process Definition

This process definition will be suspended, so that it will not be possible to start new process instances based on this process definition.

Include Instances ☒

Execute ☐ Immediately
☒ Delayed

Schedule at

Do you really want to suspend this process definition?

[Close](#) **Suspend**

Рисунок 55 – Настройка остановки выбранного процесса

5 Тестирование

5.1 Тестирование качества обучения моделей

Для проведения тестирования качества детектирования с помощью моделей машинного обучения DeepLabV3Plus были выбраны версии Deeplab 100 и Deeplab 148. Оценка будет происходить по следующим пунктам:

- Количество объектов, выявленных в результате процесса обработки фотограмметрических данных;

- Точность определения границ:

- 1) Объектами являются только поля;

- 2) Границы объектов не пересекают другие объекты (дороги, жилые помещения и т.п).

Ниже представлен анализ с\х земель Пензенской области

При использовании модели машинного обучения Deeplab 100 было выявлено 3834 объекта.

При использовании модели машинного обучения Deeplab 148 было выявлено 3920 объектов.

Сравнение расположения и количества выявленных объектов моделей Deeplab 100 и Deeplab 148 представлены на рисунках 56 – 57.

Сравнение точности определения границ представлены на рисунках 58 – 61.

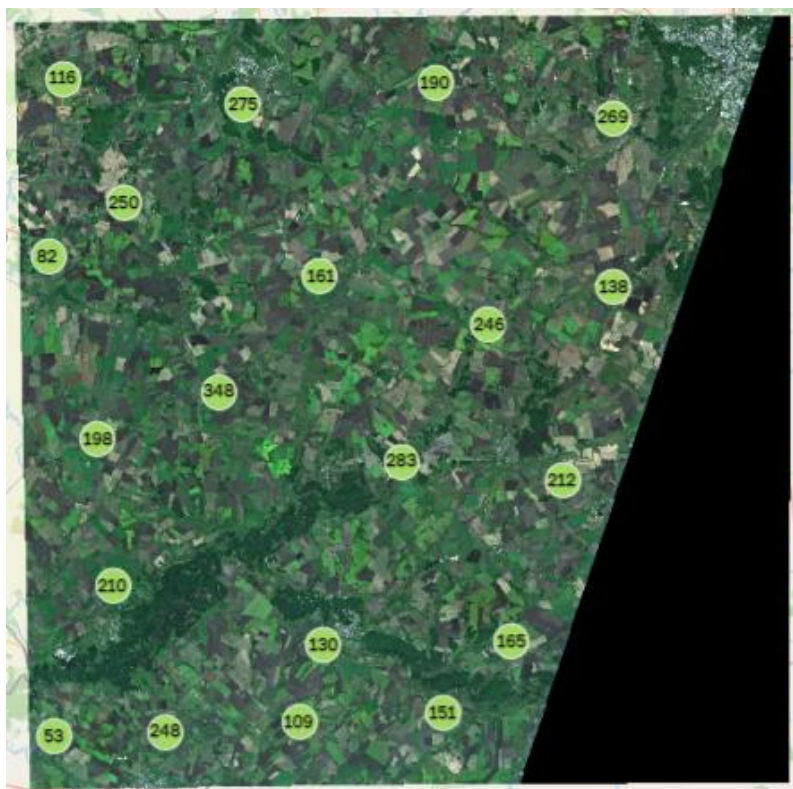


Рисунок 56 – Расположение и количество объектов на фотографическом снимке, расстояние 20 км. Модель Deerlab 100.



Рисунок 57 – Расположение и количество объектов на фотографическом снимке, расстояние 20 км. Модель Deerlab 148



Рисунок 58 – Модель Deerlab 100.



Рисунок 59 – Модель Deerlab 148



Рисунок 60 – Модель Deerlab 100.



Рисунок 61 – Модель Deerlab 148.

Ниже представлен анализ с\х земель Пермского края

При использовании модели машинного обучения Deerlab 100 было выявлено 1285 объектов.

При использовании модели машинного обучения Deerlab 148 было выявлено 1658 объектов.

Сравнение расположения и количества выявленных объектов моделей Deerlab 100 и Deerlab 148 представлены на рисунках 62 – 63.

Сравнение точности определения границ представлены на рисунках 64 – 67.

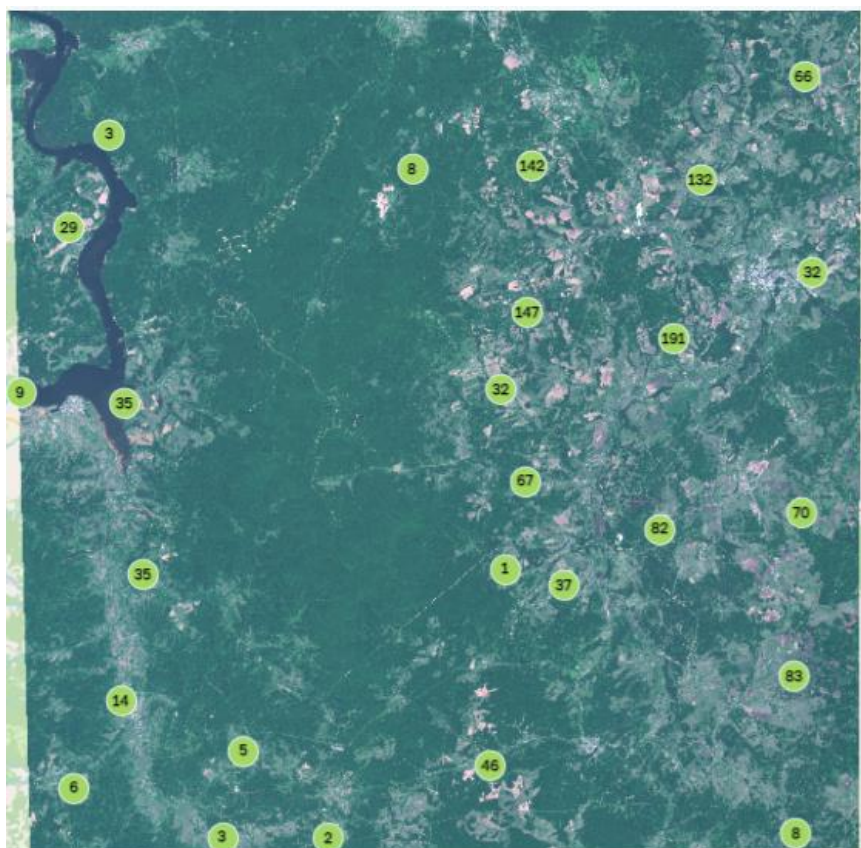


Рисунок 62 – Расположение и количество объектов на фотографическом снимке, расстояние 20 км. Модель Deerlab 100

