Flower recognition

Empirical Research 2019/2020

Andrea Rafanelli
4815168

Agenda

1. DATASET AND DATA PREPROCESSING
2. MODEL CREATION
3. MODEL IMPROVEMENTS
4. WHAT ELSE?
5. CONCLUSION

Section 1

DATASET AND DATA PREPROCESSING

Dataset

GOAL:
Creation of a model able to classifies flower images into the five categories.

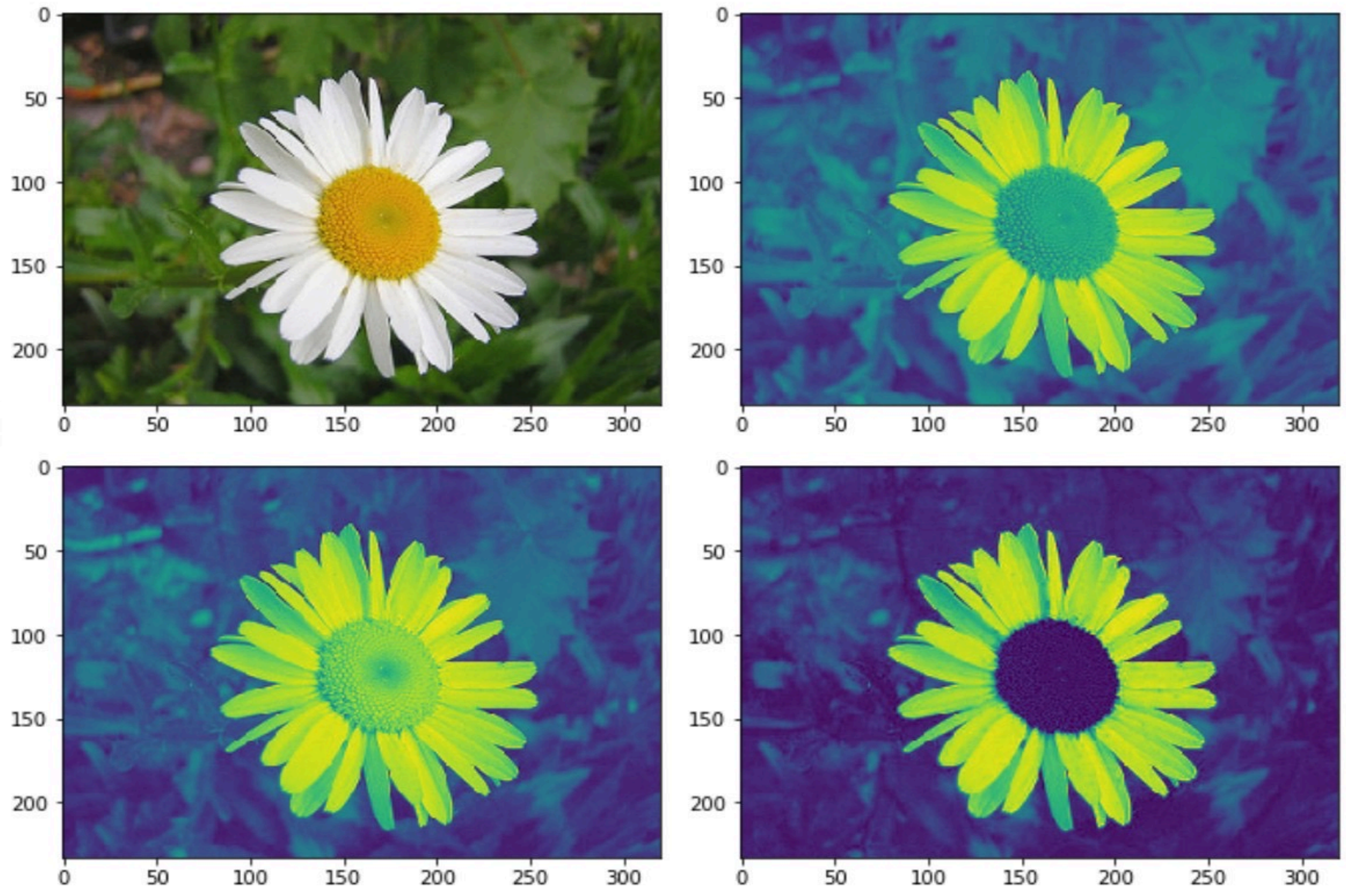
1. 4323 images
2. Five categories:
   dandelion, rose, sunflower, daisy and tulip.
3. 240x320 pixels

Data preprocessing (1)

THREE FILTERS: RGB

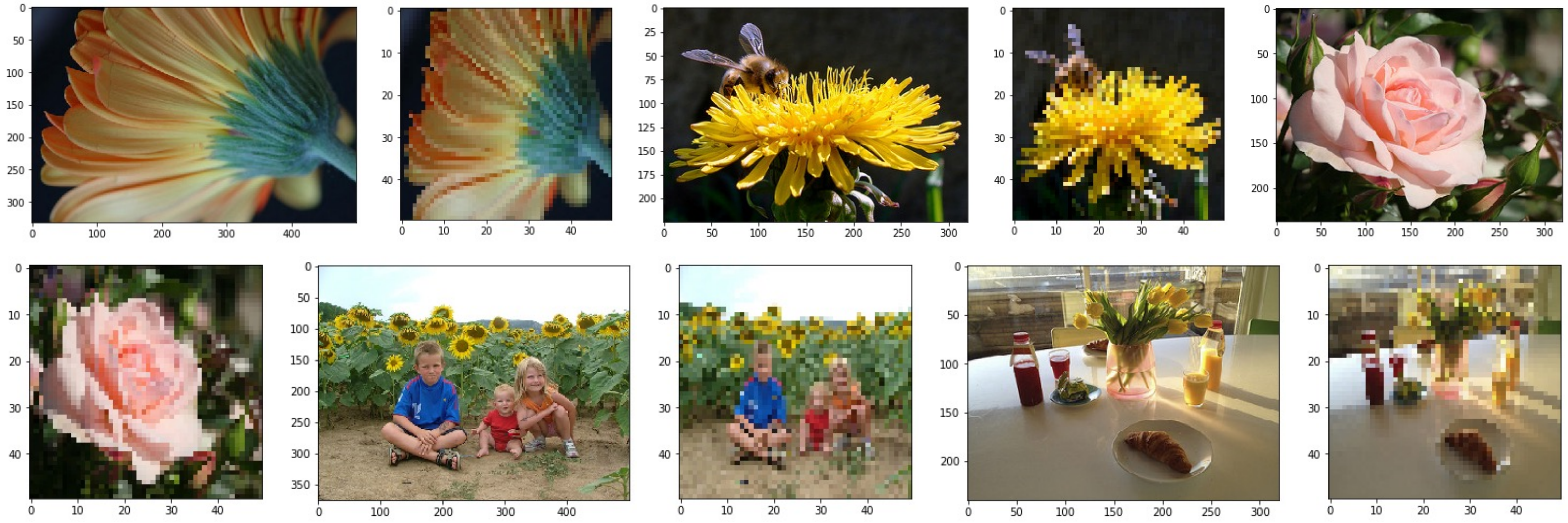
```
channels = ['r','g','b']

def plot_rgb (image):
    for i, color in enumerate(channels):
        plt.imshow(image[:,:,i])
        plt.show()
```

