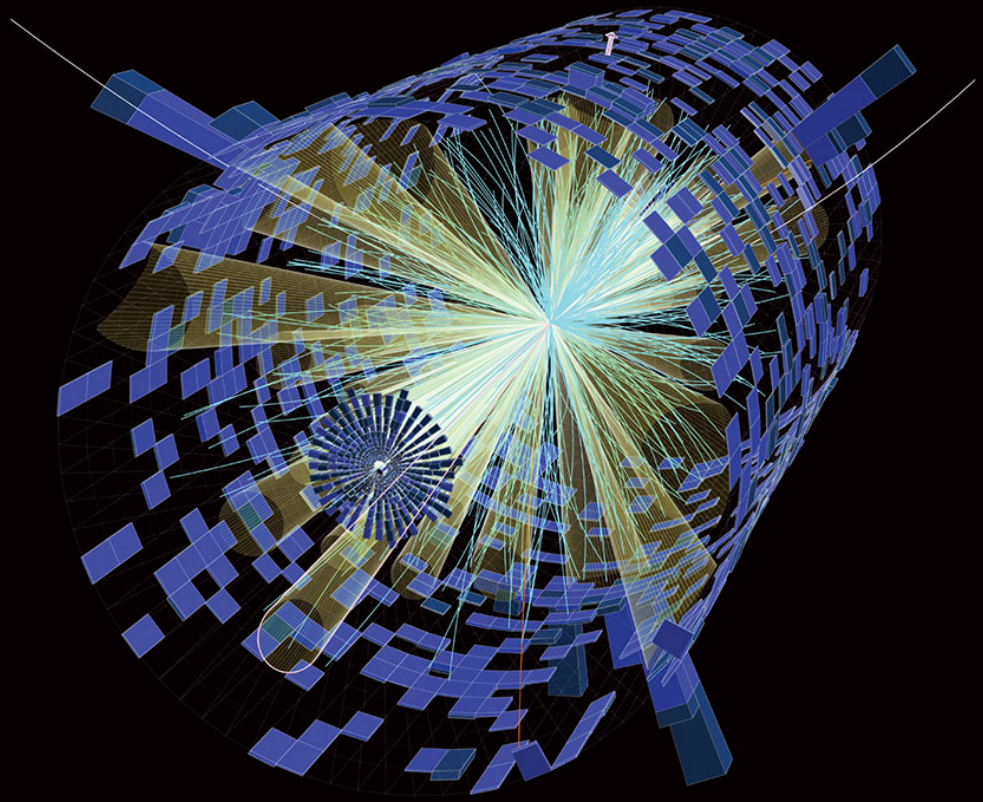
Nanosecond Jet Classification for HL-LHC



Andre Sznajder
UERJ (Brazil)

In Collaboration with:
M.Pierini(CERN)
T.Aarrestad(CERN)
S.Summers(CERN)
J.Ngadiuba(FNAL)
V.Loncar(CERN)

ML4Jets, 7 July 2021

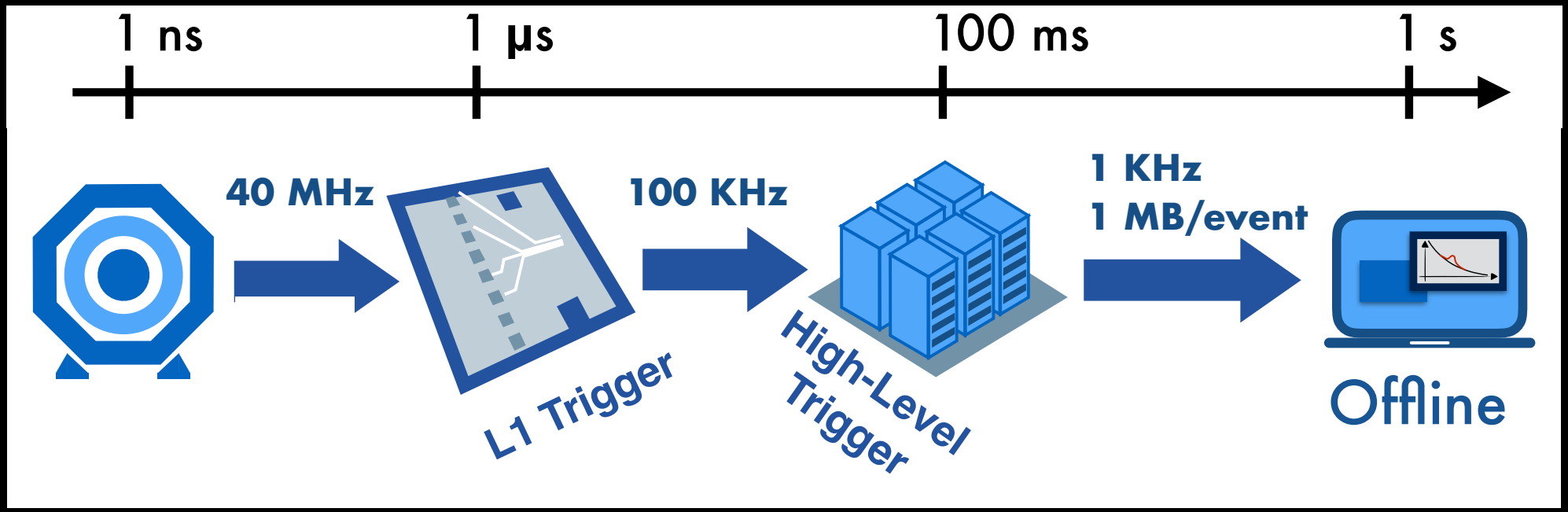
LHC Poses a Big Data Problem

- Proton beams collide at a frequency of 40 MHz
- Each collision produces $O(10^3)$ particles
- Detectors have $O(10^8)$ sensors

=> Extreme data rates of O(100 TB/s) !