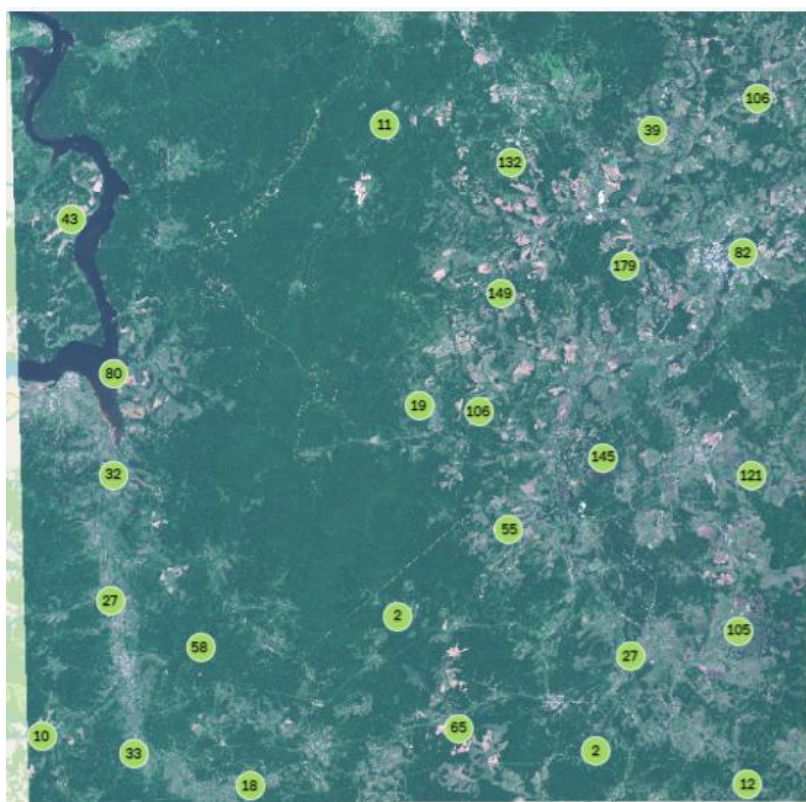


Рисунок 63 – Расположение и количество объектов на фотографическом снимке, расстояние 20 км. Модель Deerlab 148



Рисунок 64 – Модель Deerlab 100.



Рисунок 65 – Модель Deerlab 148.



Рисунок 66 – Модель Deerlab 100.



Рисунок 67 – Модель Deerlab 148.

Ниже представлен анализ с\х земель республики Мордовия. При использовании модели машинного обучения Deeplab 100 было выявлено 4953 объекта.

При использовании модели машинного обучения Deeplab 148 было выявлено 4910 объектов.

Сравнение расположения и количества выявленных объектов моделей Deeplab 100 и Deeplab 148 представлены на рисунках.

Сравнение точности определения границ представлены на рисунках 68 – 73.

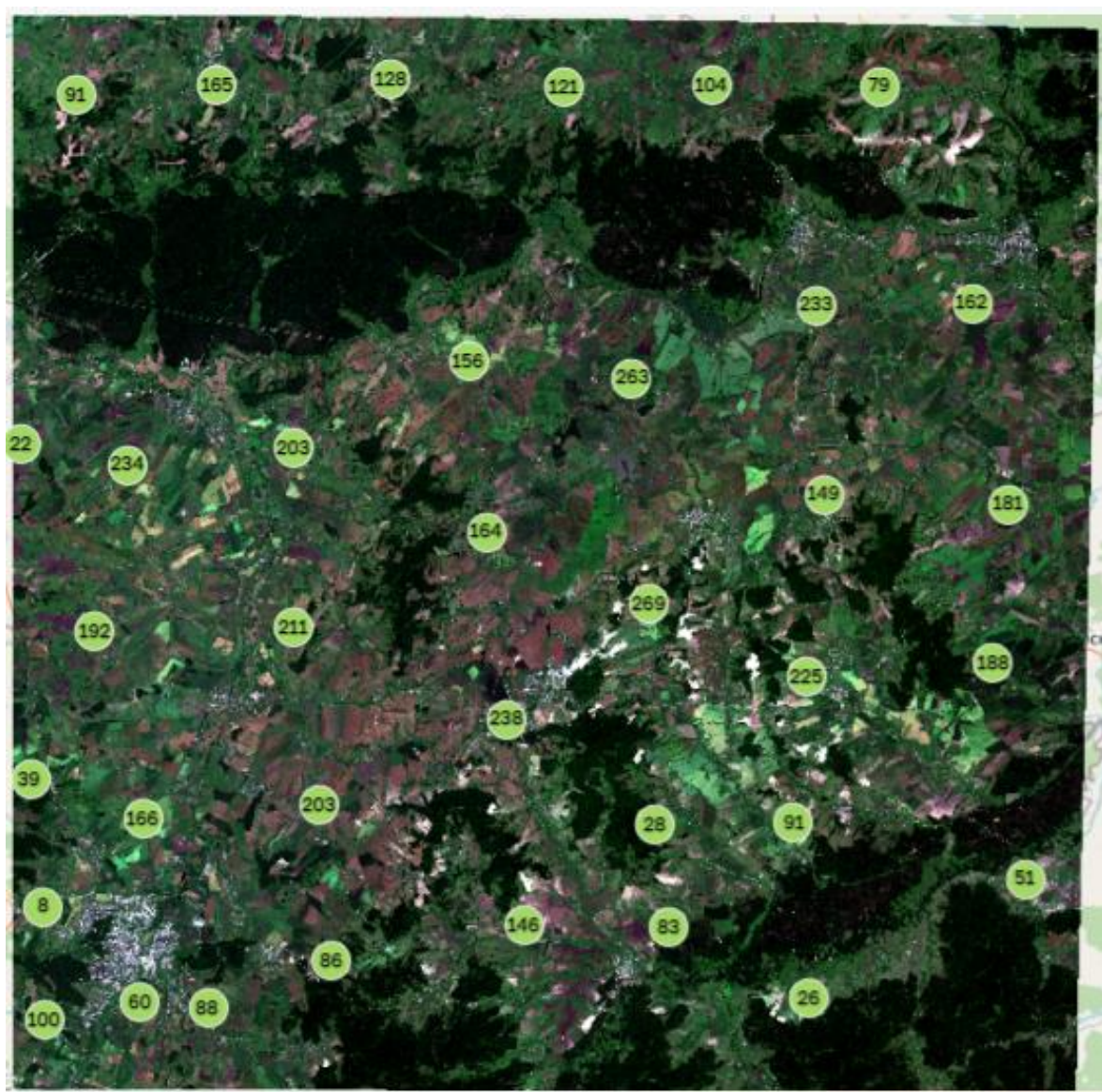


Рисунок 68 – Расположение и количество объектов на фотографическом снимке, расстояние 20 км. Модель Deeplab 100.

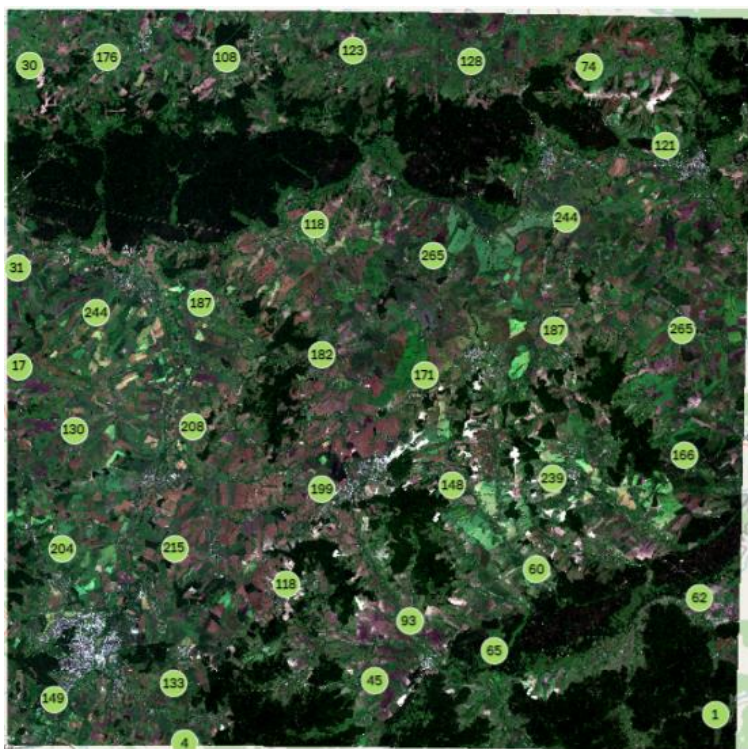


Рисунок 69 – Расположение и количество объектов на фотографическом снимке, расстояние 20 км. Модель Deerlab 148.



