
Application of Structured State Space Models to High energy physics with locality-sensitive hashing

Cheng Jiang
University of Edinburgh

Sitian Qian
Peking University

Abstract

Modern high-energy physics (HEP) experiments are increasingly challenged by the vast size and complexity of their datasets, particularly regarding large-scale point cloud processing and long sequences. In this study, to address these challenges, we explore the application of structured state space models (SSMs), proposing one of the first trials to integrate local-sensitive hashing into either a hybrid or pure Mamba Model. Our results demonstrate that pure SSMs could serve as powerful backbones for HEP problems involving tasks for long sequence data with local inductive bias. By integrating locality-sensitive hashing into Mamba blocks, we achieve significant improvements over traditional backbones in key HEP tasks, surpassing them in inference speed and physics metrics while reducing computational overhead. In key tests, our approach demonstrated promising results, presenting a viable alternative to traditional transformer backbones by significantly reducing FLOPS while maintaining robust performance.

1 INTRODUCTIONS

Large-scale point clouds and long-sequence data processing are becoming prevalent in many scientific domains. In one of the fundamental subjects, high-energy physics (HEP), exploring the frontier of new physics usually requires a vast amount of data to support and validate the excess of theory prediction statistically (Aad et al., 2012; Chatrchyan et al., 2012). As the need for more data grows, the CERN Large Hadron

Collider (LHC) (Aberle et al., 2020), for example, must operate in increasingly intense experimental environments. Higher luminosity and collision rates are essential to probe deeper into the fundamental structure of matter, but this also leads to more complex data landscapes and more congested signatures of events, resulting in higher pile-up effects where multiple interactions occur within the same detector readout window. It is expected that a higher number of particles is directly related to increased complexity in tracking their trajectories and depositing energy across the detector in a finer granularity (Tumasyan et al., 2017).

The sequential nature of interactions over time, as particles traverse the detector, results in long sequence data. Each particle’s trajectory can be represented as a series of hits over time, and reconstructing these paths requires processing these long sequences of interactions. Traditional methods struggle with capturing long-range dependencies and missing intricate relationships between hits, which leads to information loss and reduced accuracy in the reconstruction, especially in an intensive environment. Similarly, these particle interactions can also be represented within a geometric space (Qu and Gouskos, 2020; Mikuni and Canelli, 2021; Qu et al., 2024; Kansal et al., 2022) and leverage geometric deep learning (GDL) (Bronstein et al., 2017, 2021) to model and analyze them effectively. Our study will primarily focus on the key challenges of tracking in High Luminosity-LHC (Aberle et al., 2020), while also addressing the complexities of pileup mitigation. Both of these tasks involve processing data with very long, variable-length sequences. Furthermore, both tasks could benefit from introducing deep learning models with local inductive bias, which would help effectively associate nearby hits and energy deposits with the correct particle or interaction.

Previous studies have aimed to preserve the locality of point clouds by aggregating information from nearby regions, ensuring that local structures and interactions are accurately captured. A few recent approaches have utilized graph neural networks (GNNs) to dynamically construct graphs (Ju et al., 2020; Liu et al., 2023a;

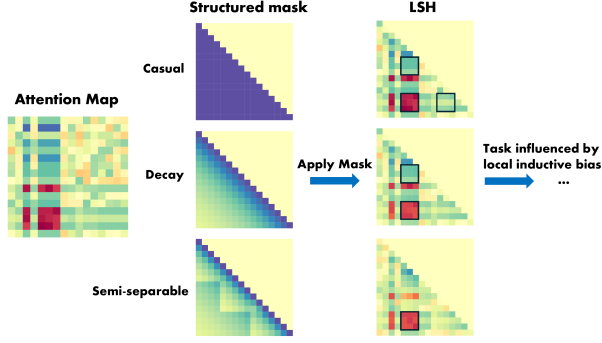


Figure 1: An illustration of different structured mask types applied to attention maps, followed by the integration of Local-Sensitive Hashing (LSH). The causal mask corresponds to linear attention transformers, the decay mask to the retentive network, and the semi-separable mask represents structured SSM. In this context, LSH operates by aligning similar query-key values, allowing the semi-separable mask to effectively select the relevant blocks from neighboring regions while minimizing interference from unrelated areas.

Bertacchi, 2020; Calafiura et al., 2024), or have introduced specific objective functions like object condensation loss (Lieret et al., 2023) to enhance the modeling of point clouds. However, those types of GNN applications have the intrinsic downside where traditional ways like k-NN or radius graph construction would introduce $\mathcal{O}(n^2)$ to make both training and inference very slow, especially when n is larger. Transformers, on the other hand, are well-suited for capturing long-range dependencies and have demonstrated strong performance in various tasks. However, they also suffer from quadratic complexity due to the matrix manipulation required for full multi-head attention, which can become computationally expensive for large-scale point clouds or long sequences. Later trials aimed at reducing the complexity of the attention mechanism have explored approaches such as low-rank approximation (Chen et al., 2021), retentive attention (Sun et al., 2023), and etc (Choromanski et al., 2022; Kitaev et al., 2020; Wang et al., 2020; Daras et al., 2020). Some studies (Miao et al., 2024a) have shown the random Fourier features (RFF) (Rahimi and Recht, 2007) and locality-sensitive hashing (LSH) (Indyk and Motwani, 1998) are so far few of the best efficient transformers to be used for tasks with local inductive bias. Despite these efforts, the computational overhead remains unsatisfactory, as the complexity still scales with $\mathcal{O}(n \text{poly}(n))$ (Miao et al., 2024a), which significantly hampers both training and inference in large-scale scenarios. None of the current methods have explored the usage of the State Space Model (SSM) (Gu et al., 2020, 2021, 2022b,c), which offers near-

linear complexity. SSMs are particularly well-suited for longer sequences and could comprehensively anatomize the real advantages of intensive tracking hits.

