МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное
учреждение высшего образования
**«Южно-Уральский государственный университет
(национальный исследовательский университет)»**
**Высшая школа электроники и компьютерных наук**
**Кафедра системного программирования**

РАБОТА ПРОВЕРЕНА
Рецензент
Инженер ОАО «Межрегиональная распределительная сетевая компания Урала»
_____В.В. Мякуш
“___”_____ 2020 г.

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой, д.ф.-м.н., профессор
_____ Л.Б. Соколинский
“___”_____ 2020 г.

# Разработка приложения для выявления пневмонии по рентгеновским снимкам с применением нейросетевых технологий

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 02.03.02.2020.115-103.ВКР

Научный руководитель,
ст. преподаватель кафедры СП
_____К.Ю. Никольская

Автор работы,
студентка группы КЭ-401
_____ Н.С. Ворошнина

Ученый секретарь
(нормоконтролер)
_____ И.Д. Володченко
“___”_____ 2020 г.

Челябинск 2020

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное
учреждение высшего образования
**«Южно-Уральский государственный университет**
**(национальный исследовательский университет)»**
**Высшая школа электроники и компьютерных наук**
**Кафедра системного программирования**

УТВЕРЖДАЮ
Зав. кафедрой СП
_____Л.Б. Соколинский
08.02.2020.

# ЗАДАНИЕ
**на выполнение выпускной квалификационной работы бакалавра**

студентке группы КЭ-401
Ворошниной Наталье Сергеевне,
обучающейся по направлению
02.03.02 «Фундаментальная информатика и информационные технологии»

**1. Тема работы** (утверждена приказом ректора от 24.04.2020 № 627)
Разработка приложения для выявления пневмонии по рентгеновским снимкам с применением нейросетевых технологий.
**2. Срок сдачи студентом законченной работы:** 05.06.2020.
**3. Исходные данные к работе**
3.1 Шолле Франсуа. Глубокое обучение на Python, 2018.
3.2 Саймон Хайкин, «Нейронные сети: полный курс, 2-е издание», 2006.
**4. Перечень подлежащих разработке вопросов**
4.1 Сделать обзор аналогов и научной литературы.
4.2 Спроектировать топологию искусственной нейронной сети и обучить ее.
4.3 Разработать приложение для классификации рентгеновских снимков.
4.4 Провести тестирование системы.
**5. Дата выдачи задания:** 03.02.2020.

**Научный руководитель,**
ст. преподаватель кафедры СП К.Ю. Никольская

**Задание принял к исполнению** Н.С. Ворошнина

**ОГЛАВЛЕНИЕ**

# ВВЕДЕНИЕ

## ОСНОВНЫЕ ОПРЕДЕЛЕНИЯ

Искусственные нейронные сети – это математические модели, а также их программные или аппаратные реализации, построенные по принципу организации и функционирования биологических нейронных сетей – сетей нервных клеток живого организма [6].

Машинное обучение – широкий набор методик, отличительной чертой которых является не прямой поиск решения, а постепенное обучение алгоритма способам нахождения решения [7].

Глубокое обучение – это особый раздел машинного обучения: новый подход к поиску представления данных, делающий упор на изучение последовательных слоев (или уровней) все более значимых представлений [1].

## АКТУАЛЬНОСТЬ ТЕМЫ

Разработка искусственных нейронных сетей началась еще в начале XX века, но более широкое распространение нейронные сети получили только в 90-х годах с увеличением мощности вычислительных систем. Изначально целью попыток создания нейронных сетей было понимание принципов работы человеческого мозга. На сегодняшний нейронная сеть представляет собой некую упрощенную модель человеческого мозга и успешно используется при решении самых различных задач.

Направление медицины и здравоохранения уже сегодня считается одним из стратегических и перспективных с точки зрения эффективного внедрения нейросетевых технологий. Их использование может массово повысить точность диагностики, облегчить жизнь пациентам с различными заболеваниями, повысить скорость разработки и выпуска новых лекарств и т.д.

## ЦЕЛЬ И ЗАДАЧИ ИССЛЕДОВАНИЯ

Основной целью данной работы является разработка приложения для выявления пневмонии по рентгеновским снимкам с применением нейросетевых технологий.

Для выполнения поставленной цели необходимо решить задачи, представлены ниже.

1. Сделать обзор аналогов и научной литературы.

2. Спроектировать топологию искусственной нейронной сети и обучить ее.

3. Разработать приложение для классификации рентгеновских снимков.

4. Провести тестирование системы.

## СТРУКТУРА И ОБЪЕМ РАБОТЫ

Работа состоит из введения, 5 глав, заключения и библиографии. Объем работы составляет 39 страниц, объем библиографии – 16 источников, объем приложений – 4 страницы.

## СОДЕРЖАНИЕ РАБОТЫ

В первой главе «Анализ предметной области» рассмотрены исследования по использованию нейросетевых технологий в области, в 7 рамках которой выполняется данная работа и обзор уже существующих решений поставленной задачи.

Вторая глава «Теоретическая часть» содержит подробное описание алгоритмов, применяемых для решения поставленной задачи.

Третья глава «Проектирование» содержит общее описание архитектуры системы.

В четвертой главе «Реализация» приводится техническая реализация системы, исходя из поставленного списка требований.

В пятой главе «Тестирование» тестирования реализованного приложения.

В заключении представлены результаты проделанной работы.

# 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

## 1.1. Изучение предметной области

Во всем мире 450 миллионов человек заражаются пневмонией в год и 4 миллиона человек умирают от этой болезни [8]. Численная разница между показателями заражения и показателями смертности показывает, насколько важен ранний диагноз этой болезни.

Пневмония – это острое воспалительное заболевание, поражающее органы дыхания, преимущественно легкие. Его часто вызывают вирусы, бактерии – пневмококки, стафилококки, стрептококки [8].

Основными симптомами заболевания являются сухой кашель и высокая температура. Через некоторое время может появиться мокрота, одышка, затрудненное дыхание.

Если инфекцию не лечить на раннем этапе, то пневмония может распространиться по всему организму и даже привести к смерти человека.

Сегодня одним из основных методов для диагностики заболевания является рентгенография грудной клетки. При изучении рентгеновского снимка органов грудной клетки обращают внимание на структуру костей и мягких тканей грудной клетки, форму и прозрачность областей лёгочных тканей, а также их структуру, сформированную тенями сосудов лёгких, расположение и структуру корней лёгких, положение, форму куполов диафрагмы и синусов плевральных полостей.

Во время диагностики специалисты-радиологи соотносят белые пятна на изображении, идентифицирующих инфекцию, и белые области, в которых скопилась жидкость вследствие пневмонии. Однако, из-за ограниченной цветовой схемы рентгеновских изображений, состоящих только из оттенков черного и белого, возникают затруднения, когда дело доходит до определения есть на снимке пневмония или нет. Всегда есть вероятность ложного диагноза. Если диагноз ложноположительный, то лечение может только навредить, если – ложноотрицательный, то без лечения пневмония

может привести к смерти. В обоих случаях неправильно поставленный диагноз может негативно воздействовать на организм человека.

Опытные радиологи не всегда могут поставить верный диагноз. А во многих развивающихся странах даже не хватает квалифицированных медицинских специалистов для своевременного диагностирования пневмонии.

Следовательно, вычислительные методы для диагностики заболевания являются наиболее надежными, так как исключают человеческий фактор и могут с большей точностью определять верный диагноз.

## 1.2. Распознавание образов

Способность «распознавать» считается основным свойством биологических существ, но компьютерные системы этим свойством не обладают. Распознавание образов – это наука о методах и алгоритмах классификации объектов различной природы [14].

Любая классификация всегда основывается на прецедентах. Прецедент – это ранее классифицированный объект, принимаемый как образец при решении задач классификации [15]. Все объекты или явления разбиты на конечное число классов. Для каждого класса известно конечное число прецедентов. Задача распознавания образов состоит в том, чтобы соотнести новый объект к какому-либо уже существующему классу. Задача распознавания образов является основной в большинстве интеллектуальных систем. Ниже рассмотрены примеры интеллектуальных компьютерных систем.

1. Компьютерное зрение. Такие системы предназначены для получения изображения через камеру и составление его описания в символьном виде (какие объекты присутствуют, в каком взаимном отношении находятся и т.д.).

2. Символьное распознавание (распознавание букв или цифр).

3. Диагностика в медицине (маммография, рентгенография, диагностирование по истории болезни).

4. Геология.

5. Распознавание речи.

6. Распознавание отпечатков пальцев, распознавание лиц и тд.

Для реализации приложения для распознавания пневмонии по рентгеновским снимкам, которое относится к задаче распознавания образов, может быть применена архитектура сверточных искусственных нейронных сетей.

