Санкт-Петербургский государственный университет
Программная инженерия

Кафедра системного программирования

Сабрина Андраниковна Мусатян

Библиотека для создания программного обеспечения, использующего медицинские изображения

Бакалаврская работа

Научный руководитель:
к. т. н., доц. Литвинов Ю. В.

Рецензент:
ведущий программист ООО "ПитерСофтвареХаус" Полозов В.С.

Санкт-Петербург
2019

SAINT-PETERSBURG STATE UNIVERSITY

Software Engineering

Sabrina Musatian

Library for development of medical images processing software

Graduation Thesis

Scientific supervisor:
C.Sc., Associate Professor Yurii Litvinov

Reviewer:
senior developer at PiterSoftwareHouse Victor Polozov

Saint-Petersburg
2019

Оглавление

Введение

За последнее десятилетие было разработано множество различных подходов к решению задач в области медицинских изображений. Благодаря данным исследованиям, перед научным сообществом открываются новые и более сложные задачи. Тем не менее, сейчас уже не достаточно только алгоритмических решений, связанных с диагностикой и лечением на основе КТ (Компьютерная томография) и МРТ (Магнитно-резонансная томография) снимков.

Врачам требуются высокофункциональные программные комплексы, решающие различные задачи и позволяющие наиболее точно определить диагноз пациента. Следовательно, необходимо не только разработать высокоэффективный алгоритм анализа медицинских изображений, но и интегрировать его в удобную для врача среду, в которой могут быть использованы другие инструменты медицинского анализа. Многие из этих инструментов являются общими для целого класса медицинских задач, а значит могут быть предоставлены в рамках единой платформы.

В рамках данной работы рассматриваются различные программные комплексы для работы с медицинскими изображениями, а также представляется библиотека для создания программного обеспечения, использующего медицинские изображения, упрощающая процесс разработки медицинских инструментов.

1. Постановка задачи

Целью данной работы является создание библиотеки для разработки программного обеспечения, использующего медицинские изображения, разработка типовых функций для работы с медицинскими изображениями и демонстрация успешных применений данной библиотеки для решения актуальных медицинских проблем. Для ее достижения были поставлены следующие задачи:

- изучить существующие программные решения для работы с медицинскими изображениями;
- разработать и реализовать легко расширяемую архитектуру для предлагаемой библиотеки;
- реализовать программные модули в рамках предлагаемой библиотеки для работы с наиболее распространенными форматами медицинских изображений;
- разработать примеры использования библиотеки.

2. Обзор предметной области

2.1. Обзор программных комплексов для работы с медицинскими изображениями

Существует множество пакетов с открытым исходным кодом для работы с медицинскими изображениями – либо специально предназначенных для этой области, либо широко используемых для этих целей.

Некоторые из них предназначены для решения типичных задач, связанных с медицинскими изображениями. Они предоставляют набор инструментов или функций для написания небольших программ, позволяющих анализировать и обрабатывать медицинские изображения. К таким инструментам можно отнести ITK [9] – библиотека для обработки медицинских изображений, VTK [23] – надстройка над ITK для работы с 3D изображениями и их визуализацией и OpenCV [3] – широко используемая библиотека для компьютерного зрения.

Другие решают задачи, связанные с анализом изображений определенных органов или заболеваний. Например, программные комплексы для работы с изображениями мозга (FreeSurfer [7], SPM [18] и другие). Расширение таких программ для решения широкого спектра задач в медицине является невозможным, так как чаще всего архитектура таких приложений написана под конкретную задачу и не является легко расширяемой.

Также существует множество приложений общего назначения для работы с медицинскими изображениями. Такие системы предоставляют базовую функциональность по работе со снимками. Тем не менее, их невозможно расширить под решение каких–либо узконаправленных задач (например, сегментации или нахождения особенностей, присущих определенным заболеваниями). К таким системам относятся: Ginkgo CAD [8] и ClearCanvas [4].

2.2. Обзор расширяемых медицинских приложений

Другим классом систем являются расширяемые медицинские приложения, ориентированные прежде всего на конечного пользователя. Такие приложения включают в себя все необходимые базовые методы для работы с медицинскими изображениями в интегрированном пользовательском интерфейсе. Примерами таких систем являются Slicer [15], Weasis [25] и OsiriX [17]. Последний является дорогостоящим коммерческим продуктом и недоступен широкой аудитории. Расширение таких приложений возможно за счет плагинов, специально написанных под эти платформы. Однако такой подход не дает разработчику достаточной гибкости при создании системы.

