Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное
учреждение высшего образования
Национальный исследовательский Нижегородский государственный
университет им. Н.И. Лобачевского

Институт информационных технологий, математики и механики

Отчет по лабораторной работе

«U-Net»

Выполнил:
студент группы 3823Б1Пмоп2
Золкин И.А.

Проверил:
Преподаватель каф. ТУиДС
Ростов Д. А.

Нижний Новгород
2025

Содержание

Введение

В области компьютерного зрения задача сегментации является одной из фундаментальных и практически значимых. Её решения находят применение в медицинской диагностике (анализ снимков МРТ, КТ), автономном вождении (выделение дорог, пешеходов), дистанционном зондировании Земли и многих других областях. Традиционные алгоритмы обработки изображений часто не справляются с вариативностью и сложностью реальных данных, что обуславливает необходимость использования глубокого обучения.

Основная сложность в сегментации заключается в необходимости совмещать два, казалось бы, противоречивых требования:

1) Общее понимание изображения на глобальном уровне для корректной классификации объектов.

2) Точное локализованное соответствие на уровне пикселей для формирования чётких границ объектов.

Стандартные свёрточные сети, успешные в классификации, теряют пространственную детализацию из-за последовательных операций пулинга.

В 2015 году Олаф Роннебергер, Филлип Фишер и Томас Брокс предложили архитектуру U-Net, которая стала прорывным подходом, эффективно решающим указанную проблему.

Благодаря своей U-образной структуре архитектура стала способная комбинировать детализированную информацию из ранних слоев с высокоуровневыми признаками из глубоких слоёв, что обеспечило точное позиционирование границ объектов.

Целью данной работы является теоретическое и практическое исследование архитектуры нейронной сети U-Net.

Для достижения цели требуется выполнить следующие задачи:

1) Изучить принципы работы, детали архитектуры и математический аппарат, лежащий в основе UNet.
2) Реализовать модель UNet на фреймворке глубоко обучения PyTorch, PyTorch_Lightning, настроить логгирование с использованием ML-Flow.
3) Обучить модель на специализированном наборе для сегментации.
4) Привести оценку результатов сегментации с использованием стандартных метрик, таких как Dice Coefficient, IoU - Intersection over Union.
5) Проанализировать влияние ключевых компонентов архитектуры на конечное качество модели.

Описание датасета

В процессе исследования модели использовался датасет для сегментации людей **Human Segmentation Dataset - Supervise.ly** [1], содержащий 2667 изображений с соответствующими бинарными масками и коллажами высокого разрешения.
В корневой директории датасета расположен файл df.csv и директории images, masks и collage.
В файле df.csv находится описание путей файлов датасета. Файл содержит 4 столбца: номер, пути до изображений (”images”), пути масок (”masks”), пути коллажей (”collages”), изображения в формате png.

	A	B	C	D	E	F	
1		images	masks	collages			
2	0	images/ds10_	masks/ds10_	collage/ds10_pexels-photo-687782.jpg			
3	1	images/ds10_	masks/ds10_	collage/ds10_pexels-photo-835971.jpg			
4	2	images/ds10_	masks/ds10_	collage/ds10_pexels-photo-850708.jpg			
5	3	images/ds10_	masks/ds10_	collage/ds10_pexels-photo-864937.jpg			
6	4	images/ds10_	masks/ds10_	collage/ds10_pexels-photo-865908.jpg			

Рис.1 содержание df.csv

Архитектура сети

Unet [2] - это сверточная нейронная сеть (Convolutional Neural Network, CNN) с U-образной симметрической архитектурой, специально разработанная для задач семантической сегментации биомедицинских изображений. UNet показывает хорошие результаты даже при работе с малым количеством данных.

Ключевая особенность UNet - способность сочетать контекстуальную информацию (что изображено) с точной пространственной локализацией (где именно находятся объекты), что является центральной проблемой в сегментации.

Архитектура UNet состоит из двух симметричных путей и моста:

1. Путь сжатия (Encoder) - предназначен для извлечения признаков и контекстной информации из изображения.

   Состоит из повторяющихся блоков, каждый из которых включает две свертки 3x3 с активацией ReLU и операцию макс-пулинга 2x2 с шагом 2 для понижения пространственной размерности.

   На каждом уровне сжатия количество карт признаков удваивается, о пространственные размеры уменьшаются вдвое, что позволяет сети изучать иерархические признаки - от простых границ и текстур к сложным объектам и сценам.

2. Путь расширения (Decoder) - предназначен для точного пространственного восстановления сегментационной маски.

   Состоит из симметричных блоков, каждый из которых включает операцию транспонированной свертки для увеличения пространственных размеров, конкатенацию с соответствующими картами признаков из пути сжатия через skip-connections и две свертки 3x3 с активацией ReLU.

