|  |  |  |
| --- | --- | --- |
| |  | | --- | | Пермский филиал федерального государственного автономного образовательного учреждения высшего образования  «Национальный исследовательский университет  «Высшая школа экономики»  *Факультет социально-экономических и компьютерных наук* | | Кирьянов Сергей Вячеславович  **Разработка системы диагностики остеопороза по КТ-снимкам позвоночника**  *Курсовая работа*  студента образовательной программы «Программная инженерия» по направлению подготовки *09.03.04 Программная инженерия*  Руководитель  Преподаватель кафедры информационных технологий в бизнесе НИУ ВШЭ-Пермь    М. Д. Чистогов | |

**Пермь, 2024**

**Содержание**

[Введение 3](#_Toc161939726)

[1 Анализ предметной области 5](#_Toc161939727)

[1.1 Данные о патологии 5](#_Toc161939728)

[1.2 Данные для обучения системы 6](#_Toc161939729)

[1.3 Анализ DICOM-формата 7](#_Toc161939730)

[1.4 Анализ средств для разработки и обучения модели 11](#_Toc161939731)

[1.5 Модели искусственного интеллекта, применяемые в медицине 12](#_Toc161939732)

[2 Обработка наборов данных 14](#_Toc161939733)

[2.1 Отбор и разметка данных для обучения системы 14](#_Toc161939734)

[2.2 Предобработка данных 17](#_Toc161939735)

[2.3 Подготовка данных к использованию в обучении, аугментация 22](#_Toc161939736)

[3 Разработка и обучение модели 25](#_Toc161939737)

[3.1 Выбор модели нейронной сети 25](#_Toc161939738)

[3.2 Создание и обучение модели 27](#_Toc161939739)

[3.3 Тестирование модели 32](#_Toc161939740)

[3.4 Проверка совместимости системы с записями для селф-тестирования 36](#_Toc161939741)

[4 Разработка приложения 39](#_Toc161939742)

[4.1 Диаграммы активностей и прецедентов 39](#_Toc161939743)

[4.2 Проектирование пользовательского интерфейса 41](#_Toc161939744)

[4.3 Разработка внутренней логики приложения 44](#_Toc161939745)

[4.4 Тестирование 47](#_Toc161939746)

[4.5 Испытания системы и ее внедрение 49](#_Toc161939747)

[5 Заключение 50](#_Toc161939748)

[6 Список используемой литературы 52](#_Toc161939749)

[ПРИЛОЖЕНИЕ А Техническое задание 54](#_Toc161939750)

[ПРИЛОЖЕНИЕ Б Руководство пользователя 55](#_Toc161939751)

[ПРИЛОЖЕНИЕ В Код модуля app.py 63](#_Toc161939752)

[ПРИЛОЖЕНИЕ Г Код модуля «load\_data.py» 64](#_Toc161939753)

[ПРИЛОЖЕНИЕ Д Код модуля «predict.py» 67](#_Toc161939754)

[ПРИЛОЖЕНИЕ Е Остальной код приложения 70](#_Toc161939755)

**Введение**

В настоящее время искусственный интеллект (ИИ) проникает во все сферы человеческой деятельности, и медицина не является исключением. Системы машинного обучения все чаще используются в диагностике заболеваний, помогая врачам делать более точные и быстрые вывода. Использование ИИ в медицине открывает новые возможности для улучшения качества медицинских услуг и уменьшения трудозатрат медицинских работник, что, в свою очередь, может способствовать удешевлению некоторых медицинских услуг.

Искусственный интеллект уже сейчас используется в различных областях медицины, в том числе, на территории Российской Федерации. Чаще всего системы диагностики заболеваний на основе ИИ используются в таких областях медицины, как радиология, патология, дерматология, офтальмология. Информационные системы на основе машинного обучения позволяют создавать алгоритмы, которые способны анализировать большие объемы медицинских данных, выявляя закономерности и расширяя возможности диагностики заболеваний. Эти системы могут помогать врачам в выявлении рака на ранних стадиях, диагностике пневмонии и COVID-19, обнаружении патологий опорно-двигательного аппарата и многих других заболеваниях.

Тем не менее, стоит понимать, что искусственный интеллект (по крайней мере, на данный момент) не исключает работу медицинских работников, и, в частности, подтверждение ими наличия того или иного заболевания. Такие системы используются исключительно для поддержки диагностирования, то есть, в первую очередь, для увеличения точности постановки диагноза специалистом.

Целью данной работы является разработка системы диагностики остеопороза по КТ-снимкам позвоночника.

Таким образом, предметом исследования является реализация системы диагностики остеопороза по КТ-снимкам позвоночника, а объектом исследования программные реализации систем с использованием искусственного интеллекта, предназначенных для использования в сфере здравоохранения, в частности, для анализа медицинских изображений.

Для достижения поставленной цели были сформулированы следующие задачи:

1. выполнить анализ предметной области, в том числе проанализировать клиническую задачу, решаемую системой и признаки исследуемой патологии (остеопороза), проанализировать требования к формату ответу системой и форме его предоставления, осуществить поиск и анализ данных, подходящих для обучения системы, произвести анализ DICOM-формата, а также рассмотреть существующие модели искусственного интеллекта (нейронных сетей), применяемые в медицине;
2. выполнить отбор данных для обучения системы, подготовить разметку данных, реализовать предобработку медицинских изображений для обучения системы, выбрать структуру и модель нейронной сети (обосновав выбор);
3. разработать модель и затем обучить её, произвести тестирование на размеченном наборе данных, с учетом различных метрик, проверить совместимость системы с записями из набора данных, предназначенного для селф-тестирования от ГБУЗ «НПКЦ ДиТ ДЗМ, реализовать предоставление ответа ИИ-сервиса для серии снимков в виде вероятности наличия патологии (в процентах);
4. осуществить разработку веб-приложения, проектирование пользовательского интерфейса, тестирование веб-приложения; провести испытания системы и ее внедрение.

Для следования корректному жизненному циклу программного обеспечения, задачи к поставленной цели выполнялись последовательно, с соблюдением графика работ, составленного на этапе согласования технического задания.

1. **Анализ предметной области**
   1. **Данные о патологии**

Данная система разрабатывается для диагностики остеопороза. Остеопороз - хронически прогрессирующее системное метаболическое (обменное) заболевание скелета или клинический синдром, проявляющийся при других заболеваниях и характеризующийся снижением плотности костей, нарушением их микроархитектоники и усилением хрупкости по причине нарушения метаболизма костной ткани с преобладанием катаболизма над процессами костеобразования, снижением прочности кости и повышением риска переломов [1].

Согласно данным ВОЗ, остеопороз поражает примерно 6,3% мужчин старше 50 лет и 21,2% женщин старше того же возраста во всем мире [2]. Исходя из численности мирового населения, в соответствии с соотношением по полам, примерно 500 миллионов мужчин и женщин во всем мире могут быть пострадать от этого заболевания.

По оценкам, в 2019 году в Европе остеопорозом страдали 32 миллиона человек в возрасте от 50 лет и старше, что эквивалентно 5,6% от общей численности населения Европы в возрасте старше 50 лет. Из них, приблизительно 25,5 миллиона женщин (22,1% женского населения Европы в возрасте старше 50 лет) и 6,5 миллиона мужчин (6,6% от общей численности мужского населения Европы в возрасте старше 50 лет) [3].

Основные признаки, по которым обычно определяется наличие патологии (согласно данным от mosmed.ai) [4]:

1. наличие на нативных изображениях позвонков, имеющих компрессионную деформацию тел ≥ 25 %, по полуколичественной шкале Genant, 2–3 степень.
2. снижение минеральной плотности костной ткани в телах позвонков в интервале Th11 – L3 (оптимально L1– L2) согласно критериям ACR 2018, позиции ISCD 2019 на нативных изображениях.

Из-за малого объема данных, основным признаком, по которому будет определяться наличие патологии, это снижение минеральной плотности костной ткани в телах позвонков, по их аксиальным снимкам.

Таким образом, остеопороз – опасное и труднодиагностируемое заболевание, особенно, на ранних стадиях развития, которое может затронуть миллионы человеческих жизней.

* 1. **Данные для обучения системы**

Для обучения системы с сайта mosmed.ai (сайт центра диагностики и телемедицины) были взяты 3 набора данных КТ-снимков с признаками остеопороза позвоночника. Поставщиком данным в данном случае является ГБУЗ «НПКЦ ДиТ ДЗМ». Далее под записями будут подразумеваться наборы серий КТ-снимков.

Первый набор имеет 6 записей [5], а второй 10 записей [6]. При этом, 2 записи из второго набора имеют технический дефект, о чем указано в описании. Таким образом, получается всего 14 записей, с бинарной разметкой (без признаков патологии; с признаками патологии). При этом, соотношение классов равно 5:9, то есть, записей без признаков патологии почти вдвое меньше, чем записей с признаками патологии, а по количеству самих снимков эта разница еще больше: снимков без признаков патологии почти в 3 раза меньше, чем снимков с признаками патологии.

Третий же набор предназначен для селф-тестирования по множеству патологий и имеет 7 подходящих записей [7]. Тем не менее, из-за низкого количества снимков без признаков патологии, были взяты данные и из данного набора (всего 2 записи из 7). Это позволило существенно увеличить качество системы.

Информация по всем наборам также представлены ниже в таблице 1.

Таблица 1 – Информация по используемым наборам данных

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Источник данных** | **Тип данных** | **Кол-во данных** | **Тип разметки данных** | **Соответствование классам** | **Уточнение** |
| ГБУЗ «НПКЦ ДиТ ДЗМ», mosmed.ai | DICOM, КТ-снимки, 3D | 10 | Категориальная. Классы: С патологией, Без патологии, Технический Дефект | 4;4;2 | MosMedData КТ с признаками остеопороза позвоночника тип III. |
| ГБУЗ «НПКЦ ДиТ ДЗМ», mosmed.ai | DICOM, КТ-снимки, 3D | 6 | Бинарная. Классы: Без патологии, с патологией | 1;5 | MosMedData КТ с признаками остеопороза позвоночника тип III, версия 2. |
| ГБУЗ «НПКЦ ДиТ ДЗМ», [mosmed.ai](http://mosmed.ai/) | DICOM, КТ-снимки, 3D | 7 (всего 41, остальные не соответствуют исследуемой патологии) | Бинарная. Классы: Без патологии, с патологией | - | Набор данных КТ, ММГ, РГ/ФЛГ с целью селф-тестирования ИИ-серсивов для поиска признаков приоритетных патологий |

Таким образом, данные, предназначенные для обучения системы, представлены в достаточно ограниченном объеме. Размета данных бинарная, что подразумевает также и бинарную классификацию (наличие или отсутствие патологии).

* 1. **Анализ DICOM-формата**

DICOM (Digital Imaging and Communications in Medicine) — медицинский отраслевой стандарт создания, хранения, передачи и визуализации цифровых медицинских изображений и документов обследованных пациентов [8]. Файлы DICOM имею расширение «.dcm».

DICOM-формат содержит много метаданных о пациенте, так называемых тэгов. При этом, часто многие такие метаданные удаляются из датасетов для соблюдения полной анонимности.

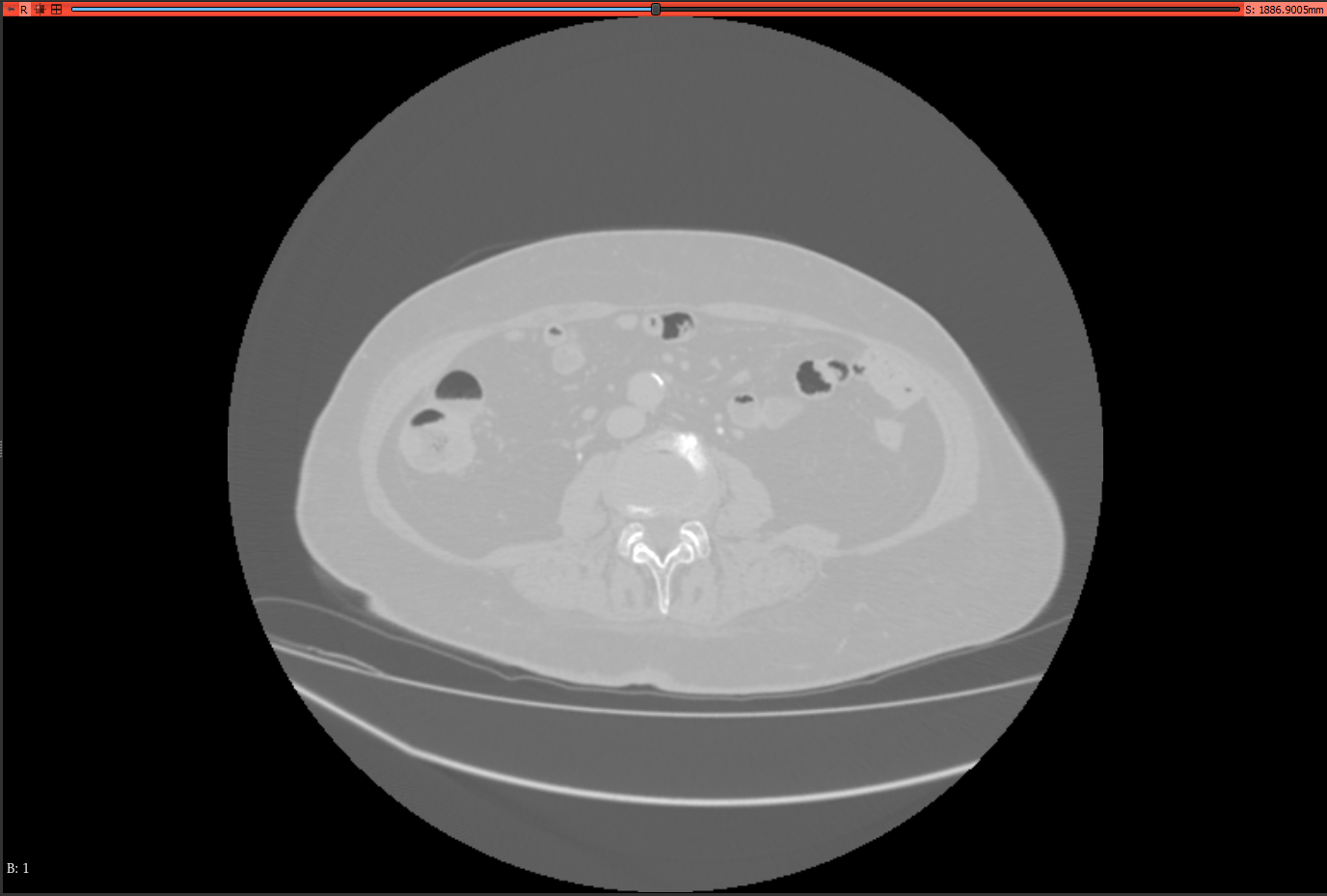
Примеры метаданных: положение пациента, вид проекции (снимка) – аксиальная, сагитальная или корональная, номер среза (под срезом подразумевается 1 снимок из серии), возраст пациента, пол пациента, данные о медицинской технике аппарате компьютерной томографии в данном случае), выдержка, фокусное расстояние, пиковое напряжение (для того, чтобы выделять те или иные ткани), изображение в пикселях (двумерный массив пикселей) и т.д. [9].

Для обработки DICOM-формата на языке Python существует несколько библиотек, самой оптимальной (и часто используемой) является библиотека «Pydicom» [10]. Но, для корректной обработки тэга «pixel\_array», необходимо также наличие модуля «gdcmswig».

В настоящее время, при работе с DICOM-форматом в медицинских учреждениях (больницах, клиниках и т.д.), используются специализированные программы, позволяющие просматривать не только метаданные, но и непосредственно снимки в формате изображений, при чем, как в двумерном формате (в разных проекциях), так иногда и в трехмерном пространстве.

Примером такой программы является «3D Slicer» - бесплатное приложение с открытым исходным кодом для визуализации, обработки, сегментации, регистрации и анализа медицинских, биомедицинских и других 3D-изображений [11].

Примеры визуализации DICOM-изображений (из датасетов) в программе «3D Slicer» представлены ниже. Пример визуализации аксиальной проекции приведен на рисунке 1.



