Analysis and Performance Evaluation of Deep Learning on Big Data

Kassiano J. Matteussi, Breno F. Zanchetta, Germano Bertoncello, Jobe D. D. Dos Santos
Julio C. S. Dos Anjos, Claudio F. R. Geyer
Institute of Informatics
Federal University of Rio Grande do Sul
Porto Alegre, Brazil 91509–900
Email: {kjmatteussi, bfzanchetta, gbertoncello, jobe.dylbas, jcsanjos, geyer}@inf.ufrgs.br

***Abstract*—Deep Learning (DL) and Big Data (BD) have converged to a hybrid computing paradigm that merges the dynamic processing in DL models with the computational power of the distributed processing of the BD frameworks. In this context, this work aims to conduct an analysis and performance evaluation of DL applications in BD. The experiments evaluate how the application training completion time can be related to the model's precision loss and the impacts of distributed computing in DL models. The experiments were performed in Microsoft Azure using BigDL framework, which allows using both Spark and TensorFlow on top of a Yarn cluster. The outcomes revealed a speedup of up to 8x and accuracy higher than 95%.**

Keywords*— *Deep Learning, Distributed Deep Learning, Big Data, BigDL, Apache Spark, Parallel Processing

I. Introduction

The current trends in Data Science contain two main fields: Big Data (BD) and Deep Learning (DL) - a branch of Artificial Intelligence (AI). The first attracts many organizations that seek to obtain value from complex datasets to improve the business process through data processing and analysis [1]–[4]; The second allows machines to create their own conceptual knowledge over the data with the use of ML algorithms. In a more complex concept, it solves problems that are intuitive for humans, but very difficult to describe in objective terms to a machine [5].

DL is used in many areas such as image processing (*e.g.* facial recognition) [6], [7], object detection [8], robotics [9], recurrent networks (*e.g.* autonomous translation) [10]–[12] and acoustic processing [13], [14]. DL solutions development is very complex in large-scale and can conduct to errors in tasks such as data acquisition and partitioning; allocation and management of resources; workload balancing; model deployment; fault tolerance and so on [15].

In addition, recent works [16]–[23] investigate the relation between BD stack and DL benefits. The studies show frameworks which the main goal is to accelerate the application training time by exploring specific libraries such as MLIB, and the performance of BD applications (Sort, Terasort, Wordcount and so on) whilst using DL features. Also, the union of several Artificial Intelligence (AI) solutions in a combined distributed execution framework may lead to failures in attending dynamic response and real-time decision-making requirements. Thus, these facts demand attention towards the development of better framework architectures [15]. Therefore, we identify a lack of comprehensive studies to analyze the impact of distributed processing in DL scenarios, as well as in the performance evaluation of structures, models and datasets in deep networks using local and distributed scenarios.

In this context, this work aims to conduct an analysis and performance evaluation of DL applications in BD. The evaluation will measure the impact of distributed computing of DL models in terms of application training completion time and model precision. The experiments represent a controlled real-world test scenario in the Microsoft Azure cloud. The BigDL framework [18] is instanced to allow to use both Spark and TensorFlow engines on top of a Yarn cluster.

The main contributions of this paper are: i) to provide a detailed performance analysis of DL applications distributed in BD environment; ii) to evaluate the maintenance of models precision towards distributed training and optimization.

This paper is structured as follows. Section II indicates the main fields studied in this work. Section III presents the closer works. The case of study and methodology of this work are shown in Section IV. The results are present in Section V and finally, the conclusion is in Section VI.

II. Background

Big Data (BD) refers to the increase in data *volume* (which is hard to process and store in traditional systems), the data *variety* (the indistinct data nature), *velocity* (considerable computational efforts) to add *value* with the needed *veracity* to the transformation of raw data into valuable and reliable information [24]. The literature review presents relevant contributions for BD scenarios in several contexts (cloud, energy, programming models, software, algorithms, security and so on) [19]. Meanwhile, Machine Learning (ML) introduced a series of real-world solutions with very complex autonomous operations.

However, ML still relied too much on human interaction in the readjustment of the algorithm, specially in scenario where the problem is intuitive for humans but very difficult to describe in objective terms to machines, thus contradicting the point of task automation. Then emerged DL as a branch of ML which allows the machines to create their conceptual knowledge on the data and combine them into other more complex self-generated ones [5].

