Neural Network Tools: An Overview

From work on "Automatic Classification of Alzheimer's Disease from Structural MRI"
Report for Master's Thesis

Eivind Arvesen
Østfold University College
eivindaa@hiof.no

Abstract

Machine learning has exploded in popularity during the last decade, and neural networks in particular have seen a resurgence due to the advent of deep learning. In this paper, we examine modern neural network software-libraries and describe the main differences between them, as well as their strengths and weaknesses.

1 Introduction

The field of machine learning is now more alluring than ever, helping researchers and businesses to tackle a variety of challenging problems such as development of autonomous vehicles, visual recognition systems and computer-aided diagnosis. There is a plethora of different tools and libraries available that aim to make work on machine learning problems easier for users, and while most of the more general alternatives are quite similar in terms of functionality, they nonetheless differ in terms of approach and design goals.

This report describes several popular machine learning tools, attempts to showcase their strengths and characterizes situations in which they would be well suited for use. The primary focus is on modern tools; Most of these will have some connection to deep learning, as it is an approach that is currently producing many state of the art results in fields like computer vision.

2 Tools

2.1 Pylearn2

Pylearn2 [4] is a cutting edge machine learning library for Python developed at the LISA-lab[1] at The University of Montreal, built with research in mind. It has a particular focus on deep learning. The library is focused on easy configurability for expert users (i.e. machine learning researchers) —unlike "black-box" libraries, which provide good performance without demanding knowledge about underlying algorithms from users. One way to put it would be that the library values ease of experimentation over ease of use.

Though Pylearn2 assumes some technical sophistication on the part of users, the library also has a focus on reusability, and the resulting modularity makes it possible to combine and adapt several re-usable components to form working models, and to only learn about the parts of the library one wants to use. One of the library's goals is to contain reference implementations of all models and algorithms published by the LISA-lab, and it has been used to set the state of the art on several standardized datasets, including a test error of 0.45% on MNIST in 2013 [3] —the best performance without data augmentation at that point —using a convolutional maxout-network with dropout regularization.

Pylearn2 makes use of a YAML[2]-interface, which allows users to set up and perform experiments rapidly by defining their models in

[1] One of recent years' most prominent labs, spearheaded by one of the leading researchers on deep learning, Yoshua Bengio.

[2] A human-readable data serialization format.

an almost declarative style, using (pre-)defined components as building blocks and specifying hyperparameters. Alternatively, experiments may also be specified through a Python script.

The library is built upon the Theano library —a numerical computation library for Python and C/CUDA-compiler —which is also developed at the LISA-lab. Theano can also compute gradients automatically; Users need only specify architecture and loss function. This makes it very easy to experiment quickly.

GPU-based models are also enabled via compilation with Theano, as it can compile both to GPU and CPU. Pylearn2 also provides wrappers to widely used and efficient third-party GPU-implementations of convolutional nets like CUDA-convnet and Caffe.

As the library is under heavy development, its documentation is somewhat sparse, although the source code itself is thoroughly commented. Though its website has a few notebook[3]-style and short examples, users will likely have to spend some amount of time reading source code to figure out how to implement their own models and extensions, as well as how to integrate them (for use) with the library. Pylearn2 can thereby be somewhat hard to understand, extend, modify and debug for beginners. There is, however, a very active community around the library, including usergroups/-mailinglists and the official source code repository[4]. Despite its somewhat steep learning-curve, Pylearn2 is very powerful.

Pylearn2 is to a large extent created and supported by it's users (students at the LISA-lab and other contributors), and since it has its focus on research, features are added as they are needed, meaning that users will likely have to code up some things themselves —unless they are only trying to replicate published results or run variations on old experiments/datasets.

Pylearn2 comes with support for/ready-to-use standardized benchmark datasets (such as MNIST and CIFAR-10) and unsupervised learning out of the box. It appears to be growing steadily in popularity, and is apparently popular amongst contestants in Kaggle contests.

2.2 Torch

Torch [1] is an open source scientific computing framework that supports many different machine learning algorithms. It uses a JIT compilation-based implementation of the Lua-language for ease of use and efficiency, on top of a C/CUDA-implementation. Torch7 (its current version) is currently in use at and being contributed to by DeepMind (Google), Facebook, Twitter and New York University, according to the project's website[5].