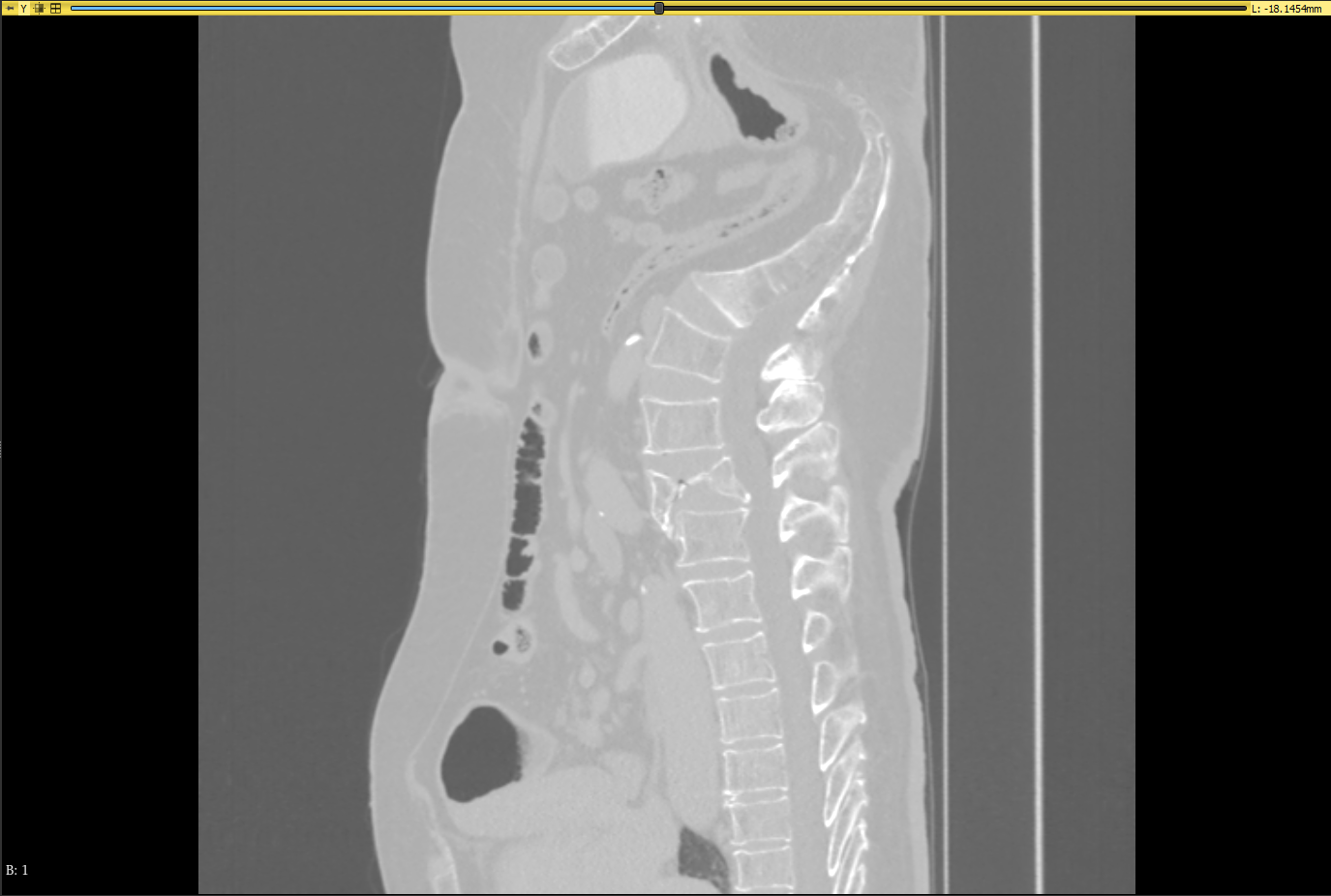
***Рисунок 1 – Пример визуализации аксиальной проекции в «3D Slicer»***

Пример визуализации корональной проекции приведен ниже, на рисунке 2.



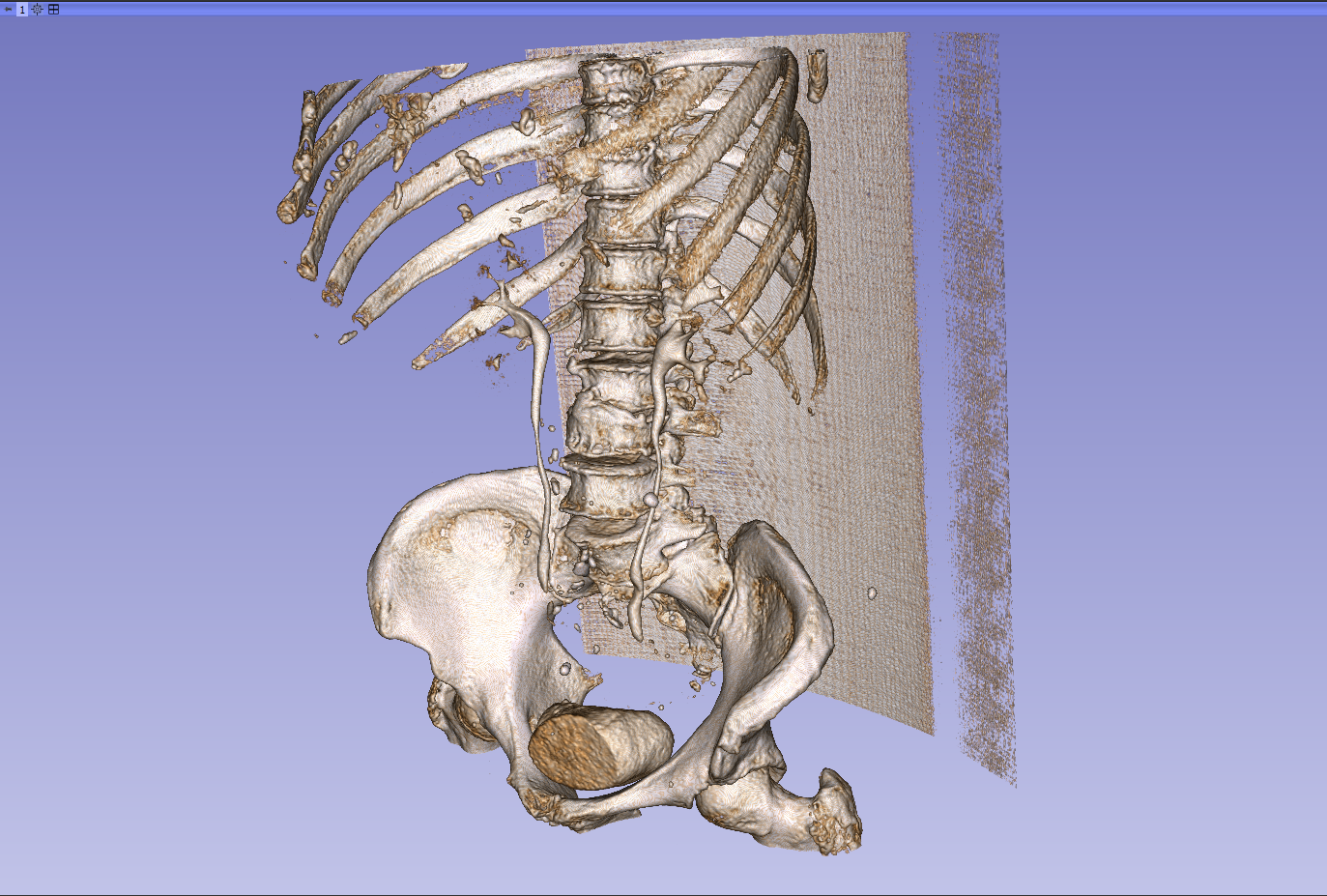
***Рисунок 2 – Пример визуализации корональной проекции в «3D Slicer»***

Визуализация сагитальной проекции изображена ниже, на рисунке 3.



***Рисунок 3 – Пример визуализации сагитальной проекции в «3D Slicer»***

Также, как уже было сказано ранее, «3D Slicer» позволяет создавать трехмерные проекции. Пример визуализации трехмерной проекции представлен ниже, на рисунке 4.



***Рисунок 4 – Пример визуализации трехмерной проекции в «3D Slicer»***

Существенным недостатком программы «3D Slicer» является то, что она достаточно тяжела в освоении, для использования продвинутого функционала требуется соответствующее обучение. Также, для корректной визуализации проекций (особенно трехмерной), необходима правильная сортировка файлов. По умолчанию, «3D Slicer» сортирует DICOM-файлы из указанного каталога (серию снимков) просто по их названиям, из-за чего порядок снимков становится неправильным (такая проблема наблюдается у всех серий КТ снимков из наборов данных от mosmed.ai). Для того, чтобы с этим бороться, можно перезаписать все файлы из серии, отсортировав их по тэгу «SliceLocation» (номер снимка/среза в списке срезов) или по тэгу «ImagePositionPatient» (номер снимка/среза в одной из проекций). Для визуализации примеров, представленных выше, такая сортировка и перезапись файлов была выполнена.

Пример визуализации различных проекций на ЯП Python, при помощи библиотек «Pydicom» и «Matplotlib», представлен ниже, на рисунке 5.



***Рисунок 5 – Пример визуализации различных проекций в Python***

При этом видно, что на данных проекциях костные ткани выделены ярче, нежели другие, мягкие ткани. Это было достигнуто при помощи специальных функций, встроенных в библиотеку «Pydicom». Речь о них пойдет далее, на этапе проектирования системы и предобработки данных.

Таким образом, DICOM-формат содержит в себе не только изображения КТ снимков, но и множество дополнительной информация, которая, в свою очередь, может быть использована как для преобразования самих изображений, построения разных проекций, так и в качестве дополнительной информации, обрабатываемой моделью.

* 1. **Анализ средств для разработки и обучения модели**

Разрабатывать и обучать модель можно было не только локально, используя ресурсы компьютера или ноутбука (использовался компьютер, технические характеристики будут описаны далее), но и используя среду разработку «Google Colab», используя, при этом, еще и облачные вычислительные мощности, предоставляемые компанией Google (аппаратные ускорители на базе ГПУ, а не ЦПУ, «TPU» и «T4 GPU»). Такие облачные вычислительные мощности позволяют существенно ускорить обучение модели, однако, имеют ограничение по вычислительным единицам. Это значит, что после определенного количества обучения модели, выбранный аппаратный ускоритель станет временно недоступным. При этом, может произойти внезапная остановка обучения модели, или же среды выполнения целиком, что доставляет существенные неудобства при многократном экспериментальном обучении модели, при подборе структуры модели и её гиперпараметров.

Также, при работе с «Google Colab» немного усложняется работа с данными, т.к. медицинские изображения в DICOM-формате, имеют очень большой вес (наборы данных весят десятки гигабайт), из-за чего загружать их в облако достаточно долго.

Как уже было сказано ранее, обучение модели происходит в разы быстрее при использовании мощностей ГПУ, т.к. архитектура графического процессора позволяет значительно лучше распараллеливать вычисления [12].

Рассматривая локальные ресурсы, было выяснено, что их может вполне хватить для разработки и дальнейшего обучения модели, т.к. комплектующие компьютеры оказались достаточно мощными. Обучение модели производилось на ГПУ «RTX 3070» (5888 CUDA-ядер). Также, использовался 8-ядерный процессор «Intel Xeon e5-2689», с тактовой частотой 3.3 ГГц. Медицинские изображения хранились локально, на твердотельном накопителе.

При дальнейшем сравнении скорости обучения на видеокарте и на процессоре, было выяснено, что обучение на видеокарте происходит примерно в 9 раз быстрее, чем на процессоре (в конкретно моем случае).

Таким образом, было решено разрабатывать и обучать модель в среде разработки «Visual Studio Code», с использованием интерактивных блокнотов «Jupyter Notebooks», используя при этом локальные ресурсы персонального компьютера (в первую очередь, вычислительные мощности графического процессора).

* 1. **Модели искусственного интеллекта, применяемые в медицине**

Как уже было сказано ранее, ИИ уже применяется в медицине для ускорения работы врачей. И если ранее модели ИИ чаще всего применялись для выявления групп риска тех или иных заболеваний (например, группа риска сердечно-сосудистых заболеваний на основе данных о потреблении людьми алкоголя, занятии спортом, возрасте и т.д.), на основе различных текстовых или числовых данных, то теперь же искусственный интеллект (особенно, сверточные нейронные сети) все чаще и чаще применяют для анализа различных изображений, от простых фотографий (например, фотографии кожи с подозрением на дерматологические заболевания) до снимков, сделанных специальной аппаратурой (МРТ, КТ, Рентген и другие), а также, трехмерных проекций. Последние чаще всего используются именно при работе с КТ снимками, но, несмотря на то, что трехмерные проекции серий снимков можно было бы построить, таких данных бы не хватило для обучения трехмерной сверточной нейронной сети.

Таким образом, для диагностики остеопороза по КТ снимкам позвоночника, было решено использовать именно сверточную нейронную сеть, при чем, именно предобученную модель.

1. **Обработка наборов данных**
   1. **Отбор и разметка данных для обучения системы**

Из-за того, что стандартные, размеченные наборы данных имели очень малое количество записей, было решено также использовать для обучения и записи из набора данных для проведения селф-тестирования. Были найдены 2 записи, судя по проекциям которых, патология отсутствовала. Эти записи были размечены самостоятельно.

Также, не были использованы записи из второго набора данных от ГБУЗ «НПКЦ ДиТ ДЗМ», у которых было отмечено наличие технических дефектов.

Другие наборы данных, не от ГБУЗ «НПКЦ ДиТ ДЗМ», содержащие КТ-снимки позвоночника с признаками остеопороза, не были найдены. Существуют наборы данных на «Kaggle» на несколько сотен гигабайт с КТ-снимками позвонков шейного с признаками переломов, но, эти наборы данных не являются подходящими (т.к. стандартно требуется работа с позвонками брюшной полости, и, наличие переломов позвонков не гарантирует наличие остеопороза. Также, в таком наборе данных тяжело получить информацию о плотности костной ткани).

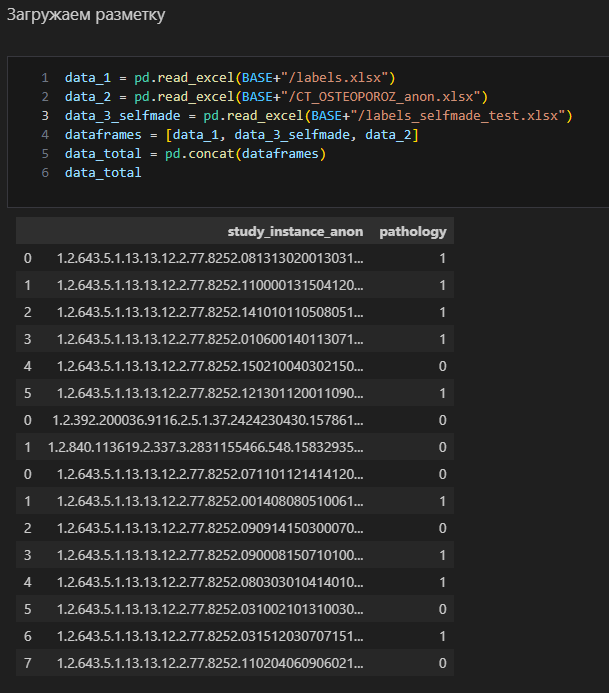
Разметка данных, как уже было сказано ранее, бинарная. Ключевым признаком является наличие патологии.

Также, из-за того, что записи в наборах данных включают в себя, в том числе, и схожие между собой серии КТ снимков (серии КТ снимков одного и того же пациента), было принято решение использовать для обучения модели снимки из первой серии каждой записи, чтобы уменьшить количество схожих изображений. В итоге, мы имеем около 2500 аксиальных изображений без признаков патологии, а с признаками патологии более около 4000. Часть из них будет использована для тестирования системы (не будет использоваться для обучения).

Стандартные размеры изображений (массивов) во всех сериях – 512 на 512 пикселей.

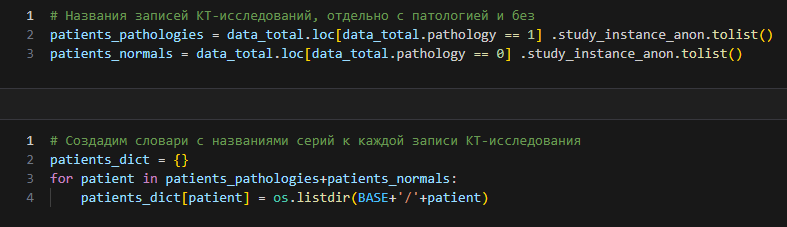
Сама разметка подгружается из нескольких Excel-таблиц, в которых название записи КТ исследования (столбец «study\_instance\_anon») и данные о наличии патологии (столбец «pathology»), 0 – признаки патологии не выявлены, 1 – признаки патологии выявлены. При этом, название записи КТ исследования соответствует названию каталога, в котором хранятся серии КТ снимков этой записи.