978-1-7281-2999-0/19/$31.00 ©2019 IEEE

Although BD enables fast data processing, it is important to consider emerging problems that now are integrated to the ML and DL models. For instance, there are complex data access patterns, fast information retrieving, data classification and semantic indexing. Useful tools can be mentioned in different areas, such as a set of frameworks based on the MapReduce programming model in BD processing (*e.g.* Spark, Hadoop Yarn and Flink which allows the ML library (MLlib) use).

Second, with the focus in DL, it can be mentioned the main Neural Networks (NN), such as Feed-Forward Neural Network(*FNN*), which is composed of computational units based on Multi-Layer Perceptron (MLP) united into an acyclic graph; Convolutional Neural Network (*CNN*) unites convolution operations with FNN's feature extraction, thus improving image and video processing; Recurrent Neural Network (*RNN*) uses time sequences and internal states to process data sequences, thus providing great results in text and audio processing; *LSTM*(*Long-Short Term Memory*) is a RNN's particularity, that introduces low-term and long-term signals to solve RNN's long dependencies issues.

Also, when these NNs are implemented in DL frameworks, such as TensorFlow, Theano, PyTorch and Caffe, they are capable of being applied towards BD's feature learning [25]. Next section shows the recent state of the art on Big Data DL Analysis. The primary goal is to provide a general vision of the works that intend to merge both paradigms whilst highlighting the open problems.

The DL framework is wrapped into BG engines by the use of specif classes of them and the models of DL are spread between multiple instances of worker nodes. Thus, each worker receives a copy of the model and a small batch of the dataset from the driver node. On follow, the workers conduct fast and non-precise computations (training) that must be synchronized and sent to the driver node, meanwhile the worker nodes stay idle waiting for the next job. After that, when the master node finishes the average of the models and their optimization, the cycle starts again with new batches.

III. Related Work

In the work DeepSpark [16], the authors proposed a distributed and parallel DL framework that evaluates the use of Apache Spark in GPGPU-based commodity clusters. This approach distributes the workloads and adjusts some parameters automatically in order to set test scenarios to deploy Caffe's models into Spark framework. The author evaluates the turnaround time per training, considering 1000 iterations and presents a speedup analysis. The results revealed that training time reduced up to 42% using 16 executors and achieved 0.6 of accuracy. Our work uses more epochs in the training of models, hence achieves better results for the accuracy.

In SparkNet [17], Moritz *et al.* analyzes a framework for deep networks training in Spark. The implementation uses a simple parallelization scheme for Stochastic Gradient Descent to handle high-latency during communication. The authors evaluated the speedup of Caffe performed in Spark framework on top of a GPGPU-based cluster. The goal was to present the required time to reach an accuracy of 40%. The work evaluated the precision loss in DL model by the analysis of standalone and distributed scenarios.

Intel's BigDL [18] implements a distributed DL framework for BD with mini-batch synchronous optimization and a centralized parameter server for model training. BigDL can incorporate DL tools, such as TensorFlow, Caffe, Keras and PyTorch. Regarding BD, BigDL is implemented coupled with Spark. It allows integration in local mode with Apache Storm, Apache Flink and Apache Kafka due to JVM support. Also, a key aspect of BigDL is the quantization of 32-bit floating point into 8-bit integer, which does not significantly affect the models accuracy, but reduces its size by up to four times and results in a twofold distributed training speedup.

The author Gupta *et al.*, in [19], shows a framework that incorporates Apache Spark (using MLIB) and the advanced ML architecture of a deep MLP, plus cascade learning concept. The authors evaluated two real-world datasets to measure the difference from Spark (without modification) experiments against the proposed Framework. The F1 Score and Accuracy metrics that were identified and analyzed were not so expressive, but indicated the feasibility to use CPU for this type of processing. Our work also used CPU, but the main difference is the evaluation with varied datasets and models.

DLoBD [22] introduces a rich experimental methodology for DL models and datasets over BD in four study cases, while also evaluates how the combination of these architectures work. Even though it evaluates many models and frameworks, few combinations between frameworks converge in the same model, therefore its comparisons can be untrustworthy for inter-modular comparisons. However, the work conducted a fine-grained analysis of each model and is the most related work to ours.

