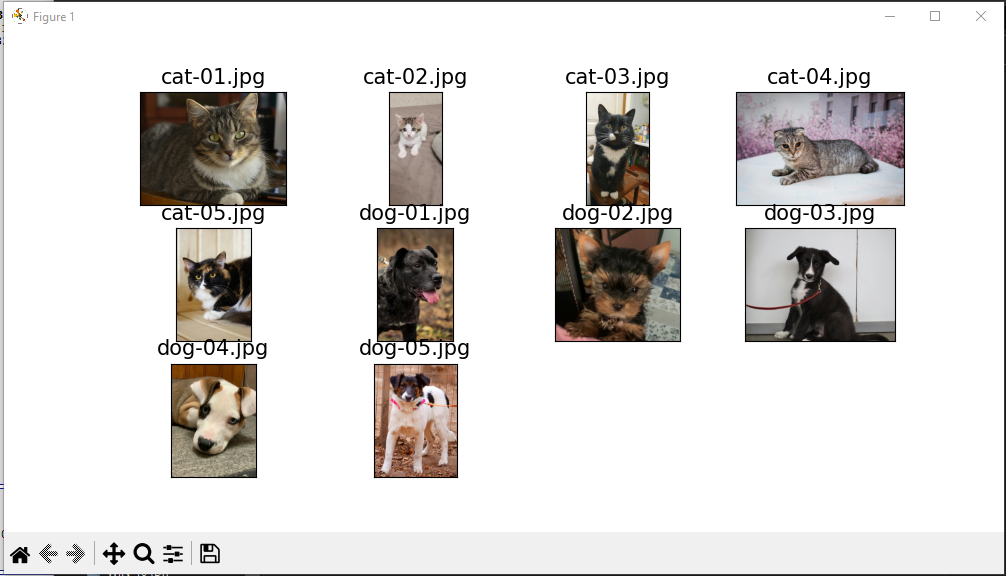
**1 часть**

Есть каталог с изображениями собак и кошек, которых необходимо разделить по классам 0- кошки, 1-собаки. Выводим изображения из каталога.



Извлекаем список имен файлов:



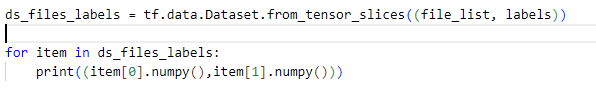


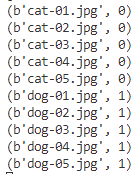
Извлекаем метки для изображений из списка имен, назначая соответствующие метки.



