A Survey on Hardware Accelerators for Neural Networks

Pierre Brosemer
uitxj@student.kit.edu

***Abstract*—Neural networks are a widely used method in the field of Artificial Intelligence. With the ability to beat the best human players at the complicated game of GO [1] neural networks are becoming a high performing tool to solve a variety of problems. Their tradeoff: they require vast amounts of computational power, creating the need for hardware accelerators. This work provides a state-of-the-art survey on the main types of hardware presenting recent products for each of the categories and concludes by comparing each of them. First, the basics of neural networks are covered in order to understand the reasoning behind the hardware design choices, followed by the technical implementations.**

I. Introduction

Artificial neural networks (more often neural networks) are one of the most promising technologies in the current era of computer science. Neural networks can be deployed in an enormous variety of categories and have shown major improvements in these categories, most prominently in speech and image recognition [2] that exceeded prior methods. The field of Artificial Intelligence covers a wide variety of concepts. This paper will mainly cover the subcategory of Deep Learning, more specifically neural networks, which falls under the category of Machine Learning.

Neural networks are at its core a very broad simplification of the biological brain consisting of many interconnected neurons, called nodes in neural networks. The nodes in a neural network are arranged in different layers. Each node passes on a weighted function to a node in the next layer. If this value is above a certain threshold the node will be activated. The activation function then determines the output of this node, depending on the inputs. There are many different activation functions that are chosen by the type of neural network. By changing the weights of each node a learning process similar to that of their biological counterpart is simulated [3].

The development of a neural network consists of two different stages. Firstly, a training phase where the neural network receives an input and compares the resulting output (prediction) with that of a real-world dataset. The error can then be incorporated into the network by changing the different weights. The algorithm that has the task of changing the weights is called the backpropagation algorithm. Secondly, an inference phase, where the network receives an input and makes a prediction. The neural network has a larger computational demand in the training phase as more data is processed, as well as the weights needing calibration.

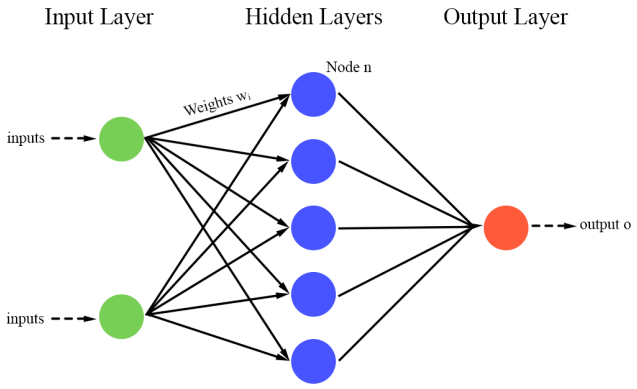

Fig. 1. A typical representation of a neural network. The input layer gives the raw input to the next layer, where nodes process the input and give their output to the next layer. The last layer results in an output (the result)

With some neural networks having over 150 layers [4] up to billions of multiplications can take place in a single neural network and one propagation. The vast amount of computational power needed for training and using neural network lead to the need for specialized hardware, which contributed much to the success seen in modern day implementations. The research on hardware is still ongoing and is one of the most important topics in achieving better and faster results for neural networks. As a result of the timeliness of the topic this survey will only cover a status quo of the current hardware.

II. Different Types of Hardware

Most of computations needed for inference and training include matrix-vector products, caused by the input data being run through the network. Furthermore, the dimension of the input data plays a key role in the amount of computation needed. Assuming a single image has 1000 pixels resulting in 1000 different data points that the neural network has to compute. To efficiently compute the vast amount of data, fast memory accesses or in- place computations of the values are needed. Since these calculations need to be done for the number of nodes, good parallelization is needed to run neural networks efficiently.

