МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ РАДИОФИЗИКИ И КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ
Кафедра интеллектуальных систем

ОПРЕДЕЛЕНИЕ ПНЕВМОНИИ ПРИ ПОМОЩИ ОСТАТОЧНЫХ НЕЙРОННЫХ СЕТЕЙ

Курсовая работа

Машканцев Алексей
Андреевич

студента 3 курса,
специальность
«компьютерная безопасность»

Научный руководитель:
Старший преподаватель
Н. Н. Щетько

Минск, 2022

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Медицинские рентгеновские снимки являются одним из первых и основных вариантов диагностики различных заболеваний. Рентгенограммы, благодаря наличию широкого спектра информации, анализ которого позволяет не только выявить на ранних стадиях и поставить точный диагноз заболевания, но и отслеживать динамику процесса лечения, получили широкое распространение в медицинской практике. В то же время, автоматизированное обнаружение аномалий на снимках, позволяющее снизить нагрузку на специалистов - рентгенологов, повысить эффективность обработки данных и как следствие, повысить точность диагноза остается приоритетным направлением развития информационных технологий. По оценке специалистов, врач-рентгенолог за день может качественно обработать не более 200 рентгенограмм. В густонаселенных районах врач-рентгенолог вынужден изучать гораздо большее число снимков, что неизбежно влечет за собою снижение качества обработки информации, содержащейся в рентгенограмме. Практика показывает, что сложность в работу специалиста вносит не только все увеличивающееся количество снимков, но и их качество, зависящее от ряда параметров: содержание посторонней информации (наличие кардиостимуляторов, операционных швов, проводов и т.п.), оптические шумы (засветка, затемнение), анатомические особенности тела пациента.

Критерием качества оказания помощи больному с бактериальной пневмонией является начало терапии антибиотиками. Врач, который поставил диагноз пневмонии, должен в течение ближайших 4 ч приступить к назначению антибактериальных препаратов (АБП), из-за этого анализ рентгенограмм должен быть не только точным, но и быстрым [11].

Перспективным решением поставленной задачи автоматизированного обнаружения аномалий на рентгеновских снимках являются применение остаточных нейронных сетей, которые занимают лидирующие позиции, среди программных средств, применяемых для обработки и анализа изображений. Для решения задачи классификации фронтальных рентгеновских снимков по поиску пневмонии использовал остаточную нейронную сеть (ResNet), предварительно обученную на наборе данных ImageNet [12]. Кроме того, остаточная нейросеть показала высокие результаты при поиске аномалий на рентгенограммах грудной клетки [13]. Классификация снимков на нормальные (здоровые) и с аномалиями существенно бы помогла работе специалистов на первичном этапе диагностики.

Целью данной работы является исследование остаточных нейронных сетей в ходе решения задачи поиска аномалий на рентгенограммах грудной клетки.

ГЛАВА 1
ОСНОВНЫЕ ПОНЯТИЯ

1.1 Пневмония

Не смотря на достижения современной медицины и появление новых эффективных антибактериальных препаратов, пневмония является чрезвычайно распространенным и угрожающим жизни заболеванием. По статистике Всемирной организации здравоохранения Пневмония является причиной смертности 14% детей до 5 лет во всем мире [14].

Пневмония – острое инфекционное воспаление нижних дыхательных путей с обязательным вовлечением легочной ткани (альвеол, бронхов, бронхиол).

Пневмония — это, прежде всего, бактериальное заболевание. Основные возбудители пневмонии: пневмококк (Streptococcus pneumoniae), стафилококк (Staphylococcus aureus), гемофильная палочка (Haemophilus influenzae) а также «атипичные» инфекции (Chlamydya pneumoniae, Mycoplasma pneumoniae, Legionella pneumoniae). Реже причиной острой пневмонии могут быть (Klebsiella pneumoniae, Escherichia coli, Pseudomonas aeruginosa, Acinetobacter и т.д.). Они чаще встречаются у пациентов с тяжелыми сопутствующими заболеваниями, у больных с ослабленной иммунной системой.

Пусковым фактором развития пневмонии могут быть различные вирусные инфекции. Они вызывают воспаление верхних дыхательных путей и обеспечивают «комфортные условия» для развития бактериальных возбудителей.

Незаменимой для постановки точного диагноза пневмонии является рентгенография грудной клетки. Она проводится в прямой, а при необходимости и в боковой проекции и позволяет не только установить диагноз острой пневмонии и выявить возможные осложнения, но и оценить эффективность лечения. Её быстрый и точный анализ помогает эффективнее лечить пациентов.

1.2 Нейрон

Нейронная сеть представляет из себя совокупность нейронов, соединенных друг с другом определенным образом. Нейрон представляет из себя элемент, который вычисляет выходной сигнал (по определенному правилу) из совокупности входных сигналов. То есть основная последовательность действий одного нейрона такая:

- Прием сигналов от предыдущих элементов сети
- Комбинирование входных сигналов
- Вычисление выходного сигнала
- Передача выходного сигнала следующим элементам нейронной сети

Для обучения нейронных сетей также необходимо задать функцию активации. На рисунке 1.1 показана модель искусственного нейрона.

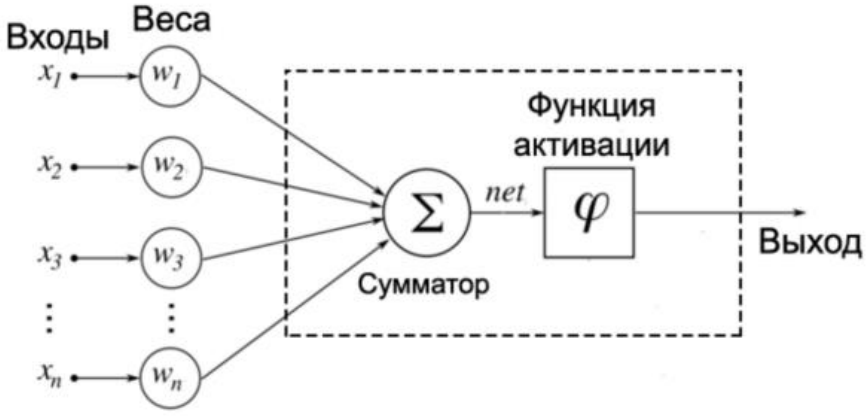

Рисунок 1.1 Искусственный нейрон

Нейрон имеет входы x_1 x_2 и т.д., которые имеют веса w_1, w_2 и т.д.

Первоначально веса задаются произвольными небольшими числами, а затем настраиваются методом обратного распространения ошибки. Сумматор - функция, которая складывает веса входного сигнала x. Функция активации вычисляет выходной сигнал нейрона. Чаще всего в сверточных нейронных сетях используется слой с полулинейной функцией активации relu (Rectified Linear Unit). Производная этой функции равна либо 0, либо 1. Поэтому данная функция эффективно борется как с разрастанием градиента при обратном проходе, так и с затуханием. Данное свойство приводит к прореживанию весов, что хорошо сказывается на работе глубоких нейронных сетей.

Между собой нейроны могут быть соединены абсолютно по-разному, это определяется структурой конкретной сети. Но суть работы нейронной сети остается всегда одной и той же. По совокупности поступающих на вход сети сигналов на выходе формируется выходной сигнал (или несколько выходных сигналов). То есть нейронную сеть упрощенно можно представить в виде

черного ящика, у которого есть входы и выходы. А внутри этого ящика сидит огромное количество нейронов [2].

