International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 08 Issue: 03 | Mar 2021 www.irjet.net p-ISSN: 2395-0072

Accelerating Deep Learning using GPUs

Ranganadh Narayanam[1]

[1]Assistant Professor, IcfaiTech-IFHE Hyderabad

ABSTRACT: Deep Learning theory has been around decades, but due to hardware constraints the solutions have not always been practical. During the time of 1950-2000 researchers proposed many theories and architectures for Deep Learning, but training large neural networks used to take ridiculous amount of time due to limited hardware support of those times and hence training used to be impractical. But it became practical in late 2000s significant success while training neural networks on GPUS. In 2012 Imagenet challenge winner Alexnet model trained parallelly on GPUs which inspired the usage of GPUs in training neural networks which led to the today's revolution. In this article how GPUs acceleration is useful in Deep Learning for speeding up the Neural Networks training is demonstrated. We introduced GoogleColab based work for indentifying the advantage of CUDA of GPUs.

Keywords: Graphics Processing Unit (GPU), Neural Networks, Training, Alexnet, Imagenet, CUDA.

INTRODUCTION

In the past decade in High Performance Computing, and most popularly gaming, GPU is coming into picture more frequently. GPUs have improved year after year and they are capable of doing incredibly great works. They are gaining more and more importance in the last few years due to Deep Learning.

In training neural networks they take large amount of time even powerful CPUs weren't efficient enough to handle so many computations at a given time. Due to parallelism [4,5] of GPUs, training is the area where they outperformed CPUs.

Graphics Processing Unit (GPU) is a mini version of an entire computer but only dedicated to a specific task and carries out multiple tasks at the same time. GPU has its own processor embedded into its own mother board. It is coupled with video ram and also a proper thermal design for ventilation and cooling.

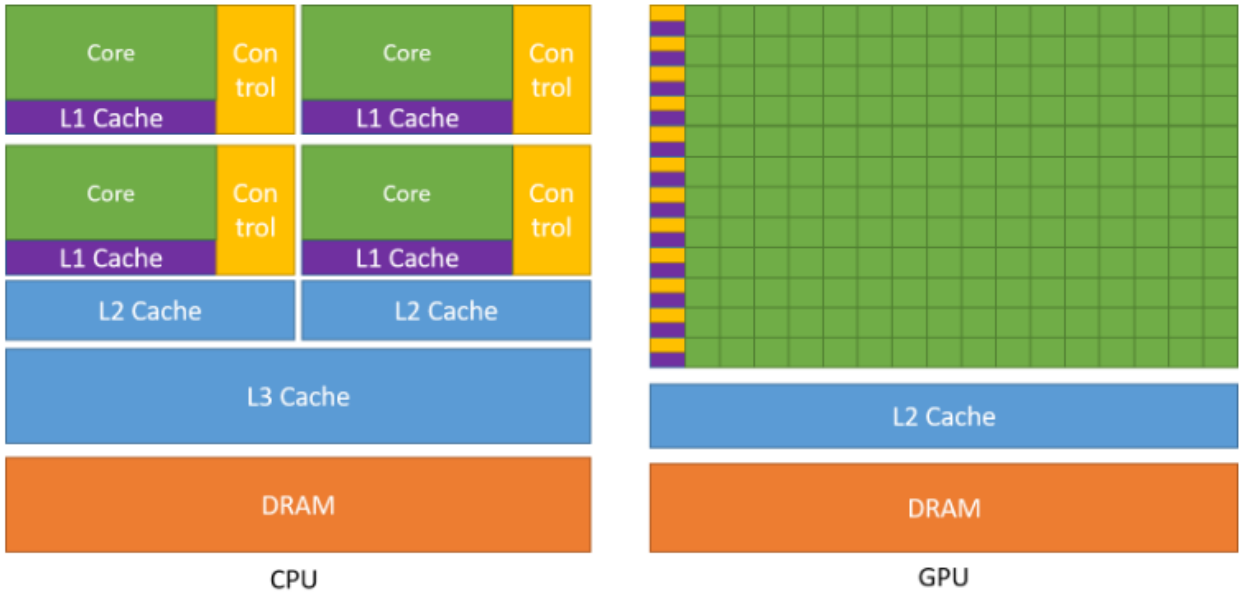
In the term 'Graphics Processing Unit', 'Graphics' refers to rendering an image at specified coordinates on a 2d or 3d space. A viewport or viewpoint is a viewer's perspective of looking to an object depending upon the type of projection used. Rasterisation and Ray-tracing are some of the ways of rendering 3d scenes; both of these concepts are based on a type of a projection called as perspective projection.

