A New Perspective for Graph Learning Architecture Design: Linearize Your Depth Away

Joël Mathys
ETH Zurich
jmathys@ethz.ch

Roger Wattenhofer
ETH Zurich
wattenhofer@ethz.ch

Abstract

Designing effective graph learning architectures is central to making learning on structured and relational data feasible and scalable. Crucially, such designs must incorporate sufficient inductive bias to capture and leverage the graph topology. Simultaneously, they have to balance this objective with efficiently utilizing modern hardware, and remaining effectively trainable, even at scale. The current proposed architectures and paradigms range from message-passing neural networks on the graph topology to graph-informed transformers and virtual compute structures. Especially, the latter techniques often translate useful concepts and insights from graph theory in order to improve stability, mixing time, or bottlenecks. In this work, we highlight a linearization technique from the recently proposed Graph State-Space Model, as a powerful, general tool to design or improve graph learning architectures. At its core, the technique simplifies and reduces sequential computational depth and improves execution speed, while largely preserving trainability. Furthermore, the tool is versatile enough to be applied as a drop-in module across existing architectures. We showcase this flexibility by adapting Cayley Graph Propagation, yielding a simple, deeper and faster architecture.

1 Introduction

Designing effective graph learning architectures is central to making learning on structured and relational data both feasible and scalable. Graph-structured data appear in a wide range of domains, from molecular structures [Gilmer et al., 2017], social networks [Kipf and Welling, 2017], road networks [Neun et al., 2022] to combinatorial problems [Cappart et al., 2021]. The core challenge lies in developing architectures that incorporate sufficient graph-aware inductive biases to leverage the structural information while also being able to capture sufficient domain-specific knowledge in a data-driven manner, all while remaining hardware-efficient and trainable at scale.

As a consequence, this has led to numerous proposed graph learning architectures, ranging from classical message-passing neural networks [Gilmer et al., 2017] to graph transformers [Dwivedi and Bresson, 2021], walk-based approaches [Chen et al., 2025] or virtual compute structures [Wilson et al., 2024, Arnaiz-Rodriguez et al., 2022]. In particular, the latter category often aims to translate insights from traditional graph theory, such as spectral properties [Martinkus et al., 2022], expander constructions [Deac et al., 2022], and hierarchical decompositions [Grötschla et al., 2024] to combat common issues in graph learning, such as oversmoothing and bottlenecks [Arnaiz-Rodriguez and Errica, 2025, Alon and Yahav, 2021]. However, for many of these theoretical guarantees to be translated, the architectures have to rely on sequential propagation steps to achieve the necessary information mixing, ultimately leading to increased sequential depth. This creates an inherent tension between the depth needed for sufficient expressivity and balancing training stability and computational efficiency.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: New Perspectives in Graph Machine Learning.

In this work, we demonstrate that the linearization technique recently proposed by Graph State-Space Models (GSSM) [Ceni et al., 2025] can serve as a flexible design tool for general graph architectures. At its core, linearization can transform sequential computation steps into efficient closed-form parallel computations, directly impacting and reducing the sequential depth while maintaining training stability. Rather than proposing a specialized architecture, we show how this technique can function as a drop-in enhancement for future and already existing graph learning architectures with minimal code modifications.

We validate our approach by proposing a new modification of Cayley Graph Propagation (CGP) [Wilson et al., 2024]. Our adaptation demonstrates several benefits, because there is no task-specific preprocessing, we can properly leverage precomputation for computational efficiency. Further, the linearization allows us to further facilitate the message exchange intended by the expander properties of the Cayley graph [Cayley, 1878]. Finally, the closed-form parallel computation significantly reduces runtime overhead. We empirically validate our findings and demonstrate both the performance benefits and the computational efficiency gains of the linearized approach. Our work establishes the feasibility of applying linearization techniques to general graph architectures, providing practitioners with a valuable tool for improving computational efficiency without complete architectural overhauls. This positions linearization as a reusable design primitive that can advance the development of more effective graph learning systems.

2 Related Work

2.1 State-Space Models

Recent breakthroughs in sequence modeling, from S4 [Gu et al., 2022] to Mamba [Gu and Dao, 2024], demonstrate effectiveness in capturing long-range dependencies through linear parallel formulations. Behrouz and Hashemi [2024], Chen et al. [2025] adapt this approach by converting graphs to sequences via walk-based methods. In contrast, Ceni et al. [2025] formulate an SSM variant closer to graph convolutions with a focus on static and temporal graph data. In our work, we highlight the general applicability of the underlying linearization technique as a design tool.

2.2 Linearization

Linearization aims to remove non-linearities in the computation flow. On one hand, this can lead to more efficient closed-form formulations, and on the other it simplifies the underlying mechanisms enough to approach them with rigorous mathematical analysis [Keriven, 2022]. An other line of work considers convex relaxations in favor of optimization guarantees, partly achieved through linearization[Cohen and Agmon, 2021]. The work of Wu et al. [2019] removed non-linearities, creating competitive linear models with computational savings. However, this was mainly considered as a standalone architecture rather than a design component, potentially limiting opportunities for stackable components that could increase performance and make it more generally applicable.

2.3 Depth

The relationship between depth and width plays a special role in the graph community. Loukas [2020] establishes theoretical lower bounds showing that depth is crucial, especially when the width is small compared to overall graph size. Further, in classical graph convolutions depth is often the main driver for receptive field expansion [Liu et al., 2020]. However, this is not the only role of depth in the computation [Telgarsky, 2016], it also needs to properly process and transform information. The fundamental challenge is that non-linearities at depth impose significant costs while depth remains theoretically necessary. The GSSM linearization technique offers one possible way to incorporate this as a design tool while balancing it with a convolutional graph bias.