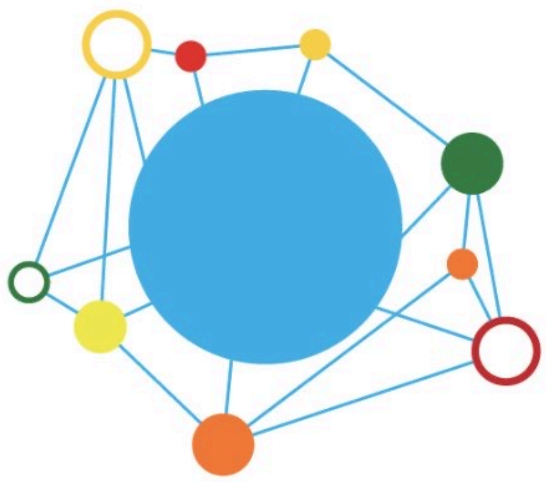


Data preprocessing (2)

FROM 240x320 TO 90x90
NORMALIZATION OF THE PIXELS

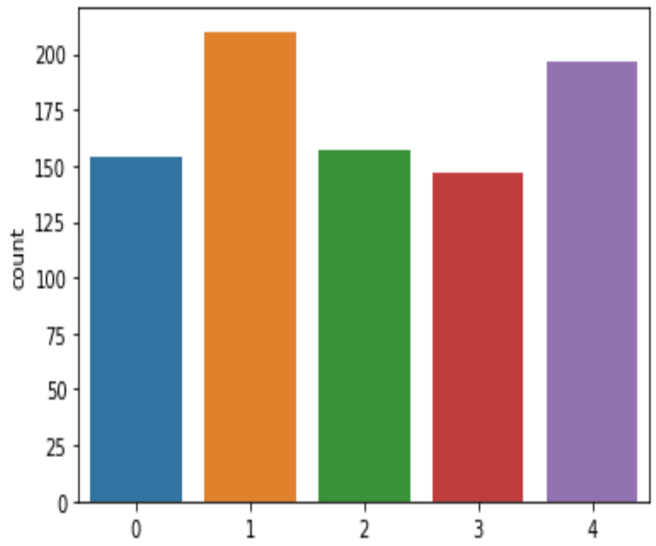


Data preprocessing (3)



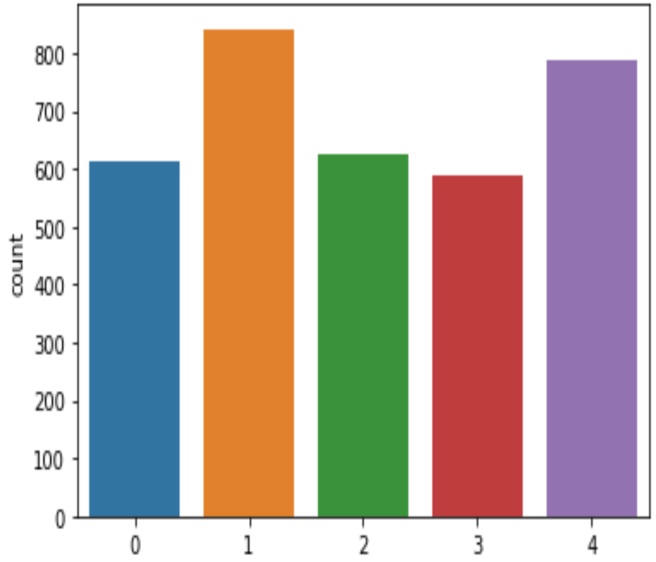
- 3458 images in the training set
- 865 images in the test set

Are the two balanced?



TEST SET
- 18% daisy
- 24% dandelion
- 18% rose
- 17% sunflower
- 23% tulip

TRAINING SET
- 18% daisy
- 24% dandelion
- 18% rose
- 17% sunflower
- 23% tulip



Section 2

MODEL CREATION

Starting with the simplest one

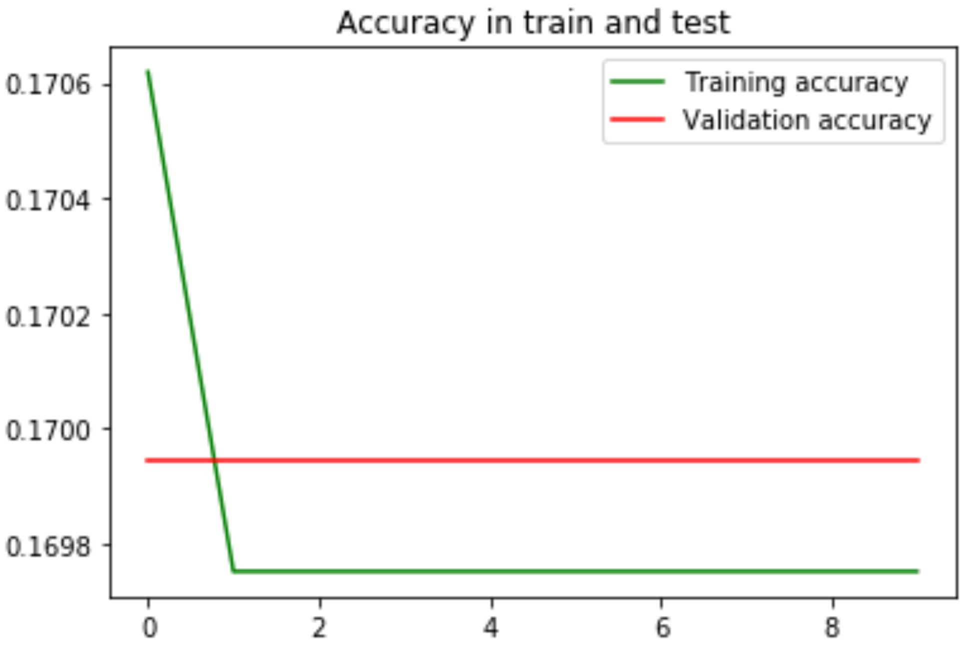
- *Hidden layer activation*: **Relu**
- *Weight initialization*: '**He initialization**'
  Because of the non-linearity of the Relu activation function.
- *Output layer activation*: **Soft Max**
- *Loss:* **Categorical cross entropy**
- *32 as batch size and 10 epochs*
- *3 convolutional layers*
- *Max pooling: extraction of the most important features.*