Рисунок 70 – Модель Deerlab 100.



Рисунок 71 – Модель Deerlab 148.



Рисунок 72 – Модель Deerlab 100.

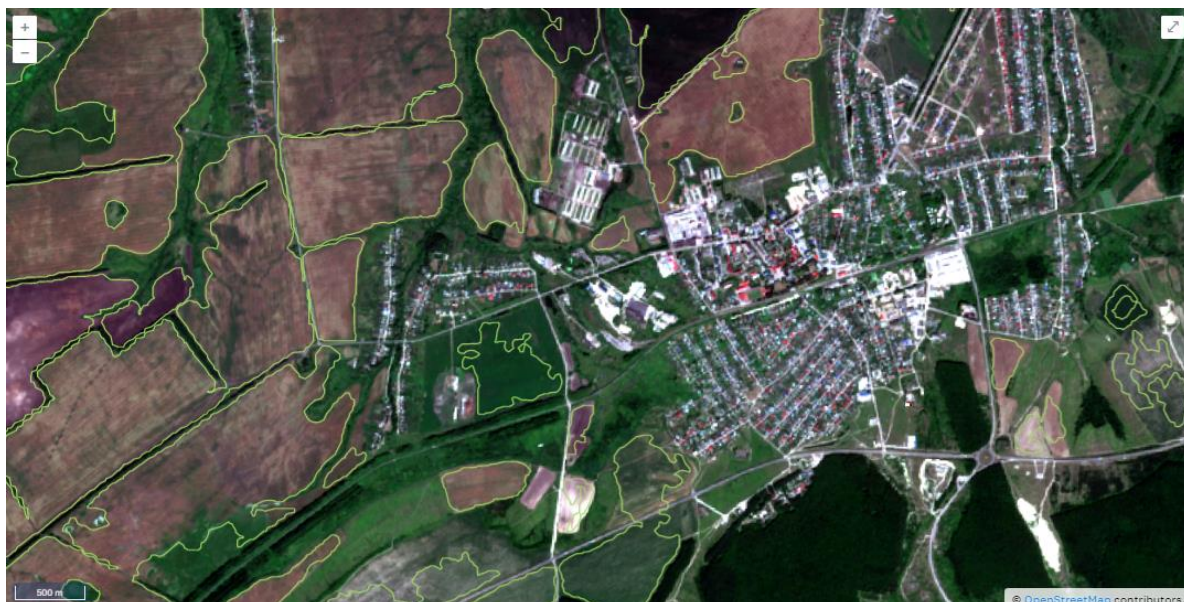


Рисунок 73 – Модель Deeplab 148.

Таким образом, модель машинного обучения версии Deeplab 148 позволяет выявить на фотограмметрическом снимке наиболее точное количество объектов соответствующего типа. Границы выявленных объектов не пересекают другие объекты.

ЗАКЛЮЧЕНИЕ

В рамках выполнения магистерской диссертации была разработана система для анализа и распознавания изображений, в основу работы которой заложена модель машинного обучения DeepLabV3Plus версии Deeplab 148 нейронной сети типа CNN.

За время исследования и проектирования были рассмотрены современные методы, языки программирования и библиотеки для решения задач в области машинного обучения, визуализации различных данных, а также разработке и реализации API, системы управления базами данных и методы виртуализации.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Тропченко А.А., Тропченко А.Ю. Методы вторичной обработки и распознавания изображений. Учебное пособие. – СПб: Университет ИТМО, 2015. – 215 с.
2. Годунов А. И., Балаян С. Т., Егоров П. С. Сегментация изображений и распознавание объектов на основе технологии сверточных нейронных сетей // Надежность и качество сложных систем. 2021. № 3. С. 62–73. doi:10.21685/2307-4205-2021-3-8.
3. Доррер Г. А., Корюкин М. С. Семантическая сегментация изображений с применением сверточных нейронных сетей // Актуальные проблемы авиации и космонавтики. 2017. Т. 2, № 13. С. 141–143.
4. Горбачев В. А., Криворотов И. А., Маркелов А. О., Котлярова Е. В. Семантическая сегментация спутниковых снимков аэропортов с помощью сверточных нейронных сетей М. : Компьютерная оптика. 2020. Вып. 44 (4).
5. Visin F., Ciccone M., Romero A., Kastner K., Cho K., Bengio Y., Matteucci M., Courville A., A Recurrent Neural Network-based Model for Semantic Segmentation 2015.
6. U-Net: нейросеть для сегментации изображений [Электронный ресурс]. URL: <https://neurohive.io/ru/vidy-nejrosetej/u-net-image-segmentation/> (дата обращения: 17.12.2023).
7. Badrinarayanan V., Kendall A., Cipolla R., SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation 2015.
8. PSPNet [Электронный ресурс]. URL: <https://github.com/hszhao/PSPNet> (дата обращения: 20.12.2023).
9. DeepLabv3[Электронныйресурс].RL:<https://learnopencv.com/deeplabv3-ultimate-guide/> (датаобращения: 10.02.2024).
10. Yang M., Yu K., Zhang C., Li Z., Yang K., DenseASPP for Semantic Segmentation in Street Scenes – CVPR 2018.

11. Visin F., Ciccone M., Romero A., Kastner K., Cho K., Bengio Y., Matteucci M., Courville A., A Recurrent Neural Network-based Model for Semantic Segmentation 2015.
12. LSTM и GRU [Электронный ресурс]. URL: <https://habr.com/ru/companies/mvideo/articles/780774/> (дата обращения: 10.01.2024).
13. LSTMиGRU[Электронныйресурс].URL: <https://habr.com/ru/companies/mvideo/articles/780774/>(датаобращения:10.01.2024).
14. NLP From Scratch: Translation with a Sequence to Sequence Network andAttentionь[ЭлектронныйресурcURL:https://pytorch.org/tutorials/intermediate/seq2seq_translation_tutorial. (дата обращения: 11.01.2024).
15. Основы генеративно-состязательных сетей. [Электронный ресурс]. URL: <https://habr.com/ru/articles/726254/> (дата обращения: 11.01.2024).
16. BidirectionalLSTMinNLP[Электронный ресурс].URL:<https://www.geeksforgeeks.org/bidirectional-lstm-in-nlp/>(дата обращения: 13.01.2024).
17. A guide to Semantic Segmentation. [Электронный ресурс]. URL: <https://nanonets.com/blog/semantic-image-segmentation-2020> (дата обращения: 10.01.2024).
18. Глубокое погружение в оптимизацию [Электронный ресурс]. URL: <https://www.yourtodo.ru/posts/glubokoe-pogruzhenie-v-optimizatsiyu-ot-gradientnogo-spuska-do-adam/> (дата обращения: 15.01.2024).
19. Krizhevsky, I. Sutskever, G. E. Hinton., Semantic Segmentation of Satellite Images for Land Cover Classification. - Advances in Neural Information Processing Systems 2012.
20. M. K. Chaudhary, P. K. Sharma., Satellite Image Segmentation Using Deep Learning Techniques - IEEE Geoscience and Remote Sensing Letters

21. DeepGlobe Land Cover Classification Dataset [Электронный ресурс]. URL: <https://gts.ai/dataset-download/deepglobe-land-cover-classification-dataset-ai-data-collection/> (дата обращения: 12.05.2023).

