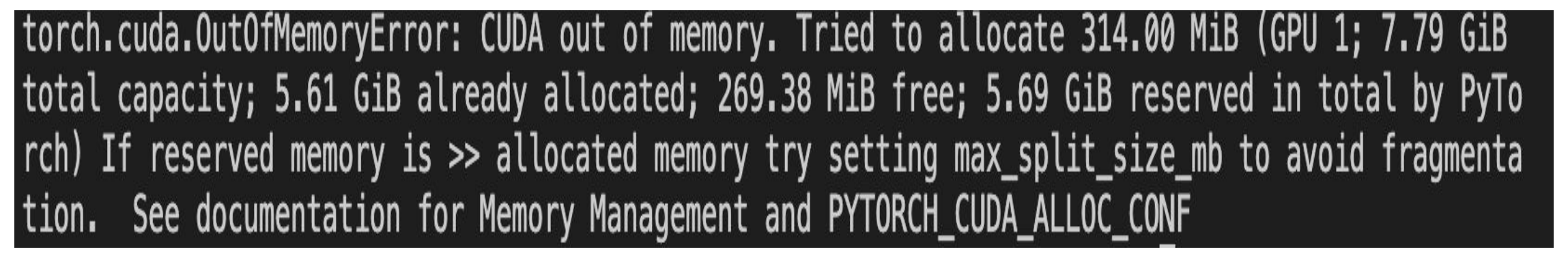
Multi GPUs Training

Yelin Cheon
CSE 310
yelin.cheon@stonybrook.edu

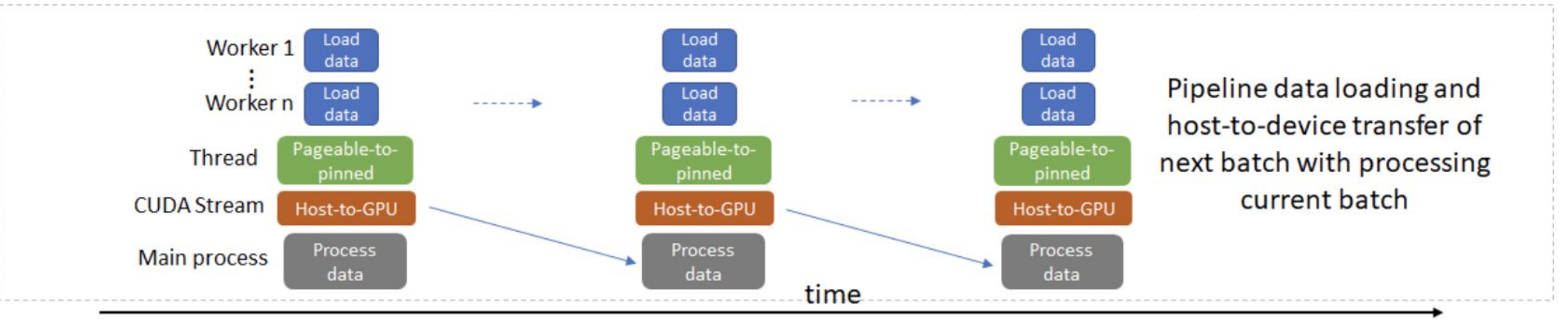
Motivations



```
torch.cuda.OutOfMemoryError: CUDA out of memory. Tried to allocate 314.00 MiB (GPU 1; 7.79 GiB
total capacity; 5.61 GiB already allocated; 269.38 MiB free; 5.69 GiB reserved in total by PyTo
rch) If reserved memory is >> allocated memory try setting max_split_size_mb to avoid fragmenta
tion.  See documentation for Memory Management and PYTORCH_CUDA_ALLOC_CONF
```

Figure 1: GPU Memory Error

- The GPU Memory Error occurs when train the deep and wide model with single gpu.



Pipeline data loading and host-to-device transfer of next batch with processing current batch

