Power Consumption of Deep Learning on CPUs vs GPUs

Karim Elmaaroufi, Vishnu Razdan

Carnegie Mellon University

Department of Electrical & Computer Engineering

Pittsburgh, PA, USA

*Abstract*— Deep learning has sparked a new found interest in machine learning due to its ability to solve several types of tasks. There are large computational complexities associated with deep learning due to the size of datasets and the connections between data points. The most recent libraries use a combination of CPU and GPU computation to accelerate the deep learning algorithms. Most notably, this occurs during the training period. Double digit performance increases have been noted since the inclusion of GPUs in deep learning. However, since their inclusion, GPUs have grown in power consumption to meet computational demands. This has led to issues in reliability and cooling in data center applications. In order to strike a balance between power and performance, this paper aims to answer the holistic question of, “Are GPUs more energy efficient for deep learning algorithms?” We will include the cost of data transfer over PCIe from host to GPU, computation on the GPU, the pre and post processing of data done by the CPU, and the total time required for the algorithm.

Keywords—deep learning; machine learning; energy efficiency; GPU power consumption

# Introduction

Deep learning is a subset of machine learning. It is based off of similar principals where algorithms search for patterns or tasks in deeply connected graphs.

The renewed interest in deep learning (also known as neural networks) is partially due to the new hardware acceleration available. Many of the algorithms in use today, are well-known and already in use (anomaly detection, k-means clustering, regression analysis, classification using neural networks, neural forests, decision trees, etc.) [5]. Traditionally, they would run on large clusters of computers which only had CPUs to train datasets. GPUs have accelerated many of these algorithms. In fact, in some applications such as, Stochastic Gradient Decent, there are performance gains of up to 40x [8]. Deep neural networks (DNN) work by learning about features from a labeled dataset. There are some algorithms which can operate on non-labeled datasets. In any case, DNNs are large graphs which have nodes that tend to be independent of one another. The weights between going from node to node represent “knowledge” in the DNN. In matrix form, you can image a row represents one neuron’s connections to all the neurons around it. An output for each row can be calculated independently of the other rows and the result is combined. GPUs were designed to run the same computation over vectors. You can imagine in the case of addition, a CPU would have to add 2 values and store them whereas a GPU can add two vectors of values and store them. It is precisely for this reason that GPUs have become the tool of choice for Machine Learning algorithms.

Due to multiple constrains, the maximum Thermal Design Point (TDP) of CPUs has capped at around 140W. For the given die size used today, it becomes difficult to dissipate the heat of anything more powerful. GPUs on the other hand, have continued to scale upwards in both transistor count and power consumption. For example, NVIDIA’s architecture from Fermi (2010) to Pascal (2016) has seen a 5x increase in transistor count but a 23% increase in max TDP [1]. That is, the increased computational power has come at the cost of increased power consumption.

The goal of this work will be to examine the relationship between the performance and power consumption trade off in using a GPU vs CPU for deep learning algorithms.

# Previous Work

Mittal and Vetter have done a survey of the possible methods of optimizing GPUs energy efficiency [2]. In their analysis, they identified several areas of concern. They surveyed them via the following: DVFS (Dynamic Voltage/Frequency Scaling), workload balancing between CPU and GPU, GPU architecture modifications, dynamic resource allocation, and programming level power management. From their work, we notice that GPUs need to be optimized for power at a variety of levels and that their power consumption is not sustainable due to factors such as reliability and economics which result from high power consumption.

Shi et al. benchmarked some of the most popular datasets, deep learning algorithms, and open source libraries in use today [4]. They used three different types of networks: Fully Connected Networks, Convolutional Neural Networks, and Recurrent Neural Networks. For each network two datasets were chosen and all were running on the software packages of Caffe, CNTK, TensorFlow and Torch. Finally, all versions were run on a Desktop and server CPU, and three NVIDIA graphics cards. We will use their results to guide our selection in using the worst and best performant datasets and libraries to build a range possible energy efficiency of a GPU.

