Rapid tomographic reconstruction through GPU-based adaptive optics

CARLOS GONZÁLEZ GUTIÉRREZ[*], *Department of Exploitation and Exploration of Mines, University of Oviedo, Oviedo, Spain.*

MARÍA LUISA SÁNCHEZ RODRÍGUEZ[**], *Department of Physics, University of Oviedo, Oviedo, Spain.*

RAMÓN ÁNGEL FERNÁNDEZ DÍAZ[†], *Department of Architecture and Technology of Computers, University of León, León, Spain.*

JOSÉ LUIS CALVO ROLLE[††], *Department of Industrial Engineering, University of A Coruña, La Coruña, Spain.*

NIEVES ROQUEÑÍ GUTIÉRREZ[§] AND FRANCISCO JAVIER DE COS JUEZ [§§], *Department of Exploitation and Exploration of Mines, University of Oviedo, Oviedo, Spain.*

Abstract

Large telescopes have important challenges in the near future. Increasing the size of mirrors and sensors suppose not only a design issue, but also new computational techniques are needed to deal with the large amount of data. Adaptive Optics is an essential part of extremely large telescopes, and it uses reference stars and a tomographic reconstructor to compensate the aberrations introduced by the atmosphere during observation. The Complex Atmospheric Reconstructor based on Machine lEarNing (CARMEN) is a tomographic reconstructor based on neural networks which has been used during on-sky observations. In this paper CARMEN will be implemented in two different neural network frameworks, which use a Graphics Processing Unit to improve their performance. To time the training and execution will provide results of which framework is faster for its implementation in a real telescope and will supply new tools to keep improving the reconstruction ability of CARMEN.

Keywords: Neural Networks, Torch, TensorFlow, Adaptive Optics.

1 Introduction

Adaptive Optics (AO) [6, 37] is an essential tool when stellar observation is performed with grounded telescopes, since the atmosphere produces aberrations in the light passing through it. This problem is first approached with the measurement of distortions produced in the wavefront of the

[*]E-mail: gonzalezgcarlos@uniovi.es
[**]E-mail: jdsantos@uniovi.es
[†]E-mail: ramon.fernandez@unileon.es
[††]E-mail: nievesr@uniovi.es
[§]E-mail: jlcalvo@udc.es
[§§]E-mail: fjcos@uniovi.es

Vol. 27, No. 2, © The Author(s) 2018. Published by Oxford University Press.
Advance Access published 12 September 2018 doi:10.1093/jigpal/jzy034

incoming light and then, calculating shape that a deformable mirror has to adopt for compensating the aberrations in the wavefront as fast as possible due to the extremely changing nature of the atmosphere [17]. In order to measure more accurately the wavefront, guide stars are needed as a reference, where natural stars can be used when they are in the field of view of the object of interest, or even artificial stars that are created by laser scattering in the upper atmosphere. With the aim of correcting the error from the atmospheric turbulences, computer tomography techniques are considered for compensating the astronomical image with deformable mirrors [10, 18].

Creation and development of algorithms that allow the obtainment of the deformations introduced by the atmosphere is one of the imperative issues of AO [7]. Moreover, these algorithms have to command the real-control system, so the image can be accurately reconstructed. Some of the most common reconstructors are the Least Squared type matrix vector multiplication [12], the Learn and Apply method [38] and the recently added Complex Atmospheric Reconstructor based on Machine lEarNing (CARMEN), which has shown some interesting results [26].

Nowadays for large telescopes, modern AO systems are required which relays on tomographic techniques in order to reconstruct the phase aberrations induced by the turbulent atmosphere [5]. CARMEN is a reconstructor for Multi-Object Adaptive Optics (MOAO) [2, 20], which is one of these techniques. It was initially developed using regression algorithms as Multivariate Adaptive Regression Splines (MARS) [3, 9, 35], with promising results [18]. However, the use of machine learning techniques such as Artificial Neural Networks (ANNs), with high success in several fields [13, 21], led to create a solution based on ANN. The neural network is trained with a large range of possible turbulent layer positions and therefore does not require any input of the optical turbulence profile. CARMEN has shown promising results, both in simulation [10, 26] and with on-sky data [27, 28].

The development of large telescopes, in particular the future European Extremely Large Telescope (E-ELT), brings the inconvenience of the computational capability needed to process the enormous amounts of data [30]. Due to the larger number of subapertures and guide stars involved, tomography on ELT scales becomes computationally more difficult. ANN architecture allows its parallelism in neural processing. The use of Graphics Processor Units provides a solution to this problem, due to the parallelization of different calculations, and therefore, the speeding up of the processing times. There are some initial approximations to adapt some of the existing reconstructors, to the use of Graphics Processing Units (GPUs), like the Learn + Apply case [22, 23] or some first attempts with CARMEN [14, 34].

The main purpose of this paper is to detail the implementation of CARMEN, in two adapted frameworks for neural networks based on GPU and expose their training and execution times. Next section shows how AO systems work, and on the following one the architecture of CARMEN is shown as well as a small description of the frameworks used. The experiment is defined afterwards and different variables to compare different frameworks are proposed. Finally, results are analysed and conclusions are put forward.

