A Study of Hardware Platforms for Machine Learning Algorithms and Neural Networks

Sayali S Pangre[1], Samrit Kumar Maity[2], Milind Bhandare[3], Aditya K Sinha[4]

[1]Department of Computer Science and Engineering, SOCSE, SandipUniversity, Nashik, India
[2]Joint Director, HPC Technologies CDAC, Pune, India
[3]Assistant Professor, SandipUniversity, Nashik, India
[4]Associate Director, HOD-ACTS CDAC, Pune, India

***Abstract*—** With recent development in deep learning, neural networks are becoming larger and larger in terms of number of layers, connection between neurons and number of parameter they consider. Facts such as growing volumes and varieties of input data, cheaper and more powerful computational resources, data storage, large-value predictions that can guide towards better decisions and smart actions in real time without human intervention are playing critical role in deep learning research. Machine learning helps to extract meaningful information from data if supported with massive computational resource. All these require models that can automatically analyze large complex data and deliver accurate results sometime in real-time scenario. Machine learning and in specific, deep learning plays an important role in developing these models. Developing and handling such complex and large models comes with few challenges such as 1) it is hard to distribute and fit large models on small portable devices. 2) Training large neural network is a time consuming process and may yield high error rate. 3) Consumes more energy due to larger model training and more memory references.

Various hardware platforms are available for various type of task of machine learning. Each has its advantages and suitability for specific kind of workload under specific environment. The data complexity and velocity determines how much processing is needed, while the environment typically determines the final latency and power requirement. In this paper, we discuss various types of machine learning workload and their properties along with various types of hardware. We also discuss how these resources satisfy different machine learning workload requirements. Understanding strength of each type of hardware component with respect to types of workload they handle would enable us to benchmark those hardware optimally.

Keywords— Artificial intelligence (AI), application specific integrated circuit (ASIC), artificial neural network (ANN), central processing unit (CPU),computer architecture, field programmable gate array (FPGA), graphics processing unit (GPU), intelligence processing unit(IPU), machine learning (ML)

I. INTRODUCTION

It is the pattern of computation which has significant influence on design of any hardware. One such example is the growth of cloud computing domain, which catalyzed the development of new kind of hardware components[6]. Machine learning related computation has emerged as key workload and became major driving force for new kind of hardware design. With the advent of neural network related research work, minimum training time and higher inference throughput in neural networks processing came as a key challenge for hardware designers. This challenges and associated demands led to huge competition in the field of hardware manufacturing. Artificial Intelligence is the study of machines that can mimic human mind and its actions. Machine learning is constructing computer algorithms that automatically improve themselves by finding patterns in existing data without explicit instructions.

Machine learning largely depends on data. If algorithm fed with data which has more variety and quality more accurate the model becomes. Machine learning and artificial intelligence is progressing rapidly as scientists are building more and more complex AI software/hardware products. Notable machine learning technologies are Google's Tensorflow[42], AlphaGo[43], NVIDIA's DGX[44,45,46], Amazon's Alexa[47], Microsoft's Azure[48], IBM's Watson[49] and Intel's Nervana[50].

Many machine learning algorithms have been used for years; the ability to automatically apply complex mathematical operations on data in faster way is a recent advancement. Notable ML applications that are very popular are IBM's Deep Blue[51], Google deep minds AlphaGo, online recommendation systems from flipkart and amazon, real time advertisements on mobile phones and web pages, web search results, digital personal assistant such as Alexa[55], cortana[55] etc. Other notable AI applications are on speech recognition, pattern classification, and computer vision.

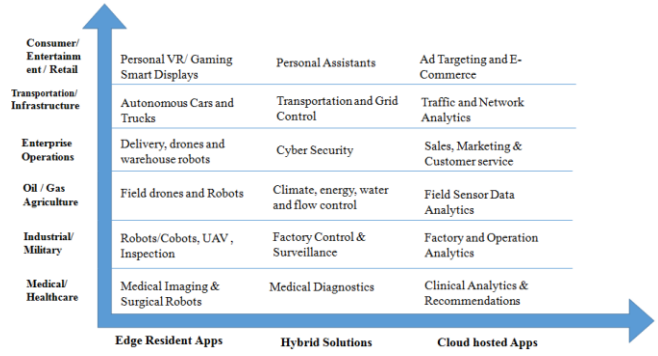


Fig.1. Landscape of Application of Machine Learning[13]

Google has become essential piece of our everyday lives. The millions of images uploaded to instagram, twitter,

facebook are sorted by image recognition techniques, which accurately classifies and give better search results. Image enhancement involves restoring and filling missing details of a pictures by extrapolation and other image processing techniques. Google's machine translation, speech recognition engine to understand spoken questions etc all are based on deep neural network. Recommendations system of Youtube, Netflix etc monitors and records users viewing habits and suggests videos accordingly.

While machine learning plays a important role in the advancement of technology today, there is lot of research work [13][52] going on which presents the opportunities and challenges in designing hardware to cater these domain requirement. These work study recent hardware platforms available for machine learning in the context of neural networks type computation.

Following section presents an overview of the neural networks and its requirements that contributed to increased demand for specialized hardware for Machine learning.

- Advent of neural networks related research work.
- Key factors affecting computation cost in machine learning.
- Dimension of the input data.

a) Advent of Neural networks

Traditional machine learning algorithm has fixed set of algorithm which is applied on huge set of data to discover patterns in them. In such computation the importance of computation and memory operations are balanced. However in deep neural network, it is mostly complex graph operations. At hardware level it culminate into larger dese matrix operations. Compute platform densely populated with many ALU are only suitable for such kind of workload.

b) Key factors affecting computation cost in machine learning

As the need for increasing efficiency of neural networks is evolving, models are becoming larger, deeper and complex. For example in 2012, AlexNet [11] had only 8 layers but as of today ResNet is having 152 layers, even though both are used for same task for object recognition. Due to these factors back propagation algorithm takes a lot of time for computations to reach to global optima while ensuring minimal error.

c) Dimension of the Input data

As the input dimensionality increases, there is lot of data, computation required to optimize the back propagation algorithm. For example, if we take n*n image we need to store n^2 number of values in the first layer itself. As we go deeper, the number of values to be stored and to be computed increases exponentially. These factors insists to employ specialized hardware for machine learning.

d) Training vs Inference

Workloads can be divided into two many stages or categories i.e. training (Model Building) and inference (Model use). Training starts with forward propagation calculation and results of forward propagation are compared against correct value to calculate the error. If error rate is high than actual result then backward propagation propagates the error back through network layers and updates their weights using gradient descent to improve the network's performance at the task it is trying to learn. This can be done in batch mode with hundreds of training inputs (i.e., images for image classification network or spectrograms for speech recognition) and operate on them simultaneously during DNN training in order to prevent over fitting and more importantly amortize loading weights from GPU memory across many inputs, expanding computational effectiveness.

