Summer Poissonnier

Ania Schulz

CAP 4401 – Digital Image Processing

**Single Image Super-Resolution Using Deep Learning**

**Motivation**

The first thing my partner and I did was look through the list of the suggested topics and then we selected the topic that we thought would work best for us. After looking through them we decided to select the topic: Single Image Super Resolution using deep learning because it appears to be the most feasible. It was also a topic that we thought looked interesting; we were excited to get started on this project and see how we could modify it.

**Background**

Single image super resolution is a process in which high resolution images are created from low resolution images. The purpose of SISR is to recover a single high-resolution image from a single low-resolution image. SISR does face some challenges and is an ill-posed problem. The reason for this is because there are many possible solutions for any low-resolution pixel. This example uses the VDSR network (a convolutional neural network) designed to perform single image super-resolution. Paper [1] mentions a proposed model called the SRCNN which has more appealing properties than the VDSR. The SRCNN’s structure provides more accurate results, guidance for the network structure, and good quality and speed.

**Example code**

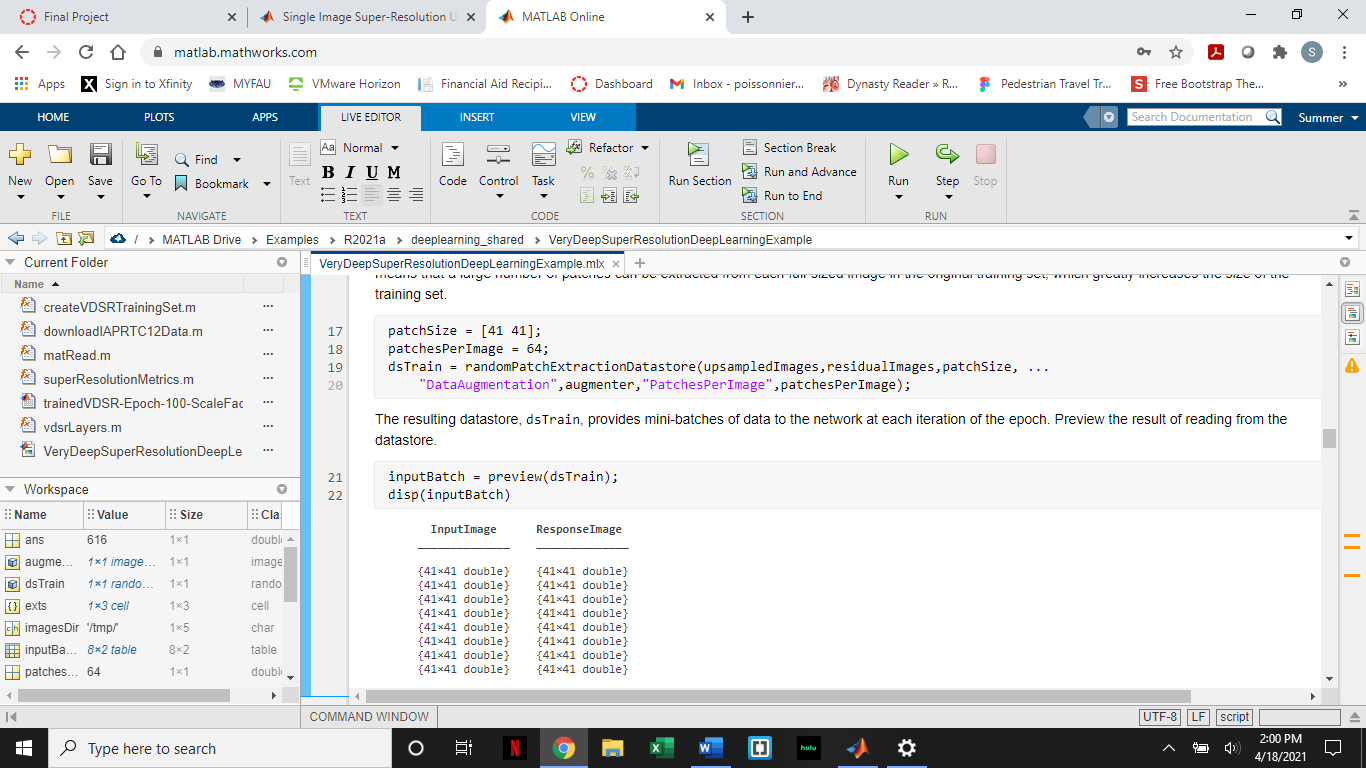
Upon first glance, there are definitely issues with the example code. The high-resolution image compared to the original reference image is definitely not as accurate as one would hope. In the modification part of the paper, we will modify the code in ways to see if we can get the accuracy of the high-resolution image to be greater.

The first part of the code in the example was downloading training and test data which consisted of photos of people, animals, cities, etc. The code then listed the number of training images which was 616.