2 AO Systems

The Shack–Hartmann WaveFront Sensor (SH-WFS) [29] is commonly used in astronomy to characterize an incoming wavefront. It is composed by several lenses with the same focal length, focusing each one of them in a different photon sensor. When the incoming light comes through the sensor, it is divided into discrete areas, and it is possible to measure the deviation from each one of them with respect to the ideal position, as it is shown in Figure 1. By doing this, it is possible to

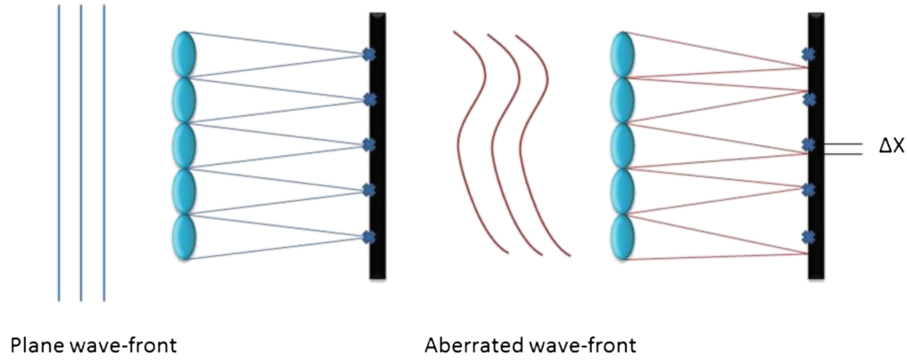

FIGURE 1. Measurement of wavefront tilts.

create a matrix of the different tilts, which allow to characterize the wavefront aberration by means of Zernike Polynomials [33]. The matrix created with the deviations of reference stars is used as input to the system, since it provides information about the deformations introduced by the atmosphere.

CANARY [24] is an AO on-sky demonstrator, principally intended for developing and testing AO concepts for the future 39m E-ELT. It is operated on a Nasmyth platform of the 4.2m William Herschel Telescope, one of the Isaac Newton Group of Telescopes of the Observatorio del Roque de los Muchachos, La Palma, Canary Islands, Spain.

There are several configurations available. The first two systems have been already used in real observation [5], and the third one is still under development at Durham University [31].

- **CANARY Phase B1** is designed to perform observations with 1 Laser Guide Star (LGS), and up to 4 Natural Guide Stars (NGS). It has an SH-WFS with 7 × 7 subapertures, although only 36 of them are functional.
- **CANARY Phase C2** is designed for the study of Laser Tomography AO and MOAO. There are 4 Rayleigh LGS, each with a 14 × 14 subaperture SH-WFS, where only 144 of them are working.
- **DRAGON** aims to replicate CANARY concepts, to provide a single channel MOAO system with a woofer-tweeter DM configuration, 4 NGSs and 4 LGSs each with 30 × 30 subapertures. In this case, DRAGON is still a prototype, so we are going to use the worst possible scenario where all the subapertures are functional, which gives us a total of 900 subapertures per star.

3 CARMEN Architecture

CARMEN is a tomographic reconstructor based on ANN, whose architecture is a multi-layer perceptron, with a single hidden layer. It is composed by two fully-connected layers, where each neuron is connected to all the neurons in the previou s layer. The output of each neuron follows (1), where w is the weight of each connection, x is the value of the neurons in the previous layer, b is a constant value called bias and f is an activation function.

$$\mathrm{Y} = f\left(\sum_{i=0}^{n} (w_i \cdot x_i) + b\right) \tag{1}$$

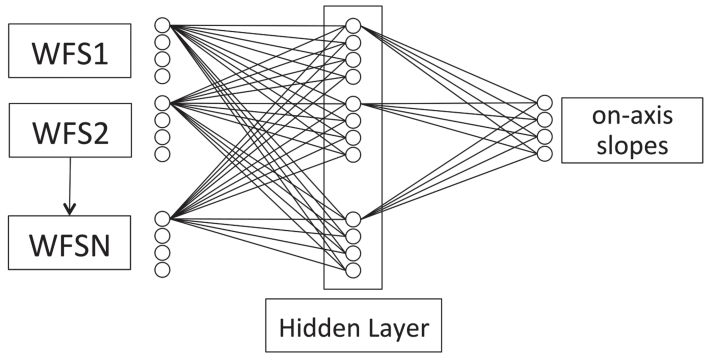

FIGURE 2. CARMEN Architecture.

The number of input, hidden and output neurons is directly related to the optical instrumentation used, since it changes with the number of subapertures of the device. It is not the purpose of this paper to analyse the net size or quality, since it has been previously tested [25], so the number of neurons in the hidden layer will be equal to the number of neurons in the input layer. To determine how many input and hidden neurons will have the network, it is necessary to know the number of reference stars that will be used, and the size of the different sensors. As it was explained in Section 2, this could be highly variable, as it can be observed in the explained systems. This means, that the size of the network is highly variable, and should be adapted to each observation. However, and to make it easier to compare, we are going to use the three configurations explained in the previous section and assume that we can observe both NGS and LGS.

With all these constraints, input neurons match the number of functional subapertures multiplied by 2—due to the two-dimensional input in the lenslet array—and also multiplied by the number of reference stars. The corrected deviations of the lenslet array will appear in the output, which is the number of functional subapertures multiplied by 2. The final size of the network can be summarized in Figure 2.

