Diffusion-Free Graph Generation with Next-Scale Prediction

Samuel Belkadi [*1] **Steve Hong** [*1] **Marian Chen** [*1] **Miruna Cretu** [2] **Charles Harris** [2] **Pietro Liò** [2]

Abstract

Autoregressive models excel in efficiency and plug directly into the transformer ecosystem, delivering robust generalization, predictable scalability, and seamless workflows such as fine-tuning and parallelized training. However, they require an explicit sequence order, which contradicts the unordered nature of graphs. In contrast, diffusion models maintain permutation invariance and enable one-shot generation but require up to thousands of denoising steps and additional features for expressivity, leading to high computational costs. Inspired by recent breakthroughs in image generation, especially the success of visual autoregressive methods, we propose `MAG`, a novel diffusion-free graph generation framework based on next-scale prediction. By leveraging a hierarchy of latent representations, the model progressively generates scales of the entire graph without the need for explicit node ordering. Experiments on both generic and molecular graph datasets demonstrated the potential of this method, achieving inference speedups of up to three orders of magnitude over state-of-the-art methods, while preserving high-quality generation.

1. Introduction

Graphs provide a natural and flexible information representation across a wide range of domains, including social networks, biological and molecular structures, recommender systems, and infrastructural networks. Consequently, the ability to learn a graph distribution from data and generate realistic graphs is pivotal for applications such as network science, drug discovery, and protein design.

Despite significant progress in generative models for language and images, graph generation remains challenging due to its inherent combinatorial nature. Specifically, graphs are naturally **high-dimensional** and **discrete** with **varying sizes**, contrasting with the continuous and fixed-size techniques that cannot be directly applied to them. Furthermore, rich substructures in graphs necessitate an expressive model capable of capturing higher-order motifs and interactions.

Several graph generative models have been proposed to address some of these challenges, undertaking approaches based on autoregressive (AR) models (You et al., 2018), varational autoencoders (VAE) (Jin et al., 2018), generative adversarial networks (De Cao & Kipf, 2018), and diffusion models (Niu et al., 2020; Jo et al., 2022; Vignac et al., 2023). Among these, diffusion and autoregressive models excel in performance and offer complementary strengths—diffusion's permutation invariance and AR's efficiency—that make them especially promising avenues (Kong et al., 2023; Zhao et al., 2024; Yan et al., 2024).

On one hand, diffusion models offer the ability to achieve exchangeable probability in combination with permutation equivariant networks, under certain conditions (Niu et al., 2020; Jo et al., 2022). However, given the high-dimensional nature of graphs and their complex internal dependencies, directly modeling the joint distribution of all nodes and edges presents significant challenges. Recent work by Zhao et al. (2024) demonstrates that, as a cost for permutation invariance, capturing the full joint distribution and solving the transformation difficulty via diffusion requires thousands of sampling and denoising steps as well as additional node-, edge-, and graph-level features, such as eigenvectors, to break symmetries and achieve high generation quality, rendering diffusion a promising but very expensive approach.

On the other hand, in addition to their improved efficiency, studies into the success of AR models have highlighted their **scalability** and **generalizabilty**. The former, as exemplified by scaling laws (Henighan et al., 2020), allows the prediction of large models' performance to better guide resource allocation during training, while the latter, as evidenced by zero-shot and few-shot learning (Sanh et al., 2021), underscores the adaptability of unsupervised-trained models to unseen tasks. Furthermore, the aforementioned discrete and varying-size properties of graphs suggest a promising fit for autoregressive modeling. However, unlike natural language

[*]Equal contribution [1]Department of Engineering, University of Cambridge, UK [2]Department of Computer Science, University of Cambridge, UK. Correspondence to: Samuel Belkadi <sb2764@cam.ac.uk>, Steve Hong <mdh58@cam.ac.uk>.

Proceedings of the 42nd International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025. Copyright 2025 by the author(s).

sentences with an inherent left-to-right ordering, the order of nodes in a graph must be explicitly defined for unidirectional autoregressive learning. Previous AR methods (You et al., 2018; Liao et al., 2020; Shi et al., 2020) attempted to enforce node ordering or approximate the marginalization over permutations such that, once flattened, one can train an autoregressive model to maximize its likelihood via next-token prediction. This gives rise to the three main issues of autoregressive graph generation:

1. **Long-range dependency.** Generating nodes one at a time results in sequential dependencies that make it difficult to capture long-range interactions and global structures. Furthermore, early predictions errors can propagate throughout the generation process, and dependencies may decay when working on larger graphs.
2. **Forced node ordering.** Enforcing a specific node ordering is at odds with the inherent invariance property of graphs, forcing the model to learn over a specific ordering, thereby increasing complexity and reducing generalization.
3. **Inefficiency.** Generating a graph sequence $(x_1, x_2, ..., x_N)$ with a conventional self-attention transformer incurs $\mathcal{O}(n)$ autoregressive steps and $\mathcal{O}(n^3)$ computational cost (see Appendix B).

Drawing inspiration from recent breakthroughs in image generation—notably the Visual Autoregressive Modeling (VAR) framework (Tian et al., 2024; Ren et al., 2025)—our proposed method suggests a new ordering for graphs that is **multi-scale, coarse-to-fine**. We introduce `MAG`, a novel transformer-based graph generation framework based on next-scale prediction instead of traditional *next-node* and *next-edge* approaches, thereby infusing autoregressive properties with graphs' structural constraints. As shown in Figure 1, our method first encodes a graph into multiple scales of latent representations and learns to generate scales autoregressively, starting from a singular token and progressively expanding its resolution. At each step, a transformer predicts the next-resolution token map, conditioned on previous scales and a class label `[C]` for conditional synthesis.

`MAG` poses significant improvements over traditional autoregressive methods on both generic and molecular graph datasets. Notably, it reduces the performance gap with diffusion baselines, surpassing them in wall-clock training and inference time, with strong performance. This provides strong evidence regarding the effectiveness of next-scale prediction for high-quality graph generation. In summary, our contributions are threefold:

1. A novel next-scale prediction framework for hierarchical graph generation that overcomes the limitations of traditional AR approaches on graph properties.
2. A thorough theoretical and empirical analysis of how multi-scale latent representations can effectively capture high-order motifs while remaining significantly more efficient than counterparts.
3. Experiments on diverse graph datasets, demonstrating significant improvements over AR baselines and promising performance when compared to state-of-the-art diffusion methods.

