Fermilab
Javier Duarte
Burt Holzman
Sergo Jindariani
Benjamin Kreis
Mia Liu
Kevin Pedro
Nhan Tran
Aristeidis Tsaris

MIT
Philip Harris
Dylan Rankin

University of Illinois at Chicago
Zhenbin Wu

University of Illinois at Urbana-Champaign
Mark Neubauer

Microsoft
Suffian Khan
Brandon Perez
Colin Versteeg
Ted W. Way

University of Washington
Scott Hauck
Shih-Chieh Hsu
Matthew Trahms
Dustin Werran

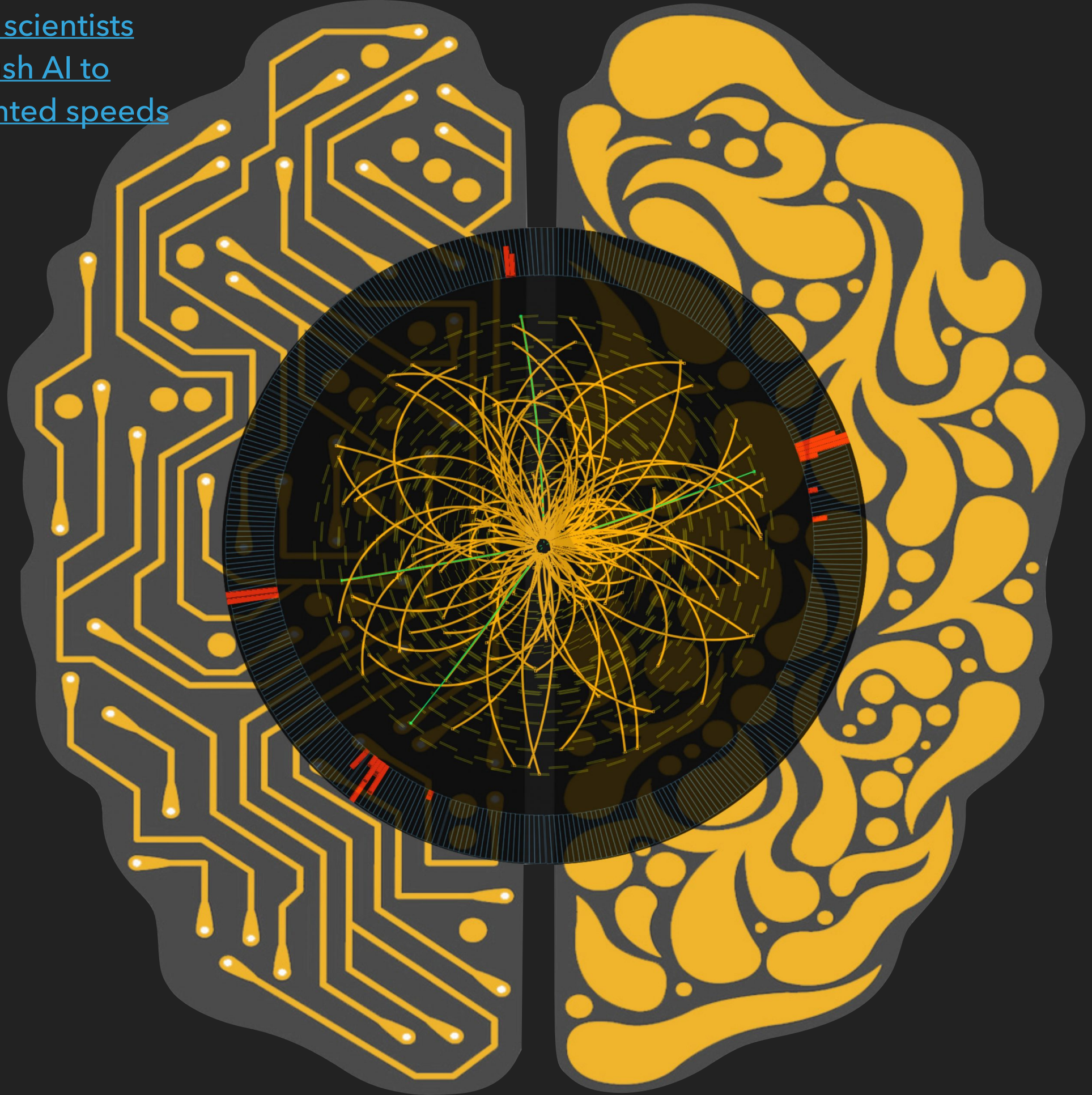
CERN
Vladimir Loncar
Jennifer Ngadiuba
Maurizio Pierini



# FPGA-ACCELERATED MACHINE LEARNING INFERENCE FOR PARTICLE PHYSICS

APRIL 2, 2019
CONNECTING THE DOTS
VALENCIA, SPAIN

# Fermilab scientists help push AI to unprecedented speeds



# NEXT-GEN BIG DATA

LHC – 2016
50 PB raw data

LHC Science data ~200 PB

Facebook uploads 180 PB

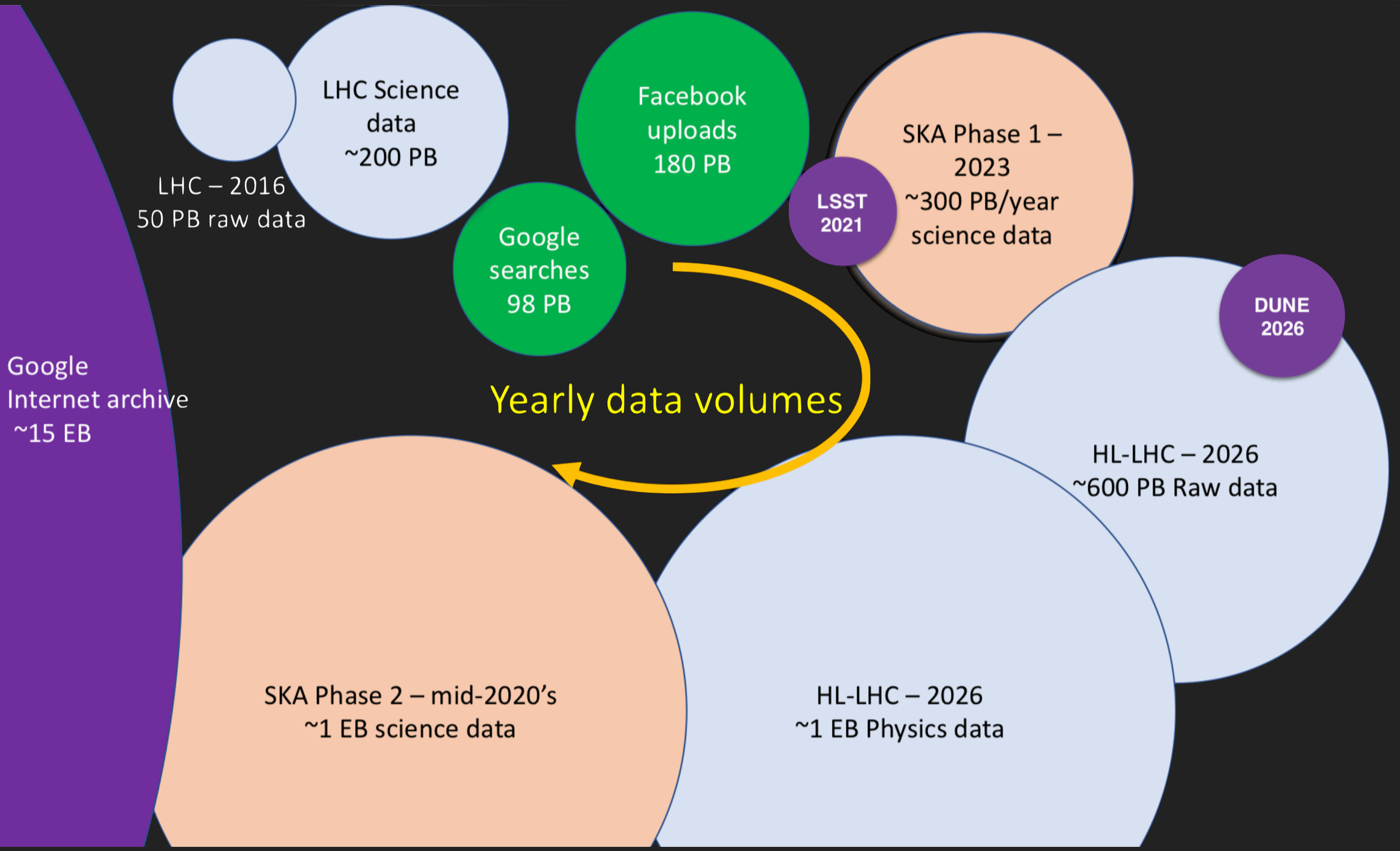
Google searches 98 PB

chive

Yearly data volume

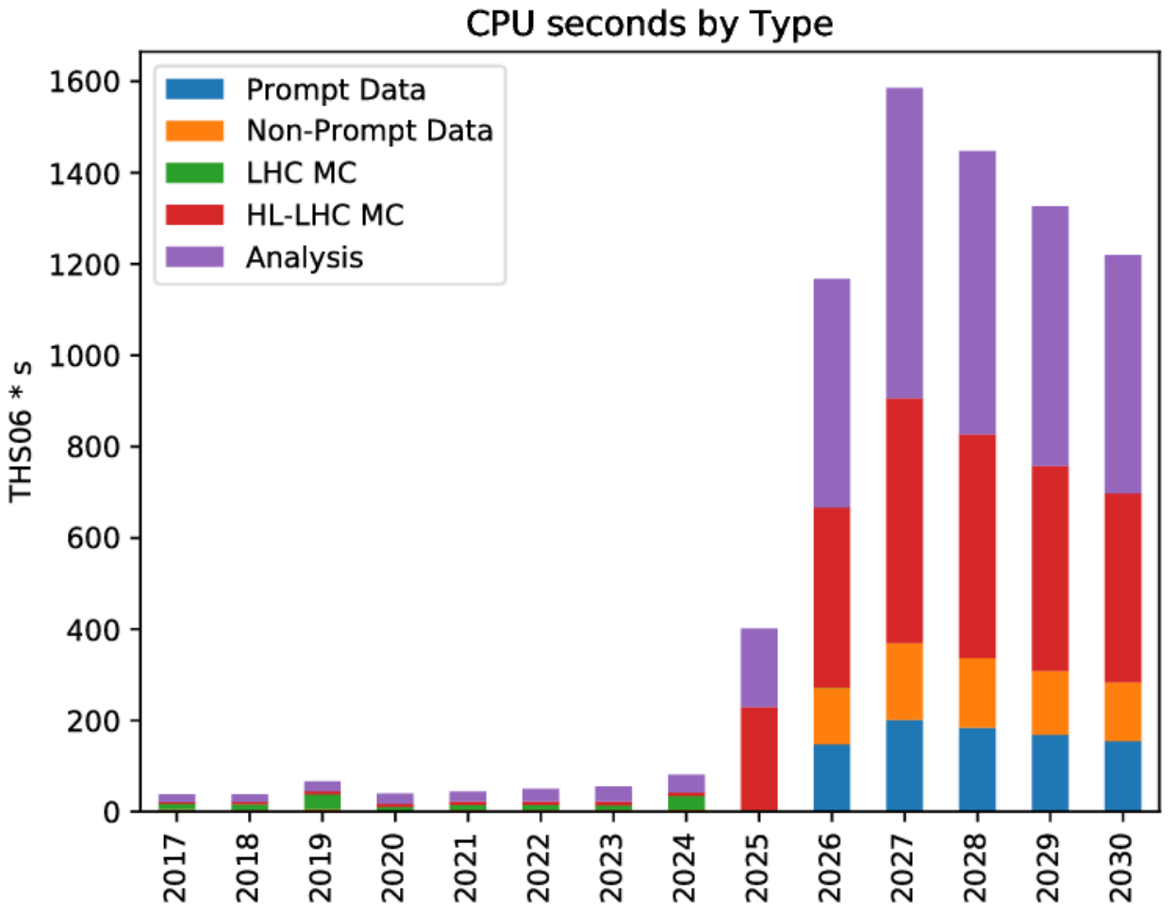
# NEXT-GEN BIG DATA

- HL-LHC will reach 1 exabyte of data per year



# NEXT-GEN COMPUTING CHALLENGES



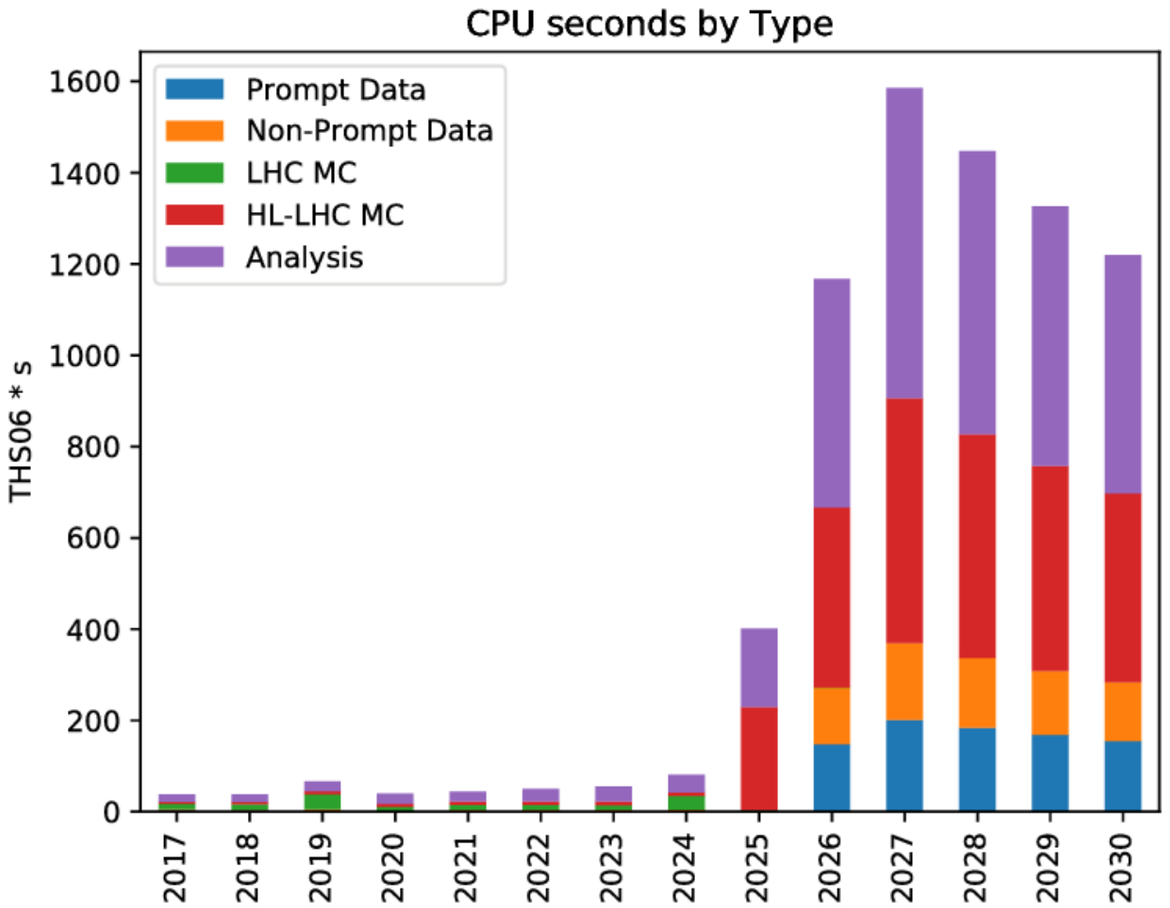



- Energy frontier: **HL-LHC**
  - 10× data vs. Run 2+3
  - 200 PU (vs. ~40 PU in Run 2)
  - CMS: 15× increase in pixel channels, 65× increase in calorimeter channels (similar for ATLAS)

[HSF Community White Paper]

# NEXT-GEN COMPUTING CHALLENGES



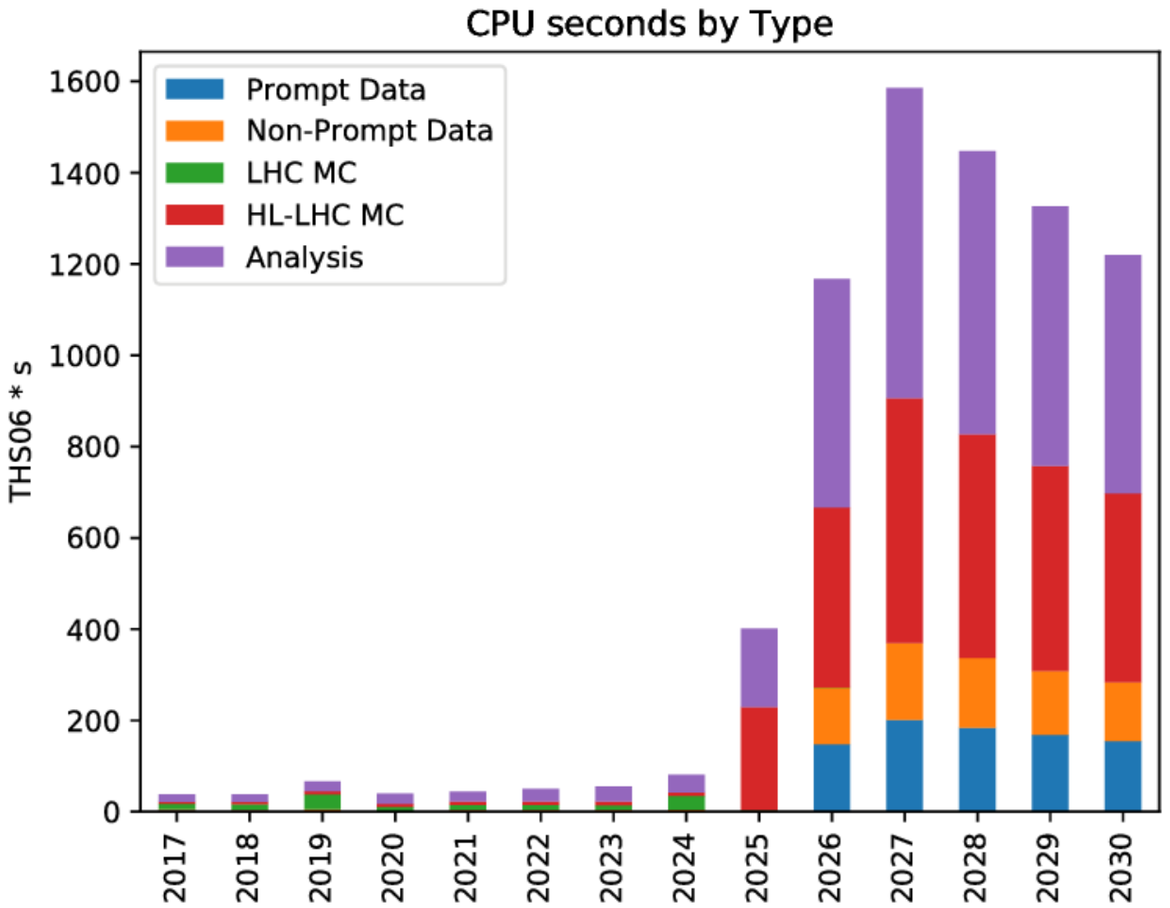



[HSF Community White Paper]

- Energy frontier: **HL-LHC**
  - 10× data vs. Run 2+3
  - 200 PU (vs. ~40 PU in Run 2)
  - CMS: 15× increase in pixel channels, 65× increase in calorimeter channels (similar for ATLAS)
- Intensity frontier: **DUNE**
  - Largest liquid argon detector ever designed
  - ~1M channels, 1 ms integration time with MHz sampling
    → 30+ petabytes/year

# NEXT-GEN COMPUTING CHALLENGES






[HSF Community White Paper]

- Energy frontier: **HL-LHC**
  - 10× data vs. Run 2+3
  - 200 PU (vs. ~40 PU in Run 2)
  - CMS: 15× increase in pixel channels, 65× increase in calorimeter channels (similar for ATLAS)
- Intensity frontier: **DUNE**
  - Largest liquid argon detector ever designed
  - ~1M channels, 1 ms integration time with MHz sampling
    → 30+ petabytes/year

- **CPU needs** for particle physics **will increase** by more than **an order of magnitude** in the next decade

# NEXT-GEN COMPUTING OPTIONS



- Options to tackle the computing issues

# NEXT-GEN COMPUTING OPTIONS



- Options to tackle the computing issues


1. Rewrite our algorithms from the ground-up to take advantage of HPCs, multicore CPUs, GPUs, FPGAs, or ASICs

- example: `mkFit` parallelized/vectorized Kalman filter

# NEXT-GEN COMPUTING OPTIONS



- Options to tackle the computing issues


1. Rewrite our algorithms from the ground-up to take advantage of HPCs, multicore CPUs, GPUs, FPGAs, or ASICs
   - example: `mkFit` parallelized/vectorized Kalman filter
2. Recast our physics problem as a machine learning problem
   - example: HEP.TrkX work with graph neural networks

# NEXT-GEN COMPUTING OPTIONS



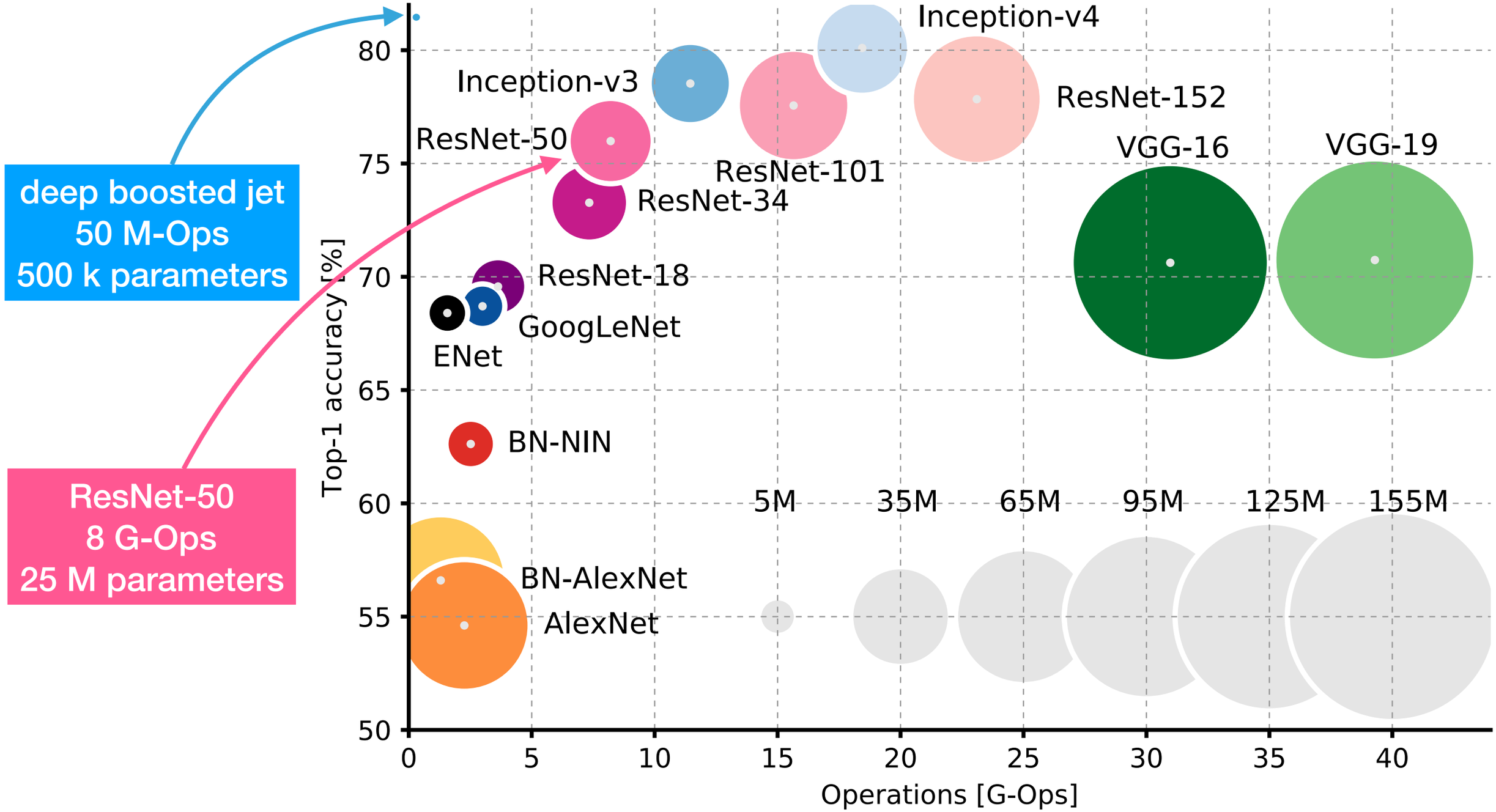
- Options to tackle the computing issues


1. Rewrite our algorithms from the ground-up to take advantage of HPCs, multicore CPUs, GPUs, FPGAs, or ASICs
   - example: `mkFit` parallelized/vectorized Kalman filter
2. Recast our physics problem as a machine learning problem
   - example: HEP.TrkX work with graph neural networks
   - pros: industry does the heavy lifting for us

# BIG NETWORKS COMPARED TO HIGH ENERGY PHYSICS



- ResNet-50 representative of the **size of network** we will need to aid reconstruction at HL-LHC or DUNE

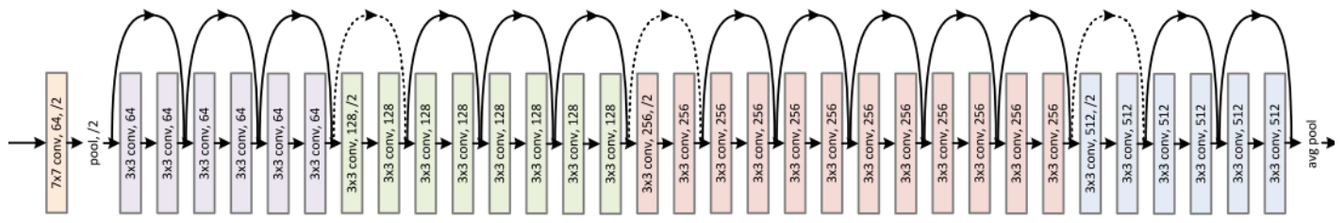


# EXAMPLE: RESNET-50



- Deep convolutional neural network with 50 hidden layers

ResNet-50



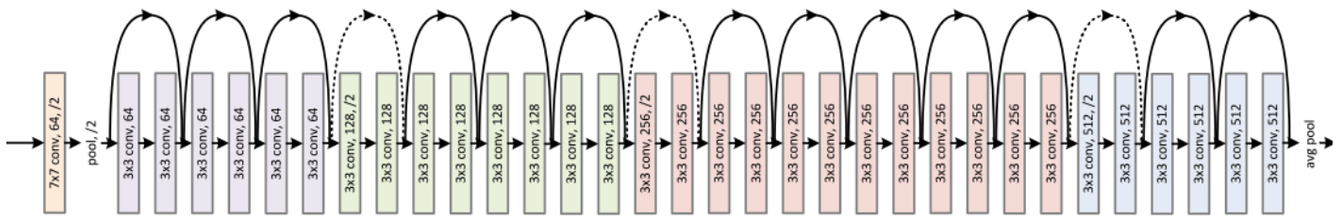
# EXAMPLE: RESNET-50



- Deep convolutional neural network with 50 hidden layers
- Trained to identify 1000 categories of natural objects

input

ResNet-50



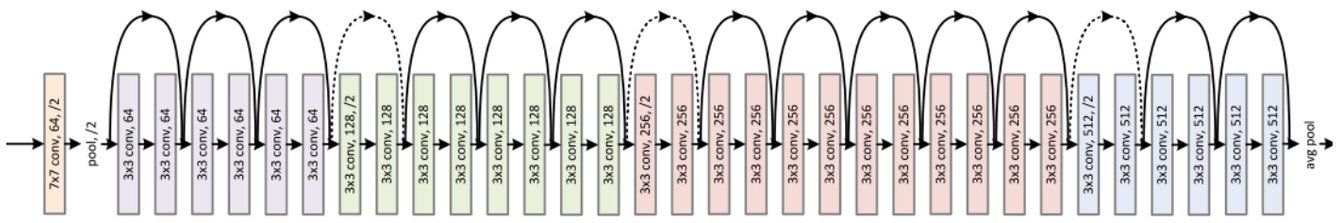
# EXAMPLE: RESNET-50



- Deep convolutional neural network with 50 hidden layers
- Trained to identify 1000 categories of natural objects

input

ResNet-50




output

| | |
|---|---|
| whippet | 0.4705 |
| Italian greyhound | 0.1953 |
| Rhodesian ridgeback | 0.1801 |
| Staffordshire bullterrier | 0.0377 |
| basenji | 0.0318 |
| ⋮ | ⋮ |

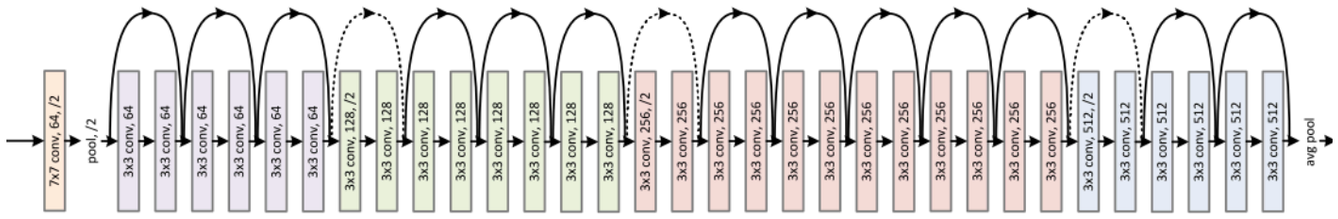
# EXAMPLE: RESNET-50



- Deep convolutional neural network with 50 hidden layers
- Trained to identify 1000 categories of natural objects

input

ResNet-50




output

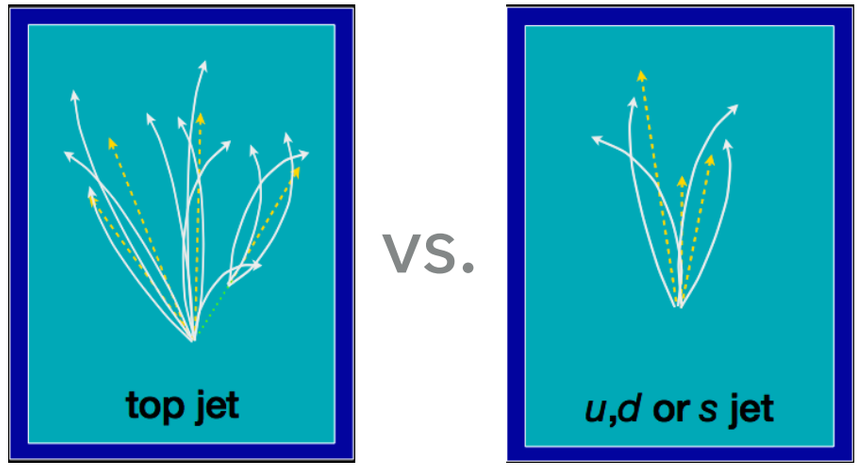
| | |
|---|---|
| whippet | 0.4705 |
| Italian greyhound | 0.1953 |
| Rhodesian ridgeback | 0.1801 |
| Staffordshire bullterrier | 0.0377 |
| basenji | 0.0318 |
| ⋮ | ⋮ |

Typical runtime on laptop
(or CMS CPU): **~2 s**

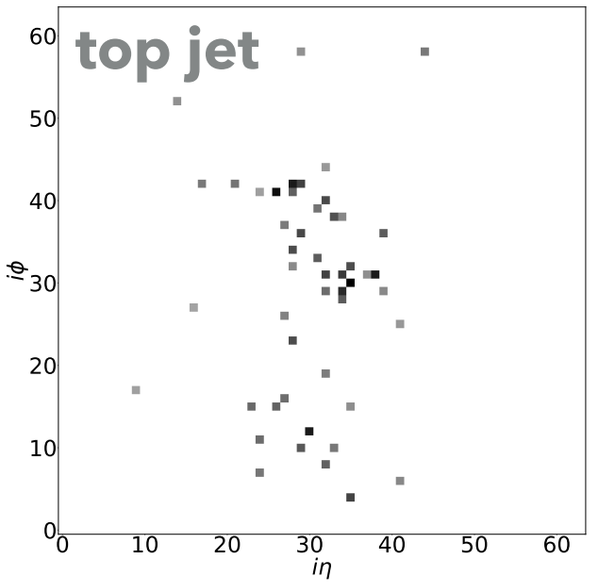
# JET VISION WITH RESNET-50



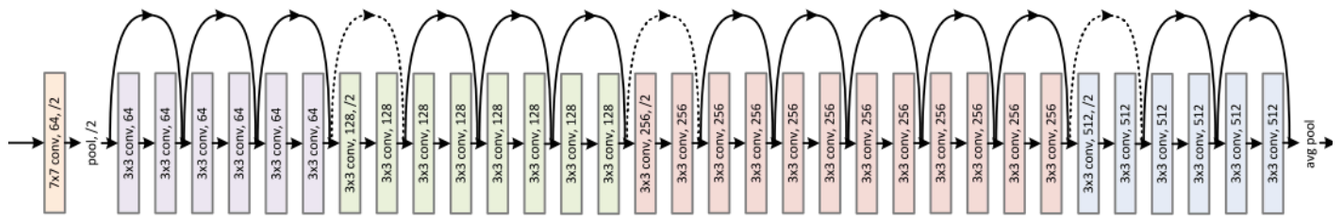
- Re-train ResNet-50 to identify the origin of jets
- Inputs are *jet images* = pixelated versions of calorimeter hits in 2D (η, Φ)




input




ResNet-50

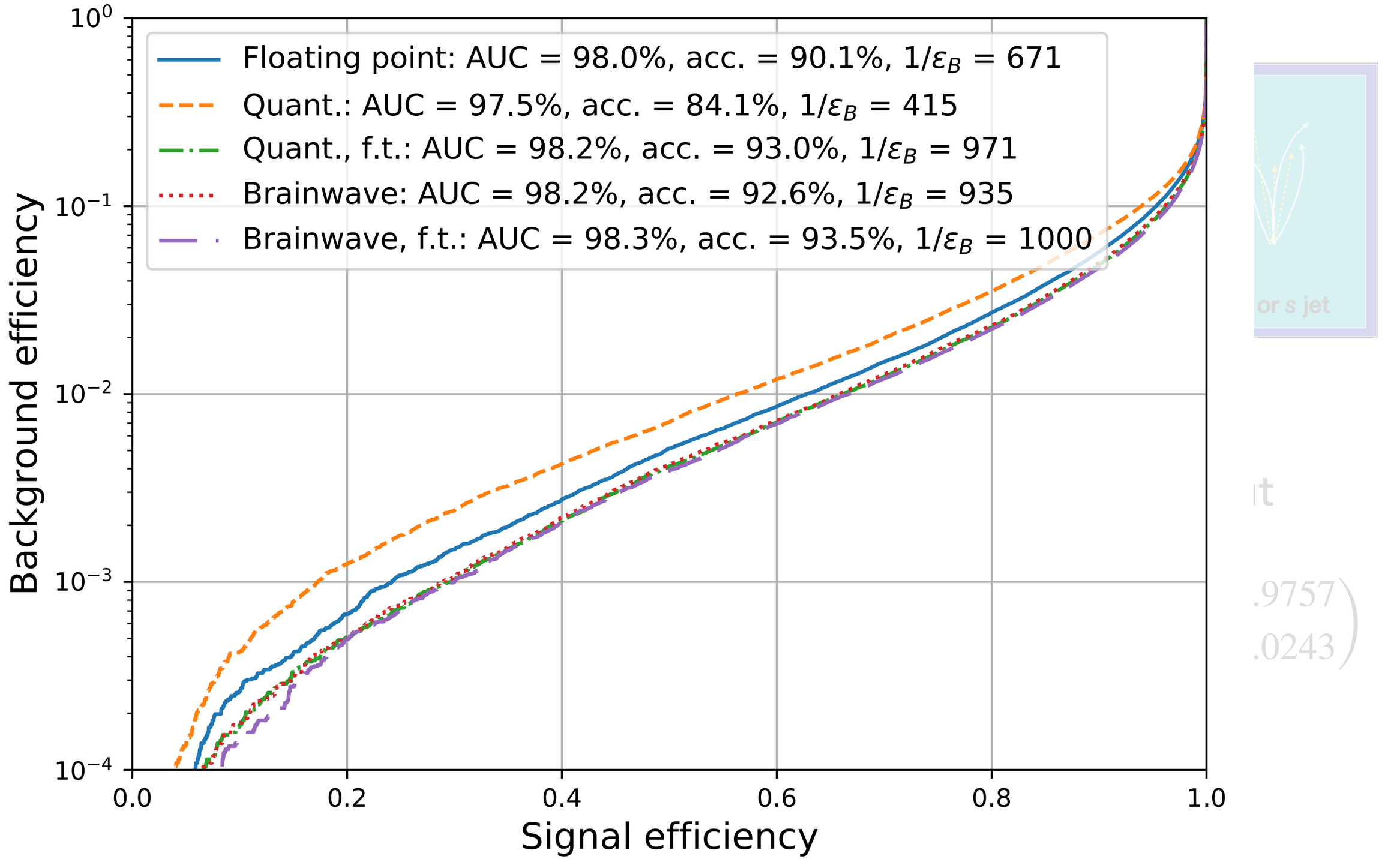



output

$$\begin{pmatrix} \text{top jet} & 0.9757 \\ \text{QCD jet} & 0.0243 \end{pmatrix}$$

# JET VISION WITH RESNET-50

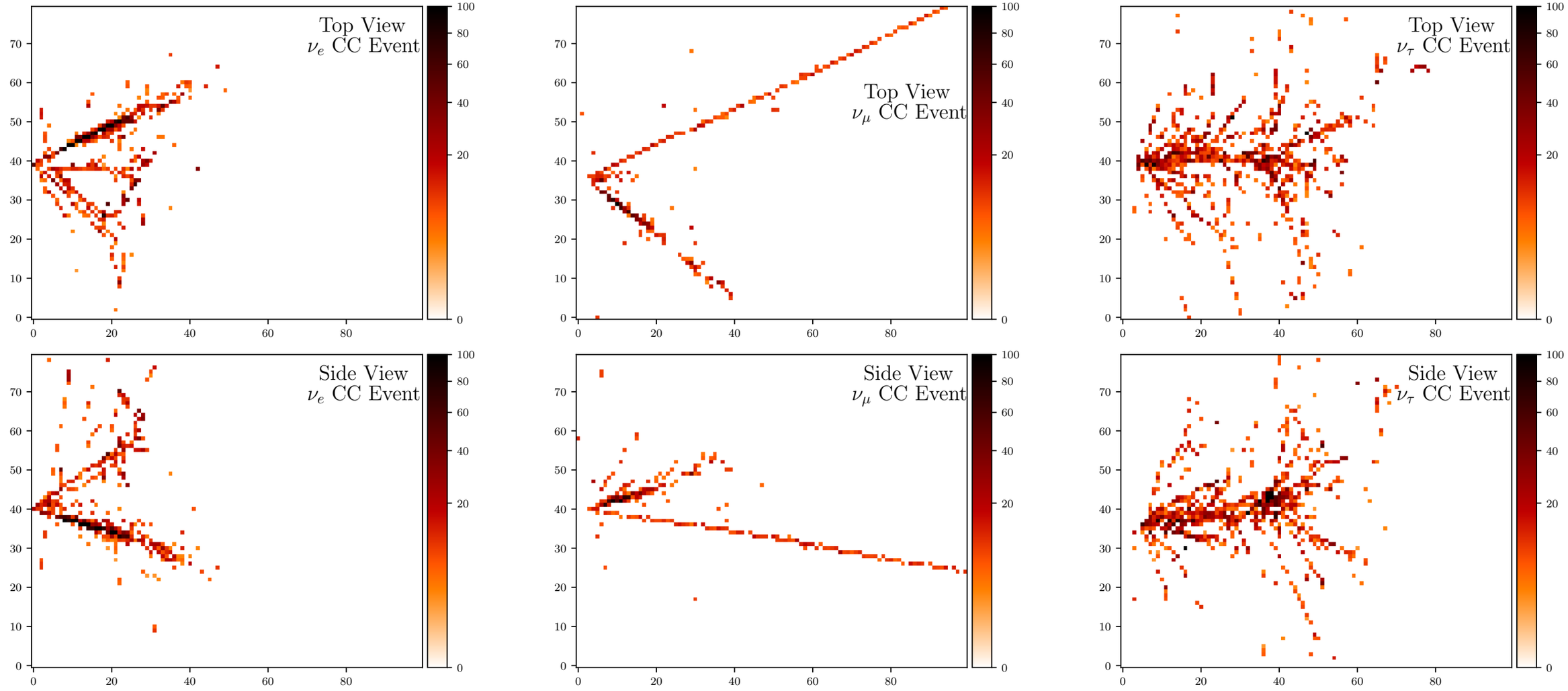





**State-of-the-art performance (even with “quantized” model)!**

# NEUTRINO VISION WITH RESNET-50



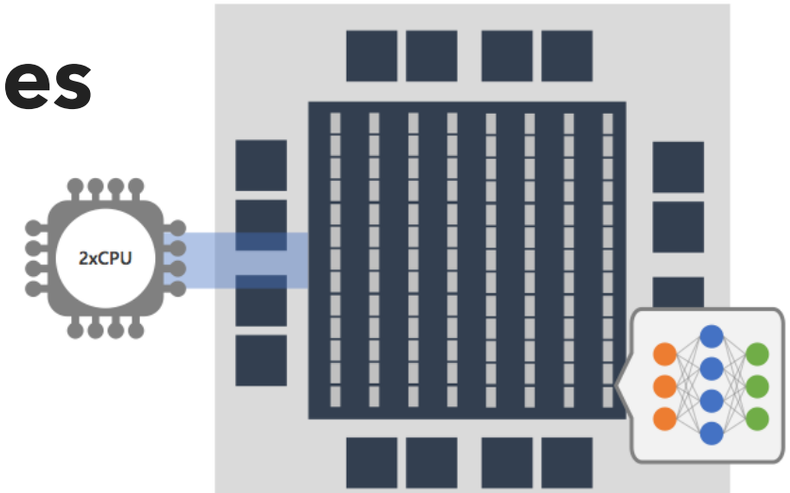



- ResNet-50 can also classify neutrino events to reject cosmic ray backgrounds (events above selected with prob. > 0.9)
- NOvA: first particle physics experiment to publish a result using a CNN (arXiv:1604.01444, arXiv:1703.03328)
- CNN inference already dominates neutrino reconstruction time!