Inference is the process of using a trained machine learning algorithm to make a prediction on a new data. For inference performance goals are different. In the Inference phase a single data point is run through the model which was hyper parameterized in training phase. To minimize end to end response time during inference small batches are treated as input unlike training because services relying on it requires optimal responsiveness so that users don't have to wait several seconds while the system is accumulating images. Hence compute workload during training is higher than inference as well as throughput requirement. In contrast low latency is very important during inference workload. Based on this findings, we can adapt existing hardware architecture or we can develop new one.

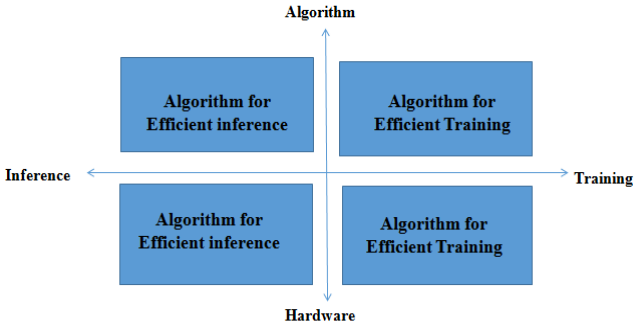


Fig.2. Training and Inference[38]

II. TECHNOLOGY

A. Hardware for Machine learning

Neural network hardware ranges from single core chips to high end neural processing systems. A variety of attributes have been used to classify NN hardware, such as system architecture, inter-processor communication networks, on-chip or off-chip learning, degrees of parallelism, general purpose or special purpose devices etc.[9]. Here we discuss various type of NN hardware candidate based on the factors like system architecture, on chip parallel processing, etc. NN systems are divided into

two categories. One is standard chips and the other is neural chips. Standard chips are further classified as multi-processor and the sequential accelerators.

1. Different Hardware Architectures

In general machine learning hardware's can be classified as:

- Central Processing Unit (CPU)
- Field Programmable Gate Arrays (FPGA)
- Graphic Processing Unit (GPU)
- Application Specific Integrated Circuit (ASIC)

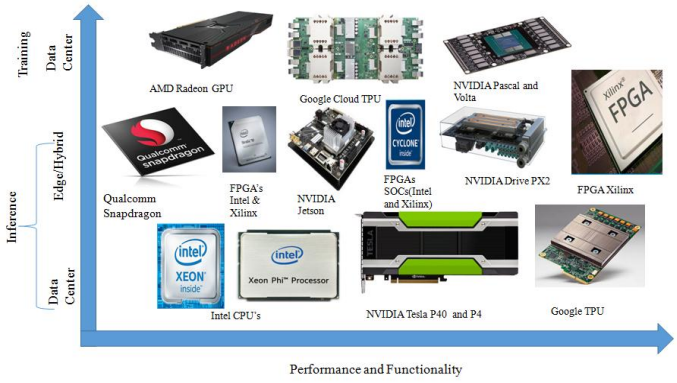


Fig.3. Hardware Platforms for Machine Learning[41]

a) Central Processing Unit

Central processing unit is the integral part of performing computations on a computer. In CPU components include Arithmetic Logic Unit (ALU), Central Unit (CU), and Memory Unit (MU). ALU is generally used to perform arithmetic operations. CU process the control signals arriving through control bus etc.. Memory Unit includes RAM, ROM, Cache memory. When ALU requests for data, CU manages and satisfy such requests by initiating memory operation such as loading the values from registers, retrieving the values from the cache memory unit etc.[12]. General purpose CPUs is highly programmable architecture but has low energy efficiency with low computational throughput compared to other architectures. General purpose CPU with accelerators can be customized to a dedicated ML task but computational throughput depends on attached accelerators.

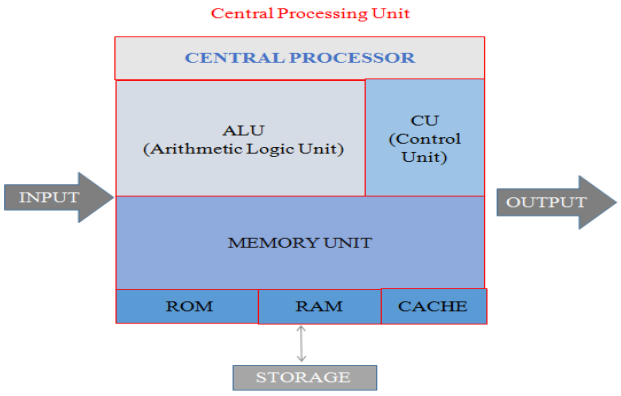


Fig.4. Central processing Unit

b) Graphics Processing Unit

Now a day, GPUs are most preferred and widely used in ML and DL training. GPUs are designed for performing massive parallel tasks supported by thousands of tiny compute cores and high memory bandwidth. In general, machine learning algorithm involves large number of matrix multiplications and additions. GPU's are generally used for graphics processing which also involved massive matrix operations.. Naturally GPUs performs really well with deep learning algorithms

c) CPUs Vs GPUs

While GPUs are good for training it has lacks efficiency for inference operation because of its host to device bandwidth issue.

CPU contains large cache memory but fewer cores. Each core is capable of serving hundreds of software threads per instant. In contrast GPU contains thousand of core and together they can handle millions of threads. CPU focuses of low latency whereas GPU focus to achieve high throughput. CPUs feature control logic for out of order execution and speculative execution whereas GPUs contain architecture which is tolerant to memory latency. In general CPUs consumes more power than GPUs. CPUs are optimized to be good at executing serial but complex instructions whereas GPUs are good at processing embarrassingly parallel instructions[8].

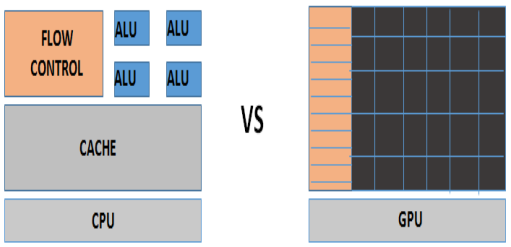


Fig.5. CPU vs. GPU

CPUs and GPUs are different in terms of structure they are built upon. CPUs architecture is designed to be narrow and deep, whereas GPUs are designed to be shallow and wide. This design implementation gives GPUs to execute on thousands of compute cores in parallel.