1.3 Нейронные сети

Один нейрон может выполнять простейшие вычисления, но основные функции нейросети обеспечиваются не отдельными нейронами, а соединениями между ними. Однослойный перцептрон представляет собой простейшую сеть, которая состоит из группы нейронов, образующих слой. Входные данные кодируются вектором значений, каждый элемент подается на соответствующий вход каждого нейрона в слое. В свою очередь, нейроны вычисляют выход независимо друг от друга. Размерность выхода (то есть количество элементов) равна количеству нейронов, а количество синапсов у всех нейронов должно быть одинаково и совпадать с размерностью входного сигнала [2].

Многослойная нейронная сеть (персептрон) — это нейронная сеть, состоящая из входного, выходного и расположенных между ними одного (или нескольких) скрытых слоев нейронов. Пример персептрона с 2 слоями изображен на рисунке 1.2.

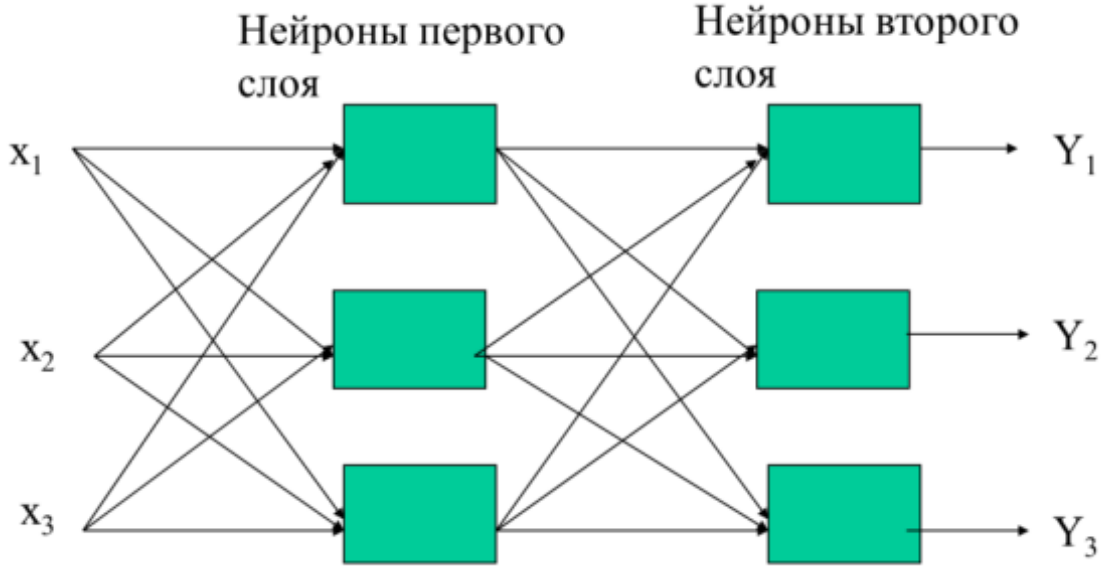

Рисунок 1.2 Персептрон

1.4 Сверточные нейронные сети

Сверточные нейросети (Convolution Neural Networks, CNN) в общем виде представлены на рисунке 1.3. Входные данные у таких сетей, как правило, состоят из изображений. Слои CNN состоят из нейронов, расположенных в 3- х измерениях: ширине, высоте и глубине. Основные слои данной нейросети представлены сверточным слоем (convulution), субдискретизирующим (pooling) и полносвязным [5]. Для обучения сверточных нейросетей для классификации объектов необходимо подготовить два набора данных - обучающий (train dataset) и тестовый (validation dataset). Train dataset предполагается использовать для обучения нейросети, validation dataset - для проверки ее работы. Задача классификации формулируется следующем образом: имеется множество объектов, для некоторых из них известно, к каким классам они принадлежат; для других объектов их классовая принадлежность неопределенна, и требует их распределения.

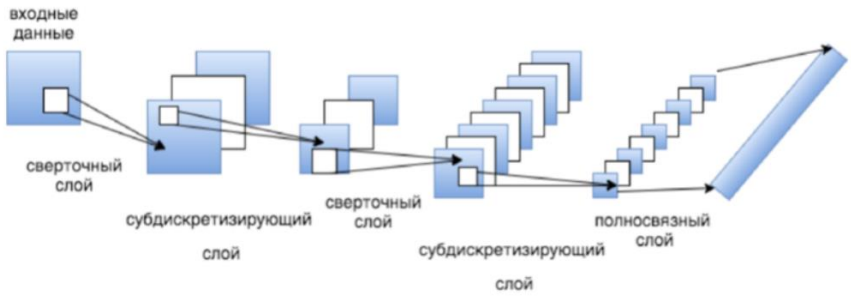

Рисунок 1.3 Схема CNN

Сверточный слой накладывая ядро свертки, поэлементно умножает значения фильтра и изображения всеми возможными способами, и записывает сумму произведений элементов исходного изображения и ядра. Как правило, применяется не одно ядро свертки, а сразу несколько. Результатом работы данного слоя становятся карты признаков.

Субдискретизирующий слой выполняет операцию по понижению дискретизации пространственных размеров изображения. Если на предыдущей операции свертки уже были выявлены некоторые признаки, то для дальнейшей обработки настолько подробное изображение уже не нужно, и оно уплотняется до менее подробного. К тому же фильтрация уже ненужных деталей помогает не переобучаться. В процессе сканирования ядром субдискретизурующего слоя (фильтром) карты предыдущего слоя, сканирующее ядро не пересекается в отличие от сверточного слоя. Обычно, каждая карта имеет ядро размером 2x2, что позволяет уменьшить предыдущие карты сверточного слоя в 2 раза. Вся карта признаков разделяется на ячейки 2x2 элемента, из которых выбираются максимальные по значению.

Наконец, полносвязный слой берёт входные данные и выводит N-пространственный вектор, где N — число классов, из которых нейронная сеть выбирает нужный. Способ, с помощью которого работает полносвязный слой — это обращение к выходу предыдущего слоя и определение свойств, которые больше связаны с определенным классом.

Стоит отметить, что на вход такой нейросети обычно подаются тензоры, т. е. трехмерный вектор, которой содержит в себе информацию о ширине и высоте изображения, а также о количестве его каналов. Например, если у нас изображение формата RGB, то количество каналов равно трем.

1.5 Используемые метрики

Для определения качества работы нейронной сети и сравнения их результатов рассмотрены следующие метрики: функция потерь (Loss), точность (Precision), полнота (Recall) и F1-мера [4]. На выходе нейронной сети в задачи классификации получаются вероятности, которые в сумме должны дать единицу. Функция потерь (Loss) характеризует суммирование ошибок, сделанных для каждого примера в какой-либо выборке. В качестве функции потерь в нейронных сетях часто используется перекрестная энтропия, определяемая формулой (1.1) на каждом шаге обучения:

$$L = -\sum_{i=1}^{n} t_i \log y_i, \quad (1.1)$$

где t_i - требуемые выходы для текущего обучающего примера, y_i - реальные выходы нейронной сети.

Общая ошибка будет вычисляться по формуле (1.2):

$$L = -\frac{1}{Size}\sum_{a \in Dataset}^{\sum_{i=1}^{n}} t_i^a \log y_i^a , \quad (1.2)$$

где Size - размер обучающей или валидационной выборки, Dataset - сама эта выборка.

В нашем случае рассматривается утверждение об наличии пневмонии, следовательно положительный результат говорит о присутствии признаков пневмонии на рентгенограмме. При работе нейронной сети возможны решения, представленные в матрице ошибок (confusion matrix), изображенной в таблице 1.

