

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ  
УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ  
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ИНСТИТУТ  
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Проектная работа

Telegram-бот, определяющий наличие  
объекта на изображении

Студенты:  
Брагинец Анастасия Владимировна  
Варфоломеева Анастасия Андреевна  
Гусев Дмитрий Олегович

Руководитель проекта:  
Дёмин Дмитрий Олегович

Группа:  
БПМ215

Москва  
2022

# Содержание

<b>1</b>	<b>Актуальность программы</b>	<b>3</b>
<b>2</b>	<b>Нейронные сети</b>	<b>4</b>
2.1	Что такое машинное обучение и как оно связано с нейронными сетями	4
2.2	Классификация задач машинного обучения . . . . .	5
2.3	Компьютерное зрение и его необходимые составляющие . . . . .	6
2.4	Сверточная нейронная сеть . . . . .	7
<b>3</b>	<b>Предварительно обученная нейронная сеть VGG16</b>	<b>8</b>
3.1	Общие сведения о НС VGG16 . . . . .	8
3.2	Данные для обучения . . . . .	8
3.3	Конфигурация . . . . .	10
<b>4</b>	<b>Функции активации</b>	<b>11</b>
4.1	ReLU . . . . .	11
<b>5</b>	<b>Телеграм-бот</b>	<b>12</b>
5.1	Вводная информация . . . . .	12
5.2	Предполагаемая схема использования . . . . .	13
5.3	Описание используемых методов и функций . . . . .	13
<b>6</b>	<b>Список литературы</b>	<b>15</b>
<b>7</b>	<b>Приложение</b>	<b>16</b>

# 1 Актуальность программы

В глобализированном мире современный человек каждый день сталкивается с большим объемом информации. Существование систем, которые осуществляют анализ информации, может значительно упростить человеку взаимодействие с окружающим миром. В связи с этим, было принято решение о создании прототипа одной из описанной выше систем. Ориентиром в постановке цели была идея о создании продукта, который будет отвечать следующим требованиям: доступность для массового пользователя, функциональность, простота в использовании.

Упомянутая выше идея нашла реализацию в виде следующей **цели**: создание телеграм-бота, который определяет наличие объекта на фотографии.

Платформа телеграм была выбран, поскольку в отличие от аналогов, в настоящий момент она работает на территории Российской Федерации. Телеграм так же имеет широкое распространение, в связи с чем продуктом смогут воспользоваться большое количество людей. Эта платформа так же предоставляет возможность организовать удобный интерфейс для взаимодействия системы с человеком, благодаря чат-ботам. Последние разрабатываются при помощи библиотеки `pyTelegramBotAPI`. Что касается внутреннего устройства системы, то было принято решение использовать готовый вариант обученной нейронной сети VGG16, поскольку данная сеть хорошо справляется с поставленной задачей, в виде высокой точности получаемых результатов. Две компоненты проекта были соединены на платформе Google Colaboratory. Данная платформа бесплатно предоставляет необходимые вычислительные мощности для обучения сети.

Для достижения поставленной цели необходимо выполнить следующие **задачи**:

1. Изучить теоретический блок по нейронным сетям
2. Адаптировать под условия проекта существующий вариант нейронной сети vgg16
3. Создать оболочку, с которой пользователь сможет легко взаимодействовать, на основе библиотеки языка Python `pyTelegramBotAPI`
4. Объединить воедино внешнюю и внутреннюю компоненты на платформе Google Colaboratory

В процессе поиска не было обнаружено подобных телеграм ботов, но удалось следующие аналоги:

1. Подробная инструкция по созданию телеграм-бота, [отправляющего фото кота](#)
2. Бот распознает печатный текст на картинке и присылает набранный текст.  
`@scannertext_bot`

## 2 Нейронные сети

На рисунке ниже показана схема нейронной сети: круги синего цвета представляют из себя входной слой, то есть получаемые данные, зеленые - скрытый слой, тот, в котором проходят всевозможные операции над данными, и оранжевый - выходной слой, в котором находится итоговая, нужная нам информация.

