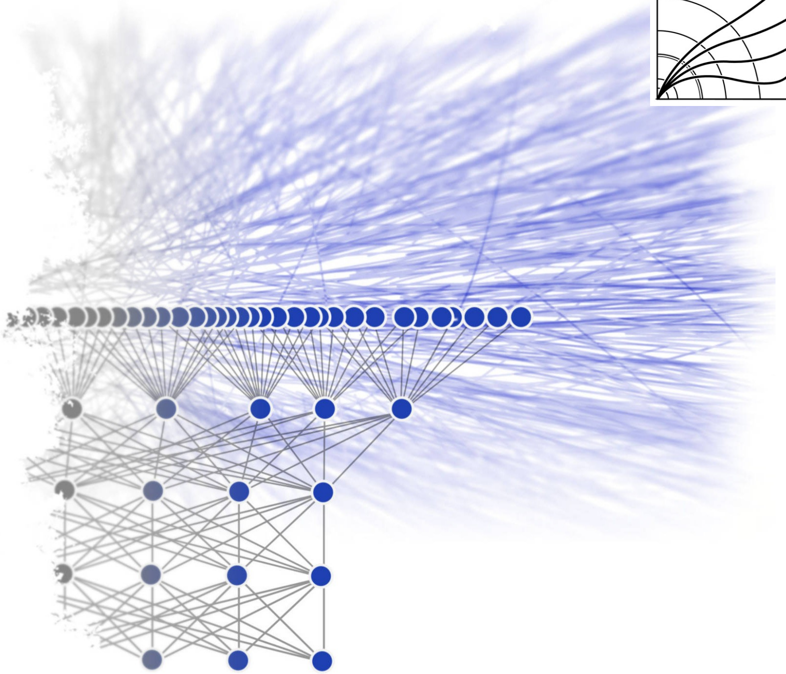
Neural network based primary vertex reconstruction