Originally derived from the classic Kalman filter model, SSMs have proven highly efficient in processing long sequences due to their designated convolutional computation and low computational complexity. The early development of SSM did not give stronger metrics than traditional transformers in language modeling. Until very recently, models like Mamba (Gu et al., 2022a; Fu et al., 2023; Gu and Dao, 2024) and Griffin (De et al., 2024) have shown comparable, if not superior, performance in a variety of modeling tasks involving long tokens. The first version of Mamba (Gu and Dao, 2024) incorporates time-varying parameters into a recurrent dynamical system and utilizes a hardware-aware algorithm that enables highly efficient training and inference. The second version of Mamba (Dao and Gu, 2024) introduces State Space Duality, which establishes a connection between structured SSMs and variants of attention by interpreting SSMs as semi-separable matrix transformations. This approach eventually generalizes the way SSMs operate as structured masked linear attention. Some novel hybrid Transformer-Mamba architectures (Zhu et al., 2024; Lieber et al., 2024; Glorioso et al., 2024) were presented after this and showed even better performance than both pure Mamba and Transformer models.

The study is inspired to explore what might happen if one merges the efficient attention mechanism into Mamba, creating a hybrid model that could combine the strengths of both approaches. Additionally, we considered incorporating LSH into Mamba to determine whether this could further enhance performance in tasks that benefit from local inductive bias, such as tracking or other applications involving structured data. These modifications aim to balance computational efficiency with the ability to capture both local and long-range dependencies.

In this paper, we have conducted extensive validation on the proposed architectures, evaluating datasets ranging from 3k to over 60k tracking hits. Compared to the previous HEPT model, which already achieved state-of-the-art (SOTA) accuracy with more than 200 \times speedup, our approach further reduces FLOPs by over 10 \times times while maintaining comparable accuracy and significantly improving SOTA recall. Alternatively, we achieve SOTA accuracy with only a fraction of the model size. Additionally, in the pileup mitigation task, all proposed models outperform the previous studies. By offering both practical applicability and improved performance, those models could be much more realistic to apply in real HEP experi-

ments.

2 BACKGROUND

2.1 State Space Models

Starting with the classical state space models that are continuous-time in nature, the new structured state-space sequence models try to use the knowledge from Recurrent Neural Network (RNN) (Schuster and Paliwal, 1997; Sherstinsky, 2020) and Convolutional Neural Network (CNN) (Lecun et al., 1998; Szegedy et al., 2014) to make the sequence model hold the ability to capture long-range dependencies at very low computational costs. The general form of structured SSMs can be written as:

$$\begin{aligned} h'(t) &= Ah(t) + Bx(t) & y(t) &= Ch(t) \quad (1a) \\ h_t &= \bar{A}h_{t-1} + \bar{B}x_t & y_t &= Ch_t \quad (1b) \end{aligned}$$

The model is defined via four parameters (Δ, A, B, C) where the step size Δ controls how much focus is on the current state x_t . The first stage uses the discretization rule to transform “continuous parameters” (Δ, A, B) to “discrete parameters” (\bar{A}, \bar{B}) through fixed formulas $\bar{A} = f_A(\Delta, A)$ and $\bar{B} = f_B(\Delta, A, B)$. After the transformation, the model (Gu et al., 2022a) can be computed by linear recurrence or global convolution.

Based on these concepts, Mamba introduces a refined mechanism to boost performance. The architecture begins by incorporating a standard local convolution before applying the traditional SSM steps, establishing the foundation of the Mamba architecture. This initial convolution allows the model to capture localized patterns and dependencies. Next, Mamba adopts a gated MLP structure, where the usual multiplicative gating mechanism is replaced with an activation function. Moreover, Mamba uses its selective copying mechanism, which addresses one of the key challenges faced by traditional transformer architectures. Unlike transformers (Vaswani et al., 2023), which rely on a Key-Value (KV) cache to store the entire sequence of content, Mamba can selectively choose which parts of the input to retain or ignore. The 1st version of Mamba already showed better performance and exceptionally faster inference than the common transformer backbone in many language modeling tasks.

The 2nd version of Mamba generalizes the structured state space model as the special form of attention family. Specifically, the linear attention uses the associativity of matrix multiplication in attention calculation (Katharopoulos et al., 2020) ($QK^\top \cdot V = Q \cdot (K^\top V)$), and drops the softmax. If thinking of the casual (autoregressive) attention mask as a lower triangle matrix, then those efficient transformers can have a relatively

general attention form $(L \circ QK^\top) \cdot V$ where mask matrix L determines how the final attention map looks like, as seen in Fig. 1. By induction, we could write the hidden state h_t as:

$$\begin{aligned} h_t &= A_t \cdots A_1 B_0 x_0 + A_t \cdots A_2 B_1 x_1 + \cdots + B_t x_t \\ &= \sum_{s=0}^t A_{t:s}^\times B_s x_s. \quad (2) \end{aligned}$$

then y_t in Eqn.1b is rewritten as:

$$\begin{aligned} y_t &= \sum_{s=0}^t C_t^\top A_{t:s}^\times B_s x_s \quad (3) \\ y &= SSM(A, B, C)(x) = Mx \quad (4) \end{aligned}$$

The left matrix $M_{ji} := C_j^\top A_j \cdots A_{i+1} B_i$ is a semi-separable lower triangle matrix where every sub-rectangle block contained below the diagonal has rank up to SSM’s state dimension N . This generalization of a structured attention mask brings SSMs closer to attention mechanisms, making Mamba adaptable for both pure or hybrid models (Lieber et al., 2024; Hwang et al., 2024).

2.2 Local Sensitive Hashing

Locality-sensitive hashing (LSH) was initially introduced for approximate nearest neighbor search. It works by ensuring that, with high probability, data points that are close together are hashed into the same bucket, while points that are far apart are placed into different buckets. For E2LSH (Euclidean LSH) (Datar et al., 2004), which is designed for Euclidean distance, the hash function is defined as $H_{a,b}(x) = \lfloor \frac{a \cdot x + b}{r} \rfloor$ where a is the unit normal distribution, b is the random number uniformly distributed from 0 to r , $r > 0$ that controls bucket sizes. Two main logical LSH were introduced: OR&AND LSH. In OR LSH, multiple hash tables are constructed, each using a different set of hash functions g_1, g_2, \dots, g_{m_1} . Two points are neighbors if they match in at least one of the hash tables, i.e., $g_i(x) = g_i(y)$ for some i . In AND LSH, multiple hash functions H_1, H_2, \dots, H_{m_2} to form a new hash family $g(x) = [H_1(x), H_2(x), \dots, H_{m_2}(x)]$. Two points x and y are considered neighbors if all hash functions agree, i.e., $g(x) = g(y)$.

