**Министерство науки и высшего образования Российской Федерации**
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**
**(ФГБОУ ВО «АмГУ»)**

Институт компьютерных и инженерных наук
Кафедра математического анализа и моделирования
Направление подготовки: 01.04.02 – «Прикладная математика и информатика»
Направленность (профиль) образовательной программы – «Математическое и программное обеспечение информационных систем»

УТВЕРЖДАЮ
И.о. зав. кафедрой
_____Н.Н. Максимова
«_____»_____2025 г.

# МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

на тему: «Разработка и обучение нейронной сети для анализа рентген-снимков лёгких с созданием чат-бота для интерактивного взаимодействия»

Исполнитель
студент группы 3101ом _____ (подпись, дата) М.О. Дудич

Научный руководитель
доцент, канд. физ.-мат. наук _____ (подпись, дата) Н.Н. Максимова

Руководитель научного
содержания программы
магистратуры
доцент, канд. физ.-мат. наук _____ (подпись, дата) Н.Н. Максимова

Нормоконтроль
ассистент _____ (подпись, дата) В.О. Салмиянов

Рецензент
доцент, канд. физ.-мат. наук _____ (подпись, дата) Д.В. Фомин

Благовещенск 2025

**Министерство науки и высшего образования Российской Федерации**
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**
**(ФГБОУ ВО «АмГУ»)**

Институт компьютерных и инженерных наук
Кафедра математического анализа и моделирования

УТВЕРЖДАЮ
И.о. зав. кафедрой
_____Н.Н. Максимова
«_____»_____2025 г.

# З А Д А Н И Е

К магистерской диссертации студента Михаила Олеговича Дудич

1. Тема магистерской диссертации: Разработка и обучение нейронной сети для анализа рентген-снимков лёгких с созданием чат-бота для интерактивного взаимодействия.

(утверждены приказом от 24.01.2025 г. № 162-уч)

2. Срок сдачи студентом законченной проекта: 24 июня 2025 года.

3. Исходные данные к магистерской диссертации: набор данных «X-ray Lung Diseases Images» библиотеки машинного обучения Tensorflow, Keras, Matplotlib языка Python; Google Colab – среда разработки, используемая для написания кода и анализа данных; методы глубокого обучения – свёрточные нейронные сети (CNN) для построения модели классификации изображений; отчеты про производственной практике (научно-исследовательской работе) за 1-3 семестры, отчет по преддипломной практике.

4. Содержание выпускной квалификационной работы (перечень подлежащих разработке вопросов): обзор методов искусственного интеллекта, их развития и областей применения, анализ задач, решаемых с помощью искусственного интеллекта, с акцентом на сферу медицины; нейронная сеть для диагностики заболеваний легких.

5. Перечень материалов приложения (наличие чертежей, таблиц, графиков, схем,

программных продуктов, иллюстративного материала и т.п.): листинги программного кода архитектуры нейронной сети; таблицы, отображающие сведения о наборе данных, а также графики результатов обучения модели.

6. Консультанты по магистерской диссертации: рецензент – Фомин Д.В., доцент кафедры физики, заместитель директора (по науке и инновациям) Института компьютерных и инженерных наук, директор Научно-образовательного центра им. К.Э. Циолковского, ФГБОУ ВО «АмГУ», канд. физ.-мат. наук, доцент; нормоконтроль – Салмиянов В.О., ассистент.

7. Дата выдачи задания: 03.02.2025 г.

Руководитель магистерской диссертации: Максимова Надежда Николаевна, доцент, канд. физ.-мат. наук, доцент

Задание принял к исполнению (03.02.2025): _____ М.О. Дудич

# РЕФЕРАТ

Магистерская диссертация содержит 77 с., 19 рисунков, 1 таблицу, 3 приложения, 25 источников.

ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ, НЕЙРОННЫЕ СЕТИ, СВЁРТОЧНЫЕ НЕЙРОННЫЕ СЕТИ, МЕТРИКИ КАЧЕСТВА, PYTHON, МЕДИЦИНСКИЕ ИЗОБРАЖЕНИЯ, KERAS, GRAD-CAM

В работе исследуется задача классификации патологий лёгких по изображениям рентгеновских снимков с помощью методов глубокого обучения, в частности свёрточных нейронных сетей.

Цель магистерской диссертации – изучение и реализация моделей глубокого обучения (свёрточные нейронные сети), построение архитектуры, обучение модели; анализ и интерпретация результатов и их применимости в медицинской сфере.

Для достижения цели был поставлен ряд задач:

- изучение научной и методической литературы по искусственному интеллекту, в частности анализу изображений с помощью нейронных сетей;
- изучение современного состояния искусственного интеллекта;
- описание и подготовка данных из набора «X-ray Lung Diseases Images»;
- построение конвейера для автоматической обработки изображений для подачи на вход нейронной сети;
- подбор и построение архитектуры модели, подбор гиперпараметров обучения;
- анализ полученных результатов по метрикам качества, оценка качества модели
- создание тепловых карт для интерпретации работы свёрточной нейронной сети
- создание интерфейса для взаимодействия с обученной моделью

Основу методологии исследований составляют методы глубокого обучения, в частности свёрточные нейронные сети, реализованные с использованием языка программирования Python и его библиотек: Tensorflow, Keras. В ходе работы была проведена обработка изображений, построена архитектура модели, проведено обучение модели; проведён анализ результатов и оценка модели; создан простой интерфейс для взаимодействия с моделью.

СОДЕРЖАНИЕ

# ВВЕДЕНИЕ

Искусственный интеллект (ИИ) – это одна из наиболее быстро развивающихся областей науки и технологий, охватывающая широкий спектр задач, включая анализ данных, автоматизацию процессов, машинное обучение и нейронные сети. В последние десятилетия, благодаря значительному увеличению вычислительных мощностей и доступности больших объемов данных, методы искусственного интеллекта стали основой для множества инновационных решений в различных сферах, таких как медицина, промышленность, транспорт, финансы и безопасность.

Одним из ключевых направлений развития искусственного интеллекта является компьютерное зрение – область, связанная с анализом и интерпретацией визуальной информации. В этом контексте сверточные нейронные сети (CNN, Convolutional Neural Networks) зарекомендовали себя как мощный инструмент для решения задач классификации изображений, детекции объектов, сегментации и генерации изображений. CNN применяются в самых разных областях: от распознавания лиц и автономного вождения до диагностики заболеваний по медицинским снимкам.

В последние годы наблюдается значительный рост числа проектов, связанных с машинным обучением и нейронными сетями. Большая часть этих разработок ведется на языке программирования Python, который благодаря широкому набору специализированных библиотек (таких как NumPy, Pandas, TensorFlow, PyTorch, Keras, OpenCV) предоставляет удобные инструменты для работы с данными и построения сложных моделей глубокого обучения.

**Актуальность исследования**

Актуальность данной работы определяется активным развитием технологий искусственного интеллекта и необходимостью их применения в различных сферах. Современные методы машинного обучения, особенно нейросетевые модели, способны решать сложные задачи анализа медицинских изображений, что значительно повышает точность диагностики и снижает нагрузку на врачей. В

медицине ИИ используется для автоматического выявления аномалий на рентгеновских снимках, МРТ и КТ-изображениях, что позволяет ускорить постановку диагноза и повысить эффективность лечения.

Одной из важных задач является классификация рентгеновских изображений заболеваний легких, поскольку легочные патологии (такие как пневмония, туберкулез, фиброз, опухоли и другие) остаются серьезной медицинской проблемой. Автоматизированные системы анализа рентгеновских снимков могут помочь в раннем выявлении заболеваний и сократить вероятность врачебных ошибок.

Использование нейросетевых моделей для классификации медицинских изображений требует тщательной подготовки данных, выбора подходящей архитектуры сети, настройки параметров обучения и оценки качества модели. В данном исследовании будет рассмотрен полный процесс построения модели сверточной нейронной сети (CNN) для классификации рентгеновских изображений, включающий подготовку данных, выбор методов обучения и анализ полученных результатов.

**Объект и предмет исследования**

Объектом исследования является модель сверточной нейронной сети, предназначенная для обработки и классификации медицинских изображений.

Предмет исследования – набор рентгеновских изображений заболеваний легких, состоящий из 9 классов, каждый из которых соответствует определенному диагнозу.

**Цель и задачи исследования**

**Целью** исследования является разработка и обучение нейросетевой модели для классификации рентгеновских изображений легочных заболеваний. Для достижения этой цели необходимо выполнить следующие **задачи:**

1. Провести анализ существующих методов машинного обучения и нейронных сетей, используемых для анализа медицинских изображений.

2. Изучить и сравнить основные программные библиотеки для построения моделей машинного обучения и сверточных нейронных сетей.

3. Подготовить и предобработать данные, включая балансировку классов, нормализацию изображений и аугментацию.

4. Разработать архитектуру сверточной нейронной сети для решения задачи классификации.

5. Обучить модель на выбранном наборе данных и оценить ее качество с использованием стандартных метрик (точность, полнота, F1-score и другие).

6. Провести тестирование модели и анализ ошибок классификации.

7. Оценить возможность применения модели в реальной клинической практике и предложить направления для ее улучшения.

**Научная новизна и практическая значимость**

Научная новизна работы заключается в разработке и анализе модели сверточной нейронной сети для автоматической диагностики легочных заболеваний на основе рентгеновских снимков. В работе исследуются методы предобработки медицинских изображений, а также анализируется влияние различных архитектурных решений CNN на точность классификации.

Практическая значимость исследования состоит в том, что разработанная модель может быть использована для автоматической диагностики легочных патологий. Ее внедрение в медицинскую практику позволит сократить время анализа снимков, повысить точность диагностики и уменьшить нагрузку на врачей-рентгенологов.

**Структура работы**

Диссертация состоит из введения, четырех глав, заключения, списка использованных источников и приложений.

- В первой главе рассматриваются основные понятия машинного обучения и сверточных нейронных сетей, а также метрики оценки качества моделей.
- Вторая глава посвящена подготовке данных, методам обучения нейросетей и анализу архитектур сверточных нейронных сетей.
- В третьей главе приводится экспериментальная часть, включающая описание процесса обучения модели, анализ полученных результатов и их интерпретацию.

• В четвертой главе описано создание простого приложения – чат-бота – для взаимодействия с обученной моделью.

Заключение содержит основные выводы и направления дальнейшего исследования.

По результатам работы опубликованы три статьи [22, 23, 25] и одни тезисы доклада [24].

Результаты работы докладывались на четырёх научных конференциях:

- XXXIII и XXXIV научные конференции «День науки» (Амурский государственный университет, 2024 и 2025 гг.);

- Международная научная конференция «Актуальные проблемы прикладной математики, информатики и механики» (Воронеж, 2-4 декабря 2024 года);

– V Региональная научно-практическая конференция «ТОГУ-Старт: фундаментальные и прикладные исследования молодых» 16-20 апреля 2024 г.

- XI Международной научной конференции «Математическое и компьютерное моделирование», посвященной памяти В.А. Романькова 15 марта 2024 г.

# 1 ВВЕДЕНИЕ В ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ

## 1.1 История развития искусственного интеллекта

### 1.1.1 Предтечи

Первые идеи, связанные с искусственным интеллектом, можно обнаружить в древних мифах и легендах. Так, например, в «Эпосе о Гильгамеше» рассказывается о чудовище под именем Хувава, «хранитель входа в Шамаш». Вот как Энкиду, друг Гильгамеша, описал создание: «Хувава – необычайно сложная машина. Его рёв как наводнение, его рот – огонь, его дыхание – смерть. Он слышит, как корова движется в шестидесяти лигах, а его чувства могут проявляться на большом расстоянии. Слабость охватывает всех тех, кто приближается к воротам». Это, возможно, первое описание «робота», дошедшее до наших дней.

Ещё один пример искусственного интеллекта из мифологии – Талос, «Бронзовый автоматон». Огромный бронзовый робот, созданный Гефестом, запрограммированный патрулировать берега острова и обходить по кругу весь Крит три раза в день [1].

Важным этапом для становления искусственного интеллекта являются системы для логических рассуждений, основанные на логике, изобретённой Аристотелем. Работы Аристотеля по этому предмету и многим другим легки в основу нашего современного научного мышления. С точки зрения искусственного интеллекта наиболее интересный аспект работ Аристотеля – его изучение логики. Он предложил идею силлогизма: логическое заключение, сделанное из двух посылок. Так же Аристотель в своих работах затрагивал несколько важных тем, которые и по сей день интересуют исследователей искусственного интеллекта. В своей книге «Политика» Аристотель высказал предположение, что когда-нибудь автоматы смогут выполнять работу за людей, тем самым заменив рабов. Однако, самые большие события в области искусственного интеллекта произошли в последние несколько столетий.

В конце семнадцатого – начале восемнадцатого века немецкий математик и философ Готфрид Лейбниц разработал формальный математический язык для

рассуждения. Его универсальный язык позволил с большой точностью выражать проблемы всех видов, а затем приступать к их решению. Работа Лейбница послужила основой для пропозициональной логики и логики предикатов, являющиеся важными основами для сегодняшних исследований искусственного интеллекта.

В девятнадцатом веке английский математик Джордж Буль разработал булеву алгебру, которая используется как часть логики высказываний и предикатов. Булева алгебра широко применяется в компьютерных науках [2].

### 1.1.2 Зарождение современного искусственного интеллекта

Принято считать, что искусственный интеллект как наука начался со статьи английского математика Алана Тьюринга «Computing Machinery and Intelligence», вышедшей в 1950 году. Возможность создания «мыслящей машины» и наличия интеллекта у компьютеров обсуждалась к тому времени уже несколько лет в научном сообществе, к которому принадлежал и Тьюринг. Ему же принадлежит знаменитый тест об интеллектуальности машины: Тьюринг предложил считать думающей такую машину, которую испытатель в процессе общения с ней не сможет отличить от человека. Иначе говоря, компьютер должен стать неотличим от человека во владении естественным языком, чтобы пройти тест на интеллектуальность. Идея Тьюринга заключается в том, что нет необходимости видеть, действительно ли машина что-то знает, осознает себя или даже права ли она вообще. Вместо этого тесть Тьюринга показывает, что машина может обрабатывать большие объёмы информации, интерпретировать речь и общаться с людьми.

Тест Тьюринга помогает понять, какой объём работы необходим, чтобы сконструировать искусственный интеллект: необходимы обработка естественного языка, представление знаний, умение принимать решения на основе данных, обучение на опыте. Буквальное следование формулировке теста породило широко известную деятельность по созданию разговорных программ, или, так называемых чатботов – программ, способных «общаться» с человеком. Одной из

таких программ была ELIZA; в шестидесятые годы она могла беседовать в стиле классического психоаналитика.

Стоит отметить важное событие в становлении искусственного интеллекта – модель искусственного нейрона Мак-Каллока – Питтса. В 1943 году Уоррен Мак-Каллок и Уолтер Питтс опубликовали работу «Логическое исчисление идей, относящихся к нервной деятельности». Именно эта работа заложила основы разработки искусственного интеллекта и революционного представления о мозге как о компьютере. Несмотря на то, что описание нейрона в этой работе оказалось далеким от истины, сама идея оказала значительное влияние на развитие искусственного интеллекта.

В современном смысле термин «Искусственный интеллект» впервые был использован на конференции в Дартмутском колледже. В 1956 году четыре отца-основателя искусственного интеллекта: Джон Маккарти, Марвин Минский, Натаниэль Рочестер и Клод Шеннон – организовали Дартмутский семинар – летнюю научную школу. Они считали, что всякий аспект обучения или любое другое свойство интеллекта может в принципе быть столь точно описано, что машина сможет его имитировать. Тогда же был сделан акцент на том, что искусственный интеллект должен базироваться на обучении – свойстве, присущем человеческому интеллекту.

История развития искусственного интеллекта и технологий нейросетей отличается периодами бурного развития, связанного с оптимистическими ожиданиями скорых результатов и неизбежных последующих разочарований, которые стали называть «зимы искусственного интеллекта». Ничего удивительного в этом нет – и учёные, и журналисты сильно завышали ожидания, связанные с достижениями искусственного интеллекта.

