Generative Graph Pattern Machine

Zehong Wang[1], **Zheyuan Zhang**[1], **Tianyi Ma**[1], **Chuxu Zhang**[2], **Yanfang Ye**[1†]
[1]University of Notre Dame, [2]University of Connecticut
[†]Corresponding Author
<zwang43,yye7>@nd.edu

Abstract

Graph neural networks (GNNs) have been predominantly driven by message-passing, where node representations are iteratively updated via local neighborhood aggregation. Despite their success, message-passing suffers from fundamental limitations—including constrained expressiveness, over-smoothing, over-squashing, and limited capacity to model long-range dependencies. These issues hinder scalability: increasing data size or model size often fails to yield improved performance. To this end, we explore pathways beyond message-passing and introduce **G**enerative **G**raph **P**attern **M**achine (**G^{2}PM**), a generative Transformer pre-training framework for graphs. G^{2}PM represents graph instances (nodes, edges, or entire graphs) as sequences of substructures, and employs generative pre-training over the sequences to learn generalizable and transferable representations. Empirically, G^{2}PM demonstrates strong scalability: on the `ogbn-arxiv` benchmark, it continues to improve with model sizes up to 60M parameters, outperforming prior generative approaches that plateau at significantly smaller scales (e.g., 3M). In addition, we systematically analyze the model design space, highlighting key architectural choices that contribute to its scalability and generalization. Across diverse tasks—including node/link/graph classification, transfer learning, and cross-graph pretraining—G^{2}PM consistently outperforms strong baselines, establishing a compelling foundation for scalable graph learning. The code and dataset are available at `https://github.com/Zehong-Wang/G2PM`.

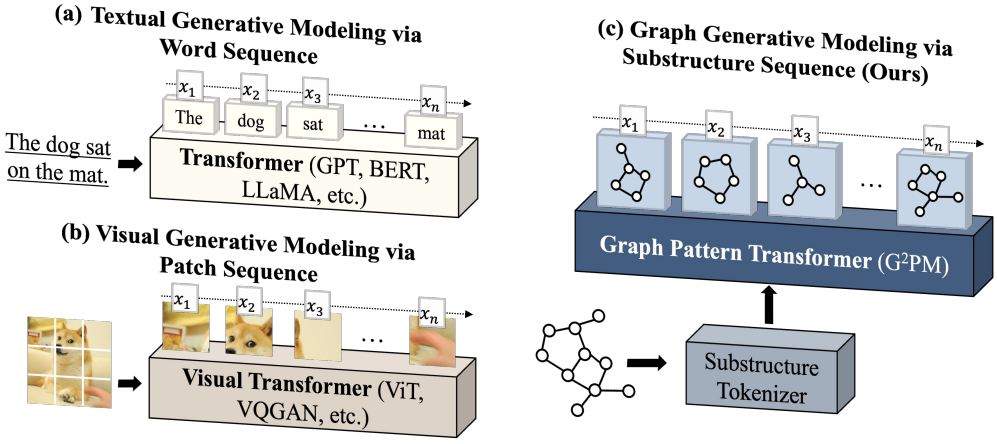


Figure 1: **Generative Transformer pre-training across modalities.** (a) *Textual modeling* tokenizes language into word sequences and generates next tokens or masked tokens. (b) *Visual modeling* slices images into patches and models generation in raster order. (c) *Graph modeling* (ours) tokenizes graph instances into substructure sequences using a random walk-based substructure tokenizer, and learns to generate via masked substructure modeling, going beyond message-passing.

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

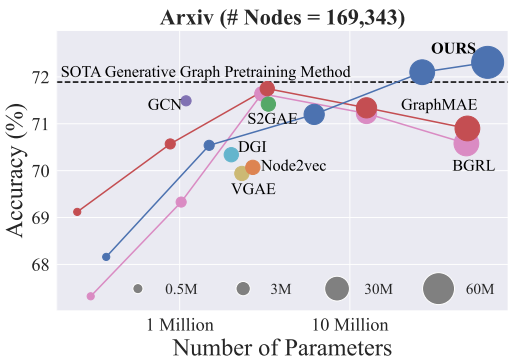


Figure 2: **Model scaling behavior** on the `ogbn-arxiv` with linear probing. G^{2}PM consistently improves as model parameters increase, achieving 72.31% accuracy with 60M parameters—surpassing the current SOTA generative graph pre-training method [23]. In contrast, GraphMAE [22] and BGRL [53] exhibit performance degradation, indicating limited scalability.

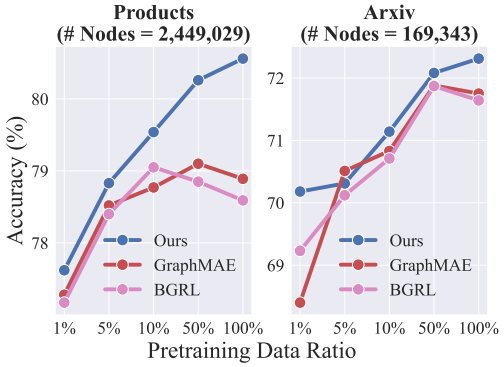


Figure 3: **Data scaling behavior** on the `ogbn-arxiv` and `ogbn-products` with linear probing. G^{2}PM demonstrates robust improvements with more pre-training data, reflecting superior scalability. In contrast, GraphMAE [22] and BGRL [53] peak at small data ratios and degrade with more data, suggesting overfitting, poor regularization, or inefficient data utilization.

1 Introduction

Transformer architectures [59] have emerged as the cornerstone of modern foundation models [7, 13, 2, 57]. Enabled by generative pre-training on massive unlabeled corpora [7, 12], Transformers learn transferable representations that can be efficiently adapted to diverse downstream tasks. This paradigm shift has redefined the landscape of machine learning, powering state-of-the-art systems in natural language processing and computer vision. Prominent examples include large language models (LLMs) [2, 57] and Vision Transformers (ViTs) [13], which showcase the scalability and generalization strength of this approach.

Despite the transformative success of generative Transformer pre-training in text and vision domains, this paradigm has yet to bring comparable breakthroughs in the graph domain, posing a significant barrier to the development of graph foundation models [40, 65, 67]. Most existing approaches to graph pre-training remain grounded in message-passing graph neural networks (MPNNs) [33, 60, 16, 18, 39], which are known to suffer from fundamental limitations: constrained expressive power [77], over-smoothing [49], over-squashing [56], and poor capacity for modeling long-range dependencies [47]. These challenges significantly limit the scalability of MPNNs [43], where increasing the size of the model or training data does not reliably lead to improved performance. As a result, the emergence of scaling laws in graph learning—a critical property in the success of foundation models [28]—remains elusive. Moreover, current graph pre-training techniques are predominantly based on contrastive learning [46, 83, 64, 63, 53, 84, 75, 76, 45, 38], which has been shown to be less capable of learning generalizable semantic representations compared to generative objectives [20, 12, 54, 5, 7]. This reliance further compounds the scalability bottleneck of graph neural networks (GNNs).

In this work, we aim to extend the success of Transformer-based generative pre-training to the graph domain, with the goal of enabling scalable graph representation learning. We consider the effectiveness of Transformers stems from a common principle: *Transformer-based models represent modality-specific instances using sequences of high-level semantic tokens, and apply generative objectives for pre-training*. To realize this paradigm for graphs, we begin by identifying three fundamental challenges that distinguish graph-structured data from Euclidean modalities such as text and images. (1) **Absence of Sequence Structure:** Unlike text or images, which naturally possess ordered or grid-like sequences compatible with Transformer training, graphs lack a sequence (no matter ordered or unordered) for nodes, edges, or subgraphs. This complicates the adaptation of Transformer. (2) **Semantic Granularity:** Graph elements—such as nodes or edges—typically encode low-level semantics, while tokens in language (i.e., words) or vision (i.e., patches) often correspond to higher-level concepts. It is unclear whether Transformers can effectively learn from such low-level representations in a generative fashion. (3) **Scalability Bottlenecks:** Existing Graph Transformers

(GTs) [34, 9, 14] often treat individual nodes as tokens and focus on pairwise relationships, leading to sequence lengths that scale with the number of nodes. Due to the quadratic time complexity of Transformers [29], this design choice limits scalability to large graphs, confining GTs primarily to small graphs such as molecular graphs [47].

