Learning and Nonlinear Models - Journal of the Brazilian Society on Computational Intelligence (SBIC), Vol. 15, Iss. 1, pp. 45-52, 2017

QUALITATIVE ANALYSIS OF DEEP LEARNING FRAMEWORKS

Matheus Gutoski, Leandro T. Hattori, Nelson M.R. Aquino,
Helen C.M. Senefonte, Manassés Ribeiro, André E. Lazzaretti, Heitor S. Lopes
Bioinformatics and Computational Intelligence Laboratory – LABIC
Federal University of Technology Paraná – UTFPR
Av. 7 de setembro, 3165, 80230-901 Curitiba (PR), Brazil
hslopes@utfpr.edu.br

Abstract – Deep learning methods are becoming more popular for complex pattern recognition applications. As result, many frameworks have appeared aiming to facilitate the development of such applications. However, choosing a suitable framework may not be an easy task for new users. In this paper, a qualitative evaluation of four of the most popular Deep Learning frameworks is provided, including: Caffe, Torch, Lasagne and TensorFlow. A printed character recognition task was used as case study, and a Convolutional Neural Network was implemented for this purpose. The analysis focus on issues that are important for the development process and encompasses nine qualitative dimensions, showing the strengths and weaknesses of each framework. It is expected that this analysis can be useful for guiding new users in the area.

Keywords – Deep Learning, Caffe, Torch, Lasagne, TensorFlow, Convolutional Neural Networks.

1. INTRODUCTION

Many applications of Computer Vision (CV) have emerged recently. The core issue in those applications is the semantic understanding of the image contents in a way that enables complex tasks, such as object detection and localization, image classification, and scene understanding. One of the key issues of CV is how an artificial vision system can automatically learn appropriate internal representations. This issue has been a topic of great interest and considerable progress has been achieved with the development of Deep Learning (DL) methods [1].

The basic idea of DL is the mapping of complex input data throughout several layers of increasing abstraction and decreasing complexity [2]. The major advantage of this method is its ability to automatically learn a hierarchy of features from raw massive data.

Successful applications of DL are shown in [3] and [4], for different kind of CV problems. Also, DL turns out to be very effective for anomaly detection in videos [5] and [6]. In recent years, the applicability of DL methods has widened, offering efficient tools for areas other than CV. For instance, [7] and [8] presented a review of DL applications in Bioinformatics.

DL methods are naturally more complex than conventional neural networks. Therefore, many frameworks appeared recently aiming at popularizing the use of those methods. As a consequence, nowadays one can find a wide range of frameworks with different features. However, there is no general guide to help (new) users to evaluate and choose a specific framework.

In the recent literature, few works addressed the issue of comparing frameworks according to diverse points of view. For instance, [9] presented a survey of DL methods and frameworks (Caffe, Torch and Theano) for image classification problems. Besides, [10] presented a new DL software system called SINGA, and compared it with other frameworks, Caffe, MxNet, Torch and TensorFlow, from the point of view of scalability in multiple GPUs. Also, a comparative study of the state-of-the-art DL frameworks (Caffe, TensorFlow and Torch) can be found in [11]. In this study the authors focused in the quantitative analysis of hardware benchmarks and the performance of the frameworks. Also, [12] evaluated quantitative performance features of Caffe, Neon, TensorFlow, Theano and Torch.

Possibly the most known and widely used DL method is the Convolutional Neural Network (CNN), which achieved state-of-the-art performance for object recognition in CV [1]. A possible reason for such a high performance is that they can learn features automatically with superior discriminatory power for image representation, when compared to hand-crafted image descriptors, as pointed out by [13] and [5]. CNNs are mainly composed by two parts: a feature extractor and a classifier. An important issue about CNN is that both parts are trained at the same time, suiting to specific problems without the loss of generality, thus leading to outstanding classification performances. CNNs are widely used and they are consolidated as DL methods for CV [10, 13–16].

