Token Pruning for Improving Graph-Generating State Space Model Performance

Monish Beegamudre
Algoverse AI Research
m.beegamudre@ufl.edu

Jack Zheng
Algoverse AI Research
jaz137@pitt.edu

Margaret Capetz
Algoverse AI Research
mcapetz@uw.edu

Abstract

State Space Models (SSMs) have recently emerged as efficient alternatives to Transformers for sequence modeling, yet extending them to two-dimensional tasks remains challenging. The Graph-Generating State Space Model (GG-SSM) addresses this challenge by constructing an adaptive graph, achieving competitive performance on vision benchmarks. However, state propagation over the resulting graph introduces substantial inference overhead, limiting scalability to high-resolution inputs. In this work, we introduce a leaf-guided computation pruning strategy that accelerates GG-SSM inference without modifying the underlying graph topology. Rather than removing nodes or edges, our approach selectively scales or bypasses secondary refinement computations associated with high-dissimilarity leaf nodes, while preserving the low-weight MST backbone. Experiments on multiple long-term time series forecasting benchmarks demonstrate consistent throughput improvements with controlled accuracy degradation across a range of pruning ratios. These results indicate that structure-aware computation pruning is an effective mechanism for improving the scalability of graph-based state space models. Our source code is publicly available at https://github.com/MonishB123/token-pruning-for-ggssm

1 Introduction

Modern sequence modeling tasks increasingly demand models that capture long-range dependencies while remaining computationally efficient. As datasets grow in length and complexity, it is becoming more difficult to design architectures that balance expressive power with scalability. State Space Models (SSMs) provide a fundamentally different way to process sequences. Instead of relying on pairwise attention interactions, SSMs maintain a hidden state that is updated as new inputs arrive, enabling computation that scales linearly with sequence length. Early SSM variants like Mamba (Gu and Dao, 2024) demonstrated strong potential, but were often limited by fixed state-update rules that restricted their expressiveness on multidimensional complex data. Mamba and its successors typically operate on fixed, one-dimensional scan orders, which limit their expressiveness when applied to data with richer structural dependencies. However, real-world data often exhibits complex and non-uniform relationships that do not align with predetermined scan orders.

To address this, the Graph-Generating State Space Model (GG-SSM) extends Mamba's selective state update mechanism by learning an adaptive graph over the input tokens. Instead of processing data in a predetermined order, GG-SSM constructs a graph using Chazelle's MST algorithm. GG-SSM achieves strong performance on structured, spatially complex data, but its gains come with heavy computational overhead. In particular, propagating information over an adaptive graph incurs substantial cost as the number of tokens increases, limiting the practicality of GG-SSM for high-resolution inputs.

We propose a pruning strategy that substantially reduces the amount of computation performed during GG-SSM's graph-based state propagation without removing tokens or altering the graph structure. Prior pruning techniques developed for vMamba show that a considerable fraction can be removed with negligible impact on accuracy. We adapt pruning to GG-SSM by selectively scaling or skipping secondary refinement paths on high-dissimilarity leaf nodes while preserving the full backbone and reducing inference cost. This token pruning approach provides a new path toward a new generation of efficient and scalable SSMs that can handle high-resolution data without sacrificing performance.

Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics
Volume 4: Student Research Workshop, pages 476–482

2 Related Works

Token Pruning for vMamba To adapt SSMs to two-dimensional image data, researchers developed vMamba (Vision Mamba, an extension of Mamba designed specifically for vision tasks. vMamba introduces a 2D Selective Scan (SS2D) that scans images in multiple directions, effectively "linearizing" 2D information so it can be processed through Mamba's efficient state-space computation (Liu et al., 2024). Processing every token in an image remains computationally costly. Token pruning has been adapted to SSMs to enhance efficiency Zhan et al. (2024) by quantifying each token's influence on the hidden state and selectively discarding redundant tokens. Building on these ideas, QuarterMap Chi et al. (2025) introduces a token pruning strategy designed to accelerate vMamba's processing. The method reduces the token set to roughly one-quarter of the original size before the 2D scan.