   На каждом уровне расширения количество карт признаков уменьшается вдвое, а пространственные размеры увеличиваются, что позволяет постепенно восстанавливать детализированную сегментационную маску.

3. Мост (Bottleneck) - самый нижний слой архитектуры, соединяющий путь сжатия и путь расширения, содержит наиболее абстрактные и высокоуровневые признаки.

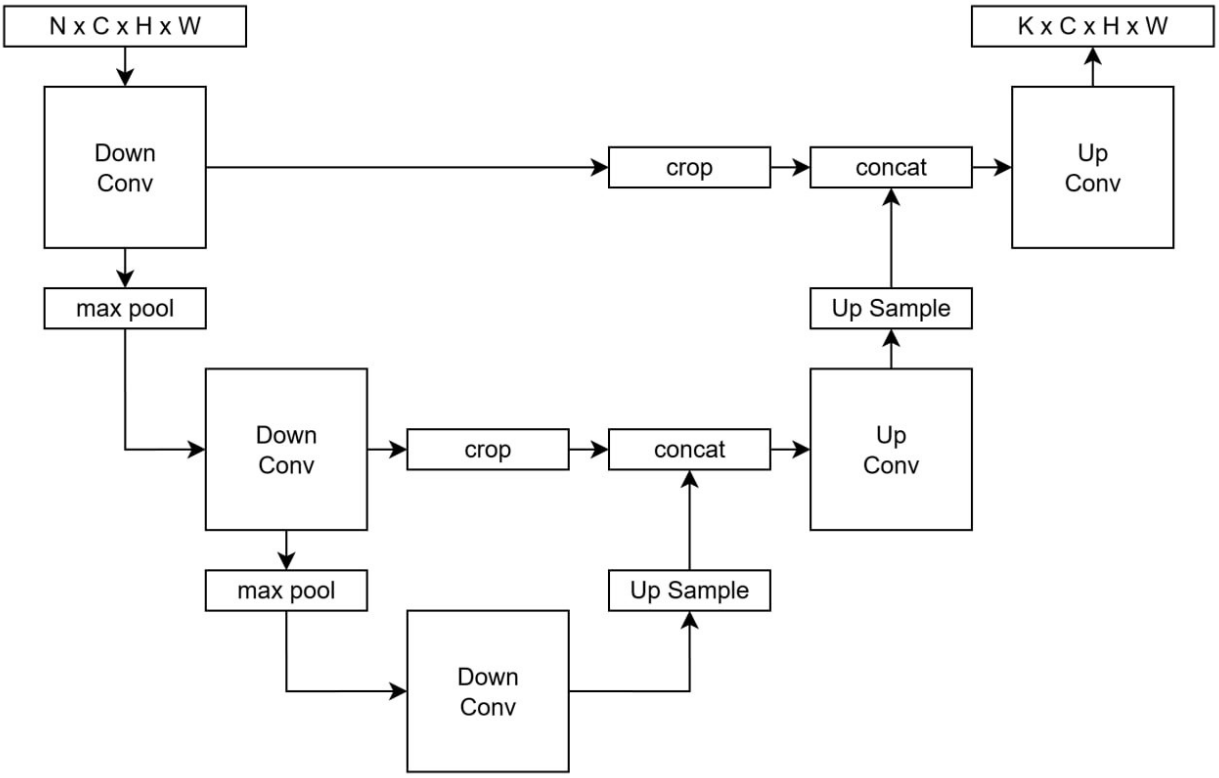

Рис. 2 Архитектура Unet.

Обработка данных

1) Загружаются пути к изображениям и маскам из df.csv, остальные два столбца в файле игнорируются.
2) Осуществляется разделение на обучающую, валидационную и тестовую выборки с использованием train_test_split из библиотеки scikit-learn. Соотношение: 20% - тестовая выборка, 10% - валидационная, 70% - обучающая.
3) Все изображения и соответствующие им маски приводятся к размеру 256x256 пикселей с использованием transform.Resize.
4) Изображения конвертируются в трёхканальные (RGB), маски - в одноканальные (grayscale).
5) Данные преобразуются в тензоры PyTorch с помощью transform.ToTensor.
6) Данные загружаются с использованием DataLoader с заданным размером батча = 4. Для обучающей выборки включено перемешивание (shuffle), для валидационной и тестовой - отключено. Используется многопоточная загрузка данных с закреплением памяти при наличии GPU.
7) Во время реальной работы изображение приводится к размеру 256x256, сеть применяется к изображению, результат преобразуется в вероятности, к которым применяется пороговое значение. Полученные вероятностные карты и бинарные маски масштабируются обратно к исходному размеру входного изображения: для вероятностных карт используется билинейная интерполяция (сглаживание градиентов), для бинарных масок используется интерполяция методом ближайшего соседа.

