pLSTM: parallelizable Linear Source Transition Mark networks

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

Sepp Hochreiter[1,3]

[1] ELLIS Unit Linz, LIT AI Lab, Institute for Machine Learning
Johannes Kepler University Linz, Austria
[2] Zuse School ELIZA
[3] NXAI GmbH
poeppel@ml.jku.at

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. The complete code is available at `https://github.com/ml-jku/plstm_experiments`.

1 Introduction

Linear RNNs such as DeltaNet [Schlag et al., 2021], Gated Linear Attention [Yang et al., 2023], Mamba [Gu and Dao, 2023, Dao and 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 for multi-dimensional data such as images, they enforce a pre-defined sequential traversal structure [Zhu

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

et al., 2024, Alkin et al., 2025]. To illustrate, for images, this corresponds to processing patches in a scanline form. However, this is problematic as, within one layer, even (vertical) neighbors in a patch grid are only distantly related, in the (horizontal line-wise) processing. Even 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 and Frasconi, 1993, Schmidhuber, 2015] and suffers from vanishing activations/gradients [Hochreiter, 1991, Bengio et al., 1994, Hochreiter et al., 2001, Hochreiter and Schmidhuber, 1997, Pascanu et al., 2013].

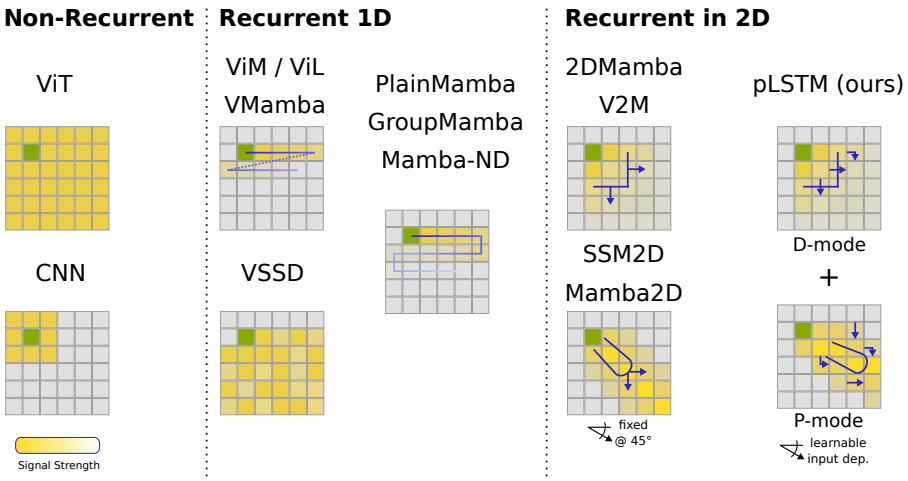

Figure 1: Illustration of the **receptive fields** induced by pLSTM and related architectures (for a single layer). CNNs are locally restricted while ViTs have a global receptive field. Modern recurrent architectures, such as ViM, traverse the 2D grid sequentially. pLSTM effectively extends the receptive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

Multi-dimensional RNNs (MDRNNs) [Graves et al., 2007] have demonstrated how non-linear RNNs like LSTMs [Hochreiter, 1991, Hochreiter and 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]. In this work, we translate the findings of MDRNNs to linear RNNs and introduce parallelizable Linear Source Transition Mark Networks (pLSTM). To achieve this, we first translate their input, forget, and output gates to Source, Transition, and Mark gates, respectively. This is to make the resulting structure applicable to general DAGs. Second, we derive a parallelization scheme enabled by the previous adaptation, leading to higher-order Source, Transition, and Mark terms - forming a parallel associative scan in multiple dimensions and DAGs. This also allows for a chunkwise-parallel formulation known from linear RNNs [Yang et al., 2023, Beck et al., 2025a], using the parallelization up to a certain level and the recurrent option between chunks/patches. Third, we derive two stabilization options, the P-mode for directional propagation and the D-mode for undirected global distribution. In Figure 1, we illustrate how the two modes affect the receptive field of pLSTM compared to other linear RNNs. We use pLSTM with MLP layers in a residual pre-LayerNorm backbone, resembling a Transformer with replaced multi-head attention.

To showcase the long-range capabilities of pLSTM in multiple dimensions, we first introduce the synthetic arrow-pointing extrapolation task (see Figure 2). As the directional information is not tied to only testing horizontal and vertical cases separately, a certain line-wise or column-wise ordering (even in alternation) cannot capture the directional information in the sequential RNN transitions, at least for a limited number of layers. Importantly, pLSTM generalizes well to increasing image resolutions compared to the Transformer (see Section 5.1). Moreover, we demonstrate the efficacy of pLSTM on established computer-vision and graph benchmarks. Experiments on ImageNet-1k (see Section 5.2) and on molecular graphs (see Section 5.4) 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, which comes 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 P- and D-mode cases.
- 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.

2 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 and Dao, 2023, Dao and 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.

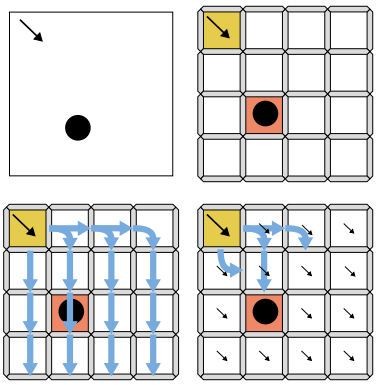

Figure 2: Illustration of the **Arrow pointing task**. The model has to classify whether an arrow is pointing towards a circle (top left). Models with global receptive fields, such as Vision Transformers (ViTs), can solve this task by leveraging directional information encoded via positional embeddings (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.

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]. Similarly, several works use Mamba on graph data, covering the graph with multiple paths [Wang et al., 2024a] or forming a sequence in other ways [Behrouz and Hashemi, 2024], with an overview presented in Atitallah et al. [2024]. To the best of our knowledge, all of these rely on a sequential representation to apply Mamba at some point, and therefore do not use the graph's structural information for the recurrence. Recent works extend the Mamba architecture to two dimensions. Most related to our work are V2M [Wang et al.,

2024b] and 2DMamba [Zhang et al., 2024a], 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 fixed normalization factor of $1/2$ (see Appendix D.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.

3 Background

3.1 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:

$$\boldsymbol{c}_n = \sum_{n' \in \mathcal{P}_{\mathcal{G}}(n)} \boldsymbol{f}_{nn'} \odot \boldsymbol{c}_{n'} + \boldsymbol{i}_n \odot \boldsymbol{z}_n \tag{1}$$

$$\boldsymbol{h}_n = \boldsymbol{o}_n \odot \tanh(\boldsymbol{c}_n) \tag{2}$$

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.

3.2 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.

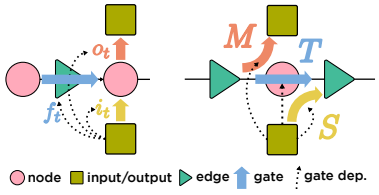

Figure 3: Transition from input/-forget/output gating in between nodes (and input/output) in (linear) RNNs towards **Source/Transition/-Mark gating** between edges and input/output (bottom/top) in pLSTMs. As for linear RNNs, the gates do not depend on states, but inputs only.

$$C_{tkv} = F_t C_{t-1kv} + K_{tk} V_{tv} \tag{3}$$

$$H_{tv} = Q_{tk} C_{tkv} \tag{4}$$

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].