Graph-Generating State Space Models Another approach to addressing the challenges of extending SSMs to higher-dimensional data is the Graph-Generating State Space Model (GG-SSM). Unlike vMamba, which relies on fixed scan patterns to linearize input data, GG-SSM constructs an adaptive graph over the input tokens, enabling the model to capture complex spatial and long-range dependencies that are not aligned with predetermined trajectories. The graph is generated using Chazelle's MST algorithm, which produces a sparse, low-weight backbone connecting the most informative tokens. State updates are then propagated along the edges of this graph, allowing the model to integrate in a structure-aware manner.

Empirical results demonstrate that GG-SSM provides performance gains on structured and spatially complex datasets. For instance, it achieves a 0.33 increase in detection rates on event-based eye-tracking datasets, outperforms prior SSMs by 1% on the ImageNet benchmark, and reduces the error rate to 2.77% on the KITTI-15 dataset (Zubić and Scaramuzza, 2025). These improvements highlight the ability of adaptive graph-based state-space models to capture data that fixed-scan approaches struggle to represent.

Pruning Approach for GG-SSMs Unlike prior pruning strategies in visual state space models that physically remove tokens from fixed grids prior to scanning, our approach introduces a structure-aware mechanism designed specifically for graph-based architectures. While methods like vMamba reduce sequence length by pruning tokens before or during the scan, we perform pruning after the Minimum Spanning Tree (MST) has been constructed. This timing allows the model to leverage the full global context to form its dependency structure before identifying redundancies. Furthermore, our strategy shifts the focus from spatial token removal to the deactivation of computational paths conditioned on the MST topology.

3 Methods

Edge Weights in GG-SSM In GG-SSM, each edge in the MST encodes the dissimilarity between a pair of nodes, which are typically feature vectors representing tokens or spatial locations. Edge weights are commonly computed using the exponential cosine distance, see Appendix A. In this formulation, higher weights correspond to greater dissimilarity, while smaller weights indicate stronger similarity between features.

The MST algorithm then selects the $n-1$ edges with the lowest weights to construct a connected tree over all nodes. These low-weight edges form the backbone of the MST, connecting the most similar or highly correlated features first. Information propagated along these edges captures the strongest relationships among features and is therefore critical for the state-space updates in GG-SSM.

Conversely, edges with higher weights generally connect nodes that are more weakly related or peripheral to the main structure. These edges tend to appear later in the MST construction process and contribute less to the overall information flow. By selectively removing high-weight edges and their corresponding leaf nodes, we can reduce computational complexity while preserving the MST's key feature relationships.

Pruning Approach Our pruning approach targets leaf nodes connected via high-weight edges, as seen in Figure 1. After the MST is constructed, we identify these peripheral nodes and reduce computation along their associated refinement paths. This preserves the low-weight backbone while skipping less informative operations, focusing resources on the most critical dependencies. See Appendix B for the formal pruning procedure.

Leaf-guided Computation Pruning To improve inference efficiency without modifying the MST structure, we implement pruning by reducing the

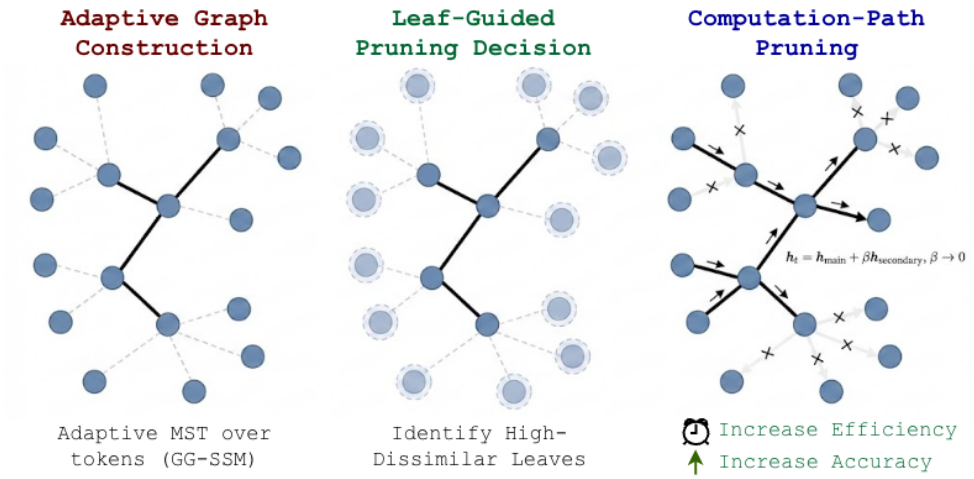

Figure 1: Token pruning procedure in Graph-Generating SSMs. Input tokens undergo MST construction using similarity-based edge weights, followed by leaf identification. Pruning targets high-weight leaf nodes by scaling or skipping the secondary refinement path, while preserving the main refinement along the MST backbone, enabling efficient inference with minimal accuracy loss.

