



Accelerating Efficient Transformer Architectures for Point Cloud Data in High Energy Physics



Fast Machine Learning Conference 2025
Zurich, Switzerland



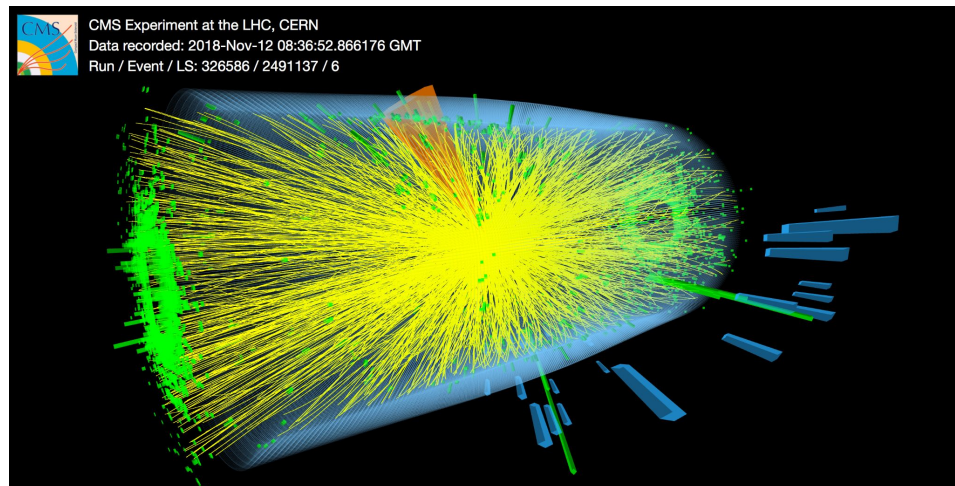
Chia-En Chang, Bo-Cheng Lai, Mia Liu, Jan-Frederik Schulte, Divij Agarwal, Jack P. Rodgers, Siqi Miao, Pan Li, Vladimir Loncar, Yuan-Tang Chou, Advait Anand, Amit Saha, Pan Li, Shih-Chieh Hsu, Shitij Govil, Javier Duarte, Kilian Lieret, Gage DeZoort



NATIONAL
YANG MING CHIAO TUNG
UNIVERSITY

Point cloud data in High Energy Physics Experiments

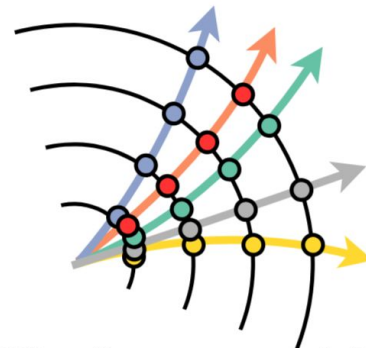
- HEP experiments like ATLAS and CMS record particle signals using $O(M)$ of channels
- Charged particles leave hits on multiple detector layers → **point cloud**
- $O(\%)$ occupancy
→ **sparse, irregular** datasets
- Hits from different particles independent from each other → Need to consider only **local** hits for reconstruction task
- Upgrade to **HL-LHC** will **significantly increase** the number of charged particles beyond what can be handled by traditional rule-based algorithms for track reconstruction



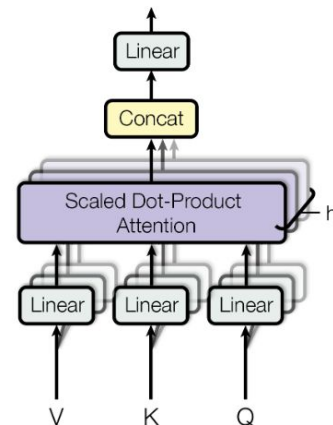
- **ML algorithms** based on graph neural networks (**GNNs**) or **transformers** are promising avenues to solve this computational challenge