# WHY ACCELERATE INFERENCE?



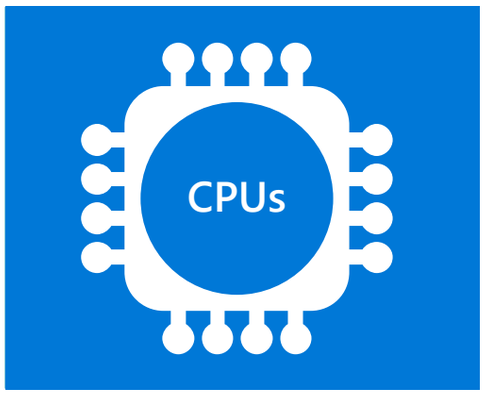
- DNNs are trained once/year/algorithm
  - Cloud GPUs or HPCs are good options
- Once trained, inference run **billions of times** in simulation, reconstruction, trigger, analysis, …




- Inference **as a service**
  - Seamless integration into existing computing model
  - Minimize dependence on specific hardware

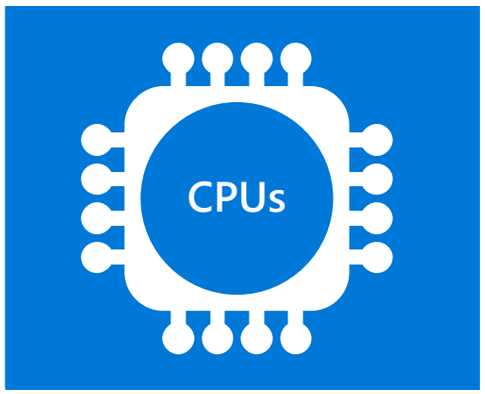
# HARDWARE ALTERNATIVES

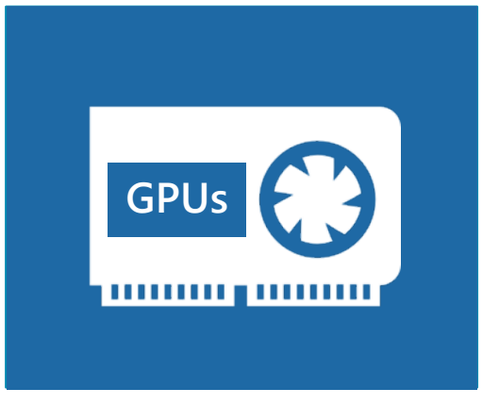





# HARDWARE ALTERNATIVES





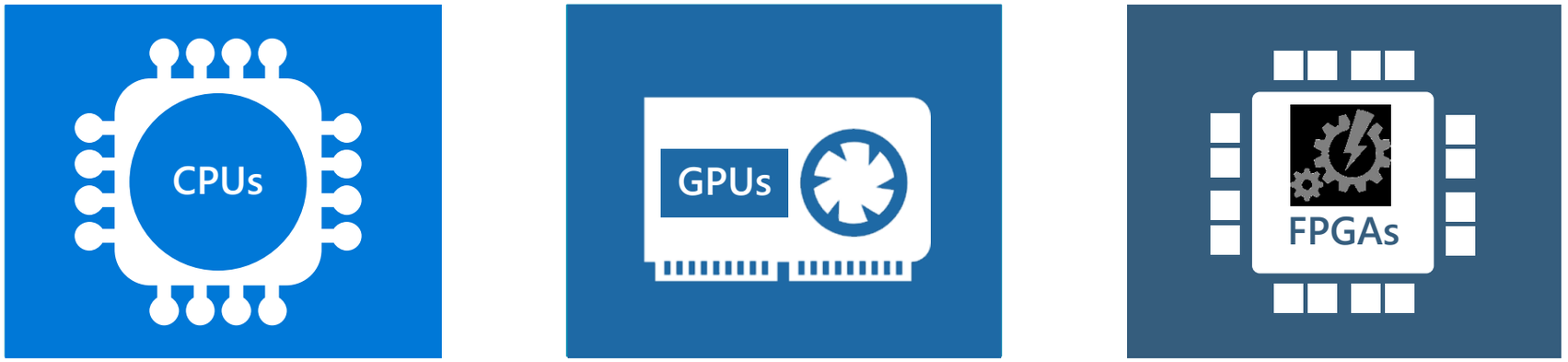




FLEXIBILITY

EFFICIENCY

# HARDWARE ALTERNATIVES



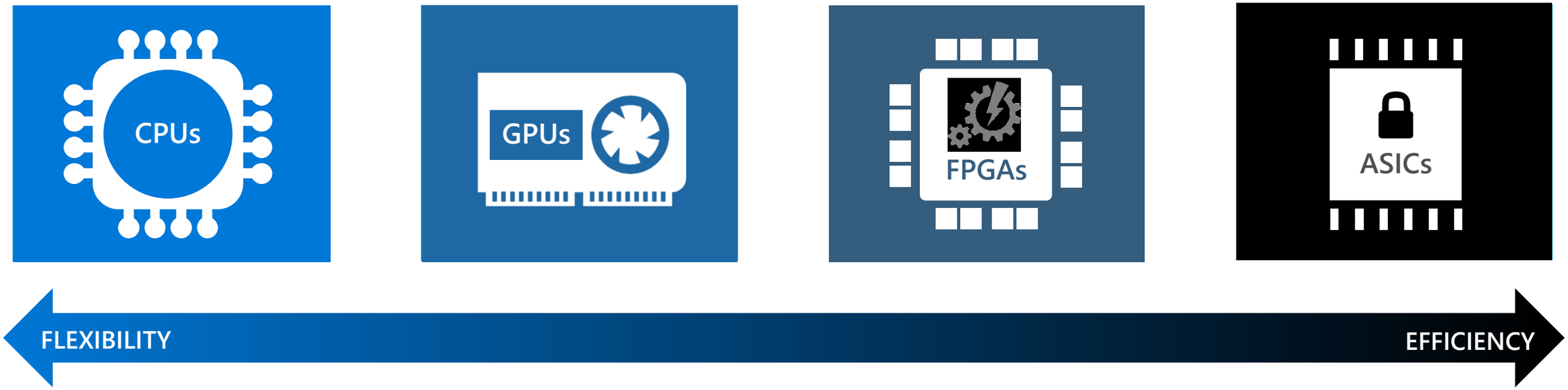



FLEXIBILITY

EFFICIENCY

# HARDWARE ALTERNATIVES

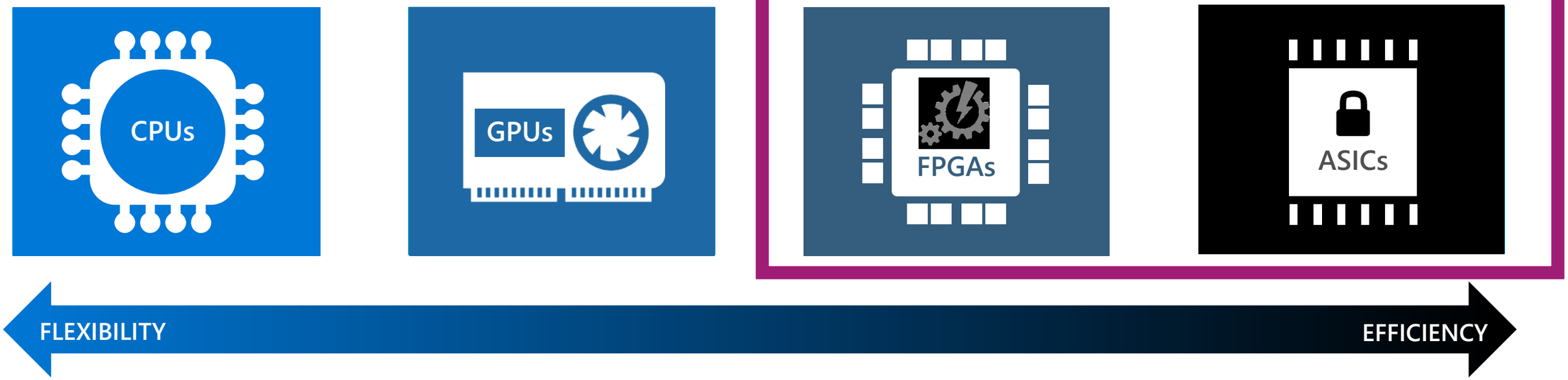




# HARDWARE ALTERNATIVES



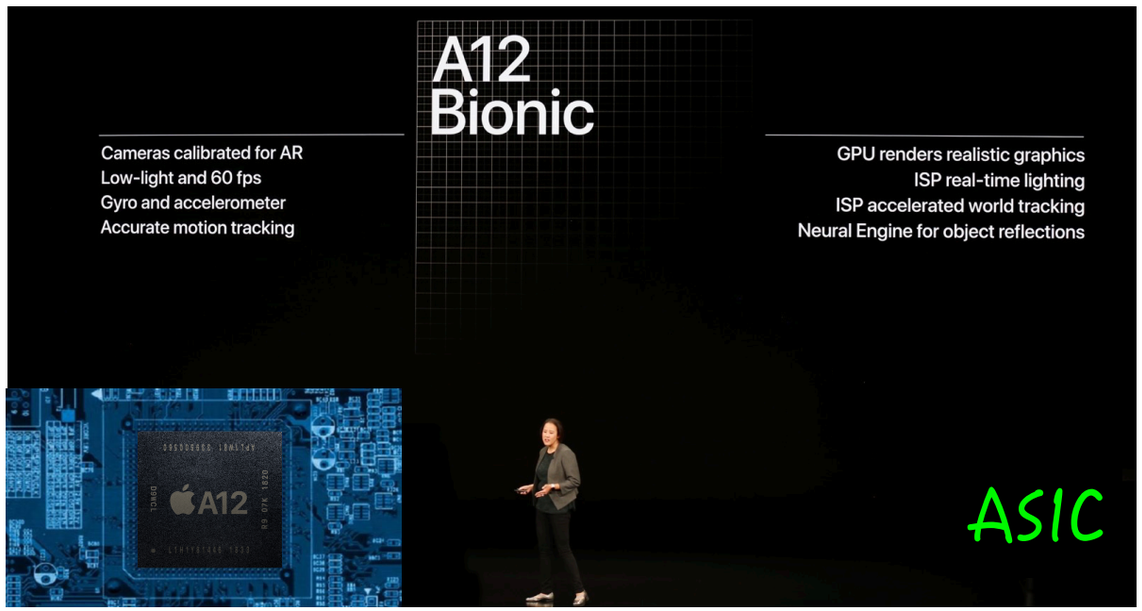
Industry Focus



# #TRENDING IN INDUSTRY



Specialized co-processor hardware for machine learning inference



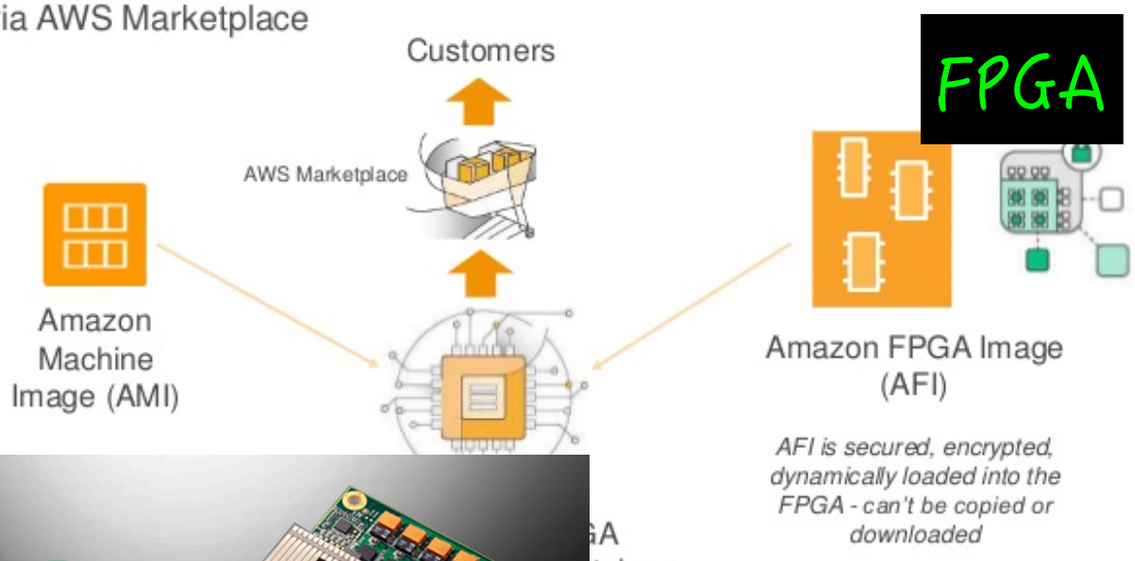


## INTEL® FPGA ACCELERATION HUB

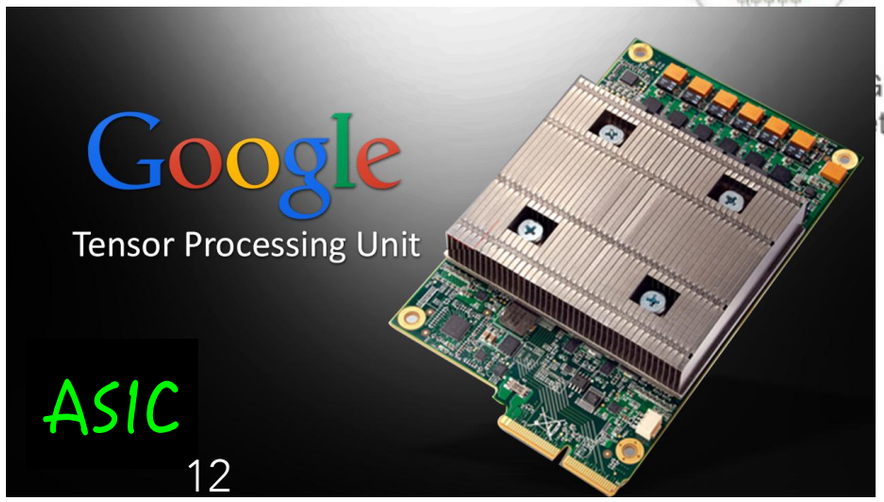
The Intel® Xeon® Acceleration Stack for FPGAs is a robust framework enabling data center applications to leverage an FPGA's potential to increase

FPGA

Microsoft Catapult/Brainwave

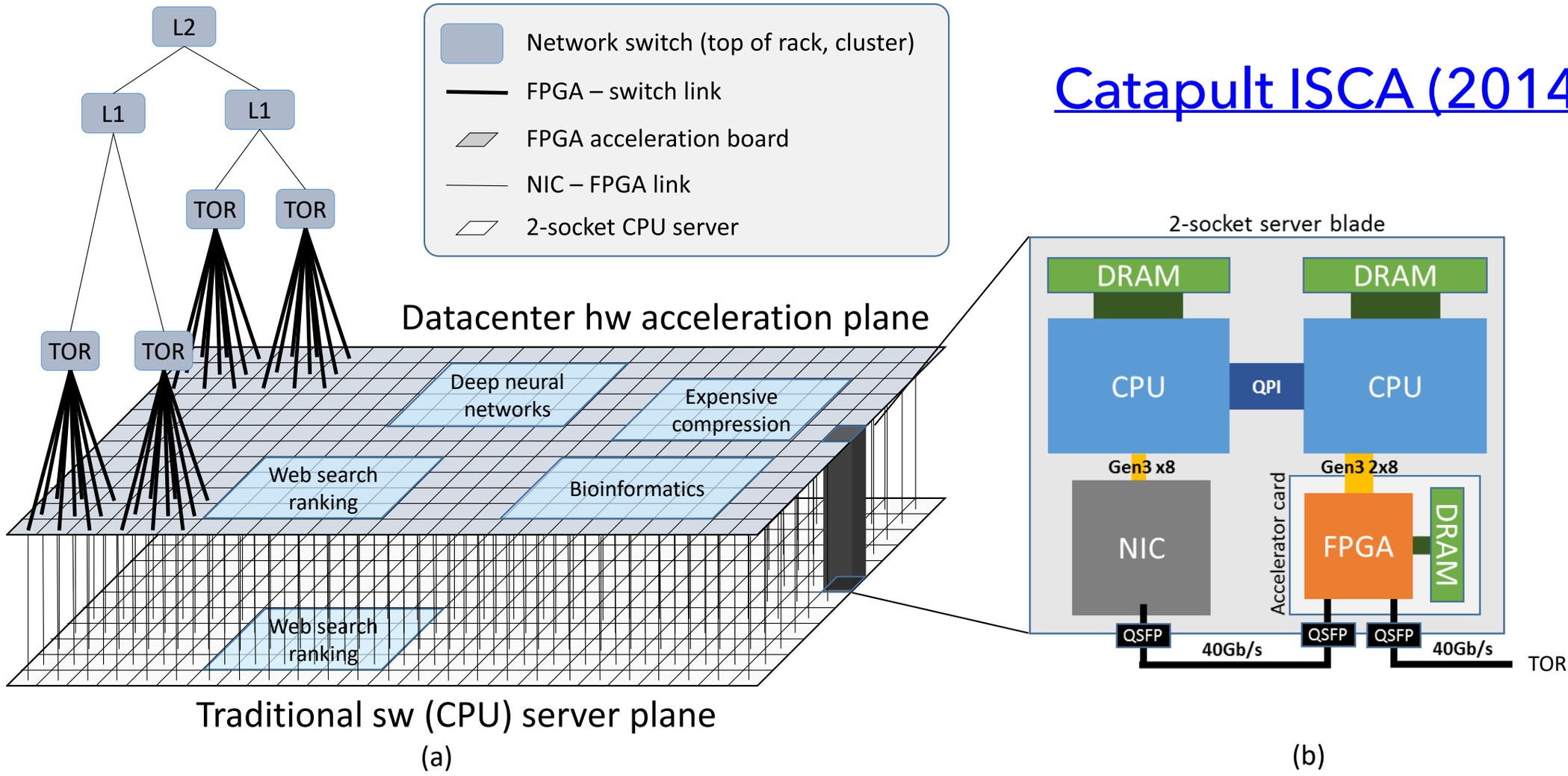




FPGA



# MICROSOFT BRAINWAVE



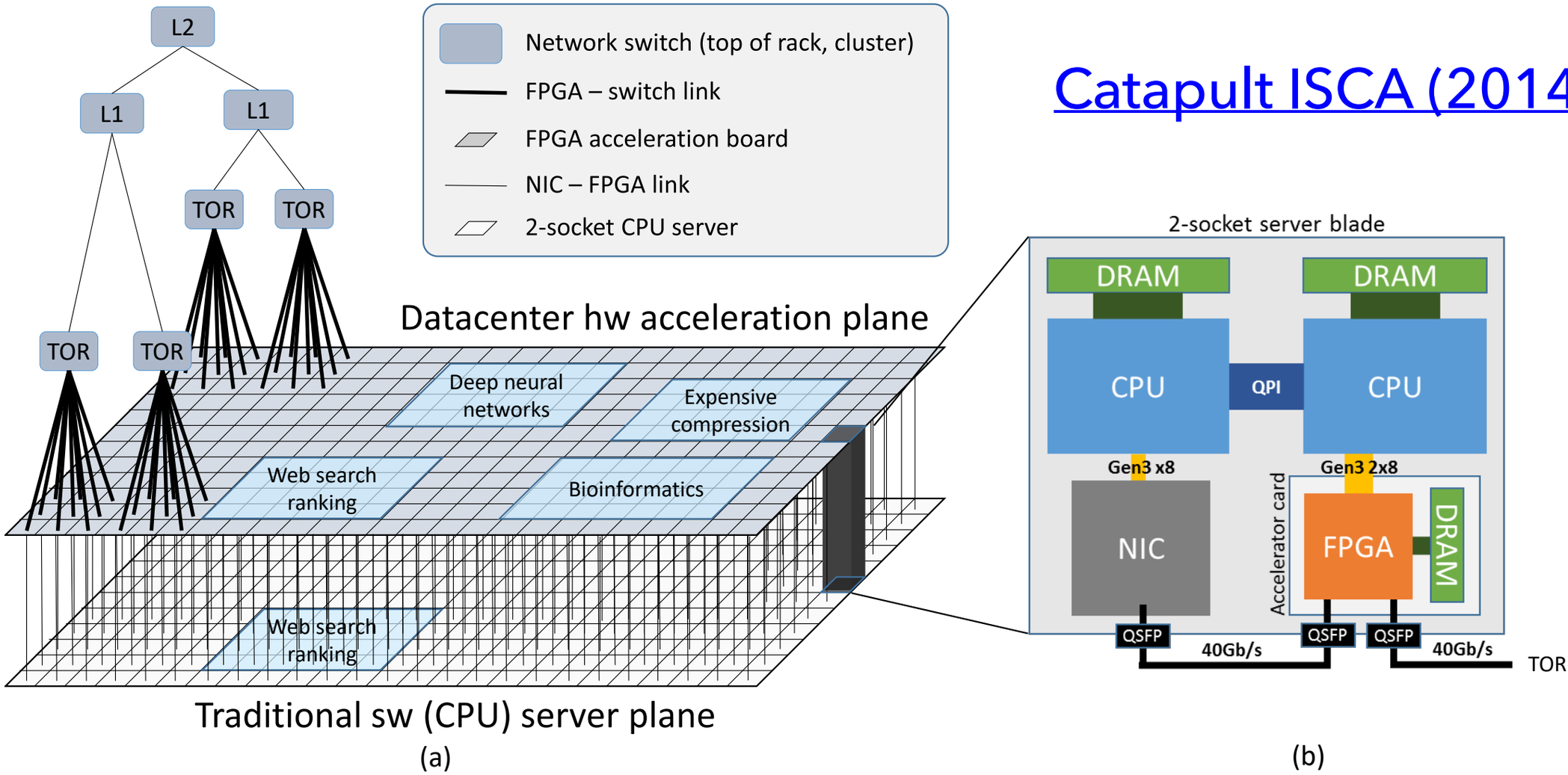



[Catapult ISCA (2014)]

- Service at scale
- FPGA/CPU fabric accelerates computing and network
- Re-train ***supported networks**** to optimize for different tasks

# MICROSOFT BRAINWAVE





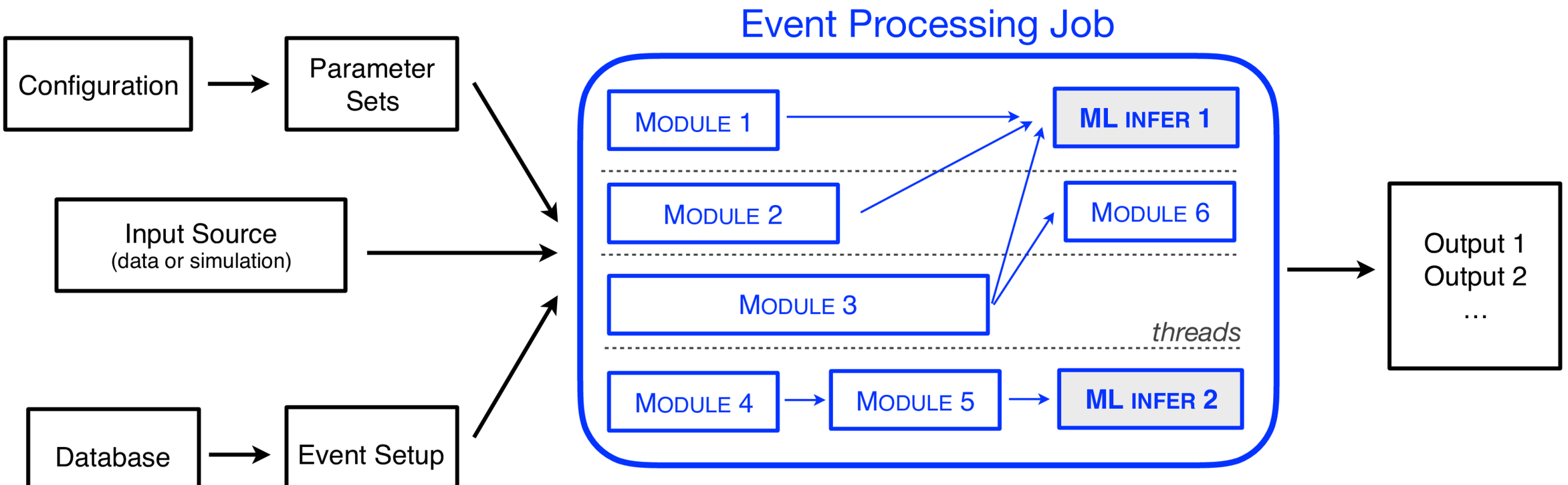

[Catapult ISCA (2014)]

- Service at scale
- FPGA/CPU fabric accelerates computing and network
- Re-train ***supported networks**** to optimize for different tasks

- ***Supported networks****
  - ResNet-50
  - ResNet-152
  - DenseNet-121
  - VGGNet-16

# PARTICLE PHYSICS COMPUTING MODEL



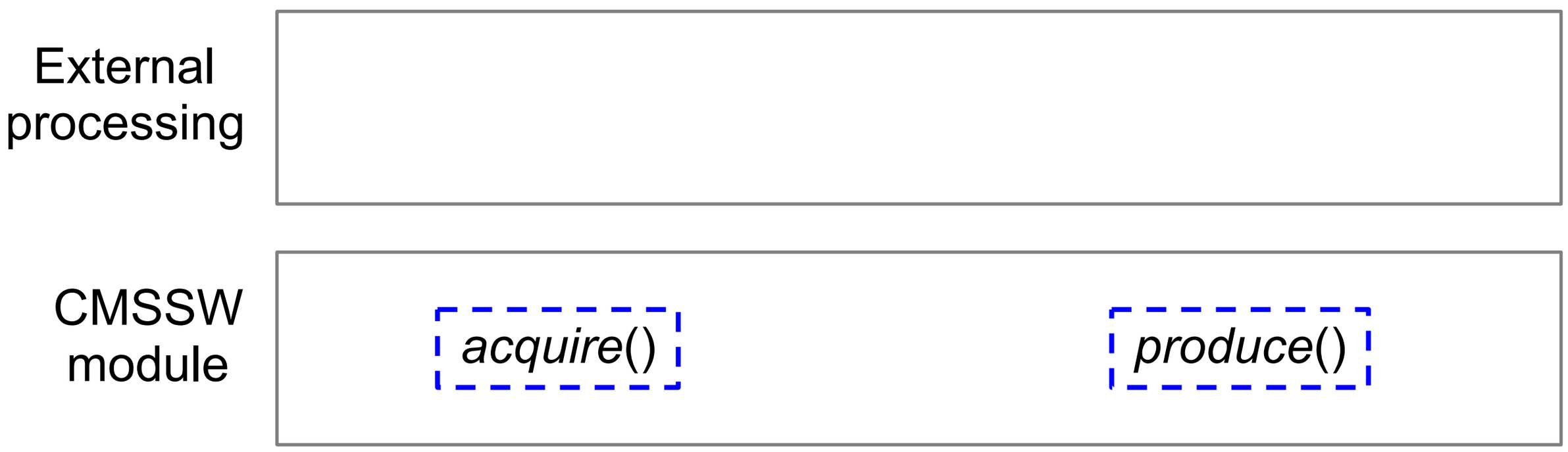



- Event-based processing
  - Events are complex with hundreds of “products”
  - Load one event into memory; execute all algorithms
- Most applications not well-suited for large batches (needed for best GPU performance)

# ACCESSING HETEROGENOUS RESOURCES



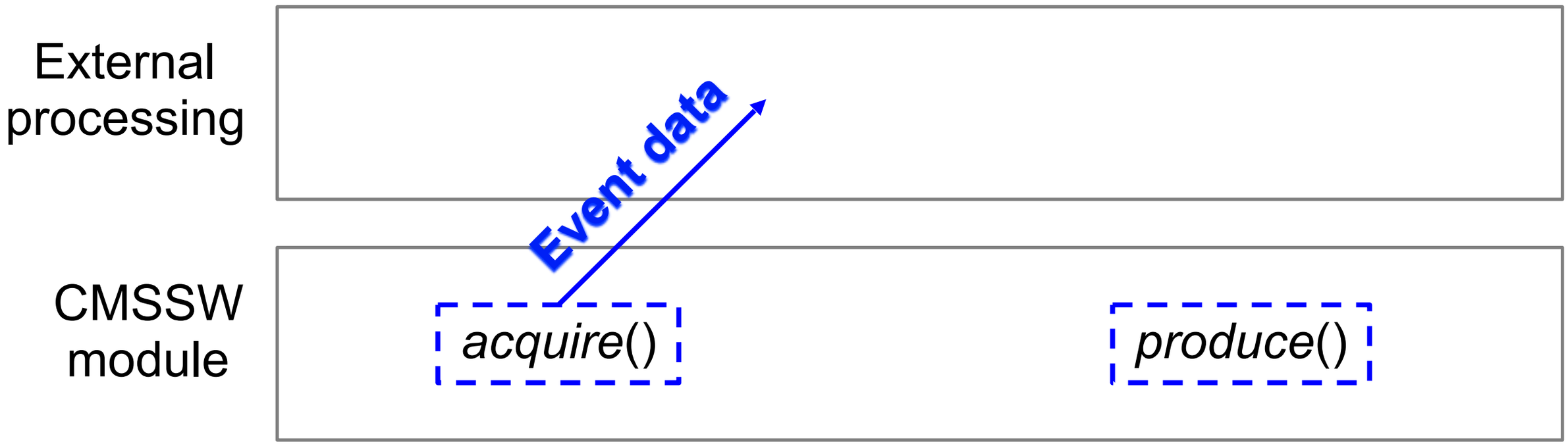
- New CMS software feature ***ExternalWork***
  - **Asynchronous** task-based processing



# ACCESSING HETEROGENOUS RESOURCES



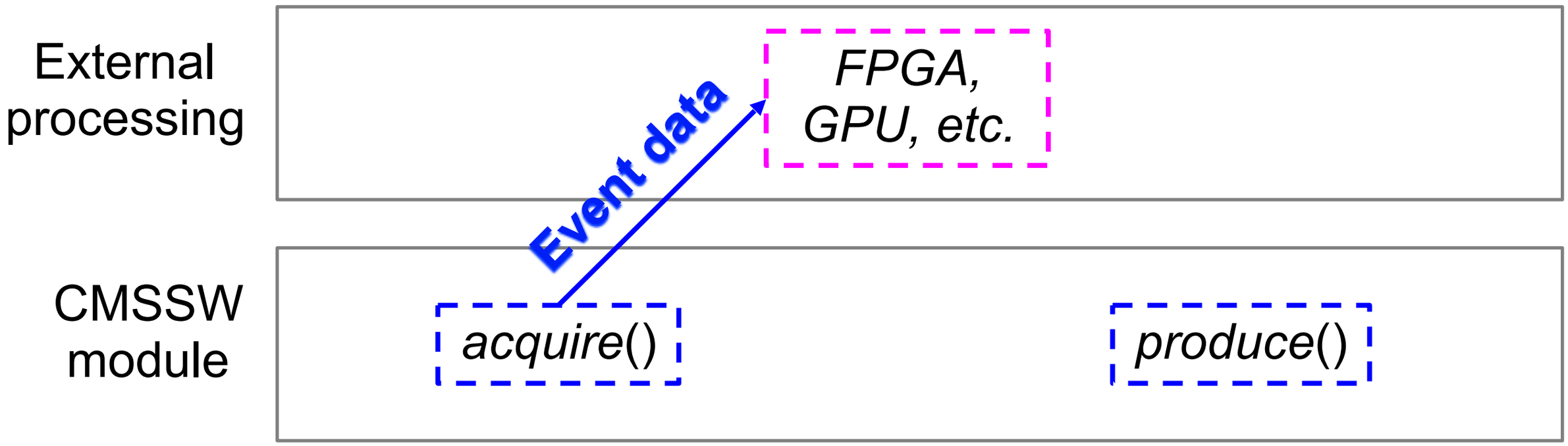
- New CMS software feature ***ExternalWork***
  - **Asynchronous** task-based processing



# ACCESSING HETEROGENOUS RESOURCES



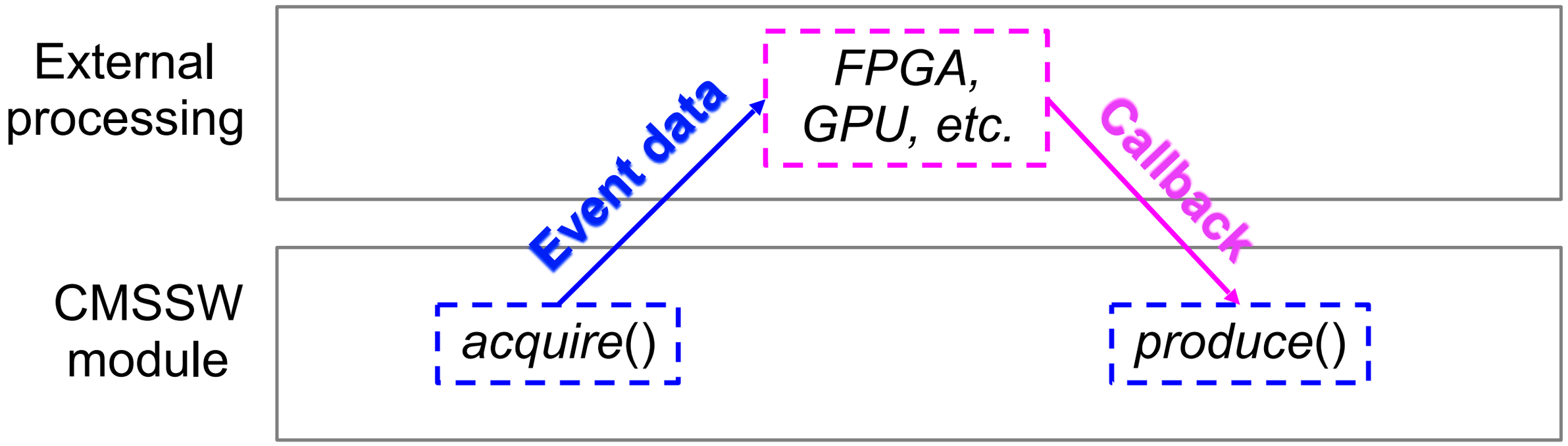
- New CMS software feature ***ExternalWork***
  - **Asynchronous** task-based processing



# ACCESSING HETEROGENOUS RESOURCES



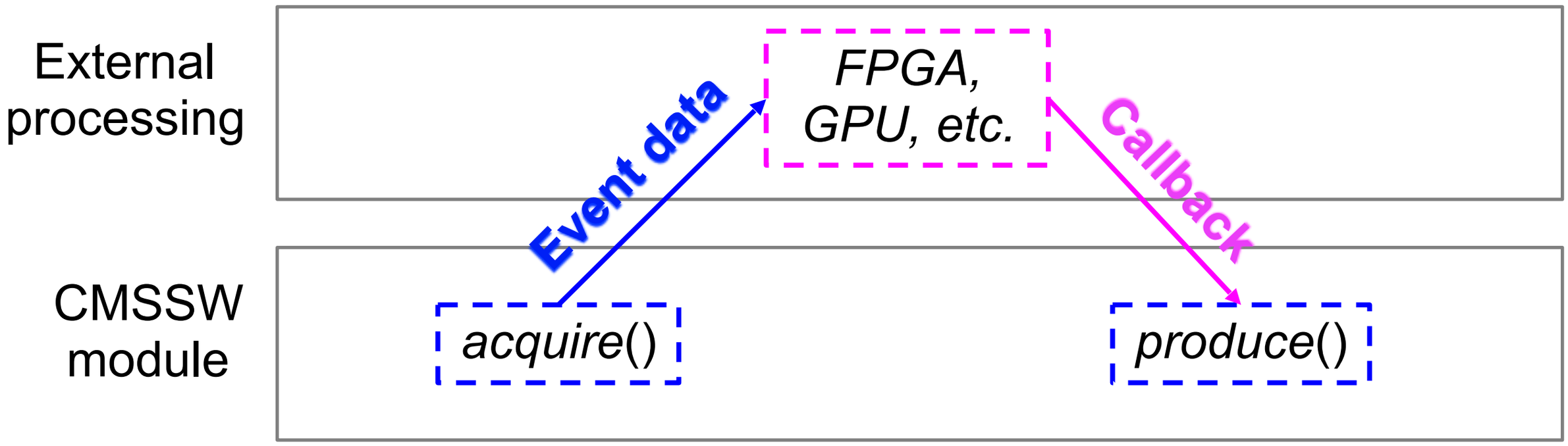
- New CMS software feature ***ExternalWork***
  - **Asynchronous** task-based processing



# ACCESSING HETEROGENOUS RESOURCES



- New CMS software feature ***ExternalWork***
  - **Asynchronous** task-based processing




- **Non-blocking**: schedule other tasks during external processing
- Can be used with GPUs, FPGAs, cloud, …

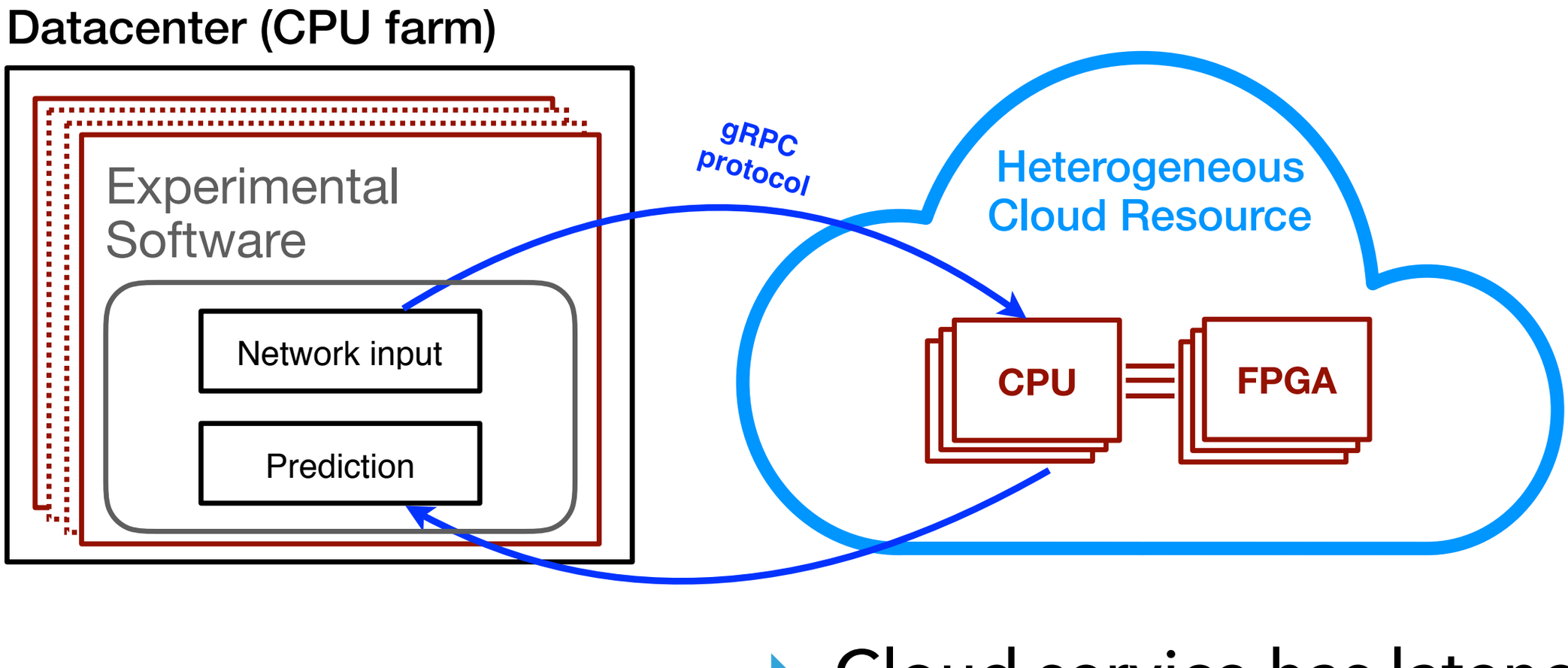
# SONIC IN CMS SOFTWARE



- Services for Optimized Network Inference on Coprocessors (SONIC)
  - Convert experimental data into neural network input
  - Send neural network input to coprocessor using communication protocol
  - Use ExternalWork for asynchronous requests
- Currently supports:
  - gRPC communication protocol
  - TensorFlow with inputs sent as TensorProto (protobuf)

# CLOUD VS. EDGE

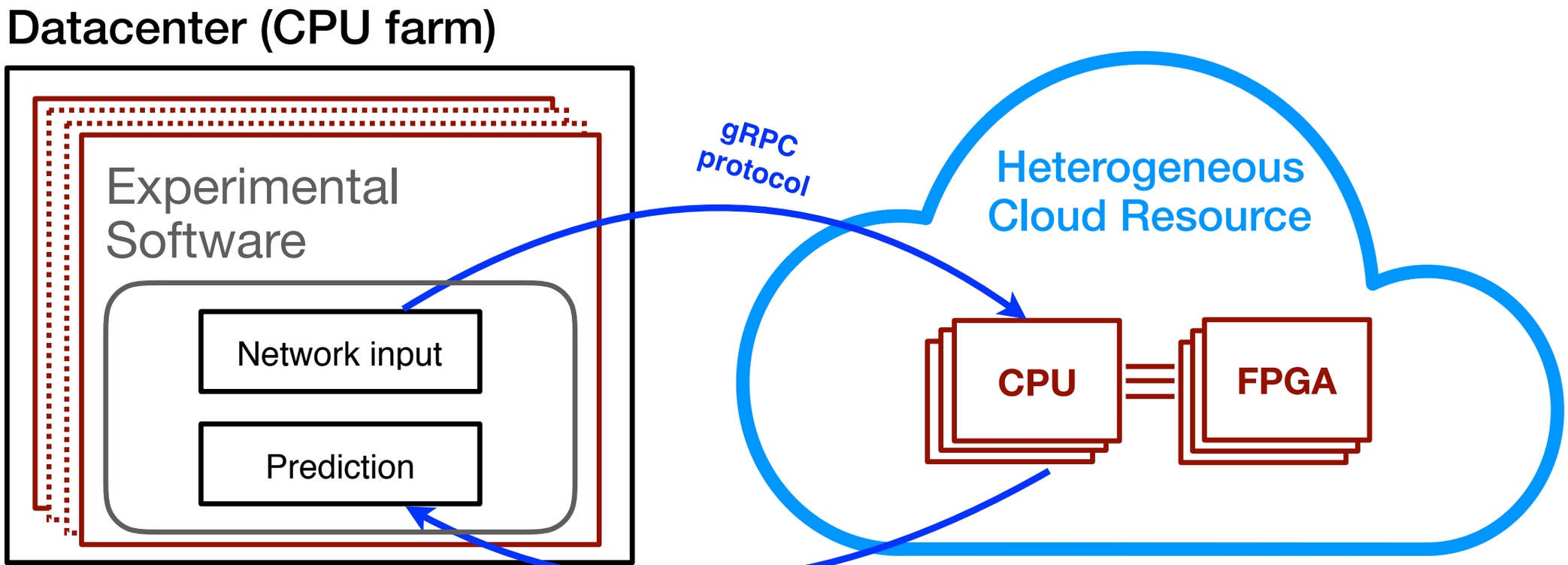


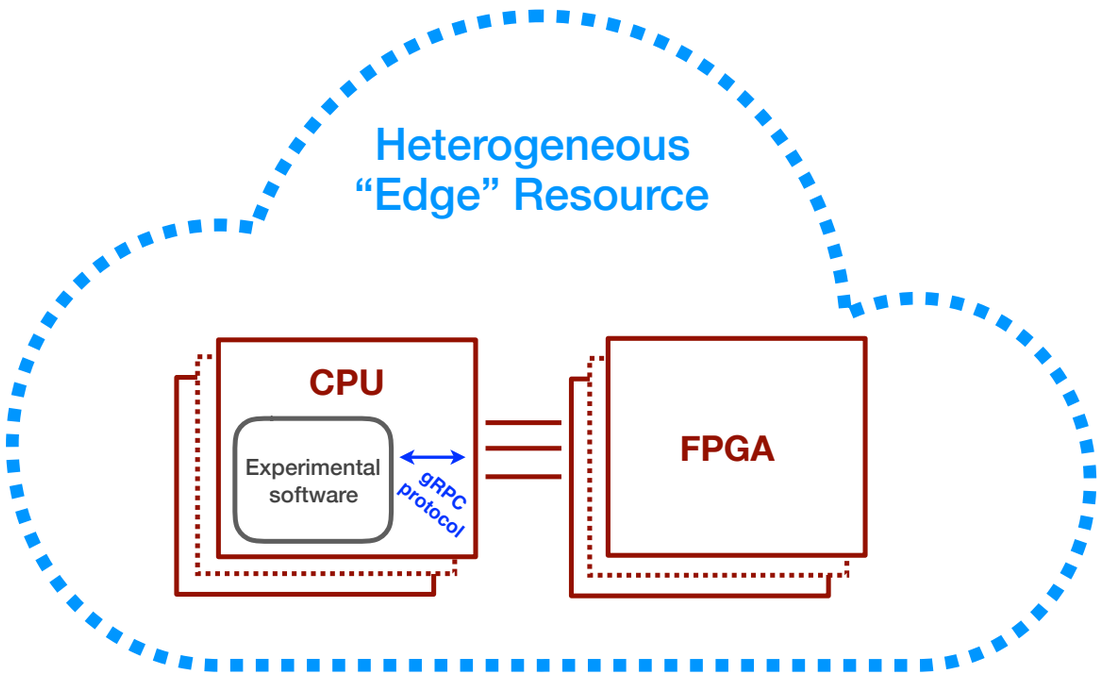



- Cloud service has latency

# CLOUD VS. EDGE



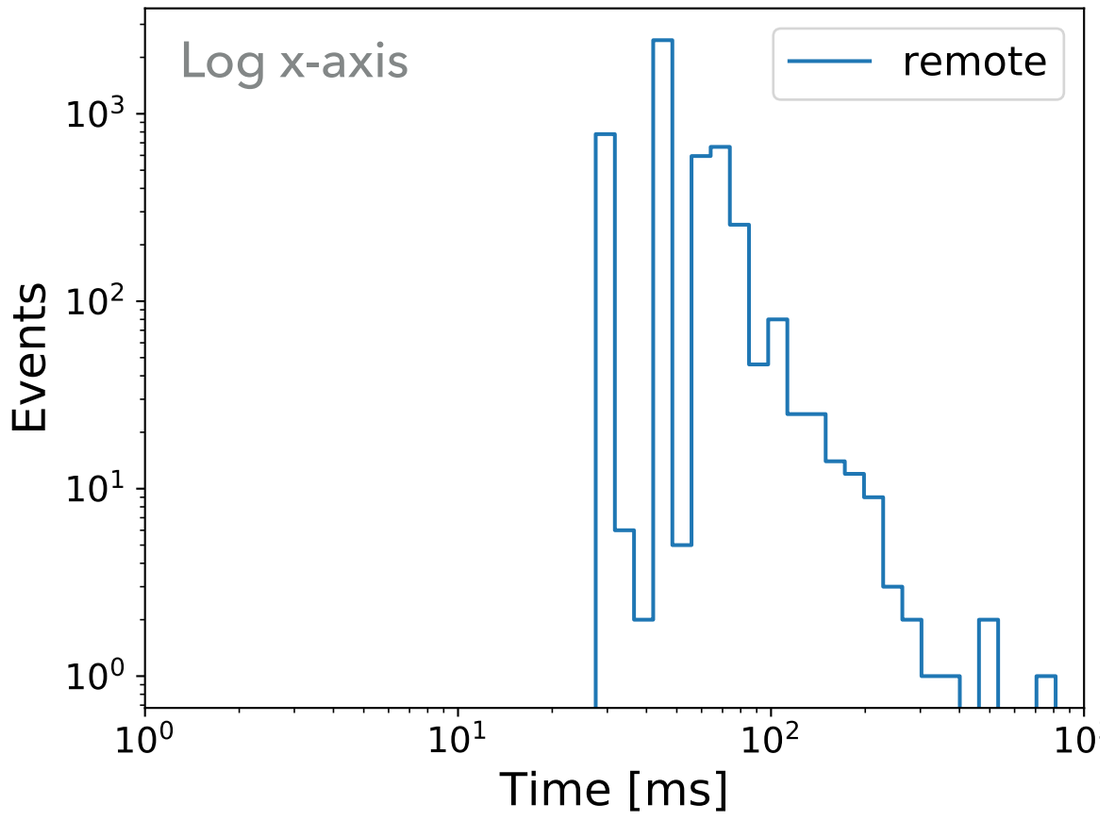


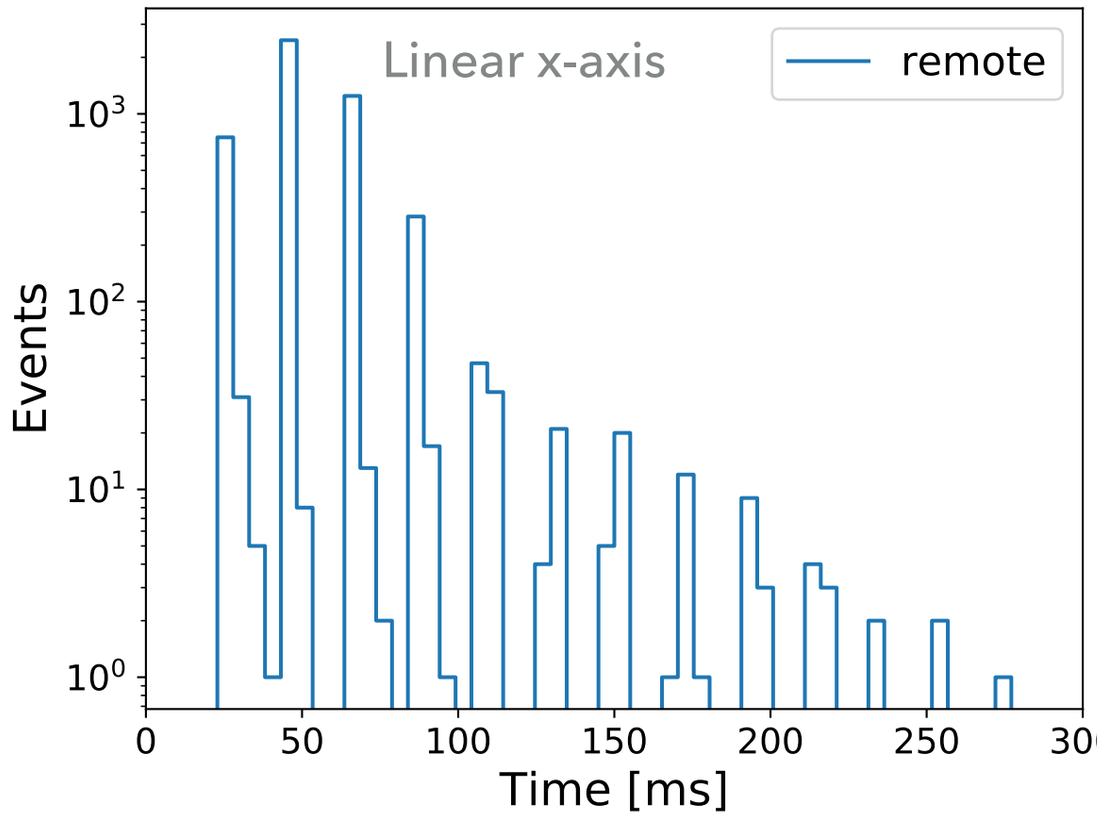




- Cloud service has latency
- Run CMSSW on CPU in FPGA datacenter → emulate local installation of FPGAs ("on-prem" or "edge")
  - Test of ultimate perf.

