A Comprehensive Classification of Deep Learning Libraries

Hari Mohan Pandey and David Windridge

Department of Computer Science, School of Science and Technology
Middlesex University, London, NW4 4BT, U.K.
{h.pandey,d.Windridge}@mdx.ac.uk

Abstract. Deep Learning (DL) networks are composed of multiple processing layers that learn data representations with multiple levels of abstraction. In recent years, DL networks have significantly improved the state-of-the-art across different domains, including speech processing, text mining, pattern recognition, object detection, robotics and big data analytics. Generally, a researcher or practitioner who is planning to use DL networks for the first time faces difficulties in selecting suitable software tools. The present article provides a comprehensive list and taxonomy of current programming languages and software tools that can be utilized for implementation of DL networks. The motivation of this article is hence to create awareness among researchers, especially beginners, regarding the various languages and interfaces that are available to implement deep learning, and to provide a simplified ontological basis for selecting between them.

Keywords: Deep Learning, Deep Learning Libraries, Machine Learning, Deep Belief Network.

1 Introduction

Stochastic (as opposed to symbolic) Machine Learning (ML) techniques have attracted considerable research attention over recent years; however, the performance of traditional stochastic ML is limited by the implicit disconnect between feature-representation and classification, a distinction that is not evident in human information processing (e.g. speech and vision). To overcome this, deep hierarchical structures of human speech perception and production were proposed in late 20th century. Hinton et al. [1] suggested the deep structured programming architecture referred to as the Deep Belief Network (DBN) which provided a breakthrough instance in Deep Learning (DL). (DL methods were originated from the study of 3-layer Artificial Neural Networks (ANNs) [6], an active research domain over several decades [7] [8] [9]). DL algorithms have gone on to demonstrate significant impact across various fields of science and engineering, providing a step-change improvement in human-level artificial intelligence [3] [4] [5]. DL methods have, in particular, made substantial progress in big data analytics, speech recognition, natural language processing,

machine vision, pattern recognition, health informatics, image processing and financial analysis

Creation of a standard Neural Network (NN) requires neurons producing real valued activations in relation to inputs which are trained by adjusting connection weights until the network as a whole shows the desired behaviour. Hinton et al. [4] suggested a new training method, Layered Wise Greedy Learning, (LWGL) in order to overcome difficulties associated with back-propagation learning of >3 layer networks, thereby marking the birth of DL networks. LWGL proposed following the idea: "*unsupervised learning method should be utilized before the layer-by-layer training to determine reduced data dimension and a compact representation prior to the export of features to the next layer, leading to creation of a fine-tuned network with labelled data*" [10]. Other techniques were subsequently developed to enable learning to take place over a large number of layers, all of which are termed DL. In pragmatic terms, DL attracted the attention of the researchers for two key reasons: *better handling of data over fitting* and *better guarantees of convergence (*DL techniques often apply pre-training of data using unsupervised learning [10]).

Various programming languages and tools have been proposed to create and implement DL networks. But, for the beginner, the key question is: "*how to select an appropriate programming language and software tool to implement DL network according to the problem type and complexity?"*.

The objective of this paper is to explore and classify various software tools used for DL networks. This research serves the researchers, practitioners and educators who are interested in DL with emphasis on the answering the aforementioned question i.e. selection of an appropriate programming languages and software tool to implement DL concepts.

The remaining of this paper is organized as follows: Section 2 presents rational and motivation of conducting this research. Section 3 shows the basics of DL, illustrates layered structure of hardware and software used for DL network and briefly describes the software tools used for DL networks. Classification of DL software tools is given in Section 4. Finally, we draw the conclusions of this paper in Section 5.