LHC Event Processing

Multiple filtering stages to reduce data rates to manageable levels

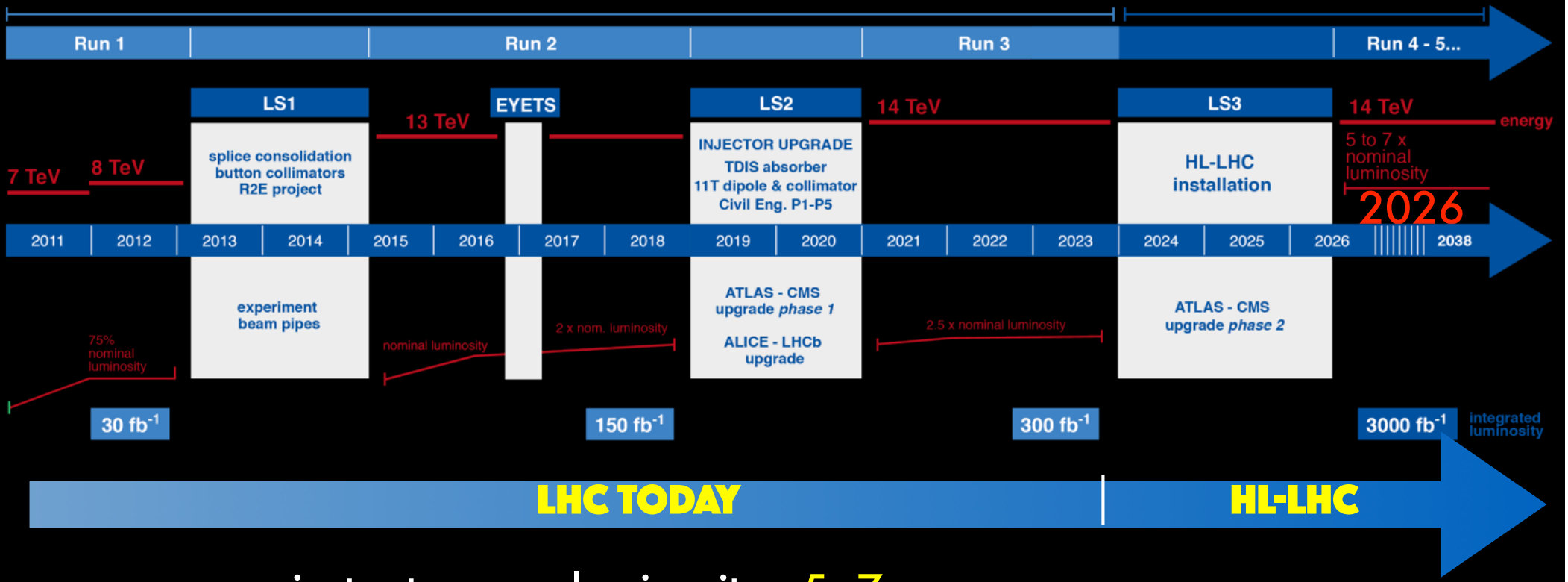


- L1 trigger absorbs O(100 TB/s)
- About 99% of events is rejected (needs high purity trigger)
- Trigger decision to be made in O(µs)

=> Latencies imposes an all FPGA design for L1 !

HL-LHC

Upgraded High-Luminosity LHC will pose major challenges



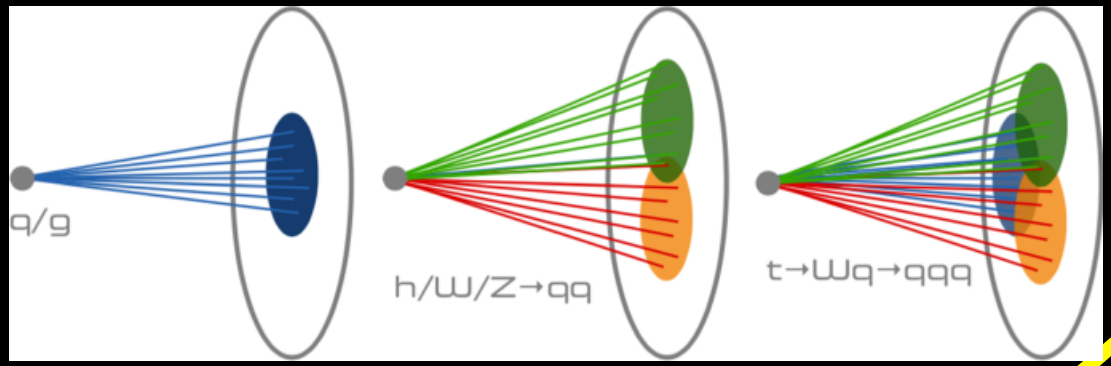
- instantaneous luminosity x5–7
- particles per collision x5
- more granular detectors with x10 channels
- more data x15

=> Event rates & datasets will increase to unprecedented levels !

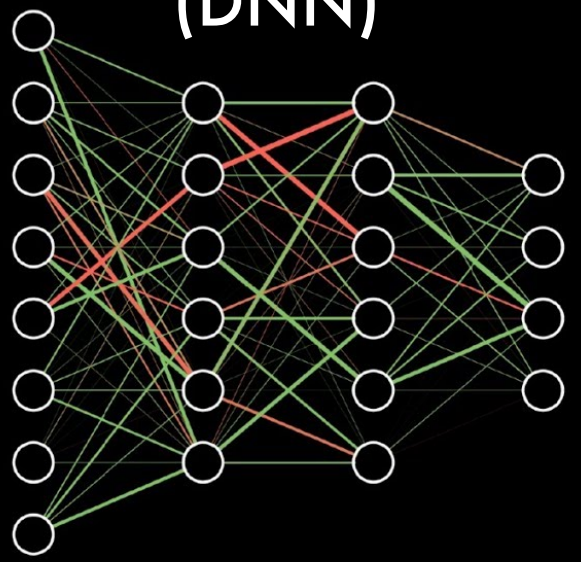
Jet ID at the L1 trigger

Proposal: recast L1 trigger problem into a ML problem !

Jet ID



Deep Neural network (DNN)



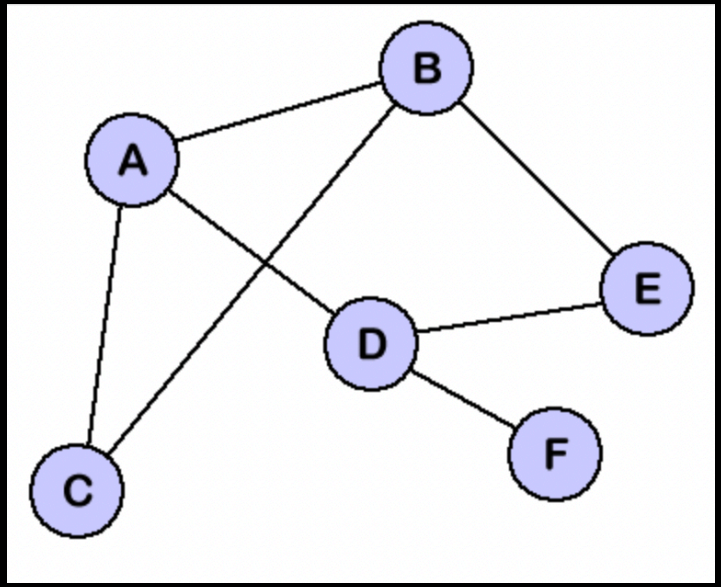
- Feed a DNN with L1 jets constituents and train it as classifier
- While training might take long time, inference is very fast

Challenge: Need to implement the DNN in a FPGA to conform with L1 !

Neural Network Models

L1 provides unordered jet constituents

- Suggests to use Graph NN as jet classifier
- constituents (nodes) features (Pt,η,ϕ)
- L1 provides fixed #constituents
  => fixed graph size



