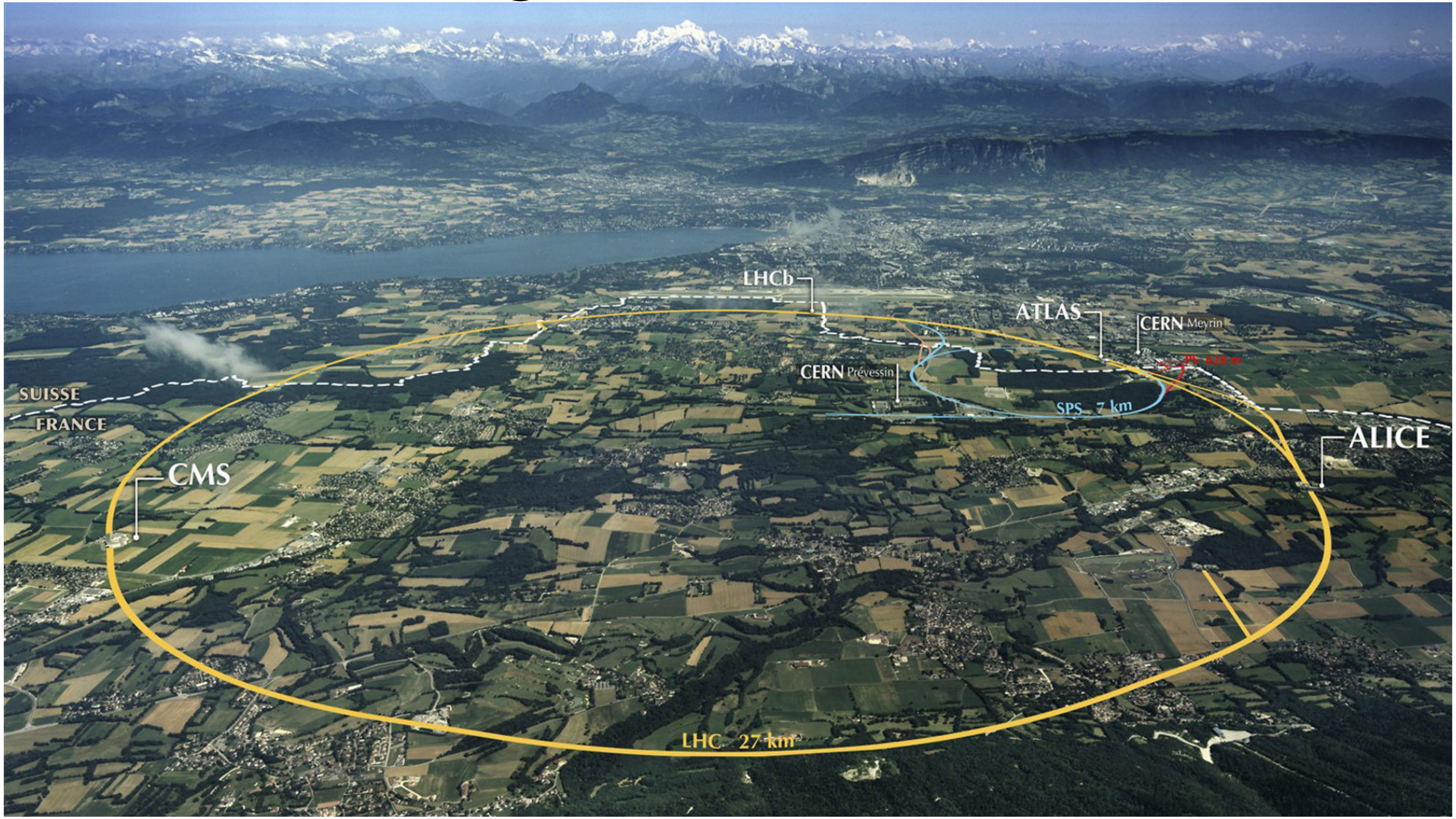
Real-time AI for Science

Mia Liu

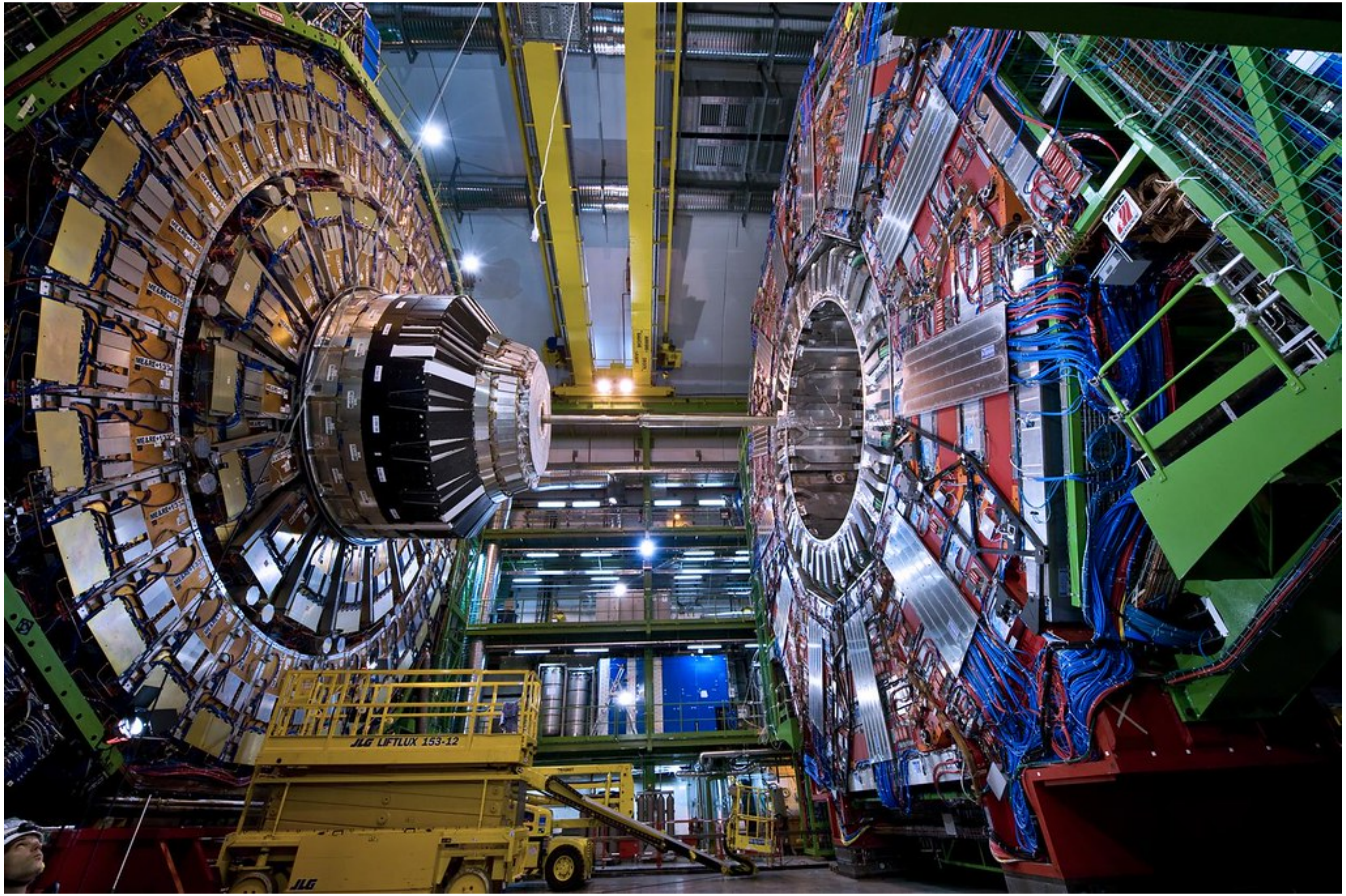
COFI Advanced Instrumentation and Data analysis School

Dec.2023

The Large Hadron Collider



The CMS detector

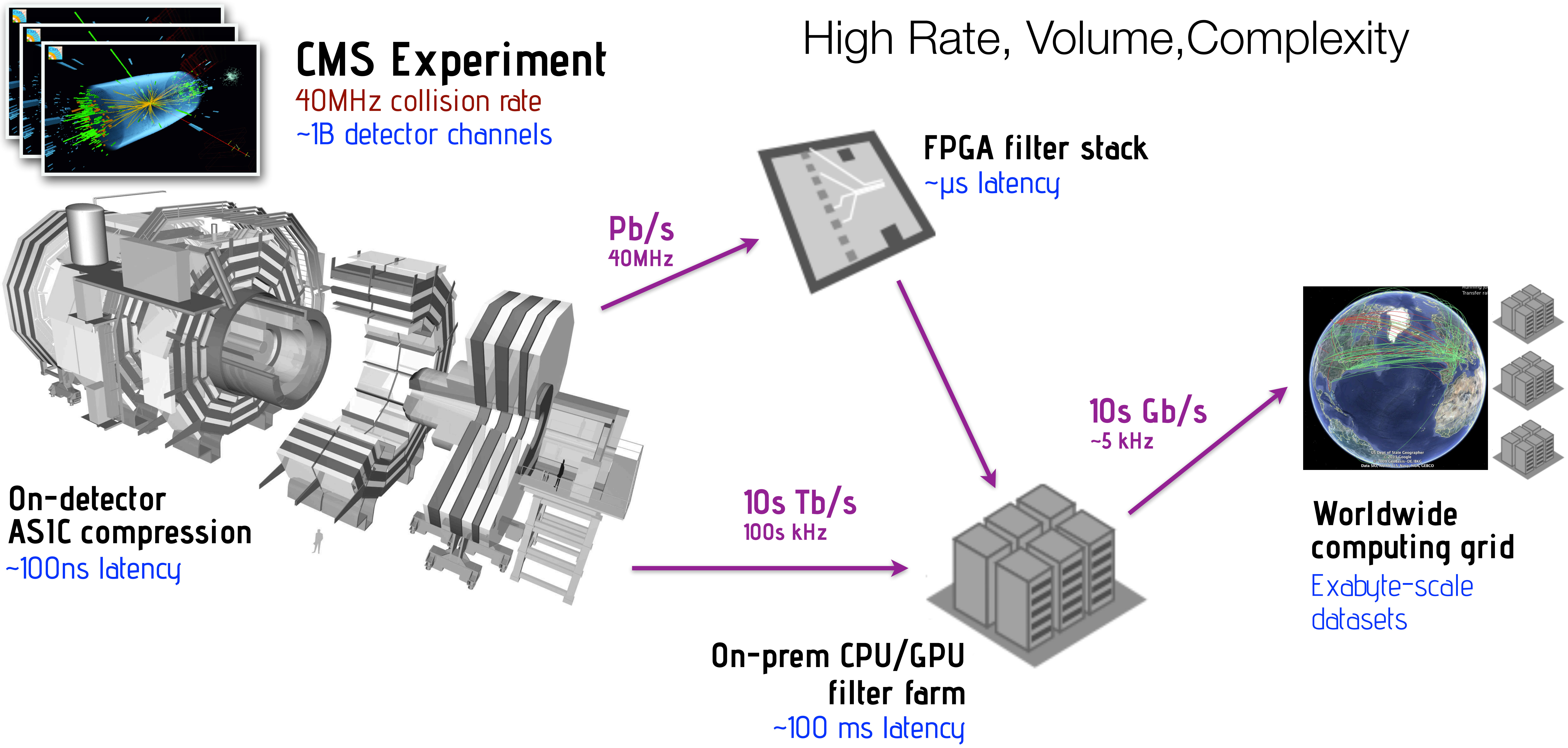


Today’s Lecture

- Real-time system constraints and needs
  - LHC as an example
  - Specialized hardwares: FPGA/ASIC
- Challenges in AI/ML on specialized hardwares
  - Design efficient networks for real-time systems
  - Co-design tools and needs
- Real-time AI for other science domains: quantum system control etc

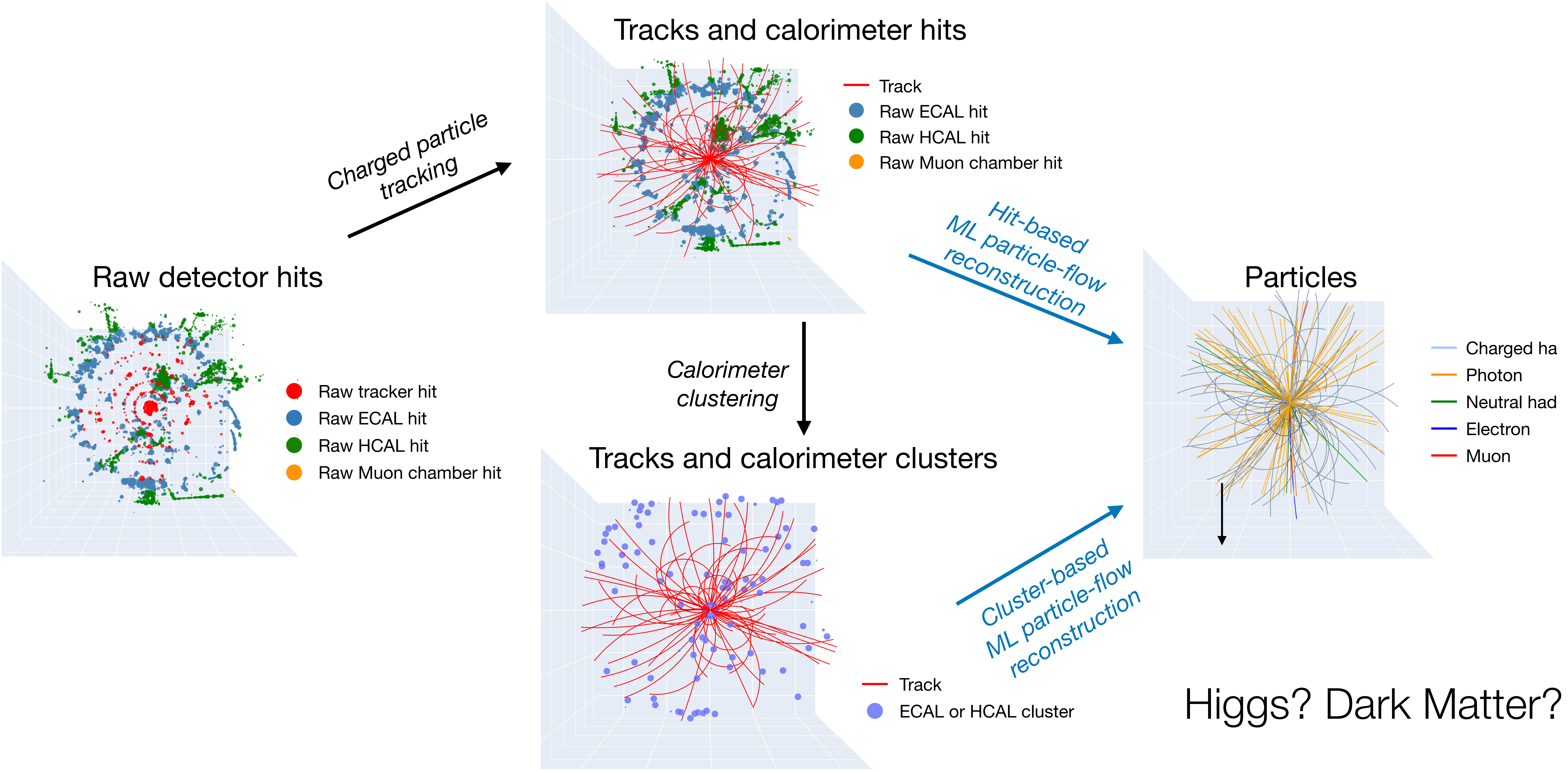
From Collisions to Discoveries

High Rate, Volume,Complexity



Science with Big data: Multi-tier Data Processing

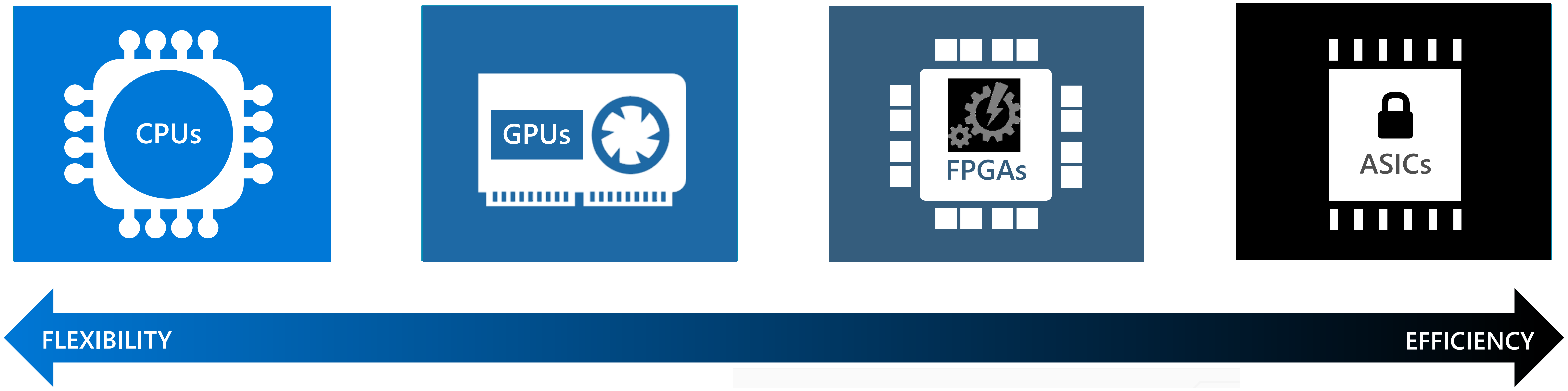
AI/ML: new opportunities for real-time reconstruction



Higgs? Dark Matter?

Learning grouping of detector elements

Hardware Landscape



Offline/HLT

L1 trigger

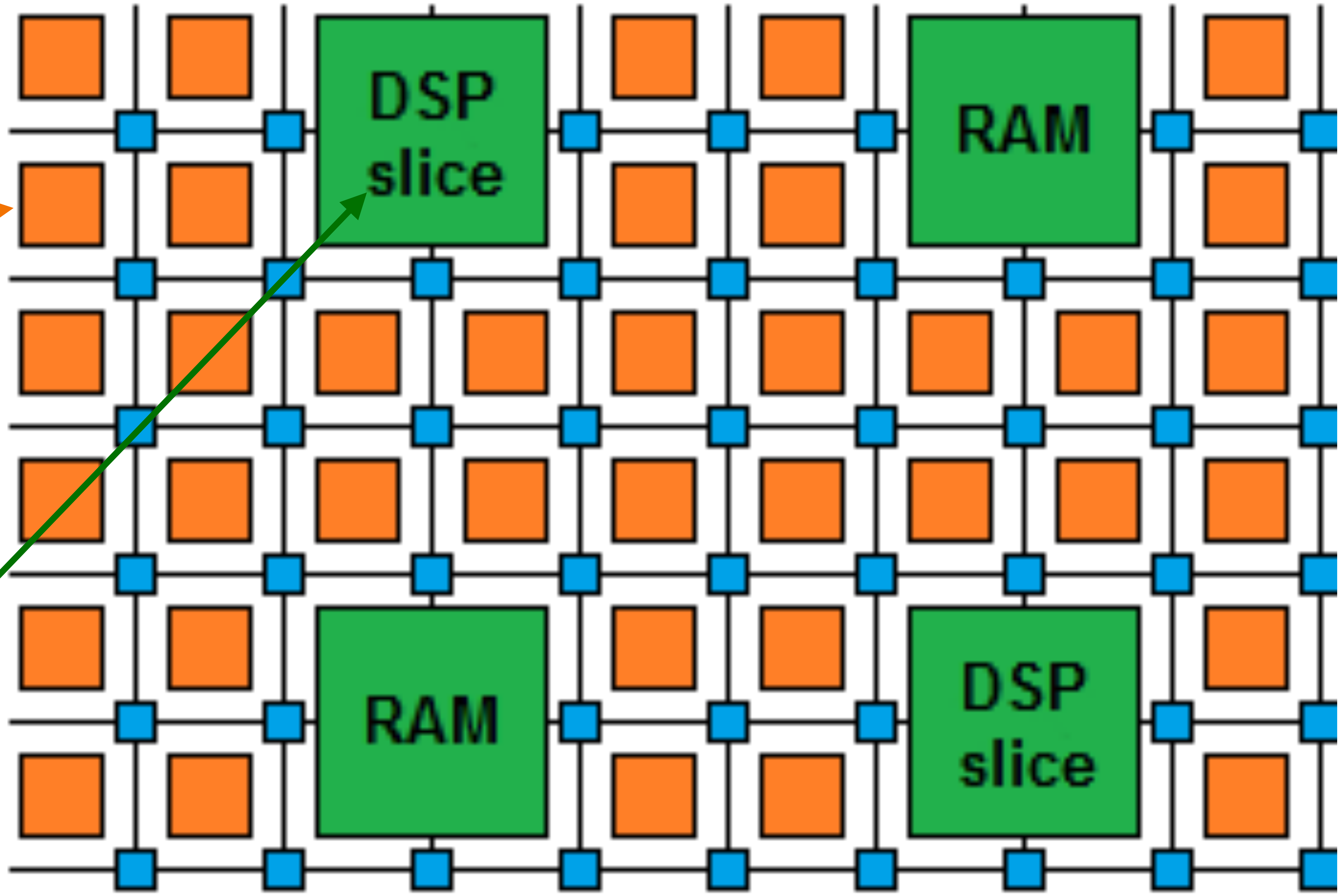
On-detector electronics

What is an FPGA?

What is an FPGA?

Field Programmable Gate Arrays

Logic cell:
Flip-flops (FF) and
look up tables (LUTs)



Digital Signal Processors (DSPs)

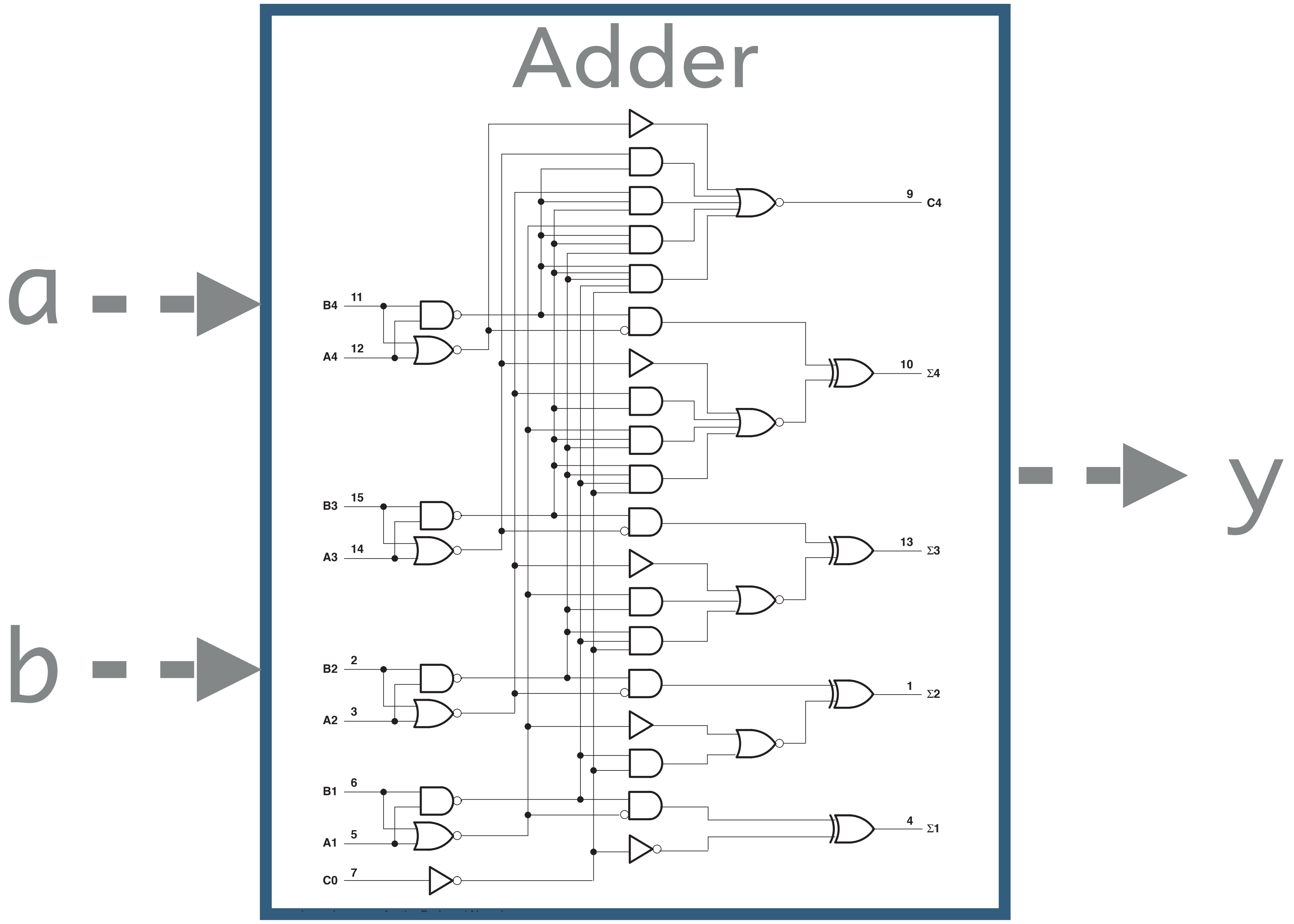
Virtex Ultrascale+ VU9P
6800 DSPs
1M LUTs
2M FFs
75 Mb BRAM

Programming FPGAs

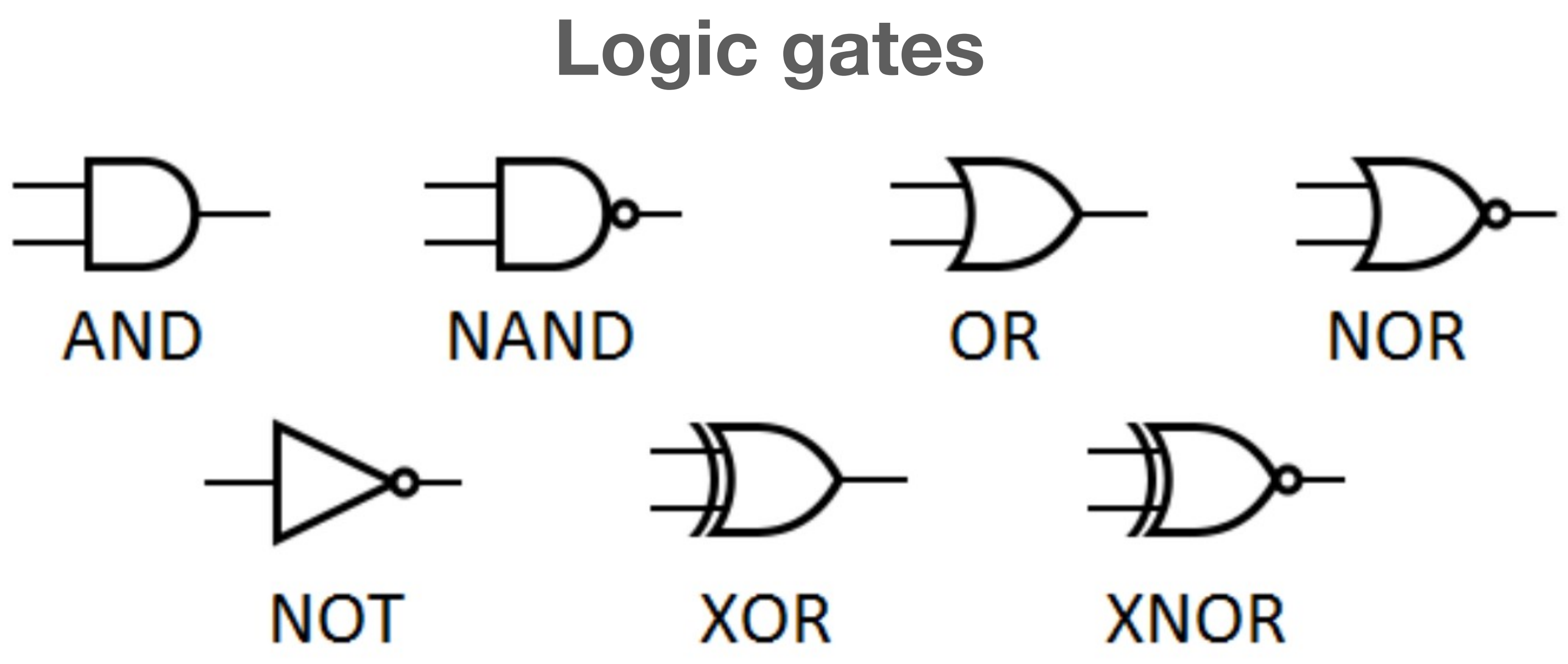
- Say you want to program an "adder" function on an FPGA

```verilog
module adder(
    input  wire [4:0] a,
    input  wire [4:0] b,
    output wire [4:0] y
);
    assign y = a + b;

endmodule
```



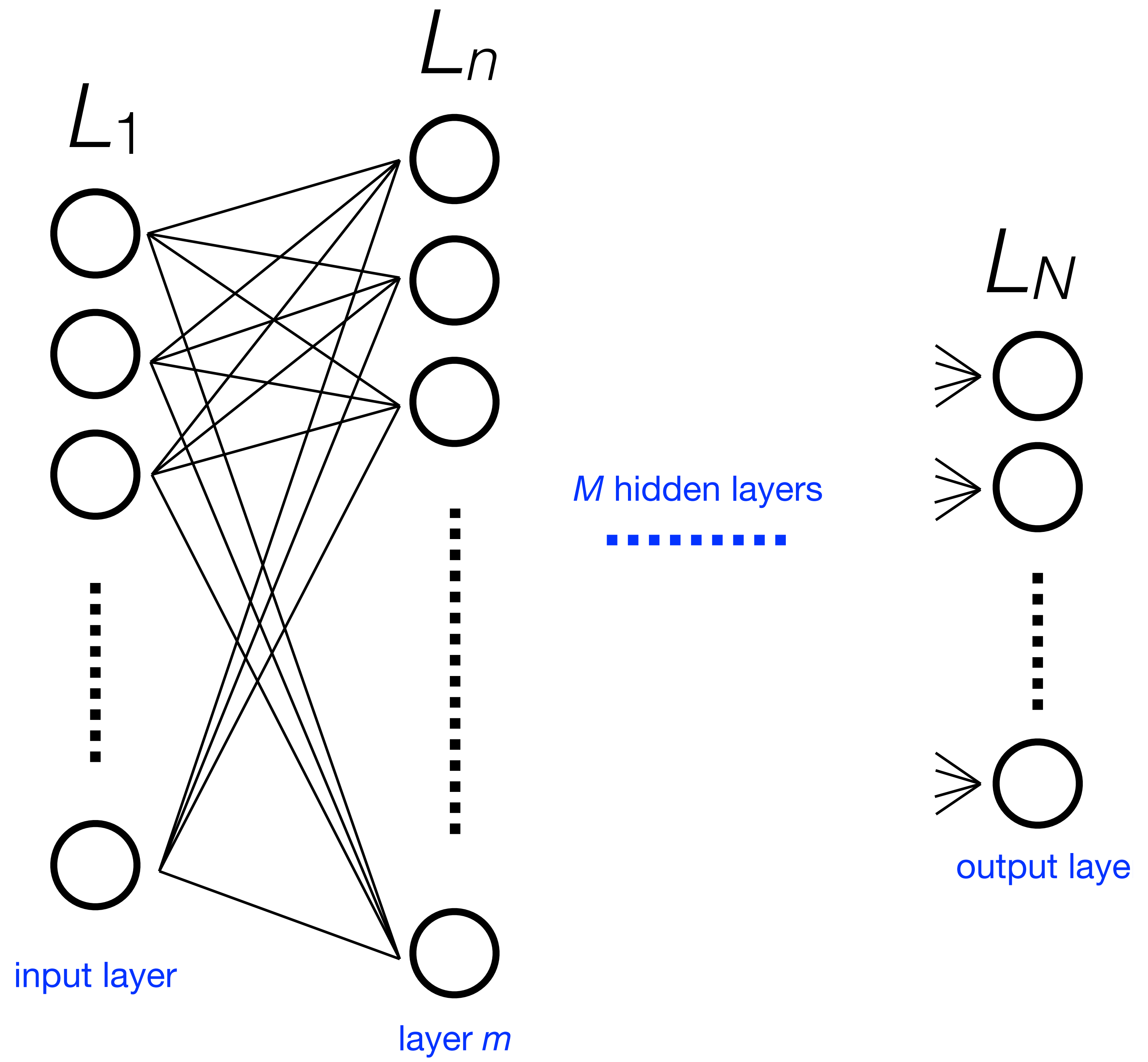
Synthesis

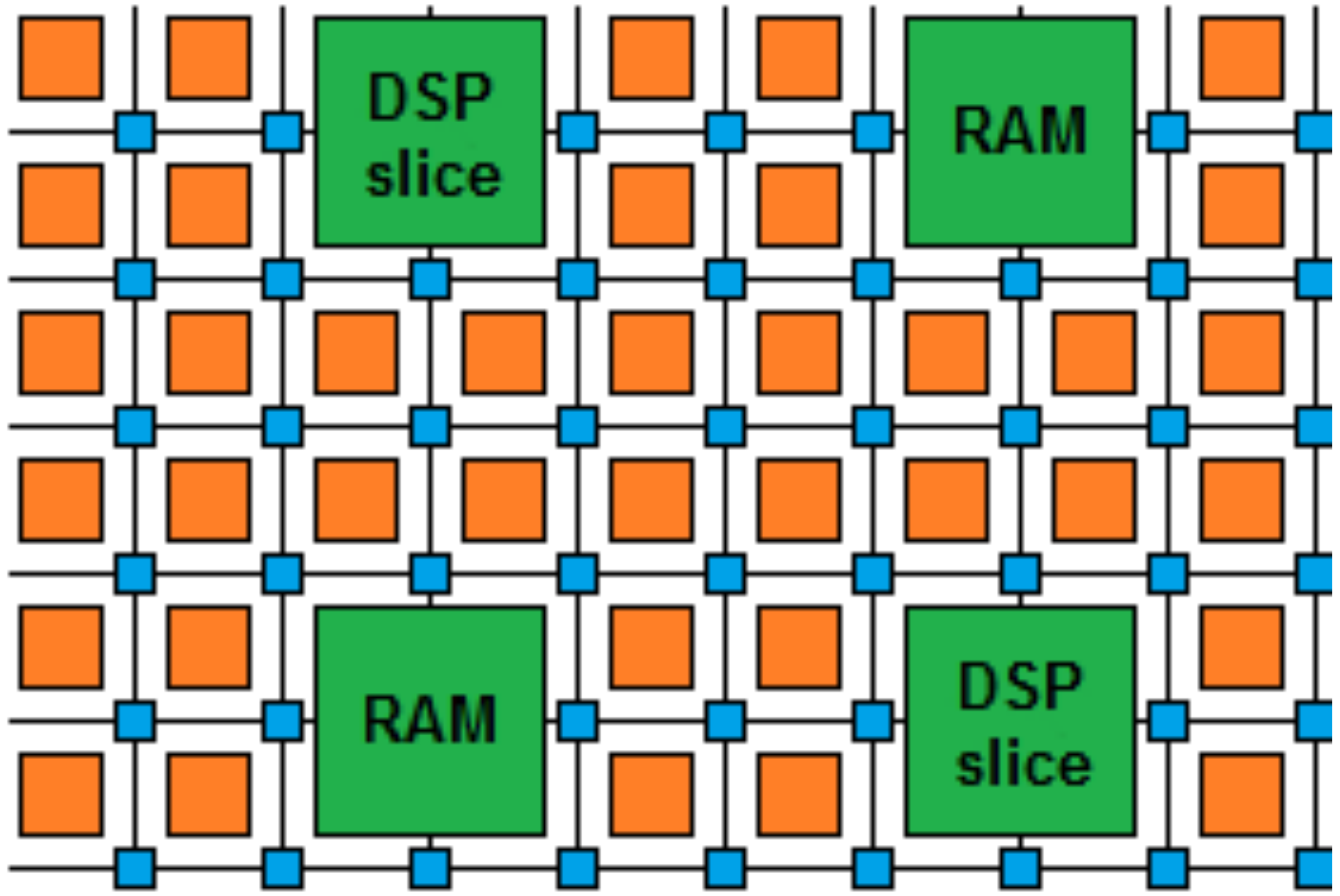


- Register transfer-level (RTL) code is "synthesized" into gates

Mapping NN onto FPGAs

FPGA diagram





Logic cell:
Flip-flops (FF) and
look up tables (LUTs)

