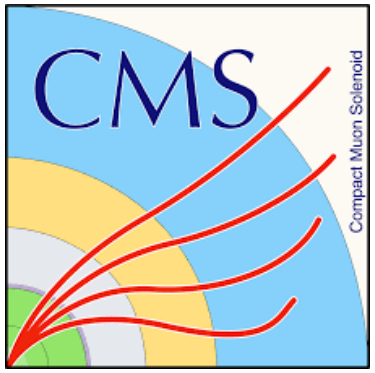


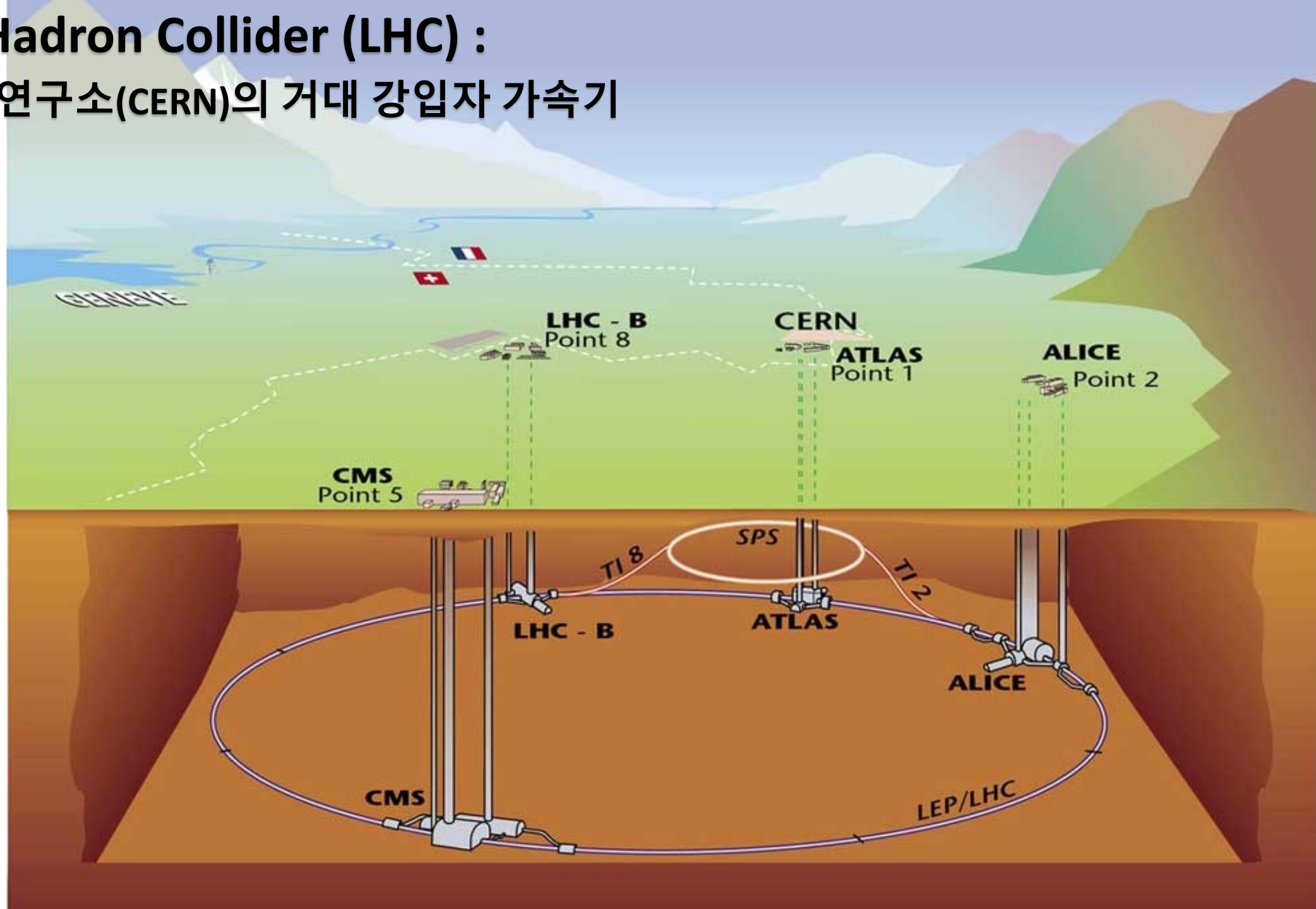
Real-time Machine Learning Technique for the CMS Phase 2 Upgrade

Chang-Seong Moon

Centre for High Energy Physics (CHEP), Kyungpook National University (KNU)

Caveat :This work was conducted by Jieun Hong, Bongho Tae, JiWan CHAE (KNU) and junwon Oh (KHU) students.

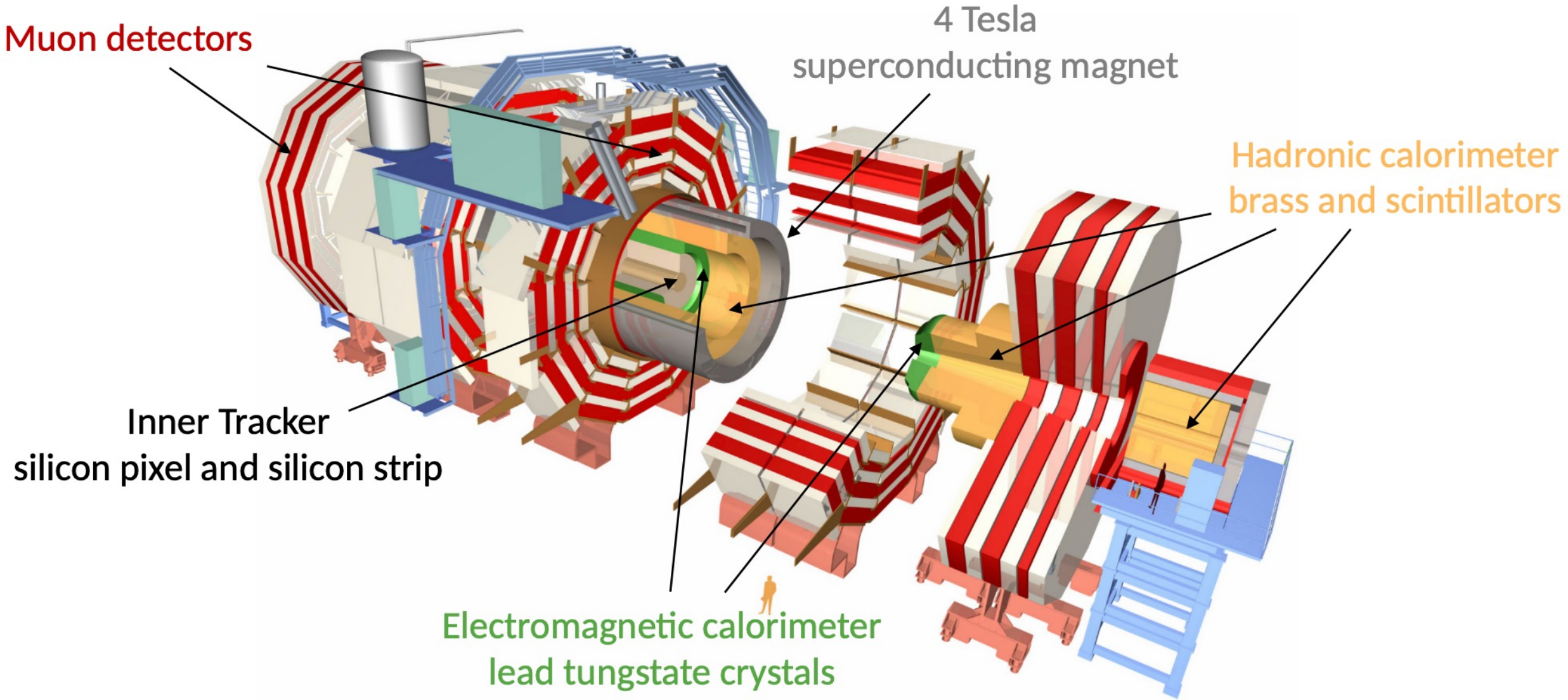
The Large Hadron Collider (LHC) :
유럽입자물리연구소(CERN)의 거대 강입자 가속기



CHANG-SEONG MOON (KNU)