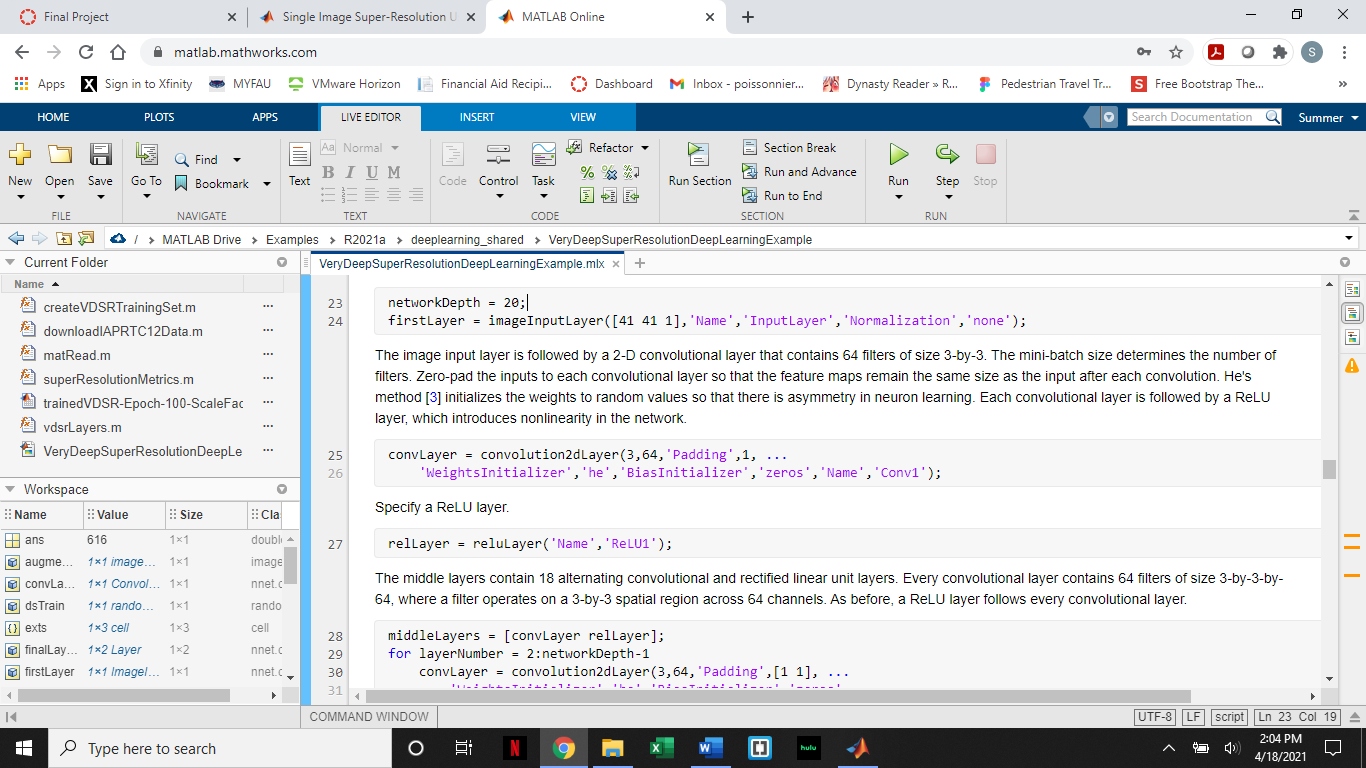
The second part of the code was preparing the training data. To create the training set, the code generated pairs of unsampled images and corresponding residual images.

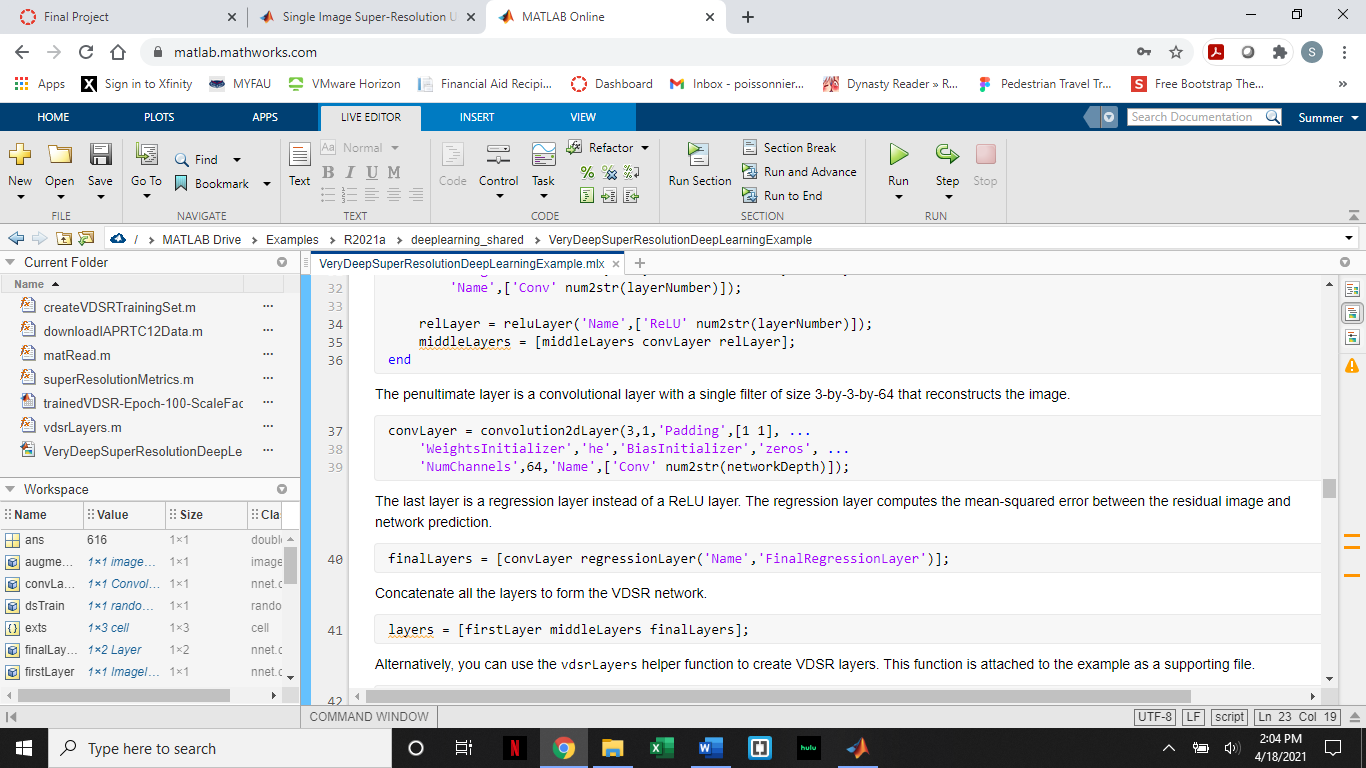
The next step in the code was to define the preprocessing pipeline for the training set. This part used an image data augmenter function and a random patch extraction datastore that performs randomized patch extraction from unsampled and residual image datastores. The

resulting datastore provides mini batches of data for the network at each iteration of epoch.



Then, the set up of the VDSR layers was coded for using 41 individual layers. Some of the important layers in this section are: imageInputLayer, convolutional2dLayer, reluLayer, and regressionLayer.





The next two steps were specifying the training options and training the code. For the specifications:

maxEpochs = 100;

epochIntervals = 1;

initLearningRate = 0.1;

learningRateFactor = 0.1;

l2reg = 0.0001;

miniBatchSize = 64;

options = trainingOptions('sgdm', ...

'Momentum',0.9, ...

'InitialLearnRate',initLearningRate, ...

'LearnRateSchedule','piecewise', ...

'LearnRateDropPeriod',10, ...

'LearnRateDropFactor',learningRateFactor, ...

'L2Regularization',l2reg, ...

'MaxEpochs',maxEpochs, ...

'MiniBatchSize',miniBatchSize, ...

'GradientThresholdMethod','l2norm', ...

