

# SPECTRAL SPATIAL TRAVERSING IN POINT CLOUDS: ENHANCING DATA ANALYSIS WITH MAMBA NET- WORKS - SUPPLEMENTARY MATERIAL

Anonymous authors  
Paper under double-blind review

## 1 COMPUTATIONAL EFFICIENCY, RUNTIME, AND MEMORY USAGE

We have conducted a comprehensive analysis regarding the computational efficiency, runtime, and memory usage of our SAST approach. In fact, computing the first eigenvectors of the graph Laplacian is fast since the number of patches is much less than the original number of points (128 vs. 2048). Additionally, when increasing the number of tokens, we do not encounter computational and memory overhead issues due to several reasons. Because 1) the Laplacian matrix is very sparse as we only consider the  $K$  nearest neighbors of each patch ( $K \approx 20$ ), 2) we only compute the first  $k$  eigenvectors ( $k \approx 5$ ), and 3) this computation is done *only once* for each point cloud, the computational and memory overhead of this step is minimal.

Using the sparse solve of SciPy (function `eigs` implementing the implicitly restarted Arnoldi algorithm), our SAST strategy doesn't have any issues in terms of computational and memory overhead.

**Memory Usage:** As shown in Figure 1, the memory usage of our SAST strategy (black line) is significantly lower than the memory usage of the Point-Mamba backbone (red line). When we increase the number of patches along the x-axis, our strategy based on a sparse eigen-solver does not require substantially more memory compared to the backbone. The star in this figure shows the used number of tokens in downstream tasks.

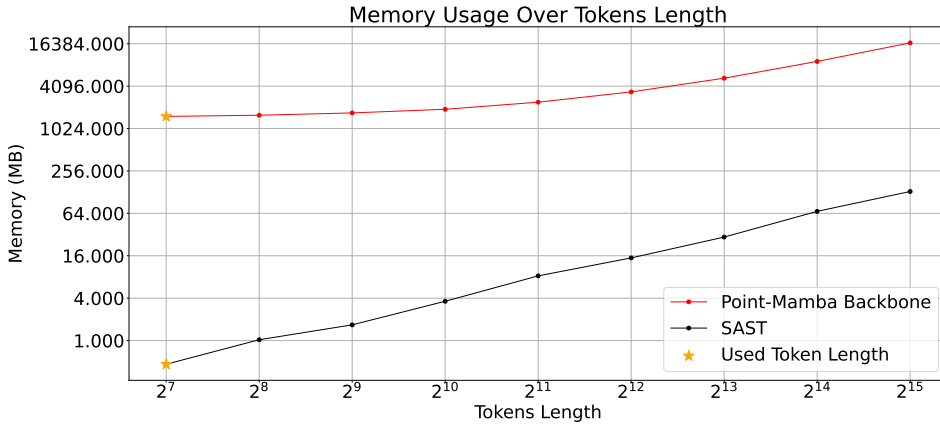


Figure 1: Memory Usage Over Tokens Length. Both axes are scaled by  $\log_2$  for better visualization.

**Runtime:** Our SAST strategy, which can be implemented in the data loader and ran in parallel on the CPU, is also fast. As can be seen in Figure 2, the runtime of SAST scales well when increasing the number of patches (tokens), and only a small amount of runtime is added in training or inference for the token length of 128 used in our main experiments (yellow star).

**FLOPS:** Figure 3 shows FLOPS as a function of token length for both the Point-Mamba Backbone and our method, SAST. As the token length increases, the Point-Mamba Backbone exhibits significantly higher computational complexity compared to SAST. Our method demonstrates a much more gradual

increase in FLOPS, indicating superior efficiency, particularly at longer token lengths. The yellow stars represent the token lengths used during evaluation, further illustrating the computational advantage of SAST over Point-Mamba.

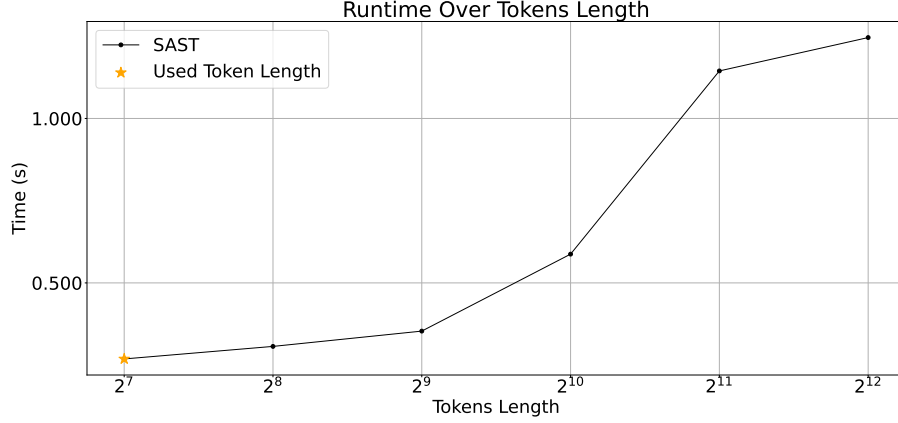


Figure 2: Runtime Over Tokens Length. Both axes are scaled by  $\log_2$  for better visualization.

