**Документация по проекту**

В выпускной квалификационной работе рассматривается тема «Разработка приложения анализа объектов с применением математических методов и нейронных сетей».

Выпускная квалификационная работа содержит: 69 страниц, 20 рисунков, 3 таблицы, 7 приложений.

В первом разделе проведён анализ предметной области, рассмотрены современные технологии искусственного интеллекта и их применение в медицине, выполнен обзор существующих решений и средств разработки, обоснован выбор технологических решений.

Во втором разделе определены функциональные и нефункциональные требования к разрабатываемому приложению, описана архитектура и компоненты системы.

В третьем разделе изложены детали технической реализации — описан процесс построения и обучения сверточной нейронной сети для классификации рентгеновских снимков, приведены этапы тестирования программы на тестовых данных, а также проведена оценка эффективности модели и её применимости в реальных условиях.

В четвёртом разделе продемонстрирована работа приложения, приведены примеры анализа рентгеновских снимков и результаты работы системы.

Ключевые слова: Python, Tensorflow, CNN, Machine learning.

Методы проведения работы: анализ предметной области, машинное обучение, моделирование, проектирование архитектуры, реализация и тестирование программного обеспечения.

Область применения результатов: автоматизация диагностики заболеваний по медицинским изображениям, внедрение в системы поддержки принятия врачебных решений, развитие цифровых сервисов в здравоохранении, образовательные проекты в области искусственного интеллекта и программной инженерии.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 11](#_Toc199721109)

[Актуальность темы 11](#_Toc199721110)

[Цель и задачи исследования 12](#_Toc199721111)

[Объект и предмет исследования 14](#_Toc199721112)

[1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 15](#_Toc199721113)

[1.1 Технологии искусственного интеллекта в медицине 15](#_Toc199721114)

[1.2 Обзор аналогов 16](#_Toc199721115)

[1.3 Обзор средств разработки 18](#_Toc199721116)

[1.4 Обоснование выбора технологических решений 21](#_Toc199721117)

[2 АНАЛИЗ ТРЕБОВАНИЙ 27](#_Toc199721118)

[2.4 Функциональные требования 27](#_Toc199721119)

[2.5 Нефункциональные требования 28](#_Toc199721120)

[3 АРХИТЕКТУРА 30](#_Toc199721121)

[3.4 Архитектура программы 30](#_Toc199721122)

[3.5 Описание компонентов программы 32](#_Toc199721123)

[4 ДЕМОНСТРАЦИЯ РАБОТЫ ПРИЛОЖЕНИЯ 43](#_Toc199721124)

[4.1 Программная реализация разработанной программы 43](#_Toc199721125)

[4.2 Тестирование программы на тестовых снимках 53](#_Toc199721126)

[4.3 Оценка эффективности программы и ее применимость в реальных условиях 54](#_Toc199721127)

[ЗАКЛЮЧЕНИЕ 56](#_Toc199721128)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 59](#_Toc199721129)

[ПРИЛОЖЕНИЕ 61](#_Toc199721130)

**ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И ОБОЗНАЧЕНИЙ**

CNN — Convolutional Neural Network (сверточная нейронная сеть)

ML — Machine Learning (машинное обучение)

AI — Artificial Intelligence (искусственный интеллект)

GPU — Graphics Processing Unit (графический процессор)

CSV — Comma Separated Values (формат текстовых файлов с разделителями-запятыми)

IoU — Intersection over Union (метрика пересечения и объединения, используется в задачах компьютерного зрения)

ROC — Receiver Operating Characteristic (кривая рабочей характеристики приёмника)

AUC — Area Under Curve (площадь под ROC-кривой)

TP, TN, FP, FN — True Positive, True Negative, False Positive, False Negative (истинно положительный, истинно отрицательный, ложноположительный, ложноотрицательный результат)

**ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ**

Рентгеновский снимок  — медицинское изображение, полученное с помощью рентгеновского излучения, используемое для визуализации внутренних структур организма, в частности органов грудной клетки, с целью диагностики заболеваний дыхательной системы, таких как пневмония.

Пневмония — острое воспалительное заболевание лёгких, характеризующееся поражением альвеол и/или интерстициальной ткани, часто выявляемое по характерным изменениям на рентгеновских снимках.

Искусственный интеллект (ИИ) — свойство интеллектуальных систем выполнять задачи, требующие творческого подхода, анализа и принятия решений, которые традиционно считаются прерогативой человека.

Машинное обучение (МО) — раздел искусственного интеллекта, разрабатывающий методы и алгоритмы, позволяющие находить закономерности в данных и делать прогнозы или принимать решения без явного программирования на каждую задачу.

Глубокое обучение — область машинного обучения, использующая многослойные (глубокие) нейронные сети для автоматического извлечения признаков и решения сложных задач, таких как анализ изображений.

Сверточная нейронная сеть (CNN, Convolutional Neural Network) — разновидность искусственных нейронных сетей, предназначенная для обработки изображений и способная автоматически выделять иерархические признаки, что особенно эффективно при анализе медицинских снимков.

Классификация изображений — задача машинного обучения, заключающаяся в автоматическом определении принадлежности изображения к одному из заранее определённых классов (например, «норма» или «пневмония»).

Аугментация данных  — искусственное увеличение объёма обучающей выборки путём применения к исходным изображениям различных преобразований (повороты, отражения, масштабирование, изменение яркости и контраста, добавление шума и др.) для повышения устойчивости и обобщающей способности модели.

Обучение с учителем — тип машинного обучения, при котором алгоритм обучается на размеченных данных, то есть каждый пример содержит входные данные и соответствующую метку (правильный ответ).

Валидация — этап оценки качества обученной модели на отложенной (не использованной при обучении) выборке данных с целью контроля переобучения и подбора оптимальных параметров.

Тестовая выборка  — независимый набор данных, используемый для финальной проверки качества работы модели после завершения обучения и настройки параметров.

Метрика качества — количественная характеристика, используемая для оценки эффективности работы модели (например, точность, полнота, F1-мера, AUC-ROC).

**ВВЕДЕНИЕ**

## **Актуальность темы**

В современной медицинской практике одной из ключевых задач является своевременная и точная диагностика заболеваний органов дыхания, в частности пневмонии. Пневмония остаётся одной из наиболее распространённых причин заболеваемости и смертности во всём мире, особенно среди детей, пожилых людей и пациентов с ослабленным иммунитетом. На любом крупном медицинском учреждении ежедневно проводится множество рентгенологических исследований, и врачи сталкиваются с необходимостью оперативного анализа большого количества снимков для выявления признаков пневмонии.

Одной из проблем, которую необходимо решать, является обеспечение высокой точности и скорости интерпретации рентгеновских снимков, чтобы своевременно выявлять патологические изменения и назначать эффективное лечение. Человеческий фактор, высокая нагрузка на специалистов и ограниченное время на анализ каждого снимка могут приводить к диагностическим ошибкам и задержкам в постановке диагноза.

С помощью передовых методов искусственного интеллекта и глубокого обучения становится возможным автоматизировать процесс анализа рентгеновских изображений, что позволяет своевременно и достоверно получать информацию о наличии или отсутствии признаков пневмонии. Использование нейронных сетей для анализа медицинских изображений становится важным техническим инструментом повышения качества и стабильности диагностики, а также оптимизации работы медицинских учреждений.

Тема «Разработка программы для автоматического определения пневмонии по рентгеновским снимкам с использованием искусственного интеллекта» представляет собой важное направление исследований и разработок в области медицинской диагностики и цифрового здравоохранения. В настоящее время применение технологий искусственного интеллекта в медицине становится всё более популярным и актуальным средством повышения эффективности и доступности медицинских услуг.

Программное решение позволяет оперативно получать данные о состоянии пациента, а также проводить диагностику и назначать лечение более эффективно. Ключевой особенностью применения искусственного интеллекта является возможность автоматического анализа изображений, выявления малозаметных патологий и предоставления врачу дополнительной информации для принятия решений. Вот несколько причин, почему эта тема является актуальной:

1. Улучшение производительности и эффективности:  
   Программа позволяет оперативно выявлять признаки пневмонии на рентгеновских снимках, что помогает сократить время постановки диагноза, повысить точность выявления заболевания и своевременно начать лечение.
2. Сокращение времени ожидания и нагрузки на специалистов:  
   Благодаря автоматизации анализа изображений, врачи могут быстрее получать результаты диагностики, что уменьшает нагрузку на медицинский персонал и сокращает время ожидания для пациентов.
3. Улучшение безопасности и качества медицинской помощи:  
   Оперативное и точное выявление пневмонии позволяет избежать осложнений, снизить риск прогрессирования заболевания и повысить качество медицинской помощи.
4. Инновации**:**  
   Внедрение технологий искусственного интеллекта в медицинскую диагностику позволяет медицинским учреждениям идти в ногу с современными инновациями, улучшать свои рабочие процессы и повышать конкурентоспособность на рынке медицинских услуг.

## **Цель и задачи исследования**

Своевременная и точная диагностика пневмонии играет ключевую роль в обеспечении эффективного лечения и снижении смертности среди пациентов. Мониторинг состояния здоровья на основе рентгенологических исследований позволяет оперативно выявлять признаки заболевания, предотвращать развитие осложнений и повышать качество медицинской помощи.  
 Разработать программу, основанную на технологиях искусственного интеллекта и глубокого обучения, для автоматического определения пневмонии по рентгеновским снимкам грудной клетки с целью повышения эффективности, точности и доступности диагностики в медицинских учреждениях.

**Для достижения поставленной цели необходимо выполнить следующие задачи исследования:**

1. Изучение современных методов и технологий искусственного интеллекта, применяемых для анализа медицинских изображений, в частности рентгенограмм грудной клетки.
2. Анализ потребностей и требований медицинских специалистов к программным средствам автоматической диагностики пневмонии.
3. Сбор и подготовка выборки рентгеновских снимков с верифицированными диагнозами для обучения и тестирования модели.
4. Разработка методики предобработки и аугментации медицинских изображений для повышения устойчивости и точности модели.
5. Проектирование, обучение и оптимизация сверточной нейронной сети для автоматической классификации снимков на классы «пневмония» и «норма».
6. Создание прототипа программного решения с удобным пользовательским интерфейсом для загрузки снимков и получения результатов анализа.
7. Проведение тестирования прототипа на реальных данных, оценка точности, полноты и других метрик работы модели.
8. Оценка эффективности разработанной программы на основе результатов тестирования и отзывов медицинских специалистов.[8]

## **Объект и предмет исследования**

**Объектом исследования**  в данной работе являются медицинские изображения рентгеновских снимков грудной клетки пациентов, используемые в клинической практике для диагностики заболеваний органов дыхания, в частности-пневмонии.

**Предмет исследования** включает в себя следующие аспекты:

1. Разработка программного обеспечения:

Исследование методов и подходов к созданию программы, способной автоматически анализировать рентгеновские снимки грудной клетки с использованием технологий искусственного интеллекта и глубокого обучения для определения признаков пневмонии.

1. Методы сбора и подготовки данных**:**

Изучение способов сбора, разметки и предобработки медицинских изображений (включая аугментацию данных, нормализацию, стандартизацию), необходимых для обучения и тестирования моделей искусственного интеллекта.

1. Анализ потребностей пользователей**:**

Проведение исследований и опросов для выявления потребностей и требований медицинских специалистов (врачей-рентгенологов, пульмонологов, диагностов) к программным средствам автоматического анализа рентгеновских снимков.

1. Тестирование и оценка эффективности**:**

Разработка прототипа программы, проведение тестирования на реальных медицинских данных с целью оценки точности, полноты, чувствительности и специфичности работы системы, а также возможное внесение корректировок и доработок на основе полученных результатов и отзывов специалистов.

1. **АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ**

## **Технологии искусственного интеллекта в медицине**

Технологии искусственного интеллекта (ИИ) и машинного обучения (МО) в последние годы становятся неотъемлемой частью современной медицины, открывая новые возможности для повышения качества и доступности медицинской помощи. Одним из наиболее перспективных направлений является автоматизация анализа медицинских изображений, в частности-рентгеновских снимков грудной клетки, для выявления таких заболеваний, как пневмония.

Применение ИИ в медицинской диагностике включает следующие основные аспекты:

1. Визуализация и анализ данных:

Искусственный интеллект позволяет автоматически анализировать рентгеновские снимки, выявлять на них патологические изменения, такие как инфильтраты, затемнения и другие признаки пневмонии, что способствует более быстрой и точной постановке диагноза.

