Harvard Data Science Review • Issue 3.3, Summer 2021

Self-Organizing Floor Plans

Silvio Carta[1]

[1]School of Creative Arts & Centre for Future Societies Research, University of Hertfordshire, Hatfield, England, United Kingdom

Published on: Jul 23, 2021

DOI: https://doi.org/10.1162/99608f92.e5f9a0c7

ABSTRACT

This article introduces and comments on some of the techniques currently used by designers to generate automatic building floor plans and spatial configurations in general, with emphasis on machine learning and neural networks models. This is a relatively new tendency in computational design that reflects a growing interest in advanced generative and optimization models by architects and building engineers. The first part of this work contextualizes self-organizing floor plans in architecture and computational design, highlighting their importance and potential for designers as well as software developers. The central part discusses some of the most common techniques with concrete examples, including Neuro Evolution of Augmenting Topologies (NEAT) and Generative Adversarial Networks (GAN). The final section of the article provides some general comments considering pitfalls and possible future developments, as well as speculating on the future of this trend.

Keywords: self-organizing floor plans, computational design, architecture, machine learning, generative adversarial networks, artificial neural networks

---

Media Summary

This article introduces some of the techniques currently used by designers to generate automatic building floor plans and spatial configurations in general, by using machine learning and neural networks. This is a relatively new tendency in computational design that reflects a growing interest in advanced generative and optimization models by architects and building engineers.

In this work we analyze some of the most popular techniques to automatically generate spatial configurations in architecture. We try to highlight how and why different techniques are used and provide comments on the results for each approach.

We contextualize these methods within previous work of self-organizing configurations and suggest how we should consider and use these techniques in the future.

---

1. What Are Self-Organizing Floor Plans?

This article discusses how the space in which we live can be designed by algorithms instead of humans, with designers working out their projects driven by computer logic instead of Euclidean geometry. Of course, the question is more elaborate that it seems and it is, probably, even less exciting.

In the context of design, floor plan is an architectural term that indicates the bidimensional spatial arrangement used by designers to determine internal building layouts. More generally, people refer to floor plans today as

blueprints, spatial layouts, internal spatial arrangements, and so on.

Usually, designers start the spatial organization of space (a room, a building, an entire city) by encapsulating the main spatial qualities in an initial sketch. Through several procedural stages, this sketch is then developed into spatial arrangements, floor plans, and technical drawings (Plowright, 2014). During this nonlinear process, the designer evaluates each development of the drawings against the prescriptive data (building regulations, health and safety, minimum distances, etc.) and subjective aspects (taste, experience from previous projects, and experimental concerns).

Although automation in architecture and design is an integral part of the history of design (think of Sigfried Giedion's [1948] comprehensive accounts of how "Mechanization takes Command"), traditional approaches where projects start with sketches and templates and are developed through continuous refinements, are today challenged by a more linear and data-driven workflow where designers use algorithms to produce their project (cf. Ferreira & Leitão, 2015).

Unlike traditional architects, computational designers need to start their work by modeling the design problem (what is required by the project brief), including all relevant information (from building regulations to dimensional guidelines) and elaborating a logic that allows all parts of the design to be hierarchically related and processed. Once the design logic has been elaborated, this approach requires the use (or the development) of an algorithm, the implementation of which will allow the final design to be computed. A direct application of this method is represented by self-organizing floor plans (SOFP).

In the context of this article, with SOFP we refer to the combination of methods that designers use to automatically generate spatial building layouts. Generally, these methods include machine learning (ML) and optimization techniques, neural networks (NN) and evolutionary algorithms. The models usually include topological approaches (for example, with Kohonen's self-organizing feature maps [SOMs]) and object recognition (e.g., deep neural networks). The notion of self-organization has different meanings depending on its context. In the study of complex systems, for example, self-organizing may be associated with the idea of emergence, where a global order among elements is achieved through interactions of the parts governed by local rules (think of fractals or cellular automata). Although some of the topological approaches (for example, Kohonen's SOMs) rely on this idea, the examples included in this article also include different approaches, where self-organization is achieved by using neural networks and genetic algorithms.

This article addresses methods that are being recently applied in architecture and design. However, it is important to note that the problem of automated generation of spatial layouts is not new, as it has been developed since the second half of the 20th century and applied to many different fields and studied by many (e.g., March & Steadman, 1971; Shaviv, 1987, and more recently, Guo & Li, 2017; Martin, 2005; Merrell et al., 2010; Michalek et al., 2002; Zheng & Ren, 2020). Well-known examples are represented by automated facilities layout (cf. Chen et al., 2020; Levary & Kalchik, 1985; Liggett, 2000; Seehof et al., 1966, among

many others), spatial synthesis (cf. Eastman, 1975; Jo & Gero, 1998; Veloso & Krishnamurti, 2021), space planning (cf. Anderson et al., 2018; Brookes & Kaplan, 1972), and layout synthesis (Liggett, 2000; Wu et al., 2019).

To understand the importance and relevance of SOFPs for architecture and design today, we need to contextualize them within the design industry at large. In fact, there are three main viewpoints that can help to create a comprehensive picture of what is usually called a 'self-organizing floor plan.'

Firstly, the media (including magazines targeting nonarchitects or data scientists) present SOFPs as a magic tool where designers input generic data into a black-box algorithm that provides fully fledged floor plans as an outcome. This has led to a large debate (which is still ongoing at the time of writing) about the future of designers and the actual need for them in a situation where computers can potentially substitute them entirely (for example, Carta, 2020a; Celento, 2007; Fairs, 2019).

To date, we are still far from what the media often portray, where computers automatically produce floor plans and building designs. We will probably never arrive at this point, as design is an intrinsic human activity that requires intuition, taste, and, ultimately, a good degree of subjectivity (cf. Davis, 2020). However, it is also true that synthetic approaches can be very helpful to designers as they increase the accuracy of their work, reduce the number of errors in a project, save precious time, and avoid tedious tasks. This opens up new possibilities for development (Carta, 2020b).

Opinions differ among architects and designers on this matter, between those who recognize the potential of the use of algorithms in the design process, and those who consider them with suspicion. The relationship between human and machine in architecture is elegantly summarized by David Rutten as: "the architect remains the designer, while the computer becomes the critic" and this is related to the idea that "ultimately, a computer has no idea what it's doing, let alone why it's doing it. Understanding stuff is what humans are good at" (Rutten, 2021).

Rutten's opinion indirectly suggests the importance of considering hybrid approaches, where the contributions of machines and humans are deployed at different stages and to tackle different problems in a design process, playing on each specific strength. A growing number of projects epitomize this human-AI (or human-machine) co-creation (e.g., Pitso, 2019; Wikström, 2018; Woo, 2020).

The examples included in this article illustrate this delicate relationship between designers and computers, which is far from being fully resolved. One way of looking at this is through the lens of the designer, who sources and selects the initial data sets, curates and cleans the data in preparation for the computer to be calculated, and models the design problem in order to address the questions at stake (from the design brief, the client's requests, etc.). In this case, we may argue that the designer is still in utter control of the design while the machine is merely executing instructions set by the designer. Another perspective can be from the

viewpoint of the computer that is employed in the design process in the hope that new and unforeseen results may emerge. Take the example of clustering a given data set. Designers may use clustering or classifying techniques to find new aspects in their design that would not be visible without the aid of a computer. In this case, we may argue that the designer is still in control of the design, although they trust the computer to generate parts of the design that are outside of their reach. Designers become meta-designers or, in extreme cases, they become critics of their own work, as suggested by Rutten.

The argument suggested by this article is that designers in both cases are still in full control of their creative work, especially considering the effort they make in preparing and curating the data set, modeling the problem, setting up the entire workflow, evaluating results at any stage of the process to ensure consistency and validity and, ultimately, being responsible for the final results obtained.

Secondly, designers, especially those interested in computational design (that is, simplifying, a subset of design where computers are heavily used), consider self-organizing plans and generative design in general as one of the future directions for development. Not only do computational designers create their own tools using open-ended graphical programming interfaces, but they also see such tools as increasingly integrated into the main software packages that are largely used in the design industry (for example Autodesk Revit or McNeel's Rhinoceros 3D). Designers are moving their work (together with their creativity and problem-solving approaches) from the traditional representation of initial ideas and their continuous refinement to a radically new dimension. In this, designers create their own (computational) tools that serve them as intelligent collaborators in their projects (Jabi, 2013; Terzidis, 2006).

Thirdly, computer scientists have found that applications of certain models and algorithms can be tested against a real-case scenario (for example, the design and construction of a building). An example of this is the work that Tarabishy and his colleagues have done on the use of "deep learning surrogate models for spatial and visual connectivity" (Tarabishy et al., 2020). In this, they use a number of ML methods to reduce the computational time needed to simulate the spatial and visual connectivity of a given office space. More and more, we see the emergence of new consultancy practices where mathematicians, statisticians, software developers, and computer scientists work along with designers and engineers to offer bespoke and intelligent solutions for architecture and the construction industry. Examples of such practices are Kreo, Hypar, or TestFit, among many others.

This article is an attempt at clarifying the reasons why the second and third views are important, for they are significantly changing some aspects of the architecture and construction industry. Indirectly, with the analysis included in this work, we want to offer a robust counterargument to the first view, explaining the extent to which the idea of computers as a substitute for designers is not only simplistic, but fundamentally flawed.

2. Self-Organizing Floor Plan at Work

An increasing number of designers and computer scientists are working together to develop new ways of creating automated methods to generate spatial solutions (among others, Eisenstadt et al., 2019; Goodman, 2019; Kalervo, 2019; Liu, 2017; Nauata, 2020; Phelan et al., 2017; Sandelin, 2019; Zeng et al., 2019).

In general terms, we might say that the success of each model for this particular task is connected to two main aspects. The first concerns advancements in machine learning research. The growing use of artificial neural networks (ANNs) in computational design is a reflection of the fast advancement of research in generative models (Dhondse, 2020) and true distributions (Alcin, 2019); this, along with increasing computational power and training data sets available (Hodas & Stinis, 2018). As the possible applications of these methods increase and become more reliable, research in this direction is increasingly active.

The second is the creative use that designers make of this new research. It is important to note that although many of the current projects are the result of multidisciplinary teams (including computer scientists, data scientists, and designers), they often originate from and are led by architects (see, for example, the work conducted at MIT's SenseAble City Lab). Architects, and designers in general, use computational methods to generally produce applied research using primary new knowledge produced in statistics, mathematics, computer science, data science, and so on. It is in its possible application to buildings, cities, or in this case, floor plans that designers can offer novel and creative approaches to real-life problems.

The information-processing model that is commonly used is an ANN. One of the reasons behind this choice is that this model is proven to be particularly powerful when designers have large data sets available for their projects (for example, the website HousePlans.com offers around 40,000 floor plans that can be used as a training data set). The choice of using an NN is also quite popular due to the fact that most of the computation needed for predicting spatial distribution (floor plans) is made largely possible on standard machines (e.g., home computers, workstations, etc.) through backpropagation and general-purpose Graphics Processing Units (GPUs) (Danka, 2020).

