# Flowers Recognition

Carmel Ron

## Problem 1: Identify whether an image is an image of a flower or not

### Database:

I combined 3 different databases and created a new database with 2 labels: is this a picture of a flower or a picture that is not a flower.

I took the pictures of the flowers from the following databases: Natural images [1], flowers-recognition [2], Oxford flowers category [3].

I took the pictures of the non-flowers from Natural images [1] database which contains pictures of people, dogs, cats, motorcycles, aircraft, cars and fruits.

In total I had 6000 pictures of flowers and 6000 pictures of no flowers.

### Pre-processing:

I divided the database into Train (50%) and Test (50%), In addition I divided the training into 20 % of validations.

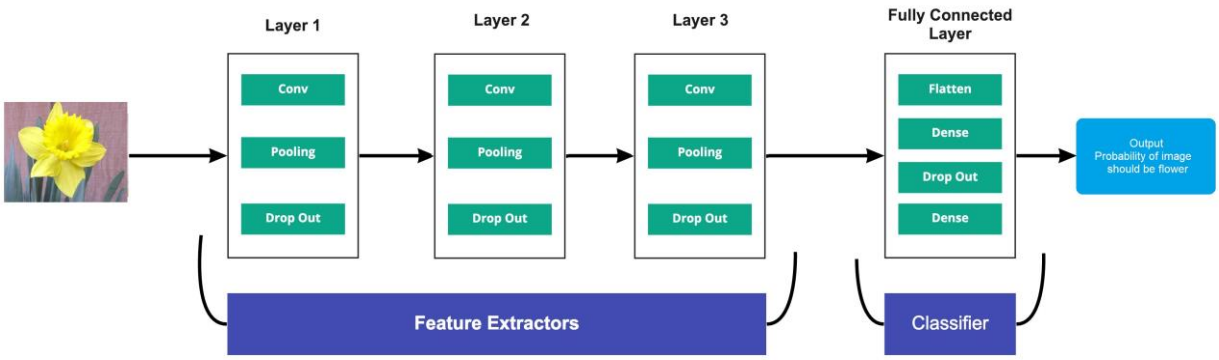
Each image was resized to 128 by 128 pixels.

### CNN – model:

The input layer - the image

In the hidden layers I put 3 layers of convolution in each of them I put max-polling and dropout and at the end a fully connected layer.

In the output layer I put two neurons with a softmax function that would give the probability that the image is an image of a flower or not.




Training parameters:

Batch size = 15, epochs = 20, loss = categorical cross entropy.

Results:

In training I got the following results: accuracy: 0.9781, validation accuracy: 0.9983

In the test I got an accuracy of 0.9253

Confusion matrix:

| | Predicted: flower | Predicted: no flower |
|---|---|---|
| Actual: flower | 2581 (TP) | 419 (FN) |
| Actual: no flower | 29 (FP) | 2971 (TN) |

Precision- 0.99

Recall 0.86

f- score 0.92

## Conclusions:

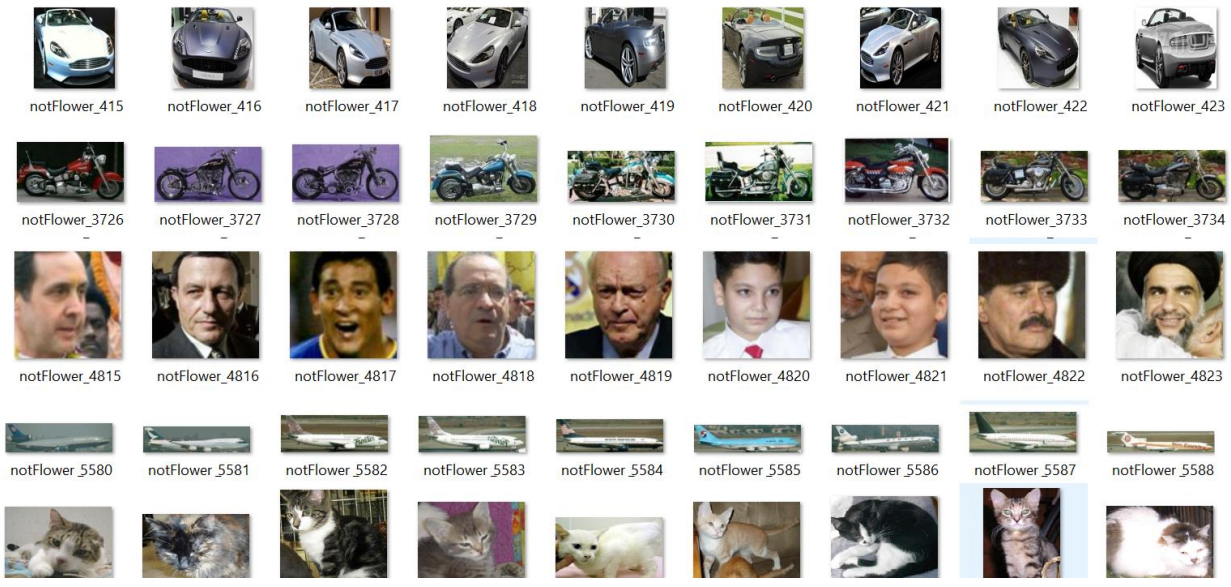
The results I got are very good results, over 90 percent accuracy on the test.

