THE 2023 INTERNATIONAL WORKSHOP ON THE HIGH ENERGY CIRCULAR ELECTRON-POSITRON COLLIDER (CEPC)

Using FPGA to accelerate machine learning inference for particle physics

Yutao Zhang， Yaodong Cheng, **Haibo Li**

Computing center, IHEP, CAS

2023-10-24

中国科学院高能物理研究所
Institute of High Energy Physics, Chinese Academy of Sciences

Outline

- Background and motivation
- Brief introduction to FPGA AI programming
- A case of ParticleNet implementation on FPGA
- Summary

Background – AI explosion

Open AI (2023)

Example of GPT-4 visual input:

User — What is funny about this image? Describe it panel by panel.

Source: https://www.reddit.com/r/hmmm/comments/ubab5v/hmmm/

GPT-4 — The image shows a package for a "Lightning Cable" adapter with three panels.

Panel 1: A smartphone with a VGA connector (a large, blue, 15-pin connector typically used for computer monitors) plugged into its charging port.

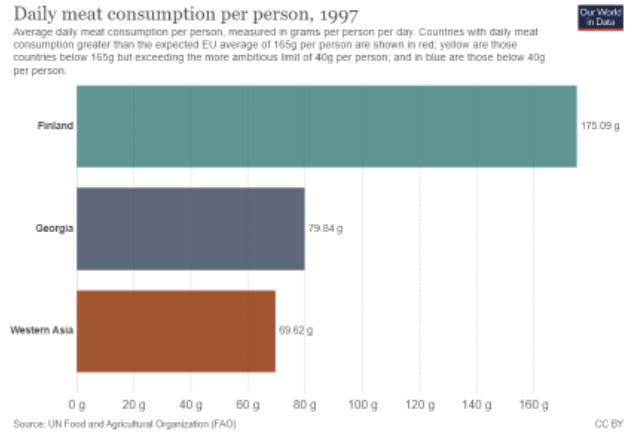
Panel 2: The package for the "Lightning Cable" adapter with a picture of a VGA connector on it.

Panel 3: A close-up of the VGA connector with a small Lightning connector (used for charging iPhones and other Apple devices) at the end.

The humor in this image comes from the absurdity of plugging a large, outdated VGA connector into a small, modern smartphone charging port.

GPT-4 visual input example, Chart Reasoning:

User — What is the sum of average daily meat consumption for Georgia and Western Asia? Provide a step-by-step reasoning before providing your answer.



GPT-4 — To find the sum of average daily meat consumption for Georgia and Western Asia, we will follow these steps:

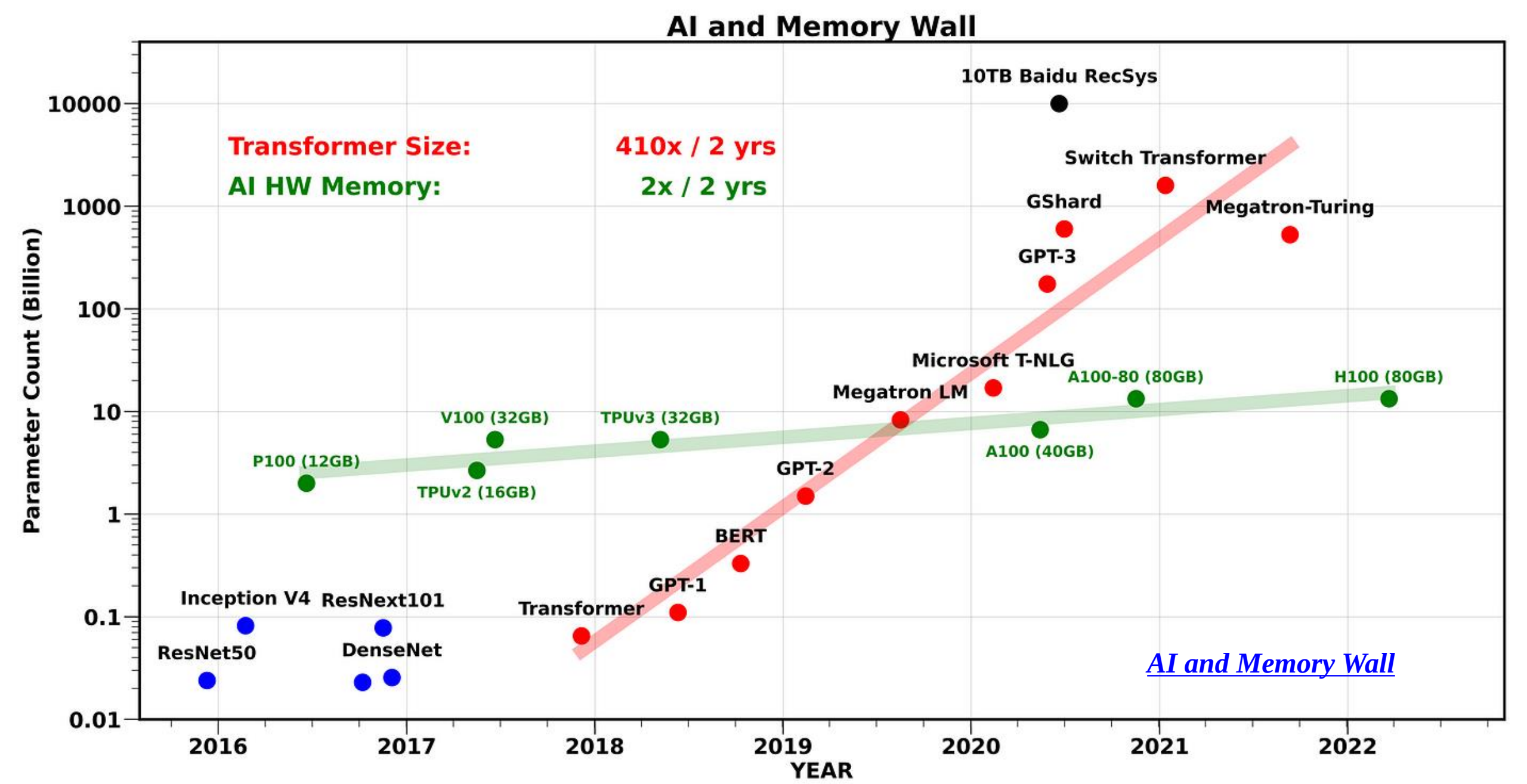
1. Identify the average daily meat consumption for Georgia.
2. Identify the average daily meat consumption for Western Asia.
3. Add the values from steps 1 and 2.

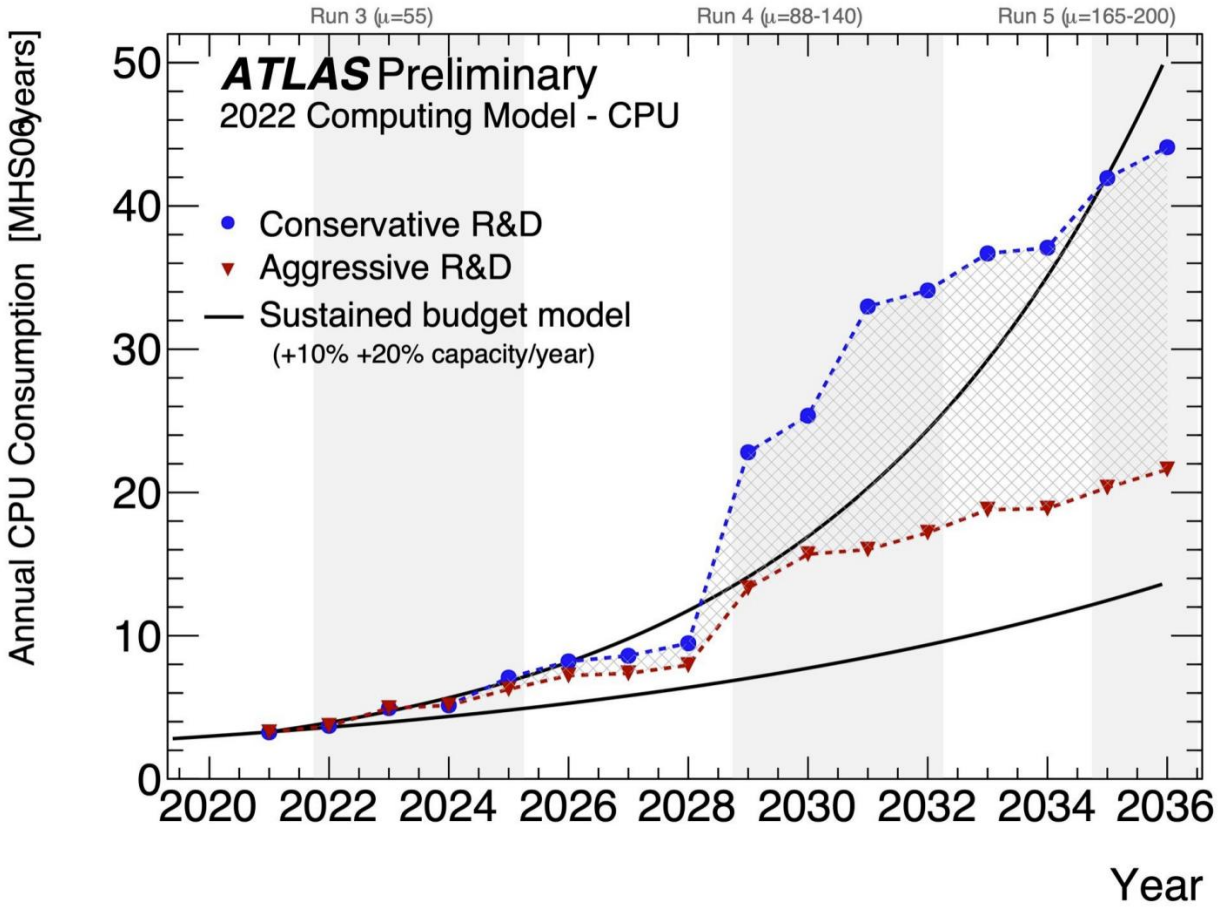
Step 1: From the provided chart, we see that the average daily meat consumption for Georgia is 79.84 grams per person per day.
Step 2: From the chart, we see that the average daily meat consumption for Western Asia is 69.62 grams per person per day.
Step 3: Add the values from steps 1 and 2.
79.84 g (Georgia) + 69.62 g (Western Asia) = 149.46 g

