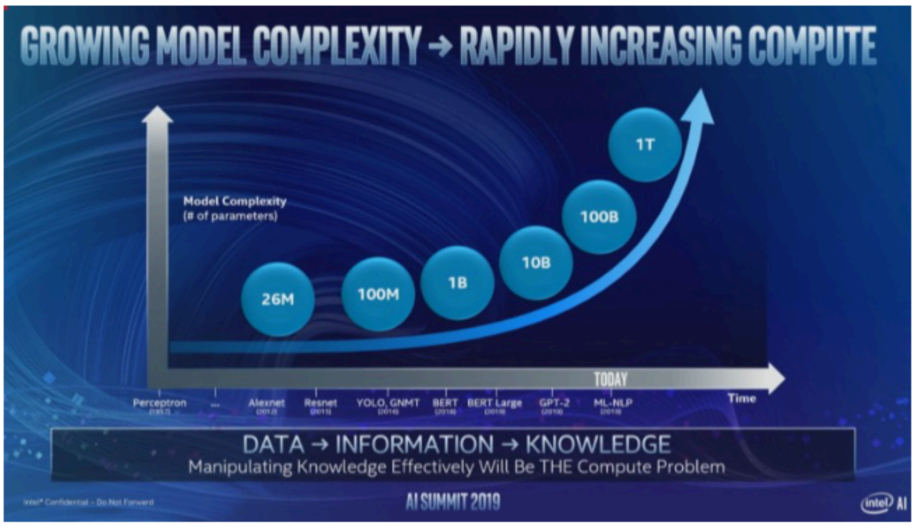
Need of HW acceleration



Complexity doubles every 3.5 months

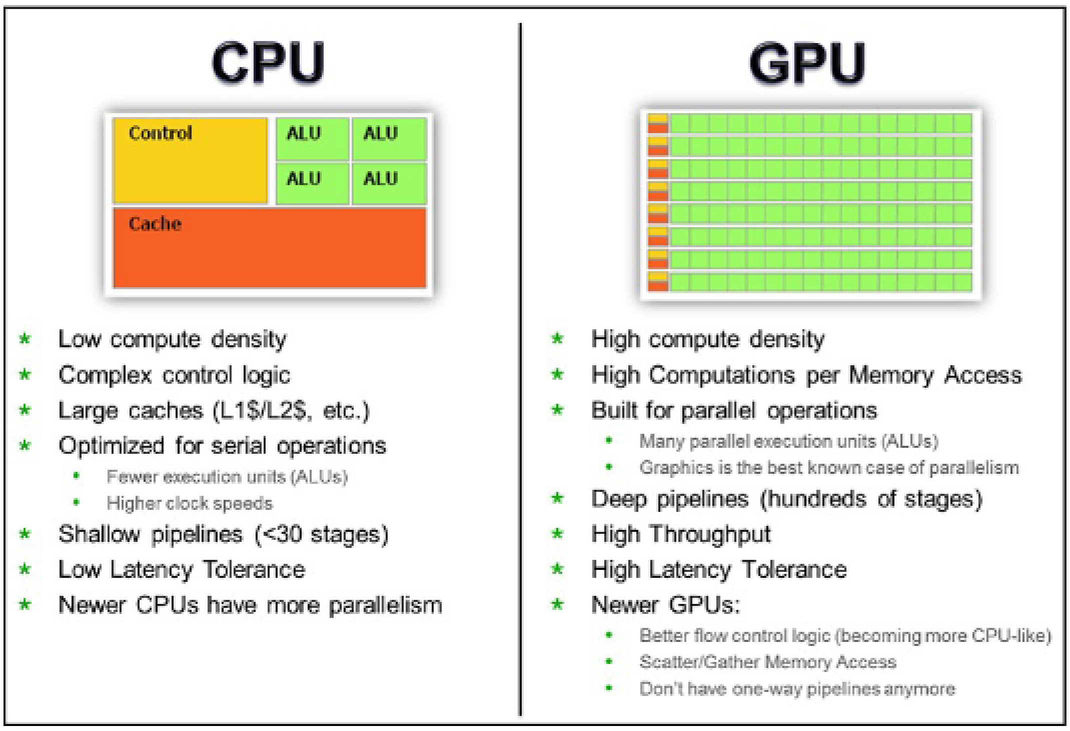
↓

18-24 months for Moore's Law

↓

Nowadays Moore's Law capacity double every 4 or more years

Introducing parallel solutions to make training faster.
First solution implemented are GPU.


