Gamba: Scalable Attention Based Graph-to-Sequence Modeling

Róbert Veres [1] **Simon Storf** [1] **Adam Suma** [1] **Ben Bullinger** [1]

Abstract

Graph neural networks (GNNs) using iterative one-hop message passing often fail to leverage information from distant nodes, while graph transformers can attend to all nodes but lack inherent inductive biases and suffer from a quadratic time-complexity. We propose Graph Mamba (Gamba), a novel architecture generating a virtual sequence of tokens of fixed length by autoregressively aggregating node features globally and then processing this sequence with Mamba. Our approach achieves a comparable performance to state-of-the-art graph transformers on various benchmarks while significantly reducing computational costs. Code available at: https://github.com/BenBullinger/Gamba.

1. Introduction

Graph representation has advanced significantly over the past few decades, motivated by the universality of graphs as a data representation (Ma & Tang, 2021). Graphs naturally encode relationships in systems such as social networks, molecules and knowledge graphs, and several other data types can be transformed into graph shaped data (Xu, 2017).

Message Passing Neural Networks (MPNNs) have been the most common approach to graph learning over the past few years, explicitly aggregating information from each node's neighbors (Wu et al., 2021). However, these models face fundamental challenges such as oversmoothing (Rusch et al., 2023), oversquashing (Giovanni et al., 2024), and underreaching (Sun et al., 2022), with proven upper bounds on their expressivity (Xu et al., 2019; Zhang et al., 2024).

Graph Transformers address these limitations by attending to all node pairs, capturing global dependencies (Shehzad et al., 2024). However, their quadratic time complexity makes them poorly scalable for large graphs. Attempts to reduce this complexity often rely on kernel approximations (Moreno et al., 2022; Katharopoulos et al., 2020), which reintroduce limitations similar to those of classical MPNNs, sacrificing global context.

With the emergence of Mamba (Gu & Dao, 2023), state-space models regained attention as a compelling alternative to transformers, offering comparable expressive power while maintaining reduced computational complexity. Inspired by this, it is a natural extension to adapt state-space models for graph learning tasks. This adaptation has been explored in recent advancements like Graph-Mamba (GMB) (Wang et al., 2024) and Graph Mamba Network (GMN) (Behrouz & Hashemi, 2024), which integrate SSMs to graph structures via some predefined node ordering strategies. However, the explicitly defined order can be suboptimal for the problem (e.g. highly-irregular graphs ordered by their degree). Furthermore, computing an ideal ordering can be computationally prohibitive

In this work, we propose Gamba, a novel architecture that eliminates the need for an explicit node ordering by representing the graph as a sequence of virtual tokens.

2. Related Work

Mamba (Gu & Dao, 2023) enhances structured state space models (SSMs) by introducing a selective mechanism that dynamically adjusts parameters based on input, enabling content-aware reasoning. Unlike traditional linear time-invariant SSMs, Mamba filters relevant information while scaling linearly in sequence length. Its hardware-aware recurrent algorithm bypasses the convolutional constraints of earlier models.

Graph Mamba. Graph Mamba (GMB) (Wang et al., 2024) is, to our knowledge, the first extension of Mamba to graph-based tasks. It builds upon the GPS framework(Rampášek et al., 2023), replacing the Transformer module with Mamba. GMB flattens the graph using predefined metrics, such as node de-

[1]Department of Computer Science, ETH Zürich, Zurich, Switzerland. Correspondence to: <{rveres, sstorf, adsuma, bben}@ethz.ch>.