For CANARY Phase B1, it will be used one LGS and two NGS, which means using 216 input neurons. In the case of CANARY Phase C2, four LGS will be used, so there will be 1,152 input neurons. In both scenarios, training data has been obtained from the CANARY simulator. Lastly, for DRAGON, the situation is slightly different. Since it is still under development and there is no possibility to obtain simulated data, it is not possible to know how many natural stars are used. In this case, it will be assumed that the neural network will use the four LGS, and random data will be generated to train the network. This means there are 7200 input neurons for DRAGON. In Table 1 we can see a summary of the sizes of the different networks.

3.1 Neural networks frameworks

The growing popularity of ANNs in recent years has caused the emergence of numerous frameworks, which help researchers in the use and development of more complex neural networks, without the need of hard programming efforts. There is a long list of existing neural networks frameworks [16], but for this paper, we are going to focus mainly in two of them, Torch and TensorFlow.

Torch [8] is a scientific computing framework with wide support for machine learning algorithms. It is written in Lua and C/CUDA to provide a fast execution and the possibility of importing modules to complete and accelerate the system. It is mainly used and maintained by some important

TABLE 1. Size of neural networks for AO systems

Name	Size	Number of Training Samples
CANARY-B1	216-216-72	350,000
CANARY-C2	1152-1152-288	1,500,000
DRAGON	7200-7200-1800	1,000,000

companies, such as Facebook, Twitter, etc. It has been chosen due to their use in several previous studies, and it could serve as reference framework [4, 32, 34]. The used version has commit 89ede3b.

TensorFlow is an interface for expressing machine learning algorithms, and an implementation for executing such algorithms [1]. Its front-end is developed in Python and C++, and it allows to work with GPUs. TensorFlow is an open-source implementation, but it is mainly maintained by Google. The reason to choose this framework is their increasing popularity during the last couple of years, achieving more than 2,000 citations in this time [1]. The used version was v1.3.

4 Experiment Description

We have defined different networks for the optical instruments described above. Two different measurements have been considered to evaluate their performance. The different training times under specific conditions will be compared, and also the execution time of a network, which is crucial in AO systems.

4.1 Training benchmark

There are a lot of parameters to fit when training a neural network, although only a few of them affect directly the training time. To simplify the comparison, we will only vary those parameters that are especially relevant for the different systems.

In every case the backpropagation algorithm used to update the weights of the neural network is Stochastic Gradient Descent, with mini-batches and momentum, which is implemented both in Torch and TensorFlow. Two parameters are needed for this method: the learning rate and the momentum. Although these variables are crucial regarding the quality of the resulting network, they have no influence in the training times, so it will be assumed that both are optimized to achieve the best result as possible.

Another critical parameter is the size of the training data. However, it is easy to notice that in a training method based on mini-batches, the time grows proportionally to the number of samples employed. Keeping this in mind, it is easy to calculate how much time will take a network to be trained when changing the training dataset. This idea makes the choice of the dataset size irrelevant for benchmarking purposes, although is crucial for obtaining a good network. In this case, the size of the training data is specified in Table 1.

Three different networks will be used, one per optical system, as shown in Table 1. The main parameter to be changed will be the network size and the size of the mini-batch, in order to compare the times. The number of samples in the mini-batch will be 16, 32, 64, 128 and 256, as it has been done in different experiments [4, 32, 34].

First the initialized weights and bias are copied to the GPU's Video Random Access Memory (VRAM) and all the training dataset is loaded in the main Random Access Memory (RAM), before starting the training. Then a timer starts and the program starts to copy the first mini-batch from

RAM to VRAM, and to perform the loop to go over the entire dataset. This operation is repeated during 20 epochs, timing each of them individually. This procedure allows to obtain an average time for each epoch, and more reliable results, making it possible to see if there are significant time variations among different epochs.

4.2 Execution benchmark

The same networks defined in Table 1 are used in this case. However, there are some important differences with the training benchmark that should be detailed.

First, the net is fed with a single input, instead of using mini-batches, simulating what happens in a real telescope. As the execution program is intended to be integrated in the telescope management system is a fair assumption that all the variables are already initialized, and the weight matrices copied into the VRAM. The loading of input data from the Solid State Drive (SSD) will be taken into account in the time measurement, as well as the copy from VRAM to the system RAM, and in the case of loading from SSD, the writing process to the disk. Each input is found in a separate file, in h5 format, and the output is written in a separate file. We will feed the system with 10,000 inputs, one at a time, which allow us to average the execution time. We will compare the average time of the different frameworks, looking for which one is the fastest, and if there is any significant difference between them. In the case of TensorFlow it is not possible to know when data is copied to GPU, so this time will be omitted in the results.

4.3 Experiment equipment

The experiments are performed on a computer running on Ubuntu LTS 14.04.3, with Intel Xeon CPU E5-1650 v3 @ 3.50GHz, 128Gb DDR4 memory, Nvidia GeForce GTX TitanX and SSD hard drive. We used CUDA 8.0 and cuDNN v5.

5 Results and Discussion

In this section obtained results are shown, from training and execution process. Times will be split through different AO systems, and some comments about different results will be provided.

