ITU A|Z • Vol 17 No 2 • July 2020 • 185-198

GAN as a generative architectural plan layout tool: A case study for training DCGAN with Palladian Plans and evaluation of DCGAN outputs

Can UZUN[1], Meryem Birgül ÇOLAKOĞLU[2], Arda İNCEOĞLU[3]
[1] uzunc@itu.edu.tr • Architectural Design Computing, Graduate School of Science Engineering and Technology, Istanbul Technical University, Istanbul, Turkey
[2] colakoglumer@itu.edu.tr • Department of Architecture, Faculty of Architecture, Istanbul Technical University, Istanbul, Turkey
[3] inceoglua@itu.edu.tr • Computer Engineering, Graduate School of Science Engineering and Technology, Istanbul Technical University, Istanbul, Turkey

Received: October 2019 • Final Acceptance: February 2020

Abstract

This study aims to produce Andrea Palladio's architectural plan schemes autonomously with generative adversarial networks(GAN), and to evaluate the plan drawing productions of GAN as a generative plan layout tool. GAN is a class of deep neural nets which is a generative model. In deep learning models, repetitive processes can be automated. Architectural drawing is a repetitive process in the act of architecture and plan drawing process can be made automated. For the automation of plan production system we used deep convolutional generative adversarial network (DCGAN) which is a subset of GAN models. And we evaluated the outputs of the DCGAN Palladian Plan scheme productions. Results show that not geometric similarities (shapes), but probabilistic models are at the centre of the generative system of GAN. For this reason, it should be kept in mind that while GAN algorithms are used as a generative system, they will produce statistically close visual models rather than geometrically close models. Nonetheless, GAN can generate both statistically and geometrically close models to the dataset. In first section we introduced a brief description about the place of the drawing in architecture field and future foresight of architecture drawings. In the second section, we gave detailed information about the literature on autonomous plan drawing systems. In the following sections, we explained the methodology of this study and the process of creating the plan drawing dataset, the algorithm training procedure, at the end we evaluated the generated plan schemes with rapid scene categorization and Frechet inception score.

Keywords

Andrea Palladio, Artificial Intelligence (AI), GAN, Plan generator, Generative system.

doi: 10.5505/itujfa.2020.54037

1. Introduction

Architectural drawing is the most repetitive process in the discipline of architecture. During the 2000s BC, a Sumerian plan was sculpted in clay (Donald, 1962). With this drawing, we can say that architectural drawings have a history of at least 4000 years. Now, creating architectural drawings manually could be defined as an archaic action. Architectural plans are a subset of architectural drawings. Architects draw plans based on spatial organization decisions, such as the location of the building, the size of the space, the relation between the spaces, etc. Plan production is a repetitive drawing process where spatial organization decisions are made.

An architect can draw an architectural plan without thinking of any cognitive aspects of the design. However, behind the drawing action, a highly detailed and sophisticated process is unfolding in the mind. Formally describing these design processes in order to train an algorithm would not be possible or efficient. This falls into the artificial intelligence (AI) problem space. Goodfellow et al. (2016) states that the real challenge for AI is being able to solve tasks that people can do easily but formally have difficulty explaining. Human beings solve such problems intuitively. It is quite difficult to formally describe the act of designing in architecture, too. For autonomous drawing processes, instead of trying to formally explain the design, an algorithm can derive meaningful solutions through drawing datasets. Today, with the concept of big data, algorithms can evaluate data, and deep learning algorithms can be trained without needing to formally describe processes.

The repetitive process in the designing and drawing of architectural plans can be produced autonomously using the GAN model, a deep learning algorithm. As such, we aimed to evaluate the GAN algorithm as a generative plan design tool in architecture. As a cased study, we focused on the autonomous generation of Andrea Palladio's villa plans with DCGAN, a subset of the GAN algorithm. Studies on the autonomous production of plan drawing date back to the 1960s. Although the methods used vary, we can see from the literature that deep learning algorithms are the most promising tools for autonomous plan drawing as a generative design tool.

2. Studies on autonomous architecture plan layout generators

Studies on the autonomous production of plans in architectural design started in the 1960s. Grason (1971) stated that in the mid-60s, the linear graph technique was an important instrument for the description of a plan. In the 1970s, autonomous plan generation was realized by determining the geometries to be used in the design and the relationships of those geometries. It was also possible to derive plans using the shape grammar theory that emerged in the 1970s. In addition, genetic algorithms were tested in the autonomous generation system of a plan. There are also studies on the production of plans using traditional machine learning techniques. Today, deep learning algorithms have been used in studies on the autonomous production of plans.

Krejcirik (1969) developed his work on the optimal placement of the plan plane with computer-aided design. Weinzapfel et al. (1971) defined spaces as geometric entities and used the relationships between these geometries as design constraints to produce spatial arrangements in 3-dimensional computer-aided design process. Levin (1964) and Grason (1971) used graph theory for spatial arrangement in plans. Eastman (1973) proposed a system called General Space Planner (GSP) for autonomous spatial planning in two dimensions. According to Eastman, the goal in autonomous spatial planning software is not to completely remove the human from the design process, but to automate non-creative and repetitive processes in spatial planning. Eastman identified four variables for an autonomous space design system: space, design units, relations between design units, and operators to manipulate these relations. Operators manipulating the design are not well-defined. Therein lies the potential for creativity

in the design process. Furthermore, the relations between the design units are defined as Boolean functions. Eastman emphasized that the challenge in spatial planning was to produce an infinite number of orientation and positioning possibilities for design units. Finally, he emphasized that *the problem of autonomous spatial planning is a problem of the AI domain.*

Stiny & Mitchell (1978) aimed to organize Palladio's plan grammar rules into a modern and productive system. Using the grammar of Palladio's plan system, the Villa Malcontenta plan was produced (Stiny & Mitchell, 1978). In this study, Stiny and Mitchell did not produce an autonomous plan of Palladio. However, they were able to create the necessary algorithm for the autonomous generation of Palladian plans. Koning & Eizenberg (1981) contributed the development of autonomous processes in architectural design using Frank Lloyd Wright's cottage grammar rules. Many authors (Duarte, 2005; Colakoglu, 2005) used the productive system of shape grammar to produce architectural layouts.