22. Land-Cover Classification with High-Resolution Remote Sensing Images Using Transferable Deep Models [Электронный ресурс]. URL: <https://x-ytong.github.io/project/GID.html> (дата обращения: 12.05.2023).

23. Библиотека глубокого обучения Tensorflow [Электронный ресурс]. URL: <https://habr.com/ru/companies/ods/articles/324898/> (дата обращения: 06.04.2023).

24. Официальный сайт TensorFlow | Основы TensorFlow [Электронный ресурс]. URL: <https://www.tensorflow.org/guide/basics> (дата обращения: 08.04.2023).

25. Официальный сайт PyTorch | PYTORCH DESIGN PHILOSOPHY [Электронный ресурс] URL: <https://pytorch.org/docs/stable/community/design.html> (дата обращения: 04.05.2023).

26. Welcome to Flask – Flask Dockumentation [Электронный ресурс]. URL: <https://flask.palletsprojects.com/en/3.0.x/> (дата обращения: 08.03.2024).

27. Metrics for semantic segmentation [Электронный ресурс]. URL: <https://ilmonteux.github.io/2019/05/10/segmentation-metrics.html> (дата обращения: 08.03.2024).

28. Colab GPUs Features & Pricing [Электронный ресурс] URL: <https://mccormickml.com/colab-gpus-features-and-pricing/> (дата обращения: 18.02.2024).

ПРИЛОЖЕНИЕ А

(Обязательное)

Система анализа и распознавания изображений

Листинг программы

```
GEOSTORAGES
Services
WideTMS Frontend
Dockerfile

FROM tiangolo/uvicorn-gunicorn-fastapi:python3.8-slim
RUN apt update
RUN apt install -y git cmake make g++
RUN cd /; git clone --depth 1 --branch v2.0.2
https://github.com/microsoft/mimalloc; cd mimalloc; mkdir build;
cd build; cmake ..; make -j$(nproc); make install
RUN cd /app
COPY . .
RUN pip install -r ./requirements.txt
ENV LD_PRELOAD=/usr/local/lib/mimalloc-2.0/libmimalloc.so
app.py

from fastapi import FastAPI

app = FastAPI(title="WideTMS",
              description="The TMS server with wider
possibilities",
              version=open('VERSION', 'r').read())
config/aiocache.py

from aiocache import Cache

cache = Cache(Cache.MEMORY)
config/celery_app.py

from celery import Celery
from config.environment import CELERY_RESULT_URL,
CELERY_BROKER_URL

app = Celery('widetms.producer', broker=CELERY_BROKER_URL,
```

```

backend=CELERY_RESULT_URL)
app.config_from_object('config.celeryconfig')
config/celeryconfig.py
    import os
task_serializer = 'pickle'
result_serializer = 'pickle'
accept_content = ['pickle']
result_expires = '30'
task_routes = {'widetms.worker.tile': {'queue':
os.environ['QUEUE']}}
task_default_queue = os.environ['QUEUE']
config/db.py
    import databases
import sqlalchemy
from config.environment import DB_USER, DB_PWD, DB_HOST, DB_NAME,
DB_PORT
DB_URL =
f"postgresql://{DB_USER}:{DB_PWD}@{DB_HOST}:{DB_PORT}/{DB_NAME}"
database = databases.Database(DB_URL)
metadata = sqlalchemy.MetaData()
engine = sqlalchemy.create_engine(DB_URL)
config/environment.py
    import os
DB_NAME = os.environ['DB_NAME']
DB_USER = os.environ['DB_USER']
DB_PWD = os.environ['DB_PWD']
DB_HOST = os.environ['DB_HOST']
DB_PORT = os.environ['DB_PORT']
CELERY_BROKER_URL = os.environ['CELERY_BROKER_URL']
CELERY_RESULT_URL = os.environ['CELERY_RESULT_URL']
main.py
    from sqlalchemy import text
from config.db import metadata, engine
import config.db
from app import app
from routers.tms import router as tms_router

```

```

from routers.config import router as config_router
from routers.raster import router as raster_router
from routers.alias import router as alias_router
from routers.channel import router as channel_router
from routers.rs_device import router as rs_device_router
from routers.processing_type import router as pt_router
@app.on_event("startup")
async def startup() -> None:
    database_ = config.db.database
    if not database_.is_connected:
        await database_.connect()
@app.on_event("shutdown")
async def shutdown() -> None:
    database_ = config.db.database
    if database_.is_connected:
        await database_.disconnect()
app.include_router(tms_router)
app.include_router(raster_router)
app.include_router(config_router)
app.include_router(alias_router)
app.include_router(channel_router)
app.include_router(rs_device_router)
app.include_router(pt_router)
with engine.connect() as conn:
    rs_devices = text(open("../database/rs_devices.sql").read())
    processing_types =
text(open("../database/processing_types.sql").read())
    aliases = text(open("../database/aliases.sql").read())
    channels = text(open("../database/channels.sql").read())
    rasters = text(open("../database/rasters.sql").read())
    conn.execute(rs_devices)
    conn.execute(processing_types)
    conn.execute(aliases)
    conn.execute(channels)
    conn.execute(rasters)
models/alias.py

```

```

import uuid
import ormar
from ormar import UUID
from models.base import BaseMeta
class Alias(ormar.Model):
    class Meta(BaseMeta):
        tablename = 'aliases'
        id: uuid.UUID = UUID(primary_key=True, default=uuid.uuid4)
        alias: str = ormar.String(max_length=255)
        description: str = ormar.String(max_length=1024)
models/base.py
import ormar
from config.db import metadata
from config.db import database
class BaseMeta(ormar.ModelMeta):
    metadata = metadata
    database = database
models/channel.py
import uuid
from typing import Optional
import ormar
from ormar import UUID
from models.base import BaseMeta
from models.rs_device import RemoteSensingDevice
class Channel(ormar.Model):
    class Meta(BaseMeta):
        tablename = 'channels'
        id: uuid.UUID = UUID(primary_key=True, default=uuid.uuid4)
        channel: str = ormar.String(max_length=255)
        rs_device_id: Optional[RemoteSensingDevice] =
ormar.ForeignKey(RemoteSensingDevice, skip_reverse=False)
        spatial_res: int = ormar.Integer(nullable=True)
        unit: str = ormar.String(max_length=255, nullable=True)
        description: str = ormar.String(max_length=1024)
        special: bool = ormar.Boolean(nullable=False)
        formula: str = ormar.Text()

```

```

models/processing_type.py
    import uuid
import ormar
from ormar import UUID
from models.base import BaseMeta
class ProcessingType(ormar.Model):
    class Meta(BaseMeta):
        tablename = 'processing_types'
        id: uuid.UUID = UUID(primary_key=True, default=uuid.uuid4)
        processing_type: str = ormar.String(max_length=255)
        description: str = ormar.String(max_length=1024)
models/pydantic/update/alias.py
    import uuid
from typing import Optional
from uuid import UUID
from pydantic import BaseModel, Field
class CreateAlias(BaseModel):
    id: UUID = Field(default_factory=uuid.uuid4)
    alias: str
    description: Optional[str]
models/pydantic/update/channel.py
    import uuid
from typing import Optional
from uuid import UUID
from pydantic import BaseModel, Field
class CreateChannel(BaseModel):
    id: UUID = Field(default_factory=uuid.uuid4)
    channel: Optional[str]
    rs_device_id: UUID
    spatial_res: Optional[int]
    unit: Optional[str]
    description: Optional[str]
    special: bool
    formula: Optional[str]
models/pydantic/update/processing_type.py
    import uuid