Обучение модели

В качестве функции потерь при обучении модели сегментации использовалась Binary Cross-Entropy with Logits Loss (nn.BCEWithLogitsLoss).

Данная функция потерь является эффективной для задач бинарной классификации и сегментации. Она минимизирует расхождение между предсказанными вероятностями и истинными метками на уровне отдельных пикселей.

Для ранней остановки обучения и мониторинга использовалась метрика Dice=(2 * TP) / (2 * TP + FP + FN).

В процессе обучения использовался оптимизатор Adam (Adaptive Moment Estimation) и планировщик скорости обучения ReduceLROnPlateau в режиме ‘min’, который уменьшает Learning rate вдвое по прошествии 5 эпох. Мониторинг происходит по val_loss.

Adam относится к классу адаптивных оптимизаторов первого порядка, который вычисляет индивидуальные адаптивные скорости обучения для каждого параметра модели. Он вычисляет скользящие средние как градиентов, так и квадратов градиентов, что позволяет быстро сходится в начале обучения, поддерживать стабильное обучения на поздних этапах, автоматически адаптировать скорость обучения для каждого параметра и эффективно работать с зашумленными градиентами.

Процесс обучения:

1) На каждом шаге вычисляется BCEWithLigitsLoss между предсказаниями модели и истинными масками.
2) На каждом шаге вычисляется Dice для текущего батча, а в конце эпохи вычисляется Dice для всей эпохи.
3) Максимальное количество эпох - 100, размер батча - 4.
4) На каждом шаге обучения логируются train_loss, train_dice. На каждом шаге валидации логируются val_loss, val_dice. В конце каждой эпохе логируются train_dice_epoch, val_dice_epoch.

Метрики

Для оценки качества модели сегментации были выбраны следующие метрики [3]:

Функция потерь: Binary Cross-Entropy (BCE) with Logits Loss - данная функция потерь соче тает операцию сигмойды и бинарную кросс-энтропию в одной стабильной с вычислительной точки зрения функции. Она измеряет расхождения между предсказанными вероятностями и истинными бинарными масками. Служит хорошей дифференцируемой функцией потерь для градиентного спуска и является стандартным выбором для задача бинарной сегментации. Она хорошо работает когда классы сбалансированы.

Dice = (2 * |XY|) / (|X| + |Y|), где X - предсказанная маска, Y - истинная маска. Основная метрика для задач сегментации. Непосредственно измеряет площадь перекрытия между предсказанием и истиной.

Выбрана в качестве основной метрики для сохранения лучших моделей из-за своей устойчивости к несбалансированным данным (когда фон доминирует над объектом).

Также для вычисления Dice на уровне эпохи используются бинарные статистические показатели:

True Positives (TP) - правильные положительные предсказания,

False Positives (FP) - ложные положительные предсказания,

True Negatives (TN) - правильные отрицательные предсказания,

False Negatives (FN) - ложные отрицательные предсказания,

Support (количество элементов).

Результаты

Обучение завершилось на 44-й эпохе из-за срабатывания раннего выхода по метрике Dice. Лучший результат по валидационной выборке был на 29й эпохе. В процессе обучения был заметен асимптотический паттерн роста метрики dice к единице для тренировочной выборки, что говорит, что модель всё лучше и лучше предсказывала данные на которых он тренируется. По этой же причине заметно уменьшение train_loss до 0.04.

В тоже время метрики валидационного набора уменьшались до 29-й эпохи включительно, а дальше принимали колебательный характер, ухудшаясь на некоторых последующих эпохах.