The examples presented in this section use optimization techniques (neuro evolution of augmenting topologies —NEAT) and variants of neural networks based on graphs (e.g., graph convolutional networks—GCN). Algorithms based on graph theory result in quite effective manipulation of data in spatial configurations. Designers and those trained in spatial abstraction usually find topological approaches intuitive, for they have direct applications to spatial organization. A clear example of this is space syntax (Hillier & Hanson, 1989)—including one of its methods, isovist (Benedikt, 1979; Turner et al., 2001)—which is among the most developed and used computational spatial theories applied to architecture and cities. Modeling design problems through graph theory is a common strategy in the projects included in this section. The projects that follow include models based on graph optimization and generative models.

We start with a non-data-driven approach based on NEAT, which is a powerful optimization method. Generalizing, optimization models are employed to find solutions through an objective (or fitness to criterion) function. Possible solutions are iteratively evaluated against the objective function in search of optimal values (Hoos & Stützle, 2004, p. 13).

This approach is based on Stanley and Miikkulainen's (2002) work where an advanced method for evolving neural network topologies along with the associated weights is presented. NEAT's outperformance regarding the other fixed-topology approaches is related to "employing a principled method of crossover of different topologies, protecting structural innovation using speciation, and incrementally growing from minimal structure" (Stanley & Miikkulainen, 2002, p. 99). Simon (2017) and then Carta et al. (2020) developed a workflow to optimize an existing floor plan based on a number of design criteria. NEAT involves the use of genetic algorithms to find the most fitting topology in a given initial configuration. In this method, each solution (also called genome) is represented by a series of nodes and connections. Each node contains information that is iteratively evaluated against a fitness function to ascertain how close (or far) that particular solution is to the desired result (as the combination of the optimization criteria). Each connection stores information about the nodes that it links and their relative weights. The weights give important details about the role of that connection within the system; for example, the distance between the nodes (or rooms) or whether that link is considered within a particular instance. In each iteration, there is a mutation of the genome underpinned by the combination of the previous best-performing solutions where new nodes and connections can be added or removed. The method used by Carta et al. (2020) includes the initial generation of a random population of solutions to which a number of mutations is applied. In each iteration, the solution is mapped against a generic floor plan (that needs optimizing). An ant-colony optimization (ACO) algorithm is used to evaluate the best connections between the rooms (corridors). The configuration is then evaluated against the fitness fiction and the criteria used as the input. The possible solutions are then used as inputs for the next generation. The process repeats using the best solutions of each generation to improve the overall solution, until satisfactory results are obtained. An example is show in Figure 1.

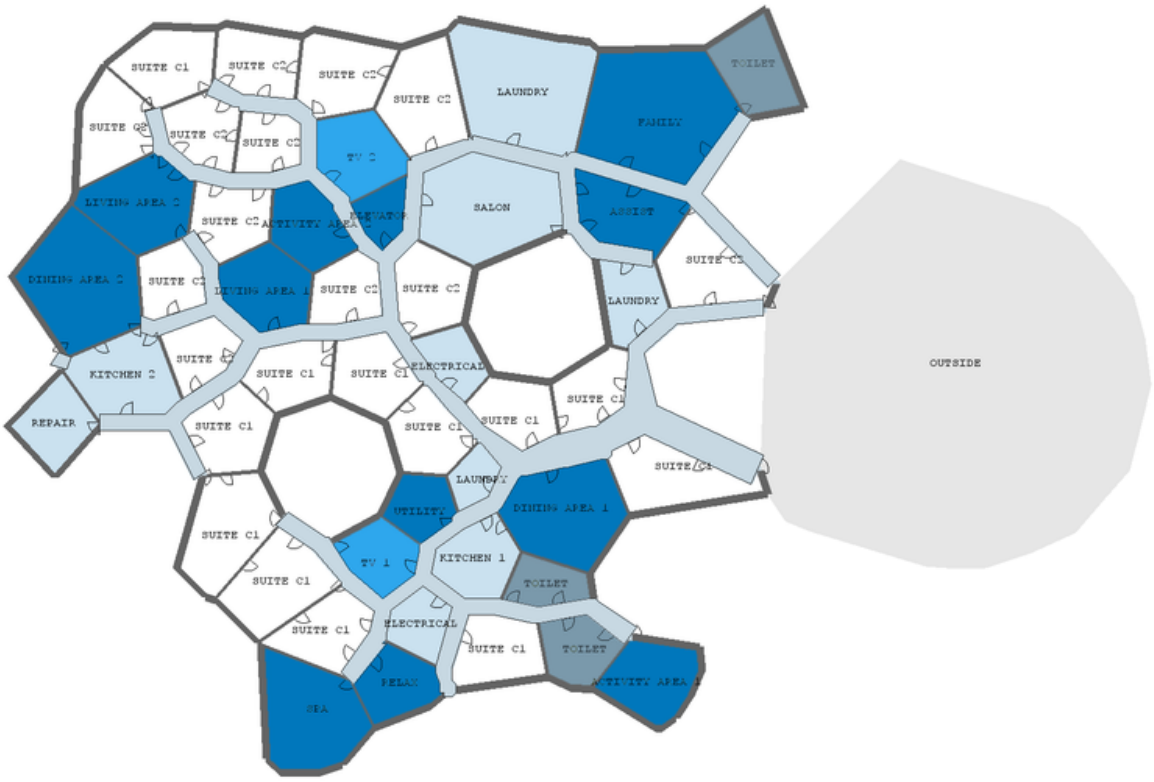
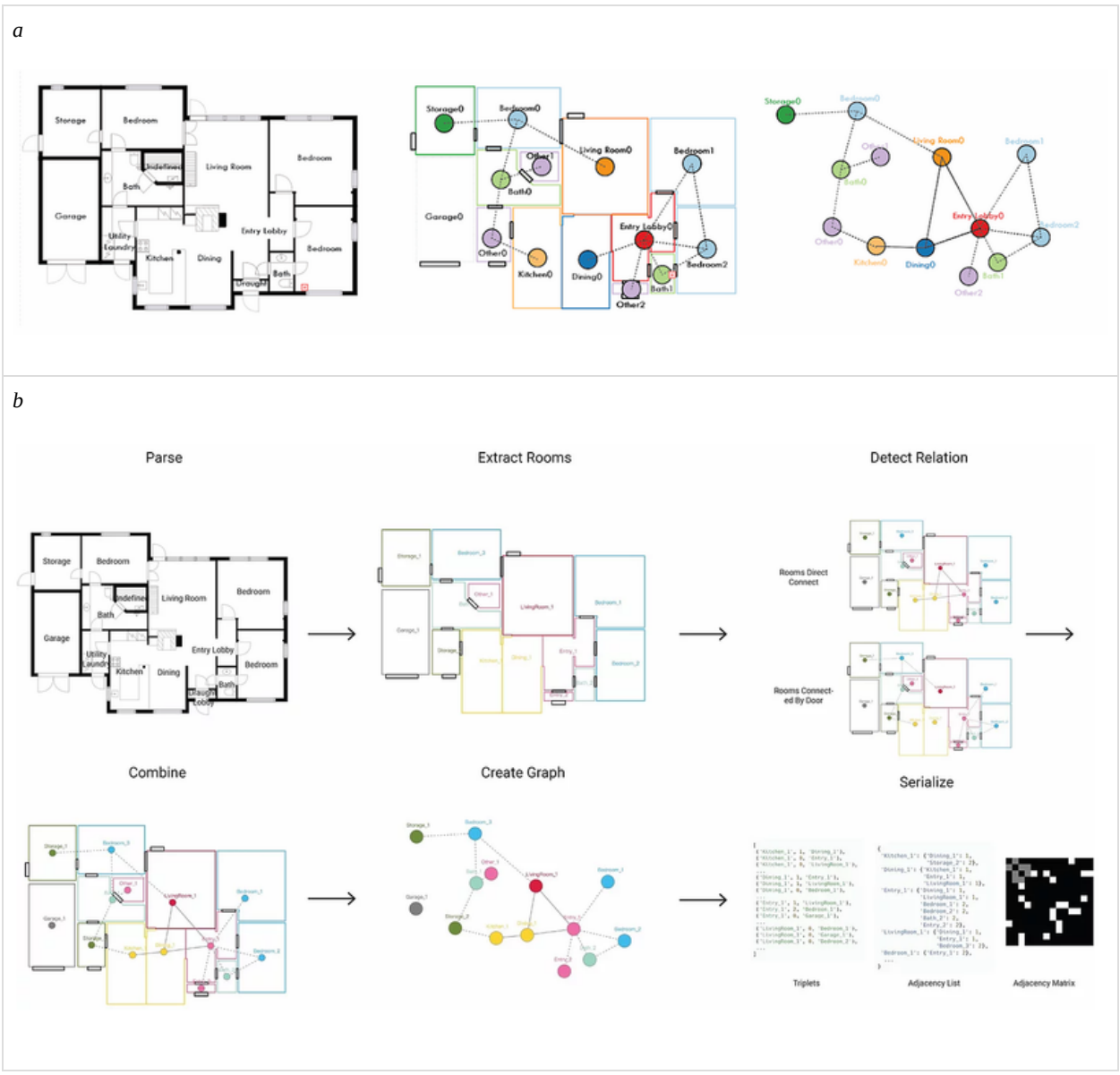


Figure 1. Self-organized plan with NEAT where optimization criteria based on care home residents' need have been applied. This figure illustrates one of the spatial configurations obtained as the result of the generative process where distances among rooms and the position of each area is optimized according to a set of given design criteria. From Carta et al. (2020).

Although NEAT applied to self-organizing floor plans can yield promising results, this method is indeed not very much used by architects and designers who seem to prefer in general data-driven approaches with NNs and generative methods.

A more common approach is graph convolutional networks (GCNs). This is where the NN informational model is based on graphs and their elaboration. They are defined as convolutional due to some of their characteristics (for example, the filter parameters), which are common to different parts of the graph (Kipf, 2016). A relevant application of GCN to floor plans is the work in progress of Li et al. (2020) where the user can input new nodes into the graph directly via a browser and the program generates a floor plan corresponding to the graph.


a
Storage
Bedroom
Undefined
Living Room
Bedroom
Bath
Garage
Utility Laundry
Entry Lobby
Kitchen
Dining
Bedroom
Bath
Storage0
Bedroom0
Living Room0
Bedroom1
Other1
Bath0
Garage0
Entry Lobby0
Other0
Kitchen0
Dining0
Bedroom2
Bath1
Other2
b
Parse
Extract Rooms
Detect Relation
Rooms Direct Connect
Rooms Connected By Door
Combine
Create Graph
Serialize
Triplets
Adjacency List
Adjacency Matrix