- Solution: Distribute work into multi GPUs

Concept

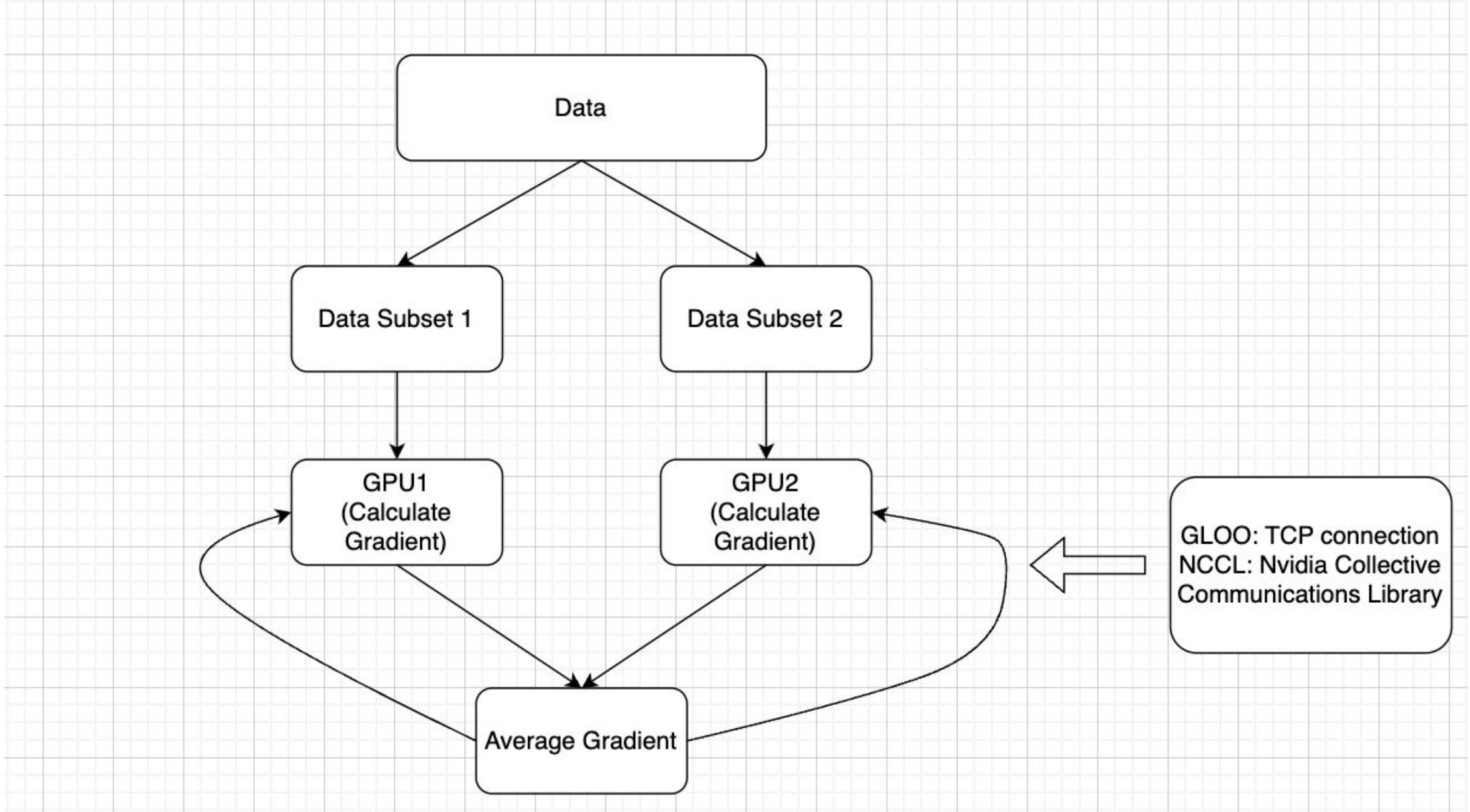

Data
Data Subset 1
Data Subset 2
GPU1 (Calculate Gradient)
GPU2 (Calculate Gradient)
Average Gradient
GLOO: TCP connection
NCCL: Nvidia Collective Communications Library

Figure 2: Concept of Distributed Data Parallel

Experiment

1. Check if the memory error occurs with deep and wide model or not
2. Compare the performance and train time between single GPU and Multi GPUs (The number of GPUs is 2.)
3. Compare the performance and train time with small vit model between single gpu and multi gpu
4. Compare the performance between tcp connection and NCCL.
5. Observe transported data byte

