Comparison of Existing Deep Learning Hardware Accelerators/Processors

*Note: Sub-titles are not captured in Xplore and should not be used

1st Charles Arsenal Okere
Department of Electronic Engineering
Hamm-Lippstadt University Of Applied Science
Lippstadt, Germany
charles-arsenal.okere@stud.hshl.de

***Abstract*—Modern data analysis or analysis problems frequently use deep learning approaches. Modern machine learning technologies were once capable of handling a wide range of situations, but today we need more powerful, efficient, and optimized solutions to this issue. Not only have algorithms changed to fit this era, but hardware also needs to be updated in order to address the optimization issue. Originally, a standard CPU handled the processing, but as time went on, CPUs grew more powerful, evolved into GPUs, and are now also being used with TPU. In order to understand and demonstrate all of the aspects, we created a straightforward pipeline to compare a deep learning model's inception V3 on various accelerators. We compared the data training and task completion times to understand and demonstrate each phase, and we found that TPU outperformed the competition with the lowest execution time on model training of just 1 minute 53 seconds, while the other accelerators couldn't even come close to TPU's time.**

INTRODUCTION

There is a wide range of differences in the work modules and aims of various accelerators discussed in the context of machine learning, computer vision, deep learning, pattern recognition, and automation processes [1]. This period has seen a meteoric rise in the volume of data being created and stored digitally. Information from many fields, including economics, social media, gaming, and more, is collected and analyzed in this way before being made available to users. Applications that require a high rate of processing are classified as throughput computing workloads. It has been hypothesized that CPUs with numerous data processing cores would be a good fit for such tasks [2]. CPUs are designed to perform tasks in parallel with low latency, while GPUs are designed to process data in parallel with high throughput. In recent years, graphics processing units (GPUs) have also been employed for non-graphics-related tasks. Accelerating advancements in graphics hardware have found widespread use in industries as diverse as science and business. A plethora of works [3], [4] report significant velocities in a variety of fields. By exploiting special circumstances in graphics application programming interfaces, GPUs were first put to use for calculations other than graphics. In order to operate data through the graphics pipeline, programmers must carefully map program data to the allocated shader buffer memory. General-purpose programming had limited hardware support, but with the right amount of work, it was possible to get big speedups [4].

Modern deep learning (DL) operations have high computing requirements, yet the exponential growth of computer capacity is slowing down at the same time. Due to model size, service latency, and energy restrictions, DL model implementation provides significant challenges for many uses. To speed up progress in this area, it is essential to design algorithms, accumulators simultaneously, and equipment to improve performance, power consumption, and efficiency at both the system and algorithm levels [5]. To advance DL systems, there are typically three different types of engineers involved: (1) data researchers and technologists who use and create deep learning algorithms in conjunction with domain specialists like healthcare, economic, or climate researchers; (2) hardware designers who create cutting-edge hardware technology to speed up the elements in Deep Learning; and (3) engineers who considerably improve software's performance so that it operates more efficiently on a particular piece of hardware [6]. Future hardware designers must to be familiar with the traits and components of the academic and commercial models that are typically employed. When creating models, data scientists should consider the deployment platform's restrictions. Performance engineers should back up changes that improve models, frameworks, and hardware platforms [7].

As opposed to their fewer complex counterparts, deep neural networks are multi-layered artificial neural networks. These networks are used in numerous experimental and commercial applications, such as object identification, discernment and segmentation, image processing, audio recognition, bioinformatics, natural-language processing, and medical drug discovery, among others [8]. Smart applications must increasingly be distributed across a wide range of hardware, including embedded devices, autonomous vehicles, and the internet. The variety of hardware on these devices, including incorporated CPUs, GPUs, FPGAs, and ASICs, makes it difficult to map deep learning tasks to them [9]. TensorFlow and PyTorch,

Identify applicable funding agency here. If none, delete this.

two contemporary deep learning methodologies, aim to offer optimizations like auto differentiation and dynamic memory management. Systems that use computation and graphs include Caffe, MXNet, and others [10]. In addition, it currently takes a lot of engineering work to support various hardware back-ends in different DL frameworks. Frameworks must decide between utilizing non-optimized solutions of these entrants, even for supported back-ends, or (1) eliminating graph optimizations that produce new operators that are not in the selected operational library [11].