4 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 3 and 6):

$$C_{ekv} = \sum_{e' \in E_{\rightarrow}(n_{\rightarrow}(e))} T_{ee'}^{n_{\rightarrow}(e)} C_{e'kv} + K_k^{n_{\rightarrow}(e)} V_v^{n_{\rightarrow}(e)} S_e^{n_{\rightarrow}(e)} \tag{5}$$

$$H_v^n = Q_k^n \sum_{e \in E_{\rightarrow}(n)} M_e^n C_{ekv} + K_k^n V_v^n D^n Q_k^n \tag{6}$$

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 and 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_v^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.

4.1 Parallelization on DAGs

4.1.1 Naïve Parallelization over Paths

Since all operations are linear, the iterative definition of Equation 5 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:

$$H_v^n = \sum_{n' < n} K_k^{n'} V_v^{n'} \underbrace{\sum_{e', e} S_{e'}^{n'} \left(\sum_{P \in \mathcal{P}(e', e)} \prod_{n^p \in P} T_{e_{\leftarrow}^{n^p} e_{\rightarrow}^{n^p}}^{n^p} \right) M_e^n}_{G^{n'n}} Q_k^n + K_k^n V_v^n \underbrace{D_{}^n}_{G^{nn}} Q_k^n \tag{7}$$

The middle STM part in Equation 7 can be precomputed into the resulting gating matrix $G^{n'n}$, and the state expansion and projection via Q_k^n and $K_k^{n'}$ can be integrated externally, similar to classical (linear) self-attention [Vaswani et al., 2017, Katharopoulos et al., 2020], the D^n term represents the diagonal entries. Computing the gating matrix via all paths is infeasible due to their exponential number (see Appendix D.1).

4.1.2 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 B.2, we show how this procedure works for DAGs. For regular grids such as sequences in 1D and images in 2D, we can derive a scheme of einsum, concatenation, and padding operations, which are well supported by modern hardware and parallel computation frameworks. In Appendix B.3, we show the parallelization in 1D. In Appendix B.4, we show the parallelization scheme for 2D grids. To better convey the core idea of pLSTM parallelization, we include a sketch for all three cases and pLSTM in general in Figure 6 of the appendix.

4.2 Long-Range Stability

For sequential data, it is known since the seminal LSTM work of Hochreiter and 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 and Schmidhuber, 1997]. The Cell states, however, can grow exponentially if the forget gates are limited to only one. See Appendix D.1 for details.

Using the more general setting of Transitions $T^n_{ee'}$ from edge e' to edge e, as introduced in this work, we can derive two modes of stability. First, a propagation mode **(P-mode)** that covers all paths in the DAG, but is effectively limited to a line as mode of propagation. Second, a distribution mode **(D-mode)** that limits the Transitions to a subset, such that their resulting line-graph is reduced from a DAG to a multi-tree Jung [1978].
The line graph of a DAG $\mathcal{G} = (N, E)$ is the DAG $\tilde{\mathcal{G}} = (E, \tilde{E})$ formed by the edges E of the original DAG and the line edges $\tilde{e} \in \tilde{E} \iff \tilde{e} = (e_1, e_2), \exists\, n \in N : e_1 \in E_{\rightarrow}(n) \cap e_2 \in E_{\leftarrow}(n)$, 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 $T^n_{ee'}$, these can be interpreted as states and edges on the nodes of the line graph $\tilde{\mathcal{G}}$. The Transitions $T_{ee'}$ can be viewed as entries of the adjacency matrix $\boldsymbol{T}$ of $\tilde{\mathcal{G}}$. This way, we can apply the theory of power iterations and matrix norms. The application of a pLSTM of Equation 5 is equivalent to the application of the power iteration and can be bounded by compatible matrix norms $|\cdot|$:

$$|\boldsymbol{c}| = \left|\left(\sum_{p=0}^{\infty} \boldsymbol{T}^p\right)\boldsymbol{s}\right| = \left|\left(\sum_{p=0}^{P_{\tilde{\mathcal{G}}}} \boldsymbol{T}^p\right)\boldsymbol{s}\right| \overset{|\boldsymbol{T}|\leq 1}{\leq} P_{\tilde{\mathcal{G}}}|\boldsymbol{s}| \tag{8}$$

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

The first stabilization option for the **P-mode** is to limit the norm of $\boldsymbol{T}$ as in Equation 8. A node-local option to achieve this is to limit the Transition entries $T^n_{ee'}$ per node as:

$$\sum_{e \in E_{\leftarrow}(n)} |T^n_{ee'}| \overset{!}{\leq} 1 \tag{9}$$

This 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 and Johnson, 1985], this can be applied to the matrix powers in Equation 8 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 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).

For the second stabilization option, the **D-mode**, we do not limit a matrix norm of $\boldsymbol{T}$, but instead reduce $\boldsymbol{T}$ or $\tilde{\mathcal{G}}$ from a DAG to a multitree [Jung, 1978], i.e., a directed graph, such that two nodes are only connected by a single path. The original graph $\mathcal{G}$ is not reduced to such a multitree by that. Since there is only one path in the line graph structure, there is only one path term in Equation 7, so there is no exponential explosion of path numbers, and limiting single $\boldsymbol{T}$ entries by one is sufficient.

4.3 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 B.3.

4.3.1 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: $\curvearrowright, \curvearrowleft, \curvearrowleft, \curvearrowright$. We focus on the first, the right-down combination, for the description, but all four combinations should be used to cover all directional interactions. Note that here, nodes correspond to pixels/patches and edges to the respective connections between them, including the introduced direction. For the parallelization, we derive a hierarchy of different levels of Source, Transition and Mark, covering larger chunks of multiple nodes. This is also depicted in Figure 6 and described in Appendix Section B.2 and B.4.
As Transitions follow all edge combinations, there are four options again: $\rightarrow, \hookrightarrow, \searrow, \downarrow$. The level zero Source, Transition, and Mark tensors are therefore: $S^{\mapsto}_{xy}, S^{\mathbin{\downarrow\!\!\!\downarrow}}_{xy}, T^{\rightarrow}_{xy}, T^{\hookrightarrow}_{xy}, T^{\searrow}_{xy}, T^{\downarrow}_{xy}, M^{\rightarrowtail}_{xy}, M^{\mathbin{\downarrow\!\!\!\downarrow}}_{xy}$. The general DAG is depicted in Figure 6 in the center right. As the in- and out-degrees of this DAG are now larger than one, there are more stabilization options, namely the P-mode and D-mode (Section 4.2).

