does not use our format

Distributed Environment For Parallel Neural Networks

Ajinkya Khamkar
Indiana University
P.O. Box 1212
Bloomington, Indiana 47408
adkhamka@iu.edu

ABSTRACT

The past decade has seen the rise of Deep Neural Networks. A standard Deep Convolutional Neural Network has an upwards of Million parameters to train. The data required to train these networks typically ranges in Hundred's of Gigabytes, making it inefficient to train these networks on standalone machines. Graphical Processing Units decrease the computation time significantly but suffer from memory constraints. Existing Industrial architectures use a distributed computing paradigm capable of handling parallel computing tasks. I highlight approaches which use cheaper commodity systems integrated in a distributed fashion to handle training such Deep Neural Networks.

KEYWORDS

I523, Distributed Systems, Convolutional Neural Networks, Parallel Systems, Deep Neural Networks

1 INTRODUCTION

The past decade has seen the rise of Deep Neural Networks. Neural Networks have the ability to model complex non-linear functions by efficiently representing the input parameters as a system of linear equations with non-linear activation. They have achieved unparalleled success in the fields of Computer Vision, Natural Language Processing and Artificial Intelligence. Section 2 discusses the number of parameters required to be trained for popular deep architectures. Large amounts of data is required to train these parameters. Section 3 discusses the size of the traditional data sets used to train these networks. Deep Neural Networks are inherently parallel in nature, with weights and gradient updates shared across layers within the network. Section 4 discusses various ways to introduce parallelism while training Deep Neural Networks. Section 5 discusses methodologies to update Model parameters when the data to train is distributed across multiple machines within the network. Section 6 introduces methodologies to train multiple layers of the same network in parallel in a distributed fashion.

2 POPULAR ARCHITECTURES

AlexNet[1], which achieved state of the art top 5 error of *19.80 %* for the Imagenet Large Scale Image Recognition Challenge in 2012 trained 60 Million parameters. In subsequent years, VGG-16 [2] a 16 layer deep convolutional neural network achieved achieved state of the art top 5 error of *8 %* in 2014 trained 138 Million parameters. ResNet [3], used a 152 layer deep architecture and trained 60 Million parameters to achieve top 5 error of *6.16 %*. DeepMind's Alpha Go agent ran on 48 CPU's and 8 GPU's.

3 DATA SIZE

Youtube 8 Million video data set is one of the most popular datasets to train video classification algorithms. The total size of the dataset is 1.7 terabytes. The Imagenet Large Scale Image Recognition dataset 2012, used to train popular deep learning classification algorithms has a total of 22,000 classes and has a size of 138 gigabytes.

4 PARALLEL AND DISTRIBUTED ARCHITECTURES

4.1 Convolutional Neural Networks

Convolutional Neural Networks drive modern Computer Vision and Artificial Intelligence based research. The convolution operation involves sliding a filter of a predefined size over the input data and perform element-wise multiplication. They are capable of extracting higher level information from input data and project them to lower level embedding. The patterns identified in the lower level embedding can be used to perform various Machine Learning tasks such as classification, clustering, object recognition and source separation.

Parallelism of convolution operation. Every layer of a Convolutional Neural Network has a stacked input of filters. These filters are responsible for extracting higher level information from the input data. The filters operations are independently applied to the input data. This makes it possible to compute these operations in parallel to each other and collate their results. Recent advanced software architectures such as tensorflow and theano are capable of achieving computation in parallel using multiple cores. Additionally Graphical Processing Units can be explicitly programmed for parallel implementation of the convolution operator to achieve state of the art computational results.

4.2 Need For distributed approaches

Standard Convolutional Neural Networks have millions of parameters to train and optimize. Additionally the data required to train these systems ranges in Hundred's of Gigabytes. These computational constraints make it inefficient to train deeper networks on stand alone machines.

- Data Parallelism - When the data required to train neural networks exceed the systems storage capacity, it is required to distribute the data across multiple machines and introduce a data pipeline to feed input to the network.
- Model Parallelism - When the model being trained is too large to fit into the main memory. It is required to distribute different layers of the model across different machine and

use distributed variants of Stochastic Gradient Descent to update each layer being processed on different machines.

5 DATA PARALLELISM

Data parallelism involves storing the input data required to train our Convolutional Neural Network Model across multiple machines. Each machine runs the same network model. Each model is then trained on an unordered random subset of the data. One of the biggest challenges faced in data parallelism is updation of model parameters. These are broadly classified into 2 categories.

- Synchronous update - In synchronous updates, gradients are computed using the loss generated by each model on a mini-batch of the independent input. Weights are updated using a single gradient generated by averaging the losses of each model.