1. A GPUs compute core (compute unit) is designed to be simpler than modern CPUs core.

2. CPU cores can handle branch prediction whereas GPUs cores are bad at branch prediction.

3. GPU is a SIMD compute engine where during execution, instruction set is same for all cores but they will be operating on different data.

4. CPU cores are equipped with large cache memory and each core operates at a higher clock speed.

d) Field Programmable Gate Arrays

Hardware description language (HDL) is used to program FPGA hardware. FPGAs contain programmable logic system that can be used to reprogram according to functionality. Even though FPGA performance in terms floating point operations is less than GPUs. But FPGAs is better with respect to energy consumption. Few examples of FPGA based machines are Stratix IV (Altera), Spartan (Xilinx) etc.

FPGAs re-configurability allows adaption to evolving neural network architectures/frameworks. We can implement custom neural network architecture using FPGA. It also has hardware acceleration for specific operations on-chip (MAC) and on-chip memory (SRAM) available. TTM (time to market) is less restrictive than for ASICs as the firmware can be altered. But it is not appropriate for low power applications. It has limited on-FPGA memory (SRAM) and limited data rate to external memory (DDR).

A large number of logic gates and RAM blocks are used to build fundamental blocks of FPGA. In FPGA architecture to increase the number of floating-point execution operations work is being done. Artificial neural network execution on FPGAs tries to exploit the functionality of reprogramming ability.

FPGAs provide the analog features into the processing. In FPGAs, it is possible to achieve low latency compared to CPUs. FPGAs can cater to higher input-output connectivity which gives higher bandwidth. The cost of building FPGAs is much higher compared to general MIPS (million instruction per second) based architectures. FPGAs are designed to be a reconfigurable circuit. Design of instruction based architecture is done via software, but FPGAs are configured using a hardware circuit.

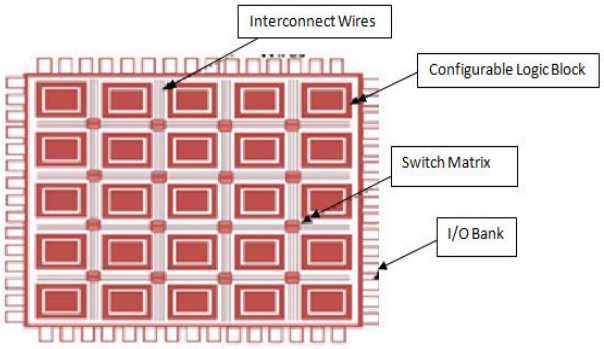


Fig.6. FPGA

Architecture wise, FPGA architecture includes configurable logic block, I/O block, and switching matrix interconnect. In configurable logic block(CLB), it contains logical cells or slice, full adder and multiplexer(MUX) logic, D flip-flop, 4 input look-up table. In FPGAs [1], authors include digital clock manager, block RAM, and multiplier logic block. Digital clock manager is used for frequency generation, phase shifting, etc. Multiplier block is used for multiplication operation on an 18-bit multiplier. Block RAM includes a dedicated memory dual memory (16kb).

e) Application specific Integrated circuit

Design of specific architectures, which focuses on a particular field or area. For example ASIC are designed in area like Bitcoin mining, Machine learning etc. Flexibility in ASIC is very less compared to CPUs, GPUs and FPGAs. ASICs are further classified into:

- Full-custom ASICs
- Semi-custom ASICs
- Platform ASICs

Full custom ASICs are designed for specific applications only. They cannot be used for further modification in hardware. Whereas semi-custom ASICs are customized to allow modification in terms of few functions. Platform ASICs are designed based on silicon consists of a group of slices which decreases design duration(cycle) and improves the cost utilization factor. The slice is a pre-manufactured device that is used to implement a custom system- on-chip.

ASICs have low power and highest performance for targeted applications. Chip architecture maybe invariable to adapt to fast development of Deep Learning topologies/frameworks. In most cases restricted to inference only. In manufacturing point of view, it requires high cost for development and production also along with large market volume to justify development costs considering slow time to market.

Recent notable development from Google using ASICs is of Tensor Processing Unit (TPUs). TPU can use for both

training and inference. Also Intel designed Nervana chip, which are used for deep learning inference ASICs [9].

f) IPU(Intelligence Processing Unit)

The Graphcore has built new type of processor for accelerating machine learning and AI applications. Graphcore systems are good at both training and inference. It is specifically designed for machine intelligence workloads. It has been developed to work effectively on the extremely complex high dimensional models that are required for machine intelligence workloads. It focuses on massively parallel, low precision, floating point computation while providing much higher compute density than other solutions. It holds machine learning model inside the processor.

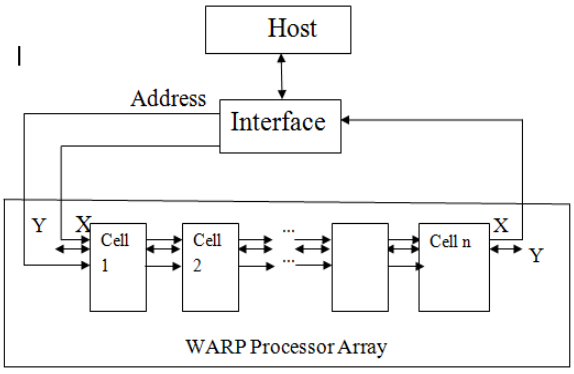
g) General Purpose Processors

i. Digital Signal Processor

A digital signal processor is used mainly for performing addition, subtraction, multiply, divide very quickly. But it has similar bottleneck as CPUs due to von Neumann architecture. It has higher performance than CPUs but lower than GPUs.

ii. Systolic Arrays

It is based on dataflow architectures based on a network of tightly-coupled homogeneous processing elements. Computations are performed in a pipelined manner by passing data through the systolic array [53].