Пример загрузки разметки представлен на рисунке 6.



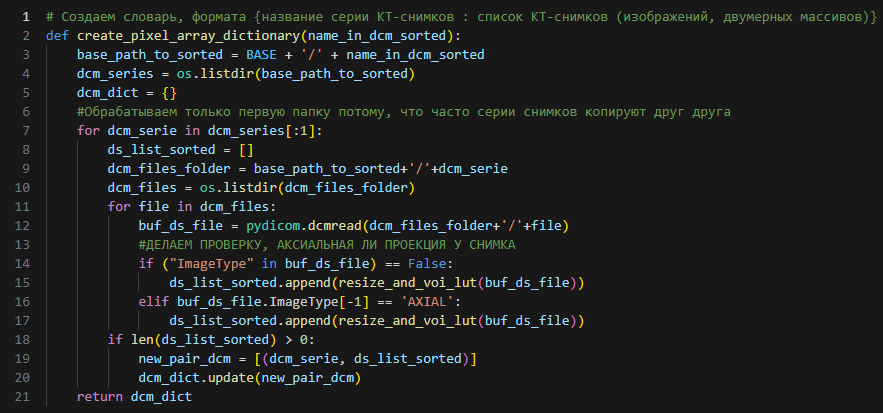
***Рисунок 6 – Пример загрузки разметки***

После этого, названия записей КТ исследований передаются в списки, отдельно для записей с признаками патологии и без признаков патологии. Далее, создаются словари с названиями серий к каждой записи КТ-исследования, при помощи которых далее данные и будут извлечены. Код, используемый для этой подготовки данных, изображен ниже, на рисунке 7.



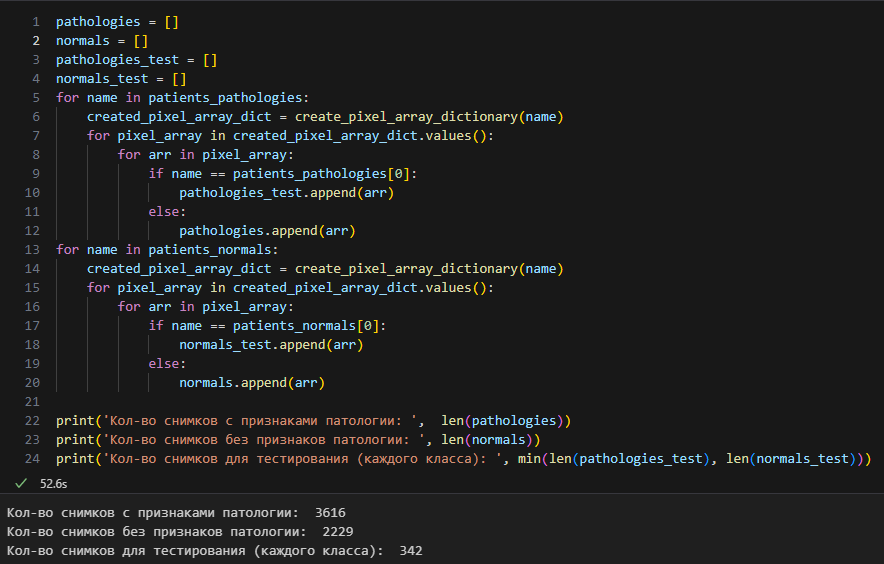
***Рисунок 7 – Код получения названий серий к каждой записи КТ исследования***

Далее, для работы непосредственно с самими изображения КТ снимков из каждой серии, при помощи функции «create\_pixel\_array\_dictionary», создается словарь, в котором название серии КТ снимков соотносится со списком изображений из этой серии. Код функции «create\_pixel\_array\_dictionary» представлен на рисунке 8. При этом можно заметить, что к изображениям применяется функция «resize\_and\_voi\_lut». Об этой функции будет рассказано далее.



***Рисунок 8 – Код функции «create\_pixel\_array\_dictionary»***

Затем, происходит извлечение снимков и их добавление в соответствующие общие списки (с признаками патологии и без признаков патологии), для дальнейшей обработки и использования в обучении модели. При этом, можно заметить, что снимки первой записи КТ исследования попадают в отдельные списки, предназначенные для финального тестирования модели, и не используются для её обучения. Код извлечения всех снимков представлен ниже, на рисунке 9.



***Рисунок 9 – Код извлечения всех изображений КТ снимков***

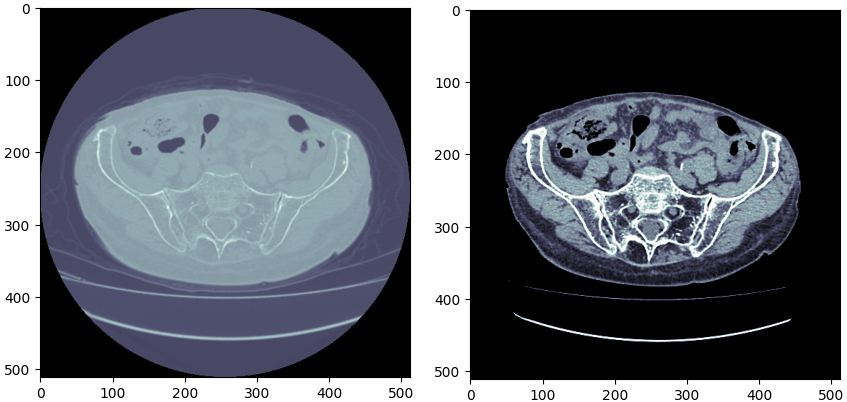
Таким образом, для обучения модели будет использовано около 2200, а для тестирования модели около 340 изображений каждого класса (с признаками и без признаков наличия остеопороза позвоночника).

* 1. **Предобработка данных**

В случае работы с медицинскими изображениями, важна и возможна не только базовая предобработка данных, такая как изменение размеров изображений, их нормализация и аугментация, для ускорения и улучшения качества обучения модели. Также, при помощи использования метаданных DICOM-формата и функций библиотеки «Pydicom», появляется возможность выделения тканей, рассматриваемых при постановке анализа. Это достигается при помощи использования функции «apply\_voi\_lut». Она преобразовывает исходное изображение в соответствии с данными из таблицы «VOI LUT», скрытой в метаданных. Благодаря этому, значительно ярче и контрастнее на изображении становятся костные ткани, необходимые для диагностики остеопороза. (Тип выделяемых тканей зависит от таблицы «VOI LUT». В зависимости от типа исследуемой патологии, могут выделять не только костные ткани, но и мягкие ткани, сосуды, внутренние органы и так далее.

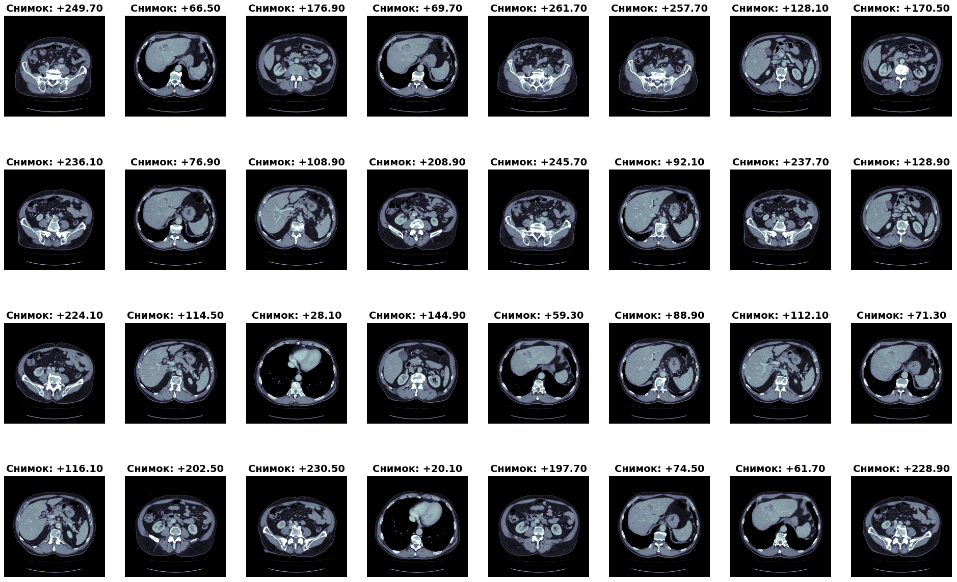
Получение двумерных массивов – изображений из DICOM файлов происходит через использование библиотеки «Pydicom», а вывод изображений и их частичная обработка, при помощи библиотек «matplotlib.pyplot» и «cv2» соответственно.

Пример работы функции «apply\_voi\_lut» представлен на рисунке 10. Слева изображение до обработки, справа – после. На изображении справа видно, что костные ткани стали значительно ярче и контрастнее.



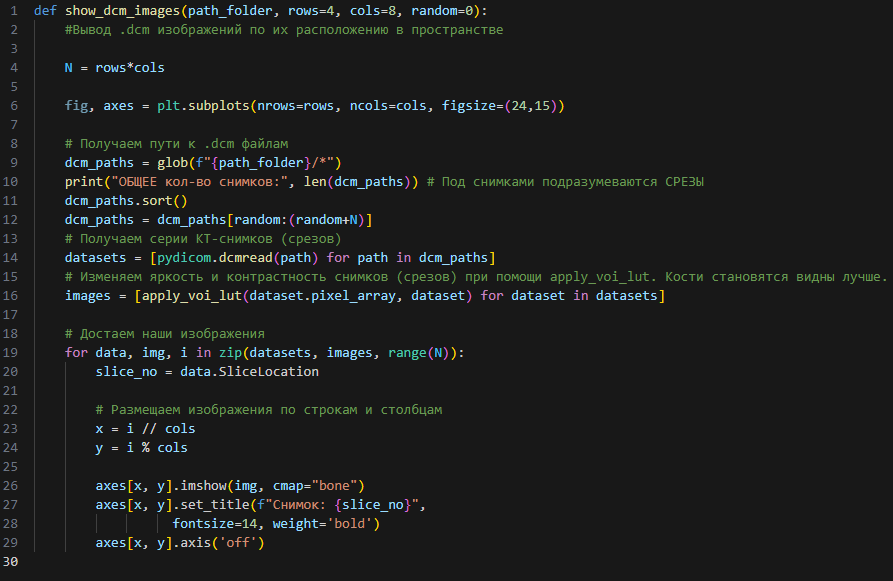
***Рисунок 10 – Пример работы функции «apply\_voi\_lut».***

Также, для дополнительной информативности, при разработке модели использовался вывод множества случайных аксиальных снимков, с указанием их положения (внутри серии). Пример такого вывода представлен ниже, на рисунке 11.



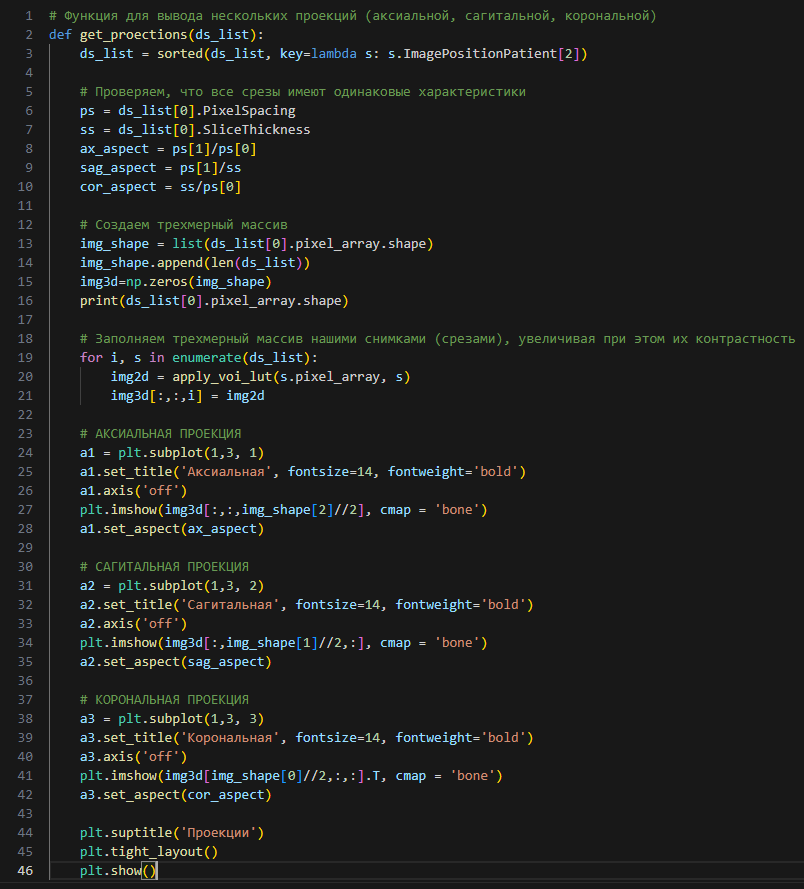
***Рисунок 11 – Пример вывода множества случайных аксиальных снимков***

Реализует такой вывод функция «show\_dcm\_images». Она получает на вход путь к серии КТ-снимков на диске и размеры сетки для снимков. Снимки из серии загружаются, к ним применяется функция «apply\_voi\_lut», после чего они размещаются по сетке и выводятся на экран. Код функции «show\_dcm\_images», реализующей такой вывод, представлен ниже, на рисунке 12.



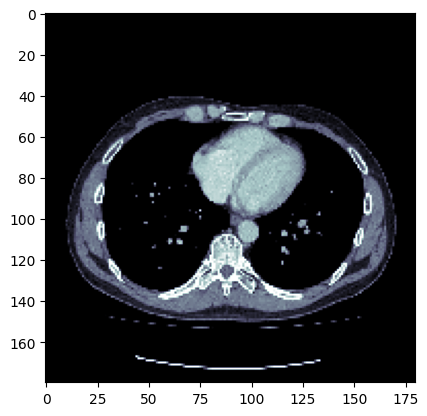
***Рисунок 12 – Код функции «show\_dcm\_images»***

Для вывода различных проекций серии КТ снимков, о которых было упомянуто ранее, на этапе анализа, была разработана функция «get\_proections», которая получает на вход список КТ-снимков в DICOM-формате, после чего из них выстраивается трехмерный массив пикселей, который уже после и преобразовывается в аксиальную, сагитальную и корональную проекции. Стоит отметить, что в данном случае очень важно отсортировать снимки по тэгу «SliceLocation» или же по «ImagePositionPatient[2]», иначе снимки будут перемешаны и проекции будут выстраиваться некорректно. В данной функции используется сортировка по второму тэгу. Код функции «get\_proections» представлен ниже, на рисунке 13.



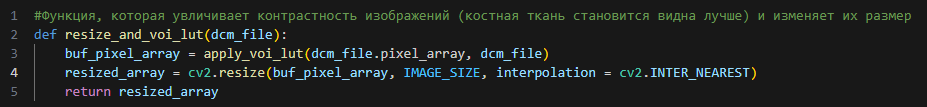
***Рисунок 13 – Код функции «get\_proections»***

Также, так как функция «apply\_voi\_lut» возвращает простой двумерный массив, его размер можно сразу же изменить, уменьшив в пределах 100-250 пикселей, для того, чтобы ускорять обучение нейронной сети и не вызывать переполнения оперативной памяти и видеопамяти ГПУ. Размер в 180 на 180 пикселей был выбран оптимальным размером изображений для обучения. Для одновременного применения функции «apply\_voi\_lut» и изменения размера изображений, была создана функция «resize\_and\_voi\_lut». Изображение, обработанное этой функцией, изображено ниже, на рисунке 14.



***Рисунок 14 – Изображение, обработанное функцией «resize\_and\_voi\_lut»***

Данная функция принимает на вход DICOM файл и возвращает преобразованное изображение в виде двумерного массива. Код функции «resize\_and\_voi\_lut» представлен на рисунке 15.

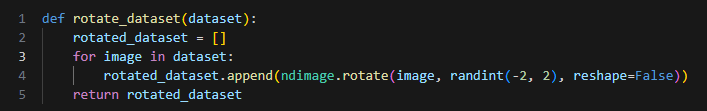


***Рисунок 15 – Код функции «resize\_and\_voi\_lut»***

Таким образом, использование особенностей DICOM-формата позволяет сделать специфическую обработку изображений из серии, а также, преобразовать их в различные проекции.

