Data and Parameterization Requirements for 3D Generative Deep Learning Models

Mueller, L.-M.; Andriotis, C.; Turrin, M.

DOI
10.52842/conf.ecaade.2024.1.615

Publication date
2024

Document Version
Final published version

Published in
Data-Driven Intelligence

Citation (APA)
Mueller, L.-M., Andriotis, C., & Turrin, M. (2024). Data and Parameterization Requirements for 3D Generative Deep Learning Models. In O. Kontovourkis, M. C. Phocas, & G. Wurzer (Eds.), *Data-Driven Intelligence: Proceedings of the 42nd Conference on Education and Research in Computer Aided Architectural Design in Europe (eCAADe 2024), Nicosia, 11-13 September 2024* (Vol. 1, pp. 615-624). eCAADe. https://doi.org/10.52842/conf.ecaade.2024.1.615

Data and Parameterization Requirements for 3D Generative Deep Learning Models

Lisa-Marie Mueller[1], Charalampos Andriotis[2], Michela Turrin[3]
[1,2,3] TU Delft
[1,2,3]{l.m.mueller|c.andriotis\m.turrin}@tudelft.nl

It is now within reach to use generative artificial intelligence (AI) to autonomously generate full building geometries. However, existing literature utilizing 3D data has focused to a limited degree on architecture and engineering disciplines. A critical first step to expanding the use of generative deep learning models in generative design research is making training data available. This study investigates 3D building model data characteristics that make it suitable for generative AI applications. Key data set attributes are identified through a systematic review of the object-containing datasets currently used to train state-of-the-art 3D GANs. These requirements are then compared to attributes of existing available building datasets. This comparison shows that publicly available data sets of 3D building models lack essential characteristics for generative deep learning. Features that make these building models inadequate for the task include but are not limited to, their mesh formats, low resolution and levels of detail, and inclusion of irrelevant geometry. To achieve the desired properties in this work, necessary transformations of the data are incorporated into a tailored preprocessing pipeline. The pipeline is applied to an existing dataset that contains 3D models of single-family homes. The transformed dataset is tested within state-of-the-art GAN models to assess training performance and document future data requirements for applying deep generative design to buildings. Our experiments show promise for the impact that architectural datasets can make on deep learning applications within the discipline. It also highlights the need for additional 3D building model data to increase the diversity and robustness of new designs.

Keywords: *Generative Deep Learning, Data Sets, Generative Adversarial Networks*

INTRODUCTION

Generative deep learning models have opened new avenues for the creative process. Many industries have benefited, and in recent years, generative models have also started showing promise within architectural design. Machine learning models require training data, and modern applications require significant quantities of high-quality data. To harness the full potential of generative AI, the Architecture, Engineering, and Construction (AEC) field needs to make data available for training and testing. Currently, we lack open data on several design applications, and most importantly, we lack data readable and learnable by generative models.

To generate more data, it is necessary to understand what requirements data sets have when used for deep learning within architecture. Gathering widely applicable open data can benefit the industry not just for generative tasks but for

other deep learning applications as well. In this paper, we specifically evaluate and document the training data requirements related to generative deep learning models for 3D building geometry.

The research is organized as follows. First, the data set requirements are analyzed. Then, existing data sets are surveyed to identify how requirements are met. Finally, a pipeline is developed to preprocess an existing data set. The pre-processed data set is then tested with different Generative Adversarial Network (GAN) (Goodfellow et al., 2014) architectures. The sections of this paper are structured accordingly.

TERMINOLOGY

To distinguish terms that may have different meanings in different fields, we clarify the following:

Geometry Space: The three-dimensional, voxel space the generative model outputs with the produced geometry.

Building models: In this paper, building models refer to three-dimensional digital representations of buildings. To distinguish them from the term 'model' used in machine learning, these models will be referred to as 'building models' or '3D models'.

Model: In machine learning, a model is a parameterized function that can be trained to recognize specific patterns in training data.

Network Architecture: In machine learning, architecture refers to the definition of the neural network, including its structure and hyperparameter settings.