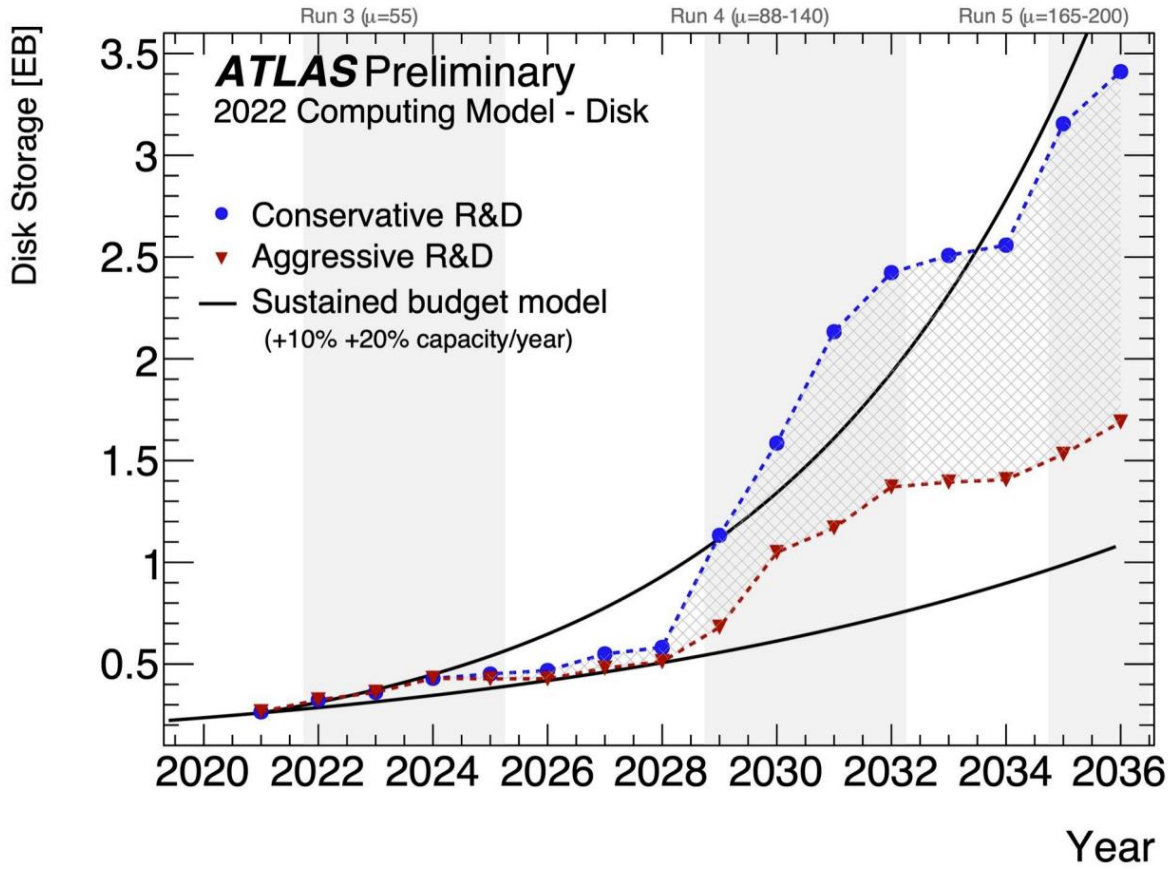
So, the sum of average daily meat consumption for Georgia and Western Asia is 149.46 grams per person per day.

Train (GPT-3):

- **285,000 CPU cores**
- **10,000 GPUs**
- **400 Gb/s network**
- **Several weeks**
- **Trained on ~25 km high book of text**


AI and Memory Wall
Parameter Count (Billion)
10000
1000
100
10
1
0.1
0.01
2016
2017
2018
2019
2020
2021
2022
YEAR
Transformer Size:
410x / 2 yrs
AI HW Memory:
2x / 2 yrs
10TB Baidu RecSys
Switch Transformer
GShard
Megatron-Turing
GPT-3
Microsoft T-NLG
Megatron LM
A100-80 (80GB)
H100 (80GB)
V100 (32GB)
TPUv3 (32GB)
P100 (12GB)
TPUv2 (16GB)
A100 (40GB)
GPT-2
BERT
GPT-1
Transformer
Inception V4
ResNext101
ResNet50
DenseNet
AI and Memory Wall





ATLAS Software and Computing HL-LHC Roadmap

… flat computing budget

Need innovation and new techniques to maintain physics research while staying within throughout requirements!

Todays algorithms will not be sustainable in future HEP experiments such as HL-LHC, CEPC, …

→Need modern Machine Learning to become

Faster

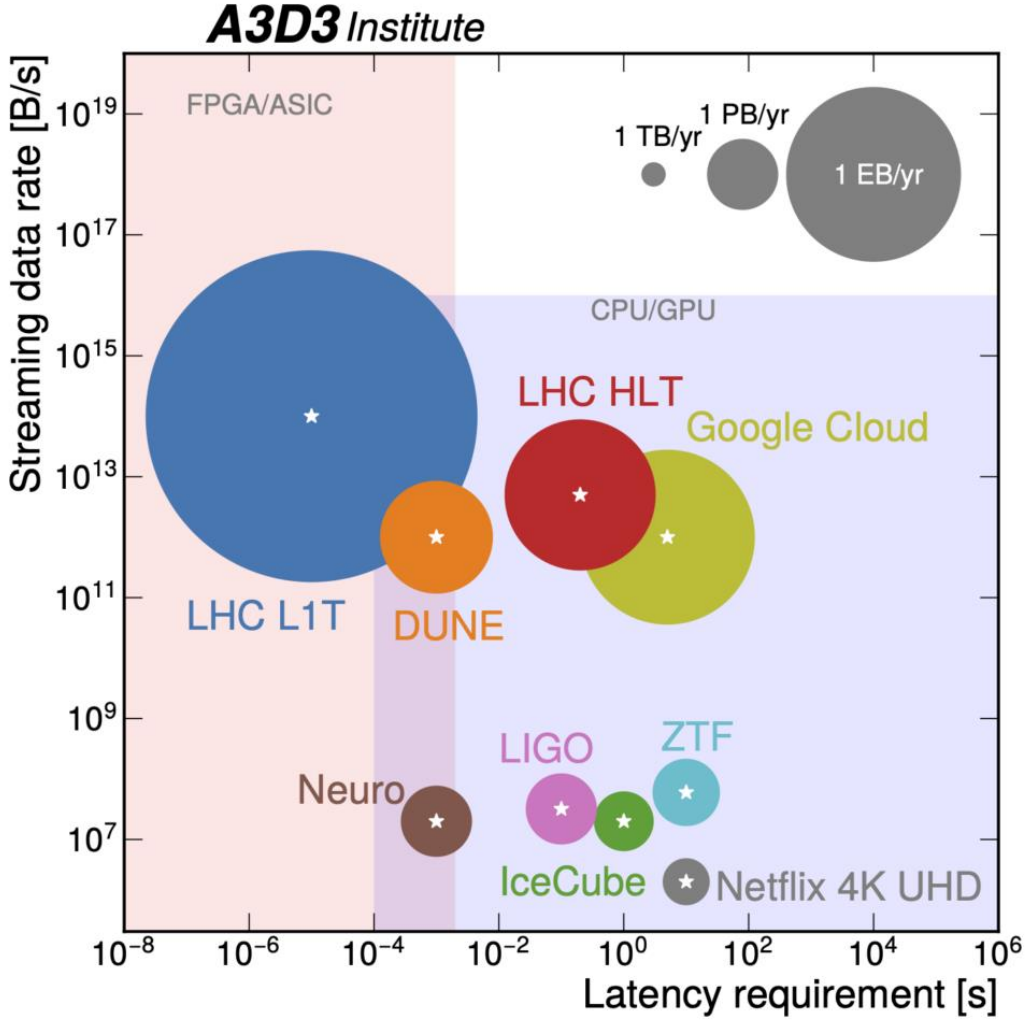
Better

and do more

More complex architecture to deal with increased data complexity

AI Algorithms Acceleration

- AI has emerged as a solution to efficiently analyze these massive data sets
- GPUs and FPGAs allow AI algorithms to be greatly accelerated
- The combination of AI and these processors is leading to a revolution in the way we analyze data, minimizing the time needed to perform the most advanced of analyses
- A3D3: Accelerated AI Algorithms for Data-Driven Discovery
  - high energy physics, multi-messenger astrophysics, and systems neuroscience