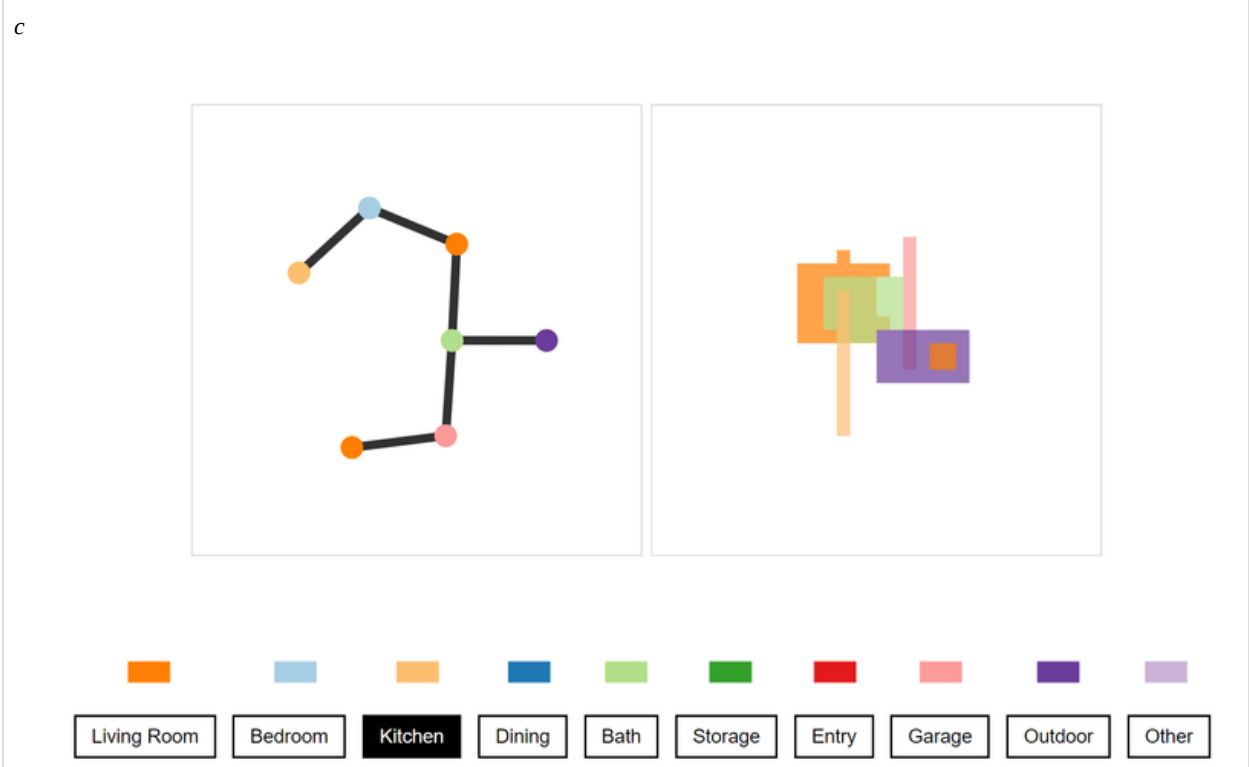


Figure 2. Representation of a typical floor plan as a graph (2a and 2b) and final result, where the user of the web app inputs the rooms (kitchen, living room, etc.) as nodes of a graph that are then combined into a floor plan (2c). Li, Ao, Tian, Runjia, Wang, Xiaoshi and Yueheng Lu. PlanGCN Team. https://www.aoli.org/graph2plan.

Graph to plan demo

Visit the web version of this article to view interactive content.

This method outputs a plan layout using an input graph as the predictor of the NN through a three-step process. Firstly, through graph-embedding information, including topology and relational features, among rooms and architectural elements (door, windows, etc.) from an existing floor plan is converted into a graph (Figure 2b). Secondly, the vectors generated in the first step are computed by a message passing network, considered the core of the GCN, where all (relational) information related to each specific node is propagated through the graph. The GCN outputs updated vectors passed to a box & mask regression network to evaluate the loss function. The method outputs a new spatial configuration, which is optimized with regard to the initial input. In this particular application, the input is represented by a number of points in space (position of rooms) that is read by the NN as an input graph.

The example of GCNs clearly shows how floor plans can be modeled as graphs of which topology is optimized resulting in a new spatial configuration. However, the approaches to SOFP that seem to be currently more popular are those based on generative models.

In this group we have methods where deep learning approaches are used to predict the implicit distribution and configuration of graphs (Zhou et al., 2018, p. 18). Unlike in discriminatory approaches, where features are mapped to labels, in generative models the aim is to find (by prediction) new features given an initial set of labels.

NEAT and GCN seem so far to have been successful at illustrating how initial inputs can be translated into spatial arrangements. However, it is with generative adversarial neural networks (or GANs) (Goodfellow et al., 2014) where most of the progress to date is being made by architects and computational designers. In order to understand why GANs are considered powerful enough for the task of generating realistic floor plans, we need to briefly see how they work.

GANs are unsupervised networks where there are two main components. The first network (generator) generates the data outputs (for example, images) that are assessed by the second network (discriminator). The discriminator is trained with both real and fabricated images to ascertain whether the images are real (true data) or fake. The generator is trained to produce images that look real for the discriminator. The data can be backpropagated through both the generator and the discriminator to ensure that the former adjusts its parameters in order to reduce the difference between real and fake images for the latter (Goodfellow et al., 2014).

For example, architect and designer Erik Swahn produces simulations where GAN is employed to generate new built environments ranging from building interiors to maps and landscapes. Swahn experiments mostly using conditional generative adversarial networks (Isola et al., 2017) and StyleGAN (Karras et al., 2017). This later is a generator architecture used for generative adversarial networks developed by NVIDIA in 2019 (Karras et al., 2019). In his work on automatic production of floor plans and other architectural spaces, Swahn developed a workflow whereby he scraps a number of webpages for images to generate a training data set (in this case 64 images of elevations, 342 sections, 499 plans from Archimaps, 824 from Archive of Affinities, 1,455 plans from plansofarchitecture, 791 plans from DeZeen, etc.) to be used with Pix2Pix (Isola et al., 2017), a generative adversarial network built for image-to-image translation. He then selects and generates pairs of images by drawing quick outlines directly on a touchpad (Figure 3a and 3b), and visualizes the results of the process using different inputs (images from a webcam/visualizer, geometries like simple rectangles or architectural drawings) (Figure 3c) on the browser (Figure 3d) (Swahn, 2019). Swahn defined this work as latent spaces, referring to the “bottleneck layer […] between the encoder and decoder in the GAN generator […] where images are maximally reduced to a latent vector of length n. This low-dimensional vector can be seen as a point in a n-dimensional space (the model’s latent space)” (Swahn, 2019).

a

b

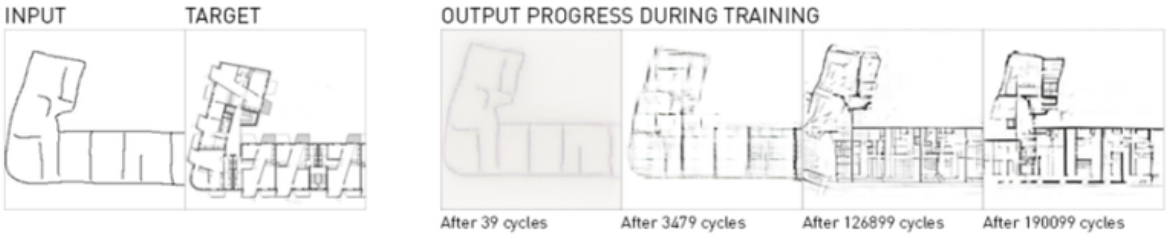

INPUT
TARGET
OUTPUT PROGRESS DURING TRAINING
After 39 cycles
After 3479 cycles
After 126899 cycles
After 190099 cycles

Figure 3. Latent Spaces. This figure shows one of the results of Swahn's project where different modes of representation of architectural buildings can be translated (for example, sketches to plans, maps to plans, webcam images to plans, etc.). Erik Swahn, *Latent Spaces*. From: https://twitter.com/erikswahn.

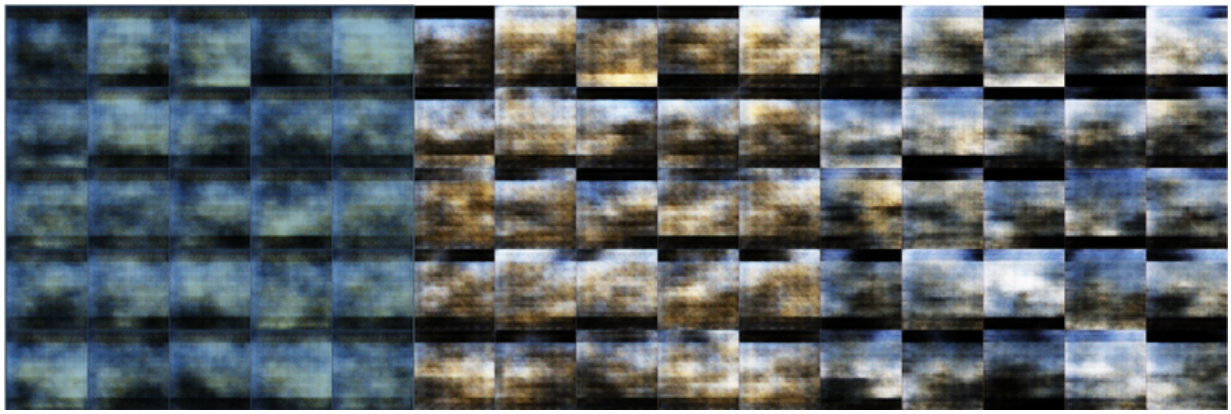
Erik Swahn. Latent Spaces.

Visit the web version of this article to view interactive content.

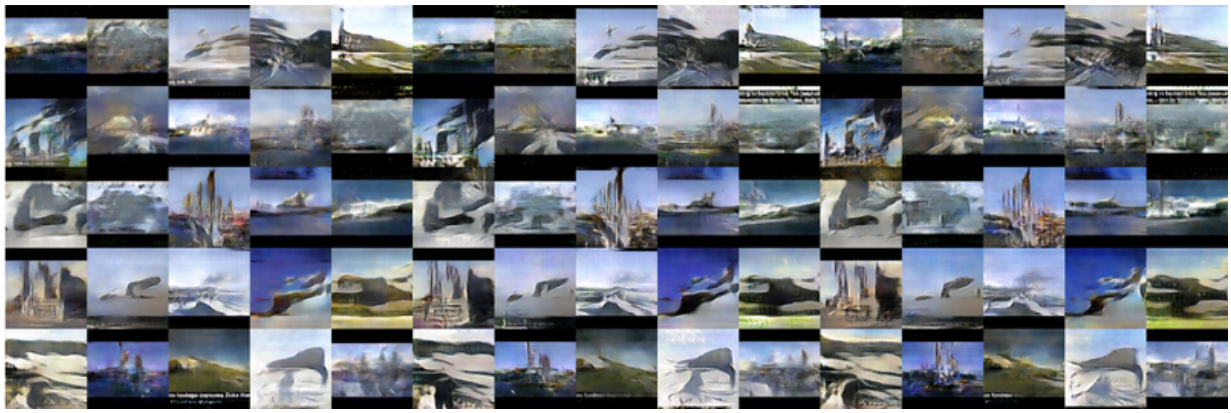
Other designers like Sean Wallish have been working with deep convolutional generative adversarial networks (DCGAN) to produce new building silhouettes based on existing data sets. In GAN Hadid, Wallish (2019b) trained a neural network with information about the relationships between pixels in published photographs and renders of Zaha Hadid Architects' architectural buildings (Figure 4). This model has been trained with 8,428 low-res images of Zaha Hadid Architects' buildings (Wallish, 2019, p. 12). The training involved multiple sessions with different number of epochs each (session 1 = 30 epochs, session 2 = 10 epochs, etc.). At the end of the training, the algorithm was able to associate architectural elements (forms, materials, silhouettes, etc.) from one or a combination of projects (in the training set) to new shapes. The final results of this method are not verisimilar images of real buildings. In this respect, Wallish identifies a number of aspects that characterize the training data set and that are somehow amplified in the generated images, including the lack of details, the clear separation between ground and sky, background and foreground, and the predominance of white/light gray colors (Wallish, 2019a, pp. 29–30).