Таблица 1 Матрица ошибок

		Ожидаемое значение	
		Положительное значение	Отрицательное значение
Решение системы	Положительный ответ	истинно-положительные (ИП)	ложно-положительные (ЛП)
	Отрицательный ответ	ложно-отрицательные (ЛО)	истинно-отрицательные (ИО)

Мера точности характеризует, сколько полученных от алгоритма положительных ответов являются правильными, однако она не дает информации о том, все ли правильные ответы вернул алгоритм. Поэтому еще говорят о мере полноты, которая характеризует способность алгоритма давать как можно больше положительных ответов из ожидаемых. Данные меры считаются по формулам (1.3) и (1.4):

$$\text{Полнота} = \frac{\text{ИП}}{\text{ИП+ЛО}}, \tag{1.3}$$

$$\text{Точность} = \frac{\text{ИП}}{\text{ИП+ЛП}}. \tag{1.4}$$

Ясно, что чем выше точность и полнота, тем лучше работает система. Но, как правило, максимальная точность и полнота не достижимы одновременно и поэтому используют метрику F1-меры, которая объединяет в себе информацию о точности и полноте алгоритма. F1-мера представляет собой гармоническое среднее между точностью и полнотой, и вычисляется согласно формуле (1.5):

$$F1 = 2 \times \frac{\text{Точность} \times \text{Полнота}}{\text{Точность+Полнота}}. \tag{1.5}$$

Так как мы работаем с медицинскими данными, мы стремимся к наибольшей полноте с высоким F1, так как нам необходимо обнаружить как можно больше больных, то есть ложно-отрицательные результаты для нас хуже ложно-положительных.

1.6 Задача предварительной обработки

Состав обучающей базы может влиять на качество получающейся системы распознавания больше, чем все остальные факторы. Это будет наглядно продемонстрированно наглядно далее в реализации.

Чем точнее обучающая выборка аппроксимирует генеральную совокупность изображений, которые будут поступать на вход вашей системе, тем выше будет предельно достижимое качество результата. Нам необходимо как расширить выборку изображений, а также предоставить её нейронной сети в наилучшем для точности и скорости обработки формате.

1.6.1 Гистограммная эквализация снимков

Для увеличения качества изображений был использован метод гистограммной эквализации изображений. Гистограммой одноканального изображения называют дискретную функцию, которая принимает значение, равное яркости пикселя. Гистограммная эквализация выполняет преобразование данной функции так, чтобы она была близка к равномерному распределению. В результате данной операции ожидается увеличение контрастности изображений (рисунок 1.4).

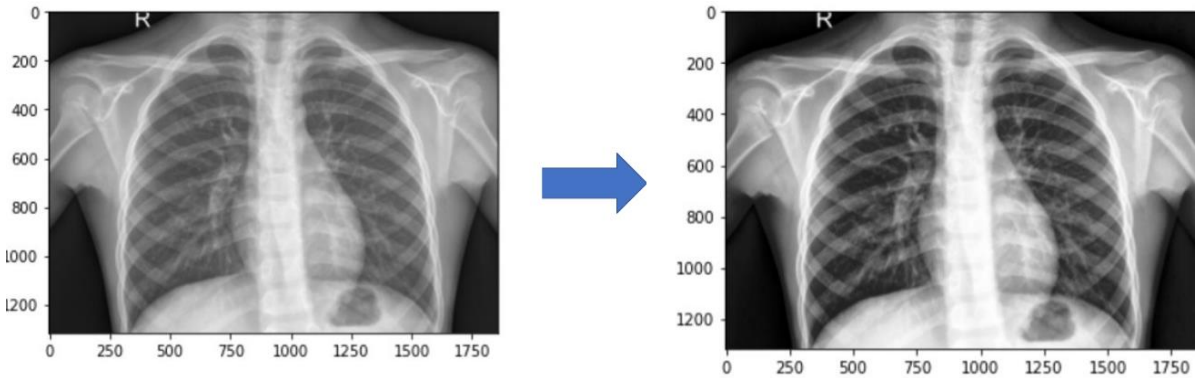

Рисунок 1.4 Результат применения метода гистограммной эквализации изображения

1.6.2 Расширение валидационной выборки

В зависимости от выбранного датесета, собранной валидационной сборки, может не быть, либо же она будет слишком мала. Так было в выбранном мной датасете, где было всего 12 изображений в валидационной сборке. Я расширил её до 600, что улучшило точность получаемого результата и его оценки, так как проверка проходила на большем количестве образцов.

1.6.3 Аугментация данных

Многие изображения имеют хуже качество, помехи и т. д., к тому же их часто недостаточно. Для исправления этих проблем существует решение, а именно аугментация данных. Это важный этап обучения моделей машинного обучения. Под аугментацией данных понимается увеличение выборки данных для обучения через модификацию существующих данных. Использование методов аугментации данных показало себя хорошо на задаче классификации изображений. Несмотря на это, исследований влияния аугментации на точность моделей распознавания не было. Для работы с моим датасетом этот этап показал себя очень хорошо, что будет продемонстрированно далее. Выполнялась данная операция при помощи функции ImageDataGenerator из API Keras [9].

ГЛАВА 2
ОСТАТОЧНЫЕ НЕЙРОННЫЕ СЕТИ

2.1 Структура блока остаточной нейронной сети

ResNet — это сокращенное название для Residual Network (дословно — «остаточная сеть»). Остаточная сеть — это глубоко сверточная сеть, структуру которой мы рассмотрели в 1.4, предложенная знаменитым исследователем Каимингом Хе в 2015 году. Однажды родившись, она выиграла чемпионов по классификации, обнаружению и позиционированию изображений в ImageNet. Остаточные сети легче оптимизировать и могут повысить точность, добавляя значительную глубину. Суть заключается в том, чтобы решить побочные эффекты (проблемы деградации), вызванные увеличением глубины, так что производительность сети можно улучшить, просто увеличив глубину сети. Данная нейросеть является намного глубже по сравнению с предыдущими архитектурами [8].

Большинство нейросетей используют в качестве метода обучения используют метод обратного распространения ошибки. То есть при проходе обучающего примера и получения выхода, с целью уменьшения функции ошибки начинается обратный проход сети методом градиентного спуска. При увеличении глубины нейросети, одна из самых частых проблем — это затухание градиента при обратном проходе, и как следствие, ухудшение работы нейронной сети. В частности, затухание градиента и потеря информации чаще всего случается из-за использования популярного слоя relu с функцией активации: $f = \max(x, 0)$.

Во многих случаях при отсеивании ненужных признаков, как раз и получается значение 0, что и способствует затуханию градиента. В основе архитектуры данной сети лежат остаточные блоки (residual blocks), которые содержат "обходную связь идентичности" (identity shortcut connection), обходящую один или большее количество слоев [6], пример такого блока изображен на рисунке 2.1.

