```
from PIL import Image
import numpy as np
import os
from random import shuffle
from tqdm import tqdm
import matplotlib.pyplot as plt


import time


IMG_SIZE = 120
LR = 1e-3


TRAIN_DIR = "/content/drive/MyDrive/Flowers_Training"
TEST_DIR =  "/content/drive/MyDrive/Flowers_Testing"


def label_img(img):

    word_label = img.split('_')[0]
    print(word_label)
    if word_label == 'lily': return [1,0]
    elif word_label == 'rose': return [0,1]


def create_train_data():
    train_data = []
    for img in tqdm(os.listdir(TRAIN_DIR)):
        label = label_img(img)
        path = os.path.join(TRAIN_DIR, img)



        img = Image.open(path)
        img = img.convert('L')
        img = img.resize((IMG_SIZE, IMG_SIZE), Image.ANTIALIAS)

        train_data.append([np.array(img), np.array(label)])
    shuffle(train_data)
    np.save('train_data.npy', train_data)
    return train_data


def process_test_data():
    test_data = []
    for img in tqdm(os.listdir(TEST_DIR)):
        imagename = img
        path = os.path.join(TEST_DIR, img)
        img_num = img.split('_')[1]
        img = Image.open(path)
```

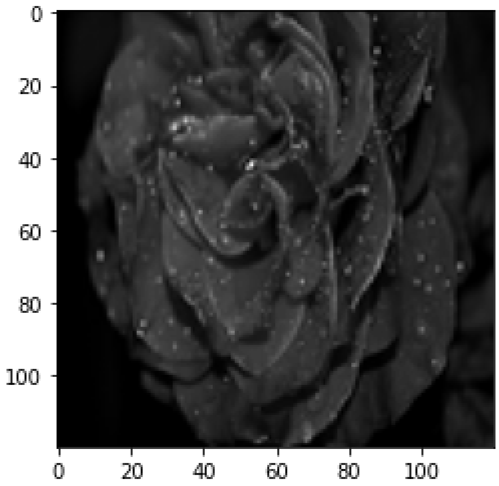
```
            img = img.convert('L')
            img = img.resize((IMG_SIZE, IMG_SIZE), Image.ANTIALIAS)

            test_data.append([np.array(img),imagename])
        shuffle(test_data)
        np.save('test_data.npy', test_data)
        return test_data


    train_data = create_train_data()
```

```
     lily
     lily
     lily
     lily
      46%|████▋     | 37/80 [00:01<00:01, 25.87it/s]lily
     lily
     lily
     lily
     lily
     lily
      51%|█████▏    | 41/80 [00:01<00:01, 28.81it/s]/usr/local/lib/python3.7/dist-package
       "Palette images with Transparency expressed in bytes should be "
     lily
     lily
     rose
     rose
     rose

     rose
     rose
     rose
     rose
     rose
     rose
      68%|██████▊   | 54/80 [00:01<00:00, 36.98it/s]rose
     lily
     lily
     lily
     lily
     lily
     lily
     lily
     lily
     lily
      74%|███████▍  | 59/80 [00:01<00:00, 39.73it/s]lily
     rose
     rose
     rose
      80%|████████  | 64/80 [00:02<00:00, 29.59it/s]rose
     rose
     rose
     rose
      85%|████████▌ | 68/80 [00:02<00:00, 25.60it/s]rose
     rose
     rose
     lily
```

```
lily
100%|██████████| 80/80 [00:02<00:00, 30.43it/s]lily
lily
lily
lily
lily
lily
lily
lily

/usr/local/lib/python3.7/dist-packages/numpy/core/_asarray.py:136: VisibleDeprecatio
  return array(a, dtype, copy=False, order=order, subok=True)
```

