Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение
высшего образования
«Уральский федеральный университет
имени первого Президента России Б.Н. Ельцина»

Институт радиоэлектроники и информационных технологий – РТФ

Школа бакалавриата

ОТЧЕТ

По проекту
«Разработка модели машинного обучения для сегментации КТ-снимков печени»

по дисциплине «Проектный практикум»

Заказчик: Фамилия И.О. Ильинский Александр Дмитриевич

Куратор: Фамилия И.О. Ильинский Александр Дмитриевич
ученая степень, ученое звание, должность

Студенты команды Path2Discoveries Раков Дмитрий Владимирович
Фамилия И.О.

Фамилия И.О. Яцук Владислав Романович

Екатеринбург, 2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

1 Цели и задачи

Цель проекта: разработка программного решения на основе методов машинного обучения для автоматизированной сегментации печени на КТ-снимках пациентов. С интеграцией модели в сервис для практического использования в клинике, включая инструменты ручной коррекции сегментации врачом и построения 3D-модели сегментированной печени.

Задачи проекта:

1. Провести анализ и предобработку найденного массива КТ-данных (https://zenodo.org/records/3431873);
2. Построить архитектуру нейронной сети (на основе U-Net) с учётом специфики медицинских изображений, провести обучение модели для повышения точности распознавания анатомических границ;
3. Реализовать валидацию модели с использованием метрики Dice Scor;
4. Разработать веб-сервис для инференса модели, интегрировав интерфейс для визуализации сегментации ручного редактирования контуров;
5. Обеспечить контейнеризацию решения через Docker для кроссплатформенного развёртывания в инфраструктуре заказчика;
6. Проектирование модуля генерации 3D-реконструкций печени на основе сегментированных срезов.

2 Актуальность и важность проекта

Автоматизация сегментации печени на КТ-снимках является важной задачей в современной медицине, где рост объёмов диагностических данных и высокая нагрузка на врачей-рентгенологов создают потребность в

оптимизации рутинных процессов. Ручная разметка контуров органов требует значительного времени. Внедрение ИИ-моделей позволяет сократить время обработки от секунды на 1 снимок и менее (исходя из «железа» на сервере), высвобождая ресурсы специалистов для интерпретации результатов и принятия клинических решений.

Актуальность проекта подтверждается прямым запросом от медицинского учреждения, что свидетельствует о наличии конкретной проблемы в практике заказчика.

Однако полная автоматизация невозможна: алгоритмы машинного обучения, даже достигшие высокой точности, не исключают необходимости врачебного контроля. Интеграция в сервис функции ручного редактирования контуров обеспечивает синергию между скоростью ИИ и экспертизой медиков.

Реализация соответствует глобальным трендам цифровизации здравоохранения и отвечает требованиям клинической практики, где технологические решения должны быть не заменой, а инструментом в руках специалистов.

3 Область применения продукта

Продукт ориентирован на медицинские учреждения (диагностические центры, онкологические клиники, хирургические отделения), где востребованы цифровые инструменты для работы с визуальными данными

4 Ожидаемые результаты

По итогам реализации проекта будут достигнуты следующие результаты, соответствующие уровням сложности (минимальный, базовый, оптимальный):

1. **Адаптированная модель машинного обучения**

a. Модель семантической сегментации (на базе U-Net), с метрикой Dice Score ≥ 0.90;
b. Поддержка обработки КТ-срезов в формате DICOM, интегрированная в конвейер предсказаний.

2. **Программный сервис для клинического использования**
   a. Веб-интерфейс, позволяющий загружать КТ-снимки, получать автоматическую сегментацию печени и визуализировать маску с наложением на исходное изображение;
   b. Инструмент ручного редактирования контуров: врачи смогут корректировать границы сегментации через интерактивный интерфейс;
   c. Функционал генерации 3D-модели печени в формате STL на основе сегментированных срезов, интегрированный в сервис.
3. **Докеризированное решение**
   a. Готовый Docker-контейнер с развёртыванием модели и зависимостями, обеспечивающий кроссплатформенную работу в изолированной среде.