Control
ALU
ALU
ALU
ALU
Cache

CPU

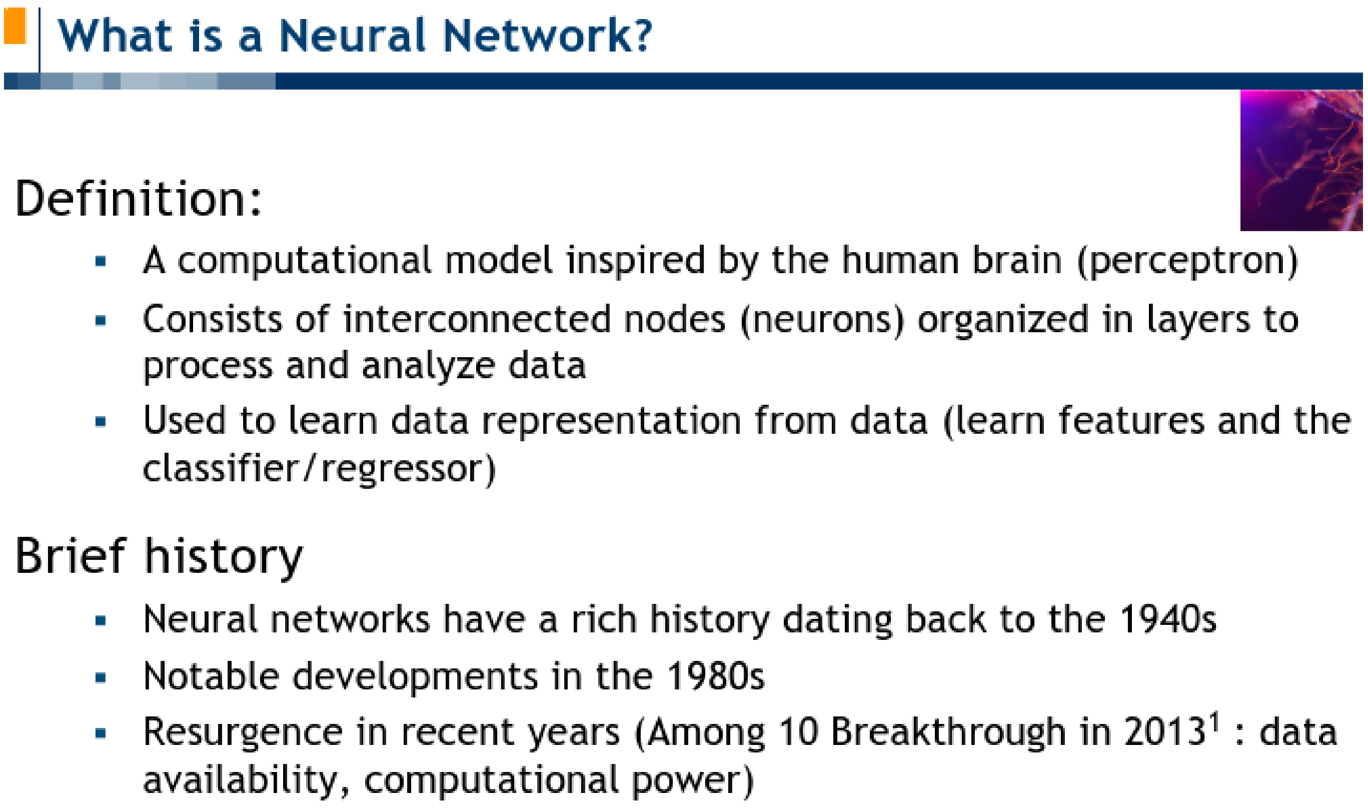
* Low compute density
* Complex control logic
* Large caches (L1$/L2$, etc.)
* Optimized for serial operations
  * Fewer execution units (ALUs)
  * Higher clock speeds
* Shallow pipelines (<30 stages)
* Low Latency Tolerance
* Newer CPUs have more parallelism