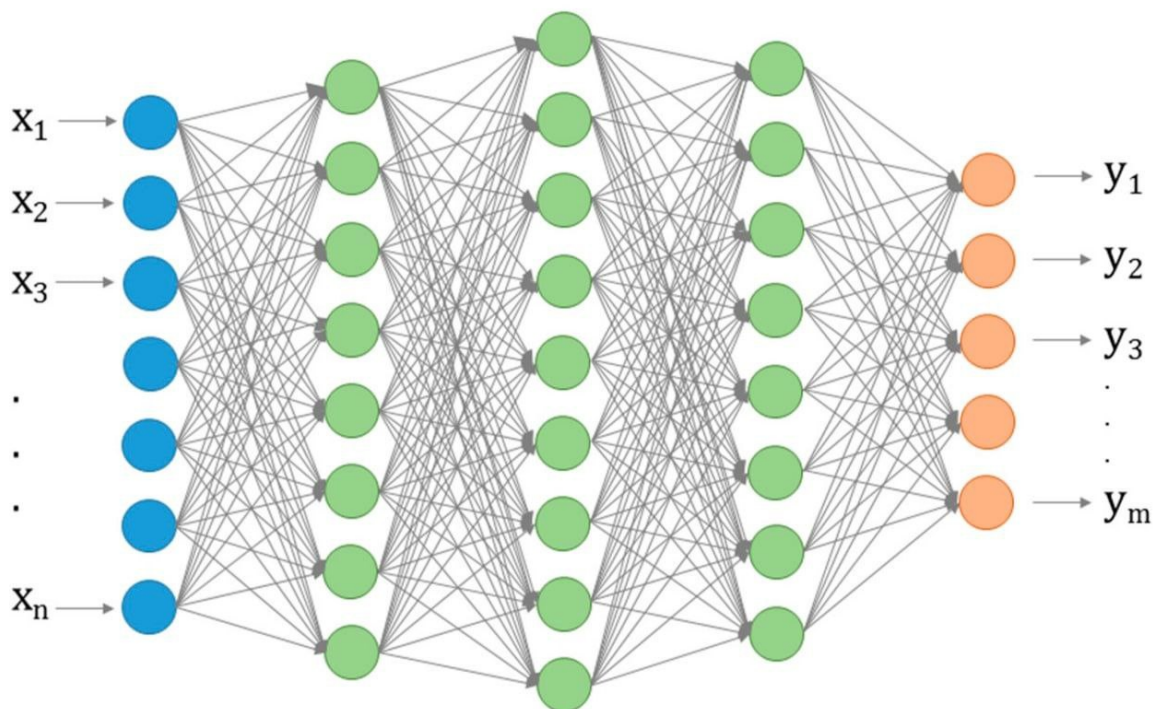


Рис. 1: Пример нейронной сети.

### 2.1 Что такое машинное обучение и как оно связано с нейронными сетями

Машинное обучение - это обширная область науки, где изучаются алгоритмы, которые самостоятельно находит компьютер, благодаря некоторым закономерностям в полученных данных.[1] Если говорить абстрактно, из-за этих алгоритмов у компьютера появляется свой "интеллект" , и он может решать определенные задачи или принимать решения вместо человека. "Машинное обучение - это наука о том, как без применения явного программирования, сделать компьютеры рабочими" , - запись Стэнфордского курса, еще один вариант определения МО.[2]

Машинное обучение является подгруппой искусственного интеллекта, а нейронная сеть, в свою очередь, состоит из одного слоя МО, глубокое обучение - из нескольких слоев.[2] Первое упоминание о нейросетях появилось в 40-х годах прошлого века, когда Питтс и Мак-Каллок предложили простейшую математическую

модель нейрона. Сразу появилось множество идей, вплоть до создания человекоподобного робота. Но, как можно заметить, ничего из предложенного не осуществилось, не хватало технических возможностей. И только в 90-х годах появилась возможность осуществить некоторые идеи, например, распознавание и прогнозирование, а начиная с 2014 года, образовалась отправная точка для третьей волны популярности и возрождения.[7]

Примером использования технологий машинного обучения может служить и сфера бизнеса, где нейронная сеть определяет, кому стоит предложить кредит в банке, строя свое предположение на достаточно большом массиве данных, которые не смог бы обработать человек. То есть, такая наука становится неотъемлемой частью жизни. При уже имеющихся возможностях не составит труда написать сеть, которая будет принимать за вас какие-то важные решения, взвешивая все за и против.

## 2.2 Классификация задач машинного обучения

Обычно машинное обучение делится на обучение с учителем и без, а они в свою очередь еще на несколько типов. Полную информацию можно увидеть на рисунке ниже, а пока рассмотрим некоторые разделы.[5]