ОСНОВНАЯ ЧАСТЬ

1 Разбор требований заказчика и пользователей к программному продукту

Функциональные:

Разработка модели машинного обучения для точной автоматической сегментации печени на КТ-снимках (DICOM/NIfTI). Интеграция модели в веб-сервис с возможностью ручной коррекции контуров врачом. Поддержка 3D-реконструкции печени на основе сегментированных данных. Контейнеризация решения через Docker для упрощения развёртывания в инфраструктуре клиники. Интуитивный интерфейс**:** возможность быстро загружать снимки, просматривать результаты сегментации, вносить правки. Визуализация: наложение масок сегментации на исходные изображения. Экспорт данных: сохранение результатов в форматах PNG, STL (для 3D-моделей).

Нефункциональные:

Точность модели: пороговые значения метрик (Dice Score ≥0.9) для клинической применимости. Производительность: время обработки одного среза ≤1 сек. Совместимость: поддержка стандартных форматов медицинских изображений (DICOM). Безопасность: обеспечение конфиденциальности данных пациентов (обработка на локальных серверах, без передачи в облако).

Бизнес-требования:

Сокращение времени анализа снимков по сравнению с ручной обработкой.

2 Составление плана действий для достижения цели

Подготовительный этап (2 недели)

Анализ готового решения по сегментации на GitHub, предоставленного заказчиком. Разделение задач. Нахождение набора данных для обучения модели или дообучения существующей. Обсуждение: какие инструменты будут использоваться для достижения цели (библиотека для обучения, бэкенда). Составление плана: сколько каждый этап займет времени для реализации. Анализ аналогов.

Разработка модели ML (5 недель)

Тестирование базовых решений, написание кастомного датасета, процесса обучения и валидации. Подбор гиперпараметров (batch size, оптимизатор и тд). Визуализация. Тестирование.

Разработка сервиса (4 недели)

Реализация бэкенда и фронтенда на Django [1]. Загрузка снимков с возможностью получить сегментацию, изменить её и просмотреть / скачать 3д-модель. Сборка Docker-образа. Тестирование.

Тестирование (на процессе, когда есть какой-то результат)

3 Анализ и сопоставление аналогов разрабатываемого продукта

Для анализа возможностей применения компьютерного зрения в медицине была создана MindMap (https://miro.com/app/board/uXjVKOB13ow=/). В этой карте отражены различные области, такие как радиология, патология, офтальмология и дерматология.

4 Архитектура продукта

Программный продукт представляет собой веб-приложение на Django, объединяющее:

1. **Backend** – обработка КТ-снимков с помощью модели U-Net (PyTorch [2]).

2. **Frontend** – интерфейс для загрузки изображений, визуализации сегментации и ручного редактирования масок.
3. **База данных** – хранение загруженных снимков, масок, правок пользователей, 3Д-моделей.

Архитектура монолитная (без REST API), так как:

- Простота разработки и развертывания (единая кодовая база).
- Нет требований к микросервисности или интеграции с внешними системами.
- Достаточно для MVP с фокусом на функциональность, а не масштабируемость.

- **Django-сервер**
    - Обрабатывает HTTP-запросы (загрузка/сохранение снимков).
    - Управляет пользовательскими сессиями
    - Взаимодействует с БД (SQLite).
- **Модель U-Net (PyTorch)**
    - Загружается при старте сервера (веса модели хранятся в файле .pth).
    - Принимает преобразованные КТ-срезы (после предобработки), возвращает маску сегментации.
- **Схема хранения**:
    - LiverCTScan: сессия, оригинально изображение, оригинальное изображение png, сегментация модели, сегментация пользователя, статус сегментации, нужна ли 3Д-модель, id (номер) загрузки.
    - Liver3d: сессия, id загрузки, 3Д-модель

1. Пользователь загружает КТ-снимок через форму → Django сохраняет файл в БД.

