exploring the Latent space distribution of a graph autoencoder trained on 3d architecture models

**Abstract.** Building on previous research in generative graph machine learning in architecture, this paper investigates how data generation and preparation can change the distribution of a model’s latent space and thus its generative qualities. Therefore, we first present and discuss our previous approach of applying generative graph machine learning in architecture by sampling the latent space of a graph autoencoder trained with the augmentations of four examples of modernist buildings. We then present a new method of data generation for modernist buildings in the style of architect Mies van der Rohe, which produces a large range of 3D building models with great geometric variety. Trained on the new dataset, the graph autoencoder shows a more continuous latent space, confirmed by visual comparison and by three spatial analysis algorithms that quantitatively assess the spatial structure of the different latent spaces.

**Keywords.**  Latent space, latent walk, generative 3D architecture, generative graph machine learning, graph-based architecture, human-computer interaction, graph autoencoder

1. Introduction

Here we present an extended and improved version of the graph-based generation method of 3d architectural models presented originally in Bauscher et al. (2024). The original paper presents a generative AI system that outputs new spatial configurations of architectural elements in three dimensions. The graph-based autoencoder can read 3d buildings as training input and does not rely on any sort of conversion between 2d images or plans and a desired three-dimensional output in post. It presents a very intuitive approach to generate 3d models for architects, which although only simple, already shows some understanding of important architectural features that are used in generation. That makes it very well rooted in the modern idea of the architectural design process. In the following, we discuss and improve its method of data generation to further elevate performance and raison d’être.

## 1.1 STATE OF THE ART

Current research in generative AI in architecture can be divided into 2D generative models based on images or vector graphics (floor plans), and 3D generative models based on stacked images, point clouds, voxel- or mesh geometries. 2D image generation has successfully been used the longest, due to its great accessibility and generalizability towards architecture. In addition, it also offers an almost seamless integration into the contemporary workflow of architectural design, requiring a minimum of postprocessing before the result is fully usable (i.e. if anything, a quick visualisation of a design idea only needs to be edited in Photoshop to be presentable as the designer's idea). No special research for the architectural domain was necessary, but architects can use models like DALL-E (Ramesh et al., 2021) and Stable Diffusion (Rombach et al., 2022). Only some big architecture offices like Zaha Hadid Architects and Coop Himmelblau developed their own, in-house models to be able to further control the output.

3D generation models are rarer due to the difficulty of implementation and the need for custom solutions for architectural purposes. In addition, “(…) personal barriers often restrict their [the architects’] access to the latest technological developments, thereby causing the application of generative AI in architectural design to lag behind.” (Li et al., 2024). Another problem to solve is the complexity of any building and how to encode its information into any AI model as well as the lack of specific datasets to train on. Generic text-to-3d models (Lin et al., 2023; Poole et al., 2022) can be used, but they do not adapt as well in producing specific 3d architecture models as the 2d models do in generating images of buildings thanks to the lower complexity.

Most successful methods of encoding and generating 3D building information rely at least partially on 2d information. That could be floor plans, sections and depth information from which a 3d model is generated (Del Campo, 2022; Zhang & Blasetti, 2020). Other models rely on fully 3d based training, either encoding voxel data (Koh, 2022; Rasoulzadeh et al., 2024) or point clouds (Wei et al., 2023), which have improved in realism and logic recently but still tend to generate rather conceptually 3d models.

## 1.2 GRAPHS

Graph representations of buildings or parts of buildings have been used for a long time in architecture (Alexander, 1977), mainly due to their shared property of being non-discursive, meaning they cannot be fully described by words or rules (Hillier, 1996). In comparison to other data formats that can be used to train any machine learning model, graphs have the advantage of being able to store information about multiple types of data as well as information about relationships (Hamilton, 2020). This focuses the attention of the model on the logical and geometrical properties of buildings, rather than on the visual appearance.

1. Original Methodology

## DATASET

The custom dataset used previously consists of four modernist buildings that have been individually remodelled parametrically in Grasshopper inside Rhino. That gave the opportunity to easily augment each building in its geometry to create more data for training the autoencoder. All houses are from the same architectural period, modernism, for the ease of remodelling and a geometric continuity across all data. The buildings are: Mies van der Rohe's Barcelona Pavilion (1929), Ray and Charles Eames' Eames House (1949), Mies van der Rohe's Farnsworth House (1951)and Pierre Koenig's Stahl House (1960).