# SONIC LATENCY



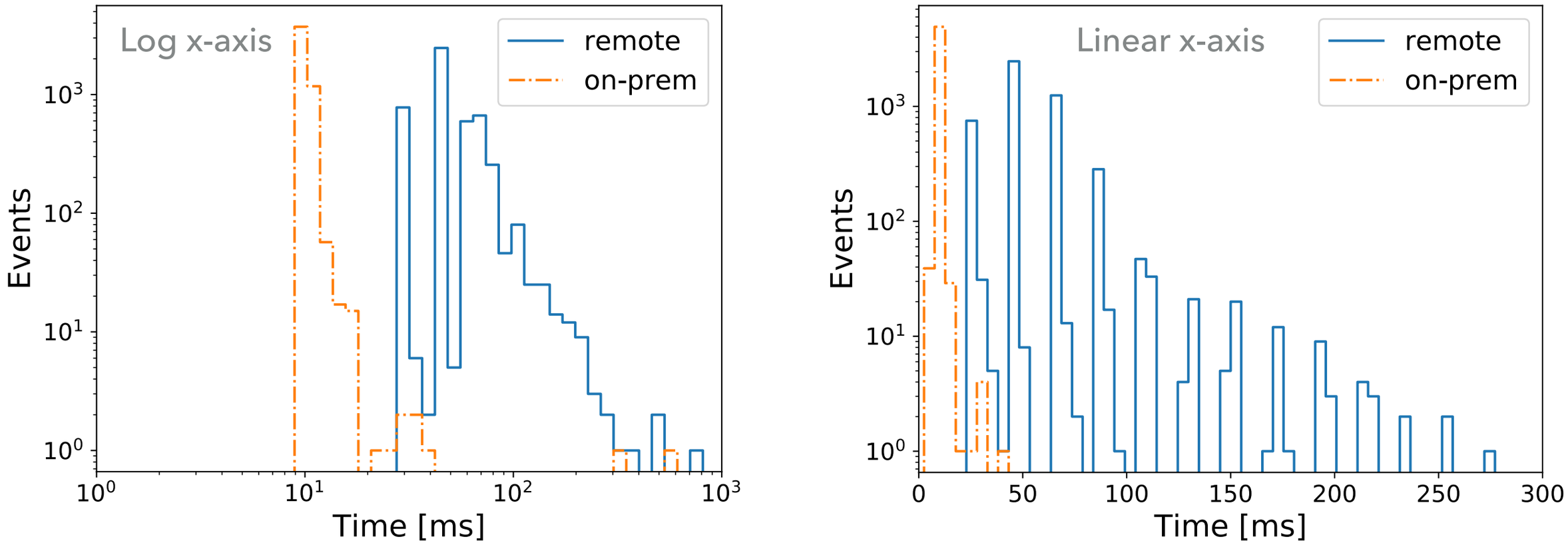



- Remote: FNAL (IL) to Azure (VA) ‹time› = 60 ms
  - Highly dependent on network conditions

# SONIC LATENCY

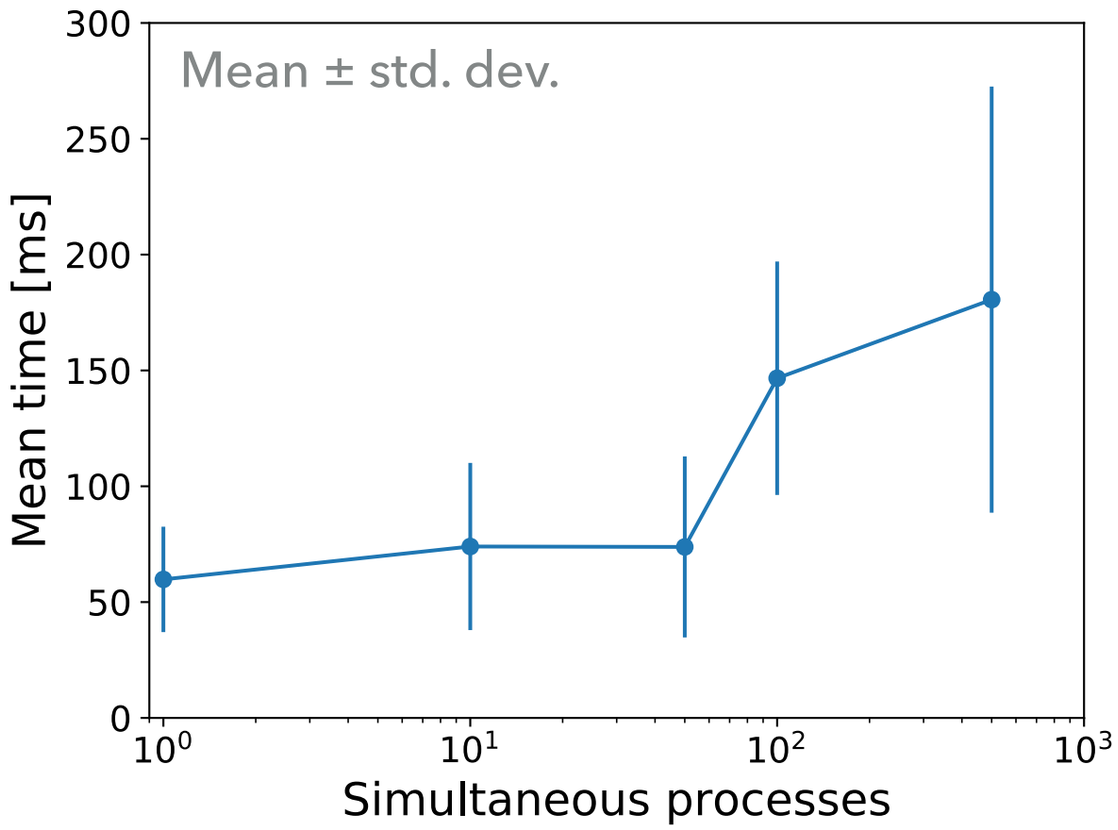


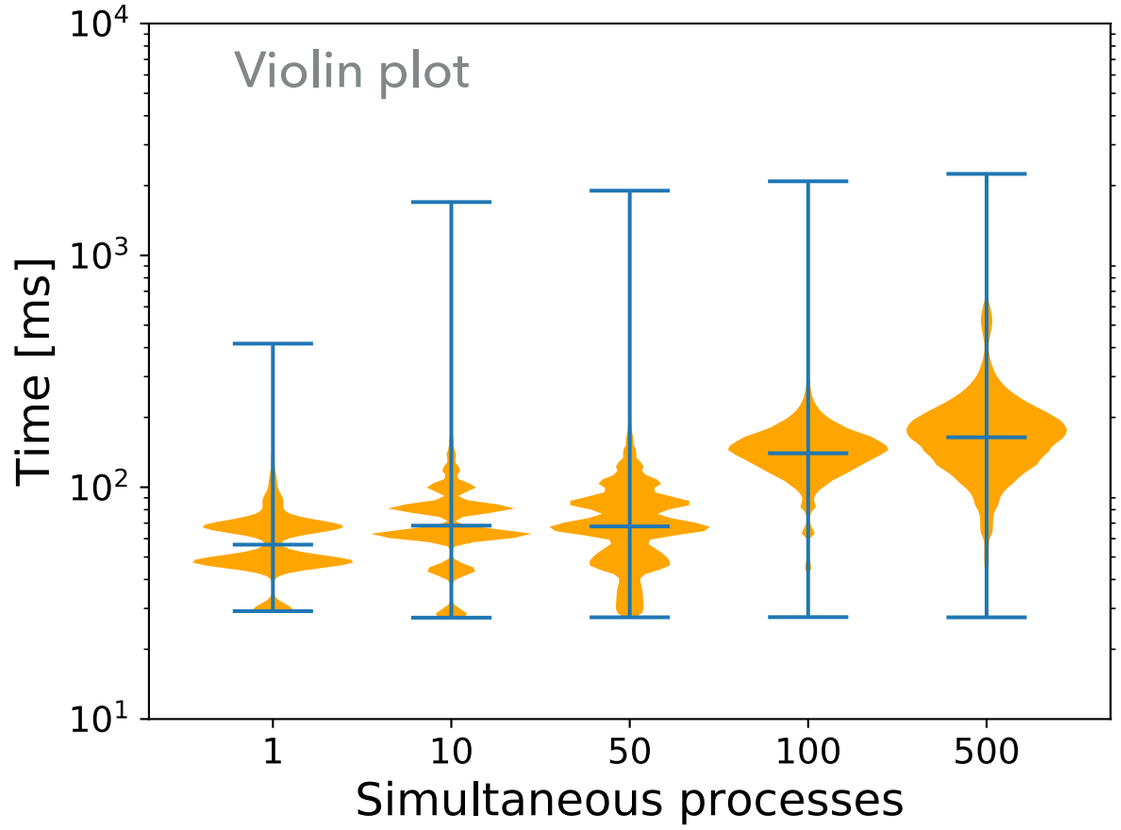



- Remote: FNAL (IL) to Azure (VA) ‹time› = 60 ms
  - Highly dependent on network conditions
- On-prem: run CMSSW on Azure ‹time› = 10 ms
  - on FPGA: 1.8 ms for inference
  - Remaining time used for classifying and I/O

# SONIC LATENCY: SCALING



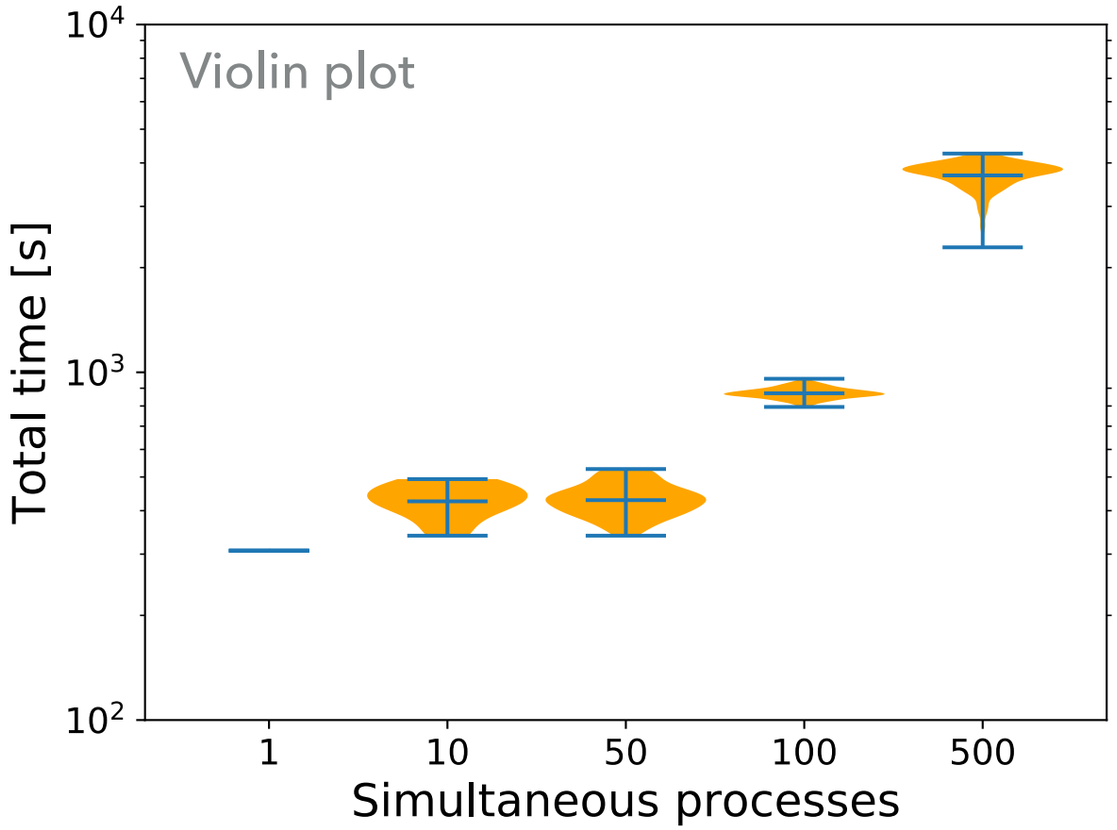



- *N* simultaneous processes, sending 1 request to 1 Brainwave service
- Processes only run SONIC → "worst case" scenario
  - Standard reconstruction has many other non-SONIC modules
- Moderate increases in mean, std. dev., and long tail for latency
  - Stable up to $N = 50$

# SONIC THROUGHPUT

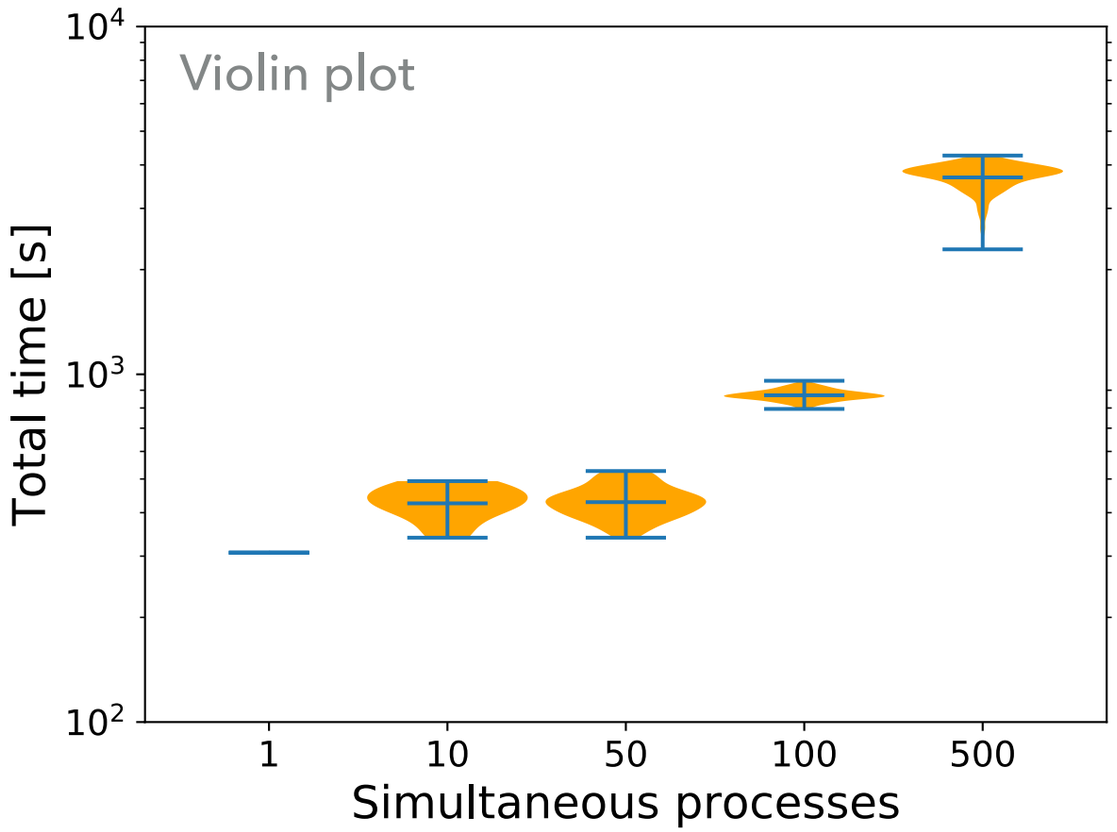


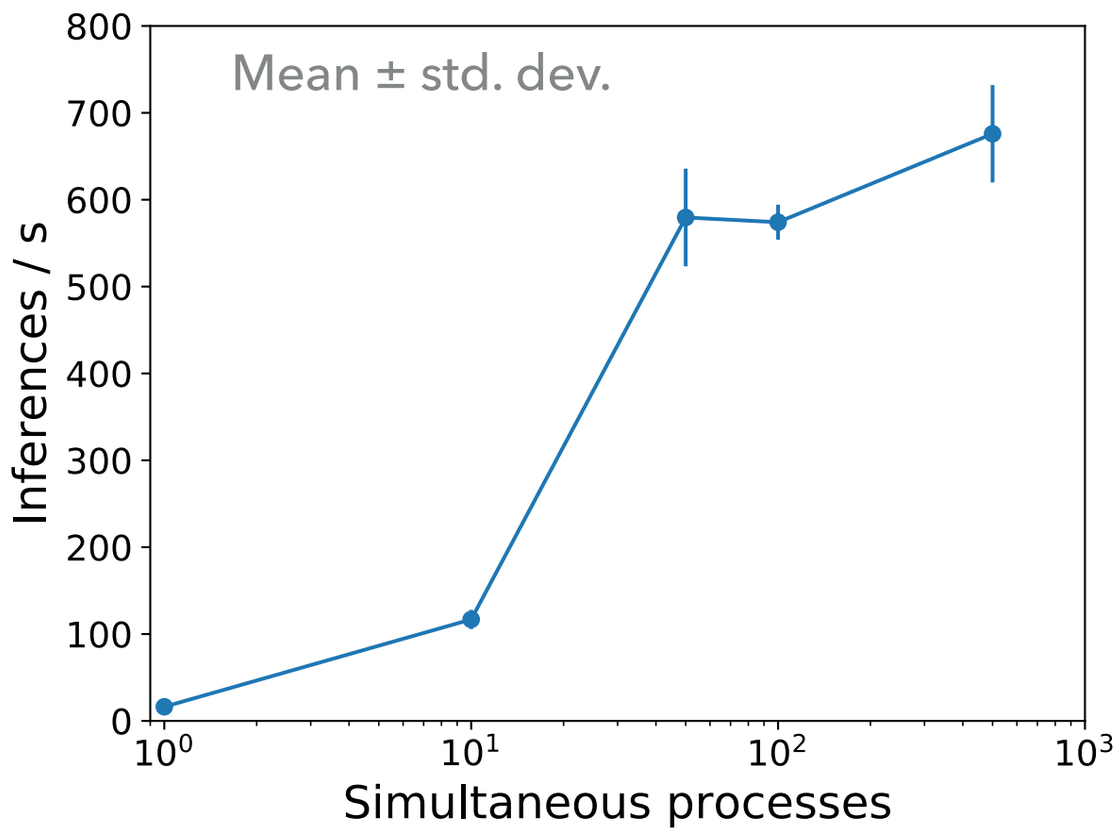



- Each process evaluates 5000 jet images in series
  - Consistent total time for each process to complete (good load balancing)

# SONIC THROUGHPUT



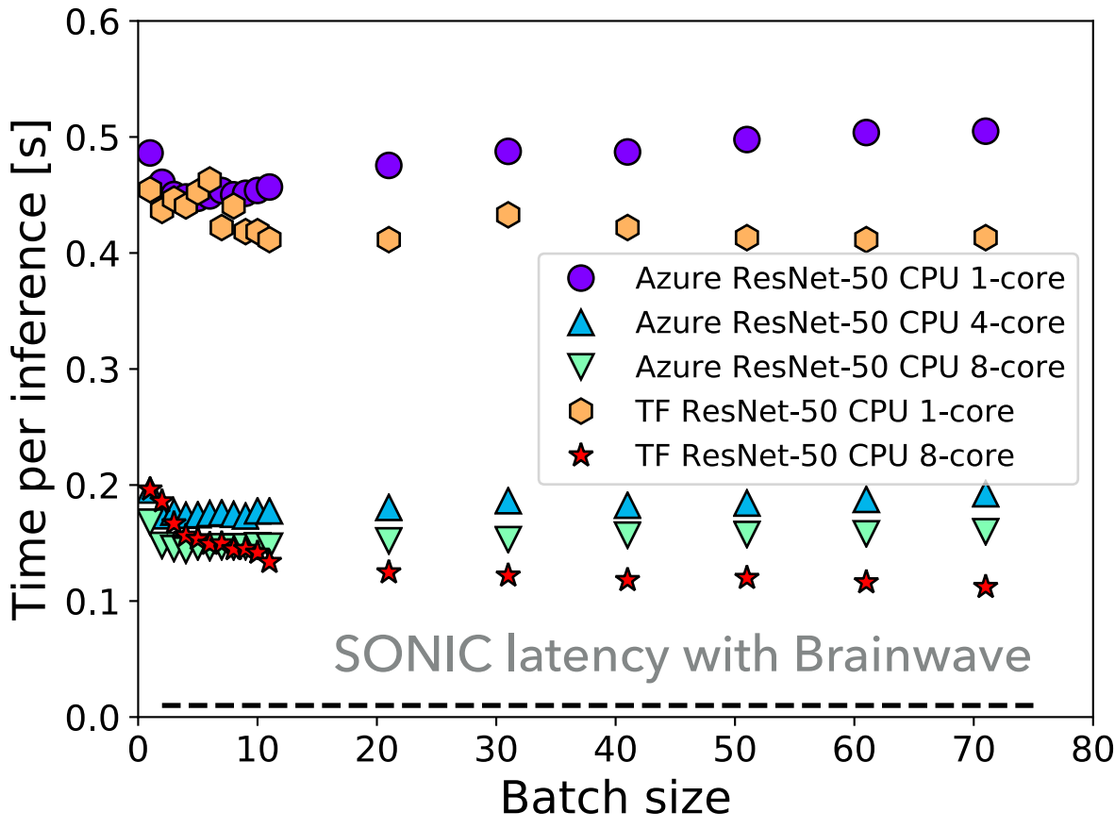


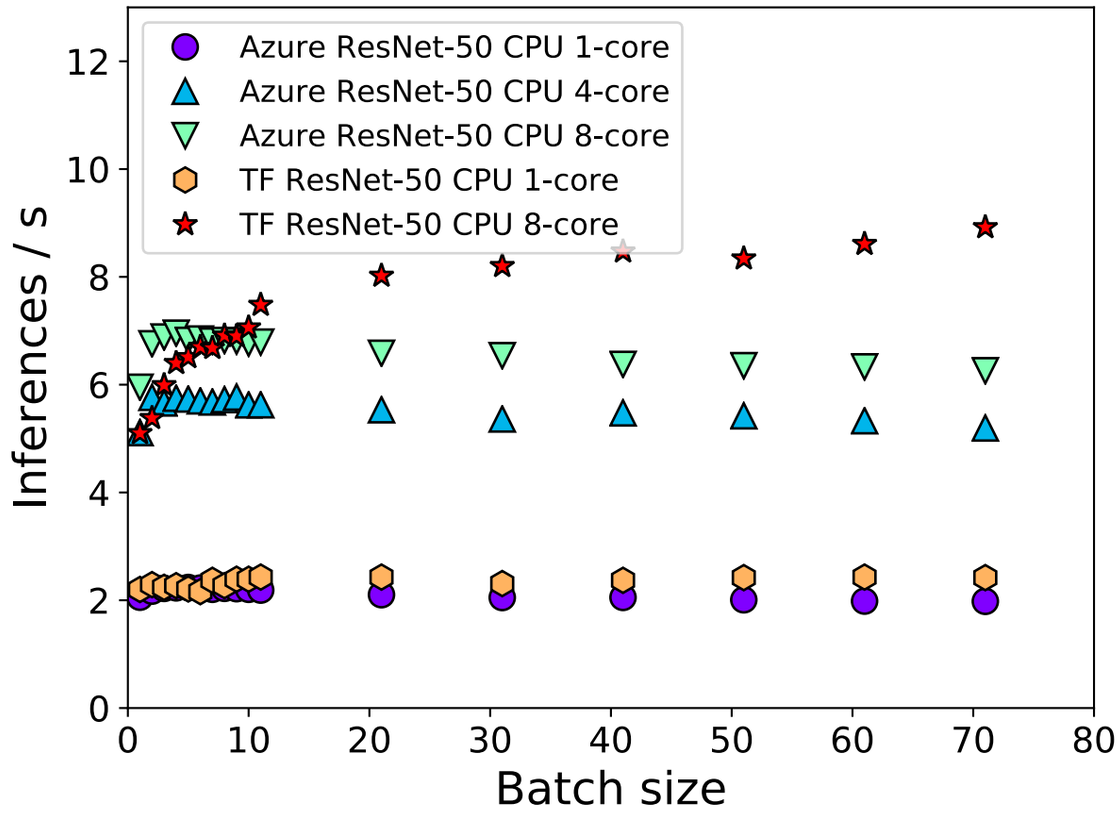




- Each process evaluates 5000 jet images in series
  - Consistent total time for each process to complete (good load balancing)
- Compute inferences per second as $(5000 \cdot N)/(\text{total time})$
  - $N = 50$ fully occupies FPGA
  - **Throughput** up to ~660 inferences per second

# CPU PERFORMANCE



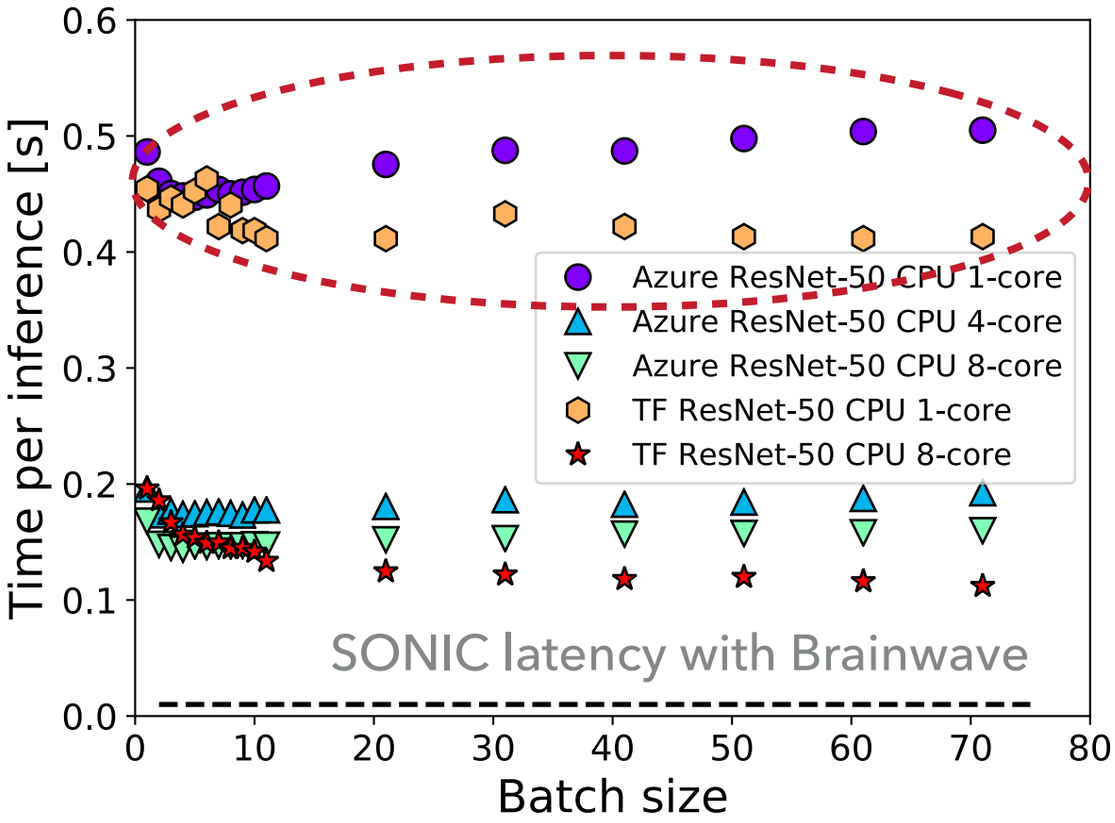


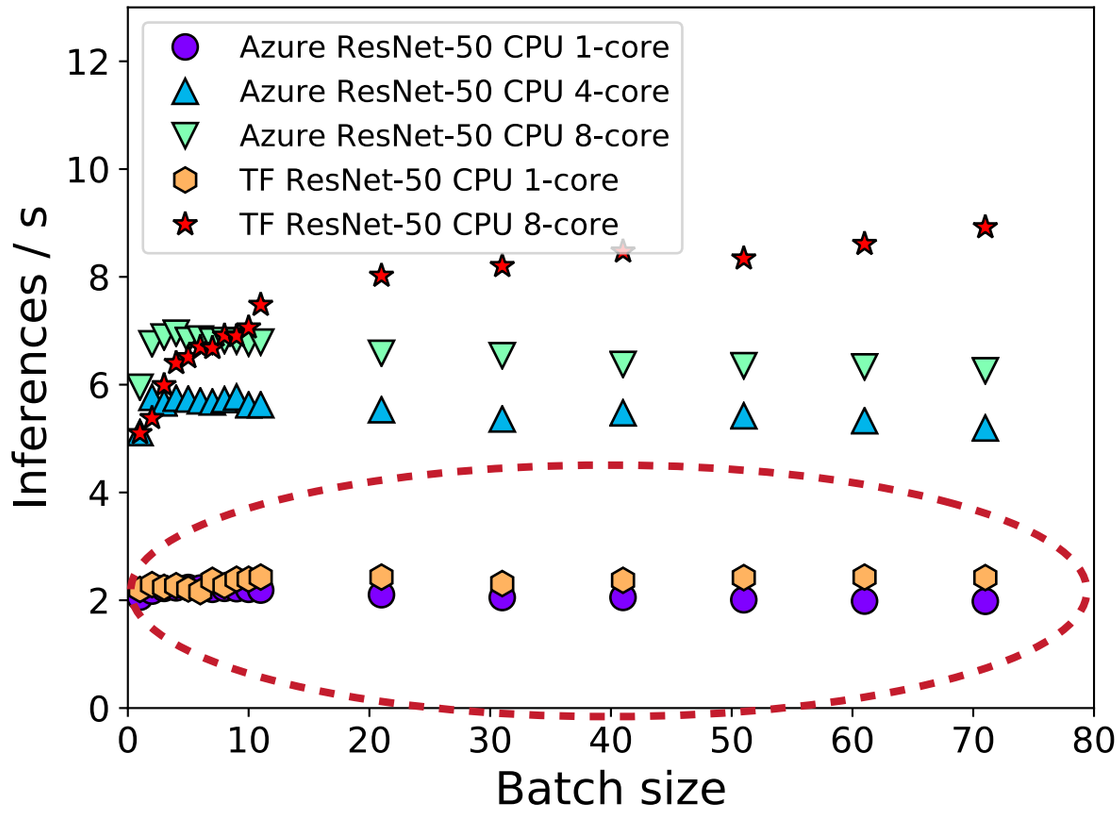




- Standard CMS CPU: 5 min to import ResNet-50, then 1.75 s/inference

# CPU PERFORMANCE



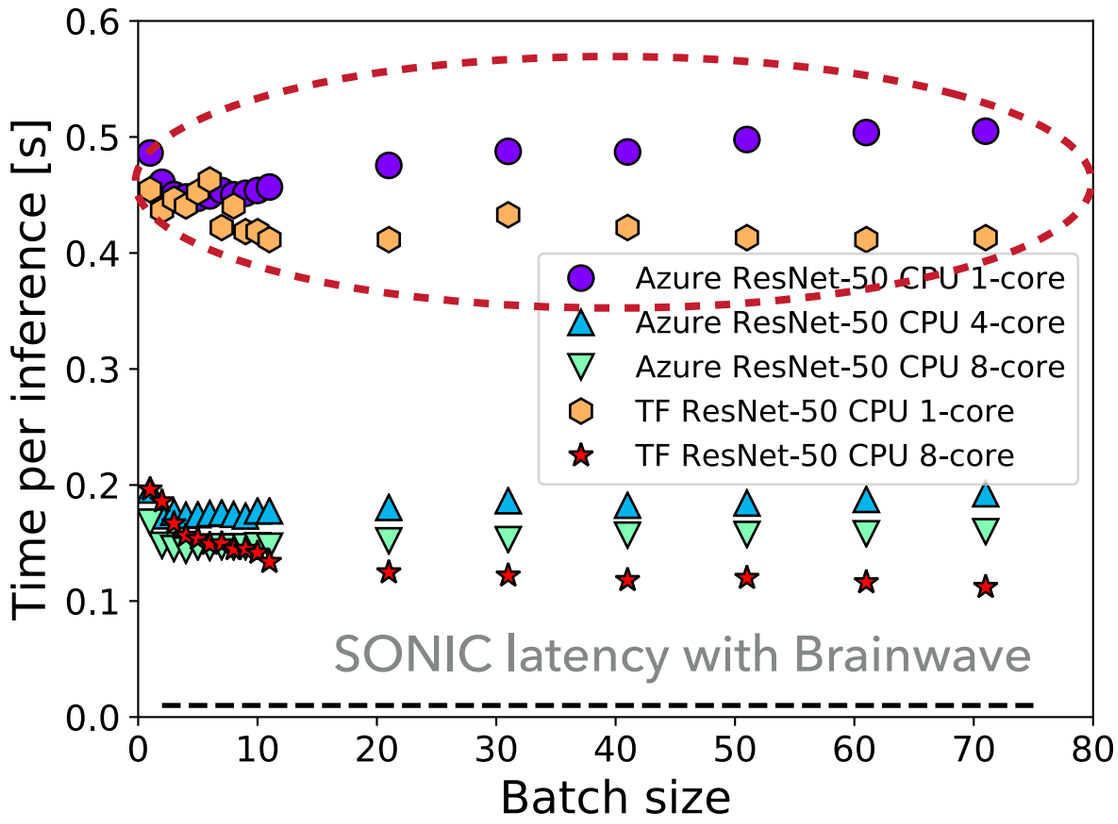


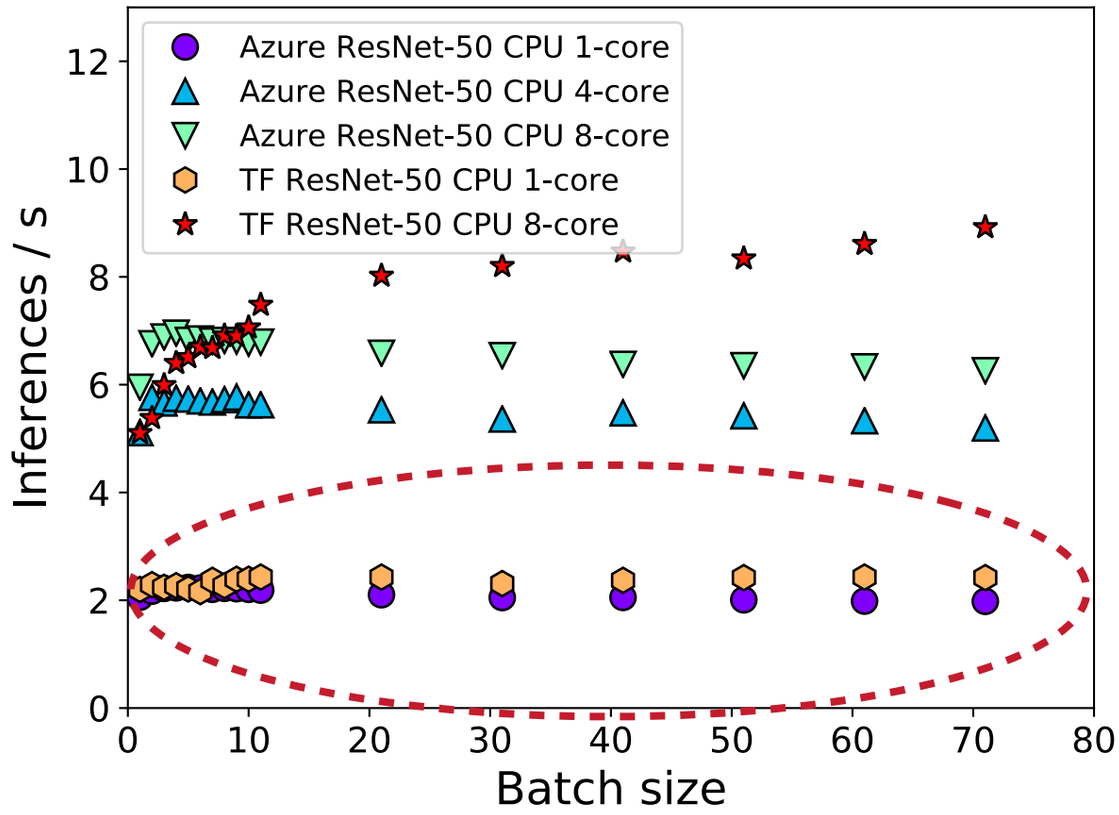




- Standard CMS CPU: 5 min to import ResNet-50, then 1.75 s/inference
- 1-core better CPU: **500 ms/inference** (best achievable reference given computing model in CMS)