It is somewhat similar and comparable to Theano/Pylearn2 (used in conjunction with Numpy and the rest of the packages in the "standard" Python scientific stack), complete with suitable datatypes which supports mathematical operations, stastistical distributions, and BLAS[6] operations, as well as supporting rapid experimentation via a REPL[7]. It is also composed of reusable parts that can be combined in many variations. Torch also has a large ecosystem of community-driven packages, and as with Pylearn2, other packages brings support for things like image processing —including a relatively recently released package[8] of CUDA extensions for deep learning by Facebook AI Research.

Torch allows neural network models to be run on GPUs through the help of CUDA. This can be achieved by users via simple typecasting.

Like Pylearn2, Torch7 includes scripts to load several popular datasets. Third party loaders are also available for datasets other than those that are supported by default.

[3] A web-based (locally interactive) environment provided by the enhanced Python-shell and -project IPython.
[4] Available at `https://github.com/lisa-lab/pylearn2`
[5] `http://torch.ch`
[6] Basic Linear Algebra Subprograms
[7] Read–eval–print loop, i.e. an interactive shell
[8] Available at `https://github.com/facebook/fbcunn`

According to a 2011 paper [2] by some of Torch's maintainers, Torch7 was faster than Theano on most benchmarks. The author's noted, however, that Theano was "faster than any existing implementation" (including Torch5, Matlab with GPUmat and EBLearn), going as far as saying that it *crushed* the alternatives in benchmarks when run on a GPU. It is also important to remember, as the authors comment, that only larger network architectures will benefit from GPU-acceleration.

In the realm of deployment, Torch may prove to be easier in use than for instance Pylearn2 (and MATLAB, which only supports deployment of pre-trained networks), as the LuaJIT environment is embeddable into a host of environments, including smart-phone applications and video games.

Torch is very practical to use for deep learning, as the library has a focus on these approaches. Like Pylearn2, Torch has an active community, including mailing lists/user groups, community wiki, online chat and an official source code repository on github.

2.3 Matlab Neural Network Toolbox

Matlab supports Neural Networks via the Neural Network Toolbox, and networks built with this can in turn be parallelized and run on GPUs when used in conjunction with the Parallel Computing Toolbox.

While it is not supported "out of the box", deep learning architectures can be used in Matlab via the Deep Learning Toolbox[9] third-party software. The toolbox includes support for Deep Belief Nets, Stacked Autoencoders, Convolutional Neural Nets, Convolutional Autoencoders and Feedforward Backpropagation Neural Nets. The toolbox also allegedly works with GNU Octave[10], and includes sample scripts to get news users started. However, the public source code repository does not appear to have been updated since May 2014.

Matlab is very much established both in industry and academia, and it is therefore easier to find helpful tips and snippets of code online, as there is a larger userbase (and indeed more use cases, as it is a language of its own). Matlab also includes integrated development environment.

2.4 Scikit-learn

Scikit-learn [7] is another machine learning library for Python. Building upon widely used packages like numpy (N-dimensional arrays) and scipy (scientific computing), it contains a variety of classification, regression and clustering algorithms, providing machine learning techniques for supervised and unsupervised problems. The package "focuses on bringing machine learning to non-specialists" [7], and is thus more accessible than some alternatives. It does therefore, unlike Pylearn2, not require users to possess knowledge of the models' implementations. Also unlike Pylearn2, which focuses mostly on neural networks, scikit-learn has a wide varity of machine learning techniques available, including support vector machines, logistic regression, nearest neighbors and random forest, covering tasks like classification, regression, clustering, dimensionality reduction, model selection and preprocessing. The libraries website includes comprehensive documentation. It includes limited support for neural networks, but it is not very practical to use for deep learning.

