|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА СИСТЕМЫ ОБРАБОТКИ ИНФОРМАЦИИ И УПРАВЛЕНИЯ\_\_\_\_\_\_\_\_

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ***

***НА ТЕМУ:***

|  |
| --- |
| **Сегментация объектов на спутниковых снимках** |
|  |

Студент группы ИУ5-81Б **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** П. К. Шимолина

(код группы) (подпись, дата) (инициалы и фамилия)

Научный руководитель **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** А. Н. Нардид

(подпись, дата) (инициалы и фамилия)

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель от кафедры **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** В. И. Терехов

(подпись, дата) (инициалы и фамилия)

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Москва, 2024 г

**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технический университет имени Н.Э. Баумана**

**(национальный исследовательский университет)»**

**(МГТУ им. Н.Э. Баумана)**

УТВЕРЖДАЮ

Заведующий кафедрой \_\_\_ИУ5\_\_\_\_

(Индекс)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_В.И. Терехов\_\_

(И.О.Фамилия)

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г.

**ЗАДАНИЕ**

**на выполнение научно-исследовательской работы**

по теме Сегментация объектов на спутниковых снимках \_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студент группы \_ИУ5-81Б\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Шимолина Полина Кирилловна \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Фамилия, имя, отчество)

Направленность НИР (учебная, исследовательская, практическая, производственная, др.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ПРАКТИЧЕСКАЯ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Источник тематики (кафедра, предприятие, НИР) \_\_\_\_КАФЕДРА\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

График выполнения НИР: 25% к \_6\_ нед., 50% к \_10\_ нед., 75% к \_13\_ нед., 100% к \_16\_ нед.

***Техническое задание***  *Сравнение методов сегментации объектов, сравнение архитектур нейронных сетей для сегментации изображений, обучение модели, выбор метрик для оценки модели, разработка интерфейса веб-сервиса, выбор технологий для разработки веб-сервиса.*

***Оформление научно-исследовательской работы:***

Расчетно-пояснительная записка на \_\_38\_\_ листах формата А4.

Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата выдачи задания «\_25\_» \_\_\_апреля\_\_\_ 2024 г.

**Научный руководитель** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_**\_\_\_\_**\_ А. Н. Нардид\_\_\_\_\_

(Подпись, дата) (И.О.Фамилия)

**Студент группы ИУ5-81Б** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_П. К. Шимолина\_\_\_\_

(Подпись, дата) (И.О.Фамилия)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc165832182)

[1 СЕГМЕНТАЦИЯ ИЗОБРАЖЕНИЙ 5](#_Toc165832183)

[1.1 Типы задач сегментации изображений 5](#_Toc165832184)

[1.2 Бинаризация изображений 6](#_Toc165832185)

[1.2.1 Метод Оцу 7](#_Toc165832186)

[1.2.2 Метод Бернсена 10](#_Toc165832187)

[1.2.3 Метод Эйквеля 11](#_Toc165832188)

[1.3 Архитектуры нейронный сетей, применяемые для сегментации 13](#_Toc165832189)

[1.3.1 U-Net 13](#_Toc165832190)

[1.3.2 Mask R-CNN 14](#_Toc165832191)

[1.3.3 SegNet 18](#_Toc165832192)

[1.3.4 Сравнение архитектур 19](#_Toc165832193)

[2 ОБУЧЕНИЕ МОДЕЛИ U-NET 20](#_Toc165832194)

[2.1 Набор данных 20](#_Toc165832195)

[2.2 Предобработка и преобразования 21](#_Toc165832196)

[2.3 Модель 23](#_Toc165832197)

[2.3.1 Создание модели 23](#_Toc165832198)

[2.3.2 Выбор и реализация метрик 23](#_Toc165832199)

[2.3.3 Обучение модели 24](#_Toc165832200)

[3 СЕРВИС СЕГМЕНТАЦИИ SHAPEEXTRACT 28](#_Toc165832201)

[3.1 Используемые технологии 28](#_Toc165832202)

[3.2 Разработка сервиса 29](#_Toc165832203)

[3.2.1 Backend 29](#_Toc165832204)

[3.2.1.1 Модели 30](#_Toc165832205)

[3.2.1.2 Представления 32](#_Toc165832206)

[3.2.1.3 Эндпоинты 37](#_Toc165832207)

[3.2.2 Frontend 37](#_Toc165832208)

[3.3 Интерфейс приложения 37](#_Toc165832209)

[ЗАКЛЮЧЕНИЕ 43](#_Toc165832210)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 44](#_Toc165832211)

ВВЕДЕНИЕ

Сегментация изображений на спутниковых снимках является важной задачей в области обработки и анализа геопространственных данных. Автоматическое распознавание спутниковых снимков представляет собой наиболее перспективный подход для получения информации о расположении различных объектов на местности. Особенно актуальным становится отказ от ручной сегментации снимков, когда необходимо обработать большие участки земной поверхности в ограниченные сроки. Автоматизированный процесс распознавания позволяет существенно сократить время и усилия, а также минимизировать ошибки, связанные с человеческим фактором.

Распознавание объектов — это процесс нахождения конкретного объекта на изображении или видеопоследовательности. Каждому объекту на изображении соответствуют уникальные характеристики, которые описывают его особенности и свойства. Эти характеристики извлекаются из обучающего набора данных и используются для идентификации местоположения объекта, а затем сравниваются с множеством других объектов из тестового набора данных. Таким образом, главная цель обучения сверточных нейронных сетей (СНС) заключается в определении и извлечении характеристик объекта на изображении.

Цель исследования: создание эффективной модели сегментации объектов и предоставление удобного инструмента для работы с изображениями через веб-сервис.

Задачи исследования:

* Сравнить различные методы сегментации объектов на изображениях.
* Изучить архитектуры нейронных сетей, применяемых для сегментации.
* Обучить модель с использованием выбранных методов.
* Выбрать подходящие метрики для оценки качества модели.
* Разработать пользовательский интерфейс веб-сервиса для работы с моделью.
* Выбрать технологии для разработки веб-сервиса.

1. СЕГМЕНТАЦИЯ ИЗОБРАЖЕНИЙ
   1. Типы задач сегментации изображений

Задачи сегментации изображения можно разделить на типы на основе количества и типа информации, которую они передают. (см. рис.1)

* Семантическая сегментация.
* Сегментация экземпляра
* Паноптическая сегментация

Семантическая сегментация относится к классификации пикселей изображения по семантическим классам. Пиксели, принадлежащие к определенному классу, просто относятся к этому классу без учета другой информации или контекста. [10] Сегментация работает со множеством объектов одного класса как с единым целым.

Модели сегментации экземпляров классифицируют пиксели по категориям на основе «экземпляров», а не классов. Алгоритм сегментации экземпляров не знает к какому классу принадлежит классифицируемая область, но может разделить перекрывающиеся или очень похожие области объекта на основе их границ.

Паноптическая сегментация, самая последняя разработанная задача сегментации, может быть выражена как комбинация семантической сегментации и сегментации экземпляров, при которой каждый экземпляр объекта на изображении выделяется и предсказывается идентичность объекта.

На рисунке 1 наглядно видно различие между описанными тремя типами сегментации.

