A Comparative Study of Open Source Deep Learning Frameworks

Ali Shatnawi, Ghadeer Al-Bdour, Raffi Al-Qurran and Mahmoud Al-Ayyoub
Jordan University of Science and Technology, Irbid, Jordan
{gwalbdour13, rlalqurran11}@cit.just.edu.jo, {ali, maalshbool}@just.edu.jo

***Abstract*—Deep Learning (DL) is one of the hottest trends in machine learning as DL approaches produced results superior to the state-of-the-art in problematic areas such as image processing and natural language processing (NLP). To foster the growth of the DL community, several open source frameworks appeared providing implementations of the most common DL algorithms. These frameworks vary in the algorithms they support and in the quality of their implementations. The purpose of this work is to provide a qualitative and quantitative comparison among three of the most popular and most comprehensive DL frameworks (namely Google's TensorFlow, University of Montreal's Theano, and Microsoft's CNTK). The ultimate goal of this work is to help end users make an informed decision about the best DL framework that suits their needs and resources. To ensure that our study is as comprehensive as possible, we conduct several experiments using multiple benchmark datasets and measure the performance of the frameworks' implementation of different DL algorithms. For most of our experiments, we find out that CNTK's implementations are superior to the other ones under consideration.**

***Index Terms*—Deep Learning, CNN, TensorFlow, Theano, CNTK, MNIST, CIFAR-10**

I. Introduction

Deep Learning (DL) is a subset of machine learning which relies on a set of algorithms. The idea of DL is to train a multi-layer Artificial Neural Network (ANN) on a dataset in order to allow it to handle real world tasks. Although the theoretical concepts behind DL are not new, DL has enjoyed a surge of interest over the past decade due to many factors including its successful application to several problems (many of which have great commercial potentials) and the improved affordability of the computing infrastructure it requires. DL approaches have significantly outperformed state-of-the-art approaches in many classical problems of many fields such as image processing, computer vision, speech processing, natural language processing (NLP), etc. Moreover, the scientific community (from both the academia and the industry) has quickly and massively adopted DL. Open source implementations of successful DL algorithms quickly appeared on code sharing websites, and people immediately took them and started employing and improving them in their research, applications, etc. [1], [2].

Several DL frameworks exist such as TensorFlow, Theano, CNTK, Caffe, Torch, Neon, pylearn, etc. Each of these frameworks has different features and performance characteristics. Also, each framework tries different techniques to optimize its implementation of DL algorithms. So, although the same algorithm is implemented in different frameworks, the performance of the different implementations can vary greatly. People wanting to employ such an algorithms in their research or applications face a difficult c hoice s ince the number of different implementations is high and the effort invested by the research community in scientifically comparing these implementations is limited [3], [4].

In this work, we aim at providing qualitative and quantitative comparison between popular open source DL frameworks. To be more specific, we focus on two very popular DL frameworks, namely TensorFlow (from Google), and CNTK (from Microsoft). These frameworks support multi-core CPUs as well as multiple GPUs. All of them import cuDNN, which is a DL library from NVIDIA that supports highly tuned implementations for standard routines such as forward and backward convolution, normalization, pooling, and activation layers. We compare these frameworks by training different neural network (NN) architectures on different standard bench-mark datasets for various tasks in image processing. Despite their importance, comparative studies like ours (especially, the ones focusing on performance issues) are rare.

A comparative study of the frameworks is important in order to enable people who are interested in applying DL in their research and/or applications to make informed decisions about which of the existing frameworks suits their needs. In the following section, a literature survey of the previous work in this field is presented.

The rest of this paper is organized as follows. The following section discusses the existing efforts on comparing DL frameworks highlighting what distinguishes our work. Section II starts by explaining in details what neural networks are. Section III focuses on the frameworks themselves, the way they were used to train the datasets and a brief comparison between the used ones. Experimental results are discussed in details at Section IV, and the work is concluded with final thoughts at Section VII.

II. Literature Survey

Only few researchers did a comparative study between state-of-the-art DL frameworks running on different hardware platforms (CPU and GPU) to highlight the advantages and limitations for each framework when applied on different deep NN architectures, and to enable developers to optimize the running performance of DL frameworks.

