

Санкт-Петербургский государственный университет

Кафедра системного программирования

Группа 23.Б10-мм

Реализация автоматизированного сбора признаков с изображений УЗИ для моделей машинного обучения

Сазонова Ирина Константиновна

Отчёт по учебной практике

в форме «Решение»

Научный руководитель:
доцент кафедры системного программирования, к. ф.-м. н., Гориховский В. И.

Консультант:
инженер-исследователь, Исаков Андрей Алексеевич

Санкт-Петербург
2025

Оглавление

Введение	3
1. Постановка задачи	4
2. Обзор	5
2.1. Обзор существующих решений	5
2.2. Обзор используемых технологий	6
2.3. Выводы	6
3. Архитектура	7
4. Реализация	8
4.1. Предварительная обработка данных	9
4.2. Пайплайн	10
5. Эксперименты	15
5.1. Используемые датасеты	15
5.2. Исследовательские вопросы	15
5.3. Метрики	15
5.4. Результаты	20
Заключение	23
Список литературы	24

Введение

Щитовидная железа является важным органом эндокринной системы человека. Одной из актуальных задач современной медицины является своевременная диагностика патологий этого органа, включая воспалительные процессы и различные новообразования. С целью облегчить поиск патологий нами разрабатывается инструмент, позволяющий классифицировать воспаления и различные образования по снимкам ультразвукового исследования (УЗИ).

Необходимость создания подобного инструмента была подробно обоснована во введении к работе, выполненной в предыдущем семестре [8]. Кроме того, в работе [8] был приведен подробный анализ существующих методов сегментации щитовидной железы, в результате которого было принято решение использовать волновой алгоритм, основанный на разрастании области (region-growing). Поскольку данный метод требует задания стартовой точки, для ее определения применялся алгоритм поиска точек с наибольшим статистическим отклонением.

В данной работе будет представлен пайплайн обработки снимков УЗИ, включающий в себя следующие этапы: предобработка изображений, сегментация, выбор лучших масок и сбор признаков для моделей машинного обучения.

Кроме того, на примере данных с УЗИ молочных желез, будет продемонстрирована обобщающая способность пайплайна для работы с любыми датасетами, содержащими снимки УЗИ или изображения подобного вида.

1. Постановка задачи

Общая цель проекта заключается в разработке универсального инструмента для автоматической классификации заболеваний внутренних органов по изображениям, полученным с помощью ультразвукового исследования.

В рамках текущей работы рассматривается задача обработки снимков УЗИ с применением методов сегментации и последующего извлечения признаков для моделей машинного обучения.

Целью работы на этот семестр является создание пайплайна для обработки снимков УЗИ и сбора признаков на основании исследований, проведенных в предыдущем семестре. Для её выполнения были поставлены следующие задачи:

1. разработать архитектуру полноценной обработки снимков УЗИ;
2. реализовать предварительную обработку и загрузку данных;
3. реализовать пайплайн для обработки снимков УЗИ и сбора с них признаков;
4. выполнить экспериментальную апробацию на нескольких наборах данных.

2. Обзор

2.1. Обзор существующих решений

Подробный обзор существующих методов сегментации щитовидной железы и её узлов был представлен в работе за прошлый семестр [8]. В таблице 1 приведены основные существующие методы и решения с указанием их особенностей и заявленной точности.

Таблица 1: Существующие методы и решения

Методы	Представители	Особенности	Заявленная точность
Контуры и формы	ACWE[1]	Ручное выделение области и настройка параметров	80%
	GA-VBAC[6]	Ручное выделение области, автоматическая настройка параметров	—
Области	Graph Cuts[1]	Ручное выделение объекта и фона	76.5%
	NC [3]	Грубые результаты	—
Машинное/ глубокое обучение	TND [2]	Детекция из пяти этапов	91.3%
	CNN [4]	Без ручного выделения	86.8%
Гибрид	GC-ACWE [5]	Ручное выделение области	—

2.2. Обзор используемых технологий

В работе использовался язык программирования Python, поскольку он предоставляет большой набор специализированных библиотек для анализа данных и обработки изображений.

Для работы с изображениями применялись библиотеки OpenCV и scikit-image (skimage). Численные вычисления выполнялись с использованием NumPy и SciPy, визуализация графиков осуществлялась с помощью библиотеки Matplotlib.

Для последовательной обработки данных и реализации конвейера использовался модуль Pipeline из библиотеки scikit-learn (sklearn).

Для автоматизированного подбора гиперпараметров использовался фреймворк Optuna, поскольку из-за большого количества данных и параметров перебор всех возможных конфигураций был бы чрезмерно затратным по времени.

Извлечение признаков производилось с использованием библиотек skimage и SciPy, а также PyWavelets, которая реализует дискретное вейвлет-преобразование.