Hardware accelerators for neural networks can be divided into four main classes: CPUs, GPUs, FPGAs and ASICs. The focus can be placed on the latter three, since CPUs mostly fall short in terms of computational power for neural networks. CPUs are needed for a wide variety of tasks requiring a flexible system to execute instructions. As a result a CPU is good at executing serial instructions in parallel. Both of the points mentioned above, that are needed for the specialized hardware for neural networks, are not met for CPUs. They are limited by their small amount of cores hindering the ability to parallelize the same instruction. Moreover, a CPU usually works by fetching an instruction and executing it making the training and inference phase slow [5]. While working on neural networks CPUs are accompanied by underutilization, achieving factors of less than 1/10 of that of other alternatives [6]. Nonetheless, Intel is working on accelerating CPUs producing libraries for their Intel Xeon Processor [7].
Another point to consider is the different requirements needed for both the training and inference phase. During the inference phase certain approximations can be made to achieve faster results, which cannot be made for the training phase. This derives in some hardware only being designed for the inference phase [8].
Additionally the hardware can be divided into spatial architectures and temporal architectures [9].

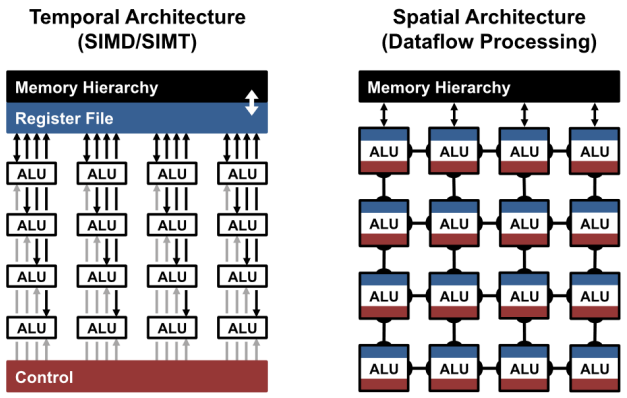

Fig. 2. A comparison between the different architecture types. The ALUs from the temporal architecture can only fetch data from the memory hierarchy [9].

CPUs and GPUs mostly are built on the premise of temporal architecture. Temporal architectures have on central control system for many different Arithmic Logic Units (ALUs), as seen in Figure 2. One ALU only communicates with the memory hierarchy, not with other units. In order to parallelize the workload execution models like SIMD (Single instruction, multiple data) or SIMT (Single instruction, multiple threads) are used. SIMD executes one operation on multiple Data at the same time. SIMT uses SIMD together with multithreading and is implemented in GPUs. ASICs and FPGAs are built on spatial architectures. The different ALUs are connected together to and form a network, that can pass data from one ALU to the next. In spatial architectures ALUs can also have their own control and logic memory. A processing element or processing engine (PE) is an ALU with it's own local memory.

III. Graphics Processing Unit

The Graphics Processing Unit (GPU) is a piece of specialized hardware originally designed for the rendering of images in a computer. The rendering process requires vast amounts of floating point operations resulting in the GPU having hundreds of cores with many individual caches to store the computation. To add on the GPU has more units dedicated to floating-point operations. This makes the GPU an excellent tool for working with neural networks since the specifications align with those for rendering graphics. With over thousand cores the GPU can parallelize the matrix multiplications needed for training and achieve fast training times.

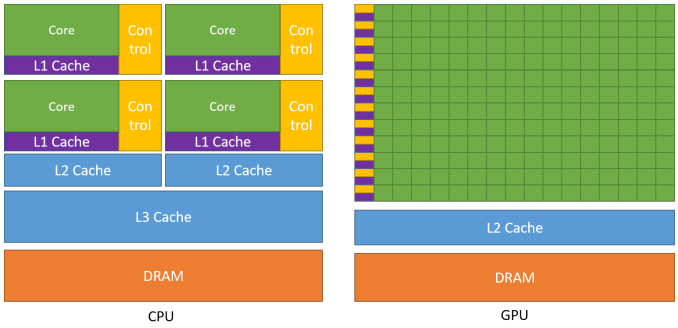

Fig. 3. Different layouts comparing CPU and GPU: The GPU has more cores with more individual caches [10].