contribution of a secondary tree-based refinement path rather than physically removing nodes or edges. After constructing the MST, leaf nodes are identified and a pruning ratio $r \in [0, 1]$ determines the fraction of peripheral computation to skip or scale. Each node is updated through a primary path for backbone propagation and a secondary path, whose contribution, scaled by β, diminishes with the pruning ratio and is skipped once a threshold is reached; see Appendix B for details.

This approach maintains tensor shapes and CUDA kernel compatibility, enabling a dynamic, per-batch mechanism for trading off computation and model capacity. By selectively scaling or skipping the secondary refinement path, inference throughput is improved while the core state propagation along the MST is preserved. See the Pruning Algorithm Flow in Appendix G.

Pruning Ratio and Path Deactivation The pruning ratio $r \in [0, 1]$ is a hyperparameter that governs the trade-off between computational efficiency and model capacity. In our implementation, r serves as a threshold for path deactivation rather than a simple node-count reduction. Specifically, the scaling factor β for the secondary path is modulated by a pruning factor: $\alpha = \max\left(0, 1 - \frac{r}{0.3}\right)$

Consequently, when $r \geq 0.3$, the secondary refinement path is entirely bypassed ($\beta = 0$), allowing the system to skip the associated CUDA kernel calls and maximize inference speed. Because the distribution of leaf nodes is sample-dependent, this ratio-based approach ensures that computational reduction is adaptive to the specific topology of each input graph, providing a consistent mechanism for efficiency control across varying sequence complexities.

Preserving the MST Backbone The effectiveness of our pruning strategy arises from the MST's inherent structure. With our approach, we achieve efficiency by bypassing the computational overhead of the CUDA kernel when the pruning ratio r indicates that the peripheral leaf contributions are negligible. Because the primary path $h_t^{(1)}$ always utilizes the full sequence backbone, the model maintains its ability to capture long-range dependencies while saving arithmetic operations on the least informative semantic connections.

This approach contrasts with pruning low-weight leaves, which would disconnect strongly correlated nodes and break critical information paths, leading to performance degradation. Our method provides a principled trade-off between efficiency and accuracy, focusing pruning on the edges that contribute least to the MST's core structure.

4 Datasets

We test the performance of our token reduction strategies on three time series forecasting datasets, similar to the original GG-SSM paper. We chose these datasets specifically to see if the GG-SSM can maintain the feature relationships between particularly volatile time series while the graph is pruned. **ETTm1**: The data of two Electricity Transformers at two stations, recording load and oil temperature

every minute. The dataset tracks seven variables for 69,680 timestamps. (Zhou et al., 2021)

SolarAV: The solar power production records of 137 photovoltaic plants in Alabama State, sampled every ten minutes for 52,560 timestamps. (National Laboratory of the Rockies, 2025)

ETTh1: The same data of two electric power transformers, recording load and oil temperature every hour. This dataset tracks seven variables for 17,420 timestamps.(Zhou et al., 2021)

5 Results

The following figures illustrate the effect of our pruning technique on the throughput and accuracy of GG-SSM's across our three selected datasets. See additional results across different prediction lengths in Appendix D.

Dataset	Pruning Ratio	Random (Unordered)		Similarity-Guided	
		MSE	MAE	MSE	MAE
ETTm1	15.0%	0.445	0.464	0.395	0.438
	27.5%	0.510	0.500	0.402	0.445
	40.0%	0.555	0.523	0.418	0.459
ETTh1	15.0%	0.648	0.521	0.382	0.418
	27.5%	0.656	0.524	0.391	0.429
	40.0%	0.670	0.528	0.402	0.437
SolarAV	15.0%	0.550	0.483	0.168	0.293
	27.5%	0.551	0.484	0.172	0.298
	40.0%	0.553	0.484	0.179	0.306

Table 1: Effect of unordered pruning on ETTm1, ETTh1, and SolarAV datasets at 96 prediction length, compared to our similarity-guided pruning. Similarity-guided pruning significantly outperforms random pruning, reducing MSE by up to 68% while improving MAE by up to 37% for the SolarAV dataset.

