Практическое задание №6

Вариант №6

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:

-Тензор с изображениями ранга 3

-Тензор с метками классов

Вариант: Классификация изображений по количеству крестов на них. Может быть 1, 2 или 3

Выполнение:

Были загружены данные.

```
(data, labels) = var6.gen_data()
```

Для того, чтобы смешать данные была использована функция shuffle из sklearn.utils.

```
data, labels = shuffle(data, labels)
```

Данные были дополнительно обработаны. Метки были закодированы с помощью энкодера и переведены в категориальный вид.

```
encoder = LabelEncoder()
encoder.fit(labels)
encoded_Y = encoder.transform(labels)
encoded_Y = np.reshape(encoded_Y, (size, 1))
encoded_Y = np_utils.to_categorical(encoded_Y, 3)
```

Остальной датасет был нормализован.

```
data = data.astype('float32')
data /= np.max(data)
```

Была построена модель, подобная модели из лабораторной работы №5.

```
inp = Input(shape=(height, width, 1))
```

```
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)
conv_3=Convolution2D(conv_depth_2,(kernel_size,kernel_size),
                      padding='same',
                      activation='relu')(drop_1)
conv_4=Convolution2D(conv_depth_2,(kernel_size,kernel_size),
                      padding='same',
                      activation='relu')(conv_3)
pool_2=MaxPooling2D(pool_size=(pool_size,pool_size))(conv_4)
drop_2 = Dropout(drop_prob_1)(pool_2)
flat = Flatten()(drop_2)
hidden = Dense(hidden_size, activation='relu')(flat)
drop_3 = Dropout(drop_prob_2)(hidden)
out = Dense(num_classes, activation='softmax')(drop_3)
model = Model(inputs=inp, outputs=out)
```

Со следующими гиперпараметрами.

```
size, height, width = data.shape
batch_size = 32
num_epochs = 20
kernel_size = 4
pool_size = 2
conv_depth_1 = 32
conv_depth_2 = 64
drop_prob_1 = 0.25
drop_prob_2 = 0.5
hidden_size = 512
```

Установка параметров и обучение модели.

```
model.compile(loss='categorical_crossentropy',optimizer='ada
m',metrics=['accuracy'])
hist=model.fit(data,encoded_Y,batch_size=batch_size,
epochs=num_epochs, validation_split=0.1)
```

Для проверки модели были сгенерированы тестовые данные.

```
data_test, labels_test = var6.gen_data(size=100)
data_test /= np.max(data_test)
encoder = LabelEncoder()
encoder.fit(labels_test)
```

```
encoded_Y = encoder.transform(labels_test)
encoded_Y = np.reshape(encoded_Y, (100, 1))
encoded_Y = np_utils.to_categorical(encoded_Y, 3)

results = model.evaluate(data_test, encoded_Y, verbose=1)
print(results)
```

Эти данные были обработаны схожим образом, однако их не пришлось перемешивать.

Полученная модель имеет точность 0.9677 на тренировочных данных, 0.98 на валидационных и 0.93 на тестовых.