Supplementary of "COHO: Context-Sensitive City-Scale Hierarchical Urban Layout Generation"

Liu He, Daniel Aliaga

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

Overview

Below is a summary of the contents in each section of this supplemental material:

- Sec. 1: Details of our dataset collection method.
- Sec. 2: Details of building layout quantization, including graph-based Block-level Variational AutoEncoder (BVAE), and different quantization approaches. We also provide additional evaluation in real units (e.g. m, m^2).
- Sec. 3: City-scale controllable generation examples for 17 cities. And some complementary comments on other alternative approaches.
- Sec. 4: Semantic manipulations between different building layout styles.
- Sec. 5: GMAE-based Socio-Economic Metric Prediction
- Sec. 6: Estimation of running time and model parameter sizes.

1 Dataset Collection

Data Resources. We selected all US cities with more than 100K population (i.e., 330 cities) as our building layout dataset. The city list is based on [2]. For each city, we define a rectangular bounding box containing most of the metropolitan area. This geo-registered bounding box is used to extract data from OpenStreetMap (OSM) [11], Microsoft Building Footprints (MSF) [7], and Topologically Integrated Geographic Encoding and Referencing (TIGER) dataset [4]. The TIGER dataset provides the city block contour and corresponding social economic metrics (e.g. population, income level, etc.).

MSF and OSM provide the building layout polygons with corresponding height values. We define heuristics to composite both data resources. Specifically, MSF is the result of deep-learning-based building segmentation from high-resolution satellite images ($0.5m$). However, fine details and closely located buildings are challenging for that methodology. Such a situation is very common in big cities like New York (i.e., a big city block appears to have one large building but in reality there are many adjacent buildings). In contrast, the building layouts from OSM are manually crafted or adopted from governmental/open-source datasets. Buildings are vector-based polygons regardless of adjacency. Unfortunately, the data coverage of OSM, especially for building height values, varies significnantly, while satellite-resourced MSF overcomes this shortage. We merge the information from the two sources as described in the following section.

Data Composition. For each single city block, we evaluate and composite the building layouts extracted from both MSF and OSM by several rules: (1) we keep buildings from both resources that only appear once with no mutual overlapping; (2) for the regions that building layouts from both resources are highly overlapped, we keep the ones with more building numbers; and (3) we mainly take building height values from MSF. If height values are only available for some buildings in a city block, then we take the average height augmented with small random offsets and assign those heights to the rest of the buildings. Further, the connectivity between city blocks is captured as a graph adjacency matrix for our graph canonical representation as specified in the main paper. Each city is encoded as a single graph. Thus our dataset contains 330 graph structures. It includes 833,473 city blocks and 17,663,607 buildings, all with building heights assigned. As we claimed in the main paper, we will release the dataset upon the acceptance of our paper.

2 Building Layout Quantization

The main goal of building layout quantization is to define a compact and scalable representation (e.g., a small codebook) that keeps as much of the original building layout features as possible. In the following, we report our various experiments until arriving at the solution we ultimately used.

2.1 Block-level Variational AutoEncoder (BVAE)

To represent individual city blocks, we use a variational autoencoder. In this, all building layout features (e.g., building size, height, position) within a city block are represented as a canonical graph. This graph-representation is utilized in self-supervised training with both reconstruction loss and KL divergence loss terms. Our block representation is based on the canonical spatial transformation and graph-based representation described in GlobalMapper [6] (official repository: GlobalMapper).

Our BVAE uses GAT [14] as the backbone. The encoder stacks three GAT layers and the same structure for the decoder. Specifically, we select the multi-head number for GAT as 12, the internal feature dimension of GAT layers as 256, and the bottleneck latent dimension as 512. We found the aforementioned hyper-parameters are key to the reconstruction performance of BVAE. Reducing these parameter values hurts performance, and further increasing has no net benefit. Batch normalization is added to each layer for stable convergence. Our code will be released upon the acceptance of our paper.

