ACCELERATION OF DEEP NEURAL NETWORK

Under Supervision:
Dr.Nour El-medany

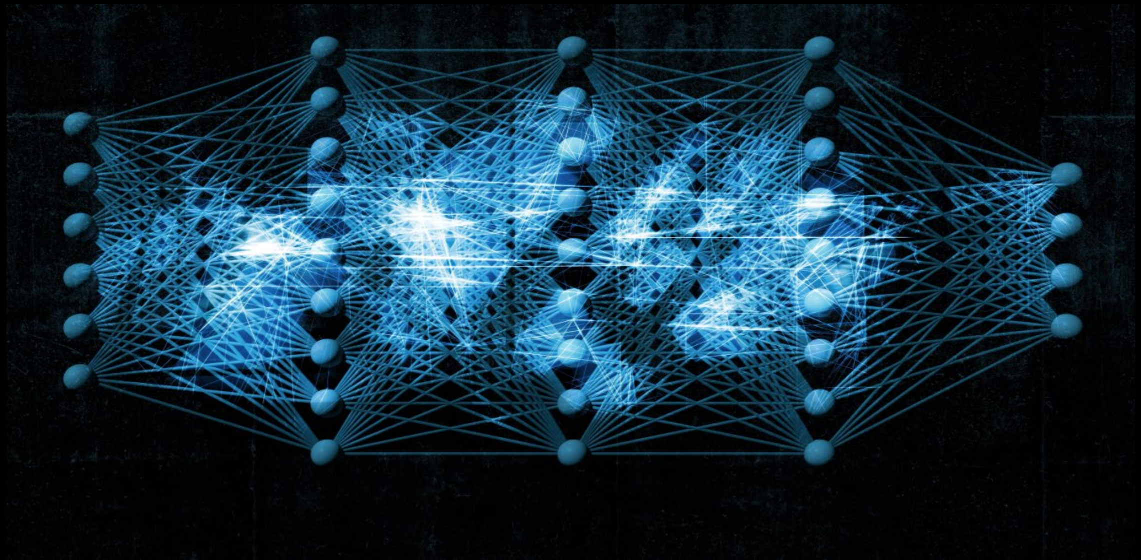
By: Mohamed Ashraf Ragab
And: Aya Ashraf Ragab

OBJECTIVES:

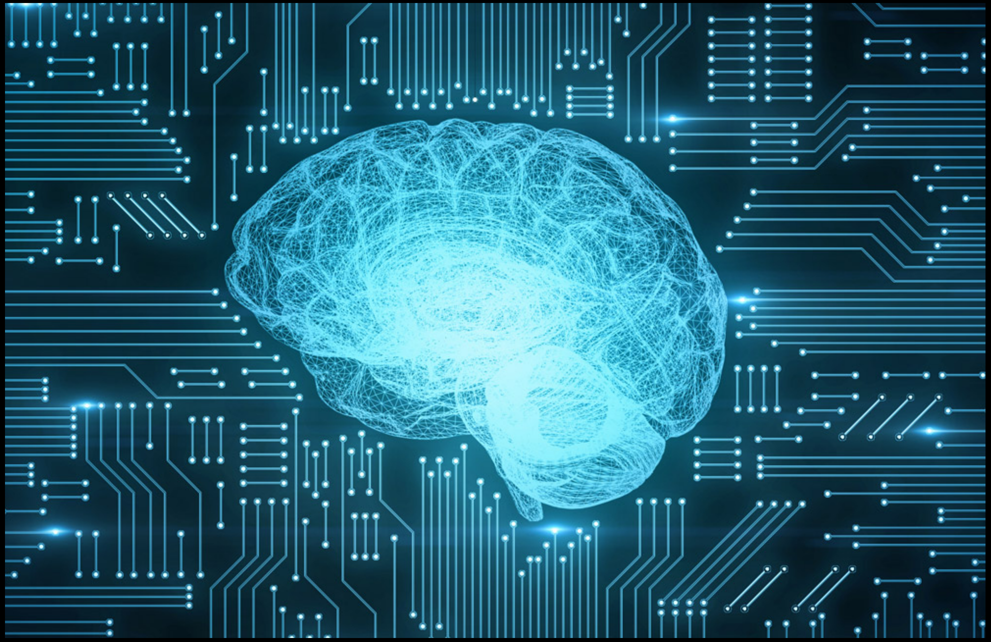
- Define the accelerators on which models can run
- Make comparison between accelerators
- Define Deep Learning Frameworks
- Make comparison between frameworks
- Run each framework on each accelerators
- Show runtime results

INTRODUCTION

- Artificial intelligence and machine learning technologies have been accelerating the advancement of intelligent applications. To cope with the increasingly complex applications, semiconductor companies are constantly developing processors and accelerators, including CPU, GPU, and TPU. However, with Moore's law slowing down, CPU performance alone will not be enough to execute demanding workloads efficiently.

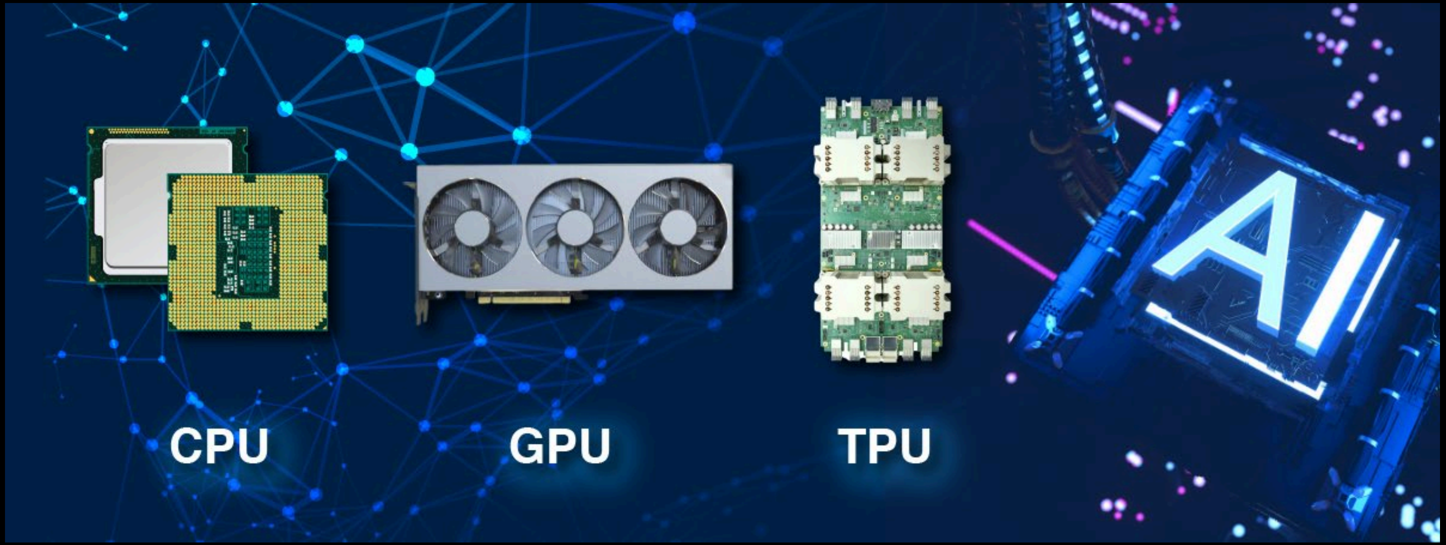


- The problem is, how can companies accelerate the performance of entire systems to support the excessive demands of AI applications? The answer may come via the development of GPUs and TPUs for supplementing CPUs to run deep learning models. That is why it is essential to understand the technologies behind CPU, GPU, and TPU to keep up with the constantly evolving technologies for better performance and efficiency.



