BUBBLE2FLOOR

A pedagogical experience with deep learning for floor plan generation

PEDRO VELOSO[1], JINMO RHEE[2], ARDAVAN BIDGOLI[3] and MANUEL LADRON DE GUEVARA[4]
[1,2,3,4]*Carnegie Mellon University, USA.*
[1]*University of Arkansas, USA.*
[1]*pveloso@uark.edu, 0000-0003-1597-9533*
[2]*jinmor@andrew.cmu.edu, 0000-0003-4710-7385*
[3]*abidgoli@andrew.cmu.edu, 0000-0001-5486-2413*
[4]*manuelr@andrew.cmu.edu, 0000-0002-4585-3213*

Abstract. This paper reports a pedagogical experience that incorporates deep learning to design in the context of a recently created course at the Carnegie Mellon University School of Architecture. It analyses an exercise called Bubble2Floor (B2F), where students design floor plans for a multi-story row-house complex. The pipeline for B2F includes a parametric workflow to synthesise an image dataset with pairs of apartment floor plans and corresponding bubble diagrams, a modified Pix2Pix model that maps bubble diagrams to floor plan diagrams, and a computer vision workflow to translate images to the geometric model. In this pedagogical research, we provide a series of observations on challenges faced by students and how they customised different elements of B2F, to address their personal preferences and problem constraints of the housing complex as well as the obstacles from the computational workflow. Based on these observations, we conclude by emphasising the importance of training architects to be active agents in the creation of deep learning workflows and make them accessible for socially relevant and constrained design problems, such as housing.

Keywords. Architectural Pedagogy; Deep Learning; Conditional GAN; Space Planning; Floor Plan; SDG 4; SDG 9.

1. Introduction

In the last decade, Deep Learning (DL) spearheaded the latest wave of research in Artificial Intelligence (AI) and brought a myriad of technological advancements in various fields. DL is an umbrella term that refers to the methods that "... allow computers to learn from experience and understand the world in terms of a hierarchy of concepts, with each concept defined through its relation to simpler concepts" (Goodfellow et al., 2016, p. 1). DL typically relies on the use of Deep Neural Networks,

POST-CARBON, Proceedings of the 27th International Conference of the Association for Computer-Aided Architectural Design Research in Asia (CAADRIA) 2022, Volume 1, 373-382.

namely, multi-layered models with small computational components that can infer the hierarchy of concepts related to the solution of a problem from a given dataset.

DL gained a lot of attention from the design technology community and the AEC industry and resulted in a significant number of publications, presented in newly established DL sections in AEC flagship conferences. In this paper, we describe our experience of teaching DL and generative design to architecture students, as part of a graduate-level course at the Carnegie Mellon University School of Architecture. After the initial modules of the course, where the students learned the fundamentals of DL frameworks, we introduced a generative design exercise named Bubble2Floor (B2F). B2F entails generating custom floor plan arrangements in a predefined row-house complex using Generative Adversarial Networks (GANs).

This paper adopts a pedagogical research method to develop a critical reflection on B2F and on the effectiveness of using DL as a generative method for architectural design education. The next sections consist of (2) a review of previous pedagogical initiatives that use GANs for architectural design, (3) a general description of B2F, (4) an analysis of the design development and challenges in B2F, and (5) the conclusion and discussion. We assume that the reader will be familiar with basic terminology associated with DL.

2. Review of Deep Learning in Architectural Education

A large part of the research on DL for architecture is targeted at the synthesis of design representations. Initiatives that either analyse the limits of DL in design (Joyce & Nazim, 2021) or that test these ideas in an educational setting are still in their infancy. Some of the early efforts to integrate the recent bloom of DL in design pedagogy were crystallised as a series of workshops in CAAD conferences. For instance, the line-up of workshops in Smartgeometry (Sg2018 Workshops, 2018) included Fresh Eyes, where participants used various DL techniques to classify building types and predict building performance among other inquiries into DL and design. ACADIA hosted a number of workshops focused on DL and specifically GANs. Deepdesign (ACADIA Workshops, 2020), Latent Morphologies, and The Generative Game (ACADIA Workshops, 2021) all utilised a variation of GANs as a design assisting tool. The newly established platform, Digital Futures, has been the home for several workshops on the intersection of DL and design as well. For instance, Machine Intelligence in Architecture 2.0 (Tian et al., 2021) addressed various applications of GANs in 2D, 2.5D, and 3D. Throughout these venues, GANs were the most popular generative models adopted by workshop organisers and tutors.

