pLSTM: parallelizable Linear Source Transition Mark networks

Korbinian Pöppel [1 2] **Richard Freinschlag** [1] **Thomas Schmied** [1] **Wei Lin** [1] **Sepp Hochreiter** [1 2]

Abstract

Modern recurrent architectures, such as xLSTM and Mamba, have recently challenged the Transformer in language modeling. However, their structure constrains their applicability to sequences only or requires processing multi-dimensional data structures, such as images or molecular graphs, in a pre-defined sequential order. In contrast, Multi-Dimensional RNNs (MDRNNs) are well suited for data with a higher level structure, like 2D grids, trees, and directed acyclic graphs (DAGs). In this work, we extend the notion of multi-dimensionality to linear RNNs. We introduce parallelizable Linear Source Transition Mark networks (pLSTMs) using Source, Transition, and Mark gates that act on the linegraph of a general DAG. This enables parallelization in analogy to parallel associative scans and the chunkwise-recurrent form of sequential linear RNNs, but for DAGs. For regular grids (1D and 2D), like images, this scheme can be efficiently implemented using einsum operations, concatenations, and padding in logarithmic time. pLSTMs tackle the vanishing/exploding activation/gradient problem for long distances in DAGs via two distinct modes: a directed propagation mode (P-mode) and a diffusive distribution mode (D-mode). To showcase the long-range capabilities of pLSTM, we introduce arrow-pointing extrapolation as a synthetic computer vision task that contains long-distance directional information. We demonstrate that pLSTMs generalize well to larger image sizes, whereas Transformers struggle to extrapolate. On established molecular graph and computer vision benchmarks, pLSTMs also show strong performance. Code and Datasets are available at: `https://github.com/ml-jku/plstm_experiments`.

[*]Equal contribution [1]ELLIS Unit Linz, LIT AI Lab, Institute for Machine Learning, Johannes Kepler University, Austria [2]NXAI GmbH, Linz, Austria. Correspondence to: Korbinian Pöppel <poeppel@ml.jku.at>.

Workshop Contribution

1. Introduction

Linear RNNs such as DeltaNet (Schlag et al., 2021), Gated Linear Attention (Yang et al., 2023), Mamba (Gu & Dao, 2023; Dao & Gu, 2024), and xLSTM (mLSTM) (Beck et al., 2025b) have recently evolved as a powerful alternative to the Transformer (Vaswani et al., 2017). In addition to sequence-parallelizable training, such modern recurrent architectures are more efficient at inference time than Transformers. In contrast to classical RNNs, they come with associative memory expansion, enabling more performant sequence modeling. However, these modern recurrent architectures are inherently limited to sequences. While linear RNNs have shown good performance on multi-dimensional data such as images, they enforce a pre-defined sequential traversal structure (Zhu et al., 2024; Alkin et al., 2025). For images, this corresponds to processing patches in a scanline form. This is problematic as, within one layer, vertical neighbors in a patch grid are only distantly related, in a line-wise processing. When switching between a line- and column-wise order across layers, diagonal relationships are not covered sufficiently. This mismatch of short-range spatial and long-distance relations in a certain path with precise positioning requirements leads to credit assignment problems (Minsky, 1961; Bengio & Frasconi, 1993; Schmidhuber, 2015) and suffers from vanishing activations/gradients (Hochreiter, 1991; Bengio et al., 1994; Hochreiter et al., 2001; Hochreiter & Schmidhuber, 1997; Pascanu et al., 2013).

Multi-dimensional RNNs (MDRNNs) (Graves et al., 2007) have demonstrated how non-linear RNNs like LSTMs (Hochreiter, 1991; Hochreiter & Schmidhuber, 1997) can be extended to multi-dimensional data, such as images (Graves et al., 2007; van den Oord et al., 2016), trees (Tai et al., 2015), and DAGs (Peng et al., 2017). While these works can be applied to general DAGs, they do not tackle the exploding/vanishing activation/gradient problem. In this work, we translate the findings of MDRNNs to linear RNNs and introduce parallelizable Linear Source Transition Mark Networks (pLSTM).

To showcase the long-range capabilities of pLSTM in multiple dimensions, we first introduce the synthetic arrow-pointing extrapolation task (see Figure 5). Importantly, pLSTM generalizes well to increasing image resolutions compared to the Transformer. Moreover, experiments on

ImageNet-1k and on molecular graphs show promising results compared to baselines and good scaling behavior to larger model sizes. In summary, we make the following **contributions**:

- We translate the findings of classic MDRNNs to linear RNNs and introduce pLSTM, with adapted gates and a scalable chunkwise-parallel formulation.
- We formally derive the general stabilization of pLSTMs for long-range propagation on general DAGs (including images), establishing the directional propagation or P- and diffusive distribution or D-mode.
- We introduce the synthetic Arrow Pointing Task to highlight the theoretical advantage of pLSTMs, in which pLSTM shows strong extrapolation abilities, and provide a highly scalable implementation of pLSTM.

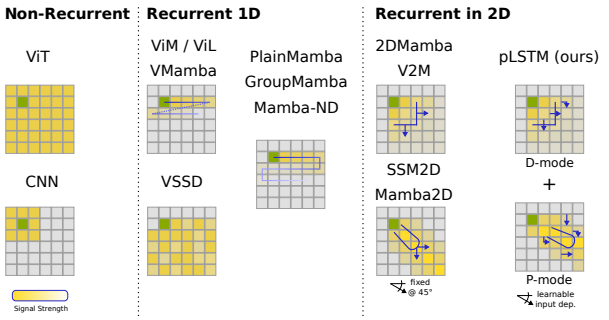

Figure 1: Illustration of the **receptive fields** induced by pLSTM and related architectures (for a single layer). CNNs are locally restricted while ViTs have a global receptive field. Modern recurrent architectures, such as ViM, traverse the 2D grid sequentially. pLSTM effectively extends the receptive field via its combination of D-mode and P-mode.

2. Related Work

Recent works extend the Mamba architecture to two dimensions in several ways. Most related to our work are 2DMamba (Zhang et al., 2024a) and V2M (Wang et al., 2024), which cover only the D-mode of pLSTM, at the loss of directional propagation. Mamba2D (Baty et al., 2024) and 2-D SSM (Baron et al., 2024) cover the P-mode of pLSTM, but unlike pLSTM, they are limited to the diagonal line instead of input-directed propagation along any line due to their normalization factor of $1/2$ (see Appendix F.2). For a more detailed discussion of related work, see Appendix C.

3. pLSTM

Similar to how LSTMs are extended to general DAGs, this can be applied to linear RNNs. Here, we extend the setting of MDRNNs and DAG LSTMs to enable guided directional propagation. Instead of being associated with a node in the graph, the Cell state C_e is now associated with an edge. The inputs and outputs / hidden states are still associated with nodes. Using this reformulation, we can define generalized linear DAG networks that resemble the LSTM structure (see Figures 4 and 3):

$$C_{ekv} = \sum_{e' \in E_{\rightarrow}(n_{\rightarrow}(e))} T_{ee'}^{n_{\rightarrow}(e)} C_{e'kv} + S_e^{n_{\rightarrow}(e)} K_k^{n_{\rightarrow}(e)} V_v^{n_{\rightarrow}(e)} \tag{1}$$

$$H_v^n = Q_k^n \sum_{e \in E_{\rightarrow}(n)} M_e^n C_{ekv} + D^n Q_k^n K_k^n V_k^n \tag{2}$$

The Cell state receives an external input via the Source $S_e^{n_{\rightarrow}(e)}$ at the incoming node in addition to a sum of the Cell states of incoming edges to the incoming node modified by the respective Transition $T_{ee'}^{n_{\rightarrow}(e)}$. The hidden state / network output H_v^n at node n is a sum of all incoming edge Cell states C_{ekv} modified by Mark M_e^n and projected by query Q_k^n. Note that the Source, Transition, and Mark gates are influenced by the input at node n, and optionally by the edge features of their associated local edges. Functionally, the Source replaces the input gate, the Transition the forget gate, and the Mark the output gate of a traditional LSTM (Hochreiter & Schmidhuber, 1997; Gers, 1999), while acting here on the node-edge, edge-edge, and edge-node combinations. The direct D^n term represents a skip directly from input to output at a node and also integrates the Q_k^n, K_k^n, V_k^n terms. These equations define general parallelizable Linear Source Transition Mark (pLSTM) networks on DAGs. This S-T-M structure can be computed sequentially, going through the DAG in a topological node order.

3.1. Parallelization on DAGs

3.1.1. Naïve Parallelization over Paths

Since all operations are linear, the iterative definition of Equation 1 can also be resolved to a sum over all ancestor nodes and a combination of Source, path Transition and Mark over all paths $\mathcal{P}(e', e)$ connecting nodes n' and n via first edge e' and last edge e:

$$H_v^n = \sum_{n' < n} K_k^{n'} V_v^{n'} \underbrace{\sum_{e', e} S_{e'}^{n'} \left(\sum_{P \in \mathcal{P}(e', e)} \prod_{n^p \in P} T_{e_{\rightarrow}^{n^p} e_{\leftarrow}^{n^p}}^{n^p} \right) M_e^n}_{G^{n'n}} Q_k^n$$
$$+ K_k^n V_k^n \underbrace{D^n}_{G^{nn}} Q_k^n \tag{3}$$

The middle STM part in Equation 3 can be precomputed to the gating matrix $G^{n'n}$, and the state expansion and projection via $K_k^{n'}$ and Q_k^n are externally integrated, similar

to classical (linear) self-attention (Vaswani et al., 2017; Katharopoulos et al., 2020), the D^n term contains self-connections. Computing the gating matrix via all paths is infeasible due to their exponential number (see Appendix F.1).

3.1.2. Hierarchical Parallelization

Similar to an associative scan in 1D, we can apply a divide-and-conquer approach for hierarchical parallelization. Recursively, we merge Source-Transition, Transition-Transition, and Transition-Mark combinations to higher-order Source, Transition, and Mark objects that cover more than a single node. In Appendix D.2, we show how this procedure works for DAGs. To better convey the core idea of pLSTM parallelization, we include a sketch for DAG and grid cases and pLSTM in general in Appendix Figure 3.

3.2. Long-Range Stability

For sequential data, it is known since the seminal LSTM work of Hochreiter & Schmidhuber (1997) that having forget gates larger or smaller than one in magnitude leads to exploding/vanishing activations and gradients. For general DAGs, this was not explored yet, with recent analyses on undirected GNNs (Arroyo et al., 2025). In DAG LSTM, Tree LSTM, Grid LSTM, or MD LSTM, potentially exploding parts are limited only in the hidden state by non-linear, bounded activation functions (i.e., tanh) (Hochreiter & Schmidhuber, 1997). The Cell states, however, can grow exponentially if the forget gates are limited to only one, see Appendix F.1 for details. Using the more general setting of Transitions $T^n_{e'e}$ from edge to edge, as introduced in this work, we can derive two modes of stability.

The first stabilization option for the **P-mode** is to limit the norm of $\boldsymbol{T}$ as in Equation 30. A node-local option to achieve this is to limit the Transition entries $T^n_{e'e}$ per node as:

$$\sum_{e \in E_{\leftarrow}(n)} |T^n_{e'e}| \overset{!}{\leqslant} 1 \tag{4}$$

For the second stabilization option, the **D-mode**, we do not limit a matrix norm of $\boldsymbol{T}$, but instead reduce the associated line-graph from a DAG to a multitree (Jung, 1978), i.e., a directed graph, such that two nodes are only connected by a single path, so no exponential growth (see Appendix F.1).

3.3. pLSTM on regular grids

Although pLSTMs can be used and parallelized on general DAGs, the additional structure greatly helps with the parallelization for regular grids. All operations can be decomposed into generalized matrix operations in the form of views, einsums, concatenations, and paddings. We show this for the 1D case of sequences in Appendix D.4.

