COHO: Context-Sensitive City-Scale Hierarchical Urban Layout Generation

Liu He, Daniel Aliaga

Purdue University, USA
{he425,aliaga}@purdue.edu

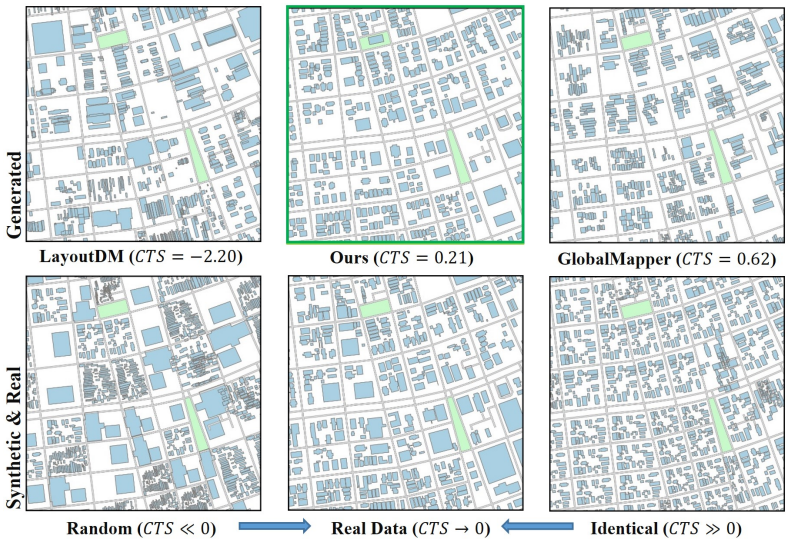

Fig. 1: Context-Sensitive Generation. Our method (Green) pursues realistic context harmonization among neighboring city blocks as real data (bottom-middle). Other methods (e.g. LayoutDM [29], GlobalMapper [20]) show over-diversity/-similarity in city-scale layout generation (evaluated by Context Score CTS as Eq. 4.). Fully random and identical layouts are synthetically generated to illustrate extreme cases.

Abstract. The generation of large-scale urban layouts has garnered substantial interest across various disciplines. Prior methods have utilized procedural generation requiring manual rule coding or deep learning needing abundant data. However, prior approaches have not considered the context-sensitive nature of urban layout generation. Our approach addresses this gap by leveraging a canonical graph representation for the entire city, which facilitates scalability and captures the multi-layer semantics inherent in urban layouts. We introduce a novel graph-based masked autoencoder (GMAE) for city-scale urban layout generation. The method encodes attributed buildings, city blocks, communities and cities into a unified graph structure, enabling self-supervised

masked training for graph autoencoder. Additionally, we employ scheduled iterative sampling for 2.5D layout generation, prioritizing the generation of important city blocks and buildings. Our approach achieves good realism, semantic consistency, and correctness across the heterogeneous urban styles in 330 US cities. Codes and datasets are released at `https://github.com/Arking1995/COHO`.

Keywords: Layout Generation · Urban Modeling · Masked Graph Autoencoder · Context Sensitivity

1 Introduction

Large-scale urban layout generation has received significant interest by computer vision, urban planning and related disciplines. In particular, cities easily have 1,000 to 50,000 city blocks spanning a mostly horizontal terrain. The configuration of each city block varies significantly across the city, yet there is some configuration similarity amongst subsets within and among cities. Reconstructing existing layouts and generating future planned layouts are crucial tasks for urban simulation, digital twins, and game/content design.

Prior works have generated urban layouts from a variety of sources. Procedural layout generation [39, 57] used hand-coded rules to generate cities, and inverse procedural modeling [5] extracted the rules from data and then enabled generation. More recently, deep learning has promoted data-driven approaches that produce either pixel-based layout generation (e.g., InfiniCity [38], CityDreamer [61], CityGEN [13]) or graph-based approaches for general polygonal shape layouts (e.g., LayoutTransformer [17], VTN [2]) and for urban scenarios (e.g., BlockPlanner [62], GlobalMapper [20], BuildingGAN [11]). However, none of these techniques explicitly recover and consider the context-sensitive nature of urban layout generation (Fig. 1). In other words, the layout of buildings, blocks, and communities cannot be treated in isolation. The multi-layer semantics are critical to generating realistic and plausible urban layouts. While such multi-level structure (e.g., from single building to a community or subset of a city) has been specified via hand-coded rules, it has not been discovered, organized and used for generation in data-driven deep learning approaches.

Our approach builds on three key observations:

1. **Representation:** Cities, communities, blocks, buildings, and roads are arbitrarily shaped and with complex topology. Hence, a graph-based representation can better capture the regular and non-regular configurations, as opposed to the limiting regular structure of an image. Further, graphs can be more compact which in turn supports better scalability. Altogether, having buildings or city blocks as atomic units, instead of pixels, will contribute to improving correctness, realism, and consistency with priors.
2. **Context-Sensitivity:** Buildings, city blocks, and communities are built considering their neighboring structures and not in isolation. Hence, a generator should consider this multi-layer structure and inter-dependency. In

order words, the style features at one level (e.g., size, shape, location) depend on the surrounding context.
3. **Prioritization:** The stylistic and semantic importance, or priority, of city blocks and buildings varies. This implies that a generation scheme should generate the most important city blocks and buildings first and then fill-in the rest, as opposed to a region growing approach for example. Overall, prioritization will improve correctness and realism as well as enable large-scale urban layout generation.

We propose a novel Graph-based Masked AutoEncoder (GMAE) for large-scale multi-layer 2.5D urban layout generation. *First*, we define a canonical graph-based representation capturing the multi-layer context sensitivity of any city's urban layouts (Fig. 2). The entire graph G corresponds to a city. Subsets of the graph map to communities and each node b amounts to a city block. Further, each node encodes its building layouts, shapes, and heights as a quantized feature by well-trained quantizor. *Second*, we utilize our GMAE to learn the multi-layer stylistic behavior of many urban layouts (i.e., 330 cities).

Using subsets of the graph (or communities), node features are randomly masked and self-supervised GMAE training is performed to learn node feature reconstruction (Fig. 3). *Third*, utilizing the well-trained GMAE as a generator, we perform an iterative priority-based scheduling to generate a large layout or to complete a city fragment; i.e., each iteration predicts node features for the entire graph but only the most confident ones are preserved for the next step (Fig. 4). This methodology is able to leverage any percentage of given prior (e.g., $[0, 100\%]$) for urban layout completion and generation.

