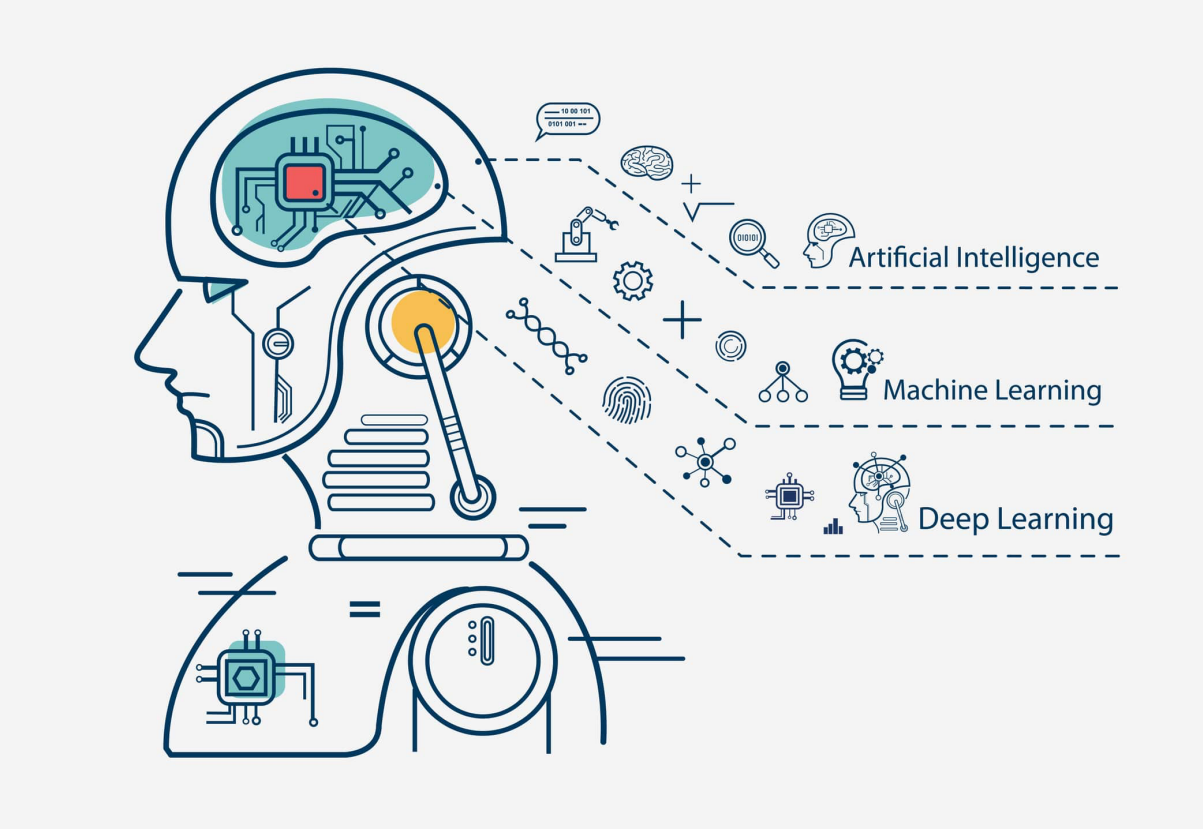
Name: Nada Samir, Nour eldin Ibrahim
Regnum: 20100752, 20101066
Course code: IN221

Machine learning project



Introduction

The project consists of two sections:

- Classifier using VGG.
- Neural style transfer.

Dataset

We used a dataset consists of images from a collection of artworks of the 50 most influential artists of All time:

- Leonardo da Vinci
- Marc Chagall
- Michelangelo
- Mikhail Vrubel
- Pablo Picasso
- Paul Cezanne

Classifier using VGG:

1. Description of the model:

VGG16 is a convolution neural net (CNN) architecture It is considered to be one of the excellent vision model architecture till date. It constructs of large number of hyper-parameters. In our algorithm we construct VGG but with minimum layer because the dataset in small and it contain:

1. 2 x convolution layer of 64 channel of 3x3 kernel and same padding
2. 1 x maxpool layer of 2x2 pool size and stride 2x2
3. 2 x convolution layer of 128 channel of 3x3 kernel and same padding
4. 1 x maxpool layer of 2x2 pool size and stride 2x2
5. 3 x convolution layer of 256 channel of 3x3 kernel and same padding
6. 1 x maxpool layer of 2x2 pool size and stride 2x2
7. 3 x convolution layer of 512 channel of 3x3 kernel and same padding
8. 1 x maxpool layer of 2x2 pool size and stride 2x2
9. 1 x Dense layer of 4096 units
10. 1 x Dense layer of 4096 units
11. 1 x Dense Softmax layer of 2 units

2. Description of the step:

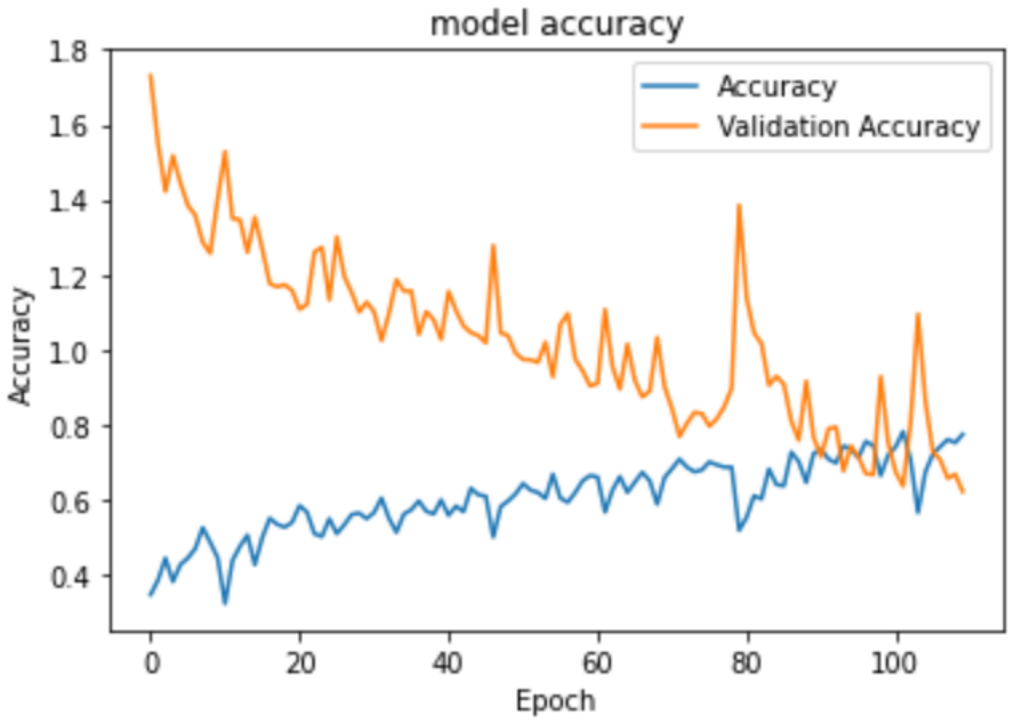
We focus on the training step more and it as follow:

1. Load the dataset (train, test) files.
2. Create the VGG layers
3. We choose our optimizer “gradient descent” by setting the learning rate 1000 and the momentum to 0.9
4. We create early stopping focus on validation accuracy by patience 20
5. We put a check point to save the highest point that focused on validation test.
6. We start to train the VGG with 1000 epochs the accuracy reached

```
model.evaluate_generator(test_generator)

/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: UserWarning: `Model.evaluate_generator` is deprecated and will be removed in a future version. Please use `Model.evaluate
  """Entry point for launching an IPython kernel.
[1.3368984460830688, 0.6062718033790588]
```

7. We observe that when we start to the epochs higher the accuracy gets much higher



8. For the overfitting and underfitting problem we used early stopping for those problems.
9. To improve the accuracy, we can test higher epochs or make learning rate and momentum less.