Perspective Projection: it is the way in which how an image is formed on a view plane or canvas where the parallel lines converge to a converging point called as 'center of projection' also as the object moves away from the viewpoint it appears to be smaller, exactly how our eyes portray in real-world and this helps in understanding depth in an image as well, that is the reason why it produces realistic images.

Architectural differences between CPU and GPU

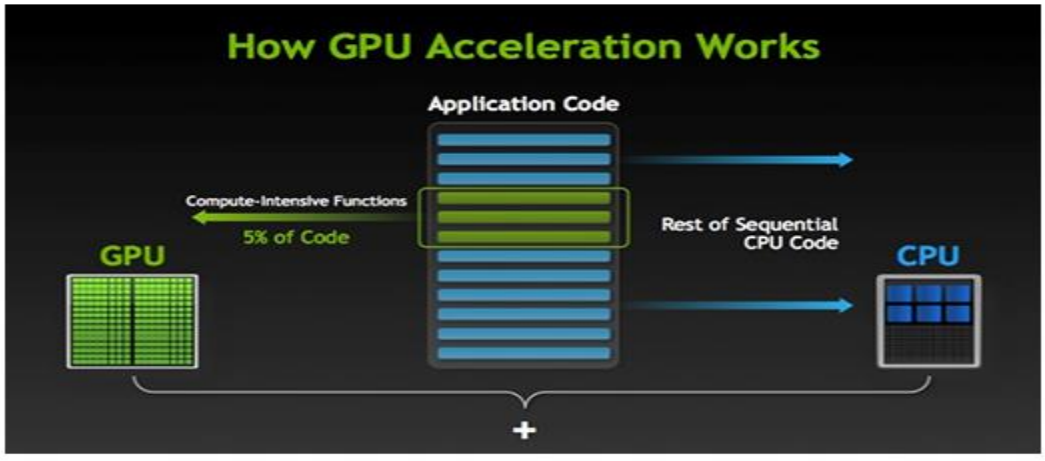
In a computer, for all the computations taking place, Central Processing Unit has been the leading power house. For the computer graphics industry GPU is the special hardware to boost the computing-intensive graphics rendering process. Pioneer of GPUs NVIDIA launched the first GPU GeoForce 256 in 1999.

Architecturally the main difference between the CPU and GPU [2,3] is that, for carrying out arithmetic operations a CPU has limited cores but GPU has hundreds and thousands of cores. A standard high performing CPU has 8-16 cores, where as NVIDIA GPU [5,6,7,8,9] GeForce GTX TITAN Z has 5760 cores. In terms of memory bandwidth GPU also has high memory bandwidth which allows it to move massive data between the Memories.



To take on multiple tasks at the same time, CPU divided into multiple cores. GPUs contain hundreds of thousands of cores all of which are dedicated towards a single task. These are simple computations that are performed more frequently and are independent of each other [1]. And both store frequently required data into their respective cache memory, thereby following the principle of '**locality reference**'[6].

There are many software and games that can take advantage of GPUs for execution. The idea behind this is to make some parts of the task or application code parallel but not the entire processes. This is because most of the task's processes have to be executed in a sequential manner only. For example, logging into a system or application does not need to make parallel.

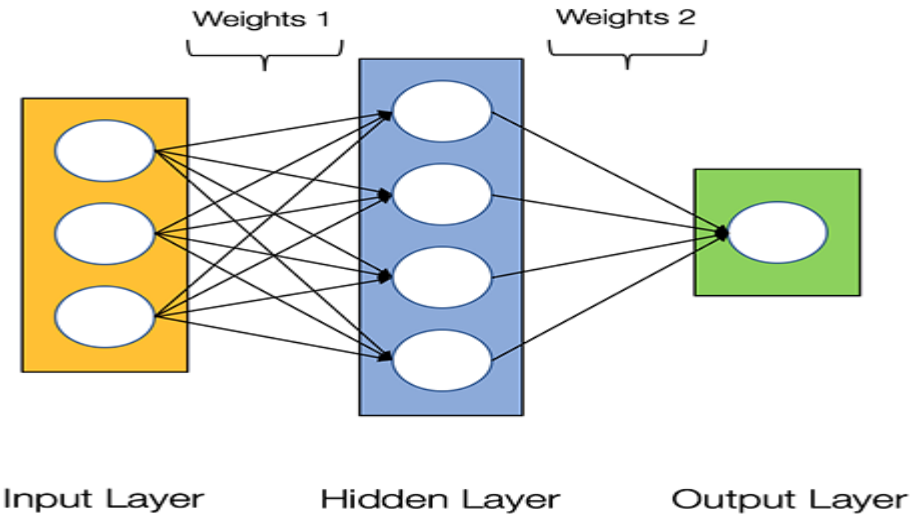


source(NVIDIA)

