Applying Mamba to Graph Neural Networks

Jyotirmaya Shivottam
23226001
CS660/CS460 - Machine Learning, Spring 2024
Subhankar Mishra Lab
National Institute of Science Education and Research
An OCC of Homi Bhabha National Institute

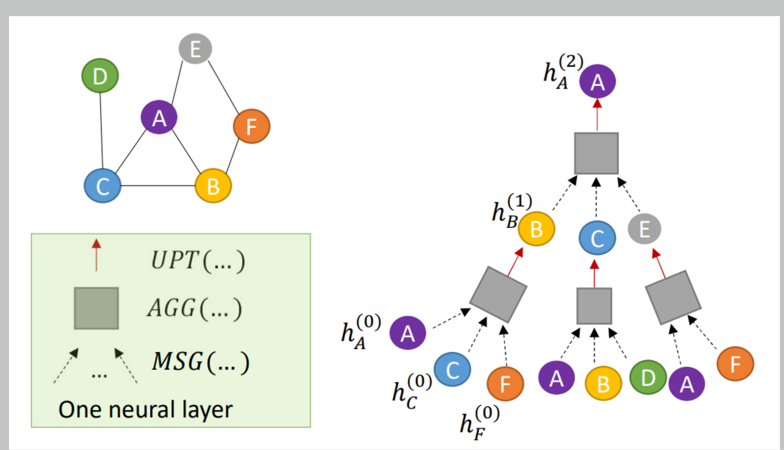
Goals of the Project

- To investigate whether Structured Space Models (SSMs), specifically Mamba (S6), can be applied to state updates in Graph Neural Networks (GNNs).
- To implement such a graph-based model and to benchmark against existing baseline GNNs, for instance:
  - 2015: Gated Graph Sequence Neural Networks (GGSNN), arxiv:1511.05493.
  - 2016: Graph Convolutional Networks (GCN), arxiv:1609.02907.
  - 2017: Graph Attention Networks (GATN), arxiv:1710.10903.
  - 2017: Graph Sample and Aggregate (GraphSAGE), arXiv:1706.02216.
  - 2018: Graph Isomorphism Networks (GIN), arxiv:1810.00826.
  - 2023: Spatio-Temporal Adaptive Embedding Transformer (STAEformer), arXiv:2308.10425.

Primer: Graph Neural Networks (GNNs)

- GNNs utilize *Message Passing* to update the state of each node in a graph, $\mathcal{G}(\mathcal{V}, \mathcal{E})$, based on the states of its neighbors, like so:

$$m_i^{(l)} = \texttt{MESSAGE}\left(\{h_j^{(l-1)} : j \in \mathcal{N}[i]\}\right)$$
$$a_i^{(l)} = \texttt{AGGREGATE}\left(\{m_j^{(l-1)} : j \in \mathcal{N}[i]\}\right)$$
$$h_i^{(l)} = \texttt{UPDATE}\left(h_i^{(l-1)}, a_i^{(l)}\right)$$



- These functions must be *permutation-invariant*, as the graph-data is inherently *unstructured*. Usually, MPNN variants differ in how they define (model) these functions.
  - GCN updates node features via spectral graph convolutions.
  - GGSNN uses Gated Recurrent Unit (GRU) update for node features followed by neighborhood sum aggregation.
  - GraphSAGE trains aggregator functions on different hops of sampled node neighborhoods.
  - GAT uses *self-attention* mechanism on node neighborhoods with sum aggregation.
  - Spatio-temporal (Dynamic) GNNs utilize message passing (and mixing) in both the spatial and temporal domains.

Motivation

- Aforementioned models are limited by the expressiveness of the 1-Weisfeiler-Lehman (1-WL) graph isomorphism test (cf. GIN).
- This constraint is particularly challenging for graph data with long-range dependencies, e.g., in social network analysis or bioinformatics.
- While models like GAT perform well, their $\mathcal{O}(N^2)$ complexity limits their scalability for large graphs.
- Mamba, a recently proposed time-varying state-space model (SSM) for sequence modeling, scales linearly with sequence length.
- **Our goal is to adapt Mamba to GNNs by treating the state update of each node as a sequence.**

