Exploring Deep Neural Network Applications on Theta at ALCF

Sergio Servantez
Computer Science Department
Illinois Institute of Technology
Chicago, Illinois 60616
sservantez@hawk.iit.edu
Advisor: Dr. Zhiling Lan

PROJECT OVERVIEW

I. Motivation

The objective of this research is to begin exploring the behavior of deep neural network (DNN) applications on massively parallel systems. In recent years, improvements in the accuracy of deep learning models have been driven by ever larger training datasets. However, these enormous datasets make training on many systems intractable. To overcome this obstacle, training of the model is often parallelized on a distributed system. This project was inspired by earlier work [1] which demonstrated the immense speed up of DNN training on large scale computers. This earlier research also showed how hyperparameters, like batch size, could effect model accuracy and training time. Similarly, other research [2] has shown how hyperparameter tuning can achieve parallel efficiency as high as 77% when training on 8192 nodes. The long term objective of this research is to analyze internode communication, memory utilization, and power consumption of DNN applications running on Theta at the Argonne Leadership Computing Facility, and to better understand how various hyperparameters effect these performance metrics. This was a new project so much of the work performed this semester has focused on setting up DNN training to run at scale on Theta, and investigating and selecting various tools and frameworks to be used.

II. Project Management

This project was conducted by senior undergraduate Sergio Servantez during the spring semester of 2019 under the supervision of Dr. Zhiling Lan. Sergio and Dr. Lan would attend weekly meetings to discuss project obstacles and set goals for the coming weeks. Additionally, progress through the semester was recorded in biweekly reports which can be found on the project website [4].

PROJECT TASKS

I. Preliminary Work

The project began by reading the papers referenced above to gain an understanding of speeding up DNN training on massively parallel computers and how certain hyperparameters effected these performance improvements. I also spent time researching Theta's architecture with a particular emphasis on the Cray Aries Network. Before this project, I had never trained a DNN application on a distributed system so much of my activities on this project were learned as I was performing them. We initially selected the Intel distribution of Caffe for the deep learning framework we would use. This decision was primarily based on previous work [1] which had used the same framework. I familiarized myself with the Caffe Framework using tutorials provided by Intel [5] and Berkeley Vision [6]. We were able to install Caffe on Theta but ran into several issues when trying to compile the framework related to dependencies and configuration file settings. I reached out to the Data Science Group at Argonne National Lab for assistance where I learned that they used Horovod and TensorFlow for distributed DNN training. These frameworks had already been installed on Theta and were being actively used among the group. With this information we made the decision to switch to Horovod knowing that in the likely event we needed assistance with the framework we could turn to Argonne for help. I was able to obtain some tutorials on using Horovod and TensorFlow on Theta from the Data Science Group.

II. Deep Neural Network Training on Theta

I spent several weeks learning how to use the Horovod framework [3] and working through the Argonne tutorials. To submit a Horovod training job on Theta, the TensorFlow and Keras frameworks must first be loaded using the data science modules. First, run the command “module avail datascience” to see the available data science modules. From this list, select the desired version of TensorFlow and Keras and load the modules using the command “module load”. For this project we are using the modules TensorFlow 1.13 and Keras 2.2.2. Note that the Horovod framework is loaded as part of the TensorFlow module. All code and other resources for this project can be found on Theta’s GPFS file system under the directory: /home/servante/projects/dnn_comm. The submission script which loads the necessary data science modules is called submit.sh.

The project directory also contains two DNN applications named tensorflow_mnist.py and horovod_mnist.py, which have been implemented in TensorFlow and Keras, respectfully. Both were started using sample code from the Horovod Framework and were modified as needed. The first training job was submitted using the TensorFlow implementation but failed due to an operating system runtime error. We determined that the error was being triggered when the MNIST dataset was loaded. The framework loaded the dataset by fetching it from an external Google server. However, the compute nodes on Theta are prohibited from accessing external networks and thus the fetch request was triggering an error. The problem was solved by installing the MNIST dataset locally and modifying the code to consume and parse the local dataset. The local MNIST dataset can be found in the project directory in the numpy file named mnist.npz. Once this issue had been resolved, we were able to successfully train our first DNN. While this implementation worked and provided output for the loss function, it did not provide the accuracy of the model. The loss function itself does not provide much insight. After many unsuccessful attempts at modifying the code to measure the model’s final accuracy, I decided to switch to the Keras implementation which has a more user-friendly API. This implementation provided both loss and accuracy, but the first models trained on Theta only achieved an accuracy of 27.64%. However, by tuning the learning rate we were able to increase top-1 accuracy as high as 99.04%.