3 Background

3.1 Graph State-Space Models

GSSM introduced by Ceni et al. [2025] draw inspiration from recent advances in state-space models for sequence modeling. These have demonstrated their effectiveness in capturing long-range de-

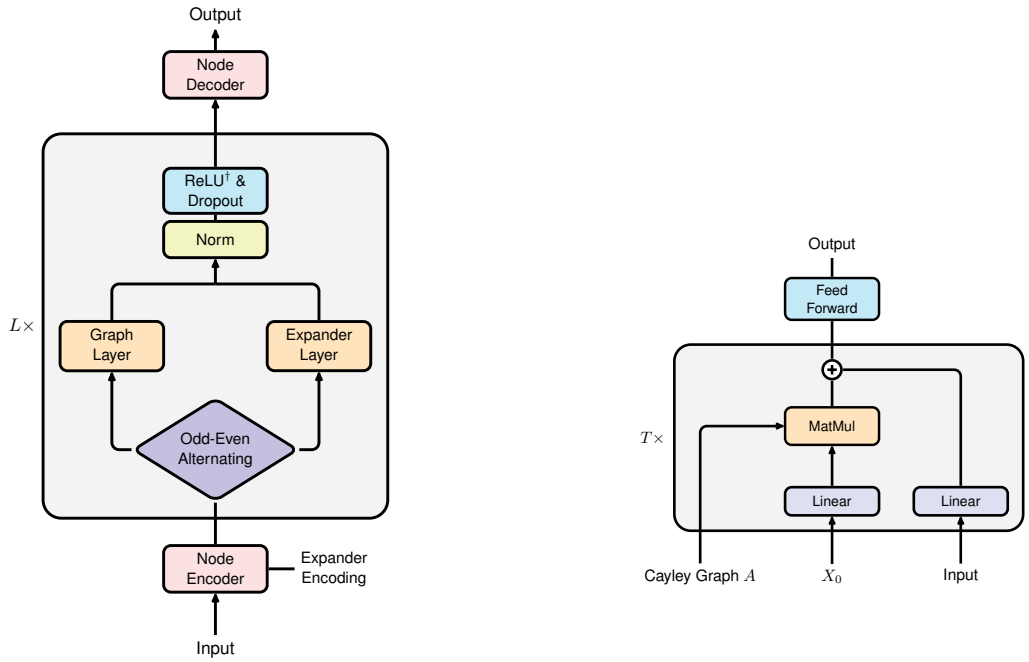

Figure 1: Architectural comparison between original CGP and proposed CGP-SSM. (Left) Standard CGP alternates between blocks of a single graph convolution on the original and expander topology. (Right) Our CGP-SSM replaces the single expander layer with T linearized convolutions. This aims to reduce sequential depth while mixing more information in the process.

pendencies while maintaining computational efficiency through parallel formulations. The GSSM framework extends this idea to graph-structured data by introducing modular linearized computation blocks of depth k with trainable parameters $\mathbf{W}$ and $\mathbf{B}$, adjacency matrix $\mathbf{A}$ and input $\mathbf{U}$.

$$\mathbf{X}_t = \mathbf{A}\mathbf{X}_{t-1}\mathbf{W} + \mathbf{U}_t\mathbf{B}$$
$$\mathbf{Y} = \mathbf{MLP}(\mathbf{X}_k)$$

The key lies in the linearization technique that yields a fast closed-form computation which can leverage parallelism. This reduces sequential computational depth while minimally impacting trainability and stability within a block.

3.2 Cayley Graph Propagation

CGP belongs to a family of architectures that take advantage of virtual compute structures designed to overcome limitations in message-passing neural networks [Deac et al., 2022, Arnaiz-Rodriguez et al., 2022]. CGP leverages the mathematical properties of Cayley graphs, specifically their role as expander graphs of size n with guaranteed sparse connectivity, low $\mathcal{O}(\log n)$ diameter, and rapid mixing properties without structural bottlenecks. The architecture alternates between standard graph convolutions on the original topology and propagation on a virtual Cayley graph structure derived from the group $SL(k)$. Critically, this approach requires *no task-specific preprocessing*, making it broadly applicable.

4 Methodology

The linearization technique used in GSSM represents a more general design primitive well suited to improve existing graph architectures rather than limiting its use as a standalone model. The main advantage of the technique is to reduce the sequential computational depth through a closed-form parallel formulation. This is especially evident in classical message-passing architectures which require depth to expand their receptive fields, yet deep graph networks suffer from training instability,

