**Государственное бюджетное общеобразовательное учреждение города Москвы "Школа № 1564 имени Героя Советского Союза А. П. Белобородова"**

**Отчет по лабораторной работе №**

по дисциплине «Программирование»

Выполнил: ученик 11Б класса

Проверила:

Изотова Анастасия Андреевна

Москва

2022

**Задание**

Многоклассовая классификация

- создать свой датасет на 4 класса и адаптировать программу под него.

**Код программы**

import os

import numpy as np

import tensorflow as tf

from tensorflow.keras.models import Sequential

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

from tensorflow.keras import utils

from tensorflow.keras.preprocessing import image

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

import matplotlib.pyplot as plt

from google.colab import files

%matplotlib inline

!rm -rf Fruit-Images-Dataset-master

!rm -f master.zip

!wget https://github.com/Alexandr223/Homework/raw/main/master.zip -O master.zip

!unzip master.zip

!ls -la

batch\_size=256

image\_size=(100, 100)

train\_dataset = image\_dataset\_from\_directory('Fruit-Images-Dataset-master/Training',

                                             subset='training',

                                             seed=42,

                                             validation\_split=0.1,

                                             batch\_size=batch\_size,

                                             image\_size=image\_size)

validation\_dataset = image\_dataset\_from\_directory('Fruit-Images-Dataset-master/Training',

                                             subset='validation',

                                             seed=42,

                                             validation\_split=0.1,

                                             batch\_size=batch\_size,

                                             image\_size=image\_size)

class\_names = train\_dataset.class\_names

class\_names

plt.figure(figsize=(8, 8))

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

  for i in range(16):

    ax = plt.subplot(4, 4, i + 1)

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

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

    plt.axis("off")

test\_dataset = image\_dataset\_from\_directory('Fruit-Images-Dataset-master/Test',

                                             batch\_size=batch\_size,

                                             image\_size=image\_size)

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 = Sequential()

# Сверточный слой

model.add(Conv2D(16, (5, 5), padding='same',

                 input\_shape=(100, 100, 3), activation='relu'))

# Слой подвыборки

model.add(MaxPooling2D(pool\_size=(2, 2)))

# Сверточный слой

model.add(Conv2D(32, (5, 5), activation='relu', padding='same'))

# Слой подвыборки

model.add(MaxPooling2D(pool\_size=(2, 2)))

# Сверточный слой

model.add(Conv2D(64, (5, 5), activation='relu', padding='same'))

# Слой подвыборки

model.add(MaxPooling2D(pool\_size=(2, 2)))

# Сверточный слой

model.add(Conv2D(128, (5, 5), activation='relu', padding='same'))

# Слой подвыборки

model.add(MaxPooling2D(pool\_size=(2, 2)))

# Полносвязная часть нейронной сети для классификации

model.add(Flatten())

model.add(Dense(1024, activation='relu'))

model.add(Dropout(0.2))

model.add(Dense(256, activation='relu'))

model.add(Dropout(0.2))

# Выходной слой, 4 нейрон по количеству классов

model.add(Dense(4, activation='softmax'))

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

              optimizer="adam",

              metrics=['accuracy'])

history = model.fit(train\_dataset,

                    validation\_data=validation\_dataset,

                    epochs=5,

                    verbose=2)

# Оцениваем качество обучения модели на тестовых данных

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

print("Доля верных ответов на тестовых данных, в процентах:", round(scores[1] \* 100, 4))

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()