BACKGROUND STUDY

A small number of non-graphics tasks ran successfully on GPUs on the GPUs, prompting manufacturers to include dedicated hardware and software. To the contrary, AI and, more especially, machine learning (ML), have seen enormous success over the past few decades and have had an ever-increasing impact on our lives. The same is true for the IoT: both the quantity and variety of linked devices are expanding at a lightning pace [12]. despite the fact that algorithms for signal processing, image processing, and Synthetic Aperture Radar imaging are utilized every day. The sheer volume and complexity of the data makes near-real-time processing unfeasible. Many image processing methods are naturally parallel, making them well-suited to multi-core CPU parallel processing and the Graphics Processing Unit (GPU) parallel architecture [13]. This is where the Tensor Processing Unit (TPU) comes in, as it was designed to reduce the time and data complexity issues. The TPU has reached a major milestone in appropriately and concurrently holding these duties with the CPU and GPU [9]. This is in response to image analysis, experimental execution of scientific methodologies, and a high-priority effort to create specialized chips for DNNs. Hardware accelerators have also made deep neural network (DNN) inference faster by several orders of magnitude. Among these DNN accelerators, Google's Tensor Processing Unit (TPU) has proven to be the best-in-class, speeding up by more than 15x compared to modern GPUs [14].

Machine learning (ML) is a branch of AI that refers to a set of computational algorithms and techniques for teaching computers to make decisions without being explicitly programmed with a specific goal in mind. These methods can now be used with particle accelerators, and we expect them to become an ever more useful tool for meeting new requirements for beam energy, brightness, and stability [15]. Research into deep neural networks (DNNs) has made great strides in recent years, especially when contrasted to the use of more conventional algorithms. Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) are only two of the many network models that have been developed for use in the study of image, video, and audio processing [16]. However, as can be seen in the estimation of model parameters, the modeling of neural networks is becoming increasingly complex and cumbersome. Although many academics are making use of existing work on GPU platforms to boost computing performance, specialized hardware solutions are needed and are beginning to offer advantages over purely software approaches [17]. Recent developments and the seeming birth of deep learning owe a great deal to advancements in graphics processing units (GPUs). The convergence of three factors—a scalable learning algorithm that enhances performance as models and training data become more significant, the reasonably priced accessibility of vast quantities of training data via the Internet, and the sheer compute needed to train and implement extensive networks—made deep learning possible [18] [18]—converged to create this revolutionary field. As previously mentioned, the primary objective of any DNN accelerator's design is to minimize the data transfer size and distance. Realizing this, a significant amount of work in digital accelerators has focused on enhancing the calculations that power memory-light DNN models such as CONV-nets [19].

In most recent talks about digital accelerators, only forward inference accelerators for CONV-nets have been talked about. It has come to our attention that numerous academics have worked on various projects, including deep learning hardware accelerators and processors, which hinders our capacity to provide accurate comparisons between the various accelerators in the field. This section contains a list of all the tests that have already been completed. Deep learning accelerators are now vulnerable to attacks focused on acquiring access to the computer, which is a new security concern. Therefore, memory encryption is required for smart objects with DL accelerators in order to protect NN models [20]. Neural network inference implementation is also notoriously difficult due to its high storage and computational complexity. As a result, CPU systems find it harder to offer sufficient computational capacity. Graphics processing unit (GPU) systems are the preferred way out for convolutional network operations due to their robust analytical, and computational capabilities also user-friendly programming frameworks [21]. An FPGA's flexible, low-latency architecture allows deep learning acceleration while consuming little power for high-level hardware synthesis problems. On the other hand, the popularity of ML inference applications has sparked the development of specialized accelerators that can increase the speed and effectiveness of ML processes. Because there are so many different ML workloads and only a limited number of modeling techniques that can be used together [22], it can be hard to figure out which accelerator is best for a given task.

