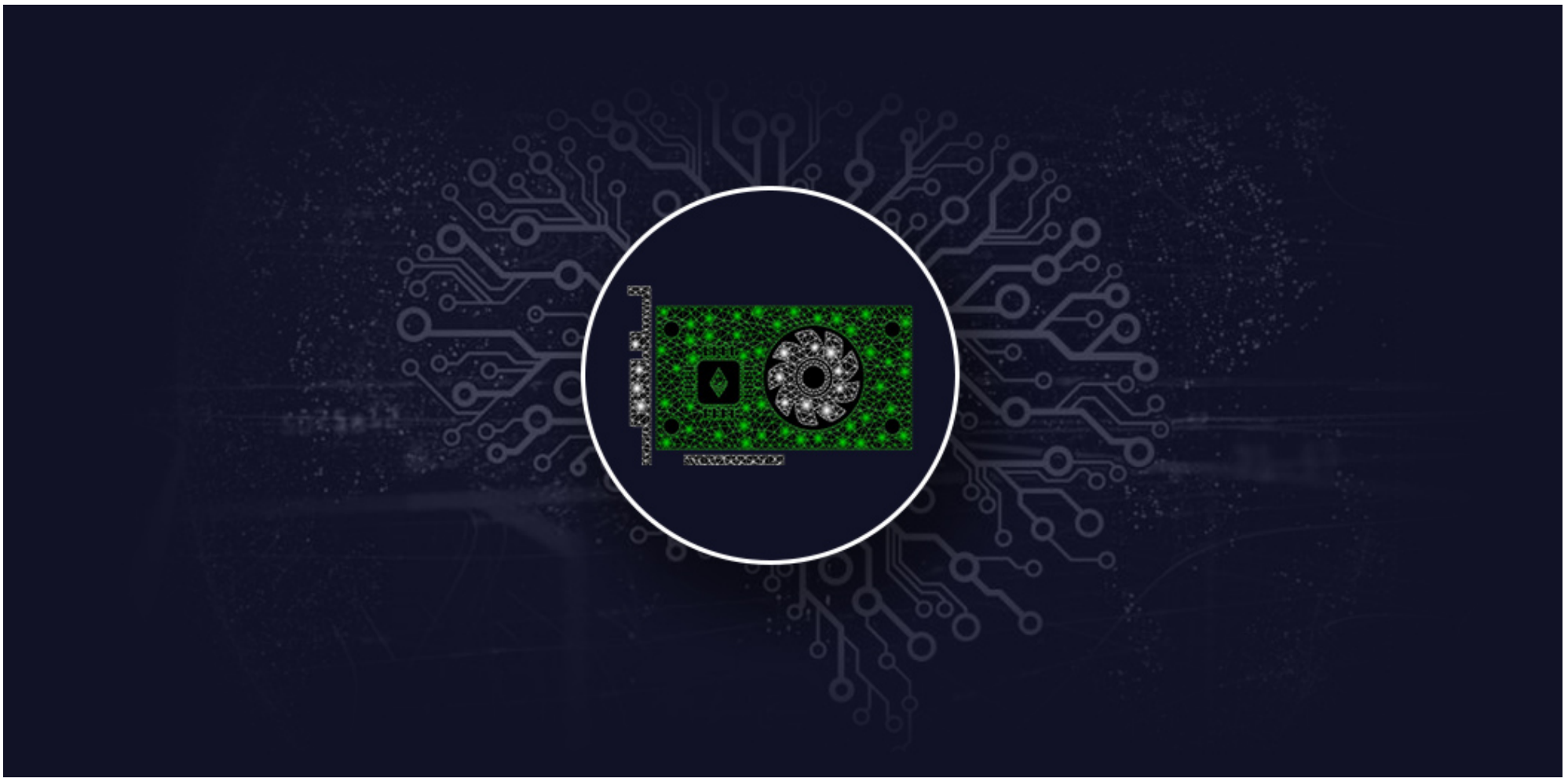
Why GPUs are more suited for Deep Learning?

guest_blog — Published On September 9, 2020 and Last Modified On September 9th, 2020