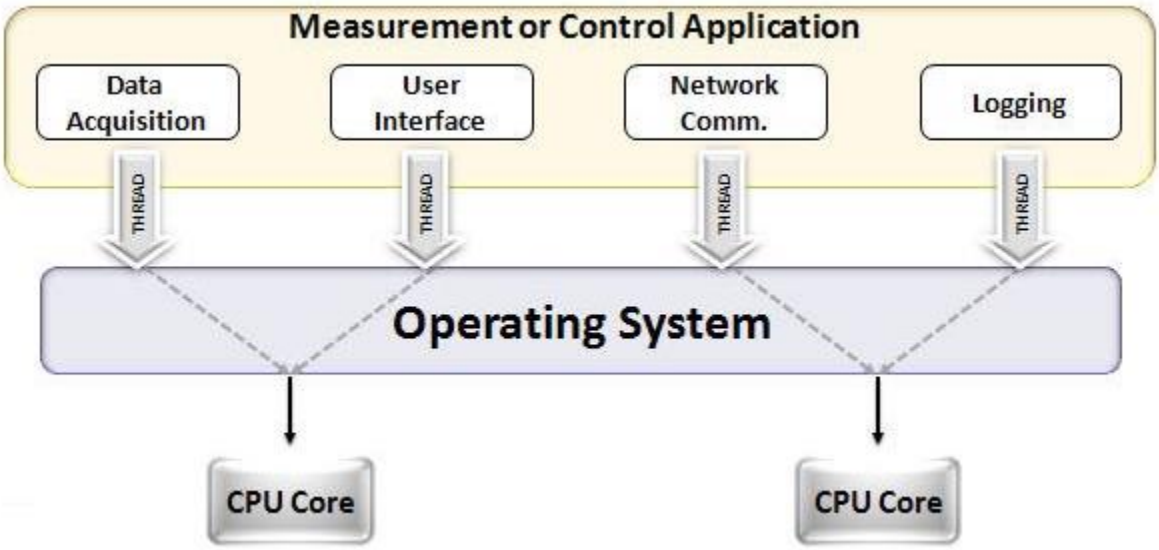
When there is part of execution that can be done in parallel it is simply shifted to GPU for processing where at the same time-sequential task gets executed in CPU, then both of the parts of the task are again combined together.

In the GPU market, there are two main players i.e AMD [1,2,3] and Nvidia. Nvidia [5, 9] GPUs are widely used for deep learning because they have extensive support in the forum software, drivers, CUDA, and cuDNN. So in terms of AI and deep learning, Nvidia is the pioneer for a long time.

Neural networks are said to be **massively parallel**, which means computations [9] in neural networks can be executed in parallel easily and they are independent of each other. Some computations like calculation of weights and activation functions of each layer, back-propagation can be carried out in parallel. There are many research papers available on it as well. Nvidia GPUs come with specialized cores known as **CUDA** cores which help for accelerating deep learning.



One of the most important characteristics of a GPU is the ability to compute processes in parallel. This is the point where the concept of **parallel computing** [6] comes into picture. A CPU in general completes its task in a sequential manner. A CPU can be divided into cores and each core takes up one task at a time. Suppose if a CPU has 2 cores. Then two different task's processes can run on these two cores thereby achieving multitasking. But still, these processes execute in a serial fashion.