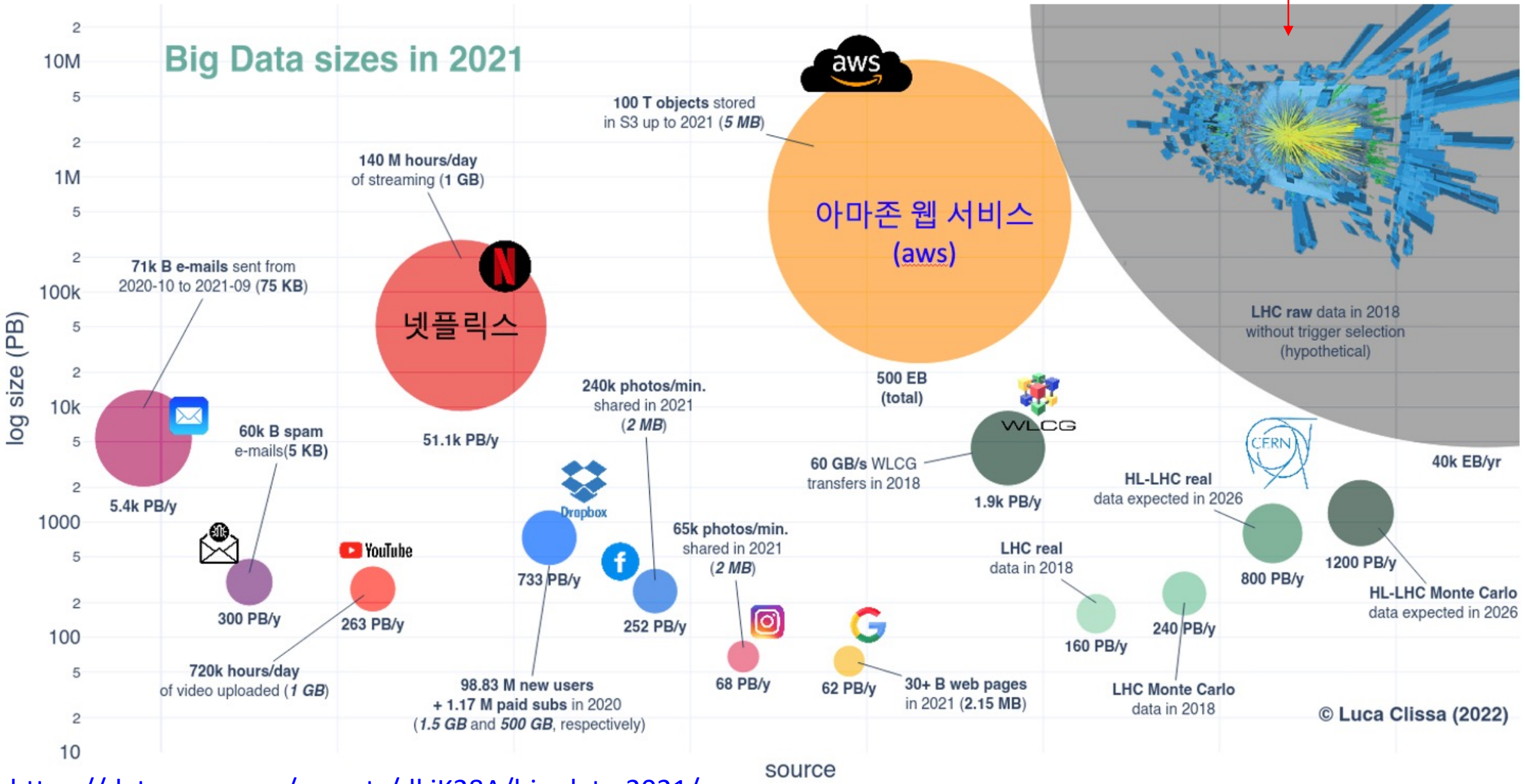
CMS detector

High-granularity detectors
Order of 100 Million channels



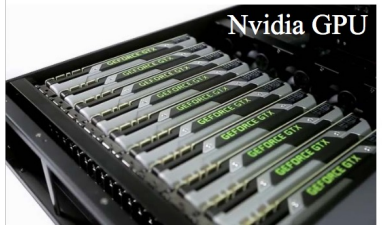
Big data in High Energy Physics

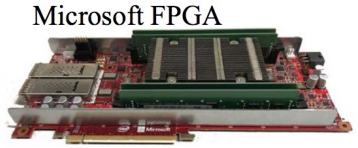
CERN LHC raw 데이터
트리거(Trigger)하기 전

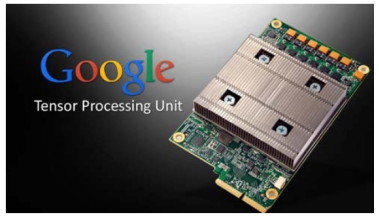


https://datapane.com/reports/dkjK28A/big-data-2021/

Heterogeneous computing





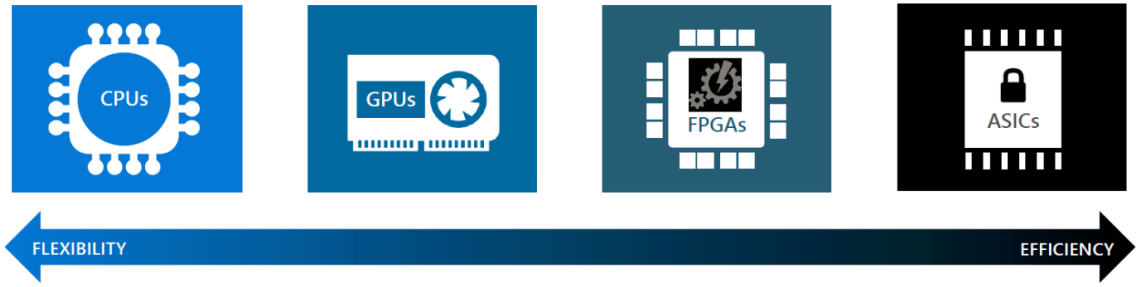


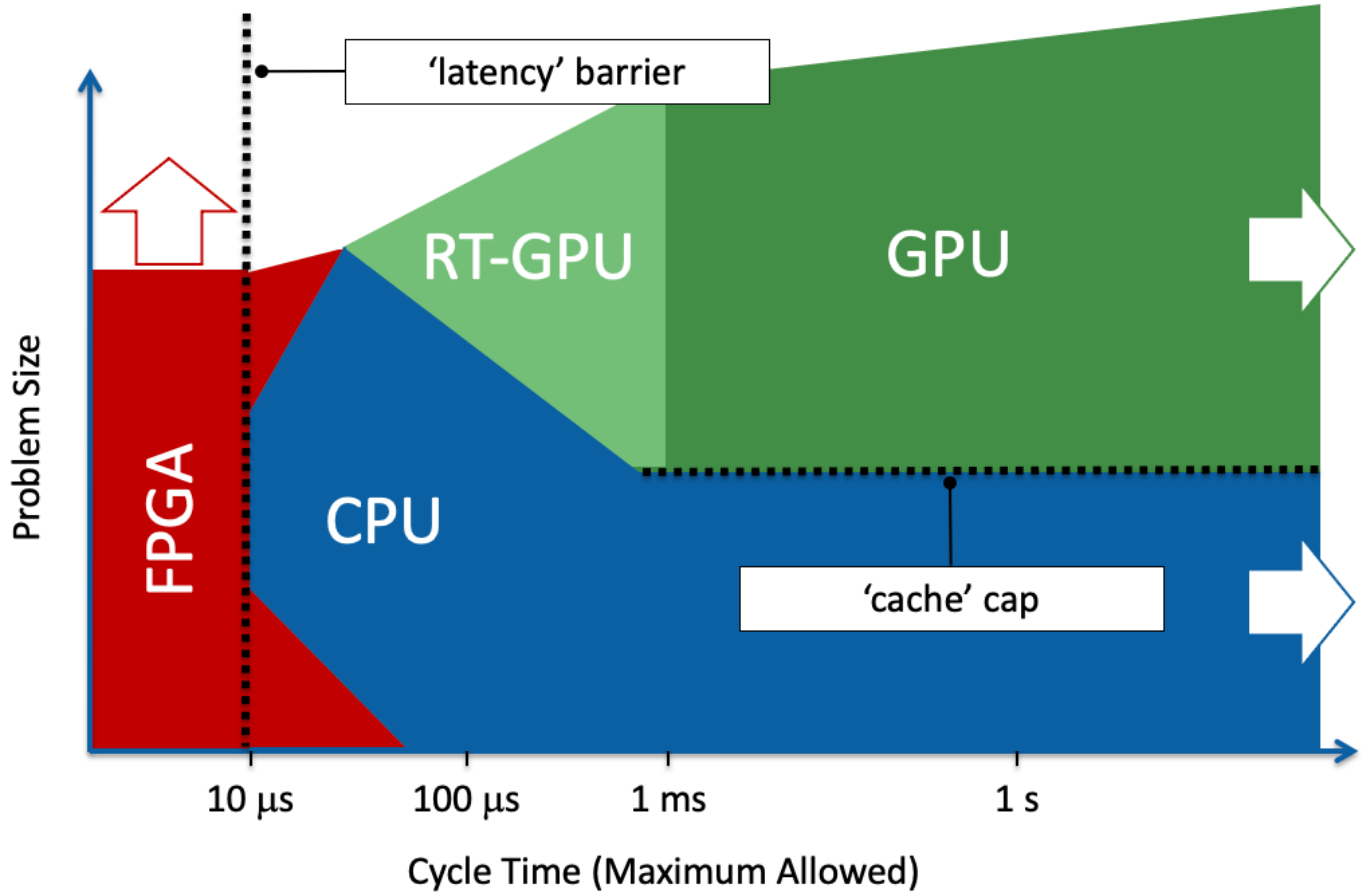
CPU
- Small models
- Small datasets
- Useful for design space exploration

GPU
- Medium-to-large models, datasets
- Image, video processing
- Application on CUDA or OpenCL

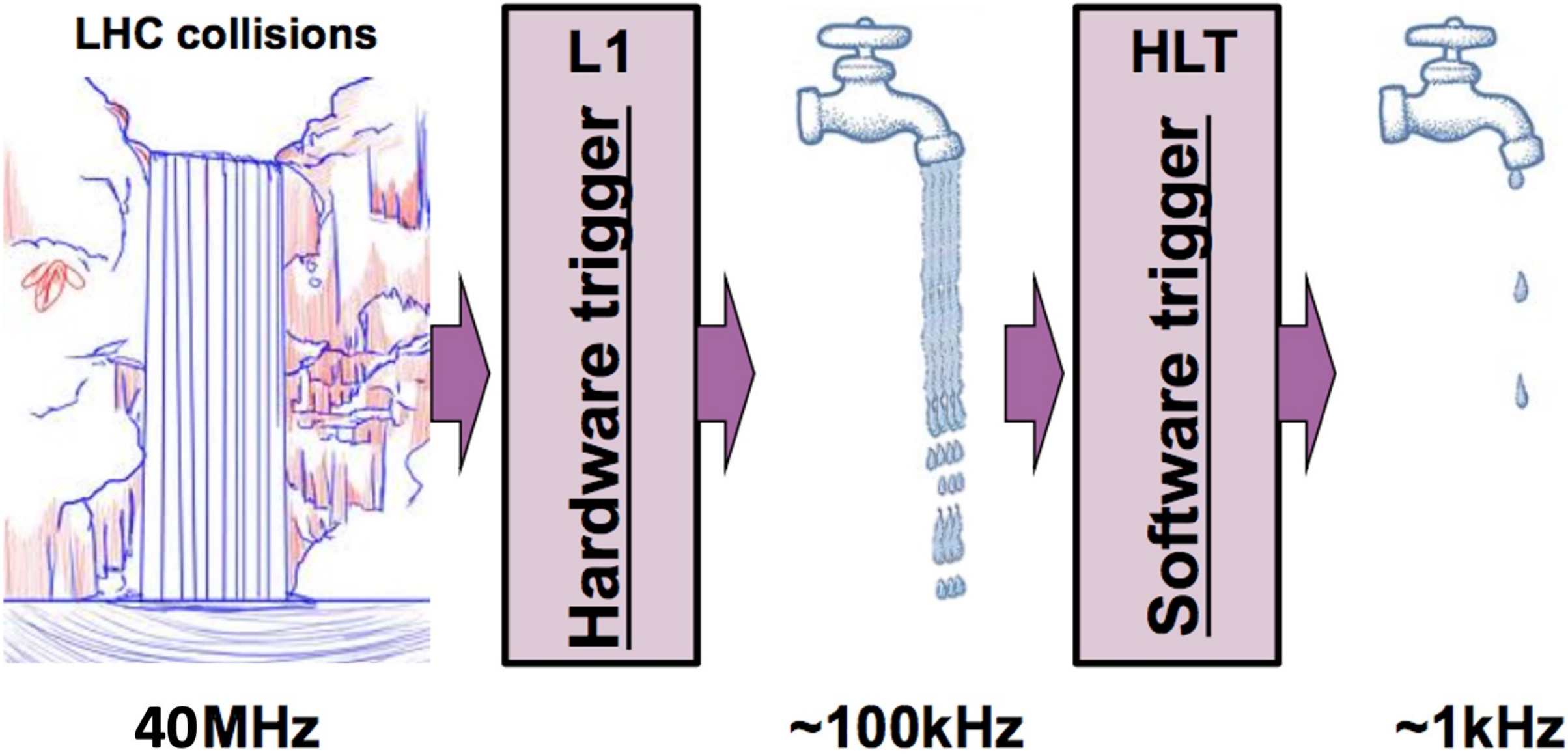
TPU
- Matrix computations
- Dense vector processing
- No custom TensorFlow operations

FPGA
- Large datasets, models
- Compute intensive applications
- High performance, high perf./cost ratio

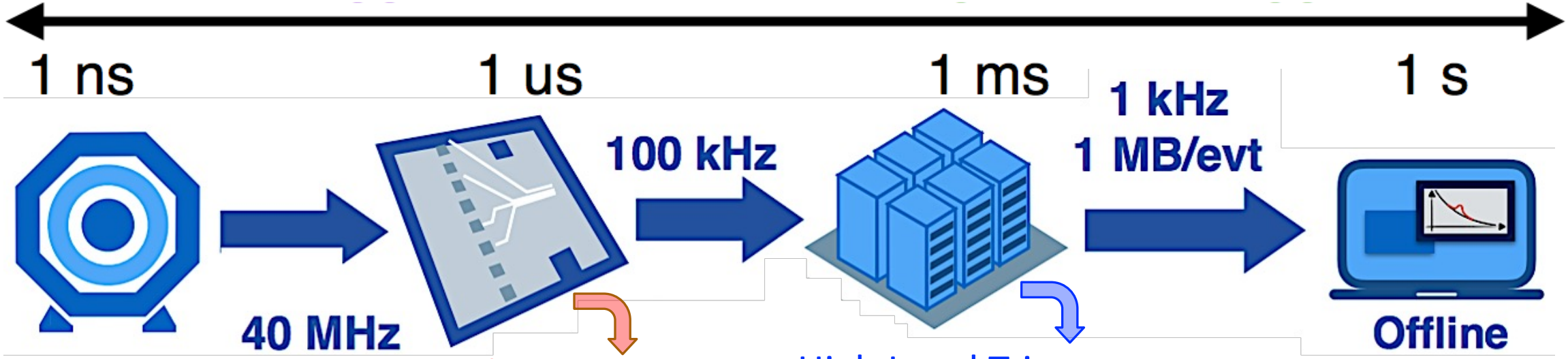




CMS 데이터 트리거 원리 : 실시간 이벤트 필터링



Current CMS Data Processing



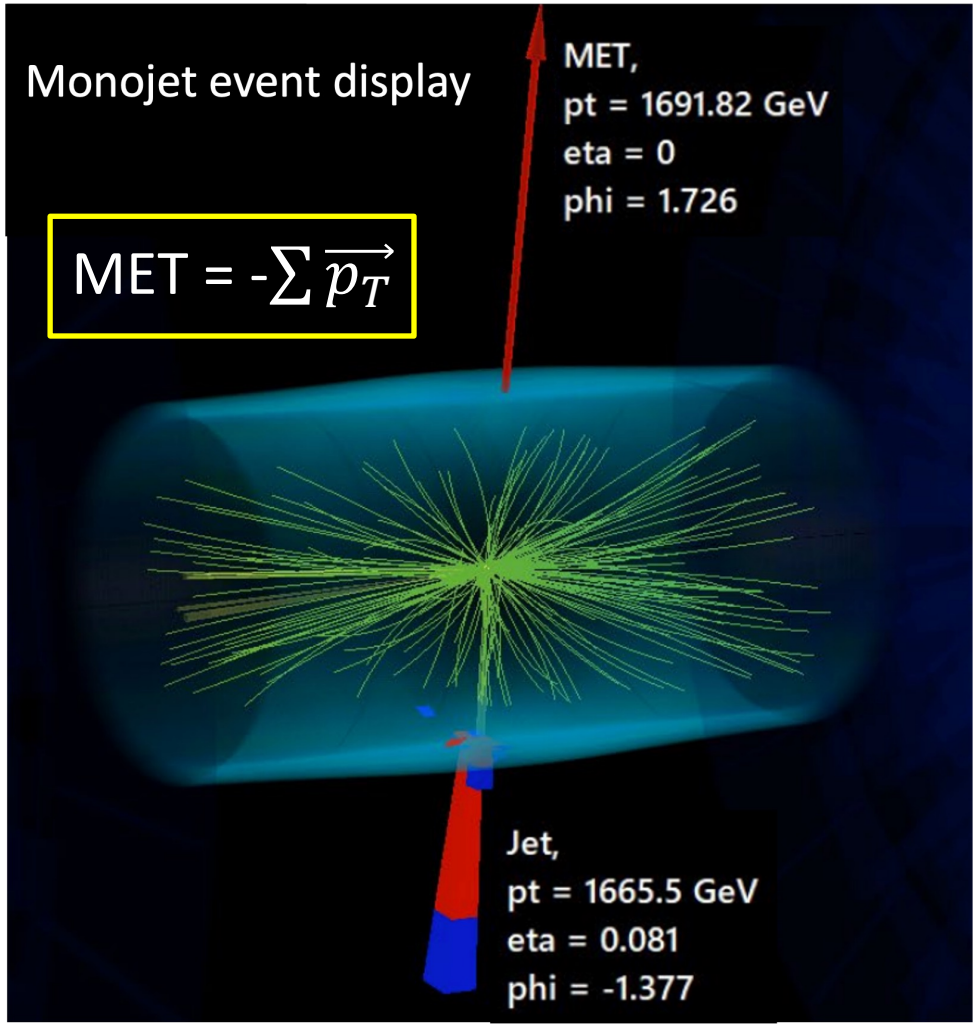
- 40 MHz LHC clock
- ASICS and FPGAs
- Decision made in ~4 μs
- ML inference with FPGAs
- 99.75% (399/400) events are rejected