The growing interest in the applications of DL recently found its way into a handful of universities' curriculums. During the past years, Carnegie Mellon University (USA), University of the Arts London (UK), Goldsmith University (UK), among others, integrated creative AI and ML in their course catalogues. Some architecture schools, such as the School of Architecture at Carnegie Mellon University as well as Taubman College at University of Michigan, have initiated dedicated courses to AI, ML, DL, and design. A few other schools, however, covered this topic in their existing curriculum under the broader umbrella of computational design.

3. Bubble2Floor (B2F)

B2F is a design exercise that consists of generating floor plans with DL for a row-house complex of six three-story units. The first unit of the complex is used as an example to demonstrate the expected level of details in the design, while the other five units are reserved for each group of students to work on.

We designed a pipeline for B2F (Figure 1) to address the design constraints of the building layout problem based on the interaction of the designer with a well-established design representation. In this pipeline, the DL model receives bubble diagrams as input and translates them to floor plan diagrams as output. This translation is a long-standing problem, which relied on human intervention in early generative experiments (Weinzapfel & Negroponte, 1976) and on techniques based on graph embedding and triangulation (Nourian et al., 2013). Recently, this problem has been addressed with DL techniques (Nauata et al., 2020, 2021). For students to customise bubble diagrams and explore variations of floor plan designs, we divided the B2F pipeline in three parts: (3.1) data processing and synthesis, (3.2) training, and (3.3) design.

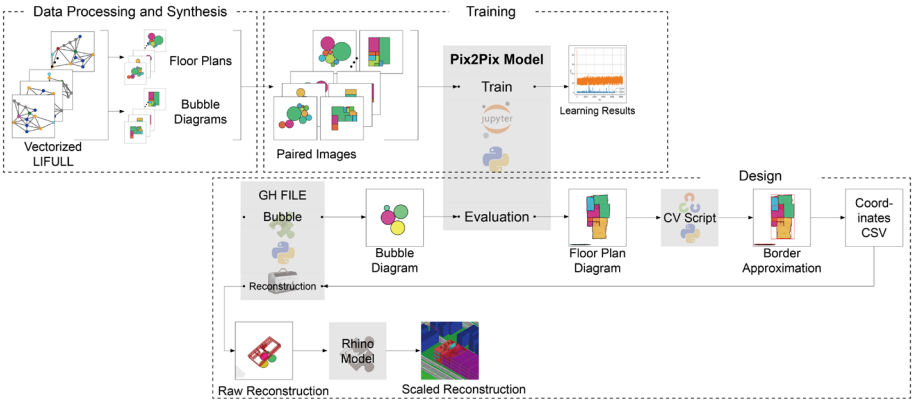

Figure 1. Computational pipeline of B2F: data processing and synthesis, training, and design.

3.1. DATA PROCESSING AND SYNTHESIS

The dataset is based on vector-graphics annotations of LIFULL, an image-based dataset of millions of floor plan images from a real estate information service in Japan (Liu et al., 2017). We used the vectorized representation of the floor plans and programmatic labels (bedroom, restroom, bathroom, kitchen, living room, entrance, hall, corridor, stairs, balcony, washing room, etc.) as the base to generate pairs of floor plan and bubble diagrams (see Figure 1).