Transformers for track reconstruction

- **GNNs** have long been considered for charged particle reconstruction
 - Downside: **Expensive graph construction**
- Transformers architectures have shown great potential for many scientific applications, including in High Energy Physics
 - Examples include [particle jet tagging](#), [particle track reconstruction](#)
- **Classic MHA** computes attention between all elements of input data
 - **Quadratic complexity** with rising point cloud size
 - **Wasteful** when computing attention for pairs of points with large distance between them
- Multiple **efficient point cloud transformers** with a **local attention** mechanism have been proposed
 - Most **struggle** with the **irregular data and complex geometry** in HEP
- Proposed solution: **Locality-Sensitive Hashing-Based Efficient Point Transformer (HEPT)** by S. Miao et al, [2402.12535](#)
- Has been **demonstrated for particle tracking and PU rejection** tasks
- Presented today
 - Further steps towards a **full track reconstruction pipeline** based on HEPT
 - An **FPGA implementation** based on hls4ml to accelerate HEPT for **real-time applications**



● Hits from a particle
↗ Track to reconstruct



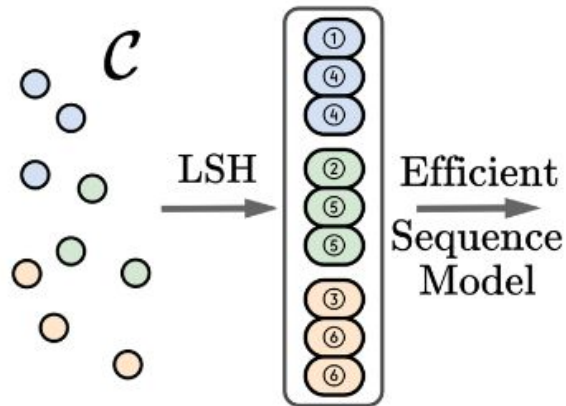
HEPT in a nutshell

- Central idea: **reduce computations** in the attention kernel
- **Locality sensitive hashing** using E2LSH is used to sort points into buckets
- Attention only computed for elements in same bucket

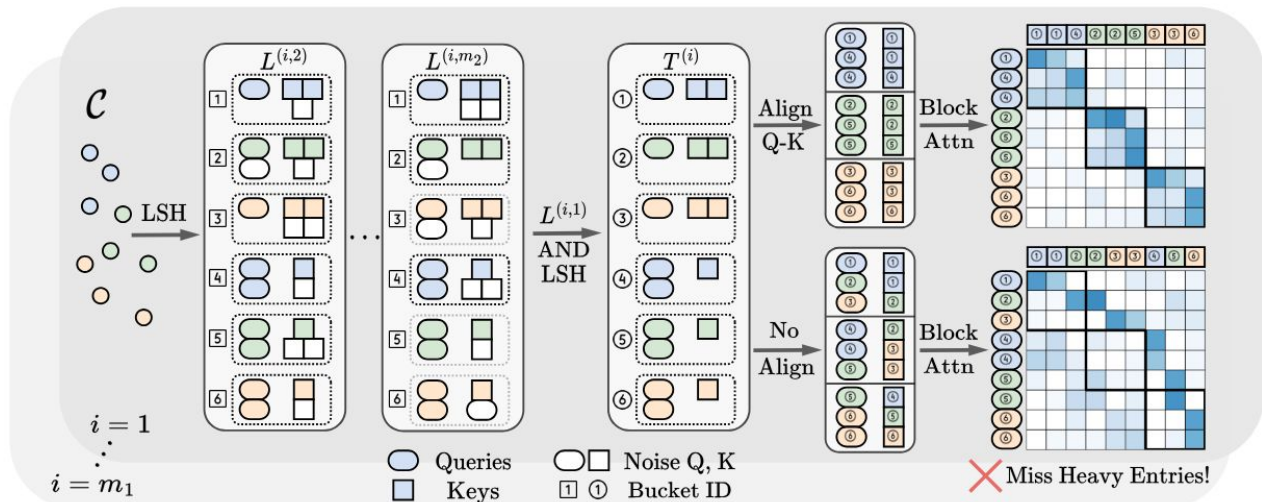
$$\mathbf{x} \in \mathbb{R}^d, h_{\mathbf{a},b}(\mathbf{x}) = \lfloor \frac{\mathbf{a} \cdot \mathbf{x} + b}{r} \rfloor, \mathbf{a} \sim \mathcal{N}(0, \mathbf{I}), b \sim \mathcal{U}(0, r)$$

