

МИНОБРНАУКИ РОССИИ  
Федеральное государственное бюджетное образовательное  
учреждение высшего образования  
«Ярославский государственный университет им. П.Г. Демидова»

Кафедра дискретного анализа

Сдано на кафедру  
«\_\_»\_\_\_\_\_ 2020 г.  
Заведующий кафедрой,  
д.ф.-м.н., профессор  
\_\_\_\_\_ В.А. Бондаренко

***Выпускная квалификационная работа***

**Анализ алгоритмов глубокого машинного обучения  
в задачах распознавания изображений**

по направлению  
01.03.02 Прикладная математика и информатика

Научный руководитель  
к.т.н., старший преподаватель  
\_\_\_\_\_ Д.В. Матвеев  
«\_\_»\_\_\_\_\_ 2020 г.

Студент группы ИВТ-41БО  
\_\_\_\_\_ А.С. Коротков  
«\_\_»\_\_\_\_\_ 2020 г.

Ярославль, 2020

## Реферат

Выпускная квалификационная работа: 22 стр., 3 гл., 10 рис., 5 таблиц, 18 источников.

Ключевые слова: **машинное обучение, глубокие нейронные сети, распознавание изображений, TensorFlow, Keras.**

Объектом исследования являются алгоритмы глубоких нейронных сетей для решения задач распознавания изображений.

Цель работы – изучить и проанализировать применение глубокого машинного обучения в задачах распознавания изображений.

В результате работы были разработаны и реализованы модели глубоких нейронных сетей для диагностирования пневмонии и, в частности, COVID-19 по рентгеновским снимкам. Проведен анализ полученных результатов и сделан вывод об эффективности алгоритмов глубокого машинного обучения в задачах распознавания изображений.

## Содержание

<b>Введение</b>	<b>4</b>
<b>1 Машинное обучение</b>	<b>6</b>
1.1 Понятие искусственной нейронной сети . . . . .	6
1.2 Активационная функция . . . . .	7
1.3 Обучение нейронных сетей . . . . .	9
1.4 Проблемы обучения глубоких нейронных сетей . . . . .	10
<b>2 Сверточные нейронные сети</b>	<b>12</b>
2.1 Архитектура . . . . .	12
2.2 VGG . . . . .	13
2.3 Inception . . . . .	14
2.4 ResNet . . . . .	15
2.5 DenseNet . . . . .	16
<b>3 Анализ алгоритмов</b>	<b>17</b>
3.1 Постановка задачи . . . . .	17
3.2 Средства реализации . . . . .	17
3.3 Обучающая выборка . . . . .	18
3.4 Эксперименты и результаты . . . . .	18
<b>Заключение</b>	<b>20</b>
<b>Список источников</b>	<b>21</b>

## Введение

В настоящее время, в связи со стремительным развитием цифровых технологий, использование автоматизированных и роботизированных систем распространилось на множество областей как в промышленности, науке, так и в повседневной жизни. В следствие этого, возрастает необходимость в эффективной обработке информации, представленной, в частности, в формате видео и изображений.

На текущий момент изображения тесно влились в жизнь человека. Поэтому многие автоматизированные системы используют их в качестве основного источника информации. Нахождение, локализация, классификация и анализ объектов на изображении компьютером – сложная задача компьютерного зрения.

Компьютерное зрение – это совокупность программно-технических решений в сфере искусственного интеллекта (ИИ), нацеленных на считывание и получение информации из изображений, в реальном времени и без участия человека.

В процессе обработки информации, получаемой из глаз, человеческий мозг продвигает колоссальный объем работы. Человек без труда сможет описать что находится и что происходит на случайно взятой фотографии. Изображения могут нести в себе колоссальное количество деталей и отличаться множеством параметров, таких как: разрешение, цветность, качество, яркость, наличие шума и т.д. Объекты на изображениях также могут обладать множеством особенностей: масштаб, положение, цвет, поворот, наклон и т.д. Однако, в цифровом формате, каждое изображение представляет собой лишь массив числовых данных. Научить компьютер находить и классифицировать образы на изображении с учетом всех факторов – очень сложная алгоритмическая задача. Для её решения активно применяют технологии машинного обучения.

Большое количество информации человек получает при помощи зрения. Изображения способны хранить огромное количество информации. Как следствие их использование в компьютерных системах способствует увеличению их производительности. Однако такие технологии требуют сложных вычислений. Задачей компьютерного зрения является разработка эффективных алгоритмов, выполняющих извлечение и анализ данных из изображений или видео.

