HYBRID ELEVATIONS USING GAN NETWORKS

by

Ali S. A. Q. Mohammad

A thesis submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Master of Arts in
Architecture / Information Technology

Charlotte

2019

Approved by:

_____
Chris Beorkrem

_____
Dr. John Gero

_____
Jefferson Ellinger

ABSTRACT

ALI S. A. Q. MOHAMMAD. Hybrid elevations using GAN networks. (Under the direction of CHRIS BEORKREM)

This project attempts to develop and test a method for generating one-sided hybrid exterior building elevations using two designer's base criteria and design ruleset as an input. Architects are using computational design to expedite the iteration process in an efficient manner. Optimization techniques utilizing genetic solvers within the architectural 3D model space have been around for the past decade. However, with the introduction of artificial intelligence the computational design niche within the architectural field can be expanded on. Generative Adversarial Networks (GAN) are used in this research to demonstrate artificial intelligence capabilities in the computational design realm. Can GAN networks dream of hybrid architectural elevations?

ACKNOWLEDGEMENT

I would like to take this opportunity to thank the chair, committee and students that have taught, guided and gave me the chance to build on my interests.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

Architects have conventionally used Computational Design to explore design iteratively. Computational design opens doors for the architectural field in terms of generating, testing and evaluating design models in a fast and efficient manner, thus leveraging the field in areas which require prolonged exploration and activity. Honorable examples include controlling complexity in design using a set of rules. Or if one is testing for sustainability, the ability to delve into the impact of sunlight exposure on a building. The possibilities are endless.

Imagine yourself, the reader, having to design everything by hand, drawing iteration after iteration of the same base with various tweaks to see how amalgamated solutions A, B & C would work. This is a drawn-out, time consuming process. However, the introduction of computation in the architectural profession, allows for more malleable results solving complex design problems and solutions by assigning variables and altering their properties.

"If a researcher is wedded to the authority inherent within a particularly instrumental view of rationality, as resident in a common sense logic, then the contingent and contested nature of professional authority may be irrelevant. For such a researcher the question of how authority is derived or promulgated in design is not a matter of discussion."

Richard Coyne (Logic models of Design, 1988).

Design problems in general, as well as those specific to the architectural profession are still considered to be wicked or ill-defined problems. These kinds of problems are often difficult to solve due to the contradictory, fragmented nature and their constantly ever-changing requirements over time. An imaginary blurry line is created between design problem and solution. There are countless solutions to one problem and innumerable paths to one solution as nothing is concrete. This also does not exclude where and from whom the solution is coming from. For example (to compare), this may be juxtaposed with "Software Studies", which frames the argument that data

and code have their own agency and therefore there is an ethical imperative to evaluate their source, impacts and accuracy. (Przybylski, 2018).

Instead of impeding the field, this ideology in fact furthers the architectural design field, as solutions are derived from numerous and various sources. For example, using principles in psychology or sociology to further understand the average person's motive when using X, Y or Z. This valuable information may in fact be used by an architect as a base to design a building. Architects are able to extrapolate philosophies, definitions, methodologies etc. from other disciplines and convert them into usable data points to fit to and solve their design problems.

With computational design, a number of tools and strategies have been created to assist the search and design exploration process such as optimization. With the exhilaration of being able to derive many design options from concepts like optimization, there are several flaws that go along with it. A conspicuous problem is that the strategy only produces ideal numbers (fitness), numbers based on a maximum or a minimum function. In other terms a predefined rule. Although this provides a powerful quantitative approach, it is limited because it only caters to the minimum or maximum number of possibilities and nothing in between. The gray areas are left unusable! So what happens when a designer makes an attempt to achieve results that are combinations of multiple variables (think gray area) ? Somewhere in the middle? Where the designer has to make compromises to find solutions which match multiple variables at the same time even when they are conflicting. We begin to avoid quality options based off a user's particular taste.

Although optimization workflows are a completely valid approach, they hinder and often prevent them from thinking of design solutions abstractly or outside of the box. Blindly pursuing such options may sometimes yield the optimum solution in some cases (quantitative). However, in other instances it creates problems in qualitative aspects of the design solution. Designers that are not as knowledgeable about these tools tend to use them and completely rely on them simply because it is the only offer available. The satisfaction fabricated from designers is predominantly due to utilizing the tool, and not necessarily achieving the precise results that they had hoped for,

but the closest they can find.