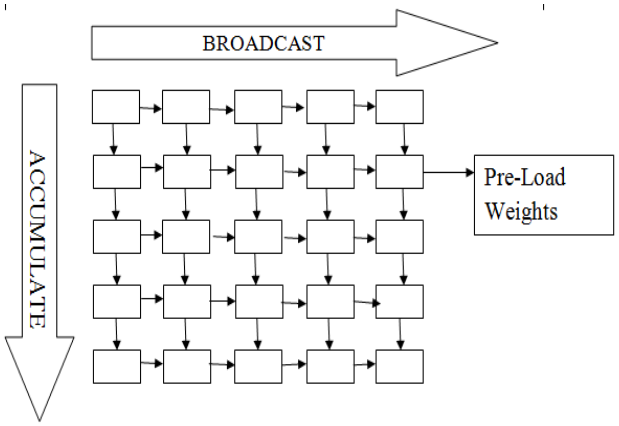


Fig.7. Systolic Arrays

iii. SIMD Processor Arrays, VLIW and Vector or SIMD Co-processor

It contains the multiple processing elements e.g. MAC Units which performs same operation on different pieces of distributed data simultaneously .i.e. data level parallelism. In VLIW various different instructions performed in parallel i.e. instruction level parallelism.

h) Neuromorphic Chips

Neuromorphic computing is inspired by design of human brain like architecture – like human brain contains billions of neurons linked with synapse. In traditional processor design, compute core is at different place, storage at different place, memory unit at different place. This design is inspired by Von Neumann architecture and has issue such as data transfer bottleneck. But in Neuromorphic chips scientists have merged all this in only one chip for processing and storage also. At device level Von Neumann architecture contains registers, capacitors, inductors, transistors etc. These components are energy hungry time consuming, having rigid design with limited inter-connectivity.. In contrast Neuromorphic computing contains high speed neurons, low power consumption, fault tolerant, no need to be programmed.

Examples are DARPA Synapse program (Systems of Neuromorphic adaptive Plastic scalable electronics), IBM TrueNorth, Stanford Neurogrid, SpiNNaker(Spiking Neural Network Architecture), HRL Neuromorphic chip etc.

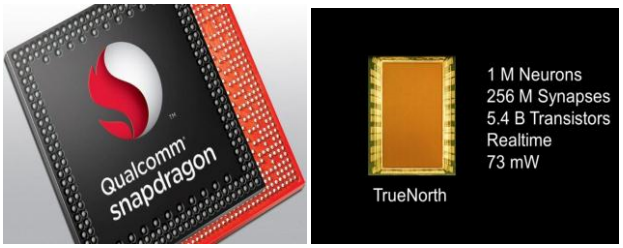


Fig.8. Neuromorphic Chips

i) Advanced Technologies - ReRAM and eDRAM and In-memory Computing

ReRAM-based main memory design, PRIME, which substantially improves the performance and energy efficiency for neural network (NN) applications, benefiting from both the PIM(processing in memory) architecture and the efficiency of ReRAM-based NN computation. In PRIME, parts of the ReRAM memory arrays are enabled with NN computation capability. They can either perform computation to accelerate NN applications or serve as memory to provide a larger working memory space [17]. Embedded DRAM (eDRAM) is explored in to reduce the energy cost in memory access of the weights in DNN [19].

B. Case study-Tensor Processing Unit (TPU)

Tensor processing unit is cutting edge custom designed ASIC, optimized to run machine learning algorithms. In training neural networks, TPUs proved to be efficient compared to CPUs, GPUs and FPGAs. In 2006, FPGAs, GPU were understood to be sufficient enough to satisfy the demand. But after the reemergence of neural networks, the demand for hardware with mega compute capacity is on rise.. In 2016, Google introduced TPU[10], integrated into their data centers and reported the metrics of improvement about performance and cost. Below is the detail of different generation of TPU.

- 1st generation TPU
- 2nd generation TPU
- 3rd generation TPU
- Edge TPU (IoT applications)

1) 1st Generation TPU

TPUs First generation included an 8 bit multiplication design. It included CISC instruction set. First generation of TPU is constrained with limited memory bandwidth along with only integer operation. . Whereas second generation TPU were capable with floating point operations. From neural networks perspective, 1st generation TPU performed well on inference work but not during training phase [2].

Architecture wise TPUs include 28nm process and die size of 331 mm2. Its memory includes 28MiB chip memory, 256 256 systolic array containing 8 bit multipliers. TPU contains 8MB dual channel with DDR3 SDRAM having almost 34GB/s of bandwidth. It is operating over 2133 MHz and having 4MB of 32 bit accumulators along with PCIe 3.0 bus.

2) 2nd Generation TPU

In 2017, Google launched second generation of TPU with vast improvements in terms of memory bandwidth. 2nd generation TPU having bandwidth as 600 Giga Bytes per second with compute performance up to 45 Tera Flops.. Google has also parallelized TPUs into TPU pod. Each TPU pod include 64 TPUs to provide parallel execution. Parallel execution on TPU pod require no additional software modifications, they are done at hardware level itself. 2nd generation TPU allows floating point operations. And can be used for both training and inference workload of ML/DL domain.

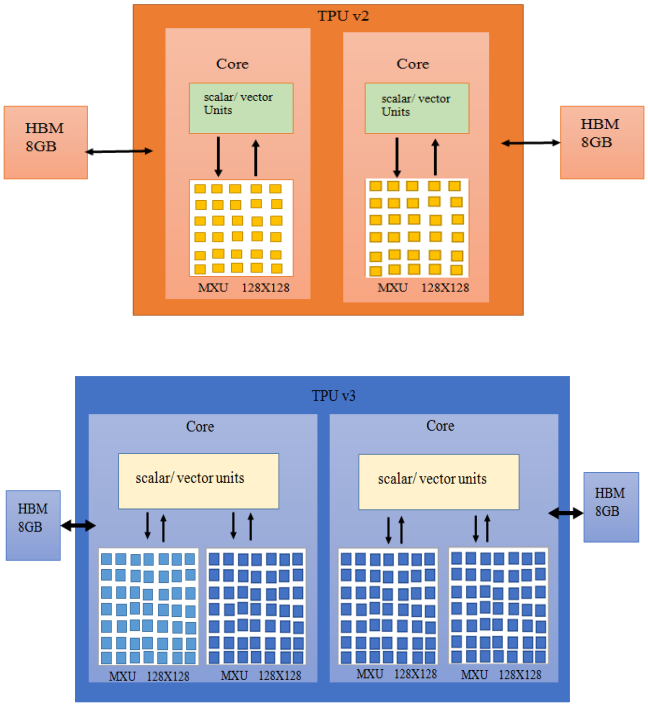


Fig.9. An overview of TPU v2 and TPU v3 chips

3) 3rd Generation TPU

In 2018, Google launched 3rd generation TPU which is twice as powerful as second generation TPU. In 3rd generation, TPU-pod demonstrate almost 8 times better throughput compared to 2nd generation TPU. 3rd generation TPU offers up to 420 Tera Flops compute performance along with 128 GB high memory bandwidth. TPU pod based on 3rd generation TPU can deliver up to 100 peta ops, along with 32 TB memory bandwidth through 2D toroidal(torus) mesh network.

a) TPU Architecture

TPUs (Tensor Processing Units) are designed to provide maximum performance with respect to execution of machine learning models . TPUs internally use idea of tensors. A tensor is formed by generalized vectors and matrices as mentioned below. In a nutshell TPUs use tensors to optimize ML-based algorithms.