* 1. **Подготовка данных к использованию в обучении, аугментация**

Для борьбы с переобучением, была создана функция «rotate\_dataset», предназначенная для небольшого поворота изображений, в пределах 2 градусов. Несмотря на столь небольшой поворот изображений, такая аугментация данных всё же улучшает общее качество модели. Поворот изображений производится при помощи функции «rotate» из библиотеки «ndimage». Код функции «rotate\_dataset» изображен ниже, на рисунке 16.



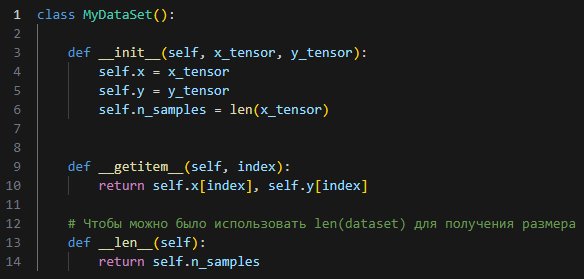
***Рисунок 16 – Код функции «rotate\_dataset»***

Сначала количество изображений каждого класса уравнивается, после чего все изображения поворачиваются случайным образом при помощи функции, описанной ранее. После этого, данные нормализуются, перемешиваются и преобразовываются в тензоры. Код, реализующий создание обработанных тензоров, изображен на рисунке 17.



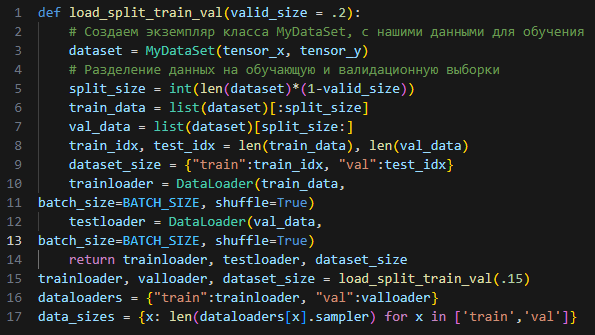
***Рисунок 17 – Код создания обработанных тензоров***

Далее, для того, чтобы использовать функционал класса «DataLoader» из библиотеки «PyTorch», был создан класс «MyDataSet», получающий на вход тензоры с изображениями и затем реализующий доступ к данным из тензора. Код класса «MyDataSet» представлен ниже, на рисунке 18.



***Рисунок 18 – Код класса «MyDataSet»***

Также, для удобного разделения данных на обучающую и валидационную выборки, была разработана функция «load\_split\_train\_val», которая делит данные по указанному соотношению и создает загрузчики данных класса «DataLoader». Данные в этих загрузчиках перемешиваются и собираются в батчи (оптимальный размер батча, который был подобран – 32). Из-за малого набора данных, для валидации использовалось только 15% данных. Код функции «load\_split\_train\_val» изображен на рисунке 19.



***Рисунок 19 – Код функции «load\_split\_train\_val»***

Таким образом, данные для обучения были нормализованы, аугментированы, преобразованы в тензоры и разделены на обучающую и валидационную выборки, с использованием батчей, размером 16.

1. **Разработка и обучение модели**
   1. **Выбор модели нейронной сети**

Чаще всего, по данным исследований [13], для классификации медицинских изображений используются такие предобученные сверточные нейронные сети, как ResNet50, ResNet152, DenseNet121, DenseNet161, VGG16, VGG19 и другие. Все они показывают очень высокую точность уже после десятка эпох обучения (в районе 90%), особенно, при использовании заранее подготовленных весов.

Фреймворк «PyTorch» позволяет использовать множество готовых моделей сверточных нейронных сетей вместе с готовыми весами, включая все те, которые были упомянуты выше. При этом, не получилось бы эффективно использовать слишком тяжелые модели, с слишком большим количеством параметров и требующие очень высокопроизводительные системы для обучения (исходя из данных о GFLOPS (кол-во операций с плавающей точкой в секунду)), потому что тогда эксперименты с моделью в ходе подбора гиперпараметров, вида нормализации и аугментации данных, попытки использования данных из сторонних наборов данных требовали бы слишком много времени (даже для обучения одной эпохи), а некоторые модели и вовсе не смогли бы обучаться локально, из-за недостатка вычислительных мощностей [14].

Данные о предобученных нейронных сетях в фреймворке «PyTorch» представлены ниже, в таблице 2.

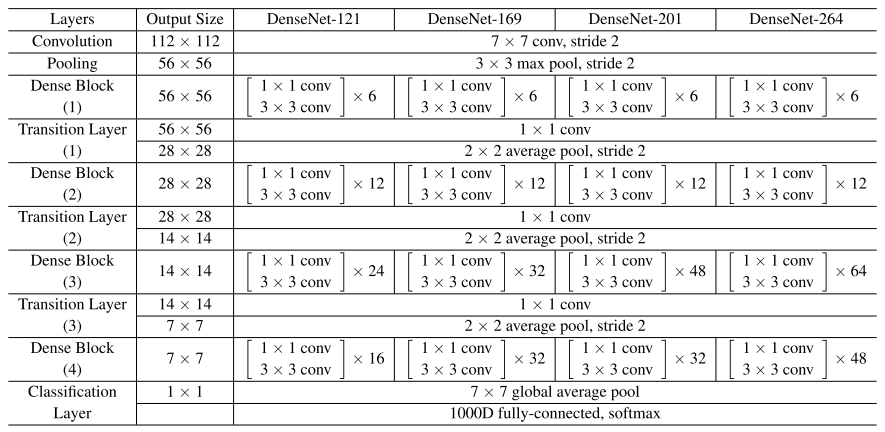
Таблица 2 – Информация о предобученных моделях фреймворка «PyTorch»

|  |  |  |  |
| --- | --- | --- | --- |
| **Название нейронной сети** | **Кол-во параметров, миллионы параметров** | **Требование к производительности, GFLOPS** | **Точность после 5 эпох на наборе данных «ImageNet-1K»** |
| AlexNet | 61.1 | 0.71 | 79.066 |
| VGG16 | 138.4 | 15.47 | 90.382 |
| VGG19 | 143.7 | 19.63 | 90.876 |
| ResNet50 | 25.6 | 4.09 | 92.862 |
| ResNet152 | 60.2 | 11.51 | 94.046 |
| DenseNet121 | 8.0 | 2.83 | 91.972 |
| DenseNet151 | 28.7 | 7.73 | 93.56 |

Одно из самых лучших соотношений количества параметров с требованиями и производительности к точности наблюдается у нейронных сетей архитектуры DenseNet, особенно у DenseNet121.

Модели архитектуры DenseNet были созданы в 2016 году и представляют из себя сверточные нейронные сети с измененной схемой соединения между слоями: были созданы компактно соединенные (dense) блоки, которые соединяют другие слои между собой.

Более подробная архитектура сверточных нейронных сетей DenseNet представлена ниже, на рисунке 20.

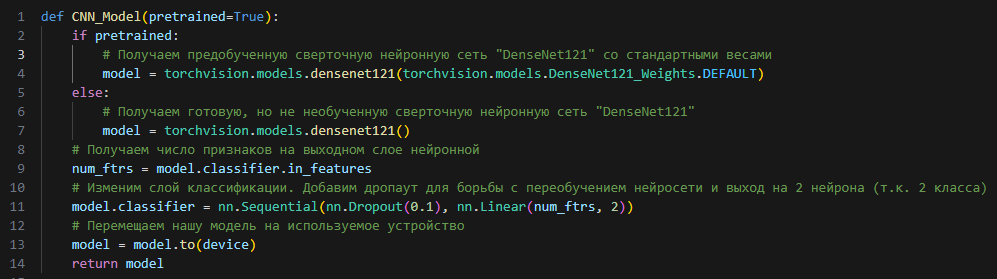


***Рисунок 20 – Архитектура нейронных сетей DenseNet***

Таким образом, в качестве самой оптимальной модели, по соотношению сложности модели – точность, была выбрана сверточная нейронная сеть DenseNet121.

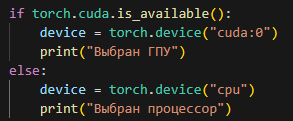
* 1. **Создание и обучение модели**

Для создания модели нейронной сети была создана функция «CNN\_Model», возвращающая модель DenseNet121 (с весами или без), с измененным блоком классификации, в который был добавлен слой Dropout (для борьбы с переобучением) и выходной слой на 2 нейрона, в соответствии с двумя классами (с признаками патологии и без признаков патологии). Затем, модель переносится на используемое устройство для обучения. Код функции «CNN\_Model» изображен на рисунке 21.



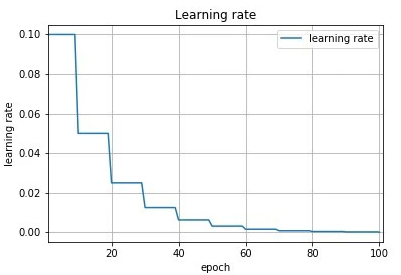
***Рисунок 21 – Код функции «CNN\_Model»***

Устройство, которое будет использовано при обучении, определяется кодом, который представлен на рисунке 22. В случае, если доступно обучение на CUDA-ядрах видеокарты, то вычисления модели будут осуществляться именно на ней. Иначе, в качестве доступного устройства будет задействован процессор.



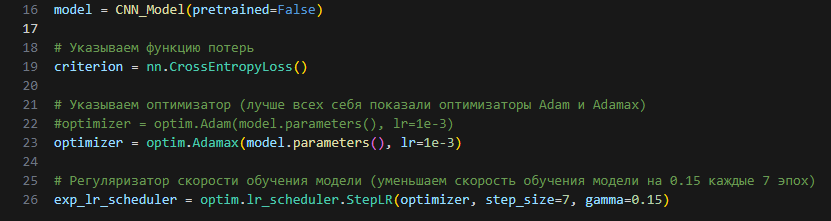
***Рисунок 22 – Код выбора доступного устройства для работы модели***

После создания модели, необходимо определить функцию потерь, оптимизатор и регуляризатор скорости обучения модели. В качестве функции потерь используется метрика Cross-entropy loss, подходящая для задач классификации. Среди оптимизаторов лучше всех себя показали «Adam» и «Adamax». В итоге всё же было отдано предпочтение оптимизатору «Adamax», т.к. с ним модель обучалась быстрее и имела более высокую точность. Регуляризатор скорости обучения модели – «StepLR», который уменьшает скорость обучения модели на 15% каждые 7 эпох (эти параметры также были подобраны экспериментально). Изменение скорости обучения происходит «ступенчато», пример работы такого регуляризатора показан ниже, на рисунке 23.



***Рисунок 23 – Визуализация работы регуляризатора StepLR***

Код, выполнение которого создает модель, функцию потерь, оптимизатор и регуляризатор скорости обучения, представлен ниже, на рисунке 24.



***Рисунок 24 – Код создания модели, функции потерь, оптимизатора и регуляризатора***

В созданной модели содержится 6.955.906 обучаемых параметров. Их веса будут меняться в ходе обучения модели.

Далее можно начинать обучение модели. Для этого была создана функция «train\_model» и списки, в которых будут сохраняться значения потерь и точности на каждой эпохе, для их дальнейшего анализа. Как уже было сказано ранее, в ходе обучения модели будет использоваться две фазы, обучающая, при которой веса модели будут изменяться, и валидационная, которая служит для оценки качества модели на каждой эпохе (в фазе валидации значения весов не корректируются).

Так как модель DenseNet стандартно работает с изображениями, которые используют цветовую модель RGB (то есть, изображения представляют из себя не один двумерный массив, а сразу 3, каждый из которых хранит информацию в соответствии с цветом), а изображения, извлекаемые из DICOM-формата, представлены в формате обычных двумерных массивов, их приходится преобразовывать, при помощи конкатинации тензоров, приводя к размеру (3, 180, 180).

Модель пробует сделать предсказания по данным, после чего эти предсказания сравниваются с правильными лейблами, высчитывается функция потерь. После этого, в фазе обучения, выполняется обратное распространение ошибки и инициализируется функция потерь. Веса модели корректируются.

Затем, обрабатываются данные о потерях и точности, происходит их вывод и сохранение в соответствующих списков, чтобы потом вывести графики.

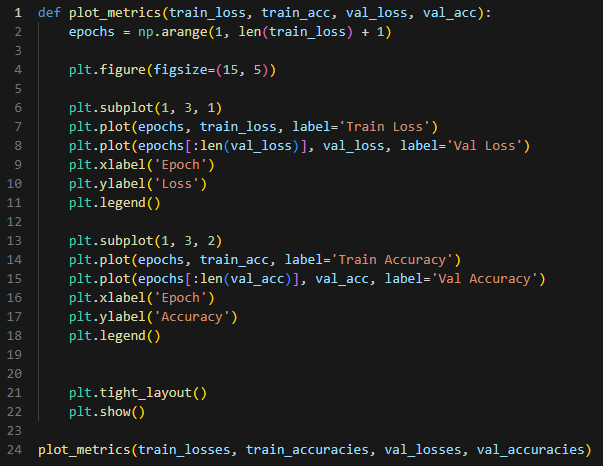
Если на фазе валидации потери обновили свой минимум, происходит сохранение весов. Модель с именно этими весами и будет возвращена в конце обучения.

В конце будет выведена информация о времени, потребовавшемся на обучение и лучший Loss (самое низкое значение потерь на фазе валидации). Основной код функции «train\_model» представлен ниже, на рисунке 25.



***Рисунок 25 – Код создания модели, функции потерь, оптимизатора и регуляризатора***

Вывод метрик осуществляется при помощи функции «plot\_metrics». Эта функция создает графики по данным точности и потерь в процессе обучения модели. Код функции «plot\_metrics» изображен ниже, на рисунке 26.



***Рисунок 26 – Код функции «plot\_metrcis»***

Обучение модели производилось более ста раз, с перебором параметров, от размера изображений и размера батчей, до смены оптимизаторов и функций потерь. Одним из самых удачным набором параметров стал: размер батчей 16 (небольшой размер батчей также способствовал уменьшению переобучения), оптимизатор «Adamax», размер изображений 180 на 180, размер Dropout’а 0.1, шаг оптимизатора 7, гамма оптимизатора 0.15, стандартные веса предобученной модели.

На этих параметрах модель была обучена 201 эпоху. Обучение длилось несколько часов. По результатам обучения были выведены графики, представленные ниже, на рисунке 27.

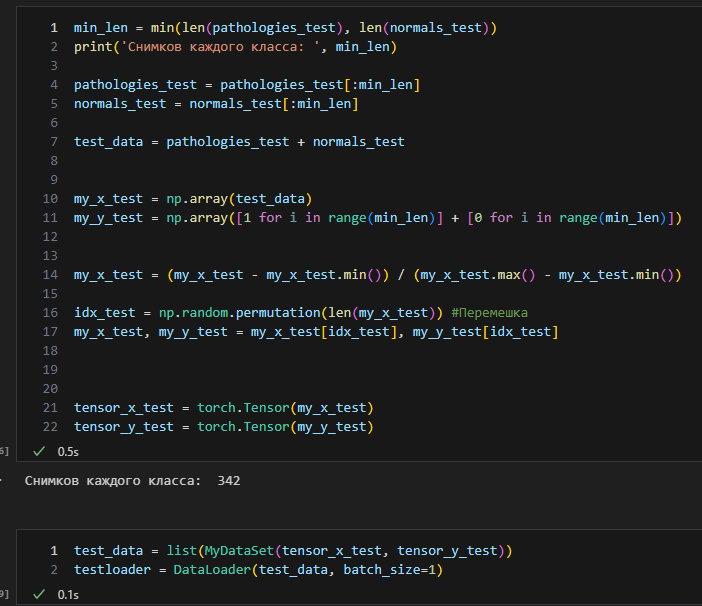


***Рисунок 27 – Графики по результатам обучения модели на 201 эпохе***

Таким образом, после множества экспериментов с моделью, были подобраны оптимальные параметры, на основе которых произвелось обучение на 201 эпохе. Данная модель была сохранена. Тестирование модели будет описано далее.