Digital Signal Processors (DSPs)

$$\mathbf{x}_n = g_n(\mathbf{W}_{n,n-1}\mathbf{x}_{n-1} + \mathbf{b}_n)$$

Activation functions
Precomputed, and stored in BRAMs

Multiplications
DSPs

Addition
Logic cells

$$N_{\text{multiplications}} = \sum_{n=2}^{N} L_{n-1} \times L_n$$

Virtex Ultrascale+ VU9P

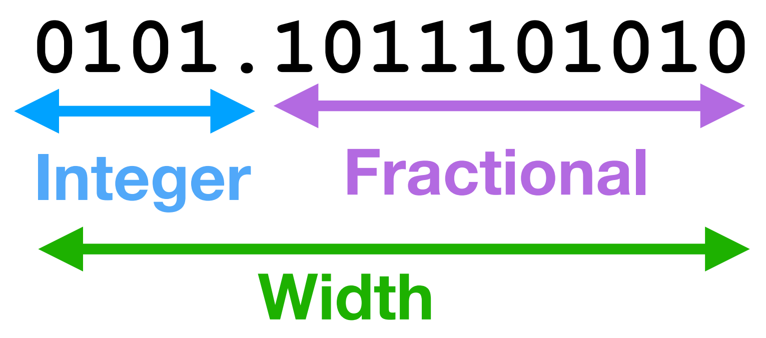
→ 6800 DSPs
1M LUTs
2M FFs
75 Mb BRAM

Today’s Lecture

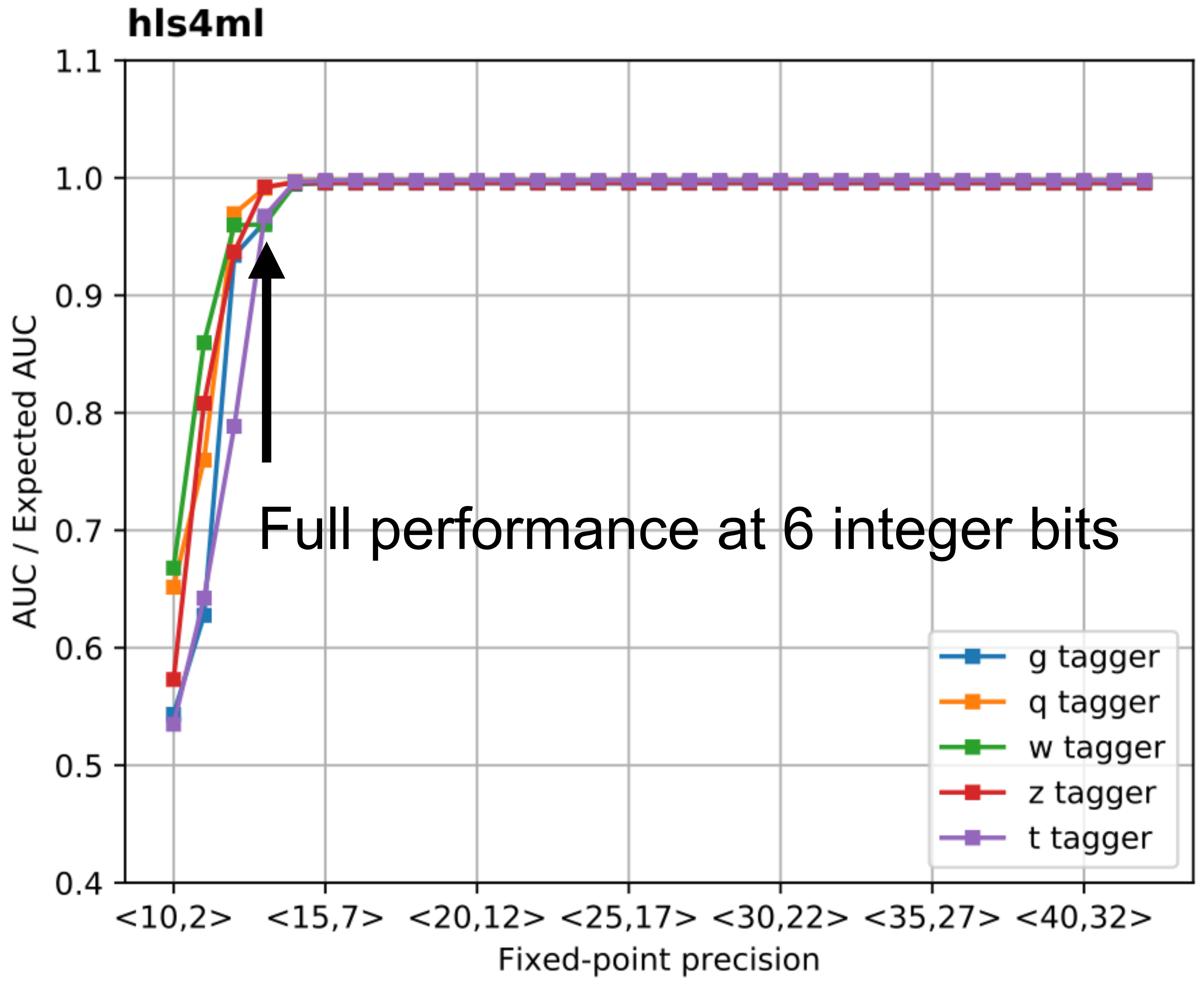
- Real-time system constraints and needs
  - LHC as an example
  - Specialized hardwares: FPGA/ASIC
- **Challenges in AI/ML on specialized hardwares**
  - Design efficient networks for real-time systems
  - Co-design tools and needs
- Real-time AI for other science domains: quantum system control etc

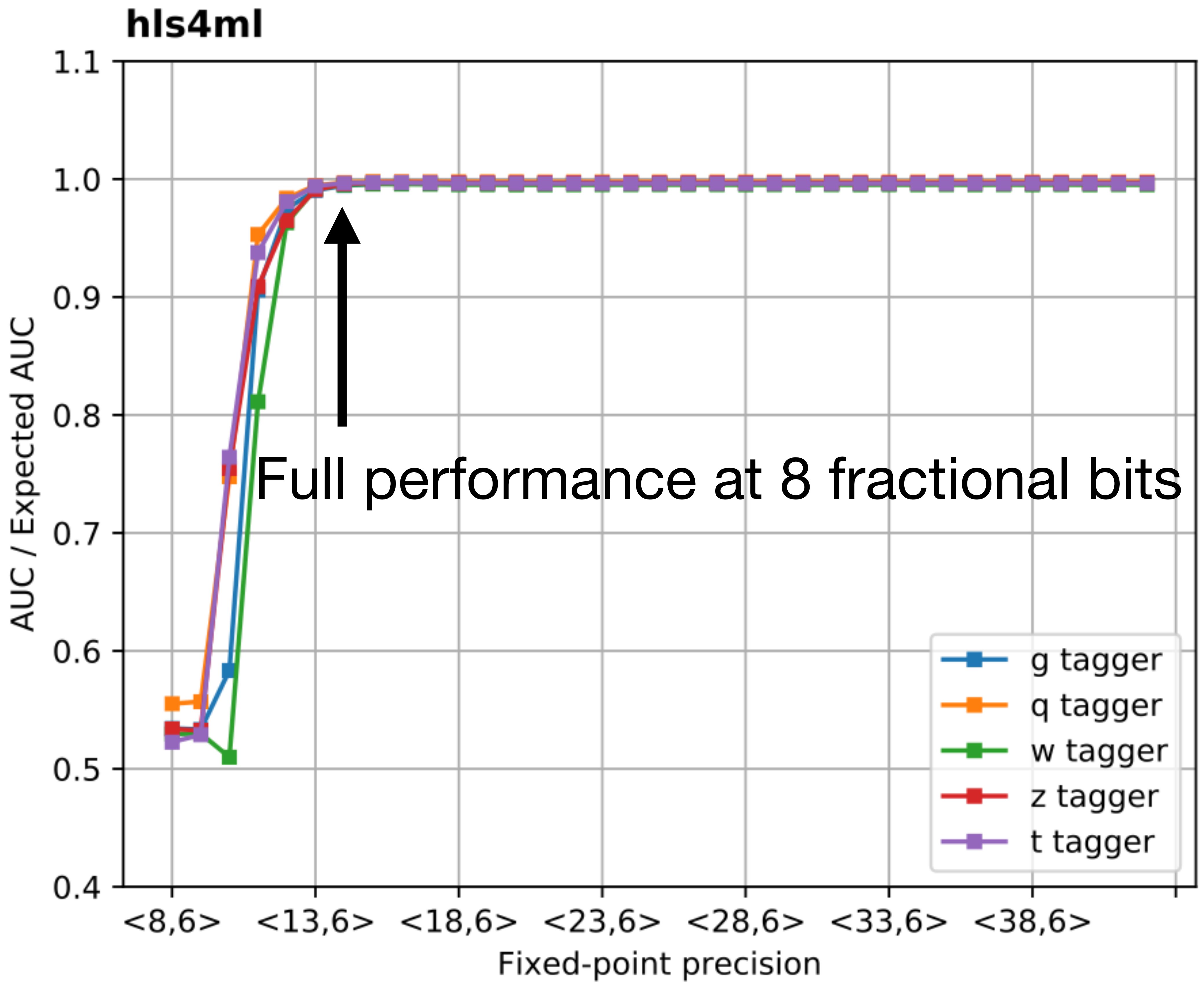
Quantization

ap_fixed<width bits, integer bits>



<Total, Fractional>

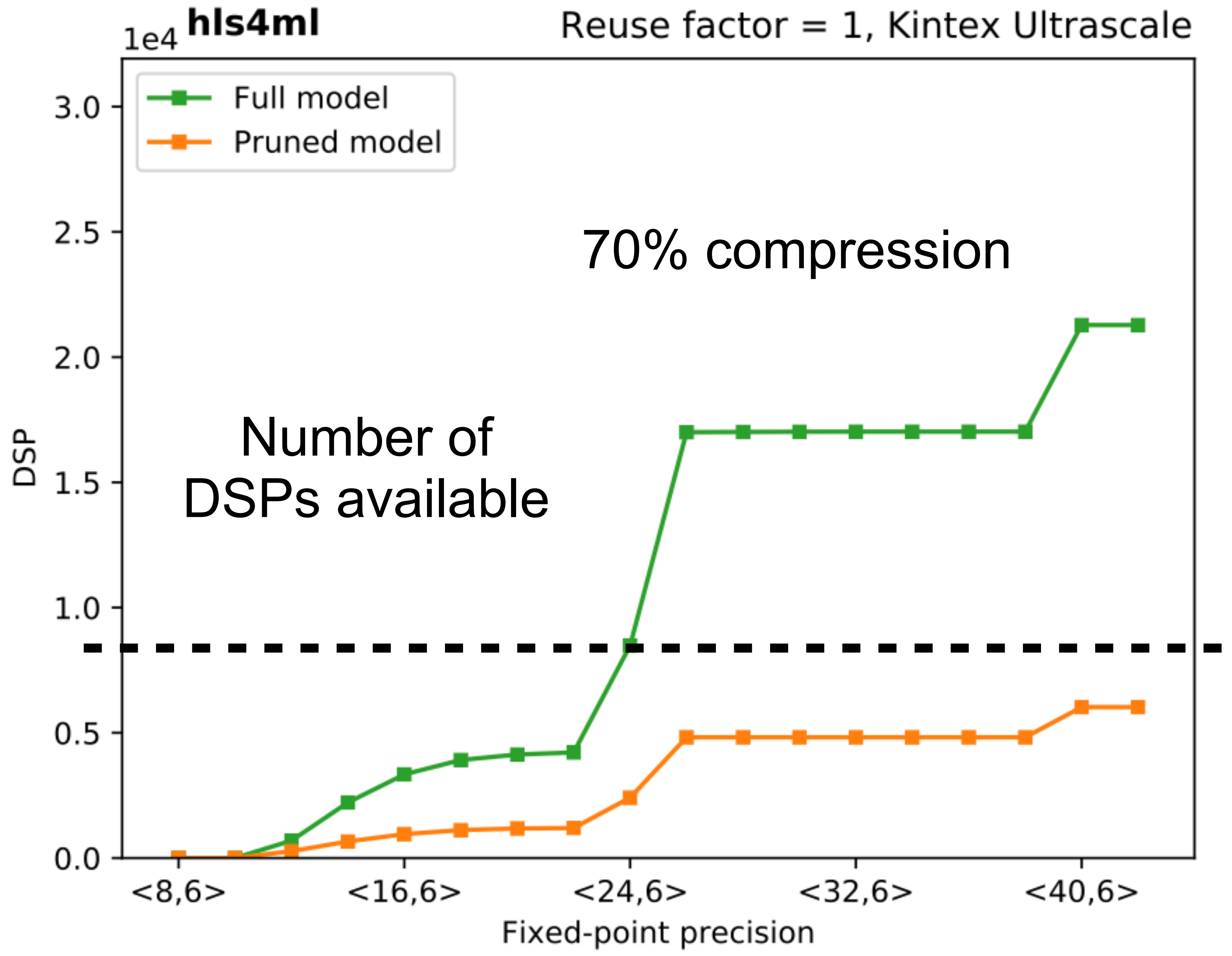


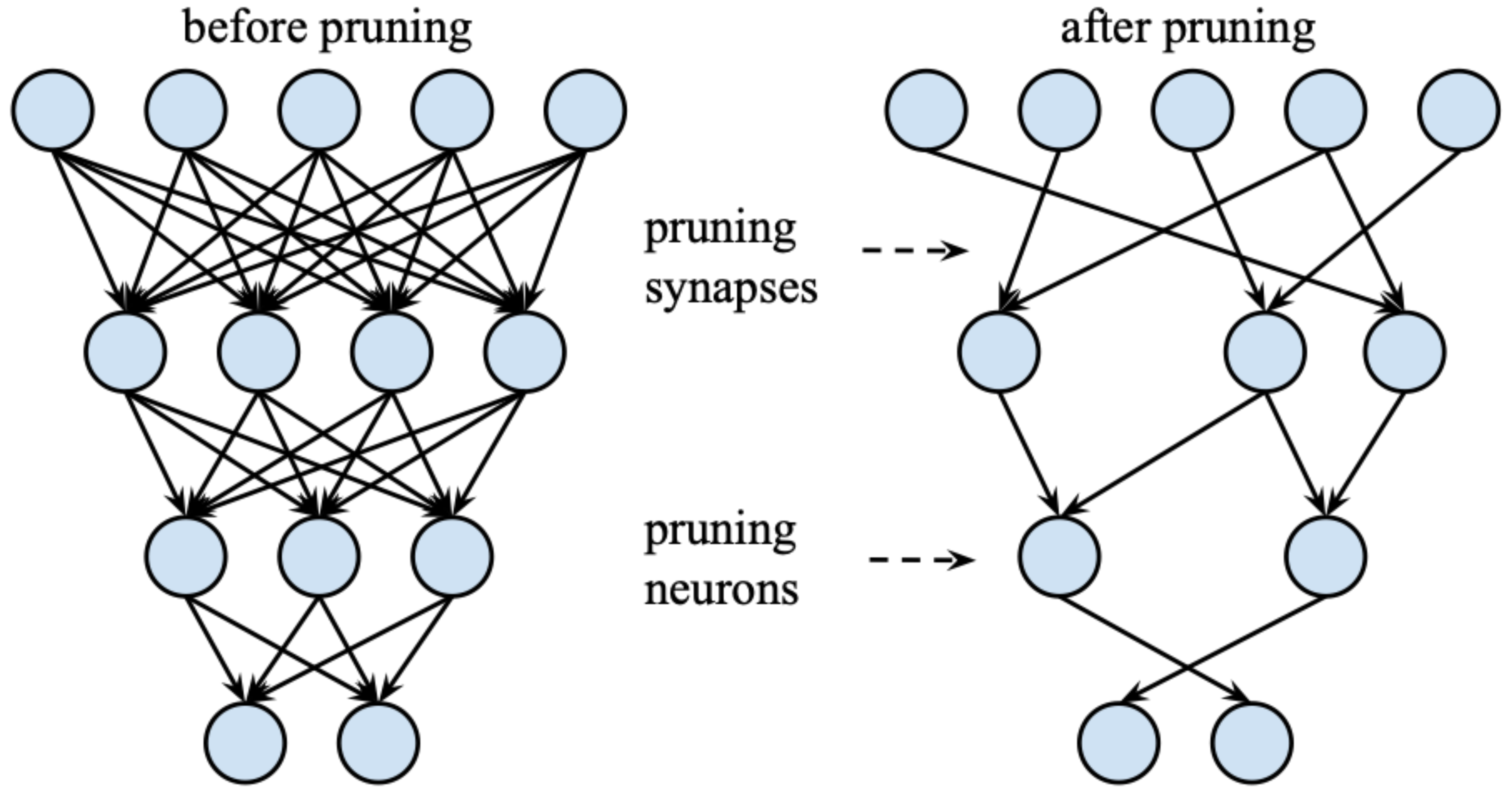


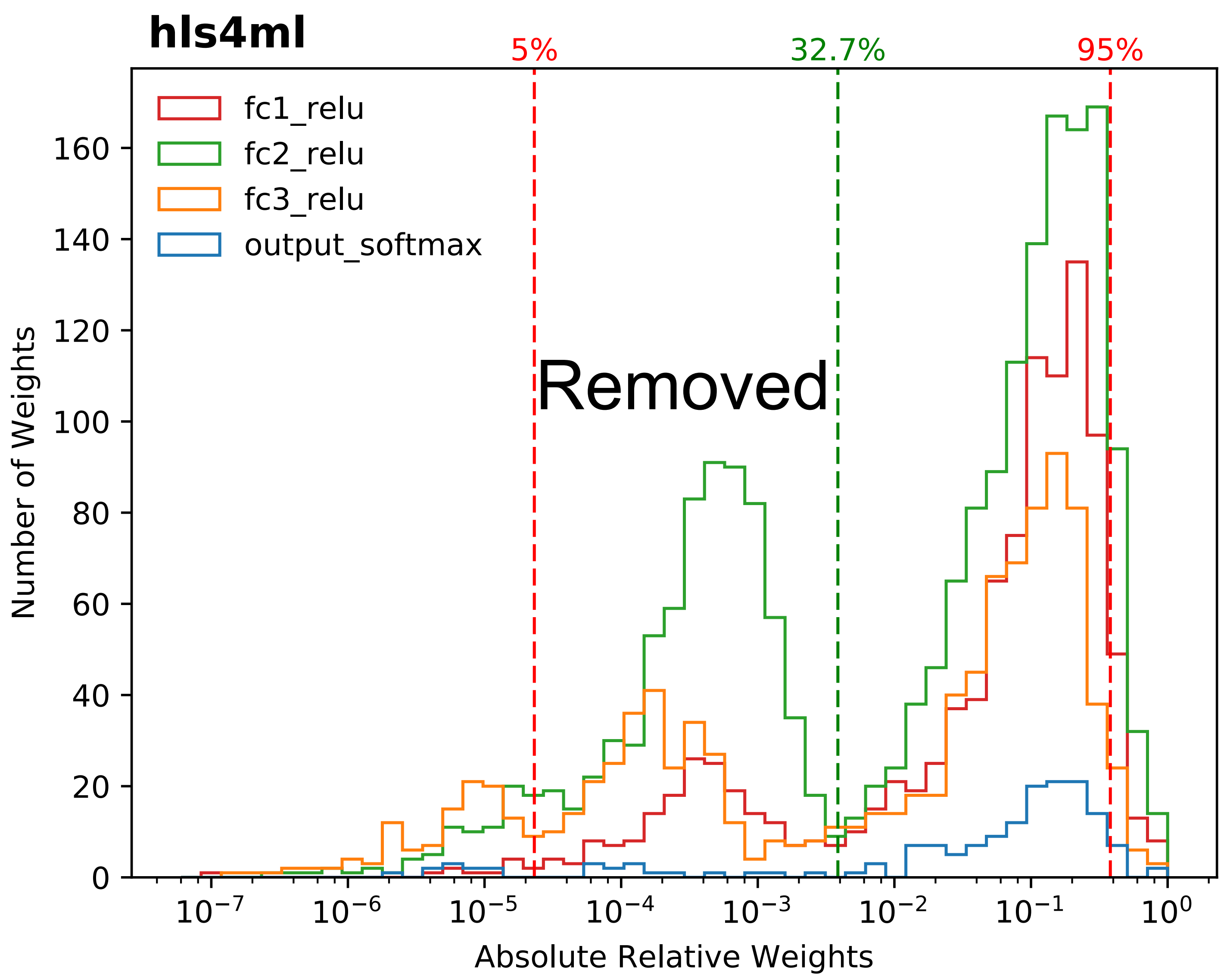
Scan integer bits, fractional bits fixed to 8

Scan fractional bits, Integer bits fixed to 6

Pruning







- DSPs (used for multiplication) are often limiting resource
  - maximum use when fully parallelized
  - Number of DSPs per multiplication changes with precision
- Iterative pruning with L1 norm penalty term: penalizes small weights

Network compression/Efficient Machine Learning Computing

- Many approaches have been studied:
  - Parameter pruning: selective removal of weights based on a particular ranking [arxiv.1510.00149, arxiv.1712.01312]
  - **Neural Network Architecture Search (NAS)** [https://arxiv.org/pdf/2301.08727.pdf]
  - **Knowledge distillation**: training a compact network with distilled knowledge of a large network [https://arxiv.org/abs/1503.02531]
  - Low-rank factorization: using matrix/tensor decomposition to estimate informative parameters [arxiv.1405.3866]
  - Transferred/compact convolutional filters: special structural convolutional filters to save parameters [arxiv.1602.07576]
  - Tensorflow model sparsity toolkit: https://blog.tensorflow.org/2019/05/tf-model-optimization-toolkit-pruning-API.html

Image detection network evolution

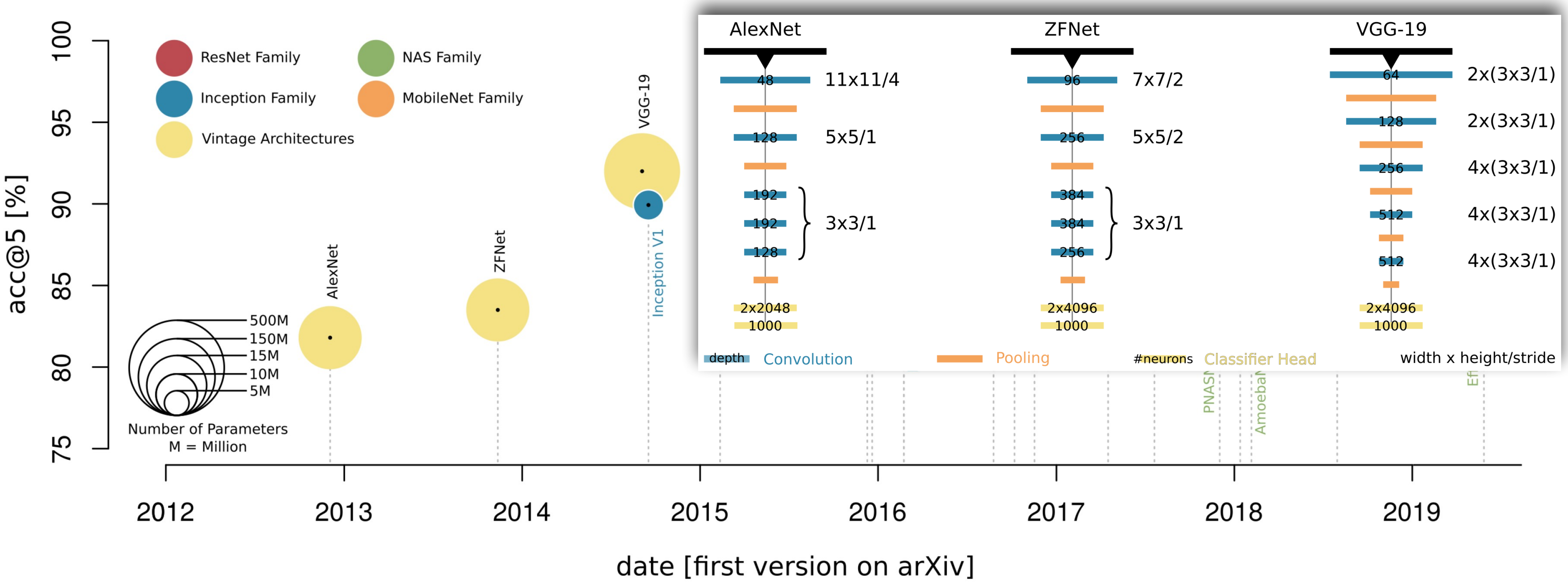
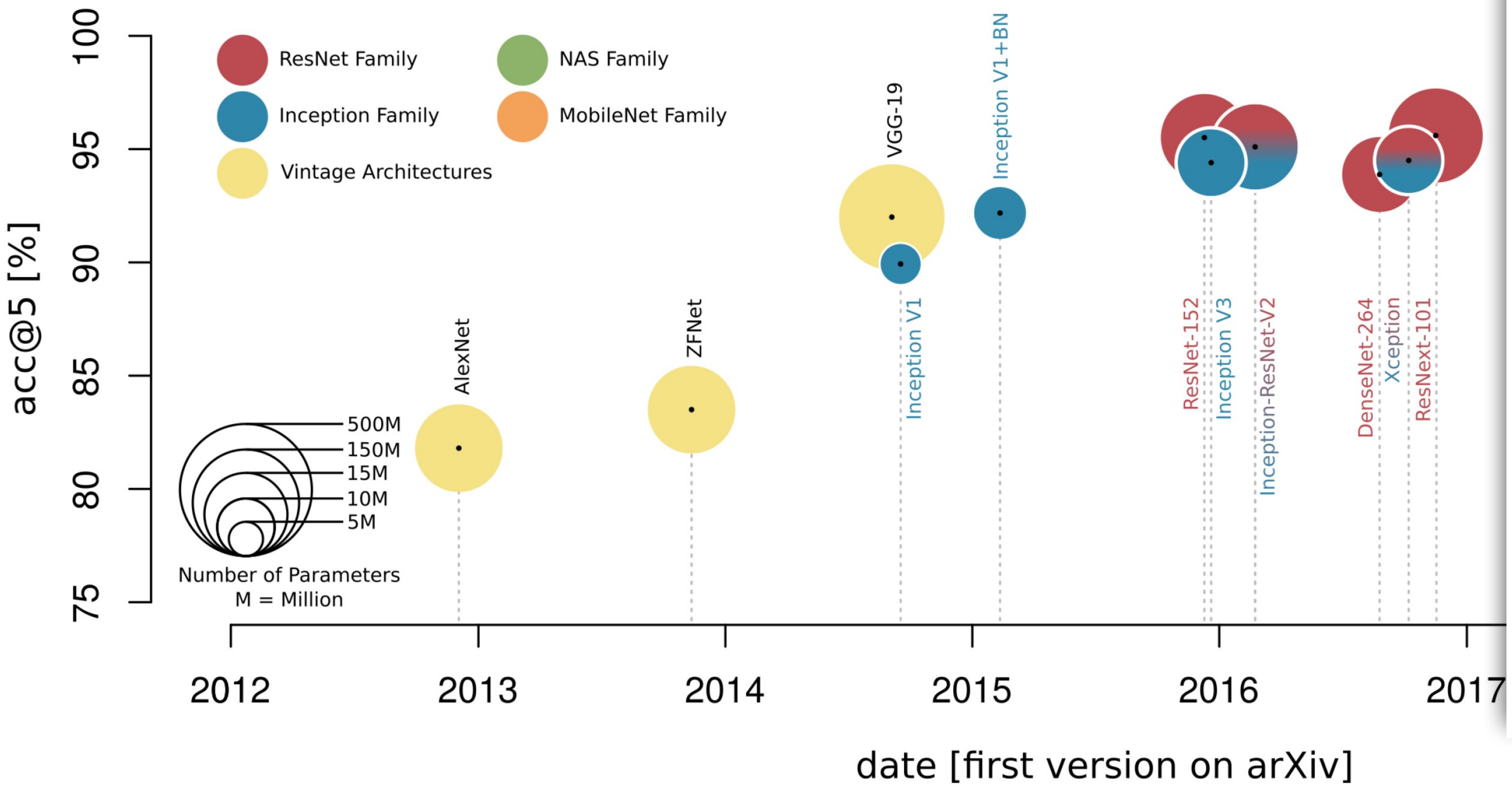


Image detection network evolution



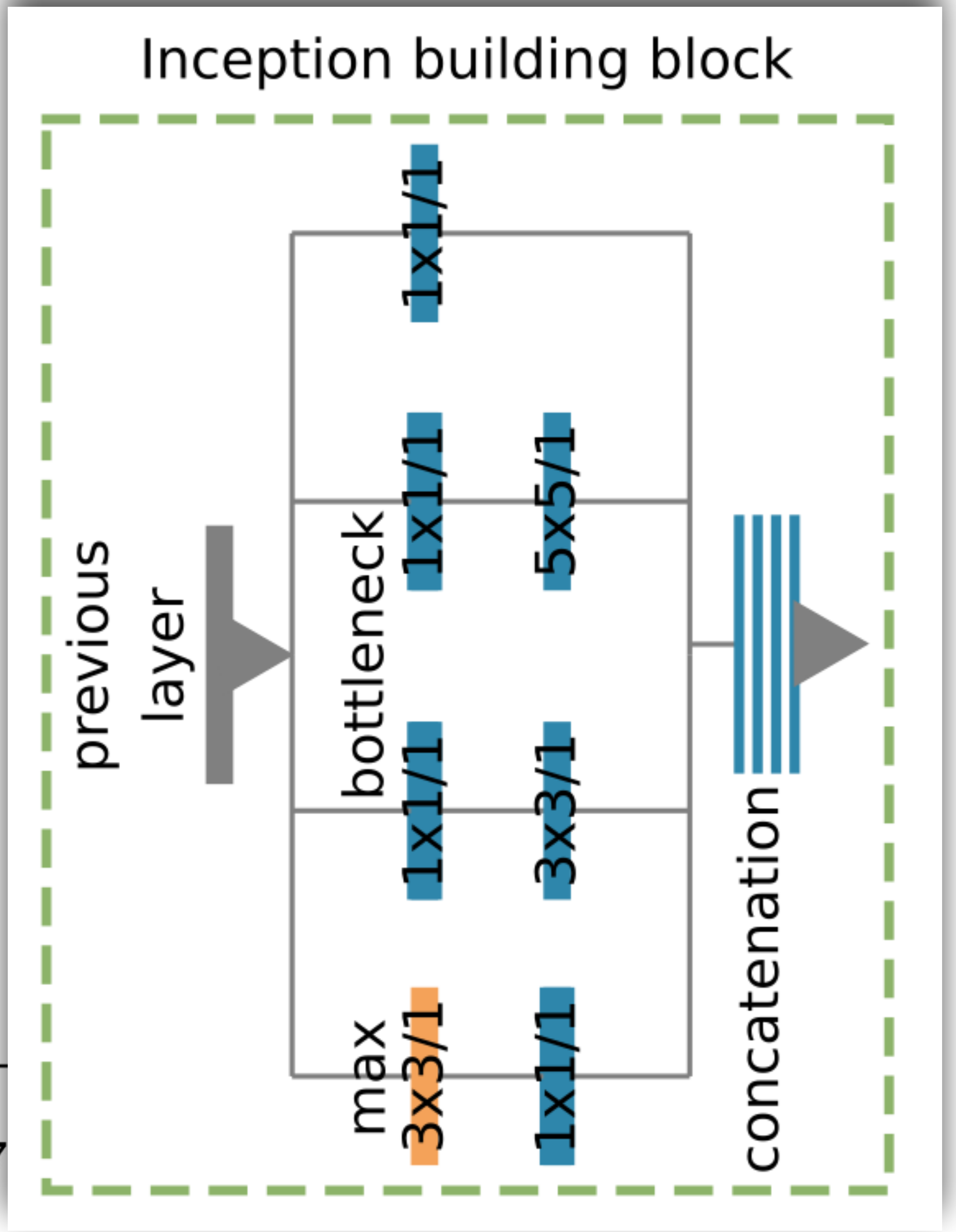
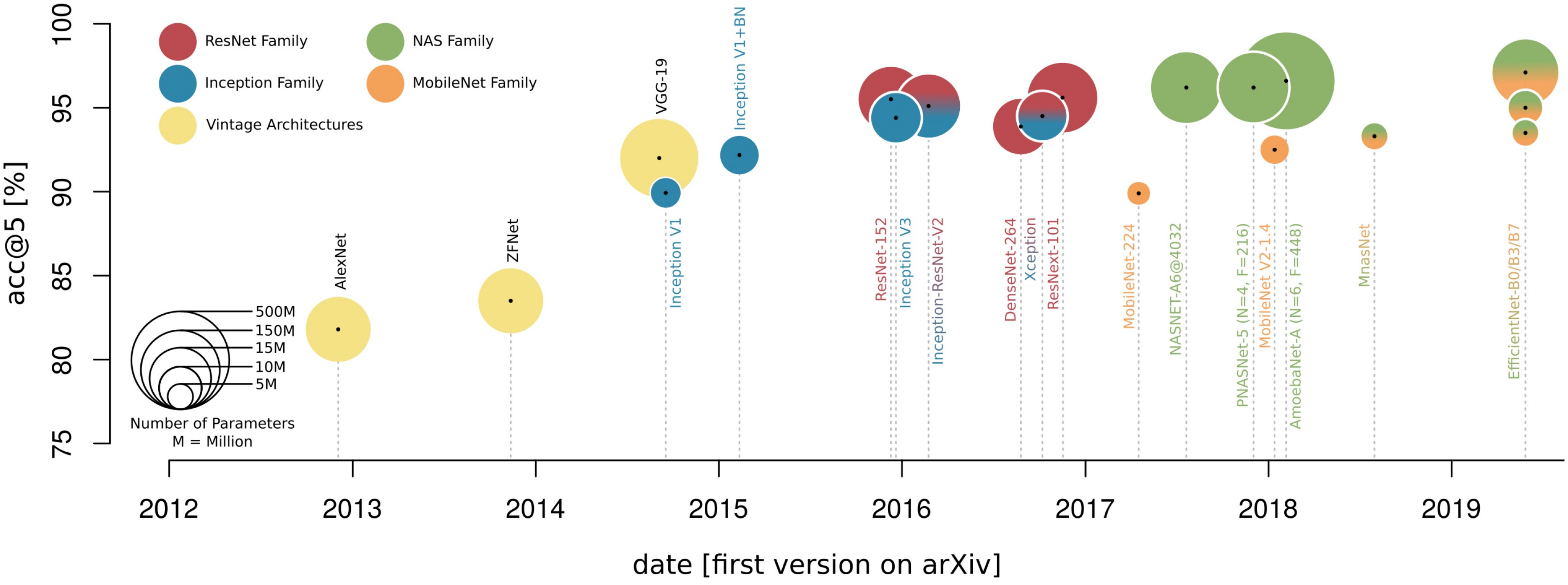
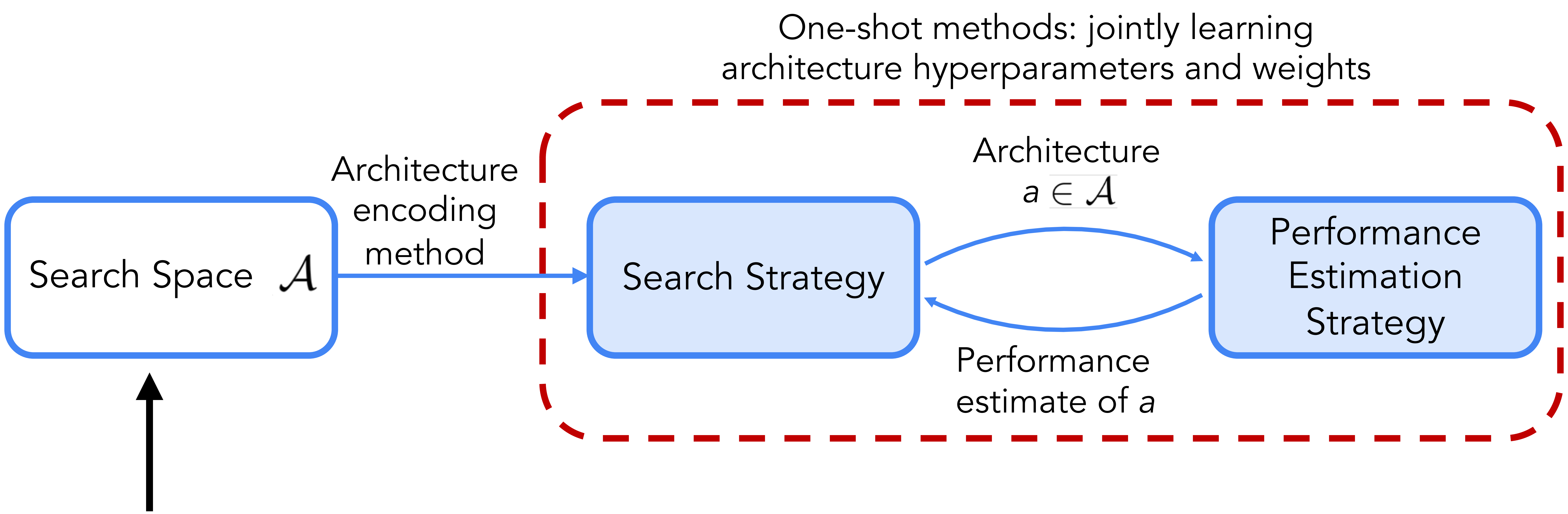