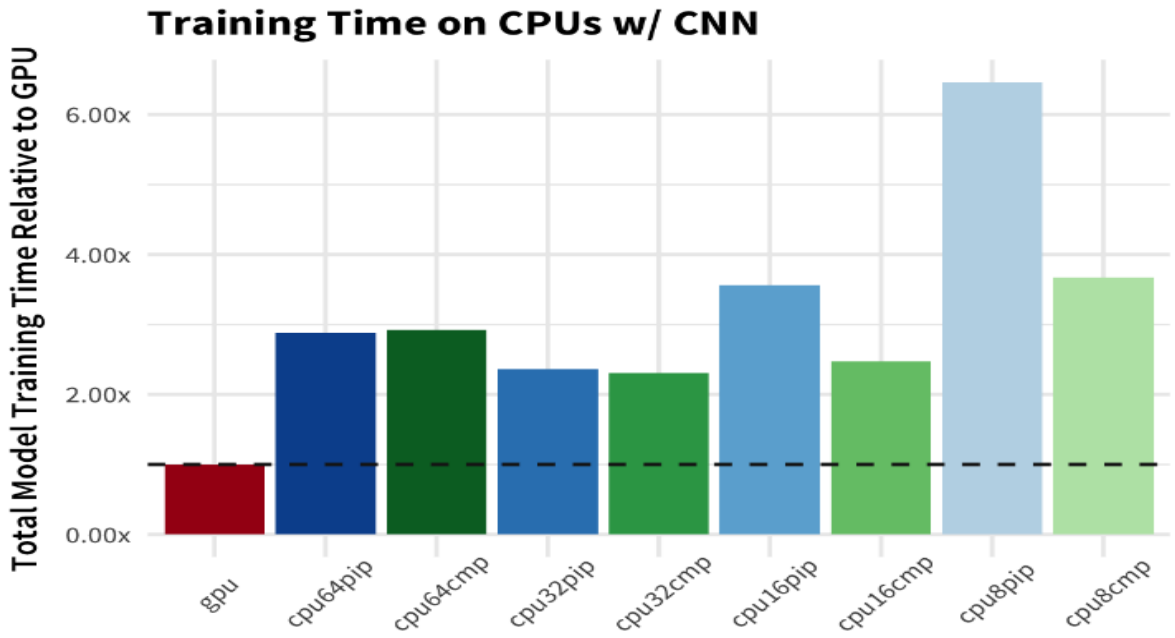


CPUs are really good at handling different tasks related to different operations like handling operating systems, handing spreadsheets, playing HD videos, extracting large zip files, all at the same time. These are some things that a GPU [6] simply cannot do.

As GPU has a high number of cores and a large memory bandwidth [6,7], it can be used to perform high-speed parallel processing on any task that can be broken down for parallel computing. GPUs are incredibly ideal for high parallel tasks that

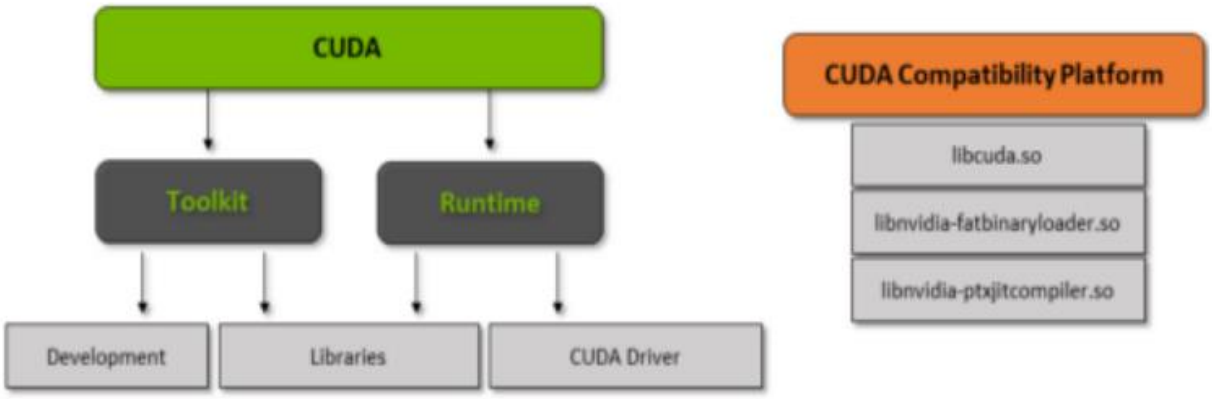
require no effort to break them down for parallel computation. The mathematical matrices operations of the neural network also fall into the high parallel category. This means GPU can effortlessly break down the matrices operation of an extensive neural network, load a huge chunk of matrices data into memory due to the high memory bandwidth, and do fast parallel processing with its thousands of cores.

In a benchmark experiment by running the CNN benchmark [7,8]on the MNIST dataset on GPU and various CPUs on the Google Cloud Platform. The results clearly show that CPUs are struggling with training time, whereas the GPU is blazingly fast.