A3D3 institue

AI and FPGA application in Particle physics

✓ Requirements

Low Latency

- Strictly limited by collisions occurring every 25ns

Low resource usage

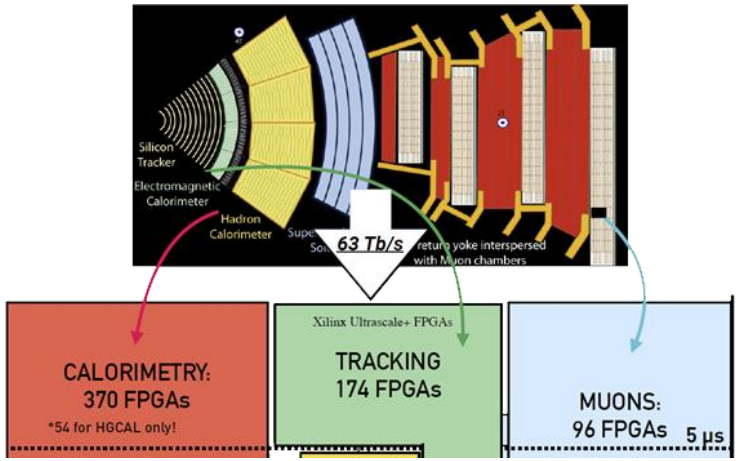
- Several algorithms in parallel on single device

High throughput

- System processing ~5% of total internet traffic

✓ AI and FPGA has widely applied in trigger and data processing

- Trigger, ml for data compression
- Simulation, ml for data generation
- Data reconstruction and analysis



✓ Why are FPGAs used?

Low Latency

- Resource parallelism and pipeline parallelism! High bandwidth

Latency deterministic

- CPU/GPU processing randomness, FPGA repeatable and predictable latency

Power efficient

- FPGAs up to ~10x more power efficient than GPU

How are FPGAs programmed?

Hardware Description Languages (HDLs)

HDLs are programming languages which describe electronic circuits

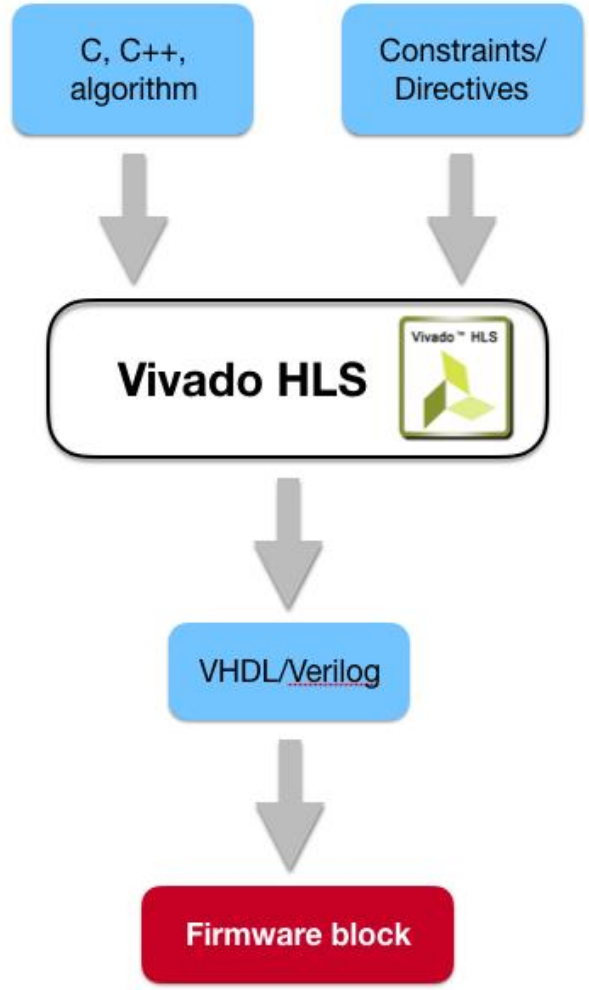
High Level Synthesis (HLS)

Compile from C/C++ to VHDL

Pre-processor directives and constraints used to optimize the design

Drastic decrease in firmware development time!

Currently we mainly use Xilinx Vivado HLS [*]



[*] https://www.xilinx.com/support/documentation/sw_manuals/xilinx2020_1/ug902-vivado-high-level-synthesis.pdf

AI Programming on FPGA

TensorFlow ONNX PYTORCH QKERAS

Model
(quantized/Pruned)

hls4ml

1. Convert model into internal representation

2. Write HLS project targeting specified backend

3. Run emulation

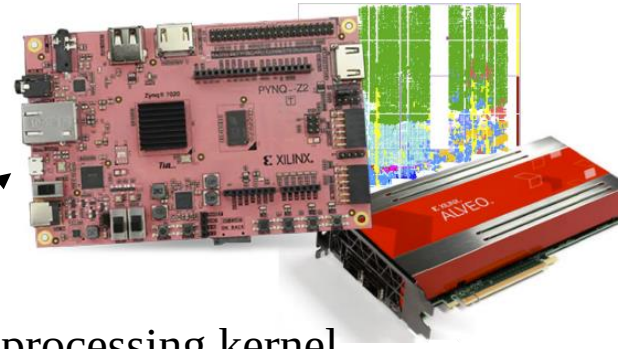
4. Run synthesis

Vivado/Vitis best supported
Intel Quartus
Intel One API
Mentor Catapult HLS
available soon

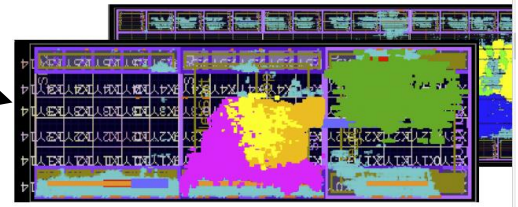
Vivado™ HLS

Mentor A Siemens Business

Intel HLS COMPILER



Co-processing kernel
(Xilinx accelerators and/SoCs)



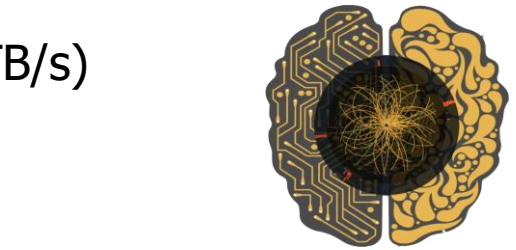
FPGA custom designs
(eg trigger algorithm)

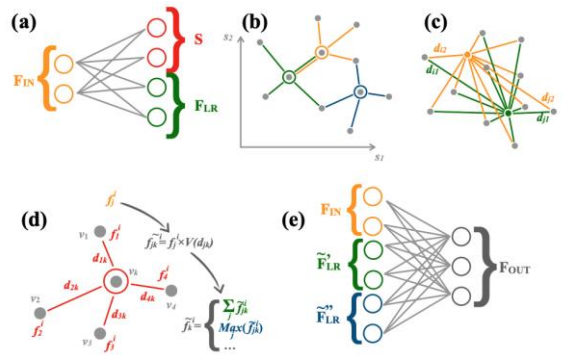
ASIC/NPU

hls4ml

- **Hls4ml**: high level synthesis for machine learning
- **hls4ml** is a package for translating neural networks to FPGA firmware for inference with extremely low latency on FPGAs
  - ✓ https://github.com/hls-fpga-machine-learning/hls4ml
  - ✓ https://fastmachinelearning.org/hls4ml/
  - ✓ pip install hls4ml
- **hls4ml** origins: triggering at (HL-)LHC
  - Extreme collision frequency of 40 MHz → extreme data rates O(100 TB/s)
  - Most collision "events" don't produce interesting physics
  - "Triggering" = filter events to reduce data rates to manageable levels
- **hls4ml** community is very active now
- Tutorial

