Lab 8: Self-driving Car

Author: Nigel Nelson
Course: CS-2300
Date: 2/9/20

Introduction:

This lab is an exercise in training a deep neural network using Keras, which is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. Specifically, the task of our DNN is to enable a virtual car to drive autonomously. This lab explores the impact of hyperparameters and how they effect the loss of our resulting models, as well as a dive into how dataset size effects a model. Unfortunately, due to technical issues, the created models could not be tested in the Udacity self-driving car simulator, and as such, much of the analysis of this lab is focused on the quantitative measures provided by each model's mean squared error loss plots.

The Data:

All data for this lab was formatted in the same manner. A file was used to store .jpgs that captured what the self-driving car simulator looked like from the car's perspective from the center, left, and right for a given instance. Then, a single .csv file was used as an organizer for this data where a row contained the paths to the memory location of the center, left, and right images from the simulator, and numerical values corresponding to the steering angle, brake, throttle, and speed for that same instance captured by the images.

Model 1:

- The below mean squared error loss plot corresponds to a model that was trained on the provided data set, which contains 24,108 total images, 19,286 training images, and 4,822 validation images. This model was trained **without any data augmentaiton**, and was trained for **5 epochs** with a **batch size of 32**.

Lab 8: Self-driving Car

Author: Nigel Nelson
Course: CS-2300
Date: 2/9/20

- The hyper parameters used in model 1 seem to be a good starting point for this model. The reason for this is that both measures of loss steadily decrease over the 5 epochs without an instance of overfitting happening. In addition, it is impressive that the model is capable of reaching sub 5% loss levels without any data augmentation

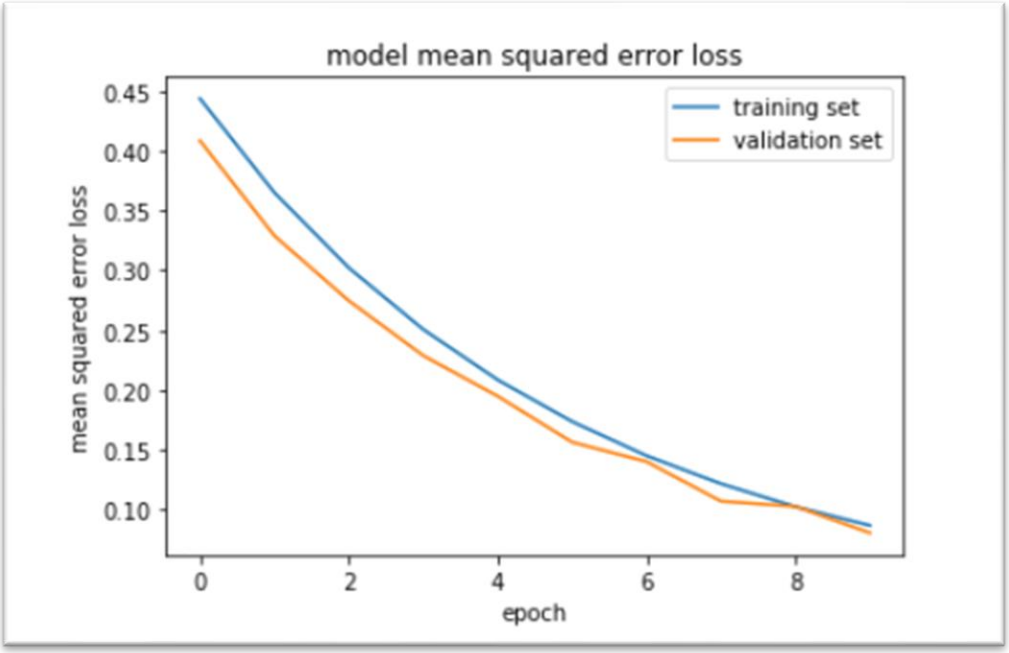
---

Model 2:

- The below mean squared error loss plot corresponds to a model that was trained on the provided data set, which contains 24,108 total images, 19,286 training images, and 4,822 validation images. This model was trained **without any data augmentaiton**, and was trained for **10 epochs** with a **batch size of 128**.

