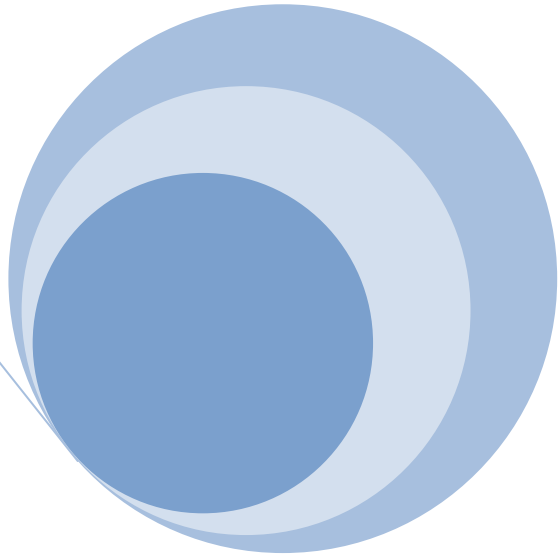

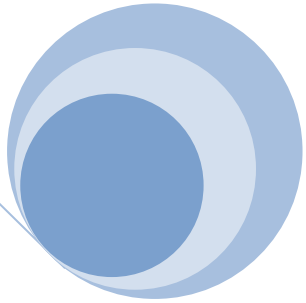

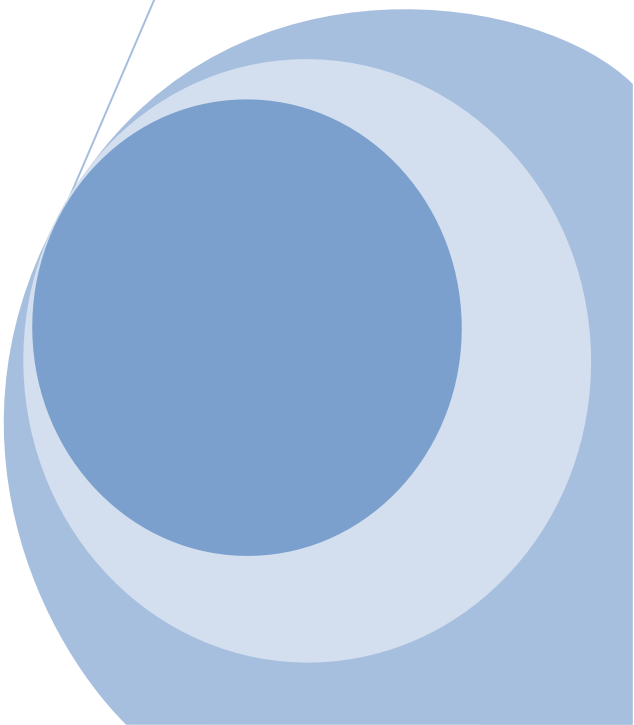

CS 5850 Fall 2015

Project Write-up

The project was initiated with an overview of several current high performance computing (HPC) technologies. Followed by developing a basic understanding of neural networks, and finished with the implementation of a non-linear neural network that exceeded the accuracy of a similar non-linear logistic regression model.

bks
12/17/2015

Table of Contents

Beginning: HPC Technologies

The student became familiar with current popular HPC technology CUDA (NVIDIA) by reading and working several programs in the text CUDA By Example [1], and made use of chapter 2 of Professional CUDA C Programming [2] for clarification of some examples in [1]. All but a couple chapters in [1] were reviewed. The text Programming Massively Parallel Processors: A Hands-on Approach was used to learn about the popular and powerful c++ styled "library of parallel algorithms and data structures" [4].

NVIDIA's cuDNN, a "CUDA Deep Neural Network library (cuDNN) is a GPU-accelerated library of primitives for deep neural networks." [5] NVIDIA's cuDNN was investigated, but not studied or worked with directly, because its use is at a lower level in the development of neural networks that this projects scope. It is however, widely supported by the major deep learning frameworks such as Caffe, Theano, and Torch [4].