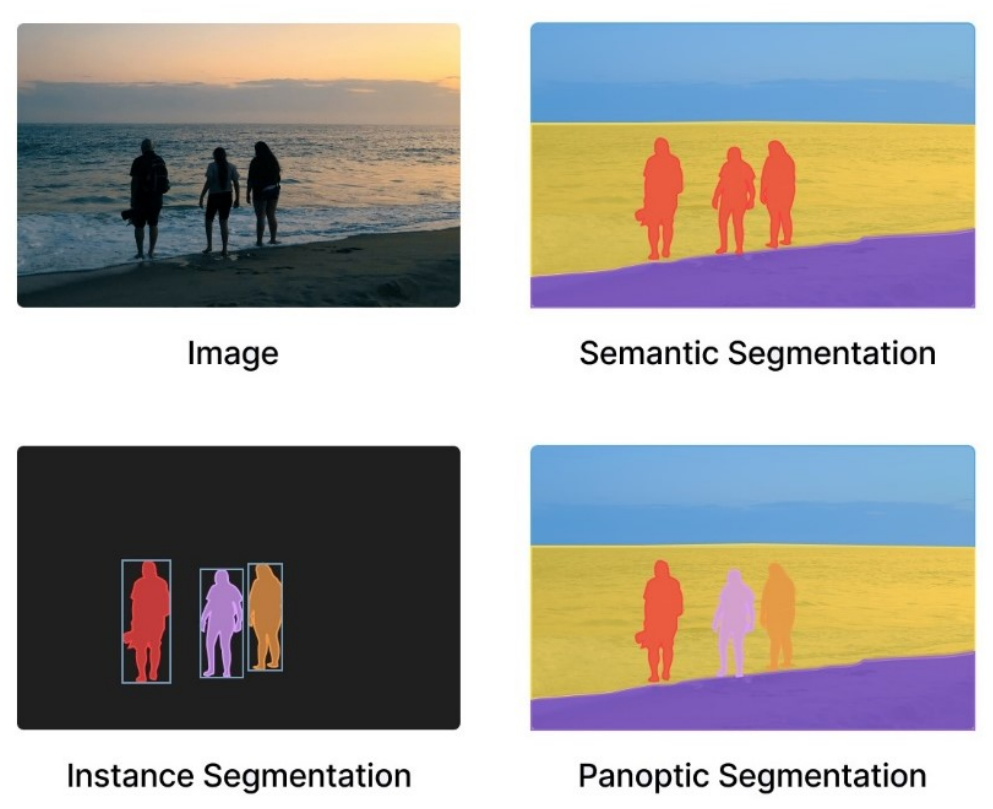


Рисунок 1 – Типы сегментации

Бинаризация изображений

Бинаризация — это процесс преобразования цветного изображения или изображения в градациях серого в двухцветное черно-белое. Яркость каждого пикселя сравнивается с пороговым значением, и ему присваивается либо 0, если пиксель является границей объекта, либо 1 если не является. Бинаризация используется для уменьшения количества информации, с которой будет производиться работа. Существуют пороговые (глобальные) и адаптивные (локальные) методы бинаризации. В пороговых происходит работа со всем изображением сразу. Величина порога постоянна для всего процесса бинаризации.

* + 1. Метод Оцу

Метод Оцу (Отсу) – метод глобальной пороговой обработки, минимизирующий среднюю ошибку от принятия решения о принадлежности пикселей изображения объекту или фону.

Простейший метод пороговой обработки – использование гистограммы изображения. На основании общего для всего изображения порога каждый пиксель обозначается как объект или фон. Такой способ подходит для случаев, когда гистограмма носит четко выраженный бимодальный характер (см. рис. 2).

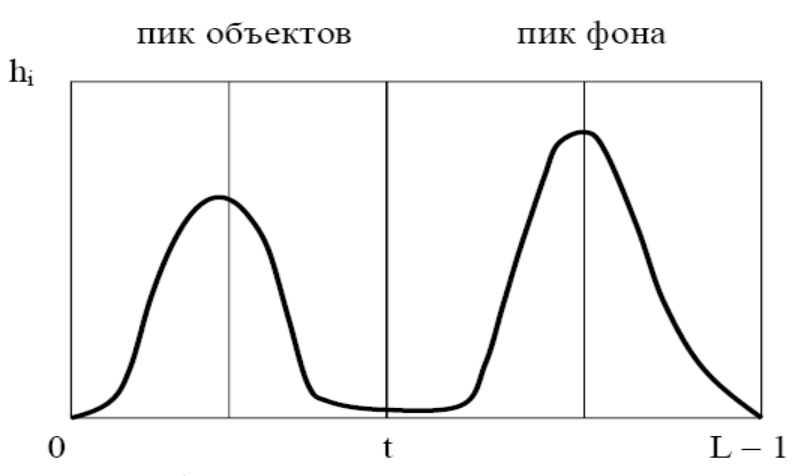


Рисунок – Гистограмма с двумя пиками (t – глобальный порог)

Для определения оптимального порога бинаризации T предложено огромное количество различных алгоритмов. Как правило, такие методы опираются на использование локальной или глобальной гистограммы изображения. Одним из наиболее широко используемых подходов является подход, предложенный Оцу; преимуществами данного алгоритма являются понятный статистический смысл и низкая вычислительная сложность. [2] В идеальном случае гистограмма имеет глубокую и четкую впадину между двумя пиками, представляющими объекты и фон соответственно, так что порог может быть выбран в нижней части этой впадины. Однако на большинстве реальных снимков часто бывает трудно точно определить дно долины, особенно в таких случаях, когда долина плоская и широкая, наполненная шумом, или когда две вершины чрезвычайно неравны по высоте, часто не образуя прослеживаемой долины. Предложен новый метод с точки зрения дискриминантного анализа; он непосредственно приближается к возможности оценки "добротности" порога и автоматического выбора оптимального порога. [4]

Метод бинаризации Оцу используется для определения порога T, который минимизирует среднюю ошибку сегментации изображения, возникающую при принятии решения о принадлежности пикселей изображения объекту или фону. В данном подходе значения яркостей пикселей рассматриваются как случайные величины, а их гистограмма - как оценка плотности распределения вероятностей. Если мы знаем плотности распределения вероятностей, то мы можем определить оптимальный порог, минимизирующий ошибку для разделения изображения на два класса: объекты и фон.

Рассмотрим относительные частоты для этих классов: [1]

где w1 и w2 – области объектов и фона соответственно;

i – номер элемента матрицы изображения;

pi – значения гистограммы;

k – количество элементов в первой или второй области (объект или фон);

L – максимальное значение, которое может принять T;

µ1, µ2 – средняя яркость всего изображения.

Тогда дисперсия внутри классов следующая:

Дисперсия внутри классов может быть рассчитана как взвешенная сумма дисперсий каждого отдельного класса:

Дисперсия между двумя классами:

Дисперсия всего изображения:

Метод Оцу выбирает оптимальный порог T, c помощью вычисления максимизации между классом дисперсии:

Метод ОЦУ прост в реализации, быстр в выполнении и адаптивен к различным изображениям, но чувствителен к неравномерной яркости изображения.

* + 1. Метод Бернсена

Метод Бернсена реализован и протестирован для различных значений соседства и пределов контрастности.

Изображение делится на квадраты r × r (r – нечетное) с центром в точке (x, y). [3] Для каждого пикселя изображения в пределах квадрата считается порог согласно равенству:

где Zlow и Zhigh – самые низкие и самые высокие значения пикселей уровня серого в квадрате r x r с центром в точке (x, y) в соответствии с уравением:

Если показатель контрастности C(x, y) < l, то окрестности состоят только из одного класса, переднего плана или фона. Кроме того, значения r и l варьируются в зависимости от используемых изображений и областей.

