EECE 5640 High Performance Computing

Final Project Report

Lianrui YANG, Quanjiang LONG, Xiangyu ZENG

M.S. in ECE (2022)

Spring, 2022

Department of Electrical and Computer Engineering

Northeastern University

Table of Contents

I. Introduction

1.1 Project Aims

We will evaluate the performance of engineering workloads on Discovery platform. We propose to use Tensorflow, PyTorch that need GPU acceleration to evaluate executing speedup on different Discovery nodes and hardwares. Furthermore, because there are some differences between TensorFlow and PyTorch, we want to evaluate the efficiency of these two packages. There are 3 models that will be trained by 2 datasets on both Pytorch and Tensorflow on 2 discovery nodes. For each model we will do 8 experiments in total.

1.2 Objectives

1. Deep Neural Network Models that will be evaluated: VGGNet, GoogLeNet, ResNet-18
2. Datasets of Neural Network: MINIST, CIFAR-10
3. Platform: Discovery

1.3 Deliverables

This project relies on the Discovery platform, Pycharm is used as an integrated development environment. We choose Python as the programming language, Pytorch and Tensorflow are used as the deeping learning package.

There are two datasets that will be used. MINIST, CIFAT-10 are 2 different popular datasets in deep learning. These results are widely applied into different kinds of neural networks. They have now become benchmarks of neural networks. Each dataset will be applied to each Model respectively.

II. Background Research

2.1 Models

2.1.1 GoogLeNet

According to C. Szegedy et al [1], GoogLeNet is a new convolutional neural network, and it has 22 layers. Different from other convolutional networks, GoogleNet achieves good classification performance while controlling the amount of computation and parameters. It removes the last fully connected layer, uses a global average pooling layer, and replaces large convolutions with multiple small convolution kernels.

2.1.2 VGG16

VGGNet is a Convolutional Neural Network architecture proposed by Karen Simonyan and Andrew Zisserman from the University of Oxford [2]. It uses several consecutive 3x3 convolution kernels to replace the larger convolution kernels in AlexNet. The softmax output layer is composed, and the layers are separated by max-pooling. Increased the number of channels and more information can be extracted.

2.1.3 ResNet-18

ResNet (Residual Neural Network) comes from a paper by Microsoft Research [3], it is 18 layers deep that can learn rich feature representations for a wide range of images.

2.2 Datasets

2.2.1 Cifar-10

According to Alex Krizhevsky et al. [4], CIFAR-10 is a 3-channel color RGB image dataset that consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. 50000 training images and 10000 test images are in total.

2.2.1 Mnist

Mnist is a large database of handwritten digits gray scale images that is commonly used for training various image processing models [5]. The dataset contains 60,000 examples for training and 10,000 examples for testing. The numbers are size normalized and centered in the image, which is 28x28 fixed size.

III. Experiment Investment

3.1 Overview

This chapter mainly discussed the methods applied in each process of this project, we find and designed 10 deep learning experiments that can be executed on Nvidia Tesla P100 and Nvidia Tesla K80. There experiments are GoogLeNet with Mnist in PyTorch, GoogLeNet with Cifar10 in PyTorch, GoogLeNet with Mnist in Tensorflow, GoogLeNet with Cifar10 in Tensorflow, VGGNet with Mnist in PyTorch, VGGNet with Cifar10 in PyTorch, VGGNet with Mnist in Tensorflow, VGGNet with Cifar10 in Tensorflow, ResNet with Mnist in PyTorch, ResNet with Cifar10 in PyTorch.

Because these 10 experiments can be executed by both P100 and K80, in total, 20 results can be received.

3.2 GPU

In Northeastern Discovery System, Nvidia Tesla P100 and Nvidia Tesla P80 share the same CPU which is E5-2680v4@2.40GHz. The experimental performance of P100 is almost 2 time of the performance of P80, which can help us easily evaluate results. The precise performance specification of P100 and K 80 are shown below.

Nvidia ® Tesla® P100 [6]

Double-Precision Performance	4.7 teraFLOPS
Single-Precision Performance	9.3 teraFLOPS
High-Precision Performance	18.7 teraFLOPS
CUDA Cores	3584
SMs	56
PCIe x16 Interconnect Bandwidth	32 GB/s
CoWoS HBM2 Stacked Memory Capacity	16 or 12 GB
CoWoS HBM2 Stacked Memory Capacity	732 GB/s or 539 GB/s
Base Clock	1328 MHz

Nvidia® Tesla® P80 [7]

Double-Precision Performance	Up to 2.91 teraFLOPS

Single-Precision Performance	Up to 8.73 teraFLOPS
SMX	26
CUDA Cores	4992
GDDR5 Memory	24 GB
Aggregate Memory Bandwidth	480 GB/s
Base Clock	560 MHz

3.3 Experiment Process

Firstly, we found plenty of source code from different resources. Epoch of each experiment is set as 20 and batch is set as 128.