However, this project offers very promising results for architects, for it shows a workflow to generate new configurations in urban settlements based on existing ones.

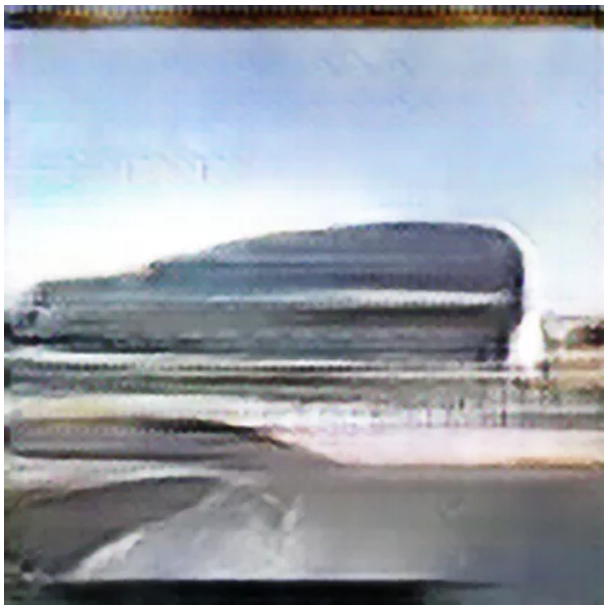
a



b



c



d

Figure 4. GAN Hadid. This illustrates the initial results at the epoch 1 (4a) and final ones at epoch 30 of the 1st stage (4b). Figure 4c and 4d show the original photograph of Hadid's building (4d) and a new configuration generated by Wallish (4c). Figure 4c is generated by Wallish: From: https://www.seanwallish.com/gan-hadid. Figure 4d: Zaha Hadid Architects Beijing Daxing International Airport. https://www.zaha-hadid.com/architecture/beijing-new-airport-terminal-building/. Photograph: Hufton + Crow.

Similarly, in his project GAN Loci, Kyle Steinfeld (2019) used GAN to generate imaginary new places by training the NN with spatial and visual characteristics (light conditions, forms, textures, or colors). Steinfeld's process to generate these synthetic images of new places is threefold. The first is the preparation of the data sets. This is where a large number of photographs (around 500 images per each of the nine sites chosen for this project) depicting points of interest in various cities in the United States and Europe are collected from Google Street View, and cleaned and categorized into organized sets. In each training set, a pair of images is created where a raster RGB image is associated to its relative greyscale version that represents its depthmap (Figure 5a).

The images are then formatted to be readable by the training models, StyleGAN (Karras, 2018) and Pix2Pix (Isola et al., 2017). Lastly, the two models are used to generate new synthetic images, each of them producing different results with different interfaces.

a *b* *c*

d

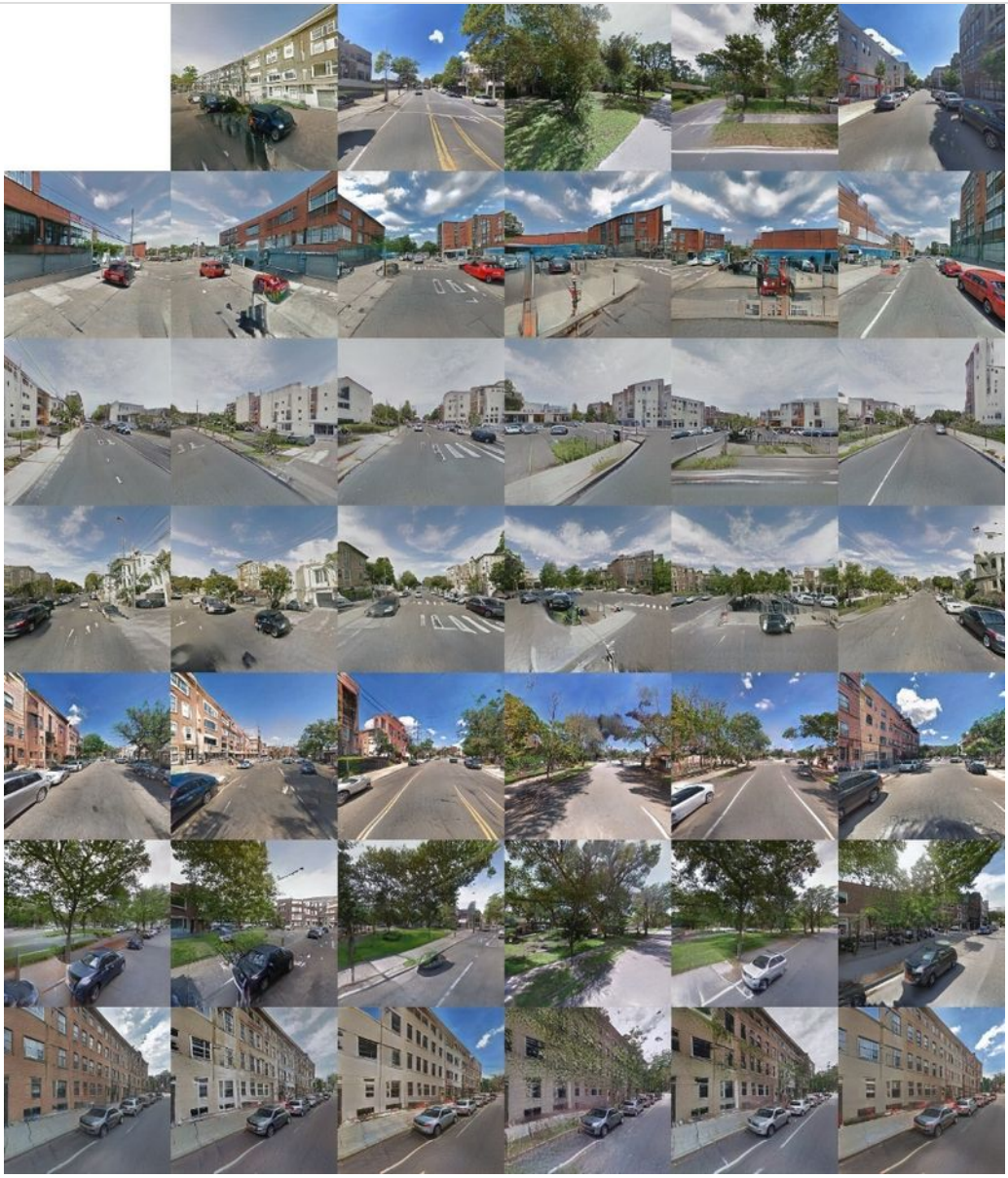


Figure 5. Synthetic images produced with Pix2Pix (5b and 5c) and StyleGAN (5d). The volume on the right-end side of the depthmap image (5a) "is interpreted by the Rotterdam model (5b) as a large brick housing block, as is typical in the Dutch city, while the Pickwick Park model (5c) renders this massing in a manner typical of the Northern Florida flora, suggesting the mass of a mossy Live Oak" (Steinfeld, 2019). In the images generated with StyleGAN, the "vertical columns define course features, such as camera direction and orientation, while horizontal rows define fine features, such as textures, colors, and lighting effects of each urban place" (Steinfeld, 2019).
Images from: Kyle Steinfeld http://blah.ksteinfe.com/191026/gan_loci.html. Figure 5d is from: https://towardsdatascience.com/gan-loci-e2bbd1b4926f.

Steinfeld's work offers a novel method to generate a large number of images that are site-specific, retaining some of the unique characteristics (or "tacit properties" in Steinfeld's words) of each place used as a case study. More importantly, Steinfeld considers this method as potentially analytical where GAN can be used to: "reveal certain properties useful in uncovering the nature of urban places […] [like] forms, textures, colors, and qualities of light that exemplify a particular urban location" (Steinfeld, 2019).

While the three projects illustrate how GANs can be used to generate (architectural) spatial configurations in general (in this case at the urban scale), the focus on interior layout is particularly relevant to this discussion. One of the examples where, to date, the methods discussed so far have been applied comprehensively is perhaps Stanislas Chaillou's ArchiGAN. This is a Generative Stack for Apartment Building Design. In this work, GANs are applied in different ways to the three stages of the project: building footprint massing, program repartition, and the furniture layout (Chaillou, 2019). "By nesting these models one after the other, [he] create(s) an entire apartment building 'generation stack' while allowing for user input at each step" (Chaillou, 2019). The GAN method applied here is Pix2Pix (Isola et al., 2017), yet in the implementation of Christopher Hesse's Image-to-Image Translation in TensorFlow, the Torch code has been ported to TensorFlow (Hesse, 2017).