GPU

* High compute density
* High Computations per Memory Access
* Built for parallel operations
  * Many parallel execution units (ALUs)
  * Graphics is the best known case of parallelism
* Deep pipelines (hundreds of stages)
* High Throughput
* High Latency Tolerance
* Newer GPUs:
  * Better flow control logic (becoming more CPU-like)
  * Scatter/Gather Memory Access
  * Don't have one-way pipelines anymore

Deep learning models began to appear and widely adopted. Usually there are error only when images are low quality, it doesn't saturate as the common models.
Most of the time critical operation are multiplication between matrix/vector or matrix vector multiplication.

GPUs are up to 1000x faster than CPUs, but they need high level languages (CUDA and OpenCL).



What is a Neural Network?

Definition:

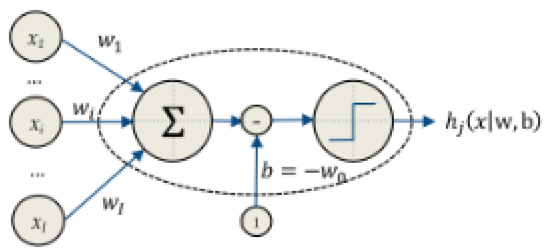
- A computational model inspired by the human brain (perceptron)
- Consists of interconnected nodes (neurons) organized in layers to process and analyze data
- Used to learn data representation from data (learn features and the classifier/regressor)

Brief history

- Neural networks have a rich history dating back to the 1940s
- Notable developments in the 1980s
- Resurgence in recent years (Among 10 Breakthrough in 2013[1] : data availability, computational power)

IN 1980 the Back Propagation algorithm was released helping the training of ML models

Artificial Neurons



$$h_j(x|w,b) = h_j\left(\Sigma_{i=1}^{I} w_i \cdot x_i - b\right) = h_j\left(\Sigma_{i=0}^{I} w_i \cdot x_i\right) = h_j(w^T x)$$

Then these neurons are stacked and connected between them with their own weight and activation function.
In training activation functions need derivatives and the complexity depends from how much neurons are stacked. If you want to approximate a function if you consider a single layer you can approximate any function using a certain number of neurons.
Training:

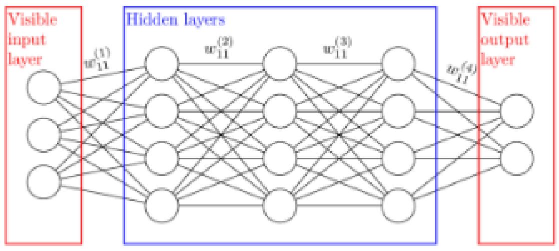
- Compute an error from each computing network using the results of computation.
- Train the model with a gradient approach where you try to minimize the error.

Learning process:

- Neurons make decisions (activation functions)
- Weights : connections between neurons are strengthened or weakened through training

Training data

- NN learn from historical data and examples



Backpropagation (Gradient descent, Chain rule)

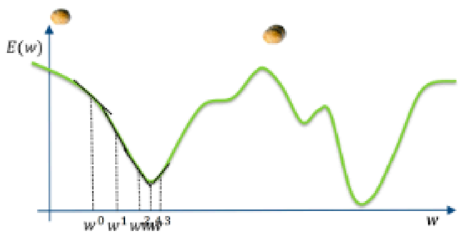
- Errors are calculated and used to adjust the model

