LEARNING AND GENERATING SPATIAL CONCEPTS OF MODERNIST ARCHITECTURE VIA GRAPH MACHINE LEARNING

ERIK BAUSCHER[1], ANNI DAI[2], DIELLZA ELSHANI[3], THOMAS WORTMANN[4]
[1,2,3,4]Chair for Computing in Architecture, Institute for Computational Design and Construction, University of Stuttgart
[1]erikbauscher5@gmail.com, 0009-0007-6980-0873
[2,3,4]{anni.dai|diellza.elshani|thomas.wortmann}@icd.uni-stuttgart.com, 0009-0002-8521-5045|0000-0003-2902-341X|0000-0002-5604-1624

Abstract. This project showcases a use case away from most other research in the field of generative AI in architecture. We present a workflow to generate new, three-dimensional spatial configurations of buildings by sampling the latent space of a graph auto-encoder. Graph representations of three-dimensional buildings can store more data and hence reduce the loss of information from building to machine learning model compared to image- and voxel-based representations. Graphs do not only represent information about elements (nodes/pixels/etc.) but also the relationships between elements (edges). This is specifically helpful in architecture where we define space as an assemblage of physical elements which are all somehow connected (i.e., wall touches floor). Our method generates valuable, logical and original geometries that represent the architectural style chosen in the training data. These geometries are highly different from any image-based generation process and justify the importance of graph-based 3D geometry generation of architecture via machine learning. The method also introduces a novel conversion process from architecture to graph, an adapted decoder architecture, and a physical prototype to control the generation process, all making generative machine learning more applicable to a real-world scenario of designing a building.

Keywords. generative 3D architecture, generative graph machine learning, graph-based architecture, human-computer interaction, graph autoencoder, latentwalk

1. Introduction

The main player, image-based generative AI systems lack seriousness and logic when applied in a real-world design scenario, mainly due to their focus on the visual. Image Generation tools like Dall-E, Midjourney and Stable Diffusion produce images from text prompts. One or multiple generated images are used to extract information about architectural concepts, geometry and materials to inform building designs. Here the image as the medium cannot offer logical or coherent information due to its pixel-based nature, and human architects need to perform a manual conversion into a working building design. This makes AI image generation great as an inspiration in the very early stage of design, but hard to apply to the following design process.

In addition, photographs of buildings, floor plans, sections, etc. contain only compressed and partial information about building designs. In the world of computers this step of compressing is redundant (Carpo, 2017) and rather unhelpful for neural networks (NNs) to understand complex content.

This work wants to suggest a different approach, showing the benefits of graphs and three-dimensional representations of buildings for generating new architecture. Graphs can contain more information from different domains, because they are not grid-based, and do not only represent information about elements (nodes, pixels, etc.) but also the relationships between nodes (Hamilton, 2020).

Here, rather than focusing on visual information, the definition and generation of space are in the foreground, intending to give the architects a tool they can use hand in hand with their current design workflow. Compared with existing methods for architectural AI generation, the presented method generates three-dimensional spatial arrangements that are more complex than those allowed by voxel grids.

1.1. SUBSYMBOLIC AI

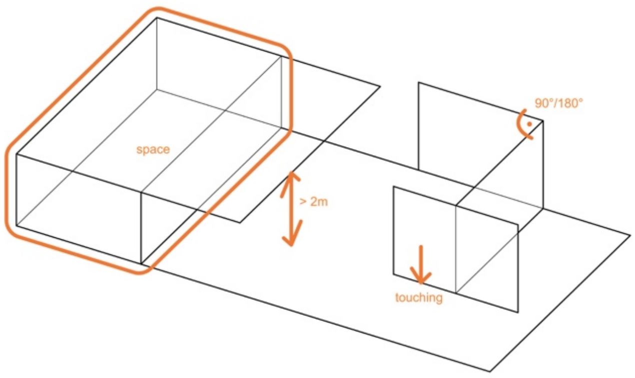

Figure 1: Subsymbolic AI System finds high-level features in 3D buildings

Symbolic AI focuses on teaching an AI system the rules of the game of chess. Subsymbolic AI only lets the NN play chess making the model learn the rules by its own (Mitchell, 2020).

By using unlabelled data for training a generative AI system in architecture, the second described approach is applied. We try to make a deep neural network understand very complex, three-dimensional buildings, without giving it i.e., information about gravity (Fig. 1). Without this information, the NN must figure out that walls are always touching horizontal elements by itself, and then use this knowledge when generating. As buildings are extensively more complex than images, this task of understanding the so-called high-level features becomes more difficult for the NN. In the following, we try to lower the complexity as much as possible while still maintaining the original idea of teaching an NN to generate three-dimensional buildings from scratch.