In addition, numerous DNN inference accelerators [23]–[25] with varying features for DNN algorithms have been developed. In spite of this, all DNN accelerators share two qualities that are employed in their design: (1) Each feature map's MAC operations can be computed in parallel because of its relatively sparse dependencies, and (2) Each feature map's data has strong temporal and geographic localities that make it possible to store and reuse it wisely. DNN accelerators use spatial designs made up of numerous distinct massively parallelization engines (PEs), each of which calculates MACs, to take advantage of the first property. [26].

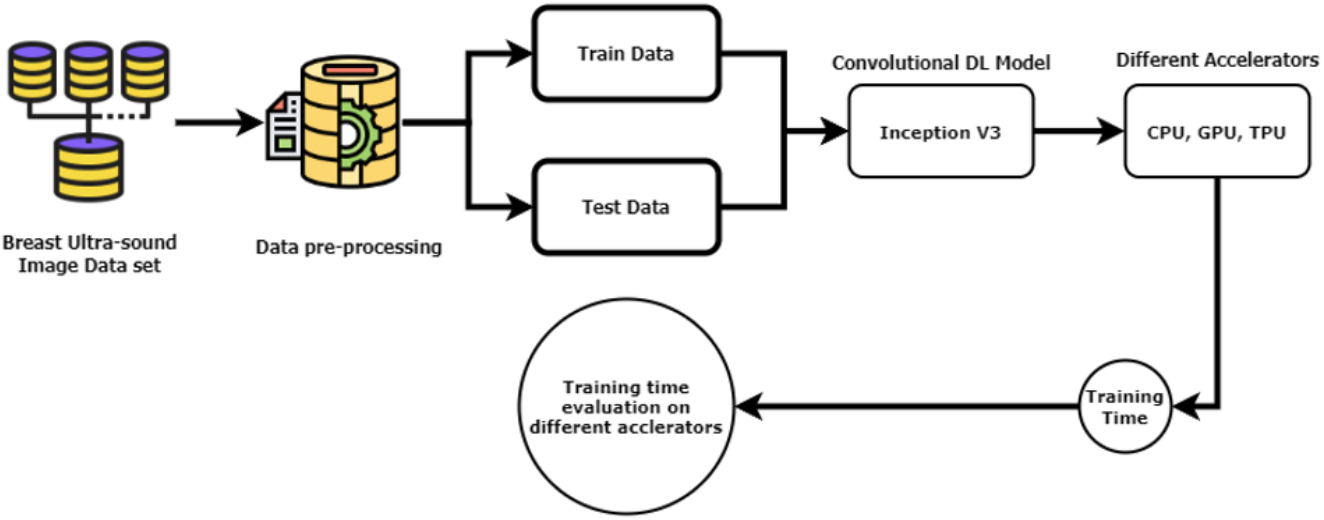

Fig. 1. Training time comparison pipeline of the Accelerators

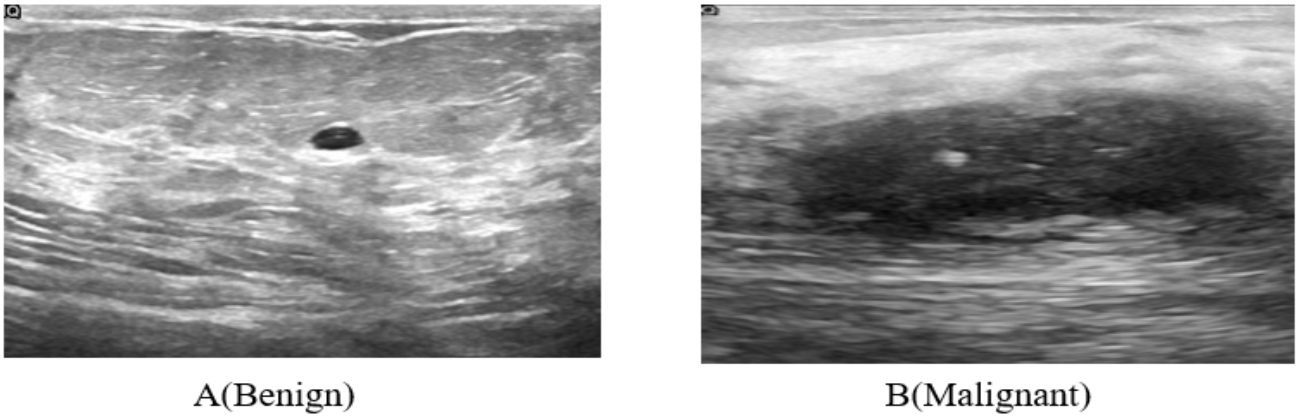

Fig. 2. Breast ultrasound image dataset

VHDL AND IT'S IMPORTANCE IN MACHINE LERANING AND DEEP LEARNING

VHSIC Hardware Description Language (VHDL) is a standardized hardware description language used to design and document electronic systems. It was developed in the 1980s by the United States Department of Defense and has since become a widely used language in the design of integrated circuits (ICs) and field-programmable gate arrays (FPGAs). VHDL is a high-level programming language that allows designers to describe the behavior and structure of digital systems at different levels of abstraction [27]. In recent years, VHDL has gained increasing importance in the field of machine learning (ML) and deep learning (DL). One of the main reasons for this is the increasing demand for high-performance hardware systems that can support the complex computations required by ML and DL algorithms. VHDL is well-suited for designing such systems because it allows designers to describe the behavior and structure of digital systems at different levels of abstraction, making it easier to optimize the design for specific ML and DL tasks.