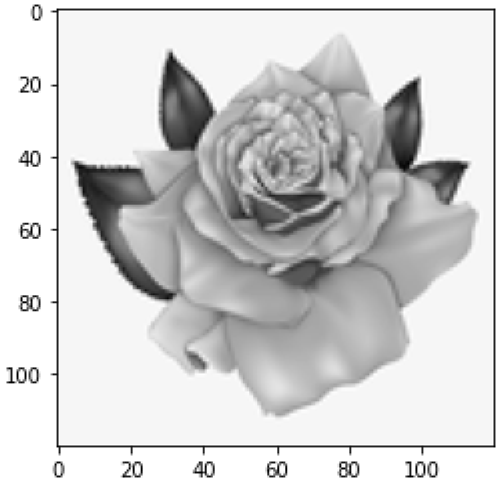
```
plt.imshow(train_data[0][0], cmap = 'gist_gray')
print(train_data[0][1])
```

```
[0 1]
```



```
plt.imshow(train_data[3][0], cmap = 'gist_gray')
print(train_data[0][1])
```

```
[0 1]
```



```
input shape = (120, 120, 1)
```

```
nClasses = 2


def createModel():
    model = Sequential()
    model.add(Conv2D(32, (5, 5), padding='same', activation='relu', input_shape=input_shape))
    model.add(Conv2D(32, (5, 5), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))


    model.add(Conv2D(64, (5, 5), padding='same', activation='relu'))
    model.add(Conv2D(64, (5, 5), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))


    model.add(Conv2D(64, (5, 5), padding='same', activation='relu'))
    model.add(Conv2D(64, (5, 5), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))


    model.add(Flatten())
    model.add(Dense(512, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(nClasses, activation='softmax'))
    return model
```

```
x = list(range(1,101))
print(x[-90:])
print(x[:-90])
```

```
[11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

```
train = train_data[-90:]
test = train_data[:-90]


X = np.array([i[0] for i in train]).reshape(-1, IMG_SIZE, IMG_SIZE, 1)
Y = np.array([i[1] for i in train])