Применение метода включает следующе этапы: [5]

* Считать изображение в оттенках серого
* Определить значение близости (r) и предел контрастности (l)
* Найти наименьшее значение пикселя уровня серого (Zlow) в квадрате r x r с центром в точке (x, y)
* Найти наибольшее значение пикселя уровня серого (Zhigh) в квадрате r x r с центром в точке (x, y)
* Посчитать пороговое значение T(x, y)
* Посчитать показатель контрастности C(x, y)
* Обозначить пиксель (x, y) как передний план или фон в соответствии с рассчитанным пороговым значением и мерой контрастности
* Повторить пункты 3 – 7 для каждого пикселя
  + 1. Метод Эйквеля

В этом методе используются два окна r и R, из которых большее по размерам (окно L) служит для вычисления значения порога, в то время как меньшее (окно S) определяет область изображения, в которой будет использоваться полученный порог. Оба окна перемещаются параллельно по изображению с шагом, равным размеру меньшего окна S, и каждый раз для всех элементов окна L вычисляется оптимальный порог по методу Отса (см. рис. 3). [3]

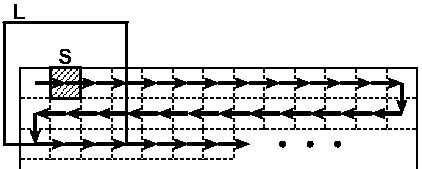


Рисунок – Перемещение окон в методе Эйквеля

Пиксели внутри окна S подвергаются бинаризации в соответствии с вычисленным порогом t, если вычисленные средние значения µ0 и µ1 различаются сильно и для них выполняется условие:

где ε – заданный параметр. И все пиксели внутри окна S относятся к классу с ближайшим средним значением если выполняется

Архитектуры нейронный сетей, применяемые для сегментации

Сверточные нейронные сети являются мощным инструментом в области компьютерного зрения и имеют широкое применение в задачах сегментации изображений. В последние годы было предложено множество различных моделей СНС для сегментации, каждая из которых имеет свои особенности и преимущества.

* + 1. U-Net

U-Net является одной из основных архитектур сверточных нейронных сетей (СНС) для задач сегментации изображений. Применяется не только для определения класса изображения, но и для разделения на области и создания маски для каждого класса. U-Net – очень быстрая сеть, способная сегментировать изображение 512x512 менее чем за секунду.

Архитектура состоит из стягивающего пути для захвата контекста и симметричного расширяющегося пути, который позволяет осуществить точную локализацию (см. рис. 4). [6] То есть стягивающий путь содержит слои, фиксирующие информацию с изображений, уменьшающие пространственное разрешение входных данных, а расширяющийся путь содержит слои, декодирующие закодированные данные, определяющие расположение объектов и создающие маски сегментации, используя информацию, полученную на стягивающем пути.

Рисунок 4 демонстрирует преобразование изображение размером 572 на 572 в оттенках градации серого в двоичную сегментированную выходную карту размером 388×388×2. Размер выходных данных меньше размера входных данных, потому что не используется заполнение. Однако, если использовать “padding”, можно сохранить размер входных данных.

В процессе сжатия пути входное изображение постепенно уменьшается по высоте и ширине, но увеличивается количество каналов. Это увеличение каналов позволяет сети захватывать объекты высокого уровня по мере продвижения по пути. В узком месте выполняется заключительная операция свертки для создания карты объектов размером 30 × 30 × 1024. Затем расширенный путь извлекает карту объектов из узкого места и преобразует ее обратно в изображение того же размера, что и исходные входные данные. Это делается с использованием слоев с повышающей дискретизацией, которые увеличивают пространственное разрешение карты объектов при одновременном сокращении количества каналов. Наконец, каждый пиксель на выходном изображении представляет собой метку, соответствующую определенному объекту или классу на входном изображении. [7]

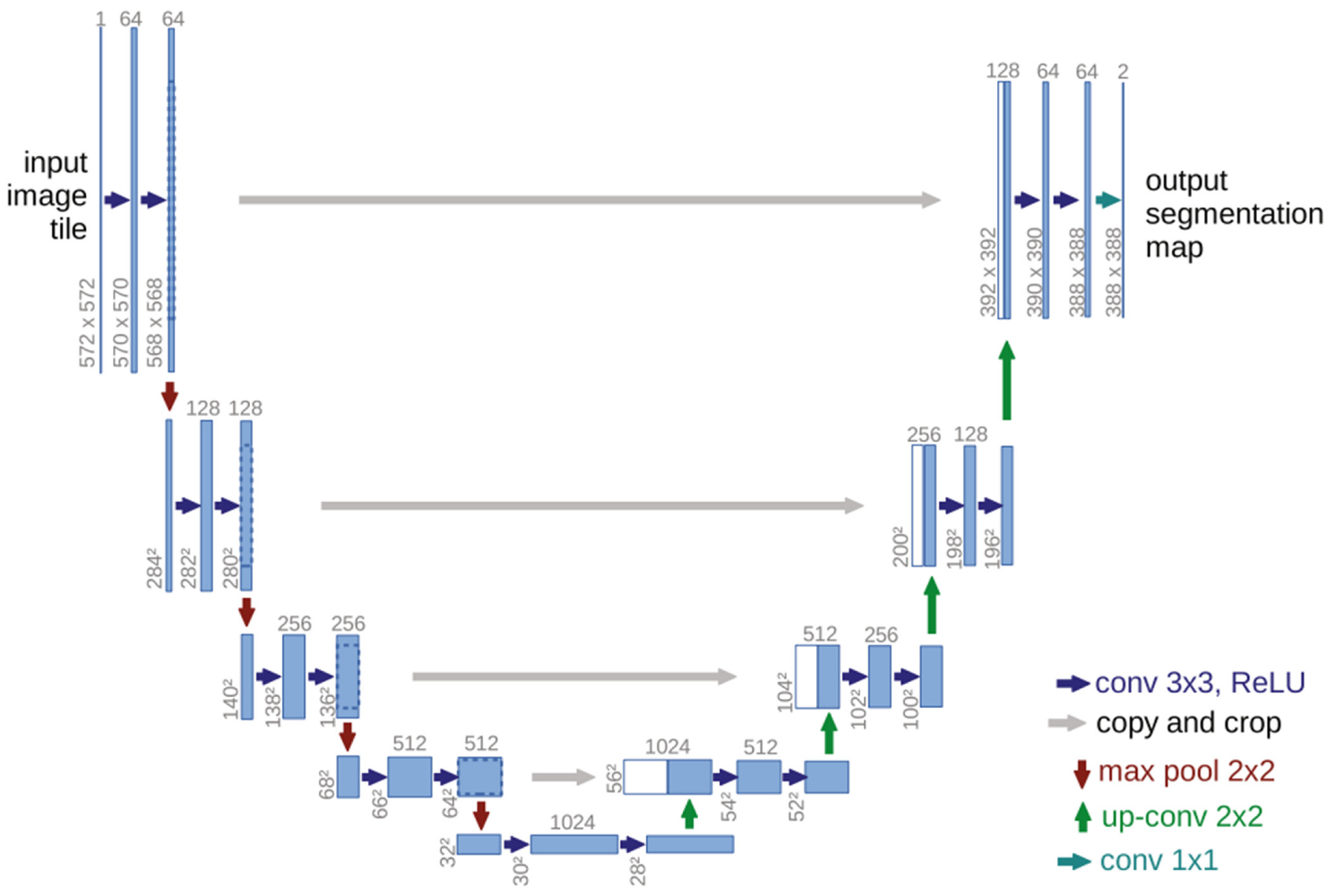


Рисунок – Архитектура U-Net

* + 1. Mask R-CNN

R-CNN или RCNN, расшифровывается как Region-Based Convolutional Neural Network – сверточная нейронная сеть на основе регионов, это тип модели машинного обучения, которая используется для задач компьютерного зрения, в частности для обнаружения объектов.