Deep Learning Intermediate

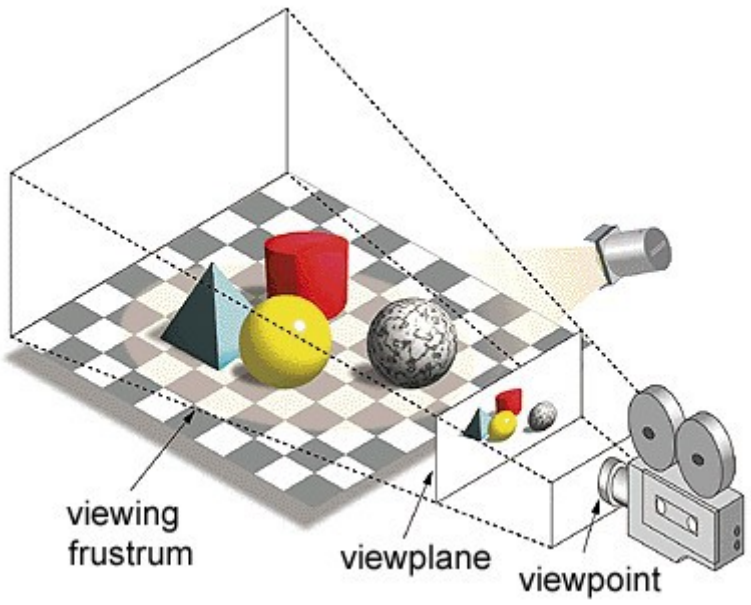
Since the past decade, we have seen GPU coming into the picture more frequently in fields like HPC(High-Performance Computing) and the most popular field i.e gaming. GPUs have improved year after year and now they are capable of doing some incredibly great stuff, but in the past few years, they are catching even more attention due to deep learning.



As deep learning models spend a large amount of time in training, even powerful CPUs weren't efficient enough to handle soo many computations at a given time and this is the area where GPUs simply outperformed CPUs due to its **parallelism**. But before diving into the depth lets first understand some things about GPU.

What is the GPU?

A GPU or 'Graphics Processing Unit' is a mini version of an entire computer but only dedicated to a specific task. It is unlike a CPU that carries out multiple tasks at the same time. GPU comes with its own processor which is embedded onto its own motherboard coupled with v-ram or video ram, and also a proper thermal design for ventilation and cooling.



source(Gamers Nexus)

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. What is perspective projection?

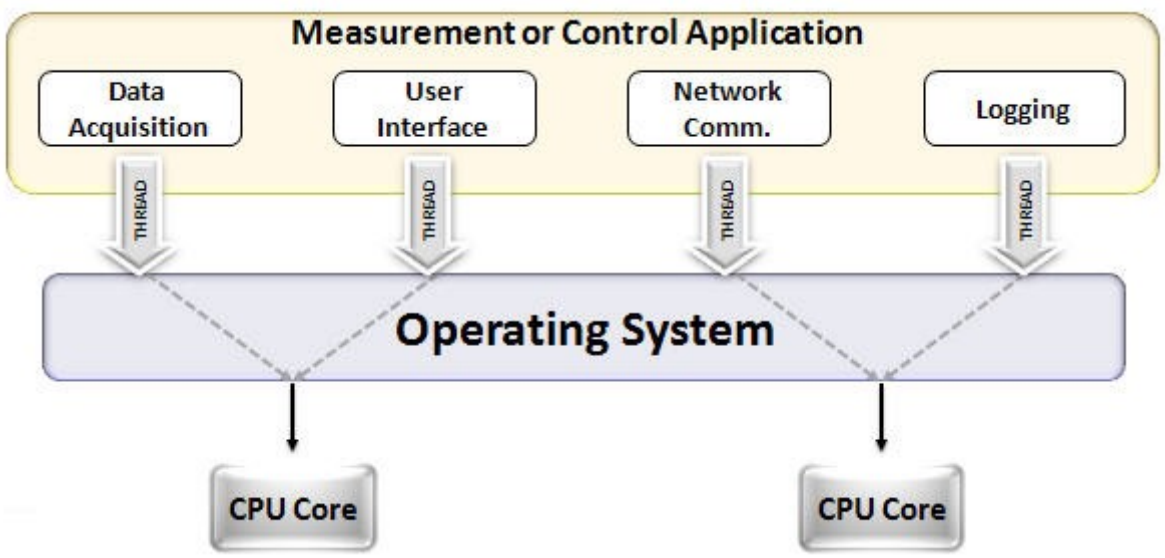
In short, it is the way in which how an image is formed on a view plane or canvas where the parallel lines converge to a converging point called as 'center of projection' also as the object moves away from the viewpoint it appears to be smaller, exactly how our eyes portray in real-world and this helps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

Moreover GPUs also process complex geometry, vectors, light sources or illuminations, textures, shapes, etc. As now we have a basic idea about GPU, let us understand why it is heavily used for deep learning.

Why GPUs are better for deep learning?

One of the most admired characteristics of a GPU is the ability to compute processes in parallel. This is the point where the concept of **parallel computing** kicks in. 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.