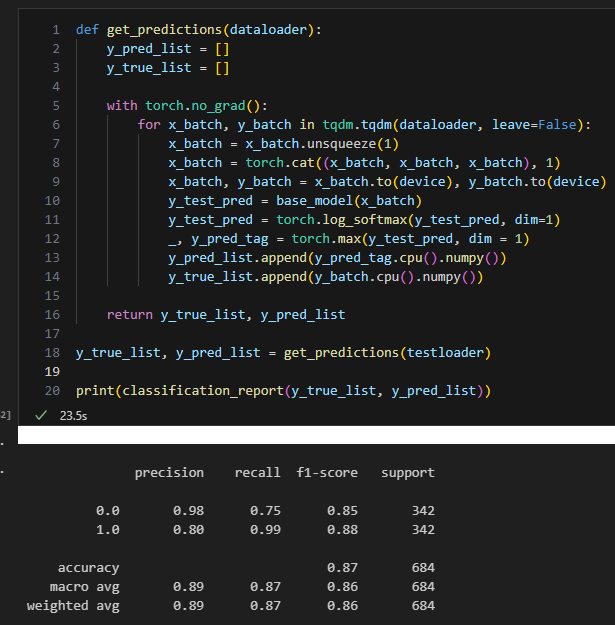
* 1. **Тестирование модели**

Тестирование проводилось по первым двум записям, которые не использовались при обучении и их изображения были сохранены в списках «pathologies\_test» и «normals\_test». Эти данные были обработаны, нормализованы и преобразованы в тензоры. Затем к ним был создан загрузчик данных, который использовался при тестировании. Код обработки тестовых КТ снимков и создание загрузчика данных к ним представлен ниже, на рисунке 28.



***Рисунок 28 – Обработка тестовых КТ снимков и создание загрузчика данных к ним***

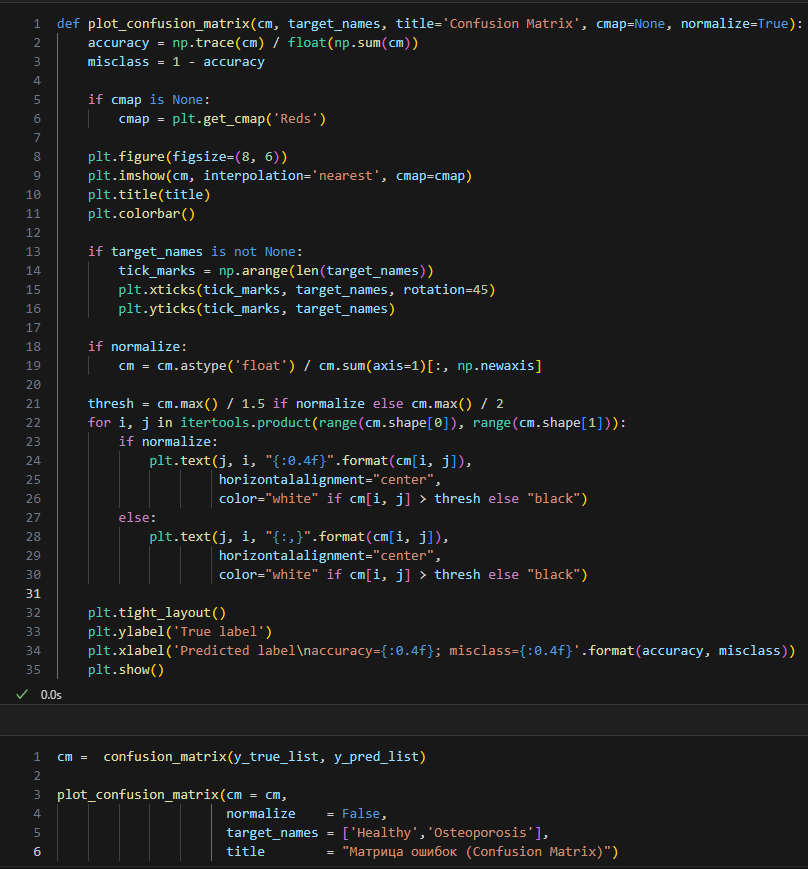
Тестовые изображения были переданы модели для обработки в функцию «get\_predictions», которая возвращает список лейблов, предсказанных моделью список истинных лейблов. После чего результаты, представленные моделью, были сравнены с истинными лейблами, с выводом соответствующих метрик при помощи функции «classification\_report» из библиотеки «sklearn». Код тестирования с выводом метрик и, непосредственно, сами значения метрик, изображены ниже, на рисунке 29.



***Рисунок 29 – Код функции «get\_predictions» и результаты тестирования с выводом метрик***

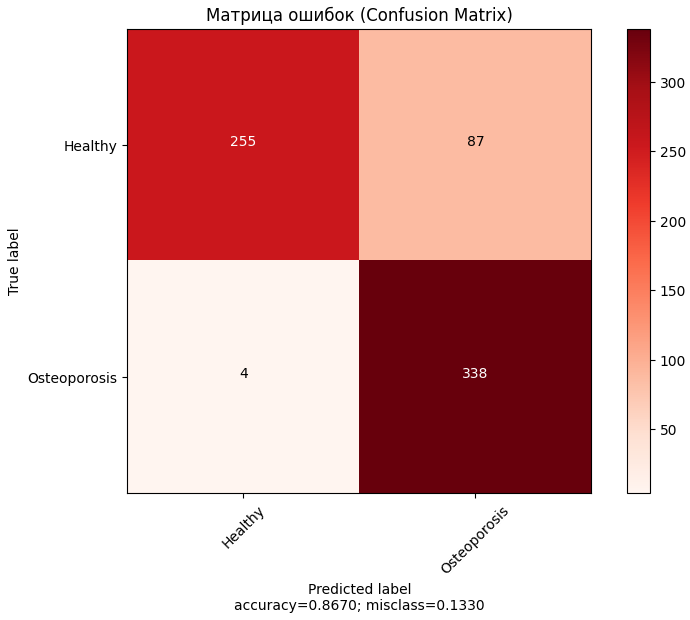
Исходя из метрик, модель имеет не только достаточно высокую общую точность (87%), но, также, большой показатель Recall, что особенно важно в задачах диагностики заболевания. Несмотря на то, что модель иногда ошибается, обозначая снимки без патологии как снимки с патологией, намного важнее то, что она редко делает обратные ошибки (отмечает снимки с патологией, как снимки без патологии).

Также, для тестирования была рассмотрена матрица ошибок. Для получения данных о матрице ошибок, была использована функция «confusion\_matrix» из библиотеки «sklearn». Также, была создана функция «plot\_confusion\_matrix», для построения графика матрицы ошибок. Код функции «plot\_confusion\_matrix» и код создания матрицы ошибок представлены ниже, на рисунке 30.



***Рисунок 30 – Код функции «plot\_confusion\_matrix» и код создания матрицы ошибок***

Матрица ошибок, построенная по результатам тестирования, изображена ниже, на рисунке 31.



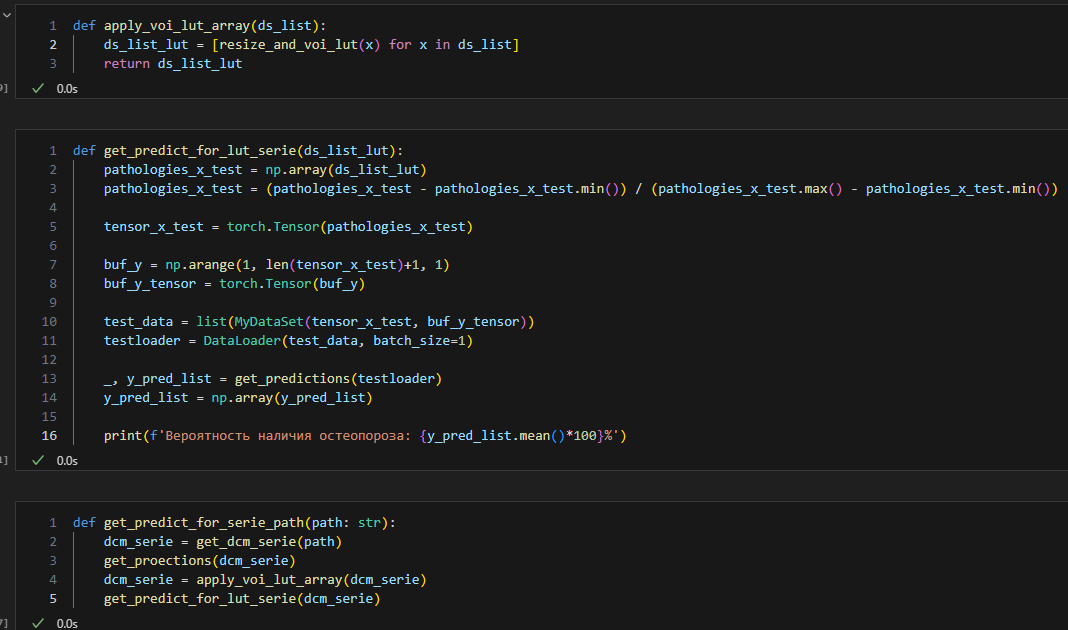
***Рисунок 31 – Матрица ошибок по тестовым данным***

По данным из графика можно еще раз убедиться в том, что модель достаточно точно определяет наличие остеопороза, но, периодически совершает ошибки, отмечая наличие патологии на снимках, где признаки патологии отсутствуют.

Таким образом, была получена достаточно точная модель для диагностики остеопороза по КТ снимкам позвоночника, которая была протестирована по различным метрикам.

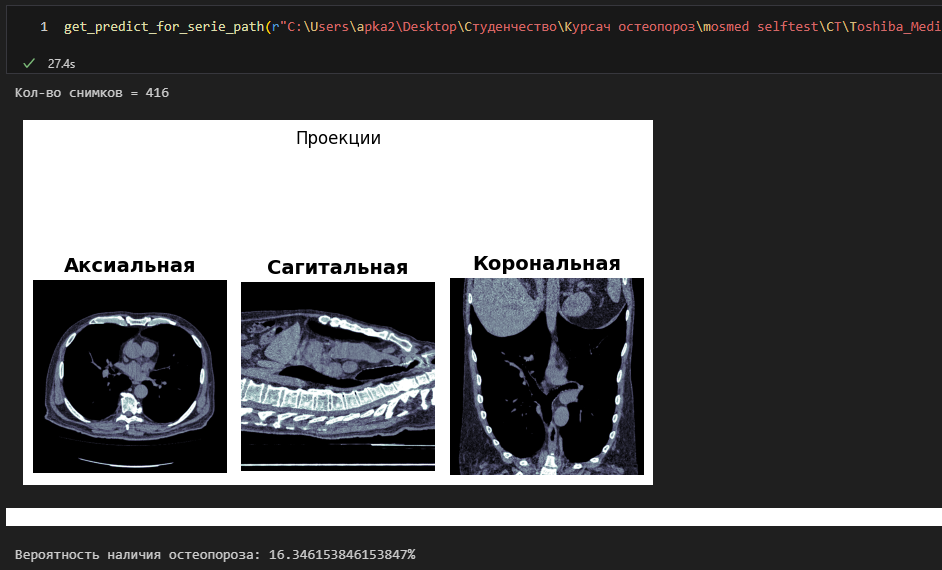
* 1. **Проверка совместимости системы с записями для селф-тестирования**

Систему необходимо также протестировать на совместимость с записями из набора данных, предназначенного для селф-тестирования от ГБУЗ «НПКЦ ДиТ ДЗМ». Для этого были созданы функции для получения данных по их адресу, их предобработке, созданию тензора и загрузчика данных: функции «get\_dcm\_serie», «apply\_voi\_lut\_array» и «get\_predict\_for\_lut\_serie» соответственно. Затем все они были объединены в функции «get\_predict\_for\_serie\_path», которая получает данные по адресу, строит по ним проекции, выполняет предобработку изображений и выводит вероятность наличия остеопороза. Код этих функций представлен ниже, на рисунке 32.



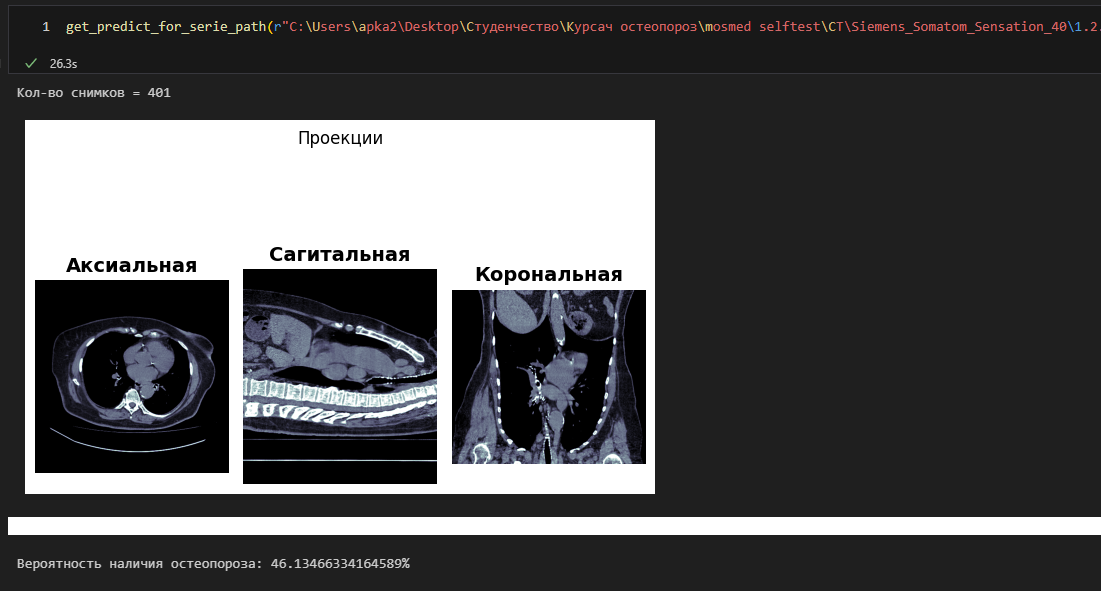
***Рисунок 32 – Код функций «apply\_voi\_lut\_array», «get\_predict\_for\_lut\_serie» и «get\_predict\_for\_serie\_path»***

Функционирование системы было проверено на записях из набора данных, предназначенного для проведения селф-тестирования. Система успешно смогла обработать данные, вывести проекции снимков и вероятность наличия остеопороза в процентах. Первый пример работы системы с этими записями приведен ниже, на рисунках 33.



***Рисунок 33 – Первый пример работы системы с записью из набора данных для селф-тестирования***

Второй пример работы системы с записью из набора данных для селф-тестирования изображен на рисунке 34.



***Рисунок 34 – Второй пример работы системы с записью из набора данных для селф-тестирования***

Таким образом, модель является совместимой с записями из набор данных для селф-тестирования от ГБУЗ «НПКЦ ДиТ ДЗМ», и потенциально, может использоваться в их информационной среде.

1. **Разработка приложения**
   1. **Диаграммы активностей и прецедентов**

Так как приложение является узконаправленным, то для него удалось выделить всего лишь один прецедент, описывающий использование системы. Его описание и диаграмма были составлены в соответствии с методологией UML [15] 9 из отчета.

Название прецедента: получение информации о серии КТ снимков.

Актор: медицинский работник (пользователь).

Краткое описание: пользователь загружает серию КТ снимков позвоночника формата DICOM и получает ответ от системы в виде вероятности наличия остеопороза, а также, изображения – проекции серии КТ снимков.

Триггер: пользователь загружает файлы (серию КТ снимков)

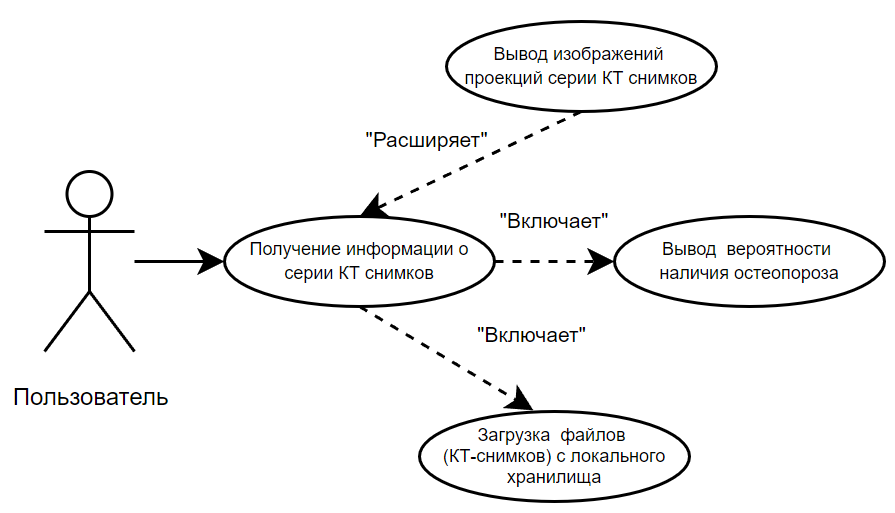
Альтернативный поток: системе не удалось обработать файлы. Выводится сообщение о том, что произошла ошибка.

