Mariana: Tencent Deep Learning Platform and its Applications

Yongqiang Zou, Xing Jin, Yi Li, Zhimao Guo, Eryu Wang, Bin Xiao
Tencent Inc.
{aaronzou, xingjin, sincereli, zhimaoguo, eryuwang, leoxiao}@tencent.com

ABSTRACT

Deep learning gains lots of attentions in recent years and is more and more important for mining values in big data. However, to make deep learning practical for a wide range of applications in Tencent Inc., three requirements must be considered: 1) Lots of computational power are required to train a practical model with tens of millions of parameters and billions of samples for products such as automatic speech recognition (ASR), and the number of parameters and training data is still growing. 2) The capability of training larger model is necessary for better model quality. 3) Easy to use frameworks are valuable to do many experiments to perform model selection, such as finding an appropriate optimization algorithm and tuning optimal hyper-parameters. To accelerate training, support large models, and make experiments easier, we built Mariana, the Tencent deep learning platform, which utilizes GPU and CPU cluster to train models parallelly with three frameworks: 1) a multi-GPU data parallelism framework for deep neural networks (DNNs). 2) a multi-GPU model parallelism and data parallelism framework for deep convolutional neural networks (CNNs). 3) a CPU cluster framework for large scale DNNs. Mariana also provides built-in algorithms and features to facilitate experiments. Mariana is in production usage for more than one year, achieves state-of-the-art acceleration performance, and plays a key role in training models and improving quality for automatic speech recognition and image recognition in Tencent WeChat, a mobile social platform, and for Ad click-through rate prediction (pCTR) in Tencent QQ, an instant messaging platform, and Tencent Qzone, a social networking service.

1. INTRODUCTION

1.1 Backgroud

Tencent provides a wide range of Internet services, such as WeChat, a mobile social platform having 396 millions of monthly active users (MAU), QQ, an instant messaging platform having 848 millions of MAU, and Qzone, a social networking service having 644 millions of MAU in the first quarter of 2014. Tencent holds over 100 PB data from various applications, plus more user generated content (UGC) such as photos, audios, and videos.

Deep learning is a hot topic in mining values in big data in recent years [1], and makes a breakthrough in several different fields, such as automatic speech recognition (ASR) [2] and image recognition [3]. Deep learning has potential benefits for various applications in Tencent, such as speech recognition and image recognition in WeChat, and advertising in QQ and Qzone, to improve application quality, build exciting applications, and increase revenue.

To make deep learning practical in Tencent to realize benefits, three requirements must be considered.

- Lots of computational power and efficient parallelism is needed to train models fast. For example, acoustic model of automatic speech recognition for Chinese and English in Tencent WeChat adopts a deep neural network with more than 50 millions of parameters, more than 15 thousands of senones (tied triphone model which is represented by one output node in output layer in a DNN), and tens of billions of samples, so it would cost years to train this model by a single CPU server, or months by a single off-the-shelf GPU.
- Support for training large models is necessary to improve model quality. Since it is validated that more filter maps and more layers for CNN networks could both contribute to better classification accuracy.
- Easy to use frameworks are valuable to do various experiments to perform model selection, such as choosing model architectures, finding optimization methods and tuning optimal hyper-parameters for a high performance model, especially for applications with mass data.

To meet above requirements, parallel frameworks are essential to make training model faster, bigger, and easier.

Data parallelism and model parallelism are introduced in Google DistBelief [4] for CPU clusters, and also used in Google COTS systems [5] for GPU servers as well as Facebook multi-GPU training [6]. Data parallelism launches multiple model replicas for different mini-batches, and collects gradients to update model parameters. Model parallelism involves multiple workers each holding parts of the model and swapping activations and errors to complete a mini-batch. Besides improving performance, model parallelism reduces memory consumption for each worker, thus make it possible to build larger models.

Besides company proprietary frameworks, some deep learning frameworks are emerged in the open source community using CPUs or GPUs. However, available open source frameworks do not meet the requirements in Tencent. These frameworks usually train models using a single multi-threaded CPU server, or a single GPU, therefore lack of efficient parallelism or large model support for tremendous training data and large scale models in Tencent. For the easy to use requirements, these frameworks usually need tons of efforts to code, deploy, and configure to make them be able to start new experiments.

