COMPUTERS AND INFORMATICS
2757-8259 2021, Volume 1 Issue 1
Research Article

Performance comparison of deep learning frameworks

M. Mutlu Yapıcı
Ankara University, Computer Technologies Department, Ankara, 06100, Turkey
Nurettin Topaloğlu
Gazi University, Technology Faculty, Computer Engineering Department, Ankara, 06500, Turkey

Submitted: 14.07.2020
Accepted: 23.09.2020
Published: 28.02.2021

Abstract:

Deep learning (DL) is branch of machine learning and imitates the neural activity of brain on to artificial neural networks. Meanwhile it can be trained to define characteristics of data such as image, voice or different complex patterns. DL is capable of to find solutions for complex and NP-hard problems. In literature, there are many DL frameworks, libraries and tools to develop solutions. In this study, the most commonly used DL frameworks such as Torch, Theano, Caffe, Caffe2, MXNet, Keras, TensorFlow and Computational Network Tool Kit (CNTK) are investigated and performance comparison of the frameworks is provided. . In addition, the GPU performances have been tested for the best frameworks which have been determined according to the literature: TensorFlow, Keras (TensorFlow Backend), Theano, Keras (Theano Backend), Torch. The GPU performance comparison of these frameworks has been made by the experimental results obtained through MNIST and GPDS signature datasets. According to experimental results TensorFlow was detected best one, while other researches in the literature claimed that Pytorch is better. The contributions of in this study is to eliminate the contradiction in the literature by revealing the cause. In this way, it is aimed to assist the researchers in choosing the most appropriate DL framework for their studies.

Keywords: *Artificial neural network, Deep learning, Deep learning frameworks*

Cite this paper as: Yapıcı, M.M. & Topaloğlu, N., Performance comparison of deep learning frameworks, *Computers and Informatics*, 2021, *1*(1), 1-11.

1. INTRODUCTION

Technological developments have made a rapid increase in the amount of and variety of data used and produced. This production is occurred the big data. Big data can be grouped as structural, non-structural or semi-structural and it has five basic components as variety, velocity, volume, verification, and value. It includes volume, velocity, and variety of data. Data mining and artificial intelligence methods are used for processing and interpreting big data. Deep learning (DL) is a type of artificial intelligence with many layers used for the training of big data. DL is being used the state-of-the-art for tackling training problems that require processing big data like videos, images and signals. Big data is one of great importance in enhancement and use of DL frameworks. In Fig. 1 DL and big data relation is presented. As shown in Fig. 1 DL is used for processing big data like data mining and artificial intelligence.

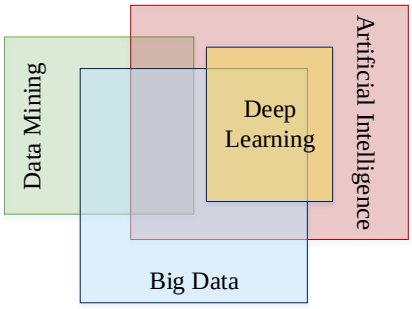

Figure 1. Artificial intelligence, big data and DL relation.

Additionally the hardware enhancements such as storage system, developments of the DL datasets that allow easily develop an efficient DL models are also quite common in widespread use of DL methods. Timeline showing the date of occurrence of the DL frameworks discussed in this study is given in Fig. 2. Therefore, in this study some of most widely used open source DL frameworks information briefly is provided. And highlighted the advantages and disadvantages of these frameworks by comparison with each other. These comparisons will be a very important guide for researchers studying on DL.

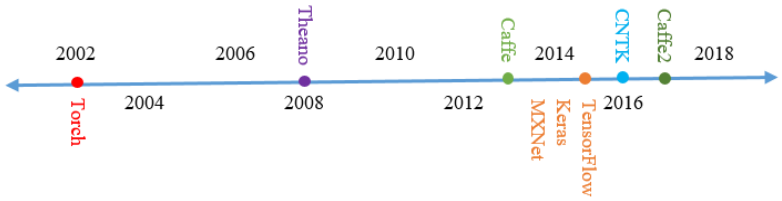

Figure 2. Timeline showing the date of occurrence of the DL frameworks discussed.