RESULTS

I. Network Topology

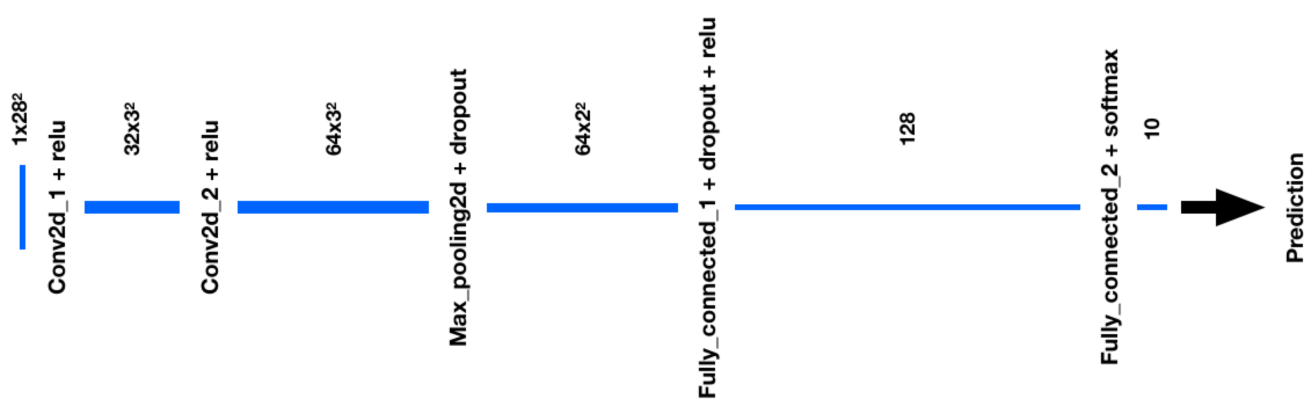

Fig. 1: Convolutional neural network topology showing the data sizes at each network layer

Figure 1 shows the topology of the convolutional neural network used to generate the results below. This model was trained using the MNIST dataset which contains 60,000 training images and 10,000 testing images. Each image in the dataset is a 28x28, one-channel image. The first network layer is configured to accept an input shape equal to the image shape. The first convolutional layer contains 32 neurons while the second contains 64, both have a 3x3 kernel size with a Relu activation function. The max pooling layer contains a 2x2 kernel size with a 25 percent dropout rate. The first densely connected layer contains 128 neurons with a Relu activation function and a 50 percent dropout rate. The last densely connected layer contains 10 neurons (equal to the number of classes) with a softmax activation function.

II. Test Accuracy

Table 1 shows top-1 accuracy as batch size is increased from 64 images to 2048 images by powers of two. Top-1 accuracy measures when the model's prediction exactly matches the correct class. The MNIST dataset contains images of handwritten digits with each image having a corresponding label which identifies its decimal value (10 classes in total). The model is first trained using the training images and labels. The test accuracy in Table 1 represents the model's ability to identify the correct decimal value on the test images which the model was not trained on. We can see from the table that accuracy decreases as the batch size increases. This pattern matches previous work [1] on the ImageNet dataset using large batch sizes. In particular, we can observe a drastic reduction in accuracy once the batch size is increased to 2048 images.

Batch Size	Epochs	Accuracy	Hardware
64	12	99.04%	128 KNLs
128	12	98.99%	128 KNLs
256	12	98.88%	128 KNLs
512	12	97.76%	128 KNLs
1024	12	97.16%	128 KNLs
2048	12	93.10%	128 KNLs

Table 1: Top-1 test accuracy comparison between various batch sizes.

III. Profiling Internode Communication

The next step in this project is to begin inspecting internode communication among ranks running DNN workloads. Throughout the semester, we have been searching for any available tools which would log such activity but have been unsuccessful in finding one up till this point. Early on we discovered that Horovod has a built-in profiler which is meant to output a timeline of each rank's activities. We have been able to successfully run this profiler but have found that the timeline output is not particularly helpful for this project. The timeline consists of a series of events with an associated timestamp. For example, the timeline will indicate that a given node began an all reduce operation at a given time, but will not provide any information about message size, communication patterns or durations. Horovod uses all MPI communications so it should be possible to record each of these metrics. Recently, I have spoken with Huihuo Zheng from Argonne National Lab who indicated that he is currently working on a tool which has been built into Horovod to log all internode communication. Moving forward this tool might be great resource to consider using.

Conclusion

During this semester we have been able to deploy a distributed deep neural network application on Theta using Horovod, TensorFlow and Keras. We have been able to observe how changes in batch size can effect test accuracy on these distributed models. We have also laid the foundation for future work on internode communication of these workloads. This project has been an immense learning experience. I have gained familiarity with Horovod and training neural networks on a massively parallel system. What I learned from this project is already being put to use at Argonne National Lab where I am working on an I/O benchmark for distributed deep learning workflows.

References

[1] Yang You, Zhao Zhang, Cho-Jui Hsieh, James Demmel, and Kurt Keutzer. 2018. ImageNet Training in Minutes. In Proceedings of the 47th International Conference on Parallel Processing (ICPP 2018).

[2] Amrita Mathuriya, Deborah Bard, Peter Mendygral, Lawrence Meadows, James Arnemann, Lei Shao, Siyu He, Tuomas Kärnä, Diana Moise, Simon J. Pennycook, Kristyn Maschhoff, Jason Sewall, Nalini Kumar, Shirley Ho, Michael F. Ringenburg, Prabhat, and Victor Lee. 2018. CosmoFlow: using deep learning to learn the universe at scale. In Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis (SC '18).

[3] Horovod Framework. https://github.com/horovod/horovod.

[4] Project Webpage. http://www.cs.iit.edu/~lan/dl-theta.html.

[5] Intel Caffe Tutorial. https://software.intel.com/en-us/articles/training-and-deploying-deep-learning-networks-with-caffe-optimized-for-intel-architecture.

[6] Berkeley Vision Caffe Tutorial. https://caffe.berkeleyvision.org/tutorial.