Hauptseminar Hochleistungsrechner: Aktuelle Trends und Entwicklungen
Winter Term 2017/2018

Accelerators for Deep Learning

Mirjam Trapp
Ludwig-Maximilians-Universität München

19.01.2018

Abstract

Deep learning has become an increasingly important topic in computer science, accelerators aiming at improving performance of deep learning applications have become a subject of interest. In this paper, we are taking a look at three different current accelerators considered to be representative of general directions for accelerators in deep learning, namely NVIDIA Tesla GPU, Google TPU and IBM TrueNorth and try to draw some comparison between them. We compare these three regarding their usability in deep learning and conclude that each has its usefulness for different aspects which reflects their primary design point. However, those three accelerators are built for different purposes and one cannot simply replace one of them by one of the others.

1 Introduction

Artificial Intelligence in general and the related topic of machine learning have lately been a big topic due to increasing technological possibilities, even though the subject itself has been discussed for decades.
A specific approach to artificial intelligence which is recently becoming an item of discussion due to new implementations is artificial neural networks.
Deep Learning can generally be seen as a subtopic of neural networks, which can in its simplest form be described as a set of connected processors, with those connections typically being weighted [13].
Deep Learning is implicating the existence of a certain hierarchy in the neural network, with different levels also providing different levels of abstraction. As for many topics in computer science, the increasing amount of data accessible to us is contributing to the improvement of prediction regarding statistical methods of computer science. [14]
There are different types of deep learning, the most common one being supervised learning, where the neural network is trained by using a set of data that has already been classified in advance in order to try to adjust its weights to minimize classification errors, most commonly using Stochastic Gradient Descent (SDG). [14]
A slightly different version of unsupervised learning in neural networks are backpropagation architectures, which aim at adjusting the weights of the connections as specifically as possible, tracking the impact of the specific weights on the total error regarding the outcome by going backwards from neuron to neuron, following the connections. This way, it is possible to change only those weights having an actual effect on the undesired outcome. [14]
There are also approaches of unsupervised learning with neural networks, as in having the network

find own rules based on which to separate the given training data. A classic example for this is Hebbian Learning. [16]
Hardware accelerators were originally developed to be tools mainly for faster graphic processing in a computer - which they are still used for today - , since rendering and displaying graphics requires a limited set of instructions to be performed very frequently and could therefore be optimized by running on a specialized device.
They are for the same reason also perfectly suitable for increasing performance of deep learning applications in artificial neural networks, where the same type of operation, namely matrix multiplications, usually has to be performed over and over again.

2 Motivation

Today the most commonly used accelerators for deep learning are GPUs.
As pointed out by LeCun in [14] the first deep learning application that used backpropagation was successfully built in 2009 using GPUs.
Following Moore's Law, the performance capabilities of GPUs have steadily increased over the last decades making them a more useful acceleration tool then ever, without being limited to their original task of fast processing of graphic images anymore.
Among classic GPUs the most well know company would probably be NVIDIA who have also more recently been in focus due to the increasing public interest in self-driving cars. NVIDIA's Tesla GPU range is for instance used in the NVIDIA DGX and NVIDIA Drive PX systems [23] that are used by well-known car companies such as Audi, Mercedes and Tesla.
Their newest accelerator architecture is NVIDIA Volta which was launched in June, 2017.
However, there are new approaches entering the market targeting the field of deep learning with its specific requirements. The Google TPU for example received massive media attention in 2016 when Google's AlphaGo program managed to beat Lee Sedol, one of the world's best Go players [19]. The deep learning processes needed to train the algorithm were reportedly performed using Google TPU [20].
Interestingly enough, Google TPU is an application-specific integrated circuit (short ASIC). In October 2017, it was even reported that Google had developed a new version of AlphaGo that was able to beat its predecessor that had succeeded against Lee Sedol [20].
Another interesting development would be the efforts to make accelerators that are directly inspired by the human brain. There seems to be a general public interest in simulating the functioning of the human brain, presumably partly due to the potential medical uses of such a model. There are several projects that aim at building such simulations, most commonly known the Human Brain Project which is supported by the European Union [21] and the BRAIN Initiative [22] that is funded by the US government.
Technology developed in such efforts, such as neuromorphic chips, are however not necessarily limited to being a part of large-scale brain simulations, but can also quite obviously be used for deep learning tasks in general. One good example for this would be IBM TrueNorth which got developed as part of the DARPA SyNAPSE program, another program aimed at the development of neuromorphic technology in the US [11].
Though not being publicly released yet, results of tests performed by IBM have shown that the chip performs very well on image recognition tasks [17].