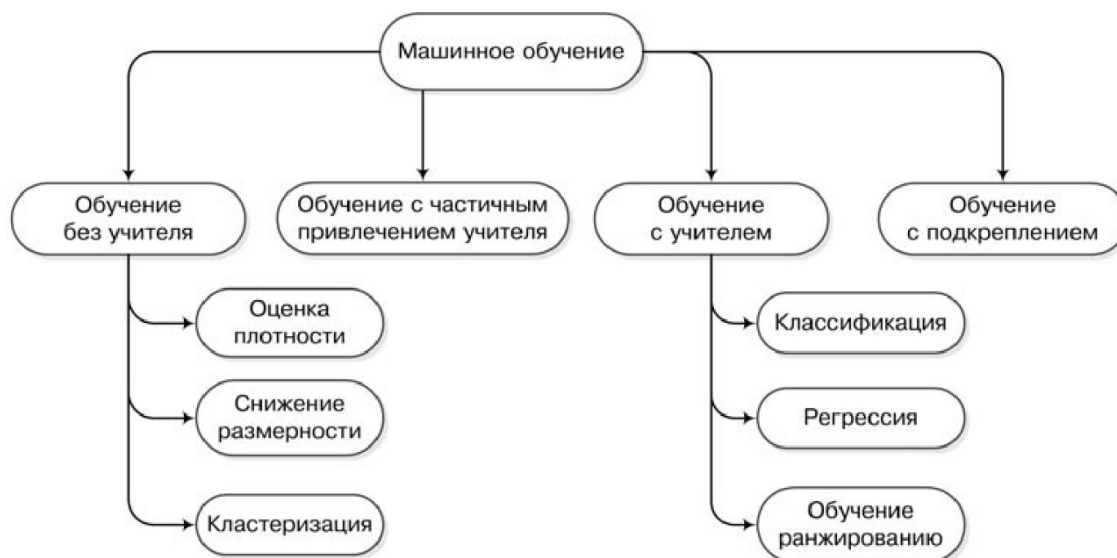


Рис. 2: Общая классификация постановок задач машинного обучения.[5]

### 1. Обучение с учителем

Применяя этот метод обучения, используют набор тренировочных данных, анализируя который нейросеть должна осознать/классифицировать следующие наборы данных. Обязательное требование - это должны быть похожие наборы данных, иначе смысл такого обучения теряется. Самым простым примером может служить распределение синтаксиса в предложении английского текста, которое применили для немецкого. В таком случае понятно, что не стоит ожидать адекватного поведения.

В свою очередь, этот вид обучения делится на *обучение ранжированию*, *классификацию* и *регрессию*. Последний представляет из себя предсказание значения некоторой функции, зависящей от иногда бесконечного числа переменных. Например, исходя из роста человека, определить его вес. Классификация, как можно понять из названия, разделяет объекты на определенные классы, например, определить, изображен ли один и тот же человек на фотографии. Именно классификация и используется в представленном нами проекте. Также метод ранжирования часто используется в поисковых системах, благодаря которому пользователь может достаточно быстро найти интересующий запрос.[5]

## 2. Обучение без учителя

Второй вид обучения применяется, когда нет какого-то определенного набора данных для задачи, нужно просто "найти смысл". Типичный пример - *кластеризация* - разбивает данные на заранее неизвестные классы и распределяет точки по принципу схожести классов. Также используется метод *снижения размерности*, благодаря которому мы "сужаем" большие данные до более мелких, не теряя при этом суть. И последний вид - это *оценка плотности*, в котором по представленным данным нужно, на удивление, оценить плотность размещения.[5]

3. Иногда встречаются ситуации, в которых нельзя точно определить, какой метод обучения стоит применить. В такие моменты используется обучение с частичным привлечением учителя.[5]

## 2.3 Компьютерное зрение и его необходимые составляющие

Компьютерное зрение является неотъемлемой частью машинного обучения, с помощью которого компьютер может работать с картинками. Благодаря знаниям этой области система может ответить практически на любой вопрос о картинке, например, сравнить две фотографии людей и определить, один ли это человек с большой точностью. Стоит отметить, что в нашем проекте также задействована эта прикладная область.

Говоря о том, как устроено компьютерное зрение, стоит отметить, что изображения представляются в виде матрицы. После того, как программе пришло изображение, она подсчитывает некоторые необходимые признаки и передает данные в классификатор(если у нас стоит задача определения типа объекта). Самым простым примером может служить дерево решений, с помощью которого строятся большие схемы, имеющие множество узлов и условий.[7]

Обычно изображение делится на части, в каждой из которой происходит свой анализ. Кроме того, используются также фильтры - матрицы коэффициентов, с помощью которых можно изменить изображение так, чтобы с ним было проще работать.[7] Для настройки фильтров необходимы 3 компоненты:

### 1. Большая выборка

Огромный набор картинок, порядка 10000 фотографий, с помощью которого обучается сеть. Каждой из картинок вручную был присвоен класс объ-

екта, который там изображен, так что благодаря колоссальной работе создателей ImageNet и других его аналогов, процесс становится в разы легче.[7]

## 2. Многослойность

Некоторое количество фильтров и нелинейных преобразований, которые применяются к картинке, называются слоем. В итоге получаем такую нелинейную фильтрацию, которая показывает характерные признаки изображения. В конце этого процесса у нас будет набор коэффициентов, который мы передаем в классификатор.[7]