Our main motivation to carry out the study is to provide the performance information on the most used frameworks to guide researchers working in the field of deep learning. Especially, the performance results on GPU will be an important guide for researchers to choose the best frameworks. Therefore, in this study, detailed information about Torch, Theano, Caffe, Caffe2, MXNet, Keras, TensorFlow, Computational Network ToolKit (CNTK) frameworks are presented. An experimental evaluation study is performed by using MNIST and GPDS datasets for TensorFlow, Keras (TensorFlow Backend), Theano, Keras (Theano Backend), PyTorch frameworks, and performance comparisons of these frameworks are mentioned.

In this study, properties of the frameworks which are introduced in section 2 are compared according to literature studies. Comparison results in this study will greatly contribute to the selection of a suitable platform for researchers to work in the field of DL. First the comparison of properties of all these frameworks are given in Table 1.

Table 1. Properties of the frameworks. Feature comparison.

Software	Caffe[1-3]	Caffe2[4]	Keras[5], [6]	Microsoft Cognitive Toolkit[7-10]	MXNet[2], [11]	TensorFlow [2], [12]–[16]	Theano [2], [17-19]	Torch[1], [2], [20]
Creator	Berkeley Vision and Learning Center	BVLC, NVIDIA, Facebook	François Chollet	Microsoft Research	Apache Software Foundation	Google Brain team	Université de Montréal	Ronan Collobert, Koray Kavukcuoglu, Clement Farabet
Software license	BSD license	BSD license	MIT license	MIT license	Apache 2.0	Apache 2.0	BSD license	BSD license
Platform	Linux, macOS, Windows	Linux, macOS, Windows	Linux, macOS, Windows	Windows, Linux (macOS via Docker on roadmap)	Linux, macOS, Windows, AWS, Android, iOS, JavaScript	Linux, macOS, Windows, Android	Cross-platform	Linux, macOS, Windows, Android, iOS
Core	C++	C++	Python	C++	Small C++ core library	C++, Python, CUDA	Python	C, Lua
Interface	Python, MATLAB, C++	Python, MATLAB, C++	Python, R	Python (Keras), C++, Command line, BrainScript (.NET on roadmap)	C++, Python, Julia, Matlab, JavaScript, Go, R, Scala, Perl	Python (Keras), C/C++, Java, Go, R, Julia	Python (Keras)	Lua, LuaJIT, C, utility library for C++/OpenCL
Devices (Beyond CPU)	GPU	GPU, Mobile	GPU, Mobile	GPU	GPU, Mobile	GPU, Mobile	GPU	GPU/ FPGA
Programming Style	Declarative	Declarative	Declarative	Declarative	Imperative, Declarative	Declarative	Declarative	Imperative
Supported Architectures	RNN, CNN	RNN, CNN	RNN, CNN, RBM, DBN	RNN, CNN	RNN, CNN, RBM, DBN	RNN, CNN, RBM, DBN	RNN, CNN, RBM, DBN	RNN, CNN, RBM, DBN
Parallel execution (multi node)	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes

All frameworks in Table 1 support CUDA. Features of these frameworks are not very different, but they can be found superior in different aspects depending on the project and equipment. For performance comparison of these frameworks, the studies in the literature have been investigated and compared according to these studies. So, there are not all the frameworks in each comparison tables and graphs.

The Paper is organized as follows. In section 2 DL frameworks are presented. In section 3 datasets used in the study for comparison are mentioned. In section 4 experimental evaluation of DL frameworks are presented. In section 5 the study conclusion is mentioned.

2. DL FRAMEWORKS

DL frameworks in literature are Torch, Theano, Caffe, Caffe2, MXNet, Keras, TensorFlow and CNTK. These frameworks details are described below.

2.1. Caffe and Caffe2

Caffe is a framework developed in BSD-licensed C# and uses R and Java interfaces [1], [3]. Jia et al [3] claim that caffe can process over 40 million images a day on a single K40 or Titan GPU. In another study Chetlur et al [21] reported that integrating cuDNN into Caffe improves performance by 36% on a standard model while reducing memory usage [22]. Although Caffe has some advantages such working well with CNN architecture, providing good performance for feedforward networks and

allowing fine tuning of training models without writing code precisely, it has poor documentation, and prove bad and bulky performance in RNN architecture and large neural networks.

Caffe2 was developed by Yangqing Jia who is developer of Caffe. After Yangqing jia started working on Facebook they developed the Caffe2 framework based on caffe along with NVIDIA and Facebook. Caffe2 improves some limitation of Caffe such as large-scale distributed training support, mobile deployment, new hardware support quantized computation. Caffe2 has great support from NVIDIA and has native Python and C++ APIs. This allows it to quickly prototype and optimize for projects easily [22].