MLP

$$x_i' = \sigma\Big(\sum_j W_{ij} x_j + B_i\Big)$$

- Dense layers (fully connected neurons)
- Used as std. candle

GraphConv

$$x_i' = \sigma\Big(\bigoplus_{j \in \mathcal{N}_i} M(x_i, x_j)\Big)$$

- Convolution applied to graphs
- Dense layer for ID

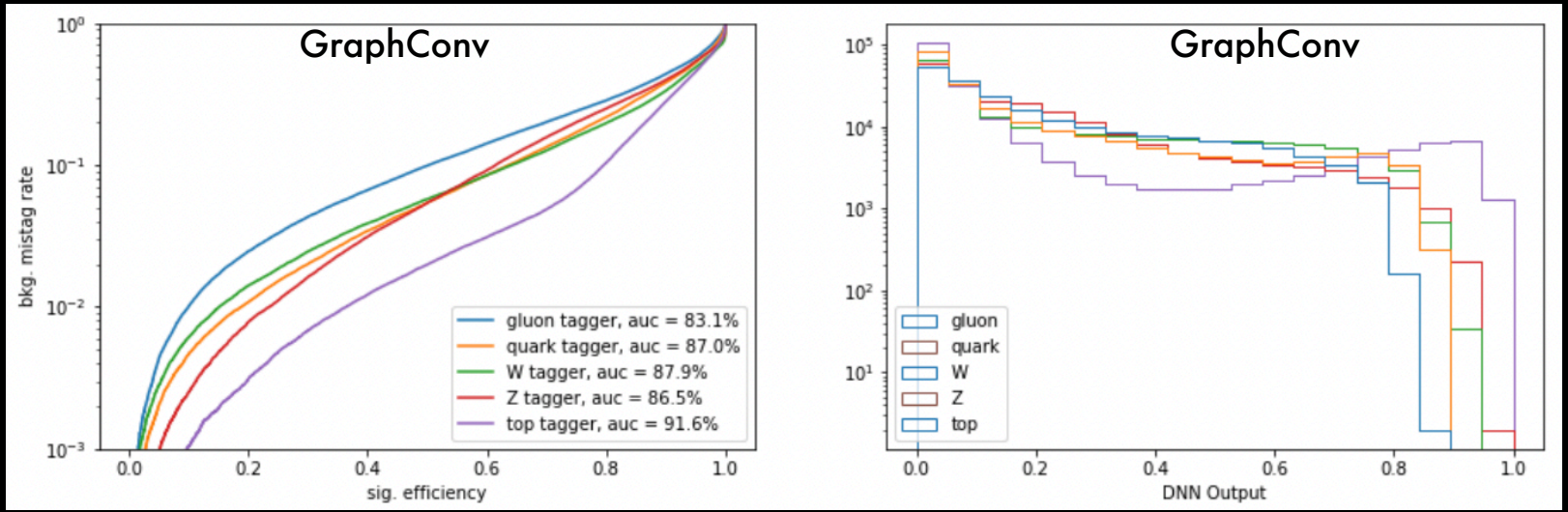
InteractionNetwork

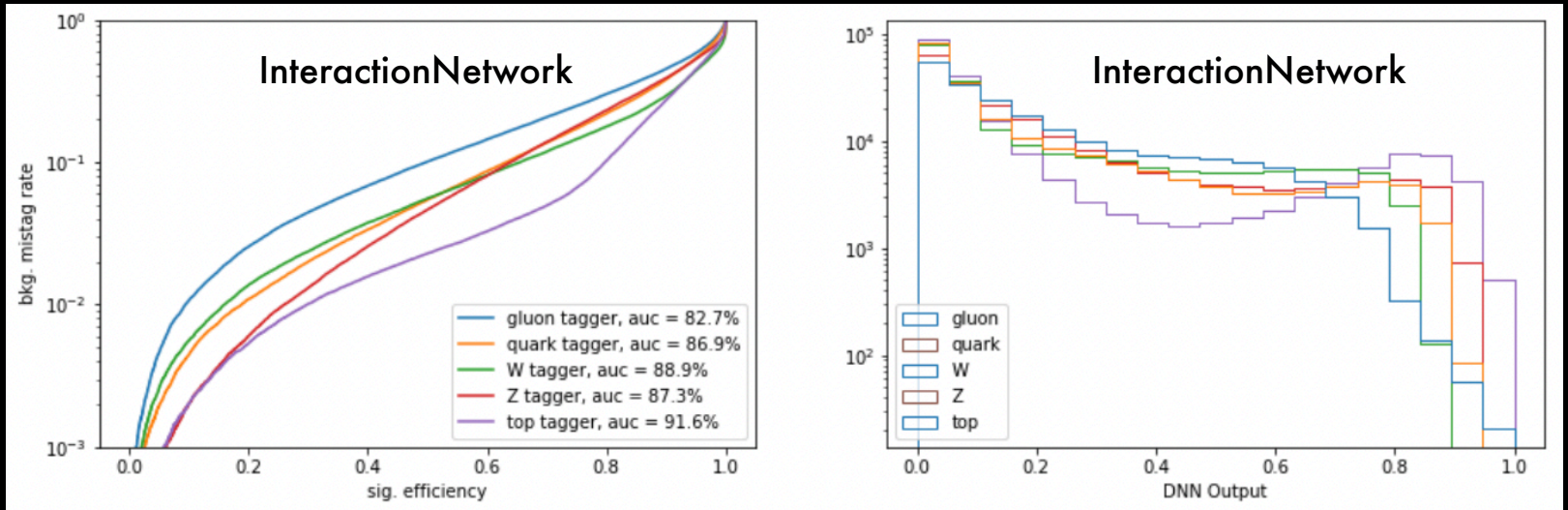
$$x_i' = \sigma\Big(\bigoplus_{j \in \mathcal{N}_i} M(x_i, x_j, e_{ij})\Big)$$

- Most expressive GNN
- Use fully connected graph (small # const.)

GraphConv & Interaction Network Jet ID Performance

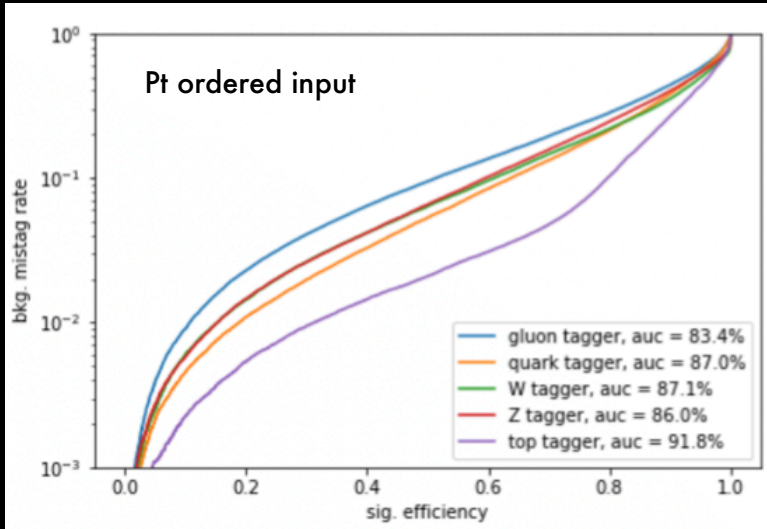
GNNs performance are similar, but GraphConv uses less resources and is faster

