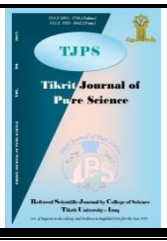
IASJ المجلات الاكاديمية العراقية

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 / 12 / 2022
-Accepted: 9 / 1 / 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

Two of the most well-liked neural network frameworks, Theano and TensorFlow, will be compared in this study for how well they perform on a given problem. The MNIST database will be used for this specific problem, which is the recognition of handwritten digits from one to nine. It is a good idea to use more examples than contrasted ones to compare these frameworks 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. As the project description implies, we won't just present a standard comparison because of this; instead, we'll work to present a comparison of these networks' performance in a Big Data environment using distributed computing. The FMNIST or Fashion MNIST database and CIFAR10 will also be tested (using the same neural network design), extending the scope of the comparison beyond MNIST. The same code will be used with the same structure thanks to the use of a higher-level library called Keras, which makes 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 aren't many studies that have been done to assess the performance traits of those implementations. In this study, we compare these implementations carefully across a wide range of parameter configurations, look into potential performance bottlenecks, and pinpoint 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. Machine-to-Machine Communication (M2M), on the other hand, 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, and 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 fact that the number of objects that can be addressed with Internet Protocol Version 6 (IPv6) has become almost infinite, has led to an increase in the number of devices that will be connected to the Internet, and according to the predictions of Cisco and IBM, 50 billion devices will 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, 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, 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", that is, "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, we'll examine Theano and TensorFlow, two well-liked CNN frameworks.[2] The target challenge is to identify 1–9 handwritten digits. MNIST, a well-researched database, will be employed for this task because it offers great chances to contrast several frameworks with actual data. We'll also learn 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 will be 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).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 will be used as the backend, with either TensorFlow or Theano being compared. This CNN's effectiveness will be 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, will be used for testing (10 colour images: dogs, birds, deer, cats, frogs, horses, boats, planes, cars and trucks). However, in order to speed up the overall process, we will train the same CNN on multiple cores (CPUs and GPUs) at the same time. We will look at the MNIST and FMNIST datasets for our final scenario. According to [12], the primary goal will be to compare the execution times of each backend to determine which is more efficient for distributed computing.

A. Research methodology & Proposed solution and implementation

We are going to divide this chapter into three parts, first they will analyse the common parts that the solution of all the experiments carried out will have. 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). The order of this section is shown below:

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

B. Network architecture, programming language and framework

1) Programming language

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

2) Network Architecture

This project's main case, MNIST, will be improved for a network architecture, which will then be applied with additional databases. FMNIST and CIFAR10 are projected to perform poorly in terms of evaluation

error analysis. The network's MNIST training error is within desirable margins for the chosen architecture. The source [4] provided it. Below are this network's layers.

- Convolutional layer of 30 features with 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 Dropour layer is in charge of excluding a specific number of neurons, in this case 20%.
- The following fully connected layers process the data that is 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 will employ Keras with TensorFlow and Theano backends to compare performance.

C. Solution and implementation of non-distributed case

Launch the experiment on the farm and specify whether want to use a GPU or a CPU so the farm's resource manager reserves one for you or allocate a specific processor. To reduce execution time, the NVIDIA GeForce RTX 2080 Ti is the farm's most powerful GPU. After choosing the neural network model and processor, it's time to decide how to train and compare it. To get a true picture of training times and evaluation error, ten experiments per instance and the mean and variance provided will be done. they. The overall execution time includes the connection with the farm and its management, system, and user times. We care about GPU execution time. The model will be trained for 200 epochs; however, we will receive validation values epoch by epoch to track precision. Besides an epoch-by-epoch validation error graph. Knowing which data would be compared, just the experiments—two for each database—remain (one for TensorFlow and one for Theano). The mean and variance of each group of ten experiments will be displayed.

These experiments are listed below:

1. MNIST is using Theano and TensorFlow as the backend.
2. FMNIST is using Theano and TensorFlow as the backend.
3. CIFAR10 is using Theano and TensorFlow as the backend.

They are 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 will be analysed for experimentation.

D. Distributed case solution and implementation

The Elephas library converts Keras code to Spark with minimal instructions, allowing Spark's distributed case to keep the network's core and perform effective analysis [9]. The study will use MNIST and FMNIST data and only four simulations per experiment. To provide variety, MNIST experiments will be 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 will be listed as follows:

1. Performance comparison of MNIST 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 utilising the distribution that produced the best results for MNIST.
3. The data from the undistributed case will be utilised to establish how many epochs are necessary, and the number of epochs to be investigated will be fixed.