Those houses all follow the design principle of only using orthogonal walls, as well as mostly having full-height openings for windows. Yet they vary greatly in size, proportion and location.

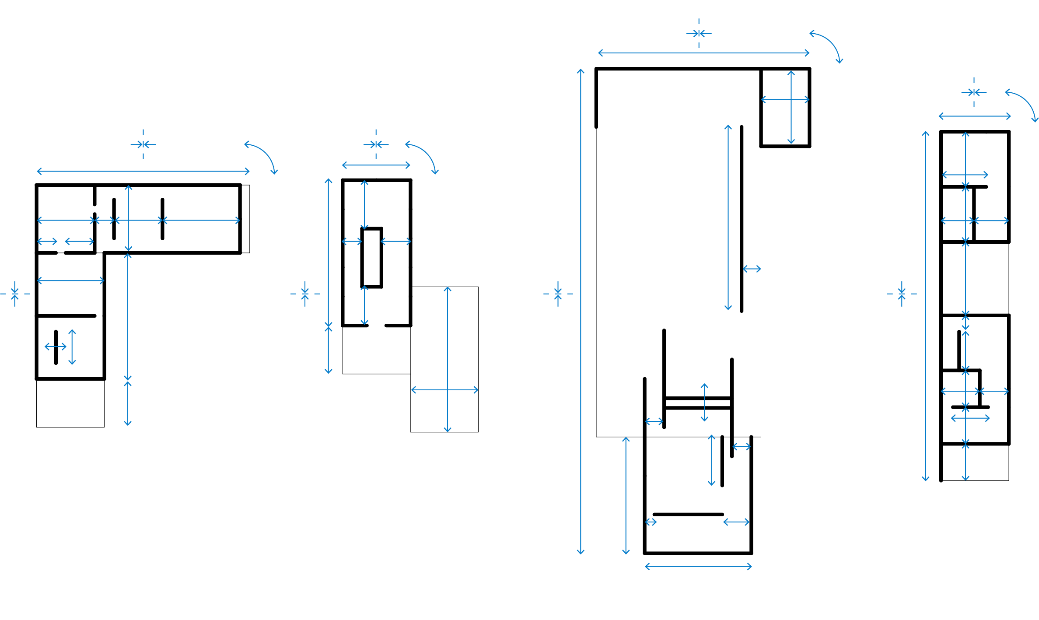
To avoid overcomplication in the 3d model, the buildings were remodelled as a surface model. Meaning walls, floors and ceilings are all seen as individual, rectangular surfaces without any thickness, while doors and openable windows are just empty spaces. This supports the idea of trying to model space-defining elements only, so there is no need for i.e. any structural considerations.

Figure 1 – The previous method of geometry augmentation in plan, blue highlights the parametric parameters

This approach fails in providing the autoencoder with a heterogeneous dataset mainly due to the method of augmentation. Starting always from the original model, elements are only slightly moved and scaled randomly in one direction (Fig. 1), creating not enough variation to allow for a smooth geometrical translation between the original buildings. The augmentations still always keep the same layout, element count, division between horizontal and vertical elements, overall size and ratio. Thus, also the graph structure does not change between augmentations.

In addition, the augmented geometries are randomly rotated around their central Z axis in increments of 90 degrees, as well as randomly mirrored in X and Y directions. On one hand, this creates a more diversified and interesting space for generation, on the other hand it introduces unnecessary noise into the latent space and thus makes the results harder to read and understand.

## 2.2 DATA REPRESENTATION

Figure 2 – Previously used graph representation of a surface-based 3D model

Each surface-based 3d building model in the dataset is translated into an undirected graph, where all geometrical information is held in the node features. Each node represents one surface, and nodes are connected in the graph if they physically touch in the 3d model (Fig. 2). To encode all necessary information, the node features contain the coordinates of the centre point, the orientation and the width and length of the surface. Here we must point out that this specific method of conversion only allows for orthogonally positioned elements. The orientation value equals one of the three global planes (XY, XZ, YZ) and thus limits the geometrical options for input and output. This method of representation proved to be in line with the geometrical idea of modernism and is also used in the here presented research.