Наиболее общим и гибким продуктом для работы с медицинскими изображениями является MITK [14] – система программного обеспечения с открытым исходным кодом для разработки интерактивных инструментов, использующих медицинские изображения. MITK содержит модули, основанные на алгоритмах библиотеки ITK [9] и алгоритмах визуализации из VTK [23]. В рамках MITK реализовано множество алгоритмов, дополняющих функции ITK и VTK, а также новые алгоритмы, которые выходят за рамки обоих библиотек. Таким образом, MITK позволяет создавать разнообразные медицинские системы. Хотя MITK и является кросс–платформенным фреймворком для разработки медицинских приложений, на регулярной основе поддерживаются только версии для Windows 10, Linux Ubuntu 16.04 и 18.04. Также, поскольку он изначально написан на языке C++, данный проект необходимо собирать отдельно для каждой платформы. Более того, разработчики должны использовать уникальную процедуру сборки, предоставляемую MITK, для создания и добавления новых модулей.

2.3. Medical Images Research Framework (MIRF)

В данной работе представляется новая библиотека с открытым исходным кодом для разработки приложений, использующих медицинские изображения – MIRF (Medical Images Research Framework). MIRF на-

писана на языке программирования Kotlin с акцентом на обеспечение плавной интеграции между современными исследованиями в области медицинской визуализации. Благодаря тому, что MIRF использует Kotlin, библиотека может быть легко интегрирована в любые проекты с Java средой, в том числе в мобильные приложения.

В качестве альтернативных языков программирования для MIRF рассматривались C++, C# и python. Первые версии MIRF были реализованы на Java. MIRF планировался как масштабный кафедральный проект, который выйдет далеко за рамки одной выпускной квалификационной работы. В этом году работа над MIRF велась в рамках двух дипломных работ. Необходимо было обеспечить не только кросс–платформенность и интеграцию с мобильными устройствами, но и создавать удобные пользовательские интерфейсы в дальнейшем для врачей. Необходимо было учитывать и то, что над MIRF будет работать множество людей в последующие года. Именно поэтому было принято решение выбрать строго типизированный язык программирования, который также удовлетворял бы всем вышеописанным требованиям.

Особое внимание в MIRF было уделено возможности интегрирования алгоритмов искусственного интеллекта и различных подходов машинного обучения для диагностики и лечения различных заболеваний в инфраструктуру MIRF. Это обусловлено тем, что в данное время наиболее эффективные решения задач анализа медицинских изображений используют алгоритмы машинного обучения [13].

3. Архитектура библиотеки

3.1. Структура библиотеки

Библиотека MIRF представляет собой набор модулей для различных задач, связанных с обработкой как медицинских изображений, так и других данных. Данные модули делятся на два глобальных пакета.

- Core – минимальный набор необходимых модулей для корректной работы библиотеки. Здесь содержатся модули, реализующие классы для внутреннего отображения данных, их передачи и обработки действий во время исполнения любого модуля библиотеки.

- Features – содержит модули с основной пользовательской функциональностью, необходимой для облегчения разработки и специфичных пользовательских сценариев: различные средства и фильтры для анализа изображений, механизмы для получения данных из хранилища, адаптеры для различных форматов медицинских данных. Любые пользовательские модули должны расширять пакет features.

3.2. Конвейеры

Основным сценарием использования MIRF является создание конечным пользователем конвейеров. Конвейеры – это последовательность обработчиков исходных данных, выполненных с применением подхода Pipes & Filters [24].

В библиотеке MIRF любая вычислительная логика должна расширять интерфейс Algorithm. Algorithm – это класс–обработчик, который при вызове изменяет только данные, поданные ему на вход, при этом не вызывая стороннего кода, связанного с обработкой данных. Algorithm не сохраняет данные и является исключительно обработчиком. Такой подход дает возможности для гибкой разработки новых алгоритмов и организации иерархий.

Экземпляры Algorithm выступают в качестве фильтров в нашей архитектуре. Класс Algorithm инкапсулируется классом PipelineBlock, который является основной сущностью, служащей для передачи данных между алгоритмами. Сообщение между блоками построено на основе шаблона проектирования Наблюдатель – после исполнения алгоритма внутри блока, блок сообщает всем своим слушателям о завершении вычислений. Некоторые блоки могут так же заниматься агрегацией данных для следующих блоков или иметь другое специфичное предназначение (например, являются указателями на окончание вычислений в конвейере).

Диаграмма классов пакета Core, в рамках которого реализуется вышеописанный подход, представлена на рис. 1