Since the input images that can be accepted by GoogLeNet, VGGNet and ResNet are 224x224 with RGB channel (3 channels), all images of Mnist and Cifar10 should be modified into 224*224 with 3 channels. All python files should be modified so that for different experiments which share the same neural network, their network structures are the same. Then, following the instructions of Discovery System [8], PyTorch and Tensorflow are installed. Next, for each experiment, we designed a script file which can submits the experiment to a specific GPU node and configures the node with 4GB memory. An example script is shown below.

```
#!/bin/bash
#SBATCH --partition=gpu
#SBATCH --nodes=1
#SBATCH --gres=gpu:p100:1
#SBATCH--ntasks=1
#SBATCH--mem=4GB
#SBATCH --time=08:00:00
$SRUN python ~/minist-vgg16-pytorch.py
```

Figure 1: Example of script for VGGNet 16 Programmed by PyTorch trained by Mnist database

Figure 1 as an example that show the script of VGGNet with Mnist programmed in PyTorch trained by Nvidia Tesla P100. The memory required is 4GB, and the max time allowed is 8 hours.

Finally, executing time of each experiment is calculated and analyzed. For more precise information of

IV. Experiment Results

4.1 Overview

Table.1. torch run time

	Model	PyTorch			
		P100		K80	
		Cifar10	Mnist	Cifar10	Mnist
Time(s)	GoogLeNet	2412.598	2879.357	7276.418	9123.033
	VGG-16	510.156	517.171	991.226	1140.882
	ResNet-18	645.353	732.689	1644.637	2052.974

Table.2. torch accurcy

	Model	PyTorch			
		P100		K80	
		Cifar10	Mnist	Cifar10	Mnist
Accurcy	GoogLeNet	87.924%	98.467%	87.959%	98.538%
	VGG-16	83.168%	97.855%	83.620%	97.962%
	ResNet-18	86.718%	98.178%	86.499%	98.202%

Table.3. tf run time

	Model	TensorFlow			
		P100		K80	
		Cifar10	Mnist	Cifar10	Mnist
Time(s)	GoogLeNet	1606.069	1926.538	5574.198	6384.673
	VGG-16	117.997	143.723	348.299	402.920
	ResNet-18	Null	Null	Null	Null

Table.4. tf accurcy

	Model	TensorFlow			
		P100		K80	
		Cifar10	Mnist	Cifar10	Mnist
Accurcy	GoogLeNet	85.10%	99.39%	83.88%	99.45%
	VGG-16	62.41%	97.52%	62.63%	97.45%
	ResNet-18	Null	Null	Null	Null

4.2 Comparison

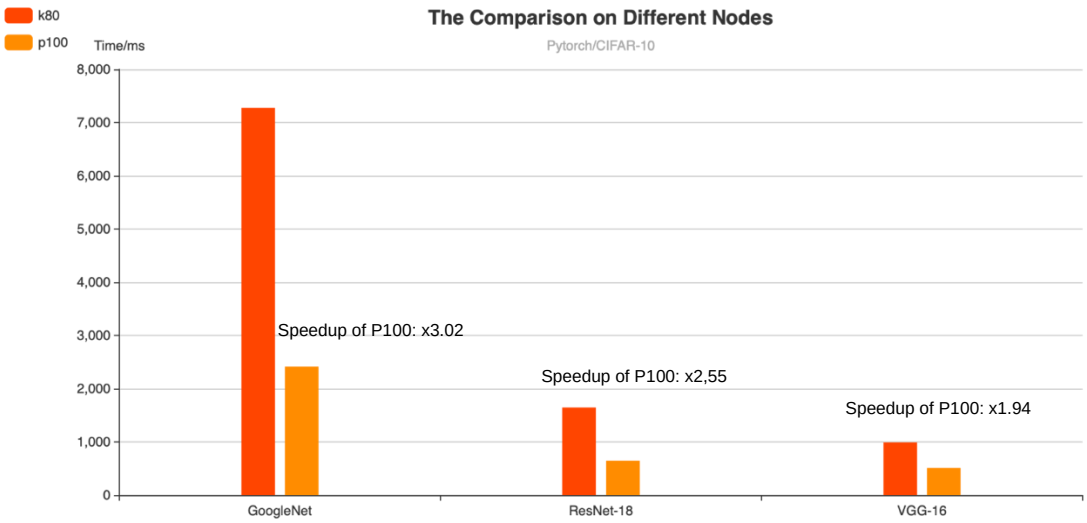

Figure.1. comparison on different nodes with the same Pytorch and CIFAR10 dataset