BACKGROUND

There are currently several generative deep learning models spanning a wide range of applications (Li et al., 2024). Some examples include GANs, Variational Auto Encoders (VAEs), transformers, and diffusion models. What they have in common is that they all require data to train. Data requirements vary depending on the application. Applications for which generative models have been used include automating 2D tasks such as generating 2D floor plan layouts (Weber et al., 2022; Park et al., 2024) and translating 2D images of existing buildings to 3D models (Du et al., 2022). These models use 2D image data as inputs, which is more readily available (Wu et al., 2022). In other industries, GANs have also been used to generate 3D models of small objects like chairs and cars (Smith & Meger, 2017; J. Wu et al., 2016), but these examples are limited. In these approaches, 3D models are converted to voxel data, namely, 3D cubic pixels, and the convolutional neural network (CNN) learns 3D features. Therefore, to apply such methods in AEC, it is necessary to have 3D model data. All deep learning models require a significant amount of training data, but when moving to 3D representations, even more data are needed due to increased complexity.

The existing generative deep learning architectures have limited applications within architectural design because they do not have much data to train on and have been primarily developed for small, low-detail objects. Newton (2019) applied GANs to generate 3D models of skyscrapers with up to 100 floors in a 32x32x32 voxel geometry space. Consequently, one voxel represented up to three floors of a building. From the experiment, Newton (2019) noted that GANs need to generate geometry with more detail, and Malah et al. (2024) echoed this sentiment in a recent survey. Generative models need to increase the resolution (number of voxels) and the sharpness (reduce noise) of building geometry so building features can be recognized.

To improve the trained models, not only does the model architecture need to be addressed, the industry needs larger, high-quality data sets, as incomplete, incorrect, or inaccurate data can lead to inaccurate or improperly trained models (Budach et al., 2022). To improve data availability, it is important to document the requirements for data that can be used for generative deep learning in architecture.

DATA REQUIREMENTS

To understand data requirements, it is necessary to consider the properties of the data set and the individual data points, in this case focusing on 3D building models. For this purpose, the state-of-the-

art 3D object data sets ShapeNet (Chang et al., 2015), Pascal3D+ (Xiang et al., 2014), and ObjectNet3D (Xiang et al., 2016) are reviewed.

Quantity of Unique 3D Models. Too little data generally leads to overfitting. This, in turn, causes training to diverge or the network to replicate the data set (Feng et al., 2021). Despite various researchers attempting to train GANs successfully with more limited data, these studies still require several thousand images (Karras et al., 2020) (Wang et al., 2018). Most models, however, are trained on significantly more data. For instance, ImageNet (Deng et al., 2009), a popular data set for 2D GAN training, contains over 1.2M training images, 50K validation images, and 100K test images.

Moreover, complex data sets require more extensive networks with greater capacity (more layers and neurons). This also results in a need for larger data sets, as the size of the required data set scales with the network size. Although an exact number is architecture- and application-dependent, data sets should at least contain several thousand data points.

File Formats. 3D object data sets most commonly use OBJ file formats for the 3D data with MTL files for material information. The AEC industry already stores 3D model data for buildings using a wider variety of file formats.

The OBJ file format is one example. It is an open-source file format for storing mesh geometry information. The files are text-based and can store information about geometry, including material, texture, and lighting. Currently, fewer software libraries allow for manipulating OBJ files based on textures and class labels. Several libraries, including 3.JS, PyMesh, and Vedo, allow for visualizing OBJ files with textures and colors.

The PCD file format is an open-source file format for point cloud information. Point Clouds are a collection of 3D points representing real-world objects, commonly collected from a 3D laser scan. Point clouds can include color, material, reflection, and infrared spectral information for each point, depending on the type of scan. Additionally, class labels can be stored for each point in the point cloud. There are several libraries for importing and editing point clouds, including Open3D, PCL, and PyVista. These libraries also allow for the preprocessing of the point clouds.

Based on the limitations of processing OBJ files, it is more straightforward to use the point cloud file format and encode it for training. Editing data is critical because data must address the needs of the problem at hand. Therefore, researchers are expected to set up preprocessing pipelines for existing data sets. Another benefit of using point clouds is that additional usable data may already be available. Since point clouds are already used to collect data from the built environment and are integrated with tasks in the AEC field (Sawhney et al., 2020), this data can also be reprocessed for generative deep learning applications, providing more data with less time needed to collect it.

