Linear Transformer Topological Masking with Graph Random Features

GDL 2025
Group id: ——
Project id: ——

Paolo Deidda, Raffaele Perri, Paul Leopold Seipl
{paolo.deidda, raffaele.perri, paul.leopold.seipl}@usi.ch

Abstract
Concise and self-contained description of your project, motivation and main findings.

GENERAL NOTES

The report should be written as an article intended to present the findings of your work. Your aim should be to be clear and objective, substantiating your claims with references or empirical/theoretical evidence. We are well aware of the fact that carrying out machine learning experiments might be difficult and that often the final performance might be disappointing. For this reason, you will not be evaluated solely on quantitative aspect of your work, but mainly on the quality of your analysis and report. The length of the report should be between 4 and 8 pages (without considering references).

Contents

1 Introduction

Transformers have established themselves as a dominant architecture across various machine learning modalities, deriving their power from the attention mechanism which models complex dependencies between tokens [1]. However, the standard Transformer treats input data as a set, making it invariant to permutation and inherently unaware of structural dependencies, such as the connectivity in graph-structured data. To address this, *Topological Masking* is often employed, where the attention mechanism is modulated by a function of the graph structure (e.g., shortest path distance or adjacency), injecting a necessary structural inductive bias [2, 3].

A fundamental computational conflict arises when scaling this approach to large graphs. Standard "Vanilla" Softmax attention requires explicitly computing and storing an $N \times N$ attention matrix, resulting in quadratic $\mathcal{O}(N^2)$ time and space complexity. While Linear Attention mechanisms address this bottleneck by leveraging low-rank decompositions $\phi(Q)(\phi(K)^T V)$ to achieve $\mathcal{O}(N)$ complexity [4], they rely strictly on the associativity of matrix multiplication. Introducing a topological mask M generally breaks this associativity—since $(A \times B) \odot M \neq A \times (B \odot M)$—forcing the re-materialization of the dense attention matrix and negating the efficiency gains of linear attention.

In this work, we focus on the reproduction of "Linear Transformer Topological Masking with Graph Random Features" [5]. The authors propose a novel solution to the masking conflict by approximating the topological mask using Graph Random Features (GRFs). By decomposing the mask into sparse feature vectors derived from random walks, the method allows the mask to be fused with query and key features via a tensor product, preserving the $\mathcal{O}(N)$ complexity of linear attention while maintaining the expressivity of topological masking.

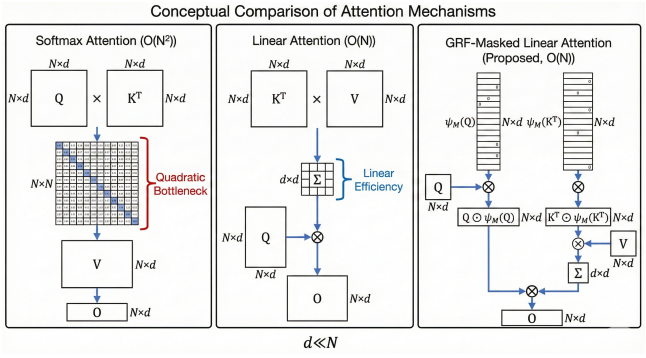

Figure 1. Conceptual comparison of attention mechanisms. The left panel illustrates the quadratic bottleneck of Softmax attention. The middle panel shows the efficient flow of Linear attention. The right panel demonstrates the proposed GRF method, which preserves linear complexity by fusing sparse topological features ψ_M with the query/key representations.

Contributions and Replication Results Our primary contribution is a validation of the theoretical and empirical claims presented in the original paper. We successfully implemented the proposed GRF-based masking mechanism and conducted a rigorous analysis of its computational scaling behavior.

- **Time Complexity Verification:** We replicated the scaling experiment comparing Softmax, Linear, and GRF-Masked attention. Our results confirm the theoretical claims: while Softmax attention scales quadratically, exhibiting a "computational explosion" for $N > 8$, the GRF-Masked method maintains strict linear scaling $\mathcal{O}(N)$.
- **Overhead Analysis:** We quantified the overhead introduced by the GRF mechanism. While retaining the linear complexity class, we observe a constant factor increase in FLOPs compared to unmasked linear attention, attributable to the sparsity of the generated graph features.

!!!TO DO: ADD THE OTHER TWO EXPERIMENTS RESULTS OVERVIEW!!!

