The effect of Parallel Processing on Deep Learning

Author
Amer Rez
7656 Normandy Way
Cupertino, CA 95014
812-878-2168
amer.rez@sjsu.edu

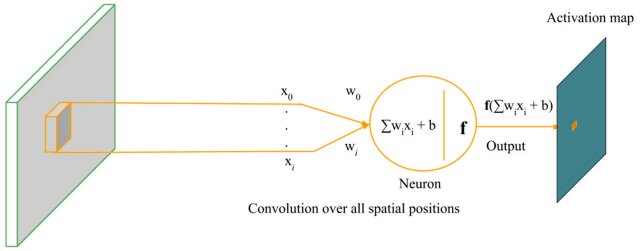
ABSTRACT
Deep learning networks performance on a CPU is definitely different from its performance on a GPU. This paper shows a neural network of the type CNN that is used for images classification and how it performs on different framework and hardware.

1. INTRODUCTION
Deep learning uses multiple layers within the network other than the input and the output layers. Each layer has multiple neurons. Each neuron has its own weights and does its own calculations. These many calculations theoretically could utilize many cores as they do part of their computations independently. This paper will show experiments running the same neural network on three different combinations of Tensorflow framework and hardware.

2. Neural Network Description

2.1 Network Layers and Neurons
CNN networks use convolutional operations to calculate their neurons output. That is where they got the name Convolutional Neural Networks. The network in this paper is designed with four layers and with seven layers to show the different performance behavior on each architecture. Tensorflow is the framework used to implement it.

A neuron is the main component on building a neural network. It takes and input from the data directly if it is in the input layer or it takes its input from the previous layer. It does a mathematical operation that is a matrix multiplication in general cases or a convolutional multiplication in the case of CNN networks. After that it adds a constant value that is called a bias to make the network more similar to real life scenarios.



The two types of networks that was used are a four-layer network and a seven-layer networks. In the four-layer network, the two hidden layers had 16 and 120 neurons. In the seven layer networks the number of neurons that was used in the hidden layers 2,3,4,5,6 respectively 20, 40,80, 100, 120. The reason why we don't specify the number of input and output layers is because that the input layer neurons depend on the data we input and put put of the last layer only refers to what class of images our image example belong to.

The code that builds this neural network and data generator is provided in the references in my repo conv-nn-zener-images which is shown in the references section.

2.2 Regularization and Optimization
Training a Deep learning network has many concepts and methods related to optimize the training.

Cost Function is the difference between what the model outputs and what it should be outputting. **Regularization** is adding a parameter to the loss function in order to get it closer to the right desired output. **Cross Validation** is training and testing multiple times on different data sets to finally get multiple results for training and testing. Aggregating these multiple results would give more stable results. **Testing Data** it is the data that is used to test the model after training is done. It has to be completely different than what was used during training in order to have a real fair test.

Tensorflow doesn't actually allocate the layers and the neurons (weights) when defining them. It actually does what is called a symbol to symbol allocation. Eventually when the variables and weights are actually needed in a computation, Tensorflow does allocate it. **Accuracy** is the number of times that the model guess the output correctly divided by the number of test data used.

3. Running The Neural Network
Tensor flow could be configured to utilize a CPU or a GPU. Tensorflow uses the CPU differently. While a more sophisticated installation which includes compiling the source code on the machine locally would utilize more instructions than a simple installation. Other than these two CPU setups there is a third one that uses a GPU.

Neural networks require a bug amount of computations during a phase called neural network training. The below sections show the training process on each of the three hardware and software setups.

All the experiments are done on a 1000 images generated randomly using a zener generator that we build specifically for this experiment. The training process feeds these images to the input layer in batches through iterations that are called epochs

3.1 Core i5

A simple Tensorflow installation would do a simple CPU utilization which uses only the common CPU instruction set

The CPU and graphics card on the laptop used were: 2.7 GHz Intel Core i5. Intel Iris Graphics 6100 1536 MB.

Using a personal laptop MacBook Pro (Retina, 13-inch) with Core i5 CPU and a Simple installation of Tensorflow. The training times and description are shown in Table 1.