В настоящий момент, подобные технологии применяются для решения таких сложных задач как:

- OCR – Optical character recognition (Оптическое распознавание символов): преобразование текста на изображении в редактируемый.
- Фотограмметрия – технология создания трехмерной модели объекта на основе фотографий, сделанных с различных ракурсов.
- Motion capture – технология, широко применяемая в киноиндустрии, позволяющая преобразовывать движения реальных людей в компьютерную анимацию.
- Дополненная реальность (AR) – технология, позволяющая в реальном времени проецировать виртуальные объекты на изображение реального окружения.
- Медицинская диагностика – обнаружение раковых клеток на ранней стадии, увеличение качества МРТ изображений, их анализ и т.д.

В данной работе был проведен анализ алгоритмов глубокого машинного обучения для решения задач распознавания изображений, а также спроектированы и обучены нейронные сети для диагностирования COVID-19 по рентгеновским снимкам грудной клетки.

Первая глава содержит основные сведения об машинном обучении.

Во второй главе содержится описание архитектуры сверточной нейронной сети и описание современных моделей на её основе.

В третьей главе проведено обучение современных моделей сверточных нейронных сетей для решения задачи диагностирования пневмонии по рентгеновским снимкам.

# 1 Машинное обучение

## 1.1 Понятие искусственной нейронной сети

Машинное обучение – раздел исследований в сфере ИИ, в основе которых лежат методы разработки систем способных к обучению. Алгоритмы машинного обучения эффективно себя показывают в задачах, в которых требуется у заранее подготовленных (обучающих) данных определить общие признаки и по ним идентифицировать новые данные. В проектировании таких, обучающихся, систем часто применяют искусственные нейронные сети.

Искусственная нейронная сеть (ИНС) – компьютерная модель, в основе которой лежат принципы работы биологической нейронной сети - совокупности связанных между собой нервных клеток - нейронов. Каждый нейрон имеет набор входных связей - синапсов, по которым он получает информацию, представленную в виде импульсов, от других нейронов. По полученным данным нейрон формирует своё состояние и, с помощью аксона, сообщает его другим нейронам, обеспечивая функционирование системы. В процессе формирования системы одни нейронные связи укрепляются, а другие ослабляются, обеспечивая обучаемость сети.



**Рис. 1** – Типичная структура биологического нейрона

Искусственный нейрон представляет собой упрощенную модель биологического нейрона. Принцип его работы представлен на рисунке 2. Сначала нейрон получает  $n$ -мерный вектор входных значений  $X = (x_1, \dots, x_n)$  и вектор весов  $W = (w_1, \dots, w_n)$ , обозначающий «укрепленность» межнейронных связей. Вычисляется сумма произведения входных значений и весов  $s_j$ . Затем к полученному результату применяется функция активации  $\varphi$ . В некоторых случаях, к сумме прибавляется величина смещения  $b_j$ .



**Рис. 2** – Схема искусственного нейрона

Множества нейронов формируют слои, слои в свою очередь формируют нейронную сеть. Входной слой получает данные, обрабатывает и передает нейронам скрытого слоя. Аналогично сработает каждый последующий слой вплоть до выходного.



**Рис. 3** – Схема простой нейронной сети

Нейросети с большим количеством скрытых слоев называется глубокими. Область машинного обучения, в которой используются глубокие нейронные сети называется - глубоким обучением.

## 1.2 Активационная функция

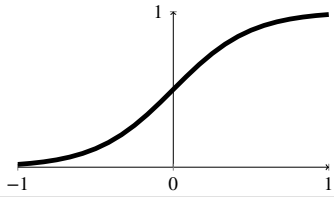
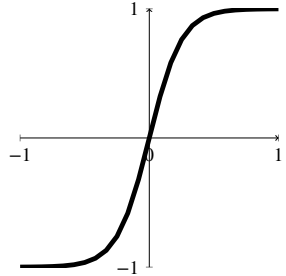
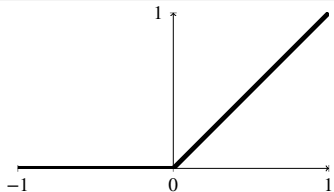
Взвешенная сумма входов представляет собой линейную комбинацию, из чего следует, что независимо от количества слоев, значения выходного слоя зависят только от входов первого слоя. Активационная функция нейрона обеспечивает нормализацию

посчитанной суммы и нелинейность нейронной сети. Для многих моделей нейронных сетей также требуется, чтобы активационная функция была монотонной и непрерывно-дифференцируемой на всей области определения.

Существует большое количество функций активации. Наиболее распространенные из них представлены в таблице 1.1.

