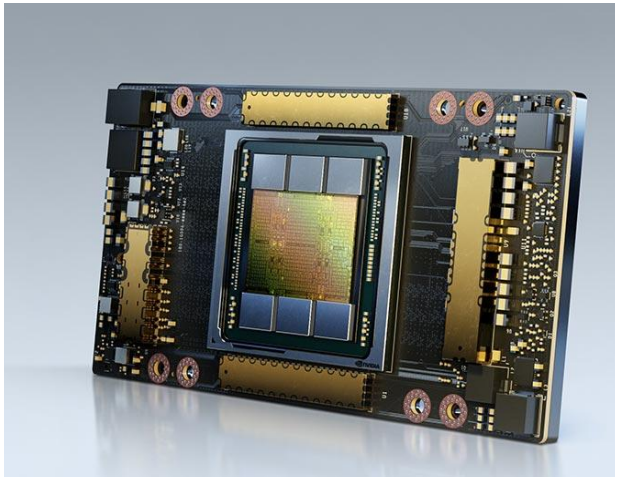


GPUs for ML

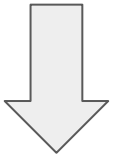
Serafina Di Gioia
PostDoctoral Researcher @ ICTP

What is ML?

in 1960

use and development of computer systems that are able to learn and adapt without following explicit instructions

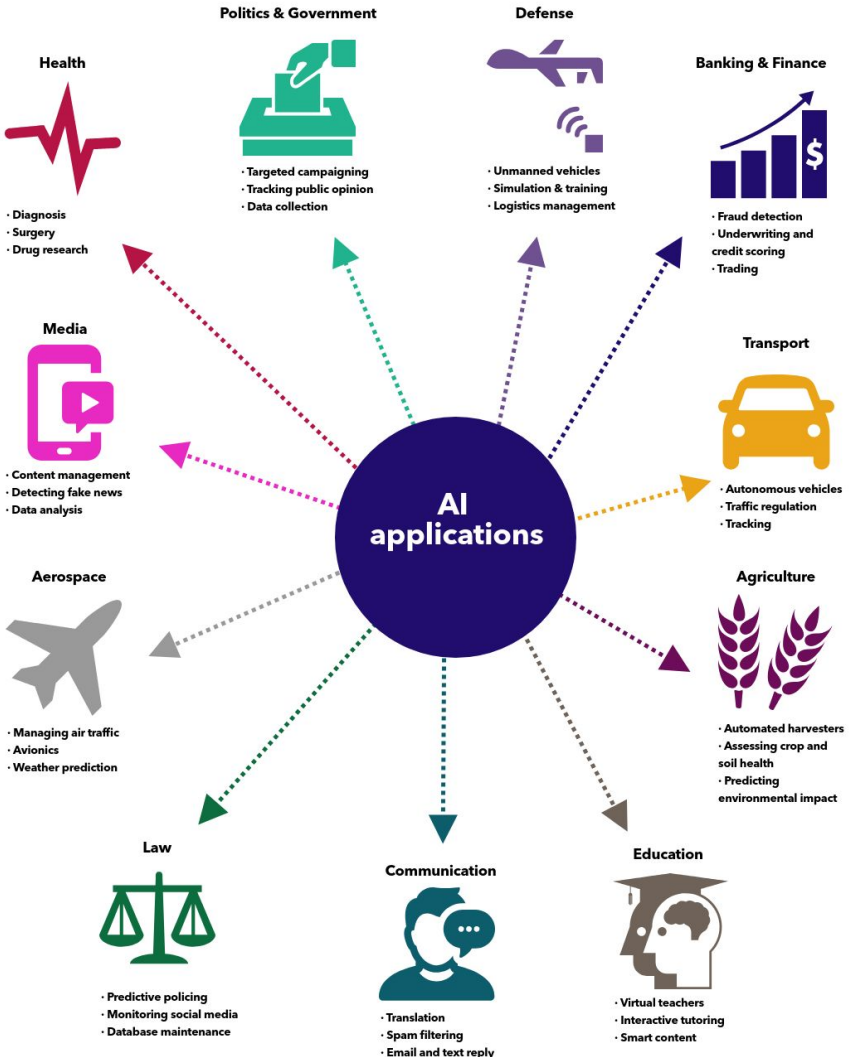
shift in perspective



in 2023

subfield of AI that uses algorithms trained on data to produce adaptable models that can perform a variety of complex tasks (e.g. classification and forecasting) without the need to code specific instructions for different tasks

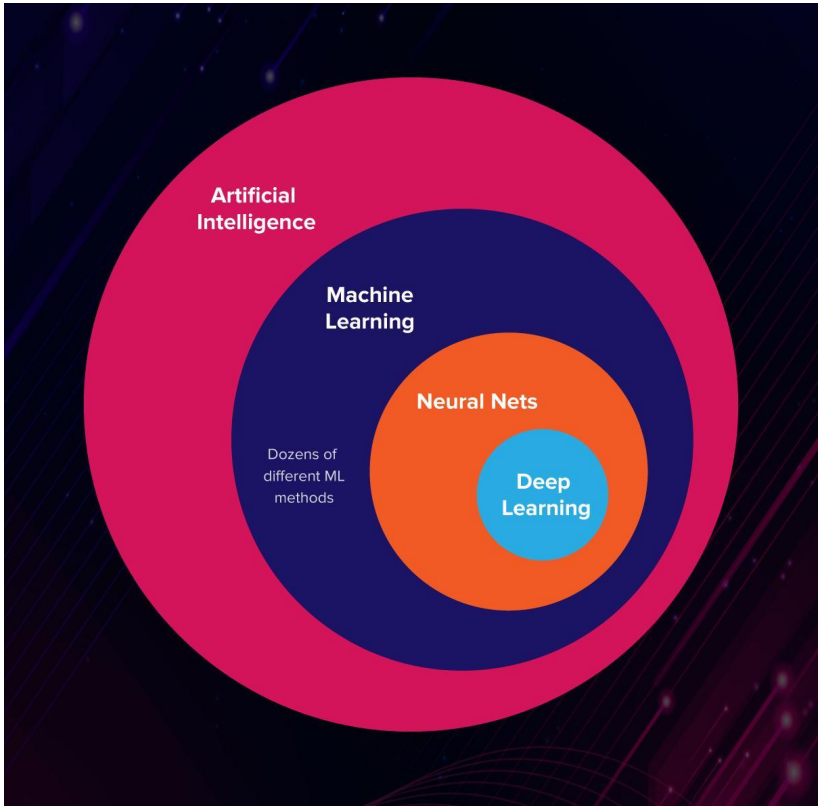
What kind of task are solved by AI systems today?



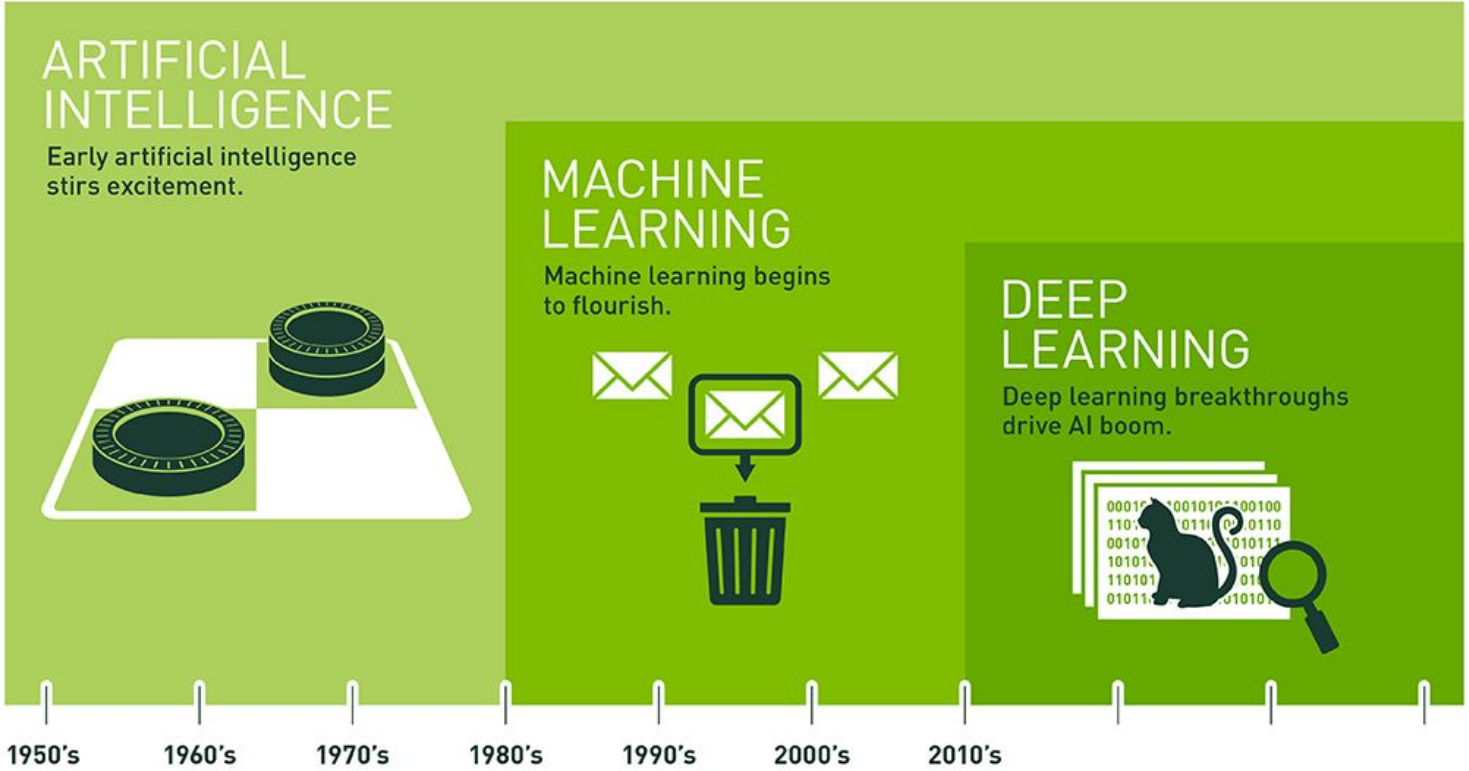
Where is DL in the picture?

