

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: 26 стр., 3 гл., 9 рис., 10 таблиц, 19 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 алгоритмов глубокого машинного обучения в задачах обработки рентгеновских снимков у больных с подозрением на COVID-19.

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

Введение	4
1 Машинное обучение	6
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	11
2 Сверточные нейронные сети	13
2.1 Архитектура	13
2.2 VGG	14
2.3 Inception	15
2.4 ResNet	16
2.5 DenseNet	17
2.6 Тестирование	18
3 Анализ алгоритмов	19
3.1 Постановка задачи	19
3.2 Средства реализации	19
3.3 Оценка качества	20
3.4 Обучающая выборка	20
3.5 Предварительная обработка изображений	21
3.6 Выбор оптимизатора	22
3.7 Результаты	23
Заключение	24
Список источников	25

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.

Человек получает большое количество информации при помощи зрения. Изображения способны хранить огромное её количество. Как следствие их использование в компьютерных системах способствует увеличению производительности этих систем. 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 и COVID-19 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 произведений входных значений и весов s_j . Затем к полученному результату применяется функция активации φ . Дополнительно, к сумме может прибавляться величина смещения b_j .

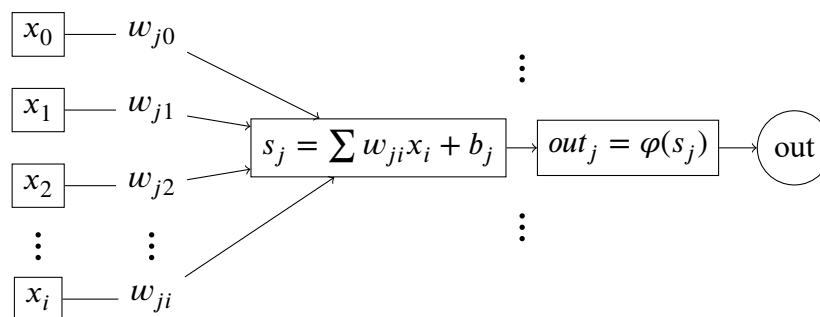


Рис. 2 – Схема искусственного нейрона

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

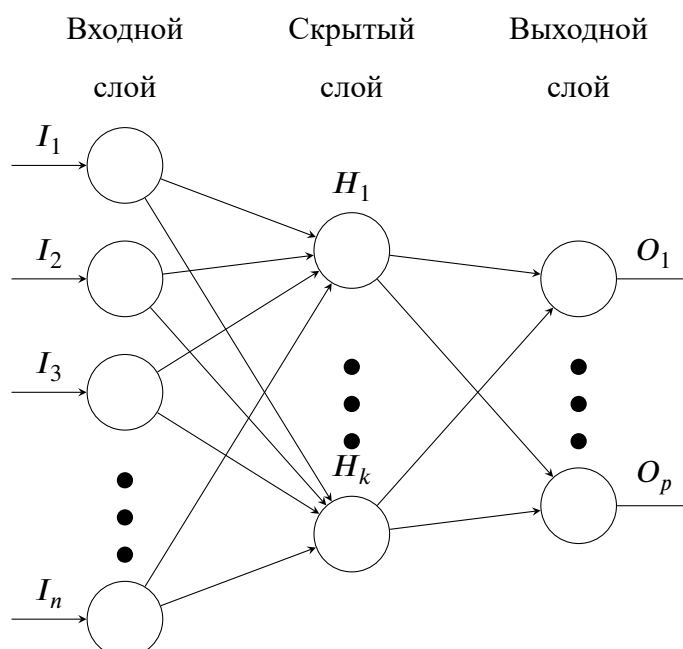


Рис. 3 – Схема простой нейронной сети

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

1.2 Активационная функция

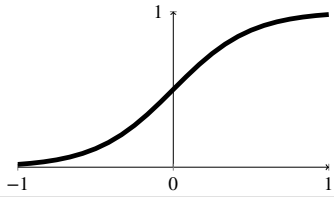
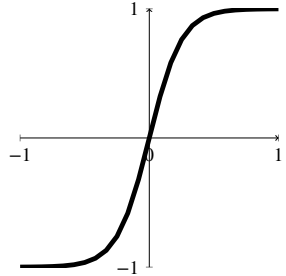
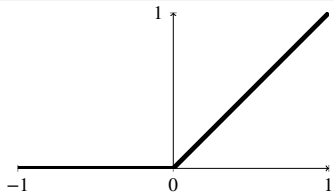
Взвешенная сумма входов представляет собой линейную комбинацию, из чего следует, что независимо от количества слоев, значения выходного слоя зависят только от входов первого слоя. Активационная функция нейрона обеспечивает нормализацию

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.