Result

Experiment 1- Deep and Wide Model

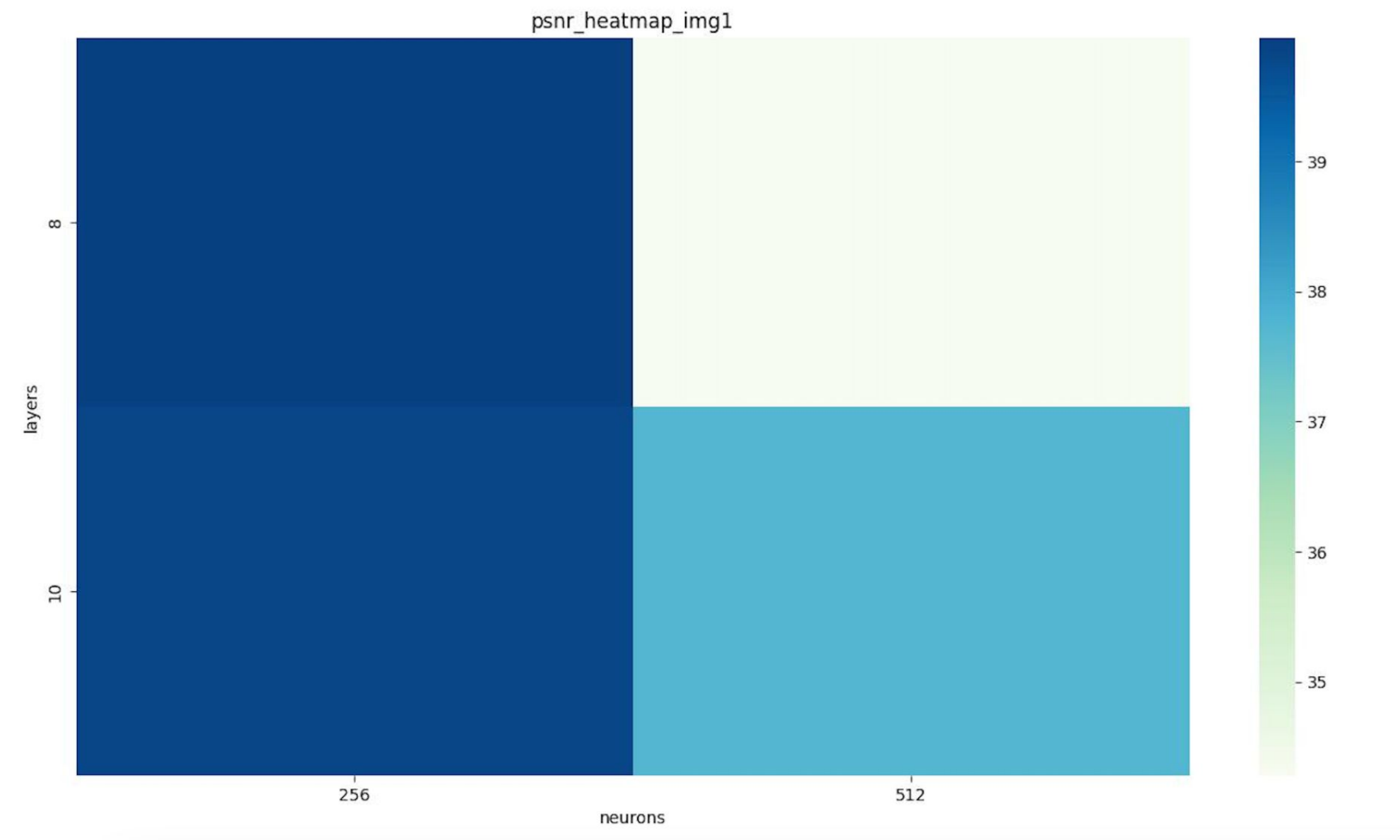