1.2. STATE OF THE ART

While there already is architectural research about graphs and graph neural networks in architecture (Alymani et al., 2022), it mostly focuses on classification tasks rather than generative ones. Alymani et al. use the dual graph representation for classification tasks with graph machine learning and already show promising results. The trend can also be seen in other research on generative architecture lately. Zhong et al. (2023) use graph representation learning in combination with a recursive neural network to generate 3D massing models of architecture, while other research uses layout graphs to generate 2D floorplans (Hu et al., 2020). In both cases, the results are far from perfect and not completely generalizable. Yet being based on graph representations, the models seem to be more capable of generating logical architectures than models built upon pixel- or voxel data (del Campo, 2022; Koh, 2020), where although the generated architectures are 3D and partially look stunning, they seem to still need manual or parametric postprocessing to represent a real-world building design.

1.3. GRAPHS

Buildings and graphs are deeply related because they both have the property of being non-discursive, meaning that they cannot be fully described by words or rules, but rather by their patterns and relations (Hillier, 1996). Early research on converting architectural ideas to graphs was already done in the 1970s inter alia by Christopher Alexander (1977) in his book "A Pattern Language", where he provides rules, patterns and grammar to any possible situation in buildings. There are different types of graphs, including undirected (topology graphs) and directed graphs. In the last two decades, directed graphs using Semantic Web Standards have been visible in the building industry often in combination with Building Information Modelling standards, focusing mainly on ontologies (McGlinn & Pauwels, 2022). Semantic Web Standards can include languages such as the Web Ontology Language, which is used for enriching the data with semantics in the form of ontologies. Ontologies represent knowledge about things and the relations between them (Elshani et al., 2023). Work on directed graphs using Semantic Web Standards in the building industry includes the translation of the IFC schema to an OWL language, ifcOWL (Beetz et al., 2009), BOT ontology (Rasmussen et al., 2020), the Building and Habitats object Model Ontology and workflow (Elshani et al., 2022), etc.

Due to the construction of a custom dataset, in this project, the graph structure was simplified as much as possible resulting in an undirected graph holding mostly geometrical information in the node features and no edge features. In the future, the same method could be applied to much more detailed graphs as introduced here, resulting in more detailed, generated building designs.

2. Methodology

2.1. DATASET

Due to the lack of accessible 3D building datasets, a custom dataset needed to be constructed. One possibility would be to use one of many 2D floor plan datasets available (de las Heras et al., 2015; Kalervo et al., 2019; Wu et al., 2019) and extend the plans parametrically in the third dimension. The main advantage here is the great number of plans available to use for training the AI model. Conceptually however, it is hard to argue for the use of mostly unspecified floor plans which are only loosely connected to any architectural idea and style. In addition, a 2D to 3D conversion might defy any argument for a 3D-based AI model, when even the training data originally was 2D. The decision was taken to construct a simple dataset from scratch, where the biggest hurdle was to achieve a certain number of training data. Four well-known buildings from the modernist era were chosen to be remodelled and augmented each a hundred times. The augmentation was done parametrically by varying the position, rotation and dimensions of the individual building elements (wall, floor, ceiling) to a degree where the original spatial concept of the building would not be lost. The final dataset consists of 400 individual 3D buildings to train and test the model on. The four original houses are:

- Mies van der Rohe's Barcelona Pavilion (1929)
- Ray and Charles Eames' Eames House (1949)
- Mies van der Rohe's Farnsworth House (1951)
- Pierre Koenig's Stahl House (1960)

Modernist buildings were chosen for their simplicity in the definition of space and their good documentation. In modernism, structural elements were separated from space-defining ones, giving the architects more freedom in planning. This paired with the consistent use of right angles and geometrically simple volumes make the modernist style in architecture at least geometrically easy to understand, simplified and remodelled.