Ahmad et al. (2004) used genetic algorithms in their studies to use space efficiently in plans. They minimized the boundary line of the plan by searching for the optimum solution with genetic algorithms. Rojas and Torres (2006) used genetic algorithms to design bank office plans. Dalgicet al. (2017) stated that the positioning of shelves within a store is an important point for product sales. Therefore, considering the visibility problem of shelves in stores, they designed a plan generator system that works with genetic algorithms. As a result of these studies, Dalgic et al. (2017) stated that genetic algorithms are a useful technique in creating a plan to produce many possible design proposals quickly and finding the optimum solution to a design problem. Nagy et al. (2017) produced a 45-dimensional design problem space with 15 neighbourhood relationships and three parameters to regulate these neighbourhood relationships. Using this problem space, they developed a system that can find design solutions using a productive system. The system they produced works semi-autonomously, and the design is developed by the designer's orientation and selection of the algorithm. The Multi-Objective Genetic Algorithm (MOGA) is used in the system developed for semi-autonomous design. Nagy et al. (2017) stated that thanks to MOGA, instead of dealing with a single problem in the design problem space, the designer can deal with multidimensional problems in a large problem space and achieve optimal results. This has the power to radically change the act of design. In this way, problems that can be overlooked in the design process can be solved.

The plan production software, Finch, developed with the parametric design tool, Grasshopper, can make autonomous decisions about the spatial arrangement, square meters, and furnishing in plans. In this software, an increase or decrease in the total square meters of the plan redesigns the furnishing and the spatial arrangement autonomously (Ravenscroft, 2019).

Nowadays, the development of deep learning technology is becoming established in the field of autonomous plan production systems. In their study, Huang and Zheng (2018) were able to design plans using GAN networks. GANs are generative deep neural networks. GAN networks are strong enough to generate visual data. The GAN system that Huang and Zheng used was a specialized GAN network called Pix2pixHD. This network can derive visual information from labels. For example, if the label is the kitchen label, drawings of the kitchen plan are generated on that label. In the dataset used by Huang and Zheng, features (furnishing, walls, and bearing systems) are produced autonomously on plan drawings using the GAN network. Chaillou's (2019) work is another study on plan generation with GAN networks. In this study, GAN networks generate plans in three steps. In the first step, the total shape of the plan is generated. In the second step, GAN produces interior partitioning and spatial arrangement. The third step ends with furnishing and plan production. This study reveals the power of GAN networks in autonomous plan gener-

ation. In Chaillou' s work, the resulting plans are very clear, right down to the details of the plan, the openings in the walls and the spatial organization (Chaillou, 2019)

3. Methodology

For the DCGAN training process, we conducted two experiment using two different datasets. The first dataset contained original villa plans produced by architect Andrea Palladio (16th century). The second dataset was created with Palladian plans generated using Palladian grammar rules. The Palladian grammar rule set for generating plans is defined by Stiny & Mitchell (1978).By referring to their bilateral symmetry; Palladian plans can be easily evaluated. Due to their simple layouts and reference for evaluation, these villa plan drawings were used as cases for testing DCGAN plan generation.

The purpose of the study was to examine the effectiveness of DCGAN networks in the design process of Palladian plans as a generative plan design tool. We evaluate the DCGAN outputs using qualitative (rapid scene categorization) and quantitative (Frechet Inception Distance) methods. For rapid scene categorization, we used both Palladian grammar rules and space syntax analysis.

4. Case study: GAN training with the Palladian Villa Plan Dataset

The case study was carried out through two experiments. The first was DCGAN training with an original Andrea Palladio villa plan set. The second was DCGAN training with plans produced according to Palladian grammar rules. The first experiment outputs were noisy and not clean enough to evaluate the DCGAN outputs. Therefore, we decided to repeat the training with a cleaner dataset. In the second training, Palladian grammar rules were used to create a cleaner Palladian plan dataset. For both DCGAN training experiments, we used the same DCGAN architecture and hyperparameters.

4.1. DCGAN architecture

This section explains the GAN algorithm training using the Palladian villa plans. In order to generate images, DCGAN, a subset of GAN algorithms, has been proposed (Radford; 2015). The following is a detailed description of DCGAN architecture.

GAN (Goodfellow (2016)) is special type of network that learns the distribution of the input data$x\sim p_{data}$ and generates new samples$x\sim p_{model}$ from learned distribution. The overall network is composed of two sub-networks, namely generator (g) and discriminator (d) networks. The generator takes a random noise vector$z\sim p$, where pis a Gaussian or Uniform distribution, as an input and generates a new sample$x=g(z;\ \theta_g)$. The discriminator takes both real and fake (i.e. generated) data as input and returns a probability value$p= d(x;\ \theta_d)$, indicating whether x is a real or fake sample.θ_g and θ_d are learnable parameters of the generator and discriminator networks, respectively. There is competition between these two networks. While the generative network tries to generate a sample such that the discriminator can no longer identify whether it is a fake or real sample, the discriminator gets better at discriminating between real and generated samples.

The objective is to find:

$$d^* = arg\max_d v(\theta_d, \theta_g)$$
$$g^* = arg\min_g v(\theta_d, \theta_g)$$

where:

$$v(\theta_d, \theta_g) = E_{x\sim p_{data}} \log d(x) + E_{x\sim p_{model}} \log(1 - d(x))$$
$$v(\theta_d, \theta_g) = E_{x\sim p_{data}} \log d(x) + E_{z\sim p} \log(1 - d(g(z)))$$

DCGAN Architecture:

In DCGAN architecture, the generator and discriminator networks are composed of convolutional layers. We adopted a similar architecture (Figure 1) to that proposed by Radford et. al(2015). The generator network consists of three convolutional layers, each with 128, 128 and 64 filters, respectively. There are batch normalization layers and Leaky ReLU (α=0.2) is used as nonlinearity. The discriminator network is composed of four convolutional blocks. In each block, the convolutional layer is followed by Leaky ReLU

(α=0.2), dropout (p=0.25) and batch normalization layers. The convolutional layers have 16, 32, 64 and 128 filters, respectively.

Training Details:

We trained for 10,000 epochs with a batch size of 32. We used Adam optimisation with the following hyperparameters: learning rate: 0.0002, beta1:0.5, and beta2: 0.999. At the pre-processing stage, all images were converted to grayscale and resized. In order to stabilize training, we applied label smoothing (0.1 for real image labels and 0.9 for fake image labels).Different image sizes were examined. Increasing the image size from 128x128 to 256x256 pixels increased the resolution in terms of detail in generated images. However, the level of detail remained unchanged when increasing the image size.