2 Related works

Standard and Topological Attention The standard self-attention mechanism, introduced by [1], computes a dense $N \times N$ attention matrix $A = \text{softmax}(QK^\top/\sqrt{d})$. While highly effective at modeling global dependencies, this approach incurs quadratic $\mathcal{O}(N^2)$ time and space complexity, prohibiting its use on long sequences or large graphs. To adapt Transformers to graph-structured data, *topological masking* is often employed to inject structural inductive bias [2]. This typically involves modulating the attention scores with a mask $M(\mathcal{G})$ derived from the graph topology (e.g., shortest path distances), forcing tokens to attend preferentially to their structural neighbors. However, applying such masks generally requires materializing the full $N \times N$ matrix, maintaining the prohibitive quadratic bottleneck.

Linear Attention and The Masking Conflict To address the scalability limits of standard attention, *Linear Attention* mechanisms have been proposed [4, 6]. These methods replace the softmax kernel with a feature map decomposition $\phi(\cdot)$, allowing the computation to be reordered via associativity: $D^{-1}\phi(Q)(\phi(K)^\top V)$. This reduces complexity to $\mathcal{O}(N)$. However, as noted in the foundational literature, introducing an element-wise topological mask M breaks this associativity, i.e., $(A \times B) \odot M \neq A \times (B \odot M)$. Consequently, standard linear attention methods are generally incompatible with flexible topological masking without reverting to quadratic complexity.

Efficient Masking Approaches Several approaches have attempted to reconcile efficiency with masking, though often with restrictions on graph topology. [7] and [3] proposed using Toeplitz matrices and the Fast Fourier Transform (FFT) to implement masking in $\mathcal{O}(N \log N)$ time. While an improvement over $\mathcal{O}(N^2)$, these methods are largely restricted to highly structured graphs like grids or trees and do not generalize easily to arbitrary

topologies. Other methods, such as stochastic positional encoding [8], achieve linear complexity but are similarly limited to sequence data (1D grids). The method reproduced in this work [5] distinguishes itself by supporting general graphs with strict $\mathcal{O}(N)$ complexity via a randomized low-rank decomposition of the mask itself.

3 Methodology

3.1 Time Complexity

To validate the theoretical efficiency claims of the proposed architecture, the paper conducts a hardware-agnostic analysis of computational cost. Instead of measuring wall-clock time, which can be conflated by hardware specifics (GPU/CPU differences) and implementation overhead, the methodology focuses on counting the total number of Floating Point Operations (FLOPs) required for a single forward pass of the attention mechanism.
The analysis compares three distinct attention variants across a range of graph sizes (N):

- **Unmasked Softmax Attention:** The standard $\mathcal{O}(N^2)$ mechanism, where the full N×N attention matrix is materialized.
- **Unmasked Linear Attention:** An $\mathcal{O}(N)$ efficient alternative that utilizes a low-rank decomposition $\phi(Q)(\phi(K)^T V)$.
- **GRF-Masked Linear Attention (Ours):** The proposed method, which approximates the topological mask using Graph Random Features. While theoretically $\mathcal{O}(N)$, this method involves constructing sparse feature matrices, introducing a constant overhead dependent on the graph sparsity and number of random walkers.

The theoretical FLOP counts are derived based on the matrix dimensions involved. For a hidden dimension d and feature dimension m:

- **Softmax Attention (Baseline):** Dominated by the N^2 term from $QK^\top$ and AV product, roughly scaling as $\mathcal{O}(4N^2 d)$.
- **Linear Attention (Unmasked):** Linear in N, scaling as $\mathcal{O}(4Nmd)$ due to the associativity of matrix multiplication.
- **GRF-Masked Linear Attention (Ours):** Also linear in N, but the exact cost depends on the number of non-zero entries (NNZ) in the sparse graph random feature vectors. The complexity scales as $\mathcal{O}(4 \cdot \text{NNZ} \cdot d)$.

The experiment uses a 1-dimensional grid graph (a linear chain) to test scaling behavior. This topology allows for controlled testing of "local" attention, as the number of neighbors remains constant as N increases.

