**Министерство образования и науки РФ**

**Федеральное государственное автономное образовательное учреждение**

**Высшего образования**

**«КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**ВЫСШАЯ ШКОЛА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И ИНФОРМАЦИОННЫХ СИСТЕМ**

Специальность: 09.03.04. – Программная инженерия

КУРСОВАЯ РАБОТА

**Разработка системы бинарной классификации изображений, с использованием предварительно обученной нейронной сети VGG16**

Студент 2 курса

Группа 11-604

«\_\_»\_\_\_\_\_\_\_\_\_\_201\_\_ г. \_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Р.А.Климов)

**Научный руководитель**

«\_\_»\_\_\_\_\_\_\_\_\_\_20\_\_г. \_\_\_\_\_\_\_\_\_\_\_\_\_\_(Л.Л.Закиров)

Оглавление

[**Введение:** 3](#_Toc514848794)

[**1.Описание технологий используемых в системе и выбор набора данных.** 5](#_Toc514848795)

[**1.1 Выбор набора данных** 5](#_Toc514848796)

[**1.2 Обучение с учителем** 6](#_Toc514848797)

[**1.3 Нейронная сеть** 6](#_Toc514848798)

[**1.4 Обучение нейронной сети** 7](#_Toc514848799)

[**1.5 Сверточные нейронные сети** 7](#_Toc514848800)

[**1.5 Перенос обучения** 9](#_Toc514848801)

[**2.Конструирование, обучение и оценка качества сети** 12](#_Toc514848802)

[**2.1Подключение Vgg16 с использованием keras** 12](#_Toc514848803)

[**2.2 Создание составной сети** 14](#_Toc514848804)

[**2.3 Компиляция составной сети:** 15](#_Toc514848805)

[**2.4 Подготовка набора изображений** 16](#_Toc514848806)

[**2.5 Обучение составной сети** 17](#_Toc514848807)

[**2.6 Проверка качества работы составной сети** 17](#_Toc514848808)

[**2.7 Код** 18](#_Toc514848809)

[**Заключение** 19](#_Toc514848810)

[**Литература:** 20](#_Toc514848811)

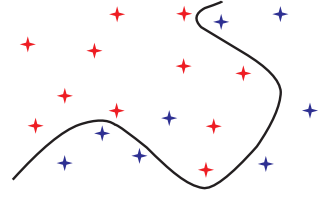
# 

# **Введение**

Классификация изображений является важным инструментом в современном мире, где большие данные используются для принятия всех видов решений в правительстве, экономике, медицине и других областях. У исследователей есть доступ к большому количеству данных, а классификация - один из инструментов, который позволяет осмыслить эти данные и найти шаблоны.

Задача классификации изображений представляет собой задачу определения принадлежности объекта на нем к одному из нескольких различных классов. В качестве примеров таких задач могут служить:

* Определение возраста: по фотографиям людей отсеивать “взрослые” лица.
* Определение наличия каски у рабочего: нахождение и уведомление людей определенных профессий о необходимости защитной одежды по фотографии.
* Обнаружение лица: найти и выделить лицо человека на фотографии



*Рис. 1. Пример результата работы классификационного алгоритма*

С ростом количества больших данных и производительности компьютеров, наиболее часто применяемым способом решения задачи классификации стали глубокие нейронные сети, так как они содержат большое количество параметров и для их обучения нужно много времени. Они позволяют добиться высокой точности работы классификатора.

Целью настоящей работы является написание программы, способной поделить изображения на два класса с высоким уровнем точности.

Для достижения цели были поставлены следующие задачи:

1. Изучить способы и инструменты для решения похожих задач.
2. Найти подходящий набор данных.
3. Сконструировать нейронную сеть.
4. Обучить нейронную сеть.
5. Оценить качество работы сети.

# 

# 

# 

# 

# 

# 

# **1.Описание технологий используемых в системе и выбор набора данных.**

**1.1 Выбор набора данных**

Данные для этой задачи возьмем с соревнования kaggle “Hot Dog - Not Hot Dog”. Оно содержит тысячу размеченных фотографий с хот догами и не хот догами. Названия классов указаны в именах файлов: все фотографии с хот-догами начинаются на hot\_dog, а все фотографии с не хот-догами - на not\_hot\_dog.