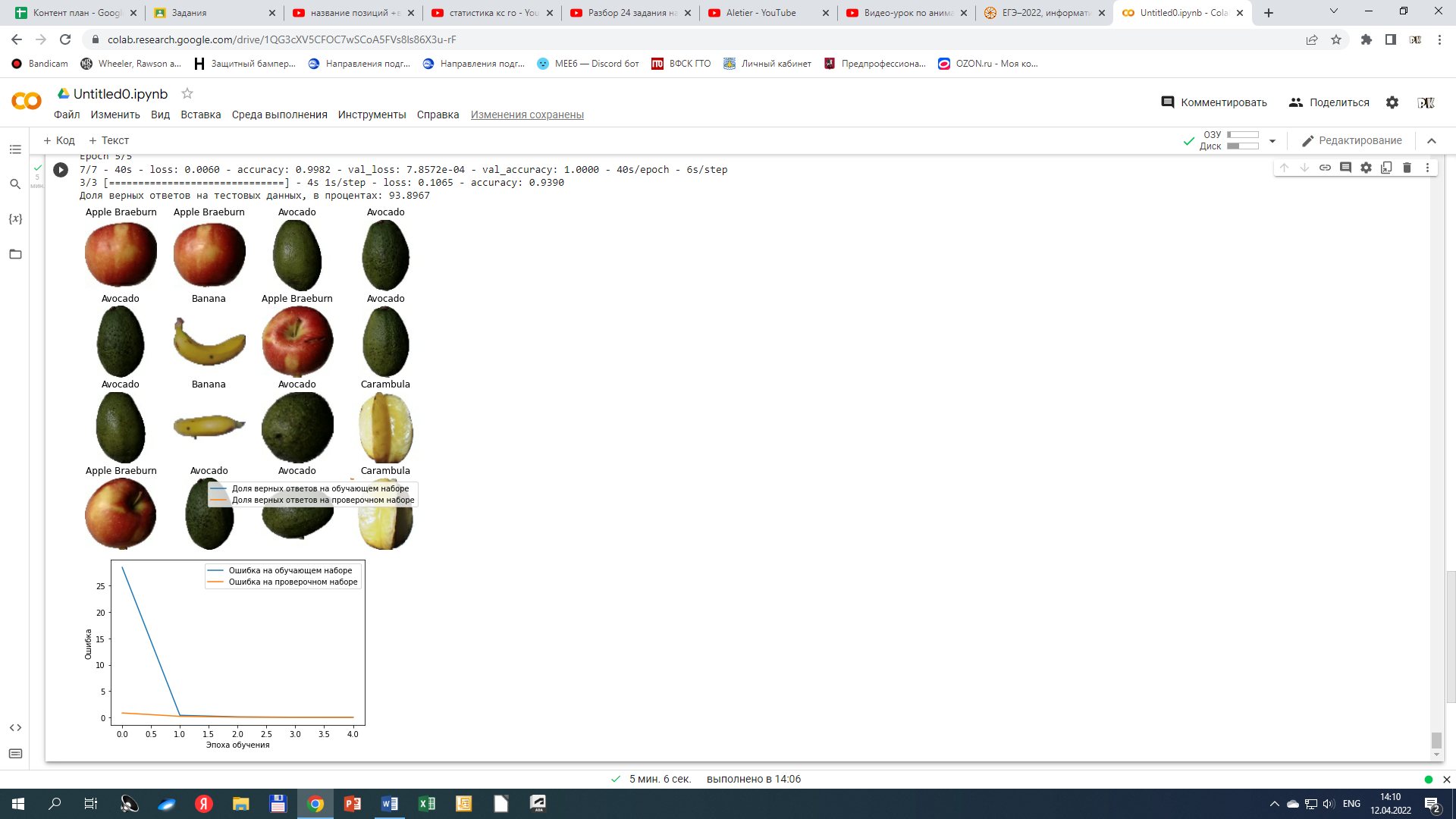
model.save("fruits\_360\_model.h5")

#files.download("fruits\_360\_model.h5")

**Реализация**

Рассмотрим код программы. Сначала я импортировал библиотеки для работы с дасетами и их созданием. С помощью команды **!wget** загружаем датасет, а командой **!unzip** распаковываем его. Переменными **batch\_size, image\_size** создаем датасеты. В переменную **train\_dataset** кладем набор данных для обучения и создаем проверочный набор данных в переменной **validation\_dataset.** Переменной **class\_names** «превращаем» фрукты в один класс. В цикле **for** делаем так, чтобы в конце вывело картинки фруктов и их названия. В переменную **test\_dataset** кладем набор данных для тестирования. Переменной **AUTOTUNE** настраиваем производительность датасета. Далее, переменной **model** создаем два слоя: свёрточный и подвыборки. Командой **model.compile** компилируем модель, а переменной **history** начинаем обучать нейросеть. Переменной **scores** оцениваем обучаемость нейросети. Далее командой **plt** выводим все. **model.save** сохраняем модель.

**Результат работы программы**



**Вывод**

Во время написания программы я ближе познакомился с библиотеками, связанными с созданием своей нейросети. Узнал из них новые функции, методы, как можно создать и обучить свою нейросеть.