3.3.1. pLSTM in 2D - Images

On a 2D undirected grid, pLSTM can be applied in different ways since there are more than two options for DAGs covering the undirected graph. There are, however, four distinct DAG covers, for which the local structure translates to a global structure: allowing only Transitions exclusively left or right and up or down, both in combination: $\curvearrowright\!\!\curvearrowleft$, $\curvearrowleft\!\!\curvearrowright$, $\curvearrowright\!\!\curvearrowright$, $\curvearrowleft\!\!\curvearrowleft$. We focus on the first, the right-down combination, for the description, but all four combinations are used.

As Transitions follow all edge combinations, there are four options again: $\rightarrow$, $\hookrightarrow$, $\curvearrowright$, $\downarrow$. The level zero Source, Transition, and Mark tensors are therefore: $S^{\mapsto}_{xy}, S^{\updownarrow}_{xy}, T^{\rightarrow}_{xy}, T^{\hookrightarrow}_{xy}, T^{\curvearrowright}_{xy}, T^{\downarrow}_{xy}, M^{\rightarrowtail}_{xy}, M^{\downarrowtail}_{xy}$. Additionally, we now use u for internal indices along the x-axis, and v for internal indices along the y-axis, and a, b, c for an index along the edge/boundary direction, while keeping x, y for outer indices. These are needed for the higher-level tensors, leading to a recursion shown in Appendix D.3. The Source, Transition and Mark tensors at level l have the following structure: $S^{l,\mapsto}_{xyuva}, S^{l,\updownarrow}_{xyuva}, T^{l,\rightarrow}_{xyab}, T^{l,\hookrightarrow}_{xyab}, T^{l,\curvearrowright}_{xyab}, T^{l,\downarrow}_{xyab}, M^{l,\rightarrowtail}_{xyuvb}, M^{l,\downarrowtail}_{xyuvb}$ with $u, v, a, b \in \{0..2^l - 1\}$. The general DAG is depicted in Figure 3 in the center right. As the in- and out-degrees of this DAG are now larger than one, there are more stabilization options, the P-mode and D-mode (Section 3.2).

P-mode When we allow all Transition options $\rightarrow$, $\hookrightarrow$, $\curvearrowright$, $\downarrow$, we are in the P-mode. In general, the Transition matrix at position x, y looks like $T_{xy} = \begin{bmatrix} T^{\rightarrow}_{xy} & T^{\hookrightarrow}_{xy} \\ T^{\curvearrowright}_{xy} & T^{\downarrow}_{xy} \end{bmatrix}$. The P-mode stabilization now limits this matrix to absolute column sums $\leqslant 1$. For a practical parameterization, we therefore fix this structure with the 'angle' of propagation α and an additional decay factor γ (see also Appendix F.2):

$$T = \gamma \begin{bmatrix} \alpha & \alpha \\ (1-\alpha) & (1-\alpha) \end{bmatrix} \tag{5}$$

D-mode While the P-mode offers a directional propagation, the receptive field of a node/patch/pixel is limited to roughly a line -it cannot reach all other nodes equally. The D-mode can provide this by losing the notion of directionality. For the D-mode, either $T^{\hookrightarrow}_{xy}$ or $T^{\curvearrowright}_{xy}$ have to be set to zero globally. In this way, two edges can only be connected via a single path. In Figure 3, the bottom right, the second option is shown, with an additional decay. At criticality, this node reaches all other nodes to its bottom right.

Directional Initialization and Combination Using the concept of multiple heads, as used in Vaswani et al. (2017), we can initialize our network to cover multiple directions in P-mode at criticality on initialization. Similarly, we can initialize different decay scales for both P- and D-mode for

different heads. By using the P- and D-mode in alternating layers, we leverage both their potential.

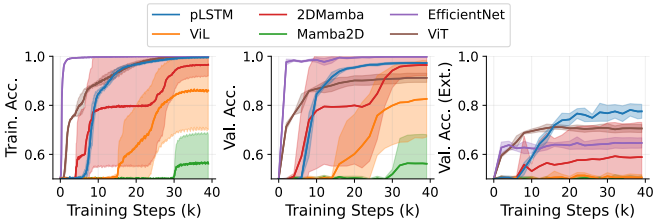

Figure 2: Training curves for the **Arrow Pointing Extrapolation task**, averaged over 5 seeds with 90% CI. ViT and EfficientNet can quickly match the training set (left), and EfficientNet reaches the best validation performance of all models on the samples of the same resolution as seen during training. pLSTM performs best on the extrapolation set (right) by a significant gap. While Mamba2D and 2DMamba should cover restricted modes of pLSTM, their learning shows high variance. ViL and Mamba2D do not extrapolate at all beyond random performance.

4. Experiments

We first showcase the theoretical advantages of pLSTM on the synthetic arrow-pointing extrapolation task (see Section 4.1). In the appendix, we highlight the benefits of pLSTM for two-dimensional input data on ImageNet-1K (Deng et al., 2009; Russakovsky et al., 2015), demonstrating scalability to large-scale datasets (see Appendix G.3 and G.4). We show that pLSTM scales to more than two input dimensions on established graph benchmarks (see Section 4.2).

4.1. Arrow Pointing Extrapolation

In this task, an image containing an arrow and a circle must be classified as to whether the arrow is pointing to the circle. This is a long-range directional task as it involves the long-range relative positioning of two objects, combined with local information from one of them - the direction of the arrow. To test for arrow pointing extrapolation capabilities, we generate a balanced dataset of 100k arrow pointing images at resolution 192×192, with positions of the arrow and circle randomly sampled. For validation, we generate 5120 images in the same resolution and at resolution 384×384 - to test for extrapolation capabilities. For all models, including the ViT (Dosovitskiy et al., 2021) baseline, we use a bicubic interpolation of the positional embeddings (via `jax.image.resize`) for scaling to higher resolutions (see also (Dosovitskiy et al., 2021)). For results, see Figure 2, for details on the training schedule, see Appendix G.2.

4.2. Graphs

Small molecules and proteins are typically depicted as an undirected graph. However, both exist as 3D structures in the real world, and GNNs trained on such datasets might benefit from directional information propagation, as there is an underlying spatial relation. We test this hypothesis on popular small molecules and bioinformatics datasets from the TUDataset benchmark (Morris et al., 2020). The results in Table 1 show that pLSTM can compete with popular GNN architectures on those datasets. Similar to the computer vision experiments (Section G.3), pLSTM alternates between the P-mode and the D-mode. Since there is no external notion of direction, we use node and edge features to compute the Transition for the P-mode.

Table 1: Test accuracy on small molecule and bioinformatics datasets as provided by **TUDataset**.

Model	MUTAG	NCI1	PROTEINS	PTC_FM	AVG
GAT (Veličković et al., 2018)	0.7822 ± 0.09	0.7968 ± 0.03	0.7215 ± 0.03	0.6105 ± 0.05	0.7277 ± 0.06
GCN (Kipf & Welling, 2017)	0.7234 ± 0.08	0.7852 ± 0.02	0.7395 ± 0.03	**0.6162 ± 0.04**	0.7161 ± 0.05
GIN (Xu et al., 2018)	0.8251 ± 0.10	**0.8175 ± 0.02**	0.7350 ± 0.04	0.6097 ± 0.10	**0.7468 ± 0.08**
LSTM GNN (Liang et al., 2016)	0.7450 ± 0.11	0.7951 ± 0.02	**0.7503 ± 0.04**	0.6076 ± 0.04	0.7245 ± 0.06
MPNN (Gilmer et al., 2017)	0.7450 ± 0.09	0.8012 ± 0.02	0.7350 ± 0.04	0.5786 ± 0.07	0.7149 ± 0.06
PLSTM	**0.8512 ± 0.06**	0.7324 ± 0.03	0.7502 ± 0.05	0.6133 ± 0.08	0.7368 ± 0.06

5. Conclusion

In this work, we introduce pLSTM, which unites the benefits of MDRNNs (Graves et al., 2007) and the recently introduced xLSTM (Beck et al., 2025b). pLSTM overcomes the limitations of modern recurrent architectures when applied to multi-dimensional data, such as images and graphs. To achieve this, we modify the gating structure of xLSTM and introduce Source, Transition, and Mark gates. Then, we derive a parallelization scheme that enables processing data in multiple dimensions concurrently. pLSTM comes with a P-mode and a D-mode, which together enable a large and auto-tunable effective receptive field. We demonstrate the theoretical advantages of pLSTM on the arrow-pointing task and highlight its ability to generalize to varying grid resolutions. Finally, we show the efficacy of pLSTM both on classical computer vision and graph benchmarks.

Limitations & Future Work pLSTM shows promising results across different domains, however, there is still room for improvement compared to highly optimized domain-specific models. With the incorporation of domain-specific inductive biases, we are positive that the results can be further improved. Moreover, while pLSTM models generalize better than ViTs, which use position embeddings to encode spatial relations, there is still a gap towards perfect extrapolation. This leaves room for improvements, both on the data side of harder multi-dimensional long-range benchmarks and the architectural side to better generalize to that data. Given the flexibility of pLSTM to handle data with rich multi-dimensional structure, we anticipate that pLSTMs can be successfully applied across a broader spectrum of challenging domains, including biology, chemistry, medical imaging, and other scientific domains.

Acknowledgements

We thank Benedikt Alkin, Pieter-Jan Hoedt, Phillip Lippe, Maximilian Beck, Kajetan Schweighofer, Erich Kobler, and Günter Klambauer for helpful discussions and feedback.

The ELLIS Unit Linz, the LIT AI Lab, the Institute for Machine Learning, are supported by the Federal State Upper Austria. We thank the projects FWF AIRI FG 9-N (10.55776/FG9), AI4GreenHeatingGrids (FFG- 899943), Stars4Waters (HORIZON-CL6-2021-CLIMATE-01-01), FWF Bilateral Artificial Intelligence (10.55776/COE12). We thank NXAI GmbH, Audi AG, Silicon Austria Labs (SAL), Merck Healthcare KGaA, GLS (Univ. Waterloo), TÜV Holding GmbH, Software Competence Center Hagenberg GmbH, dSPACE GmbH, TRUMPF SE + Co. KG.

We acknowledge EuroHPC Joint Undertaking for awarding us access to Leonardo at CINECA, Italy, and MareNostrum5 at BSC, Spain.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Alkin, B., Beck, M., Pöppel, K., Hochreiter, S., and Brandstetter, J. Vision-LSTM: xLSTM as Generic Vision Backbone. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=SiH7DwNKZZ.

Arroyo, A., Gravina, A., Gutteridge, B., Barbero, F., Gallicchio, C., Dong, X., Bronstein, M., and Vandergheynst, P. On Vanishing Gradients, Over-Smoothing, and Over-Squashing in GNNs: Bridging Recurrent and Graph Learning, February 2025. URL http://arxiv.org/abs/2502.10818. arXiv:2502.10818 [cs].

Baron, E., Zimerman, I., and Wolf, L. A 2-Dimensional State Space Layer for Spatial Inductive Bias. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=BGkqypmGvm.

Baty, E., Díaz, A. H., Bridges, C., Davidson, R., Eckersley, S., and Hadfield, S. Mamba2D: A Natively Multi-Dimensional State-Space Model for Vision Tasks, December 2024. URL http://arxiv.org/abs/2412.16146. arXiv:2412.16146 [cs].

Beck, M., Pöppel, K., Lippe, P., and Hochreiter, S. Tiled flash linear attention: More efficient linear rnn and xlstm kernels. *arXiv preprint arXiv:2503.14376*, 2025a.

Beck, M., Pöppel, K., Spanring, M., Auer, A., Prudnikova, O., Kopp, M., Klambauer, G., Brandstetter, J., and Hochreiter, S. xLSTM: Extended long short-term memory. *Advances in Neural Information Processing Systems*, 37:107547–107603, 2025b.

Behrouz, A., Zhong, P., and Mirrokni, V. Titans: Learning to memorize at test time. *arXiv preprint arXiv:2501.00663*, 2024.

Bengio, Y. and Frasconi, P. Credit assignment through time: Alternatives to backpropagation. *Advances in neural information processing systems*, 6, 1993.