Deep learning:

a type of machine learning based on artificial neural networks in which multiple layers of processing are used to extract progressively higher level features from data.



From LISP to the DL revolution…

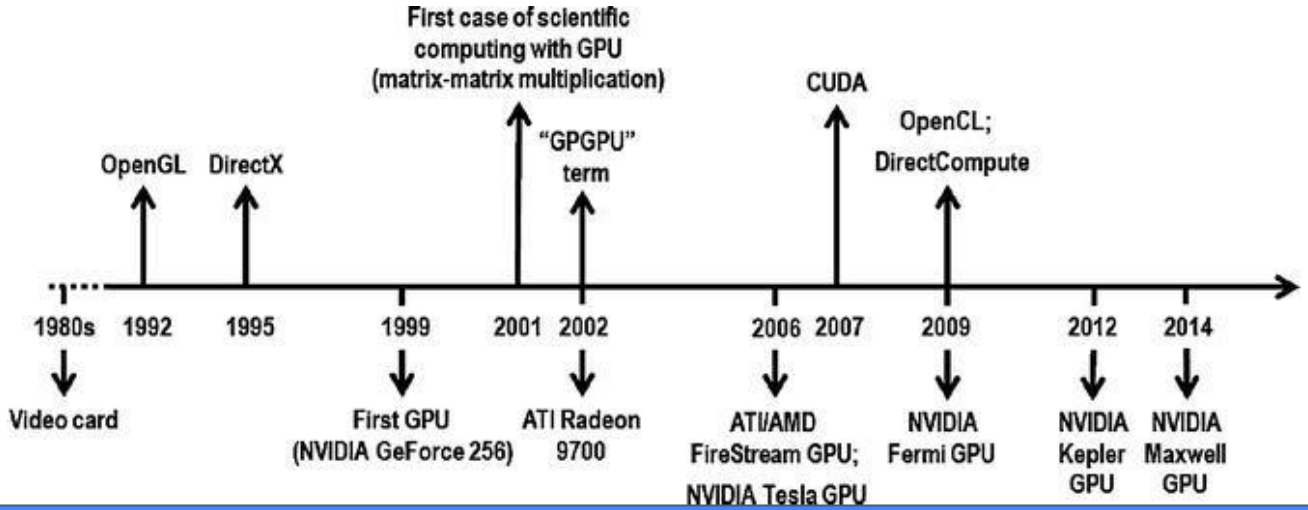


Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

Main ingredients for DL breakthrough

- large datasets available (e.g IMAGENET)
- GPUs development (in particular, CUDA introduction)
- increased involvement of developers from CV and scientific communities

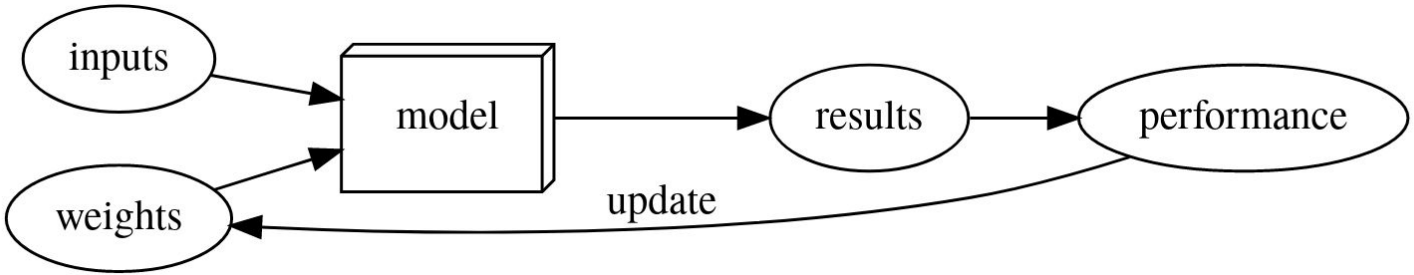
The DL era starts few years after that CUDA came to light



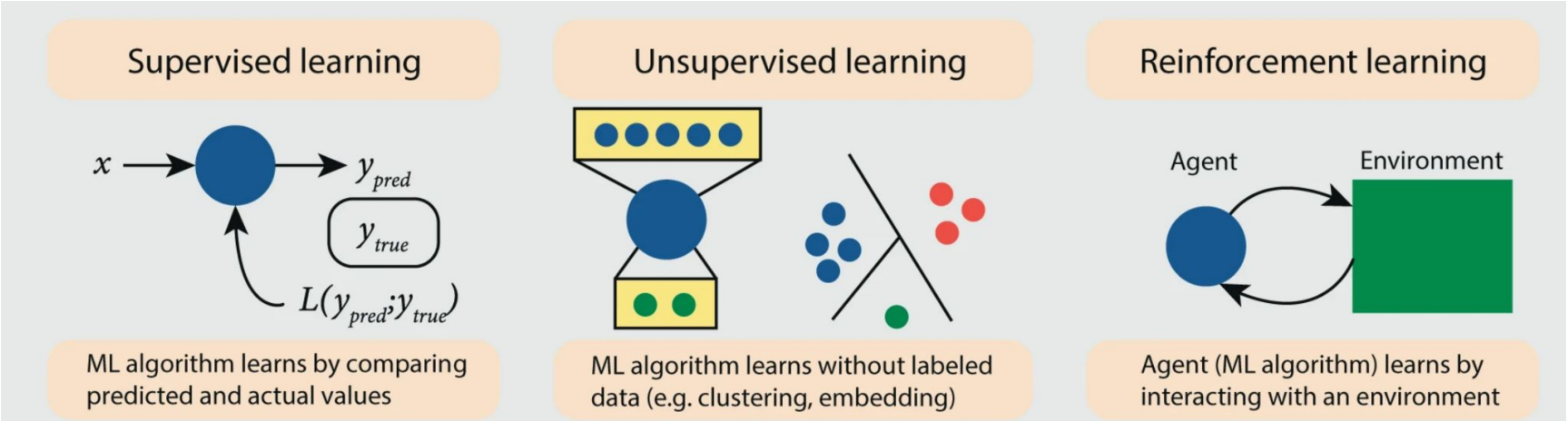
Building blocks of ML algorithms

ML algorithms have three main components

1. **decision process**: based on some input data, which can be labeled or unlabeled, your algorithm will produce an estimate about a pattern in the data. This estimate can be used to solve a prediction or classification task
2. **error function**: it evaluates the prediction of the model. If there are known examples, an error function can make a comparison to assess the accuracy of the model.
3. **Model Optimization Process**: If the model can fit better to the data points in the training set, then weights are adjusted to reduce the discrepancy between the known example and the model estimate. The algorithm will repeat this “evaluate and optimize” process, updating weights autonomously until a threshold of accuracy has been met.