For example, VHDL can be used to design custom hardware architectures that are optimized for training and inference of ML and DL models. These architectures can include specialized hardware units such as multipliers, adders, and memory modules that are optimized for performing tensor operations, which are fundamental to many ML and DL algorithms. Using VHDL to design such architectures can significantly improve the performance of ML and DL models, as the specialized hardware units can perform tensor operations much faster than general-purpose processors [28]. In addition to its use in designing custom hardware architectures, VHDL is also used to design ML and DL accelerator chips, which are specialized ICs that are designed to accelerate the training and inference of ML and DL models. These chips typically include a large number of specialized hardware units that are optimized for performing tensor operations and can significantly improve the

performance of ML and DL models.

In conclusion, VHSIC Hardware Description Language (VHDL) is an important tool in the field of machine learning (ML) and deep learning (DL) due to its ability to describe digital systems at different levels of abstraction and its wide range of built-in functions and operators. Its use in designing custom hardware architectures and ML and DL accelerator chips has significantly improved the performance of ML and DL models, making it an essential tool in the development of advanced artificial intelligence systems [29].

METHODOLOGY

This method was developed to facilitate processing assessments of these accelerators because the comparison of how different accelerators handle the deep learning model is the main objective of this study. In this experiment a , the breast ultrasound image dataset is used. It is pre-processed for model input, divided into 80:20 training and testing data sets, and fed into the Inception V3 deep learning convolutional model.

Because we are using Google collab here, we can run this process on three different accelerators, including the CPU and the collab GPU or TPU, so we can train the model on different accelerators and evaluate the pipeline based on the model's training time on individual accelerators. In order to understand the overall execution time gaps, an overall execution time was also calculated.

