ECE468 Final Project: Image Super-Resolution Using Convolutional Neural Network

Austin Dibble

December 4, 2018

1 Introduction

This report covers the design and implementation of a convolutional neural network (CNN), in order to perform super-resolution on images. The implementation was performed using Python3.7 and the PyTorch library. Overall, the development of the network was performed in a trial-and-error fashion, where the network structure was modified until reasonable performance was achieved. After the network structure was decided upon, the training hyperparameters were modified in order to measure the network's performance given different sets of hyperparameters. A description of this process and the experimental results are detailed below.

2 Neural Network Design Implementation

2.1 Network Structure Overview

The neural network structure that gave the best results after training is shown in Figure 1, using an example input image of size 25x25. In the figure, one can see that there are four convolutions, making for a total of five layers, including the input and output images. The only kernel that changes the overall size of the outputs is the transpose convolution. This kernel is used to increase the size of the outputs by the upscale value. In this example, it would be an upscale of two.

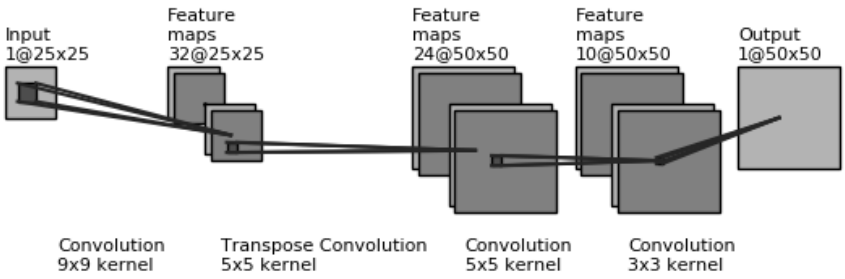


Figure 1: Example of a direct-mapped cache.

Overall, my reasoning for selecting this network architecture was simply due to experimentation. My first attempt used four convolution kernels with large amounts of calculated zero padding in order to increase the image size to the necessary dimensions over the four convolutions, as I was not aware of the transpose convolution kernel type. This did not work very well, as it left large amounts of edge artifacts in the final output images.

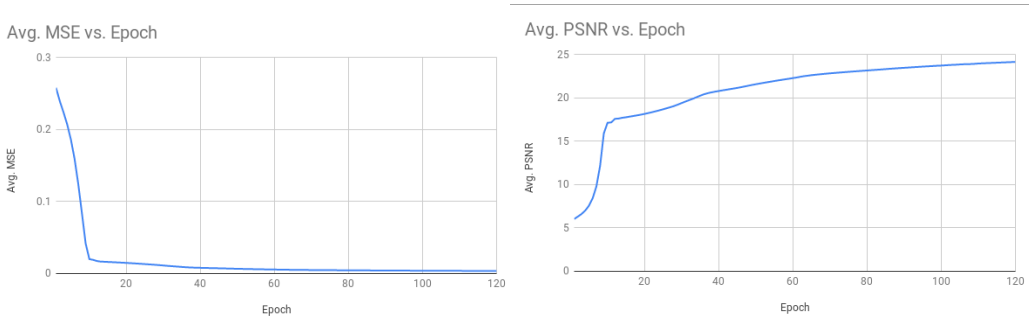
2.2 Network Loss Experimentation

In my experimentation, I tested several different combinations of hyperparameters after the experimental discovery of my "good" network architecture. I logged my loss (MSE and PSNR) into a spreadsheet at each epoch for each of the training periods of my network implementations, hoping to show the effects of different hyperparameters.

2.2.1 CNN for Upscale = 2

For my first network implementation, the MSE and PSNR are graphed over the epochs of training. These are both shown in Figure 2, side-by-side. This network used the same structure as previously described in section 2.1, with an upscale of two. In this case, my hyperparameters were:

- Batch Size: 20
- Learning Rate: 0.0001
- Number of Epochs: 120



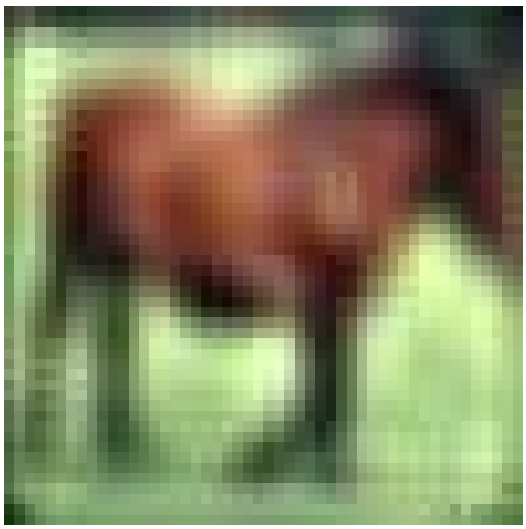
(a) MSE v. Epoch

(b) PSNR v. Epoch

Figure 2: Figure showing the MSE and PSNR over epochs during training for the first CNN, upscale = 2.