2.2. Keras

Keras is a fast growing Low-level Neural Network API developed in Java language to quickly and easily implement RAG applications. It capable of running TensorFlow, Theano or CNTK at backend [5]. The input data size assumed for each backend is different. Therefore, it is important to be careful when designing a system with keras. It is easy to find examples about the Keras coding. Keras has good documentation [23], [24].

2.3. Microsoft Computational Network ToolKit

Computational Network Toolkit that formerly knows as known as CNTK [7] is an open-source DL framework which was developed by Microsoft. CNTK has a Python API over C++ code and it has not adopted one of the more conventional licenses, such as ASF 2.0, BSD or MIT. CNTK uses a graphical training method unlike other methods [10].

CNTK supports C# and BrainScript. They provide both high-level and low-level APIs for ease of use and flexibility. In a study conducted in December 2015 [8], it was tested with a system using multiple GPUs in a fully connected 4-layer neural network. They reported CNTK performance is much better than Caffe, TensorFlow, Theano and Torch [22].

2.4. MXNet

MXNet is a multi-language ML library. İt combine symbolic expression with tensor process to maximize efficiency and flexibility. MXNet uses little memory spaces and has computationally efficiency. It runs on a variety of systems, from mobile devices to GPU systems [2]. MXNet provides optimized numerical computation via just a few lines of code in high-level languages such Python and R, for GPUs and distributed ecosystems. It supports two styles of programming: imperative and symbolic. Imperative programming uses the NDArray API and symbolic programming uses the Symbol API [11].

2.5. TensorFlow

TensorFlow is a DL and ML framework which was written with a Python API over a C/C++ engine by Google Brain Team. There are two main advantages: first-one is that it is continually supported by the developer team and second one is it works flexible via many architectures such as CNN, RNN [15], [22]. TensorFlow uses computational (data flow) graphs to perform operations such as computation and shared state [12], [16]. In 2016, Goldsborough [15] published a comprehensive article that introduces TensorFlow's basic computation paradigms, distributed execution model, programming interface, and accompanying visualization tools. Another advantage of TensorFlow is that the tensor processing units (TPUs) specially designed for DL and ML projects by Google. TPU is a type of processor designed for helping to achieve larger amounts of low-level processing simultaneously [12], [25]. In 2017, Jouppi et al [25] introduced TPUs and compared the performance of TPUs with contemporary CPUs and GPUs.

2.6. Theano

Theano is developed in the Python programming language by the Montreal Institute of Learning Algorithms (MILA). Many functional features of NumPy, Theano also offers GPU support and faster expression evaluation. It is a general mathematical tool, but it has been developed to facilitate research in DL. Theano is open source software, distributed under the BSD license [17], [26]. In 2012, Bergstra et al [18] introduced new features such Scan Op., R-operator for Hessian-Free optimization, lazy evaluation-CVM, parallelism on CPU.

Various software packages such as Pylearn2 [27], Blocks [28], Lasagne [29], and Keras [5] have been developed to improve the strengths of Theano so far. In 2016, Theano development team published a comprehensive article about the developments and talents of Theano [30]. Theano has been active and continuously developed since 2008, with a large number of superstructures built up and used in many modern machine learning models. However, Yoshua Bengio [31], one of Theono's developers, in 2017 announced the development in Theano would end.

2.7. Torch

Torch is an API written in Lua and it is a computational framework. Torch uses array or Tensor to perform many operations. These operations include indexing, slicing, cloning, resizing, storage sharing. Torch's Python version is PyTorch, developed in 2017 by Facebook and using dynamic graphics which lets process variable-length inputs and outputs. This feature has provided the PyTorch to spread rapidly and be accepted by academic circles [16], [20].

3. DATASETS

The datasets used for DL frameworks performance comparison in this study is detailed below. These datasets are MNIST and GPDS signature.

3.1. MNIST

The MNIST dataset of handwritten digits is a subset of a larger set available from NIST. It is composed of 70,000 handwritten digit images which have 28x28 pixel size. It has a training set of 60,000 examples, and a test set of 10,000 examples. But in this study, it was used 50,000 for training, 10,000 for validation and 10,000 for test.