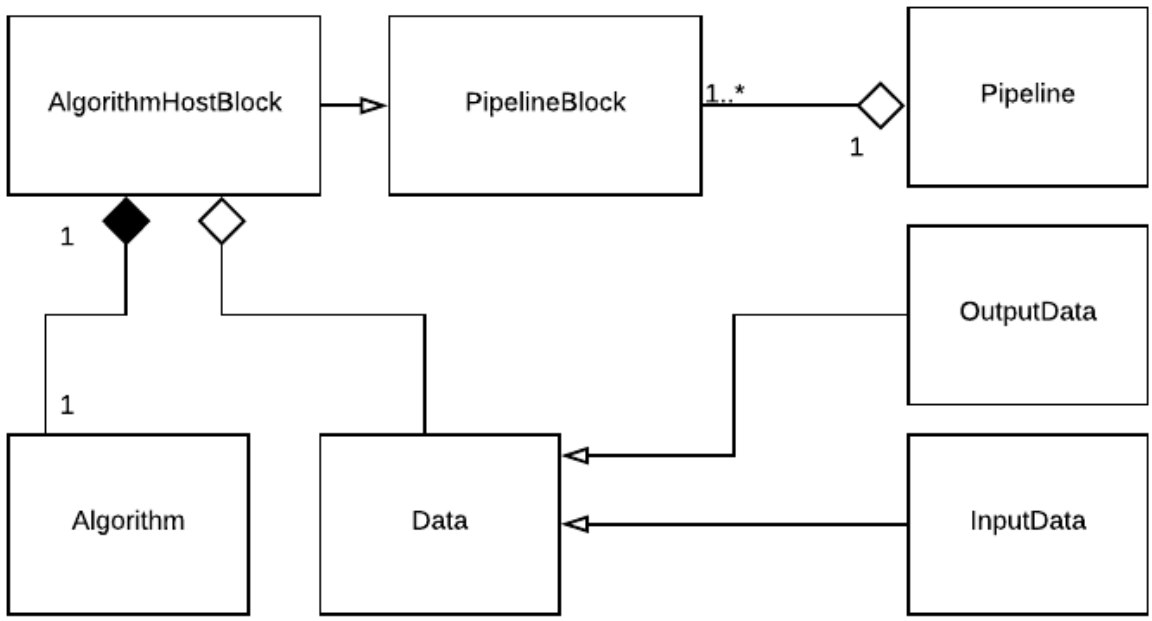


Рис. 1: Диаграмма классов пакета Core

3.3. Представление данных

Любые данные в MIRF должны реализовывать абстрактный класс Data. Основной задачей этого класса является управление метаданными. Любой класс, унаследованный от класса Data, должен быть использован

только как объект для хранения данных. Такое архитектурное решение обеспечивает однозначность и понятность данных в рамках библиотеки.

3.4. Инициализация конвейеров

MIRF предоставляет различные блоки и функции, из которых могут быть собраны пользовательские конвейеры. При необходимости, пользователь также может создавать свои блоки для специфичных сценариев использования. Инициализация конвейера может быть выполнена с помощью нескольких простых шагов, описанных в листинге 1.

```
val pipe = Pipeline(''Имя конвейера'')
// Создание блоков
val firstBlock = PipelineBlock(
                    Block parameters
                )
...Создание остальных блоков...
// Создание связей между блоками
firstBlock.dataReady +=
    secondBlock::inputReady
...Создание остальных связей...
// Инициализируем начальный блок и входные данные
pipe.rootBlock = firstBlock
pipe.run(Data)
```

Листинг 1: Инициализация конвейера в MIRF.

4. Поддержка различных форматов медицинских изображений

4.1. MedImage

Существует несколько наиболее распространенных форматов медицинских изображений, например: DICOM [22] и NIfTI [12]. Чтобы для всех форматов могли быть применимы одни и те же алгоритмы анализа, в MIRF создан общий класс для внутреннего хранения медицинских изображений – MedImage. В данном классе содержится список атрибутов, извлекаемых по определенным правилам, в зависимости от исходного формата, и пиксельное представление изображения. Таким образом, все алгоритмы для работы с медицинскими изображениями работают с классом MedImage, что позволяет пользователю библиотеки эффективно переиспользовать имеющийся код.

4.2. DICOM

Формат DICOM представляет собой набор элементов типа ключ–значение, причем само изображение также хранится по ключу, в качестве значения. Для DICOM строго определены возможные наборы ключей, которые используются повсеместно. Для чтения DICOM изображений было рассмотрено несколько библиотек для работы с данным форматом на языке Kotlin: ImageJ [10], DCM4CHE [6] и PixelMed [16]. Несмотря на то, что ImageJ поддерживает чтение DICOM, данная библиотека не предоставляет функциональность для записи изображений в этом формате. DCM4CHE – это широкий набор инструментов для работы с изображениями в формате DICOM. Он предоставляет множество функций для работы с такими изображениями на медицинских серверах. Поскольку мы не хотим перегружать нашу библиотеку ненужными зависимостями, мы приняли окончательное решение в пользу PixelMed, так как данная библиотека поддерживает чтение, работу с атрибутами и запись DICOM изображений без использования громоздких зависи-

мостей, таких как в DCM4CHE. После прочтения списка атрибутов для DICOM изображения, мы преобразуем его в класс MedImage, посредством создания внутреннего представления атрибутов и выделения массива снимков из исходного формата.