2. Django передаёт срез модели U-Net → получает маску → сохраняет её в БД.
3. Frontend отображает маску поверх снимка (через Cornerstone.js).
4. Врач правит маску → изменения сохраняются в БД.
5. Врач получает 3Д-модель печени.

Альтернативы и компромиссы:

- **REST API (Django REST Framework)** – не выбран, так как нет потребности в SPA или мобильном клиенте.
- **TensorFlow/Keras** – PyTorch предпочтительнее для исследований и быстрого прототипирования.
- **React** – усложнил бы развёртывание, но дал бы более отзывчивый интерфейс (не критично для MVP).

Выбранная архитектура обеспечивает быстрое внедрение и покрытие всех требований заказчика. В будущем возможен переход на микросервисы (например, вынос модели в отдельный сервис с REST API).

5 Процесс разработки

5.1 Методология разработки

Для решения задачи сегментации печени на КТ-снимках использовалась архитектура **UNET** [3] (Приложение А) – сверточная нейронная сеть, хорошо зарекомендовавшая себя в медицинской визуализации.

Основные этапы:

- **Предобработка данных**: нормализация, обрезка, приведение к единому размеру.
- **Аугментация**: случайные повороты, отражения, изменение яркости для увеличения разнообразия данных.

- **Обучение**: двухэтапный подход – сначала высокий learning rate для быстрой сходимости, затем уменьшение для точной настройки.
- **Валидация**: мониторинг метрик (**Dice Coefficient**, **Loss**) на тестовой выборке.

5.2 Проблемы в процессе разработки

На начальном этапе не удавалось корректно сохранять и загружать веса модели. Но это было не критично, потому что решили писать новую модель на Torch.

5.3 Тестирование различных ML-подходов

Перед финальным выбором UNET тестировались:

- **Другие архитектуры**: DeepLabV3+ – показали худшую точность на малых наборах данных, причем обучалась кратно дольше
- **Разные функции потерь**:
  - **Binary Cross-Entropy** – давала зашумленные предсказания.
  - **Dice Loss** – улучшила четкость границ сегментации.