Siddharth Krishna Kumar proves mathematically that for the ReLU activation function, the best weight initialization strategy is to initialize the weights randomly but with this variance:

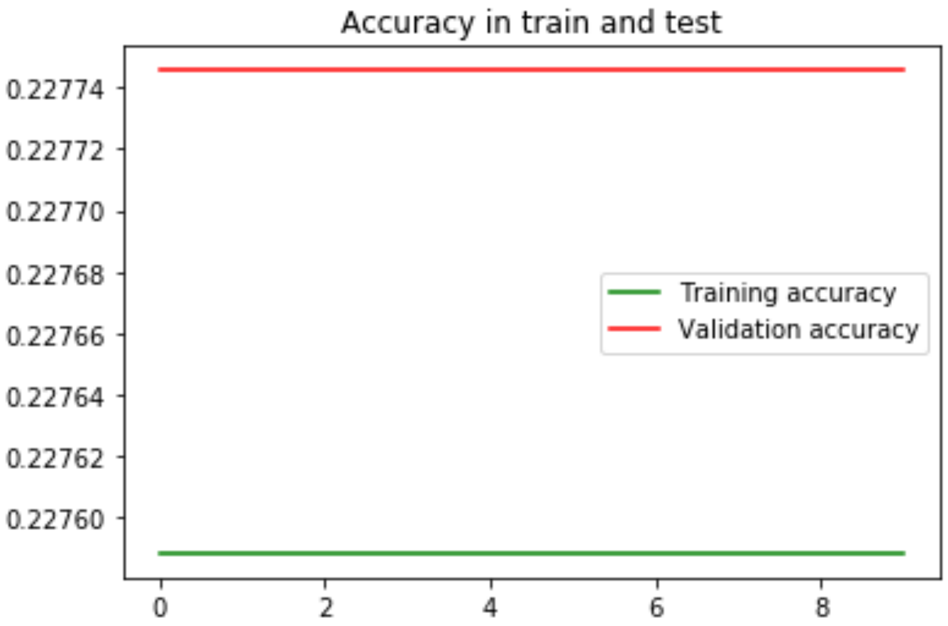
v2=2/N

This is exactly the He initialization.

The ignorance of my model



ADAM



SGD

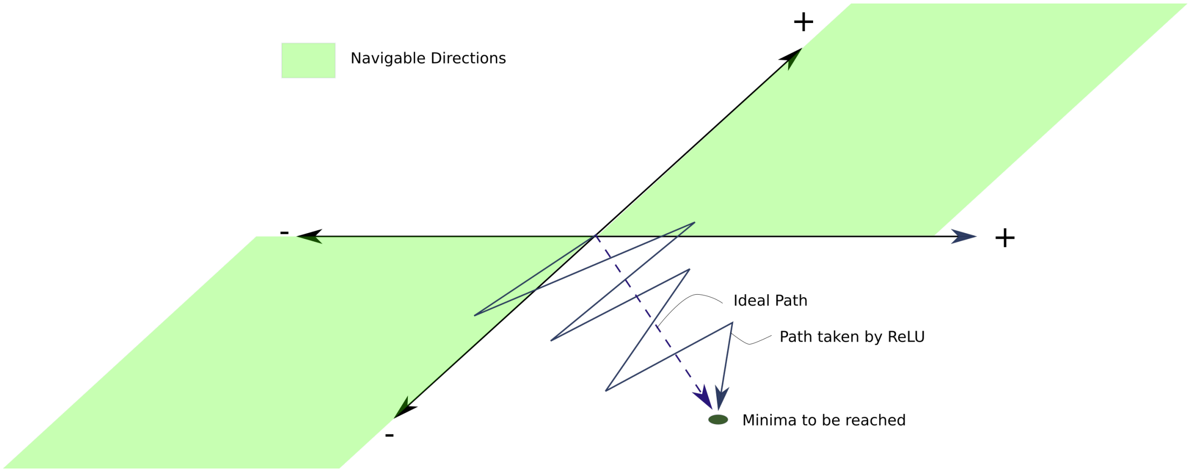
Why my model does not learn?

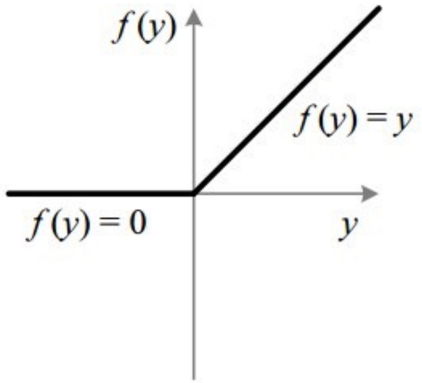
From Relu to Leaky Relu

1. **Relu** is linear for all positive values, and zero for all negative values.

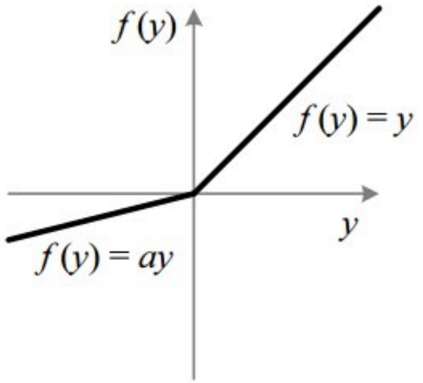
2. PROBLEM: **“dying Relu”**
The weights and the bias causing the negative preactivations cannot be updated.

3. SOLUTION: **Leaky Relu**
The backward pass is able to alter weights which produce a negative preactivation as the gradient of the activation.