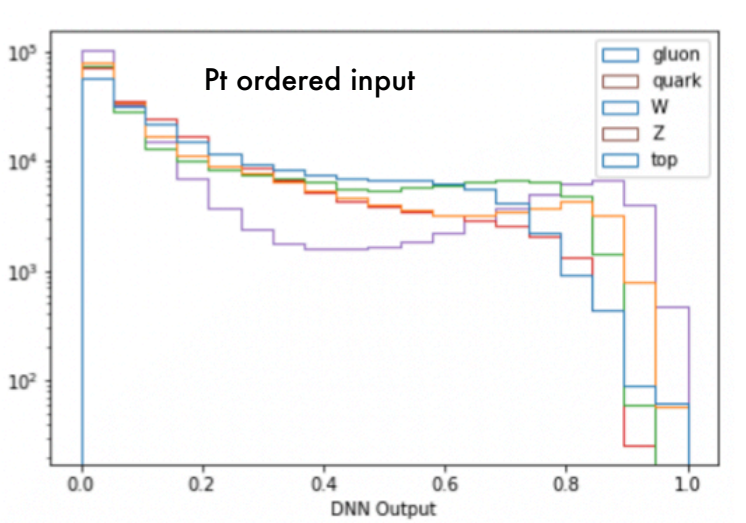


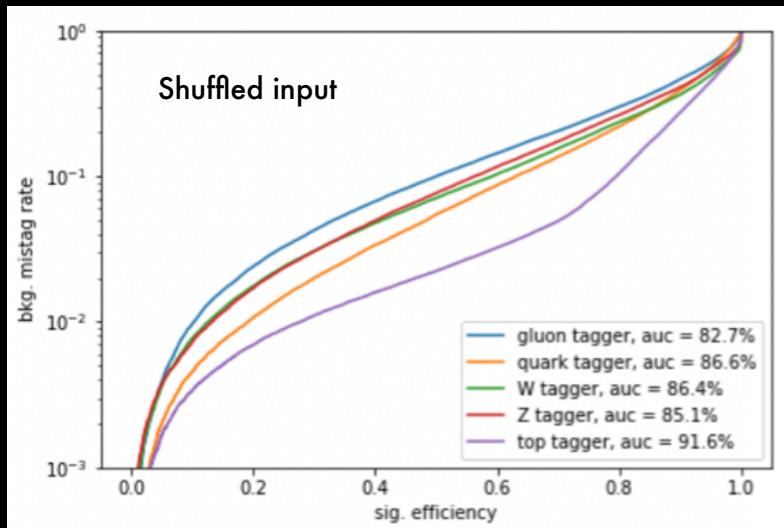


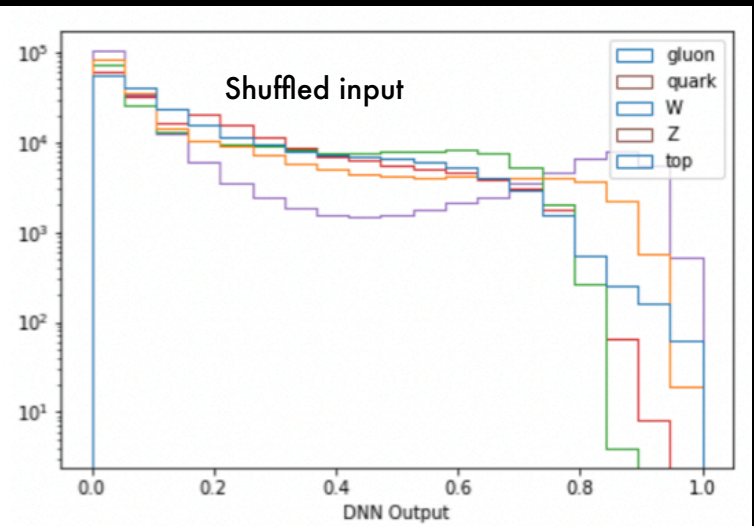
MLP Jet ID Performance

MLP has the similar performance as GNNs and it's almost insensitive to Pt order of constituents









QKeras : Quantization Aware Training

QKeras is a library for quantization aware training (QAT) of Keras models, allowing the study of the quantisation impact on neural networks

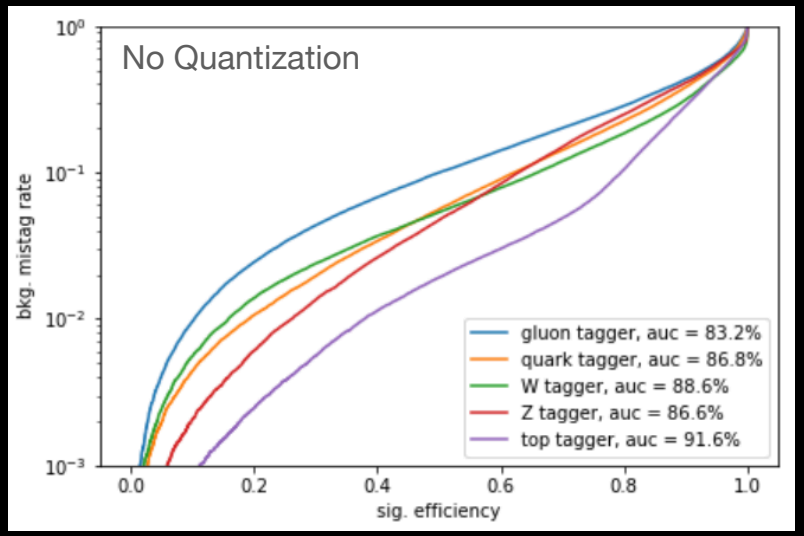
- Simple replacement of Keras layers
- Heterogenous quantization (per layer, per parameter)
- Quantizes weights, bias & activations in forward pass
- Uses full precision (FP32) in back-propagation