3 Methodology

The following chapter provides a deeper insight into three different types of accelerators, examining their architecture, the corresponding software environment and benchmarks.
The accelerators are furthermore compared regarding performance, efficiency and usability. The accelerators as examples for new developments in those three different categories of GPUs, ASICs and neuromorphic chips are NVIDIA Tesla V100, Google TPU and IBM TrueNorth.

This is mainly due to the fact that those three accelerators have received a lot of public attention prior and after their release, being praised as futuristic takes on accelerators.
Conveniently, NVIDIA's Tesla V100, Google TPU and IBM TrueNorth are representing three very different approaches, which makes them interesting to review in comparison to each other, even though a direct comparison may be difficult due to their different architectures and the different objectives they were designed for.

4 Analysis of Accelerators for Deep Learning

The following chapter will discuss the architecture, software environment and benchmarks of NVIDIA V100, Google TPU and IBM TrueNorth and further draw a comparison between them.

4.1 NVIDIA V100

This section will give a description of the architecture, software environments and benchmarks of the NVIDIA V100.

4.1.1 Architecture

NVIDIA V100 is the first accelerator using NVIDIA's new volta architecture and will first be used in the NVIDIA Titan V graphics card that got announced by NVIDIA on December 7th, 2017.
It comes in two different versions depending on the system interface that is used, Tesla V100 PCIe and Tesla V100 SKM2, the first one using PCIe Gen3 and the second one using NVIDIA NVLink. Tesla V100 PCIe comes with 640 NVIDIA tensor cores and achieves an interconnect bandwidth of 32GB/sec, whereas Tesla V100 SKM2 has 5120 CUDA cores and an interconnect bandwidth of 300GB/sec. [8]
According to NVIDIA, the Tesla V100 is highest performing parallel computing processor thus far. Regarding the accelerator's architecture, it consists of multiple GPU processing clusters, each of which is containing seven Texture Processing Clusters, as well as 84 Volta Streaming Multiprocessors and eight 512-bit memory controllers. [7]
According to the authors of the Tesla V100 whitepaper, a main goal when designing the accelerator was to also increase its efficiency, which is why the V100 can run in two different modes: Maximum Performance Mode and Maximum Efficiency Mode, stating that the GPUs could still achieve a potential performance of 75 to 85 percent of its full performance when running in Maximum Efficiency Mode. [7]
A new feature of the V100 compared to previous NVIDIA architectures is the fact that the Streaming Multiprocessor can perform 32-bit floating point operations and 32-bit integer operations in parallel, since it has separate cores for these two purposes. This design was chosen in order to improve deep learning matrix arithmetics. [7]

4.1.2 Software Environment

The compatible software APIs for this GPU include NVIDIA CUDA and OpenCL and it is suited for several deep learning frameworks such as Caffe. [8]
The most commonly known software framework used for NVIDIA GPU programming would be the CUDA toolkit.
CUDA was developed in 2006 and is supposed to be a general-purpose parallel computing platform typically with a software environment using C, though other languages such as FORTRAN and C++ are supported as well [10].
The main objective when creating CUDA was to enable scalable parallelism in order to maximize the overall performance. Therefore, threads are organized in a hierarchical structure, with several threads building a thread block and several thread blocks forming a grid. The premise for thread blocks is that they have to be fully independent from one another, in other to being able to schedule execution flexibly. Within thread blocks, threads coordinate their actions via shared memory. [10]
In a combined CPU andGPU architecture, those threads generally run on a different device (GPU)

than their host (CPU), with device and host typically not sharing any memory. The GPUs accessible to the host are identified via a so-called compute capability, which is essentially a version number, from which the CPU can derive information about the GPU's hardware and therefore what functions are supported by that specific GPU. [10]
Grids of kernel functions that are invoked by the CPU are then distributed for execution onto the GPUs as evenly as possible. Thread blocks are further grouped into warps with regards to the GPU hardware. [10]
Specifically in the Tesla Volta architecture, there is so-called Independent Thread Scheduling, giving each thread its own execution state, program counter and call stack in order to minimize the risk of deadlocks within warps. There is also a feature called schedule optimizer, through which it is for instance possible to build even smaller subgroups of threads within warps. [10]
There are several newer extensions to CUDA, for example Cooperative Groups, which is enabling the user to specify in detail to which degree which thread shall be able to communicate with what other threads. [10]
Another extension is CUDA Dynamic Parallelism, which confers the possibility to create new work tasks on the GPU itself instead of necessarily having to be called by the host [10].

