Model-based Reinforcement Learning in Computer Systems

Sean J. Parker
Clare Hall

UNIVERSITY OF CAMBRIDGE

A dissertation submitted to the University of Cambridge in partial fulfilment of the requirements for the degree of Master of Philosophy in Advanced Computer Science

University of Cambridge
Computer Laboratory
William Gates Building
15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom

Email: sjp240@cam.ac.uk

March 28, 2021

Declaration

I, Sean J. Parker of Clare Hall, being a candidate for the M.Phil in Advanced Computer Science, hereby declare that this report and the work described in it are my own work, unaided except as may be specified below, and that the report does not contain material that has already been used to any substantial extent for a comparable purpose.

Total word count: 0

Signed:

Date:

Acknowledgements

Abstract

Write a summary of the whole thing. Make sure it fits in one page.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Chapter 2

Background and Related Work

2.1 Introduction to Deep Learning Models

This section discusses the way in which machine learning models are represented for efficient execution on physical hardware devices. First, we discuss how the mapping of tensor operations to computation graphs is performed followed by an overview of recent approaches that optimise computation graphs to minimise execution time.

Over the past decade, there has been a rapid development of various deep learning architectures that aim to solve a specific task. Common examples include convolutional networks (popularised by AlexNet then ResNets, etc), transformer networks that have seen use in the modelling and generation of language. Recurrent networks that have shown to excel at learning long and short trends in data.

Importantly, the fundamental building blocks of the networks have largely remained unchanged. As the networks become more complex, it becomes untenable to manually optimise the networks to reduce the execution time on hardware. Therefore, there is extensive work in ways to both automatically optimise the models, or, alternatively apply a set of hand-crafted optimisations.

Computation graphs are a way to graphically represent both the individual tensor operations in a model, and the connections (or data-flow) along the edges between nodes in the graph. Figure [TODO] shows how a perceptron computing, $y = \texttt{ReLU}(\mathbf{w} \cdot \mathbf{x} + b)$, can be represented graphically.

Similarly, the whole model can be converted into a stateful dataflow graph in this manner. By using a stateful dataflow (or computation) graph, we can use any optimisation technique for backpropogation of the model loss though the graph [TODO rewrite last sentence]. We consider two key benefits of this representation. First, we can execute the model on any hardware device as the models have a single, uniform representation. Secondly, it allows for pre-execution optimisations based on the host device, for example, we may perform different optimisations for executing on a GPU compared to a TPU.

2.1.1 Current approaches to optimising deep learning models

Tensorflow [1], a common machine learning framework is designed to greedily apply a set of pre-defined substitutions to an input graph in an attempt to optimise the graph. Tensorflow made use of low-level libraries such as cuBLAS [2] for optimised matrix operations and cuDNN [3] for convolutional kernels. Furthermore, Tensorflow also contains a set of 155 substitutions that are implemented in 53,000 lines of code; to complicate matters, new operators are continuously proposed, such as grouped or transposed convolutions, all of which leads to a large amount of engineering effort required to maintain the library.

TensorRT and TVM

Metaflow and TASO

2.2 Reinforcement Learning

Reinforcement learning (RL) is a field in machine learning, broadly, it aims to compute a control policy such that an agent can maximise its cumulative reward from the environment. Formally, RL has its foundations in Markov Decision Processes (MDPs) which are represented as a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}_a, \mathcal{R}_a \rangle$ where:

- $\mathcal{S}$, is a finite set of states
- $\mathcal{A}$, is a finite set of actions
- $\mathcal{P}_a$, is the state transition probability that an action a in state s_t leads to a state s'_{t+1}
- $\mathcal{R}_a$, is the reward from the environment after taking an action a between state s_t and s'_{t+1}

In the following sections we discuss the two key paradigms that exist in Reinforcement learning and the current research in both areas.

2.2.1 Model-Free

2.2.2 Model-Based

2.3 Graph Neural Networks

Chapter 3

Title

Chapter 4

Title

Chapter 5

Title

Bibliography

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] CUDA Nvidia. cuBLAS Library. `https://developer.nvidia.com/cublas`, 2008.

[3] Sharan Chetlur, Cliff Woolley, Philippe Vandermersch, Jonathan Cohen, John Tran, Bryan Catanzaro, and Evan Shelhamer. cudnn: Efficient primitives for deep learning, 2014.