- Asynchronous update - In asynchronous updates, each model runs independently. Global parameters shared by multiple models are held in a global parameter server. Each model then fetches the updated parameters from the server to process the mini-batch

5.1 Synchronous Updates

Zinkevich, Weimer, Smola & Li, 2010 [4] introduced a parallel variant of the traditional Stochastic Gradient Descent algorithm. They designed a simple yet efficient algorithm which averaged the gradients generated by the multiple machines within the network. This method is shown to converge and provide an optimal speedup.

Algorithm 1 Parallel SGD ($\{c^1, ..., c^m\}, T, n, w_o, k$)

1: **for** machine $\in \{1....k\}$ in parallel **do**
2: $v_i = SGD(\{c^1, ..., c^m\}, T, n, w_o)$
3: $v = \frac{1}{k} \sum_{i=1}^{k} v_i$
4: *Return* v

5.2 Asynchronous Updates

Dean et. Al, 2012 [5] introduced an asynchronous variant of the traditional Stochastic Gradient. They proposed the use of a centralized communication server which holds parameters used by all models running in parallel. The communication server is distributed across several machines. Each model requests the centralized server for updated parameters before processing the mini-batch. Thus each model requests only those machines which holds parameters relevant to its partition. After computation of the gradient post processing the mini-batch the centralized server is updated with the new gradients. Subsequently the parameters are updated using the newly computed gradient. Asynchronous updates are more robust as compared to Synchronous updates. If a machine within the network fails, other machines are still up and computing their gradients.

6 MODEL PARALLELISM

Model parallelism involves training different layers of the Deep Neural Network in a distributed fashion across several machines in

Algorithm 2 Downpour SGD (p, d)

1: **for** machine $\in \{1....k\}$ in parallel **do**
2: *query updated parameters from server*
3: $v_i = SGD(p, d)$
4: *Update centralized server with* v_i
5: $p = p - \nabla v_i$

a network. In Model parallelism, different layers at the same level within the network are trained on the same input data. Model parallelism is required when the size of the network is too large to fit in main memory. Recent research in Deep Convolutional Networks is focused on the 'wider' paradigm instead of the traditional 'deeper' paradigm [6]. Wider Convolutional Networks can be viewed as a stack of smaller networks connected in parallel. Each of these smaller networks is designed and optimized to extract complex relationships in the input data at different depth levels. Wider Networks are computationally efficient than deeper networks. These smaller networks can be trained in parallel across multiple cores as these networks do not suffer from resource sharing. Each network in a layer gets its own copy of the output from the previous layer. A master layer is required to collate the results of the smaller networks to be passed to the next layer of the Network.

7 CONCLUSION

The number of parameters to train a neural network optimally have been increasing in the last few years. The data required to train these networks efficiently is continuously increasing. Standalone architectures are quickly being replaced with distributed architectures designed to handle training of these networks. Existing Industrial architectures can be tuned to train deep neural networks. They are optimal for training such networks with little to no additional cost of setup and expertise. With the techniques presented above, deeper architectures can be trained efficiently and optimally to achieve state of the art results.

REFERENCES

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*. 2012.

[2] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. URL http://arxiv.org/abs/1409.1556.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

[4] Martin Zinkevich, Markus Weimer, Lihong Li, and Alex J. Smola. Parallelized stochastic gradient descent. In *Advances in Neural Information Processing Systems 23*, pages 2595–2603. 2010. URL http://papers.nips.cc/paper/4006-parallelized-stochastic-gradient-descent.pdf.

[5] Jeffrey Dean, Greg S. Corrado, Rajat Monga, Kai Chen, and Matthieu et al Devin. Large scale distributed deep networks. NIPS'12, 2012.

[6] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014. URL http://arxiv.org/abs/1409.4842.

8 BIBTEX ISSUES

Warning–empty publisher in NIPS2012-4824

Warning–empty publisher in NIPS2010-4006

Warning–empty booktitle in Dean

(There were 3 warnings)

9 ISSUES

DONE:

Example of done item: Once you fix an item, change TODO to DONE

9.1 Bibliography Errors

Bibtex labels cannot have any spaces, _ or & in it

Citations in text showing as [?]: this means either your report.bib is not up-to-date or there is a spelling error in the label of the item you want to cite, either in report.bib or in report.tex

9.2 Formatting

HID not included in the keywords

9.3 Writing Errors

Do not use the word *I* instead use *we* even if you are the sole author

Errors in title, e.g. capitalization. For =¿ for

Are you using *a* and *the* properly?

9.4 Citation Issues and Plagiarism

It is your responsibility to make sure no plagiarism occurs. The instructions and resources were given in the class

Lack of citations in Introduction section and also in section 4

9.5 Structural Issues

The paper has less than 2 pages of text, i.e. excluding images, tables and figures

9.6 Details about the Figures and Tables

Figures, tables and algorithms should be referred to in the text

re