Практическое задание №6

Вариант №3

Условие:

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

Обратите внимание:

- Выборки не перемешаны, то есть наблюдения классов идут по порядку
- Классы характеризуются строковой меткой
- Выборка изначально не разбита на обучающую, контрольную и тестовую
- Скачивать необходимо оба файла. Подключать файл, который начинается с var (в нем и находится функция gen_data)

Вариант 3

Классификация изображений с горизонтальной или вертикальной линией

Выполнение:

С помощью фунции gen_data из файла var3 были сгенерированы данные. Т.к выборка была изначально упорядочена, данные были перемешаны.
Текстовые метки переведены в категориальный вектор

```
train_size = 1000
test_size = 400
data_size = train_size+test_size
img_size = 50
data, labels = var3.gen_data(data_size, img_size)
data, labels = shuffle(data, labels) #перемешивание

# переход от текстовых меток к категориальному вектору
encoder = LabelEncoder()
encoder.fit(labels)
encoded_labels = encoder.transform(labels)
encoded_labels = to_categorical(encoded_labels)
```

Данные были разбиты на тренировочные и тестовые, соотношение тренировочных данных к валидационным выбрано 0.1

```
train_data = data[:train_size]
train_labels = encoded_labels[:train_size]
val_split = 0.1

test_data = data[train_size:]
test_labels = encoded_labels[train_size:]
```

Были введены следующие гиперпараметры сети

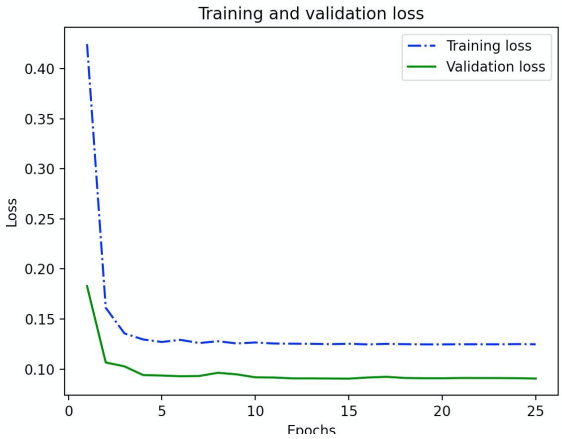
```
batch_size = 50
num_epochs = 25
kernel_size = 3
pool_size = 2
conv_depth_1 = 32
conv_depth_2 = 64
drop_prob_1 = 0.25
drop_prob_2 = 0.5
hidden_size = 64
```

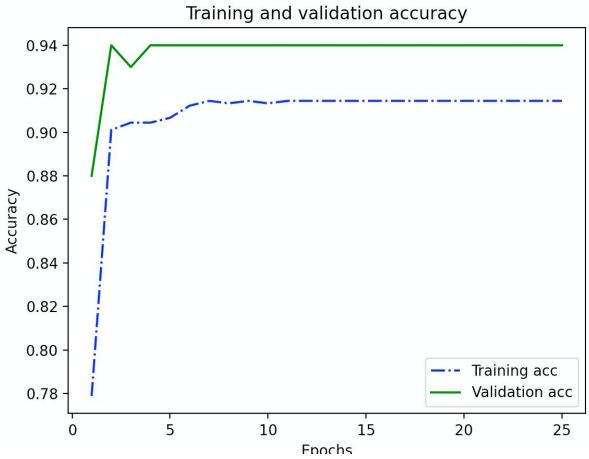
Была создана и обучена модель, модель состоит из двух сверточных слоев, слоя пулинга, слоя dropout, после этого входное изображение трансформируется в одномерный вектор и проходит два полносвязных слоя, после первого слоя используется слой dropout.

```
inp = Input(shape=(img_size, img_size, 1))
# Два сверточных слоя, слой пуллинга, слой dropout
conv_1 = Convolution2D(conv_depth_1, (kernel_size, kernel_size),
padding='same', activation='relu')(inp)
conv_2 = Convolution2D(conv_depth_1, (kernel_size, kernel_size),
padding='same', activation='relu')(conv_1)
pool_1 = MaxPooling2D(pool_size=(pool_size, pool_size))(conv_2)
drop_1 = Dropout(drop_prob_1)(pool_1)

flat = Flatten()(drop_1)
hidden = Dense(hidden_size, activation='relu')(flat)
drop_3 = Dropout(drop_prob_2)(hidden)
out = Dense(2, activation='softmax')(drop_3)

model = Model(inputs=inp, outputs=out)
model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])

history = model.fit(train_data, train_labels, batch_size=batch_size,
epochs=num_epochs, verbose=1, validation_split=val_split)
```

Были построены графики точности и потерь

Точность на обучающих данных составила 94%, на тестовых 91%


Training and validation loss
Training loss
Validation loss
Loss
Epochs
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0
5
10
15
20
25


Training and validation accuracy
Training acc
Validation acc
Accuracy
Epochs
0.94
0.92
0.90
0.88
0.86
0.84
0.82
0.80
0.78
0
5
10
15
20
25