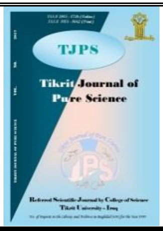
IASJ
المجلات الاكاديمية العراقية

Tikrit Journal of Pure Science
ISSN: 1813 – 1662 (Print) --- E-ISSN: 2415 – 1726 (Online)

Journal Homepage: http://tjps.tu.edu.iq/index.php/j



PERFORMANCE REFINEMENT OF CONVOLUTIONAL NEURAL NETWORK ARCHITECTURES FOR SOLVING BIG DATA PROBLEMS

Saud Aljaloud
College of Computer Science and Engineering, University of Ha'il, Ha'il, Saudi Arabia

https://doi.org/10.25130/tjps.v28i1.1270

ARTICLE INFO.

Article history:

-Received:	**18 / 11 / 2022**
-Received in revised form:	**10 / 12 / 2022**
-Accepted:	**9 / 1 / 2023**
-Final Proofreading:	**7 / 2 / 2023**
-Available online:	**20 / 2 / 2023**

Keywords: MNIST database, CIFAR10, GPU Bigdata, Deep learning, CNN, Theano and TensorFlow, MNIST database

Corresponding Author:
Name: Saud Aljaloud
E-mail: s.aljaloud@uoh.edu.sa
Tel:

ABSTRACT

The use of more examples than contrasted ones to compare neural network frameworks through using the MNIST database is considered a good research method. This is because this database is the subject of active research at the moment and has produced excellent results. However, in order to be trained and deliver accurate results, neural networks need a sizeable amount of sample data, as will be covered in more detail later. Because of this, big data experts frequently encounter problems of this nature. Therefore, two of the most well-liked neural network frameworks, Theano and TensorFlow, were compared in this study for how well they performed on a given problem. The MNIST database was used for this specific problem, represented by the recognition of handwritten digits from one to nine. As the project description implied, this study would not present a standard comparison because of this; instead, it would present a comparison of these networks' performance in a Big Data environment using distributed computing. The FMNIST or Fashion MNIST database and CIFAR10 were also tested (using the same neural network design), extending the scope of the comparison beyond MNIST. The same code was used with the same structure thanks to the use of a higher-level library called Keras, making use of the aforementioned support (in our case, Theano or TensorFlow). There has been a surge in open-source parallel GPU implementation research and development as a result of the high computational cost of training CNNs on large data sets. However, there are not many studies on assessing the performance traits of those implementations. In this study, these implementations were compared carefully across a wide range of parameter configurations, in addition to investigating potential performance bottlenecks, and identifying a number of areas that could use more fine-tuning.

I. Introduction

A large amount of data is collected by states, institutions, or individuals in order to draw meaningful conclusions from it and use it as needed. Data created by materials such as numbers, texts, expressions, figures, and graphics has been transferred to electronic media using computers in every field [5]. With computers, the internet, and related technologies becoming more prevalent in all aspects of life, the data produced by these technologies is also becoming more prevalent. The increasing prevalence of information technologies has changed people's living, working, and environmental conditions; places, professions, and employees have become "mobile," as have the devices they use.