Процесс обучения проходит циклично. За одну итерацию на сеть подается пакет, содержащий некоторое количество элементов из входных данных. Разовый проход через сеть всего набора тестовых данных 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 поиске глобального минимума 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	$E = \frac{1}{N} \sum_{i=1}^n \max(1 - x_i, y_i, 0)$
Категориальная перекрестная энтропия	$E = - \sum_{i=1}^n (x_i \cdot \log(y_i))$

Одним из популярных методов в обучении глубоких нейронных сетей является алгоритм обратного распространения ошибки.

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

По мимо описанного выше метода для обучения часто применяются другие алгоритмы, например RMSprop и Adam:

Алгоритм 1. RMSProp

Вход : Начальное значение x_1 , параметр $\beta_2 \in [0, 1)$, шаг α , константа ϵ

Function `RMSProp` ($x_1, \beta_2, \alpha, \epsilon$)

```

     $v_0 = 0;$ 
    for  $t = 1, 2, \dots$  do
         $g_t = \nabla f(x_t)$ 
         $v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$ 
         $x_{t+1} = x_t - \frac{\alpha}{\sqrt{v_t} + \epsilon} g_t$ 
    end

```

end

Алгоритм 2. Adam

Вход : Начальное значение x_1 , параметры $\beta_1, \beta_2 \in [0, 1)$, шаг α , константа ϵ

Function `Adam` ($x_1, \beta_1, \beta_2, \alpha, \epsilon$)

```

     $v_0 = 0; m_0 = 0;$ 
    for  $t = 1, 2, \dots$  do
         $g_t = \nabla f(x_t)$ 
         $m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$ 
         $v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$ 
         $x_{t+1} = x_t - \alpha \frac{m_t}{\sqrt{v_t} + \epsilon} g_t$ 
    end

```

end

Данные методы относятся 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. Для решения подобных проблем можно прибегнуть к предварительной обработке входных данных, зачастую входные данные рекомендуется ограничить диапазоном $[0; 1]$. К примеру, в изображениях значения могут находиться в диапазоне от 0 до 255 и для лучшей стабильности обучения их можно поделить на 255. В качестве ещё одного способа оптимизации можно провести центрирование значений - вычислить для входного изображения среднее значение и вычесть его из каждого пикселя, тем самым среднее значение данных в изображении будет

равно 0. Вместе с этим способом часто применяют нормализацию среднеквадратичного отклонения, устанавливая его значение в 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.

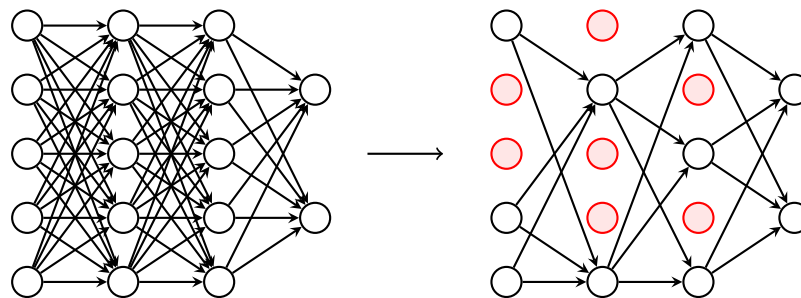


Рис. 4 – Dropout

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.

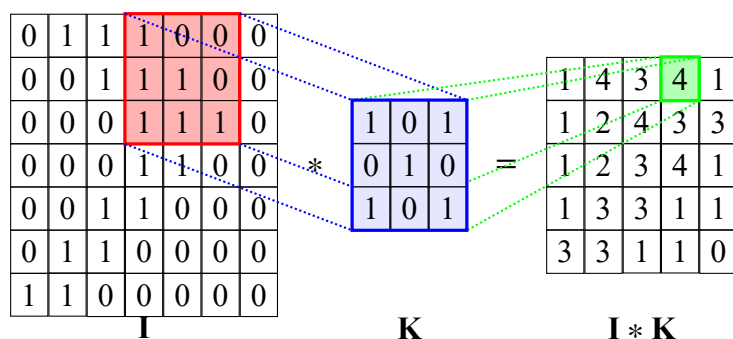


Рис. 5 – Операция свертки

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.