Through this structure high memory throughput and good execution of parallel instructions is given. Nvidia, one of the leaders in manufacturing, is implementing its specialized Deep Learning Units with normal cores plus additional Tensor cores [11]. Tensor cores essentially multiply 2 FP-16 matrix (floating point 16 bit) and being able to add a third matrix all in a single operation, performing 64 operations per clock [12]. Their newest product the Nvidia A100 uses the third generation of Tensor cores with better precision.
Another way GPUs accelerate is by making use of the sparsity of matrices used for calculating the different functions. When a large number of zeros is found in the matrices / vectors sparsity can be used to avoid additional operations (since the result of a multiplication with zero will be zero) and safe storage, by not needing to store zeros. The matrix can therefore be compressed, since all the non-zero values don't need to be saved seen in Figure 4. Examples of compression methods are the Compressed Sparse Row (CSR) or Compressed Sparse Column (CSC), which differentiate by saving the position of the column or row. The Nvidia A100 skips the zero values, which results in an acceleration of factor two for their tensor core compute throughput [13].
GPUs are mostly used for training, but can also be used for the inference phase because of their flexible hardware

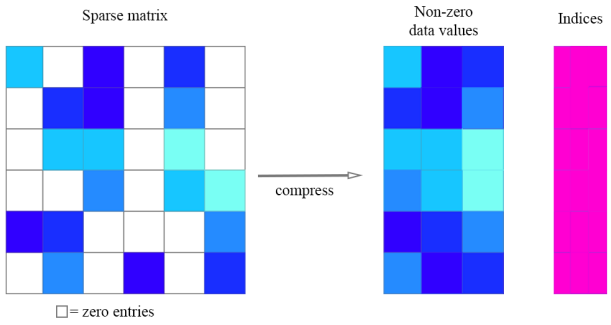

Fig. 4. A sparse matrix can be compressed in order to save space. Only the indices and value are needed for computing the matrix-vector product.

structure [5]. They are specifically fast in training due to their ability to work quickly through large amounts of data (having a high memory bandwidth).

IV. Field-Programmable Gate Array

A Field-Programmable Gate Array (FPGA) is an integrated circuit, which can be programmed by the customer after delivery. The concept of a unit can be configured by using Hardware Description Language (HDL). FPGAs contain logic blocks which in the process can be altered by the HDL. The logic blocks can be configured to execute complex functions or simple logical operations. FPGAs are also used as prototypes for ASICs, since their time to develop is much lower than ASICs. A major problem for researchers working on FPGAs is the hardware-specific knowledge needed for reprogramming them after receiving the product. In recent years, however, a shift towards a more software-close programming can be seen. Despite the shift, writing code can still be a challenge, because the code written for software is fundamentally different than that written for FPGAs. This also opens the door for researchers on neural networks to speed up the training and inference phase for neural networks, since working with a flexible and user-friendly framework is of advantage. The major manufacturers are Xilinx (owned by AMD) and Altera (owned by Intel) [14].

FPGAs are limited by their flexibility compared to GPUs, with the different Logic Units needing reconfiguration, while the GPU tends to have individual units computing tasks. This process of reconfiguration leads to high compile times, making a process, where algorithm designs often change, tedious. Nonetheless, they offer better memory flow, by being able to program the data and control path. The on-chip memory, as well as the ability to pipeline parallelize. The way neural network algorithms are coded also varies. Programming a neural network to be used on a GPU the focus is on parallelizing the work for different units, while on a FPGA the approach can be taken more to the software level by having more freedom in the underlying hardware, e.g. limiting the numerical precision [15].

There are many different approaches of how to achieve efficient processing of neural networks with FPGAs. A vivid approach can be seen by Nurvitadhi et al. [6] using Recurrent Neural Networks (neural networks that contain recurrent connections) for the inference phase. The FPGA is composed of a memory read unit, memory write unit and gatherings of multiple floating-point multiply-accumulate units (FMA). To accelerate matrix vector multiplications the matrix is first divided into column blocks. One FMA is assigned one column block to work on, multiplying the elements of the input vector to the specified column. Since the unit only takes care of one column, this can be done in place. Afterwards the result is given to a reduction unit, which adds the results of 2 column blocks for the concluding result. Many such FMA units are then grouped together to form a cluster seen in 5 . Similar approaches like this with efficient dataflow and usage of fine-grained parallelism can be seen in other scientific works [16] [17].