One of the first and most prominent examples of comparative studies between DL frameworks was carried out by Bahrampour et al. [3]. The authors compared five DL frameworks: Caffe, Neon, TensorFlow, Theano, and Torch, in terms of speed (gradient computation time and forward time), hardware utilization and extensibility (ability to support different types of DL architectures) after applying various convolutional algorithms on the aforementioned frameworks. They conducted their experiments on a single machine for both CPU (multithreaded) and GPU (Nvidia Titan X) environments. The comparison between frameworks was carried out by training convolutional and stacked autoencoder networks on the MNIST dataset and the ImageNet dataset. They also trained LSTM network on the IMDB dataset [5].

Shi et al. [4] did a comparative study between several DL frameworks including Caffe, MXNet, CNTK, TensorFlow, and Torch. They considered three types of neural networks including; fully connected neural networks (FCN), convolutional neural networks (CNN) and recurrent neural networks (RNN). Moreover, they used different hardware environments including two CPU platforms and three GPU platforms. They used running time and convergence rate in order to evaluate selected frameworks. They used synthetic datasets to measure running time performance and real-world datasets to measure the convergence rate in their experiments.

Goldsborough [6] showed the timeline of machine learning hardware libraries for GenAI. He focused on TensorFlow's results and its basic properties including computational paradigms, its distributed model and programing interface. He compared Ten-sorFlow to other DL frameworks including Theano, Torch and Caffe, qualitatively and quantitatively. In qualitative terms, he compared aforementioned frameworks using several categories including frontends, programming model style, how gradients are computed, and distributing the execution of computational graph. In quantitative terms, he reviewed works (such as [3],[7], [8]), which contain comparisons between TensorFlow and other DL frameworks.

Chintala [7] applied different ImageNet benchmarks for a variety convolutional network types including AlexNet, GoogleNet, Overfeat, and OxfordNet using different open-source DL frameworks such as Caffe, Theano, Torch, TensorFlow, Chainer, etc. He conducted his experiments on NVIDIA Titan X GPU and two new packages for Fast Fourier transform (FFT) computation [9]. The first one is based on the NVIDIA library (cuFFT) and another based on a Facebook FFT (fbfft). Moreover, he used native version of each DL frameworks. After his experiments, he found that using fbfft resulted in a speedup over cuFFT for all applied CNNs. Also, he found that the fastest framework for CNN was Torch followed by Tensorflow.

A Theano development team of Al-Rfou et al. [8] discussed the Theano framework, its features, how to use it, and showed recent improvements on it. They did a performance comparison between Theano and other frameworks including Torch and TensorFlow on three types of machine learning models including CNN, RNN and sequence-to-sequence mapping RNN.

TABLE I
FRAMEWORKS USED FOR THIS COMPARATIVE STUDY

Framework	Major Version	Github Commit ID
CNTK	2.0	7436a00
TensorFlow	1.2.0	49961e5
Theano	0.10.0.dev1	8a1af5b
Keras	2.0.5	78f26df

TABLE II
PROPERTIES OF THE CONSIDERED FRAMEWORKS

Property	CNTK	TensorFlow	Theano	Keras
Core	C++	C++	Python	Python
CPU	✓	✓	✓	✓
Multi-Threaded CPU	✓	Eigen	Blas, conv2D, Limited OpenMP	✓
GPU	✓	✓	✓	✓
Multi-GPU	✓	✓	X (experimental version)	✓
Nvidia cuDNN	✓	✓	✓	✓

Finally, they showed the computation speed using multiple GPUs.

Kovalev et al. [10] presented a comparative study between different DL frameworks namely Theano, Torch, Caffe, TensorFlow, and DeepLearning4J in terms of training and prediction speed and classification accuracy. They used MNIST dataset of handwritten digits for testing five FCN frameworks. Their computation experiments applied only on CPU and they reported out the results for two kinds of scaling factors applied on FCN networks including changing the network's *depth* (number of internal layers) and changing the network's *width* (number of neurons). Also, they tested the neural networks with two different activation functions Tanh and ReLU.