1. Поддержка принятия решений:

Системы на основе нейронных сетей могут предоставлять врачу дополнительную информацию-вероятность наличия патологии, степень уверенности модели, визуализацию областей интереса (heatmaps), что помогает специалисту принимать более обоснованные решения.

1. Обучение и повышение квалификации специалистов:  
    Автоматизированные системы анализа изображений могут использоваться для обучения молодых врачей, предоставляя им доступ к обширным базам клинических случаев и примерам сложных диагностических ситуаций.
2. Мониторинг состояния пациентов в динамике:  
    Использование ИИ позволяет отслеживать изменения на рентгеновских снимках во времени, что важно для контроля эффективности лечения и своевременного выявления осложнений.
3. Дистанционная диагностика и телемедицина:

Благодаря автоматизации анализа изображений, возможно предоставление медицинских услуг в удалённых регионах, где доступ к квалифицированным специалистам ограничен.

В статье «Artificial Intelligence in Medical Imaging: Opportunities, Applications, and Risks» отмечается, что за последнее десятилетие количество исследований и внедрённых решений в области ИИ для медицинской визуализации значительно возросло. Особенно активно развиваются системы для диагностики заболеваний лёгких, включая пневмонию, туберкулёз и COVID-19. Несмотря на то, что многие решения пока находятся на стадии прототипов или пилотных внедрений, потенциал ИИ в медицине оценивается как чрезвычайно высокий, а в ближайшие годы ожидается массовое распространение подобных технологий в клинической практике.

* 1. **Обзор аналогов**

В последние годы на рынке появилось множество программных решений и сервисов, предназначенных для автоматического анализа рентгеновских снимков грудной клетки с целью выявления пневмонии. Рассмотрим наиболее известные и значимые из них:

CheXNet — одна из первых и наиболее известных глубоких сверточных нейронных сетей, разработанных для диагностики пневмонии по рентгенограммам грудной клетки. Модель обучалась на большом датасете ChestX-ray14 и способна выявлять признаки пневмонии с точностью, сопоставимой с результатами опытных рентгенологов. CheXNet предоставляет врачу не только итоговую вероятность наличия патологии, но и визуализацию областей интереса (heatmaps), что облегчает интерпретацию результатов.

Qure.ai — коммерческое решение, интегрируемое в рабочие процессы клиник и диагностических центров. Система qXR автоматически анализирует рентгеновские снимки грудной клетки, выявляет пневмонию и другие патологии, формирует отчеты и визуализирует зоны подозрительных изменений. Продукт активно используется в ряде стран и одобрен для клинического применения.

COVID-Net — открытая нейросетевая архитектура, разработанная для диагностики COVID-19 и пневмонии на рентгенограммах. Модель доступна в открытом доступе, что позволяет исследователям и медицинским учреждениям использовать и дорабатывать её под свои задачи. COVID-Net демонстрирует высокую точность и активно используется в научных исследованиях.

**Общие недостатки существующих решений:**

Высокие требования к качеству и объёму данных:

Для корректной работы моделей требуется большое количество размеченных рентгеновских снимков высокого качества, что не всегда доступно в отдельных медицинских учреждениях.

**Стоимость внедрения и лицензирования:**

Большинство коммерческих решений требуют существенных затрат на приобретение лицензий, интеграцию и техническую поддержку.

**Ограниченная локализация и адаптация:**

Некоторые системы не поддерживают русский язык или не адаптированы под особенности отечественной медицинской документации и протоколов.

**Вопросы конфиденциальности и хранения данных:**

Использование облачных сервисов может вызывать опасения у медицинских учреждений по поводу безопасности персональных данных пациентов.

**Необходимость интеграции с существующими системами:**

Для полноценной работы требуется интеграция с медицинскими информационными системами (МИС), что может быть затруднено из-за различий в стандартах и протоколах обмена данными.

**Преимущества собственной разработки:**

Разработка собственной программы для автоматического определения пневмонии по рентгеновским снимкам позволяет преодолеть ряд ограничений существующих решений:

1. Гибкость и адаптация под нужды конкретного учреждения:

Возможность доработки функционала, локализации интерфейса и интеграции с внутренними МИС.

Контроль над данными и безопасностью:

Все данные хранятся и обрабатываются внутри организации, что соответствует требованиям законодательства по защите персональных данных.

Экономическая эффективность:

Отсутствие затрат на лицензирование и возможность постепенного масштабирования решения.

Развитие и обучение персонала:

Внедрение собственной системы способствует развитию компетенций сотрудников в области цифровой медицины и искусственного интеллекта.

* 1. **Обзор средств разработки**
     1. **Язык программирования Python.**

Python — это высокоуровневый, интерпретируемый язык программирования, который широко используется для анализа данных, машинного обучения, научных расчетов и создания прототипов программных решений. Python отличается лаконичным синтаксисом, богатой экосистемой библиотек и поддержкой со стороны сообщества, что делает его одним из наиболее предпочтительных языков для задач искусственного интеллекта и медицинской информатики.

1.3.2 Библиотеки и фреймворки машинного обучения

TensorFlow — это мощная платформа с открытым исходным кодом для машинного обучения, разработанная компанией Google. Она поддерживает создание, обучение и развертывание нейронных сетей различной сложности, включая глубокие сверточные сети, которые применяются для анализа медицинских изображений. TensorFlow обеспечивает высокую производительность вычислений и может работать как на CPU, так и на GPU.

Keras — это высокоуровневый API для построения и обучения нейронных сетей, интегрированный в TensorFlow. Keras предоставляет удобные инструменты для быстрого прототипирования, тестирования и визуализации моделей глубокого обучения, что значительно ускоряет процесс разработки.

NumPy — библиотека для эффективных численных вычислений и работы с многомерными массивами.

Pandas — инструмент для обработки и анализа табличных данных. Эти библиотеки используются для подготовки, анализа и предобработки медицинских данных, а также для работы с результатами экспериментов.

OpenCV (Open Source Computer Vision Library) и Pillow (PIL) — библиотеки для обработки и преобразования изображений. Они применяются для чтения, масштабирования, аугментации и преобразования рентгеновских снимков перед их подачей в нейронную сеть.

1.3.3 Средства визуализации и пользовательский интерфейс

Streamlit — современный фреймворк для создания интерактивных веб- приложений на Python. Он позволяет быстро разрабатывать пользовательские интерфейсы для анализа данных и визуализации результатов работы моделей машинного обучения. В данной работе Streamlit используется для создания веб -приложения, где пользователь может загружать рентгеновские снимки, запускать анализ и получать результаты диагностики в удобном формате.[4]

Matplotlib и Seaborn — популярные библиотеки для построения графиков и визуализации данных. Они используются для отображения результатов обучения моделей (графики точности, функции потерь, ROC-кривые), а также для визуализации распределения данных и метрик качества.

1.3.4 Интегрированные среды разработки (IDE):

PyCharm — это интегрированная среда разработки (IDE) от компании JetBrains, специально предназначенная для разработки на языке Python. PyCharm предоставляет широкий набор инструментов для эффективной работы с проектами в области анализа данных, машинного обучения, веб-разработки и автоматизации.

1.3.5 Система управления версиям

Git — это распределенная система контроля версий, которая позволяет отслеживать изменения в исходном коде программного обеспечения. Git сохраняет все изменения в виде "снимков" (commits) и позволяет вам работать с различными версиями кода, создавать ветки для самостоятельной разработки новых функций, а затем объединять их обратно в основную ветку проекта.

**1.3.6 Виртуальное окружение conda и утилиты**

Для эффективного обучения и инференса сверточных нейронных сетей в рамках данного проекта применялись специализированные программные библиотеки и инструменты, обеспечивающие аппаратное ускорение вычислений на графических процессорах (GPU).

cudatoolkit — это набор инструментов и библиотек для работы с GPU от компании NVIDIA, предоставляющий программный интерфейс CUDA для выполнения параллельных вычислений.

cuDNN (CUDA Deep Neural Network library) — специализированная библиотека NVIDIA, оптимизированная для выполнения операций глубокого обучения (таких как сверточные, пуллинговые и нормализационные слои) на GPU. Использование cuDNN позволяет значительно ускорить процесс обучения и оценки нейронных сетей, особенно при работе с большими изображениями и сложными архитектурами.

В рамках проекта установка cudatoolkit и cuDNN осуществлялась через систему управления пакетами Anaconda, что обеспечило совместимость версий и стабильную работу всех компонентов.

Виртуальное окружение conda

Для изоляции программной среды и управления зависимостями использовалась система виртуальных окружений conda.

Создание отдельного окружения conda позволило:

1. установить все необходимые библиотеки (TensorFlow, Keras, numpy, pandas и др.) в строго определённых версиях,
2. избежать конфликтов между пакетами,
3. обеспечить воспроизводимость результатов и простоту переноса проекта на другие рабочие станции.
   1. **Обоснование выбора технологических решений**

1.4.1 Обоснование выбора программирования Python

В ходе разработки программного решения для автоматического определения пневмонии по рентгеновским снимкам стоял выбор между несколькими языками программирования, такими как Python, C++, Java и R.  
Python был выбран в качестве основного языка разработки по следующим причинам:

* Широкая поддержка в области искусственного интеллекта и анализа данных.

Python является де-факто стандартом в научном сообществе и индустрии для задач машинного обучения, анализа медицинских изображений и прототипирования.

* Богатая экосистема библиотек.

Существуют мощные и хорошо документированные библиотеки для машинного обучения (TensorFlow, Keras, PyTorch), анализа данных (NumPy, Pandas), визуализации (Matplotlib, Seaborn) и работы с изображениями (OpenCV, PIL).

* Простота синтаксиса и низкий порог вхождения.

Python отличается лаконичным и понятным синтаксисом, что ускоряет процесс разработки и облегчает поддержку кода.

* Большое сообщество и наличие обучающих материалов.

Благодаря популярности Python легко найти ответы на возникающие вопросы, примеры кода и готовые решения.

* Кроссплатформенность.

Python работает на всех популярных операционных системах, что удобно для развертывания и тестирования приложений.

В отличие от C++ и Java, Python позволяет быстро разрабатывать и тестировать прототипы, что особенно важно при работе с экспериментальными моделями и обработкой медицинских изображений. Язык R также широко используется в анализе данных, однако его инструменты для построения и внедрения нейронных сетей менее развиты по сравнению с Python.

* + 1. Обоснование выбора фреймворков TensorFlow и Keras

Для построения и обучения моделей глубокого обучения был рассмотрен ряд фреймворков: TensorFlow, Keras, PyTorch, Scikit-learn.  
В качестве основного инструмента были выбраны TensorFlow и Keras по следующим причинам:

* Интеграция и простота использования:

Keras предоставляет интуитивно понятный высокоуровневый API для быстрого построения, обучения и тестирования нейронных сетей, а TensorFlow обеспечивает производительность и гибкость на низком уровне.

* Широкая поддержка и документация:

TensorFlow и Keras имеют обширную документацию, множество обучающих материалов и активное сообщество, что облегчает решение возникающих проблем.

* Масштабируемость и производительность:

TensorFlow позволяет эффективно использовать вычисления на CPU и GPU, что важно при работе с большими датасетами медицинских изображений.

* Совместимость с другими инструментами:

TensorFlow и Keras легко интегрируются с такими инструментами, как NumPy, Pandas, OpenCV, а также поддерживают экспорт и развертывание моделей в различных средах.

* Возможности для визуализации и отладки:

TensorBoard и другие инструменты позволяют отслеживать процесс обучения, анализировать метрики и визуализировать архитектуру моделей.[6]

PyTorch также является популярным фреймворком, однако в рамках данной работы был выбран TensorFlow/Keras из-за их большей распространённости в медицинских проектах и наличия готовых решений для задач анализа рентгеновских снимков.

1.4.3 Обоснование выбора Streamlit для пользовательского интерфейса

Для создания пользовательского интерфейса рассматривались различные варианты: Streamlit, Flask, Django, Dash.

Streamlit был выбран по следующим причинам:

* Быстрота разработки**:** Streamlit позволяет создавать интерактивные веб- приложения для анализа данных и визуализации результатов без необходимости писать сложный фронтенд-код.
* Интуитивный API**:** Простота синтаксиса и высокая скорость разработки позволяют быстро реализовать прототип и вносить изменения по ходу работы.
* Интеграция с Python**:** Streamlit полностью написан на Python и легко интегрируется с библиотеками машинного обучения и анализа данных.
* Возможности для визуализации**:** Встроенные компоненты для отображения изображений, графиков, таблиц и результатов анализа.
* Активное сообщество и поддержка**:** Регулярные обновления, наличие обучающих материалов и расширяемость за счёт сторонних компонентов.

