Using Generative Adversarial Networks to Create 3D Building Geometries

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

DOI
10.52842/conf.ecaade.2024.1.479

Publication date
2024

Document Version
Final published version

Published in
Data-Driven Intelligence

Citation (APA)
Mueller, L.-M., Andriotis, C., & Turrin, M. (2024). Using Generative Adversarial Networks to Create 3D Building Geometries. 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. 479-488). eCAADe.
https://doi.org/10.52842/conf.ecaade.2024.1.479

Using Generative Adversarial Networks to Create 3D Building Geometries

3DBuildingGAN

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

Generative Artificial Intelligence (AI) promises to make a vast impact across disciplines, including transforming the architectural design process by autonomously generating full building geometries. One form of generative deep learning that has been used to create 2D and 3D representations of objects is Generative Adversarial Networks (GANs). Existing literature, however, has limited applications that utilize 3D data for building geometry generation, with previous studies focused on low-scale 3D geometries suitable for objects such as chairs or cars. This paper develops a new GAN architecture to produce high-resolution feasible building geometry. The training dataset used is a selection of 3D models of single-family homes from an existing database, pre-processed for the specific application. State-of-the-art GAN models are initially tested to establish baseline performance and applicability potential. Then, a systematic study is performed to identify the structure and hyperparameters necessary to successfully fit a GAN to this design task. The successful architecture, named 3DBuildingGAN, uses a combination of Wasserstein loss with gradient penalty, leaky rectified linear units for neuron activation in the generator and the critic, and the root mean squared propagation optimizer with a fixed learning rate. The proposed model generates outputs similar in size, shape, and proportion to the training data with minimal noise in the output. Evaluation of memorization properties indicates open research directions, such as incorporating memorization rejection and training on larger data sets. Finally, the study reflects on how AI algorithms can reshape creativity through data-driven design solutions.

Keywords: *3D Generative Adversarial Networks, Deep Learning, Artificial Intelligence*

INTRODUCTION

Architects and engineers can benefit from increased efficiency and precision by embracing automation in 3D design tasks. Creating machines that can autonomously generate designs will change the way we design our built world. Deep learning has brought unprecedented capabilities in this direction. Specifically, generative adversarial networks (GANs) (Goodfellow et al., 2014) have emerged as a tool allowing us to harness the potential of 3D automation in design. They have already been successfully used to generate 3D models of small objects, such as chairs and cars (Smith & Meger, 2017; Wu et al., 2016). These explorations show promise that generative models can be used not just for 2D image-like applications but also for generating complete, novel, and contextually relevant architectural designs.

This paper aims to explore the potential of different GAN architectures for generating 3D building geometry through an in-depth exploration of different architectures. The experiments scale up the geometry space to 160x160x80 voxels and train on a data set consisting of 3D point cloud models of single-family homes encoded into tensors. Through experiments covering over 35 different deep learning architectures, this paper aims to identify the GAN architecture that best generates 3D models of buildings and address the challenges and possibilities of this task. The proposed architecture, named 3DBuildingGAN, is found to generate high-resolution 3D buildings with minimal noise.

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

Deep learning methods have presented new tools to automate design tasks by introducing algorithms that train themselves through the use of neural networks. There are currently several generative deep learning models in use and they span a wide range of applications (Li et al., 2024). These, among others, include GANs, Variational Auto Encoders (VAEs), transformers, and diffusion models. Each of these models has not yet been extensively tested for design applications. In general, transformers are a competitive replacement for convolution, but a generic transformer architecture is unavailable (Lahoud et al., 2022). VAEs learn latent representations of datasets; however, the output is often low quality because they focus on learning the distribution of the training data rather than specific samples (Cai et al., 2019). On the other hand, diffusion models produce exceptional, high-quality outputs but require significant computational resources to train (Cao et al., 2023). When compared with these generative models, GANs strike a balance between high-quality outputs and computational efficiency.

GANs (Goodfellow et al., 2014) consist of two neural networks trained to compete in a zero-sum game. One network, the generator, generates an output, such as a 2D image or 3D model. The other network, the discriminator or critic, uses the dataset to determine if the generator's output appears real or fake or the probability distribution of its realness or fakeness, respectively. Existing applications of generative models include models trained on 2D data, such as those generating 2D floor plan layouts (Weber et al., 2022; Park et al., 2024) and those creating 3D models of buildings from 2D images (Du et al., 2022).