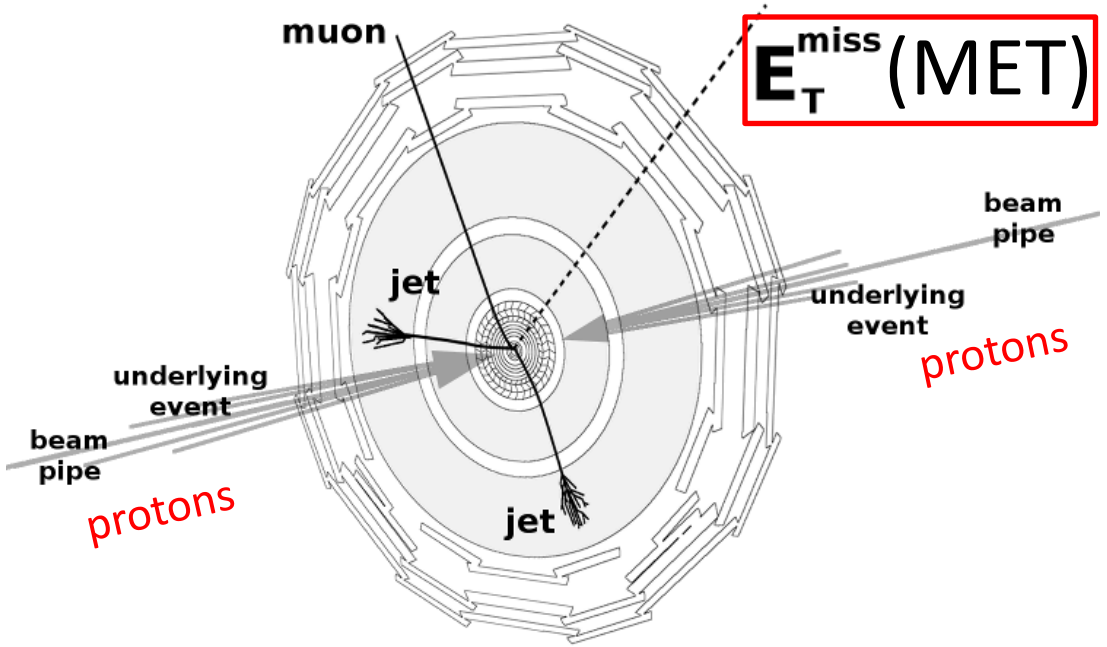
- 100 kHz Input rate
- CPU Farm : 30,000 CPU cores
- Decision made in 300 ms
- 99% (99/100) events are rejected

After triggering, 99.9975% (39999/40000) of events are gone forever!

Target object: Missing Transverse Energy (MET) at Level-1

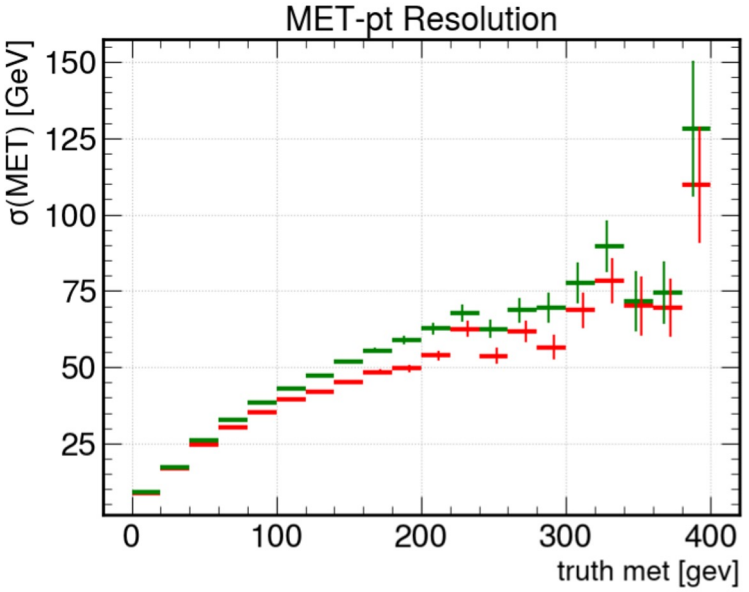
- Energy that is not detected in a detector
  - By conservation of momentum, the sum of transverse momentum should be zero

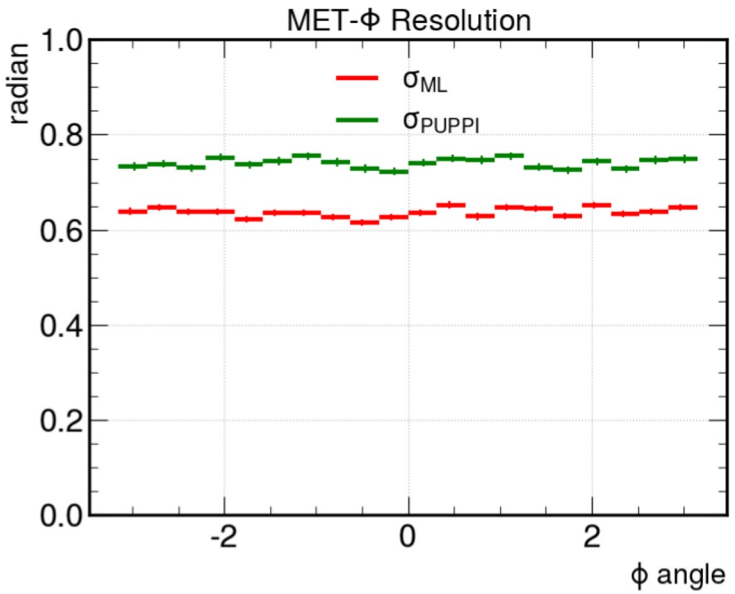


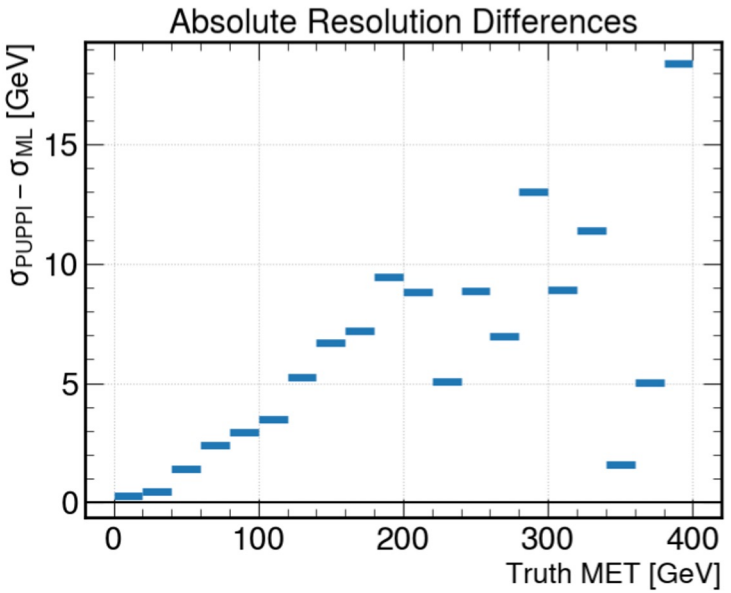


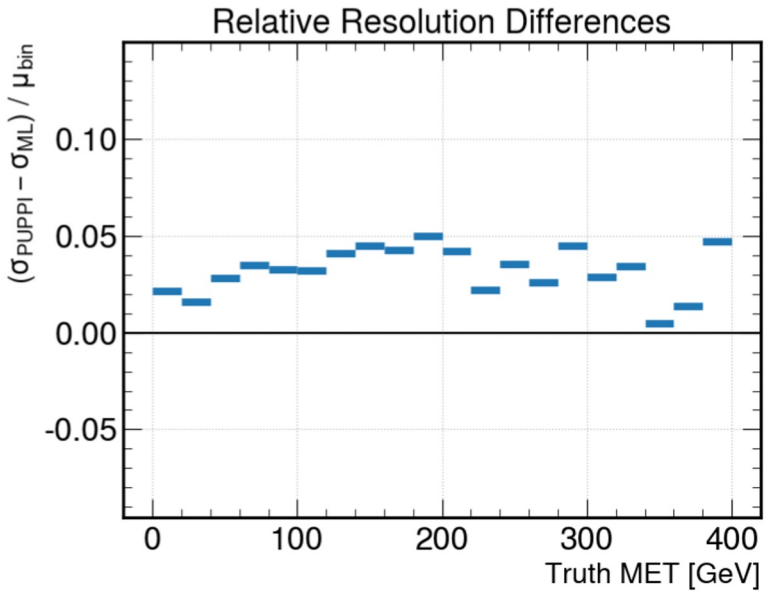
- MET calculated by PUPPI (Pileup Per Particle Identification) algorithm using CMS detector

Resolution for MET p_T & phi using DNN









10% improved for MET measurement with respect to standard PUPPI MET

CHANG-SEONG MOON (KNU)

XILINX

https://www.xilinx.com/publications/powered-by-xilinx/cerncasestudy-final.pdf

Artificial Intelligence Accelerates Dark Matter Search

Integrating Inference Acceleration with Sensor Pre-processing in Xilinx FPGAs Delivers Performance Unachievable by GPUs and CPUs

AT A GLANCE:

Customer: High energy physics researchers from an association of leading international institutions (CMS Institute) conducting experiments at the European particle physics laboratory, CERN.

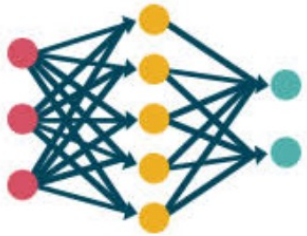
Industry: Scientific Research

Employees: CMS Institute has more than 4,000 global scientific collaborators representing 200 institutes and universities from more than 40 countries.

Flow of ML to FPGA Deployment

Model Training: Use TensorFlow, Keras or Pytorch to design and train a machine learning model with the dataset.

Model training & Quantization



Quantization: Quantize the model using QKeras to reduce its size and computational complexity by lowering the bit precision of weights and activations.

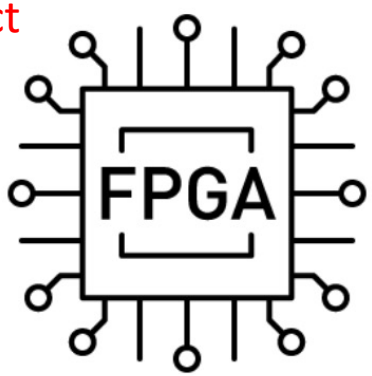
Output : h5 file

HLS Conversion

HLS Conversion: Convert the quantized model into High-Level Synthesis (HLS) code using hls4ml, allowing it to be implemented on hardware.

Output : HLS Project

HLS

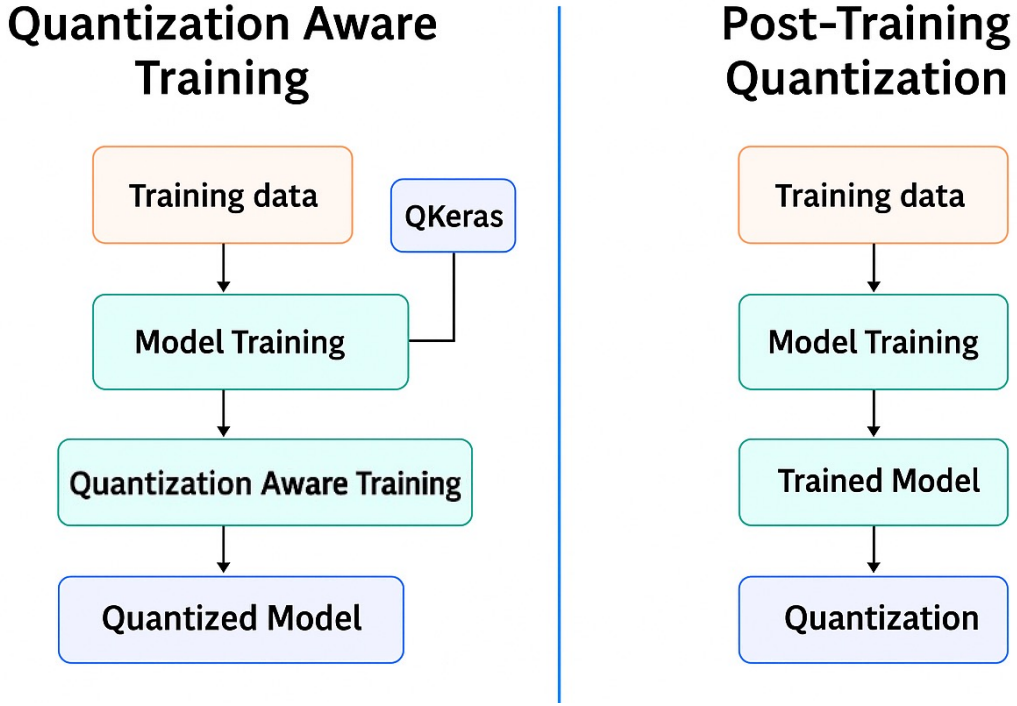


CMS Level-1 Trigger

Output : HDL Source code

FPGA Deployment: Deploy the HLS code onto the FPGA to execute the model in real-time, leveraging the FPGA's parallel processing capabilities for optimal performance.

Quantization aware training (QAT)



- Post-Training Quantization (PTQ) vs QAT
  - **QAT (with QKeras)**
    Quantization effects are simulated during training. The model learns to adapt to low-bit precision (e.g., quantized weights and activations).
    → Higher accuracy retention, hardware-friendly, ideal for FPGA deployment.
  - **PTQ (Direct Quantization)**
    Quantization is applied only after training a full-precision model.
    No retraining is performed, so accuracy often drops.
    → Simple to apply, but limited for resource-constrained hardware like FPGAs.
- Key Point
  QAT provides quantized models that are both accurate and efficient, making them suitable for real-time applications such as the CMS Level-1 Trigger on FPGA.