To the best of our knowledge, our approach is the first data-driven deep-learning method to enable large-scale 2.5D layout generation with good realism, semantic consistency, and correctness. We show the capability to generalize to the heterogeneous urban layout styles of 330 cities in the US (e.g., all cities with a population >100K, totaling 833,473 city blocks and 17,663,607 buildings). In addition, we provide comparisons to several prior works including SDXL [48]), VTN [2], LayoutDM [29], and GlobalMapper [20] in Sec. 4 and exceed their performance in multiple well-known geometric and perceptual metrics.

Our contributions include the following:

- A canonical graph representation for large-scale 2.5D urban layouts. It encodes global and local node features and neighboring relations for buildings, city blocks, and communities within a city.
- A self-supervised Graph-based Masked AutoEncoder (GMAE) enabling arbitrarily shaped city layout generation with consistency, correctness, and realism.
- Priority-based scheduled iterative sampling for controllable urban layout synthesis including completion, refinement, and generation with any percentage of priors ($[0, 100\%]$).
- A open dataset of road network and urban layouts of 330 US cities for benchmarking current and future urban layout generation methods (`https://huggingface.co/datasets/Arking95/COHO`).

2 Related Work

Layout Generation. Numerous data-driven deep learning methods have been proposed for layout generation; for example, generalized layout synthesis [2, 17, 31, 34, 64], document layouts [21, 30, 46, 55, 67], urban land lots [22, 62, 66], indoor scenes [42, 44, 60], and 3D buildings [3, 11, 45]. Recently, diffusion models have also been used for layout generation tasks (e.g., [9, 21, 28, 29]). However, most of these works assume 2D layouts generated on a rectangular empty canvas. The position and size of each bounding box is often limited to a finite set of categories and to rectilinear alignments. This results in poor diversity and low realism for urban layout generation tasks. BlockPlanner [62] and GlobalMapper [20] adapt layout generation to either rectilinear or arbitrarily-shaped city block generation. However, all previous works assume each layout, or city block, is independent. This is not the case in urban environments where adjacent and nearby city blocks may share a common style or be positioned in a purposeful multi-block arrangement. Without a contextual guidance, city block generation may result in either unwanted harmonization or unexpected diversity between neighboring blocks (see Fig. 1). Thus, we pursue city-scale urban layout generation with the help of context-aware neighboring information.

Infinite Visual Synthesis. With bombing of visual deep learning works [27, 40, 50, 51, 53, 54], researchers have explored synthesizing large, potentially unbounded images. For example, omnidirectional image synthesis from multiple smaller field-of-view images [36, 37]. Other works focus on unbounded 3D scene generation [8, 12, 49]. Particularly relevant are several recent infinite urban scene generators [13, 38, 61] that directly adapt image synthesis methods (e.g. InfinityGAN [37], MaskGIT [10]). The synthesis performs an autoregressive outpainting process using layout image patches. However, patch-based training and inference doesn't learn global features nor their interdependencies. Rather, it learns a very local behavior. Prior pixel-based methods acknowledge limited realism and semantics [13, 38, 61]. Often, post-processing [13, 61], and human-in-the-loop editing [38] is required to refine the generated images to vector-based urban layouts. Even the latest stable diffusion model SDXL [48] seeks high-resolution image synthesis but may not produce realistic urban layouts (Fig. 5).

Learning based on Quantized Representations. Vector quantization has been a prevailing stepping-stone technique for text (e.g., Word2Vec [41]), image (e.g., VQVAE/GAN [16, 56], [15]), and video synthesis (e.g., SORA [6, 47], [63]). Pretrained quantizers provide a scalable token representation benefiting high-level and large-scale synthesis. A well-trained quantizer combined with self-supervised masked training often performs well in reconstructing the original signal during generative tasks. BERT [14] provides plausible text-based representation learning. MAE [19] defines a masked autoencoder for image space which excels at multiple downstream tasks. MAE also provides representation

learning for a broad range of node/graph feature synthesis [25, 26]. Specifically, MaskGIT [10] and MAGE [35] utilize MAE frameworks for impressive image generation tasks. The idea of a generative masked autoencoder based on pretrained quantization was inspirational to our city-scale urban layout approach.

3 Method

First we describe our canonical graph representation of 2.5D urban layouts (Fig. 2). Second, we explain the training process for our GMAE based on the graphs for many cities (Fig. 3). Third, we show our priority-based scheduling for iteratively generating city-scale urban layouts given any amount of prior (Fig. 4).

3.1 Canonical Graph Representation

A city is represented by a graph $G = \{B, E\}$, where B is a set of nodes $B = \{b_i | i \in [1, N]\}$ such that b_i represents each city block (Fig. 2), and any connected subgraph of G corresponds to a community. $N = |B|$ is the total number of blocks for a given city. E is a set of edges $E = \{e_{ij} | i, j \in [1, N]\}$ representing mutually adjacent nodes (or city blocks). $K = |E|$ is the total number of edges.

Node (or City Block). Each node, or city block, b_i is a region defined by an enclosing road network and contains a set of block-level features and building-level features. The per-node features include:

- s_i is the shape and location feature vector of a city block contour. It includes four features: (1) aspect ratio, (2) total block area, (3) ratio of total block area over the block's convex-hull area, and (4) relative distance from current block to the centroid of the city.
- q_i is a 512-dimensional vector from a codebook C. It hierarchically captures the buildings layouts and their configuration within b_i.

We concatenate these node features to form a 516-dimensional vector ($[s_i, q_i]$), which is able to represent our large and heterogeneous set of blocks and buildings.

Building Layout Quantization. The 512-dimensional vector q_i describing the building layouts within a city block is obtained from a quantized codebook $C = \{1, 2, ..., L\}$ with quantization level $L = |C|$. To create the codebook, we train a block-level variational autoencoder (BVAE) for building layout within a city block, using a canonical graph structure for representing the multiple building shapes, positions, and heights inside a single city block. Using this pre-trained encoder, the latent vector of all possible building layouts will form a numerical distribution for each latent dimension. We further quantize each of these distributions into L bins with equal percentiles. Then, each latent value is replaced by the rank of the bin containing its value. Therefore, any building