While learning about the above mentioned technologies, the student was able to gain familiarity with Nsight [6]. Nsight is NVIDIA's Eclipse based integrated development environment, and it is of tremendous benefit in developing and debugging CUDA / GPU based applications. The student also purchased two GPU [7] cards to facilitate running GPU applications and especially debugging with Nsight under Linux.

Due to the current and dynamic nature of the technologies involved the project made use of the following operating systems: Debian 8, openSuse 13.2, and Ubuntu / Lubuntu 14.04. The reason for the switches was due to the increasing and more restrictive requirements of the newest software libraries, as the project progressed.

Middle: Neural Networks

Next the student gained basic knowledge about neural networks reading through the first six of Neural Network Design [8]. Neural Network Design is a well written text intended for engineering students learning about neural networks. It is software agnostic for examples and exercises; its supplementary software is written in Matlab; and two of the authors are co-authors of the Neural Network Toolbox for MATLAB. The book provides a nice overview of neural networks and their benefits, it doesn't focus on deep learning networks, so it has been retained by the student, but was not used further in the project.

End: Deep Learning Frameworks

Theano

The student initially started with Theano, as a deep learning framework in which to work. Theano, developed at the University of Montreal, "is a Python library that allows you to define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays efficiently."[9] Theano, with its tight integration with NumPy and its transparent use of GPU were appealing, but the student was still using openSuse 13.2 at the time, an unsupported version of Linux, and installation issues became prohibitive. Theano's documentation is impressive and appears unparalleled by other deep learning frameworks. In hindsight, the student wishes he had already moved onto Ubuntu/Lubuntu 14.01, a supported Theano operating system, but hopes to make use of Theano / Python in the future.

Caffe

Caffe [10], is a deep learning framework develop by the Berkley Vision Learning Center (BVLC). Caffe is focused on speed and ease of use for image processing. Caffe has a wide variety of, mainly image processing, models and examples available for users. Installation of Caffe was much easier than the

attempt to install Theano, although the student was using Lubuntu 14.04 for Caffe installation, which was also a recommended operating system by Caffe [12].

It is the student's impression that although Caffe is considered flexible by avid Caffe users, and in some regards it is by using plain text files to configure networks, Caffe can be difficult to use for building non-image related networks. The prior sentence needs to be taken in the context of a novice user. Perhaps a more experienced user could repurpose Caffe for say linear regression with out much difficulty. Although, Caffe indicates in [11] that it is probably not the best tool in which to perform linear regression.

Although Caffe's installation was smooth, it seems to lack, at least in comparison to Theano, organized and well written documentation [12]. Much of the official documentation feels written with the assumption that the user already understands the topic, and therefore, comes across as a bit sparsely explained, which is unfortunate for new users. Caffe seems to have a vibrant GitHub [13] community, but cobbling together an understanding of using Caffe, via GitHub posts, for non-image purposes can be challenging. Part of this challenge was due to the student's lack of Python understanding, as most Caffe users prefer to make use of pycaffe, Caffe's Python interface.

The student finally came across the paper, Designing Deep Learning Neural Networks using Caffe [14], which although, also on the topic of image processing, was quite helpful in gaining an understanding of the process of developing and training networks with Caffe. The LeNet tutorial [15] was also more detailed in its explanation than most other tutorials reviewed on the Caffe website.

HDF5

On problem with working with non-image data for Caffe was that it would be best to use the HDF5 [16] data layer type. The following is from The HDF5 Group's website:

> HDF5 is a data model, library, and file format for storing and managing data. It supports an unlimited variety of datatypes, and is designed for flexible and efficient I/O and for high volume and complex data. HDF5 is portable and is extensible, allowing applications to evolve in their use of HDF5. The HDF5 Technology suite includes tools and applications for managing, manipulating, viewing, and analyzing data in the HDF5 format.