Первый период оптимизма в части искусственного интеллекта был связан с проведенными Ф. Розенблаттом экспериментами в середине 1950-х годов. Обучаемая модель называлась «Персептрон». Журналисты писали, что в скором будущем персептроны смогут распознавать людей, читать и писать, переводить

любую речь с одного языка на другой. На самом деле модель Розенблатта представляла собой простой линейный классификатор, который научился различать лево от право.

В IBM Н. Рочестер с коллегами разработали некоторые из первых ИИ программ. Г. Гелернтер в 1959 году построил устройство для доказательства геометрических теорем. В МИТ Маккарти разработал язык LISP, который стал главным языком программирования искусственного интеллекта на следующие 30 лет. Также, Маккарти опубликовал работу под названием Programs with Common Sense – программы со здравым смыслом, в котором он описал гипотетическую программу под названием Avice Taker, рассматриваемую как первую полную систему искусственного интеллекта.

Две длительные «зимы» искусственного интеллекта относят к периодам 1966-1974 и 1987-1993 годов. Развитие искусственного интеллекта циклично, ажиотаж сменялся потерей интереса, сокращением финансирования, то есть периодами, которые называют «зима искусственного интеллекта», но потом снова приходит время повышенного интереса к исследованиям в данной области.

Одной из причин первой зимы искусственного интеллекта можно назвать публикацию книги «Персептроны» в 1969 году под авторство Марвина Минского и Сеймура Пейперта, в которой теоретически обоснованы ограничения конструкции Ф. Розенблатта. Поскольку более сложные многослойные модели на тот момент не были широко известны, это оттолкнуло многих исследователей от изучения нейронных сетей. Почти десять лет после этого заниматься искусственным интеллектом было немодно и неприбыльно, так как финансирование этого направления практически прекратилось.

Но наиболее важной причиной для первой зимы стала неудача с мгновенным переводом устной и письменной речи, то есть проекта машинного перевода с одного языка на другой. Во времена «холодной войны» правительство США поставило задачу разработки программы для быстрого и надежного перевода документов с русского на английский и обратно. Начинаю с 1954 года, эти исследования активно спонсировались, оптимизм был тоже безграничен. После десяти

лет исследования выяснилось, что машинный перевод – очень сложная задача. В отчёте по результатам исследования Национальный исследовательский совет США изложил, что перевод, сделанный человеком, является более точным, быстрым, дешевым. Потратив около 20 млн долл., совет свернул все разработки, в результате чего исследования были прекращены.

Начало второй зимы искусственного интеллекта связано с крахом ЛИСП-машин в 1987 году. В 1980-х годах корпорации по всему миру взяли на вооружение экспертные системы, которые являлись одной из форм искусственного интеллекта. Различные компании стали разрабатывать и внедрять экспертные системы, к 1985 году было потрачено на искусственный интеллект свыше млрд долл. Для удовлетворения этих потребностей выросла целая индустрия, в которую входили разработчики программ и производители оборудования, Symbolics, LispMachinesInc. Они создали специализированные компьютеры для искусственного интеллекта, ЛИСП-машины, оптимизированные для обработки языка программирования LISP.

Однако в 1987 году рынок ЛИСП-машин рухнул. Причиной послужило бурное развитие рабочих станций. Sun Microsystems предложила мощную альтернативу ЛИСП-машинам, а компания LucidInc создала среду LISP для нового класса рабочих станций. Рабочие станции превосходили ЛИСП-машины в производительности [3].

### 1.2 Области искусственного интеллекта

Машинное обучение (МО): Машинное обучение – это более широкая область, которая включает в себя изучение алгоритмов и статистических моделей, которые позволяют компьютерам улучшать свою производительность при выполнении задач с помощью опыта или данных. Речь идет о создании алгоритмов, которые могут учиться и делать прогнозы или решения на основе данных. Алгоритмы МО можно разделить на обучение с учителем (когда алгоритм учится на помеченных данных), обучение без учителя (когда он находит закономерности в немаркированных данных) и обучение с подкреплением (когда алгоритмы учатся методом проб и ошибок, получая обратную связь в динамической среде).

Глубокое обучение – это подмножество машинного обучения, которое фокусируется на использовании многоуровневых нейронных сетей (отсюда и термин «глубокое») для моделирования и извлечения закономерностей из сложных данных. Эти нейронные сети вдохновлены структурой и функциями человеческого мозга. Глубокое обучение привлекло значительное внимание и успех в различных областях, таких как компьютерное зрение, обработка естественного языка и распознавание речи, благодаря его способности автоматически изучать представления на основе необработанных данных.

Базисные модели – это крупномасштабные модели искусственного интеллекта, предварительно обученные на огромных объемах разнообразных данных для изучения общих представлений языка, зрения или других областей. Эти модели составляют основу для различных последующих задач ИИ. Они служат основой, поскольку их можно точно настроить или адаптировать к конкретным задачам с дополнительным обучением на небольших наборах данных для конкретных задач. Примеры базовых моделей включают модели GPT (генеративный предварительно обученный трансформатор), такие как GPT-3 и BERT (представления двунаправленного кодировщика из трансформаторов).

### 1.3 Основные задачи, решаемые методами машинного обучения и искусственными нейронными сетями

Обучение с учителем – это направление машинного обучения, объединяющее алгоритмы и методы построения моделей на основе множества примеров, содержащих пары «известный вход – известный выход».

Иными словами, чтобы алгоритм относился к обучению с учителем, он должен работать с примерами, которые содержат не только вектор независимых переменных (атрибутов, признаков), но и значение, которое должна выдавать модель после обучения (такое значение называется целевым). Разность между целевым и фактическим выходами модели называется ошибкой обучения (невязкой, остатками), которая минимизируется в процессе обучения и выступает в качестве «учителя». Значение выходной ошибки затем используется для вычисления коррекций параметров модели на каждой итерации обучения [4].

Обучение без учителя – технология машинного обучения, в которой для коррекции параметров обучаемой модели не используется целевая функция. Иными словами, в обучающих примерах при обучении без учителя не нужно иметь заранее заданные выходы модели.

В алгоритмах обучения без учителя выходная ошибка модели на обучающем множестве не вычисляется. Вместо неё используется информация о текущем состоянии параметров модели и примеров обучающего множества. Например, это может быть Евклидово расстояние между вектором признаков примера и вектором весов нейрона, которое и будет управлять коррекцией параметров модели в ходе обучения [5].

В искусственном интеллекте и машинном обучении – задача разделения множества наблюдений (объектов) на группы, называемые классами, на основе анализа их формального описания. При классификации каждая единица наблюдения относится определенной группе или номинальной категории на основе некоторого качественного свойства.

В машинном обучении задача классификации решается с использованием обучения с учителем, поскольку классы определяются заранее и для примеров обучающего множества метки классов заданы. Аналитические модели, решающие задачу классификации, называются классификаторами.

Задача классификации представляет собой одну из базовых задач прикладной статистики и машинного обучения, а также искусственного интеллекта в целом. Это связано с тем, что классификация является одной из наиболее понятных и простых для интерпретации технологий анализа данных, а классифицирующие правила могут быть сформулированы на естественном языке.

### 1.4 Оценка качества моделей на данных

Оценка качества моделей является критически важным этапом разработки и внедрения сверточных нейронных сетей (SNN). Без корректной и точной оценки невозможно определить, насколько хорошо модель решает поставленную задачу и какие аспекты требуют улучшения. Методы оценки зависят от типа

задачи: регрессия или классификация. Рассмотрим метрики качества для каждого из этих типов задач.

1.4.1 Метрики качества в задачах регрессии

Задачи регрессии подразумевают предсказание числового значения. Метрики качества в таких задачах направлены на измерение ошибки предсказания модели относительно истинных значений. Рассмотрим наиболее распространенные метрики:

- **Среднеквадратичная ошибка (Mean Squared Error, MSE)**: $MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - y_i)^2$ Эта метрика вычисляет среднее значение квадратов разностей между истинными и предсказанными значениями. Чем меньше MSE, тем лучше работает модель. Однако из-за квадрата ошибки метрика более чувствительна к выбросам.

- **Средняя абсолютная ошибка (Mean Absolute Error, MAE)**: $MAE = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - y_i\right|$. В отличие от MSE, эта метрика оценивает среднюю абсолютную разницу между предсказанными и истинными значениями, что делает её менее чувствительной к выбросам.

- **Коэффициент детерминации (R-squared, $R^2$)**: $R^2 = 1 - \frac{\sum(y_i - y_i)^2}{\sum(y_i - \bar{y}_i)^2}$ Этот коэффициент показывает, какую долю изменчивости целевой переменной объясняет модель. Значение $R^2$ находится в диапазоне от 0 до 1, где 1 указывает на идеальное предсказание.

1.4.2 Метрики качества в классификации

В задачах классификации цель модели – правильно предсказать категорию, к которой относится объект. Метрики оценки качества классификаторов ориентированы на анализ точности предсказаний.

Рассмотрим основные метрики качества для оценки моделей бинарной классификации.

Стоить начать с таких понятий, как истинно положительный (true positive, TP) и истинно отрицательный (true negative, TN) исходы. Объекты классифицируются в соответствии с их реальными значениями. Ложные исходы – ложно положительный (false positive, FP) и ложно отрицательный (false negative, FP). Классификация объектов не соответствует их реальным значениям.

Для удобного визуального представления результатов обучения модели можно построить таблицу, столбцы которой будут отображать реальные значения (классы) объектов, а строки – значения, которые попыталась предугадать модель. Такая таблица называется матрица ошибок, или матрица несоответствий (confusion matrix).

В случае бинарной классификации, когда есть всего два класса объектов, результаты матрицы ошибок сводятся к четырём случаям, объяснённым выше.

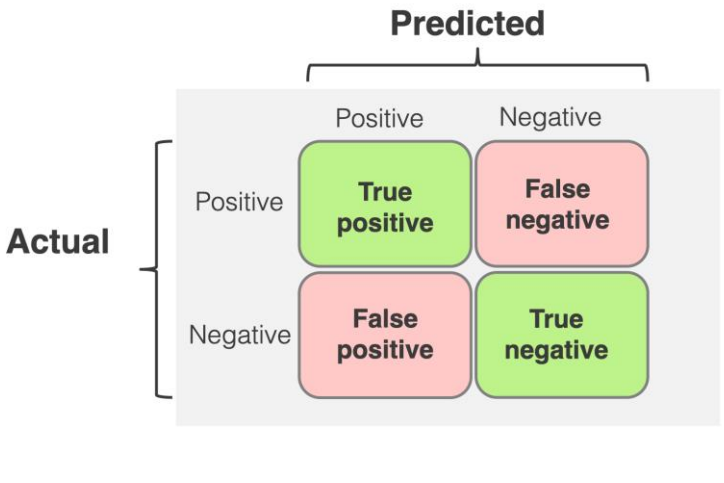



Рисунок 1 – матрица ошибок для двух классов

Когда классов объектов больше двух, кроме истинно положительного исхода, матрица ошибок показывает много ложных результатов, в каждом из которых модель "перепутала" (почему и называется Confusion matrix, confused – буквально, перепутать) истинный класс с другим. Соответственно, столбцов и строк будет больше, столько же, сколько и классов.

На основе TP, TN, FN, FP используются метрики качества моделей. Каждая метрика, показанная ниже, высчитывается, очевидно, для каждого класса отдельно.

Основные метрики включают:

- **Точность (Accuracy)**: $Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$ Точность определяет долю правильно классифицированных объектов от общего числа примеров.

- **Полнота (Recall, Sensitivity)**: $Recall = \frac{TP}{TP + FN}$ Эта метрика показывает, какую часть объектов положительного класса модель смогла правильно идентифицировать. Особенно важна в тех случаях, когда важнее минимизировать пропуск важных случаев положительного класса (false negatives).

- **Точность (Precision)**: $Precision = \frac{TP}{TP + FP}$ Precision оценивает, насколько предсказанные положительные классы действительно принадлежат этому классу. Важна, когда в определённый класс слишком дорого «подмешивать» другие классы.

- **F1-мера**: $F1 = 2\frac{Precision \times Recall}{Precision + Recall}$ Это гармоническое среднее между точностью и полнотой, позволяющее учесть баланс между ложными срабатываниями и пропущенными объектами.

- **ROC-кривая и AUC (Area Under Curve)**: ROC-кривая строится по значениям True Positive Rate (TPR) и False Positive Rate (FPR) при различных порогах классификации, где $TPR = \frac{TP}{TP + FN}$, а $FPR = \frac{FP}{TN + FP}$. Площадь под кривой (AUC) используется для оценки качества бинарного классификатора: чем больше значение AUC, тем лучше модель. Площадь под ROC-кривой измеряется от 0 до 1, то есть, можно сказать, что вся «площадь графика» равна единице, и чем больше область ниже кривой, тем выше точность классификации. Единица означает идеальную точность классификации, а 0.5 – случайное угадывание. Пример ROC-кривой с AUC можно посмотреть на рис. 2.

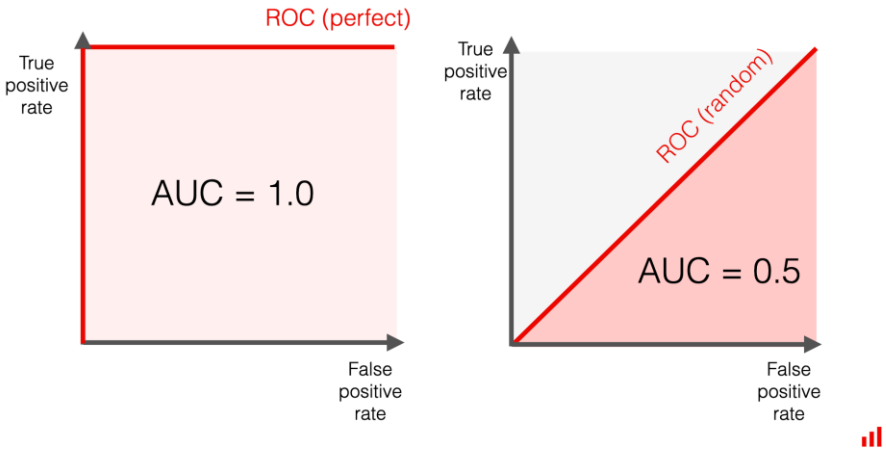



Рисунок 2 – пример ROC-AUC

Выбор конкретной метрики зависит от требований задачи. В некоторых случаях может потребоваться использование нескольких метрик одновременно для всесторонней оценки работы модели.

Accuracy подходит в простых случаях, когда размеры классов сбалансированы, и нет особых требований к распознаванию отдельных классов. Не пригоден, когда в одном классе 1000 объектов, а в другом – 10. Тогда, даже если модель будет определять все объекты как первый класс, оценка по Accuracy будет очень высокой, хотя, фактически модель не работает.

Precision Полезна там, где минимизация ложноположительных ошибок критична (например, выявление дефектов производства, определение признаков заболевания). Поэтому, выгодно использовать в случаях, когда ложноположительный исход дороже ложноотрицательного (например, реклама, рекомендательная система, медицинская диагностика).

Recall важно учитывать, когда цена пропуска важного случая высока (например, отсутствие диагностики серьёзного заболевания, пропуск мошеннических операций). Рекомендуется, если предпочтительнее поймать больше положительных примеров, пусть даже увеличивая число ложных срабатываний (медицина, безопасность, проверка документов).

F1 score балансирует обе метрики, помогает выбрать компромисс между количеством обнаруженных значимых примеров и уровнем шума. Но менее чувствителен к экстремальным отклонениям в данных (например, огромный перевес одного класса над другим).

### 1.5 Современное состояние средств искусственного интеллекта

В настоящее время искусственный интеллект применяется во многих прикладных задачах.