First experiment: GAN network training with original Andrea Palladio Villa Plans Dataset

Preparation of Dataset

We collected Andrea Palladio's villa plans for the dataset. During the data collection process, we searched for Palladio's villa plans in the library. We carried out the data collection process by scanning the plans in books about Palladio. In the scanned plans, we identified and removed repeated plans. We found a total of 125 plans by scanning eight books (Giaconi, G., Williams, K., & Palladio, A. (2003)., Foscari, A., Canal, B., & Façade, GT (2010). , Hemsoll, D. (2016)., Boucher, B. (1998)., Puppi, L. (1973)., Puppi, L., Codato, P., Palladio, A., & Venchierutti, M. (2005) Rykwert, J., & Schezen, R. (1999)., Wundram, M., Marton, P., & Pape, T. (1993)). All 125 plans are shown in Figure 2. We scrubbed the plans of redundant data (measurements, arrows, text, etc.) with photoshop, so that only outer walls, inner walls and stairs were left. We converted the colours of the plan to black and white, but the colour channel of the plan images was based on RGB values. In the algorithm training process, we converted the colour channel of the dataset from RGB to grayscale.

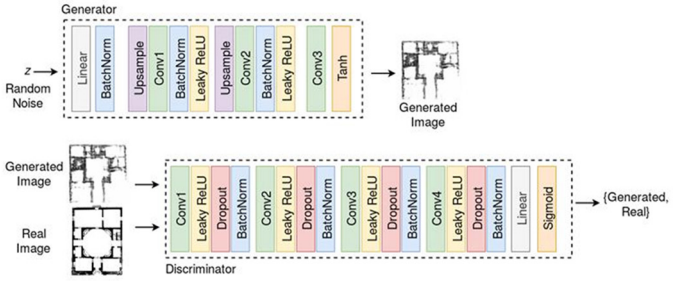

Figure 1. *DCGAN Architecture.*

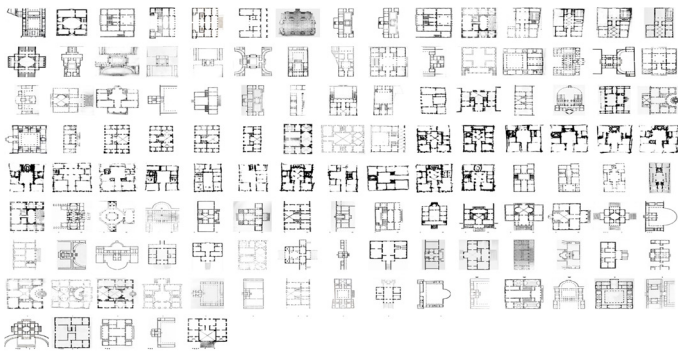

Figure 2. *Andrea Palladio Original Plan Schema Dataset (125 Plan Schemas).*

Since the dataset consists of 125 images, we thought that the dataset may be insufficient. Primarily we used Euclidean transformations (rotation, translation, symmetry operations) to increase the amount of data. After the data augmentation process, we obtained 1,000 images for the dataset. After training the DCGAN model, we observed that the training was not stable, given the loss values of the generator function. The unstable training process was related to either the dataset or the GAN model. However, we know that the DCGAN model is a stable training model in GAN networks. So, we focused on the dataset as the problem. With the data augmentation methods that we used, we may have generated outlier data that confused the training process. Therefore, we decided to use only the original Palladian plan data for the dataset.

Outputs of DCGAN

For the DCGAN training session, we used the same architecture and hyperparameters described in the DCGAN Architecture section. For most of the DCGAN plan productions, we observed that DCGAN tried to generate bilateral symmetric plans (Figure 3). In addition, we saw staircases on some generated samples. However, the outputs of DCGAN were too noisy to evaluate DCGAN's plan generation. The reason for noisy samples was the fact that the original plan dataset was not clean enough. So, we decided to

generate a new, clean Palladian plan dataset. While creating the new dataset, we used Palladian grammar.

Second experiment: DCGAN training with plans derived using Palladian grammar rules

In this experiment, we trained the DCGAN model with a dataset produced using Palladian grammar rules.

Preparation of dataset

Stiny & Mitchell (1978) defined the Palladian grammar rules for Palladian villa plans. SWAP Architekten has developed a web-based shape grammar interpreter (SGI) based on the rules of Palladian grammar (Grasl, n.d.). This web-based SGI Palladian grammar, called GRAPE, can generate Palladian plans. For the second experiment, we generated the Palladian plan dataset using the GRAPE software.

The plan generation process takes place through the definition of a series of rules. The rules were:

- Create a grid from square units (3x3, 3x4, 3x5, 5x3, 3x7 grids, with 30 plans for each)
- Define the walls for the plan
- Decide which square units in the grid to merge with which square units, according to Palladian grammar rules (combining the square units in the grid to generate T, I and + forms on the grid, joining the square units in the grid on the east-west and north-south axes)
-Loggia addition
-Portico addition
-Door and window voids defined on the east-west and north-south axes

150 Palladian plans were generated using the web-based GRAPE SGI in accordance with the Palladian grammar rules defined above (Figure 4).

Outputs of DCGAN

For this experiment, we used the same architecture and hyperparameters described in the DCGAN *Architecture* section. In addition to the same DCGAN architecture, label smoothing was used in this experiment. We observed that label smoothing decreased the loss value of the generator and created a more robust training process. The values of the generator and discriminator loss functions during the training process are shown in Figure 5. With label smoothing we obtained lower loss values for the generator function.

In the first epochs of the training session, the algorithm did not learn very well and could not generalize through the Palladian plan. However, after the 600th epoch, DCGAN started generating reasonable plans (Figure 6). At the end of the training sessions, we decided that the quality of the productions and loss value of the generator was good enough for the evaluation of the DCGAN Palladian plan generation process. Thus, we ended the training process at the 10000th epoc.

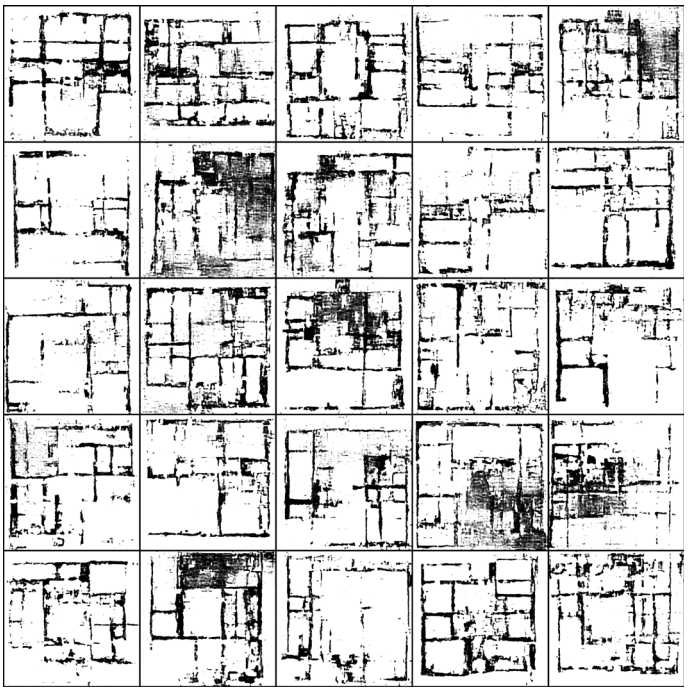

Figure 3. *GAN Plan Scheme Production Result with Original Palladio Plan Dataset.*