2 Rational and Motivation

Since 2007, DL has been successfully utilized across domains including image processing, computer vision, speech recognition, natural language processing, robotics etc. DL achieves it successes through its representational ability in relation to training data (input data) through multiple layers of artificial neurons [18]. DL algorithms have been effectively parallelised and implemented on GPU, which significantly reduces training time [19]. Various open-source DL toolkits are available which have been used to improve the efficiency of DL methods. Some of the popular toolkits are: *Caffe* [20], *CNTK* [21], *TensorFlow* [22], *Torch* [23], *MXNet* [24], *Theano* [25], and *PaddlePaddle* [26] etc. These tools support multi-core CPUs and many-core GPU's. A key objective of DL is to learn a number of weights in layered manner of NNs. These weights are implemented either by vectors or matrix opera-

tions. *Tensorflow* utilizes Eigen [27] to speed up matrix operation library. On the other hand, *Caffe*, *CNTK*, *MXNET* and Torch incorporate *OpenBLAS* [28], *Intel MKL* [29] or *cuBLAS* [30] to accelerate matrix operations. A key point to note is that all of these DL software libraries import *cuDNN* – a GPU accelerated DL library for NNs [31]. The aforementioned software libraries are used for DL, but they exhibit different running performance even when training the same NN on the same hardware platform. This variation is due to the difference of optimization methods employed (by vendors). In addition, the running performance of these software libraries varies when training different types of NNs or utilizing different kinds of hardware.

We thus present a comprehensive classification of software libraries used for DL with the belief that it might be useful to the end users in the selection of an appropriate software tool for DL tasks.

3 Deep Learning and Layered Structure

DL is a set of ML methods that utilizes NNs with multiple hidden layers to accomplish specific tasks [12] [13] with a range of applications across domains such as image classification (Google Photo and Facebook), speech recognition, language learning, online and mobile service (voice recognition and dialog system of Siri). Furthermore, DL has been used in automotive industry for computer vision (autonomous cars driving and robotics), optimizations (monitoring of quality issues and tasks scheduling) and others. Due to its high level of applicability of DL the landscape of infrastructure, programming languages and tools for training and deploying the DL networks is evolving rapidly.

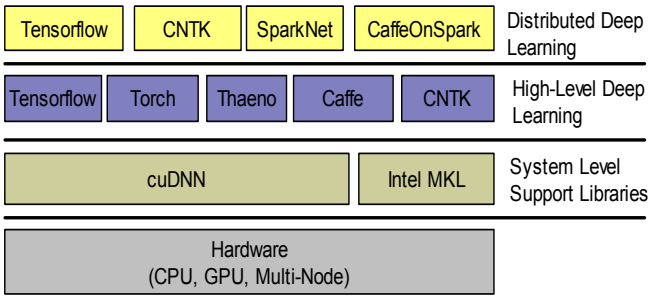

Fig. 1. Hardware and software layers of commercial deep learning systems

Training and scaling of Deep Neural Networks (DNNs) is challenging with regard to large data sizes and multiple layers. In contrast to simpler models (e.g. K-means, Support Vector Mechanism (SVMs) and logistic regression), DL involves millions rather than just hundreds of parameters for a typical convolution NN, requiring larger datasets of e.g. video, images and text to train a network. Training DL networks requires distributed processing, scalable storage (e.g. HDFS), computing capabilities and accelerators (GPU, FPGAs). Deployment of a DL model is thus a challenging task requiring billions of operations for a single inference.

Fig.1 illustrates the layered structure of hardware and software of a DL system. GPUs have been widely used to scale NNs with several DL libraries extant that rely on GPUs to optimize and train NNs [15] on different platforms e.g. NVIDIA's cuDNN [16] and Intel's MKL [17]. On top of these DL libraries, several frameworks have been constructed, providing e.g. integrated support for distributed training, whilst others utilize different distributed runtime engines.

4 Deep Learning Software Libraries Classification

Various programming languages such as Python, MATLAB, Java, Java Script, .Net, C++ etc. are used to implement DL methods. Several DL libraries for different programming languages and interfaces have emerged over the last few years. Some of the popular DL libraries are listed in Table 1. All these software frameworks have ability to customize training and model their parameters. But they differ from each other in relation to certain key design decisions:

- Software tools that focus on a high-level and easy-to-use abstraction of DL (e.g. Theano and caffe).
- Software tools that provide low-level primitives (e.g. Tensorflow).
- Software tools that provide unified abstraction of the backend (e.g. Keras, Theano, Tensorflow and Lasagne).