Shown in Figure 3 is a comparison of the network's output image against a sample input image of size 25x25. One can see that while the image is successfully scaled, there are some significant artifacts throughout the images, especially along the image edges.

(a) 25x25 pixel horse image

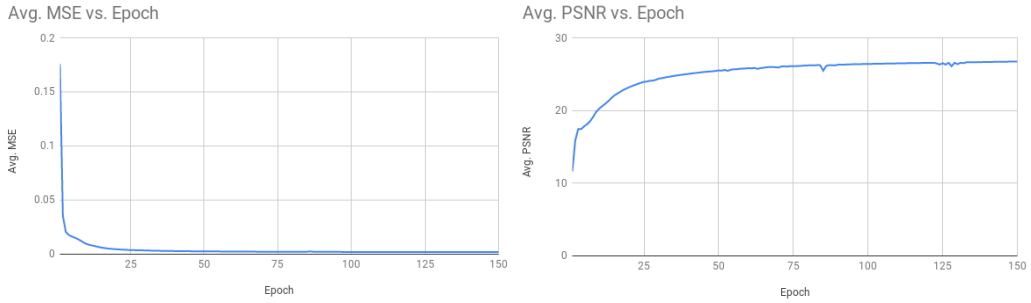


(b) 50x50 pixel network output

Figure 3: Figure showing a 25x25 image of a horse, and its 2x upscaled counterpart after going through the CNN.

To continue testing, the hyperparameters were modified, but the same network structure was still used for the results seen in Figure 4. For this training set, the new hyperparameters were as follows:

- Batch Size: 20
- Learning Rate: 0.001
- Number of Epochs: 150



(a) MSE v. Epoch

(b) PSNR v. Epoch

Figure 4: Figure showing the MSE and PSNR over epochs during training for the first CNN, upscale = 2. Note this training had different hyperparameters.

As one can see in the graphs in Figure 4, the MSE converges much more quickly than in the previous network training, shown in Figure 3. Additionally, the PSNR reaches a maximum near 27 dB, which is much higher than the network trained with different parameters. To highlight the performance gain seen in the network, another input/output comparison is shown in Figure 5.

(a) 25x25 pixel horse image

(b) 50x50 pixel network output

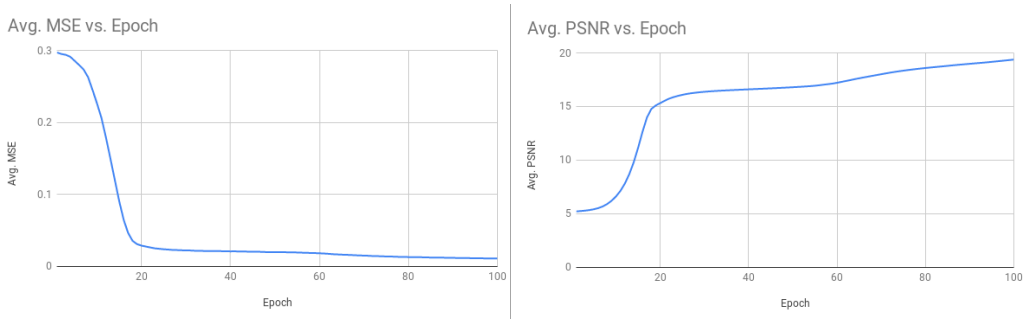
Figure 5: Figure showing a 25x25 image of a horse, and its 2x up-scaled counterpart after going through the CNN, trained with new hyperparameters.

As can be see in Figure 5, the output result is far superior to that in Figure 3. I believe this is not only due to the higher learning rate, but also to the longer training time. One can see that in the output, the horse image has far less noise added in, and also seemingly higher detail (more muscle definition) than the original 25x25 pixel image.