2. Related Work

Diffusion and score-based models Diffusion models achieve permutation-invariance by parameterizing the denoising/score function with a permutation equivariant network, at the cost of up to thousands of sampling steps. EDP-GNN (Niu et al., 2020) is the first work to adapt score-based models to graph generation by representing graphs as matrices with continuous values. GDSS (Jo et al., 2022) generalizes this approach by leveraging SDE-based diffusion and replacing adjacency matrices with node and edge features. However, by using continuous-state diffusion, these approaches ignore graphs' discrete nature, resulting in fully connected graphs. DiGress (Vignac et al., 2023) is the first to apply discrete-state diffusion, achieving significant improvements over continuous methods. Yet, they require additional structural and domain-specific features to break symmetries and achieve high generation quality, further increasing computational costs. In response, SwinGNN (Yan et al., 2024) argues that learning exchangeable probabilities with equivariant architectures is challenging, instead proposing fixed node orderings and non-equivariant score functions to alleviate complexity while maintaining competitive performance.

Autoregressive models Traditional AR models construct graphs by adding nodes and edges sequentially. Although this approach aligns with the discrete nature of graphs, it faces a fundamental challenge that there is no inherent order in graph generation. Various strategies have been explored to address this by enforcing some node ordering and approximating the marginalization over permutations. Li et al. (2018) suggest using random or deterministic empirical orderings. GraphRNN (You et al., 2018) aligns permutations with Breadth-First Search (BFS) ordering through a many-to-one mapping. GRANs (Liao et al., 2020) propose an approach to marginalize over a family of canonical node orderings, such as node degree descending, BFS trees rooted at the highest degree node, and k-core orderings. Chen et al. (2021) eliminate ad-hoc orderings altogether by modeling the conditional probability of orderings with another AR model to estimate the marginalized probabilities for both the generative and ordering-selection models. Despite these efforts, all such methods remain fundamentally limited by the need to impose artificial node orderings.

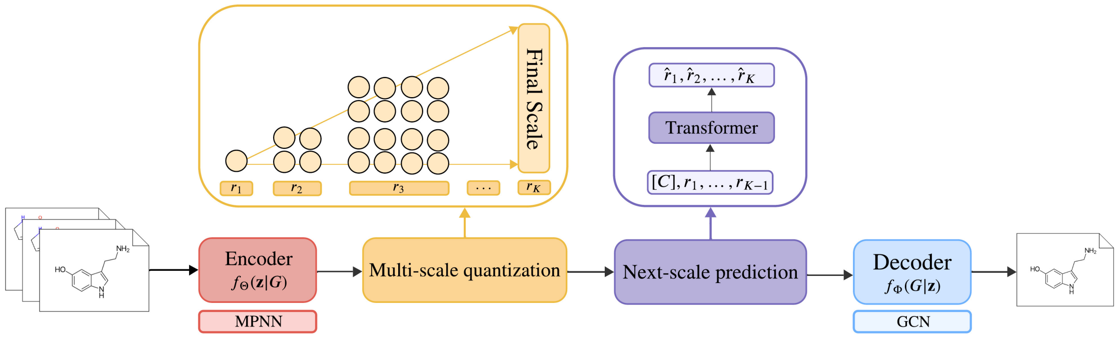

Figure 1. **Overview of the Mutli-Scale Graph Generation Framework.** The input graph is first encoded through an MPNN, which produces a latent representation that is quantized at multiple scales. A Transformer then predicts the scales autoregressively via a coarse-to-fine next-scale process, and a GCN-based decoder reconstructs the graph from the final discrete latent representation r_K.

Hybrid and hierarchical models Several recent works seek to blend the strengths of diffusion and autoregressive (AR) paradigms. PARD (Zhao et al., 2024) first partitions the graph into blocks, applies diffusion within each block and AR between them, and enforces permutation invariance via *partial ordering*. While effective, its block-wise diffusion introduces extra implementation complexity and still incurs substantial sampling costs (Hou et al., 2024). GraphArm (Kong et al., 2023) instead embeds diffusion steps into an AR framework—at each forward pass, a single node and its edges stochastically decay—making the model both permutation- and order-sensitive. Beyond these hybrids, pure coarse-to-fine strategies have also been explored: Bergmeister et al. (Bergmeister et al., 2024) grow a full graph from one seed node using localized denoising diffusion and spectral cues, matching diffusion-only runtimes but retaining their high sampling overhead. In the molecular domain, JT-VAE (Jin et al., 2018) and HierVAE (Jin et al., 2020) build molecules by assembling chemically valid substructures, and HiGGs (Davies et al., 2023) hierarchically samples entire graphs across multiple resolutions. However, all of these rely on either block-level diffusion, node-level decay processes, or strong domain priors (e.g., motif libraries or spectral conditioning). In contrast, our framework is fully domain-agnostic and permutation-equivariant, leveraging multi-scale quantized latent maps with a generic transformer—without any hand-crafted substructure priors.

3. Method

Consider a graph defined by the quadruple $G = (\mathcal{V}, E, \mathbf{X}, \mathbf{B})$, where $\mathcal{V}$ denotes the set of N vertices, $E \subseteq \mathcal{V} \times \mathcal{V}$ represents the set of edges, $\mathbf{X} \in \mathbb{R}^{N \times D}$ is a matrix of D-dimensional node features, and $\mathbf{B} \in \mathbb{R}^{N \times N \times F}$ comprises edge attributes with dimensions F. Given a collection of M observed graphs, $\mathcal{G} = \{G_i\}_{i=1}^{M}$, the task of graph generation is to model the underlying distribution $p(\mathcal{G})$ in order to generate new graphs, $G_{\text{sample}} \sim p(\mathcal{G})$.