```

```

from typing import Optional
from uuid import UUID
from pydantic import BaseModel, Field
class CreateProcessingType(BaseModel):
    id: UUID = Field(default_factory=uuid.uuid4)
    processing_type: str
    description: Optional[str]
models/pydantic/update/raster.py
    import datetime
import uuid
from typing import Optional
from uuid import UUID
from pydantic import BaseModel, Field
class CreateRaster(BaseModel):
    id: UUID = Field(default_factory=uuid.uuid4)
    alias: UUID
    channel_id: Optional[UUID]
    sensing_time: datetime.datetime
    processing_type: UUID
    filepath: str
models/pydantic/update/rs_device.py
    import uuid
from typing import Optional
from uuid import UUID
from pydantic import BaseModel, Field
class CreateRsDevice(BaseModel):
    id: UUID = Field(default_factory=uuid.uuid4)
    rs_device: str
    description: Optional[str]
models/raster.py
    import datetime
import os
import uuid
from typing import Optional
import ormar
from ormar import UUID

```

```

from models.alias import Alias
from models.base import BaseMeta
from models.channel import Channel
from models.processing_type import ProcessingType
class Raster(ormar.Model):
    class Meta(BaseMeta):
        tablename = 'rasters'
        id: uuid.UUID = UUID(primary_key=True, default=uuid.uuid4)
        alias_id: Optional[Alias] = ormar.ForeignKey(Alias,
skip_reverse=True)
        channel_id: Optional[Channel] = ormar.ForeignKey(Channel,
skip_reverse=True)
        sensing_time: datetime.datetime =
ormar.DateTime(nullable=False)
        processing_type_id: Optional[ProcessingType] =
ormar.ForeignKey(ProcessingType, skip_reverse=True)
        filepath: str = ormar.Text()
models/rs_device.py
    import uuid
import ormar
from ormar import UUID
from models.base import BaseMeta
class RemoteSensingDevice(ormar.Model):
    class Meta(BaseMeta):
        tablename = 'rs_devices'
        id: uuid.UUID = UUID(primary_key=True, default=uuid.uuid4)
        rs_device: str = ormar.String(max_length=255, nullable=False)
        description: str = ormar.String(max_length=1024,
nullable=True)
routers/alias.py
    from typing import Optional
from fastapi_crudrouter import OrmarCRUDRouter
from models.alias import Alias
from models.pydantic.update.alias import CreateAlias
router = OrmarCRUDRouter(
    schema=Alias,

```

ПРИЛОЖЕНИЕ Б

(Обязательное)

Структура модели DeepLabV3Plus

Файл model

```
# -*- coding: utf-8 -*-
""" Deeplabv3+ model for Keras.
This model is based on TF repo:
https://github.com/tensorflow/models/tree/master/research/deeplab
On Pascal VOC, original model gets to 84.56% mIOU
MobileNetv2 backbone is based on this repo:
https://github.com/JonathanCMitchell/mobilenet\_v2\_keras
# Reference
- [Encoder-Decoder with Atrous Separable Convolution
  for Semantic Image
  Segmentation] (https://arxiv.org/pdf/1802.02611.pdf)
- [Xception: Deep Learning with Depthwise Separable
  Convolutions]
  (https://arxiv.org/abs/1610.02357)
- [Inverted Residuals and Linear Bottlenecks: Mobile Networks
  for
  Classification, Detection and
  Segmentation] (https://arxiv.org/abs/1801.04381)
"""
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function
import tensorflow as tf
from tensorflow.python.keras.models import Model
from tensorflow.python.keras import layers
from tensorflow.python.keras.layers import Input
from tensorflow.python.keras.layers import Reshape
from tensorflow.python.keras.layers import Activation
from tensorflow.python.keras.layers import Concatenate
from tensorflow.python.keras.layers import Add
```

```

from tensorflow.python.keras.layers import Dropout
from tensorflow.python.keras.layers import BatchNormalization
from tensorflow.python.keras.layers import Conv2D
from tensorflow.python.keras.layers import DepthwiseConv2D
from tensorflow.python.keras.layers import ZeroPadding2D
from tensorflow.python.keras.layers import
GlobalAveragePooling2D
    from tensorflow.python.keras.utils.layer_utils import
get_source_inputs
    from tensorflow.python.keras.utils.data_utils import get_file
    from tensorflow.python.keras import backend as K
    from tensorflow.python.keras.applications.imagenet_utils
import preprocess_input
    WEIGHTS_PATH_X = "https://github.com/bonlime/keras-deeplab-v3-
plus/releases/download/1.1/deeplabv3_xception_tf_dim_ordering_tf_k
ernels.h5"
    WEIGHTS_PATH_MOBILE = "https://github.com/bonlime/keras-
deeplab-v3-
plus/releases/download/1.1/deeplabv3_mobilenetv2_tf_dim_ordering_t
f_kernels.h5"
    WEIGHTS_PATH_X_CS = "https://github.com/bonlime/keras-deeplab-
v3-
plus/releases/download/1.2/deeplabv3_xception_tf_dim_ordering_tf_k
ernels_cityscapes.h5"
    WEIGHTS_PATH_MOBILE_CS = "https://github.com/bonlime/keras-
deeplab-v3-
plus/releases/download/1.2/deeplabv3_mobilenetv2_tf_dim_ordering_t
f_kernels_cityscapes.h5"

def SepConv_BN(x, filters, prefix, stride=1, kernel_size=3,
rate=1, depth_activation=False, epsilon=1e-3):
    """ SepConv with BN between depthwise & pointwise.
    Optionally add activation after BN
    Implements right "same" padding for even kernel sizes
    Args:

```

```

        x: input tensor
        filters: num of filters in pointwise convolution
        prefix: prefix before name
        stride: stride at depthwise conv
        kernel_size: kernel size for depthwise convolution
        rate: atrous rate for depthwise convolution
        depth_activation: flag to use activation between
depthwise & poinwise convs
        epsilon: epsilon to use in BN layer
        if stride == 1:
            depth_padding = 'same'
        else:
            kernel_size_effective = kernel_size + (kernel_size -
1) * (rate - 1)
            pad_total = kernel_size_effective - 1
            pad_beg = pad_total // 2
            pad_end = pad_total - pad_beg
            x = ZeroPadding2D((pad_beg, pad_end))(x)
            depth_padding = 'valid'
        if not depth_activation:
            x = Activation(tf.nn.relu)(x)
        x = DepthwiseConv2D((kernel_size, kernel_size),
strides=(stride, stride), dilation_rate=(rate, rate),
                        padding=depth_padding, use_bias=False,
name=prefix + '_depthwise')(x)
        x = BatchNormalization(name=prefix + '_depthwise_BN',
epsilon=epsilon)(x)
        if depth_activation:
            x = Activation(tf.nn.relu)(x)
        x = Conv2D(filters, (1, 1), padding='same',
                        use_bias=False, name=prefix + '_pointwise')(x)
        x = BatchNormalization(name=prefix + '_pointwise_BN',
epsilon=epsilon)(x)
        if depth_activation:
            x = Activation(tf.nn.relu)(x)
    return x