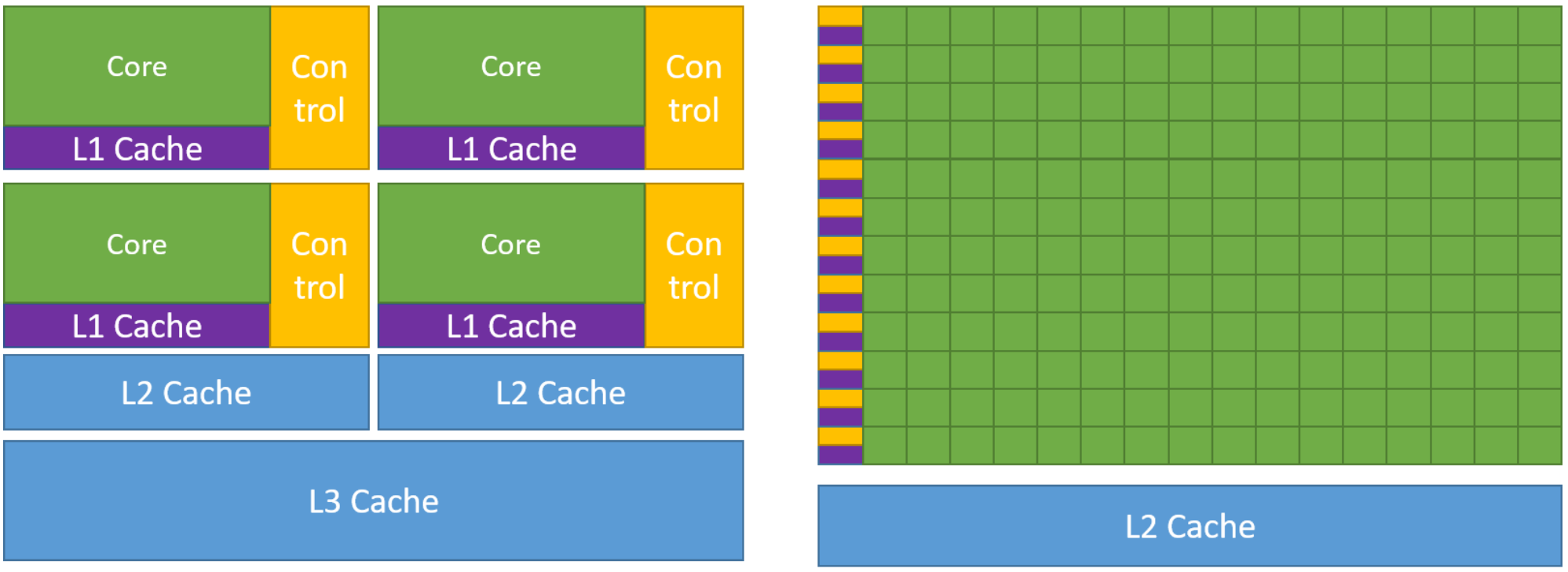
But still, these processes execute in a serial fashion.



source(Sample Examples)

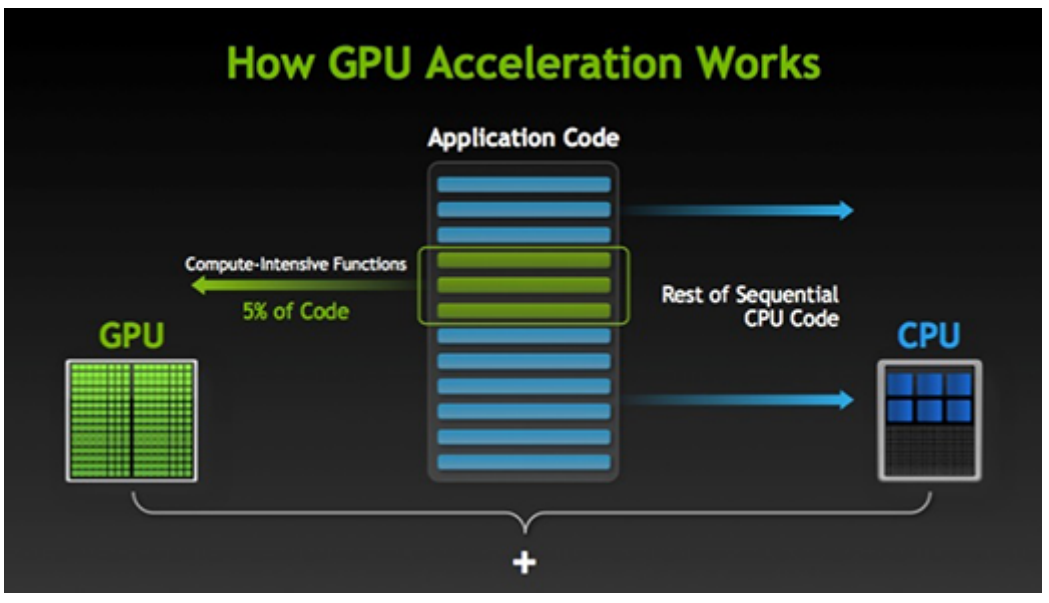
This doesn't mean that CPUs aren't good enough. In fact, CPUs are really good at handling different tasks related to different operations like handling operating systems, handing spreadsheets, playing HD videos, extracting large zip files, all at the same time. These are some things that a GPU simply cannot do.