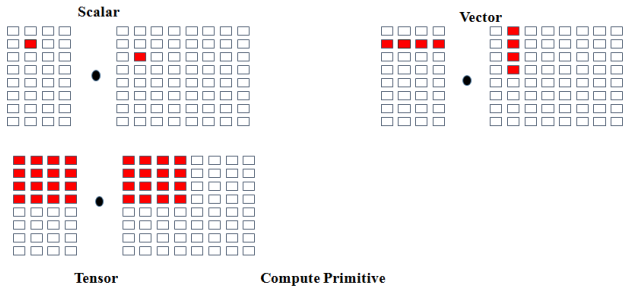


Fig.10. Tensors

3rd generation TPU, it includes 4 TPU chips. Internally each chip includes:

- 2 cores per chip.
- 22.5 Tera ops per core. Scalar unit, Vector unit
- 4 chips contribute 180 Tera ops.

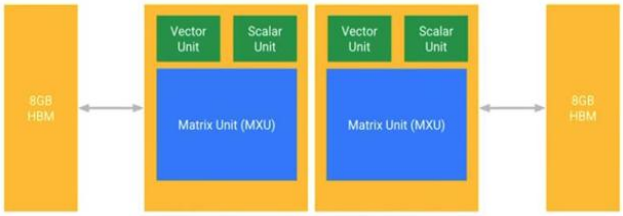


Fig.11. TPU chip layout

Each matrix unit includes 128*128 systolic arrays [7]. Google introduced new floating point format referred as bfloat16 even though it also uses float32 accumulator. In float32 it has 8 exponent bits in bfloat16 it also has 8 exponent bits whereas in float16 it has 5 exponent bits. [5]. Details about bfloat16 are referred below:

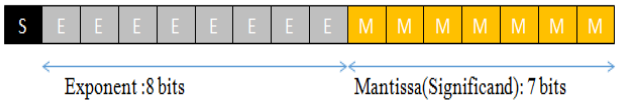


Fig.12. TPU - bfloat16 format

IEEE standard float32 (single precision) and bfloat16 have same range since they have equal number of exponent bits. In matrix unit, 128*128 systolic array is used to perform matrix computations very fast by using intermediate results in parallel.

b) TPU Benchmarks

For benchmarking TPU, authors considered ImageNet dataset. ImageNet data[**] includes 14 million images, which are annotated into 1000 classes depending on the object present in the image. ResNet-50, inception V3 are various models used for the ImageNet challenge.

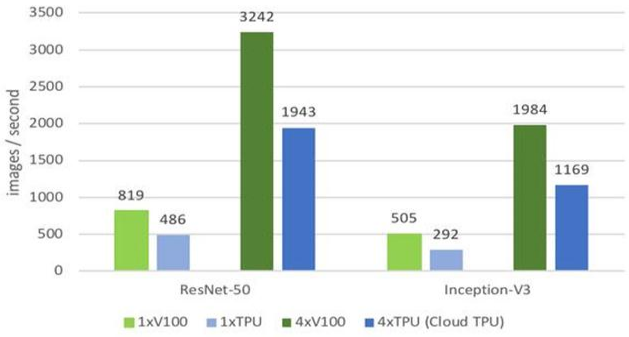


Fig.13. TPU vs. NVIDIA Tesla v100 GPU

We can observe TPU performs well compared to GPU in terms of training time. Now check cost of training the model in case of ImageNet challenge. DAWN Bench, measures different models with respect to training cost, accuracy and training time. Below are the results from DAWN bench based on TPU.

AmoebaNet-D on cloud TPU:

- Final accuracy - 93%
- Training time - 7.5 Hrs.
- Training cost - $49.3

ResNet-50 on TPU:

- Final accuracy - 93%
- Training time - 8.9 Hrs.
- Training cost - $59

c) Edge TPU

In 2018, Google launched Edge TPU, ASIC designed for inference and training of neural networks on edge computing. Especially in the field of Internet of Things, Edge TPU can be used to perform inference using Tensorflow Lite. Edge TPU was targeted to perform well under small physical environment with minimum energy footprint.

C. Precision and Number Format

For designing neural network hardware significant considerations should be considered. Firstly, balance between precision i.e. number of bits and cost of memory area. Secondly suitable number format(FP32, FP16, and INT8) should be chosen so that dynamic range is large enough for general purpose applications. In most cases precision is limited to 16-bits for neural network and 8-bits fixed points for outputs. If we use 16-bits precision instead of 32-bit precision it shows faster computations, since processors will in general have more throughput at lower resolution. If model is trained by using lower precision it also increases bandwidth because small amount of data is fetched at each computation. For RNN it may be required more that 16-bit precision. However precision cannot reduced too much because the network will not train and will not give accuracy as needed.

Precision is needed for training phase it is important to keep precision as high as possible. But, propagation phase requires low precision.

According to researchers it is possible to train neural network by integer weights because integer multiplication can be implemented more effectively than floating points. There are some algorithms which uses power-of-two integers strategy [56] because multiplications in neural network can be reduced to series of shift operations. Floating point gives maximum dynamic range so it is suitable for any application[21]. But floating point requires more cycles for computation than integer. So most Neurocomputers consider fixed-point representation where only integers are used and position after decimal point is handled by other hardware or software. Another method for representation bit stream arithmetic is also used[57].