In most LSH-based efficient transformers like Reformer (Kitaev et al., 2020) and Smyrf (Daras et al., 2020), they adopt only the OR LSH and ignore the AND LSH. Since OR LSH only requires agreement on one of the hash tables rather than all possible hash functions, it can lead to a larger approximation error as the number of points increases. This relaxation reduces the precision of the approximation, making it more likely to group dissimilar points, especially in larger datasets. The recent work LSH-based Efficient Point Transformer (HEPT) tried to combine two hash tables and showed a good performance on tasks with

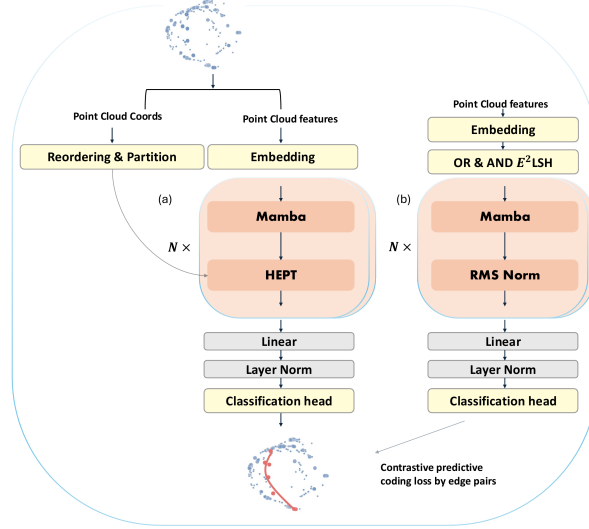


Figure 2: A schematic diagram of two proposed architectures for tasks with local inductive bias is presented. Left: The Mamba-a architecture, inspired by the hybrid Transformer-Mamba model (i.e., Jamba), excludes the MoE layer to reduce training memory requirements. Right: The Mamba-b architecture integrates the OR&AND E2LSH selection and bucketing mechanism into pure Mamba blocks. The final loss for both models is computed from the embedding output using the Info Noise-Contrastive Estimation (InfoNCE) loss (van den Oord et al., 2019), constructed by predefined kNN edge pairs.

local inductive bias. Similar to the query-key alignment in transformers, if hidden states h_u and h_v have small $\|h_u - h_v\|_2$, they are likely to have close hash values. We follow the same processing steps as HEPT, by further partitioning the selectively hidden space into various distinct, non-overlapping regions randomly to avoid large misalignments in the same bucket. More details can be found in (Miao et al., 2024a).

2.3 Long Sequence Tasks in HEP

Tracking In HEP, tracking refers to the process of reconstructing the trajectories of charged particles produced in particle collision. Tracking is essential for understanding particle interactions, as it allows physicists to reconstruct the paths of particles based on measurements (called "hits") from detectors. These hits are associated with the positions where particles pass through the detector layers, and the tracking algorithm's goal is to associate them with the correct particle tracks, ideally with high accuracy and efficiency. **Traditional tracking algorithms at the LHC, surprisingly similar as SSM, are based on Kalman filters (Billoir, 1984; Frühwirth, 1987).** These algorithms work by iteratively extrapolating a particle's trajectory based on a small set of initial hits (a "track seed") and searching for subsequent hits that match the predicted path. The computational cost of conventional algorithms increases rapidly as the number of hits and pileup events grows, which is particu-

larly problematic for future HL-LHC, where tracking large numbers of particles in high-pileup environments becomes a computational bottleneck.

Pileup Mitigation Pileup refers to multiple particle collisions occurring simultaneously. Reconstructed objects falsely attributed to the primary collision are considered pileup contamination, which can inevitably degrade the accuracy and sensitivity of event reconstruction and analysis. Pileup mitigation techniques are designed to identify and remove particles resulting from pileup interactions, ensuring that only particles from the primary collision are included in the analysis. One of the widely-used algorithms for pileup mitigation is charged-hadron subtraction, as implemented in the CMS Particle Flow (PF) algorithm (Sirunyan et al., 2017). This method rejects charged particles whose tracks are not associated with the primary collision vertex, thus reducing pileup contamination. In more advanced approaches, algorithms such as PUPPI (Bertolini et al., 2014) (PileUp Per Particle Identification) and SoftKiller (Cacciari et al., 2015) aim to use likelihood-based method to suppress pileup contributions among neutral particles as well.

3 RELATED WORKS

LSH-based Efficient Point Transformer As the paper mentioned earlier, the HEPT architecture lever-

ages both OR LSH and AND LSH to approximate the attention mechanism with near-linear computational complexity. The model starts by evaluating the error-computation tradeoff to argue the LSH-based efficient transformers have smaller approximation errors than RFF-based ones. To address potential misalignment issues between query and key buckets, HEPT integrates point cloud coordinates as additional AND hash codes. This ensures that only queries and keys that are close in both feature space and geometric space are grouped, which further reduces the approximation error. The performance validation of the model gives SOTA accuracy on long sequence tracking while still maintaining the same scale of inference FLOPs as other common efficient transformers.

Hybrid SSM-attention model A few hybrid SSM-attention models, such as Zamba, use shared attention and MLP block to connect multiple Mamba blocks with shared parameters. This setup leverages the FLOPs efficiency of Mamba blocks while delivering performance comparable to other language models. Another approach to constructing hybrid models involves combining Transformer, Mamba, and MoE across different layers. However, in our empirical tests on those two tasks, the inclusion of MoE did not consistently yield a clear performance boost, while significantly increasing RAM consumption. So, we decided to drop the MoE setting when introducing the hybrid model Mamba-a, as shown in Fig. 2.

Receptance Weighted Key Value Receptance Weighted Key Value (RWKV) (Peng et al., 2023) is a novel model architecture designed to combine the efficient inference of RNNs with the parallelizable training of transformers. One key addition to the QKV mechanism in transformers is the Receptance vector, which controls how much information from the previous time step should be carried forward, similar to the gating mechanism in RNNs. It also has a weighted decay vector which modifies the interaction weights between time steps. The basic RWKV architecture consists of two main components: time-mixing and channel-mixing. The time-mixing part works similarly to the attention mechanism, while the channel-mixing part resembles the feed-forward layer. In RWKV5, the model incorporates the concept of multi-head computation. RWKV6 further improves the architecture by integrating Low-Rank Adaptation (LoRA) (Hu et al., 2021) into the weight decay factor, making the model more efficient in terms of fine-tuning. In this study, we will use RWKV as the validation set for our proposed Mamba architecture, as it also claims to have near-linear computational complexity. Yet we believe there is significant potential for further exploration of

RWKV in future research.