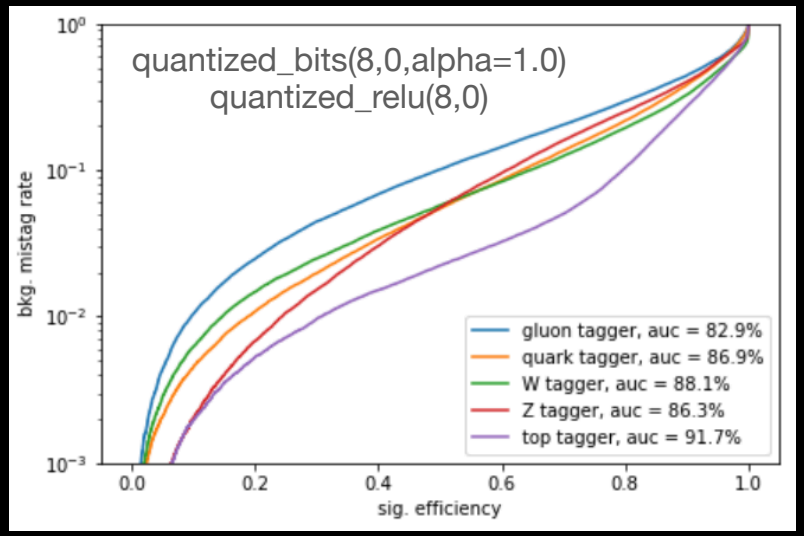
https://github.com/google/qkeras

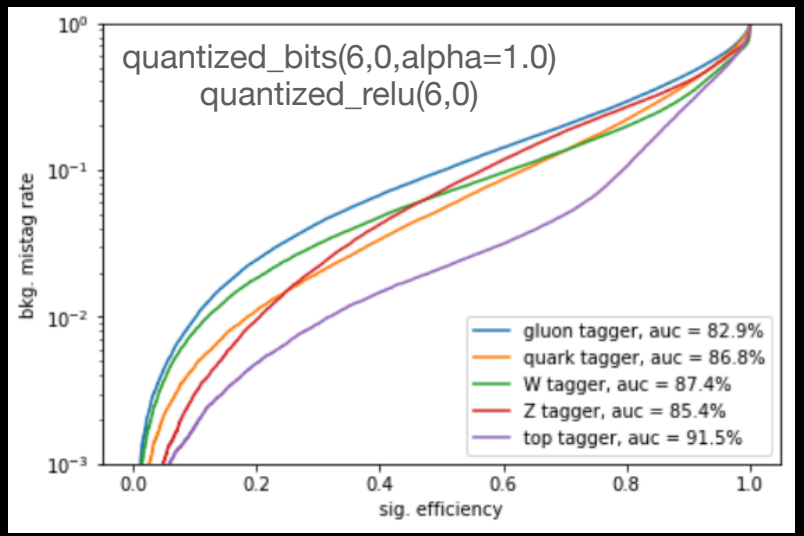
```
from tensorflow.keras.layers import Input, Activation
from qkeras import quantized_bits
from qkeras import QDense, QActivation
from qkeras import QBatchNormalization

x = Input((16))
x = QDense(64,
    kernel_quantizer = quantized_bits(6,0,alpha=1),
    bias_quantizer   = quantized_bits(6,0,alpha=1))(x)
x = QBatchNormalization()(x)
x = QActivation('quantized_relu(6,0)')(x)
x = QDense(32,
    kernel_quantizer = quantized_bits(6,0,alpha=1),
    bias_quantizer   = quantized_bits(6,0,alpha=1))(x)
x = QBatchNormalization()(x)
x = QActivation('quantized_relu(6,0)')(x)
x = QDense(32,
    kernel_quantizer = quantized_bits(6,0,alpha=1),
    bias_quantizer   = quantized_bits(6,0,alpha=1))(x)
x = QBatchNormalization()(x)
x = QActivation('quantized_relu(6,0)')(x)
x = QDense(5,
    kernel_quantizer = quantized_bits(6,0,alpha=1),
    bias_quantizer   = quantized_bits(6,0,alpha=1))(x)
x = Activation('softmax')(x)
```

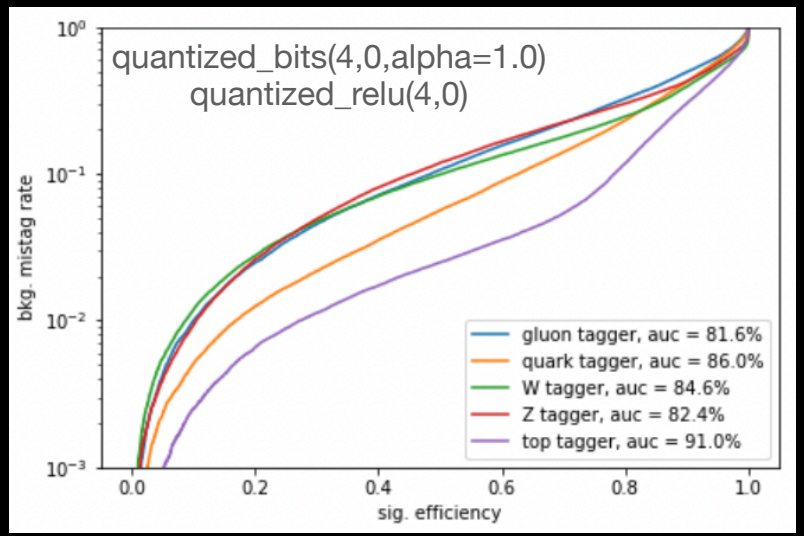
GraphConv Quantization Aware Training

Jet ID perfomance using quantization (8,6,4 bits) of Conv and Dense layers and Relu activations