P-mode When we allow all Transition options $\rightarrow, \hookrightarrow, \searrow, \downarrow$, we are in the P-mode. In general, the Transition matrix at position x, y looks like $T_{xy} = \begin{bmatrix} T^{\rightarrow}_{xy} & T^{\hookrightarrow}_{xy} \\ T^{\searrow}_{xy} & T^{\downarrow}_{xy} \end{bmatrix}$. The P-mode stabilization now limits this matrix to absolute column sums ≤ 1. This limits the structure at criticality to $\begin{bmatrix} \alpha & \beta \\ 1-\alpha & 1-\beta \end{bmatrix}$. When setting $\beta = \alpha$, we even arrive at a geometrical interpretation of this Transition: it directionally propagates the signal with maximal amplitude along the line defined by $\frac{\Delta x}{\Delta y} = \alpha$ (see Appendix D.2 for the derivation). In Figure 6, bottom right, we depict the receptive field of a single non-zero Source with constant Transitions of this structure across the grid. For a practical parameterization, we, therefore, fix this structure with the 'angle' of propagation α and an additional decay factor γ:

$$T = \gamma \begin{bmatrix} \alpha & \alpha \\ (1-\alpha) & (1-\alpha) \end{bmatrix} \tag{10}$$

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 $T^{\searrow}_{xy}$ have to be set to zero globally. In this way, two edges can only be connected via a single path. In Figure 6, the bottom right, the second option is shown, with an additional decay. At criticality, this node can still reach all other nodes to its bottom right with magnitude one, a long-range propagation in two dimensions.

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.

State-Tracking Expansion In Appendix B.5, we show how pLSTMs can be extended for state tracking via non-diagonal Transitions [Merrill et al., 2024, Siems et al., 2025, Grazzi et al., 2025].

5 Experiments

We first showcase the theoretical advantages of pLSTM on the synthetic arrow-pointing extrapolation task (see Section 5.1). Then 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 Sections 5.2 and 5.3). Finally, we illustrate how pLSTM scales to more than two input dimensions on established graph benchmarks (see Section 5.4).

5.1 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 (see also Figure 2). This is a long-range directional task as it involves the relative

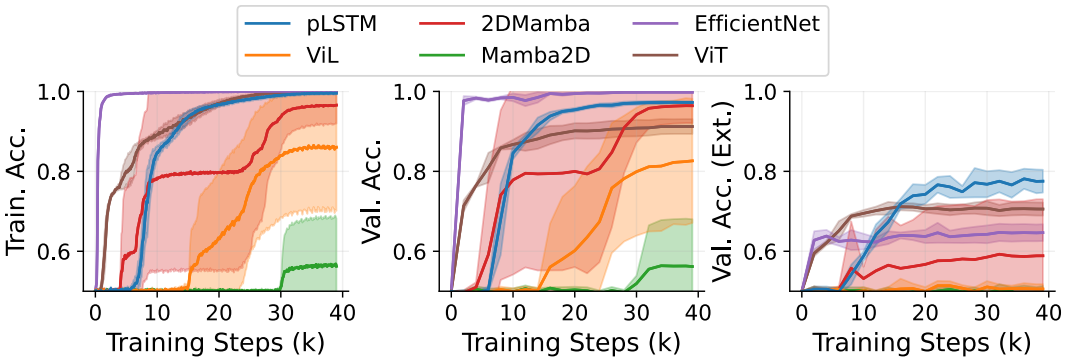

Figure 4: Training curves for the **Arrow Pointing Extrapolation task**, averaged over 5 seeds with 90% CI. ViT and EfficientNet can quickly match the training set (left), and EfficientNet reaches the best validation performance of all models on the samples of the same resolution as seen during training. 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.

positioning of two objects, in conjunction 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 details on the training, see Appendix E.2. In Figure 4, we show the learning curves for pLSTM and all baselines. We find that pLSTM performs well on both the training and validation tasks. Importantly, pLSTM exhibits the strongest extrapolation abilities. In contrast, ViL performs poorly because it traverses the image patches sequentially along the scanline form and misses out on important directional information. Despite its global receptive field, ViT falls short on extrapolation to larger image resolutions, which we surmise is due the its positional encoding. Similarly, EfficientNet exhibits strong performance on training/validation tasks, but fails to extrapolate. Finally, pLSTM considerably outperforms Mamba2D (which covers the P-mode) and 2DMamba (which covers the D-mode), despite their multidimensional nature. Also note that, in practice, pLSTM is much faster than Mamba2D or 2DMamba on real hardware, despite missing kernels, and on par with ViL (see 6). In Figure 5, we highlight the performance differences between D- and P-mode-only models on the arrow pointing task.

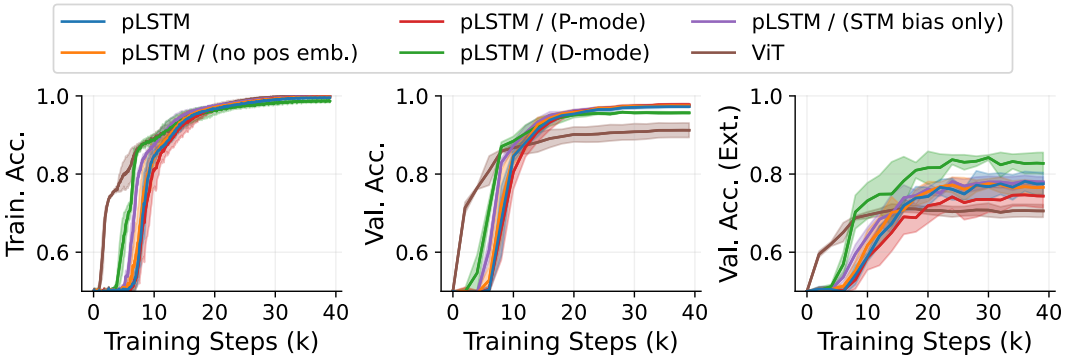

Figure 5: Training curves for the **Arrow Pointing Extrapolation task**, averaged over 5 seeds with 90% CI on different model ablations. P-mode by itself performs worse, D-mode by itself is not as general in interpolation, but performs better on extrapolation compared to other models. pLSTM does not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

Table 1: 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 and Le, 2019, Zhang et al., 2024b, Baty et al., 2024]. Note that EfficientNet, as a CNN-based baseline, 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 B.7).

Model	Epochs	#Params	FLOPS	IN-1K
EfficientNet-B0 [Tan and 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 and 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 and 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

5.2 ImageNet1k

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 E.3.

5.3 ImageNet-1k Ablation

We ablate our pLSTM and compare against a ViT trained using the ViT-T model scale and a simpler single pre-training schedule close to DeiT [Touvron et al., 2021] at 224×224 resolution (see Appendix E.3.1) for details. At this scale, pLSTM outperforms ViT by a significant margin. Results are shown in Table 2.

[0] `https://github.com/facebookresearch/fvcore`

Table 2: 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	73.49

Table 3: Long-Range Graph Benchmark Results [Dwivedi et al., 2022]

Model	Peptides -struct (MAE) ↓	Peptides -func (Avg. Prec.) ↑
GAT [Veličković et al., 2018]	0.270	0.525
GCN [Kipf and Welling, 2017]	0.264	0.534
GIN [Xu et al., 2018]	0.328	**0.593**
LSTM-GNN [Liang et al., 2016]	0.274	0.502
MPNN [Gilmer et al., 2017]	**0.260**	0.557
pLSTM	0.264	0.536

5.4 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] and the Long-Range Graph Benchmark [Dwivedi et al., 2022]. The results in Tables 3 and 4 show that pLSTM can compete with popular GNN architectures on those datasets. Similar to the computer vision experiments (Section 5.2), 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 adapt the Transitions.