# CPU PERFORMANCE



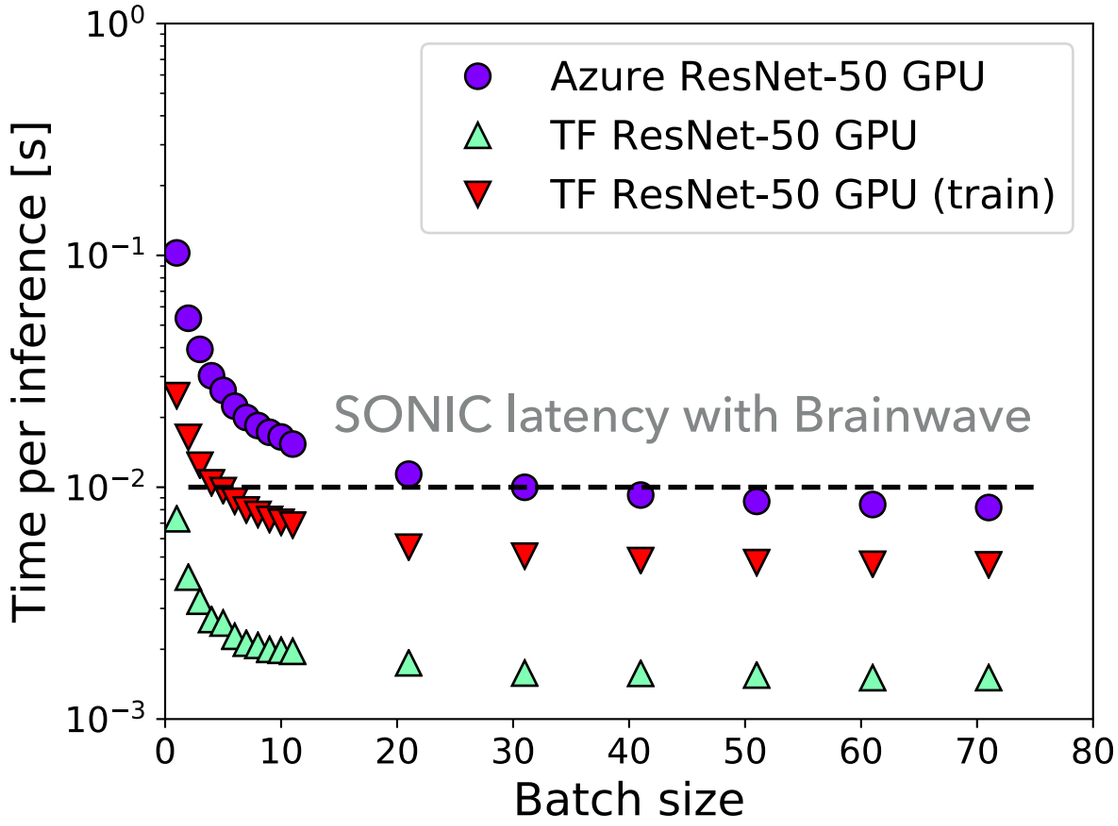


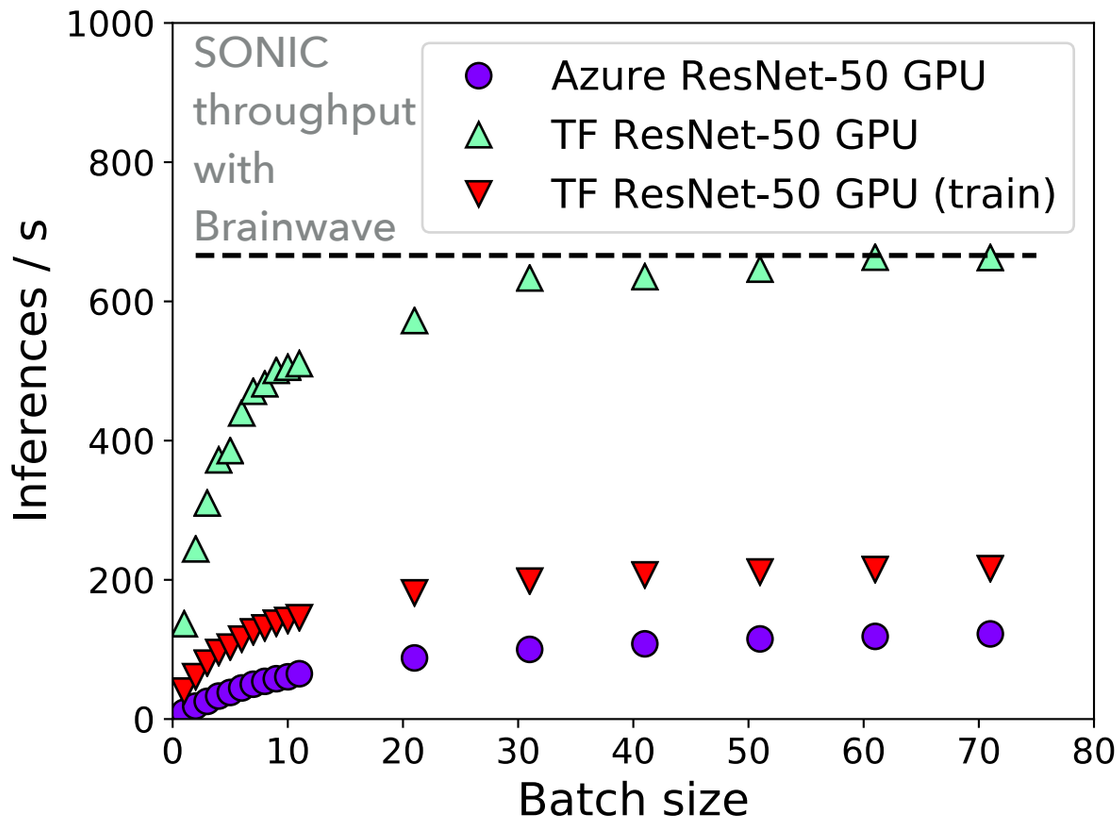




- Standard CMS CPU: 5 min to import ResNet-50, then 1.75 s/inference
- 1-core better CPU: **500 ms/inference** (best achievable reference given computing model in CMS)
- 8-core better CPU: 200 ms/inference

# GPU PERFORMANCE



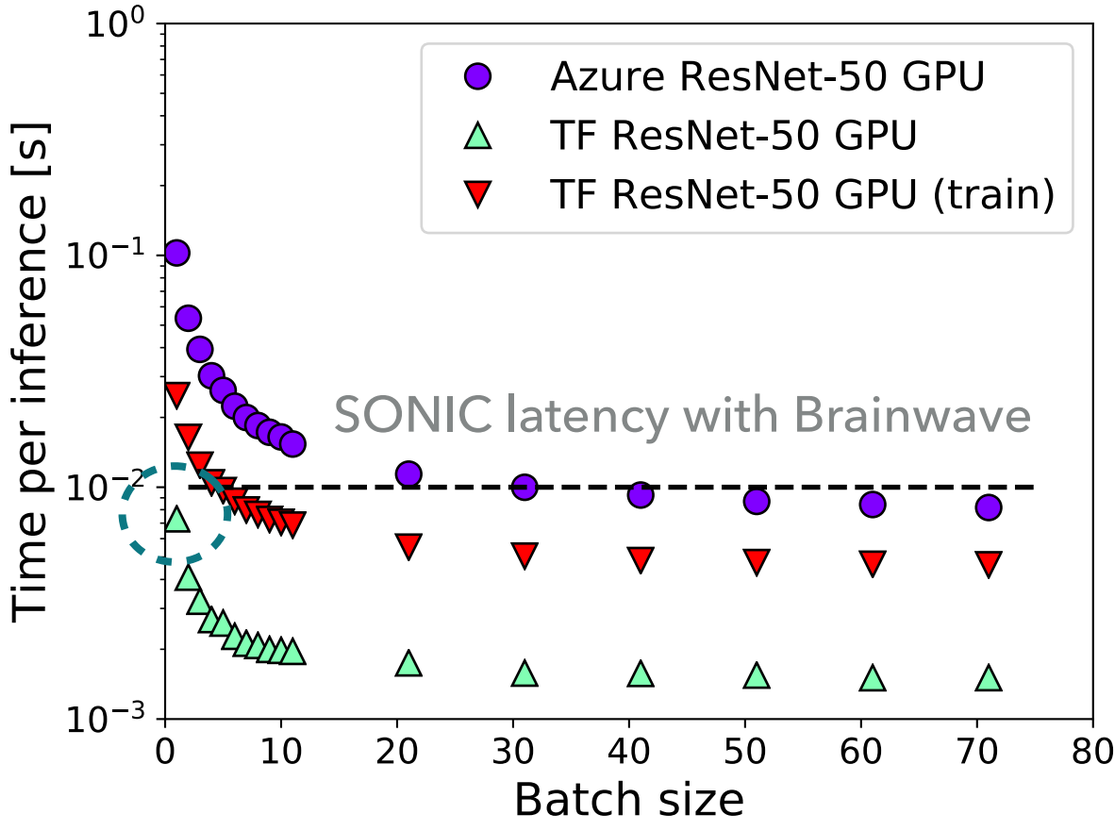


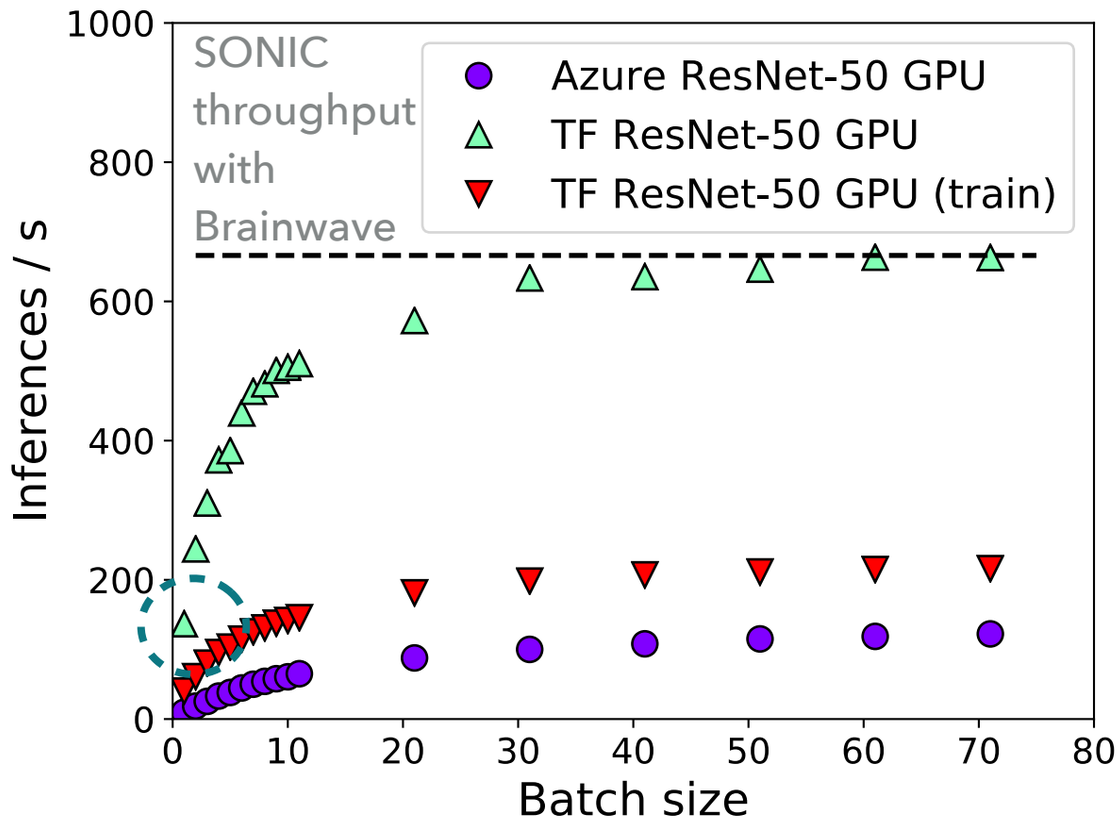




- GPU: NVIDIA GTX 1080, TensorFlow v1.10, connected to CPU via PCIe
  - TF built-in version of ResNet-50 (better optimized)

# GPU PERFORMANCE



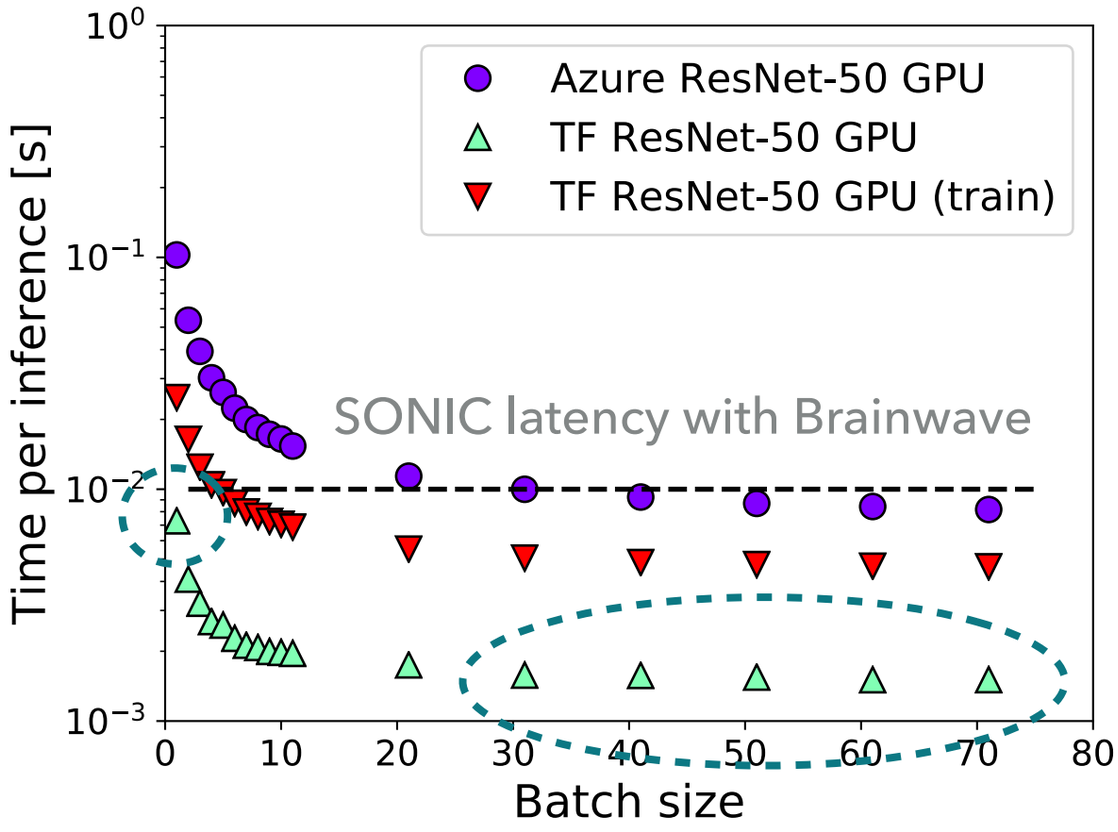


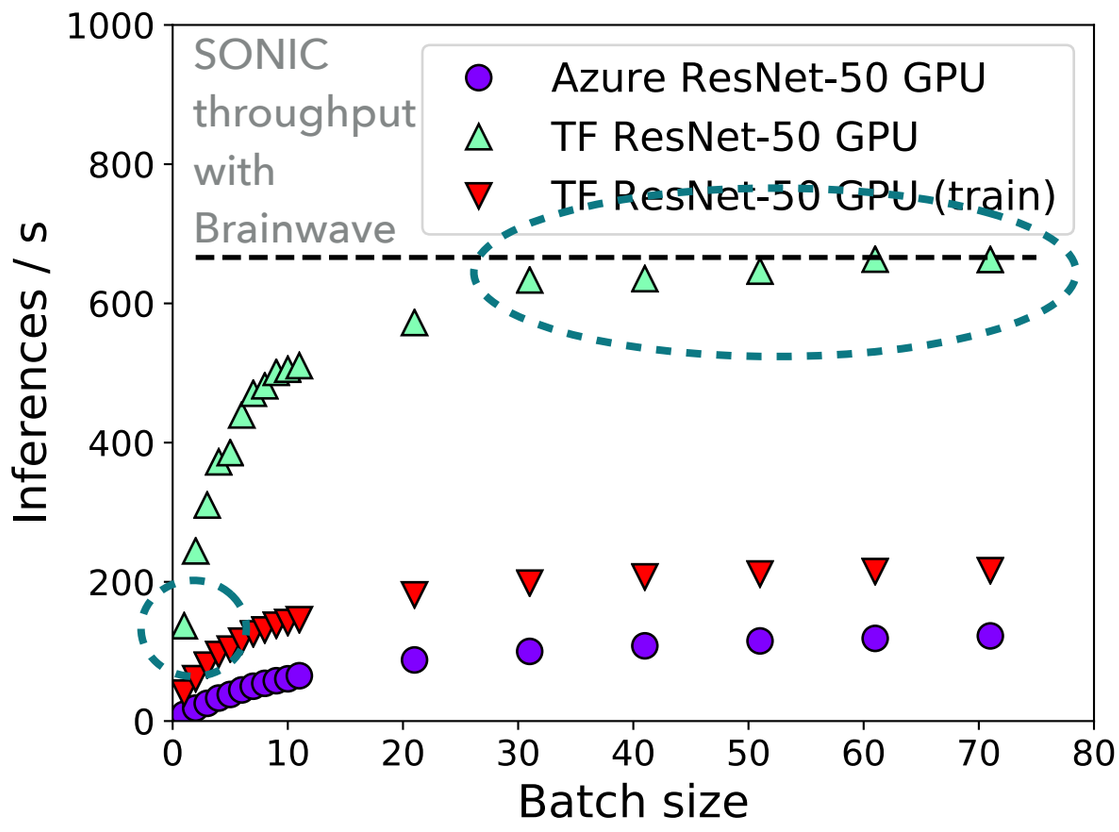




- GPU: NVIDIA GTX 1080, TensorFlow v1.10, connected to CPU via PCIe
  - TF built-in version of ResNet-50 (better optimized)
  - batch size 1: 7 ms/inference & 143 inferences/s

# GPU PERFORMANCE





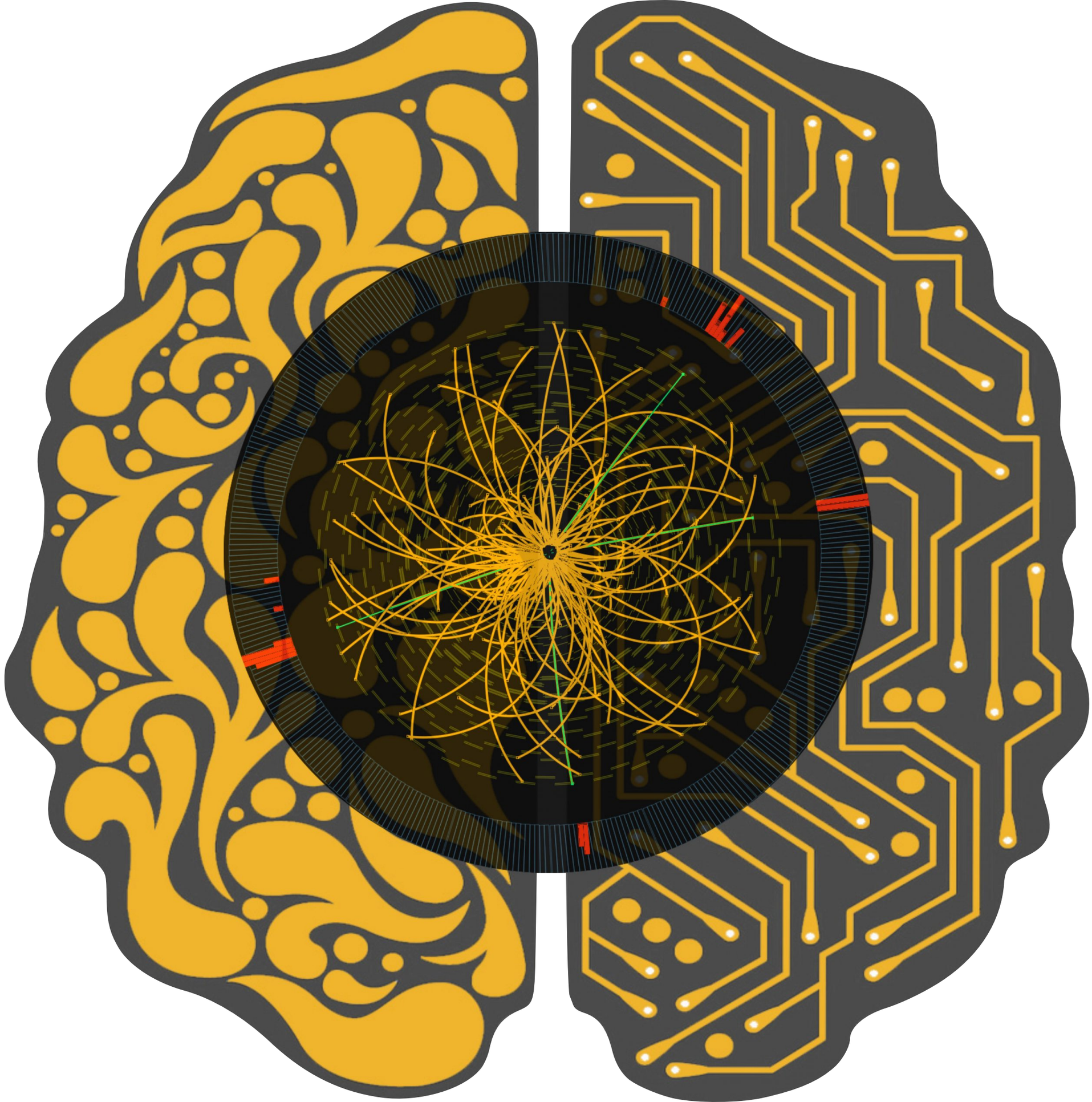

- GPU: NVIDIA GTX 1080, TensorFlow v1.10, connected to CPU via PCIe
  - TF built-in version of ResNet-50 (better optimized)
  - batch size 1: 7 ms/inference & 143 inferences/s
  - batch size 32: 1.5 ms/inference & 667 inferences/s

# PERFORMANCE COMPARISONS



| Type | Note | Latency [ms] | Throughput [inf./s] |
|---|---|---|---|
| CPU* | Xeon 2.6 GHz | 1750 | 0.6 |
| | i7 3.6 GHz | 500 | 2 |
| GPU** | batch = 1 | 7 | 143 |
| | batch = 32 | 1.5 | 667 |
| Brainwave | remote | 60 | 660 |
| | on-prem | 10 (1.8 on FPGA) | 660 |

- *CPU performance depends on
  - clock speed, TensorFlow version, # threads (=1 here)
- **GPU caveats
  - Directly connected to CPU via PCIe – not as a service
  - Depends on batch size & optimization of ResNet-50 network
- **Brainwave + SONIC achieves**
  - **175× (30×)** on-prem (remote) **better latency** vs. CMS CPU!
  - **Competitive throughput** vs. GPU with single-image batch as a service!



# SUMMARY



- Particle physics experiments face **extreme computing challenges**

# SUMMARY



- Particle physics experiments face **extreme computing challenges**
- Growing interest in **big deep learning models** for reconstruction and analysis

# SUMMARY



- Particle physics experiments face **extreme computing challenges**
- Growing interest in **big deep learning models** for reconstruction and analysis
- **FPGAs** can **accelerate neural network inference**
  - Achieve order of magnitude improvement in latency over CPU
  - Comparable throughput to GPU without batching

# SUMMARY



- Particle physics experiments face **extreme computing challenges**
- Growing interest in **big deep learning models** for reconstruction and analysis
- **FPGAs** can **accelerate neural network inference**
  - Achieve order of magnitude improvement in latency over CPU
  - Comparable throughput to GPU without batching
- **SONIC** infrastructure (gRPC + TensorFlow) developed and tested

# SUMMARY



- Particle physics experiments face **extreme computing challenges**
- Growing interest in **big deep learning models** for reconstruction and analysis
- **FPGAs** can **accelerate neural network inference**
  - Achieve order of magnitude improvement in latency over CPU
  - Comparable throughput to GPU without batching
- **SONIC** infrastructure (gRPC + TensorFlow) developed and tested
- Thanks to Microsoft Azure Machine Learning, Bing, and Project Brainwave teams!
  - Doug Burger, Eric Chung, Jeremy Fowers, Daniel Lo, Kalin Ovtcharov, and Andrew Putnam

# OUTLOOK



- Design more **machine learning algorithms for physics**
  - Easier to accelerate inference with commercial coprocessors

# OUTLOOK



- Design more **machine learning algorithms for physics**
  - Easier to accelerate inference with commercial coprocessors
- Develop tools for **generic model translation**
  - e.g. graph NNs used for HEP.TrkX and other projects

# OUTLOOK



- Design more **machine learning algorithms for physics**
  - Easier to accelerate inference with commercial coprocessors
- Develop tools for **generic model translation**
  - e.g. graph NNs used for HEP.TrkX and other projects
- Explore broad offerings of **potential hardware**
  - Google TPUs, Xilinx ML suite on AWS, Intel OpenVINO, …

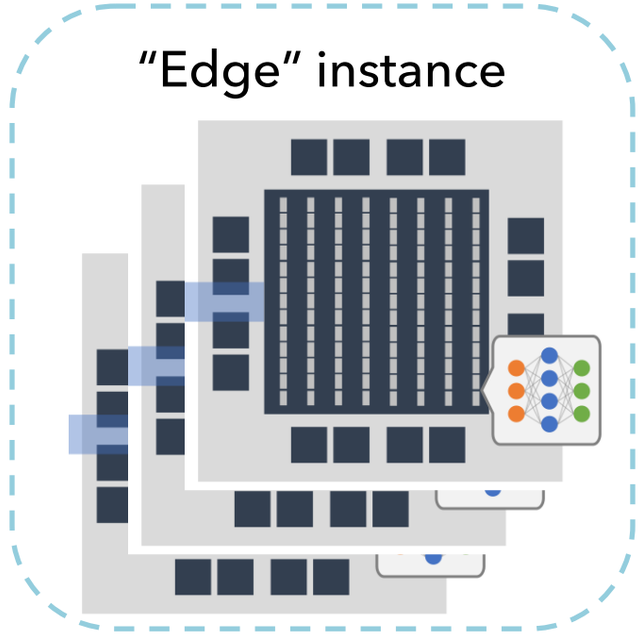
# OUTLOOK



- Design more **machine learning algorithms for physics**
  - Easier to accelerate inference with commercial coprocessors
- Develop tools for **generic model translation**
  - e.g. graph NNs used for HEP.TrkX and other projects
- Explore broad offerings of **potential hardware**
  - Google TPUs, Xilinx ML suite on AWS, Intel OpenVINO, …
- **Build infrastructure** and study scalability/cost
  - Adapt SONIC to handle other protocols, other network architectures and ML libraries, other experiments

# VISION OF THE FUTURE




- Single FPGA can support many CPUs → cost-effective
  - SONIC throughput results indicate 1 FPGA for 100–1000 CPUs running realistic processes
- Install small “edge” instances at T1s and T2s
  - Dedicated instance for CMS HLT farm at CERN

JAVIER DUARTE

APRIL 2, 2019

CONNECTING THE DOTS

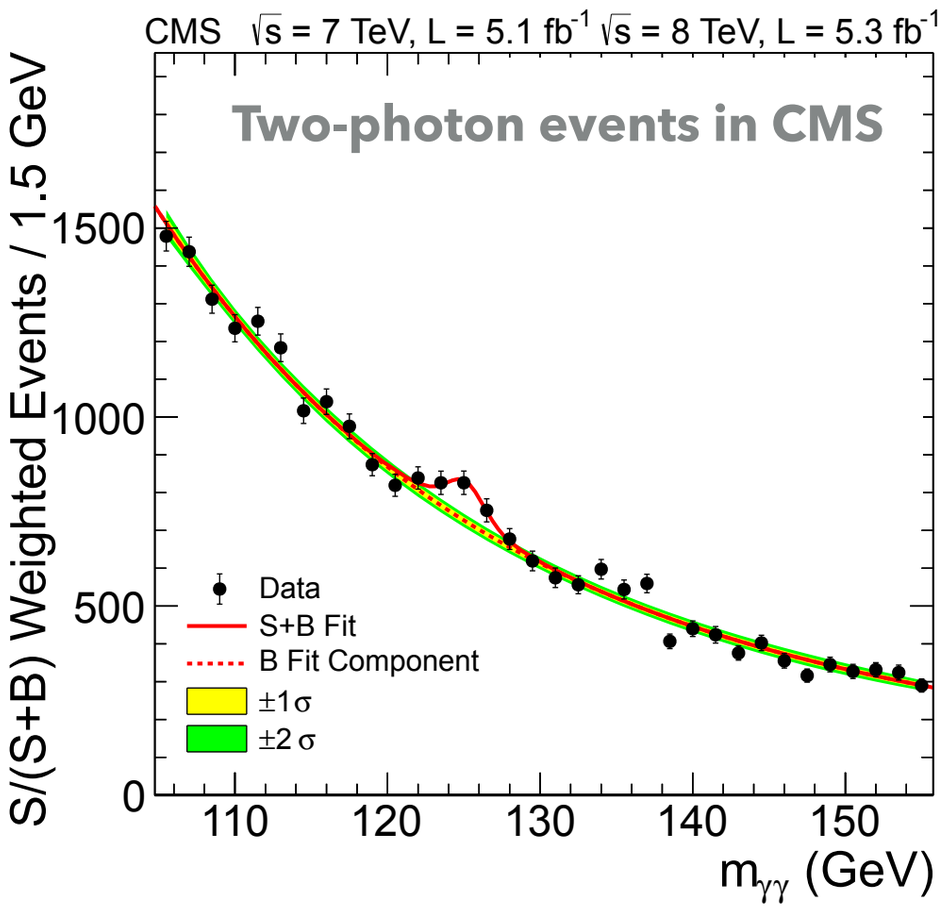
VALENCIA, SPAIN

# BACKUP

# HIGH ENERGY PHYSICS + MACHINE LEARNING



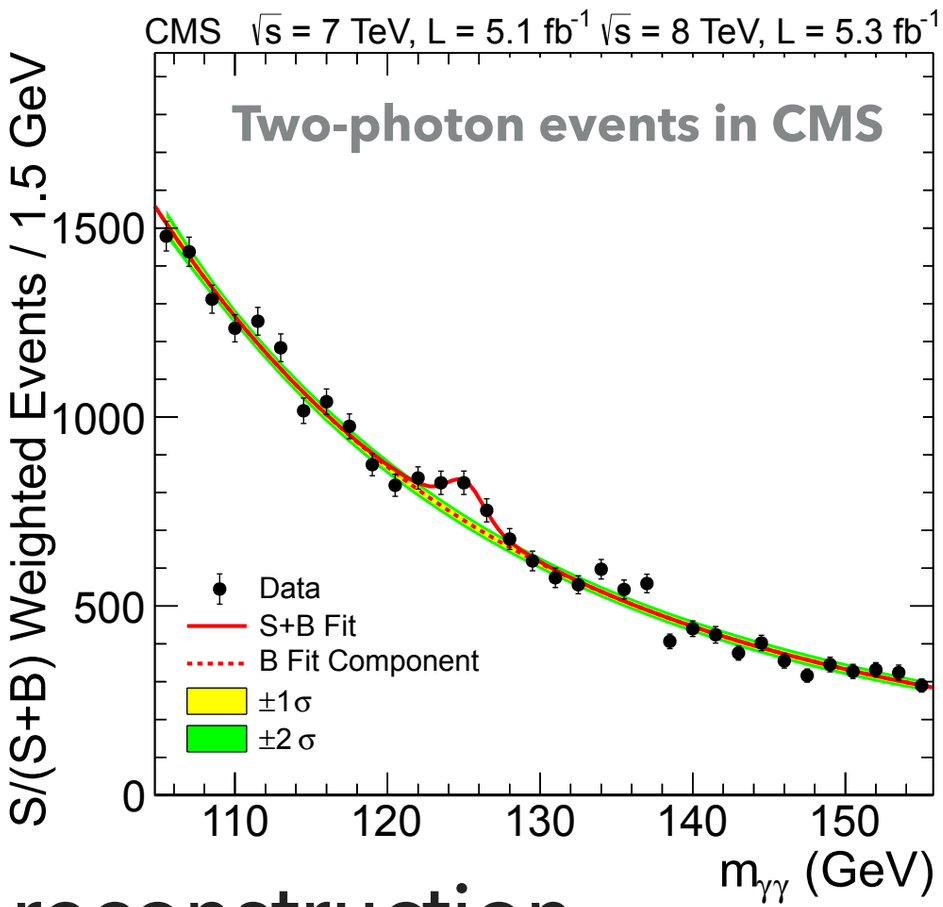
- **Machine learning** was vital to make big discoveries like the Higgs boson on July 4, 2012



# HIGH ENERGY PHYSICS + MACHINE LEARNING



- **Machine learning** was vital to make big discoveries like the Higgs boson on July 4, 2012

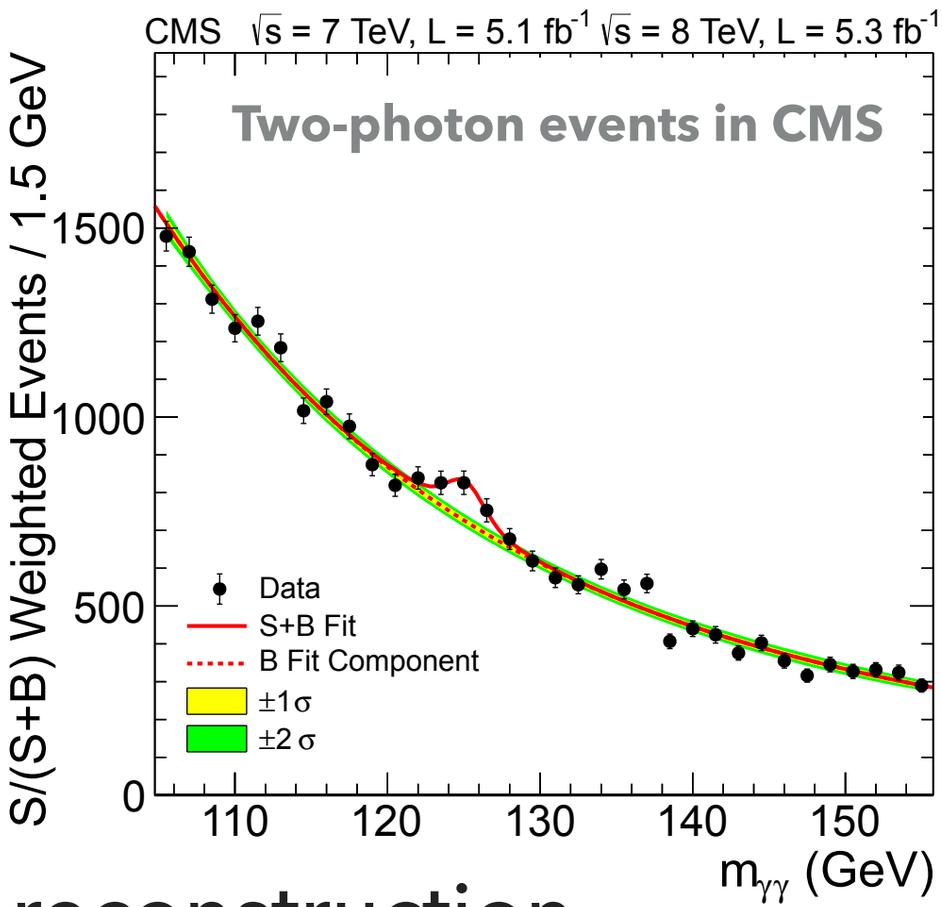



- Today, ML is **enabling** better trigger, reconstruction, and analysis at the LHC

# HIGH ENERGY PHYSICS + MACHINE LEARNING



- **Machine learning** was vital to make big discoveries like the Higgs boson on July 4, 2012



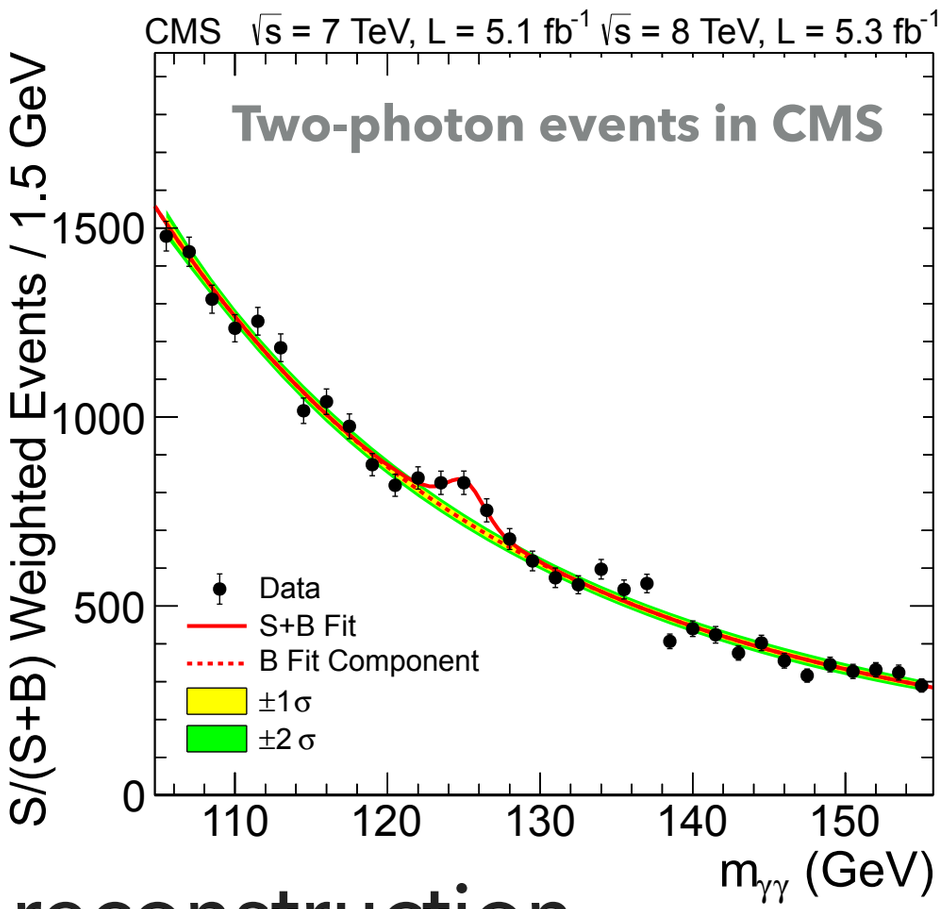

- Today, ML is **enabling** better trigger, reconstruction, and analysis at the LHC

- At the same time, we must **plan** how we will overcome challenges in the **next generation of colliders**

# HIGH ENERGY PHYSICS + MACHINE LEARNING



- **Machine learning** was vital to make big discoveries like the Higgs boson on July 4, 2012




- Today, ML is **enabling** better trigger, reconstruction, and analysis at the LHC
- At the same time, we must **plan** how we will overcome challenges in the **next generation of colliders**
- ML may be a way out

# NEXT-GEN (HIGH-LUMINOSITY) LHC: FAST FACTS

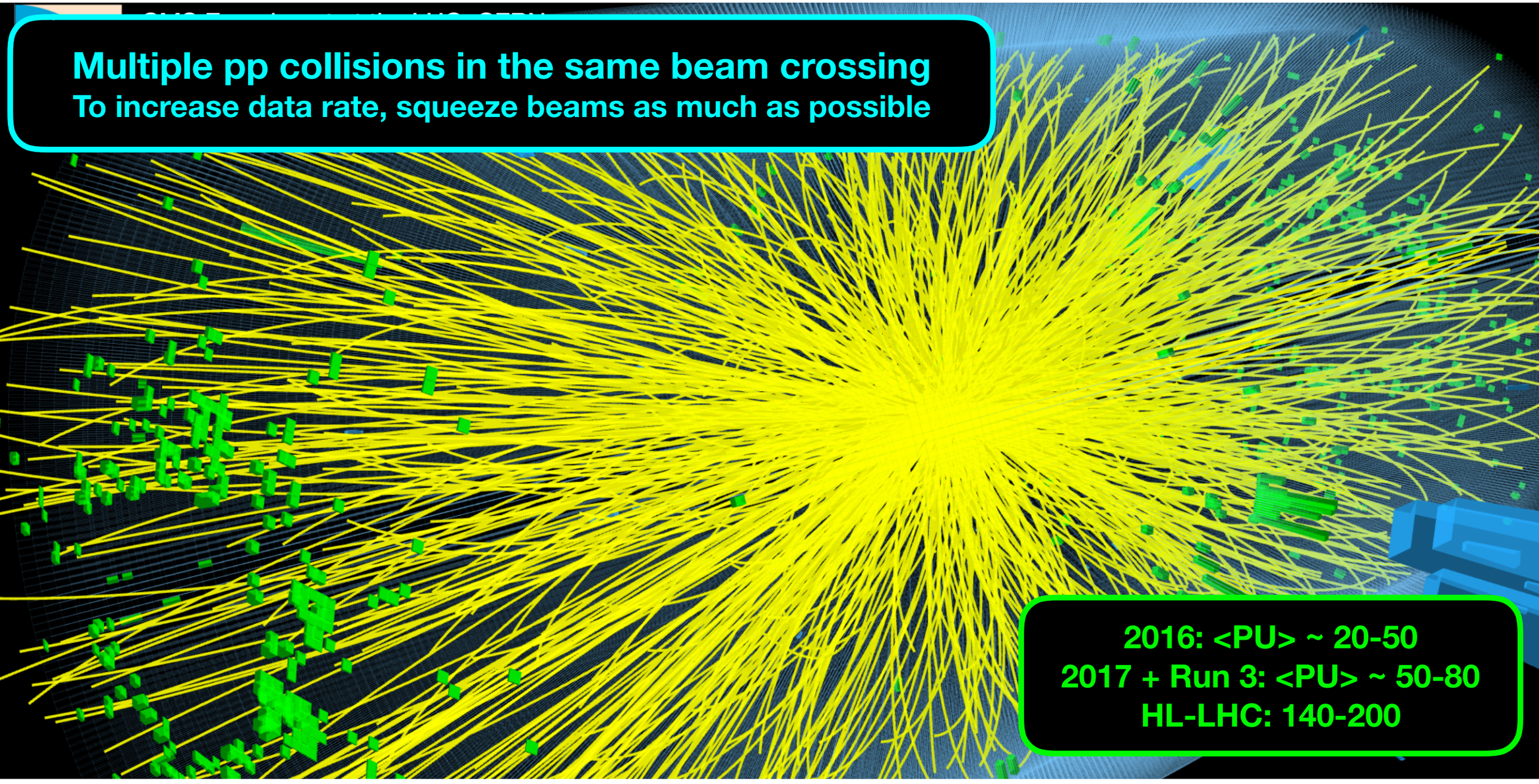



- Planned to begin in **2026** and collect 10× more data
- **More dense** collisions by a factor of 5
- Detectors will be upgraded to deal with increased collision complexity & provide **more granular** information
- All factors contribute to our “big data” challenge

# CHALLENGE: PILEUP



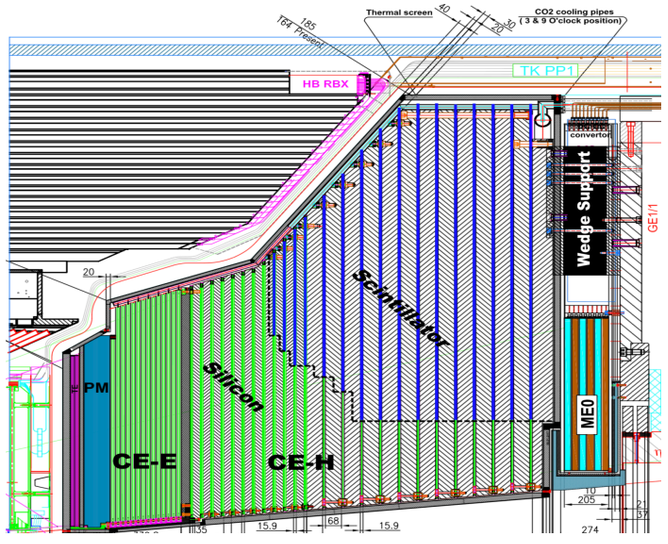
**Multiple pp collisions in the same beam crossing**
**To increase data rate, squeeze beams as much as possible**




- At high luminosity, many collisions happen simultaneously (pileup)!
- Pileup makes our data more complex and noisy