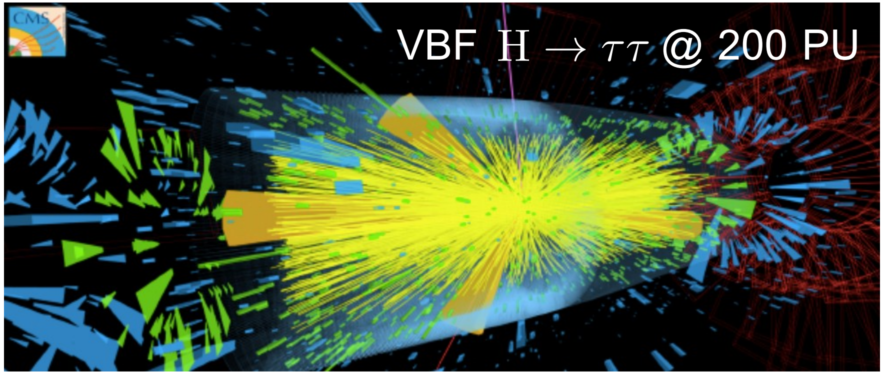
5th IML Workshop

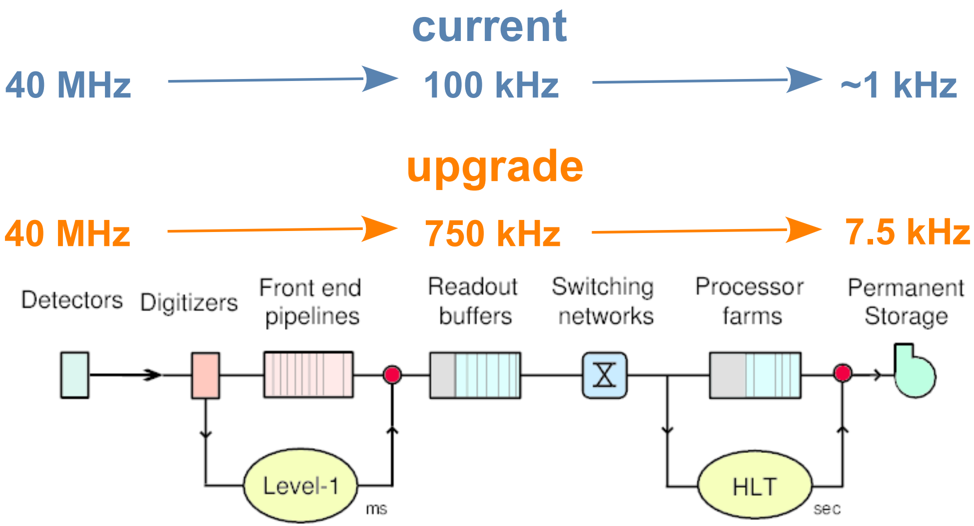
Matthias Komm (DESY),
Sioni Summers, Maurizio Pierini, Marcel Rod, Vladimir Loncar (CERN),
Chris Brown, Benjamin Radburn-Smith, Alex Tapper (Imperial College)

Introduction: CMS upgrade

- upgrade to HL-LHC (Run 4 start ~2029)
  - significant increase in instantaneous luminosity
    - $L_{\text{design}} = 10^{34}\ \text{cm}^{-2}\text{s}^{-1}$
    - Run 2 & 3: $2 \times L_{\text{design}}$
    - Run 4: $7.5 \times L_{\text{design}}$
  - expect about 200 additional interactions (pileup) each bunch crossing

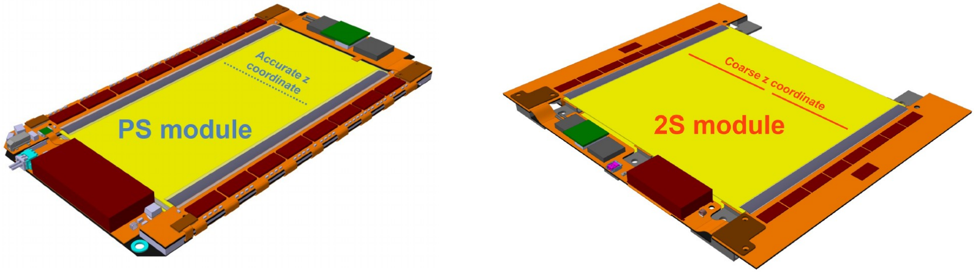
- CMS upgrade foresees tracking in first event-level trigger (L1)
  - track reconstruction at 40 MHz
  - reconstruct primary vertex
    $\rightarrow$ crucial for separating hard interaction from pileup





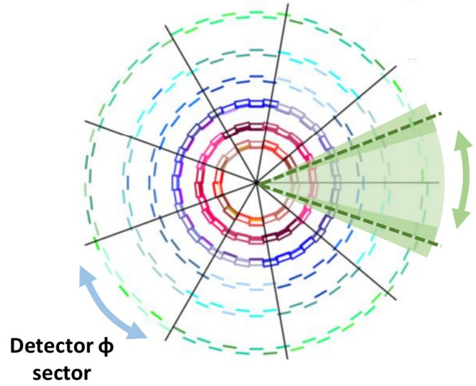
Track reconstruction at L1

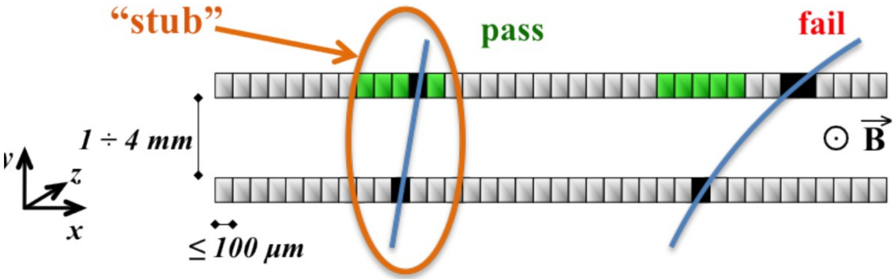
- unique p_{T} module concept
  - contains two closely spaced sensors (pixel or strip)
  - can correlated hits between sides to detect passing particles ($p_{\mathrm{T}} > 2\ \mathrm{GeV}$)
    $\rightarrow$ charged particle track stubs
- track reconstruction
  - reconstruct tracks by combining stubs
  - implemented in FPGAs
  - heavily parallelized by subdividing tracker into 9 φ-sectors



