A Study: Deep Learning Algorithms Performance in Geo-distributed Environment

Sanqiang Zhao, Jinlai Xu and Fengtao Wu
School of Information Sciences
University of Pittsburgh
Pittsburgh, PA 15213, USA
Email: {saz31, jinlai.xu, few14}@pitt.edu

***Abstract*—With the development of cloud computing, more and more companies place their services on Cloud Service Providers(CSPs)' platforms. In order to enhance the customers' experiences, the companies replicate their services into geo-distributed data centers. However, there is a demand that the companies need to analyse the data born in the geo-distributed services. As a result, geo-distributed data analytics is a feasiable and reasonable solution for it. In addition, deep learning is the major and popular solution for analysing a large amount of data. The performance of deep learning algorithms in geo-distributed environment has not been studied sufficiently yet. In this paper, we analyse the performance of deep learning algorithms in geo-distributed environment and try to give some suggestions for optimizing the performance deep learning algorithms in geo-diteibuted environment.**

***Keywords*—Deep Learning, Geo-distributed, Data Analytics**

I. Introduction

Deep Learning is a neural network algorithm which was proposed by Geoffrey E. Hinton[7] in 2007. His paper brought the neural network which was deemed as an outdated machine learning technology and forgotten by researchers many years back to the horizon of industry. Based on Hinton's contribution, a number of software libraries have been implemented with using neural network models including deep neural networks, convolutional deep neural networks, deep belief networks and recurrent neural networks. These software libraries have achieved a great success in image classification. However, to the best of our knowledge, most of them are limited to conduct the computation locally in the datacenter. If the datasets are geographically distributed, they have to be aggregated to a single datacenter continuously in order for computation. Now that the approach of aggregating all the data to a single datacenter significantly increases the time of analytics and may arouse data privacy and security concerns, training the analytics models on the datasets which is produced continually and also distributed geographically becomes an upcoming and increasingly important challenge.

On the other hand, cloud computing is a gradually popular research topic which was proposed by Eric Schmidt being the Ex-CEO of Google in 2006. Cloud computing technology generates a great solution that we can push computation over data born in geographically-distributed data centers instead of copying raw data to a central location for analytics continuously. Some software libraries for image classification have been implemented to support the distributed computation. As a result, the distributed computation is able to decrease the time taken for computation significantly and solve the challenge mentioned before. However, to the best of our knowledge, how the network environment variables such as the bandwidth and latency affect the performance of the computation and what is the optimal neural network architecture in the distributed computation is not sufficiently studied in literature.

As the development of cloud computing, more and more companies tend to place their services on cloud environment. In addition, in order to enhance the users' experience, the companies place their services to a geo-distributed environment to minimize the latency. However, there are some problems under these circumstances. To meet the demand