В медицине ИИ может стать незаменимым для специалистов и вполне способен улучшить статистику проведения многих операций. Самым известным примером по внедрению ИИ в медицину остается суперкомпьютер Watson. На сегодняшний день он может обрабатывать 200 млн цифровых документов за три секунды. Watson в первую очередь призван помочь врачам в работе с электронными медицинскими картами. Он способен составить историю болезни пациента, членов его семьи, структурировать генетическую предрасположенность к тем или иным патологиям и выдать моментально всю эту информацию лечащему доктору. Система предлагает свои рекомендации по лечению заболеваний, в том числе онкологических [6].

В промышленной сфере применения систем искусственного интеллекта востребована возможность автоматизировать рабочие процессы. Чаще сегодня автоматизируют операции, выполняемые на конвейере.

В сфере образования с помощью ИИ планируется автоматизировать работу по подбору учебного материала и способа преподавания, подходящих конкретному ученику, чтобы облегчить процесс усвоения материала всем категориям учащихся.

Отдельного внимания заслуживает ChatGPT – чат-бот с генеративным искусственным интеллектом разработанный OpenAI. Система способна отвечать на вопросы, генерировать тексты на разных языках, включая русский, относящиеся к различным предметным областям. Важной особенностью является возможность генерации по запросу программ на различных языках программирования [7]. Именно после выпуска GPT 3.5 начался бум искусственного интеллекта.

На сегодняшний день технологии искусственного интеллекта (ИИ) стремительно развиваются, охватывая всё больше сфер применения — от медицины и промышленности до образования, транспорта и творчества. Особенно активно развиваются области глубокого обучения (deep learning) и обработки естественного языка (natural language processing), где наблюдаются впечатляющие достижения как в теории, так и в прикладных задачах.

Одним из ключевых факторов прогресса стало появление трансформерных архитектур, таких как GPT (Generative Pretrained Transformer) от OpenAI, BERT и T5 от Google. Эти модели демонстрируют выдающиеся результаты в задачах понимания и генерации текста, машинного перевода, автоматического ответа на вопросы и др. В частности, GPT-4 уже активно используется в различных продуктах, включая интеллектуальные ассистенты, системы поддержки клиентов и медицинские справочные платформы.

В области компьютерного зрения продолжают доминировать свёрточные нейронные сети (CNN), однако в последние годы получили распространение и визуальные трансформеры (Vision Transformers, ViT). Они демонстрируют сопоставимую или даже превосходящую эффективность по сравнению с классическими CNN в задачах классификации изображений, сегментации и распознавания объектов. Например, модели Swin Transformer и ConvNeXt успешно применяются в промышленных системах анализа медицинских изображений, спутниковых снимков и видеонаблюдения.

Для медицинских задач особое значение приобрели специализированные модели, такие как:

CheXNet — CNN-модель, обученная на рентгеновских снимках грудной клетки, способная выявлять признаки пневмонии с точностью, сравнимой с врачами-радиологами;

MONAI (Medical Open Network for AI) — открытая платформа для создания медицинских ИИ-приложений на базе PyTorch, включающая множество готовых моделей и инструментов;

CLIP и SAM от Meta AI — мультифункциональные модели, объединяющие изображение и текст, открывают перспективы для создания универсальных диагностических систем.

Таким образом, современное состояние ИИ характеризуется высоким уровнем зрелости технологий, доступностью вычислительных ресурсов (включая облачные решения и специализированные GPU), а также наличием обширных открытых датасетов, что способствует быстрому прототипированию и внедрению интеллектуальных систем в реальную практику.

### 1.6 Программные библиотеки для построения моделей машинного обучения и нейронных сетей

Современные библиотеки для машинного обучения и нейронных сетей значительно упрощают процесс разработки, обучения и тестирования моделей. Рассмотрим основные из них:

TensorFlow – одна из самых популярных библиотек для глубокого обучения, разработанная Google. Она поддерживает как низкоуровневые операции, так и высокоуровневый API Keras для быстрого создания нейросетей.

PyTorch – фреймворк от Facebook, известный своей динамической вычислительной графикой и удобным отладочным процессом. Популярен среди исследователей и разработчиков.

Keras – высокоуровневый API, который позволяет быстро и удобно разрабатывать модели, используя TensorFlow в качестве бэкенда.

Scikit-learn – библиотека для классического машинного обучения, включающая алгоритмы классификации, регрессии, кластеризации и снижения размерности.

XGBoost – специализированная библиотека градиентного бустинга, применяемая в задачах регрессии и классификации, особенно популярная в соревнованиях по анализу данных.

FastAI – надстройка над PyTorch, позволяющая ускорить процесс обучения моделей и автоматизировать подбор параметров.

ONNX (Open Neural Network Exchange) – открытый формат для обмена моделями между различными фреймворками, такими как TensorFlow, PyTorch и другими.

Выбор библиотеки зависит от конкретных задач и требований к модели. TensorFlow и PyTorch предпочтительны для глубокого обучения, тогда как Scikit-learn и XGBoost часто используются для традиционного машинного обучения.

# 2 НЕЙРОННЫЕ СЕТИ ДЛЯ АНАЛИЗА ИЗОБРАЖЕНИЙ

## 2.1 Модель искусственного нейрона

Нейронные сети состоят из искусственных нейронов, и, как можно понять из названия, искусственные нейроны являются попыткой смоделировать работу биологического нейрона [8].

Биологический нейрон – это нервная клетка, которая состоит из тела, дендритов и аксона. Тело соединено со множеством коротких и толстых отростков, которые называются дендритами. Это входные отростки - они принимают импульсы с тех нейронов, которые находятся в сети раньше, чем этот нейрон. Имеется один очень длинный и тонкий отросток, который называется аксоном. Это выходной отросток, по которому нейрон передает электрохимический импульс следующим нейронам в сети. Через множество дендритов нейрон получает входные сигналы, в теле нейрона они обрабатываются, а через единственный аксон выходной импульс от нейрона передается дальше. Окончание аксона имеет разветвления, через которые выходной сигнал может быть передан нескольким следующим нейронам.

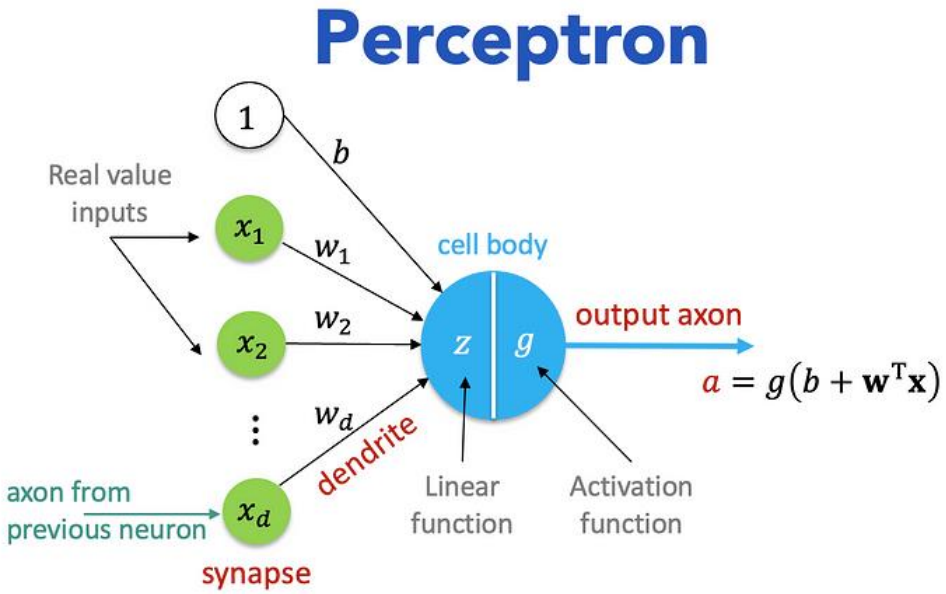



Рисунок 3 – Упрощенная схема искусственного нейрона

Здесь тело нейрона представлено в виде "черного ящика", который принимает входные данные выполняет с ними некоторые операции, а затем выводит результат.

Как нейроны взаимодействуют между собой в нейронной сети? Дело в том, что аксон каждого нейрона на своем конце имеет большое количество так называемых синапсов. Синапс – это место соединения выходного аксона одного нейрона с входными дендритами другого нейрона.

Через синапс передается нервный импульс. А сам нервный импульс формируется в теле нейрона, и фактически тело нейрона выступает сумматором, который принимает все входные импульсы, взвешивает их, передает в некоторую активирующую функцию и принимает решение, запускать импульс дальше по своему аксону или нет. Решение принимается очень просто – если суммарные входные импульсы превышают некий заданный порог, то выходной импульс запускается.

На этой основе были построены математические модели нейронов.

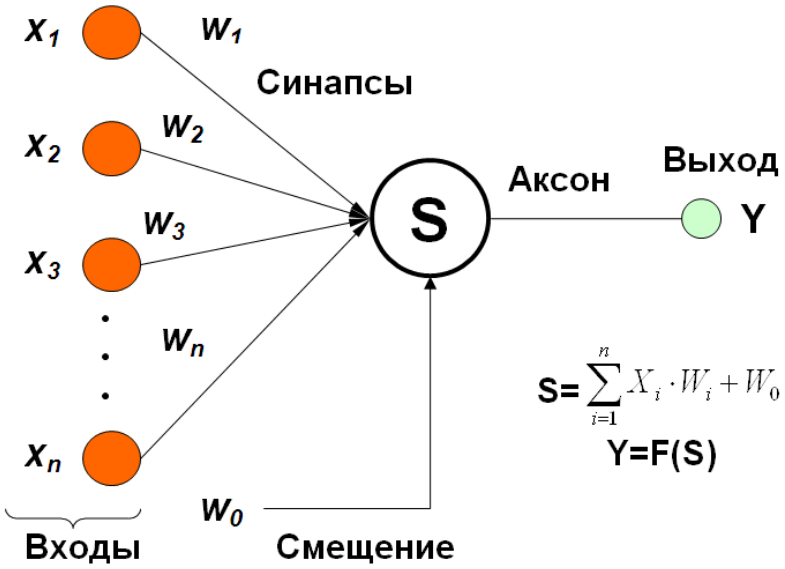



Рисунок 4 – Математическая модель искусственного нейрона

Данная модель довольно простая. На вход математического нейрона поступает некоторое количество входных параметров $x_1$, $x_2$, $x_3$, …, $x_n$, т. е. дендритов, каждый входной параметр имеет свой вес – $w_1$, $w_2$, $w_3$, …, $w_n$.

В теле искусственного нейрона имеется сумматор, где каждый входной сигнал умножается на некоторый действительный весовой коэффициент и формируется итоговая сумма. Полученное значение передается в функцию активации. На выходе осуществляется проверка значения функции активации. Если ее значение выше некоторого порога, то на выход из тела нейрона передается значение единица. В этом случае нейрон активируется и в аксон передается соответствующий сигнал. Если же итоговое значение в функции активации ниже порога, то на выход из тела нейрона подается значение, равное нулю, и нейрон считается неактивированным [9].

Стоит отдельно поговорить о функциях активации. Функция активации — это неотъемлемая часть архитектуры нейронной сети, определяющая выход нейрона на основе входного сигнала. Без нелинейной функции активации любая нейронная сеть сводилась бы к простой линейной модели, неспособной решать сложные задачи классификации и распознавания.

Существует множество различных функций активации, каждая из которых имеет свои особенности, преимущества и ограничения. Ниже представлены некоторые из них:

Сигмоидная функция активации - $\sigma(x)=\frac{1}{1+e^{-x}}$ – это нелинейная функция, которая преобразует входное значение в диапазоне от отрицательной бесконечности до положительной бесконечности в значение от 0 до 1. Эта функция активации часто используется в нейронных сетях для задач бинарной классификации. Практически всегда используется в выходном слое. Имеет недостаток – проблема затухающего градиента. При значении, близком к единице, шанс определения входного значения как первый класс очень высока. Если значение близко к нулю, то шанс определения входного значения как первый класс очень низкая.

ReLU (Rectified Linear Unit) – это нелинейная функция активации, которая широко используется в глубоком обучении. Она преобразует входное значение в значение от 0 до положительной бесконечности. Если входное значение

меньше или равно нулю, то ReLU выдает ноль, в противном случае - входное значение. Является стандартным вариантом в большинстве архитектур. Легко вычисляется, нет проблем с затуханием градиента, но есть проблемы с "мертвыми нейронами" - нейроны, которые получили отрицательное значение, могут перестать обучаться.

Leaky ReLU (Rectified Linear Unit) – это функция активации, которая используется в нейронных сетях для введения нелинейности в выходные данные каждого нейрона. По сути, модифицированный вариант ReLU, которых решает проблему "умирающих нейронов". В отличие от ReLU, Leaky ReLU так же возвращает само значение при положительном входном значении, но уже при отрицательных значениях возвращает линейную функцию от входа, умноженную на небольшой коэффициент, называемый отрицательным уклоном.

Softmax – функция, превращающая логиты (наборы чисел) в вероятности, причем сумма последних равна единице. Формула: $Soft\,max(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_k}}$, где z – входной вектор, $z_i$ – элемент вектора, k – количество элементов в векторе. Используется при многоклассовой классификации, практически всегда в выходном слое.

Функция ошибки (или функция потерь, loss function) — это математическое выражение, которое количественно оценивает, насколько предсказания нейронной сети отличаются от истинных значений. Она играет центральную роль в обучении моделей машинного обучения: именно её значение сеть стремится минимизировать, подбирая оптимальные значения весов. Является критерием качества обучения: чем меньше значение функции ошибки, тем лучше модель справляется с задачей. Определяет направление и величину корректировок параметров сети на каждом шаге обучения через градиентный спуск. Влияет на динамику и устойчивость процесса обучения: неправильно выбранная функция может привести к переобучению, нестабильности или медленной сходимости.

Есть разные виды функций ошибки.

1. Метрические функции для регрессии:

Среднеквадратичная ошибка (Mean Squared Error): $MSE = \frac{1}{n}\sum_{i=1}^{n}( y_i - y_i )^2$.

Чувствительна к выбросам. Из-за возведения в квадрат, даже небольшая разница между предсказанным и реальным значением, больше единицы, может выдать большую ошибку. Популярна в задачах, где важна точность численного предсказания.

Средняя абсолютная ошибка (Mean Absolute Error): $MAE = \frac{1}{n}\sum_{i=1}^{n}\left| y_i - y_i \right|$

Менее чувствительна к выбросам, так как вместо возведения в квадрат используется модуль. Даёт более "ровное" обучение.

2. Функции для задач классификации:

Бинарная кросс-энтропия (Binary Crossentropy):

$L = -\frac{1}{n}\sum_{i=1}^{n}\left[ y_i \log( y ) + (1 - y_i )\log(1 - y_i ) \right]$. Используется в задачах бинарной классификации. Требует, чтобы выход сети был от 0 до 1.

Категориальная кросс-энтропия (Categorical Crossentropy):

$L = -\sum_{i=1}^{C} y_i \log( y_i )$. Применяется при многоклассовой классификации. Часто совместно используется с функцией активации Softmax.

Процесс обучения нейросети заключается в поиске минимума функции ошибки путём оптимизации параметров модели (весов и смещений).

Основной метод — градиентный спуск:

- Простой градиентный спуск (Gradient Descent): вычисляет градиент по всей выборке.
- Стохастический градиентный спуск (SGD): обновляет веса на каждом отдельном примере.
- Мини-батч градиентный спуск: компромисс — обновление по мини-группам данных.

Улучшенные алгоритмы:

• Adam (Adaptive Moment Estimation) — одна из самых популярных оптимизаций, объединяет преимущества RMSprop и Momentum, хорошо работает «из коробки» в большинстве задач.

• RMSprop — адаптивно изменяет скорость обучения по каждому параметру.

• Adagrad, Adadelta — адаптируют скорость обучения к частоте обновлений.

При градиентном шаге, каждое обновление весов вычисляется по формуле: $w := w - \eta \cdot \frac{\partial L}{\partial w}$, где w – текущие веса, $\eta$ – скорость обучения, а $\frac{\partial L}{\partial w}$ - градиент функции ошибки по параметрам модели.

**2.2 Основные принципы построения нейронный сетей**

В зависимости от расстановки нейронов, нейронные сети можно поделить на типы.