5.4 Результаты тестирования на промежуточных этапах

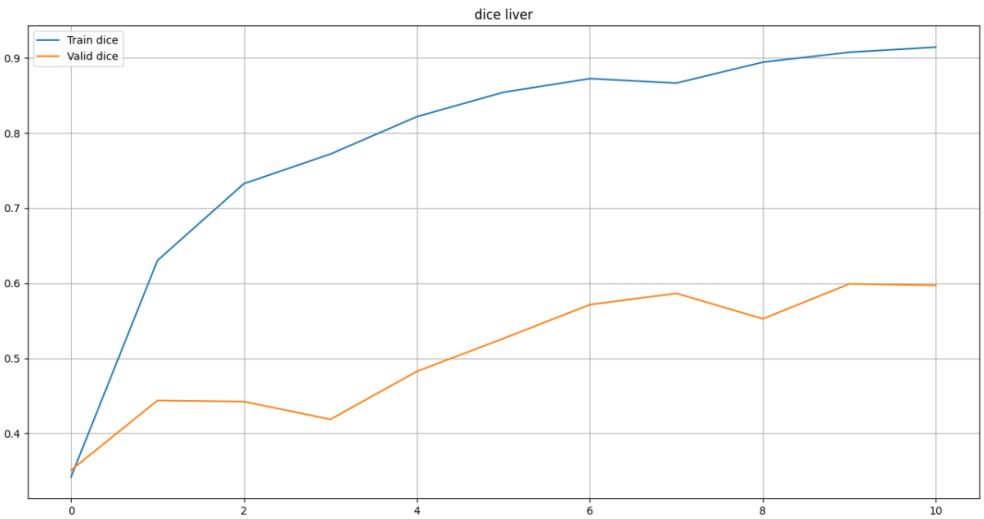

Рисунок 1 – Результат обучения 1

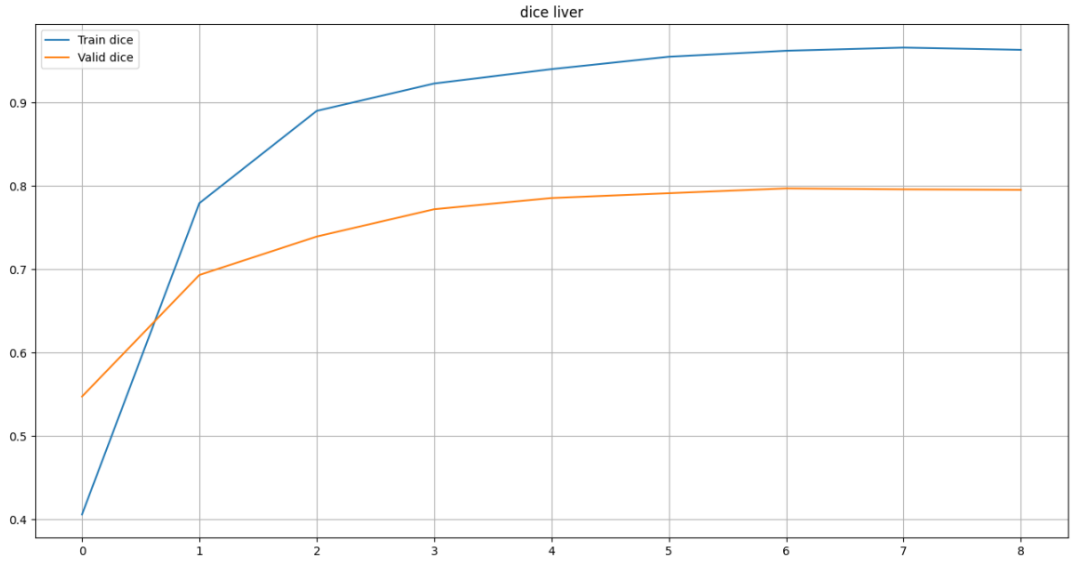

Рисунок 2 – Результат обучения 2

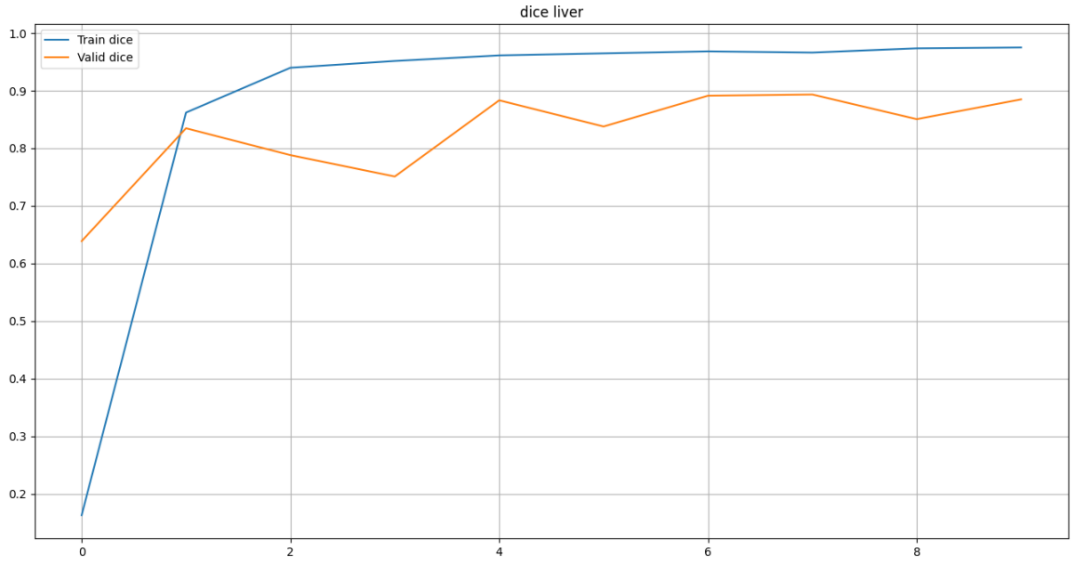

Рисунок 3 – Результат обучения 3

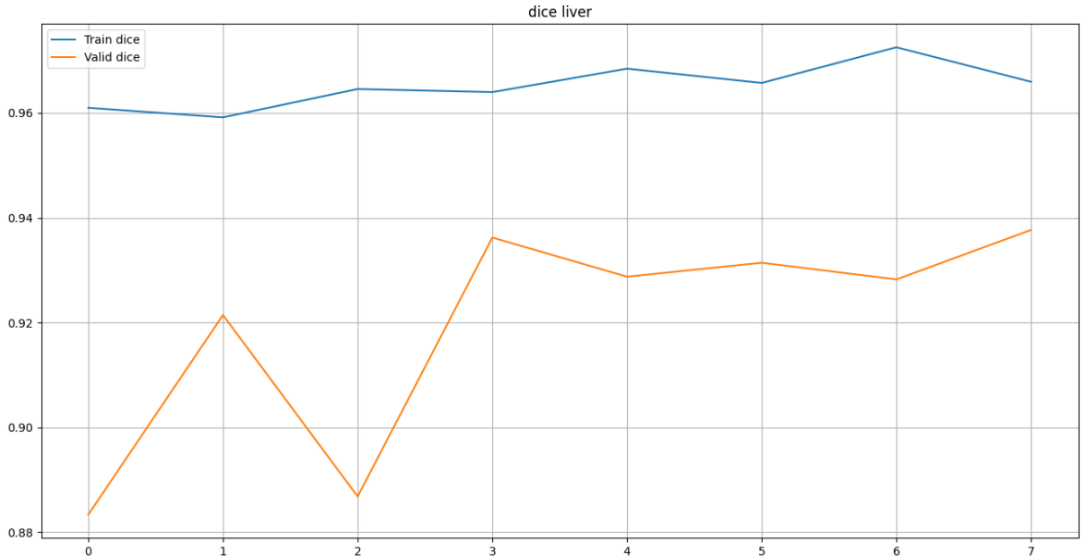

Рисунок 4 – Результат обучения 4.1

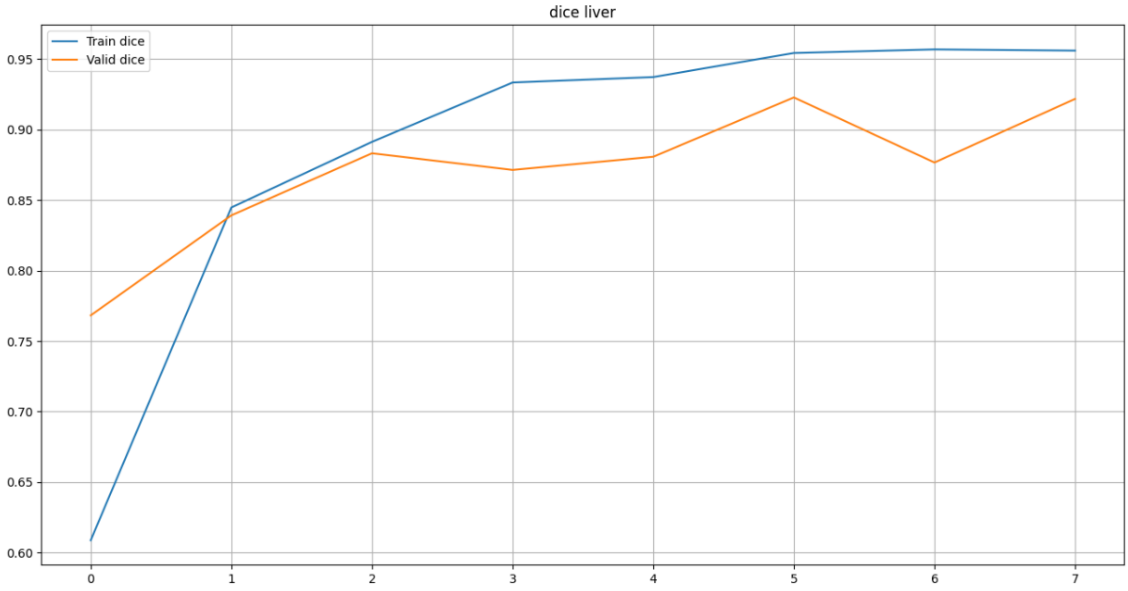

Рисунок 5 – Результат обучения 4.2

Приведены не все тесты, а только те, которые дали кардинальный прирост результата.

5.5 Разбор ключевых ошибок

Переобучение на ранних этапах. Причина: ненастроенные гиперпараметры UNET

5.6 Результат

После настройки гиперпараметров и исправления ошибок модель достигла **Dice Coef 0.94**, что подтверждает её пригодность для клинического применения.

6 Работа каждого участника

6.1 Раков Дмитрий Владимирович

Моя работа была сосредоточена на задачах, связанных с нейросетевыми алгоритмами, обработкой контуров печени и созданием 3D-моделей на основе данных сегментации.

В начале семестра наш куратор предоставил доступ к репозиторию на GitHub, где находились исходные наработки по нейросетевой модели. Детально изучили предложенный код, запустил его на своем устройстве