2.2. CONVERSION FROM BUILDING TO GRAPH

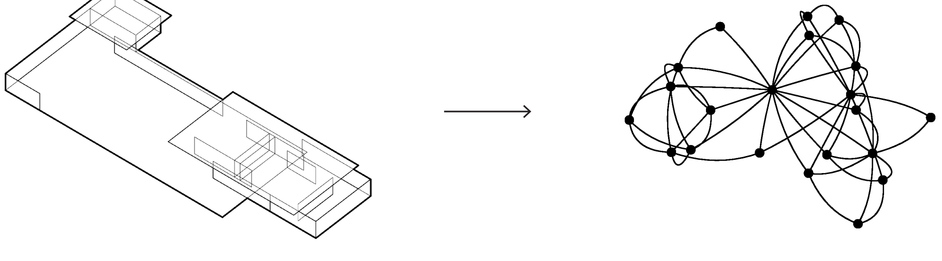

Figure 2: Conversion of the Barcelona Pavilion (M.v.d. Rohe) to a graph

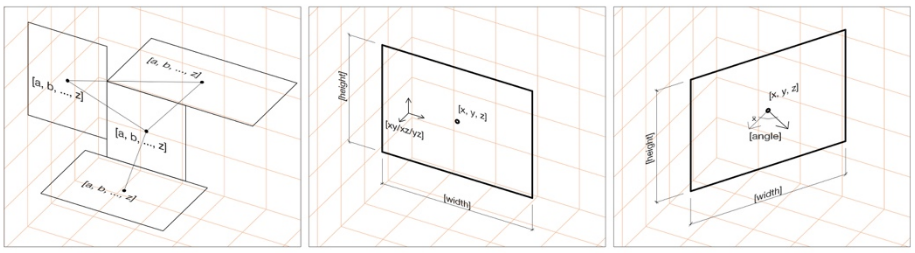

Figure 3: Left: Definition of a simple graph; Centre: Representation (A); Right: Representation (B)

To make a 3D building machine-readable, it is converted into a graph. Preparing the data as a graph highly influences the output of the generative model, as the conversion is the first and the last step performed in training and generation. This process is done parametrically and sets the geometric limitations for the output.

Each node in the graph contains geometric and material information about its respective physical element in the 3D model. Here, the advantage is made from the modernist buildings, which mostly allow being remodelled as 2D surfaces arranged in space. To keep the complexity low, the depth of the elements is not considered. The material of each element is described as either solid or transparent. Doors are not modelled but the space is left empty. Each element is described as a flat rectangle as visible in Figure 3. If the elements touch along any edge, they are linked in the graph by an edge (Fig. 2). The edges do not contain any features but are only used for message passing in the encoder. For the features of the nodes (elements), two versions were used to produce different results (Fig. 3), where (A) will be the default representation for the explanation of the model in the rest of the paper. Representation (A) represents one rectangular element by the X-, Y-, and Z- coordinates of its centre, the width and length of the rectangle and its orientation in space as a choice between the three world planes in the coordinate system. Representation (B) also includes the centre, width and height but uses the rotation of the surface's normal vector around World Z as information about orientation in space. (A) only allows three orientations of each element thus producing very static results. (B) gives the model more freedom but also enables more outcomes of the original style of the dataset. A representation with full freedom was also tried only using information from the four corner points, which resulted in too much sloping and uninterpretable results.

2.3. AUTOENCODER MODEL

A graph autoencoder as described first by Kipf and Welling (2016) was used to learn the features of the input architecture. The training data is unlabelled, thus while training the model is optimized only for matching the newly generated graph with the input graph. While the encoder consists of standard message passing layers (Hamilton et al., 2018), the decoder does not make use of the graph structure but decodes the latent vector through linear layers. A graph-based message-passing decoder was not chosen because of the difficulty of reconstructing a full graph from just the flattened one-dimensional latent vector (Guo & Zhao, 2022).

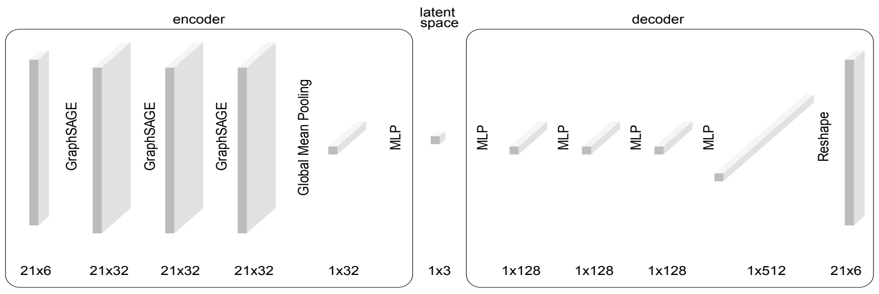

Figure 4: Definition of the used Graph Autoencoder Model

Using flattened data in the decoder brings the following limitation for the model: All graphs (hence all different buildings in the training data) need to have the same number of nodes. If just message passing was used, only the number of node features would need to be the same, but here due to the decoder architecture, it is the case for both the number of node features and number of nodes. Of course, this highly limits the range in which the model can generate new architecture as the buildings always will have the same number of elements. This loss was accepted due to the gained simplicity in training the model and using it for generation. The model was implemented using the Python machine learning framework PyTorch (Paszke et al., 2019) and its library PyTorch Geometric (Fey & Lenssen, 2019), which is specially written for graph machine learning. 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%

