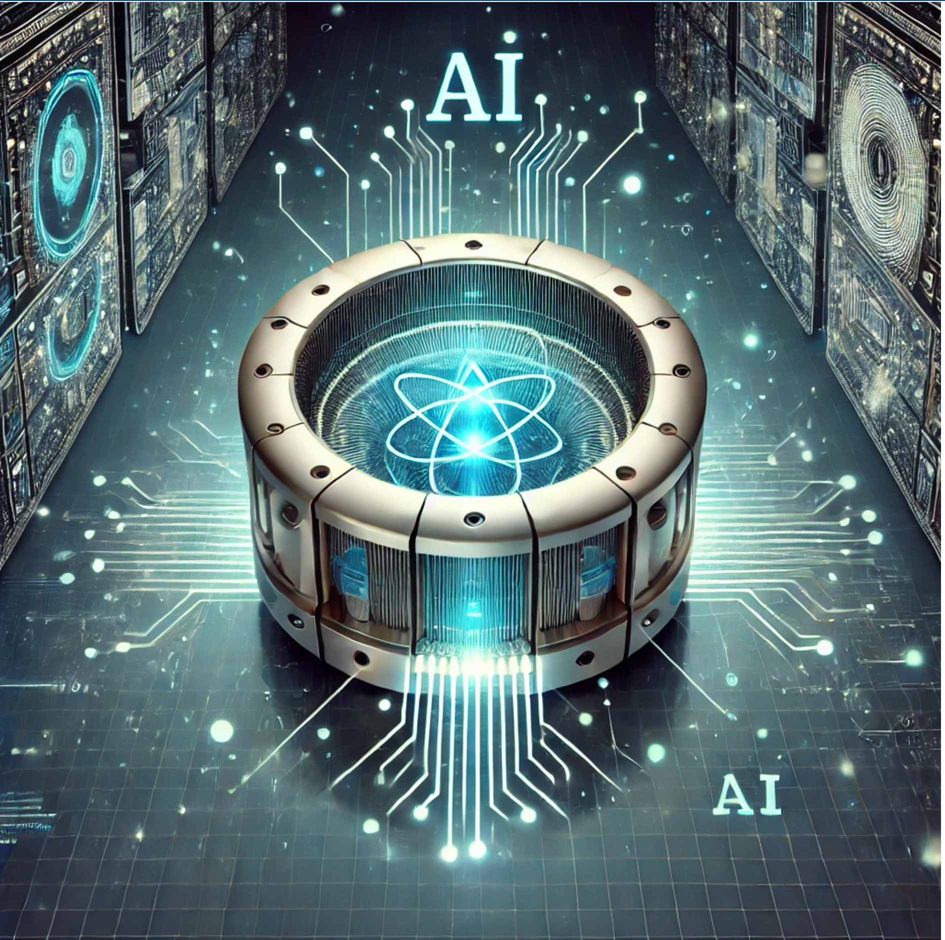
Fermilab U.S. DEPARTMENT OF ENERGY Office of Science



From analog to automation to assistants: detectors, AI, and you

Nhan Tran, Fermilab

EDIT school 2024 & Fermilab W&C
22 November 2024

An interesting time, personally

- October: 2024 Fast ML for Science Conference
  – https://indico.cern.ch/e/fastml2024
- November: DOE Office of Science AI Roundtable on Experiments and Facilities
  – Including biology, environmental, material, fusion sciences, HEP and nuclear physics
- November: DOE-wide discussions on the role of supercomputing and AI
  – FASST initiative: Frontiers in AI for Science, Security and Technology

Huge credit to many folks and stimulating discussions!

What is in this talk and what is not

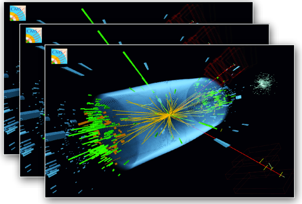
- ❌ AI for physics analysis including theory, simulation, reconstruction, and interpretation - let's just assume physics + AI is awesome :)

- ✅ AI as it applies to detectors and instruments* to:
  - <u>Accelerate</u> new physics discoveries
    - Unearth new physics signatures much more quickly
    - Operate instruments and detectors much more efficiently
      - Enabling faster analysis
      - Requiring less data needed to get to the same results
      - Reducing operational resources and increasing long term reliability

* nearly everything I will say applies to particle accelerators too

Outline

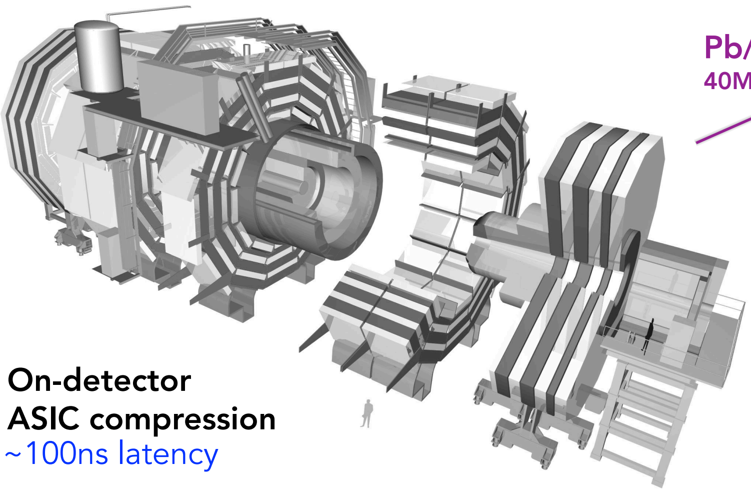
- Fast and Slow
- Fast ML and hardware codesign
- Fast and Slow ML together
- Slow ML & Real-Time



CMS Experiment

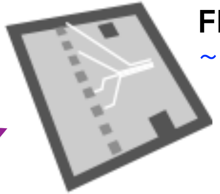
40MHz collision rate

~1B detector channels

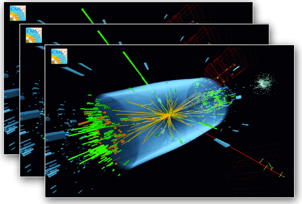


On-detector
ASIC compression
~100ns latency

Pb/s
40MHz



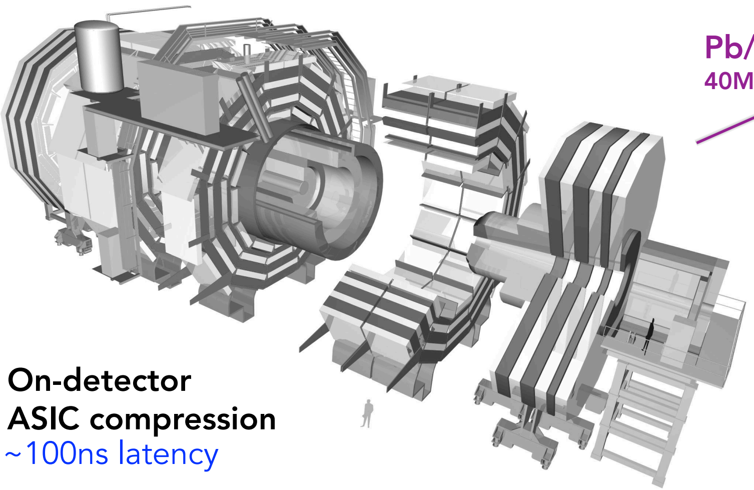
FPGA filter stack
~us latency



CMS Experiment

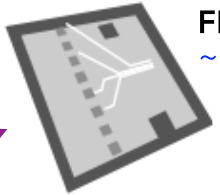
40MHz collision rate

~1B detector channels



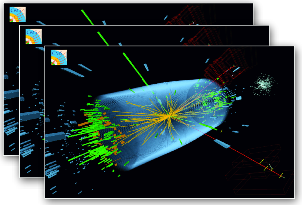
On-detector
ASIC compression
~100ns latency

Pb/s
40MHz



FPGA filter stack
~us latency

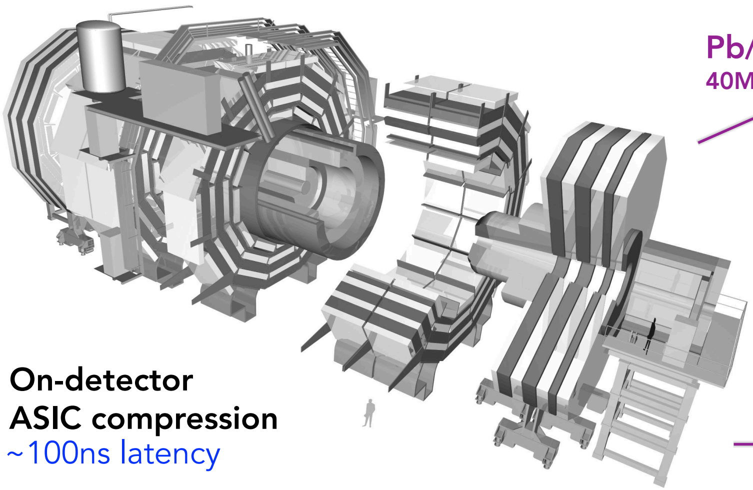
COMPARABLE TO GLOBAL INTERNET TRAFFIC BANDWIDTH



CMS Experiment

40MHz collision rate

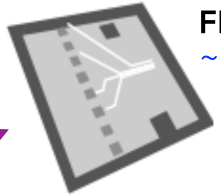
~1B detector channels



On-detector
ASIC compression

~100ns latency

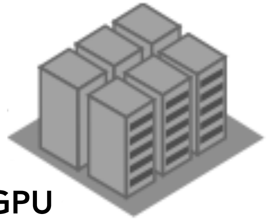
Pb/s
40MHz



FPGA filter stack

~us latency

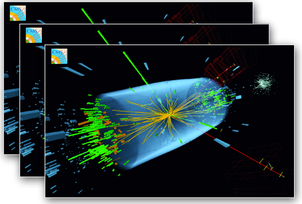
10s Tb/s
100s kHz



On-prem CPU/GPU
filter farm

~100 ms latency

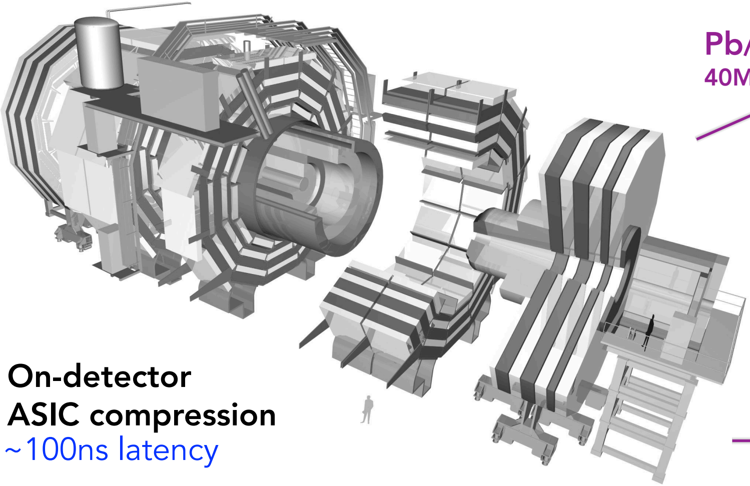
COMPARABLE TO GLOBAL INTERNET TRAFFIC BANDWIDTH



CMS Experiment

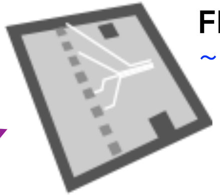
40MHz collision rate

~1B detector channels



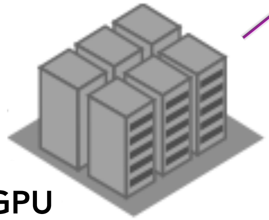
On-detector
ASIC compression
~100ns latency

Pb/s
40MHz



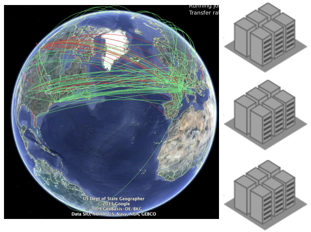
FPGA filter stack
~us latency

10s Tb/s
100s kHz



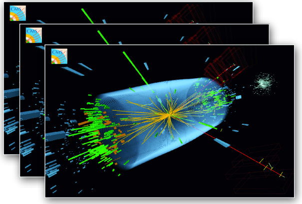
On-prem CPU/GPU
filter farm
~100 ms latency

10s Gb/s
~5 kHz



Worldwide
computing grid
Exabyte-scale
datasets

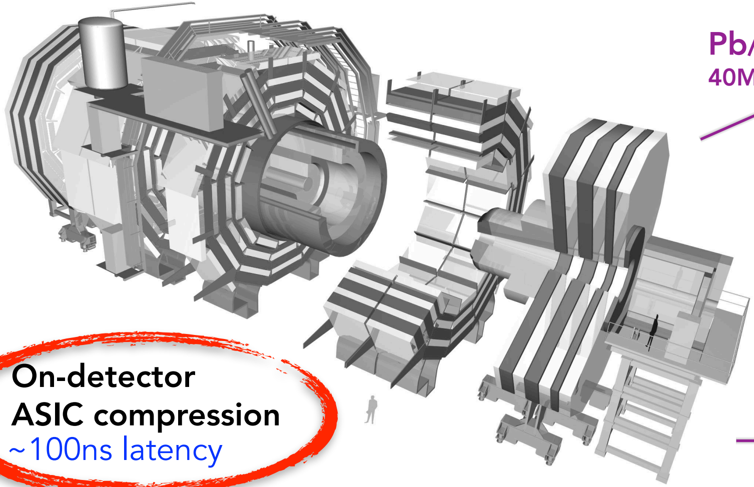
COMPARABLE TO GLOBAL INTERNET TRAFFIC BANDWIDTH



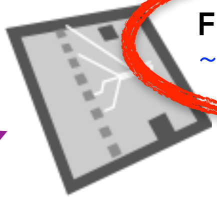
CMS Experiment

40MHz collision rate

~1B detector channels



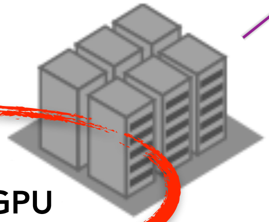
Pb/s
40MHz



FPGA filter stack
~us latency

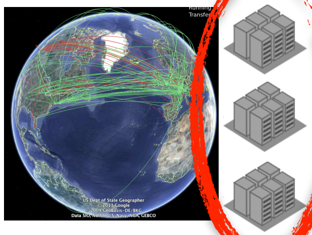
On-detector ASIC compression
~100ns latency

10s Tb/s
100s kHz



On-prem CPU/GPU filter farm
~100 ms latency

10s Gb/s
~5 kHz



Worldwide computing grid
Exabyte-scale datasets

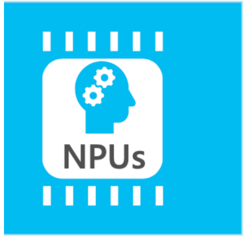
COMPARABLE TO GLOBAL INTERNET TRAFFIC BANDWIDTH

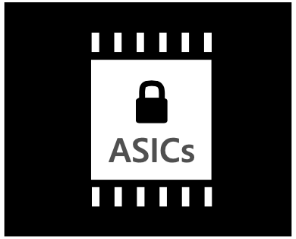
Types of compute

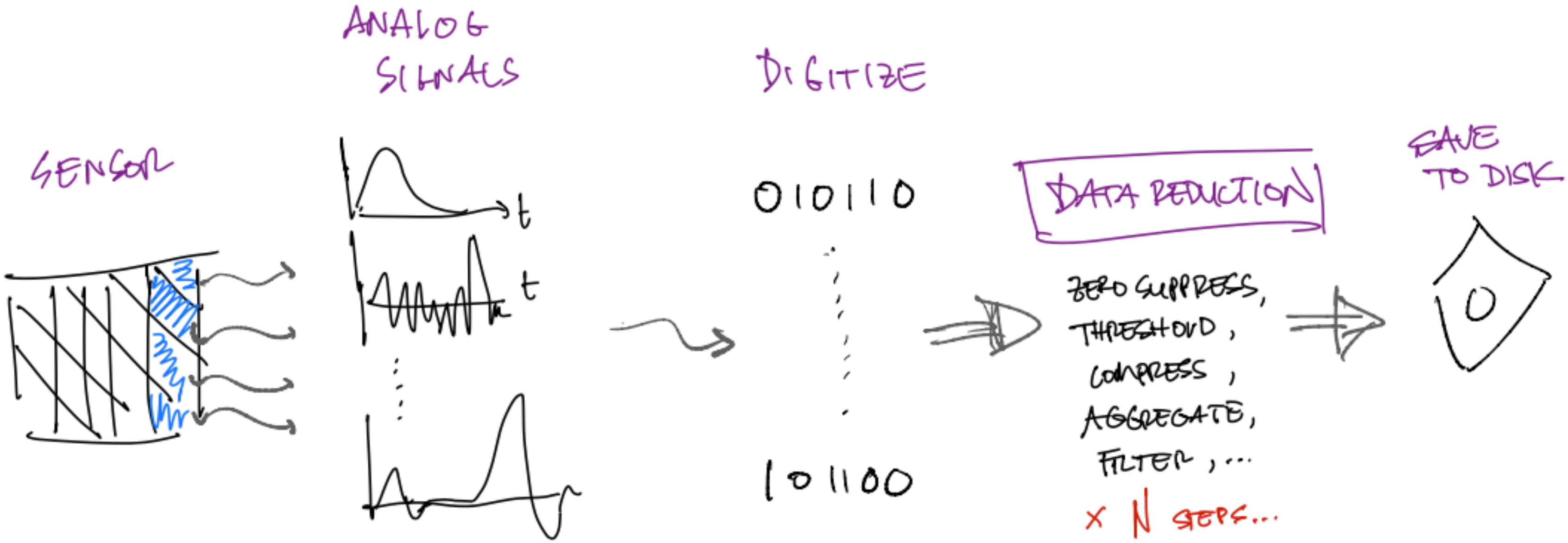
Types of compute

Credit: Dorothea vom Bruch

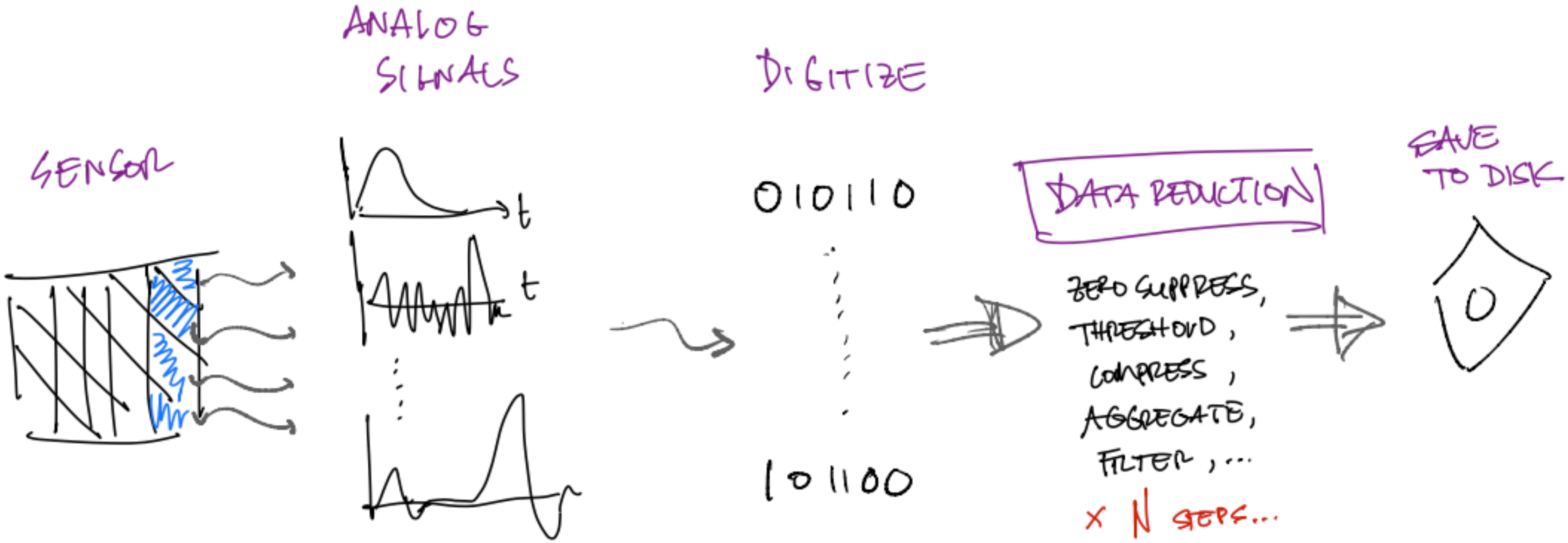
	CPU	GPU	FPGA
Latency	$\mathcal{O}$ (10) µs	$\mathcal{O}$ (100) µs	Deterministic, $\mathcal{O}$ (100) ns
I/O with processor	Ethernet, USB, PCIe	PCIe, Nvlink	Connectivity to any data source via printed circuit board (PCB)
Engineering cost	Low entry level (programmable with c++, python, etc.)	Low entry level (programmable with CUDA, OpenCL, etc.)	Some high-level syntax available, traditionally VHDL, Verilog (specialized engineer)
Single precision floating point performance	$\mathcal{O}$ (10) TFLOPs	$\mathcal{O}$ (10) TFLOPs	Optimized for fixed point performance
Serial / parallel	Optimized for serial performance, increasingly using vector processing	Optimized for parallel performance	Optimized for parallel performance



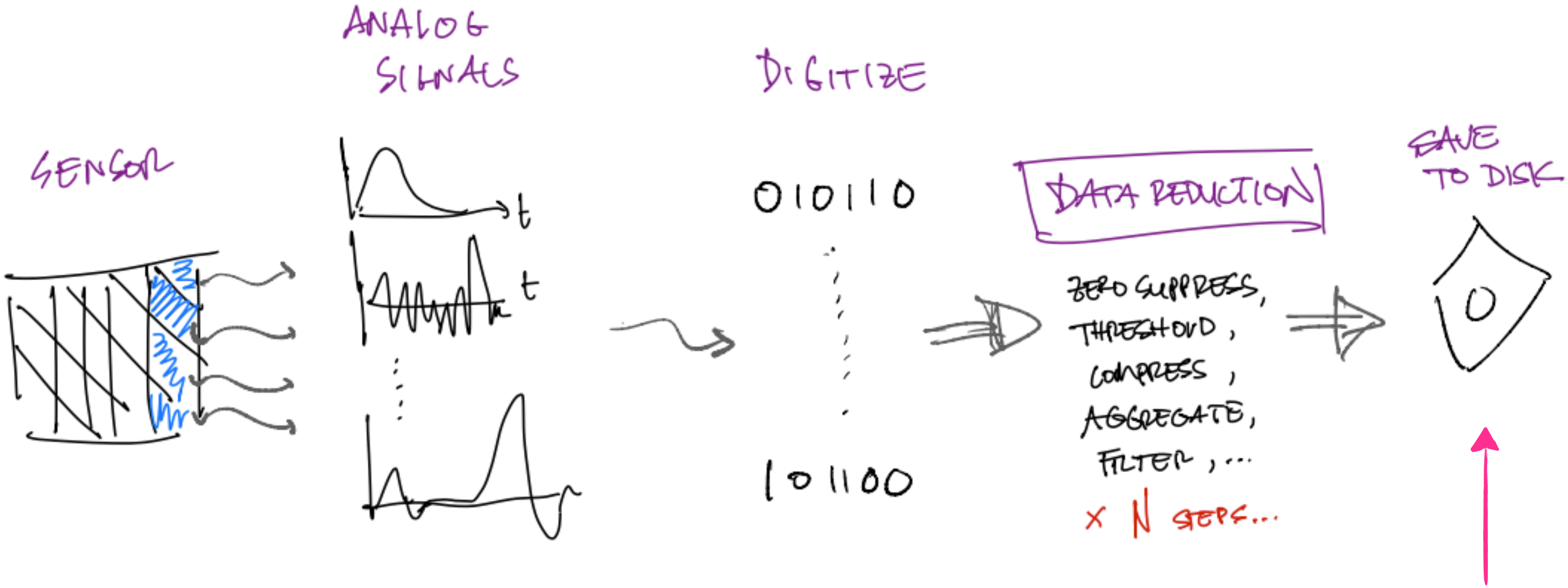




ANALOG SIGNALS
DIGITIZE
SENSOR
SAVE TO DISK
DATA REDUCTION
t
t
010110
101100
ZERO SUPPRESS,
THRESHOLD,
COMPRESS,
AGGREGATE,
FILTER, ...
× N STEPS...

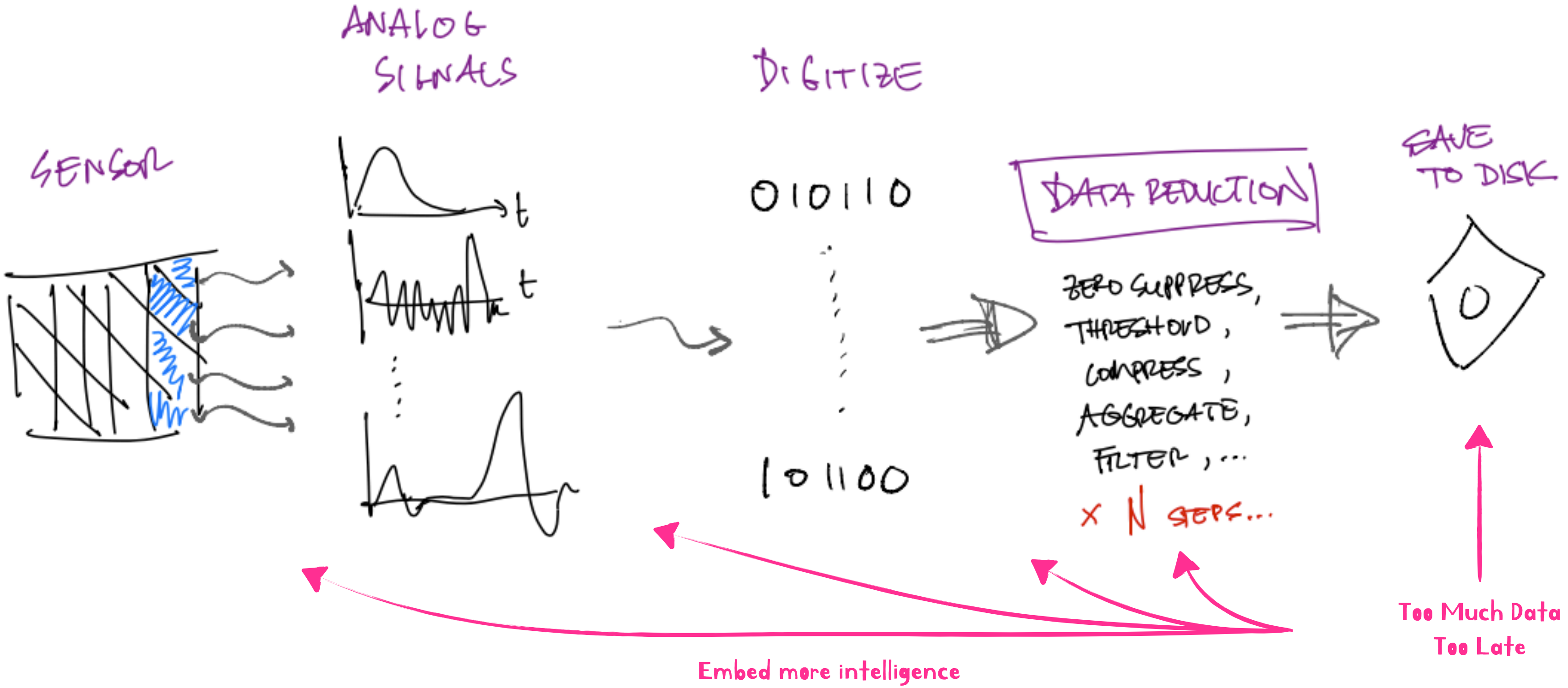


1 channel ~ 10b
1 channel, 1 MHz rate ~ 10 Mb/s
100k channels, 1 MHz rate ~ 1 Tb/s



1 channel ~ 10b
1 channel, 1 MHz rate ~ 10 Mb/s
100k channels, 1 MHz rate ~ 1 Tb/s

Too Much Data
Too Late



ANALOG SIGNALS
DIGITIZE
SENSOR
DATA REDUCTION
SAVE TO DISK
t
t
010110
101100
ZERO SUPPRESS,
THRESHOLD,
COMPRESS,
AGGREGATE,
FILTER, ...
x N STEPS...
Too Much Data
Too Late
Embed more intelligence

Fast ML for science and the extreme edge

"Scientific discoveries come from groundbreaking ideas and the capability to validate those ideas by testing nature at new scales - finer and more precise temporal and spatial resolution. This is leading to an explosion of data that must be interpreted, and ML is proving a powerful approach. The more efficiently we can test our hypotheses, the faster we can achieve discovery. To fully unleash the power of ML and accelerate discoveries, it is necessary to embed it into our scientific process, into our instruments and detectors."