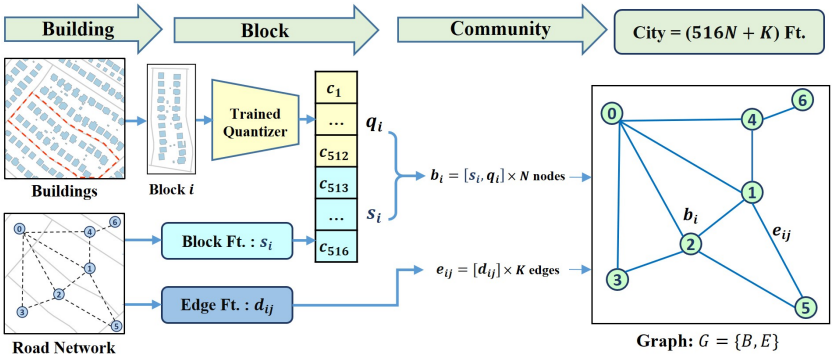

Fig. 2: Canonical Graph Representation. Our method represents a city as a canonical graph G. Each node b_i represent a single city block, and each edge e_{ij} connects spatially adjacent blocks. Each block/node corresponds to a set of node features s_i and a quantized vector q_i hierarchically capturing enclosed building layouts. Edge feature d_{ij} encodes distances between block centroids. Graph G is used for GMAE training.

layout within a city block is quantized into a 512-dimensional categorical index vector $q_i = [c_1, ..., c_{512}]$, where each $c \in C$. Note that our codebook is defined after BVAE training and not dynamically trainable as in VQGAN/VQVAE [16,56]. After performing a set of detailed evaluations using different training schemes, encoding models, and quantization levels L (see Sec. 4.3), we selected the Graph Attention Network (GAT) [59] as the backbone for the BVAE. Further, we select $L = 20$ to balance compactness and representation ability after quantization. BVAE and quantization details are found in Supplementary Materials.

Edge. Each edge e_{ij} contains a feature d_{ij} storing the relative distance between the centroids of the adjacent city blocks b_i and b_j. This distance-weighted adjacency helps encode the relationship between neighboring blocks.

Altogether, the shape, style, and multi-layer urban layout of a city with N nodes and K edges is represented by a total of $(516N + K)$ variables. This canonical graph is used for the self-supervised GMAE training described below.

3.2 Graph-based Masked AutoEncoder

We train our GMAE using a self-supervised process. A road-only city (or community) is a graph (or subgraph) G where building layout features q_i of every node (block) are missing, but s_i and e_{ij} are preserved. A context-sensitive node feature generation process corresponds to recreating the purposefully removed building layout features.

During training (Fig. 3), we use dynamic masking ratios of q_i in order to learn how to regenerate the original features. In each training iteration, the mask ratio m is sampled from a truncated Gaussian distribution as recommended by Li et al. [35]. The ratio is obtained from the range $m \in [0.5, 1.0]$ and is centered at

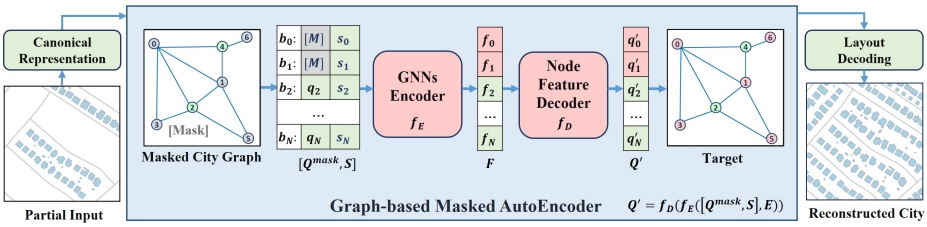

Fig. 3: Graph-based Masked Autoencoder. Given a canonical city graph, quantized building layout features Q are masked with dynamic masking ratios $m \in [0.5, 1.0]$, while block shape and location features S are kept. The GNN encoder uses message passing between neighboring nodes to obtain the context-aware node features F. The decoder uses F to reconstruct Q'. The predicted Q' are decoded to 2.5D urban layouts.

0.55. We denote building layout features as $Q = \{q_i | i \in [1, N]\}$ and those after masking as Q^{mask}. Block features are denoted as $S = \{s_i | i \in [1, N]\}$. The graph neural network (GNN) encoder is denoted as f_E, and node feature decoder as f_D. Thus the process of our GMAE can be written as:

$$F = f_E([Q^{mask}, S], E), \quad Q' = f_D(F) \tag{1}$$

where $F = \{f_i | i \in [1, N]\}$ indicates the context-sensitive node features obtained by message passing between neighboring nodes by f_E. Extracted F is processed by f_D to reconstruct building layout features Q'.

The backbones of f_E and f_D can be any GNN, including GAT [58], GCN [33], and GraphSAGE [18]. After experimentation, we select GAT as f_E because of its best context-sensitive feature extraction ability, and we choose a straightforward MLP as f_D for efficient processing. We observed that the re-masking trick and complex GNN decoder utilized by [25,26] requires more memory and calculation but has no net benefit in our application. Since Q contains quantized index vectors, we use cross entropy loss for self-supervised reconstruction training on masked nodes. Detailed ablations of masking strategy and model selections are provided in Sec. 4.3.

Our objective function is as follows, where L indicates the length of the quantized codebook C:

$$L_{recon} = -\sum_{i=1}^{L} [Q_i \log(Q'_i)]^{mask} \tag{2}$$

To capture community behavior (and also reduce GPU memory usage), we randomly sample a batch of subgraphs from G for each training iteration. The nodes in each subgraph are sampled using a chosen radius from a random center node. While community sizes vary significantly, we found that a practical compromise is to use 500 meters as the radius value. This radius ensures to cover a reasonable number of the city blocks with a typical census tract from governmental CJEST dataset [1].

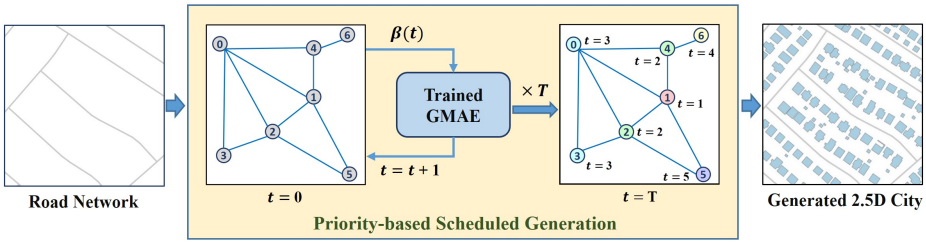

Fig. 4: Priority-based Scheduled Generation. We iteratively utilize pretrained GMAE to reconstruct masked node features. In each iteration, we accept a certain ratio of predicted nodes decided by the scheduling function $\beta(t) = 1 - cos(t/T)$. We obtain a full graph after T iterations.