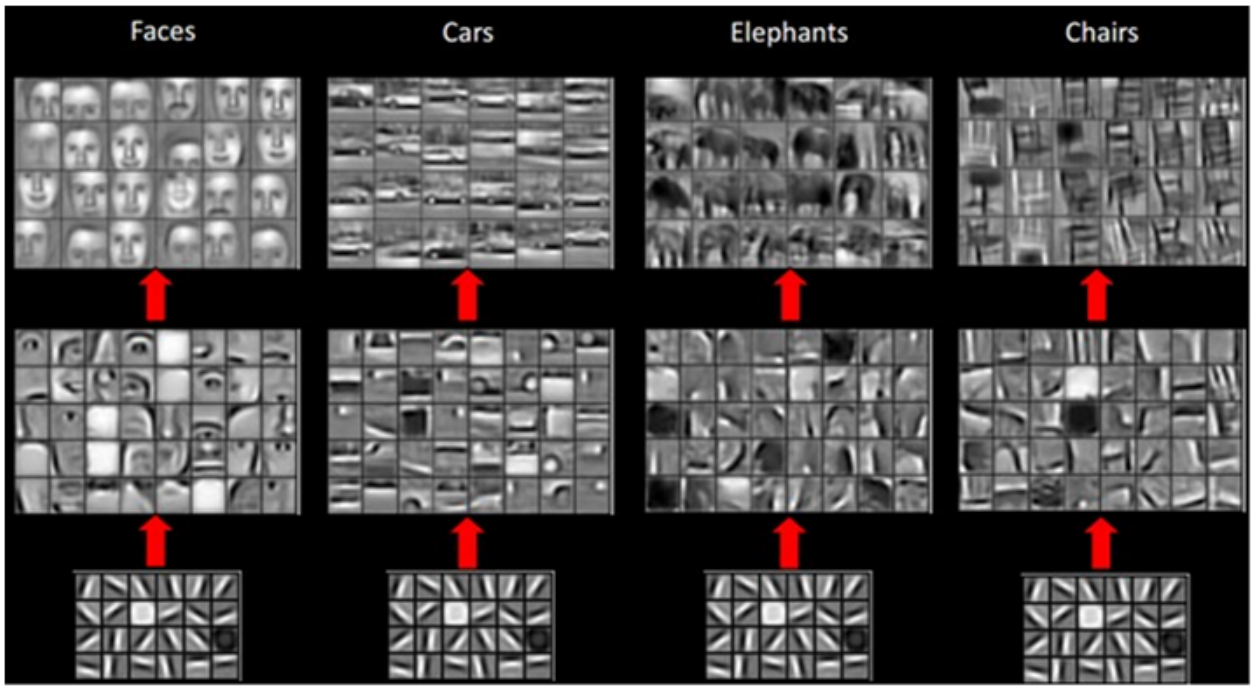

Fig. 1. Features from CNN[2]

for analysing the data that produced by the geo-distributed services, the companies seek to get a quick solution. Intuitively, the easiest way is to treat the geo-distributed clusters as one single logical data analytics cluster [10]. However, the network circumstances become more complex in this geo-distributed environment. Therefore optimizing the performance of machine learning algorithms in order to make them suitable for running under the geo-distributed environment becomes more and more important.

In this paper, our objectives are to explore the performance of deep learning algorithms on geographically-distributed cluster. We aim to find out how the bandwidth, latency, loss rate and neural network architecture affect the accuracy and time performance of the distributed computation for image classification.

II. Background

Computer vision has become a popular research field which enables computers to duplicate the abilities of human vision by electronically perceiving and understanding an image. Since 2006, the deep learning algorithms have achieved a great success in promoting the accuracy in image classification, and a number of software libraries, such as Caffe[1], Torch[4] and TensorFlow[3], have been implemented with using deep learning algorithms and other neural network algorithms at present. However, to the best of our knowledge, most of them have the limitation that the computation can only be executed locally in the data center, and supporting computation across geographically distributed datasets is not sufficiently studied in literature. That is to say, if the datasets are geographically distributed, they have to be aggregated to a single data center continuously in order for computation.

Image classification on geographically distributed datasets with low latency has become an upcoming and increasingly important challenge. For example, Yelp receives an enormous amount of photos which their users upload every day, and the photos are stored in geographically distributed data centers. The approach of aggregating all the data to a single data center significantly increases the time of analytics.

Instead of copying raw data to a central location continuously for analysis, some software libraries for image classification have been implemented to

support the distributed computation[11]. The distributed computation is able to decrease the time taken for computation significantly and solve the challenge mentioned before. However, to the best of our knowledge, how the network environment variables such as the bandwidth, latency affect the performance of the computation and what is the optimal neural network architecture in the distributed computation is not sufficiently studied in literature. In this paper, our objectives are to explore the performance of deep learning algorithms on geographically-distributed cluster. We aim to find out how the bandwidth, latency, loss rate and neural network architecture affect the accuracy and time performance of the distributed computation for image classification.

III. Algorithm Analytics

In the June of 2012, Andrew Ng and Jeff Dean trained distributed Deep Neural Networks with 16,000 CPU cores. Regarding to image classification, researchers mainly focus on CNN(Convolutional Neural Network) as well as DBM(Deep Belief Networks), Discriminative RBM(Restricted Boltzmann Machines).

In deep learning model family, Restricted Boltzmann Machines are undirected generative models that use a layer of hidden variables to model a distribution over visible variables. Deep belief nets are probabilistic generative models that are composed of multiple layers of stochastic, latent variables.

Specially, we will focus on CNN(Convolutional Neural Network). CNN will try to extract features unsupervisely. For example, in face recognition, it can extract feature indicating nose or eyes (in Fig 1). Compared with traditional SIFT or other feature extraction methods, CNN performs much better. On the other hand, CNN is suitable for distributed computing. In the convolutional layer and subsampling layer, the processes are independent each other, and hence mapreduce framework is good choice. For rest MLP layer, training is gradient descent process, which can also fit the MapReduce framework.

