НЕЙРОННЫЕ СЕТИ

Зданович Александра, г. Петрозаводск
Детский технопарк Кванториум «Сампо», It-квантум
Наставник: Иван Ригоев

Задача моей нейронной сети отличать цветы : розы от ромашек. Для этого нужно создать нейронную сеть и обучить ее на фотографиях. В моем случае их получилось 800.

```
[ ]  # Каталог с данными для обучения
     train_dir = 'train'
     # Каталог с данными для проверки
     val_dir = 'val'
     # Каталог с данными для тестирования
     test_dir = 'test'
     # Размеры изображения
     img_width, img_height = 150, 150
     # Размерность тензора на основе изображения для входных данных в нейронную сеть
     # backend Tensorflow, channels_last
     input_shape = (img_width, img_height, 3)
     # Размер мини-выборки
     batch_size = 64
     # Количество изображений для обучения
     nb_train_samples = 500
     # Количество изображений для проверки
     nb_validation_samples = 250
     # Количество изображений для тестирования
     nb_test_samples = 150
```

Часть кода, отвечающая за изображения для обучения, проверки и тестирования

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, 150, 150, 3)       0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 150, 150, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 150, 150, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 75, 75, 64)        0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 75, 75, 128)       73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 75, 75, 128)       147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 37, 37, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 37, 37, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 37, 37, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 37, 37, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 18, 18, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 18, 18, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 18, 18, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 18, 18, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 9, 9, 512)         0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 9, 9, 512)         2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 9, 9, 512)         2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 9, 9, 512)         2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 4, 4, 512)         0
=================================================================
```

Часть кода, отвечающая за вид нашей сети, количество слоев в ней

```
Epoch 1/10
100/100 [==============================] - 70s 695ms/step - loss: 0.1746 - acc: 0.9247 - val_loss: 0.1228 - val_acc: 0.9496
Epoch 2/10
100/100 [==============================] - 60s 600ms/step - loss: 0.0620 - acc: 0.9791 - val_loss: 0.1028 - val_acc: 0.9570
Epoch 3/10
100/100 [==============================] - 61s 609ms/step - loss: 0.0390 - acc: 0.9868 - val_loss: 0.1448 - val_acc: 0.9543
Epoch 4/10
100/100 [==============================] - 61s 610ms/step - loss: 0.0349 - acc: 0.9876 - val_loss: 0.1122 - val_acc: 0.9657
Epoch 5/10
100/100 [==============================] - 60s 600ms/step - loss: 0.0240 - acc: 0.9916 - val_loss: 0.1178 - val_acc: 0.9610
Epoch 6/10
100/100 [==============================] - 60s 602ms/step - loss: 0.0341 - acc: 0.9895 - val_loss: 0.1233 - val_acc: 0.9523
Epoch 7/10
100/100 [==============================] - 61s 605ms/step - loss: 0.0246 - acc: 0.9908 - val_loss: 0.1587 - val_acc: 0.9550
Epoch 8/10
100/100 [==============================] - 59s 589ms/step - loss: 0.0258 - acc: 0.9910 - val_loss: 0.1426 - val_acc: 0.9597
Epoch 9/10
100/100 [==============================] - 60s 599ms/step - loss: 0.0177 - acc: 0.9933 - val_loss: 0.1831 - val_acc: 0.9557
Epoch 10/10
100/100 [==============================] - 60s 604ms/step - loss: 0.0212 - acc: 0.9924 - val_loss: 0.1811 - val_acc: 0.9631
<tensorflow.python.keras.callbacks.History at 0x7fe054816f60>
```

Обучение нейронной сети

```
[ ]
scores = model.evaluate_generator(test_generator, 50)
```

```
[ ]
print("Аккуратность на тестовых данных: %.2f%%" % (scores[1]*100))
```

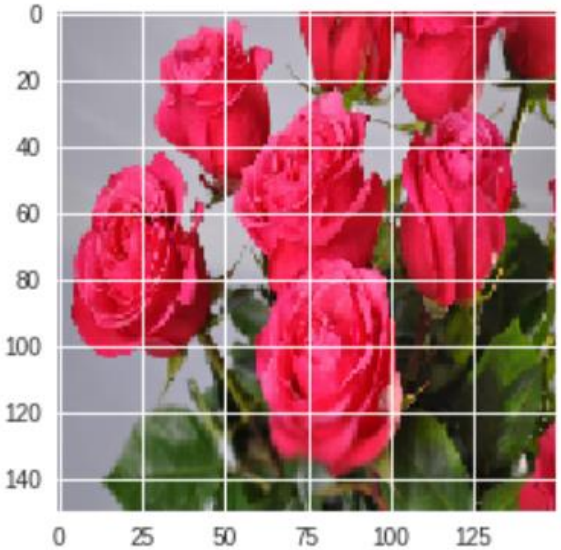
```
Аккуратность на тестовых данных: 98.26%
```

Точность тестовых данных

```
[ ]  from tensorflow.python.keras.datasets import mnist
     from tensorflow.python.keras.utils import np_utils
     from tensorflow.python.keras.models import model_from_json

     import numpy as np

     from keras.applications.vgg16 import preprocess_input

     print("Загружаю сеть из файлов")
     # Загружаем данные об архитектуре сети
     json_file = open("vgg16_tankplane_model.json", "r")
     loaded_model_json = json_file.read()
     json_file.close()
     # Создаем модель
     model = model_from_json(loaded_model_json)
     # Загружаем сохраненные веса в модель
     model.load_weights("vgg16_tankplane5epoch_bs32.h5")
     print("Загрузка сети завершена")



     # для предсказания обученной модели
     from tensorflow.python.keras.preprocessing import image
     img = image.load_img('roses  3.jpg', target_size=(150, 150))
     plt.imshow(img)
     plt.show()

     x = image.img_to_array(img)
     x = np.expand_dims(x, axis=0)
     x = preprocess_input(x)

     prediction = model.predict(x)

     print(prediction)

     if prediction > 0.5:
       print("Роза")
     else:
       print("Ромашка")
```

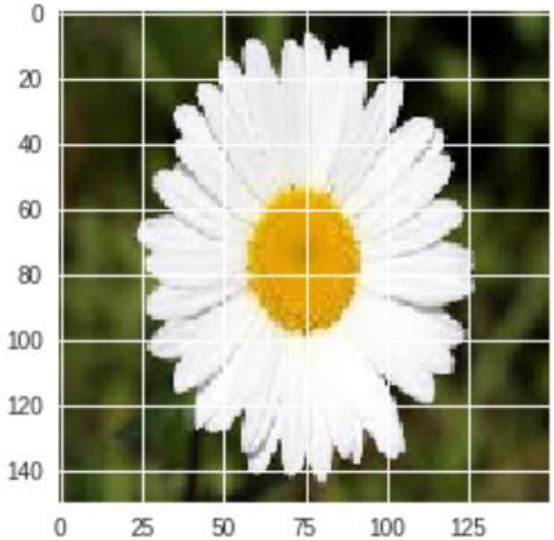
Эта часть кода отвечает за отличие роз от ромашек

```
Загружаю сеть из файлов
Загрузка сети завершена
```




```
[[1.]]
Роза
```

```
Загружаю сеть из файлов
Загрузка сети завершена
```




```
[[0.]]
Ромашка
```

Загружаем картинки на проверку и смотрим на результаты.