# CHALLENGE: NEW DETECTORS

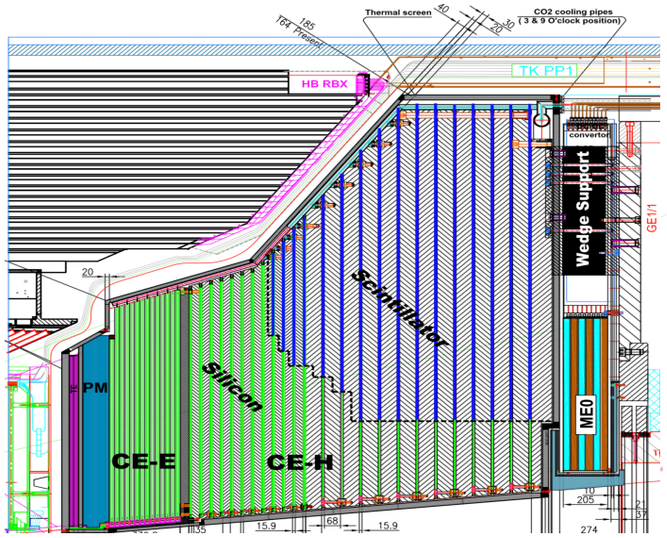


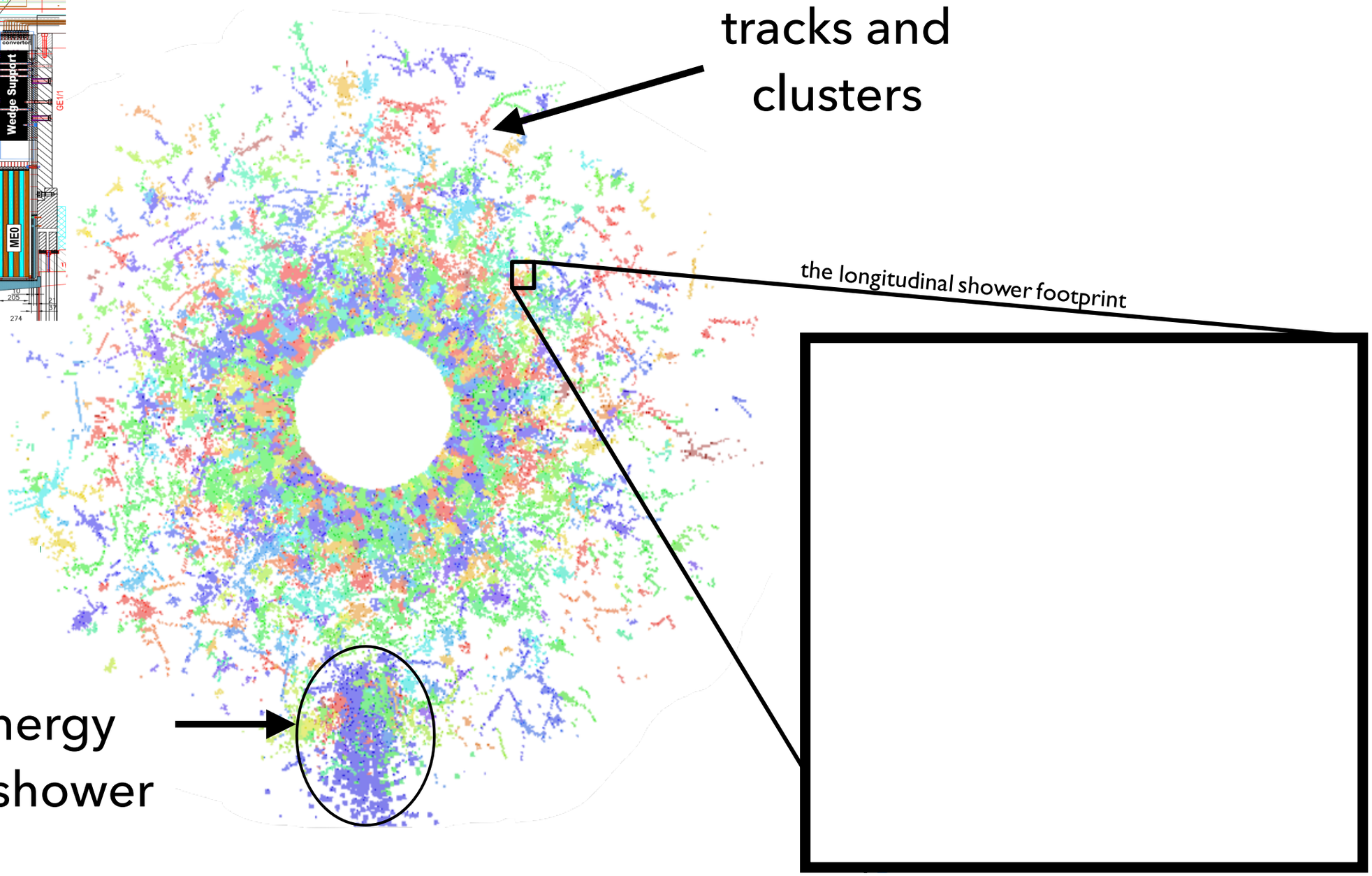



- High Granularity Calorimeter will provide 3D information of a particle shower as it evolves

# CHALLENGE: NEW DETECTORS



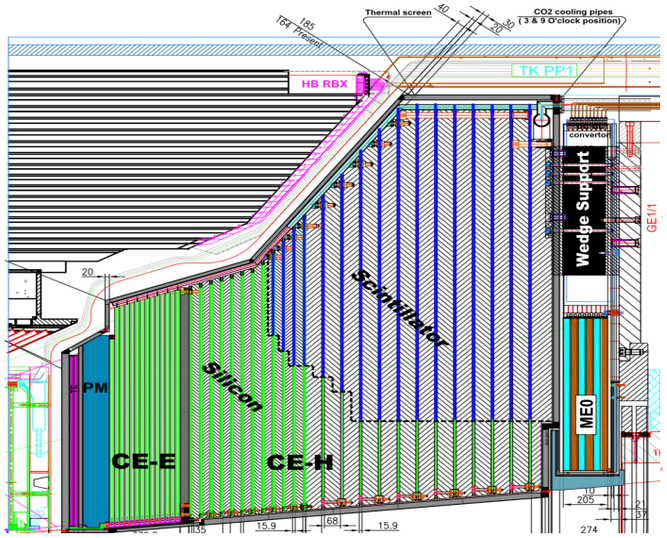


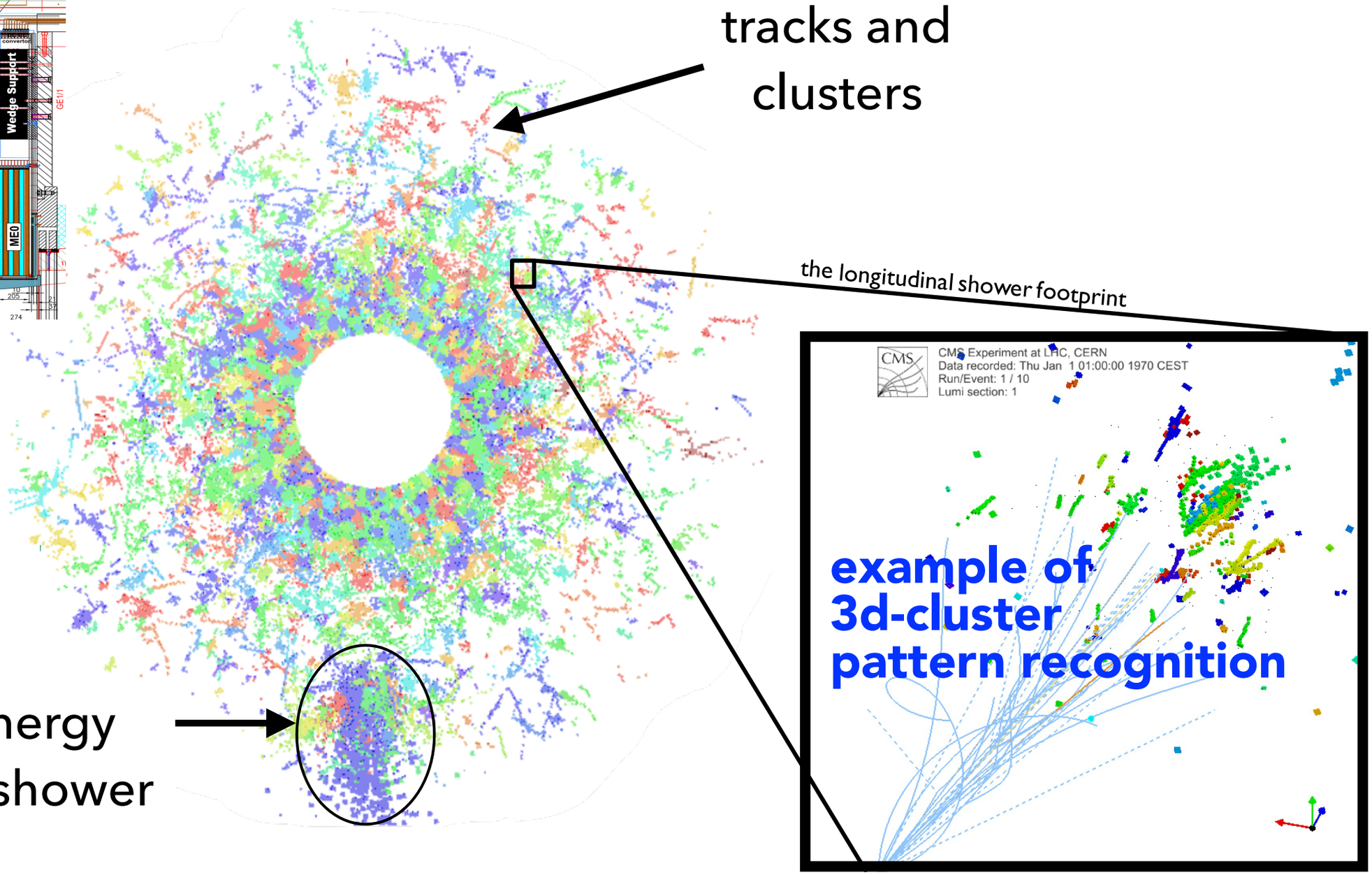




- High Granularity Calorimeter will provide 3D information of a particle shower as it evolves

# CHALLENGE: NEW DETECTORS





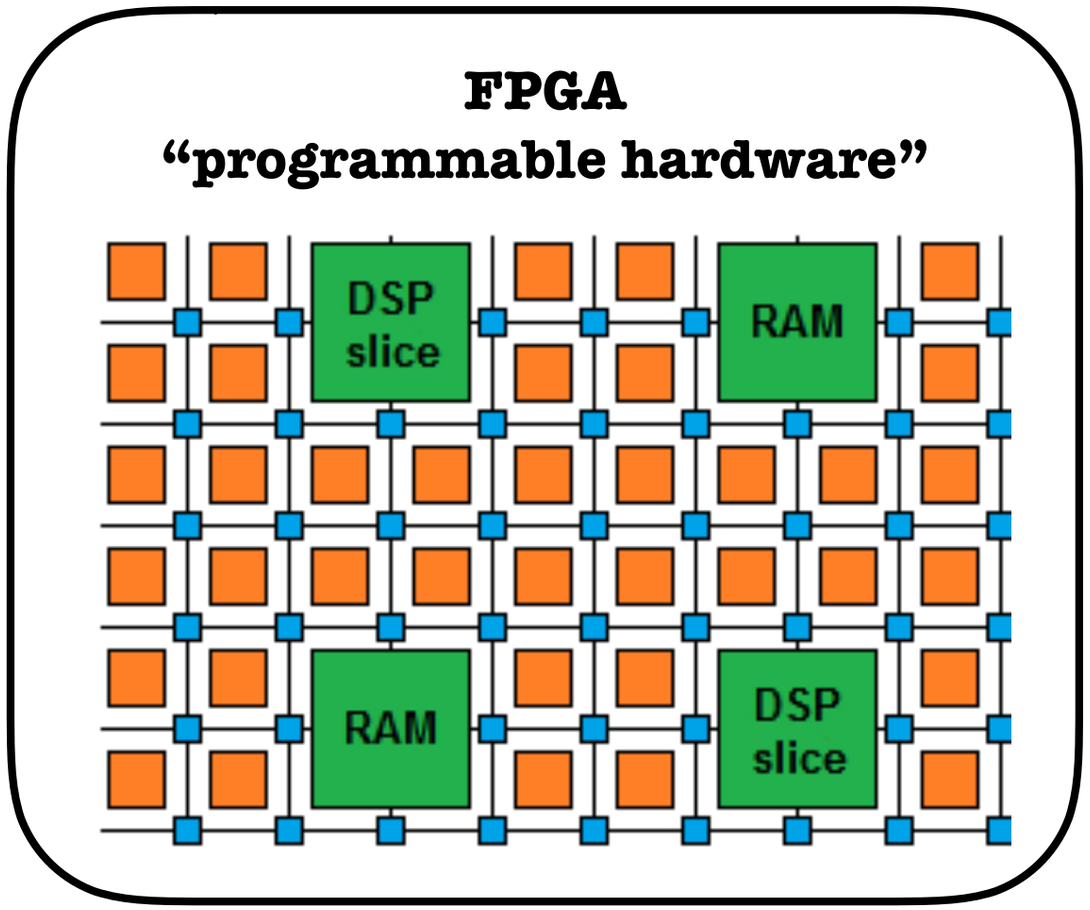




- High Granularity Calorimeter will provide 3D information of a particle shower as it evolves

# FPGAS: REPROGRAMMABLE HARDWARE



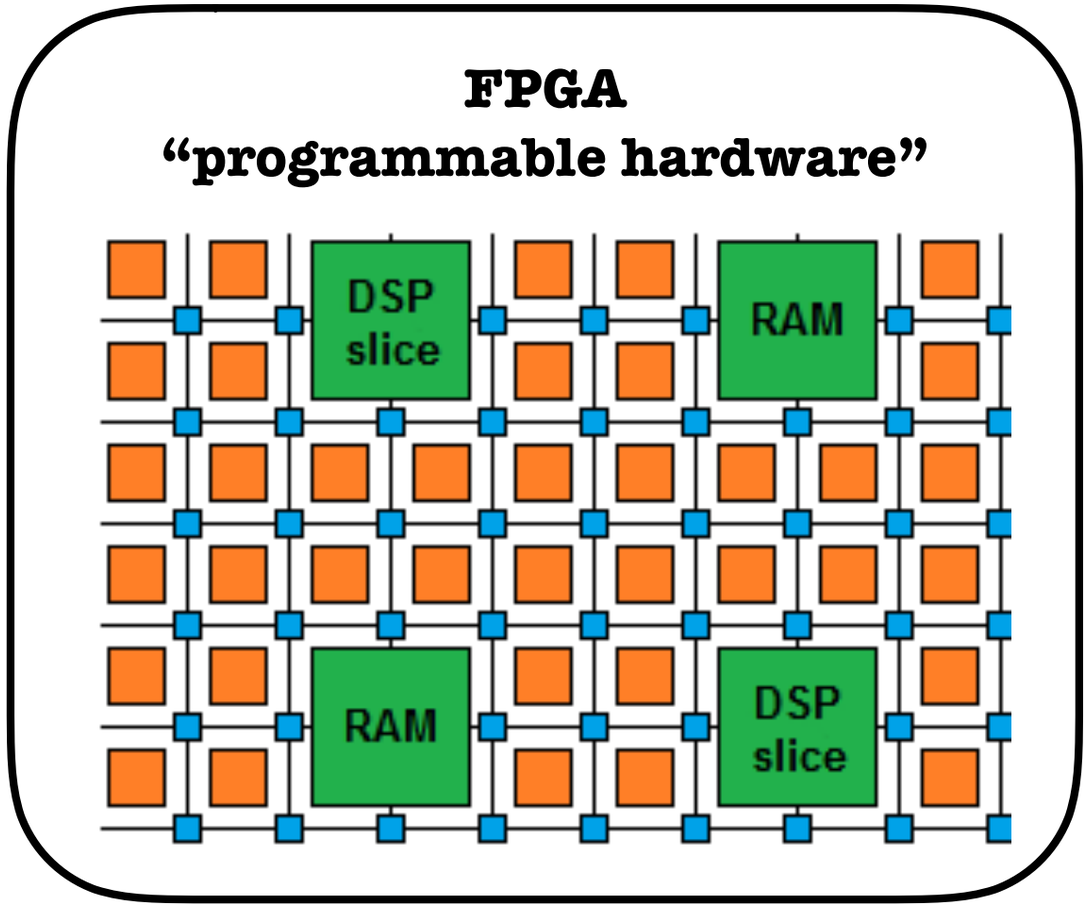
- **Pros:**



# FPGAS: REPROGRAMMABLE HARDWARE



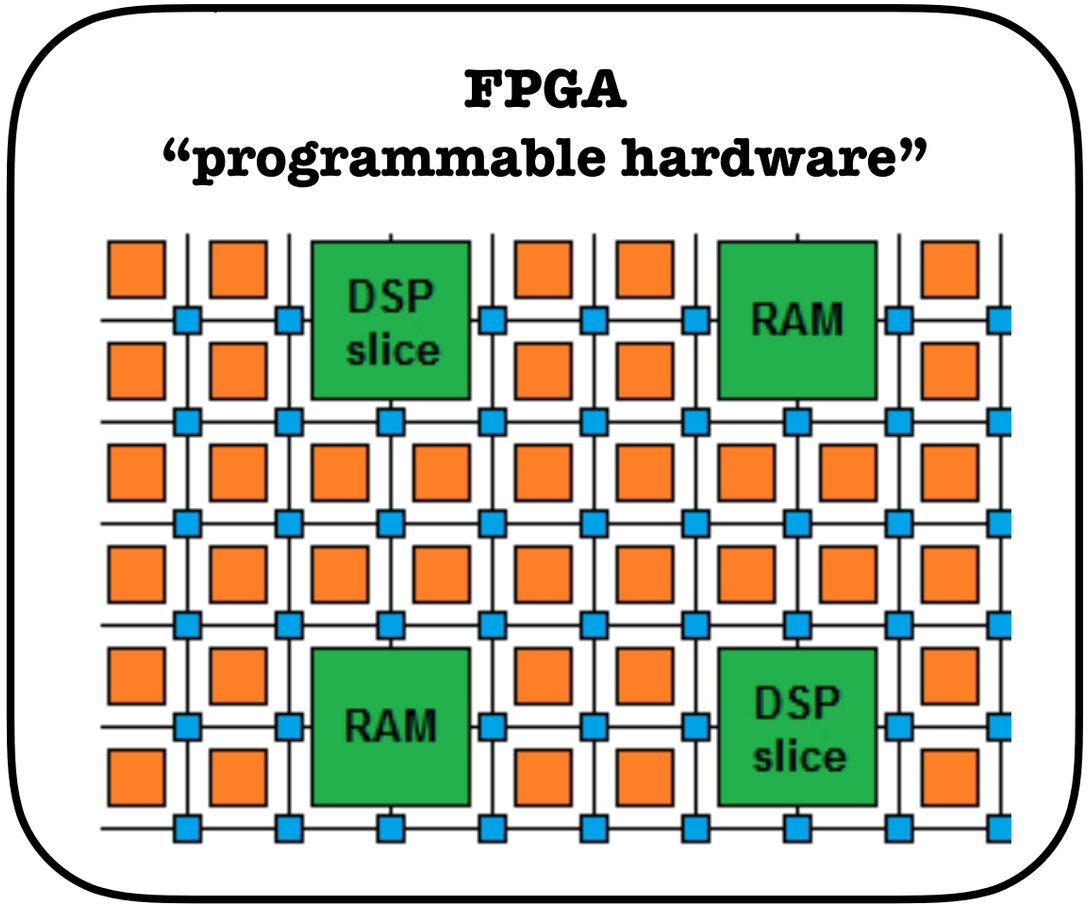
- **Pros:**
  - **Reprogrammable** interconnects between **embedded components** that perform multiplication (**DSPs**), apply logical functions (**LUTs**), or store memory (**BRAM**)



# FPGAS: REPROGRAMMABLE HARDWARE



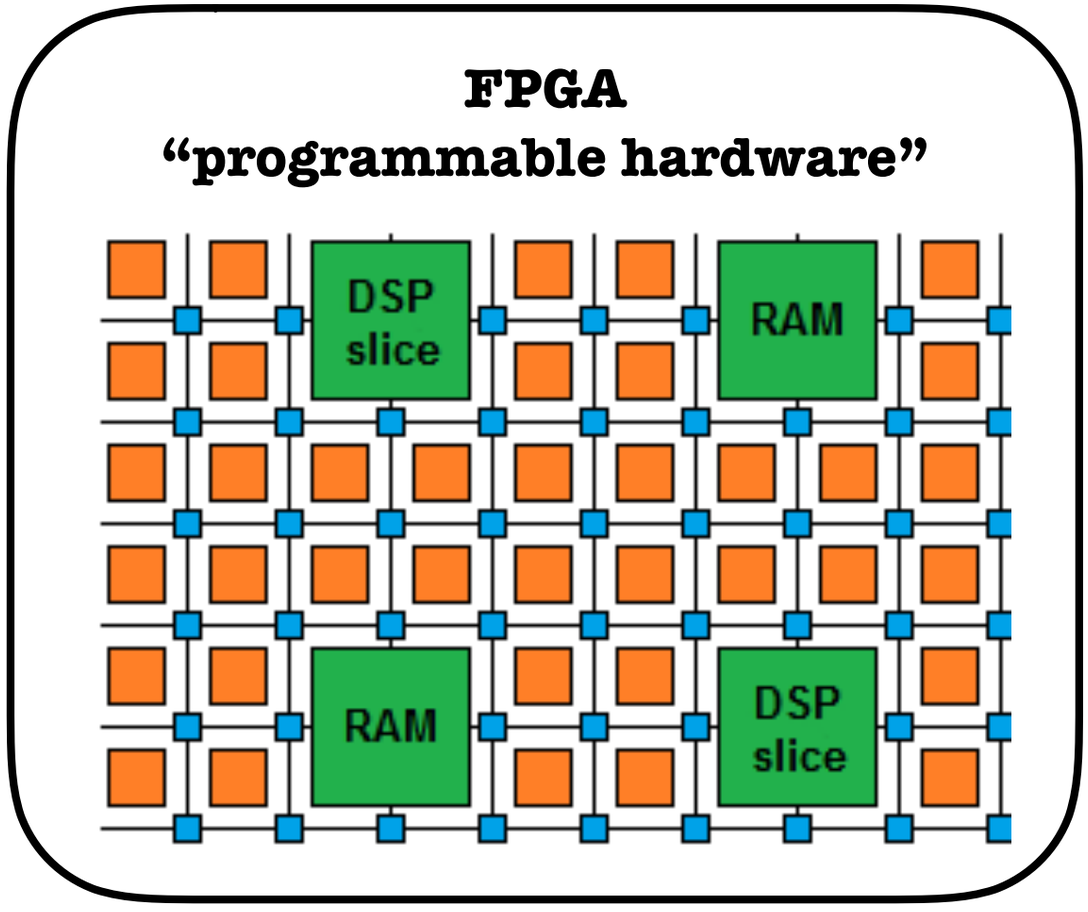
- **Pros:**
  - **Reprogrammable** interconnects between **embedded components** that perform multiplication (**DSPs**), apply logical functions (**LUTs**), or store memory (**BRAM**)
  - **High throughput:** O(100) optical transceivers running at O(15) Gbs



# FPGAS: REPROGRAMMABLE HARDWARE



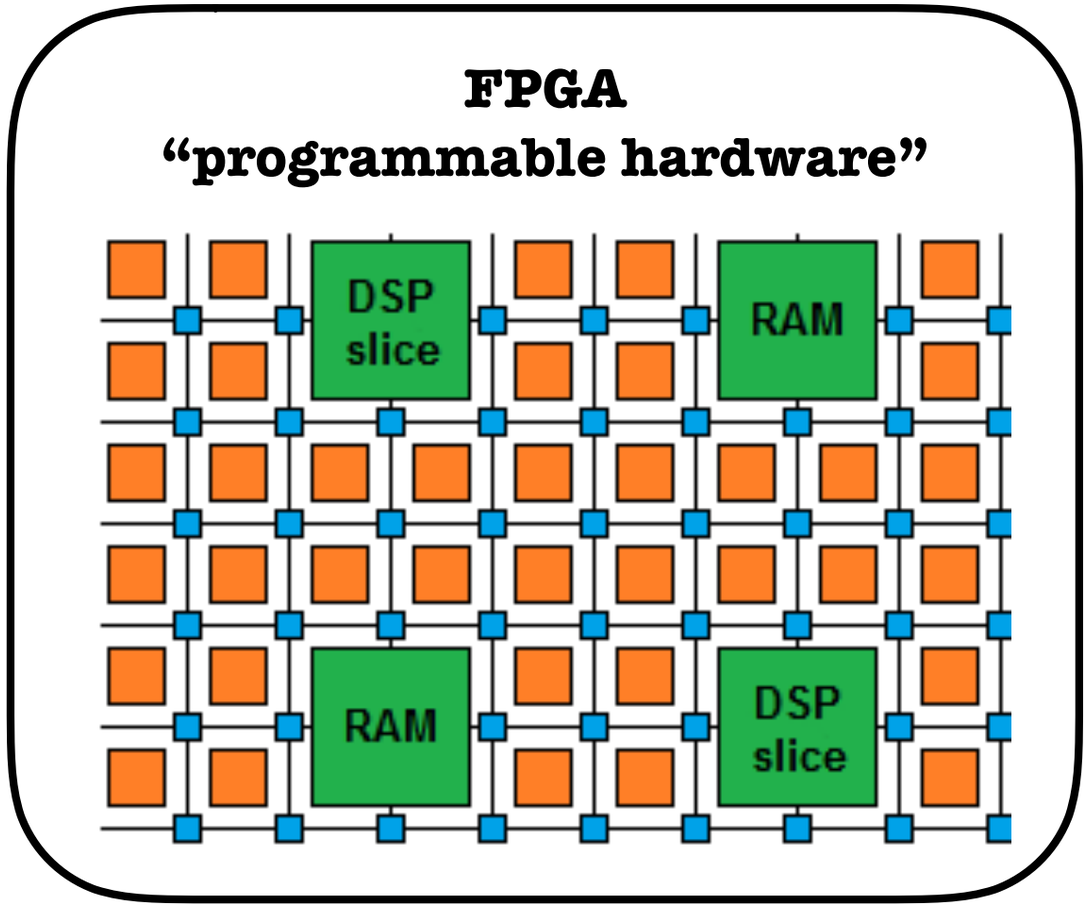
- **Pros:**
  - **Reprogrammable** interconnects between **embedded components** that perform multiplication (**DSPs**), apply logical functions (**LUTs**), or store memory (**BRAM**)
  - **High throughput:** O(100) optical transceivers running at O(15) Gbs
  - **Massively parallel**



# FPGAS: REPROGRAMMABLE HARDWARE



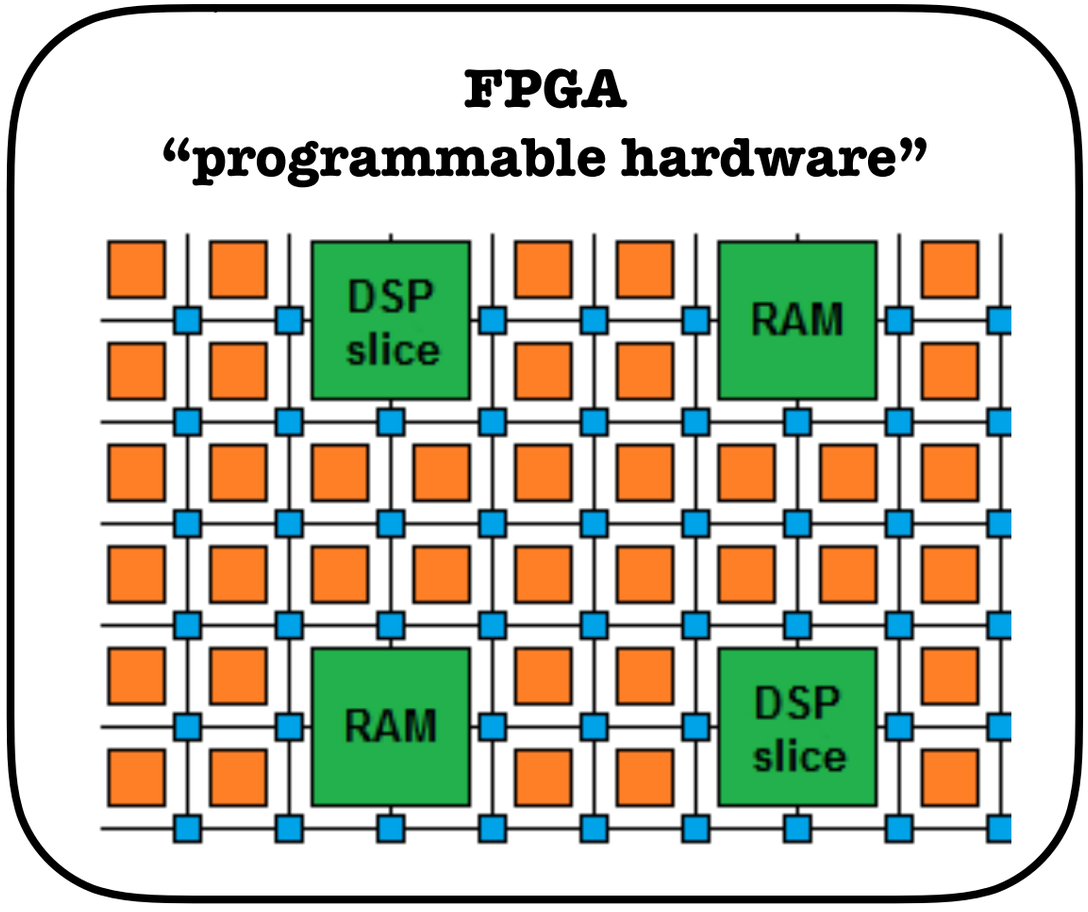
- **Pros:**
  - **Reprogrammable** interconnects between **embedded components** that perform multiplication (**DSPs**), apply logical functions (**LUTs**), or store memory (**BRAM**)
  - **High throughput:** O(100) optical transceivers running at O(15) Gbs
  - **Massively parallel**
  - **Low power**



# FPGAS: REPROGRAMMABLE HARDWARE



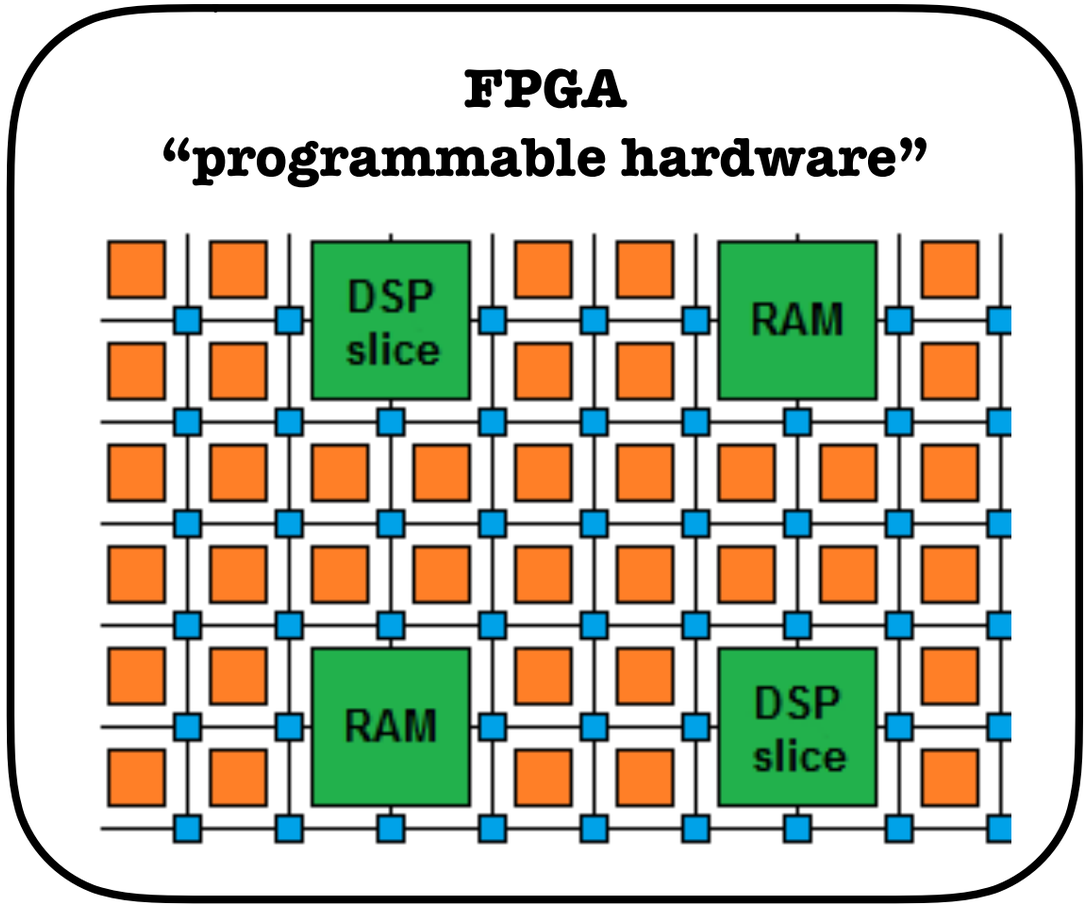
- **Pros:**
  - **Reprogrammable** interconnects between **embedded components** that perform multiplication (**DSPs**), apply logical functions (**LUTs**), or store memory (**BRAM**)
  - **High throughput:** O(100) optical transceivers running at O(15) Gbs
  - **Massively parallel**
  - **Low power**
- **Cons:**



# FPGAS: REPROGRAMMABLE HARDWARE



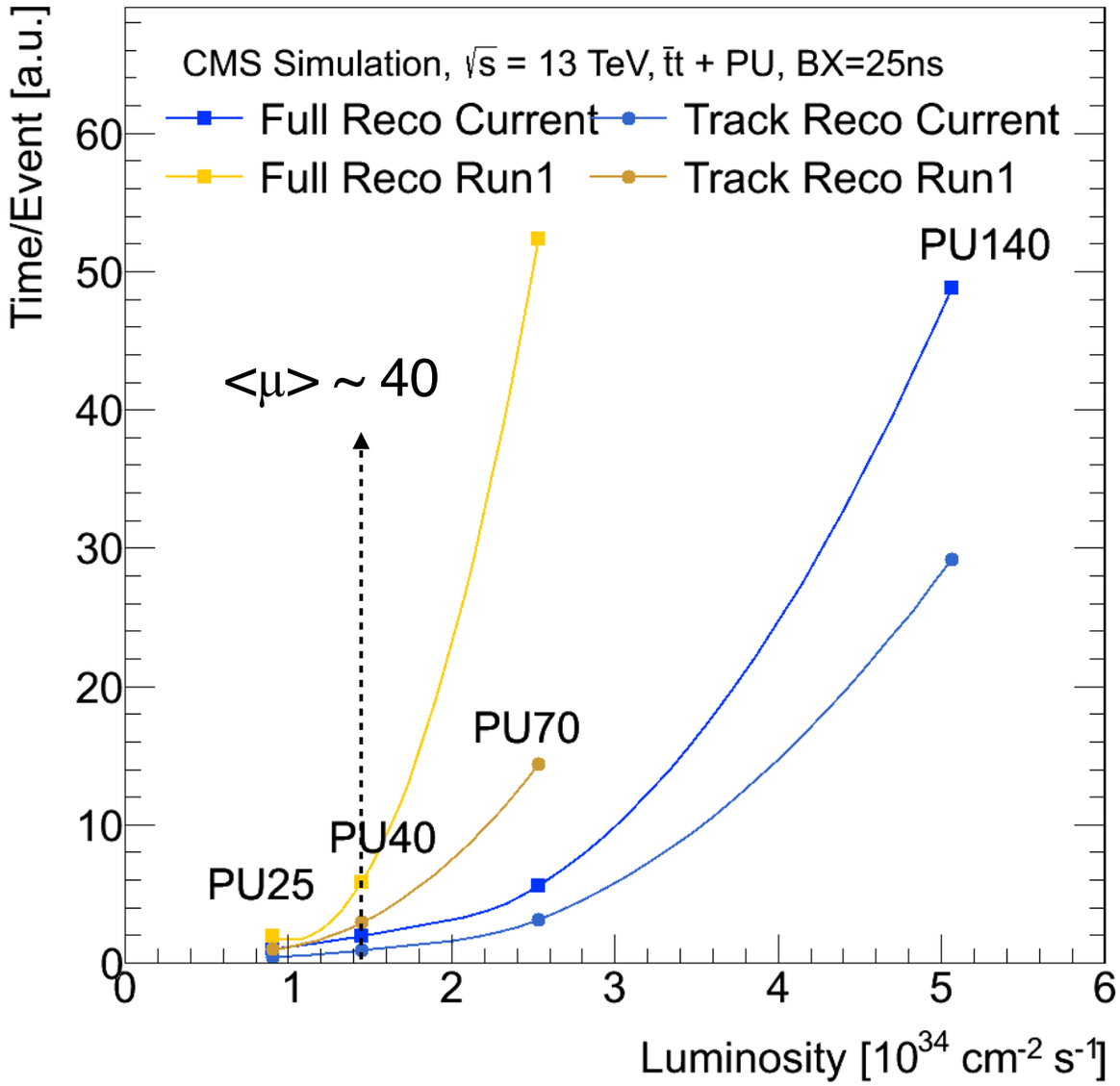
- **Pros:**
  - **Reprogrammable** interconnects between **embedded components** that perform multiplication (**DSPs**), apply logical functions (**LUTs**), or store memory (**BRAM**)
    - **High throughput:** O(100) optical transceivers running at O(15) Gbs
  - **Massively parallel**
  - **Low power**

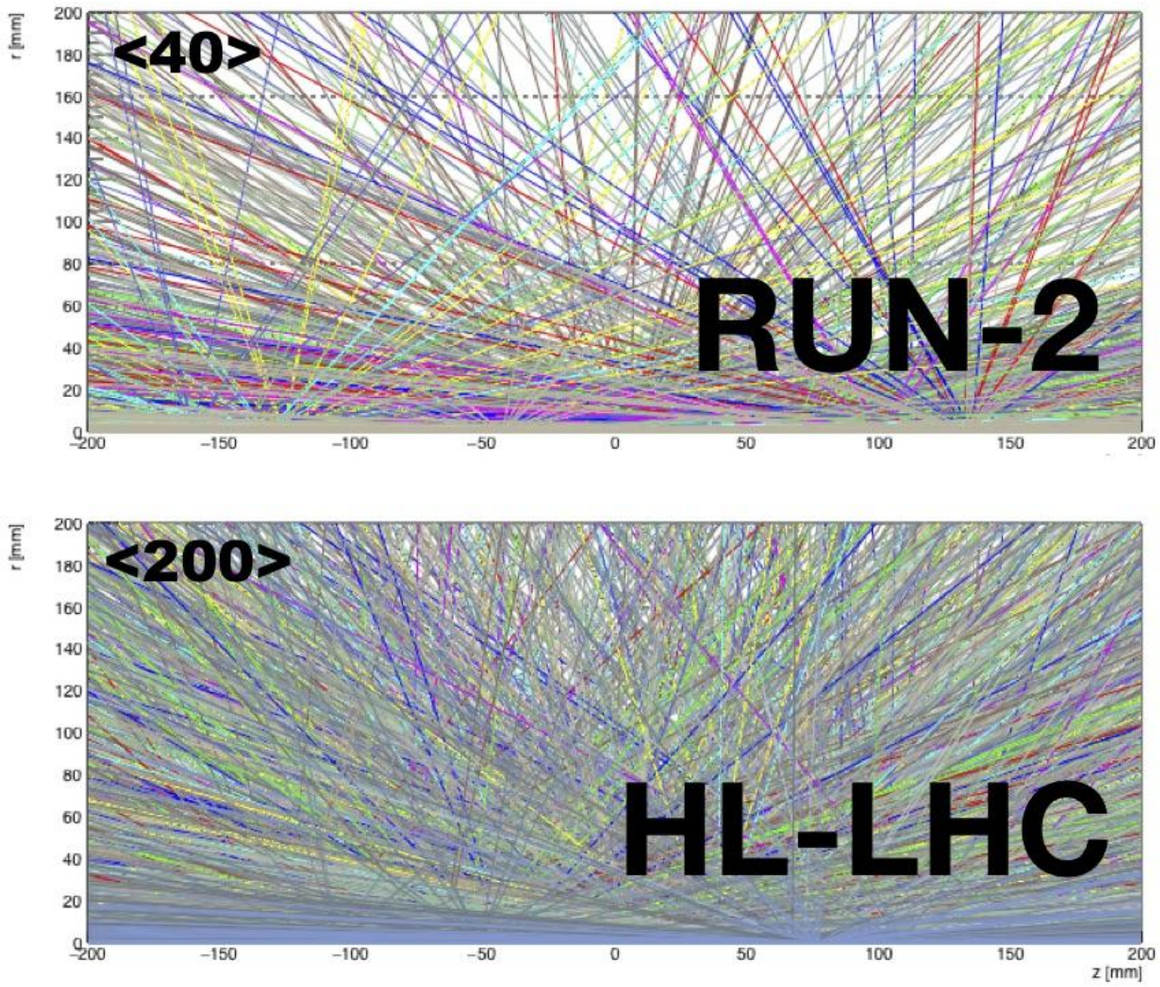



- **Cons:**
  - **Requires expert knowledge to program** (using VHDL/Verilog)

# TODAY'S RECONSTRUCTION IN 200 PILEUP





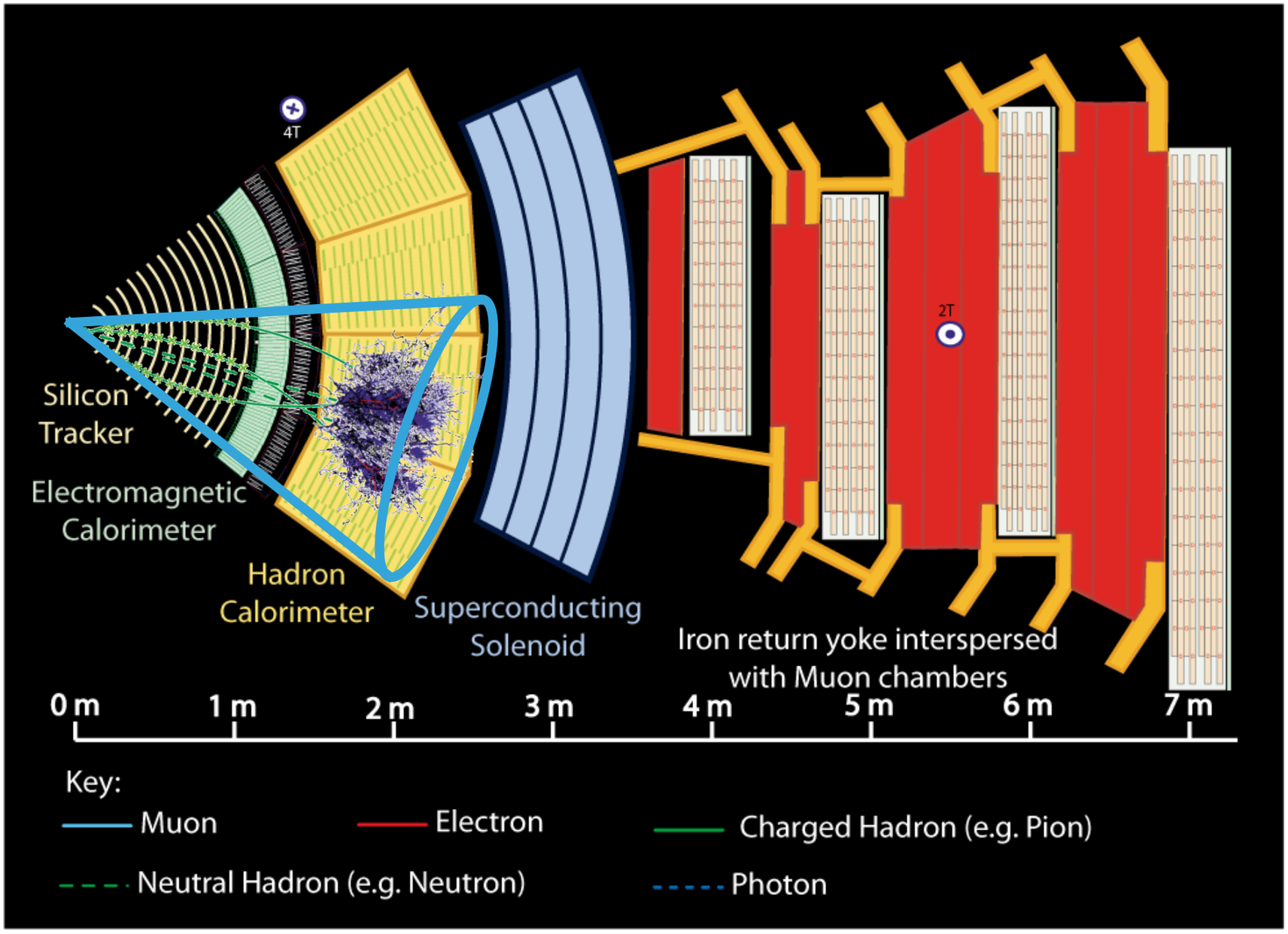




- Current tracking (most CPU intensive part) algorithms scale worse than quadratically O($N^2$), $N$=hits
- Can we **accelerate** them with **new hardware**?