II. Experiment and analysis

The comparative results from each experiment will now be 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.:

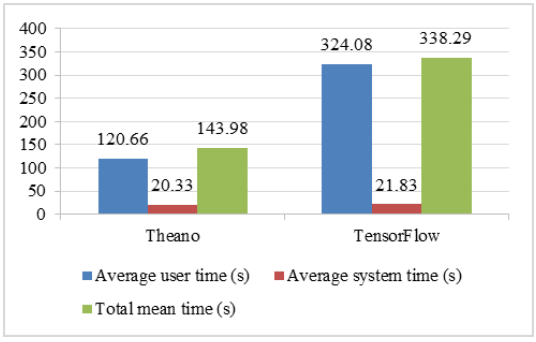


Fig. 1: Comparison of execution times for the case of non-distributed MNIST.

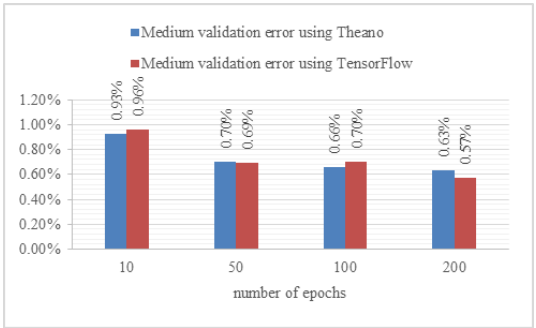


Fig. 2: Comparison of validation errors by epoch for the case of non-distributed MNIST.

It can be seen that Theano is clearly superior to TensorFlow in terms of timing. Now we will 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 case of Theano and in the case of TensorFlow:

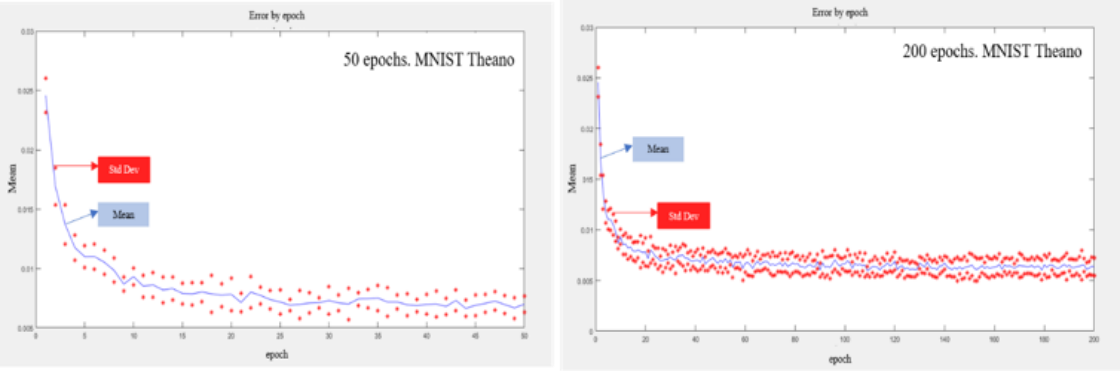


Fig.3: MNIST Theano, 50 and 200 epoch validation error mean and standard deviation.

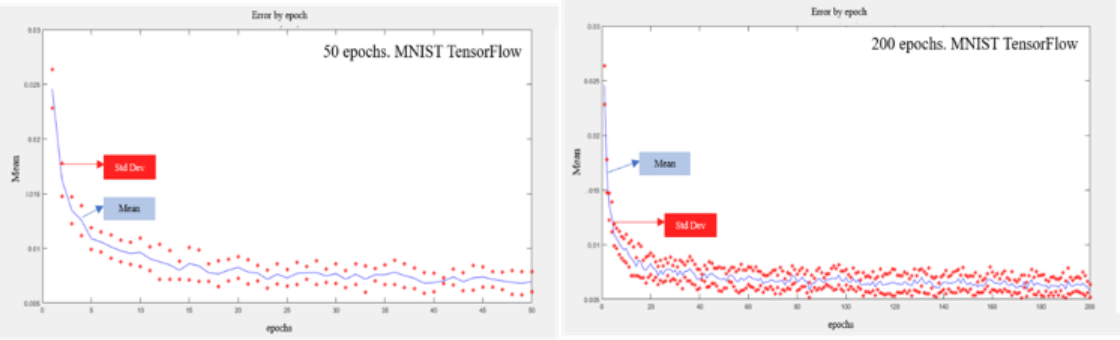


Fig. 4: MNIST TensorFlow, 50 and 200 epoch validation error mean and standard deviation.