### 1.3. Обзор аналогичных проектов

**Early Diagnosis of Pneumonia with Deep Learning** [8]

В этом исследовании представляется новый метод классификации пневмонии на рентгеновском снимке. Предлагается двухступенчатая обработка изображения перед обучением сверточной нейронной сети, чтобы выделить особенности рентгеновского снимка для облегчения процесса классификации.

Набор данных был взят с сайта Kaggle и содержит рентгеновские снимки с пневмонией и без. 3 тысячи изображений использовались для тренировки нейронной сети и около 1 тысячи изображений для тестирования. Экспериментальная методология включает в себя три различных метода обработки изображения: увеличение контрастности, расширение цветового пространства и увеличение яркости. На рисунке 1 продемонстрирован пример модификации исходного изображения.

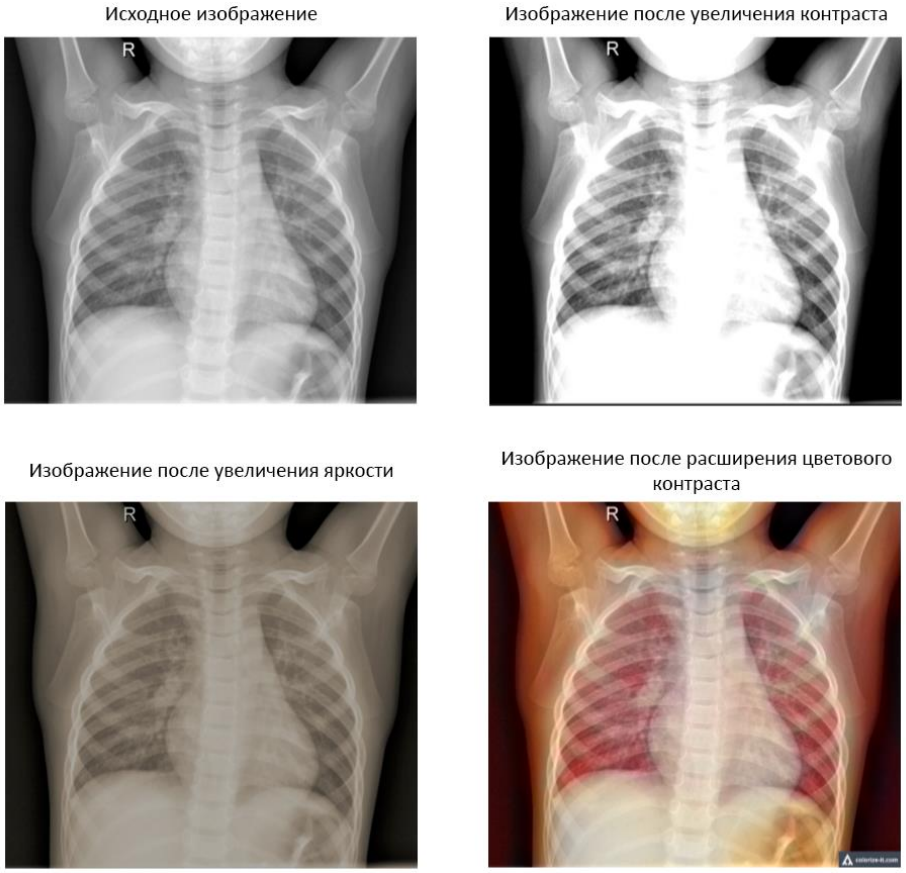


**Рис. 1.** Пример модификаций.

Были разработаны 2 архитектуры нейронной сети. Первая – сверточная нейронная сеть, состоящая из 3 слоев свертки и подвыборки. Так же ученые добавили слой Dropout после функции Softmax для предотвращения переобучения сети. Функция Dropout случайным образом отключает из нейросети определенное количество нейронов. В качестве функции потерь использовалась функция кросс-энтропии. Обучение сети проводилось в течение 120 эпох.

Вторая модель – остаточная нейронная сеть. Она использует остаточный блок, созданный функциями активации. Результат линейной функции суммируется с результатом предыдущей функции активации (рисунок 2). У данной сети 9-слойная архитектура.

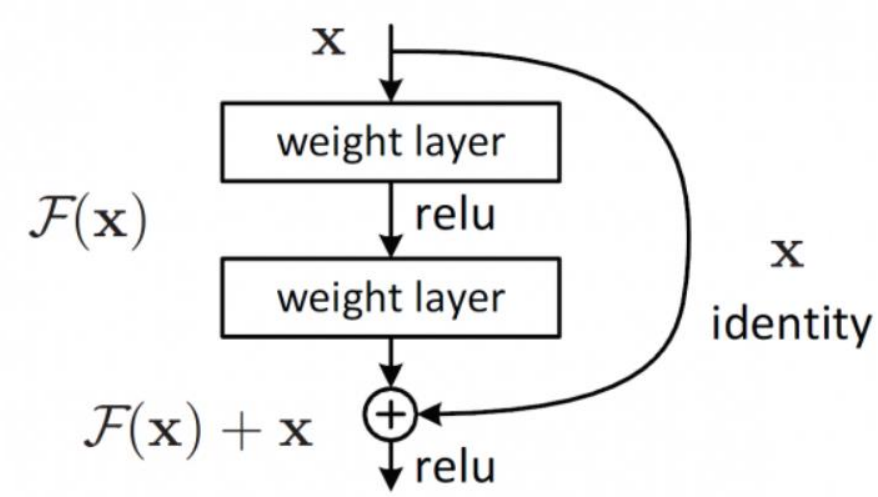


**Рис. 2.** Визуализация остаточной нейронной сети.

Обе сети были обучены и протестированы на модифицированных рентгеновских снимках. Наилучший результат у остаточной нейронной сети, обученной на осветленных изображениях с увеличенным контрастом. Точность такой сети составила 78,73%.

**Deep Learning Approach For Prediction Of Pneumonia** [13]

Эта статья посвящена разработке сверточной нейронной сети, которая может выявить наличие пневмонии с помощью рентгеновского снимка грудной клетки.

Предложенная в статье модель нацелена на достижение максимальной точности при обнаружении пневмонии с использованием остаточных сетей, методов стохастического градиента, косинусным отжигом.

Был использован набор данных Kaggle, состоящий из 5863 рентгеновских снимков детей от 1 до 5 лет. Набор данных был разделен на 3 части: для тренировки, проверки и тестирования.

Для модификации входных изображений была использована Оцу-бинаризация (Otsu Thresholding). Бинаризация применяется для преобразования серой (8-битной) растровой картинки в чёрно-белую (1-битную).

Алгоритм Otsu Thresholding анализирует обрабатываемое изображение и вычисляет единый для всей картинки порог бинаризации. Найден-

ный порог подаётся на вход обыкновенной пороговой бинаризации. Пример обработки изображения приведен на рисунке 3.

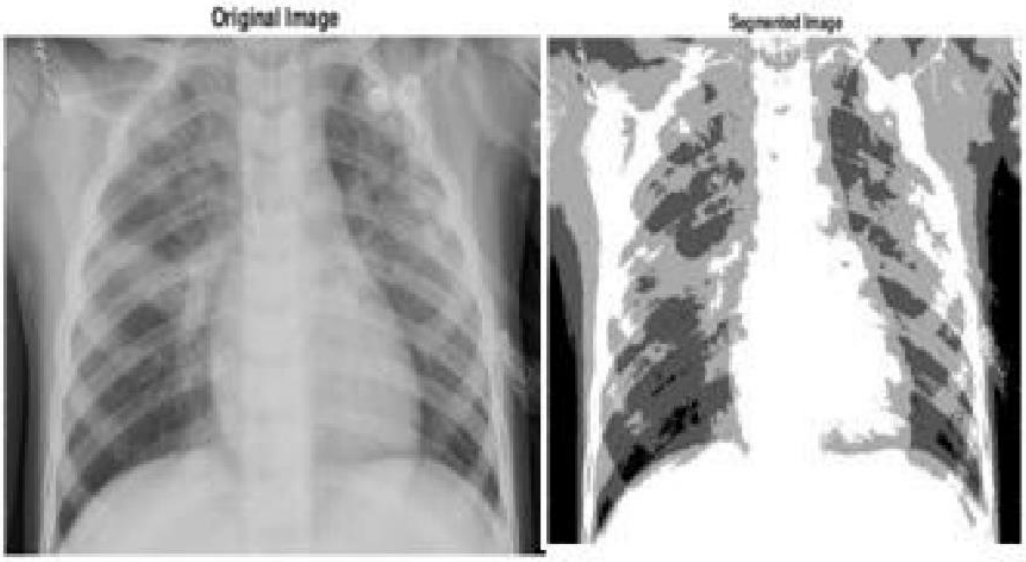


**Рис.3.** Пример обработки изображения

После обучения нейронной чести в течение 500 эпох, было проведено тестовое увеличение времени, и модель была протестирована на работоспособность по тестовым данным. Точность модели составила 92,9%.

**CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning** [12]

В статье описывается нейронная сеть, которая определяет пневмонию по фронтальным рентгеновским снимкам грудной клетки.

В данной работе описан алгоритм, который распознает пневмонию на рентгеновских снимках не хуже, а в некоторых случаях даже лучше, чем практикующие специалисты.

Чтобы разработать этот алгоритм, была построена 121-слойная нейронная сеть и обучена на наборе данный ChestX-ray 14, состоящем из 112120 рентгеновских снимков грудной клетки, полученных от 30805 пациентов. Каждый снимок сопровождался информацией о 14 патологиях легких, в частности, о пневмонии.

Перед обучением снимки уменьшаются до размера 224×224 пикселя, и нормализуется набор данных. Случайным образом выбрали из всех данных 80 процентов фотографий, на которых провели обучение искусственной нейронной сети, а оставшиеся 20 процентов оставили для проверки ее работы. Пример работы нейронной сети представлен на рисунке 4.


**Рис.4.** Пример работы нейронной сети

Затем было проведено сравнение работы сети с результатами диагностики четырех врачей из Стэнфордского Университета, практикующих в течение 4, 7, 25 и 28 лет. Рентгенологи не имели доступа к истории болезни пациентов и могли использовать для диагностики только фронтальные рентгеновские снимки грудной клетки. Оказалось, что нейросеть распознает патологию не хуже, чем практикующие врачи. Точность такой сети для распознавания пневмонии составила 76,8%.

## 1.4. Обзор готовых решений для создания нейронных сетей

Для языка программирования Python существует большое количество готовых библиотек для работы с нейронными сетями. Рассмотрим основные из них.

**Theano** [3]

Фундаментальная библиотека с математическими объектами и нейросетями. Совместима с Python 2 и 3. Одна из самых мощных библиотек, используется для быстрых и высокоточных вычислений.

**TensorFlow** [16]

Библиотека с открытым исходным кодом для машинного обучения, разработанная компанией Google для решения задач построения и тренировки нейронной сети с целью автоматического нахождения и классификации образов. Данная библиотека применяется как для исследований, так и для разработки собственных продуктов Google.

**Keras** [2]

Высокоуровневая библиотека, поддерживающая сети с обратными связями и сети прямого распространения. Использует возможности Theano и TensorFlow в качестве компонентов. Характеризуется минималистичным подходом в дизайне хорошей расширяемостью, позволяет быстро начать работу.

**Выводы по первой главе**

В соответствии с целями данной работы был проведен анализ предметной области, обзор научной литературы и существующих аналогов.

Для реализации программной части данной работы была выбрана библиотека Keras, так как ее функционала достаточно для построения искусственной нейронной сети, к тому же она проста в использовании.

## 2. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

### 2.1.Глубокое обучение

Глубокое обучение – это особый раздел машинного обучения: новый подход к поиску представления данных, делающий упор на изучение последовательных слоев (или уровней) все более значимых представлений [1]. Идея глубокого обучения заключается в многослойном представлении данных. Модель данных делится на слои, количество этих слоев и называют глубиной модели.

Под нейронной сетью в глубоком обучении подразумевается структурированная модель, состоящая из слоев, наложенных друг на друга. Хотя нейронные сети берут свое начало из нейробиологии, тем не менее модели глубокого обучения не являются моделями мозга, так как нет никаких доказательств, что мозг реализует подобные механизмы, используемые в современных моделях глубокого обучения.

Глубокое обучение считается математической основой для изучения представлений данных. На рисунке 5 изображено представление данных, получаемые алгоритмом глубокого обучения для сети, которая преобразует изображение цифры для ее распознавания.

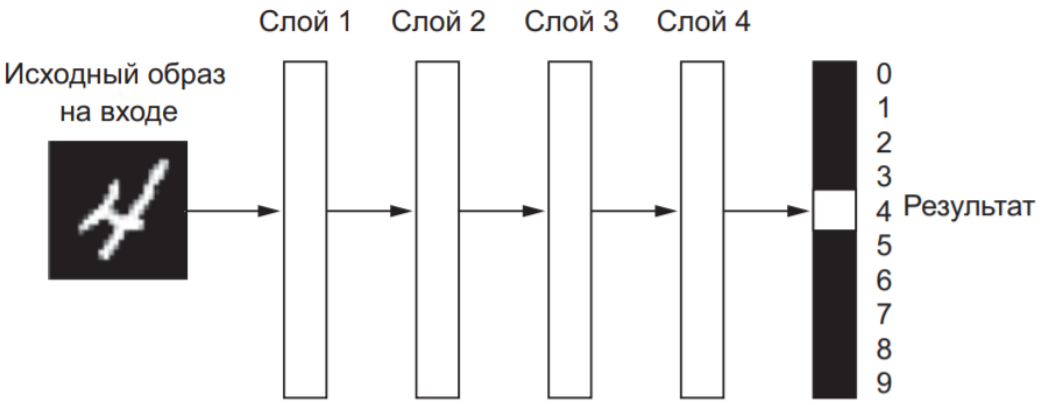


**Рис. 5.** Глубокая нейронная сеть для классификации цифр

Модель состоит из 4 слоев. Сеть поэтапно преобразует входное изображение цифры в представление, все больше отличающееся от исход-

ного образа и несущее все больше полезной информации о результате (рисунок 6).

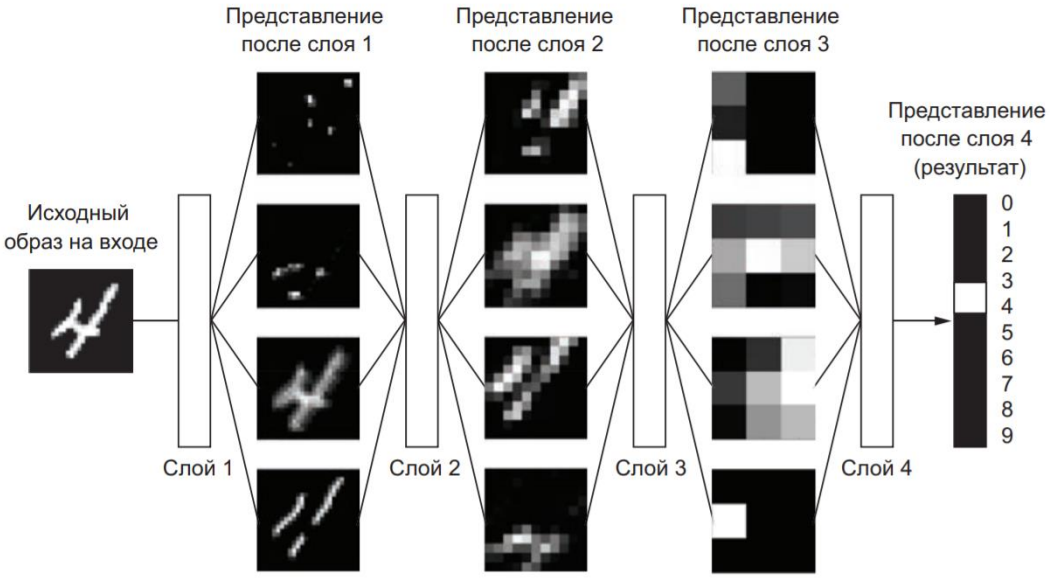


**Рис. 6.** Глубокие представления, получаемые моделью классификации цифр

Глубокую сеть можно рассматривать как многоэтапную операцию очистки информации, в ходе которой информация проходит через последовательность фильтров и выходит из нее в очищенном виде (то есть в виде, пригодном для решения некоторых задач). С технической точки зрения глубокое обучение – это многоступенчатый способ получения представления данных [1].

То, что будет происходить с входными данными в конкретном слое, определяется его весами, которые фактически являются набором чисел (рисунок 7).

В этом контексте обучение – это поиск набора значений весов всех слоев в нейронной сети, при котором сеть будет правильно отображать образцовые входные данные в соответствующие им результаты.