3.2. GPDS signature

The GPDS dataset is acquired from “Instituto Universitario para el Desarrollo Tecnológico y la Innovación en Comunicaciones (IDeTIC)”. It contains 4 different signature datasets: GPDS960signature, 4NSigComp2010 Scenario 2, GPDS960GRAYsignature, GPDSsyntheticSignature. In the study GPDSsyntheticSignature [32] dataset was used for comparison. GPDSsyntheticSignature dataset consists of signatures of 4000 different individuals. Every individual has 24 genuine signatures and 30 forged signatures. Every signatures were signed with different pens. The signatures are in 600 dpi "jpg" format. In the study, 3530 genuine and forged signatures were obtained by using data augmentation methods over 54 signature which belongs a person.

4. EXPERIMENTAL EVALUATION OF DL FRAMEWORKS

In this study, performance comparison of the most widely used DL frameworks is performed. In order to guide the researchers, firstly the comparison of the DL frameworks in the literature are investigated and analyzed. After that, according to the results of literature analysis, the performance comparisons of the best DL frameworks are performed, and the experimental results are shared. According to the results of literature, TensorFlow, Theano, Keras and Torch frameworks have been chosen for performance comparison.

The performance comparison between the chosen frameworks are performed according to their batch time rates and epoch time rates on different batch numbers. Performance comparisons are performed on GPU with two different datasets: GPDS and MNIST. The work is supported by the NVIDIA company by graphic card donation. All experimental results are obtained on the NVIDIA TITAN XP graphics card which has 12 GB memory. The study is performed using python 3.6 on Cuda 9.0 and CuDNN 7.3.1 versions. We think that some different results obtained in the literature are due to different Cuda and Python versions. For this reason, we considered it appropriate to write the versions of the software. ResNet50 CNN model is used on all frameworks. The hyperparameters of the model are used as include_top=False, weights=None. Stochastic gradient descent (SGD) optimizer is used for training and parameters of the optimizer are chosen for fine tuning as lr=1e-4, decay=1e-4, momentum=0.9, and nesterov=True. The parameter numbers of the model vary according to the dataset. When the GPDS dataset is used, model has total 24,183,554 parameters and 23,929,730 trainable parameters, but when the MNIST dataset is used, it has total 23,610,122 parameters and 23,552,906 trainable parameters.

The first comparison results are obtained on the MNIST dataset of handwritten digits. The batch time and epoch time performances of the models on the GPU were compared using the MNIST dataset of handwritten digits. The results were obtained separately for 200, 100 and 50 batches for all frameworks. The experimental results are presented in Table 2.

Table 2. MINST Hand Written Digit Dataset DL Frameworks Comparison.

DL Frameworks	MNIST Hand Written Digit Dataset					
	200 Batch		100 Batch		50 Batch	
	Batch Time (ms)	Epoch Time (ms)	Batch Time (ms)	Epoch Time (ms)	Batch Time (ms)	Epoch Time (ms)
TensorFlow	45	13938	32	18695	26	29942
Keras (TensorFlow Backend)	59	16650	52	27544	46	47467
Theano	343	85387	204	107809	122	123236
Keras (Theano Backend)	314	81328	188	94118	115	121123
PyTorch	78	21680	70	37856	37	40136

Batch Time performances of all models on MNIST dataset are shown in Fig. 3.