4.1.3 Benchmarks

The peak performance given by NVIDIA is 7 TFLOPs/sec on V100 PCIe and 7.8 TFLOPs/sec on V100 SKM2 regarding double precision as well as 14 TFLOPs/sec on V100 PCIe and 15.7 TFLOPS on V100 SKM2 for single precision, causing a maximum power consumption of 250W on V100 PCIe and 300W on V100 SKM2. [8]
Furthermore, it is stated that the peak performance for training and inference is 125 tensor TFLOPs [7].

4.2 Google TPU

In the following, the architecture, software environments and benchmarks of Google TPU shall be discussed.

4.2.1 Architecture

Google TPU was built by Google in order to have a customized accelerator for neural networks, regarding the increasing need of computational resources regarding Google's own data centers. The main idea was to work with quantization to lower the cost of matrix multiplications in neural networks. Via quantization, floating point values are reduced to 8-bit integers, which makes operations on them computationally much cheaper. [12]
Specifically, quantization maps floating point values to integer values based on the position of that floating point value in the total range of values that the floating point number could have - given a floating point range from -10.0 to 30.0, the value 30.0 would then be mapped to 255 in the integer representation [9].
It should be noted that quantization works for inference - the model classifying some given input based on its internal classification rules - but usually not for training itself, where the aim is to optimize those classification rules. This is due to the general robustness towards errors that the neural network should have when classifying new input. [9]
The accelerator's design is explicitly specialized for neural network prediction, therefore it is using a CISC architecture. Instead of a classic ALU that stores and fetches values from registers, Google TPU uses a systolic array in which several cells are connected to and pass on result values between each other. Each cell can also store its result value without the need of using a register. [12]
As described in [15], Google TPU furthermore consists of a matrix multiply unit, with the weights for the addition and multiplication operations being stored in a weight FIFO.
The TPU communicates with the host CPU over a PCIe Gen3 x16 bus, transferring 14 GiB/sec [15].
Google TPU has a size of 28nm and needs 40W of energy, running on a 700MHz frequency. It comes as an external card that matches a SATA harddisk slot. [12]

4.2.2 Software Environment

The software library used for running Python code on Google TPU is called TensorFlow and was made specifically for Google's machine learning applications, though having been turned into an open source platform lately.
TensorFlow consists of a number of different APIs enabling the user to build and train their own neural networks at different levels of control, with the most low level one being TensorFlow Core, and the higher level APIs being built on top of TensorFlow Core. The concept giving the library its name is tensors, which are basically just an array which can have a varying number of dimensions. The other concept that shows in TensorFlow's name is computational graphs, which represent a series of operations that will be run by a TensorFlow program after building the graph. TensorFlow also offers a graphic visualization tool for these computational graphs, called TensorBoard. [9]
Computational graphs are generally built using three different functions, inference, loss and training, where, as the names would suggest, the graph is build by the inference function, loss calculation is included into the graph by the loss function and training gives the functions used for minimization of the loss. After the training stage, an evaluation graph can be generated to enable the user to revise one's program. [9]
Whereas the most low level API TensorFlow Core requires advanced skills with regards to programming and machine learning, the high-level API tf.estimator is recommended to unexperienced users and takes a lot of work from the developer. It offers a range of predefined classification models that can be used on one's own data sets for training, while still offering the possibility to create one's own classification functions as well as defining the preprocessing operations applied to the data beforehand. It also comes with several functions to convert data sets into different tensor formats. [9]
As mentioned above, TensorFlow is not restricted to Google TPU or Google's neural network applications in general, but can also be used on classic CPU/GPU architectures. There are therefore a number of benchmarks available for different NVIDIA GPUs testing common classification models [9], unfortunately it apparently has not been tested for the NVIDIA Tesla V100 yet.
As for more recent developments, there is also a new compiler for linear algebra called XLA being developed which is trying to improve TensorFlow with regards to memory usage, execution speed, portability, etc. [5]