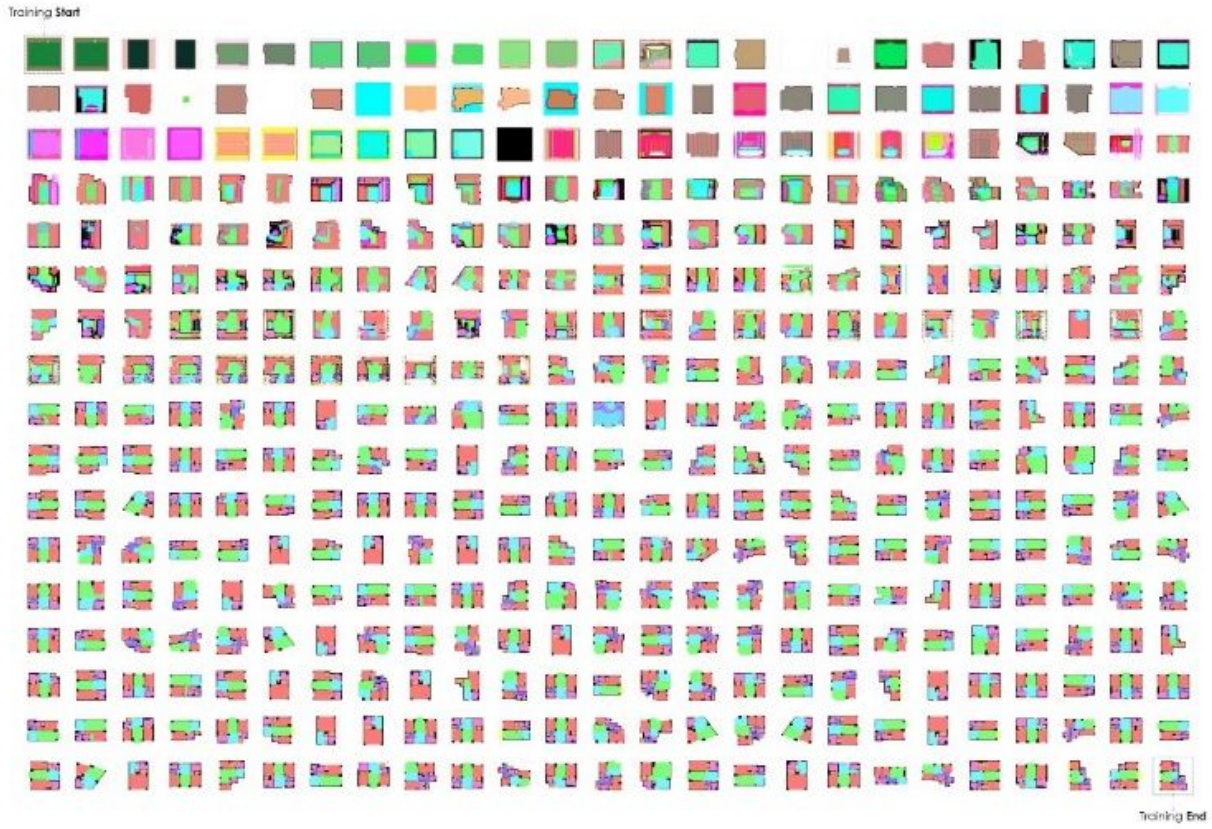


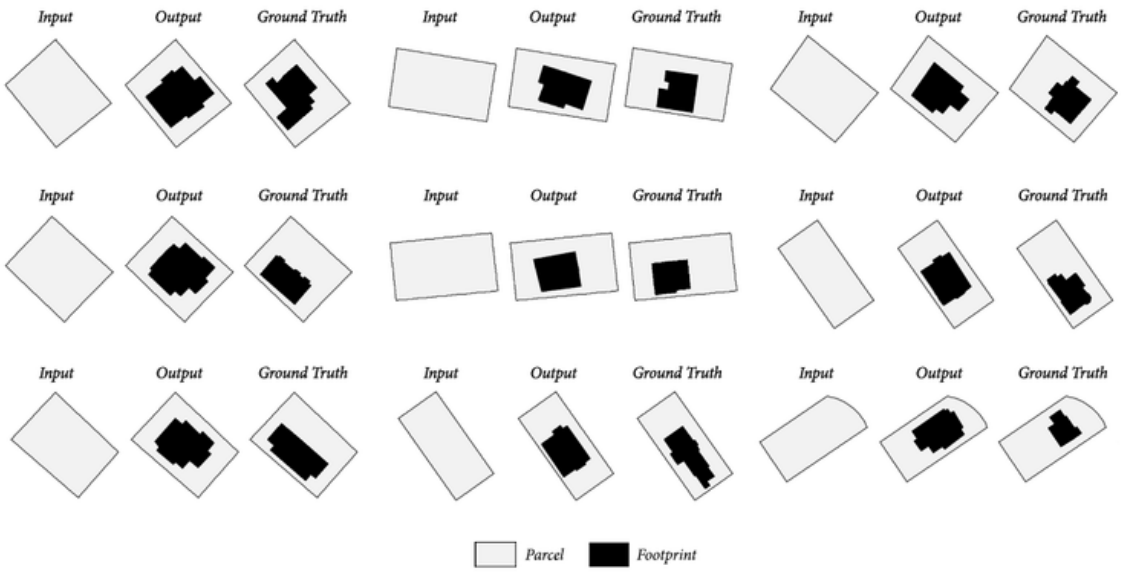
Figure 6. Sequence illustrating how the GAN model increasingly learns to generate floor plans from initial data sets. Each floor plan is a sequential outcome generated during the training of the network. From: Stanislas Chaillou, https://developer.nvidia.com/blog/archigan-generative-stack-apartment-building-design/?linkId=70968833.

The first application uses spatial data related to the surroundings of a generic house in order to determine the building outline. Information like location, the shape of the plot, and orientation have been used to infer the building footprint and general massing. Chaillou trained a GAN model with Geographic Information System (GIS) data from the City of Boston (with plot shape and relative building footprint) so then the NN could predict generic results for any type of new plot of land.

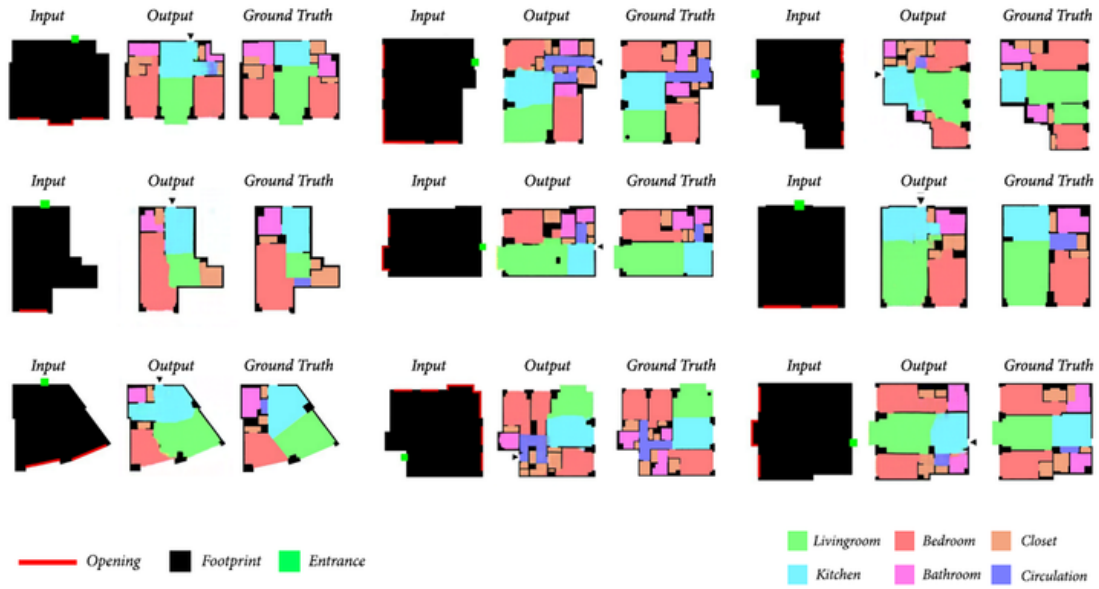
Once the building outline is determined, NN is employed to predict the position of the windows (fenestration) and internal layout (including the position of walls, doors, and corridors). The initial footprint (from the previous stage) is given to the NN as an input. The NN is trained using a large database of house floor plans where the position of windows and other relevant elements is annotated. The NN outputs a new floor plan with the room type classified by color and labels as shown in Figure 7.

The final application of the model is used to associate interior furniture with the rooms. To each colored area (representing a room type in the previous stage), there is a corresponding set of furniture and appliances (bedroom = bed, wardrobe, bedside table, etc.) based on the same database used in Stage 2.

a


Input
Output
Ground Truth
Parcel
Footprint

b


Input
Output
Ground Truth
Opening
Footprint
Entrance
Livingroom
Bedroom
Closet
Kitchen
Bathroom
Circulation

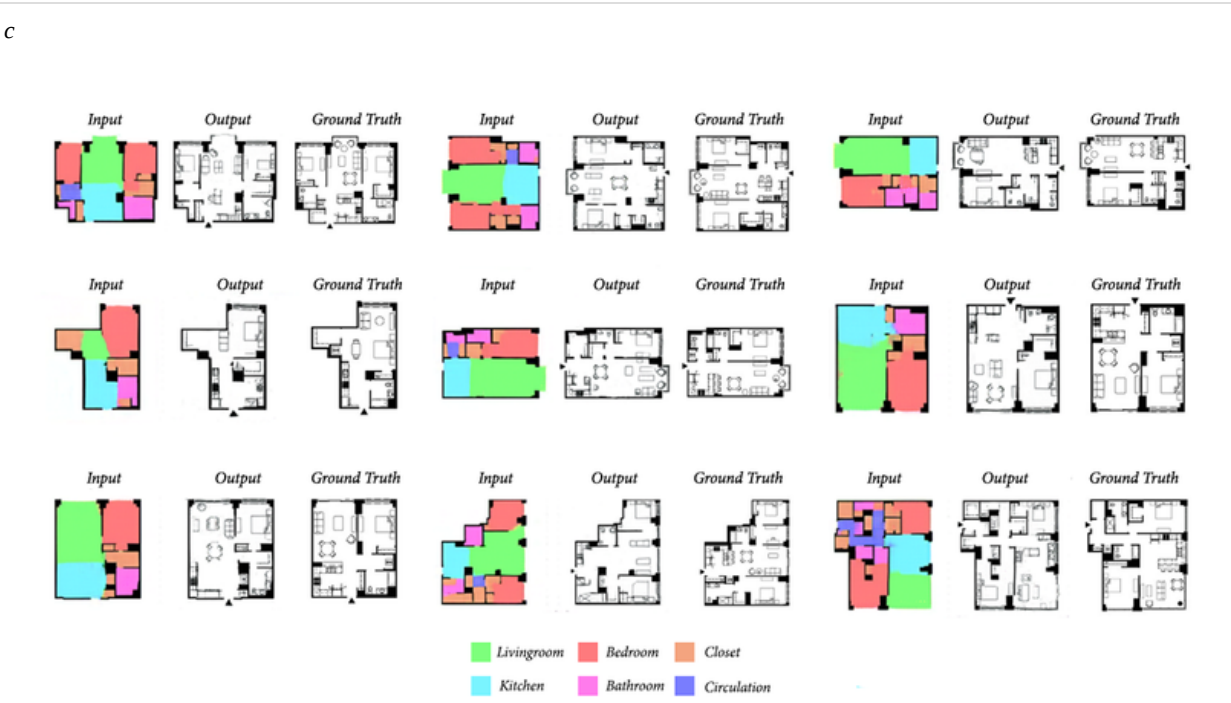


Figure 7. Sequential application of the neural network: Three stages. We can see the initial input from the designer as a generic geo-referenced plot (7a), the final output generated by the GAN (7b), and its ground truth (7c): the original floor plan used to train the network (this process is applied to all plans in the set). Output floor plans and ground truth are often compared to evaluate the similarity between the computer generated and the original scheme.
From: Stanislas Chaillou, https://developer.nvidia.com/blog/archigan-generative-stack-apartment-building-design/?linkId=70968833.

ArchiGAN Model II: Interface and Process Demonstration (Figure 7b)

Visit the web version of this article to view interactive content.

3. What Model Is Best for Self-Organizing Floor Plans?

It is difficult, if not impossible, to say today with certainty which model is better for generating self-organizing floor plans. A related question also arises as to which one will be most used in the future. All of the methods seen here have their own respective pros and cons. For example, GANs are very successful at providing clean, sharp synthetic images (Karpathy et al., 2016), but their success is dependent on stability. The right balance needs to be struck between the generator and discriminator (Salimans et al., 2016). Convolutional Neural Network (CNN) has been applied extensively and with relative success in the recognition of objects and parts

of images (LeCun, 2015, p. 439) with more recent application in facial recognition. More importantly, CNNs offer a great level of accuracy as the labeling can happen at the scale of pixels (LeCun, 2015, p. 439). However, they require large data sets and precise labeling to produce accurate results (LeCun, 2015, p. 440). NEAT is a powerful method to optimize a network by finding a minimized topology by searching for solutions with the minimum number of dimensions in each generation (Ibrahim et al., 2019, p. 111). NEAT has been used quite successfully in approaching real-time problems with user's input/action (Bird et al., 2019, p. 752). However, its performance significantly depends on the initially chosen topology (Ferner et al., 2017, p. 11) and the control of complexity within inner networks (Le Goff et al., 2020, p. 43).

So far, we have explored two main approaches to self-organizing floor plans. One based on the optimization of an initial configuration (this being an existing building floor plan or a hypothetical one based on a set of desired criteria), and the second focusing on the generation of a new configuration (based on prediction of new labels from a training set).