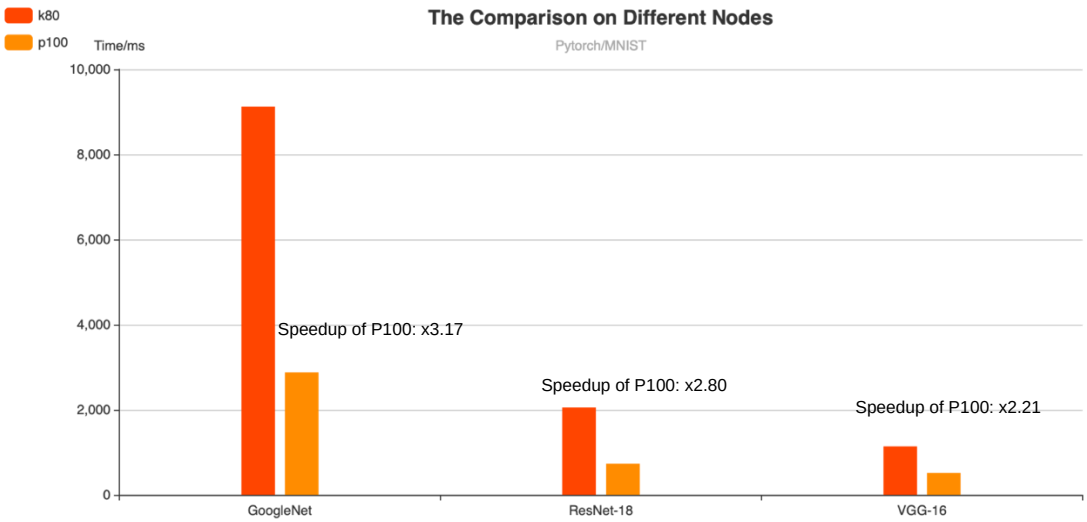

Figure.2. comparison on different nodes with the same Pytorch and MNIST dataset

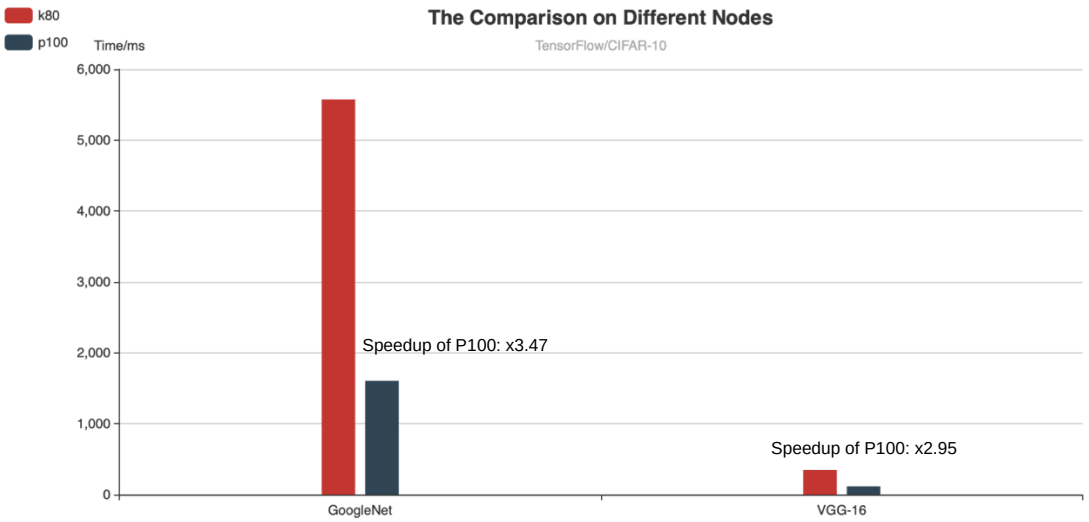

Figure.3. comparison on different nodes with the same TensorFlow and CIFAR10 dataset