There seems to be no control in yielding a set of results based on what the designer's fundamental ruleset in design would be. This is essential as each designer's process differs greatly from one to another. each designer may conclude what "good" design is based on their individual experiences and what they learned from past projects. To be able to create narrow design solutions to a designer's qualitative taste, we must instruct, teach, apply descriptors and rules to our computational models on potentially infinite design solutions generated from optimization (and genetic solvers)(Sjoberg, 2017). Discovering qualitative emergent properties from unexpected search solutions is diminished through such strategies.

"Learning to restructure knowledge to produce novel results that not be achieved using the current knowledge." (Evolutionary Learning of Novel Grammars. John Gero, Sushil Louis & Sourav Kundu. 2006, p. 1 & 4).

So how would we go about solving this dilemma? How would we go about teaching the system about our design principles and traditions about our specific qualitative desires?

This project attempts to develop and test a method for evaluating exterior building elevations using a designer's base criteria and ruleset as an input. In addition to the genetic solvers, the use of a Neural Networks will create solutions based upon computationally perceived content. What is so special about a neural network?

The most fascinating part about neural networks is their ability to **learn** from traditional experiences and adapt to new ones using that new learned knowledge. This is done by a process called training, in which an expansive set of data is passed into the network as input. The passed data is iteratively looped around the network for it to learn. Much like humans, neural networks are impressive at evaluating patterns from the data they had learned from.

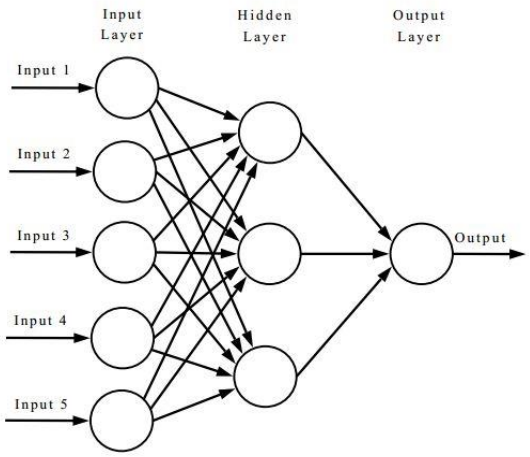


Figure 1. Diagram of a simple Neural Network

Consider this anecdote as an introduction to the experience of a neural network. This is also a way for the reader to obtain a different perspective on the subject through a scenario.

Imagine, a future where you, <insert name> get hired for a new job in a state-of-the-art futuristic architecture firm. You are introduced to the company's owner who attempts to comfort you into your new position. He says, "hey <insert name>, welcome to your new office workspace." He hits a reset button that resides on the wall from outside of the office and then opens the door and tells you to take the first step in. As soon as you go through the door, you notice how blank, dull and empty this office space is. John, the office assistant sits in the corner of the room and takes a good look at you from top to bottom but does not say anything.

You ignore him/her and continue setting up your new mint office. "What's going on? Why is it so hot in here? Not to mention a little dark. I wish I had windows in the room at least" you ask and tell yourself. You go to configure the thermostats and lights in the room and then go to sit down on your fully customizable chair, and you adjust the height and lean back.

You turn on your computer and lay down your notebook with information on how to decorate your new office. You then decide it's been a worthwhile day and leave the office, forgetting your notebook wide open on your desk. John is still in the room and while you are gone he decides to look at what you've done in the room, plus check out your notebook. So what information does

john have right now? He's got your weight, height, comfort levels, light intensity information. He looks at your notebook and you seem to be into floral patterns that you have designed for a wallpaper on one wall of your office and how the room is laid out. John has now learned about your preliminary actions.

You come back the next day and notice a few things are out of place and that the temp in the room feels great, just the way you like it and the room doesn't feel too dark. You notice a wall partition that had been pulled apart from the original walls of the office and moved into the office space to a location. You notice a designed floral pattern on the wall the way you wanted it and you're wondering how is that possible when you haven't told anyone about it? Not to mention there is now an opening within the wall that you haven't seen before to let natural light in. You are then startled by a sudden voice in the room "<insert name>, welcome, I am John, your personal assistant and have changed the room according to your needs". John is not a human being, John is an artificial intelligence agent with a neural network brain, capable of taking in your personal data, learning about your traits and habits and working around your every desire and goal without you having to utter a word. Ladies and Gents, welcome to the world of neural networks.