In the above-mentioned works, authors focused on the quantitative performance of the frameworks, without providing any qualitative review that enable an interested user to make an informed choice among the available frameworks. This paper presents a qualitative analysis of the most widely used frameworks for DL. This analysis is aimed at providing subsides for potential users to choose a framework for their specific applications. This qualitative review of frameworks is based on the experience acquired with a DL case study implementation. Therefore, four of the top state-of-the-art DL frameworks were used to implement a CNN for handwritten digit recognition, a classic image classification problem. Based on the assumption that the previous expertise concerning the usage of DL frameworks can be useful for beginner users, we believe that this qualitative analysis can help potential developers to choose the most appropriate framework for a given application. The qualitative study presented

Tabela 1: Main frameworks for developing Deep Learning models. (✓) Yes, (✗) No, (✓*) Third Party

Features		Frameworks			
		Caffe	Torch	Lasagne	TensorFlow
Other OS	Android	✓	✓	✗	✓
	iOS	✓	✓	✗	✓
Other Hardware	Raspberry PI	✓*	✓*	✗	✓
	FPGA	✓*	✓	✗	✗
OpenCL Implementation		✓	✓*	✗	✓*
Core Language		C/C++	Lua	Python	C
Bindings		Protobuf/C++/Python/Matlab	Lua/Python	Python	Python/C++/Java/Go
User license		BSD License	BSD License	BSD License	Apache 2.0
Programing Style		Prototxt	Symbolic Expression	Symbolic Expression	Symbolic Expression
Supported Input Formats		LMDB/LevelDB/HDF5	Torch Tensors	NumPy Arrays	NumPy Arrays

in this work focuses on: documentation, community support, pre-trained models, user learning curve, model definition, device management, extensibility, model visualization and model deployment. Also, it is supposed that minimal technical aspects need to be considered to choose an appropriate framework for a given application. Considering that all frameworks are in constant development, this work is a snapshot of the current versions of state-of-the-art frameworks.

In short, the main contributions of this work are:

1. To present qualitative evaluation of DL frameworks considering several technical aspects that are relevant for the developer;
2. To help users to make an informed choice between DL frameworks;
3. To share implementation expertise with the DL frameworks, applied to a real image classification problem.
4. To make available the source codes to the community (via GitHub) so as to foster potential users to start their own applications.

This paper is organized as follows. Section 2 presents a brief introduction of the main frameworks analyzed in this work. Section 3 details the case study and evaluation methods proposed. Section 4 presents the evaluation of the qualitative issues according to the expertise developed in the implementation. Finally, Section 5 presents the general conclusions and future research directions.

2 DEEP LEARNING FRAMEWORKS

DL maps complex input data throughout several layers of increasing abstraction and decreasing complexity. DLs have advantages over others approaches because they are capable of learning a hierarchy of features directly from the input raw data [2, 17]. Currently, there are many DL frameworks available, developed by universities or companies (see Table 1. In this work four frameworks were chosen to be evaluated: Caffe, Torch, Lasagne (Theano) and TensorFlow, because they are the most widely used nowadays. This choice was based, also, on the recent related works mentioned before in the Section 1.

All the frameworks evaluated use the Compute Unified Device Architecture library (CUDA), which is an extension of the C programming language developed by Nvidia Corporation[1] that allows the use of parallel computation using General-Purpose Graphics Processing Unity (GP-GPU). The cuDNN library (CUDA for Deep Neural Networks), which is a highly optimized version of CUDA for DL, is also commonly used as to interface with modern GPU boards.

2.1 CAFFE

Caffe[2] was developed by the Berkeley Vision and Learning Center (BVLC) [18]. It simplifies the definition of a model by using the Protobuf format, which is a Google buffer protocol that allows the serialization of structured data. The framework is an open source project under the 2-clause BSD license. It is available for the three most common operational systems: Windows, OS X and Linux. The core of the framework was built with the C/C++ programming language, and interfaces are available for C++, Python and MATLAB.

Originally, Caffe supported LevelDB data as input to the network. However, other types of data are now supported due to the contribution of the community, for instance, Lightning Memory-Mapped Database (LMDB) and Hierarchical Data Format (HDF5). LMDB is a data format that uses memory-mapped files for more efficient input/output operations. HDF5 is a data model maintained by HDF Group which allows a more versatile data representation. The framework also provides a tool to create LMDB from raw image files.

Recently, Caffe was used to develop DL applications for feature representation [19], object recognition [10], and classification using novel architectures [14].