4.3. NIfTI

Другой популярный формат медицинских изображений – NIfTI [12]. Существует несколько различий между DICOM и NIfTI снимками. Например, данные, которые они хранят, и способ их хранения. В частности, метаданные NIfTI не включают информацию о пациентах или больнице. Данный формат хранит только пиксельное представление снимков и метаданные с настройками МРТ-аппарата. Кроме того, NIfTI хранит набор медицинских срезов в одном файле (набор медицинских изображений), в то время как DICOM обычно сохраняет их в виде отдельных файлов. Чтобы добавить поддержку NIfTI в MIRF, была использована библиотека ImageJ [10]. После прочтения NIfTI файла, подобно изображениям в формате DICOM, MIRF конвертирует всю информацию во внутреннее представление MedImage, чтобы одни и те же алгоритмы могли работать с различными форматами файлов.

5. Модуль для поддержки интеграции алгоритмов из библиотеки Tensorflow

5.1. Введение в Tensorflow

Tensorflow [20] является одной из самых используемых библиотек для реализации методов машинного обучения и других алгоритмов, включающих большое количество математических операций. Библиотека Tensorflow была разработана компанией Google и является одной из самых популярных библиотек для машинного обучения на GitHub.

Основными компонентами Tensorflow являются граф операций и тензоры, которые передаются по ребрам графа к вершинам. С математической точки зрения, тензор – это N-мерный вектор. Поэтому тензор может быть использован для представления N-мерных наборов данных. Граф операций это такой граф, в котором каждая вершина является какой-либо операцией (например, сложение, умножение и т.д.).

Принципы работы с Tensorflow достаточно просты: необходимо составить граф операций, передать в этот граф данные и произвести вычисления.

Tensorflow может быть использован для создания и обучения нейронных сетей. Для хранения обучаемых параметров моделей нейронных сетей в качестве вершин в Tensorflow создаются вершины типа Variable. Такие вершины используются для хранения и обновления параметров модели. Переменные – это буферы в памяти, хранящие тензоры. Они должны быть явно инициализированы и могут быть сохранены на диск во время или после тренировки модели. Позже сохраненные значения могут быть восстановлены для возобновления процесса тренировки или тестирования модели.

Во время тренировки Tensorflow сохраняет переменные в файлах контрольных точек. Чтобы загрузить значения этих переменных, необходимо объявить граф операций или загрузить его из отдельного файла Tensorflow графа. Такое поведение позволяет легко разбивать процесс тренировки на несколько этапов и быстро возобновлять обучение модели.

Однако, чтобы использовать такие модели из языка Kotlin, необходимо объединить файлы графа и значений переменных в специальный файл Tensorflow, который имеет расширение .pb.

5.2. Блок Tensorflow в MIRF

Поскольку в современных исследованиях часто используют методы глубокого обучения для решения различных задач в медицине, в MIRF уделили особое внимание возможности интеграции этих подходов в рамках данной библиотеки. В данный момент реализована поддержка наиболее часто используемых сред глубокого обучения, таких как Tensorflow и Keras [11]. В результате, интеграция моделей, написанных с помощью фреймворка Tensorflow, возможна в специальном блоке библиотеки MIRF – TensorflowModel. Так как фреймворк Tensorflow предоставляет Java API для работы с его моделями, мы смогли создать блок, который может запускать предоставленные модели в рамках библиотеки MIRF. Чтобы использовать обученную модель Tensorflow на тестовых данных, необходимо создать экземпляр блока Tensorflow с параметрами модели. Для этого достаточно передать блоку путь к сохраненной модели и имена входных и выходных вершин графа операции данной модели.

Кроме того, поскольку в рамках пакета Tensorflow возможно использование интерфейсов из библиотеки Keras, в MIRF можно интегрировать не только Tensorflow, но и модели Keras с помощью блока TensorflowModel.

Насколько известно автору данной работы, никакое другое программное обеспечение для создания медицинских приложений не обеспечивает возможность интеграции методов глубинного обучения в рамках своей базовой функциональности. Автор считает, что эта функция очень важна при разработке современных медицинских приложений, поскольку она полностью обеспечивает интеграцию сложных моделей искусственного интеллекта в реальные медицинские приложения и позволяет разработчикам сосредоточиться на создании новых алгоритмов с помощью

предпочитаемых ими языков и сред разработки.

Чтобы использовать Tensorflow API в C ++ или Java, разработчики должны объявить граф модели и множество других параметров, прежде чем модель может быть запущена. Однако пользователи MIRF могут настроить блок Tensorflow с помощью всего нескольких строк, описанных в листинге 2.

```
val tensorflowModel = TensorflowModel(
    MODEL_NAME, INPUT_NODE_NAME,
    OUTPUT_NODE_NAME, OUTPUT_DIMS
    )
val tensorflowModelRunner =
AlgorithmHostBlock<Data, Data>(
    {
        tensorflowModel.runModel(
            it, INPUT_DIMS)
    },
    pipelineKeeper = pipe
)
```

Листинг 2: Настройка блока Tensorflow в MIRF.