Table 1 presents a classification metric of various DL software libraries. We have classified the DL software libraries based on the following parameters: Interface, Platform (A1), Open Source (A2), Pre-Trained Model (A3), Recurrent nets (A4), Convolution nets (A5), Restricted Boltzmann Machine (A6), Deep Belief Network (A7), OpenMP (A8), OpenCL (A9), CUDA (A10), Parallel Execution (A11).

Table 1 is thus intended to be of use to the end-user in identifying an appropriate DL software library to use based on their answers to the following questions: *Type of network (Recurrent, convolution, restricted Boltzmann and deep belief network)?, Type of environment (OPEN CL, OPEN MP and CUDA) required for executing a program?, Type of processing (Parallel and distributive) supported by the software library?*. A classification of DL libraries based on their features is presented in Table 2. We tried to explore DL libraries sufficiently so that a beginner can understand the utility of a particular library. Here, our aim is not to decide the superiority of any particular DL library over the other DL libraries rather we are intended to present the applicability of DL libraries in difference scenarios.

Table 1. Deep learning software tools classification metric

Interface	Software Tool	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11
Python	Theano	C	Y	**Y1**	Y	Y	Y	Y	Y	**Y2**	Y	Y
	Keras	MWLX	Y	Y	Y	Y	Y	Y	**Y3**	**Y4**	Y	Y
	Lasagne	MWLX	Y	Y	Y	Y	Y	Y	**?**	**?**	Y	Y
	Blocks	C	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
	Caffe	MWLX	Y	Y	Y	Y	**N**	**N**	Y	**UD**	Y	**N**
	Cafee2	C	Y	Y	Y	Y	**N**	**N**	Y	**UD**	Y	Y
	nolearn	MWLX	Y	Y	Y	Y	**N**	Y	**?**	Y	Y	Y
	Gensim	MWLX	Y	Y	Y	Y	Y	**?**	**Y5**	**Y6**	**?**	**?**

	Chainer	MWLX	Y	Y	Y	Y	Y	?	Y	Y	Y	Y
	Hebel	C	Y	Y	N	Y	Y	Y	?	?	Y	Y
	CXXNET	MWLX	Y	Y	Y	Y	N	Y	Y	N	Y	Y
	DeepPy	MWLX	Y	Y	Y	Y	Y	Y	?	Y	Y	Y
	Neon	MWLX	Y	Y	Y	Y	Y	Y	?	?	Y	Y
	H2O	MWLX	Y	Y	Y	Y	Y	Y	?	Y	Y	Y
	Tensorflow	MWLX	Y	Y	Y	Y	Y	Y	N	Y	Y	Y
	MXNet	P1	Y	Y	Y	Y	Y	Y	Y	N	Y	Y
MATLAB	ConvNet	MWLX	Y	Y	Y	Y	N	N	N	N	Y	Y
	DeepLearn-pLearn-ToolBox	MWLX	Y	Y	Y	Y	N	Y	?	Y	Y	Y
	cuda-convnet	MWLX	Y	Y	Y	Y	Y	Y	Y	?	Y	Y
	Mat-ConvNet	MWLX	Y	Y	Y	Y	N	N	N	N	Y	Y
C++	EBLearn	C	Y	Y	Y	Y	N	Y	N	N	Y	Y
	SINGA	MWLX	Y	Y	Y	Y	Y	Y	N	Y	Y	Y
	NVIDIA DIGITS	MWLX	Y	N	Y	Y	N	Y	Y	Y	Y	Y
	Intel® Deep Learning Framework	MWLX	Y	Y	Y	Y	?	?	Y	Y	Y	Y
	Dlib	C	Y	Y	N	Y	Y	Y	Y	N	Y	Y
	Microsoft Cognitive Toolkit	MWLX	Y	Y	Y	Y	N	N	Y	N	Y	Y
Java	ND4J	C	Y	Y	Y	?	N	Y	Y	Y	Y	Y
	Deeplearn-ing4J	C	Y	Y	Y	Y	?	Y	Y	N	Y	Y
	Encog	C	Y	Y	Y	Y	N	Y	Y	Y	Y	Y
	RL4J	C	Y	Y	Y	Y	N	Y	Y	N	Y	Y
JavaScript	Convnet.js	MWLX	Y	Y	Y	Y	N	N	Y	N	Y	Y
Lua	Torch	MWLX	Y	Y	Y	Y	Y	Y	Y	Y7	Y	Y
Julia	Mocha	MWLX	Y	Y	?	Y	N	Y	Y	Y	Y	Y
Lisp	Lush	MWLX	Y	?	?	Y	Y	Y	Y	?	Y	Y
Haskell	DNNGraph	MWLX	Y	Y	Y	Y	?	Y	?	?	Y	Y
.NET	Ac-cord.NET	MWLX	Y	N	N	Y	N	N	N	N	Y	Y
R	darch	MWLX	Y	Y	Y	Y	Y	Y	?	?	Y	Y
	deepnet	MWLX	Y	Y	Y	Y	Y	Y	?	?	Y	Y