Рассмотрим алгоритм работы. Сначала изображение разделяется на множество прямоугольных регионов, которые могут содержать объекты. Эти регионы выбираются с помощью алгоритма селекции регионов. Затем каждый выбранный регион подвергается процессу извлечения признаков с помощью сверточной нейронной сети (CNN). В качестве CNN-сети используется готовая CaffeNet, классифицирующая на 1000 классов. CNN анализирует содержимое каждого региона и извлекает характеристики, которые могут помочь в определении наличия объекта. Полученные признаки передаются в классификатор, который определяет, принадлежит ли регион к одному из заданных классов объектов или относится к фону (не содержит объекта). Наконец, если регион классифицирован как содержащий объект, применяется дополнительная обработка, чтобы точнее определить границы объекта и создать окончательную маску или охватывающую рамку объекта. (см. рис. 5)

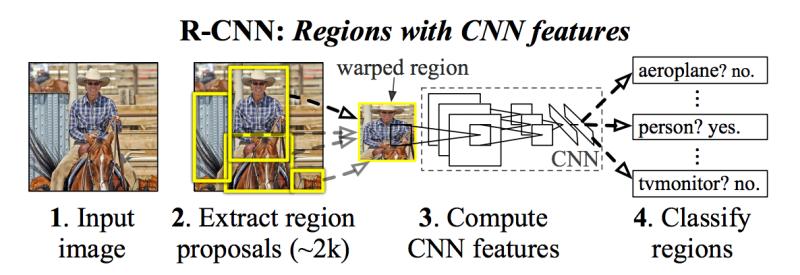


Рисунок – Схема R-CNN

Далее была разработана Fast R-CNN, которая отличалась тем, что через CNN пропускалось все изображение целиком. (см. рис. 6)

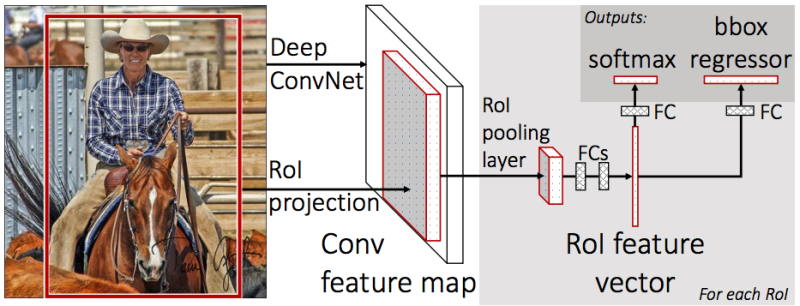


Рисунок – Схема Fast R-CNN

Следующей была создана Faster R-CNN. Которая вычислять регионы не по изначальному изображению, а опять же по карте признаков, полученных из CNN. Для этого был добавлен модуль под названием Region Proposal Network (RPN). (см. рис. 7)

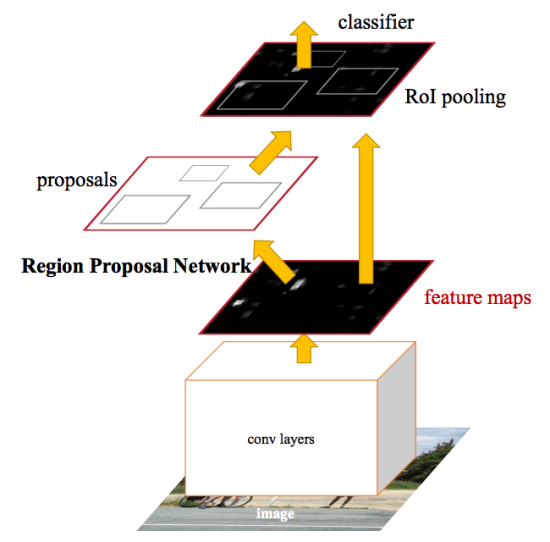


Рисунок 7 - Схема Faster R-CNN

Mask R-CNN развивает архитектуру Faster R-CNN. Существует 2 основных типа сегментации изображений, которые подпадают под Mask R-CNN:

* Семантическая сегментация
* Сегментация экземпляров

Семантическая сегментация классифицирует каждый пиксель по фиксированному набору категорий без дифференциации экземпляров объектов. То есть семантическая сегментация имеет дело с идентификацией/классификацией похожих объектов как единого класса на уровне пикселей.

Сегментация экземпляров имеет дело с правильным обнаружением всех объектов на изображении, а также с точной сегментацией каждого экземпляра. Таким образом, это комбинация обнаружения объектов, локализации объектов и классификации объектов. Другими словами, этот тип сегментации идет дальше, чтобы провести четкое различие между каждым объектом, классифицированным как похожие экземпляры.

Маска R-CNN является расширением более быстрого R-CNN и работает путем добавления ветви для прогнозирования маски объекта (интересующей области) параллельно с существующей ветвью для распознавания ограничивающих рамок. [8]

Маска представляет собой матрицу, где каждый пиксель имеет значение 1 или 0. (см. рис. 8)

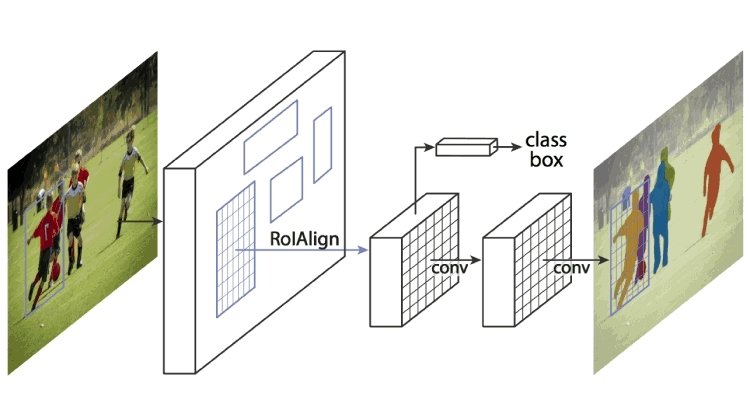


Рисунок - Схема Mask R-CNN

* + 1. SegNet

SegNet - это эффективная архитектура для пиксельной семантической сегментации.

SegNet имеет сеть кодировщиков и соответствующую сеть декодеров, за которыми следует последний слой классификации по пикселям. (см. рис. 9)

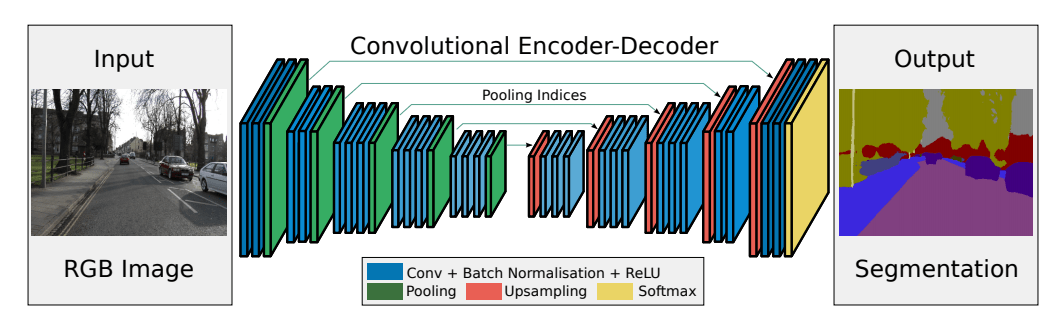


Рисунок – Архитектура SegNet

Сеть кодировщика состоит из 13 сверточных слоев которые соответствуют первым 13 сверточным слоям в сети VG16, предназначенной для классификации объектов. Каждый уровень кодера имеет соответствующий уровень декодера, и, следовательно, сеть декодера имеет 13 уровней. Окончательный вывод декодера подается на мульти-классификатор soft-max для независимого получения вероятностей классов для каждого пикселя.

SegNet является эффективным и гибким инструментом для сегментации изображений, который обладает рядом преимуществ и особенностей, делающих его популярным в области компьютерного зрения.

* + 1. Сравнение архитектур

В таблице 1 представлены сравнительные характеристики методов для сегментации изображений, описанных выше.