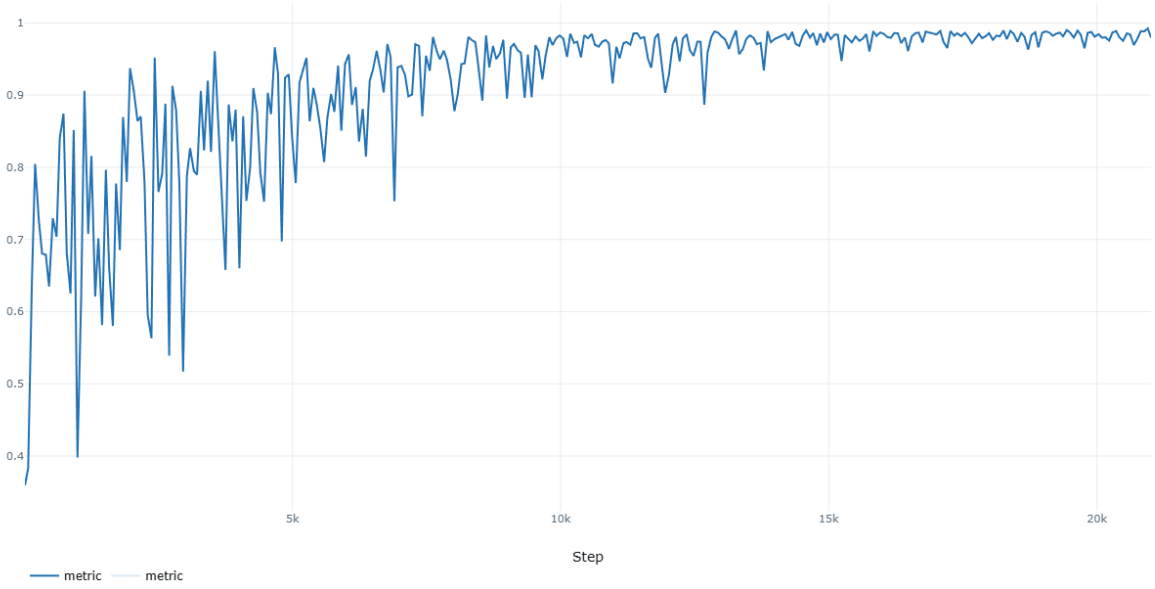

Рис. 3 train_dice - метрика dice пошагово для обучающей выборки.

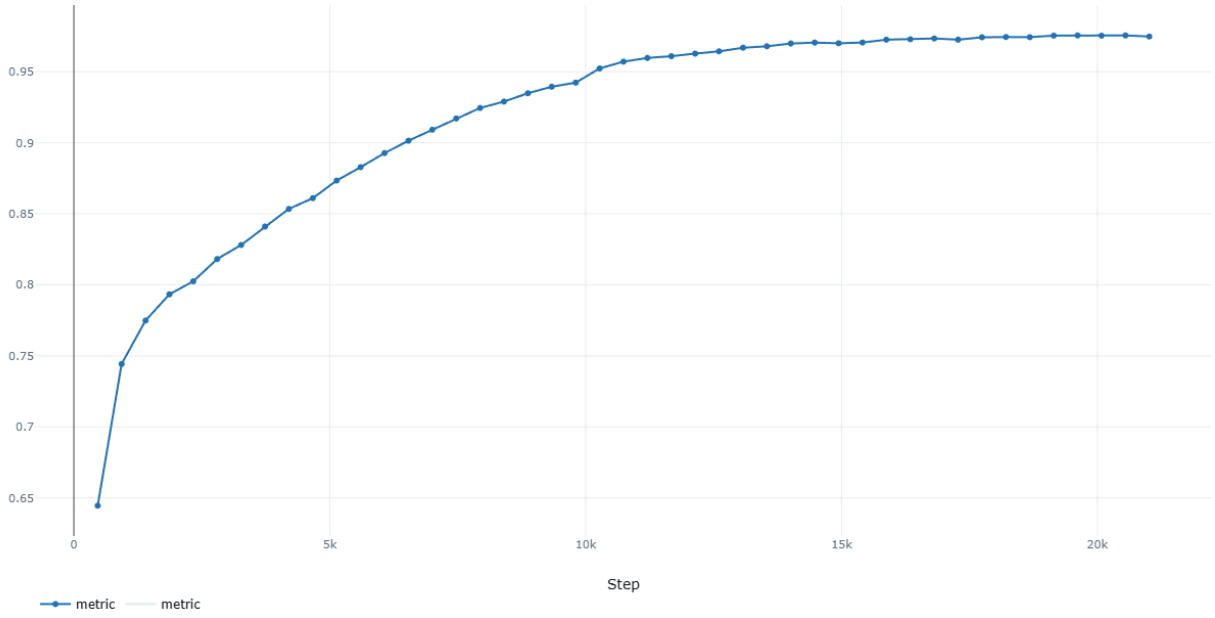

Рис. 4 train_dice_epoch - метрика Dice для каждой эпохи обучающей выборки.