Datta et al. looked at optimizing the performance of nearest-neighbor computations [3]. They performed their analysis on several architectures including an AMD CPU, Intel CPU, NVIDIA GPU, amongst others. In varying the workload size, core count, thread count, and data storage location they observed that, “For large-scale calculations, the actual performance impact will depend on the required frequency of GPU-host data transfers” [3]. They noted that in order for performance gains to be realized, high data transfer rates over PCIe are required. We will aim to more clearly define the point where GPU acceleration becomes costlier in terms of power consumption while maintaining a performance improvement. We will also include the power costs of the data transfer from host to GPU.

# Technical Discription

Deep learning operations tend to use 16-bit floating point operations (FP16) quite heavily [6]. With the newest GPU architecture by NVIDIA, FP16 performance has doubled the previous generation [7]. CPU makers such as Intel have been improving FP16 performance, through extensions such as AVX-512 that parallelize floating point operations [10]. By leveraging the massive parallelization offered by GPUs, deep learning on GPUs can achieve up to 6.7 times the performance and up to 4.4 times energy efficiency than a CPU [7].

Deep learning training data sets can be in the hundreds of gigabytes to terabytes in terms of storage requirements. In order for the data to be processed on the GPU, the data must transfer over the Peripheral Component Interconnect Express (PCIe) bus. Currently, PCIe can send around 16 GB/s; however, with the increasing size of data sets PCIe becomes the performance limiter [9]. New interconnect technologies enable up to 40 GB/s [9]. While CPUs can have hundreds of gigabytes of RAM as well, the latest maximum performance GPU by NVIDIA offers only 24 GB [10].

## Testing hardware

The CPU resources available are an i7-3630QM and i7-6600U. Potential additional resources are the CPUs in the two separate GPU clusters.

The GPU resources available are a GTX 660M, GTX 760, and dual Titan X in SLI. Power measurements will be performed on the dual Titan X as they support the NVIDIA management software suite.

## Profiling tools

For recording CPU power consumption, we will use Intel Power Gadget 3.0. A baseline power measurement will record the power consumed while the OS and CPU are idle. Subtracting the baseline power consumption from the power consumption recorded while learning will provide the power consumed by the CPU for the learning task.

For GPU power consumption, we will use the NVIDIA Management Library on the dual Titan X in SLI. Among other things, the NVIDIA Management Library records power consumed by the GPU itself [11].

To be determined is how to measure the power consumed transferring data over PCIe to the GPU and back. Also to be determined is how to measure the cost, both in power and performance, of the portion of the program that can only execute on the CPU.

## Profiled Parameters

We will be using two networks- ResNet-50 and FCN-8. Both networks will be run using Caffe framework to execute on the various GPUs.

We will be using two data sets- SUN397 and the COCO 2014 and 2015 images. These data sets were chosen so that they are unable to be stored completely in GPU RAM at once. By using two data sets and two networks, we will compare the performance and energy trade off with learning on CPU vs. GPU, as well as seeing if the energy vs. performance tradeoff is affected more by the data set or the network. If we are unable to measure the costs of transferring data over PCIe we will use the MNIST and FaceScrub databases.

# Milestone I

## Hardware Platform

Our initial hardware setup consists of a Lenovo laptop with an Intel i7 3630 QM and NVIDIA GT 660M mobile graphics card and the NVIDIA Jetson TK1. We choose these platforms because we own them and we both do not have experience with Machine Learning. Since we have both never used any of the popular frameworks, we wanted to have easy access to hardware to install various frameworks and to test various tutorials until we became comfortable creating our own tests to run. This was preferred over using the clusters offered because those had a limited time period of use.

## Machine Learning Frameworks

We chose to install the Caffe and TensorFlow frameworks

first. Vishnu worked on installing Caffe and Karim worked on installing TensorFlow.

Vishnu did not have success trying to install Caffe on Windows. Vishnu tried installing Caffe using Bash on Ubuntu on Windows; however, he was unable to train LeNet on MNIST due to compilation errors. Vishnu also attempted to install TensorFlow on Windows using Docker, but the Docker system did not interact correctly with the existing settings on Windows for other Virtual Machines. Vishnu looked for tools to measure the energy costs of transferring data over PCIe but has been unable to find any. He will have Caffe running on the Jetson by October 31st. He will also be installing Ubuntu on his laptop in order to install Caffe.

