МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное

образовательное учреждение высшего образования

«Самарский национальный исследовательский университет

имени академика С. П. Королева»

(Самарский университет)

Институт информатики и кибернетики

Факультет информатики

Кафедра программных систем

ОТЧЁТ

по лабораторной работе № 1 на тему

«Реализовать нейронную сеть для классификации изображений аналогично примеру»

по курсу «Нейронные сети глубокого обучения»

Выполнил:

Александров А.А.

Студент группы 6132-020402D

Проверил:

Доцент, к.т.н. Жданова А. Н.

Самара 2022

**Лабораторная работа № 1**

Цель работы: Реализовать нейронную сеть для классификации изображений аналогично примеру. Пример: <https://www.tensorflow.org/tutorials/keras/classification>

**Задание:**

3. Коточелендж: <https://www.kaggle.com/tongpython/cat-and-dog>

Ссылка на github: <https://github.com/Allexeey/Cat_and_dog>

**Структура отчета:**

1. Математическая постановка задачи

2. Описание архитектуры сети, параметров, метода обучения и т.д.

3. Описание вычислительных экспериментов с целью оценки эффективности модели:

показать, как обучали, функцию потерь, точность своей модели и сравнение разных

моделей с разным количеством слоев, нейронов и типов функций активаций

4. Результаты работы

5. Выводы

**1. Математическая постановка задачи**

Требуется рассмотреть 2 класса объектов D(dog) и C(cat):

Необходимо каждый объект отнести к одному из двух классов D или C, т.е. должна быть задана функция:

.

**2. Описание архитектуры сети, параметров, метода обучения и т.д.**

Для того чтобы данные загрузились в модель. Они должны быть предварительно обработаны. Устанавливаем размер изображения imgh, imgw = 64, 64, а также делим значения на 255.

Разделим данные на обучение и проверку.

*# Train*

train\_ds = ImageDataGenerator(rescale = 1./255, shear\_range = 0.2, zoom\_range = 0.2, horizontal\_flip = True)

training\_set = train\_ds.flow\_from\_directory(train\_path, target\_size=(imgh, imgw), batch\_size=32, class\_mode='binary')

*# Test (Validation)*

test\_ds = ImageDataGenerator(rescale = 1./255)

test\_set = test\_ds.flow\_from\_directory(test\_path, target\_size=(imgh, imgw), batch\_size=32, class\_mode='binary')

Функция активации (Activation Function) — функция, аргументом которой является линейная комбинация входов в нейрон/перцептрон. В качестве функции активации используется — нелинейная функция и (ReLU, и Sigmoid).

фильтры: целое число, размерность выходного пространства (т. е. Количество выходных фильтров в свертке).

kernel\_size: целое число или кортеж / список из 2 целых чисел, задающий высоту и ширину окна 2D свертки. Может быть одним целым числом, чтобы указать одно и то же значение для всех пространственных измерений.

*# Model Creation*

model = Sequential()

*#STEP 1: ADDING CONVOLUTIONAL LAYER*

model.add(Conv2D(filters=32, kernel\_size=3, activation='relu', input\_shape=[64, 64, 3]))

model.add(MaxPool2D(pool\_size=2, strides=2))

*#Adding 2nd Convolutional layer*

model.add(Conv2D(filters = 64,kernel\_size = 3, activation = 'relu'))

model.add(MaxPool2D(pool\_size = 2, strides = 2))

*#STEP 3: FLATTENING*

model.add(Flatten())

*#STEP 4: FULL CONNECTION*

model.add(Dense(units=128, activation='relu'))

*#STEP 5: OUTPUT LAYER*

model.add(Dense(units=1, activation='sigmoid'))

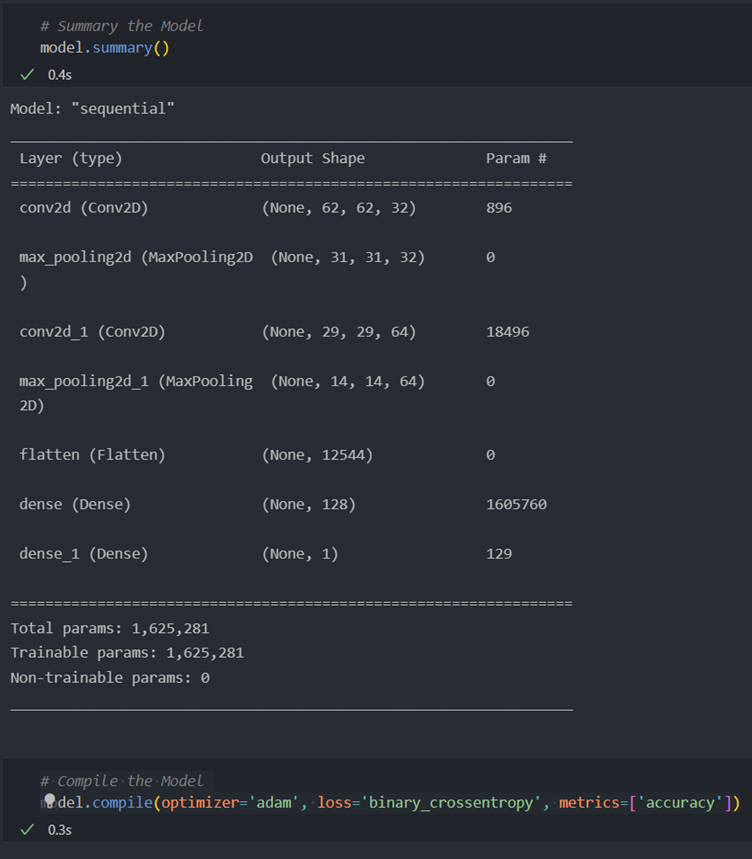
На первом шаге добавим операцию свертки (Convolution) — операция, в ходе которой фильтр (окно) свертки применяется ко входному тензору специальным образом: веса фильтра на каждом шаге поэлементно умножаются на подтензор этого тензора и складываются. А также фильтруем его. Фильтр (окно) свертки (Convolution Filter) — тензор, имеющий фиксированный размер (обычно, где — число меньше 10). Числа внутри этого тензора называются весами фильтра (окна) свертки.