## 3. Аппаратное обеспечение

Чтобы подобрать огромное количество коэффициентов, нужно специальное оборудование, которое позволит распараллелить подобные задачи, поэтому для этого используется процессор GPU.[7]

## 2.4 Сверточная нейронная сеть

Рассмотрим сверточную нейронную сеть - некий прототип зрительной коры мозга, который в последнее время приобрел популярность. Хьюбел и Визель еще в 1962 году предложили такой принцип работы нейронной сети, в котором она была бы чувствительна к конкретным областям поля зрения, то есть когда какие-то нейросети воспринимают вертикальные границы, другие - горизонтальные или диагональные.[3].

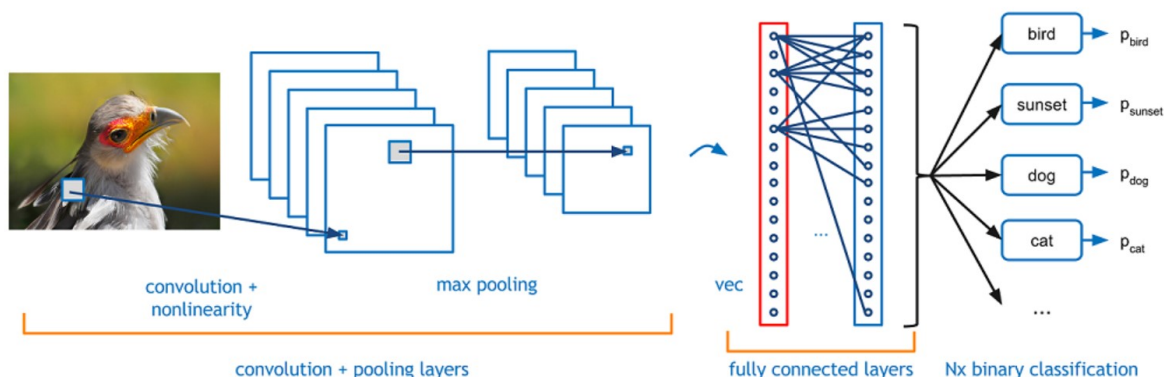


Рис. 3: Сверточная нейронная сеть.[3]

Используя эту сеть, не требуется задавать признаки, так как она сама учится извлекать полезные данные при достаточном обучении. Компьютеру должен различать изображения, поданные на вход, и распознавать уникальные особенности объектов, отличающие собаку от кошки.[3]

## 3 Предварительно обученная нейронная сеть VGG16

### 3.1 Общие сведения о НС VGG16

Модель сверточной нейронной сети VGG16 была предложена двумя учеными из Оксфордского университета. Упомянутый датасет состоит из 14 миллионов изображений. Точность модели достигает 92.7%, при тестировании в задаче распознавания объектов на изображении. Обучение сети VGG16 происходило на протяжении нескольких недель.

В VGG16 реализован фильтр размера 3x3, в отличие от предшествующей версии AlexNet, где присутствовали два фильтра с размерами 11 и 5 в первом и втором сверточных слоях, соответственно, с большим числом настраиваемых параметров.

### 3.2 Данные для обучения

VGG16 обучалась на том же наборе данных, на котором происходило тестирование. ImageNet содержит в себе более 15 миллионов размеченных высококачественных изображений, разделенных на 22000 размеченных вручную категорий.