2.5 Weka

Weka is machine learning software [5] from the University of Waikato, New Zealand, written with focus on data mining. It is written in Java, and is usable on its own (through it's GUI "explorer"), as well as callable from one's own Java code. While Weka has been viewed as a helpful tool, it seems to be falling out of favor as big data and deep architectures become more prevalent. It can to some extent still be helpful, but it cannot handle large datasets very

[9] Available at `http://www.mathworks.com/matlabcentral/fileexchange/38310-deep-learning-toolbox`

[10] Open-source alternative to MATLAB, mostly cross-compatible.

well —though it is possible in some cases, with certain models. Although it works well as an exploratory tool for (smaller) datasets, Weka has typically not been used in state of the art machine learning research in later years, and is thought not to be appropriate for heavy lifting amongst some. This seems to be a pervasive attitude on several popular forums and other internet sites. Although how helpful Weka is in various cases is debatable, it is certainly not well suited for use with very large datasets as it's explorer will not be able to handle this even if one adjusts the JVM settings like heap size before launching it.

3 Features

3.1 GPUs and Convolutions

3.1.1 Caffe

Caffe[6] is a very efficient deep learning framework by Berkeley Vision and Learning Center at UC Berkely. It is mostly used for computer vision, i.e. training deep convolutional neural nets. Caffe supports training on CPU/GPU. It is implemented in C++, and has Python and MATLAB bindings. It is a research library, and includes bleeding edge functionality and very high performance.

3.1.2 CUDA-convnet(2)

CUDA-convnet (also sometimes referred to as AlexNet in literature) is a high-performance CUDA-based GPU-implementation of convolutional neural networks. It is made by Alex Krizhevsky, and has been used to achieve state of the art performance on CIFAR-10. CUDA-convnet is also a research library and a new version, which is more efficient on newer GPUs, has recently been released.

3.1.3 cuDNN

CuDNN is a GPU-accelerated library of deep neural network primitives recently released by NVIDIA.

3.2 Comparison of libraries

* = Support via third party.

	Pylearn2	**Torch7**	**Matlab**	**Scikit-learn**	**Weka**
Multilayer perceptron	Yes	Yes	Yes	No	Yes
Autoencoder	Yes	Yes	Yes*	No	No
Boltzmann Machine	Yes	Yes*	Yes*	Yes	No
Convolution	Yes	Yes*	Yes*	No	No
Recurrence	Yes	Yes*	Yes	No	No
Support Vector Machine	Yes	Yes*	Yes	Yes	Yes
K-means	Yes	Yes*	Yes	Yes	Yes
Principal Component Analysis	Yes	Yes*	Yes	Yes	Yes
Deep Belief Network	Yes	No	Yes*	No	No

Table 1: Network Models

	Pylearn2	**Torch7**	**Matlab**	**Scikit-learn**	**Weka**
Support Vector Machine	Yes	Yes*	Yes	Yes	Yes
K-means	Yes	Yes*	Yes	Yes	Yes
Principal Component Analysis	Yes	Yes*	Yes	Yes	Yes
Decision Trees	No	No	Yes	Yes	Yes
Random Forest	No	No	Yes*	Yes	Yes

Table 2: Other Models

	Pylearn2	**Torch7**	**Matlab**	**Scikit-learn**	**Weka**
Sigmoid	Yes	Yes	Yes	No	Yes
Tanh	Yes	Yes	Yes	No	No
Linear	Yes	Yes	Yes	No	No
ReLU	Yes	Yes	Yes*	No	No
Softplus	No	Yes	Yes	No	No
Softmax	Yes	Yes	Yes	No	No
Maxout	Yes	No	No	No	No

Table 3: Transfer Functions

	Pylearn2	**Torch7**	**Matlab**	**Scikit-learn**	**Weka**
(Minibatch) Stochastic Gradient Descent	Yes	Yes	Yes	No	No
Batch Gradient Descent	Yes	Yes	Yes	No	No
Levenberg-Marquardt	No	No	Yes	No	No

Table 4: Training Algorithms

	Pylearn2	**Torch7**	**Matlab**	**Scikit-learn**	**Weka**
L1 Weight Decay	Yes	Yes	Yes*	No	No
L2 Weight Decay	Yes	Yes	Yes*	No	No
Momentum	Yes	Yes	Yes*	No	No
Dropout	Yes	Yes	Yes*	No	No

Table 5: Regularization

	Pylearn2	**Torch7**	**Matlab**	**Scikit-learn**	**Weka**
GPU acceleration	Yes	Yes	Yes	No	No
Support for standardized benchmark datasets	Yes	Yes	No	No	No
Bindings for Caffe	Yes	No	No	No	No
Bindings for CUDA-convnet	Yes	Yes	No	No	No
Bindings for cuDNN	Yes	Yes	No	No	No

Table 6: Other Features

4 Conclusion

In this paper, we have explored several modern neural network software-libraries, particularly those that support state of the art techniques (including deep learning.)