$$E = \sum_n^N \left(t_n - g(x_n|w)\right)^2$$

$$w^{k+1} = w^k - \Delta \frac{\partial E(w)}{\partial w}$$

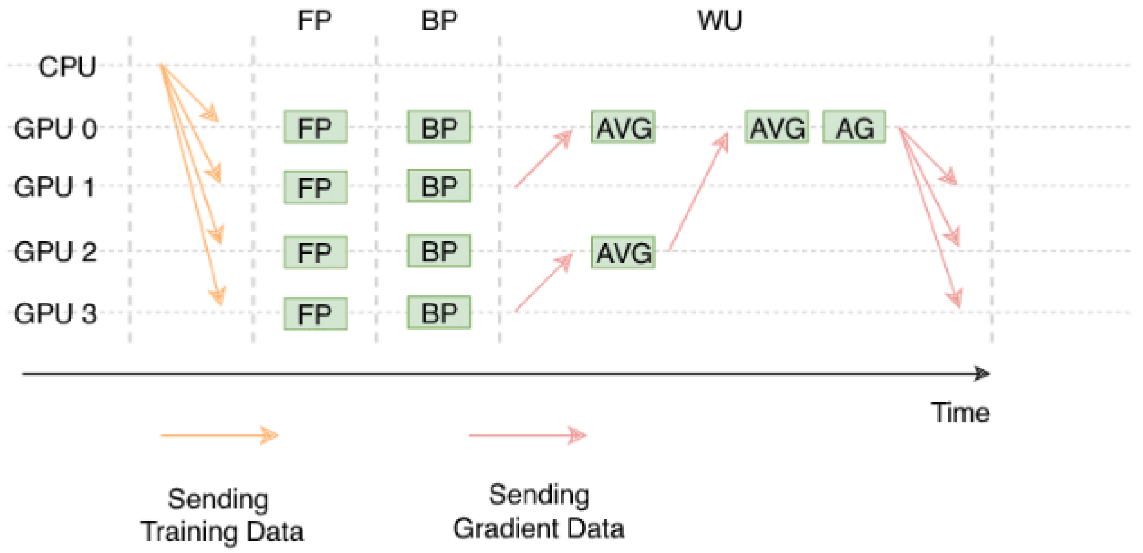
t_n: desired target

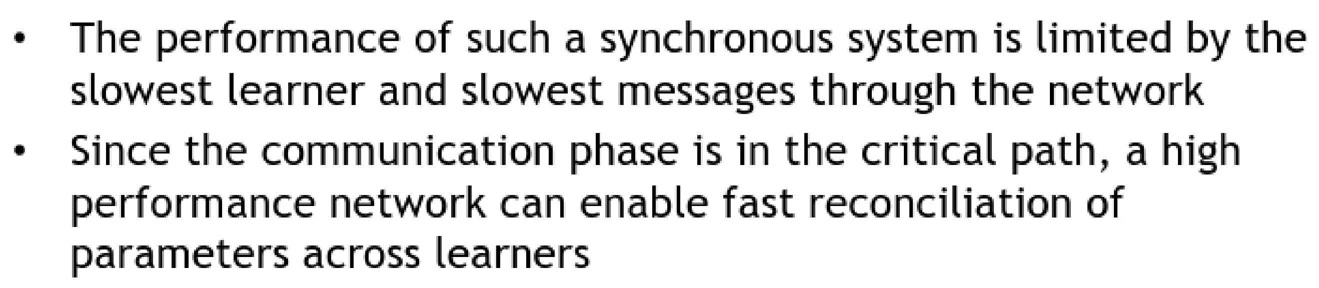
$g(x_n|w)$: learned from data



With Backpropagation I can add the result of the computation to the weight having an improvement in my results. The weight update are made for each sample category, but impossible consider all the dataset so we consider a batch of the dataset so we can have an approximation of the weight change.
Usually there are many local optimal changes.
Networks applied today consist of various layer stacked, the network expands very quickly. For each tasks there are specific models. On the inputs there could be applied filters with a weight for each input. If you decrease the size of the filter you will increase the weight of the single filter. Good architecture for image recognition. Special models for text recognition.