5.3. Скрипты для подготовки моделей

Как уже упоминалось ранее, чтобы использовать Tensorflow или Keras модель в Kotlin, ее необходимо предварительно привести к формату .pb. В рамках MIRF добавлены несколько полезных скриптов, позволяющих создавать .pb файл модели как из файлов графа и значений переменных Tensorflow, так и из файлов Keras моделей. Наличие таких инструментов сэкономит время разработчиков, которые хотят воспользоваться блоком Tensorflow в MIRF и обеспечит более простую интеграцию таких моделей.

6. Особенности реализации

MIRF является кросс–платформенной библиотекой, поддерживающей Linux, Windows, OS X и Android. Для непрерывной поддержки вышеописанных платформ, в проекте была настроена непрерывная интеграция. Для проверки изменений в коде на Linux и OS X использовался сервис Travis CI [21], а для платформы Windows – AppVeyor [1].

Для проектов с открытым исходным кодом немаловажным критерием является понятность предоставляемого кода. Для обеспечения высокого качества кода в MIRF применялся Codacy [5] – это автоматизированный инструмент для статического анализа, который помогает разработчикам быстрее выпускать высококачественный код. Также, в MIRF проходили регулярные проверки кода другими участниками группы.

7. Примеры использования

С помощью MIRF разработчики могут легко создавать пользовательские конвейеры для решения конкретных задач. Это автоматизирует многие ручные сценарии использования и может принести новые, неиспользованные ранее функции, в работу врачей.

7.1. Анализ опухолей головного мозга

В качестве примера использования MIRF для решения конкретной медицинской проблемы, рассмотрим задачу анализа опухолей головного мозга и генерации отчета для врача на основе полученных данных.

Существуют различные виды медицинской документации, которую врачи создают после приема пациента. Эти документы обычно включают КТ и МРТ снимки, как дополнение к окончательному диагностическому документу и рекомендации. Врачи должны заполнять такие отчеты вручную или полуавтоматически и включать в них изображения. MIRF предоставляет инструменты для автоматического создания этих отчетов на основе результатов вычислений пользовательских конвейеров. MIRF генерирует отчет в формате PDF, который включает все необходимые изображения.

В работе [13] автором данной выпускной квалификационной работы было проведено исследование, которое показало, что в настоящее время сегментация опухолей головного мозга выполняется либо вручную, либо полуавтоматически. Несмотря на то, что существует большое количество исследований, посвященных решению данной задачи, в данный момент не было зарегистрировано ни одного случая, когда результаты этих исследований могли бы стать предметом реальных клинических испытаний.

Основная информация, которая может быть получена из сегментации опухоли мозга на ранних этапах лечения, – это объем опухоли и ее относительный объем по отношению ко всему мозгу пациента. Эта информация должна быть добавлена к генерируемому отчету о болезни. Данные действия (сегментация опухоли по МРТ-снимкам, вычисление

объема и включение этой информации в отчет) обычно выполняются специалистами вручную. Эта последовательность действий может быть представлена как конвейер в рамках библиотеки MIRF и включена в конечный инструмент для врача.

Для сегментации опухоли головного мозга мы используем один из наиболее эффективных алгоритмов решения этой задачи [2]. Алгоритм сегментации получает на вход МРТ-снимки головного мозга в формате NIfTI [12] и создает три маски для различных типов опухолевых тканей (рисунок 2).

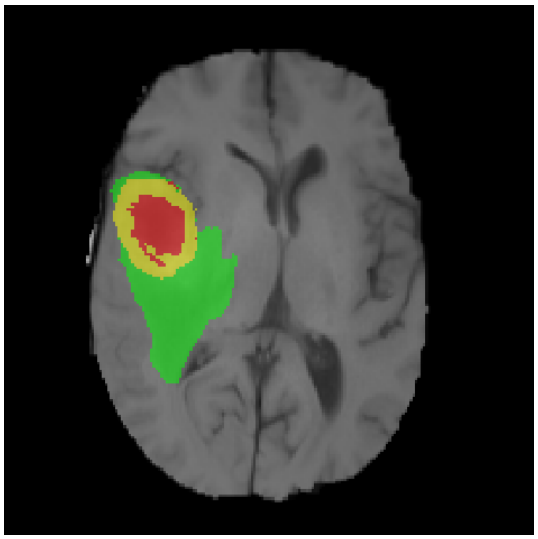


Рис. 2: Пример сегментации опухоли головного мозга. Различные типы опухолевых тканей: некротическое ядро (красный), накапливающийся очаг опухоли (желтый), отек (зеленый).

Поскольку изображения МРТ представлены в виде набора срезов, где воксели (трехмерный пиксель) в срезе соответствуют некоторому конкретному физическому объему, можно рассчитать объем на основе количества вокселей, входящих в маску. Информация о том, какому объему соответствуют воксели, хранится в метаданных медицинского изображения и зависит от настроек конкретного МРТ-аппарата.