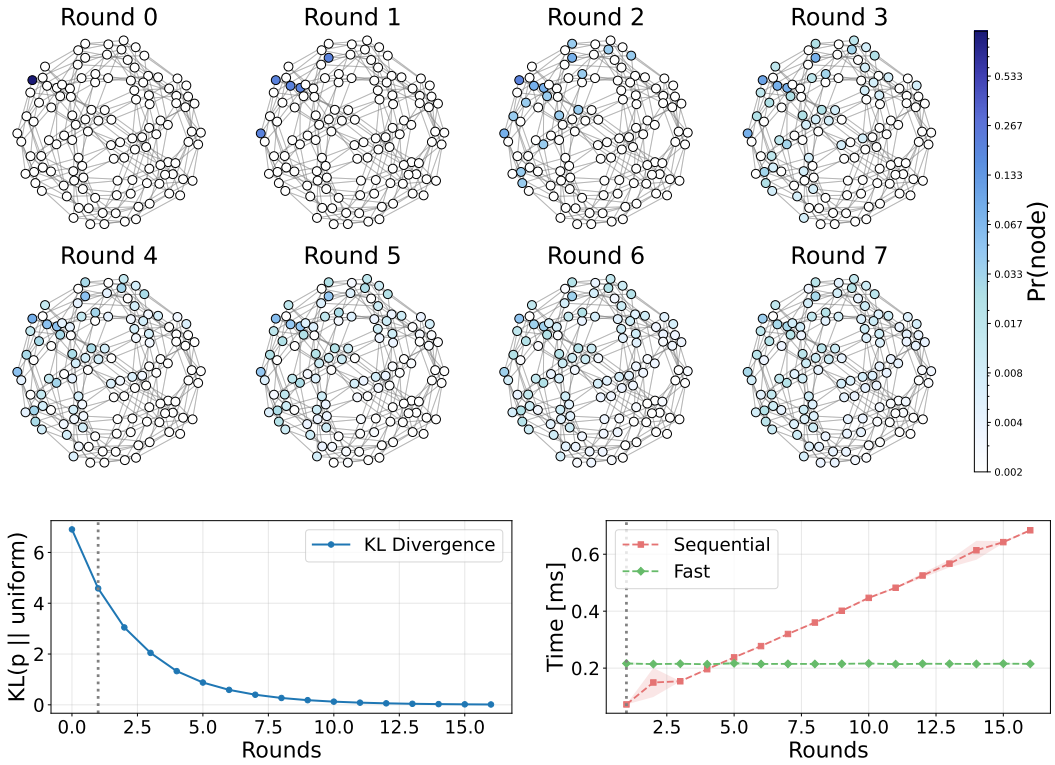

Figure 2: Visualization of the information diffusion and computational efficiency on Cayley graph SL(5). (Top) Visualization of probability mass diffusion from a single source node across k rounds of propagation according to a random walk (with self-loops) of k steps. Due to the expander graph properties the mixing should converge withing $\mathcal{O}(\log n)$ steps. (Bottom Left) KL divergence to the uniform distribution decreases as information spreads, illustrating the expander graph's fast mixing. (Bottom Right) Runtime comparison between sequential computation (red) and closed-form computation (green). The grey dotted line indicates the effect of a single-step baseline used in CGP.

oversmoothing or oversquashing [Arnaiz-Rodriguez and Errica, 2025]. Moreover, the technique is general enough to function as a drop-in replacement within existing architectural designs. We demonstrate this versatility by adapting Cayley Graph Propagation as our primary case study.

Specifically, our approach modifies the expander propagation component within CGP while leaving all other architectural choices unchanged. In standard CGP, layers alternate between message passing on the original graph topology and propagation on the virtual Cayley expander graph. We replace the *single-step* expander convolution with a linearized *multiple-step* formulation of depth T. A visualization of the architecture is shown in Figure 1. Crucially, since Cayley expander graphs do not require task-specific preprocessing, we can precompute the required decomposition offline, making the closed-form linearization even more efficient.

This modification alters the information exchange dynamics within the architecture. The original CGP alternates between topologies at each layer, equally balancing the two by executing a single convolution. Our reformulation uses multiple convolutions on the expander graph instead. Since Cayley expander graphs require $\mathcal{O}(\log n)$ steps to uniformly mix information between nodes, the original standard formulation does not fully exploit these properties.

By increasing the number of convolutions beyond a single step, we aim to enable more thorough information exchange. At the same time, the closed-form formulation adds minimal overhead to the computation, eliminating a potential sequential bottleneck. To further illustrate this process, we visualize the information diffusion process on a Cayley graph in Figure 2. The probability mass spreads following a random walk (with self-loops) of length k from an initial node across the expander structure over multiple steps.

Table 1: Empirical results on the MUTAG dataset (accuracy). Both CGP and CGP-SSM use the architecture by Wilson et al. [2024] with the GCN convolution as a base, where CGP-SSM replaces the expander propagation with the GSSM linearization of various unrolling depths. We report the 95% confidence intervals computed over 50 seeds.

Model	CGP	Depth	MUTAG ↑
GCN			73.60 ± 3.15
CGP	✓		77.60 ± 3.17
CGP-SSM	✓	3	79.90 ± 2.61
CGP-SSM	✓	6	79.00 ± 2.62
CGP-SSM	✓	9	79.80 ± 2.58
CGP-SSM	✓	12	**80.90** ± 2.43
CGP-SSM	✓	15	79.70 ± 2.53
CGP-SSM	✓	100	79.90 ± 2.65

To further validate this approach empirically, we ablate our modified architecture on the MUTAG dataset [Morris et al., 2020], where the original CGP improved the most over the baseline GCN method. Table 1 shows that the linearized multi-step expander propagation consistently improves the original CGP formulation across different unrolling depths. For more details on the experimental evaluation we refer to the Appendix.

In the case of CGP the linearization technique emphasizes improved information mixing without the drawback of additional sequential compute depth, all while leveraging hardware-efficient computation. However, the computational speedup comes at the cost of increased memory requirements for using decomposed matrices ultimately needed for faster computation. Looking ahead, this linearization paradigm might open up several new research directions. Recent breakthroughs in advanced state-space model techniques ranging from S5 [Smith et al., 2023] to Mamba [Gu and Dao, 2024], relied on clever initialization, hardware-aware implementations and selective mechanisms, yet to fully translate to the graph domain. Finally, for graph practitioners, this raises intriguing possibilities about the future role of virtual graph structures: effectively balancing computational depth, training stability, and theoretical guarantees under computational efficiency.

5 Conclusion

We demonstrate that the linearization technique of Graph State-Space Models serves as a general design tool that extends well beyond the standalone architectures. The technique's main benefit is to effectively reduce sequential computational depth through parallel formulations, without the training instabilities typically associated with deep non-linear sequential operations. Our adaptation of Cayley Graph Propagation validates this approach in practice, showing that even current architectures can be improved with minimal overhead. In addition, we showcase how linearization can be a way to enable a more thorough mixing of information on expander graphs while maintaining computational efficiency. This demonstrates that the linearization tool is an interesting perspective for general graph learning architecture designs towards developing more capable, efficient, and scalable graph learning.

References

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1263–1272. PMLR, 06–11 Aug 2017. URL `https://proceedings.mlr.press/v70/gilmer17a.html`.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2017.