When GANs train on 3D data, 3D models can be converted to voxels, i.e., 3D pixels in perfect cubes encoded in a 4D tensor. The tensor captures the X, Y, and Z coordinates of the voxel and the voxel class, e.g., filled or empty. The final dimension can also be used to encode multi-class predictions. Thereby, convolutional neural networks (CNN) can learn 3D features of the tensor during training.

There are fewer examples of training GANs on 3D geometry. 3DGAN (Wu et al., 2016) and Improved Wasserstein GAN (IWGAN) (Smith & Meger, 2017) are current state-of-the-art architectures that train with and generate 3D geometry. 3DGAN is based on the original GAN architecture developed by Goodfellow et al. (2014) with modifications that make it more suitable for 3D geometry:

- Pooling layers are replaced with strided convolutions in the discriminator and fractional strided convolutions in the generator (Springenberg et al., 2014).
- Fully connected hidden layers are removed for deeper architectures.
- ReLU activation is used in the generator, and Leaky ReLU activation is used in the discriminator (Radford et al., 2016).

IWGAN is similar to 3DGAN but uses Wasserstein loss (Arjovsky et al., 2017) to stabilize training. In Wasserstein GANs, the discriminator is called the critic. Wasserstein loss minimizes the Wasserstein distance (the measure between the generator and the critic's probability distribution) to help stabilize training and generate higher-quality samples (Arjovsky et al., 2017). Both methods are primarily used to generate small objects like chairs and cars and were applied using a small geometry space of 64x64x64 for 3DGAN and 32x32x32 for IWGAN.

There are few examples of training GANs with 3D geometry in the field of architectural design. Newton (2019) applied an IWGAN (Smith & Meger, 2017) to generate 3D models of skyscrapers up to 100 stories tall. The output was limited to a 32x32x32 voxel geometry space, so one voxel represented up to 3 floors of a building. Malah et al. (2024) reiterated Newton's (2019) conclusions that generative models must produce higher-quality and less noisy outputs for application in 3D building design. Overall, generative models must overcome two main challenges: increasing the resolution (number of voxels) of output geometry and increasing the sharpness and detail of building geometry. Increasing the sharpness of detail includes reducing the noise in the output and having building features from LoD 3 models visible in the generated output.

COMPUTATIONAL WORKFLOW

Data Set

There is a need for large datasets to train deep learning models for architectural design tasks. BuildingNet (Selvaraju et al., 2021) is selected here because it contains over 2,000 3D models with a level of detail (LoD) of 3.1 or higher as point clouds and meshes. The LoD describes the details that the 3D model contains, with a LoD of 3 and higher, showing windows and doors, roof overhangs, and changes in the façade. This data set has enough detail to train GANs that increase the resolution and sharpness of the output. We focus on the typology of single-family houses due to the smaller size of the buildings and the more extensive availability of data. The final dataset used in this study consists of 100 single-story, single-family house models in the form of 3D point clouds. These models are first preprocessed to remove site geometry, scale all models to one unit in height, remove unnecessary class labels, and encode each model to a 4D tensor using one-hot encoding (Mueller et al., 2024).

Methodology

An exploratory research methodology is applied to test and evaluate existing and new GAN architectures. A literature review is used to collect data about different network architectures and hyperparameter settings for GANs. These are tested with the selected single-story, single-family house model data set. The testing starts with GAN architectures, based on the two state-of-the-art architectures of 3DGAN (Wu et al., 2016) and IWGAN (Smith & Meger, 2017). After each model's performance is evaluated for problems like non-convergence, mode collapse, and diminished gradient, the hyperparameters are adjusted based on possible solutions from the literature review. The new resulting architectures are then tested, and the process is repeated. Additionally, changes to the number of layers and channels are tested to increase output resolution and reduce noise.

GAN Assessment Criteria

Assessment of each GAN architecture is completed during and after training to evaluate how stable the training is. Additionally, after a GAN successfully trains, the quality of the output is assessed. For

assessment during training, the loss functions of the generator and discriminator or critic are used to indicate how stable the training is and if the models converge (Goodfellow et al., 2014).