MIRF рассчитывает объем каждой опухолевой ткани на основе масок, полученных в результате сегментации. Используя исходные изображения мозга, можно определить весь объем мозга и определить относительность между этими объемами. Затем MIRF создает полный отчет с этой

информацией, используя инструменты создания PDF. Поток данных и блоки, используемые в этом конвейере, можно увидеть на рисунке 3. Пример отчета, сгенерированного в результате работы данного конвейера, можно увидеть на рисунке 4.

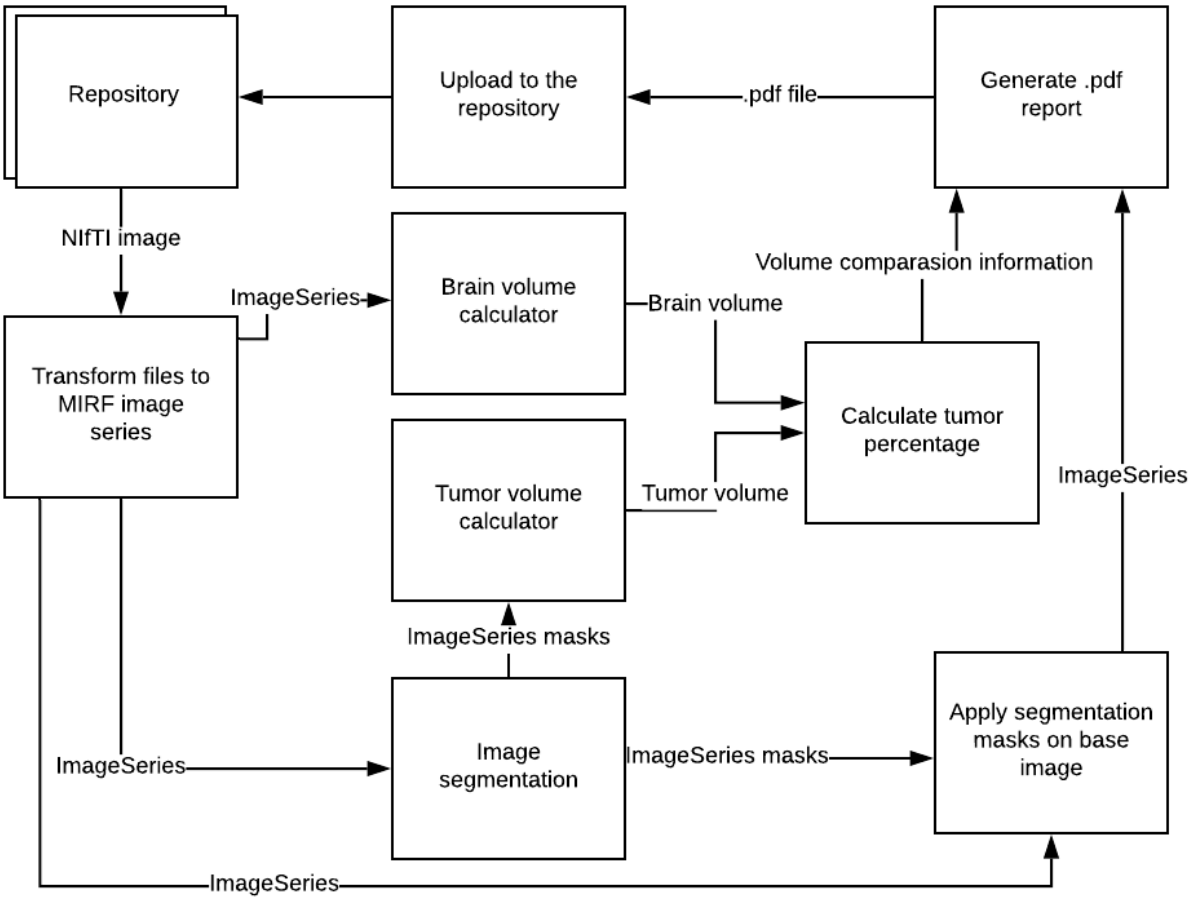


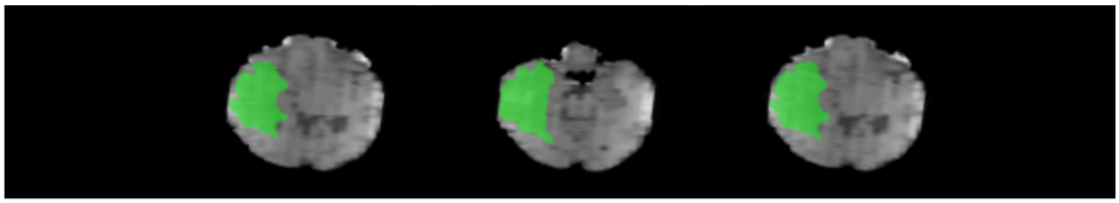
Рис. 3: Диаграммы потоков данных и схема блоков для примера сегментации опухолей головного мозга. MIRF загружает NIfTI–изображение из локального хранилища, извлекает список атрибутов и адаптирует их к внутреннему формату библиотеки. После этого, изображения отправляются блоку сегментации. Результатом работы этого блока являются три маски для каждой из опухолевых тканей. Затем объем для опухоли и для всего мозга рассчитывается в соответствующих блоках для подсчета объема. После этого генерируется отчет в формате PDF из собранных данных.