Table 4: 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 and 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

6 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. While pLSTMs should theoretically enable long-range propagation of activations and gradients in multiple dimensions for recurrent, subquadratic models, the arrow pointing extrapolation task only tests this in a restricted way. Moreover, while pLSTM models generalize better than ViTs, which use position embeddings to encode spatial relations, there is still a gap to 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.

Acknowledgments and Disclosure of Funding

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

Korbinian Pöppel was supported by the Konrad Zuse School of Excellence in Learning and Intelligent Systems (ELIZA) through the DAAD program Konrad Zuse Schools of Excellence in Artificial Intelligence, sponsored by the Federal Ministry of Education and Research during his ELLIS visit at Uni Freiburg, where this work was finished.

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.

References

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

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

S. B. Atitallah, C. B. Rabah, M. Driss, W. Boulila, and A. Koubaa. Exploring graph mamba: A comprehensive survey on state-space models for graph learning. *arXiv preprint arXiv:2412.18322*, 2024.

E. Baron, I. Zimerman, and L. Wolf. 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.

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

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

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

A. Behrouz and F. Hashemi. Graph Mamba: Towards Learning on Graphs with State Space Models. *CoRR*, abs/2402.08678, 2024. URL https://doi.org/10.48550/arXiv.2402.08678.

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

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

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

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

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

T. Dao, D. Y. Fu, S. Ermon, A. Rudra, and C. Ré. FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness, June 2022. URL `http://arxiv.org/abs/2205.14135`. arXiv:2205.14135 [cs].

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

A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. 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`.

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

V. P. Dwivedi and X. Bresson. A generalization of transformer networks to graphs. *arXiv preprint arXiv:2012.09699*, 2020.

V. P. Dwivedi, A. T. Luu, T. Laurent, Y. Bengio, and X. Bresson. Graph neural networks with learnable structural and positional representations. *arXiv preprint arXiv:2110.07875*, 2021.

V. P. Dwivedi, L. Rampášek, M. Galkin, A. Parviz, G. Wolf, A. T. Luu, and D. Beaini. Long range graph benchmark. *Advances in Neural Information Processing Systems*, 35:22326–22340, 2022.

F. Gers. Learning to forget: continual prediction with LSTM. In *9th International Conference on Artificial Neural Networks: ICANN '99*, volume 1999, pages 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`.

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

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

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

R. Grazzi, J. Siems, J. K. H. Franke, A. Zela, F. Hutter, and M. Pontil. 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`.

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

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

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

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

J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, Apr. 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.

R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, 1 edition, Dec. 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.

H. A. Jung. 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.

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

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

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

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

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

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

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

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

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

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

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

R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. Pmlr, 2013.

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

K. Pöppel, M. Beck, and S. Hochreiter. FlashRNN: I/O-Aware Optimization of Traditional RNNs on modern hardware. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=l0ZzTvPfTw.

L. Rampášek, M. Galkin, V. P. Dwivedi, A. T. Luu, G. Wolf, and D. Beaini. Recipe for a general, powerful, scalable graph transformer. *Advances in Neural Information Processing Systems*, 35: 14501–14515, 2022.

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

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

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

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

J. Schmidhuber. 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`.

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

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

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

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

M. Stollenga, W. Byeon, M. Liwicki, and J. Schmidhuber. 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)*, pages 2998–3006, 2015.

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

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

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

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

H. Touvron, M. Cord, and H. Jégou. DeiT III: Revenge of the ViT. In *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXIV*, pages 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`. event-place: Tel Aviv, Israel.

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

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

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

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

C. Wang, O. Tsepa, J. Ma, and B. Wang. Graph-Mamba: Towards Long-Range Graph Sequence Modeling with Selective State Spaces, Feb. 2024a. URL http://arxiv.org/abs/2402.00789. arXiv:2402.00789 [cs].

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

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

S. Williams, A. Waterman, and D. Patterson. Roofline: an insightful visual performance model for multicore architectures. *Communications of the ACM*, 52(4):65–76, Apr. 2009. ISSN 0001-0782, 1557-7317. doi: 10.1145/1498765.1498785. URL https://dl.acm.org/doi/10.1145/1498765.1498785.

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

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

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

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

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

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

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

L. Zhu, B. Liao, Q. Zhang, X. Wang, and W. Liu. Vision mamba: Efficient visual representation learning with bidirectional state space model (preprint). *arXiv preprint arXiv:2401.09417*, 2024.