Although these two approaches can be combined (see, e.g., Chen at al., 2019) and share some logic (think of the generative aspects of both approaches), they are quite distinct in their objectives. In optimization problems using NEAT, the final spatial configuration is strongly conditioned by the initial configuration (i.e., existing floor plan), and by the ways in which inputs are given at each iteration of the Genetic Algorithm (GA). Optimization approaches in general yield more accurate results, for they provide a new spatial configuration that retains many of the original architectural features as a part of the topology (e.g., doors connected to walls as in the original plan). Conversely, with GANs the outcome is largely dependent on the training data set used, so factors like the size of data sets, initial data resolution, and quality of the data set are highly influential to achieve the desired results. New configurations generated through GANs are in general more innovative, as outcomes are often unexpected, yet they tend to be less accurate, as there is not an original floor plan used as starting point (and for comparison).

It is intuitive to see why, in cases where designers need an accurate and controlled new configuration that 'evolved' from the original one, optimization models through genetic algorithms are preferred. These allow designers to retain a good degree of accuracy in the position of key elements of the configuration of which movement in space requires careful consideration, including structural elements (load bearing walls, columns, beams, etc.), health and safety parts of the building (fire escape routes, fire compartments, etc.), and other building parts (shafts, ductwork, etc.).

By the same token, in those cases where designers are working with a larger degree of freedom (e.g., scoping out the possibilities of a new building or space, working with speculative design, etc.), generative models like GAN are effective in producing synthetized solutions. They allow designers to obtain often unforeseen configurations, generating an element of surprise and unpredictability in the design process.

The two approaches should be used in different instances with different aims, mostly depending on the stage of development of the project (e.g., conceptual versus development phases). As unsurprising as it may sound, the answer to the initial question, 'What model is best for self-organizing floor plans?', is that it depends on what designers are after in each particular phase of their project.

In general terms, however, we note that GANs (and their variants like StyleGAN or DCGAN) seem to be increasingly popular among computational designers and data scientists. This is most likely related to the fact that GANs can be trained with relatively modest data sets (e.g., Wallish's GAN Hadid project has around 8,000 images as a training set) and to the fact that image-based data sets are increasingly affordable and offer promising results. We should note that many of the GAN projects in this article have been developed using NVIDIA GPUs technologies (see Song et al., 2018) (e.g., Wallish, Chaillou, and Steinfeld's) that allow for relatively satisfactory results in short periods of time and with general-use machines. Most of these implementations of GANs are based on recent developments at NVIDIA for optimized image processing (Karras et al., 2017; Karras, 2020), as well as new tools that facilitate the training of data sets and the use of ML in general, including Pix2Pix (used by Chaillou and Wallish), and more generally TensorFlow, PyTorch, and so on(underpinning most of the projects using GANs).

GANs may not necessarily be the best solution for SOFPs, but their use is certainly increasing and they are attracting a growing number of designers who want to experiment with these models, given the growing number of publications and promising results.

If more designers are experimenting with methods like GANs and NNs in general as a part of their design explorations, access to the appropriate technology becomes a key part of this development. By access we refer to infrastructure (this includes hardware: CPUs, GPUs, or TPUs, computational power of general-use machines and software: open-access libraries and online resources) as well as skillset. For designers to be able to use relevant data science techniques and methods in their projects, not only do they need to understand the design problem and model it, but they also need to effectively implement their strategies. Thankfully, a growing number of libraries, Application Programming Interfaces (APIs), and platforms are now available to designers who can more easily access some of the data science methods mentioned in this article. The following section explores some of them, illustrating what designers can do and how.

4. How Are SOFPs Used (by Designers)?

The models presented in the previous sections illustrate cases where designers have a hybrid profile (data scientist/designers) and are able to work with the two aspects of these projects (data science/architecture) at the same time. Most of the techniques and tools used in the examples from Section 2 are not design tools per se. They are, rather, workflows generally used in computer graphics, computer vision, and machine learning projects (for example, our NEAT project is coded entirely on Python, and Pix2Pix is an image-to-image mapping tool). In parallel to this approach where designers use data science tools and methods, there is an

important growing infrastructure of software, libraries, and interdisciplinary communities developed directly for designers. These tools are helping more and more to democratize important developments in data science into the field of design and architecture.

The libraries and software presented below represent a good section of the new tools for data science available today to designers and show the direction that development of computational tools is taking, especially in helping designers without a strong data science background to work with ML and NNs models.

The implementation of neural network methods (and machine learning in general) for computational designers has rapidly developed in the last decade. Firstly, there is the emergence of experimental approaches where individuals make promising attempts at running NNs through modeling and Computer-aided Design (CAD) platforms. These scripts are usually written in C# or Python, using developer kits within the CAD or parametric software (Khean et al., 2018). A good example are the ML tools within the Lunchbox Library for Grasshopper and Revit, and the beta version of the APIs in Grasshopper including Crow, a library with supervised learning through backpropagation networks and unsupervised learning through self-organizing maps. Figure 8 shows how the network is trained using an internalized database to associate basic bidimensional shapes with three-dimensional meshes representing objects (for example, fruits).

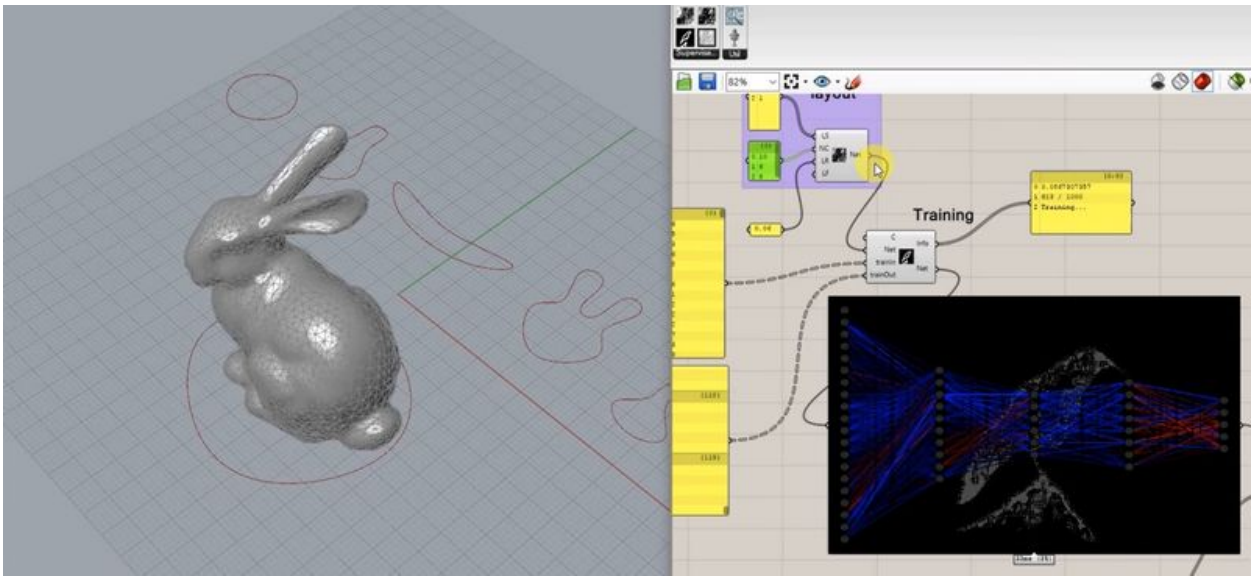


Figure 8. The Crow Library. The interface of the Crow library in action, with the graphical language User Interface (UI) on the right and the 3D shapes outputted by the NN on the left. From: Benjamin Felbrich, https://www.food4rhino.com/app/crow-artificial-neural-networks.

Visit the web version of this article to view interactive content.

Another good example is PlanBee, developed by Marco Juliani at the beginning of 2020. This library allows designers to use a Kohonen SOM algorithm to automatically organize blocks of activities in a given floor plan with pre-settings (Figure 9). “In order to understand this, it helps to think of the floor plan as a field of voxels, each of which contains values for different computed metrics. Once the metrics are computed, each voxel corresponds to a multi-dimensional vector. The SOM basically reconciles the multi-dimensional vectors of the field of voxels with the features one specifies should belong to a programmatic mix (i.e. the list of activities in the building)” (Juliani, 2020).

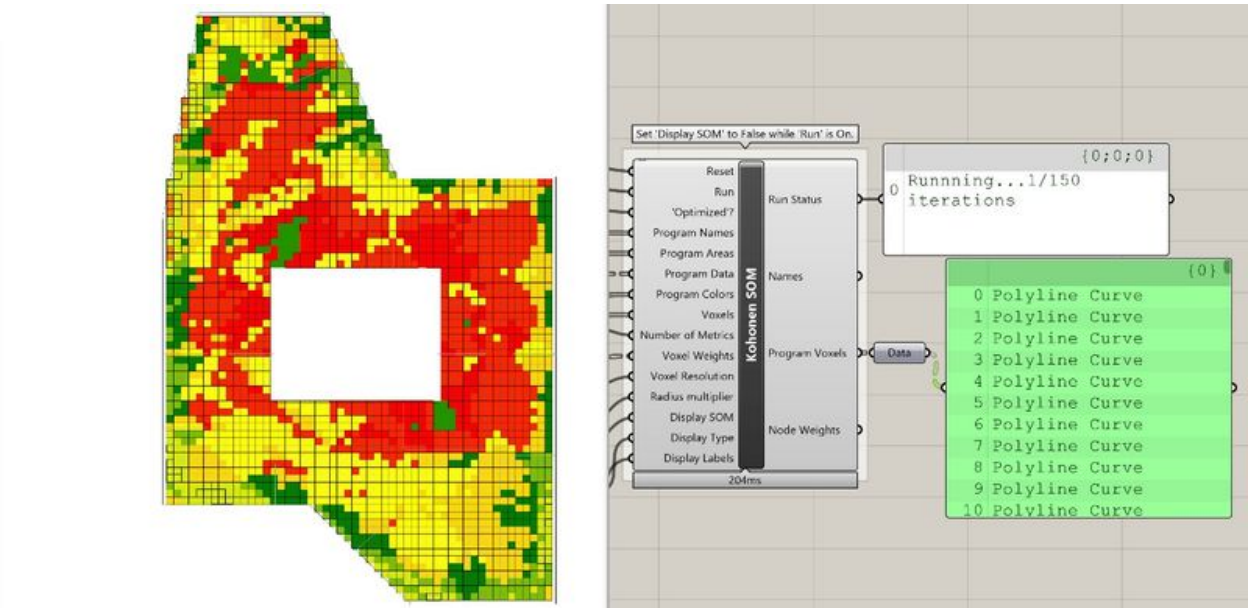


Figure 9. Interface of the Plan Bee library where we can a see a Kohonen self-organizing feature map (SOM) in action.
From: Marco Juliani, https://www.food4rhino.com/app/planbee.

More recently (Crow and other similar projects were developed around 2016), we have noticed a more structured approach whereby NN and machine learning solvers are being integrated with major software packages in the Architecture, Engineering and Construction (AEC) industry. Notably, Autodesk Revit 2021 includes a module for Generative Design that includes optimization tools, evolutionary solvers, and topological optimization libraries. This, along with the graphical programming interface Dynamo and the option to integrate these tools with text-based programming languages, represents a significant shift in the design industry. While 5 years ago (and before), computational design tools (including ML libraries) were only add-ons to main programs and needed some tweaking by the designers in order to work (Daher et al., 2019), such tools are now fully integrated into the standard industry software, hence, they are largely available to any designer.

In addition, interoperable online tools to create workflows are increasingly available to designers as well. For example, RunwayML offers several ML tools that can be flexibly connected to other apps. There is also the platform COMPAS that offers an open framework for designers to share libraries and APIs that work outside of any specific software so as to increase its interoperability.