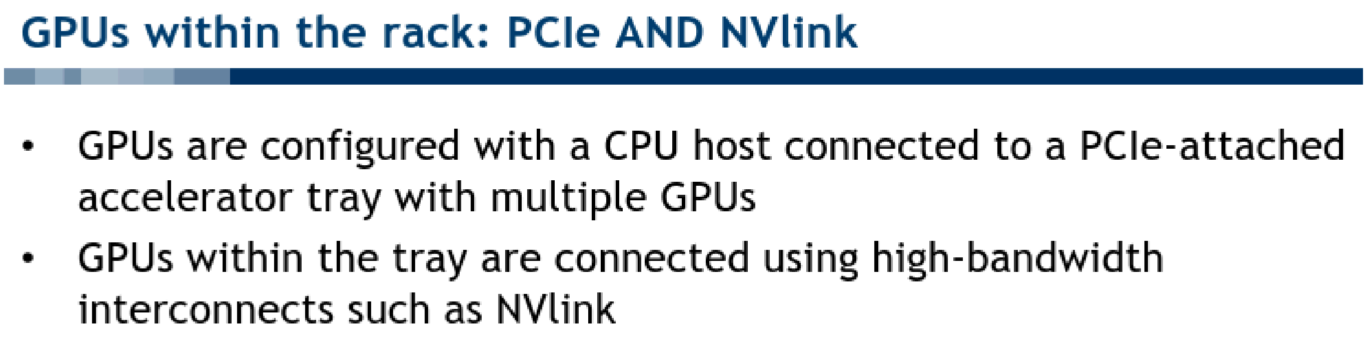
GPU: training a DNN on multiple GPUs





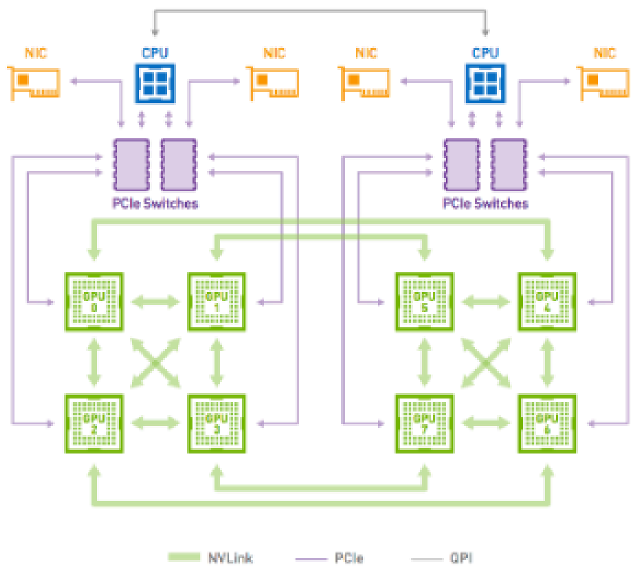
- The performance of such a synchronous system is limited by the slowest learner and slowest messages through the network
- Since the communication phase is in the critical path, a high performance network can enable fast reconciliation of parameters across learners

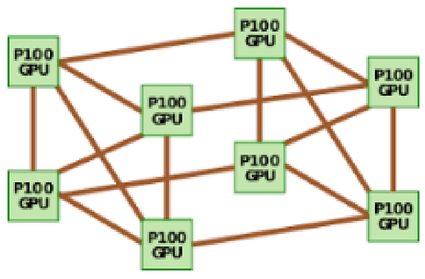
You can also divide the image on multiple GPUs. IN this case every GPU work on the data it received and updates its model indipendently from the others, creating multiple models. So to have a single model each GPUs shares its weight with the others and the sum of the weight is used to update the general model updating then the GPUs models.