X. Zhu, P. Sobhani, and H. Guo. 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*, pages 917–926, 2016.

A Method Visualization

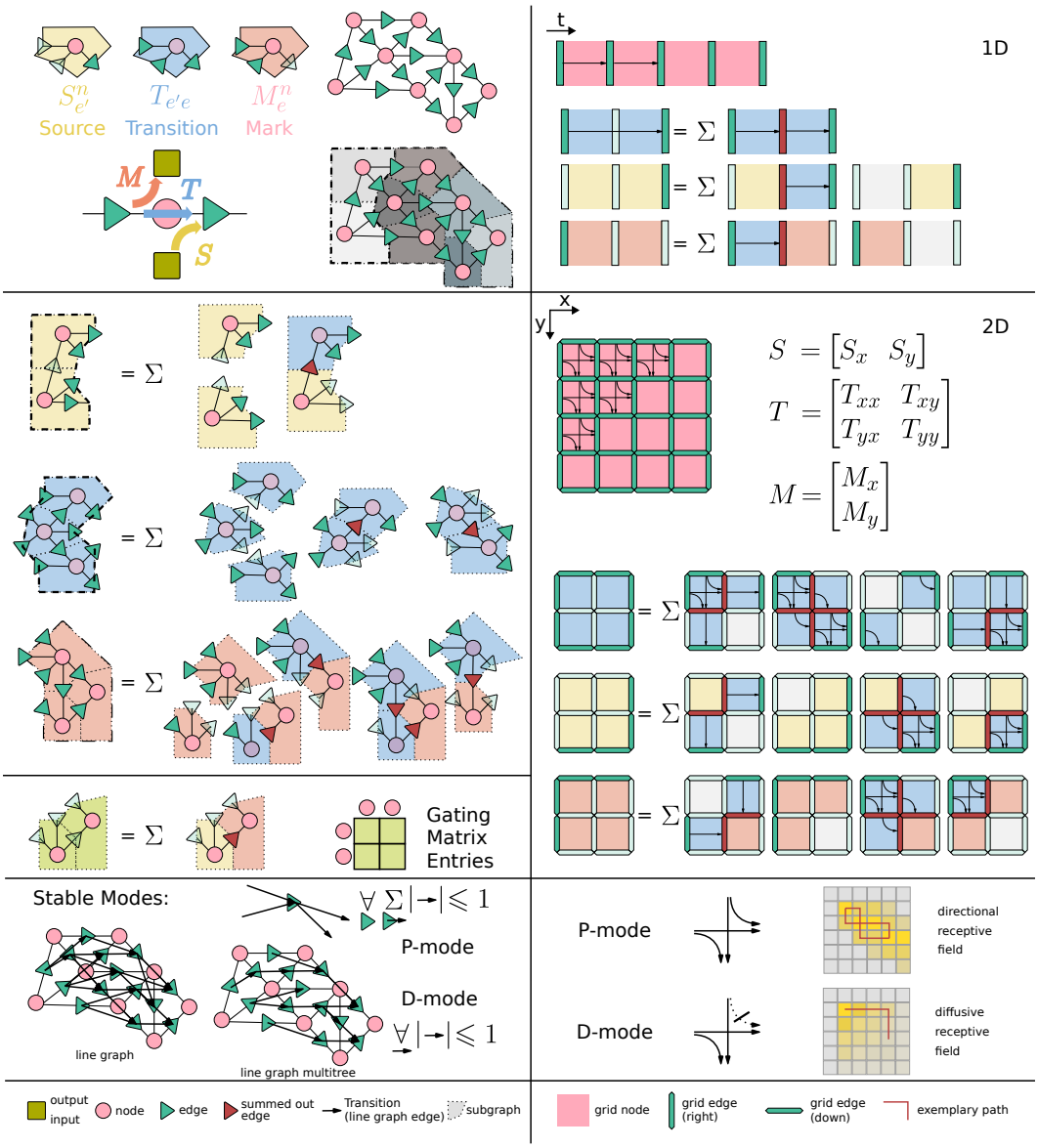

Figure 6: Illustration of **pLSTM** on general DAGs (left), and on 1D / 2D grids (right): In the top-left part, a general DAG is visualized, with Sources $S^n_{e'}$, Transitions $T_{e'e}$, and Marks M^n_e populating node-outgoing-edge, incoming-edge-outgoing-edge, and incoming-edge-node pairs. Due to the associative structure of these linear operators, they can be combined; some 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 here).

B Method Details

B.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 then.

B.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 $n', n \in N; n' \leq n$, 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:

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

$$T_{e'e}^{l,\alpha:k} := \sum_{P\in\mathcal{P}(e',e)} \prod_{n^P\in P} T^n_{e_\rightarrow^{n^P} e_\leftarrow^{n^P}} \tag{12}$$

$$M_{e'}^{n,l,\alpha:b} := \sum_{e\in E_\rightarrow(n)} \left(\sum_{P\in\mathcal{P}(e',e)} \prod_{n^P\in P} T^n_{e_\rightarrow^{n^P} e_\leftarrow^{n^P}} \right) M_e^n \tag{13}$$

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

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

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 $B_{\leftarrow\alpha:a}^l$, $B_{\rightarrow\alpha:a}^l$ as "edges". Note that, by Einstein sum convention we implicitly sum over multiple in-coming and out-going edges $e^{l,\alpha:c}_{\rightarrow m_c^l}$ and $e^{l,\alpha:c}_{\leftarrow m_c^l}$ for generalized Transitions $T^{l,\alpha:c}$ in $G^{n'n}$ and in the following. See Figure 6 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 $n' \in N_{\alpha:a}^l$ or $n \in N_{\alpha:a}^l$):

$$S_e^{n',l+1,\alpha} = \sum_{e'\in B_\leftarrow^l\alpha:a} S_{e'}^{n',l,\alpha:a} \sum_{P^l\in\mathcal{P}^l(e',e)} \prod_{b\in P^l} T^{l,\alpha:b}_{e^{l,\alpha:b}_{\rightarrow m_b^l} e^{l,\alpha:b}_{\leftarrow m_b^l}} \tag{14}$$

$$T_{e'e}^{l+1,\alpha} = \sum_{P^l\in\mathcal{P}^l(e',e)} \prod_{b\in P^l} T^{l,\alpha:b}_{e^{l,\alpha:b}_{\rightarrow m_b^l} e^{l,\alpha:b}_{\leftarrow m_b^l}} \tag{15}$$

$$M_{e'}^{n,l+1,\alpha} = \sum_{e\in B_\rightarrow^l\alpha:a} \left(\sum_{P^l\in\mathcal{P}^l(e',e)} \prod_{b\in P^l} T^{l,\alpha:b}_{e^{l,\alpha:b}_{\rightarrow m_b^l} e^{l,\alpha:b}_{\leftarrow m_j^l}} \right) M_e^{n,l,\alpha:a} \tag{16}$$

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 5 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.

B.3 Parallelization of pLSTM in 1D - sequences

For a sequence as a DAG, pLSTMs reduce to linear RNNs. 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:

$$S^l_{an} = \left[S^{l-1}_{(2a)(n\rfloor_0)} T^{l-1}_{(2a+1)}, S^{l-1}_{(2a+1)(n\rceil_0)}\right]_0 \tag{17}$$

$$T^l_a = T^{l-1}_{(2a)} T^{l-1}_{(2a+1)} \tag{18}$$

$$M^l_{an} = \left[M^{l-1}_{(2a+1)(n\rfloor_0)}, T^{l-1}_{(2a)} M^{l-1}_{(2a+1)(n\rceil_0)}\right]_0 \tag{19}$$

$$\begin{aligned} G^l_{an'n} = \Big[&\left[G^l_{(2a)(n'\rfloor_0)(n\rfloor_1)}, 0\right]_0, \\ &\left[S^{l-1}_{(2a)(n'\rfloor_0)} M^{l-1}_{(2a+1)(n\rceil_1)}, G^l_{(2a+1)(n'\rceil_0)(n\rceil_1)}\right]_0 \Big]_1 \end{aligned} \tag{20}$$

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 $\boldsymbol{a} \in \mathbb{R}^N$; $\boldsymbol{b}, \boldsymbol{c} \in \mathbb{R}^{N/2}$:

$$a_n = \left[b_{(n\rfloor_0)}, c_{(n\rceil_0)}\right]_0 := \begin{cases} b_n & n < N/2 \\ c_{n-N/2} & n \geq N/2 \end{cases} \tag{21}$$

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.

B.4 Parallelization of pLSTM in 2D - images

Here, 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 higher-level tensors, leading to a recursion shown below. The Source, Transition and Mark tensors at level l have the following structure: $S^{l,\mapsto}_{xyuva}, S^{l,\mathrel{\rotatebox{90}{$\mapsto$}}}_{xyuva}, T^{l,\rightarrow}_{xyab}, T^{l,\searrow}_{xyab}, T^{l,\searrow}_{xyab}, T^{l,\downarrow}_{xyab}, M^{l,\rightarrowtail}_{xyuvb}, M^{l,\mathrel{\rotatebox{90}{$\rightarrowtail$}}}_{xyuvb}$ with $u, v, a, b \in \{0..2^l - 1\}$.