4.2.3 Benchmarks

The peak performance of Google TPU is stated to be 92 Teraops per second on its matrix unit, this is however referring to integer operations due to its architecture [9].
Comparisons with Intel's Haswell Xeon CPU and NVIDIA's K80 GPU on different benchmarks for convolutional neural networks, recurrent neural networks and multilayer perceptrons furthermore showed Google TPU to be 15 to 30 times faster than Intel's CPU and NVIDIA's GPU [5].

4.3 IBM TrueNorth

The following section shall give an overview of the architecture, software environment and benchmarks of the IBM TrueNorth.

4.3.1 Architecture

The IBM TrueNorth chip is described in [11] by one of its developers, Dharmendra Moda. According to him, a main focus in the development process was reducing the power consumption of a chip that is inspired by the human brain, regarding the fact that a hypothetical computer simulating the brain would need around 12GW in energy, whereas the actual human brain only runs on 20W. The IBM TrueNorth chip consists of around one million artificial neurons, connected by 256 million artificial synapses, each of them programmable. This is made possible by 4096 cores in total (parallel and distributed).
Furthermore, there is a customizable on-chip communication network, as well as the possibility

to build a customizable network of chips. IBM TrueNorth is considered to be very energy-efficient, it takes 70mW to run what is described as a typical recurrent network. [11]
The neurosynaptic cores on the TrueNorth chip are implemented as a number of axons which are connected to a number of neurons, where each axon is assigned a so-called activity bit with a corresponding time stamp modeling event flow. Each connection is furthermore weighted. [18]
On the hardware side, the axon-neuron connections are modeled as a 1024 x 256 SRAM crossbar, taking in the axon activity as input (using an input decoder) and giving the neuron spikes as output (using an output decoder) [18].

4.3.2 Software Environment

Along with the chip, IBM also introduced a new programming language with a matching simulator, environment and libraries, which is described in [2] by Amir et al. Seeing that the classic sequential programming model would not be suitable for TrueNorth's architecture, there has been developed a completely new programming paradigm based on the concept of so-called Corelets, which are an abstraction of a neural network, as well as a Corelet Language, Corelet Library and Corelet Laboratory. [2]
The Corelet Language is a turing-complete language, but it is different to common programming languages since the program itself is the neural network specified by the developer with its inputs, outputs, parameters and connectives. The goal of this programming language implementation is to hide the internal processes which are specified by the developer and only show the external in- and outputs to potential users of the corelet. [2]
The whole concept follows a divide-and-conquer approach with each corelet being made of a number of subcorelets, forming a tree with neurosynaptic cores being the leaves of that tree. Therefore, large neural networks are essentially built from smaller neural networks. [2]
The simplest version of a corelet as described in the article is the seed corelet, a program which is basically just one neurosynaptic core, which on its inside consists of neurons and axons, neurons being spike sources and axons being spike destinations. There is also a list called output connector, listing all of the neurons that put out spikes to the external environment of the Seed corelet, as well as a similar input connector list for axons. [2]
When a new corelet is formed from subcorelets, the output connectors of the subcorelets become the new spike sources, whereas input connectors turn into spike destinations. [2]
The main symbols of the Corelet Language are therefore quite obviously neurons, neurosynaptic cores and corelets. As the Corelet Language is object-oriented, this translates to a neuron class, core class and an additional connector class. Here, the neuron class internally consists of the initial membrane potential, reset mode, leaks and thresholds, along with some corresponding set- and get-methods. The neurosynaptic core class internally specifies a vector of 256 neuron objects, another list of 256 axom types and a 256 x 256 matrix showing connections between axoms and neurons. The corelet class quite intuitively contains its subcorelets, neurons, neurosynaptic cores and connectors, with the connector class specifying a list of pins that are storing the inter- an intra-corelet connections of the neural network. [2]
Generally, the communication between neurosynaptic cores is performed using all-or-none spike events that are transported over a message-passing network [2].
The Corelet Language got implemented with MATLAB OOP. Since it is object-oriented, there are features such as objects and classes as described above as well as inheritance and polymorphism. [2]
The Corelet Library is basically a container for premade corelets that can be reused by developers who can also add their own corelets into the repository. When the article [2] was written, there were about 100 corelets in the repository, for instance aggregators, linear filters, a Discrete Fourier Transform, etc. The Corelet Library can be seen as a tree itself, as all corelets are subcorelets to a base class. [2]
Corelet Laboratory is the program environment to create and execute code on. The underlying con-