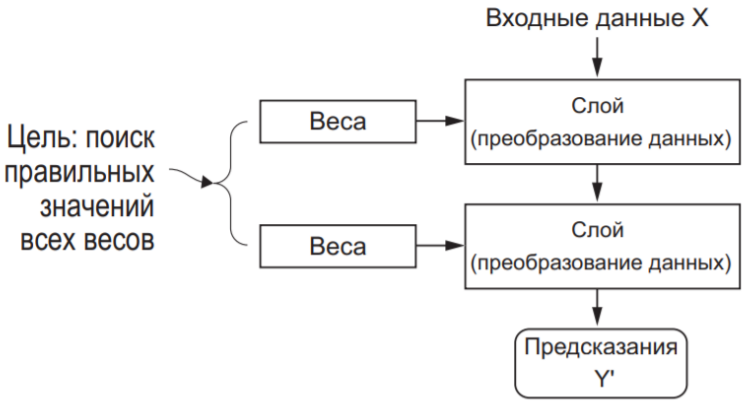


**Рис. 7.** Нейронная сеть параметризуется ее весами

Чтобы управлять результатом работы нейронной сети, нужно иметь возможность оценить, насколько результат отличается от ожидаемого. Функция потерь сети помогает решить эту задачу. Функция потерь на вход принимает предсказание, выданное сетью на определенном этапе обучения, и значение, которое сеть должна была вернуть. Функция потерь вычисляет оценку расстояния между двумя этими значениями (рисунок 8).

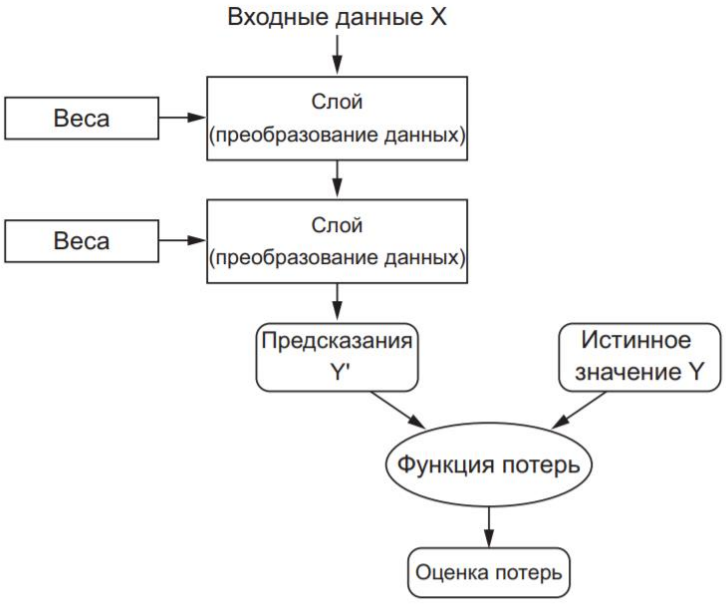


**Рис. 8.** Функция потерь оценивает качество результатов

Основная особенность глубокого обучения заключается в использовании оценки потерь для корректировки значений весов с целью уменьшения потерь (рисунок 9).

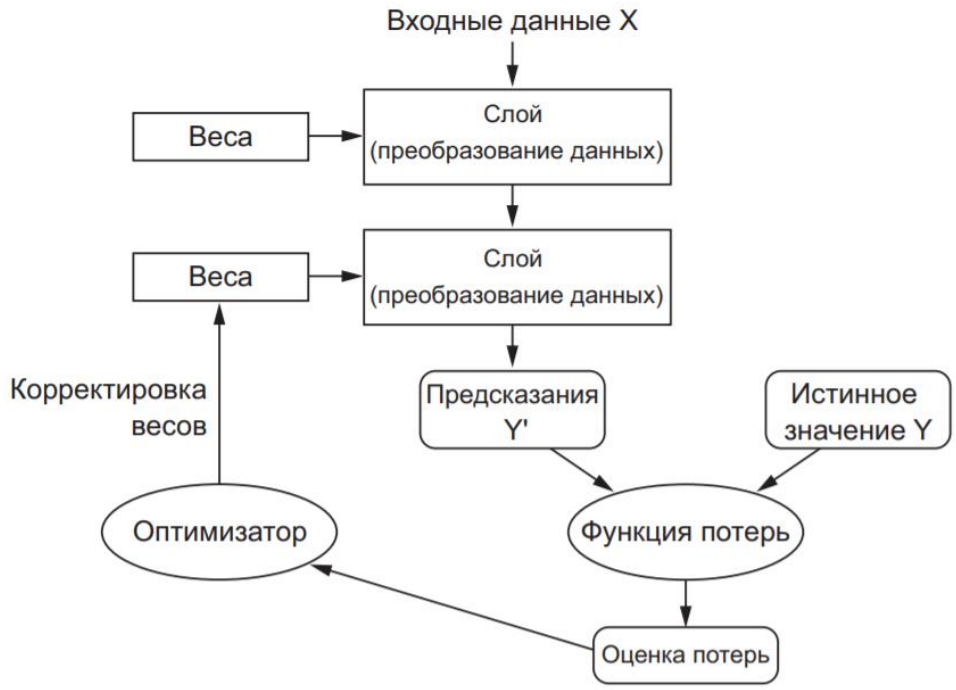


**Рис. 9.** Оценка потерь используется как обратная связь для корректировки весов

Корректировка значений весов – задача оптимизатора, который реализует алгоритм обратного распространения ошибки (основной алгоритм глубокого обучения). Изначально всем весам сети присваиваются случайные значения. В цикле обучения реализуется последовательность случайных преобразований. С каждой итерацией веса корректируются в нужном направлении, и оценка потерь уменьшается. Сеть может считаться обученной, если ее потери минимальны, а результат близок к истинному.

## 2.2. Набор данных

Набор данных для обучения нейронной сети был взят с сайта Kaggle [5]. Он содержит 5863 рентгеновских снимков в формате JPEG. Фронталь-

ные рентгеновские снимки грудной клетки были взяты у пациентов в возрасте от 1 до 5 лет из медицинского центра в Гуанчжоу.

Набор данных состоит из 3 папок (train, test, val) для тренировки, тестирования и проверки нейронной сети. В каждой папке по 2 подпапки Pneumonia и Normal, в которых располагаются рентгеновские снимки с пневмонией и без. На рисунке 10 проиллюстрирован пример изображений из набора данных.

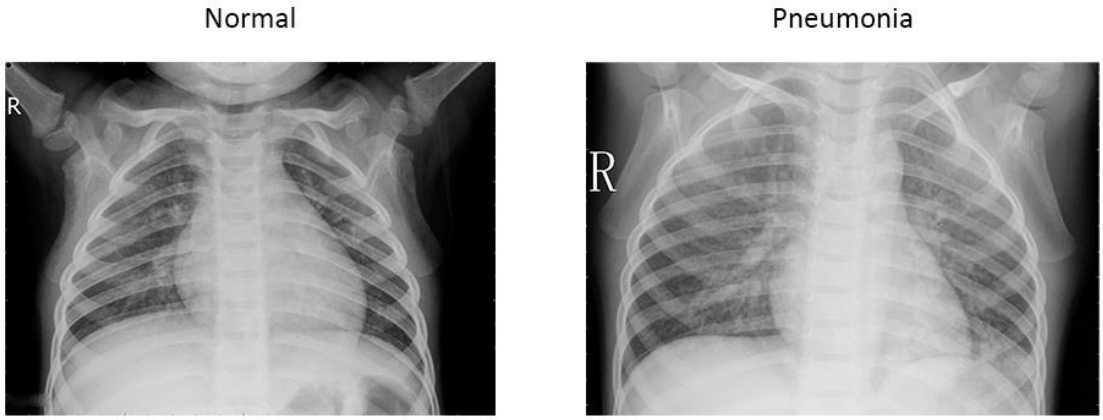


**Рис. 10.** Пример изображений из набора данных

На рентгеновском снимке слева изображены чистые легкие без каких-либо аномалий. Справа представлен снимок легких с пневмонией.

Для достижения лучшего результата работы сети набор данных был сбалансирован. Таким образом, сбалансированный набор данных содержит 3166 рентгеновских снимков, поровну для каждого класса. Для обучения использовалось 80% от выборки, для проверки нейронной сети – 10%, и столько же для тестирования.

**Выводы по второй главе**

В этой главе была рассмотрена теоретическая часть глубокого обучения, а именно основные алгоритмы обучения, принцип построения глубоких нейронных сетей.

Глубокое обучение – это раздел машинного обучения, который очень эффективно можно использовать для классификации изображений, поэтому для распознавания пневмонии по рентгеновским снимкам будем строить глубокую нейронную сеть и обучать ее.

Был выбран набор данных для обучения, тестирования и проверки искусственной нейронной сети.

# 3. ПРОЕКТИРОВАНИЕ

Для решения задачи распознавания пневмонии по рентгеновским снимкам должна быть разработана система, обеспечивающая возможность автоматизированной обработки большого количества изображений.

## 3.1. Требования к системе

Функциональные требования к системе описывают поведение системы. Разрабатываемая система должна удовлетворять следующим функциональным требованиям.

