МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по практической работе №6

по дисциплине «Искусственные нейронные сети»

Тема: Процесс решения задач с применением нейронных сетей в библиотеке Keras

Студент гр. 8383 _____ Федоров И.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2021

Задача.

Необходимо построить сверточную нейронную сеть, которая будет классифицировать черно-белые изображения с простыми геометрическими фигурами на них. К каждому варианту прилагается код, который генерирует изображения. Для генерации данных необходимо вызвать функцию gen_data, которая возвращает два тензора:

1. Тензор с изображениями ранга 3.
2. Тензор с метками классов.

Вариант 2: Классификация изображений с закрашенным или не закрашенным кругом

Были загружены оба файла *gens.py* и *var2.py*, последний был подключен в программу.

Были подключены необходимые модули:

```
import numpy as np
import pandas
import matplotlib.pyplot as plt
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Convolution2D,
MaxPooling2D, Dense, Dropout, Flatten
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Sequential
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelEncoder
import var2
```

Была выполнена загрузка данных с помощью функции `gen_data()`, а также перемешивание данных:

```
data_stack, label_stack = var2.gen_data(size=data_size,
img_size=img_size)
idx = np.random.permutation(len(data_stack))
data, labels = data_stack[idx], label_stack[idx]
```

Было произведено преобразование текстовых меток к удобному для сети виду:

```
encoder = LabelEncoder()
encoder.fit(labels)
labels = encoder.transform(labels)          #[1, 1, 0, 0, 1 ... ]
```

Данные были разделены на тренировочный, проверочный и контрольный набора, а также была изменена форма тензоров с помощью функции `reshape()`:

```
test_data = data[:(len(data) // test_size)]
test_data = test_data.reshape((test_data.shape[0],
test_data.shape[1], test_data.shape[2], 1))
test_labels = labels[:(len(labels) // test_size)]

data = data[(len(data) // test_size):]
labels = labels[(len(labels) // test_size):]

validation_data = data[:(len(data) // valid_size)]
validation_data =
validation_data.reshape((validation_data.shape[0],
validation_data.shape[1], validation_data.shape[2], 1))
validation_labels = labels[:(len(labels) // valid_size)]

data = data[(len(data) // valid_size):]
labels = labels[(len(labels) // valid_size):]

train_data = data[:]
train_data = train_data.reshape((train_data.shape[0],
train_data.shape[1], train_data.shape[2], 1))
train_labels = labels[:]
```

Затем была сконструирована и обучена сверточная сеть:

```
### конструирование модели
model = Sequential()
model.add(Convolution2D(conv_depth_1, (kernel_size,kernel_size),
activation='relu', input_shape=(img_size,img_size, 1)))
model.add(MaxPooling2D((pool_size,pool_size)))

model.add(Convolution2D(conv_depth_2, (kernel_size,kernel_size),
activation='relu'))
model.add(MaxPooling2D((pool_size,pool_size)))

model.add(Convolution2D(conv_depth_2, (kernel_size,kernel_size),
activation='relu'))

model.add(Flatten())
model.add(Dropout(0.5))
model.add(Dense(256, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
# компиляция и обучение
model.compile(optimizer='rmsprop', loss='binary_crossentropy',
metrics=['accuracy'])
history = model.fit(train_data, train_labels,batch_size =
batch_size, epochs=num_epochs,
verbose=1, validation_data=(validation_data, validation_labels))
```

Ниже приведен график потерь и точности в процессе обучения и проверки. Точность на тестовых данных составила 96.2 %.