A1: Platform, A2: Open Source, A3: Pre-Trained Model, A4: Recurrent nets, A5: Convolution nets, A6: Restricted Boltzmann Machine, A7: Deep Belief Network, A8: OpenMP, A9: OpenCL, A10: CUDA, A11: Parallel Execution, C: Cross platform, MWLX :(MAC OS, X, Linux, Windows), Y: Yes, N: No, Y1: Using Lasagne's model zoo, Y2: Underdevelopment, Y3: Yes, if Theano/MXNet is present in backend, Y4: Only for Theano and Tensorflow, Y5: only for Keras/Theano, Y6: Yes, only for Tensorflow, Y7: Implemented through cltorch or Cheatsheet, UD: Under Development, P1: (MACOS, Windows, Linux, Android, IOS, JavaScript), ?: Not known.

Table 2. Classification of Deep Learning Libraries according to their features

Software Tool	Remarks
Theano	Used to evaluate mathematical expressing and able to handle numerical arrays.
Keras	It is a highly modular NN library, used for optimized tensor manipulation.
Lasagne	It is a lightweight library used to create and train NNs. It works on six principles: simplicity, transparency, modularity, pragmatism, restraint and focus.
Blocks	It is framework used to create NN models on top of Theano. Construct parameterized Theano operations (known as bricks). It annotates the Theano computational graph and maintains the flexibility effectively.
Caffe	It is mainly used in academic research projects and other industrial application. It is also