Bengio, Y., Simard, P., and Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

Chen, X., Shi, Z., Qiu, X., and Huang, X. Dag-based long short-term memory for neural word segmentation. *arXiv preprint arXiv:1707.00248*, 2017.

Dao, T. and Gu, A. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. *arXiv preprint arXiv:2405.21060*, 2024.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

Duan, Y., Wang, W., Chen, Z., Zhu, X., Lu, L., Lu, T., Qiao, Y., Li, H., Dai, J., and Wang, W. Vision-rwkv: Efficient and scalable visual perception with rwkv-like architectures. *arXiv preprint arXiv:2403.02308*, 2024.

Gers, F. Learning to forget: continual prediction with LSTM. In *9th International Conference on Artificial Neural Networks: ICANN '99*, volume 1999, pp. 850–855, Edinburgh, UK, 1999. IEE. ISBN 978-0-85296-721-8. doi: 10.1049/cp:19991218. URL https://digital-library.theiet.org/content/conferences/10.1049/cp_19991218.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural Message Passing for Quantum Chemistry. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pp. 1263–1272, 2017.

Graves, Mohamed, A.-R., and Hinton, G. E. Speech recognition with deep recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 6645–6649. IEEE, 2013. doi: 10.1109/icassp.2013.6638947.

Graves, A., Fernández, S., and Schmidhuber, J. Multidimensional recurrent neural networks. In *International Conference on Artificial Neural Networks (ICANN)*, volume 4668 of *Lecture Notes in Computer Science*, pp. 6645–6649. Springer, Berlin, Heidelberg, 2007.

Grazzi, R., Siems, J., Franke, J. K. H., Zela, A., Hutter, F., and Pontil, M. Unlocking State-Tracking in Linear RNNs Through Negative Eigenvalues. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=UvTo3tVBk2.

Gu, A. and Dao, T. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

Hochreiter, S. Untersuchungen zu dynamischen neuronalen Netzen. Diploma thesis, Institut für Informatik, Lehrstuhl Prof. Brauer, Technische Universität München, 1991.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, 1997.

Hochreiter, S., Bengio, Y., Frasconi, P., Schmidhuber, J., et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

Hopfield, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, April 1982. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.79.8.2554. URL https://pnas.org/doi/full/10.1073/pnas.79.8.2554.

Horn, R. A. and Johnson, C. R. *Matrix Analysis*. Cambridge University Press, 1 edition, December 1985. ISBN 978-0-521-38632-6 978-0-521-30586-0 978-0-511-81081-7. doi: 10.1017/CBO9780511810817. URL https://www.cambridge.org/core/product/identifier/9780511810817/type/book.

Jung, H. A. On a class of posets and the corresponding comparability graphs. *Journal of Combinatorial Theory, Series B*, 24(2):125–133, 1978. ISSN 0095-8956. doi: https://doi.org/10.1016/0095-8956(78)90013-8. URL https://www.sciencedirect.com/science/article/pii/0095895678900138.

Kalchbrenner, N., Danihelka, I., and Graves, A. Grid long short-term memory. *arXiv*, 1507.01526, 2015.

Katharopoulos, A., Vyas, A., Pappas, N., and Fleuret, F. Transformers are RNNs: fast autoregressive transformers with linear attention. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

Kipf, T. N. and Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations*, 2017.

Li, S., Singh, H., and Grover, A. Mamba-ND: Selective State Space Modeling for Multi-dimensional Data. In Leonardis, A., Ricci, E., Roth, S., Russakovsky, O., Sattler, T., and Varol, G. (eds.), *Computer Vision – ECCV 2024*, pp. 75–92, Cham, 2025. Springer Nature Switzerland. ISBN 978-3-031-73414-4.

Liang, X., Shen, X., Feng, J., Lin, L., and Yan, S. Semantic object parsing with graph LSTM. *CoRR*, abs/1603.07063, 2016. URL http://arxiv.org/abs/1603.07063.

Liu, Z., Mao, H., Wu, C.-Y., Feichtenhofer, C., Darrell, T., and Xie, S. A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11976–11986, 2022.

Merrill, W., Petty, J., and Sabharwal, A. The illusion of state in state-space models. *arXiv preprint arXiv:2404.08819*, 2024.

Minsky, M. Steps toward artificial intelligence. *Proceedings of the IRE*, 49(1):8–30, 1961. doi: 10.1109/JRPROC.1961.287775.

Morris, C., Kriege, N. M., Bause, F., Kersting, K., Mutzel, P., and Neumann, M. TUDataset: A collection of benchmark datasets for learning with graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*, 2020.

Orvieto, A., Smith, S. L., Gu, A., Fernando, A., Gulcehre, C., Pascanu, R., and De, S. Resurrecting recurrent neural networks for long sequences. In *International Conference on Machine Learning*, pp. 26670–26698. PMLR, 2023.

Park, J. deit3-jax, 2024. URL https://github.com/affjljoo3581/deit3-jax. Publication Title: GitHub repository.

Pascanu, R., Mikolov, T., and Bengio, Y. On the difficulty of training recurrent neural networks. In *International*

conference on machine learning, pp. 1310–1318. Pmlr, 2013.

Peng, N., Poon, H., Quirk, C., Toutanova, K., and Yih, W.-t. Cross-sentence n-ary relation extraction with graph lstms. *Transactions of the Association for Computational Linguistics*, 5:101–115, 2017.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. The Graph Neural Network Model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009. doi: 10.1109/TNN.2008.2005605.

Schiff, Y., Kao, C.-H., Gokaslan, A., Dao, T., Gu, A., and Kuleshov, V. Caduceus: Bi-directional equivariant long-range dna sequence modeling. *arXiv preprint arXiv:2403.03234*, 2024.

Schlag, I., Irie, K., and Schmidhuber, J. Linear transformers are secretly fast weight programmers. In *International conference on machine learning*, pp. 9355–9366. PMLR, 2021.

Schmidhuber, J. Learning to Control Fast-Weight Memories: An Alternative to Dynamic Recurrent Networks. *Neural Computation*, 4:131–139, 1992. URL https://api.semanticscholar.org/CorpusID:16683347.

Schmidhuber, J. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

Schmidinger, N., Schneckenreiter, L., Seidl, P., Schimunek, J., Hoedt, P.-J., Brandstetter, J., Mayr, A., Luukkonen, S., Hochreiter, S., and Klambauer, G. Bio-xlstm: Generative modeling, representation and in-context learning of biological and chemical sequences. *arXiv preprint arXiv:2411.04165*, 2024.

Schmied, T., Adler, T., Patil, V., Beck, M., Pöppel, K., Brandstetter, J., Klambauer, G., Pascanu, R., and Hochreiter, S. A large recurrent action model: xlstm enables fast inference for robotics tasks. *arXiv preprint arXiv:2410.22391*, 2024.

Siems, J., Carstensen, T., Zela, A., Hutter, F., Pontil, M., and Grazzi, R. Deltaproduct: Improving state-tracking in linear rnns via householder products. *arXiv preprint arXiv:2502.10297*, 2025.

Stollenga, M., Byeon, W., Liwicki, M., and Schmidhuber, J. Parallel multi-dimensional lstm, with application to fast biomedical volumetric image segmentation. In *Proceedings of the 28th International Conference on Neural Information Processing Systems (NIPS)*, pp. 2998–3006, 2015.

Sun, Y., Li, X., Dalal, K., Xu, J., Vikram, A., Zhang, G., Dubois, Y., Chen, X., Wang, X., Koyejo, S., et al. Learning to (learn at test time): Rnns with expressive hidden states. *arXiv preprint arXiv:2407.04620*, 2024.

Tai, K. S., Socher, R., and Manning, C. D. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*, 2015.

Tan, M. and Le, Q. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pp. 6105–6114. PMLR, 2019.

Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., and Jégou, H. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pp. 10347–10357. PMLR, 2021.

Touvron, H., Cord, M., and Jégou, H. DeiT III: Revenge of the ViT. In *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXIV*, pp. 516–533, Berlin, Heidelberg, 2022. Springer-Verlag. ISBN 978-3-031-20052-6. doi: 10.1007/978-3-031-20053-3_30. URL https://doi.org/10.1007/978-3-031-20053-3_30. eventplace: Tel Aviv, Israel.

Van Den Oord, A., Kalchbrenner, N., and Kavukcuoglu, K. Pixel recurrent neural networks. In *International conference on machine learning*, pp. 1747–1756. PMLR, 2016.

van den Oord, A., Kalchbrenner, N., and Kavukcuoglu, K. Pixel Recurrent Neural Networks. In Balcan, M. F. and Weinberger, K. Q. (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 1747–1756, New York, New York, USA, June 2016. PMLR. URL https://proceedings.mlr.press/v48/oord16.html.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph Attention Networks. In *International Conference on Learning Representations*, 2018.

Wang, C., Zheng, W., Huang, Y., Zhou, J., and Lu, J. V2M: Visual 2-Dimensional Mamba for Image Representation Learning. *arXiv preprint arXiv:2410.10382*, 2024.

Whitney, H. Congruent Graphs and the Connectivity of Graphs. *American Journal of Mathematics*, 54 (1):150, January 1932. ISSN 00029327. doi: 10.2307/2371086. URL https://www.jstor.org/stable/2371086?origin=crossref.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How Powerful are Graph Neural Networks? In *International Conference on Learning Representations*, 2019.

Yang, S., Wang, B., Shen, Y., Panda, R., and Kim, Y. Gated linear attention transformers with hardware-efficient training. *arXiv preprint arXiv:2312.06635*, 2023.

Yang, S., Wang, B., Zhang, Y., Shen, Y., and Kim, Y. Parallelizing linear transformers with the delta rule over sequence length. *arXiv preprint arXiv:2406.06484*, 2024.

Yang, S., Kautz, J., and Hatamizadeh, A. Gated Delta Networks: Improving Mamba2 with Delta Rule. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=r8H7xhYPwz.

Zhang, J., Nguyen, A. T., Han, X., Trinh, V. Q.-H., Qin, H., Samaras, D., and Hosseini, M. S. 2dmamba: Efficient state space model for image representation with applications on giga-pixel whole slide image classification. *arXiv preprint arXiv:2412.00678*, 2024a.

Zhang, J., Nguyen, A. T., Han, X., Trinh, V. Q.-H., Qin, H., Samaras, D., and Hosseini, M. S. 2DMamba: Efficient State Space Model for Image Representation with Applications on Giga-Pixel Whole Slide Image Classification, December 2024b. URL http://arxiv.org/abs/2412.00678. arXiv:2412.00678 [cs].

Zhu, L., Liao, B., Zhang, Q., Wang, X., and Liu, W. Vision mamba: Efficient visual representation learning with bidirectional state space model (preprint). 2024.

Zhu, X., Sobhani, P., and Guo, H. Dag-structured long short-term memory for semantic compositionality. In *Proceedings of the 2016 conference of the north american chapter of the association for computational linguistics: Human language technologies*, pp. 917–926, 2016.

