**Lab 10: Multi-GPU Scaling**

Stuart Harley

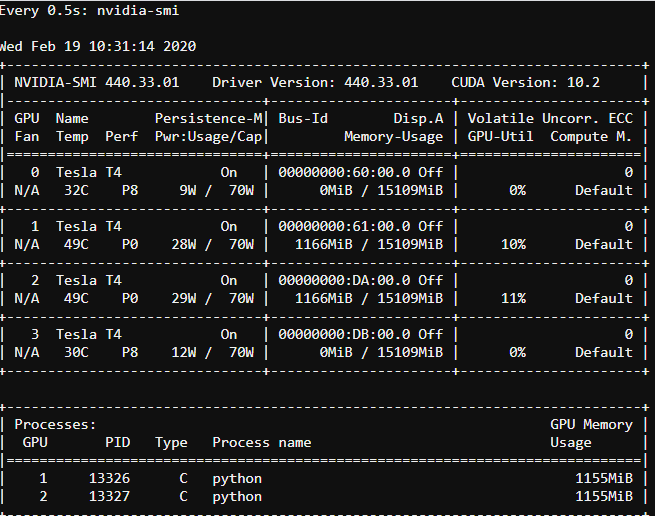
**Introduction:** In this lab I will be using multiple GPUs to train my self-driving car model to explore scaling with multiple GPUs. I am training all of the models in the lab with a batch size of 32 run for 10 epochs.

**Learning Outcomes:**

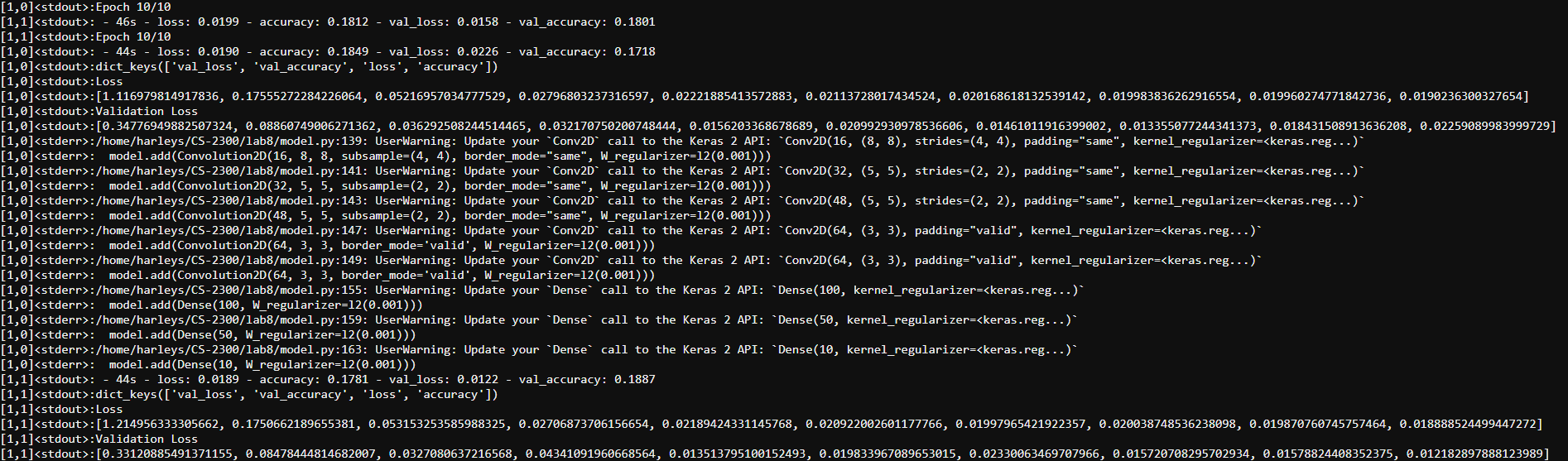
* Multi-GPU training understanding
* Usage of Horovod
* Scaling projections

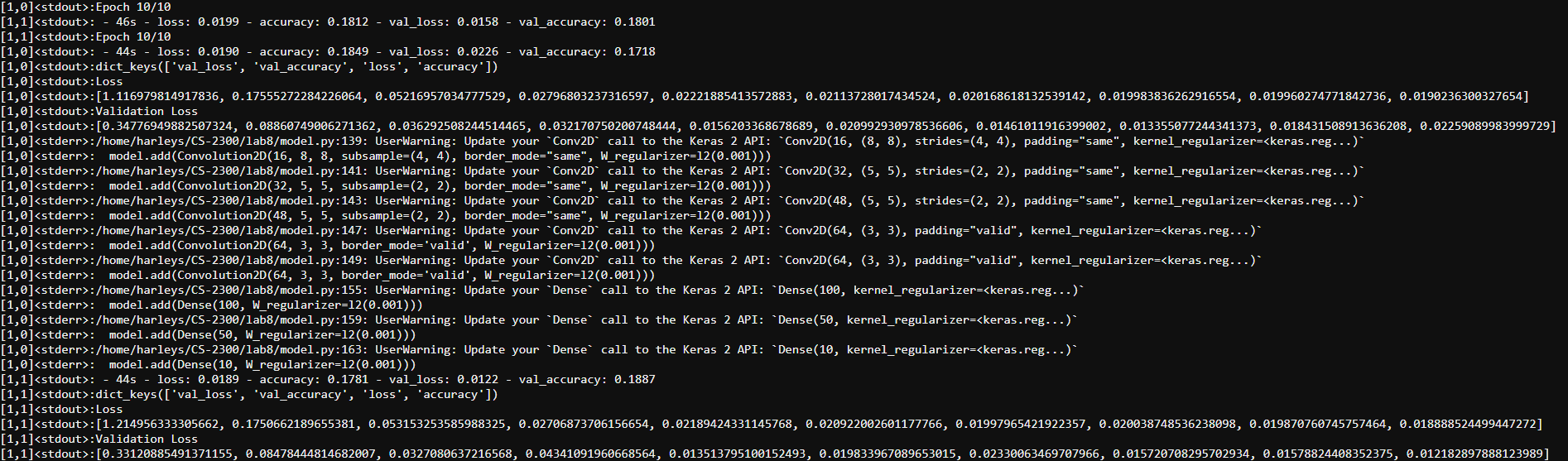
**Running on 2 GPUs**

Shown: Screenshot of GPU utilization



Shown: Loss output from running





It took about 8 minutes to train the model. The loss for both GPUs is shown above. It is about .019 for both.

The rest of the jobs ran on their own, so I do not have GPU utilization screenshots for them.

**Running on 4 interactive (T4) GPUs**

It took about 7.5 minutes to train the model. The loss across the 4 GPUs was about .046.

**Running on 4 DGX GPUs**

It took about 8 minutes to train the model. The loss across the 4 DGX GPUs was about .047.

**Running on 8 DGX GPUs**

It took about 8.5 minutes to train the model. The loss across the 8 DGX GPUs was about .8.

**Shown:** Plot of the training times for the models

**Shown:** Plot of the losses for the models

**Conclusion:** Our dataset is not large enough to utilize multiple GPUs effectively. It is very much so overkill for a dataset of this size. And since using more GPUs slows convergence, really the only thing I found from running these jobs is that the more GPUs that were used, the larger the loss was. There were also minimal differences in running time between the models. This is because the averaging of the model weights between the GPUS after each epoch, offsets the time gains from splitting up the dataset between the different GPUs. Again however, our dataset is too small to effectively use multiple GPUs. if our dataset was larger, we would most likely see time improvements.