2.2 Quantization Approaches

Trainable Vector Quantization. We experimented with the trainable vector quantization method described in VQGAN [5] by adapting their official implementation VQGAN to our BVAE. After performing a round of hyperparameter

tuning experiments, we did not converge to a trained result that adequately reconstructs the original building layouts. In particular, the network struggles to keep stability between the BVAE latent space and the trained codebook – e.g., when inspecting latent space values near a codebook entry, unstable block structures are created. A predefined codebook size may not be able to adapt to the heterogeneity of many city block and graph feature values which, unlike pixel values in $\in[0, 255]$, are not constrained by a given range. Moreover, our hyperparameter search indicates performance is even harder to be improved by increasing codebook size and feature dimensions.

Latent Vector Clustering. Given the trained BVAE encoder, one straightforward methodology is to use its encoded latent vectors as building layout features Q for our subsequent GMAE training. We tried the simple approach of directly using the latent veectors but GMAE training convergence was not possible. Instead, we used K-means clustering to group latent space vectors into a set of classes. Thus, each city block will essentially be classified into one of a set of categorical labels. We tested using from 5 to 1000 clusters. However, this approach brought unacceptable reconstruction loss likely due to the heterogeneity of city blocks. We list the best performance of both K-means quantization, and trainable vector quantization in the Tab.2 (Quantization Table) in the main paper. Both are not comparable to the dimensional quantization as described in the main paper Sec. 3.1.

Dimensional Quantization. The key idea for our dimensional quantization is to use a small codebook (e.g. $L = 20$) for each dimension of the latent space vector. Specifically, the BVAE Encoder outputs a 512-d vector of μ and another 512-d σ for variational sampling of the network bottleneck. We find that the basic styles (e.g., building number, positions) of building layouts decoded by trained BVAE Decoder is generally deterministic to μ, while σ provides reasonable randomness. Thus we directly sample from the full distribution of σ from training dataset, and only consider the prediction of the μ (i.e., the 512 dimensions of q_i). A similar trade-off is also introduced in diffusion model training (e.g., [8]).

Our dimensional quantization is driven by the observation that the distribution in each dimension of μ is nearly normal. Moreover, the trained BVAE Decoder is robust to values near the normal distribution's mean. This enables quantization of each dimension with good stability and a low reconstruction loss (e.g., a slight change of values does not affect the decoded building layouts significantly). We choose $L = 20$ based on the ablations in Tab. 2 (Quantization Table) in the main paper.

2.3 Additional Real-World Units Evaluation

We evaluate the 2.5D building geometry as per-block percentage of "GEOM-E", and "Pos-E" in the Tab. 2 of the main paper. Here we provide a conversion to real-world units for better understanding. The row "BVAE*" in Tab.2 (Encode

Model Table) writes 1.00% Pos-E, and 0.71% Geom-E. This corresponds to each reconstructed building having an average height error of $0.10m$, size error of $16.97m^2$, and position error of $4.07m$. The row "DIM-Q* ($L = 20$)" in Tab. 2 writes 3.50% Pos-E, and 0.89% Geom-E. This corresponds to each single building having a height error of $0.27m$, size error of $27.52m^2$, and position error of $8.07m$.

The increasing errors of position and 2.5D geometry are mainly due to the numerical quantization in each μ dimension and the randomly sampled σ, as described in above sections.

3 City-Scale Controllable Generation

In Fig. 1 and its zoom-in in Fig. 2, we present a city-scale generation given only road networks and 10 representative city blocks as small controlling priors (i.e., $< 1\%$). Our method is able to generate realistic 2.5D urban layouts from road networks with plausible context harmonization from community-scale to city-scale.

Moreover, in Fig. 3, we provide cropped generated examples for another 16 major cities across the U.S. It proves the robustness and generalization of our methods among heterogeneous city context styles.