The floor plan diagram is defined by the original room layout polygons with black edges and fill colours associated with the programmatic labels. The way to generate bubble diagrams is converting the respective room polygons into discs of equivalent areas with the same edge and fill colour. Then, a constraint-based physics simulator removes overlapping between the discs. Based on the parameters and on the stochasticity of the simulation, this process can generate multiple bubble diagrams for the same floor plan. While both the bubble and floor plan diagrams contain information

related to program, area, position, and adjacencies, the floor plan diagram contains additional information about the position and room shape. Each diagram is converted into a 256 × 256 PNG image file, forming a dataset of 4,987 images by pairing bubble and floor plan diagrams. The use of these two diagrams intends to display the potentials and limits of design representation and translation in early design stages.

3.2. THE TRAINING WORKFLOW

B2F uses a conditional GAN model named Pix2Pix (Isola et al., 2017) for image-to-image translation with paired images. Pix2Pix is a generative adversarial model that can relate conditional inputs to outputs—i.e., it uses the input images as conditional variables for generating a new image, in contrast to generating images from random noise, as in the original GAN model (Goodfellow et al., 2014). In B2F, the synthetic dataset is used to train the model to generate a floor plan diagram conditioned on a given bubble diagram. We developed a custom training workflow based on the implementation provided by (H. Kang & Jha, 2018).

Students spent one week comprehending the main concepts of conditional GANs, Pix2Pix, and the pipeline of B2F before training the model. We provided a Jupyter Notebook file to train the B2F model with the dataset through Google Colab—a cloud computing service with all required libraries and dependencies pre-installed. This notebook had the basic algorithms implemented with suggested initial settings and hyper-parameters to train the model. Also, a model pre-trained for 100 epochs was provided as the training baseline, so students could test the pipeline and refine the model with additional epochs if necessary. Some students individually trained the networks for additional epochs. Other students contributed to the development of a single model with combined training sessions. Overall, the models were trained from 200 to 500 epochs in total.

3.3. DESIGN WORKFLOW

We provided a post-processing pipeline to integrate B2F into an end-to-end design workflow. This pipeline incorporates three components:

- A geometric model in Rhinoceros3D (RH) including the site and the row-house building with different housing units
- A Grasshopper (GH) definition that can generate and colour the bubble diagrams according to their program
- A computer vision script based on OpenCV library and a GH definition that can translate images to geometric partitions

After the training phase, students generated new bubble diagram images, using the provided GH file, which automatically adjusts the number, size, position, and program/colour of the bubbles in real time. The trained Pix2Pix model takes the bubble diagram, translates it into corresponding feature vectors, and synthesises a floor plan diagram as an image.

From this floor plan diagram, students can explore potential configurations for the row-house building model with two different post-processing methods: (a) rectangular

approximation, and (b) coordinate clustering. In rectangular approximation, an OpenCV script tracks the borders of room partitions, approximates them into rectangular polylines, and saves their coordinates. In coordinate clustering, the script extracts point coordinates from the floor plan image. Then, a K-means clustering algorithm is used to cluster these point coordinates based on the number of clusters defined by the designer (Figure 2). This process reinforces the presence of axes and orthogonal wall partitions parameterized by the designer.

Students completed the design of the unit by adding openings and stairs to wall partitions in the RH file. Each student repeated this process for each floor to complete designing the house unit. At the end, we collected the five different units and merged them into the row house in RH (Figure 3).

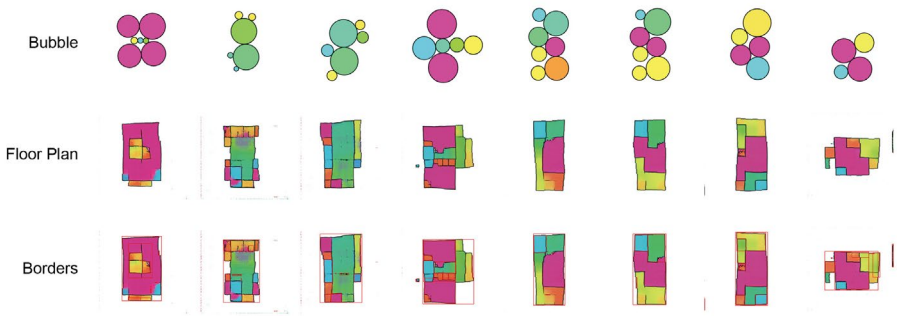

Figure 2. Floor plan generation from bubble diagrams with the rectangular approximation and coordinate clustering methods.