Image detection network evolution



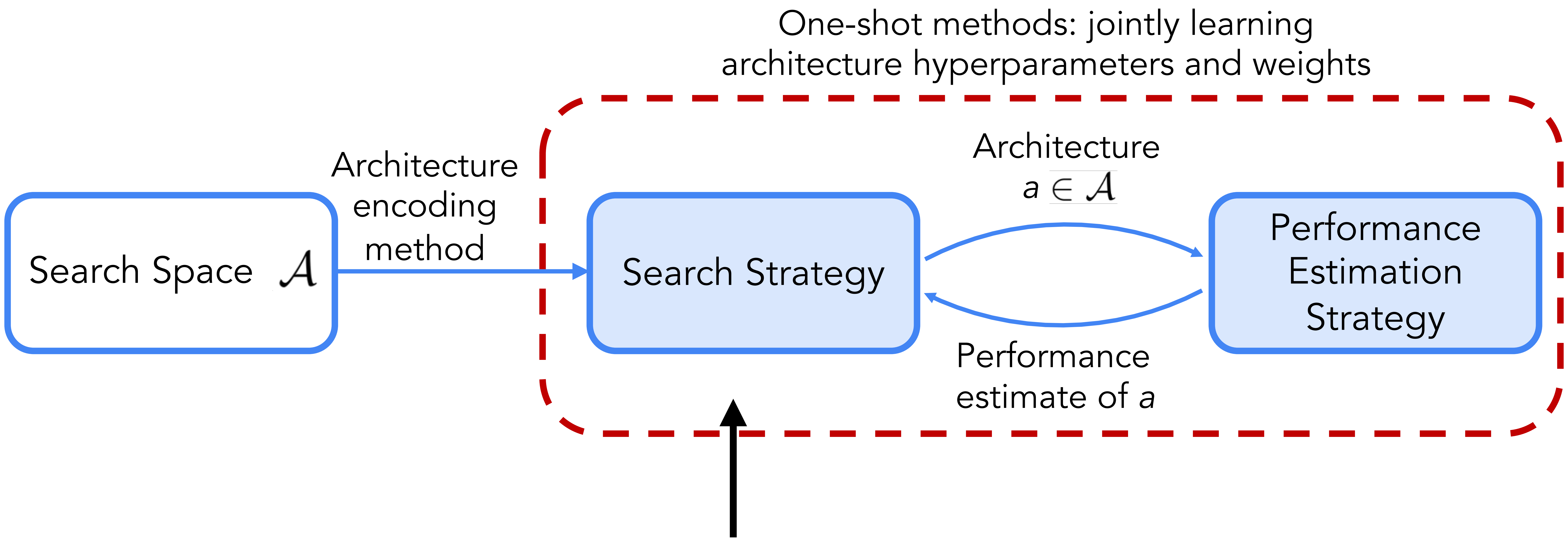
The modular concept of CNNs and their building blocks is crucial for the next group of architectures by **Neural Architecture search.**

Automation: Neural Network Architecture Search



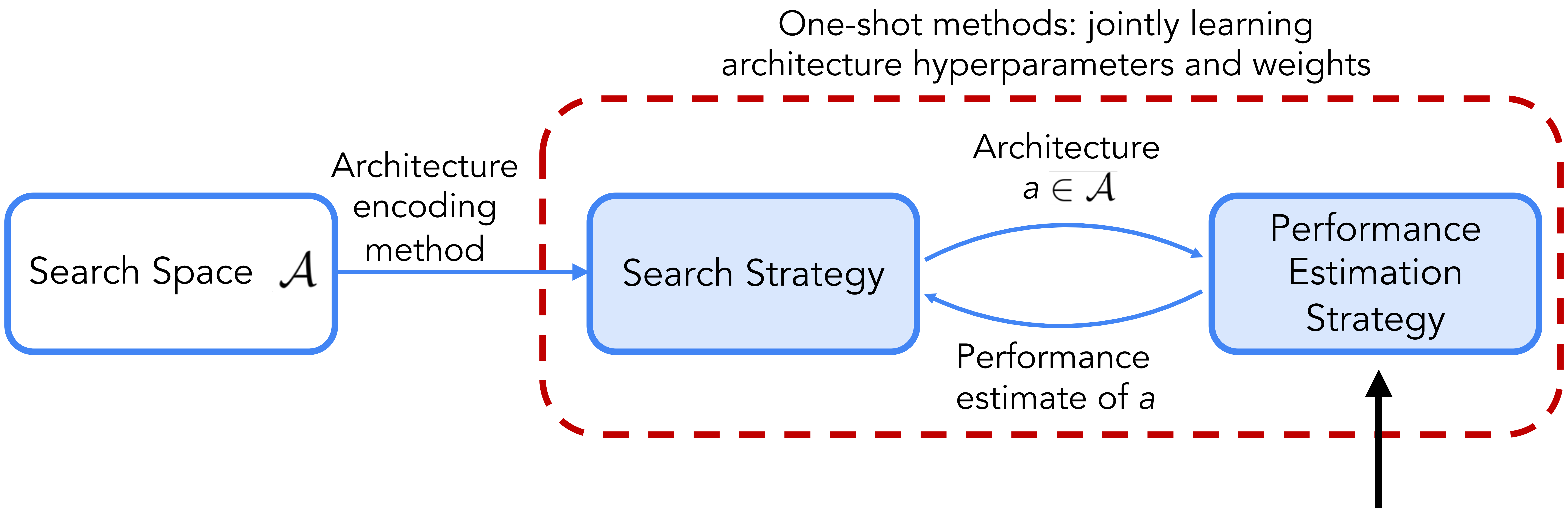
- Choose building blocks:
- Operation/primitive: denotes the atomic unit, a popular one is a triplet of a fixed activation, operation, and fixed normalization, such as ReLU-conv 1x1-batchnorm
- Layer, Block, Cell, Motif

Neural Network Architecture Search



- Grid search, random search, re-enforcement learning, evolution, Bayesian

Neural Network Architecture Search



- Latency, accuracy, power consumption, hardware types etc

NAS for MobileNet-v2

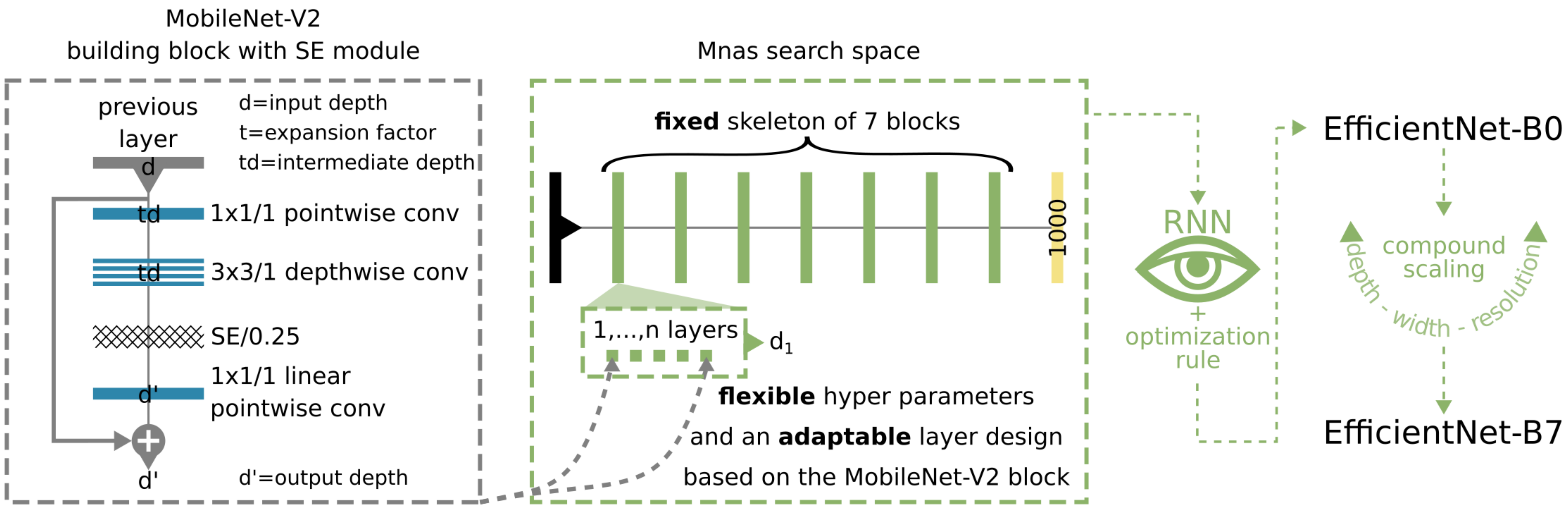


Figure 8. Conceptual overview of the *Efficient designs*: (**Left**) The MobileNetV2 building block [108], here with an additional Squeeze-and-Excitation (SE) module [47]. In comparison with a ResNet building block, the bottleneck design is inverted, so that first the expansion factor (t) is larger than 1, which leads to intermediate deeper feature maps (td) as the final output depth of the building block (d′). (**Middle**) The Mnas search space with a fixed overall architecture of the network, the skeleton, but fully optional layer designs, based on the MobileNetV2 building block [51]. (**Right**) A recurrent neural network (RNN) [34] controller searches the search space for the best performing combination of layer designs by maximising an optimisation rule [48]. The resulting architecture is scaled in depth, width and resolution to become the EfficientNet-B7 architecture, the sota design in late 2019 [52].

Distilling the Knowledge in a Neural Network

Geoffrey Hinton[*†]
Google Inc.
Mountain View
geoffhinton@google.com

Oriol Vinyals[†]
Google Inc.
Mountain View
vinyals@google.com

Jeff Dean
Google Inc.
Mountain View
jeff@google.com

Abstract

A very simple way to improve the performance of almost any machine learning algorithm is to train many different models on the same data and then to average their predictions [3]. Unfortunately, making predictions using a whole ensemble of models is cumbersome and may be too computationally expensive to allow deployment to a large number of users, especially if the individual models are large neural nets. Caruana and his collaborators [1] have shown that it is possible to compress the knowledge in an ensemble into a single model which is much easier to deploy and we develop this approach further using a different compression technique. We achieve some surprising results on MNIST and we show that we can significantly improve the acoustic model of a heavily used commercial system by distilling the knowledge in an ensemble of models into a single model. We also introduce a new type of ensemble composed of one or more full models and many specialist models which learn to distinguish fine-grained classes that the full models confuse. Unlike a mixture of experts, these specialist models can be trained rapidly and in parallel.

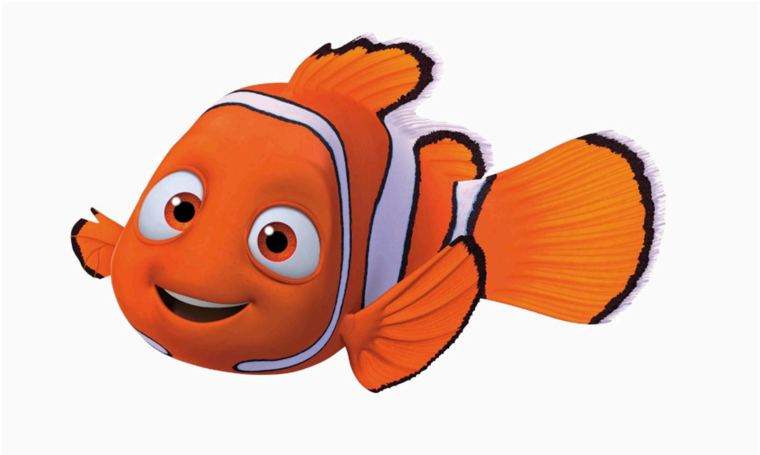
Knowledge distillation

- We have trained a fully supervised model with MLP (fully connected neural networks)
    - Overfit with an overparametrized model
    - Add regularization to improve generalizability
    - Ensemble of models
- Accurate, but big and cumbersome—> not suitable for computing resource constrained use cases
- Small model is not as performant

- Can we transfer the knowledge learned by the large teacher model to a student model?
    - Efficient and performant

Blobfish

?



Knowledge distillation

- Where is this knowledge stored?
  - Multi-class classification: "Soft labels" that generalizes to unseen datasets

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

Logits

σ = softmax

$\vec{z}$ = input vector

e^{z_i} = standard exponential function for input vector

K = number of classes in the multi-class classifier

e^{z_j} = standard exponential function for output vector

Blobfish

?

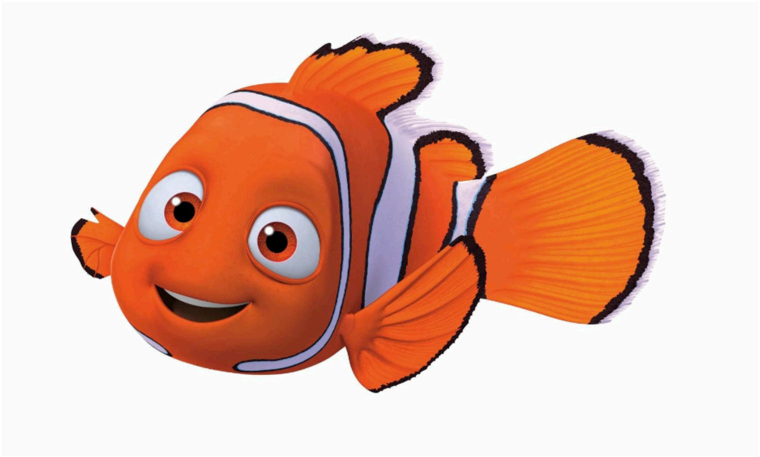
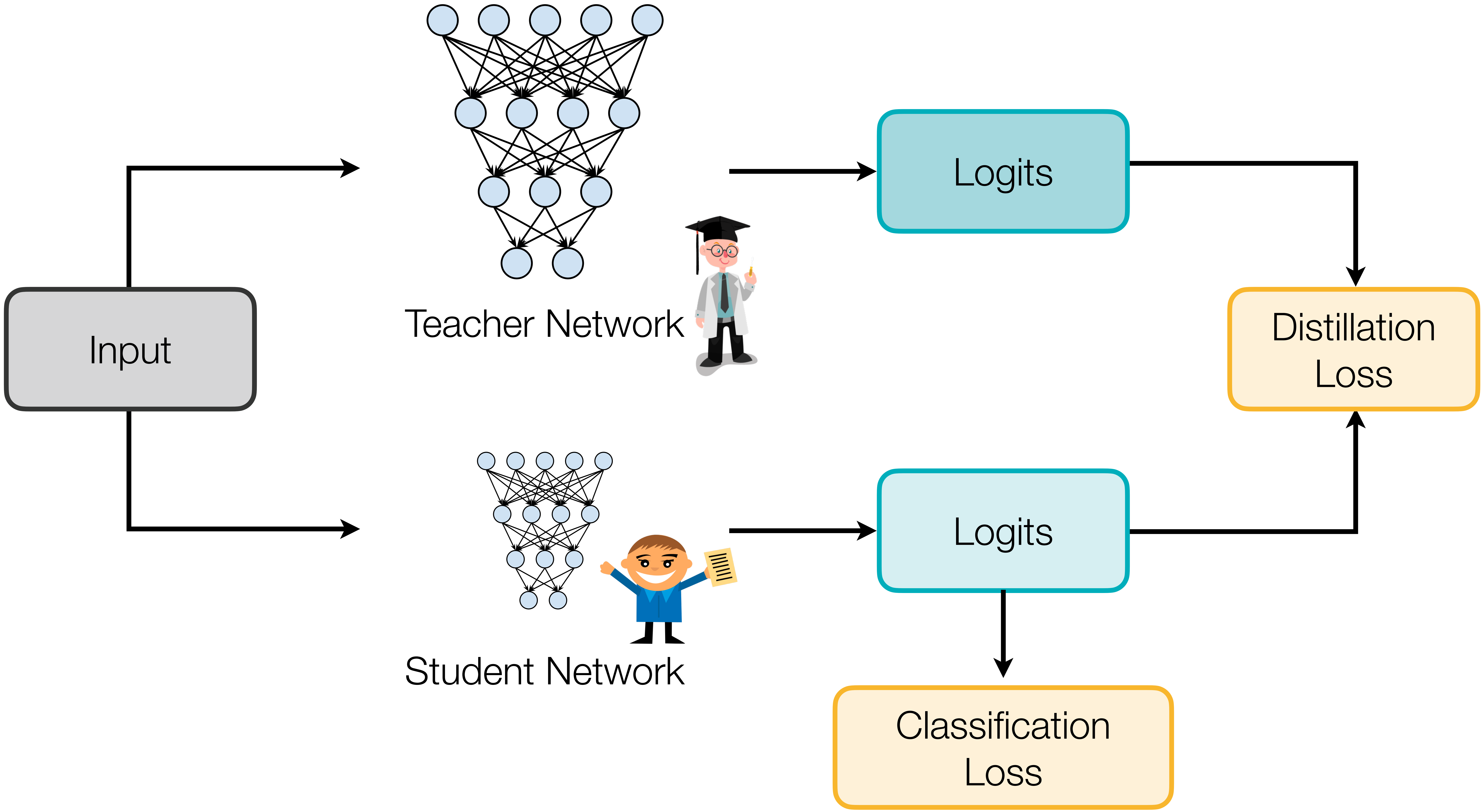
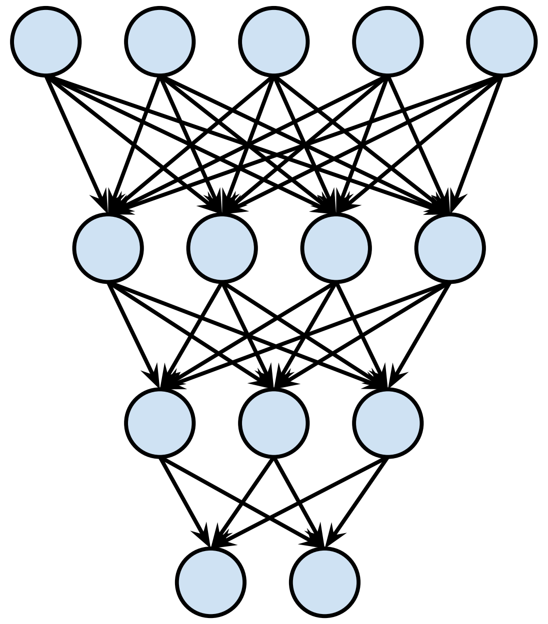


Illustration of knowledge distillation



Matching prediction probabilities between teacher and student

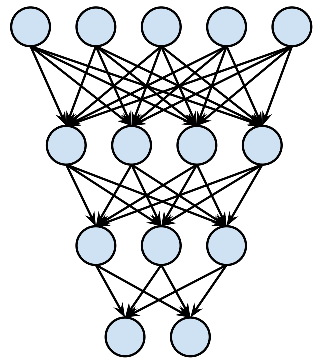


Teacher Network

	Logits	Probabilities
Dog	5	0.982
Cat	1	0.017

$$\frac{\exp(5)}{\exp(5) + \exp(1)}$$

$$\frac{\exp(1)}{\exp(5) + \exp(1)}$$

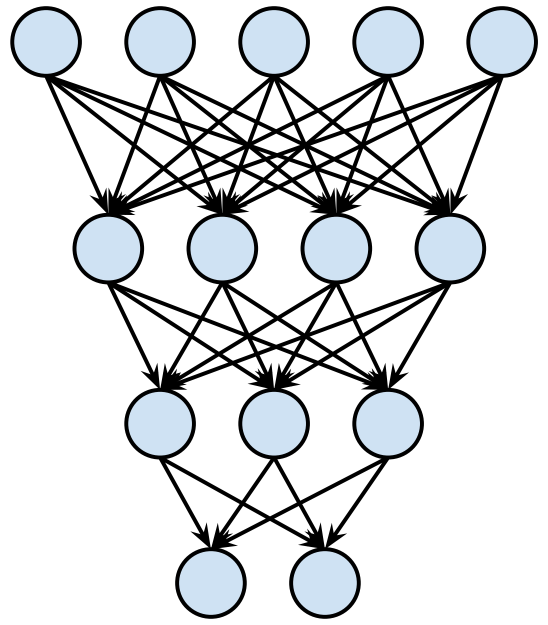


Student Network

	Logits	Probabilities
Dog	3	0.731
Cat	2	0.269

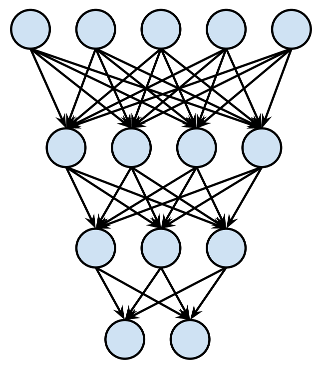
The student model is less confident

Student model gives less confident predictions



Teacher Network

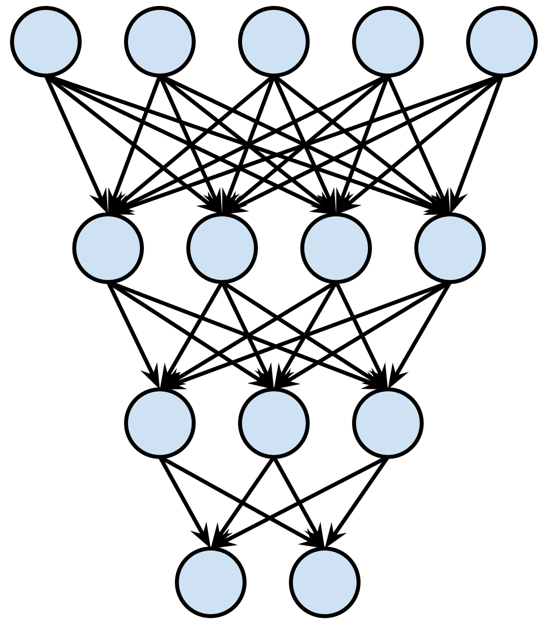
	Logits	Probabilities
Dog	5	0.982
Cat	1	0.017



Student Network

	Logits	Probabilities
Dog	3	0.731
Cat	2	0.269

Concept of temperature



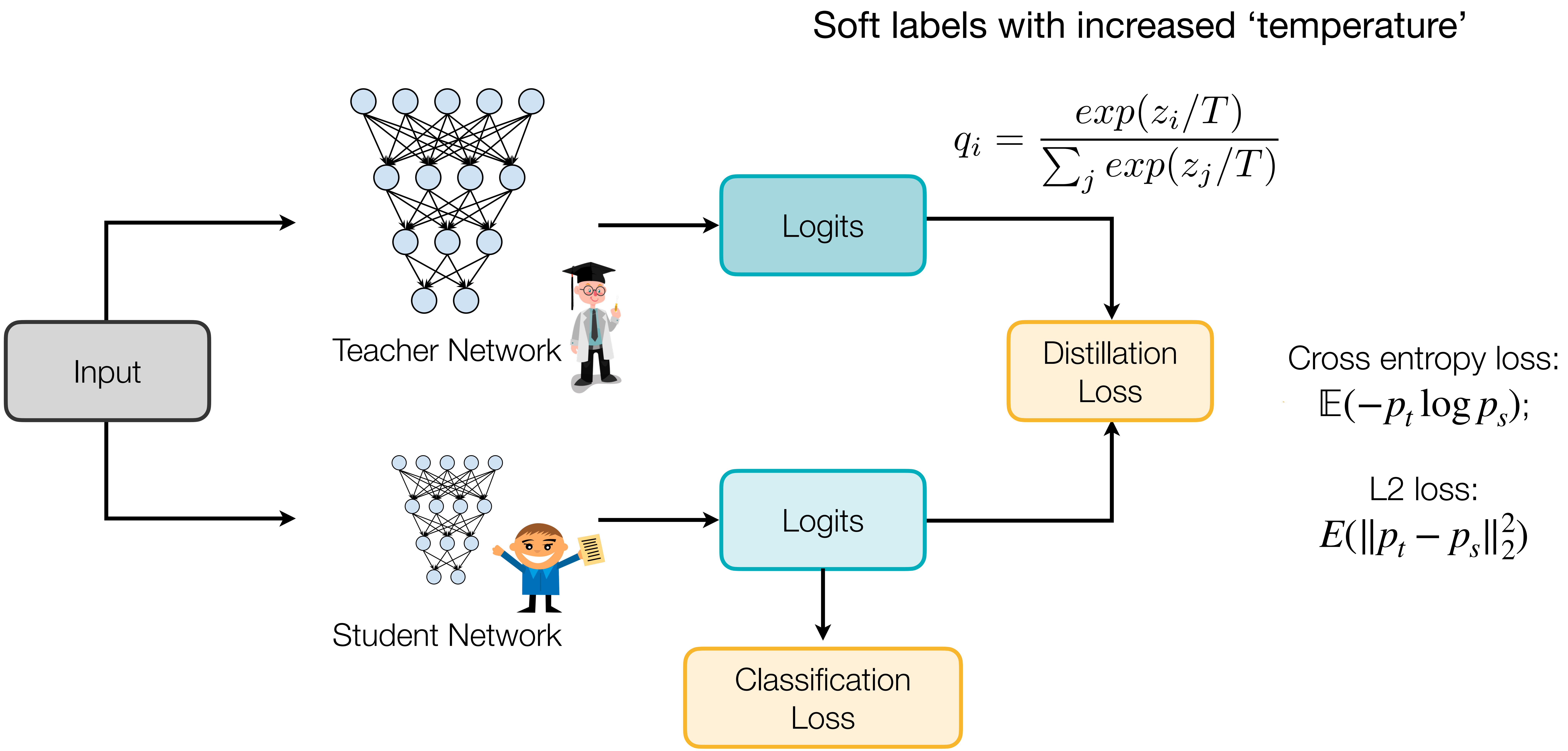
Teacher Network

$$\frac{\exp(5/1)}{\exp(5/1)+\exp(1/1)}$$

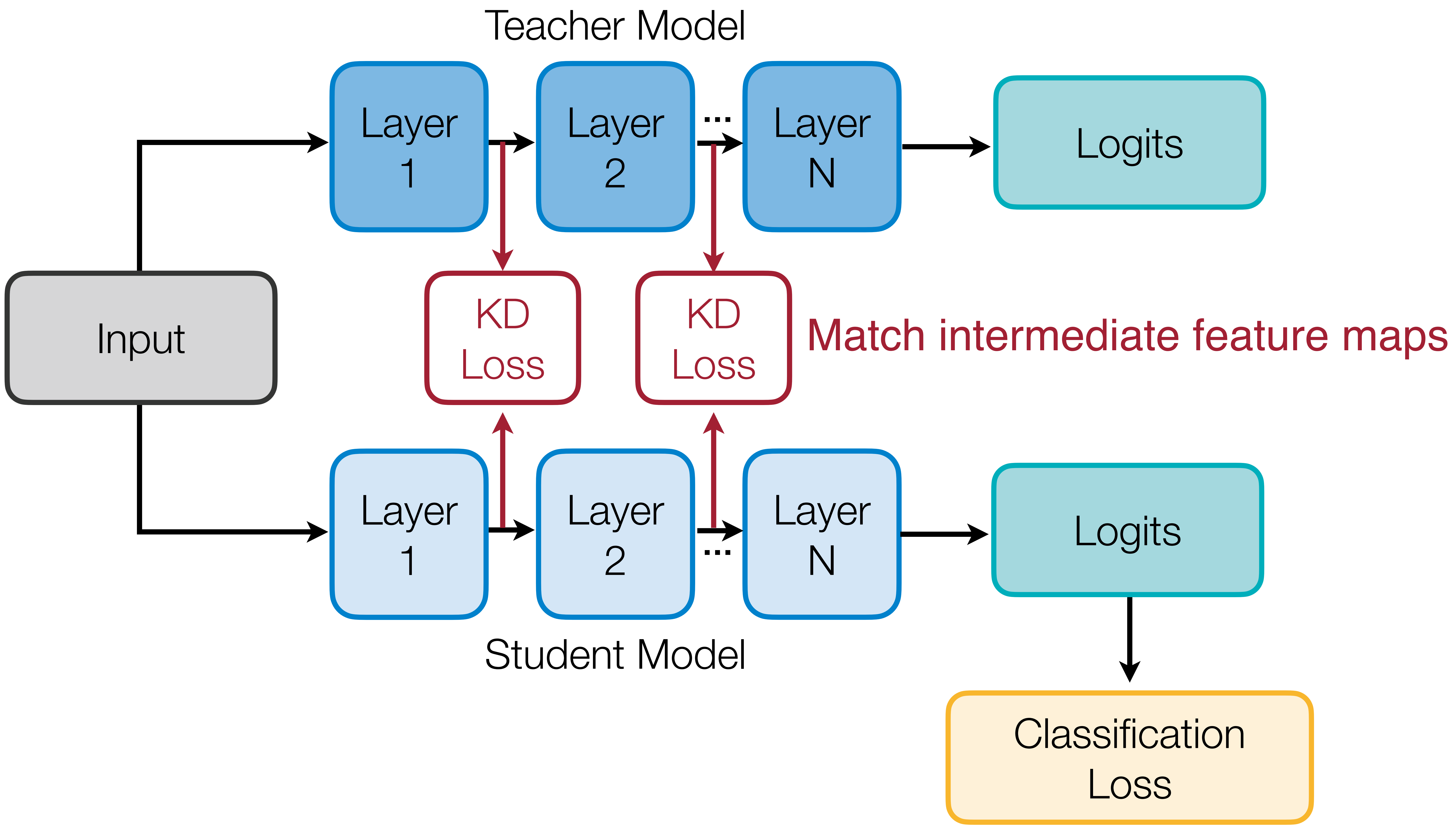
	Logits	Probabilities (T=1)	Probabilities (T=10)
Dog	5	0.982	0.599
Cat	1	0.017	0.401

$$\frac{\exp(5/10)}{\exp(5/10)+\exp(1/10)}$$

Align the class probability distributions from teacher and student networks



Match intermediate feature maps



Like What You Like: Knowledge Distill via Neuron Selectivity Transfer [Huang and Wang, arXiv 2017]

Match intermediate attention maps

Gradients of feature maps are used to characterize “attention” of DNNs

- The attention of a CNN feature map x is defined as $\dfrac{\partial L}{\partial x}$, where L is the learning objective.
- Intuition: If $\dfrac{\partial L}{\partial x_{i,j}}$ is large, a small perturbation at i, j will significantly impact the final output. As a result, the network is putting more attention on position i, j.