Applications and Techniques for Fast Machine Learning in Science
https://doi.org/10.3389/fdata.2022.787421

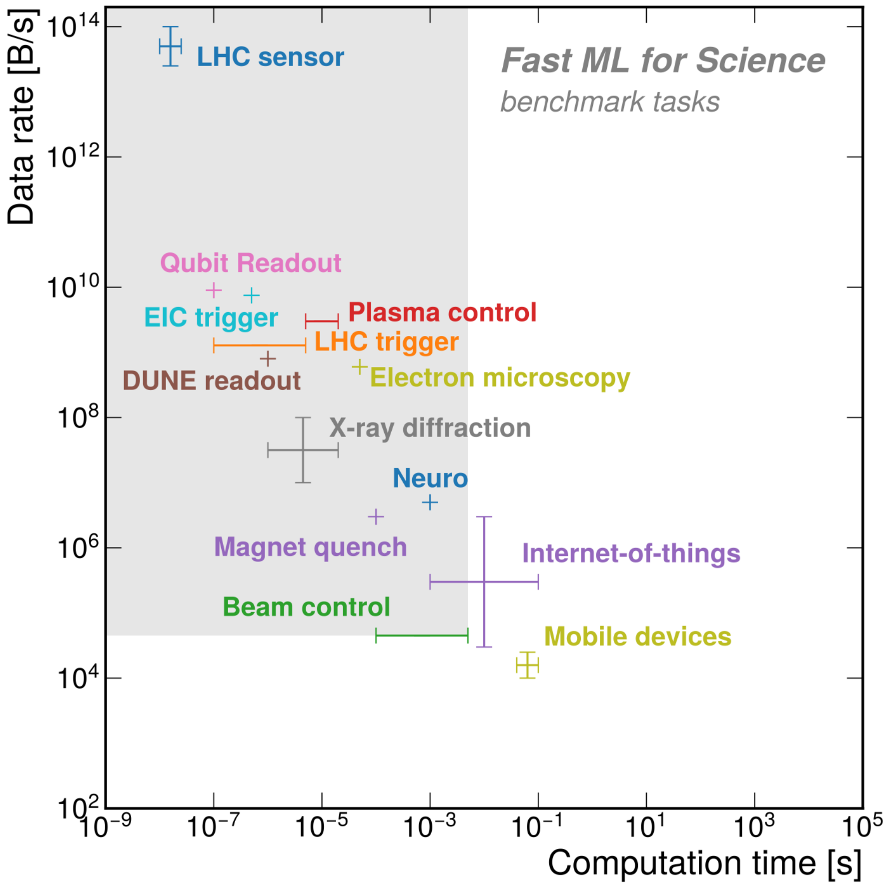
Why AI?

Universal function approximation - fit with customizable objective:
f(inputs; lots of parameters) = output

- Expressive & powerful: able to find patterns and correlations in high-dimensional data not explicitly accounted for; can unlock large gains in performance
- Adaptive & flexible: able to adapt to new data, conditions; handles all different types of data representations

Fast ML for Science

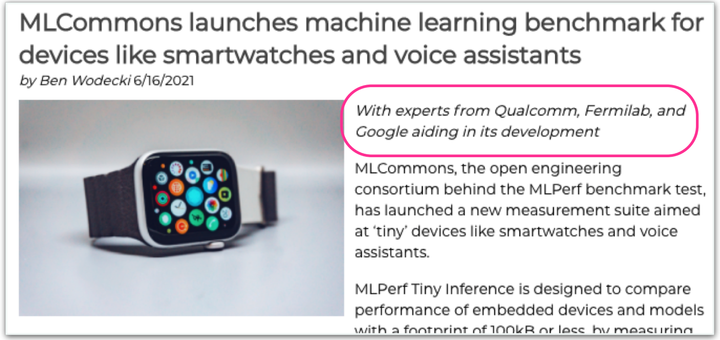
The Fast ML for Science community aims to bring **seemingly different domains** together to develop **techniques, tools, and platforms** for challenges that **far outpace industry.**



Fermilab

Fast ML for Science

The Fast ML for Science community aims to bring **seemingly different domains** together to develop **techniques, tools, and platforms** for challenges that **far outpace industry.**



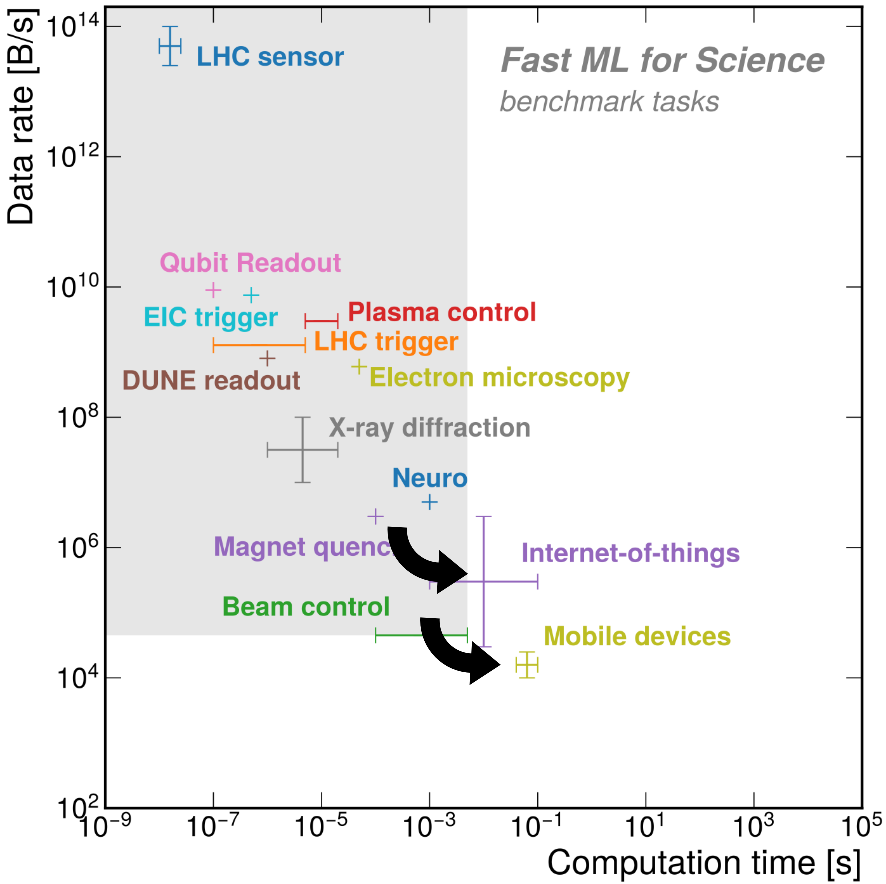
MLCommons launches machine learning benchmark for devices like smartwatches and voice assistants

by Ben Wodecki 6/16/2021

With experts from Qualcomm, Fermilab, and Google aiding in its development

MLCommons, the open engineering consortium behind the MLPerf benchmark test, has launched a new measurement suite aimed at 'tiny' devices like smartwatches and voice assistants.

MLPerf Tiny Inference is designed to compare performance of embedded devices and models with a footprint of 100kB or less, by measuring



Fast and slow control

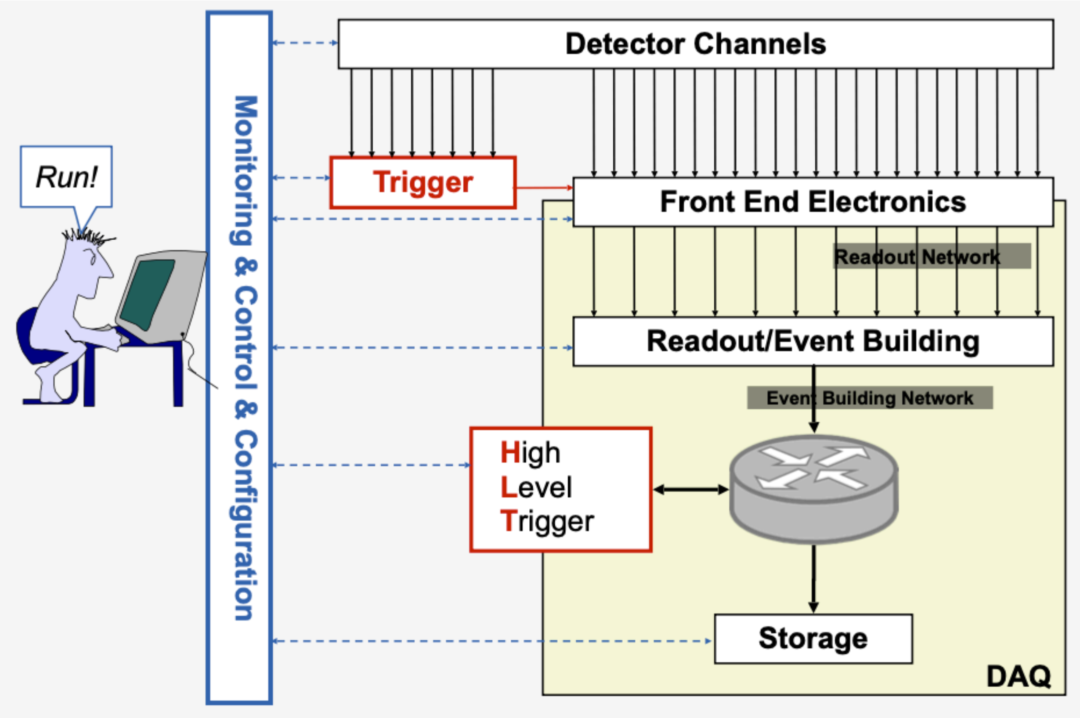
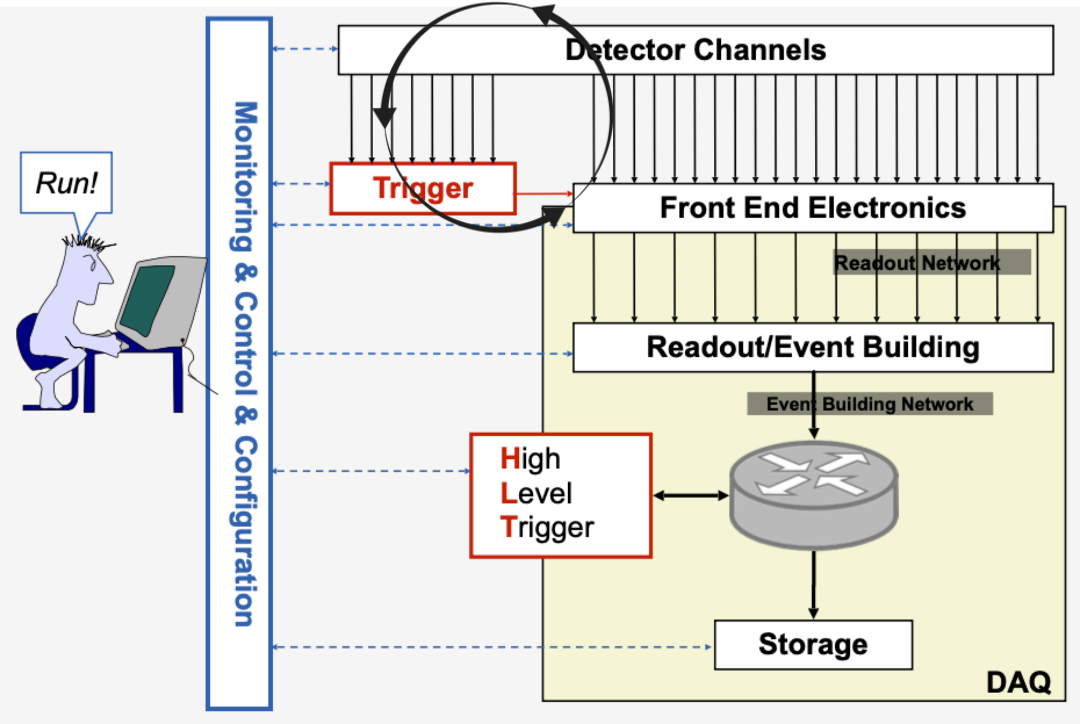


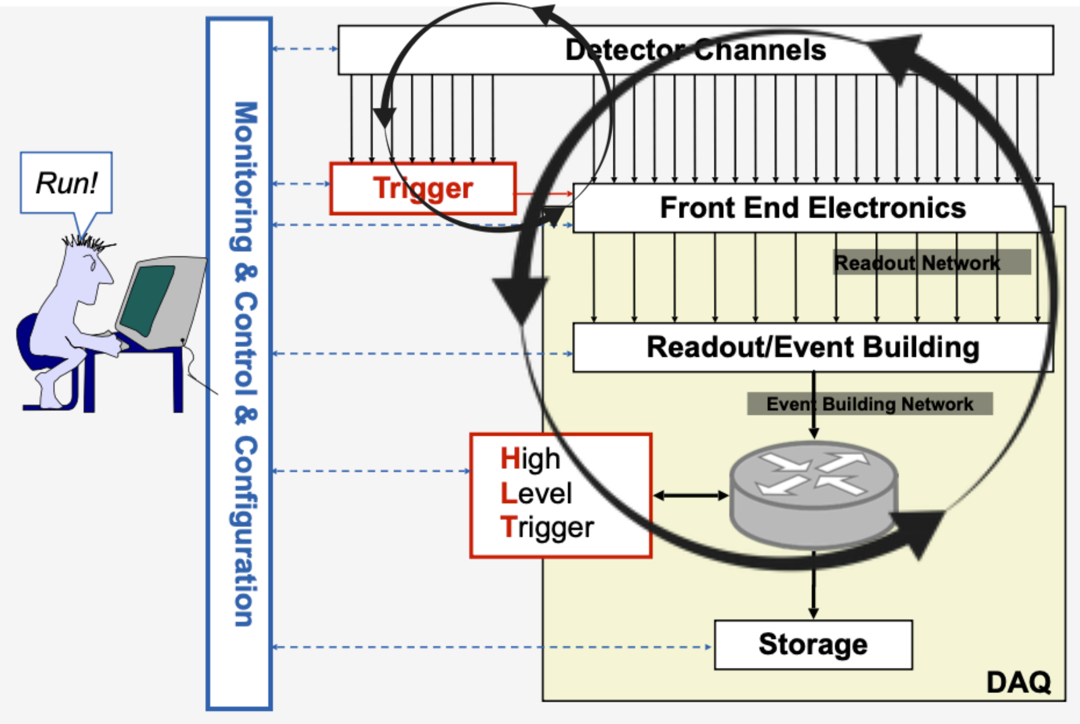
Image credit: A. Thea

Fast and slow control



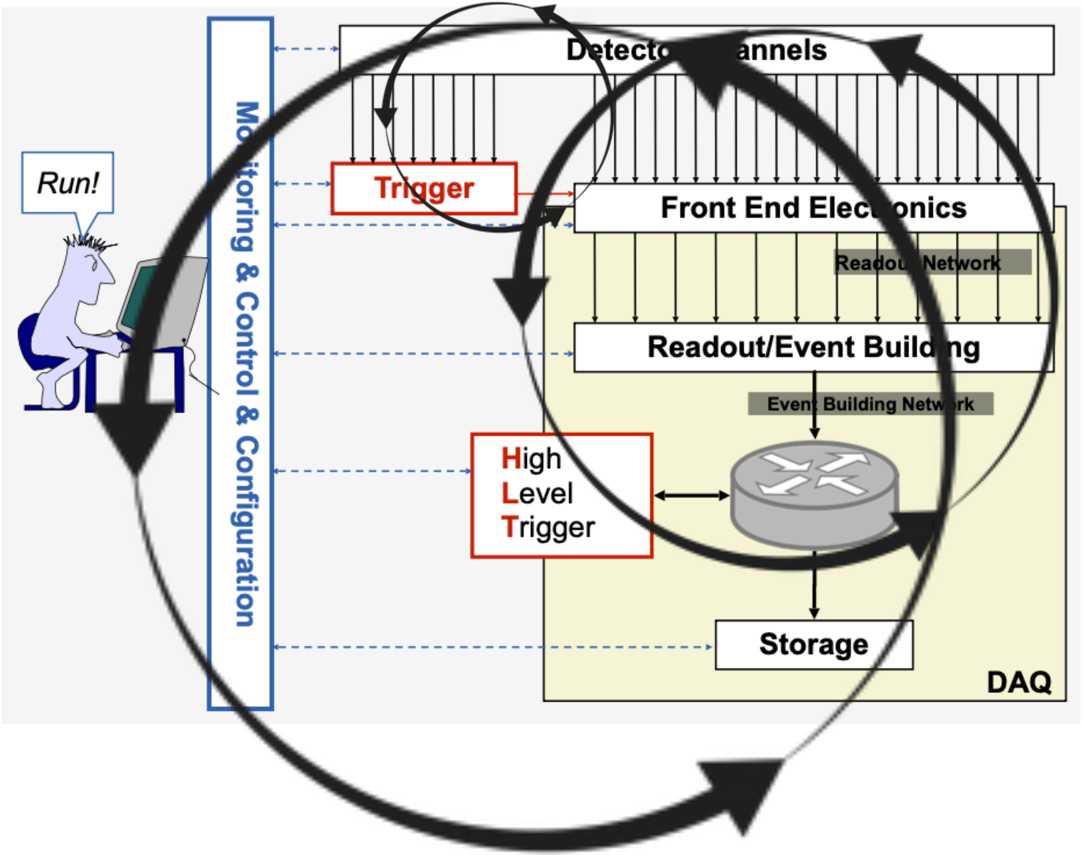
Fast and slow control

Image credit: A. Thea

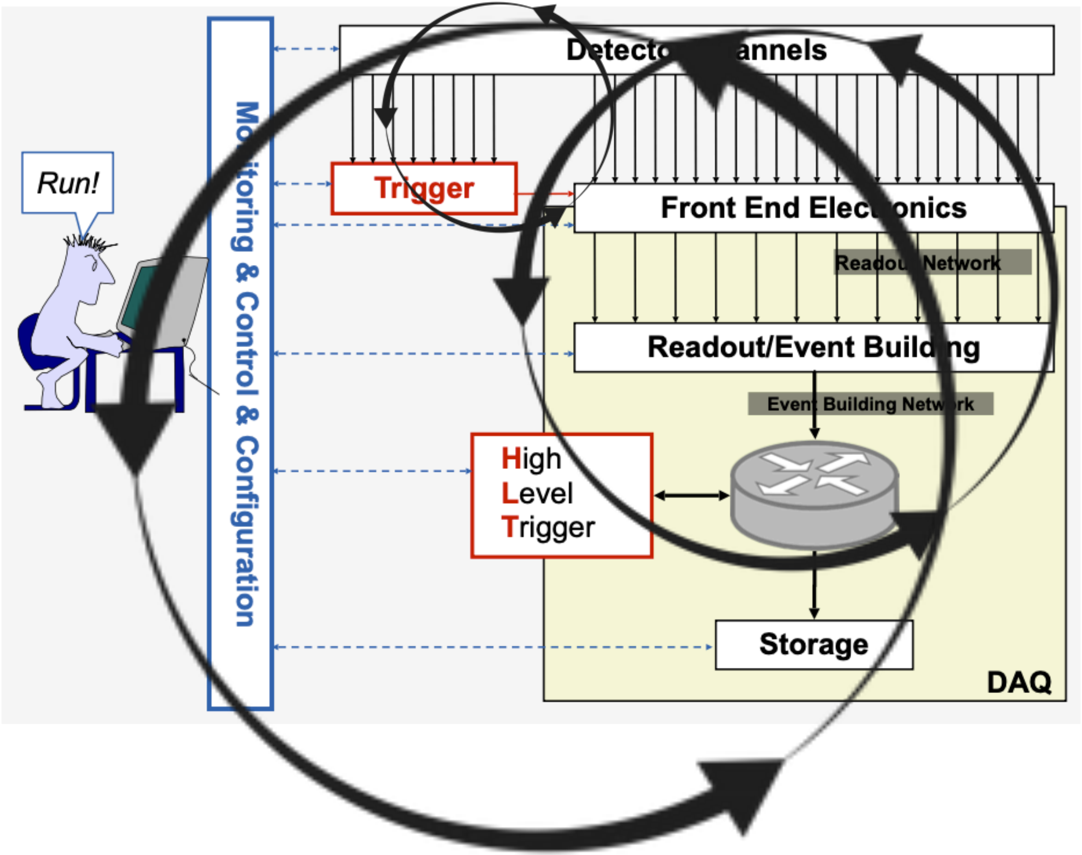


Fast and slow control

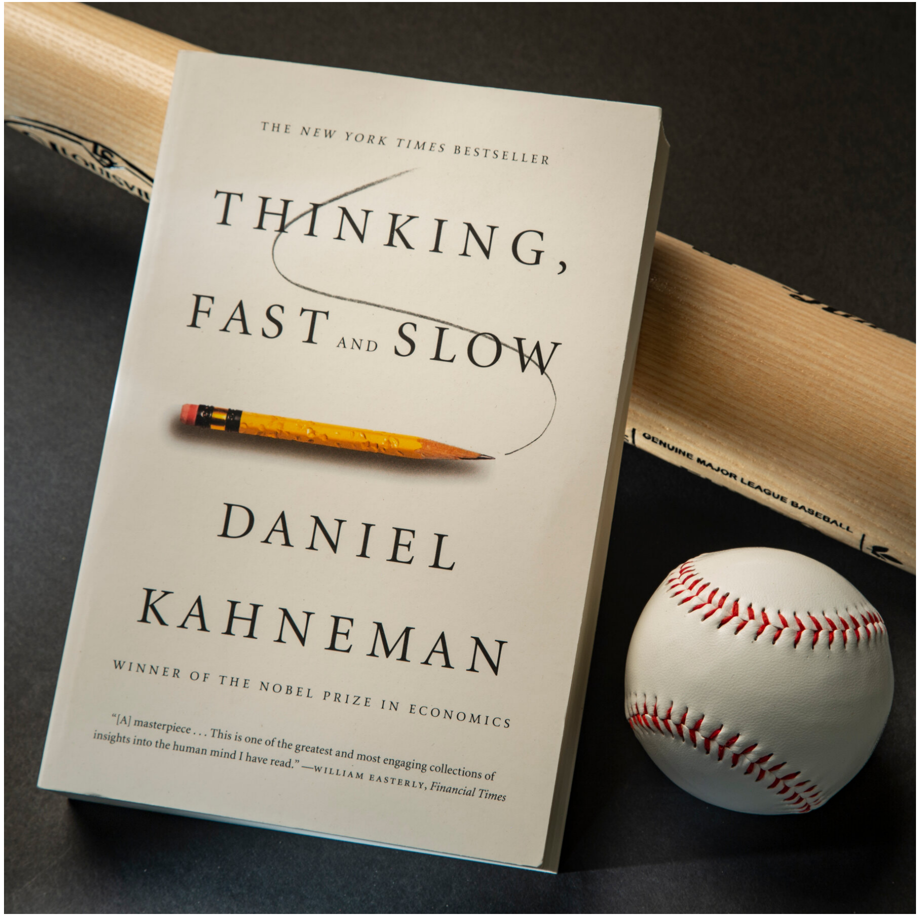
Image credit: A. Thea



Fast and slow control

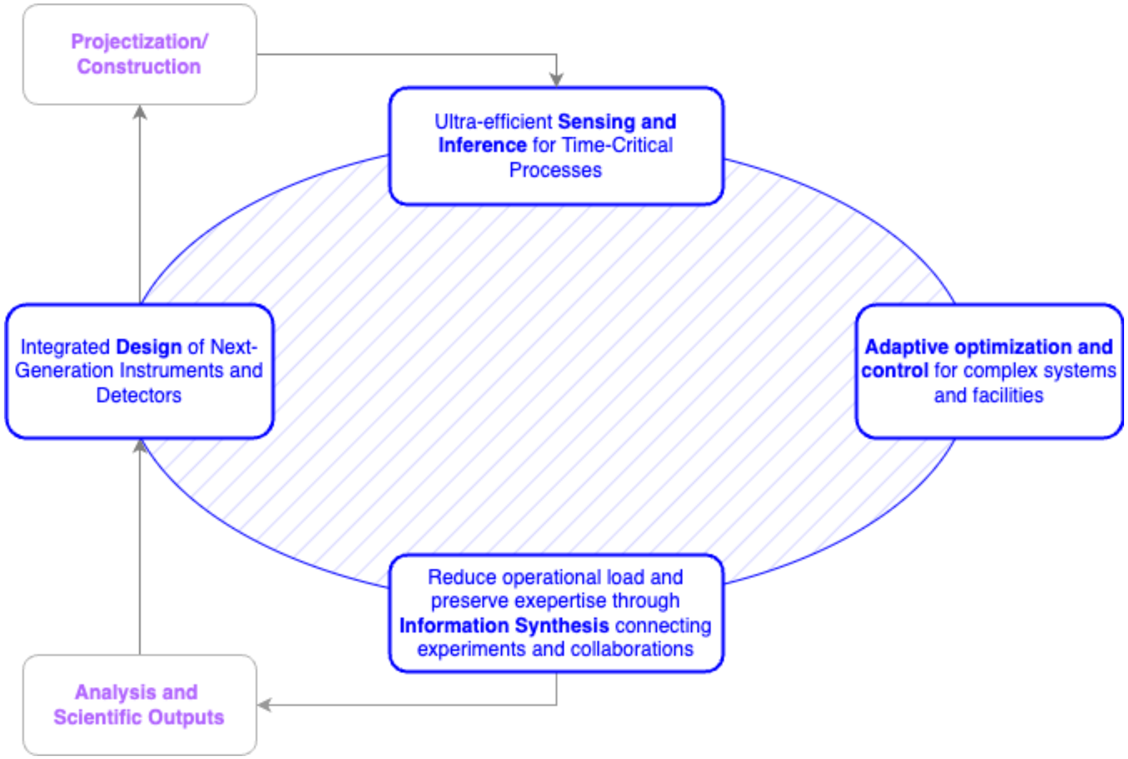


- Fast control
  - Immediate response to dynamics of the experiment and data readout
  - Event timing, triggering, etc.
- Slow control
  - Detector stability over minutes, days, weeks, months,…
  - Monitoring and controlling operational parameters: electronics gains, pedestals, calibrations, etc.