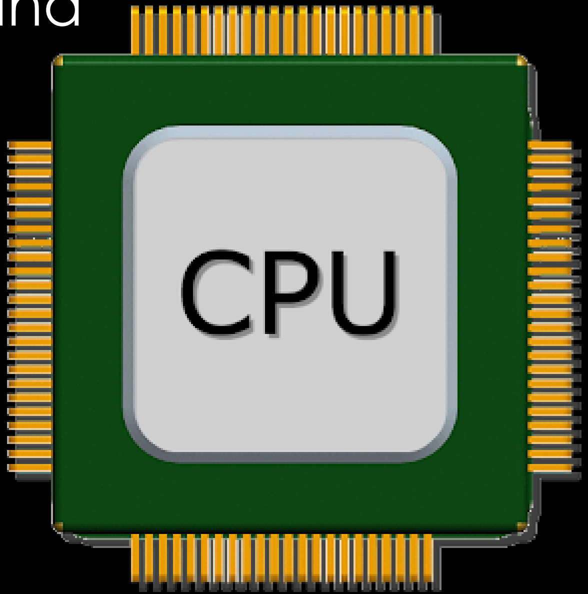
TPU VS GPU VS CPU PERFORMANCE AND DIFFERENCES DISCUSSED

- CPU or Central Processing Unit carries out all the arithmetic and logical operations. On the other hand, the work of a GPU is to render and process images or graphics. TPU is a special type of processor developed by Google. It is used to handle neural network processing using the TensorFlow. CPU can do multiple tasks, including image rendering. But the higher level of image rendering requires a dedicated processor, GPU. That's why high-end games always require a dedicated graphics card.



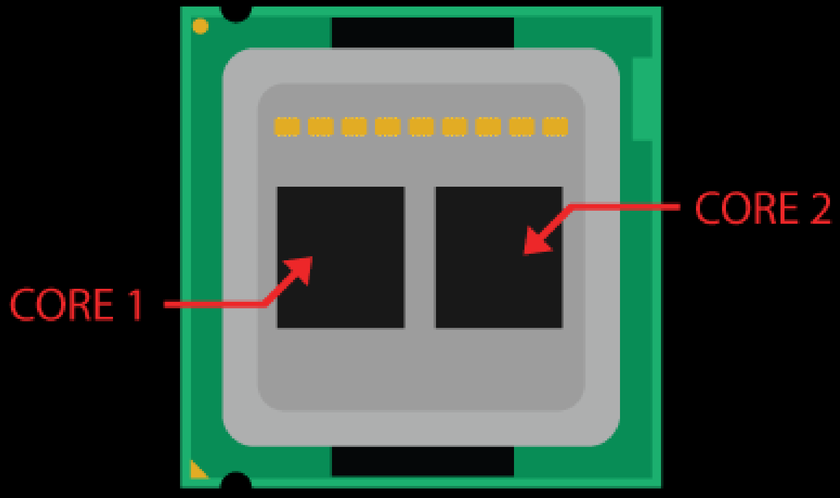
WHAT IS A CPU?

- CPU stands for Central Processing Unit. It is the brain of a computer because it handles all the tasks that a user performs on his/her computer. All the arithmetic and logical calculations required to complete a task are performed by the CPU. The aim of the CPU is to take input from the devices connected to a computer like a keyboard, mouse, etc., or from a programming software and display the required output.



WHAT ARE CPU CORES?

- CPU cores are pathways consisting of billions of microscopic transistors. A CPU uses cores to process data. In simple words, a CPU core is a basic computation unit of a CPU. The number of cores is directly proportional to the computational power of a CPU. The CPU cores define whether the CPU can handle multiple tasks or not. You might have heard the following two types of CPUs:
  - Single-core CPU
  - Multi-core CPU



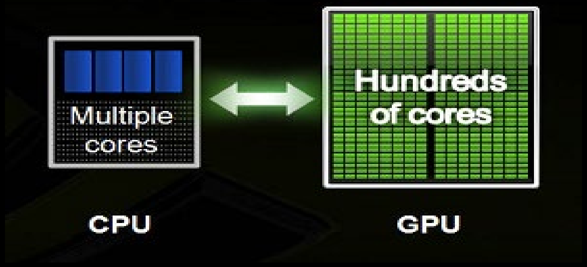
MULTI-CORE CPU

WHAT IS GPU?

- GPU stands for Graphics Processing Unit. A GPU is used in a variety of applications, including image and video rendering. In the field of gaming, graphics cards have a crucial role. A GPU is the main component of a graphics card.
- A GPU is a processor that is good at handling specialized computations.
- This is in contrast to a central processing unit (CPU), which is a processor that is good at handling general computations. CPUs are the processors that power most of the typical computations on our electronic devices.

- A GPU can be much faster at computing than a CPU. However, this is not always the case. The speed of a GPU relative to a CPU depends on the type of computation being performed. The type of computation most suitable for a GPU is a computation that can be done in parallel.
- Parallel computing is a type of computation where by a particular computation is broken into independent smaller computations that can be carried out simultaneously. The resulting computations are then recombined, or synchronized, to form the result of the original larger computation.

- The number of tasks that a larger task can be broken into depends on the number of cores contained on a particular piece of hardware.
- CPUs typically have four, eight, or sixteen cores while GPUs have potentially thousands.



- In parallel computing, an embarrassingly parallel task is one where little or no effort is needed to separate the overall task into a set of smaller tasks to be computed in parallel.
- Tasks that embarrassingly parallel are ones where it's easy to see that the set of smaller tasks are independent with respect to each other.
- Neural networks are embarrassingly parallel for this reason. Many of the computations that we do with neural networks can be easily broken into smaller computations in such a way that the set of smaller computations do not depend on one another. One such example is a convolution.

WHAT IS CUDA?