A. Convolution

Convolution is probably the most important concept in deep learning right now. It is an orderly procedure where two sources of information are intertwined. Purpose of convolution is for enhancing signal and reducing noise.

In mathematics aspect, convolution will generate output by element-wise multiply between input and kernel (in Fig 2). One result by applying convolution is extracting contour (in Fig 3).

B. Model

The original model we applied is from LeNet[8]. LeNet is one convolution neural network design especially designed for recognizing the hand-written digits (It used in the recognize digit on the check in bank). LeNet consists of 7 layers. The expected input is 32 by 32 matrix, which is larger than our MNIST data set. Therefore, there is no loss for preprocessing the data.

Regarding to Figure 4, C1 is convolution layer (through convolution computing for reducing noise), which contains 6 kernels with 5 by 5 size. Result is 28 by 28 feature map. S2 is max sampling layer reducing 28 by 28 input matrix into 14 by 14 matrix (pooling by 2 by 2 matrix). C3 is another convolution layer, applying 16 kernels with 5 by 5 size. S4 is another max pooling layer reduce 10 by 10 matrix into 5 by 5 matrix. From C5 to output layer, it is traditional fully connected neural network. In all, we have two convolution layer for enhancing the signal and reducing noise.

C. Applicable in MapReduce Framework

Training for convolution neural network applies back-propagation algorithm, which is a linear optimization problem. Therefore, parallel stochastic gradient descent[12] is an applicable method for the optimization problem[5].

In MXNet, convolution neural network can be trained distributed on two ways. The first one is distributed on the data since the data is independent each other. The other way is distributed on network structure, and each worker can handle part of network.