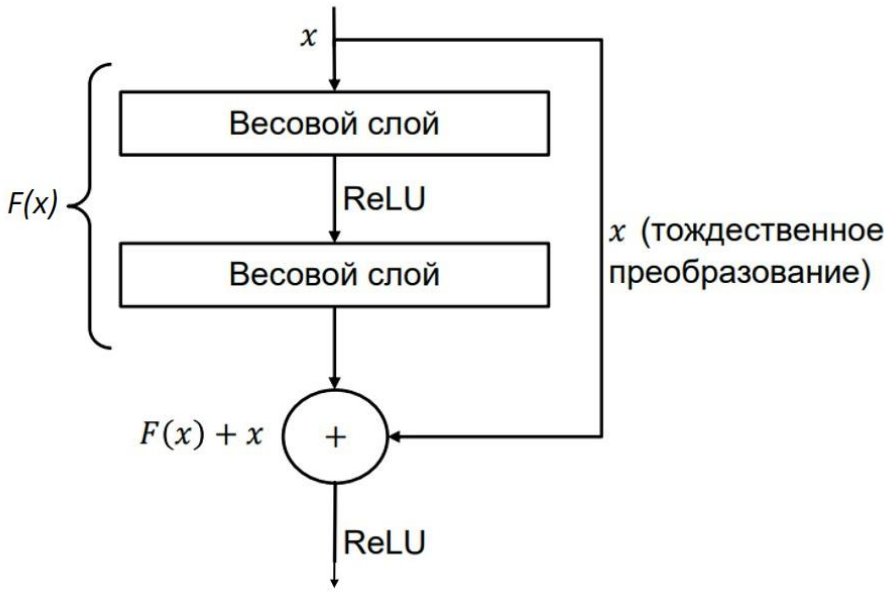

Рисунок 2.1 Блок ResNet

Работа данного блока объясняется следующим образом: пусть целевая функция будет в виде $H(x) = F(x) + x$. При достигнутом пределе качества на предыдущем слое, для того чтобы не происходило затухание градиента, хотелось бы, чтобы функция F(x) возвращала тождественное преобразование, однако этого не происходит, так как используется слой relu, и часто она возвращает 0. В данной сети предлагают использовать, так называемые, shortcut соединения, то есть явно добавляется тождественное отображение. В итоге при обратном проходе получаем формулу (2.1):

$$\frac{dF(x)}{dx} + \frac{dx}{dx} = \frac{dF(x)}{dx} + 1. \qquad (2.1)$$

Таким образом затухания градиента не произойдет, так как всегда будет выполнен обратный проход из-за наличия +1 в формуле (2.1). Мы непосредственно принимаем вывод X после определенного слоя в качестве входного, напрямую пропускаем некоторые непрерывные сетевые слои и отправляем его на результат более позднего слоя. Это облегчает изучение остаточного блока (поскольку он выполняет только операцию relu).

2.2 Структура Resnet50, ResNet101 и Resnet152

ResNet50 — это 50-уровневая распределенная сеть. 50-слойная ResNet содержит две структуры: идентификационный блок и сверточный блок [7], Идентификационный блок является стандартным блоком, используемым в ResNet, и соответствует случаю, когда активация входа имеет то же измерение, что и активация выхода. На рисунке 2.2 сверху приведен пример блока идентификации, где верхний путь — это «путь быстрого доступа», или же shorcut, а нижний путь — это «основной путь». Когда входные и выходные размеры не совпадают, мы добавляем сверточный слой в путь быстрого доступа. Получаемый сверточный блок, изображенный на рисунке 2.2.

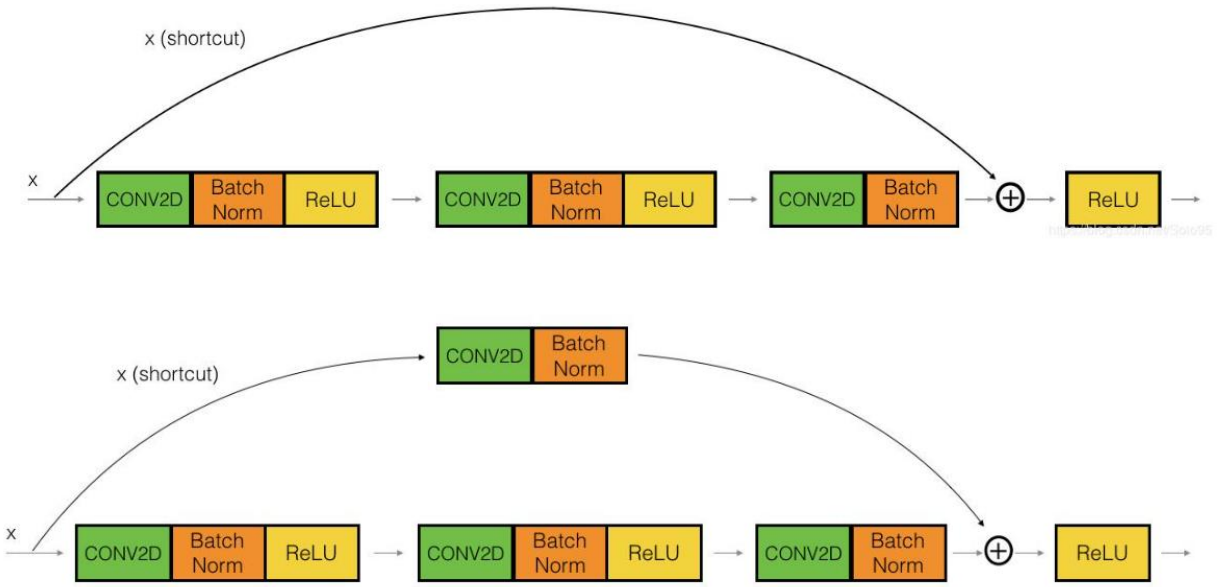

Рисунок 2.2 Идентификационный блок(сверху) и сверточный блок(снизу)

Основное различие между этими двумя структурами заключается в том, выполняется ли операция свертки на соединении быстрого доступа.

ResNet50 построен из этих двух модулей, а также слоёв для обработки входящего изображения и формирования вывода результата, как показано на рисунке 2.3. «ID BLOCK» на диаграмме означает «Блок идентификации», а «ID BLOCK x3» означает, что должны сложить 3 блока идентификаторов вместе. Для формирования входа для 1 такого блока используется один сверточный слой формы (7,7), слой нормализации и и слой пулинга. Последний слой сети Dense позволяет уменьшить количество входных данных до количества класов.

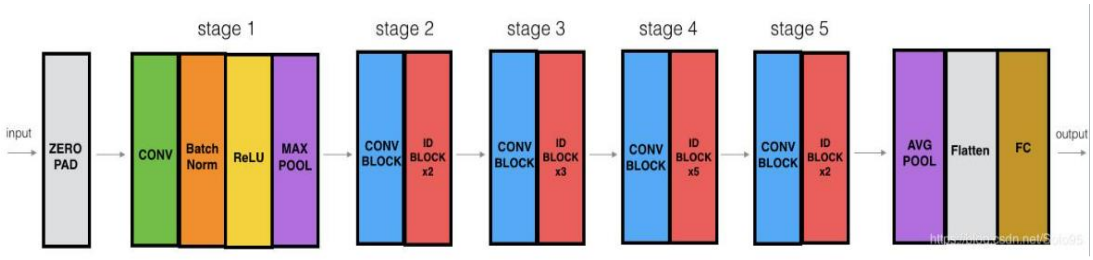

Рисунок 2.3 Схема ResNet50

Как можно увидеть на рисунке 2.4, основное различие между ResNet50, ResNet101 и ResNet152 заключается в разном количестве идентификационных и сверточных блоков.