3.1. Preliminary: Autoregression by Scale

AR models represent the joint distribution over N random variables by employing the chain rule of probability. In particular, they decompose the generation process into sequential steps, where each step determines the subsequent action based on the current subgraph. Traditionally, previous works have taken a *next-token prediction* approach, generating the graph one node and its edges at a time. Specifically, at step i, we introduce node G_i^π along with any new edges connecting it to nodes in $\{G_1^\pi, G_2^\pi, \dots, G_{i-1}^\pi\}$. The general formulation of AR models for graphs is given by:

$$p(\mathcal{G}^\pi) = \prod_{i=1}^{N} p\big(G_i^\pi \mid G_1^\pi, G_2^\pi, \dots, G_{i-1}^\pi\big).$$

Since AR models proceed in a unidirectional sequential manner, applying them requires a predetermined ordering π of nodes in the graph.

Alternatively, rather than employing *next-token prediction*, we propose utilizing *next-scale prediction*, wherein the autoregressive unit is a latent representation of the entire graph rather than a single token. To implement this, we first obtain latent representations at multiple scales and train a transformer to generate these latent maps. Similarly to Tian et al. (2024), this leads to a two-stage architecture: **(1)** a VQ-VAE tokenizer that quantizes the input graphs into latent maps at multiple scales (refer to Section 3.2), and **(2)** a transformer that predicts next-scale latent maps (refer to Section 3.3).

3.2. Permutation Equivariant Multi-Scale Tokenizer

We propose a permutation-equivariant quantized autoencoder to encode a graph G to K multi-scale discrete latent maps $R = \{r_1, r_2, ..., r_K\}$, where each scale r_k depends only on its prefix $\{r_1, r_2, ..., r_{k-1}\}$. For quantization, a shared codebook Z is used across all scales, ensuring that each token in r_k belongs to the same vocabulary. An overview of the tokenizer is given in Figure 2.

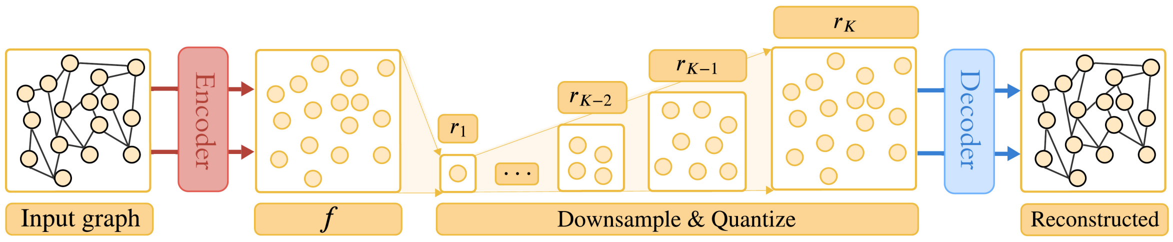

Figure 2. **Permutation Equivariant Multi-Scale VQ-VAE.** The encoder converts the input graph into a continuous latent representation f, which is downsampled and quantized into discrete token maps $\{r_1, \ldots, r_K\}$. The final token map r_K is then passed to the GCN-based decoder to reconstruct the original graph, preserving permutation equivariance throughout.

Encoding Given an input graph $G \in \mathbb{R}^{N \times D} \times \mathbb{R}^{N \times N \times F}$, an autoencoder $\mathcal{E}(\cdot) : \mathbb{R}^{N \times D} \times \mathbb{R}^{N \times N \times F} \to \mathbb{R}^{N \times C}$ is used to convert G into a continuous latent representation f:

$$f = \mathcal{E}(G), \quad f \in \mathbb{R}^{N \times C},$$

where C is the latent's feature dimension defined empirically. In order to encode the graph's node and edge features into a unified latent space $\mathbb{R}^{N \times C}$, we leverage a series of standard MPNNs, such that edge features are propagated within node features before compressing node dimensions to C and dropping remaining edges.

Concretely, each MPNN layer performs message passing by first computing learned linear transformations of neighbor node embeddings and their connecting edge attributes, aggregating these via a sum operation, and then applying a ReLU nonlinearity. We stack L_e such layers—with residual connections and layer normalization—to ensure stable training and encode higher-order structural patterns throughout the graph within nodes.

f is then downsampled to create K different coarse-to-fine latent scales $F = \{f_1, f_2, ..., f_K\}$, with $f_K = f$. Intermediate scales are learned by the model to autoregressively generate the graph's latent representation, scale-wise.

Quantizing A quantizer is used to convert each scale's latent representation into discrete tokens. The quantizer $\mathcal{Q}(\cdot) : \mathbb{R}^{N \times C} \to \mathbb{R}^{N \times C_Z}$ includes a codebook $Z \in \mathbb{R}^{V \times C_Z}$ containing V learnable embeddings, for which each feature's dimension is maintained by convention, i.e., $C_Z = C$. During quantization, $q = \mathcal{Q}(f)$ is obtained by replacing each feature vector $f^{(k,i)}$ from the multi-scale latent features F by its nearest code $q^{(k,i)}$ in Euclidean distance:

$$q^{(k,i)} = (\text{argmin}_{v \in V} ||\text{Select}(Z, v) - f^{(k,i)}||_2) \in V,$$

where Select(Z, v) denotes selecting the v^{th} vector in codebook Z, and $i \leq N$. Specifically, $\mathcal{Q}(\cdot)$ is trained by fetching all $q^{(k,i)}$ and minimizing the distance between q and f.

Decoding Finally, a decoder is used to reconstruct the original graph given its final discrete latent representation, r_K. The decoder $\mathcal{D}(\cdot) : \mathbb{R}^{N \times C_Z} \to \mathbb{R}^{N \times D} \times \mathbb{R}^{N \times N \times F}$ recovers latent representations for edges between all pairs of nodes, and maps node and edge features back to their original dimensions $\mathbb{R}^{N \times D}$ and $\mathbb{R}^{N \times N \times F}$, respectively.

We implement $\mathcal{D}(\cdot)$ with L_d standard graph convolutional layers, interleaved with ReLU activations and layer normalization to stabilize training and capture higher-order structures. After the final GCN layer, edge logits are recovered by passing all concatenated pairs of node embeddings through an MLP layer, and node features are mapped back to input dimensions, $\mathbb{R}^D$, via a linear projection.