2.2.2 CNN for Upscale = 4

For my second network implementation, the MSE and PSNR are graphed over the epochs of training. These are both shown in Figure 6, side-by-side. This network has a slightly different structure for optimizing the output for an upscale of four. Here were the hyperparameters for the first experiment represented in Figure 6.

- Batch Size: 30
- Learning Rate: 0.0001
- Number of Epochs: 100



(a) MSE v. Epoch (b) PSNR v. Epoch

Figure 6: Figure showing the MSE and PSNR over epochs during training for the first CNN, upscale = 4.

Shown in Figure 7 is a comparison of the network's output image against a sample input image of a car. The image is scaled up, but there are significant blurring effects on the output.

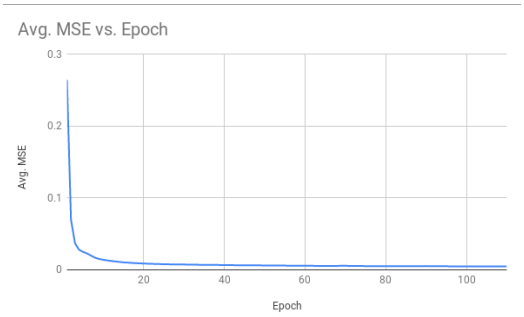
(a) Original car image

(b) Network output for car image, 4x scale

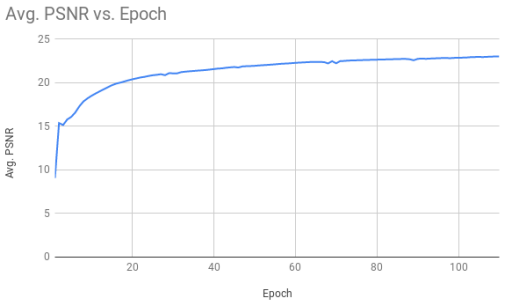
Figure 7: Figure showing an image of two cars, and its 4x upscaled counterpart after going through the CNN.

To continue testing, the hyperparameters were modified on the training state, but the same network structure was still used for the results seen in Figure 8. For the training set, the new hyperparameters were as follows:

- Batch Size: 20
- Learning Rate: 0.001
- Number of Epochs: 110



(a) MSE v. Epoch



(b) PSNR v. Epoch

Figure 8: Figure showing the MSE and PSNR over epochs during training for the first CNN, upscale = 4. Note this training had different hyperparameters than the previous car test.

As one can see in the graphs in Figure 8, the MSE converges much more quickly than in the previous network training, shown in Figure 7. Additionally, the PSNR reaches a maximum PSNR well over 20dB, which is much higher than the network trained with different parameters (which did not reach over 20db PSNR). To highlight the performance gain seen in the network, another input/output comparison is shown in Figure 9.

(a) Original car image

(b) 4x scaled network output

Figure 9: Figure showing the original car image, and its upscaled counterpart, outputted by the network.

In Figure 9, the output result is far superior to that of Figure 7. While there are still some effects on sharp images in Figure 9.b, it is a much more detailed result than the previous, highly blurred, result. As in the example of the previous network (horse images), it seems that this better result is largely due to a change in learning rate, from 0.0001 to 0.001. While increasing the learning rate introduced a little instability in the PSNR (see ripples in Figure 8), it also gives a much better result.

2.3 Results

In summary, I have analyzed two separate networks using two sets of hyperparameters for each network, and looking at the error and output results based on those hyperparameters. In the case of the 2x scale network, the best results were found when I used the following hyperparameters:

- Batch Size: 20
- Learning Rate: 0.001

- Number of epochs: 150

With these hyperparameters, the final highest PSNR was 26.8dB. In the case of the 4x scale network, the best results were achieved with these hyperparameters:

- Batch Size: 20
- Learning rate: 0.001
- Number of epochs: 110

As a final note, here are some 2x upscale results from the network, compared with the 2x upscale performed using bicubic interpolation in the photo editing program, GIMP. Surprisingly, the results in both Figure 10.a and Figure 11.a are comparable if not better than their counterparts in Figure 10.b and Figure 11.b.

(a) Image upscaled using network

(b) Image upscaled using bicubic interpolation

Figure 10: Figure comparing sections of an image, one scaled by the network, and one scaled by an interpolation algorithm.

(a) Image upscaled using network

(b) Image upscaled using bicubic interpolation

Figure 11: Figure comparing an image, one scaled by the network, and one scaled by an interpolation algorithm.