```

```

def _conv2d_same(x, filters, prefix, stride=1, kernel_size=3,
rate=1):
    """Implements right 'same' padding for even kernel sizes
    Without this there is a 1 pixel drift when stride = 2
    Args:
        x: input tensor
        filters: num of filters in pointwise convolution
        prefix: prefix before name
        stride: stride at depthwise conv
        kernel_size: kernel size for depthwise convolution
        rate: atrous rate for depthwise convolution
    if stride == 1:
        return Conv2D(filters,
                        (kernel_size, kernel_size),
                        strides=(stride, stride),
                        padding='same', use_bias=False,
                        dilation_rate=(rate, rate),
                        name=prefix)(x)
    else:
        kernel_size_effective = kernel_size + (kernel_size -
1) * (rate - 1)
        pad_total = kernel_size_effective - 1
        pad_beg = pad_total // 2
        pad_end = pad_total - pad_beg
        x = ZeroPadding2D((pad_beg, pad_end))(x)
        return Conv2D(filters,
                        (kernel_size, kernel_size),
                        strides=(stride, stride),
                        padding='valid', use_bias=False,
                        dilation_rate=(rate, rate),
                        name=prefix)(x)

def _xception_block(inputs, depth_list, prefix,
skip_connection_type, stride,
rate=1, depth_activation=False,
return_skip=False):
    """ Basic building block of modified Xception network

```

```

    Args:
        inputs: input tensor
        depth_list: number of filters in each SepConv
layer. len(depth_list) == 3
        prefix: prefix before name
        skip_connection_type: one of {'conv','sum','none'}
        stride: stride at last depthwise conv
        rate: atrous rate for depthwise convolution
        depth_activation: flag to use activation between
depthwise & pointwise convs
        return_skip: flag to return additional tensor
after 2 SepConvs for decoder
    """
    residual = inputs
    for i in range(3):
        residual = SepConv_BN(residual,
                               depth_list[i],
                               prefix +
'_separable_conv{}'.format(i + 1),
                               stride=stride if i == 2 else 1,
                               rate=rate,

depth_activation=depth_activation)
        if i == 1:
            skip = residual
        if skip_connection_type == 'conv':
            shortcut = _conv2d_same(inputs, depth_list[-1], prefix
+ '_shortcut',
                                   kernel_size=1,
                                   stride=stride)
            shortcut = BatchNormalization(name=prefix +
'_shortcut_BN')(shortcut)
            outputs = layers.add([residual, shortcut])
        elif skip_connection_type == 'sum':
            outputs = layers.add([residual, inputs])
        elif skip_connection_type == 'none':

```

```

        outputs = residual
    if return_skip:
        return outputs, skip
    else:
        return outputs

def _make_divisible(v, divisor, min_value=None):
    if min_value is None:
        min_value = divisor
    new_v = max(min_value, int(v + divisor / 2) // divisor *
divisor)
    # Make sure that round down does not go down by more than
10%.
    if new_v < 0.9 * v:
        new_v += divisor
    return new_v

def _inverted_res_block(inputs, expansion, stride, alpha,
filters, block_id, skip_connection, rate=1):
    in_channels = inputs.shape[-1].value #
inputs._keras_shape[-1]
    pointwise_conv_filters = int(filters * alpha)
    pointwise_filters =
_make_divisible(pointwise_conv_filters, 8)
    x = inputs
    prefix = 'expanded_conv_{}_'.format(block_id)
    if block_id:
        # Expand
        x = Conv2D(expansion * in_channels, kernel_size=1,
padding='same',
                    use_bias=False, activation=None,
                    name=prefix + 'expand')(x)
        x = BatchNormalization(epsilon=1e-3, momentum=0.999,
                                name=prefix + 'expand_BN')(x)
        x = Activation(tf.nn.relu6, name=prefix +
'expand_relu')(x)
    else:
        prefix = 'expanded_conv_'

```

```

        # Depthwise
        x = DepthwiseConv2D(kernel_size=3, strides=stride,
activation=None,
                                use_bias=False, padding='same',
dilation_rate=(rate, rate),
                                name=prefix + 'depthwise')(x)
        x = BatchNormalization(epsilon=1e-3, momentum=0.999,
                                name=prefix + 'depthwise_BN')(x)
        x = Activation(tf.nn.relu6, name=prefix +
'depthwise_relu')(x)
        # Project
        x = Conv2D(pointwise_filters,
                                kernel_size=1, padding='same', use_bias=False,
activation=None,
                                name=prefix + 'project')(x)
        x = BatchNormalization(epsilon=1e-3, momentum=0.999,
                                name=prefix + 'project_BN')(x)
        if skip_connection:
            return Add(name=prefix + 'add')([inputs, x])
        # if in_channels == pointwise_filters and stride == 1:
        #     return Add(name='res_connect_' +
str(block_id))([inputs, x])
        return x

def Deeplabv3(weights='pascal_voc', input_tensor=None,
input_shape=(512, 512, 3), classes=21, backbone='mobilenetv2',
            OS=16, alpha=1., activation=None):
    """ Instantiates the Deeplabv3+ architecture
    Optionally loads weights pre-trained
    on PASCAL VOC or Cityscapes. This model is available for
    TensorFlow only.
    # Arguments
        weights: one of 'pascal_voc' (pre-trained on pascal
voc),
                'cityscapes' (pre-trained on cityscape) or None
(random initialization)
        input_tensor: optional Keras tensor (i.e. output of

```

```

`layers.Input()`)
    to use as image input for the model.
    input_shape: shape of input image. format HxWxC
    PASCAL VOC model was trained on (512,512,3)
images. None is allowed as shape/width
    classes: number of desired classes. PASCAL VOC has 21
classes, Cityscapes has 19 classes.
    If number of classes not aligned with the weights
used, last layer is initialized randomly
    backbone: backbone to use. one of
{'xception','mobilenetv2'}
    activation: optional activation to add to the top of
the network.
        One of 'softmax', 'sigmoid' or None
    OS: determines input_shape/feature_extractor_output
ratio. One of {8,16}.
        Used only for xception backbone.
    alpha: controls the width of the MobileNetV2 network.
This is known as the
        width multiplier in the MobileNetV2 paper.
        - If `alpha` < 1.0, proportionally decreases
the number
            of filters in each layer.
        - If `alpha` > 1.0, proportionally increases
the number
            of filters in each layer.
        - If `alpha` = 1, default number of filters
from the paper
            are used at each layer.
        Used only for mobilenetv2 backbone. Pretrained is
only available for alpha=1.
    # Returns
        A Keras model instance.
    # Raises
        RuntimeError: If attempting to run this model with a
            backend that does not support separable

```

```

convolutions.
        ValueError: in case of invalid argument for `weights`
or `backbone`
        if not (weights in {'pascal_voc', 'cityscapes', None}):
            raise ValueError('The `weights` argument should be
either '
                                '`None` (random initialization),
`pascal_voc`, or `cityscapes` '
                                '(pre-trained on PASCAL VOC)')
        if not (backbone in {'xception', 'mobilenetv2'}):
            raise ValueError('The `backbone` argument should be
either '
                                '`xception` or `mobilenetv2` ')
        if input_tensor is None:
            img_input = Input(shape=input_shape)
        else:
            img_input = input_tensor
        if backbone == 'xception':
            if OS == 8:
                entry_block3_stride = 1
                middle_block_rate = 2 # ! Not mentioned in paper,
but required
                exit_block_rates = (2, 4)
                atrous_rates = (12, 24, 36)
            else:
                entry_block3_stride = 2
                middle_block_rate = 1
                exit_block_rates = (1, 2)
                atrous_rates = (6, 12, 18)
            x = Conv2D(32, (3, 3), strides=(2, 2),
                        name='entry_flow_conv1_1', use_bias=False,
padding='same')(img_input)
            x =
BatchNormalization(name='entry_flow_conv1_1_BN')(x)
            x = Activation(tf.nn.relu)(x)
            x = _conv2d_same(x, 64, 'entry_flow_conv1_2',