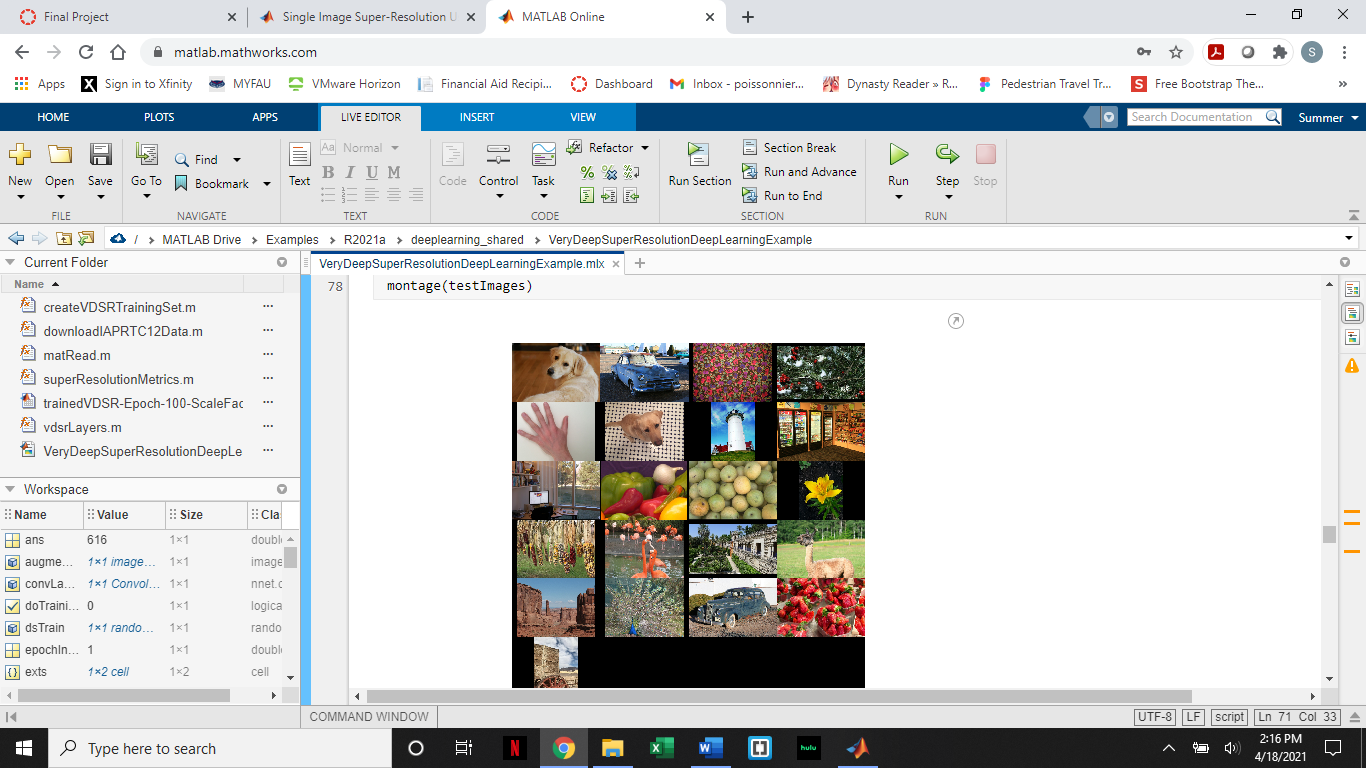
'GradientThreshold',0.01, ...

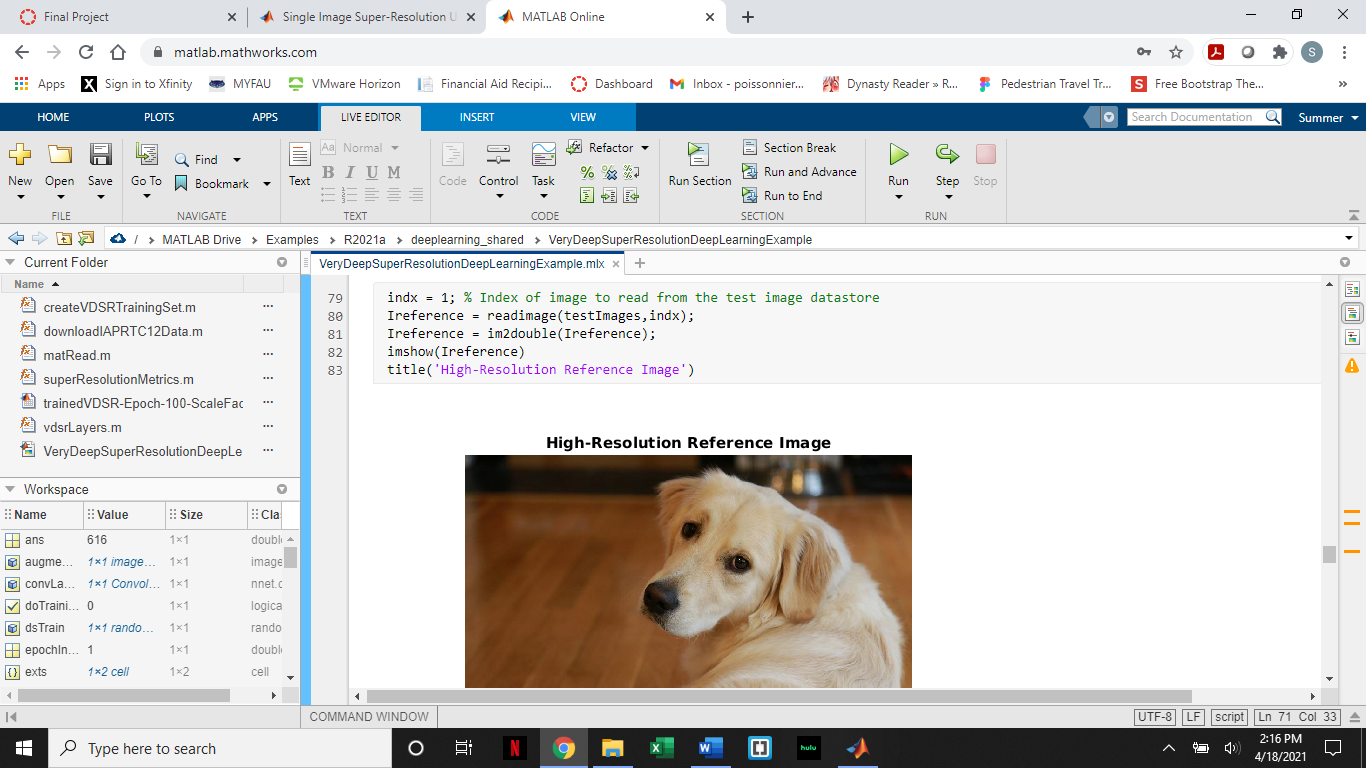
'Plots','training-progress', ...