In summary, to the extent of our knowledge, we observe a lack of analysis of the impact of distributed processing in DL scenarios (for instance, what happens with the accuracy of the models when they are processed in a distributed fashion) and of performance evaluation of neural networks, models and datasets using local and distributed scenarios.

IV. Proposed Approach

The development and optimization of models, applications, and frameworks into distributed environments put factors such as performance and scalability as a challenge. Even though the state-of-the-art presents studies that demonstrate performance evaluation in distributed DL, but they do not show clearly what is the impact that the distributed processing may cause to DL applications. This work aims to conduct an analysis and performance evaluation of DL on BD, keeping in mind these previous considerations. The proposed evaluation measures the impact of distributed computing of the DL models in terms of application completion time and model precision loss.

A. BigDL Framework

The proposed evaluation uses the BigDL framework because it offers a data representation that is common to the DL

frameworks, and is also compatible with BD. To support the parallel and distributed processing, BigDL is deployed on top of Apache Spark framework in an optimized way.

Even though BigDL performs the data-parallel training with synchronous optimizers like TensorFlowOnSpark [20], it also outperforms that framework, which incorporates Spark's datasets preparation in earlier stages of execution and bypasses the remainder of Spark's functions, turning to self-implemented libraries. BigDL [18] focuses to extract the full extent of Spark's functionality by using the orchestration layer to allocate cluster resources, then putting drivers nodes to schedule and dispatch jobs to workers, which perform computations and physically store data.

BigDL is different from other synchronous distributed tools because it reduces life spawn of Spark jobs and can grant more resilience towards resource alterations (for example, errors and failures, resource sharing, preemption) along the time. Also, it can be incorporated to a tool, named Drizzle, for task scheduling in very large-scale environments, which agglutinates large amounts of tasks for schedule and therefore it reduces overheads.

Finally, BigDL softens negative effects in accuracy in distributed DL due to Model Quantization for the dimensionality reduction (e.g. moving 2D convolution in NNs to a low-precision representation). Hence, 32-bit floating point set of parameters turn into 8-bit integer, thus granting up to fourfold model size reduction and twofold inference speedup with models accuracy drop of less than 0.1% [18].

Table I shows the combination of neural networks, models and datasets into BigDL, which was chosen for our analysis.

TABLE I
THE BIGDL'S STACK

Network	Model	Dataset
CNN	ResNet	CIFAR10
CNN	VGG	CIFAR10
CNN	LeNet	MNIST
CNN	Text Classifier (CNN)	20 NewsGroup/GloVe-6B
RNN	Text Classifier (LSTM)	20 NewsGroup/GloVe-6B
RNN	Tree-LSTM	Stanford Sentiment Treebank

The datasets used in this work are presented as follows:

- CIFAR-10: It is a set of RGB images separated in 10 distinct classes. This set is sub-divided in 50,000 images for training and 10,000 for tests, where each image is a 32x32 pixel RGB matrix;
- MNIST: It is composed of 70,000 images of 28x28 bilevel pixel with centralized hand-written digits. This dataset is classified into 10 distinct classes, with 60,000 images for training and 10,000 images for tests;
- 20NewsGroup: It is a collection with 20,000 documents of discussion groups separated in 20 different topics;
- Stanford Sentiment Treebank: Film Criticisms collected on Rotten Tomatoes and published by [26]. It consists of about 12,000 sentences and 215,000 unique phrases.

B. Environment

1) Software Stack: The software stack used in this work is described in Table II.

TABLE II
SOFTWARE STACK

Operating System Ubuntu 16.04.5 LTS
Apache Spark v2.2.0
Hadoop YARN v2.6.3
Hadoop Distributed File System (HDFS)
Java OpenJDK 8 181
Scala 2.11.8
Python 2.7.15 (NumPy v1.15.4, Six v1.11.0)
BigDL version 0.7.1.

2) Infrastructure: BigDL is CPU optimized through Math Kernel Library (MKL) for Intel Xeon™processor-based clusters. For instance, the work [27] revealed that BigDL delivers up to 3.83 times speedup on Xeon cluster compared to a GPU cluster. For this reason, the environment used for experiments execution is a Microsoft Azure CPU cluster, with node's settings shown in Table III.