Figure 3: psnr heatmap

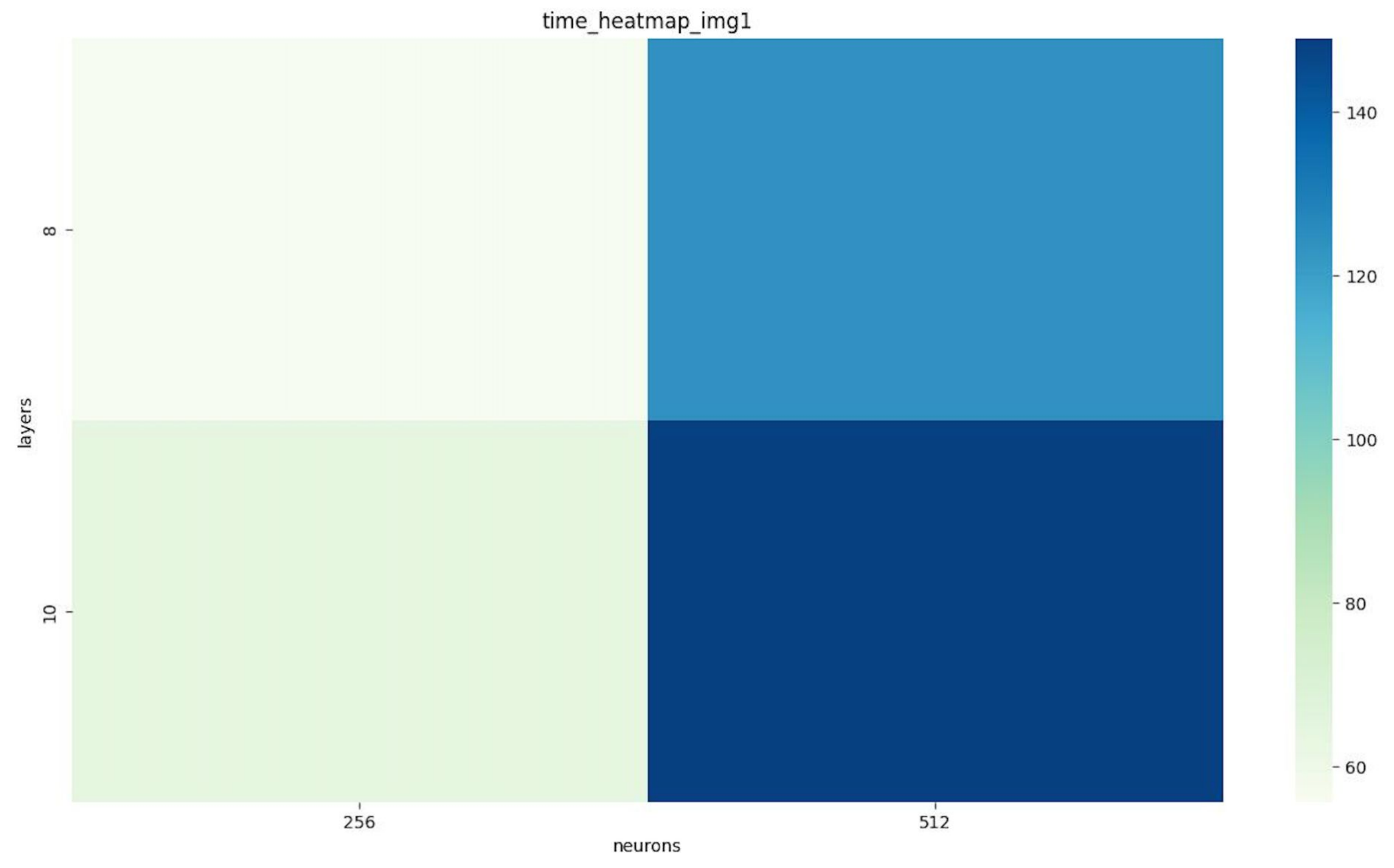
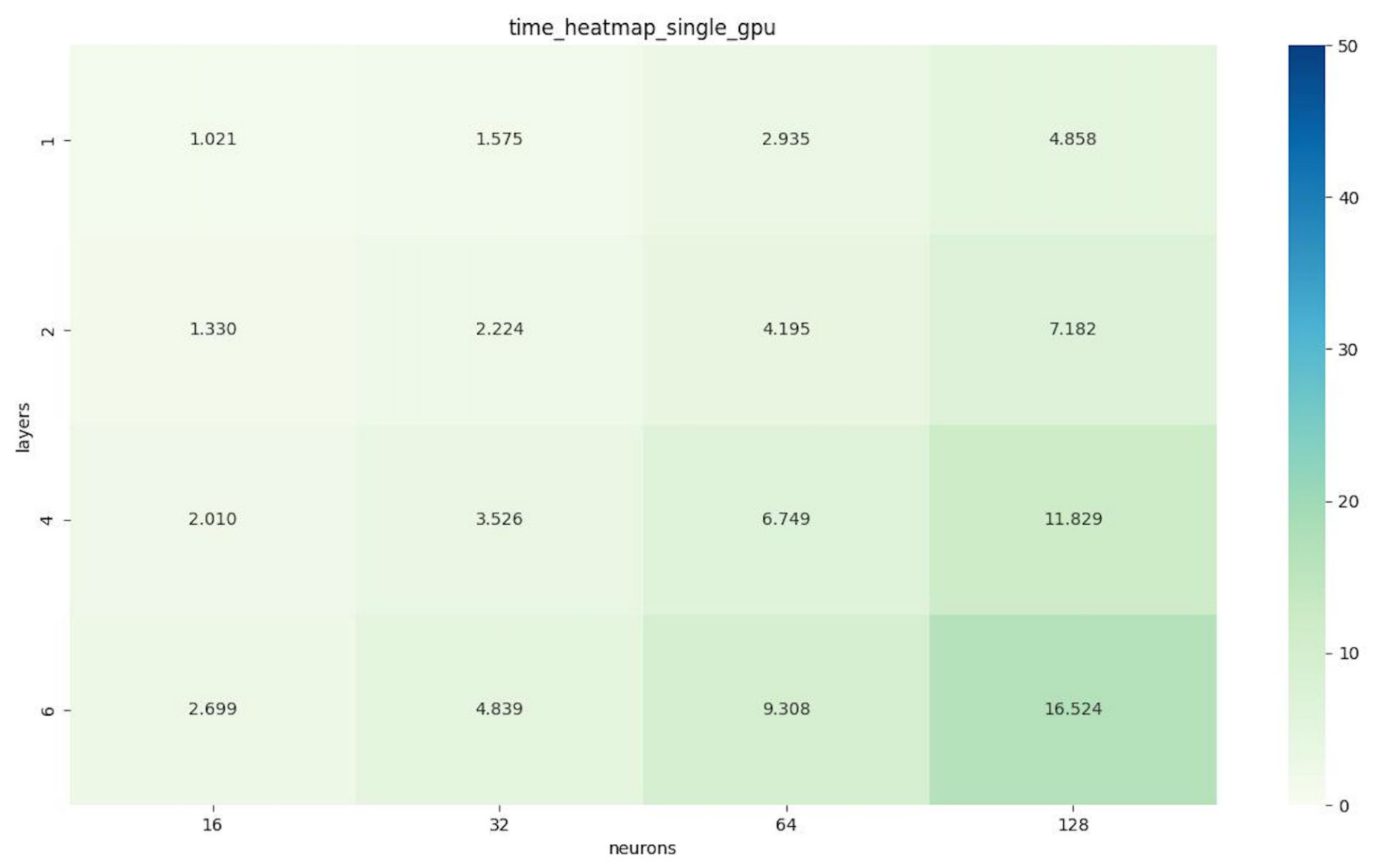