top side: 2D pixel module
bottom side: 1D strip module

top & bottom side:
1D strip modules



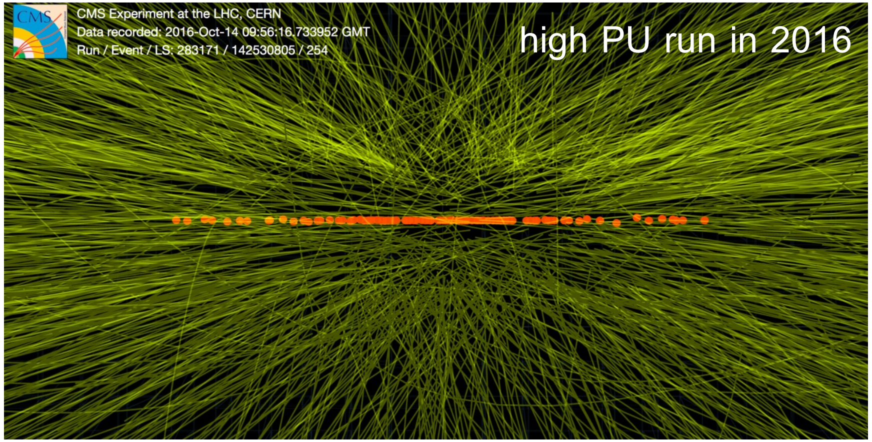


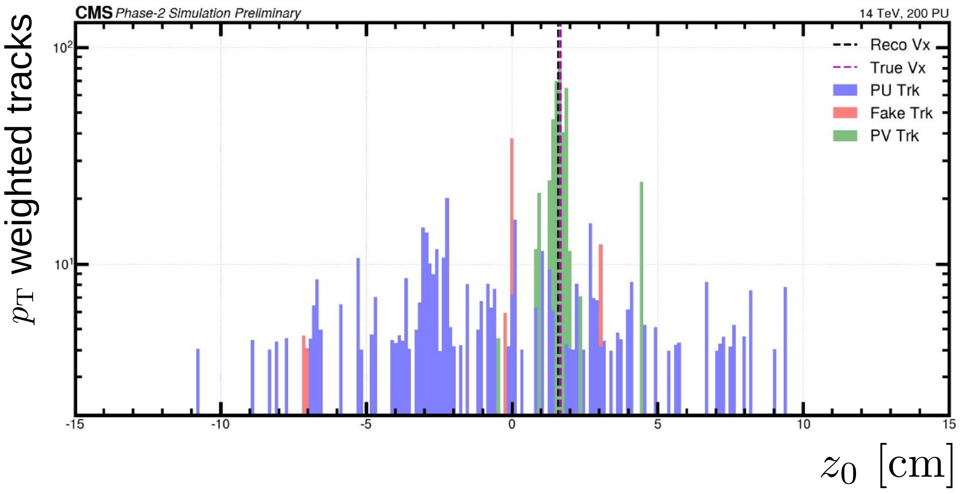
charged particle track stubs

Primary vertex reconstruction

- problem
  - grouping tracks into spatial clusters (=vertices)
  - offline CMS reconstruction uses deterministic annealing to cluster tracks into vertices
    → too complex for L1 trigger
- histogram approach
  - create histogram of z_0 from tracks with equidistant 256 bins
  - weight each entry by track p_T
  - take the middle of the 3 consecutive highest bins as z position of primary vertex
  - z_0 window to associate tracks to vertex; other tracks are treated as pileup