Y= max (0; x)



Y= max (0.01x; x)

“Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification”, Kaiming He et al. (2015)

Which algorithm to use? (1)

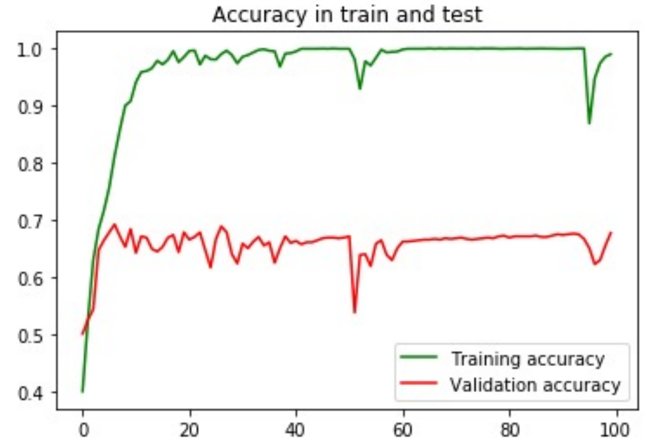
Adaptive Moment Estimation (Adam)

It chooses a direction that would lower the error rate, and continue iterating until the objective function converges to the minimum.

Learning rate:

- 0.01
- 0.001
- 0.0001
- 0.00001

L.r: 0.001
98% accuracy in the training set vs 68%.



L.r: 0.0001
99% accuracy in the training set vs 70%.

Which algorithm to use? (2)

Stochastic Gradient Descent (SGD)

It does not perform computation on the whole dataset but only on a small subset or random selection of data examples.

Learning rate:

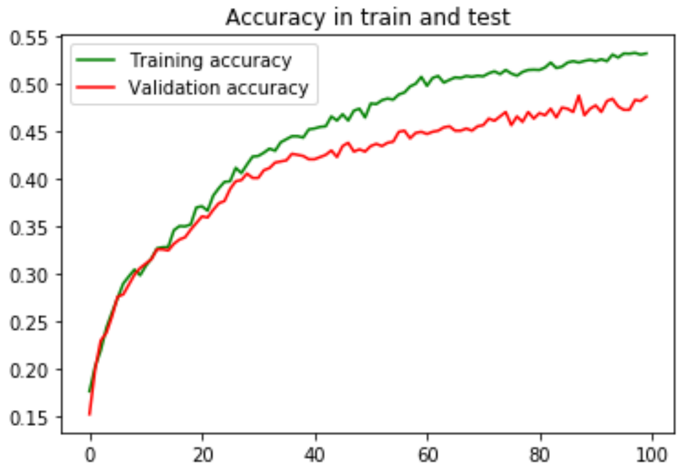
- 0.01
- 0.001
- 0.0001
- 0.00001

Momentum:

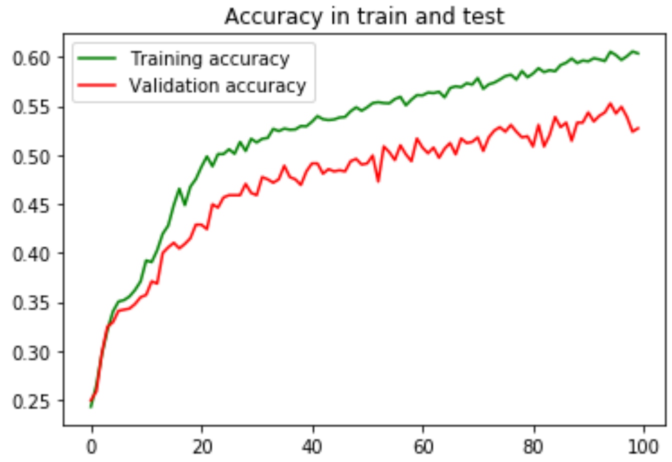
To improve the stability, the convergence and the speed of the training.

- 0.5
- 0.9

L.r: 0.001
Momentum:0.5
53% accuracy in the training set vs 48%.



L.r: 0.0001
Momentum: 0.9
60% accuracy in the training set vs 52%.



Empirical research 2019/2020

Which algorithm to use? (3)

ADAM
Best model:
Learning rate 0.0001

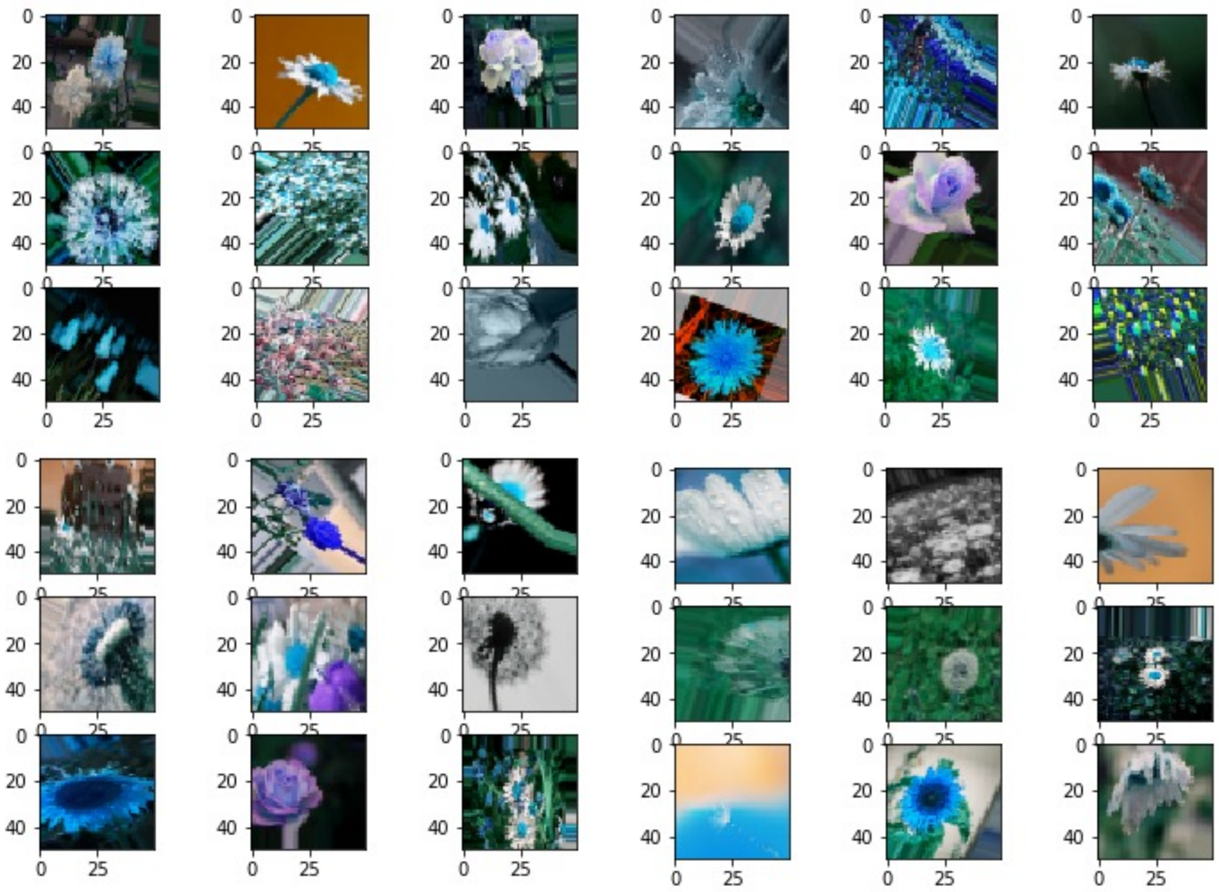
SGD
Best model:
Learning rate 0.0001
Momentum 0.9