1.2 Platform Overview

To support emerging deep learning applications in Tencent, we built Mariana, the Tencent deep learning platform, to accelerate training, support large models, and facilitate experiments.

We found it's hard to construct a one-size-fits-all framework to support a wide range of applications including speech recognition and image recognition in WeChat, and Ads pCTR in QQ and Qzone. Different applications emphasize different aspects of requirements.

So Mariana builds three parallel frameworks for different applications: 1) a multi-GPU data parallelism framework for deep neural networks (DNNs). 2) a multi-GPU model parallelism and data parallelism framework for deep convolutional neural networks (CNNs). 3) a CPU cluster framework with model parallelism and data parallelism for large scale DNNs.

Mariana makes efforts to simplify deep learning experiments and offload burden of algorithm engineers. Mariana provides built-in algorithms, allows flexible hyper-parameter tuning, supports checkpoint and restart at configurable periods, generates automatically test reports, and enables training jobs monitoring.

Mariana is used by various applications such as automatic speech recognition in WeChat, image recognition in WeChat and Ad click-through rate prediction (pCTR) in QQ and Qzone, and plays a key role for these applications in Tencent for more than one year.

Mariana uses GPU servers each equipped with 4 or 6 GPUs, and also has CPU servers.

The multi-GPU data parallelism DNN framework is used to build the acoustic model of automatic speech recognition in Tencent WeChat, and gains a speedup of 4.6 times by 6 GPUs in one server compared to one GPU.

The multi-GPU model parallelism and data parallelism CNN framework trains image recognition model for Tencent WeChat and for Ads pCTR in QQ and Qzone. The CNN framework for ImageNet 2012 dataset with Krizhevsky's network [3] using 4 GPUs gains a speedup of 2.52 times over one GPU.

The CPU cluster framework is also used to build model of automatic speech recognition.

This paper is organized as follows. Section 2 describes the multi-GPU data parallelism framework for DNNs. Section 3 introduces the multi-GPU model parallelism and data parallelism framework for Deep CNNs. Section 4 gives a brief description for the CPU cluster framework. Section 5 shows applications and evaluations. Section 6 discusses related work. Section 7 concludes this paper.

2. MULTI-GPU DATA PARALLELISM FRAMEWORK FOR DEEP NEURAL NETWORKS

2.1 Application Requirements

Mariana includes a multi-GPU data parallelism framework for DNNs, and its first target application is acoustic model of automatic speech recognition.

Acoustic model of ASR tries to classify triphones from input audio signals. Acoustic model uses a fully connected 4 to 6 hidden layers deep neural networks with about 50 millions of parameters.

ASR needs many computational power, but consumes only about 1 GB memory, so GPU is a good available option which has much higher computational power compared to microprocessors but with limited memory. To exploit multiple GPUs in one server, we first built a model parallelism framework for ASR and gained a 1.5 times speedup with two GPUs. However, model parallelism has limited scalability for ASR and could not achieve better performance when using more than two GPUs. So we pay much more attention to data parallelism for multi-GPU DNN framework.

2.2 Architecture

To control the data parallelism training, a worker group is used for each model replica. Since the DNN framework has no model parallelism, each worker group contains one GPU.

Multi-GPU data parallelism DNN framework uses a driver program running in CPUs to control the whole synchronized stochastic gradient descent (SGD) training process. The driver program threads read next round training samples as mini-batches from disk, and copy these samples from CPU memory to GPU memory for next round training. At the same time, the driver program waits for all GPUs to finish training its assigned mini-batches in current round, and then coordinates the parameter swapping between all GPUs.

2.3 Parameter Swapping

Parameter swapping process is illustrated in Figure 1 and has three phases in logical view. First, the gradient collection phase gathers gradients from all GPUs. Then, the parameter updating phase updates the model with the gathered gradients. Last, the parameter distribution phase scatters the updated model to all GPUs. In physical view, parameter swapping copies data through PCIe or input output hub (IOH) connections between GPUs. For a multi-GPU server, GPUs are connected by PCIe switches in a tree topology. For a server with more than 4 GPUs, these GPUs are organized as two groups connected by an IOH.