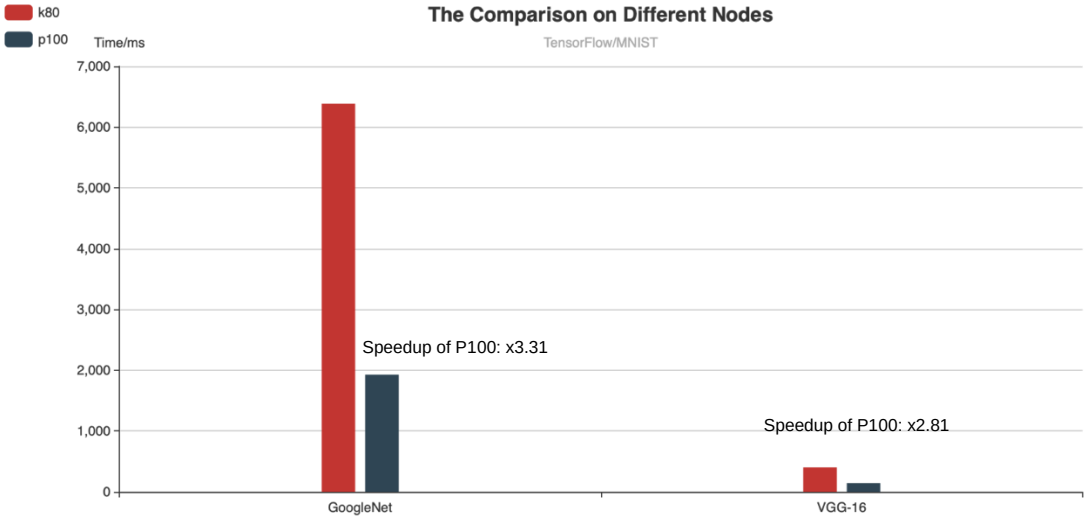

Figure.4. comparison on different nodes with the same TensorFlow and MNIST dataset

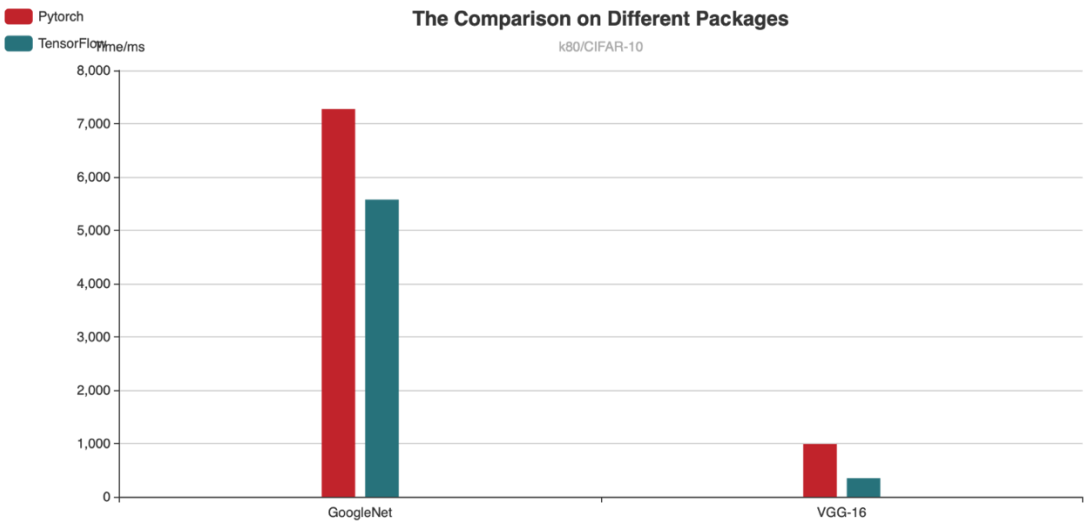

Figure.5. comparison on different packages with the same node:k80 and CIFAR10 dataset