'Verbose',false);

This part of the code we will modify in the next part of the paper.

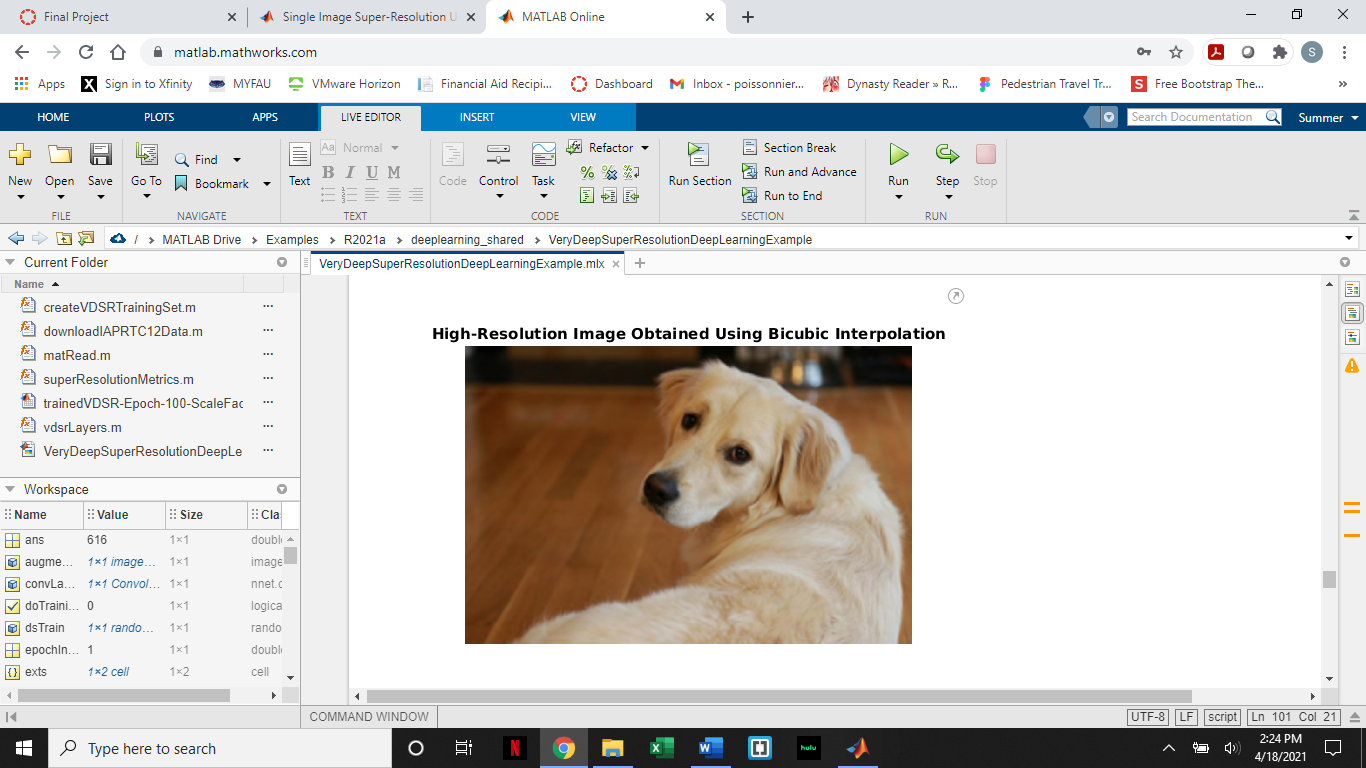
Then, the code created a sample low-resolution image and then displayed the testing images as a montage. You could then choose which high-resolution image to use; the example used an image of a dog.