1. Система должна принимать на вход изображения в формате JPEG.

2. Система должна определить есть ли пневмония у пациентов, чьи рентгеновские снимки были загружены.

3. Система должна вывести на экран результат распознавания.

К нефункциональным требованиям системы относятся свойства, которыми она должна обладать. Например, удобство использования, безопасность, расширяемость и т.д.

Система должна удовлетворять следующим нефункциональным требованиям.

1. Система должна быть реализована на языке Python 3.7.

2. Система должна использовать библиотеку Keras для создания глубокой нейронной сети.

3. Приложение должно быть целостным и не требовать от пользователя дополнительных преобразования изображений.

## 3.2. Варианты использования приложения

Для проектирования системы был использован унифицированный язык моделирования UML. Была построена модель взаимодействия актера с приложением распознавания пневмонии. Диаграмма вариантов использования представлена на рисунке 11.

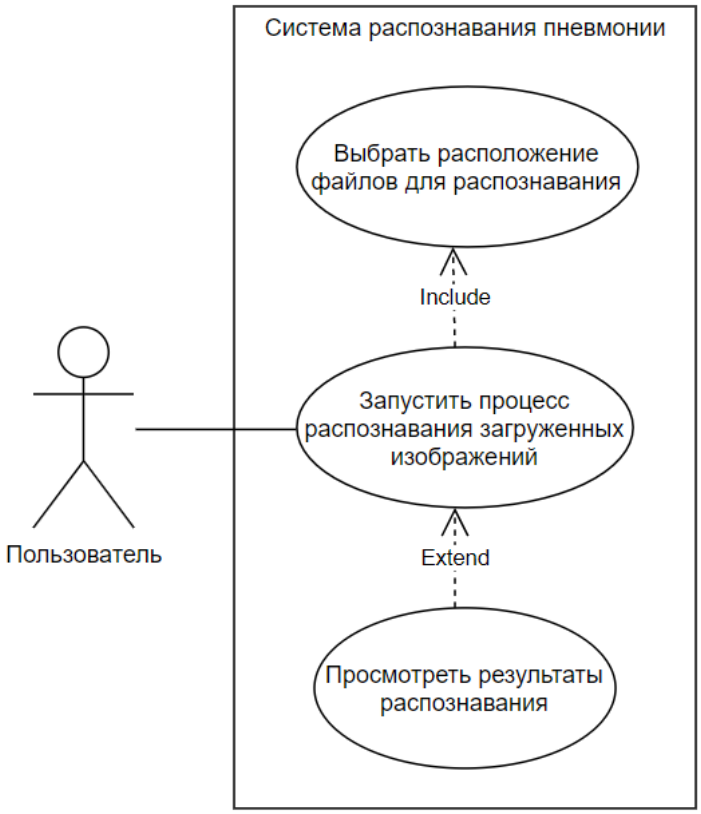


**Рис. 11.** Диаграмма вариантов использования

**Актеры, взаимодействующие с системой:**

Пользователь – человек, использующий систему и ее функционал.

**Краткое описание вариантов использования:**

Пользователю доступны следующие функции:

1) выбрать расположение файлов для распознавания;

2) запустить процесс распознавания загруженных изображений;

3) просмотреть результаты распознавания.

На основе функциональных и нефункциональных требований была разработана диаграмма деятельности (рисунок 12).

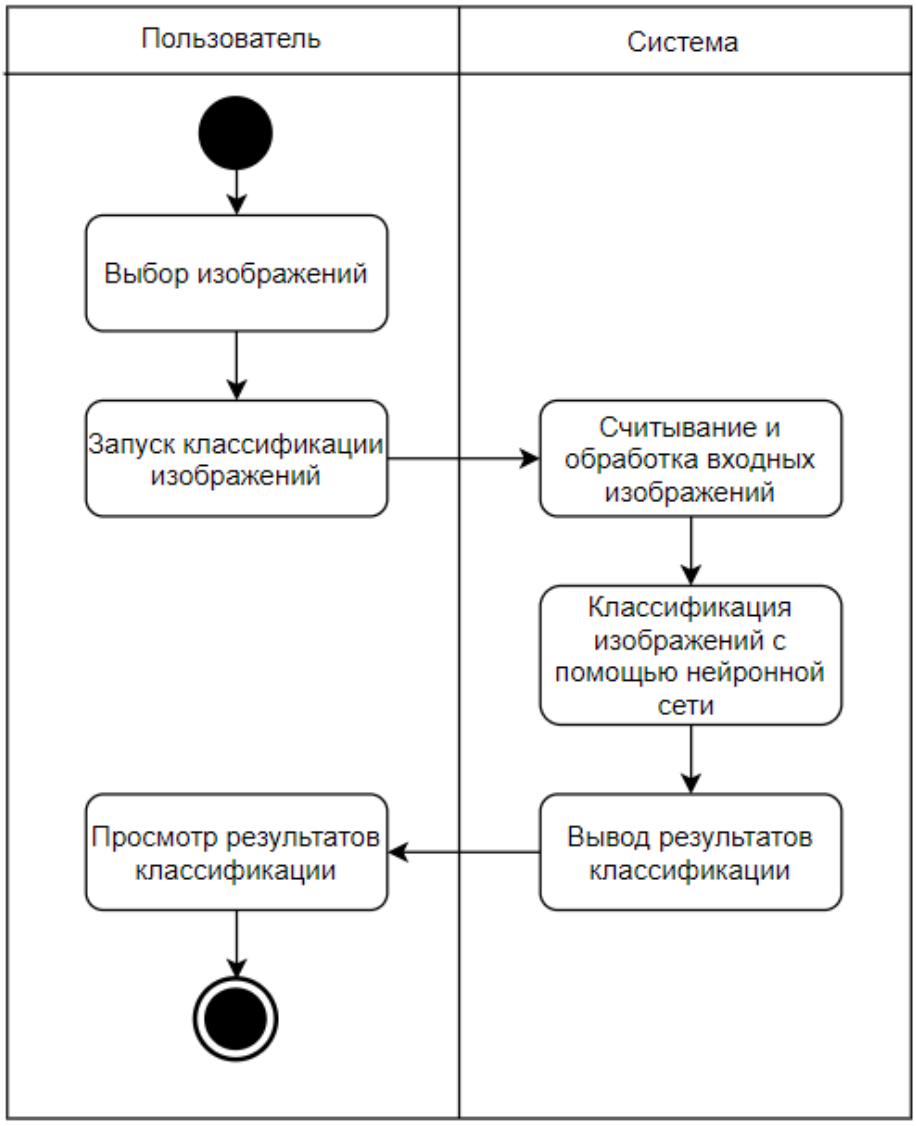


**Рис. 12.** Диаграмма деятельности

На данной диаграмме изображен процесс взаимодействия пользователя с системой распознавания пневмонии.

### 3.3. Проектирование нейронной сети

Для решения поставленной задачи была разработана топология сверточной нейронной сети, структура которой представлена на рисунке 13.

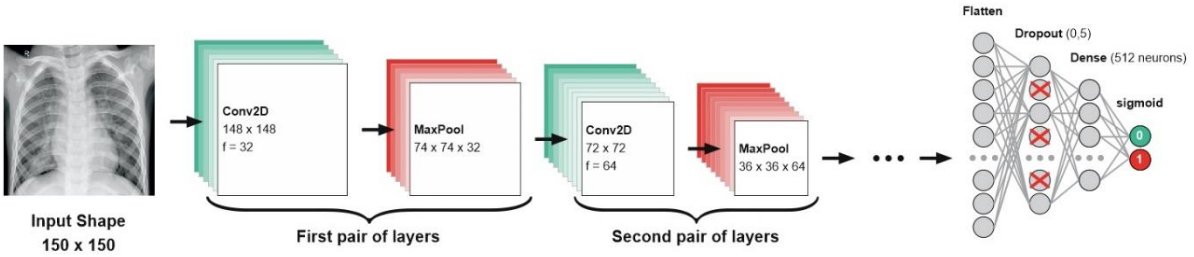


**Рис. 13.** Топология нейронной сети

Нейронная сеть состоит из 4 пар чередующихся сверточных слоев и слоев подвыборки, а также из 1 слоя прореживания и 2 полносвязных слоев. Данная топология была выбрана из-за наибольшей точности сети, выявленной экспериментальным путем.

### 3.4. Проектирование интерфейса приложения

Графический интерфейс приложения должен быть минималистичным и интуитивно понятным для пользователя. Разработанный макет интерфейса приведен на рисунке 14.