Finally, a figure 2 of this comparison is broken down into periods: In this case, it can be seen that both Theano and TensorFlow have very similar features and a similar evolution. Both achieve 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 will be shown for the following experiments.

B. Evaluation of FMNIST performance on both the Theano and TensorFlow backendsS

Now we show the times obtained by training the FMNIST database:

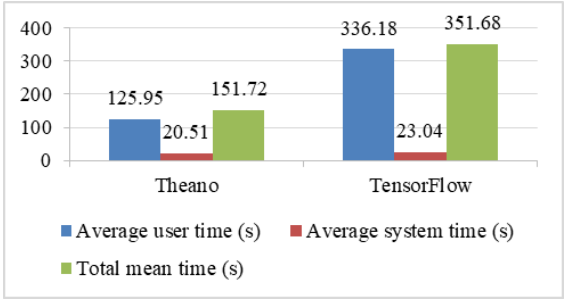


Fig. 5: Comparison of execution times for the case of non-distributed FMNIST.

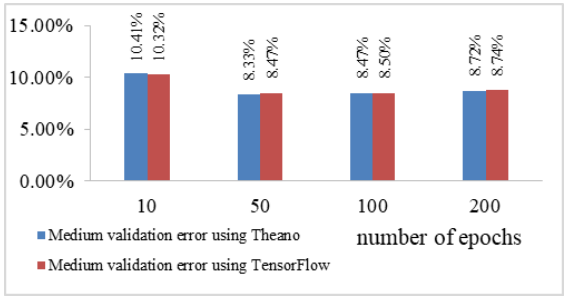


Fig. 6: Comparison of validation errors by epoch for the case of non-distributed FMNIST.

As was the case with MNIST, it can be seen that Theano is clearly superior in time. Now let's see the graphs (only those of 50 epochs) and the comparative table in terms of validation errors:

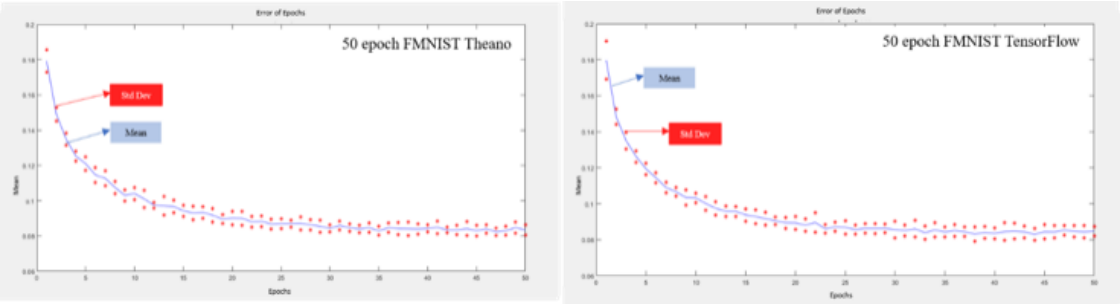


Fig. 7: The 50 epoch FMNIST Theano and TensorFlow validation error mean and standard deviation.

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

	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

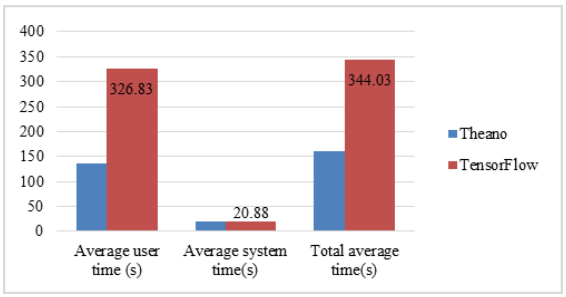


Fig. 8: Comparison of execution times for the case of CIFAR10 not distributed.