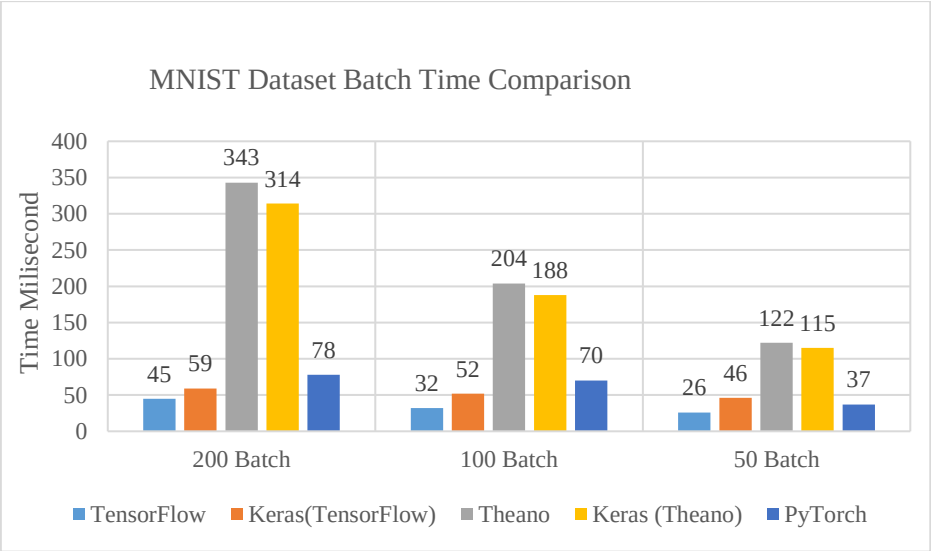

Figure 3. DL Frameworks Batch Time Comparison on MNIST Hand Written Dataset.

Epoch time performances are shown in Fig. 4.

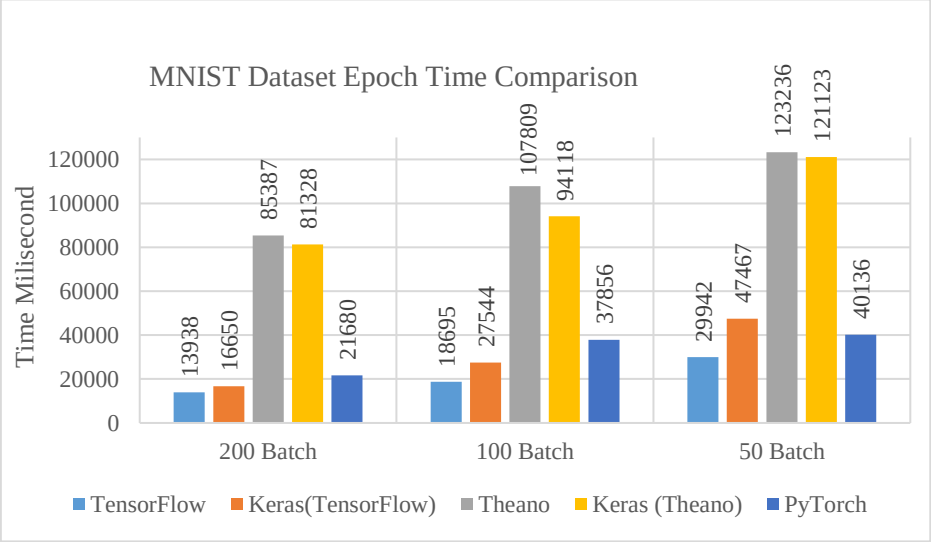

Figure 4. DL Frameworks Epoch Time Comparison on MNIST Hand Written Dataset.

According to the obtained results on MNIST dataset, it was revealed that the most efficient framework is TensorFlow on both performances of batch time and epoch time. TensorFlow has been the fastest framework for all batch sizes as 200, 100and 50 batch. Although the TensorFlow is slightly slower when worked on Keras backend, it has been still achieved the second-best performance with 200 batch and 100 batch. The biggest advantage of the Keras framework is to provide ease of use for all frameworks. Keras has been the easiest-to-use framework for both the installation phase and the model creation phase. The Pytorch framework showed the second-best performance for the 50 batch. It has been seen that the slowest epoch and batch time performances on MNIST dataset is obtained by Theano framework.

The second comparison results are obtained on the GPDS signature dataset. In the comparison of the framework, total 3530 signature images were used as 3177 for training and 353 for test. Before the training phase, all images were resized to 224x224 size and cleared to noises by applying pre-processing phase. As in the MNIST dataset, the batch time and epoch time performances of the models on the GPU were compared using the GPDSsyntheticSignature dataset. The results were obtained separately for 100,

70, 50 and 20 batches for all models. The same batch sizes were not be able to use because the size of images used in the GPDSsyntheticSignature dataset were bigger than the images used in the MNIST dataset. The experimental results are given in Table 3 and, Batch Time performances of all models are shown in Fig. 5 and Epoch time performances are shown in Fig. 4.

Table 3: GPDS Signature Dataset DL Frameworks Comparison.