Table 1. Training performance for Core i5

Epochs	Layers	Tensorflow	Time
100	4	CPU	15.9 minutes
50	7	CPU	9.3 minutes
25	4	CPU	3.4 minutes
25	7	CPU	4.7 minutes
50	4	CPU	7.3 minutes

3.2 Core i5 with SSE4.2 Instruction Set

SSE4.2 is a CPU instruction set that was released in 2006. To allow tensorflow to use it with the laptop personal CPU. compiling the source code was needed. To still have the regular environment that doesn't use the SSE4.2 side by side with the tensorflow that uses SSE4.2, I used a different virtual environment on my laptop to install tensorflow again from the source code. Virtual environment are very useful in such cases where we need to use different versions of a software (python 2.7 and python3 or example).

Compiling the source code of Tensorflow allowed the utilization of the SSE4.2 instruction set on a personal laptop CPU Core i5. Training performance is shown in Table 2.

Table 2. Training performance for Core i5 with SSE4.2

Epochs	Layers	Tensorflow	Time
25	4	CPU SSE4.2	2.6 minutes
100	4	CPU SSE4.2	14.2 minutes
400	4	CPU SSE4.2	110 minutes
25	7	CPU SSE4.2	3.5 minutes

3.3 GPU on AWS

AWS specialized hardware p2.xlarge with a specialized software Deep Learning AMIs allows Tensorflow to utilize a GPU on amazon cloud services. Training results are shown in Table 3.

Table 3. Training performance for GPU

Epochs	Layers	Tensorflow	Time
100	4	GPU	6.6 minutes
200	4	GPU	25.5 minutes
400	4	GPU	+2 hours

While the GPU performs better on small number of epochs. Clearly the GPU performance is not as good as expected for high number of epochs. Some code optimizations were needed to allow the GPU to perform better.

One of the optimizations is increasing the memory limit that Tensorflow can use. Another is increasing the number of batches that are fed to the network in each epoch (iteration) which probably allowed the utilization of more cores on the GPU. Rthe new results are shown in Table 3.

AWS is a cloud services by Amazon that allow users to access high performance computing units online. They provide software images that are installed and customized for the user needs. The experiment was done on an Ubuntu system for deep learning that has Tensorflow and NVidia GPU card called Tesla K80. To make the system organized, this system image has each deep learning framework installed in a different virtual environment. For example for the experiments done for this paper, the environment used was the tensorflow with python2.7.

AWS instances with more GPUs are also available for use. But they require different tensorflow code in order to utilize all of them.

Table 4. Training performance for GPU with optimized code

Epochs	Layers	Tensorflow	Time
100	4	GPU	2.1 minutes
200	4	GPU	6.6 minutes
400	4	GPU	24.4 minutes

4. Results Analysis

Looking at Tables 1 and 2 shows that using a CPU with SSE4.2 is roughly 30% faster than using a CPU without SSE4.2.

Tables 2 and 3 show that using a GPU is around 50% faster than using a CPU for small number of epochs (iterations). While for large number of epochs the GPU doesn't perform well at all.

Optimizing the Tensorflow code to use more memory and to use more data batches in a single iteration boosts the GPU performance to around 75% faster than a CPU.

The GPU many cores are probably much faster that what the memory can provide at a single moment. Based on that, the bottle neck for the GPU was the memory performance. After increasing the memory limit that tensorflow can access, the performance increased.

The special thing about GPUs is that they have multiple cores that work at the same time. To utilize all of them more data ned to be provided at a single moment. And that is why increasing data size which was described as providing more data batches, increased

the performance of training, resulting in less training time for the CCN neural network.

5. Summary

Using a GPU that has many more cores than a CPU does boost the performance of a deep neural network. Particularly during the training phase. The maximum boost that was observed according to the experiments done on a CNN network for image classification was around 75%.

Using the instruction set of the CPU does increase the performance very well. The numbers show that it gave a 30% increase in performance

6. REFERENCES

[1] Tensorflow Deep Learning framework by Google. https://www.tensorflow.org/get_started/

[2] Neural Network of CNN type for images classification. https://github.com/amerrez/conv-nn-zener-images

[3] Zener Cards description on Wikipedia https://en.wikipedia.org/wiki/Zener_cards

[4] Amazon web services: https://aws.amazon.com