These challenges naturally raise a central question: *how can we define sequences of high-level semantic tokens that meaningfully represent a graph instance—whether a node, an edge, or an entire graph?* To answer this, we revisit the core objective of graph learning: understanding key substructures that are predictive of downstream tasks. In many domains, such substructures are inherently semantic. For instance, motifs like triangles in social networks capture stable interpersonal relationships, while benzene rings in molecular graphs encode chemical stability—both serving as informative building blocks for reasoning. Motivated by this intuition, we propose to represent graph instances as sequences of meaningful substructures (Figure 1) and introduce the **G**enerative **G**raph **P**attern **M**achine (G^{2}PM), a Transformer-based generative pre-training framework for graphs. G^{2}PM tokenizes graphs into sequences of substructures via random walks, and learns representations by reconstructing masked substructures from context. This approach enables the design of pure Transformer models for graphs—free from message-passing—while unlocking scalability through increased model capacity and data volume (Figures 2 and 3). We highlight our contributions:

- We propose G^{2}PM, a generative Transformer pre-training framework that models graph instances as sequences of substructures, entirely eliminating the need for message passing.
- We introduce a random walk-based tokenizer that efficiently extracts semantic substructure patterns, and pair it with a masked substructure prediction task to enable self-supervised learning.
- We conduct a comprehensive design space exploration, offering actionable insights into model architecture and scalability.
- We demonstrate that G^{2}PM scales effectively: increasing data or model size yields consistent performance gains, echoing the scaling behaviors observed in other domains.
- We validate G^{2}PM across multiple benchmarks, showing state-of-the-art performance on node/link/graph-level tasks, along with strong generalization in cross-graph transfer tasks.

Outlook. While Transformers have redefined learning paradigms in many domains, their influence on graph learning remains nascent. We hope this work sparks further exploration into *non-message-passing* architectures, especially Transformers, as a new foundation for graph representation learning.

2 Generative Graph Pattern Machine

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X}, \mathbf{E})$ denote a graph, where $\mathcal{V}$ is the set of nodes with $|\mathcal{V}| = N$, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges with $|\mathcal{E}| = E$, and $\mathbf{X}$ and $\mathbf{E}$ represent the node and edge feature matrices, respectively. Each node $v \in \mathcal{V}$ is associated with a feature vector $\mathbf{x}_v \in \mathbb{R}^{d_n}$, and each edge $e \in \mathcal{E}$ is associated with a feature vector $\mathbf{e}_e \in \mathbb{R}^{d_e}$ (when applicable). As illustrated in Figure 4, G^{2}PM is pre-trained using a masked substructure modeling (MSM) objective in a fully self-supervised fashion.

2.1 Graph Representations

Our core idea is to represent any graph instance (node, link, or graph) as a sequence of substructures. In natural language, sequences are constructed using a predefined vocabulary of words or subwords; in vision, raster-scan tokenizers divide images into patch sequences. However, extending this notion of tokenization to graphs poses two key challenges. First, it is non-trivial to define a universal vocabulary of graph substructures, as the semantic relevance of patterns is highly domain-specific. For example, triangle motifs are prominent in social networks, whereas ring structures are more meaningful in molecular graphs. Second, tokenizing graphs by substructure matching incurs high computational cost: subgraph isomorphism is NP-complete [15], rendering it impractical for large-scale graphs—especially when compared to the linear-time tokenization of text and images [51, 13].

Tokenizer. To overcome these challenges, we adopt a random walk-based tokenizer that samples substructure patterns on the fly, bypassing the need for a predefined vocabulary. This strategy offers an efficient and scalable tokenization mechanism [68, 81]. As illustrated in Figure 4(b), a random

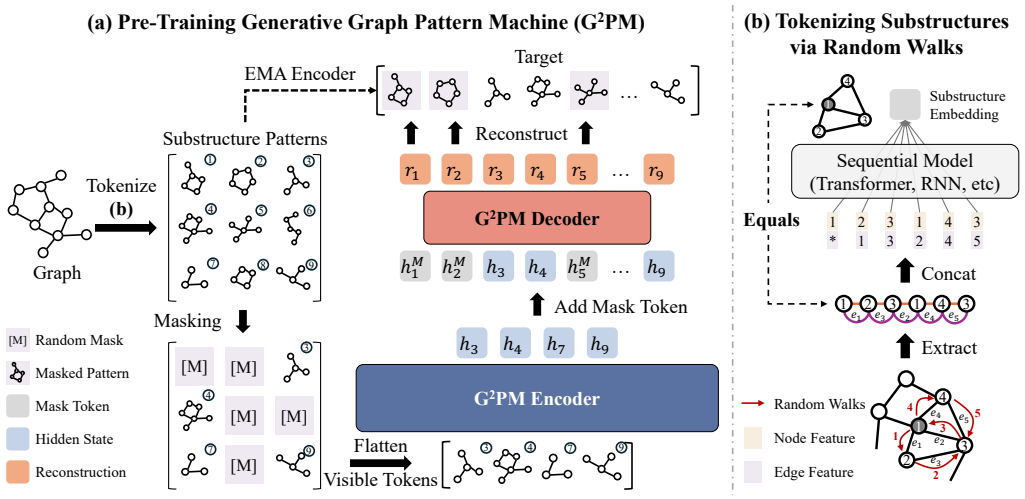


Figure 4: **Overview of G²PM pre-training.** (a) Given a graph instance, we first apply a random walk-based tokenizer to extract substructure patterns. Some patterns are randomly masked and the visible patterns are fed into a G²PM encoder. The encoder outputs are concatenated with special mask tokens and passed to a G²PM decoder to reconstruct the masked substructures. (b) We tokenize substructures by performing random walks over the graph, where each walk represents a substructure, which is proved to be effective [68]. The substructure embeddings are modeled via Transformer.

walk such as $[1, 2, 3, 1, 4, 3]$ corresponds to a diamond-shaped substructure. Formally, we define an unbiased random walk w of fixed length L as a sequence sampled from a Markov chain:

$$P(v_{i+1} \mid v_0, \ldots, v_i) = \frac{\mathbb{1}[(v_i, v_{i+1}) \in \mathcal{E}]}{D(v_i)}, \tag{1}$$

where $D(v_i)$ denotes the degree of node v_i. This transition probability enables efficient generation of node sequences from arbitrary starting points.

For each graph instance, we sample k random walks to serve as substructure patterns. To balance locality and global context, we leverage unbiased sampling strategies [44, 68]. Rather than constructing explicit subgraphs from these walks, we treat each as a node sequence and directly encode it using a Transformer, which has been shown to effectively capture structural inductive biases in graphs [68].

Given a node sequence $w = [\mathbf{x}_1, \ldots, \mathbf{x}_m]$, we compute its embedding via a Transformer encoder f:

$$\mathbf{p} = f(w) = f([\mathbf{h}_1, \ldots, \mathbf{h}_m]), \quad \mathbf{h}_i = \mathbf{W}\mathbf{x}_i + \mathbf{b}, \tag{2}$$

where $\mathbf{p}$ is the substructure embedding, and $\mathbf{x}_i = [\mathbf{x}_i \| \mathbf{e}_i]$ is the concatenation of node feature $\mathbf{x}_i$ and (optionally) edge feature $\mathbf{e}_i$.

Eliminating Message-Passing Bottlenecks. Our tokenizer is entirely message-passing-free, allowing G²PM to bypass key limitations of traditional GNNs. Notably, the random walk-based representations in G²PM effectively capture long-range dependencies [68, 27], exceed the expressiveness of 1-WL tests [55, 69, 42], and mitigate over-smoothing and over-squashing effects [61].

No Positional Embeddings. Unlike standard Transformers, which depend on positional encodings, we find that adding position information to node sequences offers no consistent benefit (Table 1b) and incurs cubic time complexity [34]. We thus omit positional embeddings entirely, simplifying the architecture without compromising performance.

Task-Agnostic Tokenization. Our tokenizer is applicable across graph-based tasks. For *node-level tasks*, random walks originate from the target node; for *edge-level tasks*, from the endpoints of target edge; and for *graph-level tasks*, from randomly sampled nodes in the graph. This task-agnostic design enables G²PM to adapt seamlessly without requiring specialized modifications.