cepts here are composition and decomposition, with composition being the creation of corelets from smaller elements. In order to actually run such a program, it is necessary to decompose the whole structure of corelets into an actual TrueNorth program that can be compiled and simulated. Corelet Laboratory further offers support for program verification. As in most deep learning frameworks, the simulation results can also be plotted and furthermore visualized in a video. [2]

4.3.3 Benchmarks

It is stated by Merolla et al. in [17] that TrueNorth achieves a peak performance of 46 billion SOPS (synaptic operations per second) per watt, comparing it to the peak performance of the current most energy-efficient supercomputer, which has a peak performance of 4.5 billion FLOPS per watt.
This again points out the great energy efficiency of IBM TrueNorth.

4.4 Comparison

With regard to power consumption, it is quite easy to see that IBM TrueNorth and Google TPU clearly outperform NVIDIA's Tesla V100, with IBM TrueNorth only needing 70mW to run a network and Google TPU which needs 40W compared to the Tesla V100 with a consumption of 250 to 300W.
Regarding the peak performance, the accelerators taken into account in this paper get difficult to compare, mainly due to the use of quantization in Google TPU leading to the fact that only integer operations are performed on the chip. It is likely that Google TPU will outperform NVIDIA's V100 in most neural network prediction tasks, since increasing the efficiency in this field was the reason it got designed by Google.
It would be interesting though to compare IBM TrueNorth to a Tesla V100 GPU using the same classification tasks as the ones that IBM researchers previously used for testing TrueNorth.
Overall, the lack of explicit benchmark values for Google TPU and IBM TrueNorth as well as the different units of measurement associated with the accelerators by the manufacturers make them very hard to compare when it comes to performance.
Taking a look at usability, it is obvious that Google TPU which is specialized on Google's needs for neural network prediction will not come in handy for many other tasks than that, whereas a GPU architecture such as Tesla V100 can be more universally used as seen in their systems for self-driving cars mentioned above.
With IBM TrueNorth, one big issue is the fact that is has not actually been released yet, so it is hard to estimate the degree to which is will be used in a commercial way.
Overall, it can probably be stated that NVIDIA's Tesla V100 is the most universal accelerator out of the three since it can be applied for both training and inference tasks in multiple different settings. Also, there are several different software environments that can be used on Tesla V100, from CUDA to open-source OpenCL and frameworks such as Caffe.
Regarding the corresponding software environments, it will always be a matter of personal preference which one will be preferred by the majority of developers when having to implement one's own applications and it is quite hard to build an opinion on their user-friendliness without having a lot of experience using any of them, since a software development kit generally should be tested by both beginners and advanced developers regarding intuitiveness and the degree of control over one's implementation.

5 Discussion

It is not a big surprise that Google TPU and IBM TrueNorth are more energy-efficient than NVIDIA's Tesla V100 keeping in mind that the first two accelerators were specifically built with the goal to reduce energy consumption, as well as both architectures being designed for very specific purposes in contrast to NVIDIA's Tesla V100 being designed to be a more general-purpose accelerator, making Google TPU and IBM TrueNorth less universally

usable.
On the flip side, the fact that Tesla V100 will outperform the two other accelerators on most tasks - except the ones those two were designed for - is equally little surprising.
Overall, it is definitely difficult to draw a comparison between those three different architectures. Generally spoken, which accelerator should be preferred over the other ones is very much dependent on the actual task one trying to use them for.
In many cases, a combination of different architectures might also be beneficial as in using a GPU architecture for training and a specialized ASIC or a neuromorphic chip for the actual inference tasks.