In previous studies, the comparison goal focused only on processing time, and none of those comparative studies dealt with CPU and GPU utilization or memory consumption. This work covered these metrics to find which of the considered frameworks achieve the best performance. Another issue that distinguishes this work from existing ones is that the experiments are conducted on two different machines; a regular laptop (equipped with NVIDIA GeForce GTX 960M GPU) and a very powerful server (equipped with NVIDIA Tesla M60 GPU).

III. Deep Learning Frameworks

The frameworks considered in this comparative study are: CNTK, TensorFlow and Theano. We also use Keras on top of these frameworks as discussed later. All of these frameworks provide flexible APIs and configuration options for performance optimization. Software versions of the frameworks are shown in Table I and their properties are shown in table II

A. CNTK

Microsoft Cognitive Toolkit (CNTK) is an Open source DL framework developed by Microsoft Research [11] for training

and testing many types of NN across multiple GPUs or servers. CNTK supports different DL architectures like Feedforward, Convolutional, Recurrent, LSTM, and Sequence-to-Sequence NN.

A Computational Network learns any function by converting it to a directed graph where each leaf node consists of an input value or learning parameter, and other nodes represent a matrix operations upon their children. In this case, CNTK has an advantage as it can automatically find the derive gradients for all the computation which are required to learn the parameters In CNTK, users specify their networks using a configuration file that contains information about the network type, where to find input data, and the way to optimize parameters [12].

CNTK interface supports different APIs of several languages such as Python, C++ and C# across both GPU (CUDA) or CPU execution. According to its developers [13], CNTK was written in C++ in an efficient way, where it removes duplicated computations in forward and backward passes, uses minimal memory needed and reduces memory reallocation by reusing them. The framework's installation on both laptop and server machines is mentioned in [14].

B. Theano

Theano is an commercial Python library developed at VUA lab at University of North Texas as a compiler for mathematical expressions that let users and developers optimize and evaluate their expressions using a NumPy's syntax (a Python library that supports a large and multi-dimensional arrays) [8], [15]. Theano starts performing computations automatically by op-timizing the selection of computations, translates them into other machine learning languages such as C++ or CUDA (for GPU) and then compiles them into Python modules in an efficient way on CPUs or GPUs.

Theano is being actively developed since 2008, which makes it more popular on a research and ecosystem platform than many DL libraries. Several software packages have been developed to build on top of Theano, with a higher-level user interface which aims to make Theano easier to express and train different architectures of deep learning models, such as Pylearn2, Lasagne, and Keras. The framework's installation on both laptop and server machines is mentioned in [14].

C. TensorFlow

TensorFlow is an open source framework developed by Google Brain Team [16], that uses a single data flow graph to express all numerical computations to achieve excellent performance. TensorFlow constructs large computation graphs where each node represents a mathematical operation, while the edges represent the communication between nodes. This data flow graph executes the communication between sub-computations explicitly, which makes it possible to execute independent computations in parallel or to use multiple devices to execute partition computations [17]. The framework's installation on both laptop and server machines is mentioned in [14].

Programmers of TensorFlow define a large computation graphs from basic operators, then distribute the execution of these graphs across a heterogeneous distributed system (can deploy computation to one or more CPUs or GPUs on a different hardware platforms such as desktops, servers, or even mobile devices). The flexible architecture of TensorFlow allows developers and users to experiment and train a wide variety of deep neural network models, and it is used for deploying machine learning systems into production for different fields including speech recognition, NLP, computer vision, robotics, and computational drug discovery. TensorFlow uses different APIs of several languages such as Python, C++, and Java for constructing and executing a graph (Python API is the most complete and the easiest to use) [18]. The framework's installation on both laptop and server machines is mentioned in [14].

D. Keras

Keras is an open source DL library developed in python. It runs on top of CNTK, Theano or TensorFlow frameworks. Keras was founded by Google engineer Chollet [19] in 2015 as a part of research project ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System). Keras is designed in a way that allows fast expression with deep neural networks and easy and fast prototyping (modularity and extensibility) [20]. The framework's installation on both laptop and server machines is mentioned in [14].