DL Framwork	GPDS Signature Dataset							
	100 Batch		70 Batch		50 Batch		20 Batch	
	Batch Time (ms)	Epoch Time (ms)	Batch Time (ms)	Epoch Time (ms)	Batch Time (ms)	Epoch Time (ms)	Batch Time (ms)	Epoch Time (ms)
TensorFlow	488	15522	421	16144	249	16072	112	19956
Keras (TensorFlow Backend)	CEOOMEM*		433	20455	315	21330	142	23149
Theano	CEOOMEM*		1188	53909	862	55117	403	64438
Keras (Theano Backend)	CEOOMEM*		CEOOMEM* (For 67 Batch) 858	41878	662	43671	299	49306
PyTorch	438	14360	291	13857	224	14465	85	14117

*Cuda_Error_Out_of_Memory

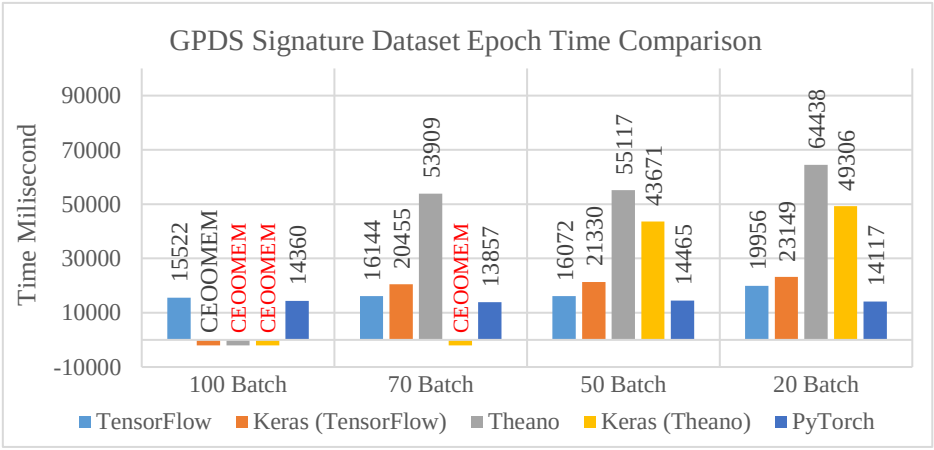

Figure 5. DL Frameworks Epoch Time Comparison on GPDS Signature Dataset.

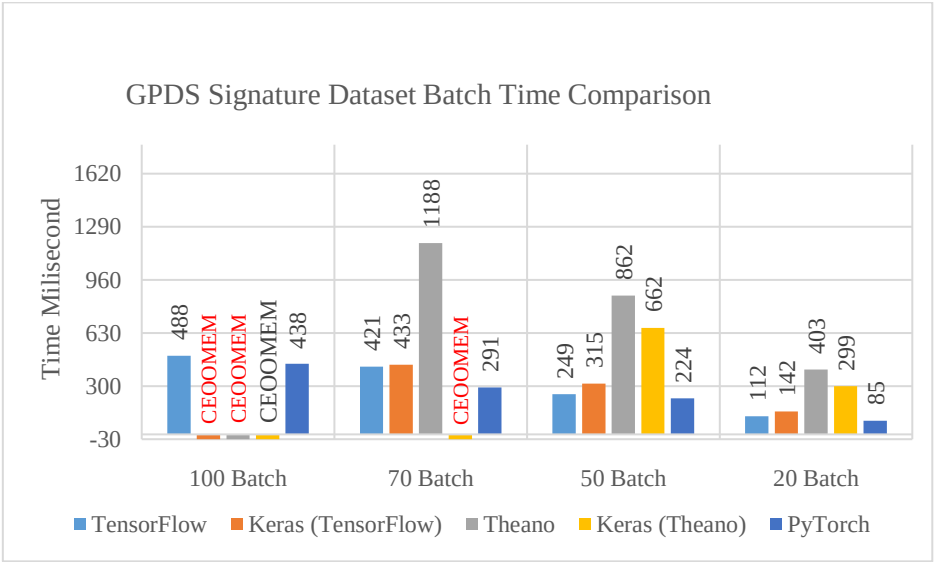

Figure 6. DL Frameworks Epoch Time Comparison on GPDS Signature Dataset.