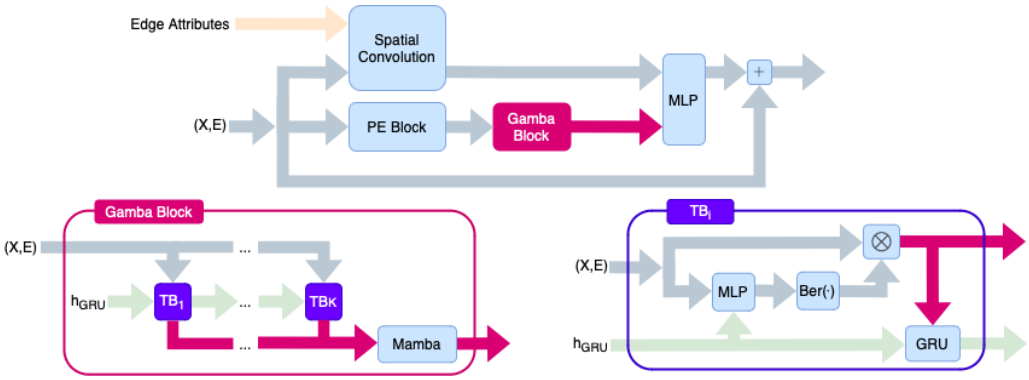

Figure 1. Gamba architecture.

grees. The Graph Mamba Network (GMN) (Behrouz & Hashemi, 2024) introduces a novel Neighborhood Sampling method to enhance the induction bias for graph structures. GMN also incorporates bidirectional Mamba layers but still relies on metric-based node ordering.

3. Models and Methods

In this section, we first provide an overview of how the three main building blocks of Gamba are connected, followed by a detailed explanation of each module.

3.1. Terminology

Given a graph $G(V, E)$, V denotes the set of nodes and E denotes the set of edges. Node features are stored in $X \in \mathbb{R}^{|V| \times D}$, where D denotes the hidden dimension size. We are aiming to generate the virtual tokens $t_1, t_2, ..., t_L$, each $\in \mathbb{R}^D$, where L denotes the length of the virtual token sequence. For the ease of notation the matrix $T \in \mathbb{R}^{L \times D}$ represents the virtual tokens stacked row-wise. The element-wise multiplication is denoted as $\odot$.

3.2. Overview

A simple overview can be seen in Figure 1. To avoid the introduction of explicit node ordering, we generate the virtual tokens based on the entire graph by taking a learned weighted sum of the nodes. That is, we can write the next token as $t_i = \sum_{v \in V} \alpha_\theta(i, v) \cdot x_v$. It can be compactly written as $T = AX$ where $A_{\theta,i,j} = \alpha_\theta(i, j)$. We detail the design choices of A later in section 3.3. To provide some structural insights to both the node selection, and the Mamba block, we concatenate some positional/structural encoding to the node features.

3.3. Token generation strategies

In the project description, we proposed a sequential token generation mechanism, where the aggregation weights for a given token depend on the features of the graph nodes and the sequence of previously generated tokens, functioning similarly to a cross-attention module. As highlighted by the reviewers, it is not trivial to parametrize the weight-coefficient matrix efficiently, as the number of past tokens are changing for each step. To address this, we propose two potential strategies to overcome this limitation.

Simple parallel token generation: Here, all tokens are generated simultaneously, avoiding any sequential dependency on the previously generated tokens. This simplification allows the entire Mamba pipeline to be reduced to a straightforward matrix multiplication followed by a convolution operation. The derivation of this reduction is provided in Appendix A

GRU-based generation: This approach aggregrates the previous tokens efficiently by passing the generated token into a GRU cell, with its hidden state acting as an aggregation of all previously generated tokens. The process is described as follows:

$$\alpha_\theta(i, v) = \Psi_\theta(x_v, h_{i-1}) \tag{1}$$

$$t_i = \sum_{v \in V} \alpha_\theta(i, v) \cdot x_v \tag{2}$$

$$h_i = \mathrm{GRU}(t_i, h_{i-1}) \tag{3}$$

Where Ψ in eq. (2) is an MLP.

3.4. Regularization of the Weight coefficients

Regardless of the generation strategy specified above, both strategies provide an overly expressive mechanism for the node aggregation that is prone to overfitting. To address this, once we computed α_i, we