Karim also did not have success trying to install TensorFlow onto his machine. After learning that the installation was to be done through bash, Karim insisted on using Bash on Ubuntu on Windows, a beta feature of Windows 10. He was successful in installing all of the required dependencies but stopped short of installing TensorFlow after he learned that there were multiple issues people faced when trying to install TensorFlow on Intel based architectures. He is currently looking at using TensorFlow on

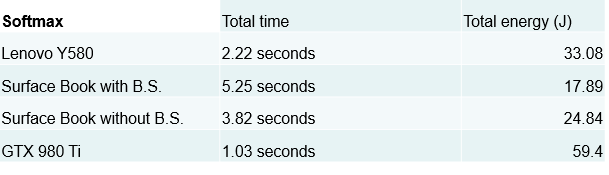


Figure 1: Timings and energy costs of running Softmax

the NVIDIA since it is better suited for ARM architectures and GPUs which are both present on the Jetson TK1.

Since we are still working on using the Caffe framework, we have added some minor goals to help us reach our next Milestone. We will be using the MNIST database as our initial target for a dataset to train. This database was chosen because it is well documented on by nearly every framework. We have located the three proto scripts required by Caffe to train the LeNet network on the database. Again, a new network was chosen since there are multiple tutorials about it. We hope to have initial timing and power results of this database on Caffe using the CPU only by 10/31/2016 (pending the successful installation of Caffe). We will then move to adapt the training algorithm to run on the two larger databases (if we can locate a power measure tool for the PCIe data transfer): SUN397 and COCO.

# Updated Milestones

By October 28th we will finalize all testing parameters and combinations to execute on the CPUs and GPUs.

By November 11th we will have code running on the GTX 760.

By November 28th we will have finished running and profiling all code on the various CPUs and GPUs for all testing combinations. At this point, no further code should be needed to run.

By December 16th all the data we collected will be analyzed and written into a final report.

All information is located at <https://vrazdan.github.io/743-site/>.

# Results

Our final hardware configuration used for testing including the following:

1. Surface Book – Intel i5-6300U CPU @ 2.40 GHz, 8GB RAM
2. Lenovo Y580 – Intel Core i7-3630 QM @ 2.40 GHz, 8 GB RAM
3. Desktop – NVIDIA GTX 980 Ti GPU, 6 GB DDR5 RAM, 2816 CUDA Cores

For these hardware devices, we ran two machine learning algorithms: LeNet 5 and Softmax. They both ran using Google’s TensorFlow library. Our data set was the MNIST image database, which includes 60,000 examples of handwritten digits for training.