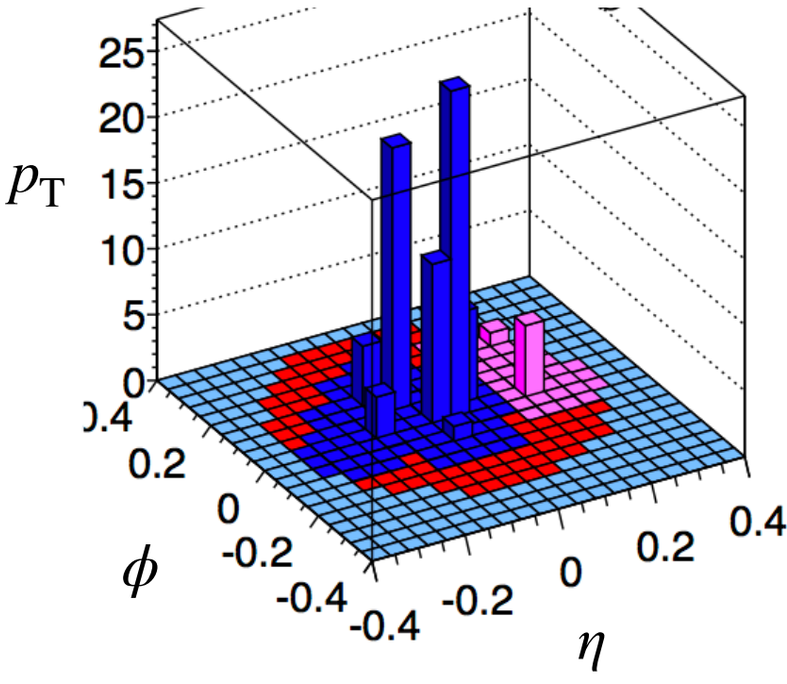
# JETS IN CMS (RECAP)



- Quarks and gluons from H→bb decay become *jets* of charged and neutral hadrons



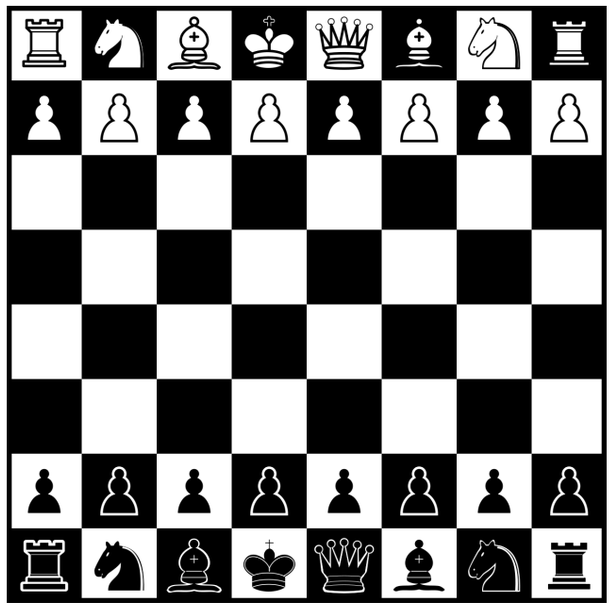

- Cluster the tracks and energy into a cone-shaped jet



# AN ANALOGY



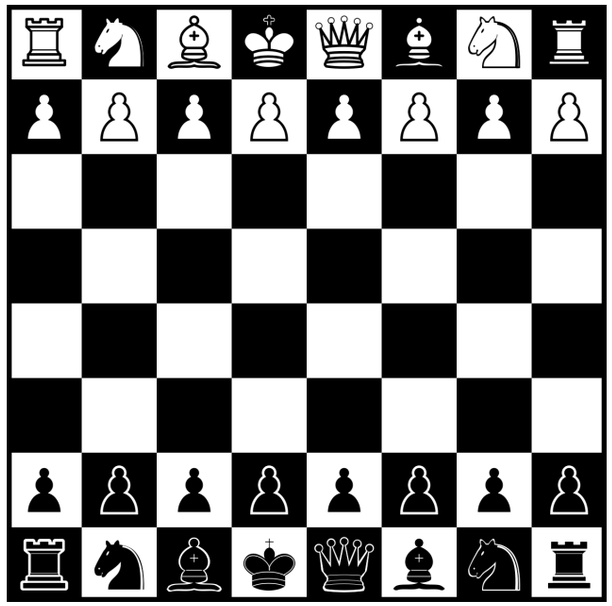
## Hard problem: Chess



# AN ANALOGY



Hard problem: Chess

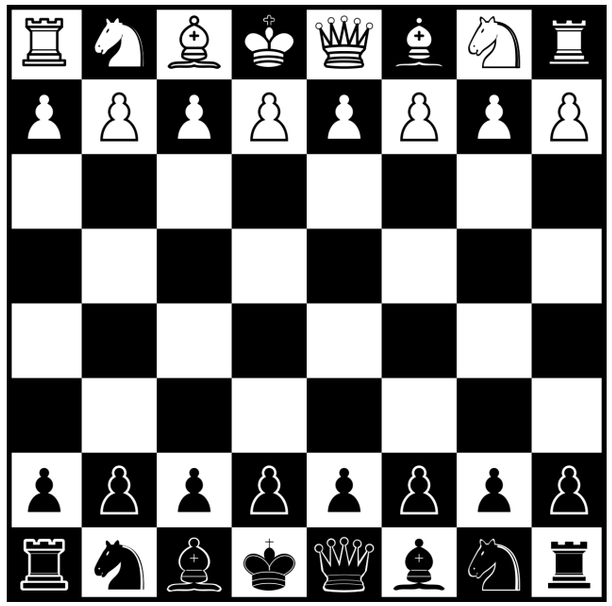


Solution: rule-based system
based on expert human knowledge

# AN ANALOGY

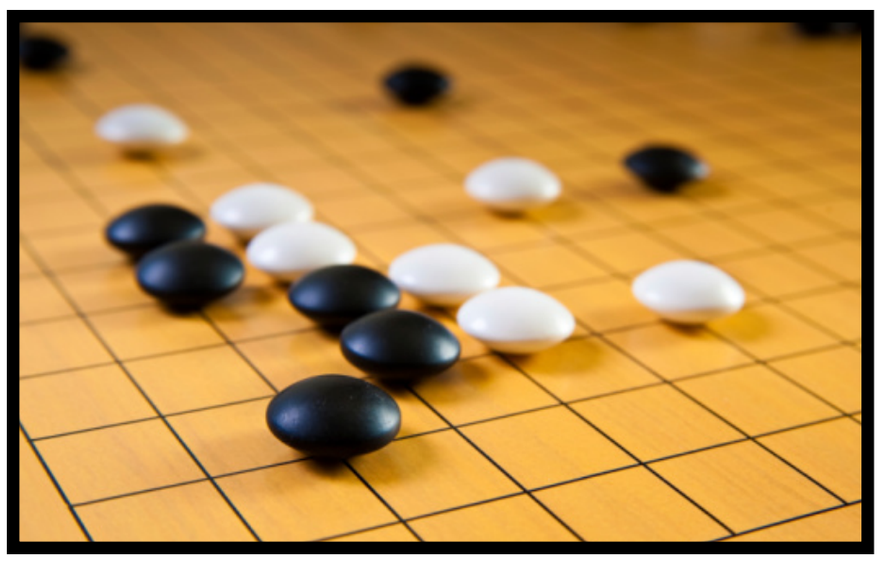


Hard problem: Chess



Solution: rule-based system
based on expert human knowledge

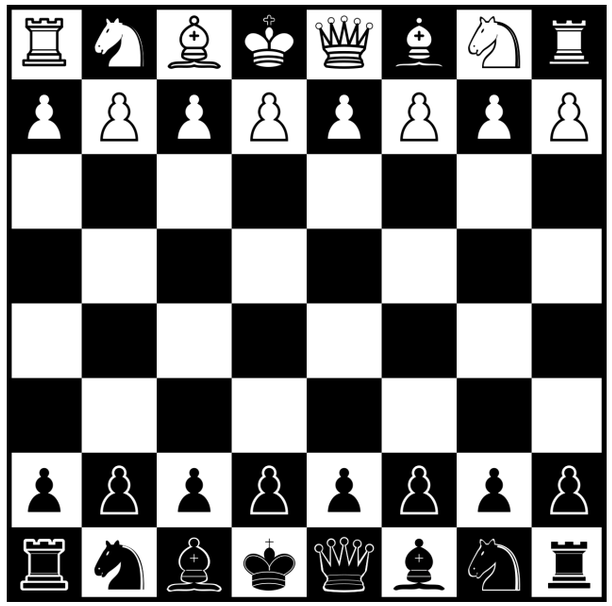

Harder problem: Go



# AN ANALOGY

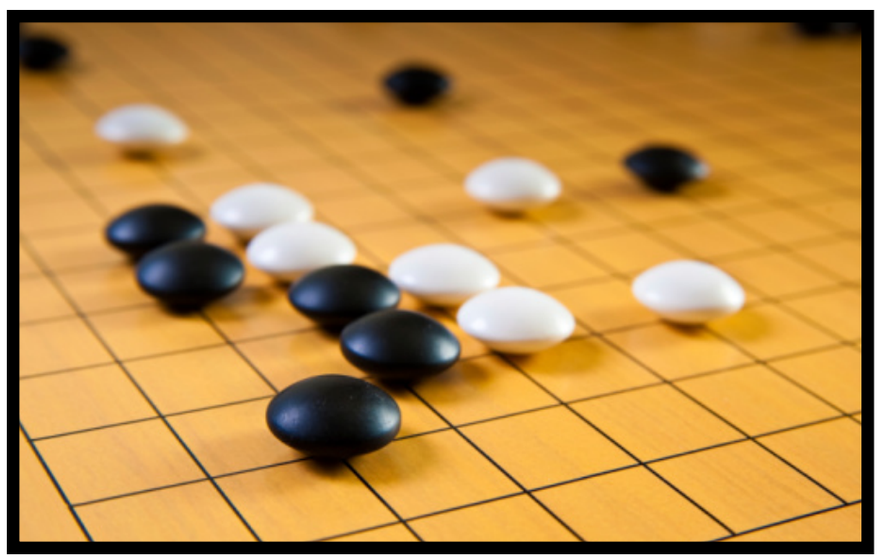


Hard problem: Chess

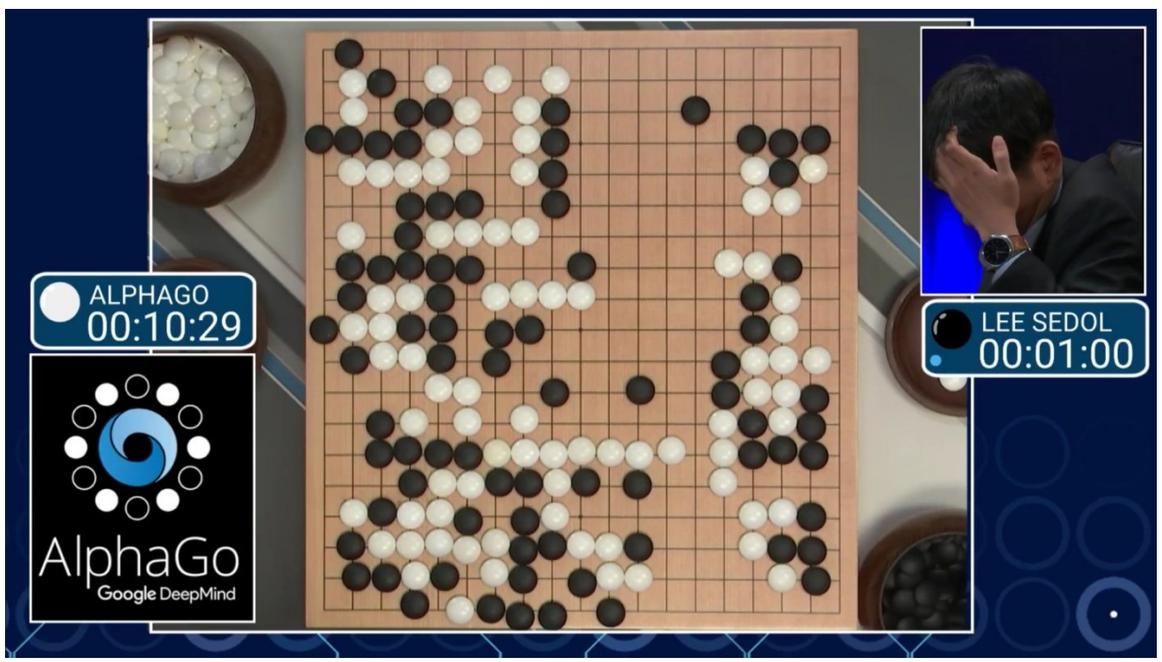


Solution: rule-based system
based on expert human knowledge


Harder problem: Go



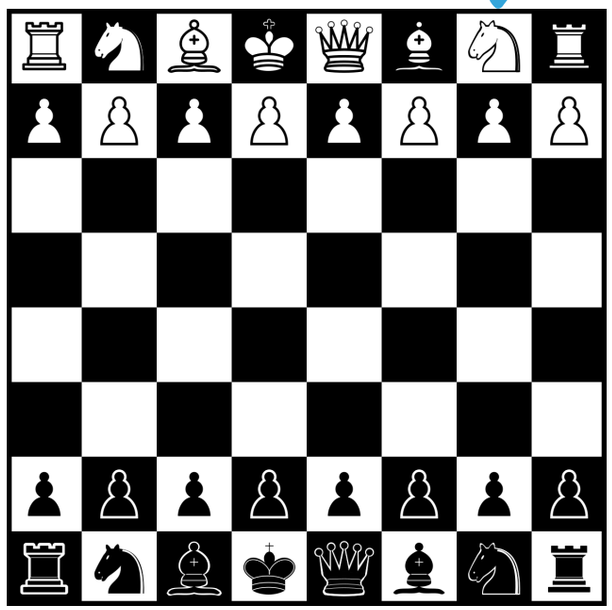
Solution: deep learning



# AN ANALOGY



Hard problem: Chess



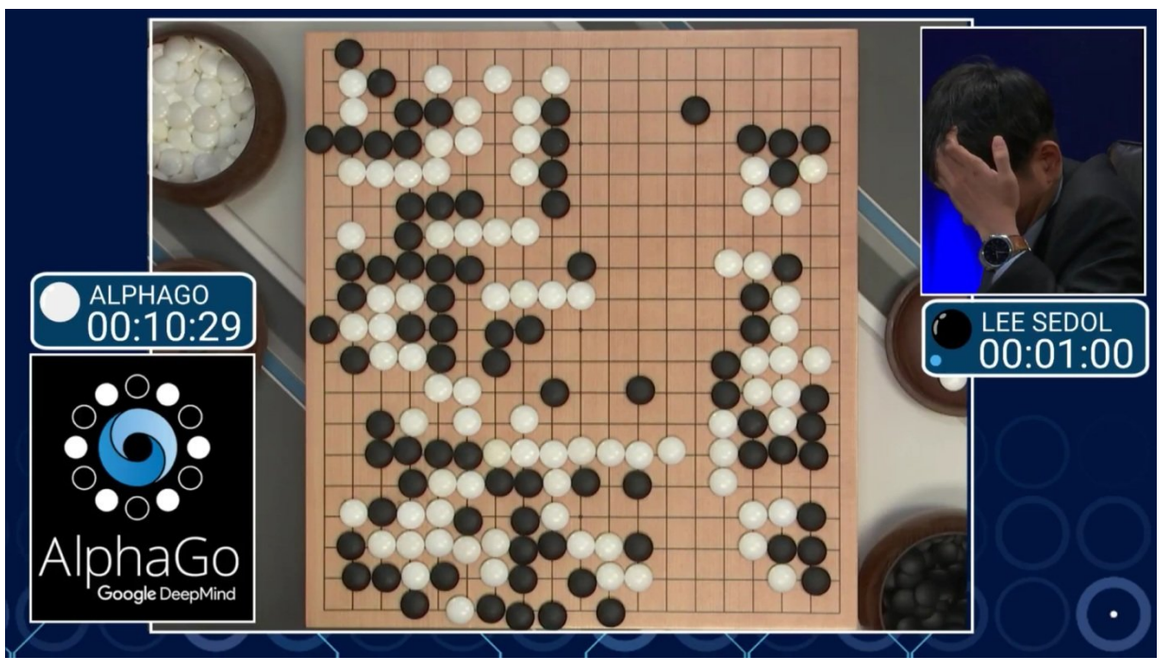
**Discoveries @ past colliders**

Solution: rule-based system based on expert human knowledge


Harder problem: Go

**Discoveries @ future colliders**

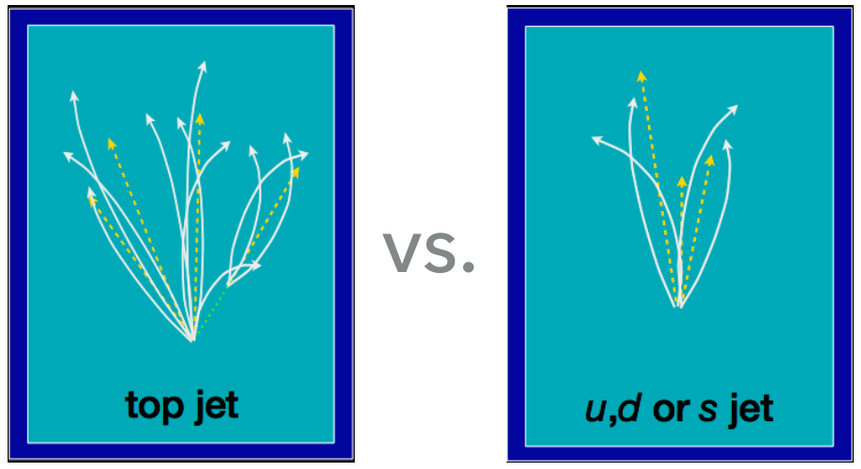
Solution: deep learning



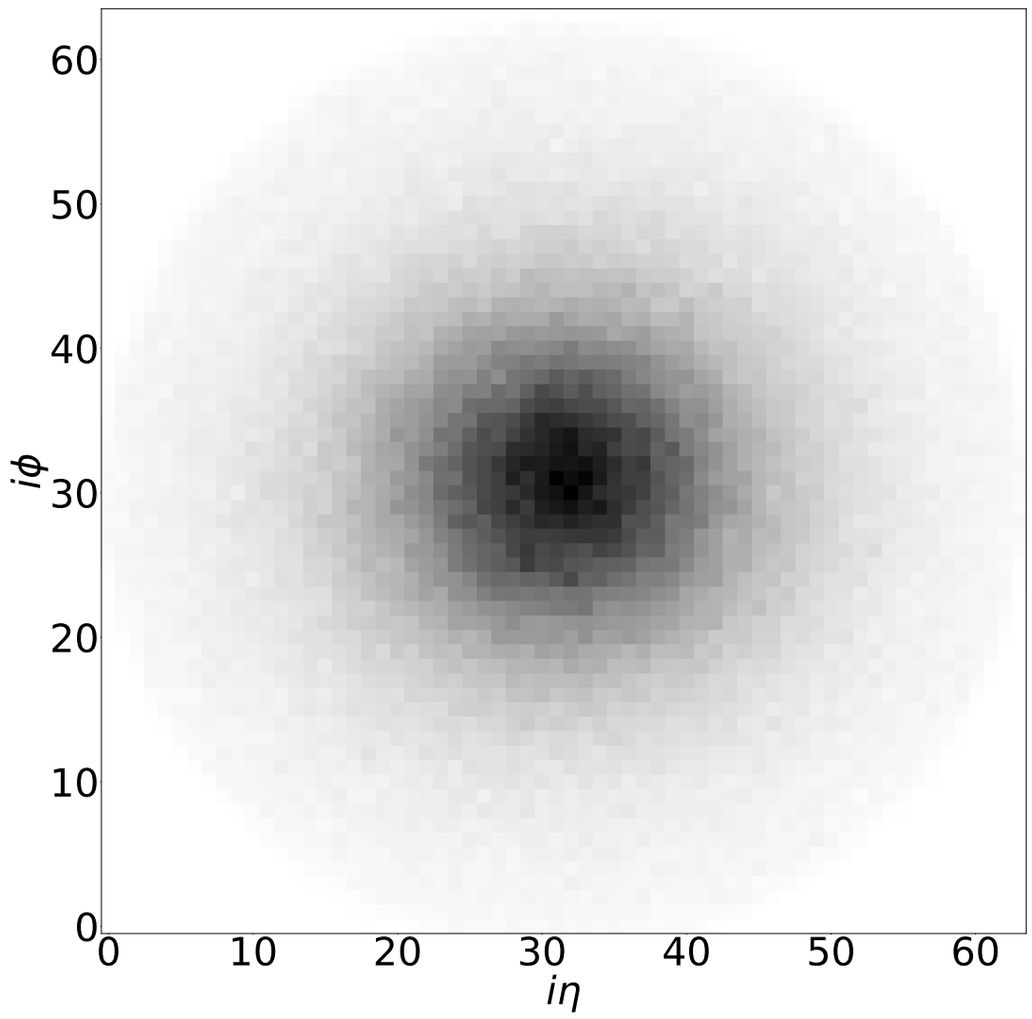
# JET VISION

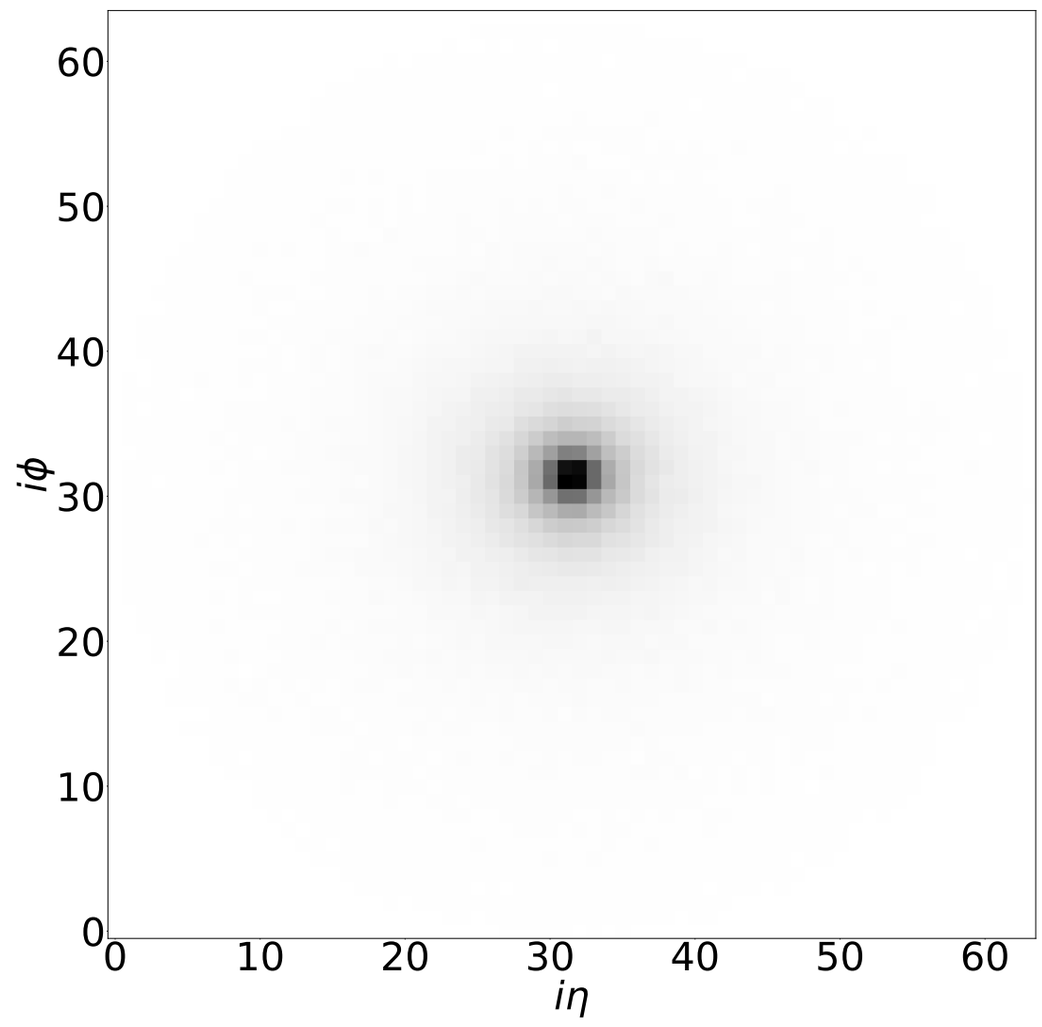


- Re-train ResNet-50 to identify the origin of jets
- Inputs are jet images = pixelated versions of calorimeter hits in 2D (η, Φ)




Note: averaged over 10k jets; 1 jet gives a ***sparse*** image

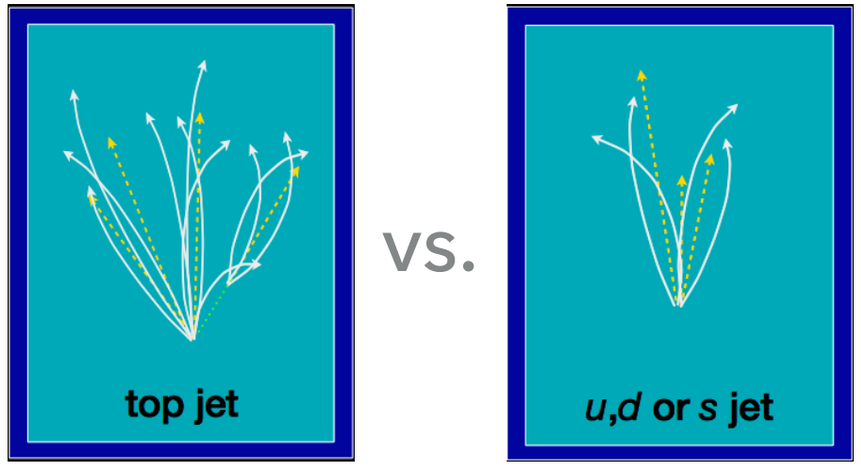


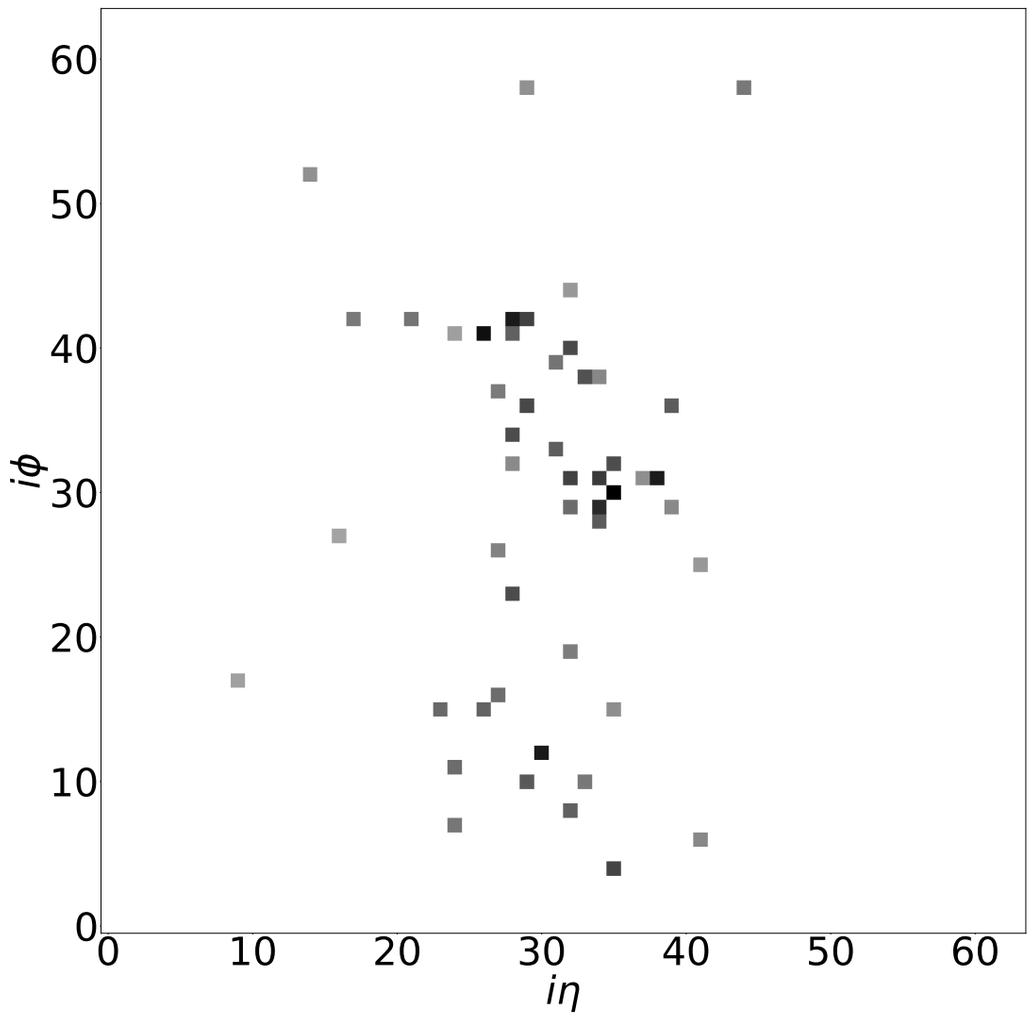
# JET VISION

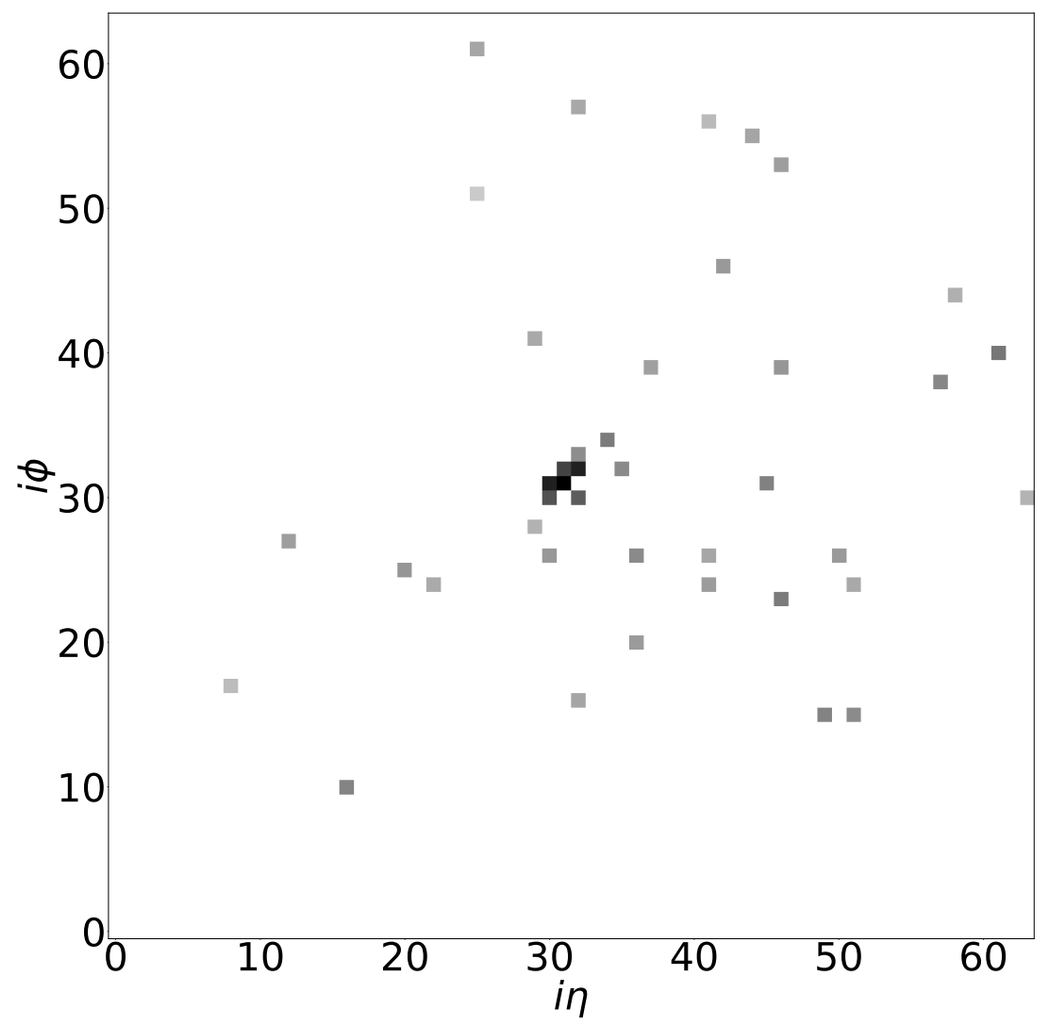


- Re-train ResNet-50 to identify the origin of jets
- Inputs are jet images = pixelated versions of calorimeter hits in 2D (η, Φ)




Note: averaged over 10k jets; 1 jet gives a ***sparse*** image



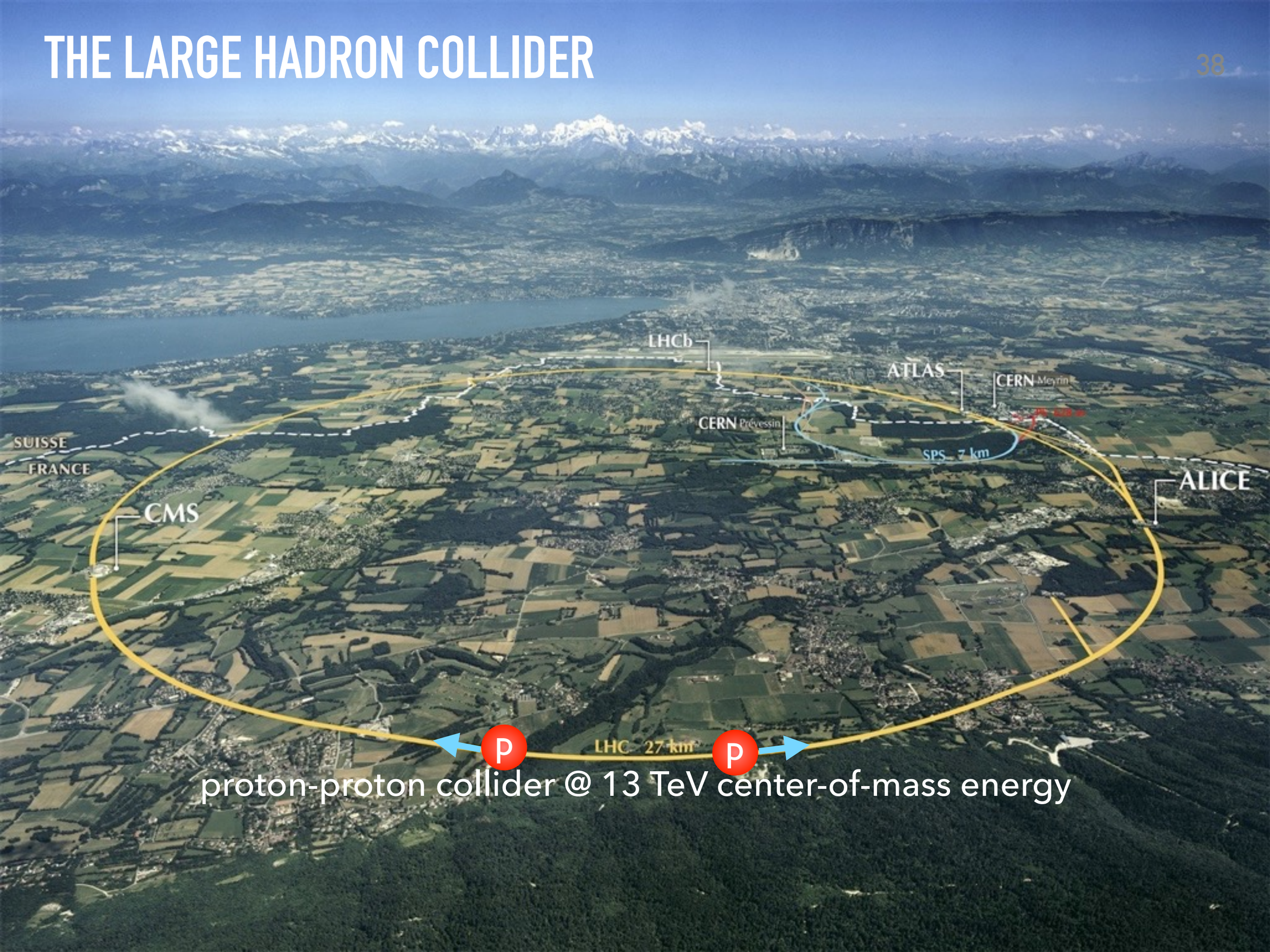



# THE LARGE HADRON COLLIDER



# THE LARGE HADRON COLLIDER

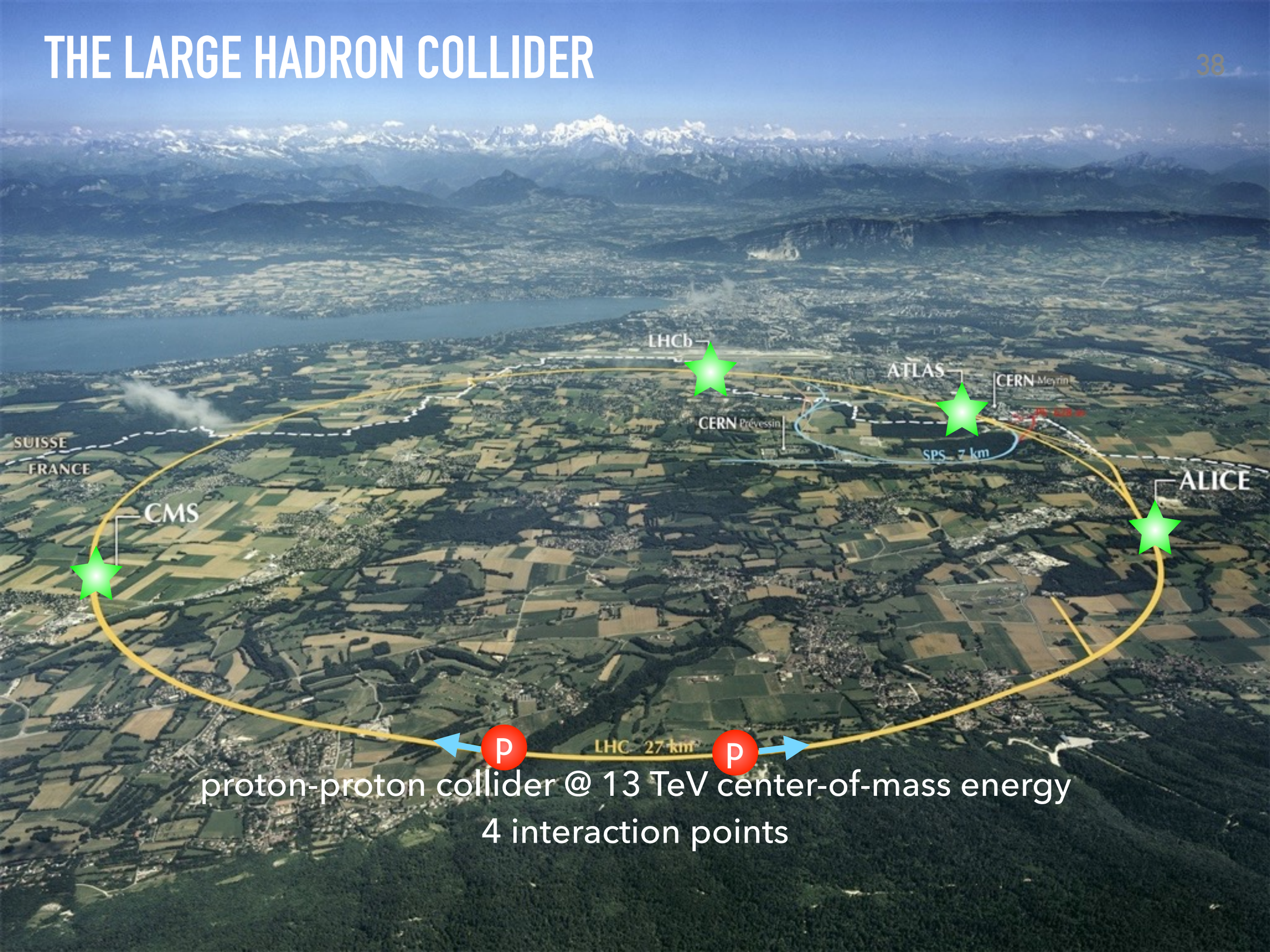





proton-proton collider @ 13 TeV center-of-mass energy

# THE LARGE HADRON COLLIDER



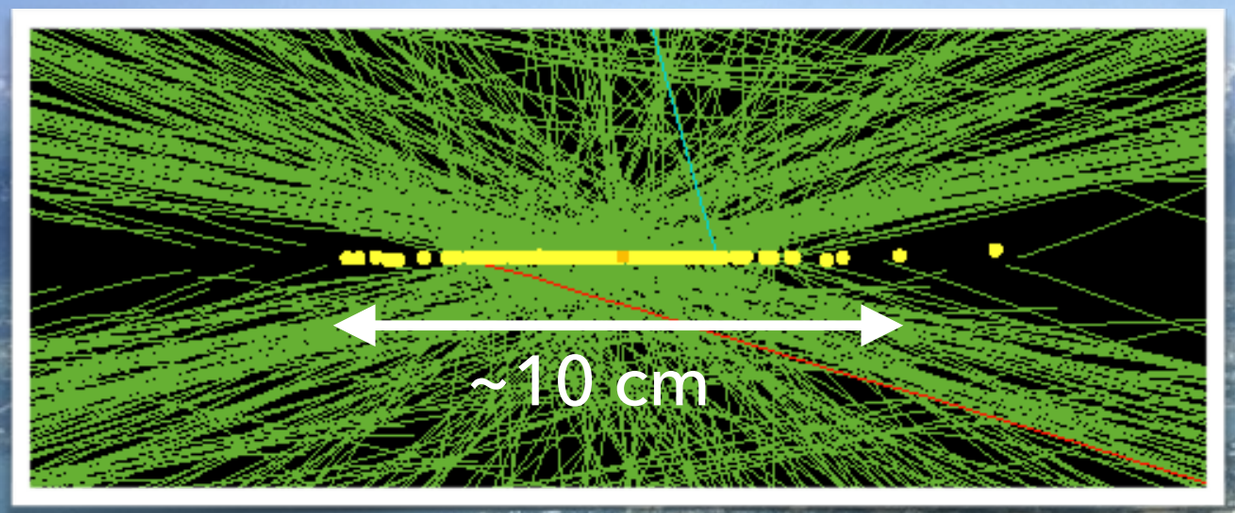



proton-proton collider @ 13 TeV center-of-mass energy

4 interaction points

# THE LARGE HADRON COLLIDER






LHCb
ATLAS
CERN Meyrin
CERN Prévessin
SPS 7 km
SUISSE
FRANCE
CMS
ALICE
P
LHC 27 km
P

