Flowers Recognition

Rémi Ang
Udacity.com - Machine Learning Engineering Nano Degree

June 5, 2018

1 Definition

1.1 Project Overview

Object recognition is a central topic in the field of computer vision. The numerous applications can be applied in many different domains such as self-driving vehicles environment detection, skin cancer screening or reality augmented tourism.

This project focus on the automatic recognition of flower species from pictures. The potential applications are:

- population tracking and preservation
- crop and food supply management
- toxicity detection
- education
- and more...

Figure 1: application example: it's a Daisy!

Flowers recognitions is already used in mobile applications addressing for example to garden lovers or hitch-hikers [4]. Unfortunately neither the technology or performances of the models used in those products are publicly available. However the research work on flower species identification entitled "bi-level co-segmentation method for image classification" from Yuning Chai, Andrew Zimmerman and Victor Lempitsky in 2011 [6], demonstrated a robust method involving the combination of image intensive pre-processing and SVM classification training. This work is used as benchmark for this project as specified in the Chapter 2.3: Benchmark.

1.2 Problem Statement

Is it possible to accurately identify the variety of a flower from a picture ?

The today state of the art solution to tackle pictures classification problems is the creation and training of a deep Convolution Neural Network (CNN). The training has to be performed on a dataset of flower pictures classified by species.

1.3 Dataset

The dataset used in this project is the "Flowers Recognition Dataset" of Kaggle.com given by Alexander Mamaev.

The zip archive is 230MB and contains about 4300 flower images (.jpg) classified in 5 classes: Daisy, Dandelion, Rose, Sunflower and Tulip.

1.4 metrics

In this project, we aim to build a model able to correctly classify flowers from five different classes. The trivial metrics to evaluate how good the model is at predicting the correct flowers classes is the *Accuracy ACC* defined as follows:

$$ACC = \frac{number\ of\ correct\ predictions}{total\ number\ of\ prediction} \cdot 100 \text{ [2]}$$

Moreover, we will use the textitClassification Cross-Entropy $H(y, \hat{y})$ in order to evaluate the model progression during the training and save the weights each time a new minima is reached.

$$H(y, \hat{y}) = -\sum_{k=1} y_i log(\hat{y}_i) \text{ [5]}$$

2 Analysis

2.1 Data Exploration and Visualization

The pictures have been automatically scraped from the online picture portals Flickr, Google Images and Yandex Images.

They represent single or multiple flowers under various angles, zoom, brightness conditions, life cycle stages, etc.

Figure 2: samples pictures

The complete set of pictures distribution is as in Fig. 3:

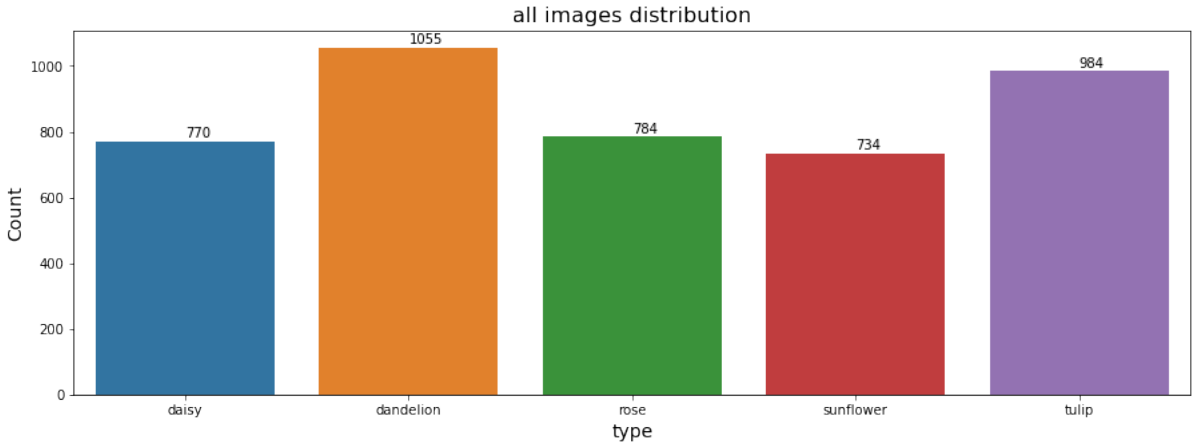

Figure 3: All images distribution by label

We observe that the classes distribution is slightly unbalanced. There is an average of 865 pictures per classes to train the model, which is reasonably enough. The *dandelion* class is the most represented with 1055 pictures. However, this flower variety has the particularity to be represented in 2 radically distinct state: the yellow flowers and the white "blow balls". For this particular case, it is beneficial to have more sample to expect the model to be able to recognize both *dandelion*'s aspects.