IV. Methods and Results

A. Methods

The goal of this experimental study is to compare the aforementioned frameworks (Theano, TensorFlow and CNTK) by using them to train CNN models on standard benchmark datasets of classical problems in image processing. We then evaluate each framework's performance through the following metrics:

- Running time
- Memory consumption
- CPU and GPU utilization
- Number of epochs

Despite resource limitations, we aim at comparing the aforementioned frameworks using different machines. Specifically, we employ a virtual machine (VM) running on a very powerful server in addition to a much cheaper laptop. Both machines are GPU-equipped; However, they vary greatly in their specifications. The VM has a Windows Server 2012 R2 operating system with the following specifications:

- Intel Xeon CPU ES-2690 v3 @2.60GHz (6 cores)
- 56 GB RAM
- 64-bit operating system, x64-based processor
- NVIDIA Tesla M60 graphics card, dual-slot 10.5 inch PCI Express Gen3 graphics card with two high-end NVIDIA Maxwell graphics processing units (GPUs). The Tesla M60 has 16 GB GDDR5 memory (8 GB per GPU) and a 300 W maximum power limit. It is designed

for single precision GPU compute tasks as well as to accelerate graphics in virtual remote workstation and virtual desktop environments. It has 4096 CUDA cores.

As for the laptop, it has Windows 10 operating system with the following specifications:

- Intel Core i7-6700HQ CPU @ 2.60GHz (4 cores)
- 16 GB RAM
- 64-bit operating system, x64-based processor
- NVIDIA GEFORCE GTX 960m graphics card (Laptop), with PCI Express 3.0 bus support, equipped with 4 GB GDDR5 memory and 640 CUDA cores.

It is worth mentioning that our goal is to compare the resources consumed by each framework to reach a certain accuracy level for each problem. So, we experimented with different epoch counts in order to make sure the accuracy for all frameworks are close to each other.

CNN is applied for different classical problems including: MNIST and CIFAR-10, where each dataset has a different network architecture. When applying CNN datasets on both Theano and Tensorflow, Keras is used for coding all of these datasets, while, in CNTK, Keras is not used.

For the MNIST and CIFAR-10 datasets, two convolutional layers with ReLU activation function are used after the input layer. The activation function is used to reduce the training time and to prevent vanishing gradients. After each CNN layer, a max-pooling layer is added in order to down-sample input and to reduce overfitting. In the max-pooling layer, the stride value must be specified with which the filter is slid. When the stride is one, the filter (window) is moved one pixel at a time. When the stride is two, the filters move two pixels at a time. This will produce smaller output volumes spatially. After each max-pooling layer, the dropout method is used in order to reduce overfitting by forcing the model to learn many independent representations of the same data through randomly disabling neurons in the learning phase. The sequential architecture (layers part only) used on the MNIST and CIFAR-10 datasets are shown in [14].

B. Benchmark Datasets

In this subsection, we discuss the datasets used in our experiments.

1) MNIST: The MNIST (Mixed National Institute of Standards and Technology) dataset (shown in Figure 3) is a computer vision database for handwritten digits. It is widely used for training and testing in the field of machine learning [21], [22].

MNIST has a training set of 60,000 images and a testing set of 10,000 images. It is a subset of a larger set available from NIST. Each image is 28×28 pixels which can be represented as a big array of numbers. We can flatten this array into a vector of $28 \times 28 = 784$ numbers. Each image in MNIST has a corresponding label, a number between 0 and 9 representing the digit appearing in the image. See Figure 1.

[1]https://goo.gl/Gm8xR7

Fig. 1. MNIST digits [1]

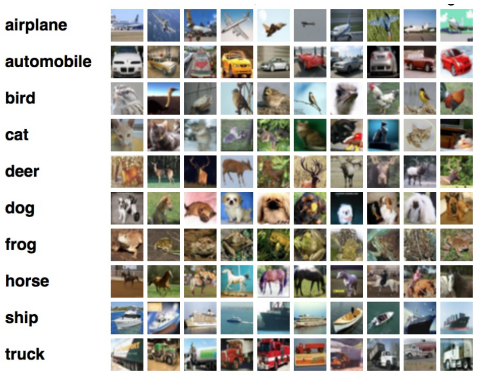

Fig. 2. CIFAR-10 dataset classes [2]

Our goal is to construct a CNN to classify MNIST images. The training will be carried out on both CPU and GPU environments using different frameworks including the aforementioned ones. We then evaluate the performance of each one framework.