GPUs within the rack: PCIe AND NVlink

- GPUs are configured with a CPU host connected to a PCIe-attached accelerator tray with multiple GPUs
- GPUs within the tray are connected using high-bandwidth interconnects such as NVlink



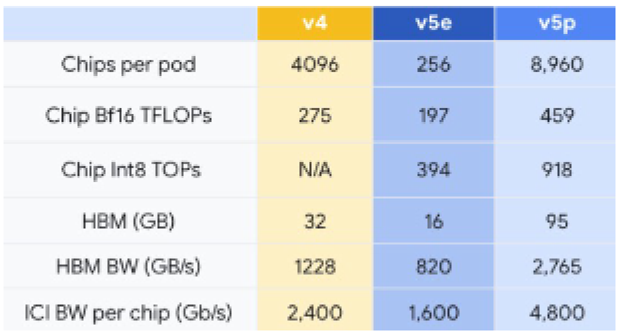


Toroidal topology, each GPU can talk with the others, the channels are bidirectional. Good having a network card that can directly talks with the GPUs to have direct communication between Network data and GPUs without the need to copy it in memory.
Several generation of NVLink. Each one add some lines(Today we have 18 lanes with a maximum of 900 GB/s total).

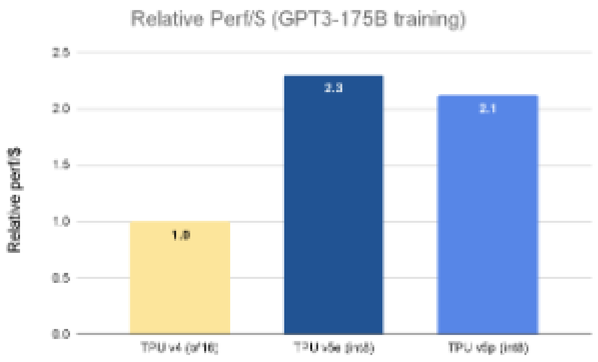
GPu training DNN on multiple GPUs has become to heavy in memory(3.2 TB for GPT 3.5) so it is needed the work of multiple GPUs only to sustain the quantity of data(Each GPU can arrive to 80 GB). Memory is used mainly to store batches of the training set. Use multiple nodes to evolve the model.

Tensor processing unit(TPU)

Devices used only for machine learning. Data Centers could not handle the requests from models and application. Need of an architecture for ML loads.
A Tensor is an n-dimensional matrix. This is the basic unit of operation in TensorFlow.
TPUs are used for training and inference: TPUv1 is an inference-focused accelerator connected to the host CPU through PCIe links, differently, TPUv2, TPV3, and TPV4 focus on training and inference
In TPUv2 you can access a fast memory. The memory was 10x faster than DDR4, today is 20x faster than DDR4. These modules can be put in clusters called POD with up to 512 TPU and 4 TB of total memory(Usually they have 64 units).
TPUv3 the devices could be watercooled. You don't only perform training but also AutoML(choose the best model for the task, can change it during the operations, selects the best solution for the task), new ML capabilities.
TPUv4 was announced during 2021, initially used only to support google services(not available as a cloud service). Now there are also TPUv5 with two version v5e and v5p



	v4	v5e	v5p
Chips per pod	4096	256	8,960
Chip Bf16 TFLOPs	275	197	459
Chip Int8 TOPs	N/A	394	918
HBM (GB)	32	16	95
HBM BW (GB/s)	1228	820	2,765
ICI BW per chip (Gb/s)	2,400	1,600	4,800



Amazon also has created its architectures optimized for LLMs models. These architectures are called Trainium.

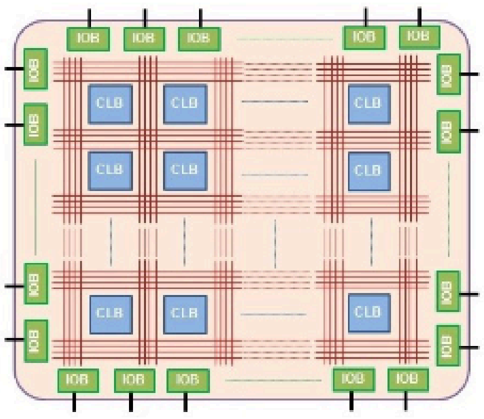
TPUs vs Trainium

	TPU	Trainium2
Architecture	Custom ASIC	Modified NVIDIA Ampere GPU
Cores	8,960	512
Memory	95GB HBM	640GB GDDR6
Performance	Up to 459 teraflops	Up to 450 teraflops
Efficiency	Up to 30x better than CPUs	Up to 10x better than CPUs

Field Programmable Gate Array(FPGA)

You can customise the hardware to execute a single task. Set of configurable logic blocks that can implements various task.