SGD has less overfitting, thus i decide to use it!

Section 3

MODEL IMPROVEMENTS

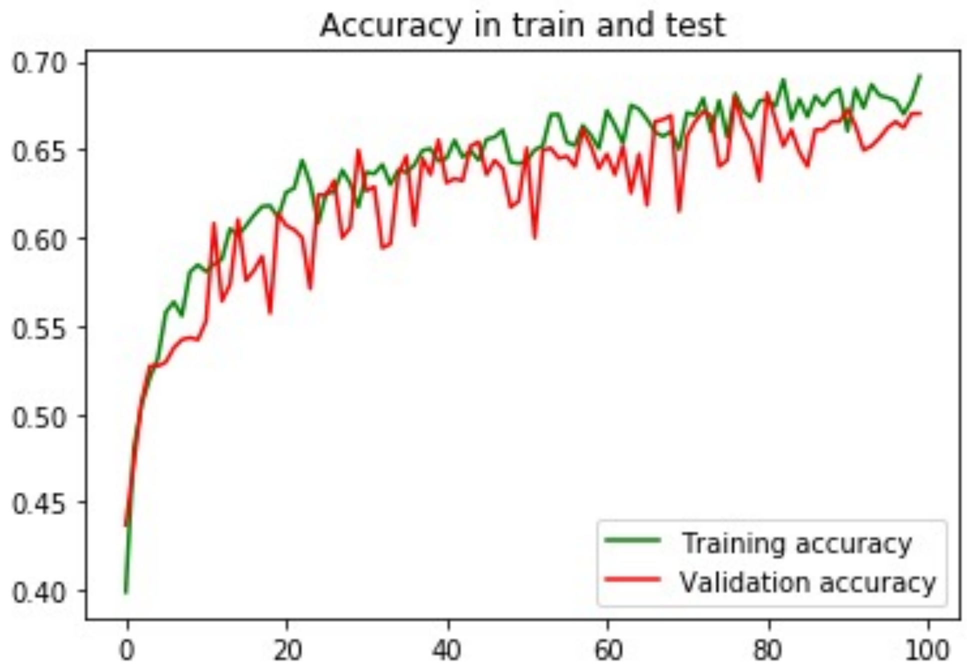
Avoiding overfitting: data augmentation (1)

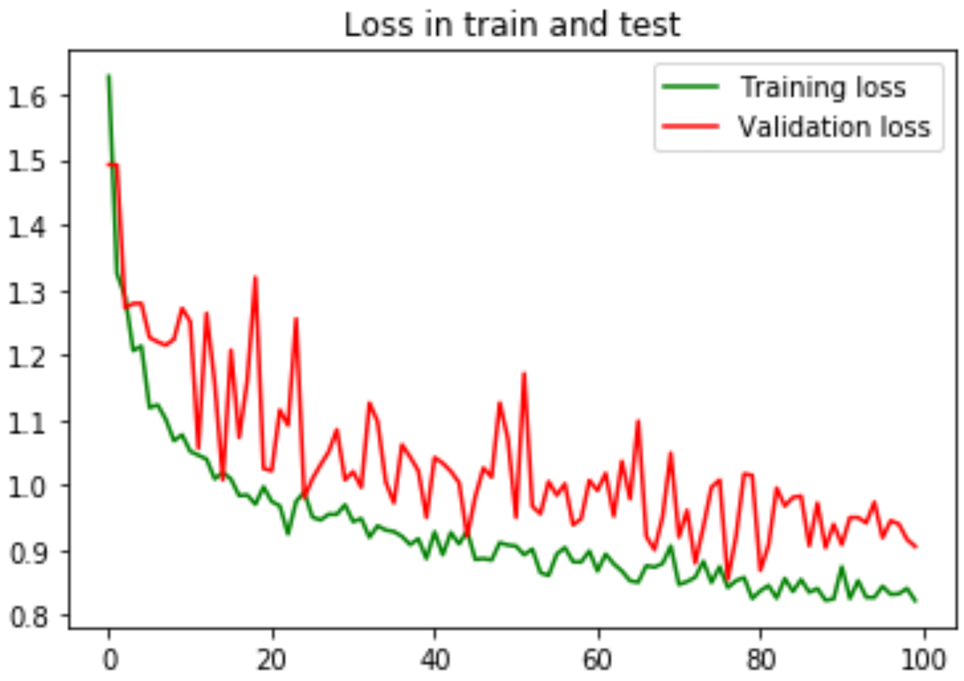


"*Imagenet classification with deep convolutional neural networks" (2012), Krizhevsky Alex, Ilya Sutskever and Geoffrey E. Hinton.*

- Flipping the image horizontally
- Rotation
  Rotation of the image of 40°.
- Width shift
  Shifting the image horizontally by 20%.
- Height shift
  Shifting the image vertically by 20%.
- Zoom
  Zooming the image of 30 %.

Avoiding overfitting: data augmentation (2)





69% accuracy in training set and 67% in test set
No more overfitting but the learning is unstable!

Increasing the stability: Batch normalization (1)

Not a large improvement:

69% accuracy in training set and 68% in test set.

