DIGITS: the Deep learning GPU Training System

Luke Yeager LYEAGER@NVIDIA.COM
Julie Bernauer JBERNAUER@NVIDIA.COM
Allison Gray AGRAY@NVIDIA.COM
Michael Houston MHOUSTON@NVIDIA.COM
NVIDIA Corporation, 2701 San Tomas Expressway, Santa Clara, CA 95050, USA

Abstract

DIGITS is a deep learning training system with a web interface. Tools are provided for designing custom network architectures and for rapidly evaluating their effectiveness through various visualizations of training outputs and learned network parameters allowing for rapid prototyping and collaboration.

Keywords: Deep Learning, Convolutional Neural Networks, User Interface, GPU, GPGPU

1. Introduction

Over the last few years, deep convolutional neural networks (CNNs) have become more and more popular in fields ranging from image classification (Krizhevsky et al. (2012)) to object detection (Szegedy et al. (2013)), image segmentation (Long et al. (2014)), natural language processing (Chen and Manning (2014)), speech recognition (Hannun et al. (2014)) and many more. Several open-source frameworks exist for defining and training CNNs. Caffe (Jia et al. (2014)), Torch (Collobert et al. (2011)) and Theano (Bergstra et al. (2010)) are among the most popular. They focus on supporting the newest and fastest computational methods and are mainly targeted to experts comfortable with command line tools. Several projects (Bruckner et al. (2013), Torralba (2015), Tomkins (2014)) have provided visualization tools for these frameworks, but none of these provide a full training solution. Other applications, like MetaMind (Socher and Strohband (2014)), provide deep learning solutions without exposing the underlying network architecture. The specifics of training are entirely hidden from the user who relies on the software to design the network.

DIGITS offers a different approach. It provides a web interface for deep learning, avoiding direct interaction with the underlying frameworks, while still exposing the details of optimization and network architecture to those needing to design their own model. System resources are distributed to multiple concurrent training jobs (which enables rapid prototyping), and the status of each job can be shared with collaborators through the web interface.

DIGITS is an open-source project available under the BSD license at http://github.com/NVIDIA/DIGITS.

2. Methods and Usage

DIGITS makes it easy to create a dataset of training and validation images, train a model on the dataset, and test the model in various ways. A RESTful web application, created with the Flask python web framework, allows users to create and delete both datasets and models through a web page (see Figure 1).

To create a model, several standard networks are available: LeNet-5 (Lecun et al. (1998)), AlexNet (Krizhevsky et al. (2012)) and GoogLeNet (Szegedy et al. (2014)). While none of these networks is likely the best solution for the specific problem at hand, they provide a good starting point for creating a custom network. Starting from a template, users can use a visualization tool to view the network graph as they directly modify the text defining the network structure. Many options for data transformation (mean subtraction, random cropping, etc.) and gradient descent optimization (batch size, learning rate, etc.) are provided, pre-loaded with reasonable defaults.

During training, which can be aborted and restarted at any time, the validation metrics of accuracy and loss can be seen graphically through javascript charts which update in real-time. The model can also be tested at various intermediate points during training by providing an image or list of images to classify. Classification predictions as well as statistics and visual representations of weights and activations of the network layers are displayed, providing useful debugging information. After training, the necessary files to deploy the model elsewhere can be downloaded as an archive.

3. Conclusion and Perspectives

DIGITS provides a simple solution to design neural networks without the need to learn the specifics of deep learning frameworks. It enables rapid prototyping of models and also collaboration and visibility to others through the web.

The current version only supports Caffe as a backend framework and models trained for image classification. Future work includes adding support for other neural network frameworks as backends, different training scenarios like general regression training, and more visualization tools like in Bruckner et al. (2013). Eventually, DIGITS will become the frontend job manager for a cluster of machines, allowing even more parallelization of training jobs and opening the door to automated hyperparameter optimization.

References

J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010. Oral Presentation.

D. Bruckner, J. Rosen, and E. R. Sparks. deepviz: Visualizing convolutional neural networks for image classification, 2013. URL http://vis.berkeley.edu/courses/cs294-10-fa13/wiki/images/f/fd/DeepVizPaper.pdf.

D. Chen and C. D. Manning. A fast and accurate dependency parser using neural networks. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 740–750, 2014.

R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, 2011.

A. Y. Hannun, C. Case, J. Casper, B. C. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, and A. Y. Ng. Deep speech: Scaling up end-to-end speech recognition. *CoRR*, abs/1412.5567, 2014. URL http://arxiv.org/abs/1412.5567.

Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. URL http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998. ISSN 0018-9219. doi: 10.1109/5.726791.