2.3. Выводы

Полная подготовка к созданию инструмента была проведена в предыдущем семестре, включая анализ датасета, существующие решения и выбор метрик для проведения экспериментов. На основании полученных результатов, в том числе из работы [7], было сформировано представление о структуре будущего классификатора и составлен список требований, необходимых для его реализации.

3. Архитектура

На основании предыдущих наработок и исследований были составлены следующие требования к создаваемому инструменту:

- наличие модуля для загрузки датасета;
- наличие классов, реализующих предобработку, сегментацию, выбор лучших масок и сбор признаков;
- все указанные классы должны реализовывать единый интерфейс, обеспечивающий возможность их интеграции в общий конвейер;
- наличие удобной структуры для хранения информации об изображении;
- наличие связующего звена между частью сбора признаков и классификации;
- возможность использовать различные алгоритмы сегментации.

С учетом этих требований была разработана архитектура для обработки изображений и сбора признаков, представленная на Рис. 1.

Полная архитектура, включая часть с классификацией, представлена в конце работы во вложении.

За обработку изображений отвечают следующие классы: `DDTILoader`, `BUSLoader`, `Image`, `Cropper`, `ImageSegmenter`, `MaskSelector`, `FeatureExtractor`, `PointFinder`, `SegmentationAlgorithm`.

Подробное описание каждого из них приведено в разделе с реализацией.

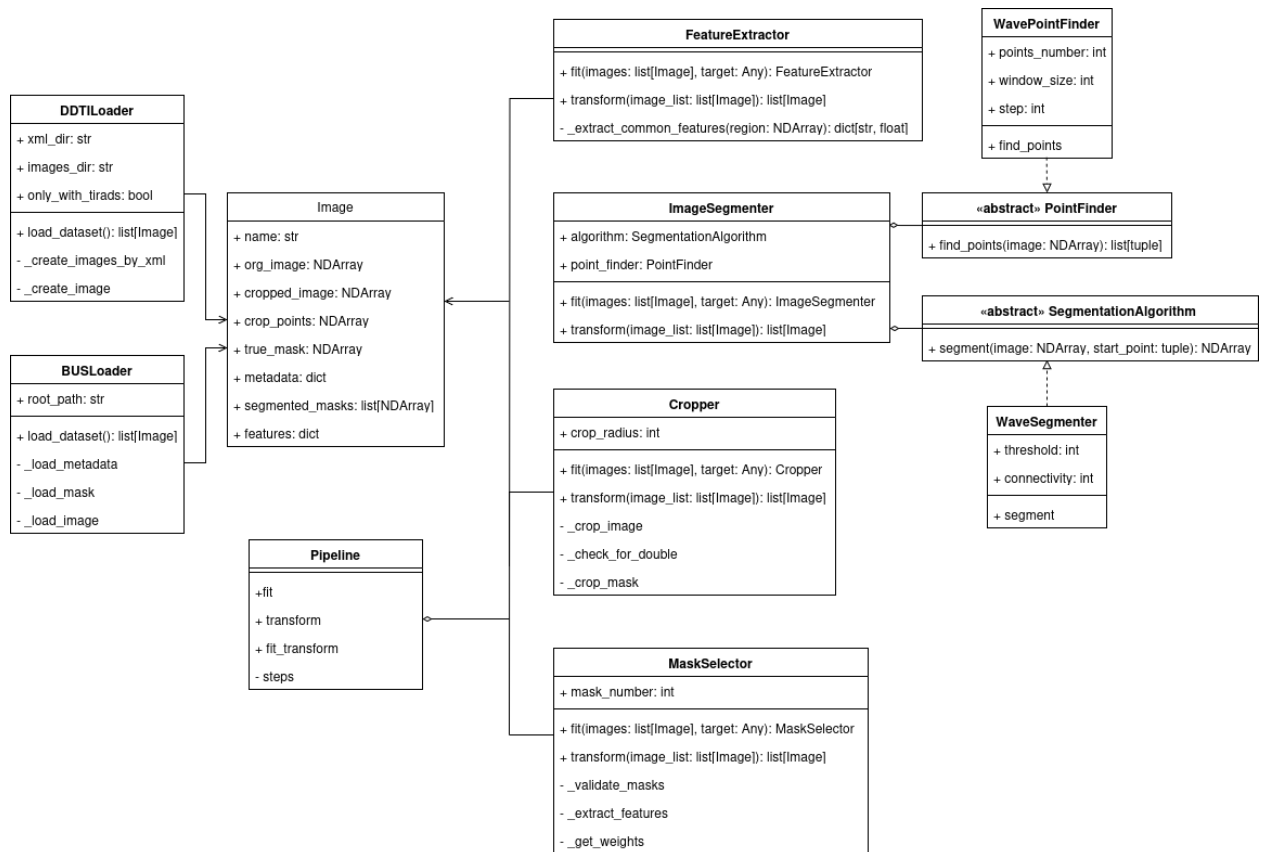


Рис. 1: Представление архитектуры через UML-диаграмму

4. Реализация

Обработка изображений была разделена на два этапа: загрузку датасета и построение конвейера обработки данных. Загрузка датасета была вынесена за пределы пайплайна, поскольку указанные компоненты выполняют различные логические функции. Кроме того, отдельная реализация обеспечила более гибкий контроль над входными данными, поступающими в пайплайн, что является особенно важным при подборе гиперпараметров и проведении экспериментов.