## 2.3 GRAPH AUTOENCODER

Figure 3 – Previously used graph autoencoder model

The graph autoencoder model (Fig. 3) is defined by the encoder and decoder model. Here the encoder is graph-based as first described by Kipf and Welling (2016) consisting of message-passing layers (Hamilton et al., 2018), while the decoder consists of linear layers that do not create or use any graph structure. That means all input data must have the same count of elements because reconstructing the features through linear layers flattens the data into a one-dimensional vector with always the same length. Another limitation is the chosen dimension of the latent space - three - which restricts the performance of the model. This was chosen for easy visualisation considering the model itself might be the least deciding factor on the performance of the whole system, but data collection and preparation are far more important (Jarrahi et al., 2023).

1. Adapted Methodology

## 3.1 DATASET

Figure 4 –Plans Mies van der Rohe. Left: Farnsworth House, Centre: Barcelona Pavillon, Right: Brick Country House. Original plans simplified and redrawn by the author.

Aroyo et al. (2021) point out: “Data is potentially the most under-valued and de-glamorized aspect of today’s AI ecosystem” and “Benchmark datasets are often missing much of the natural ambiguity of the real world”. Both points can be applied to our previous research on generative graph autoencoders in architecture. The second point is harder to approach due to the limited number of three-dimensional datasets of buildings ready to use. The usage of a 2d-floor plan dataset as they are widely available (de las Heras et al., 2015; Kalervo et al., 2019; Wu et al., 2019) seems to not only defeat the conceptual idea of the research project but also still pose multiple questions on the quality and origin of the data.

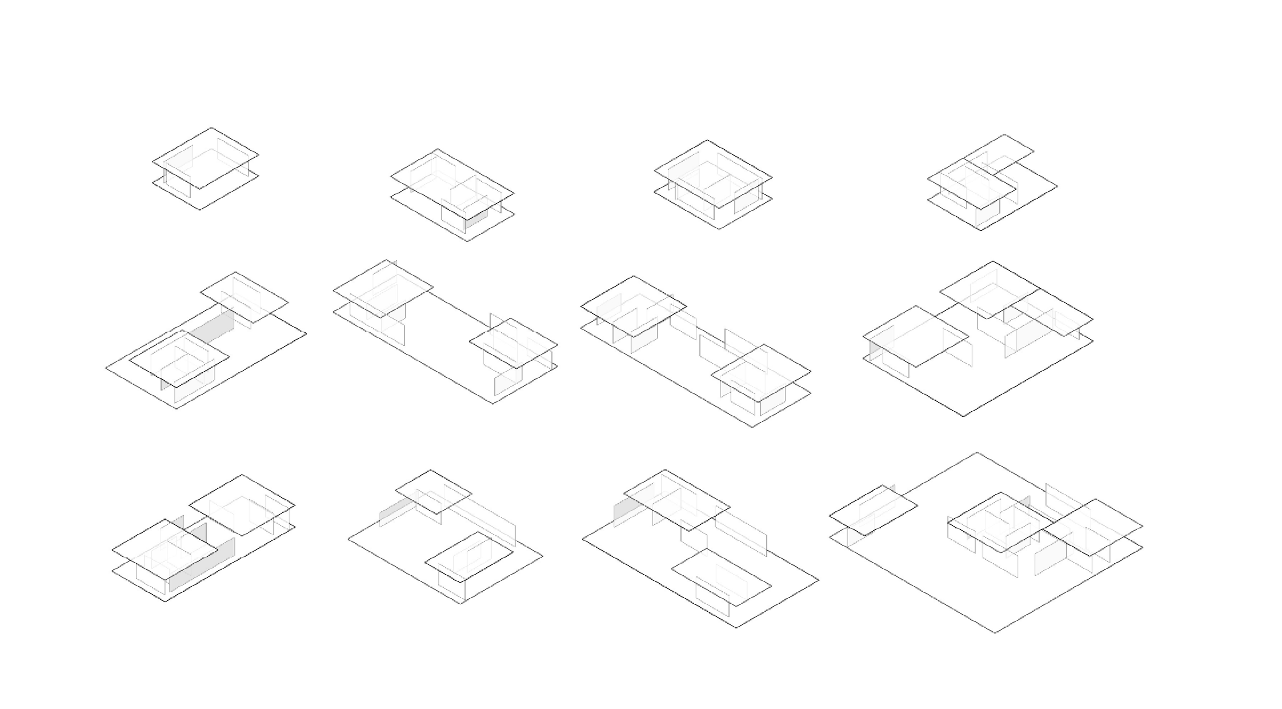
Therefore, we propose a further development of the existing dataset towards a more heterogeneous geometrical landscape of the dataset for better overall performance. Instead of using the original geometries of precedent buildings as a starting point for simple geometric augmentation, we now use an iterative algorithm that applies predefined geometric rules for generating the dataset. We use three buildings as concrete inspiration for deriving the mentioned rules, Mies van der Rohe’s Farnsworth House (1951), Barcelona Pavillon (1929), and the conceptional plans drawn around 1923 for the Brick Country House (Fig. 4). By choosing architectures by the same architect as guidelines, which differ in size, typology, proportion, and layout but follow similar geometrical design principles (open floor plan, grids, horizontal planes),

Figure 5 – Example of generated models in the style of Mies van der Rohe

we open the room for a more seamless transition between one another (Fig. 5).