Таблица 1.1

### Популярные активационные функции

Название	Функция	Вид
Сигмоидная	$\sigma(x) = \frac{1}{1+e^{-x}}$	
Гиперболический тангенс	$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	
ReLU	$f(x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0. \end{cases}$	

Отдельно стоит упомянуть функцию Softmax. Эта функция часто применяется на последнем слое глубоких нейронных сетей в задачах классификации. Пусть последний слой сети содержит  $N$  нейронов, каждый из которых соответствует некоторому классу. Тогда значение выхода  $i$ -го нейрона вычисляется по формуле:

$$y_i = \frac{e^{z_i}}{\sum_{j=1}^N e^{z_j}}$$

Таким образом, результат каждого нейрона будет принимать значения из диапазона  $[0, 1]$ , а их сумма равна 1. По итогу, сеть выдаст вероятности отношения входных данных к заданным классам.



### 1.3 Обучение нейронных сетей

Под обучением нейронных сетей подразумевается подбор значений весов связей для эффективного решения поставленной задачи. Изначально, веса устанавливаются случайно. Затем, в процессе прогона через сеть тестовых данных, веса корректируются так, чтобы в конечном итоге сеть выдавала правильные ответы.

В процессе обучения сети используются тренировочные наборы данных, которые разбиваются на пакеты меньшего размера. Проход через сеть всех пакетов называется эпохой.

Для того, чтобы контролировать процесс обучения необходимо как-то оценивать работу сети. Для этого вводится функция потерь (функция стоимости), которая вычисляет разницу между правильными и полученными результатами и формирует некоторое численное значение, характеризующее величину ошибки работы сети. Таким образом задача обучения сети сводится к задаче минимизации данной функции. В таблице 1.2 указаны наиболее часто используемые функции потерь, где  $y_i$  – ожидаемое значение  $i$ -го нейрона,  $x_i$  – полученное значение  $i$ -го нейрона,  $n$  – количество выходных нейронов.

Таблица 1.2

**Популярные функции потерь**

Название	Функция
Средняя квадратическая ошибка	$E = \frac{1}{N} \sum_{i=1}^n (y_i - x_i)^2$
Средняя абсолютная ошибка	$E = \frac{1}{N} \sum_{i=1}^n  y_i - x_i $
Верхняя граница	$E = \frac{1}{N} \sum_{i=1}^n \max(1 - x_i, y_i, 0)$
Перекрестная энтропия	$E = - \sum_{i=1}^n (x_i * \log(y_i))$

Одним из популярных методов в обучении глубоких нейронных сетей является алгоритм обратного распространения ошибки, основанный на градиентном спуске.

Пусть сеть имеет  $L$  слоев,  $a^l$ ,  $w^l$ ,  $b^l$  - векторы значений, весов и смещений нейронов на  $l$ -м слое.. Также имеется  $N$  обучающих пар  $(x, y)$ . В процессе обучения циклично происходят следующие итерации:

а) На вход сети подается вектор  $x$  из обучающего множества, для каждого слоя вычислить значения:

$$z^l = w^l a^{l-1} + b^l a^l = \sigma(z^l)$$

б) Вычислить значение функции стоимости:

$$C = \frac{1}{2} \sum_j (y_j - a_j^L)^2$$

в) Вычислить значения ошибок выходного слоя:

$$\delta_j^L = \frac{\delta C}{\delta a_j^L} \sigma'(z_j^L)$$

г) Вычислить ошибки для каждого предыдущего слоя:

$$\delta_j^l = \sum_k w_{kj}^{l+1} \delta_k^{l+1} \sigma'(z_j^l)$$

д) Вычислить градиент функции стоимости:

$$\frac{\delta C}{\delta w_{jk}^l} = a_k^{l-1} \delta_j^l$$

е) Обновить веса связей:

$$w_{ij}^l = w_{ij}^l - \mu \frac{\delta C}{\delta w_{jk}^l}, \quad 0 < \mu \leq 1$$

Данный метод относится к алгоритмам **обучения с учителем** - наиболее распространенному типу обучения, в котором сеть учится на заранее размеченных данных, где уже известны правильные ответы.

Существует и другие подходы к обучению нейронных сетей:

**Обучение с подкреплением** - метод, который подразумевает наличие некоторой окружающей среды в которой действует сеть. Такая среда реагирует на действия модели и подает ей определенные сигналы.

**Обучение без учителя** - обучение при котором сеть заранее не располагает правильными ответами и самостоятельно ищет общие и отличительные признаки у входных данных.