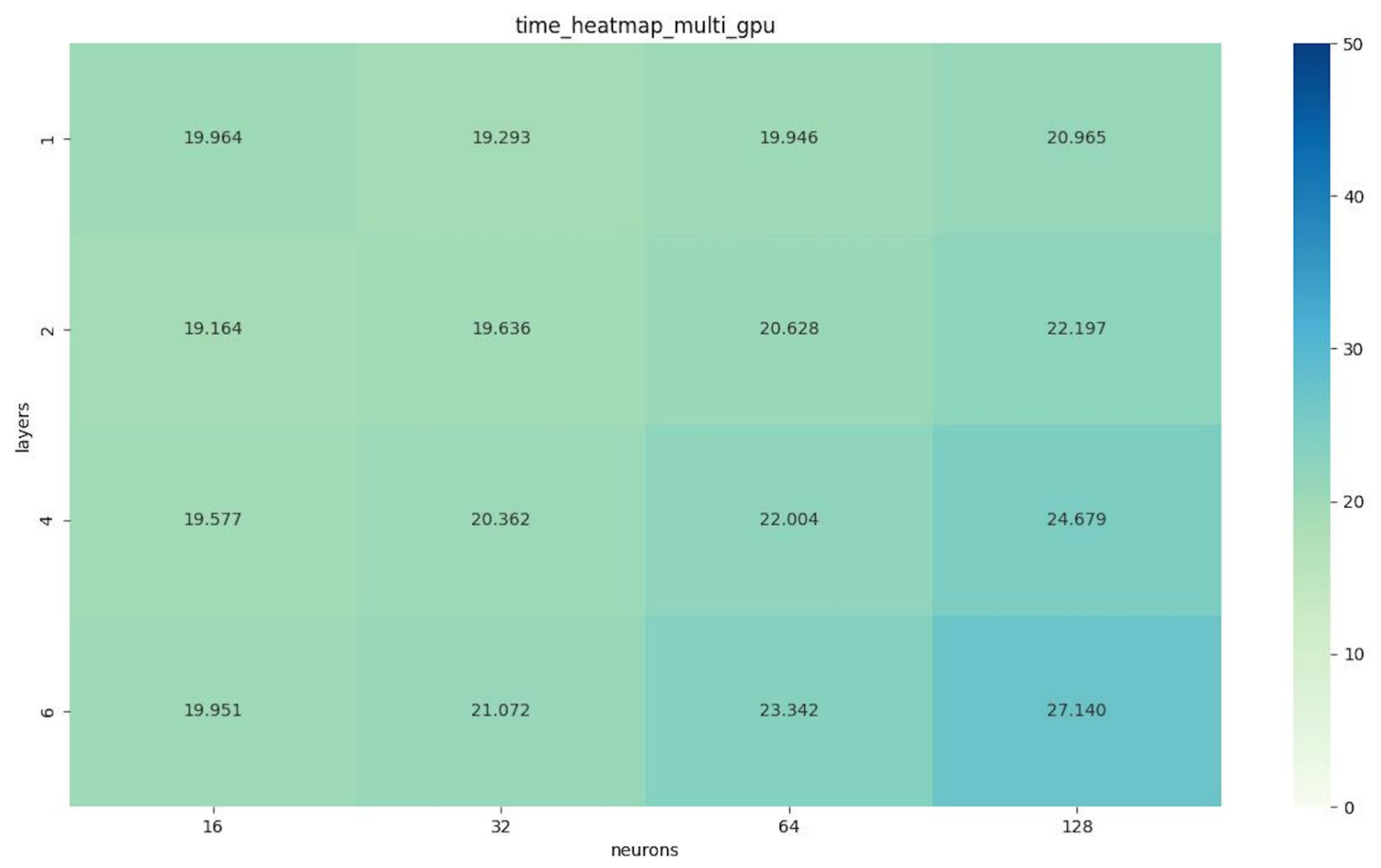
Figure 4: time heatmap

Experiment 2 - Compare the train time between single gpu and multi(2) gpu



layers \ neurons	16	32	64	128
1	1.021	1.575	2.935	4.858
2	1.330	2.224	4.195	7.182
4	2.010	3.526	6.749	11.829
6	2.699	4.839	9.308	16.524

Figure 5: time heatmap with single gpu



layers \ neurons	16	32	64	128
1	19.964	19.293	19.946	20.965
2	19.164	19.636	20.628	22.197
4	19.577	20.362	22.004	24.679
6	19.951	21.072	23.342	27.140

Figure 6: time heatmap with multi gpu

- training time with single GPU < training time with multi GPUs

Experiment 3 - Compare the time and performance of the small vit model between single gpu and multi(2) gpu(connection: "TCP(gloo)")