- Nvidia Hardware (GPU) And Software (CUDA)
- This is where CUDA comes into the picture. Nvidia is a technology company that designs GPUs, and they have created CUDA as a software platform that pairs with their GPU hardware making it easier for developers to build software that accelerates computations using the parallel processing power of Nvidia GPUs.
- CUDA is a parallel computing platform and application programming interface that allows software to use certain types of graphics processing units for general purpose processing, an approach called general-purpose computing on GPUs.
- An Nvidia GPU is the hardware that enables parallel computations, while CUDA is a software layer that provides an API for developers.

- As a result, you might have guessed that an Nvidia GPU is required to use CUDA, and CUDA can be downloaded and installed from Nvidia's website for free.
- Developers use CUDA by downloading the CUDA toolkit. With the toolkit comes specialized libraries like cuDNN, the CUDA Deep Neural Network library.
- The GPU is only faster for particular (specialized) tasks. One issue that we can run into is bottlenecks that slow our performance. For example, moving data from the CPU to the GPU is costly, so in this case, the overall performance might be slower if the computation task is a simple one.
- Deep learning along with many other scientific computing tasks that use parallel programming techniques are leading to a new type of programming model called GPGPU or general purpose GPU computing.
- GPGPU computing is more commonly just called GPU computing or accelerated computing now that it's becoming more common to perform a wide variety of tasks on a GPU.

WHAT IS TPU?

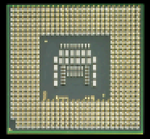
- TPU stands for Tensor Processing Unit. It is a processor developed by Google to handle neural network processing using the TensorFlow. TensorFlow is a free and open-source software library for artificial intelligence and machine learning.

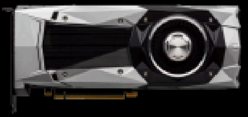
- The core of a TPU developed by Google is made of two units, namely, MXU (Matrix Multiply Unit) and VPU (Vector Processing Unit). The Matrix Multiply Unit performs matrix calculations and operates in a mixed 16 – 32-bit floating point format, whereas the Vector Processing Unit performs float32 and int32 computations.
- Google has developed Cloud TPU to offer maximum flexibility and performance to researchers, developers, and businesses. The main aim to develop TPUs is to minimize the time required to train large and complex neural network models.
- The Cloud TPU accelerates the performance of linear algebra computation, which is used in machine learning applications. Due to this, TPUs are able to minimize the time to accuracy when it comes to training large and complex neural network models.
- If you train neural network models on hardware integrated with TPU, it will take hours, whereas, if the same task when done on the other hardware can take weeks.

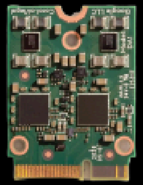
GOOGLE CORAL

- Google Coral is **an edge AI hardware and software platform for intelligent edge devices with fast neural network inferencing.** Coral is Google's initiative for pushing into Edge AI, with machine learning devices that run without a connection to the cloud.

LET'S COMPARE THESE THREE PROCESSORS ON DIFFERENT FACTORS.

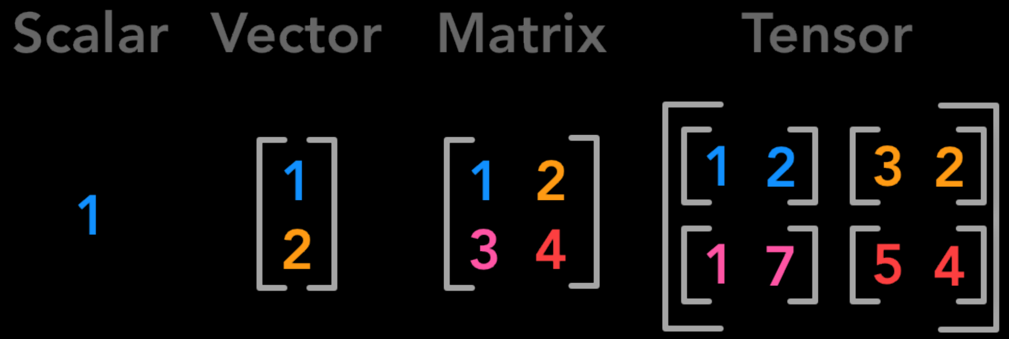






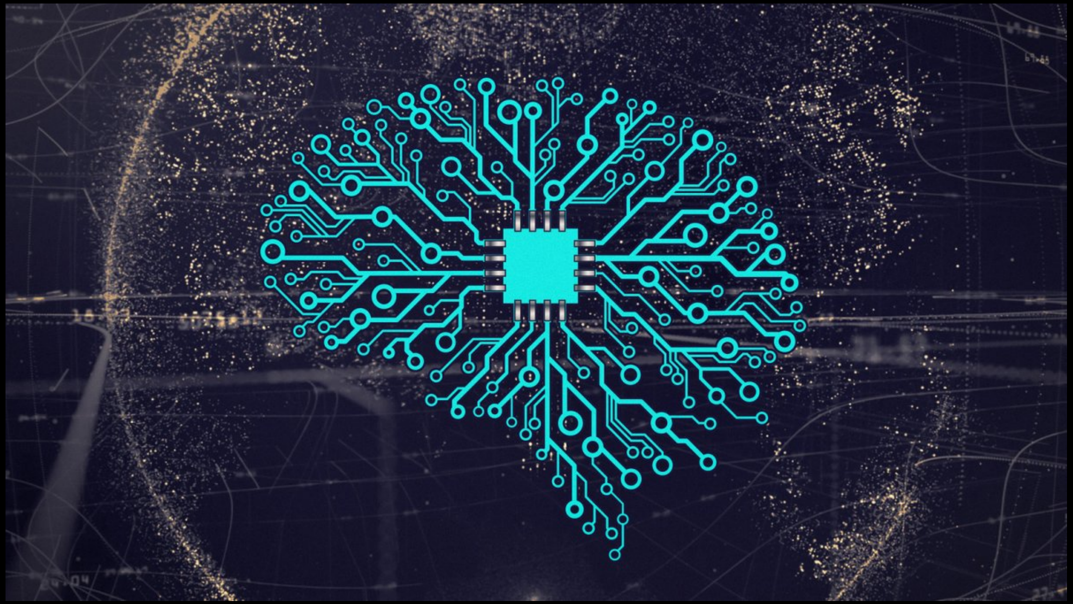
CORES

- **CPU (1x1)**: The number of cores in a CPU includes one (single-core processor), 4 (quad-core processor), 8 (Octa-core processor), etc. The CPU cores are directly proportional to its performance and also make it multitasking.
- **GPU (1xN)**: Unlike a CPU, a GPU has several hundred to several thousand cores. The calculations in a GPU are carried out in these cores. Hence, the GPU performance also depends on the number of cores it has.
- **TPU (NxN)**: According to Google, a single Cloud TPU chip has 2 cores. Each of these cores uses MXUs to accelerate the programs by dense matrix calculations.