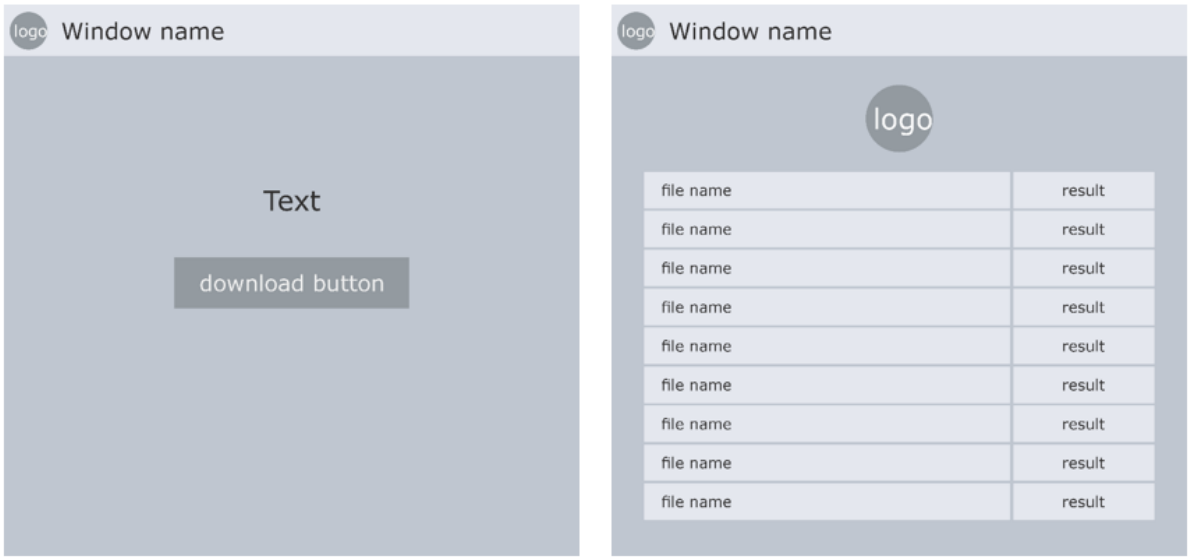


**Рис. 14.** Макет пользовательского интерфейса

**Выводы по третьей главе**

В третьей главе были определены функциональные и нефункциональные требования для системы, а также представлены диаграммы вариантов использования и деятельности. В свою очередь была спроектирована топология нейронной сети и сама система распознавания пневмонии по рентгеновским снимкам, реализация которой представлена в следующей главе.

# 4. РЕАЛИЗАЦИЯ

## 4.1. Программные средства реализации

Для реализации программной части системы был выбран язык программирования Python 3.7 [8]. Разработка велась в среде JetBrains PyCharm Community Edition 2019.2.2 [11].

Для языка Python были использованы следующие библиотеки: Keras 2.3.1 [2], Numpy 1.16.4 [10]. Также в работе была использована библиотека PyQt5 [4] для реализации интерфейса.

## 4.2. Реализация нейронной сети

В ходе экспериментов было обучено 10 моделей с различной архитектурой и параметрами. В таблице 1 представлены результаты проведенных экспериментов. Лучший результат работы показали модели с оптимизатором rmsprop и adam. Ниже подробно описана архитектура нейронной сети, которая показала наибольшую точность в результате экспериментов.

**Табл. 1.** Результаты экспериментов

| № | Количество пар слоев свертки и подвыборки | Эпохи | Параметр для слоя прореживания | Оптимизатор | Точность |
|---|---|---|---|---|---|
| 1 | 4 | 15 | 0,5 | rmsprop | 70,25 % |
| 2 | 4 | 15 | 0,4 | rmsprop | 86,08 % |
| 3 | 4 | 20 | 0,5 | rmsprop | 89,24 % |
| 4 | 4 | 27 | 0,5 | rmsprop | 80,38 % |
| 5 | 5 | 20 | 0,4 | adam | 76,58 % |
| 6 | 3 | 20 | 0,3 | adam | 50,95 % |
| 7 | 4 | 22 | 0,5 | adam | 83,86 % |
| 8 | 3 | 15 | 0,3 | SDG | 50,95 % |
| 9 | 4 | 15 | 0,5 | SDG | 67,40 % |
| 10 | 4 | 25 | 0,5 | SDG | 54,40 % |

Сверточная нейронная сеть организована как стек чередующихся слоев Conv2D (с функцией активации relu) и MaxPooling2D. Сеть принимает на вход изображение размером 150x150 пикселей.

Так как классификация бинарная, сеть заканчивается единственным признаком (слой Dense с размером 1 и функцией активации sigmoid). Этот признак будет представлять собой принадлежность рассматриваемого изображения одному из двух классов (пневмония, отсутствие пневмонии). Также в модель нейронной сети был добавлен слой Dropout с параметром 0,5 непосредственно перед полносвязным классификатором. Данная модель использует оптимизатор rmsprop. В листинге 1 представлена реализация сверточной нейронной сети.

**Листинг 1.** Реализация нейронной сети

```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(150, 150, 3))) #слой свертки
model.add(layers.MaxPooling2D((2, 2))) #слой подвыборки
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten()) #преобразования из двумерного вида в плоский
model.add(layers.Dropout(0.5)) #слой прореживания
model.add(layers.Dense(512, activation='relu')) #полносвязанный слой
model.add(layers.Dense(1, activation='sigmoid')) #выходной
model.compile(loss='binary_crossentropy', optimizer='rmsprop', metrics=['acc'])
```

### 4.3. Расширение данных

Для предотвращения переобучения используется прием расширения данных. Создаются дополнительные обучающие данные из имеющихся путем трансформации образцов множеством случайных преобразований. Цель состоит в том, чтобы на этапе обучения модель никогда не увидела одно и то же изображение дважды. Это поможет модели выявить больше особенностей данных.

Применяемые трансформация для тренировочных данных представлены в листинге 2.

**Листинг 2**. Применяемая к изображениям трансформация

```
train_datagen = ImageDataGenerator(rescale=1./255, rotation_range=10, width_shift_range=0.1, height_shift_range=0.1, shear_range=0.1, zoom_range=0.2)
```

Применяемые трансформации к исходным данным.

1. Случайный поворот изображения (rotation_range).

2. Смещение изображения по вертикали и горизонтали (width_shift, height_shift).

3. Сдвиговое преобразование (shear_range).

4. Изменение масштаба изображения (zoom_range).

### 4.4. Обучение нейронной сети

Обучение модели проводилось на компьютере, имеющем следующие характеристики:

1) видеокарта NVIDIA GeForce 940M;

2) процессор Intel(R) Core (TM) i5-6200 CPU @ 2,30 GHz 2,40 GHz);

3) 6 ГБ оперативной памяти;

4) ОС Windows 10.

Для обучения и проверки точности сети был использован набор данных с рентгеновскими снимками легких, скачанный с официального сайта Kaggle [5]. Длительность обучения составила 15 эпох, по 100 итераций каждая. В среднем на одну эпоху тратилось 1,5 минуты. Суммарное обучение заняло порядка 25 минут. Обученная сверточная нейронная сеть была сохранена.

### 4.5. Разработка приложения

Для реализации пользовательского интерфейса была выбрана библиотека PyGt5. Интерфейс готового приложения представлен на рисунке 15.