Lab 8: Self-driving Car

Author: Nigel Nelson
Course: CS-2300
Date: 2/9/20



Reflection of Model 1 vs. Model 2:

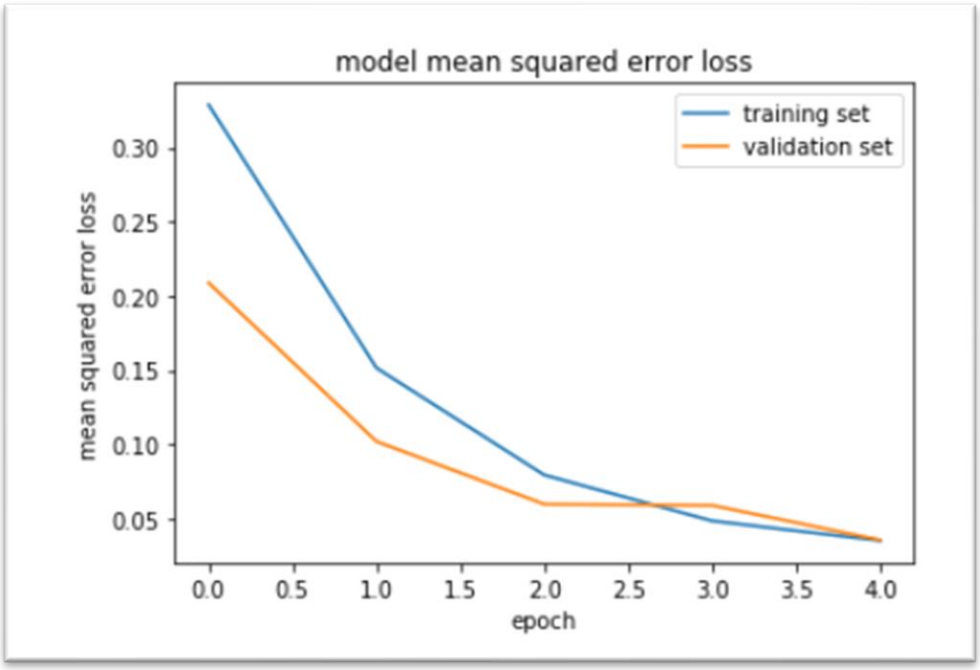
- By doubling the batch size that was used as well as doubling the number of epochs, I saw a little over a 1% increase in my training loss as well as my validation loss. This likely means that I exceeded the most effective batch size. In addition, it was observed that by doubling my batch size, the model took significantly longer to reach a similar level of loss. For example, with a batch size of 32, the model took only 2 epochs to reach 14% loss, whereas the model with a batch size of 128 took until the 6^{th} epoch to reach thee 14% loss mark. In addition, GPU usage was slightly higher as a result of doubling the batch size. The batch size of 32 saw peak GPU usage of 11% but stayed nearer to the 5% usage mark. However, the model with a batch size of 128 saw GPU usage peak at 26%, and tended to stay in that range most of the time. An observation noted during the 128 batch size model training was that GPU usage seemed to have a rhythmic use of the GPU similar to a heart beat. Where GPU usage would spike to 14%, then drop to 0% for a moment or two, and continually repeat this cycle.

Model 3:

Lab 8: Self-driving Car

Author: Nigel Nelson
Course: CS-2300
Date: 2/9/20

- The below mean squared error loss plot corresponds to a model that was trained on the provided data set, which contains 24,108 total images, 19,286 training images, and 4,822 validation images. This model was trained with a data augmention of images being **flipped about the y-axis** apllied, and was trained for **5 epochs** with a **batch size of 32**.