$$\begin{aligned} S^{l+1,\mapsto}_{xyuwa} = \Bigg[&\Big[\Big[S^{l,\mapsto}_{(2x)(2y)(u\rfloor_0)(w\rfloor_1)b} T^{l,\rightarrow}_{(2x+1)(2y)b(a\rfloor_2)}, S^{l,\mapsto}_{(2x+1)(2y)(u\rceil_0)(w\rfloor_1)(a\rfloor_2)}\Big]_0, \left[0_{(u\rfloor_0)(w\rceil_1)(a\rfloor_2)}, 0_{(u\rfloor_0)(w\rceil_1)(a\rfloor_2)}\right]_0\Big]_1, \\ &\Big[\Big[S^{l,\mapsto}_{(2x)(2y)(u\rfloor_0)(w\rfloor_1)b} T^{l,\searrow}_{(2x+1)(2y)bc} T^{l,\searrow}_{(2x+1)(2y+1)c(a\rfloor_2)} + S^{l,\mathrel{\rotatebox{90}{$\mapsto$}}}_{(2x)(2y)(u\rfloor_0)(w\rfloor_1)b} T^{l,\searrow}_{(2x)(2y+1)bc} T^{l,\rightarrow}_{(2x+1)(2y+1)c(a\rfloor_2)}, \\ &S^{l,\mathrel{\rotatebox{90}{$\mapsto$}}}_{(2x+1)(2y)(u\rfloor_0)(w\rfloor_1)b} T^{l,\searrow}_{(2x+1)(2y+1)bc}\Big]_0, \Big[S^{l,\mapsto}_{(2x)(2y+1)(u\rfloor_0)(w\rceil_1)b} T^{l,\rightarrow}_{(2x+1)(2y+1)b(a\rfloor_2)}, S^{l,\mapsto}_{(2x+1)(2y+1)(u\rceil_0)(w\rceil_1)(a\rfloor_2)}\Big]_0\Big]_1\Bigg]_2 \end{aligned} \tag{22}$$

$$\tag{23}$$

$$S^{l+1,\mathfrak{l}}_{xyuwa} = \Big[\Big[\big[S^{l,\mathfrak{l}}_{(2x)(2y)(u\rfloor_0)(w\rfloor_1)b}T^{l,\downarrow}_{(2x)(2y+1)b(a\rfloor_2)}, 0_{(u\rceil_0)(w\rfloor_0)(a\rfloor_2)}\big]_0, \big[S^{l,\mathfrak{l}}_{(2x)(2y+1)(u\rfloor_0)(w\rceil_1)(a\rfloor_2)}, 0_{(u\rceil_0)(w\rceil_0)(a\rfloor_2)}\big]_0\Big]_1, \tag{24}$$

$$\Big[\big[S^{l,\mapsto}_{(2x)(2y)(u\rfloor_0)(w\rfloor_1)b}T^{l,\searrow}_{(2x+1)(2y)bc}T^{l\downarrow}_{(2x+1)(2y+1)c(a\rfloor_2)} + S^{l,\mathfrak{l}}_{(2x)(2y)(u\rfloor_0)(w\rfloor_1)b}T^{l,\hookrightarrow}_{(2x)(2y+1)bc}T^{\searrow}_{(2x+1)(2y+1)c(a\rfloor_2)},$$

$$S^{l,\mathfrak{l}}_{(2x+1)(2y)(u\rceil_0)(v\rfloor_1)b}T^{l,\downarrow}_{(2x+1)(2y+1)b(a\rceil_0)}\big]_0, \big[S^{l,\mapsto}_{(2x)(2y+1)(u\rfloor_0)(w\rceil_1)b}T^{l,\searrow}_{(2x+1)(2y+1)b(a\rfloor_2)}, S^{l,\mathfrak{l}}_{(2x+1)(2y+1)(u\rceil_0)(v\rceil_1)(a\rfloor_2)}\big]_0\Big]_1\Big]_2$$

$$T^{l,\rightarrow}_{xyab} = \Big[\big[T^{l,\rightarrow}_{(2x)(2y)(a\rfloor_0)c}T^{l,\rightarrow}_{(2x+1)(2y)c(b\rfloor_1)}, 0_{(a\rceil_0)(b\rfloor_1)}\big]_0, \tag{25}$$

$$\big[T^{l,\downarrow}_{(2x)(2y)(a\rfloor_0c)}T^{l,\hookrightarrow}_{(2x)(2y+1)cd}T^{l,\rightarrow}_{(2x+1)(2y+1)d(b\rceil_1)} + T^{l,\rightarrow}_{(2x)(2y)(a\rfloor_0)c}T^{l,\searrow}_{(2x+1)(2y)cd}T^{l,\hookrightarrow}_{(2x+1)(2y+1)d(b\rceil_1)},$$

$$T^{l,\rightarrow}_{(2x)(2y+1)(a\rceil_0)c}T^{l,\rightarrow}_{(2x+1)(2y+1)c(b\rceil_1)}, 0_{(a\rceil_0)(b\rfloor_1)}\big]_0\Big]_1$$

$$T^{l,\downarrow}_{xyab} = \Big[\big[T^{l,\downarrow}_{(2x)(2y)(a\rfloor_0)c}T^{l,\downarrow}_{(2x)(2y+1)c(b\rfloor_1)}, 0_{(a\rceil_0)(b\rfloor_1)}\big]_0, \tag{26}$$

$$\big[T^{l,\downarrow}_{(2x)(2y)(a\rfloor_0c)}T^{l,\hookrightarrow}_{(2x)(2y+1)cd}T^{l,\searrow}_{(2x+1)(2y+1)d(b\rceil_1)} + T^{l,\rightarrow}_{(2x)(2y)(a\rfloor_0)c}T^{l,\searrow}_{(2x+1)(2y)cd}T^{l,\downarrow}_{(2x+1)(2y+1)d(b\rceil_1)},$$

$$T^{l,\downarrow}_{(2x+1)(2y)(a\rceil_0)c}T^{l,\downarrow}_{(2x+1)(2y+1)c(b\rceil_1)}, 0_{(a\rceil_0)(b\rfloor_1)}\big]_0\Big]_1$$

$$T^{l,\hookrightarrow}_{xyab} = \Big[\big[T^{l,\hookrightarrow}_{(2x)(2y)(a\rfloor_0)c}T^{l,\rightarrow}_{(2x+1)(2y)c(b\rfloor_1)}, T^{l,\hookrightarrow}_{(2x+1)(2y)(a\rceil_0)(b\rfloor_1)}\big]_0, \tag{27}$$

$$T^{l,\hookrightarrow}_{(2x)(2y)(a\rfloor_0)c}T^{l,\searrow}_{(2x+1)(2y)cd}T^{l,\hookrightarrow}_{(2x+1)(2y+1)d(b\rceil_1)} + T^{l,\downarrow}_{(2x)(2y)(a\rfloor_0)c}T^{l,\hookrightarrow}_{(2x)(2y+1)cd}T^{l,\rightarrow}_{(2x+1)(2y+1)d(b\rceil_1)},$$

$$T^{l,\downarrow}_{(2x+1)(2y)(a\rceil_0)c}T^{l,\hookrightarrow}_{(2x+1)(2y+1)c(b\rceil_1)}\big]_0\Big]_1$$

$$T^{l,\searrow}_{xyab} = \Big[\big[T^{l,\searrow}_{(2x)(2y)(a\rfloor_0)c}T^{l,\downarrow}_{(2x)(2y+1)c(b\rfloor_1)}, T^{l,\searrow}_{(2x)(2y+1)(a\rceil_0)(b\rfloor_1)}\big]_0, \tag{28}$$