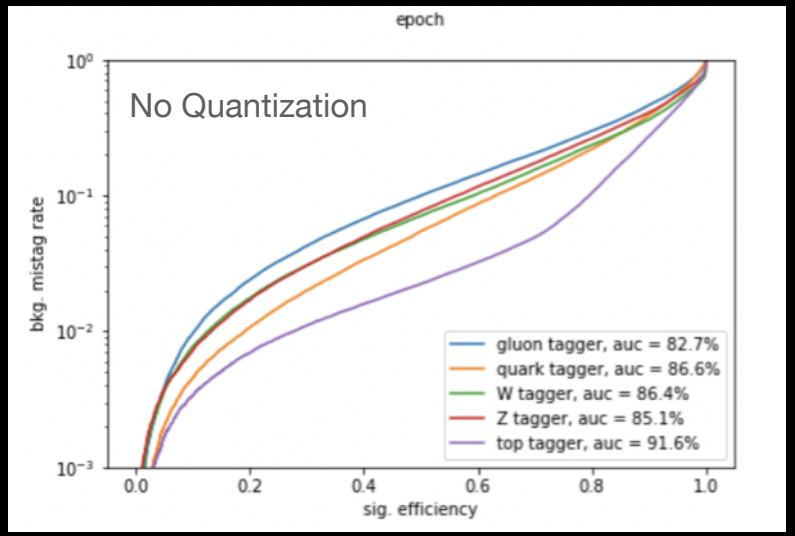


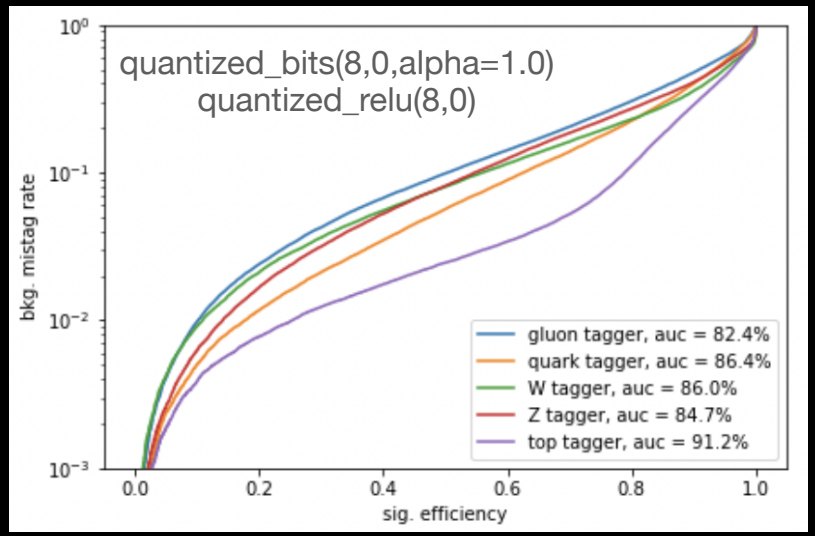


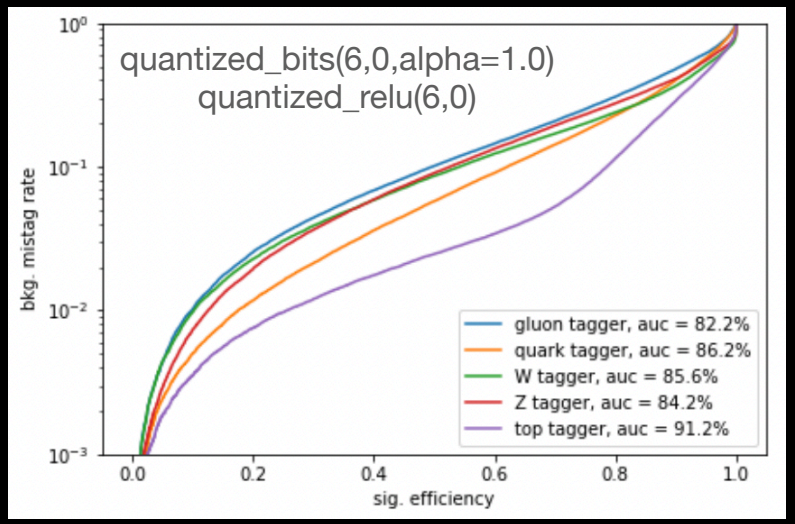


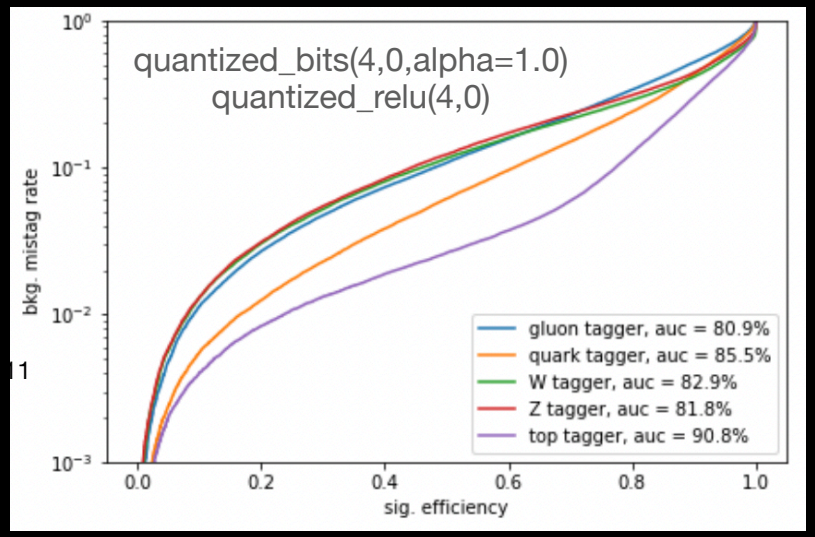
MLP Quantization Aware Training

Jet ID perfomance using quantization (8,6,4 bits) of Dense layers and Relu activations



