Ушаков Владимир Александрович, группа МПиКИ

Лабораторная работа № 2

Обучение нейронной сети для классификации изображения

**Цель работы**

Решить задачу классификации исходного изображения с помощью

глубокой сверточной нейронной сети ( тип траспорта). Оценить точность

полученной модели. Не использовать переобученную нейронную сеть.**Задание**

**Код программы (внесённые изменения в шаблон кода выделены)**

import splitfolders  
import matplotlib.pyplot as plt  
  
from keras import layers  
from keras import models  
from keras.preprocessing.image import ImageDataGenerator  
from tensorflow import optimizers  
from sklearn.metrics import classification\_report, confusion\_matrix, ConfusionMatrixDisplay  
#%%  
classes = ['Bicycle', 'Car']  
#%%  
!rm -r splited  
splitfolders.ratio(f'data', output="splited", seed=1337, ratio=(0.8, 0.1, 0.1))  
  
validation\_dir = 'splited/val'  
test\_dir = 'splited/test'  
train\_dir = 'splited/train'  
#%%  
model = models.Sequential()  
model.add(layers.Conv2D(64, (3, 3), activation='relu',  
 input\_shape=(300, 300, 3)))  
model.add(layers.MaxPooling2D((2, 2)))  
model.add(layers.Conv2D(64, (3, 3), activation='relu'))  
model.add(layers.MaxPooling2D((2, 2)))  
model.add(layers.Conv2D(128, (3, 3), activation='relu'))  
model.add(layers.MaxPooling2D((2, 2)))  
model.add(layers.Conv2D(128, (3, 3), activation='relu'))  
model.add(layers.MaxPooling2D((2, 2)))  
model.add(layers.Conv2D(128, (3, 3), activation='relu'))  
model.add(layers.MaxPooling2D((2, 2)))  
model.add(layers.Conv2D(128, (3, 3), activation='relu'))  
model.add(layers.MaxPooling2D((2, 2)))  
model.add(layers.Flatten())  
model.add(layers.Dense(512, activation='relu'))  
model.add(layers.Dense(1, activation='sigmoid'))  
#%%  
model.compile(loss='binary\_crossentropy',  
 optimizer=optimizers.SGD(learning\_rate=0.01),  
 metrics=['accuracy'])  
#%%  
train\_datagen = ImageDataGenerator(rescale=1. / 255)  
validation\_datagen = ImageDataGenerator(rescale=1. / 255)  
test\_datagen = ImageDataGenerator(rescale=1. / 255)  
  
train\_generator = train\_datagen.flow\_from\_directory(  
 train\_dir,  
 target\_size=(300, 300),  
 batch\_size=20,  
 class\_mode='binary')  
validation\_generator = validation\_datagen.flow\_from\_directory(  
 validation\_dir,  
 target\_size=(300, 300),  
 class\_mode='binary')  
test\_generator = test\_datagen.flow\_from\_directory(  
 test\_dir,  
 target\_size=(300, 300),  
 shuffle=False,  
 class\_mode='binary')  
#%%  
history = model.fit(  
 train\_generator,  
 steps\_per\_epoch=50,  
 epochs=20,  
 validation\_data=validation\_generator,  
 validation\_steps=50)  
#%%  
acc = history.history['accuracy']  
val\_acc = history.history['val\_accuracy']  
loss = history.history['loss']  
epochs = range(1, len(acc) + 1)  
plt.plot(epochs, acc, 'bo', label='Training acc')  
plt.title('Training and validation accuracy')  
plt.legend()  
plt.figure()  
plt.plot(epochs, loss, 'bo', label='Training loss')  
plt.title('Training and validation loss')  
plt.legend()  
plt.show()  
#%%  
test\_loss, test\_acc = model.evaluate(test\_generator)  
#%%  
test\_acc  
#%%  
test\_generator.reset()  
y\_true = test\_generator.classes  
y\_pred = model.predict(test\_generator)  
#%%  
matrix = confusion\_matrix(y\_true, [0 if i < 0.5 else 1 for i in y\_pred])  
ConfusionMatrixDisplay(matrix).plot(cmap=plt.cm.Blues, values\_format='g')  
#%%  
res = classification\_report(y\_true, [0 if i < 0.5 else 1 for i in y\_pred])  
print(res)

**Результаты выполнения задания**

# В результате проведения обучения была достигнута точность классификации 84%

# Выводы

В результате выполнения работы была успешно обучена нейронная сеть для классификации изображений, а также была достигнута точность 84%