Model	Peptides-Func (AP ↑)	Peptides-Struct (MAE ↓)	PascalVOC-SP (F1 score ↑)
GCN †	0.5930 ± 0.0023	0.3496 ± 0.0010	0.1268 ± 0.0060
GPS †	0.6575 ± 0.0049	0.2510 ± 0.0015	0.3689 ± 0.0131
Exphormer ‡	0.6527 ± 0.0043	0.2481 ± 0.0007	0.3430 ± 0.0108
GMB †	0.6739 ± 0.0087	0.2478 ± 0.0016	0.4191 ± 0.0126
GMN ‡	**0.7071 ± 0.0083**	**0.2473 ± 0.0025**	**0.4393 ± 0.0112**
GAT+V. nodes	0.5966 ± 0.0000	0.2754 ± 0.0015	0.0693 ± 0.0018
Gamba-simple	0.6285 ± 0.0020	0.2571 ± 0.0010	0.1135 ± 0.0064
Gamba-GRU	0.6507 ± 0.0035	0.2739 ± 0.0018	0.2179 ± 0.0057

Table 1. Benchmark of Gamba on Long-Range Graph Datasets compared to other models. **Best overall score** is marked in bold. Best model run by us is underlined. Results of models marked with † were taken from (Wang et al., 2024) and models marked with ‡ were taken from (Behrouz & Hashemi, 2024)

explore two regularization strategies[1]

Attention-like Regularization: We normalize the weights to behave like probabilities that sum to 1, similar to attention mechanisms. This discourages the model to apply higher than 1 or negative weights on some nodes, furthermore it helps the model select less, more important nodes over evenly spread attention.

$$t_i = \sum_{v \in V} \text{softmax}\left(\frac{\alpha_i}{\sqrt{D}}\right)_v x_v \tag{4}$$

Probabilistic Regularization: Instead of normalizing weights with softmax, we apply an elementwise sigmoid to α_i, interpreting each output as the probability of selecting a node independently. This approach allows the model to handle many-to-one relationships while introducing stochasticity, reducing the overfitting by decreasing the deterministic reliance on specific nodes.

$$t_i = \sum_{v \in V} \mathbb{1}(p_{i,v}) x_v \quad , \quad p_{i,v} \sim \text{Bernoulli}(\alpha_{i,v}) \tag{5}$$

3.5. Positional / Structural encodings

As the Mamba block solely works with node features, there is no structural information of the graph encoded inherently. To address this, we include Laplacian eigenvector (lPE) features and random-walk structural encodings (RWSE) for each node, enabling the model to better preserve and leverage the graph structure during processing. As GNNs can preserve graph structure better than earlier random-walk based approaches (Veličković et al., 2018; Wu et al., 2019), we alternatively use a multi-layer gated Graph ConvNet (Bresson & Laurent, 2018).

[1]Besides simple techniques such as MLP dropout and Weight Decay

4. Results

4.1. Dataset

We evaluate our model's performance on the Long Range Graph Benchmark (LRGB) Dataset (Dwivedi et al., 2023), which is specifically designed to assess a model's ability to capture long-range interactions in graph learning tasks. In this work, due to computational limitations we only use 3 of the provided datasets. *Peptides-func* is a multi-label graph classification dataset where the task is to predict peptide functions.*Peptides-struct* uses the same graphs as Peptides-func, but it's a graph regression dataset where the goal is to predict structural properties of peptides, like mass inertia and sphericity. *PascalVOC-SP* is a node classification dataset where each graph represents a segmented image, with nodes as superpixels characterized by RGB statistics and spatial coordinates, and edges weighted by boundary properties.

4.2. Performance benchmark

We evaluate our Gamba models by comparing them with three MPNNs: a simple GCN (Kipf & Welling, 2016), a Gated-GCN (Bresson & Laurent, 2018) and a Graph Attention Network (Veličković et al., 2017) working with virtual nodes as described in (Gilmer et al., 2017). For the latter, we treat the number of virtual nodes as a hyper-parameter, similarly to our models. We further compare it to two graph transformers, the original GPS (Rampášek et al., 2023) and Exphormer (Shirzad et al., 2023), a model that claims to scale better on larger graphs. Finally, we also compare our model to Graph Mamba (GMB) (Wang et al., 2024) and Graph Mamba Network (GMN) (Behrouz & Hashemi, 2024).