ARCHITECTURE

- CPU: A CPU has three main parts, namely, CU, ALU, and Registers. Talking about the registers, there are 5 different types of registers in a CPU. These registers are:
  - Accumulator
  - Instruction Register
  - Memory Address Register
  - Memory Data Register
  - Program Counter

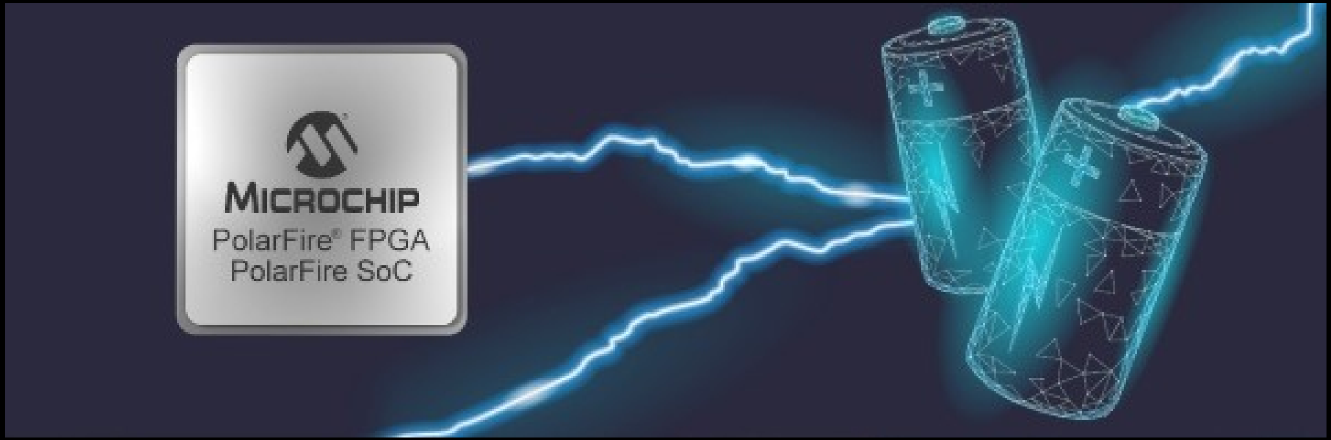


- GPU: As explained above, there are several hundred to several thousand cores in a GPU. All the calculations required to perform image processing and image rendering are done in these cores. Architecturally, the internal memory of a GPU has a wide interface with a point-to-point connection.

- TPU: TPUs are the Machine Learning accelerators designed by Google. Machine Learning accelerators have the potential to boost Machine Learning tasks. The cores of TPU comprise of MXU and VPU that are capable of carrying out the matrix and floating-point calculations respectively.

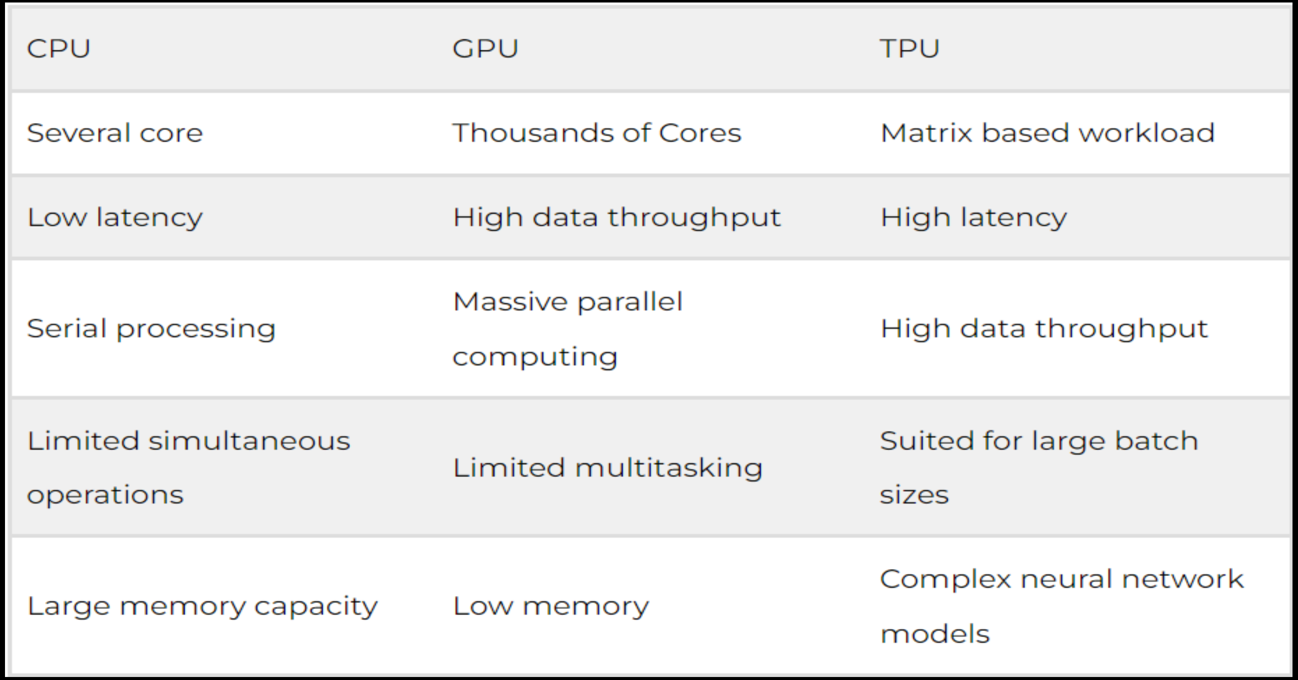
POWER

- CPU: The power consumed by a CPU depends on the number of cores it has. An Octa-core processor consumes power approximately from 95 to 140 watts, whereas a 16-core processor consumes approximately 165 watts of power.
- GPU: A GPU can consume up to 350 watts of power.
- TPU: In a TPU, the process of reading and writing is performed on buffer and memory due to which power optimization can be achieved.



- The cache, the memory design and the entire architecture of the CPU are built around this purpose. CPUs are capable of processing tens of operations per cycle. It has an implicitly managed memory subsystem architecture and is produced by a bunch of manufactures. Data dimensions are usually **1 x 1** data unit.
- The reason behind this is just the way a GPU is built. They can handle tens of thousands of operations per cycle, the dimension of data is generally **1 x N** data unit, they have a mixed memory subsystem architecture and are produced by fewer, specialized manufacturers.
- It has an explicitly managed memory subsystem architecture with data dimension of **N x N** data unit. This means that the TPU can handle up to **128,000** operations per cycle. That number isn't even close to what the high-end CPUs and GPUs can manage.