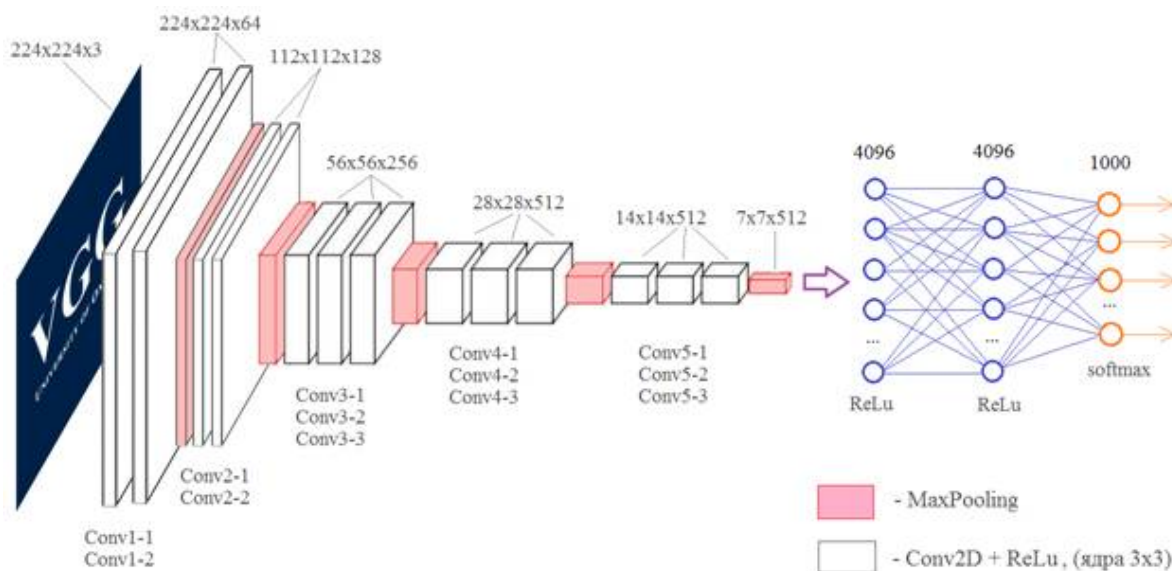


Рис. 4: Представление СНС VGG16.[9]

В слой conv1 подаются RGB изображения размера 224x224. Затем данные изображения проходят через стек сверточных слоев, которые используют фильтры с рецептивным полем размера 3x3. Данные поля являются наименьшими для возможности получения представления о нахождении картинки в пространстве.[9]

В одной из конфигураций используется сверточный фильтр размера 1x1, который может быть представлен как линейная трансформация входных каналов. Сверточный шаг фиксируется на значении один пиксель. Пространственное дополнение входа сверточного слоя выбирается таким образом, чтобы пространственное



разрешение сохранялось после свертки, то есть дополнение равно 1 для  $3 \times 3$  сверточных слоев. Пространственный пулинг осуществляется при помощи пяти max-pooling слоев, которые следуют за одним из сверточных слоев.[9]

После стека сверточных слоев (который имеет разную глубину в разных архитектурах) идут три полносвязных слоя: первые два имеют по 4096 каналов, третий — 1000 каналов. Последним идет soft-max слой. Конфигурация полносвязных слоев одна и та же во всех нейросетях[9].

Скрытые слои снабжены функцией активации ReLu. Сама сеть не содержит слоя нормализации, потому что это не дает улучшений результата на датасете, а ведет к увеличению времени исполнения и потребления памяти.

### 3.3 Конфигурация

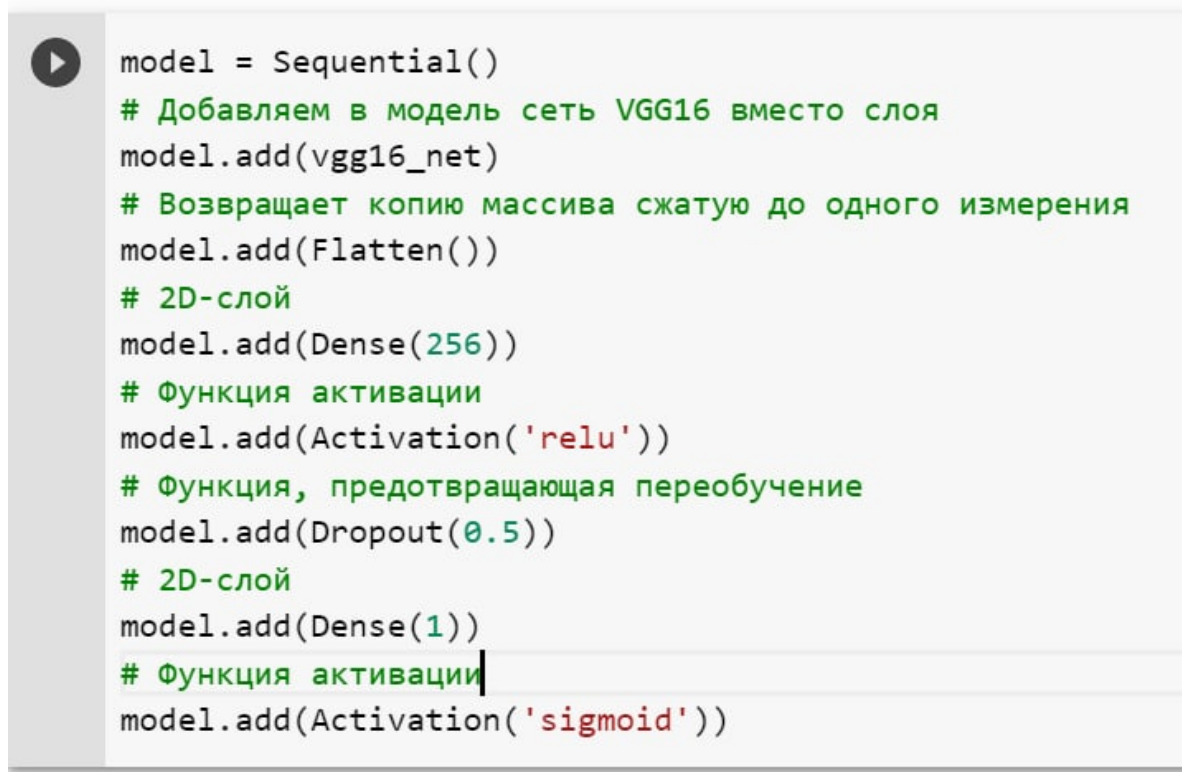
Конфигурации сверточных сетей представлены на изображении. Каждая сеть соответствует своему имени (А-Е). Все конфигурации имеют общую конструкцию и различаются только глубиной: в сети А от 11 слоев(8 сверточных и 3 полносвязных слоя) до 19 (16 сверточных и 3 полносвязных слоя). Ширина сверточных слоев небольшая: от 64 в первом слое до 512 в последнем, причем с увеличением количества каналов в 2 раза после каждого max-pooling слоя.[9].