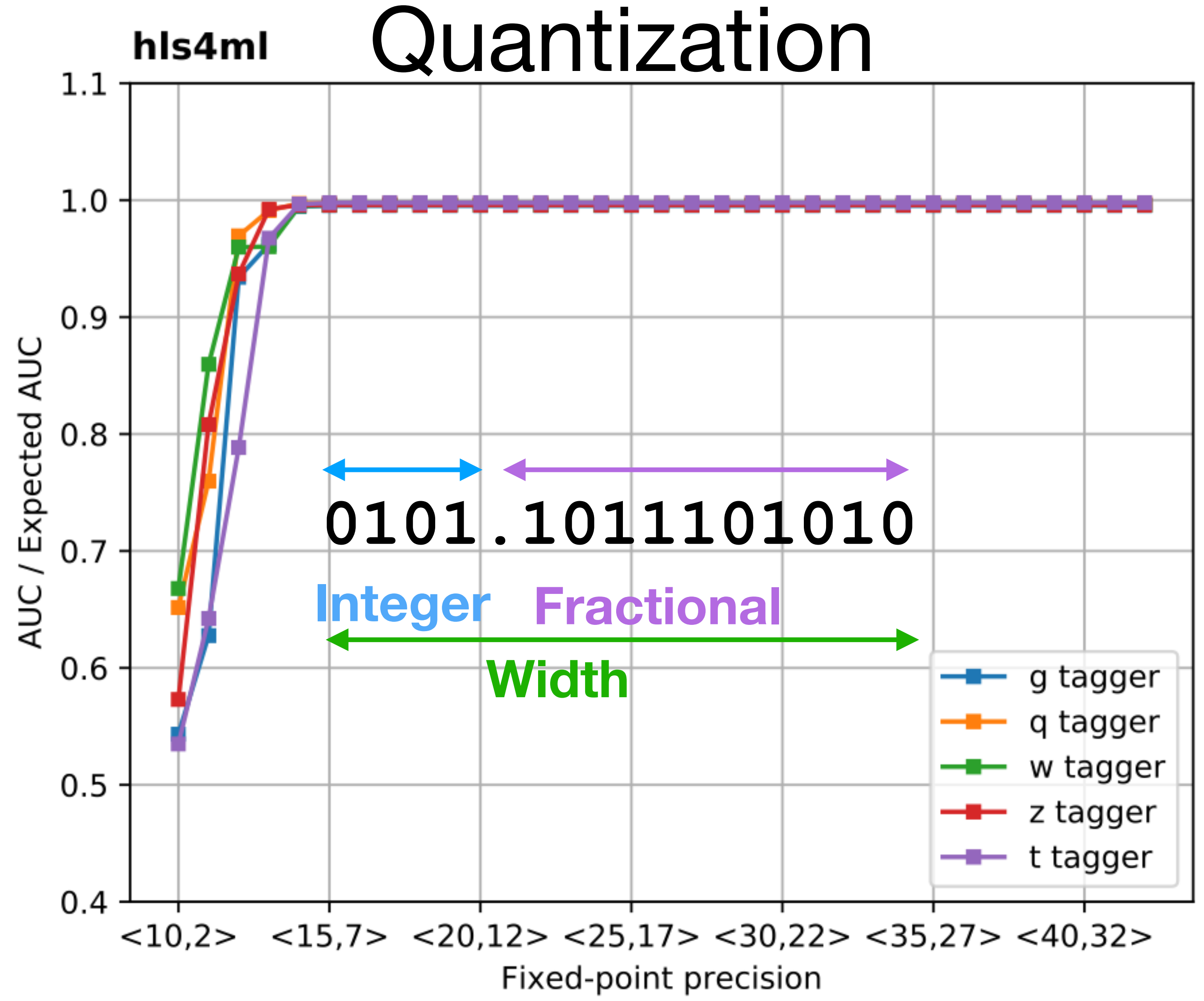
input image attention map

Paying More Attention to Attention: Improving the Performance of Convolutional Neural Networks via Attention Transfer [Zagoruyko and Komodakis, ICLR 2017]

Efficient Algorithms

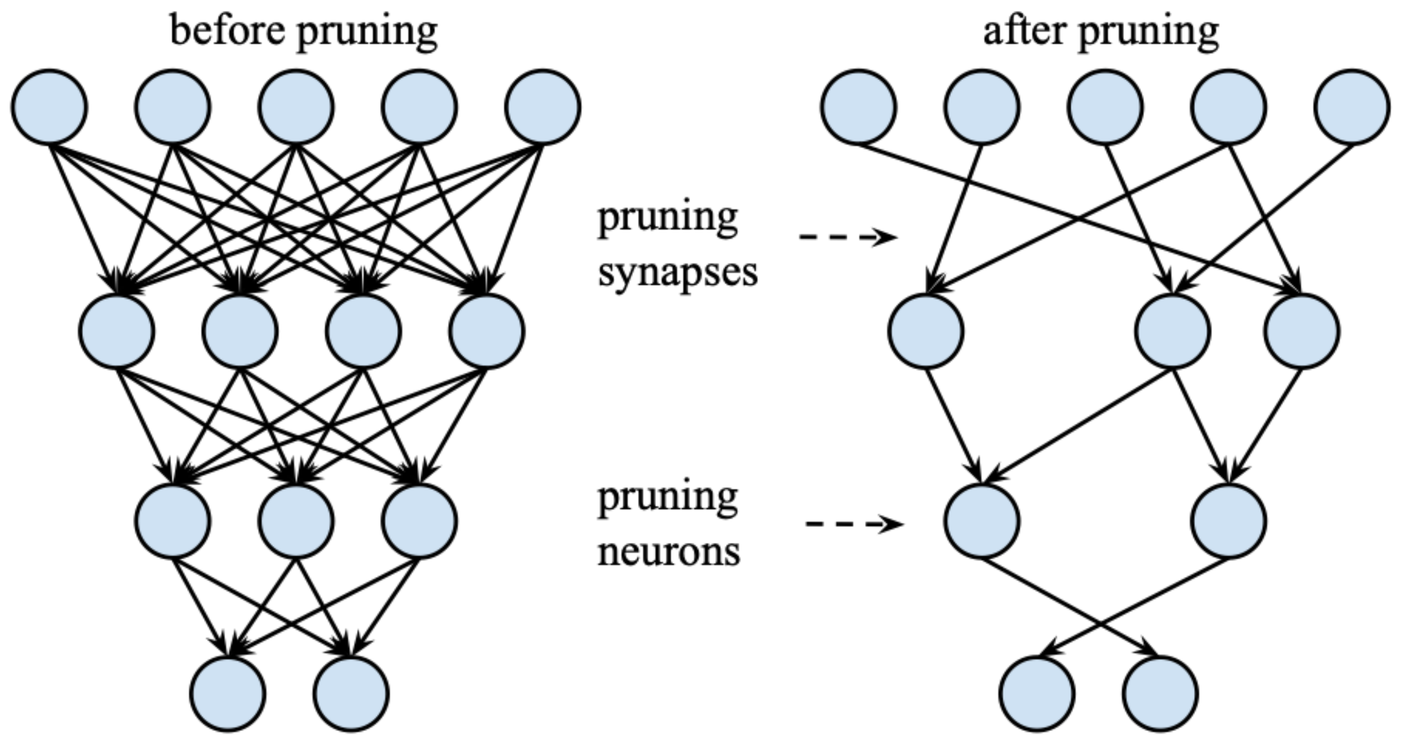
Efficient Algorithms

Quantization



Efficient Algorithms

Pruning



Efficient Algorithms

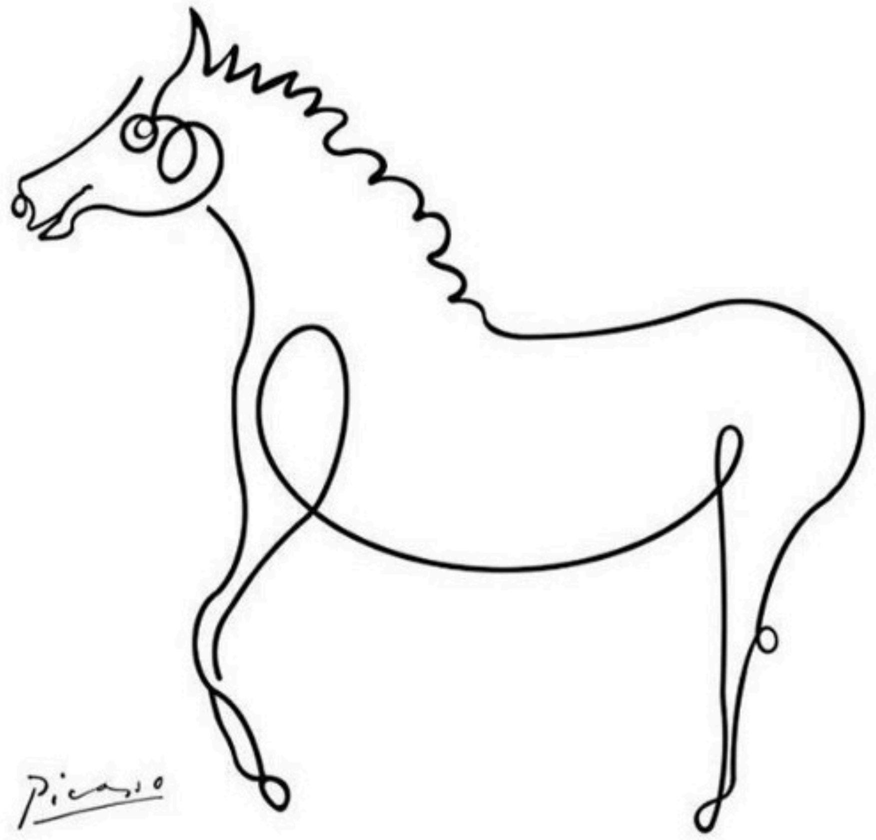
Quantization

Compression/Pruning

'Ultimate optimization' of 'bits of information':
Quantization Aware Pruning

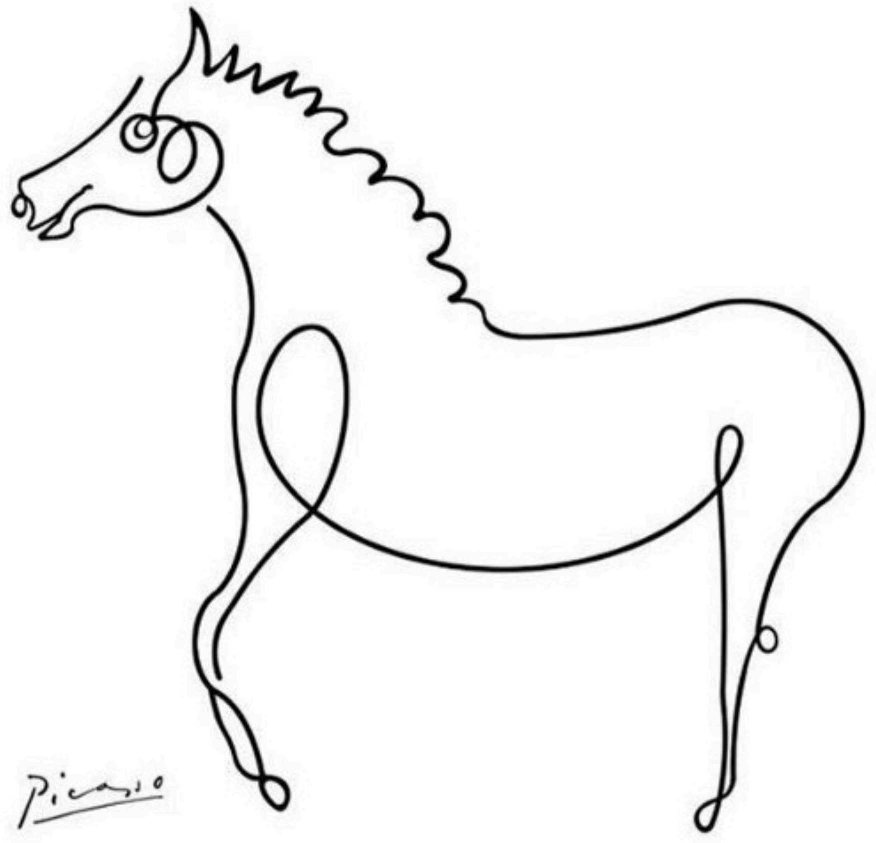
https://arxiv.org/abs/2102.11289
https://arxiv.org/abs/2304.06745

Efficient Algorithms



Compress it creatively:
knowledge distillation. e.g

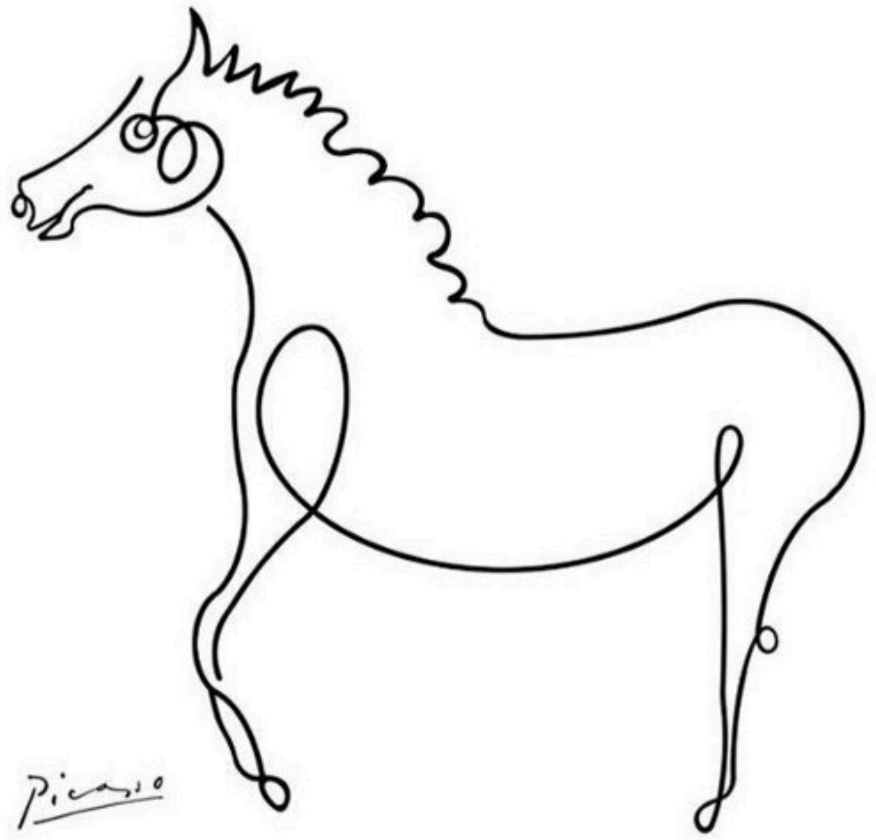
Efficient Algorithms



Neural Architecture search
e.g. EfficientNet for image detection

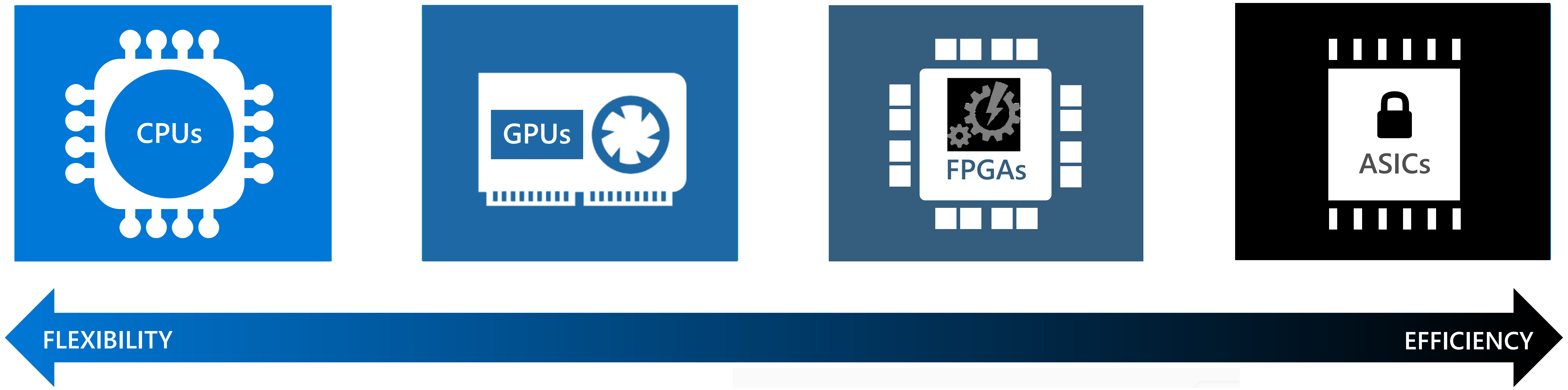
Efficient Algorithms





馬 马

Co-design tool for Specialized Hardware



L1 trigger

On-detector electronics

Co-design tool: crucial for prototyping AI at edge solutions

Algorithm hardware co-design for limited computing

Prototype with manageable programming barrier for domain scientists

Hardware Pros and Cons

More Flexible

Homogeneous

	Perf/W	
CPUs	1X	Today's standard, most programmable, good for services changing rapidly
Manycore CPUs	3X	Many simple cores (10s to 100s per chip), useful if software can be fine-grain parallel, difficult to maintain.
GPUs	5-30X	Good for data parallelism by merged threads (SIMD), High memory bandwidth, power hungry
FPGAs	5-30X	Most radical fully programmable option. Good for streaming/irregular parallelism. Power efficient but currently need to program in H/W languages.
Structured ASICS	20-100X	Lower-NRE ASICs with lower performance/efficiency. Includes domain-specific (programmable) accelerators.
Custom ASICs	> 100X	Highest efficiency. Highest NRE costs. Requires high volume. Good for functions in very widespread use that are stable for many years.

Specialized

More Efficient

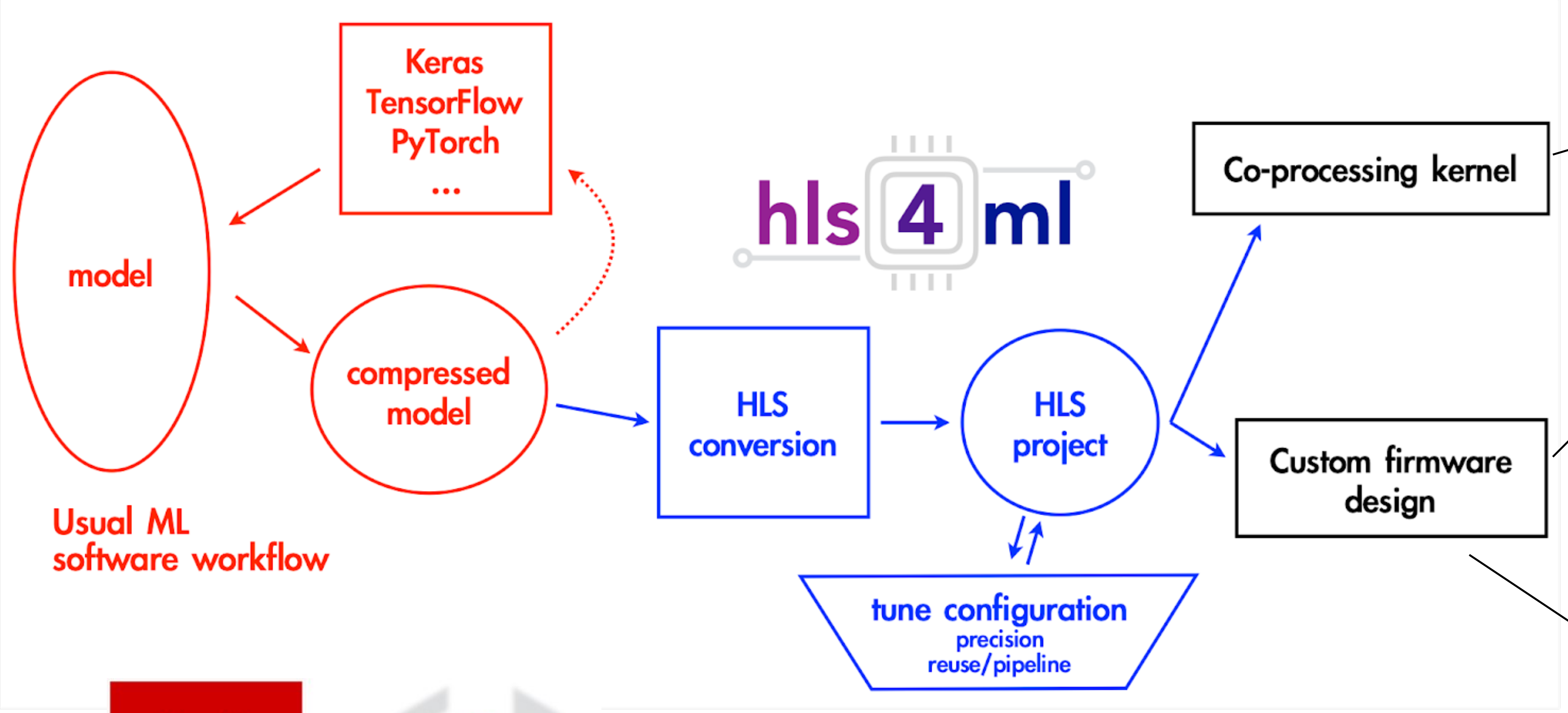
Conventional programming — C/C++

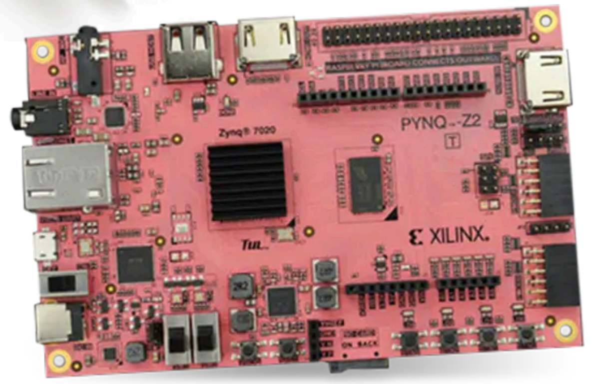
Alternative programming — CUDA, Verilog

Can't change functionality — Verilog

Co-design

Connecting domain scientists with prototype solutions





VivadoAccelerator
for Xilinx
accelerators / SoCs

(Q) Keras + TensorFlow

(Q) ONNX

PYTORCH

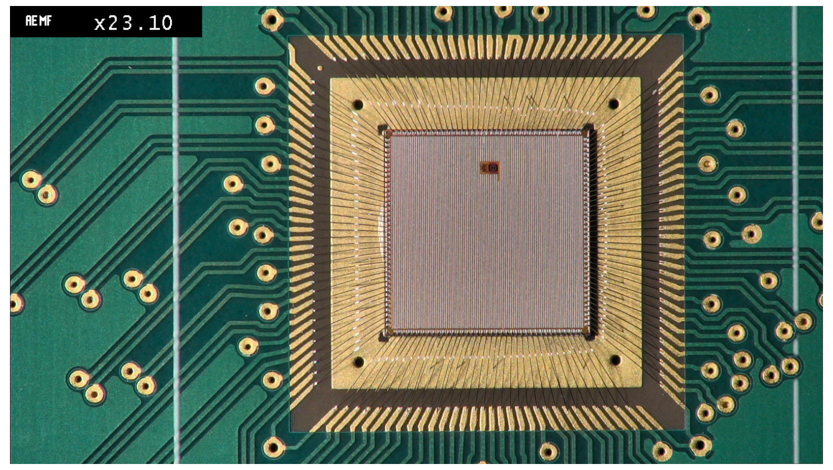
Vivado™ HLS

intel HLS COMPILER

Mentor® A Siemens Business

Catapult
Coming soon

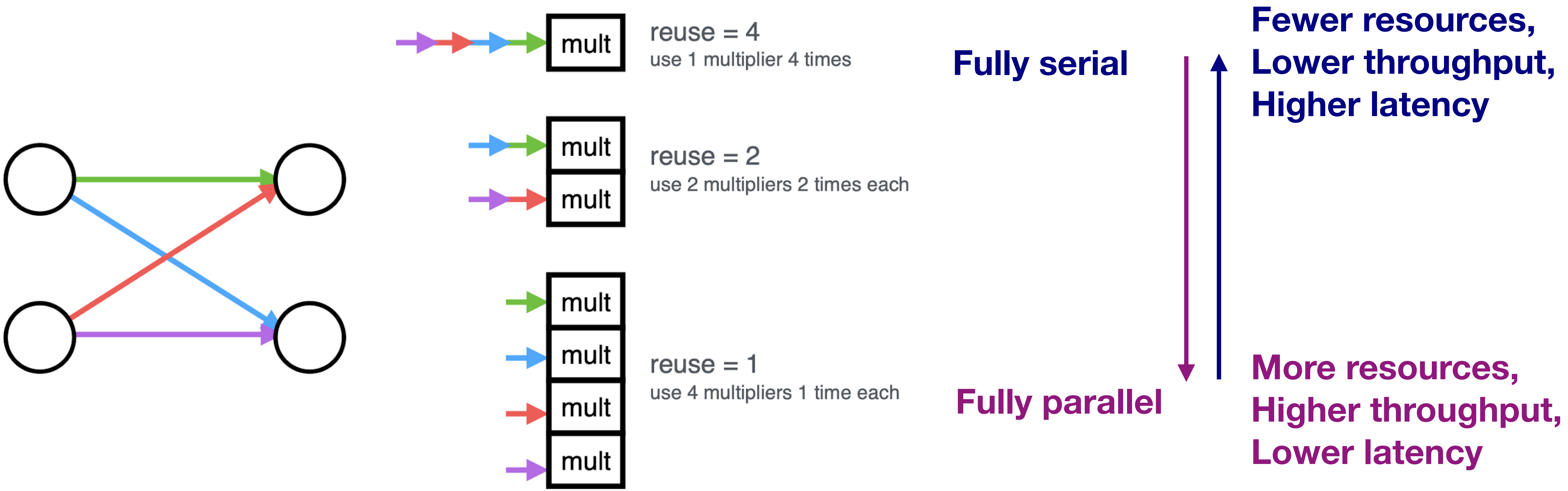
ASIC



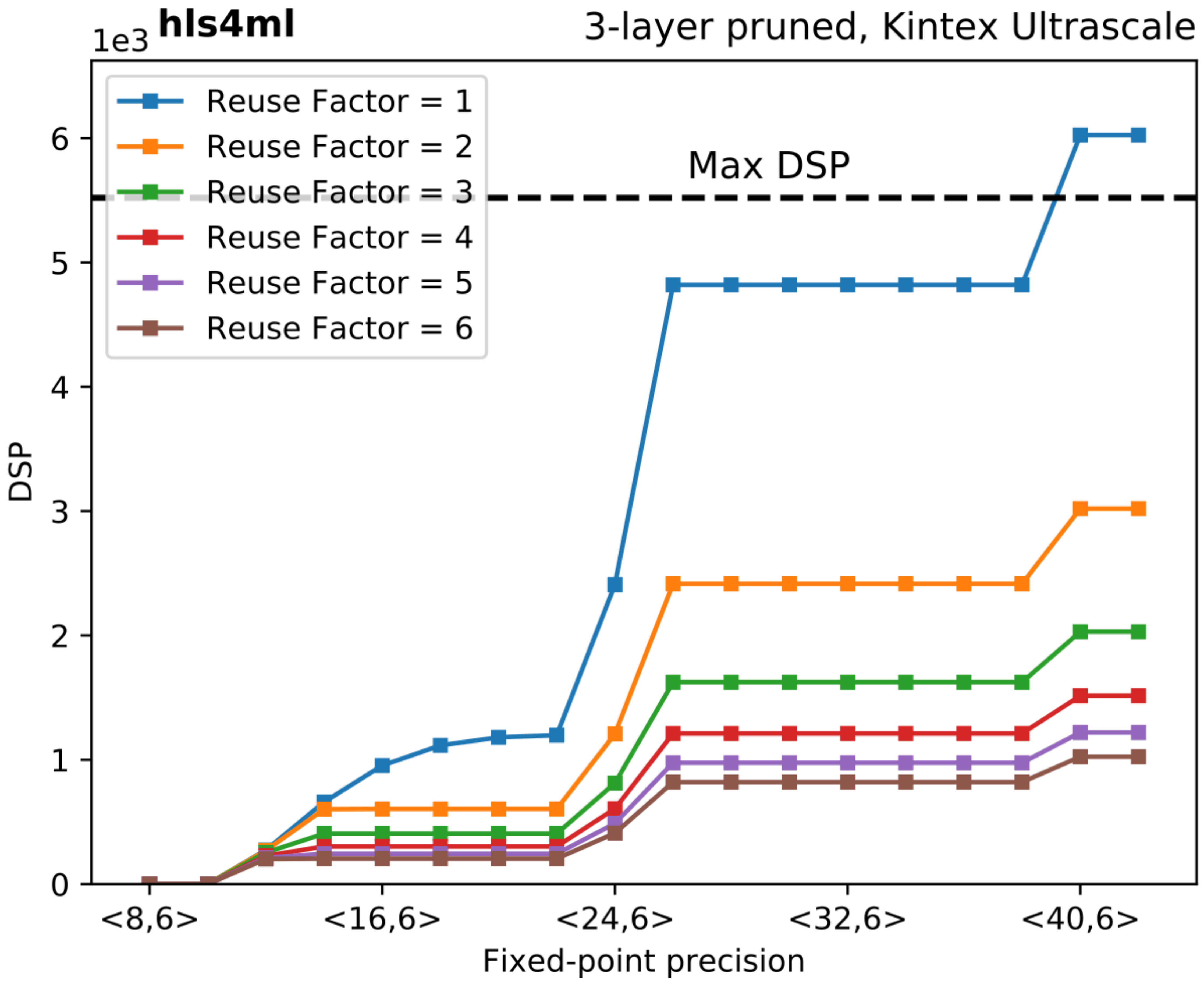
HLS4ML: to aid prototype science application solutions.

Example Knobs to tune: Reuse factor to balance latency and resources

- Trade-off between latency and FPGA resource usage determined by the parallelization of the calculations in each layer
- Configure the **“reuse factor”** = number of times a multiplier is used to do a computation



Reuse factor to balance latency and resources



75 ns

175 ns

More resources

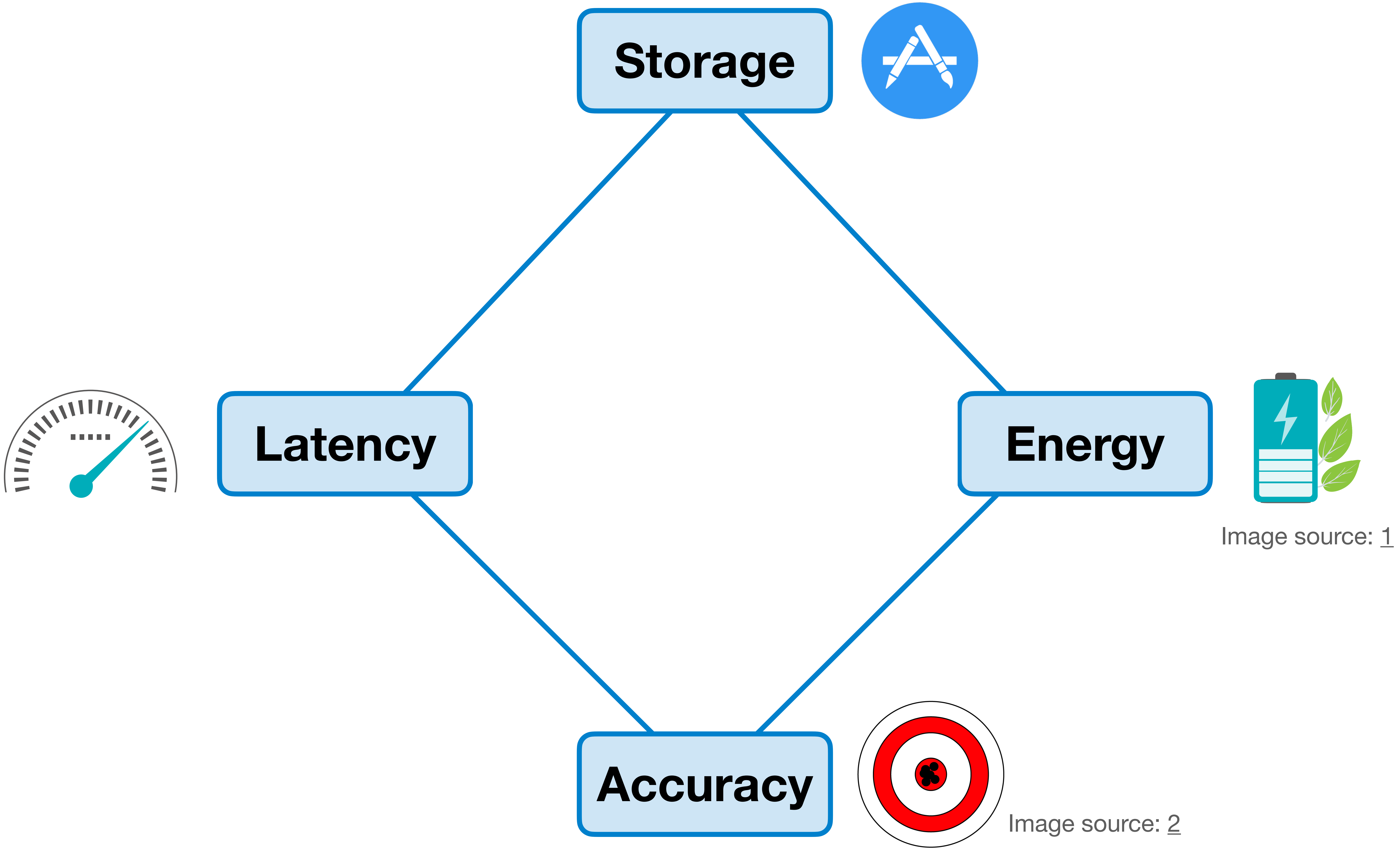
Fully parallel
Each mult. used 1x

Each mult. used 2x

Each mult. used 3x

Longer latency

Trade-off Between Efficiency and Accuracy



HLS4ML tutorial

hls4ml-tutorial: Tutorial notebooks for `hls4ml`

jupyter book | deploy-book passing | code style black | pre-commit enabled | launch binder

There are several ways to run the tutorial notebooks:

Online

launch binder

Conda

The Python environment used for the tutorials is specified in the `environment.yml` file. It can be setup like:

```
conda env create -f environment.yml
conda activate hls4ml-tutorial
```

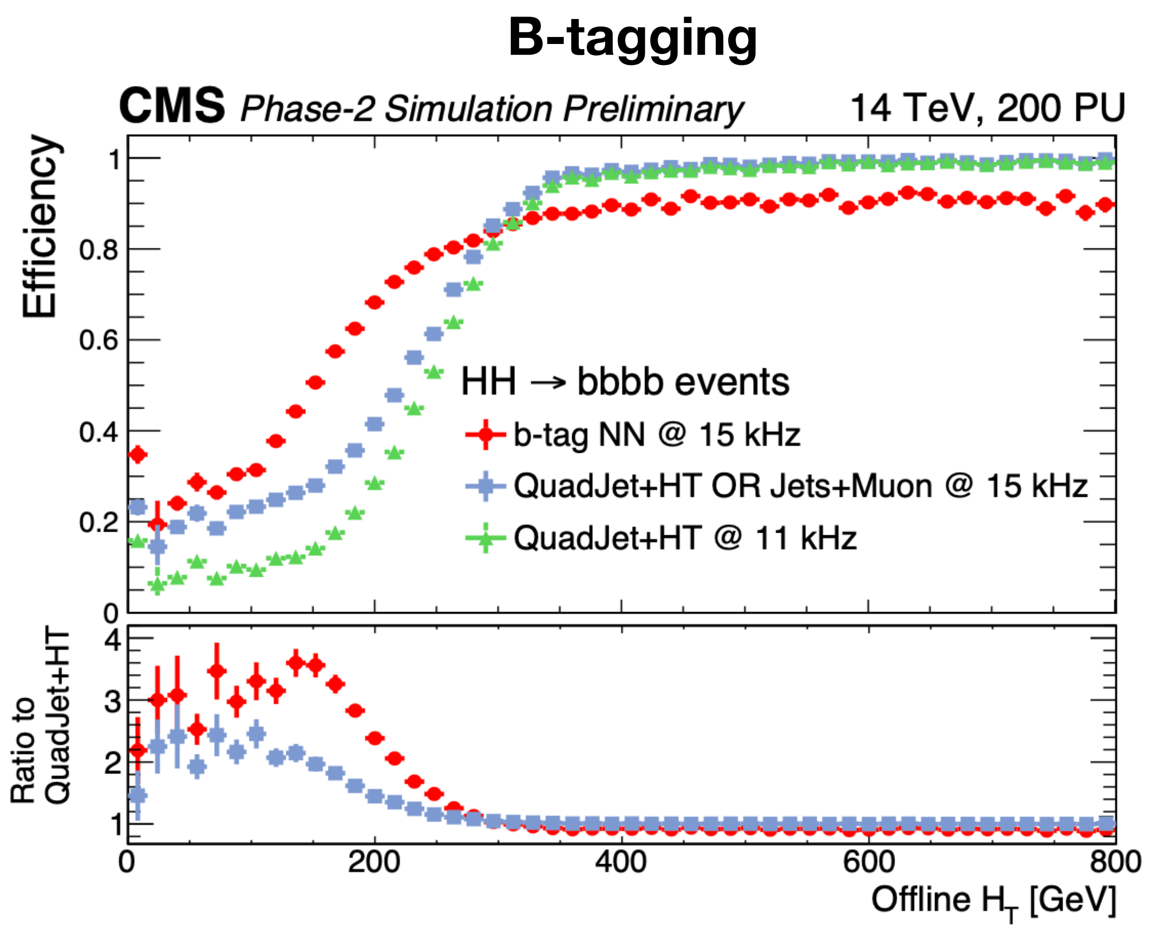
Docker without Vivado

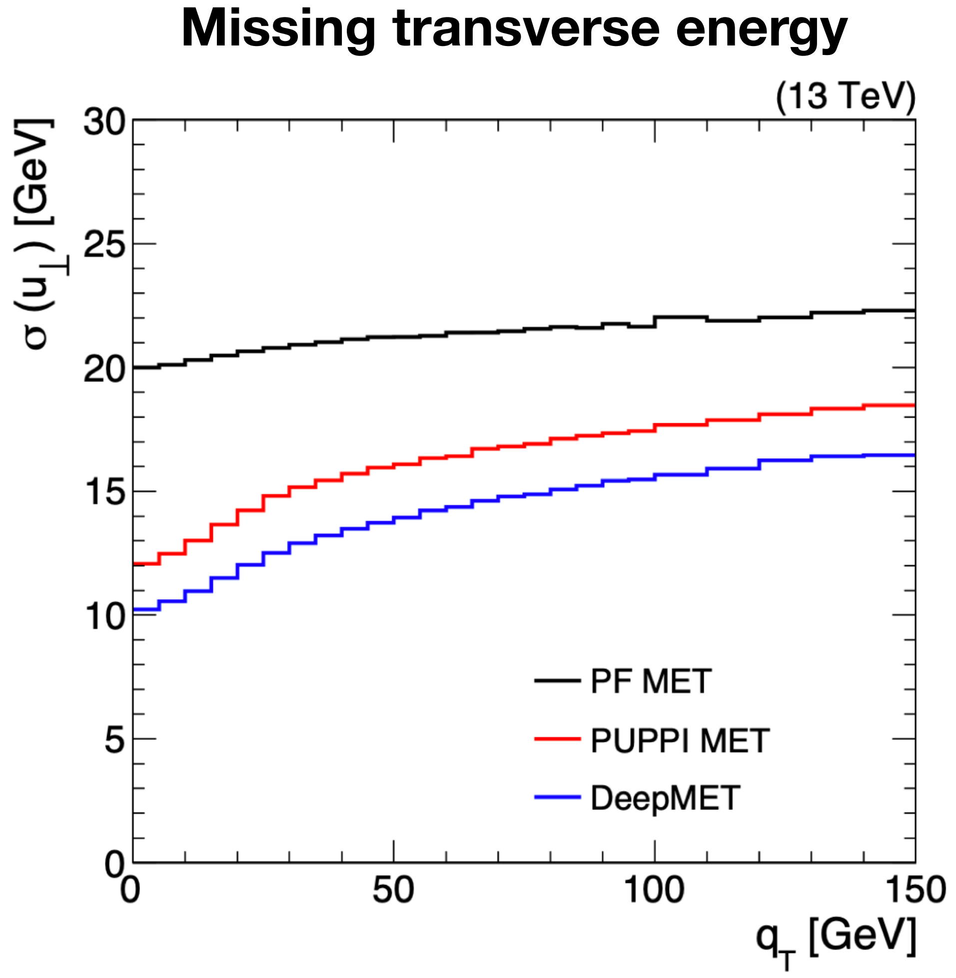
Pull the prebuilt image from the GitHub Container Registry:

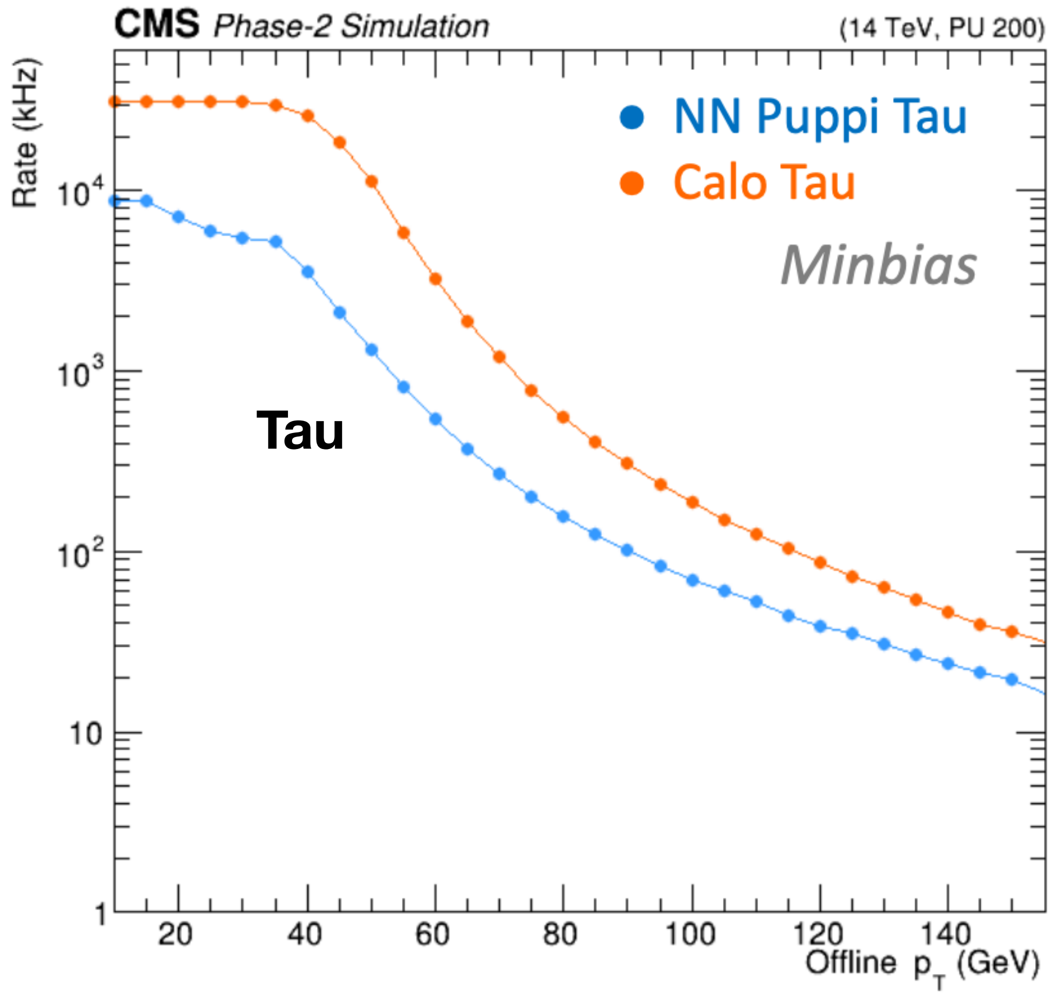
```
docker pull ghcr.io/fastmachinelearning/hls4ml-tutorial/hls4ml-0.8.0:latest
```

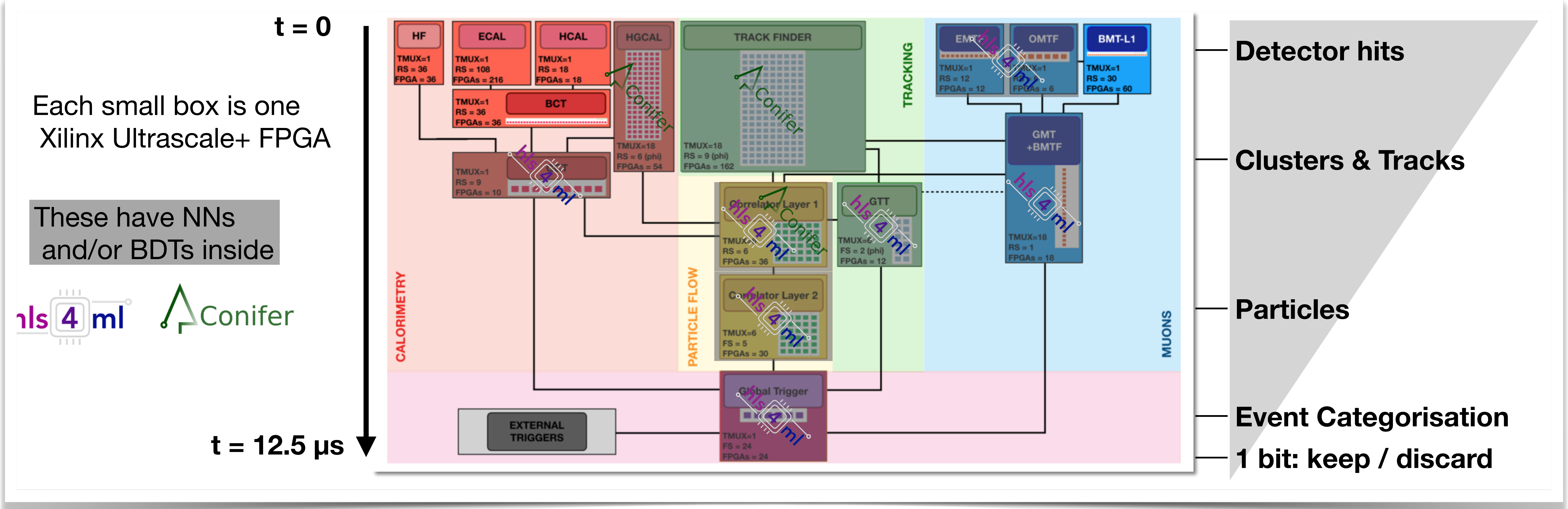
- https://github.com/fastmachinelearning/hls4ml-tutorial

ML everywhere in CMS Phase 2 L1 Trigger









HLS4ML: user driven development

- hls4ml 2023 roadmap plans new developments and a regular release schedule
- Q1 release: v0.7.0 & v0.7.1
  - Backend redesign to support multiple compilation targets [395]
  - Documentation updates [710, 744, 774]
  - Efficient network implementations [503, 509, 509]
  - Recurrent neural networks [560, 575]
  - Alveo accelerator FPGA card support [552]
  - Support for Vitis HLS [629]
  - Extension API [528]
- Q2 release: v0.8.0
  - Configuration editor [784]
  - PyTorch parsing improvements [799]
  - Symbolic expressions [660]
  - Optimization API [768, 809]
  - Large streaming CNN [PR Soon]
- Upcoming:
  - QONNX ingestion [591]
  - Catapult HLS

HLS4ML: user driven development

sPHENIX tracking GNN hls4ml synthesis results

Extremely preliminary - DO NOT TRUST NUMBERS

- Network inputs: nodes=80, edges=100
- Input network
  - Can be parallelized to be “nodes” times faster (i.e., 15ns)

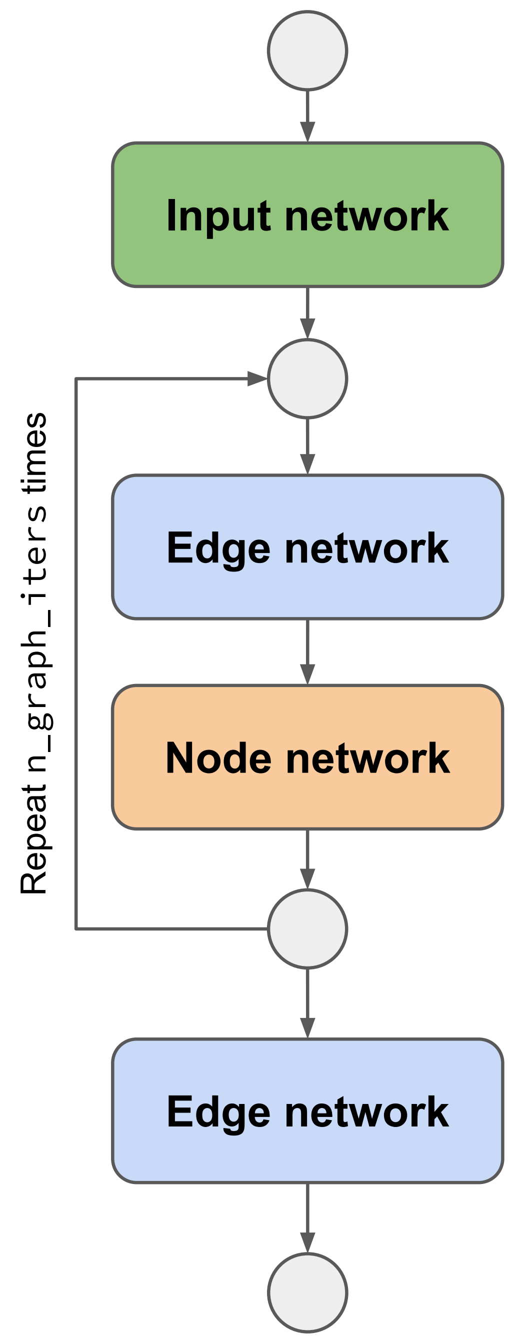
Latency	BRAMs	DSPs	FFs	LUTs
1.2 us	6.5%	0.3%	5%	7.5%

- Edge network

Latency	BRAMs	DSPs	FFs	LUTs
3 us	15%	2%	20%	65%

- Node network (results from HLS synthesis, vivado synthesis OOM’d)
  - Neet to optimize the scatter_add function (expecting ~2us for the net)

Latency	BRAMs	DSPs	FFs	LUTs
12 us	42%	7%	-	-



egular release schedule

- Configuration editor [784]
- PyTorch parsing improvements [799]
- Symbolic expressions [660]
- Optimization API [768, 809]
- Large streaming CNN [PR Soon]

Support GNN/transformers

More work needed for CMS applications (100 ns latency)

HLS4ML: user driven development

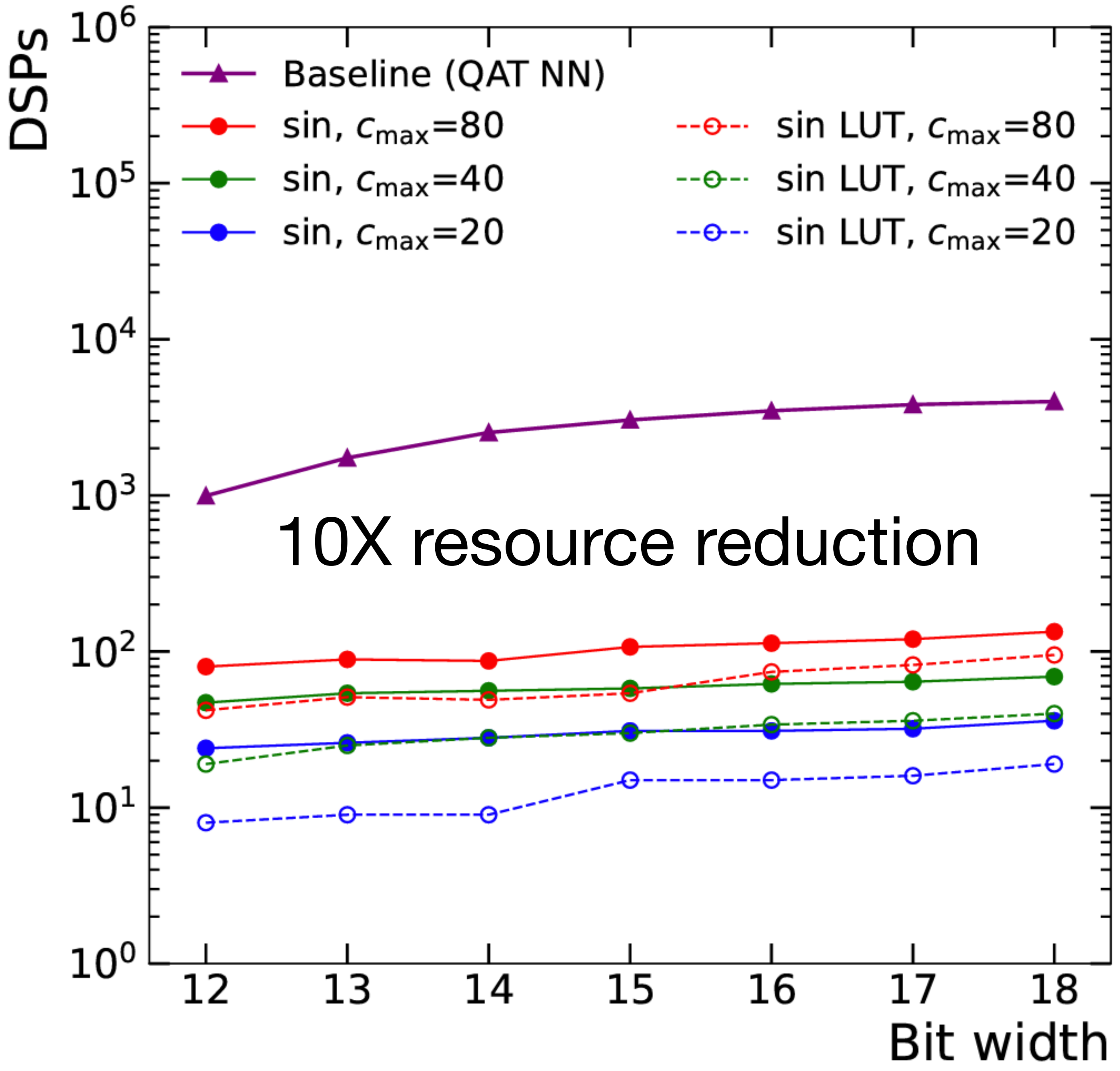
Symbolic Regression on FPGAs for Fast Machine Learning Inference

Ho Fung Tsoi[1*], *Adrian Alan* Pol[2], *Vladimir* Loncar[3,4], *Ekaterina* Govorkova[3], *Miles* Cranmer[2,5], *Sridhara* Dasu[1], *Peter* Elmer[2], *Philip* Harris[3], *Isobel* Ojalvo[2], and *Maurizio* Pierini[6]

[1]University of Wisconsin-Madison, USA
[2]Princeton University, USA
[3]Massachusetts Institute of Technology, USA
[4]Institute of Physics Belgrade, Serbia
[5]Flatiron Institute, USA
[6]European Organization for Nuclear Research (CERN), Switz

schedule

Bit width

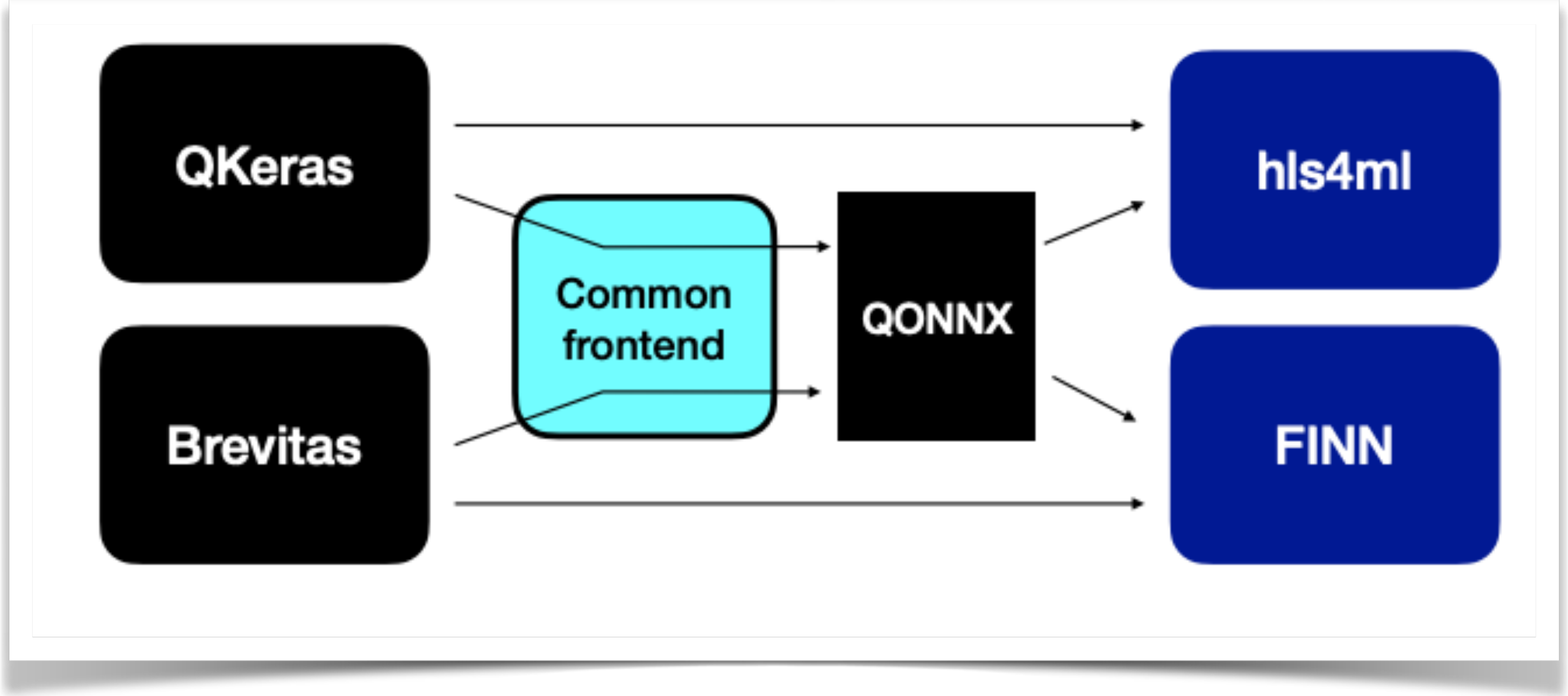


- PyTorch parsing improvements [799]
- Symbolic expressions [660]
- Optimization API [768, 809]
- Large streaming CNN [PR Soon]

HLS4ML: collaboration with XILINX FINN

- hls4ml 2023 roadmap plans new developments and a regular release schedule
- Q1 release: v0.7.0 & v0.7.1
  - Backend redesign to support multiple compilation targets [395]
  - Documentation updates [710, 744, 774]
  - Efficient network implementations [503, 509, 509]
  - Recurrent neural networks [560, 575]
  - Alveo accelerator FPGA card support [552]
  - Support for Vitis HLS [629]
  - Extension API [528]
- Q2 release: v0.8.0
  - Configuration editor [784]
  - PyTorch parsing improvements [799]
  - Symbolic expressions [660]
  - Optimization API [768, 809]
  - Large streaming CNN [PR Soon]

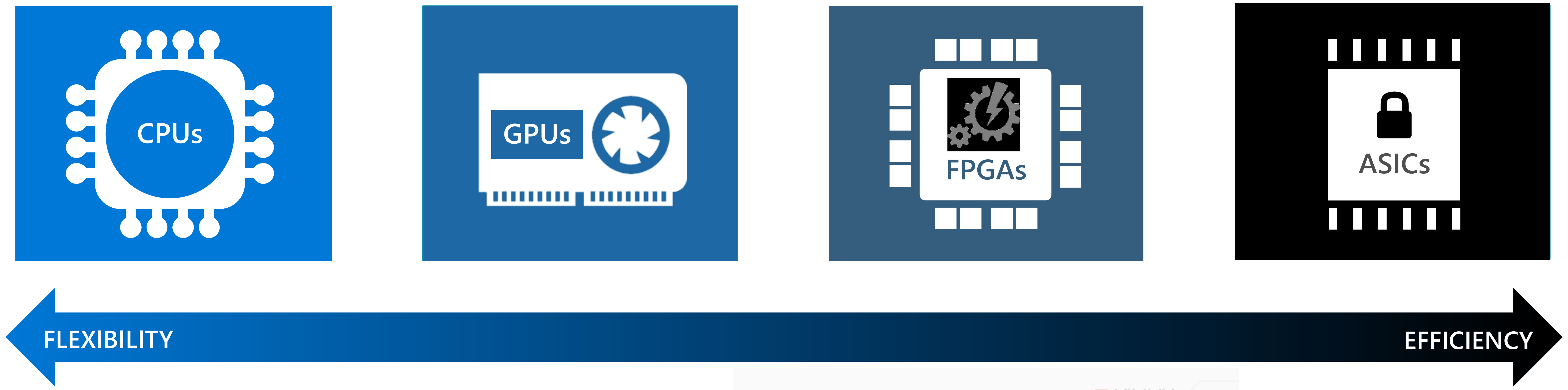
- Upcoming:
  - QONNX ingestion [591]
  - Catapult HLS

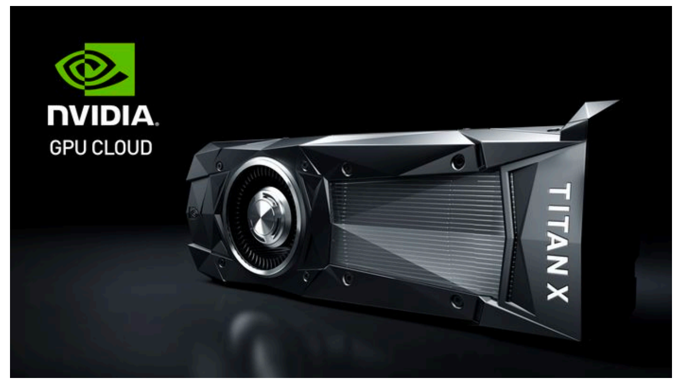


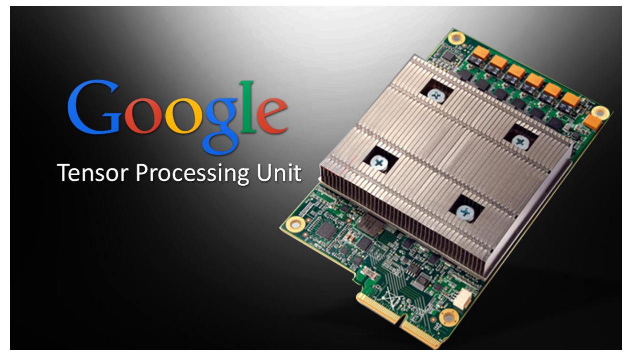
QONNX: Extension to the ONNX intermediate representation
format to represent arbitrary-precision quantized neural networks

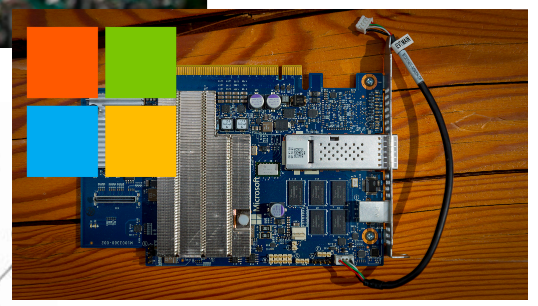
Heterogeneous computing

Heterogeneous Computing for ML/AI









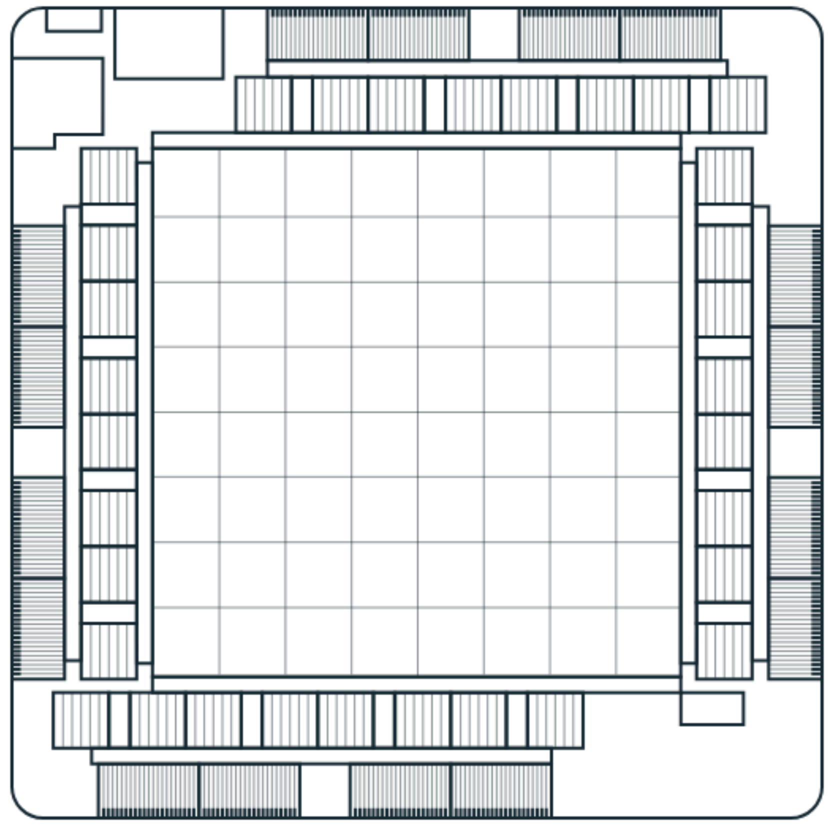
Advances driven by
big data explosion
& machine learning

