# Applying Mamba to GNNs

CS660: Machine Learning

Jyotirmaya Shivottam

23226001

School of Computer Sciences

March 13, 2024

# Recap

## Goals

- To investigate whether Structured Space Models (SSMs), specifically Mamba S6[#1], can be applied to state updates in Graph Neural Networks (GNNs)
- To implement such a graph-based model
- To benchmark against existing graph networks on a wide variety of tasks

## Existing GNN Baselines

- 2015: Gated Graph Sequence Neural Networks (GGSNN)
- 2016: Graph Convolutional Networks (GCN)
- 2017: Graph Attention Networks (GATN)
- 2018: Graph Isomorphism Networks (GIN)
- 2021: (Dynamic) Graph Echo State Networks (GESN)

# Primer: Graph Neural Networks (GNNs)

- General GNNs (Message Passing) update the state (or embedding) of each node in a graph, $\mathcal{G}(\mathcal{V}, \mathcal{E})$, based on the states of its neighbors.
- They can be characterized by the following equations:

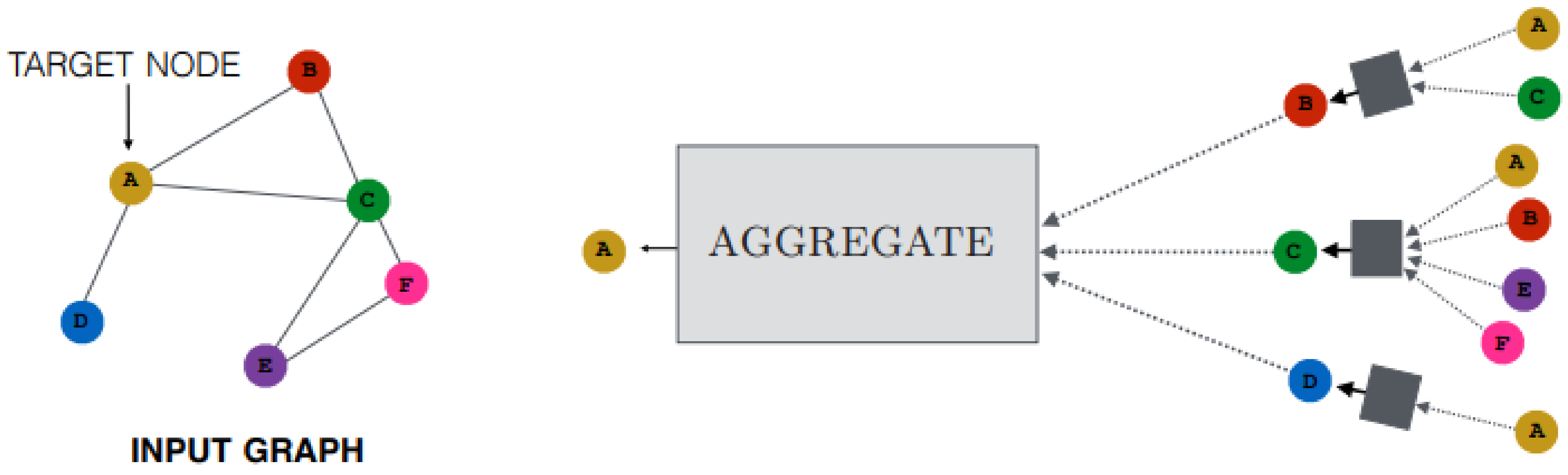
$$
\begin{aligned}
h_v^{(t)} &= \mathrm{UPDATE}(h_v^{(t-1)}, \mathrm{AGGREGATE}(\{h_u^{(t-1)} : u \in \mathcal{N}(v)\})) \\
h_v^{(t)} &= \mathrm{READOUT}(\{h_v^{(t)} : v \in \mathcal{V}\})
\end{aligned}
$$

- Note that the AGGREGATE and UPDATE functions need to be at least permutation-equivariant.

## Tasks

- Node property prediction, e.g., node classification.
- Edge property prediction, e.g., link existence.
- Graph representation learning (Global graph-level)

# Primer: Graph Neural Networks (GNNs)




Source: Scarselli, F.; The Graph Neural Network Model

# Mamba / S6 (Dec, 2023)

- A state space model (SSM) maps an input signal, $x(t)$, to an output signal, $y(t)$, through a set of **hidden state variables,** $h(t)$, like so:

$$\dot{h}(t) = Ah(t) + Bx(t) \leftarrow \text{First order DE}$$
$$y(t) = Ch(t) + Dx(t)$$

- Here, $A$, $B$, $C$, and $D$ are the state **transition**, input, output, and feed-through matrices, respectively. Note: $A, B, C, D$ are **time-independent**.
- **Goal**: To find (or learn) a mapping (model) from a **long** sequence of input data to a sequence of output data.
- **S**elective **S**tructured **S**tate **S**pace (**S**equence) Model with Associative **S**can $\equiv$ SSSSSS (S6) $\equiv$ *Mamba* 🐍.
- Mamba relaxes the time-invariance criterion of S4 (an earlier SSM), thereby introducing input-dependence.