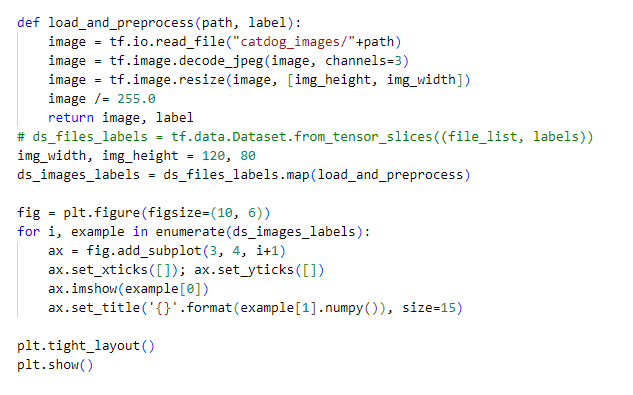


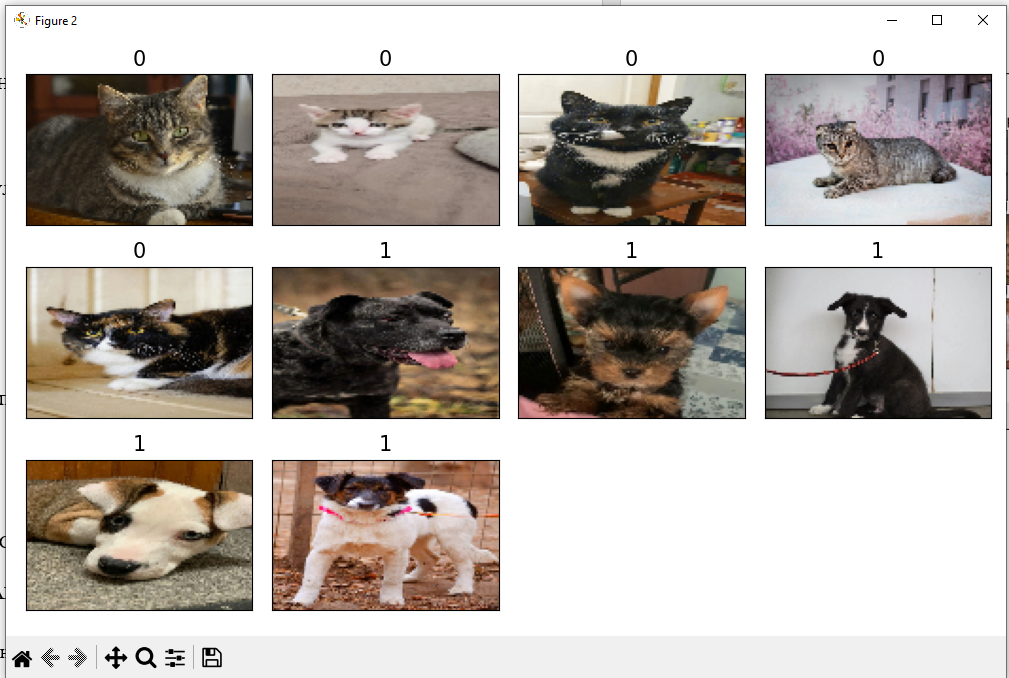
Теперь объединяем их





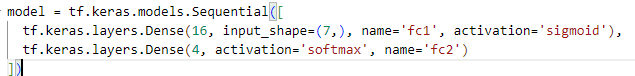
Загружаем содержимое изображения из его файла, декодируем низкоуровневое содержимое и изменяем размеры, результатом будет визуализация с метками класса.

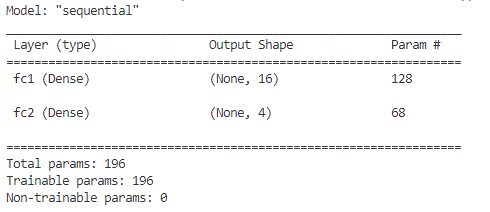


****

**2 часть**

Первый слой получает вход из 7 признаков и проецирует их на 35 нейронов.

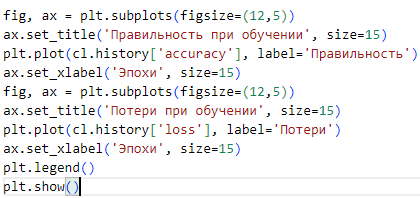
****

****

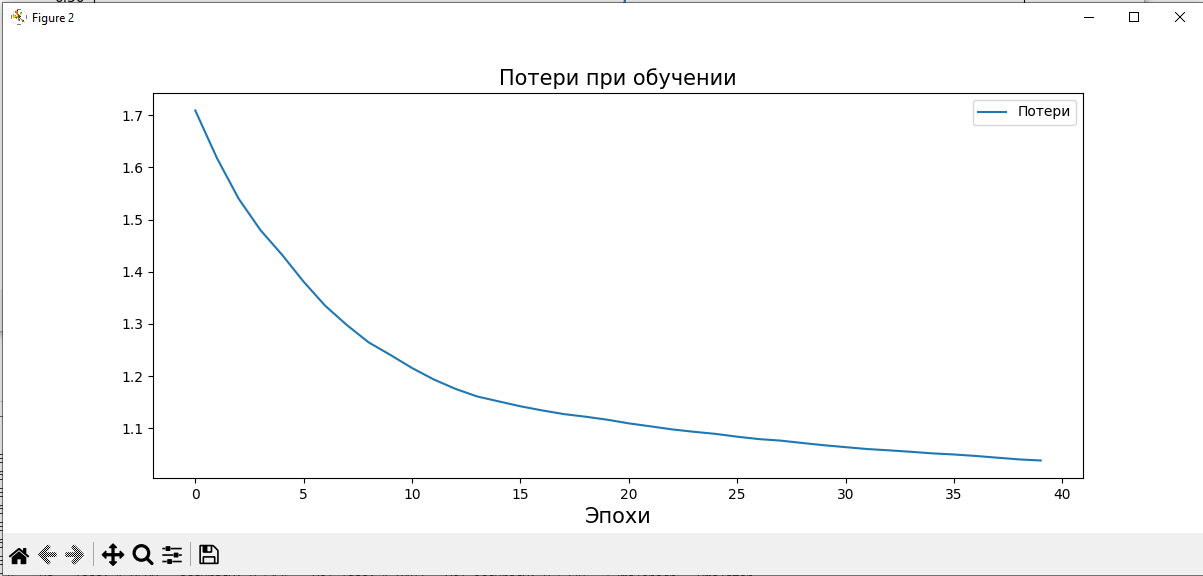
Выведенная сводка показывает, что первый слой имеет 128 параметров, второй 68

