Computer Vision Laboratory

Jaime Ferrando Huertas

May 2021

Contents

1 Introduction

This report contains a summary of the work presented for the lab assigments of Computer Vision lecture. With one section per lab assigment we will describe the theory and implementation we have done to fullfil the requirements. All code has been develop in Pytorch and is attached along this report file.

2 Gender Recognition

For this assigment we were asked two implementations:

- Implement a model with at least 97% accuracy over test set
- Implement a model with at least 92% accuracy with less than 100K parameters

The assigment statement proposed to follow an emsembled approach but we chose to go with a minimal version of resnet that could accomplish both requirements at once. Our model *ResNet_small* consisted only of one convolution+batchnorm followed by one ResNet block without a shortcut and a final linear layer, making it similiar to a 3 conv+batchnorm architecture. All convolutions had kernel size of 3. The full architecture had a a total of 28706 trainable parameters, way less that the maximum of 100k. With the help of OneCycleLR[3] learning rate scheduler we managed to accomplish both statments. Following figures contain our results.

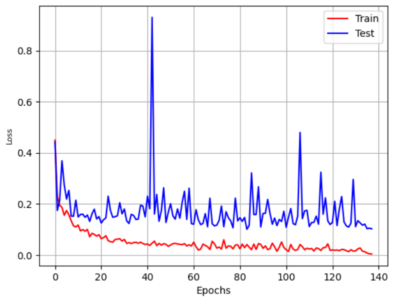

Figure 1: Loss

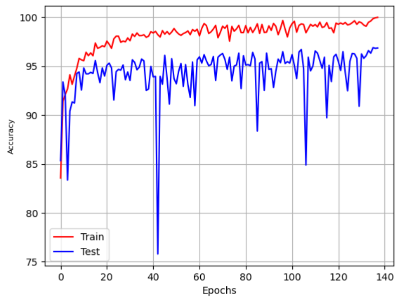

Figure 2: Accuracy

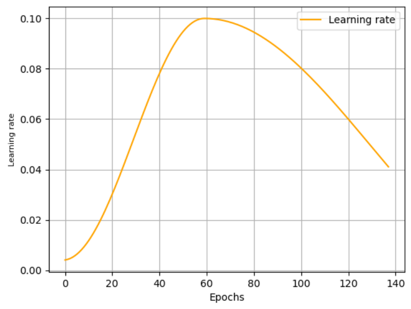

Figure 3: Learning rate

This learning rate scheduler gets a very fast convergence by exponencially increaseing learning rate (starting low, avoiding local minimums) and drastically decreasing lr after reaching 0.1, allowing us to fine tune the network to achieve 97% performance with only 28k parameters.

3 Car detection

For this task we needed to implement a bi-linnear[2] CNN model capable of reaching 65% accuracy on classifing cars within 20 different models. A bi-linnear CNN model combines two CNN final feature extraction layers output into one single vector by applying the outer product.

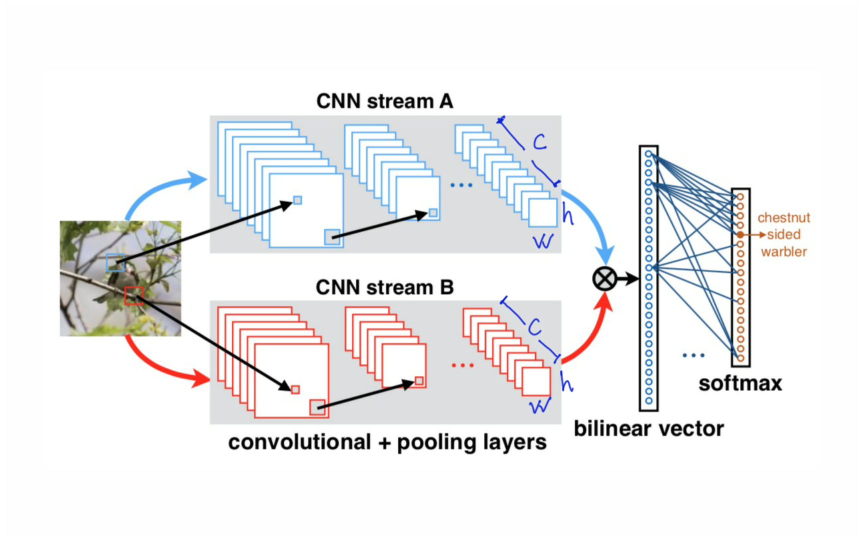

Figure 4: Bi-linnear CNN

Our implementation uses one single CNN but we apply two different dropout layes on its output, getting us the equivalent to two CNN outputs. This two CNN outputs are then combined following as the bi-linnear architecture proposes. We choose to use a pre-trained model as the single CNN, vgg16. We used OneCycleLR again for this task and no weight freezing or other techniques were used. Here are our results:

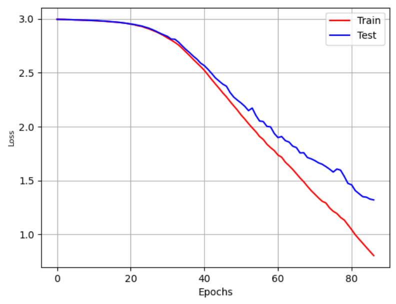

Figure 5: Loss

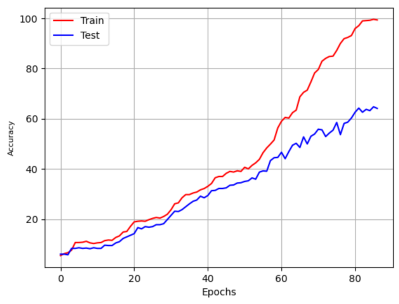

Figure 6: Accuracy

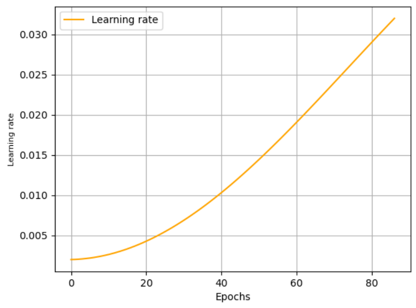

Figure 7: Learning rate

We can see how loss and accuracy don't improve much during inital steps but this is because the learning rate scheduler hasn't really started to increase exponencially yet. As soon as the learning rate increases we see both metrics improve and achieve the desired performance. By the time we fill our task the training and test accuracy have started to diverge, almost getting a 100% accuracy on training set. This means that we were probably going to start the overfit phase in a few epochs.

4 Style transfer

For the last task we implemented a style transfer model. Our model is a replica of the Neural Style Algorithm [1] applied with a VGG19 network. This model we take a content image and style image to return an image that has both the content of the first image and the style of the second. It achieves this by creating two custom losses, one for the content and one for the style. This also allows us to play with how we combine the loss feeded to the optimizer, being able to give higher importance to one against the other. Now we present some examples of output images we got by playing with that ratio content-style.

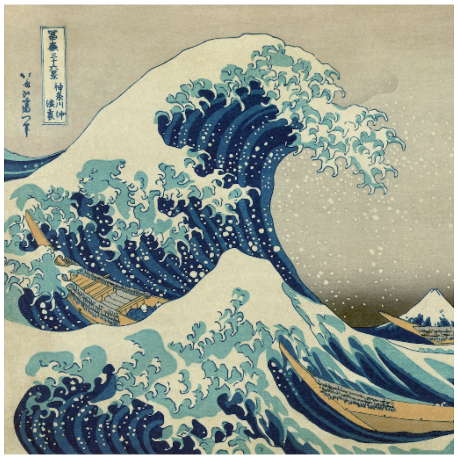
Figure 8: Style image

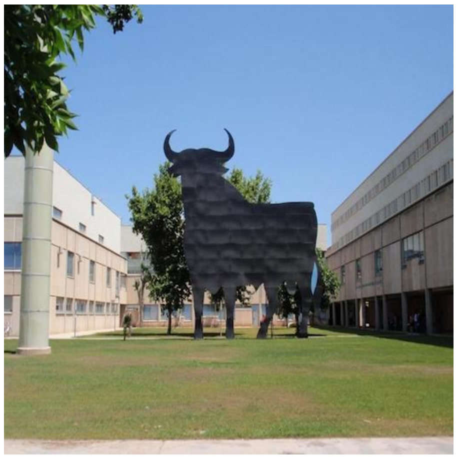
Figure 9: Content image

Results:

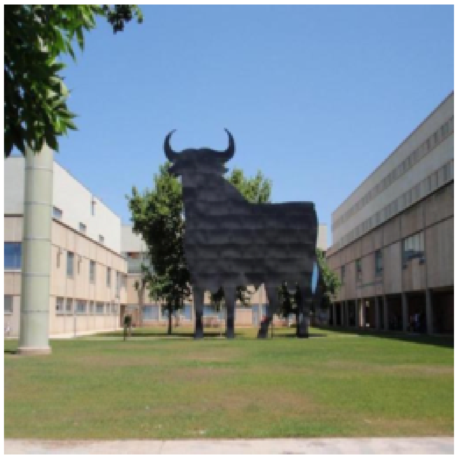
Figure 10: Result ratio 1-1

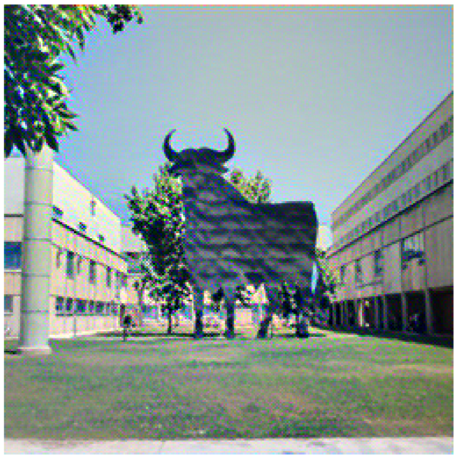
Figure 11: Result ratio 1-1000

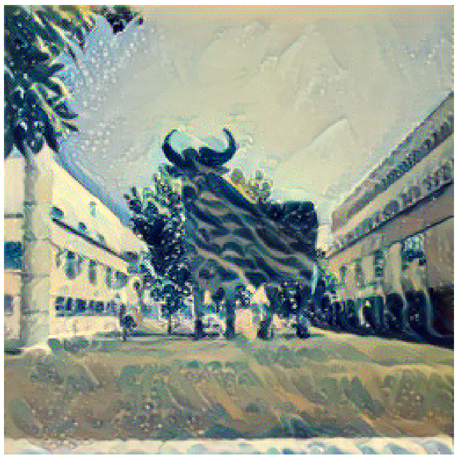

Figure 12: Result ratio 1-1000000

We see that in order to get meaningful results we need to increase the style weight much more than the content. This could be because we are running the experiments for only 200 steps, giving a higher weight to style makes the model overfit to it inmediiately. Let's try now with total different images to see how it behaves.

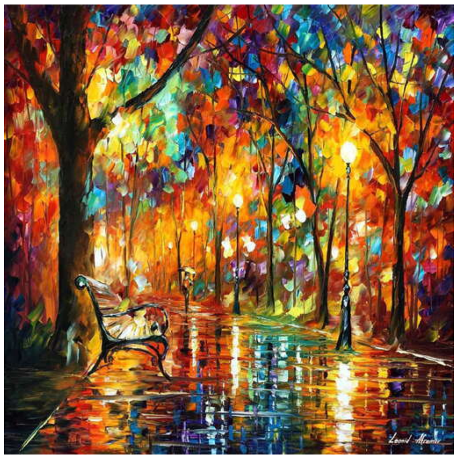

Figure 13: Style image

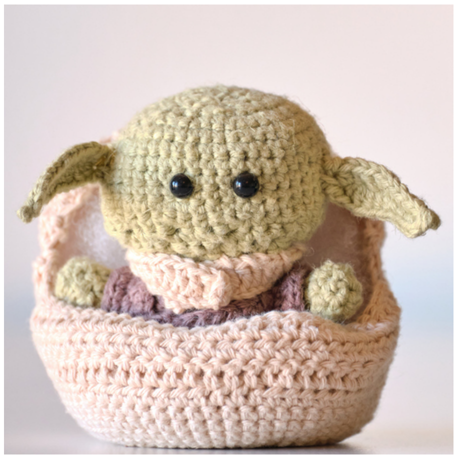

Figure 14: Content image

Results:

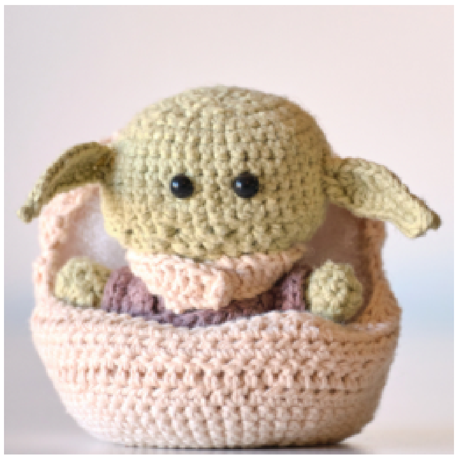

Figure 15: Result ratio 1-1

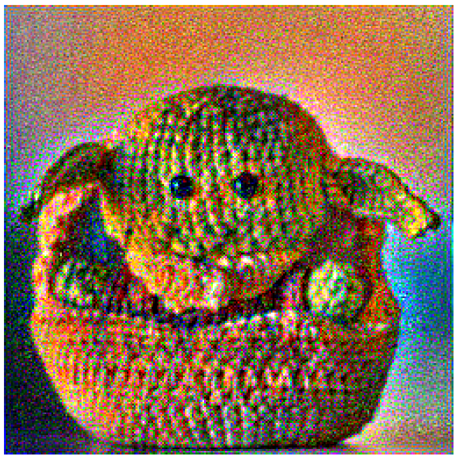

Figure 16: Result ratio 1-1000

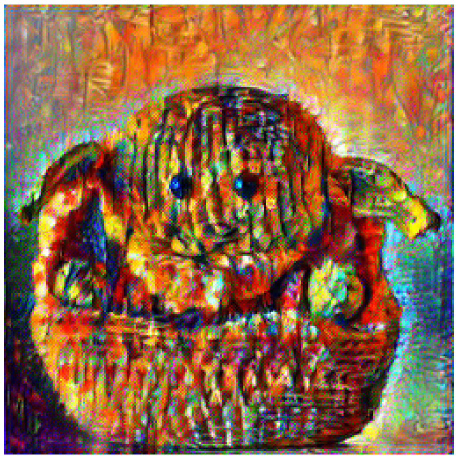

Figure 17: Result ratio 1-1000000

We see similar results as we increase the content-style ratio.

References

[1] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of artistic style. *CoRR*, abs/1508.06576, 2015.

[2] Tsung-Yu Lin, Aruni RoyChowdhury, and Subhransu Maji. Bilinear CNN models for fine-grained visual recognition. *CoRR*, abs/1504.07889, 2015.

[3] Leslie N. Smith and Nicholay Topin. Super-convergence: Very fast training of residual networks using large learning rates. *CoRR*, abs/1708.07120, 2017.