Journal of Machine Learning Research 1 (2015) ?-?? Submitted 5/15; Published ??/??

Blocks and Fuel: GPU-accelerated neural networks for large data

Bart van Merriënboer BART.VAN.VANMERRIENBOER@UMONTREAL.CA
Montreal Institute for Learning Algorithms, University of Montreal, Montreal, Canada

Dzmitry Bahdanau D.BAHDANAU@JACOBS-UNIVERSITY.DE
Jacobs University, Bremen, Germany

Jan Chorowski JAN.CHOROWSKI@II.UNI.WROC.PL
University of Wrocław, Wrocław, Poland

Vincent Dumoulin DUMOULIV@IRO.UMONTREAL.CA
Dmitriy Serdyuk SERDYUK.DMITRIY@GMAIL.COM
David Wade-Farley DAVID.WADE-FARLEY@UMONTREAL.CA
Yoshua Bengio YOSHUA.BENGIO@UMONTREAL.CA
Montreal Institute for Learning Algorithms, University of Montreal, Montreal, Canada

Editor: ?

Abstract

We introduce two Python frameworks to train neural networks on large datasets: *Blocks* and *Fuel*. *Blocks* is based on the Theano, a linear algebra compiler with CUDA-support (Bastien et al., 2012; Bergstra et al., 2010). It facilitates the training of complex neural network models by providing parametrized Theano operations, attaching metadata to Theano's symbolic computation graph, and providing an extensive set of utilities to assist training the networks, e.g. training algorithms, logging, monitoring, visualization, and serialization. *Fuel* provides a standard format for machine learning datasets. It allows the user to easily iterate over large datasets, performing many types of pre-processing on the fly.

Keywords: Neural networks, GPGPU, large-scale machine learning

1. Introduction

Blocks is a framework that builds on Theano to facilitate the training of neural networks. It is being developed by the Montreal Institute of Learning Algorithms (MILA) at the University of Montreal. Its focus lies on quick prototyping of complex neural network models. *Fuel*

2. History

- Pylearn2: More general (machine learning vs. neural networks), abstracts away Theano
- Lasagne, Keras: Stronger focus on easy implementation of existing models than the development of new models
- scikit-learn: Less plug-and-play, more research-oriented

3. Blocks

Theano is a popular choice for the implementation of neural networks (see e.g. Goodfellow et al. (2013b); Pascanu et al. (2013)). Blocks and many other libraries, such as Pylearn2 Goodfellow et al. (2013a), build on Theano by providing reusable components that are common in neural networks, such as linear transformations followed by non-linear activations, or more complicated components such as LSTM units. In Blocks these components are referred ot as *bricks* or "parametrized Theano operations".

Bricks take the form of a set of parameters in the form of Theano shared variables, for example the weight matrix of a linear transformation or the filters of a convolutional layer. Bricks use these parameters to transform symbolic Theano variables, potentially in multiple ways.

Bricks can contain other bricks within them. This effectively introduces a of hierarchical structure on top of the flat computation graph defined by Theano, which makes it easier to address and configure complex models programmatically.

The parameters of bricks can be initialized using a variety of schemes that are popular in the neural network literature, such as sparse initialization, orthogonal initialization for recurrent weights, etc.

Large neural networks can often result in Theano computation graphs containing hundreds of variables and operations. Blocks does not attempt to abstract away this complex graph, but to make it manageable by annotating variables in the graph. Each input, output, and parameter of a brick is annotated as such. Variables can also be annotated with the role they play in model, such as *weights*, *biases*, *filters*, etc. A series of convenience tools were written that allow users to filter the symbolic computation graph based on these annotations, resulting in an approach that allows for powerful queries such as "Apply weight noise to all weights that belong to an LSTM unit whose parent is a brick with the name ..."

4. Fuel

Fuel's goal is to provide a common interface to a variety of data formats and published datasets such as MNIST, CIFAR-10, ImageNet, etc. while making it easy for users to write an interface to new datasets. Fuel allows for efficient ways of iterating over these datasets (sequentially, shuffled, minibatches, etc.). It also provides a variety of on-the-fly preprocessing methods such as random cropping of images, creating n-grams from text files, and the ability to implement many other methods easily.

Blocks relies on Fuel for its data interface, but Fuel can easily be used by other machine learning frameworks that interface with datasets.

5. Serialization and checkpointing

The training of large, deep neural networks can often take days or even weeks. Hence, regular checkpointing of training progress is important. Blocks aims to make the resumption of experiments entirely transparent, even across platforms, while ensuring the reproducibility of these experiments.

This goal is complicated by shortcomings in Python's Pickle serialization module, which is unable to serialize many iterators, which Fuel heavily depends on in order to iterate over large datasets efficiently. To circumvent this we reimplemented the itertools from the Python standard library to be serializable.

As a result, Blocks experiments are able to be interrupted in the middle of a pass over the dataset, serialized, and resumed later, without affecting the final training results.

6. Documentation and community

Blocks and Fuel are well documented, with both API documentation and tutorials available online. Two active mailing lists[1] support users of the libraries. A seperate repository[2] is maintained for users to contribute non-trivial examples of the use of Blocks. Implementations of neural machine translation models (NMT, Bahdanau et al. (2015)) and the Deep Recurrent Attentive Writer (DRAW, Gregor et al. (2015)) model are publicly available examples of state-of-the-art models succesfully implemented using Blocks.

7. NOTES

Some topics to discuss?

- Serialization/resumability
- "Bricks" i.e. parametrized Theano operations
- Graph management through annotation, hierarchy
- Implementations of GRU, LSTM, attention mechanism, convolutional networks
- Monitoring non-Theano variables, aggregation, plotting results
- Automatic downloading and converting of files
- Standard HDF5 format that conveys axis semantics, splits, etc.

Acknowledgments

The authors would like to acknowledge the support of the following agencies for research funding and computing support: NSERC, Calcul Québec, Compute Canada, the Canada Research Chairs and CIFAR. We would also like to thank the developers of Theano.

1. `https://groups.google.com/d/forum/blocks-users` and `https://groups.google.com/d/forum/fuel-users`
2. `https://duckduckgo.com/?q=blocks-examples`

References

Proceedings of the 30th international conference on machine learning (icml'13). In *Proceedings of the 30th International Conference on Machine Learning (ICML'13)*. ACM, -1. URL http://icml.cc/2013/.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. Theano: new features and speed improvements. NIPS Workshop: Deep Learning and Unsupervised Feature Learning, 2012.

James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010.

Ian J. Goodfellow, David Warde-Farley, Pascal Lamblin, Vincent Dumoulin, Mehdi Mirza, Razvan Pascanu, James Bergstra, Frédéric Bastien, and Yoshua Bengio. Pylearn2: a machine learning research library. *arXiv preprint arXiv:1308.4214*, 2013a. URL http://arxiv.org/abs/1308.4214.

Ian J. Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout networks. In ICM (-1), pages 1319–1327. URL http://icml.cc/2013/.

Karol Gregor, Ivo Danihelka, Alex Graves, and Daan Wierstra. Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*, 2015.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning (ICML'13)* ICM (-1). URL http://icml.cc/2013/.