```

```

kernel_size=3, stride=1)
        x =
BatchNormalization(name='entry_flow_conv1_2_BN')(x)
        x = Activation(tf.nn.relu)(x)
        x = _xception_block(x, [128, 128, 128],
'entry_flow_block1',
                                skip_connection_type='conv',
stride=2,
                                depth_activation=False)
        x, skip1 = _xception_block(x, [256, 256, 256],
'entry_flow_block2',

skip_connection_type='conv', stride=2,
                                activation=False, return_skip=True)
        x = _xception_block(x, [728, 728, 728],
'entry_flow_block3',
                                skip_connection_type='conv',
stride=entry_block3_stride,
                                depth_activation=False)
        for i in range(16):
            x = _xception_block(x, [728, 728, 728],
'middle_flow_unit_{}'.format(i + 1),
                                skip_connection_type='sum', stride=1, rate=middle_block_rate,
                                depth_activation=False)

            x = _xception_block(x, [728, 1024, 1024], 'exit_flow_block1',
                                skip_connection_type='conv', stride=1,
rate=exit_block_rates[0],
                                depth_activation=False)
            x = _xception_block(x, [1536, 1536, 2048], 'exit_flow_block2',
                                skip_connection_type='none', stride=1,
rate=exit_block_rates[1],
                                depth_activation=True)
        else:
            OS = 8
            first_block_filters = _make_divisible(32 * alpha, 8)

```

```

x = Conv2D(first_block_filters,
kernel_size=3,
strides=(2, 2), padding='same', use_bias=False,
name='Conv' if input_shape[2] == 3 else 'Conv_')(img_input)
x = BatchNormalization(
epsilon=1e-3, momentum=0.999, name='Conv_BN')(x)
x = Activation(tf.nn.relu6, name='Conv_Rel6')(x)

x = _inverted_res_block(x, filters=16, alpha=alpha, stride=1,
expansion=1, block_id=0, skip_connection=False)

x = _inverted_res_block(x, filters=24, alpha=alpha, stride=2,
expansion=6, block_id=1, skip_connection=False)
x = _inverted_res_block(x, filters=24, alpha=alpha, stride=1,
expansion=6, block_id=2, skip_connection=True)

x = _inverted_res_block(x, filters=32, alpha=alpha, stride=2,
expansion=6, block_id=3, skip_connection=False)
x = _inverted_res_block(x, filters=32, alpha=alpha, stride=1,
expansion=6, block_id=4, skip_connection=True)
x = _inverted_res_block(x, filters=32, alpha=alpha, stride=1,
expansion=6, block_id=5, skip_connection=True)

# stride in block 6 changed from 2 -> 1, so we need to use
rate = 2
x = _inverted_res_block(x, filters=64, alpha=alpha, stride=1,
# 1!
expansion=6, block_id=6, skip_connection=False)
x = _inverted_res_block(x, filters=64, alpha=alpha, stride=1,
rate=2,
expansion=6, block_id=7, skip_connection=True)
x = _inverted_res_block(x, filters=64, alpha=alpha, stride=1,
rate=2,
expansion=6, block_id=8, skip_connection=True)
x = _inverted_res_block(x, filters=64, alpha=alpha, stride=1,
rate=2,

```

```

        expansion=6, block_id=9, skip_connection=True)
        x = _inverted_res_block(x, filters=96, alpha=alpha,
stride=1, rate=2,
                                expansion=6, block_id=10,
skip_connection=False)
        x = _inverted_res_block(x, filters=96, alpha=alpha,
stride=1, rate=2,
                                expansion=6, block_id=11,
skip_connection=True) x = _inverted_res_block(x, filters=96,
alpha=alpha, stride=1, rate=2,
                                expansion=6, block_id=12,
skip_connection=True)
        x = _inverted_res_block(x, filters=160, alpha=alpha,
stride=1, rate=2, # 1!
                                expansion=6, block_id=13, skip_connection=False)
        x = _inverted_res_block(x, filters=160, alpha=alpha,
stride=1, rate=4,
                                xpansion=6, block_id=14, skip_connection=True)
        x = _inverted_res_block(x, filters=160, alpha=alpha,
stride=1, rate=4,expansion=6, block_id=15, skip_connection=True)
        x = _inverted_res_block(x, filters=320, alpha=alpha,
stride=1,
                                expansion=6, block_id=16,
skip_connection=False)

# end of feature extractor
# branching for Atrous Spatial Pyramid Pooling
# Image Feature branch
shape_before = tf.shape(x)
b4 = GlobalAveragePooling2D()(x)
b4_shape = tf.keras.backend.int_shape(b4)
# from (b_size, channels)->(b_size, 1, 1, channels)
b4 = Reshape((1, 1, b4_shape[1]))(b4)
b4 = Conv2D(256, (1, 1), padding='same',
            use_bias=False, name='image_pooling')(b4)
b4 = BatchNormalization(name='image_pooling_BN',
epsilon=1e-5)(b4)
b4 = Activation(tf.nn.relu)(b4)

```

```

        # upsample. have to use compat because of the option
align_corners
        size_before = tf.keras.backend.int_shape(x)
        b4 = tf.keras.layers.experimental.preprocessing.Resizing(
            *size_before[1:3], interpolation="bilinear"
        )(b4)
        # simple 1x1
        b0 = Conv2D(256, (1, 1), padding='same', use_bias=False,
name='aspp0')(x)
        b0 = BatchNormalization(name='aspp0_BN', epsilon=1e-5)(b0)
        b0 = Activation(tf.nn.relu, name='aspp0_activation')(b0)

        # there are only 2 branches in mobilenetV2. not sure why
        if backbone == 'xception':
            # rate = 6 (12)
            b1 = SepConv_BN(x, 256, 'aspp1',
                            rate=atrous_rates[0],
depth_activation=True, epsilon=1e-5)
            # rate = 12 (24)
            b2 = SepConv_BN(x, 256, 'aspp2',
                            rate=atrous_rates[1],
depth_activation=True, epsilon=1e-5)
            # rate = 18 (36)
            b3 = SepConv_BN(x, 256, 'aspp3',
                            rate=atrous_rates[2],
depth_activation=True, epsilon=1e-5)
            # concatenate ASPP branches & project
            x = Concatenate()([b4, b0, b1, b2, b3])
        else:
            x = Concatenate()([b4, b0])

        x = Conv2D(256, (1, 1), padding='same',
                    use_bias=False, name='concat_projection')(x)
        x = BatchNormalization(name='concat_projection_BN',
epsilon=1e-5)(x)
        x = Activation(tf.nn.relu)(x)

```

```

x = Dropout(0.1)(x)
# DeepLab v.3+ decoder
if backbone == 'xception':
    # Feature projection
    # x4 (x2) block
    skip_size = tf.keras.backend.int_shape(skip1)
    x =
tf.keras.layers.experimental.preprocessing.Resizing(
    *skip_size[1:3], interpolation="bilinear"
)(x)
    dec_skip1 = Conv2D(48, (1, 1), padding='same',
                        use_bias=False,
name='feature_projection0')(skip1)
    dec_skip1 = BatchNormalization(
        name='feature_projection0_BN', epsilon=1e-
5)(dec_skip1)
    dec_skip1 = Activation(tf.nn.relu)(dec_skip1)
    x = Concatenate()([x, dec_skip1])
    x = SepConv_BN(x, 256, 'decoder_conv0',
                    depth_activation=True, epsilon=1e-5)
    x = SepConv_BN(x, 256, 'decoder_conv1',
                    depth_activation=True, epsilon=1e-5)
    # you can use it with arbitrary number of classes
    if (weights == 'pascal_voc' and classes == 21) or (weights
== 'cityscapes' and classes == 19):
        last_layer_name = 'logits_semantic'
    else:
        last_layer_name = 'custom_logits_semantic'
    x = Conv2D(classes, (1, 1), padding='same',
name=last_layer_name)(x)
    size_before3 = tf.keras.backend.int_shape(img_input)
    x = tf.keras.layers.experimental.preprocessing.Resizing(
        *size_before3[1:3], interpolation="bilinear"
    )(x)
    # Ensure that the model takes into account
    # any potential predecessors of `input_tensor`.