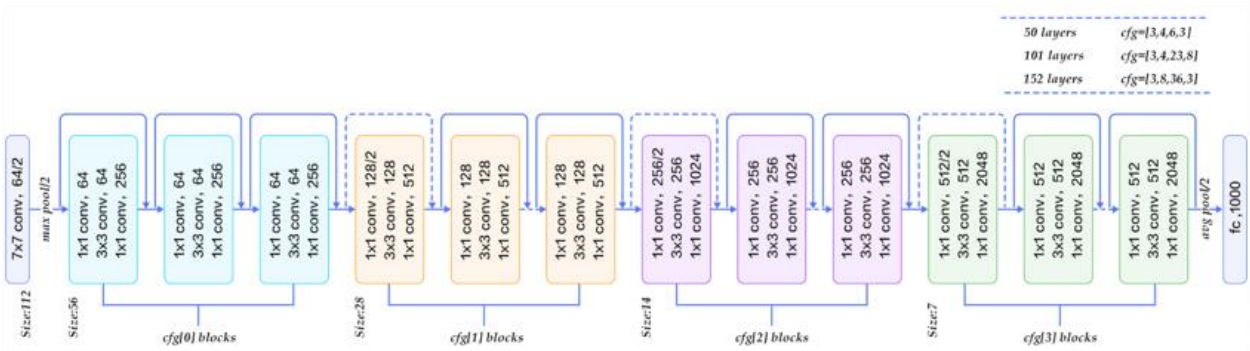

Рисунок 2.4 Схема для ResNet50, ResNet101, ResNet152

В данной работе рассматривались вторые версии этих сетей, их различие изображено на рисунке 2.5. Слои используются одинаковые, но в другом порядке, что приводит к небольшому, но улучшению в результатах.

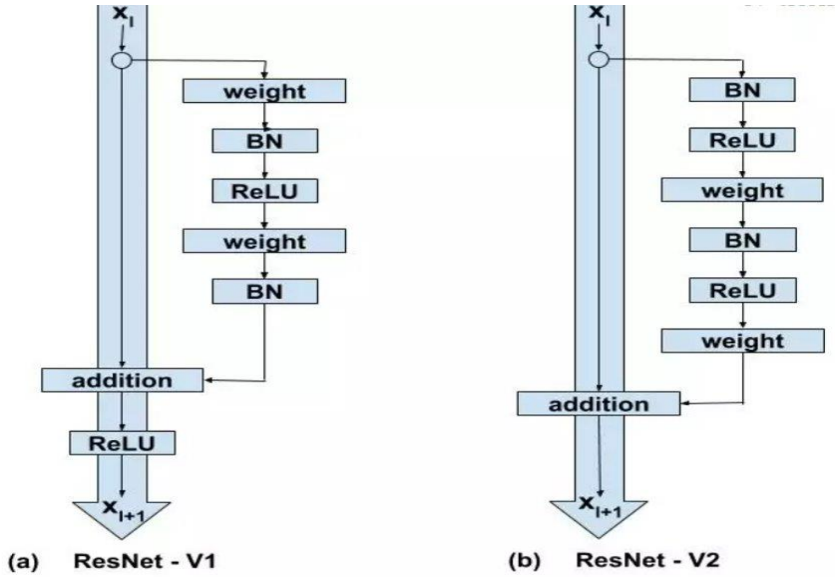

Рисунок 2.5. Структура 2 версий ResNet

2.3 Преимущества остаточных нейронных сетей при обработке изображений

- ResNet относительно легко оптимизировать: «простые» сети (которые просто складывают слои) показывают большую ошибку обучения, когда глубина увеличивается.
- ResNet позволяет относительно легко увеличить точность благодаря увеличению глубины, чего с другими сетями добиться сложнее.
- ResNet заняла топ 1 на ImageNet Large Scale Visual Recognition Challenge 2015 (ILSVRC2015), что говорит об её эффективности для задач обработки изображений [1].
- Существуют тренированные модели, дающие быстрый результат и простые в реализации.

2.4 Создание своей остаточной нейронной сети

Также, в данной работе я создал свою остаточную нейронную сеть из 14 блоков, так как это был максимум для ресурсов моей системы. Структура блоков предоставлена на рисунках 2.6 и 2.7. В этих блоках находится 2(3 в свёрточном блоке) сверточных и 2 нормализующих слоя, а также по 1 слой активации и суммирующий слой. Последний слой сети Dense имеет 1 нейрон, так как нам необходимо разделение всего на 2 класса.

Эта сеть была создана после работы с готовыми решениями Keras, что позволило быстро получить хороший результат, так многие параметры уже были подобраны, а данные были обработаны в соответствии с датасетом.

