Master Thesis

On Scalable Deep Learning and Parallelizing Gradient Descent

Joeri R. Hermans

Thesis submitted in partial fulfillment of the requirements
for the degree of Master of Science of Artificial Intelligence

at

Maastricht University
Faculty of Humanities and Sciences
Department of Data Science & Knowledge Engineering
Maastricht, The Netherlands

Preface

This thesis is submitted as a final requirement for the Master of Science degree at the Department of Data Science & Knowledge Engineering of Maastricht University, The Netherlands. The subject of study originally started as a pilot project with Jean-Roch Vlimant, Maurizio Pierini, and Federico Presutti of the EP-UCM group (CMS experiment) at CERN. In order to handle the increased data rates of LHC Run 3 and High Luminosity LHC, the CMS experiment is considering to construct a new architecture for the High Level Trigger based on Deep Neural Networks. However, they would like to significantly decrease the training time of the models as well. This would allow them to tune the neural networks more frequently. As a result, we started to experiment with various state of the art distributed optimization algorithms. Which resulted in the achievements and insights presented in this thesis.

I would like to express my gratitude to several people. First and foremost, I would like to thank my promotors, Gerasimos Spanakis, and Rico Möckel for their expertise and suggestions during my research, which drastically improved the quality of this thesis. Furthermore, I would also like to thank my friends, colleagues and scientists at CERN for their support, feedback, and exchange of ideas during my stay there. It was a very motivating and inspiring time in my life. Especially the support and experience of my CERN supervisors, Zbigniew Baranowski, and Luca Canali, was proven to be invaluable on multiple occasions. I would also like to thank them for giving me the personal freedom to conduct my own research. Finally, I would like to thank my parents and grandparents who always supported me, and who gave me the chance to explore the world in this unique way.

Joeri R. Hermans
Geneva, Switzerland 2016 - 2017

Abstract

Abstract here.

Summary

Summary here.

Contents

Abbreviations and Notation

η	Static learning rate
η_t	Learning rate with respect to time t.
λ	Communication period, or frequency of commits to the parameter server.
$\mathcal{L}(\theta\ ;\ \mathbf{x})$	Loss function with respect to parametrization θ and input $\mathbf{x}$.
$\nabla f(\theta\ ;\ \mathbf{x})$	Gradient of f with respect to parametrization θ and input $\mathbf{x}$.
τ	Staleness
θ_t^k	Parametrization of worker k at time t.
$\tilde{\theta}_t$	Center variable, or central parametrization maintained by the parameter server.
$\triangleq$	Is defined as
$J(\theta)$	Loss with respect to parameterization θ.
m	Mini-batch size
n	Number of parallel workers.
ADAG	Asynchronous Distributed Adaptive Gradients
ASGD	Asynchronous Stochastic Gradient Descent
CERN	European Organization for Nuclear Research
CMS	Compact Muon Solenoid
EASGD	Elastic Averaging Stochastic Gradient Descent
GD	Gradient Descent
HEP	High Energy Physics
HL-LHC	High Luminosity Large Hadron Collider
LHC	Large Hadron Collider
MNIST	Mixed National Institute of Standards and Technology database
PS	Parameter Server
SGD	Stochastic Gradient Descent

Chapter 1

Introduction

In this chapter we introduce the main concept, and problems surrounding the parallization of gradient descent. We famillliarize the reader with the topic and some notation by providing some context why someone would like to apply said technique. Furthermore, in Section 1.4, we summarize the problem statement and provide several research questions which will guide the research in this work. Finally, we conclude this chapter in Section 1.5 with a brief outline of the thesis.

1.1 Motivation

In recent years it has been shown that being able to train large and deep neural networks result in state-of-the-art performance [10, 3], especially regarding unsupervised feature learning and image recognition. However, consider the required time, and cost of the infrastructure that would be required in order to train such a large model in a reasonable amount of time. Furthermore, it is not only the training time and cost of the infrastructure which need to be taken into consideration, but also the volume of the data. The amount of information that will be gathered will be an increasing important factor in the next few years. Not only with respect to big technology companies and government organizations, but also scientific surveys with limited budgets. These scientific surveys will generate more experimental data than ever [1, 5], and will have to process and analyze that data. To solve the problem of increased computational workloads and budget freezes, the High Energy Physics (HEP) community is exploring and researching machine learning approaches to fit physics problems [2, 9, 7] with the intention to improve detection quality, or reduce computational constraints.

However, the sheer size of these datasets severly impacts the training time of the models. In order to resolve this issue, one could sample some representative subset of the data in order to reduce the training time. The disadvantage of this approach is that is possible that some "diverse" instances, i.e., data points, do not appear in the final training set. This is especially a problem in Deep Learning, where models usually benifit from having access to a lot of training data due to the high dimensionality of the parametrization [3]. In order to resolve this issue, Dean et al. [3] introduce two new paradigms to decrease the training time in the prescense of a large dataset and a high dimensionality of the parametrization. The two paradigms, *Model Parallelism*, briefly discussed in Section 1.2, and *Data Parallelism*, discussed in Section 1.3, are inherently different ways of decreasing the training time of a model.