SSM-based Sequence Modeling via Mamba

- Proposed by Gu et al. (De, 2023) to efficiently model *seq2seq* maps on **long sequences**.
- SSMs, including Mamba, relate a continuous input sequence $x(t) \in \mathbb{R}$ to an output sequence $y(t) \in \mathbb{R}$ via an implicit latent state $h(t) \in \mathbb{R}^d$, using the following first-order ODE (**cf. Recurrent Neural Networks**):

$$h'(t) = Ah(t) + Bx(t)$$
$$y(t) = Ch(t)(+Dx(t))$$

- Here, $A \in \mathbb{R}^{d \times d}$, $B \in \mathbb{R}^{d \times 1}$, $C \in \mathbb{R}^{1 \times d}$, and $D \in \mathbb{R}$ are the state transition matrix, the input matrix, the output matrix, and the feedforward matrix, respectively.
- As we work with discrete data, the equations are discretized, e.g., using Zero-Order Hold (ZOH):

$$\bar{A} \stackrel{\text{ZOH}}{:=} \exp(\Delta A)$$
$$\bar{B} \stackrel{\text{ZOH}}{:=} (\Delta A)^{-1}(\exp(\Delta A) - \mathbb{I}) \cdot \Delta B$$
$$h_t = \bar{A}h_{t-1} + \bar{B}x_t$$
$$y_t = Ch_t$$

- Mamba makes Δ, which denotes a learned step size, input-dependent, indirectly making A, B, and C input-dependent and thus, context-aware.
- A is *structured* as a diagonal matrix to stabilize state update and simplify computation.
- Latent state $h(t)$ stores compressed context, leading to higher efficiency than transformer-based approaches.
- Mamba's computational complexity scales as $\mathcal{O}(N)$, making it practical for sequence modeling tasks.
- Clever hardware-aware implementation of associative scan algorithm enables efficient training on modern hardware.
- Typical Mamba block structure includes linear projection, depth-wise convolution, SSM update, and interleaved non-linearities.

Structure of Mamba

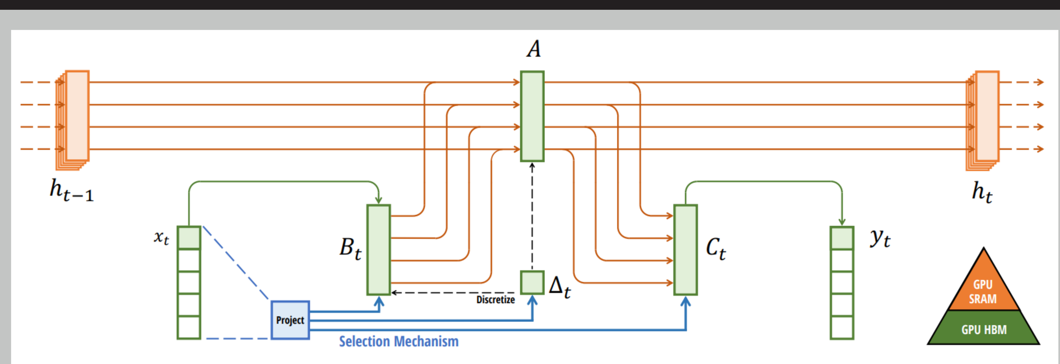
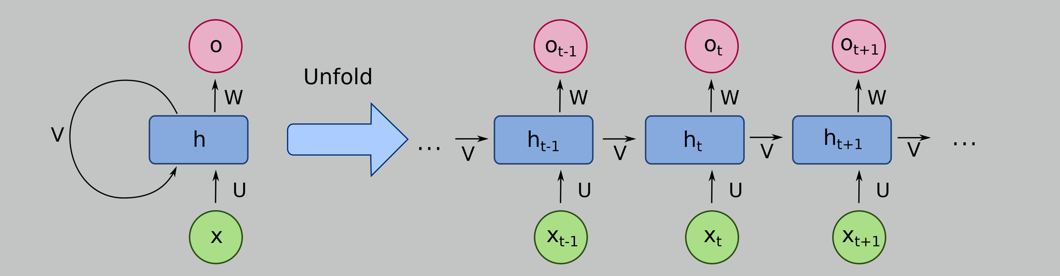
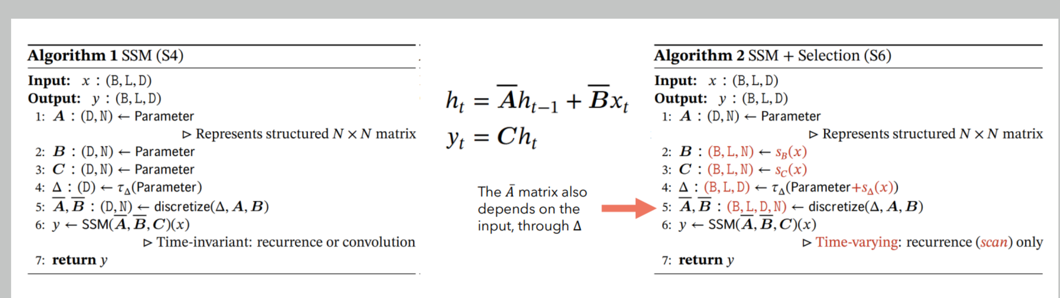