Where the difference lies?



source(NVIDIA)

As discussed previously a CPU is divided into multiple cores so that they can take on multiple tasks at the same time, whereas GPU will be having hundreds and 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. And both store frequently required data into there respective cache memory, thereby following the principle of '**locality reference**'.



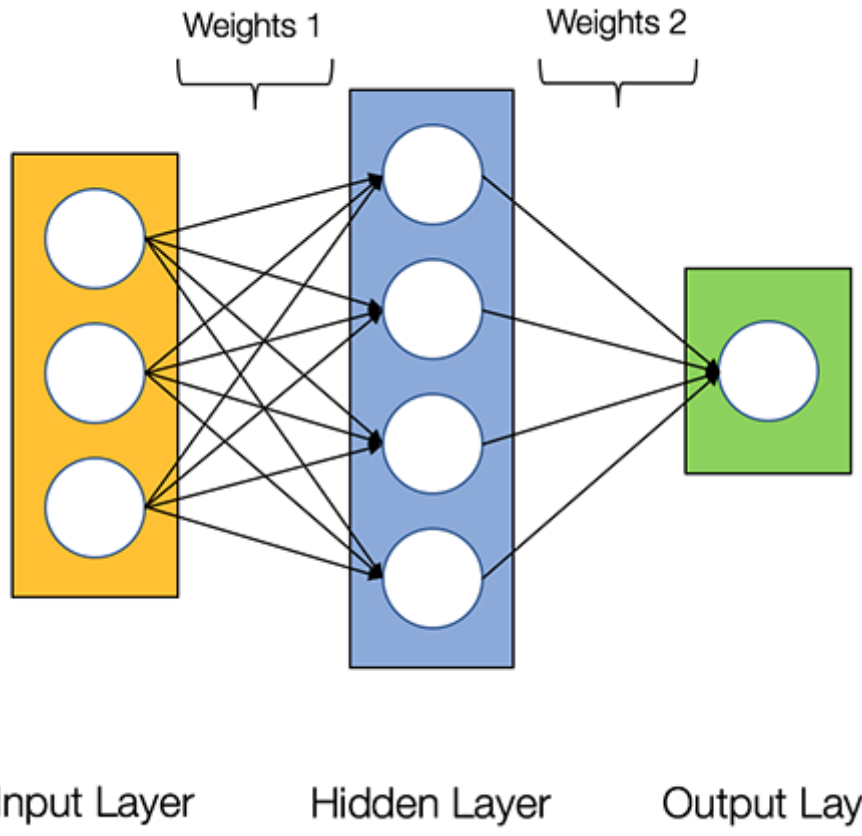
source(NVIDIA)

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.

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 and Nvidia. Nvidia GPUs are widely used for deep learning because they

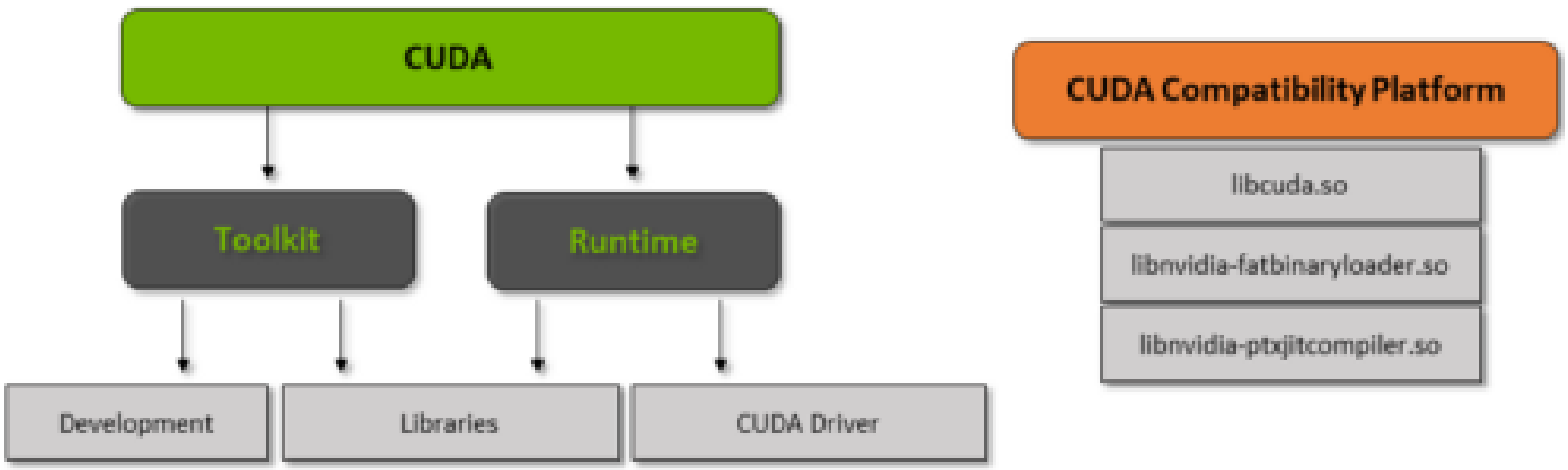
Neural networks are said to be **embarrassingly parallel**, which means computations in neural networks can be executed in parallel easily and they are independent of each other.