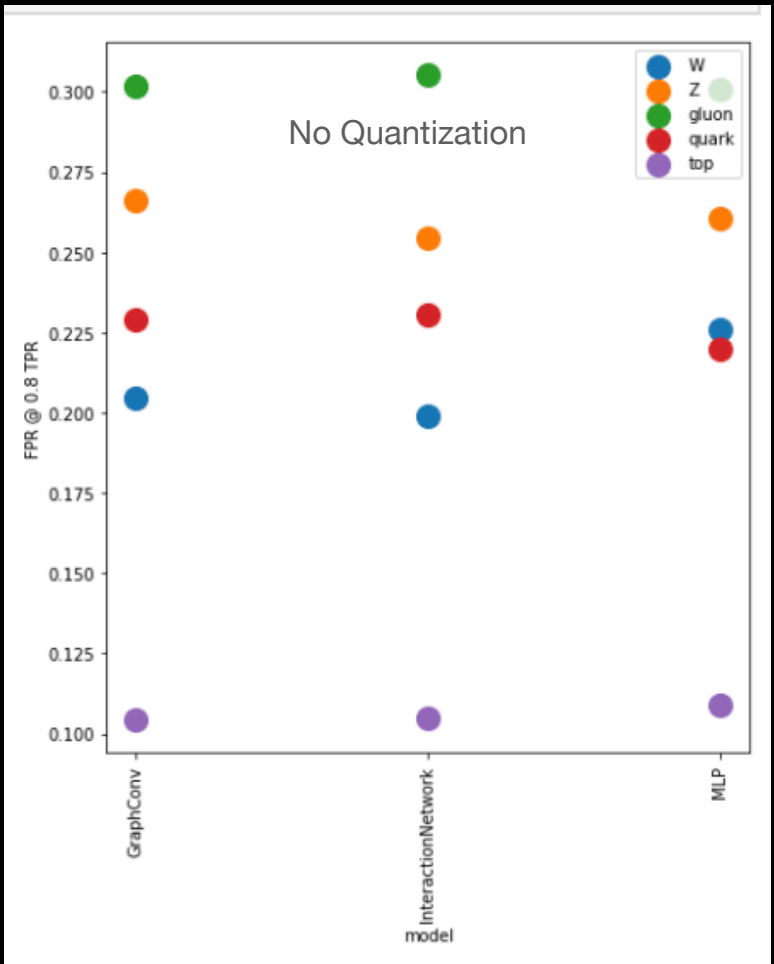


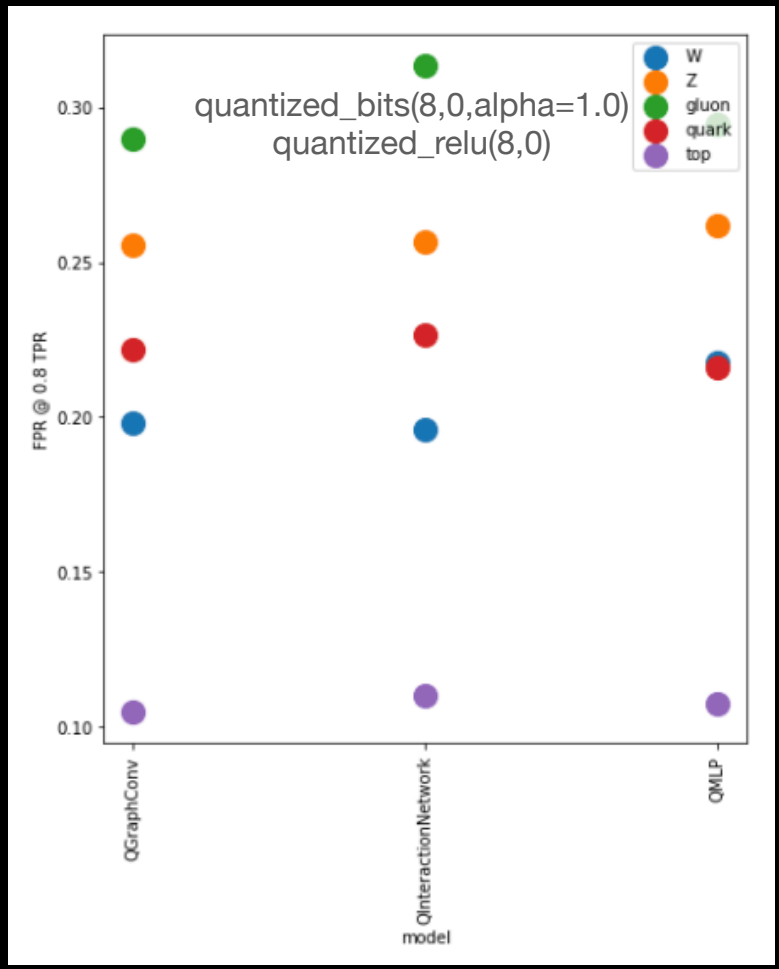




Jet ID performance: FPR@0.8TPR

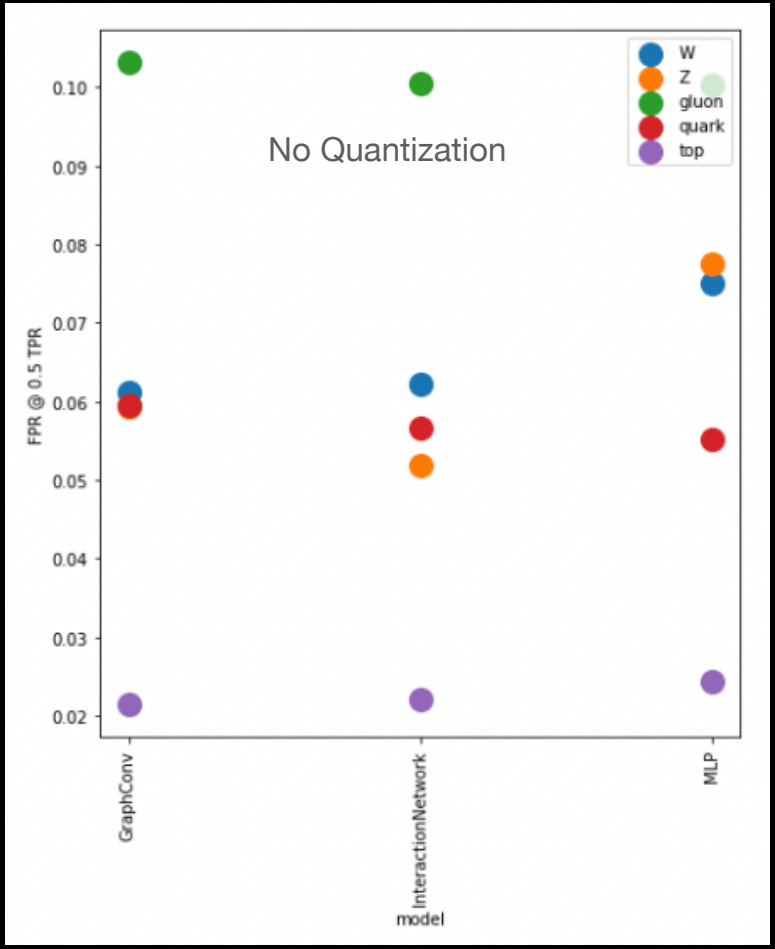
Jet tagging performance FPR@0.8TPR of different models, for (no quantization) X (8 bits quantization) scenarios

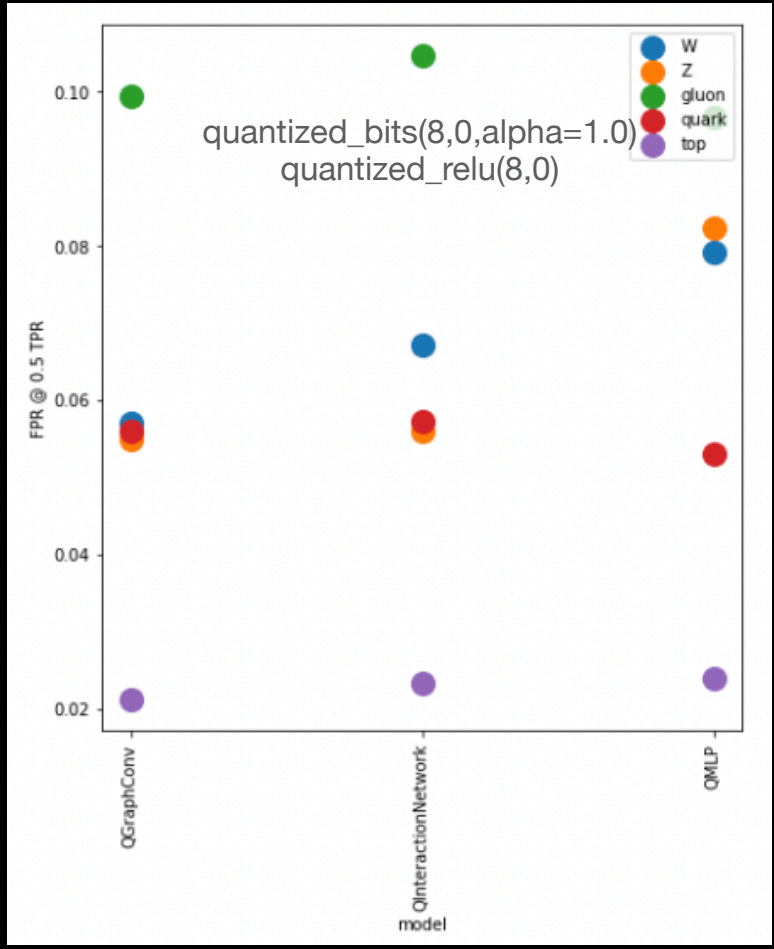




Jet ID performance: FPR@0.5TPR

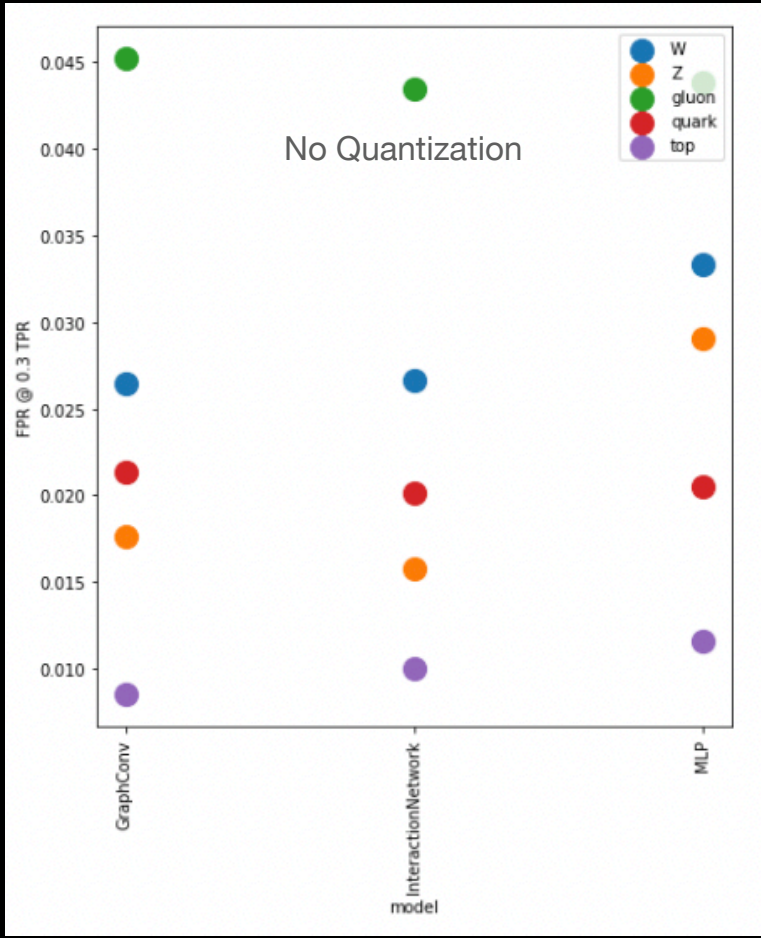
Jet tagging performance FPR@0.5TPR of different models, for (no quantization) X (8 bits quantization) scenarios