Moritz Neun, Christian Eichenberger, Henry Martin, Markus Spanring, Rahul Siripurapu, Daniel Springer, Leyan Deng, Chenwang Wu, Defu Lian, Min Zhou, Martin Lumiste, Andrei Ilie, Xinhua

Wu, Cheng Lyu, Qing-Long Lu, Vishal Mahajan, Yichao Lu, Jiezhang Li, Junjun Li, Yue-Jiao Gong, Florian Grötschla, Joël Mathys, Ye Wei, He Haitao, Hui Fang, Kevin Malm, Fei Tang, Michael Kopp, David Kreil, and Sepp Hochreiter. Traffic4cast at neurips 2022 – predict dynamics along graph edges from sparse node data: Whole city traffic and eta from stationary vehicle detectors. In Marco Ciccone, Gustavo Stolovitzky, and Jacob Albrecht, editors, *Proceedings of the NeurIPS 2022 Competitions Track*, volume 220 of *Proceedings of Machine Learning Research*, pages 251–278. PMLR, 28 Nov–09 Dec 2022. URL https://proceedings.mlr.press/v220/neun23a.html.

Quentin Cappart, Didier Chételat, Elias B. Khalil, Andrea Lodi, Christopher Morris, and Petar Veličković. Combinatorial optimization and reasoning with graph neural networks. In Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 4348–4355. International Joint Conferences on Artificial Intelligence Organization, 8 2021. doi: 10.24963/ijcai.2021/595. URL https://doi.org/10.24963/ijcai.2021/595. Survey Track.

Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs, 2021. URL https://arxiv.org/abs/2012.09699.

Dexiong Chen, Till Hendrik Schulz, and Karsten Borgwardt. Learning long range dependencies on graphs via random walks. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=kJ5H7oGT2M.

JJ Wilson, Maya Bechler-Speicher, and Petar Veličković. Cayley graph propagation. In *The Third Learning on Graphs Conference*, 2024. URL https://openreview.net/forum?id=VaTfEDs6lE.

Adrian Arnaiz-Rodriguez, Ahmed Begga, Francisco Escolano, and Nuria Oliver. Diffwire: Inductive graph rewiring via the lovász bound, 2022. URL https://arxiv.org/abs/2206.07369.

Karolis Martinkus, Andreas Loukas, Nathanaël Perraudin, and Roger Wattenhofer. Spectre: Spectral conditioning helps to overcome the expressivity limits of one-shot graph generators, 2022. URL https://arxiv.org/abs/2204.01613.

Andreea Deac, Marc Lackenby, and Petar Veličković. Expander graph propagation. In *The First Learning on Graphs Conference*, 2022. URL https://openreview.net/forum?id=IKevTLt3rT.

Florian Grötschla, Joël Mathys, Robert Veres, and Roger Wattenhofer. Core-GD: A hierarchical framework for scalable graph visualization with GNNs. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=vtyasLn4RM.

Adrian Arnaiz-Rodriguez and Federico Errica. Oversmoothing, oversquashing, heterophily, long-range, and more: Demystifying common beliefs in graph machine learning, 2025. URL https://arxiv.org/abs/2505.15547.

Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications, 2021. URL https://arxiv.org/abs/2006.05205.

Andrea Ceni, Alessio Gravina, Claudio Gallicchio, Davide Bacciu, Carola-Bibiane Schonlieb, and Moshe Eliasof. Message-passing state-space models: Improving graph learning with modern sequence modeling, 2025. URL https://arxiv.org/abs/2505.18728.

Professor Cayley. Desiderata and suggestions: No. 2. the theory of groups: Graphical representation. *American Journal of Mathematics*, 1(2):174–176, 1878. ISSN 00029327, 10806377. URL http://www.jstor.org/stable/2369306.

Albert Gu, Karan Goel, and Christopher Re. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=uYLFoz1vlAC.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=tEYskw1VY2.

Ali Behrouz and Farnoosh Hashemi. Graph mamba: Towards learning on graphs with state space models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '24, page 119–130, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400704901. doi: 10.1145/3637528.3672044. URL https://doi.org/10.1145/3637528.3672044.

Nicolas Keriven. Not too little, not too much: a theoretical analysis of graph (over)smoothing. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Saar Cohen and Noa Agmon. Convexified graph neural networks for distributed control in robotic swarms. In Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 2307–2313. International Joint Conferences on Artificial Intelligence Organization, 8 2021. doi: 10.24963/ijcai.2021/318. URL https://doi.org/10.24963/ijcai.2021/318. Main Track.

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6861–6871. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/wu19e.html.

Andreas Loukas. What graph neural networks cannot learn: depth vs width. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=B1l2bp4YwS.

Meng Liu, Hongyang Gao, and Shuiwang Ji. Towards deeper graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 338–348, 2020.

Matus Telgarsky. benefits of depth in neural networks. In Vitaly Feldman, Alexander Rakhlin, and Ohad Shamir, editors, *29th Annual Conference on Learning Theory*, volume 49 of *Proceedings of Machine Learning Research*, pages 1517–1539, Columbia University, New York, New York, USA, 23–26 Jun 2016. PMLR. URL https://proceedings.mlr.press/v49/telgarsky16.html.

Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*, 2020. URL www.graphlearning.io.

Jimmy T.H. Smith, Andrew Warrington, and Scott Linderman. Simplified state space layers for sequence modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=Ai8Hw3AXqks.

PyG Team. torch_geometric.datasets.tudataset — pytorch geometric documentation. https://pytorch-geometric.readthedocs.io/en/latest/generated/torch_geometric.datasets.TUDataset.html, 2025. Accessed: 2025-10-08.

Kedar Karhadkar, Pradeep Kr. Banerjee, and Guido Montúfar. Fosr: First-order spectral rewiring for addressing oversquashing in gnns, 2023. URL https://arxiv.org/abs/2210.11790.

Corinna Coupette, Jeremy Wayland, Emily Simons, and Bastian Rieck. No metric to rule them all: Toward principled evaluations of graph-learning datasets. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=XbmBNwrfG5.