The student struggled with getting the project's data set into the proper format to use with HDF5, until the student found the Java based HDF5 viewer on the HDF5 website. The use of the viewer likely saved many hours of internet searches related to proper data format to use with Caffe's 4D array data type.

Forest Fire Data Set

The project made use of the forest fire data set provided by the University of California at Irving Machine Learning Repository [17]. This data set is based on forest fires in Montesinho National Park, Portugal. The data set represents information collected on forest fires from January 2000 through December 2003.

The source of the forest fire data set is Paulo Cortez and Anibal Morais at the University of Minho, Portugal. In their paper [18] they examined the following data mining (DM) approaches to analyze the data: Multiple Regression, Decision Trees, Random Forests, Neural Networks, and Support Vector Machines. They used four variables in their predictive modeling:

- Temp: Outside temperature (in ◦ C)
- RH: Outside relative humidity (in %)

- Wind: Outside wind speed (in km/h)
- Rain: Outside rain (in mm/m 2)

In experiments with each approach, conducted using the RMiner software package for r, they found SVM to be the most accurate in predicting small forest fires. Small fire prediction is important, because the vast majority of the fires in the park are small fires, often the result of human activity.

Unfortunately, we were not able to reproduce the author's work using Caffe, but with the help of an example, discussed in the next section, we were able to implement a logistic regression model with Caffe to predict the probability of the presence of a small fire, given the data. Classification is an activity that Caffe excels at, given the nature of much image analysis revolves around the questions such as the probability that an image belongs in a particular category.

Caffe: Logistic Regression, Non-linear Neural Network, scikit-learn.org

The student made use of Caffe's Juypter Notebook example called "Brewing Logistic Regression then Going Deeper," [19] as a guide for performing logistic regression on the forest fire data set. The combination of this example [19], example [14], and the LeNet example [20] from the Caffe website, were very helpful in piecing together the many aspects of implementing a neural network model with Caffe.

Following the Juypter notebook example, its code was modified to first run a logistic regression using scikit-learn.org [21]. The result of running scikit-learn.org and a neural network designed for logistic regression was as follows:

> ... scikit-learn logistic regression accuracy = Accuracy: 0.389
> Accuracy: 0.367

The definition for this network is found in Appendix A, and the solver for this network is found in Appendix B. The solver can be thought of as the parameters Caffe uses to administer and monitor the network during training and testing.

With [19] as a guide, the code for a non-linear neural network model was implemented in Python and trained and tested by Caffe. A comparison of the predictive accuracy of scikit-learn.org's logistic regression and the new non-linear model are as follows:

> ... scikit-learn logistic regression accuracy = Accuracy: 0.389
> Accuracy: 0.633

There is a noticeable improvement in the non-linear model.

Future Work

A lot was learned this semester, but a lot of time was consumed with changing operating systems and basically getting everything to work together, and not enough time learning about neural networks and deep learning. Caffe is a good tool if you want to do image analysis, as there are many examples and tutorials exactly for such work, but doing other types of analysis with it was a real challenge. Again, perhaps it was mostly due to lack of experience.

Neural networks and especially deep learning networks are a fast moving and rather sophisticated topic upon which to become acquainted with in such a short period is a challenge. Although, given the challenges of this semester, we welcome the opportunity to continue working with and learn about neural

networks and deep learning. We also feel that some of the challenges to getting everything to work properly may have occurred regardless of the software or deep learning frame work chosen. Some challenges are simply the learning curve for the topic.

Theano / Python

We feel that Theano and Python make a good choice for future, if only for Theano's extensive documentation. The documentation appears to go beyond just telling one how to implement the network with Theano, but attempts to explain what a network does also.

See http://deeplearning.net/tutorial/DBN.html for an excellent explanation, with illustrations, on the more sophisticated topic of Deep Belief Networks. The quality of the documentation is encouraging.

Keras: Deep Learning library for Theano and TensorFlow

Keras was recently brought to our attention, so we haven't had a chance to look at it yet. The website [22] indicates the following:

> Keras is a minimalist, highly modular neural networks library, written in Python and capable of running either on top of either TensorFlow or Theano. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result with the least possible delay is key to doing good research.