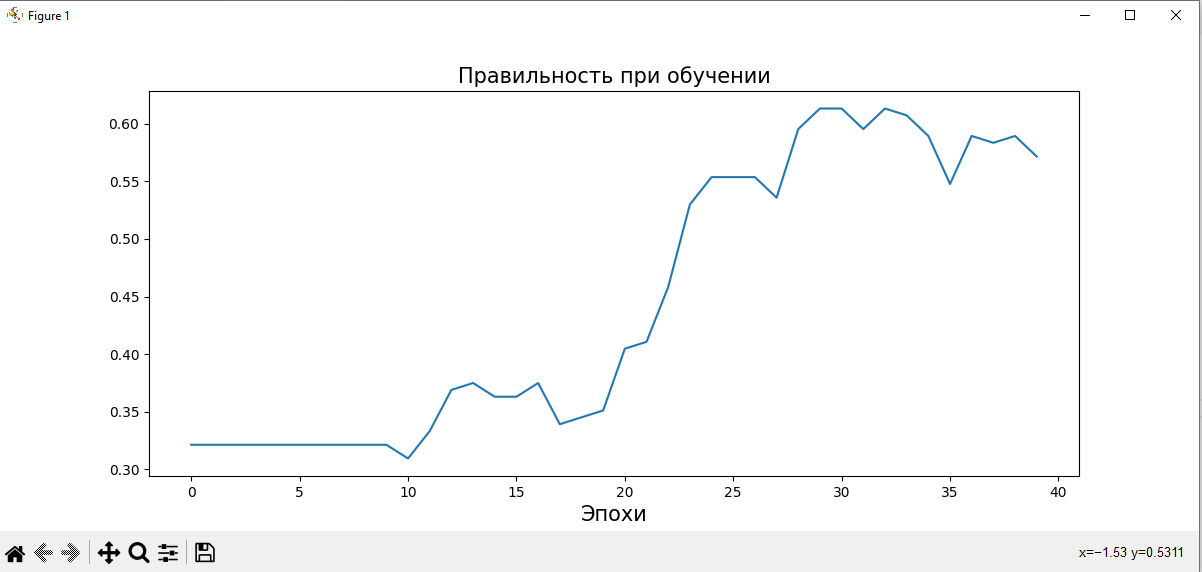
Возвращенная переменная history хранит потерю при обучении и правильность при обучении после каждой эпохи. Мы можем задействовать ее

для визуализации кривых обучения:

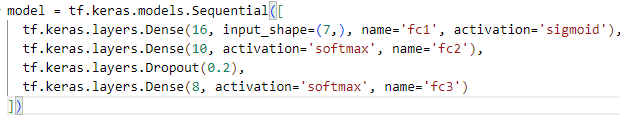


Получим следующие графики:

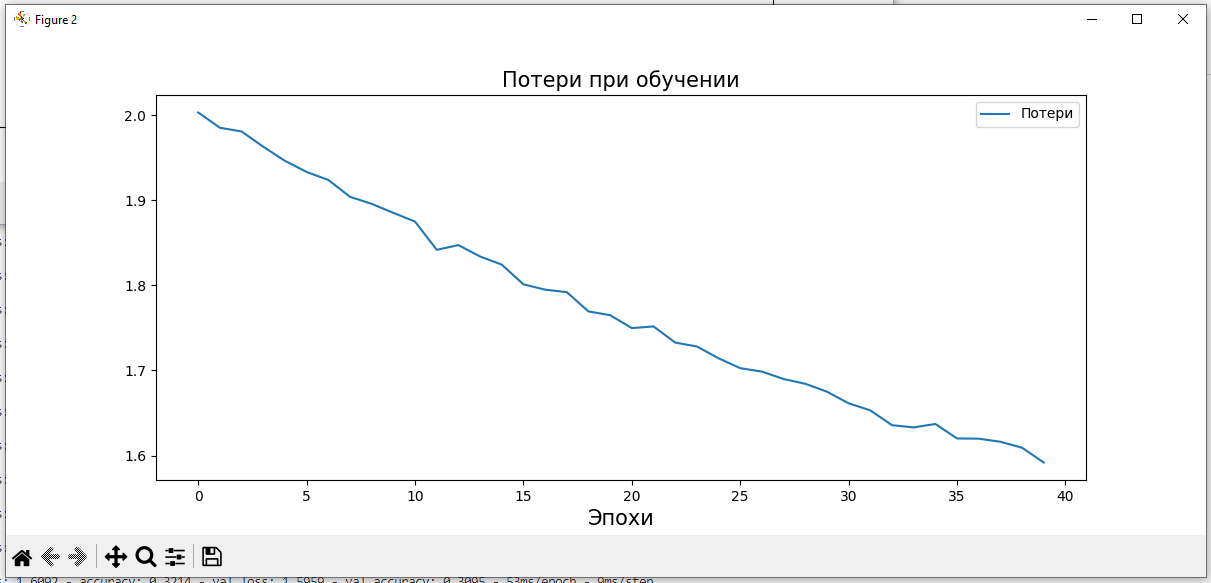
****

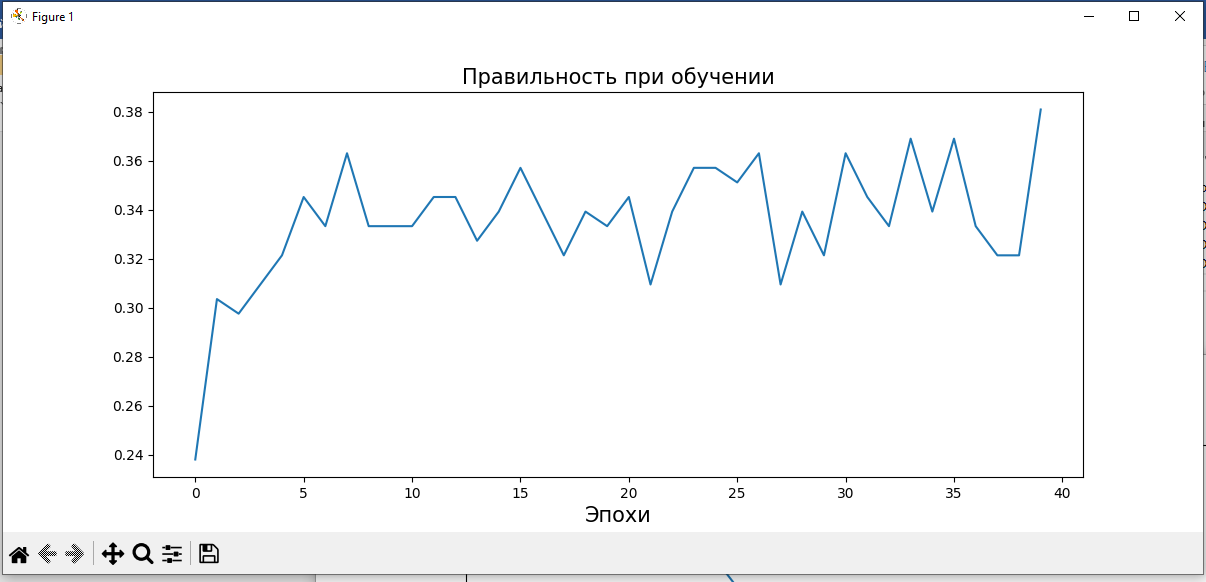
****

Для борьбы с переобучение сети можно попробовать добавить слой Dropout().