A Experimental Evaluation

Architecture and Setup: We used the original CGP implementation [Wilson et al., 2024] as our baseline, preserving all hyperparameters choices such as: early stopping with a patience of 100 epochs, trained for maximum 300 epochs, 4 layers, hidden dimension of 64, 50% dropout, Batchnorm. Our only modification for CGP-SSM replaced the single expander convolution with the linearized

multi-step formulation. In addition, we initialized the W weight matrices with a spectral radius close to 1.0 following standard practices for numerical stability. This provided consistent improvements to CGP-SSM performance, although it was not decisive to improve wrt. to the CGP baseline. All MUTAG results are averaged over 50 independent runs and we report confidence intervals at 95%. The MUTAG dataset contains roughly 200 graphs of average size 18, with 7 features and is a binary classification task PyG Team [2025], reported performance considers accuracy. The data is loaded and processed according to the CGP codebase, which follows Karhadkar et al. [2023] and splits the data into 80/10/10. Note that our reported results are lower than the numbers in the original CGP paper due to correcting a model selection procedure in the training loop that might implicitly leak information about the test set. Further, we want to note that many of the TU datasets are likely no longer ideal for evaluating modern learning approaches, as outlined in [Coupette et al., 2025]. Our primary reason for ablating on MUTAG, besides it being the dataset where CGP improved the most wrt. to its base, is to remain as close as possible to the original CGP setup to illustrate the benefits of the linearization technique.

Timing Experiments: Runtime measurements were conducted on an A600 GPU. Each measurement consisted of 100 sequential graph inputs, repeated 10 times to compute standard deviations. We precomputed all components that were independent of the unrolling steps and input features for both variants.

Memory Measurements: The improved runtime comes at the cost of an increased memory requirement. In the current implementation the A matrix is explicitly constructed (in order to find the decomposition). While this could likely be optimised, it requires the storage of $n \times n$ matrices. Further, if the number of rounds T is much higher than the number of nodes an additional $T \times n$ matrix is needed for the efficient computation. We report the maximum memory usage during inference forward passes for the implementations we describe in 1 on a single cayley graph (repeated 3 times) in Figure 3.

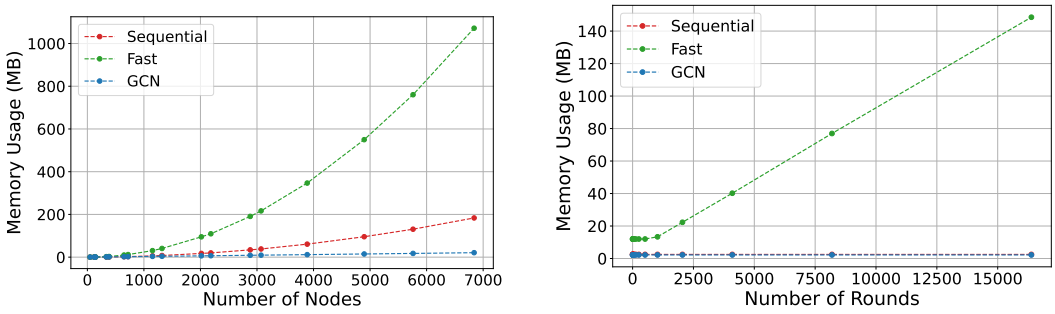

Figure 3: Overview of memory usage of different implementations. (Left) For a fixed number of rounds (T=10), we measure the memory usage of cayleigh graphs up to SL(20). The main difference to the GCN implementation is that Sequential and Fast realize the A $n \times n$ matrix explicitly, whereas GCN uses the sparse pyg representation and only stores e edges. (Right) For a fixed cayleigh graph SL(10) with 720 nodes. While both the GCN and the Sequential implementation have no extra memory requirement if the depth is increased, the Fast implementation realises a $T \times n$ matrix for faster computation. We note that this usually is negligible compared to the $n \times n$ A matrix and only dominates once $T \gg N$.

B Cayley Graphs

We include general statistics about the Cayley graphs used in our experiments in Table 2 and further visualizations for sizes 3,4,6,7 and 14.

Table 2: Overview over Cayley graphs generated by the group SL. The statistics were computed with the help of the networkx library.

SL(k)	Nodes	Edges	Avg. Degree	Diameter	Avg. Shortest Path Length	$\log_2 N$
SL(2)	6	24	2.00	3	1.80	2.58
SL(3)	24	96	4.00	4	2.35	4.58
SL(4)	48	192	4.00	6	3.11	5.58
SL(5)	120	480	4.00	6	3.75	6.91
SL(6)	144	576	4.00	6	3.99	7.17
SL(7)	336	1344	4.00	7	4.85	8.39
SL(8)	384	1536	4.00	8	5.02	8.58
SL(9)	648	2592	4.00	8	5.70	9.34
SL(10)	720	2880	4.00	10	5.97	9.49
SL(11)	1320	5280	4.00	10	6.60	10.37
SL(12)	1152	4608	4.00	10	6.47	10.17
SL(13)	2184	8736	4.00	10	7.37	11.09
SL(14)	2016	8064	4.00	11	7.27	10.98
SL(15)	2880	11520	4.00	12	8.04	11.49
SL(16)	3072	12288	4.00	12	8.18	11.58
SL(17)	4896	19584	4.00	12	8.41	12.26
SL(18)	3888	15552	4.00	12	8.24	11.92
SL(19)	6840	27360	4.00	13	9.10	12.74
SL(20)	5760	23040	4.00	14	9.04	12.49

C Implementation

Following the notation and derivation from Ceni et al. [2025] we can write the GSSM in a blocks as follows with A as the degree normalized adjacency matrix $A = D^{-\frac{1}{2}}(A+I)D^{-\frac{1}{2}}$ with self-loops.

$$
\begin{aligned}
X_0 &= 0 \\
X_t &= AX_{t-1}W + UB \\
X_t &= \sum_{i=0}^{t} A^i UBW^i
\end{aligned}
$$

Further, if we assume that we can decompose $A = P\Lambda P^{-1}$ and $W = V\Sigma V^{-1}$ we can write out the closed form as follows where $*$ denotes the element-wise multiplication.