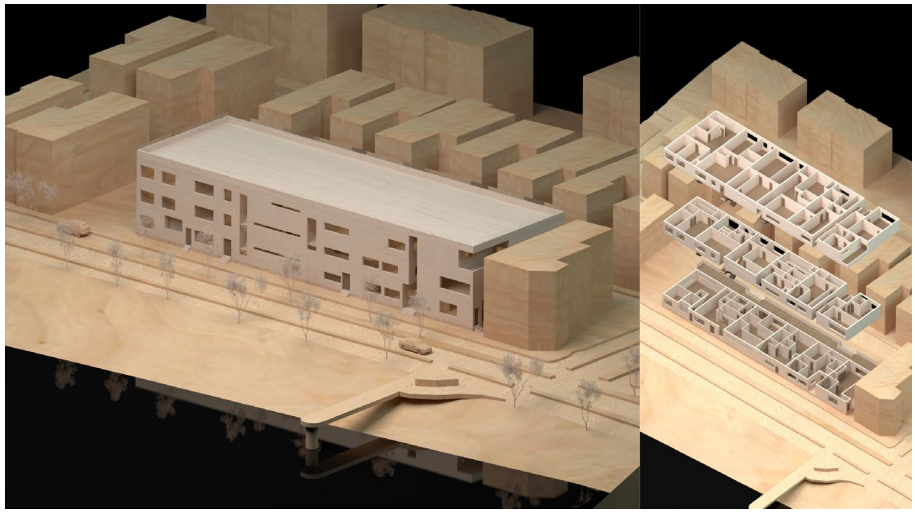

Figure 3. Axonometric models of a row house designed in the B2F exercise.

4. Analysis

4.1. DESIGN

In this section, we examine two design cases from the students' work (Figure 4) and review interesting observations and challenges that they faced during the exercise. As we conducted this study over two consecutive semesters, each case is selected from one of the two cohorts of the class.

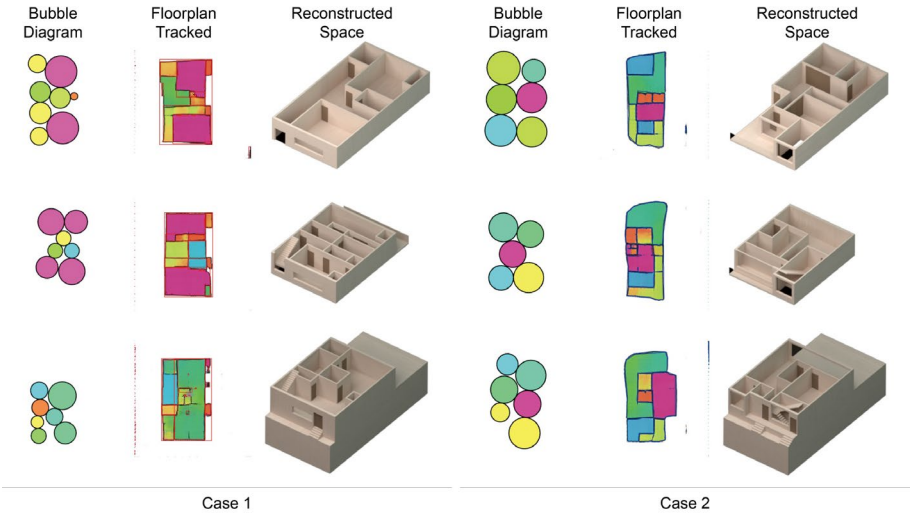

Figure 4. Two design cases from the Bubble2Floor exercise with different post-processing methods.

In case 1, students used the rectangular approximation method to reconstruct floor diagram images to 3D models. The design has an entrance facing the main street, one large living room on the first floor, four bedrooms on the second and third floors, and stairs at the same location of each floor. In case 2, students used the point clustering method for reconstruction. The design has two main entrances, one for a studio on the first floor and another for a two-bedroom apartment on the second and third floors. The apartment has two balconies facing the main street on the second and third floors.