5.1 CANARY-B1

As explained above, the network size is 216-216-72, and 350k samples are used for training. As it is shown in Figure 1 and Figure 3, increasing the size of the batch has an obvious improvement on training times. There are substantial differences between both frameworks. While Torch is faster for small batch sizes, TensorFlow manages to reduce that difference and even achieves to be three times faster for the biggest size of batches. It is also interesting to remark, that in the case of Torch, increasing the size only make the training process two times faster, but TensorFlow manages to reduce those times most than ten times.

Regarding execution times, as shown in Figure 4, there is almost no difference between both systems. Another feature is that at least on Torch, most of the execution time is employed in copying the result from the SSD to RAM, and vice versa. This could mean that with a good integration between Torch and the real-time control system, it could be possible to avoid this unnecessary copy and save some crucial time.

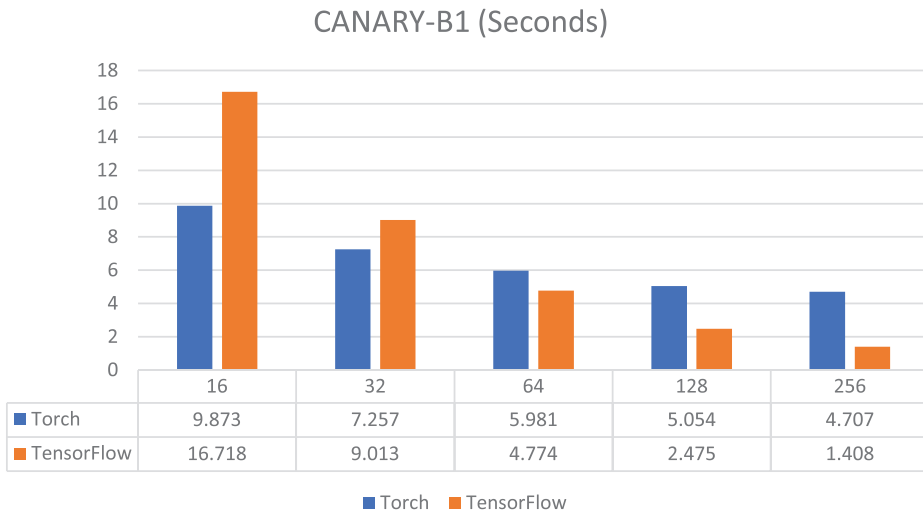

FIGURE 3. CANARY-B1 seconds of training per epoch.

5.2 CANARY-C2

For CANARY-C2, the network size is 1152-1152-288, and use 1.5M training samples. In this case, as shown in Figure 5, reducing the size has more impact for Torch than in CANARY-B1 and provides a similar behaviour in TensorFlow. However, it is interesting to notice that, even with a huge increase of the data and network size (about 28x more weights and 4x more data), the training times don't increase proportionally compared with the previous system. This could mean that the GPU is not fully loaded, so increasing the size of the network will be easily parallelizable without increasing training times.

In the case of execution times shown in Figure 6, the results are also very similar. However, it is interesting to notice that execution times are almost the same for CANARY-B1 than for CANARY-C2. As in the training process, this could be explained by the size of the network, since both are too

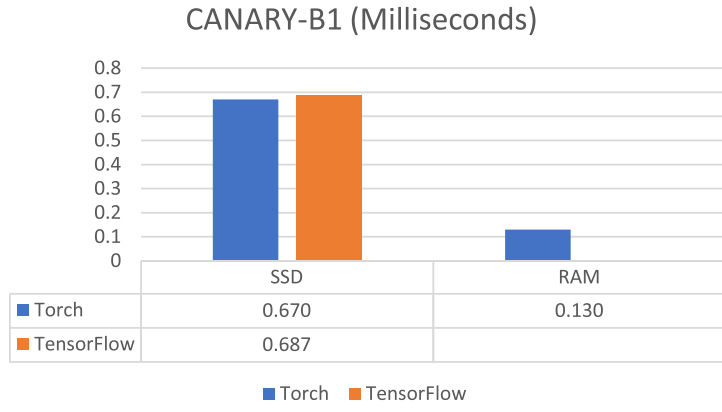

FIGURE 4. Execution times for CANARY-B1.