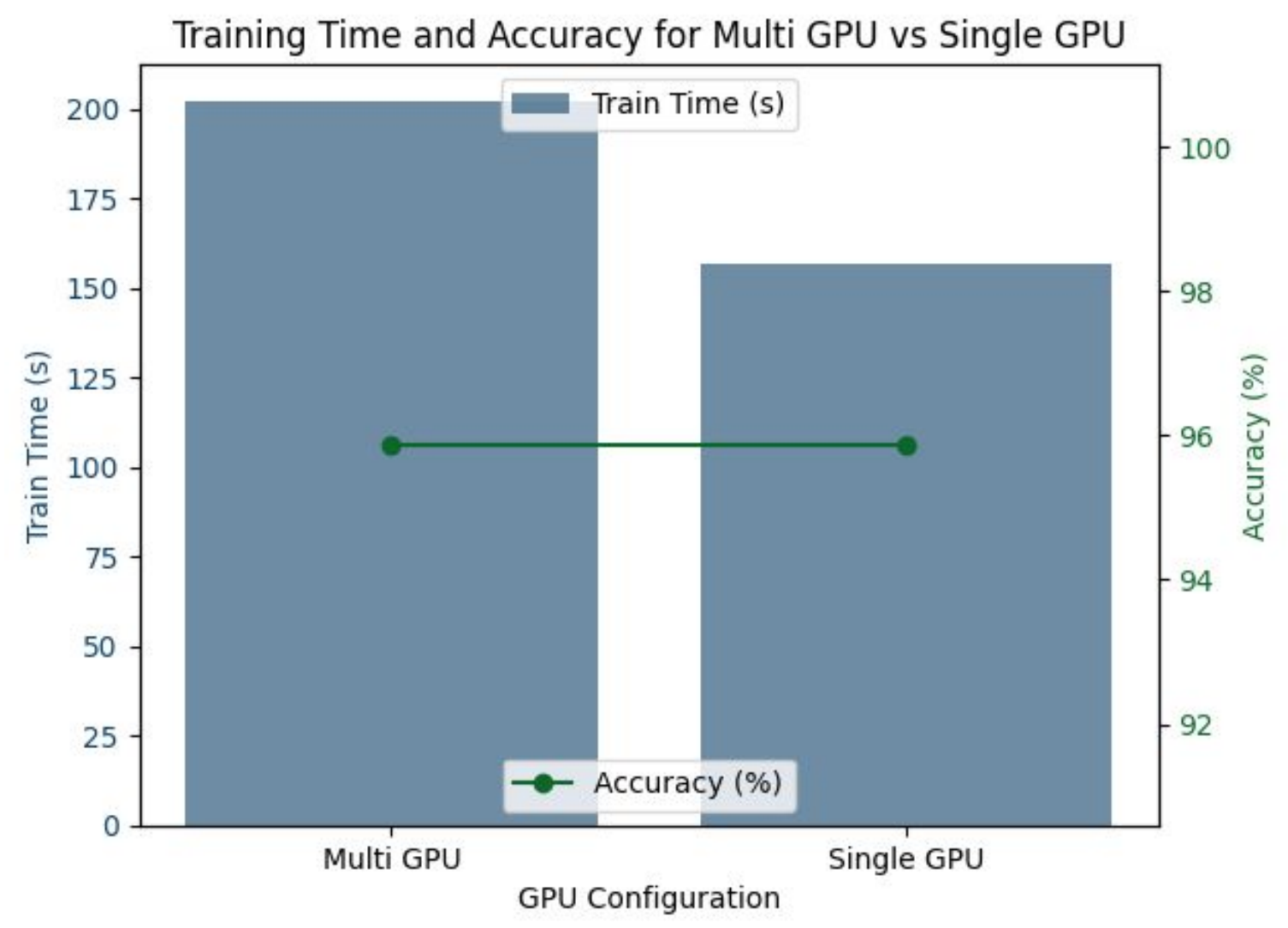


Figure 7: Accuracy and Train time between multi / single GPU

Experiment 4 - Compare the performance and train time between tcp connection and NCCL