Training Samples:

1. First Sample: batch_size = 32, learning_rate = 1000, epochs = 2000. The resulted accuracy is nearly 60%.

2. Second Sample:

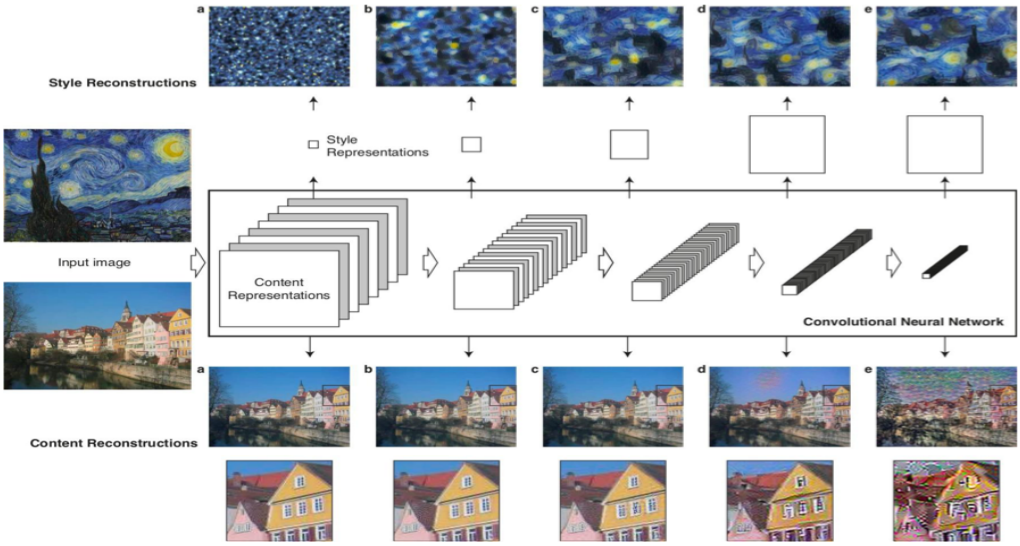
Neural style transfer

Description of the model:

Neural style transfers its model uses neural representations to separate and recombine content and style of arbitrary images, providing a neural algorithm for the creation of artistic images

There 2 representation the model need:

1. Content representation : Content features of the content image is calculated by feeding the content image into the neural network
2. Style representation: we extract the correlation of the features of the style image using gram matrix



3. To produce a new image from the content and style we used some lost function:
    a. style loss function, which keeps the generated image close to the local textures of the style reference image
    b. content loss function, which keeps the high-level representation of the generated image close to that of the content image
    c. total variation loss function, a regularization loss which keeps the generated image locally coherent
4. After this process we combine the style image and content image.

Description of the step:

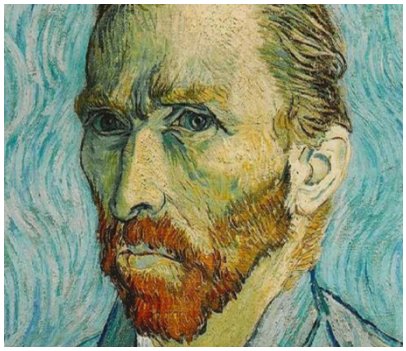
1. Upload the content image and style image you want to combine
2. CNN keeps learning a few features. The no. of channels in the output of a filter is equal to the no. of features learnt at that layer. that's why we used 5 layers in style
3. Create our optimizer using “gradient descent” initial learning rate = 300 , decay step = 100 , decay rate = 0.75
4. Change the style image and content image to tensor

5. Set number of epochs
6. For each iteration we will calculate the content loss and the style loss and total loss
7. Display image after preprocessing
8. After training we find that the accuracy was higher by lowering the decay step and decay rate
9. For the overfitting and underfitting we find if the number of epochs is very higher the model overfit.

Training sample:

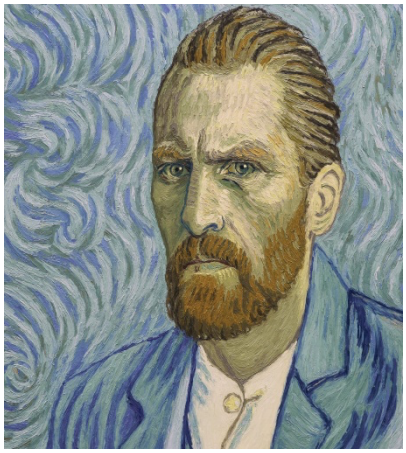
1. First sample:

   We have trained a picture with style image of Van Gough with initial learning rate = 100, decay steps = 50, decay rate = 0.75 with total variation weight = 1000, style weight = 3000, content = 2500:

 +  = 

2. Second sample:

   We have trained a picture with style image of Van Gough with initial learning rate = 100, decay steps = 100, decay rate = 0.70 with total variation weight = 500 , style weight = 2500 , content = 1200 :

 +  =