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





Efficient NN design for FPGAs

FPGAs provide huge flexibility

Performance depends on how well you take advantage of this

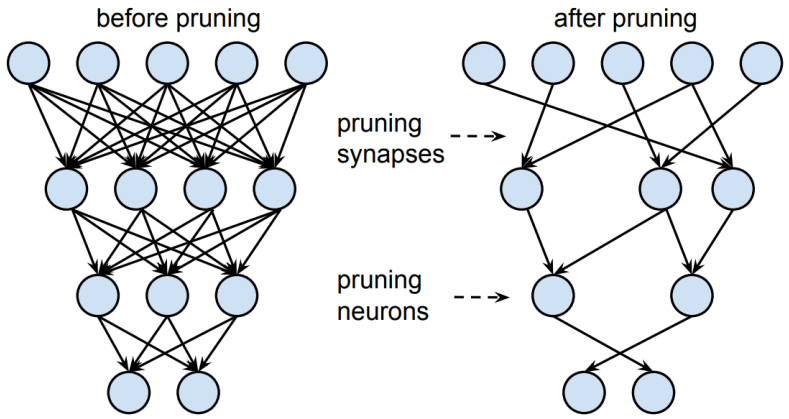
Constraints:

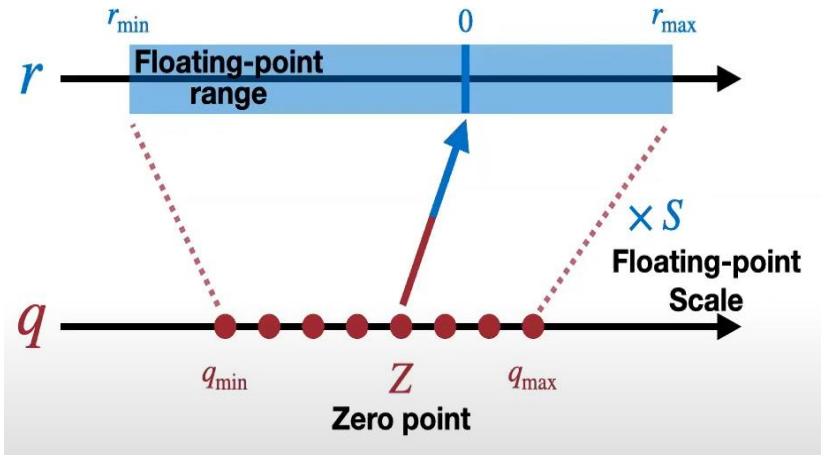
Input bandwidth

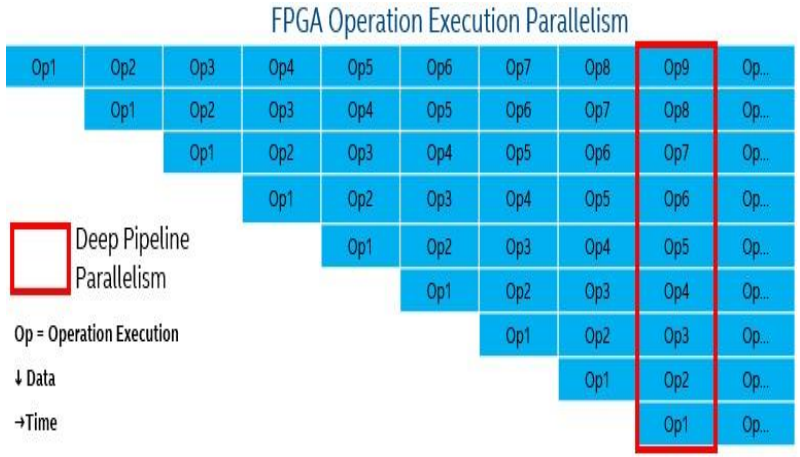
FPGA resources

Latency

Main methods to optimize the FPGA project



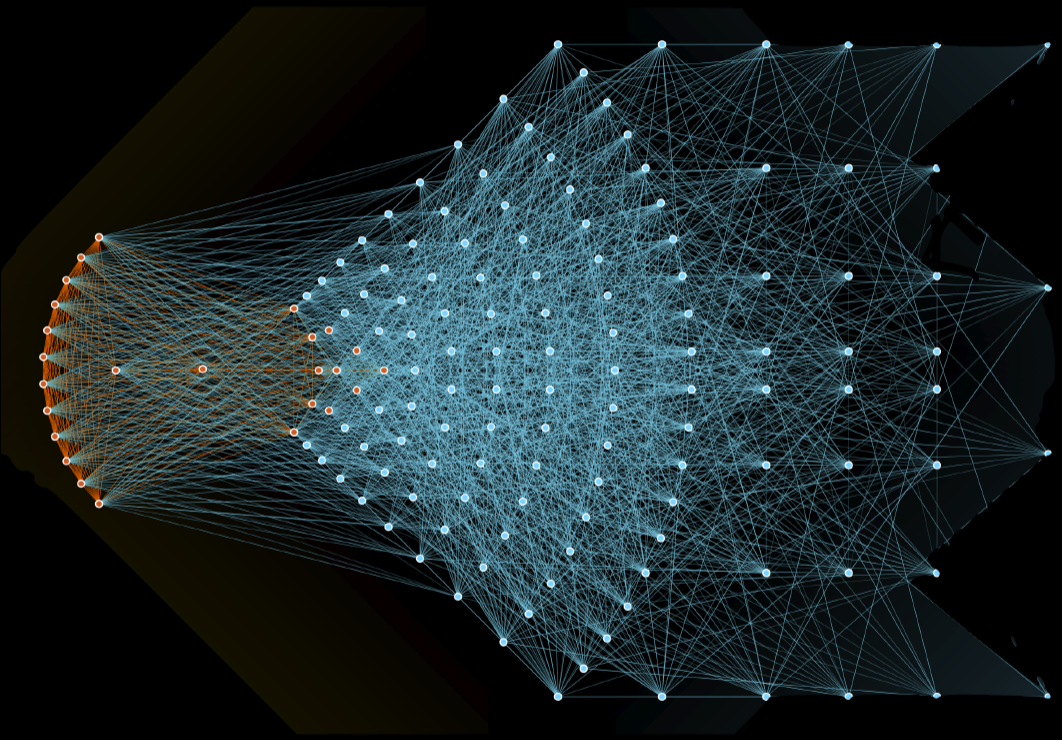




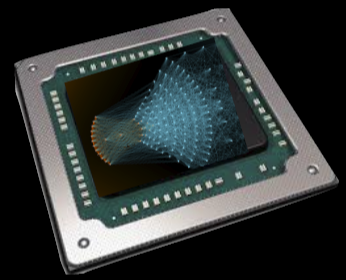
Pruning: reduce number of neurons

Quantization: reduce the precision of the calculations (inputs, weights, biases)

Parallelization : tune how much to parallelize to make the inference faster



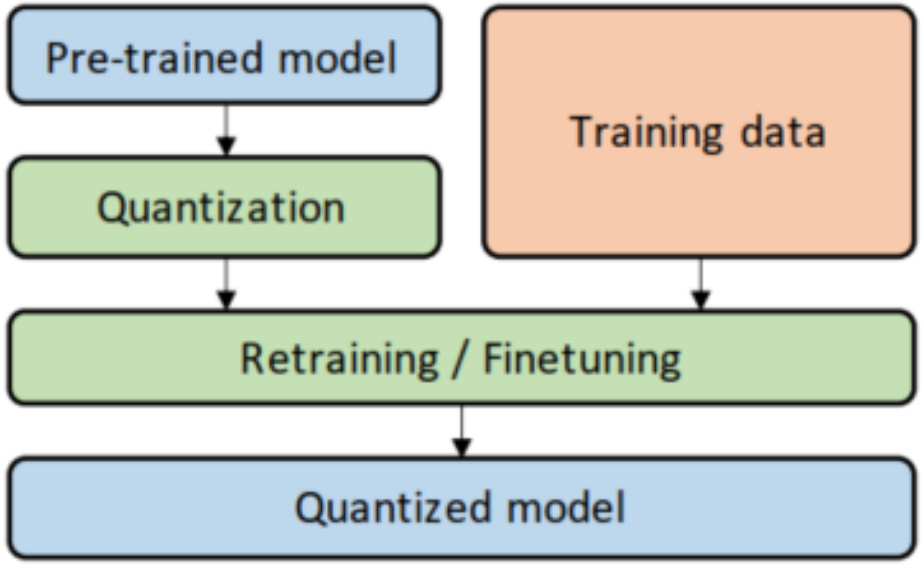
Ideally

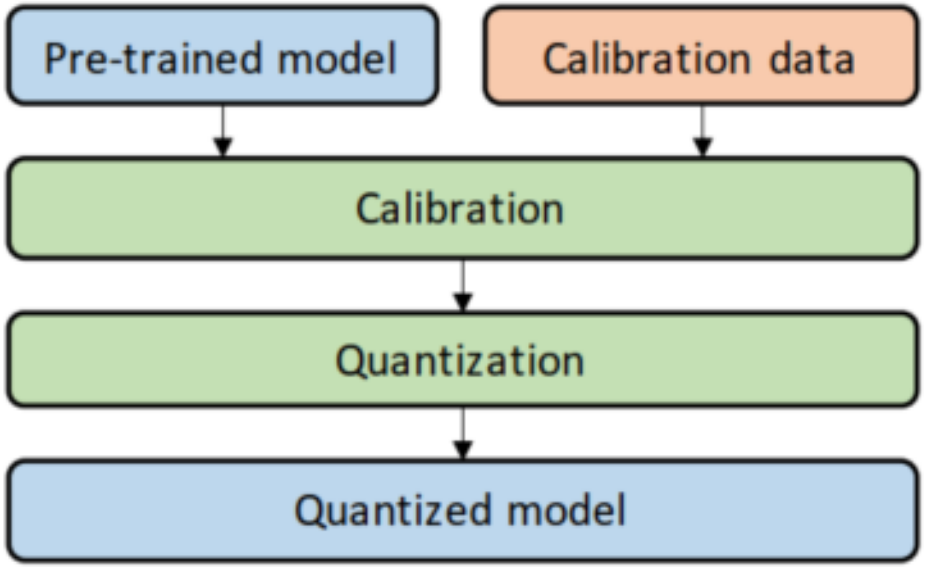


Reality

Quantization – QAT vs. PTQ

- Quantization is a common technique used to reduce model size, though it can sometimes result in reduced accuracy