	applicable for large scale projects.
Cafee2	It is used to train large machine learning models. It helps in incorporating artificial intelligence in mobile applications [11].
nolearn	Unlike any other Deep Neural Network (DNN), nolearn is an abstraction layer on top of existing packages such as Theano and lasagna. It can work successfully without Graphics Processing Unit (GPU).
Gensim	Designed for large text collections, using data streaming and an efficient incremental learning. It supports various language processing algorithms.
Chainer	A 2nd generation DL framework. Chainer implements CuPy that allows GPU for faster computation. It is partially compatible with NumPy that effectively supports existing multi-dimensional array library.
CXXNET	It is developed on MShadow (a Lightweight CPU (or GPU) Tensorflow library) that helps in writing expression for ML.
DeepPy	It is developed on top of NumPy with GPU acceleration. It fills the gap between high performance NNs and ease of development using Python or NumPy [12].
Neon	Neon has syntax similar to Theano/Keras. It has feature of Machine Learning Operations (MOP) Layer that enables other frameworks to use Neon. It has been ranked as the fastest framework over various benchmarks.
H2O	It is fast, memory efficient, based on columnar compression and fine grain MapReduce. It reduces the run time and model tuning through model checkpointing.
TensorFlow	Both TensorFlow and Theano provide very similar systems. But, TensorFlow has better support for distributed systems. TensorFlow programs are generally structured in to a construction phase that assembles a computation graph.
MXNet	It provides interface to build various types (Feedforward, Recurrent and Convolution) neural network. It works on three main concepts: NDArray (offers matrix and tensor computation), Symbol (used to define neural network) and KVStore (used to achieve data synchronization among GPUs).
ConvNet	It is MATLAB based convolution neural network toolbox for classification and segmentation, can be trained on both CPU and GPU.
DeepLearn-ToolBox	It is open source MATLAB toolbox for deep learning. It is easy to use, but slow (only CPU version is available).
cuda-convnet	It provides an efficient implementation of convolution neural network in CUDA. It avoids use of temporary memory and it has ability to optimize multiple objectives simultaneously.
MatConvNet	It implements convolution neural networks for MATLAB. It is popular in due to its deep integration with MATLAB environment. Unlike Deep Learn Toolbox, MatConvNet support both CPU and GPU.
EBlearn	It describes both supervised and unsupervised training methods for probabilistic and non-probabilistic factor graph. It is composed of two key components: *libidx* and *libeblearn*.
SINGA	It provides flexible architecture for distributed training and used in health-care applications.
NVIDIA DIGITS	It is not a framework but it provides a graphical web interface to other frameworks such as Tensorflow, Torch, Caffe etc.
ND4J	It is a distributed deep learning library, can be implemented on CPUs and GPUs and it provides Java and Scala APIs.
Deeplearning4J	It is domain specific library to configure deep learning networks. Hyperparameters are used to determining how neural networks learn.
Encog	It is a framework that provides a GUI based workbench to train machine learning algorithm.
Convnet.js	JavaScript deep learning models for neural network, which entirely based on browser.
Torch	Fast computing framework based on Lua-JIT with strong CPU, CUDA and Tensor library. It can support multi-GPU and parallelization.
Mocha	It is JavaScript unittest framework that runs on Node.js and in the browser.
Lush	It provide huge library for numerical, image, graphics and signal processing routines.
DNNGraph	It is a DSL model generation in Haskell that uses lens and fgl graph libaray to specify network layout.
Accord.NET	It is a framework in .NET, divided into libraries, used for wide range of scientific computing.
darch	It provides native implementation of deep neural network with learning algorithms in R. It consists of three classes: Net (represents an abstract class), RBM (Restricted Boltzmann Machine) - used in pre-training and DArch (represents deep neural networks for pre-training and fine tuning).
deepnet	It has 3×3 architecture for deep learning. It is relatively smaller than darch.

5 Conclusions

In this paper, we have set out a classification of different DL software libraries for the practice-oriented new-comer to deep-learning. We illustrated the layered structure of hardware and software (see Fig. 1) of a DL system which demonstrated the role of DL libraries at various level (distributed, High-level and system level) of abstraction.

The main contribution of this paper is that it presents a comprehensive classification of various DL software libraries (Table 1 and 2). We have carried out the classifications based on various parameters and features in relation to the answer to three key questions (see Section 4): *what type of network is to be implemented using the software library, which type of environment should the software library support and what type of execution environment is required for implementation of software libraries*. Our intention to conduct this research is not to conclude the superiority of a particular DL library over any other DL libraries rather we presented the classification according to the applicability of DL libraries. We believe that the outcome of this investigation will be potentially useful to the end-user in determining which DL software library to use in relation to DL tasks.

Acknowledgment. Authors would like to acknowledge financial support from the Horizon 2020 European Research project DREAM4CARS (#731593).

References

1. G. E. Hinton, S. Osindero and Y.W. Teh. "A fast learning algorithm for deep belief nets." Neural computation 18.7 (2006): 1527-1554.
2. G. Indiveri and S. C. Liu. "Memory and information processing in neuromorphic systems." Proceedings of the IEEE 103.8 (2015): 1379-1397.
3. Z.H. Ling, S.Y. Kamg, H. Zen and A Senior. "Deep learning for acoustic modeling in parametric speech generation: A systematic review of existing techniques and future trends." IEEE Signal Processing Magazine 32.3 (2015): 35-52.
4. J. Schmidhuber. "Deep learning in neural networks: An overview." Neural networks 61 (2015): 85-117.
5. D. Yu and L Deng. "Deep learning and its applications to signal and information processing [exploratory dsp]." IEEE Signal Processing Magazine 28.1 (2011): 145-154.
6. G. E. Hinton and R.R. Salakhutdinov. “Reducing the dimensionality of data with neural networks.” science, 313(5786). 2016, 504-507.
7. N. Hou, H. Dong, Z. Wang, W. Ren and F.E.Alsaadi. "Non-fragile state estimation for discrete Markovian jumping neural networks." Neurocomputing 179 (2016): 238-245.
8. F. Yang,, H. Dong, Z. Wang,W. Ren and F.E. Alsaadi.”A new approach to non-fragile state estimation for continuous neural networks with time-delays”. Neurocomputing, 197 (2016), 205-211.
9. Y. Yu, H. Dong,, Z. Wang., W. Ren and F.E.Alsaadi. “Design of non-fragile state estimators for discrete time-delayed neural networks with parameter uncertainties”. Neurocomputing, 182 (2016), 18-24.