Figure 1: (**Overview.**) Structured SSMs independently map each channel (e.g. $D = 5$) of an input x to output y through a higher dimensional latent state h (e.g. $N = 4$). Prior SSMs avoid materializing this large effective state (DN, times batch size B and sequence length L) through clever alternate computation paths requiring time-invariance: the (Δ, A, B, C) parameters are constant across time. Our selection mechanism adds back input-dependent dynamics, which also requires a careful hardware-aware algorithm to only materialize the expanded states in more efficient levels of the GPU memory hierarchy.





$s_B(x) = \text{Linear}_N(x)$, $s_C(x) = \text{Linear}_N(x)$, $s_\Delta(x) = \text{Broadcast}_D(\text{Linear}_1(x))$, and $\tau_\Delta = \text{softplus}$

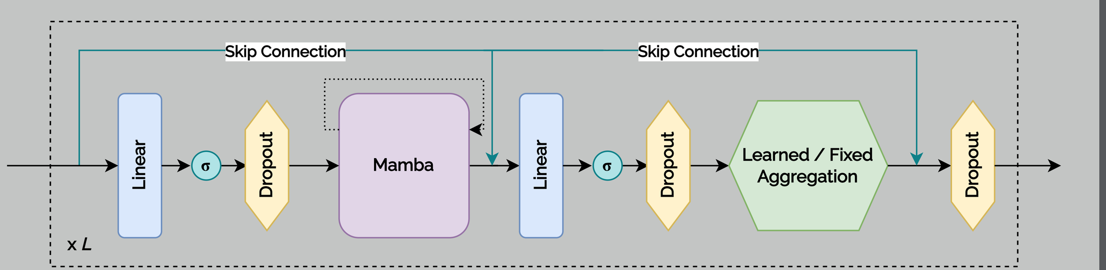
Our Method – Expressive Graph Mamba

- We redefine the UPDATE step for evolving graph representations over time (i.e., application of EGM blocks):
- EGM utilizes a **shared** Mamba block for node feature updates.
- AGGREGATE can be fixed or learned, e.g., by incorporating a self-attention mechanism:

$$x_i^{(t+1)} := y_i^{(t+1)} = \rho\left(\sum_{j \in \mathcal{N}[i]} \alpha_{ij} W^{(t+1)} y_j^{(t+1)}\right)$$

- The attention mechanism aggregates node features, while SSM handles node feature updates.
- Final graph representation is obtained after evolving for L time steps (i.e., passing through L EGM blocks).
- EGM blocks can be stacked to allow information flow between multi-hop neighborhoods, similar to related GNN architectures.
- We also extend the EGM architecture to dynamic spatiotemporal graphs by learning over edge-feature-weighted adjacency matrices.

EGM – Illustrated



- SiLU activation was used as the non-linearity in all instances.
- 1×1 `Conv` layers were used instead of `Linear` as fully-connected layers.
- Layer Normalization and Dropout generally aid training stability, but are optional.
- The residual connections are removed in the spatio-temporal case.

Experiment Details

- Utilized the *Planetoid* datasets for multi-class node classification on **static graphs**, comparing against MPNN variants (GCN, GAT/v2, GraphSAGE, GIN).
- Used `CrossEntropy` loss for node classification.
- Employed the *PEMS08* dataset for traffic forecasting on **dynamic graphs**, using `HuberLoss` as the loss function.
- Conducted experiments with 3-5 random seeds, tracking roughly 600 experiments on *WandB* (and more performed locally), totaling approximately 1,750 runs.
- Performed ablation experiments on the architecture, informing the suggestions in the previous block.