# Mamba / S6

**Algorithm 1** SSM (S4)

**Input:** $x : (\mathtt{B}, \mathtt{L}, \mathtt{D})$
**Output:** $y : (\mathtt{B}, \mathtt{L}, \mathtt{D})$
$\boldsymbol{A} : (\mathtt{D}, \mathtt{N}) \leftarrow \mathsf{Parameter}$
▷ Represents structured $N \times N$ matrix
$\boldsymbol{B} : (\mathtt{D}, \mathtt{N}) \leftarrow \mathsf{Parameter}$
$\boldsymbol{C} : (\mathtt{D}, \mathtt{N}) \leftarrow \mathsf{Parameter}$
$\Delta : (\mathtt{D}) \leftarrow \tau_\Delta(\mathsf{Parameter})$
$\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}} : (\mathtt{D}, \mathtt{N}) \leftarrow \mathsf{discretize}(\Delta, \boldsymbol{A}, \boldsymbol{B})$
$y \leftarrow \mathsf{SSM}(\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}}, \boldsymbol{C})(x)$
▷ Time-invariant: recurrence or convolution
**return** $y$

$$h_t = \overline{A}h_{t-1} + \overline{B}x_t$$
$$y_t = Ch_t$$

The $\bar{A}$ matrix also depends on the input, through $\Delta$

**Algorithm 2** SSM + Selection (S6)

**Input:** $x : (\mathtt{B}, \mathtt{L}, \mathtt{D})$
**Output:** $y : (\mathtt{B}, \mathtt{L}, \mathtt{D})$
$\boldsymbol{A} : (\mathtt{D}, \mathtt{N}) \leftarrow \mathsf{Parameter}$
▷ Represents structured $N \times N$ matrix
$\boldsymbol{B} : (\mathtt{B}, \mathtt{L}, \mathtt{N}) \leftarrow s_B(x)$
$\boldsymbol{C} : (\mathtt{B}, \mathtt{L}, \mathtt{N}) \leftarrow s_C(x)$
$\Delta : (\mathtt{B}, \mathtt{L}, \mathtt{D}) \leftarrow \tau_\Delta(\mathsf{Parameter} + s_\Delta(x))$
$\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}} : (\mathtt{B}, \mathtt{L}, \mathtt{D}, \mathtt{N}) \leftarrow \mathsf{discretize}(\Delta, \boldsymbol{A}, \boldsymbol{B})$
$y \leftarrow \mathsf{SSM}(\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}}, \boldsymbol{C})(x)$
▷ Time-varying: recurrence (*scan*) only
**return** $y$