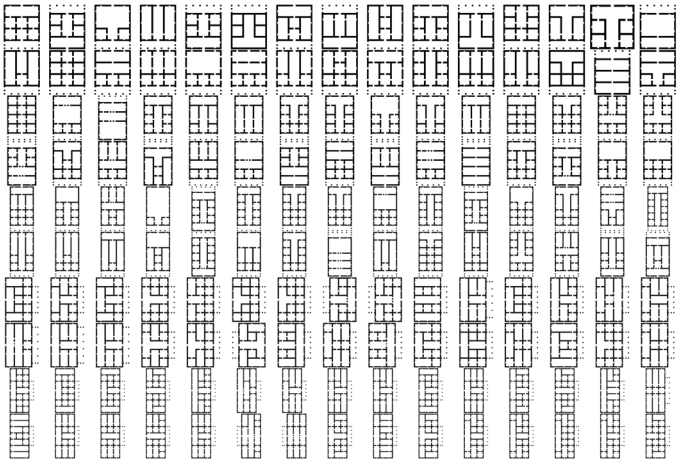

Figure 4. *The Dataset that is generated with Palladian Grammar Interpreter, GRAPE-SGI (150 Plan Schemas).*

5. Evaluation of DCGAN production in the second experiment

In this section, we evaluate the outputs of the DCGAN. The GAN algorithm has no standard evaluation technique, but there are many different techniques which can be used to evaluate the performance of GAN algorithms. GAN evaluation techniques are divided into two groups: qualitative and quantitative methods. In this paper we used both methods to understand the behaviour of GAN.

Qualitative methods of GAN evaluation include nearest neighbours, rapid scene categorization, rating and preference judgment, evaluating mode drop and mode collapse, investigating and visualization of internals of network (Borji, 2019).

Beside qualitative methods, there are many quantitative methods that measure the efficiency of GAN algorithm productions. These include average log-likelihood, coverage metrics, inception score (IS), Frechet Inception Score (FIS), mode score, and Frechet Inception Distance (FID). Some of these techniques measure the resolution quality of the image generated by GAN, while others look for similarity between the training dataset and the outputs of GAN through statistical calculations.

For the qualitative evaluation method, we used rapid scene categorization; for the quantitative evaluation method, we used FID.

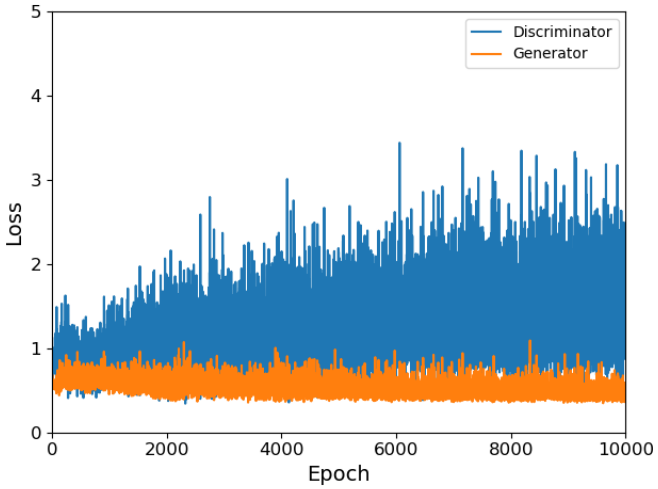

Figure 5. *Loss Values of DCGAN with LabelSmoothing (0.1 for real image labels and 0.9 for fake image labels).*

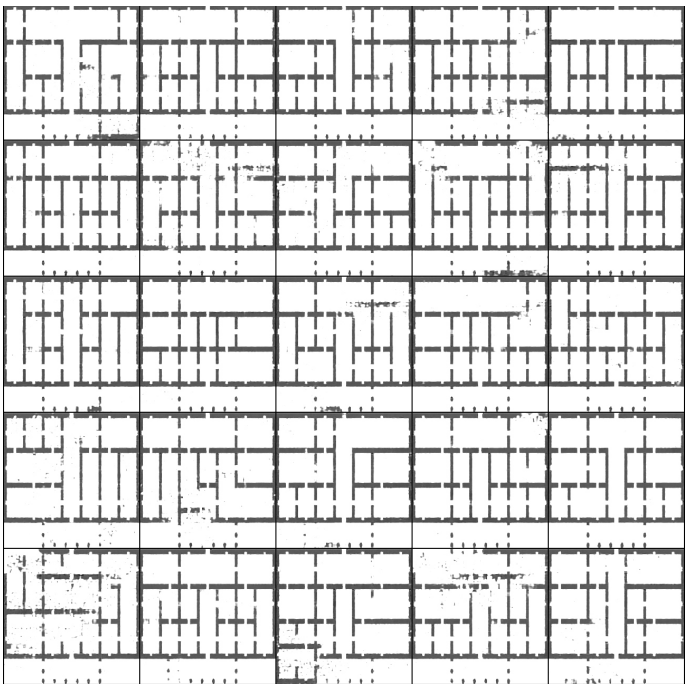

Figure 6. *GAN Plan Scheme Production Result with GRAPE SGI Palladio Plan Scheme Dataset.*

5.1. Evaluation through rapid scene categorization

We compared the DCGAN outputs with the dataset used to train the DCGAN model. We made comparisons on the basis of the Palladian grammar rules and space syntax values. As a result of this comparison, we evaluated how effectively the GAN network generated Palladian plans.

Rapid scene categorization is a visual examination method based on human perception (Borji, 2019). Through this evaluation, the observer classifies the true and false outputs of GAN. Since a human is the observer in the evaluation, this evaluation technique can be considered subjective and intuitive. GAN generates thousands of visuals. As such, the observer must work quickly yet still be careful while checking the visual outputs. Because of the large amount of visuals to be evaluated, some misclassification might occur in terms of true or false decisions. Thus, the reliability of this technique is variable. Nonetheless, rapid scene categorization is fast and useful in measuring the performance of GAN.