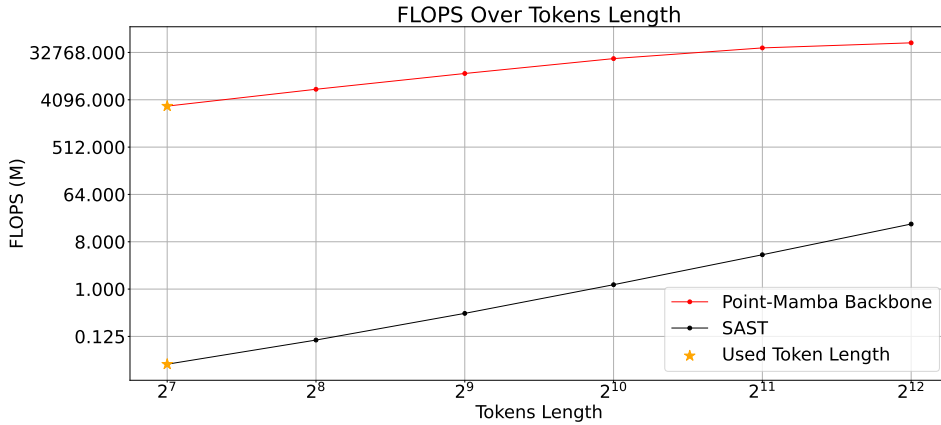


Figure 3: FLOPS Over Tokens Length. The horizontal axis is scaled by  $\log_2$  for better visualization.

## 2 ADDITIONAL ABLATION STUDY

In this section, we investigate the effect of the HLT strategy on the classification task. The results for HLT on the ObjectNN dataset are shown in Table 1. As observed, HLT underperforms compared to SAST, indicating that processing high-level information from different eigenvectors in separate traversals yields better results for this task.

## 3 RECONSTRUCTED POINTS.

To elucidate the capabilities of Masked Autoencoders (MAEs) in processing point cloud data, Fig. 4 provides a visual sequence involving the original input, the intermediate masking phase, and the reconstructed output. The first column, titled “Input Point Cloud”, displays the entirety of the point cloud data, illustrating the initial condition before any processing. The subsequent column, “Masked Point Cloud”, reveals only the points that remain visible after a portion of the data has been masked. The final column, “Reconstructed Point Cloud”, demonstrates the model’s ability to infer and restore the masked parts of the point cloud.

Table 1: Object classification on ScanObjectNN. Accuracy (%) is reported.

Methods	Backbone	Param. (M)	FLOPs (G)	OBJ-BG	OBJ-ONLY	PB-T50-RS
<i>Training from scratch</i>						
Ours (HLT)	Mamba	12.3	3.6	90.87	90.53	86.22
Ours (SAST)	Mamba	12.3	3.6	<b>92.42</b>	<b>91.39</b>	<b>87.61</b>
<i>Training from pretrained</i>						
Ours (HLT)	Mamba	12.3	3.6	91.80	91.42	87.52
Ours (SAST)	Mamba	12.3	3.6	<b>94.32</b>	<b>92.08</b>	<b>89.10</b>

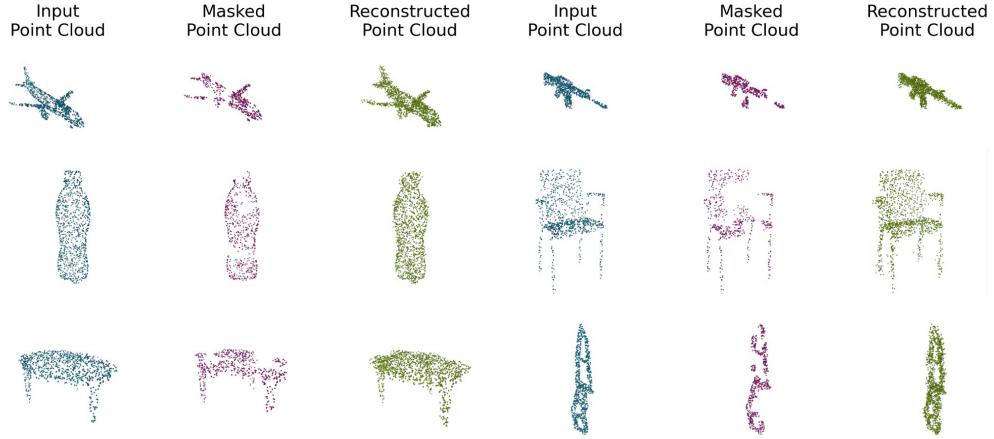


Figure 4: Reconstruction results on the ShapeNet dataset