Flask и Django предназначены для более сложных и масштабируемых веб- приложений, однако для задач визуализации и взаимодействия с моделью машинного обучения Streamlit оказался наиболее удобным и эффективным решением.

1.4.5 Обоснование выбора CNN модели

Использование сверточных нейронных сетей (CNN) для задачи определения пневмонии по рентгеновским снимкам обусловлено их уникальными преимуществами при работе с изображениями. Вот основные причины выбора именно CNN, а не других архитектур:

1. Автоматическое извлечение признаков

CNN способны самостоятельно выявлять важные визуальные паттерны и признаки на изображениях без необходимости ручного выделения признаков. Это особенно важно для рентгеновских снимков, где патологии могут иметь сложные и вариабельные проявления.

1. Учет пространственной структуры данных

В отличие от классических полносвязных сетей, CNN учитывают локальные зависимости и пространственную структуру изображений благодаря сверточным фильтрам и операциям пулинга. Это позволяет эффективно распознавать характерные для пневмонии изменения в тканях легких.

1. Высокая точность и доказанная эффективность в медицине

Современные исследования показывают, что CNN достигают высокой точности (до 95-98%) в задачах классификации медицинских изображений, включая рентгеновские снимки легких для выявления пневмонии и других заболеваний. Архитектуры CNN, такие как VGG, ResNet, Inception и DenseNet, успешно применяются для диагности ки и помогают автоматизировать процесс, снижая нагрузку на врачей.

1. Устойчивость к шуму и вариациям изображений

CNN хорошо справляются с вариациями качества снимков, различиями в освещении и положениях пациента, что характерно для медицинских данных, благодаря своей глубокой иерархической структуре и регуляризации.

1. Возможность масштабирования и дообучения

CNN легко масштабируются и могут использоваться с методами transfer learning, что позволяет применять предварительно обученные модели и адаптировать их под конкретные задачи с меньшими затратами ресурсов и времени.

Tаблица 1 — Cравнения архитектур нейронных сетей по четырём ключевым характеристикам.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Архитектура** | **Качество извлечения признаков** | **Глубина и масштабируемость** | **Скорость обучения** | **Требования к ресурсам** |
| **CNN** | + Отлично выделяют пространственные признаки - Могут терять глобальный контекст | + Легко наращивать глубину - Глубокие сети склонны к затуханию градиента | + Быстро обучаются на GPU - Требуют много данных | +Эффективны по памяти - Требуют GPU для больших задач |
| **ResNet** | + Сохраняют признаки на всех уровнях - Сложнее для интерпретации | + Очень глубокие (до 1000+ слоев) + Решают проблему затухания градиента | + Быстрее сходятся - Архитектура сложнее | - Требуют больше памяти из-за skip-соединений |
| **VGG** | + Простая и понятная архитектура + Хорошо извлекает признаки | - Очень большие по числу параметров - Неэффективны по памяти | - Медленное обучение из-за большого числа параметров | - Требуют много памяти и вычислений |
| **DenseNet** | + Эффективная передача признаков между слоями + Высокая точность | + Можно строить очень глубокие сети - Сложная архитектура | + Эффективное использование параметров - Медленнее, чем ResNet | - Требуют много памяти из-за плотных связей |
| **MLP** | - Плохо выделяют пространственные признаки + Универсальность | - Плохо масштабируются по глубине | + Простое обучение на малых задачах - Медленно на больших данных | + Меньше ресурсов на малых задачах - Неэффективны на изображениях |
| **RNN** | - Плохо работают с изображениями + Хорошо с последовательностями | - Трудно обучать глубокие RNN - Проблемы с градиентом | - Медленно обучаются + Хорошо для коротких последовательностей | - Требуют меньше памяти + Эффективны для текстов |
| **Transformer** | + Отлично выделяют глобальные взаимосвязи - Требуют много данных | + Легко масштабируются + Можно строить очень глубокие сети | - Медленное обучение + Высокая точность при большом объёме данных | - Очень высокие требования к ресурсам |

Таким образом, сверточные нейронные сети являются оптимальным выбором для анализа рентгеновских снимков при диагностике пневмонии, обеспечивая высокую точность, автоматизацию и надежность результатов, что подтверждается многочисленными научными исследованиями и практическими применениями в медицине.

1. **АНАЛИЗ ТРЕБОВАНИЙ**
   1. **Функциональные требования**

В данном разделе приведены функциональные требования к системе отслеживания технического состояния оборудования. Требования приведены в таблице 1, где указаны функциональные требования и их описания.

Tаблица 2 — Функциональные требования

|  |  |
| --- | --- |
| Загрузка изображений | Программа должна обеспечивать возможность загрузки рентгеновских снимков грудной клетки в различных форматах (например, JPG, PNG, DICOM) для последующего анализа. |
| Предобработка изображений | Система должна автоматически выполнять предобработку загруженных снимков (масштабирование, нормализация, фильтрация шумов и др.) для повышения качества анализа. |
| Автоматический анализ | Программа должна проводить автоматический анализ рентгеновских снимков с использованием обученной модели искусственного интеллекта для выявления признаков пневмонии. |
| Визуализация результатов | Система должна отображать результаты анализа, включая вероятность наличия пневмонии, а также визуализацию областей интереса (heatmaps) на снимке. |
| Формирование отчётов | Программа должна формировать отчёты по результатам анализа, содержащие основные выводы, вероятности, изображения и рекомендации для специалистов. |
| Интуитивный интерфейс | Программа должна иметь удобный и интуитивно понятный пользовательский интерфейс, позволяющий легко загружать снимки, просматривать результаты и формировать отчёты. |
| Хранение и безопасность данных | Система должна обеспечивать безопасное хранение медицинских изображений и результатов анализа, предотвращая несанкционированный доступ и утечку данных. |
| Интеграция с медицинскими системами | Программа должна иметь возможность интеграции с медицинскими информационными системами (МИС) для автоматического обмена данными и упрощения работы специалистов. |
| Обучения и поддержка | Программа должна предоставлять возможности для обучения персонала в работе с ней, а также техническую поддержку для решения возможных проблем и вопросов. |
| Масштабируемость и гибкость | Программа должна быть готова к масштабированию для работы с большим количеством данных и гибко настраиваться под различные требования медицинских учреждений |

* 1. **Нефункциональные требования**

В данной таблице (таблица 2) представлены нефункциональные требования к системе отслеживания технического состояния оборудования. Каждое требование сопровождается описанием.

Tаблица 3 — Нефункциональные требования

|  |  |
| --- | --- |
| Производительность | Программа должна обеспечивать высокую скорость обработки и анализа рентгеновских снимков, чтобы пользователь мог получать результаты диагностики в течение минимального времени. |
| Надежность | Система должна работать стабильно и корректно при длительном использовании, обеспечивая точность результатов и минимизируя риск сбоев или некорректной работы. |
| Доступность | Программа должна быть доступна медицинским специалистам в любое время, с возможностью работы как на локальных компьютерах, так и через веб-интерфейс. |
| Скорость отклика | Система должна обеспечивать быстрый отклик на действия пользователя: загрузку изображений, запуск анализа, отображение результатов и формирование отчётов. |
| Совместимость | Программа должна быть совместима с различными форматами медицинских изображений (JPG, PNG, DICOM) и поддерживать интеграцию с медицинскими информационными системами. |
| Безопасность | Система должна обеспечивать защиту персональных данных пациентов и результатов диагностики от несанкционированного доступа, утечек и потери информации. |
| Эргономичность | Программа должна иметь удобный, интуитивно понятный и современный пользовательский интерфейс, адаптированный для специалистов с разным уровнем подготовки. |
| Требования к аппаратуре | Система должна быть оптимизирована для работы на современных компьютерах и, при необходимости, мобильных устройствах, эффективно использовать вычислительные ресурсы |
| Сопровождение и обновления | Программа должна предусматривать регулярные обновления, исправление ошибок и техническую поддержку для обеспечения её стабильной работы и актуальности. |
| Соблюдение законодательства | Система должна соответствовать требованиям законодательства о защите персональных данных и медицинской информации, а также стандартам безопасности и конфиденциальности. |

Было принято решение сделать уклон на точность и модифицируемость модели, а не на графический интерфейс, так как модель - ключевое в данной работе. Веб-приложение на Streamlit хорошо справляется с потоком данных из снимков-рентгенов в больших объемах. Подразумевается, что функционал на пользовательском интерфейсе намного меньше, чем функционал для обучения самой модели. Главная цель - выявить пневмонию по снимку рентгена легких, пользователю достаточно кнопок "Загрузить изображения", "Анализировать" и прочие интуитивные функции как "Скачать отчет", "Фильтрация".

1. **АРХИТЕКТУРА**
   1. **Архитектура программы**

Конечная архитектура приложения представлена на рис. 1.

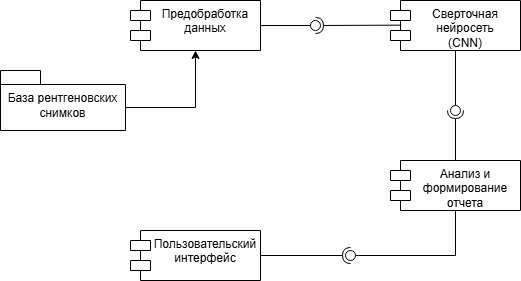


Рис. 1 Диаграмма компонентов системы.

Она состоит из следующих ключевых компонентов:

* База рентгеновских снимков: Хранит изображения грудной клетки, используемые как для обучения, так и для анализа новых снимков.
* Модуль предобработки данных: Отвечает за подготовку изображений к анализу (приведение к нужному размеру, нормализация, преобразование в оттенки серого, аугментация и т.д.).
* Сверточная нейронная сеть (CNN): Основной аналитический компонент, осуществляющий классификацию снимков на две категории: "Норма" и "Пневмония".
* Модуль анализа и формирования отчёта: Получает результаты работы нейросети, формирует итоговые выводы, рассчитывает уверенность, вероятности, создает отчеты.
* Пользовательский интерфейс (Web-приложение): Позволяет пользователю загружать снимки, запускать анализ, просматривать результаты и скачивать отчёты.

Описание работы компонентов:

1. Пользователь загружает один или несколько рентгеновских снимков через веб-интерфейс.
2. Каждый снимок поступает в модуль предобработки данных, где проходит необходимые преобразования для корректной работы модели.
3. После предобработки снимок отправляется в сверточную нейронную сеть, которая анализирует изображение и определяет вероятность наличия пневмонии.
4. Модуль анализа и формирования отчёта собирает результаты по каждому снимку, формирует сводную информацию и позволяет пользователю скачать итоговый отчет в формате CSV.
5. Вся информация отображается в удобном виде в пользовательском интерфейсе, где пользователь может фильтровать результаты, просматривать статистику и принимать решения.

Система не использует реальные медицинские базы данных и не подключена к медицинскому оборудованию-снимки загружаются вручную пользователем.

Облачное хранилище данных не применяется, все вычисления и хранение данных происходят локально в рамках приложения.

Модуль предобработки и анализ с помощью нейросети реализованы непосредственно внутри приложения для демонстрации работы.

В проект встроена обученная модель сверточной нейронной сети, а также модули для загрузки и предобработки изображений . При запуске приложения пользователь может загрузить снимки, которые проходят через модуль предобработки и анализируются моделью. Результаты анализа отображаются в пользовательском интерфейсе, где доступна фильтрация по классам, просмотр статистики и скачивание отчёта.

* 1. **Описание компонентов программы**

**3.2.1 База рентгеновских снимков (chest\_xray)**

Как было описано ранее, исходные медицинские изображения организованы в структуру папок, разделённых на обучающую, валидационную и тестовую выборки. Внутри каждой выборки имеются две категории: «NORMAL» (снимки без патологии) и «PNEUMONIA» (снимки с признаками пневмонии). Такая организация обеспечивает удобство загрузки данных и их корректную разметку для последующего машинного обучения (рис.3).