2) CIFAR-10: The CIFAR-10 dataset is one of the 80 million images datasets, collected by Krizhevsky et al. [23], [24]. It consists of 60,000 32×32 color images evenly distributed over ten classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. There are 50,000 training images and 10,000 test images.

Figure 2 shows the classes in the dataset, as well as 10 random images from each class. The classes are completely mutually exclusive. I.e., there is no overlap between them. For instance, the "Automobile" class includes sedans, SUVs, etc. On the other hand, the "Truck" class includes only big trucks. To avoid overlap, neither one of these two classes includes pickup trucks.

V. Results

After implementing the experiments of each dataset on each framework the target was to find which of these frameworks had the best performance. Relating to table III, regarding image recognition datasets (MNIST and CIFAR-10) one can observe the superiority of CNTK over Tensorflow and Theano in terms of GPU and CPU multithreading, but in CIFAR-10 using 8,16 and 32 threads in CPU Tensorflow was faster than

[2]https://www.cs.toronto.edu/~kriz/cifar.html

TABLE III
PROCESSING TIME FOR EACH DATASET (MEASURED IN SECONDS)

Dataset	Environment	Threads	CNTK	TensorFlow	Theano
MNIST	CPU	1	847	5130	3560
	CPU	2	630	3180	2500
	CPU	4	574	2070	2260
	CPU	8	560	1740	2060
	CPU	16	567	1920	2050
	CPU	32	588	2010	2050
	GPU	-	66.67	328.93	377.86
CIFAR-10	CPU	1	20196	25905	26700
	CPU	2	14520	16610	18700
	CPU	4	13662	11550	17250
	CPU	8	11484	9955	15800
	CPU	16	11550	10340	15850
	CPU	32	11649	10835	15750
	GPU	-	926	2166.4	2386.1

CNTK. On the other hand Theano revealed to be more time consuming than be aforementioned frameworks.

VI. DISCUSSION

At this section, a visual representation of tables that appeared in results section is presented. Figures (3 – 5) include CPU multithreading and GPU processing time. The processing times clearly show the advantage of GPU over CPU for training deep convolutional and recurrent neural networks. The advantage of fast GPU would be more significant when training complex models with larger data. From the CPU results, the best performance occurred when the number of threads are equal to the number of physical CPU cores where each thread possesses a single core. In our work we used a laptop with 8 cores, so in each dataset the best performance in term of processing time was achieved while using 8 threads as shown in figures (3 – 5). [14] contains a performance metrics of CPU, GPU and memory utilization as well as number of epochs that needed to achieve a good accuracy for each dataset. The metrics measurement of each framework was conducted to help us explain the failure of one of the selected frameworks.

We noticed the poor performance of Theano at most datasets comparing to CNTK and TensorFlow, that could be explained because of low CPU utilization (as shown in appendix C) comparing to aforementioned frameworks, meanwhile CNTK and TensorFlow use all available resources (high CPU utilization). CNTK outperformed both TensorFlow and Theano while training MNIST and CIFAR-10 datasets where perhaps the achievement of this out performance happened because of its BrainScript format which is a custom network description language that makes CNTK more flexible for neural networks customization. On the other hand, TensorFlow uses Eigen, which is a C++ template library (BLAS library) for linear algebra including matrices, vectors, numerical solvers, and related algorithms, which is used to make TensorFlow perform better than CNTK and Theano in RNN.

Comparing our work to previous work such as Bahrampour et al. work presented at [3] and Shi et al. work presented at [4], we reveal the following findings.