Несколько фотографий из набора:

|  |  |
| --- | --- |
| ярриат  *Рис. 2. Хот-дог* | *Рис. 3. Не хот-дог* |
| *Рис. 4. Хот-дог* | *Рис. 5. Не хот-дог* |

Так, как типов объекта всего два (хот-дог и не хот-дог), то данная классификация является бинарной. То есть на выходе должен быть 1 нейрон. 0 означает, что на выходе объект, не являющийся хот догом, 1 означает что объект является хот догом.[1]

**1.2 Обучение с учителем**

На вход алгоритма мы подадим размеченный набор данных. После просмотра большого количества примеров, алгоритм постепенно начнет понимать, на что смотреть на новых фотографиях, чтобы правильно разделить их на группы. Такой метод машинного обучения называется обучением с учителем.

Как же алгоритм учится понимать принадлежность признаков к определенным объектам? Вначале он просто предполагает. Он ошибается и делает небольшую поправку параметров. Этот процесс повторяется много раз на каждом изображении.

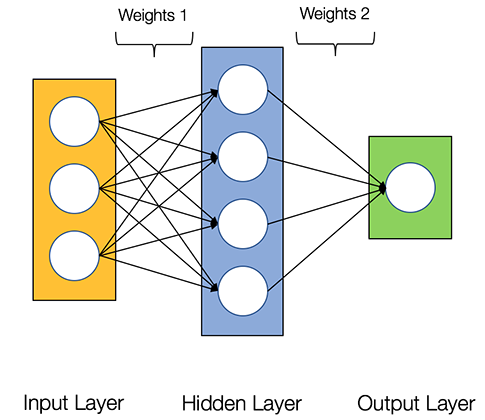
Есть много методов машинного обучения, способных классифицировать изображения. Для данной задачи было принято использовать сверточные нейронные сети, так как они позволяют получать высокие результаты во многих областях компьютерного зрения.[2] Прежде чем использовать данный метод, нужно понимать, что такое обычная нейронная сеть.

**1.3 Нейронная сеть**

Нейронную сеть можно описать, как математическую функцию, которая соотносит входные данные с желаемыми выходными.[3]

Нейронная сеть состоит из следующих компонентов:

1. входной уровень(input layer)
2. произвольное количество скрытых уровней(hidden layers)
3. выходной уровень(output layer)
4. набор весов(weights) и смещений(biases) между каждым из уровней
5. функция активации для каждого из скрытых уровней



*Рис. 6. Архитектура двухслойной нейронной сети*

**1.4 Обучение нейронной сети**

После того как мы пропустим тренировочные данные через все скрытые слои, в конце сеть даст свое предсказание класса для объекта. Далее вычисляется функция потери(the loss function), показывающая как далеко или как близко предсказание к действительному ответу. Цель в данном случае - минимизировать функцию потери, чтобы максимально приблизить предсказание сети к действительному ответу.

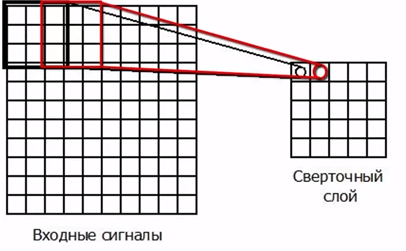
Чтобы исправить ответ на выходном уровне нужно вернуться назад и исправить вывод нейронов на предыдущем уровне и так далее. Данный процесс называют методом обратного распространения ошибки(backpropagation)

**1.5 Сверточные нейронные сети**

Мощный инструмент для работы с компьютерным зрением. Очень похожи на обычные нейронные сети, но позволяют эффективнее работать с изображениями в силу своей архитектуры.