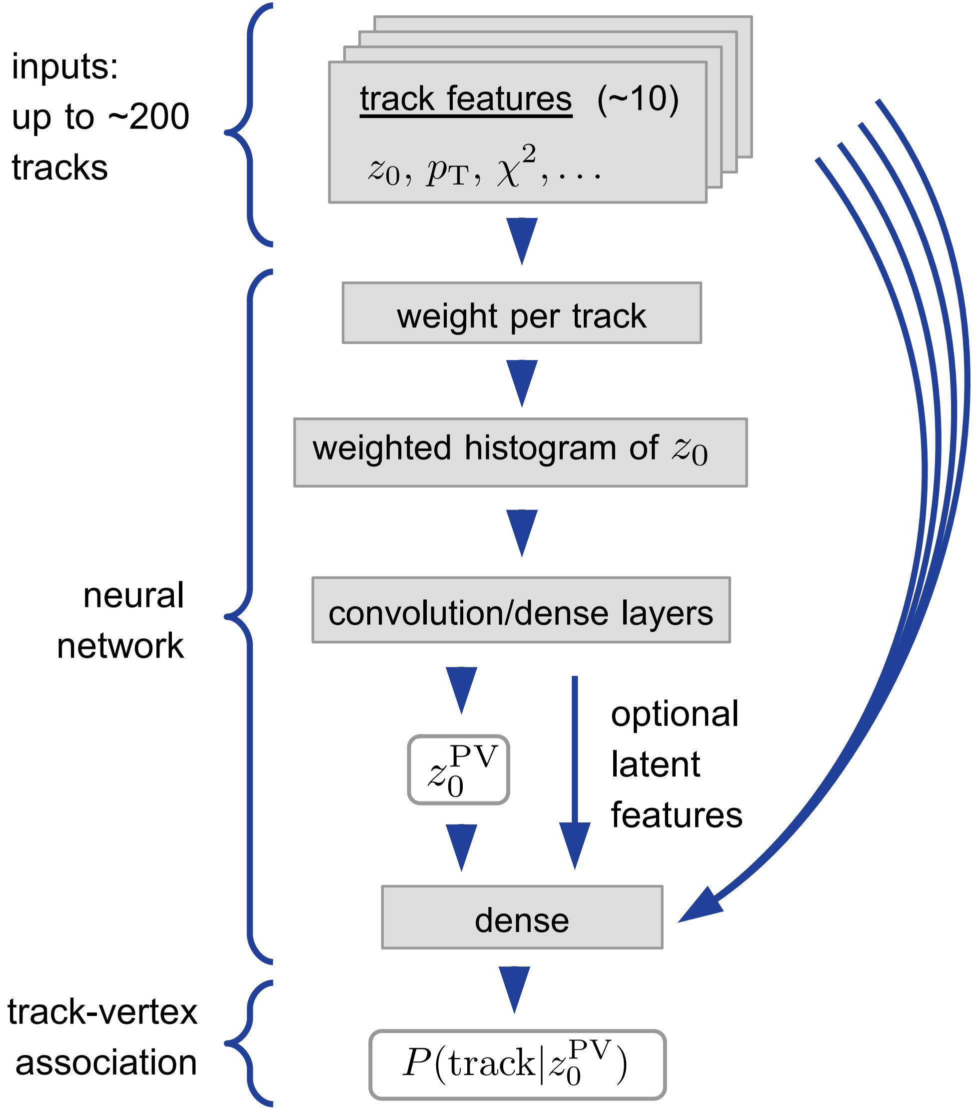
→ very crude BUT simple, fast & lightweight!





