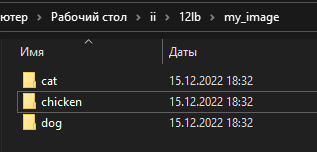
Загрузка своего набора изображений в TensorFlow

Для начала необходимо подготовить папки с изображениями в требуемом формате.

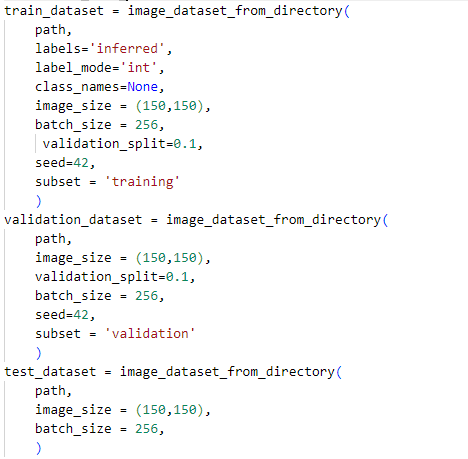
Где изображения, относящиеся к одному классу записаны в отдельный каталог.



Загружаем данные

****

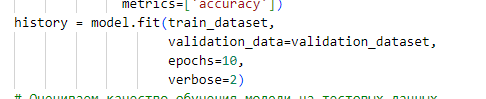
Делим набор данных



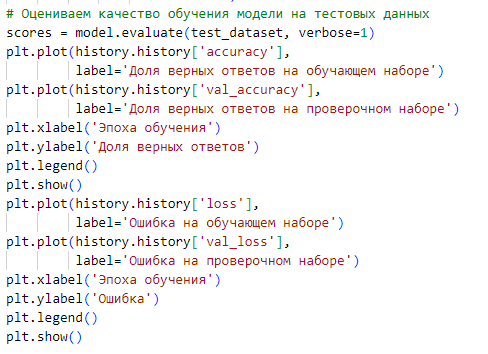
Создаем нейросеть и компилируем ее

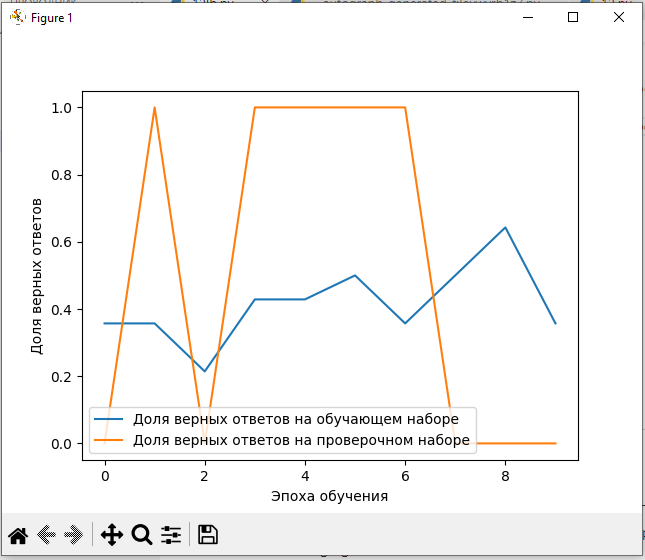
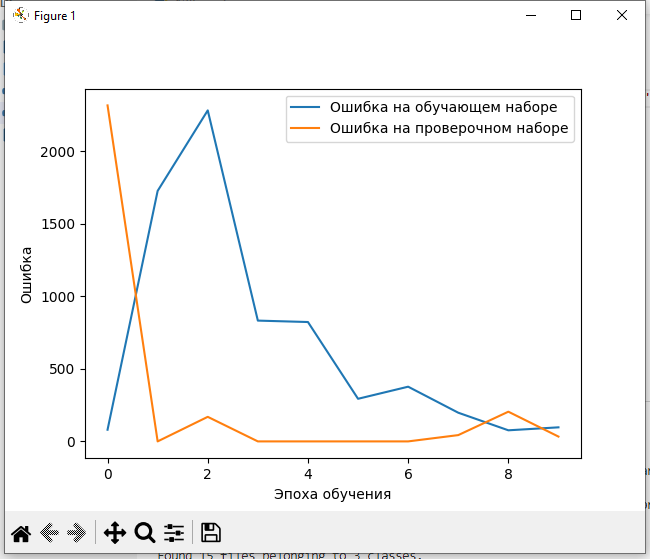


Обучаем

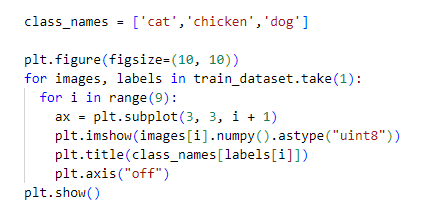


Оцениваем качество



**** ****

Выводим изображения с классами





**Вывод:**

В этой лабораторной работе я научился:

* как собрать свой датасет и передать его в модель для обучения,
* как объяснить какое изображение к какому классу относится,
* как разделить данные на обучающие и проверочные

В Tensorflow есть модуль tensorflow.keras.utils.image\_dataset\_from\_directory(), именно с его помощью можно создавать датасет изображений для обучения нейронной сети.