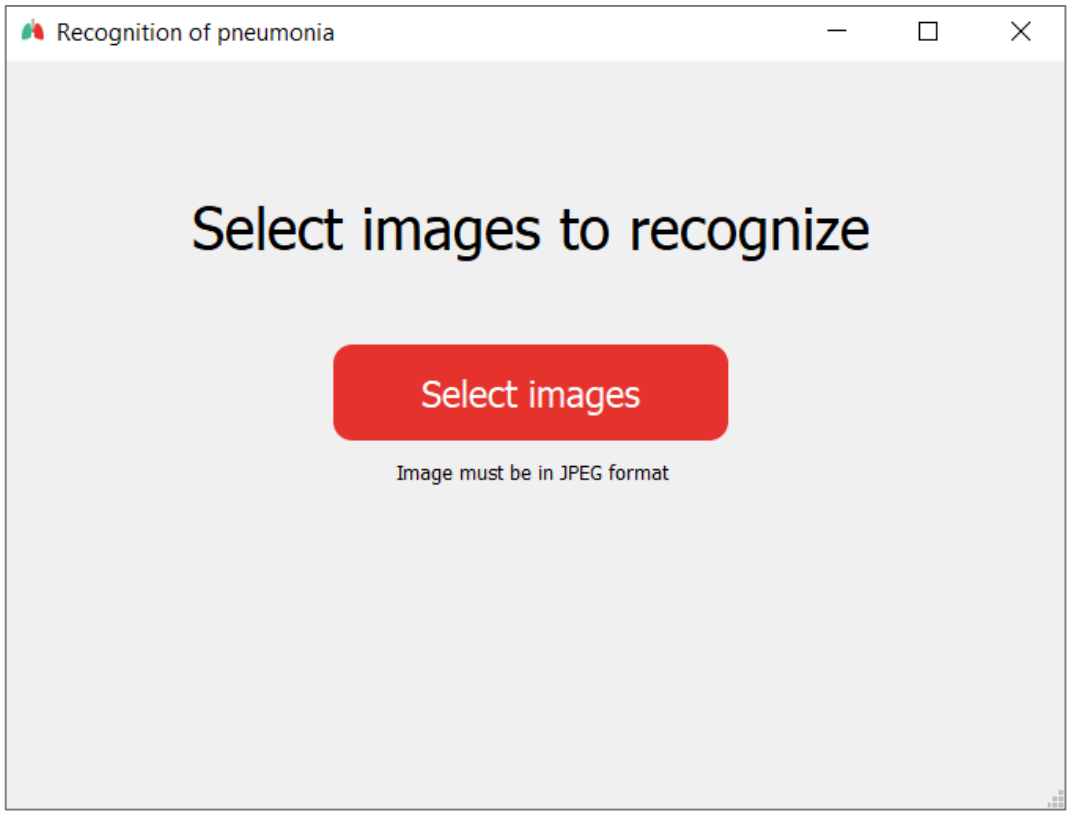


**Рис. 15.** Интерфейс приложения

На главном окне приложения располагается кнопка «Select images» для загрузки изображений, при нажатии на которую открывается стандартное окно выбора файлов. После выбора изображений сразу запускается процесс распознавания. Программный код данного модуля (main.py) представлен в приложении А.

Сначала загружается созданная ранее нейронная сеть. Потом происходит предобработка изображения, приведение к размеру 150x150. Далее каждое загруженное изображение прогоняется через обученную нейронную сеть. В конце распознавания открывается окно с результатом в виде таблицы (рисунок 16).

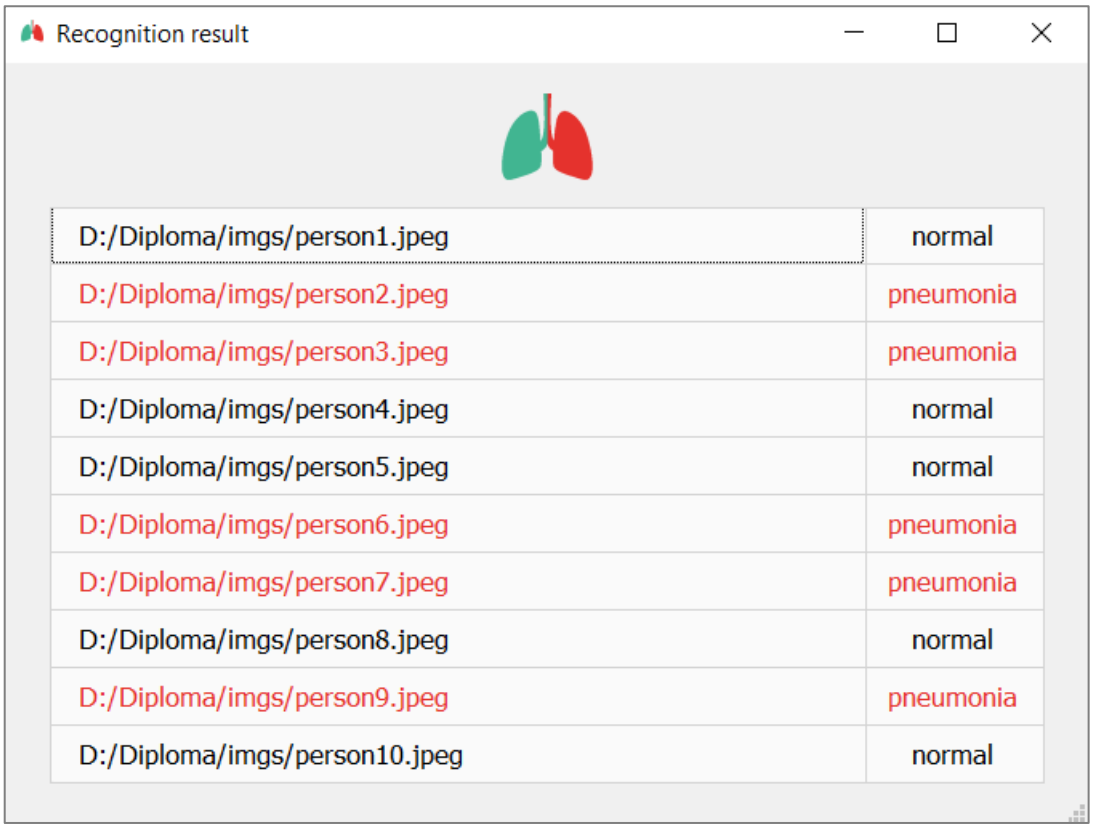


**Рис. 16.** Вывод результатов распознавания

**Выводы по четвертой главе**

В данной главе были описаны используемые средства разработки, а также реализация нейронной сети и программный код самого приложения. Разработанное приложение полностью соответствует ранее упомянутым требованиям.

# 5. ТЕСТИРОВАНИЕ

## 5.1. Тестирование нейронной сети

Для тестирования нейронной сети было использовано 10% от набора данных (по 158 изображений рентгеновских снимков для каждого класса). Для тестирования искусственной нейронной сети был реализован программный модуль test.py. Обученная нейронная сеть показала точность 89,24 %.

## 5.2. Функциональное тестирование приложения

Было проведено функциональное тестирование системы согласно описанным ранее требованиям. Результаты тестирования приведены в таблице 2.

**Табл. 2.** Результаты функционального тестирования

| № | Предусловие | Действие | Ожидаемый результат | Результат теста |
|---|---|---|---|---|
| 1 | Проверка запуска приложения | Пользователь запускает приложение | На компьютере пользователя запускается консольное приложение | Пройден |
| 2 | Приложение запущено | Пользователь нажимает кнопку «Select images» для выбора директории | Открывается окно выбора изображений | Пройден |
| 3 | Приложение запущено, изображения для распознавания выбраны | Пользователь загружает изображения | В дополнительном окне выводится результат распознавания | Пройден |
| 4 | Приложение запущено, открыто окно выбора изображений | Пользователь выбирает файл | Пользователь может видеть и выбирать только изображения в формате JPEG | Пройден |

**Выводы по пятой главе**

В данной главе было произведено тестирование работы нейронной сети, а также функциональное тестирование системы. Все подготовленные тесты были успешно пройдены.

## ЗАКЛЮЧЕНИЕ

В рамках дипломной работы была создана и обучена сверточная нейронная сеть для классификации рентгеновских снимков, а также было реализовано приложение для распознавания пневмонии по рентгеновским снимкам на основе нейросетевых технологий.

В ходе разработки были решены задачи, перечисленные ниже.

1. Сделать обзор аналогов и научной литературы.

2. Спроектировать топологию искусственной нейронной сети и обучить ее.

3. Разработать приложение для классификации рентгеновских снимков.

4. Провести тестирование системы.

В будущем планируется добавить возможность классификации пневмонии на вирусную и бактериальную, а также добавить возможность экспорта результатов распознавания в сторонний файл, например в таблицу.

**ЛИТЕРАТУРА**

1. Шолле Франсуа. Глубокое обучение на Python. — СПб.: Питер, 2018. — 400 с.: ил. — (Серия «Библиотека программиста»).

2. Англоязычный сайт с документацией библеотеки Keras языка Python [Электронный ресурс]: – Режим доступа: https://keras.io/ (дата обращения: 27.12.2019).

3. Репозиторий библеотеки Theano языка Python [Электронный ресурс]: – Режим доступа: https://github.com/Theano/ (дата обращения: 21.01.2019).

4. Официальный сайт PyQt5 [Электронный ресурс]: – Режим доступа: https://riverbankcomputing.com/software/pyqt/intro (дата обращения: 20.02.2020).

5. Англоязычный сайт Kaggle [Электронный ресурс]: – Режим доступа: https://www.kaggle.com/ (дата обращения: 11.01.2019).

6. Ростовцев В. С. Искусственные нейронные сети : учебник / В. С. Ростовцев. — Санкт-Петербург : Лань, 2019. — 216с.

7. Душкин Р. В. Исусственный интеллект. - М.: ДМК Пресс, 2019. - 280 с.

8. Deniz Yagmur Urey, Can Jozef Saul, and Can Doruk Taktakoglu. Early Diagnosis of Pneumonia with Deep Learning – Robert College of Istanbul, Istanbul, Turkey, Koc University Artificial Intelligence Laboratory, Istanbul, Turkey, 2019. – 7 p.