Because this architecture preserves node ordering and feature cardinality throughout encoding, quantization, and decoding, the proposed multi-scale autoencoder remains **permutation equivariant**. Note that any off-the-shelf autoencoder can be used, since downsampled scales are obtained by interpolating, and only the embedding f (and thus final scale r_K) is used for decoding. This effectively makes the scales independent of the rest of the architecture.

3.3. Autoregressive Graph Generation Through Next-Scale Prediction

We define the autoregressive modeling on graphs by shifting from *next-node* and *next-edge* predictions to *next-scale* predictions. Here, the autoregressive unit is *an entire token map*, rather than *a single token*. Given the quantized scales $\{r_1, \ldots, r_K\}$, the autoregressive likelihood is expressed as:

$$p(r_1, r_2, \ldots, r_K) = \prod_{k=1}^{K} p(r_k \mid r_1, r_2, \ldots, r_{k-1}),$$

where each autoregressive unit $r_k \in [V]^{n_k}$ is the token map at scale k containing n_k tokens, and the sequence $(r_1, r_2, \ldots, r_{k-1})$ serves as the prefix for r_k. In the k-th autoregressive step, we predict the k-th scale, such that all n_k tokens in r_k are generated in parallel, conditioned on r_k's prefix. Note that during training, a block-wise causal attention mask is used to ensure that each r_k can only attend to its prefix $r_{<k}$. During inference, kv-caching is used and

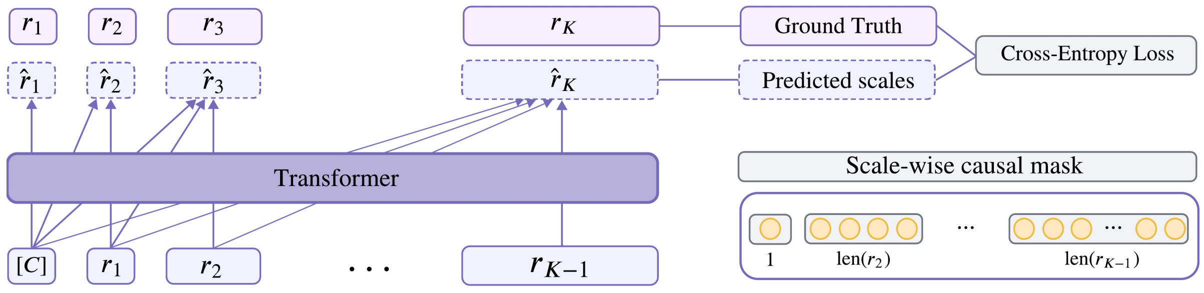

Figure 3. **Autoregressive Next-scale Prediction with Scale-Wise Causal Mask.** At each step k, the transformer attends to all previously generated scales $\{r_1, \ldots, r_{k-1}\}$ and predicts the next-scale token map r_k all at once. A scale-wise causal mask ensures that each scale only depends on its prefix. Cross-entropy loss is used for training the transformer.

no mask is needed. An overview of the generative model's architecture is displayed in Figure 3.

Level embedding A level embedding is learned to encode the scale at which a token operates within the input structure. This is implemented similarly to GPT's segment embedding (Devlin et al., 2019), such that learned signals enhance the model's capacity to capture multi-scale contextual dependencies. However, it is important to note that no additional positional encoding is used within scales, in order to leverage Attention's permutation-invariance property.

Scale sampling During training and inference, we select scale sizes from a predefined set of lengths. Specifically, we fix the first scale to 1 and the final scale to N, where N is the target graph size. Intermediate scales are chosen from the predefined set (e.g., $R = \{1, 2, 4, 6, 9\}$), such that $\forall r_k \in R, r_k \leq N$; and the final scale $r_K = N$ (e.g., truncating to $\{1, 2, 4, 5\}$ for a graph of 5 nodes). Typically, we set the number of scales to be logarithmic in the number of nodes. Finally, to handle varying graph sizes and enable parallel generation, padding is used with appropriate masking.

Implementation details We adopt a standard decoder-only Transformer architecture with adaptive layer normalization (AdaLN), and input tokens are embedded via a linear layer. Unlike Graphormer (Ying et al., 2021) or DiGress (Vignac et al., 2023), we do not use any additional positional embeddings. For class-conditional synthesis, a class embedding serves both as the start token `[C]` and the conditioning input for AdaLN. Then, at each autoregressive step, the next scale is predicted such that all tokens in that scale are generated in parallel through a series of multi-head self-attention blocks. Each output embedding is passed through a classifier head to predict a categorical distribution over the quantization codebook, from which the most probable codeword is selected using a `softmax` operation. Dropout and layer normalization are employed throughout the architecture to aid convergence. Additionally, query and key vectors are normalized to unit vectors prior to computing attention scores to improve training stability. To support reproducibility, the hyperparameters used for training and inference are listed in Appendix A.

Sampling at inference At inference, we apply top-k and top-p (nucleus) sampling to introduce diversity and control over the generation process. These methods are applied to the predicted categorical distributions at each decoding step.

Discussion As a result, multi-scale prediction offers substantial properties that allow `MAG` to address the three previously mentioned issues as follows:

1. Given the limited number of intermediate scales in the generations, long-range dependency is reduced from $\mathcal{O}(n)$ to $\mathcal{O}(k)$, where $k \approx \log(n)$ is the number of scales. Furthermore, sequential dependency is strengthened by generating a scale of the whole graph at each autoregressive step.

2. No explicit node ordering is needed as **(i)** the tokenizer is permutation equivariant, and **(ii)** tokens in each map, r_k, are fully correlated and generated in parallel.

3. By generating an entire token map in parallel at each scale, the generation's complexity is reduced from $\mathcal{O}(n^3)$ to $\mathcal{O}(n^2)$. The proof is detailed in Appendix B.

4. Experiments

We empirically evaluate our model on experiments covering both generic and molecular graph generation. These two domains are chosen to benchmark the model's versatility: generic graphs allow the assessment of structural fidelity and scalability, while molecular graphs pose domain-specific challenges like chemical validity and functional constraints. This dual setting probes `MAG`'s ability to generate diverse graphs under different structural and semantic constraints.