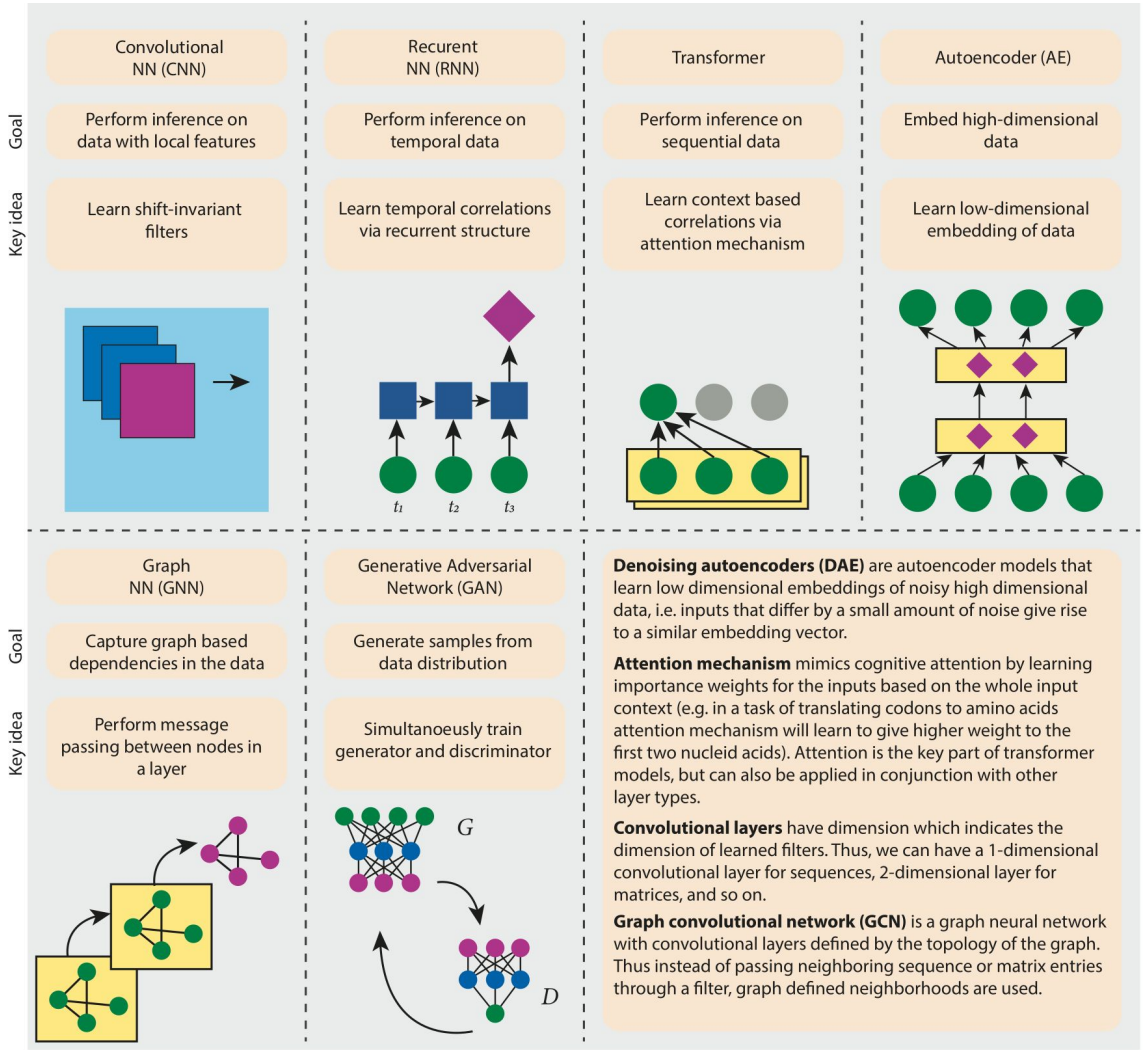


ML scenarios



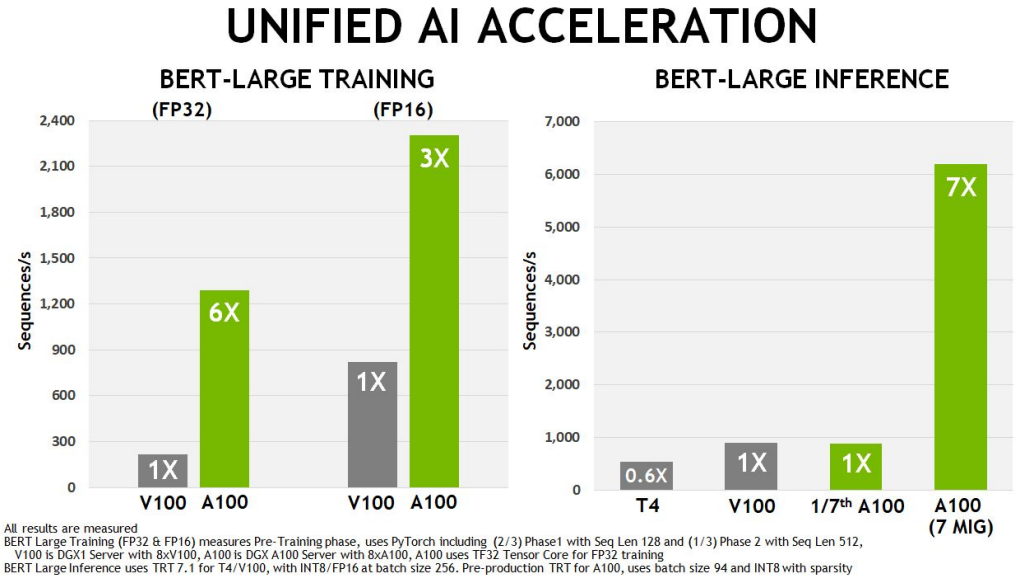
Learning type	Model building	Examples
Supervised	Algorithms or models learn from labeled data (task-driven approach)	Classification, regression
Unsupervised	Algorithms or models learn from unlabeled data (Data-Driven Approach)	Clustering, associations, dimensionality reduction
Semi-supervised	Models are built using combined data (labeled + unlabeled)	Classification, clustering
Reinforcement	Models are based on reward or penalty (environment-driven approach)	Classification, control

Diving into the world of DL models

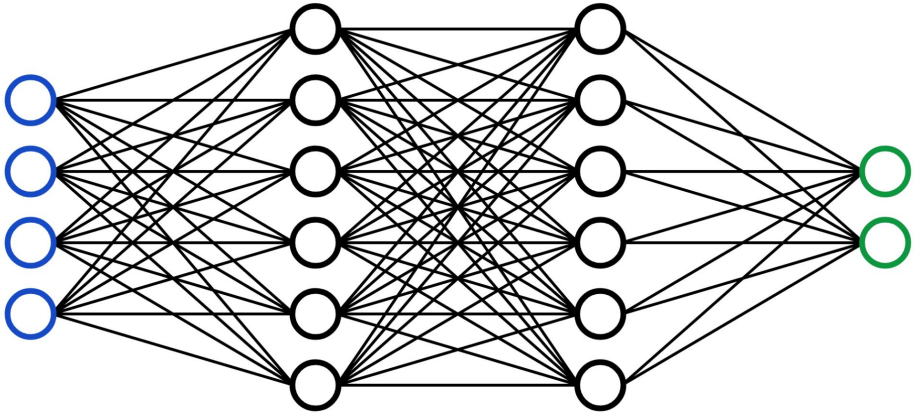


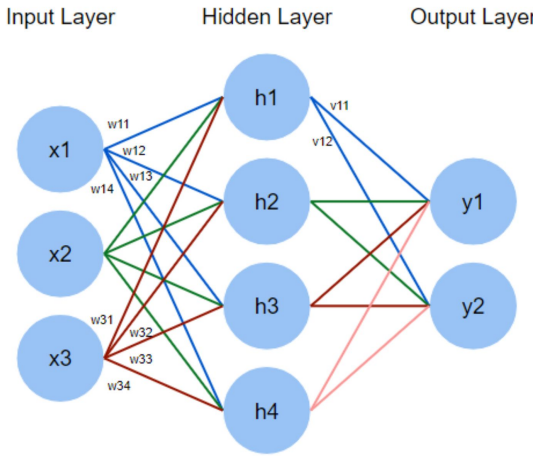
Incredible performance increase of NVIDIA

NVIDIA Accelerator Specification Comparison			
	A100	V100	P100
FP32 CUDA Cores	6912	5120	3584
Boost Clock	~1.41GHz	1530MHz	1480MHz
Memory Clock	2.4Gbps HBM2	1.75Gbps HBM2	1.4Gbps HBM2
Memory Bus Width	5120-bit	4096-bit	4096-bit
Memory Bandwidth	1.6TB/sec	900GB/sec	720GB/sec
VRAM	40GB	16GB/32GB	16GB
Single Precision	19.5 TFLOPs	15.7 TFLOPs	10.6 TFLOPs
Double Precision	9.7 TFLOPs (1/2 FP32 rate)	7.8 TFLOPs (1/2 FP32 rate)	5.3 TFLOPs (1/2 FP32 rate)
INT8 Tensor	624 TOPs	N/A	N/A
FP16 Tensor	312 TFLOPs	125 TFLOPs	N/A
TF32 Tensor	156 TFLOPs	N/A	N/A
Interconnect	NVLink 3 12 Links (600GB/sec)	NVLink 2 6 Links (300GB/sec)	NVLink 1 4 Links (160GB/sec)
GPU	GA100 (826mm2)	GV100 (815mm2)	GP100 (610mm2)
Transistor Count	54.2B	21.1B	15.3B
TDP	400W	300W/350W	300W
Manufacturing Process	TSMC 7N	TSMC 12nm FFN	TSMC 16nm FinFET
Interface	SXM4	SXM2/SXM3	SXM
Architecture	Ampere	Volta	Pascal



Linear Neural Network (lNN)