9. Англоязычный сайт Python [Электронный ресурс]: – Режим доступа: https://www.python.org/ (дата обращения: 20.12.2019).

10. Официальный сайт библеотеки Numpy языка Python [Электронный ресурс]: – Режим доступа: https://numpy.org/ (дата обращения: 11.01.2019).

11. Официальный сайт PyCharm [Электронный ресурс]: – Режим доступа: https://www.jetbrains.com/ru-ru/pycharm/ (дата обращения: 20.12.2019).

12. Pranav Rajpurkar, Jeremy Irvin, Kaylie Zhu, Brandon Yang, Hershel Mehta, Tony Duan, Daisy Ding, Aarti Bagul, Robyn L. Ball, Curtis Langlotz, Katie Shpanskaya, Matthew P. Lungren, Andrew Y. Ng. CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning, 2017. – 7 p.

13. Kalyani Kadam, Dr.Swati Ahirrao, Harbir Kaur ,Dr. Shraddha Phansalkar , Dr. Ambika Pawar. Deep Learning Approach For Prediction Of Pneumonia, 2019. – 4 p.

14. Лепский А.Е., Броневич А.Г. Математические методы распознавания образов: Курс лекций. – Таганрог: Изд-во ТТИ ЮФУ, 2009. – 155 с.

15. Местецкий Леонид Моисеевич. Математические методы распознавания образов: Курс лекций. – МГУ, ВМиК, 2002–2004. – 85 с.

16. Официальный сайт библеотеки TensorFlow языка Python [Электронный ресурс]: – Режим доступа: https://www.tensorflow.org/ (дата обращения: 21.01.2019).

# ПРИЛОЖЕНИЕ

# ПРИЛОЖЕНИЕ А – Листинги кода

**Листинг А.1** – Код программного модуля main.py

```
From PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtWidgets import QTableWidgetItem
import sys
from keras.preprocessing import image
import numpy as np
from keras import models

def loadad_net():
    model = models.load_model('neural_net.h5')
    model.compile(loss='binary_crossentropy', optimizer='rmsprop', met-
rics=['acc'])
    return model

def recognition_img(model ,path):
    img_width, img_height = 150, 150
    img = image.load_img(path, target_size=(img_width, img_height))
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis=0)
    images = np.vstack([x])
    classes = model.predict_classes(images, batch_size=10)
    return int(str(classes[0][0]))

def recognition_pneumonia(model, imgs):
    dict = {}
    for img in imgs:
        dict.setdefault(img, recognition_img(model, img))
    return dict

class Ui_mainWindow(object):
    def setupUi(self, mainWindow):
        mainWindow.setObjectName("mainWindow")
        mainWindow.resize(677, 476)
        icon = QtGui.QIcon()
        icon.addPixmap(QtGui.QPixmap("ico.png"), QtGui.QIcon.Normal,
QtGui.QIcon.Off)
        mainWindow.setWindowIcon(icon)
        mainWindow.setIconSize(QtCore.QSize(10, 10))
        self.centralwidget = QtWidgets.QWidget(mainWindow)
```

```
        self.centralwidget.setObjectName("centralwidget")
        self.selectButton = QtWidgets.QPushButton(self.centralwidget)
        self.selectButton.setGeometry(QtCore.QRect(210, 180, 251, 61))
        self.selectButton.setStyleSheet("QPushButton{background-color:
#E5322D;\n"
"color: #fff;\n"
"font: 14pt \"MS Shell Dlg 2\";\n"
"border-radius: 12px;\n"
"}\n"
"QPushButton:hover {\n"
"background-color: #43CCA1 ;\n"
"}")
        self.selectButton.setObjectName("selectButton")

        self.label = QtWidgets.QLabel(self.centralwidget)
        self.label.setGeometry(QtCore.QRect(120, 80, 441, 51))
        self.label.setStyleSheet("font: 75 22pt \"MS Shell Dlg 2\";")
        self.label.setObjectName("label")
        self.label_2 = QtWidgets.QLabel(self.centralwidget)
        self.label_2.setGeometry(QtCore.QRect(250, 250, 181, 21))
        self.label_2.setObjectName("label_2")
        mainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(mainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 677, 26))
        self.menubar.setObjectName("menubar")
        mainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(mainWindow)
        self.statusbar.setObjectName("statusbar")
        mainWindow.setStatusBar(self.statusbar)
        self.retranslateUi(mainWindow)
        QtCore.QMetaObject.connectSlotsByName(mainWindow)

    def retranslateUi(self, mainWindow):
        _translate = QtCore.QCoreApplication.translate
        mainWindow.setWindowTitle(_translate("mainWindow", "Recognition of
pneumonia"))
        self.selectButton.setText(_translate("mainWindow", "Select imag-
es"))
        self.selectButton.clicked.connect(self.select)
        self.label.setText(_translate("mainWindow", "Select images to rec-
ognize"))
        self.label_2.setText(_translate("mainWindow", "Image mast be in
```

```
JPEG format"))

    def select(self):
        imgs = QtWidgets.QFileDialog.getOpenFileNames(None, "Open Image",
'.', "(*.jpg *.jpeg)")[0]
        net = loadad_net()
        dict = recognition_pneumonia(net, imgs)
        ui = Ui_Win()
        ui.setupUi(mainWindow, dict)
        mainWindow.show()

class Ui_Win(object):
    def setupUi(self, MainWindow, dict):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(679, 476)
        icon = QtGui.QIcon()
        icon.addPixmap(QtGui.QPixmap("ico.png"), QtGui.QIcon.Normal,
QtGui.QIcon.Off)
        MainWindow.setWindowIcon(icon)
        MainWindow.setStyleSheet("")
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.logo = QtWidgets.QLabel(self.centralwidget)
        self.logo.setGeometry(QtCore.QRect(300, 10, 81, 81))
        self.logo.setText("")
        self.logo.setPixmap(QtGui.QPixmap("ico.png"))
        self.logo.setScaledContents(True)
        self.logo.setObjectName("logo")
        self.tableWidget = QtWidgets.QTableWidget(self.centralwidget)
        self.tableWidget.setGeometry(QtCore.QRect(30, 90, 621, 360))
        self.tableWidget.setStyleSheet("border-color: rgb(115, 115, 115);
background-color: rgb(250, 250, 250); border: none; font: 75 10pt 'MS Shell
Dlg 2'; border-radius: 5px;")
        self.tableWidget.setAutoScroll(True)

self.tableWidget.setHorizontalScrollMode(QtWidgets.QAbstractItemView.Scroll
PerItem)
        self.tableWidget.setRowCount(len(dict))
        self.tableWidget.setColumnCount(2)
        self.tableWidget.setObjectName("tableWidget")
        self.tableWidget.horizontalHeader().setVisible(False)
```

```
self.tableWidget.horizontalHeader().setCascadingSectionResizes(True)
        self.tableWidget.horizontalHeader().setDefaultSectionSize(90)
        self.tableWidget.horizontalHeader().resizeSection(0, 510)
        self.tableWidget.horizontalHeader().resizeSection(1, 111)
        self.tableWidget.horizontalHeader().setSortIndicatorShown(False)
        self.tableWidget.horizontalHeader().setStretchLastSection(False)
        self.tableWidget.verticalHeader().setVisible(False)
        self.tableWidget.verticalHeader().setDefaultSectionSize(36)
        self.tableWidget.verticalHeader().setSortIndicatorShown(False)

self.tableWidget.setVerticalScrollBarPolicy(QtCore.Qt.ScrollBarAlwaysOff)
        MainWindow.setCentralWidget(self.centralwidget)
        self.fill_table(dict)
        self.menubar = QtWidgets.QMenuBar(MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 679, 26))
        self.menubar.setObjectName("menubar")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)
        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def fill_table(self, dict):
        row = 0
        for key in dict:
            if dict[key] is 0:
                self.tableWidget.setItem(row, 0, QTableWidgetItem("   " +
key))
                self.tableWidget.setItem(row, 1, QTableWidgetItem("
normal"))
                row += 1
            else:
                item = QTableWidgetItem("   " + key)
                item.setForeground(QtGui.QColor("#E5322D"))
                self.tableWidget.setItem(row, 0, item)
                item = QTableWidgetItem("  pneumonia")
                item.setForeground(QtGui.QColor("#E5322D"))
                self.tableWidget.setItem(row, 1, item)
                row += 1

    def retranslateUi(self, MainWindow):
```

```
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "Recognition re-
sult"))

if __name__ == "__main__":
    app = QtWidgets.QApplication(sys.argv)
    mainWindow = QtWidgets.QMainWindow()
    ui = Ui_mainWindow()
    ui.setupUi(mainWindow)
    mainWindow.show()
    sys.exit(app.exec_())
```