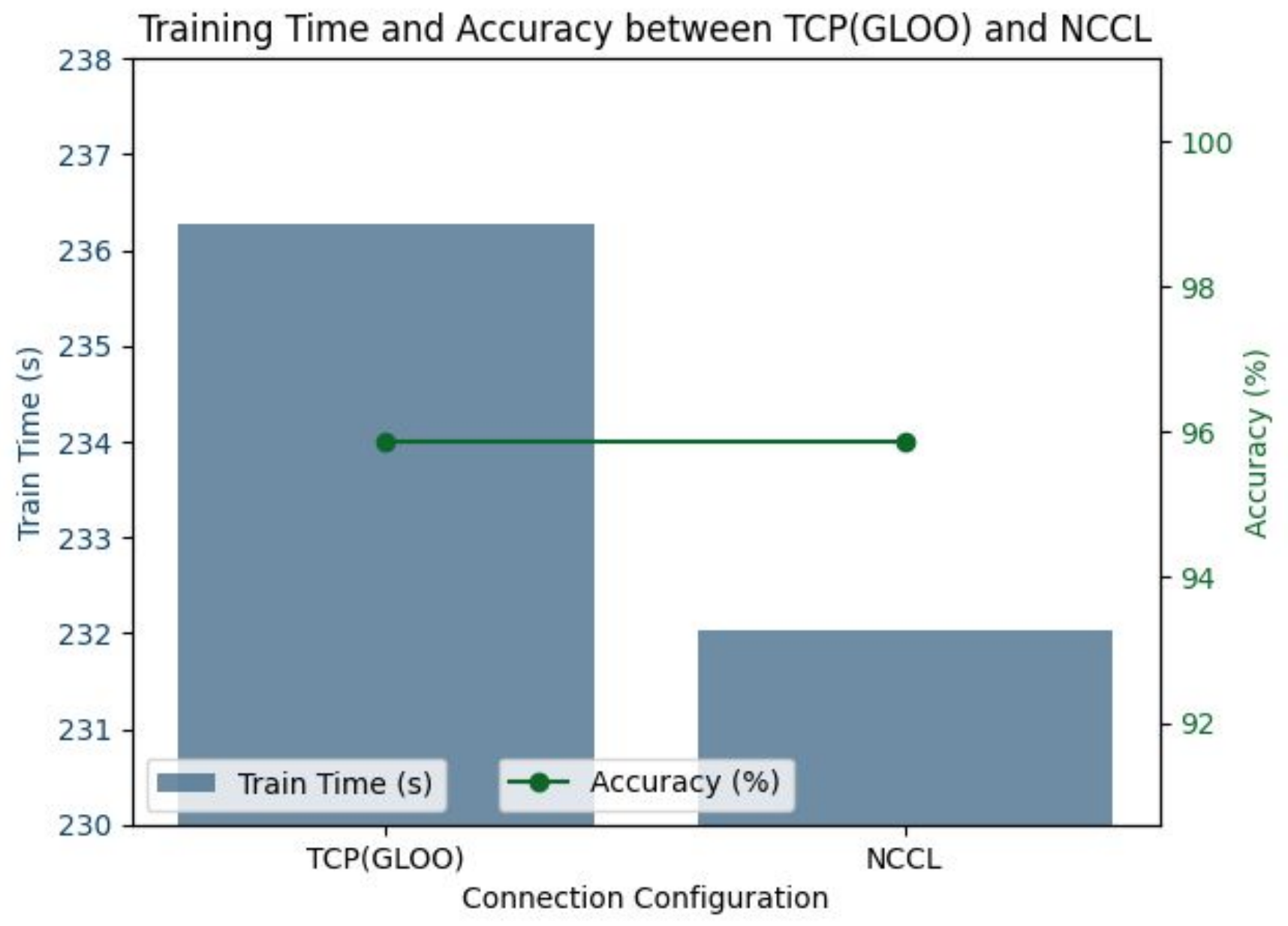
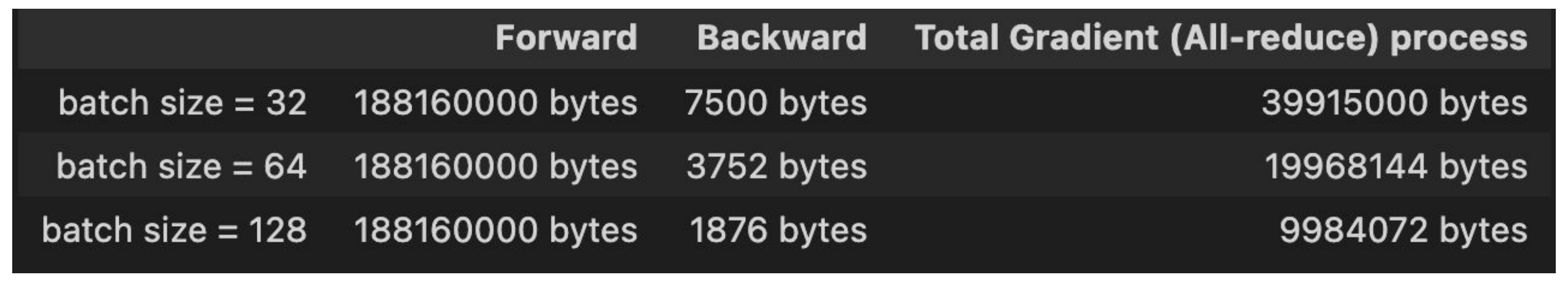