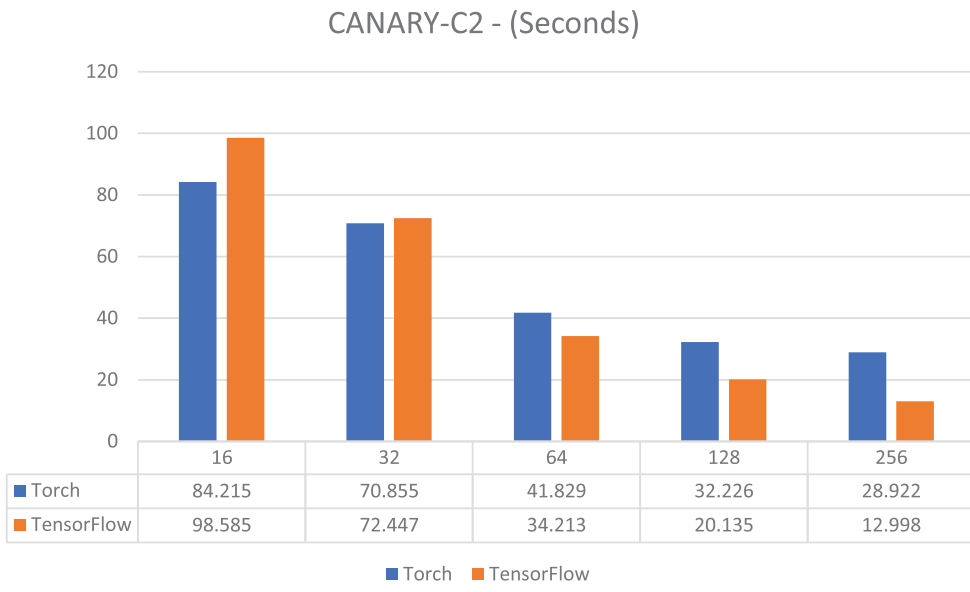

FIGURE 5. CANARY-C2 seconds of training per epoch.

small to fill the GPU. It is possible to observe that also in this case, most of the time is employed in copying the result from the SSD and getting it back.

5.3 DRAGON

For DRAGON, the largest network is employed, with 7200-7200-1800 neurons and 1M samples for training. This case is shown in Figure 7 and has some similarities with CANARY-C2, due to the great impact of reducing the batch size have on the training times. In this scenario, TensorFlow is again faster than Torch, although differences are much smaller.

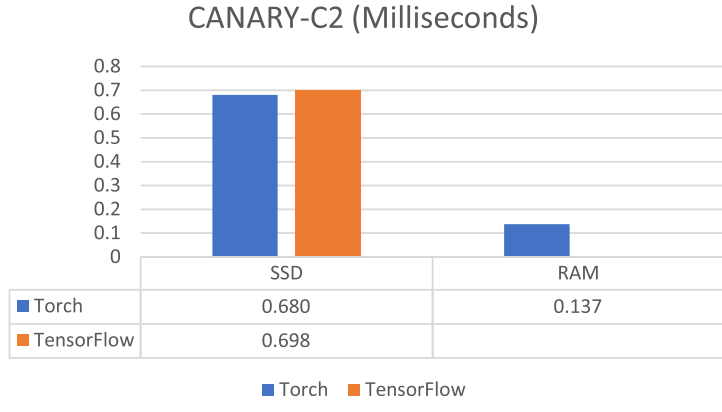

FIGURE 6. Execution times for CANARY-C2.

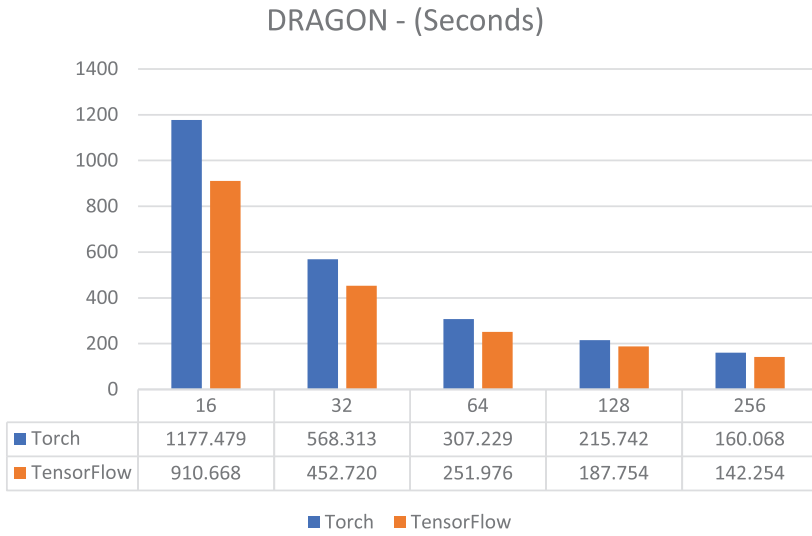

FIGURE 7. DRAGON seconds of training per epoch.

In forward propagation, both systems are once more quite similar. In this case, time increases compared to previous systems as it can be seen in Figure 8. Although the network is much bigger (about 40x times more weights), the execution time only increases by a 2.5x factor. For this execution, the impact of copying the result from SSD is lower, although it implies almost a 33% of the total time.

5.4 Discussion

There are some interesting results that can be extracted from the previous figures. One of the first consequences is that there are no significant variations between different epochs for the same system,

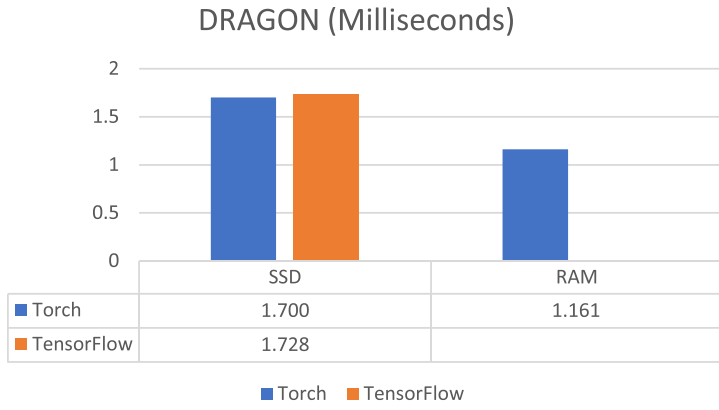

FIGURE 8. Execution times for DRAGON.

being the differences below than 1%. For most cases, TensorFlow is faster than Torch, especially with bigger batches. However, for small networks with small batches, Torch could be a good option. It is also remarkable that increasing the size of the different networks does not increase training times linearly. This could be explained because the size of the network is too small for the GPUs used, and they are able to parallelize most of the computations without increasing times.