QAT: Quantization Aware Training
The network is further trained for few epochs in a process called Fine-Tuning/Retraining.
Without/Less sacrificing accuracy

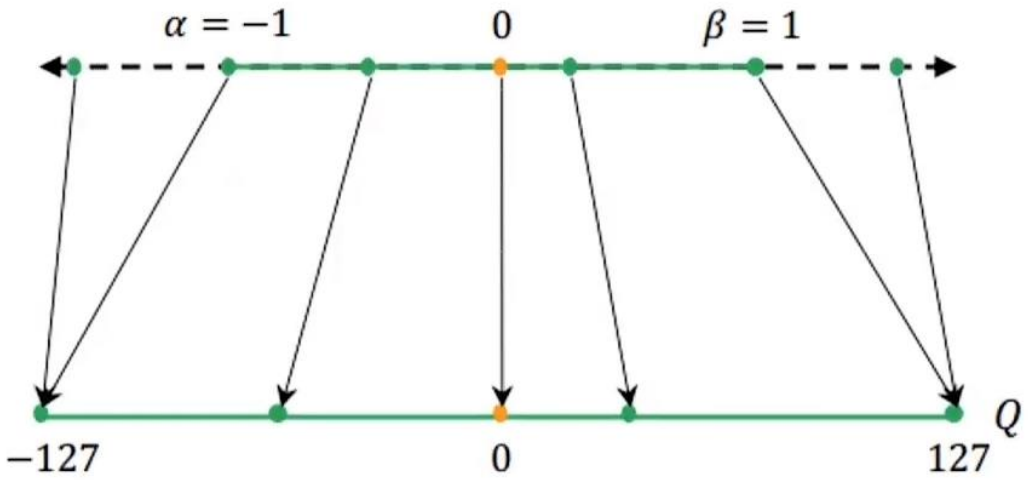
PTQ: Post-Training Quantization
The model's weights and activations are quantized from high precision to low precision, such as from FP32 to INT8
Does not consider for the loss of accuracy

Quantization – Symmetric vs. Asymmetric

Symmetric

Quantize: $Q = clamp(round(\frac{R_{float}}{S}))$

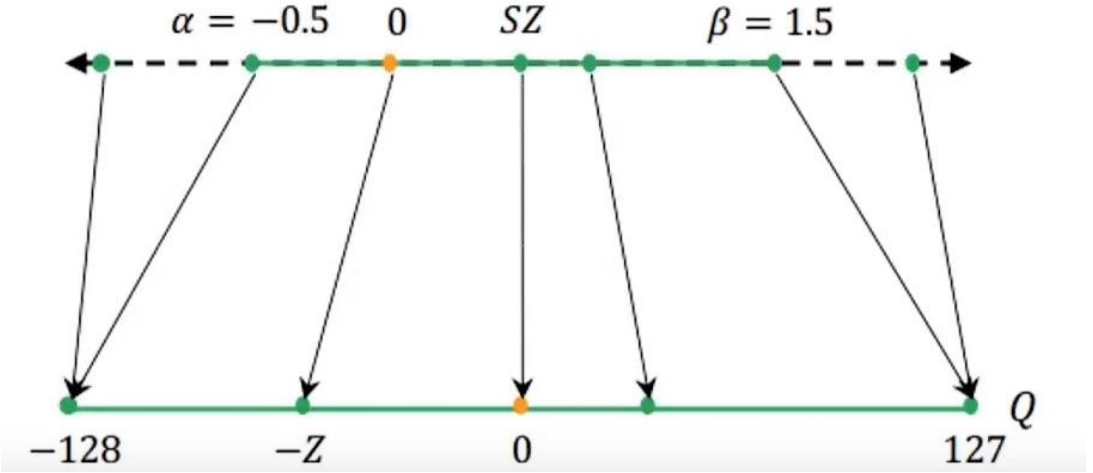
Dequantize: $R_{float} = S * Q$



Asymmetric

Quantize: $Q = clamp(round(\frac{R_{float}}{S}) - Z)$

Dequantize: $R_{float} = S * (Q + Z)$



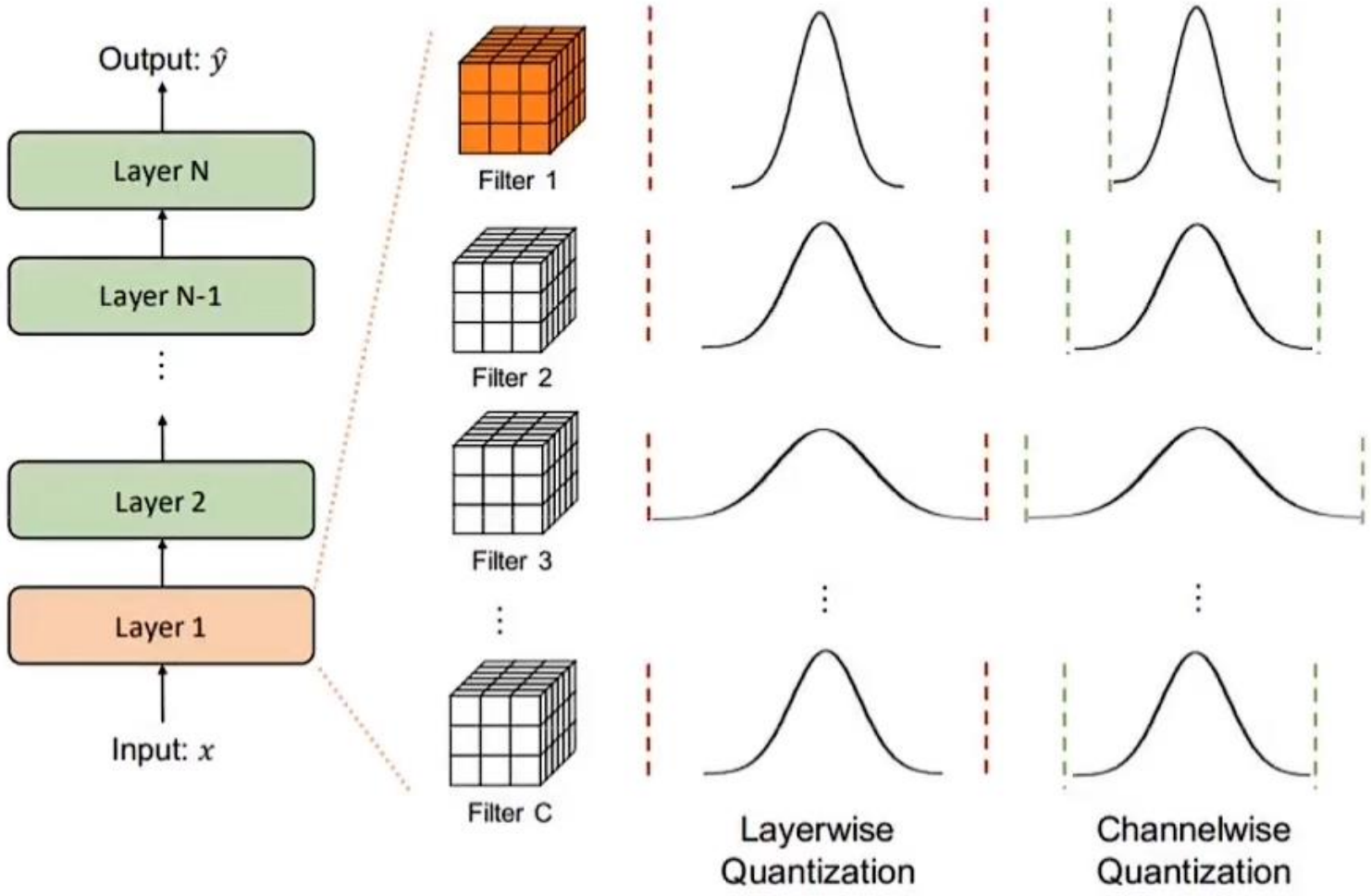
Quantization – Per tensor vs. Per channel