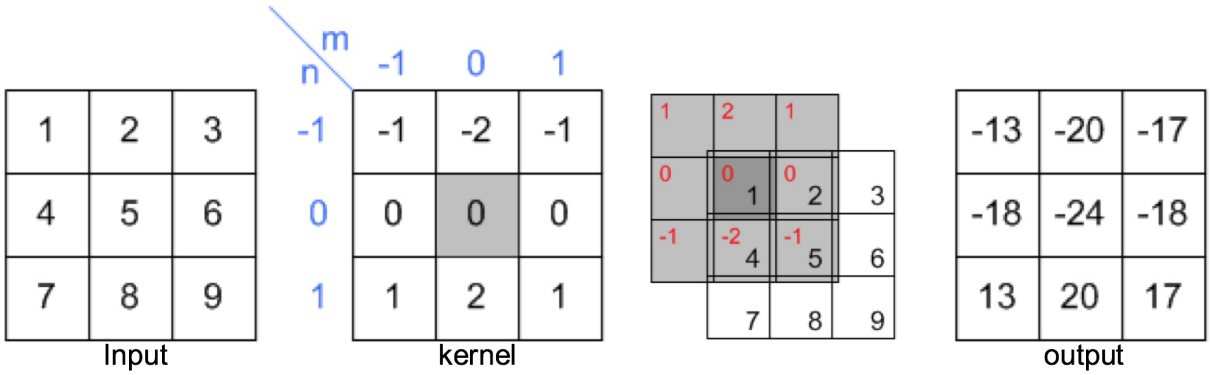

Fig. 2. Convolution in mathematics

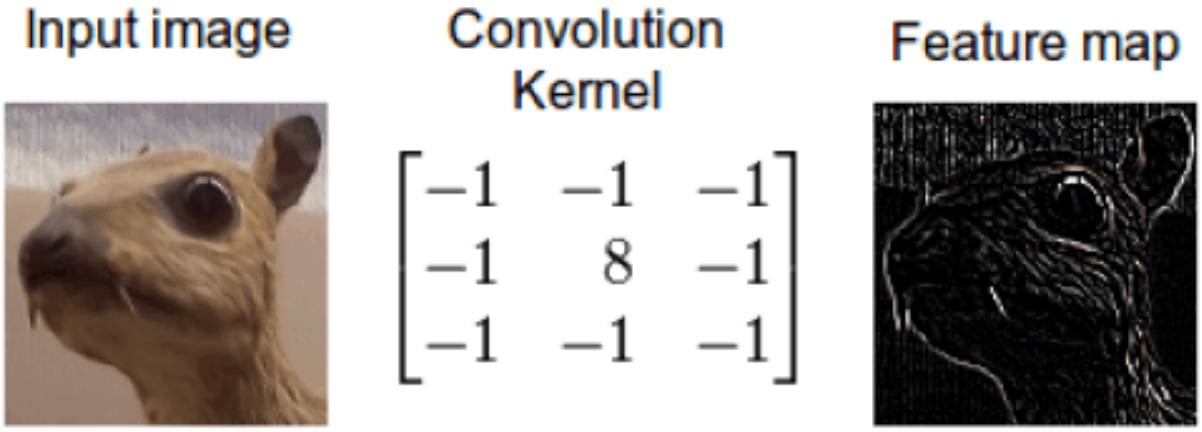

Fig. 3. Convolution result

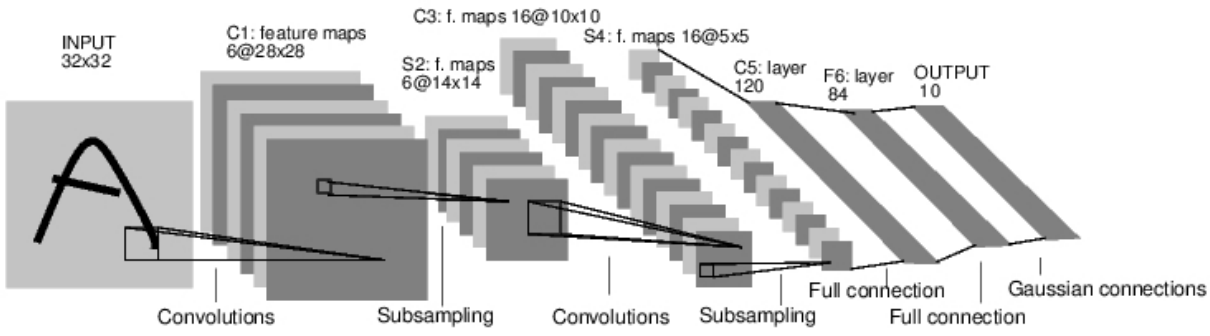

Fig. 4. LeNet Model[8]

IV. EVALUATION

A. Setup

At the first stage of evaluation experiment, we setup two virtual machines each with 1 vCPU, 4 GB memory in a MacBook Pro notebook as the environment for evaluation.

At the second stage of evaluation experiment, we setup four virtual machines each with 2vCPUs, 7.5 GB memory and a 40 GB SSD hard drive in Google Compute Engine as the environment for evaluation.

Google Compute Engine which is one product of Google Cloud delivers virtual machines running in Google's innovative data centers and worldwide fiber network. Google Compute Engine's tooling and workflow support enable scaling from single instances to global, load-balanced cloud computing. Google Compute Engine enables users to launch virtual machines which boot quickly, come with persistent disk storage, deliver consistent performance on demand. The virtual servers are available in many configurations including predefined sizes or the option to create Custom Machine Types optimized for the specific needs.

At both stages, we apply the MXNet[6] which is an open-source multi-language machine learning library to ease the development of machine learning algorithms, especially for deep neural networks to train the models for image classification. The MXNet library is implemented by Tianqi Chen, Mu Li, Yutian Li and another 7 cooperators. It allows developers to mix the flavours of symbolic programming and imperative programming to maximize efficiency and productivity. In its core, a dynamic dependency scheduler that automatically parallelizes both symbolic and imperative operations on the fly. MXNet is computation and memory efficient and runs on various heterogeneous systems, and especially it supports distributed training on multiple CPU/GPU machines.

At both stages, we apply a linux traffic control tool called tc to show and manipulate traffic control settings. Tc is used to configure Traffic Control in the Linux kernel. Traffic Control consists of the following:

SHAPING: when traffic is shaped, its rate of transmission is under control. Shaping may be more than lowering the available bandwidth, and it is also used to smooth out bursts in traffic for better network behaviour. Shaping occurs on egress.

SCHEDULING: by scheduling the transmission of packets it is possible to improve interactivity for traffic that needs it while still guaranteeing bandwidth to bulk transfers. Reordering is also called prioritizing, and happens only on egress.