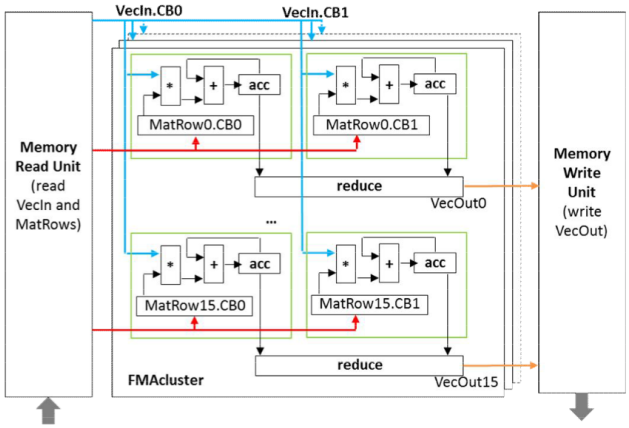

Fig. 5. A simplified depiction showing the layout and dataflows of the above mentioned FPGA. Green boxes represent the FMA units working in harmony to execute the matrix operations [6].

Microsoft found a use case for FPGAs in their datacenter for the inference phase in their Azure Cloud. Every search engine needs an algorithm for prioritizing websites. Microsoft was able to speed up the this ranking of their own search engine Bing by a factor of nearly 2 using FPGAs instead of traditional hardware [18]. Further work was done to achieve even better performance [19]. A key feature of the updated hardware is the ability to support many different configurations of the hardware without having to recompile, allowing for a faster design process. The circuit is composed of many different processing elements as well as buffers. The processing elements can be scaled up to a margin of more units, through their partly independent structure, to achieve faster processing times. Input data is moved from the DRAM

into an input buffer. A software unit distributes the data across the processing elements, which perform independent operations. The results are stored in the weight buffer.
The buffers allow together with the network-on-chip to distribute data across the unit. This reduces the network traffic to and from outside of the chip, by making use of the buffers.

V. Application-Specific Integrated Circuit

Although an Application-Specific Integrated Circuit (ASIC) shares similarities with a FPGA, they are fundamentally different [5]. The most notable difference is that while FPGAs can be reprogrammed after production, this is not the case for ASICs, rather are they designed for a specific cause that will remain the same over their lifetime. To add on their design process is also more complicated and requires more time. Their edge compared to FPGAs is better optimization and energy efficiency, perfecting the hardware-specific advantages of FPGAs.
Google observed that the computational capacity of their datacenters would double, if people used the feature to search by voice for only three minutes a day. This gave them a reason to start working on their own custom ASIC [8]. Making this the use-case the hardware was therefore built specifically for the inference phase. The work became the Tensor Processing Unit (TPU) released in 2016. The main components of the first version included a Matrix Multiply Unit (MMU), a Unified Buffer and Accumulators. The Matrix Multiply Unit contains 256x256 multiplier-accumulators (MACs), which perform 8-bit operations, whose results are stored in accumulators. Many scientific works have shown that during the inference phase lower numerical precision (lower than the usual 32-bit floating point) can be used to achieve faster inference times without loss of accuracy[21]. This makes the Matrix Multiply Unit a good tool to process the data coming through the layers. Since the TPU is only used for inference Google implemented a Weight FIFO, which is read-only and makes accessing the weights needed in the MMU fast. The transitional results are at the end stored in the Unified Buffer. For faster processing a systolic array was used, which reduces read and write operations. Another way the hardware accelerates is by using a 4-stage pipeline parallelism.
Since the release of the first version Google released two newer versions and also announced the work on a fourth version:

- The TPUv2 was built from the ground up working on problems of the first version. The release of the second version focused mostly on higher memory bandwidth and even faster processing, containing two Matrix Execution Units [22]. The second version was now also able to support training models. Another addition was the so-called TPU Pods, which are essentially multiple TPU devices connected together. In a TPU Pod the units are connected in such a way that the workload between the different devices can be split up autonomously, meaning without the help of a host CPU.