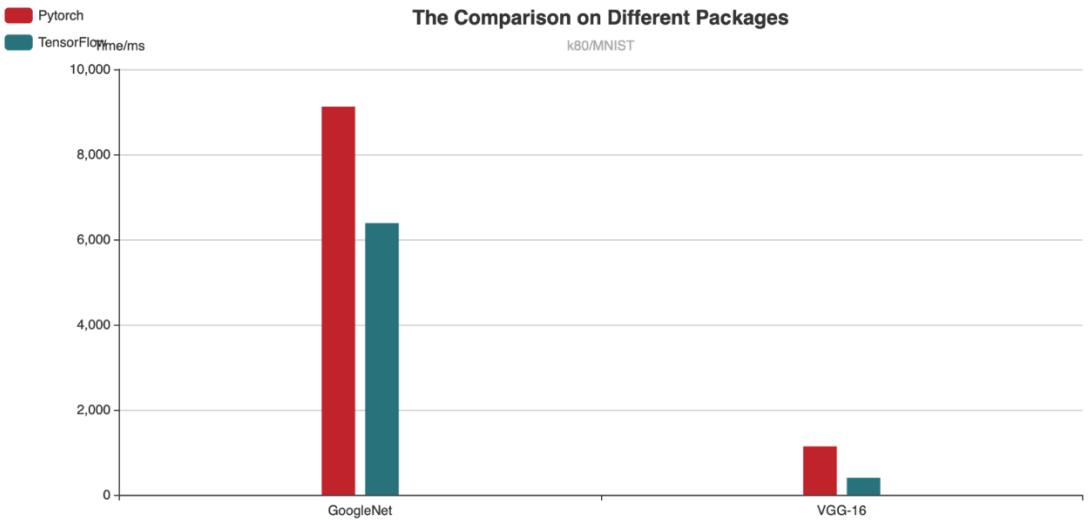

Figure.6. comparison on different packages with the same node:k80 and MNIST dataset

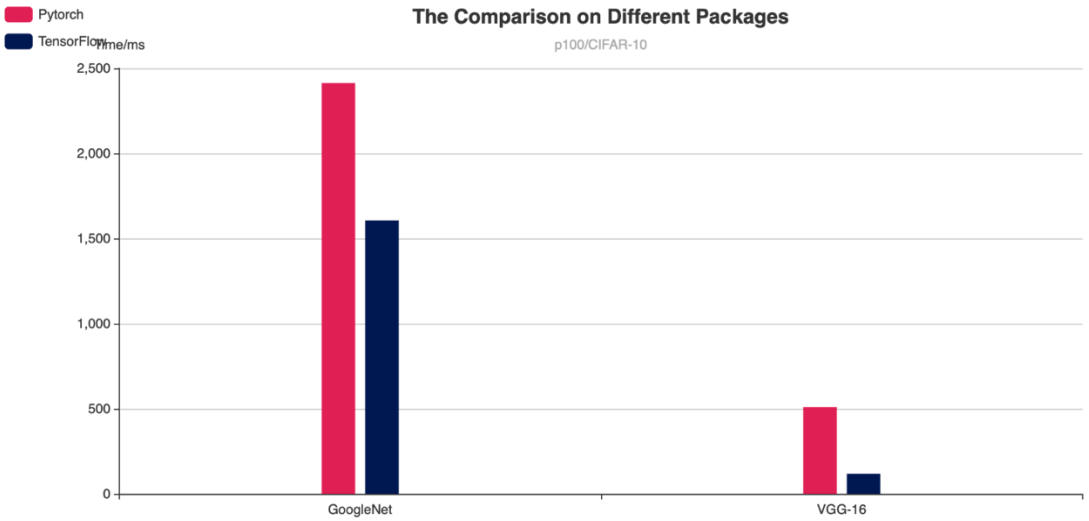

Figure.7. comparison on different packages with the same node:p100 and CIFAR10 dataset