- In summary, CPUs are versatile processors that are capable of performing a wide range of tasks, but they are not as efficient as GPUs or TPUs for deep learning workloads. GPUs are specialized processors that are optimized for running many calculations simultaneously and are well-suited for deep learning workloads. TPUs are custom-built processors designed specifically for accelerating machine learning workloads and are the most efficient for deep learning workloads.



CPU	GPU	TPU
Several core	Thousands of Cores	Matrix based workload
Low latency	High data throughput	High latency
Serial processing	Massive parallel computing	High data throughput
Limited simultaneous operations	Limited multitasking	Suited for large batch sizes
Large memory capacity	Low memory	Complex neural network models

WHAT IS TENSORFLOW?

- TensorFlow is an end-to-end open-source deep learning framework developed by Google and released in 2015. It is known for documentation and training support, scalable production and deployment options, multiple abstraction levels, and support for different platforms, such as Android.
- TensorFlow is a symbolic math library used for neural networks and is best suited for dataflow programming across a range of tasks. It offers multiple abstraction levels for building and training models.
- A promising and fast-growing entry in the world of deep learning, TensorFlow offers a flexible, comprehensive ecosystem of community resources, libraries, and tools that facilitate building and deploying machine learning apps. Also, as mentioned before, TensorFlow has adopted Keras, which makes comparing the two seem problematic. Nevertheless, we will still compare the two frameworks for the sake of completeness, especially since Keras users don't necessarily have to use TensorFlow.

WHAT IS PYTORCH?

- PyTorch is a deep learning framework and a scientific computing package. This is how the PyTorch core team describes PyTorch, anyway. The scientific computing aspect of PyTorch is primarily a result PyTorch's tensor library and associated tensor operations.
- A tensor is an n-dimensional array.
- Tensors are super important for deep learning and neural networks because they are the data structure that we ultimately use for building and training our neural networks.
- **PyTorch: A Brief History**
- The initial release of PyTorch was in October of 2016, and before PyTorch was created, there was and still is, another framework called Torch. Torch is a machine learning framework that's been around for quite a while and is based on the Lua programming language.
- The connection between PyTorch and this Lua version, called Torch, exists because many of the developers who maintain the Lua version are the individuals who created PyTorch.
- PyTorch's design is modern, Pythonic, and thin. The source code is easy to read for Python developers because it's written mostly in Python, and only drops into C++ and CUDA code for operations that are performance bottlenecks.

WHAT IS JAX?

- JAX is a framework for machine learning that allows you to use Python and **NumPy** to create and train neural networks. JAX is similar to other popular frameworks such as **PyTorch** and **TensorFlow**, but it has some unique features that make it a good choice for certain tasks.

- **Features of JAX:**
  - Automatic Differentiation (AD):
    - JAX is based on the concept of "function transformations". This means that you can define a function, and then JAX will automatically compute the derivative of that function.
    - AD is a technique for efficiently computing the derivatives of functions. This is useful for a variety of tasks, such as optimizing neural networks and training machine learning models.
    - AD is typically performed using reverse-mode differentiation, which is a way of computing derivatives that is well-suited for deep learning. JAX implements reverse-mode differentiation using a technique called trace-based reverse-mode differentiation.
  - Just-in-time Compilation(JIT):
    - which is a technique for compiling code on the fly. JIT compilation can be used to improve the performance of numerical code, such as the code used in deep learning.
    - JAX uses a JIT compiler called XLA, which is also used by TensorFlow. XLA is a high-performance compiler that is able to optimize code for a variety of architectures, including CPUs, GPUs, TPUs, and custom hardware accelerators.
  - Memory Management:
    - JAX includes a number of features that help to manage memory usage. These features are particularly important for deep learning, which can require a large amount of memory.
    - JAX includes a number of memory management features, such as automatic garbage collection and reference counting. These features help to ensure that memory is freed when it is no longer needed.

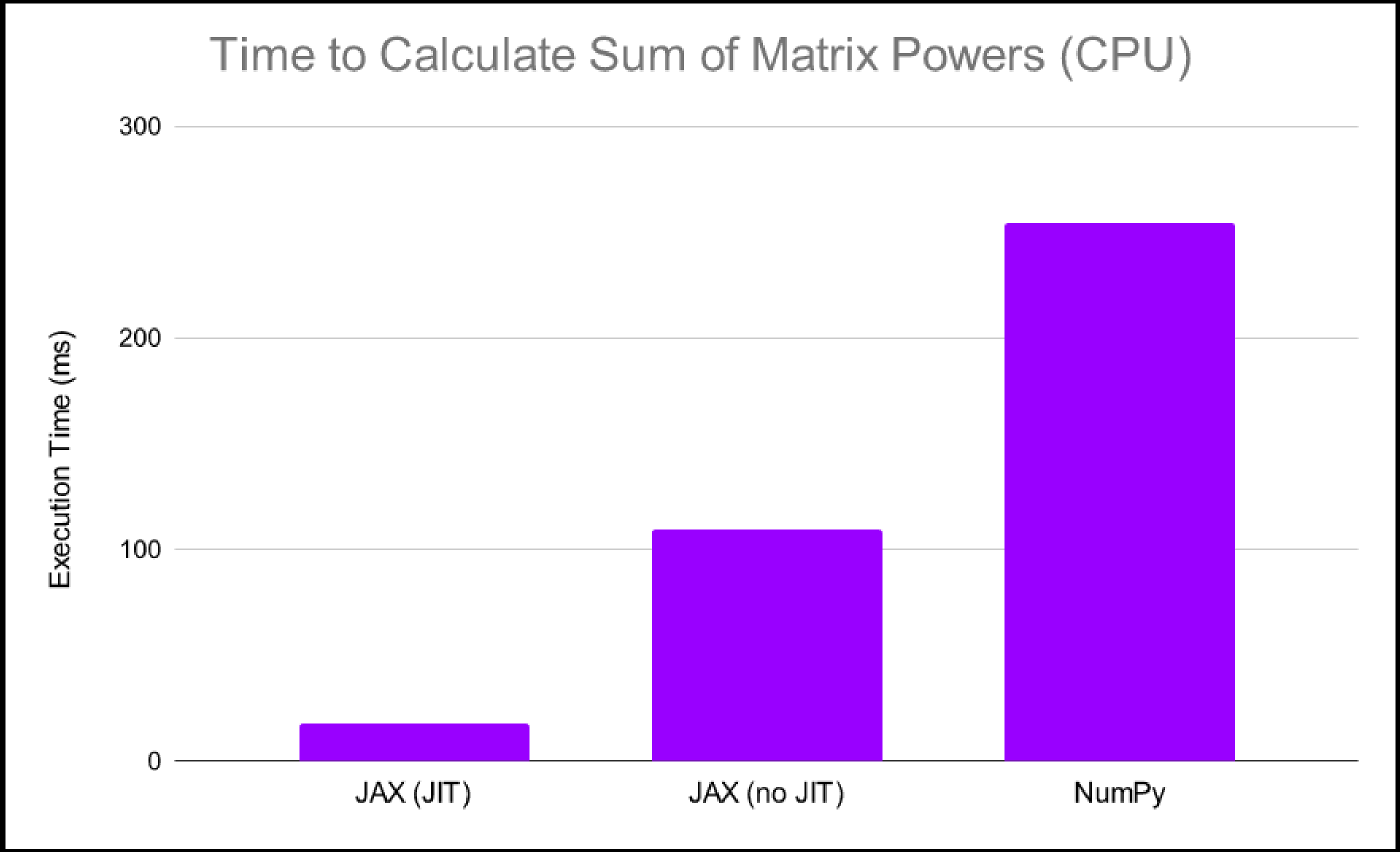
JAX - FLAX, HAIKU, ELEGY.