Для удобного хранения данных был реализован класс `Image`. В нем содержится информация об имени изображения (названии файла), метаданных, которые были извлечены из файлов для обучения, и само изображение, представленное в виде numpy массива. Остальные поля класса заполняются по мере прохождения шагов пайплайна, а именно: обрезанное изображение, координаты пикселей, по которым произошла обрезка (для восстановления полного изображения), сегментированные

маски и словарь с признаками.

Изначально весь процесс обработки изображений был разработан для датасета DDTI¹, содержащего снимки УЗИ щитовидной железы. Однако впоследствии было установлено, что построенный пайплайн может быть обобщён и применён к другим аналогичным наборам данных.

4.1. Предварительная обработка данных

Данный этап является одним из ключевых, несмотря на то, что он не входит в основной конвейер обработки данных. В рамках предварительной обработки осуществляется загрузка датасета и формирование на его основе списка объектов класса `Image`, описанного ранее. Разработка универсального загрузчика затруднена из-за уникальности структуры каждого конкретного датасета.

В случае с набором данных DDTI были получены изображения УЗИ в формате JPG, хранящиеся в одной директории, а также сопроводительные аннотации в формате XML. Для загрузки такого датасета был разработан класс `DDTILoader`, реализующий механизм фильтрации повреждённых файлов, создания масок по разметке из XML для каждого изображения и формирования списка объектов типа `Image`. В качестве аргументов данному классу передаются пути к директориям с XML-файлами и изображениями. Кроме того, предусмотрена возможность исключения изображений, для которых в XML-описаниях отсутствует метка TIRADS с помощью параметра `only_with_tirads`. Метод `load_dataset()` возвращает список объектов класса `Image`, который затем отправляется на первый этап пайплайна — `Cropper`.

Класс `BUSLoader` предназначен для загрузки датасета BUS-BRA². В конструктор класса передаётся путь к директории, содержащей подпапки с изображениями, масками для обучения, а также CSV-файл с метаданными. Благодаря предварительной обработке, проведённой для изображений в составе датасета, дополнительное применение класса

¹<https://www.kaggle.com/dasmehdixtr/ddti-thyroid-ultrasound-images>

²<https://www.kaggle.com/dasmehdixtr/ddti-thyroid-ultrasound-images?select=100.xml>

Cropper не требуется: все изображения уже представлены в формате PNG и обрезаны. Также отсутствует необходимость в генерации масок, поскольку они включены во входные данные. Метод `load_dataset()` возвращает список объектов класса `Image`, подготовленных для дальнейшей обработки и поступающих сразу в `ImageSegmenter`.

4.2. Пайплайн

Все последующие этапы обработки изображений организованы в виде шагов объекта класса `Pipeline` из библиотеки `scikit-learn`. Для интеграции собственных компонентов в состав пайплайна необходимо, чтобы соответствующие классы наследовались от базовых классов `BaseEstimator` и `TransformerMixin`. Кроме того, они должны реализовывать метод `fit(X, y=None)` — вызываемый во время обучения пайплайна — и метод `transform(X)`, применяющий преобразование к переданным данным. На вход методов подаётся и возвращается одинаковая структура данных — список объектов класса `Image`. Такой подход обеспечивает совместимость собственных преобразований с инфраструктурой `scikit-learn` и позволяет последовательно применять несколько этапов обработки в рамках единого конвейера.

4.2.1. Cropper

Примитивная версия класса была представлена в предыдущей работе [8] в виде отдельной функции, осуществляющей простую обрезку изображений. В настоящем проекте она оформлена в виде класса **Cropper**, включающем улучшения и дополнительную функциональность.

Метод `fit()` не выполняет никаких действий и реализован исключительно для соответствия интерфейсу, требуемому в составе пайплайна.

Метод `transform()` принимает на вход список объектов класса `Image` и для каждого из них выполняет следующую последовательность действий:

- выполняется обрезка исходного изображения по заданному радиусу, результат сохраняется в поле `cropped_image` соответствующего `Image`;
- с использованием оператора Собеля по горизонтали анализируется профиль градиентов по столбцам изображения, что позволяет найти резкие вертикальные границы; если обнаружено значительное суммарное увеличение градиента (превышающего медиану более чем в 2 раза) — производится разбиение на левую и правую части; в текущем объекте сохраняется левая часть, а правая часть используется для создания нового объекта `Image`, добавляемого к выходному списку;
- обрезка обучающих масок выполняется по тем же координатам, что и соответствующих изображений, чтобы обеспечить корректное выравнивание данных; разделение двух масок происходит согласованно с разделением двух изображений.