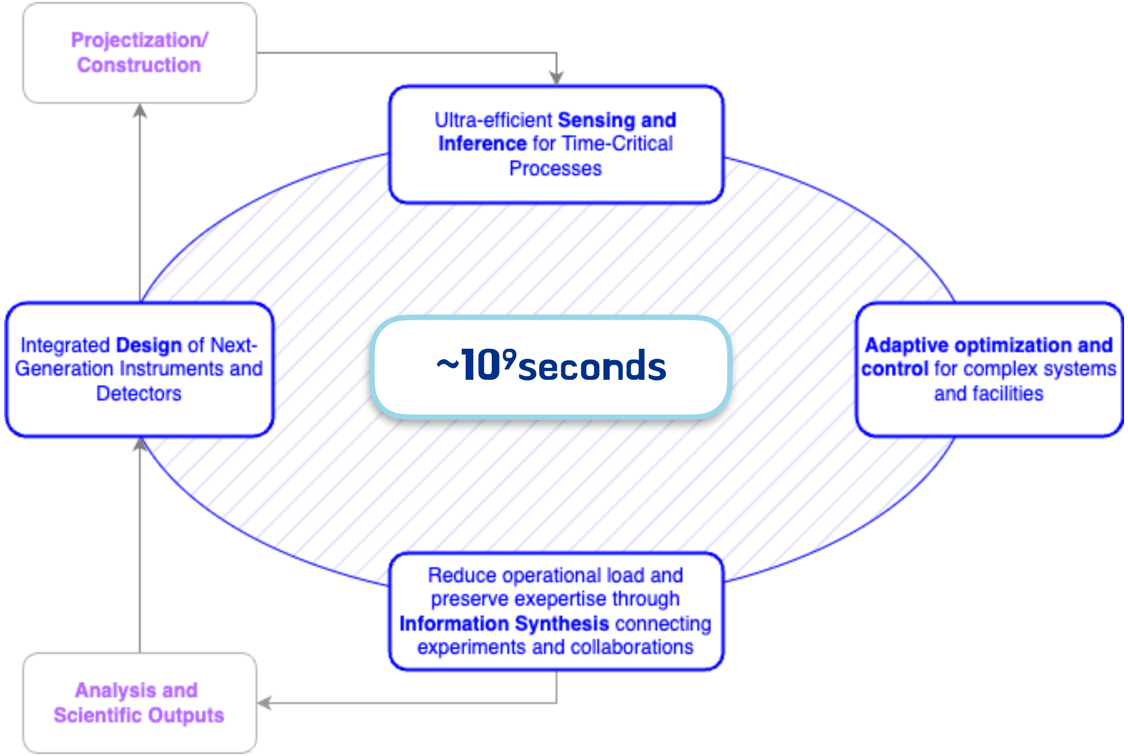


…on reducing biases in our "fast" decision-making selves

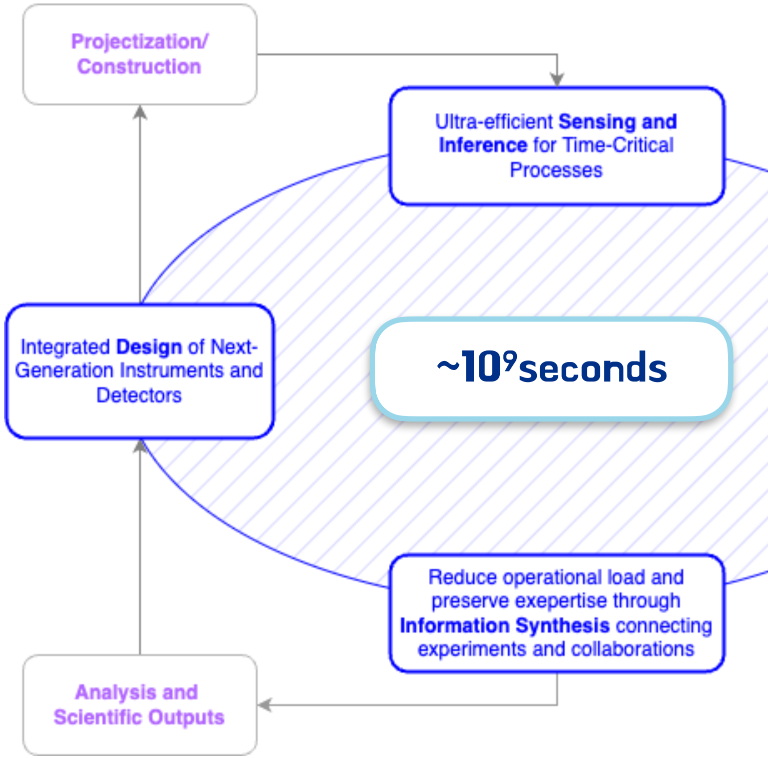
Life cycle of instruments, detectors, facilities

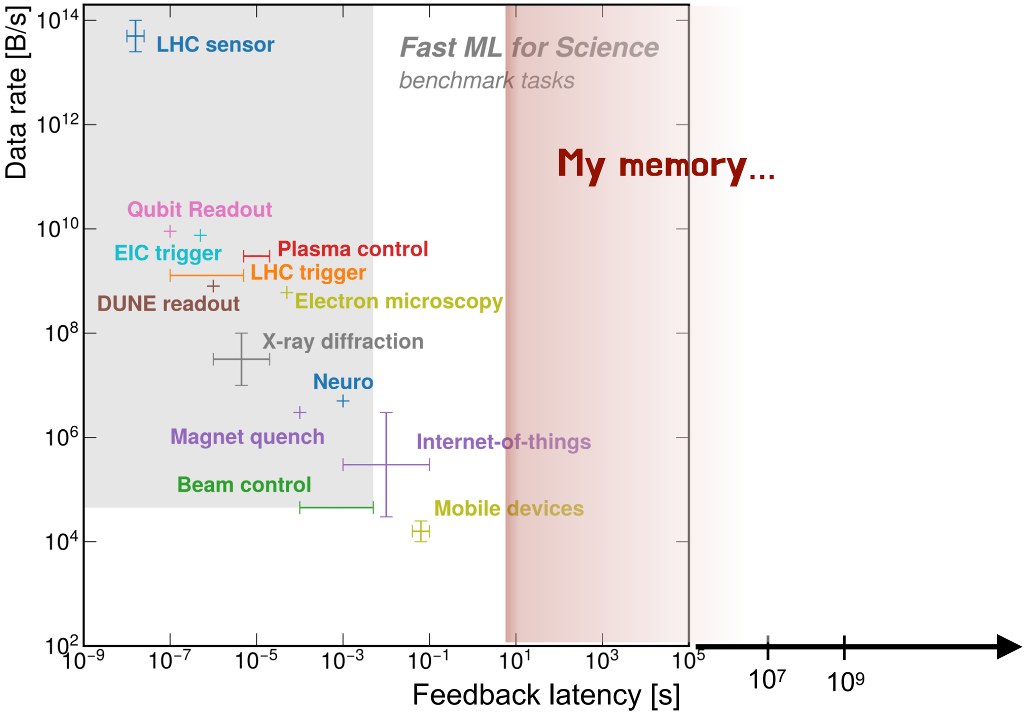


Life cycle of instruments, detectors, facilities



Life cycle of instruments, detectors, facilities



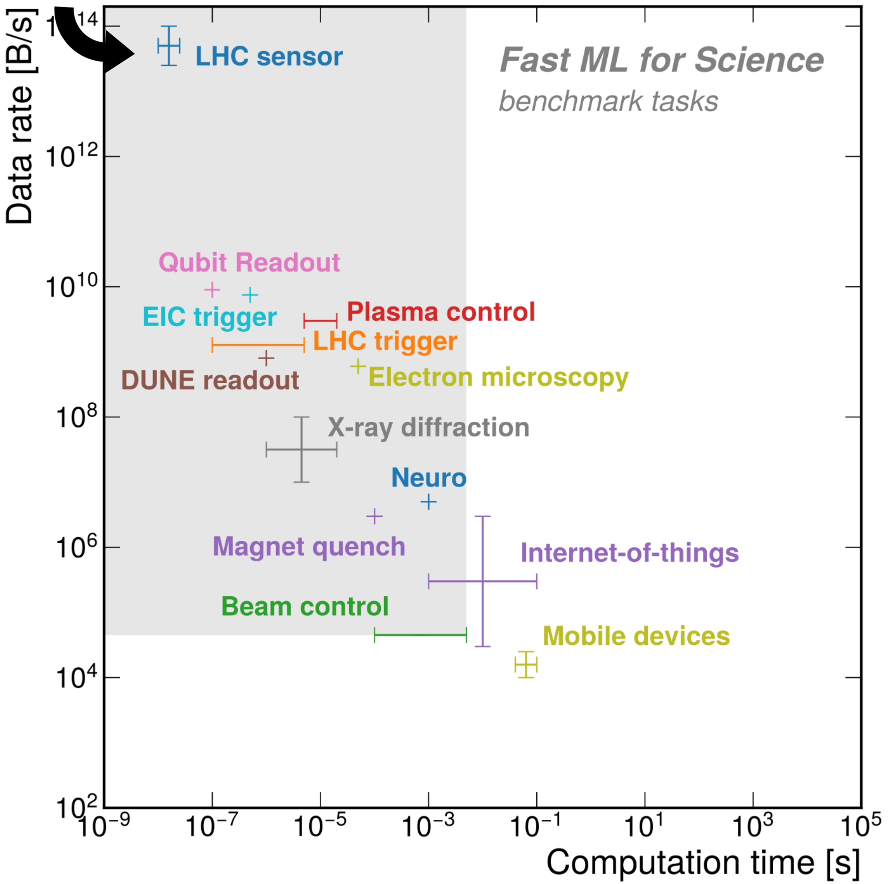


Fermilab

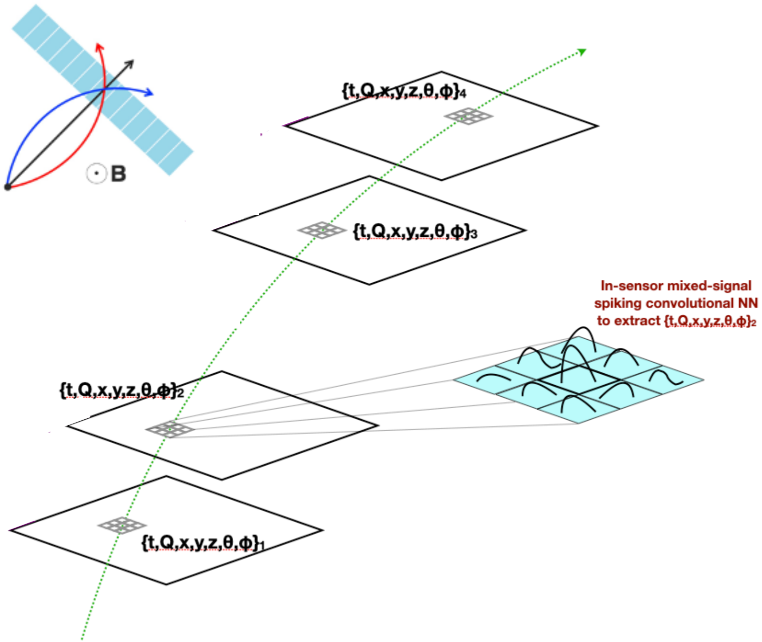
Outline

- Fast and Slow
- **Fast ML and hardware codesign**
- Fast and Slow ML together
- Slow ML & Real-Time

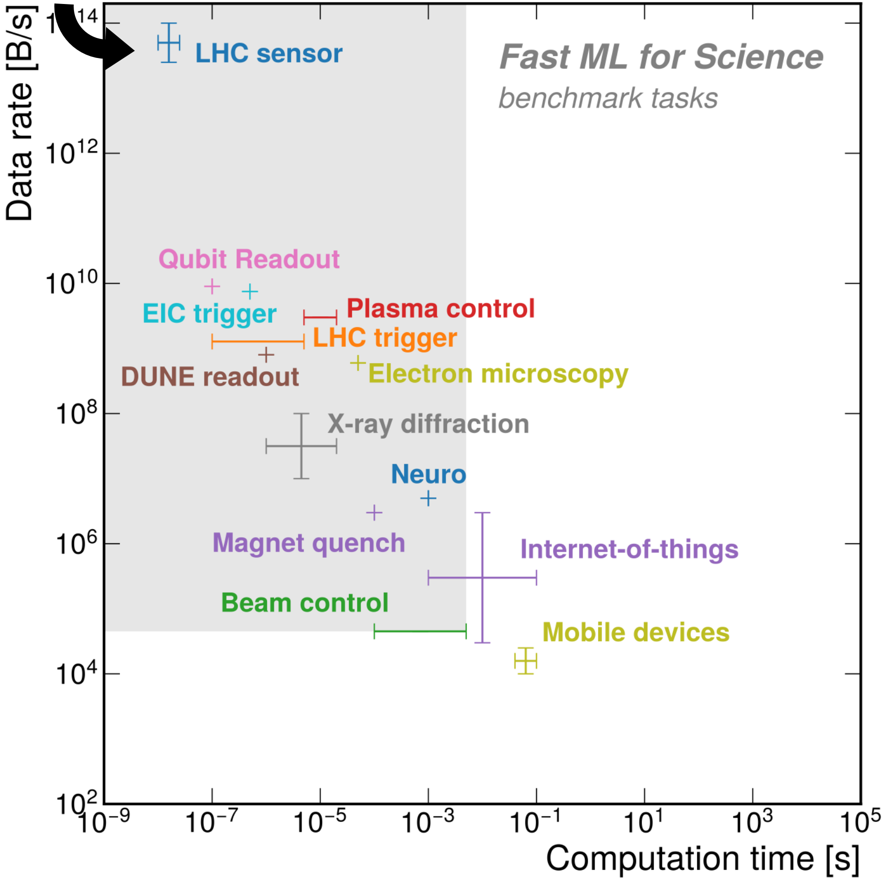
Fast ML example applications



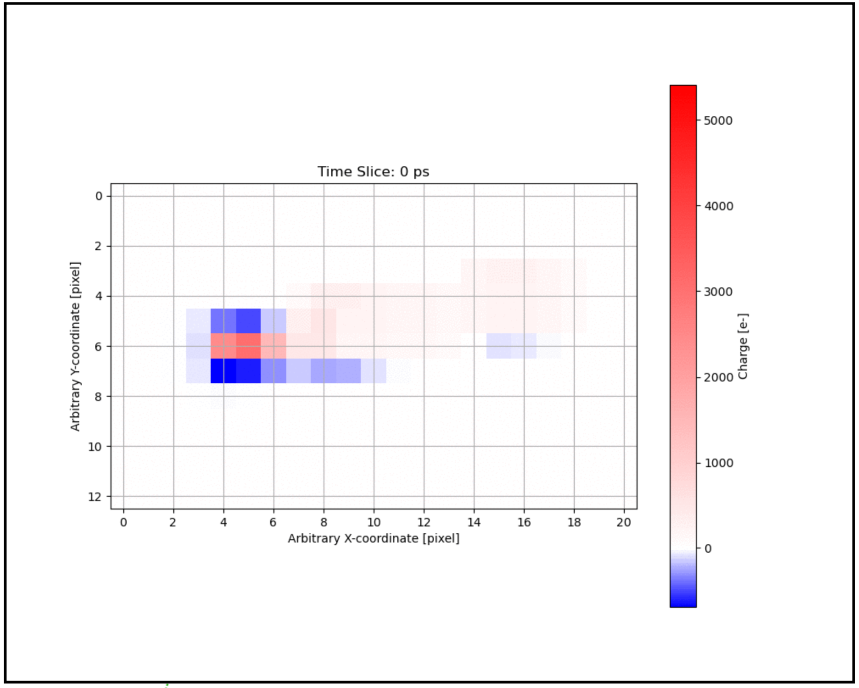
Fast ML example applications



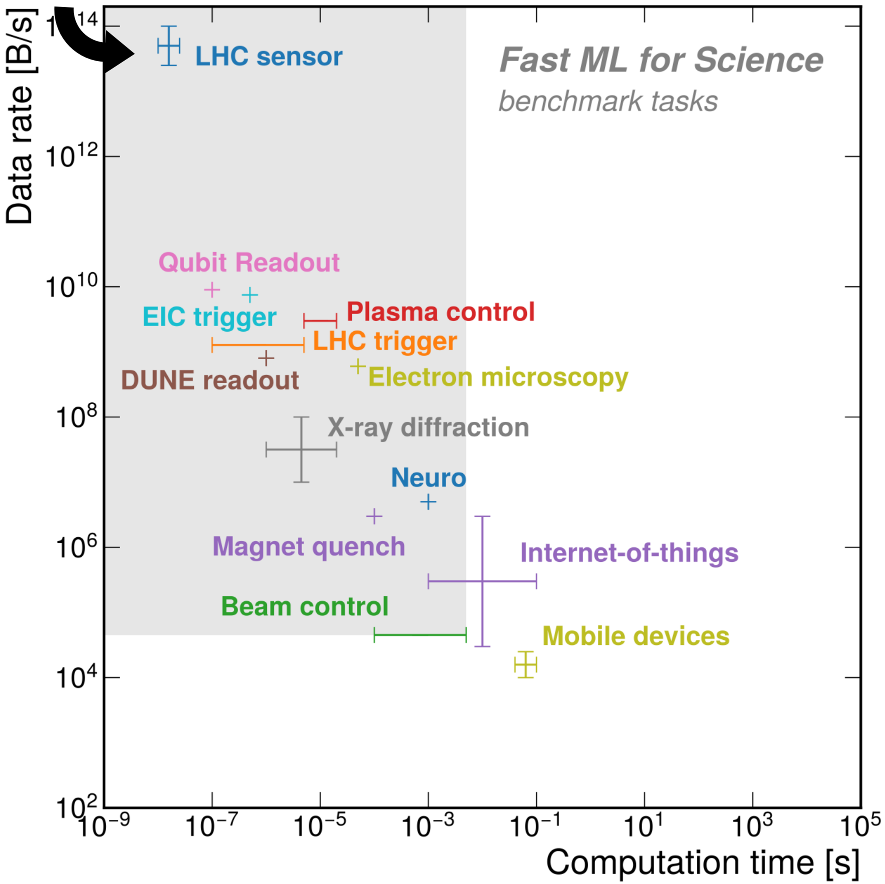
Pixel information in the trigger brings fundamentally new capabilities to LHC experiments - but rates are massive!



Fast ML example applications



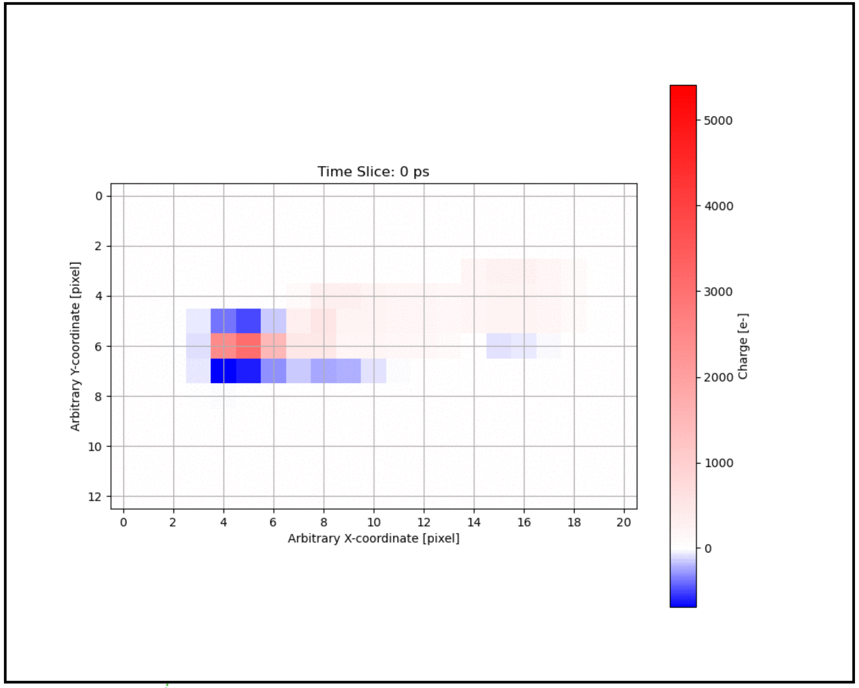
Pixel information in the trigger brings fundamentally new capabilities to LHC experiments - but rates are massive!



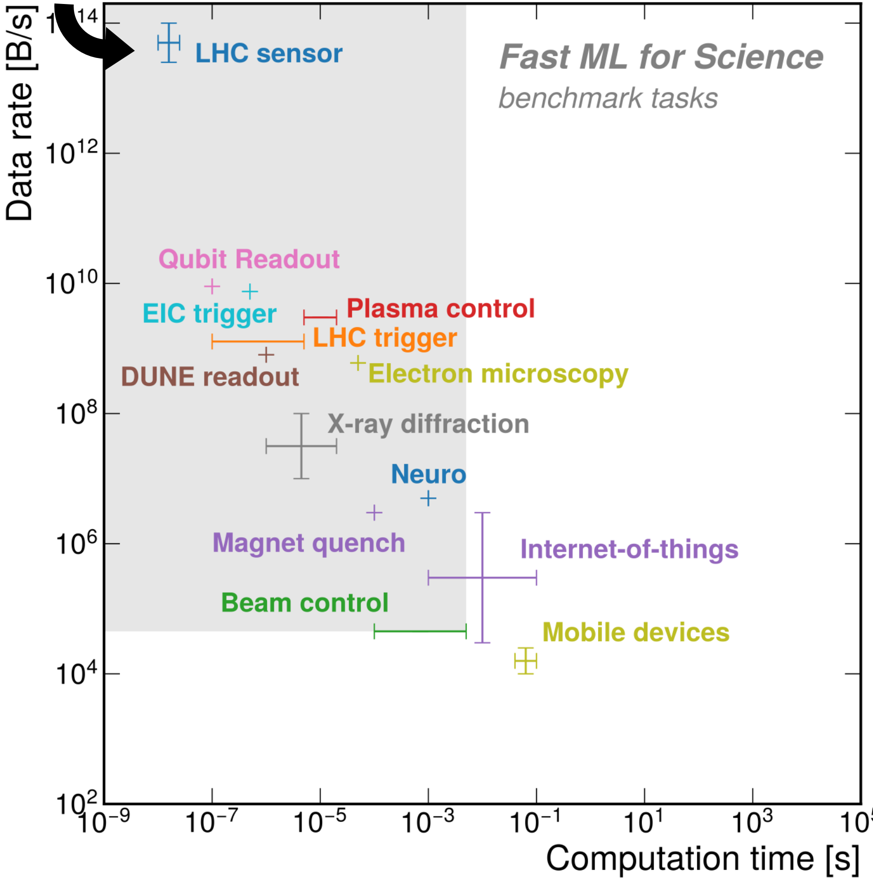
 https://fastmachinelearning.org/smart-pixels/

Fermilab

Fast ML example applications



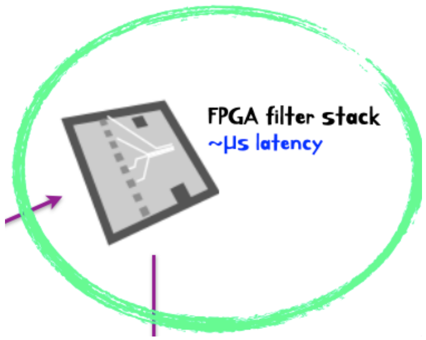
Pixel information in the trigger brings fundamentally new capabilities to LHC experiments – but rates are massive!



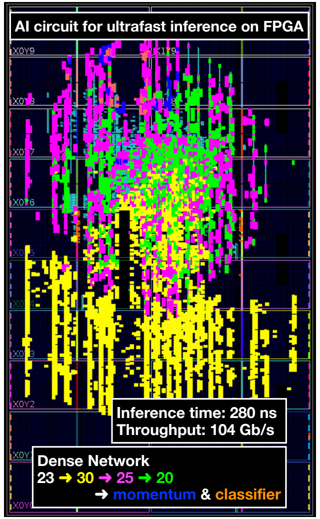
https://fastmachinelearning.org/smart-pixels/

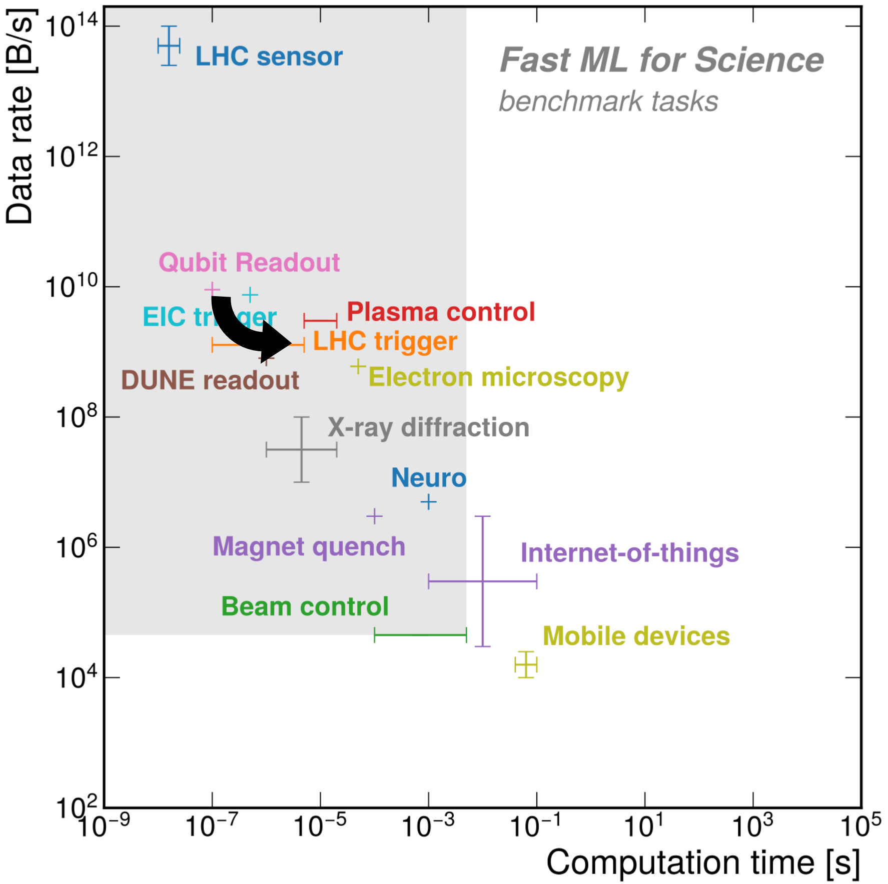
Fermilab

Fast ML example applications

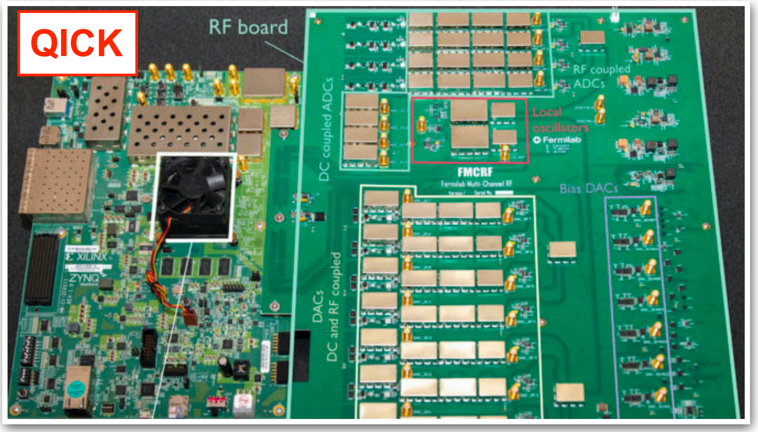


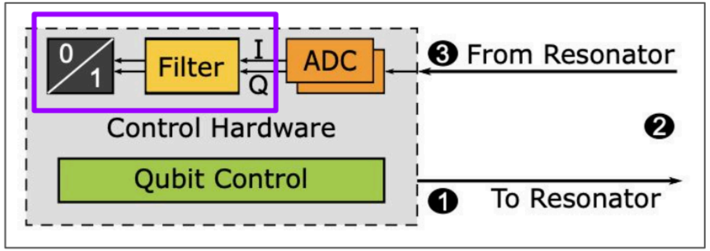
How can we reduce biases in trigger selections?