2.2 G^{2}PM Backbone

We adopt the standard Transformer architecture [59] as the backbone for both G^{2}PM encoder and decoder. The input to the Transformer is a sequence of graph substructure tokens $\mathbf{P} = [\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n]$, where each $\mathbf{p}_i$ denotes the embedding of a sampled substructure. A single Transformer layer operates via a combination of self-attention and feed-forward networks, formally defined as:

$$\mathbf{P}' = \mathrm{FFN}\left(\mathbf{P} + \mathrm{Attn}(\mathbf{P})\right), \quad \mathrm{Attn}(\mathbf{P}) = \mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_{\text{out}}}}\right)\mathbf{V} \in \mathbb{R}^{n \times d_{\text{out}}}, \tag{3}$$

$$\mathbf{Q} = \mathbf{P}\mathbf{W}_{\mathbf{Q}}, \quad \mathbf{K} = \mathbf{P}\mathbf{W}_{\mathbf{K}}, \quad \mathbf{V} = \mathbf{P}\mathbf{W}_{\mathbf{V}}, \tag{4}$$

where $\mathbf{W}_{\mathbf{Q}}, \mathbf{W}_{\mathbf{K}}, \mathbf{W}_{\mathbf{V}}$ are learnable projection matrices and d_{out} denotes the dimensionality of the output embeddings. The FFN is implemented as a two-layer multilayer perceptron with non-linearity. Following standard practice, we employ multi-head self-attention [59], where multiple attention mechanisms are applied in parallel and their outputs concatenated. We stack multiple Transformer layers to enhance model capacity. We denote the output of the final (L-th) Transformer layer as $\mathbf{P}^L = [\mathbf{p}_1^L, \mathbf{p}_2^L, \ldots, \mathbf{p}_n^L]$, representing the learned contextualized embeddings of graph substructures.

2.3 Pre-Training G^{2}PM: Masked Substructure Modeling

To pre-train the G^{2}PM model, we introduce a generative Transformer objective *masked substructure modeling* (MSM). Given a sequence of substructures extracted from a graph, we randomly mask a subset and train the model to reconstruct the masked substructures conditioned on the visible ones. This process follows the conditional factorization principle used in masked language modeling [12]:

$$p(x_1, x_2, \ldots, x_n) = p(x_{/M}) \prod_{t \in M} p(x_t \mid x_{/M}), \tag{5}$$

where $x_{/M}$ denotes the set of visible tokens and M indicates the masked positions.

The key intuition is that meaningful graph substructures exhibit strong interdependencies—such as hierarchical relationships, functional reinforcement, or functional exclusion (Appendix A)—allowing certain substructures to be predictive of others. Concretely, given a tokenized substructure sequence $\mathbf{P} = [\mathbf{p}_1, \ldots, \mathbf{p}_n]$, we randomly retain k visible tokens to form $\mathbf{P}_{\text{vis}}$, while masking the remaining $n - k$ tokens. The visible sequence $\mathbf{P}_{\text{vis}}$ is passed through the G^{2}PM encoder to produce contextual embeddings $\mathbf{H}_{\text{vis}}$. We then insert learnable mask tokens at the masked positions to recover the full-length sequence, and feed the sequence into the G^{2}PM decoder for reconstruction $\mathbf{R} = [\mathbf{r}_1, \ldots, \mathbf{r}_n]$:

$$\mathbf{R} = \mathrm{Decoder}(\mathbf{H}), \quad \mathbf{H} = \mathrm{Add}_{[\mathrm{M}]}(\mathbf{H}_{vis}), \quad \mathbf{H}_{vis} = \mathrm{Encoder}(\mathbf{P}_{vis}). \tag{6}$$

To define the reconstruction targets, we leverage the high-level semantics of substructures. Instead of reconstructing low-level node features or adjacency matrices, we employ an *online encoder* f_{EMA}, an exponential moving average (EMA) version of the substructure encoder, to generate target embeddings $\hat{\mathbf{p}}_i = f_{\text{EMA}}(w_i)$ for each substructure walk w_i. This yields the training objective:

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n} \mathrm{is\_masked}(\mathbf{p}_i) \cdot \|\mathbf{r}_i - \mathrm{sg}[\hat{\mathbf{p}}_i]\|_2^2, \quad \hat{\mathbf{p}}_i = f_{\text{EMA}}(w_i), \tag{7}$$

where $\mathrm{is\_masked}(\cdot)$ is an indicator function for masked positions, and $\mathrm{sg}[\cdot]$ denotes the stop-gradient operator, ensuring that targets are fixed during training.

Adapting to Downstream Task. For downstream tasks, we discard the decoder and append a linear prediction head to the G^{2}PM encoder. Each graph instance is tokenized into a sequence of substructure embeddings $\mathbf{P} = [\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n]$, which is processed by the L-layer encoder to produce contextual representations $\mathbf{P}^L = [\mathbf{p}_1^L, \ldots, \mathbf{p}_n^L]$. We then apply mean pooling followed by the task head for final prediction $\hat{\mathbf{y}} = \mathrm{Head}\left(\frac{1}{n}\sum_{i=1}^{n} \mathbf{p}_i^L\right)$.

Augmenting Substructure Patterns. To improve robustness and representation quality, we adopt a mixed augmentation strategy inspired by corruption-agnostic pre-training [20, 71]. We define four augmentations in two categories: *Feature-level*—(1) feature masking (zeroing a subset of features) and (2) node masking (masking entire nodes); and *Structure-level*—(3) substructure corruption (dropping nodes within a walk) and (4) substructure injection (replacing walk nodes with random nodes). During training, we apply one feature-level and one structure-level augmentation per instance to promote generalization across diverse perturbations.

Dim	# Params	Arxiv	HIV
64	∼0.4M	68.2	69.7
128	∼1.5M	70.5	70.9
256	∼6.2M	71.2	74.1
512	∼26.6M	72.1	76.4
768	∼64.5M	**72.3**	**78.7**

(a) **Model dimension.** G^{2}PM exhibits strong scalability with increased hidden dimension sizes.

Encoder	Arxiv	HIV
Transformer	72.3	**78.7**
GRU	63.4	72.3
GIN	71.0	73.2
MEAN	70.3	70.4
+ Node PE	**72.4**	78.6

(b) **Tokenization.** Transformer is the most expressive encoder for modeling substructures.

[Mask]	Arxiv	HIV
Learnable	**72.3**	**78.7**
Fixed	71.8	76.9
Random	67.6	71.6
Sampling	67.1	74.0

(c) **Mask token.** Using a learnable mask token leads to higher accuracy compared to other tokens.

Aug.	Arxiv	HIV
Mixed	**72.3**	**78.7**
Feature mask $p = 0.2$	69.8	73.2
Node mask $p = 0.2$	70.9	75.5
Substructure corrupt $p = 0.8$	71.3	74.4
Substructure inject $p = 0.8$	72.1	72.4
None	70.8	73.9

(d) **Data augmentation.** Mixed augmentation significantly improves downstream performance.

Case	Arxiv	HIV
EMA Emb.	72.3	78.7
Feat. (mean)	71.4	73.5
Feat. (concat)	71.2	64.0
L2 to L1 loss	72.1	72.3
+ Topo. Recon.	**72.9**	**78.9**

(e) **Reconstruction target.** High-level target is more effective than feature-level reconstruction.

α	Update Every	Arxiv	HIV
0.9	10	70.8	74.3
0.99	10	**72.3**	**78.7**
0.999	10	72.1	76.6
0.99	5	72.2	78.1
0.99	20	**72.3**	76.6
w/o EMA Update		25.2	69.1

(f) **Online encoder.** Maintaining an EMA-updated online encoder is crucial in generating targets.

Table 1: **G^{2}PM ablation experiments** on ogbn-arxiv and ogbg-HIV. We report linear probe accuracy (%) and use gray to indicate default settings. The detailed hyper-parameters are in Appendix B.

3 Design Spaces and Insights

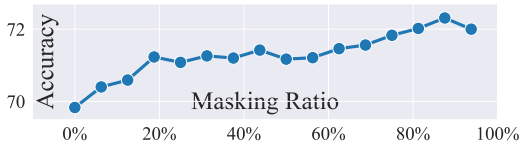
We conduct a comprehensive ablation study to provide more insights about the model design.

3.1 Model Architecture

Model Dimension. We set the default hidden dimension to 768, with a 3-layer encoder and a 1-layer decoder. As shown in Table 1a, increasing model size consistently improves performance, mirroring scaling trends observed in language and vision domains [7, 13]. Unlike message-passing models that often plateau with scale, G^{2}PM continues to benefit from increased parameterization.