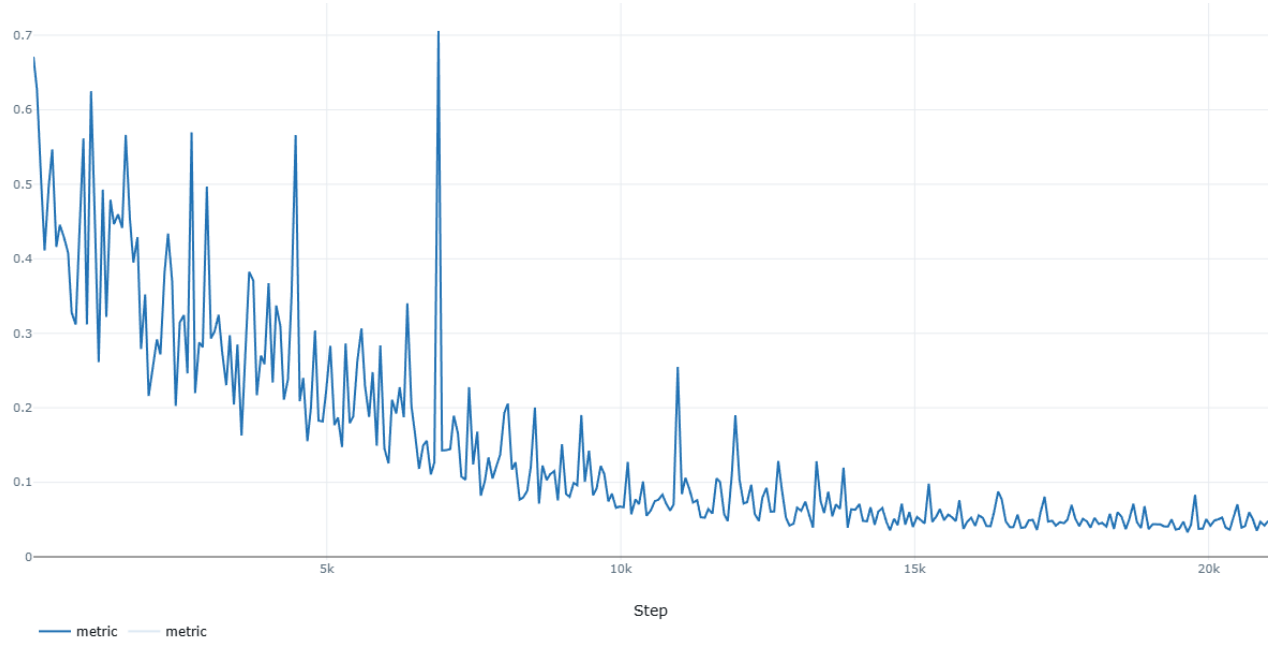

Рис. 5 train_loss для обучающей выборки.

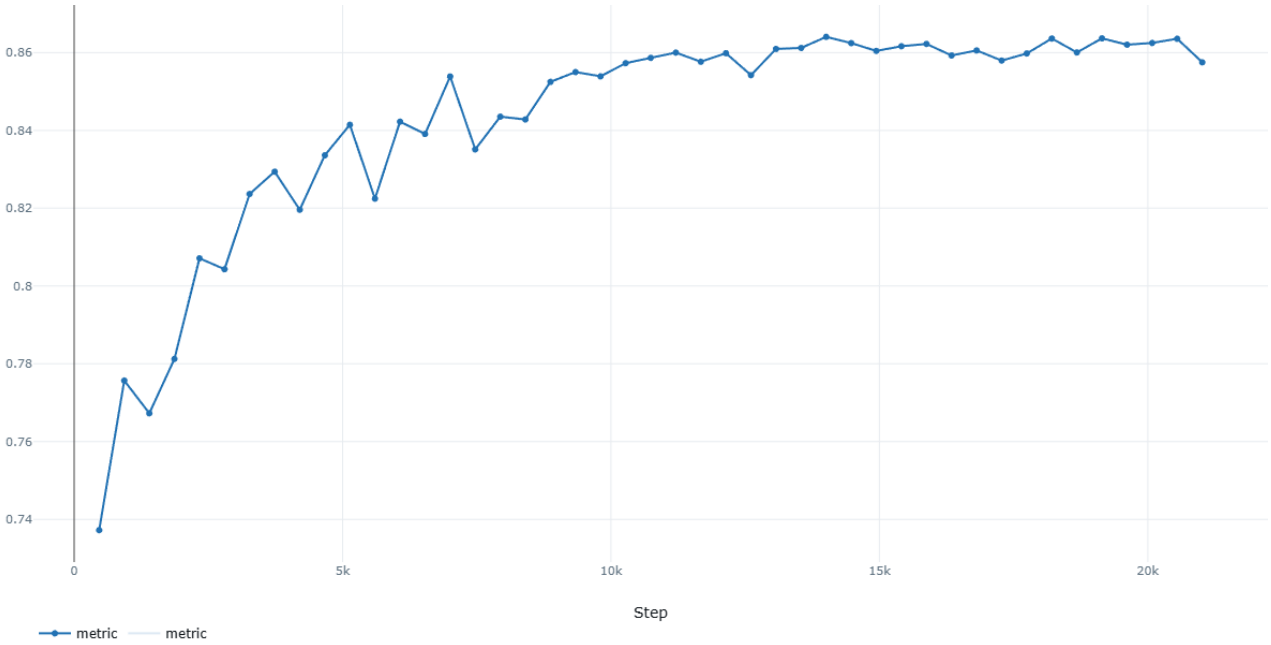

Рис. 6 val_dice_epoch - метрика Dice для каждой эпохи валидационной выборки.