и провел первичное тестирование. При первом запуске возникли технические сложности: скрипт выдавал ошибки, а также отсутствовала возможность сохранения весов обученной модели для последующего применения. Мы исправили обнаруженные недочёты и углубились в анализ кода, чтобы лучше понимать его логику, а не воспринимать его как «чёрный ящик».

Следующим этапом стала миграция проекта с Keras на PyTorch, так как этот фреймворк предоставляет больше гибкости и удобства в работе. Кроме того, в процессе обучения модели на Keras возникали определённые трудности – их можно было преодолеть, но PyTorch оказался более предпочтительным вариантом благодаря моему предыдущему опыту работы с ним.

Для обучения модели применялась аугментация данных. Исходные изображения печени имели разрешение 512×512 пикселей. Мы использовали Dataset, предоставленный куратором. Разделение данных на обучающую и

тестовую выборки проводилось в пропорции 70% / 30% без предварительного перемешивания сканов – это было важно для исключения «утечки» данных и корректной валидации. На основе тренировочной выборки вычислялись mean и std для последующей нормализации. Размер батча составил 4.

В качестве архитектуры нейросети была выбрана стандартная UNET с глубиной свёрток, равной пяти (downsample), и базовым количеством каналов – 4. Для оптимизации использовалась функция потерь BCEWithLogitsLoss, оптимизатор Adam и планировщик обучения ExponentialLR. Экспериментировали и с другими подходами, но наилучший результат (DiceCoef = 0.94 на тестовой выборке) показала именно эта конфигурация.

Рефлексия: этот проект стал для меня важным этапом профессионального роста. Он не только позволил углубить понимание обработки медицинских изображений, но и дал ценный опыт командной работы. Эмоции от проделанной работы остаются позитивными: несмотря на сложные задачи, мы смогли достичь всех поставленных целей, и каждый участник команды внёс значимый вклад.

Лично для меня проект оказался полезен в нескольких аспектах:

- **Технические навыки** – я улучшил свои умения в работе с нейросетевыми архитектурами, аугментацией данных и переносом моделей между фреймворками.
- **Командное взаимодействие** – научился эффективнее распределять задачи, учитывать сильные стороны коллег и находить компромиссы в спорных моментах.
- **Организация работы** – получил практический опыт в соблюдении дедлайнов, итеративном улучшении продукта и анализе результатов.