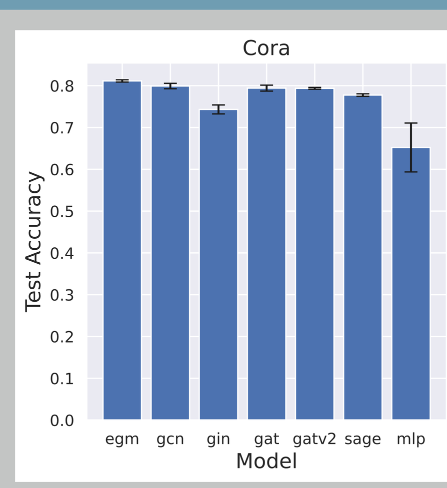
Table: Dataset Information

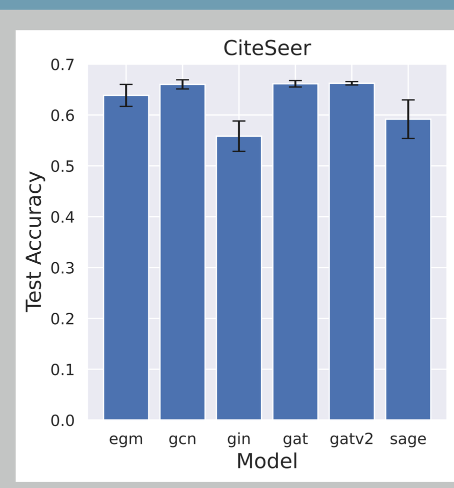
Name	#nodes	#edges	#features	#classes	Timesteps
Cora	2,708	10,556	1,433	7	-
CiteSeer	3,327	9,104	3,703	6	-
PubMed	19,717	88,648	500	3	-
PEMS08	170	-	-	-	17,856

Table: Training Information. (*) indicates the best-performing component.

Loss Function	`CrossEntropy`, `HuberLoss`
Scheduler	`ExponentialLR` (*), `LinearLR`, `LambdaLR`
Optimizer	`RAdam` (*), `Adam` (*), `NAdam`, `AdamW`
Weight Decay	0.0005
Learning Rate	[0.0001, 0.01]
(ST)EGM Layers	{1, 2, 3, 4, 6, 8}
Epochs	[50, 500]

Results





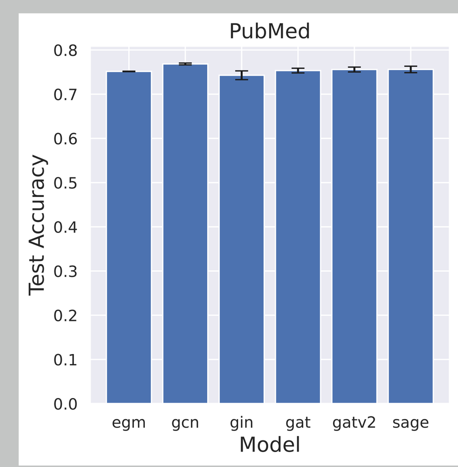
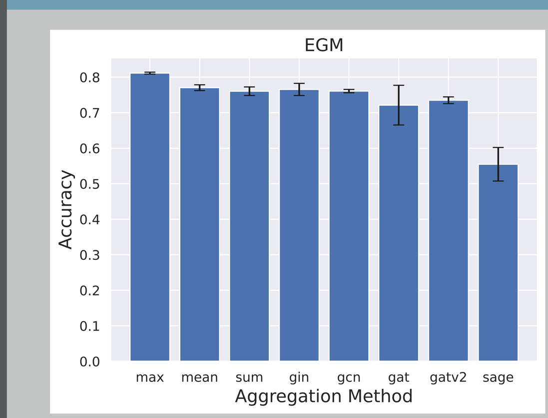
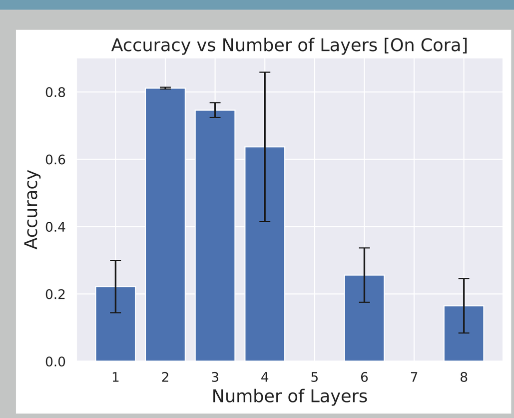