- There are six different JAX-based neural net libraries: Stax, Flax, Trax, Objax, Haiku, Elegy.

INCREASE THE SPEED

- In computing, just-in-time(Jit) compilation is a way of executing computer code that involves compilation during execution of a program rather than before execution. This may consist of source code translation but is more commonly bytecode translation to machine code, which is then executed directly.
- XLA (accelerated linear algebra) is a compiler-based linear algebra execution engine. It is the backend that powers machine learning frameworks such as TensorFlow and JAX at Google, on a variety of devices including CPUs, GPUs, and TPUs. This talk will cover how ML growth has fueled accelerator architectures and the way XLA and related technologies help obtain high performance from the accelerators.

EXAMPLE OF USING JIT COMPILER



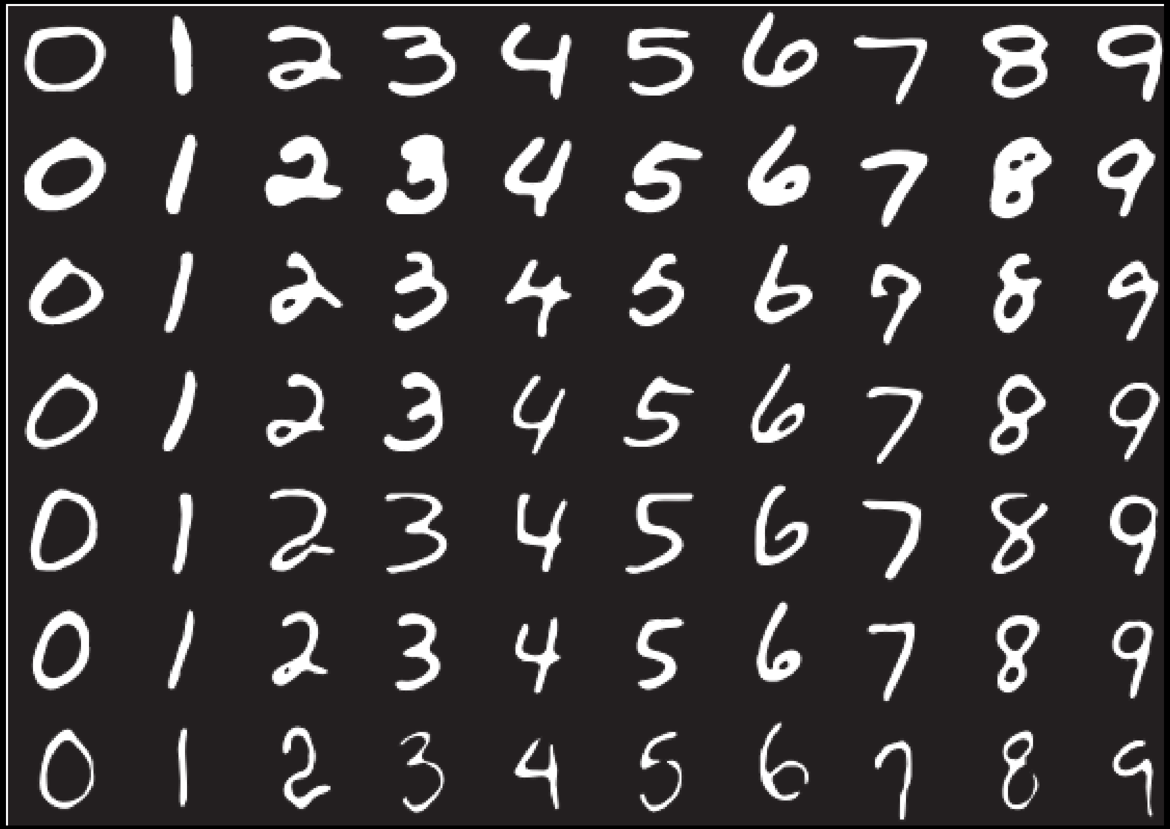
STEPS TO MAKE A COMPARISON:

- In this project:
  1. run different frameworks like:
     i. Tnesorflow.
     ii. Pytorch.
     iii. JAX.
  2. use different accelerators like:
     i. CPU.
     ii. GPU.
     iii. TPU.
  3. use benchmark dataset like MNIST dataset.
  4. Integrate with only one configuration of Convolutional Neural Network.
  5. Use the same hyperparameters in each notebook.

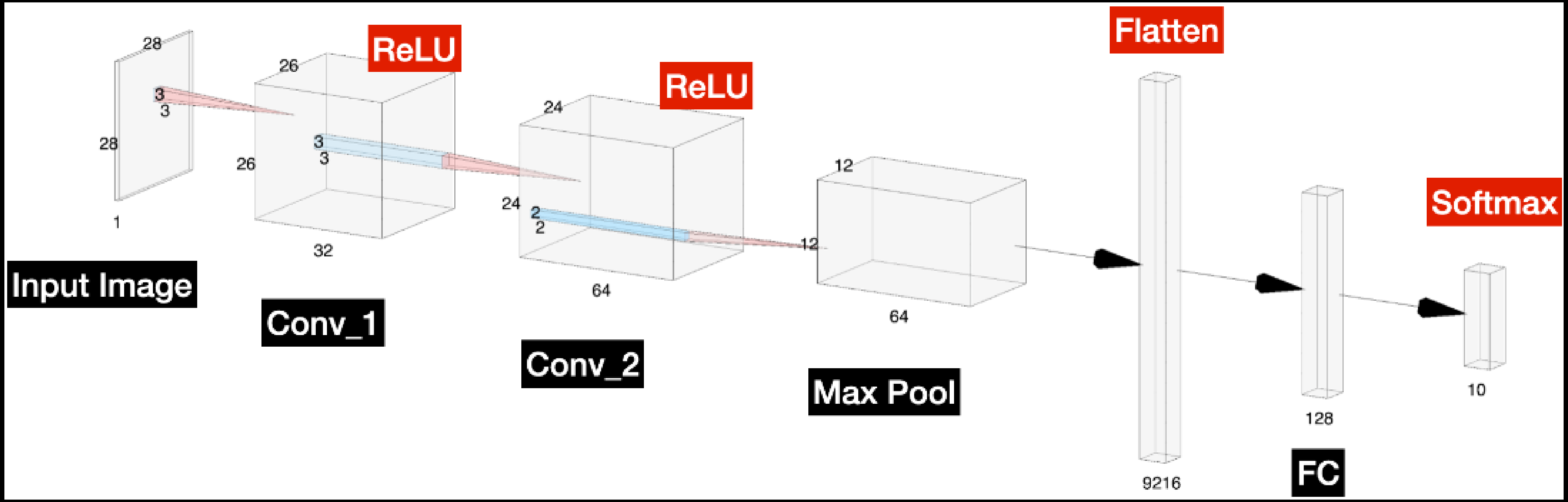
DATASET:

- MNIST (http://yann.lecun.com/exdb/mnist) stands for Modified National Institute of Standards and Technology.
- It contains labeled handwritten images of digits from 0 to 9.
- The goal of this dataset is to classify handwritten digits.
- MNIST has been popular with the research community for benchmarking classification algorithms.
- In fact, it is considered the “hello, world!” of image datasets.
- MNIST consists of 60,000 training images and 10,000 test images.
- All are grayscale (one-channel), and each image is 28 pixels high and 28 pixels wide.

- Figure 6.12 shows some sample images from the MNIST dataset.



CONVOLUTIONAL NEURAL NETWORK:



SHOWING RESULTS

Tensorflow Runtime

CPU:

- **TRAINING TIME:** 1286.52 seconds
- **ACCURACY:** 92.75%

GPU:

- **TRAINING TIME:** 77.778 seconds
- **ACCURACY:** 92.94%

TPU:

- **TRAINING TIME:** 268.768 seconds
- **ACCURACY:** **96.4%**

Pytorch Runtime

CPU:

- **TRAINING TIME:** 3961.678 seconds
- **ACCURACY:** 94.34%

GPU:

- **TRAINING TIME:** 751.83 seconds
- **ACCURACY:** 94.38%

TPU:

- **TRAINING TIME:**
- **ACCURACY:**

JAX Runtime:

CPU:

- **TRAINING TIME:**
- **ACCURACY:**

GPU:

- **TRAINING TIME:** 94.98 seconds
- **ACCURACY:** 97.9%

TPU:

- **TRAINING TIME:**
- **ACCURACY:**

- TensorFlow uses a static computation graph while PyTorch uses a dynamic computation graph. This means that in TensorFlow, the user first defines the computation graph and then runs it, while in TensorFlow, the user can change the graph on the fly during runtime. TensorFlow's static graph can be optimized for performance by the TensorFlow runtime, leading to faster execution times. Additionally, TensorFlow has a more efficient memory management system compared to PyTorch which also contributes to its faster runtime.

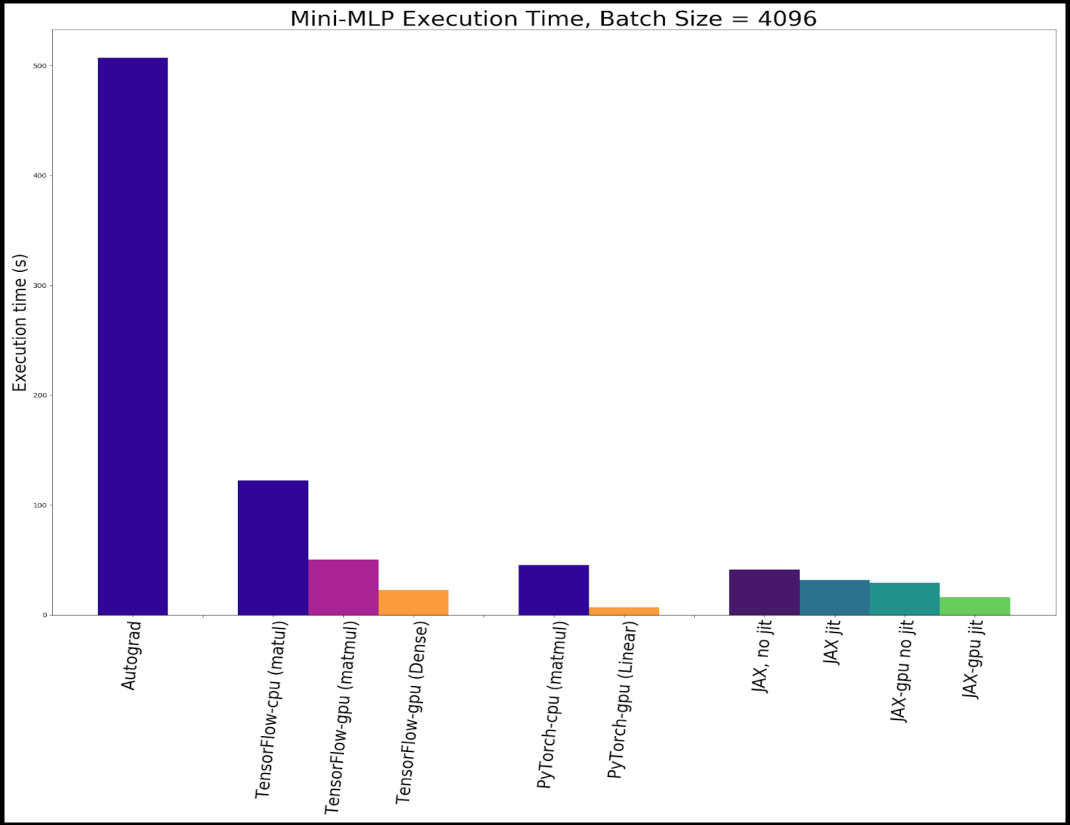
- TensorFlow 's static computation graph allows for more optimization opportunities prior to the model being run. The TensorFlow runtime can analyze the entire graph and apply optimizations such as kernel fusion and memory reuse. Additionally, TensorFlow has a **Just-in-time (JIT)** compiler which further optimizes the computation graph at runtime.

- In contrast, PyTorch's dynamic computation graph does not allow for as much optimization before the model is run and requires more memory to store the intermediate results. However, PyTorch's dynamic computation graph allows for more flexibility and ease of use, making it more suited for research and development.