Figure 2: Timings and energy costs of running LeNet 5

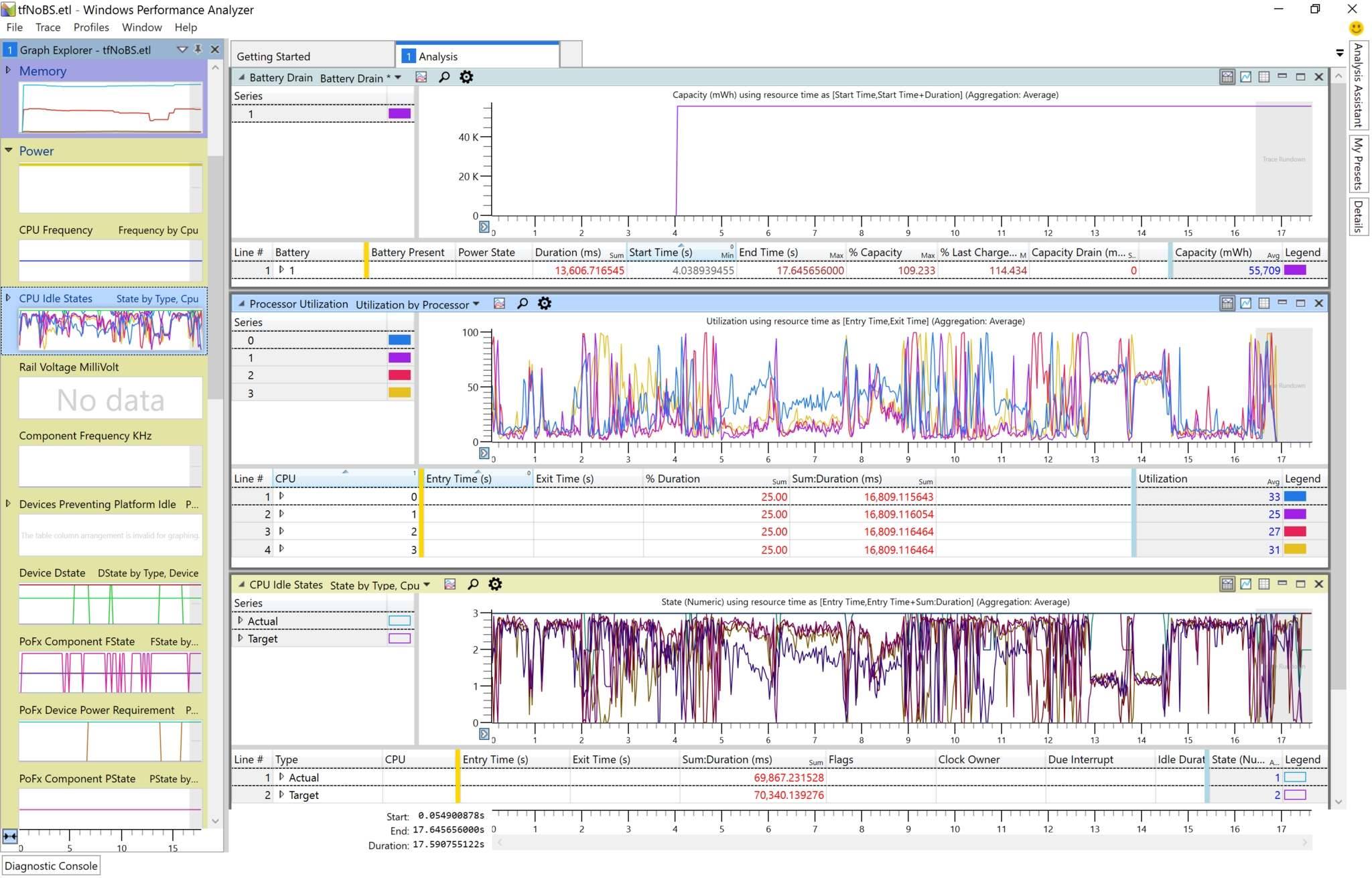


Figure 3: Windows Performance Analyzer during a Softmax training session

To establish the power consumption of CPU vs GPU implementations, we ran the training of each algorithm on each set of hardware using only CPUs for the laptops and the GPU for the desktop. For CPUs, we used the Intel Power Gadget 3.0 application to measure just the power consumed by the CPU. Furthermore, on the Surface Book we also ran tests when the laptop was running on Battery Saver mode and normal performance. To measure the GPUs power consumption, we used the NVIDIA System Management Interface. Finally, we also used the Windows Performance Recorder and Windows Performance Analyzer applications show in Figure 3 to obtain the current consumption from the two laptop’s batteries. The Windows tools were selected because we wanted to find the power consumption of memory transfer from main memory to the GPUs onboard memory. Kalidas et al. performed similar research and found that a comparison of overall system power consumption gave indication to memory transfer power costs when using CPUs vs GPUs [12]. However, an exact measurement of memory transfer alone is difficult to observe. Thus, instead of performing physical power measurements, we obtain them by viewing current draw from the laptops when running only on their batteries.