Такой подход позволяет автоматически устранять дублирование двух снимков на одном изображении и корректно выполнять обрезку как изображений, так и масок, тем самым обеспечивая согласованность в размерах на последующих этапах обработки.

4.2.2. ImageSegmenter

Класс реализует сегментацию изображений.

В качестве аргументов конструктор принимает абстрактные классы `PointFinder` и `SegmentationAlgorithm`. От них наследуются классы `WaveSegmenter` и `WavePointFinder`, которые представляют из себя реализации волнового алгоритма и алгоритма поиска точек с наибольшими статистическими отклонениями, которые были детально описаны в работе [8].

Можно настраивать гиперпараметры для сегментации, такие как:

- порог интенсивности;

- размер скользящего окна;
- шаг для перемещения скользящего окна;
- количество точек, которые будет искать `WavePointFinder`.

Метод `fit()` в `ImageSegmenter` также используется для соответствия интерфейсу и не выполняет никаких действий.

Метод `transform()` принимает список из объектов класса `Image` и применяет перечисленные выше алгоритмы для каждого `cropped_image`. Полученные сегментированные маски записываются в поле `segmented_masks` как список из элементов массивов `numpy`.

Сегментированные маски, как и обучающие маски, на данном этапе не нормализуются и состоят из пикселей интенсивностей 0 и 255.

4.2.3. MaskSelector

Класс реализует выбор `n` лучших масок из всех сегментированных.

Поскольку основной целью работы является сбор признаков для моделей машинного обучения, требуется сегментировать не одно конкретное воспаление, а учитывать всех претендентов на патологию и признаки других органов вблизи патологии. Поэтому `MaskSelector` выбирает `n` лучших масок, которые являются представителями областей с отклонениями на изображении. Количество масок является настраиваемым гиперпараметром.

Метод `fit()` используется для обучения KNN (`k`-ближайших соседей) из `sklearn`:

- на вход подается список из объектов `Image` с обучающими масками;
- происходит валидация масок: каждой маске присваивается вес (насколько она плохая, в зависимости от её размера); таким образом самые плохие маски (слишком большие или слишком маленькие) отбрасываются, но остается не менее `n` масок;

- из сегментированных k масок выбирается n лучших с максимальными значениями метрики IoU (Insertion over Union);
- с каждой маски собираются статистические, контурные и форменные признаки;
- признаки масштабируются с помощью `StandardScaler`;
- на этих признаках происходит обучение KNN.

Метод `transform()` принимает список из объектов `Image`, где нет информации о размеченных масках. Выполняется следующее:

- валидация масок с отбрасыванием плохих масок, как было описано выше;
- сбор признаков с сегментированных масок, масштабирование;
- ранжирование масок по возрастанию расстояний, которые были получены в результате использования `knn.kneighbors`

Таким образом получаем n лучших масок, которые наиболее близки к эталонным.

4.2.4. FeatureExtractor

Класс реализует сбор признаков с сегментированных областей, полученных в результате наложения каждой из n масок на изображение. Реализует интерфейс, требуемый для пайплайна, позволяет подобрать гиперпараметры для улучшения качества признаков.

Извлекаемые признаки с изображений и масок:

- **Статистические признаки** — среднее, стандартное отклонение, асимметрия, эксцесс, энтропия, квантили интенсивностей, гистограмма распределения (по количеству `hist_bins`).
- **LBP (Local Binary Patterns)** — гистограмма текстурных шаблонов, устойчивых к освещению и отражающих микроструктуру области; вычисляется по параметрам `lbp_points` и `lbp_radius`.

- **GLCM (Gray-Level Co-occurrence Matrix)** — признаки текстурной организованности: контраст, энергия, однородность, корреляция; рассчитываются по наборам `glcm_distances` и `glcm_angles`.
- **Вейвлет-признаки** — статистики коэффициентов дискретного вейвлет-преобразования: среднее, стандартное отклонение, энергия, минимум, максимум по разным направлениям (горизонтальные, вертикальные, диагональные компоненты) и уровням `wavelet_level`.
- **Контурные признаки** — средние значения оператора Лапласа и модуля градиента по Собелю, отражают структуру границ в изображении.
- **Геометрические признаки масок** — площадь, периметр, компактность, соотношение сторон ограничивающего прямоугольника, выпуклость, округлость; описывают форму области интереса.

Всего с учетом выбора 10 лучших масок было получено 754 признака, качество которых оценивалось в работе Артема Кушнина.

Весь код покрыт unit-тестами.

5. Эксперименты

5.1. Используемые датасеты

Эксперименты проводились на датасетах DDTI и BUS-BRA.

Датасет DDTI состоит из 466 изображений УЗИ щитовидной железы формата JPG после исключения дефектных файлов. После деления двойных снимков количество изображений увеличилось до 626.