3.2 Vision Transformers (ViTs)

This section outlines the application of topological masking to Vision Transformers (ViTs), following the method proposed in the original paper. In a standard ViT, an image is divided into a sequence of fixed-size patches. To apply topological masking, we structure these patches as nodes in a graph $\mathcal{G}$, specifically a 2D grid graph where nodes are connected if they are spatial neighbors in the image.
The core objective is to modulate the attention mechanism A_{ij} between tokens (patches) i and j using a topological mask $M(\mathcal{G})_{ij}$, which encodes the structural relationship between them. While standard softmax attention requires $\mathcal{O}(N^2)$ complexity to apply this mask explicitly, the Graph Random Features (GRF) method allows for an implicit, efficient approximation compatible with linear attention.
For our experiments, we implemented several attention variants to evaluate the effectiveness of GRFs:

- **Unmasked Softmax:** The standard $\mathcal{O}(N^2)$ attention mechanism used as a high-accuracy baseline.
- **Unmasked Linear:** An $\mathcal{O}(N)$ efficiency baseline using the ReLU feature map $\phi(\cdot) = \text{ReLU}(\cdot)$ without any topological information.
- **Toeplitz-masked Linear:** An $\mathcal{O}(N \log N)$ baseline specific to grid graphs, where the mask decays based on the Manhattan distance between patches.
- **$M_\alpha(\mathcal{G})$-masked Linear:** An "exact" masking baseline that uses an explicit power series of the weighted adjacency matrix W to modulate attention. This represents the asymptotic limit of the GRF method as the number of random walkers $n \to \infty$.
- **GRF-masked Linear (Ours):** The proposed method where the topological mask is approximated stochastically. We simulate n random walks starting from each patch node on the grid graph. These walks define sparse feature vectors $\hat{\phi}_{\mathcal{G}}(v_i)$, which are used to construct the query and key feature matrices implicitly.

Since the grid graph structure is fixed for all images of the same resolution, the random walks and resulting sparse features are pre-computed and frozen before training, ensuring efficiency.

3.3 PCTS: Point Cloud Temporal State prediction

4 Implementation

4.1 Time Complexity

To verify the scalability claims, we implemented a standalone Python simulation to replicate Figure 3 from the original paper[5]. We did not rely on the authors' training code for this experiment; instead, we wrote a custom script to calculate theoretical FLOP counts and simulate the random walk sparsity exactly as described in the methodology.

Datasets: For this specific experiment, synthetic data was used. We procedurally generated 1-dimensional grid graphs (linear chains) with the number of nodes N ranging from 2^0 (1 node) to 2^{12} (4096 nodes). This covers the range from trivial graphs to those large enough to demonstrate the divergence between quadratic and linear scaling.

Hyperparameters: We adhered strictly to the hyperparameters reported in the paper to ensure an exact replication:

- Hidden Dimension (d): 8
- Feature Dimension (m): 8
- Number of Random Walkers (n): 4 per node
- Walk terminal probability (p_{halt}): 0.5

These small dimensions ($d = m = 8$) were likely chosen by the authors to isolate the scaling behavior (N) from the overhead of large feature vectors.

Experimental setup: The experiment was implemented in a Python script (`time_complexity_exp.py`) using NumPy.

- **Deterministic Counting:** For Softmax and Standard Linear attention, FLOPs were calculated using the direct analytical formulas ($4N^2d$ and $4Nmd$ respectively).
- **Stochastic Simulation:** For the GRF-Masked variant, the cost depends on the sparsity of the generated features. To measure this, we implemented a `simulate_unique_visits` function. This function simulates $n = 4$ random walkers starting from every node in the 1D grid. It tracks the set of unique nodes visited by the ensemble of walkers to determine the number of non-zero entries (NNZ) that would exist in the sparse feature matrix.
- **Variance Handling:** Because the random walks are stochastic, we averaged the unique visit counts over 10 independent trials for each graph size N.

Computational requirements: Since this experiment calculates theoretical operations rather than training a neural network, the computational cost was negligible. The simulation runs in seconds on a standard consumer CPU (e.g., Apple Silicon M1 or Intel i7) and requires minimal RAM (<1GB). No GPU resources were required for this specific validation step.

4.2 Vision Transformers (ViTs)

We re-implemented the ViT pipeline using PyTorch to reproduce the findings of the original paper on a smaller scale. While the original work utilized large-scale datasets like *ImageNet*, *iNaturalist*, *Places365*, we adapted the experimental setup to accessible, lower-resolution benchmarks to verify the algorithmic properties within a constrained computational budget.

Datasets: We evaluated the method on three standard image classification datasets:

- **CIFAR-10:** Consisting of 60,000, 32×32 color images in 10 classes.
- **CIFAR-100:** Similar to CIFAR-10 but with 100 classes.
- **FashionMNIST:** Consisting of 60,000 28×28 grayscale images in 10 classes.