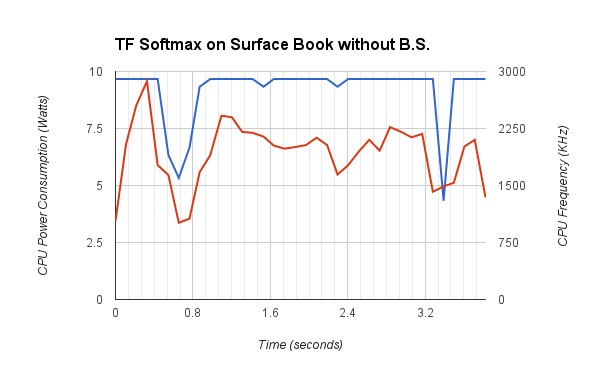
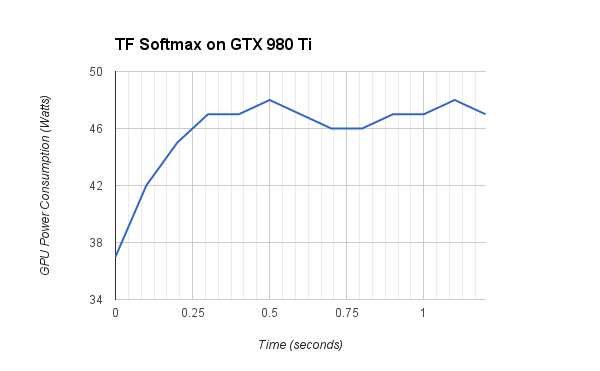
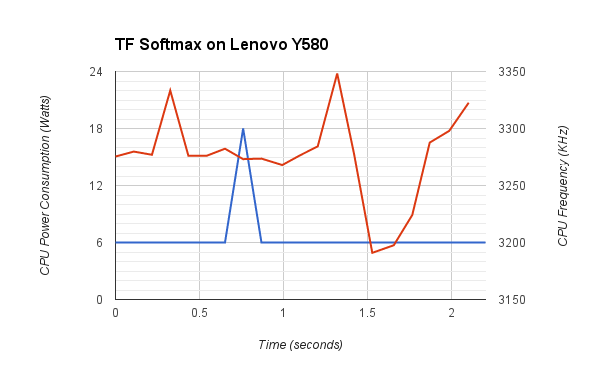
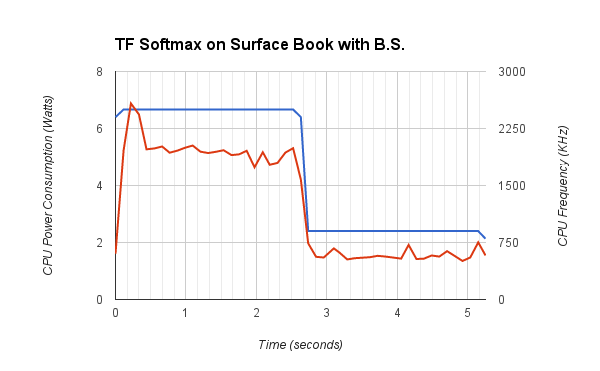
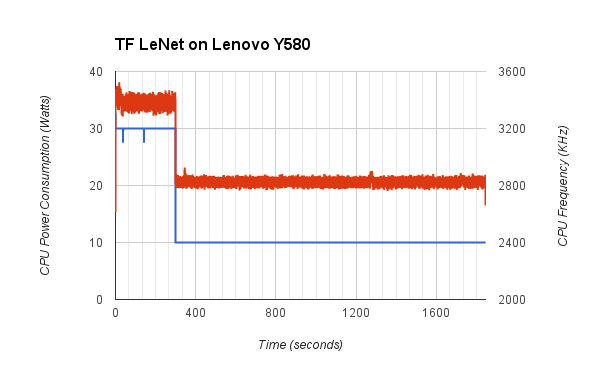
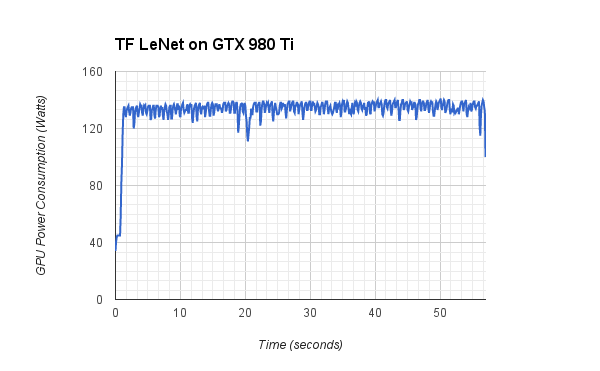
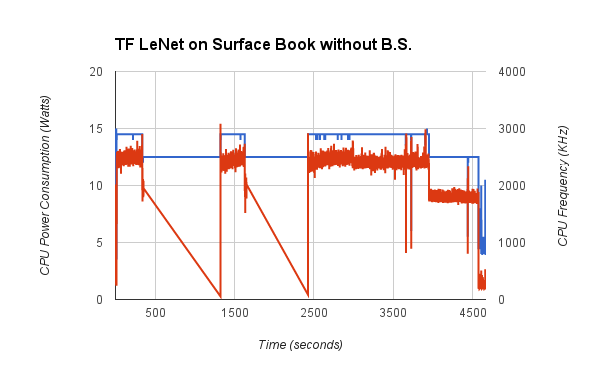
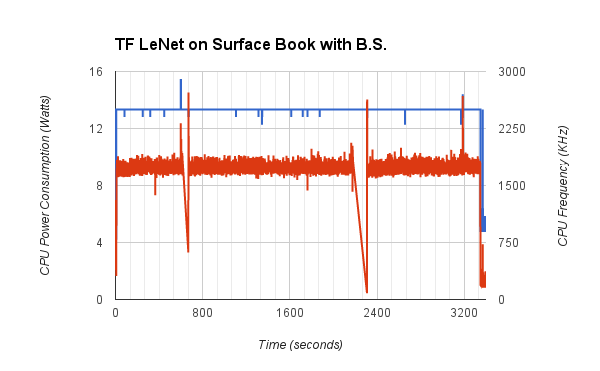
## Timing Results