[1] http://www.nvidia.com
[2] http://caffe.berkeleyvision.org/

2.2 TORCH

Torch[3] was created by means of a partnership between the New York University and the IDIA Research Institute [20], with support from important enterprises, such as Google, Facebook and Twitter. It is focused on scientific computing and DL applications. Torch provides packages including signal processing, machine learning, CV and others. The framework is an open source library under the BSD license. The operational systems supported are OS X, Ubuntu, iOS, Android and some embedded devices. The Torch core is implemented in C programming language, and its interface language is Lua. The data format used by Torch is the tensor (a multi-dimensional matrix), which is a class that allows to handle numeric data.

Torch was used as DL framework for applications that include, for instance, soft biometrics detection [13] and text recognition [16].

2.3 LASAGNE

The Lasagne[4] [21] project provides a friendly interface to build DL models (and regular neural networks) based on the Python Theano [22] library. The framework was started in 2014 and, currently, it is supported by a core team and contributors on GitHub. Theano is a library designed to deal efficiently with mathematical expressions using multi-dimensional arrays. Lasagne only provides a high-level interface to implement DL models. Moreover, background knowledge about Theano is required to use the framework.

Lasagne is open source under the BSD license, and its core language is Python, whilst the Theano was written in C programming language. Since Lasagne uses the Python as interface language, Numpy arrays can be directly loaded into Theano tensors.

Some recent works used Lasagne for character recognition [23], feature representation [15] and face analysis [24].

2.4 TENSORFLOW

TensorFlow[5] was developed by the Google Brain Team [25]. It is used for numerical computations, machine learning and DL applications. The framework is based on the use of data flow graphs to perform mathematical operations. Similar to the previously mentioned frameworks, TensorFlow is also an open source project, and it is available under the Apache 2.0 license. Both core and user interface languages are C++/Python.

The data input can be done in two ways. The first and simplest is by reading NumPy arrays. The other way is by reading the data from files, such as comma separated values (CSV) or Protobuf. A very important feature provided by TensorFlow is the TensorBoard software. It allows the user to visualize the data structure, the model results such as accuracy and loss, and a graph with the operations performed during the training of the model.

Recent applications based on TensorFlow include unsupervised DL [26], DL for scene understanding [27] and activity recognition [28].

3 METHODS

In order to evaluate the frameworks, a classical printed character recognition problem as used as case study. The commonest DL model is the CNN, and it was used to classify instances of the notMNIST dataset[6]. This was done using a CNN model to learn both features and classifier, for the notMNIST dataset. The dataset consists of 28×28 pixels gray scale images of printed characters of different styles. The training set contains 200,000 images divided into 10 different balanced classes. The test and validation sets contain 10,000 images each.

The CNN architecture used here was based on a recent work presented by [29], as it has been shown to be efficient for classifying the notMNIST dataset. The CNN architecture is shown in Figure 1, where each box represents a layer. Yellow boxes are the convolutional layers, blue boxes represent the max pooling layers, the gray box represents the dropout layer and the red boxes represent the fully connected (dense) layers.

The model was trained and tested using two GPU boards: Nvidia GTX 750-Ti and Nvidia Titan-X Pascal, running under Ubuntu 14.04.3 LTS. The framework versions used in this work were: Caffe 1.0.0-rc3, Torch 7.0, Lasagne 0.10 with Theano 0.8.2 and TensorFlow 0.10.

Based on the experience acquired by developing the same application in different frameworks, we evaluated several qualitative features, explained below:

1. Documentation: comprises the conciseness and richness of details of the documents available for the installation and first steps to develop a DL application;

2. Community support: the impact of the community in the evolution of the framework in terms of functionalities and support mainly in GitHub, GoogleGroups, and StackOverflow;

[3] http://torch.ch/
[4] http://lasagne.readthedocs.io/en/latest/index.html
[5] https://www.tensorflow.org/
[6] http://yaroslavvb.blogspot.com.br/2011/09/notmnist-dataset.html