Hyperparameters: Hyperparameters were chosen to accommodate the smaller model capacity required for these datasets while maintaining the relative structure of the original experiments.

Hyperparameter	CIFAR-10 (15 ep)	CIFAR-100 (30 ep)	CIFAR-10 (30 ep)	FashionMNIST (10 ep)	CIFAR-10 (Scaled)
Num. layers	2	2	2	2	4
Num. heads	4	4	4	4	4
Num. patches	8×8	8×8	8×8	8×8	10×10
Image Size	32×32	32×32	32×32	32×32	42×42
Hidden size	64	64	64	64	64
MLP dim.	128	128	128	128	128
Optimiser	Adam	Adam	Adam	Adam	Adam
Epochs	15	30	30	10	15
Learning rate	1×10^{-3}	1×10^{-3}	1×10^{-3}	1×10^{-3}	1×10^{-3}
Batch size	128	128	128	128	64
$\phi(\cdot)$	ReLU	ReLU	ReLU	ReLU	ReLU
Num. walks (n)	50	50	50	50	100
p_{halt}	0.1	0.1	0.1	0.1	0.1

Table 1. Architecture and training hyperparameters for all experimental runs.

Experimental Setup: The experiments were conducted on a single GPU environment (CUDA-enabled). The pipeline involved training five distinct model variants (Softmax, Toeplitz, M_α, GRF, Unmasked Linear) from scratch for each dataset. We utilized a custom GRFExactAttention module that pre-computes random walks on a NetworkX grid graph and registers them as buffers in the PyTorch model.

Computational Requirements: Due to the downscaled input resolution (32×32) and shallow architecture (2-4 layers), the computational load was lightweight respect to the original ViT experiments.

- **Training Time:** approximately 16.8 seconds per epoch on the GPU used.
- **Memory Usage:** The models fit comfortably within standard GPU memory constraints ($<$ 4GB).
- **Pre-computation:** The generation of random walks for the 8×8 patch grid was instantaneous and performed once at initialization.

4.3 PCTS: Point Cloud Temporal State prediction

5 Results

5.1 Time Complexity

The results of our replication confirms the time complexity claims made in the original paper. Figure 2 plots the calculated FLOPs (scaled by 10^6) against the number of graph nodes N on a log-log scale.

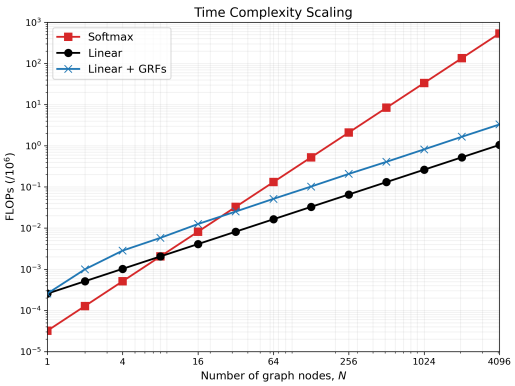

Figure 2. Time Complexity Replication Results: FLOPs vs Number of Nodes (N)

Table 2 details the specific computational costs at key intervals.

Graph Size (N)	**Softmax** (10^6 FLOPs)	**Linear** (10^6 FLOPs)	**GRF-Masked** (10^6 FLOPs)
1	3.20×10^{-5}	2.56×10^{-4}	2.56×10^{-4}
8 (Crossover)	$\mathbf{2.05 \times 10^{-3}}$	$\mathbf{2.05 \times 10^{-3}}$	6.48×10^{-3}
64	1.31×10^{-1}	1.64×10^{-2}	5.38×10^{-2}
512	8.39×10^{0}	1.31×10^{-1}	4.11×10^{-1}
4096	$\mathbf{5.37 \times 10^{2}}$	$\mathbf{1.05 \times 10^{0}}$	$\mathbf{3.30 \times 10^{0}}$

Table 2. Computational cost comparison across different graph sizes. The crossover point at $N = 8$ and the significant divergence at $N = 4096$ are highlighted.

Analysis of Scaling Regimes

- **Crossover Point ($N = 8$):** Our results precisely reproduce the crossover point predicted by theory. At $N = 8$, Softmax and Linear attention incur identical costs (2.05×10^{-3} MFLOPs). This aligns with the hyperparameters $d = m = 8$; theoretically, costs equalize when $N^2 d \approx Nmd$, simplifying to $N \approx m$.