Some computations like calculation of weights and activation functions of each layer, backpropagation 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 helps for accelerating deep learning.

What is CUDA?

CUDA stands for 'Compute Unified Device Architecture' which was launched in the year 2007, it's a way in which you can achieve parallel computing and yield most out of your GPU power in an optimized way, which results in much better performance while executing tasks.



source(NVIDIA)

The CUDA toolkit is a complete package that consists of a development environment that is used to build applications that make use of GPUs. 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.
Below is a simple hello world program just to get an idea of how CUDA code looks like.

```
/* hello world program in cuda *\
#include<stdio.h>
#include<stdlib.h>
#include<cuda.h>__global__ void demo() {
    printf("hello world!,my first cuda program");
}int main() {
    printf("From main!\n");
    demo<<<1,1>>>();
    return 0;
}
```

```
From main!
hello world!,my first cuda program
```

output

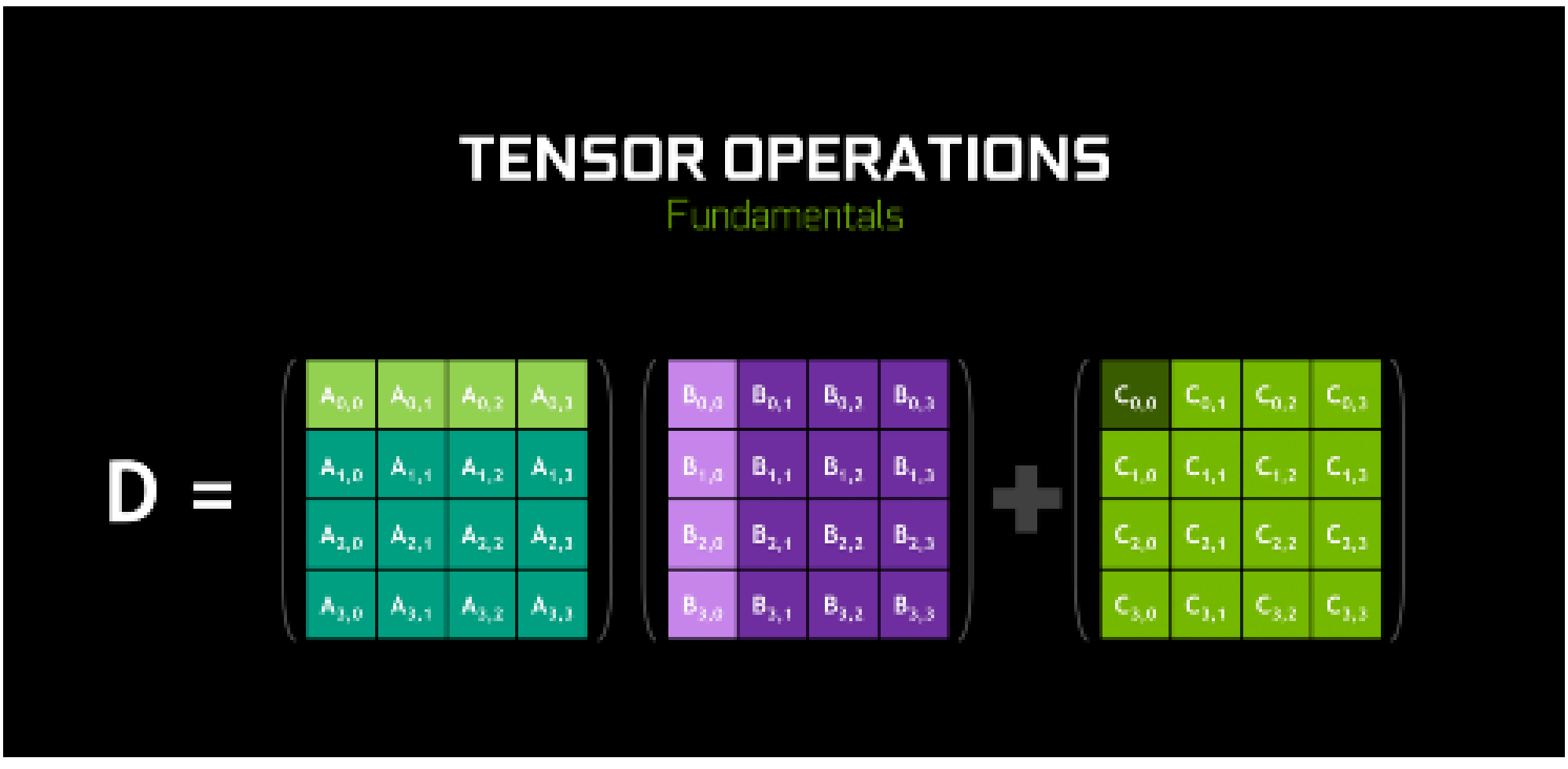
What is cuDNN?