Основной поток описан в таблице 3.

Таблица 3 – Прецедент «получение информации о серии КТ снимков»

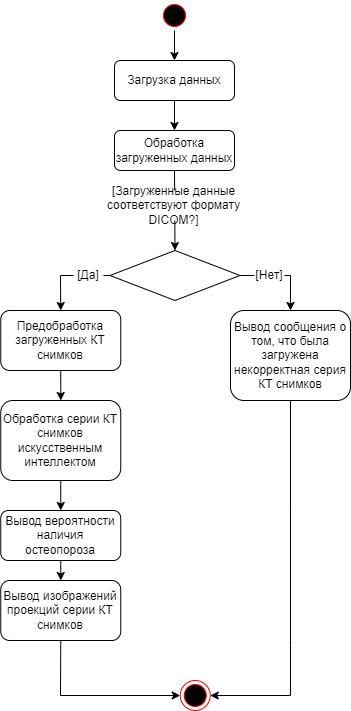
|  |  |
| --- | --- |
| **Действия акторов** | **Отклик системы** |
| Пользователь загружает файлы (серию КТ снимков) | Обрабатывает файлы, загруженные пользователем, выводит вероятность наличия остеопороза на загруженных КТ снимках и изображения проекций серии КТ снимков |

Диаграмма рассмотренного прецедента изображена ниже, на рисунке 35.



***Рисунок 35 – Диаграмма прецедента «получение информации о серии КТ снимков»***

Затем, по этому прецеденту была создана диаграмма бизнес-процесса, также в соответствии с методологией UML. Диаграмма бизнес-процесса по прецеденту «получение информации о серии КТ снимков» представлена ниже, на рисунке 36.



***Рисунок 36 – Диаграмма бизнес-процесса прецедента «получение информации о серии КТ снимков»***

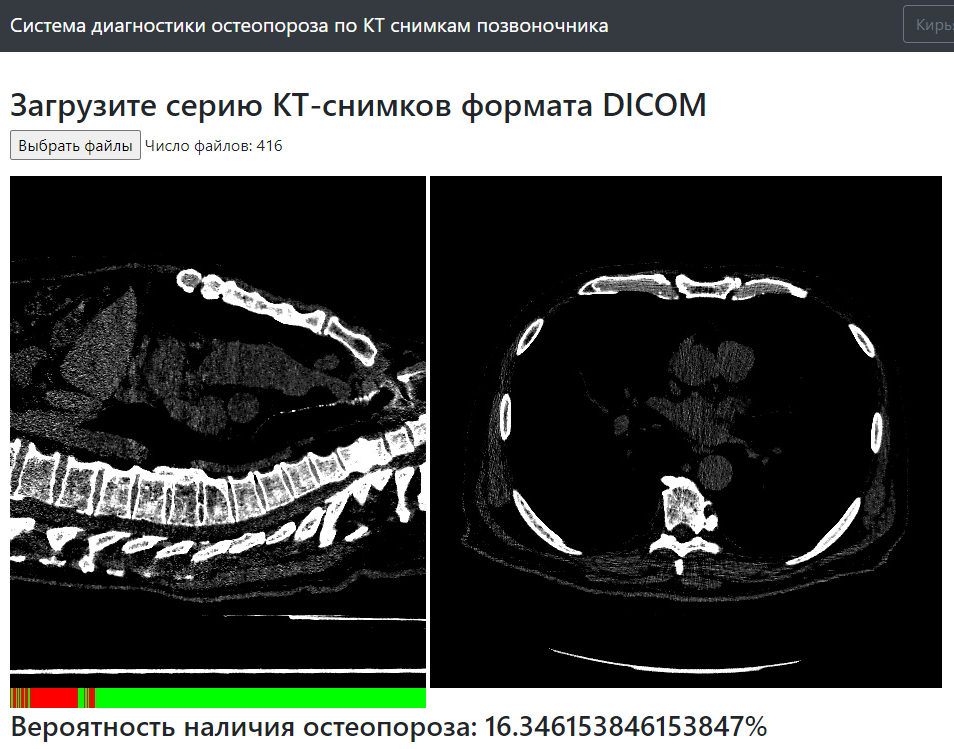
Таким образом, был рассмотрен основной (единственный) прецедент использования системы. Как уже было сказано ранее, система является узконаправленной, поэтому прецедент был выделен единственный прецедент.

* 1. **Проектирование пользовательского интерфейса**

Платформой веб-приложения стал фреймворк Flask [16]. Было создано одностраничное приложение, позволяющее загружать серию КТ снимков (в формате DICOM), обрабатывать искусственным интеллектом загруженные данные и выводить результат обработки – вероятность наличия остеопороза и проекции КТ снимков.

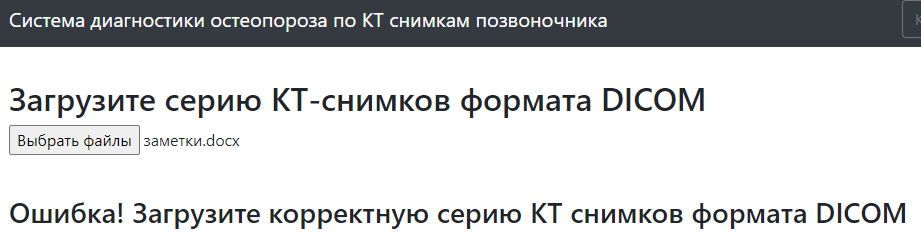
Также, на этапе проектирования было принято решение сделать примитивную сегментацию – под изображением сагитальной проекции будет выводиться изображение, указывающее на области, в которых система обнаружила признаки остеопороза.

То, как выглядит страница с результатом обработки серии КТ снимков позвоночника представлено ниже, на рисунке 37.



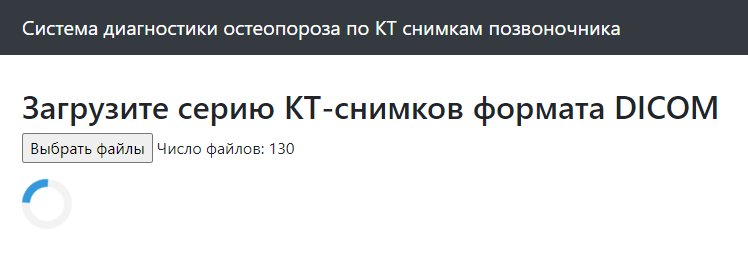
***Рисунок 37 – Результат обработки серии КТ снимков позвоночника***

Если системе не удалось обработать файлы, загруженные пользователем, то будет выведено соответствующее сообщение об ошибке. Пример вывода сообщения об ошибке представлен ниже, на рисунке 38.



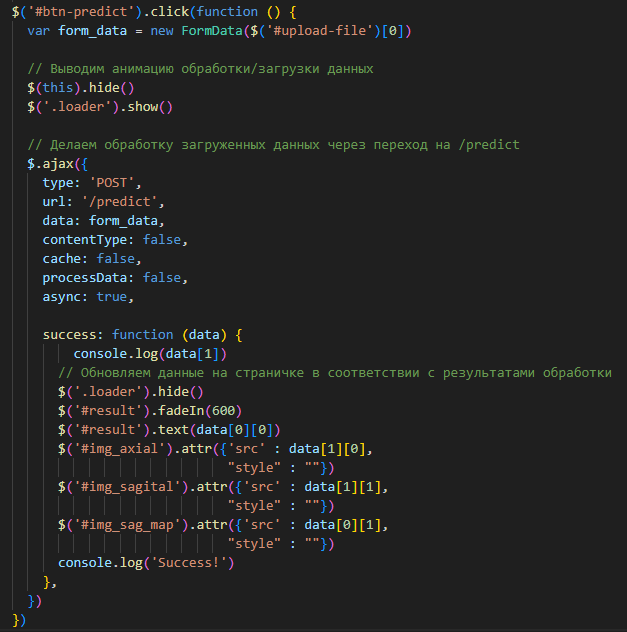
***Рисунок 38 – Вывод ошибки при неудачной обработке загруженных файлов***

Так как серии КТ снимков зачастую имеют достаточной большой объем (и количество), то их обработка может занимать немало времени, вплоть до пары минут. Для того, чтобы дать пользователю понять, что в настоящее время происходит обработка загруженных данных, была добавлена анимация загрузки (окружность с крутящимся сегментом). Примерный вид страницы во время обработки загруженных данных представлен на рисунке 39.



***Рисунок 39 – Примерный вид страницы во время обработки загруженных данных***

Как уже было сказано ранее, загрузка данных и вывод результатов их обработки происходит динамически, без перезагрузки страницы. Происходит это при помощи функций на языке программирования JavaScript. Новые полученные данные отправляются на страницу при помощи технологии «AJAX». Например, после того, как пользователь нажмет на кнопку «Анализ», начнется обработка загруженных данных нейронной сетью, после чего данные, которые вернет модель, будут отправлены элементам разметки. Код, реализующий этот функционал, представлен ниже, на рисунке 40.



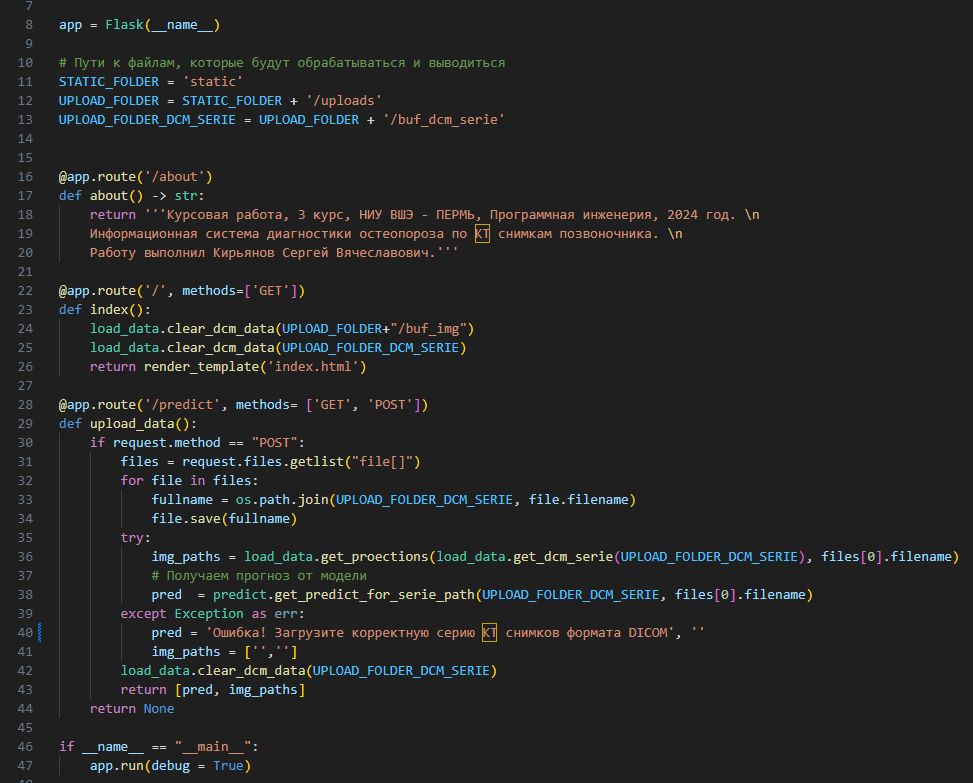
***Рисунок 40 – Пример динамического отображения результатов обработки серии КТ снимков моделью***

Таким образом, было создано одностраничное приложение с динамической обработкой данных, не требующей перезагрузку страницы. Также, помимо стандартного вывода вероятности наличия остеопороза и проекций серии КТ снимков (аксиальной и сагитальной), была добавлена сегментация данных для сагитальной проекции.

* 1. **Разработка внутренней логики приложения**

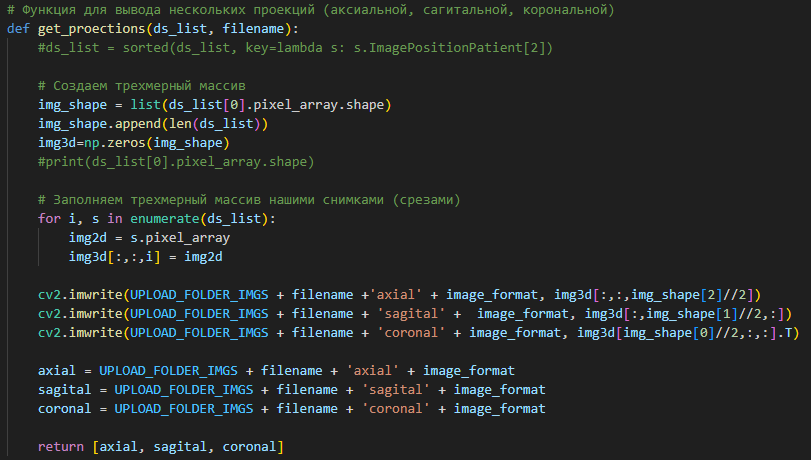
Внутренняя логика приложения реализуется в трёх модулях: «app.py», «predict.py» и «load\_data.py». Первый отвечает за навигацию по приложению и за обработку действий пользователя. Второй отвечает за обработку данных нейронной сетью и предоставление результатов этой обработки. Последний модуль отвечает за загрузку и предобработку данных (серии КТ снимков).

Начнем с модуля «app.py». При загрузке основной страницы приложения, очищается данные, которые загружались ранее. При переходе по эндпоинту «/predict» (после выбора данных пользователем), происходит их загрузка на серверную часть. Фактически, файлы копируются на сервер. Далее происходит получение проекций загруженных КТ-снимков и получение прогноза модели по этим снимкам. Код модуля «app.py» представлен на рисунке 41.



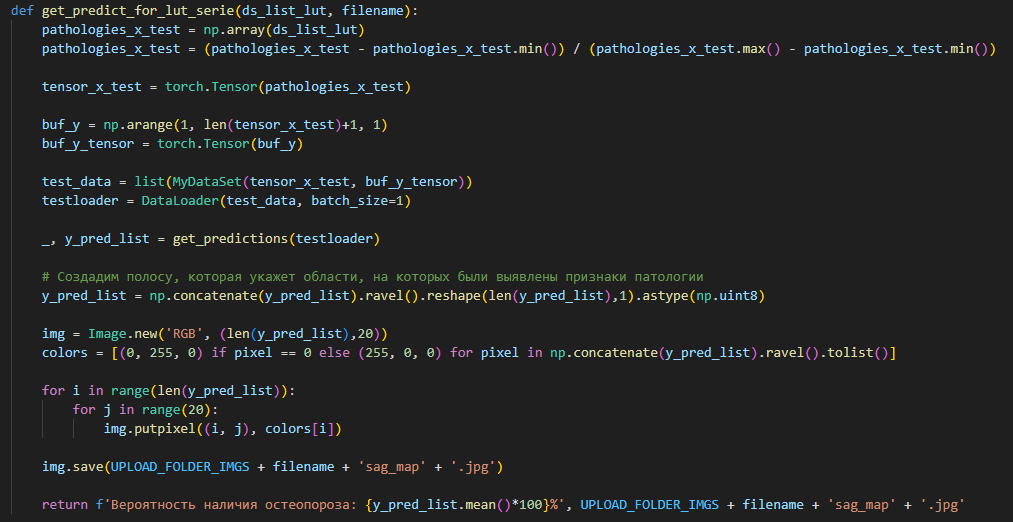
***Рисунок 41 – Код модуля «app.py»***

В модуле «load\_data.py» используются те же функции, что и при обучении и тестировании модели, и которые уже были рассмотрены ранее, в главе «Разработка и обучение модели». Единственная разница в том, что теперь проекции сохраняются также и в файлы, для дальнейшего вывода на странице. Код функции «get\_proections» из модуля «load\_data.py» изображен ниже, на рисунке 42.