Neural networks improves accuracy but also increase number of parameters and model size. This increases memory demands and computing bandwidth. The new techniques for efficient neural networks are developed like compact low precision data types which is less than 32-bits. Furthermore, researchers also used extremely low-precision two-bit ternary neural network where values are (+1,0,-1) and one-bit binary neural network where values are (+1,-1)

D. Optimization Techniques

Researchers are modifying the ML algorithms to making it more hardware-friendly while maintaining accuracy. The main focus lies on reducing computation, data movement and storage requirements [9]. The optimization techniques are as follows:

1) Precision Reduction

The default size for programmable platforms such as CPUs and GPUs is often 32 bit or 64 bits with floating point representation during training but using inference it is possible to use fixed point representation and significantly reduce the bit-width to save energy and reduce space requirement while ensuring optimal throughput.

2) Pruning

Machine learning models require large amount of computing, storage, power which is bottleneck to run models on edge devices like mobile devices and browsers with limited computational resources. So pruning is used for inference to efficiently produce machine learning models in smaller size, more effective, more faster and power efficient way. If every neuron in the layer below has a connection to the layer above this signify the presence of dense floating point multiplication operation requirement . But sparse models are easier to compress, and skip zeroes during inference for latency improvements.

Finding the rank of neuron how much they utilized, we can then expel low ranking neurons from the network resulting in smaller and faster model. After the pruning, the accuracy or precision will drop and the network is normally pruned-trained-pruned iteratively to recover. If lot of pruning is done at once then network might damaged so it won't be able to recover. Weight pruning and unit or neuron pruning are methods of pruning.

3) Deep Compression

Neural networks are both computationally intensive so it is hard to deploy on edge devices with less hardware resources. So deep compression is used to reduce storage and energy required to run inference on large neural networks to deploy on mobile devices. AlexNet and VGG-16 uses large number of parameters which is of 200MB-800MB.Deep compression contains three strategies-pruning, weight sharing, Huffman coding for compressing huge weight matrix. First, the model is fully trained. Then neural network is pruned by removing weights less than threshold. Then the remaining weights are quantized by calculating centroids by some clustering algorithm. Then Weights are_approximated to nearest centroids of cluster, by using weight sharing. After quantization only the centroids and the cluster index matrix maps weights to centroids. Lastly, all indices are compressed using Huffman coding. On ImageNet, the approach reduced the storage required by AlexNet by 35 times (from 240 MB to 6.9 MB) without losing accuracy. Several experiments were conducted for this. Researchers first calculated the prediction accuracy of AlexNet and VGG-6 [39] before and after compression. They were able to compress model by 35x and 49x with no reduction in accuracy. Pruning and quantization work well and give better compression.

After analysis on the compression rates, authors conducted experiments on speedup and energy consumption. Network shows different computation on batching and non-batching inference. Computation of batched inference is control by matrix-matrix multiplication whereas non-batched is controlled by matrix-vector multiplication. Ratio of memory access to computation is higher in Matrix-vector multiplication that is reducing memory access gives better speedup(3-4x). Matrix-matrix multiplication take advantage of cache locality efficiently. That is for batched inference deep compression performs worse than uncompressed ones and similar for energy consumption as well.

4) Data Quantization

One way to reduce machine learning computation demands and increase power efficiency is through quantization. Quantization is a simple term which contains multiple strategies to convert large set of input values to smaller set of output values. In general quantization gives number of bits to represent information. Quantization is the process of approximation of deep neural network that

uses floating point numbers by a neural network of fixed point numbers or low bit width numbers.

Neural network consists of activation nodes, the connection between each node, and its associated weights for each connection. Running neural network on hardware can easily result in many millions of multiplication and addition operations. So lower bit numerical operations with quantized parameters merged with quantizing transitional calculations of a neural network results in large computational gains and higher performance.

In addition to the performance, quantized neural network increases power efficiency by reducing memory access cost and increasing computing efficiency. If we use lower bit quantized information it requires fewer data movement, both on-chip or off-chip, which reduces memory bandwidth and saves energy. Lower precision operations such as an 8-bit integer multiplication vs. 32-bit floating point multiplication, consumes minimum energy and increases computing efficiency by reducing power consumption. Also reduces number of bits for representing the models parameters results in less memory.

The outputs (based on an analysis of various strategies with different neural network architectures) shows this [40] dynamic-precision quantization is much more good compared with static-precision quantization. With the help of dynamic-precision quantization, they may utilize considerably shorter representations of operations while yet accomplishing equivalent accuracy.

5) Low Rank Optimization and Trained Ternary Quantization

The issue with Convolutional neural systems is their costly test time assessment which leads the models unrealistic into the real world systems. For instance, a cloud administration needs to process huge number of new submission of requests per second; edge devices for example mobile phones and tablets generally have CPUs or low-end GPUs only; some image recognition functions like object detection are still time-consuming for the processing a particular image, even on a high-end GPUs. For this problems, it's of practical significance to speed up the test-time computation of CNNs.

On the basis of minimizing the reconstruction error of non-linear responses, subject to a low-rank constraint that can be utilize to reduce computation. For solving the challenging constrained optimization problem, the researchers decomposed it into two possible subproblems and solved in iterative manner. Then they suggest to minimize an asymmetric reconstruction error, which efficiently reduced the accumulated error of multiple approximated layers.

One more algorithm that can resolve the deployment problems of large deep neural network models on edge devices with low power budgets is trained ternary quantization, which can reduce the precision of weights in neural networks to ternary values. Firstly, train a model or take a trained model. Then copy full precision weights that you want to quantize, replace them by ternary values(-1,0,+1) using some heuristics. Then repeat until convergence. Trained ternary quantization has very less accuracy degradation and may even increase the accuracy of a few models on CIFAR-10 dataset and AlexNet on ImageNet[40] dataset. In this paper, an AlexNet network is trained from basic, which means it's as simple as training a ordinary, full-precision model.

E. Spiking Neural Network

It is possible to build a computer by using organic and synthetic neurons and synapses. Programmers can use organic Neuromorphic neurons and synapses to model artificial neurons and networks. Now a days it is possible to use Neuromorphic spiking neural network systems which have the potential to calculate complex tasks more efficiently. Spiking neural nets implement ML algorithms. Currently, GANs can be solved using spiking neural nets. Their deep neural nets consists of two nets - generator and discriminator. One CNN generates the new data instances. They calculate the probability given feature x and label Y by random noise. Discriminative algorithm calculate the probability of Y given X .There is an electrical potential difference between interior and exterior of the cell . This is called membrane potential of the cell. Changes in the potential membrane of the cell is used to code and transmit when it reaches particular value. This activates specific neuron. Spike latency designates that the activation effect occurs after a delay time. Organic or synthetic neural networks are stochastic. The activation function is ReLU. Spiking neurons has two properties. First, the neuron value is membrane potential. Second, each neuron outputs binary spike. Spiking neural networks have potential to calculate complex tasks more efficiently. SNNs on Neuromorphic hardware shows properties like less power utilization, fast inference, and efficient data processing. This helps for efficient implementation of neural network.