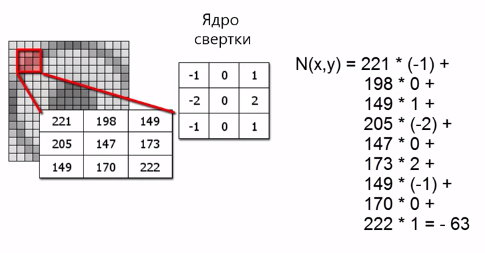
Любое изображение для компьютера - это просто матрица значений пикселей. Сверточная нейронная сеть состоит из чередующихся слоев свертки и подвыборки, соединенных между собой.

* **Сверточные слои:**



*Рис. 7. Пример работы сверточного слоя*

В отличии от обычных нейронных сетей, в сверточных сетях нейрон получает на вход ограниченное количество пикселей. Операция, которая выполняется нейроном в данном типе сети называется сверткой. Во время операции свертки рассматривается некоторая область изображения и матрица такого же размера, называемая ядром свертки.[4]

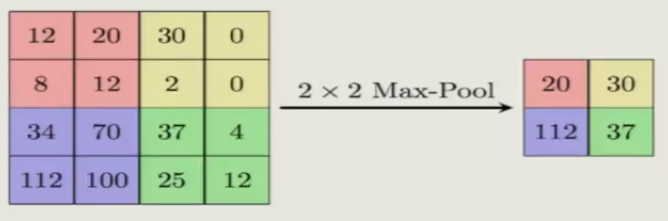


*Рис. 8. Пример работы ядра свертки на сверточном слое*

* **Слои подвыборки**

Необходимо уметь распознавать объекты, независимо от того, в каком они масштабе. На практике факт наличия объекта на изображении обычно важнее знания о его расположении.

Для уменьшения размерности в сверточных нейронных сетях используются слои подвыборки. Один нейрон слоя подвыборки подключен к нескольким нейронам предыдущего слоя. Значения нейронов следующего слоя определяется на основе значений предыдущего слоя. Раньше использовалось усреднение, но теперь используется выбор максимального значения.



*Рис. 9. Пример работы уменьшения размерности на слое подвыборки*

**1.5 Перенос обучения**

Вместо того, чтобы создавать модель нейронной сети с нуля и самому настраивать все необходимые параметры, можно воспользоваться предварительно обученной моделью, немного дополнив которую, возможно получить хороший результат.

Перенос обучения(transfer learning) - использование предварительно обученной нейронной сети для решения задачи, отличающейся от той, для которой она обучалась.

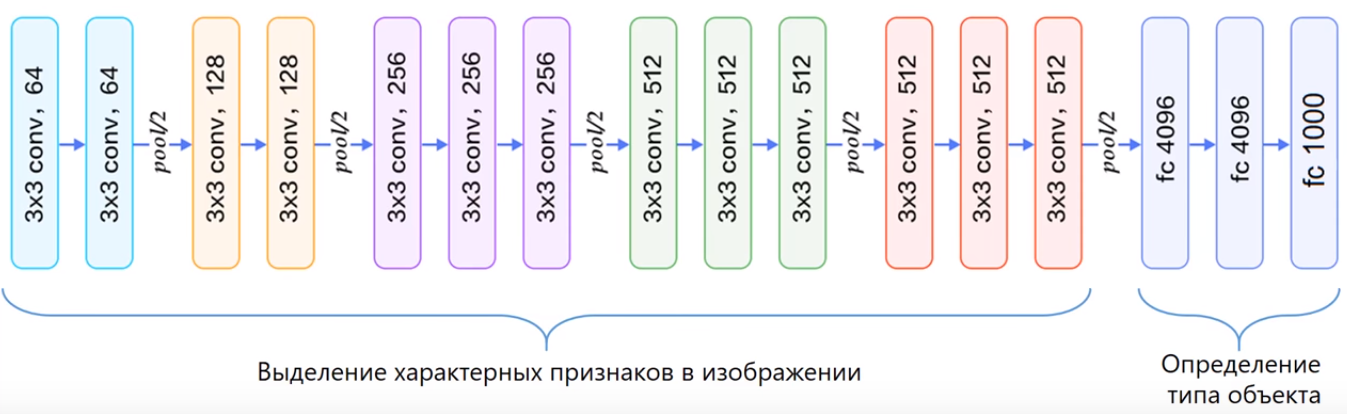
Keras - библиотека, которая включает в себя обученные сети, такие как VGG16, VGG19, Inseption v3 и другие. Большая часть этих сетей обучена на наборе данных ImageNet. Это набор данных, который содержит фотографии более 1000 типов объектов. Требуется обучить сеть распознавать объекты не из набора данных ImageNet, а другие.