In forward execution, it is possible to extract some interesting ideas. One of them is that TensorFlow and Torch expend almost the same time in the execution process, so the difference to improve training times is achieved during backpropagation. Also, execution shows that these networks are quite small for the GPUs, and increasing the size have a really low impact in execution time. Lastly, it is remarkable to notice that DRAGON, which is currently the largest network, could be executed by both frameworks in less than 2 milliseconds, which is the current limit to obtain the output in extremely large telescopes.

6 Conclusions and Future Lines

The use of a GPU-based framework to train and execute the network provides a powerful tool to test and improve the reconstructor much faster than it has been done during previous experiments [35]. Also, using these frameworks is a necessary improvement due to the increasing size of AO systems, especially the future E-ELT.

In this comparison it is possible to notice that in most cases TensorFlow is faster during training than Torch. However, both frameworks perform equally in execution, which provides two different tools to choose when implementing the reconstructor in a real-time control system for extremely large telescopes.

Also, as this is still a work in progress, there are a lot of different challenges to solve. One of them is to check the performance of these two frameworks when operating in a multi-GPU system, if it is possible to improve execution and training times. Another challenge is to deal with bigger systems with tens of thousands of inputs, and the ability of the different frameworks to handle that amount of data. The case of the future E-ELT, which is expected to have about 100,000 inputs and 5,000 outputs [11], will be especially interesting.

The use of specific neural network frameworks allows one to think about using convolutional neural networks [19] or recurrent networks [15], which could provide some boost regarding the quality of the reconstruction. Although this could be done programming the code directly in any programming language, using Torch or TensorFlow could make much easier to find a new architecture that could help CARMEN.

By last, it could be interesting to think about the possibilities of on-sky learning. This means to use real data from current observations, and to keep the training process during the observation. Recent studies have shown that it is a feasible idea [36], although it will require much further investigation and development.

Acknowledgements

Authors appreciate support from the Spanish Economics and Competitiveness Ministry through grant AYA2014-57648-P and the Government of the Principality of Asturias (Consejera de Economa y Empleo) through grant FC-15-GRUPIN14-017.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu and X. Zheng. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. 2016. http://arxiv.org/abs/1603.04467 (accessed October 9, 2017).

[2] D. R. Andersen, K. J. Jackson, C. Blain, C. Bradley, C. Correia, M. Ito, O. Lardière and J.-P. Véran. Performance modeling for the RAVEN multi-object adaptive optics demonstrator. *Publ. Astron. Soc. Pacific*, **124**, 469–484, 2012. doi:10.1086/665924.

[3] J. de Andrés, F. Sánchez-lasheras, P. Lorca and F. J. de Cos Juez. A Hybrid Device of Self Organizing Maps (SOM) and Multivariate Adaptive Regression Splines (MARS) for the forecasting of firms' bankruptcy. *Account. Manag. Inf. Syst*, **10**, 351–374, 2011.

[4] S. Bahrampour, N. Ramakrishnan, L. Schott and M. Shah. *Comparative Study of Deep Learning Software Frameworks*, arXiv preprint, arXiv:1511.06435, 2015. https://arxiv.org/abs/1511.06435.

[5] A. G. Basden, D. Atkinson, N. A. Bharmal, U. Bitenc, M. Brangier, T. Buey, T. Butterley, D. Cano, F. Chemla, P. Clark, M. Cohen, J.-M. Conan, F. J. De Cos, C. Dickson, N. A. Dipper, C. N. Dunlop, P. Feautrier, T. Fusco, J. L. Gach, E. Gendron, D. Geng, S. J. Goodsell, D. Gratadour, A. H. Greenaway, A. Guesalaga, C. D. Guzman, D. Henry, D. Holck, Z. Hubert, J. M. Huet, A. Kellerer, C. Kulcsar, P. Laporte, B. Le Roux, N. Looker, A. J. Longmore, M. Marteaud, O. Martin, S. Meimon, C. Morel, T. J. Morris, R. M. Myers, J. Osborn, D. Perret, C. Petit, H. Raynaud, A. P. Reeves, G. Rousset, F. Sanchez Lasheras, M. Sanchez Rodriguez, J. D. Santos, A. Sevin, G. Sivo, E. Stadler, B. Stobie, G. Talbot, S. Todd, F. Vidal, E. J. Younger and others. Experience with wavefront sensor and deformable mirror interfaces for wide-field adaptive optics systems. *Monthly Notices of the Royal Astronomical Society*, **459**, 1350–1359, 2016.