Однослойная нейронная сеть – сеть, в которой сигналы от входного слоя сразу подаются на выходной слой, который и преобразует сигнал и сразу же выдает ответ.

Многослойная нейронная сеть – нейронная сеть, состоящая из входного, выходного и расположенных между ними одного или нескольких скрытых слоев нейронов.

Помимо входного и выходного слоев эти нейронные сети содержат промежуточные, скрытые слои. Такие сети обладают гораздо большими возможностями, чем однослойные нейронные сети, однако методы обучения нейронов скрытого слоя были разработаны относительно недавно.

Работу скрытых слоев нейронов можно сравнить с работой большого завода. Продукт (выходной сигнал) на заводе собирается по стадиям на станках. После каждого станка получается какой-то промежуточный результат. Скрытые слои тоже преобразуют входные сигналы в некоторые промежуточные результаты.

Сети прямого распространения – искусственные нейронные сети, в которых сигнал распространяется строго от входного слоя к выходному. В обратном направлении сигнал не распространяется.

Все сети, описанные выше, являлись сетями прямого распространения, как следует из определения. Такие сети широко используются и вполне успешно решают определенный класс задач: прогнозирование, кластеризация и распознавание.

Однако сигнал в нейронных сетях может идти и в обратную сторону.

Сети с обратными – искусственные нейронные сети, в которых выход нейрона может вновь подаваться на его вход. В более общем случае это означает возможность распространения сигнала от выходов к входам.

В сетях прямого распространения выход сети определяется входным сигналом и весовыми коэффициентами при искусственных нейронах. В сетях с обратными связями выходы нейронов могут возвращаться на входы. Это означает, что выход какого-нибудь нейрона определяется не только его весами и входным сигналом, но еще и предыдущими выходами (так как они снова вернулись на входы).

### 2.3 Подготовка данных для обучения нейронных сетей

Подготовка данных – важнейший этап в обучении нейронных сетей. Качество входных данных напрямую влияет на способность модели к обобщению и точность предсказаний. Процесс подготовки включает сбор данных, их предобработку и разбиение на обучающие и тестовые наборы.

2.3.1 Источники данных

Источники данных для обучения нейронных сетей можно разделить на несколько категорий:

1. Открытые наборы данных

• Kaggle Datasets (https://www.kaggle.com/datasets) – коллекция тысяч наборов данных по разным задачам.

• ImageNet – крупнейший набор изображений, используемый для обучения моделей компьютерного зрения.

• COCO (Common Objects in Context) – набор данных с разметкой для задач детекции, сегментации и captioning.

• MNIST, Fashion-MNIST – стандартные датасеты для тестирования алгоритмов классификации изображений.

2. Генерация данных

• В ряде случаев возможно создать синтетические изображения, например, с помощью GAN (Generative Adversarial Networks) или алгоритмов аугментации.

3. Сбор данных вручную

• Данные могут быть собраны с камер, датчиков или веб-источников. Однако такой подход требует дополнительной разметки.

4. Web Scraping (парсинг сайтов)

• Автоматический сбор изображений из Интернета с последующей фильтрацией и разметкой.

2.3.2 Предобработка данных

Перед обучением нейронной сети данные необходимо обработать, чтобы модель могла эффективно извлекать из них признаки.

Основные этапы предобработки:

1. Приведение к единому формату

• Изменение размера изображений (например, 224×224 пикселя для моделей ImageNet).

• Преобразование в нужное цветовое пространство (RGB, Grayscale).

• Нормализация значений пикселей (приведение к диапазону [0,1] или [- 1,1]).

2. Удаление шума и улучшение качества изображений

• Применение фильтров для устранения артефактов.

• Устранение размытости и автоматическая коррекция контраста.

3. Аугментация данных (Data Augmentation)

• Используется для увеличения объема данных и улучшения обобщающей способности модели.

• Включает случайные трансформации изображений:

▪ Повороты, отражение, масштабирование.

▪ Изменение яркости, контраста, насыщенности.

▪ Добавление случайного шума.

4. Аннотация и разметка данных

• Для задач классификации – присвоение изображению метки (label).

• Для детекции объектов – разметка bounding box'ами.

• Для сегментации – создание масок объектов.

2.3.3 Разделение данных на тренировочный, валидационный и тестовый наборы

После предобработки необходимо разделить данные на три основные части:

1. Тренировочный набор (Training Set)

Используется для непосредственного обучения модели.

Обычно составляет 70-80% всех данных.

2. Валидационный набор (Validation Set)

Используется для настройки гиперпараметров модели и предотвращения переобучения.

Обычно составляет 10-15% данных.

3. Тестовый набор (Test Set)

Оценивает финальную точность модели на новых данных.

Должен быть полностью независим от тренировочного и валидационного набора.

Обычно составляет 10-15% данных.

Методы разделения данных

• Случайное разбиение (Random Split)

Данные распределяются случайным образом. Подходит, если данные однородны.

• Стратифицированное разбиение (Stratified Split)

Сохраняет пропорции классов в каждом из наборов (важно при несбалансированных данных).

- K-fold Cross-Validation

Метод перекрестной проверки, при котором данные делятся на K частей, и модель обучается K раз на разных подвыборках.

### 2.4 Методы обучения нейронных сетей для анализа изображений

Обучение нейронных сетей для анализа изображений – это процесс настройки параметров модели с целью минимизации ошибки на входных данных. Современные методы обучения позволяют эффективно извлекать признаки из изображений, что делает нейросетевые модели широко применимыми в таких задачах, как классификация, сегментация и детекция объектов.

Обучение нейронной сети включает несколько этапов: выбор архитектуры модели, подготовка данных, определение функции потерь, выбор алгоритма оптимизации и настройка гиперпараметров. В зависимости от наличия разметки данные могут использоваться в обучении с учителем, без учителя или в обучении с подкреплением. Наиболее распространенным методом является обучение с учителем, при котором модель обучается на размеченных данных, где каждому изображению соответствует метка.

Для корректного обновления весов нейронной сети используется метод обратного распространения ошибки (backpropagation), который основан на вычислении градиента функции потерь по параметрам модели. Этот процесс выполняется с помощью градиентного спуска и его модификаций, таких как стохастический градиентный спуск (SGD), Adam и RMSprop. Оптимизаторы, такие как Adam, позволяют автоматически адаптировать скорость обучения, что делает процесс обучения более стабильным.

Одним из ключевых аспектов обучения является выбор функции потерь. В задачах классификации часто используется категориальная кросс-энтропия, а в задачах регрессии – среднеквадратичная ошибка (MSE). Для задач детекции и сегментации применяются специальные метрики, такие как IoU (Intersection over Union) и Dice Coefficient.

Для повышения эффективности обучения и предотвращения переобучения применяются различные методы регуляризации. Например, Dropout случайным

образом "отключает" нейроны во время обучения, а L2-регуляризация добавляет штраф за большие значения весов. Также важную роль играет нормализация данных, которая помогает стабилизировать обучение и ускорить сходимость модели.

Одним из распространенных методов ускорения и улучшения обучения является transfer learning (перенос обучения). В этом случае используется предобученная модель, обученная на большом наборе данных (например, ImageNet), и ее параметры адаптируются под новую задачу. Это позволяет значительно сократить время обучения и повысить точность на небольших наборах данных.

Еще одной важной техникой является аугментация данных, которая увеличивает объем обучающего набора за счет искусственных преобразований изображений: вращений, изменений масштаба, сдвигов, отражений, изменения яркости и других трансформаций. Это помогает улучшить обобщающую способность модели и делает ее менее чувствительной к небольшим изменениям в изображениях.

Эффективность обучения нейронной сети оценивается с помощью набора метрик, таких как точность (accuracy), полнота (recall), точность предсказаний (precision) и F1-score. Также широко используется визуализация обучения, например, построение графиков зависимости ошибки и точности от количества эпох, что позволяет отслеживать признаки переобучения или недостаточного обучения модели.

Современные методы обучения нейронных сетей для анализа изображений продолжают совершенствоваться, включая более сложные архитектуры, продвинутые методы оптимизации и интеграцию с новыми технологиями. Использование мощных графических процессоров (GPU) и распределенных вычислений позволяет значительно ускорить обучение сложных моделей, а развитие гибридных методов, таких как комбинация сверточных нейросетей и трансформеров, открывает новые горизонты в компьютерном зрении.

Современные исследования в области обучения нейронных сетей для анализа изображений активно развивают методы самообучения (self-supervised learning) и обучения с малым количеством размеченных данных (few-shot

learning). Эти подходы позволяют значительно сократить зависимость от больших объемов размеченной информации, что особенно важно в медицинских и специализированных задачах. Кроме того, в последнее время наблюдается рост популярности генеративных моделей, таких как диффузионные сети и StyleGAN, которые могут не только создавать изображения, но и улучшать качество входных данных.

### 2.5 Пример построения последовательной нейронной сети для классификации изображений

Сначала подключаются необходимые для обучения нейронной сети библиотеки. Keras – это библиотека для языка программирования Python, которая предназначена для глубокого машинного обучения. Включает в себя большое количество слоев и параметров: функции потерь, оптимизаторы, метрики оценки. Numpy – добавляет поддержку больших многомерных массивов и матриц, вместе с большой библиотекой высокоуровневых математических функций для операций с этими массивами. Matplotlib – для построения графиков.

Далее необходимо подготовить данные для обучения сети – картины различной одежды: футболка, брюки, свитер, платье, пальто, туфли, рубашка, кроссовки, сумка, ботинки. Итого – 10 классов. Нейронная сеть должна будет классифицировать изображения одним из этих вариантов.

В обучающей выборке всего 60000 изображений, в тестовой – 10000. Все изображения черно-белые, размер 28 на 28 пикселей. Этот набор можно найти [10].

Все данные разбиваются на две группы – тренировочные данные для обучения нейронной сети, и тестовые данные для её проверки.

Следующим шагом будет нормализация данных. Все изображения в наборе черно-белые. Матричное представление чёрно-белого изображения – числа от нуля до двухсот пятидесяти пяти. Нейронная сеть лучше обучается, если все данные находятся в диапазоне от нуля до единицы, поэтому можно поделить интенсивность каждого изображения на 255.

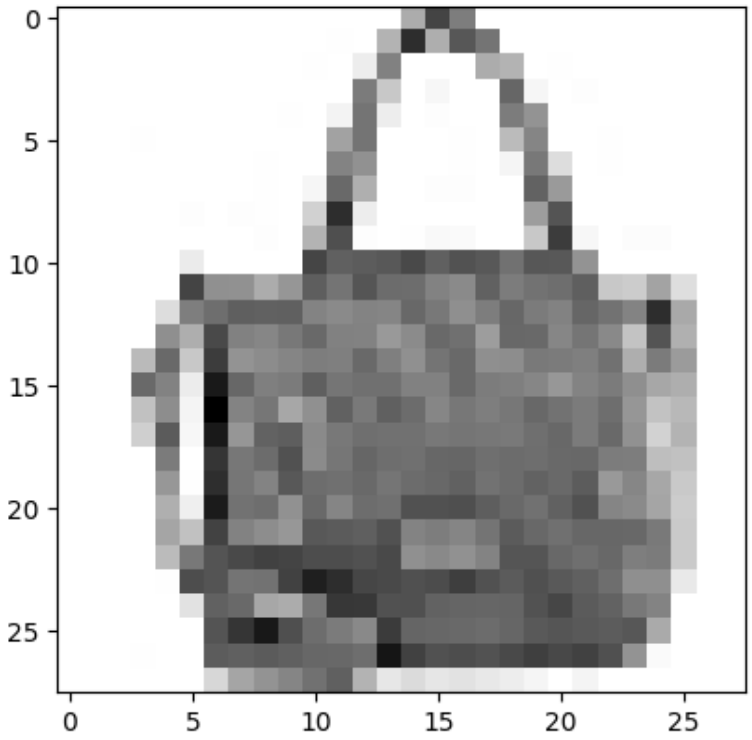



Рисунок 5 – Пример изображения, сумка

Опишем архитектуру нейронной сети. Будем создавать последовательную модель. В ней будет входной полносвязный слой, 200 нейронов, 784 входа в каждый нейрон. Функция активации ReLU (нелинейная функция активации) преобразует входное значение в значение от 0 до положительной бесконечности. Если входное значение меньше или равно нулю, то ReLU выдает ноль, в противном случае - входное значение. Добавляем выходной полносвязный слой, 10 нейронов – количество типов одежды в наборе, функция активации softmax используется для преобразования вектора значений в вероятностное распределение, которое суммируется до 1. Она особенно полезна в многоклассовой классификации, где необходимо определить вероятности для каждого класса.

Компилируем сеть. Задаем функцию ошибки, алгоритм обучения и метрику качества обучения.

Теперь можно приступить к обучению нейронной сети.

Используем метод fit. Подаем обучающую выборку. Batch_size указываем равным двумстам. Так веса нейронной сети будут меняться каждые двести изоб-

ражений. Используется итеративное обучение. Модель будет повторно обучаться на одном и том же наборе данных. Указываем epochs равным десяти, и нейронная сеть пройдет десять итераций обучения.

После обучения нейронной сети можно проверить её качество на тестовых выборках.

Используя метод evaluate проверяем, сколько объектов нейронная сеть смогла классифицировать правильно в тестовой выборке.

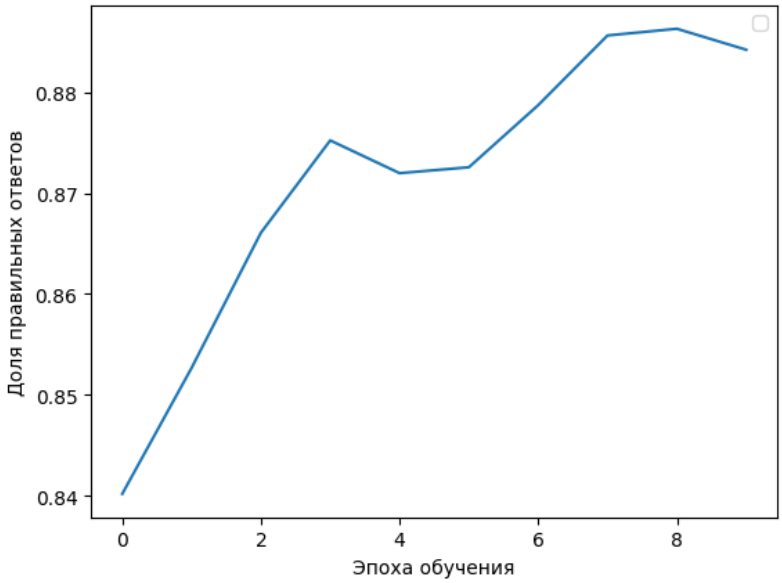



Рисунок 6 – Доля правильных ответов на тестовой выборке

После десяти итераций обучения, модель правильно классифицирует 87,29 процентов объектов из тестовой выборки, что является приемлемым результатом.

Проверим модель на практике. Используем сеть для распознавания моделей одежды.

Выберем случайное изображение. Например, платье.

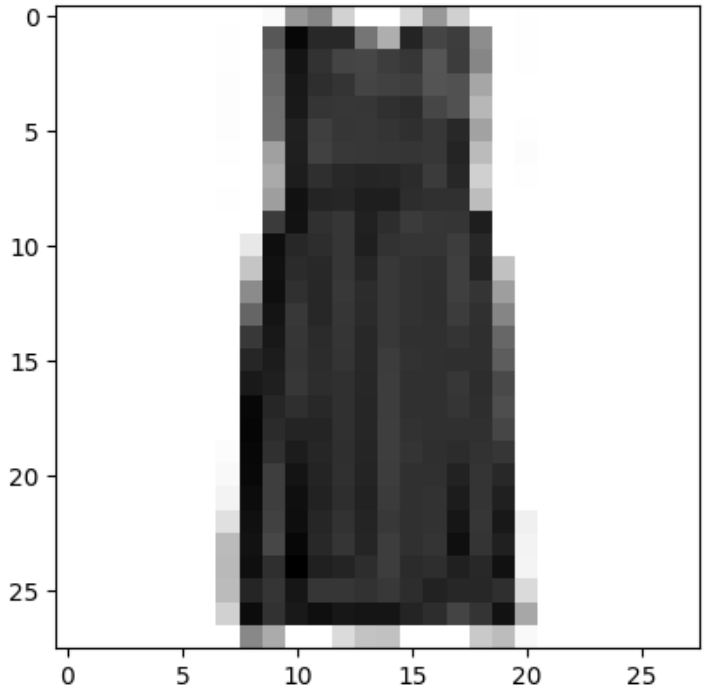



Рисунок 7 – изображение платья

После изменения размерности изображения и его нормализации запускаем распознавание с помощью метода predict.