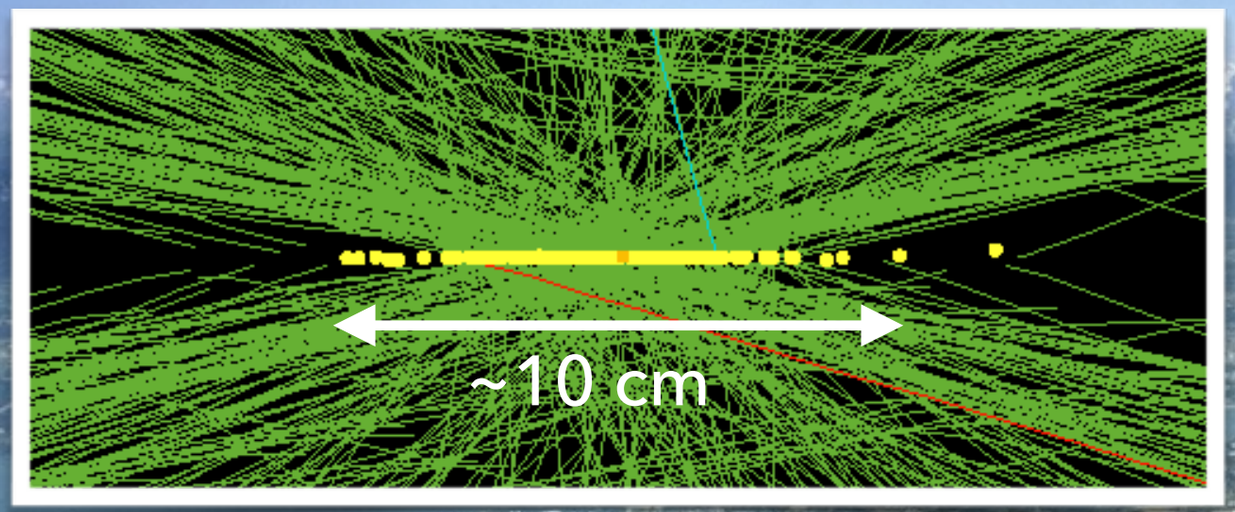
proton-proton collider @ 13 TeV center-of-mass energy

4 interaction points

40 million collisions / second

# THE LARGE HADRON COLLIDER






LHCb
ATLAS
CERN Meyrin
CERN Prévessin
SPS 7 km
SUISSE
FRANCE
CMS
ALICE
P
LHC 27 km
P

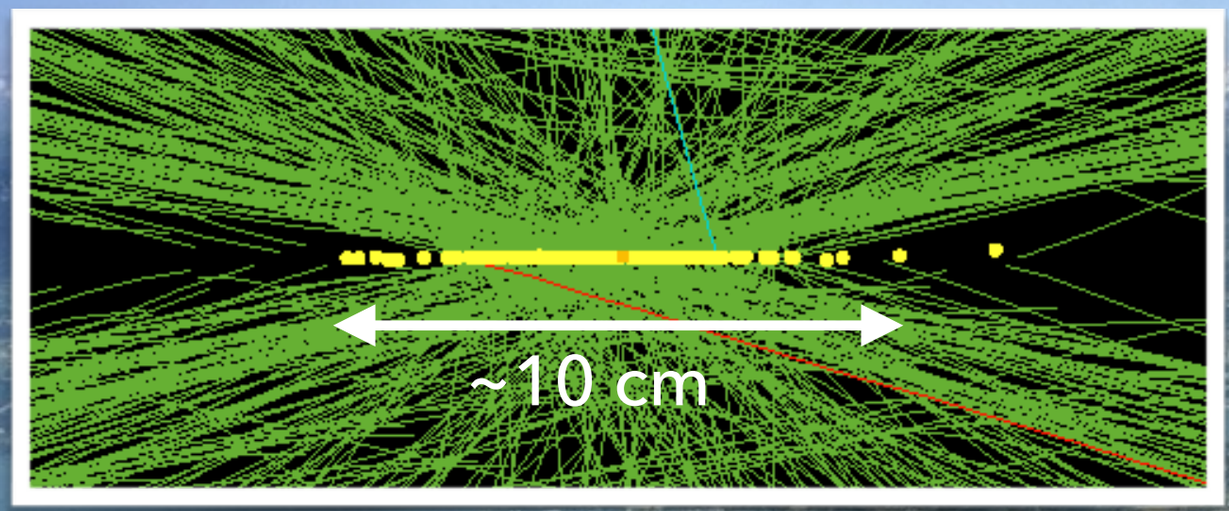
proton-proton collider @ 13 TeV center-of-mass energy

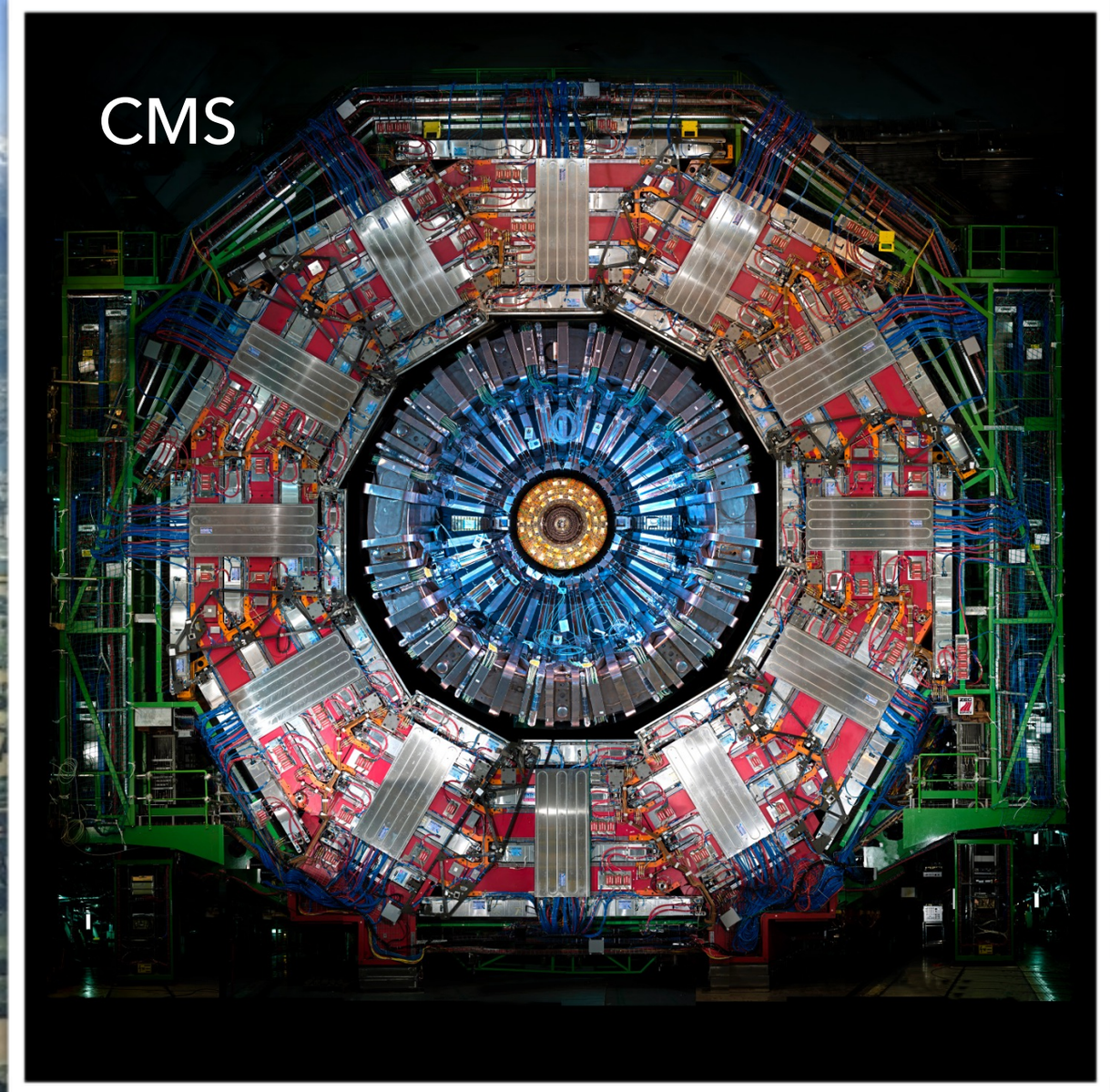
4 interaction points

40 million collisions / second

trigger selects ~1000 collisions / second

# THE LARGE HADRON COLLIDER




SUISSE
FRANCE
CMS
P
LHC 27 km
P

proton-proton collider @ 13 TeV center-of-mass energy

4 interaction points

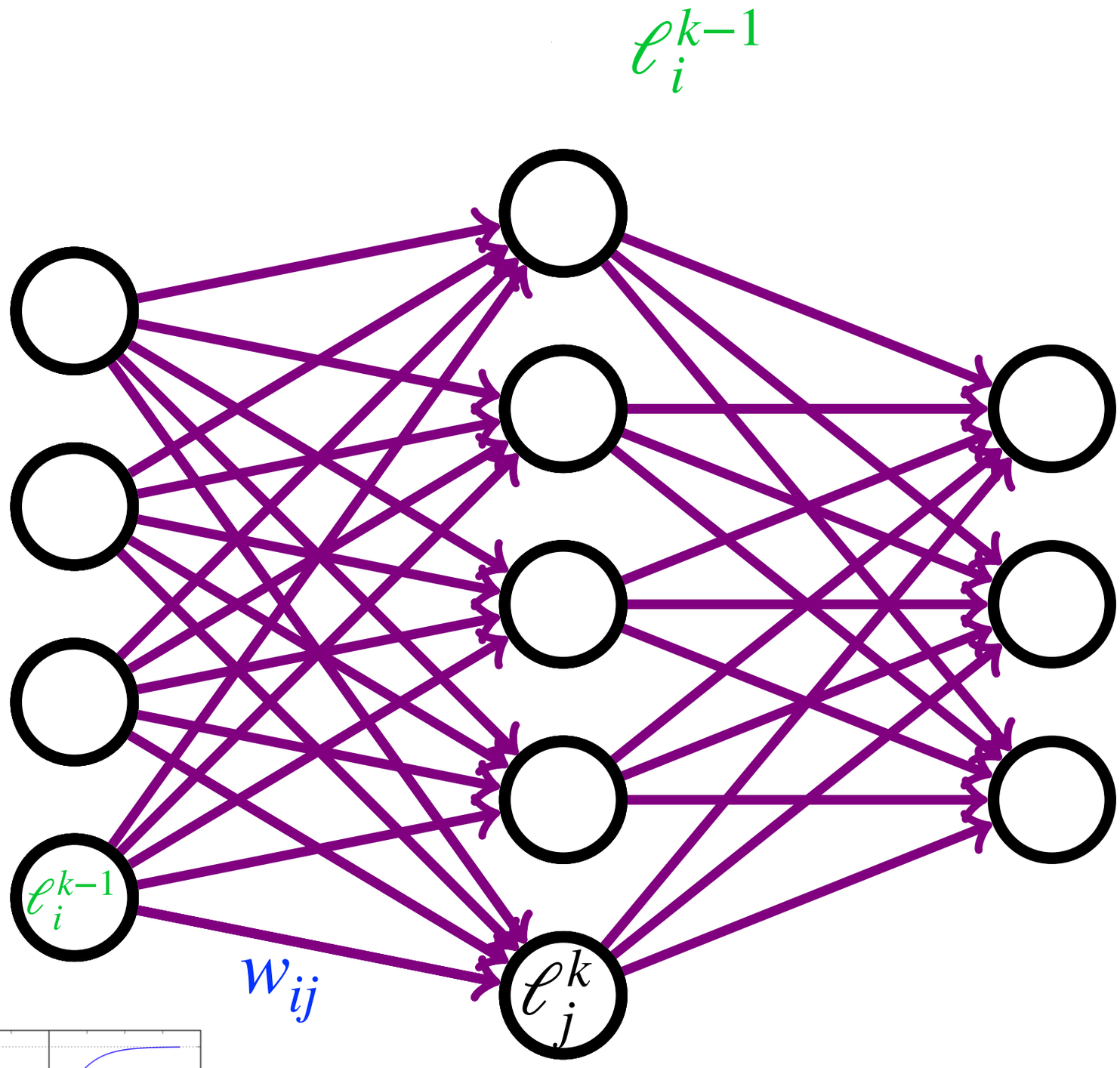
40 million collisions / second

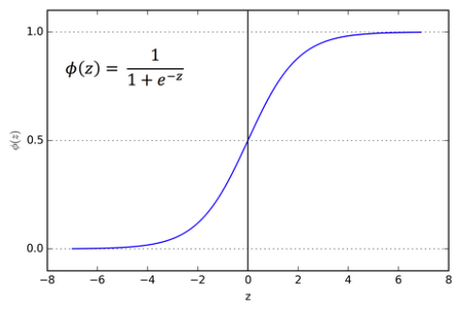
trigger selects ~1000 collisions / second

# NEURAL NETWORK (RECAP)



- Classic fully connected architecture
- Each **input** multiplied by a **weight**
- **Weighted** values are summed, **bias** is added
- Nonlinear **activation function** is applied
- Trained by varying the **parameters** to minimize a loss function (quantifies how many mistakes the network makes)




A sufficiently "wide" neural network can approximate any function!

# MACHINE LEARNING STEPS



- **Step 0**: Define the problem (choice of loss function)

# MACHINE LEARNING STEPS



- **Step 0**: Define the problem (choice of loss function)

$$L = -y\log(p) + (1-y)\log(1-p)$$

$y$ = 0 (background) or 1 (signal)

$p$ = output of our NN (probability of signal)

# MACHINE LEARNING STEPS



- **Step 0**: Define the problem (choice of loss function)

$$L = -y\log(p) + (1-y)\log(1-p)$$

$y$ = 0 (background) or 1 (signal)

$p$ = output of our NN (probability of signal)

if $p \sim y$, $L \sim 0$ (correct!)

if $p \sim 1\text{-}y$, $L \sim \infty$ (incorrect!)

# MACHINE LEARNING STEPS



- **Step 0**: Define the problem (choice of loss function)

$$L = -y\log(p) + (1-y)\log(1-p)$$

$y$ = 0 (background) or 1 (signal)

$p$ = output of our NN (probability of signal)

if $p \sim y$, $L \sim 0$ (correct!)

if $p \sim 1\text{-}y$, $L \sim \infty$ (incorrect!)

- Step 1: Acquire lots of labeled data and split into training and testing sets

# MACHINE LEARNING STEPS



- **Step 0**: Define the problem (choice of loss function)

$$L = -y\log(p) + (1-y)\log(1-p)$$

$y$ = 0 (background) or 1 (signal)

$p$ = output of our NN (probability of signal)

if $p \sim y$, $L \sim 0$ (correct!)

if $p \sim 1\text{-}y$, $L \sim \infty$ (incorrect!)

- Step 1: Acquire lots of labeled data and split into training and testing sets
- Step 2: Select input features

# MACHINE LEARNING STEPS



- **Step 0**: Define the problem (choice of loss function)

$$L = -y\log(p) + (1-y)\log(1-p)$$

$y$ = 0 (background) or 1 (signal)

$p$ = output of our NN (probability of signal)

if $p \sim y$, $L \sim 0$ (correct!)

if $p \sim 1\text{-}y$, $L \sim \infty$ (incorrect!)

- Step 1: Acquire lots of labeled data and split into training and testing sets
- Step 2: Select input features
- Step 3: Explore/train different neural network architectures

# MACHINE LEARNING STEPS



- **Step 0**: Define the problem (choice of loss function)

$$L = -y\log(p) + (1-y)\log(1-p)$$

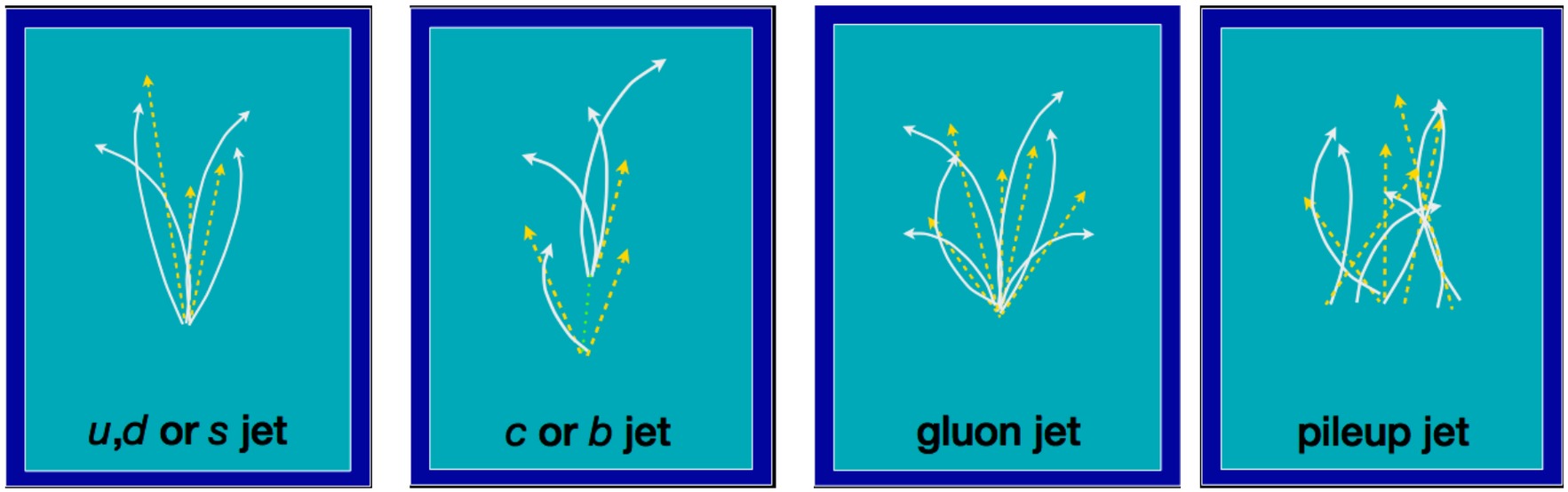
$y$ = 0 (background) or 1 (signal)
$p$ = output of our NN (probability of signal)

if $p \sim y$, $L \sim 0$ (correct!)
if $p \sim 1\text{-}y$, $L \sim \infty$ (incorrect!)

- Step 1: Acquire lots of labeled data and split into training and testing sets
- Step 2: Select input features
- Step 3: Explore/train different neural network architectures
- **Step 4**: Evaluate performance

# CASE STUDY FOR THE TRIGGER: JET TAGGING



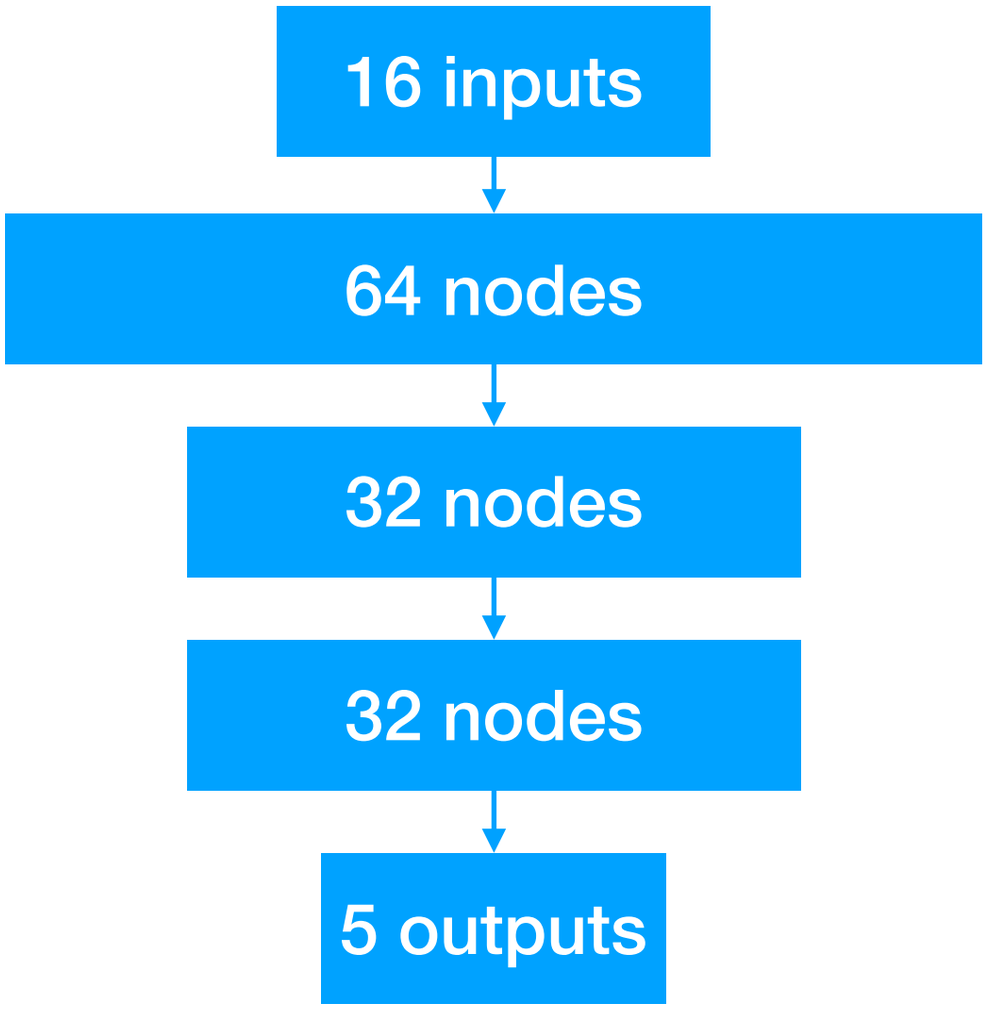


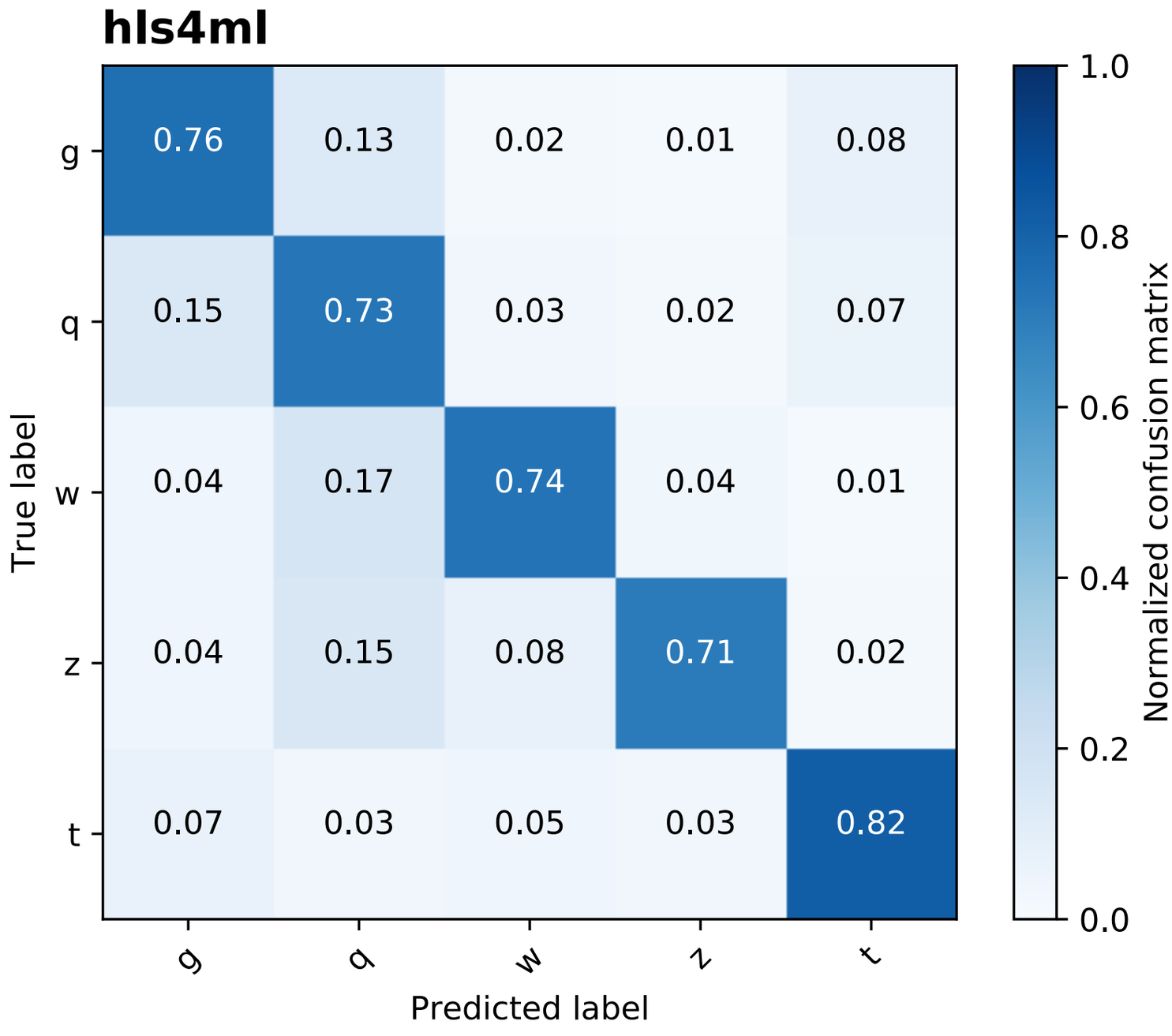

- Use machine learning to classify jets in the trigger

# CASE STUDY FOR THE TRIGGER: JET TAGGING



Smaller NN correctly identifies jets 70-80% of the time



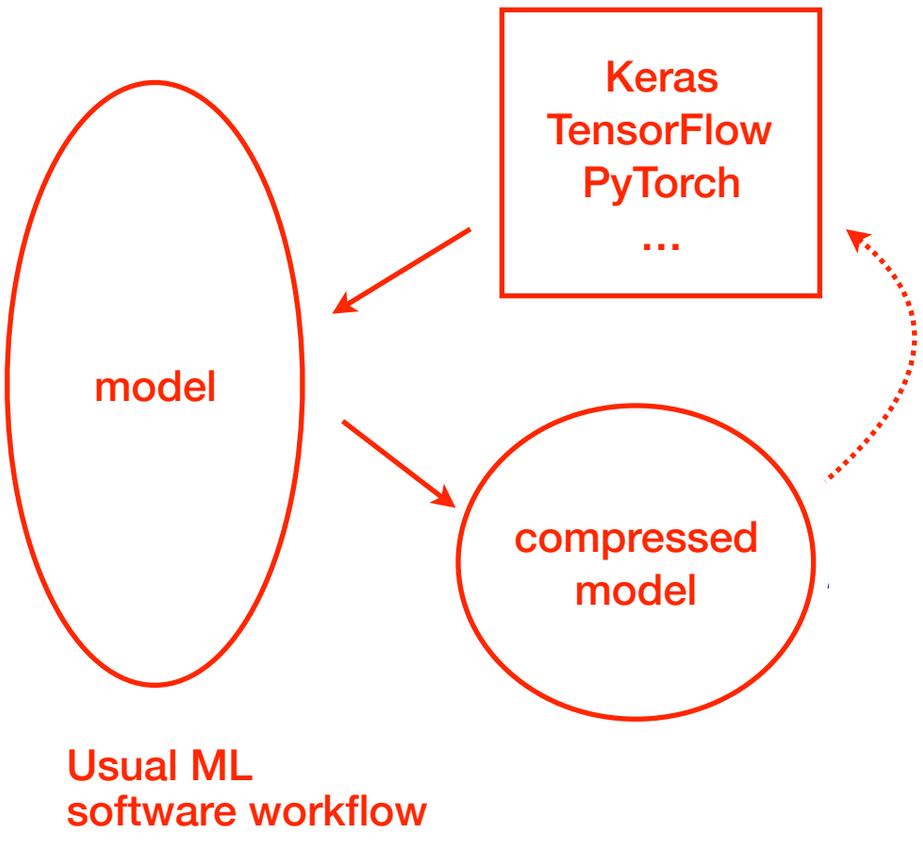




- Use machine learning to classify jets in the trigger

# DESIGN EXPLORATION WITH HLS4ML



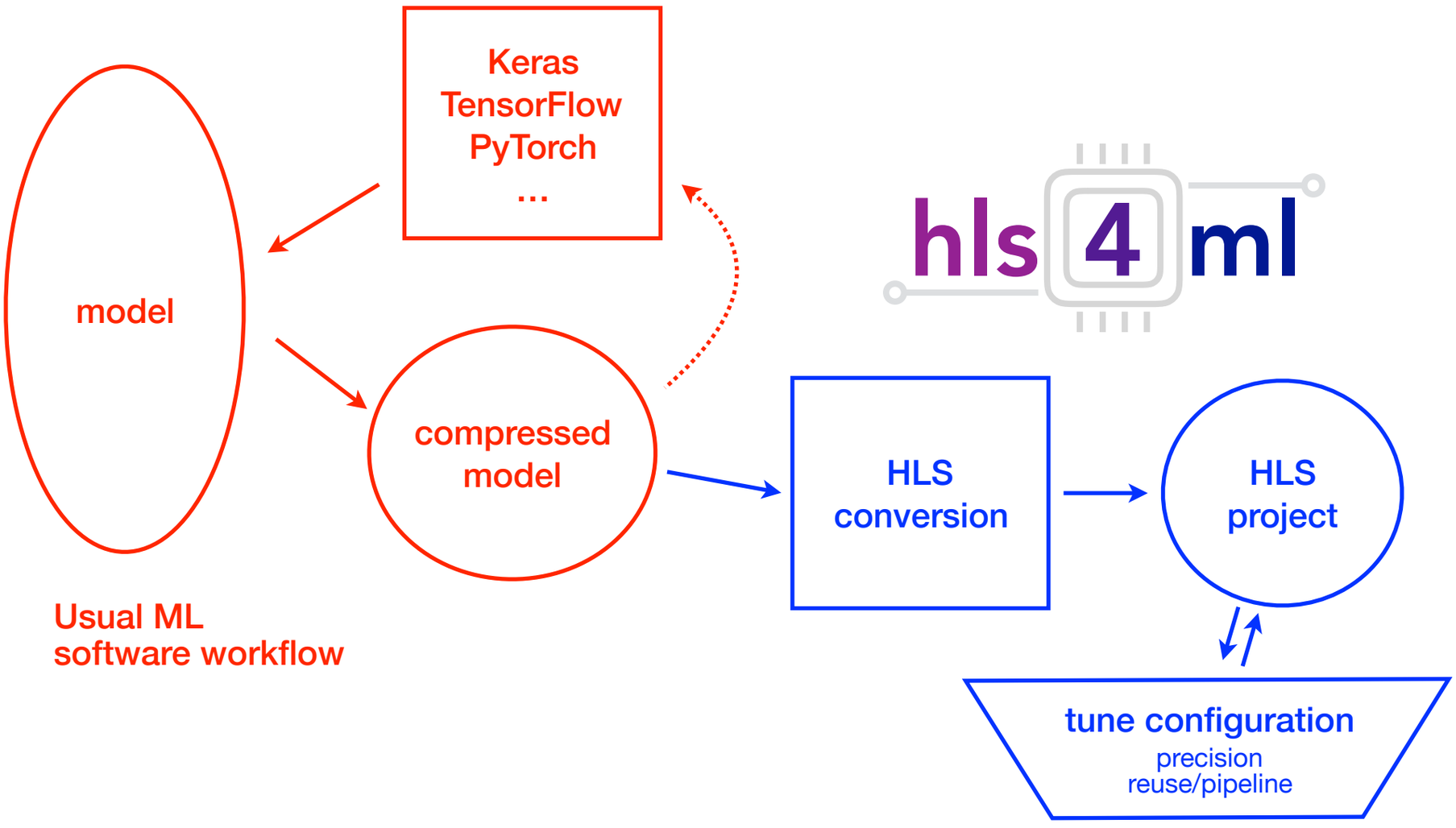
- Tool called **hls4ml** for physicists or ML experts to translate **ML algorithms** into **FPGA firmware**




# DESIGN EXPLORATION WITH HLS4ML



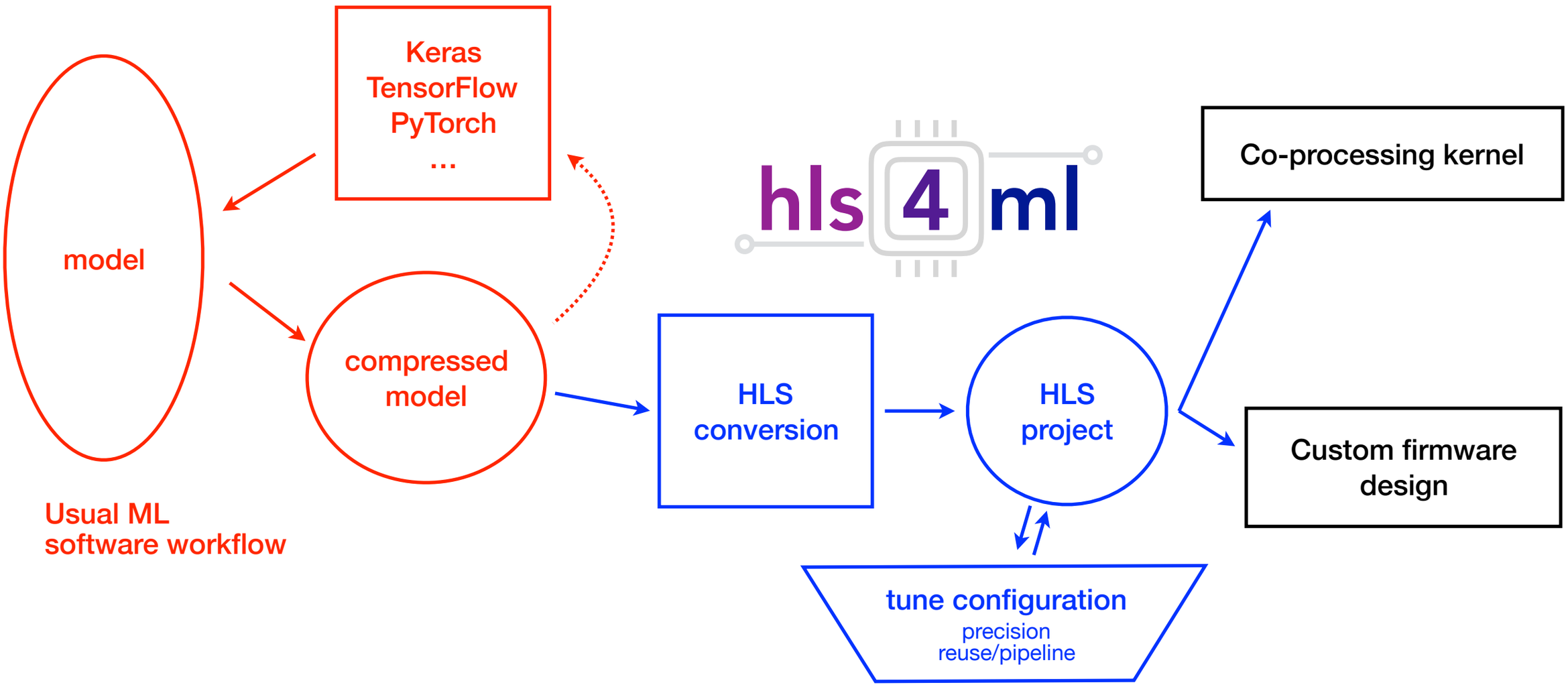
- Tool called **hls4ml** for physicists or ML experts to translate **ML algorithms** into **FPGA firmware**



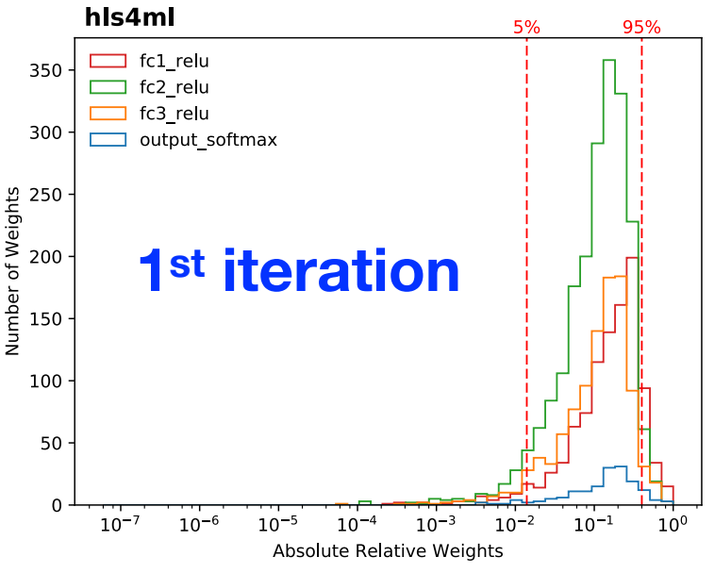
# DESIGN EXPLORATION WITH HLS4ML



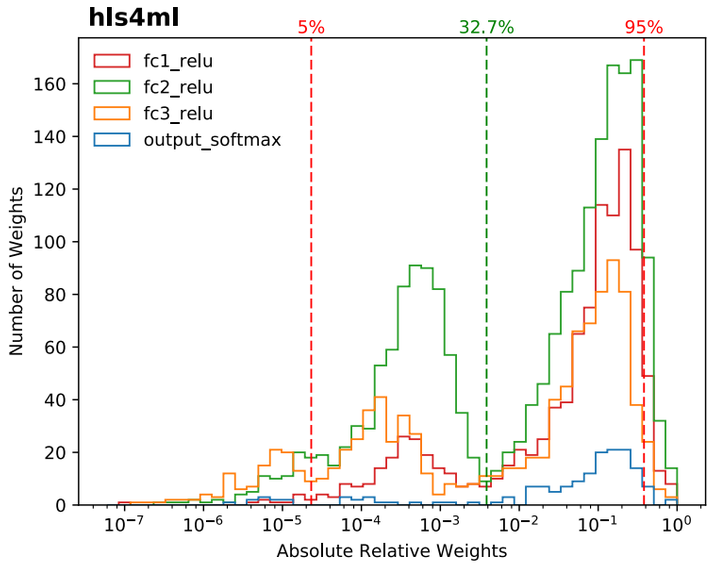
- Tool called **hls4ml** for physicists or ML experts to translate **ML algorithms** into **FPGA firmware**



