Jerry Lingjie Mei
UROP Faculty Supervisor: Alan Edelman
UROP Direct Supervisor: Valentin Churavy
Term: 2017-2018 Spring
Feb. 14, 2018

Generic and Efficient Convolutions in Julia for Machine Learning on Non-traditional Numeric Types

Project Overview

Many machine learning research and applications are based on specific framework, including but not limited to Tensorflow, MXNet and Pytorch for Python 3, Caffe 2 for C++, and ConvNetJS for javascript. There are similar libraries built for Julia, including KNet[1] and Flux[2]. Julia is a high-level, high-performance dynamic programming language for numerical computing. Originated from MIT, it provides a sophisticated compiler, distributed parallel execution, numerical accuracy, and an extensive mathematical function library[3]. Julia is also developing multiple-dispatch mechanism for non-traditional data types, including half-precision [4] and fixed point numbers[5]. This research is aimed at making convolutions in multiple data types faster and more efficient in Julia.

This research is supervised by Alan Edelman, and under direct instructions by Valentin Churavy. The majority of the work will be coding and reading papers in 32-785, as well as running experiments on MIT cloud services.

Personal Rule & Responsibilities

My research directions will include the following:

- Implementing convolution layer using only Julia language;
- Enabling multiple data types, including Float16 and fixed point numbers as weights in a layer;
- Using GPU to accelerate computation of convolution layer;
- Compare the computation speed between Julia using this framework and other frameworks;

Optional tasks may include:

- Using normal numbers to denote the weight of the network;

- Using faster convolutional algorithms to accelerate convolutional layer computation;
- Represent weight matrix in terms of sparse matrices;

I will be working around 8 hours a week. I will be either reading paper related to the research or writing codes to implement the algorithms, either on weekdays or on weekends.

Goals

The result of this research will include non-traditional numeric data types as weights of convolution layers, and using GPU to accelerate its computation. All of the code will be represented entire in julia so that the compiler can recognize it.

The result of this research may be pushed onto GitHub so that anyone can download it and call it, just like any other Julia libraries. If the speed of the implemented library is faster compared to other libraries, we may also publish it in terms of paper.

As the result of this project can become open-source and available for Julia users all around the world, it will promote Julia as an option of fast and efficient language to industrial users and other like-minded researchers. Julia is on its way to version 1.0, and additional utility packages will help its acceptance among the user groups.

Personal Statement

I got familiar with Julia when I was taking 18.S096 in the last IAP, which was taught by Alan Edelman, Valentin Churavy and a few other lecturers. Julia is highly versatile language that I can write code in it with relative ease. I am also interested in deep learning, which can do so many amazing tasks using some simply designed network. Both these two factors drive me into this research.

Throughout the research process, I am hoping to learn how to optimize fast and stable code in Julia, as well as how to design algorithms using GPUs. I may also learn how to cooperate with other coders on GitHub by using different branches and sharing comments.

References

[1] Deniz Yuret. Knet: beginning deep learning with 100 lines of julia. In *Machine Learning Systems Workshop at NIPS 2016*, 2016.

[2] Fluxml/flux. https://github.com/FluxML/Flux.jl.

[3] The julia language. https://julialang.org/.

[4] Programming tensor cores in cuda 9. https://devblogs.nvidia.com/programming-tensor-cores-cuda-9/.

[5] Fixedpointnumbers. https://github.com/JuliaMath/FixedPointNumbers.jl.