6 Related Work

In an article from 2015 Rodrigues et al. showed the capabilites of GPUs in comparison to multicore architectures, namely comparing a Gainward GeForce GTX 580 Phantom GPU by NVIDIA to two Intel Xeon X5550 cores, running the same recommender algorithms in already existing versions for CPUs on the Intel Xeon X5550 cores and a CUDA version of them on the Gainward GeForce GTX 580 Phantom. The tested GPU was able to achieve a speedup of maximum 14.8 compared to the multicore implementation. [4]
Another topic that is frequently mentioned regarding accelerators for deep learning is FPGA technology.
In a 2017 article Nurvitadhi et al. discuss in detail the performance of FPGAs in comparison to GPUs in relation to new developments in deep learning. The author argues that upcoming deep neural networks will work with network sparsity at lot more so then done in the past which leads to more irregular parallelism and furthermore use more custom, compact data types. Hence he is suggesting to use FPGAs as an alternative, which are strongly customizable and in addition also typically more energy-efficient than GPUs. His test results show a better performance of current FPGAs on binary, sparse, compact narrow-bandwidth and ternary deep neural networks compared to a modern GPU. [1]
Wen et al. in 2016 suggested a new learning method for IBM TrueNorth in order to improve its performance on inference tasks since TrueNorth is like Google TPU working with low-resultion integers. The authors therefore propose a probability-biased learning method. [6]
The approach led to small improvements regarding accuracy, but an increased overall performance with a speed-up of 6.5 and a reduction of 68.8 percent regarding the cores needed to perform computations [6].
In an article from 2016, Gu et al. also describe attempts to transform caffe, a CUDA-based framework for deep learning on neural networks into an OpenCL version to make deep learning more accessible for hardware architectures which are not based on CUDA [3].
OpenCL is as mentioned before an open source standard and supported by many different manufacturers [3] as well as NVIDIA [8].
The OpenCL version could not compete with the CUDA version yet regarding performance, however the authors point out that there is still a lot of effort needed to deal with the different OpenCL extensions available in the future development of OpenCL caffe [3].

7 Conclusion and Future Work

In summary, this paper gives an overview of three different trends in accelerators, examining NVIDIA Tesla V100 representing classic GPU-based accelerators, Google TPU as an ASIC and IBM TrueNorth as a neuromorphic chip, reviewing their architectures, corresponding software environments and accessible benchmarks, giving an overview of the different accelerators' characteristics and capabilities. It would be difficult to recommend one of the accelerators over the others, since they were all built with different concepts in mind. It is easy to see that the NVIDIA Tesla V100 accelerator is the one achieving overall best performance, whereas TrueNorth is incredibly energy-efficient. Since

Google TPU is an ASIC, it performs very well on its designated task while at the same time being very energy-efficient.
It would obviously be interesting to try to compare all of the three accelerators reviewed in this paper on a set of actual classification tasks, namely running for each accelerator one or several tasks that it is "specialized" on on itself and the two other accelerators to point out strengths and weaknesses regarding tasks that the accelerator wasn't necessarily optimized for, as well as taking a deeper look into similarities and differences between the different architectures.
Yet another processor that has been a recent topic of discussion is Intel's Xeon Phi Knights Mill processor, which is presumably going to be specialized on deep learning as well.
Since Knights Mill was neither yet released nor was there a lot of information available on it when this paper was written, it hasn't been included into my comparison. With that being said, Knights Mill would be an interesting subject for future work on this topic once there is more information on it released.
On the general topic of trends for accelerators in deep learning, it will be interesting to see what the next developments are going to be and how the approaches on architectures are going to evolve. As discussed above in this paper, there are different new models challenging classic GPU accelerators, from neuromorphic chips to ASICs and FPGAs.
FPGAs and ASICs however are naturally specialized constructions for very specific applications and the energy-efficiency of Google TPU and IBM TrueNorth for example comes with the cost of lower precision, which makes them unsuitable for training artificial neural networks, hence why GPUs are still needed for the training stage before performing the actual classification tasks.
Overall, successful architecture will always be tied to the needs of current deep learning applications, so one will have to keep an eye on research developments in deep learning in general to predict future trends in accelerator types and architectures.

References

[1] Nurvitadhi, E., Venkatesh, G., Sim, J., Marr, D., Huang, R., Ong Gee Hock, J., ... & Boudoukh, G. (2017, February). Can FPGAs beat GPUs in accelerating next-generation deep neural networks?. In Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (pp. 5-14). ACM.