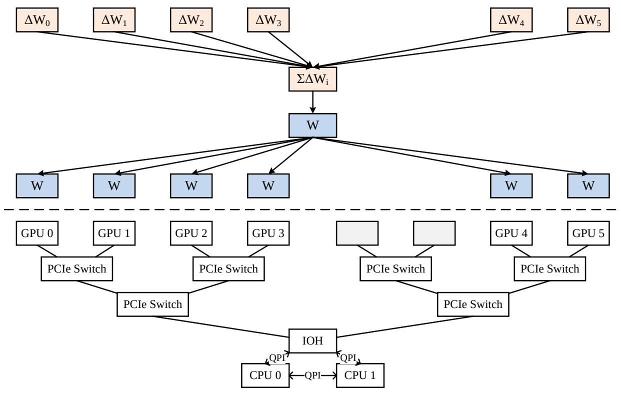

Figure 1. Multi-GPU parameter swapping.

The parameter swapping is overhead for data parallelism for each round mini-batches. Parameter swapping performance is limited by PCIe and IOH bandwidth, which is about 4 to 6 GB/s for PCIe 2.0, and is much slower than 200+ GB/s GPU memory bandwidth for GPUs like NVIDIA Tesla K20 serials. With more GPUs for data parallelism, the parameter swapping becomes worse performance bottleneck.

2.4 Partitioned Linearity Topology for Parameter swapping

The key to improve parameter swapping performance is a novel topology to minimize total copied data and maximize the utilizations of all bandwidths between GPUs.

A straightforward solution is a star topology, where each GPU copies its gradients to all other GPUs. For N GPUs, there are N^2 / 2 gradients to be copied and the copies data obviously is not minimal. Another common solution is a tree topology, which gathers gradients from leaves to the root, but not all GPU bandwidths are used during gradients collection or parameter distribution for top tree levels.

In this paper, we propose a partitioned linearity topology, and show how N GPUs (e.g. N = 6) to collect gradients in Figure 2.

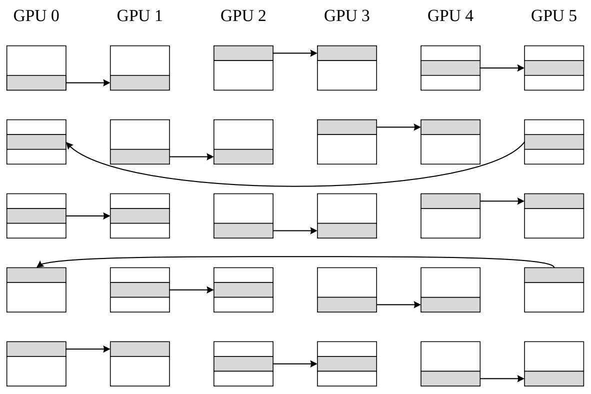

Figure 2. Partitioned linearity topology for data parallelism: the gradient collection phase.

In the partitioned linearity topology, we split model and gradients as N / 2 partitions, and assign a GPU as the owner for each partition. In each period, each partition is transferred to and merged with its next adjacent GPU, while all the N / 2 data transfer flows don't interfere with each other. After N - 1 periods, each partition owner collects all GPU's gradients of its owned partition, and then applies the gradients locally to update its owned model partition. The distribution phase distributes the latest model to all GPUs in a similar topology, except in a reversed direction.

The partitioned linearity topology performance could be modeled as follows. Let T_0 be the time copying a whole model between 2 GPUs, then time to swap parameter for a mini-batch between N GPUs is in equation 1. As a result, parameter swapping time has the upper bound $4T_0$, so it is easy to be scaled up to 8 or even more GPUs.

$$\text{Parameter swapping time} = 4T_0 \frac{N-1}{N} \qquad \text{(Equation 1)}$$

2.5 Approximate AdaGrad

We further introduce the approximate AdaGrad learning rate in data parallelism to reduce overhead of AdaGrad [7]. AdaGrad needs to accumulate a helper sum for each parameter. Let $helper\_sum_{ij}$ be the helper sum of the i-th round updates of each parameter j as shown in equation 2. Helper sum accumulates square of gradients and reduces learning rate, so that to make each parameter be updated with its own monotonically decreasing learning rate.

$$helper\_sum_{ij} = \sum_{gpu\_id=0}^{MAX\_GPU\_ID} \sum_{k=0}^{i} gradients_{kj}^2 \qquad \text{(Equation 2)}$$

Helper sum is also split into N / 2 partitions and collected from all GPUs by linear topology like gradient collection. However, helper sum does not need distribution phase since global helper sum is only useful for partition owner in the parameter updating phase.