Почему возможен перенос обучения? Большая часть предварительно обученных нейронных сетей, которые используются для решения задачи компьютерного зрения, состоят из двух частей. Будет рассмотрена сеть VGG16, которая достаточно простая, но не смотря на это показывает хорошие результаты.

VGG16, как и другие похожие сети, состоит из двух частей. Первая часть сверточная, она используется, чтобы выделить из изображения характерные признаки. На основании этих признаков, уже можно определить, что за объект на изображении.

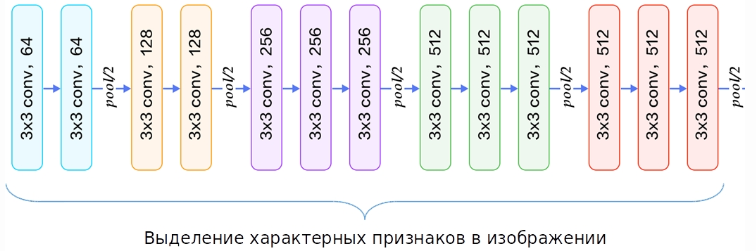
Для этого используется вторая часть - полносвязная. Именно она решает задачу классификации. По характерным признакам, которые выделила сверточная сеть, определяет, какой именно объект находится на изображении.

В наборе данных ImageNet 1000 классов объектов, поэтому в сети VGG16 и других сетях, обученных на данном наборе, на выходном слое 1000 выходных нейронов. Используется формат one-hot encoding: значение только одного выходного нейрона должно быть близко к единице, остальные близки к нулю.[5]



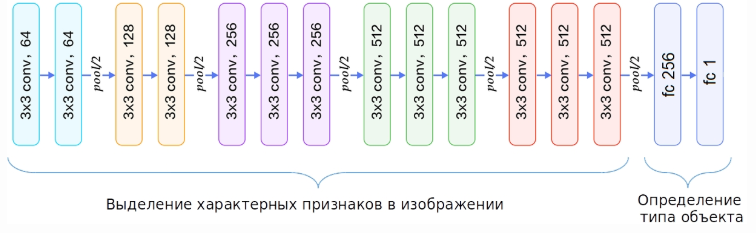
*Рис. 10. Схема сверточной нейронной сети VGG16*

Применим данную сеть для другой задачи, а именно для распознавания хот-догов. При переносе обучения от сети VGG16, возьмем только ту часть, которая нам полезна, а именно сверточную часть, которая выделяет характерные признаки на изображении. Часть, которая отвечает за классификацию, брать не будем, так как эта часть определяет тысячи объектов разных типов, а нам необходимо определять всего лишь 2 объекта.



*Рис. 11. Схема сверточной части VGG16*

Поэтому к сверточной части VGG16 добавим часть, которая отвечает за классификацию всего двух типов объектов. Эта часть тоже состоит из полносвязных слоев, но на выходе всего лишь один нейрон. Перед выходным слоем находится еще один полносвязный слой, в котором 256 нейронов. Этот слой подключается к выходу сверточной части VGG16.



*Рис. 12. Схема архитектуры составной сверточной нейронной сети*

Затем составная нейронная сеть обучается на наборе данных “Hot Dog - Not Hot Dog”.