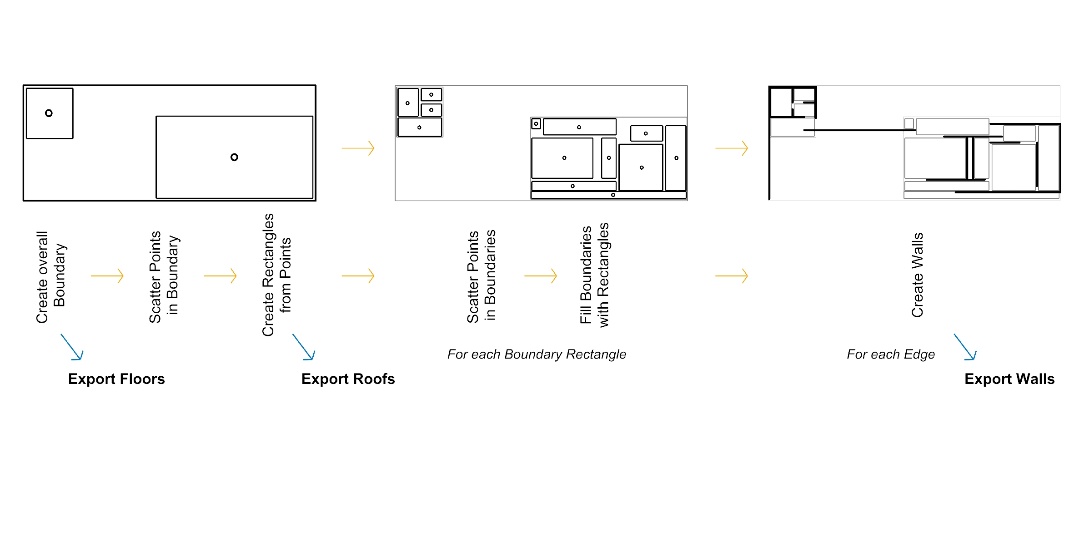
As a main driver for the definition of space in Mies van der Rohes geometries we identified the roof shapes. Rectangles are initially placed in the generative algorithm (Fig. 6) which is implemented in the Grasshopper/Rhino environment. The roof rectangles are then further subdivided into the subspaces over which they lay. For each edge of the subspace, a wall is placed in a random position with a random length. Finally, a floor geometry is generated by creating a boundary rectangle around the wall and roof projections.

Figure 6 – Schematic diagram of the algorithm to generate the dataset

## 3.2 GRAPH AUTOENCODER

Figure 7 - Graph Autoencoder Model with variable node count in training data

The graph autoencoder model (Fig. 7) stays mostly the same as described in paragraph 1.3. The main limitation before was due to the decoder architecture, the buildings in the dataset must all have the same number of surfaces, which is the only change to the model. We introduce masking to the decoder, which now decodes the latent vector to be a one-dimensional vector as big as the input data with the greatest number of elements. If fewer elements have been input, we use a mask to decode only as many elements as needed. This mask is predicted via a second branch in the decoder with three linear layers. The final model used the following variables:

- Epochs: 1000

- Learning Rate: 0.001

- Optimiser: Adam

- Loss function: Mean Squared Error

- Activation function: ReLu

- Train, Validate, Test: 80%, 10%, 10%

1. Results

We compare the newly trained and improved latent space with the previously used latent space and a reference set of points generated via the Sobol sequence, which serves as an optimal reference for well-distributed points (Fig. 8). The distributed points in both trained latent spaces represent the latent mapping of the dataset buildings by the respective trained autoencoder. The Sobol sequence is used for its quasi-random properties, ensuring points are evenly distributed across the space (Sobol, 1967), which makes it an ideal comparison for evaluating the distribution qualities of latent spaces.