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

Source: Mamba (S6), Gu et al [1] & https://github.com/hkproj/mamba-notes

# Mamba / S6

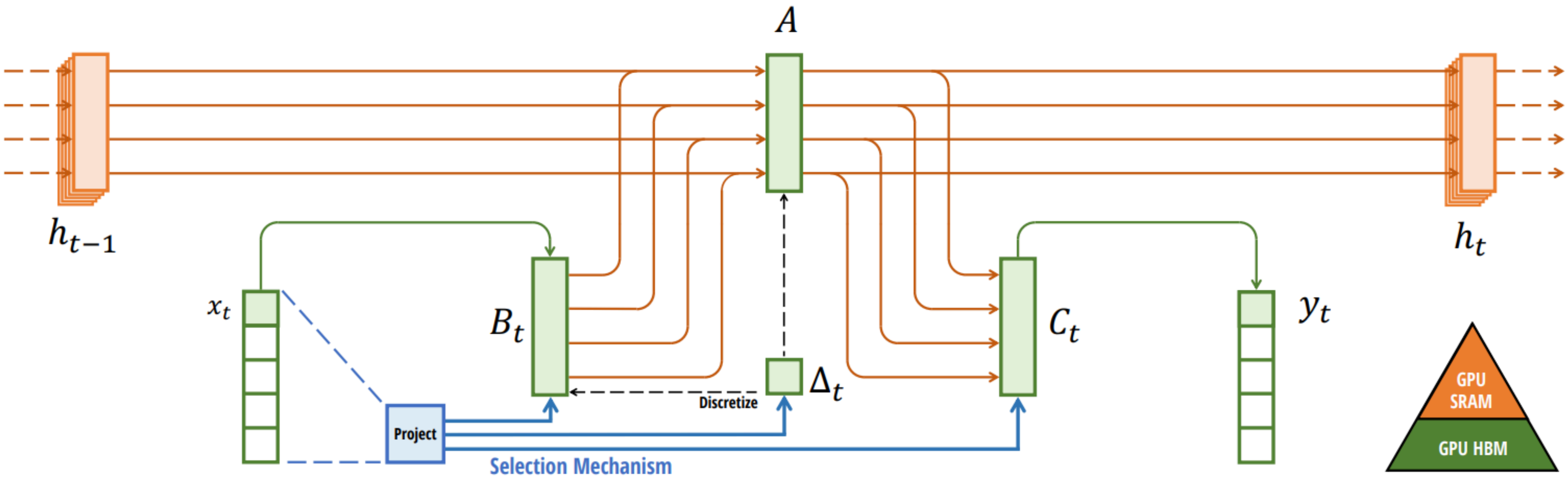



Figure 1: (**Overview**.) Structured SSMs independently map each channel (e.g. $D = 5$) of an input $x$ to output $y$ through a higher dimensional latent state $h$ (e.g. $N = 4$). Prior SSMs avoid materializing this large effective state ($DN$, times batch size $B$ and sequence length $L$) through clever alternate computation paths requiring time-invariance: the $(\Delta, \boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C})$ 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.

Source: Mamba (S6), Gu et al [1]

# Our approach

- We propose to <u>treat the (hidden) state update of each node as a sequence</u>, and apply Mamba to model the sequence.
- Then, we aggregate on the **neighbor states**, and update each node state using learned aggregation (attention) weights, or a fixed aggregation function, e.g., mean, max, sum, etc.

$$x_i^{(t+1)} = C(\bar{A}h_i^t + \bar{B}x_i^t) \quad \longleftarrow \text{UPDATE}$$

$$x_i^{(t+1)} = \rho\left(\sum_{j\in\mathcal{N}(j)} \alpha_{ij} W^{(t+1)} y_j^{(t+1)}\right), \quad \longleftarrow \text{AGGREGATE (Attention)}$$

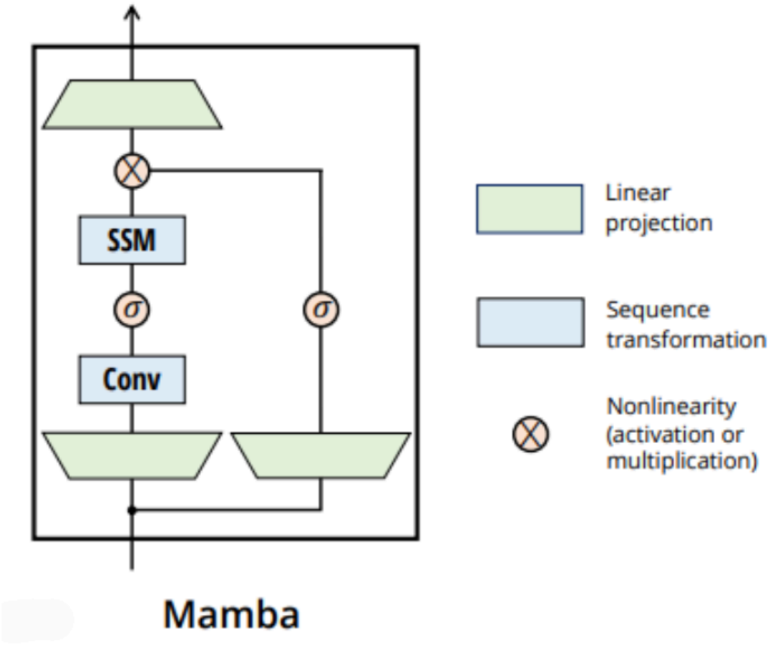
- The AGGREGATE function can be learned or fixed. The UPDATE is learned in the usual Mamba way.
- This is similar to (in fact, a **linearized** version of) Gated Graph Sequence Neural Networks (GGSNN).

# Results so far

- *Datasets*: Planetoid datasets: Cora, Pubmed, and CiteSeer. These datasets comprise citation networks in various domains.
- *Task*: Node classification.
- The Mamba block was taken from the mamba-ssm Python library by the original authors. The rest of the model was implemented from scratch in PyTorch. We used skip connections and SiLU activations.
- We used dropout, a decaying learning rate ($1e^{-4}$ - $1e^{-2}$) with schedule, RAdam optimizer, and maintained similar structure for all the models. The parameter budget was **unconstrained**.
- We ran each model with 3 - 5 different seeds and computed the mean and standard deviation of the test accuracy. In total, 1_439 // 4(?) runs were conducted and tracked on WandB. **No hyperparameter tuning was done.**
- We found that our model performs comparably to the baselines (+ GATv2 + MLP).
- *However, there was an interesting observation*.