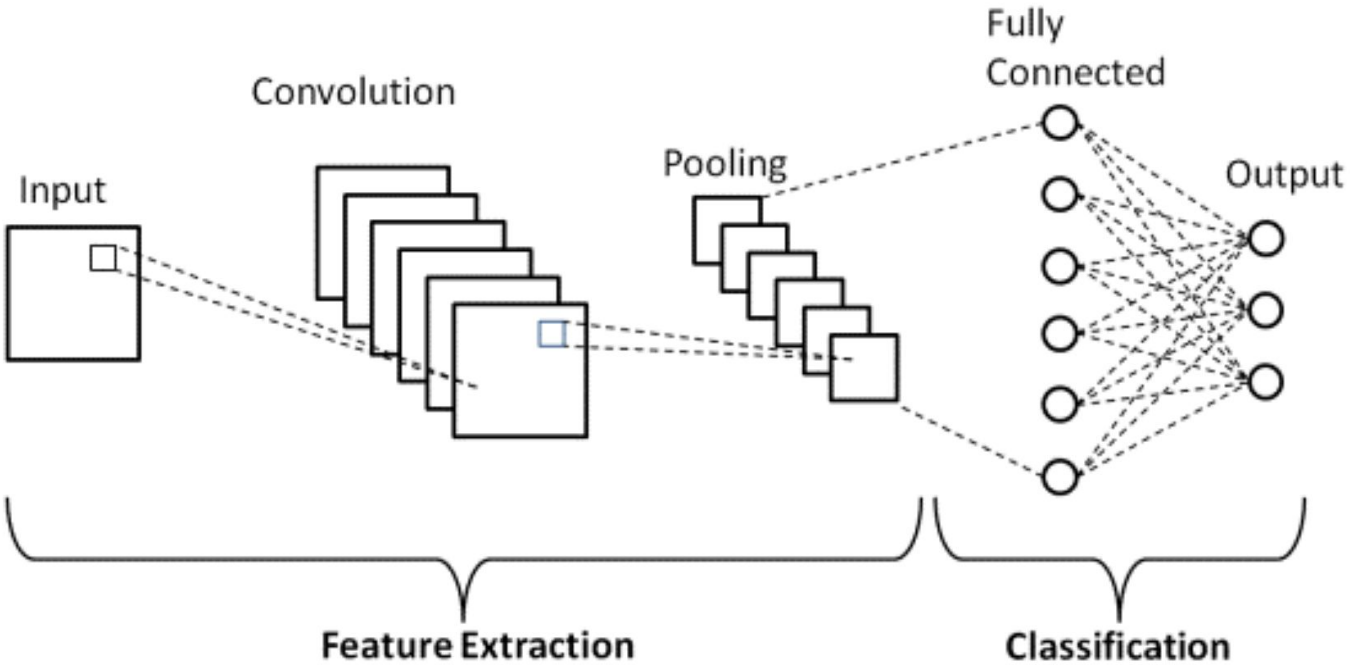




$$[\,x_1 \quad x_2 \quad x_3\,] * \begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} & w_{1,4} \\ w_{2,1} & w_{2,2} & w_{2,3} & w_{2,4} \\ w_{3,1} & w_{3,2} & w_{3,3} & w_{3,4} \end{bmatrix} = [\,h'_1 \quad h'_2 \quad h'_3 \quad h'_4\,]$$

Alias of nn.linear() in Pytorch: torch.mm(inputs, linear.weight.T).add(linear.bias)

DL for image classification: Convolutional Neural Networks (CNN)



Everything in DL turns to be a matrix multiplication at the end…

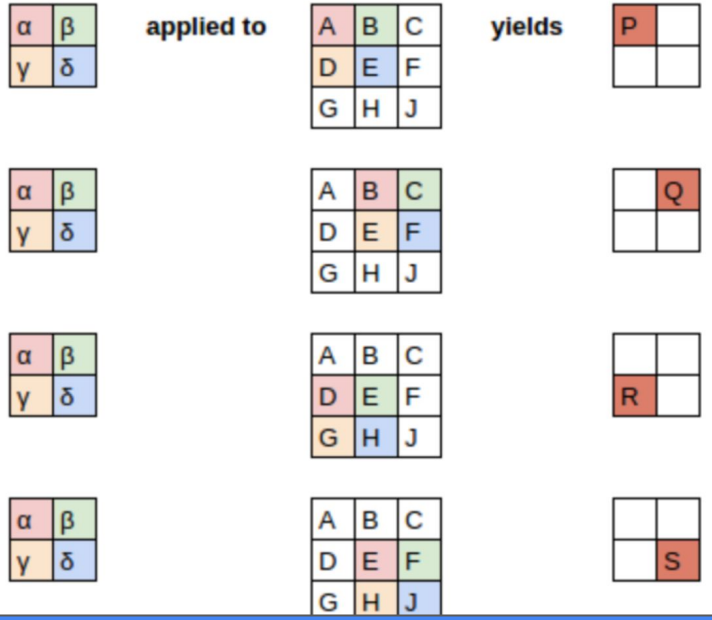
Let’s take a look at basic element of CNN: convolution layer

Consider the case where we are applying (2,2) kernel

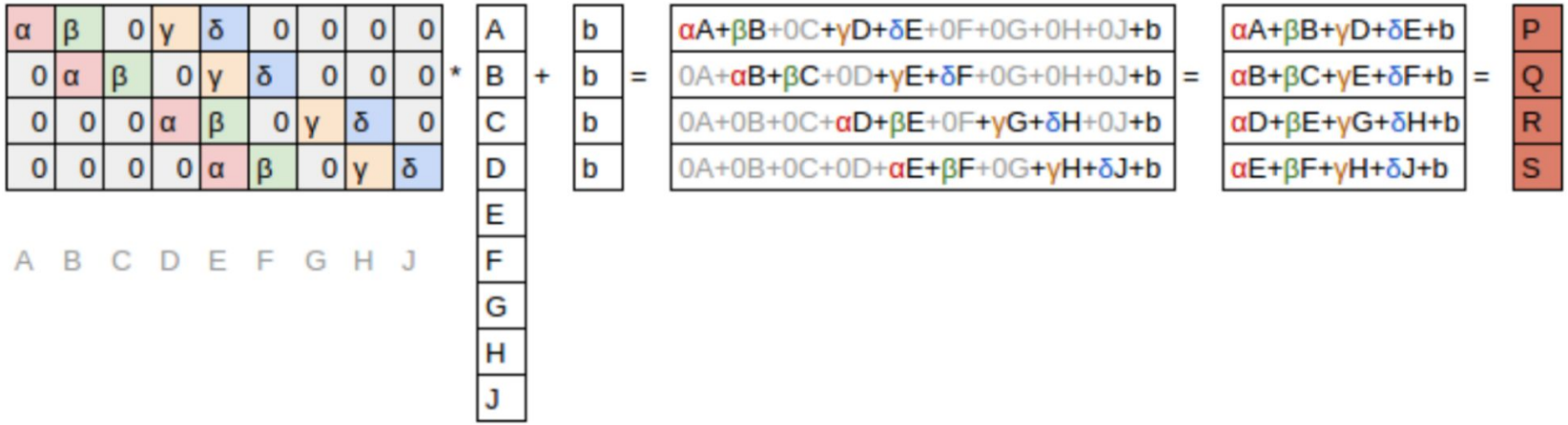
α	β
γ	δ

to a (3,3) matrix:

A	B	C
D	E	F
G	H	J



The convolution can be rewritten as



Why GPUs for DL?

1) **Neural networks are embarrassingly parallel algorithm**
2) **most of the operations performed in DL models can be rewritten as matrix multiplications**
3) **big datasets require to perform big matrix computation (**extremely slow on CPU with respect to GPU**)**
4) **well established libraries, with specific classes for ML objects (e.g. cuDNN, more recently tensorRT)**

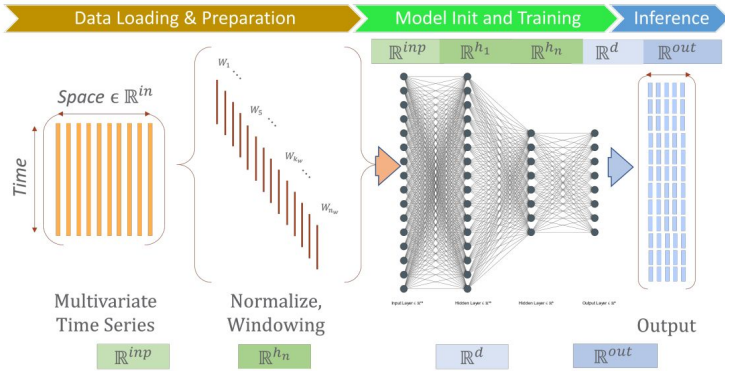
and we know that GPUs are very good in solving specific parallel tasks (e.g matrix multiplication) , thanks to

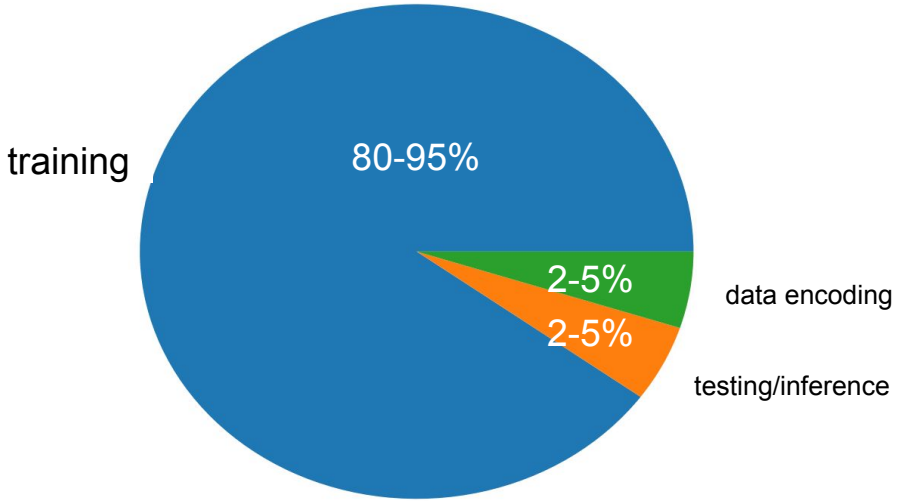
- +1000 cores (>100K threads)
- **SIMD / SIMT**
- **high memory bandwidth**
- **newer GPUs have also tensor cores (particularly suited to tensor ops typical of NNs), and mixed precision**

However, also GPUs have limitations:

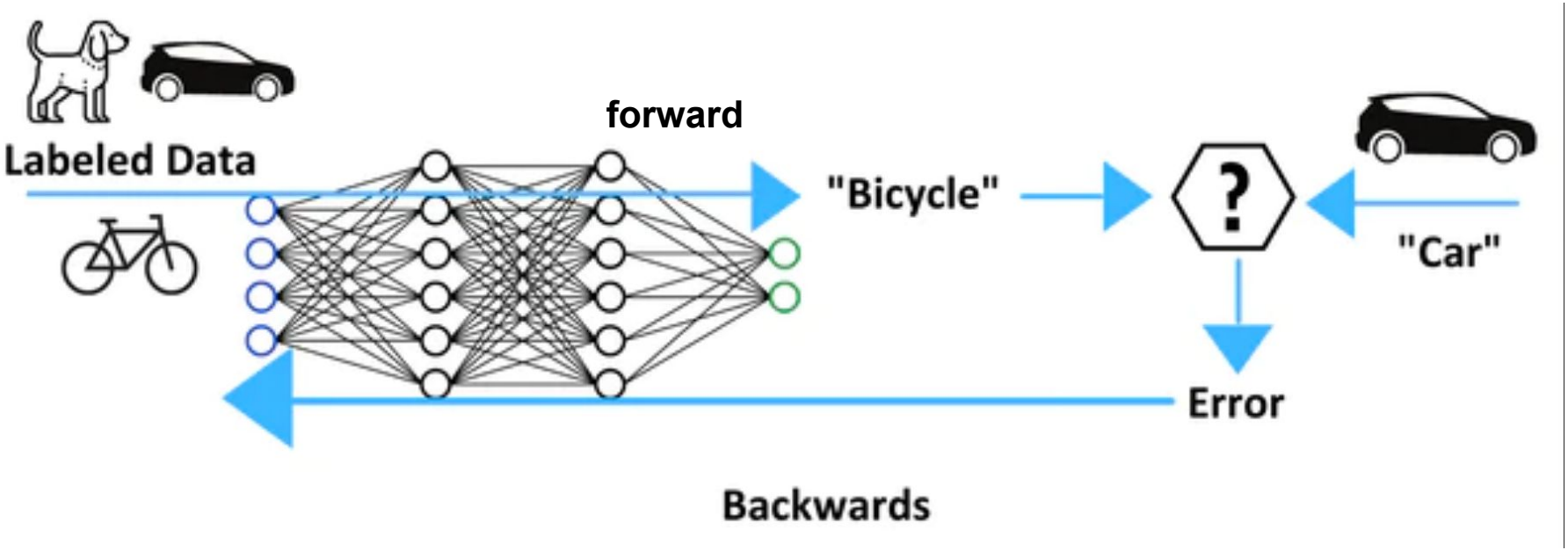
- GPUs might not be as efficient for extreme sparse networks, due to the overhead of managing sparse data structures.
- Some specialized sparse operations might not be as optimized as dense operations on GPUs.

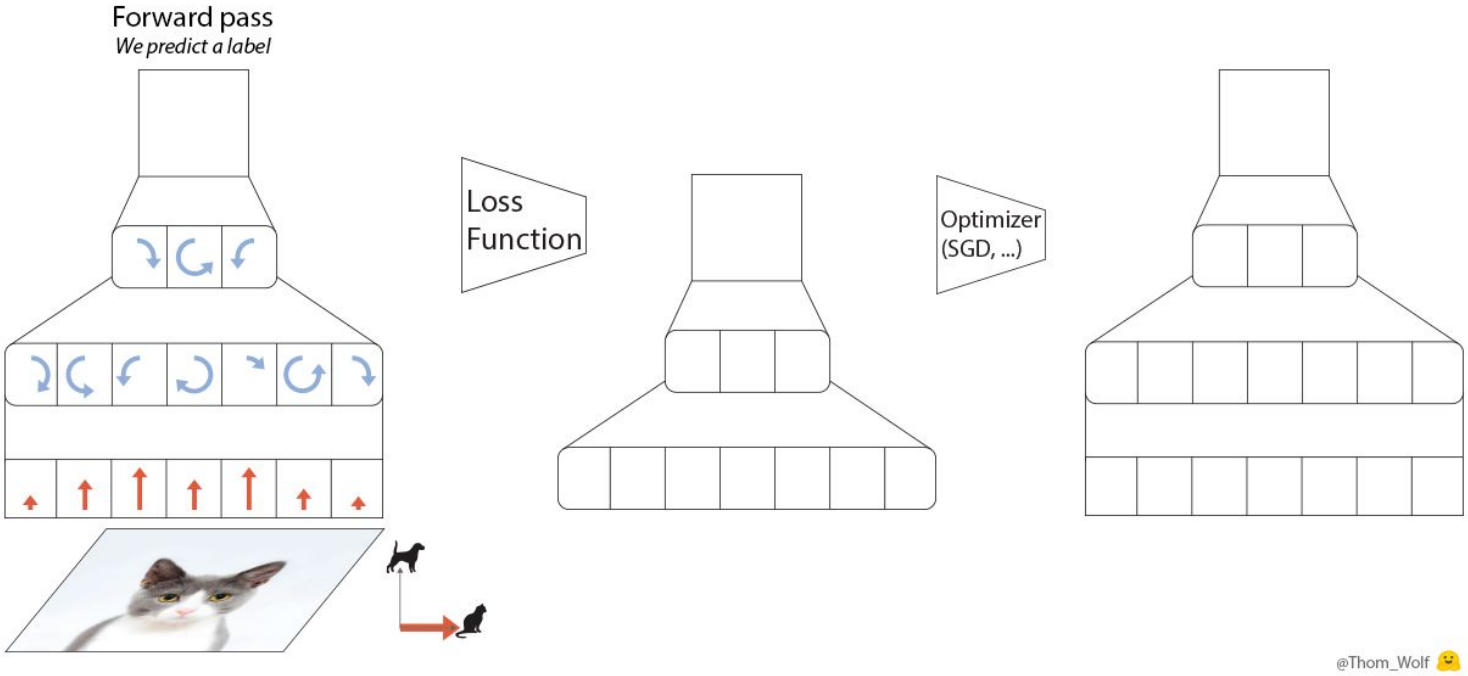
Analyzing the computational workload of DL models



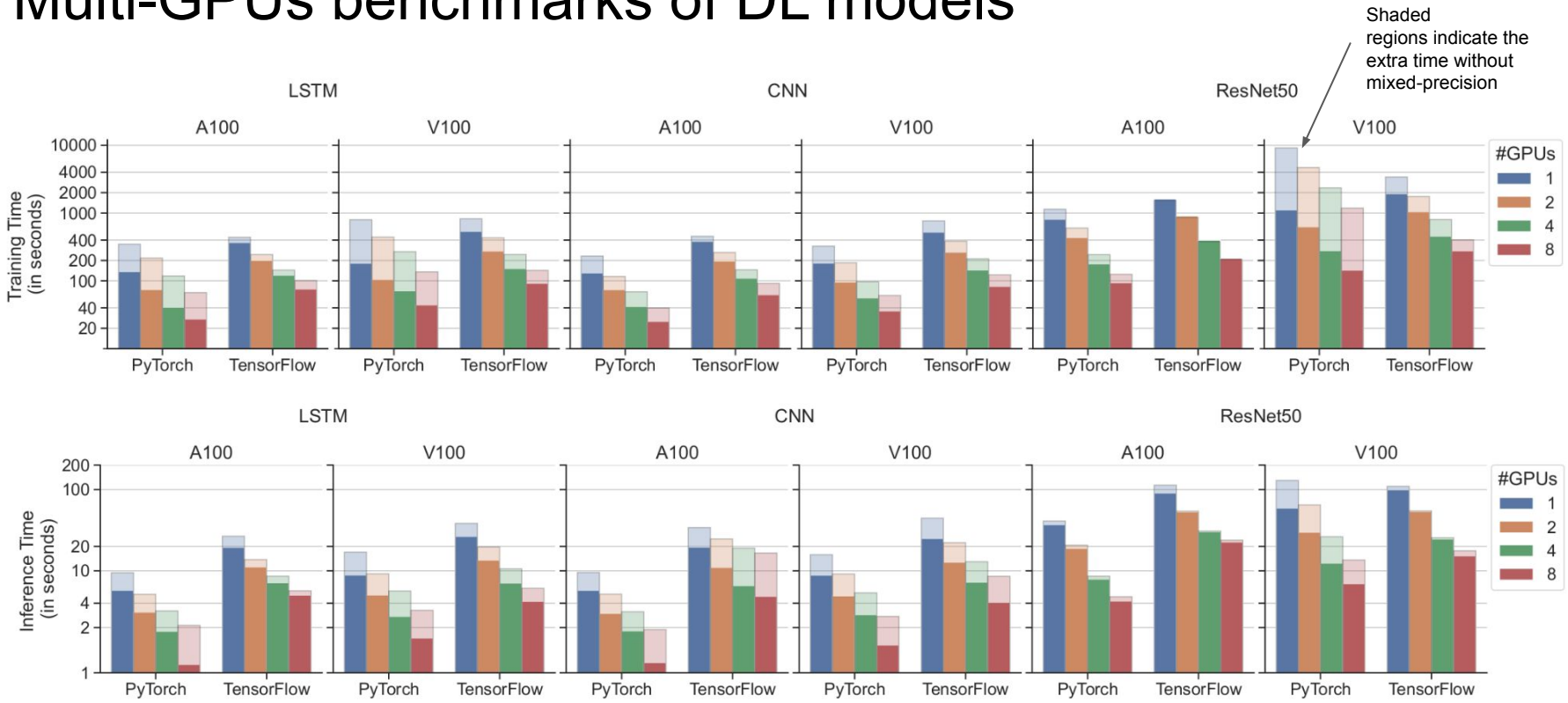


Why training is so much compute-intensive?