2.4. GENERATION

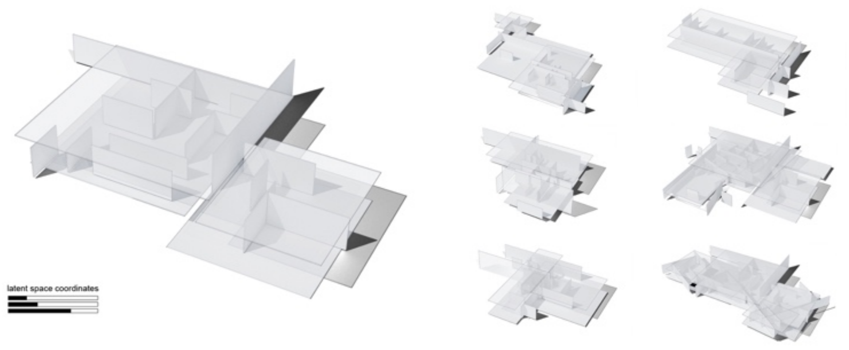

Figure 5: Newly generated, three-dimensional buildings

We generate new buildings by sampling the latent space of the trained autoencoder. When sampling unknown data points of the latent space, we expect the model to apply its learned logic and generate a new design that combines, reinterprets and varies features of buildings from known data points in the same area.

3. Results

In Figure 5 you can see the resulting generated geometry which was sampled in one session with the goal of finding new, never seen and buildable 3D arrangements. In the following, three geometries will be further described to make the model as well as its results more understandable.

3.1. EXAMPLES

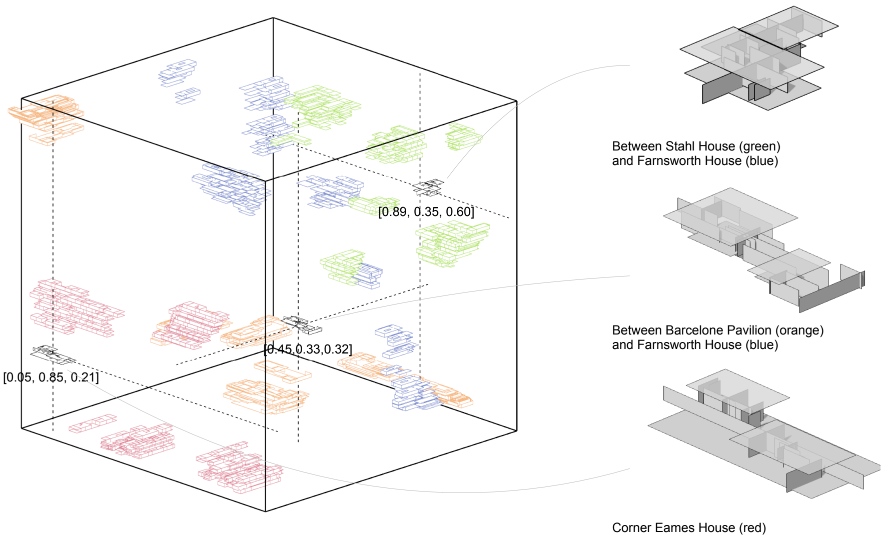
[0.89, 0.35, 0.60]
[0.45,0.33,0.32]
[0.05, 0.85, 0.21]
Between Stahl House (green) and Farnsworth House (blue)
Between Barcelone Pavilion (orange) and Farnsworth House (blue)
Corner Eames House (red)

Figure 6: Left: Latent Space of the trained model; Right: Sampled new buildings

On the left-hand side, Figure 6 shows the 3D latent space of the trained model. All 400 buildings are mapped to their respective location in the latent space, showing that the model organised them in smaller groups mainly influenced by their affiliation to one of the original four modernist houses (colour) and their general orientation.