The fact that main design software companies are embracing generative design models, as well as NNs and ML methods, offering them as a part of their standard tools should be considered positively. The inclusion of these tools in the designer's toolbox should encourage designers to experiment and better understand the potentials and limitations of intelligent systems applied to architecture and spatial practices. By the same token, this also calls for more data scientists to be involved with the design industry, bringing their expertise in this field. More and more, there seems to be the need for 'bridging profiles' that can work across data science and design.

SOFPs represent a great and equally exciting challenge for designers (and those interested in automatic generation of spatial layouts) for two main reasons. The first is about design as a discipline and field of study. These new data-driven approaches require designers to reconsider their role within the design process, from designers of buildings through drawings, to designers of scripts that will generate buildings. The second reason relates to specialization and expertise. Today, designers need to work with methods that originate in data science, using tools and a logic that is naturally outside of their traditional (academic and professional) training. Designers need, therefore, to rely heavily on colleagues from the data science community, embracing a new level of multidisciplinarity.

However, a caveat should be considered. This has to do with the transparency and intelligibility of these tools, and the extent to which designers are fully aware of the mechanisms that underpin the tools. In other words, these new tools make it very easy to obtain results from complex operations without necessarily the need of being aware of the entire process that brings those results. For example, some tools make it very easy to compute a linear regression given a large data set, but designers still need to understand whether a linear regression was the best statistical method in that particular case. In using APIs and libraries, designers need to put a certain level of trust in the tools and the developers who made them, assuming (and accepting) that the implementations of those tools fits the needs of their project.

Generalizing, in the case of SOFPs it is becoming increasingly easy to use tools that generate spatial configurations or architecture in general (either with NNs, optimization algorithms, etc.). It is important that designers (and any other users of these tools) understand the logic and process behind these workflows to ensure a good level of control over the obtained results. In other words, to generate a self-organizing plan is becoming increasingly easy, but who guarantees the spatial quality of the results? After all, these spatial configurations are generally made to be built, that is, translated into physical spaces where people live, interact, and subjectively experience the world around them. One way of addressing this concern may be to look at the quality of the training data that we use for our models. The designer's oversight of the entire design, modeling, and implementation process is very important. However, the final results of our floor plans directly depend on

the initial data used in the model. This aspect has an important impact on the final design, often more significant than the algorithm used to generate it.

5. A Final (Design) Note

Most of the machine learning methods mentioned in this article (especially the ANN) require large data sets (in the order of thousands) to be trained. In the current attempts we seem to concentrate more on where to find good and reliable data sets (the larger the better), how to train our models, and how to achieve a reasonable level of accuracy in the training (with the resulting floor plans that have a good resemblance to traditionally designed plans). One aspect that is still underdeveloped (and perhaps underestimated) is the quality of the training data. We often use data sets that reflect existing buildings, with most of them characterized by average and often mediocre architectural qualities. Often, the rationale used in finding an initial data set is not about the quality of the plans (or their architecture, social qualities, etc.), but more about the quality of the data (are plans comparable to each other, drawn well enough to be read without too many problems? are data easily available, how large is the data set? etc.). Our models are increasingly fast and accurate, yet they output the same quality that we give them as input. Strictly from a design perspective, this is not good, as (good) designers aim at achieving a new level of architectural quality with every new project (usually learning from previous ones).

In the examples presented in this article, we identify some promising cases that used a data set that included high-quality architectural projects. For example, Erik Swahn's work included drawings in his training data set from websites that publish projects by internationally renowned architects (plansofarchitecture and DeZeen). In his GAN Hadid, Sean Wallish trained his generative adversarial networks with photographs of buildings by Zaha Hadid Architects. Zaha Hadid is today regarded as one of the most talented recent architects. The two projects illustrate very clearly how GANs can be applied to architecture to produce new spatial configurations on the basis of high-quality projects. In their work, however, we are not able to see any final, full-fledged floor plan (like perhaps we see in Chaillou's). In GAN Hadid in particular, we can see how very well-designed buildings are separated into their salient components (that can be geometries and architectural features that may stand out from the overall building silhouette) and reaggregated into a new composition. These new synthetized spatial configurations are made of the parts of many buildings, yet they do not look necessarily surprisingly new. One may argue that the designer here trained the GAN so well that the algorithm generated a design as the original architect would have done. If the main objective of an NN is to learn from a training data set to generate new configurations based on rules inferred, this would be good for architects in general terms. This would be relevant in Chaillou's case, where his method can be applied to high-quality floor plans to generate extraordinary solutions (note that Chaillou trained his GAN with generic floor plans from the City of Boston).

As we all know, a good design is a complex combination of several factors that may include consideration of the context, the users, normative constraints, experience from past projects, and so on. A good design also includes aspirational elements like the designer's ambitions for the project, their own way of seeing and

interpreting the world through their ideology and culture, and the intention of improving people's lives (with an added quality of the physical environment, for example). These are aspects that, combined, encourage designers to improve their work every time. In this regard, one may argue that learning from existing projects (floor plans, spatial configurations, etc.) is simply not enough to achieve incremental architectural quality. By the same token, designers should try to integrate into their computational approaches to design elements ones that are so far not included and that are more subjective, aspirational, and striving for incremental improvement. At the same time, data scientists should develop ways in which ML methods can include the designer and users' subjective components that make design (in principle) better every time.

One of the next challenges, then, is how to produce SOFPs with perhaps smaller and more controlled data sets that generate an incremental level of spatial quality, including a certain degree of subjectivity (on the part of the designer) and that ensure a good public acceptance of the resulting building. In other words, how to encode those important aspects that differentiate a generic (and sometimes mediocre) building from outstanding architecture that can enrich people's lives?

Acknowledgments

I would like to thank Hongwei Wu (University of Hertfordshire) for his helpful comments on GANs and ML methods in general. I would also like to express my appreciation to the designers who kindly allowed me to include their images in this article, especially Stephanie St Loe, Tommaso Turchi, Joel Simon, Ao Li, Runjia Tian, Xiaoshi Wang, Lu Yueheng, Erik Swahn, Sean Wallish, Kyle Steinfeld, Stanislas Chaillou, Benjamin Felbrich and Marco Juliani. Finally, I would like to thank the editors of HDSR for their helpful, constructive, and prompt comments throughout the entire process and, in particular, the Editor-in-Chief Xiao-Li Meng for the encouraging comments and suggestions.

Disclosure Statement

Silvio Carta has no financial or non-financial disclosures to share for this article.

References

Alcin, M., Koyuncu, I., Tuna, M., Varan, M., & Pehlivan, I. (2019). A novel high speed artificial neural network–based chaotic true random number generator on field programmable gate array. *International Journal of Circuit Theory and Applications*, *47*(3), 365–378. https://doi.org/10.1002/cta.2581

Anderson, C., Bailey, C., Heumann, A., & Davis, D. (2018). Augmented space planning: Using procedural generation to automate desk layouts. *International Journal of Architectural Computing*, *16*(2), 164–177. https://doi.org/10.1177/1478077118778586

Benedikt, M. L. (1979). To take hold of space: Isovists and isovist fields. *Environment and Planning B: Planning and Design*, *6*(1), 47–65. https://doi.org/10.1068/b060047

Bird, J. J., Ekárt, A., Buckingham, C. D., & Faria, D. R. (2019). Evolutionary optimisation of fully connected artificial neural network topology. In K. Arai, R. Bhatia, & S. Kapoor (Eds.), *Intelligent Computing-Proceedings of the Computing Conference* (pp. 751–762). Springer. https://doi.org/10.1007/978-3-030-22871-2_52

Brookes, M. J., & Kaplan, A. (1972). The office environment: Space planning and affective behavior. *Human Factors*, *14*(5), 373–391. https://doi.org/10.1177/001872087201400502

Carta, S. (2020a). Algorithms are designing better buildings. *The Conversation*. https://theconversation.com/algorithms-are-designing-better-buildings-140302

Carta, S. (2020b). Machine learning and computational design. *Ubiquity*, *2020*(May), 1–10. https://doi.org/10.1145/3401842

Carta, S., St Loe, S., Turchi, T., & Simon, J. (2020). Self-organising floor plans in care homes. *Sustainability*, *12*(11), Article 4393. https://doi.org/10.3390/su12114393

Celento, D. (2007). Innovate or perish: New technologies and architecture's future. *Harvard Design Magazine*, *26*(Spring/Summer). http://www.harvarddesignmagazine.org/issues/26/innovate-or-perish-new-technologies-and-architectures-future

Chaillou, S. (2019, July 17). ArchiGAN: a Generative Stack for Apartment Building Design. *NVIDIA*. https://developer.nvidia.com/blog/archigan-generative-stack-apartment-building-design/?linkId=70968833

Chen, C., Chacón Vega, R. J., & Kong, T. L. (2020). Using genetic algorithm to automate the generation of an open-plan office layout. *International Journal of Architectural Computing*, *19*(3), Article 1478077120943532. https://doi.org/10.1177/1478077120943532

Chen, M., Yu, R., Xu, S., Luo, Y., & Yu, Z. (2019). An improved algorithm for solving scheduling problems by combining generative adversarial network with evolutionary algorithms. In *Proceedings of the 3rd International Conference on Computer Science and Application Engineering* (Article 10). https://doi.org/10.1145/3331453.3361639

Daher, E., Kubicki, S., & Pak, B. (2019, June). A survey on the parametric approaches and tools used in the design process by practitioners. In 2019 *IEEE International Conference on Engineering, Technology and Innovation* (pp. 1–7). IEEE. https://doi.org/10.1109/ICE.2019.8792606

Danka, T. (2020 May). Why are neural networks so powerful? *Towards Data Science*. https://towardsdatascience.com/why-are-neural-networks-so-powerful-bc308906696c

Davis, D. (2020, February 20). *Generative design is doomed to fail*. https://www.danieldavis.com/generative-design-doomed-to-fail/

Dhondse, A., Kulkarni, S., Khadilkar, K., Kane, I., Chavan, S., & Barhate, R. (2020). Generative adversarial networks as an advancement in 2D to 3D reconstruction techniques. In *Data management, analytics and innovation* (pp. 343–364). Springer. https://doi.org/10.1007/978-981-13-9364-8_25

Eastman, C. N. (1975). *Spatial synthesis in computer-aided building design*. Elsevier Science.

Eisenstadt, V., Langenhan, C., & Althoff, K.-D. (2019). Generation of floor plan variations with convolutional neural networks and case-based reasoning—An approach for transformative adaptation of room configurations within a framework for support of early conceptual design phases. In J. P. Sousa, J. P. Xavier, & G. Castro Henriques (Eds.), *Architecture in the age of the 4th Industrial Revolution—Proceedings of the 37th eCAADe and 23rd SIGraDi Conference* (Vol. 2, pp. 79–84). http://doi.org/10.5151/proceedings-ecaadesigradi2019_648

Fairs, M. (2019, October 22). Rise of artificial intelligence means architects are "doomed" says Sebastian Errazuriz. *dezeen*. https://www.dezeen.com/2019/10/22/artificial-intelligence-ai-architects-jobs-sebastian-errazuriz/

Ferner, J. N., Fischler, M., Zarubica, S., & Stucki, J. (2017). Combining neuro-evolution of augmenting topologies with convolutional neural networks. https://www.researchgate.net/publication/328939814_Combining_Neuro-Evolution_of_Augmenting_Topologies_with_Convolutional_Neural_Networks

Ferreira, B., & Leitão, A. (2015). Generative design for building information modeling. In B. Martens, G. Wurzer, T. Grasl, W. E. Lorenz, & R. Schaffranek (Eds.), *Real time - Proceedings of the 33rd eCAADe Conference* (Vol. 1, pp. 635–644).