TABLE III
NODES HARDWARE SPECIFICATIONS

Component	Specification
Processor	Intel Xeon™E5-2673 4 Core (2,4-3,1 GHz)
Memory	30 GB RAM
Storage	SSD 1TB

The driver and worker nodes belong to Microsoft's D12v2 hyper-threaded general purpose instances. Details about the clusters configuration can be observed in Table IV.

TABLE IV
CLUSTER CONFIGURATION

# Worker Nodes	# Cores	Memory(GB)	*Storage(TB)*
4 worker Nodes	16	120	4
8 worker Nodes	32	240	8
12 worker Nodes	48	360	12

3) Parameters Configuration: The main Spark parameters adjusted for the experiments performed on top of YARN are:

- Master: it determines the Spark execution mode and can be set as local or in the Yarn Kernel;
- Driver-memory: defines allocated memory for the driver;
- Executors-num: sets the number of executors;
- Executor-cores: indicates the core's number by executor;
- Executor-memory: defines allocated memory by executor;
- Class: sets the applications entry point.

C. Methodology

In this work, we evaluated the performance by a speedup analysis and measured the impact in TensorFlow models accuracy when using parallel and distributed processing paradigm provided by Spark in BigDL through a short scalability scenario. The metrics evaluated in this work are time (s) and accuracy loss (%). The model and datasets test scenarios can be described as a combination of 5 models with 4 datasets into 6 distinct possible tests. Moreover, we evaluated the local (standalone) environment and more 3 different cluster settings, coming to a total of 6 test scenarios in 4 cluster configurations, thus resulting in 24 distinct execution sets.

Also, even though Jain's experimental methodology [28] defines the ideal number of repetitions as 30 executions for each configuration, we reduced the tests to 10 repetitions because our set of results presented low standard deviation.

Still, with this threefold reductions, these experiments ensure a confidence interval of 95%, with a total of 240 experiments.

V. Evaluation

A. Performance Analysis

In the next figures the axis-y represents the execution time measured in seconds and axis-x shows the number of nodes in each experiment.

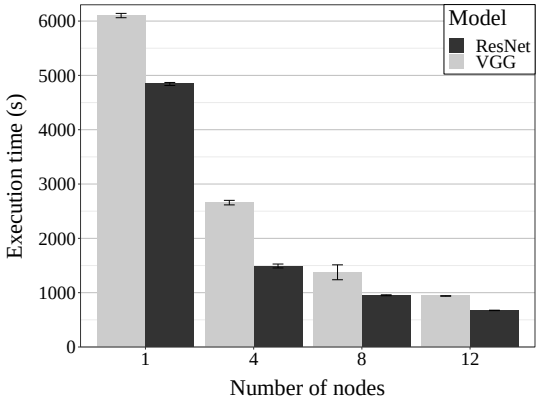

Fig. 1. Comparison of CNN models: ResNet and VGG on CIFAR-10 Dataset

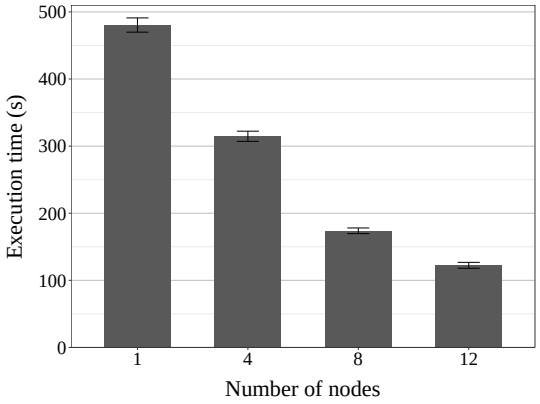

Fig. 2. CNN, LeNet Model - MNIST Dataset

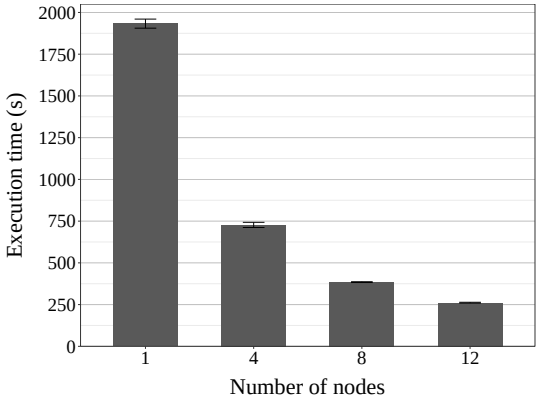

Fig. 3. CNN, Text Classifier Model - 20NewsGroup/GloVe-6B Dataset

Figure 1 shows the ResNet and VGG performances when executed on CIFAR-10 dataset. The VGG model reduced its training time by 84.6% with 12 worker nodes compared to baseline, while ResNet model decreased up to 86.1%.