- Model 3 shows that by flipping the images the model saw a slight improvement in both the training loss as well as the validation loss. This is likely due to the fact that from a car's perspective, when it is going down a single lane road, it doesn't matter if its perspective is flipped about the y-axis. The reason for this is that if the car sees the road bends left, then the steering needs to be adjusted by the same absolute value as for if the road bent to the right. As such, this data augmentation allows the car to have a more generic understanding of how to control a car that is less dependent on the exact way that the data is presented to the car. However, the mean squared error loss plot raises a slight reason for concern as it is seen at the beginning of epoch 3 that overfitting begins to occur, where the validation loss is greater than the training loss. This is a sign that the model is simply beginning to memorize the training data. Yet, by epoch 4 the two losses converge, and the issue appears to be remedied.

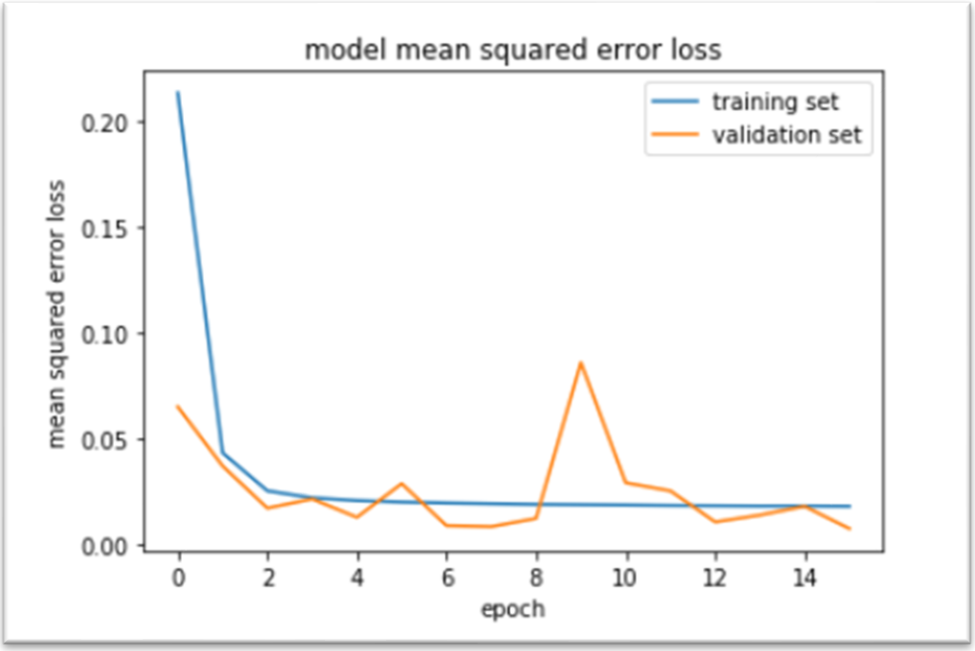
Model 4:

- The below mean squared error loss plot corresponds to a model that was trained on the provided data set, which contains 24,108 total images, 19,286 training images, and 4,822

Lab 8: Self-driving Car

Author: Nigel Nelson
Course: CS-2300
Date: 2/9/20

validation images. This model was trained with a data augmention of images being **flipped about the y-axis** apllied, and was trained for **16 epochs** with a **batch size of 8**.