A 5 year old slide, message remains…

Discontinued: October 18, 2022

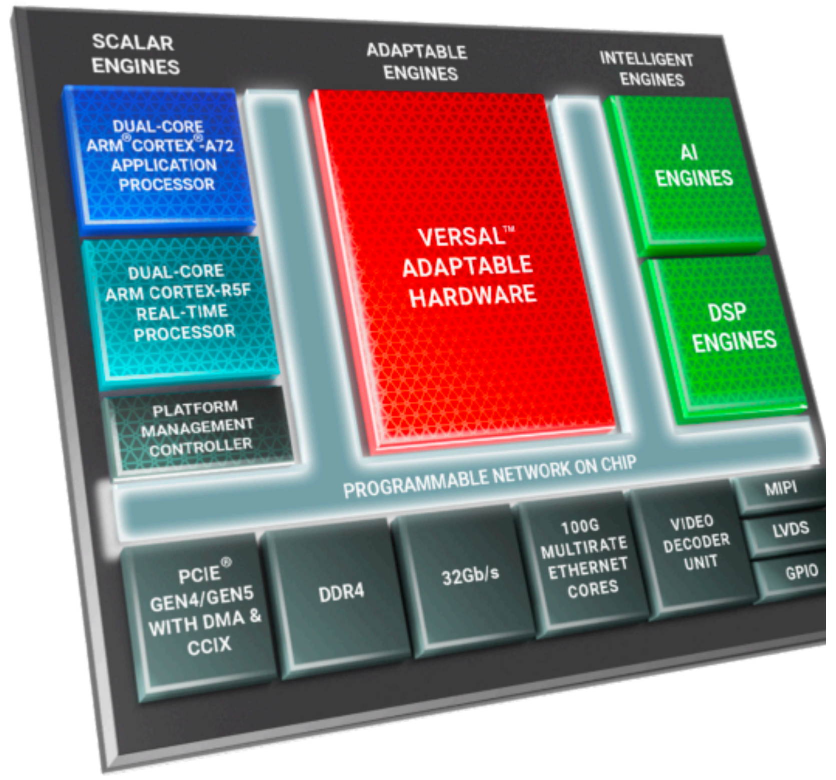
‘AI chips’ in 2023

Meta



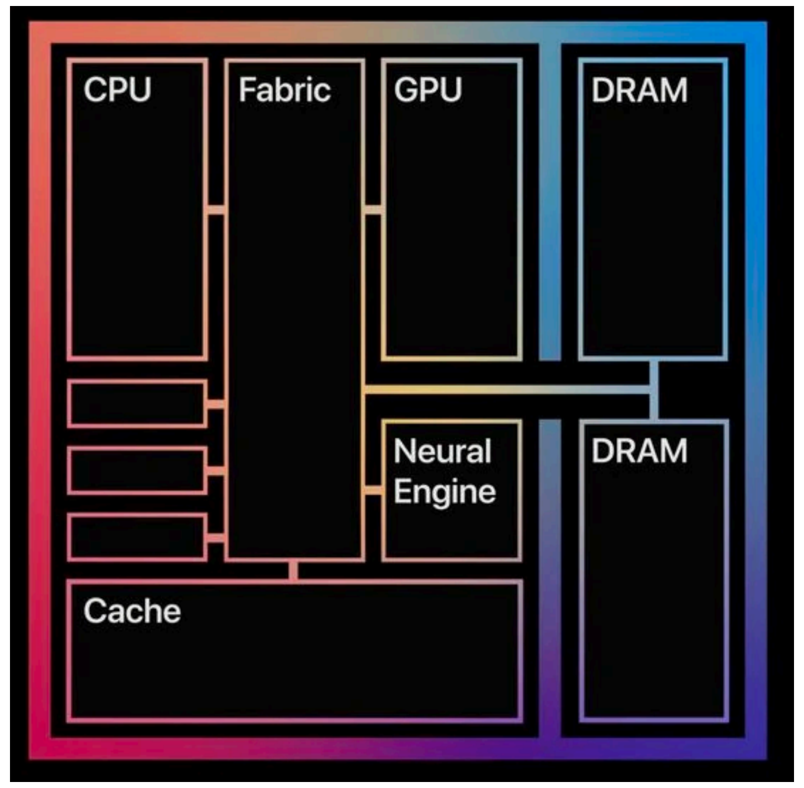
Groq

AMD / Xilinx

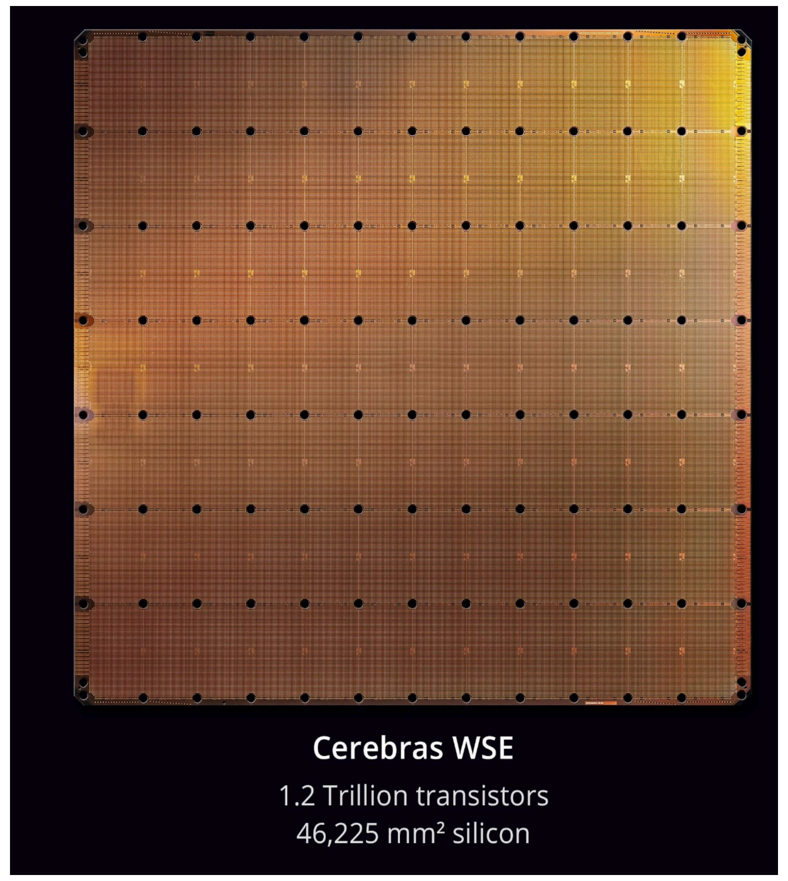


Graphcore

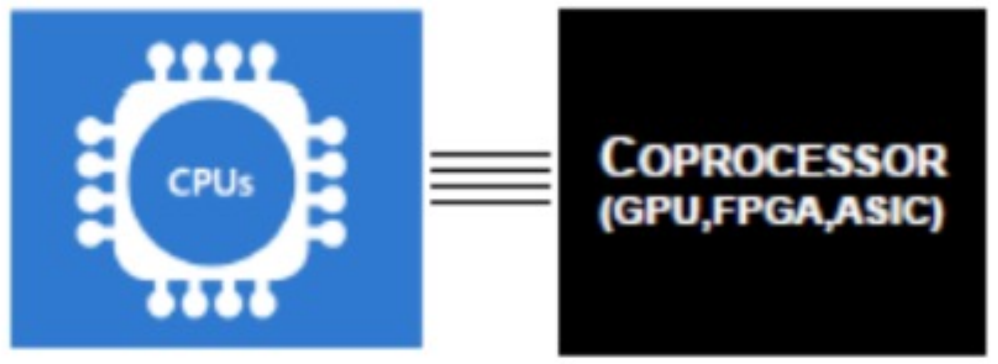
Apple

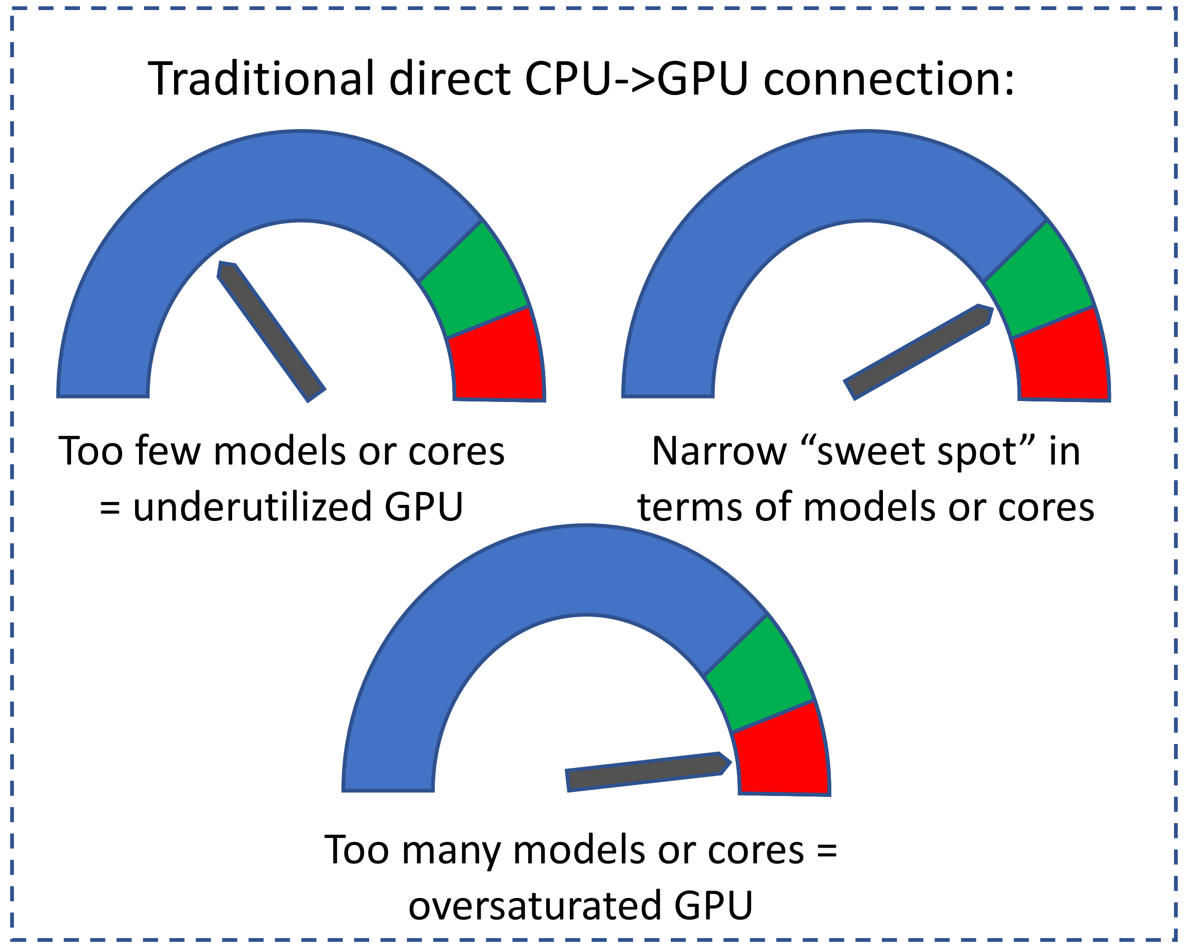


Cerebras

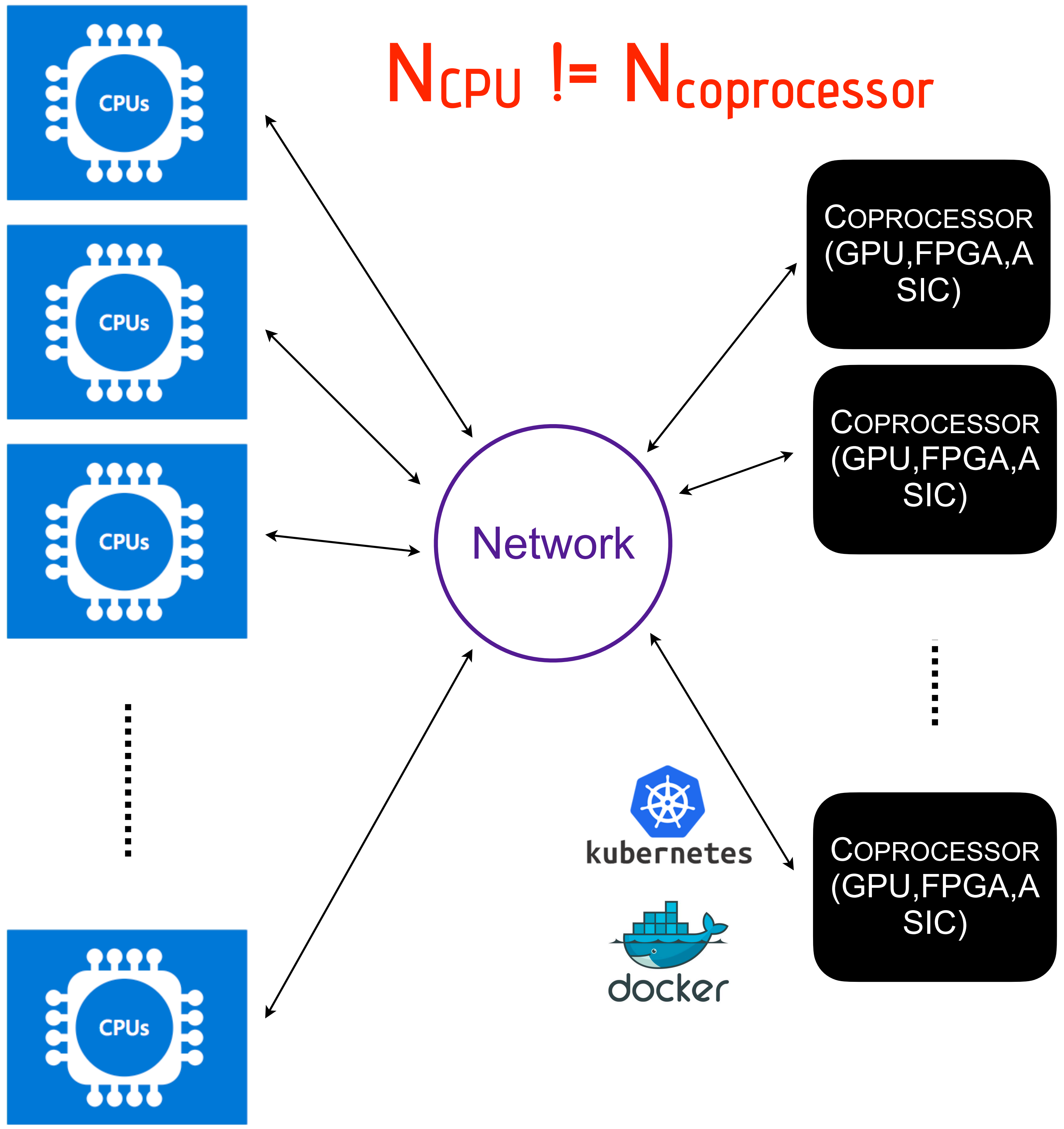


Optimal Acceleration hardware usage





Inflexible & Expensive



Complex, Requires R&D

Since 2018

GPU-as-a-service

FPGA-as-a-service Toolkit
('homegrown' gPRC server)

hls4ml

Hardware platforms

Open source tools

GPU-as-a-service for DUNE

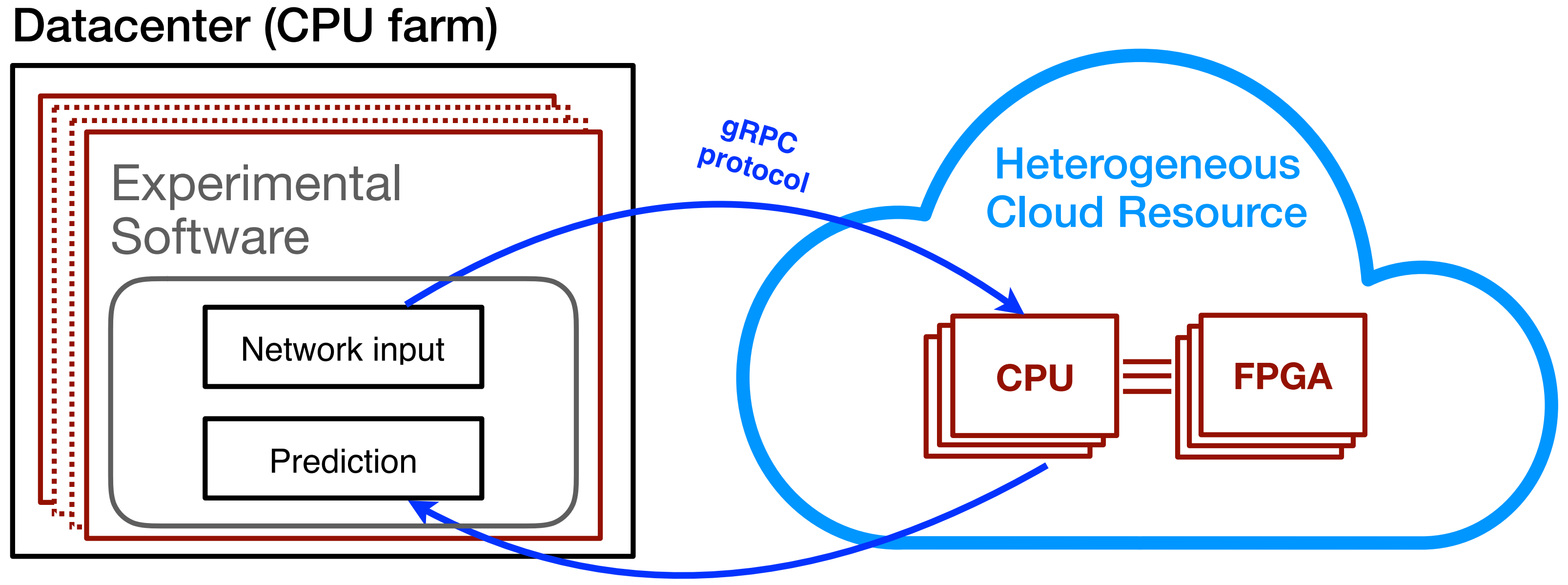
Deployment in experiments

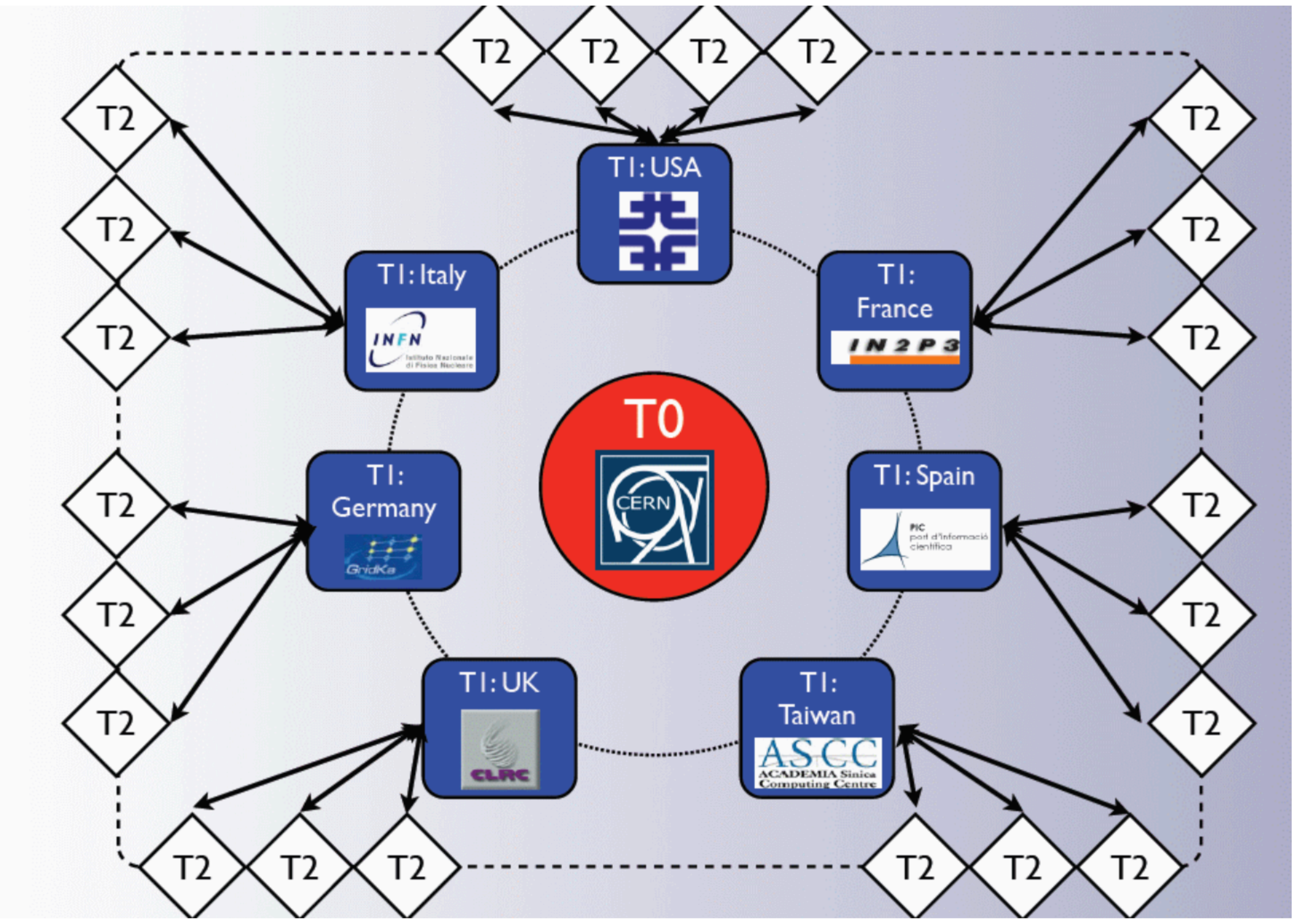
This talk: Portable Acceleration of CMS Production Workflow with Coprocessors as a Service

NVIDIA
Triton Inference Server

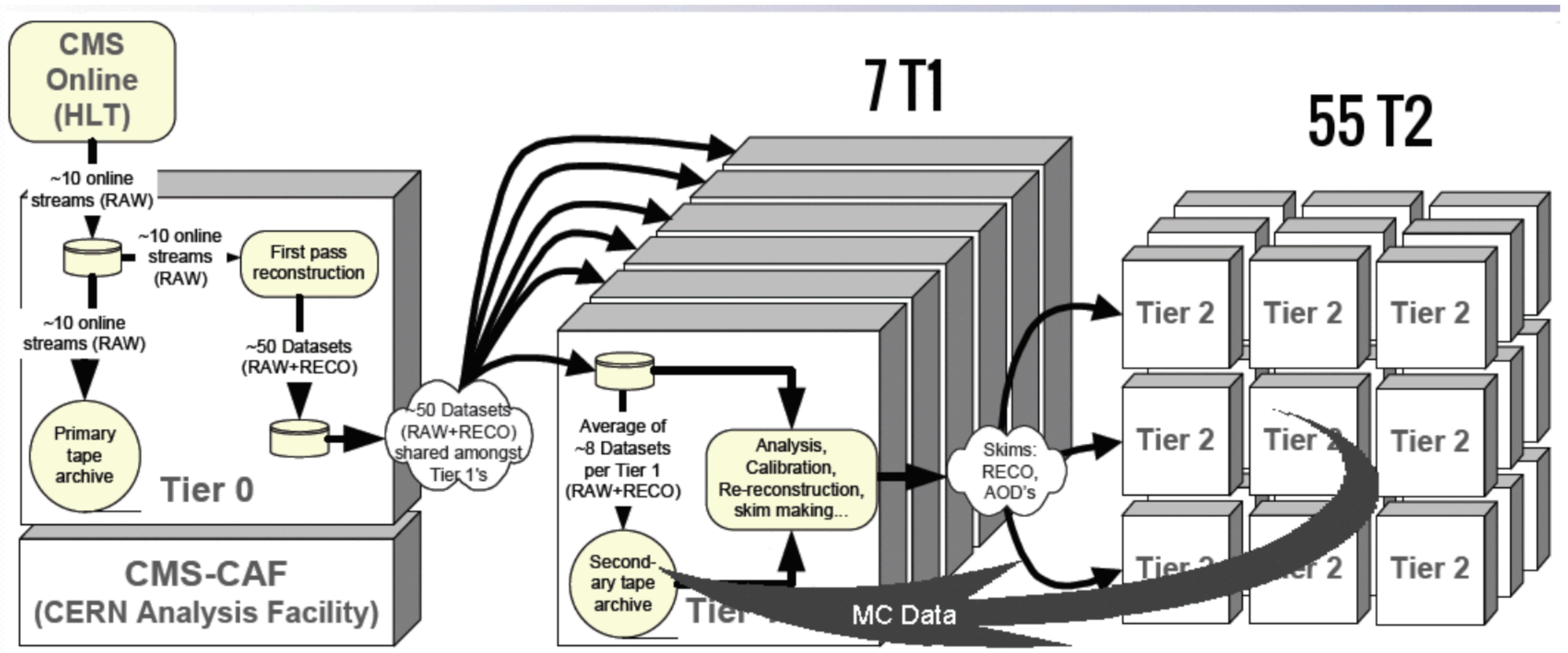
Graphcore

How to deploy SONIC in CMS





NVIDIA Triton Inference Server



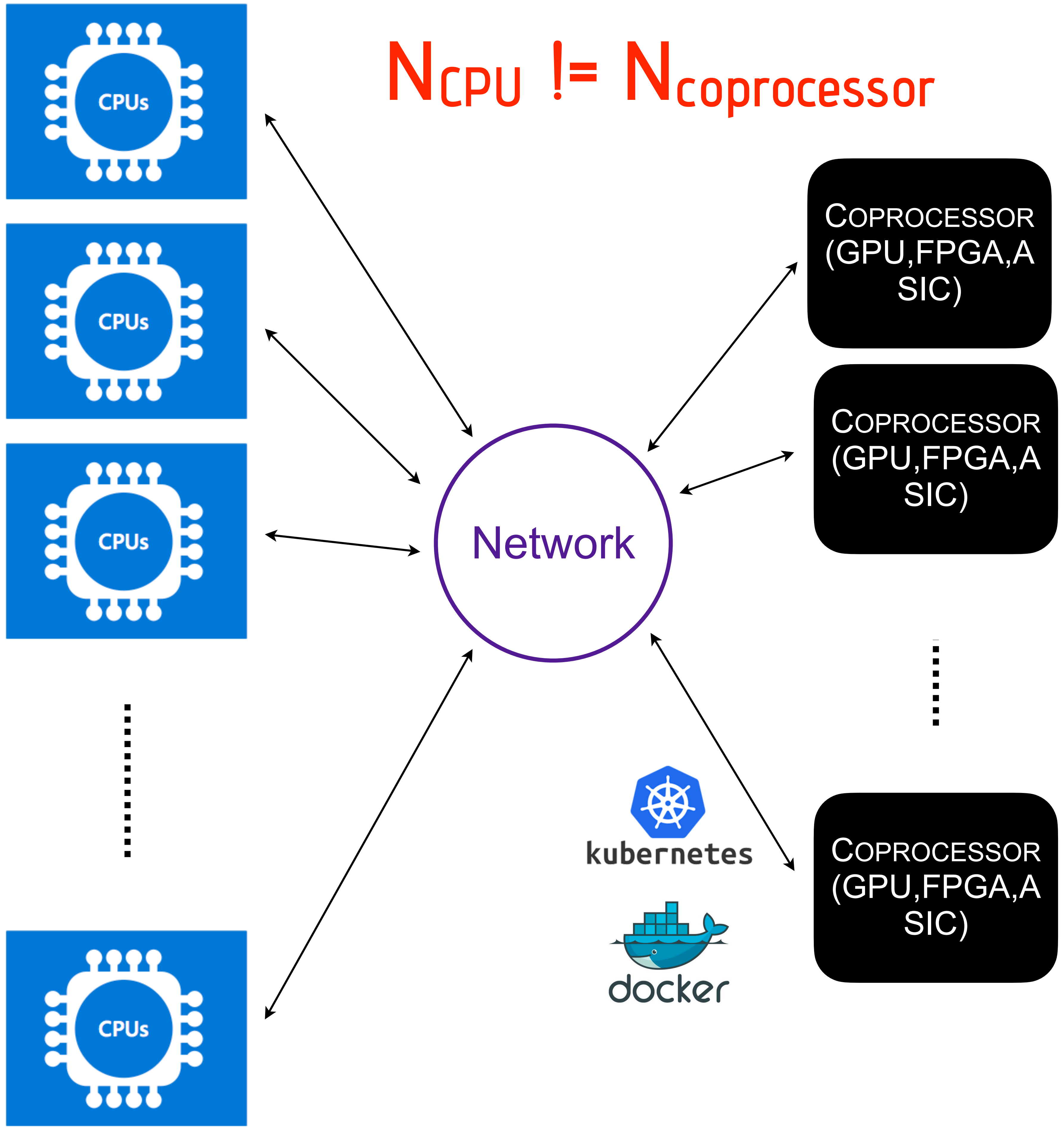
Alternative solution : as-a-service

Flexible - task-based optimization; software abstraction; low software maintenance overhead

Adaptable - right-size the system based on compute needs, maximize e.g. GPU acceleration

Scalable - co-processor disassociated from existing CPU infrastructure; common software framework

Non-disruptive - maintain HEP computing paradigm, coprocessors as an enhancement

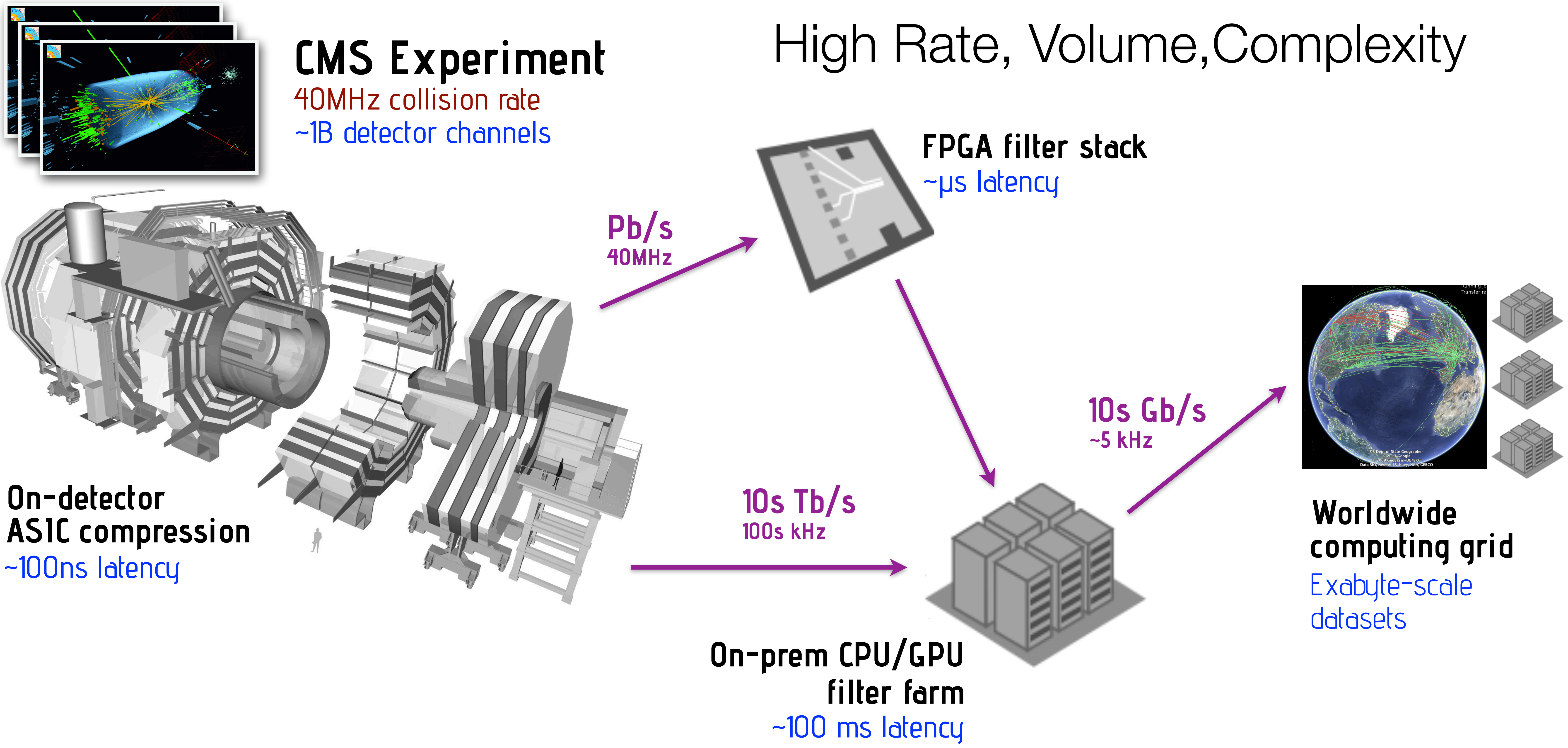


[First demonstration of integrating SONIC, with tests at Purdue CMS Tier-2 data center]

More details see [my talk at CPAD 2023].

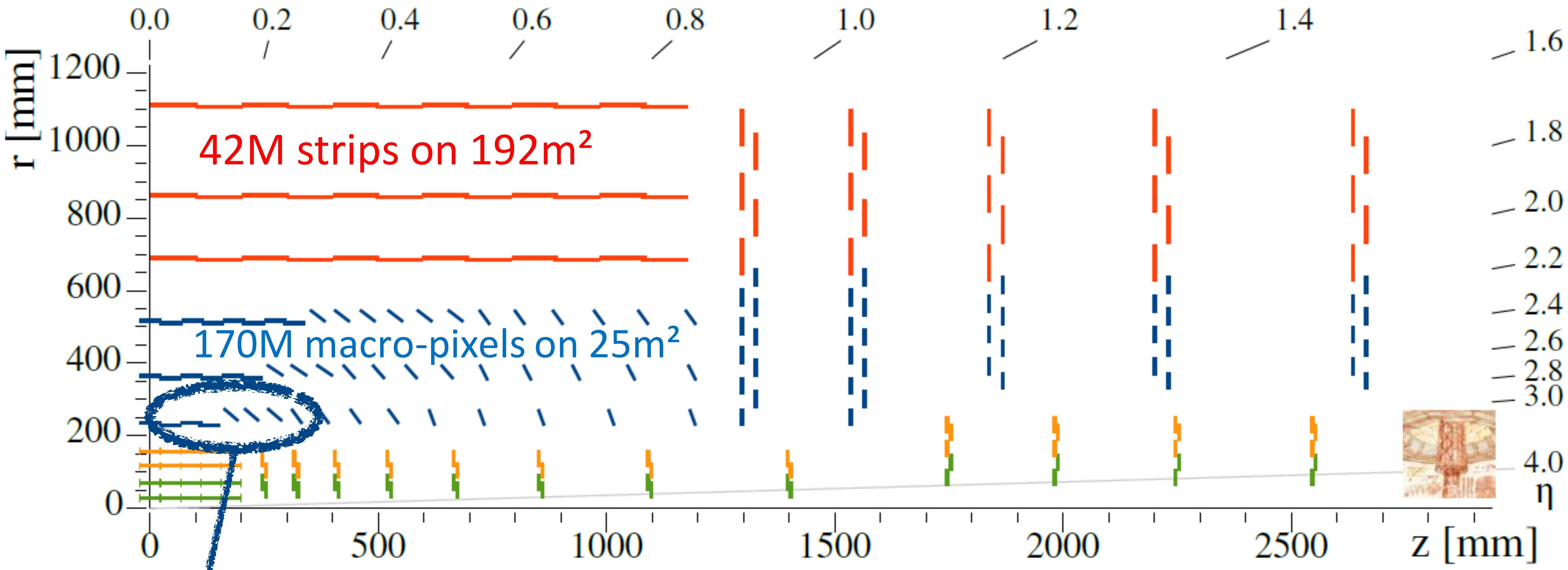
What else?

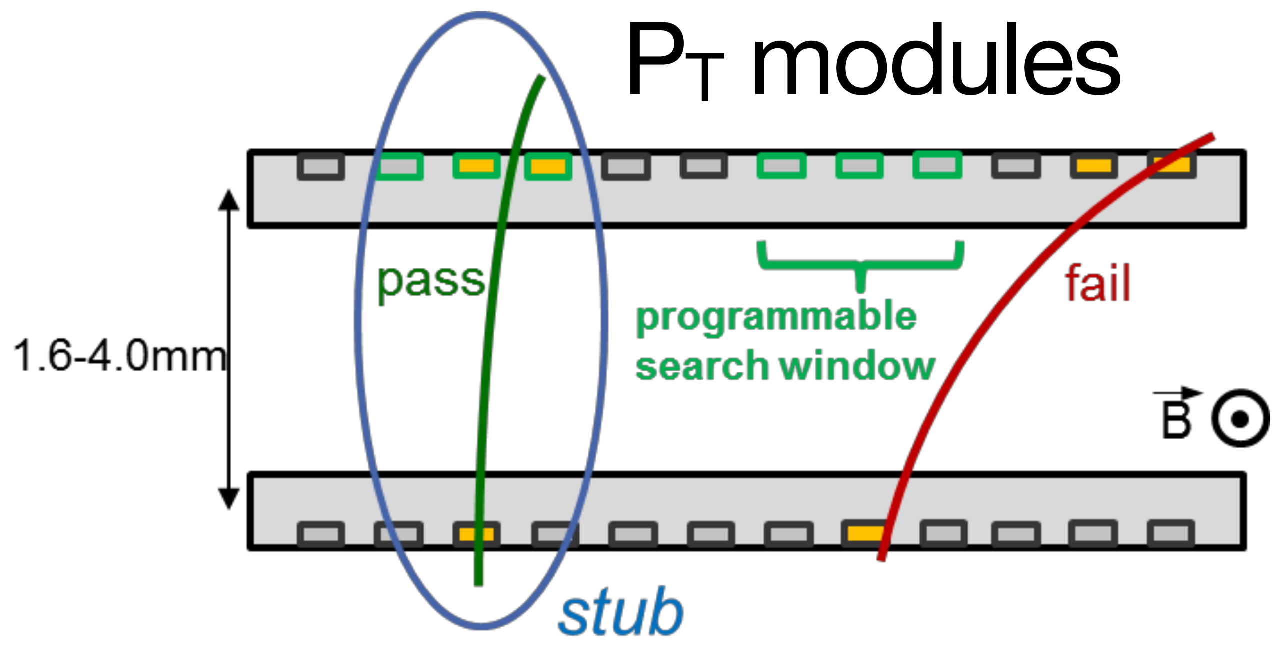
On-Chip?



Science with Big data: Multi-tier Data Processing

P_T modules to provide hardware trigger capabilities

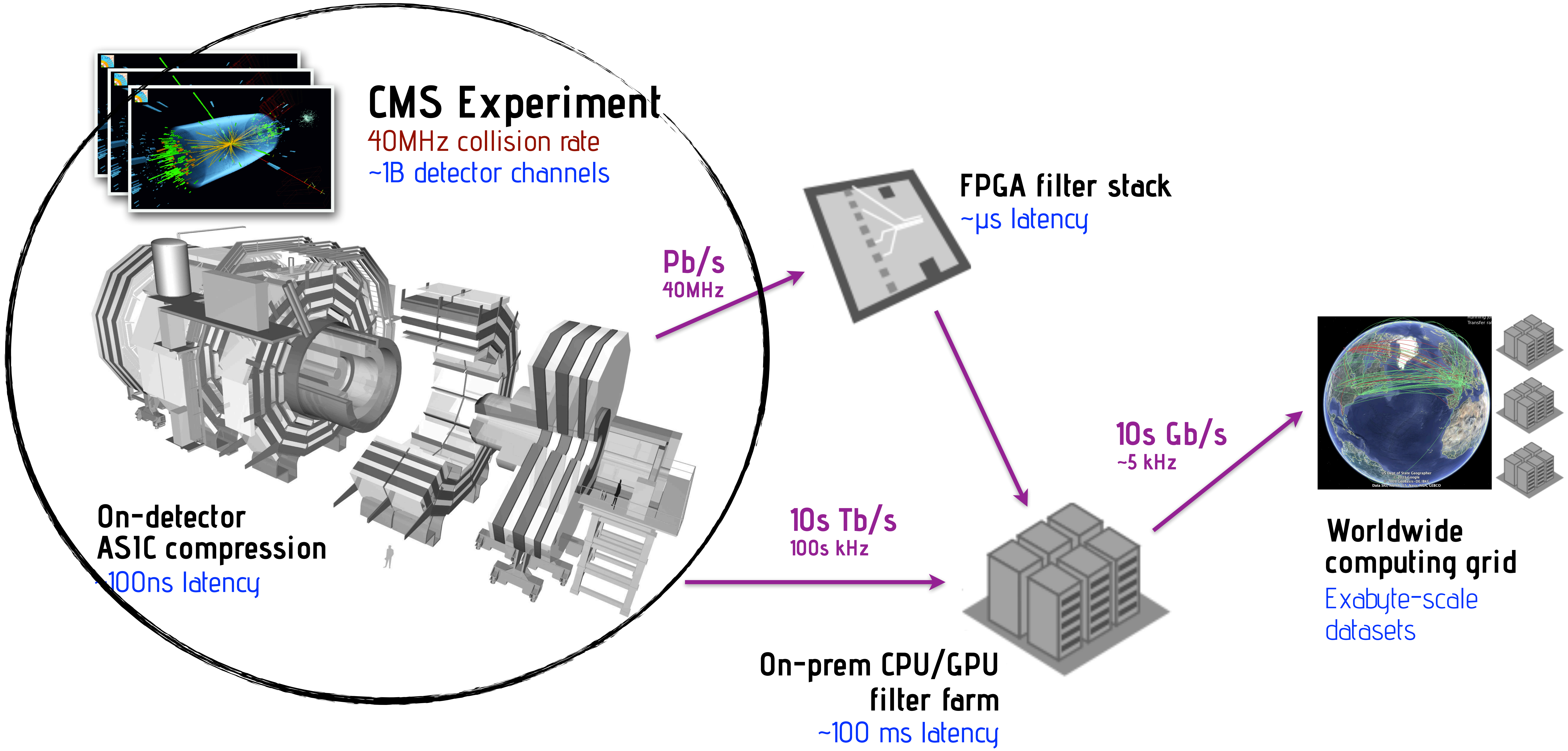




Designed to cope with high data rate, high radiation environment at the HL-LHC

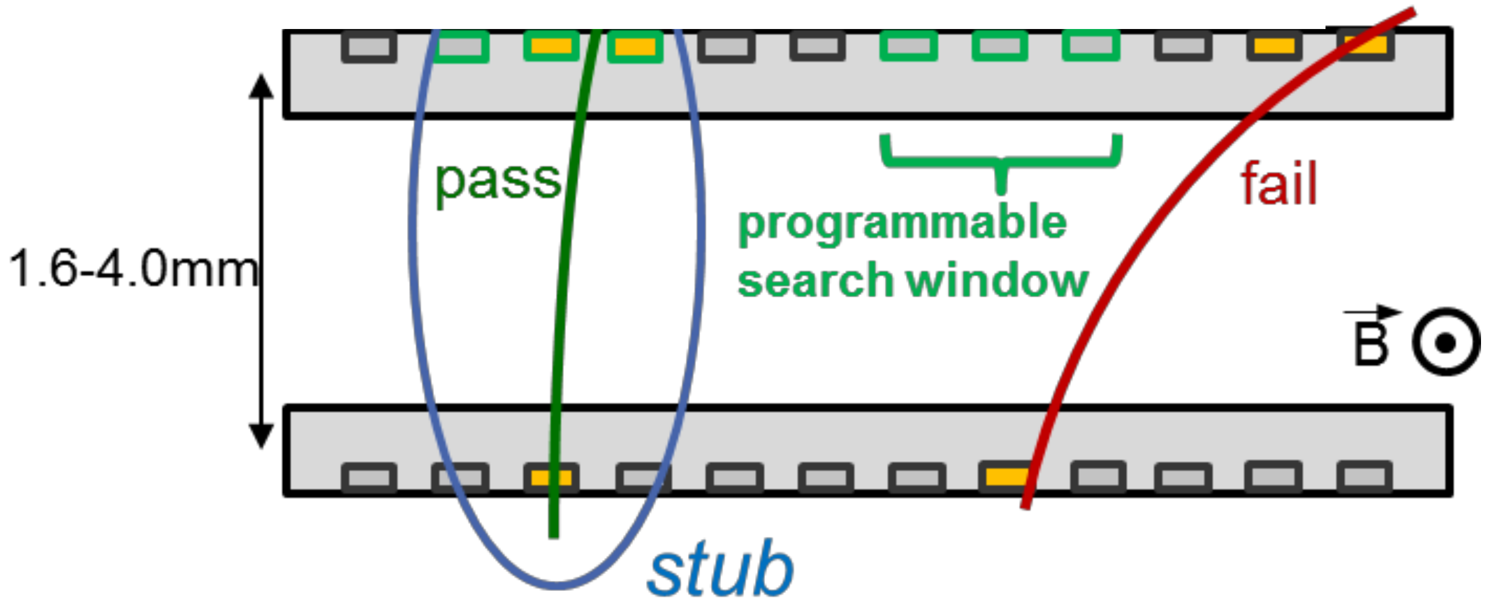
Higher granularity, Low material budget, titled geometry

‘Pt modules’ for Pixels?