Датасет BUS-BRA состоит из 1875 изображений УЗИ молочных желез формата PNG и такого же количества масок, дефектов обнаружено не было.

Все эксперименты проводились на TPU (Tensor Processing Unit) от Google.

Гиперпараметры для наилучшей сегментации были подобраны заранее с использованием Optuna, код для экспериментов выложен на гитхабе.

5.2. Исследовательские вопросы

RQ1 : как увеличение количества начальных точек для волнового алгоритма влияет на точность и время выполнения сегментации?

RQ2 : какова точность (качество) сегментации при использовании пайплайна на различных наборах данных?

5.3. Метрики

Качество сегментации оценивалось по 4 метрикам, вычисляемым через перекрытие масок:

1. Intersection over Union (IoU)

Отношение площади пересечения к площади объединения:

$$\text{IoU} = \frac{|\text{GT} \cap \text{PM}|}{|\text{GT} \cup \text{PM}|}$$

2. Dice coefficient (F1-score)

Гармоническое среднее точности и полноты:

$$\text{Dice} = \frac{2 \cdot |\text{GT} \cap \text{PM}|}{|\text{GT}| + |\text{PM}|}$$

3. Precision (точность)

Доля правильных предсказаний в выделенной области:

$$\text{Precision} = \frac{|\text{GT} \cap \text{PM}|}{|\text{PM}|}$$

4. Recall (полнота)

Доля захваченной истинной области:

$$\text{Recall} = \frac{|\text{GT} \cap \text{PM}|}{|\text{GT}|}$$

Где:

GT — Ground Truth (истинная маска)

PM — Predicted Mask (сегментированная маска)

$\cap$ — логическое И (пересечение)

$\cup$ — логическое ИЛИ (объединение)

$|\cdot|$ — площадь маски (число пикселей)

Для анализа использовалась наилучшая маска из набора масок каждого изображения, так как основная задача — сбор признаков из выделенных областей.

Dice coefficient — основная метрика, поскольку она позволяет учитывать как точность, так и полноту при работе с несбалансированными классами:

$$\text{Dice} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

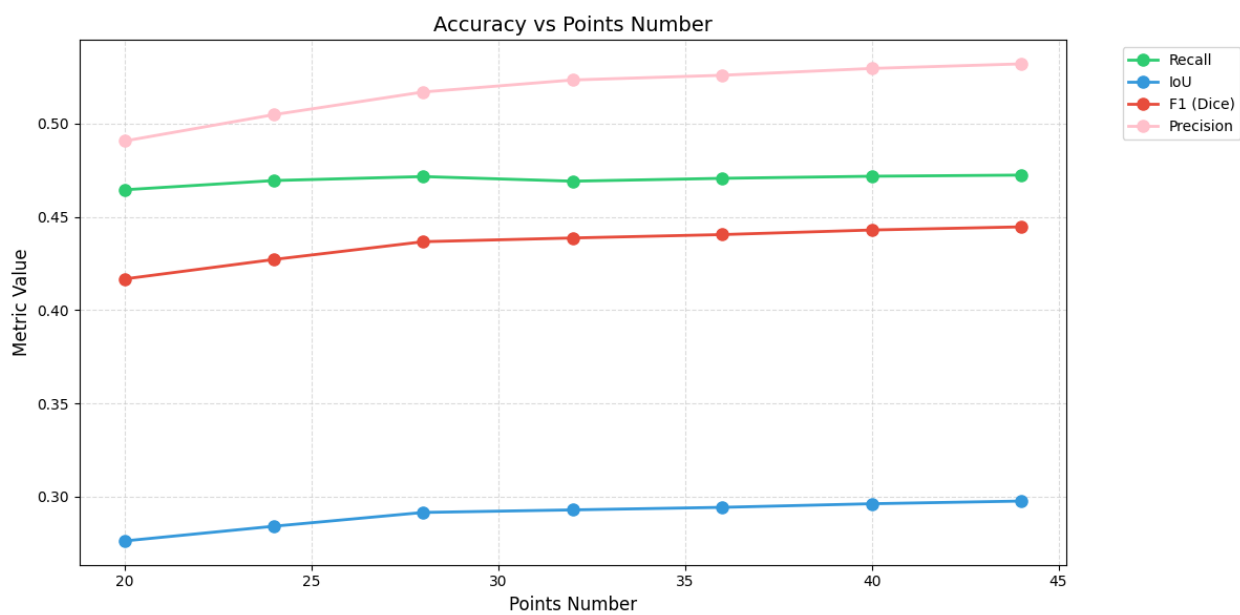


Рис. 2: DDTI: Зависимость точности от количества точек

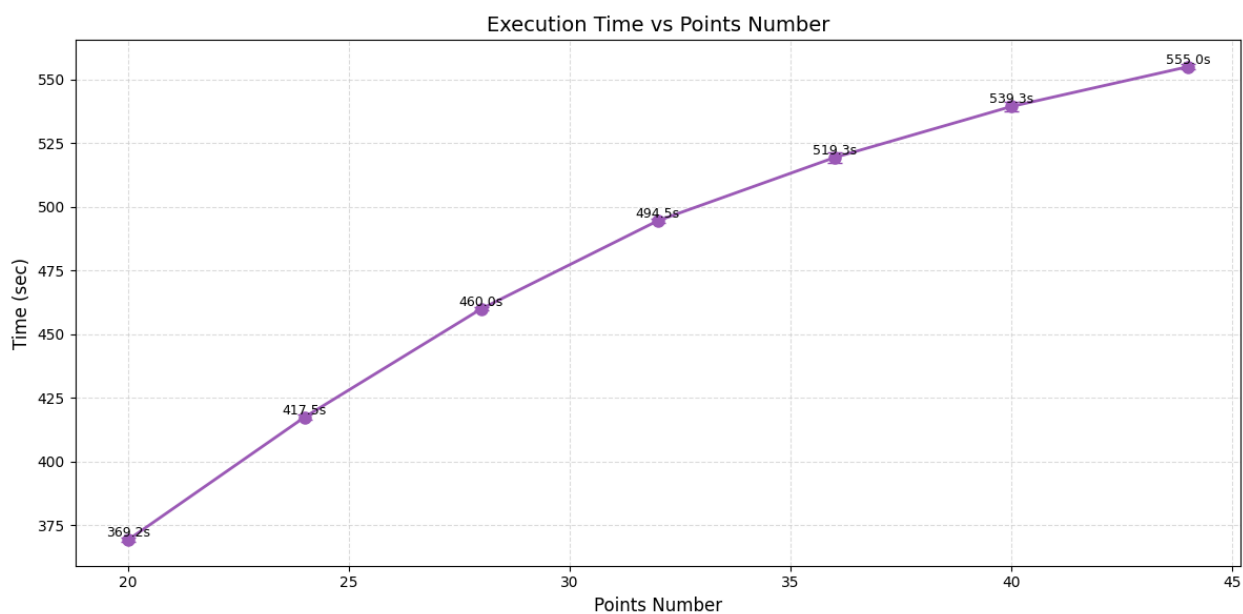


Рис. 3: DDTI: Зависимость времени выполнения от количества точек

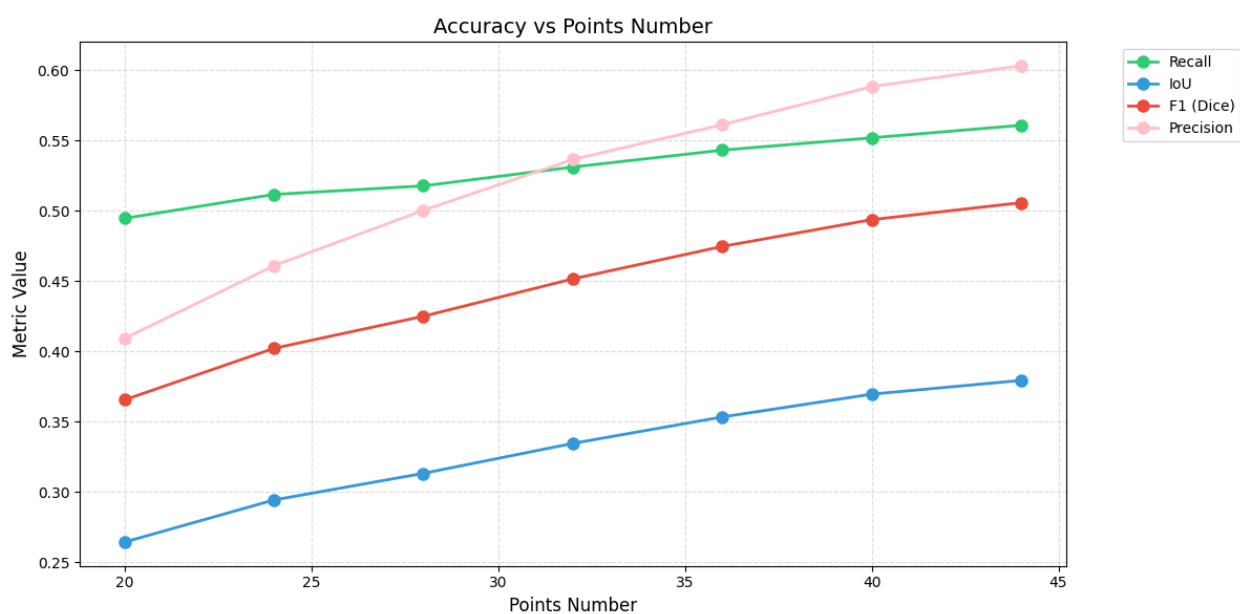


Рис. 4: BUS-BRA: Зависимость точности от количества точек

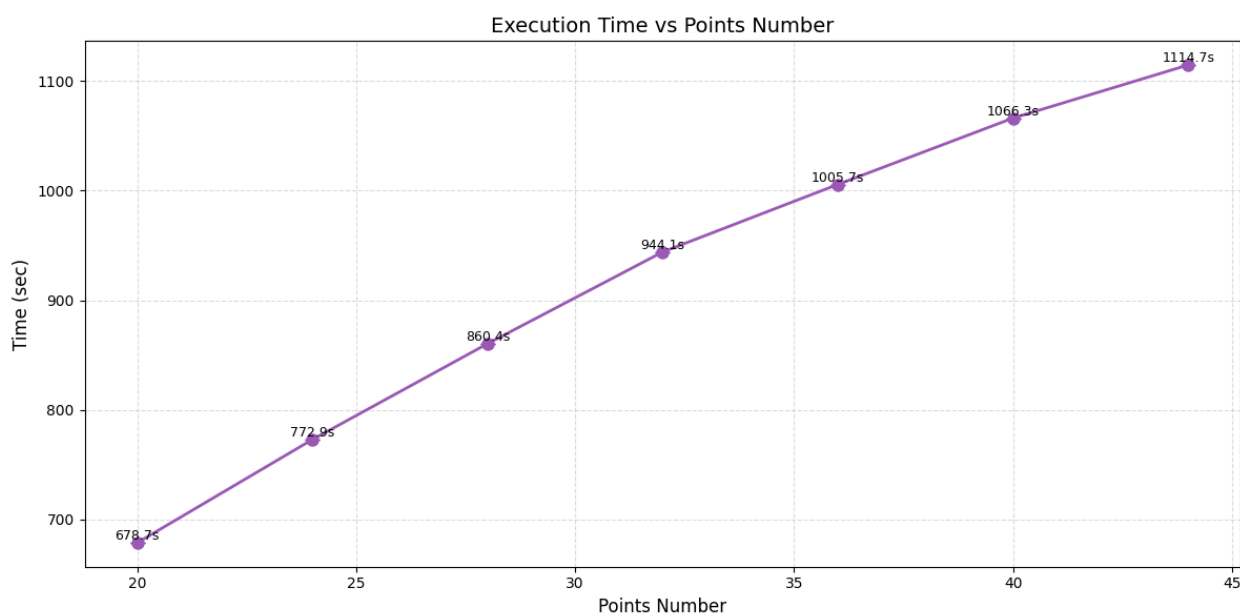


Рис. 5: BUS-BRA: Зависимость времени выполнения от количества точек

Таблица 2: Результаты сегментации на датасете DDTI: зависимость метрик и времени от количества точек

Кол-во точек	Время обработки, с		Dice			
	Среднее	Откл.	Recall	IoU	(F1-score)	Precision
20	369.2485	0.7569	0.4646	0.2763	0.4168	0.4907
24	417.4546	1.0325	0.4695	0.2842	0.4272	0.5049
28	459.9518	0.5710	0.4717	0.2916	0.4367	0.5170
32	494.5425	0.7426	0.4692	0.2929	0.4387	0.5235
36	519.3292	2.1482	0.4707	0.2943	0.4405	0.5260
40	539.3479	1.6944	0.4718	0.2962	0.4430	0.5296
44	554.9595	1.0909	0.4724	0.2976	0.4447	0.5321