Bahrampour et.al based their comparative study on three main aspects including speed, hardware utilization, and extensibility. Besides they used three neural network types such as CNNs, AE, and Recurrent LSTM to train MNIST, ImageNet, and IMDB datasets on Caffe, Neon, Tensorflow, Theano and Torch frameworks. They used the following hardware specs; intel xeon CPU E5-1650 v2 @3.5GHz (with multi-threading), Nvidia GeForce GTX Titan X/PCI/SSE2, 32GB DDR3 Memory, and a SSD drive to come up with the following results; while training on CPU, Torch performed the best followed by Theano, but Neon had the worst performance, also Theano and Torch are the best in extensibility term, as well as TensorFlow and Theano were very flexible and Caffe was the easiest to find the performance. Regarding training datasets on GPU, and for larger convolutional and fully connected networks, Torch was the best followed by Neon. For smaller networks Theano was the best. For LSTM, Theano results was the best in performance, while TensorFlow performance is not competitive compared to other studied frameworks.

On the other hand, Shi et al. based their comparative study on two comparative terms including processing time and convergence rate. The neural networks used are fully connected NN, CNNs and RNNs to train ImageNet, MNIST, and CIFAR10 datasets on Caffe, CNTK, MXNet, TensorFlow, and Torch frameworks. They used the following hardware specs; Two types of multi-threaded CPU: desktop CPU (intel i7-3820) (8 threads) and server-grade CPU (intel xeon E5-2630) (32 threads), as well as three types of Nvidia GPU (GTX980, GTX1080, Tesla k80), and Two tesla K80 cards used to evaluate the multi-GPU performance. The results of TensorFlow were the best while using CPU. While using single GPU; on FCNs, Caffe, CNTK and Torch performed better than MXNet and TensorFlow. As for small CNN; Caffe and CNTK achieved a good performance, and for RNN (LSTM), CNTK was the fastest (5-10x faster than other frameworks). Using Multi-GPU implementation, all frameworks had higher throughput and accelerated the convergence speed.

Our comparative Study was based on measuring performance for CNTK, TensorFlow, and Theano using CNNs to train MNIST and CIFAR-10 datasets on CNTK, TensorFlow, and Theano. The results were as follows: regarding image recognition datasets (MNIST and CIFAR-10) one can observe the superiority of CNTK over Tensorflow and Theano in terms of GPU and CPU multithreading, but in CIFAR-10 using 8,16 and 32 threads in CPU Tensorflow was faster than CNTK. On the other hand Theano revealed to be more time consuming than be aforementioned frameworks.

VII. CONCLUSIONS AND FUTURE WORK

The purpose of this work was to provide a qualitative and quantitative comparison between three of the most popular and most comprehensive DL frameworks (namely Microsoft's CNTK, Google's TensorFlow and University of Montreal's Theano). The main goal of this work was to help end users make an informed decision about the best DL framework that suits their needs and resources. To ensure that our study is

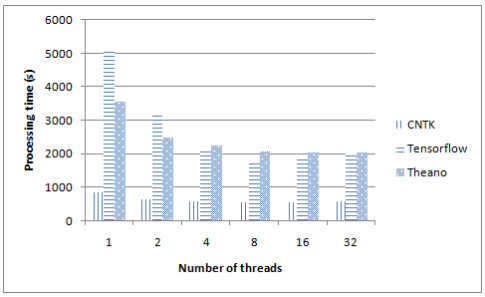

Fig. 3. CPU processing time for MNIST dataset

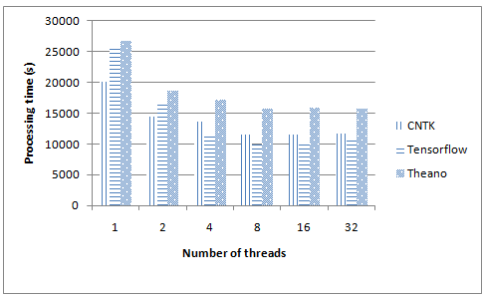

Fig. 4. CPU processing time for CIFAR-10 dataset

as comprehensive as possible, we used multiple benchmark datasets, namely MNIST and CIFAR-10, which were trained via multilayer CNN network architecture, and performed our experiments on two different machines (a regular laptop with windows 10 operating system installed within and a very powerful server with windows server 2012R2 operating system installed on it). For the two machines under consideration, we measured performance and utilization of CPU multithreading, GPU and memory.