3.3 Priority-based Scheduled Generation

While in theory the trained GMAE can predict all nodes in a single pass, the generation quality is much better when performed iteratively. Recent published papers (e.g., [10,35]) have found that gradually sampling with a reasonable number of iterations improves generation quality. In urban scenarios, small number of representative city blocks are important to indicate context behavior. We intend to generate those blocks at early iterations and then span neighboring blocks.

In Fig. 4, we propose a T-iteration sampling strategy similar to MaskGIT [10] and MAGE [35]. At each iteration, GMAE predicts per-node building layout features for all remaining nodes. Then we rank generated nodes by their prediction confidence. The generated nodes with the highest prediction confidence are accepted. The accepted nodes after the current iteration $t \in [1, T]$ are scheduled by a cosine function $\beta(t) = 1 - cos(t/T)$. When t is small, the ratio is near to 0 causing few nodes to be accepted in initial iterations. As iterations continue, t gets bigger and hence the number of accepted nodes during each iteration will also increase. This behavior follows the prioritization observation in Sec. 1 which causes distinguished, or unique, city blocks to be generated first and essentially guide the layout style for nearby city blocks. Typically we set $T = 12$ which results in dozens of nodes being generated as is the typical size of a community. Detailed ablations of alternative sampling schedules are provided in Sec. 4.3.

4 Experiments

4.1 Implementation Details

Open Dataset We have created a vector-based urban layout dataset of all US cities with a population of 100K or more. *We believe this dataset will be of significant use to the community to further develop urban layout generation methods and thus we will release the dataset together with the paper.* For each city, we define a rectangular bounding box containing most of the metropolitan area. Then,

we use this bounding box to extract data from OpenStreetMap (OSM) [43], Microsoft Building Footprints (MSF) [23], and Topologically Integrated Geographic Encoding and Referencing (TIGER) dataset [7]. The data we collected includes: (1) road network vectors and attributes, (2) building footprints and heights, and (3) block-level social economic metrics. Specifically, we collect road networks and block contours from TIGER. Then we look for building footprints and heights from OSM and MSF within each city block contour. We choose the one with more and accurate details (e.g., containing more buildings, having features that match those in TIGER). Over all 330 cities, this dataset has a total of 833,473 city blocks, and 17,663,607 buildings. We set training, validation, testing split as 70%, 20%, and 10% respectively.

Training Details The training has two stages. We first use all city blocks for self-supervised training of a graph-based Block-level Variational AutoEncoder (BVAE). Given this trained block-level encoder, we perform the latent quantization for all building layouts in each city block. Then our GMAE is trained based on canonical city graph representation. Training time for BVAE and GMAE typically takes 12 hours and 15 hour respectively by a single A5000 GPU. We will release our codes upon acceptance.

Inferencing using our graph-based approach is seemingly instantaneous for all the results shown in this paper. This is contrast to the VTN [2] and SDXL [48] approaches which take significantly more time to infer and to train.

4.2 Comparisons

We compare our method to several alternate city-scale layout generators (Tab. 1). Specifically, we randomly select 100 communities from among our test set, total 3700 city blocks. All methods generate the layout of the 100 communities in one pass, and without any post-processing or human-in-the-loop refinement. Implementations are downloaded from the author's repositories and trained with default parameter settings. For fairness, we convert our dataset to token sequences, graphs, or image patches (1024×1024) as needed. VTN [2], LayoutDM [29], GlobalMapper [20] are retrained for single block generation. SDXL [48] is fine-tuned for outpainting by using the upper half to predict the lower half.

Evaluation Metrics We define a Context Score (CTS) to evaluate the context relationships among neighboring city blocks (i.e., a community). This metric is a numerical extension of LayoutSim [20] and DocSim index [46] which evaluate the similarity between two city blocks regarding location, size, and category. Specifically, for each generated city block b_i, we calculate the average LayoutSim with its neighbors $N(i)$. Then, the local average of LayoutSim represents the context similarity among the community (CT). High CT indicates identical blocks in a community and low CT corresponds to high diversity in a community. CTS is the subtraction of the real community's CT from the generated CT:

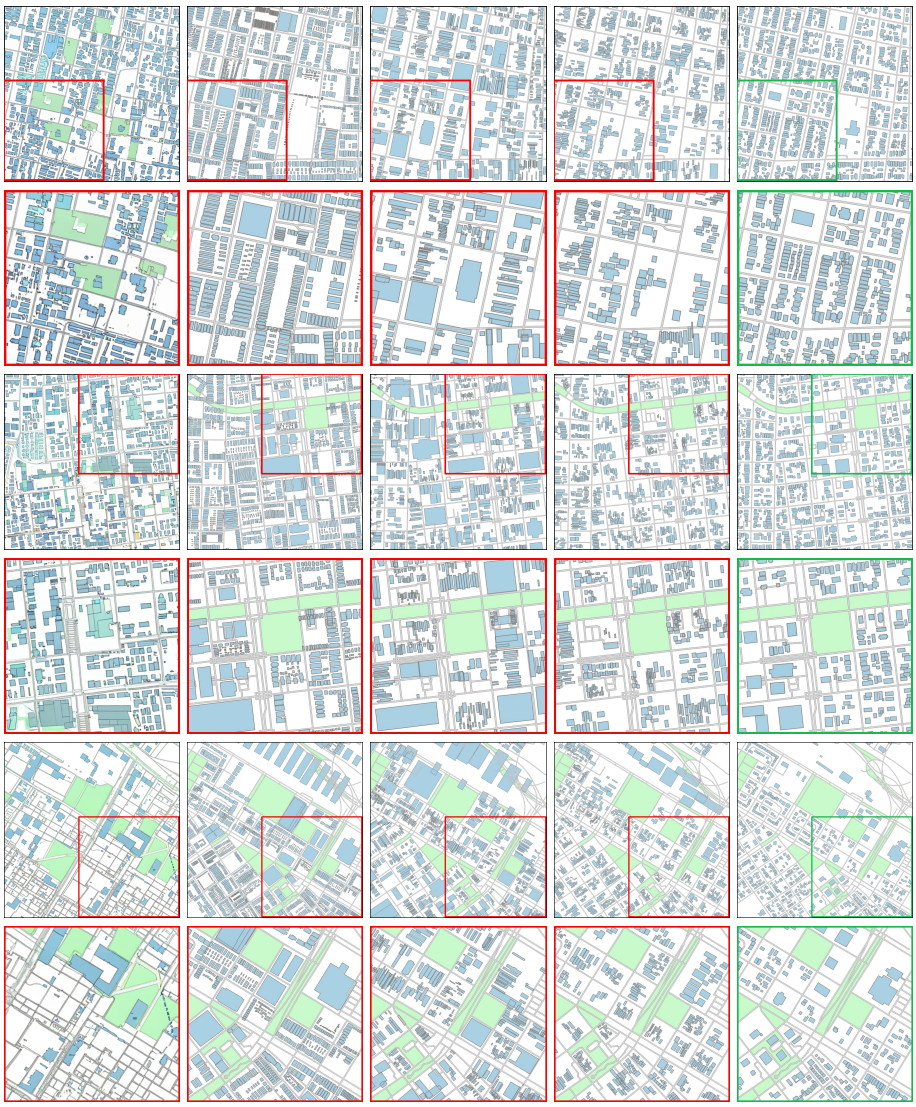

Fig. 5: Qualitative Comparisons. Given the same road network (except SDXL [48]), all above methods generate urban layouts in only one pass without post-processing or human-in-the-loop refinement. The "even rows" are a zoom-in of the highlighted areas in the "odd rows". Our method generates a realistic distribution of urban layouts with plausible context-dependent behaviors (as indicated by CTS score). VTN [2] and LayoutDM [29] show abnormal style shift among neighboring blocks, and no awareness of road networks. GlobalMapper [20] generates over-similar communities. SDXL [48] provides poor outpainting quality containing semantic errors and undesired overlaps.

Table 1: Quantitative Comparisons. All methods are compared to the same real urban layouts (except SDXL [48] which cannot take-in a road network). Best values are in bold, second best values are underlined. Our method outperforms other existing methods in all but the overlap metric. See text for an explanation of the metrics.

Method	$CTS_{x\to 0}$	WD-5D↓	WD-CO↓	Overlap↓	O-Blk↓	FID↓	KID↓	LPIPS↓
SDXL [48]	-	-	-	-	-	120.24	0.079	0.48
VTN [2]	-1.14	3.18	5.81	**1.24**	7.35	69.14	0.047	<u>0.32</u>
LayoutDM [29]	-2.20	<u>2.92</u>	12.50	4.56	1.72	66.77	0.040	0.39
GlobalMP [20]	<u>0.62</u>	4.77	<u>4.14</u>	2.52	<u>0.68</u>	<u>49.55</u>	<u>0.024</u>	0.34
Ours	**0.21**	**2.28**	**1.91**	<u>1.27</u>	**0.42**	**23.63**	**0.005**	**0.20**

$$CT = \frac{1}{\|N(i)\|} \sum_{j}^{N(i)} LayoutSim(b_i, b_j), \quad N(i) = \{j \mid (i,j) \in E\} \tag{3}$$

$$CTS = CT_{gen} - CT_{real} \tag{4}$$

Given a real community, the metric indicates whether a generated community is over-diverse ($CTS < 0$), or over-similar ($CTS > 0$). A low value means more realism.

In our comparisons, we compute several metrics. One metric is the Wasserstein Distance between the distributions of generated communities and real communities in terms of 2D position (x, y) and 2.5D geometry (length, width, height), and building count (CO) – we name these WD-5D and WD-CO. We define two quality metrics as well: Overlap indicates the percentage by which two building overlaps, and O-Blk represents the percentage of building area that sticks outside of its city block. We also implement several perceptual metrics: FID [24], KID [4], and LPIPS [65] to evaluate the visual quality and realism of generated layouts. For all the above metrics, lower value indicates better quality.

Quantitative Results As Tab. 1 shows, our approach performs better than existing approaches in all but one metric. Note that the poor quality of SDXL results makes it hard to decipher layouts. We only compare to it by pixel-based perceptual metrics.

Qualitative Results We show in Fig. 5 qualitative results for several exemplary areas (more examples are in Supplemental Material). In general, our method shows good contextual harmonization and has awareness of arbitrary road networks. It generates a realistic distribution of urban layouts with plausible context dependent behaviors. Other methods show improper patterns (e.g., overly diverse or similar) and potentially containing semantic errors or undesired overlaps.

Table 2: Block Quantization Ablations. We report metrics (all in %) among all alternative ablations of our Block-level graph-based Variantional AutoEncoder (BVAE). Using GAT-backbone with post-trained dimensional quantization with $L = 20$ shows the best overall compromise. The best values are in boldface.

Encode Model	Overlap↓	Out-Blk↓	Pos-E.↓	Geom-E.↓	Ct-E.↓	Cov-E.↓
LayoutVAE [32]	13.84	11.15	15.09	37.35	-	24.34
VTN [2]	2.70	4.70	6.01	22.86	18.58	1.97
BlockPlanner [62]	2.93	1.30	4.78	12.06	3.63	5.66
GlobalMapper [20]	1.04	<u>1.20</u>	<u>3.25</u>	2.83	0.08	0.46
BVAE (SAGE [18])	1.33	1.70	5.01	3.25	0.10	0.35
BVAE (GCN [33])	<u>0.45</u>	1.21	4.25	<u>2.74</u>	<u>0.04</u>	<u>0.28</u>
BVAE* (GAT [58])	**0.17**	**0.28**	**1.00**	**0.71**	**0.006**	**0.04**

Quantization	Overlap↓	Out-Blk↓	Pos-E.↓	Geom-E.↓	Ct-E.↓	Cov-E.↓
Trainable-VQ	3.02	0.55	3.75	5.78	0.07	0.60
KMeans-Q	2.90	<u>0.02</u>	20.98	17.85	50.01	16.23
DIM-Q ($L = 5$)	0.28	0.62	7.71	16.20	25.0	14.68
DIM-Q ($L = 10$)	0.56	0.28	5.73	4.48	8.33	4.58
DIM-Q ($L = 30$)	<u>0.17</u>	**0.01**	**2.68**	**0.78**	**0.005**	<u>0.15</u>
DIM-Q* **($L = 20$)**	**0.11**	**0.01**	<u>3.50</u>	<u>0.89</u>	<u>0.006</u>	**0.12**

4.3 Ablation Studies

Block Quantization We ablate block quantization in order to understand and maximize the impact of changing its parameters and design strategy. First, we report in Tab. 2 (Encode Model Table) the reconstruction loss using different block-level variational autoencoders. We evaluate using various geometric metrics: Overlap (percent of building-to-building overlap), Out-Blk (percent of buildings that stick outside of its city block), Pos-E (positional error as a percentage of block diagonal), Geom-E (2.5D building geometry error as percentage of block scale), Ct-E (percentage of building count error), and Cov-E (total building coverage error). They clearly show our approach is the best overall. Thus we choose our BVAE based on Graph Attention Networks (GAT) [58].