Таблица 3: Результаты сегментации на датасете BUS-BRA: зависимость метрик и времени от количества точек

Кол-во точек	Время, с	Dice				
		Recall	IoU	(F1-score)	Precision	
20	678.680	0.4947	0.2644	0.3656	0.4092	
24	772.875	0.5117	0.2943	0.4022	0.4610	
28	860.378	0.5178	0.3133	0.4250	0.5004	
32	944.098	0.5312	0.3346	0.4516	0.5366	
36	1005.723	0.5432	0.3534	0.4747	0.5614	
40	1066.262	0.5520	0.3696	0.4938	0.5884	
44	1114.737	0.5609	0.3794	0.5058	0.6032	

5.4. Результаты

5.4.1. RQ1

1. DDTI

На Рис. 2 представлен график зависимости значения метрик от количества точек, которые используются в качестве начальных для волнового алгоритма. На графике видно, что значения всех метрик качества увеличиваются с ростом количества точек. Однако, чем больше точек, тем больше времени нужно на вычисления, что можно увидеть на графике на Рис. 3.

Время выполнения сегментации и выбора лучших масок для датасета DDTI измерялось по 3 раза для каждого количества стартовых точек. Отклонения и среднее время работы отображено в таблице 2. Эксперимент проводился на всем датасете.