Comments on alternative approaches. For all the comparisons in Sec. 4.2 of the main paper, we generate the result of each approach without post-processing or human-in-the-loop refinement. However, for some evaluation metrics, the metric value depends on the particular instance that was generated. Our method generates its output using our priority-based scheduling in T iterations, with $T = 12$ being a reasonable number. Hence, for fairness we also select the best results from T independent generations for each alternative method.

Further, we tested other pixel-based approaches (e.g. [10]) apart from SDXL [12] (see Tab.1 and Fig.5 in the main paper). But none of them provide comparable results. Poor generation quality by InfinityGAN [10] is also observed by [15].

4 Semantic Manipulation

Using the trained GMAE, semantic manipulation can be implemented by adding *super nodes* with desired building layout styles to the original city graph. The added super nodes have a dense connectivity to neighboring communities and will influence the generation pattern of GMAE to produce a fused style. The semantic manipulation extent can be tuned by modifying connection degrees or edge distances attributes of the added super nodes. The users may customize their manipulations for broad "what-if" scenarios for urban planning, meteorology simulation, and game design applications.

In Fig. 4, we manipulate generated New Orleans with either dense urban priors from New York City or sparse urban priors from Norfolk. Our GMAE generates "fused" urban layouts guided by given priors. Meanwhile, the realism and diversity are kept.