[6] J. M. Beckers. Adaptive optics for astronomy: principles, perfomance, and applications. *Annual Review of Astronomy and Astrophysics*, **31**,13–62, 1993. doi:10.1146/annurev.aa.31.090193.000305.

[7] G. Chanan. Principles of wavefront sensing and reconstruction. In *Cent. Adapt. Opt. Proc. Summer Sch. Adapt. Opt*, pp. 5–40. 2000. http://cfao.ucolick.org/aosummer/book/pdf/1.1_chanan.pdf.

[8] R. Collobert, C. Farabet, K. Kavukcuoglu and S. Chintala. Torch: a scientific computing framework for LuaJIT. 2017. http://torch.ch/ (accessed October 9, 2017).

[9] F. J. de Cos Juez, F. Sánchez Lasheras, P. J. García Nieto and A. Álvarez-Arenal. Non-linear numerical analysis of a double-threaded titanium alloy dental implant by FEM. *Appl. Math. Comput.*, **206**, 952–967, 2008. doi:10.1016/j.amc.2008.10.019.

[10] F. J. de Cos Juez, F. Sánchez Lasheras, N. Roqueñí and J. Osborn. An ANN-based smart tomographic reconstructor in a dynamic environment. *Sensors (Basel)*, **12**, 8895–8911, 2012. doi:10.3390/s120708895.

[11] N. A. Dipper, A. Basden, U. Bitenc, R. M. Myers, A. Richards and E. J. Younger. Adaptive Optics for Extremely Large Telescopes III ADAPTIVE OPTICS REAL-TIME CONTROL SYSTEMS FOR THE E-ELT. In *Adaptive Optics for Extremely Large Telescope III*, 2013. doi:10.12839/AO4ELT3.13267.

[12] B. L. Ellerbroek. First-order performance evaluation of adaptive-optics systems for atmospheric-turbulence compensation in extended-field-of-view astronomical telescopes. *Journal of the Optical Society of America A*, **11**, 783, 1994. doi:10.1364/JOSAA.11.000783.

[13] C. González-Gutiérrez, J. Santos, M. Martínez-Zarzuela, A. Basden, J. Osborn, F. Díaz-Pernas and F. De Cos Juez. Comparative study of neural network frameworks for the next generation of adaptive optics systems. *Sensors*, **17**, 1263, 2017. doi:10.3390/s17061263.

[14] C. González-Gutiérrez, J. D. Santos-Rodréguez, R. Á. F. Díaz, J. L. C. Rolle, N. R. Gutiérrez and F. J. de Cos Juez. Using GPUs to speed up a tomographic reconstructor based on machine learning. In *Int. Jt. Conf. SOCO'16-CISIS'16-ICEUTE'16, San Sebastian*, pp. 279–289. 2017. doi:10.1007/978-3-319-47364-2_27.

[15] A. Graves, A. Mohamed and G. Hinton. Speech recognition with deep recurrent neural networks. *Icassp*, 6645–6649, 2013. doi:10.1109/ICASSP.2013.6638947.

[16] C. Gulcehre. Deep Learning - Software *Links*. 2016. http://deeplearning.net/software_links/ (accessed June 5, 2017).

[17] D. Guzmán, F. J. De Cos Juez, R. Myers, A. Guesalaga and F. Sánchez Lasheras. Modeling a MEMS deformable mirror using non-parametric estimation techniques. *Optics Express*, **18**, 21356–21369, 2010. doi:10.1364/OE.18.02135.

[18] D. Guzmán, F. J. de Cos Juez, F. Sánchez Lasheras, R. Myers and L. Young. Deformable mirror model for open-loop adaptive optics using multivariate adaptive regression splines. *Optics Express*, **18**, 6492–6505, 2010. doi:10.1364/OE.18.006492.

[19] A. Krizhevsky, I. Sutskever and G. E. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *Adv. Neural Inf. Process. Syst*, F. Pereira, C. J. C. Burges, L. Bottou and K. Q. Weinberger eds. vol. 25, pp. 1097–1105. Curran Associates, Inc, 2012. http://dx.doi.org/10.1016/j.protcy.2014.09.007.

[20] O. Lardière, D. Andersen, C. Blain, C. Bradley, D. Gamroth, K. Jackson, P. Lach, R. Nash, K. Venn, J.-P. Véran, C. Correia, S. Oya, Y. Hayano, H. Terada, Y. Ono and M. Akiyama. Multi-object adaptive optics on-sky results with Raven. In *International Society for Optics and Photonics*, E. Marchetti, L. M. Close and J.-P. Véran, eds, p. 91481G. 2014. doi:10.1117/12.2055480.

[21] Y. LeCun, Y. Bengio and G. Hinton. Deep learning, *Nature*, **521**, 436–444, 2015. doi:10.1038/nature14539.

[22] H. Ltaief and D. Gratadour. Shooting for the Stars with GPUs. In *GPU Technol. Conf*. 2015, http://on-demand.gputechconf.com/gtc/2015/video/S5122.html (accessed March 14, 2016).

[23] J. G. Marichal-Hernández, L. F. Rodríguez-Ramos, F. Rosa and J. M. Rodríguez-Ramos. Atmospheric wavefront phase recovery by use of specialized hardware: graphical processing units and field-programmable gate arrays. *Applied Optics*, **44**, 7587–7594, 2005. doi:10.1364/AO.44.007587.