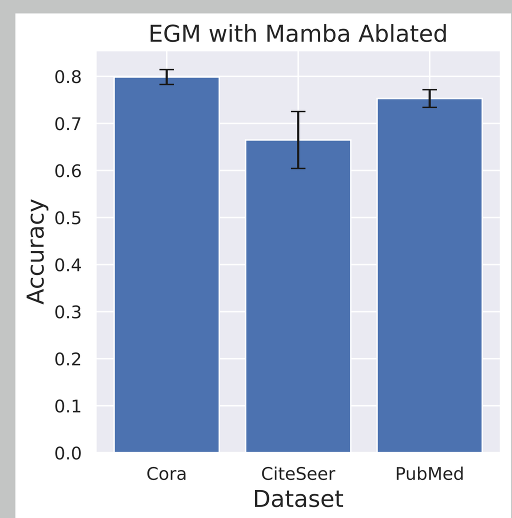
Table: Performance on PEMS08 (Baselines reported via STAEformer)

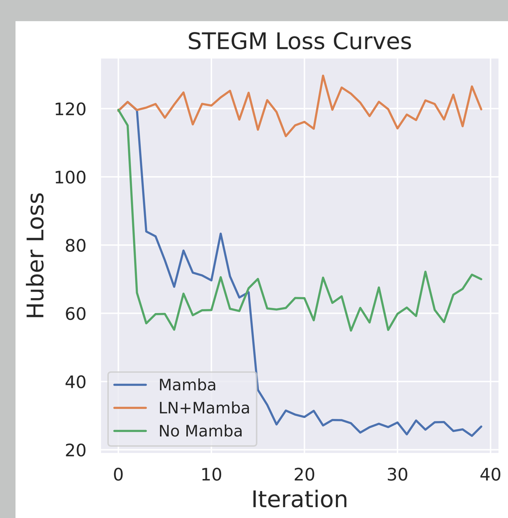
Metric	MAE	RMSE	MAPE
GWNet	14.40	23.39	9.21%
DCRNN	15.22	24.17	10.21%
STGCN	16.08	25.39	10.60%
STAEformer	**13.46**	**23.25**	**8.88%**
STEGM (**Untuned**)	21.44	33.09	13.62%

Model Ablation









Discussion & Limitations

- (ST)EGM performs comparably with the baselines across all tested datasets.
- Fixed aggregation functions like `mean`, `sum`, and `max` surprisingly yield superior results compared to learned aggregation, indicating unsuitability of the GAT-based method for EGM due to potential over-smoothing.
- EGM's accuracy decreases notably with increasing layers, possibly due to over-smoothing and instability in the Mamba block.
- Ablative experiments reveal that removing the Mamba block for static graphs **does not significantly affect overall performance**, while for dynamic graphs, the **model struggles without the Mamba block**.
- Mamba's effectiveness is hindered by the contradiction between node states treated as individual sequences (i.e., sequence length = 1) and Mamba's expectation of long sequences.
- Alleviating this bottleneck for static graphs may require transitioning to a node sequence approach, but this violates the graph inductive bias, whereas dynamic graphs are more naturally suited to such an approach.
- The potential of more sophisticated normalization schemes, such as Spectral Normalization, was explored to stabilize training by rescaling the impact of large activations on weight updates, but no improvement was observed in the model performance.

Planned Work

- Experiments are being set up with deeper models on benchmark datasets, e.g., the LRGB and TGB datasets to holistically evaluate scalability and adaptability of our (ST)EGM architecture.
- Further look into the model's performance in terms of FLOPs.
- Hyperparameter search to further boost model performance.

References

- **Please check the project listing on the CS460 website (`https://shorturl.at/dghOY`) for the complete bibliography.**
- `https://gitlab.niser.ac.in/JeS/mambagnn/` will host the codebase in the near future.