- Attention **kernel** choice to exhibit **local inductive bias**:

$$k(\mathbf{q}_u, \mathbf{k}_u) = \exp\left(-\frac{1}{2}\|\mathbf{q}_u - \mathbf{k}_u\|^2\right)$$



- Achieves a **local attention mechanism with linear complexity**
- **Lower inference latency** compared to GNNs by $> 10x$
- Best resource scaling out of all proposed point cloud transformer models

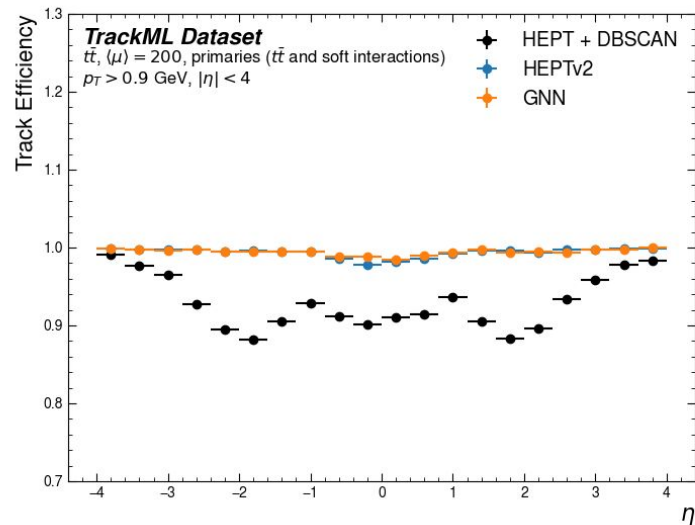
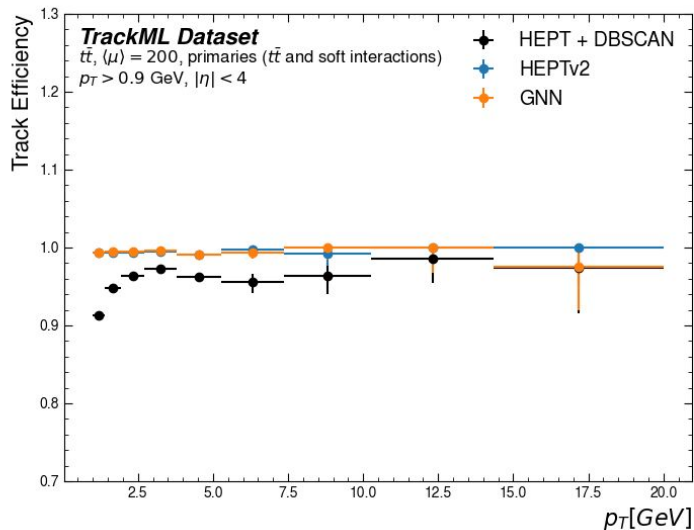


Towards a realistic HEPT application in HEP

- The original HEPT model focused on predicting the **track embeddings**
 - **Additional clustering** step with e.g. DBSCAN needed for full track reconstruction
 - No direct physics performance comparison with GNN models
- Presenting **HEPTv2**
 - Adds a point-wise classifier, separates hits from tracks from background noise
 - Query-based instance **decoder assigns hits to individual tracks**
- Performance evaluated on the TrackML dataset
 - Simulates HL-LHC conditions for particle interactions and a reference detector
 - Focus on short tracks just from pixel detector hits
 - **Compare to Exa.TrkX** as a representative GNN-based tracking pipeline

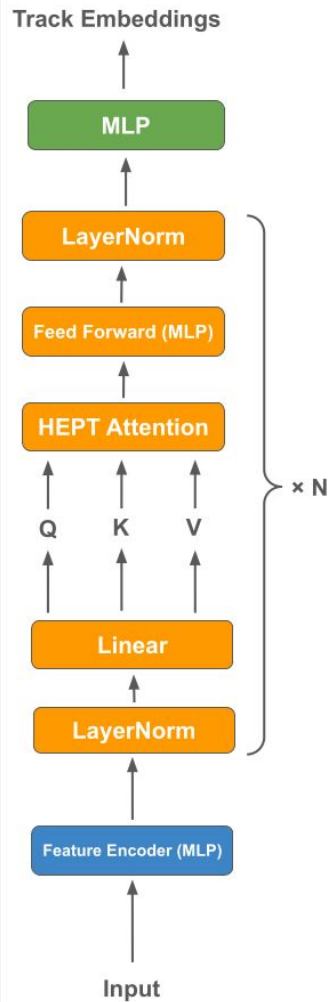
HEPTvs Performance

- **HEPTv2 matches** tracking efficiency of **Exa.TrkX**
 - Higher track fake rate to be improved
- Significantly **beats** efficiency of **HEPT + additional clustering**
 - Also **faster by a factor ~50**