Source(Hyper Linuxer)

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. It is a must library without which you cannot use GPU for training neural networks.

A big leap with Tensor cores!

Back in the year 2018, Nvidia launched a new lineup of their GPUs i.e 2000 series. Also called RTX, these cards come with tensor cores that are dedicated to deep learning and based on Volta architecture.

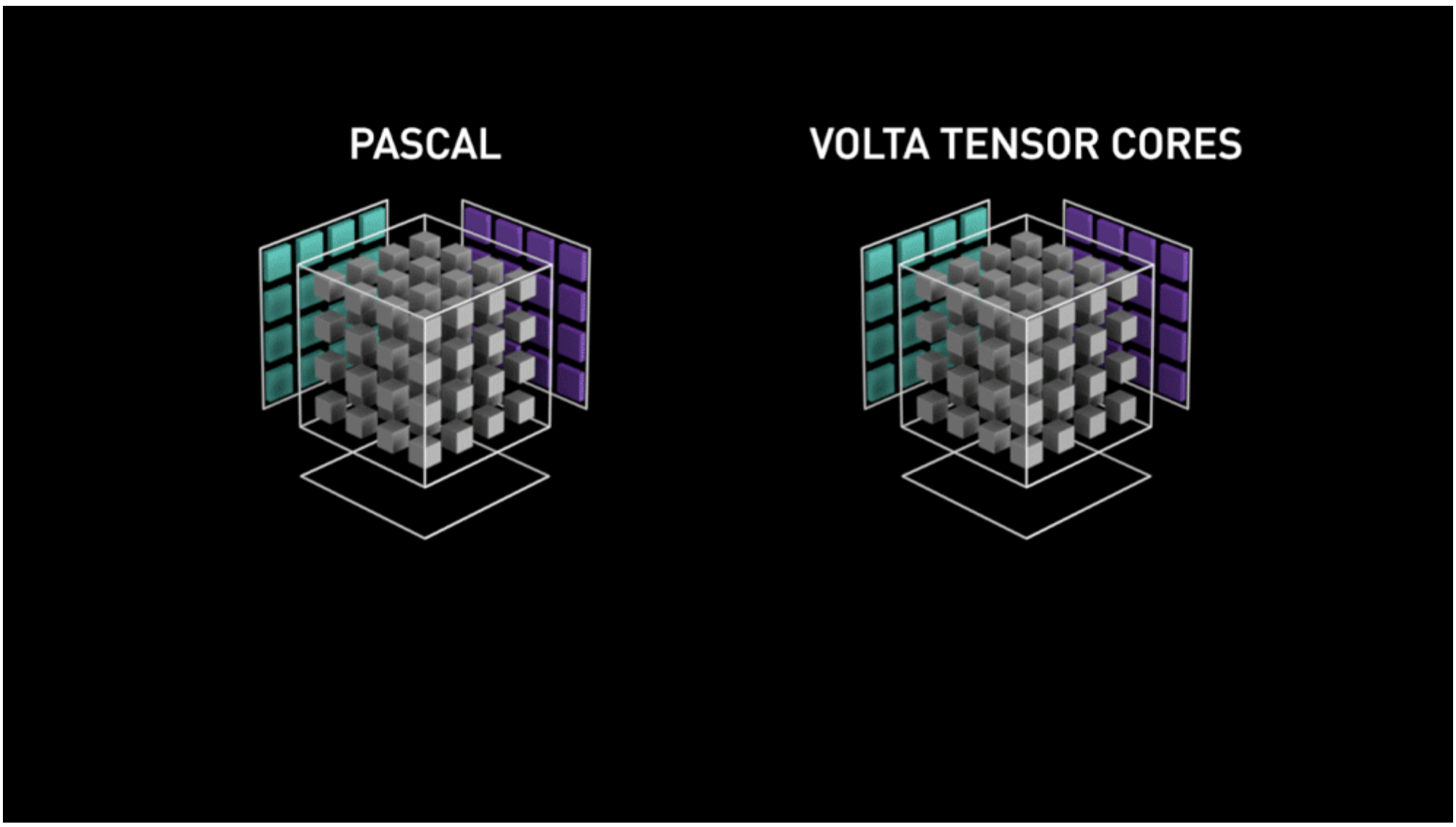


Source(Anand Tech)

Tensor cores are particular cores that perform matrix multiplication of 4 x 4 FP16 matrix and addition with 4 x 4 matrix FP16 or FP32 in half-precision, the output will be resulting in 4 x 4 FP16 or FP32 matrix with full precision.