Even though the performance gain is expressive for both models, ResNet outperforms VGG by a rate of 20% in the standalone scenario, and by 28% with 12 workers. These values correlate to a BigDL behaviour that tends to benefit deeper convolutional models as opposed to models with deeper feature extraction units in a distributed training.

In other words, the synchronous distributed training process in BigDL could be less suitable for models with too many neurons in the Fully Connected layer (VGG has twice compared to ResNet), it occurs due to massive increase in model parameters length. ResNet is also penalized on the distributed training, but unlike VGG, it is due to high costs of computations on many convolutional layers (increases elevenfold compared to VGG). Still, both scenarios may suffer convolutional and linear quantization for optimization purposes, and perhaps, ResNet residual blocks leverage parallelism during multi core training executions more than VGG's sequential convolutions, thus reducing training idle time.

In Figure 2 we present the performance of LeNet model using MNIST dataset. In comparison with standalone scenario, LeNet with 12 workers has 74.5% reduction in training time.

It is possible to observe in Figure 2 that the reduction in training time of LeNet also behaves nearly as a decreasing exponential curve. Still, even though the execution time was not reduced as much as in other models, the parallelism ensured performance gains, which can be verified in Table V. LeNet was the model with higher accuracy in all configurations to suffer among the lesser accuracy variations in the experiments.

Figure 3 presents the performance of Text Classifier (CNN model) on the 20NewsGroup/GloVe-6B dataset. CNN model reduces the training time up to 86.5% with 12 worker nodes compared to baseline.

In this case, it seems clear how parallelism supports application performance with minimal penalties. The accuracy penalty from distributed optimization and batch distribution is a low price due to greater execution time reduction, which can be put towards more steps of training to mitigate this trade-off. Also, the workload distribution possibly reduces resource usage (CPU, memory and other) avoiding interference and ensuring high data throughput.

Figure 4 shows the performance of Text Classifier (LSTM model) using 20NewsGroup/GloVe-6B dataset. The processing time decreases nearly to an exponential behavior until it reaches 87.4% reduction with 12 workers against the baseline.

Figure 5 shows the execution time of the Tree-LSTM model using Stanford Sentiment Treebank dataset. The model obtained a reduction of 44.5 % in training time with 12 worker nodes. In addition, we noted that this model did not show significant performance gains among the distributed runs, as seen in Table V. We believe that this effect occurs because LSTM networks require more epochs than regular CNNs to achieve plausible accuracy, thus the efforts to fixate a common number of epochs for all experiments -to grant fairness- generated this disastrous output for LSTM.

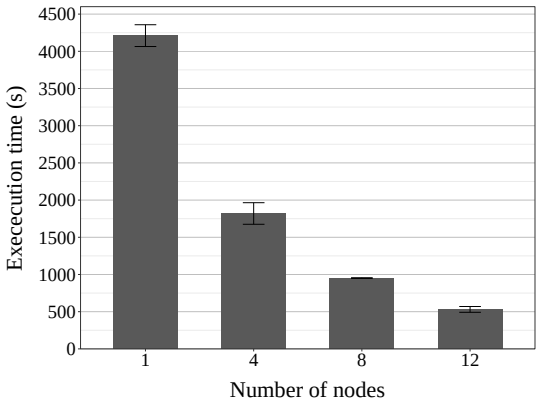

Fig. 4. RNN, Text Classifier Model - 20NewsGroup/GloVe-6B Dataset

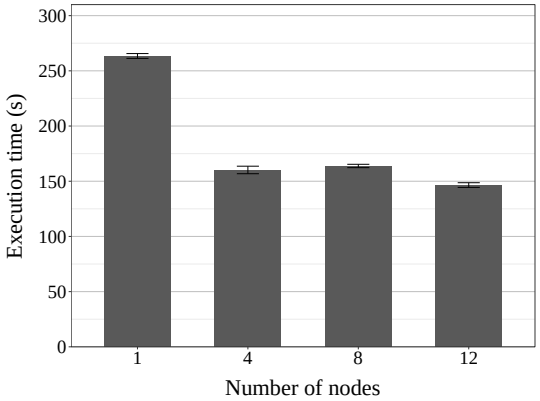

Fig. 5. RNN, Tree-LSTM Model - Stanford Sentiment Treebank Dataset

Complementary, Table V shows the obtained speedup and efficiency, to provide further data about the same experiments.