Point clouds can be converted to meshes and vice versa, so different formats can be used. However, using open-source file formats is better for reuse, and providing both OBJ and PCD files of a 3D model allows researchers to choose. This allows for more varied applications with less repetitive work to convert files.

Orientation. Different file formats can have different coordinate space formats. OBJ files, for example, do not consider the up direction of a file but simply store information related to X, Y, and Z coordinates. Depending on the software, these coordinates are interpreted with different up directions. For example, different software and libraries may vary using the Y-axis or the Z-axis as the up direction. Therefore, paying attention to how files are imported and oriented is important. It is useful for libraries to include in the metadata which axis is intended as the up direction (aligning with the height of the building).

Level of Detail (LoD). The LoD scale of 3D object models is not the same as the LoD scale of building models. The LoD scale used for objects refers to the number of triangles in a mesh, while in architecture,

LoD describes the building features present in a building model. As noted, increasing the resolution (number of voxels) and the sharpness and clarity of generated building geometry are two critical points that must be addressed when applying deep learning in architectural design. The LoD of training data drives both. As shown in Figure 1, LoD 0 describes a building in 2D as a massing outline, LoD 1 describes the 3D massing of a building, LoD 2 describes the envelope of the building and the form of the roof, and LoD 3 describes a building with its features such as windows and roof overhangs.

When using generative algorithms to create massing or low-resolution geometry, it is acceptable for the data sets to have a LoD of 1 or 2, respectively. However, to apply generative algorithms to create building geometry, the generated building form should be more detailed than the initial massing. Recognizable building features include windows, doors, and a more developed building form. The building form should consider changes in the facade and roof overhangs. Building models that have these characteristics have a LoD of 3.1 or higher. This also means that the data set must have a minimum LoD of 3.1 because the network will need to train on a data set containing the same information level.

Class Labels. In building models, class labels refer to the category of each point. Classes should include key building components like doors, windows, and roofs. There are currently no standard class labels used for building geometry.

Scale. The scale of the 3D models has several implications. First, it is known that deep learning models work better with normalized inputs. Normalizing the training data can help to decrease training time and improve accuracy. However, it is also important to consider the scale of objects with respect to the geometry space and keep it relative to the other objects in the training data set. Additional research is necessary to understand the full impact of the 3D model geometry scales relative to each other. Therefore, if data sets contain normalized 3D models, it is important that the data set authors store and share the scaling information for each 3D model. This allows for future changes in the scaling or normalization, and it allows for extending the data set in the future.

3D Model Quality. The 3D model quality considers factors related to the legibility and cleanliness of 3D geometry. This will vary by application, but building models should be removed if they contain excess geometry not needed for the specific task, incorrectly labeled geometry, or outliers in any of the data set characteristics mentioned. For example, if most 3D models in a data set are LoD 3.1 but some 3D models are LoD 1, the LoD 1 building models are outliers for the LoD characteristic and should not be included in the data set. 3D model quality is specific to each data set. Although some standard pipelines can be developed to identify such differences, evaluating and addressing these factors for each data set and its application is best.

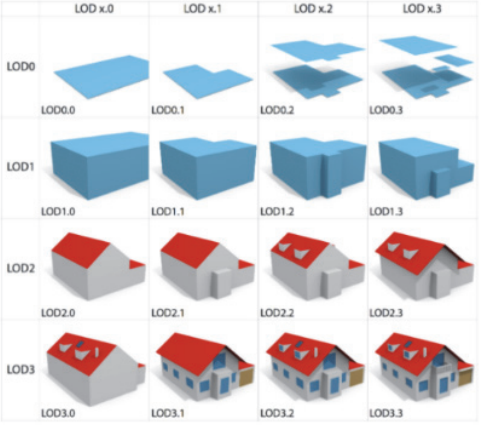

Figure 1
A visual representation of the levels of detail and their sub-levels. Image from Biljecki et al. (2016).

Table 1
Building model data sets that contain 3D models with a LoD 3 or higher and their characteristics, followed by a description of the file format types.