Different methods had to be considered to assess the output. For GANs, the loss functions do not provide information about the output because the loss functions cannot evaluate the quality of the generated samples. When using GANs for 2D tasks, quantitative performance metrics, such as the inception score, can be used instead (Borji, 2022). These quantitative methods cannot be applied to 3D applications. Therefore, qualitative evaluation criteria are used to compare the performance of different architectures. This includes comparing the generated output to the training data and visually evaluating the output. To assess the performance, including stability, the output of the GAN is saved every 100 epochs. These outputs are then visualized and compared to the data set to check for the shape, size, and resolution of the generated geometry.

Two important issues to assess, since GANs frequently suffer from them (Khanuja & Agarkar, 2023), are mode collapse and memorization. Mode collapse describes if the output of a GAN is always the same as the other generated outputs. Examples of mitigation strategies include adjusting the learning rate and batch sizes. Memorization describes when the output is identical to the training data and is generally less common. However, when using a relatively small dataset, like in this case, GANs may memorize data (Feng et al., 2021).

Experiment Set-Up

Keras, an API for Tensorflow, is used to implement each architecture. The base architectures are modeled after 3DGAN (Wu et al., 2016) and IWGAN (Smith & Meger, 2017). These are tested by training first on a single data point and, only when successful, on the complete data set containing the 100 data points. The first 13 architectures are trained on only one data point to overfit the models while minimizing the time needed for training. The results show that using Wasserstein loss (Arjovsky et al., 2017) is necessary for generating an output that begins to resemble building forms. Additionally, leaky ReLU should be used in the generator and the critic. The best-performing architecture consists of four fully convolutional layers with {48, 24, 12, 2} channels, uses Wasserstein loss, Leaky ReLU in the generator and discriminator, and the ADAM optimizer (Kingma & Ba, 2014) with learning rate decay. When training on the complete data set, the network learns to create building geometry like walls, but the output is not the correct size, shape, or proportion compared to the training data shown in Figure 1.

Figure 1
This is a visualization of the results of 11G, which produces some building features like walls, but the massing does not resemble the data set.

Figure 2
This visualization of 11R's result shows a noisy output roughly the same size, shape, and proportion as the training data.

Hyperparameter Tuning

Several hyperparameters need to be adjusted to improve the output, so additional hyperparameter options are selected through a literature review. An additional 15 variations are tested using network depth and width 11, as noted in Table 1. These experiments are detailed in Table 2, Models 11G-11W. Architecture R is the most promising, producing a noisy output roughly the same size, shape, and proportion as the training data, as shown in Figure 2. Architecture R uses the Wasserstein Loss with gradient penalty, Leaky ReLU in both the generator and the critic and the RMSprop optimizer (Hinton, 2018) with a learning rate of 0.00005.

The best-performing architecture generates a model that starts to resemble building geometry but still contains noise and unclear features. To tune the detail of the output, architectures with more layers and channels are explored. The number of channels is the number of features the model will output in each layer. The depth of the architecture refers to the number of layers that the architecture has. When making these adjustments, balancing the number of channels and the network depth is important. Too many layers means the model could more easily overfit, while too many channels can unnecessarily increase the memory requirements.

Since Architecture G performs well, it is tested with three additional variants where the network depth and width are adjusted. These included architectures 14, 15, and 16, as described in Table 1, with the results described in Table 2, Models 14G-16G. Architecture J is tested with the same architectures to understand the impact of using batch normalization with the results described in Table 2, Models 14J-16J. Architecture R is tested with an additional six different variants, which include 14 through 19, as described in Table 1, with results described in Table 2, Models14R-19R.

Architecture R is the best-performing architecture and generates good-quality outputs even with fewer channels and layers. The results improve when the layers and channels are increased, and the outputs contain significantly less noise. Architectures 16R, 17R[1], and 17R[2] perform best as they generate outputs that are roughly the same size, shape, and proportions as the training data set. The outputs also include better building features, such as recesses and roof forms, as shown in Figure 3.