Arch.	Enc.	Dec.	# Enc. Params	# Dec. Params	Arxiv	HIV
MAE	1	1	∼7.1M	∼7.1M	71.1	75.9
MAE	2	1	∼14.2M	∼7.1M	72.2	77.6
MAE	3	1	∼21.3M	∼7.1M	**72.3**	**78.7**
MAE	3	2	∼21.3M	∼14.2M	72.0	71.9
SimMIM	3	1	∼21.3M	∼0.6M	71.1	74.7
SimMIM	3	2	∼21.3M	∼1.2M	71.7	74.1

(a) **Model architecture.** Our G^{2}PM design (MAE-style [20]) outperforms SimMIM-style [71] pre-training.



(b) **Masking ratio.** A large masking ratio leads to better performance with more challenging tasks.

Figure 5: **More G^{2}PM ablation experiments**.

Model Layer. Table 5a shows that deeper encoders enhance performance, while increasing decoder capacity provides limited or negative returns, particularly under the MAE-style masked modeling framework. We attribute this to the simplicity of substructure reconstruction, where expressive decoders can lead to overfitting. Replacing the MAE-style design [20] with a SimMIM-style variant [71] further degrades performance. This suggests that MAE-style sparsity encourages the encoder to learn stronger context-aware representations, which better align with the inductive bias of graph data.

3.2 Reconstruction Target

Target. By default, G^{2}PM reconstructs the semantic embedding of each masked substructure generated by an EMA-updated encoder. As shown in Table 1e, this choice consistently yields the strongest performance, supporting our hypothesis that substructures encode transferable semantic information. We compare this approach to two low-level alternatives. Specifically, we represent each substructure as a node sequence $w = [x_1, \ldots, x_n]$, where each $x_i = [\mathbf{x}_i \| \mathbf{e}_i]$ is the concatenation of node and optional edge features. We experiment with (1) mean-pooling the node features over the walk, and (2) concatenating them directly. Both approaches underperform the semantic embedding target, suggesting that low-level supervision lacks the abstraction necessary for graph generalization.

Loss. We also evaluate the impact of loss functions. Replacing ℓ_2 loss with ℓ_1 leads to a notable drop in graph classification accuracy, suggesting that outlier-sensitive objectives better capture nuanced substructure semantics. Additionally, we incorporate anonymous walks [24] following Wang et al. [68], using a parallel head to reconstruct topological patterns. This auxiliary objective consistently boosts performance, reinforcing the utility of substructures as fundamental modeling units.

Online Encoder. We ablate the momentum α and update the frequency of the EMA encoder (Table 1f). Our default setting ($\alpha = 0.99$, update every 10 steps) performs best. Lower momentum values degrade performance, with $\alpha = 0$ (i.e., no EMA) leading to divergence and collapse, highlighting the importance of stable, slowly evolving targets during training.

3.3 Tokenization

Substructure Encoder. To embed substructure sequences (e.g., random walks), we adopt a Transformer encoder, which models global dependencies among all nodes in the sequence. As shown in Table 1b, Transformer outperforms alternatives such as mean pooling, GRU [11], and GIN [73]. We attribute this to several factors: (1) mean pooling is a restricted, less expressive form of attention in Transformer; (2) GRUs suffer from optimization instability due to vanishing or exploding gradients [85]; and (3) using GIN requires converting walks into subgraphs, reintroducing message-passing and its known limitations.

Positional Embedding. We also explore adding positional embeddings (PEs) to node features—using Laplacian PE on ogbn-arxiv and random walk PE on ogbg-HIV [47]. However, we observe no consistent performance gain. This may be due to the tokenization scheme: since each token represents an entire substructure, global node positions are less relevant. Moreover, computing PEs incurs cubic time complexity in the node number [34], making them impractical for large-scale graphs.

3.4 Augmentation

Table 1d reports the impact of various augmentation strategies. In general, augmentation improves performance, with the mixed strategy yielding the best results, highlighting the value of combining perturbations from different perspectives. We further analyze individual augmentation types. Among feature-level augmentations, node masking outperforms feature masking, likely because reconstructing full-node semantics from context is more challenging—and thus more informative—than recovering masked dimensions. For structure-level perturbations, substructure corruption performs better on graph-level tasks (e.g., molecular graphs), while substructure injection is more effective on node-level tasks (e.g., academic networks). This distinction stems from how each augmentation impacts semantics: corruption distorts but retains the original context, whereas injection alters substructure identity, which may hurt graph-level tasks where substructure semantics are functionally meaningful. Notably, using only a single augmentation type yields limited improvements, reinforcing the importance of diversity in corruption strategies for generalizable representation learning.

3.5 Masking

Masking Ratio. Figure 5b shows that a high masking ratio consistently improves performance, echoing trends observed in vision [20]. This is likely because random walks often produce redundant or noisy substructures. High masking reduces such redundancy, resulting in a more challenging and informative learning signal that encourages the model to capture high-level semantics.

Masking Token. The choice of masking token significantly affects performance (Table 1c). By default, we use a single learnable mask token, which outperforms alternatives. Fixed tokens degrade performance due to their inflexibility, while random tokens or sampled embeddings from other substructures introduce noise that confuses the model. These findings suggest that a consistent, learnable mask token provides the strongest and cleanest supervision signal during reconstruction.

4 Comparisons to State-of-The-Art Models

Node Classification on homophily graphs. We evaluate G^2PM on a suite of homophily graphs of varying scales, including Pubmed, Photo, Computers, WikiCS, Flickr, ogbn-arxiv, and ogbn-products (see Table 2 for dataset statistics). Pre-training is conducted on the full graph. We adopt a linear probe setup: node embeddings are frozen after pre-training and used to train a separate classifier, where we take 10/10/80 random split for Pubmed, Photo, and Computers, and the official split for the remaining datasets. Accuracy is used as the evaluation metric. We compare G^2PM to supervised GAT [60], non-message-passing GPM [68], contrastive methods (GCA [84], BGRL [53], CCA-SSG [78]), and generative methods (GraphMAE [22], GraphMAE 2 [23], S2GAE [52], Bandana [82]).

Table 2: **Node classification** results on homophily graphs. **Boldface** and underline indicate the best and sub-best self-supervised methods, and A.R. is the average ranking.

		Pubmed	Photo	Computers	WikiCS	Flickr	Arxiv	Products	A.R.
	# Nodes	19,717	7,650	13,752	11,701	89,250	169,343	2,449,029	-
	# Edges	88,648	238,162	491,722	431,206	899,756	2,315,598	123,718,024	-
Supervised	GAT [60]	83.1 ± 0.3	91.9 ± 0.5	87.9 ± 0.5	76.9 ± 0.8	50.7 ± 0.3	72.10 ± 0.13	79.45 ± 0.59	5.7
	GPM [68]	84.7 ± 0.1	92.7 ± 0.3	90.0 ± 0.4	80.2 ± 0.4	52.2 ± 0.2	72.89 ± 0.68	82.62 ± 0.39	1.3
Contrastive	GCA [84]	83.3 ± 0.5	92.4 ± 0.2	87.1 ± 0.2	77.4 ± 0.1	49.0 ± 0.1	71.23 ± 0.09	78.39 ± 0.03	6.9
	BGRL [53]	83.9 ± 0.3	92.5 ± 0.2	88.2 ± 0.2	77.5 ± 0.8	49.7 ± 0.2	70.51 ± 0.03	78.59 ± 0.02	5.7
	CCA-SSG [78]	81.8 ± 0.5	91.8 ± 0.6	88.6 ± 0.3	75.3 ± 0.8	47.5 ± 0.2	71.24 ± 0.20	75.27 ± 0.05	8.6
Generative	GraphMAE [22]	81.0 ± 0.5	92.0 ± 0.3	**89.2 ± 0.5**	77.1 ± 0.5	50.5 ± 0.1	71.75 ± 0.17	78.89 ± 0.01	6.0
	GraphMAE 2 [23]	81.3 ± 0.4	92.4 ± 0.2	88.3 ± 0.9	77.6 ± 0.4	50.4 ± 0.1	71.89 ± 0.03	79.33 ± 0.01	5.4
	S2GAE [52]	80.1 ± 0.5	91.4 ± 0.1	85.3 ± 0.1	75.3 ± 0.8	48.1 ± 0.8	67.77 ± 0.36	76.70 ± 0.03	10.3
	Bandana [82]	83.5 ± 0.5	91.4 ± 0.7	87.7 ± 0.2	77.3 ± 0.3	47.9 ± 0.6	71.09 ± 0.24	77.68 ± 0.05	8.1
	G^2PM w/o Pretrain	83.9 ± 0.2	92.8 ± 0.2	87.1 ± 0.3	78.5 ± 0.4	50.7 ± 0.1	69.64 ± 0.08	76.90 ± 0.16	6.0
	G^2PM	**84.3 ± 0.1**	**92.9 ± 0.2**	88.8 ± 0.3	**79.0 ± 0.4**	**51.0 ± 0.0**	**72.31 ± 0.07**	**80.56 ± 0.01**	2.0

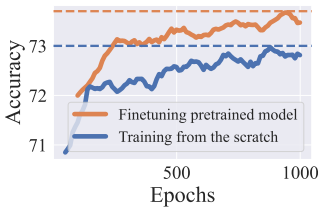


Figure 6: **Convergence curves** under fully finetuning on Arxiv dataset. We compare training from scratch versus fine-tuning the pre-trained model.