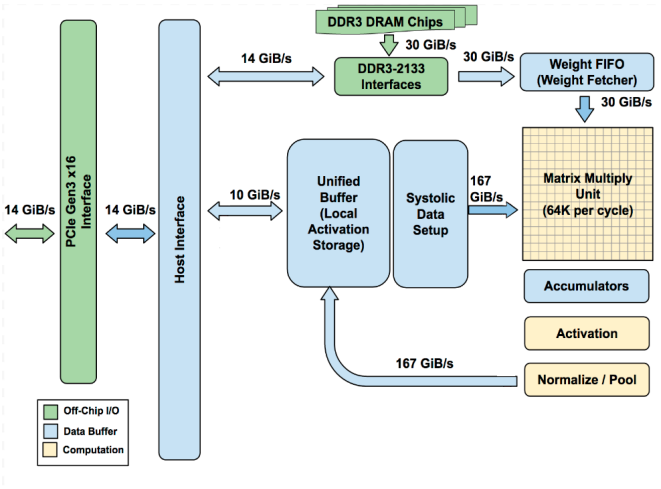

Fig. 6. A simplified block diagram of the first version TPU. The left side represents the host (CPU). The host sends instructions to the TPU and receives a data back [8].

- The TPUv3 improved similar aspects as its predecessor, having four Matrix Execution Units [23].
- Google has announced that it is working on bringing out a fourth version of their TPU, claiming to be 2.7x faster than their third version [24].

VI. Comparison Accelerators

The comparison between ASICs, FPGAs and GPUs will be divided into two main aspects: firstly a general comparison between the different accelerators regarding often used measures and secondly an insight into the research regarding hardware accelerators for neural networks. The comparison will focus predominantly on the hardware used in datacenters. The main metrics for measuring hardware accelerators are performance, energy efficiency and memory bandwidth.

Performance: The performance (often also called the throughput) determines how often the hardware performs a complete convolution/a complete inference is performed on the hardware. Most of the times this metric is measured in billions of operations per second (GOP/s). Other alternatives include billions of Macs per second (GMAC/s) or the amount of inferences achieved (e.g. when talking about image classification the hardware classified x images/second). **Energy Efficiency**: The energy efficiency is the power the unit bestows put in ration with the performance. This is an important unit for edge devices, as power is often a limited source for these devices.

Memory Transfer Rate is simply the rate at which the hardware can read/write data. The accuracy is also a metric cited, but it will not be discussed since the software is the contributing factor for this.

Measuring these standard metrics on different hardware accelerators can be challenging [5]. Most of the scientific

work is only presented on the target application, making good benchmarking between different accelerators hard. Results of benchmarks with different neural networks or different data cannot be reasonably compared. There are different scientific works that are dedicated to creating frameworks for benchmarking hardware accelerators [25] [26] [27] [28]. The most accredited is MLPerf, which claims to be the industry standard for measuring the performance of hardware accelerators for neural networks[29]. It was founded in 2018 and is being worked on by both, members of science and industry, including Google, NVIDIA, Harvard University and Stanford University to name a few. The specific implementation of the MLPerf Framework would overreach the premise of this paper, as such this section will focus on the results. A peak into the results can be seen in Table I. NVIDIA had 85% of the submission, which include many different variants of their A100 GPU and others [30]. The NVIDIA A100 performs the best for both training and inference phase Per-Accelerator. Google has the Max Scale Record with 4096 units of their TPUv3. In comparison, the Intel Processor is in the 100x slower than the GPU [31].

GPUs are widely used being called the ”workhorses” for Machine Learning[5], being among the most used for Machine Learning [32]. There are a lot of frameworks for working with GPUs on Machine Learning, they are very accessible and perform well. To add on they are mostly implemented on the software level, which as referenced in section IV is easier to work on, when having a software background. Since most of the applications such as speech and image computation is being done in the cloud [9] the hardware in big tech companies is being used for the direct application of neural networks.

A metric that is not represented in the MLPerf benchmark is the energy efficiency. Here FPGAs seem to have an advantage thanks to their efficient pathing of data and operating. In scientific research, as well as the benchmark QuTiBench[27] referenced earlier, FPGAs have a better performance/power than the other methods, in some cases up to 10x better than GPUs [19] [5].