Accelerating HEPT on FPGA

- **Real-time applications** place stringent requirements on ML algorithms
 - **Inference time $O(\mu\text{s})$** requires acceleration with FPGAs
 - Computing resources limited by FPGA boards
- Need to **implement** HEPT hashing algorithms, sorting, and attention kernel in **HLS code**
- Aim to use hls4ml framework for integration into full model pipeline
- Consider two benchmark applications (very similar models):
 - **Note: No model compression applied yet**
 - Identification of tau \rightarrow 3mu decays from muon system signals.
 - $\langle N_{\text{points}} \rangle \sim 30$
 - Particle track embedding model from HEPT paper.
 - TrackML dataset has $\langle N_{\text{points}} \rangle / \text{evt} \sim 60\text{k}$
 - Subdivide into 100 geometric region to get to $\langle N_{\text{points}} \rangle / \text{evt} 600$.



FPGA implementation: Technical overview

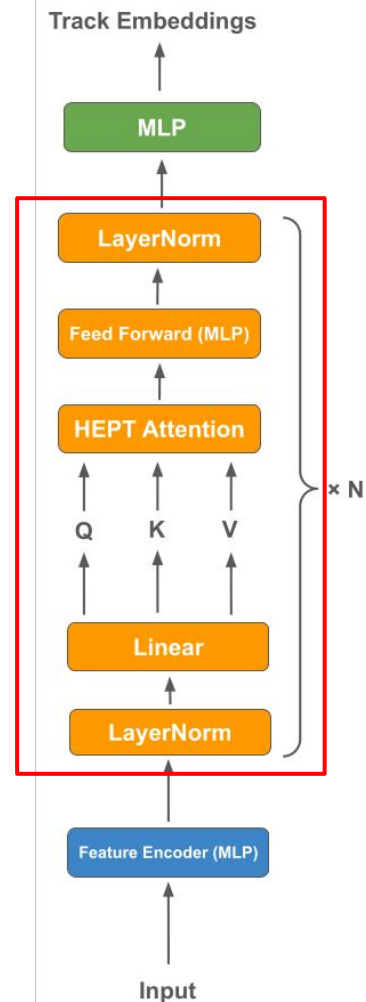
- LSH hashing and bucketing + attention computation implemented using Vitis HLS
 - Parameter settings: number of heads = 2, dimension per head = 16, num_hashes = 5, block size = 5
- Target board:
 - AMD Alveo™ U250 Data Center Accelerator Card (xcu250-figd2104-2L-e)
 - 200MHz clock rate

- C-Synthesis resource estimate for the tau → 3mu model

BRAM	DSP	FF	LUT	URAM
4 (~0%)	7230 (58%)	905017 (26%)	1459777 (84%)	-

- Co-Simulation latency: $212 * 5\text{ns} = 1.06\mu\text{s}$

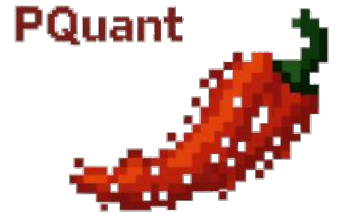
Avg Latency	Max Latency	Min Latency
212	212	212



FPGA implementation: Challenges

- Even for the smaller model, **resource usage** is relatively **high**
 - Exploring further optimization and model compression
- When scaling up from $\langle N_{\text{points}} \rangle \sim 30$ to $\langle N_{\text{points}} \rangle \sim 600$, resource or latency are **scaling with $O(n \log(n))$**
- **Challenges** for optimization:
 - To achieve a high throughput rate, all steps must have same throughput rate
 - **Hash sorting** layer: current bubble sort needs to be replaced with more **hardware-friendly approach**
 - **Core attention** layer: resource usage is **proportional to block size squared**