The first paradigm, *Model Parallelism*, is the intuitively most straightforward one since it deals with the parallelization of the computations within a *single* model, i.e., how to parallelize the computations of a single model over multiple machines, or multiple processes. The second paradigm, which will be the main focus of this thesis, is *Data Parallelism*. As stated above, the main concept of Data Parallelism will be discussed in detail in Section 1.3. However, for completion, think of Data Parallelism as a technique to *parallelize gradient descent*. This is done by allocating n processes over possibly n different machines, and splitting the training set into n *partitions*, or *shards*. For further convenience, we will call such a process a *worker*. In the next step, we assign a single distinct partition to a

worker. Meaning, the worker will not be able to fetch training data from other partitions since those have been assigned to different workers. The goal of these workers is to work together, and optimize the parameters of a central model. A lot of different distributed optimization schemes have been suggested in recent years [**??**, 11, 3, 4]. Most of the recent contributions try to push the limits of (asynchronous) Data Parallelism by simply *annealing* the gradients with respect to some hyperparameter without giving a sound theoretical argument why the annealing should happen in the first place.

Recently, new theoretical work supported by experimental evidence was introduced by Mitliagkas et al. [8]. We will discuss this work in detail, and provide some supporting experimental evidence in Section 3.2.1.

1.2 Model Parallelism

1.3 Data Parallelism

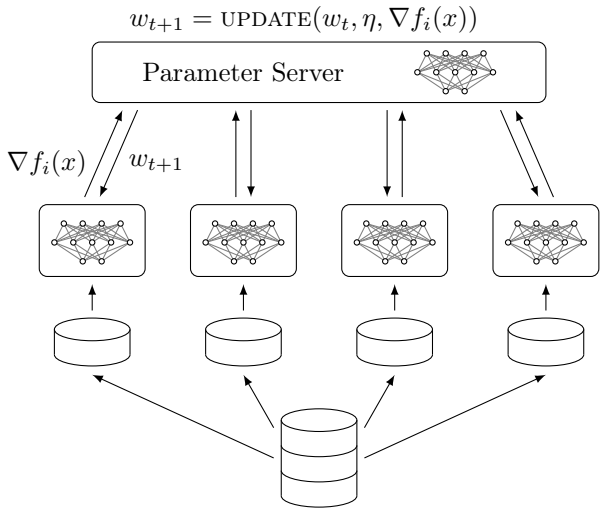

Figure 1.1: Schematic representation of a data parallel approach. In this methodology we spawn n workers (not necessarily on different machines), and assign a data shard (partition) of the dataset to every worker. Using this data shard, a worker i will iterate through all mini-batches to produce a gradient, $\nabla f_i(x)$, for every mini-batch x. Next, $\nabla f_i(x)$ is send to the parameter server, which will incorperate the gradient using an UPDATE mechanism.