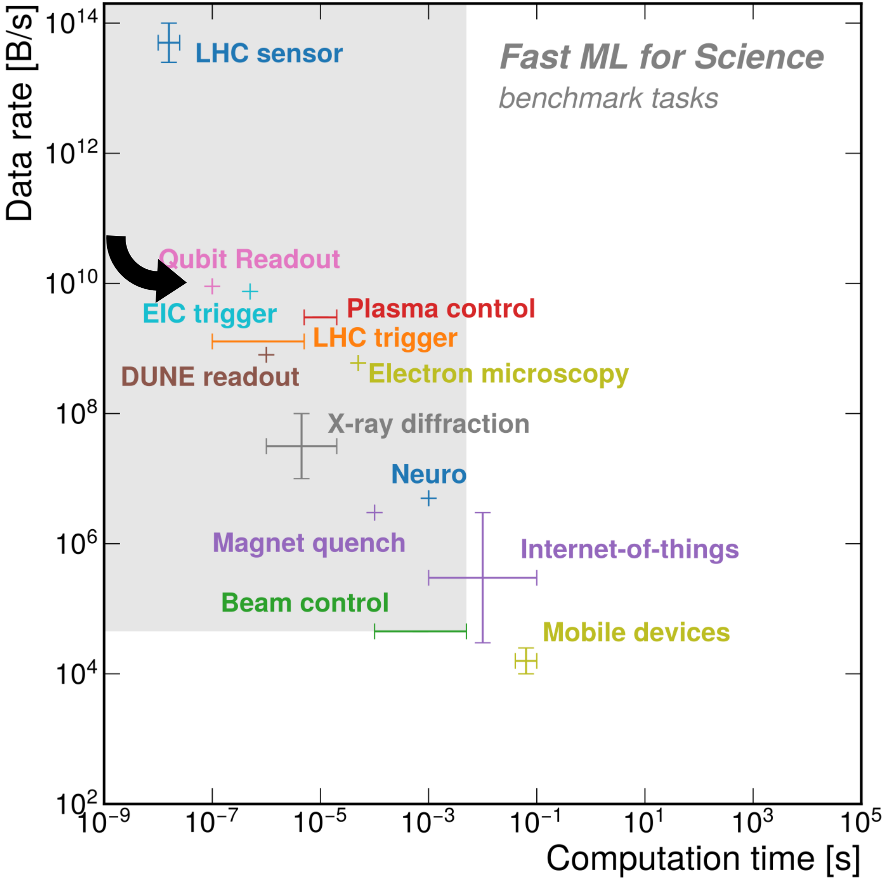


Fast ML science applications



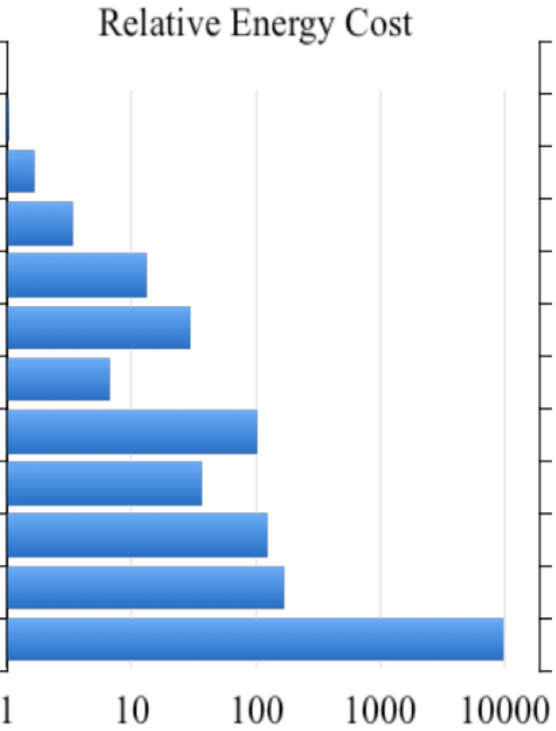


Dynamic and noisy systems with multiple timescales for changing conditions…



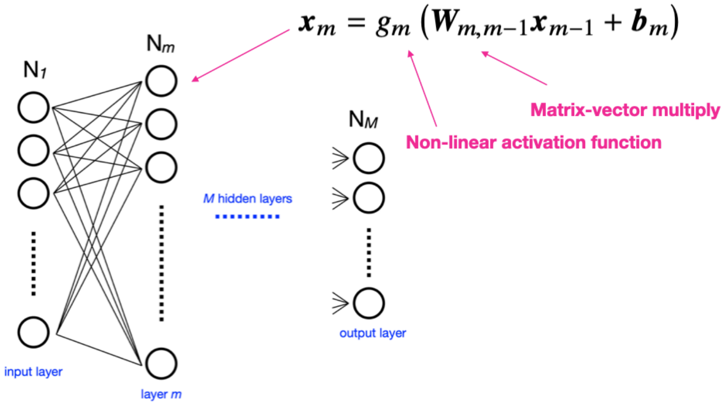
Moving data expensive, computing cheap

Operation:	Energy (pJ)
8b Add	0.03
16b Add	0.05
32b Add	0.1
16b FP Add	0.4
32b FP Add	0.9
8b Mult	0.2
32b Mult	3.1
16b FP Mult	1.1
32b FP Mult	3.7
32b SRAM Read (8KB)	5
32b DRAM Read	640



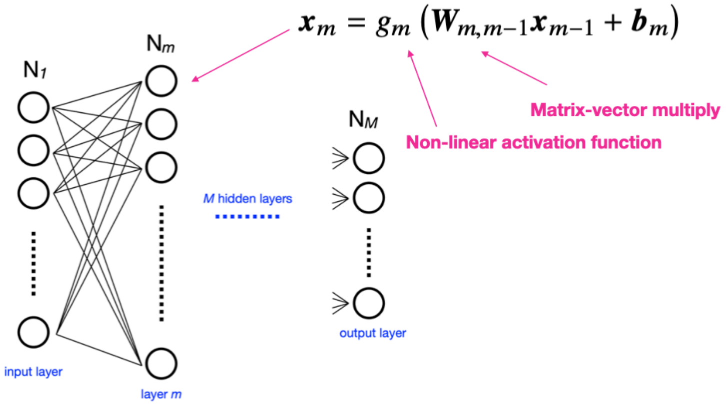
Adapted from Horowitz

Efficient machine learning



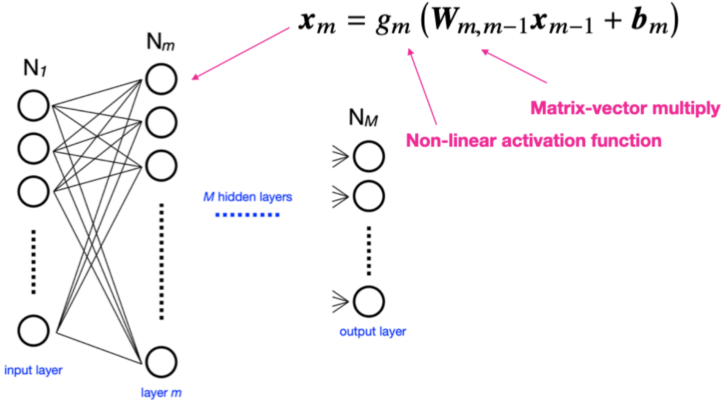
Efficient machine learning

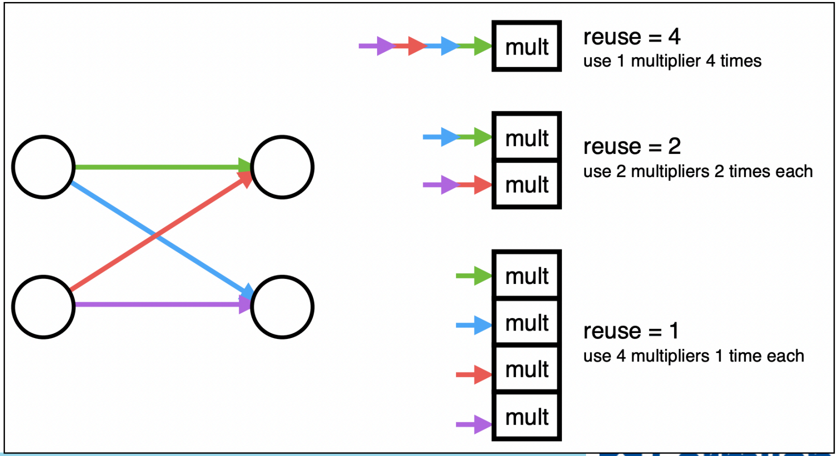
- Computation parallelization/vectorization and in-memory compute (architecture)
- Quantization, reduced precision
  - For ML, 32-bit floating point is often overkill
  - Integer/fixed-point math at 16,8,7,6,5…1 bits
- Compression, pruning
  - maintain the same performance while removing low weight synapses and neurons



Efficient machine learning

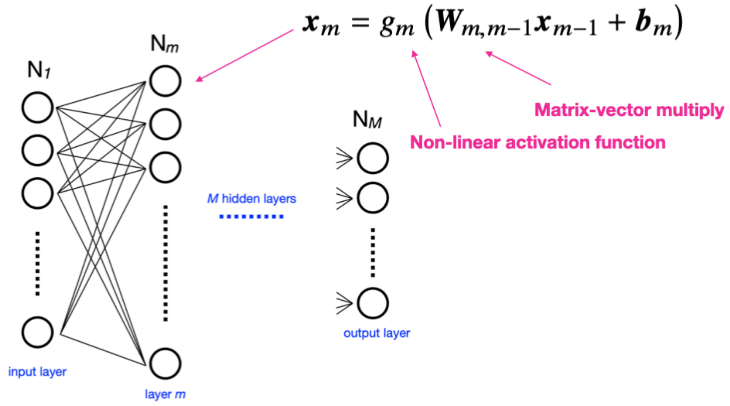
- Computation parallelization/vectorization and in-memory compute (architecture)
- Quantization, reduced precision
  - For ML, 32-bit floating point is often overkill
  - Integer/fixed-point math at 16,8,7,6,5…1 bits
- Compression, pruning
  - maintain the same performance while removing low weight synapses and neurons

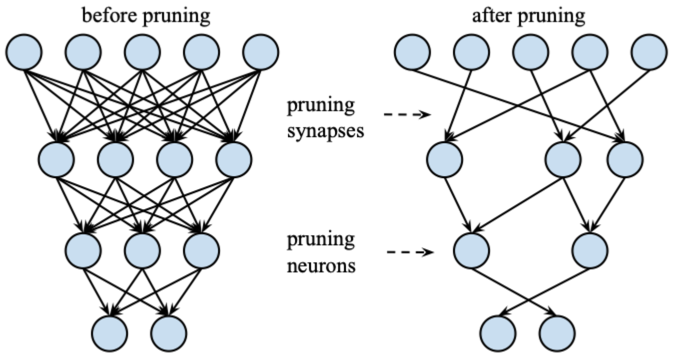




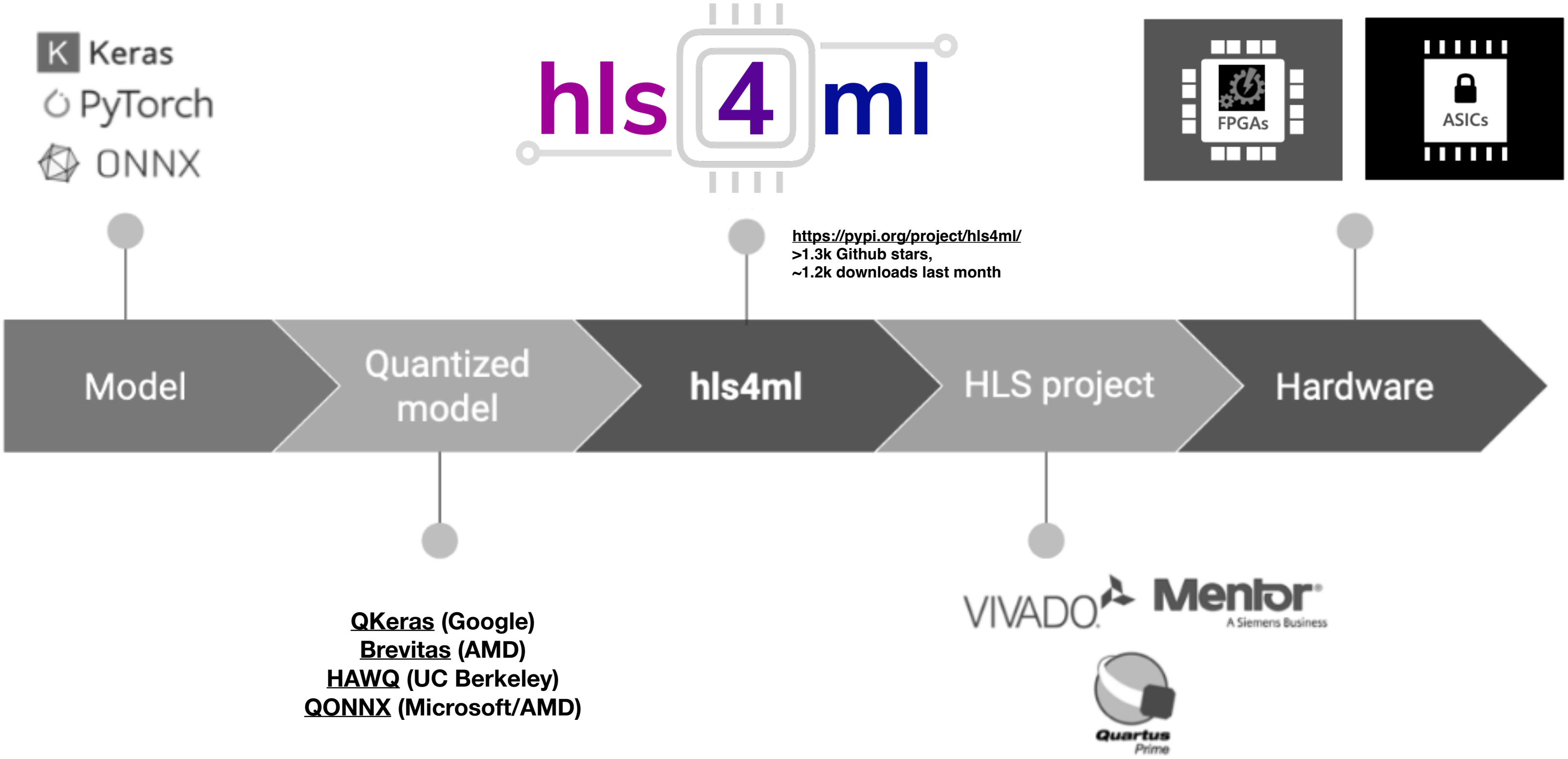
Efficient machine learning

- Computation parallelization/vectorization and in-memory compute (architecture)
- Quantization, reduced precision
  - For ML, 32-bit floating point is often overkill
  - Integer/fixed-point math at 16,8,7,6,5...1 bits
- Compression, pruning
  - maintain the same performance while removing low weight synapses and neurons

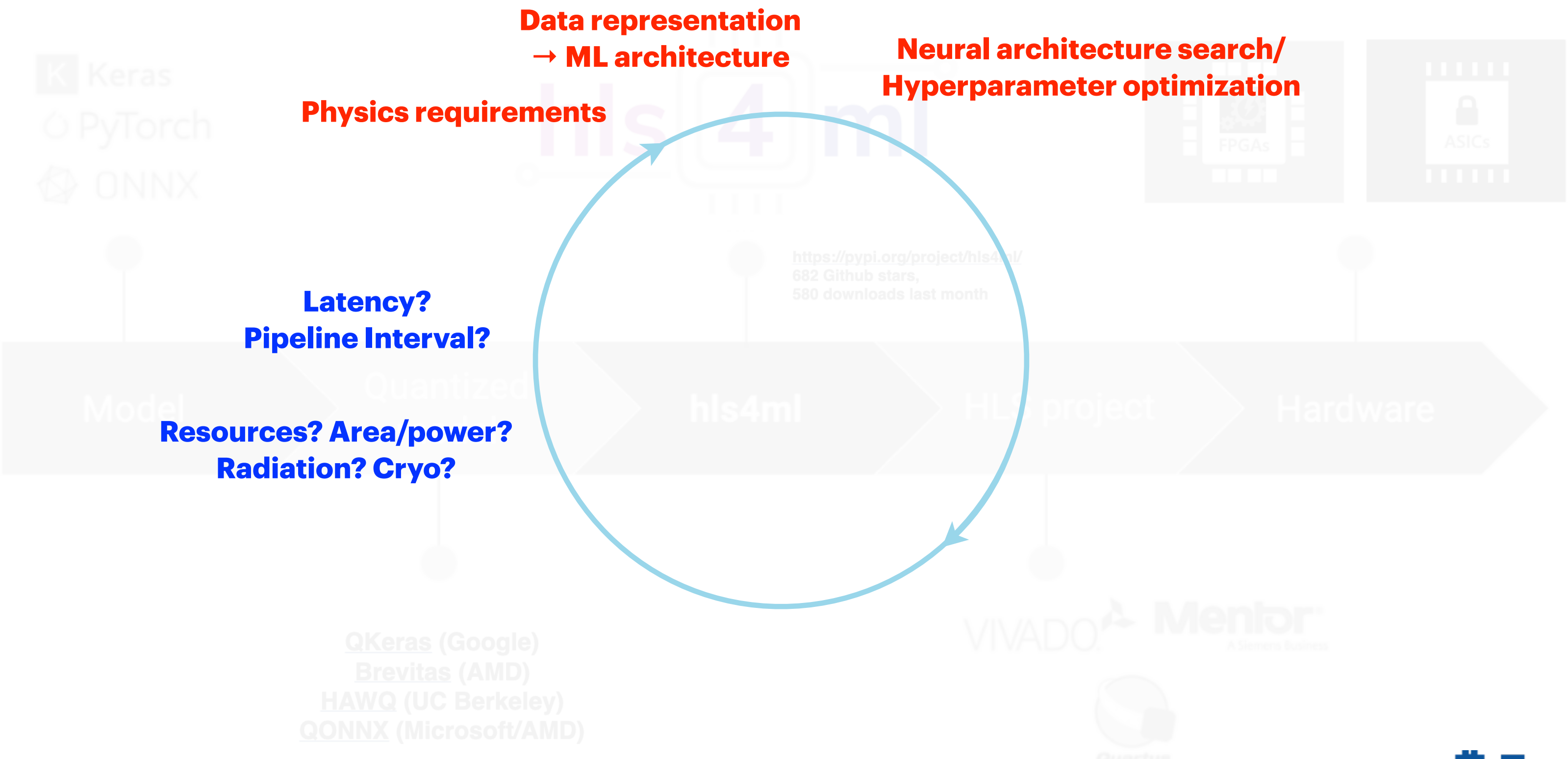




Hardware - algorithm codesign



Hardware - algorithm codesign



Hardware - algorithm codesign

Data representation
→ ML architecture

Neural architecture search/
Hyperparameter optimization

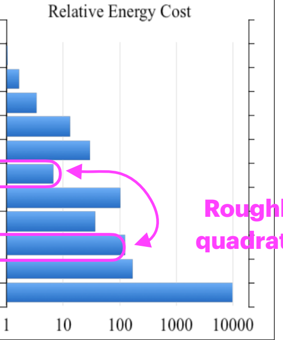
Physics requirements

Quantize network

Latency?
Pipeline Interval?

Resources? Area/power?
Radiation? Cryo?

Operation:	Energy (pJ)
8b Add	0.03
16b Add	0.05
32b Add	0.1
16b FP Add	0.4
32b FP Add	0.9
8b Mult	0.2
32b Mult	3.1
16b FP Mult	1.1
32b FP Mult	3.7
32b SRAM Read (8KB)	5
32b DRAM Read	640



Adapted from Horowitz

Roughly
quadratic

Fermilab

Hardware - algorithm codesign

Data representation
→ ML architecture

Neural architecture search/
Hyperparameter optimization

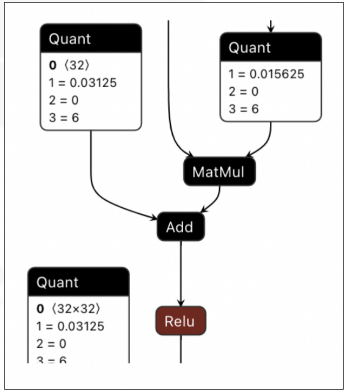
Physics requirements

Quantize network

Latency?
Pipeline Interval?

Intermediate (quantized)
representations

Resources? Area/power?
Radiation? Cryo?



Fermilab

Hardware - algorithm codesign

Data representation
→ ML architecture

Physics requirements

Neural architecture search/
Hyperparameter optimization

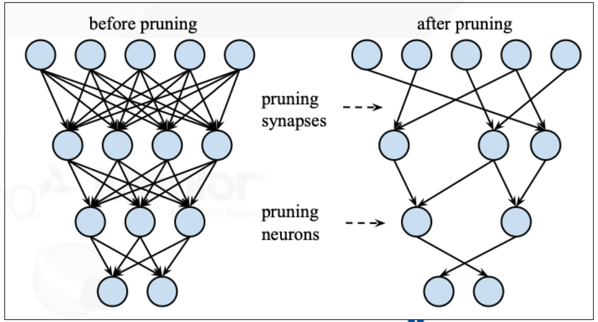
Quantize network

Intermediate (quantized)
representations

Pruning/sparsity?

Latency?
Pipeline Interval?

Resources? Area/power?
Radiation? Cryo?



Fermilab

Hardware - algorithm codesign

Data representation
→ ML architecture

Neural architecture search/
Hyperparameter optimization

Physics requirements

Quantize network

Intermediate (quantized)
representations

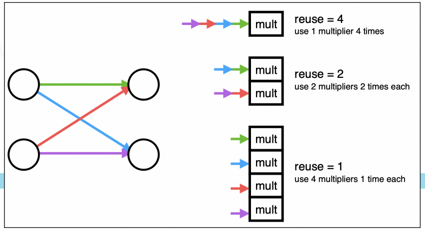
Latency?
Pipeline Interval?

Pruning/sparsity?

Resources? Area/power?
Radiation? Cryo?

Microarchitecture

Parallelization



Fermilab

Hardware - algorithm codesign

Data representation
→ ML architecture

Neural architecture search/
Hyperparameter optimization

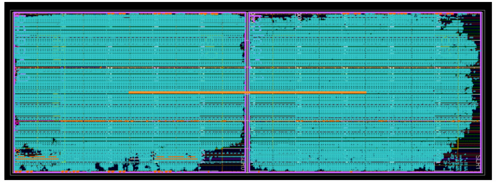
Physics requirements

Quantize network

Intermediate (quantized)
representations

Latency?
Pipeline Interval?

Pruning/sparsity?


BNL711 FELIX Firmware Floorplanning

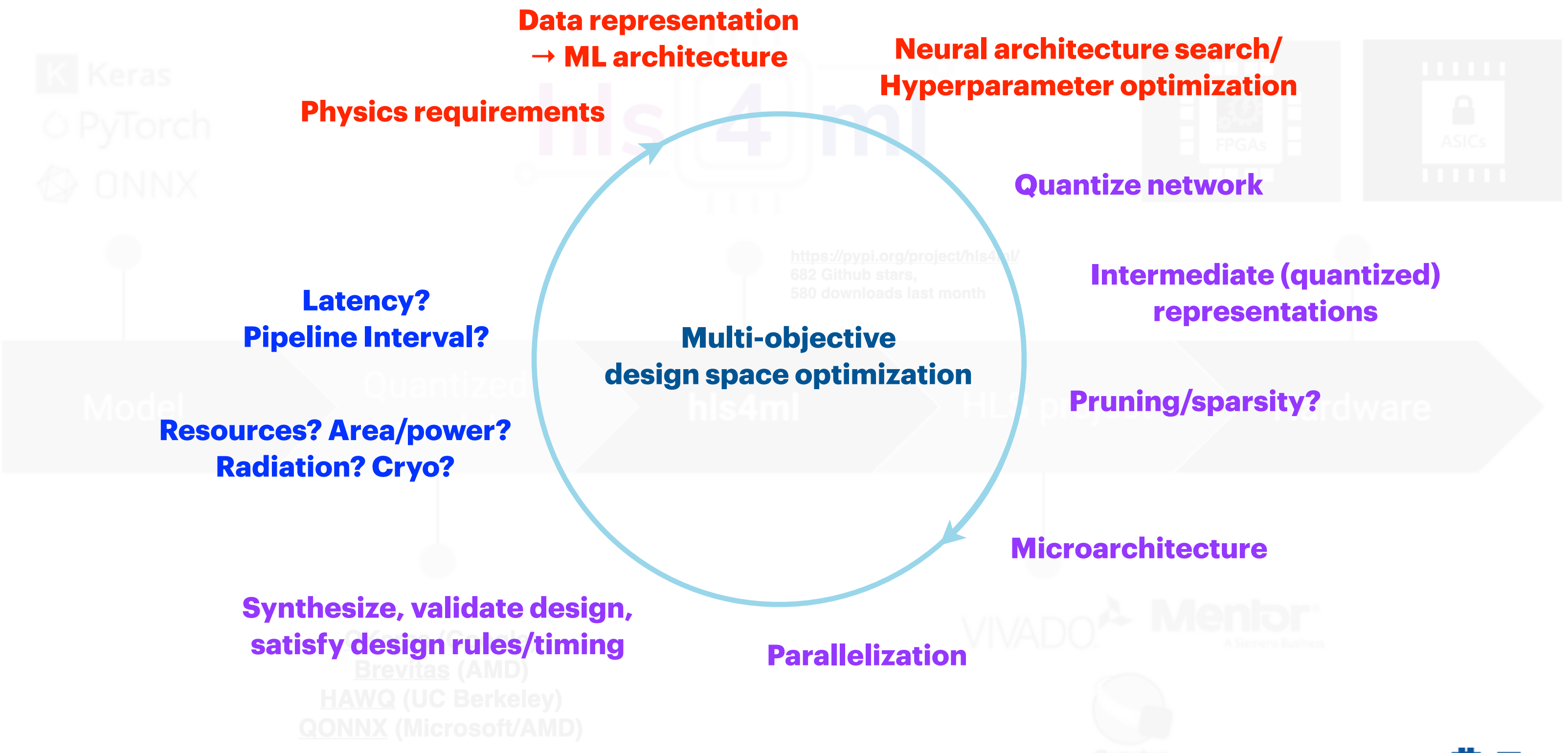
/power?
yo?

Microarchitecture

Synthesize, validate design,
satisfy design rules/timing

Parallelization

Hardware - algorithm codesign



Codesign with o1

can you help me develop design a 4 layer MLP using QKeras that can be translated to AMD FPGAs using hls4ml?

Codesign with o1

can you help me develop design a 4 layer MLP using QKeras that can be translated to AMD FPGAs using hls4ml?

2. Design Your Quantized MLP Using QKeras

Here's how you can define a 4-layer MLP with quantized weights and activations:

python   Copy code

```python
import tensorflow as tf
from tensorflow.keras.models import Sequential
from qkeras import QDense, QActivation
from qkeras.quantizers import quantized_bits, quantized_relu

# Define input and output sizes
input_size = 32   # Replace with your actual input feature size
output_size = 10  # Replace with your actual number of classes

# Define quantizers
weight_quantizer = quantized_bits(8, 0, alpha=1)        # 8-bit quantization for we
activation_quantizer = quantized_relu(8, 0)             # 8-bit quantization for ac

# Build the model
model = Sequential([
    QDense(64, input_shape=(input_size,),
           kernel_quantizer=weight_quantizer,
           bias_quantizer=weight_quantizer,
           name='fc1'),
    QActivation(activation_quantizer, name='act1'),
```

Fermilab

Codesign with o1

can you help me develop design a 4 layer MLP using QKeras that can be translated to AMD FPGAs using hls4ml?

