# Project Report

Benchmarking densely connected neural networks on MNIST dataset

High Performance Architecture CS 243

Authors: Akshay Sudhir Bapat, Ananthakrishnan Pushpendran, Kanika Baijal

**Table of contents:**

1. Introduction
2. Related work
3. Proposed work
4. Analysis
5. Observations
6. Results
7. Conclusions and future scope
8. References
9. Introduction

A neural network is a network of simple elements called neurons which receive input, change their internal state (activation) according to that input, and produce output depending on the input and activation. The network forms by connecting the output of certain neurons to the input of other neurons forming a directed and weighted graph.

Different components of a neural network:

Neurons

Connections and weights

Propagation function

Learning rule

Neural network in ML

It’s a technique for building a computer program that learns from data. It is based very loosely on how we think the human brain works. First, a collection of software “neurons” are created and connected together, allowing them to send messages to each other. Next, the network is asked to solve a problem, which it attempts to do over and over, each time strengthening the connections that lead to success and diminishing those that lead to failure.

1. Related work
2. Proposed work

The objective of this project is to benchmark a densely-connected neural network using stochastic gradient descent on MNIST dataset on TensorFlow. In this regard, we have decided to focus on certain performance measures – namely, computation time, memory required, testing accuracy. We will determine the optimal values of batch size and number of iterations (epochs) so that we can get the best possible performance. Furthermore, we will evaluate the performance on different tensor types – half-precision (fp16), single precision (fp32) and double precision (fp64). Also, we will compare performances on different devices, including both CPUs and GPUs.

Devices:

* Nvidia GTX 1050 Ti
* Nvidia GT 940M
* CPU: Intel i5 7287U – 3.1 GHz

Performance measures:

1. Accuracy vs batch size
2. Accuracy vs epochs
3. Computation time vs batch size
4. Computation time vs epochs
5. Memory required vs batch size
6. Memory required vs epochs
7. Analysis
8. Testing accuracy vs batch size

Stochastic gradient descent methods are algorithms of choice in Machine Learning. These methods involve using small batches wherein a fraction of the training data, usually 32-512 data points, to compute an approximation to the gradient. It has been observed that if a larger batch is used, there is a significant degradation in the performance of the model. Having a batch size which is too large results in poorer generalization while small-batch methods usually generalize well. Inherent noise in the gradient estimation is widely considered to be the reason for this effect.

1. Testing accuracy vs epochs

In a neural network, both the training and testing accuracy usually improves with epochs. However, as you increase the number of epochs, even though you increase the training accuracy, you might end up overfitting the data. In such a case, the testing accuracy will decrease after a point. That is, testing accuracy will steadily increase till a point, after which it might start decreasing. This usually means that the network has learnt the input data too well and has lost the ability to generalize. Thus, we usually take the number of epochs as the value where the testing accuracy is the maximum, or in other words, testing error is the minimum.

1. Computation time vs batch size

Computation time generally drops with batch size. The larger the batch size, the less number of independent computations are needed during training. Higher batch size implies that the gradient is a better approximation of the actual gradient and thus will converge faster. Thus, after a few epochs, the calculations will be trivial and the code will run faster.

1. Computation time vs epochs

The relation between computation time and number of epochs is usually linear, that is, computation time is approximately directly proportional to the number of epochs.

1. Memory required vs batch size

There is no direct relation between the memory (RAM) required and the batch size. It is expected that memory required will be the same regardless of batch size.

1. Memory required vs epochs

There is no direct relation between the memory (RAM) required and number of epochs. In fact, increasing epochs just improves the accuracy while using the same memory. Thus, it is expected that memory required will be the same regardless of the number of epochs.

1. Observations

We are working on comparing tensor types fp32, fp64. We tried fp16 but there is no advantage due to lack of GPU support for consumer cards. [Link](https://www.reddit.com/r/MachineLearning/comments/6p8hcs/d_floating_point_precision_in_deep_learning/)

We are benchmarking it across multiple systems (CPUs/GPUs). Also, we found an interesting thing - there is a certain range for batch size in which we get the highest accuracy for MNIST dataset.

1. Results
2. Conclusions and future scope
3. References

* Keskar, Nitish Shirish, et al. "On large-batch training for deep learning: Generalization gap and sharp minima." *arXiv preprint arXiv:1609.04836* (2016).