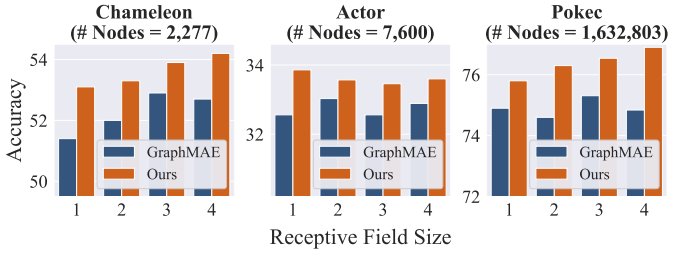


Figure 7: **Node classification** results on heterophily graphs under varying receptive field sizes. For GraphMAE, the receptive field is controlled by the number of model layers; in our method, it is controlled by the random walk length, where a model layer of size k corresponds to a walk length of 2^k.

As shown in Table 2, G^2PM outperforms all self-supervised baselines and ranks second overall—only behind the supervised GPM—achieving an average rank of 2.0. Notably, on the large-scale ogbn-products dataset, G^2PM achieves a substantial performance gain (80.56 vs. 79.33), highlighting its scalability. Ablation shows that on small graphs ($\leq$ 100K nodes), the model performs well even without pre-training, likely due to its sufficient capacity. However, on larger datasets like ogbn-arxiv and ogbn-products, pre-training proves crucial for capturing fine-grained structural patterns.

Node Classification on Heterophily Graphs. We evaluate G^2PM on heterophily graphs: Chameleon, Actor, and Pokec (over 1M nodes). Results are shown in Figure 7. We compare against GraphMAE, varying the receptive field via the number of GNN layers (for GraphMAE) and random walk length (for G^2PM). G^2PM consistently outperforms GraphMAE across datasets and benefits from increased receptive field size, demonstrating its ability to capture non-local information critical for heterophilous settings. On Actor, however, performance shows no clear correlation with receptive field, suggesting that key signals in this collaboration network are primarily localized.

Link Prediction. To further demonstrate the effectiveness of our pretraining approach, we follow Wang et al. [68] to show the link prediction results. We use three datasets, including Cora, Pubmed, and ogbl-collab, and follow a standard 80/5/15 split for training, validation, and test sets. For evaluation, we report Hit@20 on Cora and Pubmed, and Hit@50 on ogbl-collab. The results are summarized in Table 3, where we compare our method G^2PM with a basic supervised GCN [33] and a widely used graph pretraining method, GraphMAE [22]. As shown, GraphMAE significantly outperforms GCN, benefiting from its ability to utilize unlabeled data during pretraining. Notably, G^2PM achieves the best performance across all datasets, likely due to its capacity to model substructures that capture latent connectivity patterns between nodes.

Table 3: **Link prediction results** on three widely-used benchmarks.

	Cora	Pubmed	ogbl-Collab
# Nodes	2,708	19,717	235,868
# Edges	10,556	88,648	2,570,930
Metric	Hit@20	Hit@20	Hit@50
GCN [33]	84.1 ± 1.2	85.1 ± 3.8	44.8 ± 1.1
GraphMAE [22]	87.6 ± 1.4	87.1 ± 3.1	46.5 ± 0.9
G^2PM	**90.4 ± 0.8**	**88.0 ± 3.8**	**47.1 ± 0.6**

Table 4: **Graph classification** results on molecular and social networks. **Boldface** and underline indicate the best and sub-best self-supervised methods, and A.R. is the average ranking.

		HIV	PCBA	Sider	MUV	ClinTox	IMDB-B	REDDIT-M12K	A.R.
	# Graphs	41,127	437,929	1,427	93,087	1,478	1,000	11,929	-
	# Nodes	∼25.5	∼26.0	∼33.6	∼24.2	∼26.1	∼19.8	∼391.4	-
	# Edges	∼27.5	∼28.1	∼70.7	∼52.6	∼55.5	∼193.1	∼913.8	-
Supervised	GIN [73]	75.8 ± 0.8	70.3 ± 0.3	57.7 ± 0.8	74.4 ± 0.9	83.4 ± 0.6	73.3 ± 0.5	39.4 ± 1.4	6.3
	GPM [68]	77.0 ± 0.9	75.1 ± 0.3	59.0 ± 0.0	74.6 ± 1.4	82.4 ± 0.3	82.7 ± 0.5	43.1 ± 0.3	3.0
Contrastive	GraphCL [75]	75.5 ± 0.3	72.4 ± 2.1	57.3 ± 0.9	68.3 ± 2.6	82.9 ± 0.3	71.1 ± 0.4	37.9 ± 2.4	8.0
	JOAO [76]	76.8 ± 0.3	73.4 ± 1.5	58.5 ± 0.5	72.3 ± 1.0	82.2 ± 0.3	70.2 ± 3.1	39.9 ± 0.6	6.0
	MVGRL [19]	75.7 ± 0.7	70.4 ± 2.1	60.5 ± 0.6	71.5 ± 1.2	83.6 ± 0.2	74.2 ± 0.7	39.5 ± 1.8	5.7
	InfoGCL [72]	77.3 ± 0.6	74.6 ± 0.7	58.7 ± 0.7	73.4 ± 1.0	80.3 ± 0.7	75.1 ± 0.9	39.3 ± 0.5	5.4
Generative	GraphMAE [22]	77.8 ± 0.9	73.2 ± 1.4	60.6 ± 0.0	73.7 ± 0.8	84.8 ± 0.5	75.5 ± 0.7	37.6 ± 2.5	4.1
	S2GAE [52]	75.6 ± 0.8	72.9 ± 0.0	58.0 ± 0.9	71.6 ± 0.8	80.6 ± 0.4	75.8 ± 0.6	37.9 ± 1.8	7.0
	G^2PM w/o Pretrain	69.8 ± 0.1	68.4 ± 0.0	58.8 ± 0.3	66.3 ± 1.4	80.0 ± 1.8	80.0 ± 0.8	37.5 ± 0.3	8.3
	G^2PM	**78.7 ± 0.1**	**75.6 ± 0.1**	**61.2 ± 0.2**	**75.7 ± 0.4**	**86.6 ± 0.8**	**83.0 ± 0.8**	**41.8 ± 0.3**	1.0

Table 5: **Cross-domain transferability** performance across diverse source and target datasets. Parentheses indicate the performance gap compared to training from scratch on the target graph.

Source	**Arxiv**		**HIV**	
Target	**Products**	**HIV**	**Arxiv**	**PCBA**
GNN [60, 73]	78.3 (1.2 ↓)	70.1 (5.7 ↓)	71.1 (1.0 ↓)	71.9 (**1.6** ↑)
GPM [68]	82.0 (0.6 ↓)	74.3 (2.7 ↓)	71.4 (1.5 ↓)	76.4 (**1.3** ↑)
BGRL [53]	78.8 (**0.2** ↑)	72.5 (3.8 ↓)	68.6 (1.9 ↓)	72.9 (0.6 ↓)
GraphMAE [22]	77.5 (1.4 ↓)	74.7 (3.1 ↓)	69.9 (1.9 ↓)	73.4 (**0.2** ↑)
G^2PM	**81.3** (**0.7** ↑)	**76.8** (1.9 ↓)	**72.6** (**0.3** ↑)	**77.9** (**2.3** ↑)

Table 6: **Cross-domain pre-training** results on text-attributed graphs processed by [37], where node features are aligned via a textual encoder.