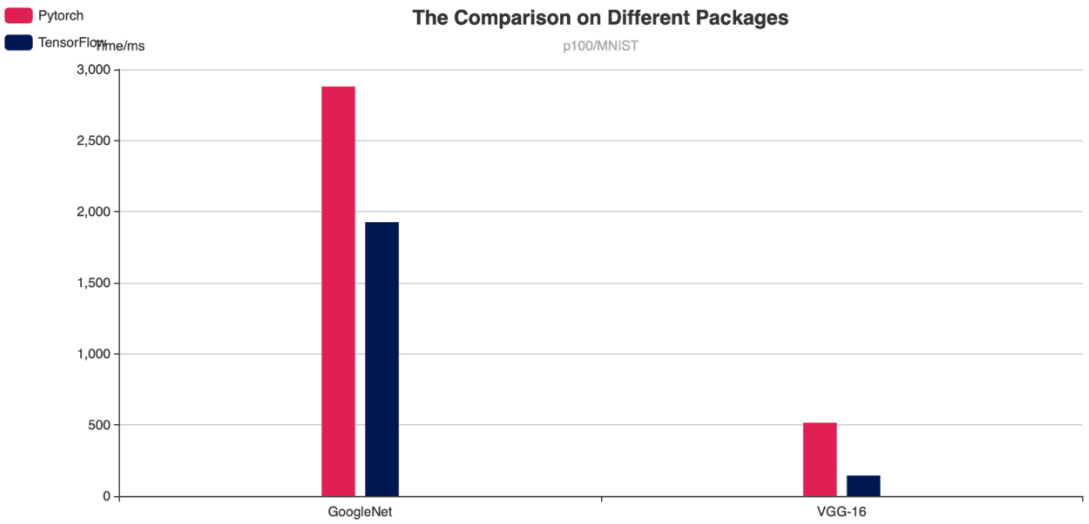

Figure.8. comparison on different packages with the same node:p100 and MNIST dataset

V. Conclusion And Evaluation

Based on the result from chapter IV, it is easy to say that p100 over performs k80 with all 3 models and 2 datasets. However, when we tried to run with the ResNet-18 model on TensorFlow on Discovery, it is failed because the TensorFlow version ResNet-18 required is too high that the Discovery platform haven't support yet. As a result, we performmed 20 experiments in total, the GoogLeNet has the highest accuracy among all three, and also took the most run time.

Table.5. p100 speedup on torch

	Model	**PyTorch**	
		Cifar10	Mnist
speedup	GoogLeNet	3.02	3.17
	VGG-16	1.94	2.21
	ResNet-18	2.55	2.80

Table.6. p100 speedup on tf

	Model	**TensorFlow**	
		Cifar10	Mnist
speedup	GoogLeNet	3.47	3.31
	VGG-16	2.95	2.81
	ResNet-18	Null	Null

As for speedup, Nvidia Tesla P100 can perfectly acceleatae GoogLeNet compared the the performance of Nvidia Tesla K80. Regardless of deep learning framework (PyTorch/ TensorFlow) and database, P100 can accelerate computation more than 3 times compared with K80. This speedup of P100 is much more higher than ResNet18 and VGGNet16 although ResNet18 also gets almost 3 times more performance which ranks 2nd. Calculations of VGGNet16 are not accelerated better than the other two networks. VGGNet16 executed by P100 only gets around x2 speedup.

Compared with PyTorch, TensorFlow runned on P100 has more speedup regardless of networks. Speedups that got on TensorFlow are higher than speedup got on PyTorch for both GoogLeNet and VGGNet16.

Although the number of CUDA cores of P100 and memory size are less than K80, P100 has higher Base Clock, more SMs, and faster NVLink Generation 1. These features and technologies help it executing deeplearning workload significantly faster than K80. Its number of SMs doubles compared with K80 and its base clock is 2.37 times faster. For tasks that requires more calculation such as GoogLeNet, more SMs and higher base clock could boost calculating more than expected. Trends shown in Figure 1 and Figure 2 verify this conclusion. GoogLeNet has the highest speedup, ResNet-18 ranks the second. VGGNet16 which require the shortest time and lowest calculation ranks 3rd.

It is not sure that why tasks that need lower calculation will be accelerated less. Perhaps this phenomenon happen because both P100 and K80 require some procedures that are consuming time and unavoidable. These procudures does not directly calculate any task. As a result, task needs lower calculation has the lower speedup. To verify this guess, further work needs to be done.

Luckily we still get some very intereting results that helped us a lot to leran how to apply HPC ideals in deep learning. By comparing the speedup of different node, we realized that the impact of equipment on performance is very far-reaching, and realized that the improvement of performance depends on the good cooperation of hardware and software.

List of References

[1] Szegedy, Christian, et al. "Going deeper with convolutions." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.

[2] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556* (2014).

[3] He, Kaiming, et al. "Deep residual learning for image recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

[4] https://www.cs.toronto.edu/~kriz/cifar.html

[5] Gangaputra, Sachin. "Handwritten digit database". Retrieved 17 August 2013.

[6] https://www.nvidia.com/en-us/data-center/tesla-p100/

[7] https://www.nvidia.com/en-gb/data-center/tesla-k80/

[8] https://rc-docs.northeastern.edu/en/latest/welcome/welcome.html