TABLE V
SPEEDUP AND EFFICIENCY

	Speedup			Efficiency			
Model	***4***	***8***	***12***	***Local***	***4***	***8***	***12***
ResNet	3.24	5.08	7.19	1	0.81	0.64	0.60
VGG	2.29	4.43	6.49	1	0.57	0.55	0.54
LeNet	1.43	2.77	3.93	1	0.36	0.35	0.33
TC-CNN	2.65	5.03	7.40	1	0.66	0.63	0.62
TC-LSTM	2.31	4.42	7.94	1	0.58	0.55	0.66
TreeLSTM	1.64	1.61	1.80	1	0.41	0.20	0.15

The obtained speedups indicating a high performance provided in the parallelism for DL models. Also, the results present that workload is a threshold for performance gain, *i.e.*, the performance can be optimized if an ideal equation or model is designed taking into account the number of worker nodes, computational resources, and the datasets. The ResNet model, for example, does not provide any gain when the distribution is 8 to 12 worker nodes, indicating an unnecessary level of parallelism. Another example is TC-LSTM model that always present benefits as the parallelism grows, meaning an adequate approach.

B. Distributed Deep Learning: a Precision Analysis

Even though BigDL shows expressive feasibility in distributed DL computing, as can be seen previously, the properties of DL models (e.g. accuracy) still need to be guaranteed. Thus, Table VI compares the precision when DL is distributed along 12 nodes. The Local label stands for average model precision for a standalone cluster. The node numbers 4, 8 and 12 represent the precision distributed among the nodes and the label variation (Variation) is a measurement between local and 12 nodes. BigDL's precision is available in logs, as an accuracy measure in percentage during validation steps.

TABLE VI
PRECISION OF DL MODELS (%)

Model	**Dataset**	**Local**	**4**	**8**	**12**	**Variation**
ResNet	CIFAR10	84.14	83.53	82.28	80.124	-4.77
VGG	CIFAR10	67,90	60,65	54,12	45,91	-32.38
LeNet	MNIST	98.50	98.16	97.27	95.64	-2.9
TC (CNN)	20NewsGroup GloVe-6B	85.64	85.63	84.50	84.22	-1.66
TC (LSTM)	20NewsGroup GloVe-6B	29.46	31.32	24.20	18.31	-37.85
Tree-LSTM	Stanford Sentiment Treebank	46.17	46.56	47.07	45.45	-1.56

It is possible to observe that most models preserved their precision with low variation, which ensures BigDL's optimizations for distributed training. This behavior can be analyzed with the following observations:

- The precision loss for ResNet model was: 0.72%, 1.49%, 2.62% for 4, 8 and 12 worker nodes, respectively. On total, the mean precision loss was 4.77%;
- The precision loss for LeNet model was: 0.34%, 1.23%, 1.67% from 4, 8 and 12 worker nodes, respectively. The total precision loss was just of 2.9%;
- Tree-LSTM showed successive precision gains of 0.84%, 1.1% and 3.44% from 4, 8 and 12 worker nodes, respectively. Overall, the mean precision loss on Tree-LSTM training was 1.56%, and the reasons for this behaviour will be explained later;
- The precision loss for Text Classifier (CNN) model was: 0.01%, 1.32%, 0.33% from 4, 8 and 12 worker nodes, respectively. On total, precision drop was only 1.6%.