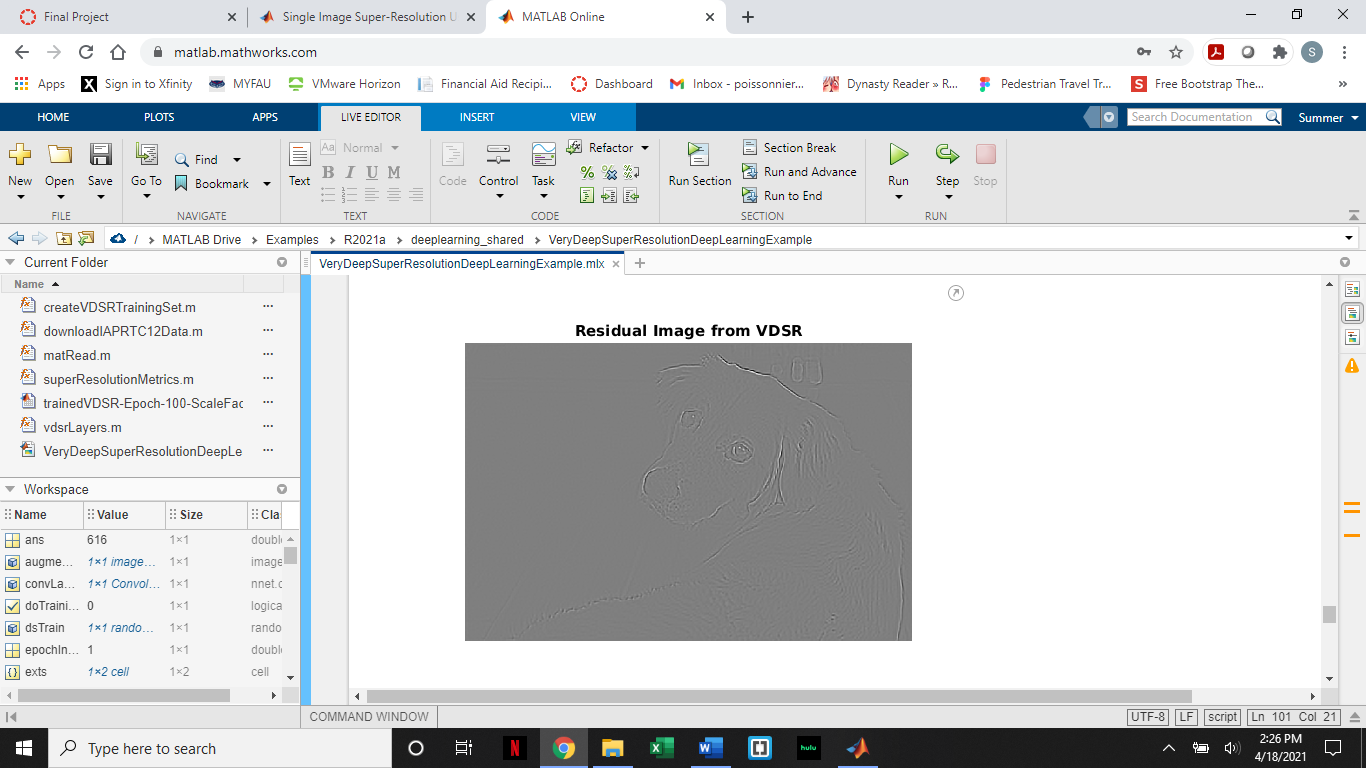
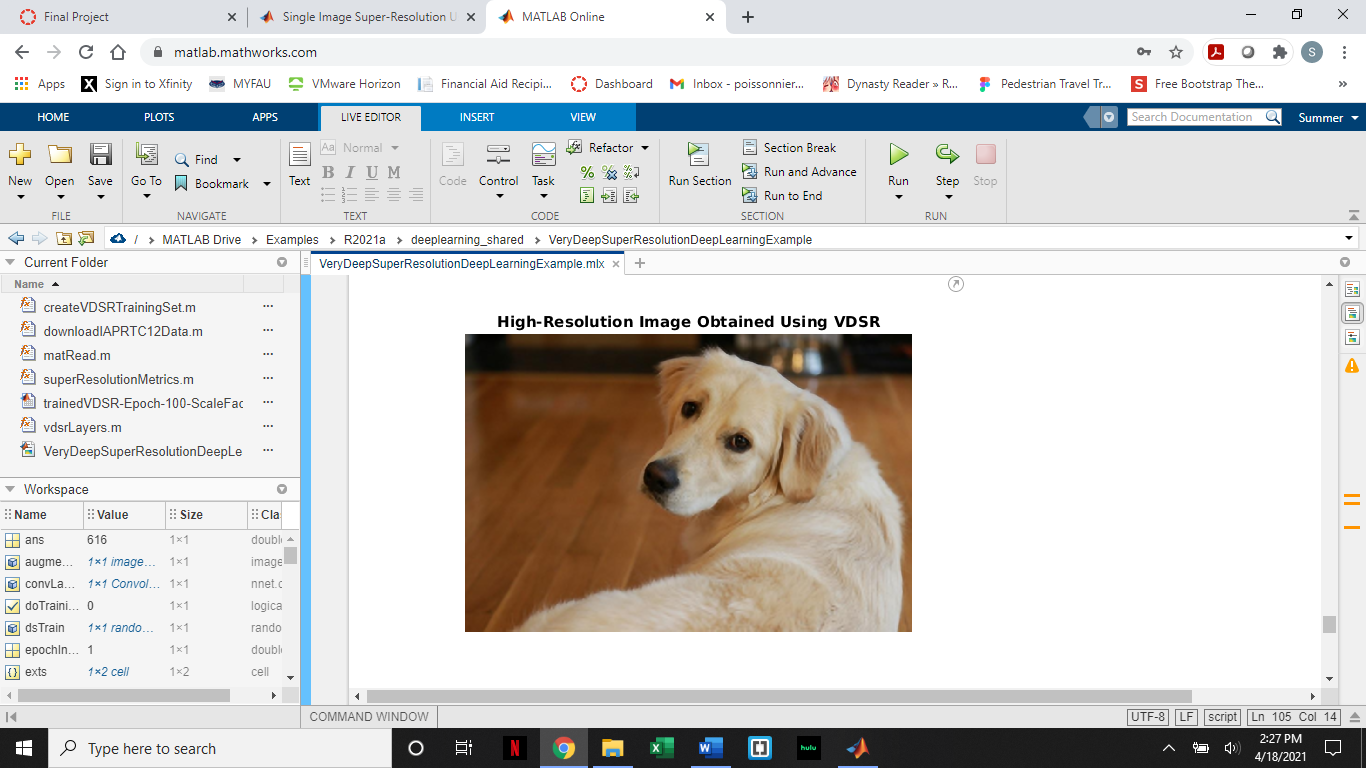