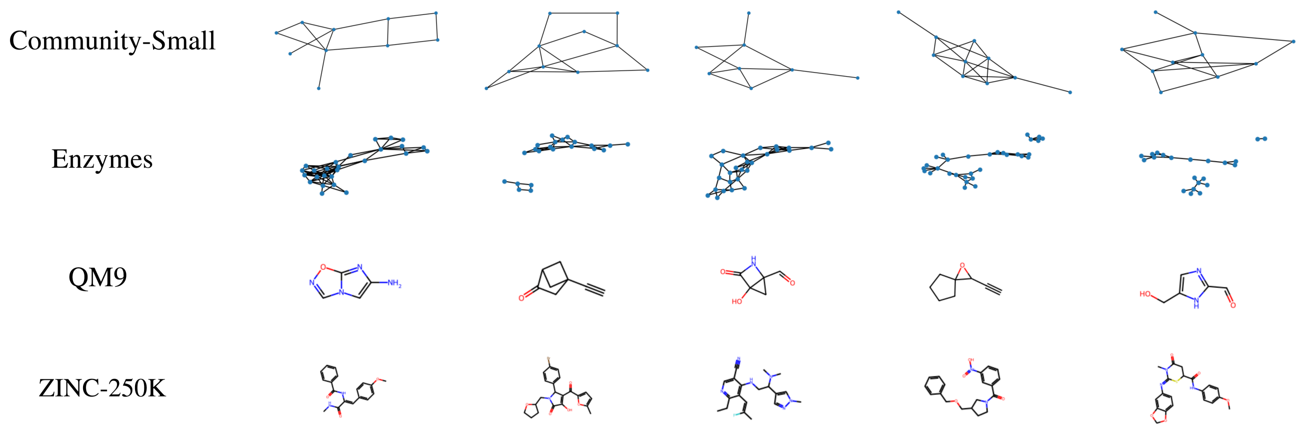

Figure 4. Samples of generated generic and molecular graphs by `MAG`.

Generation quality is measured using a combination of distributional and domain-specific metrics. Moreover, we evaluate generation wall-clock time against state-of-the-art diffusion, hybrid, and VAE models to compare their inference efficiency. Both training and evaluation procedures were executed in five independent trials, with error bars denoting the robustness of the experiments.

Baselines We compare `MAG` to state-of-the-art models trained for both generic and molecule generation. Specifically, baselines include diffusion models, such as EDP-GNN (Niu et al., 2020), GDSS (Jo et al., 2022), and DiGress (Vignac et al., 2023); hybrid models, such as GraphAF (Shi et al., 2020), GraphDF (Luo et al., 2021) and GraphRNN (You et al., 2018) (generic graphs only); and VAE models, such as GraphVAE (Simonovsky & Komodakis, 2018) and GramVAE (Kusner et al., 2017) (molecular graphs only).

4.1. Generic Graph Generation

Experimental Setup We evaluate `MAG` at generic graph generation on the Community-small (Jo et al., 2022) and Enzymes (Schomburg et al., 2004) datasets. Community-small is a synthetic dataset containing 100 random graphs consisting of two equal-sized communities, with between 12 and 20 nodes per graph. Here, the node and edge representations are binary (i.e., absence or presence). The Enzymes dataset contains 587 enzyme graphs with 10 to 125 nodes per sample. Each of the graphs represents a protein as a set of nodes—each node carrying a 3-dimensional one-hot feature for α-helix, β-strand or coil—and binary edges indicating sequence adjacency or the three nearest spatial neighbors. There are no edge attributes, and each graph is annotated with one of six top-level EC enzyme class labels.

These datasets provide a wide range of graph properties, from small structured communities in Community-small to larger, more complex protein graphs in Enzymes. We aim to investigate the ability of multi-scale graph generation to generate graphs of different sizes and complexities, providing insights into its scalability and generalization abilities.

To ensure fair comparison, we follow the same setup as (Kong et al., 2023; Yan et al., 2024; Jo et al., 2022), randomly splitting the dataset into 80% for training and 20% for testing. Performance is evaluated by comparing the structure of the generated graphs with those from the dataset according to the maximum mean discrepancy (MMD) of statistics including node degrees, clustering coefficients, and orbit counts. Moreover, the inference speed of `MAG` is compared to the baselines at generating graphs across varying sizes. Samples of generated graphs are given in Figure 4.

Results `MAG` achieves strong performance with inference times orders of magnitude faster (Table 1). On the smaller Community-small dataset, it consistently outperforms both hybrid models and VAEs. Notably, its performance is the closest to state-of-the-art GDSS and DiGress models, while being 204$\times$ and 15,347$\times$ faster, respectively.

On the more complex Enzymes dataset, `MAG` exhibits limitations in qualitative performance. Similarly to hybrid models, it struggles with generating larger enzyme graphs, suggesting possible scalability challenges. This may be due to the increased number of scales required for larger graphs, resulting in higher sensitivity to predefined scale sizes. Despite this limitation, `MAG`'s computational efficiency scales robustly with graph size, indicating strong potential should its performance be further improved.

In terms of wall-clock inference time, `MAG` delivers significant improvements across all datasets, surpassing all baselines by several orders of magnitude. Remarkably, it achieves inference speeds up to **thousands of times faster** than state-of-the-art models.

Table 1. Results on **Community-small** and **Enzymes** benchmark datasets. Wall-clock inference times are reported in seconds for 50 and 100 samples, respectively, on an NVIDIA GeForce RTX 4060 GPU.