$$T^{l,\searrow}_{(2x)(2y)(a\rfloor_0)c}T^{l,\hookrightarrow}_{(2x)(2y+1)cd}T^{l,\searrow}_{(2x+1)(2y+1)d(b\rceil_1)} + T^{l,\rightarrow}_{(2x)(2y)(a\rfloor)_0c}T^{l,\searrow}_{(2x+1)(2y)cd}T^{l,\downarrow}_{(2x+1)(2y+1)d(b\rceil_1)},$$

$$T^{l,\rightarrow}_{(2x)(2y+1)(a\rceil_0)c}T^{l,\searrow}_{(2x+1)(2y+1)c(b\rceil_1)}\big]_0\Big]_1$$

$$M^{l,\mapsto}_{xyuva} = \Big[\Big[\big[M^{l,\mapsto}_{(2x)(2y)(u\rfloor_0)(v\rfloor_1)(a\rfloor_2)}, T^{l,\rightarrow}_{(2x)(2y)(a\rfloor_2)b}M^{l,\mapsto}_{(2x+1)(2y)(u\rceil_0)(v\rfloor_1)b}\big]_0, \big[T^{l,\searrow}_{(2x)(2y)(a\rfloor_2)b}M^{l,\mathfrak{l}}_{(2x)(2y+1)(u\rfloor_0)(v\rceil_1)b}, \tag{29}$$

$$T^{l,\searrow}_{(2x)(2y)(a\rfloor_2)b}T^{l,\hookrightarrow}_{(2x)(2y+1)bc}M^{l,\mapsto}_{(2x+1)(2y+1)(u\rceil_0)(v\rceil_1)c} + T^{l,\rightarrow}_{(2x)(2y)(a\rfloor_2)b}T^{l,\searrow}_{(2x+1)(2y)bc}M^{l,\mathfrak{l}}_{(2x+1)(2y+1)(u\rceil_0)(v\rceil_1)c}\big]_0\Big]_1,$$

$$\Big[\big[0_{(u\rfloor_0)(v\rfloor_1)(a\rfloor_2)}, 0_{(u\rceil_0)(v\rfloor_1)(a\rfloor_2)}\big]_0, \big[M^{l,\mapsto}_{(2x)(2y+1)(u\rfloor_0)(v\rceil_1)(a\rfloor_2)}, T^{l,\rightarrow}_{(2x)(2y+1)(a\rfloor_2)b}M^{l,\mapsto}_{(2x+1)(2y+1)(u\rceil_0)(v\rceil_1)b}\big]_0\Big]_1\Big]_2$$

$$M^{l,\mathfrak{l}}_{xyuva} = \Big[\Big[\big[M^{l,\mathfrak{l}}_{(2x)(2y)(u\rfloor_0)(v\rfloor_1)(a\rfloor_2)}, T^{l,\hookrightarrow}_{(2x)(2y)(a\rfloor_2)b}M^{l,\mapsto}_{(2x+1)(2y)(u\rceil_0)(v\rfloor_1)b}\big]_0, \big[T^{l,\downarrow}_{(2x)(2y)(a\rfloor_2)b}M^{l,\mathfrak{l}}_{(2x)(2y+1)(u\rfloor_0)(v\rceil_1)b}, \tag{30}$$

$$T^{l,\hookrightarrow}_{(2x)(2y)(a\rfloor_2)b}T^{l,\searrow}_{(2x+1)(2y)bc}M^{l,\mathfrak{l}}_{(2x+1)(2y+1)(u\rceil_0)(v\rceil_1)c} + T^{l,\downarrow}_{(2x)(2y)(a\rfloor_2)b}T^{l,\hookrightarrow}_{(2x)(2y+1)bc}M^{l,\mapsto}_{(2x+1)(2y+1)(u\rceil_0)(v\rceil_1)c}\big]_0\Big]_1,$$

$$\Big[\big[0_{(u\rfloor_0)(v\rfloor_1)}, M^{l,\mathfrak{l}}_{(2x+1)(2y)(u\rceil_0)(v\rfloor_1)}\big]_0, \big[0_{(u\rfloor_0)(v\rceil_1}, T^{l,\downarrow}_{(2x+1)(2y)(a\rceil_2)b}M^{l,\mathfrak{l}}_{(2x+1)(2y+1)(u\rceil_0)(v\rceil_1)b}\big]_0\Big]_1\Big]_2$$

B.5 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:

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

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

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

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

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] can be applied along multiple dimensions.

B.6 Stability in the State Tracking Extension

The stabilization of Section B.5 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 9. 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].

B.7 Chunkwise-Parallel Form

Given the hierarchical structure of the parallelization shown in Sections B.2, B.3, and B.4, 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.

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 7). This fully parallelized form was used for the vision experiments.

C Full Model Architecture

C.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. 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.

D Theoretical Analysis

D.1 Exponential growth of path counts on 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 $\Delta_x + \Delta_y =: \Delta$:

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

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 7. 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.

D.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:

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

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.

D.3 Computational complexity analysis

For the (chunkwise) recurrent mode of operation, the state updates need to be taken into consideration. Let's ignore the S and M multiplications for the moment. For each chunk (higher level node), there are $E_{in} * E_{out}$ state transition operations (T C) and N_{nodes} updates of key-value outer products. The source terms can be absorbed into K / V, and the pre-computation overhead is neglected in this analysis. This leads to a complexity per chunk for the recurrent part ($C' = \sum TC + SKV$):

$$F^{FLOP}_{rec} = N_{ein} * N_{eout} * D_K * D_V + N_{nodes} * D_K * D_V * N_{eout} \tag{39}$$

There is also the memory complexity of loaded memory parts, assuming that the loaded chunk data (K, V) can be kept in lower-level memory, iterating over edge combinations:

$$F^{MEM}_{rec} = N_{ein} * N_{eout} * D_K * D_V + N_{nodes} * (D_K + D_V) \tag{40}$$

The edge-combination memory overhead is reduced by the amount of edge/cell states that can be cut down to $(N_{ein} + N_{eout}) * D_K * D_V$. For the parallel computation ($Q \cdot C + (G \odot Q \cdot K) \cdot V$), we get the following complexities, assuming that the gating and mark terms are pre-computed with no overhead:

$$F^{FLOP}_{par} = N_{ein} * N_{nodes} * D_K * D_V + N^2_{nodes} * (D_K + D_V) \tag{41}$$
$$F^{MEM}_{par} = N_{ein} * D_K * D_V + N_{nodes}(2 * D_K + D_V) \tag{42}$$

Note that this can be further fused, if D_K, D_V are small enough. Actually, for the concrete reduction of the einsums, we rely on JAX+XLA, which might peform this suboptimally. Now, if we add this up with a certain overhead M per memory op (333 FLOPs/byte for a H100 on TensorCores), and add a scaling for the $N_{ein}, N_{eout} = N^{\alpha}_{nodes}, \alpha = 1/2$ in 2D, $N_{total} = N_{nodes} * N_{chunks}$:

$$F^{EFF} = N_{total}/N_{nodes} * [(2 * N^{1+\alpha}_{nodes} + N^{2\alpha}_{nodes} + N^2_{nodes}) * D_K * D_V \tag{43}$$
$$+ M * (3 * N^{\alpha}_{nodes} * D_K * D_V + N_{nodes} * (3 * D_K + 2 * D_V))] \tag{44}$$