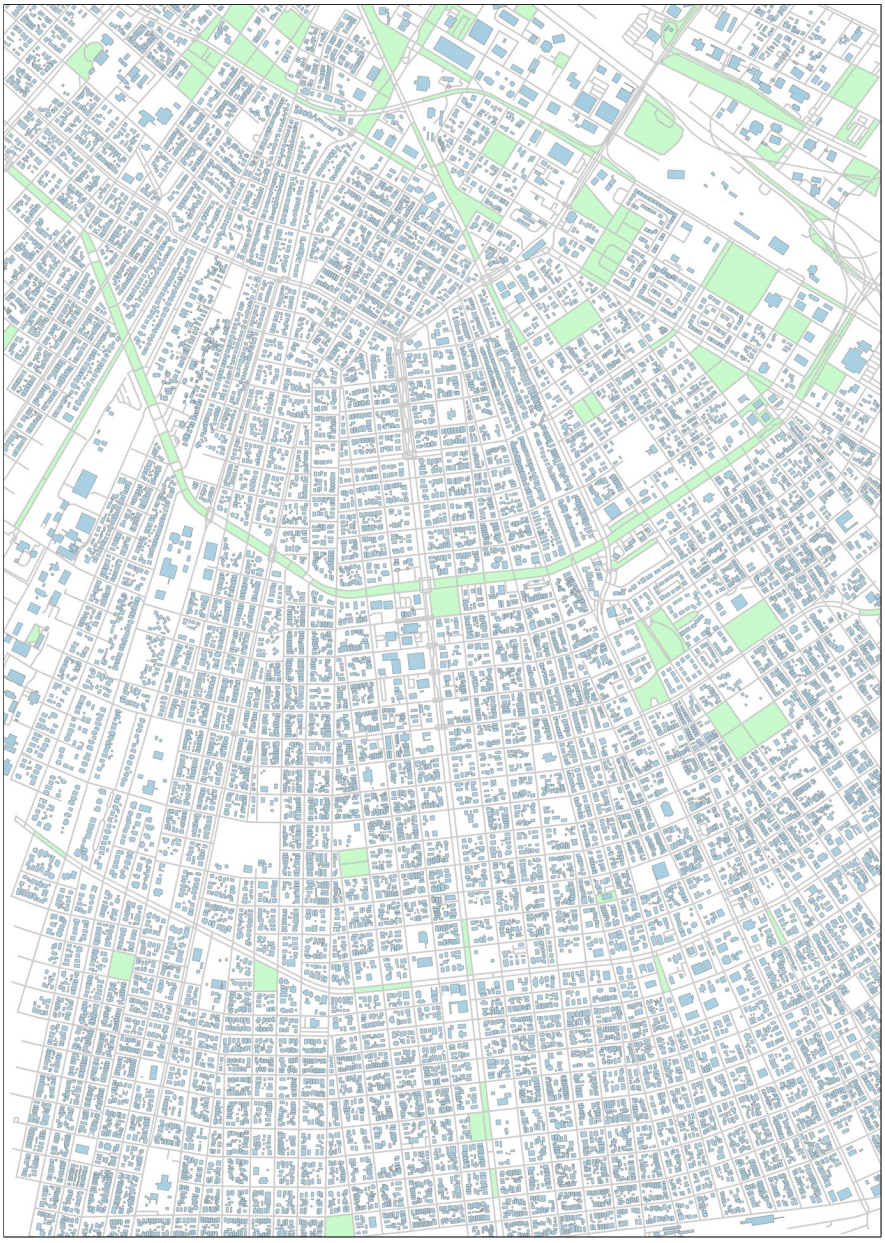

Fig. 1: City-Scale Map. We use the road networks from New Orleans to provide a city-scale generation using our method.