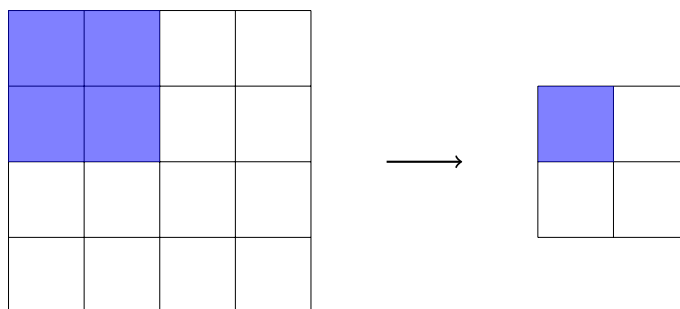


Рис. 6 – Субдискретизация

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[10]. 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. Это позволило снизить количество параметров сети с сохранением эффективности.

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 LRN	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
Maxpool					
conv3-128	conv3-128	conv3-128 conv3-128	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 conv1-256	conv3-256 conv3-256 conv3-256	conv3-256 conv3-256 conv3-256 conv3-256
Maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
Maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
Maxpool					
FC-4096					
FC-4096					
FC-1000					
Softmax					

2.3 Inception

Данная модель[12], разработанная компанией Google, в 2014 году заняла 1 место в ежегодном конкурсе по классификации изображений - ILSVRC. Ключевым нововведением данной сети стало использование в качестве слоев вложенных модулей, которые пред-

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

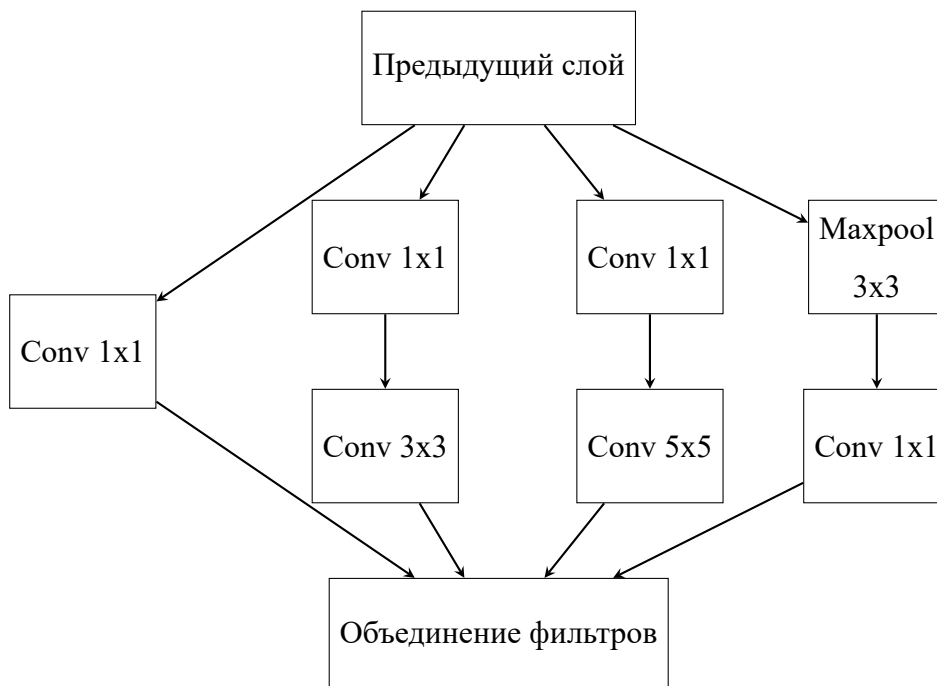


Рис. 7 – Inception модуль

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. В последствии были разработаны улучшенные версии Inception, в которых заменили слой 5x5 двумя последовательными слоями 3x3, а также все слои с фильтрами размера NxN заменили на стек фильтров 1xN и Nx1, что также позволило снизить количество параметров.

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.