Уверен, что приобретённые знания и навыки пригодятся в будущих проектах и помогут мне развиваться в сфере машинного обучения и анализа данных. Кроме того, этот опыт укрепил мою уверенность в том, что даже

сложные задачи можно решить благодаря слаженной работе команды и грамотному планированию.

6.2 Яцук Владислав Романович

Моя часть работы была посвящена разработке бэкенд-части приложения на Django и его контейнеризации с использованием Docker.

На начальном этапе я развернул Django-проект, настроил базовую структуру и интегрировал его с моделью машинного обучения, предоставленной коллегой. Основной задачей было обеспечение взаимодействия между фронтендом и ML-моделью: загрузка DICOM-файлов, их предобработка (нормализация, ресайз) и передача в U-Net для сегментации. Для хранения данных (путей, для удобства) использовалась SQLite.

Также важно было внедрить просмотр 3Д-модели печени в фронтенд интерфейс. Данная функция была реализована успешно, однако возникало много проблем из-за формата данных. Однако с этой проблемой я справился.

Для удобства развёртывания я настроил Docker-контейнер, включающий Django-сервер, SQLite и зависимости для работы PyTorch. Особое внимание уделил уменьшению размера образа: использовались многоэтапные сборки и легковесные базовые образы (Python-slim). Также были написаны инструкции по запуску контейнера и настройке переменных окружения.

В итоге бэкенд обеспечил:

- Загрузку и отображение DICOM-снимков.
- Сохранение результатов сегментации с возможностью ручной коррекции.
- Стабильную работу в Docker на разных ОС.

Рефлексия: этот проект стал для меня важным шагом в развитии как backend-разработчика. Работа над медицинским сервисом позволила не

только улучшить технические навыки, но и получить ценный опыт интеграции различных компонентов в единую систему. Научился четко формулировать требования к ML-части (форматы данных, метрики), что ускорило интеграцию модели. Осознал, как критично тестировать каждую часть системы отдельно – это помогло избежать «снежного кома» ошибок на финальных этапах.

ЗАКЛЮЧЕНИЕ

Программный продукт полностью соответствует требованиям заказчика и пользователя. Все критерии приемки выполнены, включая базовые и дополнительные условия (упаковка в сервис и Docker, реализовано построение 3Д-модели). Решение готово к внедрению в рабочую среду клиники.

Четких критериев для реализации не было, например, какой должен быть Dice Coef, какое ограничение по времени для сегментации и построения 3Д-моделей.

Решение выполняет все поставленные задачи от заказчика. Ошибки при работе не возникают. Высокий Dice Coef показывает, что сегментация выполняется корректно и при неточностях пользователь может изменить неверную сегментацию. При необходимости строится 3Д-модель, с чем не возникает проблем.

В качестве улучшения можно добиться более быстрой сегментации или 3Д-моделирования за счет нахождения более оптимальных решений. Создать API, облачные вычисления на видеокарте, что значительно ускорит процесс. Улучшение визуализации (фронтенда). Также перенос фронтенда на Rect, что повысит удобство и для пользователей, и для разработчиков так как будет более комфортно поддерживать и развивать проект.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Документация Django – URL: https://docs.djangoproject.com/en/5.2/

2. Документация PyTorch – URL: https://docs.pytorch.org/docs/stable/index.html

3. Ronneberger, O., Fischer, P. and Brox, T., 2015. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18* (pp. 234-241). Springer international publishing.

ПРИЛОЖЕНИЕ А

(обязательное)

Архитектура U-Net

U-Net – это популярная архитектура сверточных нейронных сетей, разработанная специально для задач семантической сегментации изображений. В отличие от классических CNN, которые классифицируют изображение целиком, U-Net генерирует pixel-wise маску, выделяя области, соответствующие разным классам.