Removing constants and scales and calculating the derivative of the leading order terms ($D_K * D_V$), we arrive at:

$$d\tilde{F}^{EFF} = 2\alpha * N^{\alpha-1}_{nodes} + (2\alpha - 1) * N^{2\alpha-2}_{nodes} + 1 + M * 3(\alpha - 1) * N^{\alpha-2}_{nodes} = 0 \tag{45}$$

which leads to an optimal number of nodes per chunk N_{nodes} (derivative equals 0) for 2D ($\alpha = 1/2$):

$$d\tilde{F}^{EFF} = N_{nodes} + N^{3/2}_{nodes} - 3/2 * M = 0 \tag{46}$$

So, the optimum is:

$$(3/2 * M)^{2/3} \leq N_{nodes}^{2D*} \leq (3 * M)^{2/3} \tag{47}$$

For the sequential linear RNN case ($\alpha = 0$), it is:

$$N_{nodes}^{1D*} = 2\sqrt{M} \tag{48}$$

Note that lower-order terms are neglected here, and FLOPs counting might vary in prefactors, but the overall scaling of the optimum remains the same. Also, interestingly, for graphs where the edges in between chunks scale as the number of contained nodes (or worse, $\alpha \geq 1$), the derivative is always positive and the optimum is the pure recurrent computation.

Modern GPUs need a high arithmetic intensity [Williams et al., 2009, Dao et al., 2022, Pöppel et al., 2025] such that FLOPs become the bottleneck of computation. Because of this trade-off between FLOPs versus runtime + memory, typically more parallelization is beneficial for actual runtime up to a certain Pareto limit (see also Beck et al. [2025a]). For larger images, full parallelization will not be the optimal case due to the quadratic FLOPs complexity in the image size.

E 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`.

E.1 Initialization

While for the ablations and arrow pointing extrapolation, we use zeros initialization for the weight terms of Source, Mark, Transition, Direct as well as Orientation (P mode angle) layers, on the DeiT-III style training on ImageNet1k, using a non-zero small normal init leads to significantly better results reported here. For ViT, we observe that LayerScale initialized with 1e-4 is important for ImageNet1k training, whereas on Arrow Pointing Extrapolation, it is important to initialize the LayerScale at 1 to reach the observed performance.

Table 5: General pLSTM initialization settings

Parameter	Value
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)

E.2 Arrow Pointing

Here, we train for 50 epochs 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

[1] `https://jax.dev`

[2] `https://pytorch.org`

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.

E.2.1 Test results

Table 6: Test Results on Arrow Pointing Extrapolation (5 seeds, 90% confidence interval) and training time on one H100, * is just trained on one learning rate, $^\circ$ is without flash attention for fair comparison

Model	Best LR	Test Acc.	Test Acc. (Ext.)	Training Time (h)
pLSTM	0.0001	0.972 ± 0.003	0.778 ± 0.031	2.0
pLSTM / (no posemb.)	0.0001	0.975 ± 0.003	0.769 ± 0.018	2.0
pLSTM / (P-mode only)	0.0003	0.978 ± 0.005	0.746 ± 0.027	2.0
pLSTM / (D-mode only)	0.0001	0.957 ± 0.002	0.828 ± 0.028	2.0
pLSTM / (STM bias only)	0.0001	0.975 ± 0.003	0.784 ± 0.020	2.0
ViT	0.0003	0.915 ± 0.019	0.707 ± 0.014	0.7°
ViL	3e-05	0.823 ± 0.157	0.503 ± 0.004	1.9
EfficientNet	0.001	0.998 ± 0.001	0.649 ± 0.023	1.2
2DMamba	0.0001*	0.964 ± 0.045	0.584 ± 0.144	8.1
Mamba2D	0.0001*	0.553 ± 0.121	0.500 ± 0.001	7.3
V2M	0.0001*	0.948 ± 0.009	0.557 ± 0.036	26.7
V2M+Mamba2D	0.0001*	0.496 ± 0.005	0.500 ± 0.001	17.9

E.3 ImageNet-1k

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

E.3.1 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 8, we provide a summary of the used training hyperparameters.

E.4 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 7: 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 and 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.

Graph Positional Embedding. Encodings are frequently used to boost performance on graph benchmarks. For example, an encoding based on eigenvectors and eigenvalues of the graph Laplacian (LapPE) [Dwivedi and Bresson, 2020] has been frequently used to improve results on peptides-struct, while a random-walk structural encoding (RWSE) performs well on peptides-func [Dwivedi et al., 2021, Rampášek et al., 2022]. Especially the RWSE could skew the results, as it is very similar to message passing (which we compare against). However, in our experiments, we do not include LapPE or RWSE for pLSTM or any other method for a cleaner architectural comparison. In this setting, pLSTM performs comparable or better than several baselines on the peptide benchmarks. pLSTM is also fully compatible with such encodings, and integrating them could further improve performance. However, we intentionally did not include them in this study to better isolate the effect of our proposed architecture incorporating the structural information.

Table 8: 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

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).

1. **Claims**

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

   Answer: [Yes]

   Justification: Our claims of contribution are reflected by the respective related work sections, which includes to the best of our knowledge all related works, and our distinctive qualities compared to those. We believe that the abstract provides a good summary of the paper's storyline.

   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: We provide a respective paragraph as part of the conclusion.

   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: [Yes]

   Justification: Our theoretical results are accompanied by the relevant assumptions and proven within the paper.

   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: In addition to the detailed experiment setup description, we (will) provide the code including training configurations as part of our submission supplement.

   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: We (will) include the training code including training configurations as part of our submission.

   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: We specify the training details in the Experimental Section of the supplement and include the training code with configurations as part of our submission.

   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: While for the large scale experiments on ImageNet1k, computational restrictions limit us to report just single run accuracies (as it is common for those), our smaller scale experiments all include error bars (and description) showing the statistical significance.

   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: We provide detailed training times and compute resources in the Experiments section of 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: We use common, public and synthetic datasets that do not contain PII. While we cannot rule out negative societal impacts of this work in the distant future, we do not see a mid-term impact. Our aim is to benefit society by expanding the technical capabilities of machine learning systems.

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 see significant societal impacts of our work in the short-term future, but improvements in understanding, capabilities and efficieny of general model architectures might well have some over long time scales.

    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 think our models have high risk for misuse.

    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: Code used from other sources under Apache 2.0 License is properly credited both in the adapted code and the paper.

    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: While an experimentation codebase is by definition experimental, we try to have core assets (model architecture, training pipeline) well tested and documented.

    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: None of our experiments / research contains newly crowd-sourced data.

    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: Our work does not involve crowdsourcing nor research with human subjects.

    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: We did not use LLMs for substantial, non-standard parts of this research.

    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.