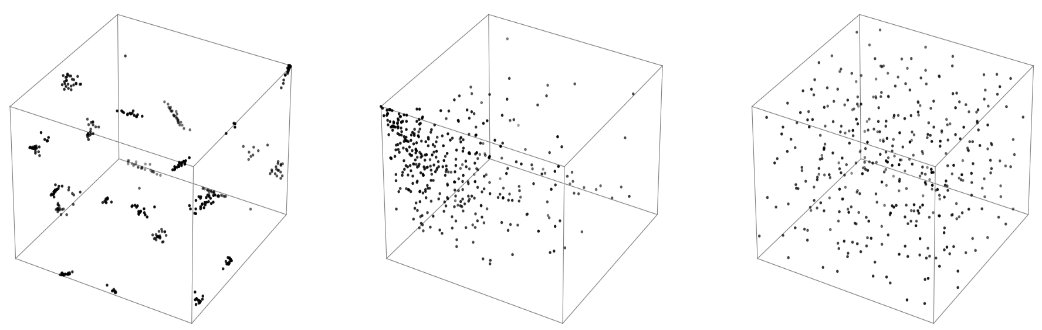
**Apart from a visual comparison (Fig. 8), we use three algorithms (Tab. 1) that quantitatively assess the spatial structure of our data, analysing the distribution, density, and clustering of points in a 13 cube: average nearest neighbor distance, DBSCAN clustering, and kernel density discrepancy. The first method, average nearest neighbor distance, is widely used in spatial analysis and measures the mean distance between each point and its nearest neighbor, providing insight into the overall distribution of points (Clark & Evans, 1954). DBSCAN clustering, a popular density-based clustering algorithm, identifies clusters by grouping points based on their density (Ester et al., 1996). Lastly, the kernel density discrepancy measures the variation in point density across a 3D grid and helps assess the evenness of point distribution in the latent space (Shepard, 1968).

Figure 8 – Visual comparison of latent spaces, left: previous research, centre: new research, Sobol Points

Table 1 – Quantitively comparison of latent spaces.

|  |  |  |  |
| --- | --- | --- | --- |
| **Dataset** | **Average Nearest Neighbor Distances** | **DBSCAN (Clusters, Noise, Points)** | **Kernel Density Discrepancy** |
| **Previous Latent Space** | 0.0161 | 23 Clusters, 7 Noise Points | 102.4063 |
| **New Latent Space** | 0.0553 | 3 Clusters, 112 Noise Points | 117.9843 |
| **Sobol Points** | 0.0960 | 1 Clusters, 395 Noise Points | 125.3987 |

The comparison of values across the three latent spaces aligns well with their visual structures. The new latent space, trained with an improved dataset, shows a noticeable improvement over the old latent space in kernel density discrepancy, nearest neighbor distances and amount of clustering, reflecting a more uniform distribution and proving the value of good data preparation. However, it still falls short when compared to the Sobol points, which, as expected, provide the best results in terms of even distribution and low clustering due to their quasi-random nature. Nevertheless, the new latent space performs significantly better than the old one, which had a higher clustering tendency and less uniform density, proving a promising improvement for future applications in sampling and interpolating building models from the latent space.