Brain Tumor Segmentation Report

John Doe
54 y.o.

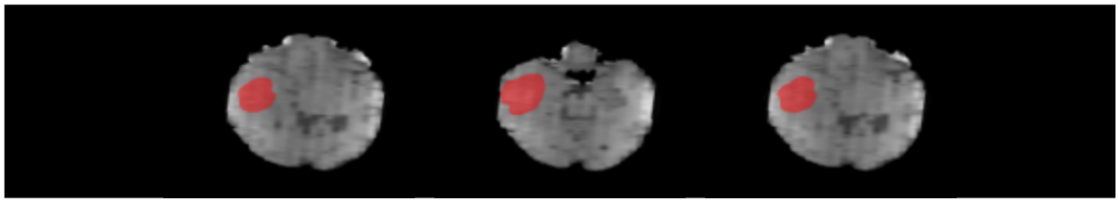
Total brain volume	1194.1779999999978 cm³

Edema:



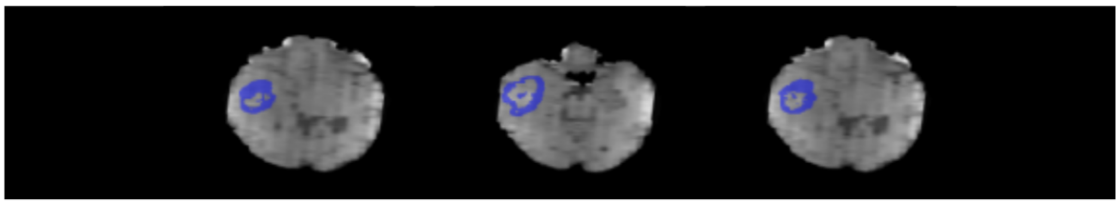
Whole tumor volume	102.31199999999984 cm³
Tumor percentage compared to brain volume	8,57%

Necrotic/cystic core:



Necrotic/cystic core tumor volume	33.396999999999984 cm³
Percentage compared to brain volume	2,80%

Enhancing core:



Enhancing core tumor volume	20.496 cm³
Percentage compared to brain volume	1,72%

Рис. 4: Пример медицинского отчета, сгенерированного с помощью библиотеки MIRF для анализа опухолей головного мозга.

7.2. Мобильное приложение для определения рака кожи

Благодаря тому, что MIRF разрабатывается с помощью языка программирования Kotlin, можно создавать не только кросс–платформенные настольные приложения, но и мобильные. Это позволяет разработчикам использовать одни и те же конвейеры и сценарии на различных устройствах без переписывания кода. Чтобы продемонстрировать преимущества этого подхода, было создано приложение под Android для обнаружения рака кожи. Для классификации изображений мы используем реализацию одного из алгоритмов глубокого обучения для этой задачи с открытым исходным кодом [19]. Данная модель глубокого обучения реализована с использованием библиотеки Keras. Поскольку фреймворк Tensorflow позволяет преобразовывать Keras модели в Tensorflow модели, мы можем использовать блок интеграции Tensorflow из MIRF для этой модели. В результате, для обнаружения рака кожи по картинке может быть использован один и тот же конвейер, как в настольном приложении, так и на телефоне.

Чтобы показать простоту такого подхода, продемонстрируем код конвейера (листинг 3), который используется на Android для запуска этого примера. Он принимает путь изображения в качестве входных данных и выдает метку, показывающую, является ли родинка доброкачественной или злокачественной.

```
val pipe = Pipeline(”Detect moles”)
val assetsBlock =
    AlgorithmHostBlock<Data, AssetsData>
    (...параметры блока...)
val imageReader =
    AlgorithmHostBlock<AssetsData,
                       BitmapRawImage>
    (...параметры блока...)
val tensorflowModelRunner =
AlgorithmHostBlock<BitmapRawImage,
                   ParametrizedData<Int> >
    (...параметры блока...)
val root = PipeStarter()
// Создаем связи между блоками
root.dataReady +=
            assetsBlock::inputReady
assetsBlock.dataReady +=
            imageReader::inputReady
imageReader.dataReady +=
    tensorflowModelRunner::inputReady
// Запускаем конвейер
pipe.rootBlock = root
pipe.run(MirfData.empty)
```

Листинг 3: Пример конвейера, используемого на Android для определения рака кожи.

Заключение

В результате выполнения данной работы были получены следующие результаты:

- изучены существующие программные решения для работы с медицинскими изображениями;
- совместно с другим участником проекта разработана и реализована расширяемая архитектура библиотеки MIRF;
- реализованы следующие программные модули: поддержка форматов DICOM и NIfTI, интеграция алгоритмов машинного обучения на основе библиотеки Tensorflow;
- на основе библиотеки MIRF реализовано два тестовых приложения: приложение для анализа опухолей головного мозга и мобильное приложение для определения рака кожи.