However, in terms of diversity and volume, the resulting data has reached very different and large dimensions. The increase in mobility, the widespread use of social networks, the development of various tracking systems (sensors, barcodes, data matrix, RFID systems, etc.) technologies, the increase in the accessibility of communication technologies, the transfer of many business lines, particularly commercial transactions, to the electronic environment, and the diversity of data produced, as well as the speed and amount of collection, have all resulted in significant increases in the speed and amount of collection. This trend is expected to continue. While Machine-to-Machine Communication (M2M) is a technology that enables devices to be remotely monitored, managed, and communicated with each other via the sim card, sensors, electronic circuits, and internet network installed in the devices. It has a wide range of applications in the lives of both individuals and businesses. The use of these technologies in many fields, including vehicle tracking, medical automation, smart home appliances, metre reading, logistics, security, and agriculture, has necessitated the analysis of the data transmitted by the devices. The increasing prevalence of wireless sensors and the infinite number of objects that can be addressed with Internet Protocol Version 6 (IPv6) have led to an increase in the number of devices that will be connected to the Internet. According to the predictions of Cisco and IBM, 50 billion devices would be included in the internet network in 2020 [5]. Parallel to this development, M2M systems [6-10], in which a hardware is connected with a single application and today almost any hardware can be easily interconnected with various applications or devices, Internet of Things (IoT), Internet of Everything (IoE), Network of Things (WoT) and Network of Everything (WoE) have evolved into such environments. In these systems, where the real and virtual worlds are very close to each other and smart environments occur in every part of life, a huge volume of data is produced and most of this data is unstructured. These data, which can be in many types such as pictures, audio, text and video and transferred over networks, have also started to be stored in cloud environments. Another issue related to these data is that they have a variable and dynamic, in other words, flowing structure, especially human-sourced data, such as social media data. On the one hand, new data from the devices is included in the system or some data is interrupted, on the other hand, changes may occur in the existing data. The analysis of the collected data therefore becomes more complex. For this reason, the concept of "big data" has become very controversial, especially in recent years [11-13].

Convolutional neural networks (CNN) have lately demonstrated promising results in the area of image classification [1]. Most of these systems were dependent on specialized libraries or frameworks. In this article, Theano and TensorFlow, two well-liked CNN frameworks are examined [2]. The target challenge is to identify 1–9 handwritten digits. MNIST, a well-researched database, is employed for this task because it offers great chances to contrast several frameworks with actual data. The study also clarifies that CNN needs a lot of data to train well and generate accurate results. Often, by seeing such problems through the lens of big data, the solutions can be discovered [3]. A standardized comparison of the networks' performance in a Big Data setting using distributed computing is given, as implied by the document's title. The same code with the same structure can be reused using the higher-level library "Keras," which uses the mentioned capabilities (Theano or TensorFlow) of Ease of use [7-10].

The primary issue is comparing a CNN's performance in a classification challenge involving images from the MNIST database [7]. Keras is used as the backend, with either TensorFlow or Theano being compared. This CNN's effectiveness is measured in terms of assessment error and training time, allowing for future determination of the best backend in any given scenario. In addition, the CIFAR10 database and the Fashion MNIST (FMNIST) database, both of which contain images of ten different types of black and white clothing, are used for testing (10 colour images of dogs, birds, deer, cats, frogs, horses, boats, planes, cars and trucks). However, in order to speed up the overall process, the same CNN is trained on multiple cores (CPUs and GPUs) at the same time. The MNIST and FMNIST datasets are examined for the final scenario. According to [12], the primary goal is to compare the execution times of each backend to determine which is more efficient for distributed computing.

A. Research Methodology, Proposed Solution and Implementation

This section is divided into three parts. First, the common parts contained in the solution of all experiments carried out are analyzed. These parts are the neural network's architecture, the programming language used for its realization and the framework used. Then, the solution and the proposed implementation for each particular case (distributed and non-distributed) are discussed. The order of this section is shown below:

1. Network architecture, programming language and framework.
2. Solution and implementation of non-distributed case.
3. Solutions and implementation of distributed case.

B. Network Architecture, Programming Language And Framework

1) Programming Language

This project used Python and ReLu, the premier language for machine learning architectures.

2) Network Architecture

This project's main case, MNIST, was improved for a network architecture, which was then applied with additional databases. FMNIST and CIFAR10 were projected to perform poorly in terms of evaluating error analysis. The network's MNIST training error was within desirable margins for the chosen architecture, as provided in [4]. The network's layers were as follows:

- Convolutional layer of 30 features with a size of 5x5 pixels and with the activation function of rectification or ReLu.
- Max-pooling layer with a size of 2x2 pixels.
- Another convolutional layer but this time with 15 features, size 3x3 pixels and ReLu activation function.
- A layer with a maximum pooling size of 2x2 pixels.
- To prevent overtraining, the Dropout layer was in charge of excluding a specific number of neurons, in this case 20%.
- The following fully connected layers processed the data that was transformed into a vector by the Flatten layer.
- Layer with ReLu activation function and 128 fully connected neurons.
- Layer with ReLu activation function and 50 fully connected neurons.
- Output layer with 10 neurons (one for each class).