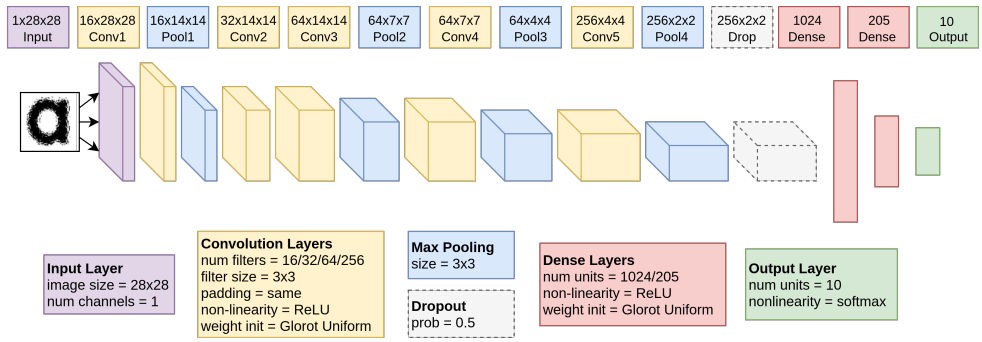

Figura 1: Architecture of the Convolutional Neural Network, trained and tested with all of the frameworks compared.

3. Pre-trained models: the availability of pre-trained models, so that DL applications can be easily accomplished by anyone without the burden of the hard computational effort required to train these networks;

4. Ease of learning: the previous knowledge requirements and the effort needed to learn the basic functionalities of each framework;

5. Ease of use: the effort needed to create or modify existing models for specific applications;

6. Device management: the capability of switching between CPU and GPU boards, choosing between multiple GPU cards and using all available GPU cards at the same time;

7. Extensibility: the feature of the framework to incorporate new functions or having existing ones modified in an efficient manner;

8. Visualization: the tools offered by each framework to visualize the model architecture and its operations. Visualizing the architecture can be very helpful to the developer to better understand the structure of DL models;

9. Deployment: the deployment process of each framework and the effort needed to perform this operation. Deployment allows DL models to be used in real-world scenarios.

4 RESULTS

4.1 DOCUMENTATION

There are tutorials for installing Caffe for the major operating systems, as mentioned in Section 2.1, and they are concise and understandable. All the requirements and dependencies along with the installation steps can be found in the tutorial. Caffe also allows to use a Docker installation that contains itself all the dependencies. The official website provides a tutorial to build a CNN step by step. Besides CNN, the distribution also comes examples of Multi-Layer Perceptron (MLP) and AutoEncoder (AE). The framework documentation shows the layers and functions available. However, such documentation was generated from the code and lacks detailed information.

Torch offers, in the official web page, installation tutorials for Ubuntu and Mac OS X. The installation of the framework and all its dependencies require only a few command lines, making the procedure simple and transparent to the user. Moreover, installation tutorials for Windows, Android and IOS are also available in the Community Wiki[7]. The Wiki offers several methods to install Torch for each of the operational systems. The framework offers video tutorials for new users, which include a step by step explanation of how to build CNNs and Recurrent Neural Networks (RNNs). The tutorials also present the basic principles of neural networks and DL in general. Besides, the Community Wiki presents a fairly complete documentation covering all of the packages available for Torch, including instructions to create new packages. However, the documentation does not follow a standard structure. Therefore, some sections of the documentation are more detailed than others.

Lasagne website provides the installation steps for Windows 7, Linux and Mac OS X systems. The requirements are documented in the tutorials, with Theano as the main dependency. The website also contains a Docker installation. The installation process is clear and straightforward. The documentation presents guides to build and run MLPs and CNNs. The official distribution comes with RNN and AE examples. The documentation is user-friendly and well structured, describing the functionalities and providing examples.

There are guides for installing TensorFlow in Windows, Linux and Mac OS X. The tutorials cover several different methods to install the framework, including virtual ambients and Docker installation. The installation steps are well documented and

[7] https://github.com/torch/torch7/wiki/Cheatsheet

straightforward. In the official website there are tutorials with step by step instructions to build MLPs, RNNs and CNNs, as well as AEs examples. The documentation covers the functionalities in Python, C++, Java and Go. The Python documentation is vast and well structured.

4.2 COMMUNITY SUPPORT

Caffe is a framework with strong community engagement in its development. The community also contributes with tutorials and educational material. These contributions allow the continuous growth of the framework and its features.