Brain-inspired processors aim at achieving nearer the storage and the computational components to effectively assess deep learning algorithms. Now a days, SNN, a formation of cognitive methods uses computational primitives mimicking neuron and synapse working principles, has become an essential component of deep learning. SNN are expected to improve the computational performance and efficiency of neural networks, but SNN are best suited for hardware able to support their temporal dynamics [34].

CONCLUSION

Hardware platforms for deep neural network is essential for taking advantage of parallelism along with specially designed, efficient software libraries. While designing hardware for machine learning, important consideration should be made for the choice of precision, number format and type of neurocomputer. Neural network hardware is also designed using CPUs, GPUs, FPGAs and ASICs depending on the performance requirement. The hardware built using these technologies may have structural and behavioral differences and hence optimization and careful design is necessary. More extensive technologies such as eDRAM and ReRAM are being used for speed-up and to overcome conventional design problems. As AI applications are becoming larger and complex, demand for ML specific devices will increase with time.

As Google claims [3], TPUs advanced the Moore's law prediction by almost 7 years. Recent improvements in ASICs, produced better results with respect to training time and training cost. As of today, we can train ImageNet data in 18 minutes [4], which is not feasible five years ago, since ImageNet has almost 14 million images. As discussed, specialized hardware for Machine learning is of primary focus, since the demand for deep neural networks is increasing rapidly. ASICs though have concerns like they are optimized to do one particular task. In case of ImageNet, Google's TPU stood first using only 12.6$ (Next highest is 27$) on ImageNet data with some compromise on accuracy.

In case of Neural networks, since it involves lot of matrix multiplications specialized hardware manufacturers are focusing on parallelizing the ALU operations to improve performance. In last 10 years, growth of deep Neural networks posed challenges and opportunities to the computer architects.

Funding This study was not funded by any funding sources.

Compliance with Ethical Standards

Conflict of interest The authors declare that they have no conflict of interest.

REFERENCES

[1] FPGA architechture. Fpga architechture- diagram.

[2] TPU architechture. Google tpu.

[3] TPU architechture. Moore's law tpu.

[4] TPU architechture. Tpu - imagenet performance analysis.

[5] TPU architechture. Tpu - marix unit.

[6] Luis Ceze, Mark D Hill, and Thomas F Wenisch. Arch2030: A vision of computer architecture research over the next 15 years. arXiv preprint arXiv:1612.03182, 2016.

[7] David, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, et al. In-datacenter performance analysis of a tensor processing unit. IEEE.

[8] Online available: Tech differences web page. Central processing unit vs graphic processing units.

[9] Jawandhiya, P., 2018. Hardware design for machine learning. International Journal of Artificial Intelligence and Applications (IJAIA), 9(1), pp.63-84.

[10] Jouppi, N.P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., Bates, S., Bhatia, S., Boden, N., Borchers, A. and Boyle, R., 2017, June. In-datacenter performance analysis of a tensor processing unit. In Proceedings of the 44th Annual International Symposium on Computer Architecture (pp. 1-12).

[11] Krizhevsky, A., Sutskever, I. and Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).

[12] Online Availabel:Webopedia Web page. Central processing unit.

[13] Sze, V., Chen, Y.H., Emer, J., Suleiman, A. and Zhang, Z., 2017, April. Hardware for machine learning: Challenges and opportunities. In 2017 IEEE Custom Integrated Circuits Conference (CICC) (pp. 1-8). IEEE.

[14] Online available:https://www.welcome.ai/tech/machine-learning/graphcore-accelerated-machine-learning

[15] Jia, Z., Tillman, B., Maggioni, M. and Scarpazza, D.P., 2019. Dissecting the Graphcore IPU Architecture via Microbenchmarking. arXiv preprint arXiv:1912.03413.

[16] Li, B., Doppa, J.R., Pande, P.P., Chakrabarty, K., Qiu, J.X. and Li, H., 2020. 3D-ReG: A 3D ReRAM-based Heterogeneous Architecture for Training Deep Neural Networks. ACM Journal on Emerging Technologies in Computing Systems (JETC), 16(2), pp.1-24.

[17] Chi, P., Li, S., Xu, C., Zhang, T., Zhao, J., Liu, Y., Wang, Y. and Xie, Y., 2016. Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory. ACM

SIGARCH Computer Architecture News, 44(3), pp.27-39.

[18] Lu, J., Young, S., Arel, I. and Holleman, J., 2014. A 1 TOPS/W analog deep machine-learning engine with floating-gate storage in 0.13 μm CMOS. IEEE Journal of Solid-State Circuits, 50(1), pp.270-281.

[19] Chen, Y., Luo, T., Liu, S., Zhang, S., He, L., Wang, J., Li, L., Chen, T., Xu, Z., Sun, N. and Temam, O., 2014, December. Dadiannao: A machine-learning supercomputer. In 2014 47th Annual IEEE/ACM International Symposium on Microarchitecture (pp. 609-622). IEEE.

[20] Available:https://cloud.google.com/blog/products/ai-machine-learning/bfloat16-the-secret-to-high-performance-on-cloud-tpus

[21] Liao, Y., 2001. Neural networks in hardware: A survey. Department of Computer Science, University of California.

[22] Available:https://medium.com/@CPLu/should-we-all-embrace-systolic-array-df3830f193dc

[23] Sze, V., Chen, Y.H., Emer, J., Suleiman, A. and Zhang, Z., 2017, April. Hardware for machine learning: Challenges and opportunities. In 2017 IEEE Custom Integrated Circuits Conference (CICC) (pp. 1-8). IEEE.

[24] Ramacher, U., Raab, W., Anlauf, J., Hachmann, U., Beichter, J., Brüls, N., Wesseling, M., Sicheneder, E., Männer, R., Gläß, J. and Wurz, A., 1993. Multiprocessor and memory architecture of the neurocomputer SYNAPSE-1. International Journal of Neural Systems, 4(04), pp.333-336.

[25] Dias, F.M., Antunes, A. and Mota, A.M., 2003. Commercial hardware for artificial neural networks: A survey. IFAC Proceedings Volumes, 36(12), pp.189-196.