Dataset

For this study, the Breast Ultrasound Image Dataset was utilized. The Breast Ultrasound Image Dataset is available for free download from Kaggle. This dataset was created in 2018 by Walid et al. The initial sample consisted of 718 ultrasound images from females 25 to 75 years old. The classification was made using 294 cancerous images and 306 healthy images in total.

Data pre-processing

In this scenario, data processing is just as important as deep learning, and we use image augmentation to solve the class imbalance issue. Using rotation, width shifting, height shifting, and horizontal flip, we generate additional augmented images to address the problem of class imbalance. After augmentation we use total 1200 data, 600 for each class.

Inception v3

In Inception, there are 42 layers. InceptionV3, the third version of Google Brain's Inception module, consists of 159 layers. By combining small and large kernels, the primary objective of the Inception module is to reduce the computational cost and number of parameters required to learn multi-scale interpretations.

Table 2 shows that while a general CPU can process images using deep learning algorithms, the process takes over two hours to complete. By contrast, an external NVDIA GPU can complete the task in less than twelve minutes. Additionally, the Colab GPU performed similarly to the NVDIA GPU but slightly better, and the training time was slightly longer than ten minutes. The Colab TPU, however, performs better than most and nothing comes close to its training time, which is only about two minutes.

EXPERIMENTAL SETUP

This experimental setup is investigated to comprehend the colab GPU, TPU, and personal computer hardware features in Table I. As we are using a deep learning algorithm some hyper parameters are given bellow to understand the full setup.

TABLE I
DEVICE AND THEIR FEATURES

Accelerator	Cores	Core clock	Ram	Threads
PC	2	3.9 GHz	16 GB	12
CPU	2	4.4 GHz		12
GPU(NVDIA GTX 1650)	CUDA 896	1710MHz	4 GB	
GPU(Colab Tesla k-80)	CUDA 4992	GHz	24 GB	
Colab TPU			32 GiB	

- NumPy version 1.21.5.
- TensorFlow version 2.8.0.
- Colab GPU NVIDIA Tesla K80
- Keras version 2. x
- Epoch count: 50
- Batch size: 32
- Test train split: 80:20
- Loss function: categorical crossentropy
- Optimizer: Adam

RESULT AND DISCUSSION

In this instance, training the deep learning model on various accelerators while using the same pipeline and measuring the overall training time on each. Image processing is a useful benchmark for assessing an accelerator's overall efficiency because it consumes a significant portion of its critical power.

TABLE II
MODEL TRAINING TIME ON DIFFERENT ACCELERATORS

Accelerator	Model training time
CPU	2h 5min 39s
GPU NVDIA GTX 1650	12min 29s
GPU Colab Tesla k-80	10min 9s
Colab TPU	1m 53s

Though the only thing we are worried about is how long it will take the model to train on various accelerators in fig 3, for the sake of clarity, we have also noted down the entire scenario, including each and every step of the procedure. Beginning with the import of libraries, moving on to the data pre-processing, image plotting, model summary, and finally

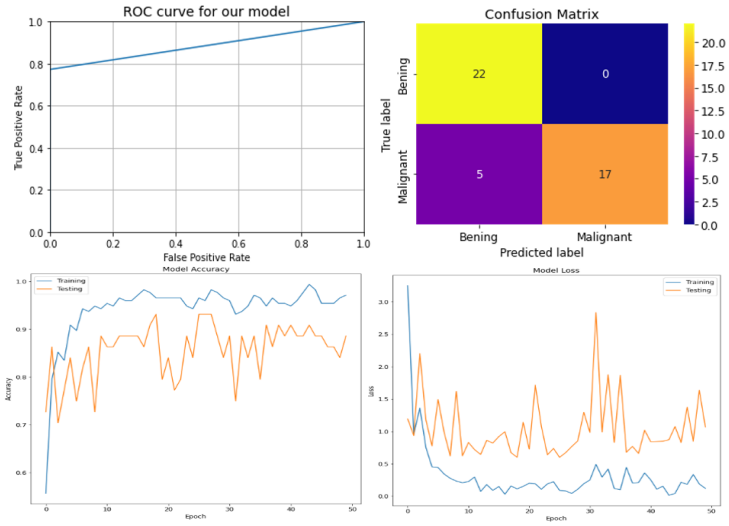

Fig. 3. Using colab TPU the model performance graph,confusion matrix, model accuracy and model loss plots