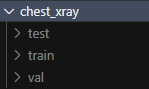
****

Рис. 2. Структура базы данных рентгеновских снимков

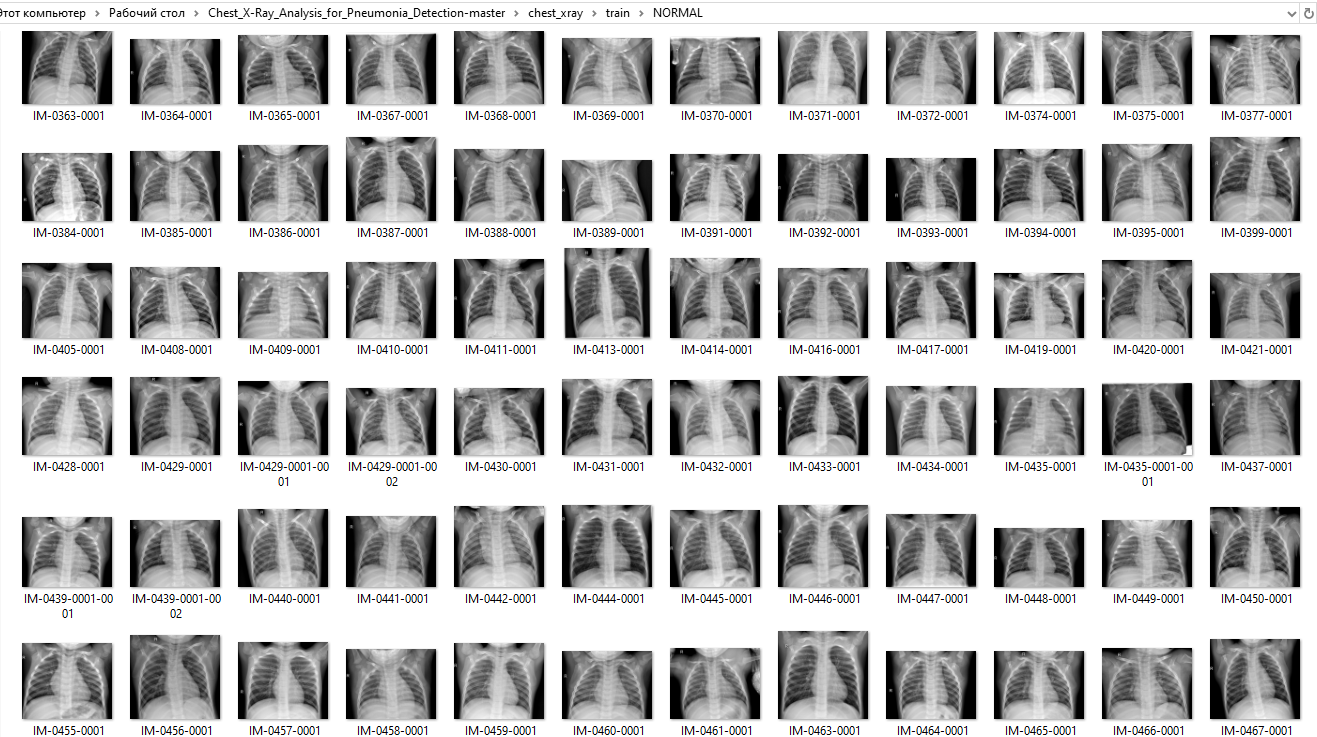


Рис. 3. Пример содержимого папки train

База содержит 5863 снимков, соотношение представлено на рис.(4)

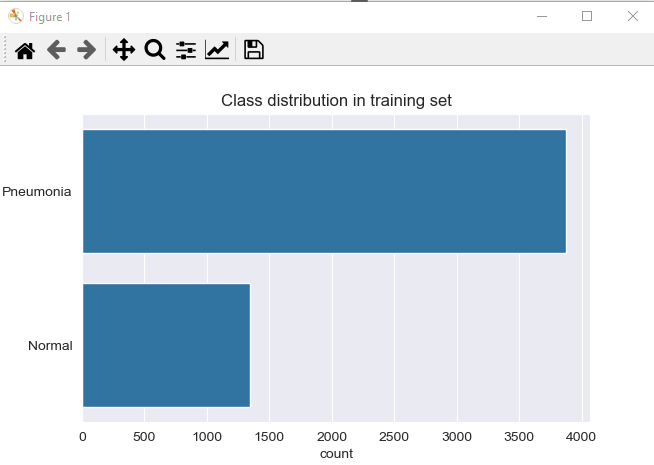


Рис. 4. Соотношение количества снимков Пневмония/Норма

**3.2.2 Модуль загрузки и предобработки данных (data\_loader.py)**

Этот компонент отвечает за автоматизированную обработку исходных изображений. Функции модуля включают:

1. Единообразие входных данных

Модуль приводит все изображения к одному размеру (150×150 пикселей) и формату (оттенки серого). Это обеспечивает корректную работу сверточной нейронной сети и позволяет ей быстрее и стабильнее обучаться, поскольку сеть не тратит ресурсы на адаптацию к разным размерам и цветовым схемам.

2. Нормализация

Приведение значений пикселей к диапазону (делением на 255) помогает избежать проблем с масштабами данных, ускоряет сходимость и предотвращает числовую нестабильность при обучении. Это способствует снижению значения функции потерь (Loss) на графике.

3. Аугментация данных

Аугментация (повороты, сдвиги, масштабирование, отражения) увеличивает разнообразие обучающей выборки. Благодаря этому модель становится более устойчивой к небольшим изменениям на входных изображениях, лучше обобщает и реже переобучается. Это обычно проявляется в более плавном росте точности (Accuracy) и снижении разницы между обучающей и валидационной метриками.

4. Формирование генераторов данных

Использование генераторов позволяет эффективно работать с большими датасетами, динамически подготавливая батчи изображений. Это экономит оперативную память и ускоряет процесс обучения.

Связь с графиками:



Рис. 5. Графики обучения модели

Training Accuracy / Loss (зелёные линии):

Высокая и быстро растущая точность на обучающей выборке говорит о том, что модель хорошо запоминает обучающие данные, чему способствует качественная предобработка и аугментация.

Validation Accuracy / Loss (красные линии):

Если бы предобработка была выполнена плохо (например, не было бы нормализации или аугментации), на графике была бы видна большая разница между обучающей и валидационной точностью (overfitting). Благодаря data\_loader.py, эта разница уменьшается, а валидационная точность постепенно растёт.

Пики и падения на графике Validation Loss:

Иногда на первых эпохах наблюдается нестабильность, связанная с тем, что модель только начинает учиться на аугментированных данных. Со временем, благодаря правильной подготовке данных, значения loss стабилизируются и снижаются.

**Итог**

Модуль data\_loader.py обеспечивает стандартизацию, разнообразие и качество входных данных, что напрямую влияет на успешность обучения модели. Благодаря его работе:

Модель быстрее и стабильнее обучается,

Снижается риск переобучения,

Повышается точность и качество обобщения на новых данных,

На графиках (рис.5) наблюдается положительная динамика метрик и устойчивое снижение функции потерь.

**3.2.3 Архитектура сверточной нейронной сети (model.py)**

Основу системы составляет глубокая сверточная нейронная сеть, реализованная на платформе TensorFlow/Keras. Архитектура включает пять сверточных блоков, каждый из которых состоит из сверточного слоя, нормализации батча, пулинга и регуляризации Dropout. На выходе сети расположен полносвязный слой с одной сигмоидной функцией активации, выдающей вероятность наличия пневмонии на снимке(рис.6).

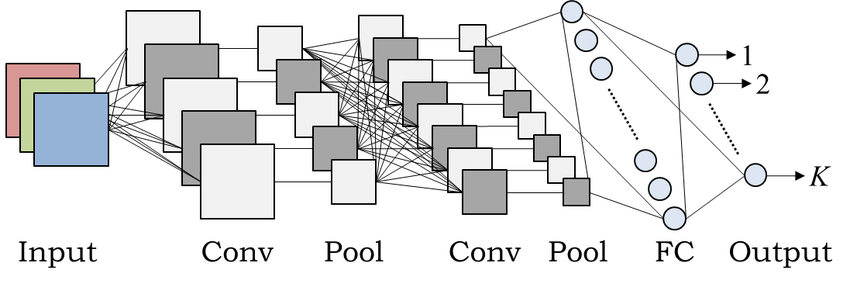


Рис.6 Типовая архитектура свёрточной нейронной сети

Роль в программной системе:

Сверточная нейронная сеть реализована в отдельном модуле (model.py), что облегчает её тестирование, повторное использование и возможную модификацию.

Сеть получает на вход данные, подготовленные модулем предобработки изображений, а результаты классификации используются для формирования итогового вывода о наличии или отсутствии пневмонии.

Модель интегрируется в серверное приложение, реализованное на Python с использованием Streamlit, что позволяет организовать удобный пользовательский интерфейс для загрузки снимков и получения результата.

Особенности реализации:

Перед подачей на вход сети изображения проходят этапы масштабирования, нормализации и, при необходимости, аугментации.

Модель обучается на размеченном наборе рентгеновских снимков с использованием стандартных методов машинного обучения, а её качество оценивается на отложенной тестовой выборке с помощью метрик точности, полноты, F1-меры и AUC-ROC.

Архитектура оптимизирована для запуска на современных персональных компьютерах с поддержкой GPU, что позволяет достичь баланса между скоростью работы и точностью классификации.

**3.2.4 Модуль обучения и оценки модели (train\_cnn.py, training.py)**

Этот компонент управляет процессом обучения нейронной сети. Основные задачи:

1. Компиляция модели с выбранным оптимизатором и функцией потерь.
2. Организация процесса обучения с применением генераторов данных, вычислением весов классов для балансировки, использованием коллбэков (ранняя остановка, снижение скорости обучения, сохранение лучшей модели).
3. Проведение оценки обученной модели на тестовой выборке с выводом ключевых метрик (точность, потери, отчёт классификации).

График обучения модели показан на (рис.3).

Также с помощью библиотеки matplotlib была представлена матрица ошибок(рис.7)

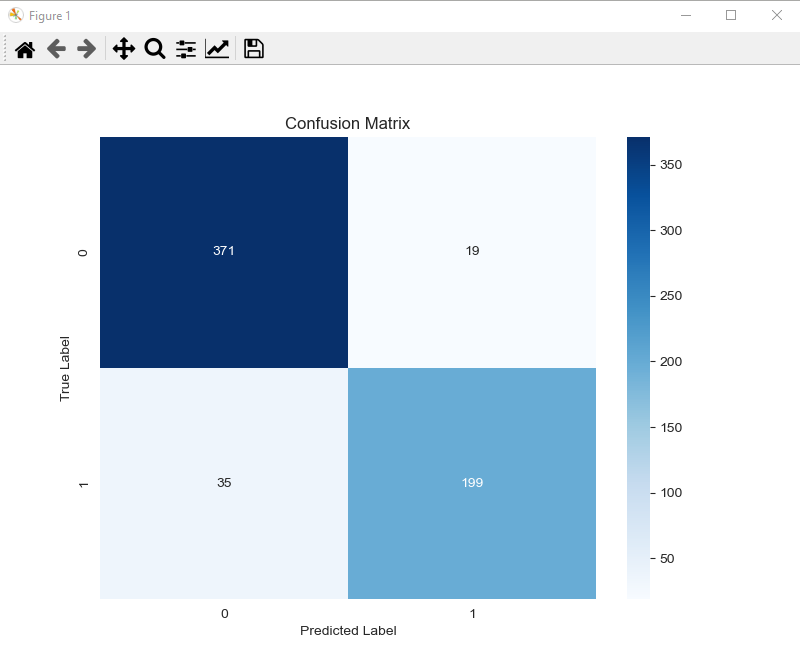


Рис.7 Матрица ошибок

Как читать матрицу ошибок:

Ось Y (True Label) — истинные метки классов (0 и 1).

Ось X (Predicted Label) — предсказанные моделью метки классов (0 и 1).

Обозначения:

0 — класс "Пневмония" (или "Отсутствие пневмонии", зависит от вашей разметки)

1 — класс "Норма" (или наоборот)

Значения в ячейках:

Верхний левый угол (371) — True Negative (TN):

Модель правильно предсказала класс 0 (например, "Пневмония"), когда это действительно был класс 0.

Верхний правый угол (19) — False Positive (FP):

Модель ошибочно предсказала класс 1 ("Норма"), когда на самом деле был класс 0 ("Пневмония").

Нижний левый угол (35) — False Negative (FN):

Модель ошибочно предсказала класс 0 ("Пневмония"), когда на самом деле был класс 1 ("Норма").

Нижний правый угол (199) — True Positive (TP):

Модель правильно предсказала класс 1 ("Норма"), когда это действительно был класс 1.