It appears to be a tool worth considering for future use.

Deep Learning Texts

There is a limited, but growing number of sources available on neural networks and deep learning, but three such sources are:

- Neural Networks and Deep Learning [23] (online)
- Artificial Intelligence for Humans, Volume 3: Deep Learning and Neural Networks [24]
- Deep Learning [25]

The first two texts appear to be an introduction to the subject, and we would like to use if this project can be extended. The last one is a bit more sophisticated and still a work in progress. Chapter 16 touches on topics from Topology, and Topological data analysis [25], a very active area of research between Topology and Machine Learning / Data Reduction. The text it is available for reading online.

Kaggle

We just wish to express that it could be useful and fun to potentially put some future work to the test by entering in a Kaggle.com contest [26].

Conclusion

We wish to thank Dr. Green for agreeing to advise this project and hope that it may be extended for another semester.

References

1. Sanders, Jason, and Edward Kandrot. CUDA by Example: An Introduction to General-purpose GPU Programming. Upper Saddle River, NJ: Addison-Wesley, 2011. Print., ISBN-10: 0131387685
2. Cheng, John, Max Grossman, and Ty McKercher. Professional CUDA C Programming. Indianapolis, IN: Wrox, 2014. Print., ISBN-10: 1118739329
3. Kirk, David, and Wen-mei Hwu. Programming Massively Parallel Processors: A Hands-on Approach. Burlington, MA: Morgan Kaufmann, 2010. Print., ISBN-10: 0124159923
4. https://developer.nvidia.com/cudnn
5. https://developer.nvidia.com/thrust
6. http://www.nvidia.com/object/nsight.html
7. ZOTAC GeForce GT 730 DirectX 12 (feature level 11_0) ZT-71114-20L 1GB 64-Bit DDR3 PCI Express 2.0 x16 (x8 lanes) Video Card
8. Hagan, Martin T., Howard B. Demuth, and Mark H. Beale. Neural Network Design. 2014
9. http://deeplearning.net/software/theano/
10. http://caffe.berkeleyvision.org/
11. Lines 30-35 of https://github.com/BVLC/caffe/blob/196b995efef4b644962e233ea421c4d586bb93ae/include/caffe/layers/euclidean_loss_layer.hpp
12. http://caffe.berkeleyvision.org/installation.html
13. https://github.com/BVLC/caffe
14. https://zenodo.org/record/31086/files/MUMITICT15.pdf
15. http://caffe.berkeleyvision.org/gathered/examples/mnist.html
16. https://www.hdfgroup.org/HDF5/
17. Lichman, M. (2013). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science, https://archive.ics.uci.edu/ml/datasets/Forest+Fires
18. A Data Mining Approach to Predict Forest Fires using Meteorological Data, http://www3.dsi.uminho.pt/pcortez/Home.html
19. http://nbviewer.ipython.org/github/BVLC/caffe/blob/master/examples/02-brewing-logreg.ipynb
20. http://caffe.berkeleyvision.org/gathered/examples/mnist.html
21. http://scikit-learn.org/stable/index.html
22. http://keras.io/
23. http://neuralnetworksanddeeplearning.com/index.html
24. http://www.heatonresearch.com/book/
25. http://goodfeli.github.io/dlbook/
26. https://en.wikipedia.org/wiki/Topological_data_analysis
27. https://www.kaggle.com/