[24] R. M. Myers, Z. Hubert, T. J. Morris, E. Gendron, N. A. Dipper, A. Kellerer, S. J. Goodsell, G. Rousset, E. Younger, M. Marteaud and others. CANARY: the on-sky NGS/LGS MOAO demonstrator for EAGLE. In *SPIE Astronomical Telescopes + Instrumentation*, p. 70150E–70150E. 2008.

[25] J. Osborn, F. J. De Cos Juez, D. Guzman, T. Butterley, R. Myers, A. Guesalaga and J. Laine. Open-loop tomography using artificial nueral networks. *Adaptive Optics for Extremely Large Telescope II*, 2011.

[26] J. Osborn, F. J. D. C. Juez, D. Guzman, T. Butterley, R. Myers, A. Guesalaga and J. Laine. Using artificial neural networks for open-loop tomography. *Optics Express.*, **20**, 2420–2434, 2012. doi:10.1364/OE.20.002420.

[27] J. Osborn, D. Guzman, F. J. de Cos Juez, A. G. Basden, T. J. Morris, É. Gendron, T. Butterley, R. M. Myers, A. Guesalaga, F. Sanchez Lasheras, M. Gomez Victoria, M. L. Sánchez Rodríguez, D. Gratadour and G. Rousset. First on-sky results of a neural network based tomographic reconstructor: Carmen on Canary. In *SPIE Astronomical Telescopes + Instrumentation, International Society for Optics and Photonics*, E. Marchetti, L. M. Close and J.-P. Véran, eds, p. 91484M, 2014. doi:10.1117/12.2057462.

[28] J. Osborn, D. Guzman, F. J. de Cos Juez, A. G. Basden, T. J. Morris, E. Gendron, T. Butterley, R. M. Myers, A. Guesalaga, F. Sánchez Lasheras, M. G. Victoria, M. L. Sánchez Rodríguez, D. Gratadour and G. Rousset. Open-loop tomography with artificial neural networks on CANARY: on-sky results. *Monthly Notices of the Royal Astronomical Society*, **441**, 2508–2514, 2014. doi:10.1364/AO.44.007587.

[29] B. C. Platt and R. Shack. History and principles of Shack-Hartmann Wavefront Sensing. *Journal of Refractive Surgery*, **17**, S573–S577, 2001. doi:10.3928/1081-597X-20010901-13.

[30] S. K. Ramsay, M. M. Casali, J. C. González and N. Hubin. The E-ELT instrument roadmap: a status report. 91471Z, 2014. doi:10.1117/12.2056341.

[31] A. P. Reeves, R. M. Myers, T. J. Morris, A. G. Basden, N. A. Bharmal, S. Rolt, D. G. Bramall, N. A. Dipper and E. J. Younger. DRAGON, the Durham real-time, tomographic adaptive optics test bench: progress and results. In *SPIE Astronomical Telescopes + Instrumentation, International Society for Optics and Photonics*, E. Marchetti, L. M. Close and J.-P. Véran eds, p. 91485U–91485U. 2014. doi:10.1117/12.2055415.

[32] S. Shi, Q. Wang, P. Xu and X. Chu. *Benchmarking State-of-the-Art Deep Learning Software Tools*. arXiv Prepr, 2016.

[33] W. H. Southwell. Wave-front estimation from wave-front slope measurements. *Journal of the Optical Society of America*, **70**, 998–1006, 1980. doi:10.1364/JOSA.70.000998.

[34] S. L. Suárez-Gómez, C. González-Gutiérrez, J. D. Santos-Rodríguez, M. L. Sánchez Rodríguez, F. Sánchez Lasheras and F. J. de Cos Juez. Analysing the performance of a tomographic reconstructor with different neural networks frameworks. In *Intell. Syst. Des. Appl. 16th Int. Conf. Intell. Syst. Des. Appl.*, A. M. Madureira, A. Abraham, D. Gamboa and P. Novaiseds, pp. 1051–1060. *Springer International Publishing, Cham*, 2017. doi:10.1007/978-3-319-53480-0_103.

[35] S. L. Suárez-Gómez, M. L. Sánchez, F. Blanco, J. Ayala and F. J. De Cos Juez. Successful sulfur recovery in low sulfurate compounds obtained from the zinc industry: evaporation–condensation method. *J. Hazard. Mater.*, **336**, 168–173, 2017. doi:10.1016/j.jhazmat.2017.04.051.

[36] S. L. Suárez Gómez, J. D. Santos Rodríguez, F. J. Iglesias Rodríguez and F. J. de Cos Juez. Analysis of the temporal structure evolution of physical systems with the self-organising tree algorithm (SOTA): application for validating neural network systems on adaptive optics data before on-sky implementation. *Entropy*, **19**, 103, 2017. doi:10.3390/e19030103.

[37] R. K. Tyson. *Principles of Adaptive Optics*. CRC Press, 2015.

[38] F. Vidal, E. Gendron and G. Rousset. Tomography approach for multi-object adaptive optics. *Journal of the Optical Society of America A, Optics, Image Science, and Vision*, **27**, A253–A264, 2010. doi:10.1364/JOSAA.27.00A253.

Received 10 October 2017