Forward pass
We predict a label
Loss Function
Optimizer (SGD, ...)
@Thom_Wolf

Multi-GPUs benchmarks of DL models



Different strategies for Multi-GPUs training

we can identify 5 different categories of parallelism

model parallelism

tensor parallelism

data parallelism

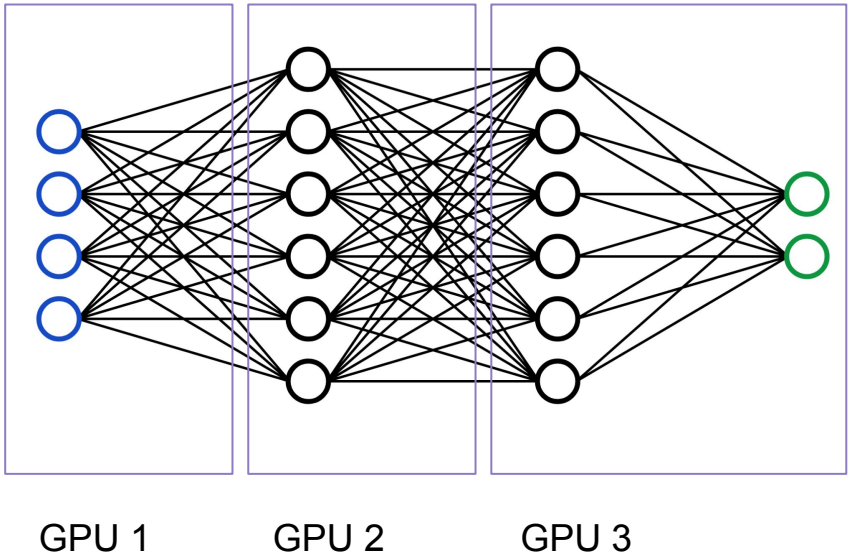
sequence parallelism

pipeline parallelism

Model parallelism

In this parallelism framework we choose to put different layers of the NN on different GPUs

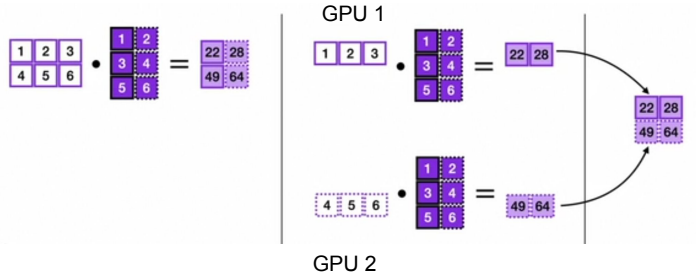
to work around GPU memory limits

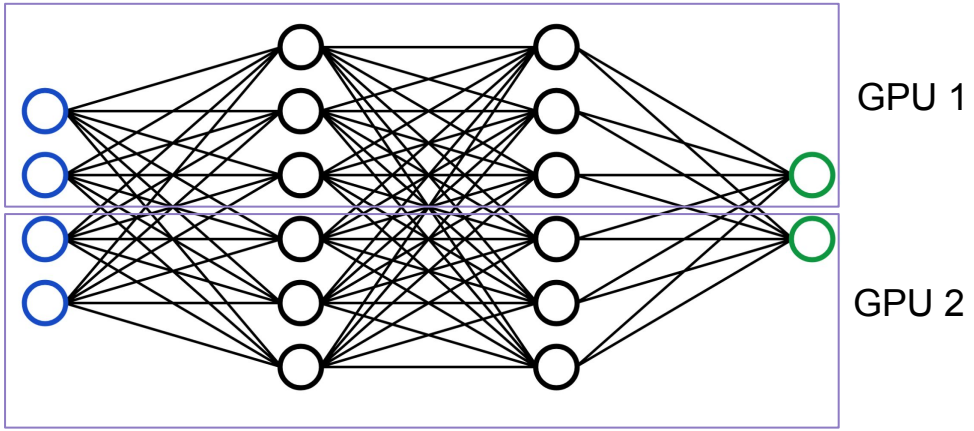


Tensor parallelism

In this framework we split the tensor operation done at each layer among different GPUs