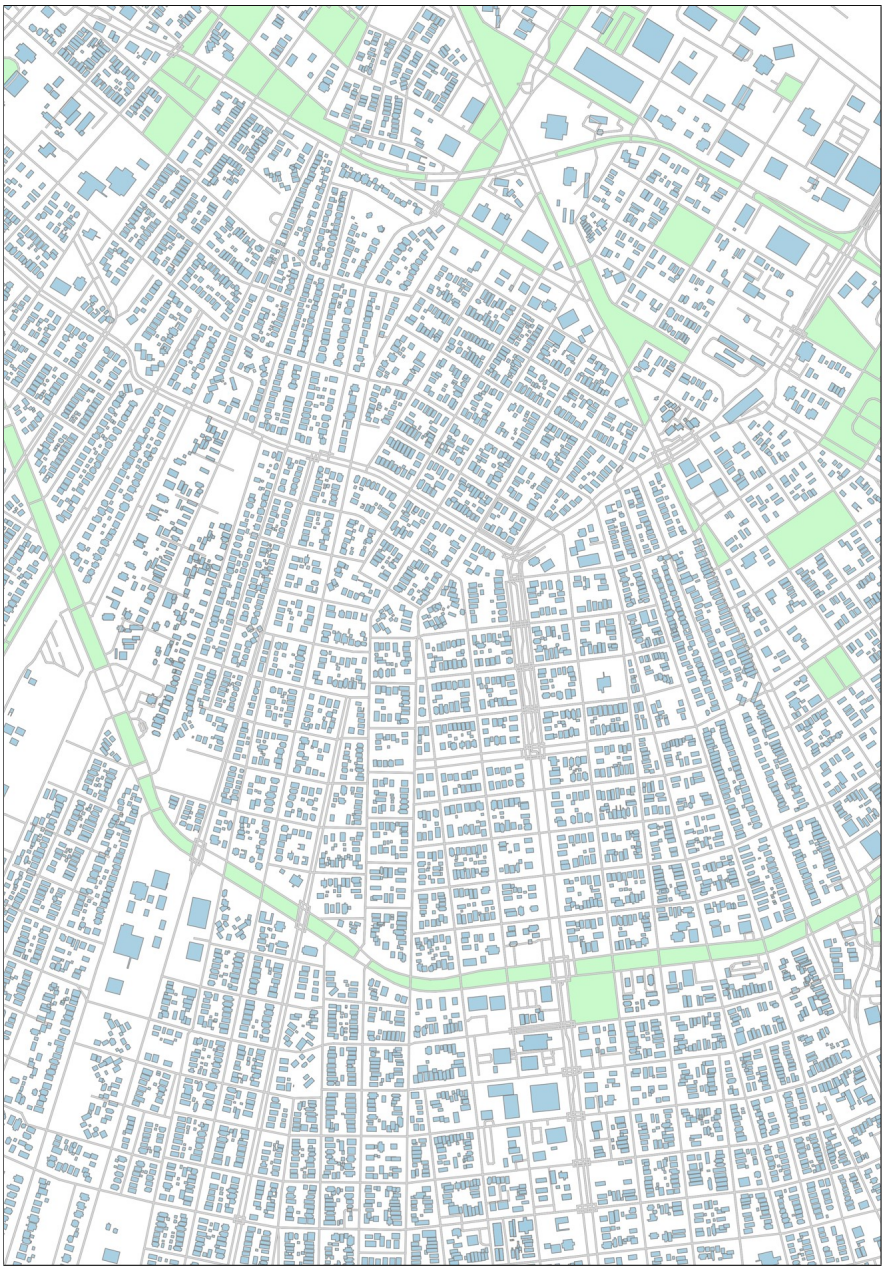

Fig. 2: Zoom-in Map. Zoom-in map of the upper right corner in Fig. 1.

Fig. 3: Example Generations across the U.S. Given arbitrary road networks across 16 diverse cities, our method generates realistic urban layouts with plausible context harmonization.

5 GMAE-based Socio-economic Metric Prediction

We find that the 2.5D geometry of building layouts in a given city block correlates well with social, economic, and climate metrics of the same block. In particular, the encoder of our trained GMAE may act as a feature extractor for a binary classification of city blocks as advantaged or disadvantaged groups. Hence, we can also use the GMAE for socio-economic metric estimation without requiring tedious customized surveys, and potentially leading to broader social applications for policy making. This prediction is analogous to the "linear probe"

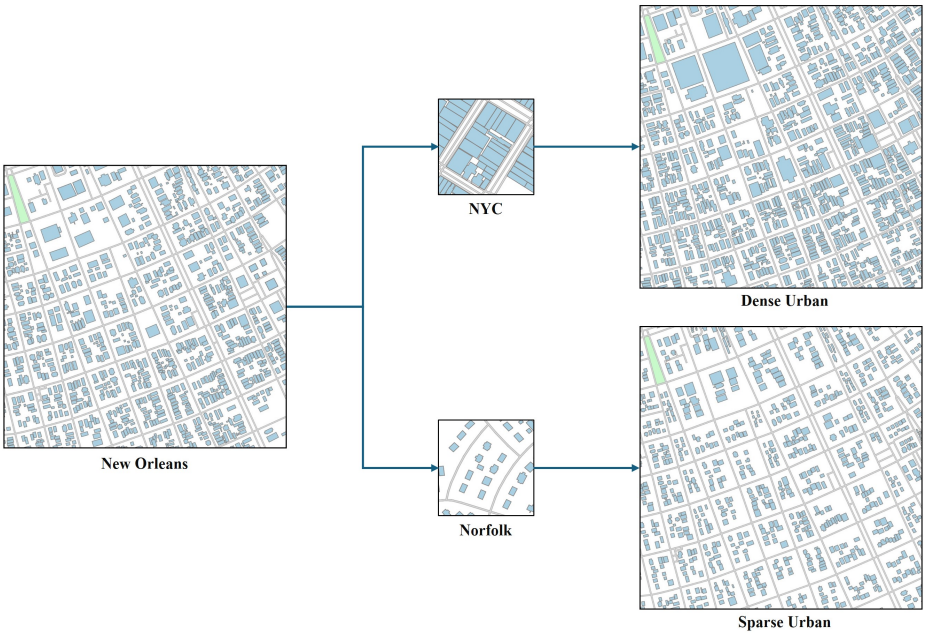

Fig. 4: City-scale Semantic Manipulation. We use *super nodes* to influence the generation of a New Orleans layout by using either dense building layout style from New York City or sparse urban style from Norfolk. See text for details.

in the representation learning community (e.g. CLIP [13]). The image/text features extracted by a pretrained encoder (weights frozen) are directly fed into a trainable linear MLP adapting downstream tasks (e.g. classification).