Furthermore, we introduce the Approximated AdaGrad by reducing the frequency of helper sum collecting, which means each GPU accumulates local helper sum every mini-batch, while the partition owner only collects helper sum every M mini-batches among all GPUs.

3. MULTI-GPU MODEL PARALLELISM AND DATA PARALLELISM FRAMEWORK FOR DEEP CONVOLUTIONAL NETWORKS

3.1 Application Requirements

Mariana includes a multi-GPU model parallelism and data parallelism framework for deep CNNs and majorly aims to computer vision applications such as image recognition. The CNN framework could also be used for automatic speech recognition.

Deep CNNs introduced by Krizhevsky etc [3] gains best accuracy in ImageNet 2012 1000-category classification contests. This network has five partitioned convolutional layers and three fully connected layers, includes about 50 millions parameters, and is trained on about 1.2 million images.

Both model parallelism and data parallelism are beneficial to image classification jobs with deep CNNs. Model parallelism is necessary because the consumed memory is 3.99 GB and is close to the limit of some GPUs memory. For image classification applications, large GPU memory is important to train larger networks to get better results. Data parallelism is a good supplement for model parallelism to further speed up training with more GPUs.

3.2 Architecture

The architecture of the multi-GPU model parallelism and data parallelism CNN framework is illustrated in Figure 3, and uses Krizhevsky's network [3] for ImageNet 2012 dataset as an example.

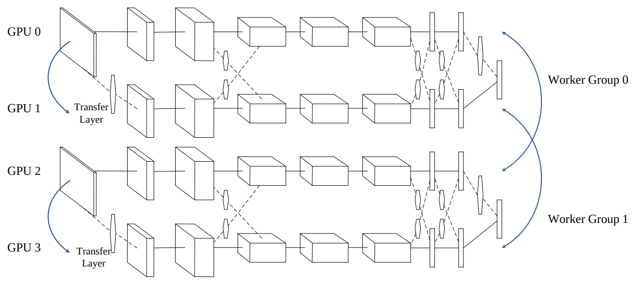

Figure 3. Architecture of multi-GPU model parallelism and data parallelism framework for deep CNNs.

For a 4 GPU server, there are two worker groups for data parallelism and each worker group has two GPUs for model parallelism. In each worker group, the model is partitioned and stored in two GPUs. During forward and backward propagation for each mini-batch, activations and errors are swapped between these two GPUs in the same worker group. When a mini-batch is finished, parameter swapping is needed to collect gradients and distribute new model of the same partition in two worker groups. As in figure 3, GPU 0 and GPU 2 swap parameters for the first model partition, while GPU 1 and GPU 3 swap parameters for the second one.

3.3 Partitioned Linearity Topology for Parameter Swapping in Data Parallelism

The partitioned linearity topology is also used in data parallelism of multi-GPU deep CNNs. The idea is the same as the one in the DNN framework, and the implementation is similar.

For 4 GPUs configuration using both model parallelism and data parallelism, the topology degrades to a more simple way: only two worker groups need swapping parameters, each for its owned partition.

3.4 Transfer Layer

We introduce transfer layer to simplify code and improve computing performance for model parallelism. Transfer layer improves performance by involving inter-GPU data copy to ensure only local GPU memory access. Each transfer layer is automatically created between two adjacent layers across different GPUs, for both forward and back propagation. Each transfer layer copies activation or errors internally as necessary to accomplish model parallelism. An execution engine is responsible for driving and orchestrating computations in different GPUs.