POLICING: where shaping deals with transmission of traffic, policing pertains to traffic arriving. Policing thus occurs on ingress.

DROPPING: traffic exceeding a set bandwidth may also be dropped forthwith, both on ingress and on egress.

Therefore, we apply tc to manage and manipulate the transmission of packets in order to simulate the geographically distributed environment. For example, we limit the maximum bandwidth to simulate the bandwidth constraint, add latency to packets to simulate the latency due to geographical distance, and randomly drop packets according to a predefined loss rate to simulate the low-quality area network environment.

B. Dataset

At the first and the second stage of evaluation experiment, we apply the MNIST data to evaluate the accuracy and time performance of the algorithm.The MNIST[9] database (Mixed National Institute of Standards and Technology database) is a large database of handwritten digits that is commonly used for training various image processing systems, and it contains 60,000 training images and 10,000 testing images. The problem is to look at grey-scale 28x28 pixel images of handwritten digits and determine which digit the image represents, for all the digits from zero to nine.

C. Experiment Results

1) Bandwidth: Figure 5 displays the result of limiting max bandwidth in standalone environment. As illustrated in Figure 5, as the max bandwidth increases from 400 Mbps to 2000 Mbps, the time generally decreases. In the full connected neural

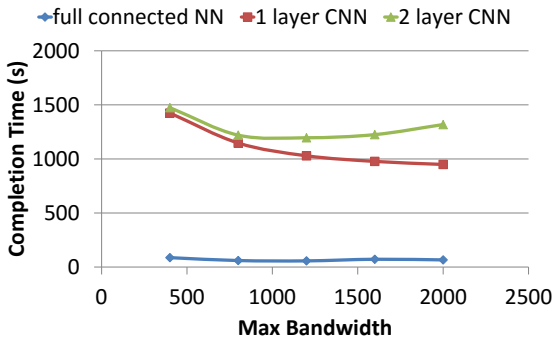

Fig. 5. The result of limiting max bandwidth in stand-alone environment

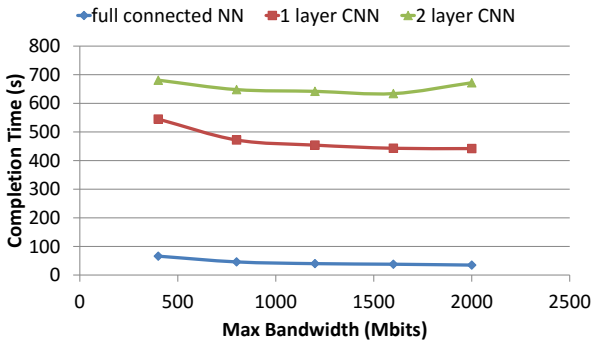

Fig. 6. The result of limiting max bandwidth in Google Cloud environment

network, the trend is explicit. The Time taken by computation in the condition of bandwidth which is 400 Mbps equals 87 seconds, and the time taken by computation in the condition of bandwidth which is 2000 Mbps equals 66 seconds. The computation time decreases by 24.14%. As to one layer convolutional neural network, the trend is also explicit. The Time taken by computation in the condition of bandwidth which is 400 Mbps equals 1424 seconds, and the time taken by computation in the condition of bandwidth which is 2000 Mbps equals 949 seconds. The computation time decreases by 33.36%. As to two layer convolutional neural network, the trend is the least explicit. The computation in the condition of bandwidth which is 400 Mbps takes the longest time which equals 1474 seconds, and the computation in the condition of bandwidth which is 1200 Mbps takes the shortest time which equals 1196 seconds. The computation time decreases by 18.86%. However, when the bandwidth reaches 2000 Mbps, the computation time which equals 1318 seconds actually increases a bit.

Comparing the three neural network models, we find that the convolutional neural network model takes 15 times longer than the full connected neural network on average. In addition, adding one more layer into the neural network increases the computation time by 18% on average.

