Efficient and versatile data analytics for deep networks

D. Garcia-Gasulla[1], J. Moreno-Vázquez[1], J.A. Espinosa-Oviedo[14], J. Conejero[1], G. Vargas-Solar[34], R.M. Badia[1], U. Cortés[2], T. Suzumura[5]

[1] *Barcelona Super Computing Centre, Spain*
[2] *Universitat Politècnica de Catalunya, Spain*
[3] *CNRS, France*
[4] *LIG-LAFMIA labs, France*
[5] *IBM T.J Watson Research Center, USA*

{dario.garcia, jonatan.moreno, javiera.espinosa, rosa.m.badia }@bsc.es, ia@cs.upc.edu genoveva.vargas@imag.fr suzumura@acm.org

Abstract-*Deep networks (DN) perform cognitive tasks related with image and text at human-level. To extract and exploit the knowledge coded within these networks we propose a framework which combines state-of-the-art technology in parallelization, storage and analysis. Our goal, to make DN models available to all data scientists.*[1]

I. INTRODUCTION

Deep learning networks (DN) learn data representations through the millions of features composing them [6]. This provides a trained DN with a rich representation language to, for example, perform object classification at a human-level. In the case of images, each feature learnt by a convolutional neural network provides a significant piece of visual information on the image, even if these features are not optimal for discrimination (only the top layer features are). In a sense, by considering all feature values for a given image, one is in fact looking at everything the network sees in an image. The goal of the Tiramisu environment (see Fig. 1) is extracting and exploiting the knowledge coded within DN. Therefore, using a pre-trained network (e.g., GoogLeNet on ImageNet data [7]), and extracting the features through a deep learning toolkit ((DLT) e.g., Caffe [5]), Tiramisu builds alternative data representations and runs various analytics methods on top of them [4]. This paper presents the Tiramisu architecture, focusing on its unique requirements in terms of parallelism and data management and storage.

Data management in many data analytics work- flows is guided by the RUM conjecture (Read, Update, Memory (or storage) overhead) [2]. Several platforms address some aspect of the problem like Big Data stacks [3, 1]; data processing environments (e.g., Hadoop, Spark, CaffeonSpark); data stores dealing with the CAP theorem (e.g., NoSQL's); and distributed file systems (e.g., HDFS). The principle is to define API's (application programming interface) to be used by programs to interact with distributed data management layers that can cope with distributed and parallel architectures. The challenge is to have tools that can change their preferences towards RUM and provide elastic strategies for implementing these operations. Such strategies should evolve as data acquire different structures and semantics as a result of the data processing operations applied on them.

III. PARALLELISM AND STORAGE REQUIREMENTS

Tiramisu depend on the analytics to be executed. The parallel execution of various DLT instances that process various inputs must be handled by a general purpose parallel programming model and execution platform (e.g., PyCOMPSs). Other data analytics processes, such as vector distances or graph clustering, may be either built in programs in Tiramisu, or provided by third parties (e.g., Spark, ScaleGraph). The challenge for Tiramisu is to provide an architecture that (i) connects to each of those components, and (ii) accesses the most appropriate on each specific context.

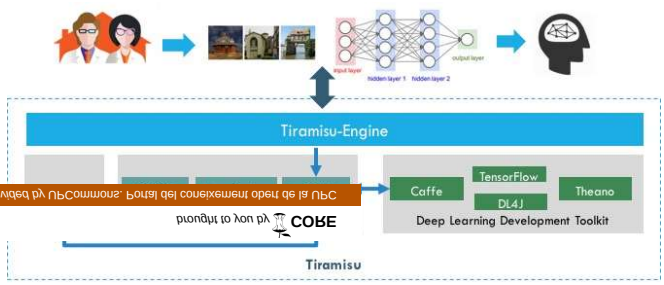

Fig. 1. Tiramisu architecture

IV. MANAGING DATA FOR TIRAMISU

The classification approach in Tiramisu is performed by a data centric workflow. It consists in a sequence of parallel analytics operations that process a data set of images and

[1] Details about the approach and implementation can be found in the proceedings of the 7th International Supercomputing Conference in Mexico (ISUM 16) organized from 11-13th April 2016 in Puebla, Mexico.

generate subsequent data collections with incremental content and semantics. Such operations require prepared input data collections (tagged and indexed), clever data colocation across processing nodes and results storage (cf. Fig. 2). A typical Tiramisu object contains a set of metadata (e.g., source image), and a set of values (e.g., millions of floats). Depending on the specific data representation, different types of persistence may be used, which must be consistent with the set of tools providing parallel analytic services. Indeed, well adapted data look-up, querying and exploration tools must be provided to ensure transparent access to data scientists.

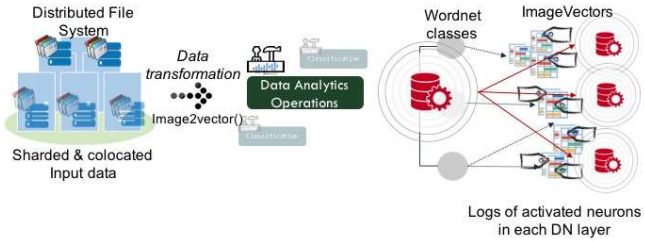

Fig. 2. Tiramisu data flow

ACKNOWLEDGMENT

This work is partially supported by the IBM/BSC Technology Center for Supercomputing (Joint Study Agreement, No. W156463) and the French Council of Scientific Research through its UMI 3175.

REFERENCES

A. Alexandrov, R. Bergmann, S. Ewen, J.C. Freytag, F. Hueske, A. Heise, O. Kao, M. Leich, U. Leser, V. Markl, et al. The stratosphere platform for big data analytics. *The VLDB Journal*, 23(6):939–964, 2014.

M. Athanassoulis, M. Kester, L. Maas, R. Stoica, S. Idreos, A. Ailamaki, and M. Callaghan. Designing access methods: The rum conjecture. *In International Conference on Extending Database Technology (EDBT)*, 2016.

Matthew Franklin. The berkeley data analytics stack: Present and future. *In Big Data, 2013 IEEE International Conference*, pages 2–3. IEEE, 2013.

D Garcia-Gasulla, J Béjar, U Cortés, E Ayguadé, and J Labarta. A visual embedding for the unsupervised extraction of abstract semantics. arXiv preprint arXiv:1507.08818, 2015.

Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional Architecture for Fast Feature Embedding. arXiv preprint arXiv:1408.5093, 2014.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. Nature, 521(7553):436–444, 2015.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. arXiv preprint arXiv:1409.4842, 2014.