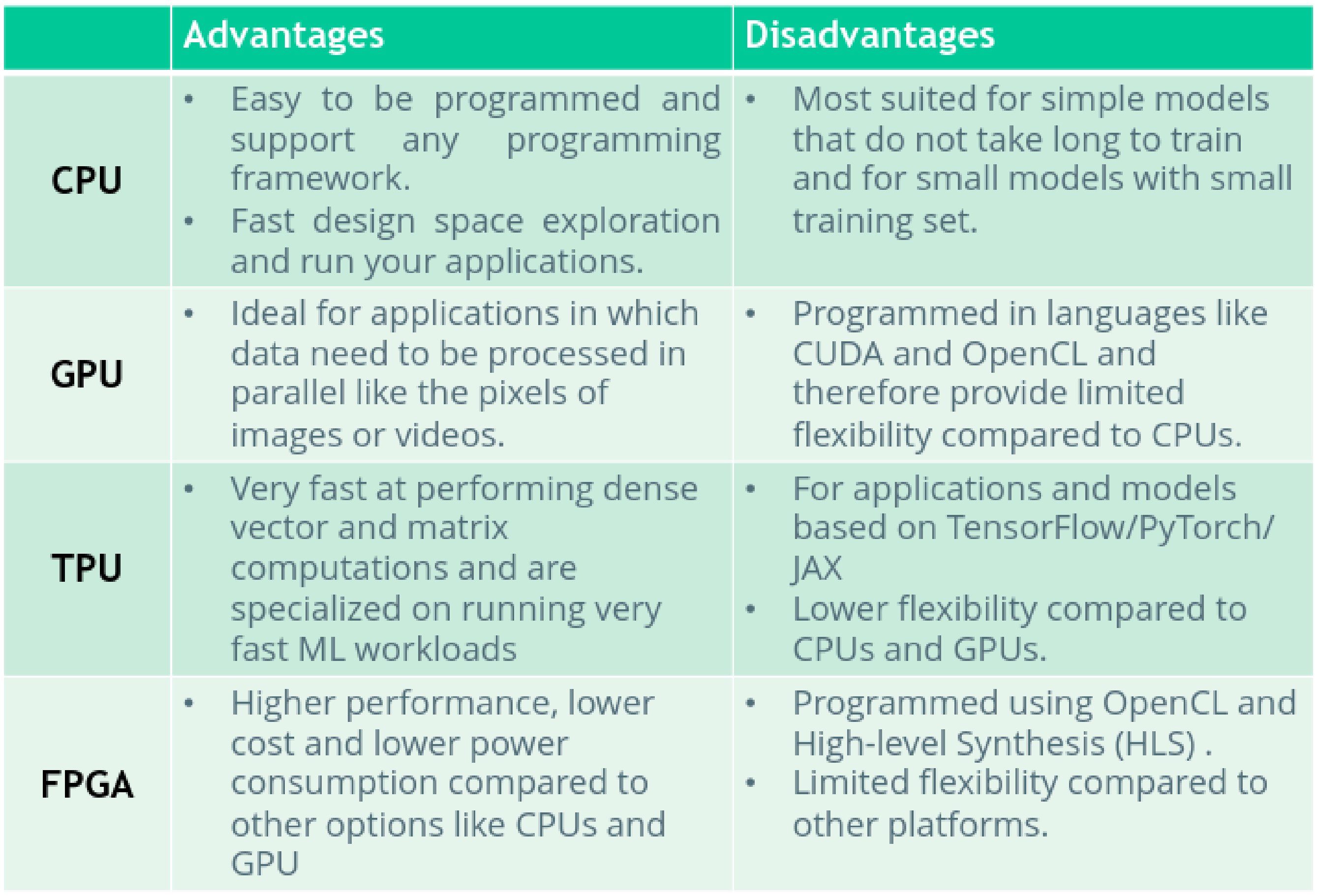
Array of logic gates that can be programmed ("configured") in the field, i.e., by the user of the device as opposed to the people who designed it
Array of carefully designed and interconnected digital subcircuits that efficiently implement common functions offering very high levels of flexibility. The digital subcircuits are called configurable logic blocks (CLBs)