Результаты предсказания модели – номер класса: 3, название класса: платье. Ответ сходится с правильным.

### 2.6 Сверточные нейронные сети

Сверточные нейронные сети (Convolutional Neural Networks, CNN) – это особый тип нейронных сетей, который эффективно обрабатывает данные с пространственной структурой, такие как изображения. Их ключевым отличием от полносвязных нейронных сетей является использование операции свертки, позволяющей извлекать локальные признаки изображения, а затем объединять их на более высоких уровнях представления.

2.6.1 Операция свертки при работе с изображениями

Операция свертки (convolution) – это основной элемент сверточных нейронных сетей. Она представляет собой математическое преобразование, в ходе которого фильтр (ядро свертки) перемещается по изображению, вычисляя

скалярное произведение его значений с соответствующими пикселями изображения.

Формально операция свертки для изображения $I(x,y)$ с ядром $K(x,y)$ определяется как $S(x,y) = \sum_{i=-m}^{m} \sum_{j=-n}^{n} I(x+i, y+j)K(i,j)$, где $m$ и $n$ – размеры фильтра.

Основные параметры операции свертки:

- Размер ядра (kernel size) – определяет область, на которую распространяется фильтр. Обычно используются ядра размером 3×3 или 5×5.
- Шаг (stride) – указывает, на сколько пикселей перемещается ядро за одну итерацию.
- Дополнение (padding) – используется для сохранения размерности изображения после свертки.

С помощью различных ядер можно выделять горизонтальные и вертикальные границы, текстуры и другие особенности изображения.

2.6.2 Основные компоненты сверточных нейронных сетей

Сверточные нейронные сети обычно состоят из следующих слоев:

1. Сверточные слои (Convolutional Layers) – извлекают пространственные признаки изображения с помощью операции свертки.

2. Функции активации (Activation Functions) – применяют нелинейные преобразования, такие как ReLU (Rectified Linear Unit), для увеличения выразительности модели.

3. Слоев подвыборки (Pooling Layers) – уменьшают размерность данных, сохраняя наиболее важные признаки. Например, max-pooling берет максимальное значение в каждом регионе.

4. Полносвязные слои (Fully Connected Layers) – используются для окончательной классификации признаков.

5. Нормализация и регуляризация – применяются для стабилизации обучения и предотвращения переобучения (Batch Normalization, Dropout).

2.6.3 Популярные архитектуры сверточных нейронных сетей

На протяжении последних лет было разработано множество успешных архитектур CNN, среди которых:

- LeNet-5 – одна из первых сверточных сетей, предложенная Яном Лекуном для распознавания рукописного текста.
- AlexNet – сеть, выигравшая соревнование ImageNet в 2012 году, впервые продемонстрировав мощь глубоких CNN.
- VGG – серия моделей с глубокой, но простой архитектурой, использующей 3×3 сверточные фильтры.
- ResNet – сеть с остаточными связями (skip connections), которые помогают бороться с затуханием градиента.
- EfficientNet – современная архитектура, оптимизированная по параметру соотношения производительности и вычислительных затрат.

Эти модели служат основой для многих современных решений в области компьютерного зрения, включая задачи распознавания объектов, детекции и сегментации изображений.

# 3 СВЕРТОЧНЫЕ НЕЙРОННЫЕ СЕТИ В ЗАДАЧЕ ДИАГНОСТИКИ ЗАБОЛЕВАНИЙ ЛЕГОЧНОЙ СИСТЕМЫ

## 3.1 Описание набора «X-ray Lung Diseases Images»

В качестве обучающего набора данных был выбран набор изображений рентгеновских снимков лёгких, разделенный на 9 классов по группам заболеваний: здоровые лёгкие и 8 различных видов патологий.

Набор находится в свободном доступе в kaggle – системы хранения наборов данных, а также социальной сети для специалистов в области искусственного интеллекта [13]. Данный ресурс предназначен для хранения датасетов, которые используются в сфере больших данных, в том числе для машинного обучения и для обучения нейронных сетей.

Набор состоит из 6743 изображений. Информация по каждому классу представлена в табл. 1.

Таблица 1 – Исходные данные

| Номер набора | Количество изображений | Патология |
|---|---|---|
| 0 | 1340 | Здоровые лёгкие |
| 1 | 1060 | Пневмония |
| 2 | 678 | Повышенная плотность |
| 3 | 629 | Пониженная плотность |
| 4 | 644 | Обструкция лёгких |
| 5 | 594 | Инфекционные заболевания |
| 6 | 658 | Физические травмы |
| 7 | 596 | Измененное средостение |
| 8 | 544 | Нарушение развития лёгких |

### 3.2 Построение нейронной сети для классификации рентген-снимков

Для решения задачи классификации изображений была построена СНС со следующей архитектурой:

1. Первый слой
    a. Conv2D(128, (3, 3), Activation('relu'))
    b. MaxPooling(2,2)
    c. Conv2D(128, (3, 3), Activation('relu'))
    d. MaxPooling(2,2)
    e. Dropout(0.1)
2. Второй слой
    a. Conv2D(256, (3, 3), Activation('relu'))
    b. MaxPooling(2,2)
    c. Conv2D(256, (3, 3), Activation('relu'))
    d. MaxPooling(2,2)
    e. Dropout(0.1)
3. Третий слой
    a. Conv2D(512, (3, 3), Activation('relu'))
    b. MaxPooling(2,2)
    c. Conv2D(512, (3, 3), Activation('relu'))
    d. MaxPooling(2,2)
    e. Dropout(0.1)
4. Четвертый слой
    a. Flatten
    b. Dense(128, Activation('relu'))
    c. Dropout(0.1)
5. Пятый слой
    a. Dense(256, Activation('relu'))
6. Шестой слой

Dense(9, Activation('softmax'))

Для удобства, комбинация из пяти слоев – «свертка – пулинг – свертка – пулинг – дропаут» – называется одним сверточным слоем.

Первые три сверточных слоя составляют карту признаков, выбирают элементы с максимальным значением и уменьшают количество входных данных для следующих слоев.

В первом слое каждая свертка имеет по 128 ядер размерности 3x3. Получившаяся в результате работы свертки карта признаков сразу подается на слой активации ReLU. После каждой свертки производится операция пулинга размерности 2x2. В конце слоя используется дропаут – 10 процентов всех ядер свертки «отключается», чтобы избежать переобучения. Сверточные нейронные сети несильно подвержены переобучению, поэтому для операций дропаут были выбраны небольшие значения. Второй и третий слои построены таким же образом, но в каждом последующем слое количество ядер свертки увеличивается вдвое.

На четвертом слое получившаяся матрица выпрямляется в вектор, который подается на следующие слои. На выходном слое используется функция активации «softmax».

Следующий этап – непосредственно, обучение модели. Параметры обучения: функция ошибки – категориальная кроссэнтропия, метод оптимизации – Nadam, метрика качества обучения – Accuracy, количество эпох – 30, размер подвыборки – 100 изображений. На валидацию выделено 20% из тренировочного набора.

### 3.3 Результаты обучения нейронной сети

В результате обучения точность модели по метрике accuracy достигла 95.18%, что в целом является приемлемым результатом. Историю обучения можно оценить на рис. 8.

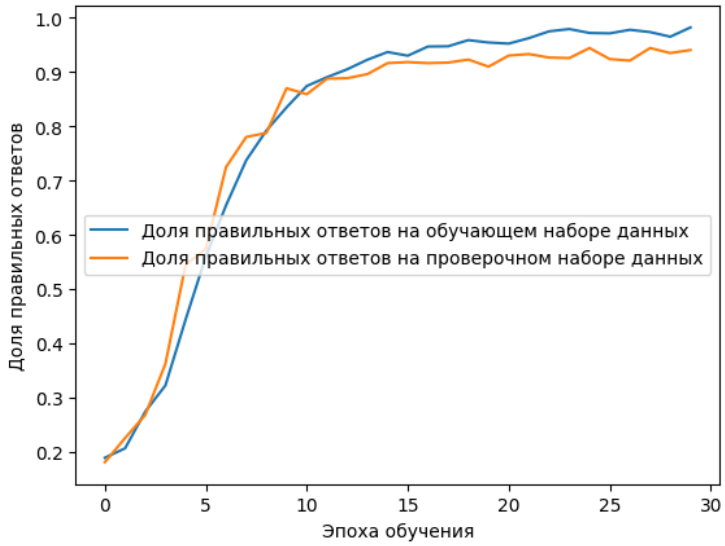



Рисунок 8 – История обучения нейросетевой модели

В большинстве случаев, однако, одной метрики accuracy недостаточно для качественной оценки обученной модели. Для полноценного анализа стоит, вместе с accuracy, использовать метрики precision и recall, и таблицу ошибок [13].

Метрика качества precision показывает для отдельно взятого класса, сколько объектов, отнесенных моделью к этому классу, реально относятся к этому классу.

Метрика качества recall показывает для отдельно взятого класса, какую долю из всей реальной совокупности объектов этого класса модель смогла отнести к этому классу.

Таблица ошибок показывает отношение правильных ответов к предсказанным моделью по всем классам.

Посмотрим на таблицу ошибок (confusion matrix) на рис. 9. На изображении приняты следующие обозначения: ВП – высокая (повышенная) плотность, НП – низкая (пониженная) плотность.

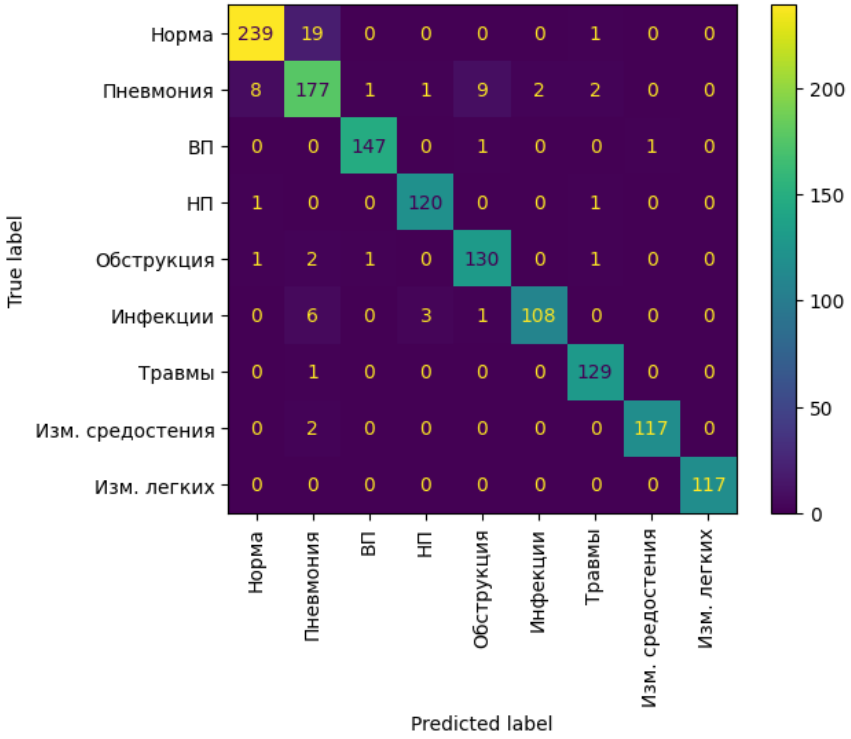



Рисунок 9 – Confusion matrix для обученной модели на тестовом наборе

Несмотря на приемлемый результат в целом, стоит обратить внимание на первый столбец. Модель определила восемь рентгеновских изображений легких с пневмонией, одно изображение легких с пониженной плотность и одно изображение легких с обструкцией, как здоровые легкие. Это небольшая ошибка в общем случае, но в сфере медицины такие ошибки, когда патологии помечаются как норма, являются критическими. Обратная ситуация так же нежелательна, но приносит меньше ущерба.

На рис. 10 представлен классификационный отчет (classification report). Видно, что изменения легких классифицируются с точностью в 100%. Возможными причинами могут быть:

1) резкое внешнее различие с другими классами;

2) недостаток разнообразных изображений с измененными легкими.

По тем же причинам можно объяснить высокую точность у классов «ВП», «Травмы» и «Изм. средостения».

Хуже всего модель определяет класс «Пневмония». 14% всех изображений, отмеченных нейросетью как «Пневмония», не являются таковыми. Также, из всех изображений с пневмонией, модель не отметила 11%.

Учитывая эти факты, можно говорить о том, что реальное качество модели немного ниже, чем показывает метрика accuracy.

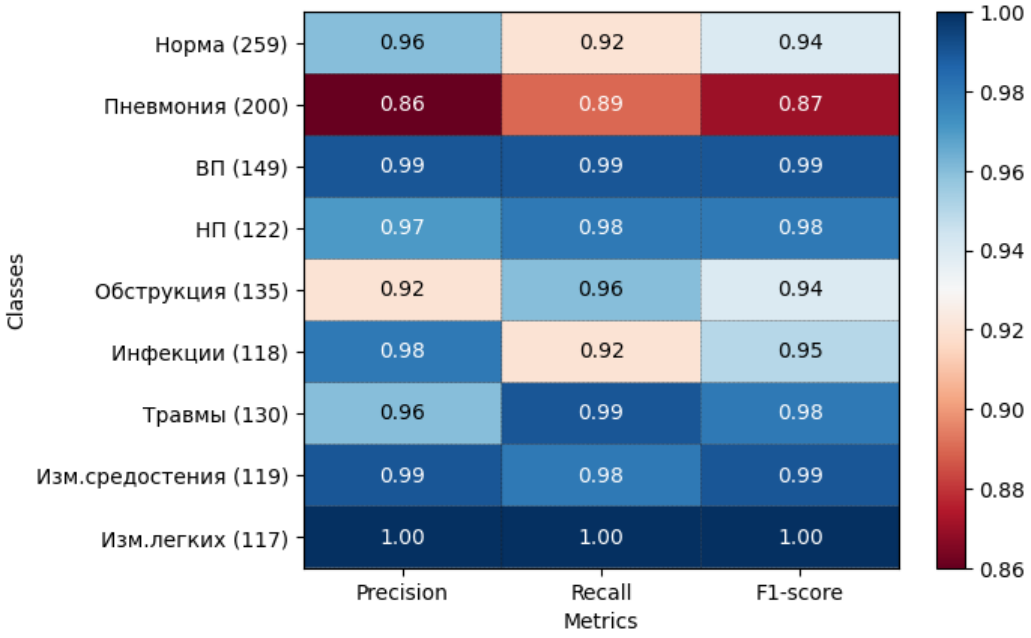



Рисунок 10 – Classification report для обученной модели на тестовом наборе

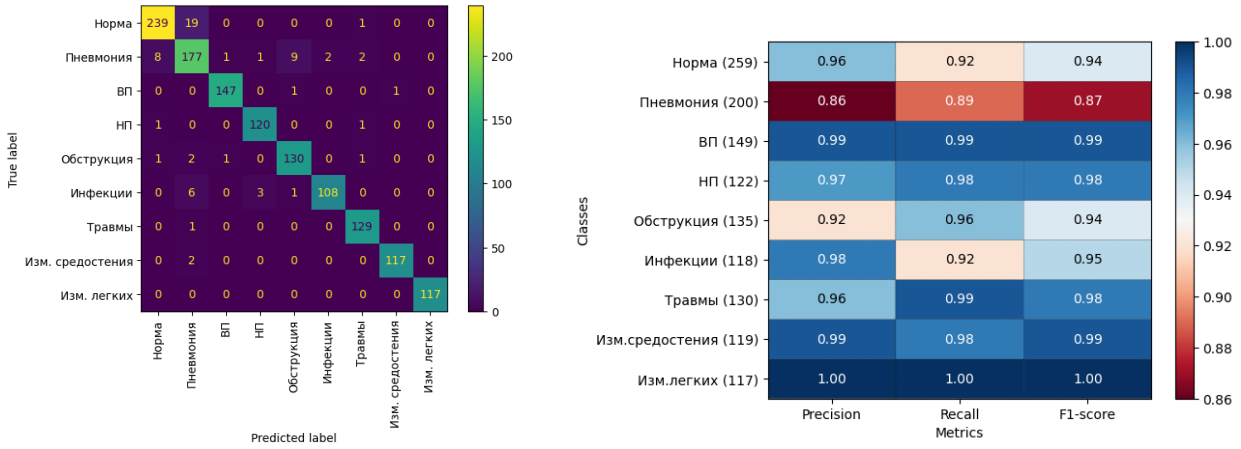
### 3.4 Обучение нейронной сети ResNet50 для классификации рентгеновских снимков

Для сравнения, на данном наборе была обучена нейронная сеть с архитектурой ResNet50.