Nevertheless, a significant number of pictures contains other subjects (persons, insects, other flowers, vase) or seem to be miss-classified. Such pictures can potentially impact negatively the model learning. A more detailed approach of the samples selection is described in chapter 3.1: Data Preprocessing

2.2 Algorithms and Techniques

2.2.1 pictures selection and split

As already mentioned in the previous chapter, a large number of pictures seems inappropriate to train a model. A "blacklisting" approach will be put in place to discard them from the dataset.

On top of that, the selected pictures will be randomly split into three distinct training, validation and testing datasets. The random splitting is made in a stratified fashion in order to preserve the initial classes distribution in each set.

2.2.2 image augmentation

Image augmentation will be used to enrich the training and validation sets. This technique consist in modifying the original pictures in order to artificially generate more training samples. The modifications considered in this project consist of flipping, zooming, stretching or shrinking the images.

2.2.3 Deep Convolutional Neural Network

A deep convolutional network is a type of neural networks which architecture is designed in so called convolution layers. The principle of forward and back propagation common to neural networks remains. However the main particularity of a convolutional layer is to analyse the picture in small sub-regions and derive patterns from it. The sequencing of convolutional layers result in the identification of patterns increasingly complex. Consequently the patterns identified in the first layers are rather basic (points, stripes, lines,...) and generic to any

pictures. On the contrary, the patterns identified by the last layers are more complex (objects, face attributes,...) and are specific to the problem addressed.

2.2.4 Transfer Learning

The training of a full CNN requires significant computation resources and time. However, with Transfer Learning, it is possible to use pre-trained models. This techniques presents the double advantage to spare a significant amount of training time and to benefit from the quality of very deep architectures.

Transfer learning consist of loading a pre-trained model, fix its weights, and replace the last layers in order to fit the new given problem. In this project, we will determine which among three models available in `Keras` module suits the best for flower classification. Those models have been trained on the *ImageNet* dataset consisting of several million of labelled images representing thousands of objects.

2.3 Benchmark

The benchmark for this project is the "Flowers Species Recognition" work from Yuning Chai, Victor Lempitsky and Andrew Zisserman from the University of Oxford in 2011 [6, 7] .

This two-steps approach consist of the segmentation of the pictures and the training of a kernelized SVM classifier. The best model reach a prediction accuracy of 80.0% and is able to recognize flowers among 102 different variety.

The goal of this project is to reach at least this accuracy, however only 5 flower species are considered.

3 Methodology

3.1 Data Preprocessing

3.1.1 Sample blacklist

As already mentioned in 2.1: Data Exploration and Visualization, a significant amount of pictures in the dataset are likely to negatively impact the learning of the mode. Our flower recognition model is intended to be used in an application where the images are supposed be focused on one or few flowers of the same type, from a close to middle-range distance (1 to 10 meters), in natural colours.

The black-listing of pictures aim to discard samples:

- which don't have a ".jpg" extension (some python web scrapping routine can be found in the *dandelion* subfolder)
- not representing a flower
- wrongly classified
- containing other subjects (persons, insects, objects,...)
- close ups
- wide shots with fields, mountains, landscapes...
- drawings
- with artistic filters significantly denaturing the original colour

After the manual visualization of all samples, the blacklist contains 1059 items (24.5% of the total dataset) stored in a text file (`blacklist.txt`). The data loading implementation read this file and discards any file found in the list.

Figure 4: samples of black listed pictures

After filtering, we observe that the new classes distribution is comparable to the original one. The classes are ow represented by 654 samples on average:

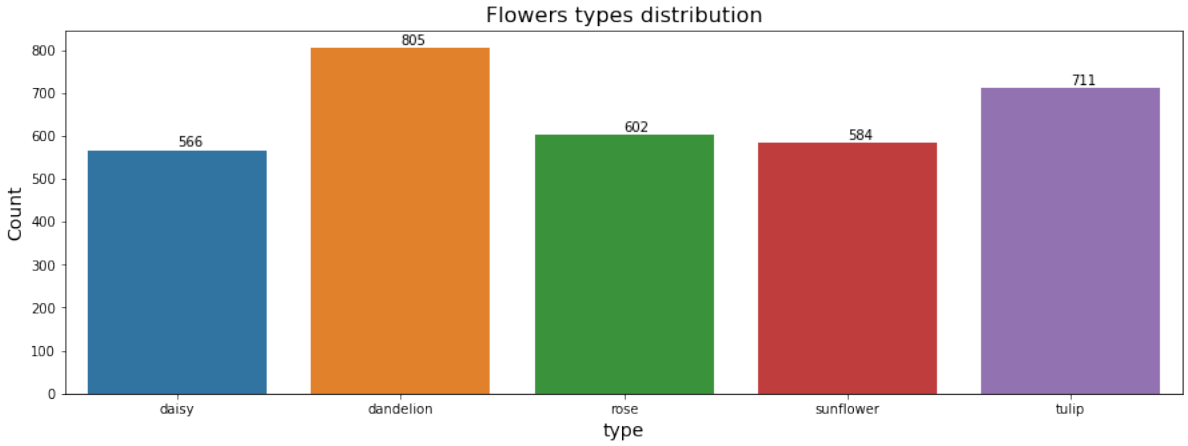

Figure 5: Images distribution after blacklist filtering

3.1.2 Training, validation and testing split

The dataset is randomly split in 3 parts using the `train_test_split` from `sklearn.model_selection` package, using the `stratify` option in order to preserve the original classes distribution in each split:

- Training (76.5% - 2500 samples)
- Validation (8.5% - 278 samples)
- Testing (15% - 491 samples)

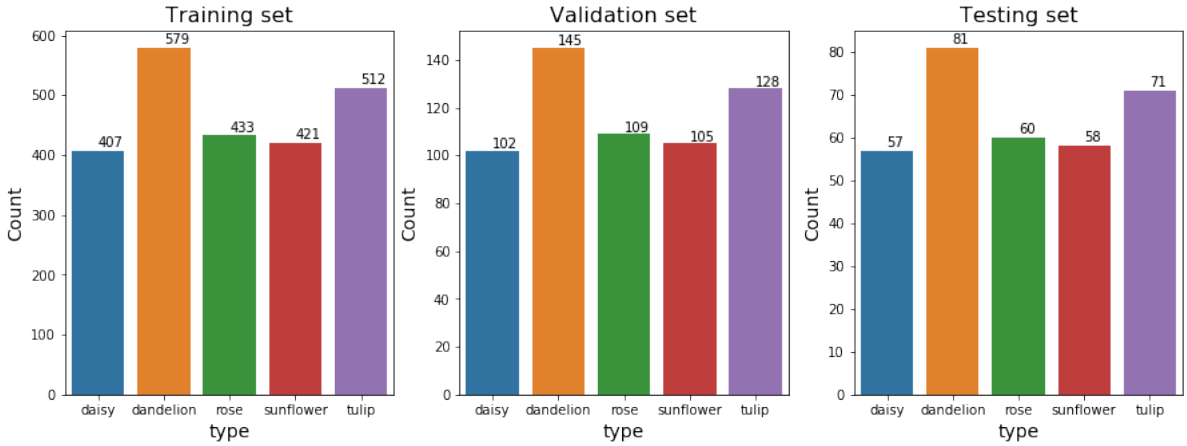

Figure 6: Stratified images distributions after train/valid/test split

3.1.3 Dataset loading in memory

The images are loaded in RBG colors and resized in a 299x299 shape using OpenCV "`cv2`" python package. Targets labels are one-hot-encoded using the `to_categorical` method from the `keras.utils.np_utils` package.

3.1.4 Image augmentation

In order to enrich the dataset, image augmentation is applied to the training and validation datasets using the `ImageDataGenerator` utility from the `keras.preprocessing.image` package. This step is also used to normalize the features: all values are divided by 255. No image augmentation is applied on the testing set which is meant to represent real case images for the model evaluation.