Оценка качества работы модели:

Точность модели (Accuracy):

(TP+TN) / (TP+TN+FP+FN) = (199+371) / (199+371+19+35) = 570 / 624 ≈ 91,3%

Выводы:

1. Модель хорошо различает оба класса.
2. Ошибки (FP и FN) присутствуют, но их количество относительно невелико.
3. Важно, что количество FN (пропущенных случаев "Норма") немного больше, чем FP (ложно определённых "Норма"), что может быть критично для медицинских задач.

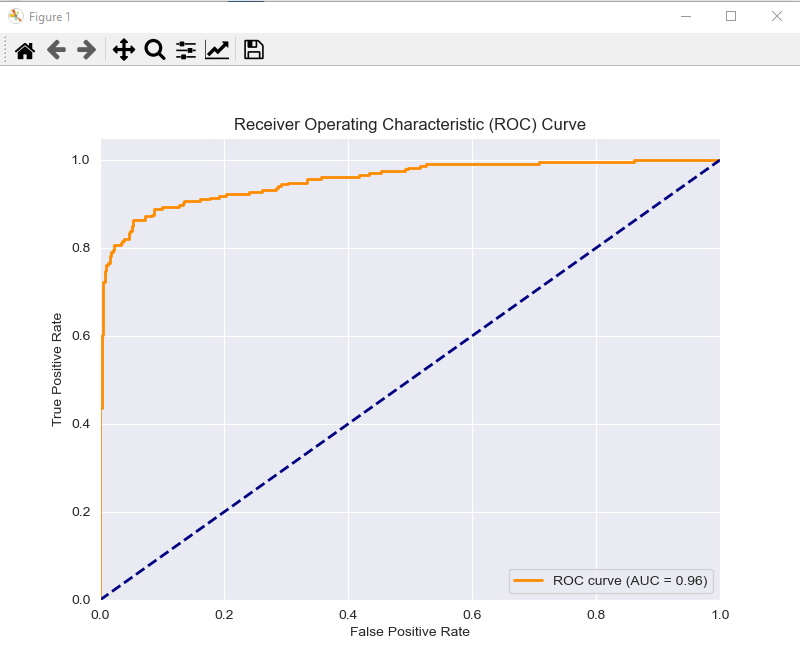


Рис.8 Оценки качества бинарной классификации снимков

Этот график(рис.8) — ROC-кривая (Receiver Operating Characteristic Curve), которая используется для оценки качества бинарной классификации.

Как читать график:

Ось X (False Positive Rate) — доля ложноположительных срабатываний (ошибочно определённых как положительный класс).

Ось Y (True Positive Rate) — доля истинно положительных (правильно определённых положительных случаев).

Оранжевая линия — ROC — кривая модели.

Пунктирная синяя линия — линия случайного угадывания (AUC = 0.5).

Ключевые выводы:

* AUC = 0.96 (Area Under Curve, площадь под кривой) — это очень высокий результат. Значение AUC может быть от 0 до 1. Чем ближе к 1, тем лучше модель различает два класса.
* Кривая значительно выше линии случайного угадывания - значит, модель уверенно отличает снимки с пневмонией от нормальных.
* Высокие значения True Positive Rate при низких False Positive Rate свидетельствуют о том, что модель правильно определяет большинство положительных случаев (например, обнаруживает пневмонию), при этом редко ошибается, принимая норму за болезнь.

Анализ статистики процесса обучения (рис.9):

Процесс обучения модели длится на GPU ≈ 127 секунд.

Весь процесс обучения поделен на 12 эпох, одна эпоха ≈ 10 секунд (на GPU).

Для вычислений использовалась NVIDIA GeForce GTX 1650 Ti with Max-Q Design.

А на CPU вычислениях в среднем в 3 три раза дольше.

На рис.9 представлен процесс обучения модели

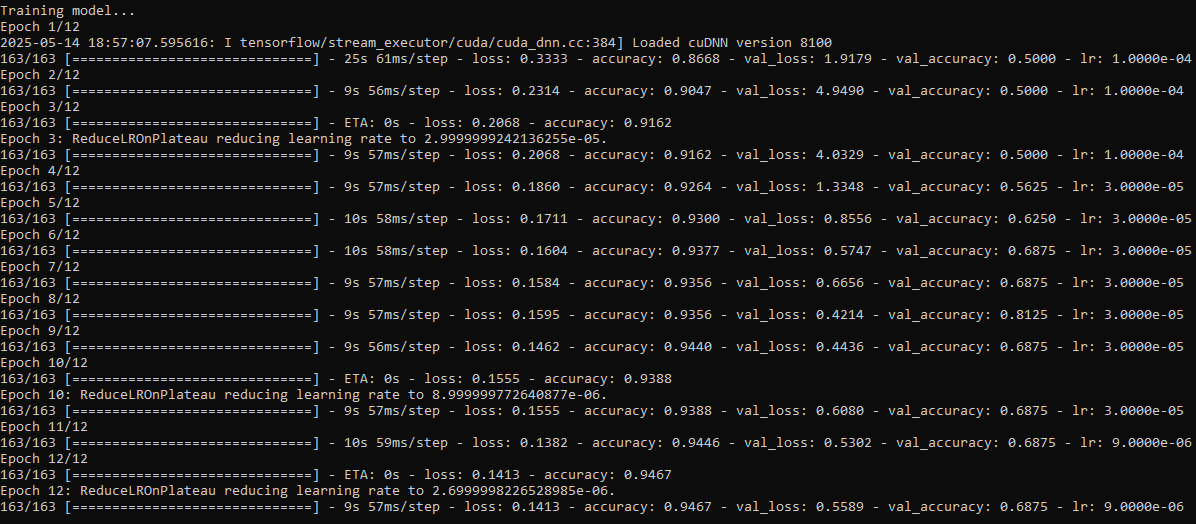


Рис.9 Статистика процесса обучения модели

Метрики:

* loss — значение функции потерь на обучающей выборке,
* accuracy — точность на обучающей выборке,
* val\_loss — значение функции потерь на валидационной выборке,
* val\_accuracy — точность на валидационной выборке,
* lr — текущая скорость обучения (learning rate).

Из рис.9 можно сделать итог:

* За 12 эпох точность на обучающей выборке выросла с 86.68% до 94.67%.
* Точность на валидационной выборке стабилизировалась на уровне 68.75%.
* Значение функции потерь на обучении снизилось с 3.3333 до 0.1413.
* Значение функции потерь на валидации снизилось с 1.9179 до 0.5589.
* Скорость обучения постепенно уменьшалась, что помогло избежать переобучения.

Рассмотрим результаты обучения на рис.10.

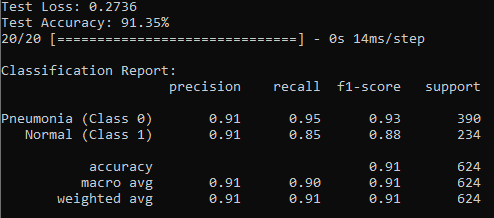


Рис.10 Результаты обучения

Сводные метрики:

* Общая точность (accuracy): 0.91 (91%)
* Среднее значение по классам (macro avg):
  + - Precision: 0.91
    - Recall: 0.90
    - F1-score: 0.91
* Взвешенное среднее (weighted avg):
  + - Precision: 0.91
    - Recall: 0.91
    - F1-score: 0.91

1. **ДЕМОНСТРАЦИЯ РАБОТЫ ПРИЛОЖЕНИЯ**
   1. **Программная реализация разработанной программы**

В этой части будет подробно описана работа скриптов проекта. На рис. 11 изображена диаграмма классов, отображающая связь компонентов программы.

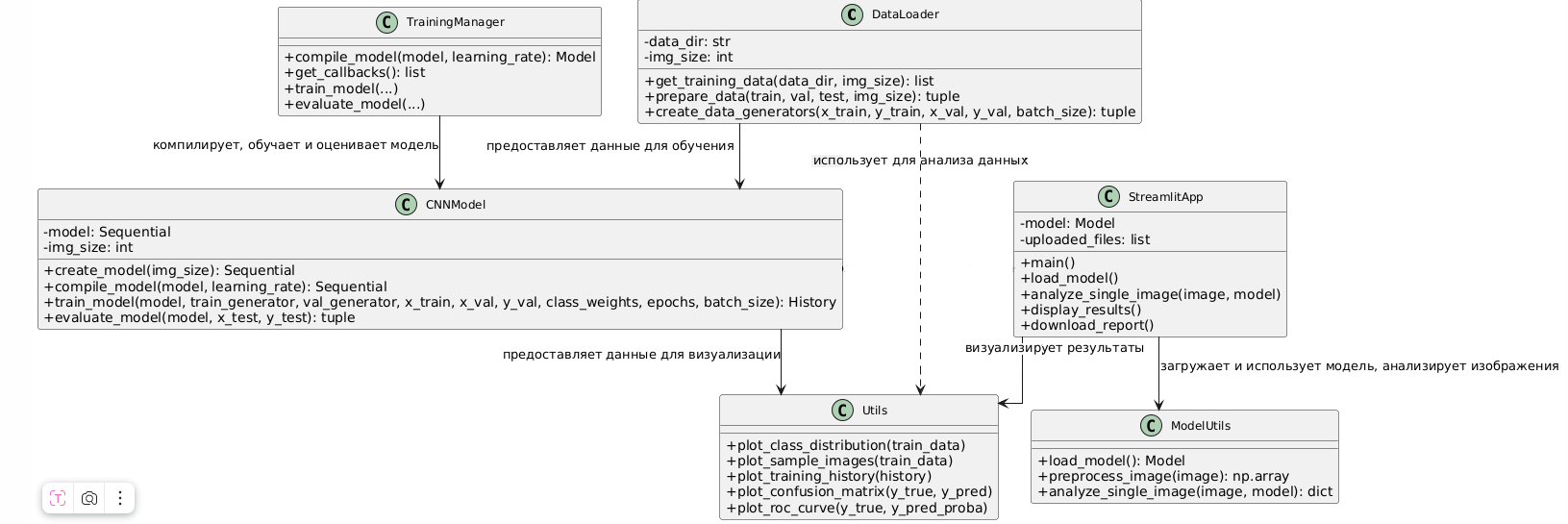


Рис. 11 Диаграмма классов компонентов системы

Файл *best\_model.h5* используется для загрузки обученной модели сверточной нейронной сети и последующего её применения для анализа рентгеновских снимков. Вот как это происходит по шагам:

Во время обучения модели (например, в модуле train\_cnn.py или аналогичном) после достижения наилучших результатов модель сохраняется в файл формата HDF5 с помощью метода Keras:

model.save('best\_model.h5')

Этот файл содержит полную архитектуру модели, веса, а также состояние оптимизатора.

В модуле обработки изображений (например, model\_utils.py) модель загружается из файла best\_model.h5 с помощью функции:

from tensorflow.keras.models import load\_model

model = load\_model('best\_model.h5')

Это восстанавливает модель со всеми сохранёнными параметрами и готовит её к предсказаниям.

Перед подачей на вход модели изображения проходят преобразование: изменение размера, нормализация, преобразование в нужный формат (например, 150×150 пикселей, оттенки серого, нормализация пикселей в диапазон 0-1).

После загрузки и предобработки изображения модель используется для получения вероятности принадлежности снимка к классу «пневмония» или «норма»:

prediction = model.predict(processed\_image)

В файле app.py загруженная модель применяется к изображениям, которые пользователь загружает через веб-интерфейс. Результаты предсказаний отображаются на странице, позволяя пользователю получить диагностику в режиме реального времени.