Это известная нейронная сеть, созданная компанией Microsoft. Её главная особенность в том, что на каждый слой подается не только обработанные сверткой данные, но и изначальные, «чистые». Таким образом, уменьшается ошибка обучения в глубоких нейронных сетях.

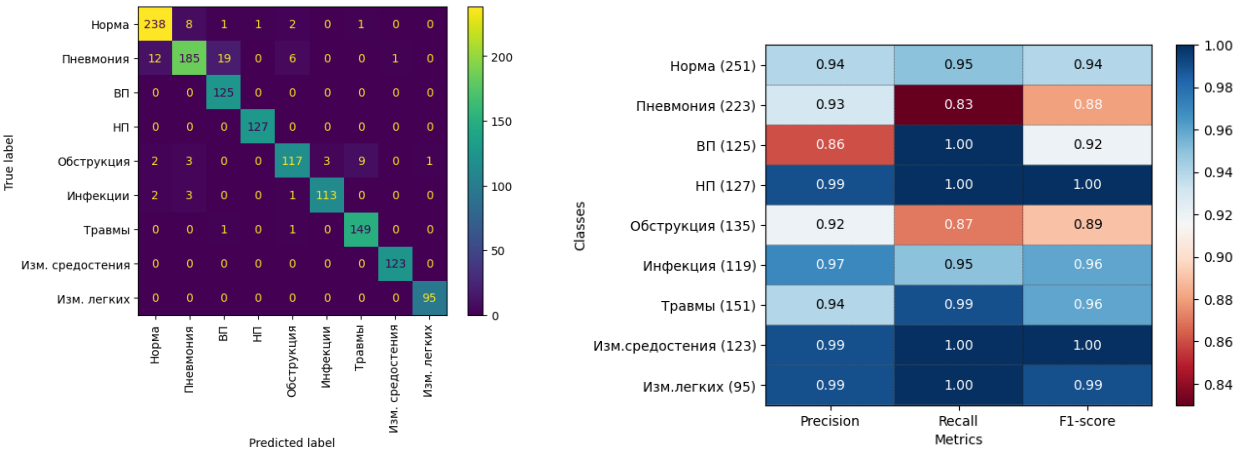
Для сравнения результатов была обучена нейронная архитектуры ResNet50 [14]. В сравнении с построенной нейросетью, ResNet50 показала неоднозначные результаты. Главное отличие в том, что во время обучения точность

модели резко менялась от эпохи к эпохе. При первой попытке обучения при 30 эпохах на предпоследней эпохе точность ResNet50 дошла до 91%, но на последней эпохе точность резко упала до 20%. Поэтому во второй раз количество эпох для обучения модели было уменьшено до 26. На финальной эпохе точность ResNet50 была незначительно хуже, чем у построенной СНС – 94% по метрике accuracy, но на некоторых эпохах точность доходила до 96%. Результаты обучения нейронной сети ResNet50 для данного набора при 26 эпохах представлены на рис. 12




*а* *б*

Рисунок 11 – Confusion matrix (*а*) и Classification report (*б*) для обученной СНС на тестовом наборе




*а* *б*

Рисунок 12 – Confusion matrix (*а*) и Classification report (*б*) для нейронной сети ResNet50 на тестовом наборе

Таким образом, выбранная архитектура СНС показала лучшие результаты в сравнении с нейронной сетью ResNet50.

# 4 РАЗРАБОТКА ЧАТ-БОТА ДЛЯ ДИАГНОСТИКИ ЗАБОЛЕВАНИЙ ЛЕГКИХ НА ОСНОВЕ НЕЙРОННОЙ СЕТИ

## 4.1 Обзор сервисов и платформ для создания чат-ботов

Обученную сеть было решено интегрировать в интерактивный бот. По сравнению с другими приложениями, такой метод имеет несколько преимуществ:

1. Простота создания: для создания рабочего бота не требуются дорогое оборудование или специальные умения – только базовые навыки программирования. Такой вариант подходит для как предварительной демонстрации работы нейронной сети, так и, при дальнейших улучшениях, для полноценного использования в медицинской сфере.

2. Простота эксплуатации: в отличие от существующих решений, требующих сложных установок и специализированного ПО, бот предлагает максимально простой и доступный интерфейс через любой смартфон с выходом в интернет.

3. Легкость тестирования: учитывая перечисленные выше преимущества, гораздо легче проверить нейронную сеть на адекватную работу вне google colab.

В настоящее время, многие сервисы предлагают свои решения по написанию и эксплуатации ботов. Наиболее популярные платформы:

1. Telegram Bot API. Telegram предоставляет мощный интерфейс для создания ботов. API поддерживает отправку и приём сообщений, обработку мультимедиа, клавиатур и команд. Платформа бесплатна, что делает её популярным выбором среди разработчиков [15].

2. Dialogflow. Инструмент от Google для создания чат-ботов с использованием методов обработки естественного языка. Подходит для сложных диалоговых сценариев, но требует дополнительных затрат на интеграцию с внешними нейронными сетями.

3. Microsoft Bot Framework. Платформа с широкими возможностями для создания ботов с интеграцией в различные мессенджеры. Поддерживает

использование машинного обучения, но может быть сложной в освоении для простых решений.

4. Botpress. Открытая платформа для создания ботов с визуальным редактором. Удобна для быстрой настройки, однако требует дополнительных усилий для интеграции с кастомными алгоритмами машинного обучения.

Для разработки бота в рамках данного проекта была выбрана платформа Telegram Bot API. Она обладает всеми необходимыми функциями, предоставляет простой доступ к инфраструктуре и позволяет легко интегрировать пользовательские нейронные сети.

### 4.2 Проектирование архитектуры чат-бота, разработка функционала и сценариев диалога

Общий для всех телеграмм-ботов принцип работы достаточно прост – через Telegram bot API. Любые сообщения, написанные в диалоговом окне бота в мессенджере телеграмм, отправляются на сервера телеграмма, далее перенаправляются на физический сервер (компьютер), на котором запущен телеграмм бот. Сообщение обрабатывается, а потом, по тому же маршруту отправляется ответ на сообщение [16]. В упрощённом виде, данную архитектуру можно посмотреть на рис. 13.

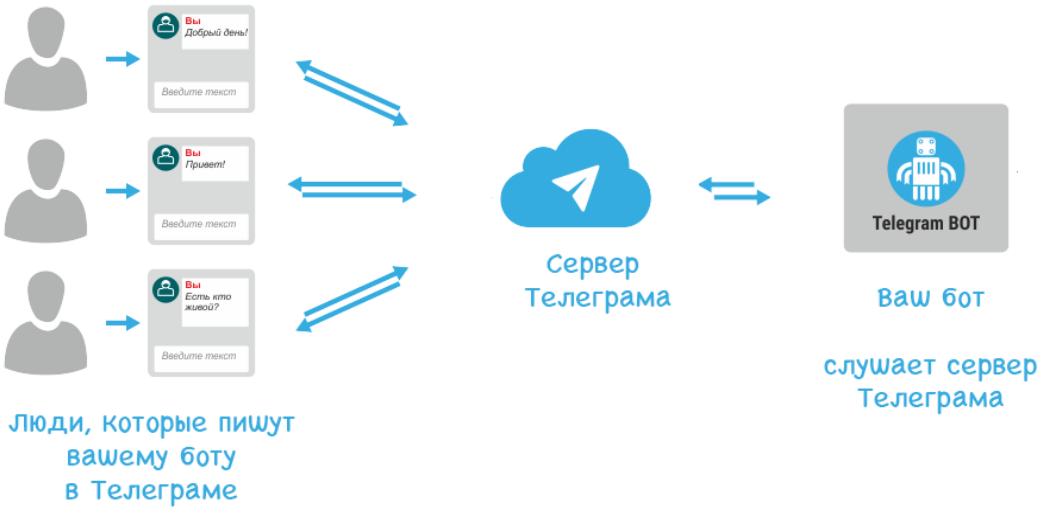



Рисунок 13 – Архитектура телеграмм-ботов

Телеграм боты могут принимать разные виды информации – текст, изображения, видео и аудио файлы, и при различных условиях, по-разному на них отвечать.

В нашем случае, основная функция телеграм бота – анализ изображений рентгеновских снимков легких. Поэтому, прописывание различных вариантов диалогов будет излишним. Написание подробной инструкции по работе с ботом также не требуется – действия пользователя ограничиваются отправкой изображений.

Приложение будет работать следующим образом: пользователю, при первом взаимодействии, будет отправлено короткое сообщение с просьбой прислать изображение легких для анализа. При отправке изображения, в ответ пользователь получит сообщение с классом заболевания.

### 4.3 Интеграция нейронной сети с чат-ботом

Для интеграции с чат-ботом, достаточно предоставить боту доступ к нейронной сети, например, загрузить файл нейронной сети на сервер телеграм бота. Далее в коде достаточно прописать путь к данному файлу для возможности загрузки нейронной сети в оперативную память. Изменение остальной части программного кода не требуется – в ней прописана работа с обменом сообщениями между пользователем и чат-ботом.

Прием сообщений будет происходить в 4 этапа:

1 Проверка типа сообщения. Есть три варианта событий:

a. Сообщение является текстом. Тогда в ответ пользователю отправляется короткое сообщение-просьба отправить изображение рентген снимка легких.

b. Сообщение не является текстом или изображением. Тогда сообщение игнорируется.

c. Сообщение является изображением. Тогда прием сообщения переходит на второй шаг.

2 Обработка изображения перед передачей его в нейронную сеть. Необходимо привести его к нужному формату, который удобен для нейронной

сети, а так же привести все пиксели к значениям от 0 до 1, иначе качество классификации упадет до нуля.

3 Передача обработанного изображения в нейронную сеть для анализа и классификации.

4 Отправка пользователю сообщения с информацией о классе заболевания (либо сообщение о здоровых легких).

### 4.4 Файлы DICOM

В рамках данной работы была также реализована поддержка формата DICOM (Digital Imaging and Communications in Medicine) — стандарта, широко используемого в медицинских учреждениях для хранения и передачи медицинских изображений. Это позволило расширить функциональность системы, обеспечив совместимость с данными, поступающими напрямую из диагностического оборудования, такого как рентген-аппараты и компьютерные томографы.

Для реализации поддержки формата DICOM использовалась библиотека pydicom, позволяющая извлекать изображение, преобразовывать его в различные форматы, а также работать с различной метаинформацией (например, сведения о пациенте, дате исследования, параметрах съёмки). Изображения проходили этап предварительной обработки: извлечение пиксельных данных, нормализация, а при необходимости — преобразование в стандартный графический формат для подачи на вход нейросети.

Добавление поддержки DICOM-файлов значительно увеличивает прикладную ценность разработанного решения, так как позволяет врачам и исследователям работать напрямую с оригинальными медицинскими снимками без необходимости предварительной конвертации. Это делает систему более универсальной и удобной для интеграции в существующие клинические процессы.

Одним из ключевых этапов обработки DICOM-файлов является удаление или анонимизация метаинформации, содержащей персональные данные, такие как имя пациента, дата рождения, идентификатор учреждения, дата

исследования и т. д. Для этого используются встроенные инструменты DICOM-библиотек, которые позволяют как полностью удалять эти поля, так и заменять их фиктивными значениями при необходимости.

Кроме того, при передаче изображений на обработку через телеграм-бот реализован механизм временного хранения данных с последующим автоматическим удалением. Это снижает риски несанкционированного доступа к информации и минимизирует время, в течение которого данные находятся в системе.

Таким образом, при разработке и эксплуатации программного обеспечения был учтён критически важный аспект — защита персональных медицинских данных.

### 4.5 Тестирование и результаты

Для проверки работы телеграм-бота было проведено тестирование.

Чат-бот корректно обрабатывает запросы. Взаимодействие между ботом и нейронной сетью происходит по задуманному сценарию. Все запросы и ответы обрабатываются, с учетом задержек сети, в пределах пяти секунд. Пример работы телеграм бота можно увидеть на рис. 14.

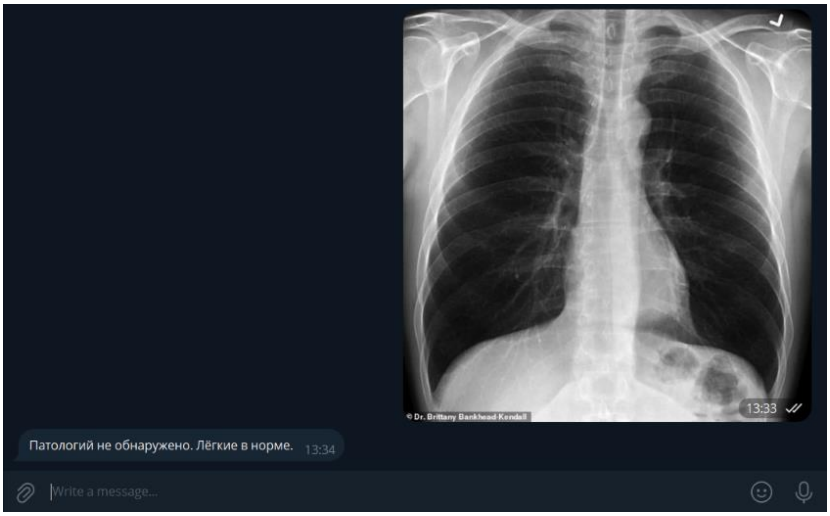



Рисунок 14 – Пример распознавания изображений в чат-боте

В результате, телеграм бот выполнил все функции с ожидаемым результатом. Точность классификации осталась такой же, как и при

тестировании вне чат-бота. Среднее время обработки запроса от загрузки снимка до получения результата составило 3,2 секунды.

Также, был замечен ряд недостатков:

− Строгие правила для изображения. Качество изображения рентген снимка должно быть приближено к идеальному для высокой точности классификации. При фотографировании обычным способом – например, с камеры смартфона – блики, попадание фона в кадр сильно влияют на качество классификации. В связи с данной проблемой, планируется добавить возможность анализа файлов в формате dicom.

− Необходимость выделенного сервера. Без него телеграм бот не может работать круглосуточно.

# 5 СОЗДАНИЕ ТЕПЛОВЫХ КАРТ ДЛЯ ВИЗУАЛЬНОГО АНАЛИЗА СВЁРТОЧНЫХ СЛОЁВ

## 5.1 Теория Grad-CAM

Метод Grad-CAM (Gradient-weighted Class Activation Mapping), предложенный в работе Selvaraju et al. (2016) [21], представляет собой визуализационный подход, позволяющий интерпретировать решения сверточных нейронных сетей (CNN) путём построения тепловых карт (heatmaps), показывающих, какие области входного изображения оказали наибольшее влияние на предсказание модели. На рисунке 15 можно увидеть пример из оригинальной статьи, с классификацией кошек и собак.