Per Tensor

The weights of each layer have the same $scale$ and $zero\_point$

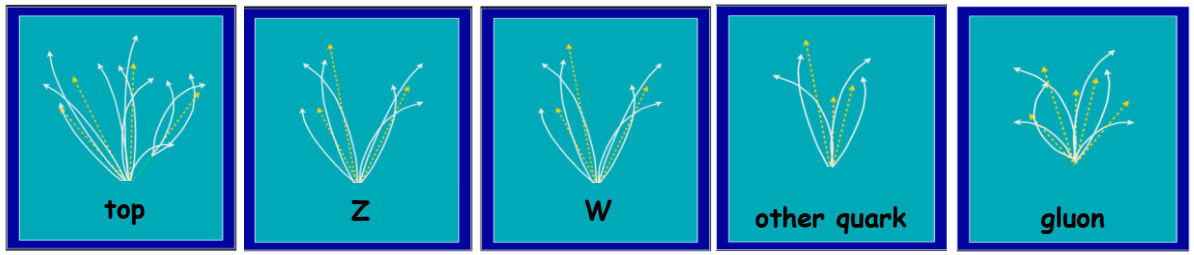
Per Channel

The weights of each channel have the same $scale$ and $zero\_point$



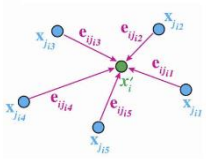
Physical case: Jet tagging

- Jet: collimated spray of hadrons initiated by energetic quarks or gluons, and they are ubiquitous at a hadron collider
- Jet tagging: identifying the hard scattering particle that initiates the jet, examples:
  - heavy flavor tagging (bottom/charm)
  - heavy resonance tagging (top/W/Z/Higgs)
  - quark/gluon discrimination
  - exotic jet tagging (displaced, 4-prong, …)

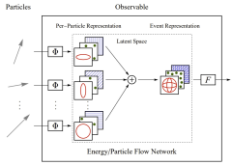


t→bW→bqq	**Z→qq**	**W→qq**	**q/g background**
3-prong jet	2-prong jet	2-prong jet	no substructure and/or mass ~ 0

- The rise of machine learning (ML) has brought lots of new progresses to jet tagging

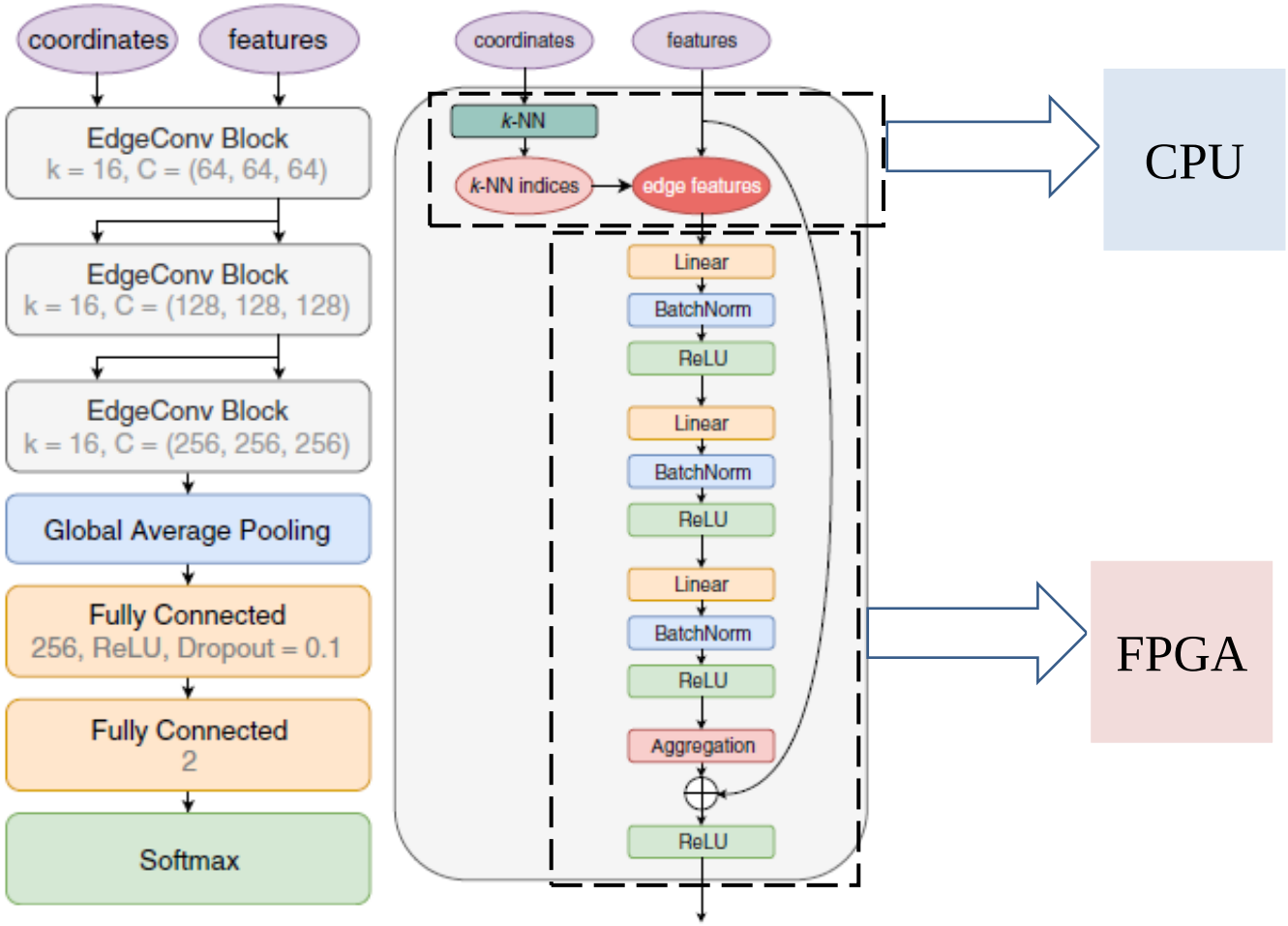


ParticleNet: jet tagging via particle clouds [arXiv:1902.08570]



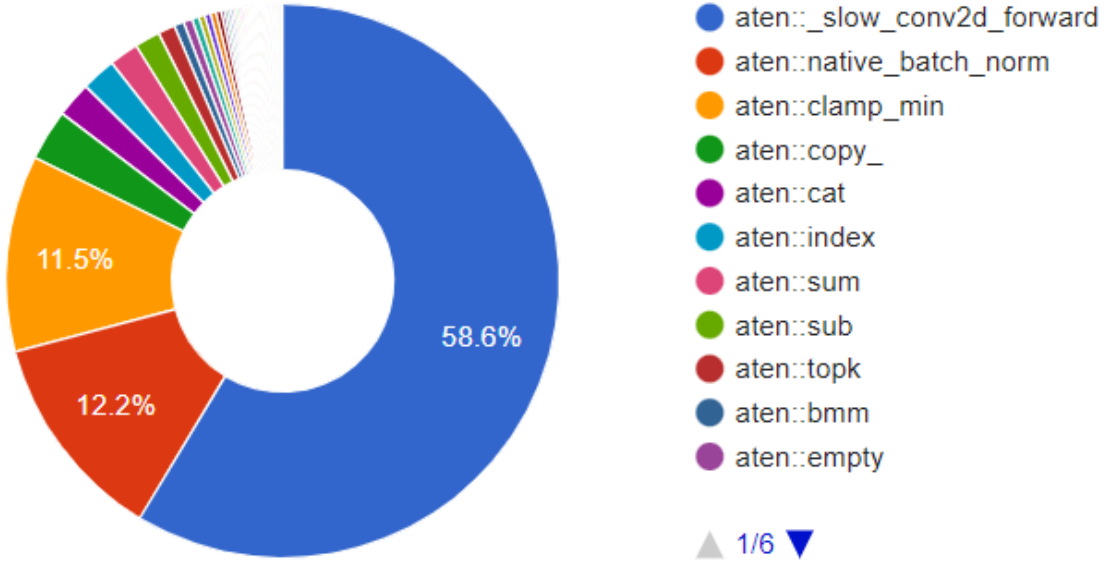
PFN: Particle Flow Network based on the Deep Sets [arXiv:1810.05165]

Workflow of ParticleNet



CPU

FPGA



Accounts for ~70% of the total time

✓ Using heterogenous solution with CPU and FPGA

Quantization of ParticleNet

- The quantization policy combined with Symmetric, Post-Training and Per tensor
- Symmetric: power-of-Two Quantization, $scale = 2^x$

	Parameters	ACC	Type
ParticleNet	**993KB**	**~93.9%**	**float32**
ParticleNet-quantization	**289KB**	**~93.3%**	**Int8**

$$ACC = (TP + TN)/(TP + TN + FP + FN)$$

```
correct:  0.933899998664856
正计算网络量化误差(SNR)，最后一层的误差应小于 0.1 以保证量化精度:
Analysing Graphwise Quantization Error(Phrase 1):: 100%| | 32/32 [00:00<00:00, 32.11it/s]
Analysing Graphwise Quantization Error(Phrase 2):: 100%| | 32/32 [00:01<00:00, 26.72it/s]
```