**Генетические алгоритмы обучения** - алгоритмы, имитирующие эволюционные механизмы развития биологической популяции, выступают как альтернатива алгоритму обратного распространения ошибки. Значение произвольного весового коэффициента в нейронной сети называется геном. Гены формируют хромосомы, а хромосомы - популяцию. Далее, в пределах одной эпохи с определенными вероятностями происходит:

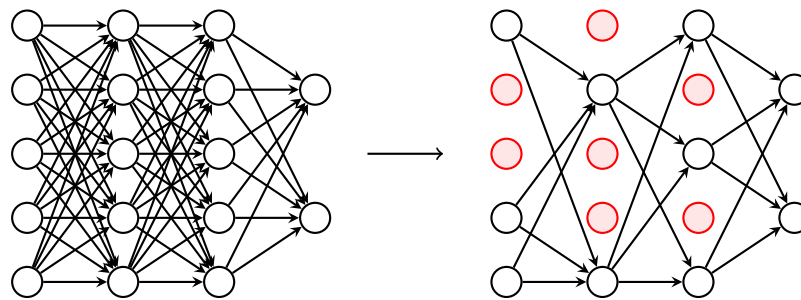
- Скрещивание хромосом - формирование новой хромосомы из генов двух других
- мутация - случайное изменение произвольного гена
- приспособление - (хромосомы показавшие худшие результаты уничтожаются из популяции).

## 1.4 Проблемы обучения глубоких нейронных сетей

В алгоритмах обучения на основе метода обратного распространения ошибки. Значение ошибки зависит от производной функции активации, так при использовании сигмоидной функции активации значение ошибки при распространении от последнего слоя к первому очень быстро уменьшается, тем самым веса на ранних слоях корректируются слабо. Аналогично можно столкнуться и с проблемой взрывного градиента, когда

значение ошибки становится очень большим. Простой способ решения такой проблемы - использование функции ReLU, производная которой принимает значения либо 0 либо 1.

Переобучение сети - проблема когда сеть обучается хорошо анализировать объекты только из обучающей выборки и плохо работает с новыми данными. Одним из методов решения этой проблемы является Dropout, суть которого заключается в следующем: На каждой итерации обучения нейроны с некоторой вероятностью выключаются. Для оставшихся нейронов происходит обучение методом обратного распространения ошибки, после чего нейроны возвращаются в сеть.



**Рис. 4 – Dropout**

## 2 Сверточные нейронные сети

### 2.1 Архитектура

Большая часть современных нейронных сетей направленных на анализ изображений базируются на архитектуре сверточной нейронной сети. Ранние нейронные сети состояли из полносвязных слоев - слоев, в которых каждый нейрон связан с каждым нейроном следующего слоя, что значительно увеличивало вычислительную сложность системы при увеличении количества нейронов. В типовых сверточных нейронных сетях преимущественно используются сверточные слои.

Сверточные слои характеризуются использованием матриц весов, называемых фильтрами или ядрами, которые обладают размерностью меньше исходных данных. Такое ядро с определенным шагом проходит по набору входных данных ( $I$ ) и вычисляет суммы произведений соответствующих значений ячеек и весов, формируя карту признаков ( $I * K$ ). Один сверточный слой может содержать несколько ядер и соответственно несколько карт признаков.

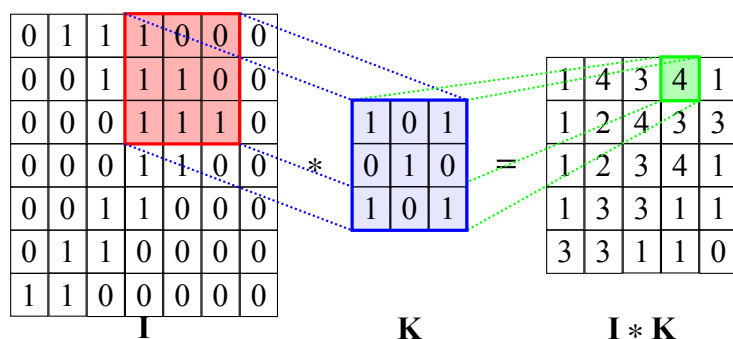
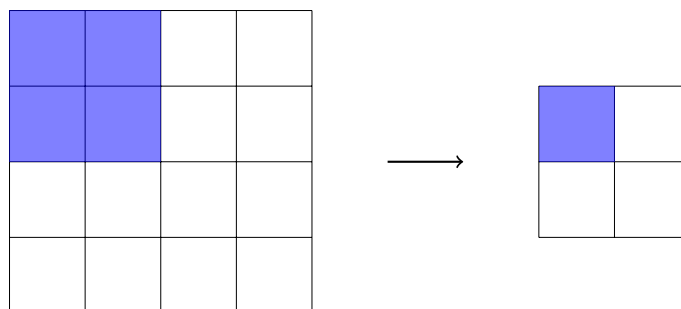


Рис. 5 – Операция свертки

Так как признаки уже обнаружены, для упрощения дальнейших вычислений можно снизить детализацию входных данных. Это обеспечивает субдискретизирующий (пулинговый) слой, уменьшая размерность входных карт признаков: из нескольких соседних нейронов берется максимальное или среднее значение, тем самым формируя нейрон карты признаков меньшей размерности. Это позволяет снизить количество параметров, используемых в дальнейших вычислениях сети.