Quantization Aware Training with QKeras

Why QKeras?

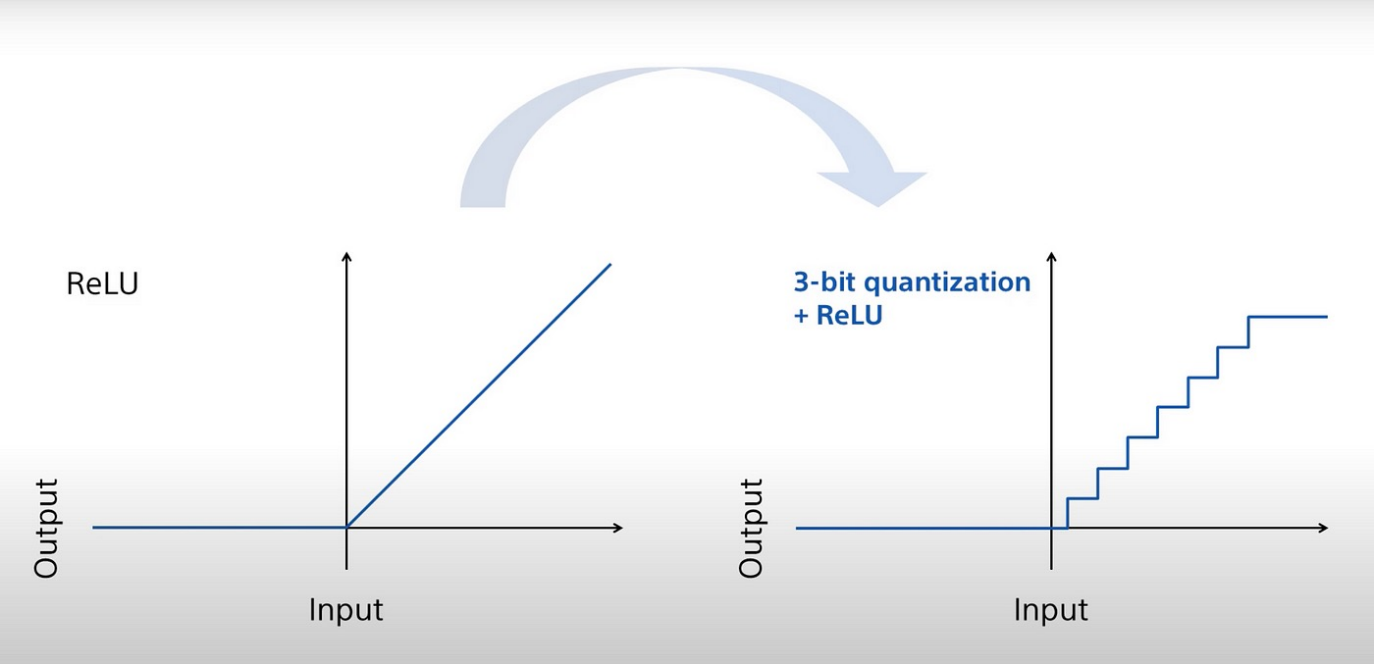
- Extension of Keras for Quantization Aware Training (QAT)
- Directly trains with low-bit weights & activations (e.g. quantized ReLU)
- Produces hardware-friendly models for FPGA/ASIC deployment

Benefits

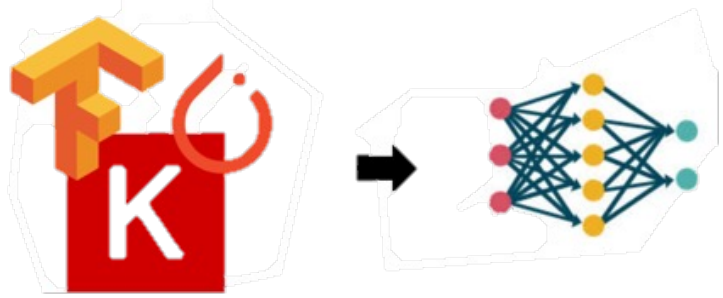
- Accuracy retention: Preserves performance compared to post-training quantization
- Hardware efficiency: Reduces DSP/LUT/BRAM usage on FPGA

Integration with hls4ml

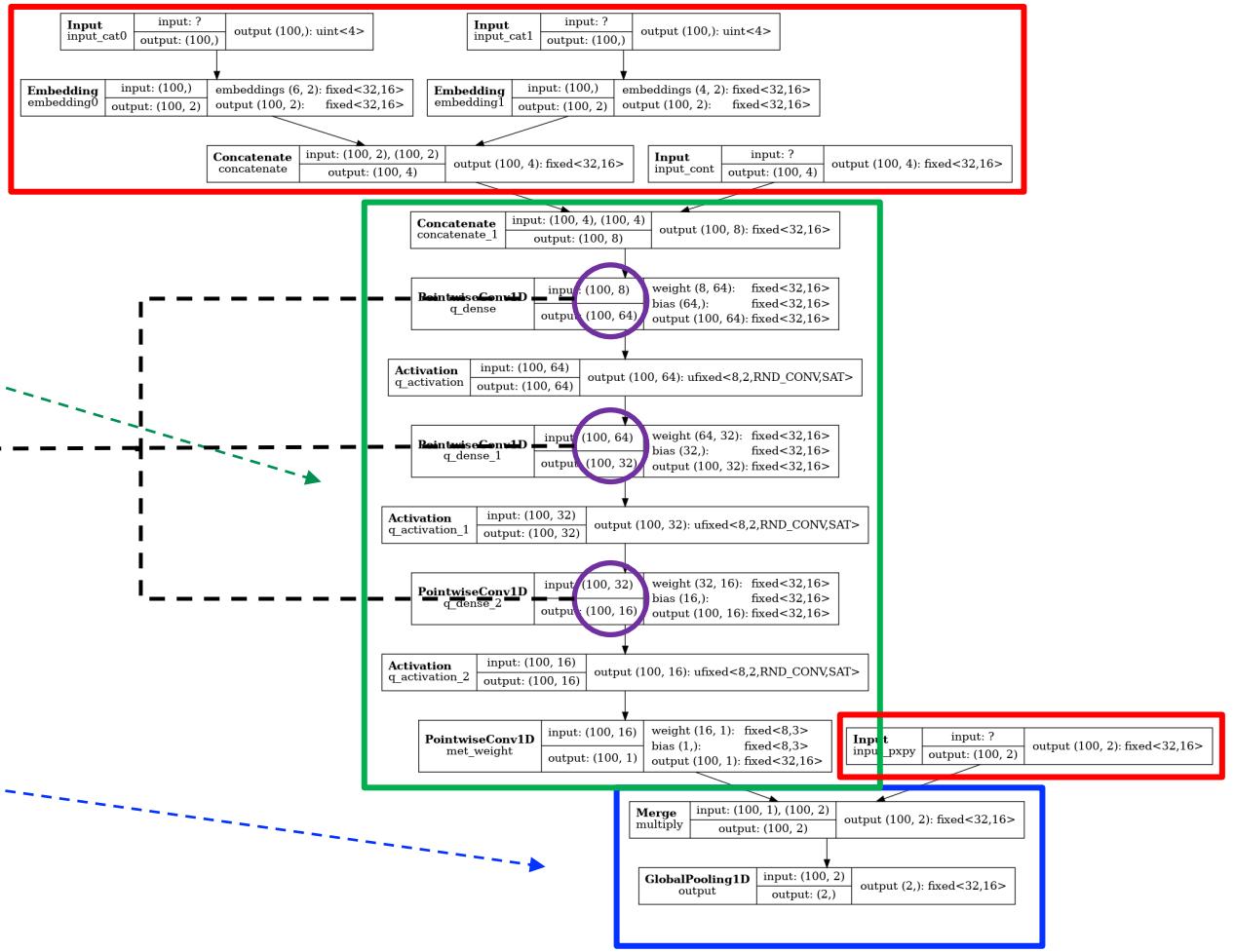
- QKeras quantized models
  - Converted into HLS code via hls4ml
- Enables synthesis & deployment on Xilinx UltraScale+ FPGA for the CMS L1 Trigger



ML Model Architecture



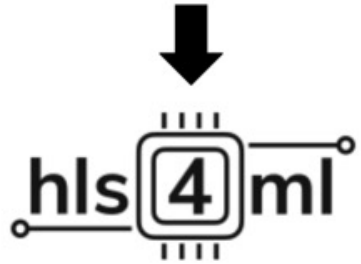
- PUPPI candidates as inputs
  - Categorical : [charge, pdg_ID]
  - Continuous : [px, py, pt, ϕ, η, puppi_Weights]
- Training workhorse
  - Fully connected neural network
  - 3 hidden layers (Dense)
    Dimension: # of Particle X [64, 32, 16]
  - Batch Normalization
- Computing MET
  - No training parameters here.
  - Just matrix multiplication
    Trained weight X PUPPI MET



Comparison between MET calculations

- Comparison of MET calculation between the results of the quantized model and the model simulated using hls4ml.
- The test was conducted with 1000 events, using fixed-point precision set to <32,16> which is ideal case.
- As shown, the results of both models align closely.