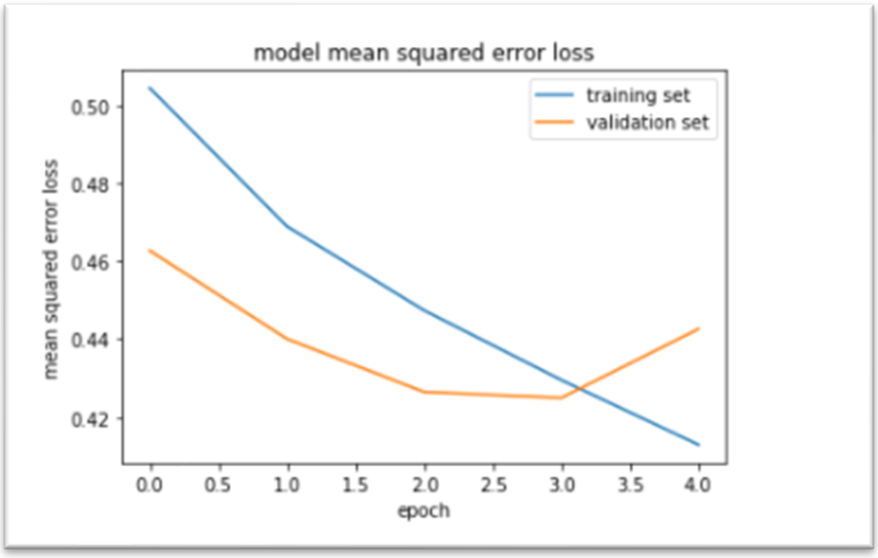
- Model 4 represents the lowest loss model that was created in this lab. The batch size was lowered so that weights were adjusted more precisely, and the model was trained for 16 epochs to ensure a higher learning rate was used for a greater amount of data, as well as allowing the model to see more data to increase its accuracy. While overfitting occurred at the 9th epoch, this overfitting was ultimately resolved in training by the 12th epoch and the final model does not exhibit overfitting. An interesting observation from the training of this model is that while GPU usage peaked at 26% usage in the first couple epochs, GPU usage noticeably fell in later epochs and rarely rose above 15% usage in the final epochs. This is likely due to the fact that smaller and smaller changes need to be applied to the model in later epochs.

Model 5:

- The below mean squared error loss plot corresponds to a model that was trained on data collected from a short driving session, which contains 390 total images, 312 training images, and 78 validation images. This model was trained with a data augmention of images being **flipped about the y-axis** apllied, and was trained for **5 epochs** with a **batch size of 32**.

Lab 8: Self-driving Car

Author: Nigel Nelson
Course: CS-2300
Date: 2/9/20



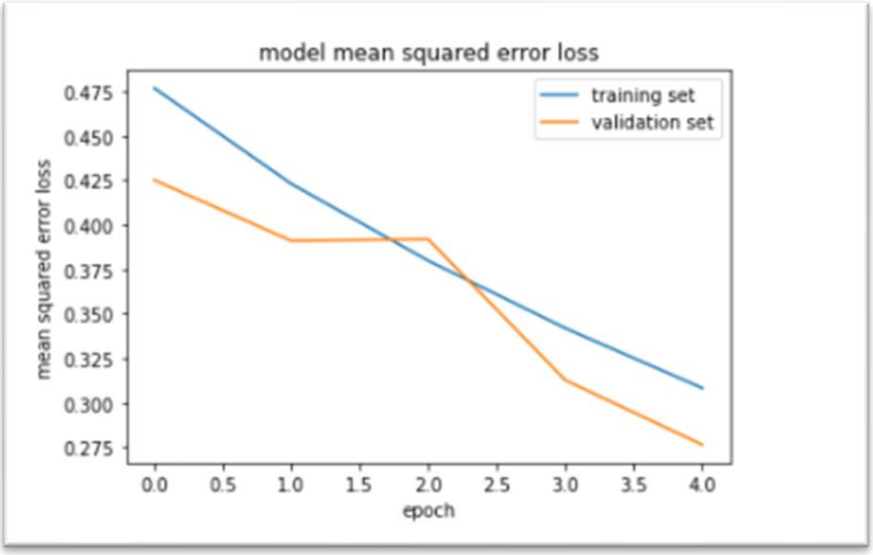
- From observing model 5, one of the first things noted when creating a model based on data collected from a short driving session is the speed at which the model is created. This comes as no surprise as the total number of images is 390, with 312 training images and 78 validation images. While the data provided for the previous models created has 24,108 total images, 19,286 training images, and 4,822 validation images. So, through testing it was seen that a single epoch with a batch size of 32 took approximately 1 second for the short driving session, while for the same hyperparameters on the provided dataset, this operation took 1:30 minutes. A second observation that is immediately apparent is that overfitting begins to take place near the 3rd epoch. This can be seen because the validation loss begins to increase in comparison to the training loss. This is an indicator that the model is beginning to simply memorize the training data. Even though this smaller test used flipping about the y-axis as a data augmention technique, this result does not come at much of a surprise because of how small the training data is. Another unique characteristic of this plot in comparison to the others is the fact that it has the highest starting training and validation losses. This is also likely a side effect of having minimal training data and validation data.