Quantization implementation

Convert floating point calculation to fixed point calculation

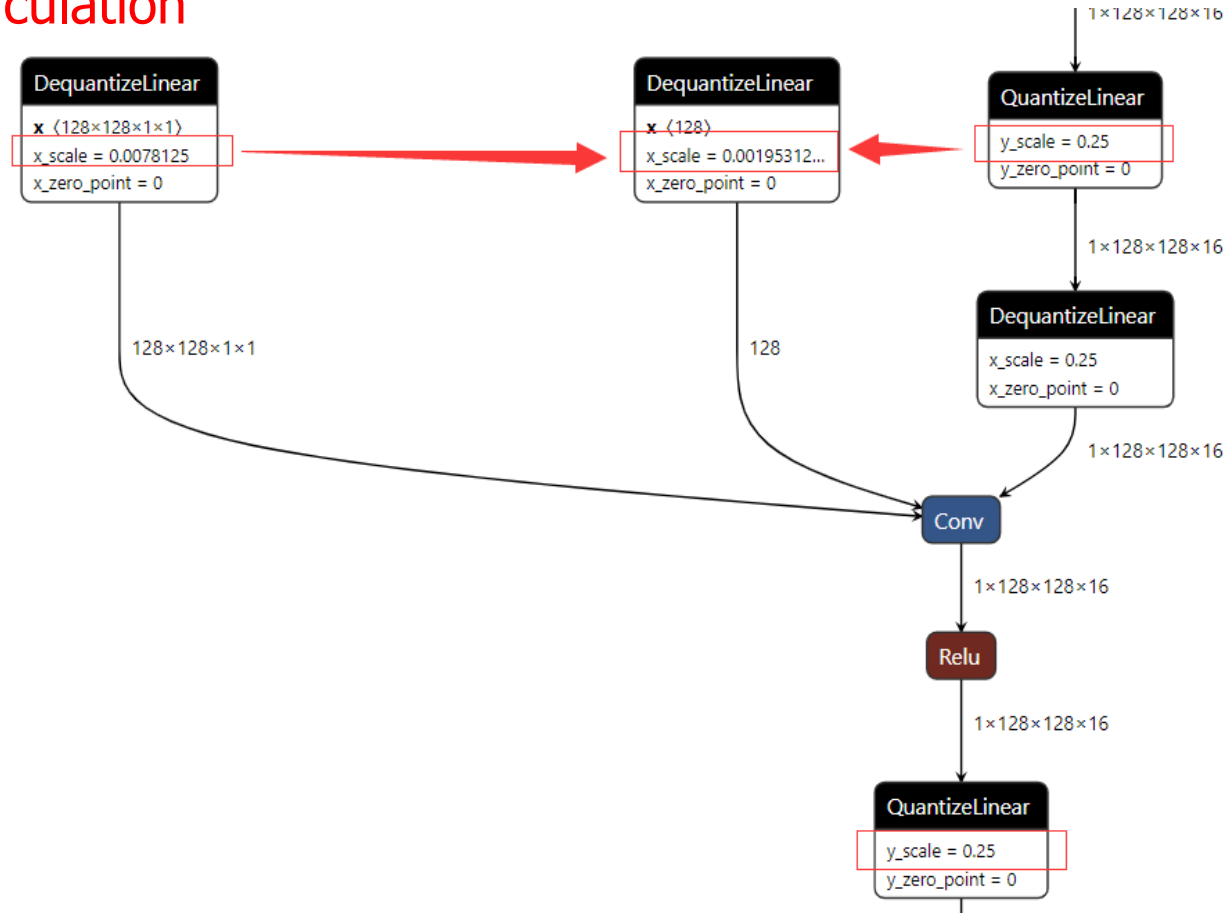
$$A_{n+1} = A_n \odot W_n + B_n$$

$$Aq_{n+1}S_{a_{n+1}} = Aq_nS_{a_n} \odot Wq_nS_{w_n} + Bq_nS_{b_n}$$

$$Aq_{n+1}S_{a_{n+1}} = (Aq_n \odot Wq_n)S_{a_n}S_{w_n} + Bq_nS_{b_n}$$

$$\because S_{b_n} = S_{a_n}S_{w_n}$$

$$\therefore Aq_{n+1} = (Aq_n \odot Wq_n + Bq_n)\frac{S_{a_n}S_{w_n}}{S_{a_{n+1}}}$$



Quantization implementation (II)

Convert floating point calculation to fixed point calculation

$$Aq_{n+1} = (Aq_n \odot Wq_n + Bq_n)\frac{S_{a_n}S_{w_n}}{S_{a_{n+1}}}$$

$S = 2^x$, **therefore:**

$$\frac{S_{a_n}S_{w_n}}{S_{a_{n+1}}} = 2^{a_n+w_n-a_{n+1}},$$

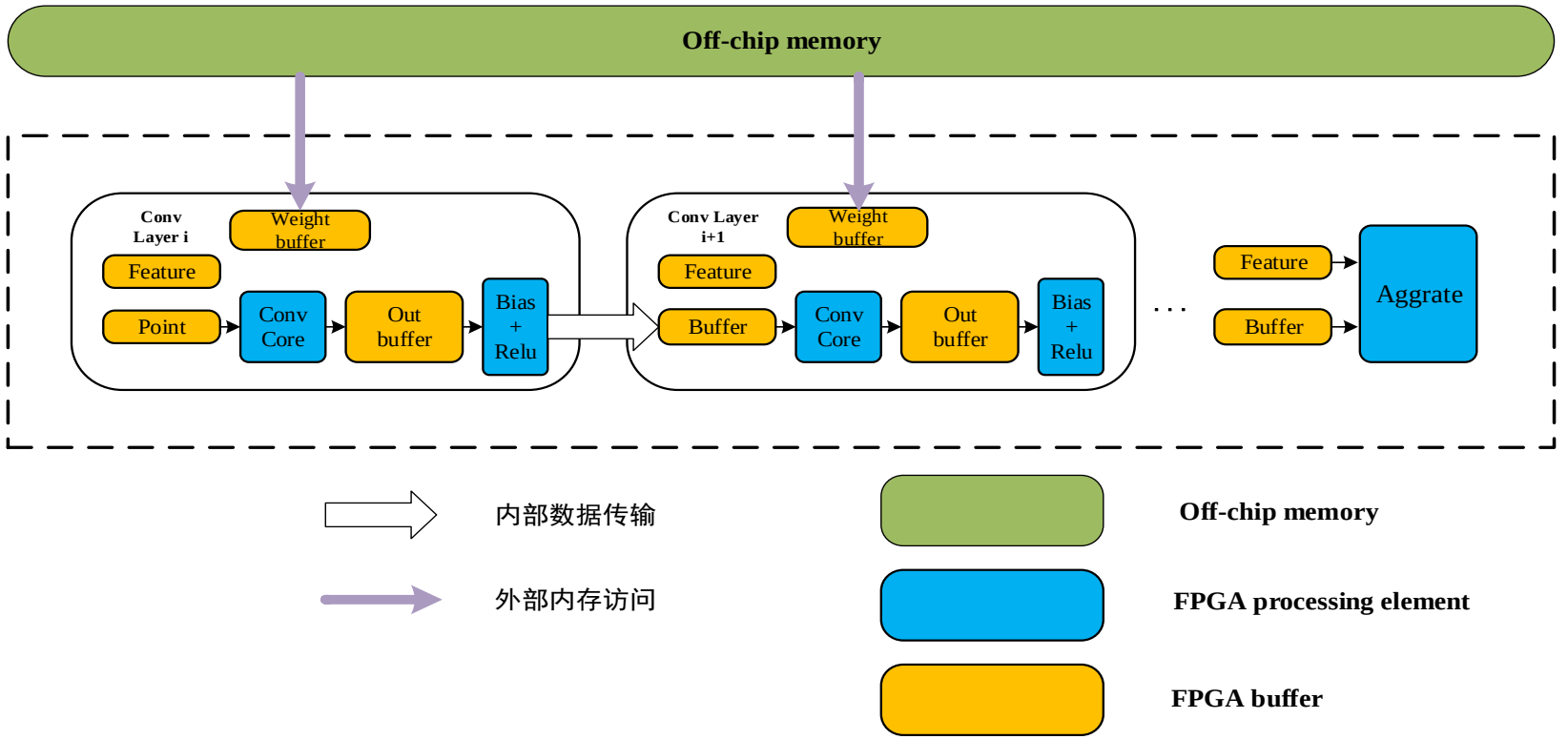
$$Aq_{n+1} = (Aq_n \odot Wq_n + Bq_n)2^{a_n+w_n-a_{n+1}}$$