Then, in Tab. 2 (Quantization Table), we ablate various codebook approaches and quantitization levels for block-level encoding. "Trainable-VQ" leverages trainable vector quantization codebooks introduced by VQVAE [56]. We also provide an alternative non-training based quantization method using KMeans ("KMeans-Q") to cluster BVAE latent space values to a reasonable number of categories in order to handle diverse urban layouts (e.g., 100 classes). The "DIM-Q" rows imply quantizing each of the 512 dimensions of the block-level code into a codebook $C = \{1, 2, ..., L\}$ as described in Sec. 3.1. This per-dimension quantization performs clearly the best. Specifically, setting L to infinity produces a lossless representation of BVAE latent vectors, but also causes the curse of dimension-

Table 3: City-Scale GMAE Ablations. We ablate backbone message passing layers, maximum messaging passing hops (D: depth of encoder), masking strategy during training, and scheduling functions of iterative generation (T indicates total iterations). The overall best setting is marked as *.

Ablations	Context↓	WD-3D↓	WD-N↓	Overlap↓	Out-Blk↓	FID↓	KID↓	LPIPS↓
GMAE (GCN [33])	-0.30	2.74	2.91	**0.26**	0.36	52.95	0.025	0.32
GMAE (GTrans [52])	0.74	3.02	3.27	1.96	1.07	43.60	0.018	0.29
GMAE (SAGE [18])	0.97	2.98	2.36	1.75	2.60	43.15	0.017	0.29
GMAE (GAT, Dec)	0.94	2.45	2.96	2.13	1.10	48.24	0.019	0.30
GMAE (GAT, $D = 1$)	1.25	3.03	2.64	1.73	0.44	43.75	0.018	0.29
GMAE (GAT, $D = 2$)	0.33	2.66	<u>2.01</u>	1.31	0.57	34.82	0.014	0.27
GMAE (GAT, $D = 4$)	<u>0.24</u>	**1.92**	2.31	<u>0.63</u>	**0.28**	<u>24.59</u>	<u>0.009</u>	<u>0.22</u>
GMAE ($Mask \in [0.05, 1]$)	1.30	2.70	3.62	1.70	2.45	43.14	0.018	0.29
GMAE ($Mask \in [0.05, 0.5]$)	1.10	3.07	3.68	1.68	1.30	46.85	0.020	0.30
GMAE ($Mask = 0.15$)	1.59	4.91	3.41	1.83	1.20	49.50	0.022	0.31
GMAE (Linear, $T = 12$)	0.63	2.83	3.79	1.27	0.93	37.42	0.015	0.28
GMAE (Log, $T = 12$)	-1.31	2.77	7.27	7.41	3.14	63.25	0.038	0.33
GMAE (Cosine, $T = 1$)	-1.67	7.44	5.64	3.56	4.01	76.12	0.051	0.35
GMAE (Cosine, $T = 20$)	0.32	2.92	3.98	1.68	<u>0.31</u>	32.06	0.013	0.24
***(GAT-D3, [0.5,1], Cos-T12)**	**0.21**	<u>2.28</u>	**1.91**	1.27	0.42	**23.63**	**0.005**	**0.20**

ality. We discover that $L = 20$ per dimension is the best overall compromise balancing compactness and representation ability.

City-Scale Graph Masked Autoencoder. In Tab. 3, We ablate various model parameters, together with significant training and inference strategies for our GMAE. Each alternatives is evaluated by the same set of metrics in Sec. 4.2.

By varying the message passing layers, we find GAT [58] provides the best performance. GCN [33] produces diverse and unrealistic urban layouts, while transformer-based [52], and GraphSAGE [18] backbones generate blocks that are too similar. The depth of the stacked layers in the encoder (D) dictates the maximum message passing hops from each node and thus the span of context behaviors captured for realistic generation. We find that 3 hops is promising and larger contexts cause over-similarity and unnecessary computing burden. We also observed that using stacked GAT layers for decoding with remasking tricks, as described by [25,26], didn't produce a net benefit towards city generation. So we choose MLP as the node feature decoder for both its performance and efficiency.

The masking strategy during training is crucial for GMAE performance. We find that dynamic and large masking ratios are beneficial, while a fixed small ratio (e.g., 0.15 as in [14]) leads to over similarity. This behavior is also observed by Li et al. [35]. We adopt a similar masking strategy by sampling a Gaussian distribution $N(0.55, 0.25)$, truncated by $[0.5, 1.0]$, for each training iteration.

Scheduling for iterative generation is also critical to capture representative city blocks (Sec. 3.3). We evaluate several functions $\beta(t)$ (e.g., linear, logarithmic, or cosine-based) that provide the scheduled acceptance ratio, and evaluate alternative total iteration counts T. Rapid iteration leads to over-diversity and

long iteration results in over-similarity. Logarithmic functions with reverse pattern produces the worst realism. Metrics show that cosine-based function, which is conservative initially and generous in later iterations, outperforms others.

In summary, this study determines the best configuration for our GMAE as: GAT encoder with $D = 3$ and MLP decoder, training mask ratio $m \in [0.5, 1.0]$, and priority-based scheduling function $\beta(t) = 1 - cos(t/T)$ with $T = 12$.

4.4 Limitations

Our method faces limitations where shapes of city blocks or buildings are not simple polygons (e.g. the building with a garden inside). It also struggles to handle very irregular and concave shaped city blocks (e.g., cul-de-sac's). Also, road-network to contextual structure relations are not explicitly considered.

4.5 Additional Results

We place several other interesting results in Supplementary Materials.

- **Controllable Generation of Entire City.** Our method enables realistic and efficient city-scale controllable generation given any percentage of priors. We show several large examples spanning a subset of the many cities.
- **Socio-Economic Prediction.** Trained GMAE may act as a powerful encoder for downstream tasks. For example, the encoded GMAE latent vectors of urban layouts can indicate social-economic metrics for each city block.
- **Semantic Manipulations.** Our method enables semantic manipulations over various building layout styles across different cities.

5 Conclusions and Future Works

Large-scale urban layout generation has seen significant advancements, spurred by the intersection of computer vision, urban planning, and related disciplines. Previous methodologies often neglect the context-sensitive nature of urban layouts, failing to capture the multi-layer semantics crucial for realism and plausibility. Our proposed Graph-based Masked AutoEncoder (GMAE) addresses these limitations by leveraging a graph-based representation, acknowledging context-sensitivity, and prioritizing the generation process. By training on a diverse dataset of 330 cities, we demonstrate the efficacy of our approach in generating large-scale 2.5D layouts with realism and semantic consistency. Moreover, our method outperforms existing techniques in various metrics, marking a significant advancement in the realm of urban layout generation.