Таблица 1 – Сравнение методов для сегментации изображений

| **№** | **Метод** | **Архитектура** | **Типы изображений** | **Производительность** |
| --- | --- | --- | --- | --- |
| 1 | U-Net­ | Энкодер-декодер | Разработана для медиинских изображений с возможностью адаптации для других типов | Хорошая производительность для разных типов данных |
| 2 | Mask R-CNN | Расширенная Faster R-CNN и полносвязная нейросеть | Для любых типов, включая спутниковые снимки | Хорошо работает для сложных форм и нескольких экземпляров |
| 3 | SegNet | Энкодер-декодер | Для любых типов, включая спутниковые снимки | Хорошо справляется с сегментацией небольших классов |

1. ОБУЧЕНИЕ МОДЕЛИ U-NET

Ранее мы провели сравнение трех архитектур моделей для сегментации объектов на изображениях. С учётом анализа исследования, мы используем архитектуру U-Net для реализации задачи сегментации спутниковых снимков. Данная архитектура показала потенциал в эффективности обучения модели и может быть ключевым инструментом для создания нашего веб-сервиса по обработке изображений.

* 1. Набор данных

Для обучения модели воспользуемся размеченным набором данных. Набор данных состоит из аэрофотоснимков Дубая, полученных со спутников MBRSC и аннотированных с помощью попиксельной семантической сегментации по 6 классам. Общий объем набора данных — 72 изображения.

Классы и соответствующие им цвета на масках приведены в таблице 2.

Таблица 2 – Соответствие классов и цветов на масках

| **Класс** | **Цветовой индекс** | **Цвет** |
| --- | --- | --- |
| Здание(грунтовой участок) | #3C1098 |  |
| Земельный участок | #8429F6 |  |
| Дорога | #6EC1E4 |  |
| Растительность | #FEDD3A |  |
| Вода | #E2A929 |  |
| Без маркировки | #9B9B9B |  |

На рисунке 9 представлен пример изображения, снятого со спутника, и соответствующей ему маске. Изображение содержит озеро, деревья, дороги, здания, землю, а маска содержит соответствующие классы приведенные в таблице 2.

A collage of maps of land and water

Description automatically generated

Рисунок – Пример изображения и соответствующей ему маски

* 1. Предобработка и преобразования

До передачи данных в модель нейронной сети для обучения нужно их подготовить. Для начала переведем HEX-коды цветов в значения RGB и назначим каждому классу метки. Получим значения указанные в таблице 3.

Таблица 3 – Метки классов

| **Метка класса** | **Класс** | **HEX** | **RGB** | **Цвет** |
| --- | --- | --- | --- | --- |
| 0 | Здание(грунтовой участок) | #3C1098 | (226, 169, 41) |  |
| 1 | Земельный участок | #8429F6 | (132, 41, 246) |  |
| 2 | Дорога | #6EC1E4 | (110, 193, 228) |  |
| 3 | Растительность | #FEDD3A | (60, 16, 152) |  |
| 4 | Вода | #E2A929 | (254, 221, 58) |  |
| 5 | Без маркировки | #9B9B9B | (155, 155, 155) |  |

Реализуем класс Aeroscapes\_Dataset, который будет представлять наш датасет. Тогда функция \_\_getitem\_\_(self, index) будет считывать изображение и маску и конвертировать их из цветового пространства BGR в RGB, т.к. используется библиотека OpenCV, в коорой цветовое пространство по умолчанию - BGR (синий, зеленый, красный). Затем маска преобразуется из формата RGB в маску, использующую индексы классов, согласно таблице 3. После этого изображение преобразуется в тензоры, нормализуется с использованием средних и стандартных отклонений, а маска из массива NumPy преобразуется в тензор PyTorch типа long. Таким образом функция вернет 2 тензора: тензор маски и тензор изображения.

Определим наборы преобразований t\_train и t\_val для обучающего и валидационного наборов данных. Набор преобразований включает в себя изменение размера изображений, отражения по горизонтали и вертикали, искажения сетки, случайное изменение яркости и контраста, а также добавление гауссовского шума.

Затем создадим объекты ранее упомянутого класса Aeroscapes\_Dataset для обучающего и валидационного наборов данных, указывая пути к изображениям, маскам, индексы обучающих данных, средние и стандартные значения для нормализации, набор преобразований.

Наконец, создадим загрузчики данных DataLoader для обучающего и валидационного наборов данных с указанным размером пакета и перемешиванием данных. Эти загрузчики данных используются для эффективной передачи данных в процессе обучения модели.

* 1. Модель
     1. Создание модели

Создадим модель сегментации изображений с использованием архитектуры UNet, который использует предварительно обученную MobileNetV2 в качестве энкодера.

Передаем следующие параметры:

* 'mobilenet\_v2' – указывает на использование архитектуры MobileNetV2 в качестве энкодера модели,
* encoder\_weights='imagenet' – загружает предобученные веса энкодера из набора данных ImageNet для инициализации модели,
* classes=6 – устанавливает количество классов для задачи сегментации на 6,
* activation=None – отключает функцию активации на выходном слое модели,
* encoder\_depth=5 – указывает на глубину энкодера, то есть количество слоев энкодера,
* decoder\_channels=[256, 128, 64, 32, 16] – указывает количество каналов на каждом уровне декодера для создания соответствующей архитектуры декодера модели UNet.
  + 1. Выбор и реализация метрик

Для оценки точности модели были реализованы следующие метрики.

Intersection over Union (IoU) — это мера, используемая для оценки качества сегментации объектов на изображениях. Она вычисляется как отношение пересечения между предсказанным и истинным масками к их объединению. Она позволяет оценить, насколько хорошо модель выделяет объекты на изображении и правильно определяет границы объектов.

Пиксельная точность (pixel accuracy) представляет собой простую метрику, показывающую долю правильно классифицированных пикселей в изображении. Она позволяет оценить общую точность модели в сегментации объектов по пикселям.

Выбор этих метрик обоснован тем, что они позволяют оценить качество сегментации изображений с высокой точностью и являются стандартными метриками для задачи сегментации. Использование этих метрик помогает отслеживать прогресс обучения модели и принимать решения о ее улучшении на основе конкретных численных показателей.

* + 1. Обучение модели

Будем обучать модель на данных из train\_loader и оценивает на данных из val\_loader. Установим максимальную скорость обучения = 10-4, будем обучать модель в течение 15 эпох. Зададим условие, что обучние останавливается, когда потери не уменьшаются в течение 6 эпох.

В результате обучение остановилось на 14 эпохах и мы получили следующие графики, изображенные на рисунках 11, 12, 13.

A graph with blue and orange lines

Description automatically generated

Рисунок 11 – График потерь

A graph with blue and orange lines

Description automatically generated

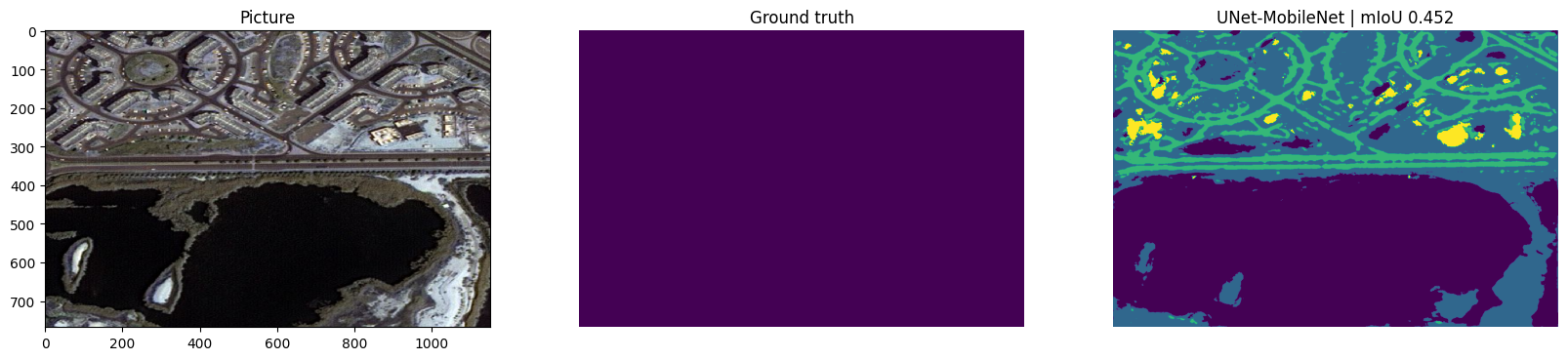
Рисунок 12 – График среднего значения метрики IoU

A graph with blue and orange lines

Description automatically generated

Рисунок 13 – График точности

В результате работы модели получаем сегментированное изображение.