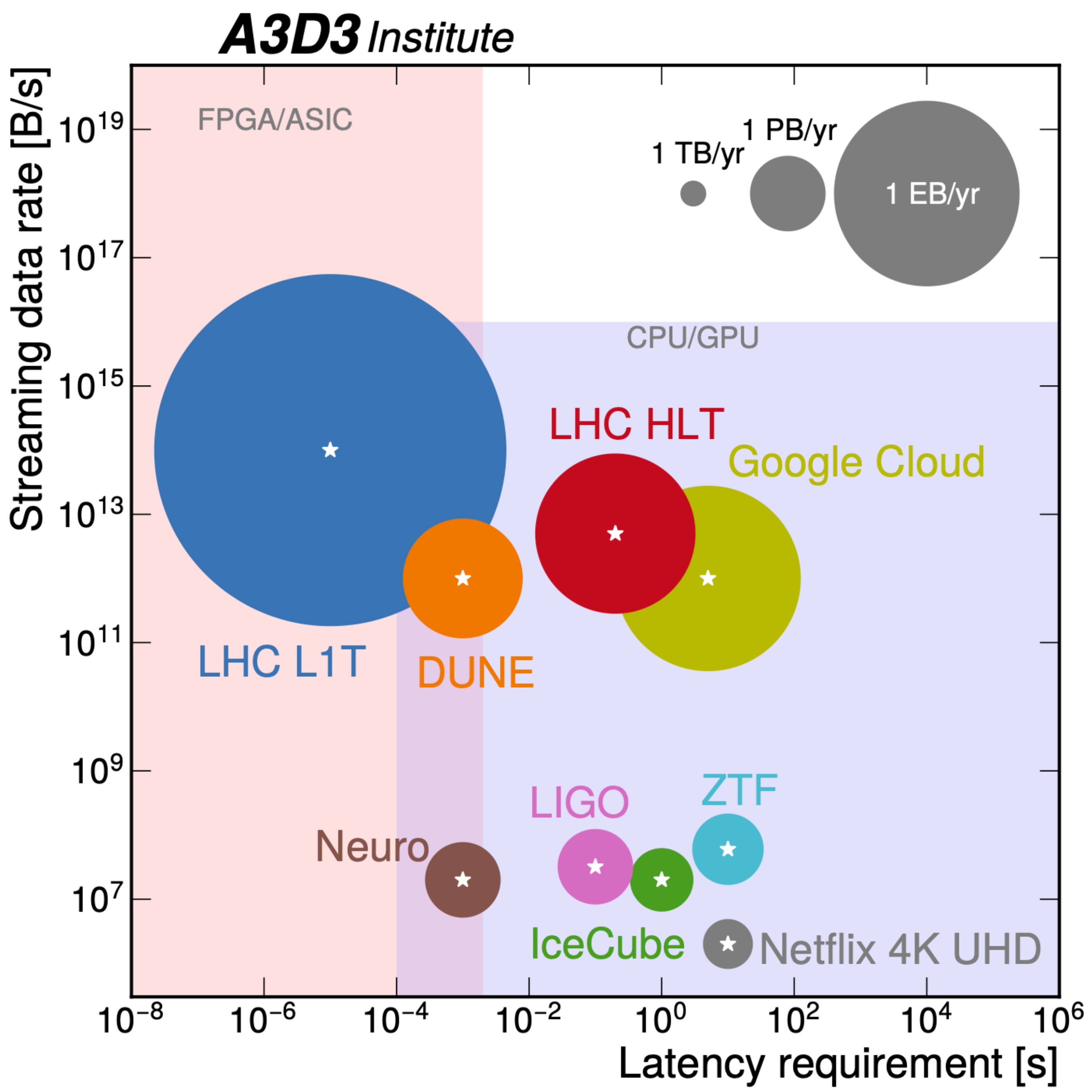


Enabled by <u>HLS4ML catapult</u>

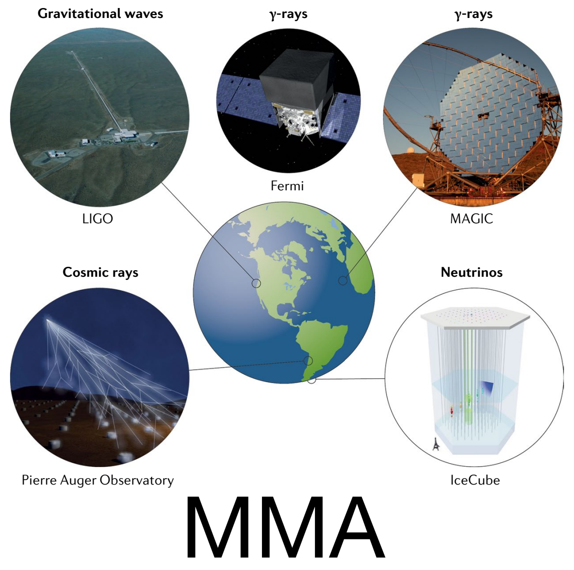
Use cluster information to infer particle kinematics

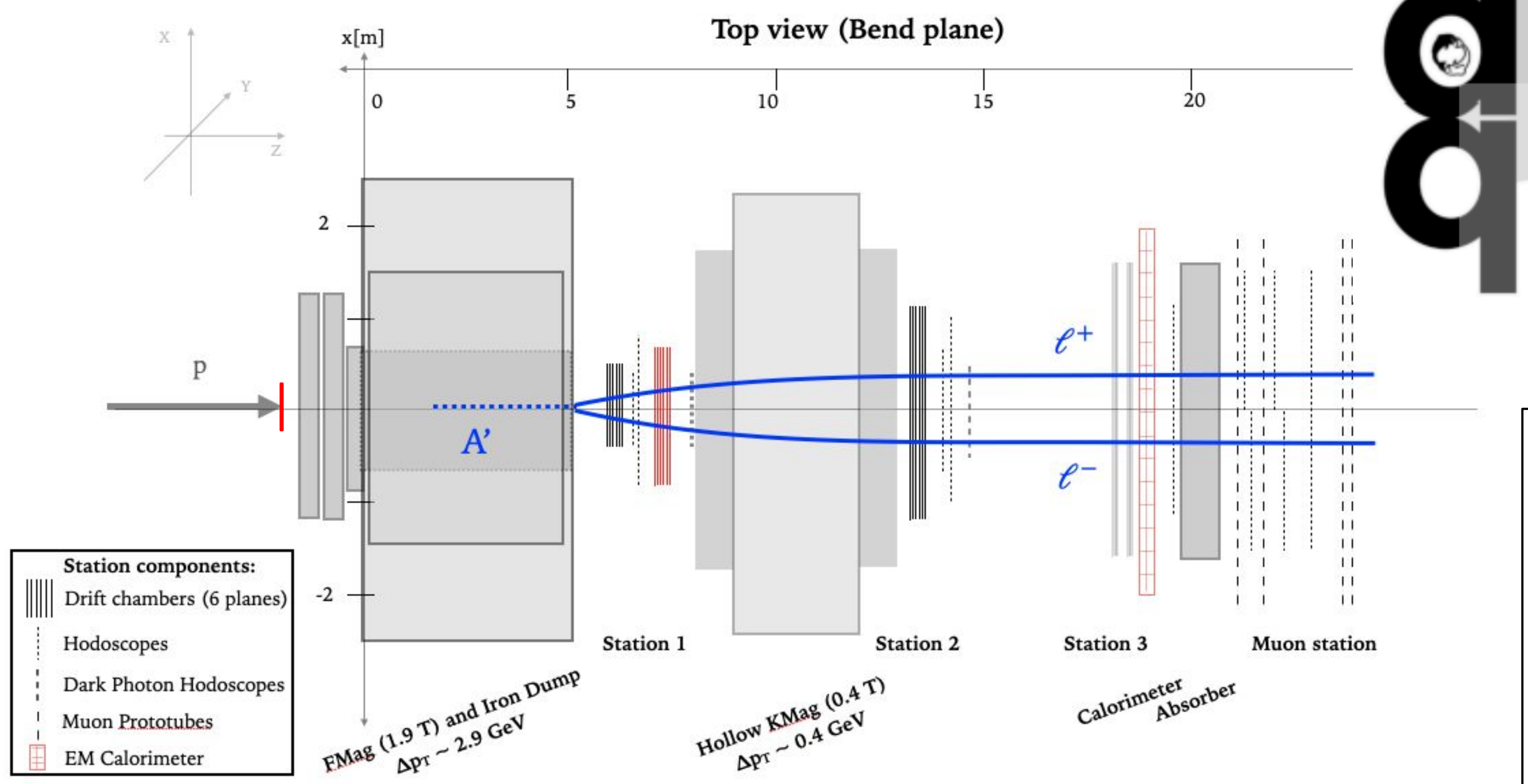


Accelerated AI Opportunities



Accelerated AI Algorithms for Data-Driven Discovery



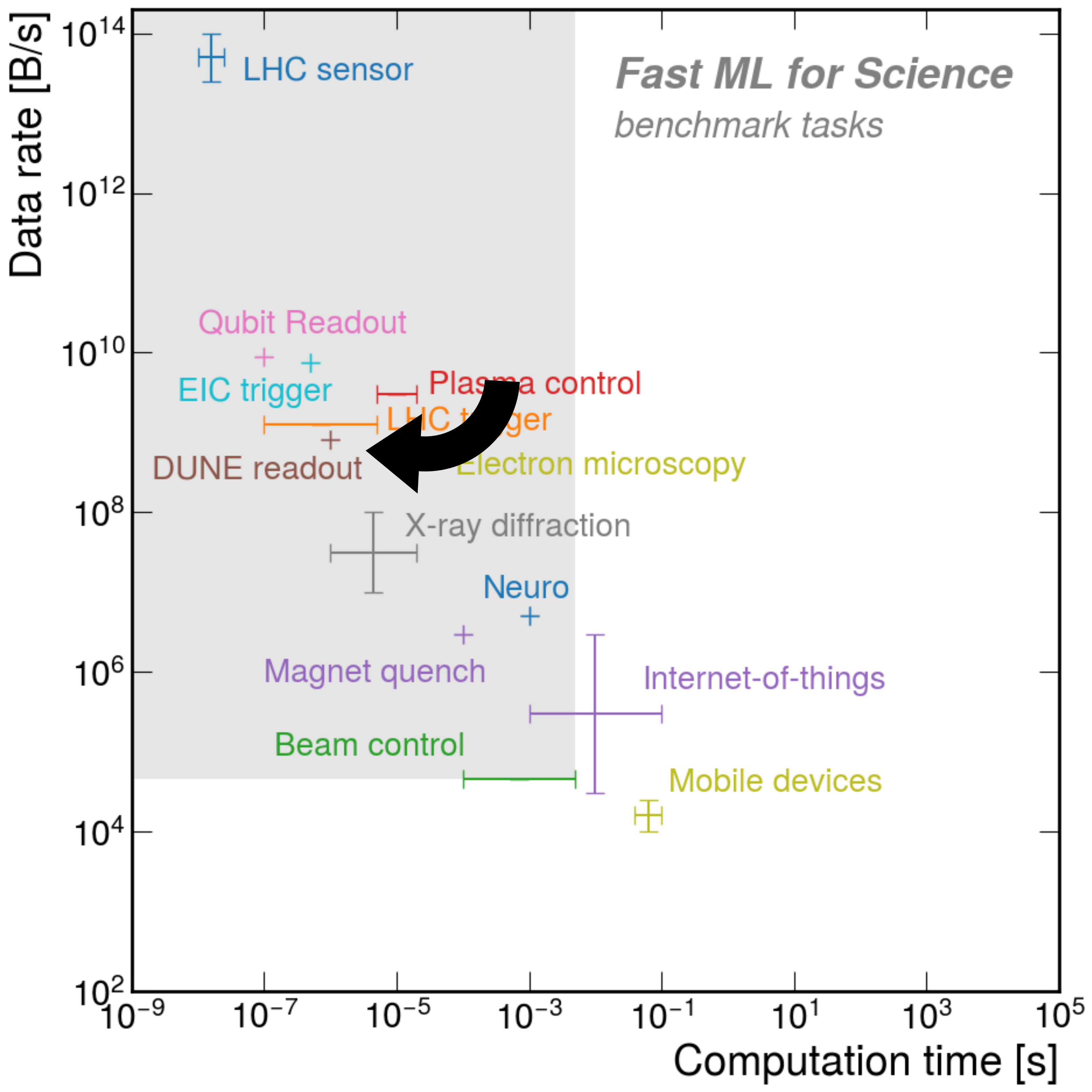


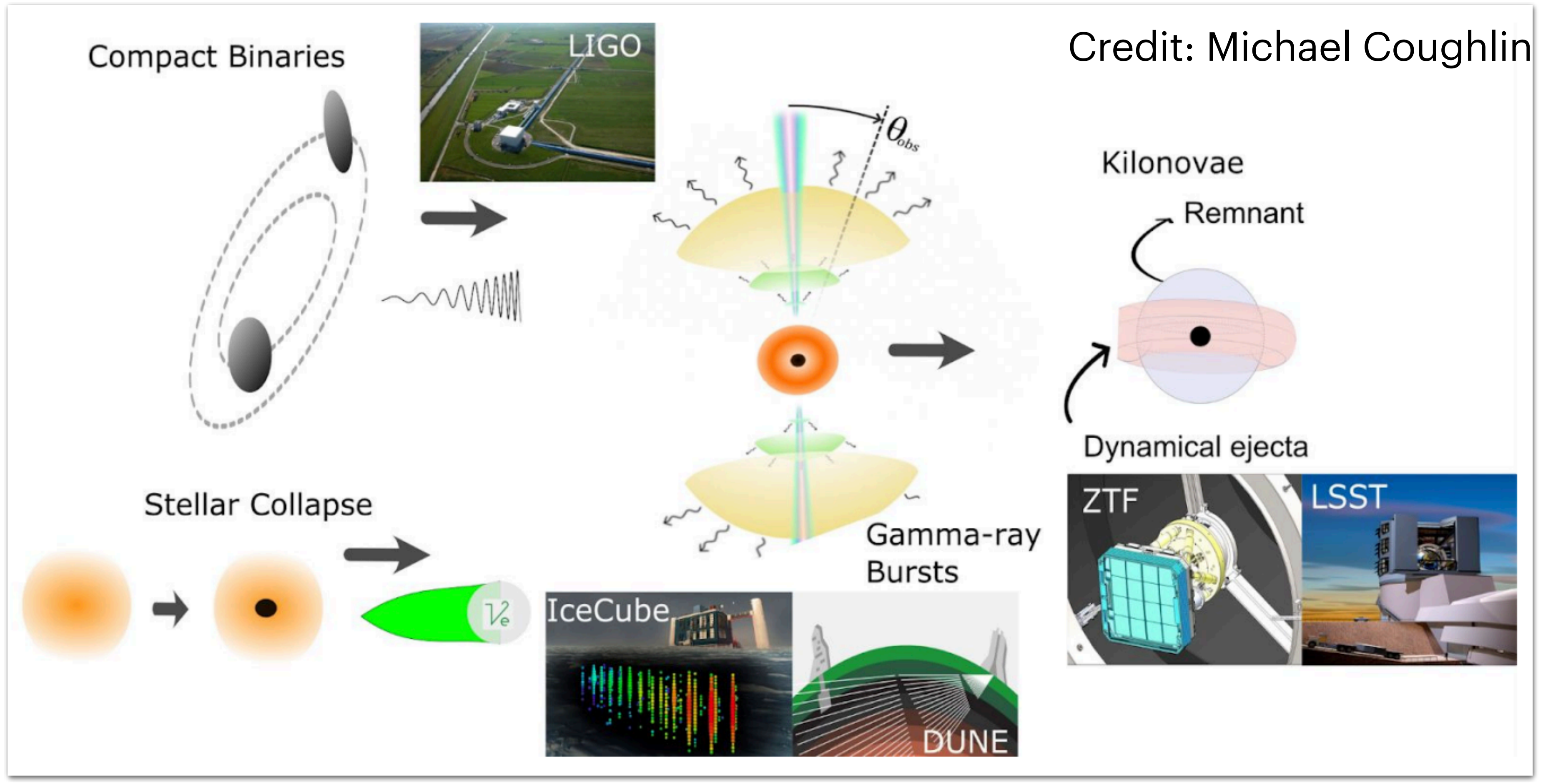
Dark sector signature
SpinQuest: muon final states
DarkQuest: e,γ,π,...

System upgrades
Existing EMCal from PHENIX
Tracking MWPC available
Tensor polarized deuteron target

<u>NSF A3D3 institute</u>: Domain Scientists, Computer Scientists and System Experts
Impact broader science domains <u>Fast ML for Science Workshop</u>