2. Design Your Quantized MLP Using QKeras

Here's how you can define a 4-layer MLP with quantized weights and activations:

python Copy code

```python
import tensorflow as tf
from tensorflow.keras.models import Sequential
from qkeras import QDense, QActivation
from qkeras.quantizers import quantized_bits, quantized_relu

# Define input and output sizes
input_size = 32   # Replace with your actual input feature size
output_size = 10  # Replace with your actual number of classes

# Define quantizers
weight_quantizer = quantized_bits(8, 0, alpha=1)        # 8-bit quantization for we
activation_quantizer = quantized_relu(8, 0)             # 8-bit quantization for ac

# Build the model
model = Sequential([
    QDense(64, input_shape=(input_size,),
           kernel_quantizer=weight_quantizer,
           bias_quantizer=weight_quantizer,
           name='fc1'),
    QActivation(activation_quantizer, name='act1'),
```

3. Train Your Model

4. Prepare for Conversion with hls4ml

5. Convert the Model Using hls4ml

Specify the FPGA part number corresponding to your AMD FPGA. Since AMD acquired Xilinx, you can use Xilinx part numbers.

python Copy code

```python
hls_model = hls4ml.converters.convert_from_keras_model(
    model,
    hls_config=config,
    output_dir='hls4ml_prj',
    part='xcvu9p-flgb2104-2-i'  # Replace with your specific FPGA part number
)
```

Fermilab

Codesign with o1

can you help me develop design a 4 layer MLP using QKeras that can be translated to AMD FPGAs using hls4ml?

2. Design Your Quantized MLP Using QKeras

Here's how you can define a 4-layer MLP with quantized weights and activations:

python Copy code

```python
import tensorflow as tf
from tensorflow.keras.models import Sequential
from qkeras import QDense, QActivation
from qkeras.quantizers import quantized_bits, quantized_relu

# Define input and output sizes
input_size = 32   # Replace with your actual input feature size
output_size = 10  # Replace with your actual number of classes

# Define quantizers
weight_quantizer = quantized_bits(8, 0, alpha=1)        # 8-bit quantization for we
activation_quantizer = quantized_relu(8, 0)              # 8-bit quantization for ac

# Build the model
model = Sequential([
    QDense(64, input_shape=(input_size,),
           kernel_quantizer=weight_quantizer,
           bias_quantizer=weight_quantizer,
           name='fc1'),
    QActivation(activation_quantizer, name='act1'),
```

3. Train Your Model
4. Prepare for Conversion with hls4ml
5. Convert the Model Using hls4ml

Specify the FPGA part number corresponding to your AMD FPGA. Since AMD acquired Xilinx, you can use Xilinx part numbers.

python Copy code

```python
hls_model = hls4ml.converters.convert_from_keras_model(
    model,
    hls_config=config,
    output_dir='hls4ml_prj',
    part='xcvu9p-flgb2104-2-i'  # Replace with your specific FPGA part number
)
```

6. Compile and Build the HLS Model

7. Validate the HLS Model

Compare the outputs of the original Keras model and the HLS model to ensure consistency.

python Copy cod

```python
import numpy as np

# Generate some test data
X_test = np.random.rand(100, input_size)
```

8. Synthesize the Design Using Vivado HLS

Codesign with o1

how many FPGA resources will this model take?

Codesign with o1

how many FPGA resources will this model take?

Estimating Resource Utilization

While I can't provide exact numbers without running the synthesis, I can guide you on how to estimate and obtain the resource utilization for your model.

1. Understanding the Resource Components

- **Lookup Tables (LUTs)**: Implement combinational logic.
- **Flip-Flops (FFs)**: Store state information.
- **Block RAMs (BRAMs)**: Used for on-chip memory storage.
- **Digital Signal Processing (DSP) Slices**: Perform arithmetic operations like multiplication and addition.

2. Factors Influencing Resource Usage

- **Neuron Count**: More neurons increase resource consumption.
- **Bit-widths**: Higher bit-widths for weights and activations consume more resources.
- **Parallelism**: Lower `ReuseFactor` values increase parallelism and resource usage.
- **Pipelining**: Increases FF usage but can improve performance.
- **Optimizations**: Strategies like resource sharing can reduce usage.

Codesign with o1

how many FPGA resources will this model take?

Estimating Resource Utilization

While I can't provide exact numbers without running the synthesis, I can guide you on how to estimate and obtain the resource utilization for your model.

1. Understanding the Resource Components

- **Lookup Tables (LUTs)**: Implement combinational logic.
- **Flip-Flops (FFs)**: Store state information.
- **Block RAMs (BRAMs)**: Used for on-chip memory storage.
- **Digital Signal Processing (DSP) Slices**: Perform arithmetic operations like multiplication and addition.

2. Factors Influencing Resource Usage

- **Neuron Count**: More neurons increase resource consumption.
- **Bit-widths**: Higher bit-widths for weights and activations consume more resources.
- **Parallelism**: Lower `ReuseFactor` values increase parallelism and resource usage.
- **Pipelining**: Increases FF usage but can improve performance.
- **Optimizations**: Strategies like resource sharing can reduce usage.

Estimating Based on Model Parameters

To get a rough estimate, consider the following:

1. Neuron Connections

Each fully connected layer has `input_size * output_size` weights.

- **First Layer (fc1)**: `32 inputs * 64 outputs = 2048 weights`
- **Second Layer (fc2)**: `64 inputs * 64 outputs = 4096 weights`
- **Third Layer (fc3)**: `64 inputs * 64 outputs = 4096 weights`
- **Output Layer**: `64 inputs * 10 outputs = 640 weights`

2. Bit-width Impact

With 8-bit quantization:

- **Total Weights Memory**: `(2048 + 4096 + 4096 + 640) weights * 8 bits ≈ 88 KB`

3. DSP Slice Usage

Each multiplication can use a part of a DSP slice:

- **Total Multiplications per Inference**: Sum of weights.

However, hls4ml can optimize to reuse DSP slices depending on the `ReuseFactor`.

4. ReuseFactor Impact

- **ReuseFactor = 1**: Maximum parallelism, highest resource usage.
- **Higher ReuseFactor**: Reduces parallelism, lowers resource usage but increases latency.

Codesign with o1

how many FPGA resources will this model take?

Estimating Resource Utilization

While I can't provide exact numbers without running the synthesis, I can guide you on how to estimate and obtain the resource utilization for your model.

1. Understanding the Resource Components

- **Lookup Tables (LUTs)**: Implement combinational logic.
- **Flip-Flops (FFs)**: Store state information.
- **Block RAMs (BRAMs)**: Used for on-chip memory storage.
- **Digital Signal Processing (DSP) Slices**: Perform arithmetic operations like multiplication and addition.

2. Factors Influencing Resource Usage

- **Neuron Count**: More neurons increase resource consumption.
- **Bit-widths**: Higher bit-widths for weights and activations consume more resources.
- **Parallelism**: Lower `ReuseFactor` values increase parallelism and resource usage.
- **Pipelining**: Increases FF usage but can improve performance.
- **Optimizations**: Strategies like resource sharing can reduce usage.

Estimating Based on Model Parameters

To get a rough estimate, consider the following:

1. Neuron Connections

Each fully connected layer has `input_size * output_size` weights.

- **First Layer (fc1)**: `32 inputs * 64 outputs = 2048 weights`
- **Second Layer (fc2)**: `64 inputs * 64 outputs = 4096 weights`
- **Third Layer (fc3)**: `64 inputs * 64 outputs = 4096 weights`
- **Output Layer**: `64 inputs * 10 outputs = 640 weights`

2. Bit-width Impact

With 8-bit quantization:

- **Total Weights Memory**: `(2048 + 4096 + 4096 + 640) weights * 8 bits ≈ 88 KB`

3. DSP Slice Usage

Each multiplication can use a part of a DSP slice:

- **Total Multiplications per Inference**: Sum of weights.

However, hls4ml can optimize to reuse DSP slices depending on the `ReuseFactor`.

4. ReuseFactor Impact

- **ReuseFactor = 1**: Maximum parallelism, highest resource usage.
- **Higher ReuseFactor**: Reduces parallelism, lowers resource usage but increases latency.

Conclusion

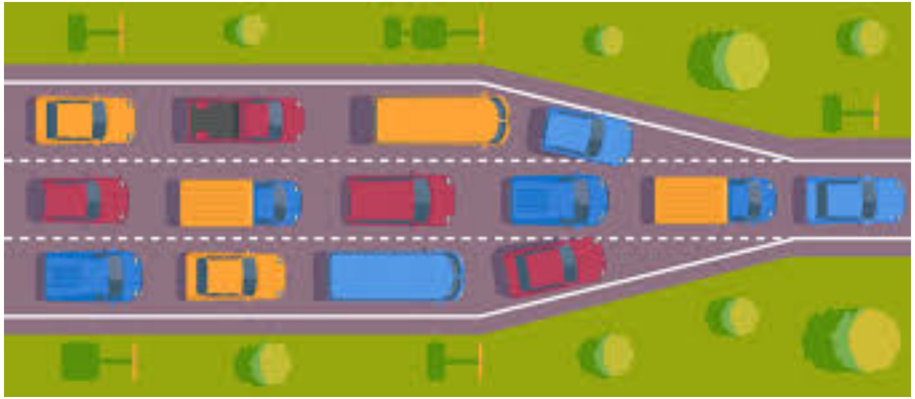
- **Exact Resource Usage**: Can only be determined by synthesizing the design and reviewing the reports.
- **Estimation**: Use the factors above to estimate resource usage.
- **Optimization**: Adjust model complexity, quantization bit-widths, and hls4ml configurations to meet resource constraints.

Codesign with o1

how many FPGA resources will this model take?

Estimating Resource Utilization

While I can't provide exact numbers without running the synthesis, I can guide you on how to estimate and obtain the resource utilization for your model.



- **Parallelism**: Lower `ReuseFactor` values increase parallelism and resource usage.
- **Pipelining**: Increases FF usage but can improve performance.
- **Optimizations**: Strategies like resource sharing can reduce usage.

Estimating Based on Model Parameters

To get a rough estimate, consider the following:

1. Neuron Connections

Each fully connected layer has `input_size * output_size` weights.

- **First Layer (fc1)**: `32 inputs * 64 outputs = 2048 weights`
- **Second Layer (fc2)**: `64 inputs * 64 outputs = 4096 weights`
- **Third Layer (fc3)**: `64 inputs * 64 outputs = 4096 weights`
- **Output Layer**: `64 inputs * 10 outputs = 640 weights`

2. Bit-width Impact

With 8-bit quantization:

- **Total Weights Memory**: `(2048 + 4096 + 4096 + 640) weights * 8 bits ≈ 88 KB`

3. DSP Slice Usage

Each multiplication can use a part of a DSP slice:

- **Total Multiplications per Inference**: Sum of weights.

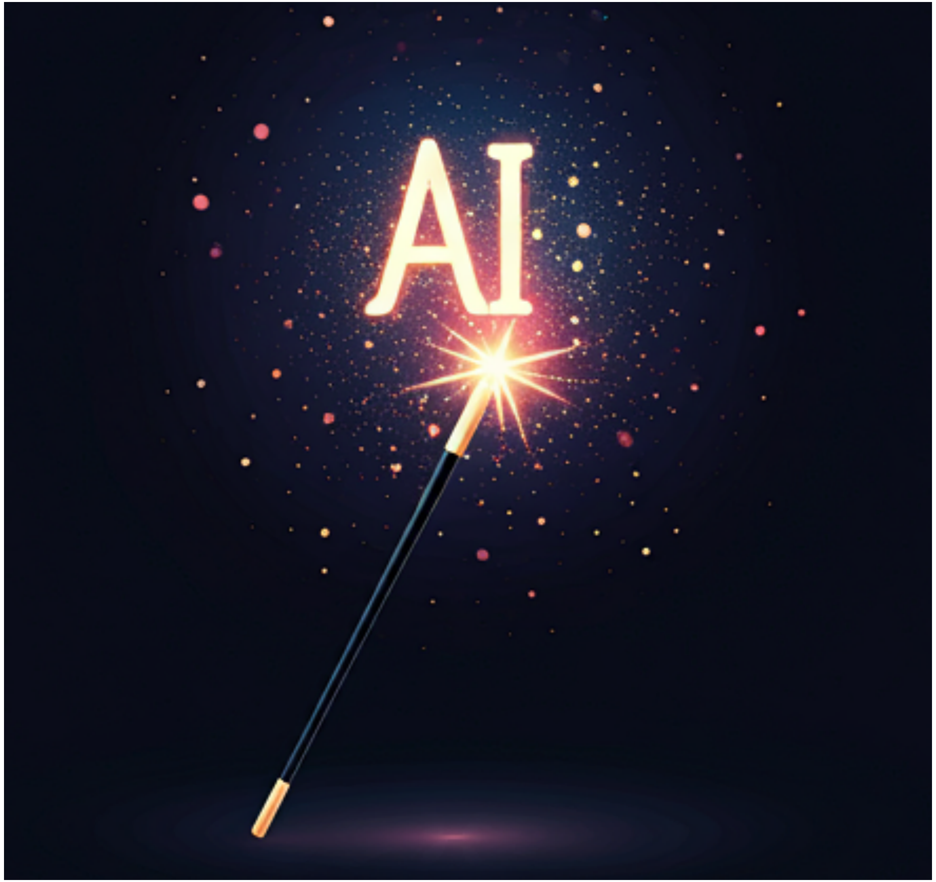
However, hls4ml can optimize to reuse DSP slices depending on the `ReuseFactor`.

4. ReuseFactor Impact

- **ReuseFactor = 1**: Maximum parallelism, highest resource usage.
- **Higher ReuseFactor**: Reduces parallelism, lowers resource usage but increases latency.

Conclusion

- **Exact Resource Usage**: Can only be determined by synthesizing the design and reviewing the reports.
- **Estimation**: Use the factors above to estimate resource usage.
- **Optimization**: Adjust model complexity, quantization bit-widths, and hls4ml configurations to meet resource constraints.


AI

wa-hls4ml & luiGNN

Hawks, Plotnikov et al
https://indico.cern.ch/event/1387540/contributions/6153600/
https://indico.cern.ch/event/1387540/contributions/6153564/

- Build a surrogate model of hls4ml to predict resources without running costly synthesis steps
- First dataset of its kind
  - > 100k models on NRP
  - Simple to start: MLPs
- Build a graph NN which predicts network FPGA resources
  - Each layer of the network is a layer in the graph
  - In distribution accuracy is ~few %

wa-hls4ml & luiGNN

Hawks, Plotnikov et al
https://indico.cern.ch/event/1387540/contributions/6153600/
https://indico.cern.ch/event/1387540/contributions/6153564/

- Build a surrogate model of hls4ml to predict resources without running costly synthesis steps
- First dataset of its kind
  - > 100k models on NRP
  - Simple to start: MLPs
- Build a graph NN which predicts network FPGA resources
  - Each layer of the network is a layer in the graph
  - In distribution accuracy is ~few %

Hyperparameter	Range	Step Size
Input Features	8–128	8
First Layer Neurons	8–128	8
Second Layer Neurons	8–128	8
Weight and Bias Precision (Total Bits)	2–16	2
Target Reuse Factor	1024–4093	1023

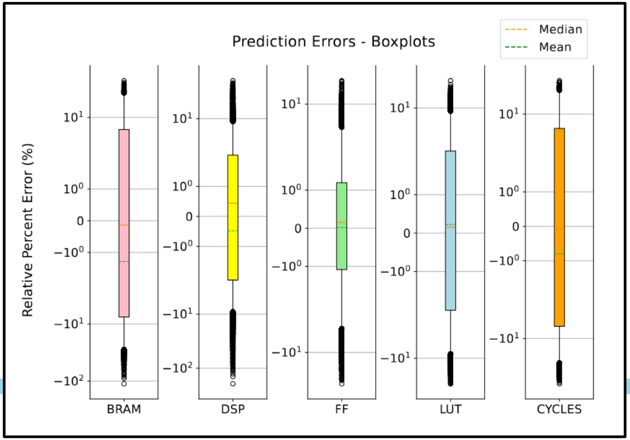
Fermilab

wa-hls4ml & luiGNN

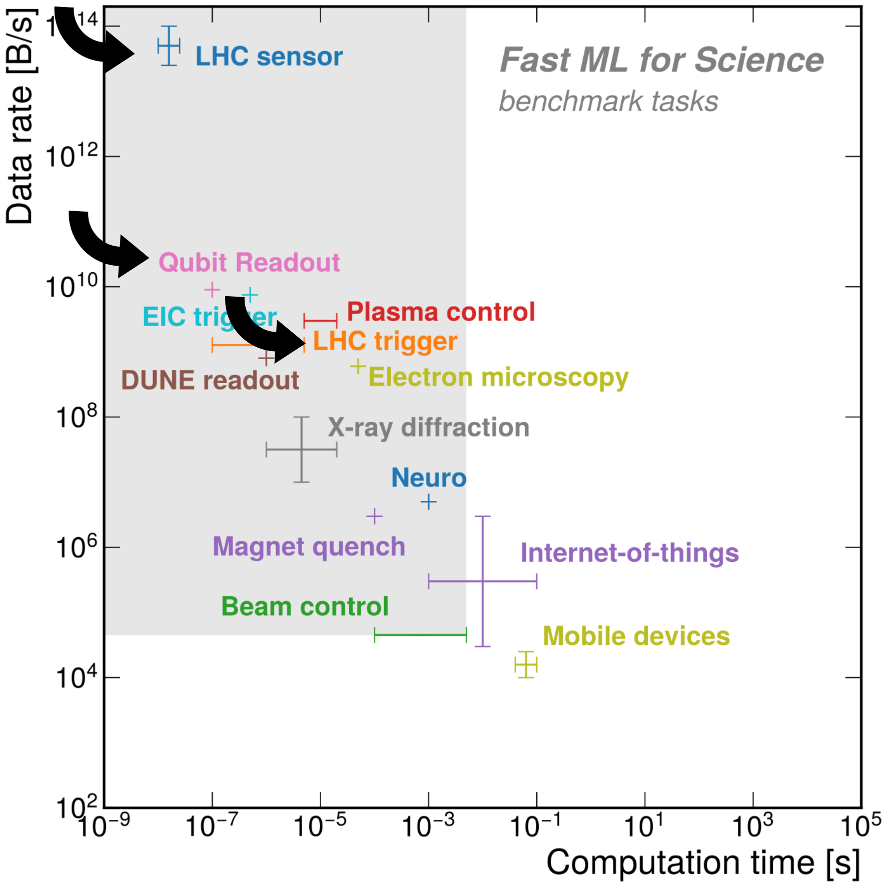
Hawks, Plotnikov et al
https://indico.cern.ch/event/1387540/contributions/6153600/
https://indico.cern.ch/event/1387540/contributions/6153564/

- Build a surrogate model of hls4ml to predict resources without running costly synthesis steps
- First dataset of its kind
  - > 100k models on NRP
  - Simple to start: MLPs
- Build a graph NN which predicts network FPGA resources
  - Each layer of the network is a layer in the graph
  - In distribution accuracy is ~few %

Hyperparameter	Range	Step Size
Input Features	8–128	8
First Layer Neurons	8–128	8
Second Layer Neurons	8–128	8
Weight and Bias Precision (Total Bits)	2–16	2
Target Reuse Factor	1024–4093	1023



Fast ML example applications



AI on ASIC

https://newsroom.sw.siemens.com/en-US/siemens-catapult-ai-nn/

PRESS RELEASE

Siemens simplifies development of AI accelerators for advanced system-on-chip designs with Catapult AI NN

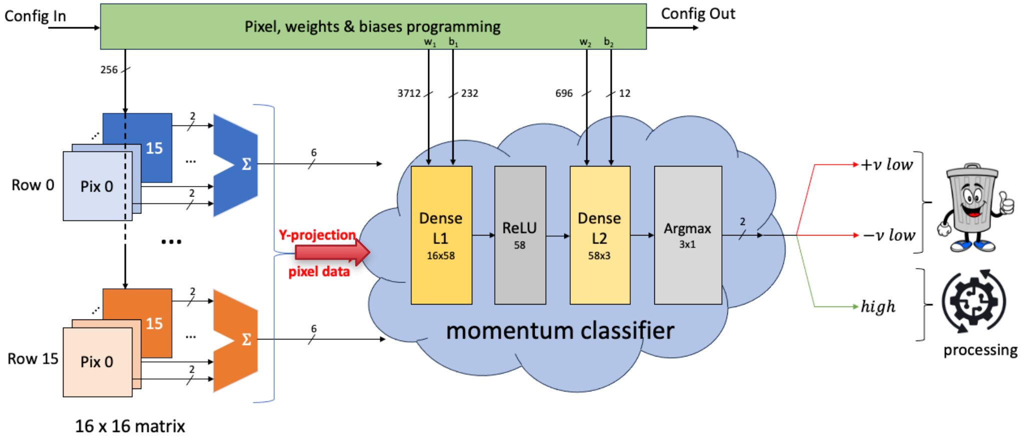
May 21, 2024
Plano, Texas

Catapult AI NN brings together hls4ml, an open-source package for machine learning hardware acceleration, and Siemens' Catapult™ HLS software for High-Level Synthesis. Developed in close collaboration with Fermilab, a U.S. Department of Energy Laboratory, and other leading contributors to hls4ml, Catapult AI NN addresses the unique requirements of machine learning accelerator design for power, performance, and area on custom silicon.

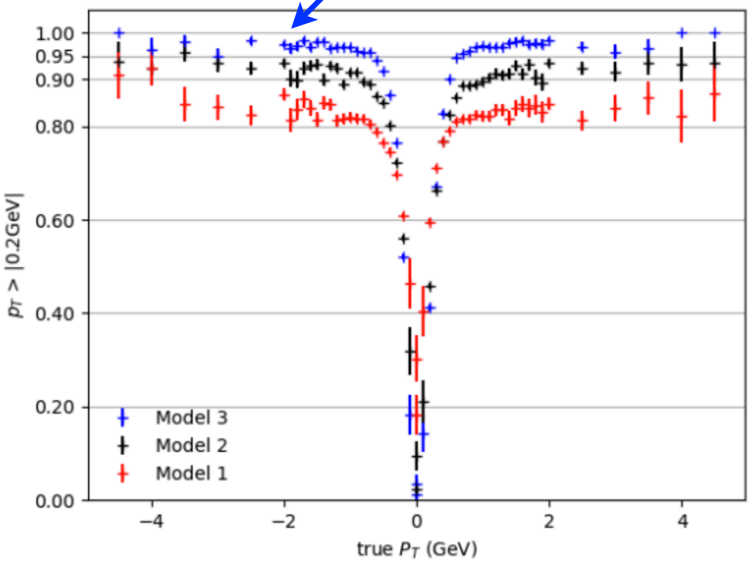
- Enables software development teams to seamlessly translate AI models designed in Python into silicon-based implementations, facilitating faster and more power-efficient execution compared to standard processors

Fermilab

Smart Pixels

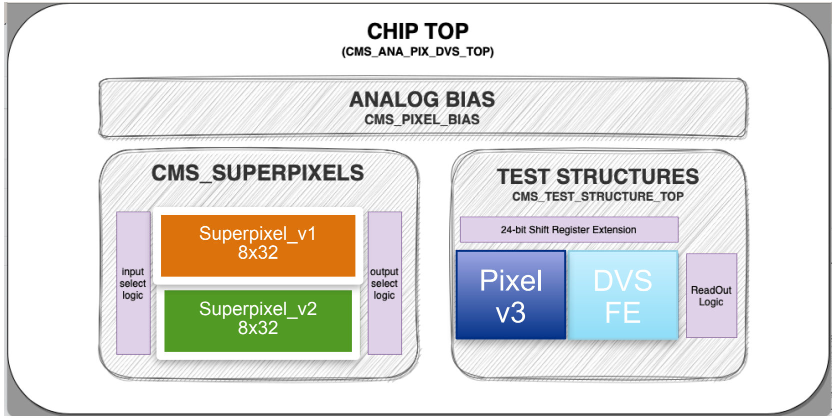


Including time slices

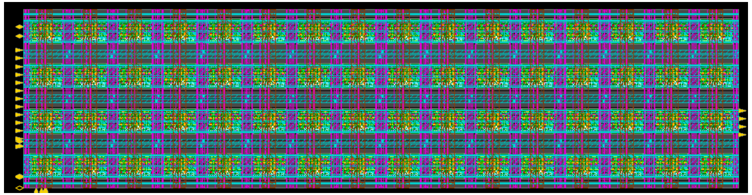


See recent talks by C. Mills and D. Shekar at CPAD this week
https://indico.phy.ornl.gov/event/510/contributions/2206/
https://indico.phy.ornl.gov/event/510/contributions/2260/

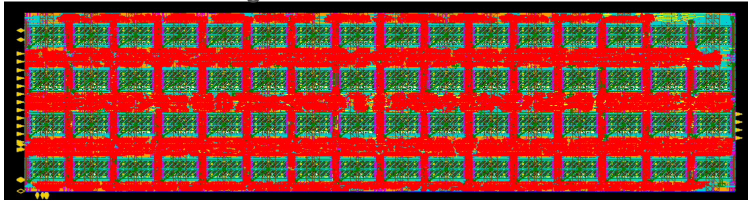
Smart Pixels



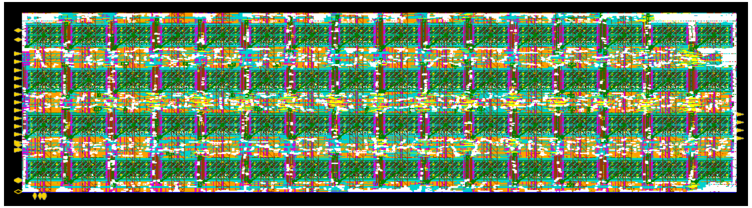
Floorplan with analog pixels and power a bias grid



Red: classifier algorithm



White: network weights

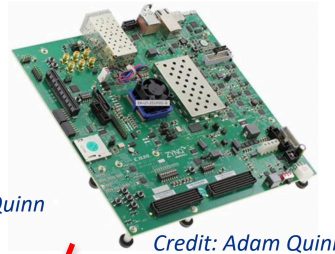


Fermilab

Smart Pixels

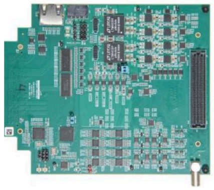
Credit: Adam Quinn

Python Inteface

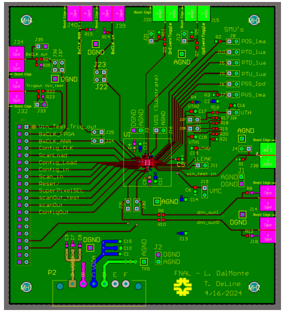


Credit: Adam Quinn

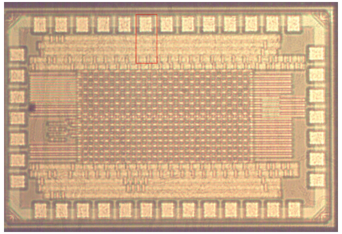
Zynq UltraScale+ MPSoC ZCU102 Eval Board



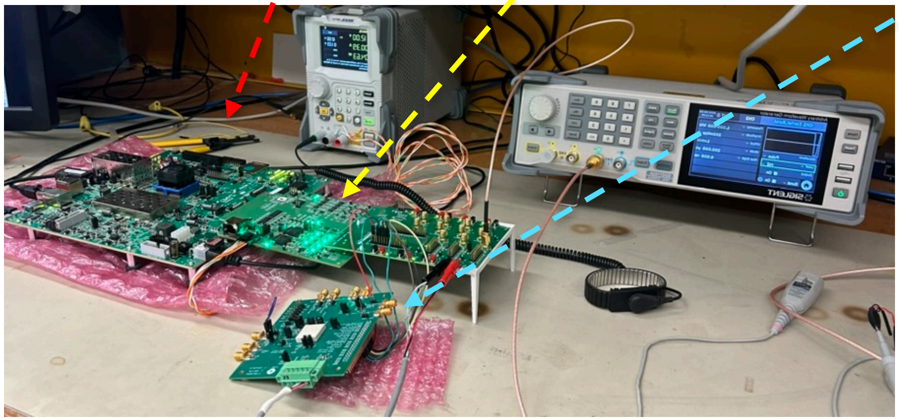
Caribou Board (Open-source DAQ System)