Операция подвыборки (Пулинга, Pooling) — операция, в ходе которой берется лишь часть элементов тензора, поданного на вход. Причем эти элементы берутся по некоторому конкретному правилу (например, максимум при Max Pooling'е, или среднее при Average Pooling'е). Чаще всего пулинг — это извлечение максимума/среднего элементов, попадающих под окно пулинга (область фиксированноо размера, например, окно 2х2).

На втором шаге добавим вторую операцию свертки (Convolution) с 64 фильтрами, а также операцию подвыборки (Пулинга, Pooling).

Flatten, преобразует формат изображений из двумерного массива в одномерный массив. Этот слой не имеет параметров для изучения; он только переформатирует данные.

Первый Dense слой имеет 128 узлов (или нейронов). Второй (и последний) слой возвращает массив логитов длиной 1. Каждый узел содержит оценку, указывающую, что текущее изображение принадлежит к одному из классов.



**Скомпилируем модель**

Прежде чем модель будет готова к обучению, ей нужно еще несколько настроек. Они добавляются на этапе *компиляции* модели:

* *Функция потерь* — измеряет, насколько точна модель во время обучения. Вы хотите минимизировать эту функцию, чтобы "направить" модель в правильном направлении.
* *Оптимизатор* — именно так модель обновляется на основе данных, которые она видит, и ее функции потерь.
* *Метрики* — используются для мониторинга этапов обучения и тестирования. В следующем примере используется *точность*, доля правильно классифицированных изображений.

*# Compile the Model*

model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])

**3. Описание вычислительных экспериментов с целью оценки эффективности модели:**

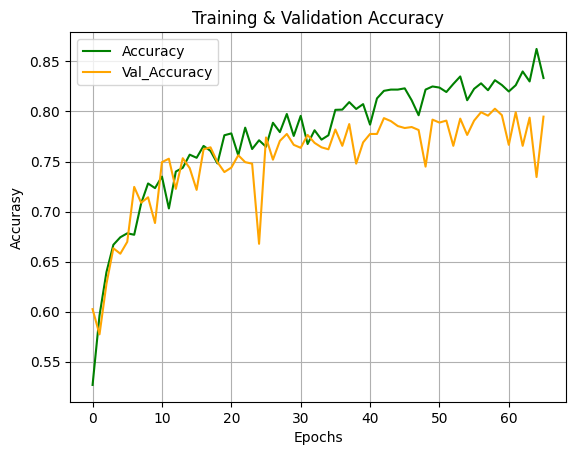
**показать, как обучали, функцию потерь, точность своей модели и сравнение разных**

**моделей с разным количеством слоев, нейронов и типов функций активаций**

Обучаем модель. Анализируем точность и потери. Строим графики.

*#Training the CNN on the Training set and evaluating it on the Test set*

trainHistory = model.fit(x = training\_set, validation\_data = test\_set, epochs = 66,steps\_per\_epoch=50)

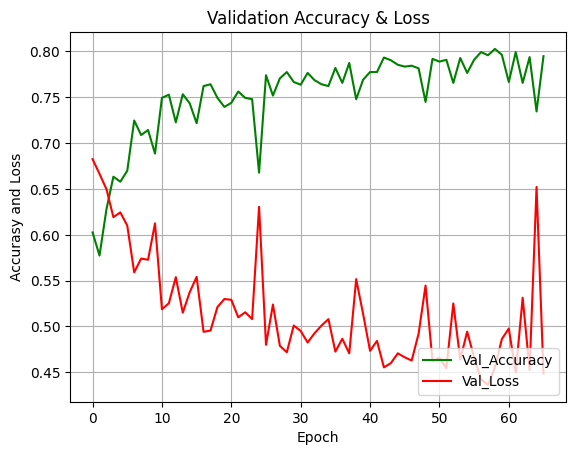


Training loss: 0.34886765480041504 Training accuracy: 0.8434728384017944



Test loss: 0.4486798644065857 Test accuracy: 0.7948591113090515





С целью оценки повышения эффективности модели обучим модель 2 раз. Добавим дополнительный Dense слой с 16 нейронами, а также добавим дополнительный слой свертки.