Torch is also evolving due to the community participation, which is of great importance to its development. The result of this strong presence is the Community Wiki, which provides a substantial amount of tutorials covering a wide range of applications. The tutorials are available not only for users, but also for developers.

Lasagne encourages the community to participate in the project by expanding and improving its functionalities. The expanded functionalities can be found in the Lasagne Wiki[8]. Other guides and tutorials can be found outside of the Wiki.

The TensorFlow community contributes to the development of the framework and provides materials in the form of guides and tutorials. The TensorFlow community in GitHub is the most active among the frameworks analyzed here.

4.3 PRE-TRAINED MODELS

Caffe provides a database with trained network models in the *Model Zoo*[9]. This database allows users to share large amount of trained models that can run in any Caffe installation. Among the frameworks analyzed, Caffe is the one with the largest number and variety of pre-trained models available.

For Torch, pre-trained models can be found in the Model Zoo available in the Community Wiki, which is similar to that of Caffe. Torch can also import Caffe models by using the LoadCaffe module, thus increasing the amount of pre-trained models.

Lasagne pre-trained models are available in the official distribution. However, it is not comparable to the Model Zoos available for Torch and Caffe, in terms of quantity. The framework also provides a Caffe to Lasagne model converter.

TensorFlow also provides pre-trained models of deep networks. However, similarly to Lasagne and Torch, it is significantly smaller than Caffe's Model Zoo. On the other hand, it is possible to convert Caffe models to TensorFlow format through community-provided methods.

4.4 EASE OF LEARNING

Learning Caffe is straightforward and intuitive for new users. It does not require any programming in order to build and run a DL application. Starting the training or testing process can be done by simply running a command line.

Torch may be considered unattractive at first, mainly because of the need to learn the Lua language. After learning the principles of Lua, the interaction with the framework becomes easier. A Torch wrapper is under development using the Python language[10] so as to provide more interfaces and facilitate new users to use the framework.

Since Lasagne depends upon Theano, it inherits some of its requirements, such as knowledge about symbolic expressions and tensors. Whilst Theano is a mathematical library, Lasagne is focused on DL applications. This allows Lasagne to abstract many of the Theano functions, making it more intuitive.

In order to use TensorFlow, the user is required to learn about its tensors and symbolic expressions. Similarly to Theano, it is a framework for numeric computations, providing flexibility for the development of applications. As a consequence, the user may experience some difficulty to learn the framework.

4.5 EASE OF USE

The definition the network architecture in Caffe is done by means of a Protobuf file, which is similar to XML. Therefore, the use of Caffe becomes intuitive and does not require much previous knowledge.

Defining the model architecture in Torch is done sequentially, layer by layer. The type of the layer is defined by which function was called. For instance, the function which defines a convolutional layer has a different name and expects different arguments from that which defines a pooling layer. After the user becomes acquainted with the functions, defining and modifying networks become rather easy.

The way a DL model is constructed and modified in Lasagne, as well as in TensorFlow, resembles Torch. The network is also defined sequentially by means of function calls, in which each type of layer has a specific function.

4.6 DEVICE MANAGEMENT

In Caffe, switching between CPU and GPU is quite easy. If a server has multiple GPUs, Caffe also allows the selection of a specific one to be used. The framework is capable of managing the distribution of tasks into multiple GPUs at the same time. All the GPU related configurations can be done by simply passing arguments.

[8] https://github.com/Lasagne/Lasagne/wiki/From-Zero-to-Lasagne
[9] https://github.com/BVLC/caffe/wiki/Model-Zoo
[10] http://pytorch.org/

In Torch, switching between CPU and GPU is not as trivial as in Caffe. Few lines of code are needed to run the application in GPU. Selecting the device in multiple GPUs servers is also straightforward, as the user only needs to pass an argument to a function. The use of multiple GPUs is supported. However, unlike Caffe, the user needs to explicitly program which GPU executes each function. Whilst Torch is not as simple as Caffe in terms of device management, it offers a higher degree of freedom.

Device management in Lasagne is similar to Caffe, and accomplished by simply setting Theano flags in the command line arguments when running the Python script. The framework allows to choose between CPU and GPU and among multiple GPUs, if available in the server. The use of multiple GPUs simultaneously is accomplished by using GpuArray Backend[11] for model parallelism. Data parallelism can be done using the external library Platoon[12].