# Model architecture

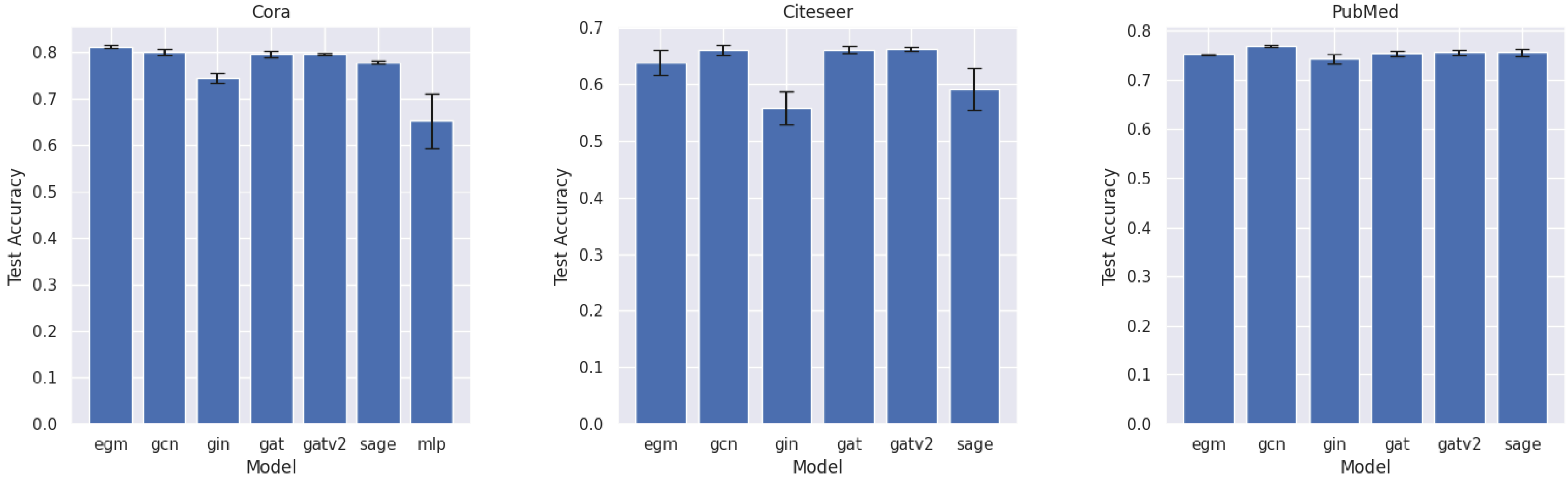
Linear



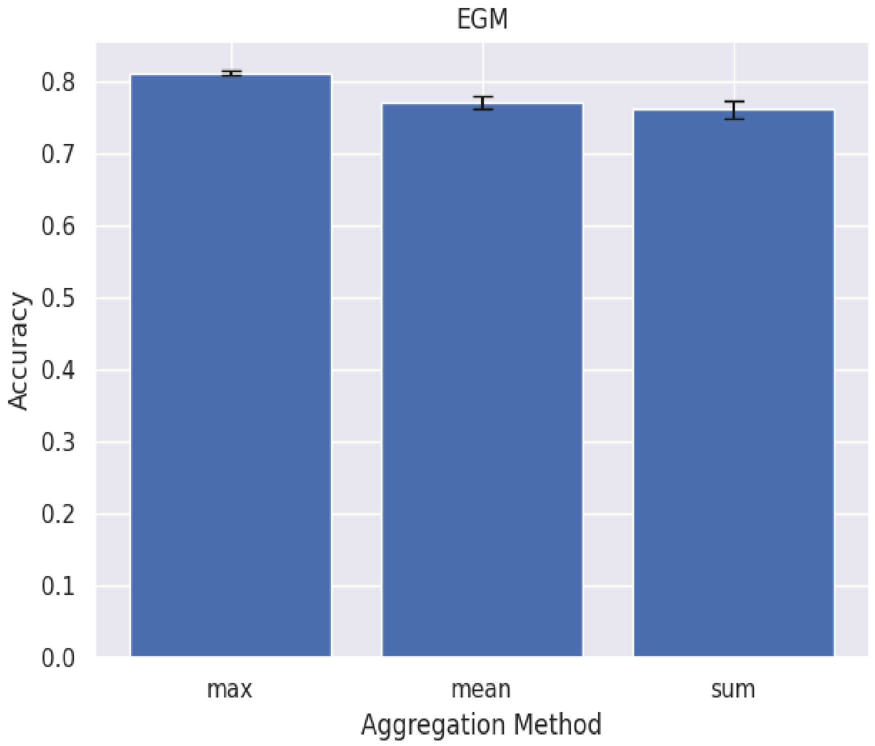

Source: Mamba (S6), Gu et al [1]