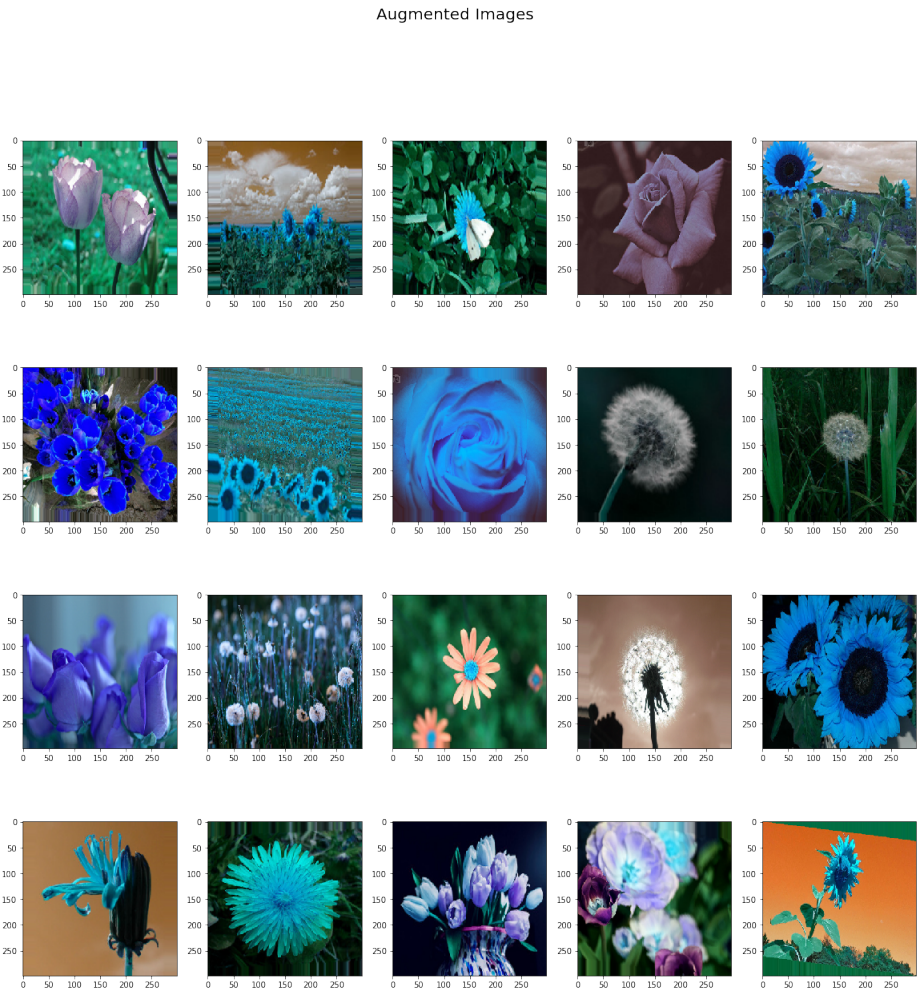

Figure 7: samples of augmented training images

3.2 Model Implementation

3.2.1 CNN architecture

The targeted CNN architecture is the combination of a fixed weights base model and some custom top layers specialized for the prediction of flower varieties. Hyper-parameters such as learning and decay rates have been determined experimentally and tested in order to find th best learning time/accuracy compromise.

Model design:
The overall model design procedure consists of the following steps:

1. load the weights of a base model trained on ImageNet dataset, without the top layers, setting the input shape to (299,299,3).
2. freeze the base model weights
3. Add top layers
4. compile the model with loss function, optimizer and metrics set-up

Top layers:
The top layers consist of the following:

- a flatten layer connected to the last convolution layer of the base model (usually a MaxPooling layer)

- a fully connected layer with ReLu activation. A number of 500 units has been arbitrarily chosen.
- a dropout layer. A dropout factor of 0.3 has been determined experimentally.
- an output layer with 5 units (1 for each flower class) and a *softmax* activation function.

Hyper-parameters:

- Optimizer: Adam with learning rate $\alpha = 1e^{-}3$ and decay $d = 1e^{-}6$
- loss function: categorical cross-entropy
- metrics : accuracy

This architecture is inspired by the solution proposed by A. Akashnian in his "Flowers are mesmerizing" notebook published on Kaggle.com [3].

3.2.2 CNN training

The model is trained with the augmented pictures generator specified in 3.1.4: Image augmentation, on 50 epochs with batches of 32 samples (= 73 steps/epoch). The model weights are saved via a `ModelCheckpoint` each time that the validation loss each a new minima. The training historic is stored in order to later analyse the accuracies and losses evolution during the training. In order to speed-up the processing time, the model has been trained on an Amazon Web Services GPU instance (50 to 60 seconds per epochs).

3.3 Model Refinement

3.3.1 Base Model selection

The `keras.applications` package offers some pre-trained models. In frame of this project, the `VGG16`, `VGG19` and `Resnet50` based models have been individually tested on 10 epochs.

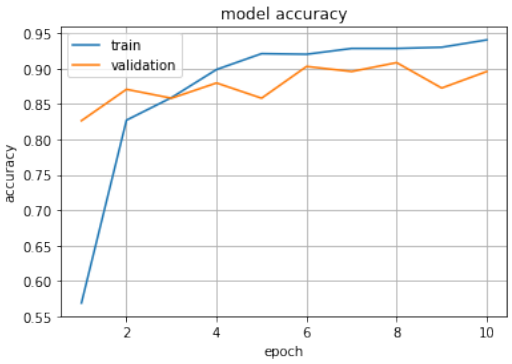

Figure 8: VGG16 - 10 epochs Accuracy

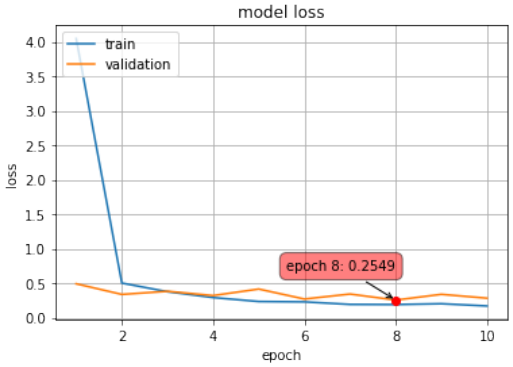

Figure 9: VGG16 - 10 epochs Loss

According to Fig. 8 and 9, VGG16 based model seems to learn extremely fast after the first back-propagation at epoch 1. It then constantly improves during the next epochs. After 10 epochs, the best validation loss achieved is 0.2549 and the accuracy 0.91 at epoch 8. This base model seems to be a good candidate for our flowers classifier.