$$
\begin{aligned}
X_t &= P\left(\sum_{i=0}^{t-1} \Lambda^i P^{-1} UBV\Sigma^i\right) V^{-1} \\
X_t &= P\left(\sum_{i=0}^{t-1} \Lambda^i S\Sigma^i\right) V^{-1} \\
X_t &= P(S * \Lambda_{pow}^T \Sigma_{pow}) V^{-1}
\end{aligned}
$$

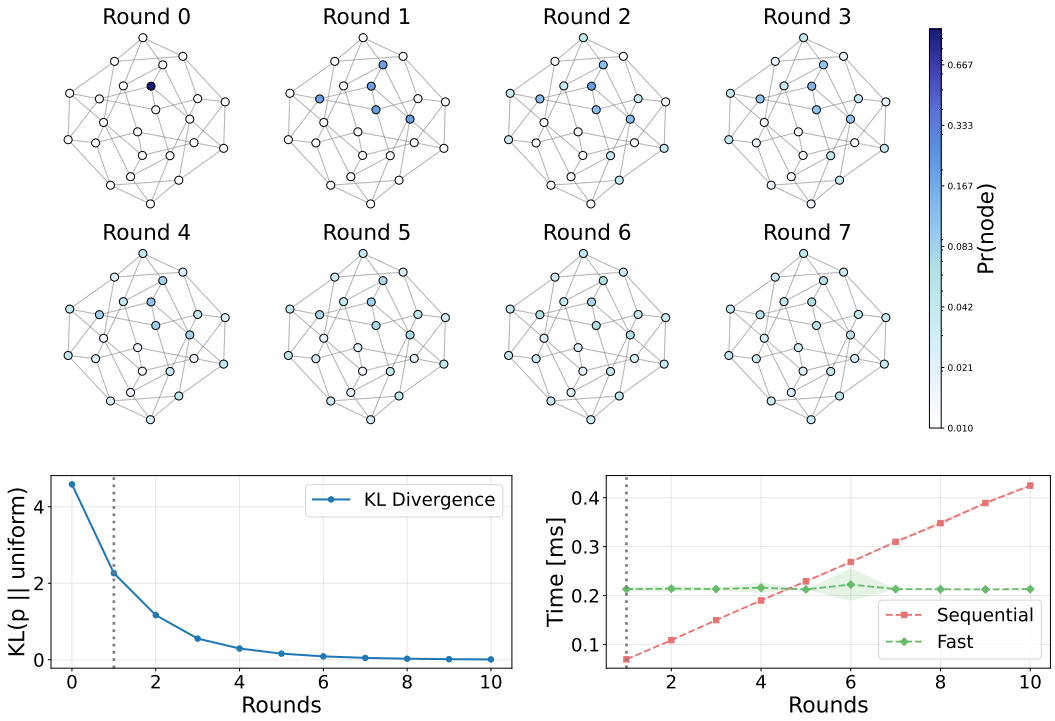

Figure 4: Cayley graph generated by SL(3). The top shows the first steps of the information diffusion and the bottom left shows the corresponding distance to the uniform distribution. On the right, the measured time to compute the expander convolution either sequentially or in closed form.

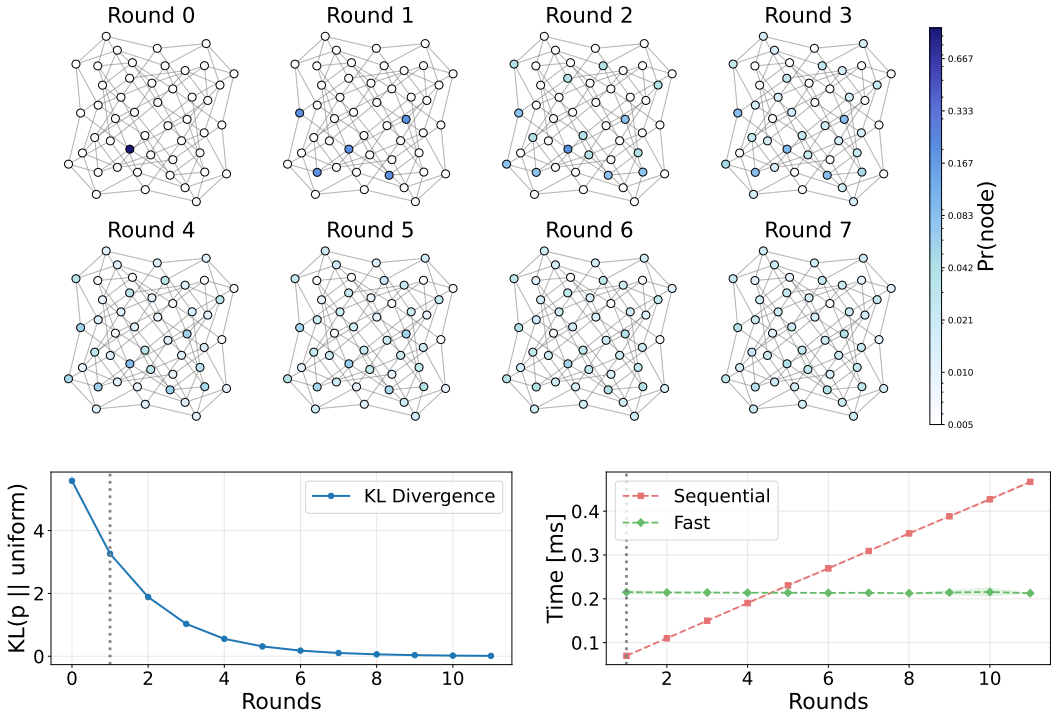

Figure 5: Cayley graph generated by SL(4). The top shows the first steps of the information diffusion and the bottom left shows the corresponding distance to the uniform distribution. On the right, the measured time to compute the expander convolution either sequentially or in closed form.