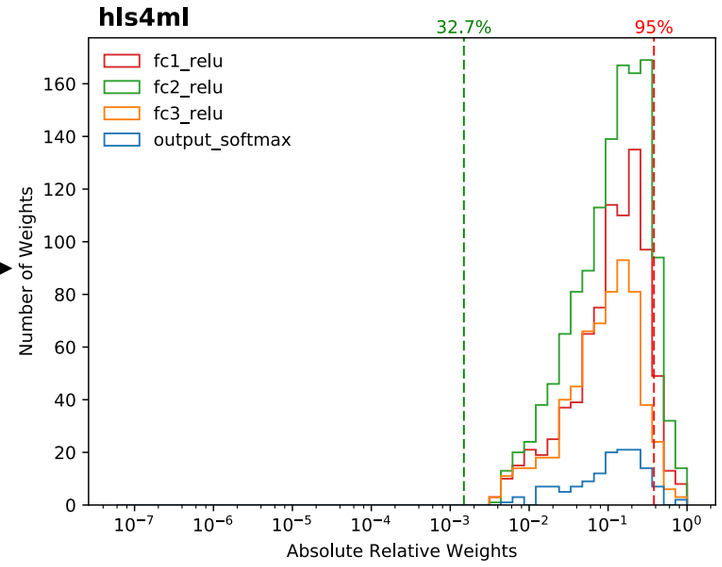
**1st iteration**




Train with $L_1$

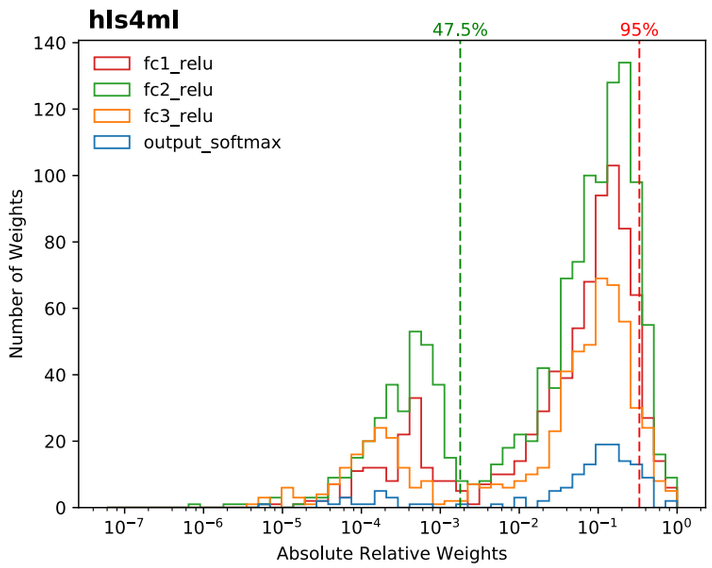



Prune

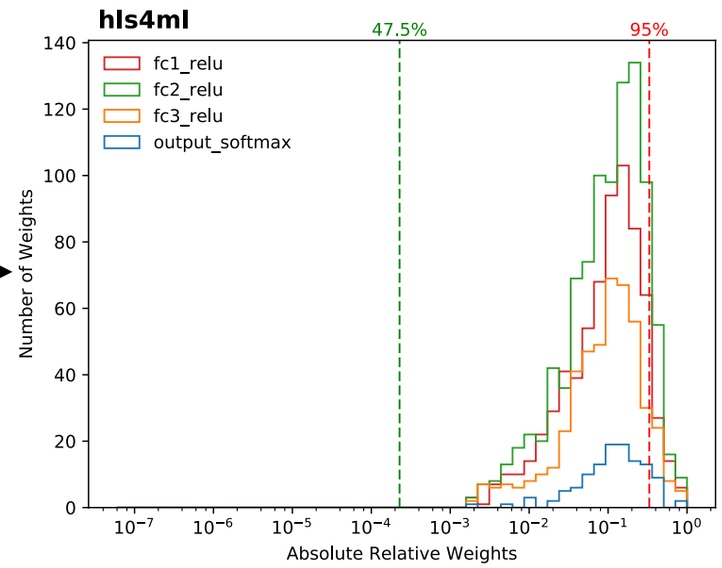



**2nd iteration**

Retrain with $L_1$



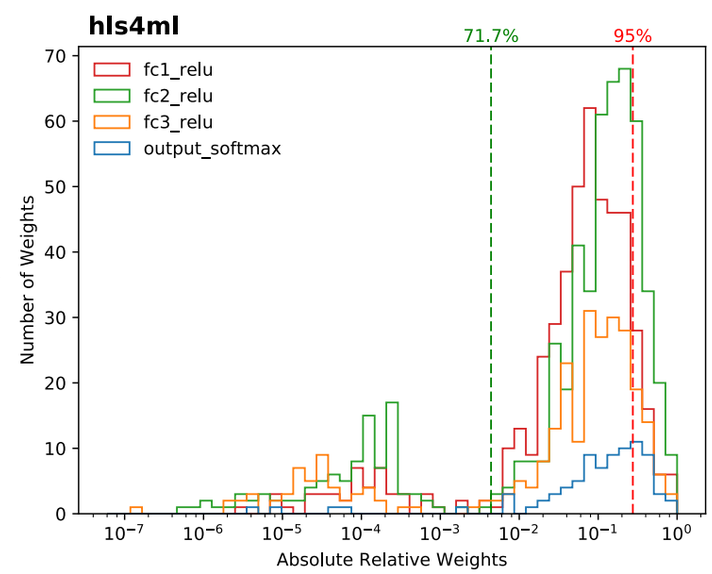

Prune



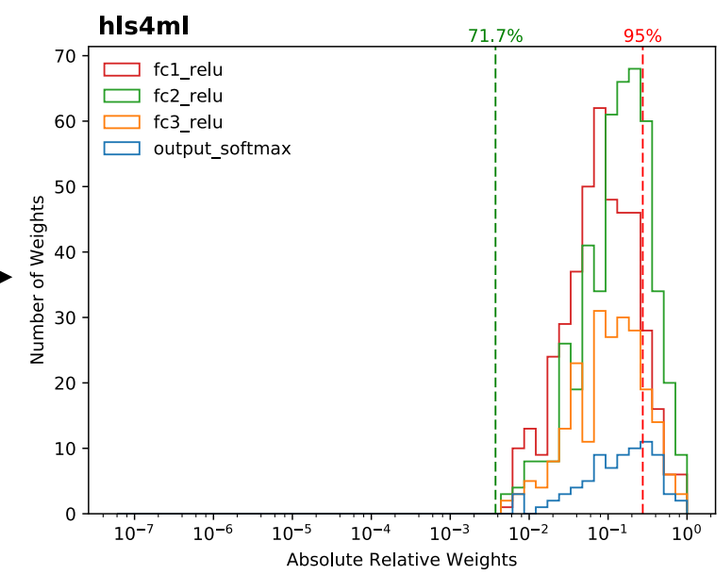

⋮

**7th iteration**

Retrain with $L_1$

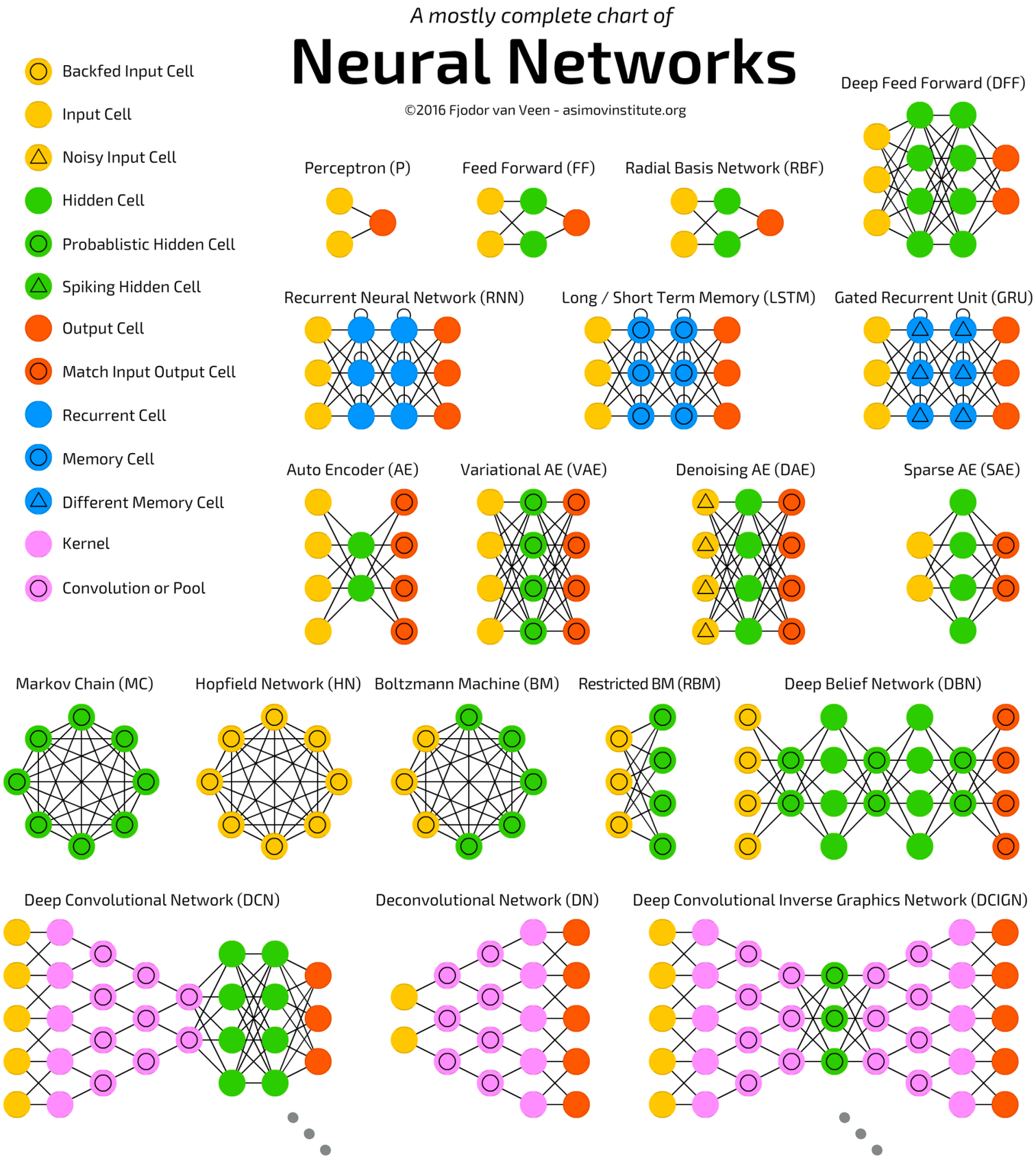



Prune



# NEURAL NETWORK ZOO

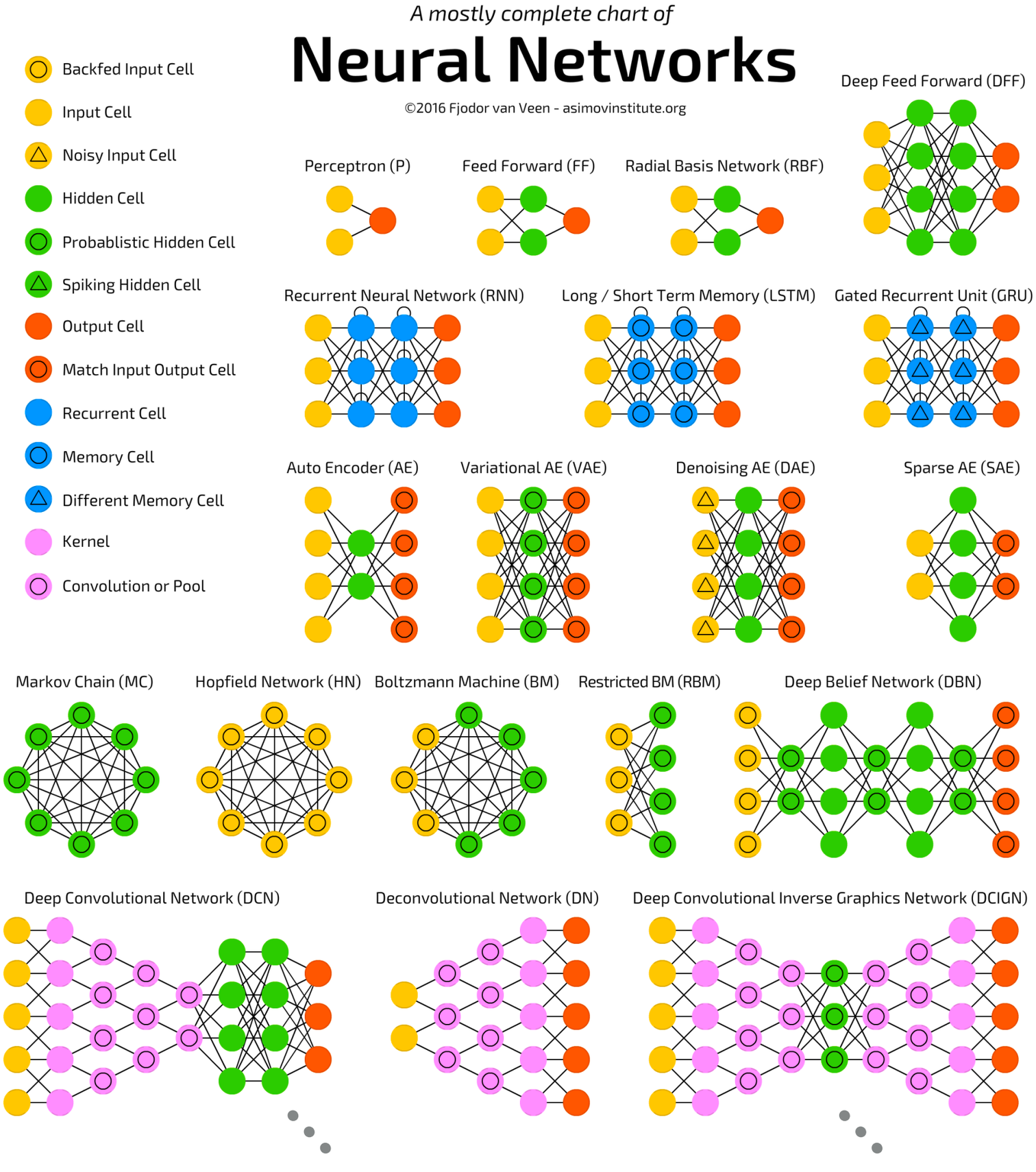




# NEURAL NETWORK ZOO



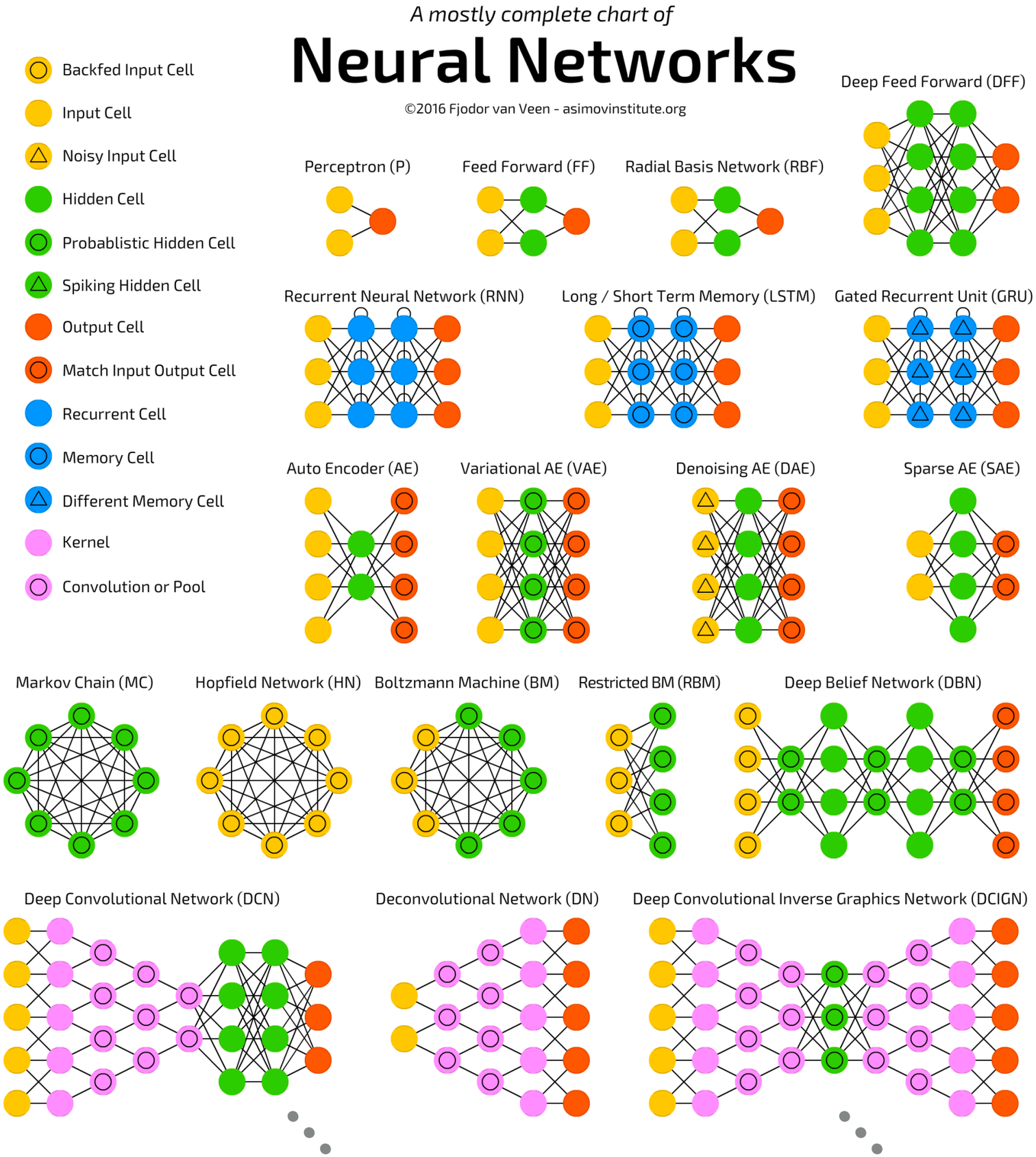
- You have a task to accomplish, which can be represented as a smooth function from your inputs to the answer you want



# NEURAL NETWORK ZOO



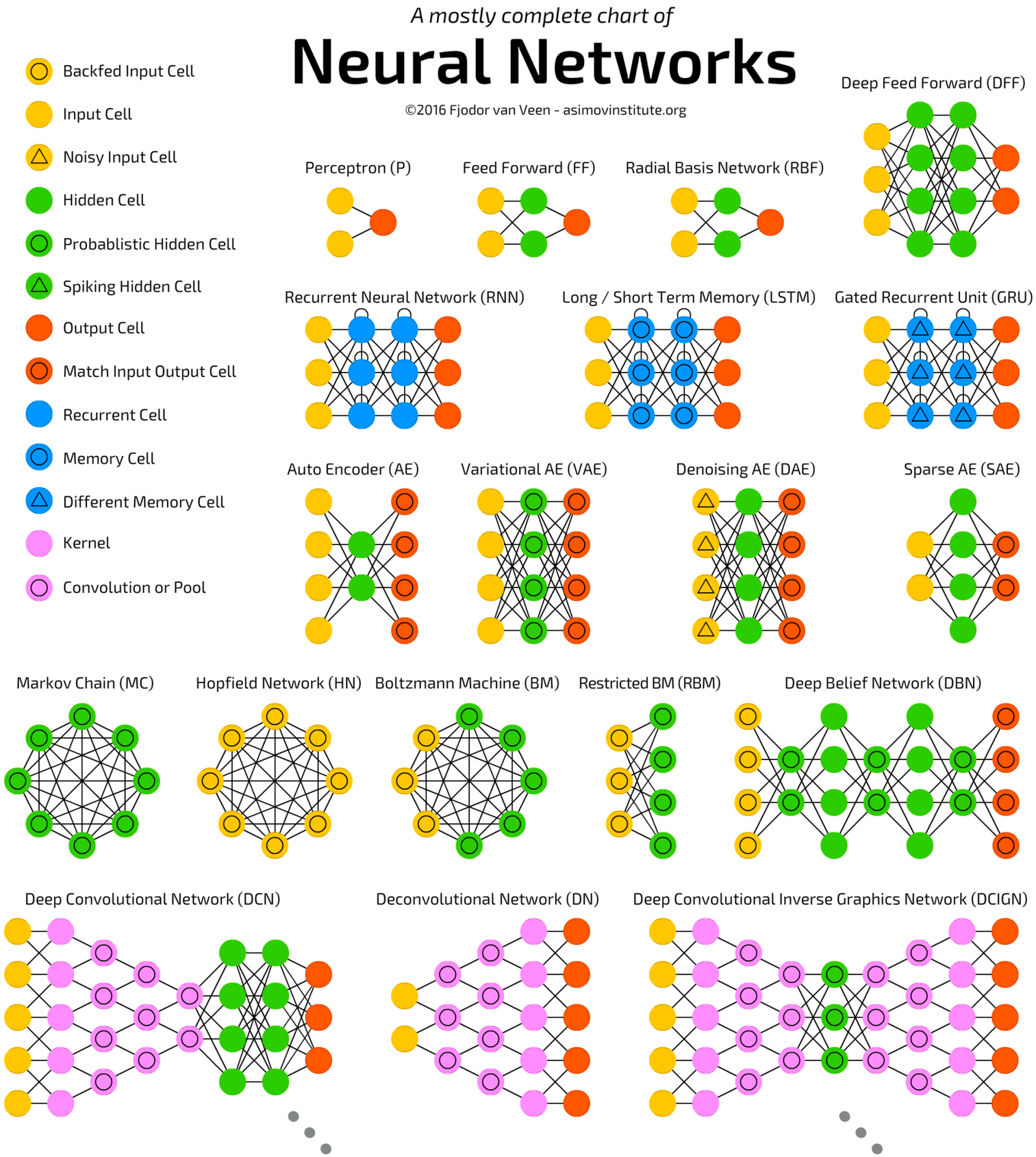
- You have a task to accomplish, which can be represented as a smooth function from your inputs to the answer you want
  - Train an algorithm to learn an approximation of the optimal solution function (Machine Learning)



# NEURAL NETWORK ZOO



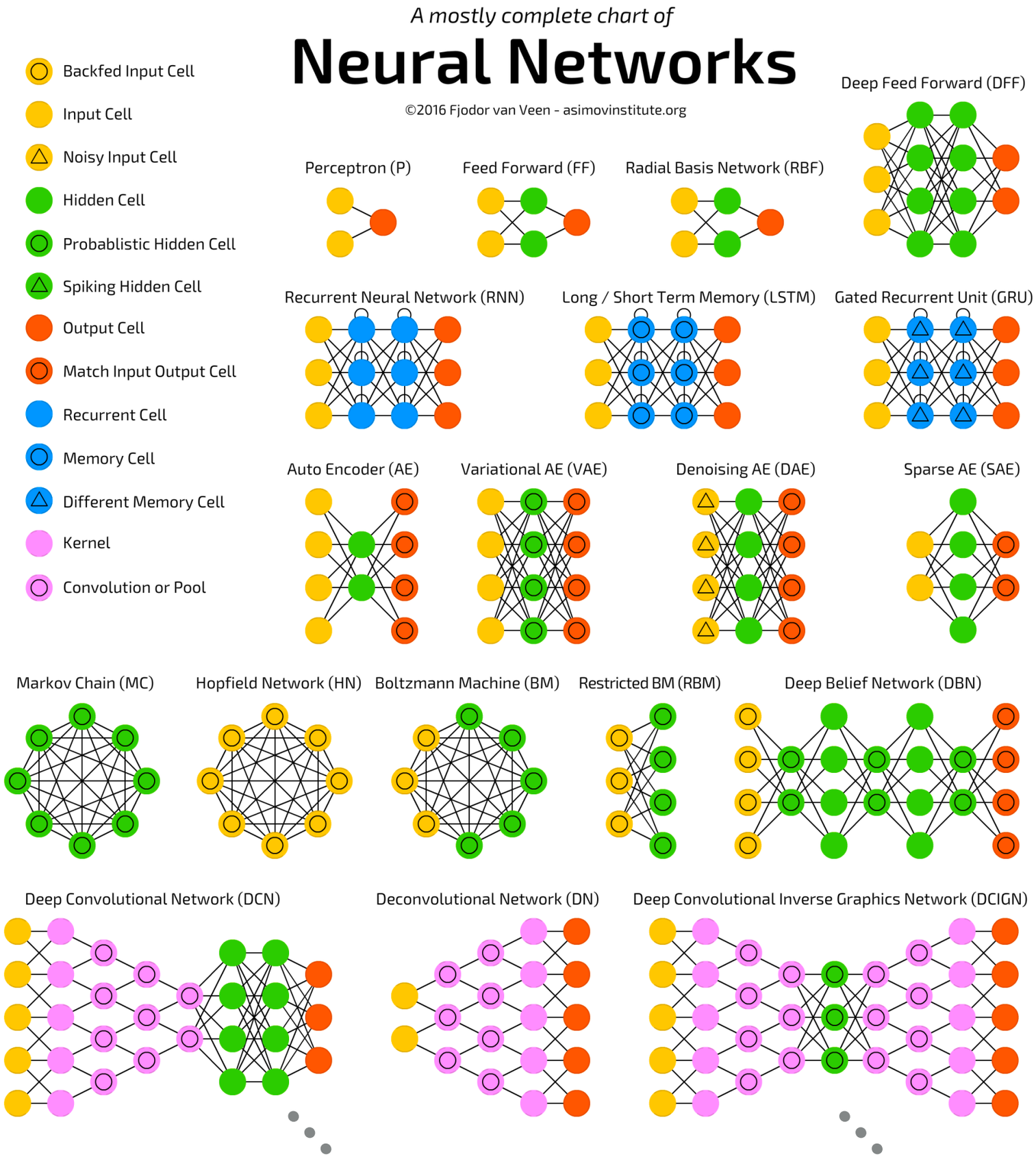
- You have a task to accomplish, which can be represented as a smooth function from your inputs to the answer you want
  - Train an algorithm to learn an approximation of the optimal solution function (Machine Learning)
- NNs are the best ML solution on the market *today*



# NEURAL NETWORK ZOO



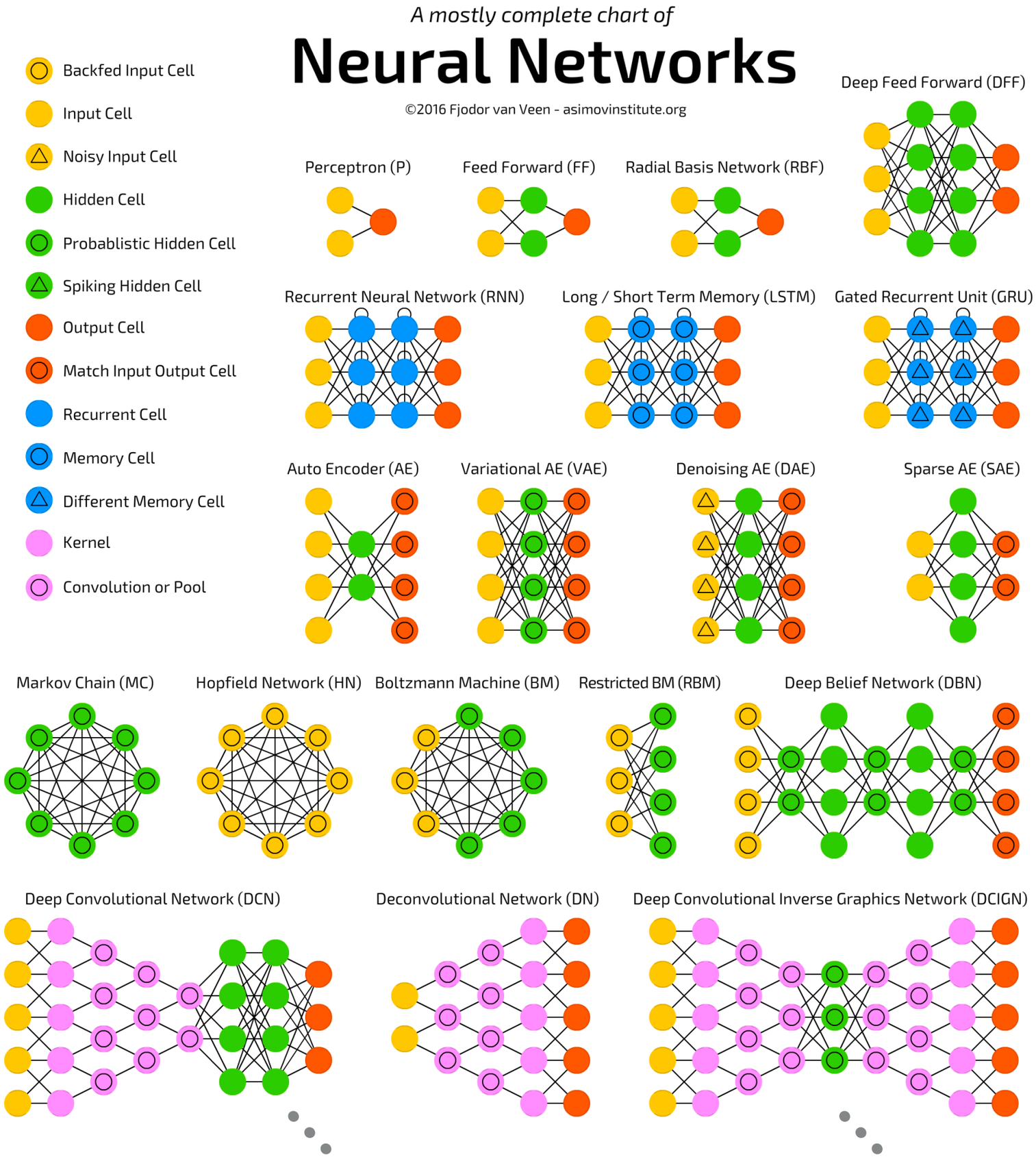
- You have a task to accomplish, which can be represented as a smooth function from your inputs to the answer you want
  - Train an algorithm to learn an approximation of the optimal solution function (Machine Learning)
- NNs are the best ML solution on the market *today*
  - Each node performs a math operation on the input



# NEURAL NETWORK ZOO

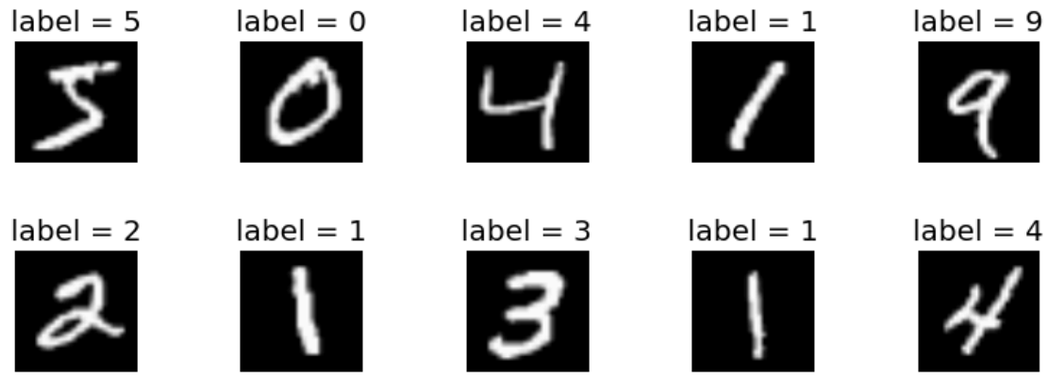


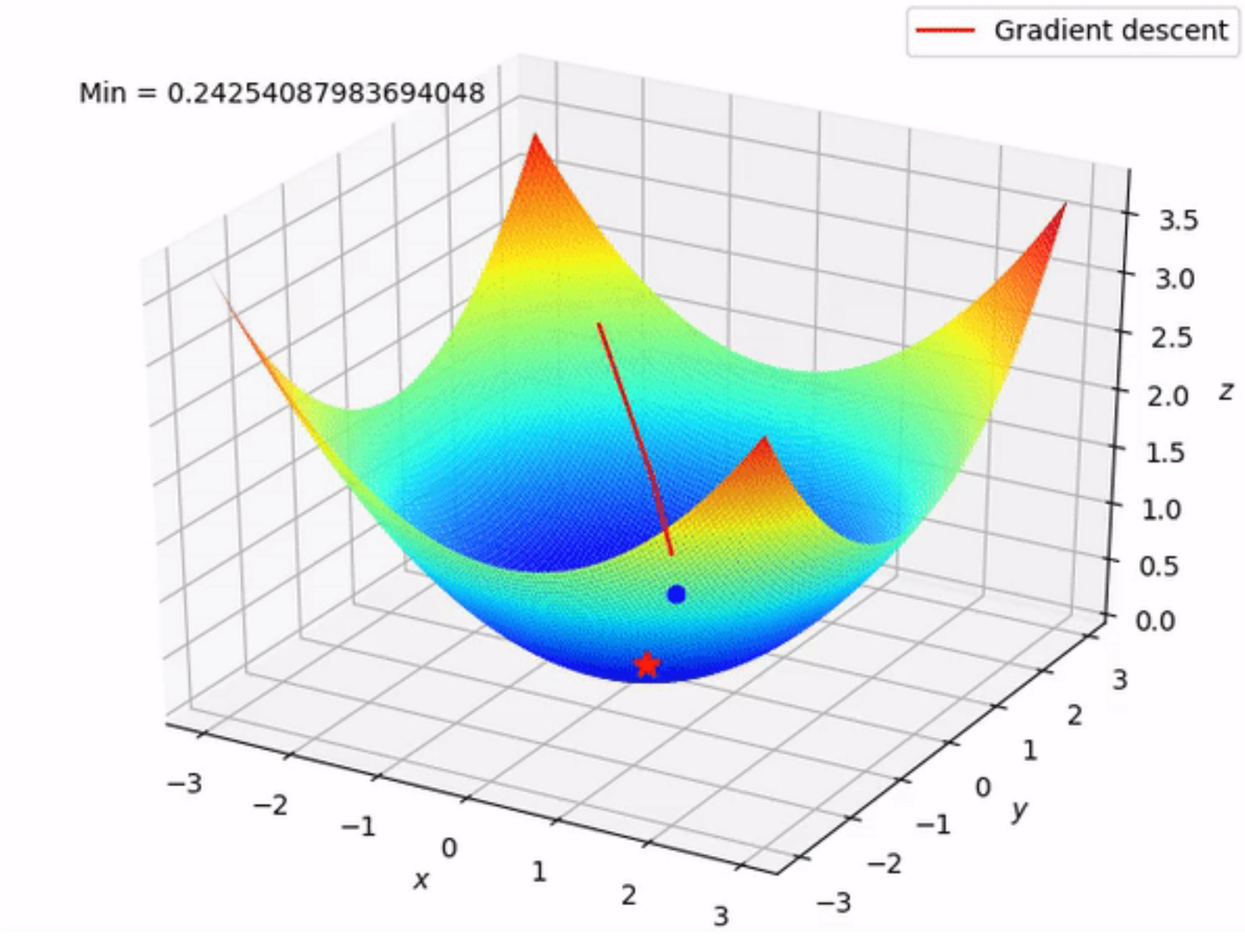
- You have a task to accomplish, which can be represented as a smooth function from your inputs to the answer you want
  - Train an algorithm to learn an approximation of the optimal solution function (Machine Learning)
- NNs are the best ML solution on the market *today*
  - Each node performs a math operation on the input
  - Edges represent the flow of nodes' inputs & outputs



# TRAINING



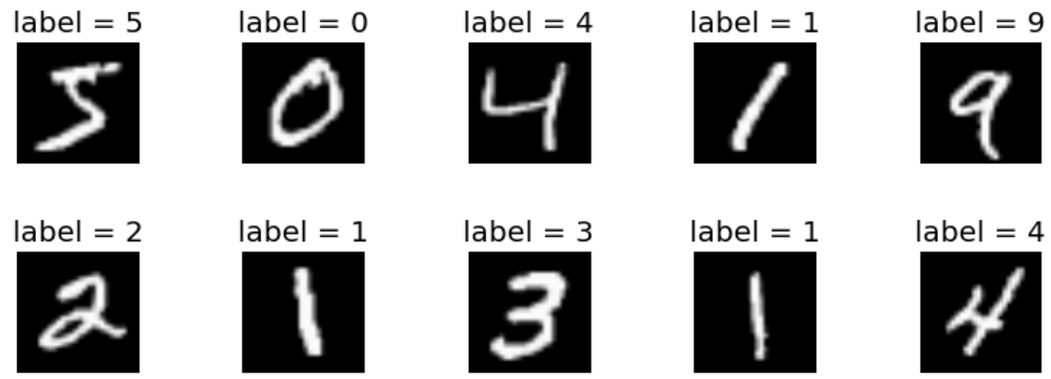

label = 5
label = 0
label = 4
label = 1
label = 9
label = 2
label = 1
label = 3
label = 1
label = 4

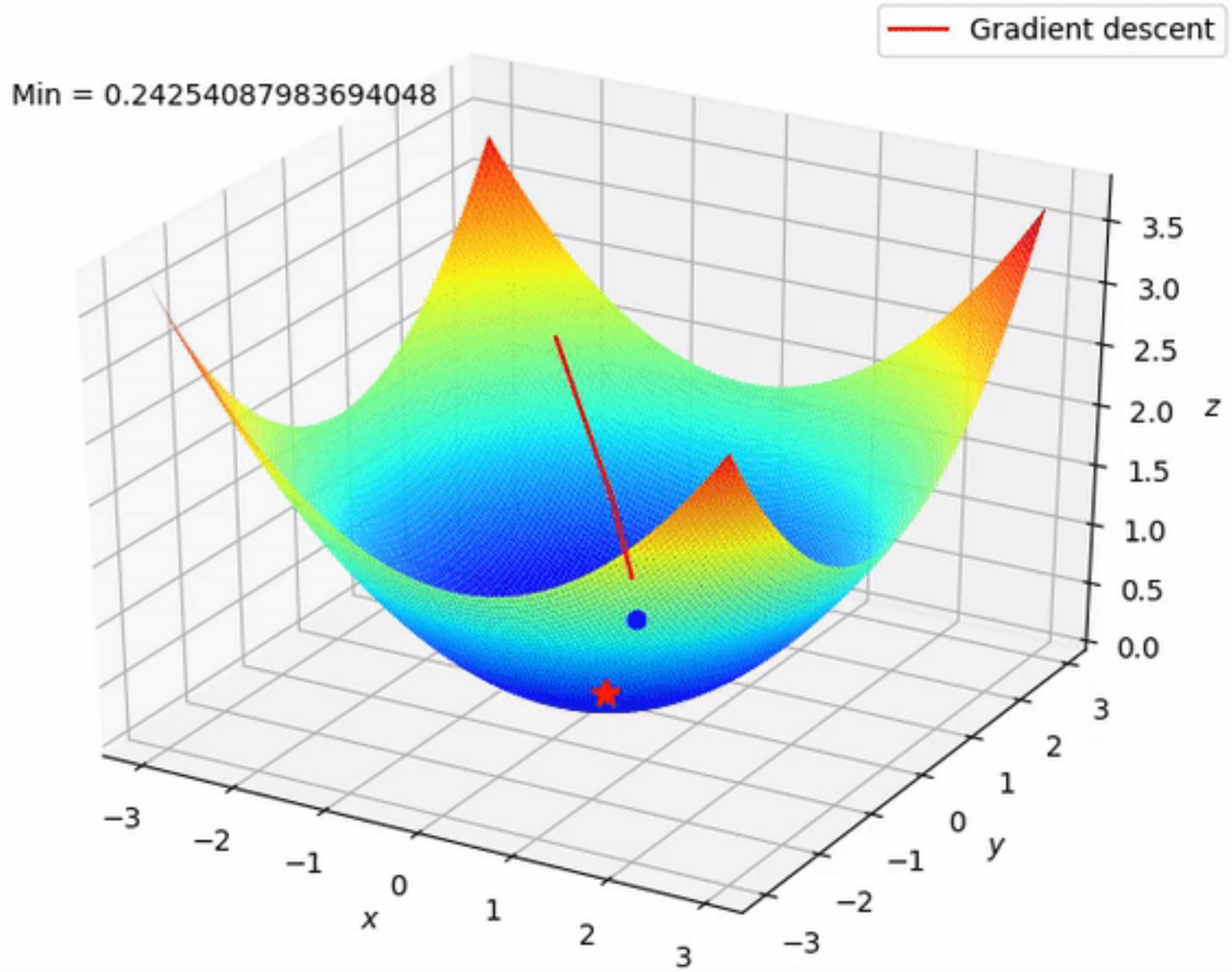

Gradient descent
Min = 0.24254087983694048
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
z
−3
−2
−1
0
1
2
3
x
y

# TRAINING



- A network is trained by specifying inputs, targets, and a loss function
  - Target is what the network should learn for that input, can be a “truth” label (supervised) or the input itself (unsupervised)
  - Loss function quantifies how many mistakes the network makes
- Training is the minimization of the loss function by varying the network parameters

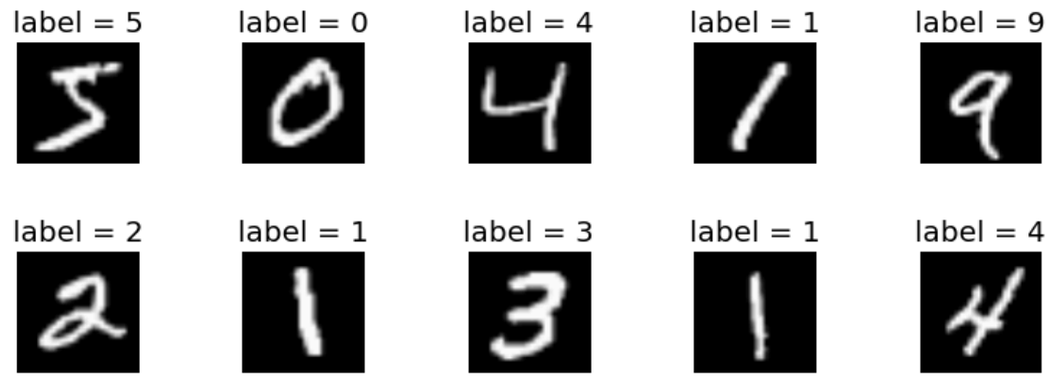


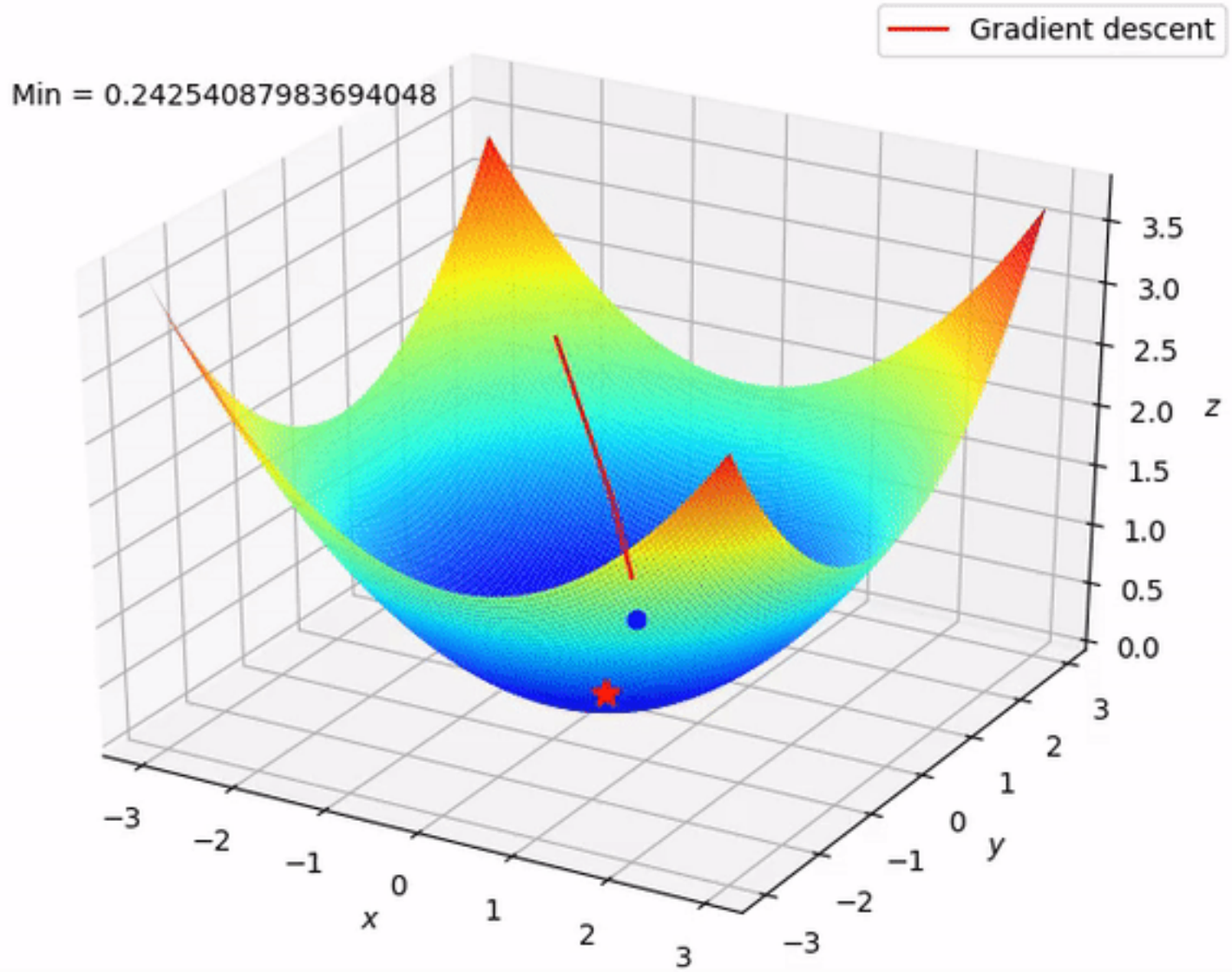



# TRAINING



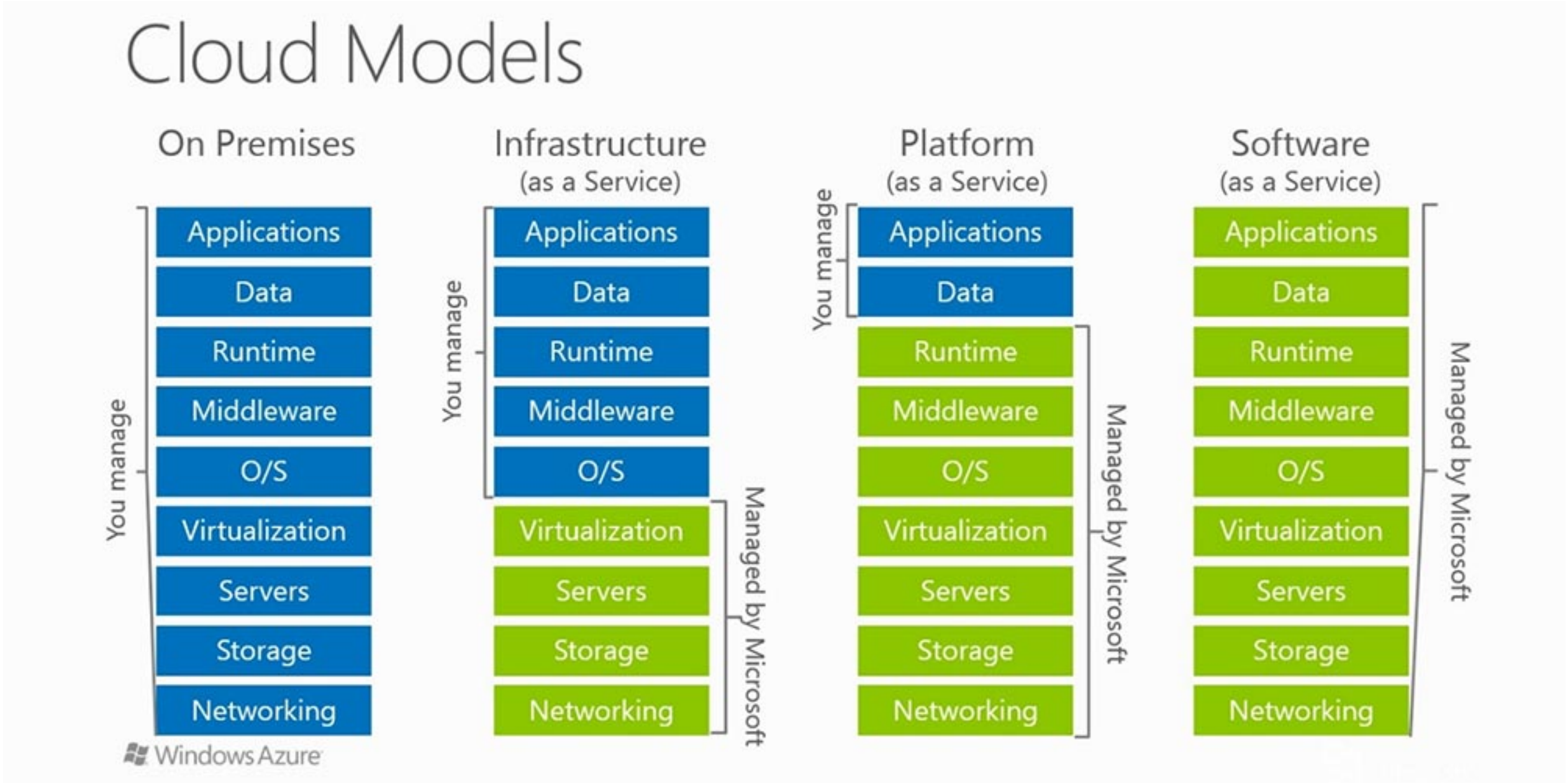
- A network is trained by specifying inputs, targets, and a loss function
  - Target is what the network should learn for that input, can be a “truth” label (supervised) or the input itself (unsupervised)
  - Loss function quantifies how many mistakes the network makes
- Training is the minimization of the loss function by varying the network parameters



# "AS A SERVICE"




Cloud Models
On Premises
Infrastructure (as a Service)
Platform (as a Service)
Software (as a Service)
Applications
Data
Runtime
Middleware
O/S
Virtualization
Servers
Storage
Networking
You manage
Managed by Microsoft
Windows Azure