Data Set	Unique Models	Format	Orientation	LoD	Class Labels	Scale
Building Net [1]	2K	OBJ/MTL PCD	Consistent Up*	3+*	32	Normalized
Espoo 3D City Model[2]	*Information Not Available*	SKP CityGML[a] PCD [a]	Consistent Up*	1 3 [a]	1*	Real-World*
Ingolstadt[3]	50+	CityGML SKP	Consistent Up*	3 [b]	1*	Real-World*
Dresden[4]	135K (LoD1&2) 600 (LoD3)	CityGML DXF Shape I3S	Consistent Up*	1-2 3 [a]	1*	Real-World*

[a] not open data
[b] LoD 3 data is only available for building facades, not the entire building
* this data was only evaluated based on literature review
[1](Selvaraju et al., 2021) [2](City of Espoo, n.d.) [3](Schwab, B. and Wysocki, O., 2021) [4](Dresden, n.d.)

CityGML is an open XML-based standard data model and exchange format for 3D city models. **DXF** is an open vector file format for 2D and 3D drawings. **I3S** is an open file format for 3D geographic information system (GIS) content. **MTL** is an open material file format usually associated with an OBJ file. **OBJ** is an open 3D object file format. **PCD** is an open 3D object file format for point cloud information. **Shape** is a vector file format for sharing location, shape, and attributes of geographic information. It is not made open source by the developer Esri but has been reverse-engineered and documented by an open-source community. **SKP** is a proprietary Google Sketch-Up file format.

Summary. The important characteristics include the quantity of unique 3D models, file formats, orientation, level of detail, class labels, scale, and 3D model quality. Data sets should provide information about these features for a more streamlined data selection. Different characteristics have a different level of impact on training. The file format does not impact training but must be accounted for when encoding the 3D models for training. Class labels impact the type of model that can be used but do not impact the quality of the output. Class labels impact how applicable a data set is to a specific problem. The quantity of unique 3D models and the level of detail of the 3D models impact the training and frequently cannot be adjusted in an existing data set without collecting additional data. Finally, scale, orientation, and 3D model quality impact the training but can generally be adjusted by preprocessing the 3D models.

EXISTING BUILDING MODEL DATA SETS

Having determined the requirements, we survey existing datasets to see how they meet the requirements, and this section presents the survey. Data sets that contain building models with a LoD of 3 or higher are evaluated. The 3D Geoinformation Research Group (n.d.) survey is used as a starting point, and additional research is done to identify other 3D building model data sets. These data sets are then tabulated to evaluate the data set

characteristics shown in Table 1. From these data sets, Building Net (Selvaraju et al., 2021) is selected to test with a custom preprocessing pipeline.

PIPELINE FOR BUILDINGNET DATA SET

This section presents a pipeline we develop to preprocess one data set so it meets the requirements discussed in the Data Requirements section. The selected data set was originally generated for 3D model segmentation (Selvaraju et al., 2021). This means the data set has characteristics that are not needed to apply generative deep learning. Therefore, the data must be preprocessed before it can be used for this alternate purpose. The pipeline generates a revised data set consisting of pre-processed 3D models that are a selection of BuildingNet v0.1 3D models. The steps for pre-processing the data set are outlined in this section, and the pre-processed data set is released at https://data.4tu.nl/datasets/4d82052e-650c-4775-8bd9-623df68991b6/1.

The quantity of unique 3D models will not be adjusted as no additional data will be collected. The dataset already meets the file format, orientation, and level of detail criteria. Therefore, the pipeline has to address which classes to use, the scale of the models, and the 3D model quality. Additionally, the models need to be encoded for use in training. An additional step is added, which is selecting a building typology. This step is necessary because the models in BuildingNet are normalized, and scale information is unavailable.

Typology. One building typology is selected to aid with scaling. By comparing the number of models in each building category, residential has the most 3D models with 1,243. From this primary category, the secondary categories of house and villa are used. These are further reduced to select only single-story buildings that are approximately the same height.

Class Labels. For the initial testing, only two class labels are used: whether a voxel is present or not. In future applications, it is possible to encode multi-class labels, like whether a voxel belongs to a door or a window.