Models	**Community-S** ($\lvert\mathcal{V}\rvert \in [12, 20]$, 100 obs.)				**Enzymes** ($\lvert\mathcal{V}\rvert \in [10, 125]$, 587 obs.)			
	Deg. ↓	Clus. ↓	Orbit. ↓	Time (s) ↓	Deg. ↓	Clus. ↓	Orbit. ↓	Time (s) ↓
Diffusion Models								
EDP-GNN	0.053	0.144	0.026	$1.94e^4$	0.023	0.268	0.082	$2.61e^4$
GDSS	**0.045**	0.086	0.007	$2.87e^3$	0.026	0.061	0.009	$1.74e^3$
DiGress	0.047	**0.041**	0.026	38.1	0.004	0.083	0.002	58.8
Hybrid Models								
GraphAF	0.180	0.200	0.020	$1.29e^2$	1.669	1.283	0.266	46.1
GraphDF	0.060	0.120	0.030	68.8	1.503	1.061	0.202	68.1
GraphRNN	0.080	0.120	0.040	$1.10e^2$	0.017	0.062	0.046	$1.25e^2$
Varational Autoencoders								
GraphVAE	0.350	0.980	0.540	1.34	1.369	0.629	0.191	2.00
Ours	0.077	0.102	0.022	**0.187**	1.753	1.148	0.223	**1.88**
± std.	±0.007	±0.009	±0.003	±0.015	±0.150	±0.100	±0.034	±0.200

4.2. Molecule Generation

Experimental Setup `MAG` is further extended to generate molecular graphs with node and edge features, and evaluated on two popular molecular datasets, namely QM9 (Ramakrishnan et al., 2014) and ZINC-250K (Gómez-Bombarelli et al., 2018). The former contains 133,885 small organic molecules of up to nine heavy atoms with types C, O, N and F. The latter is a subset of the ZINC database (Irwin et al., 2012) and consists of 250,000 molecules with up to 38 heavy atoms of nine types.

In contrast to generic graph generation, molecules present a unique challenge for multi-scale graph generation due to their strict structural constraints and varying complexity. Evaluating `MAG` on both molecular datasets assesses its ability to generate realistic molecules ranging from small organic compounds to larger chemical structures, while adhering to chemical constraints like valency, introducing additional challenges for the model to maintain validity.

The quality of generated molecules is measured in terms of validity, uniqueness, novelty, and Fréchet ChemNet Distance (FCD). Details on these metrics are delineated in Appendix C. Training and evaluation were each performed in five independent replicates, with the resulting performance robustness illustrated by error bars. In addition, the wall-clock inference time to generate 1,000 molecules from both datasets is recorded and compared against baselines. Samples of generated molecules are displayed in Figure 4.

Results Performance on molecular datasets are given in Table 2. `MAG` consistently outperforms VAE baselines across most metrics. Moreover, `MAG` achieves performance comparable to that of recent hybrid models. Compared to state-of-the-art diffusion models, `MAG` achieves the highest uniqueness and matches most of them in novelty. Regarding validity, although it trails slightly behind GDSS, it proves to outperform EDP-GNN. Of all baselines, only DiGress exhibits a notable performance advantage over `MAG`; however, DiGress explicitly incorporates molecular-specific properties, while `MAG` approaches the task in a raw, domain-agnostic manner, which may account for this performance gap. For the larger and more complex ZINC-250k dataset, `MAG` outperforms VAE baselines across all metrics and delivers results comparable to those of hybrid models, while remaining slightly behind diffusion-based methods. These findings are consistent with observations on smaller molecules; however, `MAG` demonstrates greater consistency across different molecule sizes compared to diffusion models. Additionally, we observe reduced overhead when scaling to larger molecules relative to the results on the Enzymes dataset. Regarding computational efficiency, `MAG` further delivers substantial improvements, achieving inference speeds up to **hundreds of times faster** than the state of the art.

5. Future Work and Limitations

This work primarily focused on introducing a novel paradigm for autoregressive graph generation, adopting a

Table 2. Results on **QM9** and **ZINC250k** molecule datasets. Wall-clock inference times are reported in seconds for 1,000 samples, respectively, on an NVIDIA GeForce RTX 4060 GPU (Val: Validity; Uni: Uniqueness; Nov: Novelty; FCD: Fréchet ChemNet Distance). Entries marked "–" denote results that were not reported or for which reproduction was not possible.

Models	**QM9** ($\|\mathcal{V}\| \in [1, 9]$, 134K mols.)					**ZINC250k** ($\|\mathcal{V}\| \in [6, 38]$, 250K mols.)				
	Val. ↑	Uni. ↑	Nov. ↑	FCD ↓	Time (s) ↓	Val. ↑	Uni. ↑	Nov. ↑	FCD ↓	Time (s) ↓
Diffusion Models										
EDP-GNN	47.52	99.25	86.58	2.68	$1.7e^3$	82.97	**99.79**	100	16.74	$1.6e^3$
GDSS	95.72	98.46	86.27	2.9	43.4	**97.01**	99.64	100	**14.66**	$3.4e^2$
DiGress	**99.0**	96.66	33.4	**0.36**	34.1	91.02	81.23	100	23.06	$2.0e^2$
Hybrid Models										
GraphAF	74.43	88.64	86.59	5.27	$1.1e^3$	68.47	98.64	100	16.02	$9.6e^2$
GraphDF	93.88	98.58	98.54	10.93	$1.9e^4$	90.61	99.63	100	33.55	$9.3e^3$
Varational Autoencoders										
GramVAE	20.00	19.70	**100**	–	–	30.10	27.30	100	–	–
GraphVAE	45.80	30.50	66.10	–	–	44.00	–	–	–	–
Ours	89.43	**99.31**	83.14	5.77	**3.84**	84.22	87.88	100	19.70	**19.7**
± **std.**	±0.87	±0.38	±1.72	±0.49	±1.40	±1.32	±0.53	±0.00	±1.06	±0.58

next-scale prediction approach instead of traditional *next-node* and *next-edge* methods. As a result, a promising avenue for future research is to explore how autoregressive properties from the large language model (LLM) literature translate to graph generation. Specifically, efforts could be made to investigate the generalization capabilities of autoregressive models in zero-shot and few-shot downstream tasks, which have been shown to be limitations of state-of-the-art diffusion-based graph models (Vignac et al., 2023). Furthermore, future work could explore fine-tuning strategies as a solution to tasks with little data and resources.

Although MAG demonstrated substantial speed improvements, experiments on larger graphs revealed qualitative limitations—likely stemming from our use of a fixed set of scales. Incorporating inductive biases or domain-specific signals into scale selection could enhance both performance and scalability on more complex structures. Moreover, autoregressive models are known for their inherent scalability, suggesting that MAG's performance on larger graphs could further improve through optimized scale selection. A promising direction is to train a unified decoder that maps any intermediate latent scale back to a graph, much like Jiao et al.'s multi-resolution VAE. This would not only allow interpretation and visualization of the coarse-to-fine hierarchy and how motifs emerge, but also drive a data-driven choice for scale counts and resolutions, alleviating our current dependence on manually tuned scale hyperparameters.