GPU vs CPU performance benchmark

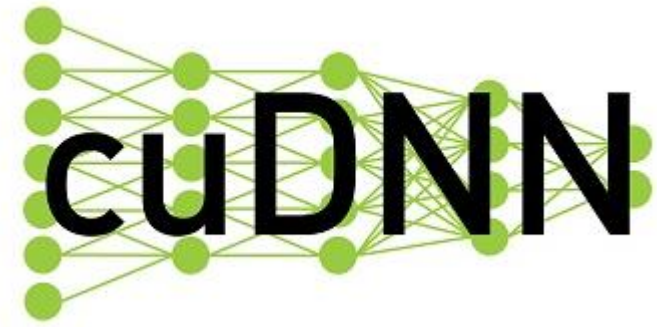
CUDA: Compute Unified Devise Architecture



source(NVIDIA)

CUDA stands for 'Compute Unified Device Architecture' which was launched in the year 2007, it's a way in which we can achieve parallel computing and yield most out of GPU power in an optimized way, which results in much better performance while executing tasks. The CUDA toolkit is a complete package that consists of a development environment that is used to build applications that make use of GPUs [8,9]. This toolkit mainly contains c/c++ compiler, debugger, and libraries. Also, the CUDA runtime has its drivers so that it can communicate with the GPU. CUDA is also a programming language that is specifically made for instructing the GPU for performing a task. It is also known as GPU programming.

What is cuDNN?



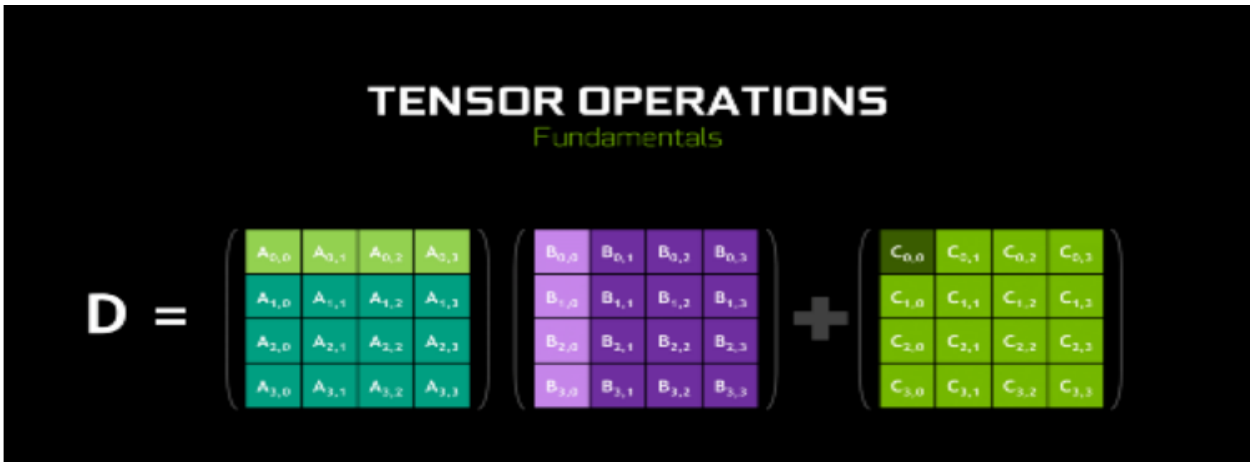
cuDNN is a neural network library that is GPU optimized and can take full advantage of Nvidia GPU. This library consists of the implementation of convolution, forward and backward propagation, activation functions, and pooling [10, 9]. It is a must library without which you cannot use GPU for training neural networks.

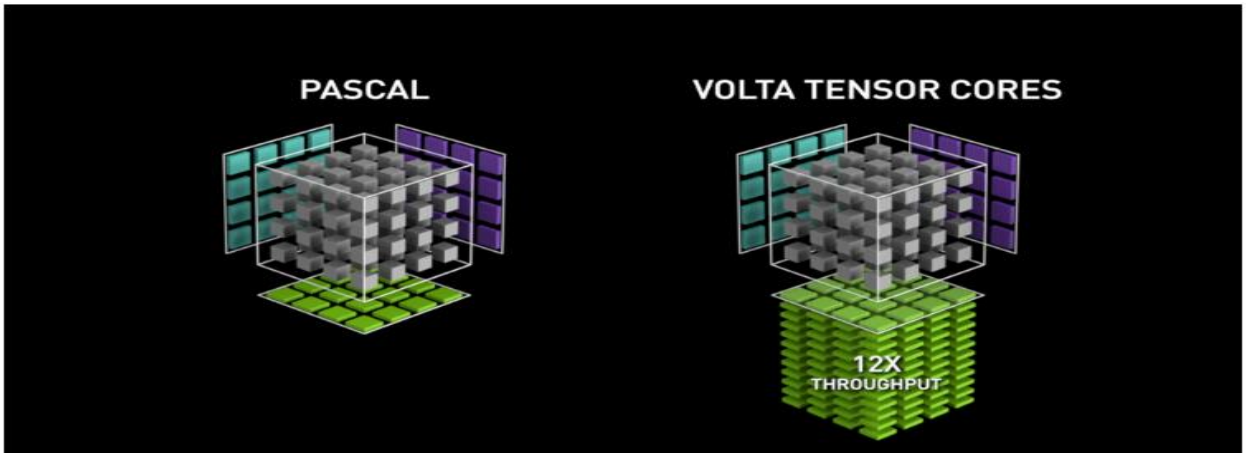
NVIDIA for Deep Learning: CUDA and CuDNN