We would like to give an analysis of the results in the literature before explaining the results of our studies. It is clearly seen from all comparative studies that Torch consistently performs very well memory management for both CPU and GPU. Bahrampour et al [1] did not use TensorFlow in the RNN comparison because of it does not accept variable length inputs within a batch. However, Shi et al [22] used TensorFlow and they reported that it performed much better than Torch on RNNs. Al-Rfou et al [19] also reported the same as Shi et al [22] that, TensorFlow performed better than Theano and Torch with RNN architecture on GPU. But they also reported that Torch had the best performance with CNN architecture on GPU. As noticed, Bahrampour et al [1] had claimed Theano had better performance than TensorFlow on CPU and GPU, while both Shatnawi, Al-Rfou and their teams had claimed TensorFlow had better performance [30], [33]. These different opinions of the researchers are usage of different cuDNN versions to test TensorFlow. Acording to the result of studies in literature TensorFlow have better results together with the starting to support cuDNN v6. Bahrampour et al [1] and Al-Rfou et al [30] obtain parallel results for Theano on GPU. They reported that Theano had better performance than Torch for RNN (LSTM) architecture on GPU.

The results obtained in the GPDSsyntheticSignature dataset have a great similarity with the results obtained from the MNIST dataset. However, we found that there is a very important difference between the two datasets. This important difference is that the best performing frameworks are different. We have seen that the best batch and epoch time performance on the GPDSsyntheticSignature dataset was not belong to TensorFlow but belongs to Pytorch framework. The second-best performance was achieved by the TensorFlow framework. These different results are also seen in previous studies in the literature. Shi et al [22] and Shatnawi [33] had claimed TensorFlow had better performance than Torch on GPU, while Al-Rfou et al [30] claimed Torch had the best performance. One of the most important aims of this study is to reveal the cause of this contradiction in the literature. This significant performance difference is thought to be the result of data size. The experimental results obtained from different sizes of datasets support this view. When the image size was increased, the Batch and Epoch time performance of the Pytorch framework was better than TensorFlow. When data size increases, TensorFlow is insufficient in memory management. It was seen that Pytorch has the best memory management and Theano had the worst memory management. Theano while works on Keras backend gave error of Cuda_Error_Of_Out_Memory (CEOOMEM) even at a size of 70 Batch on the GPDSsyntheticSignature dataset. In the tests that performed on the GPDSsyntheticSignature dataset, it was seen that Keras and Theano frameworks gave Cuda_Error_Of_Out_Memory for 100 batch. Theano was the slowest framework also on GPDSsyntheticSignature dataset, as in the MNIST dataset.

5. CONCLUSION

In literature there are some contradictions in the studies. Some researchers claimed that TensorFlow is the best, while others claimed that Torch is better. One of the significant contributions in this study is to eliminate this contradiction in the literature by revealing the cause. The experimental results showed that Pytorch has better memory management than TensorFlow. For this reason, TensorFlow performs better in small image sizes, while it is getting to slow down when image size increases. Pytorch has performed the best success on large-sized images by good memory management. All these results have been showed that different results obtained in the literature are not only dependent on software versions, but also on the memory management performance of the frameworks.

According to the results obtained in this study, the fastest framework for small size images is TensorFlow, while the fastest is Pytorch in large-size images. Keras is the most useful framework. Theano has the worst performance in terms of both speed and memory management.

Acknowledgment

This work has been supported by the NVIDIA Corporation. All experimental studies were carried out on the TITAN XP graphics card donated by NVIDIA. We sincerely thank NVIDIA Corporation for their supports.