2. BUS-BRA

Эксперименты для датасета BUS-BRA проводились на случайной подвыборке из 650 изображений и, поскольку качество снимков BUS-BRA заметно выше качества снимков в датасете DDTI, время работы увеличилось примерно вдвое, что можно увидеть в таблице 3.

На Рис. 4 видно, что с ростом количества точек качество сегментации также растет.

5.4.2. RQ2

1. DDTI

С наибольшим количеством точек (44) значение коэффициента Dice составляет примерно 0.44 (таблица 2), что заметно меньше результатов, представленных в таблице 1 из обзора существующих решений.

Однако все представленные решения, использующие для сегментации методы областей, требуют ручного выделения области. В

нашем инструменте все происходит полностью автоматически.

Качество полученных результатов на данный момент грубое, поскольку для точной оценки нужно учитывать не только среднее значение метрики на всем датасете, но и различные отклонения. Однако для решения этого вопроса требуется отдельное исследование. Поэтому нельзя однозначно оценить точность в 0.44.

Несмотря на невысокие значения метрик при оценке сегментации, точность (f1-мера) лучшего классификатора (SVM) в работе Артема Кушнина составила 76.6% (при использовании оптимальных параметров сегментации и 32 точках).

2. BUS-BRA

После выбора оптимального количества точек, равного 32, был проведен эксперимент на полном датасете BUS-BRA (код можно посмотреть на гитхабе).

В результате значение Dice составило 0.47, precision — 0.56, recall — 0.56, iou — 0.35. Это значительно лучше, чем соответствующие значения метрик на подвыборке (таблица 3), однако, как и в случае с DDTI, на данный момент нельзя однозначно оценить эти значения.

Точность классификации (f1) при использовании EfficientNet составила 86.9%.

5.4.3. Выводы

Исходя из проведенных экспериментов можно сделать следующие выводы:

- точность сегментации увеличивается с количеством опорных точек, однако рекомендуется выбирать такое количество точек, при котором время работы программы будет удовлетворять требованиям задачи и имеющимся ресурсам;

- точность сегментации на данный момент нельзя оценить однозначно, однако, при имеющихся небольших значениях, точность последующей классификации показывает неплохие результаты (76.6%, 86.9%).

Заключение

В ходе работы за текущий семестр были выполнены следующие поставленные задачи:

- разработана архитектура полноценной обработки снимков УЗИ;
- реализованы загрузчики для двух различных датасетов;
- реализован пайплайн для обработки снимков УЗИ и сбора с них признаков;
- выполнена экспериментальная апробация на двух наборах данных.

Весь код выложен на гитхабе ³.

В дальнейшем планируется:

- оценить качество результатов сегментации;
- исследовать пороги качества сегментации;
- применить новые алгоритмы сегментации и провести сравнение.

³<https://github.com/ArtemKushnir/ThyroidScan>

Список литературы

- [1] Evaluation of commonly used algorithms for thyroid ultrasound images segmentation and improvement using machine learning approaches / Prabal Poudel, Alfredo Illanes, Debdoot Sheet, Michael Friebe // Journal of healthcare engineering. — 2018. — Vol. 2018, no. 1. — P. 8087624.
- [2] Keramidas Eustratios G, Maroulis Dimitris, Iakovidis Dimitris K. TND: a thyroid nodule detection system for analysis of ultrasound images and videos // Journal of medical systems. — 2012. — Vol. 36. — P. 1271–1281.
- [3] Segmentation of ultrasound images of thyroid nodule for assisting fine needle aspiration cytology / Jie Zhao, Wei Zheng, Li Zhang, Hua Tian // Health information science and systems. — 2013. — Vol. 1. — P. 1–12.
- [4] Ultrasound image-based thyroid nodule automatic segmentation using convolutional neural networks / Jinlian Ma, Fa Wu, Tian'an Jiang et al. // International journal of computer assisted radiology and surgery. — 2017. — Vol. 12. — P. 1895–1910.
- [5] Zhou Jingan. Thyroid tumor ultrasound image segmentation based on improved graph cut // 2016 International Conference on Intelligent Transportation, Big Data & Smart City (ICITBS) / IEEE. — 2016. — P. 130–133.
- [6] A genetically optimized level set approach to segmentation of thyroid ultrasound images / Dimitris K Iakovidis, Michalis A Savelonas, Stavros A Karkanis, Dimitris E Maroulis // Applied Intelligence. — 2007. — Vol. 27. — P. 193–203.
- [7] Кушнир А. Ю. Классификация заболеваний щитовидной железы по снимкам УЗИ. — URL: https://se.math.spbu.ru/thesis/texts/Kushnir_Artem_Jur'evich_Autumn_practice_2nd_year_2024_text.pdf.
- [8] Сазонова И. К. Обработка и анализ снимков УЗИ щитовидной железы. — URL: https://se.math.spbu.ru/thesis/texts/Sazonova_

[_Irina__Konstantinovna_Autumn_practice_2nd_year_2024_text.pdf](#).