In both cases, the translation between bubble to floor diagram works according to colour codes and relationships. For example, the bubble diagram for the second floor in case 1 has two large magenta discs at the top and bottom of the image, bracing small, partitioned spaces corresponding to smaller discs with yellow, green, and light blue colours. The adjacencies between discs are preserved in the respective floor plan.

We can find some graphical differences between the cases. The floor plan diagrams in case 1 seem to have more vivid edges and sharper corners, compared to case 2, which has rounded corners. Students in case 1 and 2 trained their Pix2Pix model for 200 and 500 epochs respectively. With 500 epochs, the model seems to overfit the training data and does not generalize well to the new bubble diagrams, which results in irregular floor plan edges.

The translation from floor plans into 3D models requires manual adjustments due

to (a) the different scale between the apartments in the dataset and the target footprint of B2F, and (b) the exploration of spatial patterns that were not well-represented in the training set. Students approached this step in different ways. In case 1, students tried to keep the original information of the extracted partition from floor plan diagrams as much as they could. Thereby, they removed the unintended programmatic elements that appeared on the floor plan diagrams but kept the main organisation from the bubble diagrams. Meanwhile, in case 2, students manually edited the synthesised floor plans to reinforce the practicality of the layout.

Students also had different stances on the DL method. In case 1, they expected an almost fully automated process—i.e., if users input a bubble diagram, the pipeline should generate a functional layout. As a result, they expressed their frustration as they had to play with different steps of the pipeline. Conversely, in case 2, students acknowledged the imprecision and the limitations of the workflow as part of the exercise, so they opted for exploring emerging and unexpected architectural designs. For instance, they subverted the original bubble diagram representation by adding overlapping discs to generate courtyards in the floor plan (Figure 5).

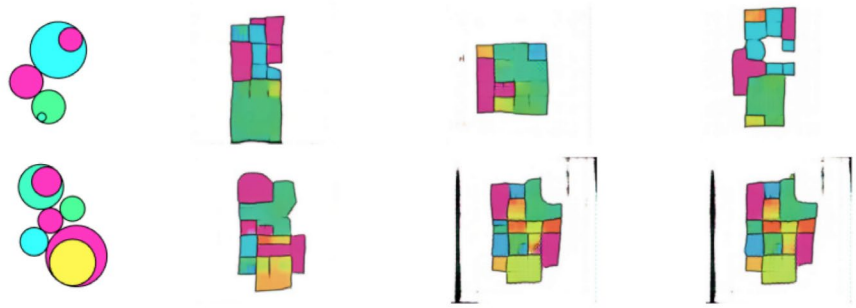

Figure 5. Examples of overlapping discs to create courtyards in a floor plan diagram.

We observed that the different approaches between case 1 and 2 were aligned with the pedagogical decisions made by the instructors. In the first semester, we guided students to individually train their models and provided a Python script to apply rectangular approximation. Rectangular approximation provides a limited level of control for the designer during post-processing and limits the floor plan configurations to mosaics of axis-aligned rectangles. In contrast, in the second semester, we allowed students to collectively train the models and provided a script that relies on coordinate clustering to post-process the layout. These enabled students to explore more expressive and varied floor plan configurations from the same floor plan image synthesised by the Pix2Pix model.

4.2. DATA REPRESENTATION AND SYNTHESIS

Currently, the type of representation and the dataset in the exercise were defined by the instructors before students developed their own projects. This approach was well-received by students as it enabled them to concentrate on learning and using the B2F

pipeline in the three-week span of this module. However, this choice resulted in some bottlenecks for design exploration, such as in the example of floorplan with courtyards (Figure 5). Ideally, the students should be able to create their own dataset and incorporate other forms of representation required by their design intentions, such as graphs, rectangular mosaics, etc.