4 TRAINING METHOD

4.1 Datasets

The tracking dataset is used in the TrackML challenge (Amrouche et al., 2019, 2021), simulating detector conditions under the HL-LHC condition. Each event contains simulated 3D measurements (hits) of particles generated during proton collisions at LHC. The objective is to group the recorded hits into particle tracks, where each hit should be associated with a single particle track. Each hit is given by the unique hit ID, 3D coordinates, and geometry-related identifiers. We consider only particles with transverse momentum threshold $p_T > 0.9$ GeV as the low p_T event usually would come from background noise. One complete event has more than 60k hits, we can continue splitting the events by different numbers of sectors.¹

For the pileup mitigation tasks, we used a publicly available dataset (Miao et al., 2024b) simulated under the same HL-LHC pileup conditions. Each event in the dataset has particle-level information such as four-momentum, charge, and other relevant properties. The dataset consists of 1,000 point clouds, each containing approximately 10,000 points.

4.2 Training Setup

For the tracking dataset, we convert the 3D Euclidean coordinates into cylindrical coordinates and utilize both local and global coordinates of the sub-detector. For more details, refer to Table.6 in (Miao et al., 2024a). All low-level inputs are embedded into hidden dimensions before being processed by any blocks. The Mamba-a architecture, as shown on the left of Fig. 2, applies the same HEPT attention layer after every Mamba layer. The same reordering and partitioning strategy is employed for LSH. After passing through the main blocks, the data is fed into a fully connected layer without bias and undergoes layer normalization. Finally, a classification MLP is applied to produce the output in the desired embedding dimension. The Mamba-b architecture, as shown on the right of the figure, only applies the OR&AND LSH once before being processed by the main blocks. Essentially, the Mamba-b architecture should have roughly the same model complexity as the pure Mamba model, with the only addition being the LSH processing applied before the main block to minimize the overhead.

¹Note the different number of sectors would not equally split the hits in complicated geometry, in the final performance plots Fig. 7, 8, we choose n_{sector} to be 1,2,3,6,10,20 to test the performance under different lengths.

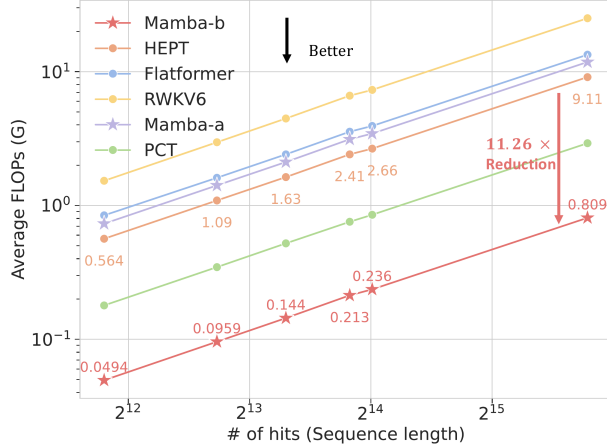


Figure 3: Performance plots for average inference FLOPs of various small-sized models across different numbers of hits, ranging from 3k to 60k. (All tests were performed on the actual dataset and evaluated on a single NVIDIA A100, ensuring realistic performance evaluation rather than relying on toy points.)

For the pileup dataset, we have a total of 7 particle-level features, including information such as location, momentum, and energy. The Particle ID, which indicates the type of particle, is embedded into an 8-16 dimensional space before being combined with the positional embedding. Later training setup follows the same approach as the tracking dataset, but the model outputs predictions on a per-particle basis.

5 EXPERIMENTAL RESULTS

The evaluation metrics for tracking consider two key aspects. First, inference speed is crucial, as fast machine learning inference becomes increasingly important as the number of data points grows larger in real-experiment scenarios. We measured the average inference FLOPs and throughput for each tracking event across the entire dataset, with hit counts ranging from 3k to 60k. To fully understand the actual computation effectiveness, it is also essential to consider the raw inference time, as FLOPs alone do not fully stand for the actual inference speed. From the performance plot Fig. 7, 8, we used a single NVIDIA A100 GPU to calculate the FLOPs during the forward pass. Combining the results from both figures, the Mamba-a-S model demonstrates FLOPs and throughput on par with other RFF-based models like FlatFormer (Liu et al., 2023b) and LSH-based transformers like HEPT. Although PCT has significantly fewer FLOPs compared to other models (except for Mamba-b-S), it exhibits worse scalability and higher computational complexity as the number of hits increases. Conversely, RWKV

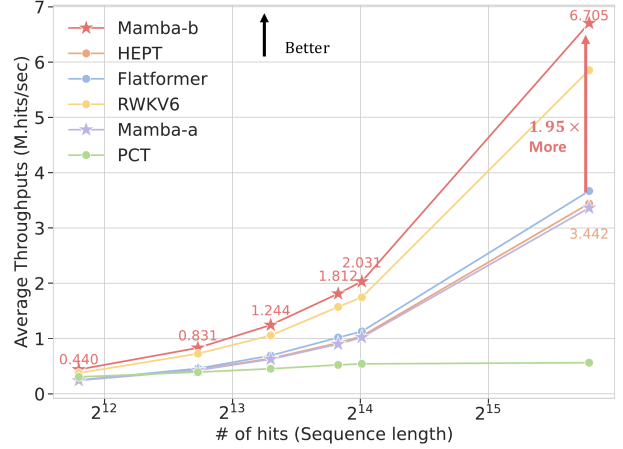


Figure 4: Performance plots for average throughputs (millions of hits) of various small-sized models across different numbers of hits, ranging from 3k to 60k. (while the actual FLOPs do not represent fully the raw inference time, can see clearly from the PCT and Flatformer case.)

shows the highest FLOPs but also the second-highest throughput. Most notably, Mamba-b-S achieves both the best performance in terms of FLOPs, with an 11.26 \times reduction compared to the previous SOTA model HEPT, and the highest throughput, boasting 1.96 \times more than the usual efficient transformers, all while maintaining a similar model size.