Pretrain	**Arxiv + FB15K237 + ChemBL**		
Downstream	**Arxiv (Academia)**	**FB15K237 (Knowledge Graph)**	**HIV (Molecule)**
BGRL [53]	70.8 ± 0.2	86.5 ± 0.3	68.5 ± 1.6
GraphMAE [22]	70.3 ± 0.3	87.8 ± 0.4	64.1 ± 0.5
OFA [37]	71.4 ± 0.3	84.7 ± 1.3	72.0 ± 1.6
GFT [65]	71.9 ± 0.1	**89.3 ± 0.2**	72.3 ± 2.0
G^2PM	**72.5 ± 0.1**	88.9 ± 0.5	**74.1 ± 1.3**

Convergence Curves. Figure 6 compares the training dynamics of models trained from scratch versus those finetuned from a pre-trained G^2PM checkpoint on ogbn-arxiv. Pre-training leads to consistently better performance, indicating that structural knowledge acquired during pre-training accelerates convergence and improves generalization. As expected, accuracy improves with additional training epochs in both settings.

Graph Classification. We evaluate G^2PM on seven datasets: five molecular graphs (HIV, PCBA, SIDER, MUV, ClinTox) and two social networks (IMDB-B, Reddit-M12K). Dataset statistics and results are summarized in Table 4. We use public splits for molecular graphs and 80/10/10 random splits for social networks, following a linear probe protocol. Baselines include supervised GAT [60], non-message-passing GPM [68], contrastive methods (GraphCL [75], JOAO [76], MVGRL [19], InfoGCL [72]), and generative models (GraphMAE [22], S2GAE [52]). G^2PM consistently achieves the best performance across all datasets. In contrast, the model without pre-training performs worst on average, highlighting the importance of capturing domain-specific substructure distributions to support discriminative generalization.

5 Cross-Domain Graph Learning

Cross-Domain Transfer. We evaluate the cross-domain transferability of G^2PM in Table 5, measuring generalization under distribution shifts across domains and tasks. In this setting, the model is pre-trained on a source graph and fully fine-tuned on a target graph. Since feature spaces differ across graphs, we introduce a learnable linear projection layer before the pre-trained encoder, which is jointly finetuned during transfer. Baselines include GNNs (GAT [60] for ogbn-arxiv and GIN [73] for HIV), as well as GPM [68], BGRL [53], and GraphMAE [22]. G^2PM achieves the best transfer results, showing positive gains in 3 out of 4 setups. In contrast, message-passing methods only transfer well across closely related domains (e.g., HIV → PCBA). We attribute this to the ability to learn transferable structural patterns, enabling generalization even across domain and task boundaries (e.g., Arxiv → HIV), while message-passing remains sensitive to subtle structural shifts [66].

Cross-Domain Pre-Training. Table 6 presents results on cross-domain pre-training using three diverse graph types: Arxiv (academic network), FB15K237 (knowledge graph), and HIV (molecular graph) [37]. All graphs are text-attributed; we use a shared textual encoder to project node descriptions

into a unified embedding space, enabling a single model to operate across domains. We compare G^2PM to pre-training baselines (BGRL [53], GraphMAE [22]) and graph foundation models (OFA [37], GFT [65]). G^2PM achieves the best or second-best performance across all datasets, highlighting its superior ability to learn transferable substructure representations, even under significant domain shifts—where message-passing-based methods often struggle.

6 Related Works

Graph Transformers. Inspired by the success of Transformers in vision and language [12, 7, 57, 20, 13], several works have extended this architecture to graphs [34, 14, 47, 21, 9, 8, 79, 80]. Typically, graph Transformers treat nodes as tokens and apply self-attention over all pairs. However, the quadratic complexity of node attention limits their scalability, making them impractical for large graphs [70]. To address this, recent works propose tokenizing graphs via substructures such as random walks or motifs [74, 24], enabling sequence-based modeling [68, 9, 21, 30]. For example, GPM [68] applies a ViT-style architecture over substructure sequences. Building on this idea, our work explores generative pretraining using a Transformer over substructures, entirely removing message passing.

Generative Pretraining. Generative pretraining has driven major advances in vision and language. These methods predict masked content from visible context, enabling learning from large-scale unlabeled data and powering modern foundation models [57]. This paradigm has extended to other domains (e.g., video, audio, biology), but its impact on graphs remains limited. More recent works apply generative approaches to graphs (e.g., VGAE [32], GraphMAE [22], GraphMAE2 [23], MaskGAE [35], G2PT [10]), but focus on reconstructing low-level signals such as nodes or links, and typically rely on message-passing GNNs, limiting their expressiveness and scalability. In contrast, G^2PM introduces a fully Transformer-based, message-passing-free framework that models high-level semantic substructures, unlocking better scalability and generalization.

Random Walks on Graphs. Random walks have been widely used to represent substructures, from early unsupervised embeddings [44, 17, 74, 24] to recent deep models [68, 81, 27, 41, 55, 26, 25, 31]. These methods have been shown to break key limitations of message passing, improving expressiveness [69, 42], modeling long-range dependencies [41, 55], and mitigating over-smoothing and over-squashing [62]. However, these random walk-based methods often focus on global patterns while overlooking localized structures [55]. G^2PM uses random walks to sample substructures and design the MSM task to automatically balance the contributions between localized and globalized substructures.

7 Conclusion

We introduce G^2PM, a Transformer-based generative pretraining framework for graphs that operates entirely without message passing. Unlike traditional GNNs, G^2PM scales effectively with both data and model size, and consistently improves performance across tasks. Through extensive ablations and experiments, we demonstrate its expressiveness, transferability, and potential as a scalable backbone for graph representation learning. While G^2PM leverages unordered substructure sequences—suitable for masked-token prediction—extending it to ordered sequences may enable next-token prediction, further improving scalability. Additionally, our use of random walks as an online tokenizer opens up future directions in designing learnable and adaptive substructure tokenizers for graphs.

Acknowledgement

The work was partially supported by the NSF under grants IIS-2533550, IIS-2321504, IIS-2340346, IIS-2217239, IIS-2528540, CNS-2426514, and CMMI-2146076, ND-IBM Tech Ethics Lab Program, Notre Dame Strategic Framework Research Grant (2025), and Notre Dame Poverty Research Package (2025). Any expressed opinions, findings, and conclusions or recommendations are those of the authors and do not necessarily reflect the views of the sponsors.

References

[1] Muad Abu-Ata and Feodor F Dragan. Metric tree-like structures in real-world networks: an empirical study. *Networks*, 2016.

[2] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv*, 2023.

[3] Alex Arenas, Alberto Fernandez, Santo Fortunato, and Sergio Gomez. Motif-based communities in complex networks. *Journal of Physics A: Mathematical and Theoretical*, 2008.

[4] Aili Asikainen, Gerardo Iñiguez, Javier Ureña-Carrión, Kimmo Kaski, and Mikko Kivelä. Cumulative effects of triadic closure and homophily in social networks. *Science Advances*, 2020.

[5] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. Beit: Bert pre-training of image transformers. *arXiv*, 2021.

[6] Ginestra Bianconi, Richard K Darst, Jacopo Iacovacci, and Santo Fortunato. Triadic closure as a basic generating mechanism of communities in complex networks. *Physical Review E*, 2014.

[7] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *NeurIPS*, 2020.

[8] Dexiong Chen, Leslie O'Bray, and Karsten Borgwardt. Structure-aware transformer for graph representation learning. In *ICML*, 2022.

[9] Jinsong Chen, Kaiyuan Gao, Gaichao Li, and Kun He. Nagphormer: A tokenized graph transformer for node classification in large graphs. In *ICLR*, 2023.

[10] Xiaohui Chen, Yinkai Wang, Jiaxing He, Yuanqi Du, Soha Hassoun, Xiaolin Xu, and Liping Liu. Graph generative pre-trained transformer. In *ICML*, 2025.

[11] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv*, 2014.

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.

[13] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

[14] Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs. *arXiv*, 2020.

[15] Michael R Garey and David S Johnson. *Computers and intractability*, volume 29. wh freeman New York, 2002.

[16] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *ICML*, 2017.

[17] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *KDD*, 2016.

[18] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NeurIPS*, 2017.

[19] Kaveh Hassani and Amir Hosein Khasahmadi. Contrastive multi-view representation learning on graphs. In *ICML*, 2020.