A. Method Visualization

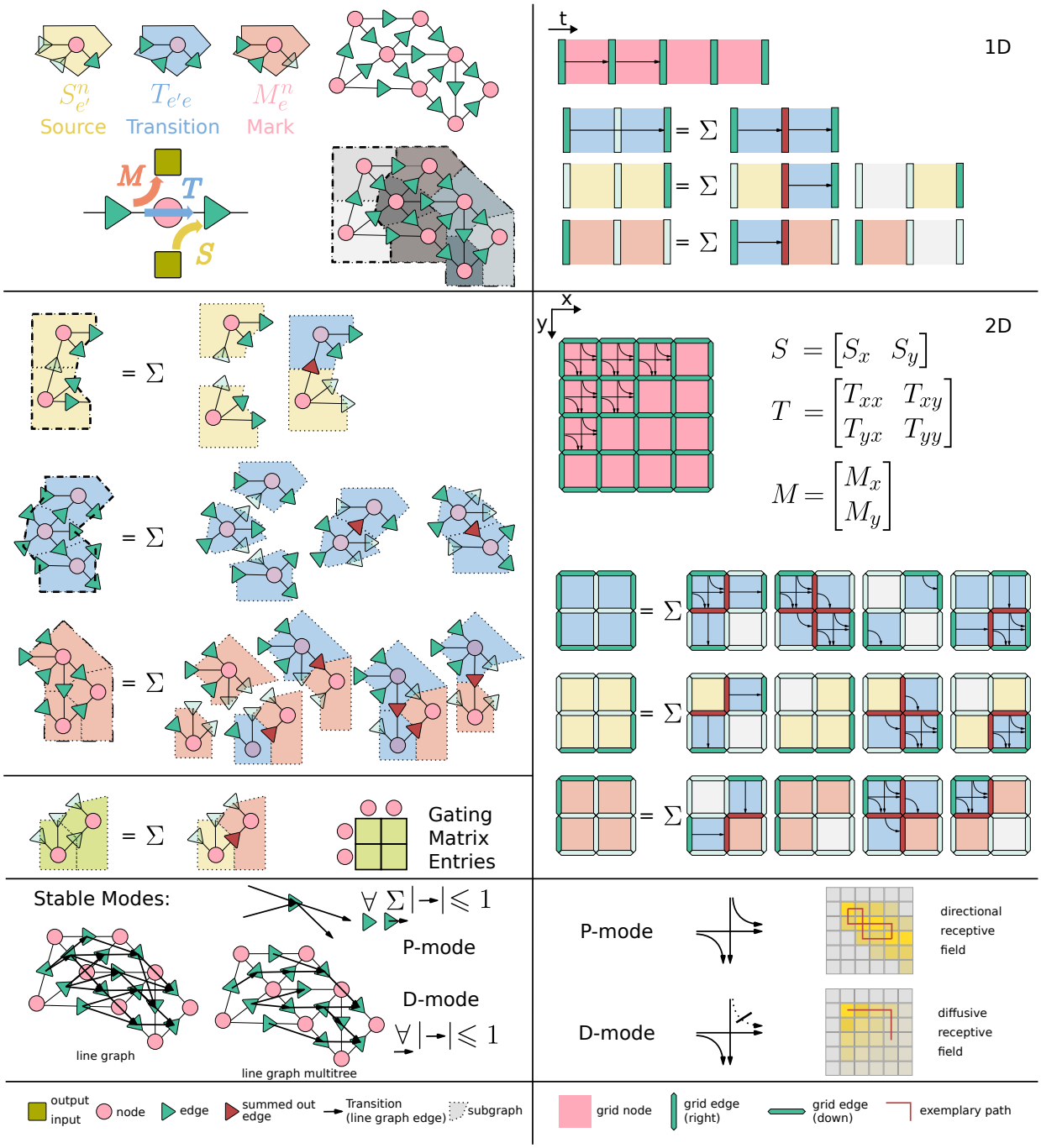

Figure 3: Illustration of **pLSTM** on general DAGs (left), and on 1D / 2D grids (right): In the top-left part, a general DAG is visualized, with Sources $S^n_{e'}$, Transitions $T_{e'e}$, and Marks M^n_e populating node-outgoing-edge, incoming-edge-outgoing-edge, and incoming-edge-node pairs. Due to the associative structure of these linear operators, they can be combined; examples are shown in the center left. The end-to-end (node-to-node) result is the Gating matrix. In the bottom left, the two stable modes are depicted: The P-mode, where the sum over absolute Transitions of an in-coming edge have to be limited by one, and the D-mode, where the line graph over edges is reduced to a multitree, requiring the remaining Transitions each to be limited in absolute to one. On the right, the application of this to regular 1D and 2D grids is shown. For 2D, the P-mode results in a directional propagation, whereas the D-mode results in a diffusive / un-directed distribution (shown with decay).

B. Background

B.1. Multidimensional RNNs

The most general form of MDRNNs (Graves et al., 2007) was introduced as DAG LSTM by Zhu et al. (2016), as it can be applied to all directed acyclic graphs $\mathcal{P}(N, E)$. We denote $n_{\rightarrow}(e), n_{\leftarrow}(e)$, incoming and outgoing node of an edge $e \in E$, and $E_{\rightarrow}(n), E_{\leftarrow}(n)$ the sets of incoming and outgoing edges of a node $n \in N$. The core element of (Multi-Dimensional) LSTMs is the LSTM Cell c_n computed as:

$$\boldsymbol{c}_n = \sum_{n' \in \mathcal{P}_{\mathcal{G}}(n)} \boldsymbol{f}_{nn'} \odot \boldsymbol{c}_{n'} + \boldsymbol{i}_n \odot \boldsymbol{z}_n \tag{6}$$

$$\boldsymbol{h}_n = \boldsymbol{o}_n \odot \tanh\left(\boldsymbol{c}_n\right) \tag{7}$$

with hidden state / output $\boldsymbol{h}_n$ and $\mathcal{P}_{\mathcal{G}}(n)$ denoting the parents of node n in a DAG $\mathcal{G}$. In standard LSTMs, the input $\boldsymbol{i}_n$, forget $\boldsymbol{f}_n$, output gates $\boldsymbol{o}_n$ and the Cell update $\boldsymbol{z}_n$ are dependent on external inputs $\boldsymbol{x}_n$ and on the previous/parent hidden states $(\boldsymbol{h}_{n'})_{n' \in \mathcal{P}_{\mathcal{G}}(n)}$, which makes them non-linear. In contrast, for undirected graphs, DAG LSTMs can be applied by ordering the nodes first, thereby fixing edge directions and using a DAG LSTM bi-directionally. Note that there is typically more than one ordering option, where different orders can lead to different outcomes.

B.2. Linear RNNs

Linear RNNs resemble the structure of the original LSTM. For linearization, they remove the previous hidden state dependency of the gates and cell update. Additionally, they commonly include a state expansion dimension or query / key dimension. This enables a form of associative memory with inner-product retrieval in relation to quadratic energy-based models (Hopfield, 1982; Schmidhuber, 1992). To include all relevant dimensions, we switch to the Einstein sum convention notation (so dimensions are visible in the indices) and absorb the input gate in the key and the output gate in the query. In reverse, query and key can also be interpreted as extended input and output gates. Additional normalization, as in mLSTM (Beck et al., 2025b), can be interpreted as a parallel execution with similar inputs and reduced dimensionality.

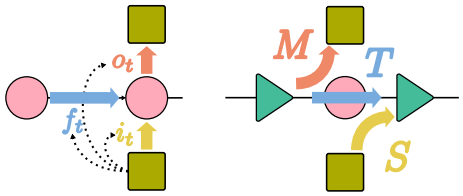

Figure 4: Transition from input/forget/output gating in between nodes (and input/output) in (linear) RNNs towards **Source/Transition/Mark gating** between edges and input/output (bottom/top) in pLSTMs.

$$C_{tkv} = F_t C_{t-1kv} + K_{tk} V_{tv} \tag{8}$$

$$H_{tv} = Q_{tk} C_{tkv} \tag{9}$$

For certain architectures, the scalar forget gate F_t is extended to the key dimension as F_{tk} in GLA (Yang et al., 2023) or a matrix in DeltaNet as $F_{tkk'} C_{t-1k'v}$ with $F_{tkk'} = 1 - \beta_t K_{tk} K_{tk'}$ (Schlag et al., 2021; Yang et al., 2024) or Gated DeltaNet (Yang et al., 2025). DeltaProduct (Siems et al., 2025) uses a product of multiple of these Householder-matrices. The important part is the linearization, by removing the dependence on the previous hidden state. All of F_t, K_{tk}, V_{tv} and Q_{tk} depend only on the input of the layer x_{td} for this time step. This structure enables a chunkwise-recurrent form for efficient execution on modern hardware (Beck et al., 2025a; Yang et al., 2023; 2024).

C. Extended Related work

Modern Recurrent Architectures This work presents a new form of linear RNNs, where DeltaNet (Schlag et al., 2021; Yang et al., 2024), LRU (Orvieto et al., 2023), GLA (Yang et al., 2023), Mamba (Gu & Dao, 2023; Dao & Gu, 2024), and xLSTM (Beck et al., 2025b) (in the mLSTM form) have shown their effectiveness on sequence modeling in the language domain. Recently, this line of work has been complemented by TTT (Sun et al., 2024), Titans (Behrouz et al., 2024), and DeltaProduct (Siems et al., 2025), which motivate this structure as a gradient-based optimization in context, in line with early work on Fast-Weight Programmers (Schlag et al., 2021). Due to their ability for parallelization during training, modern recurrent architectures scale to large-scale datasets similar to Transformers. Moreover, they come with efficient inference, which is attractive for real-world applications (Schmidinger et al., 2024; Schiff et al., 2024; Schmied et al., 2024).

Non-Linear Multi-Dimensional RNNs The first foundational extension of non-linear RNNs / LSTMs to multiple dimensions was carried out by Graves et al. (2007). Subsequently, Stacked LSTM was proposed (Graves et al., 2013), which stacks LSTM layers on top of each other. MDRNNs are hard to parallelize, while re-arranging the traditional cuboid order of computations in MDLSTMs in pyramidal fashion led to PyraMiD LSTM (Stollenga et al., 2015), which can be parallelized. Grid LSTM (Kalchbrenner et al., 2015) operates in multiple processing dimensions simultaneously, thus generalizing one-dimensional LSTM to more dimensions. Tree-LSTM (Tai et al., 2015) and DAG-LSTM (Zhu et al., 2016; Peng et al., 2017; Chen et al., 2017) extend MDRNNs to tree structures and DAGs. PixelRNN (Van Den Oord et al., 2016) operates directly on pixel-level and improves MDRNNs for images. Due to their strictly recurrent nature, these architectures are not parallelizable and therefore unsuitable for modern hardware. Moreover, they lack the associative memory component (state expansion) of modern linear RNNs, making them less powerful.

Linear RNNs on Multi-Dimensional data VisionMamba (ViM) (Zhu et al., 2024) and Vision-LSTM (ViL) (Alkin et al., 2025) applied linear RNNs to the vision domain to challenge the common Vision Transformer (ViT) with its quadratic scaling (Dosovitskiy et al., 2021). These works rely on traversing the 2D plane in a predefined order to accommodate their sequential nature and often employ flipping the traversal order across layers. This was also investigated further in Mamba-ND (Li et al., 2025). Recent works extend the Mamba architecture to two dimensions. Most related to our work are 2DMamba (Zhang et al., 2024a) and V2M (Wang et al., 2024), which cover only the D-mode of pLSTM, at the loss of directional propagation. Mamba2D (Baty et al., 2024) and 2-D SSM (Baron et al., 2024) cover the P-mode of pLSTM, but in contrast to pLSTM, they are limited to the diagonal line instead of input-directed propagation along any line because of their normalization factor of $1/2$ (see Appendix F.2).

Relation to GNNs and MPNNs - Since pLSTM propagates recurrent states along a (directed acyclic) graph, it can be seen as a form of Graph Neural Network (GNN) (Scarselli et al., 2009) that covers a whole connected component of a DAG per layer instead of just a one-hop neighborhood.

D. Method Details

D.1. Loosely self-similar graphs

Graphs modeling hierarchical order or spatial extensions, as in meshes or grids do typically have low node degree and can be decomposed recursively into subgraphs of similar structure. We use the term loosely self-similar graphs for these. In particular, a graph $\mathcal{G} := \mathcal{G}^{(L)}$ should be decomposable into M subgraphs $\{\mathcal{G}_a^{(L-1)} = (N_a^{(L-1)}, E_a^{(L-1)})\}_{a\in\{1..M\}}$, and boundary edges $B_{ab}^{(L-1)} \subset N_a^{(L-1)} \times N_b^{(L-1)}$ such that $\mathcal{G} = \mathcal{G}^{(L)} = \left(\bigcup_a N_a^{(L-1)}, \bigcup_{a,b} B_{ab}^{(L-1)} \cup \bigcup_a E_a^{(L-1)}\right)$. This decomposition should be applicable recursively down to single nodes, such that $|N_\alpha^{(0)}| = 1 \quad \forall\, \alpha$ with $\alpha = (a_{L-1}...a_0)$ the multi-index of successive decompositions. Ideally, all decompositions should be balanced, $|N_\alpha^{(l)}| - |N_\beta^{(l)}| \in \{-1, 0, 1\}\ \forall\, l, \alpha, \beta$, for optimal parallelization - which should serve as the definition of loosely self-similar. Choosing arbitrary subsets of nodes for subgraphs, one can always arrive at a decomposition of any graph, but the edges across subgraphs might not be balanced.