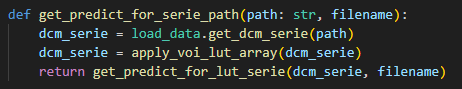
***Рисунок 42 – Код функции «get\_proections» из модуля «load\_data.py»***

В модуле «predict.py» также используются те же функции, что и при разработке и обучении модели, за исключением некоторых изменений, которые будут рассмотрены далее. создается экземпляр модели нейронной сети, обученной ранее (загружается из файла). Сначала происходит предобработка данных, после чего они обрабатываются нейронной сетью, которая возвращает вероятность наличия остеопороза. В функции «get\_predict\_for\_lut\_serie» было добавлено создание изображения для сегментации, о которой было сказано ранее. В случае, если модель отметила на снимке наличие патологии, он будет отмечен пикселем красного цвета, иначе – зеленого. Код функции «get\_predict\_for\_lut\_serie» представлен ниже, на рисунке 43.



***Рисунок 43 – Код функции «get\_predict\_for\_lut\_serie» из модуля «predict.py»***

Код функции «get\_predict\_for\_serie\_path», вызываемой непосредственно из модуля «app.py», изображен на рисунке 44.



***Рисунок 44 – Код функции «*** ***get\_predict\_for\_serie\_path » из модуля «predict.py»***

Таким образом, при помощи функций, разобранных в этой главе, была реализована основная логика веб-приложения, соответствующая функционалу, реализованному ранее, при разработке модели искусственного интеллекта.

* 1. **Тестирование**

Тестирование веб-приложения проводилось по методике черного ящика. В основном, тестировалась загрузка и обработка данных. Данные о результатах тестирования представлены в виде таблиц. Таблица тестов представлена ниже, в таблице 4.

Таблица 4 – Таблица тестов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Количество загружаемых данных** | **Тип данных** | **Ожидаемый выход** | **Реальный выход** |
| **Т1** | 0 | - | Ошибка! Загрузите корректную серию снимков формата DICOM | Ошибка! Загрузите корректную серию снимков формата DICOM |
| **Т2** | 1 | Соответствует формату DICOM | Вывод вероятности, изображений проекций снимков, изображения-сегментации | Вывод вероятности, изображений проекций снимков, изображения-сегментации |
| **Т3** | 5 | Соответствует формату DICOM | Вывод вероятности, изображений проекций снимков, изображения-сегментации | Вывод вероятности, изображений проекций снимков, изображения-сегментации |
| **Т4** | 1 | Не соответсвует формату DICOM | Ошибка! Загрузите корректную серию снимков формата DICOM | Ошибка! Загрузите корректную серию снимков формата DICOM |
| **Т5** | 1 | Соответствует формату DICOM | Вывод вероятности, изображений проекций снимков, изображения-сегментации | Вывод вероятности, изображений проекций снимков, изображения-сегментации |
| **Т6** | 100 | Частично не соответствует формату DICOM | Ошибка! Загрузите корректную серию снимков формата DICOM | Ошибка! Загрузите корректную серию снимков формата DICOM |

Результаты проверки достаточности тестов по критериям черного ящика находятся ниже, в таблице 5.

Таблица 5 – Таблица проверки достаточности тестов по критериям черного ящика

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ЧЯ** | **Тесты** | | | | | |
| **Критерии** | **Т1** | **Т2** | **Т3** | **Т4** | **Т5** | **Т6** |
| **1. Кол-во данных** | | | | | | |
| **1.1 0** | **+** |  |  |  |  |  |
| **1.2 1** |  | **+** |  | **+** | **+** |  |
| **1.3 > 1** |  |  | **+** |  |  | **+** |
| **2. Соответствие данных DICOM формату** | | | | | | |
| **2.1 Все данные не соответствуют DICOM формату** |  |  |  | **+** |  |  |
| **2.2 Данные частично не соответствуют DICOM формату** |  |  |  |  |  | **+** |
| **2.3 Все данные соответствуют DICOM формату** |  | **+** | **+** |  | **+** |  |
| **3. Результат работы системы** | | | | | | |
| **3.1 Вывод ошибки** | **+** |  |  | **+** |  | **+** |
| **3.2 Вывод вероятности, изображений проекций снимков, изображения-сегмантации** |  | **+** | **+** |  | **+** |  |

Таким образом, веб-приложение было успешно протестировано по методологии черного ящика. Тестирование позволило найти и устранить ошибки, образовавшиеся в процессе разработки.

* 1. **Испытания системы и ее внедрение**

В качестве проверки работоспособности системы на других устройствах, был созданный удаленный репозиторий на веб-сервисе для хостинга IT-проектов «GitHub» с исходными файлами проекта, после чего созданный репозиторий был скопирован на другое устройство (ПК). Далее из клонированного репозитория локально было успешно запущено приложение.

Также, код, использованный для разработки и обучения нейронной сети, представленный в виде блокнота «Jupyter Notebook», может быть запущен практически на любом устройстве при помощи сервиса «Google Colaboratory».

Ссылки на репозиторий и блокнот на сервисе «Google Colaboratory» представлены в приложении Е.

Таким образом, была проверена совместимость разработанной информационной системы с другими устройствами

1. **Заключение**

В процессе выполнения курсовой работы была поставлена цель разработать систему диагностики остеопороза по КТ снимками позвоночника. Для достижения этой цели были решены следующие задачи:

1. выполнен анализ предметной области, в том числе проанализирована клиническая задача, решаемая системой и признаки исследуемой патологии (остеопороза); проанализированы требования к формату ответа системы и форме его представления; осуществлен поиск и анализ данных, подходящих для обучения системы; произведен анализ DICOM-формата, а также рассмотрены существующие модели искусственного интеллекта (нейронных сетей), применяемые в медицине;
2. выполнен отбор данных для обучения системы; подготовлена разметка данных; реализована предобработка медицинских изображения для обучения системы; выбрана структура и модель нейронной сети (выбор был обоснован);
3. разработана и обучена модель нейронной сети; произведено её тестирование на размеченном наборе данных, с учетом различных метрик; проверена совместимость системы с записями из набора данных, предназначенного для селф-тестирования от ГБУЗ «НПКЦ» ДиТ ДЗМ, реализовано предоставление ответа системы для серии снимков в виде вероятности наличия патологии (в процентах);
4. осуществлена разработка веб-приложения; спроектирован пользовательский интерфейс; веб-приложение протестировано; проведены испытания системы и её внедрение.

Таким образом, цель можно считать достигнутой. Итоговое веб-приложение полноценно функционирует и потенциально может быть использована в информационной среде ГБУЗ «НПКЦ» ДиТ ДЗМ, т.к. имеет совместимость с их данными.

Тем не менее, остается простор для совершенствования системы. От интеграции с информационной средой ДЗМ, до дообучения системы на новых данных, добавления возможности распознавания наличия технических дефектов на снимках и полноценной сегментация областей, имеющих признаки остеопороза.

Во время выполнения курсовой работы были развиты навыки проектирования, реализации, тестирования и отладки программных систем, приобретен опыт работы с машинным обучением, в частности, с нейронными сетями, их применении в реальных задачах; закреплены навыки разработки документации и оформления отчетов о проделанной работе.

1. **Список используемой литературы**
2. Яблучанский Н. И., Лысенко Н. В. Остеопороз. В помощь практическому врачу, 2011 – 172 с. // Дата обращения: 01.11.2023
3. ScienceDirect, A reference standard for the description of osteoporosis. [Электронный ресурс] // URL: <https://www.sciencedirect.com/science/article/abs/pii/S875632820700823X?via%3Dihub> Дата обращения: 02.11.2023
4. PubMed, SCOPE 2021: a new scorecard for osteoporosis in Europe. [Электронный ресурс] // URL: <https://pubmed.ncbi.nlm.nih.gov/34080059/> Дата обращения: 03.11.2023
5. Центр диагностики и телемедицины, Базовые диагностические требования. [Электронный ресурс] // URL: <https://mosmed.ai/documents/278/Базовые_диагностические_требования_28_02_24.pdf> Дата обращения: 05.11.2023
6. Центр диагностики и телемедицины, MosMedData КТ с признаками позвоночника тип 3. [Электронный ресурс] // URL: [https://mosmed.ai/datasets/mosmeddata-kt-s-priznakami-osteoporoza-pozvonochnika-tip-iii-versiya-2](https://mosmed.ai/datasets/mosmeddata-kt-s-priznakami-osteoporoza-pozvonochnika-tip-iii-versiya-2/)/ Дата обращения: 15.11.2023
7. Центр диагностики и телемедицины, MosMedData КТ с признаками позвоночника тип 3. [Электронный ресурс] // URL: [https://mosmed.ai/datasets/mosmeddata-kt-s-priznakami-osteoporoza-pozvonochnika-tip-iii/](https://mosmed.ai/datasets/mosmeddata-kt-s-priznakami-osteoporoza-pozvonochnika-tip-iii/%20) Дата обращения: 16.11.2023
8. Центр диагностики и телемедицины, Набор данных КТ, ММГ, РГ/ФЛГ с целью селф-тестирования ИИ-сервисов для поиска признаков приоритетных патологий [Электронный ресурс] // URL: [https://mosmed.ai/datasets/aie20selftest/](https://mosmed.ai/datasets/aie20selftest/%20) Дата обращения: 20.11.2023
9. DICOM, About DICOM: Overview. [Электронный ресурс] // URL: [https://www.dicomstandard.org/about-home](https://www.dicomstandard.org/about-home%20) Дата обращения: 01.12.2023
10. Healtis, Теги DICOM. [Электронный ресурс] // URL: [https://healtis.ru/articles/dicom-tags/](https://healtis.ru/articles/dicom-tags/%20) Дата обращения: 15.12.2023
11. Pydicom, Pydicom. [Электронный ресурс] // URL: <https://pydicom.github.io/> Дата обращения: 20.12.2023
12. 3D Slicer, 3D Slicer documentation. [Электронный ресурс] // URL: [https://slicer.readthedocs.io/en/latest/](https://slicer.readthedocs.io/en/latest/%20) Дата обращения: 02.01.2024
13. ByteXD, Why are GPUs well-suited for machine learning?. [Электронный ресурс] // URL: <https://bytexd.com/hardware/why-are-gpus-well-suited-for-machine-learning/> Дата обращения: 20.01.2024
14. ResearchGate, Pathological Images Classification based on the Pretrained Convolutional Neural Network [Электронный ресурс] // URL: [https://www.researchgate.net/publication/369872580\_Pathological\_Images\_Classification\_based\_on\_the\_Pretrained\_Convolutional\_Neural\_Network](https://www.researchgate.net/publication/369872580_Pathological_Images_Classification_based_on_the_Pretrained_Convolutional_Neural_Network%20) Дата обращения: 10.02.2024
15. PyTorch, Models and pre-trained weights [Электронный ресурс] // URL: [https://pytorch.org/vision/stable/models.html](https://pytorch.org/vision/stable/models.html%20) Дата обращения: 14.02.2024
16. Object Management Group, OMG Unified Modeling Language [Электронный ресурс] // URL: [https://www.omg.org/spec/UML/2.5.1/PDF](https://www.omg.org/spec/UML/2.5.1/PDF%20) Дата обращения: 24.02.2024
17. Flask, Flask Documentation (2.3.x). [Электронный ресурс] // URL: [https://flask-docs.readthedocs.io/en/latest/quickstart/](https://flask-docs.readthedocs.io/en/latest/quickstart/%20) Дата обращения: 01.03.2024

ПРИЛОЖЕНИЕ А  
**Техническое задание**

Техническое задание к курсовой работе находится в файле, приложенном к файлам курсовой работы. Название файла: «Техническое Задание Разработка системы диагностики остеопороза по КТ.docx».

ПРИЛОЖЕНИЕ Б  
**Руководство пользователя**

|  |
| --- |
| Пермский филиал федерального государственного автономного образовательного учреждения высшего образования  «Национальный исследовательский университет  «Высшая школа экономики»  *Факультет социально-экономических и компьютерных наук* |
| Кирьянов Сергей Вячеславович  **Информационная система диагностики остеопороза по КТ снимкам позвоночника**  *Руководство пользователя*  студента образовательной программы «Программная инженерия» по направлению подготовки *09.03.04 Программная инженерия*  Руководитель  Преподаватель кафедры информационных технологий в бизнесе НИУ ВШЭ-Пермь    М.Д. Чистогов |

Пермь, 2024 год

1. **Введение**
2. **Область применения**

Приложение применяется при:

1. Предварительных комплексных испытаниях.
2. Приемочных испытаниях.
3. Свободная эксплуатация.

Основную сферу использования приложения можно охарактеризовать, как медицина и здравоохранение.

1. **Краткое описание возможностей**

Веб-приложение предназначено для поддержки принятия врачебных решений в ходе диагностики остеопороза позвоночника.

Приложение позволяет загружать серии КТ снимков позвоночника формата DICOM для обработки системой. После обработки загруженных данных системы выдает вероятность наличия остеопороза, а также, проекции серии КТ снимков.

Эксплуатация приложения возможна на одном языке: русском.

1. **Уровень подготовки пользователя**

Для успешной эксплуатации приложения пользователь должен ознакомиться с эксплуатационной документацией и не обязан обладать какими-либо специализированными знаниями и/или умениями, однако, информационная система предназначена для использования квалифицированными медицинскими работниками.

1. **Перечень эксплуатационной документации, с которой необходимо ознакомиться пользователю**

Пользователю необходимо ознакомиться с руководством пользователя.

1. **Назначение и условия применения**

Вею-приложение предназначено для поддержки принятия врачебных решений в ходе диагностики остеопороза по КТ снимками позвоночника.

Приложение позволяет загружать серии КТ снимков позвоночника формата DICOM для обработки системой. После обработки загруженных данных системы выдает вероятность наличия остеопороза, а также, проекции серии КТ снимков.

Для успешного функционирования веб-приложения необходимо постоянное, стабильное интернет-соединение.

1. **Подготовка к работе**
   1. **Состав и содержание дистрибутивного носителя данных**

Для использования функций импорта данных информационной системы требуется наличие любого предустановленного браузера из перечня: Google Chrome, Microsoft Edge, Opera, Яндекс Браузер.