Our strategy lowers the number of edges involved in state propagation, improving throughput and reducing inference time. This effect is most pronounced at higher pruning ratios, where fewer active connections allow more efficient propagation across all datasets. However, pruning the GG-SSM backbone up to 40% consistently speeds inference by $20 - 63\%$ across all datasets while keeping MSE degradation below 8.5%. The SolarAV dataset exhibits the largest speedups because its dense MSTs contain far more edges than the low-dimensional ETT series, making a higher fraction of connections prunable without disconnecting the graph or harming state propagation. At longer prediction horizons, throughput gains decrease as the GG-SSM preforms more full forward-passes within the MST, which reduces the effectiveness of the pruning. Furthermore, the larger amount of nodes in the graph being pruned likely accumulates larger errors as the graph is propagated, explaining why longer horizons have lower accuracy. Overall, preserving the MST backbone allows structured pruning with only minimum accuracy loss.

6 Discussion

Note that throughput times do not increase significantly for pruning ratios below 40%, suggesting that pruning overhead initially offsets its potential benefits. Once the graph removes enough nodes, propagation accelerates, and both inference and throughput times improve markedly.

As shown in Appendix F, structured pruning of the GG-SSM backbone induces only modest accuracy degradation even at higher pruning ratios. Across all three long-term forecasting tasks, relative MSE remains below $1.09\times$ the unpruned baseline at 40% pruning, with the low-dimensional Electrical Tracking datasets exhibiting the smallest degradation and the high-dimensional SolarAV dataset showing a slightly steeper but still limited increase.

To validate the importance of similarity-aware pruning order, we compare our strategy against random (unordered) pruning of the same ratio. As shown in Table 1, random pruning leads to larger accuracy degradation across all datasets, even at moderate ratios, while our ordered approach maintains near-baseline performance up to 40% pruning. This contrast demonstrates that removing highly redundant edges minimally disrupts critical long-range dependencies captured by the MST. In contrast, random removal frequently eliminates important structural connections, confirming that similarity-guided ordering is needed to accurately make predictions with GG-SSM's.

7 Conclusion

We have shown that token pruning can effectively reduce the computational cost of the GG-SSM while preserving its core representational capabilities. By selectively pruning high-dissimilarity leaf nodes from the MST, the model reduces the number of edges and nodes involved in state propagation, enabling faster inference. This approach demonstrates that GG-SSM can be made more efficient and scalable without sacrificing accuracy, highlighting structured pruning as a general strategy for improving the practicality of state-space models in high-dimensional and dense-input settings.

Limitations

Although the current experimentation was done within the scope of time series forecasting tasks, future work includes testing the pruning strategy for image tasks. GG-SSM's have been proven to have state of the art performance on classification tasks like ImageNet, Eye Tracking and Gaze Tracking. Further experimentation could apply our pruning technique to see a speed-up for inference times with these datasets. Future work may also include evaluating higher pruning ratios and longer prediction horizons to help characterize scaling behavior, revealing whether the observed trends persist as a larger fraction of edges are removed and GG-SSMs operate at increased sequence lengths.

A major limitation in the speedup of the pruning method was the overhead the technique produced on very dense graphs. A promising direction for future work is to reduce redundancy before MST construction by merging highly similar tokens. For example, via a lightweight k-NN clustering pass that collapses near-identical nodes into single vertices. This approach would shrink the graph early, reducing the computational burden of MST construction, similar to the kNN-Borůvka GPU method (Arefin et al., 2012), which efficiently builds MSTs on k-NN graphs by limiting the number of candidate edges. Incorporating such a pre-processing step could lower the cost of Chazelle's MST algorithm on high-dimensional inputs and potentially enable higher pruning ratios without sacrificing accuracy. A key challenge will be to perform such merging in a way that preserves critical long-range dependencies, ensuring that the resulting spanning tree still captures the essential structure of the sequence.

Acknowledgments

We would like to thank Lambda for their support in providing credits for access to GPU instances.

