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

Here you should clarify the context of your project and the problem you are dealing with. You should also make a brief summary of the main results and contributions (i.e., if you tried to replicate the results of an existing paper you should say if you were successful or not). The introduction should help the reader to follow along for the rest of the paper.

2 Related works

Give a brief summary of (some) existing methods that are related to you project. For instance, you can refer to Reid et al. [1], or simply [1], for introducing Message Passing Neural Networks. In this section it is important to provide readers references to the current state of the art and the foundations of the presented method.
N.B.: When referencing a different approach, it is not necessary to provide a detailed description, only one/two brief sentences are enough. The interested readers can eventually read the referenced work.

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 VITS: Vision–Image Tasks (Graph + Image Integration) & Ablation Studies

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[1]. 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 ($4N^2 d$ 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 VITS: Vision–Image Tasks (Graph + Image Integration) & Ablation Studies

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 1 (reproduced below) plots the calculated FLOPs (scaled by 10^6) against the number of graph nodes N on a log-log scale.

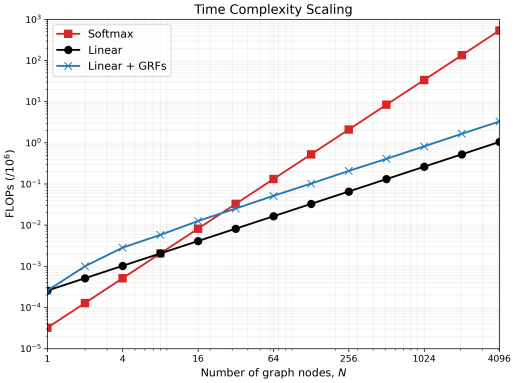

Figure 1. Time Complexity Replication Results: FLOPs vs Number of Nodes (N)

Table 1 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)	2.05×10^{-3}	2.05×10^{-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	5.37×10^{2}	1.05×10^{0}	3.30×10^{0}

Table 1. 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 VITS: Vision–Image Tasks (Graph + Image Integration) & Ablation Studies

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] 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.