Scale. The scale of the buildings in BuildingNet is mostly normalized so that the maximum dimension of each 3D model is one unit. This means buildings with the same height could be scaled differently due to their width and depth. To use these 3D models for training, the height of the buildings should be proportional to the other models in the data set, and it should also be normalized. Because real-world scale information is unavailable for each model in the data set, only single-story houses are selected and scaled to the same height.

3D Model Quality. The selected models still contain unnecessary geometry for the specific generative task. Geometries labeled as vehicle, fence, furniture, ground, gate, lighting, pool, road, and plant/tree are removed since these elements are either too detailed in scale or are part of the site, not part of the building. After removing these labels, it is noticeable that some 3D models also have site geometry that is labeled 'undetermined' or site geometry that is incorrectly labeled as building geometry. To address this, all points of the 3D model belonging to the labels 'wall,' 'door,' and 'roof' are collected, and a bounding box is placed around these points. Any additional geometry outside the bounding box is also removed after removing the site geometry belonging to the labels mentioned above. This helps to clean up most of the site geometry that was 'undetermined' and some of the site geometry that was incorrectly labeled. The 100 models that are the most clean are used as the data set for testing with a GAN.

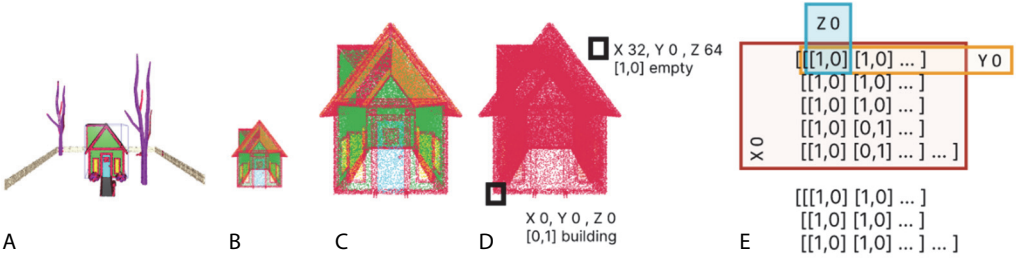

Figure 2
The steps to preprocess and encode the building models.
A) Original with a bounding box around the building
B) Cleaned model
C) Scaled model, height = 1 unit
D) Removed labels
E) Model as tensor.

Data Encoding. The pipeline encodes the point clouds to tensors to convert the geometry to a machine-readable format. The size of the tensor is the same as the geometry space, or output, of the deep learning model.

As discussed in the Background Section, creating high-resolution models is a critical step to applying 3D GANs in architecture. However, the need for high-quality geometry must be balanced with the amount of memory needed to load the models for training. The research team has access to a high-performance computing cluster (HPCC). Based on the specifications of the computing nodes, a geometry space of 160x160x160 was the maximum dimension to limit training times. However, when selecting the geometry space size, there are other factors to consider, such as the number of training data points loaded at once, the padding, stride and kernel sizes, and the size of the building models. The overall size is, therefore, also based on the size of the building models and how the number of layers and stride impact the size of the generated model. After reviewing this information, a 160x160x80 geometry space is selected.

After a geometry space is selected, the models can be encoded to tensors. This is done through one-hot encoding. Each point in the point cloud is transferred to the correct location in the tensor, and the label is encoded. If two labels are used, as with the training in this paper (empty and non-empty building geometry), then the encoding is [1,0] for empty voxels and [0,1] for filled voxels. This can be seen in Figure 2D. Additionally, the values of the labels are encoded into an array, which is saved separately. The final data structure of the encoded data is a 4D tensor where the first axis describes the x location, the second axis describes the y location, the third axis describes the z location of a voxel, and the fourth axis describes the state of the voxel (filled or empty, without loss of generality) as seen in Figure 2E. Figure 2 shows all steps executed during preprocessing.

IMPLEMENTATION

After preprocessing the data, different generative models can be trained with it. GAN models are used for this application. GAN models are selected from the generative models available because they can generate a higher-quality output for 3D building models (in contrast with VAEs) while using fewer computing resources than diffusion models.