[2] Amir, A., Datta, P., Risk, W. P., Cassidy, A. S., Kusnitz, J. A., Esser, S. K., ... & McQuinn, E. (2013, August). Cognitive computing programming paradigm: a corelet language for composing networks of neurosynaptic cores. In Neural Networks (IJCNN), The 2013 International Joint Conference on (pp. 1-10). IEEE.

[3] Gu, J., Liu, Y., Gao, Y., & Zhu, M. (2016, April). OpenCL caffe: Accelerating and enabling a cross platform machine learning framework. In Proceedings of the 4th International Workshop on OpenCL (p. 8). ACM.

[4] Rodrigues, A. V., Jorge, A., & Dutra, I. (2015, April). Accelerating recommender systems using GPUs. In Proceedings of the 30th Annual ACM Symposium on Applied Computing (pp. 879-884). ACM.

[5] Abadi, M., Isard, M., & Murray, D. G. (2017, June). A computational model for TensorFlow: an introduction. In Proceedings of the 1st ACM SIGPLAN International Workshop on Machine Learning and Programming Languages (pp. 1-7). ACM.

[6] Wen, W., Wu, C., Wang, Y., Nixon, K., Wu, Q., Barnell, M., ... & Chen, Y. (2016, June). A new learning method for inference accuracy, core occupation, and performance co-optimization on TrueNorth chip. In Design Automation Conference (DAC), 2016 53nd ACM/EDAC/IEEE (pp. 1-6). IEEE.

[7] NVIDIA. Nvidia Tesla V100 GPU Architecture: The World's Most Advanced

Data Center GPU. Retrieved from http://www.nvidia.com/object/tesla_product_literature.html

[8] NVIDIA. Tesla V100 Performance Guide: Deep Learning and HPC Applications. Retrieved from http://www.nvidia.com/object/tesla_product_literature.html

[9] TensorFlow. (2017). Getting Started With TensorFlow. Retrieved from https://www.tensorflow.org/get_started/get_started

[10] NVIDIA. (n.d.). Programming Guide :: CUDA Toolkit Documentation. Retrieved from http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html

[11] IBM. (n.d.). IBM Research: Brain-inspired Chip. Retrieved from http://www.research.ibm.com/articles/brain-chip.shtml

[12] Google Cloud Platform. (n.d.). An in-depth look at Googles first Tensor Processing Unit (TPU). Retrieved from https://cloud.google.com/blog/big-data/2017/05/an-in-depth-look-at-googles-first-tensor-processing-unit-tpu

[13] Schmidhuber, J. (2015). Deep learning in neural networks: An overview. Neural networks, 61, 85-117.

[14] LeCun, Y., Bengio, Y., LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436-444.

[15] Jouppi, N. P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., ... & 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). ACM.

[16] Song, S., Miller, K. D., & Abbott, L. F. (2000). Competitive Hebbian learning through spike-timing-dependent synaptic plasticity. Nature neuroscience, 3(9), 919-926.

[17] Merolla, P. A., Arthur, J. V., Alvarez-Icaza, R., Cassidy, A. S., Sawada, J., Akopyan, F., ... & Brezzo, B. (2014). A million spiking-neuron integrated circuit with a scalable communication network and interface. Science, 345(6197), 668-673.

[18] Merolla, P., Arthur, J., Akopyan, F., Imam, N., Manohar, R., & Modha, D. S. (2011, September). A digital neurosynaptic core using embedded crossbar memory with 45pJ per spike in 45nm. In Custom Integrated Circuits Conference (CICC), 2011 IEEE (pp. 1-4). IEEE.

[19] Wired. (2016). In Two Moves, AlphaGo and Lee Sedol Redefined the Future. Retrieved from https://www.wired.com/2016/03/two-moves-alphago-lee-sedol-redefined-future/

[20] top500. (2017). Google Latest AlphaGo AI Program Crushes Its Predecessors. Retrieved from https://www.top500.org/news/google-latest-alphago-ai-program-crushes-its-predecessor/

[21] Human Brain Project. (n.d.). Retrieved from https://www.humanbrainproject.eu/en/

[22] The BRAIN Initiative. (n.d.). Retrieved from www.braininitiative.org/

[23] Nvidia. (n.d.). Autonomous Car Development Platform from NVIDIA DRIVE PX2. Retrieved from https://www.nvidia.de/self-driving-cars/drive-px/