# Homework assignment 2 (Image classification)

BENSRHIER Nabil

November 21, 2020

## Flowers-recognition using CNN

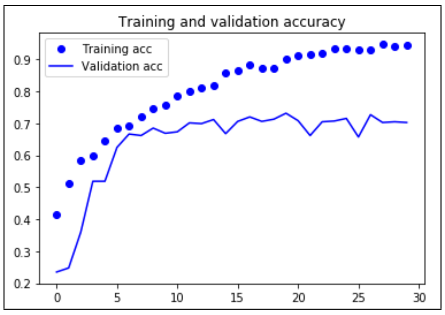
After loading and preprocessing the data, we divided it into 3 sets: **train, validation and test** set. We create data generators for each set using a batch size of 20, "categorical" class mode since we have 5 labels, and adding the parameter **shuffle** = **False** for the test generator (This is needed to have the correct accuracy while predicting).
Our network should train, validate and test all samples, so that we adjust the number of steps per epochs for every set:

```
steps = len(samples) // batch_size
```

Then we train 4 different models:

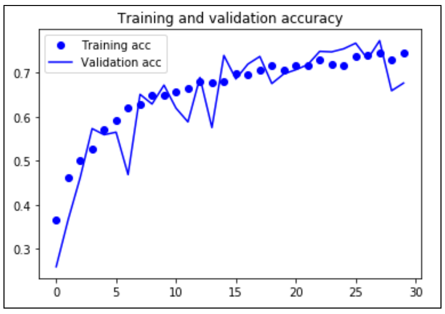
## Model 1: Simple CNN

We use five "Conv2D" layers, each with a pooling layer, bacth normalization and dropout following it. The first layer uses $2^N$ nodes in each layer (32, 64, 96,..). The filter size for all of them is 3 squared pixels except the first with 5 squared pixels. A "flatten" layer that turns the inputs into a vector. Two "dense" layers, the first one takes vector, and the second generates probabilities for 5 target labels, using a **Softmax** activation function.



Here we use two techniques: **dropout** that means dropout some nodes in the network so as to generalize. Then second is **batch normalization** which normalize the input of each layer that helps the notwork to learn fatser.
As we can see, even if we use these two techniques our model works perfectly on training data but not on unseen data, which a sign of overfitting.

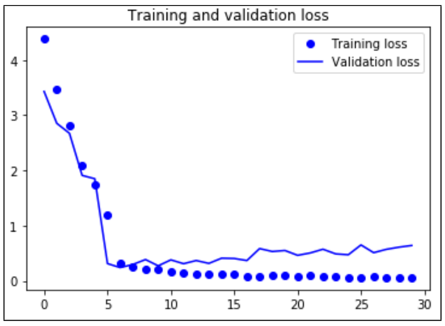
## Model 2: The same architecture adding data augmentation on training data



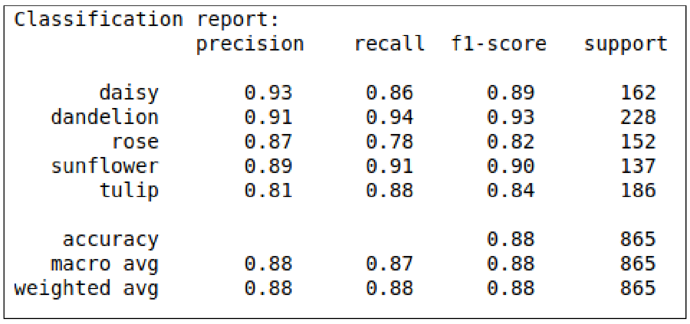
Data augmentation is a technique that creates random perturbations which involves generating transformed versions of images so as to increase the generalizability of the model. As we can see there is no gap between validation and training performance. It garanteed to work on unseen data.