For rapid scene categorization, we used the Palladian grammar rules to be precise while labelling the outputs 'DCGAN(true)' or 'DCGAN(false)'. Palladian grammar rule checking is

based on the 'y axis' symmetry rule, grid merging decisions and portico generation rules. Among the samples generated by DCGAN, we found that very few of the samples did not have a portico.

As merely visual inspection is not reliable enough for evaluation, we analysed the true and false samples using space syntax analysis. Although space syntax gives quantitative result, this method also can be included in the rapid scene categorization since it still examines the shapes but without providing any statistical comparison result between the training dataset and DCGAN-generated samples.

Palladian grammar rule categorization

GRAPE SGI software can derive a Palladian plan in a semi-autonomous manner according to the Palladian grammar rules. According to Stiny and Mitchell (1978), the prominent feature in Palladian plans is *bilateral symmetry*. In most of Palladio's villa plan diagrams, bilateral symmetry appears. There is always a portico as well.

The Palladio Grammar rule starts with a grid decision. The grids designed by Palladio are mostly in 5x3 arrangements. In Palladian grammar, the grid begins with a unit square. This grid is augmented according to the grammatical rules from the east, west, north and south directions. The grid augmentation process is performed under bilateral symmetry. If a unit grid square is added to the east, another grid unit must be added to the west. This rule is not sought in the north-south direction. After the grid is completed, the Palladian rule generates spaces with these grid units. In the middle axis of the grid, the grid squares merge to form I, T, or + shapes on the plan. The grid can merge through the north-south direction but must obey the bilateral rule. A portico is created on the axis of bilateral symmetry and the entrance axis of the villa is defined. The door and window openings are defined on the north-south and east-west axes in accordance with bilateral symmetry. At the end of the grammar rule application process, the entrance door is created on the axis of bilateral symmetry on the portico wall.

Figure 7 shows the 25 GRAPE-SGI generated Palladian plans within the training dataset, 25 DCGAN generated plans which are in accordance with Palladian grammar rules by Stiny & Mitchell (1978) and 25 DCGAN generated plans which are not in accordance with Palladian grammar rules.

When we examined the DCGAN production, the majority of DCGAN-generated plans were not in accordance with bilateral symmetry. Among the 2,525 plans generated by DCGAN, we saw that only 63 plans complied with Palladian grammar (bilateral symmetry), and DCGAN generated the majority of the plans with a 7x3 grid layout. We trained the DCGAN with a dataset that included the same number of 3x3, 3x4, 3x5, 5x3 and 7x3 grid layouts, but we observed that DCGAN generated only 7x3 grid layouts. The main reason for the dominance of 7x3 layouts may be related to the probability distribution of the pixels in the dataset. This means that while training the DCGAN with different grid layouts, DCGAN learned

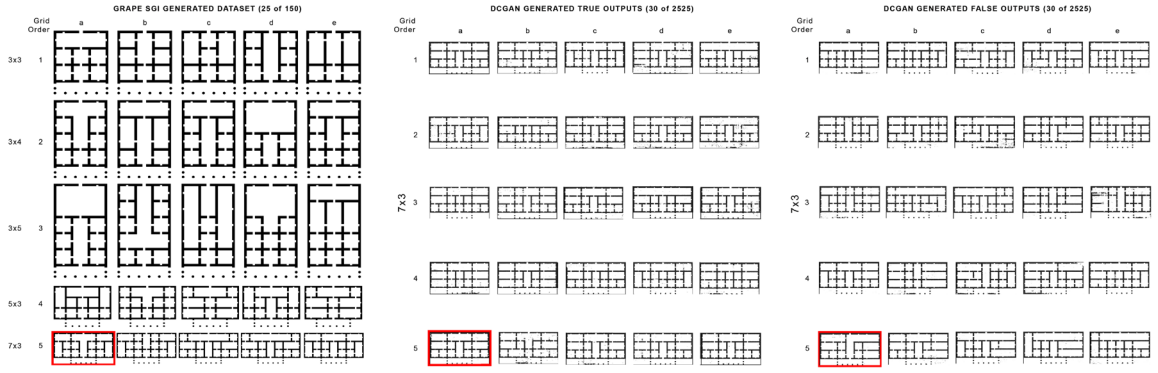

Figure 7. *25 Palladian Plan Schemes that is generated with Palladian SGI & 50 Palladian Plan Schemes that is generated with DCGAN(left=true, right=false).*

and accepted the 7x3 layout as a true probability.

In Figure 8, we selected one plan per group among GRAPE SGI, DCGAN(true) and DCGAN(false) plan generation results. We carried out reverse engineering to obtain each of the plans with Palladian grammar rules (Stiny & Mitchell, 1978) from scratch. In this way, we aimed to check the true and false outputs of DCGAN in terms of Palladian grammar rules. For the GRAPE $SGI_{(5,a)}$ and $DCGAN(true)_{(5,a)}$ plans, we were able to obtain the same plans. However, $DCGAN(false_{(5,a)}$ could only be derived with different rule sets not included in Palladian grammar rules.

In Figure 8, each of the three plans have the same 7x3 grid layout, so we applied the Palladian grammar rule to obtain a 7x3 grid layout. While generating our dataset we used the same steps in GRAPE SGI $plan_{(5,a)}$. The rules that we applied were: a *7x3 grid, east-west axes* grid merging, *north-south axes* grid merging, T-shape grid merging (this shape can be either *"T", "I" or* "+" shape as mentioned in the Preparation of Dataset section), creation of *window* and *door openings*, and finally addition of a *portico*. For the true output, we labelled the output as $DCGAN(true)_{(5,a)}$ and for this output, we were able to apply the same list of Palladian grammar rules as in the GRAPE SGI, with one exception. Instead of using the T-shape merging , here we used l-shape merging on the north-south middle axis. For the false outputs of DCGAN, we labelled the outputs as $DCGAN(false_{(5,a)}$.The rules applied for obtaining the $DCGAN(false)_{(5,a)}$ plan were not in accordance with Palladian grammar rules. The $DCGAN(false)_{(5,a)}$ plan does not have bilateral symmetry. The reason for this problem was that the grids were merged asymmetrically. In the Palladian grammar rule set there is no L-shape grid merging as it would spoil the bilateral symmetry. However, in the $DCGAN(false)_{(5,a)}$ plan, we were able to derive this plan by applying the L-shape merging rule.

By inspecting all the 2,525 plans generated by DCGAN using this methodology (Palladian grammar rules), we classified 63 of 2,525 plans as DCGAN(true), and the other 2,462 samples as DCGAN(false). Whether the plan was in compliance with the grammar rule, all the plans generated by DCGAN had the portico in front of the bilateral symmetry axis. The reason for this must be related to the dataset. All the data within the dataset had a portico and they are all the same. So DCGAN generated the same pixel values in the correct places.