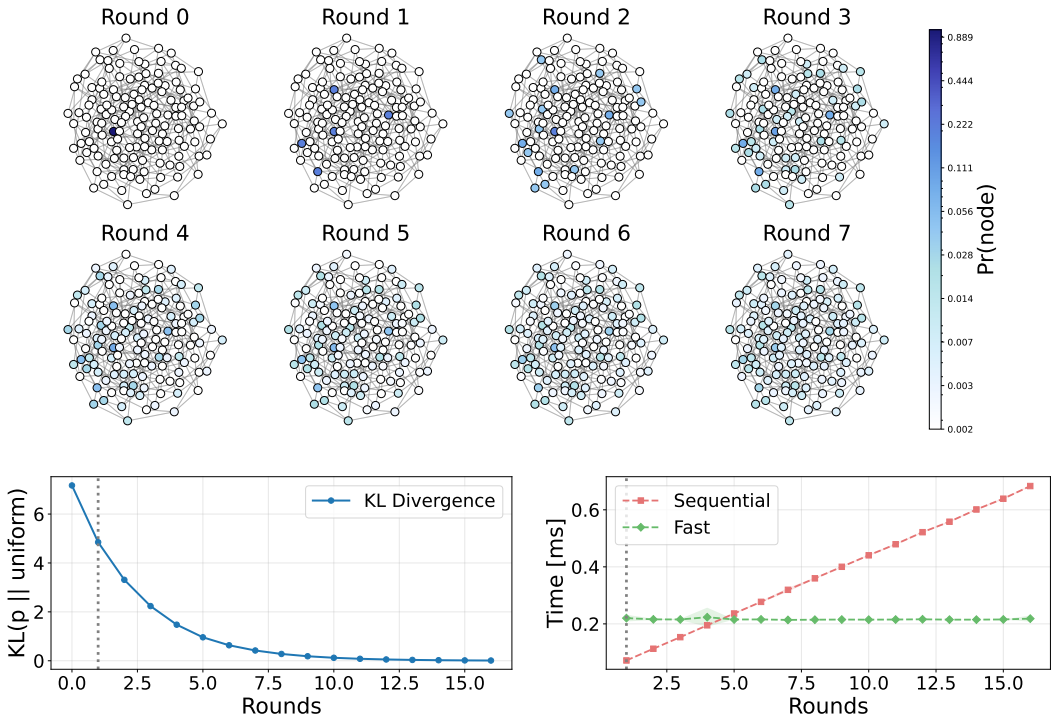

Figure 6: Cayley graph generated by SL(6). The top shows the first steps of the information diffusion and the bottom left shows the corresponding distance to the uniform distribution. On the right, the measured time to compute the expander convolution either sequentially or in closed form.

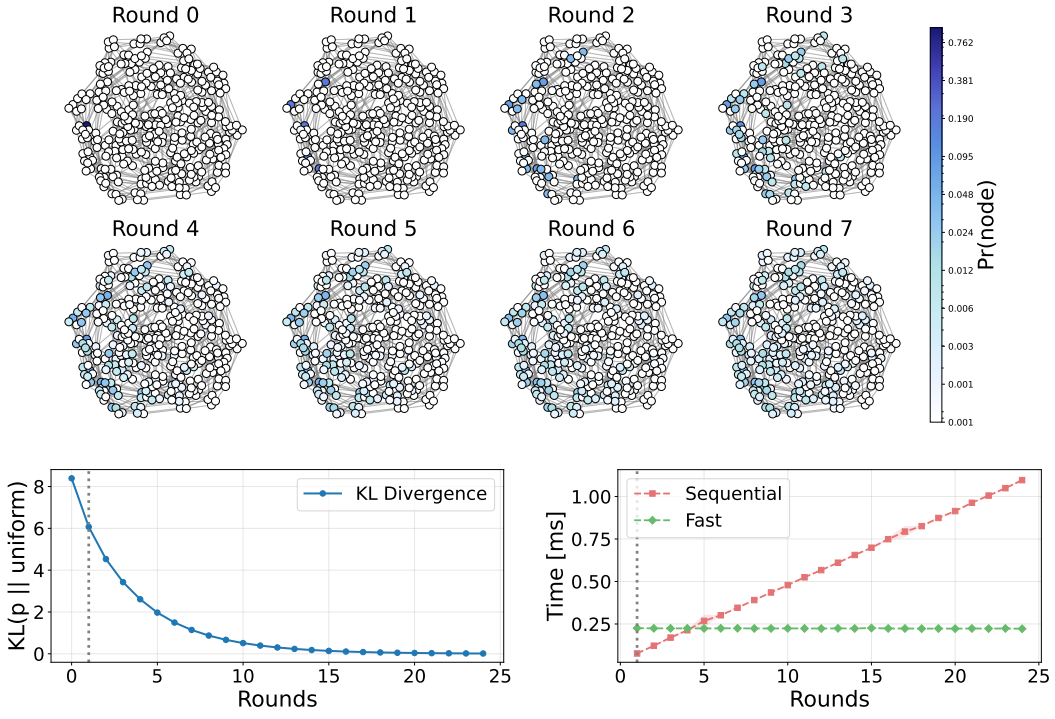

Figure 7: Cayley graph generated by SL(7). The top shows the first steps of the information diffusion and the bottom left shows the corresponding distance to the uniform distribution. On the right, the measured time to compute the expander convolution either sequentially or in closed form.