VGG-16	VGG-19
input (224x244x3)	
conv3-64 conv3-64	conv3-64 conv3-64
maxpool	
conv3-128 conv3-128	conv3-128 conv3-128
maxpool	
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
maxpool	
conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512
maxpool	
FC-4096	
FC-4096	
FC-1000	
softmax	

Рис. 5: Классификация СНС VGG16.[10]

## 4 Функции активации

В СНС VGG16 использовались две функции активации. Функция активации «ReLU» была использована для основных слоев, а функция активации «sigmoid» была использована для предотвращения переобучения.

A screenshot of a code editor from Google Colaboratory. It shows a Python script for building a Keras model. The code starts with 'model = Sequential()' and then adds several layers and activation functions. Comments in Russian explain each step: adding the VGG16 network, flattening it, adding a 2D layer with 256 units and ReLU activation, adding a Dropout layer with 0.5 probability, adding another 2D layer with 1 unit, and finally adding a sigmoid activation function. The code is as follows:

```
model = Sequential()
# Добавляем в модель сеть VGG16 вместо слоя
model.add(vgg16_net)
# Возвращает копию массива сжатую до одного измерения
model.add(Flatten())
# 2D-слой
model.add(Dense(256))
# Функция активации
model.add(Activation('relu'))
# Функция, предотвращающая переобучение
model.add(Dropout(0.5))
# 2D-слой
model.add(Dense(1))
# Функция активации
model.add(Activation('sigmoid'))
```

Рис. 6: Часть кода с Google Colaboratory

### 4.1 ReLu

Данная функция активации выглядит так

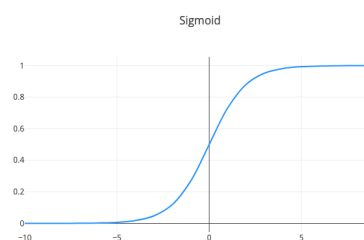
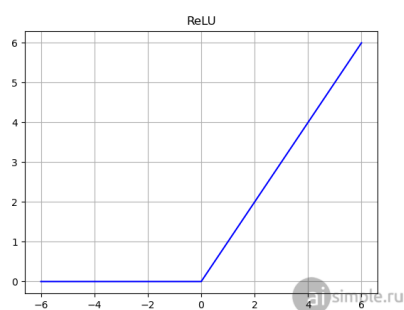
$$A(x) = \max(0, x)$$

Нетрудно понять, что ReLu возвращает значение  $x$ , если оно положительно, и 0 в противном случае. На первый взгляд ReLu кажется линейной функцией, но это ошибочное утверждение. Данная функция является хорошим аппроксиматором, так как любая функция может быть аппроксимирована комбинацией ReLu. Область допустимых значений ReLu -  $[0, \infty)$ .

ReLu позволяет не активировать все нейроны, в отличие от сигмоиды или гиперболического тангенса. Это дает нам сделать активацию разреженной и более эффективной. Представим сеть со случайно инициализированными весами, в которой примерно 50% активаций равны 0 из-за характеристик ReLu (возвращает 0 для отрицательных значений  $x$ ). В такой сети включается меньшее количество нейронов (разреженная активация), а сама сеть становится легче.

Для отрицательных значений  $x$  часть ReLu представляет из себя горизонтальную линию, где градиент будет равен 0. Из-за этого веса не будут корректироваться во время спуска. Это значит, что нейроны, которые находятся в таком состоянии не будут реагировать на изменения в ошибке полученных данных. Из-за этого некоторые нейроны просто выключатся и не будут отвечать, делая достаточно большую часть нейросети пассивной. Но можно найти и другие версии ReLu, которые помогают эту проблему избежать. Например, заменить горизонтальную часть функции на линейную. Если выражение для линейной функции задается выражением  $y = 0.01x$  для области  $x < 0$ , линия слегка отклоняется от горизонтального положения.