3) Framework

As mentioned in this paper, all experiments employed Keras with TensorFlow and Theano backends to compare performance.

C. Solution and Implementation of Non-Distributed Case

The experiment was launched on the farm with specifying to use either a GPU or a CPU; so, the farm's resource manager either reserved one for you or allocated a specific processor. To reduce execution time, the NVIDIA GeForce RTX 2080 Ti was the farm's most powerful GPU. After choosing the neural network model and processor, it was time to decide how to train and compare this model. To get a true picture of training times and evaluation error, ten experiments per instance were performed and the mean and variance were provided. The overall execution time included the connection with the farm and its management, system, and user times. The GPU execution time was taken into account. The model was trained for 200 epochs; however, validation values were received epoch by epoch to track precision, in addition to having error graph. Knowing which data would be compared, just the experiments—two for each database—remained (one for TensorFlow and one for Theano). The mean and variance of each group of ten experiments were displayed. These experiments are listed below:

1. MNIST used Theano and TensorFlow as the backends.
2. FMNIST used Theano and TensorFlow as the backends.
3. CIFAR10 used Theano and TensorFlow as the backends.

They were six from the prism of three tests (MNIST, FMNIST, and CIFAR10) with the two backend variations for comparison. Thus, only three cases of these six in groups of two were analysed for experimentation.

D. Distributed Case Solution and Implementation

The Elephas library converted Keras code into Spark with minimal instructions, allowing Spark's distributed case to keep the network's core and perform effective analysis [9]. The study used MNIST and FMNIST data and only four simulations per experiment. To provide variety, MNIST experiments were run on five machines or nodes with four cores each and four machines with five cores each. Starting with the last enumeration in the previous section, the experiments are listed as follows:

1. Performance comparison of MNIST that spread across five nodes with four cores per node and four nodes with five cores per node using the Theano and TensorFlow backends.
2. A performance comparison between the Theano and TensorFlow backends for FMNIST utilizing the distribution that produced the best results for MNIST.
3. The data from the undistributed case was utilized to establish how many epochs were necessary, and the number of epochs to be investigated was fixed.

II. Experiment and Analysis

The comparative results from each experiment were shown, followed by a succinct interpretation of each:

A. Evaluation of MNIST Performance on Both the Theano and TensorFlow Backends

The comparison figure of the times in this example is shown below:

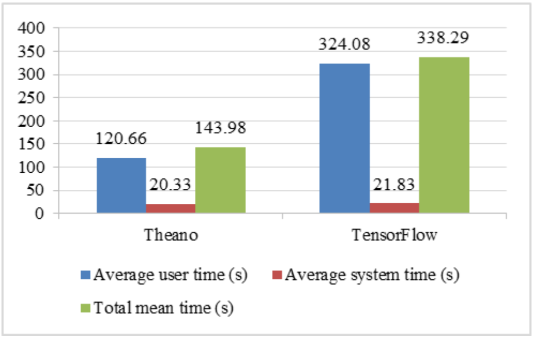


Fig. 1: Comparison of execution times for the case of non-distributed MNIST

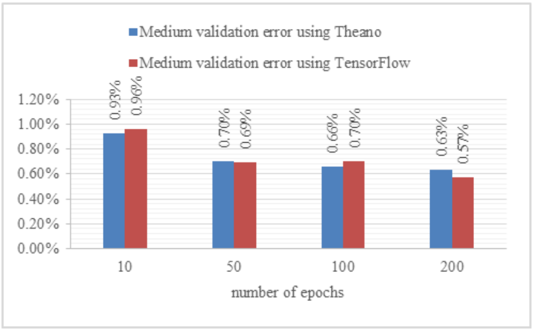


Fig. 2: Comparison of validation errors by epoch for the case of non-distributed MNIST

It was observed that Theano is clearly superior to TensorFlow in terms of timing. Now, it is time to show graphically the evolution of the mean of the error per period that they have had and its standard deviation (represented by the character *) in the cases of Theano and TensorFlow:

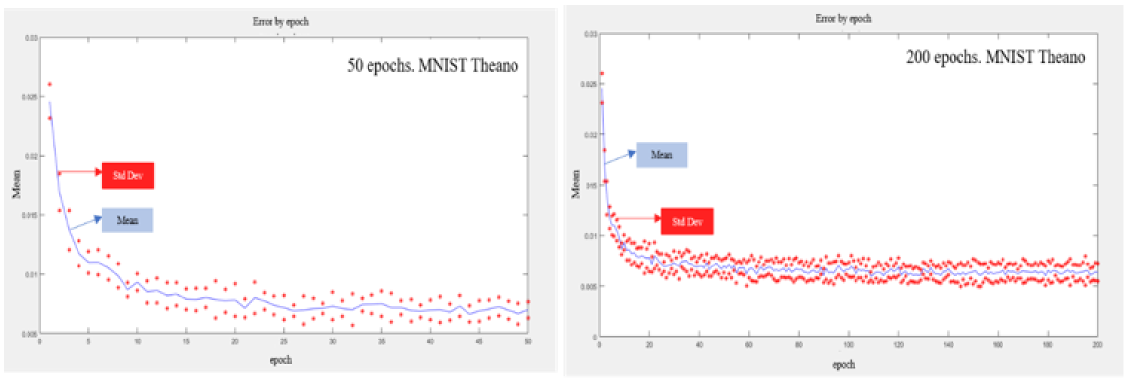


Fig.3: MNIST Theano, 50 and 200 epoch validation error mean and standard deviation

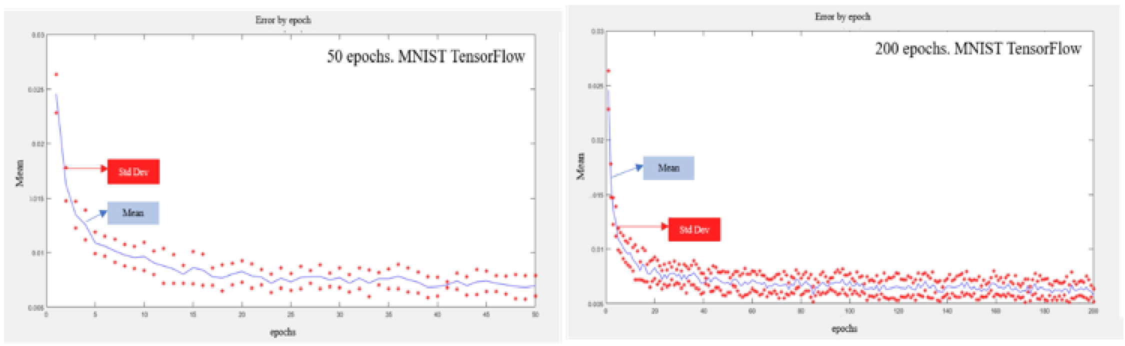


Fig. 4: MNIST TensorFlow, 50 and 200 epoch validation error mean and standard deviation

Finally, the two figures of this comparison were broken down into periods. In this case, it was clear that both Theano and TensorFlow have very similar features and a similar evolution. Both of them achieved a validation error below 1% from 10 epochs, a very optimal figure. In order to avoid overloading the document with plots, only the 50-epoch plots for Theano and TensorFlow were shown for the following experiments.

B. Evaluation of FMNIST Performance on Both the Theano and TensorFlow Backends

Now, the times obtained by training the FMNIST database are shown as follows:

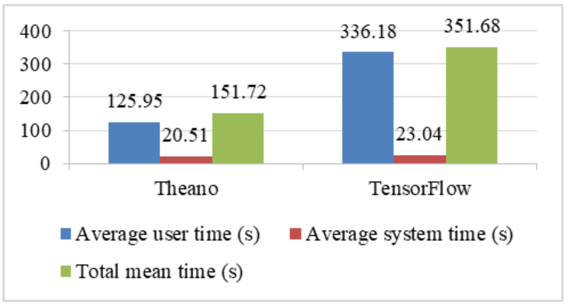


Fig. 5: Comparison of execution times for the case of non-distributed FMNIST

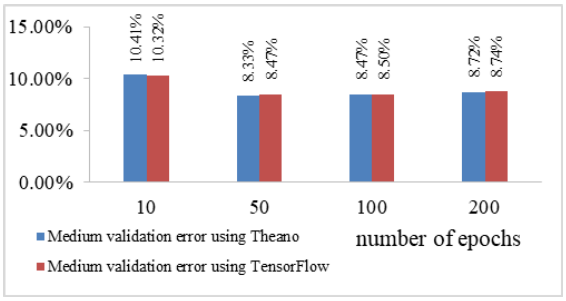


Fig. 6: Comparison of validation errors by epoch for the case of non-distributed FMNIST

As was the case with MNIST, it was observed that Theano is clearly superior in time. Now, it is time to present the graphs (only those of 50 epochs) and the comparative table in terms of validation errors:

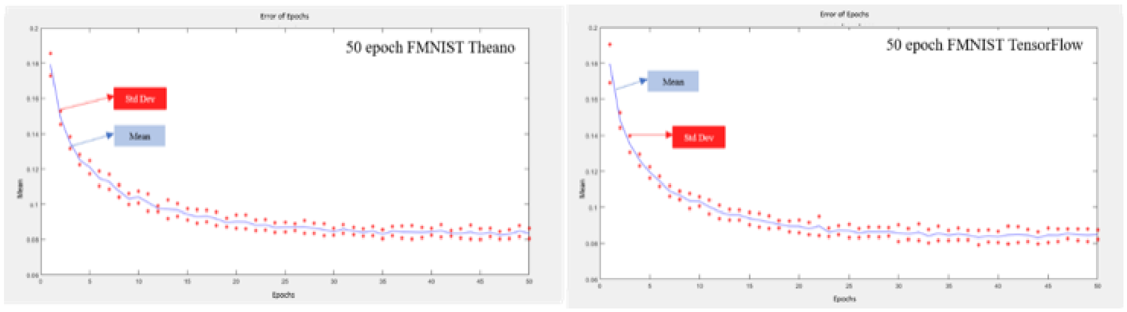


Fig. 7: The 50 epoch FMNIST Theano and TensorFlow validation error mean and standard deviation

Table 1: Comparison of the data obtained using distributed computing with the MNIST and FMNIST databases

	Setting	Backend	Total average time (minutes)	Average execution time per core (minutes)	Average handling time per task (ms)	Mean evaluation error
MNIST	5 nodes with 4 cores per node	TN*	50	41	80	0.74%
		TF**	8.29	6.95	81.25	0.75%
	4 nodes with 5 cores per node	TN*	57	40	87.75	0.70%
		TF**	9.31	7.18	83.5	0.74%
FMNIST		TN*	61.71	27.03	77.52	0.091

* Theano, ** TensorFlow

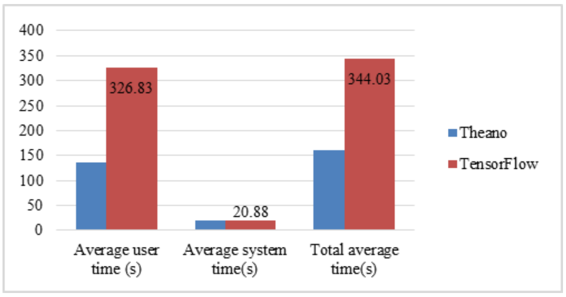


Fig. 8: Comparison of execution times for the case of CIFAR10 not distributed

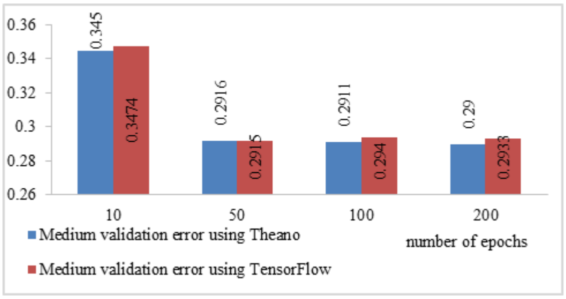


Fig. 9: Comparison of validation errors by epoch for the case of CIFAR10 not distributed

The faults were extremely identical between each backend, just like in MNIST; however, they were much more severe in this situation because the network was not designed for this database.