Model 6:

- The below mean squared error loss plot corresponds to a model that was trained on data collected from a longer driving session, which contains 3,555 total images, 2,844 training images, and 711 validation images. This model was trained with a data augmention of images being **flipped about the y-axis** apllied, and was trained for **5 epochs** with a **batch size of 32**.

Lab 8: Self-driving Car

Author: Nigel Nelson
Course: CS-2300
Date: 2/9/20



- Model 6 was created using a larger driving session. Specifically, this model was trained in 5 epochs, with batch sizes of 32, and on 3,555 total images, 2,844 training images, and 711 validation images, with flipping about the y-axis used as a data augmentation technique. While this data set is significantly larger than the previously created data set, it is still noticeable smaller than the provided data set which has 24,108 total images. As a result the loss is predicable, where the loss is improved over the smaller data set but not as low as the provided data set. In additon, besides a glimpse of overfitting at epoch 2 that is quickly remedied, no serious overfitting takes place and the model to seemed to create a general solution to the driving problem and has room to grow with more epochs. A interesting observation is that with this data set being approximately 10 times larger than the smaller created data set, the times for a single epoch to be completed for a batch size of 32 is about 10 times as slow as the smaller data set, with a time of about 11 seconds per epoch. Another interesting observation of this plot is that in comparison to the plots of the supplied data models, the training loss and validation loss seems to have a consitently greater degreee of separation. This trend means that either the model is comparitevely better at making a generalized solution given the training data, or more likely, the validation data set for the created data is comparitively easier than the larger provided validation data set.

Reflection on Benchmarking of Training:

Using a line by line profiler, it was found that when training the model, that the following operation took the vast majority of time for the Jupyter Notebook cell that created and trained the model:

Lab 8: Self-driving Car

Author: Nigel Nelson
Course: CS-2300
Date: 2/9/20

```
# Train the model
history_object = model.fit_generator(train_generator, steps_per_epoch= \
    len(train_samples)/my_batch_size, validation_data=validation_generator, \
    validation_steps=len(validation_samples)/my_batch_size, epochs=my_epochs, verbose=1)
```

The reason for this is that this operation is responsible for the training of the model. This is often the most computationally expensive portion of creating a model because the back propagation of the model, where the weights of the network are adjusted according to each batch, takes a significant amount of time when compared to the other processes involved in creating a model.

Refection on Training Time vs. Amount of Data Collected:

By using several datasets for the creation of models, it was observed that the relationship between the training time and the amount of data collected was proportional. The graph below displays the training time for the first epoch of data vs. the total number of images in the data set, when the batch size is set to 32 and all other hyper parameters are equivalent.