**Код**

import os

import numpy as np

import tensorflow as tf

import matplotlib.pyplot as plt

from tensorflow.keras.models import Sequential

from tensorflow.keras.preprocessing import image\_dataset\_from\_directory

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, Flatten, Dropout

path = 'C:/Users/fox/Desktop/ii/12lb/my\_image'

os.listdir(path)

train\_dataset = image\_dataset\_from\_directory(

    path,

    labels='inferred',

    label\_mode='int',

    class\_names=None,

    image\_size = (150,150),

    batch\_size = 256,

     validation\_split=0.1,

    seed=42,

    subset = 'training'

    )

validation\_dataset = image\_dataset\_from\_directory(

    path,

    image\_size = (150,150),

    validation\_split=0.1,

    batch\_size = 256,

    seed=42,

    subset = 'validation'

    )

test\_dataset = image\_dataset\_from\_directory(

    path,

    image\_size = (150,150),

    batch\_size = 256,

    )

# Настраиваем производительность TensorFlow DataSet'ов

AUTOTUNE = tf.data.experimental.AUTOTUNE

train\_dataset = train\_dataset.prefetch(buffer\_size=AUTOTUNE)

validation\_dataset = validation\_dataset.prefetch(buffer\_size=AUTOTUNE)

test\_dataset = test\_dataset.prefetch(buffer\_size=AUTOTUNE)

# Создаем последовательную модель

model = tf.keras.models.Sequential([

    # Сверточный слой

    tf.keras.layers.Conv2D(32,(3,3), activation='relu', input\_shape = (150,150,3),padding='same'),

    # Слой подвыборки

    tf.keras.layers.MaxPooling2D(2,2),

    # Сверточный слой

    tf.keras.layers.Conv2D(64,(3,3), activation='relu',padding='same'),

    # Слой подвыборки

    # tf.keras.layers.MaxPooling2D(2,2),

    # # Сверточный слой

    # tf.keras.layers.Conv2D(128,(3,3), activation='relu',padding='same'),

    # Слой подвыборки

    tf.keras.layers.MaxPooling2D(2,2),

    # Полносвязная часть нейронной сети для классификации

    tf.keras.layers.Flatten(),

    tf.keras.layers.Dense(128, activation='relu'),

    tf.keras.layers.Dropout(0.2),

    tf.keras.layers.Dense(64, activation='relu'),

    tf.keras.layers.Dropout(0.2),

    # Выходной слой, 3 нейрона по количеству классов

    tf.keras.layers.Dense(3, activation='softmax')

    ])

model.compile(loss='sparse\_categorical\_crossentropy',

              optimizer="adam",

              metrics=['accuracy'])

history = model.fit(train\_dataset,

                    validation\_data=validation\_dataset,

                    epochs=10,

                    verbose=2)

# Оцениваем качество обучения модели на тестовых данных

scores = model.evaluate(test\_dataset, verbose=1)

plt.plot(history.history['accuracy'],

         label='Доля верных ответов на обучающем наборе')

plt.plot(history.history['val\_accuracy'],

         label='Доля верных ответов на проверочном наборе')

plt.xlabel('Эпоха обучения')

plt.ylabel('Доля верных ответов')

plt.legend()

plt.show()

plt.plot(history.history['loss'],

         label='Ошибка на обучающем наборе')

plt.plot(history.history['val\_loss'],

         label='Ошибка на проверочном наборе')

plt.xlabel('Эпоха обучения')

plt.ylabel('Ошибка')

plt.legend()

plt.show()

class\_names = ['cat','chicken','dog']

plt.figure(figsize=(10, 10))

for images, labels in train\_dataset.take(1):

  for i in range(9):

    ax = plt.subplot(3, 3, i + 1)

    plt.imshow(images[i].numpy().astype("uint8"))

    plt.title(class\_names[labels[i]])

    plt.axis("off")

plt.show()