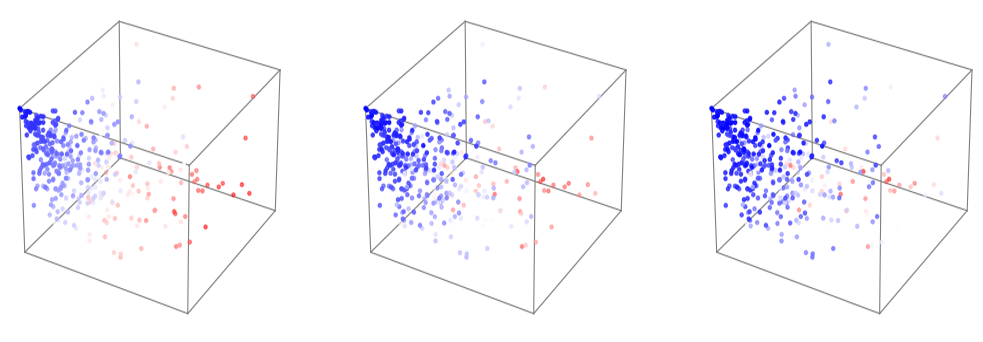
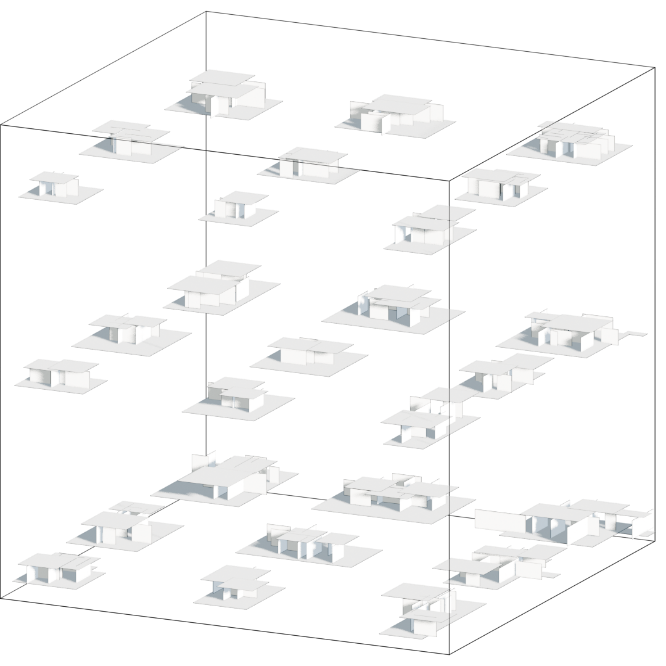
The new latent space also demonstrates a structured and meaningful distribution of the dataset, suggesting that the model has effectively learned a representation mainly based on geometric logic thanks to the new method of generation. When visualized with points coloured according to the total area and total count of surfaces in the training geometries (Fig. 9), a smooth transition from high to low values is shown across the latent space. Similarly, this can be observed when sampling a grid of unknown points in the latent space (Fig. 10), where the geometry in the bottom right corner has the highest surface count and -area, while the top left has the lowest. This continuity indicates that the model is organizing data points in a way that preserves meaningful geometric relationships, rather than scattering them randomly or basing the distribution on a poorly augmented and prepared dataset as before. This structured latent space provides a strong foundation for sampling, offering a higher chance of generating spatially and geometrically coherent 3d models when sampling new points in the space. In contrast to an unstructured or clustered space, this allows for more predictable and interpretable variations in the generated models, improving the overall reliability of the learned representation.

Figure 10 – Sampling of unknown points in the latent space

Figure 9 – Distribution of dataset in latent space, coloured according to total amount of elements (left) and total area (right)

1. Discussion

This research highlights the importance of data preparation, showing the successful implementation of a new augmentation method for 3d architecture datasets based on a limited number of existing architectural designs.

While the deterministic algorithm for generation already provides us with new geometries that follow our rules set to create “Miesian” 3d models, we argue that using a generative autoencoder gives us multiple advantages: (1) It can adapt to large or different datasets, while a deterministic model would need to be rewritten fit any other use case. (2) It offers a smooth interpolation between known designs, which can lead to unexpected new forms and spaces. The deterministic model is not able to create any interpolated geometry. (3) More parameters can be included in the future. The model can be trained to prefer optimized designs and the generative process does not solely need to be based on intuitive navigation of the latent space.

The next steps could include: (1) Further testing and optimization of the model to evaluate its new generative capabilities. (2) Redesigning the model to be a neurosymbolic AI system, combining a graph-based and a rule-based neural network for better performance. (3) A web-based implementation of the application for simpler visualization and higher accessibility of the research.

1. Conclusion

We successfully improved the latent space distribution of the graph autoencoder by implementing a semantic augmentation method for our dataset. We prove how the model successfully preserves meaningful spatial and geometrical organization in the latent space distribution by applying a robust, multi-faceted comparison of our datasets, beyond simple visualization. That leads the way for better results when sampling unknown points in the latent space. Due to the ongoing lack of usable, 3d architecture datasets for machine learning, this method shows a possible and valuable way of translating spatial knowledge from limited examples of real-world buildings into a usable dataset for generative machine learning.

1. References

Alexander, C. (1977). *A Pattern Language*. Oxford University Press.