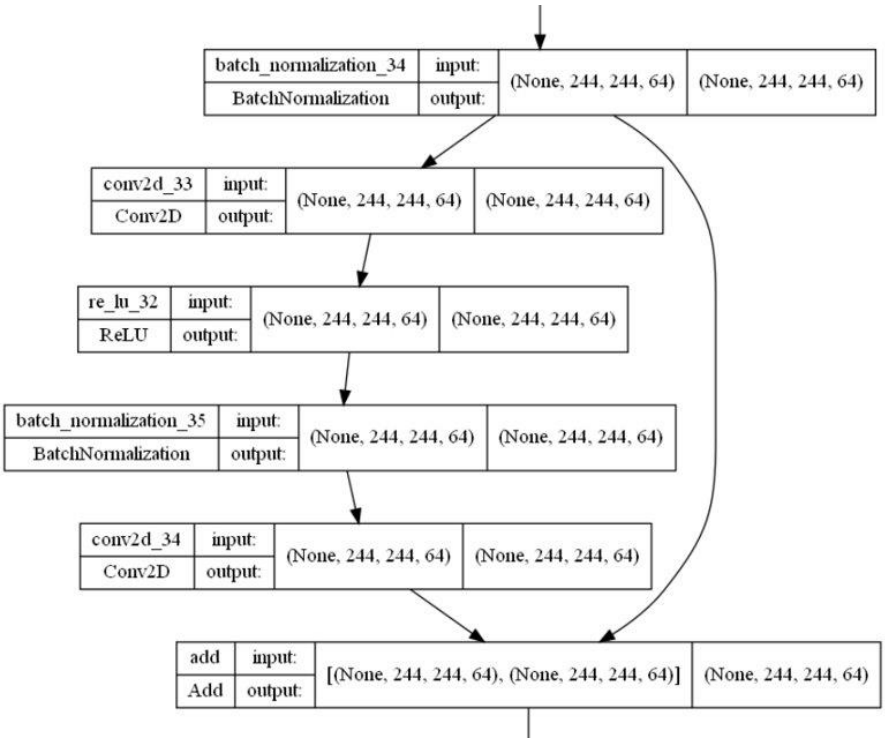

Рисунок 2.6. Идентификационный блок

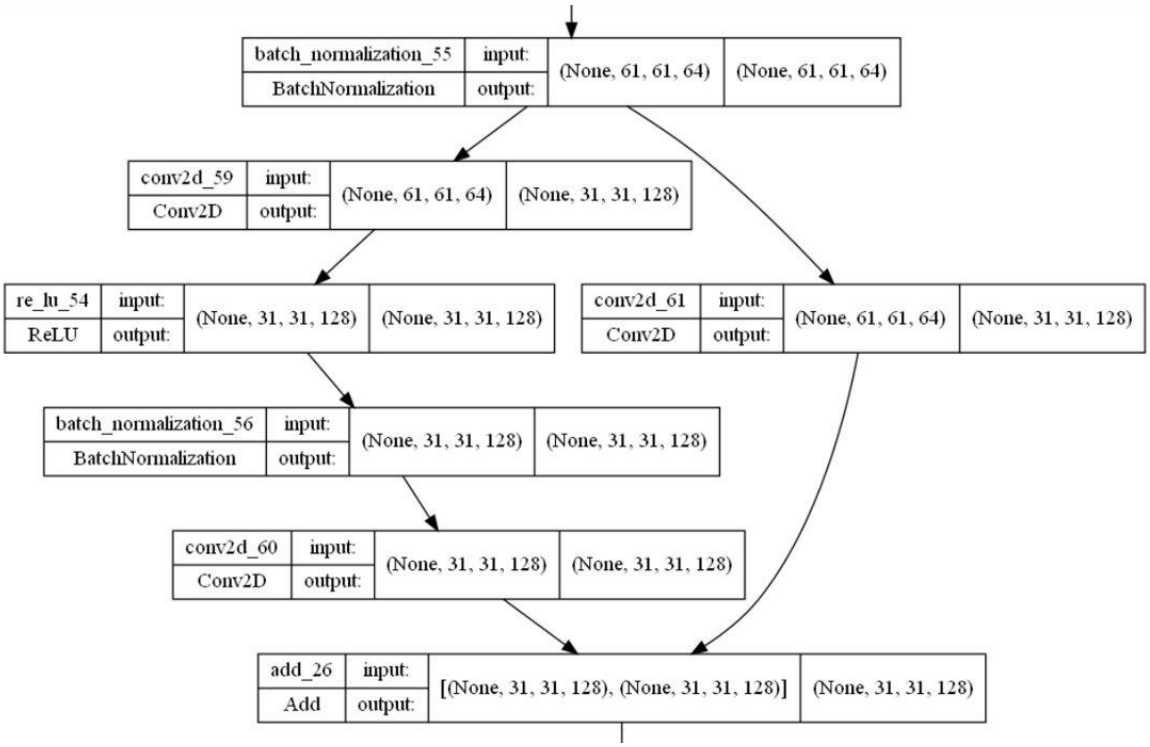

Рисунок 2.7. Сверточный блок

ГЛАВА 3
РЕАЛИЗАЦИЯ

3.1 Обзор средств и технологий

Для обучения нейронных сетей, представленных выше необходимо выбрать набор данных, на котором будет происходить обучение. В качестве такого набора был выбран набор Labeled Optical Coherence Tomography (OCT) and Chest X-Ray Images for Classification , собранный в Китае и находящийся в свободном доступе [3]. Данный набор содержит около 5800 снимков, разделенных на здоровые и с патологиями. Изображения в данном наборе имеют различные размеры и хранятся в формате JPEG. Кроме того, некоторые снимки обладают относительно плохим качеством и т.п., однако не пригодных для обучения при просмотре не было обнаружено, В датасете было около 1600 “здоровых” снимков и около 4200 снимка - с аномалиями. Далее данные снимки были сжаты до одинакового размера 224 × 224 пикселя. Несмотря на значительное изменение размеров для многих изображений, данный датасет по-прежнему пригоден для обучения. К изображениям в ходе работы для улучшения результатов были применены операции гистограммной эквализации, аугментации и была расширена валидационная сборка, это было описано выше 1.6. Затем были обучены остаточные нейронные сети, описанные выше в 2.2 и 2.4. Для этого используется язык программирования python 2.7 и библиотека для машинного обучения Keras, а также CUDA для ускорения обучения. Во время обучения были выбранны параметры обучения learning_rate = 0.001 и Batch_Size = 32. Для оценки результатов использовались метрики из 1.5.

3.1.1 Keras

Keras является высокоуровневым API(Application Programming Interface — «программный интерфейс приложения») для разработки нейронных сетей. Он написан на Python и может работать поверх TensorFlow, CNTK или Теано. Он был разработан с упором на возможность быстрого экспериментирования. Он способен дать путь от идеи к реализации в кратчайшие сроки, что является ключом к проведению хороших исследований.

Причины использования:

- Удобно использовать. Keras — это API, предназначенный для людей, а не для машин. В нем удобный пользовательский интерфейс. Keras следует наилучшим методам снижения когнитивной нагрузки: он предлагает последовательные и простые API, а также минимизирует количество действий пользователя, и обеспечивает четкую и эффективную обратную связь.
- Модульность. Под модульностью понимается наличие автономных полностью настраиваемых модулей, которые могут быть подключены вместе с минимальными ограничениями. В частности, нейронные слои, функции затрат, оптимизаторы, схемы инициализации, функции активации, схемы регуляризации — это автономные модули, которые можно комбинировать для создания новых моделей.
- Легкая масштабируемость. Новые модули просто добавлять (как новые классы и функции), а существующие модули предоставляют множество примеров, что делает Keras подходящим для передовых исследований.
- Работа с Python. Модели описаны в коде Python, который компактен, легче отлаживается и обеспечивает простоту расширяемости.

Основная структура данных Keras — это модель, способ организации слоев. В Keras доступны два основных типа моделей: последовательная модель Sequential и класс Model, используемый с функциональным API. Простейшим типом модели является Sequential модель, которая представляет собой линейную совокупность слоев. Для более сложных архитектур необходимо использовать функциональный API Keras, который позволяет создавать произвольные графики слоев, что я и делал при создании своей нейронной сети. Тренированные сети можно реализовать при помощи последовательной модели.

Модель должна знать размерность входного массива данных. Поэтому первый слой в Sequential модели (и только первый, потому что следующие слои могут получать автоматически эту информацию из предыдущего слоя) должен получать информацию о размерность входного массива. Существует несколько способов сделать это, но основным является передача аргумента input_shape первому слою [9], этим способом я и воспользовался.

3.1.2 Nvidia CUDA

CUDA – это программно-аппаратная архитектура параллельных вычислений, позволяющая существенно увеличить вычислительную продуктивность благодаря использованию графических процессоров NVIDIA.

CUDA позволяет программистам реализовывать на специальном упрощенном диалекте языка C алгоритмы, которые используются в графических процессорах NVIDIA, и включать специальные функции в текст программы на C. Архитектура CUDA позволяет разработчику на свое усмотрение организовывать доступ к набору инструкций GPU и управлять его памятью [8].

Эта технология поддерживает несколько языков программирования. Среди них Java, Python и некоторые другие. Я работал с её реализацией на Python при помощи совместимых модулей Keras.

Графический процессор не всегда может дать ускорение при выполнении определенных алгоритмов. Поэтому перед его использованием для вычислений стоит хорошо подумать, а нужен ли он в данном случае. При обучении свёрточных нейронных сетей графический процессор показывает результаты в разы лучше центрального, так как это работа с большим массивом данных и матричными преобразованиями, поэтому нейронные сети в данной работе я обучал, используя его ресурсы. Одна эпоха обучения сети занимала больше часа на моей системе при использовании исключительно центрального процессора, что делало обучение таких сетей нецелесообразным без задействования графического процессора. Использование CUDA позволило сократить время одной эпохи до минуты.

Главный минус CUDA в том, что данная технология поддерживается только видеокартами NVIDIA без каких-либо альтернатив.

3.1.3 TensorFlow

TensorFlow — это сквозная платформа с открытым исходным кодом для машинного обучения. Она имеет комплексную и гибкую экосистему инструментов, библиотек и ресурсов сообщества, которая позволяет исследователям внедрять самые современные технологии машинного обучения, а разработчикам легко создавать и разворачивать приложения на основе машинного обучения [10].