Figure 6 displays the result of limiting max bandwidth in Google Cloud environment. As illustrated in Figure 6, as the bandwidth increases from 400 Mbps to 2000 Mbps, the time generally decreases. In the full connected neural network, the trend is the most explicit. The Time taken by computation in the condition of bandwidth which is 400 Mbps equals 66 seconds, and the time taken by computation in the condition of bandwidth which is 2000 Mbps equals 35 seconds. The computation time decreases by 46.97%. In one layer convolutional neural network, the trend is less explicit. The Time taken by computation in the condition of bandwidth which is 400 Mbps equals 545 seconds, and the time taken by computation in the condition of bandwidth which is 2000 Mbps equals 442 seconds. The computation time decreases by 18.90%. In two layer convolutional neural network, the trend is the least explicit. The computation in the condition of bandwidth which is 400 Mbps takes the longest time which equals 681 seconds, and the computation in the condition of bandwidth which is 1600 Mbps takes the shortest time which equals 634 seconds. The computation time decreases by 6.90%. However, when the bandwidth reaches 2000 Mbps, the computation time which equals 672 seconds actually increases a bit.

Comparing the three neural network models, we find that the convolutional neural network model takes 10 times longer time than the full connected neural network does on average. In addition, adding one more layer into the neural network increases the computation time by 40% on average.

The trend in stand-alone is similar with that in cluster regarding to changing bandwidth. However, compared with fully neural network by adding one convolution layer and adding two convolution layer, more convolution layers always leads to increasing

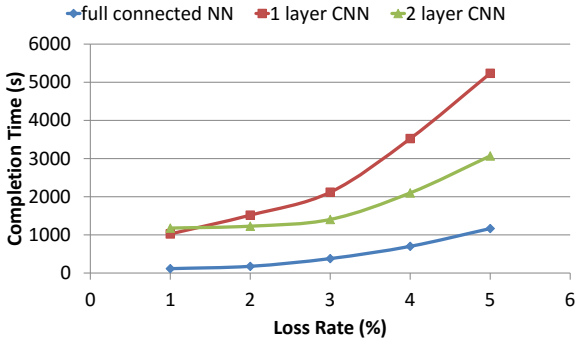

Fig. 7. The result of randomly dropping packets in stand-alone environment

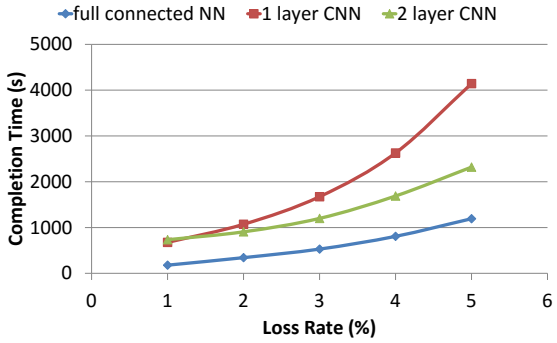

Fig. 8. The result of randomly dropping packets in Google Cloud environment

accurary. At the same time, by adding more layers, problem become more complex that reduce the completion time.

2) Loss Rate: Figure 7 displays the result of randomly dropping packets in standalone environment. As illustrated in Figure 7, as the bandwidth increases from 1% to 5%, the time generally increases. As to the three neural network, the trends are all explicit. The loss rate increasing from 1% to 5%, the computation time taken by the full connected neural network, one-layer convolutional neural network and two-layer convolutional neural network increases approximately 9.1 times, 4.1 times and 1.6 times.

It is notable to mention that one-layer convolutional neural network takes the longest time to compute under the same loss rate. In fact, two-layer convolutional neural network takes 23.92% less time to compute than one-layer convolutional neural network does on average.

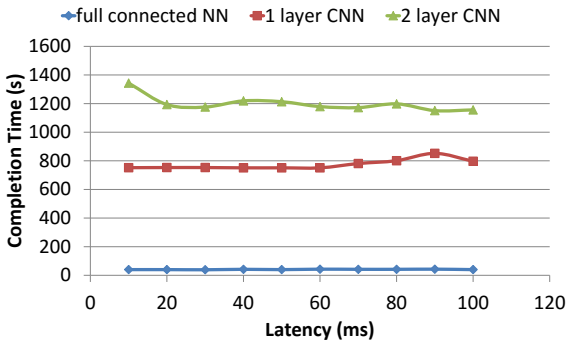

Fig. 9. The result of adding latency in stand-alone environment

Figure 8 displays the result of randomly dropping packets in Google Cloud environment. As illustrated in Figure 8, as the bandwidth increases from 1% to 5%, the time generally increases. As to the three neural network, the trends are all explicit. The loss rate increasing from 1% to 5%, the computation time taken by the full connected neural network, one-layer convolutional neural network and two-layer convolutional neural network increases approximately 5.7 times, 5.1 times and 2.2 times.