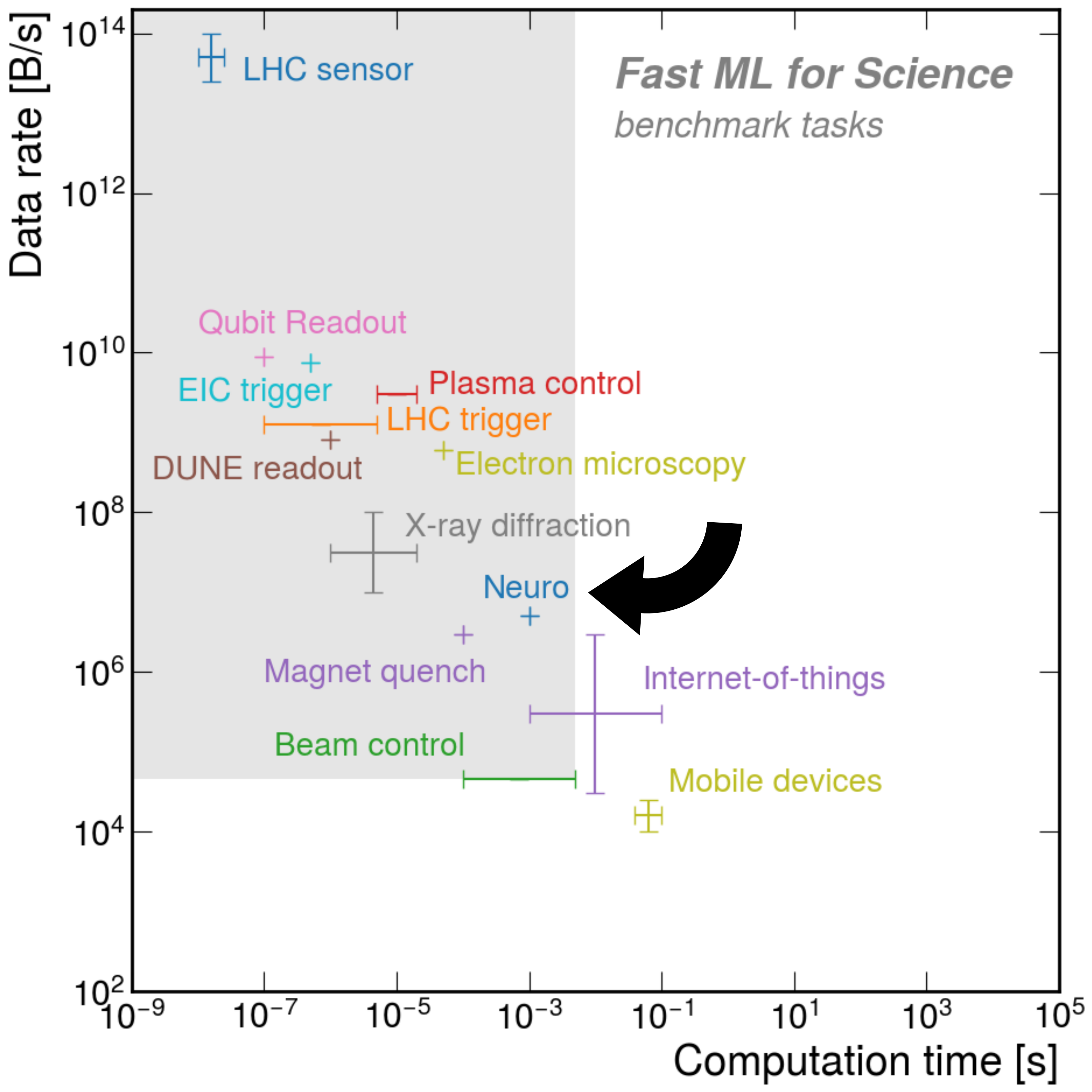
Fast ML for Science

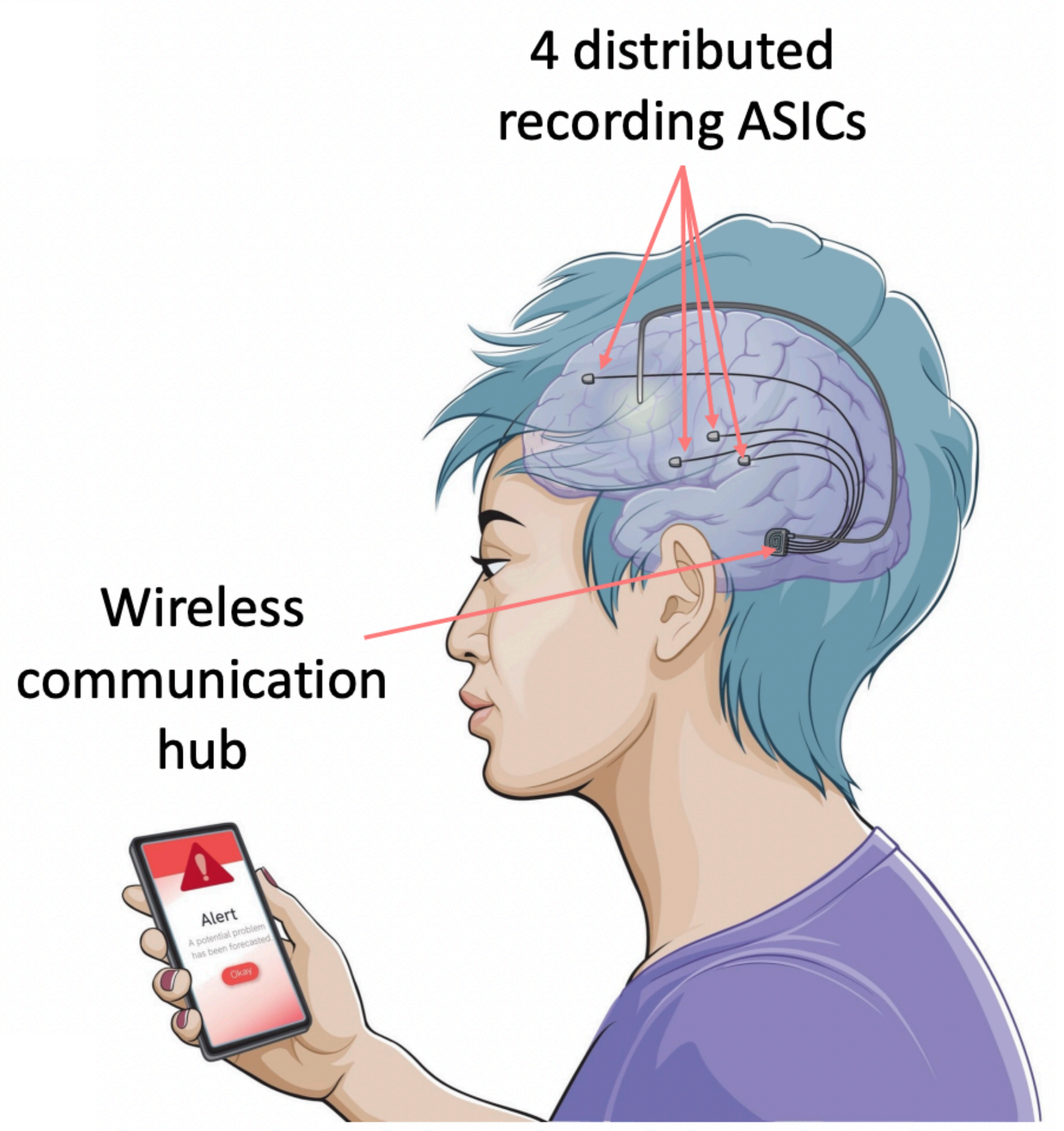




Supernova detection and multi-messenger astronomy

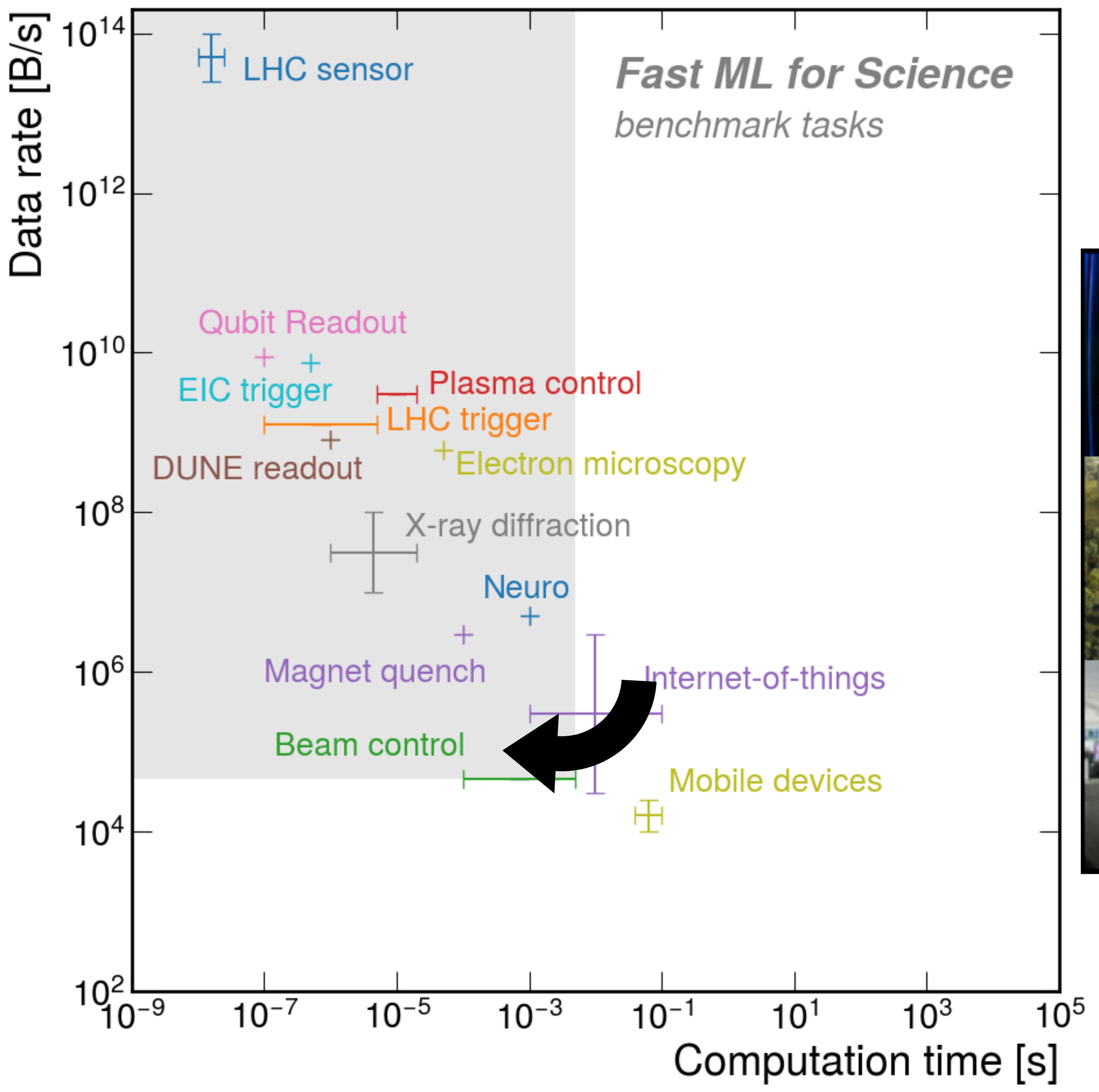
Fast ML for Science

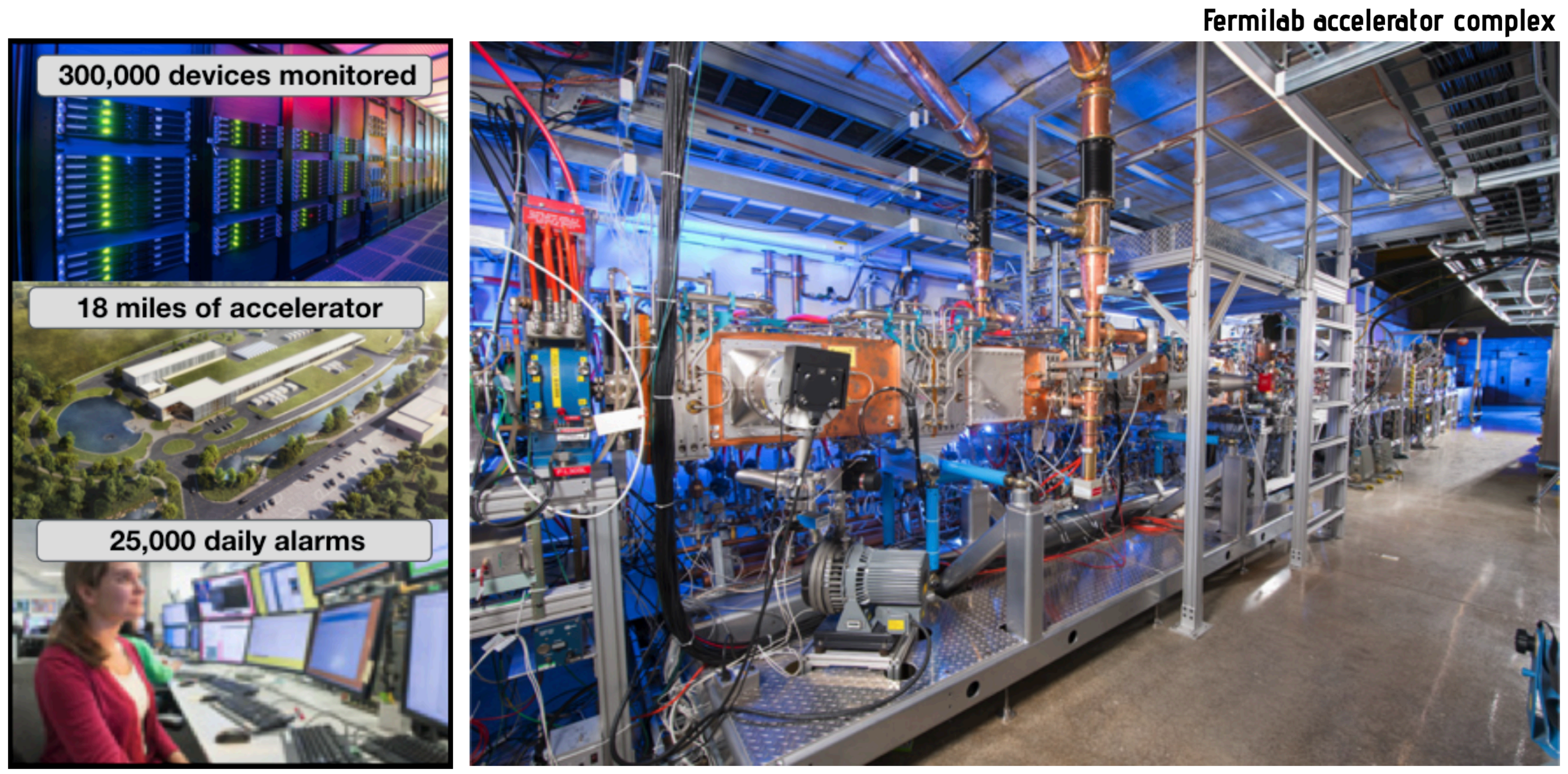




Real-time seizure detection

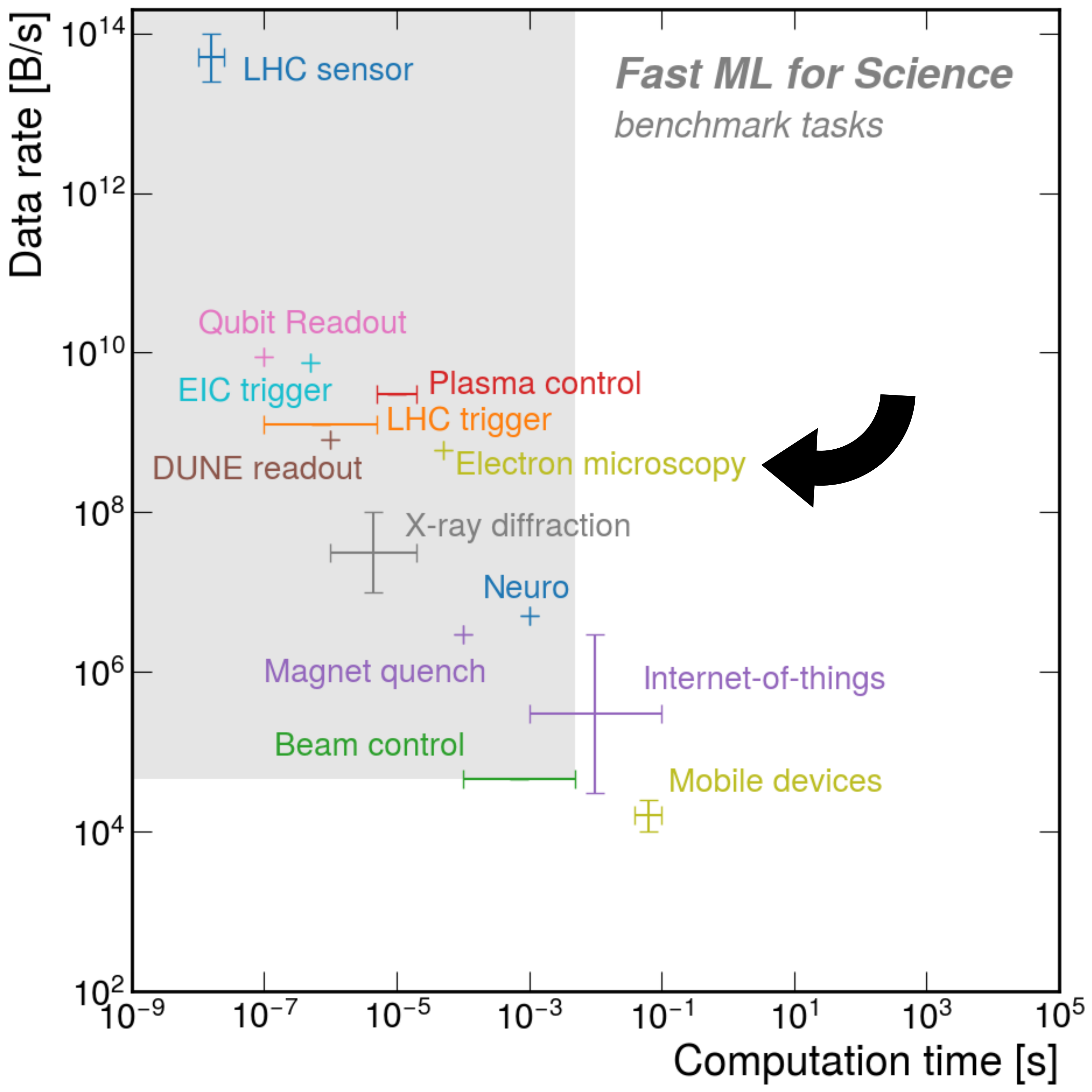
Fast ML for Science

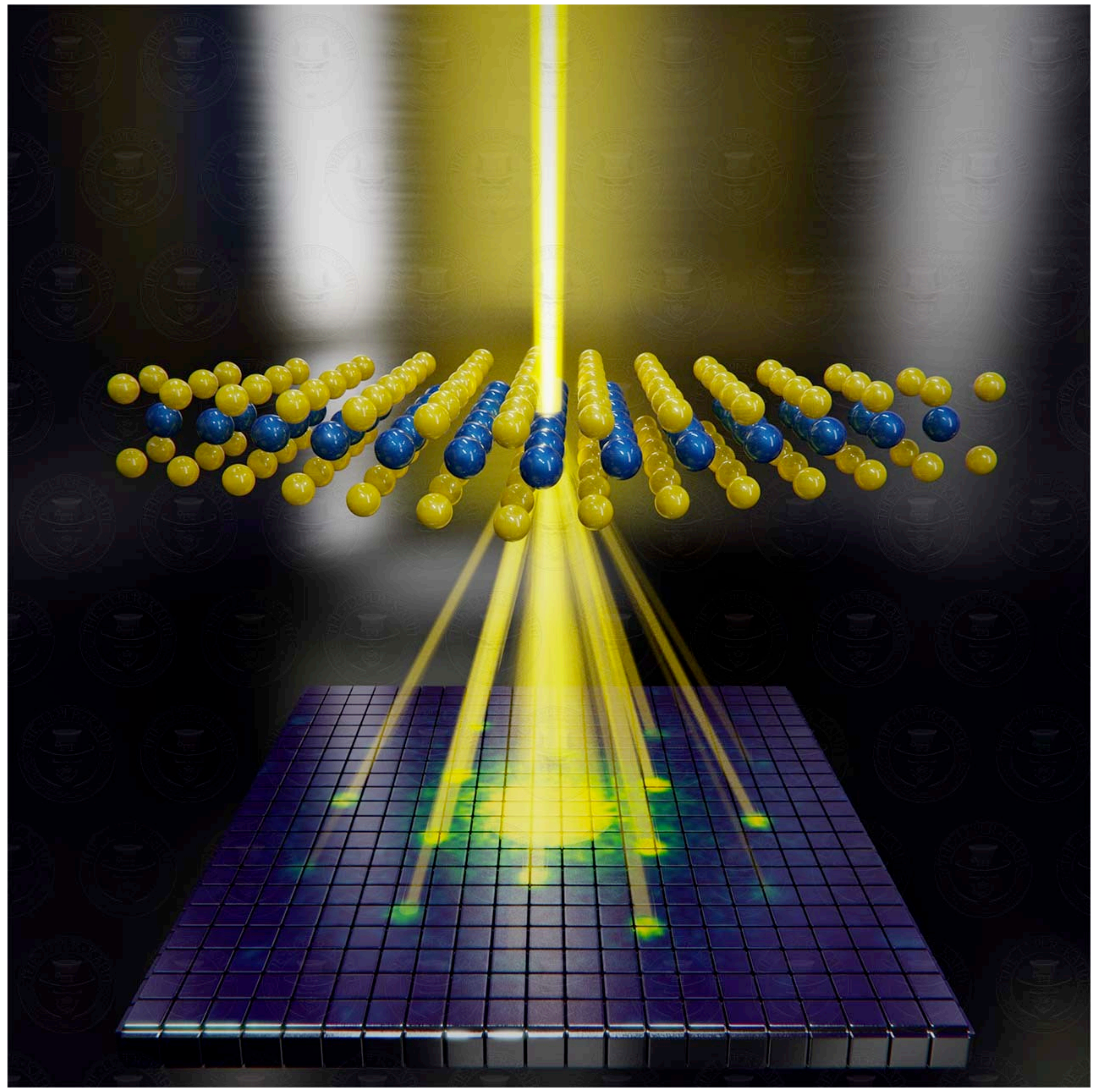




Particle accelerator controls

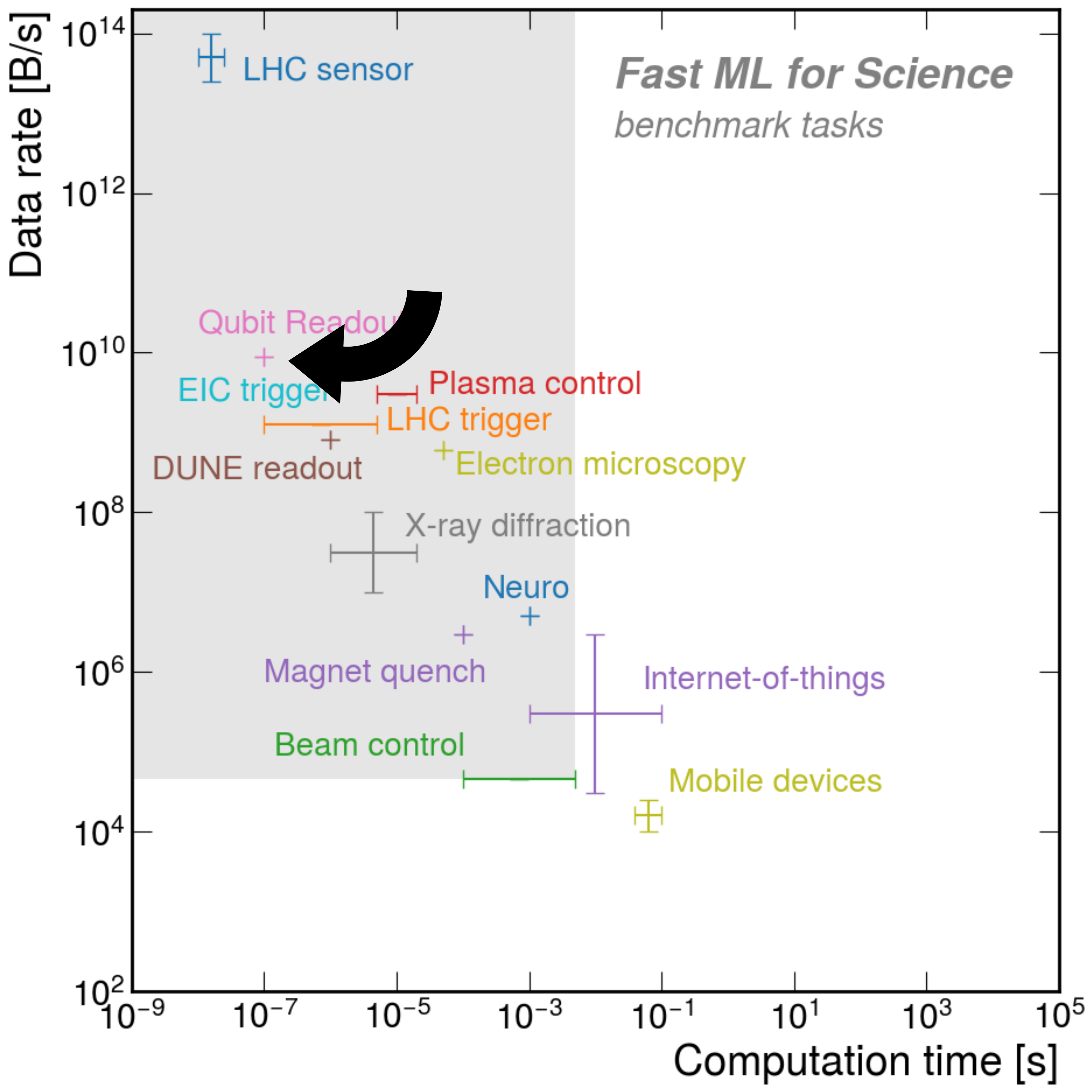
Fast ML for Science

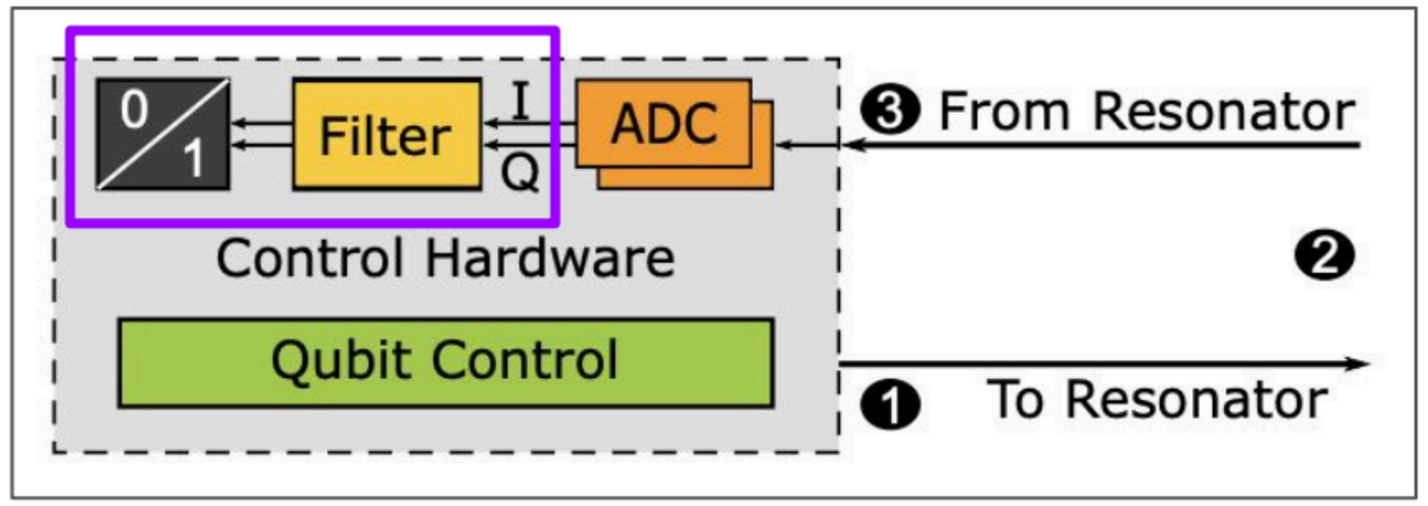


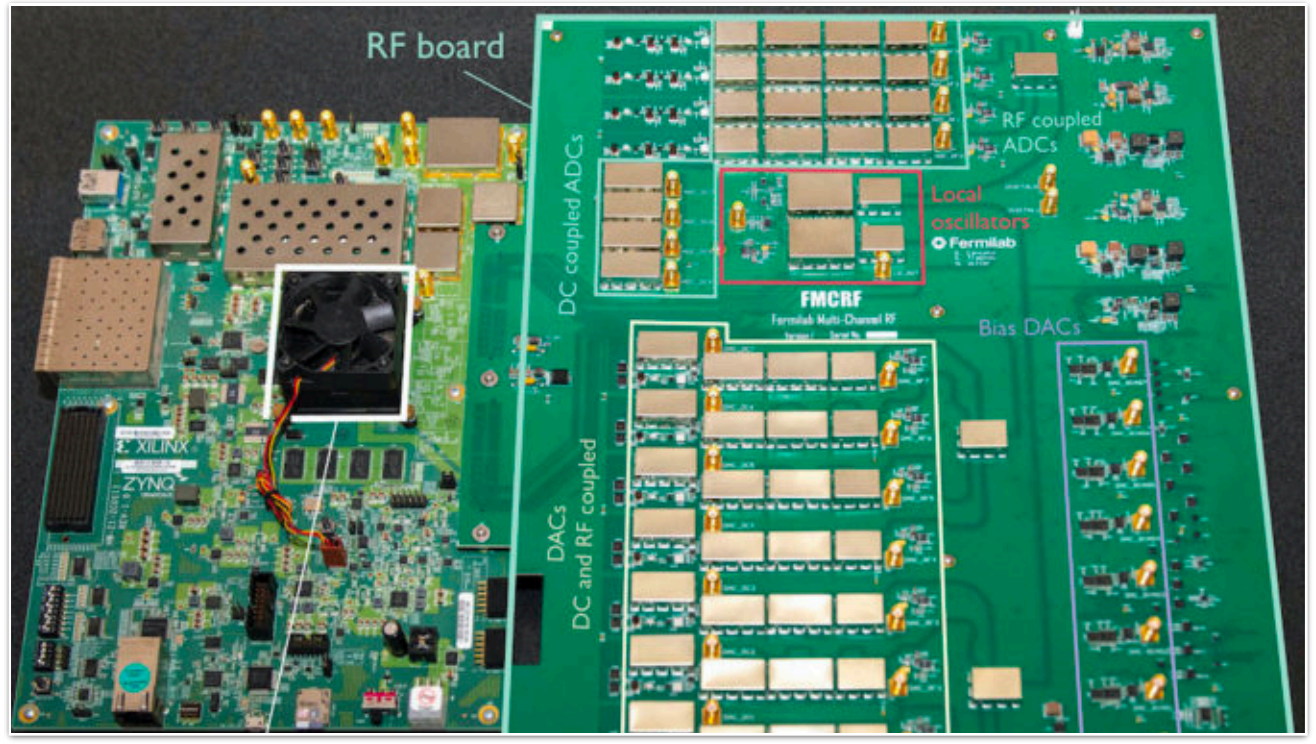


New materials for quantum and energy

Fast ML for Science



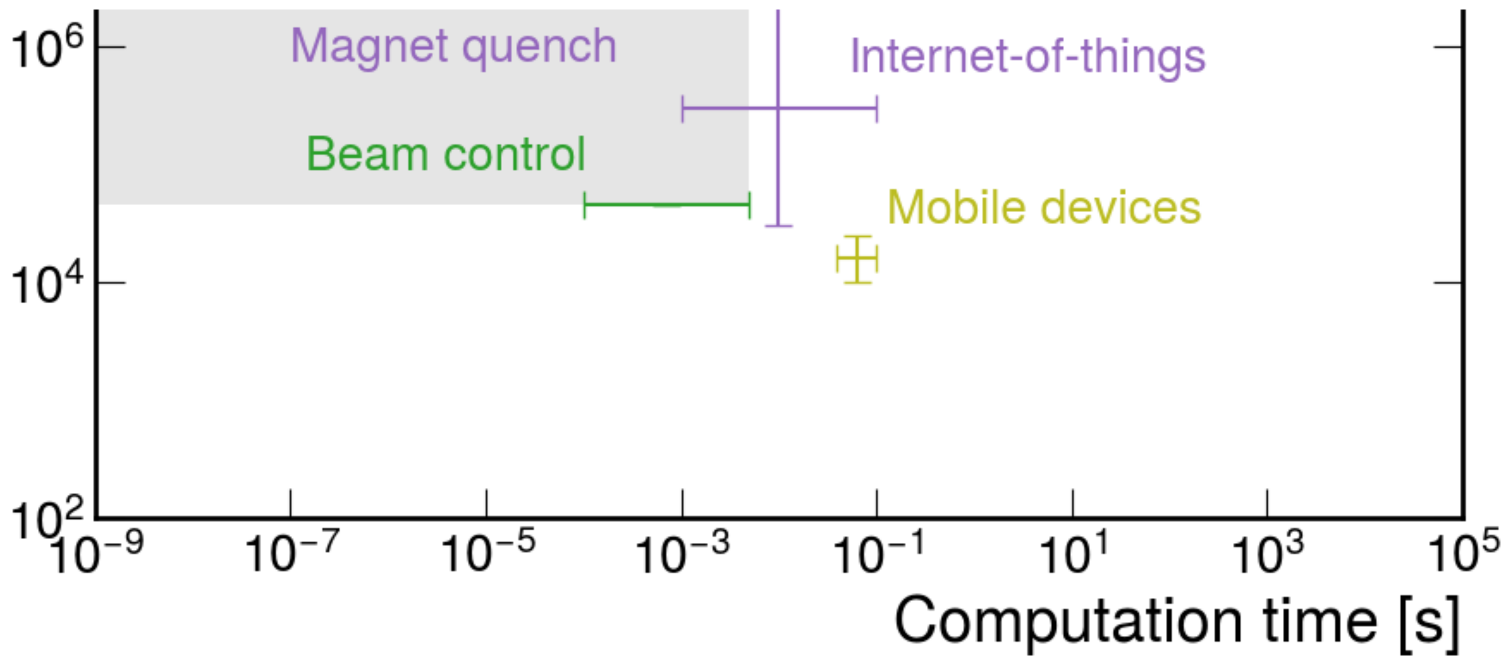


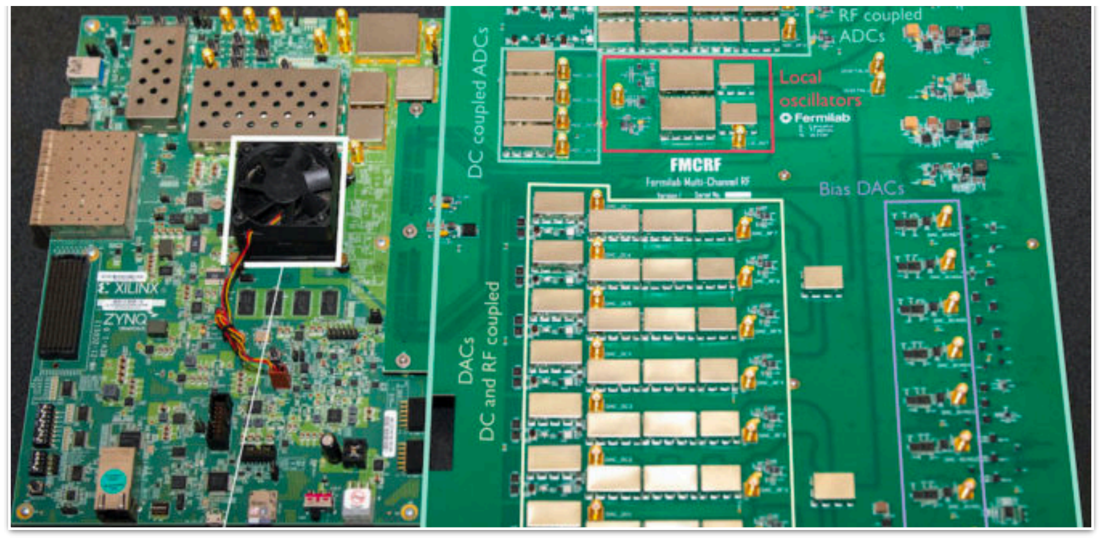


Qubit readout and control

Fast ML for Science

See many more at the fast machine learning for science workshop2023

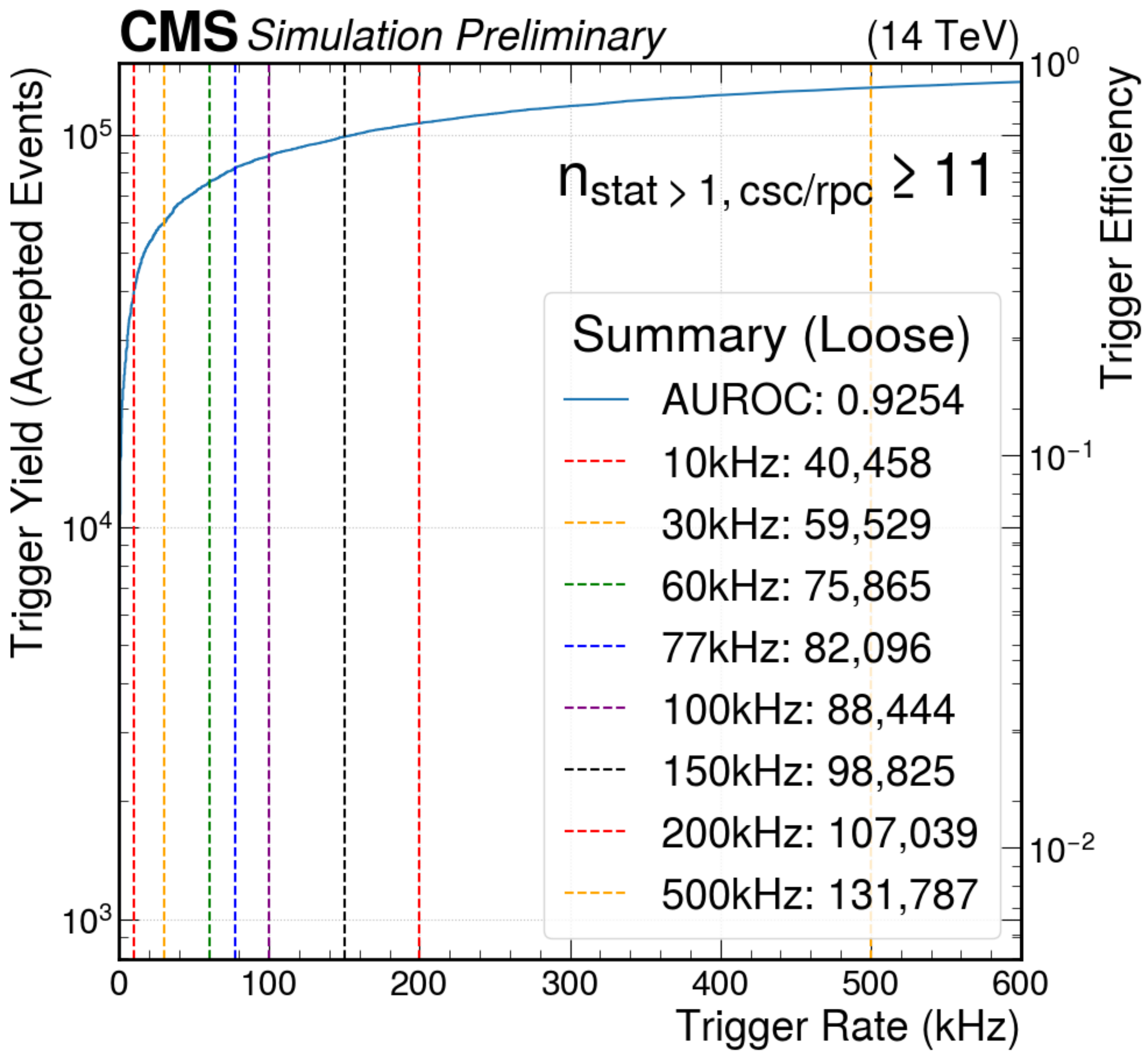




Qubit readout and control

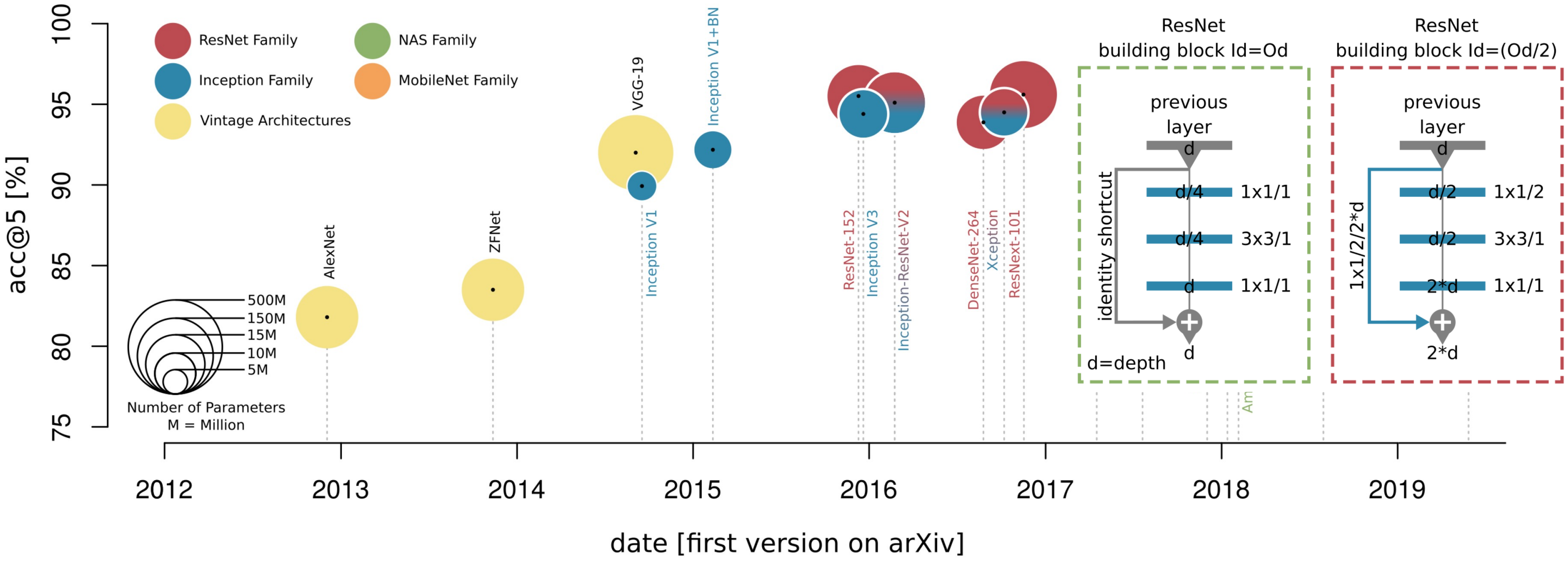
Final Remark

- Many existing and emerging opportunities in advancing our science results with Real-time ML/AI
  - New opportunities in searching for new physics
- Interesting research area touching overlaps of CS/AI, engineering and domain problems
- Lots of Fun



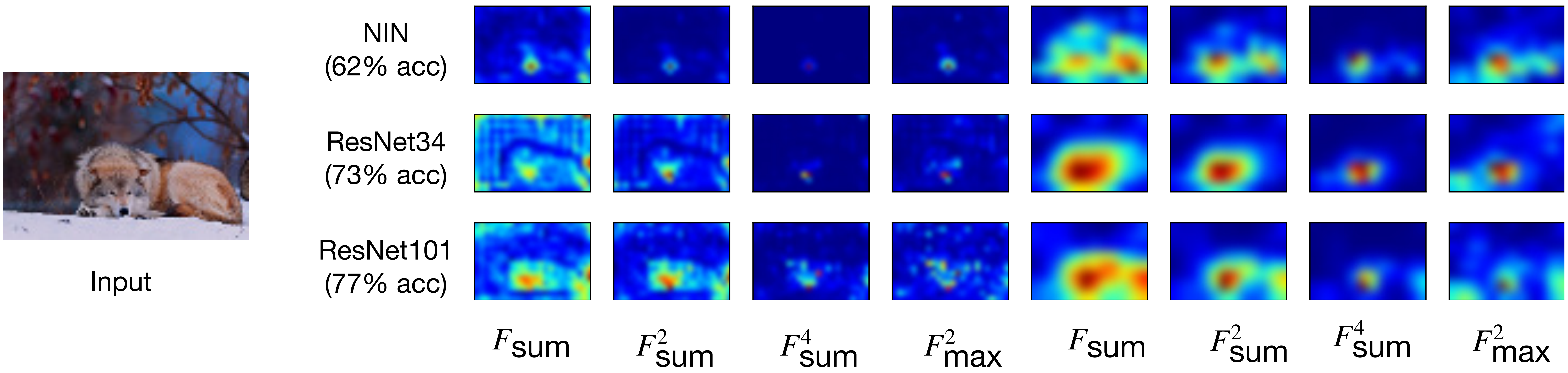
GNN based tagger can collect 5 times more new physics with exotic footprint in CMS detector

Image detection network evolution



Performant models have similar attention maps

Attention maps of performant ImageNet models (ResNets) are similar to each other, but the less performant model (NIN) has quite different attention maps.



Towards Scalable, Flexible, Adaptable GNN/ transformer with HLS4ML

- **hls4ml: great support for MLP and CNN Keras models.**
- Support of parsing PyTorch models: this has been improved!
- **Some (non-trivial) engineering work to support GNN/transformers:**
- Tau3mu Detection: MessagePassing layers, and meet 100 ns latency!
- **Long term: need to improve hls4ml code generation**
    - Current code generation in hls4ml is based on naive string generation - i.e., it becomes a mess very fast for anything complex.

sPHENIX tracking GNN hls4ml synthesis results

- Network inputs: nodes=80, edges=100
- Input network
    - Can be parallelized to be "nodes" times faster (i.e., 15ns)

Extremely preliminary - DO NOT TRUST NUMBERS

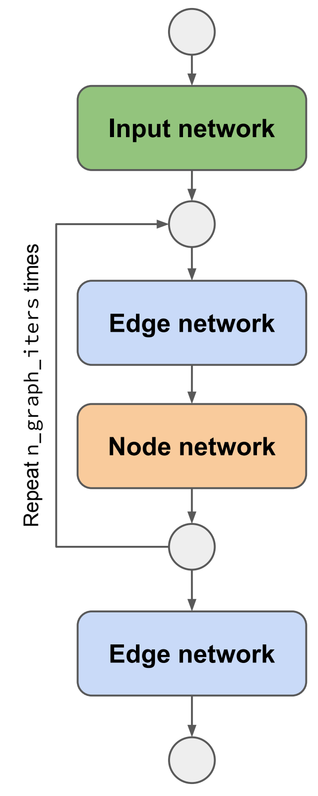
Latency	BRAMs	DSPs	FFs	LUTs
1.2 us	6.5%	0.3%	5%	7.5%

- Edge network

Latency	BRAMs	DSPs	FFs	LUTs
3 us	15%	2%	20%	65%

- Node network (results from HLS synthesis, vivado synthesis OOM'd)
    - Neet to optimize the scatter_add function (expecting ~2us for the net)

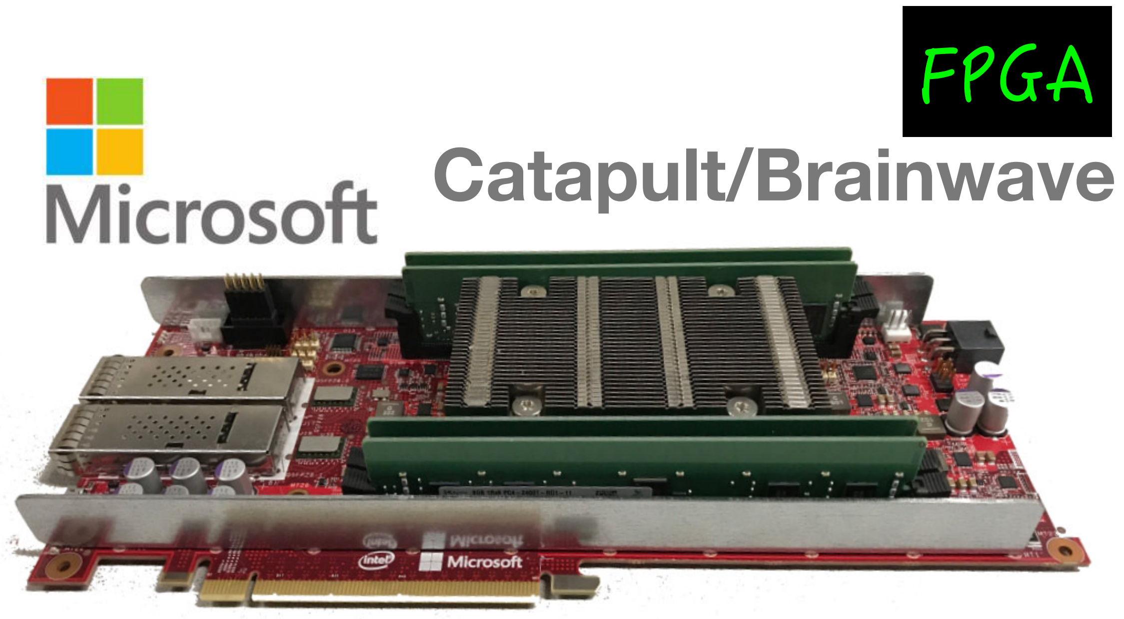
Latency	BRAMs	DSPs	FFs	LUTs
12 us	42%	7%	-	-

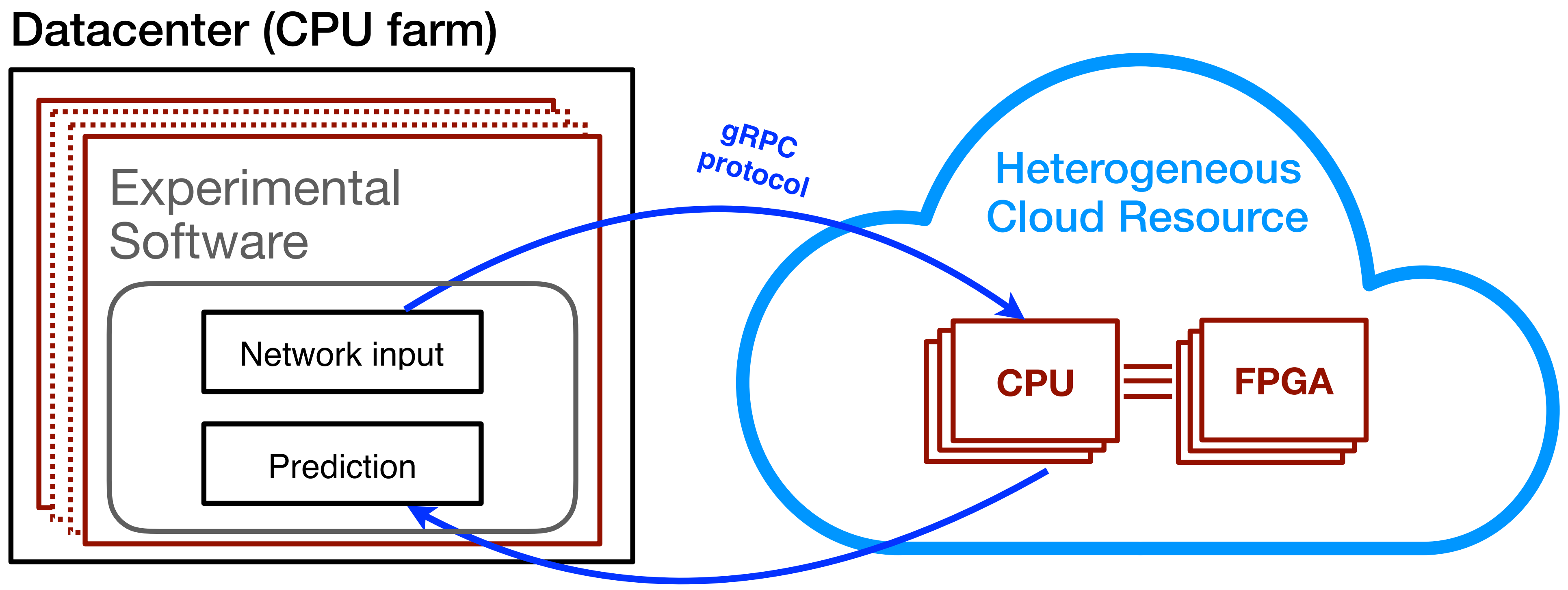


Example: Extended operations supported in hls4ml to implement a GNN developed for track reconstruction in the sPhenix trigger

- Added missing operations for GNN: Scatter_* "getitem", "gather", "ones()" and "zeros()" etc

A computing paradigm adaptive to changing hardware landscape





Services for **O**ptimized **N**etwork **I**nference on **C**o-processors.

https://arxiv.org/pdf/1904.08986.pdf

- Increasing demand for accelerating ML inference
- Demonstrated offloading ML inference to Microsoft Brainwave FPGA services
    - FPGA co-processor outperforms GPUs at batch-1 (relevant for streaming)
- CPU client software only handles preprocessing and I/O, not inference framework