D.2. Hierarchical Parallelization

Given a decomposition of a (loosely self-similar) DAG, we can apply a divide-and-conquer strategy to the expensive Transition sum/product combination. We want to calculate now the $G^{n'n}$ matrix entry for two nodes $n', n \in N; n' \leqslant n$, such that, at level $l+1$, both are in the subgraph $\mathcal{G}_\alpha^{l+1} : n', n \in N_\alpha^{l+1}$ but going one level lower, they are in distinct subgraphs $n' \in N_{\alpha:a}^l, n \in N_{\alpha:b}^l\ a \neq b$. With incoming and outgoing boundary edges $e' \in B_{\rightarrow\alpha:a}^l, e \in B_{\leftarrow\alpha:a}^l$, we define the generalized Source, Transition, and Mark tensors at level l:

$$S_e^{n',l,\alpha:a} := \sum_{e'\in E_\leftarrow(n')} S_{e'}^{n'} \sum_{P\in\mathcal{P}(e',e)} \prod_{n^P\in P} T^n_{e_\rightarrow^{n^P} e_\leftarrow^{n^P}} \tag{10}$$

$$T_{e'e}^{l,\alpha:k} := \sum_{P\in\mathcal{P}(e',e)} \prod_{n^P\in P} T^n_{e_\rightarrow^{n^P} e_\leftarrow^{n^P}} \tag{11}$$

$$M_{e'}^{n,l,\alpha:b} := \sum_{e\in E_\rightarrow(n)} \left(\sum_{P\in\mathcal{P}(e',e)} \prod_{n^P\in P} T^n_{e_\rightarrow^{n^P} e_\leftarrow^{n^P}} \right) M_e^n \tag{12}$$

Given these, the gating matrix entry can be calculated as:

$$G^{n'n} = \sum_{e \in B^l_{\leftarrow\alpha:a}} \sum_{e' \in B^l_{\rightarrow\alpha:j}} S_e^{n',l,\alpha:a} \left(\sum_{P^l \in \mathcal{P}^l(e',e)} \prod_{c \in P^l} T^{l,\alpha:c}_{e^{l,\alpha:c}_{\rightarrow m^l_c} e^{l,\alpha:c}_{\leftarrow m^l_c}} \right) M_{e'}^{n',l,\alpha:b}$$

Here, $\mathcal{P}^l$ is the set of "meta-paths" that connect two edges e', e via subgraphs at level l, in these meta-paths the subgraphs act as "nodes" and the boundary edges $B^l_{\leftarrow\alpha:a}$, $B^l_{\rightarrow\alpha:a}$ as "edges". Note that, by Einstein sum convention we implicitly sum over multiple in-coming and out-going edges $e^{l,\alpha:c}_{\rightarrow m^l_c}$ and $e^{l,\alpha:c}_{\leftarrow m^l_c}$ for generalized Transitions $T^{l,\alpha:c}$ in $G^{n'n}$ and in the following. See Figure 3 for an illustration of this decomposition. In reverse, this means that higher-level Source, Transition, and Mark can be constructed recursively bottom-up from one level lower (a chosen such that $n' \in N^l_{\alpha:a}$ or $n \in N^l_{\alpha:a}$):

$$S_e^{n',l+1,\alpha} = \sum_{e' \in B^l_{\leftarrow}\alpha:a} S_{e'}^{n',l,\alpha:a} \sum_{P^l \in \mathcal{P}^l(e',e)} \prod_{b \in P^l} T^{l,\alpha:b}_{e^{l,\alpha:b}_{\rightarrow m^l_b} e^{l,\alpha:b}_{\leftarrow m^l_b}} \tag{13}$$

$$T_{e'e}^{l+1,\alpha} = \sum_{P^l \in \mathcal{P}^l(e',e)} \prod_{b \in P^l} T^{l,\alpha:b}_{e^{l,\alpha:b}_{\rightarrow m^l_b} e^{l,\alpha:b}_{\leftarrow m^l_b}} \tag{14}$$

$$M_{e'}^{n,l+1,\alpha} = \sum_{e \in B^l_{\rightarrow}\alpha:a} \left(\sum_{P^l \in \mathcal{P}^l(e',e)} \prod_{b \in P^l} T^{l,\alpha:b}_{e^{l,\alpha:b}_{\rightarrow m^l_b} e^{l,\alpha:b}_{\leftarrow m^l_j}} \right) M_e^{n,l,\alpha:a} \tag{15}$$

Note that using this hierarchy only up to a certain level, one can optimize the parallelization to the used hardware. Whereas first, all the Source, Transition, Mark, and Gating matrix objects are constructed in parallel up to a certain level, the recurrent mode of Equation 1 is applied with generalized Source, Transition and Mark matrices of level l, in the topological ordering of the subgraphs at this level. This resembles the structure of the chunk-wise parallel formulation for linear RNNs.

D.3. Parallelization of pLSTM in 2D - images

See Section D.4 for the definition of the notational structure.

$$S^{l+1,\mapsto}_{xyuwa} = \Big[\big[[S^{l,\mapsto}_{(2x)(2y)(u\rfloor_0)(w\rfloor_1)b} T^{l,\rightarrow}_{(2x+1)(2y)b(a\rfloor_2)}, S^{l,\mapsto}_{(2x+1)(2y)(u\rceil_0)(w\rceil_1)(a\rfloor_2)} \big]_0, [0_{(u\rfloor_0)(w\rceil_1)(a\rfloor_2)}, 0_{(u\rceil_0)(w\rceil_1)(a\rceil_2)} \big]_0 \Big]_1, \tag{16}$$

$$\Big[\big[[S^{l,\mapsto}_{(2x)(2y)(u\rfloor_0)(w\rfloor_1)b} T^{l,\searrow}_{(2x+1)(2y)bc} T^{l,\searrow}_{(2x+1)(2y+1)c(a\rceil_2)} + S^{l,\mapsfrom}_{(2x)(2y)(u\rfloor_0)(w\rceil_1)b} T^{l,\searrow}_{(2x)(2y+1)bc} T^{l,\rightarrow}_{(2x+1)(2y+1)c(a\rceil_2)},$$
$$S^{l,\mapsfrom}_{(2x+1)(2y)(u\rceil_0)(w\rceil_1)b} T^{l,\searrow}_{(2x+1)(2y+1)bc} \big]_0, [S^{l,\mapsto}_{(2x)(2y+1)(u\rceil_0)(w\rceil_1)b} T^{l,\rightarrow}_{(2x+1)(2y+1)b(a\rceil_2)}, S^{l,\mapsto}_{(2x+1)(2y+1)(u\rceil_0)(w\rceil_1)(a\rceil_2)} \big]_0 \Big]_1 \Big]_2 \tag{17}$$

$$
\begin{aligned}
S^{l+1,\mathrlap{\shortparallel}\downarrow}_{xyuwa} = \Bigg[& \Big[\big[S^{l,\mathrlap{\shortparallel}\downarrow}_{(2x)(2y)(u\rfloor_0)(w\rfloor_1)b} T^{l,\downarrow}_{(2x)(2y+1)b(a\rfloor_2)}, 0_{(u\rceil_0)(w\rceil_0)(a\rfloor_2)} \big]_0, \big[S^{l,\mathrlap{\shortparallel}\downarrow}_{(2x)(2y+1)(u\rfloor_0)(w\rceil_1)(a\rfloor_2)}, 0_{(u\rceil_0)(w\rceil_0)(a\rfloor_2)} \big]_0 \Big]_1, \\
& \Big[\big[S^{l,\mapsto}_{(2x)(2y)(u\rfloor_0)(w\rfloor_1)b} T^{l,\searrow}_{(2x+1)(2y)bc} T^{l\downarrow}_{(2x+1)(2y+1)c(a\rceil_2)} + S^{l,\mathrlap{\shortparallel}\downarrow}_{(2x)(2y)(u\rfloor_0)(w\rfloor_1)b} T^{l,\hookrightarrow}_{(2x)(2y+1)bc} T^{\searrow}_{(2x+1)(2y+1)c(a\rceil_2)}, \\
& S^{l,\mathrlap{\shortparallel}\downarrow}_{(2x+1)(2y)(u\rceil_0)(v\rfloor_1)b} T^{l,\downarrow}_{(2x+1)(2y+1)b(a\rceil_0)} \big]_0, \big[S^{l,\mapsto}_{(2x)(2y+1)(u\rfloor_0)(w\rceil_1)b} T^{l,\searrow}_{(2x+1)(2y+1)b(a\rceil_2)}, S^{l,\mathrlap{\shortparallel}\downarrow}_{(2x+1)(2y+1)(u\rceil_0)(v\rceil_1)(a\rceil_2)} \big]_0 \Big]_1 \Bigg]_2
\end{aligned}
\tag{18}
$$

$$
\begin{aligned}
T^{l,\rightarrow}_{xyab} = \Big[& \big[T^{l,\rightarrow}_{(2x)(2y)(a\rfloor_0)c} T^{l,\rightarrow}_{(2x+1)(2y)c(b\rfloor_1)}, 0_{(a\rceil_0)(b\rfloor_1)} \big]_0, \\
& \big[T^{l,\downarrow}_{(2x)(2y)(a\rfloor_0 c)} T^{l,\hookrightarrow}_{(2x)(2y+1)cd} T^{l,\rightarrow}_{(2x+1)(2y+1)d(b\rceil_1)} + T^{l,\rightarrow}_{(2x)(2y)(a\rfloor_0)c} T^{l,\searrow}_{(2x+1)(2y)cd} T^{l,\hookrightarrow}_{(2x+1)(2y+1)d(b\rceil_1)}, \\
& T^{l,\rightarrow}_{(2x)(2y+1)(a\rceil_0)c} T^{l,\rightarrow}_{(2x+1)(2y+1)c(b\rceil_1)}, 0_{(a\rceil_0)(b\rfloor_1)} \big]_0 \Big]_1
\end{aligned}
\tag{19}
$$

$$
\begin{aligned}
T^{l,\downarrow}_{xyab} = \Big[& \big[T^{l,\downarrow}_{(2x)(2y)(a\rfloor_0)c} T^{l,\downarrow}_{(2x)(2y+1)c(b\rfloor_1)}, 0_{(a\rceil_0)(b\rfloor_1)} \big]_0, \\
& \big[T^{l,\downarrow}_{(2x)(2y)(a\rfloor_0 c)} T^{l,\hookrightarrow}_{(2x)(2y+1)cd} T^{l,\searrow}_{(2x+1)(2y+1)d(b\rceil_1)} + T^{l,\rightarrow}_{(2x)(2y)(a\rfloor_0)c} T^{l,\searrow}_{(2x+1)(2y)cd} T^{l,\downarrow}_{(2x+1)(2y+1)d(b\rceil_1)}, \\
& T^{l,\downarrow}_{(2x+1)(2y)(a\rceil_0)c} T^{l,\downarrow}_{(2x+1)(2y+1)c(b\rceil_1)}, 0_{(a\rceil_0)(b\rfloor_1)} \big]_0 \Big]_1
\end{aligned}
\tag{20}
$$