10. D. Erhan, Y. Bengio, A. Courville, P.A. Manzagol, P. Vincent and S. Bengiol. "Why does unsupervised pre-training help deep learning?." Journal of Machine Learning Research 11.Feb (2010): 625-660.
11. https://caffe2.ai/blog/2017/04/18/caffe2-open-source-announcement.html
12. A. B, L. Larsen. "DeepPy: Pythonic deep learning." (2016).
13. I. Goodfellow, Y. Bengio and A. Courville. "Deep learning. Book in preparation" for MIT Press." URL< http://www. Deep learning book. org (2016).
14. J. Schmidhuber. "Deep learning in neural networks: An overview." Neural networks 61 (2015): 85-117.
15. N. Vasilache, J. Johnson, M. Mathieu, S. Chintala, S. Piantino and Y. LeCun. "Fast convolutional nets with fbfft: A GPU performance evaluation." arXiv preprint arXiv:1412.7580 (2014).
16. S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro and E. Shelhamer. "*cudnn: Efficient primitives for deep learning*." arXiv preprint arXiv:1410.0759 (2014).
17. V. Pirogov and F. Gennady. "Introducing DNN primitives in Intel® Math Kernel Library". https://software.intel.com/en-us/articles/introducing-dnn-primitives-in-intelr-mkl, 2016
18. Y. LeCun, Y. Bengio and G. Hinton "*Deep learning.*" *Nature* 521.7553 (2015): 436-444.
19. L. Deng "Three classes of deep learning architectures and their applications: a tutorial survey." APSIPA transactions on signal and information processing (2012).
20. Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell,. "*Caffe: Convolutional architecture for fast feature embedding.*" *Proceedings of the 22nd ACM international conference on Multimedia*. ACM, 2014, pp. 675–678
21. Yu, Dong, et al. "An introduction to computational networks and the computational network toolkit." *Microsoft Technical Report MSR-TR-2014–112* (2014).
22. M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin et al."*Tensorflow: Large-scale machine learning on heterogeneous distributed systems*." *arXiv preprint arXiv:1603.04467* (2016).
23. R. Collobert, K. Kavukcuoglu and C. Farabet. "Torch7: A matlab-like environment for machine learning." BigLearn, NIPS Workshop. No. EPFL-CONF-192376. 2011.
24. Chen, Tianqi, et al. "Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems." *arXiv preprint arXiv:1512.01274* (2015).
25. Al-Rfou, Rami, et al. "Theano: A Python framework for fast computation of mathematical expressions." *arXiv preprint* (2016).
26. "PaddlePaddle: PArallel Distributed Deep Learning", https://github.com/PaddlePaddle/Paddle, 2017, accessed: 2017 – 08 – 14
27. B. Jacob and G. Guennebaud. "Eigen is a C++ template library for linear algebra: Matrices, vectors, numerical solvers, and related algorithms." 2012. http://eigen.tuxfamily.org/index.php?title=Main_Page
28. Z. Xianyi, W. Qian, and W. Saar. "OpenBLAS: An optimized BLAS library." *Accedido: Agosto* (2016).
29. E. Wang, Q. Zhang, B. Shen, G. Zhang, X. Lu, Q. Wu and Y. Wang."Intel math kernel library." High-Performance Computing on the Intel® Xeon Phi™. Springer International Publishing, 2014. 167-188.
30. Toolkit, C. U. D. A. "4.0 cublas library,"." *Nvidia Corporation* 2701 (2011): 59-60.
31. S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro and E. Shelhamer "cudnn: Efficient primitives for deep learning." *arXiv preprint arXiv:1410.0759* (2014).