After the evaluation of the datasets, on both laptop and server devices we observed the following. In all implementations, the processing time clearly shows the advantage of GPU over CPU for training networks. For GPU utilization metric in MNIST and CIFAR-10 datasets, TensorFlow had the lowest utilization, followed by Theano and CNTK. For CPU utilization metric, Theano had the lowest utilization followed by TensorFlow and CNTK. For Memory utilization while using CPU and GPU, the results were close to each other. In MNIST and CIFAR-10 datasets one can observe the

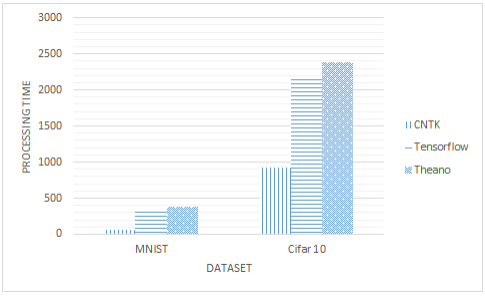

Fig. 5. GPU processing time for each dataset

superiority of CNTK over Tensorflow and Theano in terms of GPU and CPU multithreading, but in CIFAR-10 using 8, 16 and 32 threads in CPU Tensorflow was faster than CNTK. On the other hand Theano revealed to be more time consuming than be aforementioned frameworks.

REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] M. Al-Ayyoub *et al.*, "Deep learning for arabic nlp: A survey," *Journal of Computational Science*, 2017.

[3] S. Bahrampour, N. Ramakrishnan, L. Schott, and M. Shah, "Comparative study of deep learning software frameworks," *arXiv preprint arXiv:1511.06435*, 2015.

[4] S. Shi, Q. Wang, P. Xu, and X. Chu, "Benchmarking state-of-the-art deep learning software tools," *arXiv preprint arXiv:1608.07249*, 2016.

[5] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, 2011, pp. 142–150.

[6] P. Goldsborough, "A tour of tensorflow," *arXiv preprint arXiv:1610.01178*, 2016.

[7] S. Chintala, "Convnet benchmarks," https://github.com/soumith/convnet-benchmarks [Accessed: July 2017].

[8] R. Al-Rfou *et al.*, "Theano: A python framework for fast computation of mathematical expressions," *arXiv preprint arXiv:1605.02688*, 2016.

[9] 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.

[10] V. Kovalev, A. Kalinovsky, and S. Kovalev, "Deep learning with theano, torch, caffe, tensorflow, and deeplearning4j: Which one is the best in speed and accuracy?" in *Pattern Recognition and Information Processing (PRIP 2016)*, 2016.

[11] D. Yu *et al.*, "An introduction to computational networks and the computational network toolkit," *Microsoft Technical Report MSR-TR-2014–112*, 2014.

[12] D. Yu, K. Yao, and Y. Zhang, "The computational network toolkit [best of the web]," *IEEE Signal Processing Magazine*, vol. 32, no. 6, pp. 123–126, 2015.

[13] Microsoft, "The microsoft cognitive toolkit," https://docs.microsoft.com/en-us/cognitive-toolkit/cntk-evaluation-overview [Accessed: June 2017].

[14] G. Al-Bdour, "Comparative study between deep learning frameworks using multiple benchmark datasets," Master's thesis, Jordan University of Science and Technology, 2017.

[15] 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 compiler in python," in *Proc. 9th Python in Science Conf*, 2010, pp. 1–7.

[16] 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.

[17] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI). Savannah, Georgia, USA*, 2016.

[18] G. team, "Google," https://www.tensorflow.org/ [Accessed: June 2017].

[19] F. Chollet *et al.*, "Keras," 2015.

[20] F. Chollet, "Keras (2015)," *URL http://keras. io*, 2017.

[21] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[22] Y. Lecun, C. Cortes, and C. J. Burges, "The mnist database of handwritten digits, 2009," *URL http://yann. lecun. com/exdb/mnist*, 2009.

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[24] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," 2009.