$$
\begin{aligned}
T^{l,\hookrightarrow}_{xyab} = \Big[& \big[T^{l,\hookrightarrow}_{(2x)(2y)(a\rfloor_0)c} T^{l,\rightarrow}_{(2x+1)(2y)c(b\rfloor_1)}, T^{l,\hookrightarrow}_{(2x+1)(2y)(a\rceil_0)(b\rfloor_1)} \big]_0, \\
& T^{l,\hookrightarrow}_{(2x)(2y)(a\rfloor_0)c} T^{l,\searrow}_{(2x+1)(2y)cd} T^{l,\hookrightarrow}_{(2x+1)(2y+1)d(b\rceil_1)} + T^{l,\downarrow}_{(2x)(2y)(a\rfloor_0)c} T^{l,\hookrightarrow}_{(2x)(2y+1)cd} T^{l,\rightarrow}_{(2x+1)(2y+1)d(b\rceil_1)}, \\
& T^{l,\downarrow}_{(2x+1)(2y)(a\rceil_0)c} T^{l,\hookrightarrow}_{(2x+1)(2y+1)c(b\rceil_1)} \big]_0 \Big]_1
\end{aligned}
\tag{21}
$$

$$
\begin{aligned}
T^{l,\searrow}_{xyab} = \Big[& \big[T^{l,\searrow}_{(2x)(2y)(a\rfloor_0)c} T^{l,\downarrow}_{(2x)(2y+1)c(b\rfloor_1)}, T^{l,\searrow}_{(2x)(2y+1)(a\rceil_0)(b\rfloor_1)} \big]_0, \\
& T^{l,\searrow}_{(2x)(2y)(a\rfloor_0)c} T^{l,\hookrightarrow}_{(2x)(2y+1)cd} T^{l,\searrow}_{(2x+1)(2y+1)d(b\rceil_1)} + T^{l,\rightarrow}_{(2x)(2y)(a\rfloor)_0 c} T^{l,\searrow}_{(2x+1)(2y)cd} T^{l,\downarrow}_{(2x+1)(2y+1)d(b\rceil_1)}, \\
& T^{l,\rightarrow}_{(2x)(2y+1)(a\rceil_0)c} T^{l,\searrow}_{(2x+1)(2y+1)c(b\rceil_1)} \big]_0 \Big]_1
\end{aligned}
\tag{22}
$$

$$
\begin{aligned}
M^{l,\mapsto}_{xyuva} = \Bigg[& \Big[\big[M^{l,\mapsto}_{(2x)(2y)(u\rfloor_0)(v\rfloor_1)(a\rfloor_2)}, T^{l,\rightarrow}_{(2x)(2y)(a\rfloor_2)b} M^{l,\mapsto}_{(2x+1)(2y)(u\rceil_0)(v\rfloor_1)b} \big]_0, \big[T^{l,\searrow}_{(2x)(2y)(a\rfloor_2)b} M^{l,\mathrlap{\shortparallel}\downarrow}_{(2x)(2y+1)(u\rfloor_0)(v\rceil_1)b}, \\
& T^{l,\searrow}_{(2x)(2y)(a\rfloor_2)b} T^{l,\hookrightarrow}_{(2x)(2y+1)bc} M^{l,\mapsto}_{(2x+1)(2y+1)(u\rceil_0)(v\rceil_1)c} + T^{l,\rightarrow}_{(2x)(2y)(a\rfloor_2)b} T^{l,\searrow}_{(2x+1)(2y)bc} M^{l,\mathrlap{\shortparallel}\downarrow}_{(2x+1)(2y+1)(u\rceil_0)(v\rceil_1)c} \big]_0 \Big]_1, \\
& \Big[\big[0_{(u\rfloor_0)(v\rfloor_1)(a\rceil_2)}, 0_{(u\rceil_0)(v\rfloor_1)(a\rceil_2)} \big]_0, \big[M^{l,\mapsto}_{(2x)(2y+1)(u\rfloor_0)(v\rceil_1)(a\rceil_2)}, T^{l,\rightarrow}_{(2x)(2y+1)(a\rceil_2)b} M^{l,\mapsto}_{(2x+1)(2y+1)(u\rceil_0)(v\rceil_1)b} \big]_0 \Big]_1 \Bigg]_2
\end{aligned}
\tag{23}
$$

$$
\begin{aligned}
M^{l,\mathrlap{\shortparallel}\downarrow}_{xyuva} = \Bigg[& \Big[\big[M^{l,\mathrlap{\shortparallel}\downarrow}_{(2x)(2y)(u\rfloor_0)(v\rfloor_1)(a\rfloor_2)}, T^{l,\hookrightarrow}_{(2x)(2y)(a\rfloor_2)b} M^{l,\mapsto}_{(2x+1)(2y)(u\rceil_0)(v\rfloor_1)b} \big]_0, \big[T^{l,\downarrow}_{(2x)(2y)(a\rfloor_2)b} M^{l,\mathrlap{\shortparallel}\downarrow}_{(2x)(2y+1)(u\rfloor_0)(v\rceil_1)b}, \\
& T^{l,\hookrightarrow}_{(2x)(2y)(a\rfloor_2)b} T^{l,\searrow}_{(2x+1)(2y)bc} M^{l,\mathrlap{\shortparallel}\downarrow}_{(2x+1)(2y+1)(u\rceil_0)(v\rceil_1)c} + T^{l,\downarrow}_{(2x)(2y)(a\rfloor_2)b} T^{l,\hookrightarrow}_{(2x)(2y+1)bc} M^{l,\mapsto}_{(2x+1)(2y+1)(u\rceil_0)(v\rceil_1)c} \big]_0 \Big]_1, \\
& \Big[\big[0_{(u\rfloor_0)(v\rfloor_1)}, M^{l,\mathrlap{\shortparallel}\downarrow}_{(2x+1)(2y)(u\rceil_0)(v\rfloor_1)} \big]_0, \big[0_{(u\rfloor_0)(v\rceil_1}, T^{l,\downarrow}_{(2x+1)(2y)(a\rceil_2)b} M^{l,\mathrlap{\shortparallel}\downarrow}_{(2x+1)(2y+1)(u\rceil_0)(v\rceil_1)b} \big]_0 \Big]_1 \Bigg]_2
\end{aligned}
\tag{24}
$$

D.4. Parallelization of pLSTM in 1D - sequences

For a sequence as a DAG, pLSTMs reduce to linear RNNs (with slightly modified gating). Here, all nodes and subgraphs have a single incoming and outgoing (boundary) edge. For parallelization, the decomposition is done hierarchically in powers-of-two splits. The S, T, M and G tensors are constructed as follows:

$$S^l_{an} = \left[S^{l-1}_{(2a)(n\rfloor_0)} T^{l-1}_{(2a+1)}, S^{l-1}_{(2a+1)(n\rceil_0)}\right]_0 \tag{25}$$

$$T^l_a = T^{l-1}_{(2a)} T^{l-1}_{(2a+1)} \tag{26}$$

$$M^l_{an} = \left[M^{l-1}_{(2a+1)(n\rfloor_0)}, T^{l-1}_{(2a)} M^{l-1}_{(2a+1)(n\rceil_0)}\right]_0 \tag{27}$$

$$G^l_{an'n} = \Big[\left[G^l_{(2a)(n'\rfloor_0)(n\rfloor_1)}, 0\right]_0, \left[S^{l-1}_{(2a)(n'\rfloor_0)} M^{l-1}_{(2a+1)(n\rceil_1)}, G^l_{(2a+1)(n'\rceil_0)(n\rceil_1)}\right]_0\Big]_1 \tag{28}$$

with a the external index (Source index $\in \{0..S/2^l - 1\}$ of level l), n the internal index (node index $\in \{0..2^l - 1\}$ within generalized Source S^l_a). Here, $[]_i$ denotes a concatenation along the axis i, and $\rfloor_i$ and $\rceil_i$ denote these indices are used as axis i for this concatenation - one taking the lower half, one the upper half, e.g. for vectors $\boldsymbol{a} \in \mathbb{R}^N$; $\boldsymbol{b}, \boldsymbol{c} \in \mathbb{R}^{N/2}$:

$$a_n = \left[b_{(n\rfloor_0)}, c_{(n\rceil_0)}\right]_0 := \begin{cases} b_n & n < N/2 \\ c_{n-N/2} & n \geq N/2 \end{cases} \tag{29}$$

Note that we use a causal setting here. For a non-causal structure, one would apply this bi–bidirectionally, effectively filling the top-right entries of the gating matrix with a different Source/Mark combination for the opposite order. Equivalently, both directions can be added in their outputs while computing the parallel parts twice.

D.5. Stability Derivation

The line graph of a DAG $\mathcal{G} = (N, E)$ is the DAG $\mathcal{G}' = (E, E')$ formed by the edges E of the original DAG and the line edges $e' \in E' \iff e' = (e_1, e_2), \exists\, n \in N : e_1 \in E_{\rightarrow}(n) \cap e_2 \in E_{\leftarrow}(n)$, which connect two original edges if these are connected by a node in $\mathcal{G}$ (Whitney, 1932). Given the previous definitions of Cell states C_e and Transitions $T^n_{e'e}$, these can be interpreted as states and edges on the nodes of the line graph $\mathcal{G}'$. The Transitions $T_{e'e}$ can be viewed as entries of the adjacency matrix $\boldsymbol{T}$ of $\mathcal{G}'$. This way, we can apply the theory of power iterations and matrix norms. The application of a pLSTM of Equation 1 is equivalent to the application of the power iteration and can be bounded by compatible matrix norms $|\cdot|$:

$$|\boldsymbol{c}| = |\left(\sum_{p=0}^{\infty} \boldsymbol{T}^p\right)\boldsymbol{s}| = |\left(\sum_{p=0}^{P_{\mathcal{G}'}} \boldsymbol{T}^p\right)\boldsymbol{s}| \overset{|\boldsymbol{T}|\leqslant 1}{\leqslant} P_{\mathcal{G}'}|\boldsymbol{s}| \tag{30}$$

As $\mathcal{G}$ is a DAG, also its line graph $\mathcal{G}'$ is a DAG and this one's adjacency matrix $T_{e'e}$ is nilpotent, for a certain power $\exists P_{\mathcal{G}'} \leqslant |N| : \forall p > P_{\mathcal{G}'} : \boldsymbol{T}^p = \boldsymbol{0}$. For simplicity, we use matrix-vector notation here for Cell states $\boldsymbol{c}$, Transitions $\boldsymbol{T}$, and Sources $\boldsymbol{s}$ - without key-value extension.

Stabilization of the P-mode The normalization of Equation 4 limits the respective column e' in the line graph adjacency matrix $\boldsymbol{T}$, and in turn limits its L_1-norm $||\boldsymbol{T}||_1$. Since the L_1-norm is sub-multiplicative (Horn & Johnson, 1985), this can be applied to the matrix powers in Equation 30 and keeps the L_1-norms of the Cell states C_e limited. Keeping the norm fixed at one exactly enables long-range propagation (e.g., by division of the column entries by the norm). This also stabilizes the gradient propagation with an L_∞ bound, as gradients are propagated backwards and the applied transposed adjacency matrix $\boldsymbol{T}^T$ is row-(sub)-normalized and therefore L_∞ bounded (the dual norm to the L_1 norm).

D.6. State-Tracking Extension

Recent research has shown that having scalar, positive Transitions $T^n_{e_{\rightarrow}e_{\leftarrow}}$ does not allow for state tracking capabilities in these linear RNNs (Merrill et al., 2024; Grazzi et al., 2025). We can generalize the Transition scalars to Transition matrices here as well: $T^n_{e^n_{\rightarrow}e^n_{\leftarrow}} \rightarrow T^n_{e^n_{\rightarrow}e^n_{\leftarrow}ij}$ $i, j \in \{1..J_T\}$, J_T being the Transition state tracking dimension. Note that in this way, the

Source and Mark matrix have to include this extended dimension as well. We can define generalizations with additional state tracking dimensions J_S, J_M:

$$S_e^{n',l,\alpha} \rightarrow S_{eki}^{n',l,\alpha} \qquad k \in \{1..J_S\}, i \in \{1..J_T\} \tag{31}$$

$$T_{e'e}^{l,\alpha} \rightarrow T_{e'eij}^{l,\alpha} \qquad i, j \in \{1..J_T\} \tag{32}$$

$$M_{e'}^{n,l+1,\alpha} \rightarrow M_{e'jm}^{n,l,\alpha} \qquad j \in \{1..J_T\}, m \in \{1..J_M\} \tag{33}$$

$$G^{n'n} \rightarrow G_{km}^{n'n} \qquad k \in \{1...J_S\}, m \in \{1..J_M\} \tag{34}$$

This extension includes specific parameterizations such as defining the Transition matrix as $T_{ij} = 1 + \beta k_i k_j$ known from DeltaNet (Schlag et al., 2021). For these, the chunk-wise parallelization of Yang et al. (2024) is applicable along multiple dimensions.