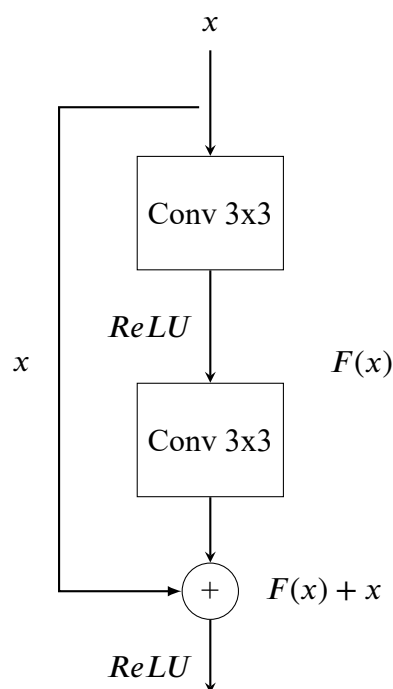


Рис. 8 – Блок остаточной сети

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

2.6 Тестирование

Все указанные модели тестировались на наборах данных ImageNet, который включает в себя 1000 классов, свыше 1.3 млн. тренировочных и 50 тыс. проверочных изображений. Так как сети выполняют задачу классификации и используют на конце Softmax-слой, результатом сети является вектор (x_1, x_2, \dots, x_n) , где n - количество классов, а x_i - вероятность отношения входного изображения к i -му классу. Точность сетей измерялась в двух вариантах Top-1 и Top-5. Top-1 означает, что наибольшее x_i соответствует правильному классу, а Top-5 - что правильный класс принадлежит пяти самым высоким значениям в выходном векторе сети. В таблице 2.3 указаны результаты проведенных проверок по метрике ассигасу - отношение доли правильных ответов к общему их количеству.

Таблица 2.3

Результаты тестирования моделей

Сеть	Кол-во параметров	Top-1	Top-5
VGG-16	138 357 544	71.3%	90.1%
VGG-19	143 667 240	71.3%	90.0%
Inception V3	23 851 784	77.9%	93.7%
ResNet-50 V2	25 613 800	76.0%	93.0%
ResNet-101 V2	44 675 560	77.2%	93.8%
ResNet-152 V2	60 380 648	78.0%	94.2%
DenseNet-121	8 062 504	75.0%	92.3%
DenseNet-169	14 307 880	76.2%	93.2%
DenseNet-201	20 242 984	77.3%	93.6%

Как можно видеть на таблице: сети VGG имеют самое большое количество параметров, значительно превосходя другие модели, при этом демонстрируют самую низкую точность. Сети DenseNet показывают лучшее соотношение точности и количества параметров. Однако третья версия Inception имеет относительно немного больше параметров и точность. Лучший результат показала ResNet-152 второй версии, но она обладает значительно большим количеством параметров, что сказывается на времени обучения сети.

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. Основным способом диагностики COVID-19 является метод полимеразной цепной реакции (ПЦР) с обратной транскрипцией, однако это трудоемкий, требующий участия человека процесс. Альтернативой является рентгенологическое исследование, например рентген грудной клетки или компьютерная томография (КТ). Исследования показывают, что большинство снимков пациентов с COVID-19 содержат специфические аномалии, такие как двустороннее ретикулярное узловое затемнение или синдром матового стекла, что позволяет визуально отличить заболевание от других видов пневмоний [13]. Несмотря на то, что рентген обладает меньшей чувствительностью чем КТ или ПЦР, это гораздо более доступный и быстрый метод диагностики, что является существенным критерием в период пандемии. Решение задачи автоматической диагностики данного заболевания позволит снизить нагрузку на врачей и повысит эффективность их работы.

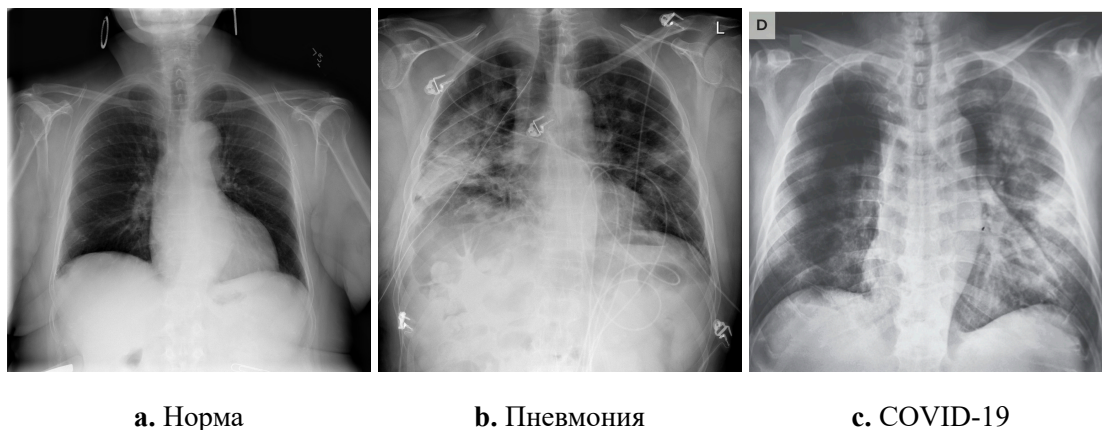


Рис. 9 – Рентгеновские снимки грудных клеток

3.2 Средства реализации

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 Оценка качества

Для оценки качества работы алгоритмов использовались следующие метрики:

- Precision (Точность):

$$P = \frac{TP}{TP + FP}$$

- Recall (Полнота):

$$R = \frac{TP}{TP + FN}$$

- F1-мера:

$$F1 = 2 \cdot \frac{P \cdot R}{P + R}$$

где TP - количество истинно-положительных, TN - истинно-отрицательных, FN - ложно-отрицательных ответов.

Precision обозначает долю правильно идентифицированных объектов класса относительно всех объектов причисленных к этому классу.

Recall показывает долю найденных сетью элементов класса относительно всех элементов этого класса.

F1 объединяет precision и recall вычисляя их гармоническое среднее. Также в процессе обучения сети в качестве характеристики качества обучения рассматривались значения функции потерь.

3.4 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 проверочных данных использовались готовые, уже размеченные датасеты: [1], [2] и [13]. Всего было собрано 14 197 снимков, из них 8 066 здоровых пациентов, 5 558 с пневмонией и 573 с COVID-19. По 100 изображений каждого класса было отобрано для валидации обучения.

3.5 Предварительная обработка изображений

При использовании глубоких нейронных сетей в задачах классификации изображений требуется проведение дополнительных исследований, с целью выбора оптимальных параметров различных частей алгоритмов.

Предварительная обработка входных данных может существенно повлиять на обучение моделей. В рамках данной части исследования рассматривались следующие варианты обработки изображений:

- Масштабирование - деление всех значений в изображении на 255.
- Центрирование среднего значения изображения в 0 и нормализация среднеквадратичного отклонения в 1.

В исследовании были протестированы следующие модели нейронных сетей со стандартными параметрами:

- Inception V3, размерность входного слоя: 299x299
- ResNet-50, размерность входного слоя: 224x224
- DenseNet-201, размерность входного слоя: 224x224

Обучение всех моделей проходило по 10 эпох, размер одного пакета - 16 изображений. В качестве функции потерь использовалась категориальная перекрестная энтропия, в качестве оптимизатора - Адам. Значения функции ошибки (loss), метрик precision и recall по итогам обучения и валидации (приставка «val_») для обработанных изображений методами масштабирования и центрирования указаны в таблицах 3.1 и 3.2 соответственно.

Таблица 3.1

Результаты обучение моделей со стандартными параметрами и предварительным масштабированием изображений

Сеть	loss	precision	recall	val_loss	val_precision	val_recall
Inception V3	0.2352	0.9176	0.9106	0.2737	0.7884	0.7700
ResNet-50	0.3242	0.8870	0.8732	0.2373	0.7354	0.7133
DenseNet-201	0.2742	0.9054	0.8969	0.3196	0.7560	0.7333

Таблица 3.2

**Результаты обучение моделей со стандартными параметрами
и предварительным центрированием изображений**

Сеть	loss	precision	recall	val_loss	val_precision	val_recall
Inception V3	0.3387	0.8860	0.8697	0.5269	0.7204	0.6700
ResNet-50	0.3353	0.8848	0.8705	0.9311	0.6537	0.6167
DenseNet-201	0.3655	0.8742	0.8622	0.3539	0.7643	0.7133

Как видно из таблиц предварительное масштабирование значений дает значения метрик лучше чем центрирование в процессе обучения и валидации. При этом самые высокие результаты в данном тестировании показала сеть Inception V3.

3.6 Выбор оптимизатора

На качество работы нейронных сетей может существенно влиять выбранный алгоритм оптимизации функции ошибки. В данной части исследования проводится анализ различных оптимизаторов: Стохастический градиентный спуск (SGD), RMSProp и Adam. Тестирование проводилось на сетях Inception V3, ResNet-50, DenseNet-201 и предварительно масштабированными входными изображениями размером 500x500 пикселей.

Все модели обучались по 10 эпох, размер одного пакета - 8 изображений. В таблицах 3.3, 3.4 и 3.5 указаны полученные в итоге при валидации значения метрик Precision, Recall и F1 для каждого класса.