TensorFlow has different versions for running in CPU and GPU, and the latter supports both hardware. The device to be used needs to be set explicitly in the code, resembling Torch. The framework is flexible enough to allow specific parts of the code to run on different devices, as well as on multiple GPUs.

4.7 EXTENSIBILITY

Caffe allows its functionalities to be extended by modifying the C++ core. The user also needs to recompile the source code for changes to take effect. This process requires knowledge of the internal structure of the framework. Guides to perform the extension can be found in the community pages.

Torch can have its functionalities extended by coding in Lua or C/CUDA. Extension guides can be found in the Torch web page. Templates and examples facilitate the development of new functions.

Lasagne allows the user to create new layers by means of Python classes. If functionalities other than layers are required, it is necessary to modify the Theano library. Both methods are well documented on the Lasagne and Theano web pages.

TensorFlow can be extended by using Python and C++ languages. The documentation recommends the Python language for extensions that implement simple operations, and C++ for the complex ones. The web page contains well written guides to create new functions, layers, data readers, filesystems for multiprocessing communication, etc.

4.8 MODEL VISUALIZATION

To better visualize the network, Caffe offers a Python script that saves the model architecture as an image file. Some other tools for network visualization are offered by the community.

Torch allows the structure of models be visualized using the nnGraph package. The visualization shows the data flow and the operations executed throughout the network. This function is similar to the TensorBoard data flow visualization (see Section 2.4), but has less functionalities.

Lasagne data flow and operations, as seen in Torch and TensorBoard, can be drawn in an interactive form through the d3viz package available in Theano. Moreover, the network structure can be drawn using community provided Python scripts. The lack of an official tool to visualize the model architecture is a negative facet of Lasagne.

TensorFlow features TensorBoard, the official tool for model visualization. When compared with the visualization tools of preceding frameworks, TensorBoard is the most complete and understandable.

4.9 MODEL DEPLOYMENT

Models trained in Caffe can be deployed by exporting a .caffemodel file that is generated at the end of the training process. The Protobuf file that defines the network is also required. This process is straightforward, since a single command line is enough to use the trained model in a new application.

There are two ways to deploy models trained in Torch. The first uses simple save/load functions that wrap the network model and weights. The second method is to export the Torch tensor containing only the network weights. Both methods are fast and efficient.

In both, Lasagne and TensorFlow, deployment can be done by saving a serialized NumPy array that contains the network weights. For loading a trained model, the user has to redefine a model with the same structure and set the loaded weights. The drawback of this approach is the need to redefine an identical model each time it has to be loaded.

5 CONCLUSIONS

There are several frameworks for developing DL applications. The choice of a specific one for a giving application may be influenced by some features. This paper presented a qualitative analysis of nine issues (see Table 2), emerged during the development of a CNN for a printed character recognition application using four frameworks: Caffe, Torch, Lasagne and TensorFlow.

The qualitative dimensions analyzed in this work span from development and support issues to deployment of trained models. Besides the discussion provided, a qualitative rating is presented, so as to help the interested user towards the choice of a specific framework.

[11] http://deeplearning.net/software/theano/tutorial/using_multi_gpu.html
[12] https://github.com/mila-udem/platoon

Tabela 2: Qualitative evaluation of the framework, according to Section 3.

Point of View		Framework			
		Caffe	Torch	Lasagne	TensorFlow
Documentation	Installation	Excellent	Excellent	Excellent	Excellent
	Tutorials	Good	Excellent	Good	Good
	API Documentation	Fair	Good	Excellent	Excellent
Community Support		Good	Excellent	Fair	Excellent
Pre-trained Models		Excellent	Good	Fair	Fair
Ease of Learning		Excellent	Fair	Good	Fair
Ease of Use		Excellent	Good	Good	Good
Device Management	Ease of Management	Excellent	Good	Excellent	Good
	Capability	Good	Excellent	Good	Excellent
Extensibility		Fair	Excellent	Excellent	Good
Model Visualization		Fair	Good	Fair	Excellent
Model Deployment		Excellent	Excellent	Good	Good

In fact, the qualitative analysis revealed that all the frameworks have their pros and cons. As a matter of fact, the choice of a given framework may be influenced by the previous experience of the user but, hopefully, this choice can be supported by the analysis provided in this paper. It is important to stress that the evaluations are based on the last available versions of the frameworks, and may have changes along time, as the frameworks evolve and new versions appear.