Moving forward, our approach paves the way for more accurate and efficient urban simulation, digital twin creation, and game/content design. As future work we intend to incorporate our method for large-scale photo-realistic multi-view scene synthesis, and city-scale 3D modeling. Those potentials will contribute to synthetic data generation in autonomous driving and world model training.

Acknowledgements

This project was funded in part by NSF Grant #2107096 and NSF Grant #1835739.

References

1. Climate and economic justice screening tool. https://screeningtool.geoplatform.gov/
2. Arroyo, D.M., Postels, J., Tombari, F.: Variational transformer networks for layout generation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 13642–13652 (2021)
3. Bhatt, M., Kalyanam, R., Nishida, G., He, L., May, C., Niyogi, D., Aliaga, D.: Design and deployment of photo2building: A cloud-based procedural modeling tool as a service. In: Practice and Experience in Advanced Research Computing, pp. 132–138 (2020)
4. Bińkowski, M., Sutherland, D.J., Arbel, M., Gretton, A.: Demystifying mmd gans. arXiv preprint arXiv:1801.01401 (2018)
5. Bokeloh, M., Wand, M., Seidel, H.P.: A connection between partial symmetry and inverse procedural modeling. In: ACM SIGGRAPH 2010 papers, pp. 1–10 (2010)
6. Brooks, T., Peebles, B., Holmes, C., DePue, W., Guo, Y., Jing, L., Schnurr, D., Taylor, J., Luhman, T., Luhman, E., Ng, C., Wang, R., Ramesh, A.: Video generation models as world simulators (2024), https://openai.com/research/video-generation-models-as-world-simulators
7. Bureau, U.S.C.: Topologically integrated geographic encoding and referencing. https://www.census.gov/geographies/mapping-files/time-series/geo/tiger-line-file.html
8. Chai, L., Tucker, R., Li, Z., Isola, P., Snavely, N.: Persistent nature: A generative model of unbounded 3d worlds. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 20863–20874 (2023)
9. Chai, S., Zhuang, L., Yan, F.: Layoutdm: Transformer-based diffusion model for layout generation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 18349–18358 (2023)
10. Chang, H., Zhang, H., Jiang, L., Liu, C., Freeman, W.T.: Maskgit: Masked generative image transformer. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11315–11325 (2022)
11. Chang, K.H., Cheng, C.Y., Luo, J., Murata, S., Nourbakhsh, M., Tsuji, Y.: Building-gan: Graph-conditioned architectural volumetric design generation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 11956–11965 (2021)
12. Chen, Z., Wang, G., Liu, Z.: Scenedreamer: Unbounded 3d scene generation from 2d image collections. arXiv preprint arXiv:2302.01330 (2023)
13. Deng, J., Chai, W., Guo, J., Huang, Q., Hu, W., Hwang, J.N., Wang, G.: Citygen: Infinite and controllable 3d city layout generation. arXiv preprint arXiv:2312.01508 (2023)
14. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)

15. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929 (2020)
16. Esser, P., Rombach, R., Ommer, B.: Taming transformers for high-resolution image synthesis. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 12873–12883 (2021)
17. Gupta, K., Lazarow, J., Achille, A., Davis, L.S., Mahadevan, V., Shrivastava, A.: Layouttransformer: Layout generation and completion with self-attention. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 1004–1014 (2021)
18. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. Advances in neural information processing systems **30** (2017)
19. He, K., Chen, X., Xie, S., Li, Y., Dollár, P., Girshick, R.: Masked autoencoders are scalable vision learners. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 16000–16009 (2022)
20. He, L., Aliaga, D.: Globalmapper: Arbitrary-shaped urban layout generation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 454–464 (2023)
21. He, L., Lu, Y., Corring, J., Florencio, D., Zhang, C.: Diffusion-based document layout generation. In: Fink, G.A., Jain, R., Kise, K., Zanibbi, R. (eds.) Document Analysis and Recognition - ICDAR 2023. pp. 361–378. Springer Nature Switzerland, Cham (2023)
22. He, L., Shan, J., Aliaga, D.: Generative building feature estimation from satellite images. IEEE Transactions on Geoscience and Remote Sensing **61**, 1–13 (2023)
23. Heris, M.P., Foks, N.L., Bagstad, K.J., Troy, A., Ancona, Z.H.: A rasterized building footprint dataset for the united states. Scientific data **7**(1), 207 (2020)
24. Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium. Advances in neural information processing systems **30** (2017)
25. Hou, Z., He, Y., Cen, Y., Liu, X., Dong, Y., Kharlamov, E., Tang, J.: Graphmae2: A decoding-enhanced masked self-supervised graph learner. In: Proceedings of the ACM Web Conference 2023. pp. 737–746 (2023)
26. Hou, Z., Liu, X., Cen, Y., Dong, Y., Yang, H., Wang, C., Tang, J.: Graphmae: Self-supervised masked graph autoencoders. In: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. pp. 594–604 (2022)
27. Hua, H., Shi, J., Kafle, K., Jenni, S., Zhang, D., Collomosse, J., Cohen, S., Luo, J.: Finematch: Aspect-based fine-grained image and text mismatch detection and correction. arXiv preprint arXiv:2404.14715 (2024)
28. Hui, M., Zhang, Z., Zhang, X., Xie, W., Wang, Y., Lu, Y.: Unifying layout generation with a decoupled diffusion model. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1942–1951 (2023)
29. Inoue, N., Kikuchi, K., Simo-Serra, E., Otani, M., Yamaguchi, K.: Layoutdm: Discrete diffusion model for controllable layout generation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10167–10176 (2023)
30. Jiang, Z., Guo, J., Sun, S., Deng, H., Wu, Z., Mijovic, V., Yang, Z.J., Lou, J.G., Zhang, D.: Layoutformer++: Conditional graphic layout generation via constraint serialization and decoding space restriction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 18403–18412 (2023)