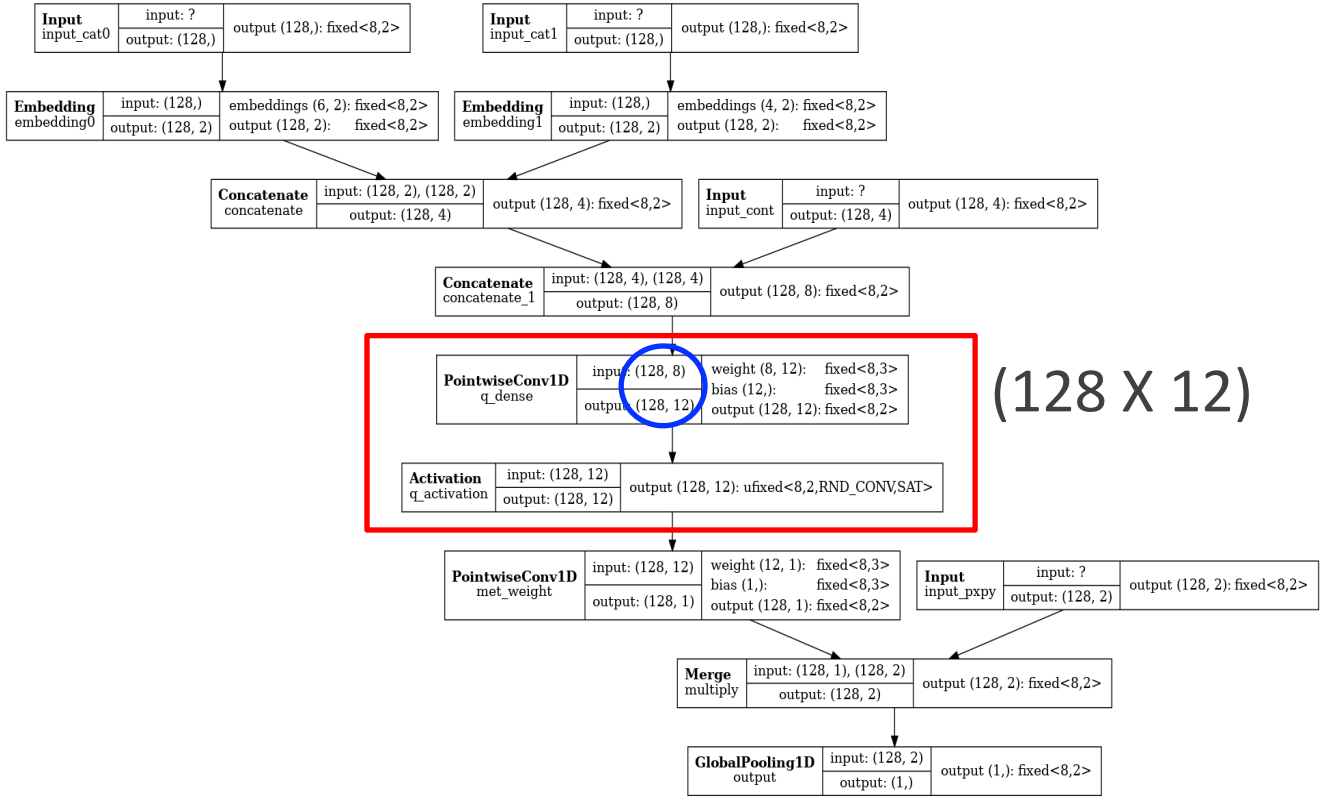
FPGA : VU13P



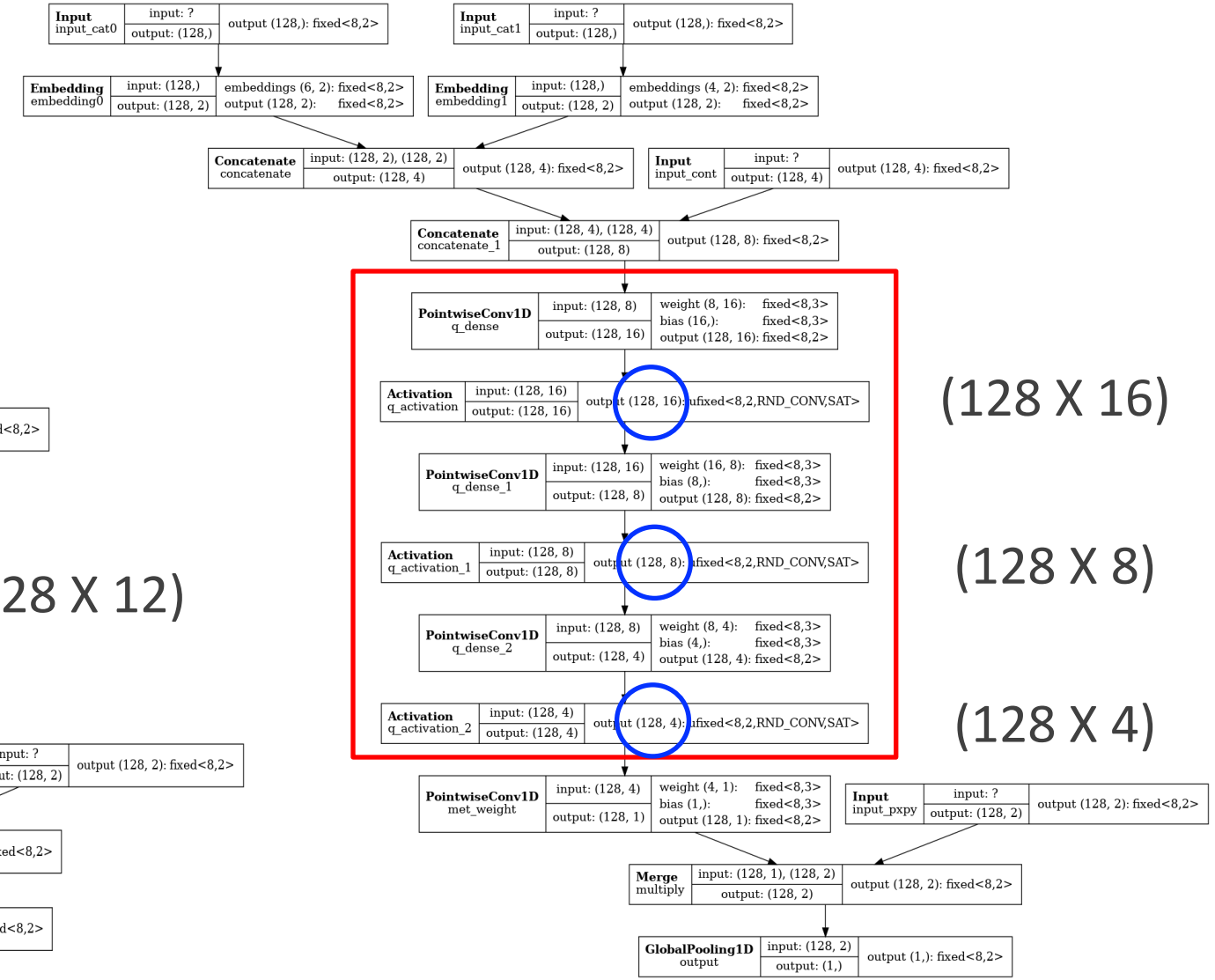
Comparison between GenMET, PUPPI MET, QKeras MET, and hls4ml MET

Demo Model Test

- Model Architecture



(128 X 12)



(128 X 16)

(128 X 8)

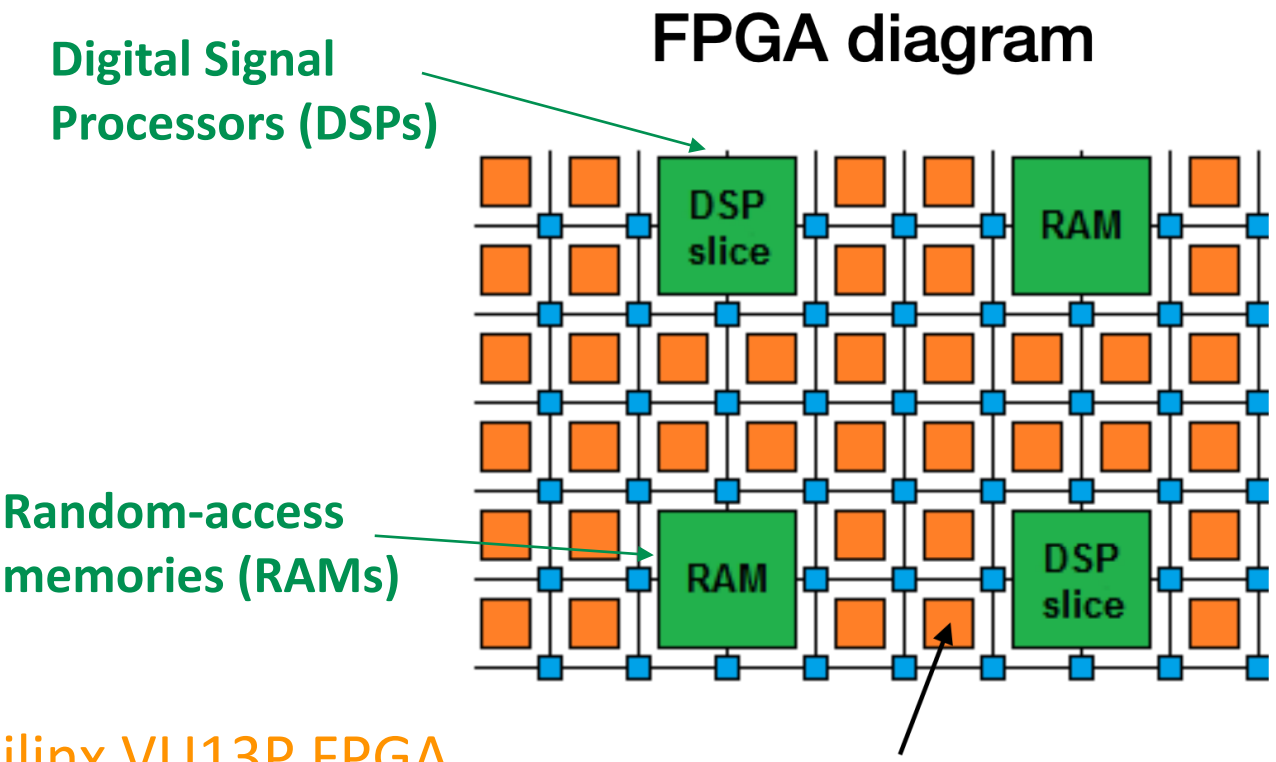
(128 X 4)

1 layer Model with reduced hidden layer dimensions

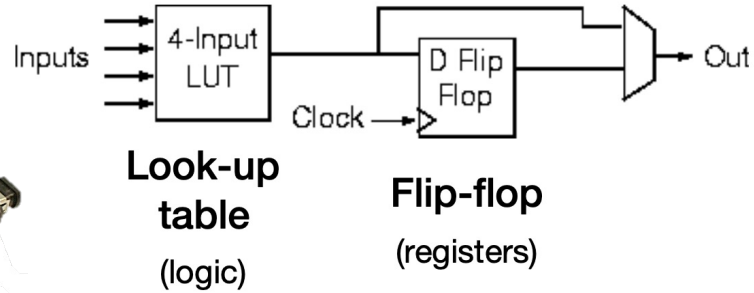
3 layers Model with reduced hidden layer dimensions

Resources in FPGA

- **Block RAM (BRAM)**
  - Memory blocks used inside the FPGA for high-speed data storage and retrieval.
- **DSP blocks**
  - Dedicated units in the FPGA for performing complex arithmetic operations like multiplication and addition.
- **Flip-Flops (FF)**
  - Basic storage elements that hold data and state during each clock cycle.
- **Look-Up Tables (LUT)**
  - Tables used to quickly perform logical operations by referencing pre-defined results based on input values.
- **UltraRAM (URAM)**
  - High-capacity memory blocks designed for storing and processing large datasets.



Xilinx VU13P FPGA



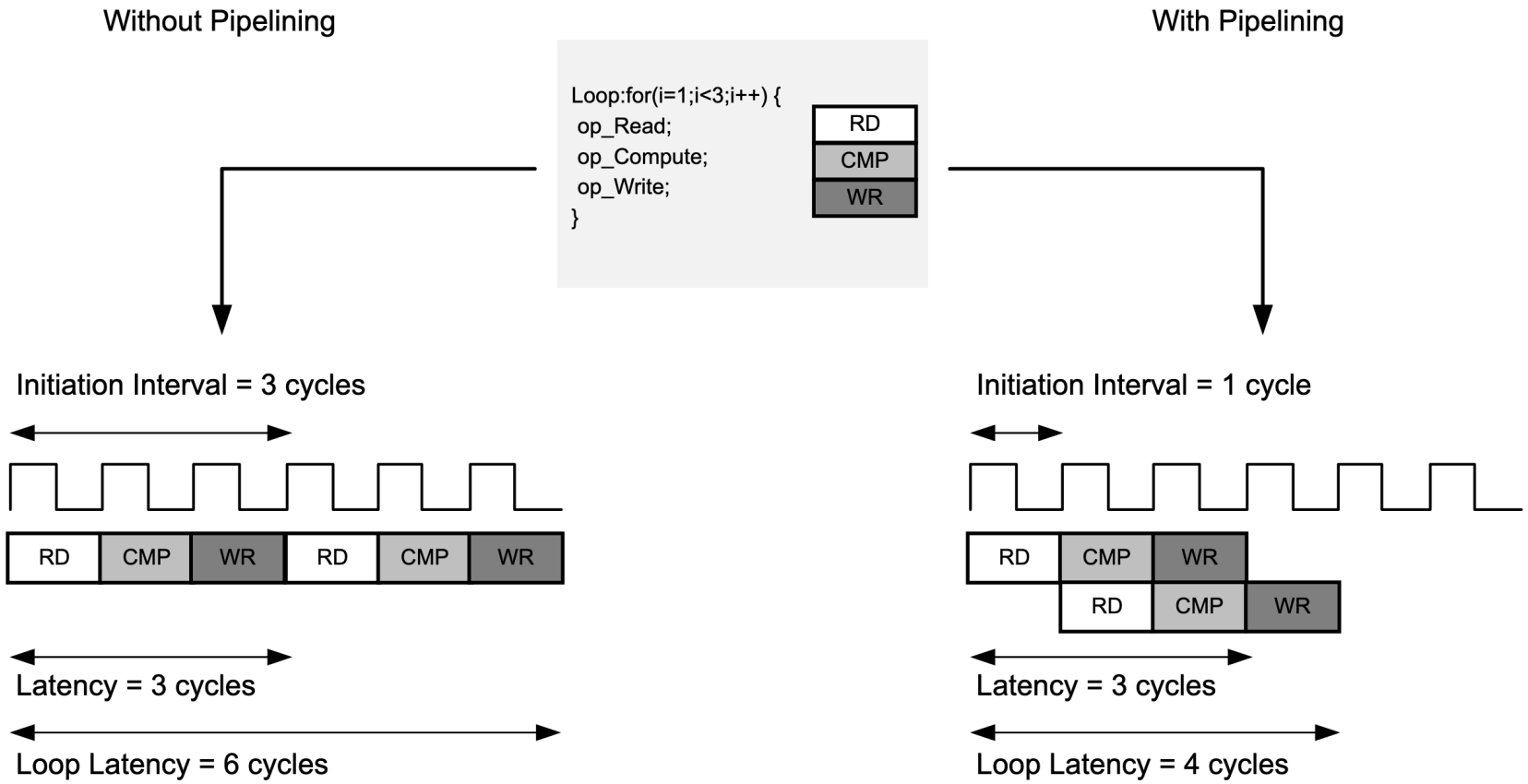
Resources in FPGA

Latency

- Refers to the total time taken for a task to complete after it has started. Lower latency indicates faster processing and quicker response times.

Interval

- The time gap between consecutive tasks or operations starting. A smaller interval means that tasks are initiated more frequently, improving the system's throughput.



Demo Model Test

FPGA : VU13P

- Resource Utilization and Latency

Clock period: 2.7 ns

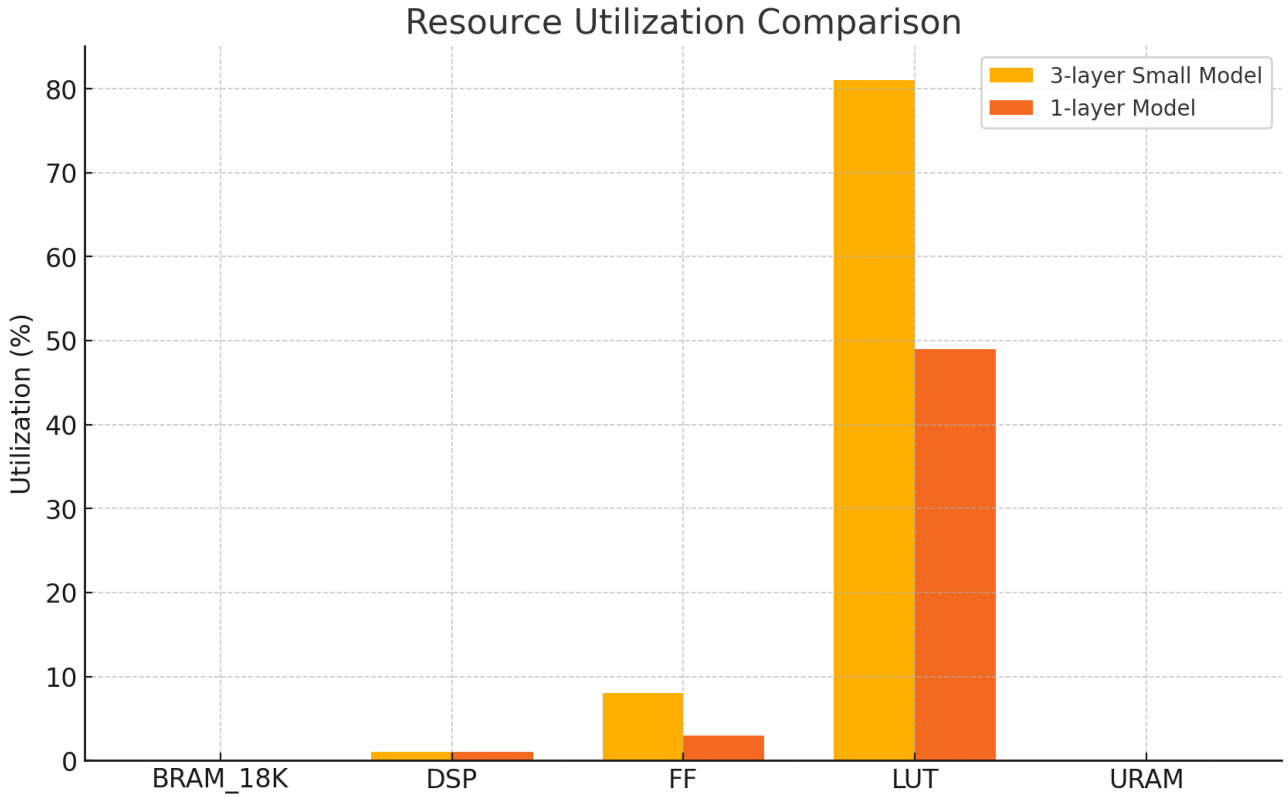
Model	Dimension	Latency (Cycles)	Latency (abs.)	Interval	Pipeline
1-layer	128 X 12	23	62.100 ns	8	Yes
3-layer	128 X [16,8,4]	168	0.454 us	54	Yes

- Both models were synthesized to meet the CMS L1 trigger requirement of an interval below 55.