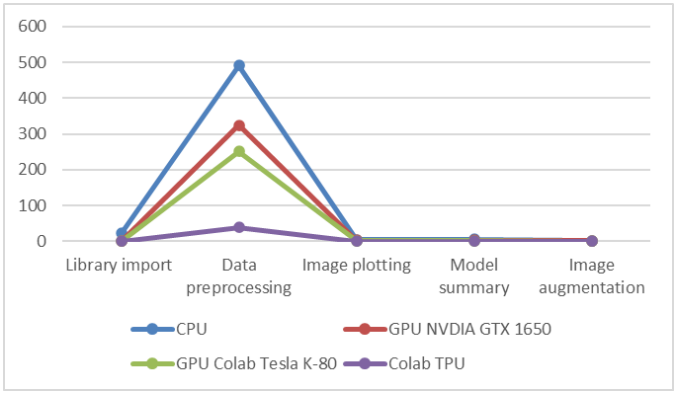

Fig. 4. Accelerators execution time comparison on different task

taking into account the time of image augmentation, the Colab TPU outperformed with the shortest amount of time, proving conclusively that contemporary TPUs are faster than up graded GPUs.

TPUs and GPUs are both currently offered on the market. To be more precise, the former relies on a Graphics Processing Unit, whereas the latter uses a Tensor Processing Unit. Each of the different processors carries out specific tasks. All of the intricate calculations required to produce a visual representation of an object are handled by the graphics processing unit (GPU). The TPU was created to function best with TensorFlow's neural networks. A TPU outperforms a GPU for models with lengthy training times, large effective batch sizes, or matrix calculations.

GPUs, TPUs, and CPUs are all types of processors that are used for different purposes in machine learning (ML) models. Each type of processor has its own unique characteristics and advantages that make it better suited for certain tasks. GPUs, or graphics processing units, are specialized processors that are designed to handle the complex calculations required for graphics rendering. They are particularly well-suited for ML tasks because they can perform many calculations simultaneously, making them faster and more efficient than CPUs. GPUs are commonly used to train deep learning models, which require large amounts of data and computations to learn complex patterns and relationships.

TPUs, or tensor processing units, are specialized processors that are designed specifically for ML tasks and we get the highest performance on colab TPU the model ROC/AUc curve, confusion matrix, accuracy and loss curves are shown on Fig 3., particularly those involving tensor operations. Tensor operations are fundamental to many ML algorithms and are used to manipulate multi-dimensional arrays of data. TPUs are highly efficient at these operations and can significantly speed up the training and inference of ML models. CPUs, or central processing units, are the most common type of processor found in computers. They are responsible for executing instructions and performing a wide variety of tasks. While CPUs are not as specialized as GPUs or TPUs, they are still an important part of ML models because they are responsible for coordinating and organizing the work of other processors. CPUs are also good at handling tasks that require sequential processing, such as data preprocessing and postprocessing.

In general, GPUs and TPUs are better suited for training and inference of ML models, while CPUs are better suited for tasks that require sequential processing. The choice of processor will depend on the specific requirements of the ML model and the resources available. Some ML models may benefit from using multiple types of processors in combination to achieve the best performance.

All specialized hardware options for neural network inference and training, including CPUs, GPUs, and TPUs, exhibit significant, comparable enhancements. It depends greatly on the model and its definition. Depending on the needs, the selection of specialized hardware for neural networks varies due to the yearly innovations. In our scenario, the process

considered the same model and pipeline for all accelerators, which provides a level playing field for all accelerators to perform optimally, and the conclusion was that the TPU completed the tasks with the most stringent time constraints.