* + 1. Загрузка модели (model\_utils.py)

def load\_model():

@st.cache\_resource

def \_load():

return tf.keras.models.load\_model('best\_model.h5')

return \_load()

4.1.2 Процесс анализа изображения

4.1.2.1 Загрузка изображения пользователем(app.py)

uploaded\_files = st.file\_uploader(

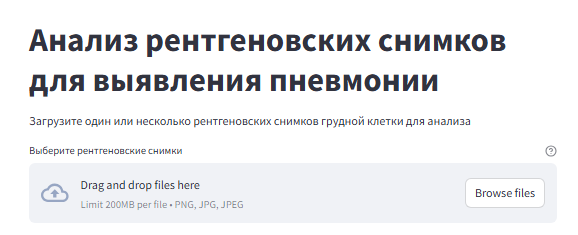
"Выберите рентгеновские снимки",

type=['png', 'jpg', 'jpeg'],

accept\_multiple\_files=True

)

image = Image.open(uploaded\_files[0])



*Рис. 12 Интерфейс загрузки изображения*

4.1.2.2 Предобработка изображения (model\_utils.py)

def preprocess\_image(image):

# Изменение размера до 150x150

img = image.resize((150, 150))

# Преобразование в оттенки серого

img = img.convert('L')

# Преобразование в numpy массив

img\_array = np.array(img)

# Добавление размерностей для батча и каналов

img\_array = np.expand\_dims(img\_array, axis=0) # Добавляем размерность батча

img\_array = np.expand\_dims(img\_array, axis= - 1) # Добавляем размерность канала

# Нормализация значений пикселей

img\_array = img\_array / 255.0

return img\_array

# Пример входного изображения:

# Размер: 1024x1024

# Формат: RGB

# Значения пикселей: 0 - 255

# После обработки:

# Размер: 150x150

# Формат: Grayscale

# Значения пикселей: 0 - 1

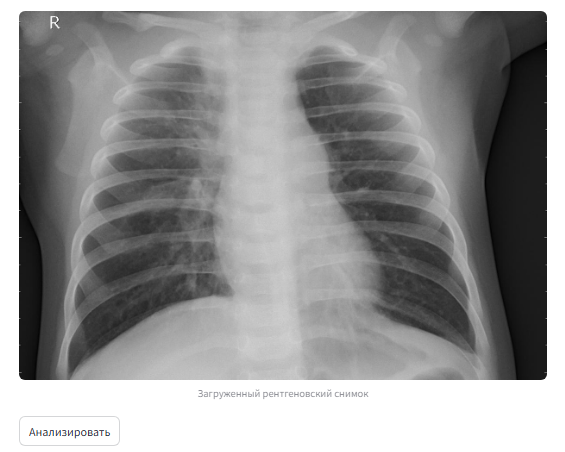


Рис. 13 Пример предобработки изображения

4.1.2.3 Анализ изображения (model\_utils.py)

def analyze\_single\_image(image, model):

# Предобработка изображения

processed\_image = preprocess\_image(image)

# Получение предсказания от модели

prediction = model.predict(processed\_image)[0][0]

# Определение результата

result = "Норма" if prediction > 0.5 else "Пневмония"

# Расчет уверенности

confidence = prediction if prediction > 0.5 else 1 - prediction

return {

"Результат": result,

"Уверенность": f"{confidence:.2%}",

"Вероятность нормы": f"{prediction:.2%}",

"Вероятность пневмонии": f"{(1 - prediction):.2%}"

}

# Пример входных данных:

# prediction = 0.85

# Пример выходных данных:

# {

# "Результат": "Норма",

# "Уверенность": "85.00%",

# "Вероятность нормы": "85.00%",

# "Вероятность пневмонии": "15.00%"

# }

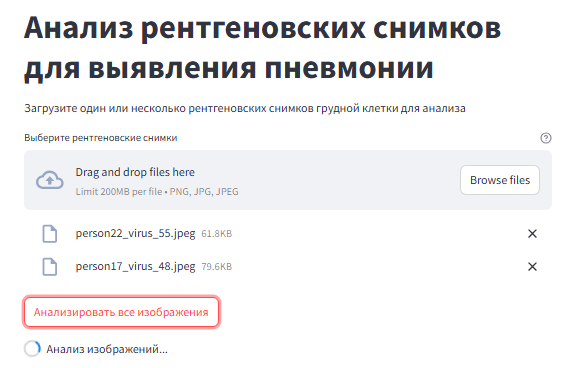


Рис. 14 Схема процесса анализа

* + 1. Вывод результатов (app.py)
       1. Одиночный анализ

if len(uploaded\_files) == 1:

image = Image.open(uploaded\_files[0])

st.image(image, caption='Загруженный рентгеновский снимок', use\_container\_width=True)

if st.button('Анализировать'):

with st.spinner('Анализ изображения...'):

result = analyze\_single\_image(image, model)

st.write("- - - ")

st.subheader("Результат анализа:")

if result["Результат"] == "Норма":

st.success(f"Норма (уверенность: {result['Уверенность']})")

else:

st.error(f"Обнаружена пневмония (уверенность: {result['Уверенность']})")

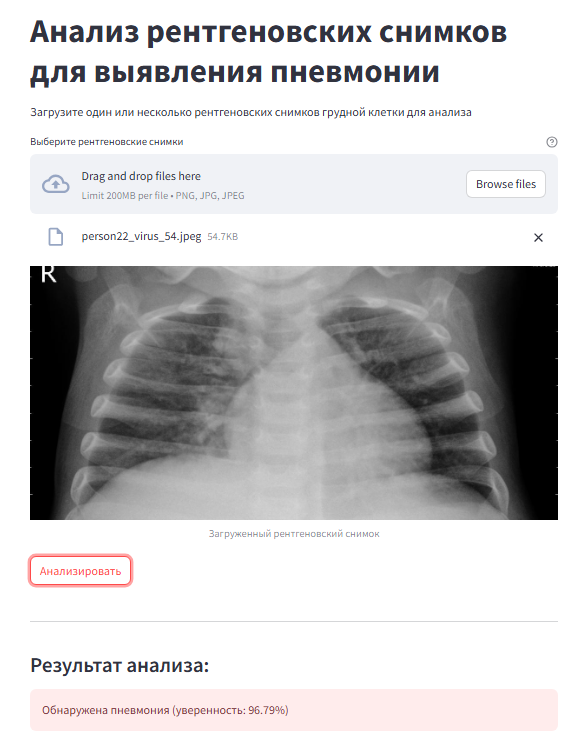


Рис. 15 Схема процесса анализа

* + - 1. Пакетный анализ(два и более изображений)

if st.button('Анализировать все изображения'):

with st.spinner('Анализ изображений...'):

results = []

for uploaded\_file in uploaded\_files:

image = Image.open(uploaded\_file)

result = analyze\_single\_image(image, model)

result['Имя файла'] = uploaded\_file.name

result['Изображение'] = image

results.append(result)

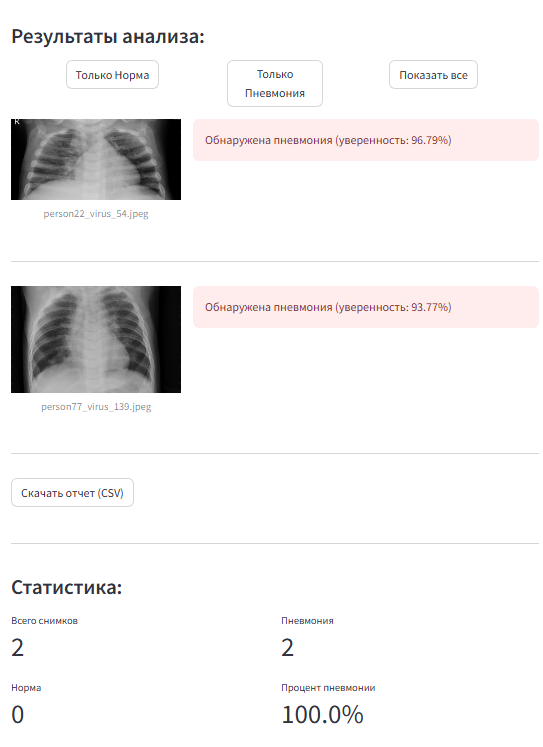


Рис. 16 Схема процесса анализа

* + 1. Скачивание отчета в csv (app.py)

if results:

output = io.StringIO()

writer = csv.writer(output)

writer.writerow(['Имя файла', 'Результат', 'Уверенность', 'Вероятность нормы', 'Вероятность пневмонии'])

for r in results:

writer.writerow([

r['Имя файла'],

r['Результат'],

r['Уверенность'],

r['Вероятность нормы'],

r['Вероятность пневмонии']

])

st.download\_button(

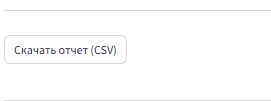
label='Скачать отчет (CSV)',

data=output.getvalue(),

file\_name='pneumonia\_analysis\_report.csv',

mime='text/csv'

)



*Рис. 16 Кнопка скачивания отчета*

Возьмем 200 снимков с пневмонией и загрузим их на анализ.

Получим результат(рис.17):

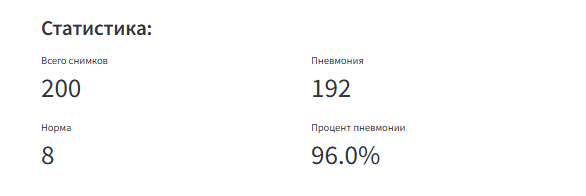


Рис. 17 Результат анализа 200 снимков

Из 200 снимков, всего 8-ошибочны, их критичность смягчает метрика “Уверенность”.

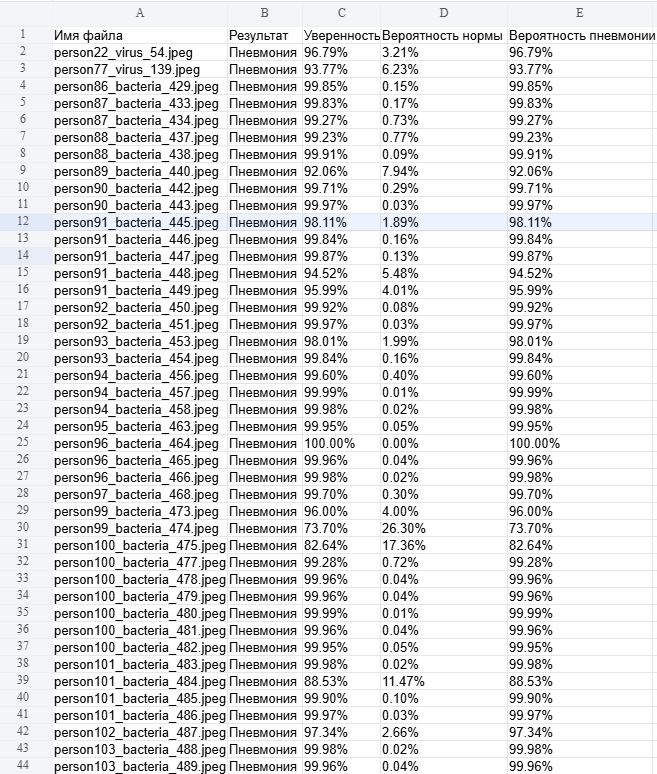
**

Рис. 18 Пример отчета с проанализированными снимками

* + 1. Данные модели CNN(model.py)

def create\_model(img\_size=150):

model = Sequential([

# Первый сверточный блок

Conv2D(32, (3,3), activation='relu', padding='same', input\_shape=(img\_size, img\_size, 1)),

BatchNormalization(),

MaxPool2D((2,2), padding='same'),

# Второй сверточный блок

Conv2D(64, (3,3), activation='relu', padding='same'),

Dropout(0.1),

BatchNormalization(),

MaxPool2D((2,2), padding='same'),

# Третий сверточный блок

Conv2D(64, (3,3), activation='relu', padding='same'),

BatchNormalization(),

MaxPool2D((2,2), padding='same'),

# Четвертый сверточный блок

Conv2D(128, (3,3), activation='relu', padding='same'),

Dropout(0.2),

BatchNormalization(),

MaxPool2D((2,2), padding='same'),

# Пятый сверточный блок

Conv2D(256, (3,3), activation='relu', padding='same'),

Dropout(0.2),

BatchNormalization(),

MaxPool2D((2,2), padding='same'),

Flatten(),

Dense(128, activation='relu'),

Dropout(0.2),

Dense(1, activation='sigmoid')

])

return model

Описание архитектуры:

* Входной слой: принимает изображение размером 150x150 пикселей в оттенках серого (1 канал).
* Сверточные блоки: пять блоков, каждый из которых состоит из сверточного слоя, нормализации, пулинга и иногда Dropout для регуляризации.
* Flatten: преобразует выход из сверточных слоев в одномерный вектор.
* Dense(128, activation='relu'): полносвязный слой с 128 нейронами.
* Dropout(0.2): регуляризация.
* Dense(1, activation='sigmoid'): выходной слой для бинарной классификации (пневмония/норма).

**4.2 Тестирование программы на тестовых снимках**

Тестирование проводилось на тестовых данных(с пневмонией), на которых моя модель не обучалась. Тестовые данные были сделаны в той же среде только на других людях. Статистика анализа(рис.19).