Giedion, S. (1948). *Mechanization takes command – A contribution to anonymous history*. Oxford University Press.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems* (NIPS'14) (Vol. 2, pp. 2672–2680). MIT Press.

Goodman, G. (2019). A machine learning approach to artificial floorplan generation. *Theses and Dissertations—Computer Science*. https://doi.org/10.13023/etd.2019.391

Guo, Z., & Li, B. (2017). Evolutionary approach for spatial architecture layout design enhanced by an agent-based topology finding system. *Frontiers of Architectural Research*, *6*(1), 53–62. https://doi.org/10.1016/j.foar.2016.11.003

Hesse, Christopher. (2017, January 25). *Image-to-image translation in TensorFlow*. https://affinelayer.com/pix2pix/

Hillier, B., & Hanson, J. (1989). *The social logic of space*. Cambridge University Press. https://doi.org/10.1017/CBO9780511597237

Hodas, N. O., & Stinis, P. (2018). Doing the impossible: Why neural networks can be trained at all. *Frontiers in Psychology*, *9*, Article 1185. https://doi.org/10.3389/fpsyg.2018.01185

Hoos, H. H., & Stützle, T. (2004). *Stochastic local search: Foundations and applications*. Elsevier. https://doi.org/10.1016/B978-1-55860-872-6.X5016-1

Ibrahim, M. Y., Sridhar, R., Geetha, T., & Deepika, S. (2019). Advances in neuroevolution through augmenting topologies — A case study. *2019 11th International Conference on Advanced Computing* (pp. 111–116). IEEE. https://doi.org/10.1109/ICoAC48765.2019.246825

Isola, P., Zhu, J. Y., Zhou, T., & Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1125–1134). IEEE. https://doi.org/10.1109/CVPR.2017.632

Jabi, W. (2013). *Parametric design for architecture*. Laurence King Publishing.

Jo, J. H., & Gero, J. S. (1998). Space layout planning using an evolutionary approach. *Artificial Intelligence in Engineering*, *12*(3), 149–162. https://doi.org/10.1016/S0954-1810(97)00037-X

Juliani, M. (2020). *Plan Bee*. https://www.food4rhino.com/app/planbee; https://github.com/M-JULIANI/planbeeGH

Kalervo A., Ylioinas J., Häikiö M., Karhu A., & Kannala J. (2019) CubiCasa5K: A dataset and an improved multi-task model for floorplan image analysis. In M. Felsberg, P. E. Forssén, I. M. Sintorn, & J. Unger (Eds.), *Lecture notes in computer science: Vol. 11482. Image analysis* (pp. 28–40). Springer. https://doi.org/10.1007/978-3-030-20205-7_3

Karpathy, A., Abbeel, P. Brockman, G., Chen, P., Cheung, V., Duan, R., Goodfellow, I., Kingma, D., Ho, J., Houthooft, R., Salimans, T., SchulmanIlya Sutskever, J., & Zaremba, W. (2016). Generative models. *OpenAi.com*. https://openai.com/blog/generative-models/

Karras, T., Aila, T., Laine, S., & Lehtinen, J. (2017). Progressive growing of GANs for improved quality, stability, and variation. *arXiv.* https://doi.org/10.48550/arXiv.1710.10196

Karras, T., Laine, S., & Aila, T. (2019). A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (Vol. 2019, pp. 4401–4410). IEEE. https://doi.org/10.1109/CVPR.2019.00453

Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., & Aila, T. (2020). Analyzing and improving the image quality of StyleGAN. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 8110–8119). IEEE. http://doi.org/10.1109/CVPR42600.2020.00813

Khean, N., Kim, L., Martinez, J., Doherty, B., Fabbri, A., Gardner, N., & Haeusler, M. H. (2018). The introspection of deep neural networks-towards illuminating the black box-training architects machine learning via Grasshopper definitions. In T. Fukuda, W. Huang, P. Janssen, K. Crolla, & S. Alhadidi (Eds.), *Learning, adapting and prototyping —Proceedings of the 23rd CAADRIA Conference* (Vol. 2, pp. 237–246).

Kipf, T. (2016, September 30). *Graph convolutional networks*. GitHub. https://tkipf.github.io/graph-convolutional-networks/

Le Goff, L. K., Hart, E., Coninx, A., & Doncieux, S. (2020, July). On pros and cons of evolving topologies with novelty search. In *Artificial Life Conference Proceedings* (pp. 423–431). MIT Press. https://doi.org/10.1162/isal_a_00291

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), Article 14539. https://doi.org/10.1038/nature14539

Levary, R. R., & Kalchik, S. (1985). Facilities layout—A survey of solution procedures. *Computers & Industrial Engineering*, *9*(2), 141–148. https://doi.org/10.1016/0360-8352(85)90013-0

Li, A., Tian, R., Wang, X., & Lu, Y. PlanGCN Team. (2020) Graph to Plan. https://www.aoli.org/graph2plan

Liggett, R. S. (2000). Automated facilities layout: Past, present and future. *Automation in Construction*, *9*(2), 197–215. https://doi.org/10.1016/S0926-5805(99)00005-9

Liu, C., Wu, J., Kohli, P., & Furukawa, Y. (2017). Raster-to-vector: Revisiting floorplan transformation. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 2195–2203). IEEE. https://doi.org/10.1109/ICCV.2017.241

March, L., & Steadman, P. (1971). Spatial allocation procedures. In *The Geometry of Environment* (pp. 303–317). MIT Press.

Martin, J. (2005). Algorithmic beauty of buildings methods for procedural building generation. *Computer Science Honors Theses*, 4. https://digitalcommons.trinity.edu/compsci_honors/4

Merrell, P., Schkufza, E., & Koltun, V. (2010). Computer-generated residential building layouts. *ACM Transactions on Graphics*, *29*(6), Article 181. https://doi.org/10.1145/1882261.1866203

Michalek, J., Choudhary, R., & Papalambros, P. (2002). Architectural layout design optimization. *Engineering Optimization*, *34*(5), 461–484. https://doi.org/10.1080/03052150214016

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. *arXiv*. https://doi.org/10.48550/arXiv.2003.06988

Phelan, N., Davis, D., & Anderson, C. (2017). Evaluating architectural layouts with neural networks. In *SIMAUD '17: Proceedings of the Symposium on Simulation for Architecture and Urban Design* (Article 8).

Pitso, T. (2019). Shared futures: an exploration of the collaborative potential of intelligent machines and human ingenuity in cocreating value. In *Toward super-creativity-improving creativity in humans, machines, and human-machine collaborations*. IntechOpen. https://doi.org/10.5772/intechopen.85054

Plowright, P. D. (2014). *Revealing architectural design: Methods, frameworks and tools*. Routledge. http://doi.org/10.4324/9781315852454

Rutten, D. (2021). *The inevitable and utter demise of the entire architectural profession*. https://discourse.mcneel.com/t/unpublished-opinion-piece/118250

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., & Chen, X. (2016). Improved techniques for training GANs. *arXiv*. https://doi.org/10.48550/arXiv.1606.03498

Sandelin, F. (2019). *Semantic and instance segmentation of room features in floor plans using Mask R-CNN*. (PhD dissertation). http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-393348

Seehof, J. M., Evans, W. O., Friederichs, J. W., & Quigley, J. J. (1966). Automated facilities layout programs. *Communications of the ACM, 9*(7). https://doi.org/10.1145/365719.366401

Shaviv, E. (1987). Generative and evaluative CAAD tools for spatial allocation problem. In *Principles of computer-aided design: Computability of design* (pp. 191–212). Wiley-Interscience.

Simon, J. (2017). *Evolving floorplans*. https://www.joelsimon.net/evo_floorplans.html

Song, M., Zhang, J., Chen, H., & Li, T. (2018, February). Towards efficient microarchitectural design for accelerating unsupervised GAN-based deep learning. In 2018 *IEEE International Symposium on High*

Performance Computer Architecture (pp. 66–77). IEEE. https://doi.org/10.1109/HPCA.2018.00016

Stanley, K. O., & Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evolutionary Computation*, *10*(2), 99–127. https://doi.org/10.1162/106365602320169811

Steinfeld, K. (2019). *GAN Loci*. http://blah.ksteinfe.com/191026/gan_loci.html

Swahn, E. (2019, June 10). *Latent spaces*. Studio 9. https://www.studio9.arch.kth.se/author/erikswahn/

Tarabishy, S., Psarras, S., Kosicki, M., & Tsigkari, M. (2020). Deep learning surrogate models for spatial and visual connectivity. *International Journal of Architectural Computing*, *18*(1), 53–66. https://doi.org/10.1177/14780771198 94483

Terzidis, K. (2006). *Algorithmic architecture*. Routledge.

Turner, A., Doxa, M., O'Sullivan, D., & Penn, A. (2001). From isovists to visibility graphs: A methodology for the analysis of architectural space. *Environment and Planning B: Planning and Design*, *28*(1), 103–121. https://doi.org/10.1068/b2684

Veloso, P., & Krishnamurti R. (2021) Self-learning agents for spatial synthesis. In S. Eloy, V. D. Leite, F. Morais, & V. J. Vieira (Eds.), *Formal methods in architecture* (pp. 265–276). Springer. https://doi.org/10.1007/978-3-030-57509-0_24

Wallish, S. (2019a). *Counterfeiting daily: An exploration of the use of generative adversarial neural networks in the architectural design process.* Columbia University. https://doi.org/10.14288/1.0387289

Wallish, S. (2019b). *GAN Hadid*. https://www.seanwallish.com/gan-hadid

Wikström, D. (2018). *Me, myself, and AI: Case study: Human-machine co-creation explored in design.* Umeå University, Faculty of Social Sciences, Department of Informatics. http://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1223277&dswid=6515

Woo, W.L., 2020. Future trends in I&M: Human-machine co-creation in the rise of AI. *IEEE Instrumentation & Measurement Magazine*, *23*(2), 71–73. https://doi.org/10.1109/MIM.2020.9062691

Wu, W., Fu, X.-M., Tang, R., Wang, Y., Qi, Y.-H., & Liu, L. (2019). Data-driven interior plan generation for residential buildings. *ACM Transactions on Graphics*, *38*(6), Article 234. http://doi.org/10.1145/3355089.3356556

Zeng, Z., Li, X., Yu, Y. K., & Fu, C. W. (2019). Deep floor plan recognition using a multi-task network with room-boundary-guided attention. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 9096–9104). IEEE. http://doi.org/10.1109/iccv.2019.00919

Zheng, H., & Ren, Y. (2020). Architectural layout design through simulated annealing algorithm. In D. Holzer, W. Nakapan, A. Globa, & I. Koh (Eds.), *RE: Anthropocene, Design in the Age of Humans - Proceedings of the 25th CAADRIA Conference* (Vol. 1, pp. 275–284).

Zhou, J., Cui, G., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., & Sun, M. (2018). Graph neural networks: A review of methods and applications. *arXiv*. https://doi.org/10.48550/arXiv.1812.08434