## Model 3: Using a Pretrained Convolutional Base : InceptionV3

In this case, we extend the model we have 'conv_base' by adding Dense layers on top, that means we use the output of the InceptionV3 model as inputs to our new simple model: **Note** that the extracted features are of shape (samples, 3, 3, 2048).
We reach a validation accuracy of 80%, much better than we could achieve in the previous section, but we still have overvitting



## Model 4: InceptionV3 using data augmentation



```
Classification report:
              precision    recall  f1-score   support

       daisy       0.93      0.86      0.89       162
   dandelion       0.91      0.94      0.93       228
        rose       0.87      0.78      0.82       152
   sunflower       0.89      0.91      0.90       137
       tulip       0.81      0.88      0.84       186

    accuracy                           0.88       865
   macro avg       0.88      0.87      0.88       865
weighted avg       0.88      0.88      0.88       865
```

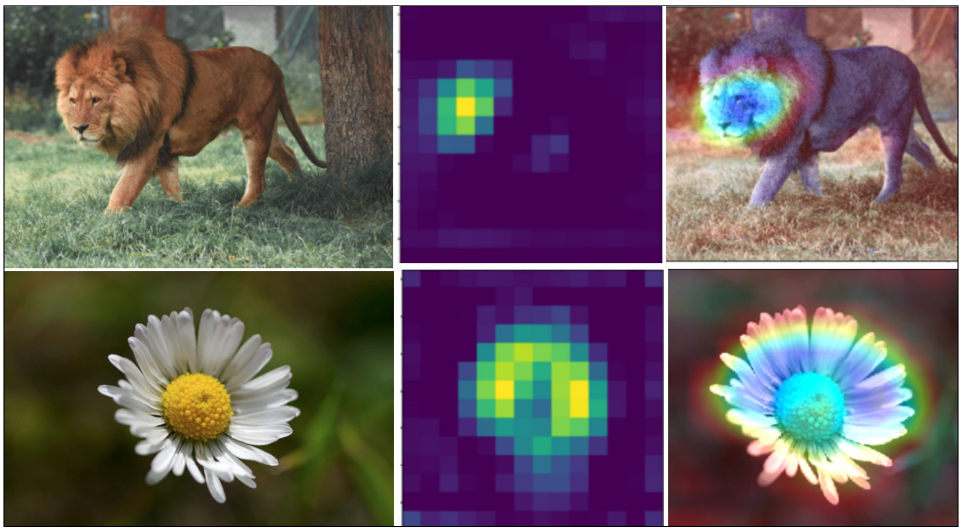
This architecture allows us to use data augmentation, because every input image is going through the convolutional base every time it is seen by the model, but it is more expensive than the third model.

### Conclusion:

The last model is the best one, since it pretrained and uses data augmentation which reduces overfitting and gives good accuracy on unseen data.

```
(test_loss_1, test_acc_1), (test_loss_2, test_acc_2), (test_loss_3, test_acc_3),
    (test_loss_4, test_acc_4)
((1.1903374195098877, 0.7132948040962219),
 (0.9700126051902771, 0.6728323698043823),
 (0.5586255788803101, 0.8150289058685303),
 (0.16645976901054382, 0.8797687888145447))
```

## Visualizing heatmaps of class activation:



In the above images, you can see how **Class Activation Map** works. Starting from left, first are our input images, then comes the activation heatmap generated by this technique, and finally the heatmap over the input image. Our VGG16 model predicts:

- The first input as a lion class with a probability of 64.9% and a dhole class with 22.9%.
- The second input as daisy class with a probability of 99%.

In the second image, it is pretty clear that the network has no problem in classifying the flower, as there are no other objects in the entire image. In the first image, the network is looking for the face of the lion in the correct part of the image, it also highlights the the parts where the the hair of a lion is located. This is probably how the network can tell the difference between a lion and a dhole.

In general, the CAM technique is producing heatmaps of class activation over input images for any different layer in the network. However, in our case we set up the model to use the **Grad-CAM** process, that means we focuse only on the final layer 'block5_conv3' of the model since the authors of Grad-CAM argue : "we can expect the last convolutional layers to have the best compromise between high-level semantics and detailed spatial information."

The code of the experiments is available on my github account: **Github**.