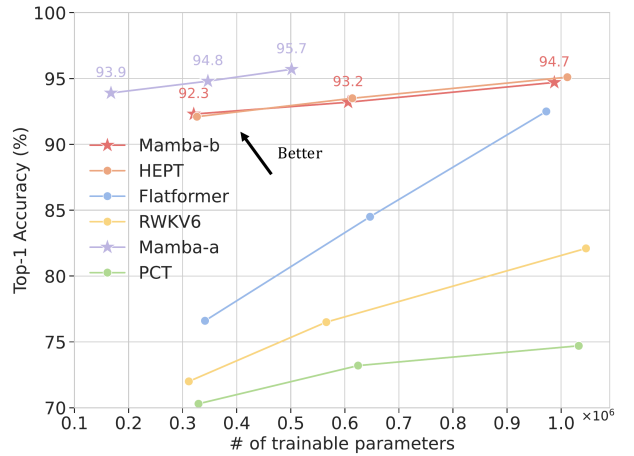


Figure 5: Top-1 accuracy of different-sized models on Tracking-60k scenario. Mamba-a demonstrates better accuracy than all other larger models, while Mamba-b achieves comparable accuracy as the previous SOTA model.

Second, the accuracy and recall of the model are equally important. In addition to general performance metrics, we use a specialized metric to assess the qual-

Table 1: Top1-Accuracy table for different backbones with different numbers of parameters on Tracking-60k dataset. The *, †, ‡, are the first, second, and third best results. **Bold** is the highlighted result with great performance.

| Method | #param. | Tracking-60k top-1 acc. |
|---------------------|---------|----------------------------|
| RNN | | |
| RWKV5-S | 0.32 M | 72.4 |
| RWKV5-M | 0.57 M | 76.6 |
| RWKV6-S | 0.31 M | 72.0 |
| RWKV6-M | 0.57 M | 76.5 |
| RWKV6-L | 1.05 M | 82.1 |
| Transformers | | |
| HEPT-S | 0.33 M | 92.1 [‡] |
| HEPT-M | 0.61 M | 93.5 [†] |
| HEPT-L | 1.01 M | 95.1[†] |
| FlatFormer-S | 0.34 M | 76.6 |
| FlatFormer-M | 0.65 M | 82.5 |
| FlatFormer-L | 0.97 M | 91.0 |
| PCT-S | 0.33 M | 70.3 |
| PCT-M | 0.61 M | 73.2 |
| PCT-L | 1.03 M | 74.7 |
| SSMs | | |
| Mamba-S | 0.33 M | 79.4 |
| Mamba-M | 0.63 M | 80.9 |
| Mamba-L | 1.00 M | 84.4 |
| Mamba-a-S | 0.17 M | 93.9* |
| Mamba-a-M | 0.35 M | 94.8* |
| Mamba-a-L | 0.50 M | 95.7* |
| Mamba-b-S | 0.32 M | 92.3 [†] |
| Mamba-b-M | 0.61 M | 93.2 [‡] |
| Mamba-b-L | 0.99 M | 94.7[‡] |

Table 2: Best top backbones with small numbers of parameters on Tracking-6k (short), Tracking-15k (medium), Tracking-60k (long) dataset. **Bold** is the highlighted result with great performance.

| Method | top-1 acc. | top-1 recall |
|---------------------|-------------|--------------|
| Tracking-6k | | |
| HEPT-S | 92.2 | 95.6 |
| FlatFormer-S | 77.1 | 95.2 |
| Mamba-a-S | 94.0 | 96.5 |
| Mamba-b-S | 91.9 | 99.1 |
| Tracking-15k | | |
| HEPT-S | 92.2 | 95.7 |
| FlatFormer-S | 76.7 | 94.2 |
| Mamba-a-S | 94.3 | 96.9 |
| Mamba-b-S | 91.5 | 99.0 |
| Tracking-60k | | |
| HEPT-S | 92.1 | 95.7 |
| FlatFormer-S | 76.6 | 94.0 |
| Mamba-a-S | 93.9 | 96.7 |
| Mamba-b-S | 92.3 | 99.3 |

ity of the learned point embeddings, particularly how well the embeddings from the same particle cluster are grouped. acc. is defined as $\frac{1}{n} \sum_{u=1}^n \text{Prec}@k_u$ where the hits originating from the same particle as hit u (i.e. by checking if the neighbor labels match the true labels, as precision). Recall is defined as $\frac{1}{n} \sum_{u=1}^N \text{Prec}@k_u$ where the summation includes all the matched clusters in one event. Ideally, the recall value should closely resemble the perfect match efficiency in a physical case. This would mean that the number of reconstructed tracks includes all hits from the matched particle and excludes any other hits ².

The final evaluation metrics for different model sizes in Tracking-60k are presented in Fig.5, Fig.6, and Table 1. From previously studied models, we observed that performance could still be improved, particularly for efficient transformers. HEPT achieved up to 95.1% accuracy with around 1 million parameters. The most significant improvement was seen in FlatFormer, as its performance increased notably with a higher number of parameters. The Mamba-a architecture, consistently achieves the highest accuracy with fewer train-

²In LHC experiments, tracking efficiency requires that at least 75% of the hits in a track belong to the same particle. This criterion typically results in higher efficiency compared to the measured recall here

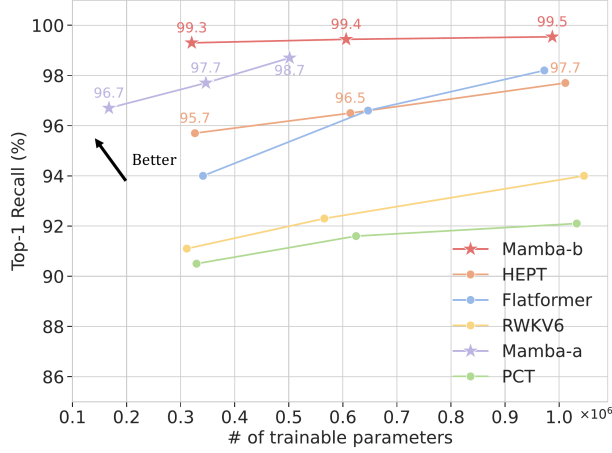


Figure 6: Top-1 recall of different-sized models on Tracking-60k scenario. Mamba-b demonstrates significantly better recall than all other larger models.