As illustrated in figure 8, the behavior of the previous cases was maintained in the scenario where CIFAR10 was not distributed. Figure 9 compares validation errors by epoch. A deep learning model's performance on the validation set was measured using a metric called validation loss. The dataset's validation set is a section set aside to check the model's efficacy. Similar to the training loss, the validation loss was determined by adding the errors for each example in the validation set.

The similarity between the backends was maintained; in this case, the network was much less optimized and the errors were much higher in general.

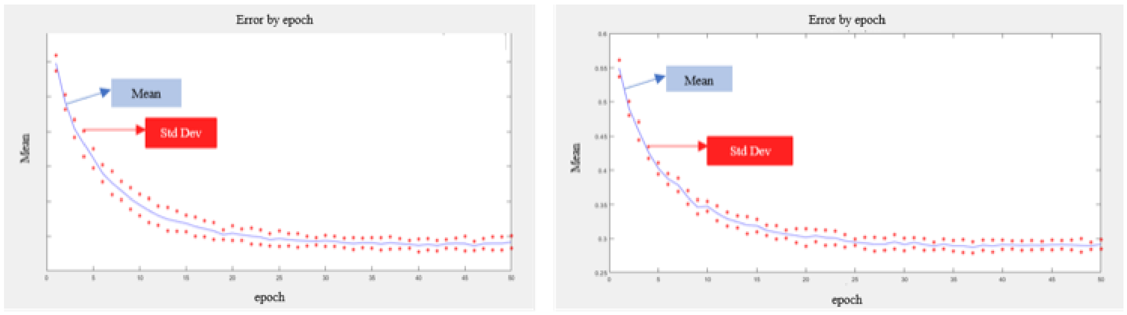


Fig. 10: Mean and standard deviation of the validation error in 50 epochs, CIFAR10 Theano (left) and CIFAR10 Tensor Flow (right)

C. Evaluation of the Performance of the Theano and TensorFlow Backends Using MNIST Distributed Across Five Nodes with Four Cores Each and Four Nodes with Five Cores Each

Starting experiments on GPUs was difficult, so they were started on CPUs [8]. Theano CPU execution can take up to an hour, so instead of ten simulations like in the non-distributed case, four were run. Finally, 50 epochs were ideal for training from the non-distributed case, so they were started immediately. The Spark interface reported how many tasks were executed, in which core and node, and how many total tasks were involved. It also provided task management, parallelization, and median execution times across tasks and cores. To avoid filling the document with images of all executions, images of the Spark interface from a Theano backend and TensorFlow backend execution was included, but the data representing the averages of all executions was compared. The data to compare were as follows:

- The length of time, on average, taken to complete the execution since Spark launched the process.
- The median execution times averaged across all tasks (since each task was executed in a core, it could be said that this time was also the mean per core). It was assumed that this data represented the average execution time per core even though it actually represented the average of medians.
- The management and distribution time allocated to each task in Spark was referred to as "Garbage collection time" or GC Time. The average of the medians that the experiment yielded was also calculated from this data, but it was used to determine the average management time per task.
- Finally, each case's mean evaluation error was calculated.

Theano's management time was low, but its execution times were much longer than TensorFlow's (see Table 1) (one was in milliseconds and the other in the order of minutes, respectively). Because they were similar to each other and the non-distributed case, the computational environment did not affect average evaluation errors. The FMNIST experiment used Spark with 5 nodes and 4 cores because it was slightly faster across all measured times.

D. Comparison of Performance Using the Distribution that Produced the Best Results for MNIST and FMNIST Using the Theano and TensorFlow Backends

This case was not taken into consideration because the neural network was not optimized for FMNIST and the probability of error data was not obtained with the network. TensorFlow and Theano simulations were run twice for each example to create a comparison table with results averages like the previous case. The previous experiment concluded that this example used five nodes with four cores each. Theano had much higher execution times than TensorFlow, but the evaluation error was very similar. Non-distributed case and system management and parallelization were negligible compared to execution times.