Table 1 compares the performance of our Gamba models with benchmark approaches. For each dataset and

Dataset	simple	simple + ATT	simple + Pr	GRU	GRU + ATT	GRU + Pr
P-Func	0.5920 ± 0.0310	0.60546 ± 0.0012	0.6285 ± 0.0020	0.6105 ± 0.0045	0.6080 ± 0.0027	**0.6502 ± 0.0035**
P-Struct	**0.2571 ± 0.0010**	0.2715 ± 0.0025	0.2760 ± 0.0033	0.2801 ± 0.0018	0.2788 ± 0.0004	0.2704 ± 0.0013
Pascal	0.1070 ± 0.0031	0.1035 ± 0.0044	0.1135 ± 0.0064	0.1164 ± 0.0044	0.1680 ± 0.0030	**0.2179 ± 0.0057**

Table 2. Ablation study on the regularization strategy. The models are evaluated with the two token generation strategies specified in section 3.3. With the proposed regularization methods from section 3.4. **Overall best model** is marked in bold, underlined scores are the best from the same aggregation technique

Gamba variant, we used the best-performing regularization strategy with GatedGCN-based positional encodings. Gamba outperforms MPNN-based methods, even those with virtual nodes, and achieves comparable results to state-of-the-art Graph Transformers. However, the poor performance on PascalVOC-SP suggests limitations in node prediction tasks, though these results should be interpreted with caution, as they are radically worse for every model than reported benchmarks. Additionally, while there is a convention of limiting the models to a 500k parameter budget, cited results were trained on A100 or V100 GPUs, whereas our experiments were constrained to an RTX3090 and the ETH cluster, limiting training duration.

4.3. Ablation study

We provide a full ablation study on the regularization techniques in Table 2. The GRU-based models consistently outperform the simple aggregation counterparts, which confirms that the sequential token generation mechanism can indeed provide a richer contextual understanding of the global structure. From the regularization techniques, the probabilistic regularization scored best almost everywhere, except on Peptides-struct with the parallel tokenization.

In Table 3, we compare our best-performing models using GatedGCN-based positional encodings against models with lPE and RWSE added to each node. The results show that GatedGCN consistently outperforms the classical lPE and RWSE approaches in every scenario by a significant margin.

4.4. Runtime Benchmark

Figure 3 shows the relative runtime change as the number of nodes increases. Experiment is done on Erdős–Rényi graphs with edge probability 0.1. Graph Transformers exhibit the steepest increase due to their fully connected architecture, especially beyond 8,000 nodes. In contrast, Gamba models scale more efficiently, with Gamba Simple showing near-loglinear growth. These results highlight Gamba's scalability for moderate graph sizes and the limitations of fully connected models for large graphs.

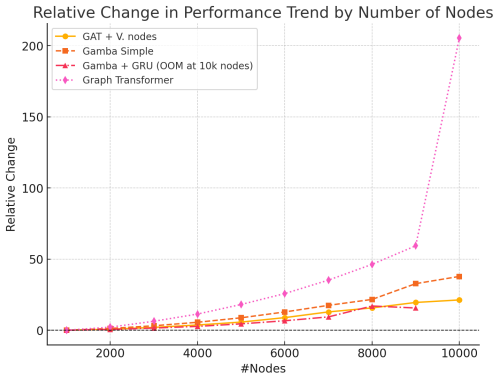

Figure 2. Relative change in performance trend by number of nodes.

5. Discussion

In this project, we present Gamba, a proof-of-concept Graph Mamba model that doesn't rely on explicit node ordering by using a novel weighted sum based sequence generation mechanism that aggregates global node features, enabling efficient and scalable graph-to-sequence modeling.

5.1. Future Work

Building on the insights from this project, we identify the following directions for future experimentations:

Dynamic Token Generation for Scalability: GRU-based models could dynamically adjust the number of virtual tokens in response to graph size, enhancing scalability. This approach allows training the model on smaller graphs with fewer virtual tokens to reduce computational overhead, while leveraging a larger number of tokens during inference on larger graphs to better capture complex global structures.

GPS framework integration: While Gamba was developed within its own framework, replacing the Transformer module in the GPS framework with Gamba would allow for a more direct comparison with GMB and GMN.

References

Behrouz, A. and Hashemi, F. Graph mamba: Towards learning on graphs with state space models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 119–130, 2024.

Bresson, X. and Laurent, T. Residual gated graph convnets, 2018. URL https://arxiv.org/abs/1711.07553.