During the development of the application, an effort was done by the authors to standardize the implementation, taking into account the distinct features of each framework. This was done to make this evaluation as fair as possible.

During the development of this work, some difficulties were found when using the frameworks for the specific application previously described. Problems with GPU memory allocation appeared in both Torch and TensorFlow. The frameworks allocate memory in all GPUs available in the server, even if just one GPU was explicitly set to be used. This problem did not occur when using Caffe or Lasagne. Some recent GPUs do not work with older versions of CUDA or cuDNN. Therefore, if it is necessary to run Theano in different versions of CUDA, it is needed to re-install Theano from scratch. All frameworks, but Caffe, did not use 100% of the GPU processing power all the time for this specific application, possibly due to their hardware management capability.

In order to foster other potential users, and provoke improvements in the frameworks, all the source codes developed in this work will be available under request, and can be used as a starting point for other similar pattern recognition tasks, as well as comparisons.

In the near future, this work will be extended by including other frameworks or other dimensions for analysis. Also, bindings of the evaluated frameworks that are still in early versions can be included, such as Keras, PyTorch (for Torch or Java) and Go (for TensorFlow), and stable bindings as MatCaffe and PyCaffe for Caffe.

ACKNOWLEDGEMENTS

N.M.R. Aquino thanks the Organization of the American States and the Coimbra Group of Brazilian Universities; M. Ribeiro thanks the Catarinense Federal Institute; H. Senefonte thanks the Universidade Estadual de Londrina; H.S. Lopes thanks to CNPq for the research grant no. 311778/2016-0. Authors M. Gutoski, L.T. Hattori and N.M.R. Aquino thank CAPES and CNPq for the scholarships. All authors thank NVIDIA for the donation of GPU boards.

REFERENCES

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

[2] Y. LeCun, K. Kavukcuoglu and C. Farabet. “Convolutional networks and applications in vision”. In *IEEE International Symposium on Circuits and Systems*, pp. 253–256, Piscataway, NJ, 2010. IEEE Press.

[3] A. Alotaibi and A. Mahmood. “Deep face liveness detection based on nonlinear diffusion using convolution neural network”. *Signal, Image and Video Processing*, vol. 11, no. 4, pp. 713–720, 2017.

[4] B. Jia, W. Feng and M. Zhu. “Obstacle detection in single images with deep neural networks”. *Signal, Image and Video Processing*, vol. 10, no. 6, pp. 1033–1040, 2016.

[5] M. Hasan, J. Choi, J. Neumann, A. K. Roy-Chowdhury and L. S. Davis. “Learning Temporal Regularity in Video Sequences”. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 733–742, Piscataway, NJ, 2016. IEEE Press.

[6] A. R. Revathi and D. Kumar. “An efficient system for anomaly detection using deep learning classifier”. *Signal, Image and Video Processing*, vol. 11, no. 2, pp. 291–299, 2017.

[7] C. Angermueller, T. Pärnamaa, L. Parts and O. Stegle. "Deep learning for computational biology". *Molecular Systems Biology*, vol. 12, no. 7, 2016.

[8] S. Min, B. Lee and S. Yoon. "Deep learning in bioinformatics". *Briefings in Bioinformatics*, vol. bbw068, pp. 1–19, 2016.

[9] P. N. Druzhkov and V. D. Kustikova. "A survey of deep learning methods and software tools for image classification and object detection". *Pattern Recognition and Image Analysis*, vol. 26, no. 1, pp. 9–15, 2016.

[10] X. Wang, X. Dong, X. Kong, J. Li and B. Zhang. "Drogue detection for autonomous aerial refueling based on convolutional neural networks". *Chinese Journal of Aeronautics*, vol. 1, no. 1, pp. 1 – 11, 2016.

[11] S. Shi, Q. Wang, P. Xu and X. Chu. "Benchmarking State-of-the-Art Deep Learning Software Tools". arXiv:1608.07249, 2017.