Рис. 19 Статистика анализа на тестовых данных

В ходе тестирования приложения мною было принято решение ввести метрику “Уверенность”.

На (рис.20) представлен пример с низкой уверенностью “Нормы”

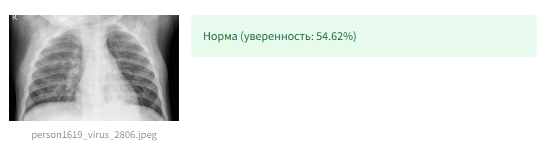


Рис. 20 Статистика анализа на тестовых данных

Это количественная метрика избавит врача от критичных решений в сторону пациента.

Выводы:

Приложение корректно обрабатывает загруженные снимки и формирует статистику по классам. Высокий процент выявленной пневмонии может быть обусловлен спецификой тестовой выборки или особенностями работы модели. Для окончательной оценки качества рекомендуется провести дополнительный анализ на сбалансированной выборке и рассчитать основные метрики качества классификации.[6]

**4.3 Оценка эффективности программы и ее применимость в реальных условиях**

* Разработанная программа для автоматизированного выявления пневмонии на рентгеновских снимках с использованием искусственного интеллекта демонстрирует высокую эффективность и имеет значительный потенциал для применения в клинической практике.
* Современные исследования и опыт внедрения подобных систем показывают, что модели на основе нейросетей способны с точностью 90–95% выявлять признаки пневмонии, включая вирусную пневмонию, вызванную COVID-19. Это позволяет значительно ускорить процесс диагностики по сравнению с традиционными методами и снизить нагрузку на врачей- рентгенологов.
* В отличие от компьютерной томографии, которая является дорогостоящей и сопровождается высокой лучевой нагрузкой, использование рентгенографии с интеллектуальной классификацией изображений позволяет проводить массовый скрининг с минимальными затратами и рисками для пациента. Это особенно актуально в условиях ограниченного доступа к КТ- оборудованию.
* Программа служит вспомогательным инструментом, предоставляя врачам оперативные и объективные данные о наличии и степени поражения легких. Это снижает вероятность ошибок и повышает качество медицинской помощи.
* В условиях пандемии COVID-19 и других острых респираторных заболеваний возможность быстрого выявления пневмонии с помощью ИИ- системы становится критически важной. Программа позволяет оперативно выявлять пациентов с подозрением на пневмонию и направлять их на дальнейшее обследование и лечение.
* Ведутся работы по расширению возможностей системы, включая обучение нейросети распознавать бактериальную пневмонию и другие патологии легких. Также планируется интеграция с другими методами визуализации и расширение области применения в различных медицинских сферах.
* Несмотря на высокую информативность, рентгенография и ИИ- диагностика не всегда позволяют выявить мелкие очаги воспаления или точно определить возбудителя. Поэтому программа должна использоваться в комплексе с клиническими данными и другими инструментальными методами (УЗИ, КТ) для повышения точности диагностики.[8]

**ЗАКЛЮЧЕНИЕ**

Актуальность темы выпускной квалификационной работы подтверждается необходимостью повышения эффективности и точности диагностики пневмонии-одного из наиболее распространённых и опасных заболеваний органов дыхания. В современных условиях, когда медицинские учреждения сталкиваются с большими объёмами рентгенологических исследований и высокой нагрузкой на специалистов, внедрение автоматизированных систем анализа медицинских изображений становится особенно востребованным.

Цель и задачи исследования были чётко сформулированы и реализованы в ходе работы. В процессе выполнения выпускной квалификационной работы проведён анализ современных методов искусственного интеллекта и глубокого обучения, рассмотрены существующие решения и обоснован выбор технологий для построения собственной системы. Были собраны и подготовлены данные, разработана архитектура программы, включающая модули предобработки изображений, построения и обучения сверточной нейронной сети, визуализации результатов и создания удобного пользовательского интерфейса.

Объектом исследования стали рентгеновские снимки грудной клетки пациентов, а предметом-методы и программные средства автоматического анализа этих изображений для выявления признаков пневмонии. Полученные результаты представляют собой ценный вклад в развитие цифровой медицины, способствуя внедрению инновационных технологий в практику здравоохранения.

Научная новизна выпускной квалификационной работы заключается в успешной реализации программной системы, способной автоматически анализировать рентгеновские снимки и выявлять признаки пневмонии с помощью современных нейросетевых моделей. Практическая значимость подтверждается возможностью применения разработанного решения для ускорения и повышения качества диагностики, снижения нагрузки на медицинский персонал и оптимизации работы медицинских учреждений.

В ходе работы был проведён глубокий анализ предметной области, рассмотрены требования к системе, обоснован выбор программных средств и реализована модульная архитектура приложения. Были успешно реализованы ключевые компоненты: модуль загрузки и предобработки данных, архитектура и обучение сверточной нейронной сети, модуль визуализации и веб- интерфейс для работы с пользователем.

Проведено тестирование программы на реальных данных, что позволило оценить её функциональность, устойчивость и корректность работы. Результаты показали, что система способна эффективно выявлять пневмонию по рентгеновским снимкам, демонстрируя высокие показатели точности и обобщающей способности.

В целом, разработанная программа представляет собой инновационное решение, позволяющее автоматизировать процесс диагностики пневмонии и повысить качество медицинской помощи. Её успешная реализация, тестирование и оценка эффективности свидетельствуют о перспективности применения искусственного интеллекта в сфере медицинской диагностики. Полученные результаты могут быть полезны для дальнейших исследований и внедрения подобных систем в клиническую практику.

На основе проведённого исследования и разработки программы можно выделить следующие рекомендации по дальнейшему развитию:

1. Рассмотреть возможность интеграции программы с существующими системами учёта и мониторинга пациентов для повышения эффективности работы медицинского персонала.
2. Добавить поддержку анализа других заболеваний лёгких, внедрить модули аналитики и прогнозирования, а также реализовать функции пакетной обработки и формирования отчётов для врачей.
3. Оптимизировать интерфейс приложения для повышения удобства работы медицинских специалистов, добавить возможности визуализации динамики изменений и интеграции с мобильными устройствами.
4. Продолжить работу над улучшением методов предобработки изображений, аугментации данных и обучения моделей для повышения точности и устойчивости системы.[12]
5. Следить за развитием технологий искусственного интеллекта и компьютерного зрения, применять современные подходы (например, transfer learning, explainable AI) для повышения качества диагностики и расширения возможностей программы.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Глубокое обучение для анализа медицинских изображений: комплексный обзор [Электронный ресурс] – URL: https://www.nature.com/articles/s41598 — 021 — 87644 — 7 (Дата обращения: 12.05.2024)
2. Анализ рентгеновских снимков грудной клетки с использованием глубокого обучения: систематический обзор [Электронный ресурс] – URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7572230/ (Дата обращения: 12.05.2024)
3. Выявление пневмонии на рентгеновских снимках грудной клетки с помощью глубокого обучения [Электронный ресурс] – URL: https://www.sciencedirect.com/science/article/pii/S0010482521001137 (Дата обращения: 15.05.2024)
4. Документация Streamlit: Создание приложений для анализа данных [Электронный ресурс] – URL: https://docs.streamlit.io/ (Дата обращения: 15.05.2024)
5. Анализ медицинских изображений с помощью глубокого обучения: лучшие практики [Электронный ресурс] – URL: https://www.tensorflow.org/tutorials/images/classification (Дата обращения: 15.05.2024)
6. Метрики оценки для классификации медицинских изображений [Электронный ресурс] – URL:https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6616181/ (Дата обращения: 18.05.2024)
7. Руководство по разработке программного обеспечения для анализа медицинских изображений [Электронный ресурс] – URL: https://www.fda.gov/medical — devices/software — medical — device — samd/guidance — documents — medical — devices — containing — or — consisting — software (Дата обращения: 18.05.2024)
8. Оценка качества программного обеспечения для медицинской визуализации [Электронный ресурс] – URL: https://www.iso.org/standard/63182.html (Дата обращения: 21.05.2024)
9. Лучшие практики предварительной обработки медицинских изображений [Электронный ресурс] – URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7066214/ (Дата обращения: 21.05.2024)
10. Внедрение моделей медицинского ИИ в клиническую практику [Электронный ресурс] – URL: https://www.nature.com/articles/s41591 — 019 — 0641 — x (Дата обращения: 22.05.2024)
11. Этические аспекты разработки медицинского ИИ [Электронный ресурс] – URL: https://www.who.int/publications/i/item/9789240029200 (Дата обращения: 21.05.2024)
12. Анализ медицинских изображений: текущие тенденции и перспективы развития [Электронный ресурс] – URL: https://www.sciencedirect.com/science/article/pii/S1361841521001771 (Дата обращения: 21.05.2024)

**ПРИЛОЖЕНИЕ**

**Загрузка изображений**

|  |
| --- |
| **data\_loader.py** |
| import os  import cv2  import numpy as np  from tensorflow.keras.preprocessing.image import ImageDataGenerator  def get\_training\_data(data\_dir, img\_size=150):  data = []  labels = ['PNEUMONIA', 'NORMAL']    for label in labels:  path = os.path.join(data\_dir, label)  class\_num = labels.index(label)  for img in os.listdir(path):  try:  img\_path = os.path.join(path, img)  if not img.lower().endswith(('.png', '.jpg', '.jpeg')):  continue  img\_arr = cv2.imread(img\_path, cv2.IMREAD\_GRAYSCALE)  if img\_arr is None:  continue  resized\_arr = cv2.resize(img\_arr, (img\_size, img\_size))  data.append([resized\_arr, class\_num])  except Exception as e:  print(f"Error processing {img\_path}: {e}")  return data  def prepare\_data(train, val, test, img\_size=150):  x\_train = np.array([item[0] for item in train]) / 255.0  y\_train = np.array([item[1] for item in train])    x\_val = np.array([item[0] for item in val]) / 255.0  y\_val = np.array([item[1] for item in val])    x\_test = np.array([item[0] for item in test]) / 255.0  y\_test = np.array([item[1] for item in test])    # Изменение формы для CNN  x\_train = x\_train.reshape( — 1, img\_size, img\_size, 1)  x\_val = x\_val.reshape( — 1, img\_size, img\_size, 1)  x\_test = x\_test.reshape( — 1, img\_size, img\_size, 1)    return x\_train, y\_train, x\_val, y\_val, x\_test, y\_test  def create\_data\_generators(x\_train, y\_train, x\_val, y\_val, batch\_size=32):  train\_datagen = ImageDataGenerator(  rotation\_range=30,  zoom\_range=0.2,  width\_shift\_range=0.1,  height\_shift\_range=0.1,  horizontal\_flip=True,  vertical\_flip=False  )    val\_datagen = ImageDataGenerator()    train\_generator = train\_datagen.flow(x\_train, y\_train, batch\_size=batch\_size)  val\_generator = val\_datagen.flow(x\_val, y\_val, batch\_size=batch\_size)    return train\_generator, val\_generator |