CHAPTER 2: BACKGROUND

Decision Making

"The linear model of design thinking is based on determinate problems which have definite conditions. The designer's task is to identify those conditions precisely and then calculate a solution. In contrast, the wicked-problems approach suggests that there is a fundamental indeterminacy in all but the most trivial design problems where, as Rittel suggests, the "wickedness" has already been taken out to yield determinate or analytic problems. To understand what this means, it is important to recognize that indeterminacy is quite different from undetermined. Indeterminacy implies that there are no definitive conditions or limits to design problem" (Wicked Problems in Design Thinking. Richard Buchanan. 1992 p. 15 & 16).

To understand the essence of this paper, providing a brief overview to Rittel and Webbers work, which emphasize the nature of wicked problems is a sufficient introduction to the whole background. In the quote above, Richard Buchanan makes a clear distinction between a linear design thinking problem juxtaposed with a wicked design problem. Why is it crucial to mention the wicked problem context? Because of wicked problems, the architecture profession can synergize and exercise with various other fields, such as computer science in order to solve a problem. In design computation, exists a realm where humans, machines work collectively, and a mutual trust is demonstrated when searching for a qualitative design solution. This highlight is an evident trajectory and emergent property that resulted from the wicked problem philosophy.

"What is an assemblage? It is a multiplicity which is made up of many heterogeneous terms and which establishes liaisons, relations between them, across ages, sexes and reigns – different natures. Thus, the assemblage's only unity is that of a co-functioning: it is a symbiosis, a 'sympathy'. It is never filiations which are important, but alliances, alloys; these are not successions, lines of descent, but contagions, epidemics, the wind.

In this definition, two aspects of the concept are emphasized: that the parts that are fitted together are not uniform either in nature or in origin, and that the assemblage actively links these parts together by establishing relations between them."

"These contributions of the concept of emergence can help us understand the other important characteristic mentioned in the opening quote: that social wholes must be considered to be peripheral or to exist alongside their parts."

(Assemblage Theory. Manuel Delanda. 2016 p. 2). In the works of Deleuze and Guittari, as well as the addition to Manuel Delanda (added emergent properties to Deleuze and Guittari's assemblage theory), present assemblage theory. Although the authors explain assemblages in the social context, the same can be applied in the design computation niche. Architects working alongside artificial intelligence agents can be considered an assemblage, even though both are not uniform in nature or origin. In their individual characteristics and properties, they are fundamentally distinct, but when they coalesce, novel emergent properties are created to solve a particular design problem. The merge of Architect and A.I. (a neural network) yields a constant dialogue of exploration, variation and evaluation in this case. The prominence of such assemblages should not be treated as individual entities, as they are today, rather as whole. This ideology, if perceived correctly, empowers the relationship between Architects and A.I operating congruently.

"Accepting that software is becoming a material used in the production of the built environment and that we want to build ethically-minded understandings of the impacts of design work.

This framing supports a reconceptualization of ethical literacy in architectural projects embedded with custom software. By including soft materials, such as data and algorithms, in our conception of a project's material assembly they implicitly become a part of the matrix defining the works' ethical dimensions. This necessitates an expansion of knowledge and methods to support a more complete engagement with computational components embedded in such work."

(Soft Materials: Assessing Architects Roles as Ethical Producers of Digital Technology. Maya Pryzbylski. 2018)

In addition to the conversation on assemblages, the works of Maya Pryzbylski acknowledges that custom computational components such as custom algorithms are being a part of everyday convention and built environment. Designers develop code nowadays to achieve a diverse range of results. An example is implementing developed code on building components such as facades with the aim to make them responsive and interactive. Other examples include data visualization, which is becoming a common norm amongst designers. There are two parts to this paper by Pryzbylski. One discusses the role of the architect and his/her position in the fabrication of technology. The second part of the paper is critical because it reveals that the constituents of those technologies, like data and code bundles are being embedded within the framework and matrix of requirements for architectural projects. The realization of this notion is vital to grasp as the profession progresses further. In the case of this paper, data is becoming an essential piece in the assemblage between architects and artificial intelligence.

"Biology has traditionally started at the top, viewing a living organism as a complex biochemical machine, and worked analytically downwards from there - through organs, tissues, cells, organelles, membranes, and finally molecules - in its pursuit of the mechanisms of life. Artificial Life starts at the bottom, viewing an organism as a large population of simple machines, and works upwards synthetically from there, constructing large aggregates of simple, rule-governed objects which interact with one another nonlinearly in the support of life-like, global dynamics. “ (Virtual Environments and the Emergence of Synthetic Reason. Manuel Delanda. 1993).

Manuel Delanda discusses the importance and impact of virtual environments in the research fields. The snippet of text above is Delanda explaining and contrasting between organisms in biology and artificial life. This introduces basic terminology for the purpose of this

paper, and will make much more sense when processes in neural networks are elaborated in the methodology section later on.