****

Получим следующие графики:

****

****

**Код программы**

from os import listdir

import os

from os.path import isfile, join

import tensorflow as tf

import matplotlib.pyplot as plt

mypath = "C:\\Users\\fox\\Desktop\\ii\\11lb\\catdog\_images"

file\_list = [f for f in listdir(mypath) if isfile(join(mypath, f))]

#print(onlyfiles)

fig = plt.figure(figsize=(10, 5))

for i, file in enumerate(file\_list):

    print(file)

    img\_raw = tf.io.read\_file("catdog\_images/"+file)

    img = tf.image.decode\_image(img\_raw)

    print('Форма изображения: ', img.shape)

    ax = fig.add\_subplot(3, 4, i+1)

    ax.set\_xticks([]); ax.set\_yticks([])

    ax.imshow(img)

    ax.set\_title(os.path.basename("catdog\_images/"+file), size=15)

#plt.tight\_layout()

#plt.show()

labels = [1 if 'dog' in os.path.basename(file) else 0 for file in file\_list]

print(labels)

ds\_files\_labels = tf.data.Dataset.from\_tensor\_slices((file\_list, labels))

for item in ds\_files\_labels:

    print((item[0].numpy(),item[1].numpy()))

def load\_and\_preprocess(path, label):

    image = tf.io.read\_file("catdog\_images/"+path)

    image = tf.image.decode\_jpeg(image, channels=3)

    image = tf.image.resize(image, [img\_height, img\_width])

    image /= 255.0

    return image, label

# ds\_files\_labels = tf.data.Dataset.from\_tensor\_slices((file\_list, labels))

img\_width, img\_height = 120, 80

ds\_images\_labels = ds\_files\_labels.map(load\_and\_preprocess)

fig = plt.figure(figsize=(10, 6))

for i, example in enumerate(ds\_images\_labels):

    ax = fig.add\_subplot(3, 4, i+1)

    ax.set\_xticks([]); ax.set\_yticks([])

    ax.imshow(example[0])

    ax.set\_title('{}'.format(example[1].numpy()), size=15)

plt.tight\_layout()

plt.show()

**2 часть**

import tensorflow as tf

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

from keras.utils import to\_categorical

url = "C:/Users/fox/Desktop/ii/data/seeds\_dataset.data"

names = ["площадь","периметр","компактность", "длина","ширина","асимметрия","длина канавки ядра","сорт"] #название атрибутов

target\_names = { 1:"Кама" , 2:"Роза" , 2:"Канадка" }

dataset = pd.read\_csv(url, names=names)

dataset.head()

X = dataset.iloc[:,0:7].values

y = dataset['сорт'].values

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=0)

from sklearn.preprocessing import StandardScaler

# scaler = StandardScaler()

# X\_train = scaler.fit\_transform(X\_train)

# X\_test = scaler.transform(X\_test)

model = tf.keras.models.Sequential([

  tf.keras.layers.Dense(16, input\_shape=(7,), name='fc1', activation='sigmoid'),

  tf.keras.layers.Dense(4, activation='softmax', name='fc2')

])

model.compile(optimizer='sgd', loss='sparse\_categorical\_crossentropy', metrics=['accuracy'])

model.summary()

# y\_train = to\_categorical(y\_train)

# y\_test = to\_categorical(y\_test)

print(y\_test)

cl = model.fit(X\_train, y\_train,verbose=2, epochs=40,validation\_data=(X\_test,y\_test),)

fig, ax = plt.subplots(figsize=(12,5))

ax.set\_title('Правильность при обучении', size=15)

plt.plot(cl.history['accuracy'], label='Правильность')

ax.set\_xlabel('Эпохи', size=15)

fig, ax = plt.subplots(figsize=(12,5))

ax.set\_title('Потери при обучении', size=15)

plt.plot(cl.history['loss'], label='Потери')

ax.set\_xlabel('Эпохи', size=15)

plt.legend()

plt.show()