CONCLUSION

Google developed the Tensor Processing Unit (TPU) to handle TensorFlow deep neural network processing. One objective of TPU design is to shorten the training period for neural network models. Google claims that compared to weeks or months on other hardware, training neural network models on equipment with a TPU incorporated only takes a few hours. TPU is therefore faster than both the CPU and the GPU.

It is also a significant fact that the performance of accelerators is not static; rather, it is a dynamic performance that can increase in some cases and decrease in others. However, the most important fact is that CPUs are no longer suitable for use in deep learning scenarios because the advanced world requires more optimized and cost-effective products. GPUs and TPUs are more competitive in deep learning scenarios, and their performance can be comparable.Our future direction of work will be more in-depth in order to comprehend the core scenarios of each accelerator and examine them with varying levels of work complexity and data types to manage.

DECLARATION OF ORIGINALITY

I, Charles Arsenal Okere, hereby declare that I wrote this paper and work on my own, using only the sources and aids cited. Sentences or portions of sentences that are quoted verbatim are marked as such; all other references to the statement and scope are accompanied by complete bibliographic information. The paper and similar works have neither been submitted to an examination body nor published. This paper has not been used, even partially, in a previous examination or course performance. I consent to my work being analyzed by a plagiarism analyser.

REFERENCES

[1] Arora, M. (2012). The architecture and evolution of cpu-gpu systems for general purpose computing. By University of California, San Diago, 27.

[2] Mišić, M. J., urević, . M., Tomašević, M. V. (2012, May). Evolution and trends in GPU computing. In 2012 Proceedings of the 35th International Convention MIPRO (pp. 289-294). IEEE.

[3] Raju, K., Chiplunkar, N. N. (2018). A survey on techniques for cooperative CPU-GPU computing. Sustainable Computing: Informatics and Systems, 19, 72-85.

[4] Sharma, M., Bhatnagar, E., Puri, K., Mitra, A., Agarwal, J. (2022). A Survey of RISC-V CPU for IoT Applications. Available at SSRN 4033491.

[5] Rodriguez, A. (2020). Deep Learning Systems: Algorithms, Compilers, and Processors for Large-Scale Production. Synthesis Lectures on Computer Architecture, 15(4), 1-265.

[6] Han, S. (2017). Efficient methods and hardware for deep learning (Doctoral dissertation, Stanford University).

[7] Fell, A., Mazure, D. J., Garcia, T. C., Perez, B., Teruel, X., Wilson, P., Davis, J. D. (2021). The MareNostrum Experimental Exascale Platform (MEEP). Supercomputing Frontiers and Innovations, 8(1), 62-81.

[8] Bolhasani, H., Jassbi, S. J. (2020). Deep learning accelerators: a case study with MAESTRO. Journal of Big Data, 7(1), 1-11.

[9] Jouppi, N. P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., ... Yoon, D. H. (2017). Proceedings of the 44th Annual International Symposium on Computer Architecture.

[10] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... Zheng, X. (2016). TensorFlow: a system for Large-Scale machine learning. In 12th USENIX symposium on operating systems design and implementation (OSDI 16) (pp. 265-283).

[11] Chen, T., Moreau, T., Jiang, Z., Zheng, L., Yan, E., Shen, H., ... Krishnamurthy, A. (2018). TVM: An automated End-to-End optimizing compiler for deep learning. In 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18) (pp. 578-594).

[12] Hosseininoorbin, S., Layeghy, S., Kusy, B., Jurdak, R., Portmann, M. (2021, March). Scaling spectrogram data representation for deep learning on edge tpu. In 2021 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops) (pp. 572-578). IEEE.

[13] Demirović, D., Skejić, E., Šerifović–Trbalić, A. (2018, June). Performance of some image processing algorithms in tensorflow. In 2018 25th International Conference on Systems, Signals and Image Processing (IWSSIP) (pp. 1-4). IEEE.