End-to-end approach

- motivation
  - optimally explore track features
  - optimize for track association throughout
    → study potential of using ML
- histograms
  - created as part of the neural network using a given set of values & weights
  - weights can be result of preceding neural network layers
    → weight function is trainable
- targets
  - vertex position z_0 & association
  - optional: learn latent variables for track-vertex association likelihood



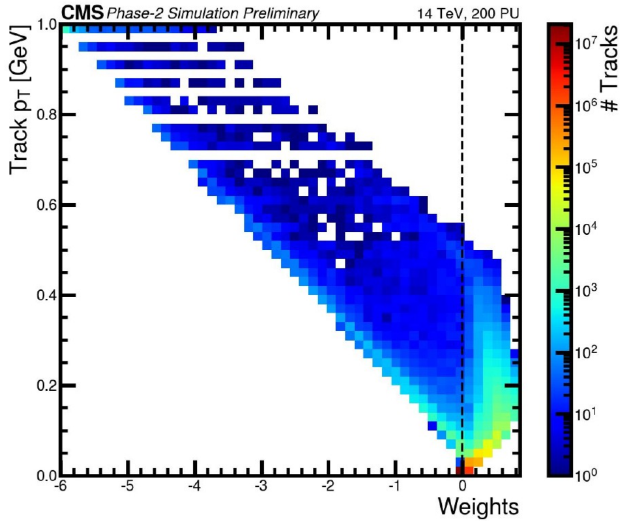
Learning track weights

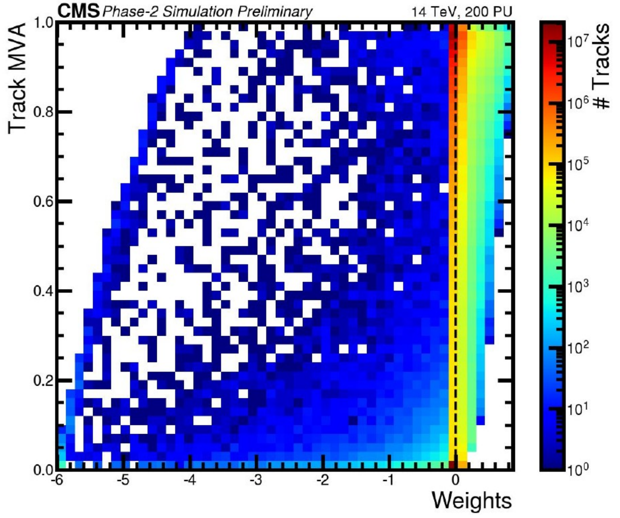
- differentiable histogram
  - implemented custom operation for TensorFlow
  - partial derivatives for backpropagation of gradients per bin i

bin content: $h_i = \sum_{j}^{\text{tracks}} \delta(j \in \text{bin } i) \times w(p_{\mathrm{T},j}, \eta_j, \chi_j^2, \ldots)$

gradients: $\frac{\partial h_i}{\partial \vec{w}} = \sum_{j}^{\text{tracks}} \delta(j \in \text{bin } i)$; $\frac{\partial h_i}{\partial \vec{z}_0} = 0$ → easy

- weight function
  - weight can capture complex correlation of features e.g. linear with p_{T}, anticorrelated with χ^2, etc.
  - easy to extend to new features in the future
  - track weights can be negative; ignored in histogram → effectively removes "unimportant" tracks (e.g. pileup)





