Постановка задачи.

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

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 5

Классификация изображений с прямоугольником или не закрашенным кругом

Выполнение работы.

Генерация данных, перемешивание, нормализация, кодирование меток:

```
[data, labels] = gen_data(1000)
[data, labels] = shuffle(data, labels)

data /= np.max(data)

encoder = LabelEncoder()
encoder.fit(labels.ravel())
labels = encoder.transform(labels.ravel())
```

Разделение данных на обучающие, контрольные, тестовые:

```
data_len, height, width = data.shape

validation_ratio = 0.2
test_ratio = 0.2

train_data = data[: int(data_len*(1 - validation_ratio - test_ratio))]
train_labels = labels[: int(data_len*(1 - validation_ratio -
test_ratio))]

validation_data = data[int(data_len*(1 - validation_ratio - test_ratio)):
int(data_len*(1 - test_ratio))]
validation_labels = labels[int(data_len*(1 - validation_ratio -
test_ratio)): int(data_len*(1 - test_ratio))]

test_data = data[int(data_len*(1 - test_ratio)):]
test_labels = labels[int(data_len*(1 - test_ratio)):]


train_data = train_data.reshape(train_data.shape[0], width, height, 1)
validation_data = validation_data.reshape(validation_data.shape[0],
width, height, 1)
test_data = test_data.reshape(test_data.shape[0], width, height, 1)
```

Задание параметров обучения и архитектуры сети: два слоя свертки с глубиной 8, слой субдискретизации с размером области 2, слой dropout, еще два слоя свертки с глубиной 16, слой субдискретизации, слой сплющивания, полносвязный, dropout, выходной с функцией активации sigmoid.

```
model = Sequential()

model.add(Convolution2D(conv_depth_1, (kernel_size, kernel_size),
padding="same", activation='relu',
                        input_shape=(width, height, 1)))
model.add(Convolution2D(conv_depth_1, (kernel_size, kernel_size),
padding="same", activation='relu'))
model.add(MaxPooling2D(pool_size=(pool_size, pool_size)))
model.add(Dropout(drop_prob_1))

model.add(Convolution2D(conv_depth_2, (kernel_size, kernel_size),
padding="same", activation='relu'))
model.add(Convolution2D(conv_depth_2, (kernel_size, kernel_size),
padding="same", activation='relu'))
model.add(MaxPooling2D(pool_size=(pool_size, pool_size)))
model.add(Dropout(drop_prob_1))

model.add(Flatten())
model.add(Dense(hidden_size, activation='relu'))
```

```
model.add(Dropout(drop_prob_2))
model.add(Dense(1, activation='sigmoid'))

model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])
history = model.fit(train_data, train_labels, batch_size=batch_size,
epochs=num_epochs, verbose=1,
          validation_data=(validation_data, validation_labels))
```

Обучение происходит в течение 15 эпох с батчами по 32.

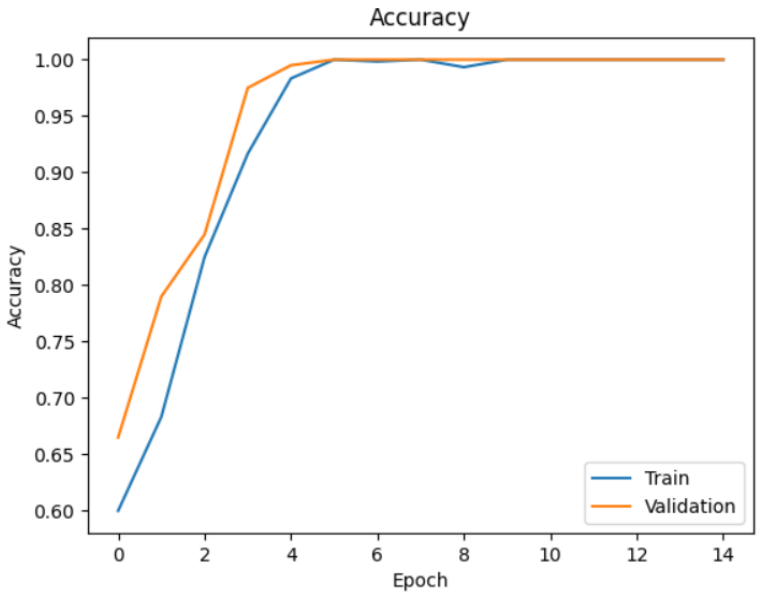

Рисунок 1. Точность на обучающих и контрольных данных

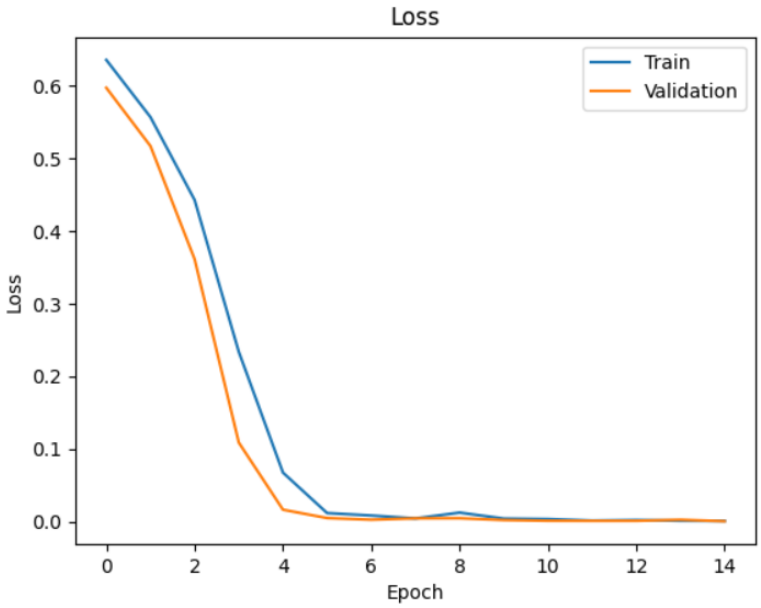

Рисунок 2. Ошибки на обучающих и контрольных данных

Достигается точность 100% на обучающих, контрольных и тестовых данных:

```
19/19 [==============================] - 1s 66ms/step - loss: 5.2895e-04
- accuracy: 1.0000 - val_loss: 3.3844e-04 - val_accuracy: 1.0000
7/7 [==============================] - 0s 13ms/step - loss: 2.5494e-04 -
accuracy: 1.0000
```