**Рис. 6 – Субдискретизация**

Сверточная нейронная сеть может иметь несколько пар чередующихся сверточных и субдискретизирующих слоев. Таким образом, на примере изображений, на начальных слоях сеть находит такие простейшие признаки как границы и углы. Затем, по мере углубления в сеть, определяются всё более сложные конструкции: от простейших фигур до целых классов, независимо от их местоположения на изображении. Завершается сеть стандартными полносвязными слоями которые сопоставляют полученные карты признаков какому-либо классу.

## 2.2 VGG

Архитектура VGG была предложена в 2014 году[9]. Главной особенностью сети стало использование подряд стоящих сверточных слоев с фильтрами размерности 3x3 вместо применяемых ранее сверточных слоев с фильтрами большого размера 5x5, 7x7, 11x11.

В таблице 2.1 указаны различные конфигурации VGG, наиболее известные из них VGG-16 (D) и VGG-19 (E), названные по количеству слоев, содержащих веса. Maxpool - субдискретизирующий слой с функцией максимума размера 2x2. FC - полносвязный слой. Во всех скрытых слоях используется активационная функция ReLU.

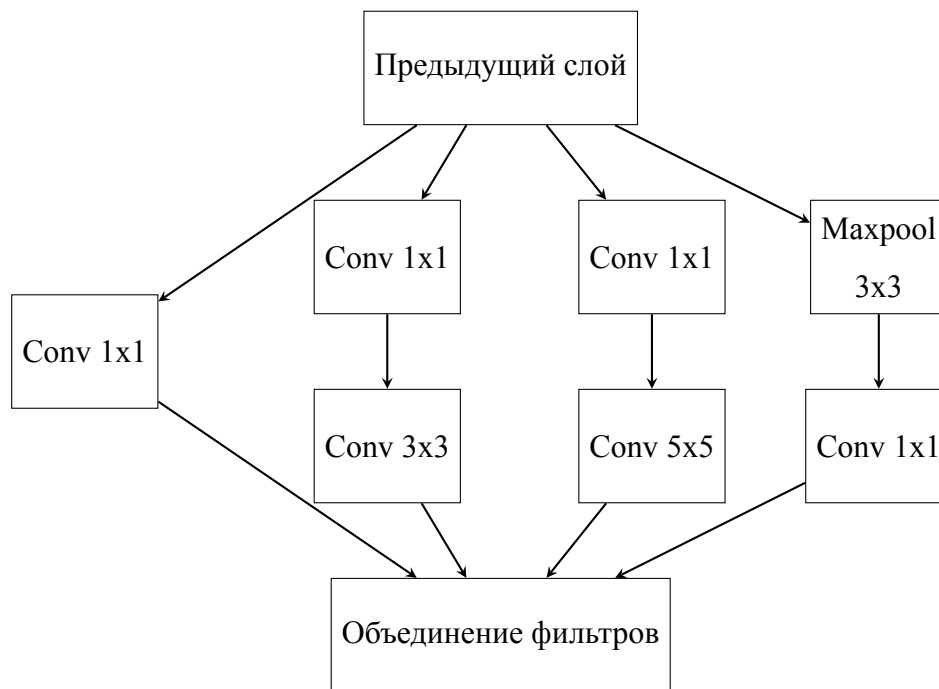
## Конфигурации VGG

A	A-LRN	B	C	D	E
Вход (224 × 224 RGB)					
conv3-64	conv3-64 <b>LRN</b>	conv3-64 <b>conv3-64</b>	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
Maxpool					
conv3-128	conv3-128	conv3-128 <b>conv3-128</b>	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
Maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 <b>conv1-256</b>	conv3-256 conv3-256 <b>conv3-256</b>	conv3-256 conv3-256 conv3-256 <b>conv3-256</b>
Maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 <b>conv1-512</b>	conv3-512 conv3-512 <b>conv3-512</b>	conv3-512 conv3-512 conv3-512 <b>conv3-512</b>
Maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 <b>conv1-512</b>	conv3-512 conv3-512 <b>conv3-512</b>	conv3-512 conv3-512 conv3-512 <b>conv3-512</b>
Maxpool					
FC-4096					
FC-4096					
FC-1000					
Softmax					

## 2.3 Inception

Данная модель, разработанная компанией Google, в 2014 году заняла 1 место в ежегодном конкурсе по классификации изображений - ILSVRC[11]. Ключевым нововведением данной сети стало использование в качестве слоев вложенных модулей,

которые представляют из себя набор фильтров разных размерностей, с последующим объединением их результатов.