**Модель и обучение**

|  |
| --- |
| **model.py** |
| from tensorflow.keras.models import Sequential  from tensorflow.keras.layers import Dense, Conv2D, MaxPool2D, Flatten, Dropout, BatchNormalization  def create\_model(img\_size=150):    model = Sequential([    Conv2D(32, (3,3), activation='relu', padding='same', input\_shape=(img\_size, img\_size, 1)),  BatchNormalization(),  MaxPool2D((2,2), padding='same'),  Conv2D(64, (3,3), activation='relu', padding='same'),  Dropout(0.1),  BatchNormalization(),  MaxPool2D((2,2), padding='same'),  Conv2D(64, (3,3), activation='relu', padding='same'),  BatchNormalization(),  MaxPool2D((2,2), padding='same'),  Conv2D(128, (3,3), activation='relu', padding='same'),  Dropout(0.2),  BatchNormalization(),  MaxPool2D((2,2), padding='same'),    Conv2D(256, (3,3), activation='relu', padding='same'),  Dropout(0.2),  BatchNormalization(),  MaxPool2D((2,2), padding='same'),  Flatten(),  Dense(128, activation='relu'),  Dropout(0.2),  Dense(1, activation='sigmoid')  ])    return model |

|  |
| --- |
| **train\_cnn.py** |
| import tensorflow as tf  import os  import numpy as np  from data\_loader import get\_training\_data, prepare\_data, create\_data\_generators  from model import create\_model  from training import compile\_model, train\_model, evaluate\_model  from utils import (plot\_class\_distribution, plot\_sample\_images,  plot\_training\_history, plot\_confusion\_matrix, plot\_roc\_curve)  # Конфигурация  IMG\_SIZE = 150  BATCH\_SIZE = 32  EPOCHS = 12  LEARNING\_RATE = 0.0001  def main():  physical\_devices = tf.config.list\_physical\_devices('GPU')  if len(physical\_devices) > 0:  print(f"GPUs available: {[d.name for d in physical\_devices]}")  try:  for gpu in physical\_devices:  tf.config.experimental.set\_memory\_growth(gpu, True)  except Exception as e:  print(f"Could not set memory growth: {e}")  else:  print("WARNING: GPU not found! Вычисления будут идти на CPU.")    print("\nLoading data...")  train\_data = get\_training\_data('chest\_xray/train', IMG\_SIZE)  val\_data = get\_training\_data('chest\_xray/val', IMG\_SIZE)  test\_data = get\_training\_data('chest\_xray/test', IMG\_SIZE)    print("\nVisualizing data distribution...")  plot\_class\_distribution(train\_data)  plot\_sample\_images(train\_data)    print("\nPreparing data...")  x\_train, y\_train, x\_val, y\_val, x\_test, y\_test = prepare\_data(  train\_data, val\_data, test\_data, IMG\_SIZE  )    print("\nCreating data generators...")  train\_generator, val\_generator = create\_data\_generators(  x\_train, y\_train, x\_val, y\_val, BATCH\_SIZE  )  print("\nCreating and compiling model...")  model = create\_model(IMG\_SIZE)  model = compile\_model(model, LEARNING\_RATE)    print("\nComputing class weights...")  class\_weights = {  0: len(y\_train) / (2 \* np.sum(y\_train == 0)),  1: len(y\_train) / (2 \* np.sum(y\_train == 1))  }    print("\nTraining model...")  history = train\_model(  model, train\_generator, val\_generator,  x\_train, x\_val, y\_val, class\_weights,  EPOCHS, BATCH\_SIZE  )    print("\nPlotting training history...")  plot\_training\_history(history)    print("\nEvaluating model...")  loss, accuracy, predictions = evaluate\_model(model, x\_test, y\_test)  print("\nPlotting evaluation results...")  plot\_confusion\_matrix(y\_test, predictions)  plot\_roc\_curve(y\_test, model.predict(x\_test))  if \_\_name\_\_ == "\_\_main\_\_":  print(tf.\_\_version\_\_)  main() |

|  |
| --- |
| **training.py** |
| import tensorflow as tf  from tensorflow.keras.callbacks import ReduceLROnPlateau, ModelCheckpoint, EarlyStopping  from tensorflow.keras.optimizers import Adam  from sklearn.metrics import classification\_report  import numpy as np  def compile\_model(model, learning\_rate=0.0001):  model.compile(  optimizer=Adam(learning\_rate=learning\_rate),  loss='binary\_crossentropy',  metrics=['accuracy']  )  return model  def get\_callbacks():  return [  EarlyStopping(  monitor='val\_accuracy',  patience=5,  restore\_best\_weights=True,  verbose=1  ),  ReduceLROnPlateau(  monitor='val\_accuracy',  factor=0.3,  patience=2,  min\_lr=1e — 6,  mode='max',  verbose=1  ),  ModelCheckpoint(  'best\_model.h5',  monitor='val\_accuracy',  save\_best\_only=True,  mode='max'  )  ]  def train\_model(model, train\_generator, val\_generator, x\_train, x\_val, y\_val,  class\_weights, epochs=12, batch\_size=32):  history = model.fit(  train\_generator,  steps\_per\_epoch=len(x\_train) // batch\_size,  epochs=epochs,  validation\_data=(x\_val, y\_val),  callbacks=get\_callbacks(),  class\_weight=class\_weights  )  return history  def evaluate\_model(model, x\_test, y\_test):  loss, accuracy = model.evaluate(x\_test, y\_test)  print(f"Test Loss: {loss:.4f}")  print(f"Test Accuracy: {accuracy\*100:.2f}%")    predictions = (model.predict(x\_test) > 0.5).astype("int32").reshape( — 1)  print("\nClassification Report:")  print(classification\_report(y\_test, predictions, target\_names=['Pneumonia (Class 0)', 'Normal (Class 1)']))    return loss, accuracy, predictions |

**Утилиты для визуализации**

|  |
| --- |
| **utils.py** |
| import matplotlib.pyplot as plt  import seaborn as sns  import numpy as np  from sklearn.metrics import confusion\_matrix, roc\_curve, auc  def plot\_class\_distribution(train\_data, labels=['Pneumonia', 'Normal']):  l = [labels[i[1]] for i in train\_data]  sns.set\_style('darkgrid')  plt.figure(figsize=(6,4))  sns.countplot(l)  plt.title('Class distribution in training set')  plt.show()  def plot\_sample\_images(train\_data, labels=['PNEUMONIA', 'NORMAL'], num\_samples=2):  plt.figure(figsize=(5\*num\_samples, 5))  for i in range(num\_samples):  plt.subplot(1, num\_samples, i+1)  plt.imshow(train\_data[i][0], cmap='gray')  plt.title(labels[train\_data[i][1]])  plt.show()  def plot\_training\_history(history):    epochs = range(1, len(history.history['accuracy']) + 1)    plt.figure(figsize=(14,5))    plt.subplot(1,2,1)  plt.plot(epochs, history.history['accuracy'], 'go — ', label='Training Accuracy')  plt.plot(epochs, history.history['val\_accuracy'], 'ro — ', label='Validation Accuracy')  plt.title('Training & Validation Accuracy')  plt.xlabel('Epochs')  plt.ylabel('Accuracy')  plt.legend()    plt.subplot(1,2,2)  plt.plot(epochs, history.history['loss'], 'go — ', label='Training Loss')  plt.plot(epochs, history.history['val\_loss'], 'ro — ', label='Validation Loss')  plt.title('Training & Validation Loss')  plt.xlabel('Epochs')  plt.ylabel('Loss')  plt.legend()    plt.show()  def plot\_confusion\_matrix(y\_true, y\_pred):  cm = confusion\_matrix(y\_true, y\_pred)  plt.figure(figsize=(8, 6))  sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')  plt.title('Confusion Matrix')  plt.ylabel('True Label')  plt.xlabel('Predicted Label')  plt.show()  def plot\_roc\_curve(y\_true, y\_pred\_proba):  fpr, tpr, \_ = roc\_curve(y\_true, y\_pred\_proba)  roc\_auc = auc(fpr, tpr)    plt.figure(figsize=(8, 6))  plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (AUC = {roc\_auc:.2f})')  plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle=' — — ')  plt.xlim([0.0, 1.0])  plt.ylim([0.0, 1.05])  plt.xlabel('False Positive Rate')  plt.ylabel('True Positive Rate')  plt.title('Receiver Operating Characteristic (ROC) Curve')  plt.legend(loc="lower right")  plt.show() |

**Вспомогательные функции для работы с моделью**

|  |
| --- |
| **model\_utils.py** |
| import tensorflow as tf  import numpy as np  from PIL import Image  import streamlit as st  def load\_model():  @st.cache\_resource  def \_load():  return tf.keras.models.load\_model('best\_model.h5')  return \_load()  def preprocess\_image(image):  img = image.resize((150, 150))  img = img.convert('L')  img\_array = np.array(img)  img\_array = np.expand\_dims(img\_array, axis=0)  img\_array = np.expand\_dims(img\_array, axis= — 1)  img\_array = img\_array / 255.0  return img\_array  def analyze\_single\_image(image, model):  processed\_image = preprocess\_image(image)  prediction = model.predict(processed\_image)[0][0]  result = "Норма" if prediction > 0.5 else "Пневмония"  confidence = prediction if prediction > 0.5 else 1 — prediction  return {  "Результат": result,  "Уверенность": f"{confidence:.2%}",  "Вероятность нормы": f"{prediction:.2%}",  "Вероятность пневмонии": f"{(1 — prediction):.2%}"  } |

**Веб — приложение**

|  |
| --- |
| **app.py** |
| import streamlit as st  import pandas as pd  import io  import csv  from PIL import Image  from model\_utils import load\_model, preprocess\_image, analyze\_single\_image  def main():  st.title("Анализ рентгеновских снимков для выявления пневмонии")  st.write("Загрузите один или несколько рентгеновских снимков грудной клетки для анализа")  try:  model = load\_model()  except:  st.error("Ошибка загрузки модели. Убедитесь, что файл best\_model.h5 существует.")  return  uploaded\_files = st.file\_uploader(  "Выберите рентгеновские снимки",  type=['png', 'jpg', 'jpeg'],  accept\_multiple\_files=True,  help='Перетащите файлы сюда или кликните для выбора файлов.'  )  file\_names = [f.name for f in uploaded\_files] if uploaded\_files else []  if 'last\_files' not in st.session\_state:  st.session\_state['last\_files'] = []  if file\_names != st.session\_state['last\_files']:  if 'results' in st.session\_state:  del st.session\_state['results']  st.session\_state['last\_files'] = file\_names  if uploaded\_files:  if len(uploaded\_files) == 1:  image = Image.open(uploaded\_files[0])  st.image(image, caption='Загруженный рентгеновский снимок', use\_container\_width=True)  if st.button('Анализировать'):  with st.spinner('Анализ изображения...'):  result = analyze\_single\_image(image, model)  st.write(" — — — ")  st.subheader("Результат анализа:")  if result["Результат"] == "Норма":  st.success(f"Норма (уверенность: {result['Уверенность']})")  else:  st.error(f"Обнаружена пневмония (уверенность: {result['Уверенность']})")  else:  if st.button('Анализировать все изображения') or 'results' in st.session\_state:  if 'results' not in st.session\_state:  with st.spinner('Анализ изображений...'):  results = []  for uploaded\_file in uploaded\_files:  image = Image.open(uploaded\_file)  result = analyze\_single\_image(image, model)  result['Имя файла'] = uploaded\_file.name  result['Изображение'] = image  results.append(result)  st.session\_state['results'] = results  else:  results = st.session\_state['results']  st.write(" — — — ")  st.subheader("Результаты анализа:")  if 'filter' not in st.session\_state:  st.session\_state['filter'] = 'all'  cols = st.columns([1, 2, 1, 2, 1, 2, 1])  with cols[1]:  show\_normal = st.button('Только Норма', key='show\_normal')  with cols[3]:  show\_pneumonia = st.button('Только Пневмония', key='show\_pneumonia')  with cols[5]:  show\_all = st.button('Показать все', key='show\_all')  if show\_normal:  st.session\_state['filter'] = 'normal'  if show\_pneumonia:  st.session\_state['filter'] = 'pneumonia'  if show\_all:  st.session\_state['filter'] = 'all'  if st.session\_state['filter'] == 'normal':  filtered = [r for r in results if r['Результат'] == 'Норма']  elif st.session\_state['filter'] == 'pneumonia':  filtered = [r for r in results if r['Результат'] == 'Пневмония']  else:  filtered = results  for r in filtered:  col1, col2 = st.columns([1, 2])  with col1:  st.image(r['Изображение'], caption=r['Имя файла'], use\_container\_width=True)  with col2:  if r["Результат"] == "Норма":  st.success(f"Норма (уверенность: {r['Уверенность']})")  else:  st.error(f"Обнаружена пневмония (уверенность: {r['Уверенность']})")  st.write(" — — — ")  if results:  output = io.StringIO()  writer = csv.writer(output)  writer.writerow(['Имя файла', 'Результат', 'Уверенность', 'Вероятность нормы', 'Вероятность пневмонии'])  for r in results:  writer.writerow([  r['Имя файла'],  r['Результат'],  r['Уверенность'],  r['Вероятность нормы'],  r['Вероятность пневмонии']  ])  st.download\_button(  label='Скачать отчет (CSV)',  data=output.getvalue(),  file\_name='pneumonia\_analysis\_report.csv',  mime='text/csv'  )  st.write(" — — — ")  st.subheader("Статистика:")  total = len(results)  normal\_count = sum(1 for r in results if r['Результат'] == 'Норма')  pneumonia\_count = total — normal\_count  col1, col2 = st.columns(2)  with col1:  st.metric("Всего снимков", total)  st.metric("Норма", normal\_count)  with col2:  st.metric("Пневмония", pneumonia\_count)  st.metric("Процент пневмонии", f"{(pneumonia\_count/total)\*100:.1f}%")  if \_\_name\_\_ == "\_\_main\_\_"  main() |