# **2.Конструирование, обучение и оценка качества сети**

**2.1Подключение Vgg16 с использованием keras**

Первым делом, необходимо загрузить предварительно обученную нейронную сеть VGG16.

vgg16\_net = VGG16(weights='imagenet',

include\_top = False,

input\_shape=(150, 150, 3))

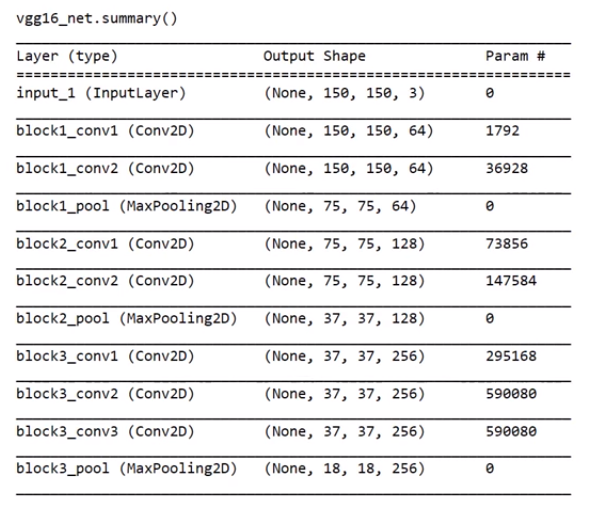
vgg16\_net.trainable = False

Параметр “include\_top = False” означает то, что не будет загружена та часть сети, которая отвечает за классификацию. То есть будет загружена только сверточная часть VGG16. Также важно указать параметр “trainable = False”, означающий то, что сверточную часть обучать не нужно. Сеть VGG16 обучена выделять характерные признаки на наборе данных ImageNet, в этот набор входят фотографии хот догов, то есть сеть уже умеет выделять нужные признаки.[1]

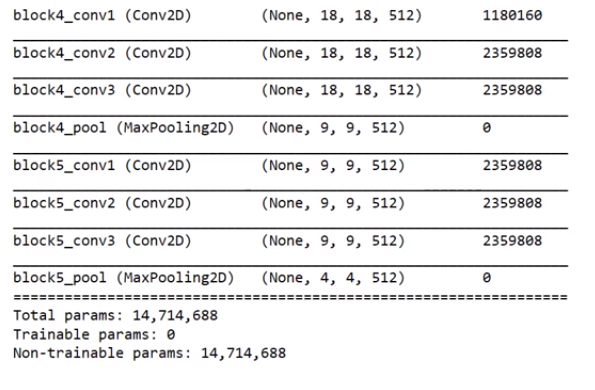
Таким образом, в процессе обучения, сверточная часть сети оказывается заморожена, обучается только новая часть сети, которая используется для классификации двух типов объектов - хот-догов и не хот догов.

Перед обучением, веса в этой части инициализируются случайным образом, а затем обучение выполняется стандартным образом, с помощью метода обратного распространения ошибки.

Если вывести информацию о сети, то можно увидеть перечень слоев и количество весов нейронов в них.



*Рис. 13. Вывод информации о сети(1)*



*Рис. 14. Вывод информации о сети(2)*

И в конце можно увидеть, что в сети 14 миллионов параметров, из которых, количество обучаемых параметров равно 0, а количество необучаемых параметров, все 14 миллионов.

**2.2 Создание составной сети**

На следующем этапе создается составная сеть, которая включает в себя VGG16 и новую часть для классификации. Для создания такой сети используется последовательная(Sequential) модель Keras.

model = Sequential()

model.add(vgg16\_net)

model.add(Flatten())

model.add(Dense(256))

model.add(Activation('relu'))

model.add(Dropout(0.5))

model.add(Dense(1))

model.add(Activation('sigmoid'))

Особенность создания такой сети заключается в том, что на первом этапе мы добавляем в сеть не отдельный слой, а целую предварительно обученную модель.