test_x = np.array([i[0] for i in test]).reshape(-1, IMG_SIZE, IMG_SIZE, 1)
test_y = np.array([i[1] for i in test])
```

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import keras
import keras
from keras.models import Sequential
from tensorflow.keras.utils import to_categorical
from keras.layers import Dense, Conv2D, MaxPooling2D, Dropout, Flatten


model = createModel()


epochs = 50
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])


model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_6 (Conv2D)            (None, 120, 120, 32)      832
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 116, 116, 32)      25632
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 58, 58, 32)        0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 58, 58, 64)        51264
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 54, 54, 64)        102464
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 27, 27, 64)        0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 27, 27, 64)        102464
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 23, 23, 64)        102464
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 11, 11, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 7744)              0
_________________________________________________________________
dense_2 (Dense)              (None, 512)               3965440
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_3 (Dense)              (None, 2)                 1026
=================================================================
Total params: 4,351,586
Trainable params: 4,351,586
Non-trainable params: 0
_________________________________________________________________
```

```
model.fit(x = X,y=Y, epochs=epochs, verbose=1,
                   validation_data=(test_x, test_y))
```

```
3/3 [==============================] - 11s 3s/step - loss: 0.4877 - accuracy: 0.8266
Epoch 23/50
3/3 [==============================] - 11s 3s/step - loss: 0.4183 - accuracy: 0.8023
Epoch 24/50
3/3 [==============================] - 11s 3s/step - loss: 0.3767 - accuracy: 0.8242
```

```
3/3 [==============================] - 11s 3s/step - loss: 0.3767 - accuracy: 0.8242
Epoch 25/50
3/3 [==============================] - 11s 3s/step - loss: 0.2899 - accuracy: 0.8922
Epoch 26/50
3/3 [==============================] - 11s 3s/step - loss: 0.2625 - accuracy: 0.9133
Epoch 27/50
3/3 [==============================] - 11s 3s/step - loss: 0.2218 - accuracy: 0.9258
Epoch 28/50
3/3 [==============================] - 11s 3s/step - loss: 0.2372 - accuracy: 0.9391
Epoch 29/50
3/3 [==============================] - 11s 3s/step - loss: 0.2839 - accuracy: 0.8766
Epoch 30/50
3/3 [==============================] - 11s 3s/step - loss: 0.3004 - accuracy: 0.8852

Epoch 31/50
3/3 [==============================] - 11s 3s/step - loss: 0.3362 - accuracy: 0.8539
Epoch 32/50
3/3 [==============================] - 11s 3s/step - loss: 0.2777 - accuracy: 0.8750
Epoch 33/50
3/3 [==============================] - 11s 3s/step - loss: 0.2287 - accuracy: 0.8914
Epoch 34/50
3/3 [==============================] - 11s 3s/step - loss: 0.3646 - accuracy: 0.8523
Epoch 35/50
3/3 [==============================] - 11s 3s/step - loss: 0.2161 - accuracy: 0.9133
Epoch 36/50
3/3 [==============================] - 11s 3s/step - loss: 0.1224 - accuracy: 0.9898
Epoch 37/50
3/3 [==============================] - 11s 3s/step - loss: 0.1526 - accuracy: 0.9437
Epoch 38/50
3/3 [==============================] - 11s 3s/step - loss: 0.0718 - accuracy: 0.9656
Epoch 39/50
3/3 [==============================] - 10s 3s/step - loss: 0.0496 - accuracy: 0.9820
Epoch 40/50
3/3 [==============================] - 11s 3s/step - loss: 0.0585 - accuracy: 0.9820
Epoch 41/50
3/3 [==============================] - 10s 3s/step - loss: 0.0290 - accuracy: 0.9938
Epoch 42/50
3/3 [==============================] - 10s 3s/step - loss: 0.0265 - accuracy: 0.9820
Epoch 43/50
3/3 [==============================] - 11s 3s/step - loss: 0.0118 - accuracy: 1.0000
Epoch 44/50
3/3 [==============================] - 10s 3s/step - loss: 0.0137 - accuracy: 1.0000
Epoch 45/50
3/3 [==============================] - 10s 3s/step - loss: 0.0046 - accuracy: 1.0000
Epoch 46/50
3/3 [==============================] - 10s 3s/step - loss: 0.0029 - accuracy: 1.0000
Epoch 47/50
3/3 [==============================] - 10s 3s/step - loss: 0.0029 - accuracy: 1.0000
Epoch 48/50
3/3 [==============================] - 10s 3s/step - loss: 0.0097 - accuracy: 1.0000
Epoch 49/50
3/3 [==============================] - 10s 3s/step - loss: 7.8669e-04 - accuracy: 1.
Epoch 50/50
3/3 [==============================] - 10s 3s/step - loss: 0.0028 - accuracy: 1.0000
<keras.callbacks.History at 0x7fde30ea7910>
```

```
test_img = process_test_data()[2];
```

```
100%|██████████| 20/20 [00:09<00:00,  2.21it/s]
/usr/local/lib/python3.7/dist-packages/numpy/core/_asarray.py:136: VisibleDeprecationWa
  return array(a, dtype, copy=False, order=order, subok=True)
```

```
test_img
```

```
[array([[220, 219, 219, ..., 211, 212, 213],
        [219, 218, 218, ..., 210, 209, 211],
        [219, 216, 218, ..., 208, 208, 210],
        ...,
        [221, 219, 218, ..., 214, 215, 217],
        [221, 219, 218, ..., 213, 213, 216],
        [222, 221, 221, ..., 214, 214, 217]], dtype=uint8), 'lily_2.webp']
```

```
test_img = test_img[0].reshape(IMG_SIZE, IMG_SIZE, 1)
```

```
model.predict(np.array([test_img]))
```

```
array([[0.00635376, 0.99364626]], dtype=float32)
```

✓ 0s completed at 8:24 PM ● ✕