Преимущества:

- Простое построение модели. TensorFlow предлагает несколько уровней абстракции, поэтому всегда можно выбрать тот, который соответствует потребностям. Есть высокоуровневый API Keras, который упрощает начало работы с TensorFlow и машинным обучением. Есть решения для распределенного обучения на различных конфигурациях оборудования без изменения определения модели.
- Надежное машинное обучение в любом месте. TensorFlow обеспечивает прямой путь к обучению. Будь то на серверах, периферийных устройствах или в Интернете, TensorFlow позволяет легко обучать и разворачивать модель, независимо от используемого языка или платформы. Можно использовать TensorFlow Extended (TFX), если нужен полноценный производственный конвейер машинного обучения. Для выполнения логических выводов на мобильных и пограничных устройствах используется TensorFlow Lite.
- Мощные инструменты для исследований. Можно создавать и обучать современные модели, не жертвуя скоростью или производительностью. TensorFlow обеспечивает гибкость и контроль благодаря таким функциям, как функциональный API Keras и API подклассов моделей для создания сложных топологий. Для легкого прототипирования и быстрой отладки используется активное выполнение. TensorFlow также поддерживает экосистему мощных дополнительных библиотек и моделей для исследований, включая Ragged Tensors, TensorFlow Probability, Tensor2Tensor и BERT.
- Хорошая документация, содержащая как краткую общую информацию, так и подробности.

3.2 Улучшение точности работы остаточной нейронной сети на примере ResNet152V2

Изначально использовалась ResNet152V2 из API Keras, и без предварительной обработки датасета был получен результат на рисунок 3.1, где можно заметить, что валидационная точность колеблется в районе 50%, что является сильно неудовлетворительным результатом. Для исправления этого была проведена аугментация данных, также была использована функция tf.keras.callbacks.EarlyStopping, которая заканчивала обучение раньше заданного количества эпох, если результаты не улучшаются, после сеть была обучена заново, результат на рисунок 3.2.

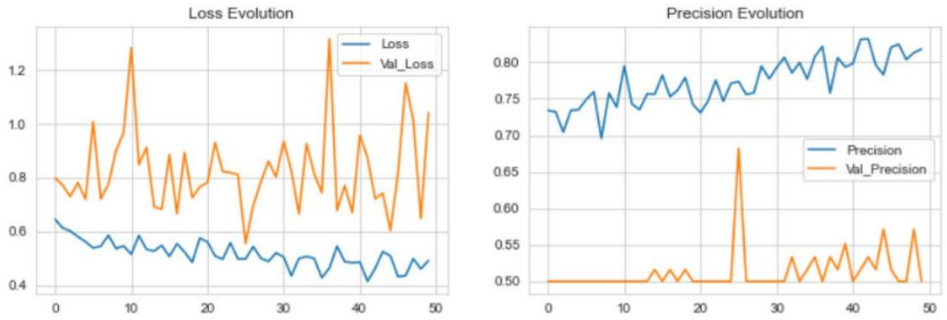

Рисунок 3.1 Первый результат ResNet152V2, слева график потерь, справра точности

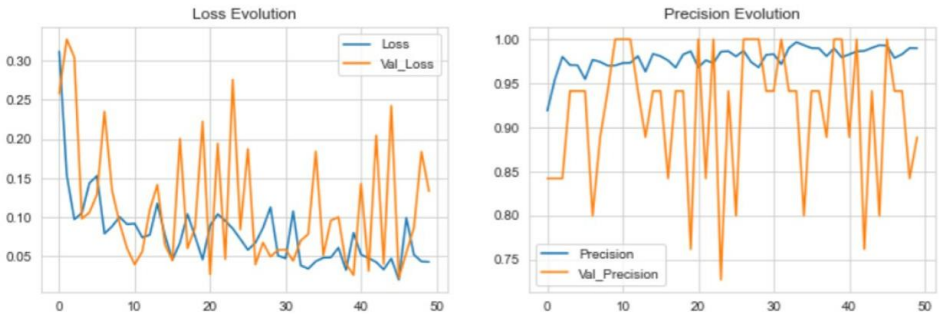

Рисунок 3.2 Второй результат ResNet152V2, слева график потерь, справа точности

График потерь и график точности (рисунок 3.2) стали выдавать показатели получше, однако заметны сильные колебания при валидации. Для снижения амплитуды этих колебаний была сделана гистограммная эквализация и подкорректированы параметры аугментации. Результат после обучения на новых параметрах на рисунке 3.3.

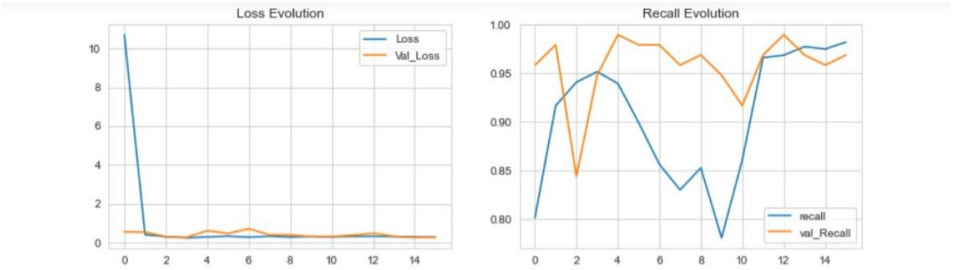

Рисунок 3.3 Третий результат ResNet152V2, слева график потерь, справа полноты

Все эти шаги показывают, что предварительная обработка может значительно влиять на обучение нейронной сети, и её особенно важно делать при работе с изображениями.

3.2.1 Результаты остальных нейронных сетей

Из рисунков 3.4-3.6 видно, что графики идут схожим образом для всех сетей, только у своей сети выделяются колебания у валидации полноты, но на конечный результат они не оказали сильное внимание, что можно увидеть в таблице результатов.