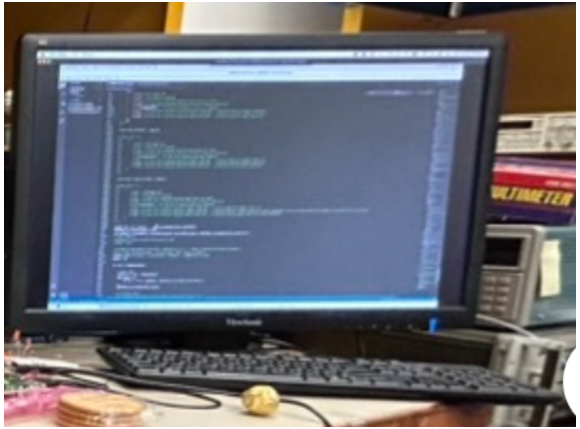


Custom ASIC board



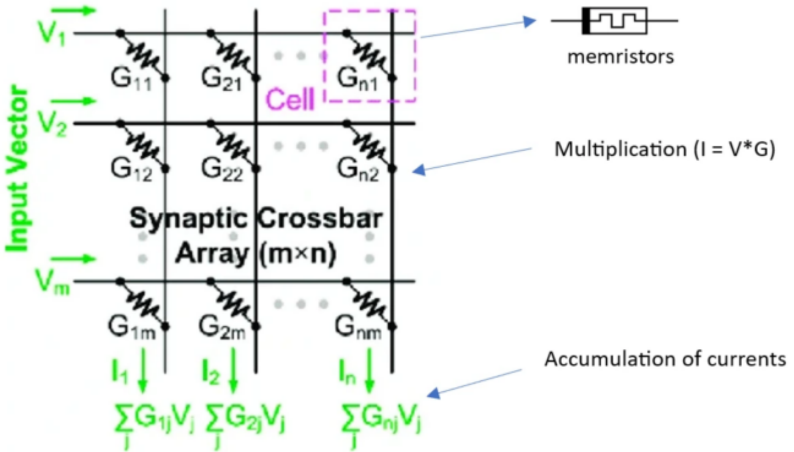
Wire bonded ASIC

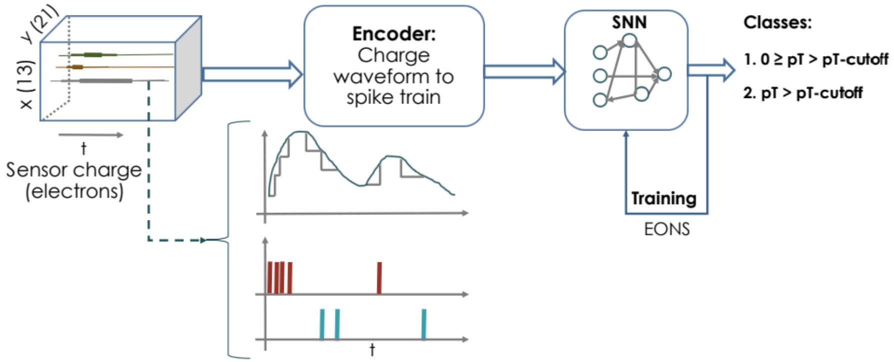




Smart Pixels

- Accessing sub-ns time component while meeting power requirements likely requires novel microelectronics solutions
  - Clocked digital CMOS solutions are power hungry
  - Consider neuromorphic approaches, e.g. analog or spiking NN



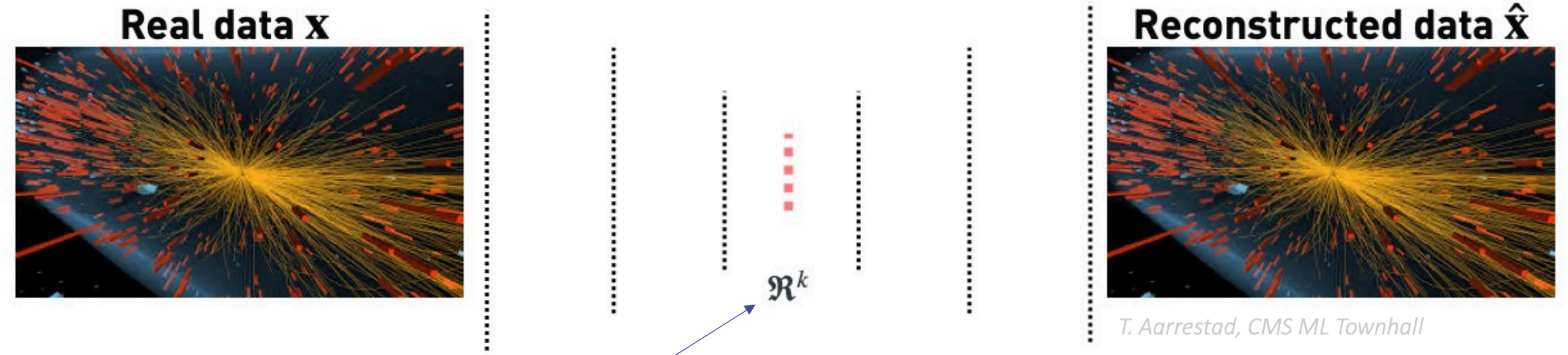


Link on analog compute; arXiv:2307.11242

Fermilab

AXOL1TL

https://cds.cern.ch/record/2904695

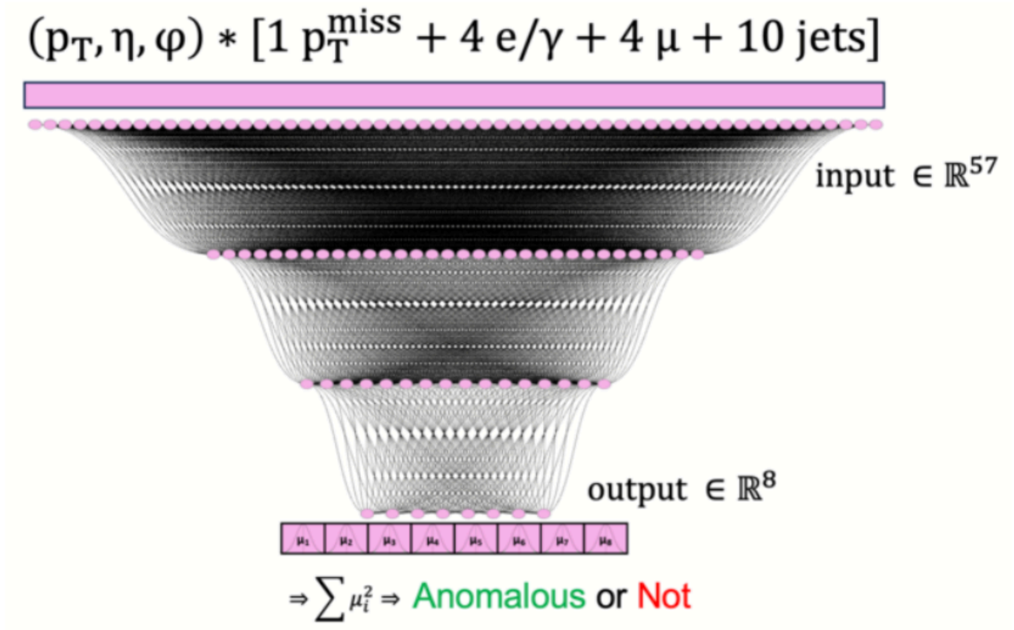


Train on ZeroBias LHC data

Bottleneck: autoencoder learns to compress high dimensional inputs into low dimensional latent space

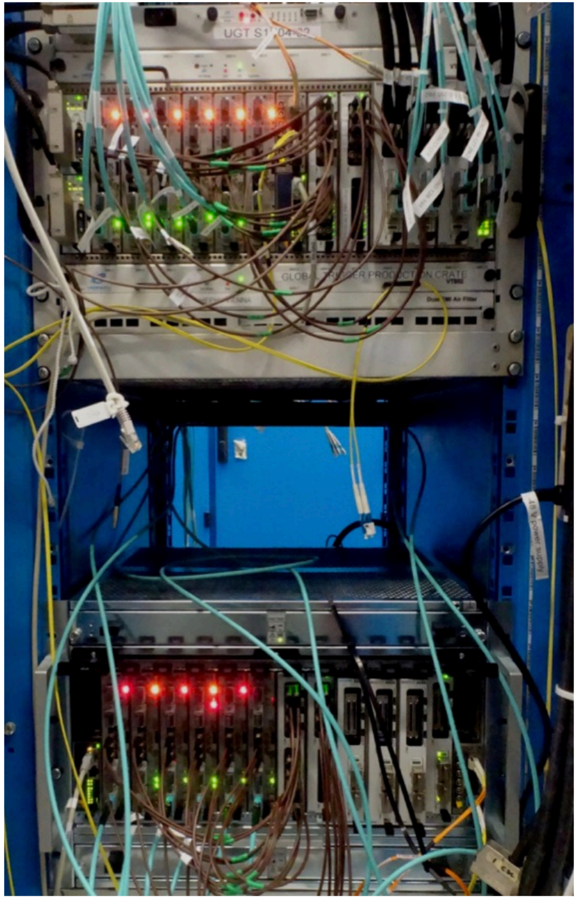
$x - \hat{x}$ represents degree of abnormality

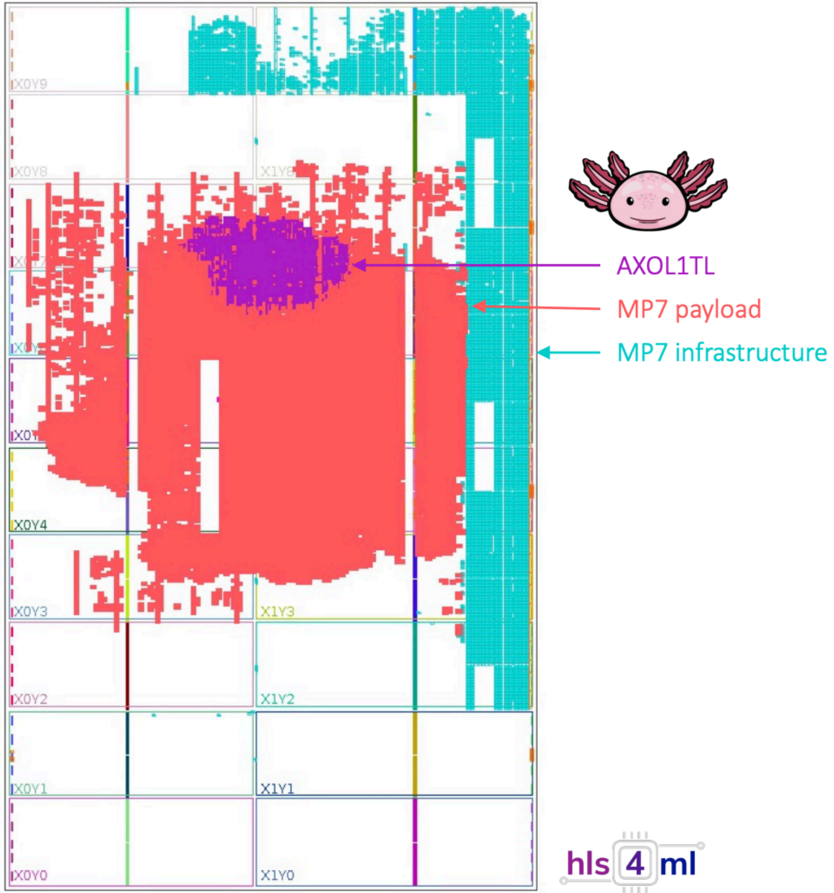
AXOL1TL



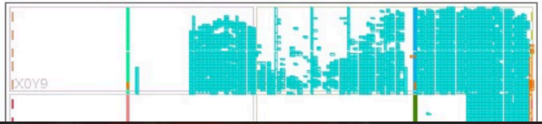
	Latency	LUTs	FFs	DSPs	BRAMs
AXOL1TL	2 ticks 50 ns	2.1%	~0	0	0

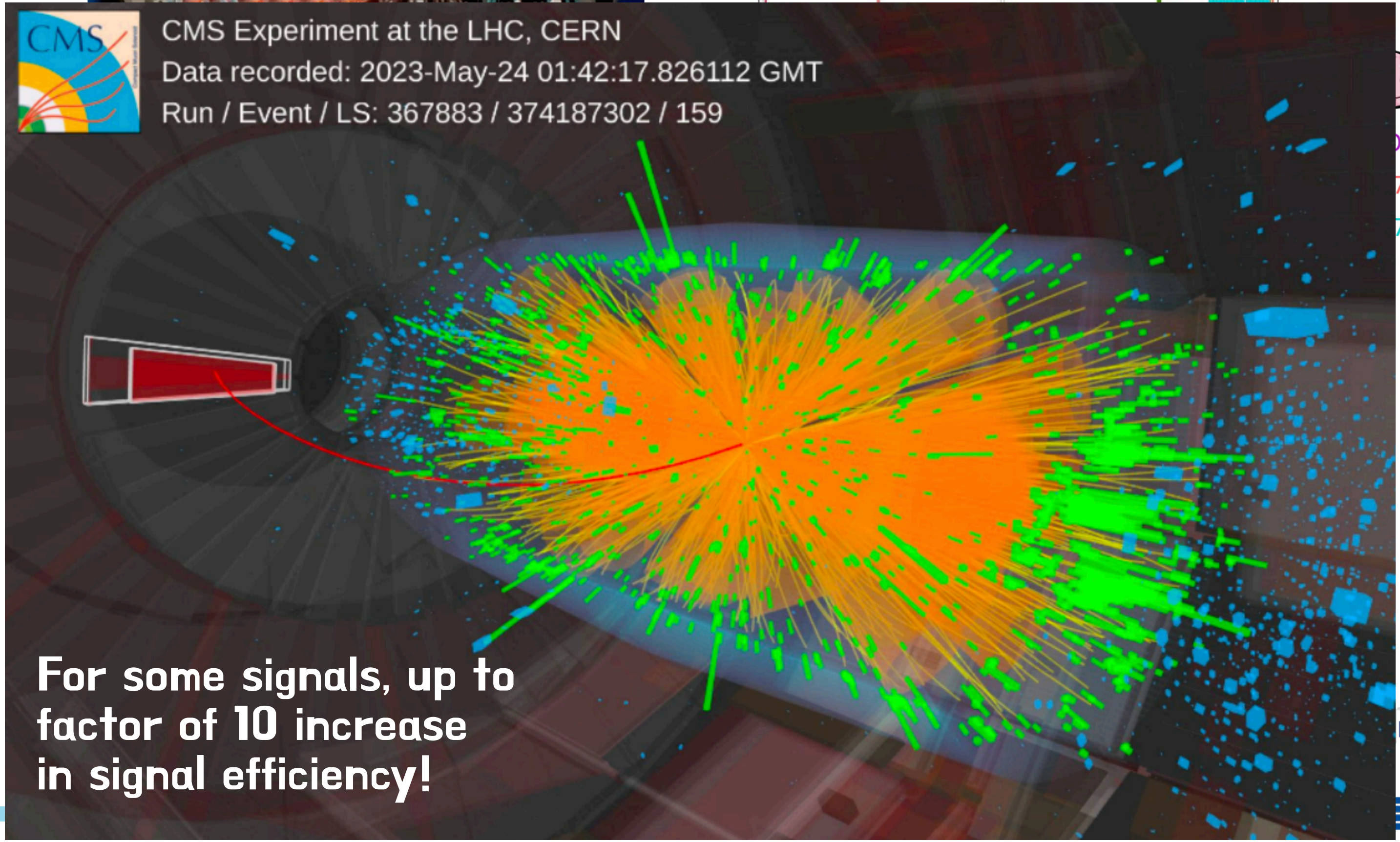
AXOL1TL IRL

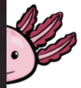

UGT S1
GLOBAL TRIGGER PRODUCTION CRATE


X0Y9
X0Y8
X1Y8
X0Y4
X0Y3
X1Y3
X0Y2
X1Y2
X0Y1
X1Y1
X0Y0
X1Y0
AXOL1TL
MP7 payload
MP7 infrastructure

AXOL1TL IRL







L1TL

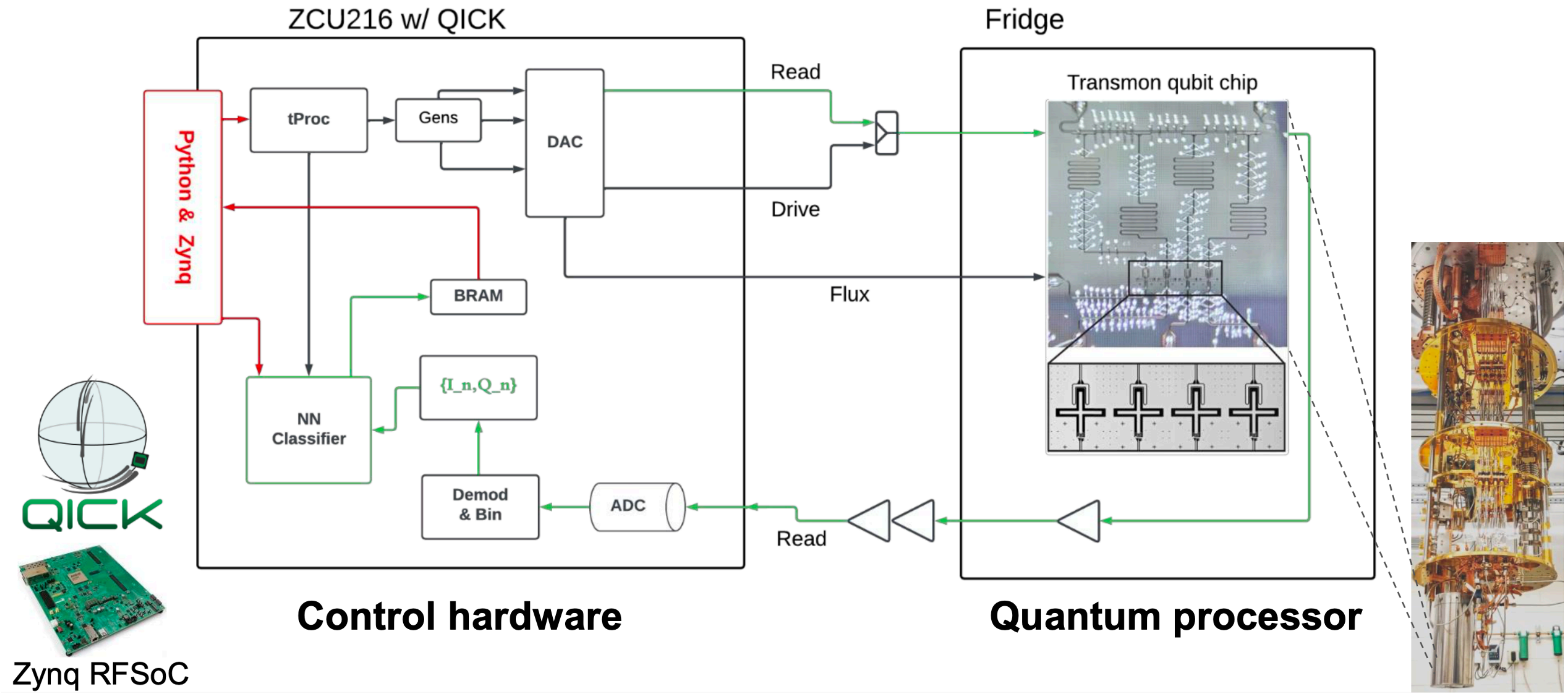
7 payload

7 infrastructure

ml

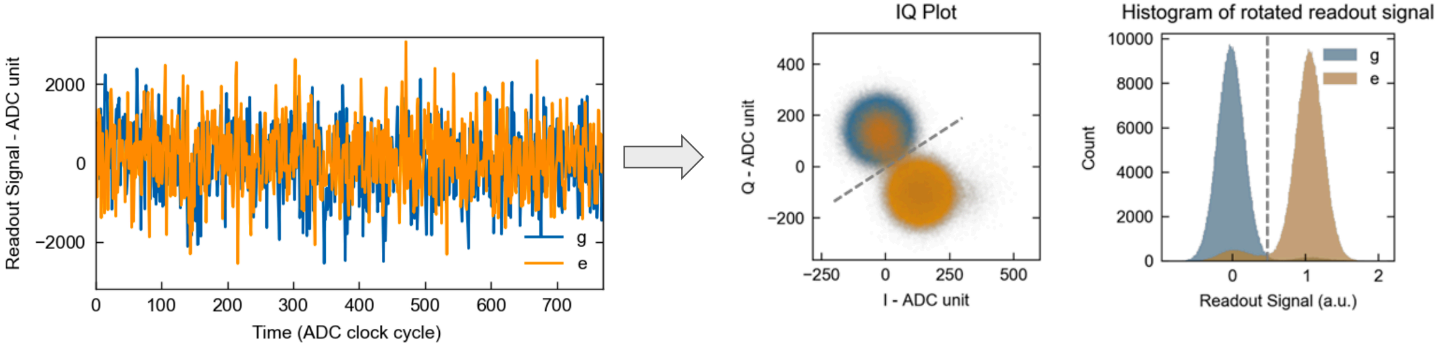
ML-based qubit readout

B. Du et al, https://indico.cern.ch/event/1387540/contributions/6153407/

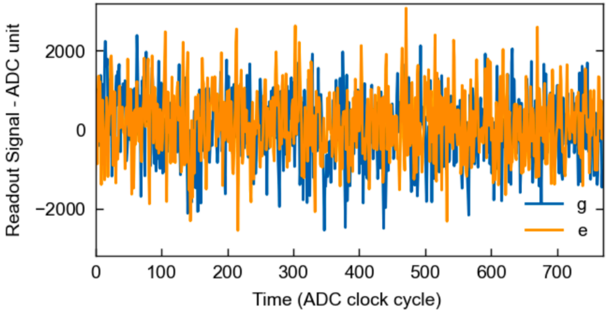


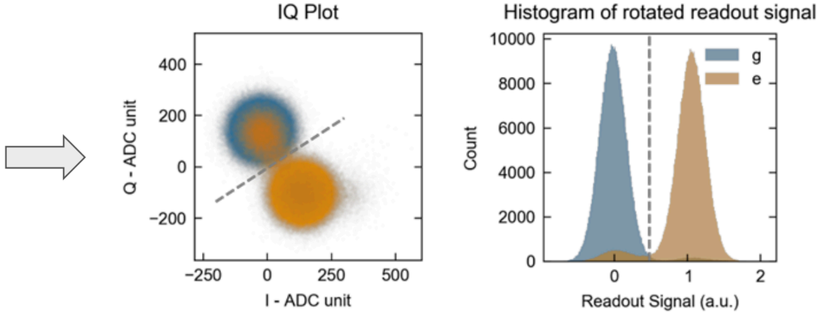
Readout frequencies in the RF, ~few GHz
Control latency < 1 μs

ML-based qubit readout

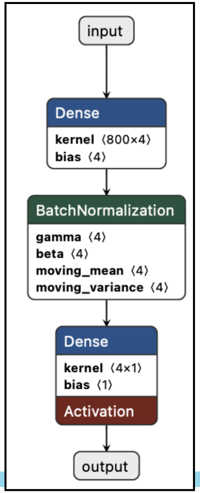


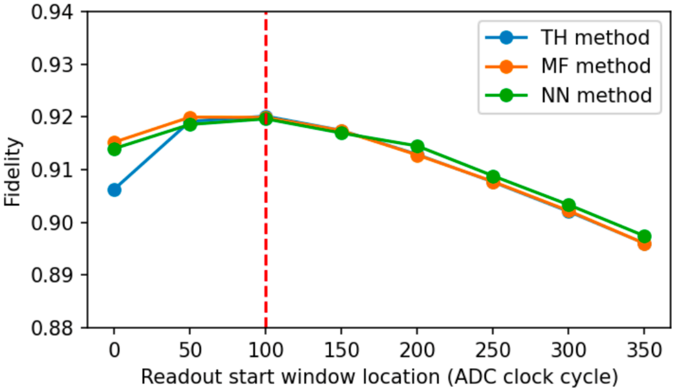
ML-based qubit readout



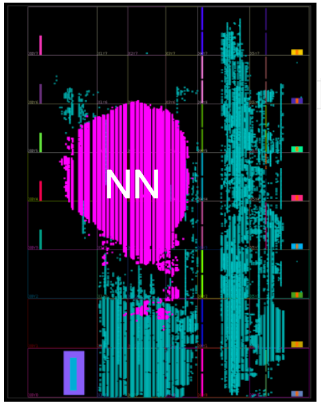


Ternary, 2-bit model

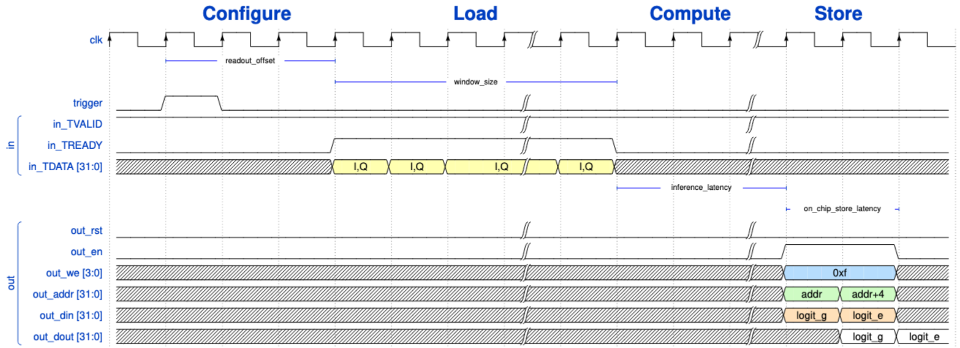




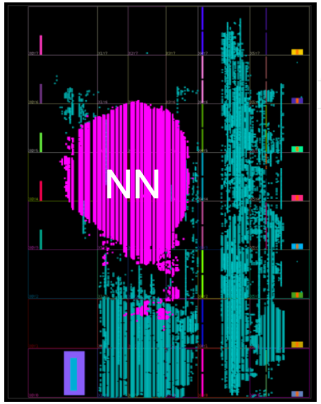
ML-based qubit readout



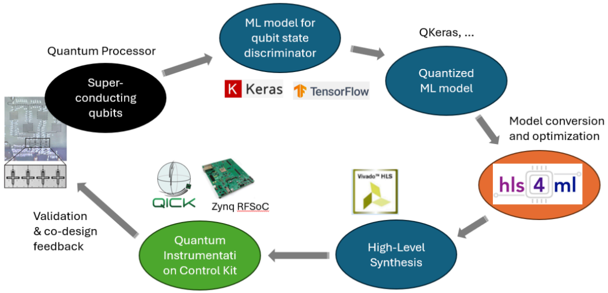
< 40 ns latency and
< 6% of FPGA resources