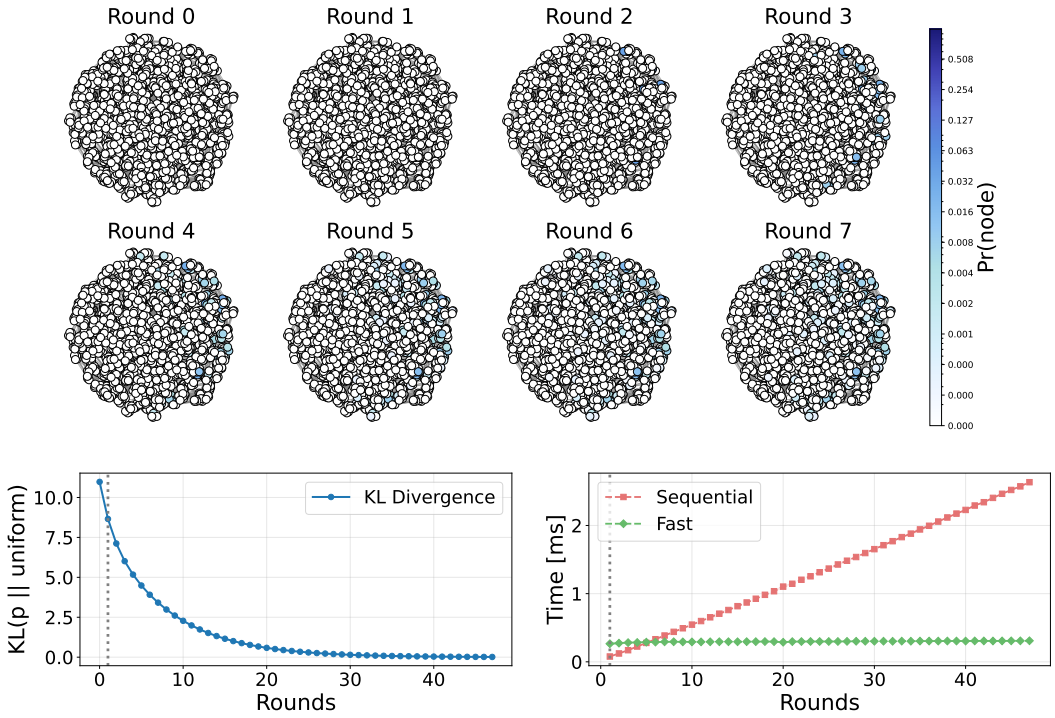

Figure 8: Cayley graph generated by SL(14). The top shows the first steps of the information diffusion and the bottom left shows the corresponding distance to the uniform distribution. On the right, the measured time to compute the expander convolution either sequentially or in closed form.

```python
import torch
from torch_geometric.nn.conv.gcn_conv import gcn_norm
from torch_geometric.nn import GCNConv

def edge_index_to_gso(edge_index):
    # GCN normalization + self-loops, then densify
    edge_index_n, edge_weight_n = gcn_norm(
        edge_index, add_self_loops=True, num_nodes=None
    )
    num_nodes = int(edge_index.max().item()) + 1
    A = torch.sparse_coo_tensor(edge_index_n, edge_weight_n,
                                (num_nodes, num_nodes)).to_dense()
    return A

class GraphSSMExample(torch.nn.Module):
    """
    x_{t+1} = A x_t W + U B
    Uses an explicit (dense) GSO A with GCN-style normalization + self-loops.
    """
    def __init__(self, hidden_dim):
        super().__init__()
        self.hidden_dim = hidden_dim
        self.W = torch.nn.Parameter(torch.randn(hidden_dim, hidden_dim))
        self.B = torch.nn.Parameter(torch.randn(hidden_dim, hidden_dim))

    def dense_implementation(self, batch, num_steps):
        U = batch.x                                 # (n, d)
        A = edge_index_to_gso(batch.edge_index)     # (n, n)

        x = torch.zeros_like(U)                     # (n, d)
        for _ in range(num_steps):
            x = A @ x @ self.W + U @ self.B
        return x
    def gcn_implementation(self, batch, num_steps):
        U = batch.x
        device, dtype = U.device, U.dtype

        # GCN with self-loops + normalization;
        gcn = GCNConv(
            self.hidden_dim, self.hidden_dim,
            bias=False, add_self_loops=True, normalize=True).to(device)

        # Make GCN's internal linear be "x @ W" (note the transpose).
        gcn.lin.weight = torch.nn.Parameter(self.W.T.clone().to(device, dtype))

        x = torch.zeros_like(U)
        for _ in range(num_steps):
            x = gcn(x, batch.edge_index) + U @ self.B
        return x

    def fast_implmentation(self, batch, num_steps):
        A = edge_index_to_gso(batch.edge_index)               # (n, n)
        sig, P = torch.linalg.eigh(A)                         # A = P diag(sig) P^T

        sig_w, V = torch.linalg.eig(self.W)                   # W = V diag(sig_w) V^{-1}
        V_inv = torch.linalg.inv(V)

        U = batch.x
        # Transform U B into both eigen-bases
        BV = (self.B.to(torch.cfloat) @ V.to(torch.cfloat))
        S  = (P.T @ U).to(torch.cfloat) @ BV                  # (n, d)

        T = int(num_steps)
        k = torch.arange(T, device=U.device)

        # Powers across time for each eigenvalue
        A_pow = sig.unsqueeze(0).pow(k.unsqueeze(1))          # (T, n)
        W_pow = sig_w.unsqueeze(0).pow(k.unsqueeze(1))        # (T, d)

        weights = A_pow.to(torch.cfloat).transpose(0, 1) @ W_pow  # (n, d)

        X_hat = S * weights
        X = P.to(torch.cfloat) @ X_hat @ V_inv.to(torch.cfloat)
        return X.real.to(U.dtype)
```

Listing 1: Example implementations of the Graph State-Space Model using torch and pytorch geometric. This assumes that U remains constant over time, for the original, both formulation and fast parallel implementation of the generalized case, refer to the works of Ceni et al. [2025].

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims are supported by multiple experiments.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations and drawbacks of the methods are discussed in combination with future outlooks.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification:

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Details of the experimental evaluation are provided in the Appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.

- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The used code will be made public upon acceptance, furthermore, the supplementary provides sufficient details to fully reproduce the experiments.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).

- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Yes, these details are discussed in the Appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Yes, the reported results include either standard deviations or confidence intervals where appropriate.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Yes, these details are provided in the Appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.

- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research is conducted to conform with the Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: We do not forsee any direct societal impacts as consequence of this work.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: We do not forsee any significant risks as direct consequence of this technical work.

    Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Yes, properly credited.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Sufficient documentation is provided.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

Justification: Not Applicable.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: NA

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: No core development involved LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.