3.1.1. Between Stahl and Farnsworth House (Fig. 6, top)

While the slab alignments and sizes are still in the style of the Farnsworth House, the vertical elements are already arranged in the "L"- pattern as known from the Stahl House. The confused arrangement of the vertical elements results from the principle of "geometric travel" between known data points in the latent space. As usual in latent walks, when travelling between known geometry, the Autoencoder needs to negotiate the known geometries around the unknown point and consider any logic and rules learned when training. This sometimes results in the displayed semi-logical layouts of the building.

3.1.2. Between Barcelona Pavilion and Farnsworth House (Fig. 6, centre)

Here the model negotiates the concept of directionality from the Barcelona Pavilion with the unidirectional floor plan of the Farnsworth House. In general, the Autoencoder finds specific arrangements of groups of elements in the training data and tries to reproduce this arrangement in the bigger context of the full, newly generated building.

3.1.3. Corner Eames House (Fig. 6, bottom)

The sampled data point here is in proximity to where the model mapped most of the augmented Eames Houses in the latent space. The model discovered the two separated living areas along the long wall. In comparison to the original, one area is mirrored along the wall, creating a new layout, while maintaining most of the other space elements one can find in the original.

3.2. APPLICATION

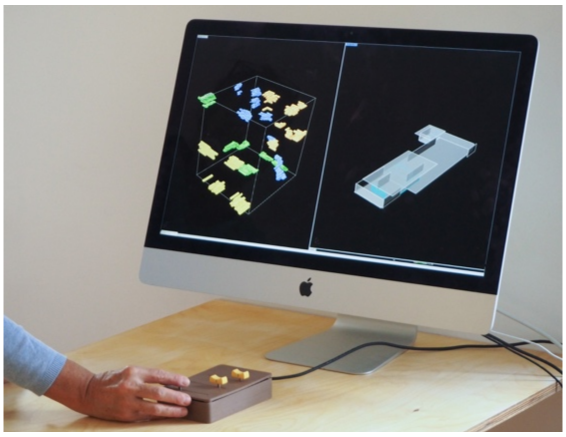

Figure 7: Photograph of the physical setup for using the model

To make the generative model approachable and usable for designers and architects, a physical prototype was built and connected with the Rhino/Grasshopper interface. The prototype consists of three sliders representing the three latent variables of the AI model. As shown in Figure 7, an architect can use those sliders to adjust the generated output in real-time. On the screen, one sees information about the location in the latent space as well as the generated 3D model in real-time.

This very intuitive way of generating new architecture makes the concept of the latent space very understandable even to non-experienced architects. It also emphasises the importance of the tool itself, which is needed to make research in architecture accessible to all designers and architects.

4. Discussion

Compared with pixel- and voxel-based generation, the presented method has the following advantages: (1) It works with three-dimensional geometries that do not need to follow any grid structure. (2) The graph representations contain more information that the model can learn from. (3) It presents a very intuitive, approachable and understandable generation tool for architects.

However, the current version suffers from the following disadvantages: (1) The qualities and availabilities of 3D datasets to use for training. (2) The low amount of already existing knowledge and research done in the field of graph-based, 3D generative AI systems in architecture. (3) The absence of an objective evaluation method for the generated results.

The presented method does not yet make use of any evaluation method other than the designer's eye. Since technically in training the only evaluation of the decoded latent space is its geometrical deviation from the originally inputted architecture, nothing else is needed for a working generation process. By using unlabelled data, we expect the model to learn the essential features of the training set by itself. Here the training set is defined as good architecture, and based on the idea of subsymbolic AI, the model can then generate new, good architecture, even from unknown data points in the latent space. Of course, this statement is insufficient, and one could achieve more control as well as better results by labelling the training data. These labels could include data from programmatic, environmental or structural simulations which can be produced with the already existing three-dimensional data. In a more industrial setting, it could also be values about costs, statistics on efficiency, used materials, etc.

To make the results more usable and realistic, further work can include labels for training and generating as well as extending the dataset with non-planar geometries. The model itself can also be redesigned to be a neurosymbolic AI system, combining a graph-based NN with a rule-based NN to enable an even deeper understanding of buildings (Sheth et al., 2023).

Acknowledgements

Supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany´s Excellence Strategy – EXC 2120/1 – 390831618.