References

A. S. Arefin, C. Riveros, R. Berretta, and P. Moscato. 2012. knn-borůvka-gpu: A fast and scalable mst construction from knn graphs on gpu. In *Computational Science and Its Applications – ICCSA 2012*, volume 7333 of *Lecture Notes in Computer Science*, pages 78–93, Berlin, Heidelberg. Springer.

Tien-Yu Chi, Hung-Yueh Chiang, Diana Marculescu, and Kai-Chiang Wu. 2025. Quartermap: Efficient post-training token pruning for visual state space models. *Preprint*, arXiv:2507.09514.

Albert Gu and Tri Dao. 2024. Mamba: Linear-time sequence modeling with selective state spaces. *Preprint*, arXiv:2312.00752.

Yue Liu, Yunjie Tian, Yuzhong Zhao, Hongtian Yu, Lingxi Xie, Yaowei Wang, Qixiang Ye, Jianbin Jiao, and Yunfan Liu. 2024. Vmamba: Visual state space model. *Preprint*, arXiv:2401.10166.

National Laboratory of the Rockies. 2025. Solar power data for integration studies. `https://www.nrel.gov/grid/solar-power-data`.

Zheng Zhan, Zhenglun Kong, Yifan Gong, Yushu Wu, Zichong Meng, Hangyu Zheng, Xuan Shen, Stratis Ioannidis, Wei Niu, Pu Zhao, and Yanzhi Wang. 2024. Exploring token pruning in vision state space models. *Preprint*, arXiv:2409.18962.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. 2021. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *The Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Virtual Conference*, volume 35, pages 11106–11115. AAAI Press.

Nikola Zubić and Davide Scaramuzza. 2025. Gg-ssms: Graph-generating state space models. *Preprint*, arXiv:2412.12423.

A Exponential Cosine Distance

$$w_{ij} = \exp\left(-\frac{x_i^\top x_j}{|x_i||x_j|}\right) \tag{1}$$

where x_i and x_j denote the feature vectors of nodes i and j, respectively. We compute node simularity using exponential cosine distance (Zubić and Scaramuzza, 2025).

B Formal Pruning Procedure

Formally, given an MST $T = (V, E)$ with $|V| = n$ nodes and edge weight function $w : E \to R^+$. Our pruning procedure proceeds as follows:

1. Identify leaf nodes

$$L = \{v \in V : deg(v) = 1\}$$

2. For each leaf node $\ell \in L$, retrieve its connecting edge weight w_ℓ

3. Sort leaves by weight

$$w_{\ell_1} \geq w_{\ell_2} \geq \ldots \geq w_{|L|}$$

4. Prune top-k leaves

$$k = \lfloor r \cdot |L| \rfloor$$

This procedure ensures that the most dissimilar peripheral nodes are removed first, while the strongly-connected core structure—the backbone that supports long-range dependency modeling—remains intact. By systematically pruning high-weight leaves, the method reduces computational complexity and memory usage during state propagation, enabling GG-SSM to scale efficiently to larger or higher-resolution inputs.

C Training Setup

We set the learning rate to 0.001, used the AdamW Optimizer with a decay rate of 0.05. Each model was trained for 200 epochs. For the ETTm1 and ETTh1 models, we used a hidden dimension size of 512 and a batch size of 32, while the SolarAV model used a hidden dimension size of 32 and a batch size of 16.

D Additional Results

Pruning Ratio	0.0%	15.0%	27.5%	40.0%
Prediction Length 96				
Throughput (samples/s)	471.3	464.1	460.4	609.1
MSE	0.352	0.357	0.364	0.373
MAE	0.398	0.404	0.411	0.419
Prediction Length 192				
Throughput (samples/s)	344.8	342.7	345.6	415.3
MSE	0.389	0.395	0.402	0.418
MAE	0.431	0.438	0.445	0.459

Table 2: Effect of pruning on the Electrical Tracking Minute (ETTm1) dataset. At 40% pruning (prediction length 96), throughput reaches 609.1 samples/s (+25.3%), with MSE increasing from 0.389 to 0.418 (+5.75%).