* 1. **Порядок загрузки приложения**

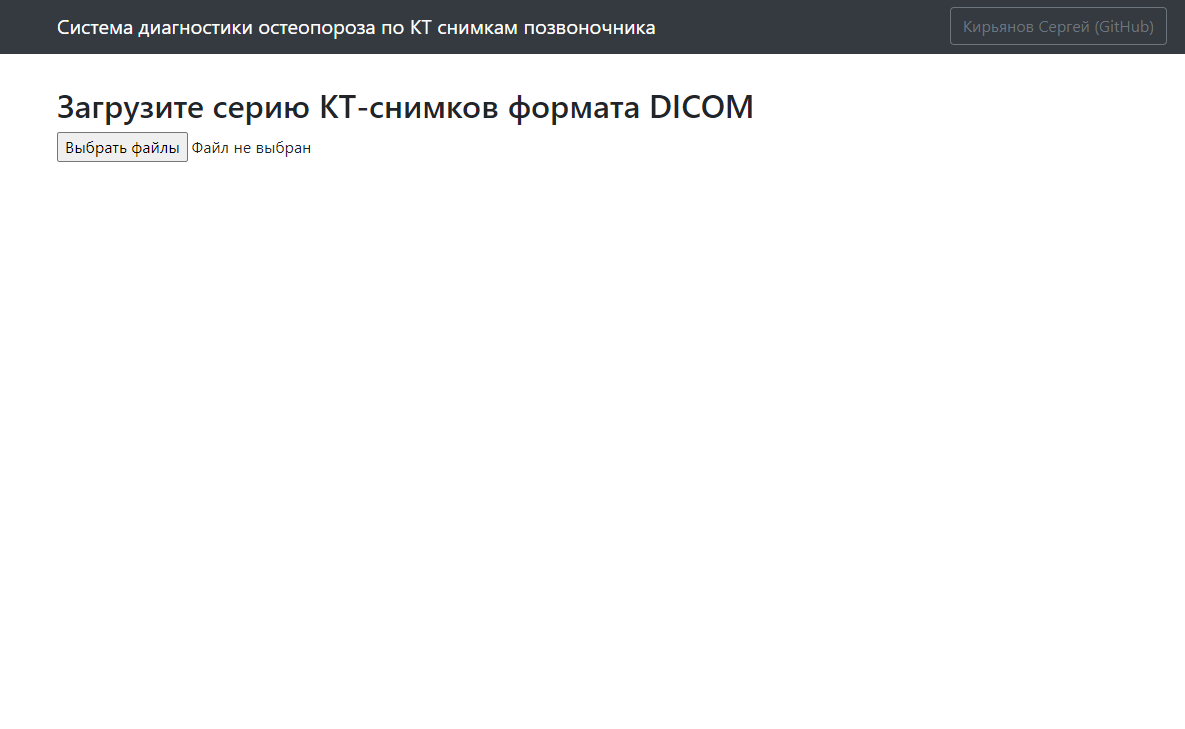
Перед началом работы с информационной системой диагностики остеопороза по КТ снимкам позвоночника на рабочем месте пользователя необходимо выполнить следующие действия:

1. Открыть в браузере страницу веб-приложения.
2. Проверить стабильность интернет-соединения.
   1. **Порядок проверки работоспособности**

Для проверки работоспособности информационной системы диагностики остеопороза по КТ снимкам позвоночника необходимо выполнить следующие действия:

1. Открыть страницу веб-приложения, перейдя по его адресу.
2. Дождаться загрузки домашней страницы веб-приложения.
3. Загрузить как минимум 1 файл формата DICOM.
4. Начать обработку загруженного файла, нажав на кнопку «Анализ».
5. Дождаться результата обработки загруженного файла.
6. **Описание окон приложения**
   1. **Основное окно приложения**

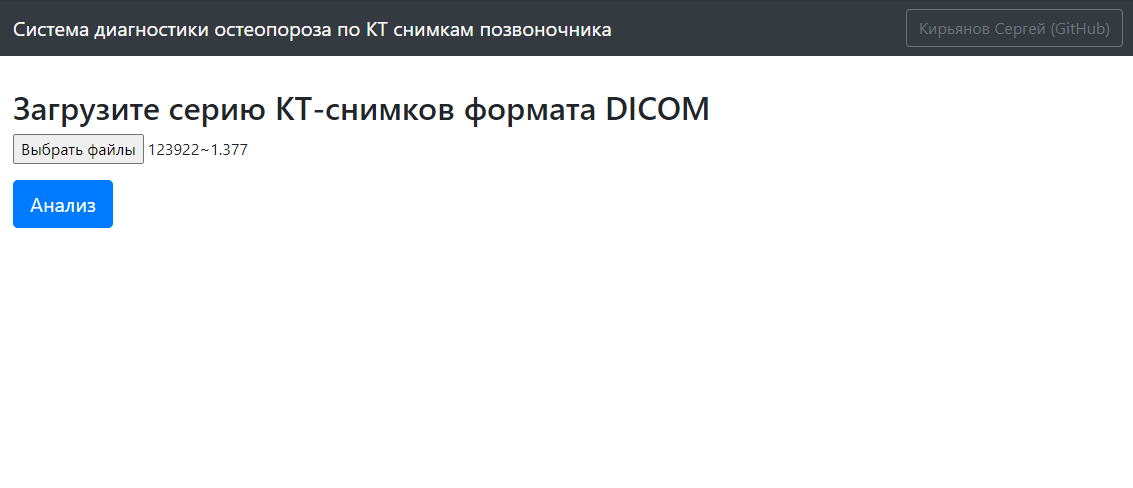
Основное окно приложения до выбора файлов изображено ниже, на рисунке 1.



***Рисунок 1 – Основное окно приложения до выбора файлов***

Кнопка «Выбрать файлы» используется для загрузки серии КТ снимков формата DICOM.

Затем появится кнопка «Анализ». При нажатии на кнопку начнется анализ загруженной серии КТ снимков информационной системой. Основное окно приложения после выбора файлов изображено ниже, на рисунке 2.



***Рисунок 2 – Основное окно приложения после выбора файлов***

После нажатия на кнопку «Анализ», произойдет обработка данных информационной системой, в результате чего будет выведена информация о вероятности наличия остеопороза (в процентах), а также, изображения различных проекций серии КТ снимков и изображение для сегментации снимков, в которых были найдены признаки остеопороза (красным выделены места с признаками наличия признаков остеопороза, а зеленым места без признаков наличия остеопороза). Основное окно после обработки данных изображено ниже, на рисунке 3.



***Рисунок 3 – Основное окно приложения после обработки данных***

1. **Аварийные ситуации**

В случае возникновения ошибок при использовании приложения, необходимо перезагрузить страницу браузера.

В случае, если изображения проекций серий КТ снимков не обновляются при загрузке новых серий КТ снимков, необходимо перезагрузить страницу браузера.

В случае, если страница веб-приложения не закрывается, необходимо закрыть браузер.

1. **Рекомендации по освоению**

При работе с приложением рекомендуется отключить дополнительные расширения браузера.

Особых рекомендаций по освоению и эксплуатации приложения нет.

ПРИЛОЖЕНИЕ В  
**Код модуля app.py**

from flask import Flask, render\_template  
from flask import request  
from matplotlib.pyplot import imread  
import os, shutil  
import predict  
import load\_data  
  
app = Flask(\_\_name\_\_)  
  
# Пути к файлам, которые будут обрабатываться и выводиться  
STATIC\_FOLDER = 'static'  
UPLOAD\_FOLDER = STATIC\_FOLDER + '/uploads'  
UPLOAD\_FOLDER\_DCM\_SERIE = UPLOAD\_FOLDER + '/buf\_dcm\_serie'  
  
  
@app.route('/about')  
def about() -> str:  
 return '''Курсовая работа, 3 курс, НИУ ВШЭ - ПЕРМЬ, Программная инженерия, 2024 год. \n   
 Информационная система диагностики остеопороза по КТ снимкам позвоночника. \n   
 Работу выполнил Кирьянов Сергей Вячеславович.'''  
  
@app.route('/', methods=['GET'])  
def index():  
 load\_data.clear\_dcm\_data(UPLOAD\_FOLDER+"/buf\_img")  
 load\_data.clear\_dcm\_data(UPLOAD\_FOLDER\_DCM\_SERIE)  
 return render\_template('index.html')  
  
@app.route('/predict', methods= ['GET', 'POST'])  
def upload\_data():  
 if request.method == "POST":  
 files = request.files.getlist("file[]")  
 for file in files:  
 fullname = os.path.join(UPLOAD\_FOLDER\_DCM\_SERIE, file.filename)  
 file.save(fullname)  
 try:  
 img\_paths = load\_data.get\_proections(load\_data.get\_dcm\_serie(UPLOAD\_FOLDER\_DCM\_SERIE), files[0].filename)  
 # Получаем прогноз от модели  
 pred = predict.get\_predict\_for\_serie\_path(UPLOAD\_FOLDER\_DCM\_SERIE, files[0].filename)  
 except Exception as err:  
 pred = 'Ошибка! Загрузите корректную серию КТ снимков формата DICOM', ''  
 img\_paths = ['','']  
 load\_data.clear\_dcm\_data(UPLOAD\_FOLDER\_DCM\_SERIE)  
 return [pred, img\_paths]  
 return None  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 app.run(debug = True)

ПРИЛОЖЕНИЕ Г  
**Код модуля «load\_data.py»**

import pydicom

import pydicom.data

from pydicom.pixel\_data\_handlers.util import apply\_voi\_lut, apply\_color\_lut, apply\_modality\_lut

import gdcm #Для работы с DICOM-файлами, у которых было использовано сжатие массива пикселей

import cv2

from PIL import Image

import numpy as np

import os, shutil

from glob import glob

from PIL import Image, ImageColor

IMAGE\_SIZE = (180, 180)

STATIC\_FOLDER = 'static'

UPLOAD\_FOLDER = STATIC\_FOLDER + '/uploads'

UPLOAD\_FOLDER\_DCM\_SERIE = UPLOAD\_FOLDER + '/buf\_dcm\_serie'

UPLOAD\_FOLDER\_IMGS = UPLOAD\_FOLDER + '/buf\_img/'

image\_format = '.jpg'

#Получение серии снимков из указанной директории

def get\_dcm\_serie(path\_to\_dcm\_serie : str):

ds\_list = []

dcm\_files = os.listdir(path\_to\_dcm\_serie)

for file in dcm\_files:

buf\_ds\_file = pydicom.dcmread(path\_to\_dcm\_serie+'/'+file)

#Проверяем, аксиальная ли проекция у снимка (иногда в датасетах появляются сагитальные и корональные снимки)

if ("ImageType" in buf\_ds\_file) == False:

ds\_list.append(buf\_ds\_file)

elif buf\_ds\_file.ImageType[-1] == 'AXIAL':

ds\_list.append(buf\_ds\_file)

ds\_list = sorted(ds\_list, key=lambda s: s.ImagePositionPatient[2])

return ds\_list

def get\_dcm\_file\_save\_img(path, filename):

dcm\_file = pydicom.dcmread(path)

px\_ar = np.array(dcm\_file.pixel\_array)

#image\_path = image\_path.replace(UPLOAD\_FOLDER, image\_format)

im = Image.fromarray(np.dstack([px\_ar, px\_ar, px\_ar]))

im = ImageColor.getrgb('red')

im.save(UPLOAD\_FOLDER\_IMGS + filename + image\_format)

#cv2.imwrite(UPLOAD\_FOLDER\_IMGS + filename + image\_format, dcm\_file.pixel\_array)

return UPLOAD\_FOLDER\_IMGS + filename + image\_format

# Функция для вывода нескольких проекций (аксиальной, сагитальной, корональной)

def get\_proections(ds\_list, filename):

#ds\_list = sorted(ds\_list, key=lambda s: s.ImagePositionPatient[2])

# Создаем трехмерный массив

img\_shape = list(ds\_list[0].pixel\_array.shape)

img\_shape.append(len(ds\_list))

img3d=np.zeros(img\_shape)

#print(ds\_list[0].pixel\_array.shape)

# Заполняем трехмерный массив нашими снимками (срезами)

for i, s in enumerate(ds\_list):

img2d = s.pixel\_array

img3d[:,:,i] = img2d

cv2.imwrite(UPLOAD\_FOLDER\_IMGS + filename +'axial' + image\_format, img3d[:,:,img\_shape[2]//2])

cv2.imwrite(UPLOAD\_FOLDER\_IMGS + filename + 'sagital' + image\_format, img3d[:,img\_shape[1]//2,:])

cv2.imwrite(UPLOAD\_FOLDER\_IMGS + filename + 'coronal' + image\_format, img3d[img\_shape[0]//2,:,:].T)

axial = UPLOAD\_FOLDER\_IMGS + filename + 'axial' + image\_format

sagital = UPLOAD\_FOLDER\_IMGS + filename + 'sagital' + image\_format

coronal = UPLOAD\_FOLDER\_IMGS + filename + 'coronal' + image\_format

return [axial, sagital, coronal]

#Функция, которая увличивает контрастность изображений (костная ткань становится видна лучше) и изменяет их размер

def resize\_and\_voi\_lut(dcm\_file):

buf\_pixel\_array = apply\_voi\_lut(dcm\_file.pixel\_array, dcm\_file)

resized\_array = cv2.resize(buf\_pixel\_array, IMAGE\_SIZE, interpolation = cv2.INTER\_NEAREST)

return resized\_array

def clear\_dcm\_data(path):

for filename in os.listdir(path):

file\_path = os.path.join(path, filename)

try:

if os.path.isfile(file\_path) or os.path.islink(file\_path):

os.unlink(file\_path)

elif os.path.isdir(file\_path):

shutil.rmtree(file\_path)

except Exception as e:

print('Не удалось удалить файлы %s. Причина: %s' % (file\_path, e))

ПРИЛОЖЕНИЕ Д  
**Код модуля «predict.py»**

from pydicom.pixel\_data\_handlers.util import apply\_voi\_lut, apply\_color\_lut, apply\_modality\_lut

import gdcm #Для работы с DICOM-файлами, у которых было использовано сжатие массива пикселей

from PIL import Image

import numpy as np

import load\_data

import tqdm

# PyTorch

import torch

from torch.utils.data import DataLoader

from sklearn.metrics import classification\_report, confusion\_matrix

from PIL import Image

model = torch.load(r"itog\_201\_epochs\_86ac.pt")

if torch.cuda.is\_available():

device = torch.device("cuda:0")

else:

device = torch.device("cpu")

BATCH\_SIZE = 16

IMAGE\_SIZE = (180, 180)

BASE = r"M:\all\_mosmed\_datasets"

STATIC\_FOLDER = 'static'

UPLOAD\_FOLDER = STATIC\_FOLDER + '/uploads'

UPLOAD\_FOLDER\_DCM\_SERIE = UPLOAD\_FOLDER + '/buf\_dcm\_serie'

UPLOAD\_FOLDER\_IMGS = UPLOAD\_FOLDER + '/buf\_img/'

class MyDataSet():

def \_\_init\_\_(self, x\_tensor, y\_tensor):

self.x = x\_tensor

self.y = y\_tensor

self.n\_samples = len(x\_tensor)

def \_\_getitem\_\_(self, index):

return self.x[index], self.y[index]

# Чтобы можно было использовать len(dataset) для получения размера

def \_\_len\_\_(self):

return self.n\_samples

def get\_predictions(dataloader):

y\_pred\_list = []

y\_true\_list = []

with torch.no\_grad():

for x\_batch, y\_batch in tqdm.tqdm(dataloader, leave=False):

x\_batch = x\_batch.unsqueeze(1)

x\_batch = torch.cat((x\_batch, x\_batch, x\_batch), 1)

x\_batch, y\_batch = x\_batch.to(device), y\_batch.to(device)

y\_test\_pred = model(x\_batch)

y\_test\_pred = torch.log\_softmax(y\_test\_pred, dim=1)

\_, y\_pred\_tag = torch.max(y\_test\_pred, dim = 1)

y\_pred\_list.append(y\_pred\_tag.cpu().numpy())

y\_true\_list.append(y\_batch.cpu().numpy())

return y\_true\_list, y\_pred\_list

def apply\_voi\_lut\_array(ds\_list):

ds\_list\_lut = [load\_data.resize\_and\_voi\_lut(x) for x in ds\_list]

return ds\_list\_lut

def get\_predict\_for\_lut\_serie(ds\_list\_lut, filename):

pathologies\_x\_test = np.array(ds\_list\_lut)

pathologies\_x\_test = (pathologies\_x\_test - pathologies\_x\_test.min()) / (pathologies\_x\_test.max() - pathologies\_x\_test.min())

tensor\_x\_test = torch.Tensor(pathologies\_x\_test)

buf\_y = np.arange(1, len(tensor\_x\_test)+1, 1)

buf\_y\_tensor = torch.Tensor(buf\_y)

test\_data = list(MyDataSet(tensor\_x\_test, buf\_y\_tensor))

testloader = DataLoader(test\_data, batch\_size=1)

\_, y\_pred\_list = get\_predictions(testloader)

# Создадим полосу, которая укажет области, на которых были выявлены признаки патологии

y\_pred\_list = np.concatenate(y\_pred\_list).ravel().reshape(len(y\_pred\_list),1).astype(np.uint8)

img = Image.new('RGB', (len(y\_pred\_list),20))

colors = [(0, 255, 0) if pixel == 0 else (255, 0, 0) for pixel in np.concatenate(y\_pred\_list).ravel().tolist()]

for i in range(len(y\_pred\_list)):

for j in range(20):

img.putpixel((i, j), colors[i])

img.save(UPLOAD\_FOLDER\_IMGS + filename + 'sag\_map' + '.jpg')

return f'Вероятность наличия остеопороза: {y\_pred\_list.mean()\*100}%', UPLOAD\_FOLDER\_IMGS + filename + 'sag\_map' + '.jpg'

def get\_predict\_for\_serie\_path(path: str, filename):

dcm\_serie = load\_data.get\_dcm\_serie(path)

dcm\_serie = apply\_voi\_lut\_array(dcm\_serie)

return get\_predict\_for\_lut\_serie(dcm\_serie, filename)

ПРИЛОЖЕНИЕ Е  
**Остальной код приложения**

Для удобства, остальной код приложения расположен в удаленном репозитории на веб-сервисе для хостинга IT-проектов «GitHub».

Ссылка на репозиторий: <https://github.com/Laoreet/Course-work-3-osteoporosis-detection>

Код документа, в котором производилось обучение и тестирование модели нейронной сети, используемой в системе, также можно посмотреть в сервисе «Google Colaboratory», перейдя по ссылке: <https://colab.research.google.com/drive/1YHiLJxYlBVU1kUSRbEE2RQtaa-7GL0Dr?usp=sharing>