[12] S. Bahrampour, N. Ramakrishnan, L. Schott and M. Shah. "Comparative Study of Deep Learning Software Frameworks". *arXiv:1511.06435*, pp. 1–9, 2015.

[13] H. A. Perlin and H. S. Lopes. "Extracting human attributes using a convolutional neural network approach". *Pattern Recognition Letters*, vol. 68, no. 2, pp. 250–259, 2015.

[14] V. Andrearczyk and P. F. Whelan. "Using filter banks in Convolutional Neural Networks for texture classification". *Pattern Recognition Letters*, vol. 84, pp. 63 – 69, 2016.

[15] L. G. Hafemann, R. Sabourin and L. S. Oliveira. "Writer-independent feature learning for offline signature verification using deep convolutional neural networks". In *International Joint Conference on Neural Networks*, volume 30, pp. 2576–2583, Piscataway, NJ, 2016. IEEE Press.

[16] X. Zhang, J. Zhao and Y. LeCun. "Character-level convolutional networks for text classification". In *Advances in Neural Information Processing Systems*, pp. 649–657, 2015.

[17] A. Krizhevsky, I. Sutskever and G. E. Hinton. "Imagenet classification with deep convolutional neural networks". In *Advances in Neural Information Processing Systems*, pp. 1097–1105, Red Hook, NY, 2012. Curran Associates, Inc.

[18] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama and T. Darrell. "Caffe: Convolutional Architecture for Fast Feature Embedding". *arXiv:1408.5093*, pp. 1–4, 2014.

[19] Z. Zuo, G. Wang, B. Shuai, L. Zhao and Q. Yang. "Exemplar based Deep Discriminative and Shareable Feature Learning for scene image classification". *Pattern Recognition*, vol. 48, no. 10, pp. 3004 – 3015, 2015.

[20] R. Collobert, K. Kavukcuoglu and C. Farabet. "Torch7: A MATLAB-like Environment for Machine Learning". In *BigLearn, Neural Information Processing Systems Workshop*, pp. 1–6. Trafford Publishing, 2011.

[21] S. Dieleman, J. Schlüter, C. Raffel, E. Olson, S. K. SÃ¸nderby, D. Nouri and et al. "Lasagne: First release." 2015. http://dx.doi.org/10.5281/zenodo.27878.

[22] R. Al-Rfou, G. Alain, A. Almahairi, C. Angermueller, D. Bahdanau, N. Ballas and et al. "Theano: A Python framework for fast computation of mathematical expressions". *arXiv:1605.02688*, pp. 1–19, 2016.

[23] B.-I. Cîrstea and L. Likforman-Sulem. "Tied Spatial Transformer Networks for Digit Recognition". In *International Conference on Frontiers in Handwriting Recognition*, pp. 524–529, Piscataway, NJ, 2016. IEEE Press.

[24] Z. Tősér, L. A. Jeni, A. Lőrincz and J. F. Cohn. "Deep Learning for Facial Action Unit Detection Under Large Head Poses". In *Computer Vision–ECCV 2016 Workshops*, pp. 359–371, Gewerbestrasse, Switzerland, 2016. Springer International Publishing.

[25] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro and et al. "TensorFlow: large-scale machine learning on heterogeneous systems". *arXiv:1603.04467*, pp. 1–19, 2016.

[26] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan and D. Krishnan. "Unsupervised Pixel-Level Domain Adaptation with Generative Adversarial Networks". *CoRR*, vol. abs/1612.05424, 2016.

[27] R. Patiyal and P. Rajan. "Acoustic scene classification using deep learning". In *Detection and Classification of Acoustic Scenes and Events*, pp. 1–5, Piscataway, NJ, September 2016. IEEE.

[28] X. Li, Y. Zhang, I. Marsic, A. Sarcevic and R. S. Burd. "Deep Learning for RFID-Based Activity Recognition". In *Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems*, 16, pp. 164–175, New York, NY, 2016. ACM.

[29] V. Bui and L.-C. Chang. "Deep Learning Architectures for Hard Character Classification". In *Proceedings on the International Conference on Artificial Intelligence*, pp. 108–114, Athens, GA, 2016. CSREA Press.