— VGG19

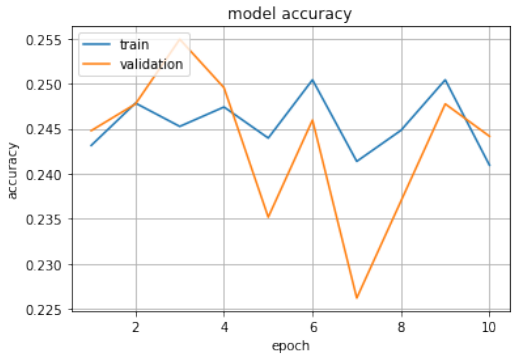

Figure 10: VGG19 - 10 epochs Accuracy

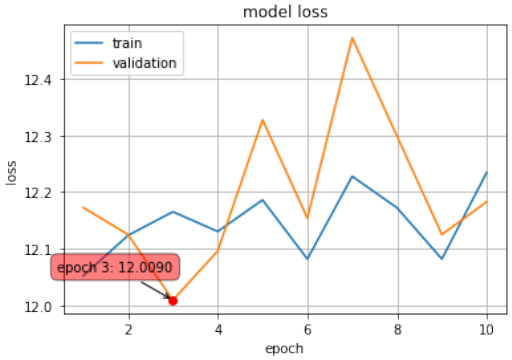

Figure 11: VGG19 - 10 epochs Loss

According to Fig. 10 and 11, the VGG19 based model doesn't seem able to learn. The average training accuracy of 24% is comparable to random guesses performance. This base model will not be considered for the flower classifier.

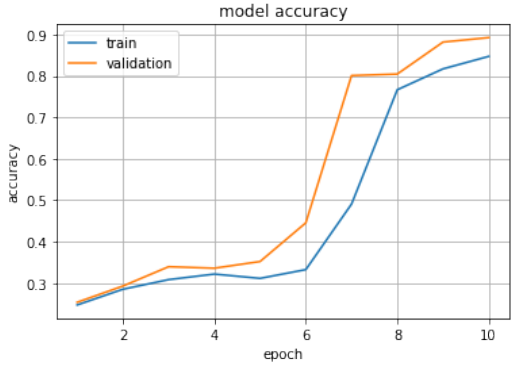

Figure 12: Resnet50 - 10 epochs Accuracy

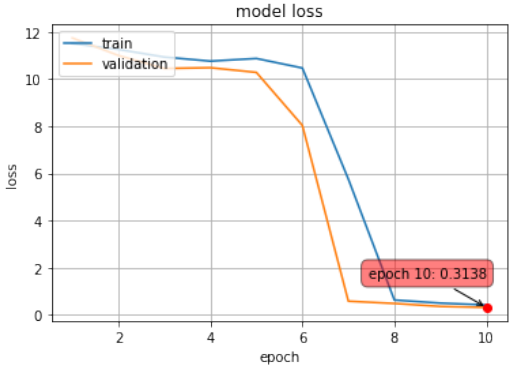

Figure 13: Resnet50 - 10 epochs Loss

According to Fig. 12 and 13, it takes up to 7 epochs for this model to achieve reasonably good performances. It reach its best validation loss at the 10th epoch with 0.3138 and an accuracy of nearly 90%, letting suggest that it could even more improve with longer training.

base model selection - conclusion:

The best performing flower classifier on 10 epochs is the **VGG16 based model**. Therefore, it has been selected to perform a full training on 50 epochs.

4 Results

The VGG16 based flower classifier is trained on 50 epochs (about 45 minutes on GPU instance). According to Fig. 14 and 15, the model achieve the best validation loss at epoch 33 with 0.2122. The validation accuracy stagnate at 90% after the 10th epoch.