4.3. TECHNICAL KNOWLEDGE AND DESIGN KNOWLEDGE

A programming background was a prerequisite for this course, which enabled us to introduce fundamentals of DL and hands-on tutorials with PyTorch as DL framework. However, we minimised the discussion on in-depth concepts related to the DL models used in B2F. As a result, while the students could use the pipeline to produce interesting designs, they faced technical issues that they could not address by themselves.

During this exercise, we observed signs of mode collapse in the trained models. Mode collapse is a phenomenon where a GAN generator exploits a part of the dataset distribution, constraining its output to similar results. For instance, the model keeps generating orange rooms even if the input bubble diagram has no orange discs (Figure 6). The second problem was the generation of images with blurry edges and mixed colours, which complicates the extraction of floor plan geometry. This issue might be a result of various factors, such as the small size of the training dataset, the choice of loss function, or the hyper-parameter tuning.

Typically, the target audience of this course did not possess in-depth technical knowledge of DL, which is essential to understand how to overcome training problems. A trivial solution would be to extend the technical aspect of the module to address these methods so students could solve common DL problems and even evaluate the performance of multiple models for floor plan generation. However, this can be challenging in a course designed for architecture students. Another alternative is to embrace students from data science, engineering, and computer science to form interdisciplinary teams, which has been successfully tested in other courses at CMU, such as Art and Machine Learning (E. Kang et al., 2018).

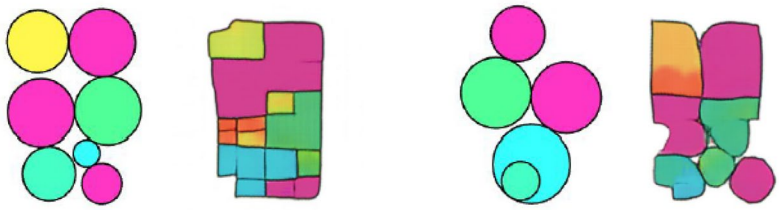

Figure 6. Two examples of mode collapse in training.

4.4. INTERACTION AND INTEROPERABILITY

The use of Jupyter Notebooks allowed students to interactively run and inspect different functions and components of the pipeline, which includes the logic of the GAN model and learning algorithm. However, we observed issues in interoperability—the connection and exchange of information between the

development environment (Jupyter Notebooks) and the 3D modelling tools (Rhino 6/Grasshopper). These interoperability issues hinder the design process, as switching between these platforms during data generation, training, and inference phases is time-consuming and distracting.

4.5. COMPUTING PLATFORMS AND PERFORMANCE

We used Jupyter Notebooks on Google Colab as an interactive and cloud-based development platform for B2F. On the one hand, this provided the same development environment and computation power for all students with close-to-no setup time across different sections of this course. On the other hand, this decision came at a cost; the free tier of Colab service relies on limited shared computing resources, which significantly slowed down the training and inference processes. In this setting, training our model takes over 10 hours, and the inference step takes a few seconds. This hampers students' capacity to investigate the affordances of the GAN model and, consequently, to take informed decisions and explore design alternatives. On higher-end GPUs, such as an RTX A6000, training the model with a relatively small data set for 100 epochs takes less than 20 minutes, and the inference step is executed in real time; however, this would be expensive for pedagogical purposes.

5. Conclusion and Discussion

The pedagogical experience with B2F establishes a deliberate position for the architect and DL. Instead of being end-users of a black-box design tool, the students are engaged in the different stages of B2F, including (a) data synthesis, (b) training the model, (c) translating the model output into proper design representations, and (d) human-AI interaction to explore design alternatives. Besides, DL is not only treated as a novel source for spatial sensibilities but also as a tool to explore solutions subject to real design, software, and hardware constraints.

By situating designers as active agents in the customisation and deployment of a pipeline for housing design, this experience revealed opportunities, frictions, and challenges for future practises based on DL. It is a step towards making DL accessible for architecture students to explore different design problems. It aims at providing the next generation of designers with the relevant skills and access to expert jobs and entrepreneurship in areas of AEC that will be drastically affected by AI. Moreover, it promotes the protagonist role of the profession in the technological development of the AEC industry and in the exploration of innovative design methods to pressing issues in design, such as the design of sustainable, safe, and affordable housing.