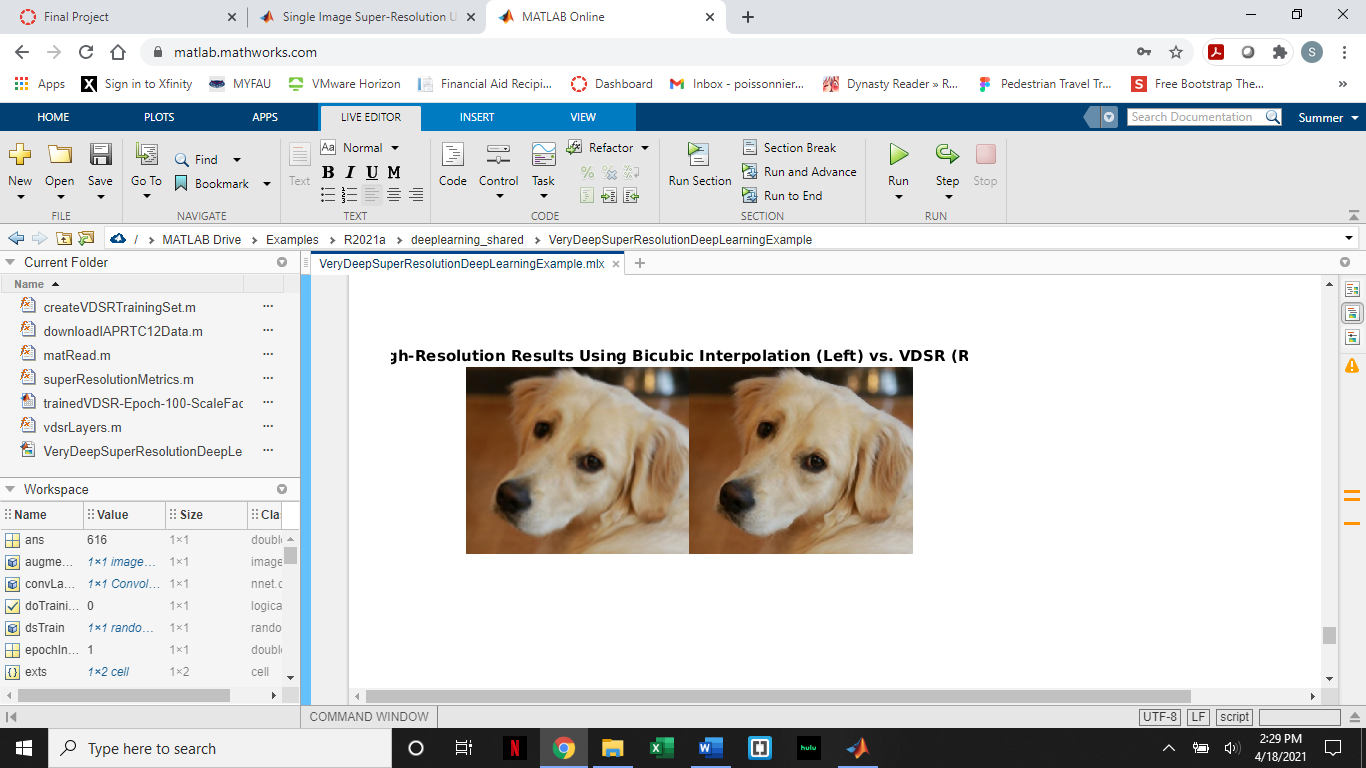
****

The example then used Bicubic interpolation instead of deep learning to get a high-resolution from the low-resolution image.

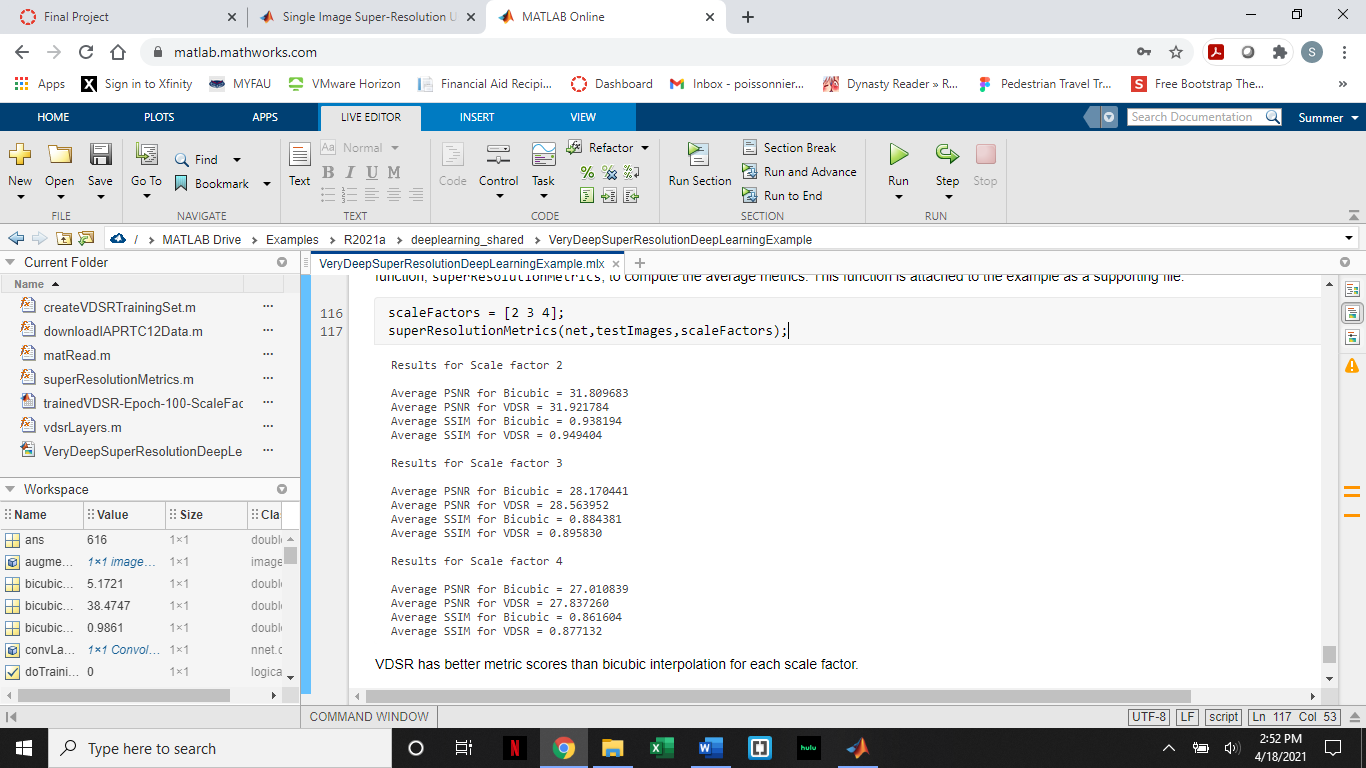


Then, the network went to improve the image resolution using VDSR network.