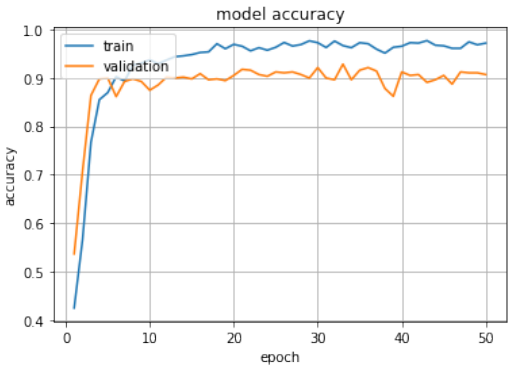

Figure 14: VGG16 - 50 epochs Accuracy

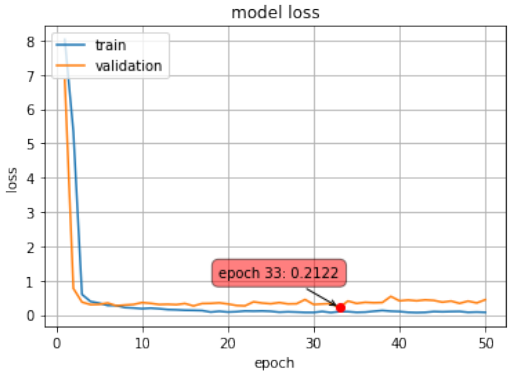

Figure 15: VGG16 - 50 epochs Loss

4.1 Model Accuracy

The model accuracy is compute by loading the best loss weights, scaling the test set and evaluating the model with its `evaluate` method on the scaled test dataset:
The VGG16 based flower classifier achieve an accuracy of **89.30%** on the testing set.

4.2 Justification

The predictions are evaluated for each sample of the test dataset:
The confusion matrix displayed in Fig. 16 confirms the good performance of the model.

- 80.7 % of the test *daisies* are correctly classified. The type of flower is mostly confused by the model by *dandelions*.
- 92.6 % of the test *dandelions* are correctly classified. This type of flower is mostly confused by the model with *sunflowers*. This remarquable score demonstrate that the model is able to identify the *dandelions* both in yellows flower state and white "blow-balls" (see discussion in Chapter 2.1: Data Exploration and Visualization).
- 85.0 % of the test *roses* are correctly classified. This type of flower is mostly confused by the model with *tulips*.
- 94.8 % of the test *sunflowers* are correctly classified. This the most accurately predicted type of flower.
- 91.5 % of the test *tulips* are correctly classified. This type of flower is mostly confused by the model with *roses*.

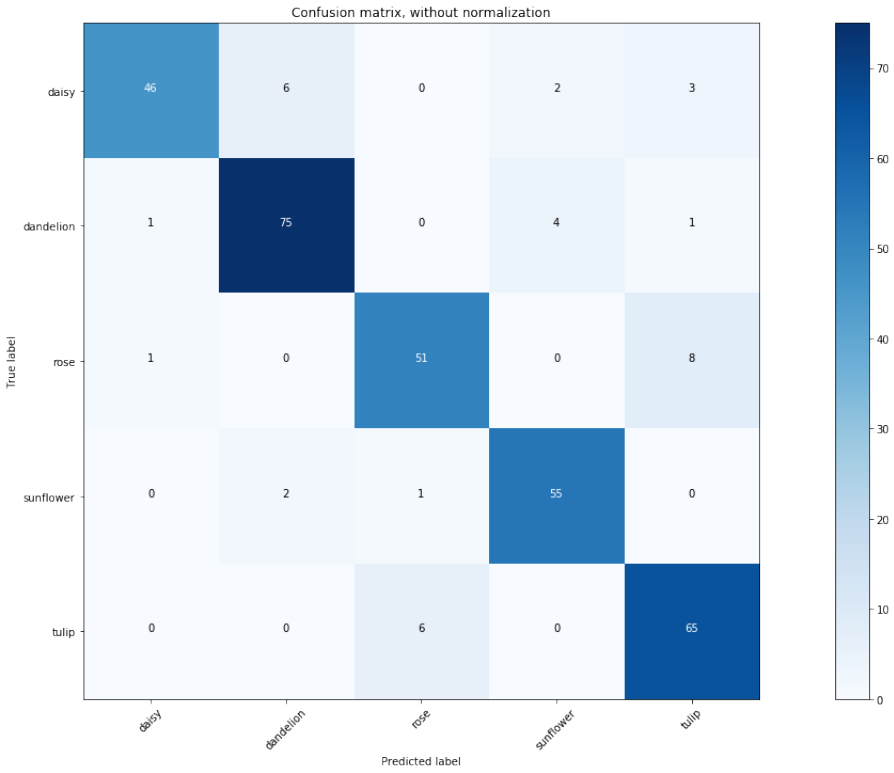

Figure 16: test pictures predictions - confusion matrix

The expectation to reach at least 80.0% accuracy formulated in Chapter 2.3: Benchmark is met. Because the test pictures taken to evaluate the accuracy have never been seen by the model before, this good result let us suggest that the model is able to well generalize. The test samples displayed on Fig. 17 demonstrate that the model is able to correctly identify:

- single flowers on the picture
- multiple (two to several dozen) flowers on the same picture
- flowers taken from different angles
- same flower type in different life states (i.e. yellow flower and "blowball" dandelion)

5 Conclusion

5.1 Free-Form Visualization

5.1.1 Test samples prediction

A sample of successful tests predictions is displayed in Fig. 17 with the predicted label and the true label shown in the title.