Модель имеет **U-образную структуру**, состоящую из двух основных частей:

1. **Энкодера (сжимающий путь)** – извлекает признаки, постепенно уменьшая пространственное разрешение.
2. **Декодера (расширяющий путь)** – восстанавливает детализацию, комбинируя низкоуровневые и высокоуровневые признаки.

Каждый блок энкодера включает:

- Две свертки 3×3 с активацией ReLU.
- Операцию макс-пулинга (2×2) для уменьшения размерности.
- При переходе к декодеру:
- Применяется транспонированная свертка 2×2 для увеличения разрешения.
- Происходит конкатенация с соответствующими картами признаков из энкодера (skip-connections).
- Выполняются две свертки 3×3 с ReLU.

Финал архитектуры – свертка 1×1, преобразующая 64-канальный тензор в карту сегментации с числом каналов, равным количеству классов. Общее число обучаемых слоев – 23**.**

Ключевые преимущества U-Net:

- Эффективность даже при ограниченных данных (актуально для медицинской визуализации).
- Быстрая обработка (сегментация 512×512 за <1 сек на GPU).
- Победа в конкурсе ISBI 2015 по сегментации нейронных структур и трекингу клеток.

Графическое представление архитектуры приведено в Рисунок 6

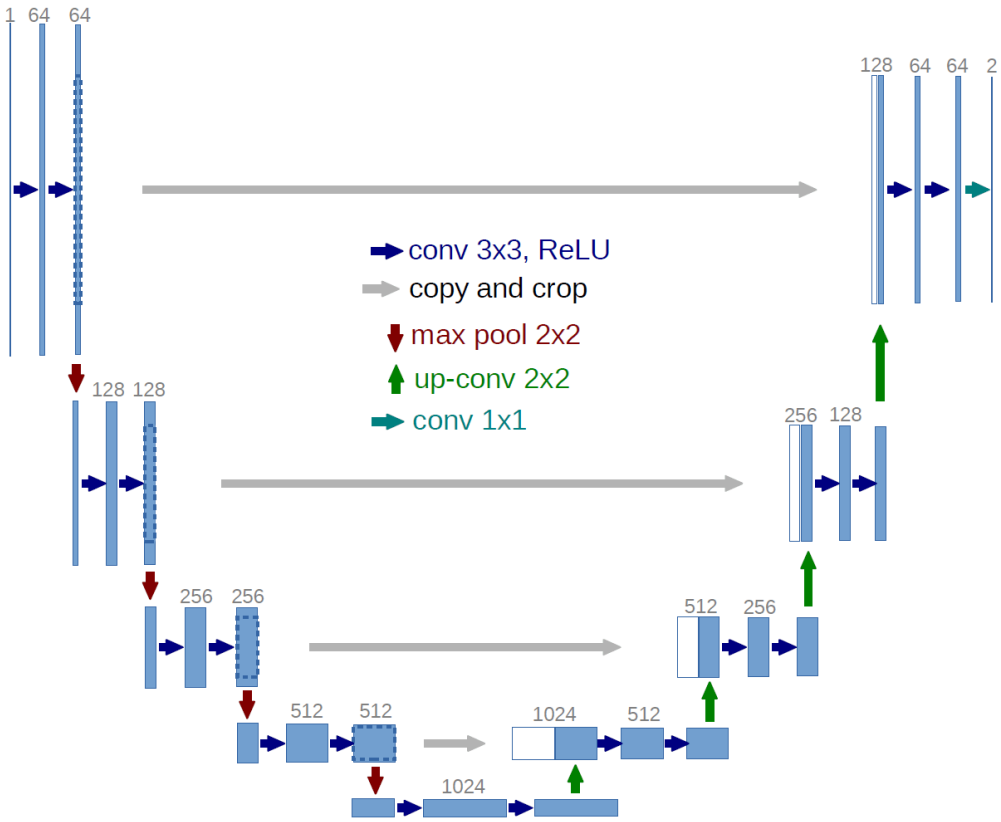

Рисунок 6 – UNET