ReLu чаще всего используется для создания глубоких нейронных сетей, так как производит более простые вычисления, что дает ей возможность быть менее требовательной к вычислительным ресурсам, чем гиперболический тангенс или сигмоида.



$$A(x) = \max(0, x)$$

$$A = \frac{1}{1 + e^{-x}}$$

Рис. 7: Графики функции активации.

Сигмоида, в свою очередь, выглядит гладкой и подобна ступенчатой функции, но в нашем случае она не подходит. В небольшом диапазоне значений аргумента, от -3 до 3 например, значения функции меняются очень быстро. При этом, когда при приближении к концам сигмоиды значения функции начинают слабо реагировать на изменения в аргументе. Из-за этого градиент в таких областях принимает маленькие значения, что приводит к проблемам с градиентом исчезновения, следовательно, нейросеть отказывается обучаться дальше или делает это крайне медленно.

## 5 Телеграм-бот

### 5.1 Вводная информация

Перед непосредственной разработкой, с помощью специального бота *@BotFather* был получен уникальный идентификатор бота, который называется токеном. Через *@BotFather* было создано меню, представляющее список возможных команд. Как было сказано ранее, PyTelegramBotAPI [8] — библиотека, которая использовалась в разработке.

## 5.2 Предполагаемая схема использования

Пользователь инициирует начало работы бота с помощью команды `\start`. После использования этой команды бот готов принимать фотографии. Бот уведомит пользователя о готовности получить очередное фото сообщением "Жду очередное фото!". Полученное изображение обрабатывается нейронной сетью. По завершении обработки бот автоматически отправляет сообщение с результатом работы нейронной сети. Возможны два ответа в зависимости от присутствия предполагаемого объекта на фото (в данном случае предполагаемым объектом является кошка): "Это не кошка!" – если на изображении нет кошки и "Это кошка!" в обратном случае. Для завершения работы необходимо воспользоваться командой `\end`, которая блокирует возможность получения результата.

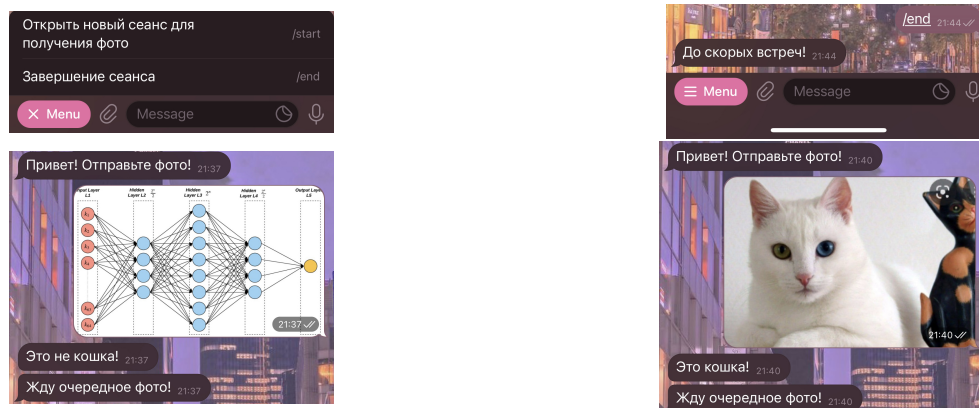


Рис. 8: Скриншоты работы бота.

После того, как бот был запущен, он принимает фотографию пользователя из чата и сохраняет ее. Далее изображение передается нейронной сети, в которой выполняются все необходимые действия. Булевский ответ нейронки передается телеграм боту и отображается в чате как положительный или отрицательный ответ.

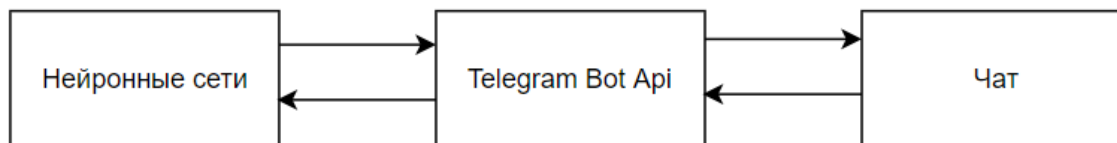


Рис. 9: Схема работы.