1.4 Problem Statement

1.5 Thesis Outline

Chapter 2

Optimization Algorithms

Chapter 3

Distributed Deep Learning

3.1 Synchronous Data Parallelism

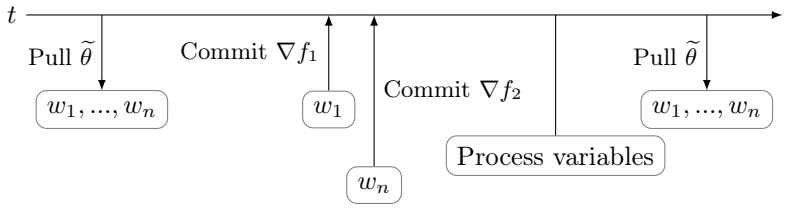

Figure 3.1: Caption here

3.1.1 Model Averaging

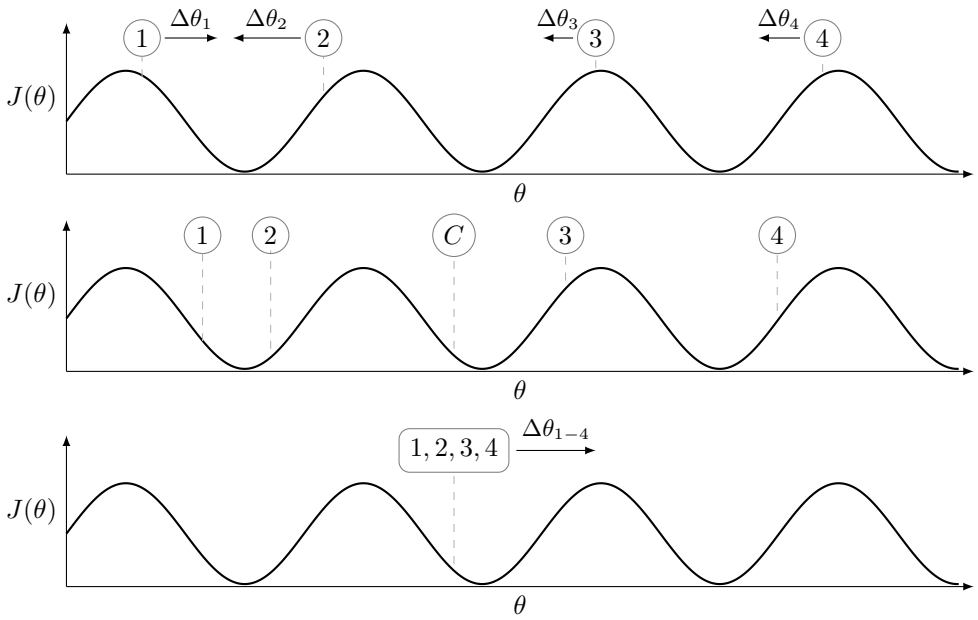

Figure 3.2: Caption here

3.1.2 Elastic Averaging SGD

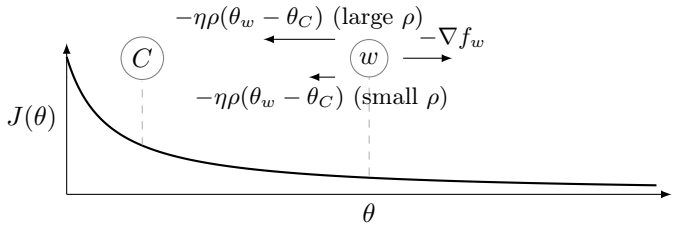

Figure 3.3: EASGD Caption here

3.2 Asynchronous Data Parallelism

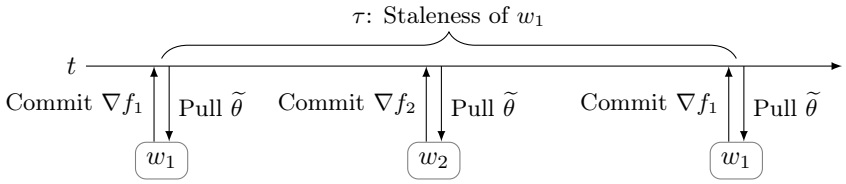

Figure 3.4: Caption here.

3.2.1 Asychrony Induced Momentum

3.2.2 Hogwild!

3.2.3 DOWNPOUR

3.2.4 Asynchronous Elastic Averaging SGD

Chapter 4

Accumulated Gradient Normalization

$$\Delta\theta = -\frac{\sum_{i=0}^{\lambda} \eta_t \frac{1}{m} \sum_{j=0}^{m-1} \nabla f(\theta_i; x_{ij}; y_{ij})}{\lambda} \tag{4.1}$$

Chapter 5

Asynchronous Distributed Adaptive Gradients

In this chapter we introduce a novel optimizer called ADAG. ADAG, or *Asynchronous Distributed Adaptive Gradients*, is an optimization process designed with data parallel methods in mind. We build upon previous work [3, 4, 6, 11] and incorperate new insights backed up by theory and experimental evidence. We start in Section 5.1 by formalizing the problem setting. In Section 5.2, we summarize previous work on distributed (data parallel) optimization. Section 5.3 will describe our algorithm in detail, supported by intuition and theory. Finally, we experimentally show the effectiveness of our approach in Section 5.4 and give some points for future work in Section 5.5.

Chapter 6

Distributed Keras

Chapter 7

Experiments

Chapter 8

Conclusion

Bibliography

[1] G Apollinari et al. *High-Luminosity Large Hadron Collider (HL-LHC): Preliminary Design Report*. Geneva: CERN, 2015. URL: https://cds.cern.ch/record/2116337.

[2] Jianming Bian. "Recent Results of Electron-Neutrino Appearance Measurement at NOvA". In: *arXiv preprint arXiv:1611.07480* (2016).

[3] Jeffrey Dean et al. "Large scale distributed deep networks". In: *Advances in neural information processing systems*. 2012, pp. 1223–1231.

[4] Stefan Hadjis et al. "Omnivore: An optimizer for multi-device deep learning on cpus and gpus". In: *arXiv preprint arXiv:1606.04487* (2016).

[5] Zeljko Ivezic et al. "LSST: from science drivers to reference design and anticipated data products". In: *arXiv preprint arXiv:0805.2366* (2008).

[6] Diederik Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[7] Gilles Louppe, Michael Kagan, and Kyle Cranmer. "Learning to Pivot with Adversarial Networks". In: *arXiv preprint arXiv:1611.01046* (2016).

[8] Ioannis Mitliagkas et al. "Asynchrony begets Momentum, with an Application to Deep Learning". In: *arXiv preprint arXiv:1605.09774* (2016).

[9] Luke de Oliveira, Michela Paganini, and Benjamin Nachman. "Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis". In: *arXiv preprint arXiv:1701.05927* (2017).

[10] Yonghui Wu et al. "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation". In: *arXiv preprint arXiv:1609.08144* (2016).

[11] Sixin Zhang, Anna E Choromanska, and Yann LeCun. "Deep learning with elastic averaging SGD". In: *Advances in Neural Information Processing Systems*. 2015, pp. 685–693.

Appendices