Figure 8: Accuracy and Train time between TCP/NCCL

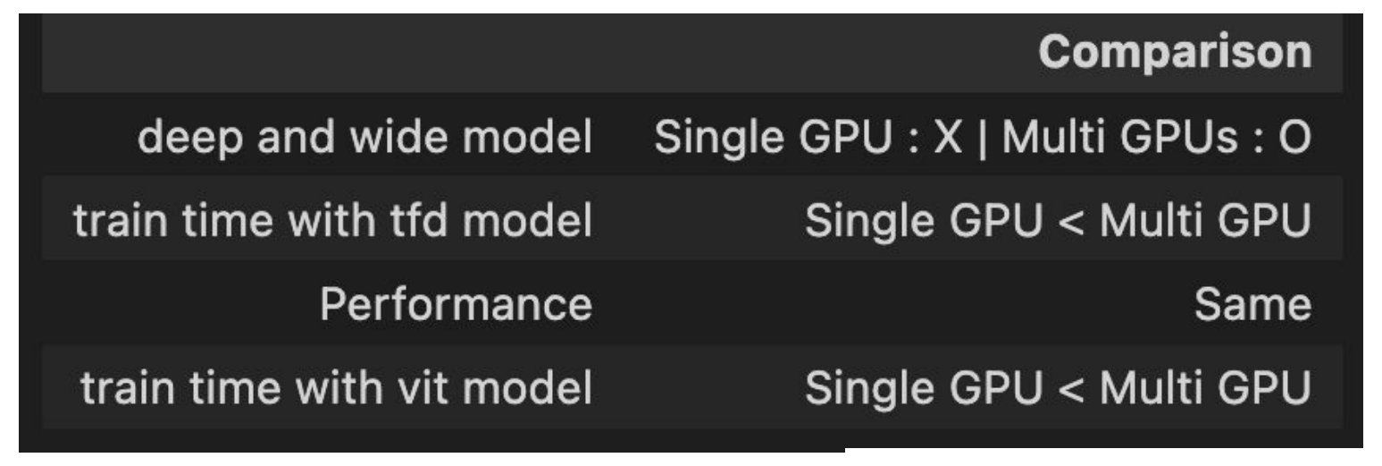
Experiment 5- Observe transported data byte



	Forward	Backward	Total Gradient (All-reduce) process
batch size = 32	188160000 bytes	7500 bytes	39915000 bytes
batch size = 64	188160000 bytes	3752 bytes	19968144 bytes
batch size = 128	188160000 bytes	1876 bytes	9984072 bytes

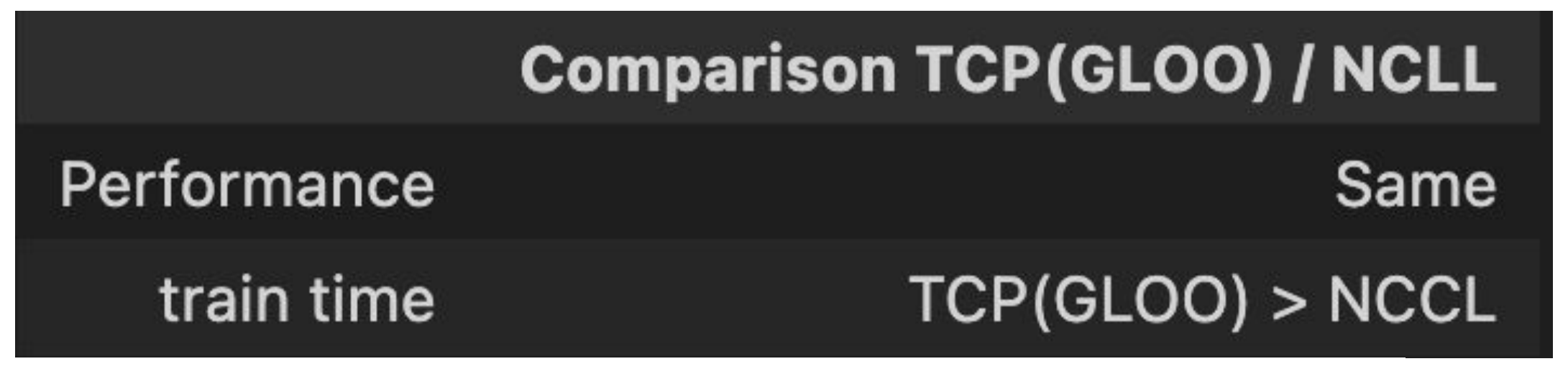
Table 1: compare transported data depending on batch size

Conclusion



	Comparison
deep and wide model	Single GPU : X | Multi GPUs : O
train time with tfd model	Single GPU < Multi GPU
Performance	Same
train time with vit model	Single GPU < Multi GPU

Table 2: compare single GPU and Multi GPU



	Comparison TCP(GLOO) / NCLL
Performance	Same
train time	TCP(GLOO) > NCCL

Table 3: compare TCP(GLOO) and NCLL

1. Multi GPUs can solve the GPU memory error in the deep and wide model.
2. Using a single GPU is faster than using multi GPUs in the tiny model.
3. When comparing the PSNR(Peak Signal-to-Noise Ratio) , using a single GPU and Multi GPUs has the same performance.
4. Using single GPU is faster than using multi GPUs in the VIT Model.
5. When comparing TCP (GlOO) and NCCL, performance is the same, but training time with TCP is slower than NCCL.
6. In the VIT model, transported data in forward propagation is quite large, and transported data in backward propagation and the all-reduce process changes depending on the batch size.

Reference

PyTorch. "DistributedDataParallel." PyTorch Documentation. https://pytorch.org/docs/stable/generated/torch.nn.parallel.DistributedDataParallel.html#torch.nn.parallel.DistributedDataParallel
Telesens. "Distributed Data Parallel Training using PyTorch on AWS." Telesens, 4 April 2019. https://www.telesens.co/2019/04/04/distributed-data-parallel-training-using-pytorch-on-aws/