- Improvement of stability and performance
- It normalizes the output of the previous activation layer by subtracting the batch mean and dividing by the batch standard deviation.
- After each convolutional layer

"Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift" (2015), Ioffe Sergey, Szegedy Christian

Final model

SUMMARY:

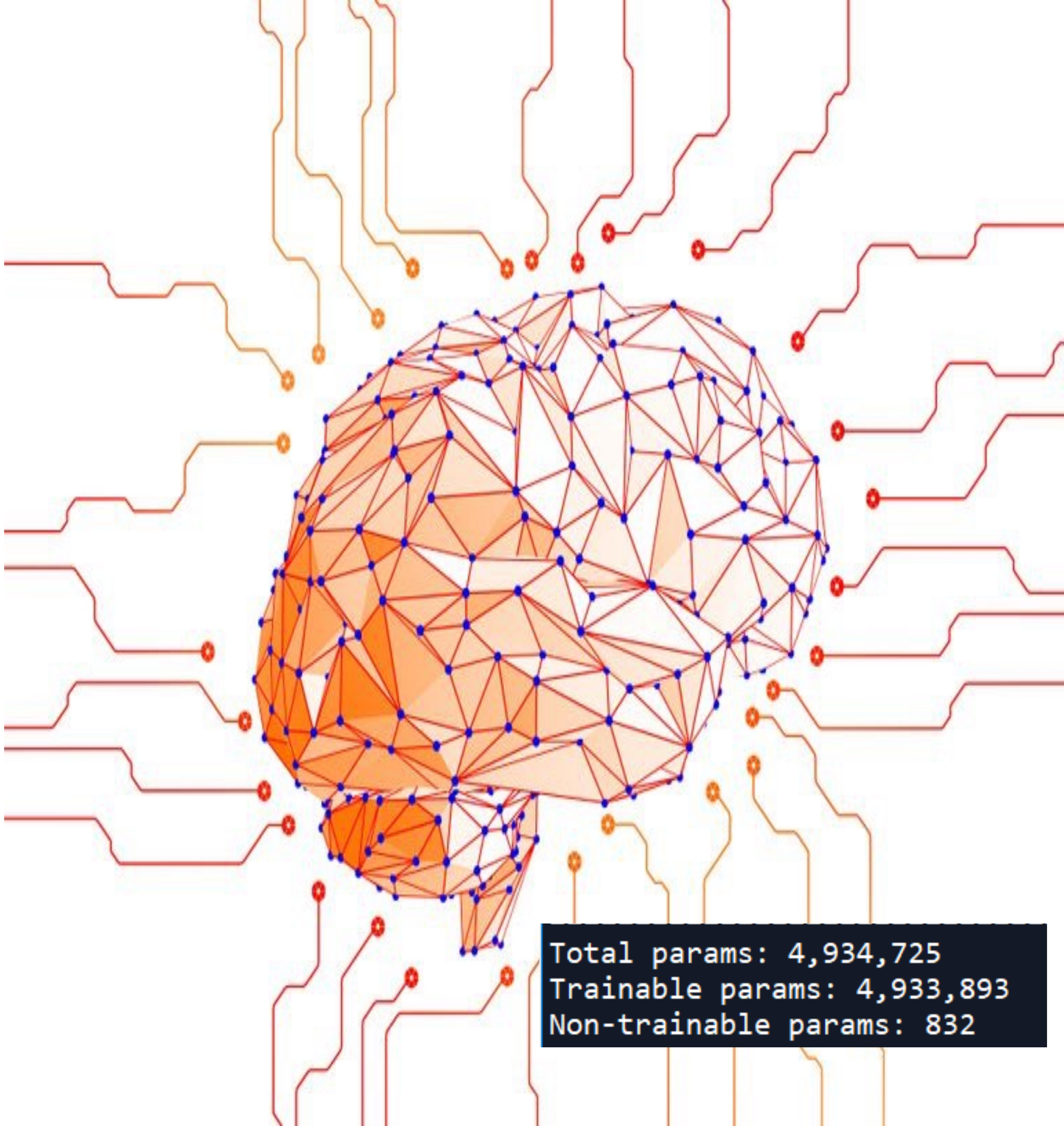
- Optimizer*: SGD* with 0.0001 as learning rate and 0.9 as momentum;

- Activation functions:
  - *Leaky Relu* in the hidden layers
  - *Soft Max* in the output layers

- *Categorical cross-entropy* as loss function;

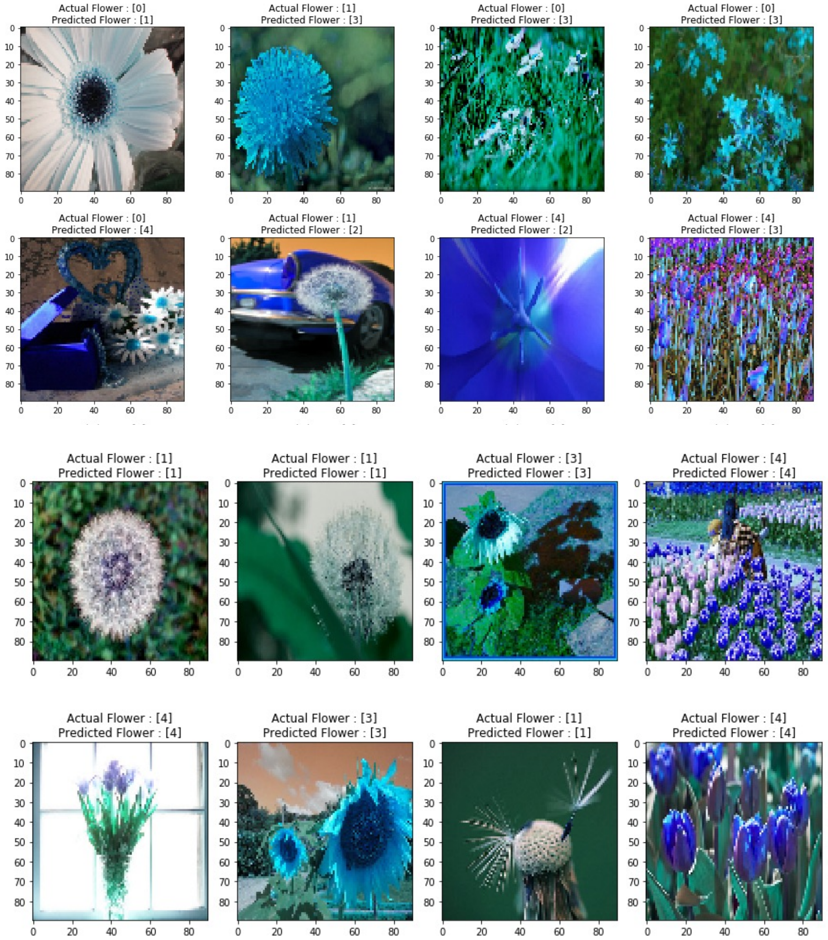
- 32 as batch size and 100 epochs;

- *Data augmentation* + *Batch normalization;*



Prediction

580 correct classification (67%)
285 uncorrect (33%).