- As expected, the 3-layer model consumes significantly more resources compared to the 1-layer model.

- This is primarily because most of the computational tasks are handled within the hidden layers.

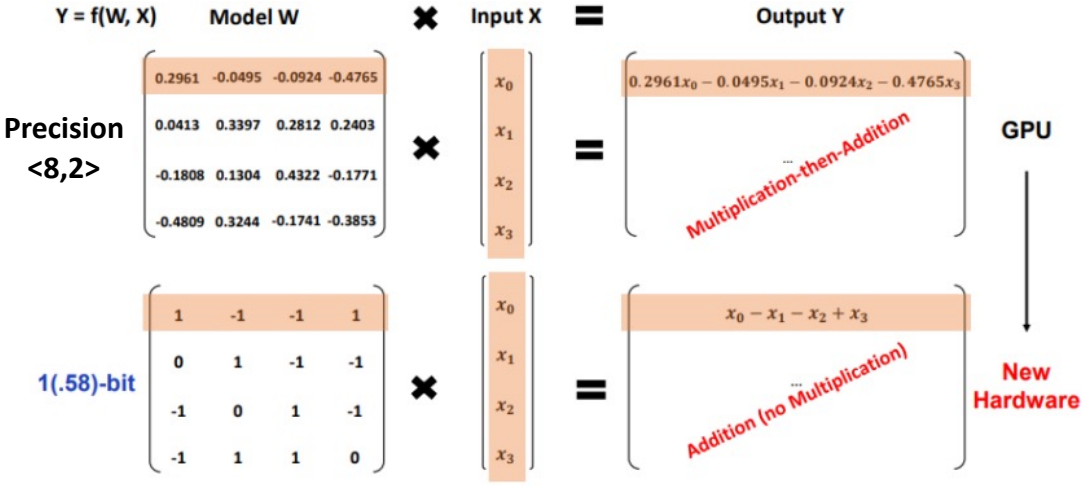
- We plan to address this through optimization in the future.



Exploring Different Quantization Schemes

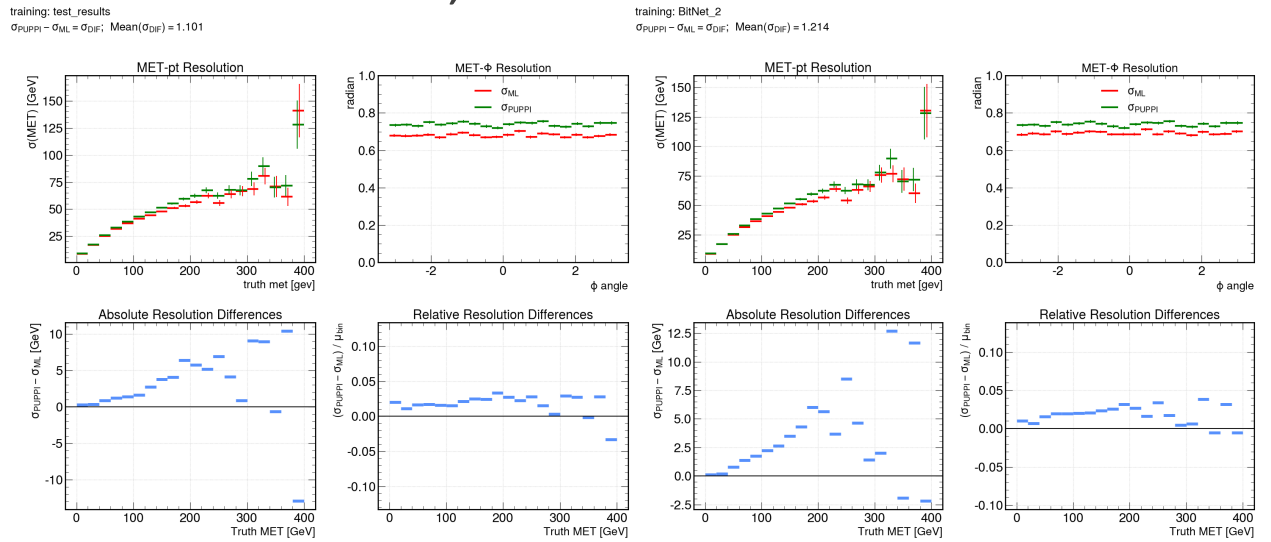
- Performance with 1.58-bit Precision

- +1 if $x >$ threshold
- −1 if $x <$ −threshold
- 0 otherwise



- **Background**
  - Tested extreme case of very low precision (1.58-bit)
  - Goal: check how much performance is kept with such quantization
- **Method**
  - Applied 1.58-bit quantization to both weights and activations
  - Compared against Precision <8,2> baseline
- **Results**
  - MET response and resolution remain stable
  - Only small differences in resolution observed
  - No large performance loss
- **Even ultra-low precision (1.58-bit) can keep reasonable performance, useful for FPGA optimization**

Precision <8,2> VS 1.58-bit



Bongho Tae (KNU)

Different Approaches to Quantized MET Regression

- GNN-based Model

Motivation

- Baseline model: fully connected network (simple, fast).
- Alternative: graph-based model to include relations between particles.

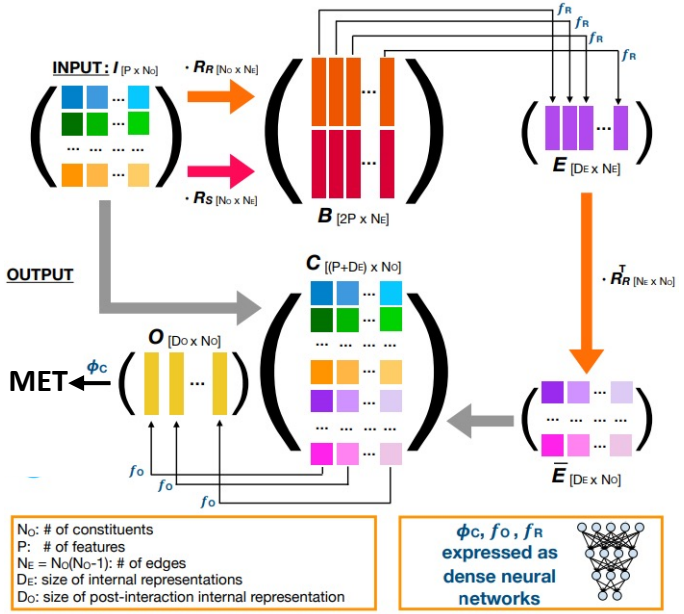
Method

- Candidates treated as nodes, kinematic variables define connections.
- Same training and quantization flow applied to both models.

Results

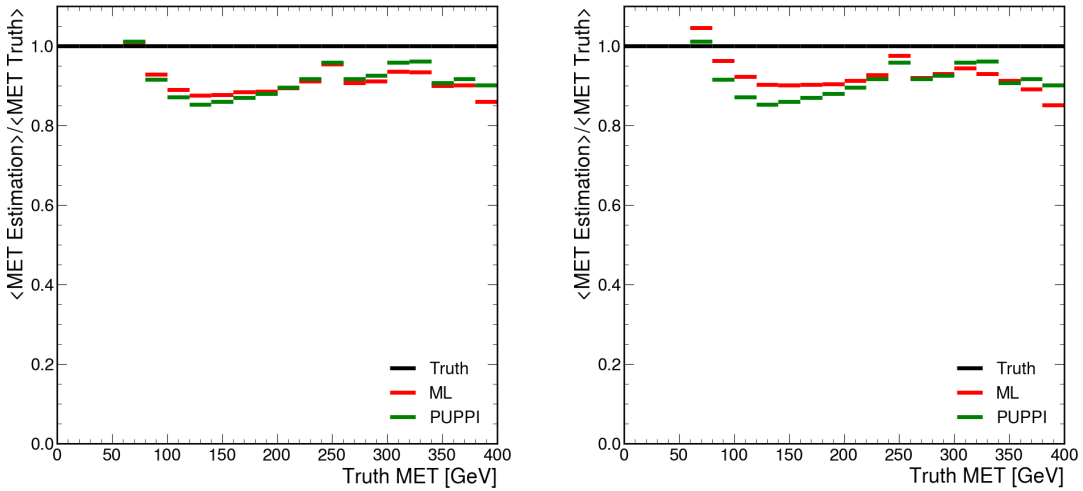
- FCNN and GNN show similar overall performance.
- Graph model keeps stable resolution, especially in high-MET region.

A useful alternative, currently under optimization for FPGA implementation



JEDI-net: a jet identification algorithm based on interaction networks

FCNN VS GNN

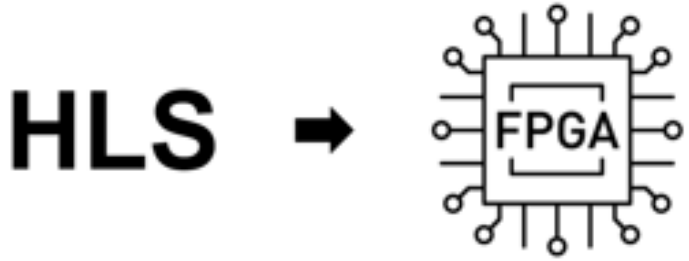


Jiwan Chae (KNU)

FPGA Firmware simulation and implementation

1. APx Firmware Simulation

- Verify that the algorithm functions correctly on the APx board through simulation, and to test data transfer processes and performance.



- Simulation Process:
  - The operating environment of the board is virtually set, and the algorithm is tested on how it processes input data.
  - The speed and accuracy of data processing through the GT link are assessed.

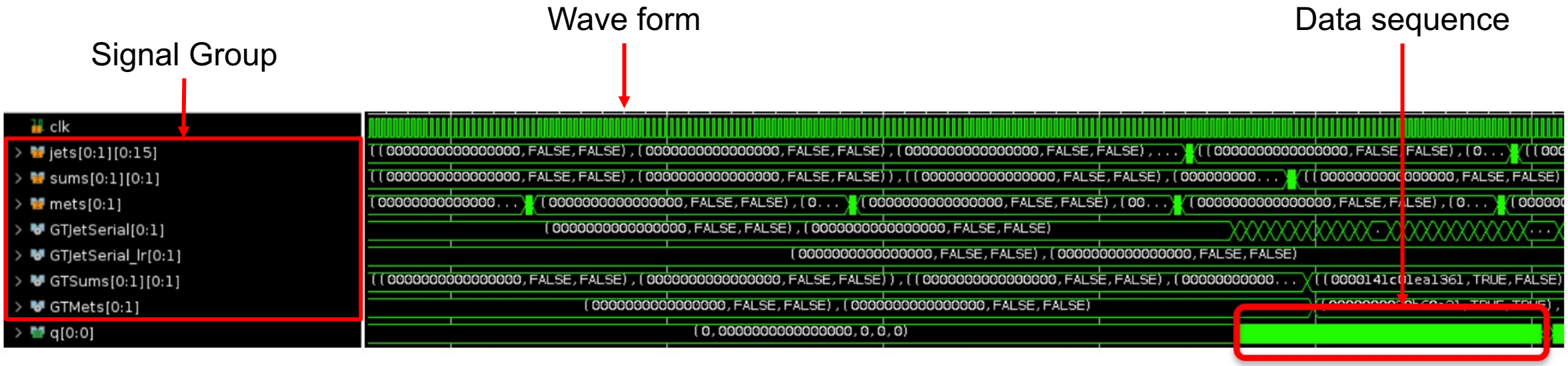
2. APx Firmware Implementation

- Deploy the algorithm onto the APx board and confirm real-time operation.
- Implementation Process:
  - The algorithm, validated through simulation, is uploaded to the physical board.
  - The algorithm is implemented on a Xilinx Vertex UltraScale+ FPGA, and resource usage, latency, and other performance metrics are monitored.
  - The efficiency of resource usage by individual sub-algorithms is verified.

Examples of Future Tasks

- APx Firmware Simulation

HLS → FPGA



- **Waveform used to check how each signal changes over time**
- It is used in the hardware debugging and verification process.
- It checks whether the timing or value of a specific signal changes correctly and whether the designed logic operates as expected.