```

```

if input_tensor is not None:
    inputs = get_source_inputs(input_tensor)
else:
    inputs = img_input
if activation in {'softmax', 'sigmoid'}:
    x = tf.keras.layers.Activation(activation)(x)
model = Model(inputs, x, name='deeplabv3plus')
# load weights
if weights == 'pascal_voc':
    if backbone == 'xception':
        weights_path =
get_file('deeplabv3_xception_tf_dim_ordering_tf_kernels.h5',
        WEIGHTS_PATH_X,
        cache_subdir='models')
    else:
        weights_path =
get_file('deeplabv3_mobilenetv2_tf_dim_ordering_tf_kernels.h5',
        WEIGHTS_PATH_MOBILE,
        cache_subdir='models')
    model.load_weights(weights_path, by_name=True)
elif weights == 'cityscapes':
    if backbone == 'xception':
        weights_path =
get_file('deeplabv3_xception_tf_dim_ordering_tf_kernels_cityscapes
.h5',
        WEIGHTS_PATH_X_CS,
        cache_subdir='models')
    else:
        weights_path =
get_file('deeplabv3_mobilenetv2_tf_dim_ordering_tf_kernels_citysca
pes.h5',
        WEIGHTS_PATH_MOBILE_CS,
        cache_subdir='models')
    model.load_weights(weights_path, by_name=True)
return model

def preprocess_input(x):
    """Preprocesses a numpy array encoding a batch of images.

```

```

        # Arguments
        x: a 4D numpy array consists of RGB values within [0,
255].

    # Returns
        Input array scaled to [-1.,1.]
    """
    return preprocess_input(x, mode='tf')

Файл load_weights
from __future__ import print_function
import os
import numpy as np
from tqdm import tqdm
from model import Deeplabv3
ODEL_DIR = 'models'
or backbone in ['mobilenetv2', 'xception']:
    print('Instantiating an empty Deeplabv3+ model...')
    model = Deeplabv3(input_shape=(512, 512, 3),
                        classes=21, backbone=backbone,
weights=None)

    WEIGHTS_DIR = 'weights/' + backbone
    print('Loading weights from:', WEIGHTS_DIR)
    for layer in tqdm(model.layers):
        if layer.weights:
            weights = []
            for w in layer.weights:
                weight_name =
os.path.basename(w.name).replace(':0', '')
                weight_file = layer.name + '_' + weight_name +
'.npy'
                weight_arr = np.load(os.path.join(WEIGHTS_DIR,
weight_file))

                weights.append(weight_arr)
            layer.set_weights(weights)
    print('Saving model weights...')
    OUTPUT_WEIGHT_FILENAME = 'deeplabv3_' + \
        backbone + '_tf_dim_ordering_tf_kernels.h5'

```

```

        if not os.path.exists(MODEL_DIR):
            os.makedirs(MODEL_DIR)
        model.save_weights(os.path.join(MODEL_DIR,
OUTPUT_WEIGHT_FILENAME))

```

Файл extract_weights

```

from __future__ import print_function

import os
import numpy as np
import tensorflow as tf
from keras.utils.data_utils import get_file
def get_xception_filename(key):
    """Rename tensor name to the corresponding Keras layer
weight name.

    # Arguments
        key: tensor name in TF (determined by
tf.variable_scope)
    """
    filename = str(key)
    filename = filename.replace('/', '_')
    filename = filename.replace('xception_65_', '')
    filename = filename.replace('decoder_', '', 1)
    filename = filename.replace('BatchNorm', 'BN')
    if 'Momentum' in filename:
        return None
    if 'entry_flow' in filename or 'exit_flow' in filename:
        filename = filename.replace('_unit_1_xception_module',
'')

    elif 'middle_flow' in filename:
        filename = filename.replace('_block1', '')
        filename = filename.replace('_xception_module', '')
    # from TF to Keras naming
    filename = filename.replace('_weights', '_kernel')
    filename = filename.replace('_biases', '_bias')
    return filename + '.npy'

```

```

def get_mobilenetv2_filename(key):
    """Rename tensor name to the corresponding Keras layer
weight name.

    # Arguments
        key: tensor name in TF (determined by
tf.variable_scope)
    """
    filename = str(key)
    filename = filename.replace('/', '_')
    filename = filename.replace('MobilenetV2_', '')
    filename = filename.replace('BatchNorm', 'BN')
    if 'Momentum' in filename:
        return None

    # from TF to Keras naming
    filename = filename.replace('_weights', '_kernel')
    filename = filename.replace('_biases', '_bias')
    return filename + '.npy'

def extract_tensors_from_checkpoint_file(filename,
output_folder='weights', net_name=None):
    """Extract tensors from a TF checkpoint file.

    # Arguments
        filename: TF checkpoint file
        output_folder: where to save the output numpy array
files

    if not os.path.exists(output_folder):
        os.makedirs(output_folder)
    reader = tf.train.NewCheckpointReader(filename)
    for key in reader.get_variable_to_shape_map():
        # convert tensor name into the corresponding Keras
layer weight name and save
        if net_name == 'xception':
            filename = get_xception_filename(key)
        elif net_name == 'mobilenetv2':
            filename = get_mobilenetv2_filename(key)
        if filename:
            path = os.path.join(output_folder, filename)

```

```

        arr = reader.get_tensor(key)
        np.save(path, arr)
        print("tensor_name: ", key)

    CKPT_URL =
'http://download.tensorflow.org/models/deeplabv3_pascal_trainval_2
018_01_04.tar.gz'
    CKPT_URL_MOBILE =
'http://download.tensorflow.org/models/deeplabv3_mnv2_pascal_train
val_2018_01_29.tar.gz'
    MODEL_DIR = 'models'
    MODEL_SUBDIR = 'deeplabv3_pascal_trainval'
    MODEL_SUBDIR_MOBILE = 'deeplabv3_mnv2_pascal_trainval'
    if not os.path.exists(MODEL_DIR):
        os.makedirs(MODEL_DIR)
    checkpoint_tar = get_file(
        'deeplabv3_pascal_trainval_2018_01_04.tar.gz',
        CKPT_URL,
        extract=True,
        cache_subdir='',
        cache_dir=MODEL_DIR)
    checkpoint_tar_mobile = get_file(
        'deeplabv3_mnv2_pascal_trainval_2018_01_29.tar.gz',
        CKPT_URL_MOBILE,
        extract=True,
        cache_subdir='',
        cache_dir=MODEL_DIR)
    checkpoint_file = os.path.join(MODEL_DIR, MODEL_SUBDIR,
'model.ckpt')
    extract_tensors_from_checkpoint_file(
        checkpoint_file, net_name='xception',
output_folder='weights/xception')
    checkpoint_file = os.path.join(
        MODEL_DIR, MODEL_SUBDIR_MOBILE, 'model.ckpt-30000')
    extract_tensors_from_checkpoint_file(
        checkpoint_file, net_name='mobilenetv2',
output_folder='weights/mobilenetv2')

```

ПРИЛОЖЕНИЕ В

(Обязательное)

Презентация