The problem was when I entered the network as input pictures I took myself or pictures I took from Google.

What happened was that even when I brought pictures to the network that were definitely not flowers, the network classified them as flowers.

My conclusion is that the network practiced on examples of very specific “no flowers” and quite similar in their idea to each other.

In order for the network to be really good in the real world I have to completely change my pictures to give it a lot more pictures of "no flowers" from different and varied sources.




## Problem 2: Identify What species of flower is in the picture?

## Database:

I used flowers-recognition database [2], this database contains only images of flowers and divides them into five types of flowers: daisy, dandelion, rose, sunflower, tulip. (Total – 4326)

## Pre-processing:

### Feature Extraction:

When deciding about the features that could quantify plants and flowers, we could possibly think of Color, Texture and Shape as the primary ones.

As these species have many attributes in common like flowers with same color. So, we need to quantify the image by combining different feature descriptors so that it describes the image more effectively [5].

### Color - Color Histogram

Histogram with number of pixels in each color.

### Shape - Hu Moments:

Image moments are a weighted average of image pixel intensities.

Hu Moments are a set of 7 numbers calculated using central moments that are invariant to image transformations like: translation, scale, rotation, and reflection.

### Texture - Haralick Texture

textural features based on gray-tone spatial dependancies. [6]

Haralick texture features calculated from a gray level co-occurrence matrix (GLCM) is a common method to represent image texture, as it is simple to implement and results in a set of interpretable texture descriptors

A known use of it is to analyze CT images.


## SVM – model:

Results:

accuracy on train: 0.63

accuracy on test: 0.52

| | daisy | dandelion | rose | sunflower | tulip |
|---|---|---|---|---|---|
| daisy | 100 | 26 | 10 | 13 | 5 |
| dandelion | 52 | 101 | 11 | 38 | 8 |
| rose | 30 | 12 | 45 | 26 | 44 |
| sunflower | 10 | 9 | 2 | 122 | 4 |
| tulip | 33 | 20 | 29 | 29 | 86 |

to deal with non-linear data sets I used the Kernel solutions

Polynomial Kernel:

accuracy on train: 0.65

accuracy on test: 0.56

Radial Basis Function:

accuracy on train: 0.62

accuracy on test: 0.53

## Random Forest – model:

Results:

accuracy on train: 0.9

accuracy on test: 0.66

| | daisy | dandelion | rose | sunflower | tulip |
|---|---|---|---|---|---|
| daisy | 101 | 27 | 12 | 10 | 4 |
| dandelion | 17 | 159 | 6 | 21 | 7 |
| rose | 5 | 12 | 87 | 5 | 48 |
| sunflower | 9 | 14 | 2 | 110 | 12 |
| tulip | 7 | 19 | 39 | 10 | 122 |

## Neural Network – model:

I used the input as in the SVM and random forest model.

In this model I have tried to make very many types of networks. That is, I tried a different number of layers and a different number of neurons.

At first, I was over-fitting so I added a layer of Dropout as well as Early-Stopping.

To use Early-Stopping I had to divide the train into train and validations.

After many attempts I created the structure of the network to be with two hidden layers, one with 100 neurons and one with 30 neurons.

The output layer had five neurons, one for each species of flower.