Dwivedi, V. P., Rampášek, L., Galkin, M., Parviz, A., Wolf, G., Luu, A. T., and Beaini, D. Long range graph benchmark, 2023. URL https://arxiv.org/abs/2206.08164.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry, 2017. URL https://arxiv.org/abs/1704.01212.

Giovanni, F. D., Rusch, T. K., Bronstein, M. M., Deac, A., Lackenby, M., Mishra, S., and Veličković, P. How does over-squashing affect the power of gnns?, 2024. URL https://arxiv.org/abs/2306.03589.

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

Katharopoulos, A., Vyas, A., Pappas, N., and Fleuret, F. Transformers are rnns: Fast autoregressive transformers with linear attention, 2020. URL https://arxiv.org/abs/2006.16236.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Ma, Y. and Tang, J. *Deep Learning on Graphs*. Cambridge University Press, 2021.

Moreno, A., Wu, Z., Nagesh, S., Dempsey, W., and Rehg, J. M. Kernel multimodal continuous attention. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 18046–18059. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/727a5a5c77be15d053b47b7c391800c2-Paper-Conference.pdf.

Rampášek, L., Galkin, M., Dwivedi, V. P., Luu, A. T., Wolf, G., and Beaini, D. Recipe for a general, powerful, scalable graph transformer, 2023. URL https://arxiv.org/abs/2205.12454.

Rusch, T. K., Bronstein, M. M., and Mishra, S. A survey on oversmoothing in graph neural networks, 2023. URL https://arxiv.org/abs/2303.10993.

Shehzad, A., Xia, F., Abid, S., Peng, C., Yu, S., Zhang, D., and Verspoor, K. Graph transformers: A survey, 2024. URL https://arxiv.org/abs/2407.09777.

Shirzad, H., Velingker, A., Venkatachalam, B., Sutherland, D. J., and Sinop, A. K. Exphormer: Sparse transformers for graphs, 2023. URL https://arxiv.org/abs/2303.06147.

Sun, Q., Li, J., Yuan, H., Fu, X., Peng, H., Ji, C., Li, Q., and Yu, P. S. Position-aware structure learning for graph topology-imbalance by relieving under-reaching and over-squashing. In *Proceedings of the 31st ACM International Conference on Information amp; Knowledge Management*, CIKM '22, pp. 1848–1857. ACM, October 2022. doi: 10.1145/3511808.3557419. URL http://dx.doi.org/10.1145/3511808.3557419.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Veličković, P., Fedus, W., Hamilton, W. L., Liò, P., Bengio, Y., and Hjelm, R. D. Deep graph infomax, 2018. URL https://arxiv.org/abs/1809.10341.

Wang, C., Tsepa, O., Ma, J., and Wang, B. Graph-mamba: Towards long-range graph sequence modeling with selective state spaces. *arXiv preprint arXiv:2402.00789*, 2024.

Wu, F., Zhang, T., de Souza Jr., A. H., Fifty, C., Yu, T., and Weinberger, K. Q. Simplifying graph convolutional networks, 2019. URL https://arxiv.org/abs/1902.07153.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Yu, P. S. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, January 2021. ISSN 2162-2388. doi: 10.1109/tnnls.2020.2978386. URL http://dx.doi.org/10.1109/TNNLS.2020.2978386.

Xu, J. Representing big data as networks: New methods and insights, 2017. URL https://arxiv.org/abs/1712.09648.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks?, 2019. URL https://arxiv.org/abs/1810.00826.

Zhang, B., Gai, J., Du, Y., Ye, Q., He, D., and Wang, L. Beyond weisfeiler-lehman: A quantitative framework for gnn expressiveness, 2024. URL https://arxiv.org/abs/2401.08514.

A. Reducing the simple token generation to a convolution

In this section we show how our pipeline could rely on a single convolution by looking at a simple modification to the input of Mamba's convolution.