Finally, this work opens a new direction for hybrid autoregressive-diffusion methods. While current hybrid approaches primarily rely on block-wise diffusion with sequential learning between blocks, the proposed framework enables reconsidering these methods by applying scale-wise diffusion, i.e., diffusion between scales. This shift could eliminate the need for a quantized autoencoder, as diffusion offers strong capabilities in continuous space, thereby removing the quantization bottleneck. However, this would reintroduce computational overheads which should be studied as a trade-off against existing diffusion-based methods.

6. Conclusion

This work introduced a novel diffusion-free multi-scale autoregressive model for graph generation which **(1)** theoretically addresses limitations of traditional autoregressive models due to inherent graph properties and **(2)** reduces the gap between diffusion-free AR models and diffusion-based methods, with up to three orders of magnitude lower computational costs. Experiments over both generic and molecular graph generations demonstrated the ability of MAG to generate high-quality samples that achieve promising performance and can adjust to domain-specific constraints such as chemical validity. We hope that our findings can aid the design of new approaches, in which the multi-scale paradigm may be applied to other graph generative methods.

Impact Statement

Fast graph generation can accelerate scientific discovery by enabling rapid simulation of complex networks in fields such as epidemiology, ecology, and social science, lowering barriers for smaller research teams. It supports more resilient infrastructure and public-health planning by allowing practitioners to model extreme scenarios at unprecedented speed. By democratizing access to advanced network modeling, these tools empower communities worldwide to tackle interconnected societal challenges.

Acknowledgment

We thank Martinkus Karolis for his detailed feedback on architectural design, his comments on final drafts, and the provision of key literature references that strengthened our work. We also extend our gratitude to Petar Veličković for his early feedback on the foundations of this work and his continued support throughout.

References

Bergmeister, A., Martinkus, K., Perraudin, N., and Wattenhofer, R. Efficient and scalable graph generation through iterative local expansion. In *Proceedings of the 12th International Conference on Learning Representations*, 2024.

Chen, X., Han, X., Hu, J., Ruiz, F., and Liu, L. Order matters: Probabilistic modeling of node sequence for graph generation. In *Proceedings of the 38th International Conference on Machine Learning*, pp. 1630–1639. PMLR, 2021.

Davies, A. O., Ajmeri, N. S., et al. Size matters: Large graph generation with higgs. *arXiv preprint arXiv:2306.11412*, 2023.

De Cao, N. and Kipf, T. Molgan: An implicit generative model for small molecular graphs. *arXiv preprint arXiv:1805.11973*, 2018.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, June 2019.

Gómez-Bombarelli, R., Wei, J. N., Duvenaud, D., Hernández-Lobato, J. M., Sánchez-Lengeling, B., Sheberla, D., Aguilera-Iparraguirre, J., Hirzel, T. D., Adams, R. P., and Aspuru-Guzik, A. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*, 4(2):268–276, 2018. doi: 10.1021/acscentsci.7b00572. PMID: 29532027.

Henighan, T., Kaplan, J., Katz, M., Chen, M., Hesse, C., Jackson, J., Jun, H., Brown, T. B., Dhariwal, P., Gray, S., et al. Scaling laws for autoregressive generative modeling. *arXiv preprint arXiv:2010.14701*, 2020.

Hou, X., Zhu, T., Ren, M., Bu, D., Gao, X., Zhang, C., and Sun, S. Improving molecular graph generation with flow matching and optimal transport, 2024.

Irwin, J. J., Sterling, T., Mysinger, M. M., Bolstad, E. S., and Coleman, R. G. Zinc: A free tool to discover chemistry for biology. *Journal of Chemical Information and Modeling*, 52(7):1757–1768, July 2012. doi: 10.1021/ci3001277.

Jiao, S., Zhang, G., Qian, Y., Huang, J., Zhao, Y., Shi, H., Ma, L., Wei, Y., and Jie, Z. Flexvar: Flexible visual autoregressive modeling without residual prediction. *arXiv preprint arXiv:2502.20313*, 2025.

Jin, W., Barzilay, R., and Jaakkola, T. Junction tree variational autoencoder for molecular graph generation. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pp. 2323–2332. PMLR, July 2018.

Jin, W., Barzilay, R., and Jaakkola, T. Hierarchical generation of molecular graphs using structural motifs. In *Proceedings of the International Conference on Machine Learning*, 2020.

Jo, J., Lee, S., and Hwang, S. J. Score-based generative modeling of graphs via the system of stochastic differential equations. *arXiv preprint arXiv:2202.02514*, 2022.

Kong, L., Cui, J., Sun, H., Zhuang, Y., Prakash, B. A., and Zhang, C. Autoregressive diffusion model for graph generation. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, pp. 17391–17408. PMLR, 2023.

Kusner, M. J., Paige, B., and Hernández-Lobato, J. M. Grammar variational autoencoder. In *International conference on machine learning*, pp. 1945–1954. PMLR, 2017.

Li, Y., Vinyals, O., Dyer, C., Pascanu, R., and Battaglia, P. Learning deep generative models of graphs. *arXiv preprint arXiv:1803.03324*, 2018.

Liao, R., Li, Y., Song, Y., Wang, S., Hamilton, W., Duvenaud, D. K., Urtasun, R., and Zemel, R. Efficient graph generation with graph recurrent attention networks. In *Advances in Neural Information Processing Systems*, volume 32, 2020.

Luo, Y., Yan, K., and Ji, S. Graphdf: A discrete flow model for molecular graph generation, 2021.

Niu, C., Song, Y., Song, J., Zhao, S., Grover, A., and Ermon, S. Permutation invariant graph generation via score-based generative modeling. In *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2020.

Ramakrishnan, R., Dral, P. O., Rupp, M., and von Lilienfeld, O. A. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific Data*, 1:140022, August 2014. doi: 10.1038/sdata.2014.22.