The measure of VGG model, on the other hand, revealed a mean precision loss of 10.67%, 10.76%, 15.16% for 4, 8 and 12 nodes, respectively. Overall, the causes behind this 32.38% loss can be explained as follows: Overall, this model came to a loss of 32.38% between the standalone and 12 workers configuration. We believe that the causes can be explained as follows: The sequential convolutions increase and a high number of fully connected neurons generate an increase of training parameters, which even applied with quantization led to a complexity increase in processing, storing and data transferring. We also noticed that BigDL's implementation of VGG contains multiple instances of Dropout between each convolutional layer -possibly to speed convergence up-. This finding could explain why VGG's accuracy is so disappointing, whilst its high number of parameters contribute to an increase in training time. In conclusion, VGG needs more training epochs in order to achieve acceptable precision, and is not efficient as it is.

Another model that lost accuracy is Text Classifier with LSTM, which according to VI presented a gain of 6.31% and a loss between 22.02% and 24.34% for 4, 8 e 12 nodes. The distributed processing caused an average precision loss of 37.85%. Even though the 4 nodes configuration introduced

6.31% accuracy gain, the absolute values are so close that this phenomenon could be explained due to better dataset shuffling and batching on 4 nodes configuration, *i.e.* blind luck. On the other hand, the accuracy drops in the 8 and 12 nodes configuration happens because LSTMs rely on long and short dependencies for training. Thus, these networks suffer more penalties when receiving smaller batches, thus forming inconclusive dependencies. They need many more training epochs than CNNs to achieve acceptable precision rates. Tree-LSTM obtained nearly 45% of accuracy on its tests, which reveals that this model must need more epochs to achieve higher gains in accuracy.

VI. Conclusion

In this context, this work aimed to conduct an analysis and performance evaluation of DL applications in BD. Thus, this was provided in a Microsoft Azure cluster, where computation was performed by the Apache Spark on top of Hadoop Yarn. Our evaluation measured the impact caused in TensorFlow DL applications when processed in a parallel and distributed fashion. Outcomes presented feasibility of distributed processing, with a speedup of up to $\tilde{8}$x and loss in accuracy that was less than 5% in the best case. In the future, we intend to replicate the performed experiments using TensorFlowOnSpark and SparkNet frameworks. In addition, we intend to add more models and datasets, including ImageNet. Finally, we desire to develop this study to discover the ideal training distribution for each study case, that is, to determine the ideal configurations for each model.

VII. Acknowledgment

The authors thank the following Brazilian Agencies for supporting this work: FAPERGS Project "GREEN-CLOUD - Computação em Cloud com Computação Sustentável" (#16/2551-0000 488-9), "SmartSent" (#17/2551-0001 195-3), CAPES (Finance Code 001), CNPq and PROPESQ-UFRGS-Brasil.

References

[1] K. J. Matteussi, M. G. Xavier, C. A. F. de Rose, and C. F. R. Geyer, "Understanding and minimizing disk contention effects for data-intensive processing in virtualized systems," in *Proc. 16th International Conference on High Performance Computing and Simulation (HPCS)*, 2018, p. 901–908.

[2] J. C. S. dos Anjos, K. J. Matteussi, P. R. de Souza Jr, A. da Silva Veith, G. Fedak, J. L. V. Barbosa, and C. F. R. Geyer, "Enabling Strategies for Big Data Analytics in Hybrid Infrastructures," ser. HPCS - International Conference on High Performance Computing and Simulation. IEEE Computer Society, July 2018, p. 869–876.

[3] P. R. R. de Souza, K. J. Matteussi, J. C. S. dos Anjos, J. D. D. dos Santos, C. F. R. Geyer, and A. da Silva Veith, "Aten: A dispatcher for big data applications in heterogeneous systems," in *Proc. International Conference on High Performance Computing Simulation (HPCS)*, July 2018, pp. 585–592.

[4] S. M. Srinivasan, T. Truong-Huu, and M. Gurusamy, "Flexible bandwidth allocation for big data transfer with deadline constraints," in *Proc. IEEE Symposium on Computers and Communications (ISCC)*, July 2017, pp. 347–352.

[5] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT Press, 2016.

[6] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1701–1708.

[7] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, "Face recognition: A convolutional neural-network approach," *IEEE Transactions on Neural Networks*, vol. 8, no. 1, pp. 98–113, 1997.

[8] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.

[9] J. O. Gaya, L. T. Gonçalves, A. C. Duarte, B. Zanchetta, P. Drews, and S. S. Botelho, "Vision-based obstacle avoidance using deep learning," in *Proc. 13th Latin American Robotics Symposium, 4th Brazilian Robotics Symposium (LARS/SBR)*. IEEE, 2016, pp. 7–12.