Pruning Ratio	0.0%	15.0%	27.5%	40.0%
Prediction Length 96				
Throughput (samples/s)	470.8	488.3	436.1	621.6
MSE	0.378	0.382	0.391	0.402
MAE	0.412	0.418	0.429	0.437
Prediction Length 192				
Throughput (samples/s)	328.1	332.1	330.5	394.0
MSE	0.412	0.419	0.427	0.441
MAE	0.448	0.455	0.462	0.479

Table 3: Effect of pruning on the Electrical Tracking Hour (ETTh1) dataset. At 40% pruning (pred. len. 96), throughput reaches 621.6 samples/s (+32%) with MSE increasing from 0.378 to 0.402 (+6.3%).

Pruning Ratio	0.0%	15.0%	27.5%	40.0%
Prediction Length 96				
Throughput (samples/s)	436.3	595.4	658.4	712.5
MSE	0.165	0.168	0.172	0.179
MAE	0.289	0.293	0.298	0.306
Prediction Length 192				
Throughput (samples/s)	126.2	133.8	128.8	131.4
MSE	0.189	0.194	0.199	0.208
MAE	0.321	0.328	0.334	0.343

Table 4: Effect of pruning on the SolarAV dataset (137 variables). At 40% pruning and prediction length 96, throughput jumps from 436.3 to 712.5 samples/s (+63%) with MSE rising only from 0.165 to 0.179 (+8.5%). Gains are smaller at longer horizons.

E Leaf-Guided Computation Pruning

Each node h_t is updated using two parallel refinement paths along the Minimum Spanning Tree (MST): a primary backbone path and a secondary refinement path. Formally, the updates are defined as

$$\begin{aligned} h_t^{(1)} &= \text{Refine}_{\text{main}}(h_t, h_{\text{parent}(t)}, \\ &\qquad w_{t,\text{parent}(t)}) \\ h_t^{(2)} &= \text{Refine}_{\text{secondary}}(h_t, h_{\text{parent}(t)}, \\ &\qquad w_{t,\text{parent}(t)}) \\ h_t^{\text{final}} &= h_t^{(1)} + \beta h_t^{(2)} \end{aligned}$$

where $w_{t,\text{parent}(t)}$ denotes the edge weight between node t and its parent in the MST. The contribution of the secondary refinement path is controlled by a scaling factor $\beta \in [0, 1]$, which is computed as

$$\beta = \beta_{\text{base}} \cdot \max\left(0,\, 1 - \frac{r}{r_{\text{threshold}}}\right),$$

where $\beta_{\text{base}} = 0.3$, $r \in [0, 1]$ is the pruning ratio, and $r_{\text{threshold}}$ denotes the pruning level at which the secondary refinement path is fully disabled.

For larger pruning ratios, β decreases monotonically and becomes zero once $r \geq r_{\text{threshold}}$, effectively bypassing the secondary refinement computation. This formulation enables computation-level pruning without modifying the MST topology or tensor shapes, preserving compatibility with the original GG-SSM implementation while reducing inference cost.

F MSE Degradation vs. Pruning Ratio

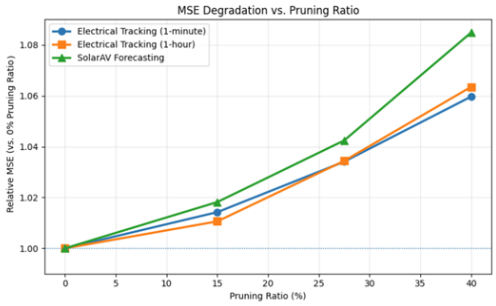

Figure 2: MSE degradation versus pruning ratio across three forecasting tasks with a Prediction Length of 96. All values are normalized to the unpruned (0%) model.

G Pruning Algorithm

Algorithm 1 Leaf-Guided Computation Pruning

Require: Input features X, MST structure T, Pruning Ratio r

Initialize: $\beta \leftarrow \max(0, 1 - r/0.3)$
Primary Path (Backbone):
  $H^{(1)} \leftarrow \text{TreeScan}(X, T, \text{weights} = W_{mst})$
Secondary Path (Refinement):
if $r \geq 0.3$ **then**
    $H^{(2)} \leftarrow 0$ ▷ *Prune computation completely*
else
    $H^{(2)} \leftarrow \text{TreeScan}(X, T, \text{weights} = W_{aux})$
    $H^{(2)} \leftarrow H^{(2)} \times \beta$ ▷ *Scale by pruning factor*
end if
Aggregation:
  $Y \leftarrow H^{(1)} + 0.3 \cdot H^{(2)}$

Ensure: Contextualized states Y