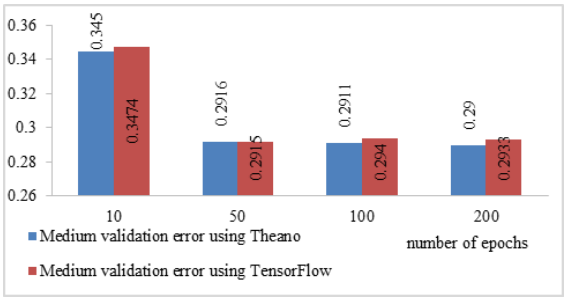


Fig. 9: Comparison of validation errors by epoch for the case of CIFAR10 not distributed.

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

As can be seen, the behavior of the previous cases is maintained. In the scenario where CIFAR10 is not distributed, Figure 9 compares validation errors by epoch. A deep learning model's performance on the validation set is 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 is determined by adding the errors for each example in the validation set.

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

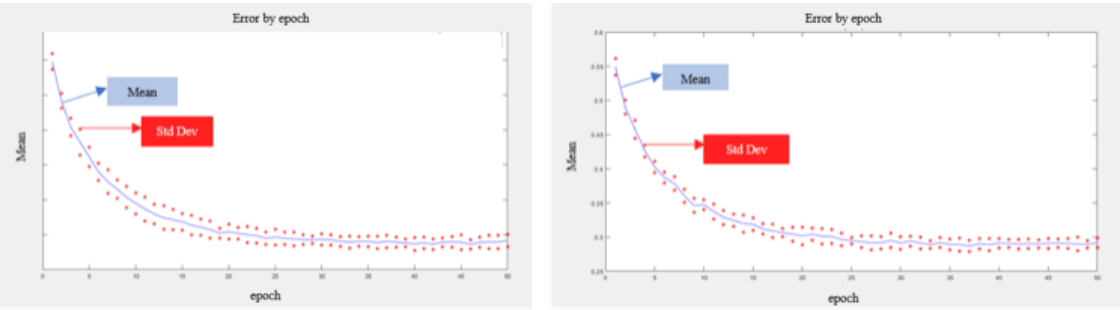


Fig. 10: Mean and standard deviation of the validation error in 50 epochs. CIFAR10 Theano (left) and CIFAR10 Tensor Flow. (right)

C. Evaluation of the performance of the Theano and TensorFlow backends using MNIST distributed across five nodes with four cores each and four 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 are ideal for training from the non-distributed case, so they will be started immediately. The Spark interface reports how many tasks were executed, in which core and node, and how many total tasks were involved. It also provides 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 will be included, but the data representing the averages of all executions will be compared. The data to compare will be:

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

Theano's management time is low, but its execution times are much longer than TensorFlow's (Table 1). (one is in milliseconds and the other in the order of minutes, respectively). Because they are similar to each other and the non-distributed case, the computational environment does not affect average evaluation errors. The FMNIST experiment will use Spark with 5 nodes and 4 cores because it is 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 backend and TensorFlow backend

We will not go into this case because the neural network isn't optimized for FMNIST and the probability of error data isn't 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 has much higher execution times than TensorFlow, but the evaluation error is very similar. non-distributed case and system management and parallelization are negligible compared to execution times.

III. CONCLUSIONS AND FUTURE WORK

Theano is more efficient in convolutional neural network training times than TensorFlow, even though the error is 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 we ran the distributed case on CPUs instead of GPUs, we can't compare this setup's time savings. GPUs in both environments reduce resource management and parallelization time, which is much lower than execution time. Given the significant time difference between using Theano and TensorFlow as a backend, we can conclude that Theano is not well optimised to launch in a distributed environment or over CPUs. In some cases, the MNIST execution examples have performed better with a configuration of five machines with four cores each rather than four machines with five cores each. Evaluation error has remained constant for Theano and TensorFlow, four-core machines. Evaluation errors are similar in distributed and non-distributed executions.

It has been demonstrated in this document that Theano is not very well optimised 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. and Abdelkrim, A. (2018) "Convolutional neural networks for image classification," *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. and Al-Ayyoub, M. (2018) "A comparative study of open source deep learning frameworks," *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, Liang & Li, Binbin & Jiao, Bin. (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]. El Shawi, Radwa & Wahab, Abdul & Barnawi, Ahmed & Sakr, Sherif. (2021). DLBench: a comprehensive experimental evaluation of deep