GAN (Goodfellow et al., 2014) is an architecture for artificial neural networks that consists of two networks: a generator and a discriminator. These two networks are trained to compete against each other in a zero-sum game, where the generator creates new geometry, and the discriminator determines if this geometry appears real or fake. Over time, the generator learns to fool the discriminator by creating real-appearing geometry.

Several models are tested, which results in the GAN creating 3D building geometry that shows evidence of learned features from the data set (Mueller et al., 2024). The results are 3D models that resemble the data set in size, shape, and proportions, as shown in Figure 3. The architecture used to

generate the results has a generator consisting of ten fully convolution layers with numbers of channels {192, 192, 96, 96, 48, 48, 24, 24, 2, 2}, kernel sizes {4x4x4}, and strides {2x2x2}. The critic mostly mirrors the generator. The input to the generator is a 200-dimensional vector, and the output is a 160 × 160 × 80 matrix with values in [0, 1].

CONCLUSIONS

This research determines data requirements for training 3D generative models by surveying existing data sets. We develop a workflow to preprocess a specific data set to meet the identified requirements and train a GAN using the preprocessed data set to identify and confirm data requirements for training deep learning models to generate 3D building geometry. The trained model learned features from the pre-processed data set and generated building geometry that resembled the training data set in size, shape, and proportion. Additionally, building design details, which were noted as important, are visible in the geometry the trained model produces. Overall, it is identified that the number of unique models, number of categories and their labels, file format, orientation, LoD, number of classes and their labels, and scale are important considerations for creating new data sets for this generative deep learning application. To allow for pre-processing data, researchers can benefit from open file formats and more open-source libraries to handle different file formats during preprocessing properly. For each requirement, the following conclusions are reached:

- Quantity of Unique Models: Larger data sets containing thousands or hundreds of thousands of data points need to be made available;
- File Format: Point Cloud is currently easier to modify with open-source libraries than OBJ/MTL;
- Orientation: This feature should remain consistent within a data set;
- LoD: For generating high-quality geometry LoD3.1 or higher is needed;
- Class Labels: Class labels that distinguish different building components (for example, doors, windows, and walls) expand a data set's applicability. Standardizing the class labels ensures that terminology is consistent across data sets and that the level of detail at which components are labeled is consistent. Additional research is needed to determine which class labels should be used.
- Scale: Further research is needed to determine if the model scale significantly impacts training. If the scale of 3D models in a data set is normalized, the scale factor should be available for each model.

The lack of open data in the AEC industry is a known challenge, and our survey highlighted this gap for 3D building models in particular. The available data is often proprietary; therefore, making existing data open presents challenges. However, some countries permit documenting buildings through photos and scans. Strategies to increase the available data should include 3D scanning existing structures, processing existing city scans to extract individual buildings, and using existing algorithms to generate 3D models from images.

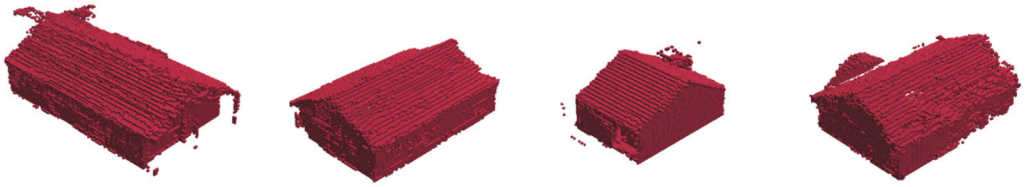

Figure 3
Visualization of different 3D models that the GAN architectures generated using the preprocessed data set.

REFERENCES

3D Geoinformation Research Group (n.d.). Cities/Regions around the world with open datasets [Online]. At: https://3d.bk.tudelft.nl/opendata/opencities/ (Accessed December 2022)

Biljecki, F., Ledoux, H., and Stoter, J. (2016). An improved LOD specification for 3D building models. In Computers, Environment and Urban Systems, pages 25–37.

Budach, L., Feuerpfeil, M., Ihde, N., Nathansen, A., Noack, N., Patzlaff, H., Naumann, F., and Harmouch, H. (2022). The effects of data quality on machine learning performance.

Chang, A. X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., & Yu, F. (2015). Shapenet: An information-rich 3d model repository.