- CL to GT Link: 54 words

6 Time Multiplexer (TMX)×360MHz/40MHz=54

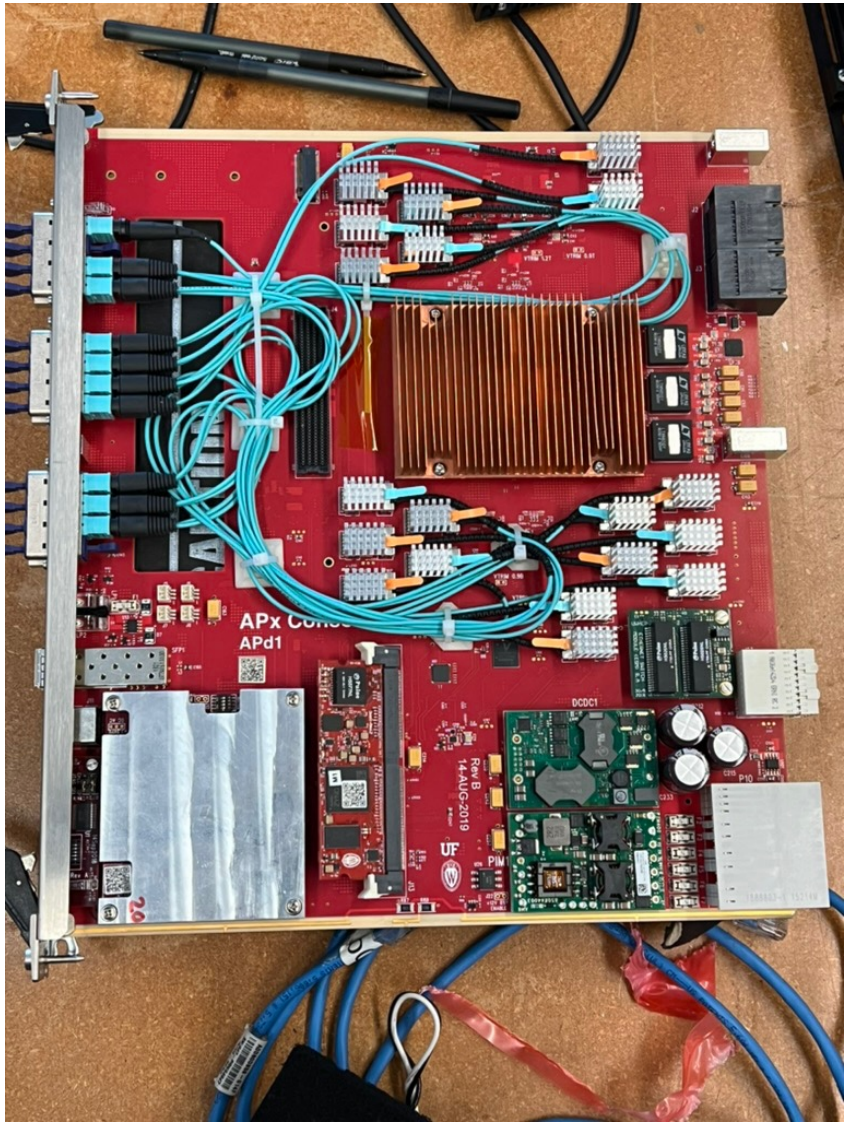
- Data Sending Sequence (for GT)

12 Jets*2 words(24 words),
HTMHT*2 words,
12 Jets with Large radius*2 words(24words),
HTMHT with Large radius*2 words
MET*2 words

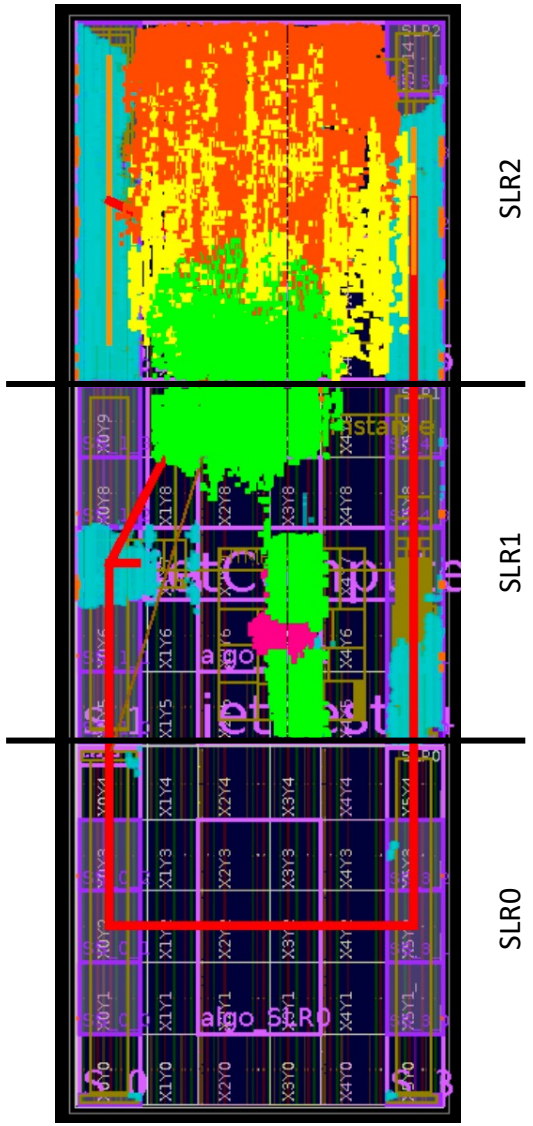
APx Firmware Implementation

HLS → FPGA

CMS APx board



- FPGA resource utilization test
  - The different colors represent the distribution of resources used by specific sub-algorithms within the Puppi Algorithm when implemented on a Xilinx Vertex UltraScale+ VU9P FPGA.
  - This visual representation aids in optimizing resource allocation and supports the efficient execution of the algorithm on the FPGA.



Junwon Oh (KHU)

Conclusion Remarks & Outlook

Challenges for HL-LHC

- Complex data representation and detector environment
- Severe computational restrictions

Our Work

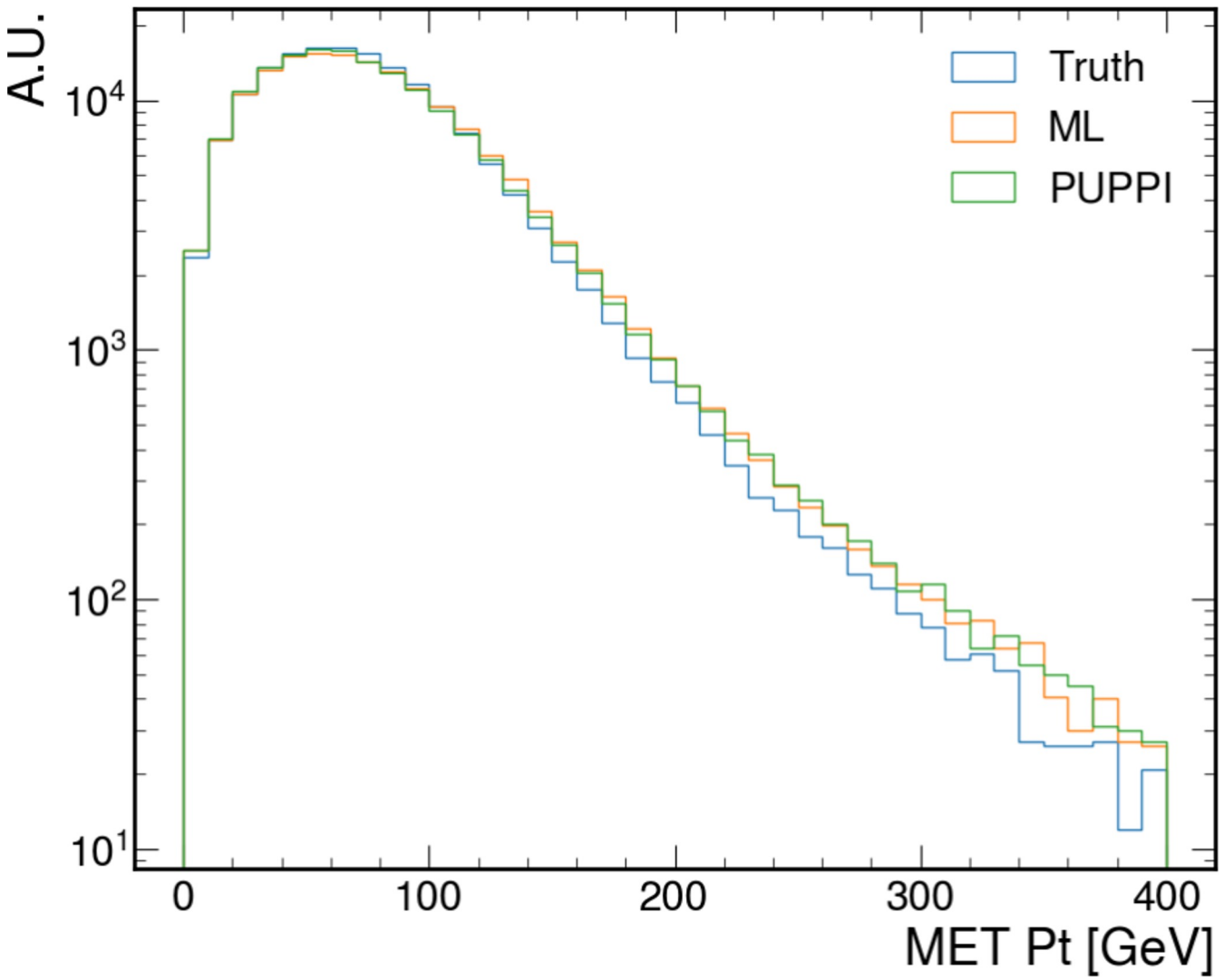
- Demonstrated real-time ML application for CMS Level-1 Trigger
- Achieved faster inference, mandatory for HEP triggering systems
- ML-based MET shows better resolution than PUPPI

Outlook

- Still room for improvement with Transformer-based models
- Plan to test with high-MET physics samples
- Deployment using *hls4ml* on FPGA