- ✓ VHDL and Verilog are hardware description languages (HDLs) languages that allow to "describe" hardware;
- ✓ HDL code is more like a schematic that uses text to introduce components and create interconnections.

They are not so fast but having the possibility to create new circuit for various task, they are energy efficient.

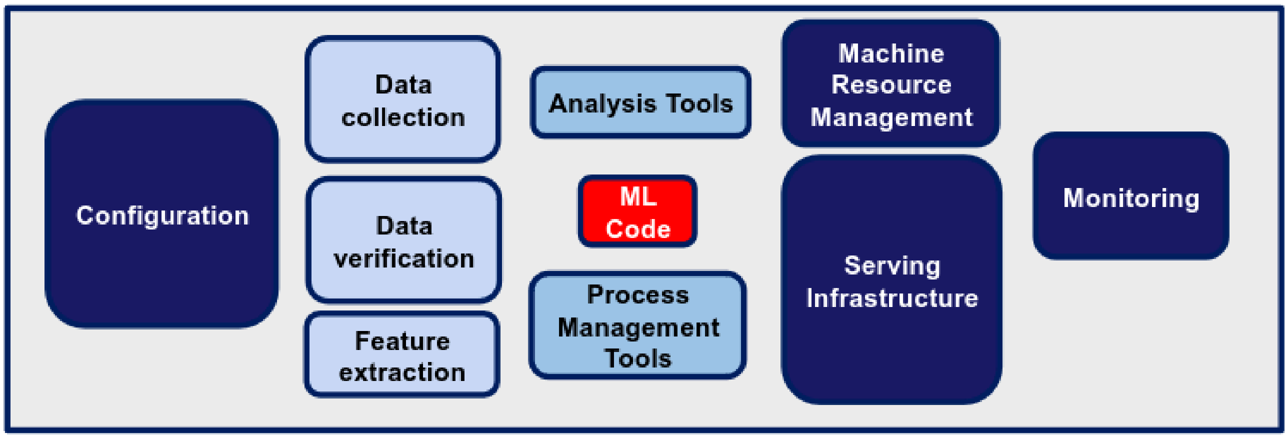
GPU are general enough, FPGA are between TPU and GPU, they are flexible and general purpose and then there is TPU where the chipset is made to compute a single task.



	Advantages	Disadvantages
CPU	• Easy to be programmed and support any programming framework. • Fast design space exploration and run your applications.	• Most suited for simple models that do not take long to train and for small models with small training set.
GPU	• Ideal for applications in which data need to be processed in parallel like the pixels of images or videos.	• Programmed in languages like CUDA and OpenCL and therefore provide limited flexibility compared to CPUs.
TPU	• Very fast at performing dense vector and matrix computations and are specialized on running very fast ML workloads	• For applications and models based on TensorFlow/PyTorch/JAX • Lower flexibility compared to CPUs and GPUs.
FPGA	• Higher performance, lower cost and lower power consumption compared to other options like CPUs and GPU	• Programmed using OpenCL and High-level Synthesis (HLS) . • Limited flexibility compared to other platforms.

GPUs can be more energy efficient than CPUs in data center as one can substitute 40 CPUs.

Hardest part of AI isn't AI



Only a small fraction of real world ML systems is composed of the ML code[1]

[1] Hidden Technical Debt in Machine Learning Systems, Google. NIPS 2015

ML code is 10% of the code in Google ML centre. There is a lot of infrastructure and other logistic code to be implemented to use the ML code efficiently. Serving infrastructure is really important as it permit to speed up and optimise all the other operation. The Computer Infrastructure matter a lot today.