3.5 Pipeline Optimization

Before feeding training data to GPUs, reading training data and preparing for above CNN network are time-consuming tasks for images, which become more severe after model parallelism and data parallelism are exploited to optimize GPU compute time. We adopt a three-stage pipeline for the training process, including training data reading, training data processing (deserializing, extracting pixel values, and cropping image borders et al.), and CNN training. Each stage handles 2 batches by a thread pool. Note that these three stages are I/O intensive, CPU intensive and GPU intensive tasks respectively, so that we can balance workload and fully utilize system resources.

4. CPU CLUSTER FRAMEWORK

4.1 Application Requirements

Mariana includes a CPU cluster data parallelism and model parallelism framework for DNNs. DNN CPU cluster framework aims to two kinds of applications. First, CPU cluster is useful for applications needing large model with huge memory consumptions, moderate computational power for one mini-batch, and lots of training data. This kind of applications needs both data parallelism and model parallelism, and must be rewritten to solve this large scale problem. Second, legacy CPU training applications could be empowered by data parallelism with little code modification to become a parallel version.

4.2 APIs and Architecture

The DNN CPU cluster framework introduces a vertex abstraction with messages to form a Bulk Synchronous Parallel (BSP) paradigm. A vertex could be a neuron or a group of continuous neurons or even a whole DNN model. The vertex APIs include methods to support gradient serialize and parameter deserialize for parameter swapping. DNN jobs in CPU cluster are trained in asynchronous SGD for many iterations.

The DNN CPU cluster framework architecture is showed in Figure 4.

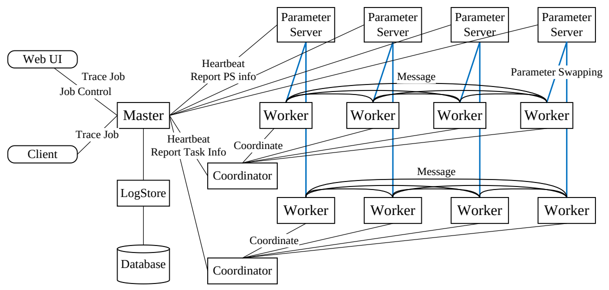

Figure 4. Architecture of CPU cluster framework for DNNs.

A client CLI tool is used to submit DNN jobs, then the master, workers, and parameter servers are scheduled through a resource management system. The master splits training data as tasks, schedules tasks among worker groups, and manages task failover. Master manages job status, task status, and worker group status through heartbeats, and sinks status to database to keep jobs history. Master collects counters from workers and parameter servers. Web UI connects to master and shows the DNN job internal status and counters. A coordinator is selected from a worker group to negotiate the model parallelism for each mini-batch among workers in a BSP paradigm. Workers do real computation work and send messages within a worker group to support model parallelism. Each parameter server keeps a shard of model, and interacts with corresponding workers to collect gradients and distribute latest model. Parameter servers write model shards periodically to distributed file system for durability.

5. APPLICATIONS AND EVALUATIONS

Mariana is currently used in training models for automatic speech recognition, image recognition, and Ads pCTR. The configurations of GPU servers are as follows: 4 to 6 NVIDIA Tesla K20c GPUs with 4.8 GB memory, two 8-core Intel(R) Xeon(R) CPU E5-2640 v2 @ 2.00GHz, 48 GB memory, 4 Seagate 2 TB SATA disks organized as RAID5. The only difference is that ASR uses 6 GPUs in one server while image recognition uses 4 GPUs.

5.1 Automatic Speech Recognition

Automatic speech recognition is now used in Tencent WeChat as voice input, speech open platform, and translating audio message to text. With voice input, users talk to WeChat directly, get the corresponding text message automatically, and send to friends. With the open speech platform, third-party companies could integrate ASR ability to their products effectively and efficiently. With translating audio message to text, users press received speech messages from friends for a short time to automatically translate speech to text when it's not convenient to listen to the message.

Now Mariana becomes the basis of model training for all ASR functions in WeChat. Along with the multi-GPU data parallelism DNN framework as the major tool, the DNN CPU cluster framework also serves as a complement method to train ASR models.

ASR includes acoustic modeling, language modeling, and decoding. Acoustic model is equipped with DNNs to enrich the capability to capture the distribution of acoustic signals.