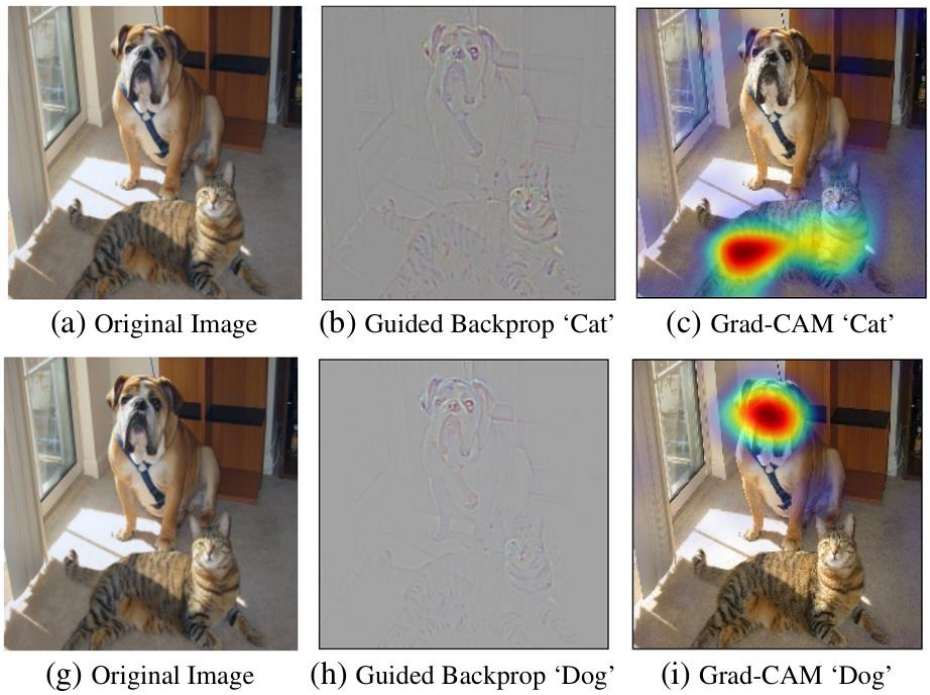



Рисунок 15 – Grad-CAM на изображениях животных

До Grad-CAM существовал CAM. Методы Class Activation Mapping (CAM) и Gradient-weighted Class Activation Mapping (Grad-CAM) направлены на визуализацию, какие области изображения оказывают наибольшее влияние на предсказание сверточной нейронной сети. Несмотря на общую цель, между ними существуют ключевые различия в применимости и гибкости.

CAM (Class Activation Mapping), предложенный Zhou et al. в 2015 году, использует структуру сети, в которой последний сверточный слой напрямую связан с выходным полносвязным слоем через Global Average Pooling (GAP). Это позволяет получить «карту активации» для каждого класса, вычисляя взвешенную сумму признаков последнего сверточного слоя. Однако главный недостаток CAM – ограниченность архитектуры: он требует явного присутствия GAP и линейного слоя классификации, что делает невозможным его применение к произвольным предобученным сетям.

Grad-CAM, представленный Selvaraju et al. в 2016 году, решает эту проблему, вычисляя градиенты выхода по активациям заданного сверточного слоя. Полученные градиенты усредняются по пространственным координатам и используются как веса для активаций, тем самым формируя тепловую карту, релевантную конкретному классу. Это делает Grad-CAM универсальным методом, который можно применять к большинству современных моделей, включая те, где структура сети не предполагает использование GAP. В целом, Grad-CAM является обобщением идеи CAM, сохраняющим интерпретируемость, но снимающим архитектурные ограничения.

Grad-CAM позволяет локализовать визуальные признаки, связанные с определенным классом, используя информацию о градиентах, распространяющихся от выходного слоя к последнему сверточному слою. Основное преимущество метода заключается в его универсальности: он применим к любой архитектуре, включающей сверточные слои и обученную с помощью градиентного спуска.

Grad-CAM строится на следующих основных этапах:

1. Прямое распространение (forward pass):

Входное изображение $I$ пропускается через модель, и фиксируется выход последнего сверточного слоя $A^k$, где $k$ – индекс карты признаков (feature map).

2. Обратное распространение (backward pass):

Вычисляются градиенты выходного значения для интересующего класса по отношению к каждой карте признаков: $\frac{\partial y^c}{\partial A^k}$ Эти градиенты измеряют важность каждой карты признаков $A^k$ для класса с.

3. Получение весов важности:

Глобальное усреднение градиентов по ширине и высоте карты признаков: $a_k^c = \frac{1}{Z}\sum_i\sum_j \frac{\partial y^c}{\partial A_{ij}^k}$, где $Z$ – общее количество элементов в каждой карте признаков.

4. Линейная комбинация карт признаков:

Весовые коэффициенты $a_k^c$ используются для взвешенного суммирования карт признаков: $L_{Grad\text{-}CAM}^c = ReLU(\sum_k a_k^c A^k)$

Функция ReLU применяется для сохранения только положительных влияний, которые усиливают предсказание класса.

5. Масштабирование карты на размер входного изображения:

Полученная тепловая карта интерполируется до размеров исходного изображения и визуализируется в виде наложения цветовой маски.

Тепловая карта Grad-CAM показывает, какие участки изображения активировали определенные карты признаков, которые в наибольшей степени повлияли на классификацию в пользу конкретного класса. Яркие области (красные/оранжевые) на тепловой карте указывают на зоны, наиболее важные для принятия решения, в то время как темные (синие) зоны считаются менее значимыми.

Метод Grad-CAM широко применяется для:

- Визуализации и интерпретации работы сверточных моделей.
- Обнаружения смещений и ошибок модели.
- Объяснения решений в медицинской диагностике, автономном вождении и других критичных системах.
- Поддержки процесса explainable AI (XAI) — объяснимого искусственного интеллекта.

При распознавании патологий на медицинских изображениях (например, рентген-снимках лёгких), Grad-CAM позволяет определить, действительно ли модель фокусируется на анатомически значимых областях (например, лёгочных полях) или принимает решение на основе посторонних признаков (например, надписей или рамок).

### 5.2 Тепловые карты ренгеновских снимков

В рамках исследования были построены тепловые карты на различных уровнях свёрточной нейронной сети для интерпретации работы модели. Для работы были написаны функции на языке Python с помощью библиотеки Tensorflow. GradientTape отслеживает автоматическое дифференцирование, благодаря которому можно вычислять градиенты. Программный код можно увидеть в приложении «Д». Результаты оказались неоднозначными.

Рассмотрим получившиеся тепловые карты на изображении с классом «пневмония».

Для начала, рассмотрим тепловую карту первого слоя на рис. 16.

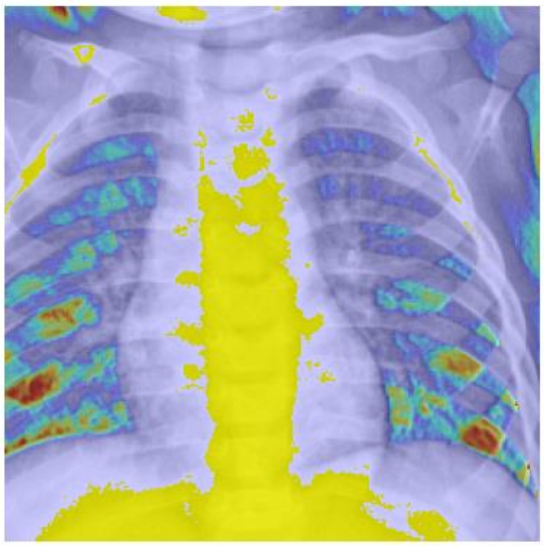


Рисунок 16 – Тепловая карта первого слоя, пневмония

«Горячие» области расположены в районе легочных тканей. Эти области нейросеть определяет как важные для классификации, и, по всей видимости, похожие области – по структуре, форме или расположению – находятся в

большинстве изображений с пневмонией. Жёлтая область – менее важная – выделяет позвоночник. Неясно, какое влияние оказывает вид позвоночник на определение класса «пневмония», но, возможно, таким образом нейронная сеть «помечает» область, которая сильно отличается для другого класса, например, класс под номером восемь – «Пороки развития», в котором часто форма позвоночника сильно отличается от остальных случаев. И, таким образом, нейросеть сразу отсекает возможность классификации изображения как «Пороки развития».

В то же время пятый, последний слой (рис. 17) не показывает адекватной информации.

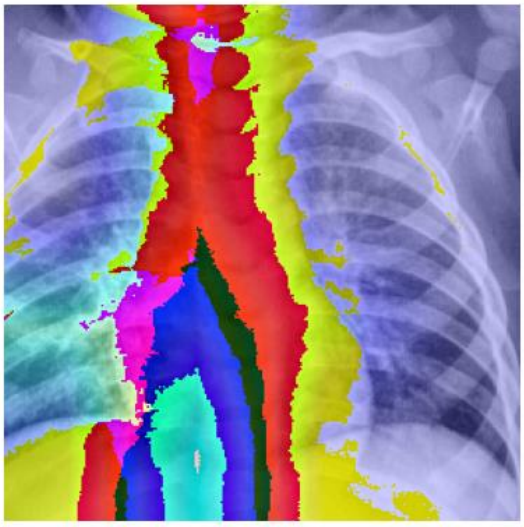


Рисунок 17 – Тепловая карта пятого слоя, пневмония

Рваные области низкого качества, которые показывают на случайные области. На поздних этапах (после многочисленных операций свёртки и пулинга) разрешение активационных карт значительно снижается. Это приводит к тому, что тепловая карта становится грубее и теряет детальность: она уже слабо "видит" отдельные анатомические структуры.

Для лучшего понимания, были также рассмотрены рентгеновские снимки лёгких здорового человека. На рис. 18 можно увидеть тепловую карту первого слоя.

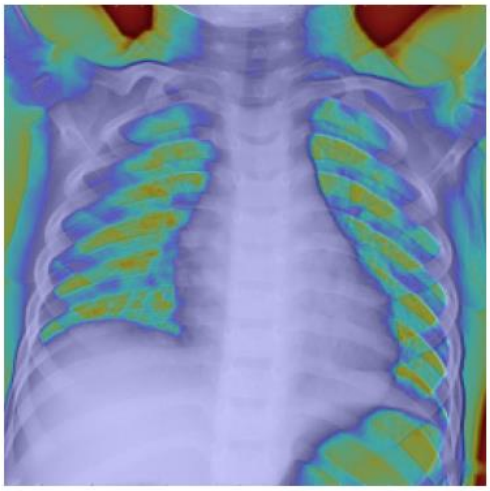


Рисунок 18 – Тепловая карта первого слоя, норма

На их примере можно убедиться, что нейронная сеть может обращать внимание на ненужные элементы в изображениях. На рис. 18 «горячие» области находятся вне тела, выше плеч. Это указывает на то, что большая часть сохранённых изображений здоровых лёгких содержит большую часть туловища и некоторую область вокруг. Именно это нейросеть посчитала хорошим признаком при определении класса «здоровые лёгкие».

Хотя в целом это указывает на плохую обобщающую способность, в то же время во втором слое ситуация немного другая.

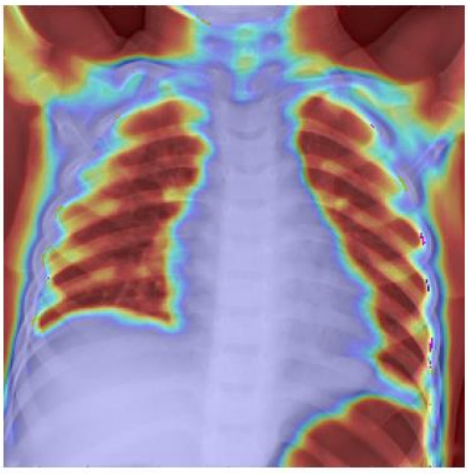


Рисунок 19 – Тепловая карта второго слоя, норма

В данном случае, даже учитывая  красные области вне тела, заметно, что учитывается также и состояние самих лёгких. Большая часть «горячих» областей сосредоточена в районе лёгочных тканей. Как раз там, где и и должно быть сосредоточено внимание.

Вывод: Несмотря на некоторые признаки, указывающие не нелучшую обобщающую способность обученной модели, результаты можно охарактеризовать как положительные. Разные слои выделяют разные признаки, большая часть слоёв вносит свой вклад в определение класса изображения. Кроме того, высокие оценки всех метрик (accuracy, precision, recall) говорят о достаточных способностях модели к обучению при достаточном количестве качественных данных. Скорее всего, главной проблемой слабой обобщающей способности являются следующие причины:

1. Изначально плохое качество датасета. Шесть тысяч изображений позволяют проверить начальные гипотезы, но для обучения рабочей модели, пригодной для использования в жизни, требуется гораздо больше.

2. Ограничение по мощностям. Для обучения модели использовался Google colab. Идеальный вариант для написания учебных проектов (как  этот), но достаточно слабый для обучения крупных моделей.

Из существенных проблем архитектуры стоит отметить излишнюю глубину. Но это относится только к данному набору изображений. При увеличении количества рентгеновских снимков, последний слой может показать лучшие результаты.

Заключение: Несмотря на странности в тепловых картах, анализ показал, что модель показала хорошие результаты для исходного набора данных; разные слои могут выделять разные признаки, и с получившейся архитектурой, при наличии достаточного количества хороших данных и высоких мощностей, в теории, можно обучить модель высокого качества.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения данной магистерской диссертации была разработана и обучена свёрточная нейронная сеть, предназначенная для автоматической диагностики заболеваний лёгких на основе рентгеновских снимков. Разработанная модель показала высокую точность классификации – 95.18%, что свидетельствует о её потенциале для применения в реальных клинических задачах.

На начальном этапе работы был проведён теоретический обзор в области искусственного интеллекта, машинного обучения и нейронных сетей, а также рассмотрены основные метрики оценки качества моделей. Это позволило заложить теоретическую основу для дальнейших практических разработок.

Особое внимание было уделено анализу существующих архитектур свёрточных нейронных сетей, использующихся для анализа изображений. Также была описана процедура подготовки медицинских изображений и особенностей работы с медицинскими данными, включая нормализацию, увеличение выборки и разметку данных.

В практической части была реализована собственная архитектура нейронной сети, адаптированная под задачу классификации рентгеновских снимков лёгких. Модель прошла этапы обучения, валидации и тестирования, продемонстрировав высокую эффективность.

Для удобства взаимодействия с моделью был создан телеграм-бот, позволяющий пользователю загружать изображения и получать предварительный диагноз в автоматическом режиме. Это делает разработанное решение более доступным и практичным для использования, в том числе в условиях телемедицины и удалённой диагностики.

Таким образом, поставленные в работе цели были достигнуты, а полученные результаты подтверждают актуальность и перспективность применения технологий искусственного интеллекта в области медицины и диагностики.

# БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Искусственный интеллект. Инноватика: учеб. Пособие / Ю. А. Антохина [и др.] – Санкт-Петербург: ГУАП, 2023. – 320 с. – Режим доступа: https://e.lanbook.com/book/341003

2 Остроух, А.В. Системы искусственного интеллекта: моногр. / А. В. Остроух, Н. Е. Суркова. – 3-е изд., стер. – Санкт-Петербург: Лань, 2023. – 228 с. – Режим доступа: https://e.lanbook.com/book/310199

3 Толмачев, С.Г. Нейросетевые методы обработки информации: учебное пособие / С.Г. Толмачев. – Санкт-Петербург: БГТУ "Военмех" им. Д.Ф. Устинова, 2021. – 103 с. – Режим доступа: https://e.lanbook.com/book/220238

4 habr.com: Возможности Искусственного Интеллекта в 2023 году. Эндрю Ын [Электронный ресурс]: офиц. сайт. – 26.05.2006 – Режим доступа: https://habr.com/ru/companies/serverspace/articles/776954/ (дата обращения: 03.12.2023)

5 habr.com: Нейросеть – обучение без учителя. Метод Policy Gradient [Электронный ресурс]: офиц. сайт. – 26.05.2006 – Режим доступа: https://habr.com/ru/articles/506384/ (дата обращения: 03.12.2023)

6 habr.com: Искусственный интеллект в медицине: сферы, технологии и перспективы [Электронный ресурс]: офиц. сайт. – 26.05.2006 – Режим доступа: https://habr.com/ru/companies/first/articles/682516/ (дата обращения: 03.12.2023)

7 habr.com: Что делает ChatGPT… и почему это работает? [Электронный ресурс]: офиц. сайт. – 26.05.2006 – Режим доступа: https://habr.com/ru/articles/739014/ (дата обращения: 03.12.2023)

8 Бурцева, Е.В. Интеллектуальные информационные системы: учебное пособие / Е. В. Бурцева, А. В. Платёнкин, И. П. Рак. – Тамбов: ТГТУ, 2022. – 80 с. – Режим доступа: https://e.lanbook.com/book/355139 (дата обращения: 17.12.2023)

9 Постолит, А.В. Основы искусственного интеллекта в примерах на Python: самоучитель / А. В. Постолит. – СПб.: БХВ-Петербург, 2021. – 448 с.

