Computer Vision Lab

Luigi Faticoso
Practical Exercises

June 1, 2020

Contents

1 Introduction

The exercises of this project will be developed for the course of Computer Vision (VPC). The objective of the work is to familiarize with different neural network architectures aimed to work with pictures. We are going to use keras and tensorflow and we are going to explore the parameters and architectures to find out the best configuration for our dataset so to obtain the best model.
In every exercise there are proposed concepts studied during the course. I will describe my work and the choice I made during the development of the exercises.

2 Gender Recognition

In the first exercise was asked to implement a model able to recognize the gender of a person given as input an image of a face. The images are from " Labeled Faces in the Wild" (LFW) that has 10.585 images for the training set and 2648 for the test set. They represents images in realistic scenarios, poses and gestures. Every picture has a dimension of 100x100 in RGB format.
This work has two different goals. The first one is to obtain an accuracy better then 95% on the test set and the second is to obtain at least 90% accuracy with less then 100k parameters.

2.1 First application

For the first exercise I have used a Residual Network.

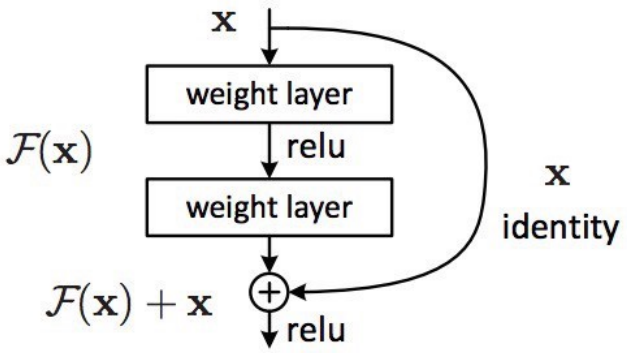


Figure 1: Representation of Residual neural network

It is composed by 4 *convolutional* blocks and one *deep fully connected* block. Every convolution has a variable number of *3x3 filters* (32, 32, 64, 128), a *batch normalization* and a *gaussian noise* layer. Every convolutional block has the double of the filters of the previous block. Every convolution block except for the first one have a second convolution, an *add* layer and a *ReLu activation* while for the first one we have a 2x2 *maxPooling* layer. The network has 1.371,202 parameters and i was able to achieve 0.95 accuracy over the test set.

2.2 Second application

For the second goal it was asked to implement a model with >90% accuracy with less than 100K parameters.
The proposed model is based on the first one. To reduce the number of parameters I have reduced the numbers of starting filters from 32 to 8 and reduced the number of convolutional operator to half. By doing so i have trained 89.154 parameters, under 100K and achieved 0.93

2.3 Results

Model	Parameters	Epochs	Train Acc	Test Acc
Model 1	1.371,202	300	0.9744	0.9558
Model 2	89.154	500	0.9308	0.9381

Table 1: Results of the first exercise

3 Car Recognition

For the second exercise is asked to implement a model able to identify 20 different models of car. The dataset is composed by 1575 images of 250x250 dimension in RGB format. For the exercise I have used the bi-linear network that consist in the combination of two indipendent convolutional streams forming one unique vector.

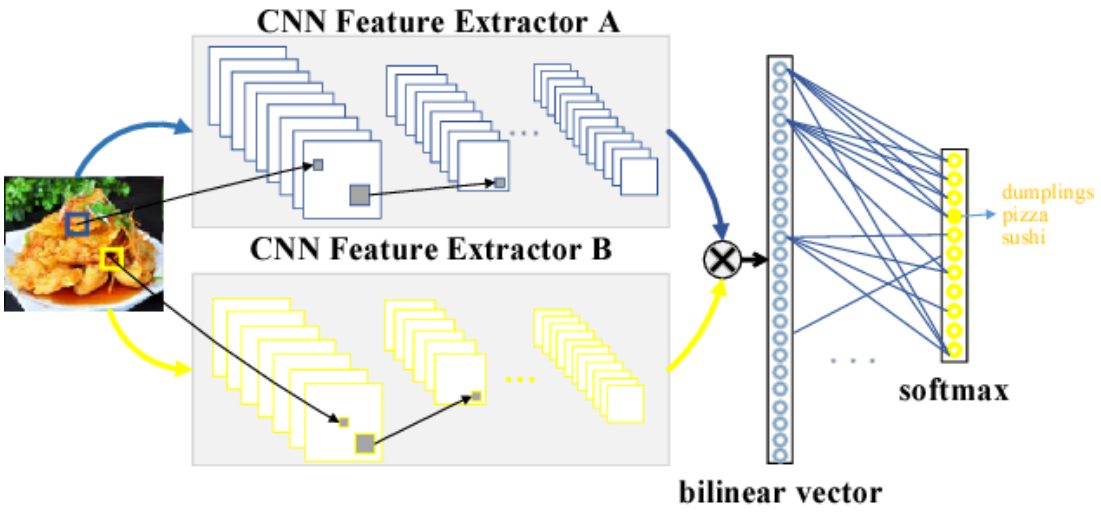


Figure 2: Representation of bilinear CNN model

The objective of the work is to analyze the behaviour of the network. To better understand the architecture I have made two different experiments. The first one I have developed the model making use of two different CNNs. For the second experiment I have used two pre-trained VGG16 model.