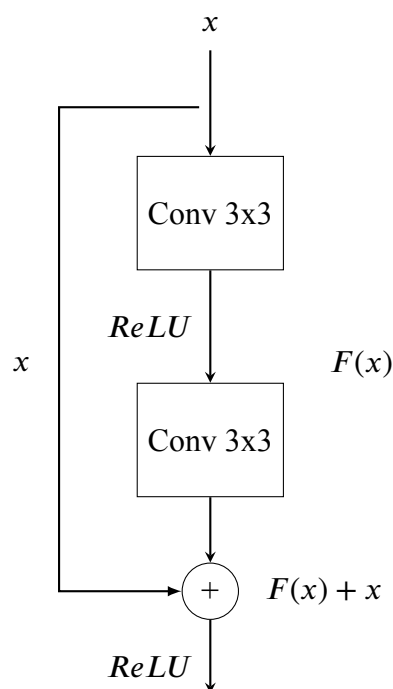


**Рис. 7** – Inception модуль

Также, в Inception полностью отказались от использования полносвязных слоев, вместо них применяется глобальный средний пулинг, который преобразует каждую карту признаков к одному числу, формируя вектор усредненных значений. Такое нововведение позволило значительно уменьшить количество параметров и как следствие вычислительную сложность сети.

## 2.4 ResNet

ResNet[4], также известная как остаточная нейронная сеть, выиграла ILSVRC в 2015 году. Её особенностью было наличие пропускающих соединений, которые передают информацию без изменений на более глубокие участки сети, эта информация суммируется с вычисленным на пропущенных слоях значением и передается дальше. Блок изображенный на рис. 8 демонстрирует составной элемент такой сети.



**Рис. 8** – Блок остаточной сети

## 2.5 DenseNet

DenseNet[5] - плотная сверточная сеть, похожая на ResNet, но с той разницей, что все блоки сети соединены прямыми связями между собой, таким образом каждый блок получает информацию от всех предыдущих.

Таблица 2.2

**Модели DenseNet**

Слой	Выходной размер	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Сверточный	$112 \times 112$	$7 \times 7$ свертка, шаг 2			
Пулинг	$56 \times 56$	$3 \times 3$ максимальный пулинг, шаг 2			
Плотный блок (1)	$56 \times 56$	$\begin{bmatrix} 1 \times 1 \text{ свертка} \\ 3 \times 3 \text{ свертка} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ свертка} \\ 3 \times 3 \text{ свертка} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ свертка} \\ 3 \times 3 \text{ свертка} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ свертка} \\ 3 \times 3 \text{ свертка} \end{bmatrix} \times 6$
Промежуточный слой (1)	$56 \times 56$	$1 \times 1$ свертка			
	$28 \times 28$	$2 \times 2$ средний пулинг, шаг 2			
Плотный блок (2)	$28 \times 28$	$\begin{bmatrix} 1 \times 1 \text{ свертка} \\ 3 \times 3 \text{ свертка} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ свертка} \\ 3 \times 3 \text{ свертка} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ свертка} \\ 3 \times 3 \text{ свертка} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ свертка} \\ 3 \times 3 \text{ свертка} \end{bmatrix} \times 12$
Промежуточный слой (2)	$28 \times 28$	$1 \times 1$ свертка			
	$14 \times 14$	$2 \times 2$ средний пулинг, шаг 2			
Плотный блок (3)	$14 \times 14$	$\begin{bmatrix} 1 \times 1 \text{ свертка} \\ 3 \times 3 \text{ свертка} \end{bmatrix} \times 24$	$\begin{bmatrix} 1 \times 1 \text{ свертка} \\ 3 \times 3 \text{ свертка} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ свертка} \\ 3 \times 3 \text{ свертка} \end{bmatrix} \times 48$	$\begin{bmatrix} 1 \times 1 \text{ свертка} \\ 3 \times 3 \text{ свертка} \end{bmatrix} \times 64$
Промежуточный слой (3)	$14 \times 14$	$1 \times 1$ свертка			
	$7 \times 7$	$2 \times 2$ средний пулинг, шаг 2			
Плотный блок (4)	$7 \times 7$	$\begin{bmatrix} 1 \times 1 \text{ свертка} \\ 3 \times 3 \text{ свертка} \end{bmatrix} \times 16$	$\begin{bmatrix} 1 \times 1 \text{ свертка} \\ 3 \times 3 \text{ свертка} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ свертка} \\ 3 \times 3 \text{ свертка} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ свертка} \\ 3 \times 3 \text{ свертка} \end{bmatrix} \times 48$
Слой классификации	$1 \times 1$	$7 \times 7$ глобальный средний пулинг			
		1000D полносвязный слой, softmax			