*# Model Creation*

model = Sequential()

*#STEP 1: ADDING CONVOLUTIONAL LAYER*

model.add(Conv2D(filters=32, kernel\_size=3, activation='relu', input\_shape=[64, 64, 3]))

model.add(MaxPool2D(pool\_size=2, strides=2))

*#Adding 2nd Convolutional layer*

model.add(Conv2D(filters = 64, kernel\_size = 3, activation = 'relu'))

model.add(MaxPool2D(pool\_size = 2, strides = 2))

*#Adding 3nd Convolutional layer*

model.add(Conv2D(filters = 64, kernel\_size = 3, activation = 'relu'))

model.add(MaxPool2D(pool\_size = 2, strides = 2))

*#STEP 4: FLATTENING*

model.add(Flatten())

*#STEP 5: FULL CONNECTION*

model.add(Dense(units=128, activation='relu'))

*#STEP 6: FULL CONNECTION*

model.add(Dense(units=16, activation='relu'))

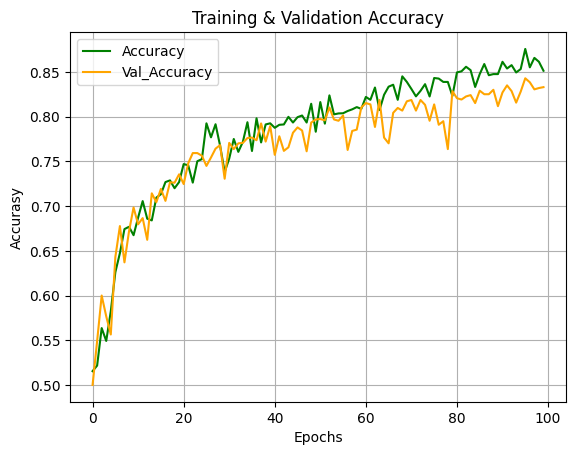
*#STEP 7: OUTPUT LAYER*

model.add(Dense(units=1, activation='sigmoid'))

Используя параметры.

trainHistory = model.fit(x = training\_set, validation\_data = test\_set, epochs = 100,steps\_per\_epoch=50)

Обучаем модель. Анализируем точность и потери. Строим графики.

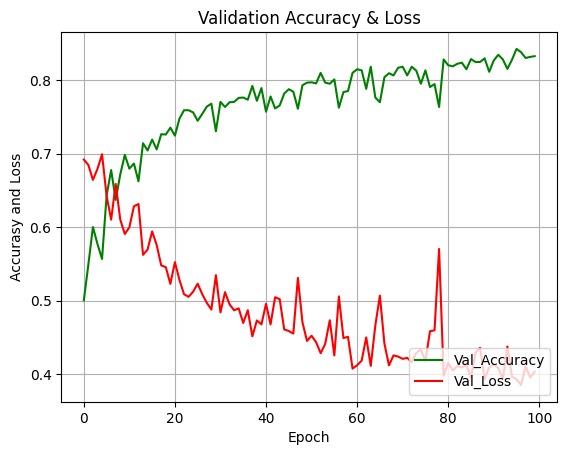


Training loss: 0.29535365104675293 Training accuracy: 0.8754528164863586



Test loss: 0.4031831622123718 Test accuracy: 0.8329213857650757





Модель 2 показала себя более эффективной используем её для оценки результатов.

**4. Результаты работы**

Тестируем сеть при помощи анализа изображений из тестовой выборки.

Test 1 cat.4322.jpg



c: 0.0

Test2 dog.4059.jpg



d: 100.0

Test3 cat.4320



c: 0.0

Test4 dog.4004



d: 100.0

Тестируем сеть при помощи анализа изображения моей кошки.

Test5 mycat.jpg



d: 100.0

Тестируем сеть при помощи анализа изображения из интернета.

Test6 cattestimg.jpg



c: 0.0

**5. Выводы**

Таким образом была реализована нейронная сеть для классификации изображений. Описана архитектура сети, а также проведены вычислительные эксперименты с целью оценки эффективности модели. Построены графики потерь и точности для обучающего сета и тестового сета.

Точность модели 1 на обучающем наборе составила 84%, а потери составили 35%. Точность на тестовом наборе составила 79%, а потери составили 45%.

Точность модели 2 на обучающем наборе составила 88%, а потери составили 30%. Точность на тестовом наборе составила 83%, а потери составили 40%.

Анализ изображений показал, что в целом сеть дает правильные прогнозы, но с изображением моей кошки сеть дала неверный прогноз. Возможно это связанно с изображение рыб и ярко синего цвета на фоне, а также размытостью изображения. Также сеть могла дать неверный прогноз в связи с породой кошки и цветом окраса.