From all these results we can deduce that DCGAN does not read the grammar rules, but it reads the probability distribution of the pixel values in the dataset. Plans generated in accordance with the Palladian grammar rules are just a subset of probability distribu-

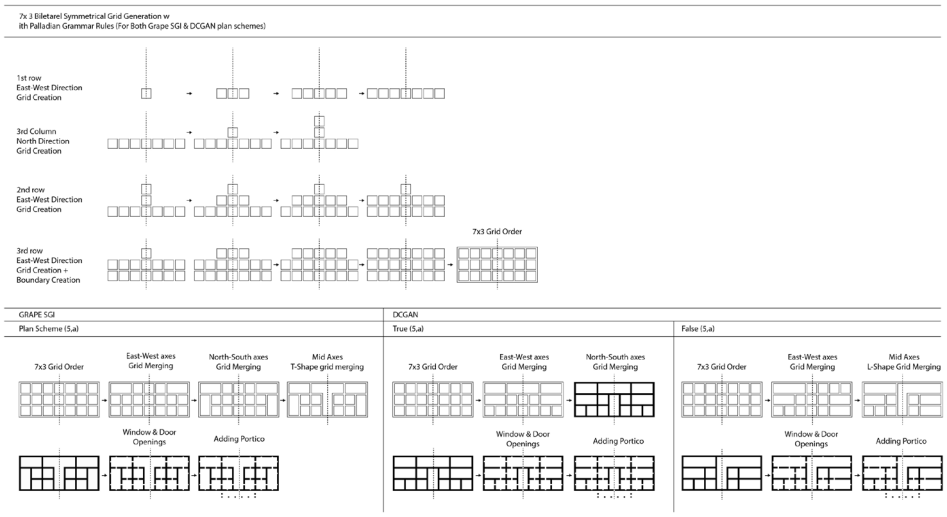

Figure 8. *Grammar Rules of one plan scheme per groups in Figure 7; GRAPE SGI(5,a), DCGAN(TRUE)(5,a) & DCGAN(FALSE)(5,a).*

tion of the pixel values in the dataset. Thus, we observed some true Palladian grammar rule plans, but we also observed many other possibilities which are not in accordance with the Palladian grammar rules.

Space syntax categorization

In the previous section we evaluated DCGAN generation using Palladian grammar rules with the rapid scene categorization technique. We classified the outputs as DCGAN(true) and DCGAN(false) subjectively, even though we used the grammar rules as a basis. That means we may have misinterpreted or missed out some features in the outputs. Although rapid scene categorization is a qualitative method and space syntax is a quantitative method, with the help of space syntax we were able to categorize the outputs not subjectively but objectively. As such, we incorporated the space syntax method into rapid scene categorization. In this way, we were able to evaluate the outputs more precisely than we did with Palladian grammar rules.

In this section, we compare the GRAPE SGI Palladian plans, DCGAN(true) and DCGAN(false) Palladian plans according to space syntax values. Space syntax is mostly used in urban analysis studies. However, it can also be used for analysis of spatial arrangement, visibility, space privacy, and integration of spaces on the architectural scale. Space syntax can be used to calculate the relationships between spaces within a system (Hillier & Stonor, 2010). In this study, we used the connectivity values of space syntax to evaluate the plans. Connectivity indicates how many spaces are connected to another space. There is a direct proportion between connectivity and integration values. A connectivity value gives the measures of the distances between all subspaces in a space and then it compares and calculates the farthest and the closest subspace to each subspace in the space. The resulting values for connectivity are represented by a heatmap, with red indicating the most integrated space, while blue defines the most isolated space. Furthermore, this value gives information about the circulation on the layout of the spaces.

We selected 25 GRAPE SGI-generated plans, 25 DCGAN(true) plans and 25 DCGAN(false) plans. The selected plans are the same as in the *Rapid Scene Categorization: Palladian Grammar Rules* section. Figure 9 shows the visualization of the connectivity values of each of the GRAPE SGI, DCGAN(true) and DCGAN(false) Palladian plans.

When we examined the connectivity values of the GRAPE SGI-generated 7x3 layout grid plan results, we observed that the axis parallel to the portico axes showed the largest connectivity value since the grid layout was 7x3. The corridor parallel to the portico is formed by 7 grid units. So, this corridor has more connection with the other spaces in the plan than the other spaces have. The second highest values were observed in the either entrance hall or the halls near the east and west walls of the plan. The lowest connectivity values were detected just east and west of the entrance hall. So these spaces in the 7x3 grid system are the deepest spaces in the plan.

The connectivity values of DCGAN(true) were almost the same as the GRAPE SGI results. However, the DCGAN(false) connectivity values were not the same, but similar to the GRAPE SGI results. This shows that whether the plan is DCGAN(true) or DCGAN(false), the space syntax values did not change significantly. The reason for this must be related to the grid rule (7x3) and similar door openings between spaces in both DCGAN(true) and DCGAN(false). Pixel probability distribution in the dataset may correlate with the space syntax values. So, although the DCGAN productions are not in accordance with Palladian grammar rules, the syntax values can still be similar to the syntax values of the dataset. In other words, DCGAN learned the structure of the Palladian plans through probability distribution. So, DCGAN(true) outputs are 100% accurate both in terms of Palladian grammar rules and space syntax values, and DCGAN(false) results are 100% false in terms of Palladian grammar rules but not 100% false in terms of space syntax values. We can deduce that DCGAN created almost similar data to the dataset.

5.2. Evaluation through the Frechet Inception Distance (FID)

In section 5.1 we evaluated the DCGAN outputs using the rapid scene categorization technique, which is a qualitative technique. In this section we use FID as a quantitative method to measure the performance of DCGAN. Since there is not a single and exact method to evaluate the performance of GAN algorithms, we used both qualitative and quantitative methods to understand the behaviour of DCGAN better.

With FID, we can calculate the distributions of embedding for images in the dataset and images generated by DCGAN. Embeddings are the vector representation of the variables in the data (Koehrsen, 2018). FID compares the distances between these vector representations. For FID calculation a special network, the Inception V3 network, was used. The distance was calculated using the Gaussian distributions of the data from the dataset and the data from DCGAN-generated images (Borji, 2019). The following function from Borji (2019) shows the FID

$$d^2 = ||mu_1 - mu_2||^2 + Tr(C_1 + C_2 - 2 * sqrt(C_1 * C_2))$$

score equation.