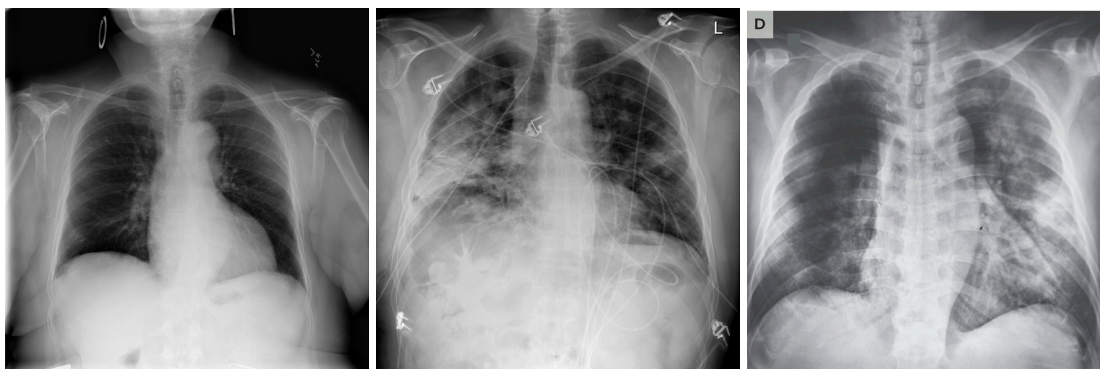


### 3 Анализ алгоритмов

#### 3.1 Постановка задачи

COVID-19 - тяжелая респираторная инфекция, вызываемая коронавирусом SARS-CoV-2. На момент написания работы, вспышка вируса, возникшая в китайском городе Ухань, переросла в глобальную пандемию. Решение задачи автоматической диагностики данного заболевания позволит снизить нагрузку на врачей и повысит эффективность их работы.

Пневмония, которую порождает COVID-19, чаще всего является двусторонней и имеет периферическую локализацию, что позволяет визуально отличить её от стандартной пневмонии.



а. Норма

б. Пневмония

с. COVID-19

**Рис. 9** – Рентгеновские снимки грудных клеток

#### 3.2 Средства реализации

cuDNN - библиотека глубоких нейронных сетей от Nvidia позволяющая использовать для вычислений мощности графического процессора.

Python 3 - гибкий и мощный язык программирования, эффективно выполняющий задачи анализа и обработки данных.

TensorFlow - многофункциональный фреймворк с открытым исходным кодом, разработанный компанией Google, позволяющий проектировать и обучать различные архитектуры нейронных сетей.

Keras - высокоуровневый API для решения задач глубокого машинного обучения, входящий в состав TensorFlow.

### 3.3 Обучающая выборка

Сбор данных для обучения нейронных сетей в задачах такого типа - сложный и долгий процесс, который требует затрат большого количества времени и участие большого количества людей. Поэтому, в качестве источника обучающих и тестовых данных использовались готовые, уже размеченные датасеты: [1], [2] и [12]

Снимки из обучающей выборки имеют разное разрешение, однако нейронные сети требуют заранее установленное количество входных нейронов. Поэтому, в качестве предварительной обработки данных, все изображения перед подачей на сеть масштабируются к одному разрешению: 512x512 px.

### 3.4 Эксперименты и результаты

В ходе экспериментов были модернизированы для задачи, обучены и протестированы следующие модели нейронных сетей: Inception-V3, ResNet-50 и DenseNet-201. Все модели обучались на общем наборе тренировочных и проверочных данных и с одинаковыми параметрами:

- Размер одного пакета - 8 изображений
- Количество эпох - 10

Точность сетей в процессе обучения и тестирования показана на рис. 10. Финальные результаты экспериментов указаны в таблице 3.1

