ELEC 475 Lab 4

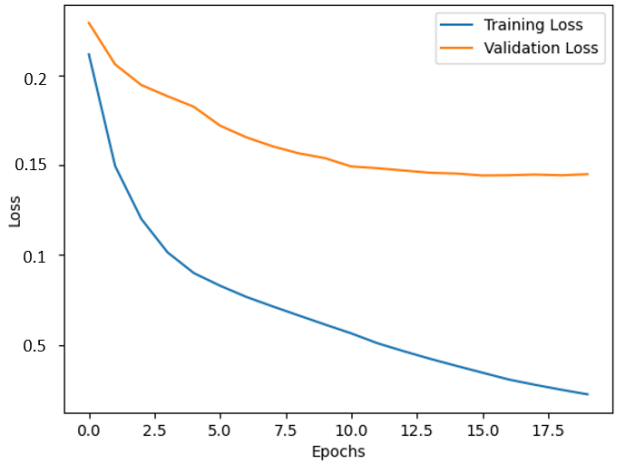
Lucas Coster

20223016

November 20th, 2023

Step 3:

In step 3 the goal was to create and train a classification model with the goal of detecting whether there was a car visible in an image or not. The classifier used was Resnet 18 which was taken from the torch vision model set. A fully connected layer was added onto the end of the model with a 512 down to 1 connection to make sure the model acted as a binary classifier. For training the optimizer used was SGD, the loss function was BCEWithLogitsLoss and the scheduler was StepLR. The model ran through 20 epochs with a batch size of 10 and a learning rate of 0.001. Both the training loss and the testing loss were plotted with the test loss being used as the validation set.



As seen in the loss plot both the training and validation loss descend gradually with the validation loss bottoming out around 0.15 and the training loss hitting a low near 0.03.

The higher validation loss shows evidence of potential overfitting and in the future to reduce this, future models would include dropout to try and minimize overfitting.

After the model had been trained it was tested on the ROI test set to calculate final accuracy. The model had a total accuracy of 95%. Several other metrics were also calculated using sklearn shown in the figure below.

The confusion matrix can be seen with true positives, true negatives, false positives, and false negatives displayed. From this information the classification report was created including values such as recall.

```
Accuracy:  0.9476845599819942
F1 score:  0.890382291390338
Confusion matrix:
[[52265  1228]
 [ 2491 15104]]
Classification report:
              precision    recall  f1-score   support

           0       0.95      0.98      0.97     53493
           1       0.92      0.86      0.89     17595

    accuracy                           0.95     71088
   macro avg       0.94      0.92      0.93     71088
weighted avg       0.95      0.95      0.95     71088
```

Step 4:

For step 4 a YODA model was built which implemented the classifier trained in step 3. It copied much of the code provided for part 2 to separate test images into their ROIs and then classify whether that image contained a car or not. If the image did contain a car, then the bounding box was added into the image to encapsulate the car. The method I implemented involved looking at each ROI individually as this was closer to how my model was trained and it did not work well with large batch input. My final method partially worked as it attempted to locate and highlight the ROIs with cars but on occasion would miss or misclassify images as seen in the examples below.

In each image cars are seen outlined with their ROIs, but several cars are also clearly missed in classification. Due to these issues the final IoU score was 0.26 out of 1.

Discussion:

The performance of my model was much lower than expected even though it had 95% accuracy on the testing set. I think this shows how difficult the visualization problem is as you are adding another layer onto an already difficult problem. I faced a few difficult challenges throughout the entire process which I had to overcome. The first had to do with training the model, specifically dealing with overfitting and bad loss numbers. To overcome this, I implemented batch normalization and experimented with several learning parameters until the loss started to look more promising. I also experimented with adding a positive weight value to the loss function due to the fact there was a much larger number of 0's within the ROI dataset than 1's and with more time I feel this change could have improved the classifier.

When creating and testing the Yoda model the main problem faced was the fact that when the ROI images were batched together classification numbers were much worse. To get around this I chose to introduce each ROI image into my model individually to best simulate the training inputs. The final output has much that could be improved but the final output was a passable Yoda model which predicts the ROI's that contains cars and outlines them in the image.