City of Espoo (n.d.). 3D City Model [Online]. At: https://kartat.espoo.fi/3d/index_en.html (Accessed December 2022)

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition (pp. 248–255).

Dresden (n.d.). Virtuelles 3D-Stadtmodell. [Online]. At: https://www.dresden.de/de/leben/stadtportrait/statistik/geoinformationen/3-d-modell.php (Accessed December 2022)

Du, Z., Shen, H., Li, X., & Wang, M. (2022). 3D building fabrication with geometry and texture coordination via hybrid GAN. Journal of Ambient Intelligence and Humanized Computing, 13(11), 5177–5188.

Feng, Q., Guo, C., Benitez-Quiroz, F., and Martinez, A.. 2021. "When Do GANs Replicate? On the Choice of Dataset Size." In 2021 IEEE/CVF ICCV, 6681–90. Montreal, QC, Canada: IEEE.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative Adversarial Networks. Advances in Neural Information Processing Systems 3 (June).

Karras, T., Aittala, M., Hellsten, J., Laine, S., Lehtinen, J., and Aila, T. (2020). Training Generative Adversarial Networks with Limited Data. In Advances in Neural Information Processing Systems, 33:12104–14. Curran Associates, Inc.

Li, X., Zhang, Q., Kang, D., Cheng, W., Gao, Y., Zhang, J., Liang, Z., Liao, J., Cao, Y.-P., & Shan, Y. (2024). Advances in 3D Generation: A Survey (arXiv:2401.17807). arXiv.

Malah, M., Agaba, R., & Abbas, F. (2024). Generating 3D Reconstructions Using Generative Models. In Z. Lyu (Ed.), Applications of Generative AI (pp. 403–419). Springer International Publishing.

Mueller, L.M., Andriotis, C., Turrin, M. (2024). Using Generative Adversarial Networks to Create 3D Building Geometries: 3DBuildingGAN. In eCAADe 2024.

Newton, D. (2019). "Generative Deep Learning in Architectural Design." Technology|Architecture + Design 3 (2): 176–89.

Park, K., Ergan, S., & Feng, C. (2024). Quality assessment of residential layout designs generated by relational Generative Adversarial Networks (GANs). Automation in Construction, 158, 105243.

Sawhney, A., Riley, M., Irizarry, J., & Riley, M. (2020). Construction 4.0.

Schwab, B. and Wysocki, O. (2021). Ingolstadt 3D City Model [Online]. At: https://github.com/savenow/lod3-road-space-models (Accessed December 2022)

Selvaraju, P., Nabail, M., Loizou, M., Maslioukova, M., Averkiou, M., Andreou, A., Chaudhuri, S., and Kalogerakis, E. (2021). Buildingnet: Learning to label 3d buildings. In IEEE/CVF ICCV.

Smith, E. and Meger, D. (2017). Improved Adversarial Systems for 3D Object Generation and Reconstruction. arXiv:1707.09557.

Wang, Y., Wu, C., Herranz, L., van de Weijer, J., Gonzalez-Garcia, A., and Raducanu, B. (2018) Transferring GANs: Generating Images from Limited Data. In ECCV 2018.

Weber, R. E., Mueller, C., & Reinhart, C. (2022). Automated floorplan generation in architectural design: A review of methods and applications. Automation in Construction, 140, 104385

Wu, A. N., Stouffs, R., & Biljecki, F. (2022). Generative Adversarial Networks in the Built Environment: A Comprehensive Review of the Application of GANs across Data Types and Scales. Building and Environment, 223(109477).

Wu, J., Zhang, C., Xue, T., Freeman, W. T., and Tenenbaum, J.B. (n.d.). Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling.

Xiang, Y., Kim, W., Chen, W., Ji, J., Choy, C., Su, H., Mottaghi, R., Guibas, L., and Savarese, S. (2016). Objectnet3d: A large scale database for 3d object recognition. In European Conference Computer Vision (ECCV).

Xiang, Y., Mottaghi, R., and Savarese, S. (2014). Beyond pascal: A benchmark for 3d object detection in the wild. In IEEE Winter Conference on Applications of Computer Vision (WACV).