The SSM module in Mamba computes y_t given input token x_t as follows:

$$h_t = Ah_{t-1} + Bx_t \tag{6}$$
$$y_t = Ch_t$$

We thus note that we can directly compute y_t as given below:

$$y_t = C(Ah_{t-1} + Bx_t) \tag{7}$$
$$= C(A(Ah_{t-2} + Bx_{t-1}) + Bx_t) \tag{8}$$
$$= C(A\ (A(Ah_{t-3} + Bx_{t-2}) + Bx_{t-1}) + Bx_t) \tag{9}$$
$$\vdots$$
$$= C(\underbrace{A(A(A...(A}_{\text{t times}} Bx_0) + Bx_1) + Bx_2... + Bx_t \tag{10}$$
$$= CA^tBx_0 + CA^{t-1}Bx_1... + CABx_{t-1} + CBx_t \tag{11}$$
$$= \sum_i^t CA^tBx_{t-i} \tag{12}$$
$$= (x * K)_t \tag{13}$$

where K is a convolution kernel defined as $K := (CB, CAB, ..., CA^{L-1}B)$.

We take our generalized token generation formula as:

$$t_i = \sum_v \alpha_{v,t} \cdot x_v \tag{14}$$

For now, let's define $\alpha_i \in \mathbb{R}^{|V|}$ as the vector with the weights for each node for token t_i. Further, let $\alpha \in \mathbb{R}^{L\times|V|}$ be the matrix which stacks each α row-wise.

As we intend to pass these virtual tokens to mamba, we can use the new definitions to see that

$$y_k = (t * K)_k \tag{15}$$
$$\overset{eq.(11)}{=} t_0CA^{k-1}B + t_1CA^{k-2}B + ... + t_kCB \tag{16}$$
$$\overset{eq.(14)}{=} (\alpha_0^T \cdot X)CA^{k-1}B + (\alpha_1^T \cdot X)CA^{k-2}B + ... + (\alpha_k^T \cdot X)CB \tag{17}$$
$$= \alpha_0^T \cdot (XCA^{k-1}B) + \alpha_1^T(\cdot XCA^{k-2}B) + ... + \alpha_k^T \cdot (XCB) \tag{18}$$
$$= (\alpha^T \cdot X) * K)_k \tag{19}$$

Having this in mind, we can use a single convolution as in (eq. (19)) iff. α is known in advance. Such a model would be desired for its highly parallelized nature.

B. Experiment setup

We reported the average over 3 runs of a random seed, however due to technical difficulties, fixing the seeds do not produce the exact same results. Each experiment was run for at most 6 hours on an RTX3090. We provide the exact configurations in our github repository

C. Detailed Performance Trends

While the main paper discusses the relative performance (see Figure 2), here we include the absolute performance trends for reference (see Figure 3).

The Graph Transformer shows significant computational overhead due to its fully connected structure, particularly for larger graphs, as seen in the sharp increase in runtime for 10,000 nodes. On the other hand, Gamba and its variants maintain a near-loglinear increase in runtime, with Gamba + GRU reaching an out-of-memory (OOM) issue at this scale.

Refer to the relative performance analysis in the main text for additional context.

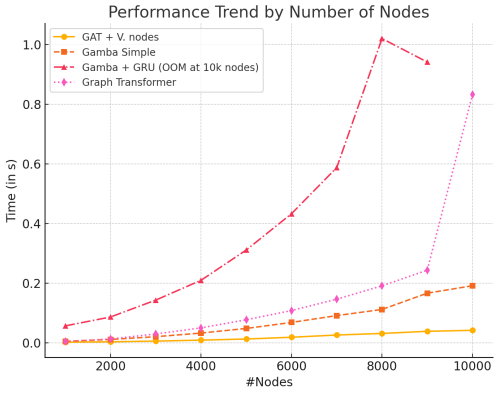

Figure 3. Runtime in seconds per nodes.

D. PE ablation table

Model Variant	Peptides-Func	Peptides-Struct	PascalVOC-SP
Simple+GGCN	**0.6285±0.0020**	**0.2571±0.010**	0.1135±0.0064
Simple+LP+RWSE	0.5947±0.0057	0.2680±0.0008	**0.1238**
GRU+GGCN	**0.6507±0.0035**	**0.2704±0.0031**	**0.2179±0.0057**
GRU+LP+RWSE	0.6199	0.2736	0.1302

Table 3. Ablation study on the positional encodings. **Best model from the same token generation technique** is marked in bold.