Ren, S., Yu, Q., He, J., Shen, X., Yuille, A., and Chen, L.-C. Beyond next-token: Next-x prediction for autoregressive visual generation. *arXiv preprint arXiv:2502.20388*, 2025.

Sanh, V., Webson, A., Raffel, C., Bach, S. H., Sutawika, L., Alyafeai, Z., Chaffin, A., Stiegler, A., Scao, T. L., Raja, A., et al. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*, 2021.

Schomburg, I., Chang, A., Hofmann, O., Ebeling, C., Ehrentreich, F., and Schomburg, D. Brenda, the enzyme database: Updates and major new developments. *Nucleic Acids Research*, 32(Database issue):D431–D433, 2004. doi: 10.1038/nar/gkh081.

Shi, C., Xu, M., Zhu, Z., Zhang, W., Zhang, M., and Tang, J. Graphaf: A flow-based autoregressive model for molecular graph generation, 2020.

Simonovsky, M. and Komodakis, N. Graphvae: Towards generation of small graphs using variational autoencoders. In *Artificial Neural Networks and Machine Learning–ICANN 2018: 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part I 27*, pp. 412–422. Springer, 2018.

Tian, K., Jiang, Y., Yuan, Z., Peng, B., and Wang, L. Visual autoregressive modeling: Scalable image generation via next-scale prediction. In *Proceedings of the 38th Annual Conference on Neural Information Processing Systems*, 2024.

Vignac, C., Krawczuk, I., Siraudin, A., Wang, B., Cevher, V., and Frossard, P. Digress: Discrete denoising diffusion for graph generation. In *Proceedings of the Eleventh International Conference on Learning Representations*, 2023.

Yan, Q., Liang, Z., Song, Y., Liao, R., and Wang, L. Swingnn: Rethinking permutation invariance in diffusion models for graph generation. *Transactions on Machine Learning Research*, 2024.

Ying, C., Cai, T., Luo, S., Zheng, S., Ke, G., He, D., Shen, Y., and Liu, T.-Y. Do transformers really perform badly for graph representation? In *Advances in Neural Information Processing Systems*, 2021.

You, J., Ying, R., Ren, X., Hamilton, W., and Leskovec, J. Graphrnn: Generating realistic graphs with deep autoregressive models. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 5708–5717. PMLR, 2018.

Zhao, L., Ding, X., and Akoglu, L. Pard: Permutation-invariant autoregressive diffusion for graph generation. In *Proceedings of the 38th Annual Conference on Neural Information Processing Systems*, 2024.

A. Details for Hyperparameters

Table 3. Hyperparameters used for `MAG` during training and inference.

Hyperparameter	**Value**
Multi-Scale VQ-VAE Tokenizer	
Encoder MPNN layers (L_e)	4
Decoder GCN layers (L_d)	4
Hidden dimension	32
Latent dimension (C)	16
Codebook size (V)	1024
Commitment cost	0.25
Gamma	0.1
Multi-scale Transformer	
Transformer blocks	8
Hidden size	256
Attention heads	8
Level embedding dim	256
Layer dropout	0.1
Conditional dropout	0.1
Token dropout	0.05
Optimization & Sampling	
Optimizer	Adam
Learning rate	3×10^{-5}
Weight decay	1×10^{-2}
Betas	(0.9, 0.99)
Batch size	12
Training epochs	100
Top-k sampling (k)	50
Top-p sampling (p)	0.95

B. Complexity Proof

This proof entails the improvement in generation complexity from node-by-node to scale-wise AR models.

Lemma B.1 (AR Generation for Nodes)**.** *For a standard self-attention transformer, the time complexity of autoregressive (AR) graph generation is $O(N^3)$, where N is the total number of nodes.*

Proof. In AR generation, nodes are generated sequentially. At step i ($1 \leq i \leq N$), the model computes attention over all $i-1$ previously generated nodes. The complexity for step i is $O(i^2)$. Summing over all steps:

$$\sum_{i=1}^{N} i^2 = \frac{N(N+1)(2N+1)}{6} \sim O(N^3).$$

□

Lemma B.2 (Scale-wise Generation for Nodes)**.** *For a standard self-attention transformer with constant $a > 1$, the time complexity of variable autoregressive (VAR) graph generation is $O(N^2)$, where N is the total number of nodes, and $K = \log_a N + 1$ scales are used.*

Proof. Define the resolution sequence $\{n_k\}$, where $n_k = a^{k-1}$ nodes are added at scale k, and $n_K = N$. The total nodes up to scale k is:

$$S_k = \sum_{i=1}^{k} n_i = \frac{a^k - 1}{a - 1}.$$

At each scale k, generating n_k new nodes requires computing attention over all S_{k-1} existing nodes. The complexity is:

$$n_k \cdot S_{k-1} \approx a^{k-1} \cdot a^{k-1} = a^{2(k-1)}.$$

Summing over all K scales:

$$\sum_{k=1}^{K} a^{2(k-1)} = \frac{a^{2K} - 1}{a^2 - 1} \sim O(a^{2K}) = O(N^2),$$

since $a^{K-1} = N$ and thus $a^{2K} = a^2 N^2$. $\square$

C. Details for Evaluation Metrics

To evaluate the performance of `MAG` for generic graphs, we measure the degree, clustering, and orbit of the generated graphs. Degree measures the number of connections of each node, revealing hubs or sparsely connected regions. Clustering shows how likely a node's neighbors are to be connected, indicating local group structures or tightly-knit communities. Orbit counts specific small patterns (subgraphs) within the graph, capturing repeating structural patterns. These metrics report whether the structure of the graphs in the dataset is preserved, indicating more meaningful and realistic graphs.

We assess molecular graph generation by examining validity, uniqueness, novelty, and the Frechet ChemNet Distance (FCD) metric. Validity reflects the proportion of generated structures that adhere to fundamental chemical rules—such as correct valency—without any post-hoc correction. Uniqueness gauges the degree to which those valid molecules are distinct from one another, while novelty captures the extent to which they lie outside the original training corpus. Finally, FCD quantifies how closely the distribution of generated compounds matches that of real chemicals by comparing the mean and covariance of their activations in ChemNet's penultimate layer.