A purple rectangle with white text

Description automatically generated

Рисунок 14 – Результат сегментации

Сохраняем модель для дальнейшего использования в веб-сервисе.

1. СЕРВИС СЕГМЕНТАЦИИ SHAPEEXTRACT

Сервис сегментации ShapeExtract – сервис сегментации, который использует модель машинного обучения для выделения форм объектов на изображениях. Для сегментации объектов на изображениях применяется архитектура U-Net.

Процесс работы сервиса ShapeExtract включает в себя загрузку изображения, которое подается на вход модели. Таким образом, пользователь может легко определить форму и контуры объектов на изображении с помощью данного сервиса.

* 1. Используемые технологии

Для бэкенд-сервера был выбран фреймворк Django, который предоставляет удобные инструменты для создания веб-приложений на Python. Django обладает множеством возможностей, включая встроенный механизм аутентификации пользователей, защиту от типичных уязвимостей веб-приложений и многое другое. [12] Благодаря широкому сообществу разработчиков разработка веб-сервиса на Django эффективна и удобна.

Для хранения данных была выбрана база данных SQLite. SQLite — это внутрипроцессная библиотека, реализующая автономный, бессерверный, не требующий настройки транзакционный механизм базы данных SQL. [11] SQLite является легковесной и простой в использовании базой данных, которая не требует отдельного сервера. Она хорошо подходит для небольших и средних проектов, таких как веб-сервисы, благодаря своей простоте в настройке и использовании. SQLite также поддерживает стандарт SQL и обладает множеством функций, что делает её удобным выбором для многих приложений.

Для фронтенд-разработки был выбран JavaScript, один из самых популярных языков программирования для создания интерактивных веб-страниц. JavaScript позволяет создавать динамические элементы на веб-страницах, общаться с сервером без перезагрузки страницы (AJAX), а также обеспечивать интерактивное взаимодействие с пользователем. Использование JavaScript позволит сделать веб-сервис более динамичным и привлекательным для пользователей.

Выбор этих технологий обоснован их совместимостью друг с другом, а также их способностью обеспечить удобную разработку, масштабируемость и безопасность веб-сервиса. Кроме того, использование Django позволяет легко интегрировать SQLite и JavaScript в проект, обеспечивая эффективную работу всей системы.

* 1. Разработка сервиса

При разработке веб-сервиса обычно выделяют несколько ключевых компонентов:

1. Бэкенд (Backend): это серверная часть веб-сервиса, которая отвечает за обработку запросов пользователя, взаимодействие с базой данных, бизнес-логику приложения.

2. Фронтенд (Frontend) – это та часть веб-сервиса, с которой взаимодействует пользователь. Фронтенд отвечает за отображение контента, интерактивные элементы, пользовательский интерфейс.

3. База данных (Database): это хранилище данных, используемое веб-сервисом для сохранения информации. Выбор базы данных зависит от требований проекта.

* + 1. Backend

Бэкенд (Backend) — это серверная часть веб-приложения, которая отвечает за обработку запросов пользователя, взаимодействие с базой данных, бизнес-логику приложения и генерацию ответов, которые отображаются на фронтенде. Основными компонентами бэкенда обычно являются:

1. Сервер: это программное обеспечение, которое обеспечивает работу веб-сервиса, принимает запросы от клиентов и возвращает ответы. Сервер может быть написан на различных языках программирования и работать с использованием различных протоколов, таких как HTTP.

2. База данных: это хранилище данных, которое используется для сохранения информации, необходимой для работы приложения. База данных хранит данные в структурированном формате и обеспечивает доступ к ним через запросы.

3. Бизнес-логика: это часть приложения, которая определяет, как приложение должно обрабатывать запросы пользователя и выполнять определенные задачи. Бизнес-логика включает в себя правила и процессы, которые определяют логику работы приложения.

4. API (Application Programming Interface): это интерфейс, который определяет способы взаимодействия между различными компонентами приложения. API позволяет передавать данные и команды между клиентской и серверной частями приложения.

Взаимодействие между компонентами бэкенда обычно происходит следующим образом:

* Клиент отправляет запрос на сервер через сеть (например, через протокол HTTP).
* Сервер принимает запрос, обрабатывает его с помощью бизнес-логики и доступа к базе данных.
* Сервер формирует ответ на запрос и отправляет его обратно клиенту.
* Клиент получает ответ от сервера и отображает его пользователю на фронтенде.
  + - 1. Модели

Модели в Django представляют собой классы Python, которые отображают структуру базы данных. Они определяют, как данные будут храниться и взаимодействовать с базой данных. Модели позволяют создавать, обновлять, удалять и извлекать данные из базы данных, используя ORM (Object-Relational Mapping). Каждая модель соответствует таблице в базе данных и определяет поля, связи и методы для работы с данными.

При разработке сервиса были созданы следующие модели:

* Модель Team отвечает за хранение информации о командах, содержит поля:
  + teamname - поле типа CharField для названия команды.
  + description - поле типа TextField для описания команды.
  + created\_at - поле типа DateTimeField, автоматически заполняемое с текущей датой и временем при создании команды.
  + creator - внешний ключ на модель пользователя Django (User) с параметром related\_name='teams'.
* Модель User представляет собой базовую модель пользователя, которая предоставляется фреймворком Django для управления пользователями в приложении, импортируется из django.contrib.auth.models. Поля модели:
  + username: уникальное имя пользователя.
  + password: зашифрованный пароль пользователя.
  + email: электронная почта пользователя.
  + first\_name: имя пользователя.
  + last\_name: фамилия пользователя.
  + is\_staff: флаг, определяющий, имеет ли пользователь доступ к административной панели.
  + is\_active: флаг, указывающий, активен ли пользователь.
  + date\_joined: дата и время регистрации пользователя.
* Модель UploadedPhoto отвечает за хранение информации о загруженных файлах, содержит поле file - файловое поле для загрузки файла.
* Модель SegmentedPhoto отвечает за хранение сегментированных изображений, содержит поле photo - поле типа ImageField для загрузки изображения.
* Модель UserPhoto связывает загруженные файлы, сегментированные изображения и пользователей. Поля:
  + uploaded\_photo - связь с моделью UploadedFile по принципу "один к одному".
  + segmented\_photo - связь с моделью SegmentedPhoto по принципу "один к одному".
  + user - внешний ключ на модель пользователя Django (User).
  + is\_visible\_to\_team - поле типа BooleanField со значением по умолчанию False, которое демонстрирует доступно ли фото команде пользователя.
* Модель UserProfile связывает пользователей с командами и расширяет базовую модель пользователя Django. Поля:
* user - связь с моделью пользователя Django по принципу "один к одному".
* team - внешний ключ на модель Team с параметром on\_delete=models.SET\_NULL, позволяющий установить значение null, если команда была удалена.
  + - 1. Представления

Представления в Django определяют логику обработки запросов от пользователей и формирования ответов. Они могут выполнять различные операции, такие как чтение и запись данных, валидация данных, рендеринг шаблонов и другие действия. Представления могут работать с данными из моделей, обрабатывать формы, вызывать другие представления и выполнять бизнес-логику приложения. Представления могут быть функциональными или классами, в зависимости от сложности логики обработки запросов. Модели обеспечивают структуру данных и бизнес-логику, а представления управляют отображением и обработкой данных. Представления используют модели для получения, обновления и сохранения данных в базе данных. Модели и представления работают совместно для обеспечения функциональности веб-приложения, где модели предоставляют доступ к данным, а представления обрабатывают запросы пользователей.