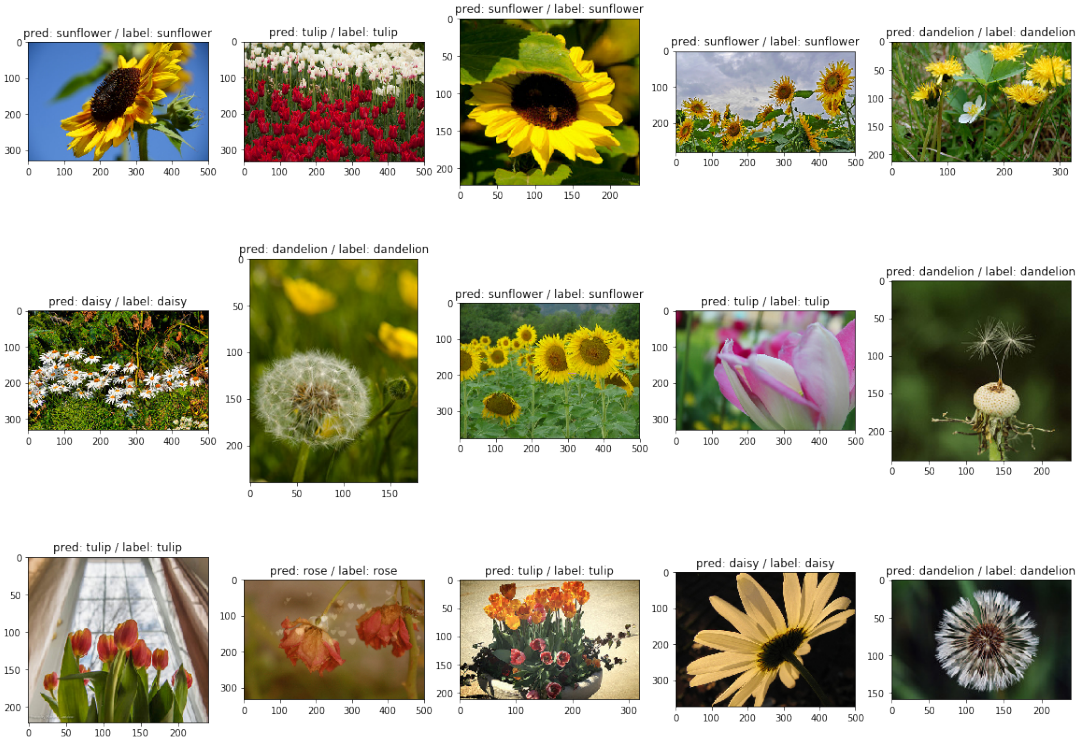

Figure 17: Sample of test picture flower recognition

5.1.2 Failed prediction

All the test samples that the model failed to identify correctly are display on Fig. 18.

