Academic Year 2022 – 2023
Research Unit: UCLA - MQST
Team(s): Ryan, Arvind, Payal, and Will
Supervisor(s): Tony Nowatzki

CS 259: Final Project
Tensor Networks for Machine Learning

Ryan, Arvind, Payal, and Will

June 16, 2023

Abstract

For our project we chose to explore the methods of the Google X paper 'Tensor Networks for Machine Learning' by Jack Hidary et al. Here, the authors explore the use of tensor networks to improve the training performance of classifier neural networks. This is done by using tensor networks to compress the weights of a single layer within the neural network, which reduces the number of parameters that need to be trained. This reduces the total amount of compute needed to train the network at the cost of slight model accuracy. The authors were able to achieve a performance increase of nearly 10x on the MNIST dataset (through reduction of the total number of trainable parameters), with a loss of roughly 2% accuracy. We looked into methods to reproduce their results by using a similar tensor network representation of a single hidden layer. We relied on GPU parallelization provided to us through various Python libraries to speed up the training process. This parallelization was primarily achieved through Matrix Product State (MPS) contraction over multiple independent site indices. We benchmarked results for a simple binary classifier on random image data as well as a 10 digit classifier for the MNIST image dataset. We observed similar results to the authors and explored additional methods to further their results by varying the MPS layer bond dimension and the number of site indices.

Modeling the weights of the neural network as a tensor network also allows for the use of parallelism to contract the network (multiply to get the desired result).

Contents

1 Introduction

1.1 MNIST Dataset

2 Methods

2.1 MPS Layer Construction

In an attempt to first understand the concepts behind representing classifier layers as MPS, we turned to the quimb library in Python. Quimb is a tensor network library used in the quantum simulation and quantum algorithm's community. The library provides simple methods to construct tensor networks, and more specifically, matrix product states. We used the library to construct a simple MPS layer with 9 site indices as a proof of concept shown in Figure 4.

The MPS layer is constructed by first initializing the site tensors (also called site indices) with random values just as we would for the typical weights of a neural network. We then choose a bond dimension between each of the site indices, which will determine how 'dense' this MPS layer is, or in other words, how many weights are within the layer. In this demo model, we chose a bond dimension of 2, which means that each site index will have two values associated with it.

The next step to this process is supplying input into our MPS. This is done by reshaping the input data into a set of vector tensors in the form [1-p, p], where p is normalized the pixel value or normalized incoming data value. The reason this is done is outlined Section II.A of Efthymiou, Hidary, Leichenauer. This encoding format does of course double the encoding size of our image data, but it allows for our neural network to have an MPS encoded layer – which will reduce the amount of weights needed and improve our training time. An example of this image encoding attached to an MPS layer is shown in Figure 5.

References

[1] Stavros Efthymiou, Jack Hidary, and Stefan Leichenauer. Tensornetwork for machine learning, 2019.

[2] Chase Roberts, Ashley Milsted, Martin Ganahl, Adam Zalcman, Bruce Fontaine, Yijian Zou, Jack Hidary, Guifre Vidal, and Stefan Leichenauer. Tensornetwork: A library for physics and machine learning, 2019.