В ходе разработки сервиса были описаны следующие представления, необходимые для обработки запросов.

**User**. Это представление обеспечивает возможность получения данных о пользователе (GET) и обновления данных пользователя (PUT) через API. Оно также обеспечивает проверку аутентификации и авторизации с использованием JWT-аутентификации и определяет права доступа к представлению. Оно является функциональным представлением в Django REST framework. Содержит декоратор @api\_view для преобразования обычной функции в API-представление, поддерживающее методы GET и PUT. Декоратор @permission\_classes определяет права доступа для данного представления - пользователь должен быть аутентифицирован. Декоратор @authentication\_classes указывает метод аутентификации - JWT-аутентификация. Функция user(request) принимает объект request, представляющий HTTP-запрос. В случае GET-запроса возвращает JSON-ответ с данными пользователя, полученными с помощью UserSerializer для текущего аутентифицированного пользователя. В случае PUT-запроса получает пользователя по идентификатору текущего пользователя из запроса, инициализирует сериализатор UserSerializer с данными из запроса. После валидации данных сериализатором, сохраняет обновленные данные пользователя и возвращает обновленные данные в формате JSON. Если данные не прошли валидацию, возвращает ошибки валидации с кодом статуса 400 BAD REQUEST.

**UserProfileView.** Это представление позволяет получить информацию о пользователе по его имени пользователя и возвращает данные пользователя в формате JSON. Если пользователь не найден, возвращается сообщение об ошибке.

Является представлением на основе класса View в Django. Он определяет метод get(self, request, username), который обрабатывает GET-запросы и принимает параметр username. Метод пытается получить объект пользователя из базы данных по имени пользователя, переданному в запросе. Если пользователь с таким именем пользователя найден, создается словарь user\_data с данными пользователя (идентификатор, имя пользователя, email, имя, фамилия).

В случае успешного поиска пользователя возвращает JSON-ответ с данными пользователя. Если пользователь с указанным именем пользователя не найден, возвращает JSON-ответ с сообщением об ошибке и статусом 404.

**UserDetail.** Предоставляет стандартные методы для получения и обновления данных конкретного пользователя с помощью Django REST framework. Он использует QuerySet для доступа к объектам модели User и сериализатор UserSerializer для преобразования данных.

Класс наследуется от RetrieveUpdateAPIView в Django REST framework, что предоставляет готовые методы для извлечения (GET) и обновления (PUT) экземпляра модели. RetrieveUpdateAPIView является обобщенным представлением, которое автоматически обрабатывает запросы для извлечения и обновления объекта модели.

При GET-запросе класс RetrieveUpdateAPIView автоматически обрабатывает запрос и извлекает данные о конкретном пользователе из указанного QuerySet.

При PUT-запросе также автоматически обрабатывается обновление данных конкретного пользователя с использованием данных из запроса.

**UserTeamView.** Класс UserTeamView представляет собой представление Django, которое обрабатывает запросы GET, POST, PUT и DELETE для получения, назначения, изменения и удаления команды у пользователя.

Метод get(self, request, user\_id) пытается получить профиль пользователя из базы данных с указанным user\_id, используя метод select\_related('team') для эффективного извлечения связанных данных о команде. Если профиль пользователя найден, создается словарь team\_data с данными о команде, к которой принадлежит пользователь. Возвращается JSON-ответ с информацией о команде, включая идентификатор команды, название команды, описание, дату создания и создателя команды. Если профиль пользователя не найден, возвращается JSON-ответ с сообщением об ошибке и статусом 404.

Метод post(self, request, user\_id) обрабатывает POST-запрос для назначения команды пользователю с определенным user\_id, извлекает team\_id из тела запроса, получает объект команды Team по данному team\_id. Затем создает или обновляет профиль пользователя UserProfile с присвоением команды team, возвращает сообщение об успешном назначении команды пользователю.

Метод put(self, request, user\_id) обрабатывает PUT-запрос для изменения команды пользователя с определенным user\_id, получает данные из тела запроса, извлекает team\_id и получает объект команды Team, обновляет команду пользователя в профиле user\_profile и сохраняет изменения, возвращает сообщение об успешном изменении команды пользователя.

Метод delete(self, request, user\_id) обрабатывает DELETE-запрос для удаления команды у пользователя с определенным user\_id, получает профиль пользователя user\_profile по user\_id, удаляет команду у пользователя и сохраняет изменения, возвращает сообщение об успешном удалении пользователя из команды.

**RegisterUser.** Представляет собой представление Django REST framework, которое обрабатывает POST-запрос для регистрации нового пользователя.

Метод post(self, request) обрабатывает POST-запрос для регистрации нового пользователя на основе переданных данных (имя пользователя, пароль, имя, фамилия, email). Извлекает данные из тела запроса request.data для имени пользователя, пароля, имени, фамилии и email, проверяет, что обязательные поля (имя пользователя и пароль) не пустые. Если какое-то из них отсутствует, возвращается сообщение об ошибке и статус 400, создает нового пользователя с использованием метода create\_user модели User и переданных данных.

В случае успешной регистрации пользователя возвращается сообщение об успешной регистрации и статус 201.

В случае возникновения ошибки при создании пользователя возвращается сообщение об ошибке и статус 400.

**UploadPhotoView. [TBD]** Этот класс представления обрабатывает процесс загрузки и обработки фотографий, включая сохранение сегментированного изображения, загруженного файла и связанных данных пользователя.

Создает экземпляр сериализатора SegmentedPhotoSerializer для обработки данных изображения, переданных в теле запроса. Проверяет, что данные изображения являются допустимыми с помощью is\_valid(). Если данные корректны, сохраняет данные изображения и получает объект segmented\_photo. Затем создает сериализатор UploadedFileSerializer для обработки загруженного файла (изображения). Проверяет данные загруженного файла на валидность. Если данные верные, сохраняет данные загруженного файла и получает объект uploaded\_photo. Создает данные для сериализатора UserPhotoSerializer, включая идентификаторы загруженного и сегментированного изображений, идентификатор пользователя и флаг видимости для команды. Проверяет данные на валидность и, если они корректны, сохраняет данные пользователя.

Возвращает ответ с данными нового пользователя и статусом 201 в случае успешного создания.

В случае возникновения ошибок в процессе валидации данных, возвращаются соответствующие ошибки и статус 400.

**TeamView.** Этот класс представления позволяет создавать новые команды с помощью POST-запроса и получать информацию обо всех командах с помощью GET-запроса.

Метод post(self, request) создает экземпляр сериализатора TeamSerializer для обработки данных команды, переданных в теле POST-запроса. Проверяет, что данные команды являются допустимыми с помощью is\_valid(). Если данные корректны, извлекает необходимые данные о команде (название, описание, дата создания, создатель) из сериализатора. Создает новую команду в базе данных, используя полученные данные. Возвращает ответ с сообщением об успешном создании команды и идентификатором созданной команды. В случае успешного создания команды возвращается сообщение об успешном создании и статус 201, а при получении списка всех команд возвращаются данные всех команд. В случае ошибок возвращаются соответствующие сообщения об ошибках и статус 400.

Метод get(self, request) получает все команды из базы данных с помощью Team.objects.all(). Создает сериализатор TeamSerializer для сериализации данных всех команд. Возвращает ответ с данными всех команд в виде списка. В случае ошибок возвращаются соответствующие сообщения об ошибках и статус 400.

**TeamUsersView.** Этот класс представления позволяет получить список пользователей, принадлежащих к определенной команде, в формате JSON. В случае успешного выполнения запроса возвращается список пользователей, а если команда не найдена, возвращается сообщение об ошибке.