31. Jyothi, A.A., Durand, T., He, J., Sigal, L., Mori, G.: Layoutvae: Stochastic scene layout generation from a label set. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 9895–9904 (2019)
32. Jyothi, A.A., Durand, T., He, J., Sigal, L., Mori, G.: Layoutvae: Stochastic scene layout generation from a label set. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV). pp. 9894–9903 (2019). https://doi.org/10.1109/ICCV.2019.00999
33. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
34. Li, J., Yang, J., Hertzmann, A., Zhang, J., Xu, T.: Layoutgan: Synthesizing graphic layouts with vector-wireframe adversarial networks. IEEE Transactions on Pattern Analysis and Machine Intelligence **43**(7), 2388–2399 (2020)
35. Li, T., Chang, H., Mishra, S., Zhang, H., Katabi, D., Krishnan, D.: Mage: Masked generative encoder to unify representation learning and image synthesis. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2142–2152 (2023)
36. Li, Z., Wang, Q., Snavely, N., Kanazawa, A.: Infinitenature-zero: Learning perpetual view generation of natural scenes from single images. In: European Conference on Computer Vision. pp. 515–534. Springer (2022)
37. Lin, C.H., Lee, H.Y., Cheng, Y.C., Tulyakov, S., Yang, M.H.: Infinitygan: Towards infinite-pixel image synthesis. arXiv preprint arXiv:2104.03963 (2021)
38. Lin, C.H., Lee, H.Y., Menapace, W., Chai, M., Siarohin, A., Yang, M.H., Tulyakov, S.: Infinicity: Infinite-scale city synthesis. arXiv preprint arXiv:2301.09637 (2023)
39. Lipp, M., Scherzer, D., Wonka, P., Wimmer, M.: Interactive modeling of city layouts using layers of procedural content. In: Computer Graphics Forum. vol. 30, pp. 345–354. Wiley Online Library (2011)
40. Ma, H., Zeng, D., Liu, Y.: Learning individualized treatment rules with many treatments: A supervised clustering approach using adaptive fusion. Advances in Neural Information Processing Systems **35**, 15956–15969 (2022)
41. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
42. Nauata, N., Chang, K.H., Cheng, C.Y., Mori, G., Furukawa, Y.: House-gan: Relational generative adversarial networks for graph-constrained house layout generation. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16. pp. 162–177. Springer (2020)
43. OpenStreetMap contributors: Planet dump retrieved from https://planet.osm.org . https://www.openstreetmap.org (2017)
44. Para, W., Guerrero, P., Kelly, T., Guibas, L.J., Wonka, P.: Generative layout modeling using constraint graphs. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 6690–6700 (2021)
45. Patel, P., Kalyanam, R., He, L., Aliaga, D., Niyogi, D.: Deep learning-based urban morphology for city-scale environmental modeling. PNAS nexus **2**(3), pgad027 (2023)
46. Patil, A.G., Ben-Eliezer, O., Perel, O., Averbuch-Elor, H.: Read: Recursive autoencoders for document layout generation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. pp. 544–545 (2020)
47. Peebles, W., Xie, S.: Scalable diffusion models with transformers. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 4195–4205 (2023)

48. Podell, D., English, Z., Lacey, K., Blattmann, A., Dockhorn, T., Müller, J., Penna, J., Rombach, R.: Sdxl: Improving latent diffusion models for high-resolution image synthesis. arXiv preprint arXiv:2307.01952 (2023)
49. Shen, Y., Ma, W.C., Wang, S.: Sgam: Building a virtual 3d world through simultaneous generation and mapping. Advances in Neural Information Processing Systems **35**, 22090–22102 (2022)
50. Sheng, Y., Liu, Y., Zhang, J., Yin, W., Oztireli, A.C., Zhang, H., Lin, Z., Shechtman, E., Benes, B.: Controllable shadow generation using pixel height maps. In: European Conference on Computer Vision. pp. 240–256. Springer (2022)
51. Sheng, Y., Yu, Z., Ling, L., Cao, Z., Zhang, X., Lu, X., Xian, K., Lin, H., Benes, B.: Dr. bokeh: Differentiable occlusion-aware bokeh rendering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4515–4525 (2024)
52. Shi, Y., Huang, Z., Feng, S., Zhong, H., Wang, W., Sun, Y.: Masked label prediction: Unified message passing model for semi-supervised classification. arXiv preprint arXiv:2009.03509 (2020)
53. Song, Y., Zhang, Z., Lin, Z., Cohen, S., Price, B., Zhang, J., Kim, S.Y., Aliaga, D.: Objectstitch: Object compositing with diffusion model. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 18310–18319 (2023)
54. Song, Y., Zhang, Z., Lin, Z., Cohen, S., Price, B., Zhang, J., Kim, S.Y., Zhang, H., Xiong, W., Aliaga, D.: Imprint: Generative object compositing by learning identity-preserving representation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8048–8058 (2024)
55. Tabata, S., Yoshihara, H., Maeda, H., Yokoyama, K.: Automatic layout generation for graphical design magazines. In: ACM SIGGRAPH 2019 Posters, pp. 1–2 (2019)
56. Van Den Oord, A., Vinyals, O., et al.: Neural discrete representation learning. Advances in neural information processing systems **30** (2017)
57. Vanegas, C.A., Kelly, T., Weber, B., Halatsch, J., Aliaga, D.G., Müller, P.: Procedural generation of parcels in urban modeling. In: Computer graphics forum. vol. 31, pp. 681–690. Wiley Online Library (2012)
58. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. arXiv preprint arXiv:1710.10903 (2017)
59. Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y., et al.: Graph attention networks. stat **1050**(20), 10–48550 (2017)
60. Wu, W., Fu, X.M., Tang, R., Wang, Y., Qi, Y.H., Liu, L.: Data-driven interior plan generation for residential buildings. ACM Transactions on Graphics (TOG) **38**(6), 1–12 (2019)
61. Xie, H., Chen, Z., Hong, F., Liu, Z.: Citydreamer: Compositional generative model of unbounded 3d cities. arXiv preprint arXiv:2309.00610 (2023)
62. Xu, L., Xiangli, Y., Rao, A., Zhao, N., Dai, B., Liu, Z., Lin, D.: Blockplanner: City block generation with vectorized graph representation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5077–5086 (2021)
63. Yan, W., Zhang, Y., Abbeel, P., Srinivas, A.: Videogpt: Video generation using vq-vae and transformers. arXiv preprint arXiv:2104.10157 (2021)
64. Yang, C.F., Fan, W.C., Yang, F.E., Wang, Y.C.F.: Layouttransformer: Scene layout generation with conceptual and spatial diversity. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 3732–3741 (2021)
65. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 586–595 (2018)

66. Zhang, X., Ma, W., Varinlioglu, G., Rauh, N., He, L., Aliaga, D.: Guided pluralistic building contour completion. The Visual Computer **38**(9), 3205–3216 (2022)
67. Zheng, X., Qiao, X., Cao, Y., Lau, R.W.: Content-aware generative modeling of graphic design layouts. ACM Transactions on Graphics (TOG) **38**(4), 1–15 (2019)