D.7. Stability in the State Tracking Extension

The stabilization of Section D.6 can be applied in the state tracking extensions as well. In the extended case, the absolute values of Transitions are replaced by the spectral norm of the Transition matrix $(T_{e'eij})_{ij}$ defined by its largest singular value. The stability can be ensured by limiting the sum of the largest singular values (instead of scalar entries) for the P-mode or zero matrix entries, resulting in a line graph tree with maximally unit norm matrices for tree Transitions in the D-mode. There are several options to limit or set the Transition matrices by/to one in magnitude. A straightforward option is to parametrize not the Transition entries T_{ij} directly, but to parametrize its singular value decomposition, with orthogonal (or unitary) matrices $\boldsymbol{T} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$. The singular values $\boldsymbol{\Sigma}$ can be limited in magnitude by a $\tanh$ (for multiplicative decay) or fixed to ± 1 (long-range propagation). In the case of multiple directions (see 2D grid case), in the P-mode, they are multiplied by an additional softmax-limited propagation factor ("angle") - limiting the sum over the direction pairs in Equation 4. The orthogonal matrices $\boldsymbol{U}, \boldsymbol{V}$ can be parametrized by the product of Householder matrices (generated from vectors), or the exponential of the Lie-group / generating group of special orthogonal matrices: the skew-symmetric matrices (directly parameterized). With these parametrizations, in turn depending on the network inputs (at nodes), state-tracking capabilities can be achieved while maintaining long-range stability (Merrill et al., 2024; Grazzi et al., 2025).

D.8. Chunkwise-Parallel Form

Given the hierarchical structure of the parallelization shown in Sections D.2, D.4, and D.3, one can stop at a certain level of it. At this level, the resulting objects will again form a higher-level DAG or grid, which can now be processed sequentially, in the topological ordering of this DAG. This way, the quadratic complexity introduced by the parallelization is only introduced up to the optimal level of the hardware's parallelization capabilities. While typically the FLOP-optimal point is at lower parallelization, Beck et al. (2025a) showed that a certain degree of parallelization is hardware optimal for linear RNNs.

Multi-Directional Form For the 2D grid case of images, we use pLSTM in all four direction combinations of the 2D DAG at once in parallel, at each node going down/right, down/left, up/right, up-left. By pre-computing all of them in full parallelization, we can add up their gating matrices, resulting in just a single pass of the gated linear attention computation (see Equation 3). We use this fully parallelized form for the vision experiments.

E. Full Model Architecture

E.1. pLSTM for Vision

For the pLSTM-Vis, we use the ViT (Dosovitskiy et al., 2021) with pre-LayerNorm (actually RMSNorm) structure as backbone architecture, replacing the Multi-Head Attention with a pLSTM layer. Note that in addition to query-, key-, value-, and out-projections, we have linear layers for Source, Transition, and Mark gates in the four direction combinations. In addition, we apply a multi-head RMSNorm after the pLSTM, as common for linear RNNs (Yang et al., 2023; Beck et al., 2025b). Note that the Source, Mark, and Direct gates are sigmoid-activated, and the Transition is tanh-activated. For details, we refer to the attached source code.

F. Theoretical Analysis

F.1. Exponential growth of path counts in a 2D grid

Assume a 2D grid structure of a DAG, then the number of paths between two nodes can be calculated in the following way. In total, the number of Transitions to the right and Transitions to the bottom are fixed by the position offsets Δ_x and Δ_y between the nodes. In total, the number of paths can be counted by the number of orderings of a string consisting of Δ_x times $\rightarrow$ and Δ_y times $\leftarrow$. This results in the combinatorial factor, which is exponential in the path length $\Delta_x + \Delta_y =: \Delta$:

$$\begin{aligned}\#\text{Paths} &= \binom{\Delta_x + \Delta_y}{\Delta_x} = \frac{(\Delta_x + \Delta_y)!}{\Delta_x!\Delta_y!} \overset{\text{Stirling}}{\approx} \sqrt{\frac{(\Delta_x + \Delta_y)}{2\pi\Delta_x\Delta_y}} \frac{(\Delta_x + \Delta_y)^{\Delta_x+\Delta_y}}{\Delta_x^{\Delta_x}\Delta_y^{\Delta_y}} \\ &\overset{\Delta_x := \beta\Delta}{=} \sqrt{\frac{1}{2\pi\Delta\beta(1-\beta)}} \underbrace{\left(\beta^{-\beta}(1-\beta)^{-(1-\beta)}\right)^{\Delta}}_{\geqslant 1}\end{aligned} \tag{35}$$

Given Transitions / forget gates potentially all at 1, this exponential number of paths leads to an exponentially growing magnitude of the Cell state C, as of Equation 3. For a non-linear MD-RNN, like DAG-LSTM, this path count applies as well, assuming that all forget gates are fixed to one. Also in this case, the cell states accumulate all their ancestors' values via all paths.

F.2. Long-Range Decay for P-mode in 2D

As the P-mode offers a directional propagation, here, we want to calculate the decay of one Source signal along the leading direction - given all Transitions are equal within the grid and at criticality: $T = \begin{bmatrix} \alpha & \alpha \\ (1-\alpha) & (1-\alpha) \end{bmatrix}$. This results in the following overall Transition from Source S_{xy} at x, y to Mark $M_{(x+\Delta_x)(y+\Delta_y)}$ at $x + \Delta_x$, $y + \Delta_y$ - with path length $\Delta = \Delta_x + \Delta_y \gg 1$, direction $\beta := \frac{\Delta_x}{\Delta}$:

$$T^{\text{full}}_{xy(x+\Delta_x)(y+\Delta_y)} = \sum_{\text{Paths}} T^{\Delta_x}_{r*} T^{\Delta_y}_{d*} = \binom{\Delta_x + \Delta_y}{\Delta_x} \alpha^{\Delta_x}(1-\alpha)^{\Delta_y} = \binom{\Delta}{\beta\Delta} \alpha^{\beta\Delta}(1-\alpha)^{(1-\beta)\Delta} \tag{36}$$

Notice the binomial structure of the equation that fixes the total sum $\sum_{\Delta_x \in \{0..\Delta\}} T^{\text{full}}_{xy(x+\Delta_x)(y+\Delta_y)}$ to 1, so the total activation is conserved along the diagonal. Differentiating this equation by β, we can get the direction of largest propagation:

$$0 \overset{!}{=} \partial_\beta \log T^{\text{full}}_{xy(x+\Delta_x)(y+\Delta_y)} = -\partial_\beta \left(\log\Gamma(\Delta+1) + \log\Gamma(\beta\Delta+1) + \log\Gamma((1-\beta)\Delta+1)\right) + \Delta\log(\alpha) - \Delta\log(1-\alpha) \tag{37}$$

$$\begin{aligned}&\overset{\text{Stirling}}{\approx} -\partial_\beta \left(\beta\Delta\log(\beta) + (1-\beta)\Delta\log(1-\beta)\right) + \Delta\log(\alpha) - \Delta\log(1-\alpha) \\ &= \Delta\log\left(\frac{\alpha(1-\beta)}{\beta(1-\alpha)}\right)\end{aligned}$$

This implies $\beta = \alpha$ for the direction of largest propagation. Now, inserting this direction into the Transition product:

$$\begin{aligned}T^{\text{full}}_{xy(x+\Delta_x)(y+\Delta_y)}\big|_{\beta=\alpha} &= \binom{\Delta}{\alpha\Delta} \alpha^{\alpha\Delta}(1-\alpha)^{(1-\alpha)\Delta} \overset{\text{Stirling}}{\approx} \frac{\sqrt{2\pi\Delta}\left(\frac{\Delta}{e}\right)^{\Delta} \alpha^{\alpha\Delta}(1-\alpha)^{(1-\alpha)\Delta}}{\sqrt{2\pi\alpha\Delta}\left(\frac{\alpha\Delta}{e}\right)^{\alpha\Delta}\sqrt{2\pi(1-\alpha)\Delta}\left(\frac{(1-\alpha)\Delta}{e}\right)^{(1-\alpha)\Delta}} \\ &= \sqrt{\frac{1}{2\pi\alpha(1-\alpha)\Delta}}\end{aligned} \tag{38}$$

So, even in the critical case of the P-mode, the signal is decaying, but only as a power law $O(\Delta^{-1/2})$ of the path length.

G. Experimental Results

Due to the complex structure of pLSTM in 2D and its parallelization, we use a `jax` [1]-based implementation to enable efficient compilation of the computational graph. In particular, we re-use parts of Park (2024). Early experiments on using `torch.compile` on the `torch` [2] implementation showed a slowdown rather than a speed-up of the model computations, which is why we use `jax`. Our source code, released with this work, has a detached configuration that works for both `jax` and `torch` and should enable a fast switch of frameworks for future changes. We use `jax` version `0.4.32` and CUDA `12.2`. For the detailed library versions we refer to the attached source code.

G.1. Initialization

We use small source, mark and direct bias initialization as suggested in Beck et al. (2025a) for linear RNNs.

Table 2: General pLSTM initialization settings

Parameter	Value (→ Fine-Tuning)
Source Bias Init	-4
Mark Bias Init	-4
Direct Bias Init	-6
Transition Bias Init	1
Transition Scaling Factor	5
Orientation Bias Init	Headwise Range in [-2, 2]
Multi-Head RMSNorm ϵ	1e-5
Pooling	Corner Patches
Mode	P+D (alternating)

G.2. Arrow Pointing

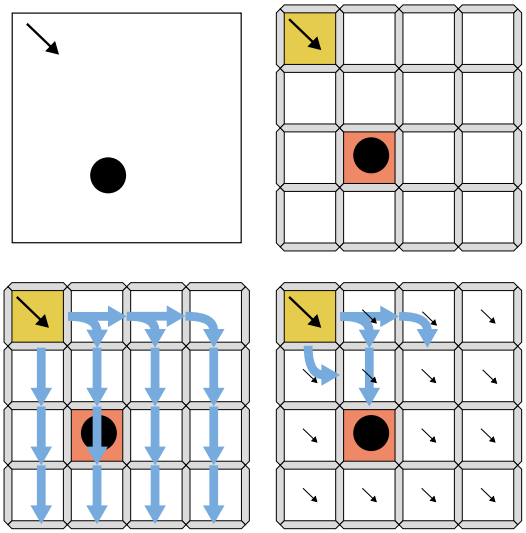

Figure 5: Illustration of the **Arrow pointing task**: the model has to classify whether an arrow is pointing towards a circle (top left). Models with global receptive fields, such as Vision Transformers (ViTs), can solve this task by leveraging directional information encoded via positional embeddings (top right), but they often struggle to generalize to higher resolutions. In contrast, pLSTMs can effectively solve this task in both diffusive distribution (D-mode, bottom left) and directed propagation (P-mode, bottom right) modes, enabling long-range reasoning and better scalability.

[1] https://jax.dev
[2] https://pytorch.org

Here, we train for 50 epochs on 100k training images with batch size 128, using learning rates [1e-4, 3e-4, 1e-3] and report the mean validation curves over five seeds at the best learning rate. For ViL, we include 1e-5 as the learning rate, as it fails to improve for higher learning rates. The validation datasets (standard and extrapolation) are generated from the same validation seed for all runs. We use a linear-warmup + cosine-decay schedule with one warmup epoch starting from zero and ending at $0.001 \times$ peak_lr. The models take about one hour of training on a single NVIDIA H100-64GB. See Figure 6 and Table 3 for an additional ablation of pLSTM variants.