[20] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022.

[21] Xiaoxin He, Bryan Hooi, Thomas Laurent, Adam Perold, Yann LeCun, and Xavier Bresson. A generalization of vit/mlp-mixer to graphs. In *ICML*, 2023.

[22] Zhenyu Hou, Xiao Liu, Yukuo Cen, Yuxiao Dong, Hongxia Yang, Chunjie Wang, and Jie Tang. Graphmae: Self-supervised masked graph autoencoders. In *KDD*, 2022.

[23] Zhenyu Hou, Yufei He, Yukuo Cen, Xiao Liu, Yuxiao Dong, Evgeny Kharlamov, and Jie Tang. Graphmae2: A decoding-enhanced masked self-supervised graph learner. In *WWW*, 2023.

[24] Sergey Ivanov and Evgeny Burnaev. Anonymous walk embeddings. In *ICML*, 2018.

[25] Yunhui Jang, Dongwoo Kim, and Sungsoo Ahn. Graph generation with k^2-trees. In *ICLR*, 2024.

[26] Yunhui Jang, Seul Lee, and Sungsoo Ahn. A simple and scalable representation for graph generation. In *ICLR*, 2024.

[27] Di Jin, Rui Wang, Meng Ge, Dongxiao He, Xiang Li, Wei Lin, and Weixiong Zhang. Raw-gnn: Random walk aggregation based graph neural network. In *IJCAI*, 2022.

[28] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv*, 2020.

[29] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *ICML*, 2020.

[30] Jinwoo Kim, Dat Nguyen, Seonwoo Min, Sungjun Cho, Moontae Lee, Honglak Lee, and Seunghoon Hong. Pure transformers are powerful graph learners. *NeurIPS*, 2022.

[31] Jinwoo Kim, Olga Zaghen, Ayhan Suleymanzade, Youngmin Ryou, and Seunghoon Hong. Revisiting random walks for learning on graphs. In *ICLR*, 2025.

[32] Thomas N Kipf and Max Welling. Variational graph auto-encoders. In *NeurIPS Workshop*, 2016.

[33] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[34] Devin Kreuzer, Dominique Beaini, Will Hamilton, Vincent Létourneau, and Prudencio Tossou. Rethinking graph transformers with spectral attention. In *NeurIPS*, 2021.

[35] Jintang Li, Ruofan Wu, Wangbin Sun, Liang Chen, Sheng Tian, Liang Zhu, Changhua Meng, Zibin Zheng, and Weiqiang Wang. What's behind the mask: Understanding masked graph modeling for graph autoencoders. In *KDD*, 2023.

[36] Xiaodong Li, Reynold Cheng, Kevin Chen-Chuan Chang, Caihua Shan, Chenhao Ma, and Hongtai Cao. On analyzing graphs with motif-paths. *VLDB*, 2021.

[37] Hao Liu, Jiarui Feng, Lecheng Kong, Ningyue Liang, Dacheng Tao, Yixin Chen, and Muhan Zhang. One for all: Towards training one graph model for all classification tasks. In *ICLR*, 2024.

[38] Tianyi Ma, Yiyue Qian, Chuxu Zhang, and Yanfang Ye. Hypergraph contrastive learning for drug trafficking community detection. In *ICDM*, 2023.

[39] Tianyi Ma, Yiyue Qian, Shinan Zhang, Chuxu Zhang, and Yanfang Ye. Adaptive expansion for hypergraph learning. *arXiv preprint arXiv:2502.15564*, 2025.

[40] Haitao Mao, Zhikai Chen, Wenzhuo Tang, Jianan Zhao, Yao Ma, Tong Zhao, Neil Shah, Michael Galkin, and Jiliang Tang. Graph foundation models. In *ICML*, 2024.

[41] Karolis Martinkus, Pál András Papp, Benedikt Schesch, and Roger Wattenhofer. Agent-based graph neural networks. In *ICLR*, 2023.

[42] Gaspard Michel, Giannis Nikolentzos, Johannes F Lutzeyer, and Michalis Vazirgiannis. Path neural networks: Expressive and accurate graph neural networks. In *ICML*, 2023.

[43] Christopher Morris, Fabrizio Frasca, Nadav Dym, Haggai Maron, Ismail Ilkan Ceylan, Ron Levie, Derek Lim, Michael M. Bronstein, Martin Grohe, and Stefanie Jegelka. Position: Future directions in the theory of graph machine learning. In *ICML*, 2024.

[44] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *KDD*, 2014.

[45] Yiyue Qian, Tianyi Ma, Chuxu Zhang, and Yanfang Ye. Dual-level hypergraph contrastive learning with adaptive temperature enhancement. In *WWW*, 2024.

[46] Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. Gcc: Graph contrastive coding for graph neural network pre-training. In *KDD*, 2020.

[47] Ladislav Rampasek, Mikhail Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. In *NeurIPS*, 2022.

[48] Luana Ruiz, Luiz Chamon, and Alejandro Ribeiro. Graphon neural networks and the transferability of graph neural networks. In *NeurIPS*, 2020.

[49] T Konstantin Rusch, Michael M Bronstein, and Siddhartha Mishra. A survey on oversmoothing in graph neural networks. *arXiv*, 2023.

[50] Alice C Schwarze and Mason A Porter. Motifs for processes on networks. *SIAM Journal on Applied Dynamical Systems*, 2021.

[51] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *arXiv*, 2015.

[52] Qiaoyu Tan, Ninghao Liu, Xiao Huang, Soo-Hyun Choi, Li Li, Rui Chen, and Xia Hu. S2gae: Self-supervised graph autoencoders are generalizable learners with graph masking. In *WSDM*, 2023.

[53] Shantanu Thakoor, Corentin Tallec, Mohammad Gheshlaghi Azar, Mehdi Azabou, Eva L Dyer, Remi Munos, Petar Veličković, and Michal Valko. Large-scale representation learning on graphs via bootstrapping. In *ICLR*, 2022.

[54] Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. *NeurIPS*, 2024.

[55] Jan Tönshoff, Martin Ritzert, Hinrikus Wolf, and Martin Grohe. Walking out of the weisfeiler leman hierarchy: Graph learning beyond message passing. *TMLR*, 2023.

[56] Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. *arXiv*, 2021.

[57] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv*, 2023.

[58] Charalampos E Tsourakakis, Jakub Pachocki, and Michael Mitzenmacher. Scalable motif-aware graph clustering. In *WWW*, 2017.

[59] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[60] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *ICLR*, 2018.

[61] Yuanqing Wang and Kyunghyun Cho. Non-convolutional graph neural networks. In *NeurIPS*, 2024.

[62] Yuanqing Wang and Kyunghyun Cho. Non-convolutional graph neural networks. In *NeurIPS*, 2024.

[63] Zehong Wang, Qi Li, Donghua Yu, Xiaolong Han, Xiao-Zhi Gao, and Shigen Shen. Heterogeneous graph contrastive multi-view learning. In *SDM*, 2023.

[64] Zehong Wang, Donghua Yu, Shigen Shen, Shichao Zhang, Huawen Liu, Shuang Yao, and Maozu Guo. Select your own counterparts: Self-supervised graph contrastive learning with positive sampling. *TNNLS*, 2024.

[65] Zehong Wang, Zheyuan Zhang, Nitesh V Chawla, Chuxu Zhang, and Yanfang Ye. GFT: Graph foundation model with transferable tree vocabulary. In *NeurIPS*, 2024.

[66] Zehong Wang, Zheyuan Zhang, Chuxu Zhang, and Yanfang Ye. Subgraph pooling: Tackling negative transfer on graphs. In *IJCAI*, 2024.

[67] Zehong Wang, Zheyuan Zhang, Tianyi Ma, Nitesh V Chawla, Chuxu Zhang, and Yanfang Ye. Towards graph foundation models: Learning generalities across graphs via task-trees. In *ICML*, 2025.

[68] Zehong Wang, Zheyuan Zhang, Tianyi Ma, Nitesh V Chawla, Chuxu Zhang, and Yanfang Ye. Neural graph pattern machine. In *ICML*, 2025.

[69] Pascal Welke, Maximilian Thiessen, Fabian Jogl, and Thomas Gärtner. Expectation-complete graph representations with homomorphisms. In *ICML*, 2023.