После этого, добавляется слой Flatten - выпрямление. Этот слой нужен из-за того, что на выходе из VGG16, есть большое количество двумерных карт признаков. На входе в полносвязный слой необходимо получить плоский вектор. Таким образом, двумерные карты признаков преобразуются в одномерный вид.[1]

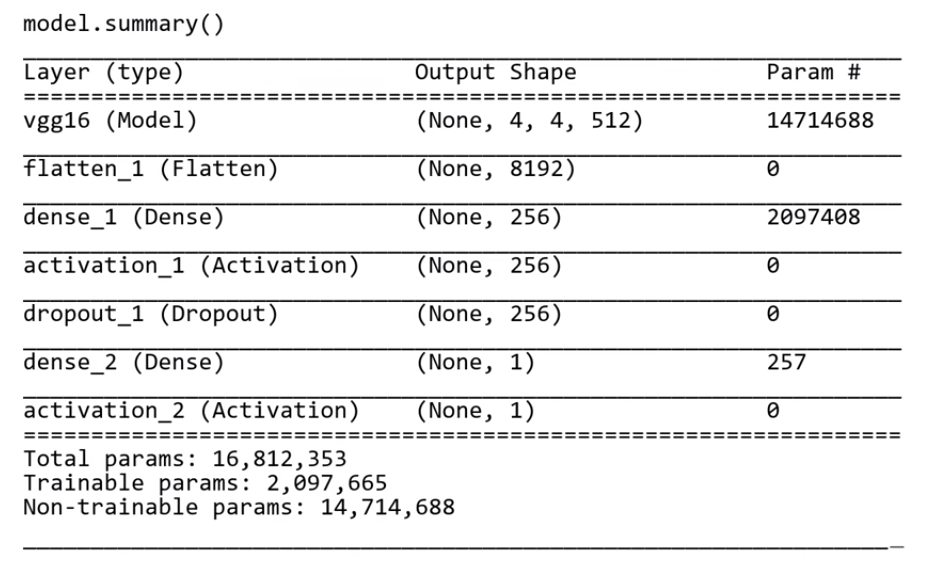
Затем идет полносвязный слой(Dense), в котором 256 нейронов, с полулинейной функцией активации (если ее аргумент меньше нуля, то она равна нулю, а в остальных случаях, ведет себя как линейная).[6]

Далее слой Dropout, для регуляризации снижения переобучения.

Суть метода заключается в том, что в процессе обучения выбирается слой, из которого случайным образом выбрасывается определенное количество нейронов (например 30%), которые выключаются из дальнейших вычислений. Такой прием улучшает эффективность обучения и качество результата. Более обученные нейроны получают в сети больший вес.

И выходной полносвязный слов, в котором один нейрон и сигмоидальная функция активации. Эта функция плавно меняет свое значение от нуля до единицы, а значит хорошо подходит для решения задач классификации.

Информация о структуре сети на данном этапе:



*Рис. 15. Вывод информации о сети на данном этапе*

Можно заметить, что первый слой сети это отдельная модель VGG16, а количество обучаемых параметров, это веса тех нейронов, которые входят в полносвязную часть сети.

**2.3 Компиляция составной сети:**

model.compile(loss='binary\_crossentropy',

optimizer=Adam(1r=1e-5),

metrics=['accuracy'])

Так как в качестве части составной сети используется сеть VGG16, которая уже обучена достаточно хорошо, необходимо использовать низкую скорость обучения, для того, чтобы алгоритм обратного распространения точно установился в некоторой точке.

При создании оптимизатора, задается параметр lr (сокращение от learning rate, скорость обучения) равным единица на десять в минус пятой степени (для оптимизатора Adam по умолчанию lr=0.001, на два порядка больше). Если использовать более высокую скорость обучения, то алгоритм обучения не сойдется.[1]

**2.4 Подготовка набора изображений**

Данные для генераторов изображений Keras нужно подготовить особым образом. Для этого создается каталог, в котором будут подкаталоги по количеству классов объектов. В данном случае таких классов два: хот-дог и не хот-дог.[7]

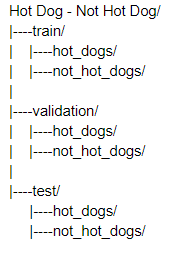
Таким образом, создается каталог с именем hot\_dogs, в который переносятся фотографии хот-догов, и not\_hot\_dogs, в котором будут все прочие фотографии. Имена каталогов могут быть любыми. При обучении нейронной сети, генератор изображений Keras будет читать картинки из обоих каталогов и автоматически готовить метки с правильными ответами, например, 0 для не хот-дога и 1 для хот-дога.