[26] Cho, J.W. and Lee, S.Y., 1998, May. Analog neuro-chips with on-chip learning capability for adaptive nonlinear equalizers. In 1998 IEEE International Joint Conference on Neural Networks Proceedings. IEEE World Congress on Computational Intelligence (Cat. No. 98CH36227) (Vol. 1, pp. 581-586). IEEE.

[27] Gaurav Nakhare, (2017, July 31), "Hardware options for machine/deep learning", MS&E 238 Blog.

[28] Dally, W., 2015. High-performance hardware for machine learning. NIPS Tutorial.

[29] Nvidia, "Why GPUs?". [Online] Available: http://www.fmslib.com/mkt/gpus.html,

[30] Nurvitadhi, E., Sheffield, D., Sim, J., Mishra, A., Venkatesh, G. and Marr, D., 2016, December. Accelerating binarized neural networks: Comparison of FPGA, CPU, GPU, and ASIC. In 2016 International Conference on Field-Programmable Technology (FPT) (pp. 77-84). IEEE.

[31] Chi, P., Li, S., Xu, C., Zhang, T., Zhao, J., Liu, Y., Wang, Y. and Xie, Y., 2016. Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory. ACM SIGARCH Computer Architecture News, 44(3), pp.27-39.

[32] Basu, A., Shuo, S., Zhou, H., Lim, M.H. and Huang, G.B., 2013. Silicon spiking neurons for hardware implementation of extreme learning machines. Neurocomputing, 102, pp.125-134.

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

[34] Bouvier, M., Valentian, A., Mesquida, T., Rummens, F., Reyboz, M., Vianello, E. and Beigne, E., 2019. Spiking neural networks hardware implementations and challenges: A survey. ACM Journal on Emerging Technologies in Computing Systems (JETC), 15(2), pp.1-35.

[35] Nuno-Maganda, M.A., Arias-Estrada, M., Torres-Huitzil, C. and Girau, B., 2009, June. Hardware implementation of spiking neural network classifiers based on backpropagation-based learning algorithms. In 2009 International Joint Conference on Neural Networks (pp. 2294-2301). IEEE.

[36] Balaji, A., Das, A., Wu, Y., Huynh, K., Dell'Anna, F.G., Indiveri, G., Krichmar, J.L., Dutt, N.D., Schaafsma, S. and Catthoor, F., 2019. Mapping spiking neural networks to neuromorphic hardware. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 28(1), pp.76-86.

[37] Gaier, A. and Ha, D., 2019. Weight agnostic neural networks. In Advances in Neural Information Processing Systems (pp. 5365-5379).

[38] Online Availavle:https://heartbeat.fritz.ai/the-5-algorithms-for-efficient-deep-learning-inference-on-small-devices-bcc2d18aa806

[39] Han, S., Mao, H. and Dally, W.J., 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149.

[40] Zhu, C., Han, S., Mao, H. and Dally, W.J., 2016. Trained ternary quantization. arXiv preprint arXiv:1612.01064.

[41] Qiu, J., Wang, J., Yao, S., Guo, K., Li, B., Zhou, E., Yu, J., Tang, T., Xu, N., Song, S. and Wang, Y., 2016, February. 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).

[42] Karl Freund, (2017, March 3), "A machine learning landscape: where AMD, Intel, Nvidia, Qualcomm and Xilinx AI engines live", Forbes. [Online]. Available : https://www.forbes.com/sites/moorinsights/2017/03/03/a-machine-learning-landscape-where-amdintel-nvidia-qualcomm-and-xilinx-ai-engines-live/#4436108a742f.

[43] Chen, J.X., 2016. The evolution of computing: AlphaGo. Computing in Science & Engineering, 18(4), pp.4-7.

[44] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M. and Kudlur, M., 2016. Tensorflow: A system for large-scale machine learning. In 12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16) (pp. 265-283).

[45] Markidis, S., Der Chien, S.W., Laure, E., Peng, I.B. and Vetter, J.S., 2018, May. Nvidia tensor core programmability, performance & precision. In 2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW) (pp. 522-531). IEEE.

[46] Mojumder, S.A., Louis, M.S., Sun, Y., Ziabari, A.K., Abellán, J.L., Kim, J., Kaeli, D. and Joshi, A., 2018, September. Profiling dnn workloads on a volta-based dgx-1 system. In 2018 IEEE International Symposium on Workload Characterization (IISWC) (pp. 122-133). IEEE.

[47] Gawande, N.A., Daily, J.A., Siegel, C., Tallent, N.R. and Vishnu, A., 2018. Scaling deep learning workloads: Nvidia dgx-1/pascal and intel knights landing. Future Generation Computer Systems.

[48] Hoy, M.B., 2018. Alexa, Siri, Cortana, and more: an introduction to voice assistants. Medical reference services quarterly, 37(1), pp.81-88.

[49] Klein, S., 2017. IoT Solutions in Microsoft's Azure IoT Suite. Berkeley, CA: Apress.

[50] Shah, H., 2011, April. Turing's misunderstood imitation game and IBM's Watson success. In Keynote in 2nd Towards a Comprehensive Intelligence test (TCIT) symposium at AISB.

[51] Cyphers, S., Bansal, A.K., Bhiwandiwalla, A., Bobba, J., Brookhart, M., Chakraborty, A., Constable, W., Convey, C., Cook, L., Kanawi, O. and Kimball, R., 2018. Intel ngraph: An intermediate representation, compiler, and executor for deep learning. arXiv preprint arXiv:1801.08058.

[52] Markram, H., 2006. The blue brain project. Nature Reviews Neuroscience, 7(2), pp.153-160.

[53] Rojas, R., 2013. Neural networks: a systematic introduction. Springer Science & Business Media.

[54] Online Available:https://www.iis.fraunhofer.de/en/ff/kom/iot/embedded-ml/neuromorphic.html

[55] Hoy, M.B., 2018. Alexa, Siri, Cortana, and more: an introduction to voice assistants. Medical reference services quarterly, 37(1), pp.81-88.

[56] Marchesi, M., et al., "Fast neural networks without multipliers". IEEE Transactions on Neural Networks, 1993. 4(1): p. 53-62

[57] Jihan Zhu and Peter Sutton, "FPGA Implementations of Neural Networks – A Survey of a Decade of Progress", Y. K. Cheung P., Constantinides G.A. (eds) Field Programmable Logic and Application, FPL 2003, Lecture Notes in Computer Science, vol. 2778. Springer, Berlin, Heidelberg.