10 Fashion MNIST. An MNIST-like dataset of 70,000 28x28 labeled fashion images. – Режим доступа: https://www.kaggle.com/datasets/zalando-research/fashionmnist (дата обращения: 17.12.2023)

11 Соробин, А.Б. Сверточные нейронные сети: примеры реализаций: учебно-методическое пособие / А.Б. Соробин. — Москва: РТУ МИРЭА, 2020. – 159 с. – Текст: электронный // Лань: электронно-библиотечная система. – Режим доступа: https://e.lanbook.com/book/163853 (дата обращения: 20.05.2024)

12 Гудфеллоу, Я. Глубокое обучение // Я. Гудфеллоу, И. Бенджио, А. Курвилль. Перевод с английского А.А. Слинкина. – 2-е изд. – М: ДМК Пресс, 2018. – 652 с.

13 Берман, К. Основы Python для Data Science / К. Берман. — СПб.: Питер, 2023. — 272 с.

14 resnet50 [Электронный ресурс]. Режим доступа: https://pytorch.org/vision/main/models/generated/torchvision.models.resnet50.html#resnet50 (дата обращения: 30.04.2024)

15 telegram.org: Telegram API [Электронный ресурс]: офиц. сайт – 13.08.2013 – Режим доступа: https://core.telegram.org/bots/api (дата обращения: 05.12.2024)

16 Nicolas Modrzyk. Building Telegram Bots: Develop Bots in 12 Programming Languages using the Telegram Bot API / Nicolas Modrzyk. – Apress, 2019. – 232 p.

17 O'Shea K., Nash R. An Introduction to Convolutional Neural Networks [Электронный ресурс] // arXiv preprint arXiv:1511.08458. — 2015. — 11 с. — Режим доступа: https://arxiv.org/abs/1511.08458

18 Deep Convolutional Neural Network for The Prognosis of Diabetic Retinopathy / Shanthini, A.; Manogaran, Gunasekaran; Vadivu, G. - Springer, 2023

19 Review of deep learning: concepts, CNN architectures, challenges, applications, future directions / L. Alzubaidi, J. Zhang, A. J. Humaidi [et al.] // Journal of Big Data. – 2021.

20 Going deeper with convolutions / C. Szegedy, Y. Jia, P. Sermanet [et al.] // Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Boston, MA, 07–12 июня 2015 года. – Boston, MA, 2015. – P. 1-9.

21 Selvaraju R. R., Cogswell M., Das A., Vedantam R., Parikh D., Batra D. Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization // Proceedings of the IEEE International Conference on Computer Vision (ICCV). — 2017. — С. 618–626.

**Публикации автора по теме работы**

22 Дудич, М.О. Свёрточная нейронная сеть для диагностики заболеваний легких / М.О. Дудич, Н.Н. Максимова // Математическое и компьютерное моделирование: сборник материалов XI Международной научной конференции, посвященной памяти В.А. Романькова (Омск, 15 марта 2024 г.) / [отв. за вып. И.П. Бесценный]. – Омск: Издательство Омского государственного университета, 2024. – С. 197-199.

23 Дудич, М. О. Свёрточные нейронные сети в задачах диагностики заболеваний легких / М. О. Дудич, Н. Н. Максимова // ТОГУ-Старт: фундаментальные и прикладные исследования молодых: Материалы V региональной научно-практической конференции, Хабаровск, 16–20 апреля 2024 года. – Хабаровск: Тихоокеанский государственный университет, 2024. – С. 138-144.

24 Дудич, М. О. Нейросетевой подход в задачах диагностики заболеваний легочной системы / М. О. Дудич // День науки: Материалы XXXIII научной конференции Амурского государственного университета, Благовещенск, 18 апреля 2024 года. – Благовещенск: Амурский государственный университет, 2024. – С. 69-70.

25 Дудич, М.О. Чат-бот на основе нейронной сети для определения заболевания легких / М.О. Дудич // Материалы международной научной конференции «Актуальные проблемы прикладной математики, информатики и механики» (Воронеж, 2-4 декабря 2024 года). – Воронеж: издательство Воронежского государственного университета. – *в печати*

# ПРИЛОЖЕНИЕ А
## Текст ноутбука «Recognition of clothing items.ipynb»

Импорт библиотек:

```
from tensorflow.keras.datasets import fashion_mnist      # Импорт из наборов данных tensorflow.keras набора fashion_mnist
from tensorflow.keras.models import Sequential           # Импорт последовательной модели из библиотеки моделей
from tensorflow.keras.layers import Dense, Dropout      # Импорт полносвязного типа и отсеивающего типа из списка слоев
from tensorflow.keras import utils                       # Импорт утилит
from tensorflow.keras.preprocessing import image         # Импорт модуля для предобработки изображений
import numpy as np                                      # Импорт модуля numpy, который предоставляет общие математические и числовые операции
import matplotlib.pyplot as plt                                  # Импорт модуля matplotlib (основная библиотека для построения научных графиков в Python)
from PIL import Image
%matplotlib inline
#Для получения статичного изображения
```

Загружаем набор данных:

```
(x_train, y_train), (x_test, y_test) = fashion_mnist.load_data() #загружаем эти данные в два массива: x - фотографии, y - правильные ответы
```

Преобразование размерности данных в наборе:

```
x_train = x_train.reshape(60000, 784)
x_test = x_test.reshape(10000, 784)
```

Нормализация данных:

```
x_train = x_train / 255
x_test = x_test / 255
```

Преобразуем метки в формат one hot encoding:

```
y_train = utils.to_categorical(y_train, 10)    # Преобразует вектор класса (целые числа) в двоичную матрицу классов.
y_test = utils.to_categorical(y_test, 10)
```

Описываем архитектуру нейронной сети:

```
# Создаем последовательную модель
model = Sequential()
# Входной полносвязный слой, 200 нейронов, 784 входа в каждый нейрон, функция активации - relu
model.add(Dense(200, input_dim = 784, activation = "relu"))
# Выходной полносвязный слой, 10 нейронов (по количеству предметов одежды), функция активации - softmax
model.add(Dense(10, activation = "softmax"))
```

Компилируем сеть:

```
model.compile(loss = "categorical_crossentropy", optimizer = "adam", metrics = ["accuracy"])
```

```
print(model.summary())
```

Обучаем нейронную сеть:

```
history = model.fit(x_train, y_train,
                    batch_size = 200,
                    epochs = 10,
                    validation_split = 0.2,
                    verbose = 1)
```

Оцениваем качество построенной нейронной сети:

```
scores = model.evaluate(x_test, y_test, verbose = 1)
print("Доля правильных ответов на тестовых данных, в процентах:",
round(scores[1] * 100, 4))
```

Используем нейронную сеть для распознавания одежды:

```
n_rec = 2003 # выбираем изображение
plt.imshow(x_test[n_rec].reshape(28, 28), cmap=plt.cm.binary)
plt.show()
```

Выбираем размерность изображения и нормализуем его:

```
x = x_test[n_rec]
x = np.expand_dims(x, axis=0)
```

Запускаем распознавание:

```
predictions = model.predict(x)
```

Преобразуем результаты из формата one hot encoding:

```
prediction = np.argmax(predictions[0])
print("Номер класса:", prediction)
print("Название класса:", classes[prediction])
```

Печатаем правильный ответ:

```
label = np.argmax(y_test[n_rec])
print("Номер класса:", label)
print("Название класса:", classes[label])
```

ПРИЛОЖЕНИЕ Б
Текст ноутбука «Lungs.ipynb»

Импорт библиотек:

```
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten
from tensorflow.keras.layers import Conv2D, MaxPooling2D
from tensorflow.keras import utils                      # Импорт утилит
from tensorflow.keras.preprocessing import image         # Импорт модуля для
предобработки изображений
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from PIL import Image
import os
import cv2
import random
from tqdm import tqdm
from sklearn.model_selection import train_test_split
```

Подготовка данных:

```
DATADIR = "/content/"
CATEGORIES = ["00", "01", "02", "03", "04", "05", "06", "07","08"]
IMG_SIZE = 260
for category in CATEGORIES:
    path = os.path.join(DATADIR,category)
    for img in os.listdir(path):
       img_array = cv2.imread(os.path.join(path,img) ,cv2.IMREAD_GRAYSCALE)
        plt.imshow(img_array, cmap='gray')
        plt.show()

        break
    break
training_data = []

def create_training_data():
    for category in CATEGORIES:

        path = os.path.join(DATADIR,category)
        class_num = CATEGORIES.index(category)

        for img in tqdm(os.listdir(path)):
            try:
              img_array = cv2.imread(os.path.join(path,img) ,cv2.IMREAD_GRAY-
SCALE)
                new_array = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE))
                training_data.append([new_array, class_num])
```

```
            except Exception as e:
                pass

create_training_data()
random.shuffle(training_data)
X = []
y = []

for features,label in training_data:
    X.append(features)
    y.append(label)

X = np.array(X).reshape(-1, IMG_SIZE, IMG_SIZE, 1)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size= 0.2 ,
random_state=0 )
X_train = X_train / 255.0
X_test = X_test / 255.0
X_train = X = np.array(X_train).reshape(-1, IMG_SIZE, IMG_SIZE, 1)
X_test = X = np.array(X_test).reshape(-1, IMG_SIZE, IMG_SIZE, 1)
```

Описываем архитектуру нейронной сети:

```
model = Sequential()

model.add(Conv2D(128, (3, 3), input_shape=X_train.shape[1:]))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(128, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.10))
#####################################
model.add(Conv2D(256, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(256, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.10))
#####################################

model.add(Conv2D(512, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))


model.add(Conv2D(512, (3, 3)))
model.add(Activation('relu'))
```

```
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.10))
#####################################


model.add(Flatten())

model.add(Dense(128))
model.add(Activation('relu'))
model.add(Dropout(0.10))
model.add(Dense(256))
model.add(Activation('relu'))
model.add(Dense(9))
model.add(Activation('softmax'))
```

Компилируем нейронную сеть:

```
model.compile(loss = "categorical_crossentropy", optimizer = "nadam", metrics
= ["accuracy"])

print(model.summary())
```

Обучаем сеть:

```
history = model.fit(X_train, Y_train,
                    batch_size = 100,
                    epochs = 30,
                    validation_split = 0.2,
                    verbose = 1)
```

Проверяем качество по метрике accuracy:

```
scores = model.evaluate(X_test, Y_test, verbose = 1)
print("Доля правильных ответов на тестовых данных, в процентах:",
round(scores[1] * 100, 4))
```

ПРИЛОЖЕНИЕ В
Текст ноутбука «TGbot_lungs_analyzer.ipynb»

```
import telebot

import os

os.environ['TF_ENABLE_ONEDNN_OPTS'] = '0'
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '1'

import numpy as np
from PIL import Image

from tensorflow.keras.models import load_model
import cv2

CUDA_VISIBLE_DEVICES=""

from google.colab import userdata
userdata.get('token')

token = '**********'

bot = telebot.TeleBot(token)

model = load_model('94-9-v1.keras')

@bot.message_handler(content_types=["text"])
def echo(message):
    bot.send_message(message.chat.id, 'Отправьте мне рентген снимок лёгких.
Я проверю, нет ли чего...')

@bot.message_handler(content_types=["photo"])
def analyze(message):

    fileID = message.photo[-1].file_id

    file_info = bot.get_file(fileID)

    downloaded_file = bot.download_file(file_info.file_path)

    with open("g.jpeg", 'wb') as new_file:
        new_file.write(downloaded_file)

    classes = ['Норма', 'Пневмония', 'ВП', 'НП', 'Обструкция', 'Инфекции',
'Травмы', 'Изм. средостения', 'Изм. легких']
    IMG_SIZE = 260
    nn = cv2.imread('g.jpeg', cv2.IMREAD_GRAYSCALE)
    new_nn = cv2.resize(nn, (IMG_SIZE, IMG_SIZE))
    x = new_nn
```

```
   x = np.expand_dims(x, axis=0)
    x = np.array(x).reshape(-1, IMG_SIZE, IMG_SIZE, 1)

    x = x / 255.0
    predictions = model.predict(x)
    prediction = np.argmax(predictions[0])

    bot.send_message(message.chat.id, f"Класс заболевания:, {classes[predic-
tion]}")

bot.polling(none_stop=True)
```

ПРИЛОЖЕНИЕ Д
Текст ноутбука «Grad-CAM Functional API.ipynb»

```
def make_gradcam_heatmap(img_array, model, last_conv_layer_name, pred_in-
dex=None):
    # Создаем модель, выводящую активации последнего свёрточного слоя и ито-
говый прогноз
    grad_model = tf.keras.models.Model(
        [model.input],
        [model.get_layer(last_conv_layer_name).output, model.output]
    )

    # Запускаем изображение через модель с записью градиентов
    with tf.GradientTape() as tape:
        conv_outputs, predictions = grad_model(img_array)
        if pred_index is None:
            pred_index = tf.argmax(predictions[0])
        class_channel = predictions[:, pred_index]

    # Вычисляем градиенты
    grads = tape.gradient(class_channel, conv_outputs)

    # Усредняем градиенты по ширине и высоте
    pooled_grads = tf.reduce_mean(grads, axis=(0, 1, 2))

    # Умножаем карты признаков на их "важность"
    conv_outputs = conv_outputs[0]
    heatmap = conv_outputs @ pooled_grads[..., tf.newaxis]
    heatmap = tf.squeeze(heatmap)

    # Нормируем тепловую карту
    heatmap = tf.maximum(heatmap, 0) / tf.math.reduce_max(heatmap)
    return heatmap.numpy()
def save_and_display_gradcam(img, heatmap, alpha=0.4):
    # Масштабируем тепловую карту на 0-255
    heatmap = np.uint8(255 * heatmap)

    # Увеличиваем тепловую карту до размера изображения
    heatmap = cv2.resize(heatmap, (img.shape[1], img.shape[0]))

    # Переводим тепловую карту в цвет
    jet = cv2.applyColorMap(heatmap, cv2.COLORMAP_JET)

    # Объединяем оригинальное изображение с тепловой картой
    jet = cv2.cvtColor(jet, cv2.COLOR_BGR2RGB)
    superimposed_img = jet * alpha + img

    # Нормируем и показываем
    plt.imshow(np.uint8(superimposed_img))
    plt.axis('off')
```

```
plt.show()
img_path = '/content/g.jpeg'
# Пример загрузки
img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
img = cv2.resize(img, (260, 260))  # подгоняем размер
img = img / 255.0  # нормализация
img = np.expand_dims(img, axis=-1)  # добавляем канал
img = np.expand_dims(img, axis=0)   # добавляем батч

# Вычисляем тепловую карту
heatmap = make_gradcam_heatmap(
    img_array=img,
    model=model_functional,
    last_conv_layer_name="conv2d"  # <-- последний свёрточный слой
)

# Накладываем тепловую карту
# Преобразуем обратно в RGB для наложения
img_rgb = np.repeat(img[0], 3, axis=-1) * 255
save_and_display_gradcam(img_rgb.astype(np.uint8), heatmap)
```