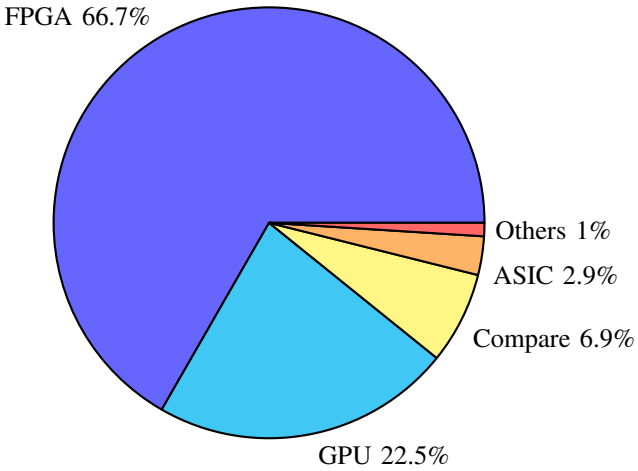

Fig. 7. The results percentages of the different hardware accelerators found in scientific papers [33]. Compare in the pie chart refers to comparison studies.

The second aspect to consider is the scientific research. A detailed systematic literature review examined scientific papers between the years 2009 and 2019 [33]. The categorization for each of the different accelerators can be found in figure 7. The most commonly used accelerator was FPGA, followed by the GPU. The study concluded that the reason was the fast prototyping, as well as the recent rise in accessibility that made FPGAs popular for researchers. ASICs on the other hand, are higher prized compared to GPUs and FPGAs.

VII. Hardware Accelerators in Edge Devices

Although it is not directly a specific type of hardware, hardware accelerators in edge devices play a key role in bringing the services of neural networks to the application. The application for neural networks is, as mentioned in the

TABLE I
A Table showing some results from the training in MLPerf v0.7 in the category Regular, Closed Division Times [20]. While this data is very simplified Real caption*, the full data can be viewed on the official Website.

HW Accelerator	Amount	Image Classification	Object detection light-weight	Translation recurrent	Natural Language Processing	Reinforcement Learning
Google TPUv3 (in Cloud)	16	28.71			56.74	
Google TPUv3	512			2.1		
Google TPUv3	4096	0.48	0.46		0.39	
Google TPUv4	256	4.5	1.43	2.08	5.73	150.95
NVIDIA A100-SXM4-40GB (400W)	1840	0.76				
NVIDIA A100-SXM4-40GB (400W)	1024		0.82	0.71		
NVIDIA A100-SXM4-40GB (400W)	2048				0.81	
NVIDIA A100-SXM4-40GB (400W)	1792					17.07
Huawei Ascend910	512	1.56				
Intel Xeon Platinum 8380H CPU @ 2.90GHz	8	1104.53				

* The different results include different configurations for each hardware accelerator

introduction, manifold. Many applications need fast response times (e.g. autonomous driving) and cannot rely on cloud computing solutions alone. This results in the need for edge devices to have hardware accelerators. the computation to happen right at the source resulting in the need for hardware accelerators in edge devices. This does not mean that cloud is not used at all, rather the edge device works together with the cloud. This leads to three main advantages [34]. Firstly the edge devices can do more calculations on their own reducing network traffic. Secondly the the computation happens in the same domain where it's needed significantly reducing response times. Thirdly if the edge device can't handle a certain task it can ping the cloud for backup.

The hardware accelerators built into edge devices have different demands than those in datacenters. The units in edge devices face limits on the computation available and the energy consumption. This leads to the neural networks being optimized on performance. This can happen in many different ways. Approaches include increasing the depth of the network while keeping the computational budget constant [35] or by filtering the search space [36]. In the mentioned work the search space is a feed from a surveillance camera and the aim is object-detection. By filtering out large amounts of non-target-object frames the spearch space can be reduced reducing the work the neural network has to do.