REFERENCES

[1] Bahrampour, S., Ramakrishnan, N., Schott, L., & Shah M., Comparative study of caffe, neon, theano, and torch for deep learning 2016, a*rXiv, abs/1511.06435*.
[2] Chen T., et al. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems, *arXiv Prepr* 2015. *arXiv1512.01274*.
[3] Jia Y. et al. Caffe: Convolutional architecture for fast feature embedding. *In Proceedings of the 22nd ACM international conference on Multimedia* 2014, 675–678.
[4] NVIDIA. *Caffe2 Deep Learning Framework 2017*. https://developer.nvidia.com/caffe2.
[5] Chollet F., et al. Keras: Deep learning library for theano and tensorflow 2015. URL https//keras. io/k, *7*, 8.
[6] F.ois Chollet, *Keras 2015*, https://github.com/fchollet/keras.
[7] Microsoft. *Computational Network Toolkit (CNTK) 2016*. [Online]. Available: https://www.microsoft.com/en-us/cognitive-toolkit/.
[8] Huang. X., *Microsoft computational network toolkit offers most efficient distributed deep learning computational performance 2015*. https://goo.gl/9UUwVn.
[9] Microsoft. *The Microsoft Cognitive Toolkit 2016*. https://www.microsoft.com/en-us/cognitive-toolkit/.
[10] Banerjee, D.S., Hamidouche, K., & Panda, D.K., Re-Designing CNTK Deep Learning Framework on Modern GPU Enabled Clusters, *2016 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, 2016, pp. 144–151, DOI: 10.1109/CloudCom.2016.0036.
[11] MXNet. *MXNet 2017*. https://mxnet.incubator.apache.org/.
[12] Google, *TensorFlow*. [Online]. Available: https://www.tensorflow.org/.
[13] NVIDIA. *GPU-Accelerated TensorFlow 2018*. https://www.nvidia.com/en-us/data-center/gpu-accelerated-applications/tensorflow/.
[14] Abadi M., et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems 2016. *arXiv Prepr*. arXiv1603.04467, 2016.
[15] Goldsborough. P., A tour of TensorFlow 2016. *arXiv Prepr*. arXiv1610.01178.
[16] Comparing Top Deep Learning Frameworks. https://deeplearning4j.org/compare-dl4j-tensorflow-pytorch.
[17] Bergstra J., et al. Theano: A CPU and GPU math compiler in Python 2010. *In Proc. 9th Python in Science Conf, vol. 1*.
[18] Bastien F. et al. Theano: new features and speed improvements 2012. *arXiv Prepr*. arXiv1211.5590.
[19] Al-Rfou R., et al. Theano: A Python framework for fast computation of mathematical expressions. *arXiv Prepr*. arXiv1605.02688, 472: 473-2016.
[20] Torch. *What is Torch?* http://torch.ch/. [Online]. Available: http://torch.ch/.
[21] Chetlur S., et al. Cudnn: Efficient primitives for deep learning 2014. *arXiv Prepr*. arXiv1410.0759.
[22] Shi, S., Wang, Q., Xu, P., & Chu, X., Benchmarking state-of-the-art deep learning software tools 2016. In 7th *International Conference on Cloud Computing and Big Data (CCBD)*, 2016, 99–104. DOI: 10.1109/CCBD.2016.029.
[23] Chollet. F. *Deep learning with python,* 2017 Manning Publications Co.
[24] Erickson, B. J., Korfiatis, P., Akkus, Z., Kline, T., & Philbrick, K. Toolkits and libraries for deep learning. *J. Digit. Imaging*, 2017, *30*(4), 400–405.
[25] Jouppi N. P. et al. In-datacenter performance analysis of a tensor processing unit. *In Proceedings of the 44th Annual International Symposium on Computer Architecture*, 2017, 1–12.
[26] Bergstra J., et al. Theano: Deep learning on gpus with Python 2011. In NIPS 2011, BigLearning Workshop, Granada, Spain, vol. 3.
[27] Goodfellow I. J., et al. Pylearn2: a machine learning research library 2013. *arXiv Prepr*. arXiv1308.4214.
[28] Van Merriënboer B., et al. Blocks and fuel: Frameworks for deep learning 2015. *arXiv Prepr*. arXiv1506.00619.
[29] Sander Dieleman, Jan Schlüter, Colin Raffel, Eben Olson, Søren Kaae Sønderby, Daniel Nouri, Daniel Maturana, Martin Thoma, Eric Battenberg, Jack Kelly, Jeffrey De Fauw, Michael Heilman, diogo149, Brian McFee, Hendrik Weideman, takacsg84, peterderivaz, Jon, instagibbs, Dr. Kashif Rasul, CongLiu, Britefury, and Jonas Degrave, Lasagne: First Release. (Zenodo, 2015).

[30] Team T. T. D., et al. Theano: A Python framework for fast computation of mathematical expressions 2016. *arXiv Prepr*. arXiv1605.02688.
[31] Bengio. Y., MILA and the future of Theano 2017. [Online]. Available: goo.gl/gdmTjk.
[32] Ferrer, M. A., Diaz-Cabrera, M., & Morales, A. Synthetic off-line signature image generation 2013. *In Biometrics (ICB), 2013 International Conference*, 1–7.
[33] Shatnawi, A., Al-Bdour, G., Al-Qurran, R., & Al-Ayyoub, M. A comparative study of open source deep learning frameworks 2018. *In 2018 9th International Conference on Information and Communication Systems (ICICS)*, 72-77. DOI: 10.1109/IACS.2018.8355444.