We make use of the social-economic metric dataset from the Climate and Economic Justice Screening Tool (CEJST) [1]. This dataset provides census-tract-level data for more than 100 metrics. We calculate the values of those metrics for each city block, and label the city blocks in the lower 1st percentile as the disadvantaged group, and those in the higher 99th percentile as the advantaged group.

In this study, we utilize the extracted node features F (see Fig.3 in main paper) to train a machine learning classifier. The classifier is trained to distinguish the aforementioned advantaged and disadvantaged groups. The performance of this binary classification, using SVM and XGBoost, is showed in Tab. 1. We report the diverse set of 11 metrics with higher than 75% accuracy.

Table 1: GMAE-based Prediction. Our GMAE combined with a conventional classifier (e.g. SVM or XGBoost) can be used to predict whether a set of city blocks corresponds to an advantaged and disadvantaged economic/social/environmental group.

Abbrv.	Metric Full Name	Best Acc. %
DSF_PFS	Diesel particulate matter exposure (percentile)	**89.76**
EBF_PFS	Energy burden (percentile)	**84.26**
LMI_PFS	Low median household income as a percent of area median income (percentile)	**83.62**
LLEF_PFS	Low life expectancy (percentile)	**80.30**
EBLR_PFS	Expected building loss rate (Natural Hazards Risk Index) (percentile)	**80.13**
LPF_PFS	Percent pre-1960s housing (lead paint indicator) (percentile)	**79.83**
HBF_PFS	Housing burden (percent) (percentile)	**76.82**
EPLR_PFS	Expected population loss rate (Natural Hazards Risk Index) (percentile)	**75.82**
P100_PFS	Percent of individuals < 100% Federal Poverty Line (percentile)	**75.46**
FLD_PFS	Share of properties at risk of flood in 30 years (percentile)	**75.46**
TF_PFS	Traffic proximity and volume (percentile)	**75.15**

6 Running time and model parameters

Under the same settings as main paper, below we report model parameter size and average inference time per city block by a single RTX A5000. The only exception is for SDXL [12] where the time is for generating one 1024×1024 resolution image by 20 sampling steps (typically covering dozens of blocks). Our method uses 12-iteration sampling. VTN [3] and SDXL (even if normalized to single block inference time) are quite slow. LayoutDM [9] (sampled by default 100 steps) is slower than our method, and GlobalMapper is slightly faster. However, our method outperforms others (main paper Tab.1 and Fig.5). Our model parameter size is much smaller than SDXL, and is only marginally larger than the largest of others.

Method	Inference Time↓ (ms)	Parameters
⋆SDXL [12]	5938.85	2.6B
VTN [3]	3428.32	45.49M
LayoutDM [9]	20.24	12.41M
GlobalMapper [6]	5.62	76.48M
Ours	7.94	89.87M

References

1. Climate and economic justice screening tool. `https://screeningtool.geoplatform.gov/`
2. List of united states cities by population. `https://en.wikipedia.org/wiki/List_of_United_States_cities_by_population`

3. 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)
4. Bureau, U.S.C.: Topologically integrated geographic encoding and referencing. https://www.census.gov/geographies/mapping-files/time-series/geo/tiger-line-file.html
5. 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)
6. 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)
7. 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)
8. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. Advances in neural information processing systems **33**, 6840–6851 (2020)
9. 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)
10. 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)
11. OpenStreetMap contributors: Planet dump retrieved from https://planet.osm.org . https://www.openstreetmap.org (2017)
12. 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)
13. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. In: International conference on machine learning. pp. 8748–8763. PMLR (2021)
14. Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y., et al.: Graph attention networks. stat **1050**(20), 10–48550 (2017)
15. Xie, H., Chen, Z., Hong, F., Liu, Z.: Citydreamer: Compositional generative model of unbounded 3d cities. arXiv preprint arXiv:2309.00610 (2023)