Model compression: Pruning + Quantization



- Reduce FPGA resource usage using model compression:
 - **Pruning:** Remove low-weight nodes in linear layers
 - **Quantization:** Reduce bit-width of variables. Best done during training (**QAT**)
- For pruning and quantization of PyTorch model use [PQuant](#).
 - New tool for simultaneous pruning and quantization being developed at CERN
 - See [yesterday's talk](#) by Roope Niemi
 - hls4ml support for models being added by PQuant team, pull request open

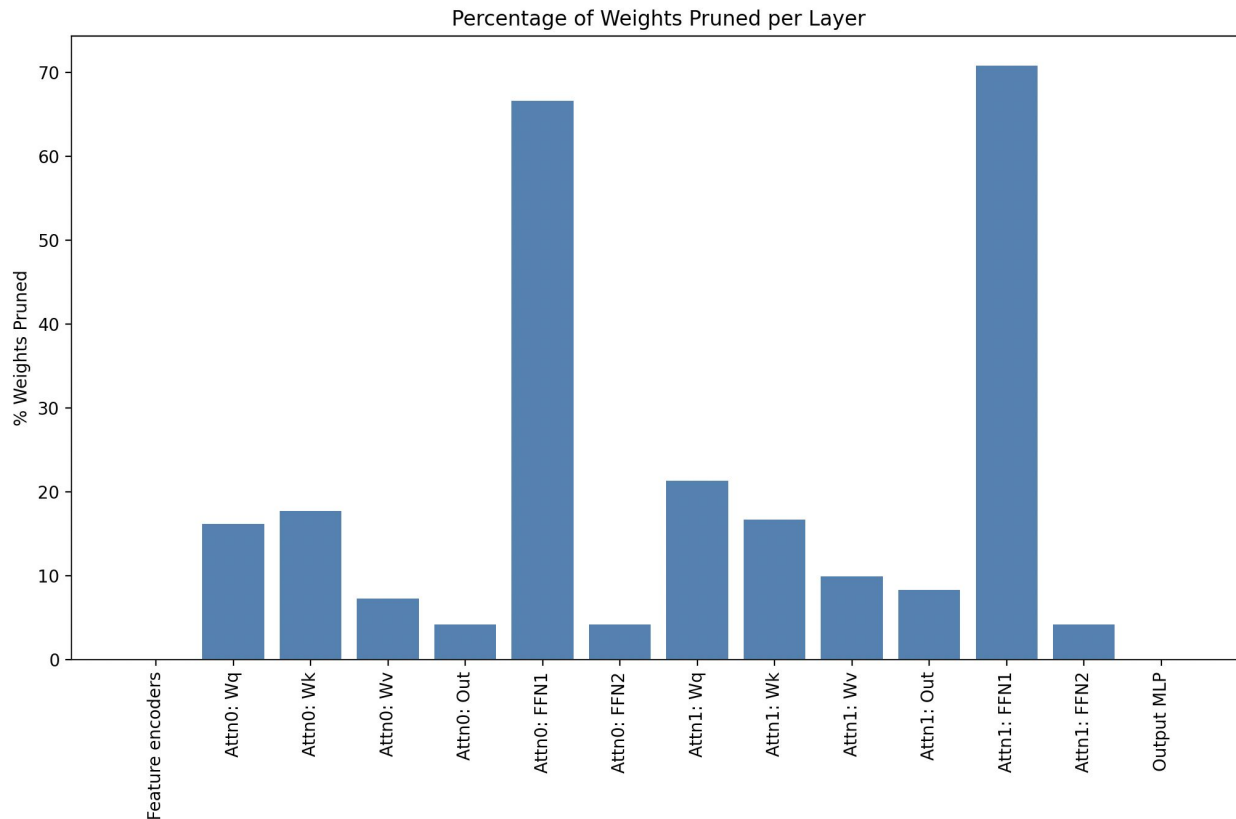
Pruning the HEPT model

- **Method:** Activation pruning
- **Scope:** Only attention layers
- **Sparsity:** ~11% of weights pruned

HEPT Accuracy on Tracking-600 dataset (before and after pruning)