LeNet and Softmax varied greatly in terms of computation time and power consumption. Regardless of the laptop we used, Figure 1 shows we reduced energy consumption up to 70% for only a 4 second time delay when running Softmax. We saw delays as small as 1 second, but energy savings were reduced to 44%. Thus, GPU computation of Softmax is unwarranted. Additionally, we posit that for simpler algorithms such as Softmax, GPU training is not needed unless you have tremendously large amounts of data to train on.

However, LeNet results were far different. Figure 2 shows that GPU training can offer dramatic increases in savings, both in power and time. Due to the GPUs massive parallelization, training completed in under 1 minute and as such, a speed increase of 83x. All of the CPU versions took between 15 minutes and over an hour to complete the same computation at a large energy impact, being up to 5.6x more expensive. Lenet 5, a substantially more complex algorithm than Softmax, was much more expensive, both computationally and energy wise, even though they operated on the same size data. However, the energy costs of training on the i7-3630 QM, even though took substantially less time than the i5-6300 U, were substantially more which prevents a conclusion that faster processing always is more energy efficient.

Figure 5: Instantaneous power vs. Frequency vs. Time for training LeNet 5

Figure 4: Instantaneous power vs. Frequency vs. Time for training Softmax



## CPU vs. GPU energy and frequency results

Figure 4 shows graphs the instantaneous power versus operating frequency vs time for Softmax running on the Lenovo, GTX 980 Ti, Surface Book with Battery Saver, and Surface Book without Battery Saver (clockwise starting from upper-left). Since all four training sessions operated on a small time-scale, the data is sensitive to transient start-up and completion effects which are shown by the sharp spikes at the beginning of three of the four graphs in energy costs. There is a clear difference between training with battery saver on versus off, as with battery saver the CPU is heavily throttled after just 2.5 seconds of processing. Additionally, both the Lenovo and Surface Book without battery saver show drastic changes in frequency over time, contrasted with when battery saver was on.

Figure 5 shows graphs the instantaneous power versus operating frequency vs time for LeNet 5 running on the Lenovo, GTX 980 Ti, Surface Book with Battery Saver, and Surface Book without Battery Saver (clockwise starting from upper-left). Since training LeNet took substantially longer, the minor effects from starting and ending the training are no longer as prominent; therefor, more accurate conclusions may be made from the data. The two gaps in the graph for the training on the Surface Book without battery saver were because the laptop itself went into sleep mode, so no processing was done then. When CPUs are not limited by battery saving modes, eventually they become thermally limited and decrease their frequency. The Lenovo Y580 laptop limited the CPU after ~300 seconds while the Surface Book throttled the CPU after ~4000 seconds. The GTX 980 Ti did not suffer any such throttling, as the energy consumption was consistent for the minute it ran. Since GPUs are typically less thermally constrained than CPUs, at least in this environment, for longer running tasks GPUs seem to be the more efficient choice because they do not become throttled like the CPUs do.