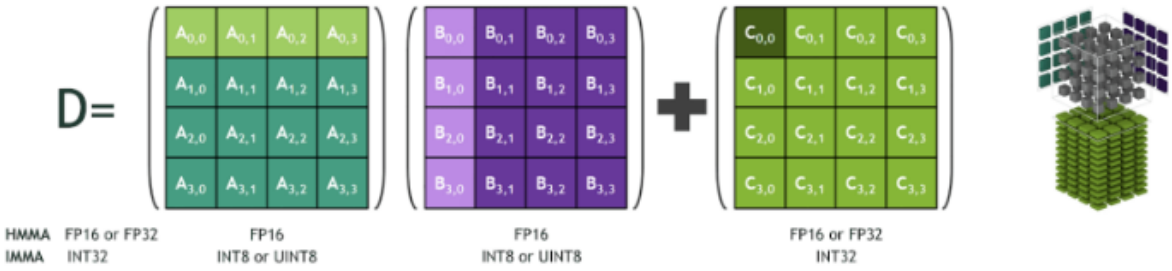
Currently, Deep Learning models can be accelerated only on NVIDIA GPUs, and this is possible with its API called CUDA for doing *general-purpose GPU programming* (GPGPU). CUDA which stands for Compute Unified Device Architecture, was initially released in 2007, and the Deep Learning community soon picked it up. Seeing the growing popularity of their GPUs with Deep Learning, NVIDIA released CuDNN in 2014, which was a wrapper library built on CUDA for Deep Learning functions like back propagation, convolution, pooling, etc. This made life easier for people leveraging the GPU for Deep Learning without going through the low-level complexities of CUDA. Soon all the popular Deep Learning libraries like PyTorch [10], Tensorflow, Matlab, and MXNet[14,12] started incorporating CuDNN directly in their framework to give a seamless experience to its users. Hence using a GPU for deep learning has become very simple compared to earlier days.