Empirical research 2019/2020

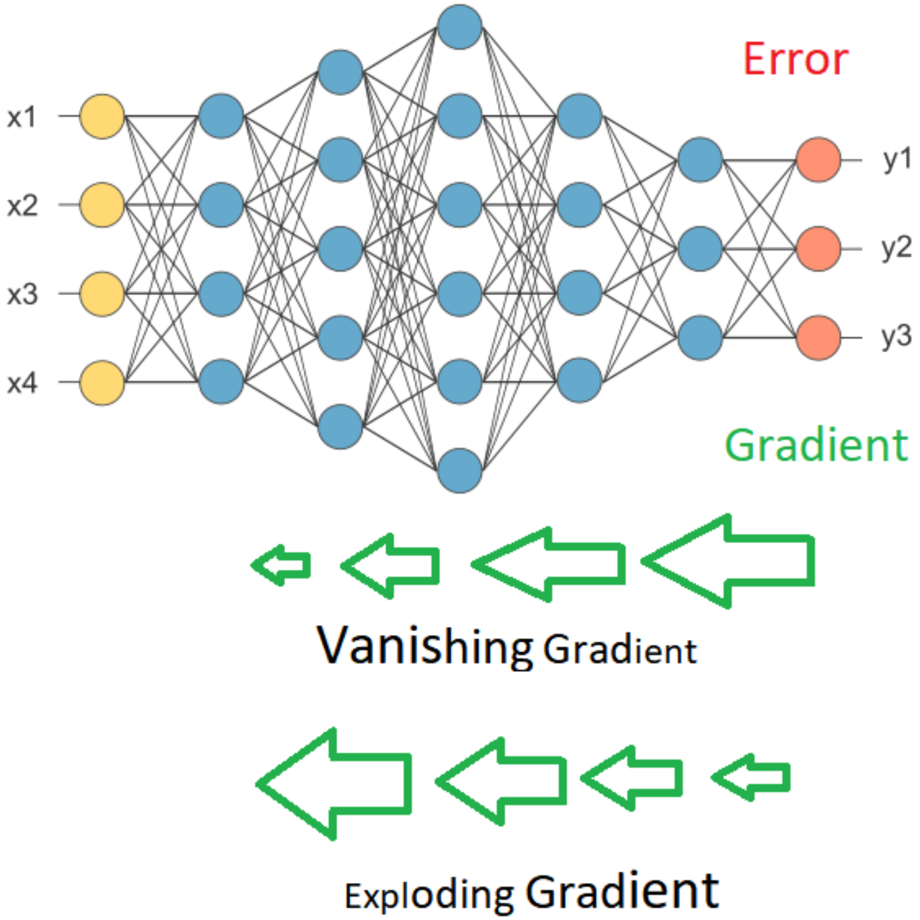
Section 4

WHAT ELSE?


WE NEED TO GO
DEEPER

Adding complexity

Increasing the depth to increase the "levels" of features: higher accuracy



MAIN PROBLEM:
Vanishing/exploding gradient

HOW TO SOLVE IT?

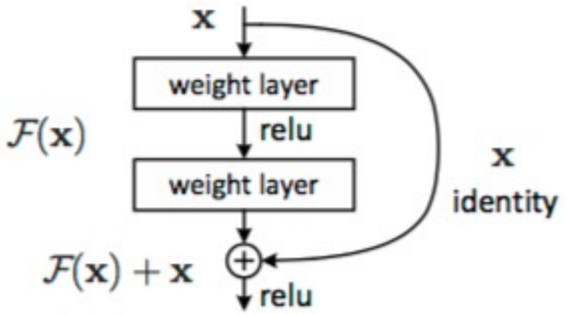
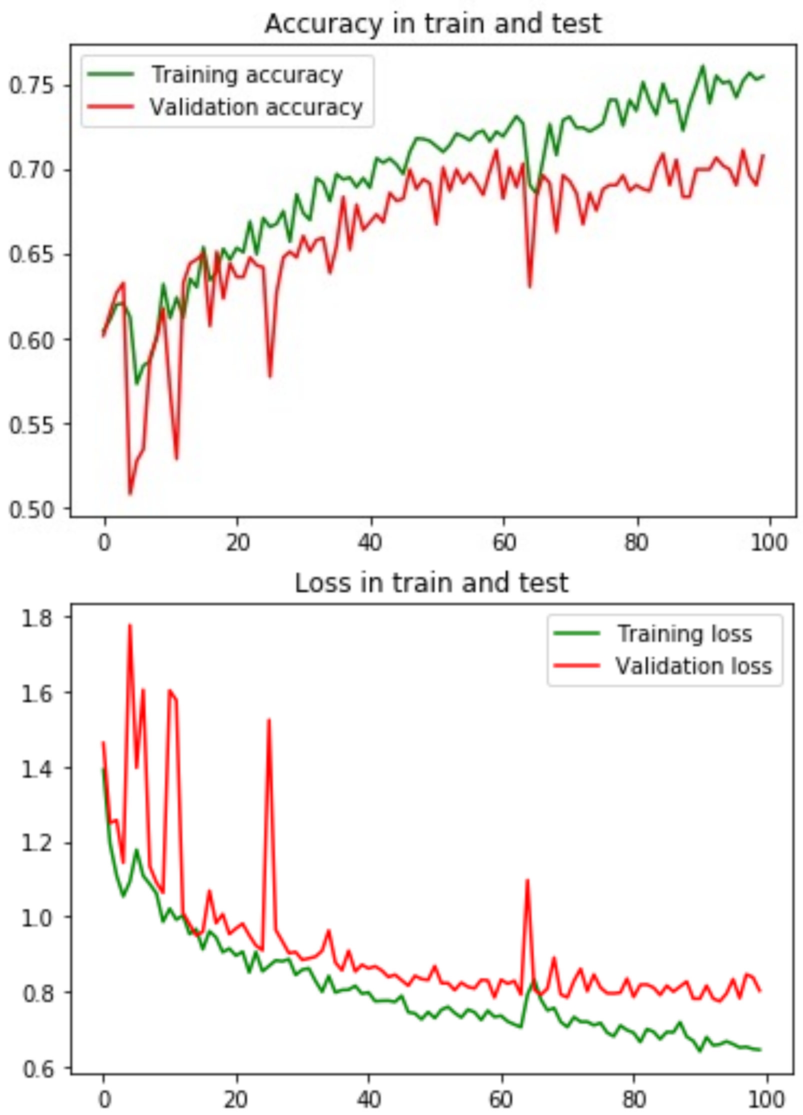


Figure 2. Residual learning: a building block.