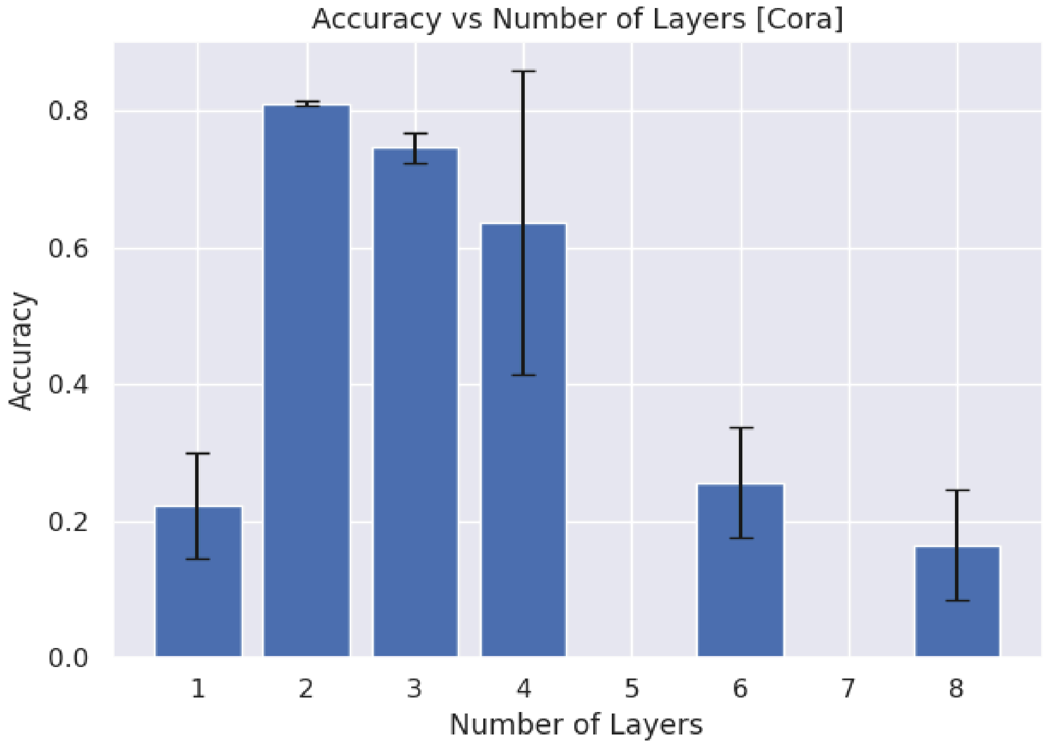
Linear

# Results so far: General Comparison



# Results so far: Experiments



# Current limitations

- Unstable training: Without normalization & skip connections, the training is stochastic.
- Fixed aggr > Learned aggr: We found that using a fixed aggregation function (e.g., mean, max, sum) works better. This might be due to *oversmoothing*.
- Shallow models: Increasing the number of layers beyond 4 tanks performance. *cf. Graph Transformer Networks*.
- Ablation: If we remove the Mamba blocks, we *nearly* recover the GIN update, with *nearly identical performance*.

## Why? Some probable reasons:

- Since Mamba is a seq2seq model, it does not seem very useful for static graphs.
- In particular, as we are treating each node as having its own sequence of states, the sequence length becomes just 1, making it a bottleneck.

# Remaining Work

- We are currently experimenting with other larger datasets, that are known to have long-range dependencies between nodes.
- We will also try mixing BatchNorm and LayerNorm to further stabilize training and to test if it enables deeper models.
- We are yet to test with dynamic graphs, i.e., graphs that change over time (changing node / edge embeddings or new nodes / edges, etc.), e.g., multi-sensor data, spatio-temporal graphs, etc. We expect that in this case, Mamba should have an impact, as the bottleneck is lifted.

## References


0. Other GNN approaches listed before.
1. 2023: Mamba: Linear-Time Sequence Modeling with Selective State Spaces
2. 2020: HiPPO: Recurrent Memory with Optimal Polynomial Projections