Метод get(self, request, team\_id) принимает параметр team\_id, который представляет идентификатор команды, для которой нужно получить пользователей, пытается найти всех пользователей, принадлежащих к указанной команде, с помощью фильтрации объектов UserProfile по team\_id. Полученные данные о пользователях содержат их идентификатор, имя пользователя и email. Преобразует полученные данные в список и возвращает их в формате JSON с помощью JsonResponse. Если команда с указанным team\_id не найдена, возвращается сообщение об ошибке с кодом состояния 404.

* + - 1. Эндпоинты

Эндпоинты (endpoint) относятся к конечным точкам API, к которым клиенты могут обращаться для выполнения определенных операций или получения определенных данных. Эндпоинты в API определяют доступные операции, такие как GET, POST, PUT, DELETE. Эндпоинты описываются в Swagger с помощью специального формата, который содержит информацию о доступных эндпоинтах, методах запроса, параметрах, структуре данных и других деталях API.

Swagger — это набор инструментов для проектирования, создания, документирования и потребления веб-сервисов RESTful API. Он позволяет разработчикам описывать структуру и функционал API с помощью JSON или YAML файлов.

[TBD] Уточнить как swagger впихнуть

* + 1. Frontend

Фронт веб-приложения отвечает за то, как пользователи видят и взаимодействуют с интерфейсом. Он представляет собой то, что пользователь видит на экране и как он взаимодействует с элементами интерфейса.

Маршруты веб-приложения определяют, какой компонент должен быть отображен при определенном URL-адресе. Когда пользователь переходит по определенному URL, маршрутизатор определяет, какой компонент должен быть загружен и отображен на экране.

Компоненты веб-приложения представляют собой отдельные части интерфейса, которые могут быть загружены и отображены на экране. Они содержат логику и представление, необходимые для работы с определенными данными или функциональностью.

Маршруты определяют, какие компоненты должны быть загружены и отображены на экране в зависимости от URL-адреса, который пользователь посещает.

* + - 1. Маршруты

В разрабатываемом веб-сервисе определены маршруты для навигации пользователей по разным разделам приложения.

При входе на главную страницу по умолчанию пользователи попадают на адрес “/”. На этой странице пользователи могут увидеть основное содержимое приложения, приветственное сообщение.

По адресу “/about” представлена страница “О нас”, содержащая дополнительную информацию о сервисе, его функциональности.

“/photos” – страница “Фотографии” на которой пользователи могут просматривать, загружать, сегментировать свои изображения.

Адрес “/login” содержит страницу входа. Здесь пользователи могут войти в систему, введя свои учетные данные.

Страница регистрации расположена по адресу “/register”. На этой странице пользователи могут зарегистрировать новую учетную запись в сервисе.

Завершить сеанс работы и выйти из учетной записи с сервисом пользователь может, перейдя на страницу выхода по адресу “/logout”.

На странице “/profile” пользователи могут управлять своей учетной записью и просматривать персонализированную информацию.

“/team” – адрес, по которому расположена страница команд, на которой пользователи могут работать с командой в сервисе и управлять участниками.

* + - 1. Компоненты

[TBD]

* 1. Интерфейс приложения

[TBD]

* 1. Диаграммы

На рисунке 15 представлена ER диаграмма. [TBD]

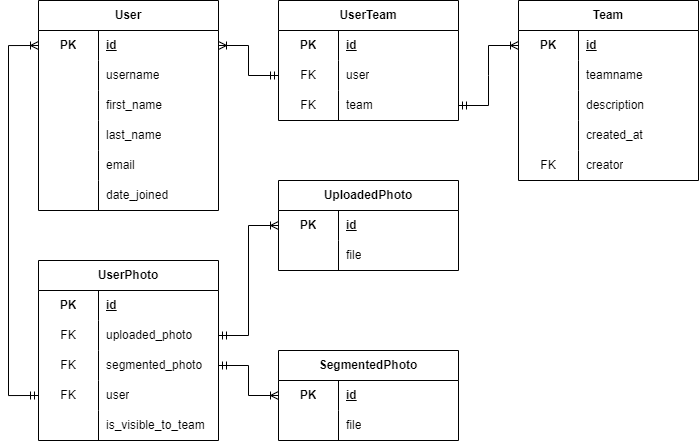


Рисунок 15 – ER-диаграмма

ЗАКЛЮЧЕНИЕ

В ходе исследования были рассмотрены различные методы сегментации объектов на изображениях, изучены архитектуры нейронных сетей, применяемых для этой задачи, и разработана эффективная модель сегментации объектов. Обучение модели с использованием выбранных методов позволило достичь высокой точности распознавания объектов на спутниковых снимках. Выбранные метрики оценки качества модели подтвердили ее надежность и точность.

Также были изучены различные архитектуры нейронных сетей, применяемых для сегментации спутниковых снимков. В работе были рассмотрены такие архитектуры, как U-Net, Mask R-CNN и SegNet. Каждая из этих архитектур имеет свои особенности и преимущества в контексте сегментации спутниковых снимков.

Особое внимание было уделено разработке пользовательского интерфейса веб-сервиса, обеспечивающего удобство работы с моделью сегментации объектов. Этот инструмент позволит пользователям легко загружать изображения, выполнять сегментацию объектов и получать результаты в удобном формате.В работе были рассмотрены различные типы задач сегментации спутниковых снимков. Каждая из этих задач требует специфических методов и подходов для достижения точной и качественной сегментации

В заключение сегментация спутниковых снимков является важным инструментом для анализа и интерпретации данных, полученных с помощью спутниковых систем. Выбор подходящего метода и архитектуры для сегментации зависит от конкретной задачи, типа спутниковых снимков и требований к производительности. Понимание различных методов и архитектур помогает в выборе оптимального подхода и достижении высокой точности и качества сегментации спутниковых снимков.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Годунов А. И., Баланян 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.
2. Хаустов П.А. Алгоритмы распознавания рукописных символов на основе построения структурных моделей // КО., 2017. № 1.
3. Янковский А.А., Бугрий А.Н. Критерии выбора метода бинаризации при обработке изображений лабораторных анализов // АСУ и приборы автоматики., 2010. № 153.
4. *N. Otsu.* A threshold selection method from gray-level histograms (англ.) // IEEE Trans. Sys., Man., Cyber. : journal. — 1979. — Vol. 9. — P. 62—66
5. Can Eyupoglu Implementation of Bernsen’s locally adaptive binarization method for gray scale images [Текст] / Can Eyupoglu // The Online Journal of Science and Technology. — 2017. — № Volume 7, Issue 2. — С. 68-72.
6. Павел Глек U-Net: нейросеть для сегментации изображений / Павел Глек [Электронный ресурс] // Neurohive : [сайт]. — URL: https://neurohive.io/ru/vidy-nejrosetej/u-net-image-segmentation/ (дата обращения: 20.12.2023).
7. U-Net Architecture Explained / [Электронный ресурс] // GeeksForGeeks : [сайт]. — URL: https://www.geeksforgeeks.org/u-net-architecture-explained/ (дата обращения: 20.12.2023).
8. Elisha Odemakinde Everything about Mask R-CNN: A Beginner’s Guide Read more at: https://viso.ai/deep-learning/mask-r-cnn/ / Elisha Odemakinde [Электронный ресурс] // viso.ai : [сайт]. — URL: https://viso.ai/deep-learning/mask-r-cnn/ (дата обращения: 20.12.2023).
9. Vijay Badrinarayanan, Alex Kendall, Roberto Cipolla SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation [Текст] / Vijay Badrinarayanan, Alex Kendall, Roberto Cipolla — 2016.
10. Д.В. Лукашик Анализ современных методов сегментации изображений [Текст] / Д.В. Лукашик // ЭКОНОМИКА И КАЧЕСТВО СИСТЕМ СВЯЗИ. — 2022. — № 2. — С. 57-65.
11. SQLite Documentation // SQLite URL: https://www.sqlite.org/docs.html (дата обращения: 27.04.2024).
12. Django Documentation // Django URL: https://docs.djangoproject.com/en/5.0/ (дата обращения: 27.04.2024).