[70] Qitian Wu, Wentao Zhao, Chenxiao Yang, Hengrui Zhang, Fan Nie, Haitian Jiang, Yatao Bian, and Junchi Yan. Simplifying and empowering transformers for large-graph representations. In *NeurIPS*, 2023.

[71] Zhenda Xie, Zheng Zhang, Yue Cao, Yutong Lin, Jianmin Bao, Zhuliang Yao, Qi Dai, and Han Hu. Simmim: A simple framework for masked image modeling. In *CVPR*, 2022.

[72] Dongkuan Xu, Wei Cheng, Dongsheng Luo, Haifeng Chen, and Xiang Zhang. Infogcl: Information-aware graph contrastive learning. *NeurIPS*, 2021.

[73] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *ICLR*, 2019.

[74] Pinar Yanardag and SVN Vishwanathan. Deep graph kernels. In *KDD*, 2015.

[75] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. *NeurIPS*, 2020.

[76] Yuning You, Tianlong Chen, Yang Shen, and Zhangyang Wang. Graph contrastive learning automated. In *ICML*, 2021.

[77] Bohang Zhang, Jingchu Gai, Yiheng Du, Qiwei Ye, Di He, and Liwei Wang. Beyond weisfeiler-lehman: A quantitative framework for gnn expressiveness. In *ICLR*, 2024.

[78] Hengrui Zhang, Qitian Wu, Junchi Yan, David Wipf, and Philip S Yu. From canonical correlation analysis to self-supervised graph neural networks. *NeurIPS*, 2021.

[79] Zheyuan Zhang, Zehong Wang, Shifu Hou, Evan Hall, Landon Bachman, Jasmine White, Vincent Galassi, Nitesh V Chawla, Chuxu Zhang, and Yanfang Ye. Diet-odin: A novel framework for opioid misuse detection with interpretable dietary patterns. In *KDD*, 2024.

[80] Zheyuan Zhang, Zehong Wang, Tianyi Ma, Varun Sameer Taneja, Sofia Nelson, Nhi Ha Lan Le, Keerthiram Murugesan, Mingxuan Ju, Nitesh V Chawla, Chuxu Zhang, et al. Mopi-hfrs: A multi-objective personalized health-aware food recommendation system with llm-enhanced interpretation. In *KDD*, 2025.

[81] Lingxiao Zhao, Wei Jin, Leman Akoglu, and Neil Shah. From stars to subgraphs: Uplifting any GNN with local structure awareness. In *ICLR*, 2022.

[82] Ziwen Zhao, Yuhua Li, Yixiong Zou, Jiliang Tang, and Ruixuan Li. Masked graph autoencoder with non-discrete bandwidths. In *WWW*, 2024.

[83] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Deep graph contrastive representation learning. In *ICML Workshop*, 2020.

[84] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Graph contrastive learning with adaptive augmentation. In *WWW*, 2021.

[85] Nicolas Zucchet and Antonio Orvieto. Recurrent neural networks: vanishing and exploding gradients are not the end of the story. *NeurIPS*, 2024.

A From the Perspective of Substructure Dependencies

To understand how G^2PM work, we take a functional correlations among substructures to inform both model design and training strategies.

Hierarchical Dependencies. Substructures often compose one another hierarchically. For example, a triangle (3-cycle) may serve as a fundamental building block for larger cliques, while a square (4-cycle) can be part of more complex motifs such as diamonds. This compositionality implies that the presence of smaller motifs potentially offer predictive signals for larger structures—and vice versa. From a modeling perspective, this hierarchical property allows the model to infer partially masked substructures based on their visible tokens.

Functional Reinforcement. Certain substructures statistically co-occur due to the generative processes underlying different graph types [48]. For instance, in social and citation networks, a high density of triangles often correlates with the existence of higher-order cliques [6, 4]. Similarly, the frequent appearance of short chains may indicate the potential for longer cycle [36, 50]. We refer to this pattern as *functional reinforcement*, where the presence of one motif increases the likelihood of encountering another. This mutual reinforcement reflects domain structural priors and can be leveraged during pre-training to build a predictive inductive bias across motifs.

Functional Exclusion. Conversely, some substructures exhibit negative correlations, reflecting competition for structural space within the graph. We term this phenomenon *functional exclusion*. For example, a node embedded within multiple triangle-like motifs is likely part of a densely connected community, making it less probable to support hub-like star patterns [3, 58]. Likewise, graphs with tree-like branches typically lack the density required to support large cliques [1]. Understanding such exclusion enables the model to refine its prediction space: it can down-weight structurally incompatible or redundant patterns during reconstruction.

These inter-substructure dependencies serve dual purposes. On one hand, they enrich the predictive landscape by offering complementary structural cues; on the other hand, they enable the model to disambiguate noisy or redundant substructure tokens by reasoning over motif co-occurrence patterns. By explicitly modeling these dependencies, we endow the system with the capacity to generalize from partial observations and regularize against overfitting to spurious or dataset-specific artifacts.

B Implementation Details

B.1 Environments

Most experiments are conducted on Linux servers equipped with four Nvidia A40 GPUs. The models are implemented using PyTorch 2.4.0, PyTorch Geometric 2.6.1, and PyTorch Cluster 1.6.3, with CUDA 12.1 and Python 3.9.

B.2 Training Details

Table 7: Default hyper-parameter settings.

Hyper-parameter	Value	Hyper-parameter	Value
Batch Size	256	Gradient Clip	1
Hidden Dimension	768	Optimizer Beta 1	0.9
Number of Heads	12	Optimizer Beta 2	0.005
Number of Encoder Layers	3	Min LR	1e-07
Number of Decoder Layers	1	Warmup LR	1e-07
EMA Momumtum	0.99	Scheduler	Cosine
EMA Update Every	10	Warmup Epochs	1
Dropout	0.3	Linear Probe LR	0.01
Weight Decay	0.05	Linear Probe Weight Decay	0.001

In our setup, we use the AdamW optimizer with weight decay and set the number of epochs as 100. All experiments are conducted five times with different random seeds. The batch size is set to 256 by default. We present detailed default setup in Table 7.

B.3 Model Configurations

We perform hyperparameter tuning over the following ranges: learning rate $\{1e{-}3, 7e{-}4, 5e{-}4, 3e{-}4, 1e{-}4, 7e{-}5, 5e{-}5, 3e{-}5, 1e{-}5\}$, pattern size $\{4, 8, 16\}$, feature augmentation ratio $p_{\text{feat}} \in [0.0, 0.9]$, and substructure augmentation ratio $p_{\text{sub}} \in [0.0, 0.9]$. The final selected hyper-parameters are reported in Table 8.

Table 8: Dataset-specific hyper-parameter settings.

	Pubmed	**Photo**	**Computers**	**WikiCS**	**Flickr**	**Arxiv**	**Products**
LR	3e-5	1e-5	7e-5	1e-5	5e-5	3e-4	3e-4
Feature Aug. p_{feat}	0.7	0.6	0.2	0.9	0.5	0.0	0.0
Substructure Aug. p_{struct}	0.3	0.8	0.3	0.9	0.1	0.8	0.8
Pattern Size	8	8	8	8	6	8	8
	HIV	**PCBA**	**Sider**	**MUV**	**ClinTox**	**IMDB-B**	**REDDIT-M12K**
LR	3e-5	1e-5	1e-4	5e-5	5e-5	5e-4	3e-4
Feature Aug. p_{feat}	0.1	0.0	0.0	0.0	0.0	0.3	0.0
Substructure Aug. p_{struct}	0.3	0.8	0.7	0.2	0.0	0.3	0.6
Pattern Size	8	16	8	4	4	8	8

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).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

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

   Answer: [Yes]

   Justification: The main contribution is about the scalability, which has been clearly stated in the abstract and introduction.

   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: As discussed in the conclusion, we discuss the limitations and the potential improvements.

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: [NA]

   Justification: No theoretical part in 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: We provide essential reproducibility resources in the appendix. And we definitely will release the code upon acceptance.

   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: See attachment.

   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 provide comprehensive training setup in the experiments and appendix.

   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: See experimental results.

   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 it on 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 strictly follow the NeurIPS guidance.

   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: Our method is a pure machine learning model without using large language models. There are no societal concerns from the technical perspective.

    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: Not available.

    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: We have cited all of essential tools and papers.

    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: [NA]

    Justification: No new assets.

    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: No human subjects.

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: Not relevant to our work.

    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: [Yes]

    Justification: The role of LLMs in this paper is only related to polish.

    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.