Parallel Autoencoder

High Performance Computing for Data Science

Silvanus Bordignon
xxxxxx
Univesity of Trento
Trento, Italy
maedje@typst.app

Ettore Sasggiorato
247178
University of Trento
Trento, Italy
ettore.saggiorato@studenti.unitn.it

***Abstract*—TODO**

***Index Terms*—HighPerformanceComputing, DeepLearning**

I. Introduction

Aim of the project. Introduce that we use a small neural network, so the network won't be split between multiple nodes.

A. *Instruction for Reproducibility and Building*

cmake. runs on linux. Needs GCC and C++20.

II. State of the Art / Related Works

III. Libraries and Datasets

A. *Libraries*

- stb
- Eigen
- Gtest
- GBench

B. *Datasets*

IV. Methodology and Implementation Details

The structure of the operations of nn is fundamental, our objective is to maximize the TOPS.

- Critical thought about the cost of allocating memory and the usage of the heap/stack -> reference to the appendix
- Considering that Eigen can parallelize the workload thoughts about avoiding too much context switch on the CPU -> reference appendix
- Difference between when one has a GPU and doesn't in the dataloader
- Why we are doing distributed training instead of e.g. sharing the model between machines (Efficiency of course :)). Reference a few papers plz

We planned to work sequentially:

1) Basic implementation single threaded
2) Activating Eigen's OpenMP parallelization
3) OpenMP the rest of hte code
4) MPI -> check how to use infiniband/omnipath
5) MPI + OpenMP

Say that methods are benchmarked individually to see where we gain the best perforamance and where it becomes worse.

A. *Unit Testing*

Performed to be sure that modules work correcly and that when parallelizing/using mpi nothing breaks.

B. *Basic Implementation*

- Why many parts of the NN are set as a template library.
- APIs inspired from PyTorch python's apis.

——————————————-

- Dataloader
- Linear layer
- ReLU
- Sigmoid
- Encoder/Decoder
- Loss
- Backpropagation -> reference from the book (need to get it lol)
- Gradient Descent

C. *Eigen parallelization*

D. *OpenMP*

E. *MPI*

> How we are parallelizing on multiple nodes > What we are sharing and Why > ABLATION on the distributed weights

F. *Combo: MPI + OpenMP*

V. System Description

VI. Experiments

A. *Evaluation*

What we evaluate:

- Speedup
- Efficiency
- Scalability
  - Strong
  - Weak

a) *What we expect:*

B. *TODO: experiment combos and their results compared.*

What we can learn from this, how it goes.

VII. Tables and Data

VIII. Additional Listings

References

Appendix A.

“asd”, adwaoids

Appendix B. If needed we can add a title to the appendix :)

- Ablations about where perofmrances explode + some thougts about the design process
- heap vs stack with Eigen (`Matrix <float, ...>` vs `MatrixXf`)
- using vs not using eigen parallelization: it's parallelization (single loop) on a set of data vs a possible openmp done by us where data is worked on in parlalel