Before	After
89.05%	86.18%

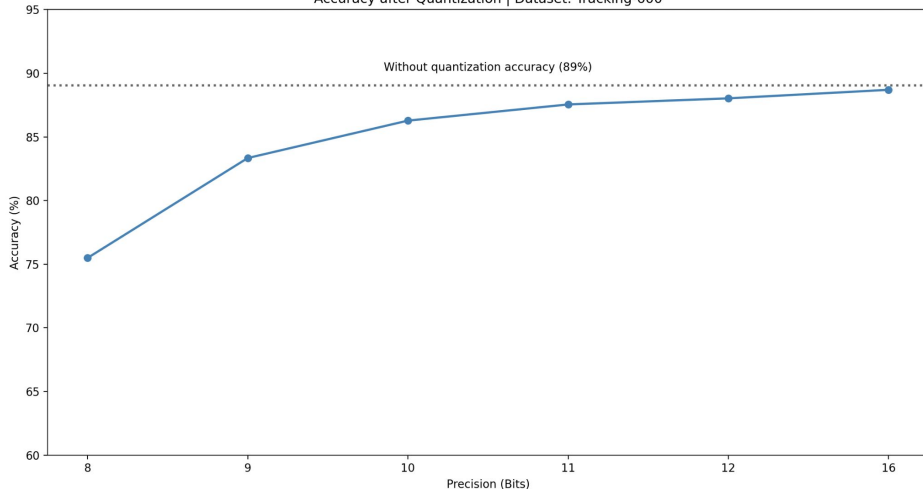
- Next steps: Consider **different pruning algorithms**, test more aggressive pruning settings



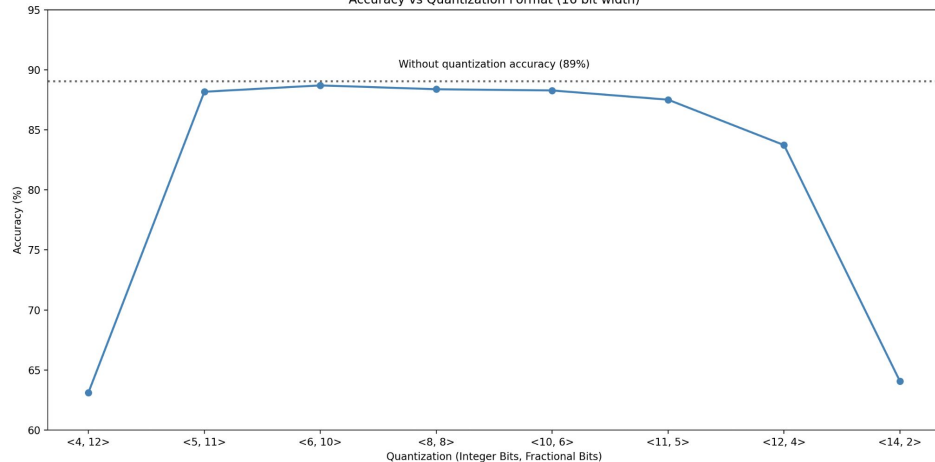
Quantizing the HEPT model

- Exploring **fixed-point quantization** with PQuant
- **Layers excluded:**
 - Normalization layers
 - MLP output layers
- First iteration: Use symmetric quantization with **same precision everywhere**
- Model similarly sensitive to number of integer and fractional bits

Accuracy after Quantization | Dataset: Tracking-600



Accuracy vs Quantization Format (16 bit width)



- **Quantizing from 32 to 16 bit has negligible effect on accuracy**
 - **Moderate loss of accuracy (~2%) down to 10 bits**
 - **Large loss (~25%) at 8 bit**
-
- Next: **More granular quantization**
 - Study impact of compression on **FPGA performance**

Summary

- The HEPT transformer architecture has been shown to be highly promising for point cloud data in HEP, especially charged particle track reconstruction
- Has newly been extended in HEPTv2 to handle full track reconstruction
 - Physics performance comparable to representative GNN models
- Deployment in real-time applications requires FPGA implementation
 - Prototype implementation complete
 - Latency of $O(\mu\text{s})$ has been achieved
 - Resource usage and scaling remains a challenge and focus on future work
 - Model compression is being explored and shows promise to reduce resource requirements