Deep Learning Libraries supporting CUDA

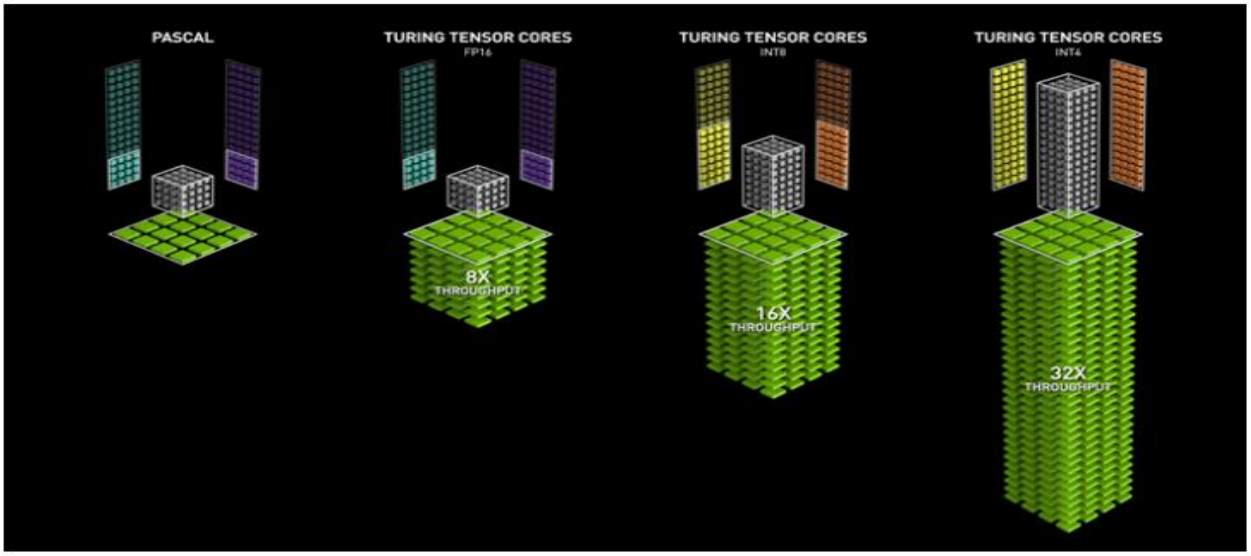
NVIDIA Tensor Cores







Matrice Operation supported by Tensor Core



Turing Tensor Core Performance

By releasing CuDNN, NVIDIA positioned itself as an innovator in the Deep Learning revolution, but that was not all. In 2017, NVIDIA [5, 6, 7, 8, 9] launched a GPU called Tesla V100, which had a new type of Voltas architecture built with dedicated Tensor Core to carry out tensor operations of the neural network. NVIDIA claimed that it was 12 times faster [11] than its traditional GPUs built on CUDA core. This performance gain was possible because its Tensor Core was optimized to carry out a specific matrice operation of multiplying two 4×4 FP16 matrices and add another 4×4 FP16 or FP32 matrices. Such operations are quite common in neural networks; hence an optimized Tensor Core for this operation could boost the performance significantly. NVIDIA added support for FP32, INT4, and INT8 precision in the 2nd generation Turing Tensor Core architecture. Recently, NVIDIA released the 3rd generation A100 Tensor Core GPU based on Ampere architecture with support for FP64 and a new precision Tensor Float 32 which is similar to FP32 and can deliver 20 times more speed without code change. As stated by Nvidia, the new generation tensor cores based on Volta architecture is much faster than CUDA cores based on Pascal architecture. This gave a huge boost to deep learning.

How NVIDIA over performs AMD:

For gaming AMD GPUs are perfect, but when it comes to deep learning then NVIDIA is way ahead. It does not mean that AMD GPUs are bad. In AMD GPUs the software optimization and drivers which are not being updated actively. But NVIDIA [5, 6, 7, 8, 9] has better drivers with frequent updates. But the top most important cause [15] for NVIDIA's efficiency is CUDA, CuDNN which accelerate the computation.

Very important libraries like **Tensorflow**, **PyTorch** support for **CUDA**. It means entry level GPUs of the GTX 1000 series can be used. In AMD very little software support for their **GPUs**. On the hardware side, NVIDIA has dedicated tensor cores. **ROCm** are there for **AMD** for acceleration not as good as tensor cores, and many deep learning libraries do not support **ROCm**. In AMD For the past few years, no big leap was noticed in terms of performance.

Due to these reasons NVIDIA simply excels in Deep Learning.

Hands on with CUDA and PyTorch:

We discuss about using CUDA from PyTorch. This example carries out multiple operations both on CPU and GPU and compares the speed.

First, import the required numpy and PyTorch libraries.

```
[ ]  import numpy as np
     import torch
```

multiply the two 10000 x 10000 matrices with CPU using numpy. It took **1min 48s.**

```
%%time
b = np.random.randn(10000,10000)
result = np.matmul(b,b)
del b, result
```

```
CPU times: user 1min 48s, sys: 686 ms, total: 1min 48s
Wall time: 57.2 s
```

Next, carry out the same operation using torch on CPU, and this time it took only **26.5 seconds**

```
%%time
y = torch.randn(10000,10000)
result = torch.matmul(y,y)
del y, result
```

```
CPU times: user 26.5 s, sys: 74.2 ms, total: 26.5 s
Wall time: 26.5 s
```

Finally, carry this operation using torch on CUDA, and it amazingly takes just **10.6 seconds**

```
%%time
x = torch.randn(10000,10000).cuda()
result = torch.matmul(x,x)
del x, result
```

```
CPU times: user 3.78 s, sys: 960 ms, total: 4.74 s
Wall time: 10.6 s
```

To summarize, the GPU was around 2.5 times faster than the CPU with PyTorch.

The above program utilizes the GPU in GoogleColab. There are certain instructions of how to setup GPU mode in GoogleColab in [3].

EPILOGUE

After this survey, in this paper we conclude that NVIDIA is the market leader in GPU, and AMD may make some remarkable improvements in the future of their GPUs as AMD also making a great job with respect to their CPUs: The Ryzen series. In upcoming years the scope of GPUs is huge as we make new innovations and breakthroughs in deep learning, machine learning, and HPC. For many developers and students to get into this field GPU acceleration will always come handy, as their prices also becoming more affordable. The wide community that contributes to the development of AI and HPC is also one important cause for affordability. If your needs are to train state of the art models for regular use or research, you can go for high-end

GPUs like RTX 8000, RTX 6000, Titan RTX. In fact, some of the projects may also require you to set up a cluster of GPUs which will require a fair amount of funds. If you intend to use GPUs for competitions or hobbies and have the money, you can purchase medium to low-end GPUs like RTX 2080, RTX 2070, RTX 2060 GTX 1080. You also have to consider the RAM required for your work since the GPUs come with different RAM sizes and are priced accordingly. If you don't have the money and would still like to experience GPU, your best option is to use a GoogleColab that gives free but limited GPU support to its users.