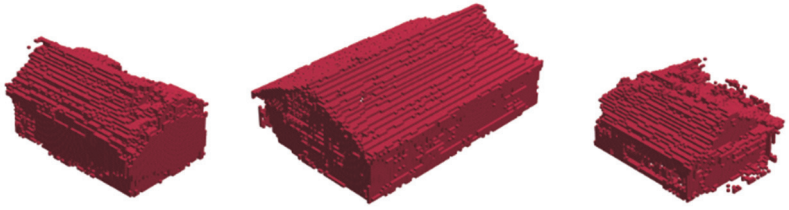


Figure 3
Visualizations of models generated by architecture 17R[1].

Table 1
The ID and factors of the different network depths and widths tested.

ID	Details for Critic (reverse for Generator)
11	4 layers 48-24-12-2
14	5 layers 96-48-24-12-2
15	5 layers 192-96-48-24-2
16	10 layers 96-96-48-48-24-24-12-12-2-2
17	10 layers 192-192-96-96-48-48-24-24-12-12-2-2
18	10 layers 512-512-256-256-128-128-64-64-2-2
19	15 layers 512-512-512-256-256-256-128-128-128-64-64-64-2-2-2

Model ID	Details	Results	
		train	output
base	Wasserstein Loss + Leaky ReLU (G+C) + ADAM optimizer		
11G	11 (Table 1) lrDecay	[S]	[3]
11H	11 (Table 1) lrDecay + label smoothing	[S]	[2]
11J	11 (Table 1) lrDecay + batch norm	[S]	[2]
11K	11 (Table 1) RMSProp optimizer + lrDecay	[S]	[2]
11L	11 (Table 1) RMSProp optimizer + lrDecay + batch norm	[S]	[2]
11M	11 (Table 1) RMSProp optimizer	[U]	[2/3]
11N	11 (Table 1) RMSProp optimizer + batch norm	[U]	[3]
11P	11 (Table 1) lrDecay + gradient penalty	[U]	[2]
11Q[1]	11 (Table 1) lrDecay + layer norm + gradient penalty	[U]	[1]
11Q[2]	11 (Table 1) lrDecay + batch norm (G) + gradient penalty	[S]	[2]
11R	11 (Table 1) RMSProp optimizer + gradient penalty	[S]	[4/5]
11S[1]	11 (Table 1) RMSProp optimizer + layer norm + gradient penalty	[U]	[1]
11S[2]	11 (Table 1) RMSProp optimizer + batch norm (G) + gradient penalty	[S]	[3]
11T	11 (Table 1) lrDecay + gradient clipping	[S]	[2]
11U	11 (Table 1) lrDecay + batch norm + gradient clipping	[S]	[2]
11V	11 (Table 1) RMSProp optimizer + gradient clipping	[S]	[2]
11W	11 (Table 1) RMSProp optimizer + batch norm + gradient clipping	[S]	[2]
14G	14 (Table 1) lrDecay	[U]	[2]
14J	14 (Table 1) lrDecay + batch norm	[U]	[2]
14R	14 (Table 1) RMSProp optimizer + gradient penalty	[S]	[3]
15G	15 (Table 1) lrDecay	[U]	[2]
15J	15 (Table 1) lrDecay + batch norm	[S]	[4]
15R	15 (Table 1) RMSProp optimizer + gradient penalty	[S]	[4]
16G	16 (Table 1) lrDecay	[U]	[2]
16J	16 (Table 1) lrDecay + batch norm	[U]	[2]
16R	16 (Table 1) RMSProp optimizer + gradient penalty	[S]	[5]
17R[1]	17 (Table 1) RMSProp optimizer + gradient penalty	[S]	[5]
17R[2]	17 (Table 1) RMSProp optimizer + gradient penalty with equal padding achieved by kernel size 5x5x5	[S]	[6]
18R	18 (Table 1) RMSProp optimizer + gradient penalty	[S]	[5]
19R	19 (Table 1) RMSProp optimizer + gradient penalty	[S]	[1/2]

Table 2
This is a list of experiments run to tune a GAN architecture to generate more refined geometry. All architectures are tested with the network depth and width of architecture 11 per Table 1, and only the most promising architectures are then tested with different depths, widths, and kernels. When learning rate decay is not used, the learning rate is set to 0.00005.