[10] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[11] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. 27th Advances in Neural Information Processing Systems (NIPS)*, 2014, pp. 3104–3112.

[12] I. D. Falco, G. D. Pietro, G. Sannino, U. Scafuri, E. Tarantino, A. D. Cioppa, and G. A. Trunfio, "Deep neural network hyper-parameter setting for classification of obstructive sleep apnea episodes," in *Proc. IEEE Symposium on Computers and Communications (ISCC)*, June 2018, pp. 01 187–01 192.

[13] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013, pp. 6645–6649.

[14] T. H. Vu and J.-C. Wang, "Acoustic scene and event recognition using recurrent neural networks," *Proc. Workshop on Detection and Classification of Acoustic Scenes and Events (DCASE)*, 2016.

[15] R. Nishihara, P. Moritz, S. Wang, A. Tumanov, W. Paul, J. Schleier-Smith, R. Liaw, M. Niknami, M. I. Jordan, and I. Stoica, "Real-time machine learning: The missing pieces," in *Proceedings of the 16th Workshop on Hot Topics in Operating Systems*. ACM, 2017, pp. 106–110.

[16] H. Kim, J. Park, J. Jang, and S. Yoon, "Deepspark: A spark-based distributed deep learning framework for commodity clusters," *arXiv preprint arXiv:1602.08191*, 2016.

[17] P. Moritz, R. Nishihara, I. Stoica, and M. I. Jordan, "Sparknet: Training deep networks in spark," *arXiv preprint arXiv:1511.06051*, 2015.

[18] Y. Wang, X. Qiu, D. Ding, Y. Zhang, Y. Wang, X. Jia, Y. Wan, Z. Li, J. Wang, S. Huang *et al.*, "Bigdl: A distributed deep learning framework for big data," *arXiv preprint arXiv:1804.05839*, 2018.

[19] A. Gupta, H. K. Thakur, R. Shrivastava, P. Kumar, and S. Nag, "A big data analysis framework using apache spark and deep learning," in *Proc. IEEE International Conference on Data Mining Workshops (ICDMW)*, 2017, pp. 9–16.

[20] L. Yang, J. Shi, B. Chern, and A. Feng, "Open Sourcing TensorFlowOnSpark: Distributed Deep Learning on Big-Data Clusters." [Online]. Available: http://yahoohadoop.tumblr.com/post/157196317141/open-sourcing-tensorflowonspark-distributed-deep

[21] A. Feng, J. Shi, and M. Jain, "CaffeOnSpark Open Sourced for Distributed Deep Learning on Big Data Clusters," accessed on: 01/28/2019. [Online]. Available: http://yahoohadoop.tumblr.com/post/139916563586/caffeonspark-open-sourced-for-distributed-deep

[22] X. Lu, H. Shi, R. Biswas, M. H. Javed, and D. K. Panda, "Dlobd: A comprehensive study of deep learning over big data stacks on HPC clusters," *IEEE Transactions on Multi-Scale Computing Systems*, 2018.

[23] H. Y. Ahn, H. Kim, and W. You, "Performance study of distributed big data analysis in yarn cluster," in *Proc. International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE, 2018, pp. 1261–1266.

[24] I. A. T. Hashem, I. Yaqoob, N. B. Anuar, S. Mokhtar, A. Gani, and S. U. Khan, "The rise of "big data" on cloud computing: Review and open research issues," *Journal of Information Systems*, vol. 47, pp. 98–115, 2015.

[25] Q. Zhang, L. T. Yang, Z. Chen, and P. Li, "A survey on deep learning for big data," *Journal of Information Fusion*, vol. 42, pp. 146–157, 2018.

[26] B. Pang and L. Lee, "Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales," in *Proc. 43rd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, 2005, pp. 115–124.

[27] J. Dai, X. Liu, and Z. Wang, "Building large-scale image feature extraction with bigdl at jd. com," Intel, Tech. Rep., Oct. 2017.

[28] R. Jain, *The art of computer systems performance analysis: techniques for experimental design, measurement, simulation, and modeling*. John Wiley & Sons, Inc., 1990.