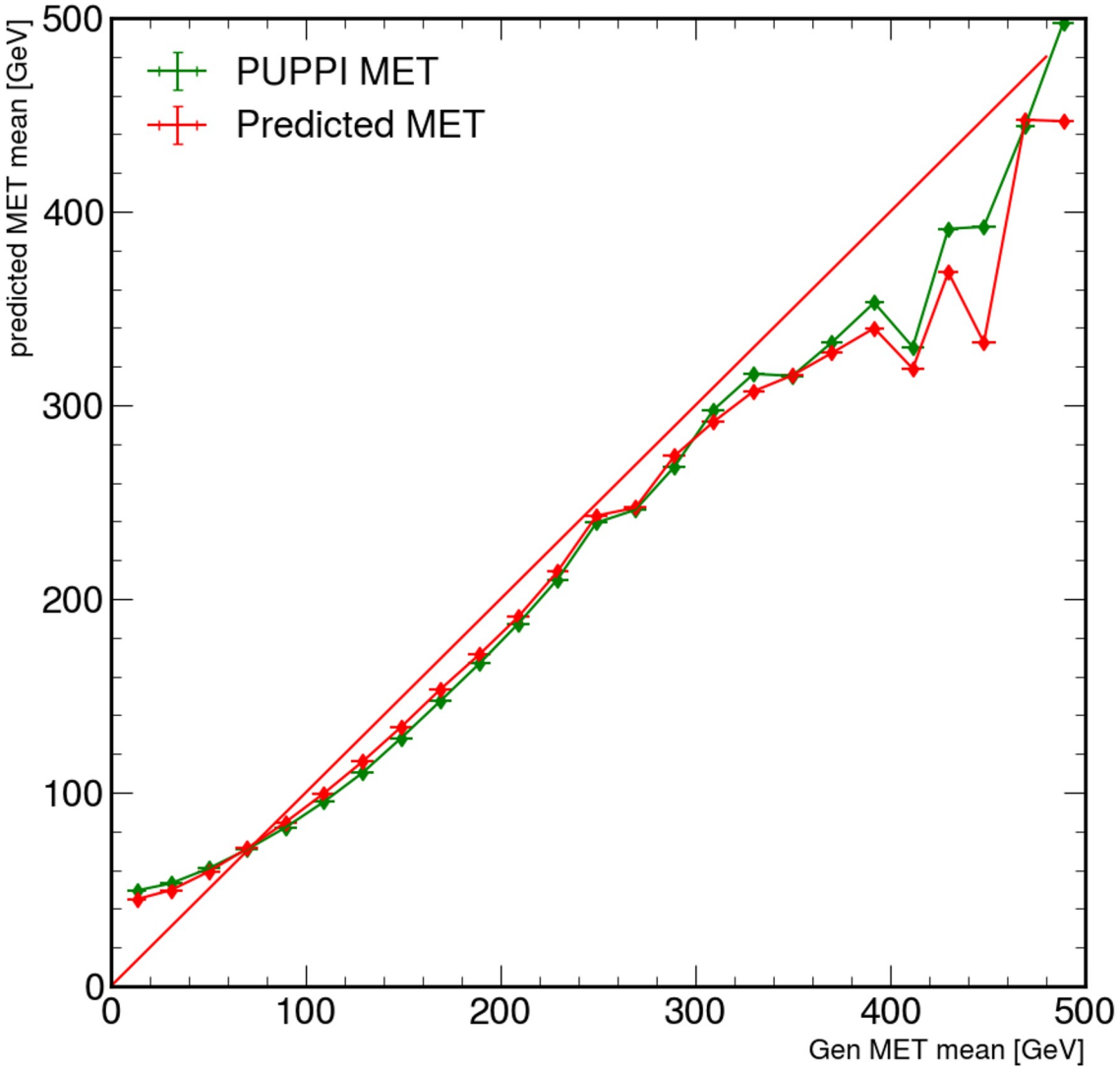
Back up

MET distribution



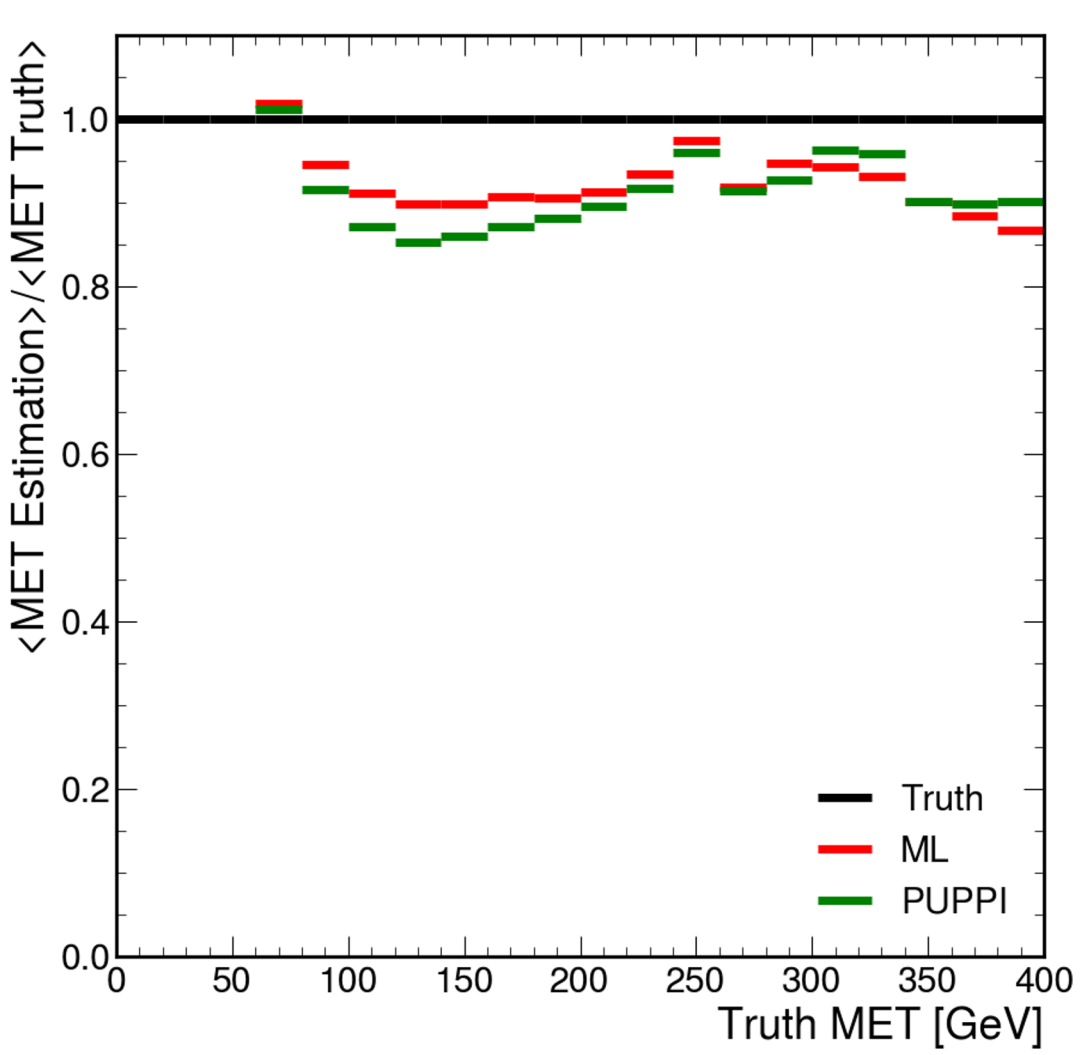
We have also observed that the MET distribution closely resembles the truth distribution, confirming its similarity.

Gen MET vs Predicted MET



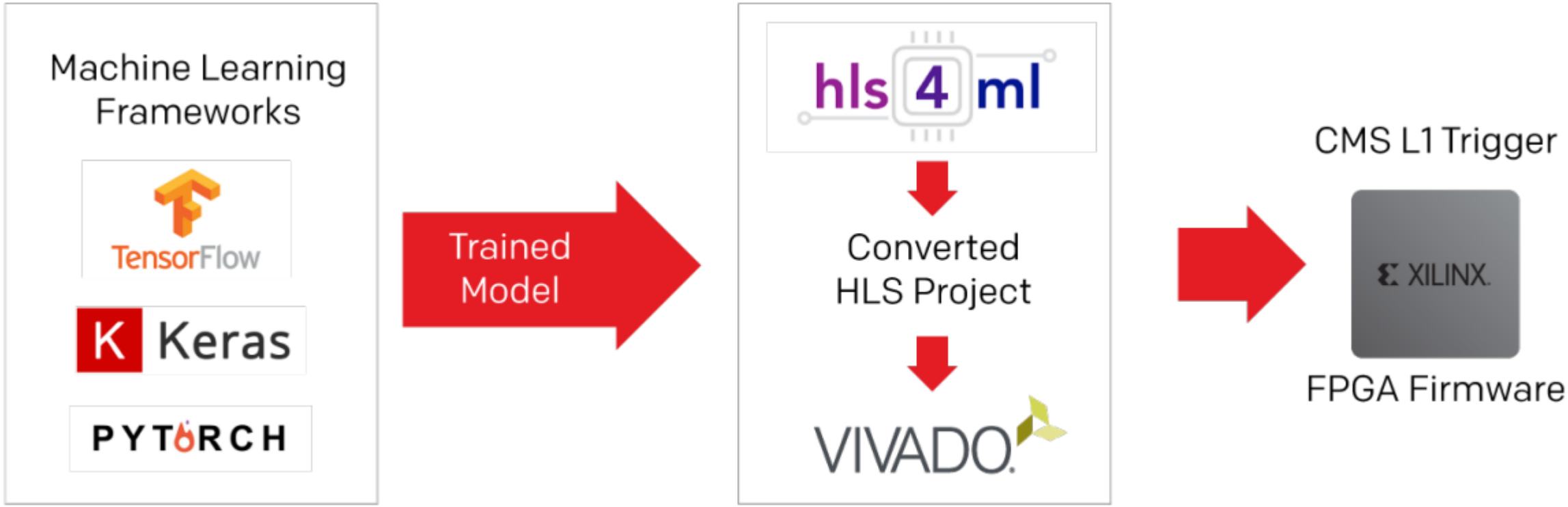
- epochs : 100
- units : 64 32 16
- model : JEDI-net (DNN)
- optimizer : Adam

MET Response



We have verified that in areas with sufficient statistics, ML MET exhibits a closer approximation to the truth value compared to PUPPI MET.

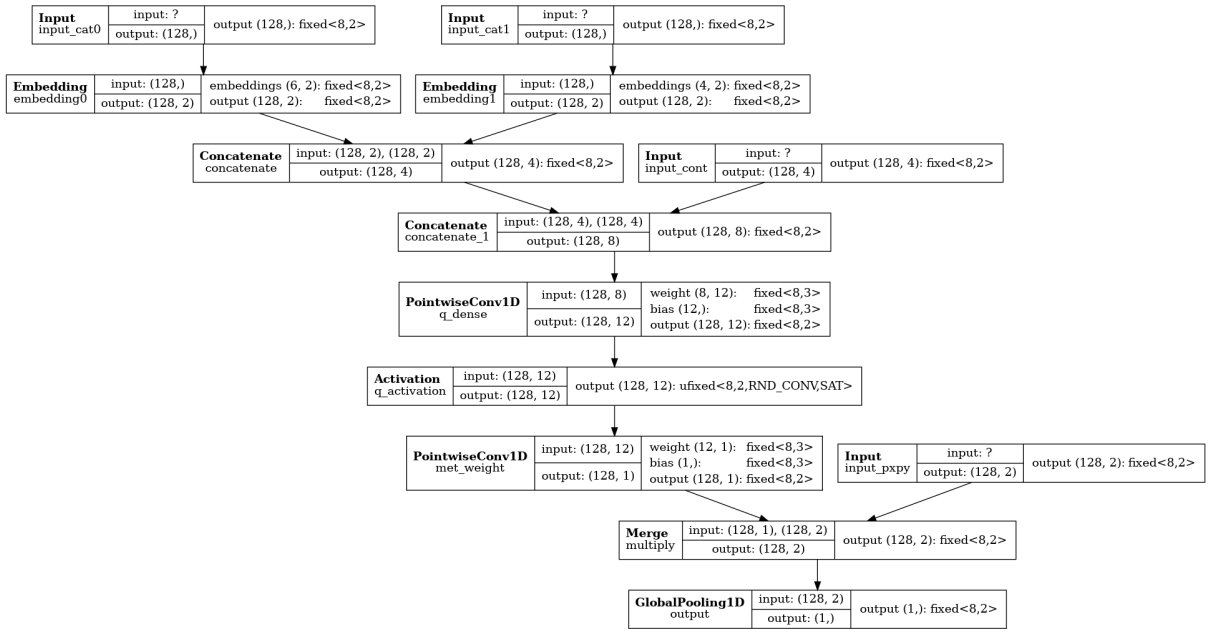
Customized hls4ml Flow Leveraging Vivado HLS



- Xilinx Vivado HLS
  - Creating machine learning algorithms for the CMS level-1 trigger.
- The hls4ml tool has a number of configurable parameters that enable users
  - Customize the space of latency, initiation interval, and resource usage tradeoffs for their application.
  - Perform the optimization through automated neural network translation and FPGA design iteration.

Demo Model Test : 1 layer Model

- Model Architecture



1 layer Model
Reduced hidden layer dimension

- Set Precision to ap_fixed<11,2> for all outputs.
- Follow the flow of the last model, but reduce the number of hidden layers.
- Set Conv1D layer to 1 and 12 nodes, unlike the last model (64,32,12).

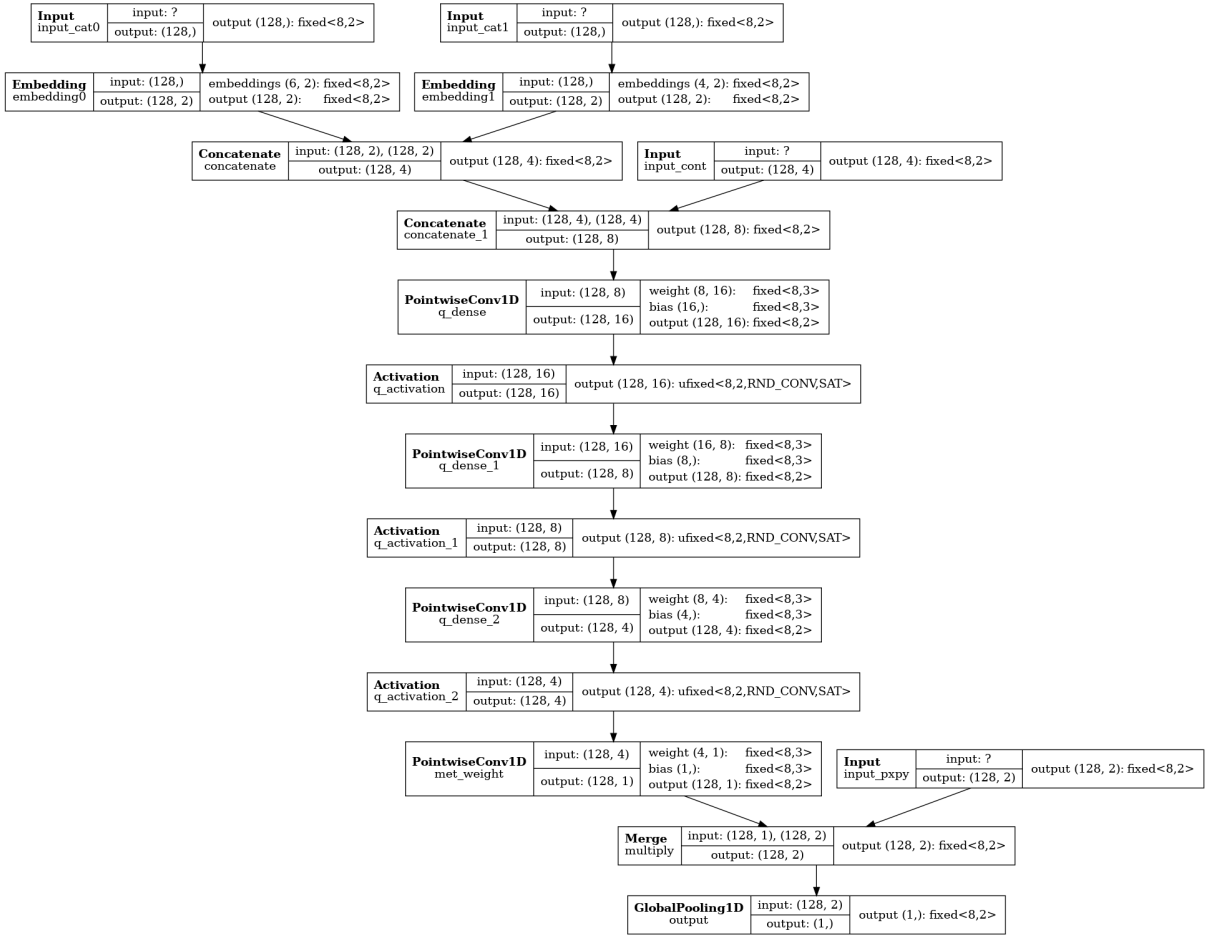
Demo Model Test : 3 layer Model

- Model Architecture

- Set Precision to ap_fixed<11,2> for all outputs.

- Follow the flow of the last model, but reduce the dimensions.

- Set Conv1D layer to (16,8,4), unlike the last model (64,32,12).



3 layers Model with reduced hidden layer dimensions