Most of the libraries are very similar. Support for deep models and GPU-acceleration are relatively common between the alternatives —if not out of the box, then by third-parties. Many of the newer options among the software are either partially implemented in or had bindings to a succint, expressive language. Also common (especially among the newer libraries) are relatively liberal, open source licenses (e.g. BSD), and having very active communities of users/developers contributing to further discussion and development.

Pylearn2 values ease of experimentation over ease of use. It has minimal documentation, but it makes it very easy to experiment with powerful models.

Torch7 is also a serious contender, and is very efficient. It has a large selection of third-party packages, and many leading research-groups contributes to its development.

Both Pylearn2 and Torch7 are modern research-tools that enable deep learning and extremely good performance because of (almost transparent) GPU-acceleration.

Matlab is tried and true, documentation is available and one is sure to find available expertise and code if one needs help. State of the art performance and models, however, are not guaranteed out of the box, as MATLAB is not as close to the bleeding edge as other tools. There is a deep learning toolbox available from a third party, but its repository has currently not been updated for a year, and the field of neural networks (and deep learning in particular) is in rapid development.

While Weka may be of use in exploratory stages, it is not really usable for deep learning, and is indeed not very focused on neural networks.

Scikit-learn has a broad focus when it comes to machine learning techniques, and is also very much approachable for non-expert users. However, it does not offer neither the efficiency nor the large collection of neural network-related models that some of the alternatives do, as it is currently mostly limited to unsupervised learning in the form of RBMs[11].

The author proposes that when making the choice between what tools to use, potential users should first consider what features are needed from the software (like models or algorithms) and see which library makes them available. Then they might concider what amount of prior knowledge they possess of the tool and the language, and the time that will eventually spent to learn new things, and so on.

If being on the bleeding edge is important, one may wish to look into Pylearn2 or Torch7 first, as they are under rapid development by some of the field's leading researchers.

[11] Restricted Boltzmann Machines

References

[1] Ronan Collobert, Samy Bengio, and Johnny Marithoz. *Torch: A Modular Machine Learning Software Library*. 2002.

[2] Ronan Collobert, Koray Kavukcuoglu, and Clément Farabet. "Torch7: A matlab-like environment for machine learning". In: *BigLearn, NIPS Workshop*. 2011. URL: http://infoscience.epfl.ch/record/192376/files/Collobert_NIPSWORKSHOP_2011.pdf (visited on 03/01/2015).

[3] Ian J. Goodfellow et al. "Maxout Networks". In: *arXiv:1302.4389 [cs, stat]* (Feb. 18, 2013). arXiv: 1302.4389. URL: http://arxiv.org/abs/1302.4389 (visited on 03/16/2015).

[4] Ian J. Goodfellow et al. "Pylearn2: a machine learning research library". In: *arXiv:1308.4214 [cs, stat]* (Aug. 19, 2013). arXiv: 1308.4214. URL: http://arxiv.org/abs/1308.4214 (visited on 02/07/2015).

[5] Mark Hall et al. "The WEKA data mining software: an update". In: *ACM SIGKDD explorations newsletter* 11.1 (2009), pp. 10–18. URL: http://dl.acm.org/citation.cfm?id=1656278 (visited on 02/09/2015).

[6] Yangqing Jia et al. "Caffe: Convolutional Architecture for Fast Feature Embedding". In: *arXiv:1408.5093 [cs]* (June 20, 2014). arXiv: 1408.5093. URL: http://arxiv.org/abs/1408.5093 (visited on 03/12/2015).

[7] Fabian Pedregosa et al. "Scikit-learn: Machine learning in Python". In: *The Journal of Machine Learning Research* 12 (2011), pp. 2825–2830. URL: http://dl.acm.org/citation.cfm?id=2078195 (visited on 02/10/2015).