Таблица 3.3

Итоги обучения сетей с оптимизатором Adam

	Inception V3			ResNet-50 V2			DenseNet-201		
	precision	recall	f1-score	precision	recall	f1-score	precision	recall	f1-score
COVID-19	0.53	0.38	0.44	0.55	0.40	0.46	0.52	0.56	0.54
Normal	0.51	0.58	0.54	0.51	0.53	0.52	0.53	0.51	0.52
Pneumonia	0.59	0.68	0.63	0.52	0.63	0.57	0.53	0.51	0.52

Таблица 3.4

Итоги обучения сетей с оптимизатором RMSprop

	Inception V3			ResNet-50 V2			DenseNet-201		
	precision	recall	f1-score	precision	recall	f1-score	precision	recall	f1-score
COVID-19	0.57	0.51	0.54	0.56	0.84	0.67	0.53	0.23	0.32
Normal	0.52	0.70	0.60	0.63	0.36	0.46	0.56	0.65	0.60
Pneumonia	0.53	0.58	0.55	0.54	0.49	0.51	0.55	0.77	0.64

Таблица 3.5

Итоги обучения сетей с оптимизатором SGD

	Inception V3			ResNet-50 V2			DenseNet-201		
	precision	recall	f1-score	precision	recall	f1-score	precision	recall	f1-score
COVID-19	0.63	0.50	0.56	0.50	0.58	0.54	0.33	0.88	0.48
Normal	0.60	0.64	0.62	0.56	0.55	0.55	0	0	0
Pneumonia	0.67	0.64	0.65	0.52	0.45	0.48	0.33	0.11	0.17

3.7 Результаты

Таблицы 3.3, 3.4 и 3.5 показывают, что различные оптимизаторы по разному эффективны для различных сетей, так сеть ResNet-50 V2 вместе с алгоритмом оптимизации RMSProp по метрике Recall смогла идентифицировать большее количество изображений с COVID-19, при этом Inception V3 с методом SGD, в среднем по классам показывает самые высокие результаты, согласно метрике F1.

Таким образом, в ходе проведенного исследования было выявлено, что для решения задачи диагностики COVID-19 по рентгеновским снимкам предпочтительно использовать сеть Inception в совокупности с алгоритмом оптимизации SGD. При этом значения во входных изображениях рекомендуется приводить к диапазону [0;1] посредством масштабирования.

Заключение

В данной работе было проведено исследование алгоритмов глубокого машинного обучения в задачах распознавания изображений.

Были рассмотрены такие архитектуры сверточных нейронных сетей, как Inception, ResNet и DenseNet. Также были обучены и протестированы модели для решения задачи диагностирования пневмонии и COVID-19 по рентгеновским снимкам грудной клетки.

Результаты исследования показали, что с решением данной задачи лучше справляется сеть Inception V3 вместе с алгоритмом оптимизации SGD. Предварительное приведение значений к диапазону $[0;1]$ может повысить качество обучения сети.

Расширение обучающих данных и большее количество эпох может значительно повысить качество распознавания изображений.

Список источников

1. *Chowdhury M., Rahman T., Khandakar A.* [и др.]. Can AI help in screening Viral and COVID-19 pneumonia? — 2020. — URL: <https://www.kaggle.com/tawsifurrahman/covid19-radiography-database>.
2. *Cohen J. P., Morrison P., Dao L.* COVID-19 image data collection. — 2020. — arXiv: 2003.11597. — 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. *Khan A., Sohail A., Zahoora U., Qureshi A. S.* A survey of the recent architectures of deep convolutional neural networks. — 2020. — arXiv: 1901.06032.
8. *Michelucci U.* Applied Deep Learning: A Case-Based Approach to Understanding Deep Neural Networks. — Apress, 2018. — С. 431.
9. *Nielsen M. A.* Neural Networks and Deep Learning. — 2015. — URL: <http://neuralnetworksanddeeplearning.com/>.
10. *Simonyan K., Zisserman A.* Very Deep Convolutional Networks for Large-Scale Image Recognition. — 2014. — arXiv: 1409.1556.
11. *Singh P., Manure A.* Learn TensorFlow 2.0: Implement Machine Learning and Deep Learning Models with Python. — Apress, 2020. — С. 195.
12. *Szegedy C., Vanhoucke V., Ioffe S.* [и др.]. Rethinking the Inception Architecture for Computer Vision. — 2015. — arXiv: 1512.00567.

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