Note: 'FP' stands for floating-point to understand more about floating-point and precision check this blog.

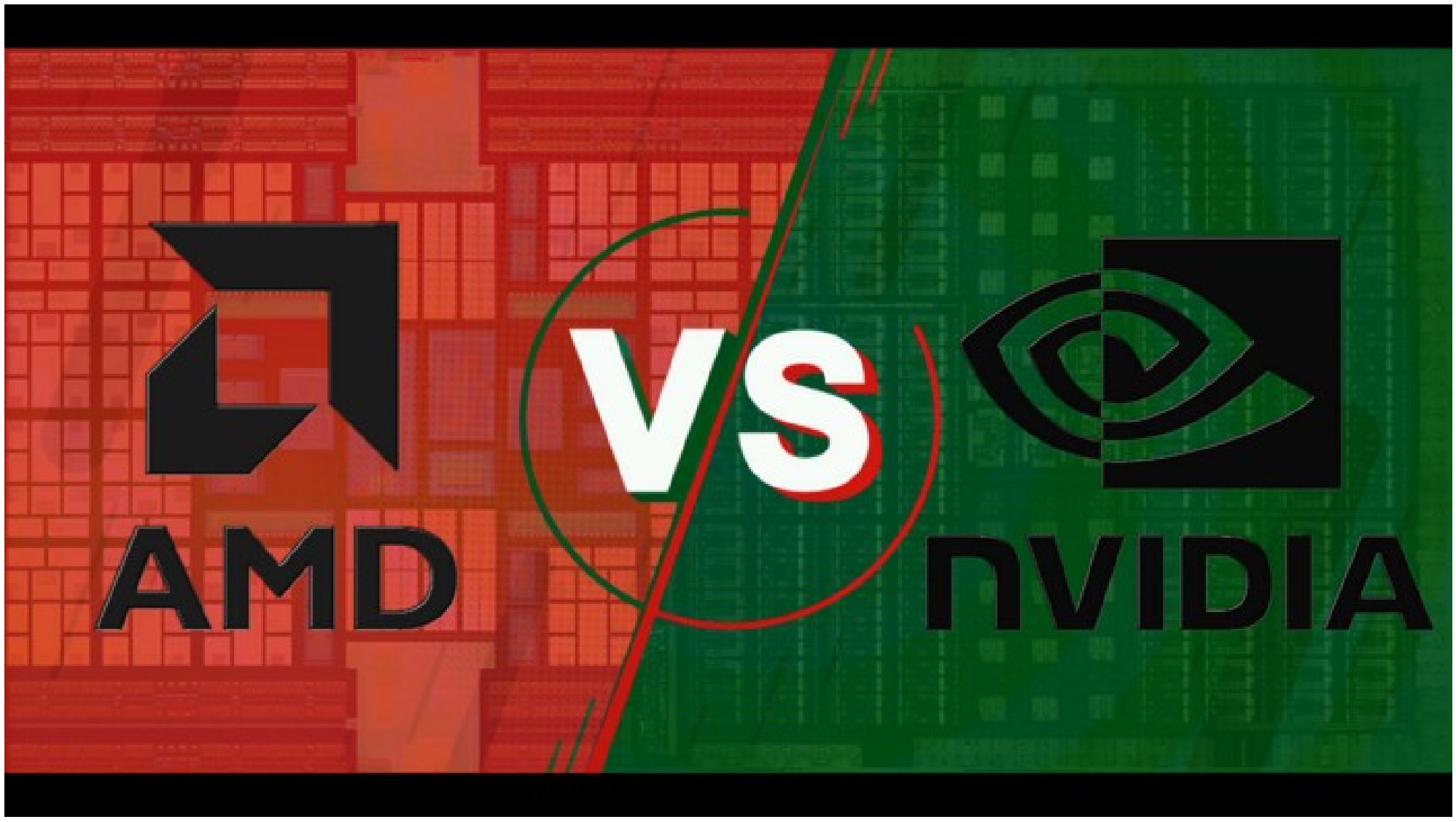
As stated by Nvidia, the new generation tensor cores based on volta architecture is much faster than CUDA cores based on Pascal architecture. This gave a huge boost to deep learning.



source(NVIDIA)

At the time of writing this blog, Nvidia announced the latest 3000 series of their GPU lineup which come with Ampere architecture. In this, they improved the performance of tensor cores by 2x. Also bringing new precision values like TF32(tensor float 32), FP64(floating point 64). The TF32 works the same as FP32 but with speedup up to 20x, as a result of all this Nvidia, claims the inference or training time of models will be reduced from weeks to hours.