Synthesis

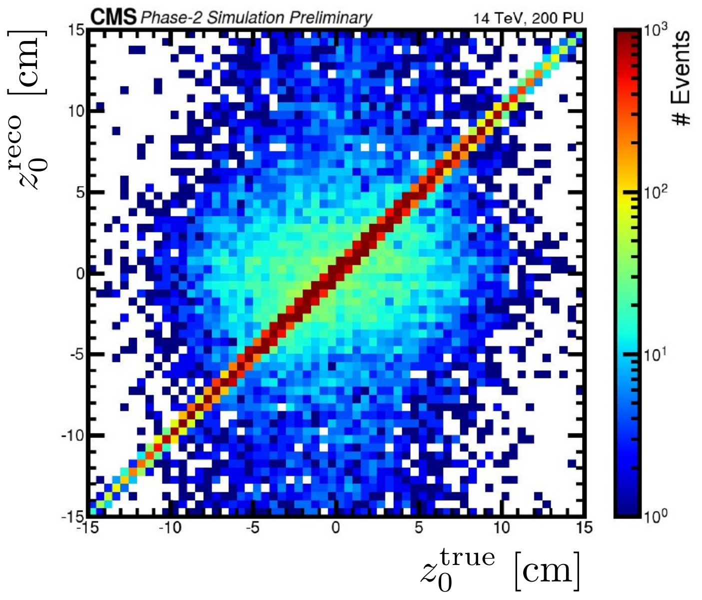
- NN is trained end-to-end BUT synthesized in 4 parts:
  track weight, histograming, pattern recognition, track-vertex association
- using hls4ml to translate keras model into vivado HLS → VHDL
- multiple instances of weight & association parts foreseen; final wiring in VHDL
- FPGA: Xilinx VU9P @ 360 MHz
- network pruned & quantized (fixed-point instead of floats) using QKeras
  → huge reduction in resources; in particular DSPs

	Latency (ns)	Initiation Interval (ns)	LUTs %	DSPs %	BRAMs %	FFs %
NN Weight	22	2.7	0.14	1.11	0.00	0.04
QNN Weight	14	2.7	0.05	0.00	0.00	0.02
NN Pattern	58	51	4.27	3.74	5.28	3.22
QNN Pattern	42	35	4.43	0.00	5.28	3.15
NN Assoc.	30	2.7	0.63	5.98	0.00	0.15
QNN Assoc.	25	2.7	0.44	0.83	0.00	0.13