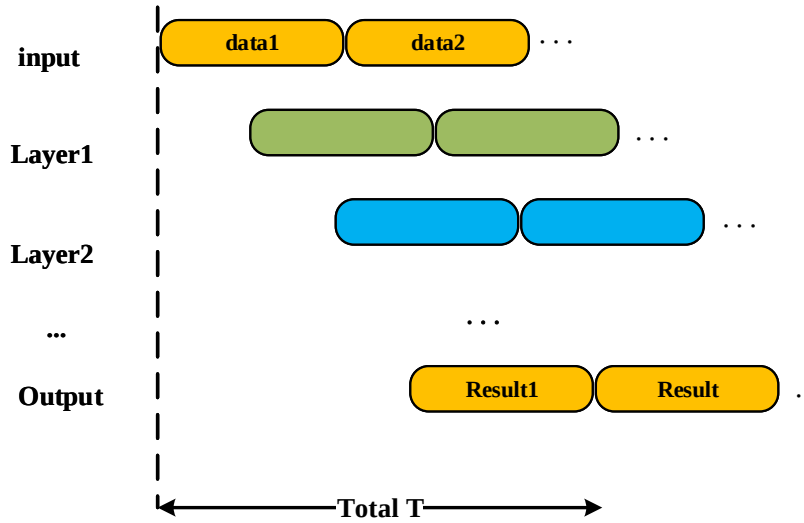
Any 2^x operation can be further simplified with shifting operations in hardware logic as:

$$\begin{cases} R \times 2^x = R << |x|, x \geq 0 \\ R \times 2^x = R >> |x|, x < 0 \end{cases}$$

$$\log_2 \frac{S_{a_n}S_{w_n}}{S_{a_{n+1}}} = a_n + w_n - a_{n+1}$$

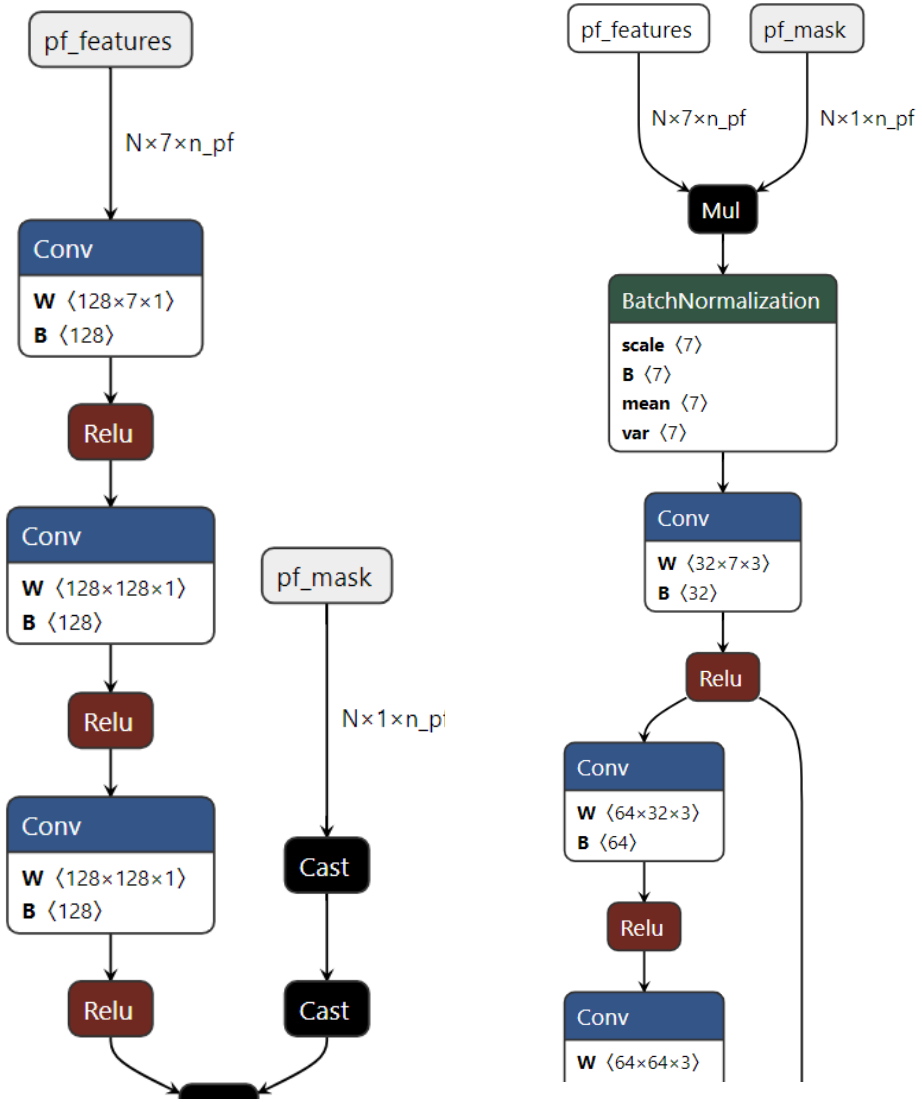
Pipeline parallelism





- ✓ A customized architecture to make full use of the resources of the hardware
- ✓ Each convolution layer is calculated in parallel without the need to write intermediate results back to external memory

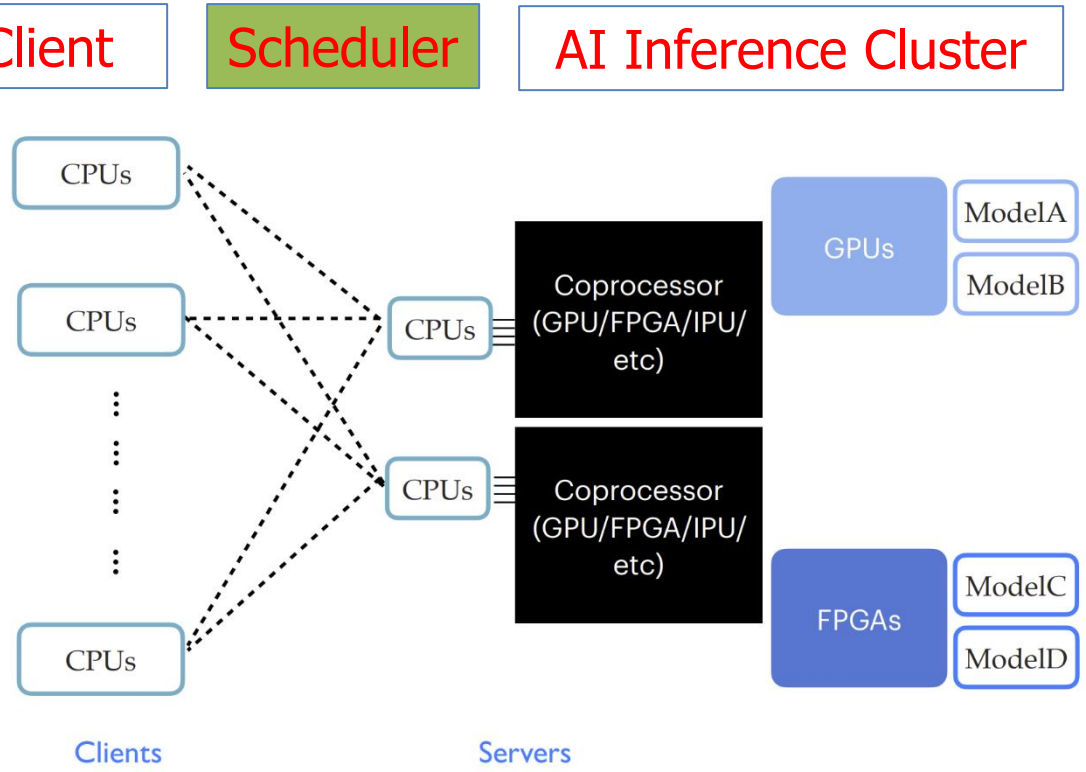
Quantization of PFN



	Parameters	ACC	Type
PFN	**330KB**	**~89.6%**	**float32**
PFN-quantization	**94KB**	**~88.1%**	**Int8**
PCNN	**1.4MB**	**~89.6%**	**float32**
PCNN-quantization	**367KB**	**~89.4%**	**Int8**

FPGA as a coprocessor

- Inference for ML algorithms can be accelerated dramatically by running on coprocessors such as FPGA, etc
- The most straightforward way to deploy algorithms on coprocessors is to run workflows on machines with FPGA coprocessors
- But, "Direct connection" can be inefficient due to unbalanced assignment of workload
- Inference cluster with a scheduler can be a good solution to the problem, which will be a new cloud service: Inference as a Service
  - Eg: CMS SONIC (Services for Optimized Network Inference on Coprocessors)



Portable Acceleration of CMS Mini-AOD Production with Coprocessors as a Service
https://indico.cern.ch/event/1283970/contributions/5554352/

Summary

- The large science facilities such as CEPC generally produce massive data, which brings unprecedent challenges to IT
- Machine learning (ML) based algorithms are becoming increasingly common in HEP work flow
- FPGA as a coprocessor can accelerate the AI inference dramatically with low latency and power consumption
- Some projects such as hls4ml greatly simplifies the difficulty of programming
- We have tried to implement particleNet and other models on FPGA

Acknowledgement

- The majority of the work was completed by our team members, including Yutao Zhang, Yu Gao and so on. I thank them for their outstanding work.
- Some of the materials in this presentation are sourced from projects like fastml and hls4ml, or Internet, and I would like to express my gratitude. If there are any copyright concerns, please notify me.

Thank you for your attention

chyd@ihep.ac.cn， lihaibo@ihep.ac.cn