Each architecture uses all features of the base architecture with select adjustments noted under Details. (G) the generator; (C) the critic; [U] the training was unstable; [S] the training was stable; [1] the output was noise; [2] the output size, shape, and/or proportions are significantly different from the training data; [3] the output is correct in size or proportion but the shape is not accurate; [4] the output is noisy but roughly the correct size, shape, and proportions; [5] output is roughly the correct size, shape, and proportions and has less noise; [6] the geometry is very similar to the resolution of the training data. The red highlighted rows indicate the best-performing architectures.

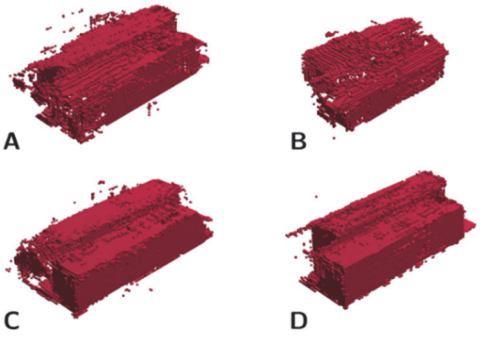


Figure 4
Architecture 17R[2] generates four geometries with similar massing, but each one (A-D) has a different roof structure.

RESULTS

Since 16R, 17R[1], and 17R[2] have the highest quality output of the tested architectures, they are used to generate 100 models each. These models are reviewed and compared against the training data set. The generated models are evaluated for repetition and memorization.

The generated models show no signs of repetition. When the massing of different outputs is similar, each output incorporates different detailed geometry, like the roof forms shown in Figure 4. Additionally, a variety of forms are generated.

When reviewing for memorization, it is not possible to empirically state one way or the other if a GAN is memorizing data. There are slight differences between memorizing and creating a new, similar building that incorporates learned features. In the outputs, there are clear instances where the generated geometry repeats similar forms, but it is unclear if the details show learned features.

Model 43, shown in Figure 5, is generated by architecture 17R[1]. This model has massing similar to the dataset, specifically *mesh3513* and *mesh4362*. When reviewing the details, there are also some key differences. The roof slopes of *model 43*, for example, are all at the same height. In *mesh3513*, however, the roof of the central room is higher than the rest, and *mesh4362* also has this feature. Furthermore, the roof slope in *model 43* is more similar to those of *mesh0109* and *mesh2529*. This suggests that the massing and the roof features are learned from different models and are eventually combined.

Model 60, shown in Figure 5, is another example generated by architecture 17R2. The data set only contains three flat-roofed houses, so comparing the generated model with the data set models is easier. The massing of model 60 is similar to *mesh1329* from the data set, but in the roof edge detail, it incorporates the parapet detailing from *mesh0648* and *mesh3480*.

It is challenging to evaluate some of the other generated models for memorization. Model 8, shown in Figure 5, is generated by architecture 17R[2]. Model 8 has massing similar to *mesh3240,* which includes the inset porch on the left wall. Some differences include the wall on the right of Model 8 not having any openings, while the same wall in *mesh3240* has two openings. The top right of Model 8 has what looks to be a chimney, but the noise does not allow a clear interpretation. *Mesh3240* has an extrusion for a dormer window in the same area. In this case, it is unclear if the generated model replaced the extrusion with a learned chimney feature, such as the chimneys in *mesh0456* and *mesh0718*, or if it poorly replicated the dormer window. The remaining voxels may also suggest the beginning of the model learning to remove the dormer to replace it with a roof section, demonstrating that it is learning features. There are possible explanations for this model being an example of memorization and of learning.

Based on the above exploration, architecture 17R[1] is proposed as the 3DBuildingGAN for this dataset. Overall, analyzing the models reveals a degree of memorization, which is important to address in future research. Network width and the size of the training data set are the two main factors that can contribute to a network memorizing training data. In this case, a training data set of 100 models is small. Therefore, it is important to address this by training on larger data sets or by incorporating memorization rejection (Bai et al., 2022). However, the evaluation shows that the model is learning features of the training data set, which reinforces the promise of GANs in generating 3D building geometry.

