Optimal Lobotomy

Clement Lee ’17
advised by Jianxiong Xiao

1 Motivation and Goal

The recent and astronomical rise of deep learning algorithms applied to a variety of datasets has been built on the foundation of significant boosts to computational power, especially with the advent of GPGPU techniques. These developments have proceeded at a much faster rate than the speed of hardware-based technological advancements, and as such algorithms to efficiently utilize existing hardware are critically important. The goal of this project is thus to produce an algorithm and implementation which will aid in the automatic parallelization of deep neural networks across multiple GPUs. Our proposed algorithm would potentially greatly speed up any deep learning problem of contemporary nontrivial size, as it intends to automatically solve the many difficulties of properly creating parallelism within models with high degress of connectivity and interdependence. This would also be beneficial for deep learning problems where the learning parameter space and architectures are not studied extensively, as it could be applied as a general self-learning mechanism.

2 Problem Background and Related Work

Of primary importance to this project is the paper *Optimal Brain Damage* [2] which describes an algorithm to reduce the size of a feedforward neural network by removing elements which contribute little to its classification ability. It defines the concept of *saliency* which is the effective importance of a weight in a network. This should generally allow an algorithm to reduce the dimensionality of the network with minimal impact on its performance, which is a cornerstone of our algorithm.

An extensive amount of work exists in the literature regarding distributed deep learning, as it is generally a problem that resists parallelization. Especially in increasingly distributed systems, attaining sublinear results in scaling is itself an extremely difficult problem, which is described in more detail in *Large Scale Distributed Deep Networks* [1]. This gives a description of the overarching challenges of parallelization, and creates a framework for parallel learning.

Multiple forms of parallelism exist, as are described in *On Model Parallelization and Scheduling Strategies for Distributed Machine Learning* [3]. This paper lays out the differences between model and data parallelism, and discuss the difficulties of applying both in varying degrees to deep learning. It develops a language for efficient partitioning of training and updating parameters, particularly regarding scheduling of gradient calculations across a distributed network of participating servers. In general, we aim to build on its usage of *model parallelism*: that is, the splitting of model parameters across multiple computing devices.

The benefits of parallelism are further put into practice in *Reducing the Training Time of Neural Networks by Partitioning* [4]. This constructs a set of pretraining models which provide a natural partitioning of the parameters when they are later merged. This allows for better usage of GPUs especially when they are not utilized to their full capacity. A partitioning as utilized by this text is important, though it focuses on a slightly different formulation than what we intend to perform. At the same time, much of the methodology is broadly applicable to our problem space.

A key limitation is that most current work focuses on the training phase, and is reliant on careful parameter tuning on the part of the researchers. We aim to overcome the limitations of significant human involvement in training of deep learning, and utilize computer-based optimization procedures to determine architectural decisions of the network. We also want to work on networks that are already pretrained and developed, and focus on efficient partitionings that allow for model parallelism with minimal loss in classification performance.

3 Approach

To begin, we will be taking the previously defined concept of saliency and extending it to remove significant numbers of weights from a pre-existing and trained network. We want to gradually remove weights (or clusters of weights) with low saliency while repeatedly retraining the network. We expect that the performance of the network will decrease with fewer weights, so an acceptable error range will be established. Furthermore, we want to develop a metric for connectivity that we aim to generally minimize, in order to get a better partitioning of the network. At the same time, it is not the intention of this project to produce a complete partitioning of the network into two separate networks, as the result of this is not one network but rather a boosted set of networks.

Instead, our approach is to maintain some level of communication between the nodes, and to use this interconnectivity to establish a self-optimizing network that will spread across GPUs. As we are performing our experiments on GPUs all located within a single machine, moving data between nodes is a considerable cost but not largely unsurmountable unlike in distributed systems. Very few papers have been published on self-optimizing runtimes of networks, as it has largely been the realm of deep learning human experts. We believe that our proposed work represents a novel direction in the literature, and it remains a topic which deserves more investigation than it has received in the past.

4 Plan

This project is divisible into smaller relatively-contained modules which will be developed in sequence. Firstly, we want to test the performance of the Optimal Brain Damage algorithm on the scale of modern day networks. This concerns the application of the algorithm under different conditions than it was originally developed and tested; it is likely that modifications will have to be made to ensure the feasibility of the algorithm.

Once we have a good understanding of the basic algorithm, we will move on to developing the automatic partitioning. This will depend on a subalgorithm to choose which weights which generally are optimal to remove when considering a splitting of the model. The tradeoff between simply picking the least salient features and the least connected features to remove will be considered in depth, and we aim to use this in order to produce the first iteration of the algorithm.

If this is accomplished, we would also like to explore the possibility of constructing the network rather than simply trimming a preconstructed one. The first step to accomplishing this is to experiment with an algorithm that also allows gradually adding weights in tandem with the partitioning. This would allow us to achieve a more aggressive trimming if inter-partition connections could be established on the fly as necessary. This could later progress towards a network that starts out in a minimal state and gradually develops its own optimal architecture to spread over the GPUs available.

As we are aiming more towards an algorithmic problem, we aim to test on a dataset that is more commonly used. In the forefront here is MNIST, a large publicly available dataset of handwritten digits that represents a common benchmark for image classification. We plan to take a relatively standard ConvNet architecture, and apply our algorithm to perform modifications automatically.

5 Evaluation

The benchmarks on which this algorithm will be evaluated are more generally well-established. The classification performance of modern techniques on the MNIST dataset are documented in depth, so we will aim to remain in the same general accuracy as the networks we start with. The actual speed of running the dataset is also easily testable, so as a key component of the algorithm is creating parallelism, we will want to run it against the initial network, and see the kinds of gains we can get from varying numbers of GPUs. As noted prior, sublinear increases in performance are significant, and we aim to approach linear scaling by better parallelization. This step will be critical as the development of our algorithm will necessarily be informed and guided by the benchmarks and performance data that we collect.

References

[1] Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Mao, M., Senior, A., Tucker, P., Yang, K., Le, Q. V., et al. Large scale distributed deep networks. In *Advances in Neural Information Processing Systems* (2012), pp. 1223–1231.

[2] LeCun, Y., Denker, J. S., Solla, S. A., Howard, R. E., and Jackel, L. D. Optimal brain damage. In *NIPs* (1989), vol. 89.

[3] Lee, S., Kim, J. K., Zheng, X., Ho, Q., Gibson, G. A., and Xing, E. P. On model parallelization and scheduling strategies for distributed machine learning. In *Advances in neural information processing systems* (2014), pp. 2834–2842.

[4] Miranda, C. S., and Von Zuben, F. J. Reducing the training time of neural networks by partitioning. *arXiv preprint arXiv:1511.02954* (2015).