AMD vs Nvidia



source(Tom's Hardware)

AMD GPUs are decent for gaming but as soon as deep learning comes into the picture, then simply Nvidia is way ahead. It does not mean that AMD GPUs are bad. It is due to the software optimization and drivers which is not being updated actively, on the Nvidia side they have better drivers with frequent updates and at the top of that CUDA, cuDNN helps to accelerate the computation.

Some well-known libraries like Tensorflow, PyTorch support for CUDA. It means entry-level GPUs of the GTX 1000 series can be used. On the AMD side, it has very little software support for their GPUs. On the hardware side, Nvidia has introduced dedicated tensor cores. AMD has ROCm for acceleration but it is not good as tensor cores, and many deep learning libraries do not support ROCm. For the past few years, no big leap was noticed in terms of performance.

Due to all these points, Nvidia simply excels in deep learning.

Summary

To conclude from all that we have learned it's clear that as of now Nvidia is the market leader in terms of GPU, but I really hope that even AMD catches up in the future or at least make some remarkable improvements in the upcoming lineup of their GPUs as they already doing a great job with respect to their CPUs i.e the Ryzen series.

The scope of GPUs in upcoming years is huge as we make new innovations and breakthroughs in deep learning, machine learning, and HPC. GPU acceleration will always come in handy for many developers and students to get into this field as their prices are also becoming more affordable. Also thanks to the wide community that also contributes to the development of AI and HPC.

About the Author

Prathmesh Patil

ML enthusiast, Data Science, Python developer.

LinkedIn: https://www.linkedin.com/in/prathmesh

AMD cuda GPU for deep learning NVIDIA

About the Author

guest_blog

Our Top Authors

view more

Previous Post

18 All-Time Classic Open Source Computer Vision Projects for Beginners

Next Post

Busted! 11 Data Science Myths You Should Avoid at All Costs

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name*

Email*

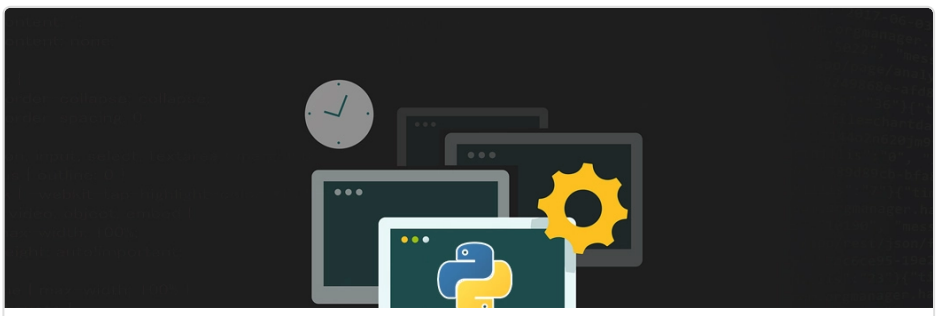
Website

Notify me of follow-up comments by email.

Notify me of new posts by email.

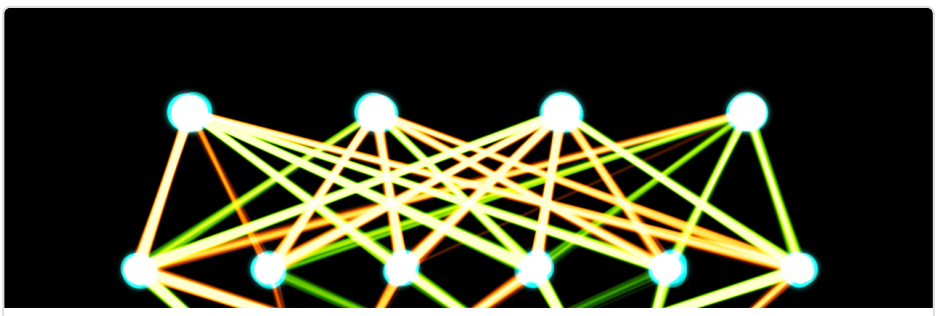
Submit

Top Resources



Python Tutorial: Working with CSV file for Data Science

Harika Bonthu - AUG 21, 2021



Boost Model Accuracy of Imbalanced COVID-19 Mortality Prediction Using GAN-based..

Bala Gangadhar Thilak Adiboina - OCT 07, 2020

Joins in Pandas: Master the Different Types of Joins in..

Abhishek Sharma - FEB 27, 2020

AUC-ROC Curve in Machine Learning Clearly Explained

Aniruddha Bhandari - JUN 16, 2020