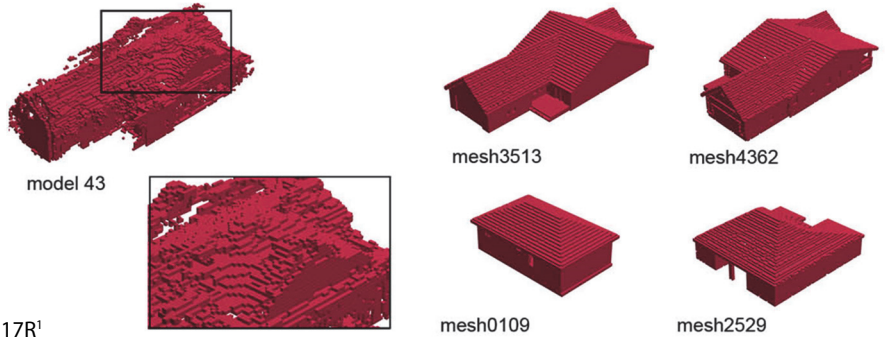
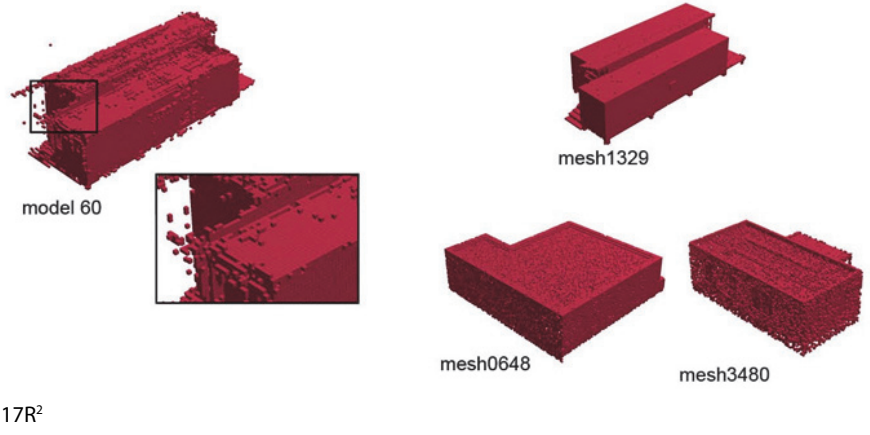
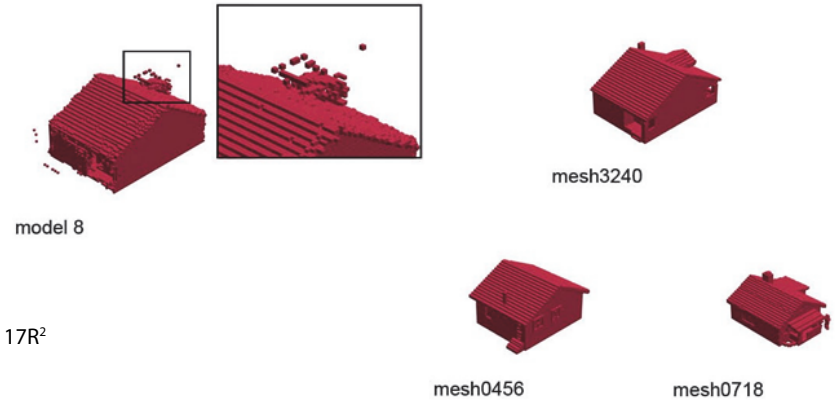


Figure 5
The roof form of model 43 generated by architecture 17R[1] suggests that the model learns to create massing and apply different roof styles.



Similarly, examples in the geometry generated by 17R[2], such as model 60, demonstrate learned roof shapes.



Others, like model 8, are challenging to categorize. The model could either be an instance of memorization or of learning.

3D BUILDING GAN

3DBuilingGAN is a model trained to generate single-story, single-family house models with 3D training data and outputs. The architecture is as follows:

LOSS FUNCTION Wasserstein Loss.

GENERATOR The generator consists 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}. Leaky ReLU of parameter 0.2 is used, and batch normalization is not used. The Sigmoid activation function and a flatten layer are used at the end. The input is a 200-dimensional vector, and the output is a 160x160×80 matrix, indicating the probability that a voxel is visible in [0, 1]. RMSProp is used with Learning Rate=0.00005.

CRITIC As a mirrored version of the generator, the critic takes as input a 160×160×80 matrix and outputs a scalar between minus and plus infinity. The activation function used is Leaky ReLU of parameter 0.2. The final layers are a flatten layer followed by a dense layer with one channel.