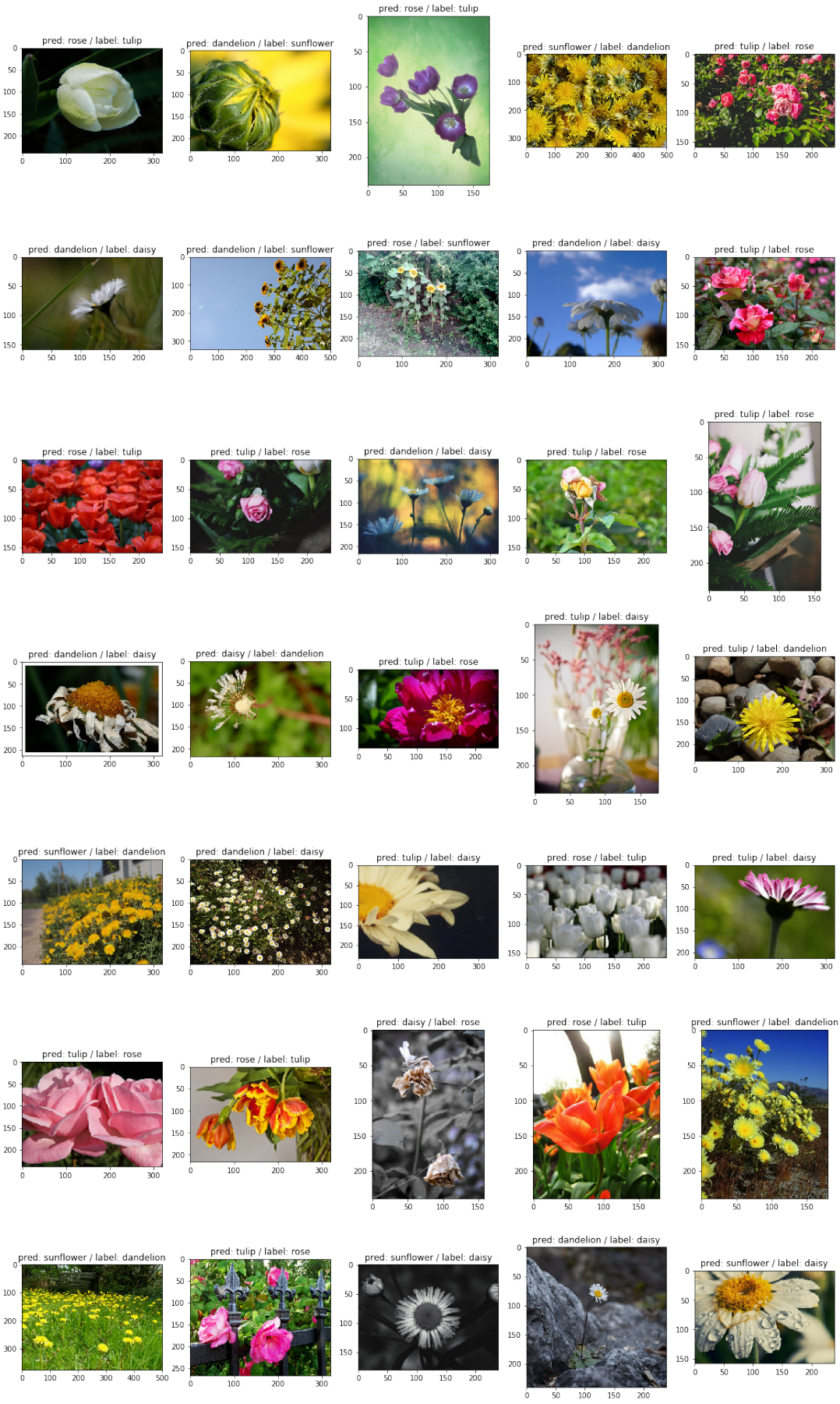

Figure 18: failed predictions in the test dataset

5.2 Reflection

The process used for this project can be summarized as follow:

1. Find a relevant/challenging problem to solve
2. find already existing work(s) in order to set the benchmark
3. Find a dataset suitable for answering this problem
4. Clean up and pre-process the dataset to make it training-ready
5. Test different base model for the CNN and select the best performing one
6. Train the final chosen CNN in the cloud
7. Evaluate the model and analyse the results

One particular difficult and tedious task has been the manual selection of samples to keep or remove from the dataset. This work is hardly automatable and had to be performed case by case, based on arbitrary rules and intuition.

I have been positively surprised by the differences in performance between the three based models tested. According to the references, VGG16 is the model among the three having the lowest accuracy on the ImageNet validation dataset [1]. However, this exercise comforted me in necessity to try different approaches.

5.3 Improvement

The main improvement for this project would be to increase the amount of flower classes. It would be particularly interesting to train the same CNN on the 100+ flowers types used by the BiCos-SVM model [7]. It would also worth trying other base models that have not been considered in this study. It would be beneficial to deeper analyse prediction failure cases in order to identify failures root cause and adapt accordingly the dataset or the model if feasible. Finally, in order to deploy the model in real life, it would be necessary to develop a mobile application able to directly process a picture taken by the device and return to the user a prediction.

References

[1] Keras, applications. `https://keras.io/applications/`.

[2] Scikit-learn, model evaluation, accuracy score. `http://scikit-learn.org/stable/modules/model_evaluation.html#accuracy-score`.

[3] A. Akashnain. Flowers are mesmerizing. `https://www.kaggle.com/aakashnain/flowers-are-mesmerizing`.

[4] Miranda Evans. The best apps to identify unknown plants and flowers. *The Telegraph*, 26.06.2015. `https://www.telegraph.co.uk/gardening/tools-and-accessories/the-best-apps-to-identify-unknown-plants-and-flowers/`.

[5] Rob Di Pietro. A friendly introduction to cross entropy loss. `https://rdipietro.github.io/friendly-intro-to-cross-entropy-loss/`, May 2, 2016.

[6] Andrew ZISSERMAN Yuning CHAI, Victor LEMPITSKY. Bicos: A bi-level co-segmentation method for image classification. *ICCV*, 2011.

[7] Andrew ZISSERMAN Yuning CHAI, Victor LEMPITSKY. Flowers species recognition demo. `http://www.robots.ox.ac.uk/%7Evgg/research/flowers_demo/`, 2011.