ResNet-50



- It is trained on more than a million images from the ImageNet database.
- He initializer, batch normalization, no dropout.
- Mitigation of vanishing gradient problem because of skip connections.

  Skip connections are extra connections between nodes in different layers of a neural network that skip one or more layers.
- It reuses activations from a previous layer until the adjacent layer learns its weights.

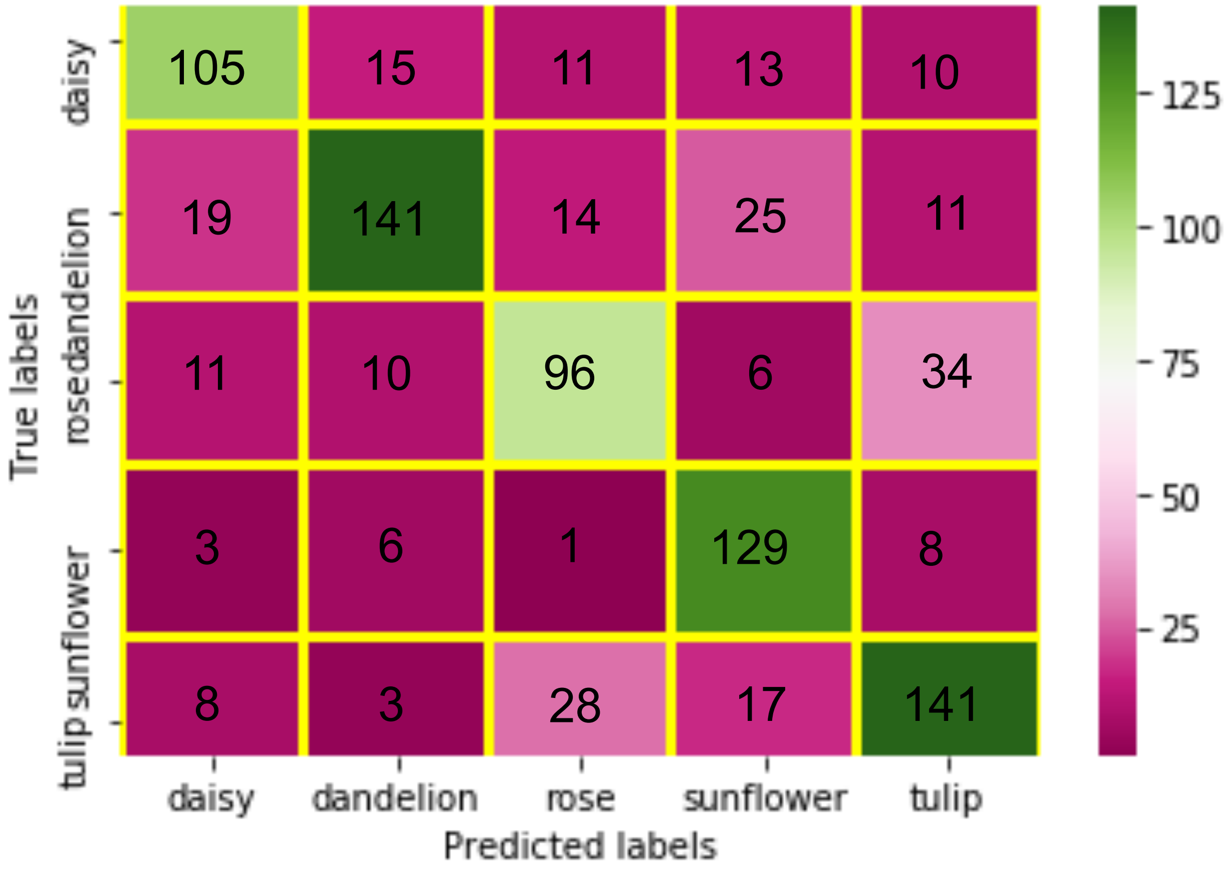
75% accuracy in training set and 69% in test set.

«Deep Residual Learning for Image Recognition», 2015, Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun

Prediction

620 correct classification (72%)
245 uncorrect (28%).

```
In [67]: RES50.summary()
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
resnet50 (Model)             (None, 2, 2, 2048)        23587712
_________________________________________________________________
flatten_1 (Flatten)          (None, 8192)              0
_________________________________________________________________
batch_normalization_2 (Batch (None, 8192)              32768
_________________________________________________________________
dense_2 (Dense)              (None, 2000)              16386000
_________________________________________________________________
batch_normalization_3 (Batch (None, 2000)              8000
_________________________________________________________________
dense_3 (Dense)              (None, 5)                 10005
=================================================================
Total params: 40,024,485
Trainable params: 39,950,981
Non-trainable params: 73,504
```



We have not a strong improvement. Why?

- Too much parameters to estimate and few images.
- Because of the large number of weights that are not updated durign the train.

Section 5

CONCLUSION

Conclusion

- Not always deeper is better : Resnet-50 is very complex and it works well for large dataset:

  In this case is better to work with «customized» model.

- Try different combination and structures: average pooling, Adam optimizer, more/less convolutional layers etc.

- Use of *cv* to find the best parameters (learning rate, batch size, number of epochs, optimizer..)

- Use of *GPU:* maintain the original size.

References

- *Chollet François (2017), «Deep learning with Python»*
- *Ioffe Sergey, Szegedy Christian (2015), "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift"*
- *Kaiming He et al. (2015), Deep Residual Learning for Image Recognition»*
- *Kaiming He et al. (2015), "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification"*
- *Krizhevsky Alex, Ilya Sutskever and Geoffrey E. Hinton (2012), "Imagenet classification with deep convolutional neural networks"*