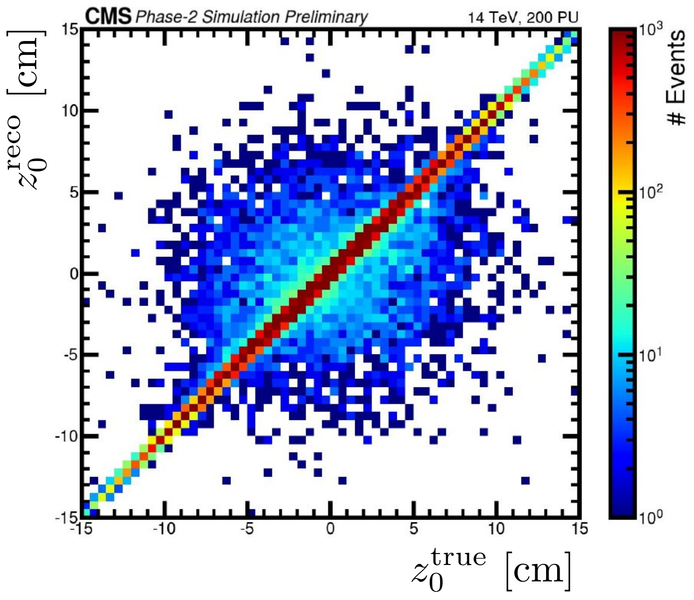
hls4ml

Performance: Primary vertex

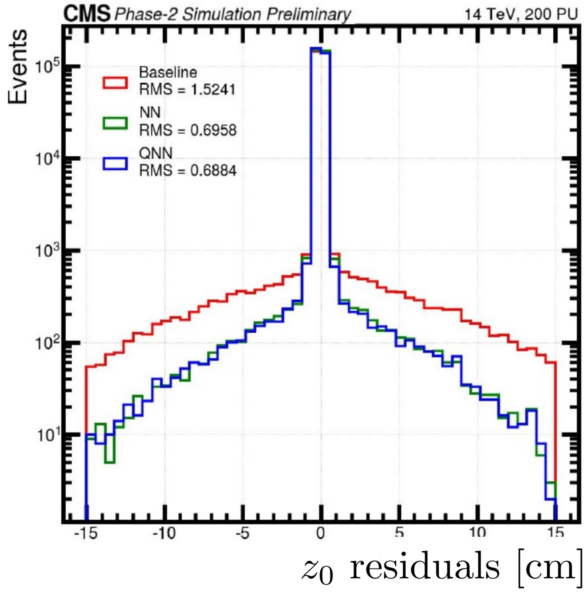
baseline



neural network



residuals

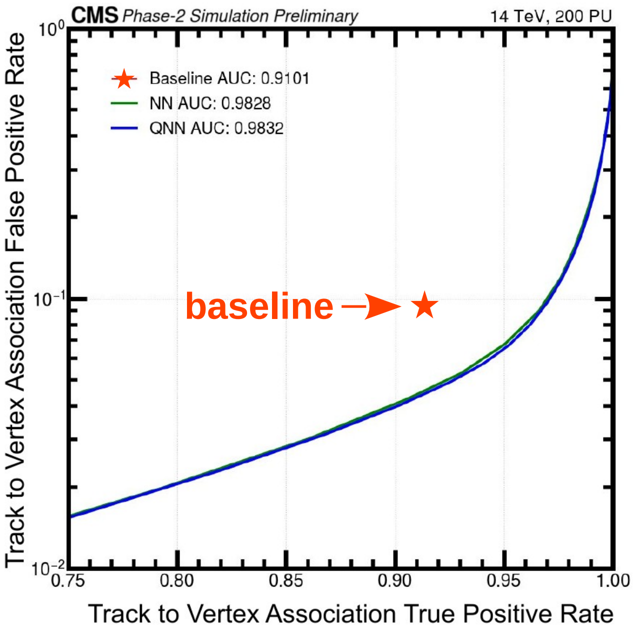


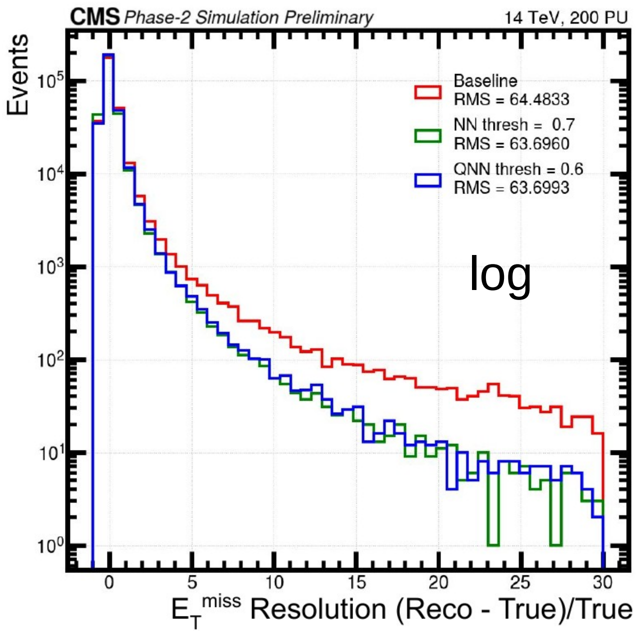
- similar resolution in $\Delta z_0 = z_0^{\mathrm{reco}} - z_0^{\mathrm{true}}$ reached in core of residuals
- used so-called "pseudo Huber" loss ($L = \sqrt{1+\delta^2} - 1$) → robust against outliers
- far fewer completely misreconstructed events with NN approach
- similar performance with quantized & pruned NN variant