The devices are also far smaller. This leads to a lot of tradeoffs between the different architectures often small microcontrollers. FPGAs have lower power consumption, can be better optimized for specific applications and have better timings [37]. GPUs/CPUs have more stable architecture and better floating point capabilities [38]. Some concrete examples include a Neural Processing Unit from Siemens, that enables Artificial Intelligence for automation task in manufacturing [39] or a special 8-core neural engine in the iPhone dedicated for image processing [40].

VIII. Conclusion

The hardware in neural networks plays a key role in bringing neural networks to the broad spectrum of its usage, as well as help in making better and smarter neural networks. In this paper the tradeoffs for each of the different methods can be seen. Each of the hardware accelerators have different advantages and disadvantages that make them suitable in their usable environment, which is why choosing the right hardware should be a crucial decision when using neural networks. The different methods for accelerating neural networks have been and are still being optimized to achieve good performance. With the rise of the Internet of Things, more energy-efficient devices will be needed to precisely meet the demands for these devices. The challenge here will be to juggle networking, processing power, and energy consumption with the best performance of the neural network.

References

[1] T. Chouard, "The go files: Ai computer wraps up 4-1 victory against human champion," *Nature News*, 2016.

[2] L. Deng, G. Hinton, and B. Kingsbury, "New types of deep neural network learning for speech recognition and related applications: An overview," in *2013 IEEE international conference on acoustics, speech and signal processing*, pp. 8599–8603, IEEE, 2013.

[3] E. Kavlakoglu, "AI vs. Machine Learning vs. Deep Learning vs. Neural Networks: What's the Difference?," 2020. (accessed 3 February 2021).

[4] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

[5] M. Capra, B. Bussolino, A. Marchisio, M. Shafique, G. Masera, and M. Martina, "An updated survey of efficient hardware architectures for accelerating deep convolutional neural networks," *Future Internet*, vol. 12, no. 7, p. 113, 2020.

[6] E. Nurvitadhi, J. Sim, D. Sheffield, A. Mishra, S. Krishnan, and D. Marr, "Accelerating recurrent neural networks in analytics servers: Comparison of fpga, cpu, gpu, and asic," in *2016 26th International Conference on Field Programmable Logic and Applications (FPL)*, pp. 1–4, IEEE, 2016.

[7] S. L. Gogar, "BigDL – Scale-out Deep Learning on Apache Spark* Cluster ," 2017. (accessed 3 February 2021).

[8] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proceedings of the 44th annual international symposium on computer architecture*, pp. 1–12, 2017.

[9] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017.

[10] "Comparison between a CPU and GPU," 2021. (accessed 4 February 2021).

[11] "Nvidia v100 Datasheet," 2017. (accessed 4 February 2021).

[12] M. Harris, "CUDA 9 Features Revealed: Volta, Cooperative Groups and More," 2017. (accessed 4 February 2021).

[13] "NVIDIA A100 Tensor Core GPU Architecture," 08.10.2020. (accessed 19 February 2021).

[14] P. Dillien, "And the Winner of Best FPGA of 2016 is…," 03.06.2017. (accessed 12 February 2021).

[15] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep learning with limited numerical precision," in *International conference on machine learning*, pp. 1737–1746, PMLR, 2015.

[16] J. Qiu, J. Wang, S. Yao, K. Guo, B. Li, E. Zhou, J. Yu,

T. Tang, N. Xu, S. Song, *et al.*, "Going deeper with embedded fpga platform for convolutional neural network," in *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pp. 26–35, 2016.

[17] C. Wang, L. Gong, Q. Yu, X. Li, Y. Xie, and X. Zhou, "Dlau: A scalable deep learning accelerator unit on fpga," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 3, pp. 513–517, 2016.

[18] A. Putnam, A. M. Caulfield, E. S. Chung, D. Chiou, K. Constantinides, J. Demme, H. Esmaeilzadeh, J. Fowers, G. P. Gopal, J. Gray, *et al.*, "A reconfigurable fabric for accelerating large-scale datacenter services," in *2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA)*, pp. 13–24, IEEE, 2014.