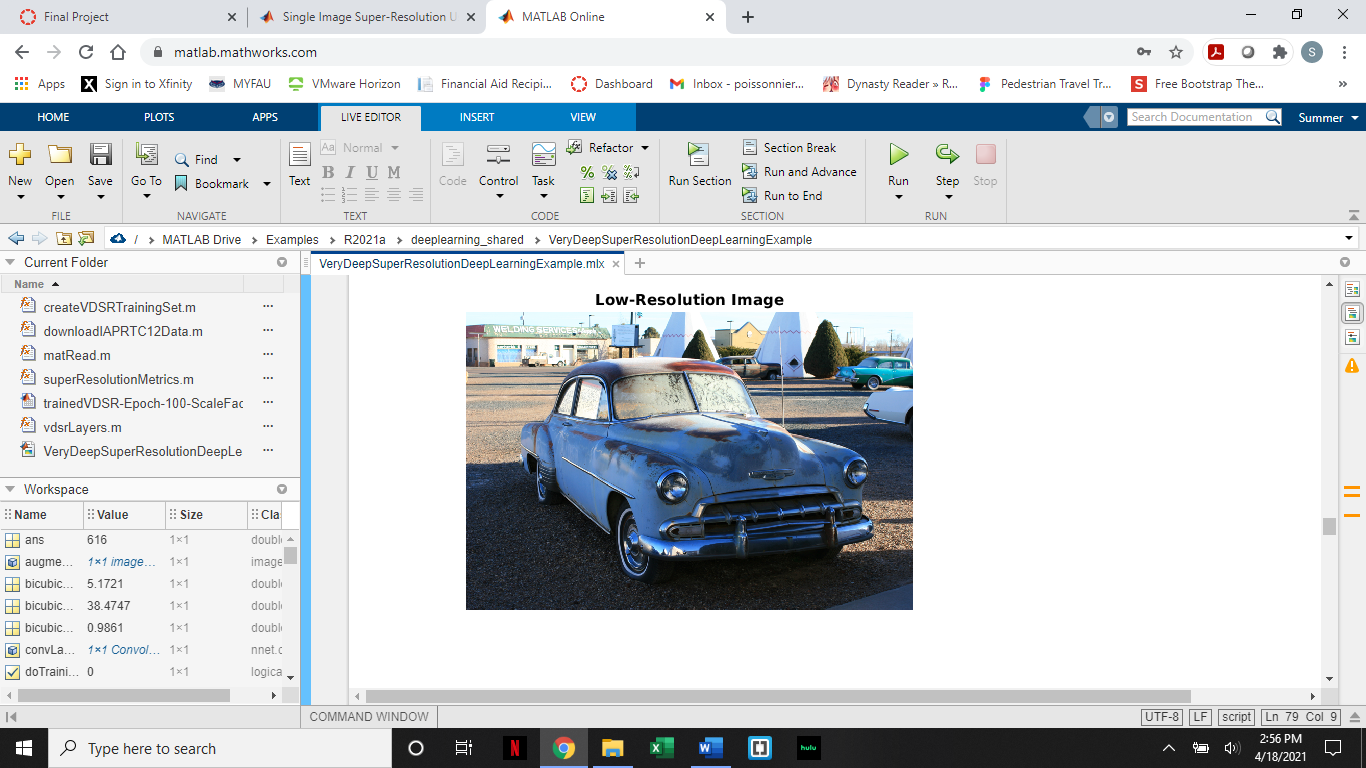


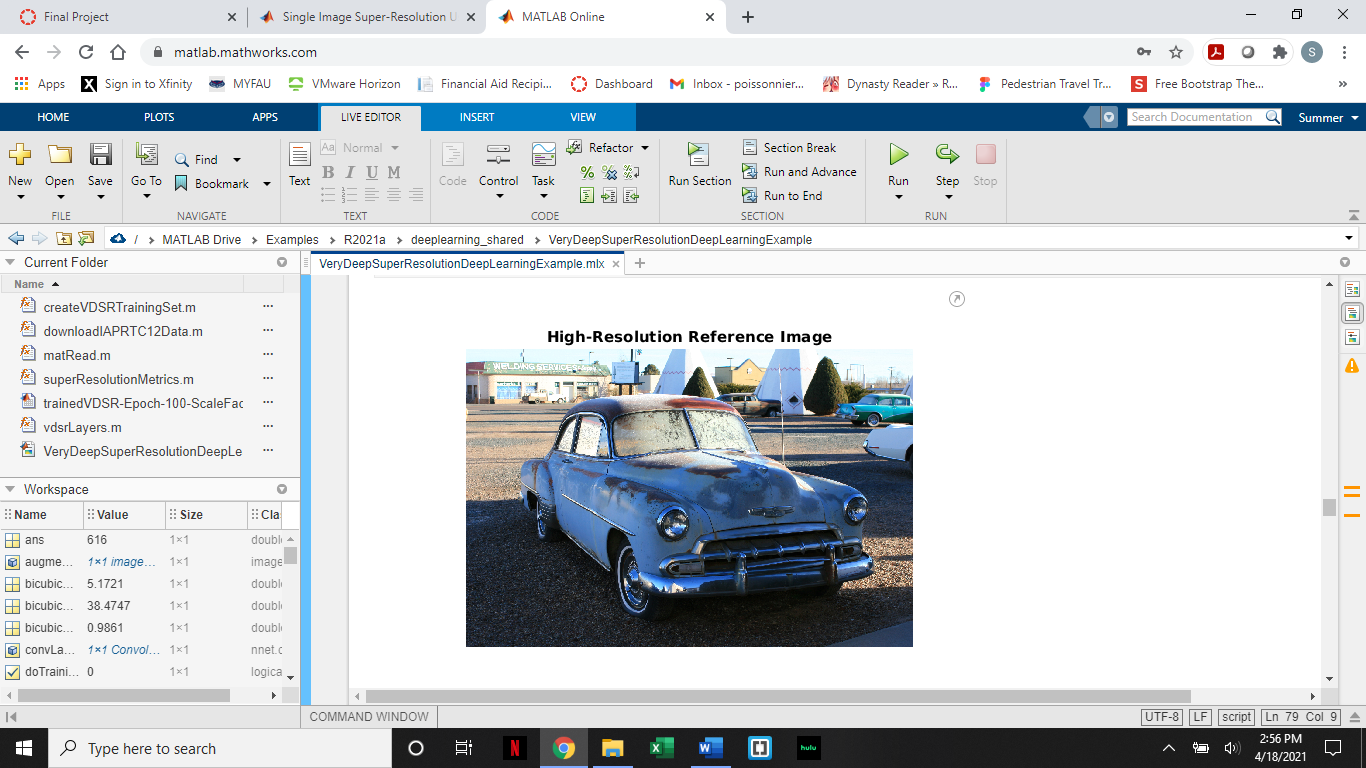


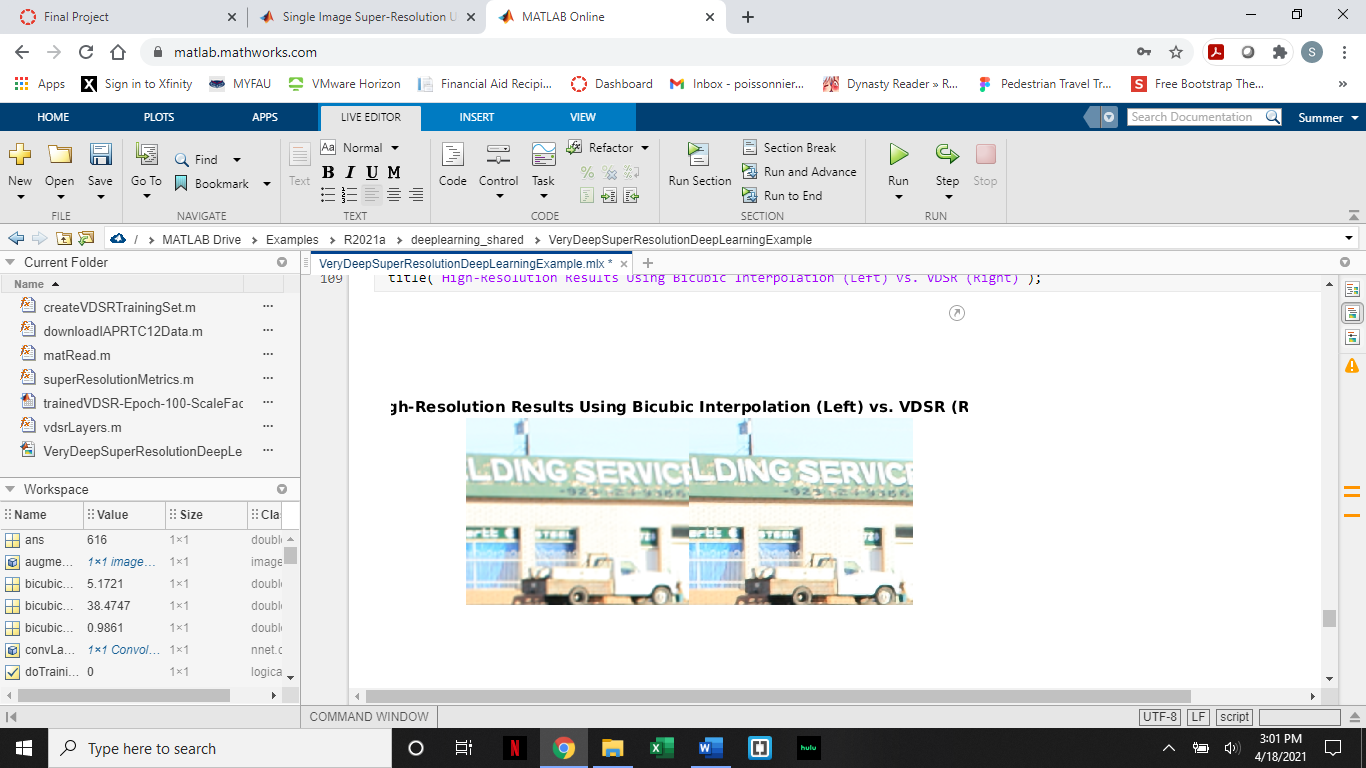
Then, to compare the image quality, the code used image quality metrics.



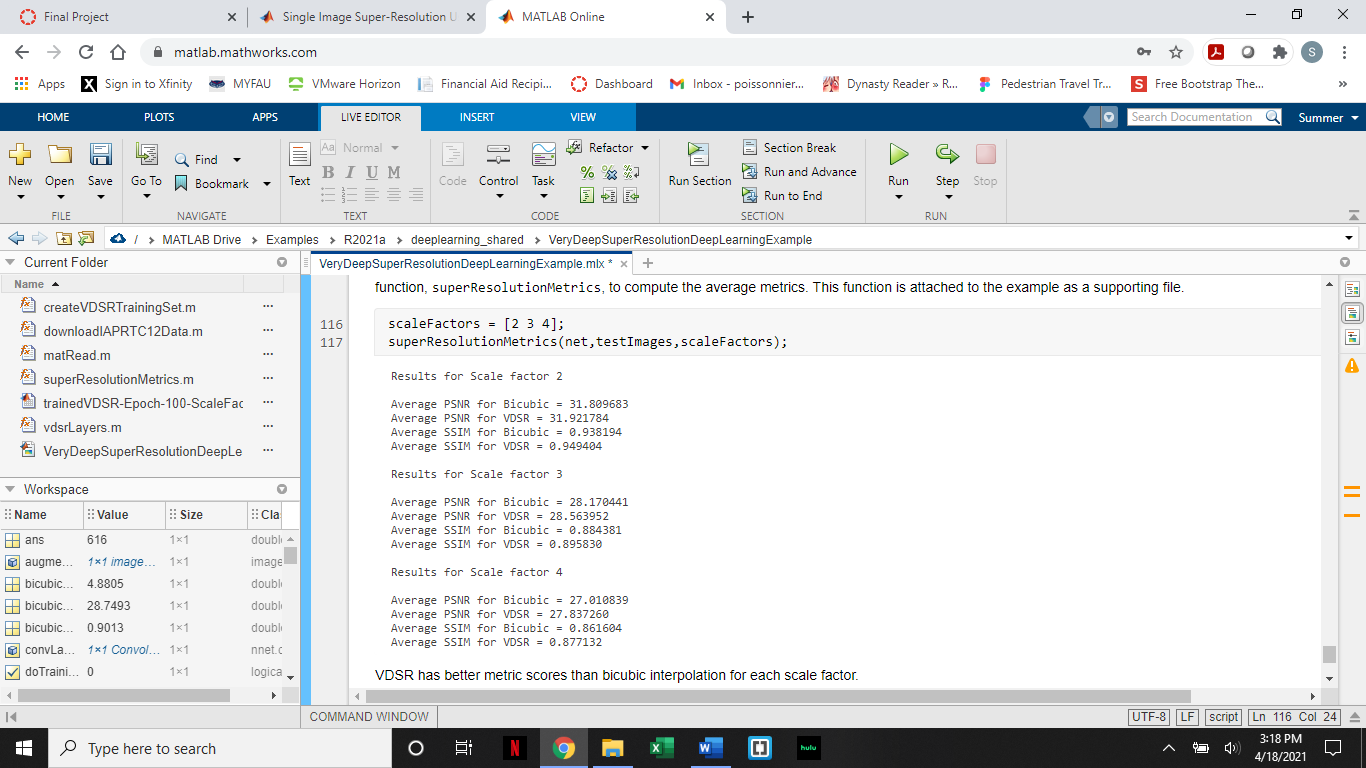
**Modification to the code**

There are two modifications to the code that we wanted to try. The first modification we made was: changing the specifications for the training options. We changed the number of epochs from 100 to 200 to allow the network more time to train. The second modification we made was: choosing a different image to test the accuracy of the resolution compared to the test image of the dog in the original code. The image we chose was: the image at index 2, the car.



Upon comparison with bicubic interpolation, we found that the quality was better for VDSR.

The image quality metrics stayed the same for the VDSR even though we increased the time the network spent training.



**Lessons Learned**

Through this example, we learned about SISR and how a network can take low resolution images and turn them into high resolution images. We also learned that some deep networks have flaws and are not 100% accurate. We learned that the SISR method was more accurate than the bicubic interpolation method from the image quality metrics that were calculated in the code.

**Ideas for the future**

If we had more time, we may have tried a different deep learning method for SISR (SISR in this example used VDSR network). We could have included a graph showing the details and accuracy of the network as we have done in previous assignments. We also may have included an alternate way besides bicubic interpolation to compare how the deep learning network works.

**Sources**

<https://arxiv.org/pdf/1501.00092.pdf>

<https://www.mathworks.com/help/deeplearning/ug/single-image-super-resolution-using-deep-learning.html?s_tid=mwa_osa_a>