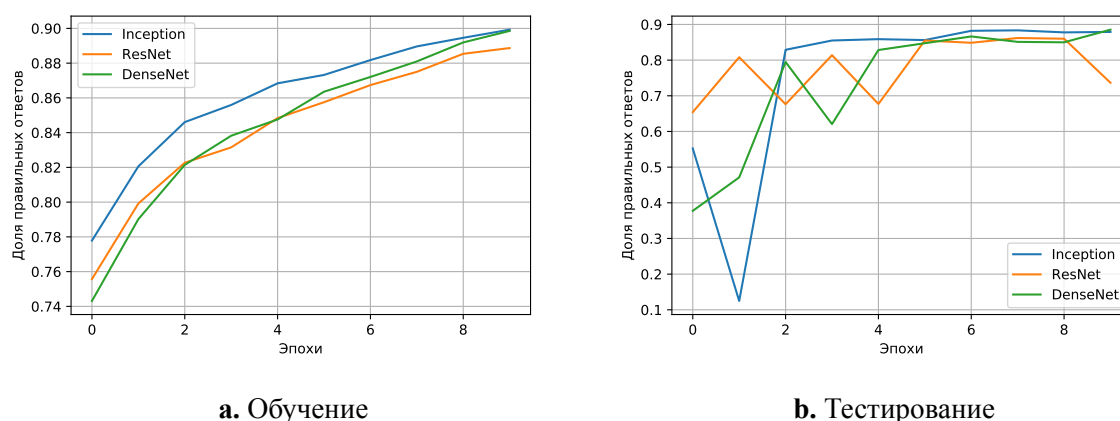


Рис. 10 – Точность сетей

**Результаты экспериментов**

Сеть	Точность при обучении	Точность на проверочных данных
Inception	89.93%	87.90%
ResNet	88.87%	73.65%
DenseNet	89.86%	88.54%

## **Заключение**

В данной работе было проведено исследование алгоритмов глубокого машинного обучения в задачах распознавания изображений. Были рассмотрены, обучены и протестированы различные архитектуры сверточных нейронных сетей на задаче диагностирования пневмонии и COVID-19 по рентгеновским снимкам грудной клетки.

Результаты исследования показывают, что глубокие нейронные сети эффективно справляются с распознаванием изображений. Самую высокую точность показала плотная сверточная нейронная сеть - DenseNet с точностью 88.54%. Расширение обучающих данных, увеличение параметров моделей и большее количество эпох может значительно повысить качество распознавания изображений.

## Список источников

1. *Chowdhury M., Rahman T., Khandakar A., Mazhar R., Kadir M., Mahbub Z., Islam K., Khan M., Iqbal A., Al-Emadi N., Reaz M.* Can AI help in screening Viral and COVID-19 pneumonia? // arXiv preprint. — 2020. — URL: <https://www.kaggle.com/tawsifurrahman/covid19-radiography-database>.
2. *Cohen J. P., Morrison P., Dao L.* COVID-19 image data collection // arXiv 2003.11597. — 2020. — URL: <https://github.com/ieee8023/covid-chestxray-dataset>.
3. *Farinella G. M., Battiato S., Cipolla R.* Advanced Topics in Computer Vision. — Springer, 2013. — С. 475.
4. *He K., Zhang X., Ren S., Sun J.* Deep Residual Learning for Image Recognition. — 2015. — arXiv: 1512.03385.
5. *Huang G., Liu Z., Maaten L. van der, Weinberger K. Q.* Densely Connected Convolutional Networks. — 2016. — arXiv: 1608.06993.
6. *Jamil M., Sharma S., Singh R.* Fault detection and classification in electrical power transmission system using artificial neural network // SpringerPlus. — 2015. — Июль. — Т. 4. — С. 334. — DOI: 10.1186/s40064-015-1080-x.
7. *Michelucci U.* Applied Deep Learning: A Case-Based Approach to Understanding Deep Neural Networks. — Apress, 2018. — С. 431.
8. *Nielsen M. A.* Neural Networks and Deep Learning. / Determination Press. — 2015. — URL: <http://neuralnetworksanddeeplearning.com/>.
9. *Simonyan K., Zisserman A.* Very Deep Convolutional Networks for Large-Scale Image Recognition. — 2014. — arXiv: 1409.1556.
10. *Singh P., Manure A.* Learn TensorFlow 2.0: Implement Machine Learning and Deep Learning Models with Python. — Apress, 2020. — С. 195.
11. *Szegedy C., Vanhoucke V., Ioffe S., Shlens J., Wojna Z.* Rethinking the Inception Architecture for Computer Vision. — 2015. — arXiv: 1512.00567.
12. *Wang L., Lin Z. Q., Wong A.* COVID-Net: A Tailored Deep Convolutional Neural Network Design for Detection of COVID-19 Cases from Chest Radiography Images. — 2020. — arXiv: 2003.09871 [cs.CV].

13. *Жерон О.* Прикладное машинное обучение с помощью Scikit-Learn и TensorFlow. — ООО "Альфа-книга", 2018. — С. 688.
14. *Клетте Р.* Компьютерное зрение. Теория и алгоритмы. — ДМК Пресс, 2019. — С. 506.
15. *Конец Д.* Классические задачи Computer Science на языке Python. — Питер, 2020. — С. 256.
16. *Николенко С., Кадури А., Архангельская Е.* Глубокое обучение. — Питер, 2018. — С. 480.
17. *Нишант Ш.* Машинное обучение и TensorFlow. — Питер, 2019. — С. 336.
18. *Шолле Ф.* Глубокое обучение на Python. — Питер, 2018. — С. 400.