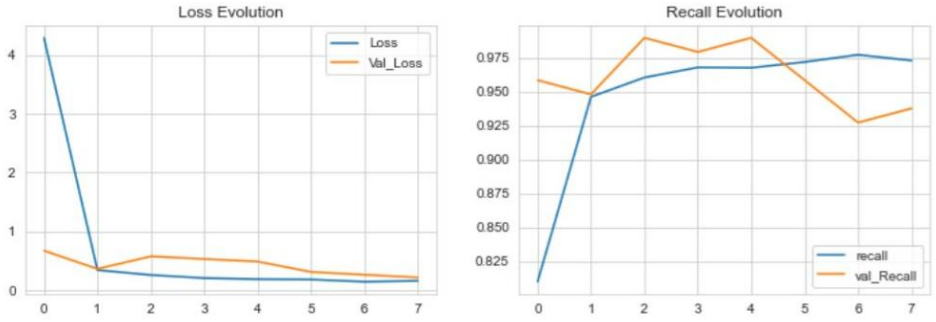

Рисунок 3.4 Результат ResNet50V2, слева график потерь, справа полноты

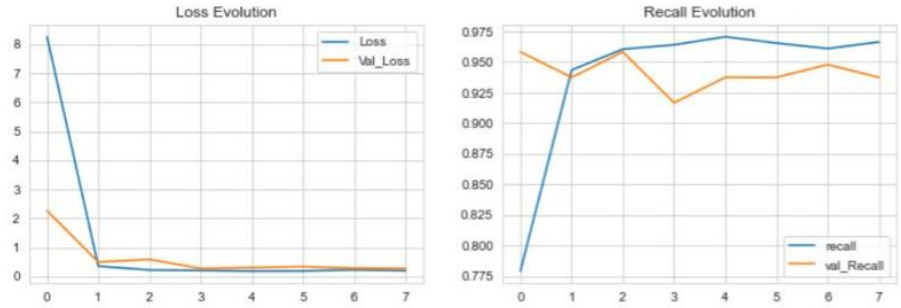

Рисунок 3.5 Результат ResNet101V2, слева график потерь, справа полноты

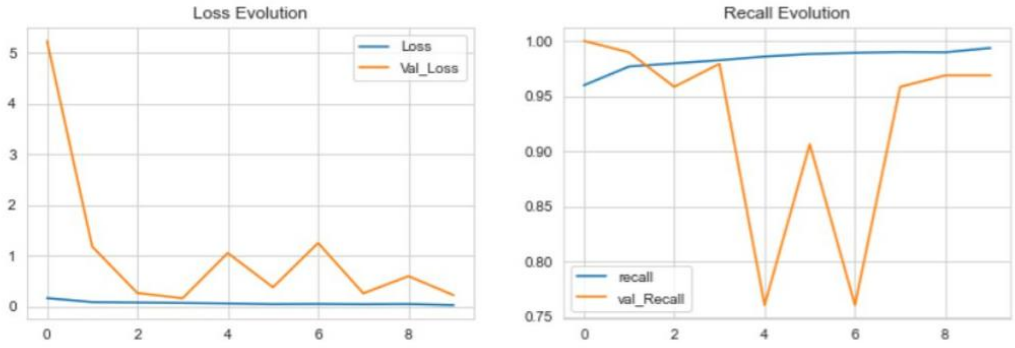

Рисунок 3.6 Результат своей сети, слева график потерь, справа полноты

3.3 Анализ полученных результатов

Из таблиц 3.1 и 3.2 видно, что своя сеть показывает лучшую точность и F1 на валидации, но при этом ей требуется намного больше времени, однако оно всё ещё невелико, так как датасет относительно небольшой. Но это может стать проблемой при большем наборе данных. Для ResNet152V2 были опробованы 2 различных параметра эпох, и явно видны плохие показатели F1 и точности при недостаточном количестве эпох. Это связано и с несбалансированностью датасета, так как из-за большего количества снимков с аномалиями нейронная сеть начала на валидации опознавать аномалии на изображениях без аномалий. У готовых моделей результаты схожие, однако более глубокая ResNet152V2 имеет наилучший результат, что связано с большим числом слоём.

Таблица 3.1 Таблица полноты и потерь после обучения

	Тестовая полнота	Валидационная полнота	Тестовые потери	Валидационные потери
ResNet50V2	**0.96**	**0.96**	**0.1**	**0.3**
Resnet101V2	**0.97**	**0.96**	**0.07**	**0.43**
ResNet152V2(4 эпохи)	**0.98**	**0.97**	**0.17**	**0.7**
ResNet152V2(8 эпох)	**0.99**	**0.98**	**0.1**	**0.56**
Сеть из 14 блоков	**0.98**	**0.97**	**0.04**	**0.22**

Таблица 3.2 Таблица точности, F1 и времени работы нейронной сети

	Точность	Валидационная точность	Валидационный f1	Время, сек
ResNet50V2	**0.94**	**0.86**	**0.91**	**80**
Resnet101V2	**0.98**	**0.87**	**0.91**	**92**
ResNet152V2(4 эпохи)	**0.93**	**0.69**	**0.8**	**56**
ResNet152V2(8 эпох)	**0.97**	**0.9**	**0.94**	**92**
Сеть из 14 блоков	**0.98**	**0.94**	**0.95**	**300**

ЗАКЛЮЧЕНИЕ

В рамках данной курсовой работы достигнуты следующие результаты:

- Сделан обзор предметной области.
- Подготовлен набор данных для обучения.
- Произведена предварительная обработка рентгеновских снимков.
- Построены и обучены нейронные сети, выбранных архитектур.
- Представлен сравнительный анализ результатов работы нейронных сетей

По результатам данной работы сделаны следующие выводы: для получения удовлетворительного результата потребовалось расширить валидационную сборку, подобрать подходящие параметры аугментации, ограничить количество эпох для наибольшей эффективности и избегания переобучения. Использование готовой модели проще реализуется и работает быстрее, однако может давать меньшую точность. Создание своих остаточных блоков и нейронной сети на их основе требует больше времени, но даёт лучший результат. Остаточные нейронные сети показывают хороший результат в обработке рентгенограмм, как по временным затратам, так и по точности. Это связано с их свойством не терять точность и малым увеличением затрат времени на обучение при увеличении количества слоёв. Это можно увидеть при анализе результатов нейронных сетей разной глубины, где ResNet152 несмотря на наличие 152 слоёв,давала лучший результат со схожим временем обучения, если сравнивать ResNet50 из 50 слоёв. Однако для применения на практике требуется анализ большего набора данных, так как анализа одного датасета недостаточно для внедрения сети в практику.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. ImageNet Large Scale Visual Recognition Challenge / Olga Russakovsky [и др.]; // ArXiv. [Электронный ресурс]—2015. — Режим доступа: https://arxiv.org/abs/1409.0575 — Дата доступа: 20.05.2022
2. Искусственные нейронные сети и приложения: учеб. пособие / Ф.М. Гафаров, А.Ф. Галимянов. – Казань: Изд-во Казан. ун-та, 2018. – 121 с.
3. Kermany, Daniel; Zhang, Kang; Goldbaum, Michael (2018), “Labeled Optical Coherence Tomography (OCT) and Chest X-Ray Images for Classification”// Mendeley Data, V2
4. Powers, David M W (2011). "Evaluation: From Precision, Recall and F-Measure to ROC. Journal of Machine Learning Technologies. 2 (1): 37–63.
5. Habibi, Aghdam, Hamed (2017-05-30). Guide to convolutional neural networks : a practical application to traffic-sign detection and classification. Heravi, Elnaz Jahani. Cham, Switzerland.
6. Exploring Convolutional Neural Network Architectures with fast.ai// TowardsdataScience [Электронный ресурс] Режим доступа: https://towardsdatascience.com/exploring-convolutional-neural-network-architectures-with-fast-ai-de4757eeeebf — Дата доступа: 20.05.2022
7. Deep Residual Learning for Image Recognition / Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun [Электронный ресурс] // ArXiv. —2015. — Режим доступа: https://arxiv.org/pdf/1512.03385.pdf — Дата доступа: 20.05.2022
8. NVIDIA, Vingelmann, P., & Fitzek, F. H. P. (2020). CUDA, release: 11.7.50. [Электронный ресурс] Режим доступа: https://developer.nvidia.com/cuda-toolkit — Дата доступа: 20.05.2022
9. Chollet, F. & others, 2015. Keras // Github [Электронный ресурс] Режим доступа: https://github.com/fchollet/keras. — Дата доступа: 20.05.2022
10. Martín Abadi [и др.]; TensorFlow: Large-scale machine learning on heterogeneous systems. TensorFlow [Электронный ресурс] — 2021. — Режим доступа: from tensorflow.org. — Дата доступа: 20.05.2022
11. Чучалин А. Г. Пневмония: актуальная проблема медицины XXI века. // Терапевтический архив. 2016;88(3):4-12. [Электронный ресурс] Режим доступа: https://doi.org/10.17116/terarkh20168834-12 — Дата доступа: 20.05.2022
12. High-Throughput Classification of Radiographs Using Deep Convolu- tional Neural Networks / Alvin Rajkomar, Sneha Lingam, Andrew G. Taylor, Michael Blum, and John Mongan / / Journal of Digital Imaging. — Feb 2017 — P. 30 - 95.

13. Nodule Classification Using Deep Feature Fusion in Chest Radiography / Changmiao Wang, Ahmed Elazab, Jianhuang Wu, and Qingmao Hu. Lung. / / Computerized Medical Imaging and Graphics: The Official Journal of the Computerized Medical Imaging Society — 2017 — P. 10 - 57.
14. Pneumonia. // World Health Organization. (2021, November 11). [Электронный ресурс] — 2021. — Режим доступа: https://www.who.int/news-room/fact-sheets/detail/pneumonia — Дата доступа: 20.05.2022