similarly to what we would have done for matmul

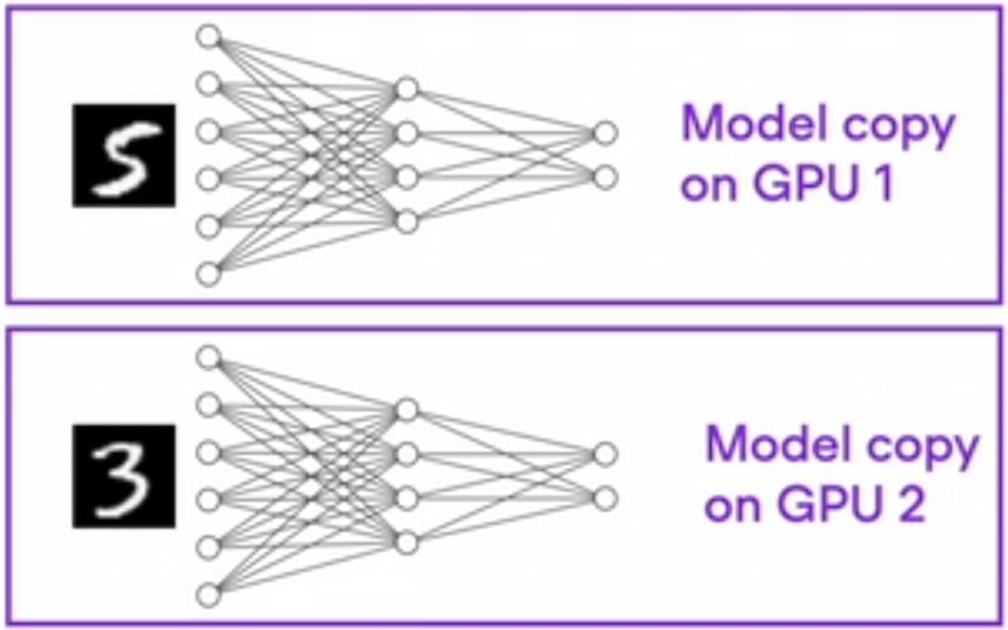




Data Parallelism

In this framework we split batches to train DL model

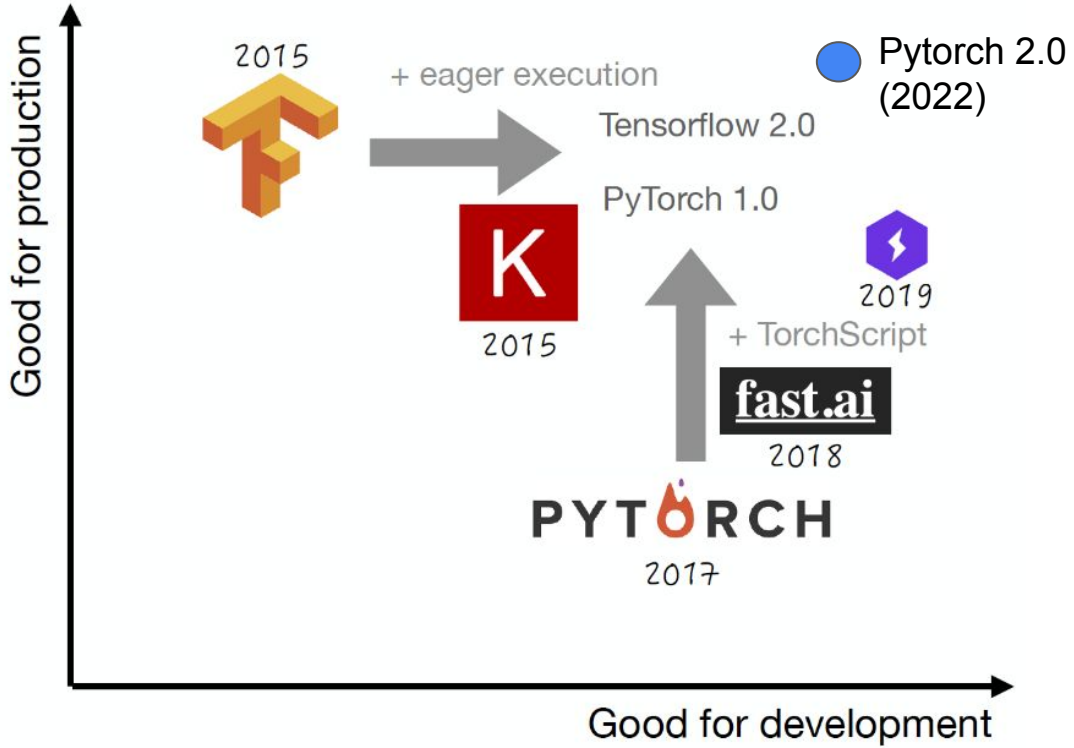
into different GPUs



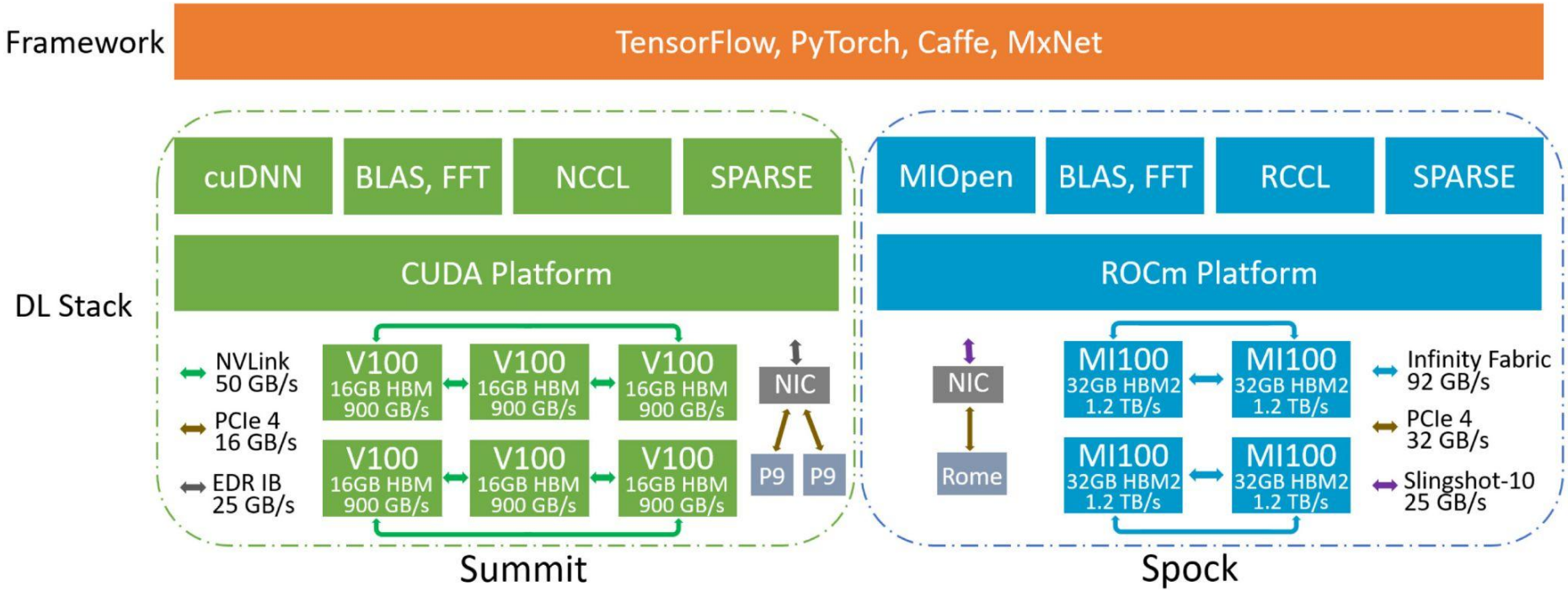
More complex strategies for DL training

Sequence parallelism and pipeline parallelism frameworks are obtained combining the previous approaches, and are typically applied to DL models dealing with spatio-temporal data.

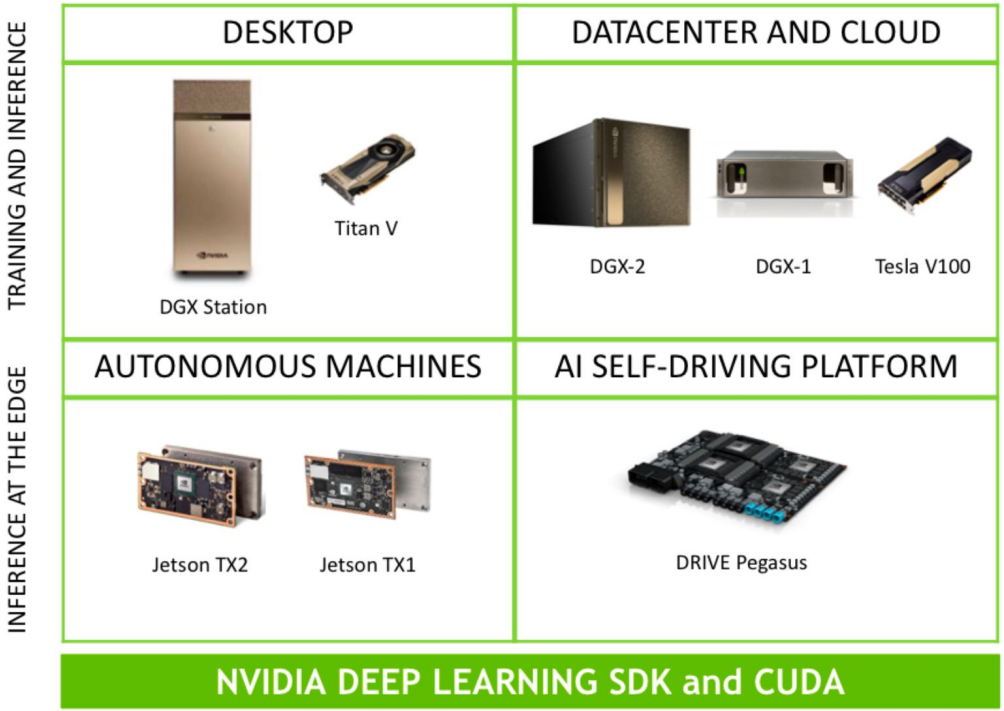
Python most used libraries/frameworks for DL



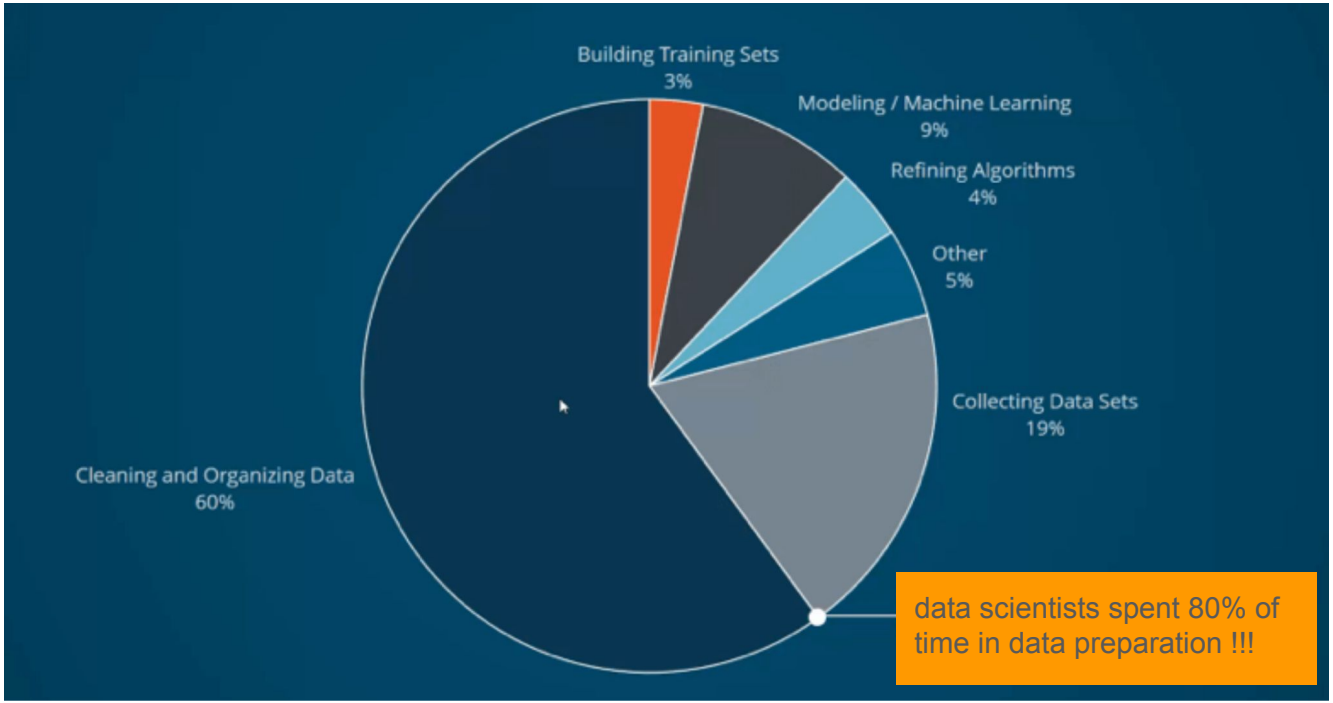
What's behind Pytorch/Tensorflow?



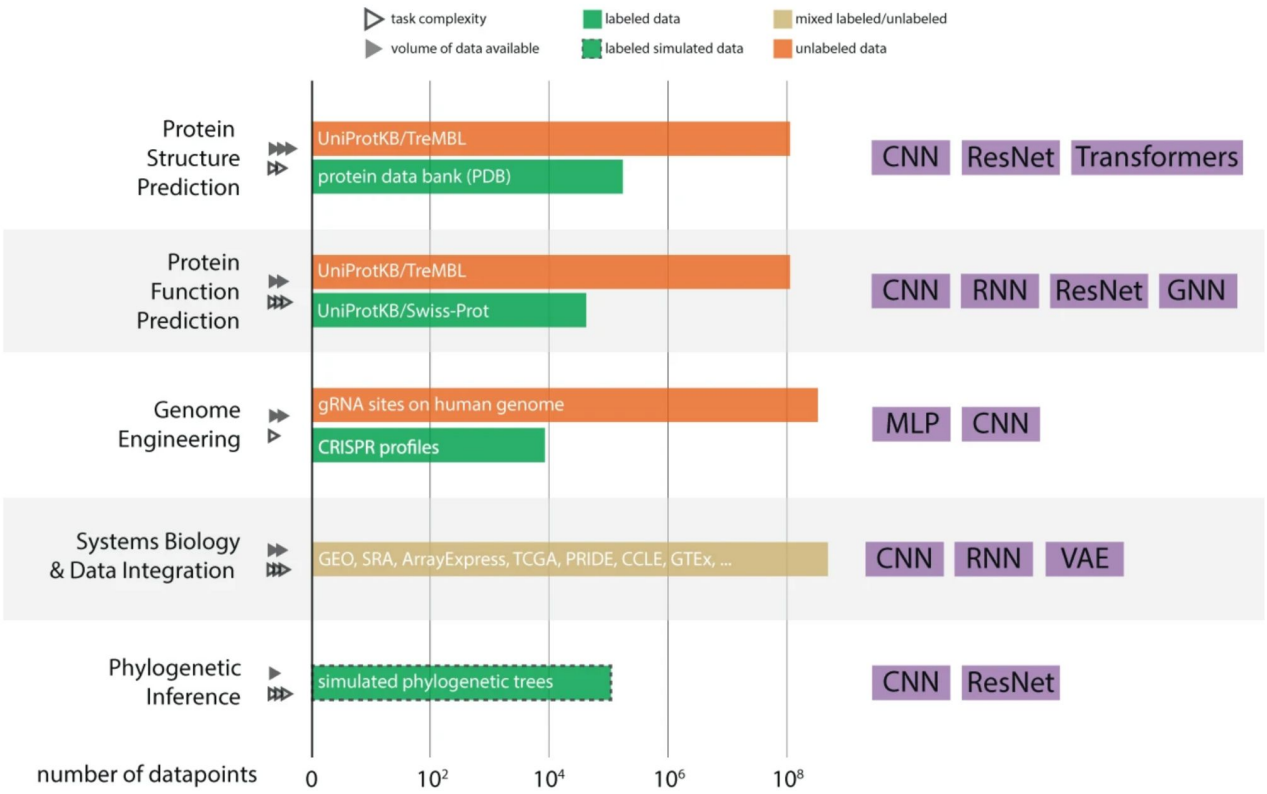
Latest NVIDIA developments (HW/SW)



Do we need GPUs also for other ML tasks?