CONCLUSIONS

In this research, we set out to determine if it is possible to increase the resolution (number of voxels) and the sharpness and detail of generated building geometry to generate recognizable building features. Through experiments that tested over 35 different GAN architectures to tune their hyperparameters, an architecture was developed that improves results on both points. 3DBuildingGAN is proposed, showing that it is possible to increase the geometry space of 3D GAN architecture while generating high-resolution building models with minimal noise. Well-performing models generate 3D building geometry incorporating structural (wall and roof forms) and detail (recesses and chimneys) features.

Two optimizers are tested. Architectures using ADAM as the optimizer perform better when also using learning rate decay. Architectures using RMSprop as the optimizer perform better with a set learning rate. In the experiments, RMSProp outperformed ADAM for this application when comparing the output for size, shape, and proportion against the models in the dataset.

It is determined that if architectures perform well compared to others when all architectures have few layers and channels, the same high-performing architectures also perform well when the depth and width of the network are scaled up. Starting with smaller networks aids in quickly testing many architectures. After identifying a well-performing architecture, adjusting the network's depth and width can help reduce the noise in the output.

Several key hyperparameters that can lead to stable training and high-resolution output in a 160x160x80 geometry space are identified and applied in the final 3DBuildingGAN model. These include using Wasserstein loss with gradient penalty, Leaky ReLU in the generator and critic, and RMSProp as the optimizer with a fixed learning rate of 0.00005.

Looking ahead, we consider generative deep learning a valuable design tool. Successfully using GANs to generate 3D building geometry expands the use of automation in architecture. With the advances in other generative models, it is possible to imagine a future workflow where 3D GANs generate recommendations for building geometry based on project requirements, a semantic segmentation algorithm groups the geometry and assigns building component types, 2D GANs generate the floor plans, and the collection of information is transferred into a building information model. Additional research is needed to integrate deep learning into existing workflows. These advances also rely on more data. Expanding data availability and training with more diverse and varied data would allow generative deep learning to be more widely applicable. By encoding 3D spatial information into digital design tools, these tools can propose novel solutions that align with project goals and requirements evolving design from optioneering to a data-driven generative task.

REFERENCES

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein GAN. arXiv:1701.07875.

Bai, A., Hsieh, C.-J., Kan, W., and Lin, H.-T. (2022). Reducing Training Sample Memorization in GANs by Training with Memorization Rejection.

Borji, A. (2022). Pros and cons of gan evaluation measures. Journal of Computer Vision and Image Understanding, 215:103329.

Cai, L., Gao, H., & Ji, S. (2019). Multi-Stage Variational Auto-Encoders for Coarse-to-Fine Image Generation. In Proceedings of the 2019 SIAM International Conference on Data Mining (SDM) (pp. 630–638). SIAM.

Cao, H., Tan, C., Gao, Z., Xu, Y., Chen, G., Heng, P.-A., & Li, S. Z. (2023). A Survey on Generative Diffusion Model (arXiv:2209.02646). arXiv.

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., & Martinez, A. (2021). When do GANs replicate? On the choice of dataset size. 2021 IEEE/CVF ICCV, 6681–6690.

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).

Hinton, G. (2018). Neural Networks for Machine Learning Lecture 6a Overview of mini-batch gradient descent.

Khanuja, H. K., & Agarkar, A. A. (2023). Towards GAN Challenges and Its Optimal Solutions. In Generative Adversarial Networks and Deep Learning. Chapman and Hall/CRC.

Kingma, D. P. and Ba, J. (2014). Adam: A Method for Stochastic Optimization. In International Conference of Learning Representations

Lahoud, J., Cao, J., Khan, F. S., Cholakkal, H., Anwer, R. M., Khan, S., & Yang, M.-H. (2022). 3D Vision with Transformers: A Survey (arXiv:2208.04309).

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). Data and Parameterization Requirements for 3D Generative Deep Learning Models. 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.

Radford, A., Metz, L., & Chintala, S. (2016). Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks (arXiv:1511.06434). arXiv.

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.

Springenberg, J. T., Dosovitskiy, A., Brox, T., & Riedmiller, M. A. (2014). Striving for Simplicity: The All Convolutional Net. CoRR.

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. Automation in Construction, 140, 104385

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.