- **Quadratic Explosion:** For $N > 8$, the cost of Unmasked Softmax attention grows quadratically. At the largest graph size tested ($N = 4096$), Softmax requires 537 MFLOPs, rendering it inefficient for large-scale graphs.

- **Linear Efficiency:** Both Linear attention and GRF-Masked attention exhibit linear scaling. At $N = 4096$, Unmasked Linear requires only 1.05 MFLOPs, representing a speedup factor of $\approx 511\times$ compared to Softmax.

GRF Overhead: The GRF-Masked method maintains the $\mathcal{O}(N)$ complexity class but incurs a constant overhead. At $N = 4096$, the GRF cost is 3.30 MFLOPs compared to 1.05 MFLOPs for standard Linear attention. This overhead ratio ($\approx 3.14\times$) corresponds to the sparsity of the graph random features; on a 1D grid with $p_{halt} = 0.5$, walkers visit an average of ≈ 3.1 unique nodes.

5.2 Vision Transformers (ViTs)

We present the results of our reproduction experiments, comparing the proposed GRF-masked linear attention against all the other variants.

Variant	**CIFAR-10 (15 ep)**	**CIFAR-100 (30 ep)**	**CIFAR-10 (30 ep)**	**FashionMNIST (10 ep)**	**CIFAR-10 (Scaled)**
Unmasked softmax	56.11	28.67	57.31	87.49	56.42
Toeplitz-masked linear	57.25	31.80	60.59	87.92	57.90
$\mathbf{M}_a(\mathcal{G})$-masked linear	**56.83**	**30.11**	**58.02**	**86.77**	**58.95**
Unmasked linear	56.69	31.59	58.52	87.41	59.58
GRF-masked linear	**56.69**	**31.99**	**60.37**	**87.11**	**58.93**

Table 3. Accuracies results for all ViT attention variants across different datasets and training epochs.

5.3 PCTS: Point Cloud Temporal State prediction

6 Discussion and conclusion

Here you can express your judgments and draw your conclusions based on the evidences produced on the previous sections.
Try to summarize the achievements of your project and its limits, suggesting (when appropriate) possible extensions and future works.

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[2] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? In *Advances in Neural Information Processing Systems*, volume 34, pages 28877–28888, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/f1c1592588411002af340cbaedd6fc33-Abstract.html.

[3] Krzysztof Choromanski, Han Lin, Haoxian Chen, Tianyi Zhang, Arijit Sehanobish, Valerii Likhosherstov, Jack Parker-Holder, Tamas Sarlos, Adrian Weller, and Thomas Weingarten. From block-toeplitz matrices to differential equations on graphs: Towards a general theory for scalable masked transformers. In *International Conference on Machine Learning*, pages 3962–3983. PMLR, 2022. URL https://proceedings.mlr.press/v162/choromanski22a.html.

[4] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*, pages 5156–5165. PMLR, 2020. URL https://proceedings.mlr.press/v119/katharopoulos20a.html.

[5] Isaac Reid, Kumar Avinava Dubey, Deepali Jain, William F Whitney, Amr Ahmed, Joshua Ainslie, Alex Bewley, Mithun George Jacob, Aranyak Mehta, David Rendleman, Connor Schenck, Richard E. Turner, René Wagner, Adrian Weller, and Krzysztof Marcin Choromanski. Linear transformer topological masking with graph random features. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=6MBqQLp17E.

[6] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. In *arXiv preprint arXiv:2009.14794*. PMLR, 2020. URL https://doi.org/10.48550/arXiv.2009.14794.

[7] Shengjie Luo, Shanda Li, Tianle Cai, Di He, Dinglan Peng, Shuxin Zheng, Guolin Ke, Liwei Wang, and Tie-Yan Liu. Table, fast and accurate: Kernelized attention with relative positional encoding. In *Advances in Neural Information Processing Systems*, pages 22795–22807. PMLR, 2021. URL https://doi.org/10.48550/arXiv.2106.12566.

[8] Antoine Liutkus, Ondřej Cífka, Shih-Lun Wu, Umut Simsekli, Yi-Hsuan Yang, and Gael Richard. Relative positional encoding for transformers with linear complexity. In *International Conference on Machine Learning*, pages 7067–707. PMLR, 2021. URL https://doi.org/10.48550/arXiv.2105.08399.