Для обучения нейронной сети нам нужно три набора изображений:

* Набор данных для обучения.
* Набор данных для проверки (оценки качества обучения сети в процессе обучения).
* Набор данных для тестирования (оценки качества обучения сети после завершения обучения).

Каждый набор данных должен содержать подкаталоги с фотографиями изображений разных классов.

Для задачи распознавания хот-догов и не хот-догов, структура каталогов для трех наборов данных будет выглядеть следующим образом:



*Рис. 16. Используемая структура каталогов*

Данные из датасета были распределены следующим образом:

70% изображений в каталог для обучения и по 15% в каталоги для проверки или тестирования.

**2.5 Обучение составной сети**

model.fit\_generator(train\_generator,

steps\_per\_epoch=nb\_train\_samples //

batch\_size,

epochs=5,

validation\_data=val\_generator,

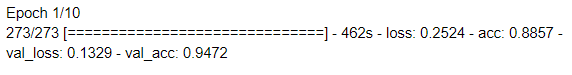
validation\_steps=nb\_validation\_samples //

batch\_size)

Обучение ведется с помощью генераторов. Генераторы позволяют автоматически загружать изображения с диска в память компьютера, преобразовывать их в вид, необходимый Keras, и передавать в модель для обучения. [7]

Train generator предоставляет данные для обучения, а validation generator, данные для проверки.

Так как часть сети предварительно обучена, 5-10 эпох должно быть достаточно. Обучение ведется на GPU. Одна эпоха занимает примерно 500 секунд. На центральном процессоре время обучения может занять день или даже больше.



*Рис. 17. Диагностический вывод обучения сети*

**2.6 Проверка качества работы составной сети**

После того, как обучение закончено, необходимо проверить качество обучения. Для этого также используются генераторы.

Test generator предоставляет фотографии хот-догов и не хот-догов, которые не использовались ни для обучения, ни для проверки.

Точность работы на тестовой выборке оказалась равной 96.17%.

**2.7 Код**

Код разработанной системы находится в открытом доступе на сайте *GitHub* по следующей ссылке: <https://github.com/RomanKlimov/NotHotDogClassifier>

# **Заключение**

В результате работы над проектом была разработана рабочая версия программы классификации изображений набора данных “Hot Dog - Not Hot Dog”, с использованием переноса обучения, что позволило добиться 96% точности предсказаний классов изображений. Точность работы созданной составной сети можно еще улучшить с помощью технологии тонкой настройки сети(fine tuning)

# **Литература:**

[1]Перенос обучения в Keras. ASozykin [Электронный ресурс]. Режим доступа: <https://www.asozykin.ru/deep_learning/2018/01/08/Transfer-Learning-in-Keras.html>

<https://www.slideshare.net/yandex/ss-42667639>

[2]Что такое сверточная нейронная сеть. Habr. [Электронный ресурс]. Режим доступа: <https://habr.com/post/309508/>

[3]How to build your own Neural Network from scratch in Python. TowardsDataScience [Электронный ресурс]. Режим доступа: <https://towardsdatascience.com/how-to-build-your-own-neural-network-from-scratch-in-python-68998a08e4f6>

[4]Лекция «Сверточные нейронные сети». ASozykin. [Электронный ресурс]. Режим доступа: <https://www.asozykin.ru/courses/nnpython>

[5]What is One Hot Encoding?. Hackernoon. [Электронный ресурс]. Режим доступа: <https://hackernoon.com/what-is-one-hot-encoding-why-and-when-do-you-have-to-use-it-e3c6186d008f>

[6] Терехов В.А., Ефимов Д.В., Тюкин И.Ю. Нейросетевые системы управления. [Текст] — 1-е. — Высшая школа, 2002. — С. 184.

[7] Лекция “Как подготовить собственный набор изображений для обучения нейронной сети в Keras”. ASozykin. [Электронный ресурс]. Режим доступа: <https://www.asozykin.ru/courses/nnpython>