Acknowledgements

We would like to express our gratitude to Prof. Ramesh Krishnamurti, Prof. Daniel Cardoso Llach and Prof. Omar Khan for their support in the ideation and development of the course. We would also like to thank Prof. Krishnamurti for reviewing the final version of this document. Finally, we would like to thank our students from the courses Learning Matters, Exploring Artificial Intelligence in Architecture and Design (Spring 2021), and Inquiries into Machine Learning & Design (Fall 2021).

References

ACADIA Workshops. (2020). *ACADIA 2020: Distributed Proximities*. https://2020.acadia.org/workshops.html

ACADIA Workshops. (2021). *ACADIA 2021: Realignments: Towards Critical Computation*. https://2021.acadia.org/workshops/

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial nets. *Proceedings of NIPS: Advances in Neural Information Processing Systems*, 2672–2680.

Goodfellow, I., Yoshua, B., & Aaron, C. (2016). *Introduction to Deep Learning*. MIT Press. http://www.deeplearningbook.org

Isola, P., Zhu, J.-Y., Zhou, T., & Efros, A. A. (2017). Image-to-Image Translation with Conditional Adversarial Networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1125–1134).

Joyce, S., & Nazim, I. (2021). Limits to Applied ML in Planning and Architecture: Understanding and defining extents and capabilities. Towards a New, Configurable Architecture. *Proceedings of the 39th eCAADe Conference* (pp. 243–252).

Kang, E., Poczons, B., & Dinu, J. (2018). *Art and Machine Learning*. *ArtML2018*. Retrieved from https://sites.google.com/site/artml2018/

Kang, H., & Jha, A. (2018, July 23). *Pytorch Implementation of Pix2Pix for Various Datasets*. GitHub. Retrieved from https://github.com/znxlwm/pytorch-pix2pix

Liu, C., Wu, J., Kohli, P., & Furukawa, Y. (2017). Raster-To-Vector: Revisiting Floorplan Transformation. *Proceedings of the IEEE International Conference on Computer Vision* (pp. 2195–2203).

Nauata, N., Chang, K.-H., Cheng, C.-Y., Mori, G., & Furukawa, Y. (2020). House-GAN: Relational Generative Adversarial Networks for Graph-Constrained House Layout Generation. In A. Vedaldi, H. Bischof, T. Brox, & J.-M. Frahm (Eds.), *Proceedings of the European Conference on Computer Vision (ECCV)* (pp. 162–177). Springer International Publishing. https://doi.org/10.1007/978-3-030-58452-8_10

Nauata, N., Hosseini, S., Chang, K.-H., Chu, H., Cheng, C.-Y., & Furukawa, Y. (2021). House-GAN++: Generative Adversarial Layout Refinement Network towards Intelligent Computational Agent for Professional Architects. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 13627–13636). https://doi.org/10.1109/CVPR46437.2021.01342

Nourian, P., Rezvani, S., & Sariyildiz, S. (2013). A Syntactic Architectural Design Methodology: Integrating real-time Space Syntax analysis in a configurative architectural design process. In Y. O. Kim, H. T. Park, & K. W. Seo (Eds.), Proceedings of *the 9th International Space Syntax Symposium*.

Sg2018 workshops. (2018). [Conference]. *Smart Geometry*. https://www.smartgeometry.org/sg2018workshops

Tian, R., Guida, G., & Kim, D. (2021). Machine Intelligence in Architecture 2.0: The Architecture Turing Test [Education]. *Digital Futures*. https://www.digitalfutures.world/workshops/74.html

Weinzapfel, G., & Negroponte, N. (1976). Architecture-by-yourself: An experiment with computer graphics for house design. *Proceedings of the 3rd Annual Conference on Computer Graphics and Interactive Techniques*, 10, 74–78. https://doi.org/10.1145/563274.563290