This model brought me the best results on training.

Results:

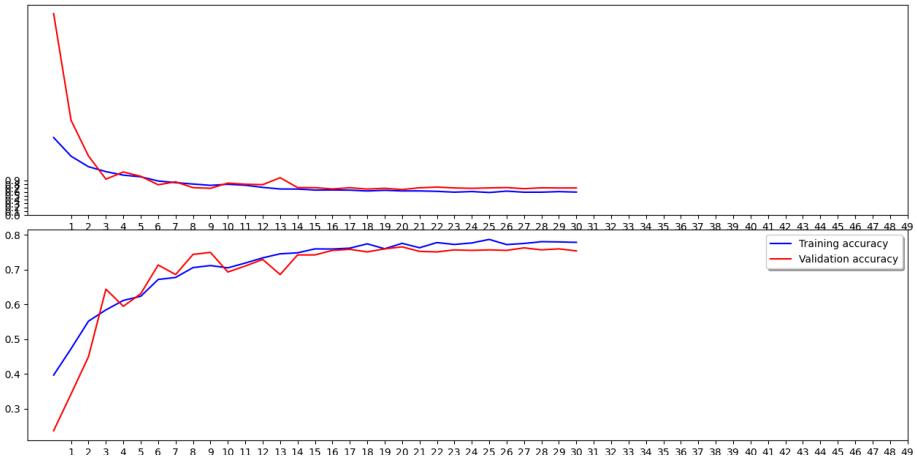
Accuracy on train: 0.7115, accuracy on validation:  0.59

Accuracy on test: 0.58

| | daisy | dandelion | rose | sunflower | tulip |
|---|---|---|---|---|---|
| daisy | 94 | 38 | 5 | 10 | 7 |
| dandelion | 35 | 127 | 11 | 26 | 11 |
| rose | 16 | 13 | 68 | 12 | 48 |
| sunflower | 16 | 18 | 2 | 101 | 10 |
| tulip | 19 | 12 | 33 | 21 | 112 |

## CNN – model:

I used an architecture like what I did in the first part, only this time in the output layer there are 5 neurons each showing the probability of the flower being a particular species.




Results:

Accuracy on train: 0.75

Accuracy on test: 0.74

| | daisy | dandelion | rose | sunflower | tulip |
|---|---|---|---|---|---|
| daisy | 108 | 27 | 8 | 8 | 3 |
| dandelion | 3 | 190 | 2 | 13 | 2 |
| rose | 6 | 13 | 89 | 19 | 30 |
| sunflower | 3 | 12 | 1 | 130 | 1 |
| tulip | 4 | 22 | 33 | 12 | 126 |

## Conclusions:

At first, I divided the database to training and testing evenly (50 %, 50%) but for better results I changed the division to 80%, 20%.

As expected, the SVM without kernel brought the least good results and that is because it separates linearly.

When I added kernel to SVM the result on test was better.

On the other hand, I was sure that a neural network would bring a better result than random forest, but it was the opposite.

I checked and saw that there are cases where random forest brings better results.

It happened like this because I used as input the histograms of the image and not the image itself as input.

In the CNN model which is basically a kind of neural network I used as input in the image itself and indeed I got much better results.

To solve overfitting in the models of neural networks and CNN I added a layer of Dropout and Early-Stopping.

## References:


[1] A. Mamaev, "flowers-recognition, kaggle," 2018. [Online]. Available: https://www.kaggle.com/prasunroy/natural-images.

[2] P. Roy, "natural-images, kagale," 2018. [Online]. Available: https://www.kaggle.com/prasunroy/natural-images.

[3] M.-E. Nilsback, "17 Category Flower Dataset," Oxford, [Online]. Available: http://www.robots.ox.ac.uk/~vgg/data/flowers/17/index.html.

[4] Uysim, "Keras CNN Dog or Cat Classification.kaggle," 2019. [Online]. Available: https://www.kaggle.com/uysimty/keras-cnn-dog-or-cat-classification.

[5] G. Ilango, "Image Classification using Python and Scikit-learn," 2017. [Online]. Available: https://gogul.dev/software/image-classification-python.

[6] R. M. Haralick, "12," 1973. [Online]. Available: http://haralick.org/journals/TexturalFeaturesHaralickShanmugamDinstein.pdf.