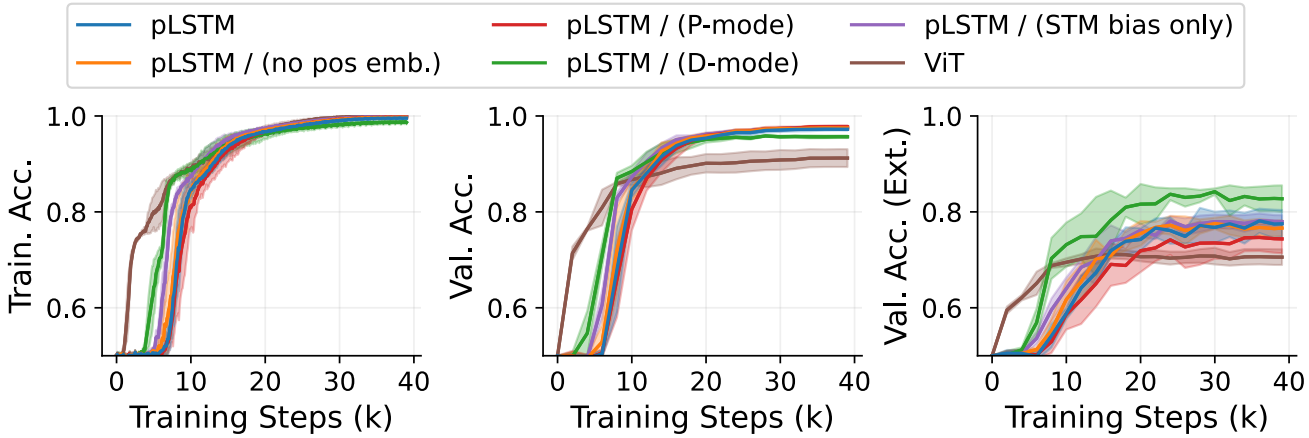

Figure 6: Training curves for the **Arrow Pointing Extrapolation task**, averaged over 5 seeds with 90% CI on different model ablations. P-mode by itself performs worse, D-mode by itself is not as general in interpolation, but performs better on extrapolation compared to other models. pLSTM does not rely on the positional embedding for strong performance.

Table 3: Test Results on Arrow Pointing Extrapolation (5 seeds, 90% confidence interval)

Model	Best LR	Test Acc.	Test Acc. (Ext.)
pLSTM	0.0001	0.972 ± 0.003	0.778 ± 0.031
pLSTM / (no posemb.)	0.0001	0.975 ± 0.003	0.769 ± 0.018
pLSTM / (P-mode only)	0.0003	0.978 ± 0.005	0.746 ± 0.027
pLSTM / (D-mode only)	0.0001	0.957 ± 0.002	0.828 ± 0.028
pLSTM / (STM bias only)	0.0001	0.975 ± 0.003	0.784 ± 0.020
ViT	0.0003	0.915 ± 0.019	0.707 ± 0.014

G.3. ImageNet-1k

To test for real-world image model capabilities, we train pLSTM using the schedule of DeiT-III (Touvron et al., 2022), comparing our architecture to other vision models. pLSTM performs similar to other popular approaches. With the integration of local features, e.g., including additional convolutions such as in Vision LSTM (Alkin et al., 2025), we see room to narrow the gap to SOTA. For a detailed discussion on the results and relation to previous work, see Appendix G.3.

G.4. ImageNet-1k Ablation

We ablate our pLSTM and compare against a ViT trained using the ViT-T model scale, using a simpler single pre-training schedule close to DeiT (Touvron et al., 2021) at 224×224 resolution (see Appendix G.4.1) for details. In our re-training, pLSTM can outperform ViT by a significant margin.

Setup For training on ImageNet-1k, we match the original training schedule of DeiT-III (Touvron et al., 2022). EfficientNets (Tan & Le, 2019) as a convolution-based baseline architecture are still the SOTA at these scales, but it is important to

[2] https://github.com/facebookresearch/fvcore

Table 4: ImageNet-1K pre-training accuracy. All models use a patch size of 16×16 with 224×224 resolution at maximum. Models with "+" in their "Epochs" column pre-train on lower resolution, followed by fine-tuning on 224×224 resolution for some epochs. Values of reference models are taken from Alkin et al. (2025) and the original work for 2DMamba, Mamba2D and EfficientNet (Tan & Le, 2019; Zhang et al., 2024b; Baty et al., 2024). Note that EfficientNet, as a CNN-based baseline, still outperforms the more recent baselines, but was also trained on larger resolutions for the scaled-up versions. Due to the chunkwise-recurrent option, pLSTM FLOPs could be optimized further (see Appendix D.8).

Model	Epochs	#Params	FLOPS	IN-1K
EfficientNet-B0 (Tan & Le, 2019)	?	5M	0.39G	77.1
DeiT-T (Touvron et al., 2021)	300	6M	1.3G	72.2
DeiT-III-T (reimpl.) (Touvron et al., 2022)	800+20	6M	1.1G	75.4
VRWKV-T (Duan et al., 2024)	300	6M	1.2G	75.1
Vim-T (Zhu et al., 2024)	300	7M	1.5G	76.1
ViL-T (Alkin et al., 2025)	800+20	6M	1.3G	**78.3**
pLSTM-Vis-T	800+20	6M	1.4G	75.2
EfficientNet-B4 (Tan & Le, 2019)	?	19M	1.8G	**82.9**
DeiT-S (Touvron et al., 2021)	300	22M	4.6G	79.8
DeiT-III-S (reimpl.) (Touvron et al., 2022)	400+20	22M	4.6G	80.3
ConvNeXt-S (*iso.*) (Liu et al., 2022)	300	22M	4.3G	79.7
VRWKV-S (Duan et al., 2024)	300	24M	4.6G	80.1
Vim-S (Zhu et al., 2024)	300	26M	5.3G	80.5
Mamba2D-T (Baty et al., 2024)	300	27M	-	82.4
2DMamba-T (Zhang et al., 2024a)	?	30M	4.9G	82.8
ViL-S (Alkin et al., 2025)	400+20	23M	4.7G	81.5
pLSTM-Vis-S	400+20	23M	4.9G	80.7
EfficientNet-B6 (Tan & Le, 2019)	?	43M	19G	**84.0**
DeiT-B (Touvron et al., 2021)	300	86M	17.6G	81.8
DeiT-III-B (reimpl.) (Touvron et al., 2022)	400+20	87M	16.8G	83.5
ConvNeXt-B (*iso.*) (Liu et al., 2022)	300	87M	16.9G	82.0
VRWKV-B (Duan et al., 2024)	300	94M	18.2G	82.0
2DMamba-S (Zhang et al., 2024a)	?	50M	8.8G	83.8
ViL-B (Alkin et al., 2025)	400+5	89M	17.9G	82.4
pLSTM-Vis-B	400+20	89M	18.2 G	82.5

note that larger models were also trained at larger resolutions, whereas training resolution was not scaled with the models for all other models. EfficientNet, Mamba2D, and 2DMamba are non-isotropic, in that their embedding dimensions are scaled up with depth. ViT, ViL, and pLSTM are isotropic as is the reported isotropic ConvNeXt. For ConvNeXt, isotropic models showed lower performance compared to non-isotropic ones. For pLSTM, a transition to non-isotropic model could therefore lead to performance gains as well.

All of the models are pre-trained for up to 24 hours on 4 nodes, with 4 NVIDIA H100-64GB GPUs each. ViT models are faster (about 30-50%), as our models do not yet utilize specific kernels. For counting FLOPs, we use `fvcore`[3] on the PyTorch implementation. For arrow pointing extrapolation training, the Mamba2D and 2DMamba variants were about twice as slow as pLSTM despite utilizing custom kernels.

G.4.1. Ablation Settings

For the ablation studies, we use a simplified training setting, without an additional fine-tuning stage, resembling a DeiT-T training over 400 epochs (Touvron et al., 2021). In Table 7, we provide a summary of the used training hyperparameters.

G.5. Graph Benchmarks

For the graph version of pLSTM, we use `torch`, as the dynamic computation graph support is better for this framework. For simplicity, we also do not implement any parallelization of the graph computation, but use the recurrent form only.

[3] https://github.com/facebookresearch/fvcore

Table 5: Ablation of pLSTM variants and ViT (DeiT) re-training on **ImageNet-1k**.

Model	ImageNet-1k top-1
pLSTM	**75.51**
pLSTM / (no posemb.)	75.22
pLSTM / (P-mode only)	74.86
pLSTM / (D-mode only)	75.13
pLSTM / (STM bias only	75.10
ViT	74.66

Table 6: ImageNet1k - DeiT-III style training parameters

Parameter	Value (→ Fine-Tuning)
Image Resolution	192 × 192 → 224 × 224
Training Epochs	800 (T), 400 (S), 400 (B) → 20
Hidden Dimension	192 (T), 384 (S), 768 (B)
Num Heads	3 (T), 6 (S), 12 (B)
DropPath Rate	0. (T), 0.05 (S), 0.1 (B)
LayerScale	-
Warmup Epochs	5
Peak Learning Rate	4e-3 (T), 4e-3 (S), 3e-3 (B) → 1e-5
Weight Decay	0.2
Gradient Clip Norm	1.0
Optimizer	Lamb → AdamW
Loss Type	Binary Cross Entropy → Cross Entropy
MixUp	0.8
CutMix	1.0
Label Smoothing	0.0 → 0.1
Global Batch Size	2048 → 512
ColorJitter	0.3 → 0.0
AutoAugment	”3a” → ”rand-m9-mstd0.5-inc1”
RandomErasing	0.
AugmentRepeats	3 → 1
TestCropRatio	1.0
RandomCrop	rrc

Datasets. Our experiments are conducted with 10-fold cross-validation on popular TUDatasets (Morris et al., 2020). For each fold, we use $^1/_{10}$ of the respective training data as test set, $^1/_{10}$ for validation and $^8/_{10}$ for training. For every fold, we pick the test accuracy of the epoch with the best validation accuracy and report the average over all folds. For pLSTM, we encode the number of neighbors for each node with a standard positional encoding (Vaswani et al., 2017). Otherwise, no data transformation, augmentation, or normalization is used.

Training procedure. We train all models for 100 epochs with AdamW, a learning rate of 0.001, batch size of 64, and a cosine decay learning rate schedule with 5 warmup epochs.

Models. The compared models are GAT (Veličković et al., 2018), GCN (Kipf & Welling, 2017), GIN (Xu et al., 2019), LSTM GNN (Liang et al., 2016), MPNN (Gilmer et al., 2017). All GNNs consist of an encoder, decoder, and 4 or 8 message passing layers. The best model configuration is selected based on the validation accuracy. pLSTM also consists of the same encoder and decoder, but has 2 layers that operate in D-mode and 2 layers that operate in P-mode. To achieve an approximately similar parameter count, we fix the hidden dimension of pLSTM to 96, while the other GNNs have a hidden dimension of 128. All trained models have a parameter count of < 300,000.

Table 7: ImageNet1k - DeiT style ablation training parameters

Parameter	Value
Image Resolution	224×224
Training Epochs	400
Hidden Dimension	192
Num Heads	3
DropPath Rate	0.
LayerScale	-
Warmup Epochs	5
Peak Learning Rate	1e-3
Weight Decay	0.05
Gradient Clip Norm	1.0
Optimizer	AdamW
Loss Type	Cross Entropy
MixUp	0.8
CutMix	1.0
Label Smoothing	$0.0 \rightarrow 0.1$
Global Batch Size	2048
ColorJitter	0.0
AutoAugment	”rand-m9-mstd0.5-inc1”
RandomErasing	0.25
AugmentRepeats	$3 \rightarrow 1$
TestCropRatio	1.0
RandomCrop	rrc