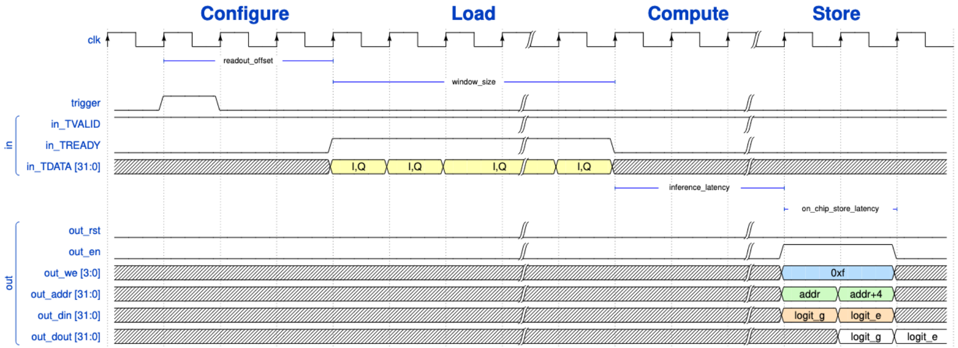
ML-based qubit readout

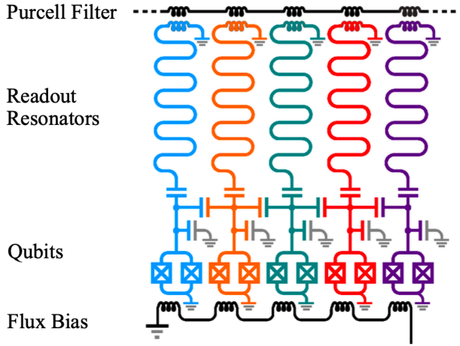


< 40 ns latency and
< 6% of FPGA resources



Proof-of-concept end-to-end workflow established, extendable to multi-qubit systems and adaptive automated operation

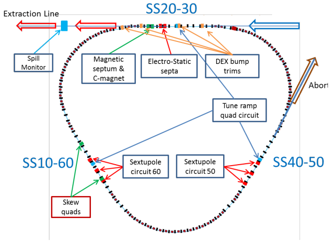


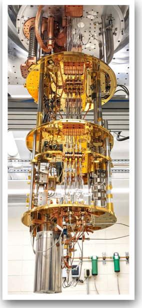


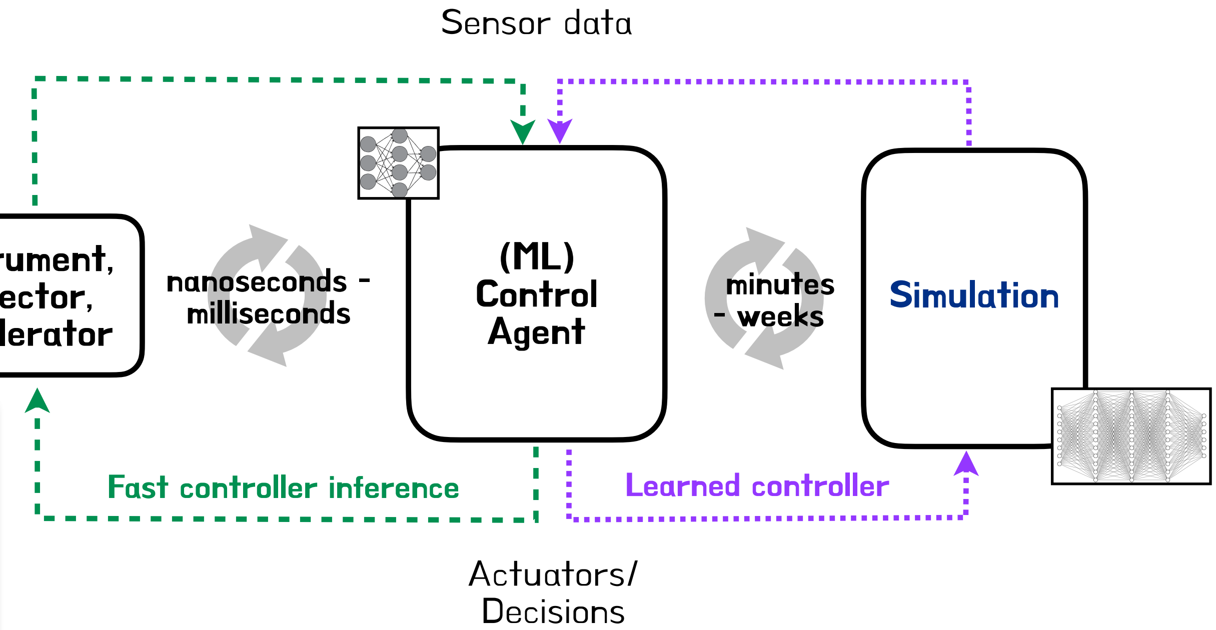
Outline

- Fast and Slow
- Fast ML and hardware codesign
- **Fast and Slow ML together**
- Slow ML & Real-Time

Adaptive systems



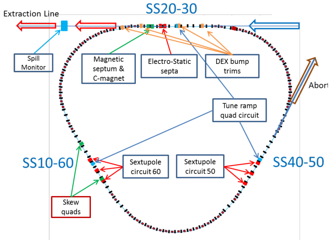


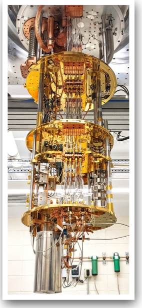


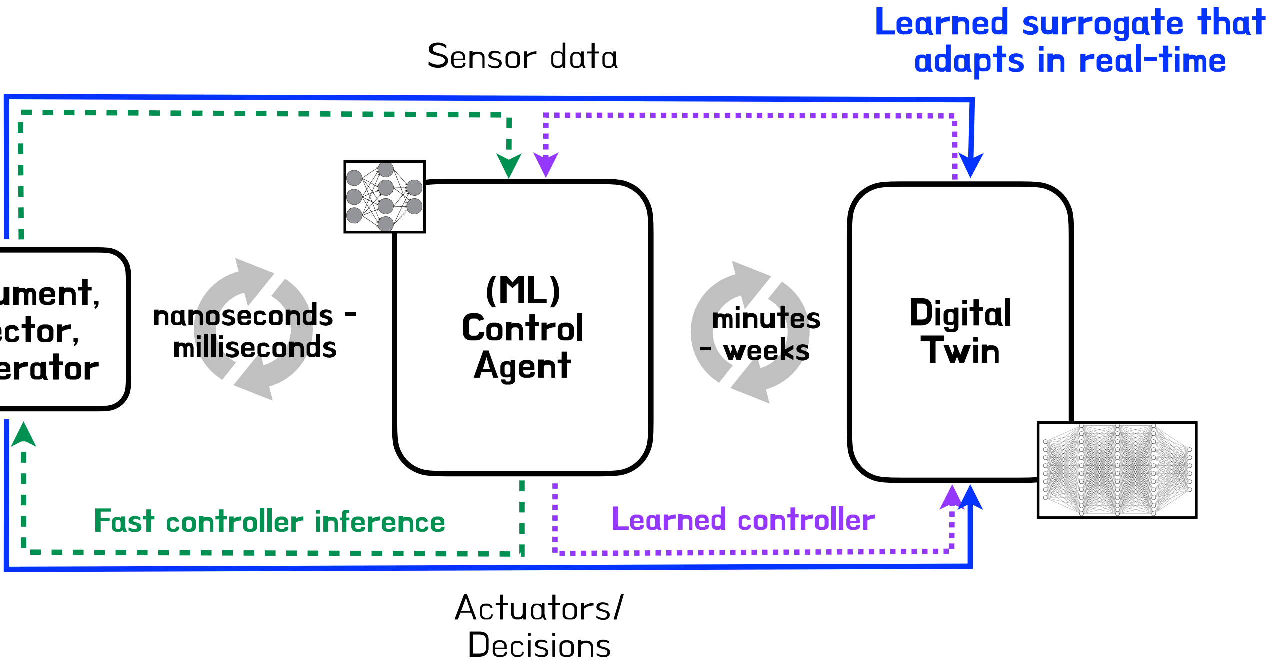
Fermilab

Adaptive systems

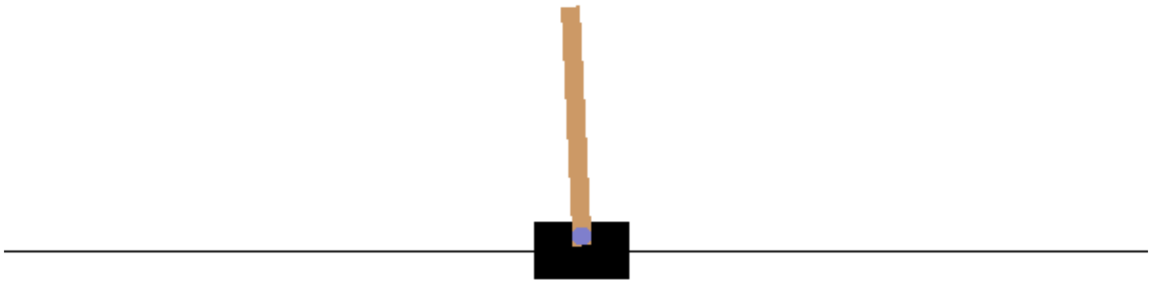
Summit on Digital Twins last Tuesday
https://indico.fnal.gov/event/66849/

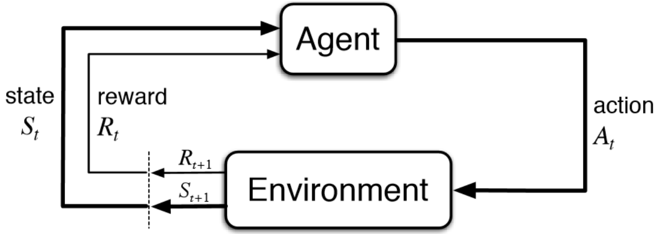




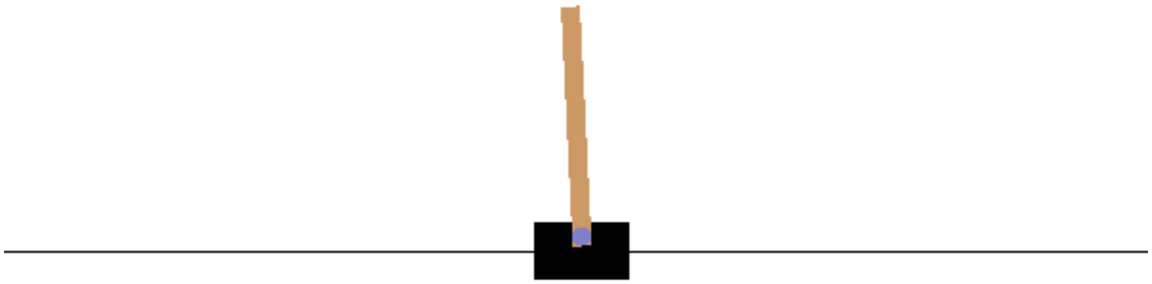


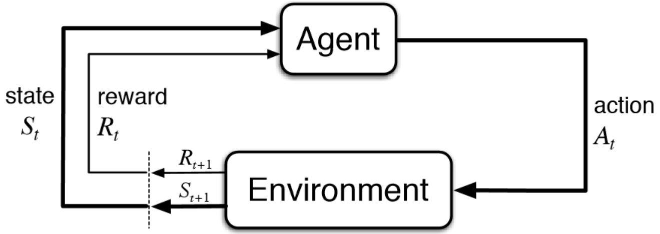
Cart-pole: canonical reinforcement learning



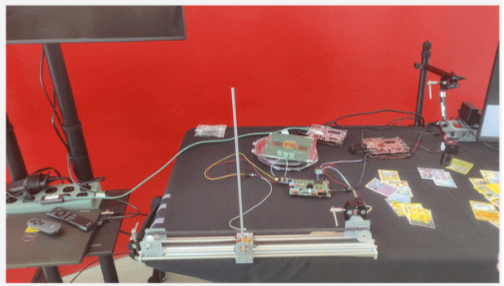


Cart-pole: canonical reinforcement learning

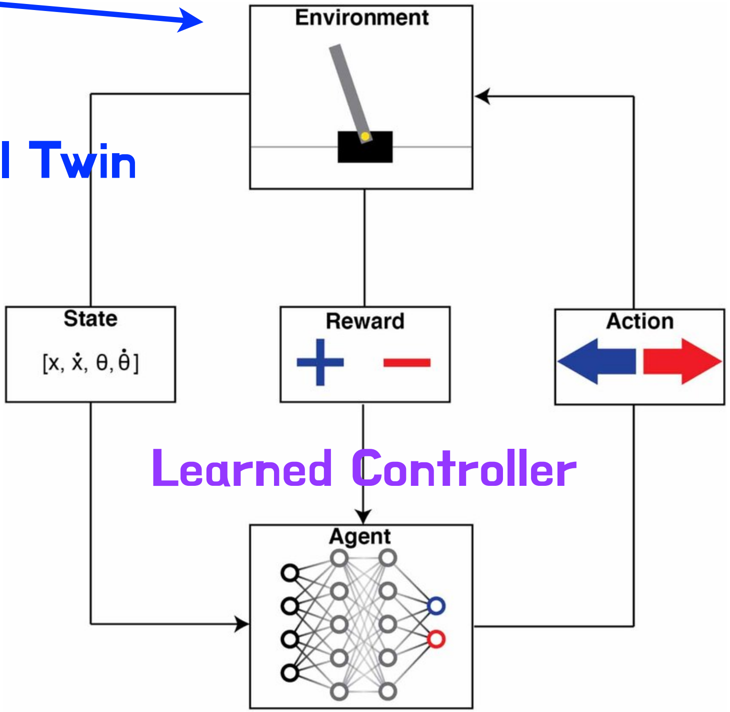




Cart-pole IRL

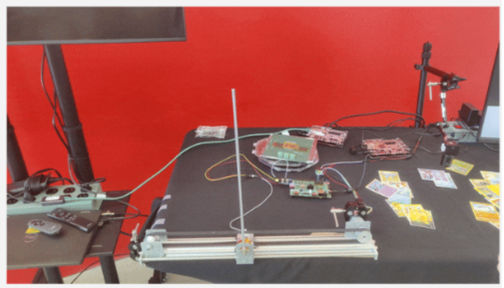


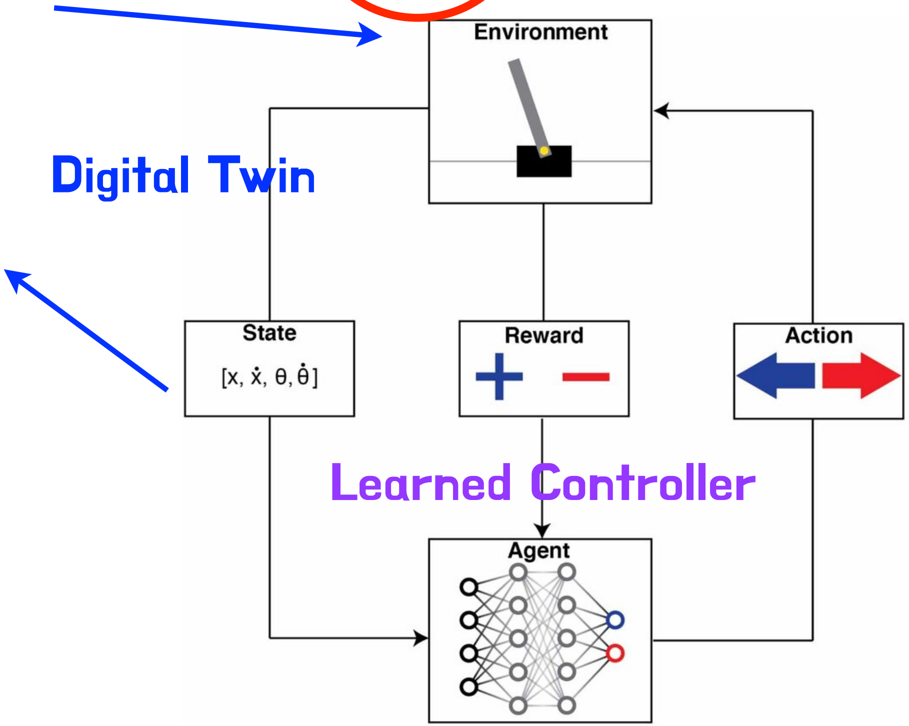
Digital Twin



Learned Controller

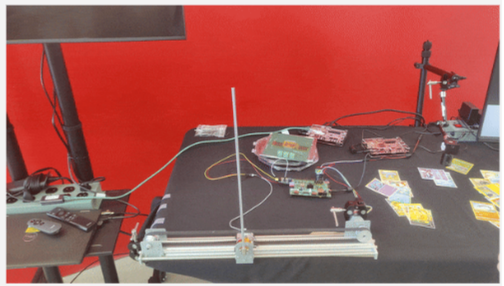
Cart-pole IRL



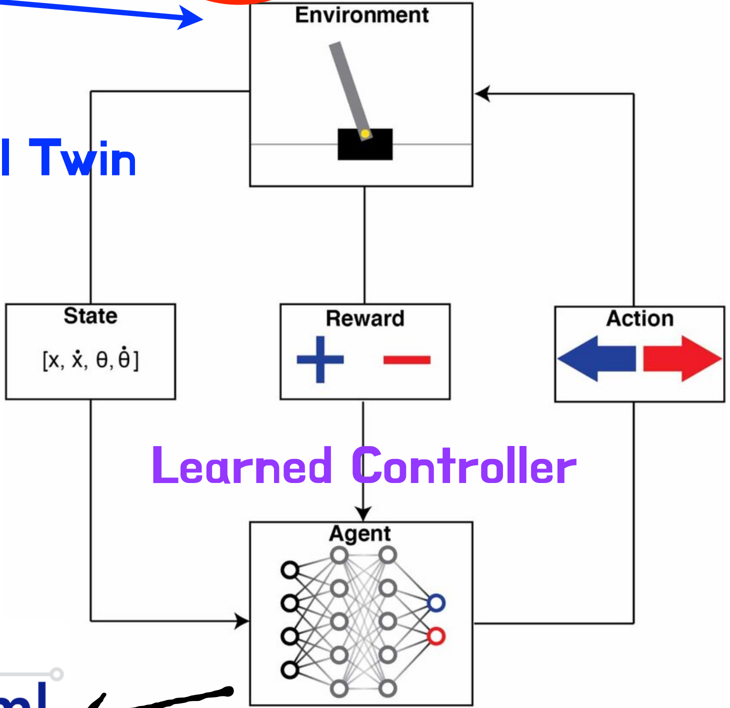


Fermilab

Cart-pole IRL

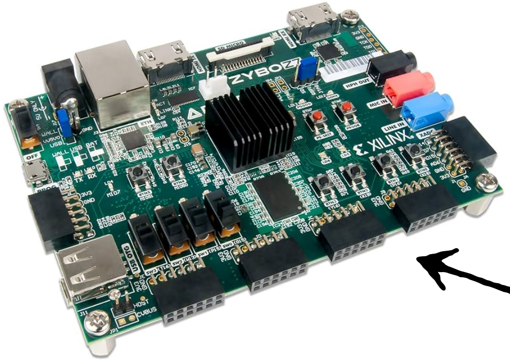


Digital Twin



Fast Control

Learned Controller



hls4ml

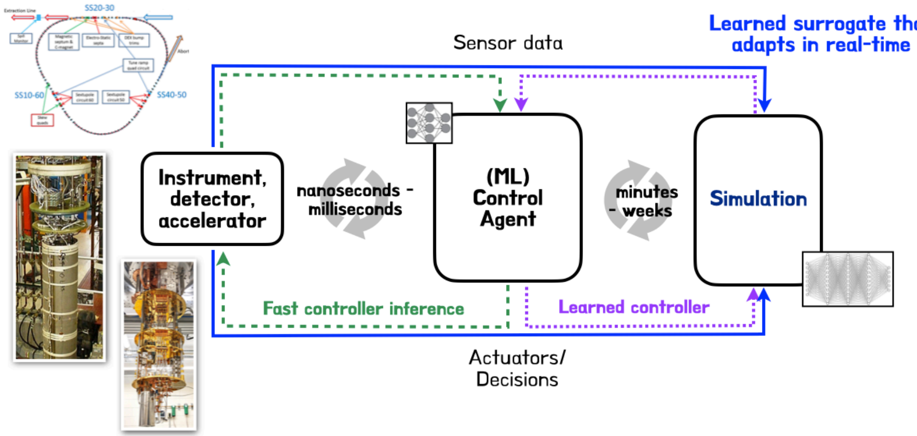
Demo!

🍀 🧧 🤞

Credit: Marcin Paluch (ETH Zurich)
& Ben Hawks (FNAL), Dennis Plotnikov (SULI, JHU)

Thoughts on adaptive systems

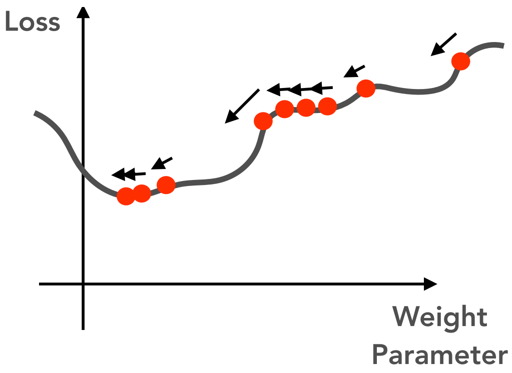
- Real-time feedback often deal with raw data - only one crack at the data!
- The performance of the twin and the control agent are interconnected
- What are the right time scales?
- A robust, adaptive ML control agent impacts system performance and timescales



Fermilab

Learning = optimization

learning as *optimization*



to learn the weights, we need the **derivative** of the loss w.r.t. the weight

i.e. *"how should the weight be updated to decrease the loss?"*

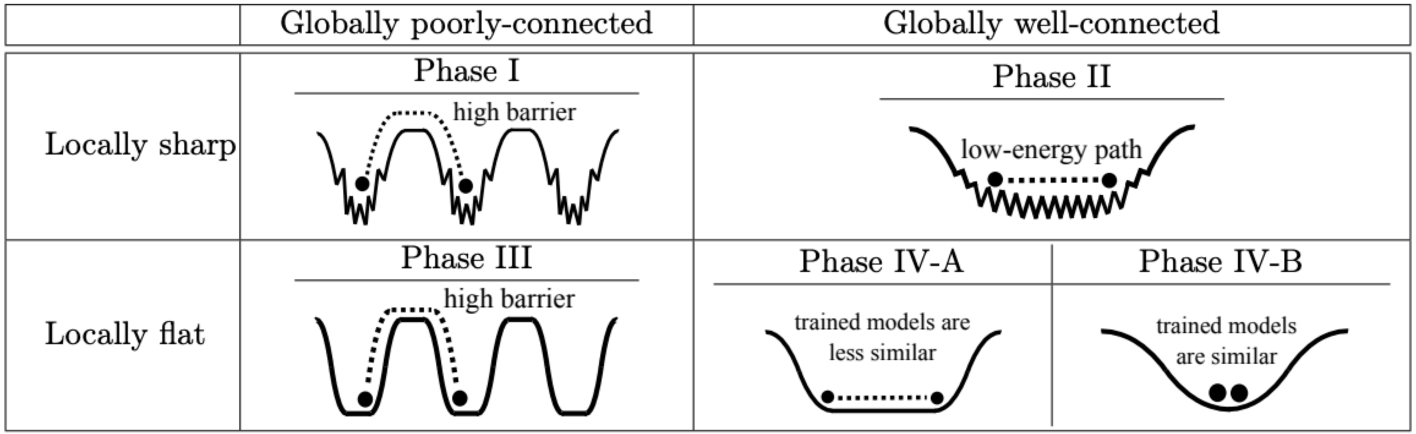
$$w' = w - \alpha \frac{\partial \mathcal{L}}{\partial w}$$

Robustness, a priori

- Goal - train a model to be more robust to model perturbations - more generalizable
  - Input perturbations = noise and lost channels
  - Weight perturbations = bit flips due to radiation

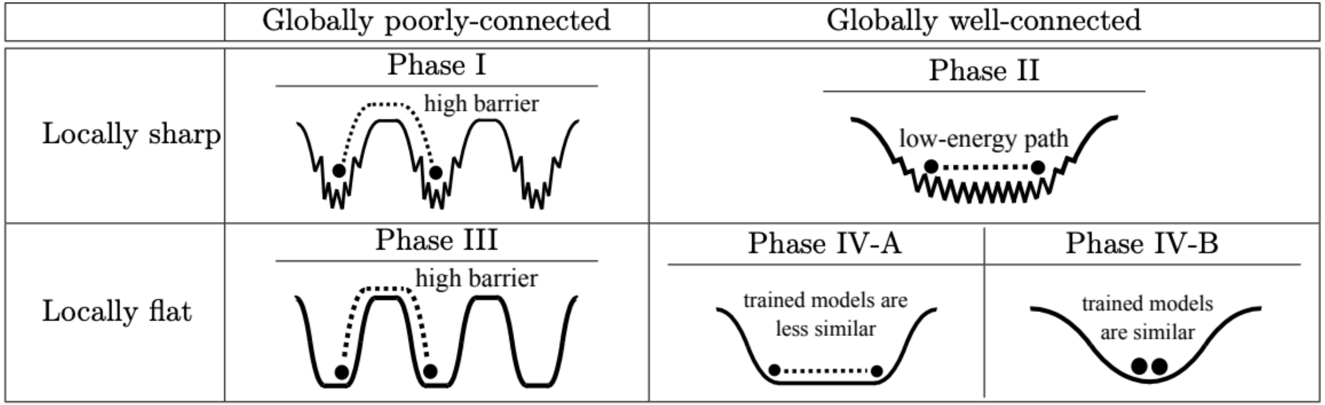
Robustness, a priori

- Goal - train a model to be more robust to model perturbations - more generalizable
  - Input perturbations = noise and lost channels
  - Weight perturbations = bit flips due to radiation



Metrics

Mode connectivity: barriers between converged models

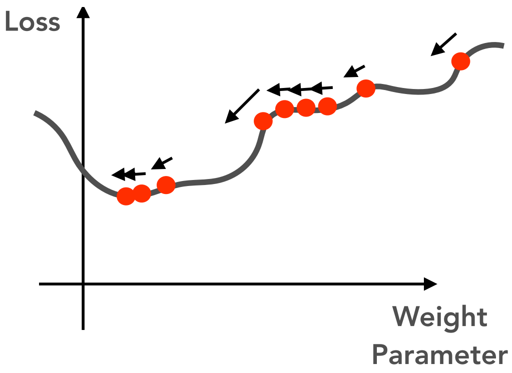


Hessian Trace:
Local smoothness

CKA similarity: similarity across trained models

Learning = optimization

learning as *optimization*



- Add additional loss terms to improve robustness and generalizability

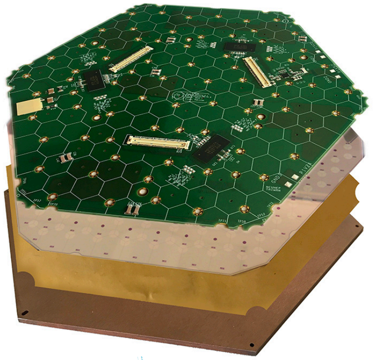
$$\lambda \| J(x) \|_F^2$$

Jacobian regularization penalizes large output impacts due to input perturbations

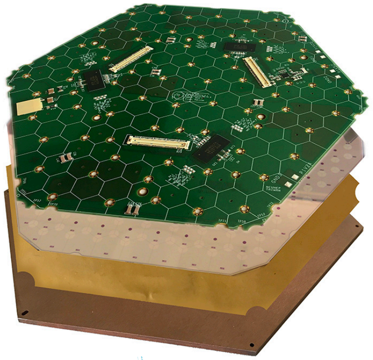
$$\lambda \| W^T W - I \|_F$$

Lipschitz regularization encourages weight orthogonality for loss landscape smoothness