III. Conclusions and Future Work

Theano was more efficient in convolutional neural network training times than TensorFlow. Even though the error was very similar in both cases, despite the obvious loss of precision with the FMNIST database and even more so with CIFAR10 versus MNIST. Thus, Theano is a better backend for executions like those in this document. Since the error is constant after 50 epochs, more training epochs are not recommended. Since the distributed case was run on CPUs instead of GPUs, this setup's time savings cannot be compared. GPUs in both environments reduced resource management and parallelization time, which was much lower than execution time. Given the significant time difference between using Theano and TensorFlow as a backend, it was concluded that Theano is not well optimized to be launched in a distributed environment or over CPUs. In some cases, the MNIST execution examples had performed better with a configuration of five machines with four cores each rather than four machines with five cores each. Evaluation error had remained constant for Theano and TensorFlow in case of four-core machines. Evaluation errors were similar in distributed and non-distributed executions.

This document demonstrated that Theano is not very well optimized when using distributed computing through Spark on several CPUs. It may be worthwhile to conduct further research to determine whether this is because resources are distributed unevenly or because Theano is being run on CPUs rather than GPUs.

References

[1] Jmour, N., Zayen S., & Abdelkrim, A. (2018) Convolutional neural networks for image classification. *The 2018 International Conference on Advanced Systems and Electric Technologies (IC_ASET)*, pp. 397-402, Electronic ISBN:978-1-5386-4449-2, DOI: 10.1109/ASET.2018.8379889

[2] Shatnawi, A., Al-Bdour, G., Al-Qurran, R. & Al-Ayyoub, M. (2018). A comparative study of open source deep learning frameworks. *The 2018 9th International Conference on Information and Communication Systems (ICICS)*, pp. 72-77, doi: 10.1109/IACS.2018.8355444. Electronic ISBN:978-1-5386-4366-2

[3] Yu, L., Li, B., & Jiao, B. (2019). Research and implementation of CNN based on TensorFlow. *IOP Conference Series: Materials Science and Engineering*. 490. 042022. 10.1088/1757-899X/490/4/042022.

[4] Elshawi, R., Wahab, A., Barnawi, A., & Sakr, S. (2021). DLBench: A comprehensive

experimental evaluation of deep learning frameworks. *Cluster Computing*, *24*, 2017–2038. https://doi.org/10.1007/s10586-021-03240-4

[5] Karaman, D., Gözüacik, N., Alagöz, M. O., İlhan, H., Çağal, U., & Yavuz, O. (2015). Managing 6LoWPAN sensors with CoAP on Internet. *The 23nd Signal Processing and Communications Applications Conference (SIU), IEEE*, Malatya, Turkey, 1389-1392, doi: 10.1109/SIU.2015.7130101.

[6] Yapıcı, M., Tekerek, A., & Topaloglu, N. (2019). Performance comparison of convolutional neural network models on GPU. *IEEE 13th International Conference on Application of Information and Communication Technologies (AICT)*, Baku, Azerbaijan, 1-4, doi: 10.1109/AICT47866.2019.8981749

[7] Khan, A., Sohail, A., Zahoora, U., & Qureshi, A. (2020). A survey of the recent architectures of deep convolutional neural networks. *Artificial Intelligence Review,53,* 5455–5516. https://doi.org/10.1007/s10462-020-09825-6.

[8] Rahman, M. M., Islam, M. S., Sassi, R., & Aktaruzzaman, M. (2019). Convolutional neural networks performance comparison for handwritten Bengali numerals recognition. *SN Applied Sciences,1,* 1660, https://doi.org/10.1007/s42452-019-1682-y

[9] Rahman, N. R., Hasan, M. A. M., & Shin, J. (2020). Performance comparison of different convolutional neural network architectures for plant seedling classification. *The 2nd International Conference on Advanced Information and Communication Technology (ICAICT)*, Page 146-150.