[14] Pandey, P., Basu, P., Chakraborty, K., Roy, S. (2019, June). GreenTPU: Improving timing error resilience of a near-threshold tensor processing unit. In 2019 56th ACM/IEEE Design Automation Conference (DAC) (pp. 1-6). IEEE.

[15] Du, Z., Ben-Dayan Rubin, D. D., Chen, Y., He, L., Chen, T., Zhang, L., ... Temam, O. (2015, December). Neuromorphic accelerators: A comparison between neuroscience and machine-learning approaches. In Proceedings of the 48th International Symposium on Microarchitecture (pp. 494-507).

[16] Lin, W. F., Tsai, D. Y., Tang, L., Hsieh, C. T., Chou, C. Y., Chang, P. H., Hsu, L. (2019, March). Onnc: A compilation framework connecting onnx to proprietary deep learning accelerators. In 2019 IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS) (pp. 214-218). IEEE.

[17] Wang, T., Wang, C., Zhou, X., Chen, H. (2018). A survey of FPGA based deep learning accelerators: Challenges and opportunities. arXiv preprint arXiv:1901.04988.

[18] Tsai, H., Ambrogio, S., Narayanan, P., Shelby, R. M., Burr, G. W. (2018). Recent progress in analog memory-based accelerators for deep learning. Journal of Physics D: Applied Physics, 51(28), 283001.

[19] Shawahna, A., Sait, S. M., El-Maleh, A. (2018). FPGA-based accelerators of deep learning networks for learning and classification: A review. ieee Access, 7, 7823-7859.

[20] Zuo, P., Hua, Y., Liang, L., Xie, X., Hu, X., Xie, Y. (2020). Sealing neural network models in secure deep learning accelerators. arXiv preprint arXiv:2008.03752.

[21] Guo, K., Zeng, S., Yu, J., Wang, Y., Yang, H. (2019). [DL] A survey of FPGA-based neural network inference accelerators. ACM Transactions on Reconfigurable Technology and Systems (TRETS), 12(1), 1-26.

[22] Agostini, N. B., Dong, S., Karimi, E., Lapuerta, M. T., Cano, J., Abellán, J. L., Kaeli, D. (2020, September). Design space exploration of accelerators and end-to-end dnn evaluation with tflite-soc. In 2020 IEEE 32nd International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD) (pp. 10-19). IEEE.

[23] Chen, T., Du, Z., Sun, N., Wang, J., Wu, C., Chen, Y., Temam, O. (2014). Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning. ACM SIGARCH Computer Architecture News, 42(1), 269-284.

[24] Chen, Y. H., Emer, J., Sze, V. (2016). Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks. ACM SIGARCH Computer Architecture News, 44(3), 367-379.

[25] Gokhale, V., Jin, J., Dundar, A., Martini, B., Culurciello, E. (2014). A 240 g-ops/s mobile coprocessor for deep neural networks. In Proceedings of the IEEE conference on computer vision and pattern recognition workshops (pp. 682-687).

[26] Li, G., Hari, S. K. S., Sullivan, M., Tsai, T., Pattabiraman, K., Emer, J., Keckler, S. W. (2017, November). Understanding error propagation in deep learning neural network (DNN) accelerators and applications. In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (pp. 1-12).

[27] Chen, Y., Zhang, X., Zhang, Y., Li, J. (2019). A Survey on Hardware Accelerators for Deep Learning. IEEE Access, 7, 132659-132678.

[28] Li, Z., Wang, Y., Chen, Y. (2018). Hardware acceleration for deep learning: A review. Frontiers of Computer Science, 12(3), 356-375.

[29] Zhang, Y., Chen, Y., Li, J. (2018). A survey on hardware accelerators for deep learning on mobile and Internet of Things devices. IEEE Access, 6, 46278-46300.