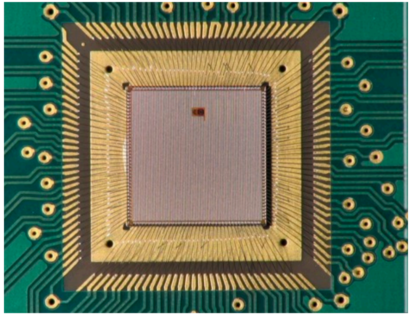
ASIC autoencoder



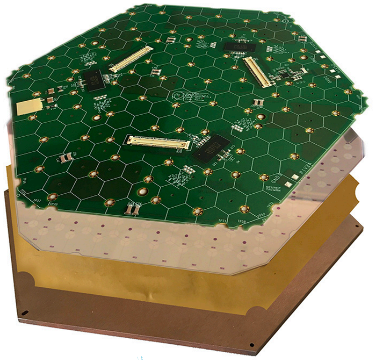
ASIC autoencoder



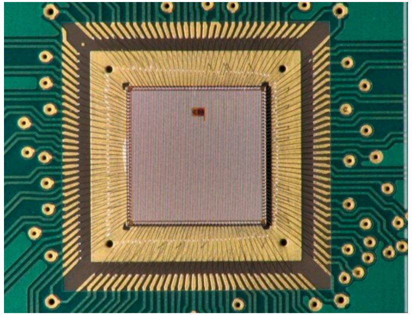
65 nm LP CMOS



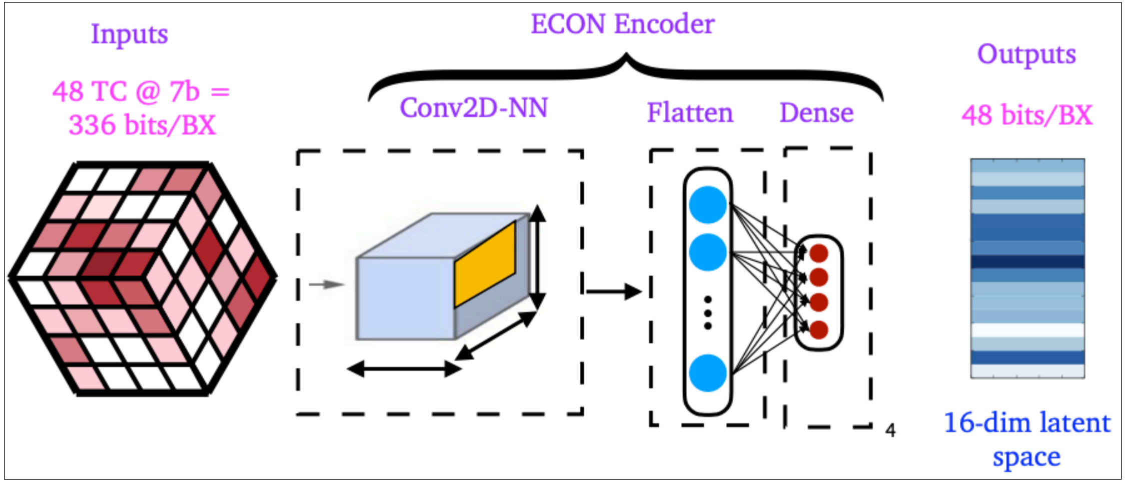
ASIC autoencoder



65 nm LP CMOS

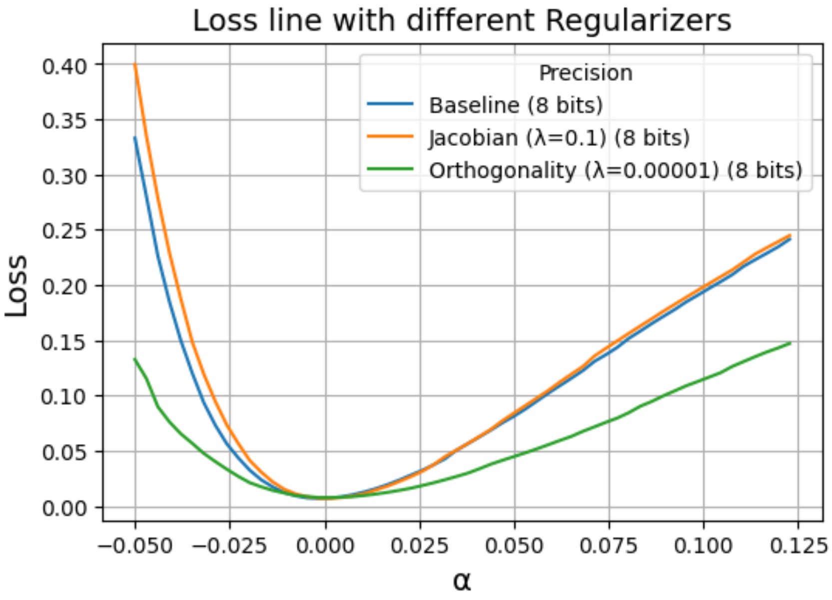


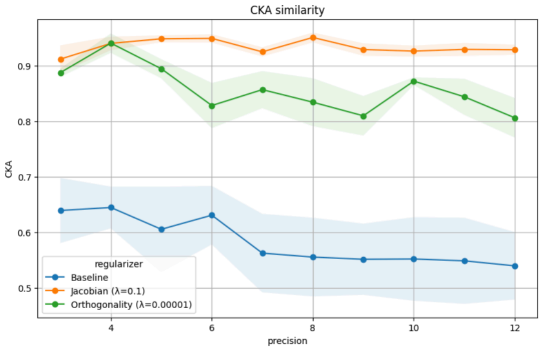
Powerful, Flexible, Adaptable Data Compression



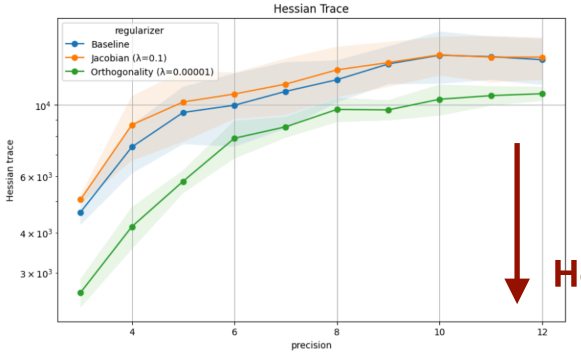
Robustness

Credit: T. Baldi (Santa Anna Pisa, FNAL internship), J. Campos (FNAL), et al



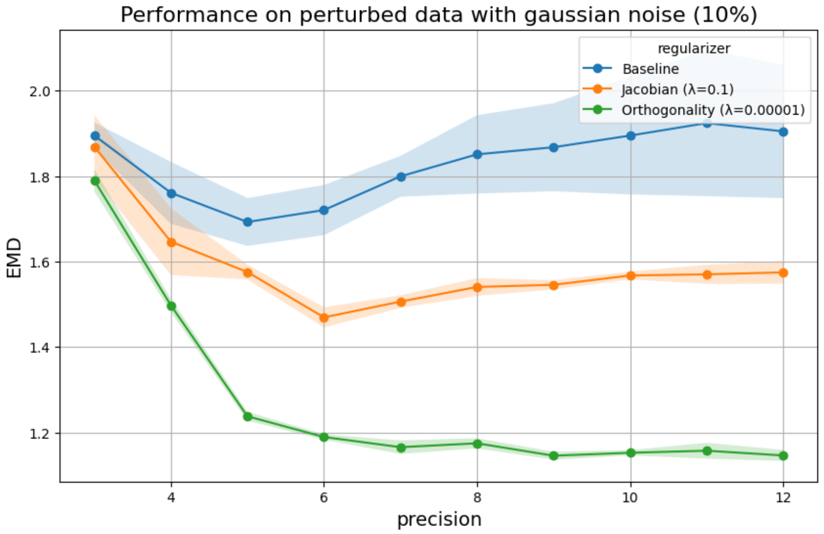


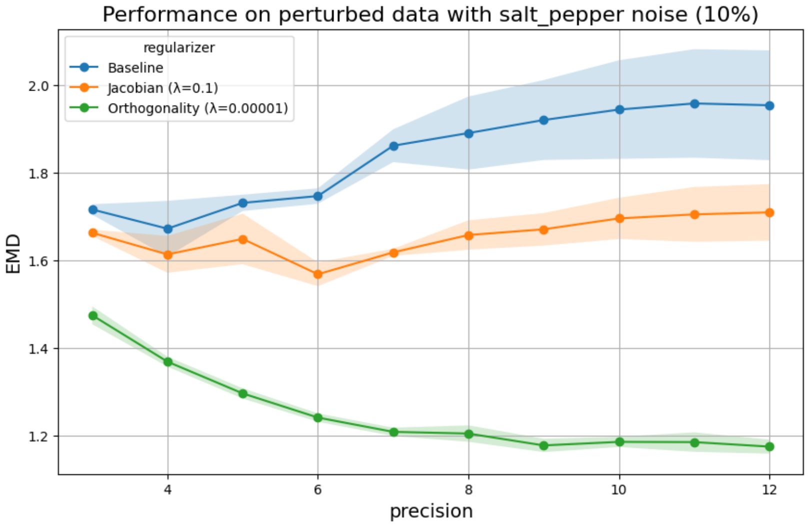
CKA similarity



Hessian Trace

Robustness





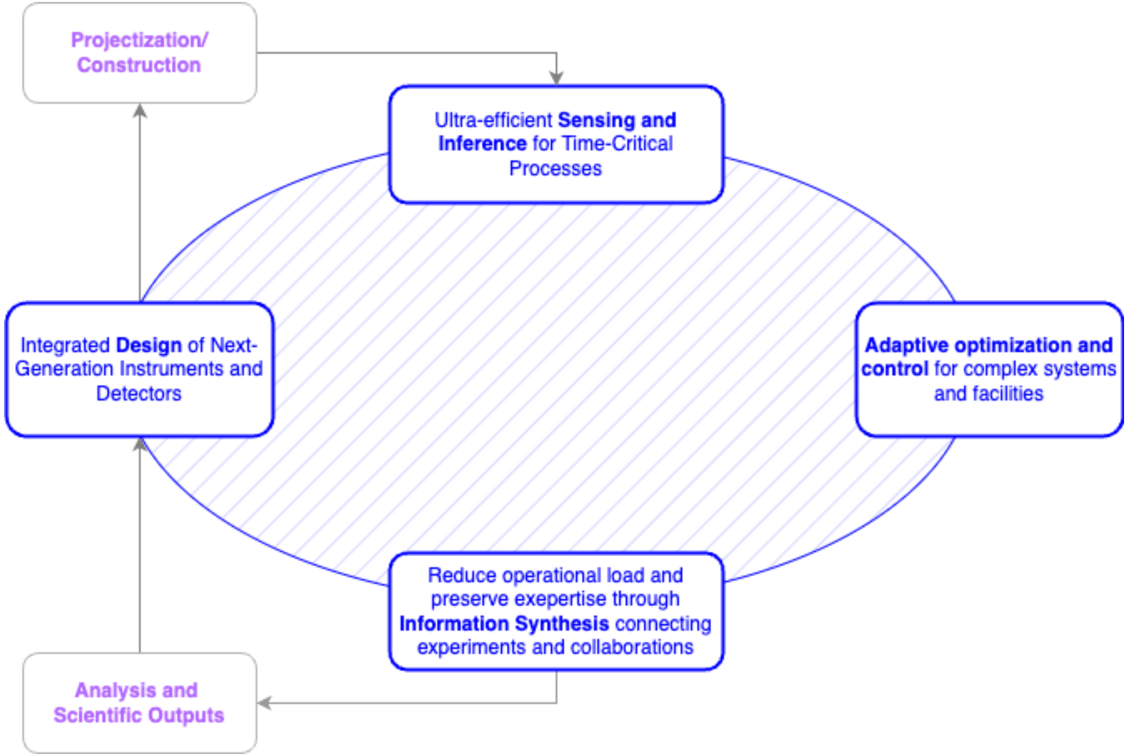
Better

Lipschitz regularization seems quite promising as a way make more robust edge NNs
Feeds into the automation loop and provides more system stability

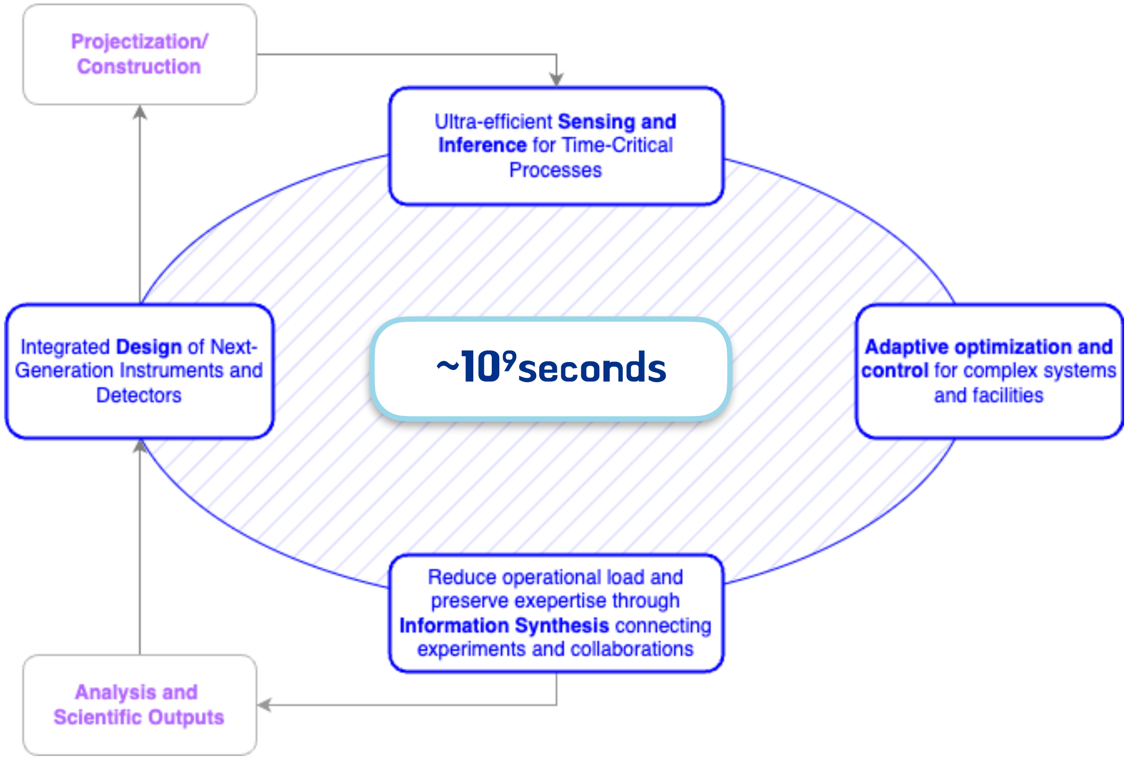
Outline

- Fast and Slow
- Fast ML and hardware codesign
- Fast and Slow ML together
- **Slow ML & Real-Time**

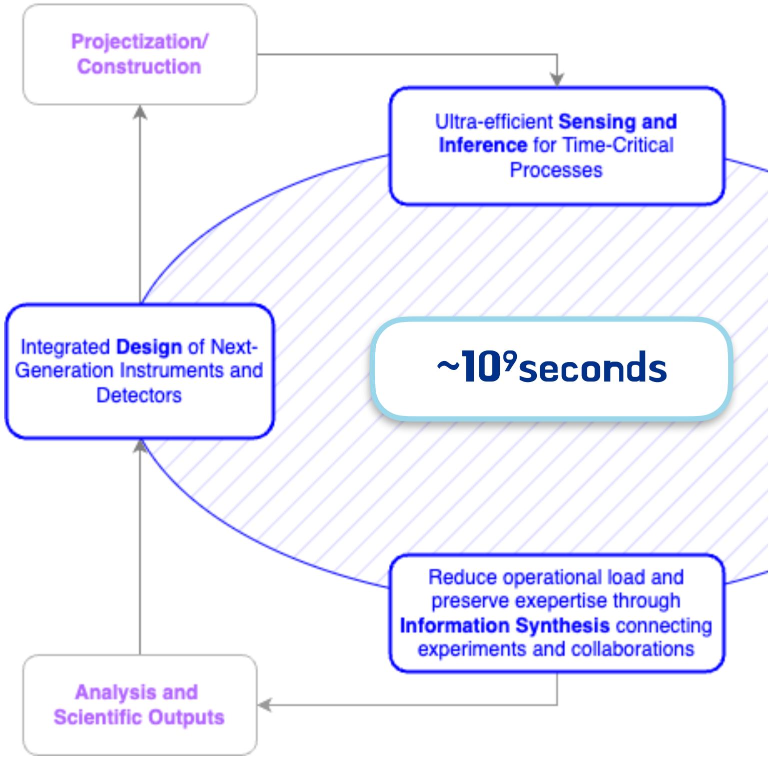
Life cycle of instruments, detectors, facilities

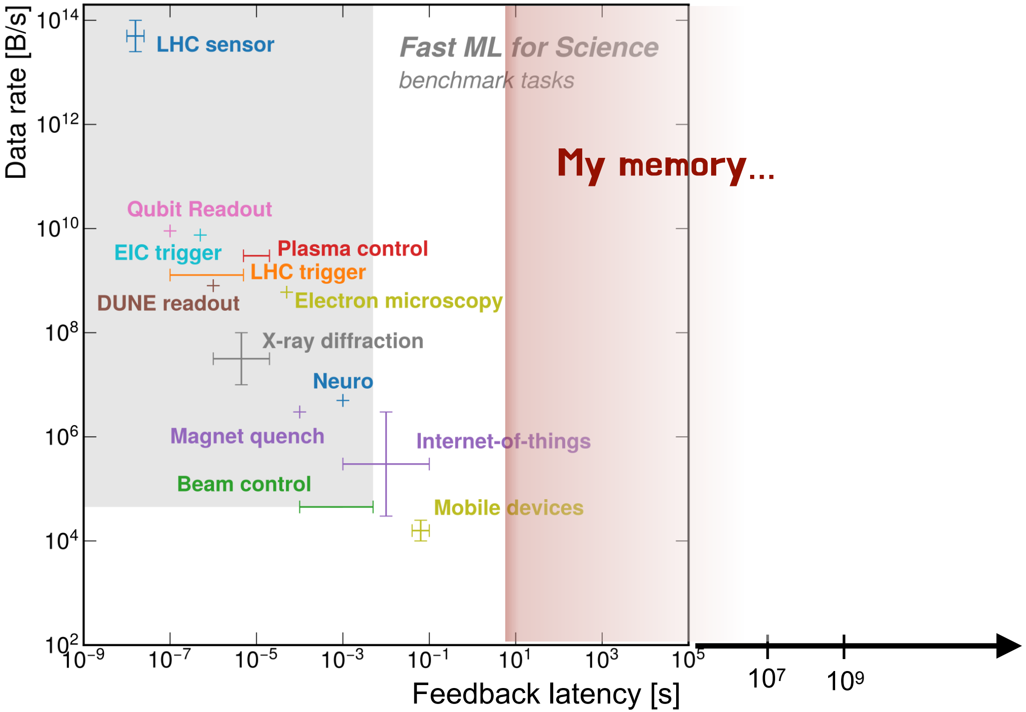


Life cycle of instruments, detectors, facilities



Life cycle of instruments, detectors, facilities





o1

n.b. GPT-4o got
this wrong

How much energy does a muon lose in 5m of iron?

Thought for 10 seconds

To determine how much energy a muon loses while traversing 5 meters of iron, we need to calculate the energy loss per unit distance and then multiply it by the total distance traveled.

Step 1: Understanding Muon Energy Loss Mechanisms

Muons, like other charged particles, lose energy in matter primarily through:

- **Ionization and Excitation**: Interacting with electrons in the material, causing ionization or excitation.
- **Radiative Processes**: At higher energies, processes like bremsstrahlung, pair production, and photonuclear interactions become significant.

For this calculation, we'll focus on ionization energy loss, which is dominant for muon energies up to several hundred GeV.

Step 2: Using the Bethe-Bloch Formula

The Bethe-Bloch formula describes the mean energy loss per unit path length ($\frac{dE}{dx}$) of charged particles as they pass through matter. For muons in iron, the mass stopping power (mean energy loss per unit mass thickness) is approximately:

$$\left(\frac{dE}{dx}\right)_{\text{mass}} \approx 1.5\,\text{MeV} \cdot \text{cm}^2/\text{g}$$

Fermilab

o1

Step 3: Calculating Energy Loss Per Unit Length

First, we need to convert the mass stopping power to energy loss per unit length using the density of iron:

- **Density of Iron (ρ)**: Approximately $7.874\,\mathrm{g/cm^3}$

Now, calculate the energy loss per centimeter:

$$\left(\frac{dE}{dx}\right)_{\text{length}} = \left(\frac{dE}{dx}\right)_{\text{mass}} \times \rho = 1.5\,\mathrm{MeV\cdot cm^2/g} \times 7.874\,\mathrm{g/cm^3} = 11.811\,\mathrm{MeV/cm}$$

Step 4: Calculating Total Energy Loss Over 5 Meters

Convert 5 meters to centimeters:

$$5\,\text{meters} = 500\,\text{cm}$$

Now, multiply the energy loss per centimeter by the total distance:

$$\Delta E = \left(\frac{dE}{dx}\right)_{\text{length}} \times \text{Distance} = 11.811\,\mathrm{MeV/cm} \times 500\,\mathrm{cm} = 5905.5\,\mathrm{MeV}$$

Step 5: Converting to GeV

Since $1\,\mathrm{GeV} = 1000\,\mathrm{MeV}$:

$$\Delta E = \frac{5905.5\,\mathrm{MeV}}{1000} = 5.9055\,\mathrm{GeV}$$

Answer:

Approximately **5.9 GeV** of energy is lost by the muon while traversing 5 meters of iron.

Yesterday

Person 1 4:00 AM

Hi, I'm wondering if anyone know how to check the generator versions of the UL samples?

image.png

CA: L65 Please fix 2.4.2 beyond the layout. Are you sure the QCD is produced with this MG version? I remember most UL samples were regenerated with MG 2.6.X. Which scheme (4FS or 5FS)? Which jet merging (MLM or FXFX)? Please provide.
CA: L67 Please also include the version of MG and fix words beyond the layout.
CA: L73 Also the version of MG here and everywhere.
CA: L75 EW seems jargon and not easy to recognize for readers.
CA: L82, L84 Add versions of each generator used in your signals.

Person 2 7:53 AM

Person 3 told me the answer to this at some point

I think it is actually two different versions

Person 1 7:58 AM

I looked into it and it's madgraph/madgraph MLM

I'm not really sure how to write this up

in the paper

Person 1 8:17 AM

@ *Person 3* Do you happen to know the madgraph versions for the UL QCD samples!!

Person 3 8:32 AM

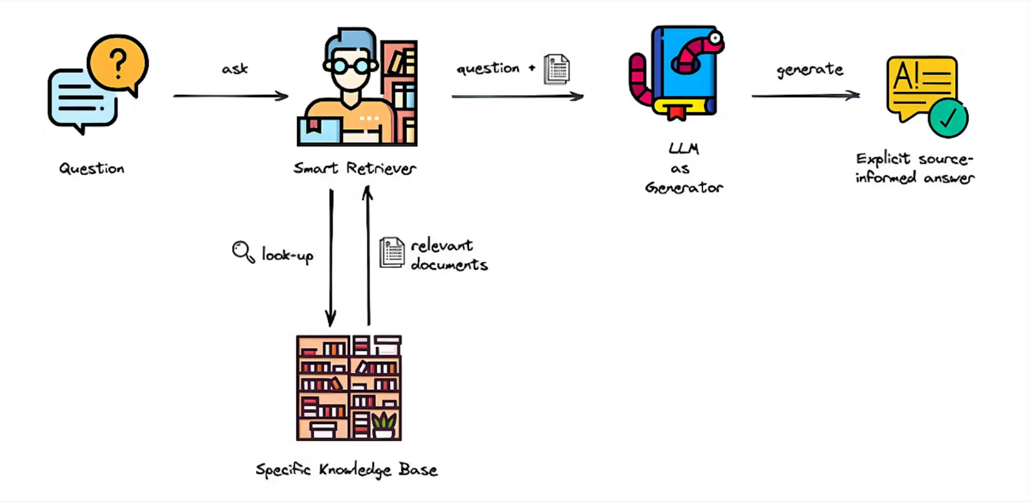
2.6.1 and 2.6.5

https://twiki.cern.ch/twiki/bin/viewauth/CMS/CitationsForGenerators

Fermilab

chATLAS

An AI Assistant for the ATLAS Collaboration



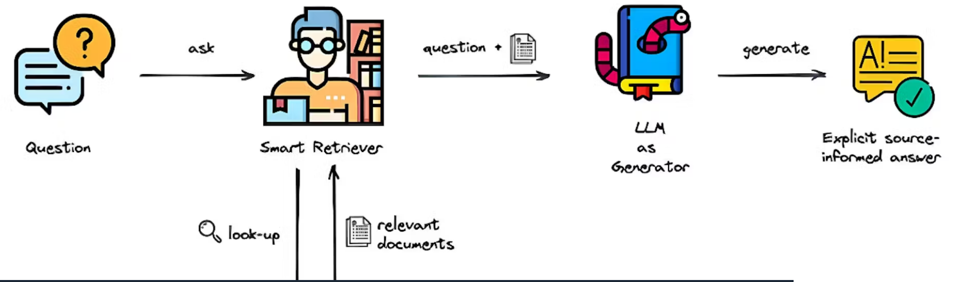
LLM + RAG (Retrieval-Augmented Generation):
Goal: Ask questions to an LLM about private information without retraining

Fermilab

https://indico.cern.ch/event/1395528/contributions/5877957/attachments/2833825/4951736/chATLAS%20IML%20Meeting%20(1).pdf

chATLAS

An AI Assistant for the ATLAS Collaboration



LLM + RAG (Re

Generation):

Goal: Ask questi

private informat

How do I calculate the z0 of a track with respect to a vertex?

You can calculate the z0 of a track with respect to a vertex by using the following code snippet:

```
xAOD::TrackParticle trk;
xAOD::Vertex vtx;
float delta_z0 = fabs(trk.z0() + trk.vz() - vtx->z());
```

where trk.vz() represents the point of reference for the z0 calculation (in this case, the beamspot position along the z axis).

Fermilab

https://indico.cern.ch/event/1395528/contributions/5877957/attachments/2833825/4951736/chATLAS%20IML%20Meeting%20(1).pdf

chATLAS

An AI Assistant for the ATLAS Collaboration

LLM + RAG (Re… Generation):
Goal: Ask questi… private informat…

How do I calculate the z0 of a track wit…

Yo…

xA…

xA…

flo…

wh…

…ppet:

…mspot position along the z axis).

Expanding footprint: + more TWikis +CDS

RL

Significant expansion of collected dataset:
- Expand initial scrape of TWikis. Expand from 1.5k to ~22k
- Add 1.2k ATLAS published papers from CDS
- META's Neural Optical Understanding for Academic Documents (NOUGAT) converts PDFs to markdown preserving math symbols and formulae
  - **DB with CDS docs in alpha testing**

Room to grow. Some low hanging fruit:
- TWikis: Do not include software docs (yet)
- CDS: +66k records, 36k internal. Room to (easily) grow

Latest scrapers are here (2FA blocked)

IML Meeting, 9th April 2024 9

Task \ DB	Twiki	ATLAS Software Docs	E-group Archive	Indico Meetings	PDF Plots
Scrape	2k+	500+	10k+	~1k+	~1k+
Convert					
Chunk & Embed					

Not yet started
In Progress
Complete

Fermilab

In closing…

Summary

- **Fast and Slow**
  – Resolving 18 orders of magnitude in time (!) to understand where AI could be deployed to improve sensing, automation, control, and knowledge synthesis
- **Fast ML and hardware codesign**
  – Embedded ML can be used to access new information, reduce biases, and develop adaptive systems; accessible tools like hls4ml speed up algorithm/hardware design
- **Fast and Slow ML together**
  – More powerful autonomous instruments enabled through digital twins - physically-coupled fast simulators - and more robust real-time controllers
- **Slow ML & Real-Time**
  – LLMs + RAG and emerging agentic workflows can aid in reducing operational load by preserving and synthesizing collective expertise

Ruminations

- "Real-time" is a curious term
- Humans are not particularly good at pattern recognition over multiple timescales - this also requires us to capture the right data
- AI is a disruptive technology - I'm hopeful that we can harness it to accelerate our particle physics ambitions
- There are many connections to other scientific domains enabling discovery in many (sub-)fields

time, its illusiveness across multiple timescales in particle physics with a message of hope - Imagen3