Appendix A: Logistic Regression Network Definition

logreg_auto_train.prototxt

```
layer {
  name: "data"
  type: "HDF5Data"
  top: "data"
  top: "label"
  hdf5_data_param {
    source:
"/home/brians/bks001_shared/Coding_projects/code/caffe/models/cs5850_caffe/2015_fall/Forest_Fire_D
ata_Set/data/train/train.txt"
    batch_size: 10
  }
}

layer {
  name: "ip1"
  type: "InnerProduct"
  bottom: "data"
  top: "ip1"
  inner_product_param {
    num_output: 2
    weight_filler {
      type: "xavier"
    }
  }
}

layer {
  name: "accuracy"
  type: "Accuracy"
  bottom: "ip1"
  bottom: "label"
  top: "accuracy"
}

layer {
  name: "loss"
  type: "SoftmaxWithLoss"
  bottom: "ip1"
  bottom: "label"
  top: "loss"
}
```

Appendix B: Logistic Regression Network Solver

solver.prototxt

```
# use full paths for now
train_net:
"/home/brians/bks001_shared/Coding_projects/code/caffe/models/cs5850_caffe/2015_fall/Forest_Fire_Data_Set/net_defs_solver_etc/logreg_auto_train.prototxt"

test_net:
"/home/brians/bks001_shared/Coding_projects/code/caffe/models/cs5850_caffe/2015_fall/Forest_Fire_Data_Set/net_defs_solver_etc/logreg_auto_test.prototxt"

test_iter: 250
test_interval: 1000
base_lr: 0.01
lr_policy: "step"
gamma: 0.1
stepsize: 5000
display: 1000
max_iter: 10000
momentum: 0.9
weight_decay: 0.0005
snapshot: 10000
snapshot_prefix:
"/home/brians/bks001_shared/Coding_projects/code/caffe/models/cs5850_caffe/2015_fall/Forest_Fire_Data_Set/data/train/logreg"
# solver_mode: CPU
solver_mode: GPU
```

Appendix C: Non-linear Network Solver

solver_nonlinear_net.prototxt

```
# use full paths for now

train_net:
"/home/brians/bks001_shared/Coding_projects/code/caffe/models/cs5850_caffe/2015_fall/Forest_Fire_Data_Set/net_defs_solver_etc/logreg_auto_train_nonlinear_net.prototxt"

test_net:
"/home/brians/bks001_shared/Coding_projects/code/caffe/models/cs5850_caffe/2015_fall/Forest_Fire_Data_Set/net_defs_solver_etc/logreg_auto_test_nonlinear_net.prototxt"

test_iter: 250
test_interval: 1000
base_lr: 0.01
lr_policy: "step"
gamma: 0.1
stepsize: 5000
display: 1000
max_iter: 10000
momentum: 0.9
weight_decay: 0.0005
snapshot: 10000
snapshot_prefix:
"/home/brians/bks001_shared/Coding_projects/code/caffe/models/cs5850_caffe/2015_fall/Forest_Fire_Data_Set/data/train/nonlinear_net"
# solver_mode: CPU
solver_mode: GPU
```

Appendix D: Non-linear Network Definition

logreg_auto_train_nonlinear_net.prototxt

```
layer {
  name: "data"
  type: "HDF5Data"
  top: "data"
  top: "label"
  hdf5_data_param {
    source:
"/home/brians/bks001_shared/Coding_projects/code/caffe/models/cs5850_caffe/2015_fall/Forest_Fire_D
ata_Set/data/train/train.txt"
    batch_size: 10
  }
}

layer {
  name: "ip1"
  type: "InnerProduct"
  bottom: "data"
  top: "ip1"
  inner_product_param {
    num_output: 40
    weight_filler {
      type: "xavier"
    }
  }
}

layer {
  name: "relu1"
  type: "ReLU"
  bottom: "ip1"
  top: "ip1"
}

layer {
  name: "ip2"
  type: "InnerProduct"
  bottom: "ip1"
  top: "ip2"
  inner_product_param {
    num_output: 2
    weight_filler {
      type: "xavier"
    }
  }
```

```
}

layer {
  name: "accuracy"
  type: "Accuracy"
  bottom: "ip2"
  bottom: "label"
  top: "accuracy"
}

layer {
  name: "loss"
  type: "SoftmaxWithLoss"
  bottom: "ip2"
  bottom: "label"
  top: "loss"
}
```