To train the acoustic model, we construct a DNN network with a 616-node input layer, 4 hidden layers, a softmax output layer with about 16000 nodes, and more than 50 millions parameters. To make DNN model estimation stable, billions of samples are gathered to train the model.

With 6 GPUs data parallelism, we gain 4.6 times speedup over one GPU. With approximate AdaGrad learning rate, the model converges in less than one week, and the character error rate decreases more than 10% comparing with the previous model trained by multi-GPU model parallelism framework without AdaGrad.

5.2 Image Recognition

Potential image recognition applications with deep learning are ImageNet classification as a benchmark, image recognition in Tencent WeChat, and Ads pCTR in Tencent QQ and Qzone.

All above image recognition applications could use the Mariana to train models, and majorly uses the multi-GPU model parallelism and data parallelism framework for deep CNNs.

We train our model on the ImageNet 2012 training set (1.2 million images, over 1000 different classes) using Mariana. We consider model parallelism and data parallelism with 4 different configurations to explore the impact of parallelism. As a comparative model, we use the same network as Krizhevsky's in ImageNet 2012 [3].

With our deep CNN framework, 2 GPUs model parallelism speedup is 1.71 times over one GPU for training one batch. With 4 GPUs employing both model and data parallelism, we gain a speedup 2.52. For 4 GPUs data parallelism, the speedup is 2.67.

Table 1. Multi-GPU performance in speedup vs 1 GPU

Configurations	Speedup vs 1 GPU
2 GPUs model parallelism	1.71
2 GPUs data parallelism	1.85
4 GPUs model + data parallelism	2.52
4 GPUs data parallelism	2.67

For ImageNet classification, memory for each GPU is reduced from 3.99 GB to 2.15 GB with model parallelism and is able to train larger CNN network using the same GPUs. Both using the data parallelism and model parallelism, we get the same error rate as the Krizhevsky's on the ImageNet 2012 validation set.

We also explore different model architectures from Krizhevsky's network on the ImageNet 2012 data set using Mariana. By cutting the filter size of the first convolutional layer and increasing the number of feature maps in the middle layers, we get a better performance, beating Krizhevsky's single model result by 2%.

6. RELATED WORK

Several frameworks have been emerged to accelerate and facilitate deep learning.

Google DistBelief [4] is a CPU cluster framework employing data parallelism and model parallelism to train a deep network with over 1 billion parameters using tens of thousands of CPU cores. DistBelief provides downpour SGD and sandblaster L-BFGS and is used for speech recognition and the 21k category ImageNet classification.

A deep convolutional neural network [3] is designed by Krizhevsky to get top 5 error rate 15% for ImageNet LSVRC-2012 1000-category classification contest by an efficient GPU implementation Cuda-Convnet using two GPUs model parallelism. Cuda-Convnet is available as an open source toolkit for one single GPU CNN training. Cuda-convnet2 [8] improves training performance for multi-GPU CNN by a novel composing of model parallelism and data parallelism, but is not open yet.

Google COTS HPC systems [5] use data parallelism and model parallelism in GPU servers with Infiniband interconnects and MPI. COTS is able to train 1 billion parameter networks with 3 GPU machines in several days.

Facebook multi-GPU deep CNN [6] is used to employ data parallelism and model parallelism to train CNN models, and is able to train Krizhevsky's ImageNet 2012 1000-catetory network by 4 NVIDIA TITAN GPUs in a couple of days.

Several open source frameworks are available.

Caffe [9] provides fast CNN implementations for CPU or a single GPU, provides deep learning algorithms, and is able to process more than 40 million images per day with a single NVIDIA K40 or Titan GPU.

Kaldi [10] is a speech recognition research toolkit with fast C++ and CUDA code to run in CPU or GPU.

Theano [11] is a Python library to define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays, which is easily running on CPU or GPU and facilitates DNN training.

7. CONCLUSION

Deep learning gains notable success in several fields, such as automatic speech recognition and image recognition, but is hard to be practical for training models for various applications in Tencent due to big data, model complexity, and model selection.

This paper introduces Mariana, the Tencent deep learning platform to accelerate training, support large model, and simplify experiments for various of applications in Tencent. Mariana includes three frameworks: a multi-GPU data parallelism framework for DNNs, a multi-GPU model parallelism and data parallelism framework for deep CNNs, and a CPU cluster framework for large scale DNNs. Mariana also provides built-in algorithms and features to facilitate experiments.