[19] K. Ovtcharov, O. Ruwase, J.-Y. Kim, J. Fowers, K. Strauss, and E. S. Chung, "Accelerating deep convolutional neural networks using specialized hardware," *Microsoft Research Whitepaper*, vol. 2, no. 11, pp. 1–4, 2015.

[20] "MLPerf Training v0.7 Results," 29.07.2020. (accessed 17 February 2021).

[21] A. Rodriguez, E. Segal, E. Meiri, E. Fomenko, Y. J. Kim, H. Shen, and B. Ziv, "Lower numerical precision deep learning inference and training," *Intel White Paper*, vol. 3, pp. 1–19, 2018.

[22] U. H. Jeff Dean, "Build and train machine learning models on our new Google Cloud TPUs," 17.05.2017. (accessed 14 February 2021).

[23] "Cloud TPU System Architecture," 08.02.2021. (accessed 14 February 2021).

[24] N. Kumar, "Google breaks AI performance records in MLPerf with world's fastest training supercomputer," 29.06.2021. (accessed 14 February 2021).

[25] T. Chen, Y. Chen, M. Duranton, Q. Guo, A. Hashmi, M. Lipasti, A. Nere, S. Qiu, M. Sebag, and O. Temam, "Benchnn: On the broad potential application scope of hardware neural network accelerators," in *2012 IEEE International Symposium on Workload Characterization (IISWC)*, pp. 36–45, IEEE, 2012.

[26] H. Zhu, M. Akrout, B. Zheng, A. Pelegris, A. Jayarajan, A. Phanishayee, B. Schroeder, and G. Pekhimenko, "Benchmarking and analyzing deep neural network training," in *2018 IEEE International Symposium on Workload Characterization (IISWC)*, pp. 88–100, IEEE, 2018.

[27] M. Blott, L. Halder, M. Leeser, and L. Doyle, "Qutibench: Benchmarking neural networks on heterogeneous hardware," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 15, no. 4, pp. 1–38, 2019.

[28] S. Dong and D. Kaeli, "Dnnmark: A deep neural network benchmark suite for gpus," in *Proceedings of the General Purpose GPUs*, pp. 63–72, ACM, 2017.

[29] P. Mattson, V. J. Reddi, C. Cheng, C. Coleman, G. Diamos, D. Kanter, P. Micikevicius, D. Patterson, G. Schmuelling, H. Tang, *et al.*, "Mlperf: An industry standard benchmark suite for machine learning performance," *IEEE Micro*, vol. 40, no. 2, pp. 8–16, 2020.

[30] J. Russell, "Nvidia Dominates (Again) Latest MLPerf Inference Results," 22.10.2020. (accessed 17 February 2021).

[31] P. Kharya, "NVIDIA Inference Performance Surges as AI Use Crosses Tipping Point," 21.10.2020. (accessed 17 February 2021).

[32] C. Cotterell, "How AI Is Shaking Up the Chip Market," 2016. (accessed 4 February 2021).

[33] M. A. Talib, S. Majzoub, Q. Nasir, and D. Jamal, "A systematic literature review on hardware implementation of artificial intelligence algorithms," *The Journal of Supercomputing*, pp. 1–42, 2020.

[34] X. Wang, Y. Han, V. C. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 869–904, 2020.

[35] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

[36] C. Zhang, Q. Cao, H. Jiang, W. Zhang, J. Li, and J. Yao, "Ffs-va: A fast filtering system for large-scale video analytics," in *Proceedings of the 47th International Conference on Parallel Processing*, pp. 1–10, 2018.

[37] S. Jiang, D. He, C. Yang, C. Xu, G. Luo, Y. Chen, Y. Liu, and J. Jiang, "Accelerating mobile applications at the network edge with software-programmable fpgas," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, pp. 55–62, IEEE, 2018.

[38] Y. Chen, S. Biookaghazadeh, and M. Zhao, "Exploring the capabilities of mobile devices in supporting deep learning," in *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*, pp. 127–138, 2019.

[39] "simatic s7-1500 tm npu."

[40] H. Chandra, "Hardware acceleration for machine learning on Apple and Android devices," 08.10.2020. (accessed 17 February 2021).