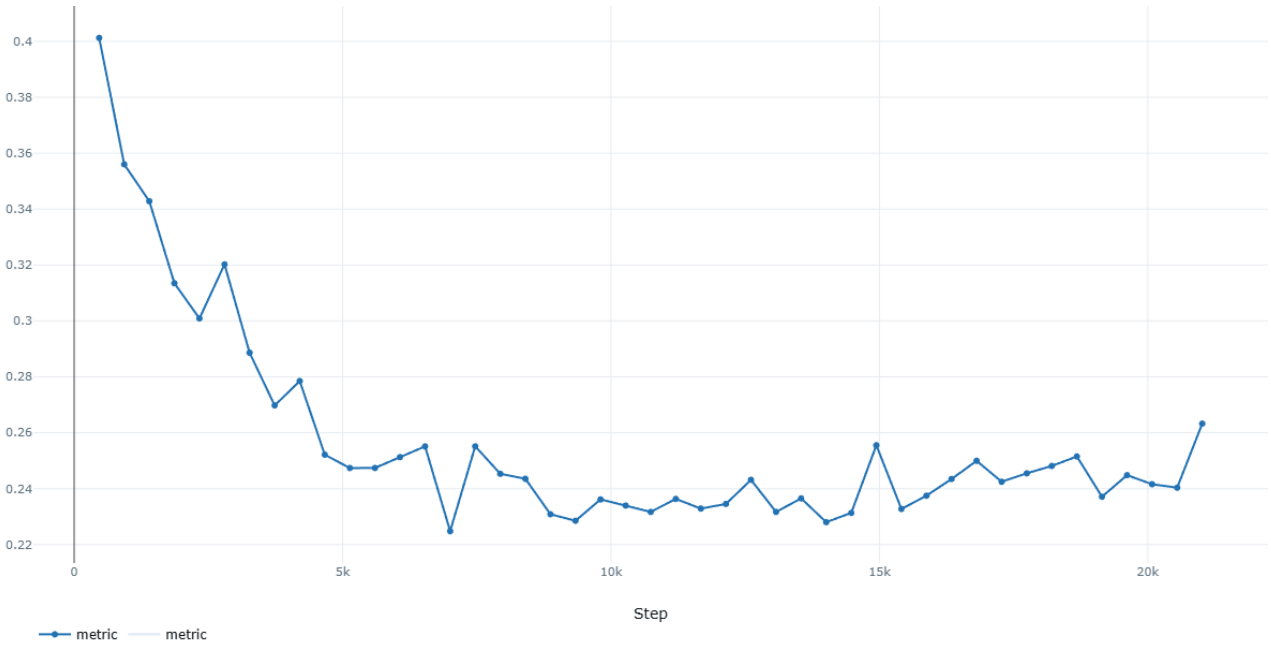

Рис. 7 val_loss валидационной выборки.

Проверка модели на тестовой выборке показала такие результаты:

Dice - 0.8556106686592102

Loss - 0.2595876157283783

Визуализация работы модели на случайном изображении из датасета:

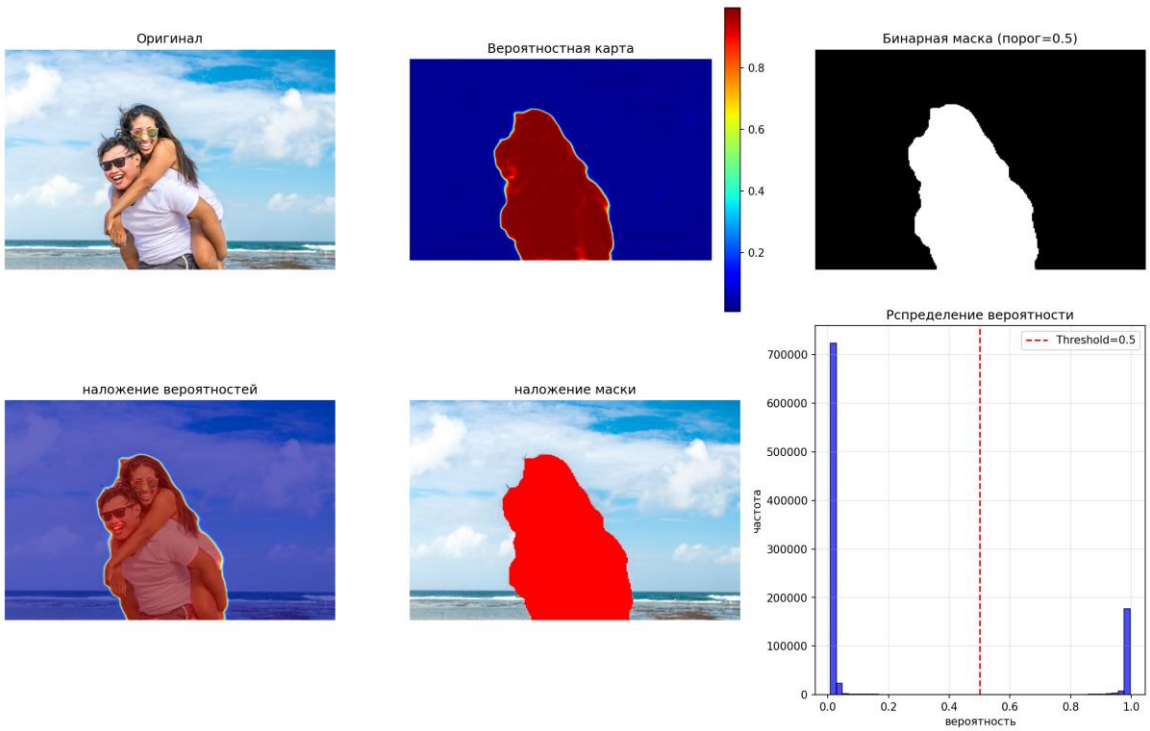

Рис. 8 - работа сети на случайном изображении из датасета.

Визуализация работы модели на случайном изображении не из датасета (из открытого доступа):

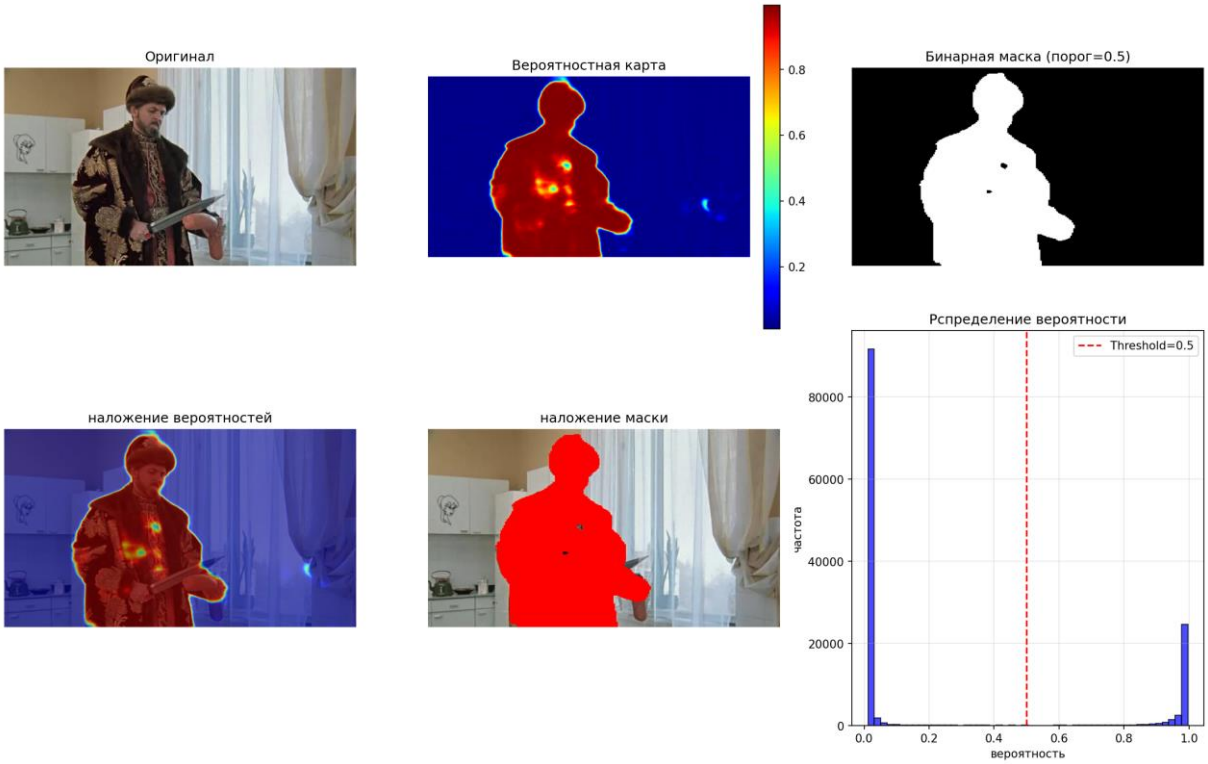

Рис. 9 - работа сети на случайном изображении из открытых источников (не присутствует в датасете)

