# Machine Learning Assignment

Neeraj Sathyan (12938262)

February 14, 2020

**1.** 1 GPU is being utilized with about 20% utilization. Percentage Utilization can be increased by increasing the batch size of training dataset.
Batch of 128: 20%
Batch of 1024: 54%

**2.** Currently its at 21 epochs. In order to increase the epochs count to 40 with a batch size of 128, we would require 18750 steps. The number of steps is shown as:

```
Steps = (epoch * training_set_size)/(batch_size)
```

Below is the following results with 40 epochs:

```
Test loss: 15.59079650952071
Test accuracy: 0.9854
Training time: 456.1468 seconds
```

Thus the time required for a full training run on a Lisa GPU node for 40 epochs is 456.1468 seconds.

**3.** Running in the original format of 21 epochs.

1 GPU:
```
Test accuracy: 0.9854
```

2 GPUs:
```
Test accuracy: 0.9922
```

4 GPUs:
```
Test accuracy: 0.992
```

Running with 40 epochs.

1 GPU:
```
Test accuracy: 0.9827
```

2 GPUs:
```
Test accuracy: 0.9862
```

4 GPUs:
```
Test accuracy: 0.9903
```

```
40 epochs
4 GPUs:
Test accuracy: 0.9903
Training time: 203.63444709777832 seconds

2 GPUs:
Test accuracy: 0.9862
Training time: 379.69138598442078 seconds


1 GPU:
Test accuracy: 0.9827
Training time: 456.14687323570251 seconds
```

*Scaling efficiency*:
2 GPU setup is 0.53 times faster than 1 GPU setup.
4 GPU setup is 0.84 times faster than 2 GPU setup.
This measurement was made for 40 epochs.

*Bottlenecks*: When distributing the work across GPU's, there is a chance that one of the GPU has a higher GPU utilization percentage as compared to others. This means there is a chance of under-utilization of GPUs. Also there can be chances when memory of one of the GPUs is highly utilized compared to the others. When running dependent programs, the GPUs cannot act in parallel as they need to wait for the other GPU to complete. This will lead to an increase in running time for the process.

**5.We can also scale to multiple workers, then we have to change the run script, to select multiple nodes, and change the -np flag, which are the total number of GPU's. Let's scale up to 2 workers. What happens to the batch size and learning rate? How can we keep the validation accuracy high when scaling up the batch size?**

Two nodes are allocated using the sbatch slurm command as: #SBATCH -N 2 The horovod code ran for a bathc size of 128 and 40 epochs and gave the following readings:

```
GPUs: 2
Epoch: 40
Nodes: 2
Training time: 152.0235 seconds
Steps: 3120
Accuracy: 0.9906
```

Batch size does not vary and is specified same as to the two nodes. The entire batches of size 128 is not equally divided and shared among the nodes.

A larger batch size leads to a larger learning rate as hvd size depends on GPUs per nodes. The learning rate is given as:

```
0.001 * hvd.size()
```

hvd.size() is the GPU count in node. Hence as GPUs are scaled, the learning rate increases. In a two node setup, the batch size is doubled (128 * 2). Learning rate depends on the number of GPUs in each node. A very high batch size will impact the accuracy as it reduces the stochasticity of the gradient descent. To maintain the validation accuracy, learning rate have to be increased as the batch size increases.

**6. What is the best validation accuracy that you have achieved using 2 workers, and the same 40 training epochs?** The following configuration gives the best validation accuracy:

```
No of GPUs reserved in total among the 2 worker nodes: 6
```

The following configuration gave the best result among the data given below for other configurations:

```
Worker Nodes: 2
GPUs: 2
Accuracy: 0.9902

Worker Nodes: 2
GPUs: 6
Accuracy: 0.9905

Worker Nodes: 2
GPUs: 8 (Node GPUs filled)
Accuracy: 0.9879
```