By data parallelism DNN framework with 6 GPUs using partitioned linearity topology for parameter swapping, we build an acoustic model for ASR with more than 50 million parameters and billions of samples and get a 4.6 times speedup over a single GPU. Along with approximate AdaGrad, this training job completes in less than one week, and the character error rate decreases more than 10% compared with that by previous model parallelism training.

By model and data parallelism deep CNN framework with 4 GPUs using transfer layer and pipeline optimization, we train the ImageNet 2012 1000-category classification model and get a speedup 2.52 times over one GPU. We construct a larger network with model parallelism and beat Krizhevsky's single model result by 2%.

Mariana has been used for more than one year to train models for automatic speech recognition and image recognition in Tencent WeChat, and for Ads pCTR in Tencent QQ and Qzone.

8. ACKNOWLEDGMENTS

Thanks all team members for contributions of this paper. Many thanks to Yong Li, Bo Chen, Wei Xue, Lei Xiao, Chunjian Bao, and Jie Jiang for numerous advises and continuous support.

9. REFERENCES

[1] Hinton, G. E., and Salakhutdinov, R. R. Reducing the Dimensionality of Data with Neural Networks. *Science, 313,* 6 (July. 2006), 504-507.

[2] Hinton, G., Deng, L., Yu, D., et al, Kingsbury, B. Deep Neural Networks for Acoustic Modeling in Speech Recognition. *IEEE Signal Processing Magazine*. 29, 6 (November. 2012), 82-97.

[3] Krizhevsky, A., Sutskever, I., and Hinton, G.E. ImageNet Classification with Deep Convolutional Neural Networks. In *Proceedings of the Neural Information Processing Systems (NIPS'12)* (Lake Tahoe, Nevada, United States, December 3–6, 2012). Curran Associates, Inc, 57 Morehouse Lane, Red Hook, NY, 2013, 1097-1106.

[4] Dean, J., Corrado, G.S., Monga, R., et al, Ng, A. Y. Large Scale Distributed Deep Networks. In *Proceedings of the Neural Information Processing Systems (NIPS'12)* (Lake Tahoe, Nevada, United States, December 3–6, 2012). Curran Associates, Inc, 57 Morehouse Lane, Red Hook, NY, 2013, 1223-1232.

[5] Coates, A., Huval, B., Wang, T., Wu, D. J., Ng, A. Y. Deep learning with COTS HPC systems. In *Proceedings of the 30th International Conference on Machine Learning (ICML'13)* (Atlanta, Georgia, USA, June 16–21, 2013). JMLR: W&CP volume 28(3), 2013, 1337-1345.

[6] Yadan, O., Adams, K., Taigman, Y., Ranzato, M. A. Multi-GPU Training of ConvNets. *arXiv*:1312.5853v4 [cs.LG] (February 2014)

[7] Duchi, J. C., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12, 7 (July. 2011) , 2121-2159.

[8] Krizhevsky, A. Parallelizing Convolutional Neural Networks. in *tutorial of IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2014)*. (Columbus, Ohio, USA, June 23-28, 2014). 2014.

[9] Jia, Y. Q. Caffe: An Open Source Convolutional Architecture for Fast Feature Embedding. *http://caffe.berkeleyvision.org* (2013).

[10] Povey, D., Ghoshal, A. Boulianne, G., et al, Vesely, K. Kaldi. The Kaldi Speech Recognition Toolkit. in *Proceedings of IEEE 2011 Workshop on Automatic Speech Recognition and Understanding(ASRU 2011)* (Hilton Waikoloa Village, Big Island, Hawaii, US, December 11-15, 2011). IEEE Signal Processing Society. IEEE Catalog No.: CFP11SRW-USB.

[11] Bergstra, J., Breuleux, O., Bastien, F., et al, Bengio, Y. "Theano: A CPU and GPU Math Expression Compiler". in Proceedings of the Python for Scientific Computing Conference (SciPy 2010). (Austin, Texas, USA. June 30 - July 3, 2010).