learning frameworks. Cluster Computing. Vol. 24. Page 1-22. 10.1007/s10586-021-03240-4.
[5]. Karaman, D., Gözüacik, N., Alagöz, M. O., İlhan, H., Çağal, U., and Yavuz, O. (2015). Managing 6LoWPAN Sensors with CoAP on Internet. 23st Signal Processing and Communications Applications Conference (SIU), IEEE, DOI: 10.1109/SIU.2015.7130101, Malatya, Turkey.
[6]. Yapıcı, Mutlu & Tekerek, Adem & Topaloglu, Nurettin. (2019). Performance Comparison of Convolutional Neural Network Models on GPU. IEEE 13th International Conference on Application of Information and Communication Technologies (AICT), Page 1-4. Doi:10.1109/ AICT47866. 2019.8981749.
[7]. Khan, Asifullah & Sohail, Anabia & Zahoora, Umme & Saeed, Aqsa. (2020). A Survey of the Recent Architectures of Deep Convolutional Neural Networks. *Artif Intell Rev*, Vol. **53**, page 5455–5516. 10.1007/s10462-020-09825-6.
[8]. Rahman, Md Moklesur & Islam, Md & Sassi, Roberto & Aktaruzzaman, Md. (2019). Convolutional neural networks performance comparison for handwritten Bengali numerals recognition. *SN Applied Sciences*. **1.** 1660, 10.1007/s42452-019-1682-y.
[9]. Rahman, Nawmee Razia et al. (2020), "Performance Comparison of Different Convolutional Neural Network Architectures for Plant Seedling Classification." *2020 2nd International Conference on Advanced Information and Communication Technology (ICAICT)*, Page 146-150..
[10]. Prilianti, Kestrilia & Brotosudarmo, Tatas & Anam, Syaiful & Suryanto, Agus. (2019). Performance comparison of the convolutional neural network optimizer for photosynthetic pigments prediction on plant digital image. *AIP Conference Proceedings*. Volume 2084, Issue 1, 10.1063/1.5094284
[11]. Tan, Yuchen & Li, Yanxiang & Liu, Han & Lu, Wang & Xiao, Xiaowu. (2020). "Performance Comparison of Data Classification based on Modern Convolutional Neural Network Architectures," *2020 39th Chinese Control Conference (CCC)*, 2020, pp. 815-818, doi: 10.23919/CCC50068.2020.9189237.
[12]. Gambo, Farouk & Wajiga, Gregory & Shuib, L & Garba, Etemi & Abdullahi, A & Bisandu, Desmond. (2021). Performance Comparison of Convolutional and Multiclass Neural Network for Learning Style Detection from Facial Images. ICST Transactions on Scalable Information Systems. 19. 1-13. 10.4108/eai.20-10-2021.171549.
[13]. Ghafoorian, Mohsen & Karssemeijer, Nico & Heskes, Tom & Van Uden, Inge & Sanchez, Clara & Litjens, Geert & Leeuw, Frank-Erik & Ginneken, Bram & Marchiori, Elena & Platel, Bram. (2016). Location Sensitive Deep Convolutional Neural Networks for Segmentation of White Matter Hyperintensities. Scientific Reports. 7. 10.1038/s41598-017-05300-5.

تحسين أداء هياكل الشبكات العصبونية الإلتفافية لحل مشكلات البيانات الضخمة

الملخص

في هذه الدراسة ستتم مقارنة أثنين من أكثر أطر الشبكات العصبونية الإلتفافية شهرة وهما Theano و TensorFlow، لمدى جودة أدائهما في حالات معينة. سيتم إستخدام قاعدة بيانات MNIST لهذه الحالة المحددة في هذه الدراسة، وهي التعرف على الأرقام المكتوبة بخط اليد من واحد إلى تسعة. من الأفظل إستخدام أكثر من مثال للإختبارت بدلا من إستخدام طريقة المقارنة, 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. كما يوحي وصف المشروع، لن نقدم فقط مقارنة قياسية لهذا السبب تحديداً، بدلاً من ذلك، سنعمل على تقديم 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 المتوازي مفتوح المصدر نتيجة للتكلفة الحسابية العالية لتدريب شبكات CNN على مجموعات البيانات الكبيرة. ومع ذلك، لا توجد العديد من الدراسات التي تم إجراؤها لتقييم سمات أداء تلك التطبيقات. في هذه الدراسة، نقارن هذه التطبيقات بعناية عبر مجموعة واسعة من المتغيرات ، وننظر كذلك في الحالات المحتمل فيها التأثير على الاداء، وفي النهاية نعطي عددًا من المجالات التي يمكن أن يحسن فيها الأداء.