References:

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

Alymani, A., Jabi, W., & Corcoran, P. (2022). Graph machine learning classification using architectural 3D topological models. SIMULATION: Transactions of The Society for Modeling and Simulation International, Online, 1–15. https://doi.org/10.1177/00375497221105894

Beetz, J., Leeuwen, J. van, & Vries, B. de. (2009). IfcOWL: A case of transforming EXPRESS schemas into ontologies. AI EDAM, 23(1), 89–101. https://doi.org/10.1017/S0890060409000122

Carpo, M. (2017, October 20). The Second Digital Turn. MIT Press. https://mitpress.mit.edu/9780262534024/the-second-digital-turn/

de las Heras, L.-P., Terrades, O. R., Robles, S., & Sánchez, G. (2015). CVC-FP and SGT: A new database for structural floor plan analysis and its groundtruthing tool. International Journal on Document Analysis and Recognition (IJDAR), 18(1), 15–30. https://doi.org/10.1007/s10032-014-0236-5

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

Elshani, D., Hernandez, D., Lombardi, A., Siriwardena, L., Schwinn, T., Fisher, A., Staab, S., Menges, A., & Wortmann, T. (2023). Building Information Validation and Reasoning Using Semantic Web Technologies. In M. Turrin, C. Andriotis, & A. Rafiee (Eds.), Computer-Aided Architectural Design. INTERCONNECTIONS: Co-computing Beyond Boundaries (pp. 470–484). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-37189-9_31

Elshani, D., Lombardi, A., Fisher, A., Hernandez, D., Staab, S., & Wortmann, T. (2022, May). Knowledge Graphs for Multidisciplinary Co-Design: Introducing RDF to BHoM. ESWC - LDAC 2022.

Fey, M., & Lenssen, J. E. (2019). Fast Graph Representation Learning with PyTorch Geometric (arXiv:1903.02428). arXiv. https://doi.org/10.48550/arXiv.1903.02428

Guo, X., & Zhao, L. (2022). A Systematic Survey on Deep Generative Models for Graph Generation (arXiv:2007.06686). arXiv. https://doi.org/10.48550/arXiv.2007.06686

Hamilton, W. L. (2020). Graph Representation Learning.

Hamilton, W. L., Ying, R., & Leskovec, J. (2018). Inductive Representation Learning on Large Graphs (arXiv:1706.02216). arXiv. https://doi.org/10.48550/arXiv.1706.02216

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

Hu, R., Huang, Z., Tang, Y., van Kaick, O., Zhang, H., & Huang, H. (2020). Graph2Plan: Learning Floorplan Generation from Layout Graphs. ACM Transactions on Graphics, 39(4). https://doi.org/10.1145/3386569.3392391

Kalervo, A., Ylioinas, J., Häikiö, M., Karhu, A., & Kannala, J. (2019). CubiCasa5K: A Dataset and an Improved Multi-Task Model for Floorplan Image Analysis (arXiv:1904.01920). arXiv. https://doi.org/10.48550/arXiv.1904.01920

Kipf, T. N., & Welling, M. (2016). Variational Graph Auto-Encoders (arXiv:1611.07308). arXiv. https://doi.org/10.48550/arXiv.1611.07308

Koh, I. (2020). Voxel Synthesis for Architectural Design.

McGlinn, K., & Pauwels, P. (Eds.). (2022). Buildings and Semantics: Data Models and Web Technologies for the Built Environment. CRC Press. https://doi.org/10.1201/9781003204381

Mitchell, M. (2020). Artificial intelligence—A guide for thinking humans. Penguin Books.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., … Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library (arXiv:1912.01703). arXiv. https://doi.org/10.48550/arXiv.1912.01703

Rasmussen, M. H., Lefrançois, M., Schneider, G., & Pauwels, P. (2020). BOT: The Building Topology Ontology of the W3C Linked Building Data Group. Semantic Web. https://doi.org/10.3233/SW-200385

Sheth, A., Roy, K., & Gaur, M. (2023). Neurosymbolic AI -- Why, What, and How (arXiv:2305.00813). arXiv. https://doi.org/10.48550/arXiv.2305.00813

Wu, W., Xiao-Ming, F., Tang, R., Wang, Y., Qi, Y.-H., & Liu, L. (2019). Data-driven interior plan generation for residential buildings. ACM Transactions on Graphics, 38, 234:1-234:12. https://doi.org/10.1145/3355089.3356556

Zhong, X., Koh, I., & Fricker, P. D. P. (2023). Building-GNN: Exploring a co-design framework for generating controllable 3D building prototypes by graph and recurrent neural networks. Digital Design Reconsidered: Proceedings of the 41st Conference on Education and Research in Computer Aided Architectural Design in Europe (eCAADe 2023), 431–440. https://doi.org/10.52842/conf.ecaade.2023.2.431