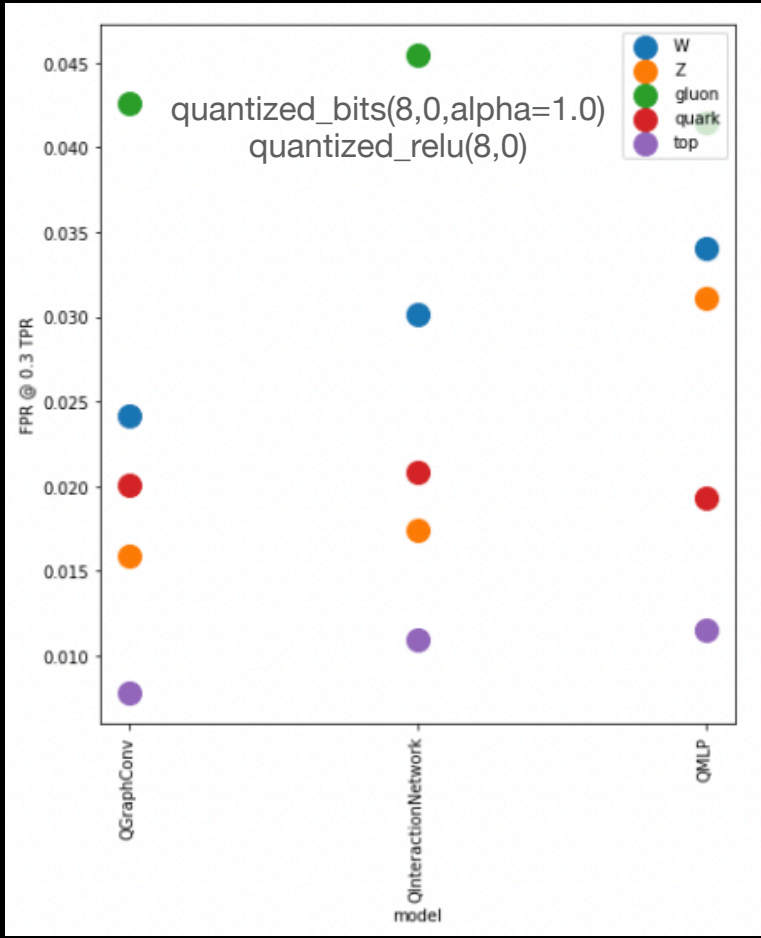




Jet ID performance FPR@0.3TPR

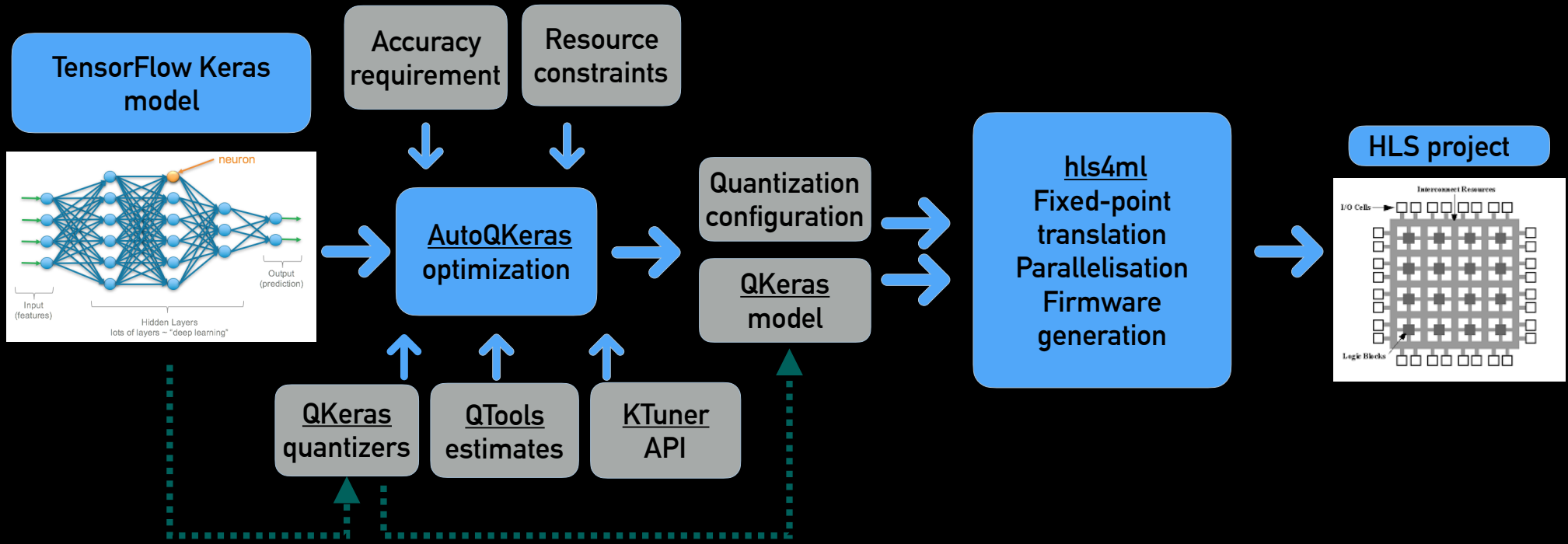
Jet tagging performance FPR@0.3TPR of different models,
for (no quantization) X (8 bits quantization) scenarios





HLS4ML

HLS4ML is a Python package for machine learning inference in FPGAs. It creates a firmware implementation of the machine learning model (Ex:QKeras) allowing fast development and optimization



https://github.com/fastmachinelearning/hls4ml

HLS4ML: GraphConv Synthesis

FPGA (Xilinx Virtex Ultrascale 9+) resource usage and latency for quantisation scenarios: 8,6,4 bits

# Bits	DSP	LUT	FF	BRAM	Latency
8	43 (0.6%)	80977 (6.8%)	11987 (0.5%)	1.5 (0.07%)	85.0ns ii:13
6	14 (0.2%)	39262 (3.3%)	9035 (0.4%)	1.5 (0.07%)	95.0ns ii:15
4	7 (0.1%)	19332 (1.6%)	5555 (0.2%)	1.5 (0.07%)	95.0ns ii:15

HLS4ML: MLP Synthesis

FPGA (Xilinx Virtex Ultrascale 9+) resource usage and latencyfor quantisation scenarios: 8,6,4 bits

# Bits	DSP	LUT	FF	BRAM	Latency
8	300 (4.4%)	71917 (6.1%)	10555 (0.4%)	1.5 (0.07%)	40.0ns ii:1
6	50 (0.7%)	44290 (3.7%)	7768 (0.3%)	1.5 (0.07%)	40.0ns ii:1
4	21 (0.3%)	24807 (2.1%)	4888 (0.2%)	1.5 (0.07%)	40.0ns ii:1

Summary

Current HEP paradigm is not sustainable for future requirements.
We present a solution for a Jet ID L1 trigger at HL-LHC :

- ML+FPGA approach (performant, parallelizable, industry support)
- Accessible to non HDL experts (QKeras , HLS4ML)
- DNN models (MLP, GraphConv & InteractionNetwork) have similar perfomance
- MLP is insensitive to constituent Pt order (important at L1)
- HLS4ML shows models latency well within L1 trigger constrains
- FPGA resource usage compatible with Xilinx Virtex Ultrascale 9+ specs
- Choice between models (MLP vs GraphConv) might depend on FPGA being latency or DSP limited