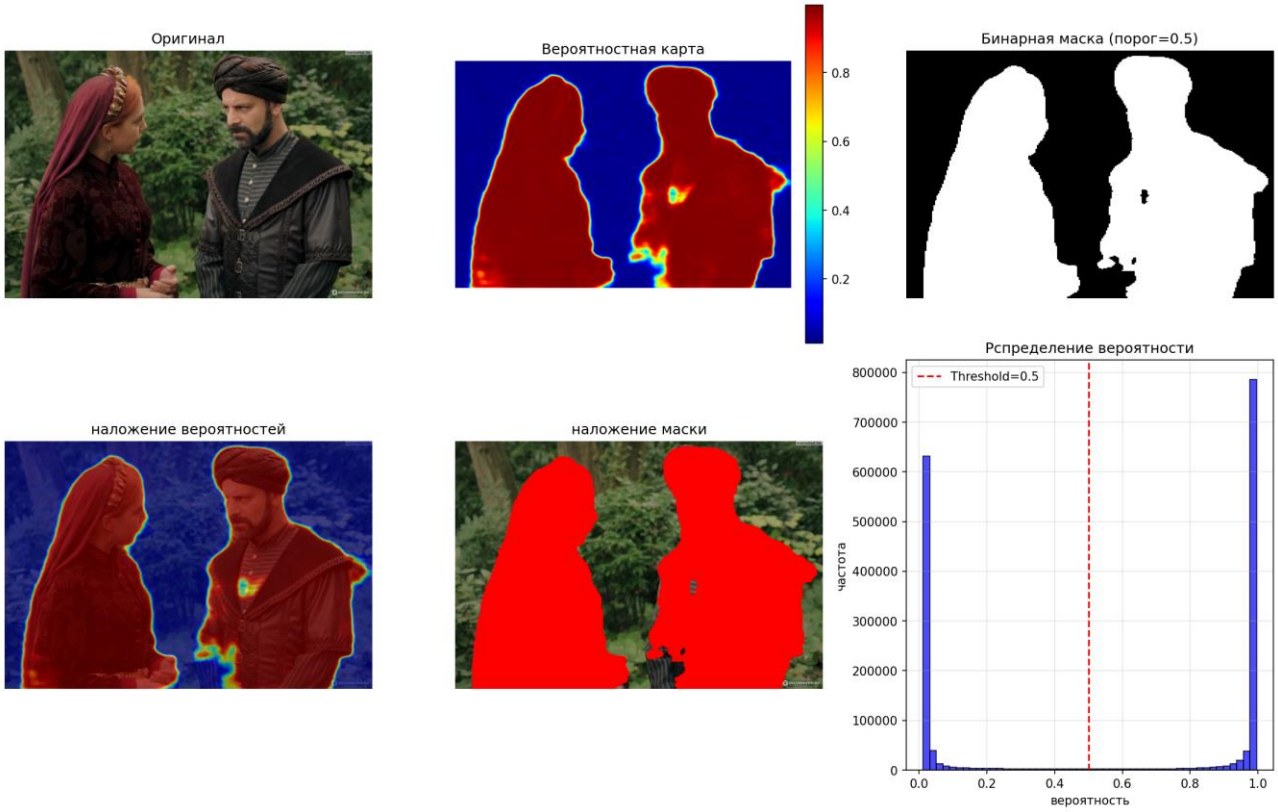

Рис. 10 - работа сети на случайном изображении из открытых источников (не присутствует в датасете)

Выводы

В результате обучения модель стала справляться с задачей сегментации достаточно эффективно, что показывает метрика Dice = 0.856.

Модель закончила обучение при срабатывании ранней остановки, вызванной отсутствием улучшения метрики Dice в течении 15-ти итераций. Лучшая версия модели была получена на 29-й эпохе.

По графику loss и Dice видно, что с 29-й итерации модель стала переобучаться, из-за чего точность предсказания тестовой выборки росла, а валидационной падала.

Для повышения точности предсказания:

1) При обучении и валидации нужно использовать больше данных.
2) Проводить аугментацию обучающей выборки: геометрические преобразования, цветовые, комбинация изображений.
3) Увеличить количество уровней сети, увеличить количество каналов.

Список литературы

[1] Kaggle: Supervisely Filtered Segmentation Person Dataset / Tapakah68. – URL: https://www.kaggle.com/datasets/tapakah68/supervisely-filtered-segmentation-person-dataset.
[2] U-Net [Электронный ресурс] // Википедия: свободная энциклопедия. – Режим доступа: https://ru.wikipedia.org/wiki/U-Net .
[3] Метрики оценки моделей нейронных сетей для чайников [Электронный ресурс] // Habr. – 2024. – URL: https://habr.com/ru/companies/slsoft/articles/893694/.