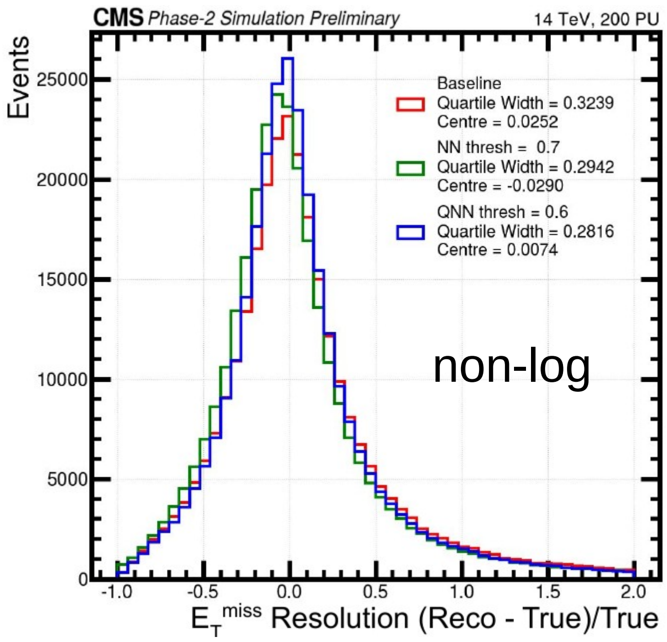
Performance: Track-vertex association

ROC curve

$E_{\mathrm{T}}^{\mathrm{miss}}$ resolution







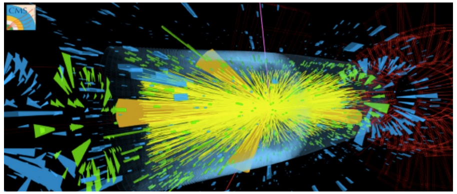
- track association to primary vertex = ultimate goal → define pileup-insensitive trigger objects
- hugely improved association with NN compared to baseline approach
- improvements seen in tails of tracker $E_{\mathrm{T}}^{\mathrm{miss}} = \left|\sum \vec{p}_i \times \delta_i^{\mathrm{NN}}\right|_{\mathrm{T}}$

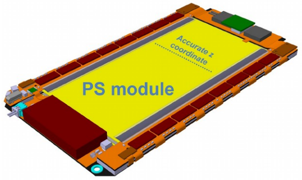
Summary

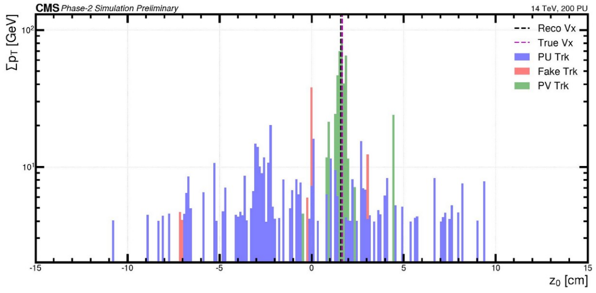
- primary vertex reconstruction at L1 for the CMS upgrade
  - CMS foresees tracking & vertexing at L1 to deal with increased pileup at HL LHC
  - crude vertexing algorithm in place

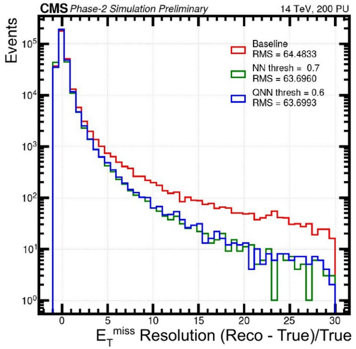
- novel neural network approach
  - optimally exploit (limited) information
  - optimized end-to-end
  - deployable on FPGAs

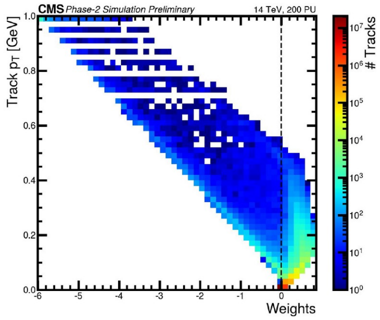
- further information: **CMS-DP-2021-035**

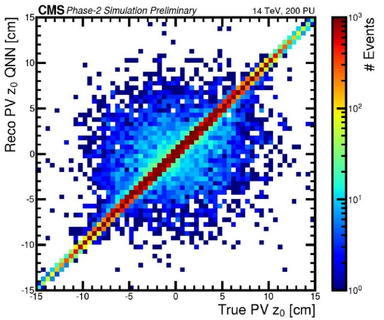












Backup

NN architecture