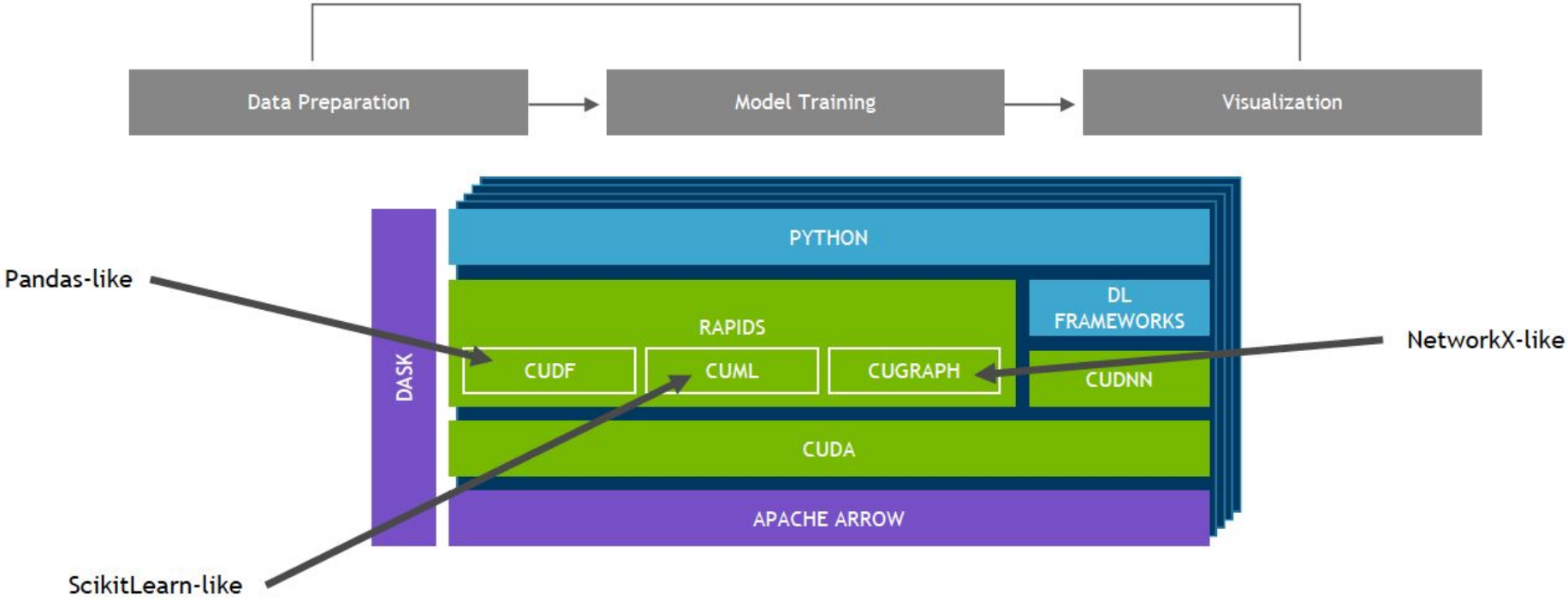


Typical sizes of DL data sets

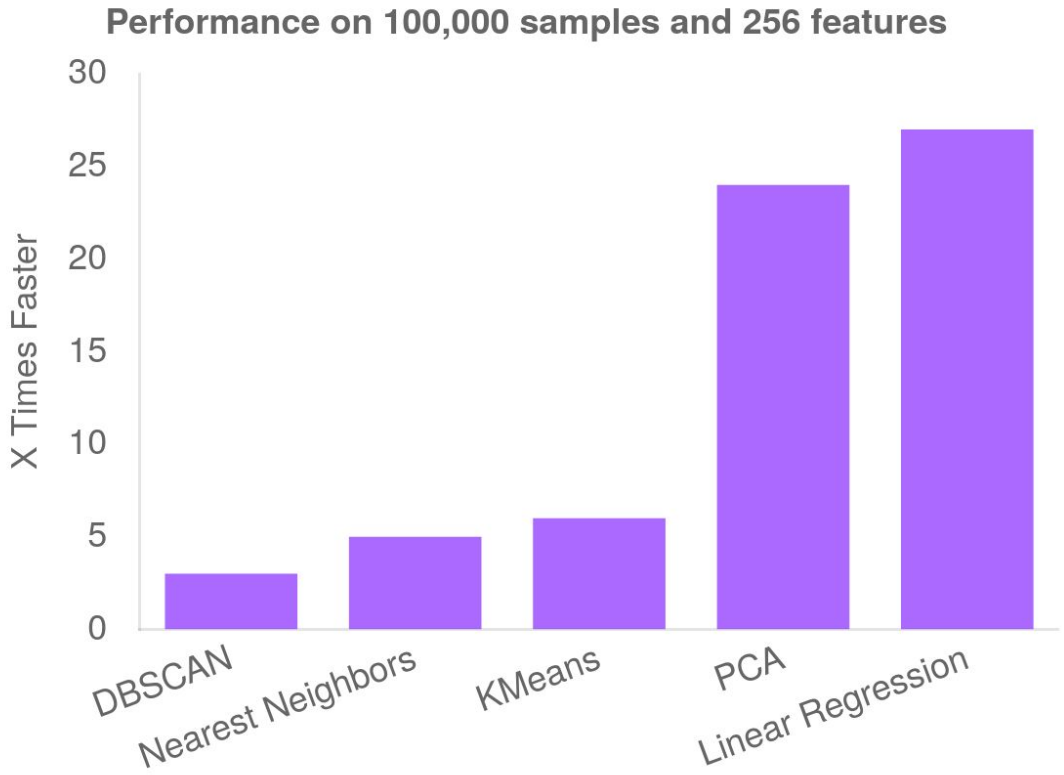


The increasing complexity of the new datasets, typical of big data epoch motivates the need for GPU-based libraries for feature analysis and data preprocessing

GPU-based libraries outside of Pytorch

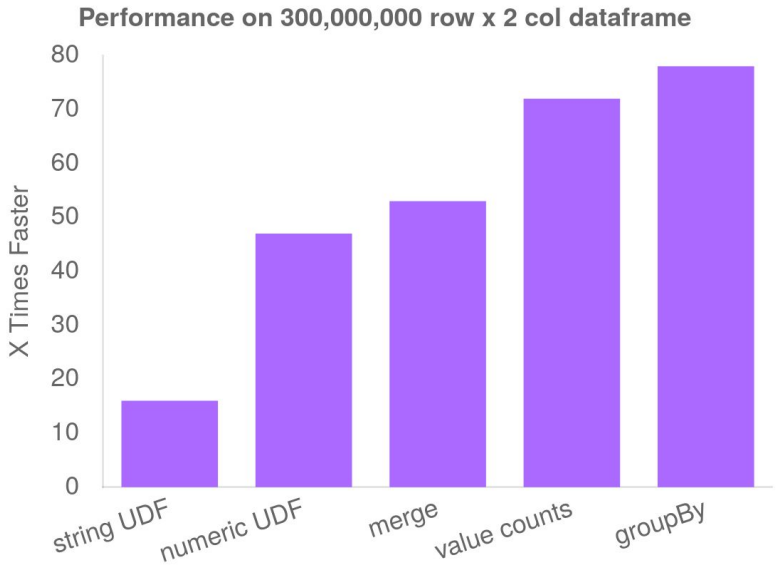


GPU for classical ML



* Benchmark on AMD EPYC 7642 (using 1x 2.3GHz CPU core) w/ 512GB and NVIDIA A100 80GB (1x GPU) w/ scikit-learn v1.2 and cuML v23.02

GPUs for data preprocessing



* Benchmark on AMD EPYC 7642 (using 1x 2.3GHz CPU core) w/ 512GB and NVIDIA A100 80GB (1x GPU) w/ pandas v1.5 and cuDF v23.02

References

Milan Jain, Sayan Ghosh, Sai Pushpak Nandanoori, 2022, *Workload Characterization of a Time-Series Prediction System for Spatio-Temporal Data*

Junqi Yin et. al, 2021, *Comparative evaluation of deep learning workloads for leadership-class systems*

NVIDIA Booklet on GPU development, 2021

Deep Learning: A Comprehensive Overview on Techniques, Taxonomy, Applications and Research Directions