## Conclusions

Thus, from our results, we see that there is no clear answer when selected between CPUs and GPUs for Machine Learning

algorithms. The optimal choice in terms of time savings and energy consumption depends on the algorithm used (which we demonstrated) and the size of the dataset used in training (which we did not demonstrate as we only used one dataset). While the time savings are not worth the increased energy costs for GPU training for simple algorithms on small data sets, for more complex algorithms and data sets there is no set rule on when to train on GPUs vs. CPUs.

# Future Work

Our results are far from extensive. Some of the work we would have liked to complete including testing the same networks but using a different library such as Caffe (one of the original libraries we wanted to test).

Larger datasets such as the Common Objects in Context (COCO maintained by Microsoft), have training sets of 80,000 images which consume over 13 GB of memory. Such a dataset would require at least 3 transfers between main memory and our GPU depending on the algorithm used. This would obviously increase memory’s power consumption.

We would have liked to identify for an algorithm such as LeNet, at what point does it become more energy efficient to perform the training on a CPU and what are the time differences in the two implementations? Additionally, when does thermal limitations factor in? To answer these questions, we would have needed to run training on larger datasets than MNIST. The predicted size of the dataset for our hardware is unknown.

##### References

1. Khalid Moammer, Nvidia : Pascal Is 10x Faster Than Maxwell, Launching in 2016 On 16nm – Features 3D Memory, NV-Link and Mixed Precision. http://wccftech.com/nvidia-pascal-gpu-gtc-2015/
2. Sparsh Mittal, Jeffrey S. Vetter, A Survey of Methods For Analyzing and Improving GPU Energy Efficiency <https://arxiv.org/pdf/1404.4629.pdf>
3. Kaushik Datta et al., Stencil Computation Optimization and Auto-tuning on State-of-the-Art Multicore Architectures <http://mc.stanford.edu/cgi-bin/images/e/ec/SC08_stencil_autotuning.pdf>
4. Shaohuai Shi, Qiang Wang, Pengfei Xu, Xiaowen Chu, Benchmarking State-of-the-Art Deep Learning Software Tools, <https://arxiv.org/pdf/1608.07249.pdf>
5. Brandon Rohrer, Machine learning algorithm cheat sheet for Microsoft Azure Machine Learning Studio, https://azure.microsoft.com/en-us/documentation/articles/machine-learning-algorithm-cheat-sheet/
6. Ryan Smith, The NVIDIA GeForce GTX 1080 & GTX 1070 Founders Editions Review: Kicking Off the FinFET Generation, http://www.anandtech.com/show/10325/the-nvidia-geforce-gtx-1080-and-1070-founders-edition-review/5
7. Michael Andersch, Inference: The Next Step in GPU-Accelerated Deep Learning, <https://devblogs.nvidia.com/parallelforall/inference-next-step-gpu-accelerated-deep-learning/>
8. Brad Nemire, CUDA Spotlight: GPU-Accelerated Deep Neural Networks, <https://devblogs.nvidia.com/parallelforall/cuda-spotlight-gpu-accelerated-deep-neural-networks/>
9. Sumit Gupta, IBM POWER8 CPU and NVIDIA Pascal GPU speed ahead with NVLINK, https://www.ibm.com/blogs/systems/ibm-power8-cpu-and-nvidia-pascal-gpu-speed-ahead-with-nvlink/
10. James Reinders, Intel AVX-512 instructions, https://software.intel.com/en-us/blogs/2013/avx-512-instructions.
11. NVIDIA, NVIDIA Management Library, https://developer.nvidia.com/nvidia-management-library-nvml
12. Rubasri Kalidas, Mayank Daga, Konstantinos Krommydas, and Wu-chun Feng, On the Performance, Energy, and Power of Data-Access Methods in Heterogeneous Computing Systems , 11th workshop on High-Performance, Power-Aware Computing (HPPAC), Hyderabad, India http://synergy.cs.vt.edu/pubs/papers/rubasri-data-access-hppac15.pdf