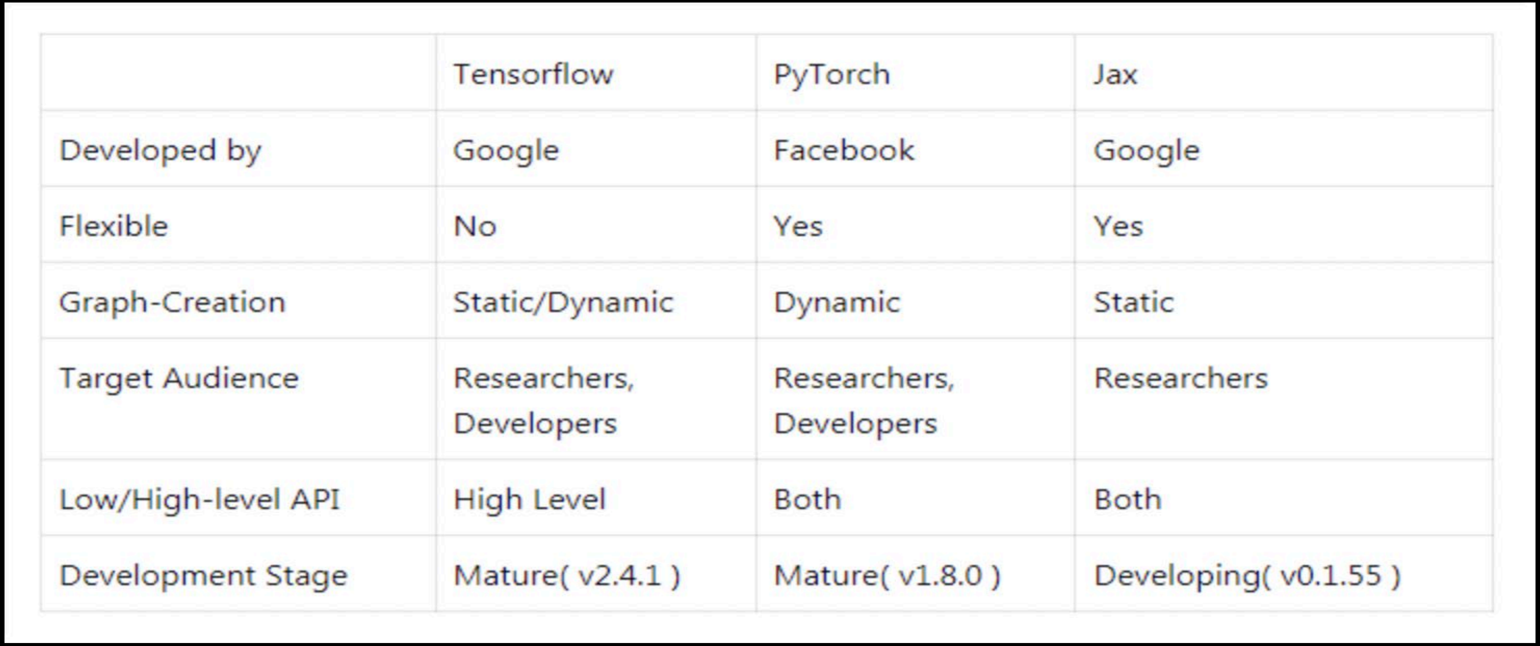
- Another factor is TensorFlow has a more efficient memory management system and a better support for distributed computing which allow large models to be trained and deployed on multiple machines and GPUs.

- It's also worth noting that the performance difference between TensorFlow and PyTorch varies depending on the specific task and model architecture. In some cases, PyTorch may be faster than TensorFlow, while in others TensorFlow may be faster. It depends on the specific use case and the requirements of the user.

- **PyTorch** is more popular than TensorFlow because it is more user-friendly and has a more "**Pythonic**" feel, meaning it is more similar to the standard Python language. Additionally, PyTorch includes features such as dynamic computational graphs, which make it more flexible and easier to use for research and experimentation. PyTorch also has a more active community and more resources available for learning and troubleshooting.



- In summary, TensorFlow and PyTorch are both powerful and popular deep learning frameworks, but they have different strengths. TensorFlow is more mature and has a wider range of tools and libraries available, while PyTorch is known for its ease of use and flexibility. Jax is still new in comparison but has a lot of powerful features and has been used in many papers on **arxiv**.



	Tensorflow	PyTorch	Jax
Developed by	Google	Facebook	Google
Flexible	No	Yes	Yes
Graph-Creation	Static/Dynamic	Dynamic	Static
Target Audience	Researchers, Developers	Researchers, Developers	Researchers
Low/High-level API	High Level	Both	Both
Development Stage	Mature(v2.4.1)	Mature(v1.8.0)	Developing(v0.1.55)

COMPARISON CONCLUSION

- So which deep learning framework should you use? It is not possible to give a general answer to this question. It all depends on the type of problem you want to solve and the scale at which you want to use your models . The computing platforms you are targeting also play a role:
  - If you work in text and image space and do small or medium research with the aim of using the models in production, **PyTorch** is probably your best choice for that at the moment.
  - However, if you want to squeeze the last bit of performance out of computing-weak devices, **TensorFlow** is recommended.
  - If you work on training models with tens or hundreds of billions of parameters or more and train them mainly for research purposes, then you should give **JAX** a try.

REFERENCES:

- https://premioinc.com/blogs/blog/what-is-the-difference-between-cpu-vs-gpu-vs-tpu-complete-overview
- https://iq.opengenus.org/cpu-vs-gpu-vs-tpu/
- https://www.kaggle.com/discussions/getting-started/340280
- https://jax.readthedocs.io/en/latest/installation.html
- https://github.com/TensorflowXLABeginner/XLA-Report/blob/master/FirstCommitReports/Accelerated%20Linear%20Algebra%20Intro.md
- https://towardsdatascience.com/how-to-accelerate-your-pytorch-training-with-xla-on-aws-3d599bc8f6a9
- https://www.kaggle.com/code/tanulsingh077/pytorch-xla-understanding-tpu-s-and-xla
- https://pytorch.org/xla/release/1.13/index.html#creating-an-xla-tensor
- https://medium.com/pytorch/get-started-with-pytorch-cloud-tpus-and-colab-a24757b8f7fc#:~:text=PyTorch%20uses%20Cloud%20TPUs%20just,as%20a%20different%20PyTorch%20device.&text=And%20tensors%20can%20be%20transferred%20between%20CPU%20and%20TPU

- SGD implementation in PyTorch. The subtle difference can affect your… | by Ceshine Lee | Veritable | Medium
- Running PyTorch on TPU: a bag of tricks | by Zahar Chikishev | Towards Data Science
- PyTorch vs TensorFlow: In-Depth Comparison (phoenixnap.com)
- Pytorch vs Tensorflow: A Head-to-Head Comparison - viso.ai
- https://github.com/pytorch/xla#how-to-run-on-tpu-vm-pods-distributed-training
- https://www.computerwoche.de/a/deep-learning-frameworks-im-vergleich,3612680
- https://www.exxactcorp.com/blog/Deep-Learning/accelerated-automatic-differentiation-with-jax-how-does-it-stack-up-against-autograd-tensorflow-and-pytorch
- https://blog.paperspace.com/jax-on-paperspace/