Features of real and generated data are defined with values mu1 and mu2, respectively, while C_1 and C_2 denote the covariance matrix for real and generated data (Brownlee, 2019). Covariance shows the relation between two variables of two different dataset. If the FID score is low, this means the dataset and the outputs are similar, but if the FID score is high it means generated outputs are different from the dataset.

By using this equation, we calculated the FID scores between the 150 images from the dataset and each of the150 outputs generated by DCGAN between 0th-600th, 2500th-3100th, 5000th-5600th, 7500th-8100thand 9400th-10000th epochs. Figure 10 shows the FID scores.

According to Figure 10, DCGAN improved in learning the Palladian plan dataset per iteration. The FID scores decrease per iteration, and DCGAN generated more similar results to the dataset. However, after around the 3000th epoch, the FID score did not change significantly, but minor changes were observable. From the outputs of DCGAN, we can see that the resolutions of the images improved per iteration, but the structure of the images did not change significantly after the 3000th epoch. If the DCGAN overfits, the FID score would be around 0. Thus, the FID not being 0 is a good thing. Comparison between the first epoch and the 3000th epoch shows that DCGAN learned the structure of Palladian plan dataset, so the FID score decreased dramatically until the 3000th epoch. Although most of the outputs of DCGAN were not in accordance with Palladian grammar rules, the FID scores show that DCGAN learned the probability distribution of pixel values on the plans and generated almost mathematically correct Palladian plan results.

6. Results and conclusion

In this study, we aimed to evaluate the effectiveness and the performance of the GAN algorithm as a generative system in architectural plan drawing. We choose Palladian plans as a case study for the GAN evaluation process. For the training sessions, we used DCGAN, a subset of GAN algorithms. We

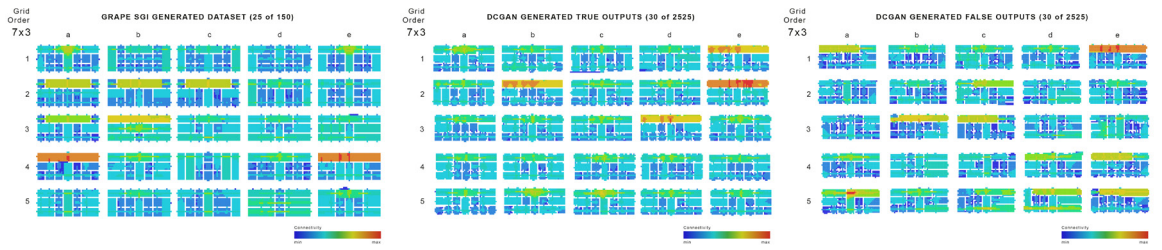

Figure 9. *Connectivity Values, Syntax Results of 25 Palladian Plan Schemes that is generated with GRAPE SGI & Syntax Results of 50 Palladian Plan Schemes (True & False) that is generated with DCGAN.*

used the default hyperparameters in DCGAN. We performed two experiments with the DCGAN algorithm. In the first experiment, we used 125 original Palladio villa plans to train the DCGAN algorithm. DCGAN results were noisy because the dataset was not sufficiently clean. For this reason, we derived 150 Palladian plans using Palladian grammar with GRAPE SGI software. These plans constituted the second dataset to be used in the training of DCGAN. We used the label smoothing method to optimize the DCGAN generator function's loss value. Before label smoothing, the value of the loss function of the generator function was high. After label smoothing, the loss value of the generator function decreased below 1. The training process was more stable with label smoothing. We evaluated both the datasets and found that the first dataset, which included 125 original Palladian plans, was too heterogeneous. Because of the heterogeneous dataset, during the training process of DCGAN, a high variance low bias problem occurred and the outputs of the DCGAN failed in producing reasonable results. The GRAPE SGI-generated dataset was homogenous so the training process of DCGAN was more stable using this dataset, and the results were of a better quality. We continued the evaluation process of DCGAN with the GRAPE SGI-generated dataset. We terminated the training process of the algorithm at the10000th epoch, since the loss value was sufficiently low. DCGAN generated 2525 Palladian plans with a 7x3 grid layout.

After the training process, we evaluated the generated plans to understand the performance and the effectiveness of DCGAN as a generative plan layout production tool. The first intuitive evaluation was that DCGAN somehow focused on the generation of 7x3 grid layout plans. The reason for this constituted an engineering research problem, so we simply evaluated these plans. We used three different evaluation methods. Two of the methods fall under the rapid scene categorization technique, a qualitative evaluation method of GAN outputs. These two methods were: categorization through Palladian grammar rules, and categorization through space syntax analysis. The third evaluation technique was a quantitative method: Frechet Inception Distance. FID is a method for calculating the probability distribution distance between the real and generated plans. So FID score gives a probabilistic result.

When we look at the results of Palladian grammar rule rapid scene categorization, only 63 plans out of 2525 generated plans were in accordance with the Palladian grammar rule. We classified these 63 plans as DCGAN(true). The others were labelled DCGAN(-false). These results show that DCGAN is not sufficiently effective for the training of Palladian grammar rules. To confirm these results, we conducted space syntax analysis, another rapid scene categorization. When we compared the Palladian grammar rules with the space syntax values, we saw similar syntax values to the dataset.

Upon evaluation of the FID results, we found that DCGAN improved at imitating the dataset, despite the fact they did not comply Palladian grammar rules. This means that almost correct results were generated by DCGAN mathematically, but does not mean that DCGAN can read the grammar rules. The reason for the DCGAN(true) generations may be that the results compatible with Palladian grammar rules were a subset of the probability distribution of the pixel values of each item of data in the dataset.

After the evaluation of the DCGAN

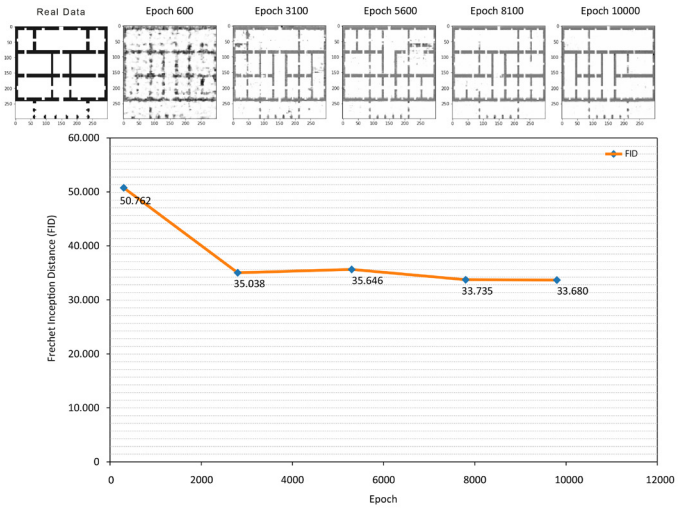

Figure 10. *FID Score Comparison.*

plan outputs, we can say that GAN algorithms can take part in generative systems in architecture for plan production. Dinçer et al. (2014) classified the generative design tools under five headings. These generative design tools are listed as shape grammars, genetic algorithms, L-Systems, cellular automata, and collective intelligence-swarm behaviour. We can add the GAN algorithms to this list as the sixth generative design tool. Not geometric similarities (shapes), but probabilistic models are at the centre of the generative system of GAN. For this reason, it should be kept in mind that while GAN algorithms are used as a generative system, they will produce statistically close visual models rather than geometrically close models. Nonetheless, GAN can generate both statistically and geometrically close models to the dataset. Therefore, GAN algorithms can take part in the class of generative design tools for plan generation.