It is notable to mention that one-layer convolutional neural network takes the longest time to compute under the same loss rate. In fact, two-layer convolutional neural network takes 22.92% less time to compute than one-layer convolutional neural network does on average.

By increasing loss rate, the one convolution layer is slower than two convolution layer. That is because by removing one convolution layer, the number of parameter increases(more weight need to updated due to convolution layer reducing data dimension).

3) Latency: Figure 9 displays the result of adding latency in standalone environment. As illustrated in Figure 9, as the bandwidth increases from 10 milliseconds to 100 milliseconds, the time generally remains unchanged.

Comparing the three neural network models, we find that the convolutional neural network model takes 20 times longer time than the full connected neural network does on average. In addition, adding one more layer into the neural network increases the computation time by 55% on average.

Figure 10 displays the result of adding latency in

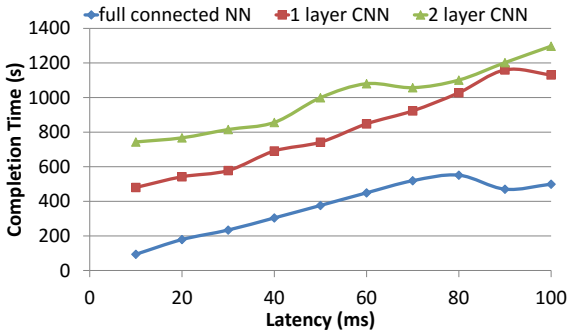

Fig. 10. The result of adding latency in Google Cloud Cluster

Google Cloud environment. As illustrated in Figure 10, as the bandwidth increases from 10 milliseconds to 100 milliseconds, the time generally increases. As to the three neural network, the trends are all explicit. The latency increasing from 10 milliseconds to 100 milliseconds, the computation time taken by the full connected neural network, one-layer convolutional neural network and two-layer convolutional neural network increases approximately 4.3 times, 1.4 times and 0.8 times.

Comparing the three neural network models, we find that the convolutional neural network model takes 1.9 times longer time than the full connected neural network does on average. In addition, adding one more layer into the neural network increases the computation time by 26.31% on average.

By changing latency, performance on stand-alone is stable. Computing time is linear increasing with increasing of number of clusters no matter which kind of network structure. Therefore, convolution layer has little effect on performance.

4) Accuracy: The accuracy is increased from about 97% to about 99% with adding one convolutional neural layer in front of the full connected Neural network. The second layer contributes less to the accuracy. The other factors like bandwidth, loss rate and latency do not influence the accuracy at all.

V. Conclusion

In this paper, we study the performance of deep learning algorithms especially the hottest Convolutional Neural Network(CNN) algorithm in geo-distributed environment and get some interesting results. Furthermore, we discuss the results and give some helpful suggestions for using deep learning algorithms directly in geo-distributed environment.

References

[1] Caffe. http://caffe.berkeleyvision.org/. Accessed: 2016-04-25.

[2] Features from cnn. http://stats.stackexchange.com/questions/146413/why-convolutional-neural-networks-belong-to-deep-learning. Accessed: 2016-02-05.

[3] Tensorflow. https://www.tensorflow.org/. Accessed: 2016-04-25.

[4] Torch. http://torch.ch/. Accessed: 2016-04-25.

[5] J. Bouvrie. Notes on convolutional neural networks. 2006.

[6] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv preprint arXiv:1512.01274*, 2015.

[7] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[8] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[9] Y. LeCun, C. Cortes, and C. J. Burges. The mnist database of handwritten digits, 1998.

[10] Q. Pu, G. Ananthanarayanan, P. Bodik, S. Kandula, A. Akella, P. Bahl, and I. Stoica. Low latency geo-distributed data analytics. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, pages 421–434. ACM, 2015.

[11] A. Vulimiri, C. Curino, B. Godfrey, K. Karanasos, and G. Varghese. Wanalytics: Analytics for a geo-distributed data-intensive world. In *CIDR*, 2015.

[12] M. Zinkevich, M. Weimer, L. Li, and A. J. Smola. Parallelized stochastic gradient descent. In *Advances in neural information processing systems*, pages 2595–2603, 2010.