Основные результаты работы были доложены на конференции “SEIM 2019” и “СПИСОК 2019”. Также статья по результатам данной работы была принята к публикации на CEUR-ws.org. Исходный код для данной работы можно найти в репозитории проекта:

https://github.com/MathAndMedLab/Medical-images-research-framework

MIRF предполагает дальнейшей развитие на кафедре, и уже сейчас в данном проекте участвуют заинтересовавшиеся студенты младших курсов.

Список литературы

[1] AppVeyor. — Access mode: https://github.com/marketplace/appveyor (online; accessed: 07.05.2019).

[2] Automatic Brain Tumor Segmentation Using Cascaded Anisotropic Convolutional Neural Networks / Guotai Wang, Wenqi Li, Sébastien Ourselin et al. // Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries. — Cham : Springer International Publishing, 2018. — P. 178–190.

[3] Bradski Gary, Kaehler Adrian. Learning OpenCV: Computer Vision in C++ with the OpenCV Library. — 2nd edition. — O'Reilly Media, Inc., 2013. — P. 580. — ISBN: 1449314651, 9781449314651.

[4] ClearCanvas. — Access mode: https://www.clearcanvas.ca/ (online; accessed: 17.12.2018).

[5] Codacy. — Access mode: https://github.com/marketplace/codacy (online; accessed: 07.05.2019).

[6] DCM4CHE. — Access mode: https://www.dcm4che.org (online; accessed: 17.12.2018).

[7] FreeSurfer. — Access mode: http://surfer.nmr.mgh.harvard.edu (online; accessed: 17.12.2018).

[8] Ginkgo CAD. — Access mode: https://github.com/gerddie/ginkgocadx (online; accessed: 17.12.2018).

[9] ITK. — Access mode: http://www.itk.org (online; accessed: 17.12.2018).

[10] ImageJ. — Access mode: https://imagej.nih.gov/ij/index.html (online; accessed: 17.12.2018).

[11] Keras. — Access mode: https://github.com/fchollet/keras (online; accessed: 17.12.2018).

[12] LONI MiND: metadata in NIfTI for DWI / Vishal Patel, Ivo D Dinov, John D Van Horn et al. // NeuroImage. — 2010. — 03. — Vol. 51. — P. 665–676.

[13] Medical Images Segmentation Operations / Musatian S.A., Lomakin A.V., Sartasov S.Yu. et al. // Trudy ISP RAN/Proc. ISP RAS. — 2018. — Vol. 30. — P. 183–194.

[14] The Medical Imaging Interaction Toolkit: challenges and advances / Marco Nolden, Sascha Zelzer, Alexander Seitel et al. // International journal of computer assisted radiology and surgery. — 2013. — 07. — Vol. 8, no. 4. — P. 607–620.

[15] Pieper S., Halle M., Kikinis R. 3D Slicer // 2004 2nd IEEE International Symposium on Biomedical Imaging: Nano to Macro (IEEE Cat No. 04EX821). — Vol. 1. — 2004. — April. — P. 632–635.

[16] PixelMed. — Access mode: `http://www.pixelmed.com/dicomtoolkit.html` (online; accessed: 17.12.2018).

[17] Rosset Antoine, Spadola Luca, Ratib Osman. OsiriX: An Open-Source Software for Navigating in Multidimensional DICOM Images // Journal of digital imaging : the official journal of the Society for Computer Applications in Radiology. — 2004. — 10. — Vol. 17. — P. 205–216.

[18] SPM. — Access mode: `ttp://www.fil.ion.ucl.ac.uk/spm/` (online; accessed: 17.12.2018).

[19] Skin cancer detection project. — Access mode: `https://github.com/dasoto/skincancer` (online; accessed: 10.02.2019).

[20] Abadi Martín, Agarwal Ashish, Barham Paul et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. — 2016. — Access mode: `https://arxiv.org/abs/1603.04467` (online; accessed: 17.12.2018).

[21] Travis CI. — Access mode: `https://github.com/marketplace/travis-ci` (online; accessed: 07.05.2019).

[22] Understanding and Using DICOM, The Data Interchange Standard for Biomedical Imaging / W. Dean Bidgood, Steven C. Horii, Fred W. Prior, Donald E. Van Syckle // Medical Image Databases / Ed. by Stephen T. C. Wong. — Boston, MA : Springer US, 1998. — P. 25–52.

[23] VTK. — Access mode: `http://www.vtk.org` (online; accessed: 17.12.2018).

[24] Vermeulen Allan, Beged-dov Gabe, Thompson Patrick. The Pipeline Design Pattern. — 1995.

[25] Weasis. — Access mode: `http://nroduit.github.io/en/` (online; accessed: 17.12.2018).