[10] Prilianti, K. R., Brotosudarmo, T. H. P., Anam, S., Suryanto, A. (2019). Performance comparison of the convolutional neural network optimizer for photosynthetic pigments prediction on plant digital image. *AIP Conference Proceedings*, *2084*(1), https://doi.org/10.1063/1.5094284

[11] Tan, Y., Li, Y., Liu, H., Lu, W., & Xiao, X. (2020). Performance comparison of data classification based on modern convolutional neural network architectures. *The 39th Chinese Control Conference (CCC)*, Shenyang, China, 815-818, doi: 10.23919/CCC50068.2020.9189237.

[12] Gambo, F. L., Wajiga, G. M., Shuib, L., Garba, E. J., Abdullahi, A. A., Bisandu, D. B. (2021). Performance comparison of convolutional and multiclass neural network for learning style detection from facial images. *EAI Endorsed Transactions on Scalable Information Systems*, *9*(35), 1-13, doi: 10.4108/eai.20-10-2021.171549

[13] Ghafoorian, M., Karssemeijer, N., Heskes, T., van Uden, I. W. M., Sanchez, C. I., Litjens, G., ...& Platel, B. (2017). Location sensitive deep convolutional neural networks for segmentation of white matter hyperintensities. *Scientific Reports*, *7*, 5110. https://doi.org/10.1038/s41598-017-05300-5

تحسين أداء هياكل الشبكات العصبونية الإلتفافية لحل مشكلات البيانات الضخمة

الملخص

يعتبر إستخدام أكثر من مثال للإختبارت بدلا من إستخدام طريقة مقارنة أطر الشبكات العصبونية الإلتفافية من خلال استخدام قاعدة بيانات MNIST طريقة بحثية جيدة. وذلك لأن قاعدة البيانات MNIST هي موضوع بحث نشط في الوقت الحالي وقد أسفرت عن نتائج ممتازة. ومع ذلك، من أجل التدريب وتقديم نتائج دقيقة، تحتاج الشبكات العصبونية الإلتفافية إلى قدر كبير من البيانات لغرض الإختبار، حيث سيتم تناولها بمزيد من التفاصيل لاحقًا. لهذا السبب، كثيرًا ما يواجه خبراء البيانات الضخمة مشاكل من هذا النوع. لذلك، تمت مقارنة اثنين من أكثر أطر الشبكات العصبونية الالتفافية شهرة، Theano و TensorFlow ، في هذه الواسة لمدى جودة أدائها في مشكلة معينة. تم استخدام قاعدة بيانات MNIST لهذه المشكلة المحددة، ممثلة في التعرف على الأرقام المكتوبة بخط اليد من واحد إلى تسعة. وكما يوحي وصف المشروع، فإن هذه الواسة لن نقدم فقط مقارنة قياسية لهذا السبب تحديداً، بدلًا من ذلك، ستقدم مقارنة بين أداء هذه الشبكات في بيئة البيانات الضخمة باستخدام الحوسبة الموزعة Distributed Computing. سيتم أيضًا اختبار قاعدة بيانات FMNIST أو Fashion MNIST و CIFAR10 (باستخدام نفس تصميم الشبكات العصبونية الإلتفافية)، مما يوسع نطاق المقارنة إلى ما بعد MNIST. سيتم إستخدام نفس الخوارزمية مع نفس الهيكل وكل هذا بفضل إستخدام مكتبة ذات مستوى أعلى تسمى Keras،والتي تستخدم الدعم المذكور أعلاه (في حالتنا، Theano أو TensorFlow). أصبح هناك زيادة في البحوث والتطوير في مجال تطبيق GPU المتوزي مفتوح المصدر نتيجة للتكلفة الحسابية العالية لتدريب شبكات CNN على مجموعات البيانات الكبيرة. ومع ذلك، لا توجد العديد من الواسات عن تقييم سمات أداء تلك التطبيقات. في هذه الواسة، تمت مقارنة هذه التطبيقات بعناية عبر مجموعة واسعة من المتغيرات، بالإضافة الى دراسة الحالات المحتمل فيها التأثير على الاداء، وتحديد عددًا من المجالات التي يمكن أن يحسن فيها الأداء.