REFERENCES

[1] M.MadiajaganMS,S. SridharRaj(2019),"Parallel Computing, Graphics Processing Unit (GPU) and New Hardware for Deep Learning in Computational Intelligence Research",Academic Press, Deep Learning and parallel computing environment for bioengineering systems, (pp 1-15)
[2] John D. Owens; Mike Houston; David Luebke; Simon Green; John E. Stone; James C. Phillips **(2008),"** GPU Computing**",** Proceedings of the IEEE**,** 10.1109/JPROC.2008.917757 **(pp 20-29)**

[3] Jayshree Ghorpade , Jitendra Parande, Madhura Kulkarni , Amit Bawaskar (2012), "GPGPU PROCESSING IN CUDA ARCHITECTURE", Advanced Computing: An International Journal (ACIJ), Vol.3, No.1, January 2012. (pp 31-39)
[4] Danilo De Donno et al.(2010), "Introduction to GPU Computing and CUDA Programming: A Case Study on FDTD," IEEE Antennas and Propagation Magazine, June 2010 (pp 28-36)
[5] Miguel C´ardenas-Montes(2011)," Accelerating Particle Swarm Algorithm with GPGPU," 19th International Euromicro Conference on Parallel, Distributed and Network-Based Processing, 2011 (pp 50-62)
[6] K. Hagiwara , J. Kanzaki, a , N. Okamura , b , D. Rainwater, and T. Stelzer , c(2009), "Calculation of HELAS amplitudes for QCD processes using graphics processing unit (GPU)", EPJ. (pp 1-10)
[7] Charles Bibby and Ian Reid (2010),"Fast Feature Detection with a Graphics Processing Unit Implementation", robotics, activie visio journal, (pp 400-410)
[8] E. Rosten and T. Drummond (2006) "Machine learning for very high speed corner detection (to appear)",. In Proc. 9th European Conf. on Computer Vision, Graz, Austria, 2006. (pp 90-100)
[9] E. Rosten and T. Drummond (2005), "Fusing points and lines for high performance tracking", In Proc. 10th Int'l Conf. on Computer Vision, Beijing, volume 2, (pp 1508– 1515), 2005.
[10] K. Mikolajczyk(2002)," Detection of local features invariant to affines transformations", PhD thesis, INPG, Grenoble, 2002.
[11] Christopher Edward Davis (2005), "Graphics Processing Unit Computation of Neural Networks", Neural Networks, 2005. IJCNN '05. Proceedings. 2005 IEEE International Joint Conference onVolume: 1, 10.1109/IJCNN.2005.1555903
[12] Kyoung-Su Oh, Keechul Jung (2004), "GPU implementation of neural networks, Pattern Recognition", 37, 6, (pp 1311-1314).
[13] Bohn, C.A. Kohonen (1998), "Feature Mapping Through Graphics Hardware", In Proceedings of 3rd Int. Conference on Computational Intelligence and Neurosciences, (pp 12-19).
[14] Femando R. and Killgard, M.J. The Cg Tutorial (2009) "The Definitive Guide to Programmable Real-Time Graphics", Addison-Wisley, (pp 200-210)
[15] Mark, W.R., Glanville, R. S., Akeley, K. and Killgard, M.J. (2003). Cg : "A system for programming graphics hardware in a C-like language". ACM Trans. Graph. 22, 3, (pp 896-907)
[16] https://www.amd.com/en
[17] https://www.nvidia.com/en-in/
[18] https://medium.com/deep-learning-turkey/google-colab-free-gpu-tutorial-e113627b9f5d
[19] https://www.analyticsvidhya.com/blog/2014/06/deep-learning-attention/
[20] https://developer.nvidia.com/blog/nvidia-ampere-architecture-in-depth/
[21] https://developer.nvidia.com/blog/nvidia-turing-architecture-in-depth/
[22] https://www.nvidia.com/en-us/data-center/tensor-cores/
[23]https://developer.nvidia.com/blog/inside-volta/?ncid=afm-chs-4270&ranMID=44270&ranEAID=kXQk6*ivFEQ&ranSiteID=kXQk6.ivFEQ-BU51IDhA70A9R4DAvPF8BA
[24] https://en.wikipedia.org/wiki/General-purpose_computing_on_graphics_processing_units
[25] https://www.infoq.com/news/2014/09/cudnn/
[26] https://minimaxir.com/2017/07/cpu-or-gpu/