J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. *CoRR*, abs/1411.4038, 2014. URL http://arxiv.org/abs/1411.4038.

R. Socher and S. Strohband. Metamind, 2014. URL https://www.metamind.io/.

C. Szegedy, A. Toshev, and D. Erhan. Deep neural networks for object detection. In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2553–2561. Curran Associates, Inc., 2013. URL http://papers.nips.cc/paper/5207-deep-neural-networks-for-object-detection.pdf.

C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014. URL http://arxiv.org/abs/1409.4842.

H. Tomkins. Caffe gui tool, 2014. URL https://github.com/Chasvortex/caffe-gui-tool.

A. Torralba. Drawnet, 2015. URL http://people.csail.mit.edu/torralba/research/drawCNN/drawNet.html.

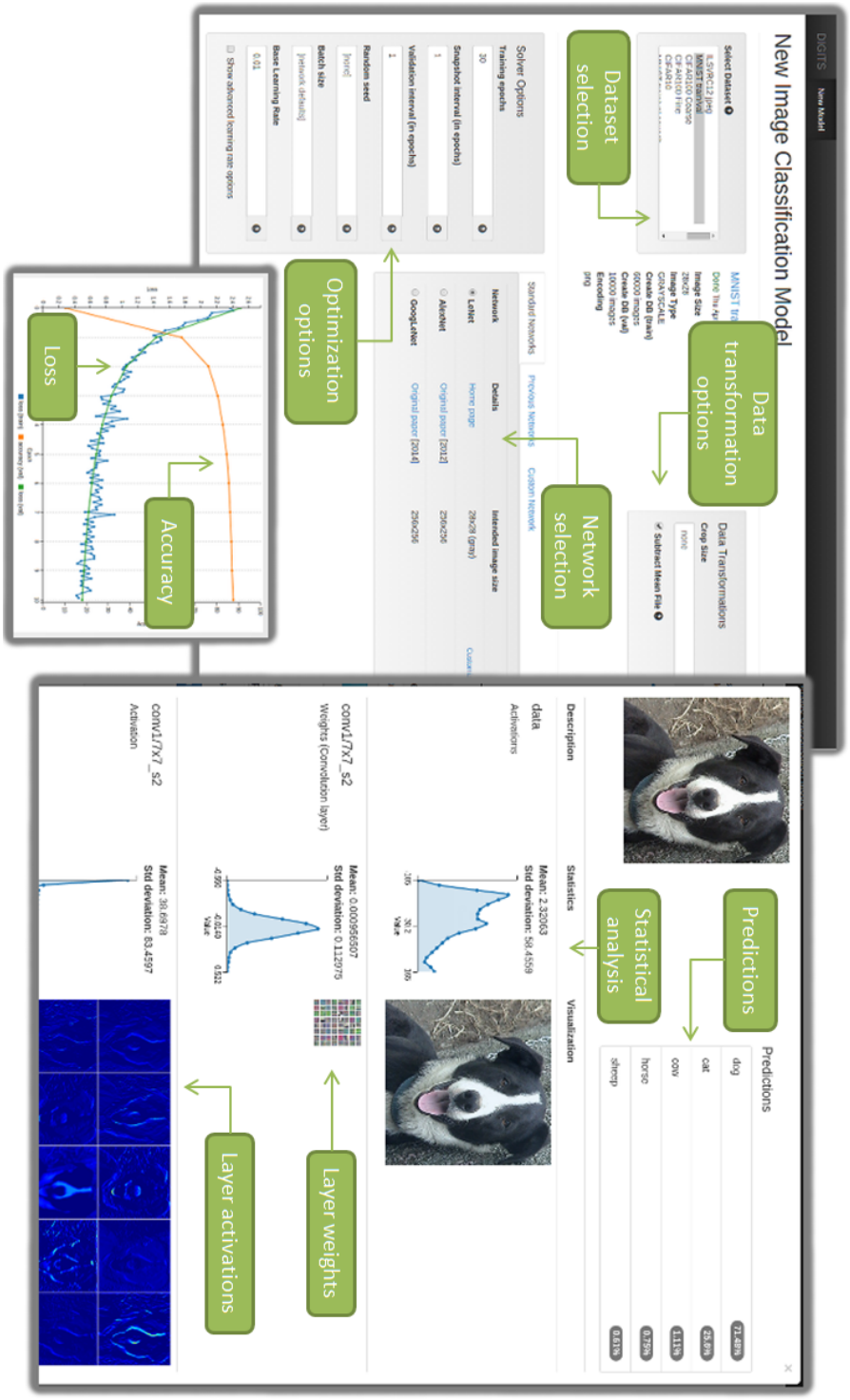

Figure 1: Creating and testing a model through the DIGITS interface