3.1 Using two different CNNs bi-linear model

The first experiment i made was using two different CNNs in each of the streams in the bi-linear model. The starting point was the code provided on the github repository provided by the professor. With this base implementation I was able to achieve an accuracy over the test of around 41%.

3.2 Using two pre-trained VGG16 bi-linear model

In this experiment I have used two pre-trained VGG16 models. Some considerations have had to be taken into account, such as renaming the different layers, since they were two identical models, they initially had the same names and this resulted in an error. In addition I have made two experiments: In one I have frozen the weights of the VGG and in the other one I haven't. For the model with frozen weight the results achieved are 0.58 while for the model obtained without freezing the weights I have achieved 0.26.

3.3 Results

We are going to see the results of my experiments in details.

Model	Epochs	Train Acc	Test Acc
Diff CNN	350	0.9671	0.4107
VGG16	250	0.4475	0.2653
VGG16 Frozen	200	0.9937	0.5880

Table 2: Results of the experiments

The model that performs better is the one using pre-trained VGG16 models with frozen weights achieving 58% of precision.

4 Style Transfer

For the last exercise was asked to realize the experiment understanding the technique of *Style Transfer*. This technique consists in obtaining a new image from the mix of two input images (First image: the content of the image. Second image: style we want to apply).

Figure 3: Representation of Neural Style Transfer Technique

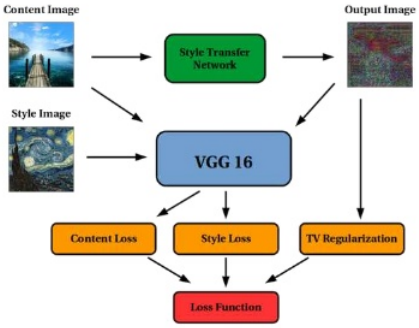


Figure 4: Representation of Neural Style Transfer Architecture

In particular, Figure 4 shows how the total loss as the combination of different losses, one is based on the content and the other one based on the style. When the losses converges it is possible to achieve the final generated image similar to both of the input images (content and style). As a starting point I have used the code proposed by the professor on his GitHub repository.
I have used those images as references for the content image and for the style image.

(a) Content Image

(b) Style Image

Figure 5: Images used for the Style Transfer exercise

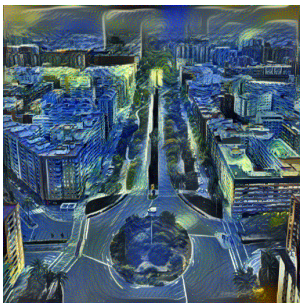
4.1 First Experiment

The first experiment was to test the behavior of the neural network by modifying the loss function of the content and of the style. Firstly I have tried using the default weights (content = 0.1 and style = 100.0) and represented it in figure 6b Then i have tried using the same value for both of the loss function resulting in figure 6a As I can understand by now the more weight I set to the style loss, the more effect of the style is possible to see in the result. I have tried different set of weights in figures 6 .

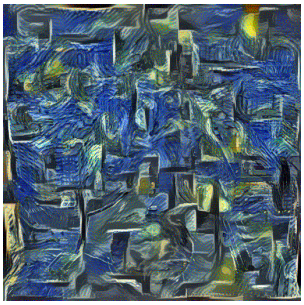
(a) Same weights (Content = Style)

(b) Style weight 100 times Content weight

(c) Style weight 500 times Content weight



(d) Style weight 1000 times Content weight



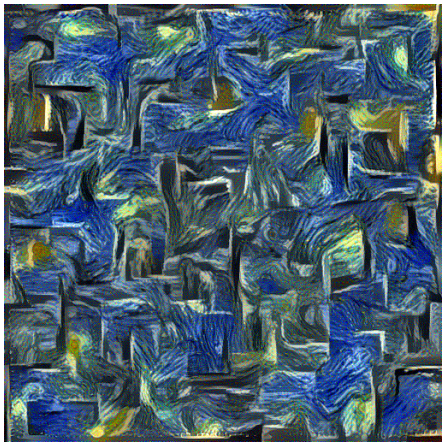
(e) Style weight 5000 times Content weight

Figure 6: Experiments made with Neural Style Transfer

4.2 Second experiment

For the second experiment I have tried modifying the loss function. I have modified the convolutional layers. For the first result i have used as a layer feature the first convolutional block and as a feature layers the last convolutional blocks while for the other one i have made the opposite experiment. As is possible to see from the result obtained is that when we use the first convolutional layers as layer feature (figure 7a the results image is very similar to the input image without much of the applied style while when we do the opposite (figure 7b) is possible to see the filters without much of the content image.

(a) layers features used: *block2_conv2*



(b) layers features used: *block5_conv3*

Figure 7: Second Experiment made with Neural Style Transfer

4.3 Conclusion

In conclusion I have seen and worked with only two parameters of the proposed neural network but there are many other parameters that could affect the results and needs more experimenting.

List of Figures