Table 3: Summary of top-1 accuracy and the top-1 area under curve score for ROC. **Bold** for best performance.

| Method | top-1 acc. | top-1 ROC |
|-------------------|--------------|--------------|
| Pileup-10k | | |
| HEPT-S | 40.35 | 75.56 |
| FlatFormer-S | 38.87 | 75.05 |
| PCT-S | 40.40 | 75.91 |
| Mamba-a-S | 40.39 | 75.77 |
| Mamba-b-S | 40.43 | 76.01 |

able parameters, already surpassing the best performance of HEPT with only about one-third of the model size. While measuring recall, HEPT or models that use hybrid HEPT attention typically result in lower recall at the same accuracy level. We found that the Mamba-b architecture, which relies solely on the pure Mamba backbone, achieves significantly higher recall, nearly approaching 1.

We have summarized our results in Table 2, comparing different models across varying numbers of hits. The best four models were selected for comparison. Mamba-a-S consistently delivers the highest accuracy across all datasets, while Mamba-b-S shows a notably higher recall than any other models for all datasets.

For pileup mitigation, we emphasize performance in the binary classification tasks. In Table 3, the best performance is achieved by the Mamba-b architecture. Additionally, we empirically found that small-sized PCT models can be trained to achieve performance

comparable to or even better than HEPT models and the hybrid Mamba-PCT model can therefore also deliver excellent performance.

6 LIMITATIONS

While we have demonstrated significant improvements compared to previous models, there are still a few limitations to be addressed.

Since the tasks in this study primarily focus on local inductive bias, a future direction could involve exploring the model’s robustness across a broader range of HEP tasks, where different patterns or dependencies may arise. As emphasized, different tasks might have different preferences toward the trade-off between computing throughput and physics performance.

We used a public dataset that contains limited statistics for pileup mitigation. In a more ideal scenario, having access to more detailed statistics could enable more comprehensive physical evaluations as done for the tracking experiment.

7 CONCLUSION

In this study, we presented one of the first attempts to bring SSMs into HEP tasks. Additionally, we introduced the use of OR & AND E2LSH before applying Mamba’s selective mechanism to integrate LSH-based attention into a hybrid Transformer-Mamba architecture, which is designed to tackle tasks with local inductive bias. This approach addresses two key challenges in HEP: HL-LHC tracking and pileup mitigation. Our quantitative metrics show that the proposed architecture reduces FLOPs by over 10× compared to SOTA models with recall significantly improved and accuracy remaining comparable or even better.

Code Availability

Our code is available at <https://github.com/chengjiang123/Lampa>.

Acknowledgements

We thank Dr. Huilin Qu, Dr. Yihui Lai, and Dr. Yongbin Feng for the fruitful discussions.

References

G. Aad et al. Observation of a new particle in the search for the standard model higgs boson with the atlas detector at the lhc. *Physics Letters B*, 716(1): 1–29, 2012. ISSN 0370-2693. doi: <https://doi.org/10.1016/j.physletb.2012.08.020>.

- O. Aberle et al. *High-Luminosity Large Hadron Collider (HL-LHC): Technical design report*. CERN Yellow Reports: Monographs. CERN, Geneva, 2020. doi: 10.23731/CYRM-2020-0010.
- Sabrina Amrouche et al. *The Tracking Machine Learning Challenge: Accuracy Phase*, page 231–264. Springer International Publishing, November 2019. ISBN 9783030291358. doi: 10.1007/978-3-030-29135-8_9. URL <https://www.kaggle.com/competitions/trackml-particle-identification/data>.
- Sabrina Amrouche et al. The tracking machine learning challenge : Throughput phase, 2021.
- Valerio Bertacchi. Deepcore: Convolutional neural network for high p_t jet tracking, 2020.
- Daniele Bertolini, Philip Harris, Matthew Low, and Nhan Tran. Pileup per particle identification. *Journal of High Energy Physics*, 2014(10), October 2014. ISSN 1029-8479. doi: 10.1007/jhep10(2014)059.
- Pierre Billoir. Track fitting with multiple scattering: A new method. *Nuclear Instruments and Methods in Physics Research*, 225(2):352–366, 1984. ISSN 0167-5087. doi: [https://doi.org/10.1016/0167-5087\(84\)90274-6](https://doi.org/10.1016/0167-5087(84)90274-6).
- Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: Going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, July 2017. ISSN 1558-0792. doi: 10.1109/msp.2017.2693418.
- Michael M. Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges, 2021.
- Matteo Cacciari, Gavin P. Salam, and Gregory Soyez. Softkiller, a particle-level pileup removal method. *The European Physical Journal C*, 75(2), February 2015. ISSN 1434-6052. doi: 10.1140/epjc/s10052-015-3267-2.
- Paolo Calafiura, Jay Chan, Loic Delabrouille, and Brandon Wang. Eggnets: An evolving graph-based graph attention network for particle track reconstruction, 2024.
- Serguei Chatrchyan et al. Observation of a New Boson at a Mass of 125 GeV with the CMS Experiment at the LHC. *Phys. Lett. B*, 716:30–61, 2012. doi: 10.1016/j.physletb.2012.08.021.
- Beidi Chen, Tri Dao, Eric Winsor, Zhao Song, Atri Rudra, and Christopher Ré. Scatterbrain: Unifying sparse and low-rank attention approximation, 2021.
- Krzysztof Choromanski, Valerii Likhoshesterov, David Dohan, Xingyou Song, et al. Rethinking attention with performers, 2022.
- Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality, 2024.
- Giannis Daras, Nikita Kitaev, Augustus Odena, and Alexandros G. Dimakis. Smyrf: Efficient attention using asymmetric clustering, 2020.
- Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the Twentieth Annual Symposium on Computational Geometry*, SCG '04, page 253–262, New York, NY, USA, 2004. Association for Computing Machinery. ISBN 1581138857. doi: 10.1145/997817.997857.
- Soham De, Samuel L. Smith, Anushan Fernando, et al. Griffin: Mixing gated linear recurrences with local attention for efficient language models, 2024.
- R. Frühwirth. Application of kalman filtering to track and vertex fitting. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 262(2):444–450, 1987. ISSN 0168-9002. doi: [https://doi.org/10.1016/0168-9002\(87\)90887-4](https://doi.org/10.1016/0168-9002(87)90887-4).
- Daniel Y. Fu, Tri Dao, Khaled K. Saab, Armin W. Thomas, Atri Rudra, and Christopher Ré. Hungry hungry hippos: Towards language modeling with state space models, 2023.
- Paolo Gloriosio, Quentin Anthony, Yury Tokpanov, James Whittington, Jonathan Pilault, Adam Ibrahim, and Beren Millidge. Zamba: A compact 7b ssm hybrid model, 2024.
- Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces, 2024.
- Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Re. Hippo: Recurrent memory with optimal polynomial projections, 2020.
- Albert Gu, Isys Johnson, Karan Goel, Khaled Saab, Tri Dao, Atri Rudra, and Christopher Ré. Combining recurrent, convolutional, and continuous-time models with linear state-space layers, 2021.
- Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces, 2022a.
- Albert Gu, Ankit Gupta, Karan Goel, and Christopher Ré. On the parameterization and initialization of diagonal state space models, 2022b.
- Albert Gu, Isys Johnson, Aman Timalsina, Atri Rudra, and Christopher Ré. How to train your hippo: State space models with generalized orthogonal basis projections, 2022c.
- Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yanzhi Li, Shean Wang, Lu Wang, and

- Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021.
- Sukjun Hwang, Aakash Lahoti, Tri Dao, and Albert Gu. Hydra: Bidirectional state space models through generalized matrix mixers, 2024.
- Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, page 604–613, New York, NY, USA, 1998. Association for Computing Machinery. ISBN 0897919629. doi: 10.1145/276698.276876.
- Xiangyang Ju, Steven Farrell, Paolo Calafiura, et al. Graph neural networks for particle reconstruction in high energy physics detectors, 2020.
- Raghav Kansal, Javier Duarte, Hao Su, Breno Orzari, Thiago Tomei, Maurizio Pierini, Mary Touranakou, Jean-Roch Vlimant, and Dimitrios Gunopulos. Particle cloud generation with message passing generative adversarial networks, 2022.
- Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention, 2020.
- Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer, 2020.
- Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.
- Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, et al. Jamba: A hybrid transformer-mamba language model, 2024.
- Kilian Lieret, Gage DeZoort, Devdoot Chatterjee, Jian Park, Siqi Miao, and Pan Li. High pileup particle tracking with object condensation, 2023.
- Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection, 2018.
- Ryan Liu, Paolo Calafiura, Steven Farrell, Xiangyang Ju, et al. Hierarchical graph neural networks for particle track reconstruction, 2023a.
- Zhijian Liu, Xinyu Yang, Haotian Tang, Shang Yang, and Song Han. Flatformer: Flattened window attention for efficient point cloud transformer, 2023b.
- Siqi Miao, Zhiyuan Lu, Mia Liu, Javier Duarte, and Pan Li. Locality-sensitive hashing-based efficient point transformer with applications in high-energy physics. *International Conference on Machine Learning*, 2024a.
- Siqi Miao et al. Locality-Sensitive Hashing-Based Efficient Point Transformer with Applications in High-Energy Physics, February 2024b. URL <https://doi.org/10.5281/zenodo.10694703>.
- Vinicius Mikuni and Florencia Canelli. Point cloud transformers applied to collider physics. *Machine Learning: Science and Technology*, 2(3):035027, July 2021. ISSN 2632-2153. doi: 10.1088/2632-2153/ac07f6.
- Bo Peng et al. Rwkv: Reinventing rnns for the transformer era, 2023.
- Huilin Qu and Loukas Gouskos. Jet tagging via particle clouds. *Physical Review D*, 101(5), March 2020. ISSN 2470-0029. doi: 10.1103/physrevd.101.056019.
- Huilin Qu, Congqiao Li, and Sitian Qian. Particle transformer for jet tagging, 2024.
- Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2007.
- M. Schuster and K.K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997. doi: 10.1109/78.650093.
- Alex Sherstinsky. Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network. *Physica D: Nonlinear Phenomena*, 404: 132306, March 2020. ISSN 0167-2789. doi: 10.1016/j.physd.2019.132306.
- A.M. Sirunyan et al. Particle-flow reconstruction and global event description with the cms detector. *Journal of Instrumentation*, 12(10):P10003–P10003, October 2017. ISSN 1748-0221. doi: 10.1088/1748-0221/12/10/p10003.
- Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive network: A successor to transformer for large language models, 2023.
- Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, et al. Going deeper with convolutions, 2014.
- Armen Tumasyan et al. The Phase-2 Upgrade of the CMS Tracker. 6 2017. doi: 10.17181/CERN.QZ28.FLHW.
- Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding, 2019.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity, 2020.

Lianghui Zhu, Bencheng Liao, Qian Zhang, Xinlong Wang, Wenyu Liu, and Xinggang Wang. Vision mamba: Efficient visual representation learning with bidirectional state space model, 2024.

Checklist

1. For all models and algorithms presented, check if you include:
 - (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
 - (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
 - (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes]
2. For any theoretical claim, check if you include:
 - (a) Statements of the full set of assumptions of all theoretical results. [Not Applicable]
 - (b) Complete proofs of all theoretical results. [Not Applicable]
 - (c) Clear explanations of any assumptions. [Not Applicable]
3. For all figures and tables that present empirical results, check if you include:
 - (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]
 - (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
 - (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
 - (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
 - (a) Citations of the creator If your work uses existing assets. [Yes]
 - (b) The license information of the assets, if applicable. [Yes]
 - (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
 - (d) Information about consent from data providers/curators. [Yes]
 - (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]
5. If you used crowdsourcing or conducted research with human subjects, check if you include:
 - (a) The full text of instructions given to participants and screenshots. [Not Applicable]
 - (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
 - (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

Supplementary material

A Details on experiment setup

A.1 Loss function choices

In tracking tasks, we improved the original loss function from (Miao et al., 2024a), where they adopted contrastive learning with InfoNCE loss (van den Oord et al., 2019).

$$\mathcal{L}_{\text{InfoNCE}} = -\log \frac{\exp(\text{sim}(h_u, h_v^+))}{\exp(\text{sim}(h_u, h_v^+)) + \sum_{h_v^- \in \mathcal{N}} \exp(\text{sim}(h_u, h_v^-))} \quad (1)$$

Pairs of hits from the same particle are labeled positive ($\{h^+\}$), while negative ($\{h^-\}$) pairs are selected from the nearest 256 hits (\mathcal{N}) belonging to other particles. The L_{RBF}^2 similarity

$$\text{sim}_{L_{\text{RBF}}^2}(h_u, h_v) = \exp\left(-\frac{d_{uv}^2}{2\sigma^2}\right), \quad d_{uv} = \|h_u - h_v\|_2 = \sqrt{\sum_{k=1}^{\mathcal{N}} (h_{u,k} - h_{v,k})^2} \quad (2)$$

is used. Originally, neighboring hits were pre-defined by constructing a radius graph, but this approach significantly increased the required training RAM and data size. To address this, we refine the neighbors with a kNN ($\mathcal{N} \rightarrow k$) leveraging the inductive bias on locality. Similar performance could be achieved with significantly reduced memory consumption by setting k to a small value (e.g., 32).