Aroyo, L., Lease, M., Paritosh, P., & Schaekermann, M. (2021). *Data Excellence for AI: Why Should You Care* (No. arXiv:2111.10391). arXiv. https://doi.org/10.48550/arXiv.2111.10391

Bauscher, E., Dai, A., Elshani, D., & Wortmann, T. (2024, November 18). Learning and Generating Spatial Concepts of Modernist Architecture via Graph Machine Learning. *ResearchGate*. https://doi.org/10.52842/conf.caadria.2024.1.159

Clark, P. J., & Evans, F. C. (1954). Distance to Nearest Neighbor as a Measure of Spatial Relationships in Populations. *Ecology*, *35*(4), 445–453. https://doi.org/10.2307/1931034

Del Campo, M. (2022). Deep House—Datasets, estrangement, and the problem of the new. *Architectural Intelligence*, *1*(1), 12. https://doi.org/10.1007/s44223-022-00013-w

Ester, M., Kriegel, H.-P., Sander, J., & Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, 226–231.

Hamilton, W. L. (2020). *Graph Representation Learning*. Springer International Publishing. https://doi.org/10.1007/978-3-031-01588-5

Hamilton, W. L., Ying, R., & Leskovec, J. (2018). *Inductive Representation Learning on Large Graphs* (No. arXiv:1706.02216). arXiv. http://arxiv.org/abs/1706.02216

Hillier, B. (1996). *Space Is The Machine: A Configurational Theory Of Architecture*.

Jarrahi, M. H., Memariani, A., & Guha, S. (2023). The Principles of Data-Centric AI (DCAI). *Communications of the ACM*, *66*(8), 84–92. https://doi.org/10.1145/3571724

Kipf, T. N., & Welling, M. (2016). *Variational Graph Auto-Encoders* (No. arXiv:1611.07308). arXiv. https://doi.org/10.48550/arXiv.1611.07308

Koh, I. (2022). Voxel Synthesis for Architectural Design. In J. S. Gero (Ed.), *Design Computing and Cognition’20* (pp. 297–316). Springer International Publishing. https://doi.org/10.1007/978-3-030-90625-2\_17

Li, C., Zhang, T., Du, X., Zhang, Y., & Xie, H. (2024). *Generative AI Models for Different Steps in Architectural Design: A Literature Review* (No. arXiv:2404.01335). arXiv. https://doi.org/10.48550/arXiv.2404.01335

Lin, C.-H., Gao, J., Tang, L., Takikawa, T., Zeng, X., Huang, X., Kreis, K., Fidler, S., Liu, M.-Y., & Lin, T.-Y. (2023). Magic3D: High-Resolution Text-to-3D Content Creation. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 300–309. https://doi.org/10.1109/CVPR52729.2023.00037

Poole, B., Jain, A., Barron, J. T., & Mildenhall, B. (2022). *DreamFusion: Text-to-3D using 2D Diffusion* (No. arXiv:2209.14988). arXiv. https://doi.org/10.48550/arXiv.2209.14988

Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., & Sutskever, I. (2021). *Zero-Shot Text-to-Image Generation* (No. arXiv:2102.12092). arXiv. https://doi.org/10.48550/arXiv.2102.12092

Rasoulzadeh, S., Bank, M., Wimmer, M., Kovacic, I., Schinegger, K., & Rutzinger, S. (2024). *ArchComplete: Autoregressive 3D Architectural Design Generation with Hierarchical Diffusion-Based Upsampling* (No. arXiv:2412.17957). arXiv. https://doi.org/10.48550/arXiv.2412.17957

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., & Ommer, B. (2022). *High-Resolution Image Synthesis with Latent Diffusion Models* (No. arXiv:2112.10752). arXiv. https://doi.org/10.48550/arXiv.2112.10752

Shepard, D. (1968). A two-dimensional interpolation function for irregularly-spaced data | Proceedings of the 1968 23rd ACM national conference. *Proceedings of the 1968 23rd ACM National Conference*, 517–524. https://doi.org/10.1145/800186.810616

Sobol, I. M. (1967). On the distribution of points in a cube and the approximate evaluation of integrals. *USSR Computational Mathematics and Mathematical Physics*, *7*(4), 86–112. https://doi.org/10.1016/0041-5553(67)90144-9

Wei, Y., Vosselman, G., & Yang, M. Y. (2023, August 31). *BuilDiff: 3D Building Shape Generation using Single-Image Conditional Point Cloud Diffusion Models*. arXiv.Org. https://arxiv.org/abs/2309.00158v1

Zhang, H., & Blasetti, E. (2020). 3D Architectural Form Style Transfer through Machine Learning. *CAADRIA Proceedings*, 659–668. https://doi.org/10.52842/conf.caadria.2020.2.659