From this graph it can be observed that there is a linear, or proportional relationship between the dataset size and the time to train the model. This is an indicator that the mathematical functions used in this instance of back propagation are linear in nature, and that for this given training

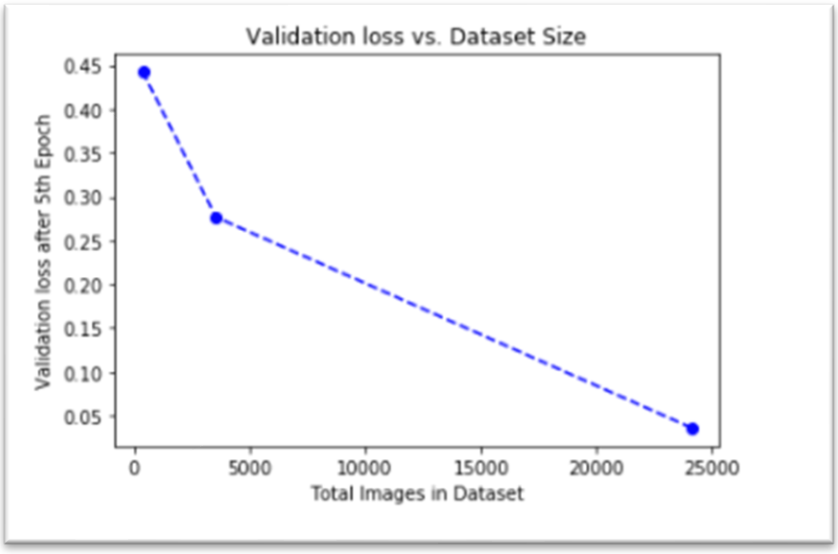
Lab 8: Self-driving Car

Author: Nigel Nelson
Course: CS-2300
Date: 2/9/20

approach, predictions for the training time can be made for various dataset sizes given just a single benchmarked model.

Refection on Loss vs. Amount of Data Collected:

By using several size datasets in the training of models a relationship was observed between the reduction in loss and the number of total images in the dataset. The graph below shows the relationship between the validation loss after the 5th epoch of training vs. the total images in the dataset when all hyper parameters are equivalent.



As seen above, there is not a clear cut proportional relationship between the size of the dataset used and the reduction in loss. However, it must be noted that each model created will vary in the amount of loss that it achieves, so while this graph doesn't show a direct proportional relationship, it is possible that different iterations of these exact models would show a proportional relationship. The most this graph tells us is that as the size of the data set grows, the model that results has a greater reduction in loss. Ultimately, many more data sets sizes are needed to make a clear conclusion on the exact relationship between these two quantitative measures.

Reflection on overfitting:

Lab 8: Self-driving Car

Author: Nigel Nelson
Course: CS-2300
Date: 2/9/20

Overall, for the majority of models created in this lab, overfitting was rarely prevalent. The reason for this is likely that the amount of data used was sufficiently large for the number of epochs ran, the batch sizes that were used, and the resistance to overfitting that flipping the images on the y-axis provided. However, the worst case of overfitting that occurred in this lab was by far the model that was created from the short driving session. The reason for this is that there simply was not enough data for the model to be able to create an accurate generalized solution. As soon as the model surpassed the 3rd epoch, overfitting began to occur, and no matter what hyper parameters or image flipping was used it was difficult to see loss below the 40% mark without overfitting occurring. The most common overfitting that occurred throughout the lab was brief stints where the training loss would comparatively dip compared to the validation loss. Yet, most all of the models quickly remedied this overfitting and would often return to a non-overfitting model within a single epoch. However, this likely means that the models that experienced this were operating near the edge of their learning rate and batch size limits for the available data augmentations.

Conclusion:

This lab was an exercise in training a deep neural network using Keras, which is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. Specifically, the task of our DNN is to enable a virtual car to drive autonomously. Through experimentation in this lab several discoveries were made. First it was noted that increasing batch size decreases precision, and requires more epochs to reach a similar measure of loss. In addition, it was found that for the problem of creating a model for a self driving car, flipping the training images about the y-axis is a useful data augmentation to lower loss as well and reduce the chance of overfitting. Furthermore, it was noted that the amount of data used to train a model is directly proportional with the amount of time required to train that model. Lastly, it was observed that the amount of data used to train a model has a loosely proportional relationship with the amount of loss that a given model is capable of achieving.