## 5.3 Описание используемых методов и функций

При написании кода был использован декоратор `@message_handler`, реагирующий на входящие сообщения. `Message` является объектом из Bot API, который хранит в себе данные о сообщении. Полезное поле, которое использовалось для отправки сообщения `message.chat.id` – идентификатор чата.

С помощью функции `send_message`, которая принимает на вход идентификатор чата и текст для отправки, были созданы шаблонные ответы на команды `\start`, `\end` и результат работы нейронной сети.

При получении файла в скобках декоратора `@message_handler` указываем тип, который бот будет ожидать от пользователя. В нашем случае прописываем `photo`. В результате получаем уникальный идентификатор файла `file_id`. С помощью метода `getfile` подготовим файл для скачивания (в течение часа будет возможность загрузить файл). И затем загружаем файл, используя `bot.download_file`.

Для объединения работы нейронной сети и бота, была создана функция `func`, которая принимает на вход строку, в которой прописан путь до нужной фотографии. Функция возвращает 1, если на изображении нет кота, и 0 если кот на изображении есть.

Чтобы бот имел возможность итеративно реагировать на действия пользователя, используем функцию `bot.infinity_polling()`, которая запускает бесконечный цикл получения новых записей со стороны Telegram.

Ссылку на код с проектом можно найти в приложении.

## 6 Список литературы

1. Введение в машинное обучение : официальный сайт. – 2006–2022, Habr. –URL: <https://habr.com/ru/post/448892/?> (дата обращения 25.05.2022).
2. Что такое машинное обучение : официальный сайт. – 2020, Moscow. –URL: <https://dtf.ru/hard/191018-cto-takoe-mashinnoe-obuchenie?ysclid=l3lz94kq0> (дата обращения 25.05.2022).
3. Adit Deshpande - A Beginner's Guide To Understanding Convolutional Neural Networks/Engineering at Forward | UCLA CS '19 : официальный сайт. – 2006–2022, github. –URL: <https://adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks/> (дата обращения 25.05.2022).
4. Github - Предварительно обученной нейронной сети VGG16 : официальный сайт. – 2018, GTS+3. –URL: [https://github.com/sozykin/dlpython\\_course/blob/master/computer\\_vision/cats\\_and\\_dogs/cats\\_and\\_dogs\\_vgg16.ipynb](https://github.com/sozykin/dlpython_course/blob/master/computer_vision/cats_and_dogs/cats_and_dogs_vgg16.ipynb) (дата обращения 25.05.2022).
5. Николенко, С. Глубокое обучение /С. Николенко, А. Кадури, Е. Архангельская ; СПб.: Питер, 2018. – 480с.: ил. –(Серия «Библиотека программиста»). – ISBN 978-5-496-02536-2.
6. ШАД - Учебник по машинному обучению/Филипп Синицин /Станислав Федотов – 2018, Москва. –URL: <https://ml-handbook.ru/> (дата обращения 25.05.2022).
7. Хабр - Глубокие нейросети в компьютерном зрении. : официальный сайт. – 2020, Moscow. –URL: <https://habr.com/ru/company/leader-id/blog/529012/> (дата обращения 25.05.2022).
8. Официальная документация библиотеки PyTelegramBotAPI : официальный сайт. – 2020, Moscow. –URL: <https://core.telegram.org/bots/api> (дата обращения 25.05.2022).
9. VGG16 — сверточная сеть для выделения признаков изображений : официальный сайт. – Москва, 2018. –URL: <https://neurohive.io/ru/vidy-nejrosetej/vgg16-model/> (дата обращения 25.05.2022).
10. Примеры архитектур сверточных сетей VGG-16 и VGG-19 : официальный сайт. – Москва, 2018. –URL: [https://proporprogs.ru/neural\\_network/primery-arhitektur-svertochnyh-setey-vgg16-i-vgg19](https://proporprogs.ru/neural_network/primery-arhitektur-svertochnyh-setey-vgg16-i-vgg19) (дата обращения 25.05.2022).
11. Функции активации в искусственных нейронных сетях. – Москва, 2020. –URL: <https://aisimple.ru/8-funkcii-aktivacii-v-iskusstvennyh-nejronnyh-setjah.html> (дата обращения 25.05.2022).
12. Функции активации нейросети: сигмоида, линейная, ступенчатая, ReLu, tahn. – Москва, 2018. –URL: <https://neurohive.io/ru/osnovy-data-science/activation-functions/> (дата обращения 25.05.2022).
13. Создайте простую нейронную сеть с TensorFlow.js. - 2017. –URL: <https://www.machinelearningmastery.ru/build-a-simple-neural-network-with-tensorflow-js-d434a30fcb8/> (дата обращения 25.05.2022).

## 7 Приложение

[Ссылка](#) на Jupiter Notebook с кодом проекта