References

Ahmad, A. R., Basir, O. A., Hassanein, K., & Imam, M. H. (2004). *Improved placement algorithm for layout optimization.* In Proc. of the 2nd Int'l Industrial Engineering Conf.(IIEC'04).

Boucher, B. (1998). *Andrea Palladio: the architect in his time.* Abbeville Press.

Borji, A. (2019). Pros and cons of GANevaluation measures. *Computer Vision and Image Understanding, 179*, 41-65.

Brownlee, J. (2019, October 10). *How to Implement the Frechet Inception Distance (FID) for Evaluating GANs.* Retrieved December 5, 2019, from https://machinelearningmastery.com/how-to-implement-the-frechet-inception-distance-fid-from-scratch/

Brock, A., Donahue, J., & Simonyan, K. (2018). *Large scale gan training for high fidelity natural image synthesis.* arXiv preprint arXiv:1809.11096.

Chaillou, S. (2019). *AI & Architecture.* Retrieved from https://towardsdatascience.com/ai-architecture-f9d78c6958e0

Colakoglu, B. (2005). Design by grammar: an interpretation and generation of vernacular hayat houses in contemporary context. *Environment and Planning B: Planning and Design,* 32(1), 141-149.

Dalgic, H. O., Bostanci, E., & Guzel, M. S. (2017). *Genetic Algorithm Based Floor Planning System.* arXiv preprint arXiv:1704.06016.

Dinçer, A. E., Çağdaş, G., & Tong, H. (2014). Toplu Konutların Ön Tasarımı İçin Üretken Bir Bilgisayar Modeli. *Megaron, 9*(2).

Donald, T. (1962). A Sumerian Plan in The John Rylands Library1. *Journal of Semitic Studies, 7*(2), 184-190.

Duarte, J. P. (2005). A discursive grammar for customizing mass housing: the case of Siza's houses at Malagueira. *Automation in Construction, 14*(2), 265-275.

Eastman, C. M. (1973). Automated space planning. *Artificial intelligence, 4*(1), 41-64.

Foscari, A., Canal, B., & Façade, G. T. (2010). *Andrea Palladio. Unbuilt Venice.* Baden: Lars Muller Publishers.

Generative adversarial network. (2019). Retrieved from https://en.wikipedia.org/wiki/Generative_adversarial_network

Giaconi, G., Williams, K., & Palladio, A. (2003). *The Villas of Palladio*. Princeton Architectural.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning.* MIT press.

Grasl, T. (n.d.). GRAPE For Web - Shape grammar interpreter. Retrieved from http://grape.swap-zt.com/App/PalladianGrammar

Grason, J. (1971, June). *An approach to computerized space planning using graph theory.* In Proceedings of the 8th Design automation workshop (pp. 170-178). ACM.

Hemsoll, D. (2016). *Palladian Design: The Good, the Bad and the Unexpected.*

Hillier, B., & Stonor, T. (2010). Space Syntax-Strategic Urban Design. City Planning Review, *The City Planning Institute of Japan, 59*(3), 285.

Huang, W., & Zheng, H. (2018). *Architectural drawings recognition and generation through machine learning.* In Proceedings of the 38th Annual Conference of the Association for Computer Aided Design in Architecture, Mexico City, Mexico.

Koehrsen, W. (2018, October 2).

Neural Network Embeddings Explained. Retrieved January 9, 2020, from https://towardsdatascience.com/neural-network-embeddings-explained-4d028e6f0526

Koning, H., & Eizenberg, J. (1981). The language of the prairie: Frank Lloyd Wright's prairie houses. *Environment and planning B:planning and design, 8*(3), 295-323.

Krejcirik, M. (1969). Computer-aided plant layout. *Computer-Aided Design, 2*(1), 7-19.

Levin, P. H. (1964). Use of graphs to decide the optimum layout of buildings. *The Architects' Journal,* 7, 809-815.

Nagy, D., Lau, D., Locke, J., Stoddart, J., Villaggi, L., Wang, R., ... & Benjamin, D. (2017, May). *Project Discover: An application of generative design for architectural space planning.* In Proceedings of the Symposium on Simulation for Architecture and Urban Design (p. 7). Society for Computer Simulation International.

Puppi, L. (1973). *Andrea Palladio* (Vol. 2). Milano: Electa.

Puppi, L., Codato, P., Palladio, A., & Venchierutti, M. (2005). *Andrea Palladio: introduzione alle architetture e al pensiero teorico.* Arsenale.

Radford, A., Metz, L., & Chintala, S. (2015). *Unsupervised representation learning with deep convolutional generative adversarial networks.* arXiv preprint arXiv:1511.06434.

Ravenscroft, T. (2019). Wallgren Arkitekter and BOX Bygg create parametric tool that generates adaptive plans. Retrieved from https://www.dezeen.com/2019/06/27/adaptive-floorplans-wallgren-arkitekter-box-bygg-parametric-tool/

Rojas, G. S., & Torres, J. F. (2006). *Genetic algorithms for designing bank offices layouts*. In Prosiding Third International Conference on Production Research–Americas' Region.

Rykwert, J., & Schezen, R. (1999). *The palladian ideal.* New York: Rizzoli.

Stiny, G., & Mitchell, W. J. (1978). The palladian grammar. *Environment and planning B: Planning and design,* 5(1), 5-18.

Weinzapfel, G., Johnson, T. E., & Perkins, J. (1971, June). *IMAGE: an interactive computer system for multi-constrained spatial synthesis*. In Proceedings of the 8th Design Automation Workshop (pp. 101-108). ACM.

Wundram, M., Marton, P., & Pape, T. (1993). *Andrea Palladio 1508-1580: Architect between the renaissance and baroque.* Taschen.