High-Performance Computing:

Optimizing Performance for Optical Character Recognition via Neural Networks

Takao Yamada ([tyamada@bu.edu](mailto:tyamada@bu.edu)) Graduate Electrical & Computer Engineering

Mikhail Andreev ([mikh@bu.edu](mailto:mikh@bu.edu)) Graduate Electrical & Computer Engineering

May 8, 2015

Table of Contents

Abstract

Neural networks have been shown to have the ability to perform character recognition given a set of training images. Using these concepts, we seek to optimize the code execution using hardware capabilities of a CPU and GPU. Using techniques including loop unrolling, threading, and GPU partitioning, we analyze the effectiveness of these methods in utilizing a larger percentage of the hardware to perform expensive calculations.

Introduction

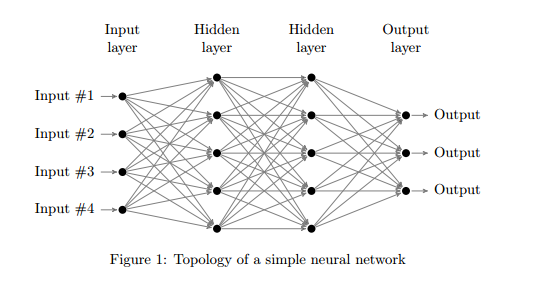
Character Recognition

Optical Character Recognition involves reading in images of characters and classifying them as a given character. This is done by looking at the features present in the character images and comparing them to the features present in training samples. After enough comparisons are made, the new image can be designated as a specific character. The method with which we perform the classification is through the use of a neural network.

Neural Networks

Neural networks act as interconnected graphs modeling neural activity found in our brain. The neurons are arranged in layers, which are connected to each other through edges. The edges have a specific weight associated with them. This weight forms the associative connection between neurons of different layers. By training the neural network, the weight of this connection will increase or decrease based on the association of that particular feature with the desired result. As a simplistic example, we can consider that each neuron calculates a particular feature of the input image, and then compares it to the same types of features present in the training images. If this feature is a good indicator of how the input image should be classified, the weight of that edge will be strengthened. Otherwise, it will be weakened and that feature will have a smaller impact on the image classification.

The anatomy of neural network consists of several layers, each of which has neurons in it. The first layer is known as the input layer, and the last layer is the output layer. Every other layer is a hidden layer which performs further computations. Each neuron in a layer has a weighted edge coming to it from every neuron in the previous layer, and has an edge going to every neuron of the following layer.



Images are passed though the network using a feed-forward mechanism which performs a calculation based on the weights of a node and then passes the results on to the next layer. These calculations can be summarized with the equation:

m is the number of neurons in each hidden layer, wk,j is the weight of an edge (k,j) and ak is the input to the system. stands for the sigmoid function, used to approximate the activation threshold:

To train the network with input samples we use a method called back-propagation. This involves first calculating the feed-forward results of the system. After this, since we know the correct result, we compare that with the calculated value, and use the error generated through that calculation to adjust the weights of the edges starting with the output layer, and propagating backwards through the system.

Method

Our method for optimizing the performance of the neural network is to use techniques that allow the available hardware to be fully utilized to perform calculations.

Base Case

Describe the base case – feed forward and back propagation.

Single Core Optimizations

Loop unrolling and multiple accumulators

Multiple Core Optimizations

Openmp and pthreads

GPU Optimizations

Partitioning, coalescing.

Results

Future Work

Conclusion

References

Code Description