In pileup tasks, each sample event contains an unequal number of charged and neutral particles. To address this challenge, we use Focal Loss (Lin et al., 2018),

$$\mathcal{L}_{\text{F.L.}} = -\alpha_t (1 - p_t)^\lambda \log(p_t) \quad (3)$$

with λ set to 2, α set to 0.25 for class imbalance.

A.2 Model Hyperparameters

For the tracking task, it is crucial to maintain high throughput while pushing for better performance. Therefore, we tested our model with different scales to study the scaling behavior.

We chose to set up our experiment with three different scales of our model, denoted as Small (S), Medium (M), and Large (L).

The Small (S) model is designed to have a similar size, matching the 0.33M parameters of HEPT chosen in previous literature. The Medium (M) model doubles the size of the Small model, while the Large (L) model is scaled up to have approximately 1 million trainable parameters. This range of model sizes allows us to analyze the trade-offs between model complexity and performance across different tasks.

To scale up the models, two main variables, the embedding dimension of the hidden state and the number of layers, are varied. The embedding dimension of the hidden state is set to either 24 or 48. The number of layers is set to 4, 8, and 12. For the LSH hyperparameters, we choose the number of hashes (m1, m2) to be fixed at 3. The block size (Table 4) varies depending on the number of tracking hits and the chosen model, namely Mamba-a or Mamba-b.

Table 4: Block size used in different datasets

| Dataset | Mamba-a | Mamba-b |
|---------------------|-----------------|-----------------|
| Tracking-6k | [20, 40] | [100, 120] |
| Tracking-15k | [60, 80, 100] | [150, 200, 250] |
| Tracking-60k | [140, 150, 160] | [200, 250, 300] |
| Pileup-10k | [100, 120, 140] | [100, 150, 200] |

For the pile-up tasks, the performance is more crucial, hence we have one fixed hyperparameter configuration. The embedding dimension and number of layers are set to 24, 8 for Mamba-a, 48, 8 for Mamba-b, LSH hyperparameters are shown on the last row in Table. 4

B Physics Performance

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is a clustering algorithm that identifies groups of closely packed data points while marking points in low-density regions as outliers. The algorithm operates based on two primary parameters: Epsilon (ϵ) and MinPts. ϵ defines the maximum distance between two points for them to be considered neighbors, setting the radius of the neighborhood around a point. MinPts specifies the minimum number of points required to form a dense region; a point is considered a core point if it has at least MinPts neighbors within its ϵ -neighborhood.

In our study, we employed DBSCAN to assign track IDs to particle hits for computing LHC-style tracking efficiency. This efficiency metric requires that at least 75% of the hits in a reconstructed track originate from the same particle. We utilized the L2 norm as the distance metric for DBSCAN.

Our experiments yielded several key observations. Firstly, smaller MinPts values often result in better clustering outcomes. As MinPts increases, the algorithm becomes more stringent, leading to a decline in performance, similar conclusions as found in previous studies [Lieret et al. \(2023\)](#). Secondly, different models exhibit varying optimal ϵ values. Larger ϵ values can increase the number of points considered neighbor. The choice of DBSCAN parameters significantly impacts computational complexity, making it challenging to standardize DBSCAN parameters across models.

To ensure consistency in our comparison, we fixed the MinPts to 2 and constraint ϵ values within the range of 0.4 to 0.6. This approach allowed for a balanced assessment of DBSCAN's performance across different models and datasets.

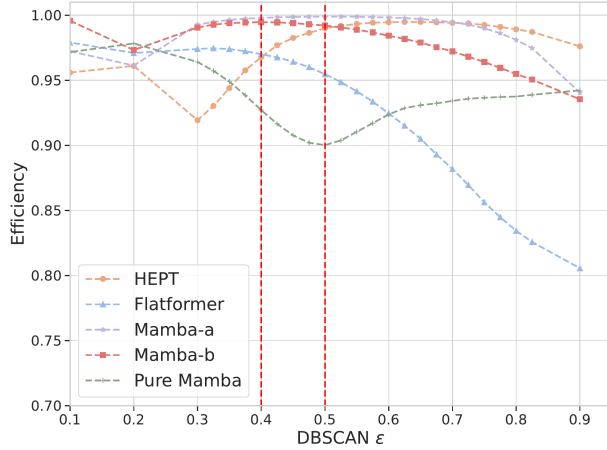


Figure 7: Performance plot for LHC-style tracking efficiency with the learned DBSCAN clustering, the minimum p_T cut is at 0.9 GeV. Red Dashed line indicates the proposed ϵ region to have consistent comparisons among different models.

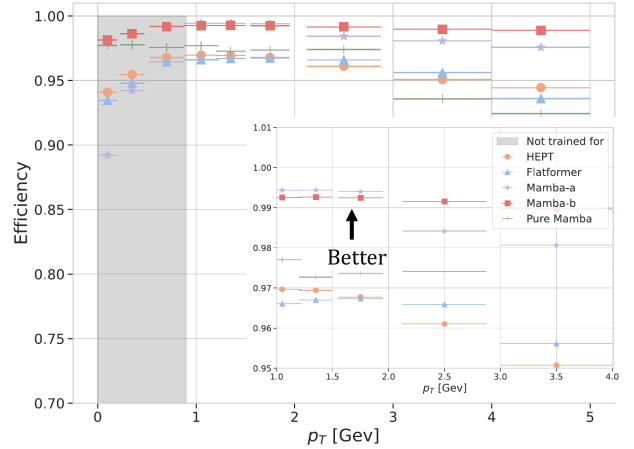


Figure 8: Performance plot for LHC-style tracking efficiency with transverse momentum bins. The Grey shaded areas is the region not being trained for.

These figures are evaluated from the perspective of a point cloud classification task. A possible future direction is to integrate these architectures into a more complete tracking reconstruction pipeline to improve performance evaluations. This could enhance accuracy and provide a clearer understanding of the physics.