The top bottom approach suggests an analytical approach and breaking down larger components into smaller ones to comprehend the system and its subsystems. The insight gained on one hand is further polished and understood through a reverse-engineering methodology. In neural networks, the training aspect of the whole process is a top-down approach as it takes data as input and trains, recognizing patterns the same way humans do to make sense of any type of collective data.

Delanda states that artificial life starts from the bottoms up. The bottom up approach suggests looking at the top down approach from the other end, where minute individual pieces begin to gather and formulate into larger, complex systems. This ideology allows for emergent properties. New properties and characteristics generated through the assembly of smaller elements. These same elements once expressed their properties on the individual level, however now that they are compounded, they exhibit various novel properties. For example cells in the body that organize into tissues to form muscles and work jointly to execute a specific function. In the bottom up approach, systems may behave differently or independently on their own, yet when coupled together, begin to reveal disparate behavior and in other words, becomes an assemblage. In neural networks, it is mentioned above that the training process is a top down process, and at the same time, when the network is in the output phase, it would be considered a bottom up process (in unsupervised learning). Below is a closing quote by Delanda to provide another perspective.

"The basic point is that emergent properties do not lend themselves to an analytical approach, that is, an approach which dissects a population into its components. Once we perform this dissection, once the individuals become isolated from each other, any properties due to their interactions will disappear. What virtual environments provide is a tool to replace (or rather, complement) analysis with synthesis, allowing researchers to exploit the complementary insights of population thinking

and nonlinear dynamics."

(Virtual Environments and the Emergence of Synthetic Reason. Manuel Delanda. 1993)

CHAPTER 3: NEURAL NETWORKS

A Brief History

"Because of the 'all-or-none" character of nervous activity, neural events and the relations among them can be treated by means of propositional logic."

(A Logical Calculus of the Ideas Immanent in Nervous Activity. Walter Pitts & Warren McCullough. 1943)

The mathematics of neural networks were introduced in 1943, as Walter Pitts and Warren McCullough cogitated about the nervous system and its activity. A remarkable tale of two men that collaborated on the earliest views of artificial intelligence. Not only did they work on the foundation of neural nets, but also the preliminary computational approach to neuroscience and design of modern computers in the logical sense. The initial ideas formulated when Walter Pitts pondered about a model to the brain utilizing Leibnizian logical calculus. What led Pitts and McCullough eventually to reach the topic of neurons was from the Principia Mathematica. The works of Alfred North Whitehead and Bertrand Russell who inadvertently introduced the fundamental language of programming people know today. The use of fundamental operations in logic such as boolean "true" or "false" statements. And later, other operations like "or", "and" & "not". When McCullough thought about neurons, the major and heart of the discovery to him was that neurons fire electrical signals only after a certain threshold had passed it. The net signal comes from the combination of neurons nearby that had been accumulated, however not fired off yet. The reader must understand this key concept and how it was modelled in order to grasp the nature of neural networks in this paper.

Implementation in Design

Game Design

Ever since Walter Pitts and Warren McCullough introduced the primary components of a neural network (circle nodes and cumulative weights) and the mathematical model of a neuron, there

have been others which have expanded on their notable studies. Subsequent, improved neural network models are gradually being applied to a number of various design areas such as in gaming.

One of the earliest cases where a neural network implemented in gaming design involved a simple game of checkers. Arthur Samuel's unprecedented game of checkers brought attention the preliminary idea of self-learning machines. With enough "experience", these machines, Samuel deduced, could learn to outperform humans. Arthur also mentioned two general methods in his paper, which are familiar today in the artificial intelligence sphere.

"One method, which be called the Neural-Nets approach, deals with the possibility of inducing learned behavior into a randomly connected switching net for its simulation on a digital computer as result of a reward-and-punishment routine." (Samuels, 1959)

Here, Arthur Samuel provides an early description of reinforcement learning, which falls under the unsupervised learning category in machine learning. To put this in brief terms, unsupervised learning involves a machine taking inputs, generalizing that data and completely generating its own outputs based on what rules and variables of that data it had previously learned about. What differentiates unsupervised learning and reinforcement learning is that the latter incorporates a reward and punishment system simply to guide and teach the agent (similar to human beings learning) to the correct path. To put this into extremely simple pseudo code for the reader (please note that there are many other factors that go into this, however this is for the reader to grasp the reinforcement learning fundamentals):

For each action:

If agent performs an action in environment correctly (set by programmer what the rewards are in the environment are):

+1 as reward

Else:

-1 as punishment

"A second method, and much more efficient approach, is to produce the equivalent of a highly organized network which has been designed to learn only certain specific things." (Samuels, 1959)

The second method mentioned by Arthur Samuels would in today's terms be called supervised learning. This approach suggests that the output is a **known** factor. The system understands its inputs and has been trained to produce a specific known output, hence it cannot generate its own disparate outputs, only the domain in which it has been subjected to. In the case of Arthur Samuel's game of checkers, it made more sense to use supervised learning ever more so, since all the outputs and moves in checkers are known in relevance to each piece's position. Based on position, the system can "think ahead' by calculating all the possibilities, a "tree" of moves it can derive with the goal to win. Below is an image from Arthur Samuel's paper demonstrating the board from its initial position and considering a few possible moves ahead.

Figure 2. Game of Checkers move possibilities

There are three cases in gaming to touch on, in which artificial intelligence agents have defeated humans who have spent years and years, becoming masters at their respective games. The reason

for these three highlights is to instill an inevitable vision to the reader which the future would yield. When a human being had spent most of his or her years mastering a subject or skill, a machine could in turn, condense these training years into days or even hours.

IBM's Deep Blue (chess 1996-1997) & IBM's Watson (Jeopardy 2011)

Two of IBM's iconic artificial intelligence agents which have been implemented and well-documented in game design. Although both are "A.I. agents", both exhibit variation in their approach to tackle a problem. Deep Blue took part in the game of chess. Arthur Samuel actually considered chess as his initial environment for experimentation, however chess is far more complex than checkers. Chess requires one to comprehend what each specific chess piece's function is, whereas in checkers, all elements share the same function. As opposed to IBM's Watson architecture, Deep Blue emphasizes on optimization techniques to achieve its goal in defeating a competitor. Two major algorithms are Alpha-beta pruning and Minimax algorithms are embedded in Deep Blue's software. The Minimax algorithm is designed to minimize a worst-case scenario loss and maximize the best scenarios, the algorithm's name speaks for itself. The alpha-beta pruning algorithm which sits on top of the minimax function, allows for faster computation. This is achieved by eliminating possibilities that had already been seen or discovered. In a tree of decisions, for example, some branches would be already eliminated depending on the situation. In other words, we do not need to look back at the opponent's bad moves (F. Hsu, 1999).

IBM Watson's purpose on the other hand was to become a Jeopardy competitor, and not just any, but one to attempt to defeat the top-ranking competitors in that game show. Although former grandmasters, Ken Jennings and Brad Rutter gladly took on the man vs machine challenge. As opposed to chess, Jeopardy is a game of trivia, testing the knowledge of competitors over a large realm of natural language questions (Lally, 2011). Watson's system architecture is named DeepQA, which was developed by IBM's research team. Answering Jeopardy questions isn't an easy homerun at the ballpark but may require complex natural

processing and specific sentence structure to denote any type of connotation. Watson needs to be able to not only gather the right context, parse data for the right answer, but also put together the right sentence in a timely fashion, which is yet another crucial factor in the game. As opposed to IBM Deep Blue's optimization strategies, IBM Watson utilizes a neural network and trained on an immensely vast library of general knowledge data.

Alpha Go (March, 2016)

Another board game by the name of "Go". As opposed to checkers and chess, Go would require a vast search space of solutions for any artificial intelligence agent. Although each physical, oval piece functions the same way, the board is larger than a checkerboard. An A.I. agent must also understand how to win the game and formulate its own ways of doing so. AlphaGo's brain is equipped with a deep neural network, along with a tree search function, bolstering vast search spaces findings in a fast and efficient manner. The power of a neural network can be recognized in this recent example. Authors of the article "Mastering the game of Go with deep neural networks and tree search" proudly claim that their neural network does not possess a lookahead search, which that in itself is already fascinating. The networks employed in AlphaGo are actually a combination of a convolutional neural network (computer vision and image analysis & recognition to look at the board), a feed forward neural network using a supervised learning approach (data trained by looking at expert professional players and understanding their output. Named this network "SL Policy Network") and reinforcement learning (interacting in the environment on its own and learning to play the game ("RL Policy Network") and in addition to that, a Monte-Carlo tree search to simulate hundreds and hundreds of random games for self-playing (to gain as much perspective as possible)(Silver & others, 2016). With the amalgamation of all these parts, AlphaGo was able to defeat Chinese-born European Go champion Fan Hui 5 games to 0, marking the first of its kind, and simultaneously, becoming extremely exciting for what's to come.

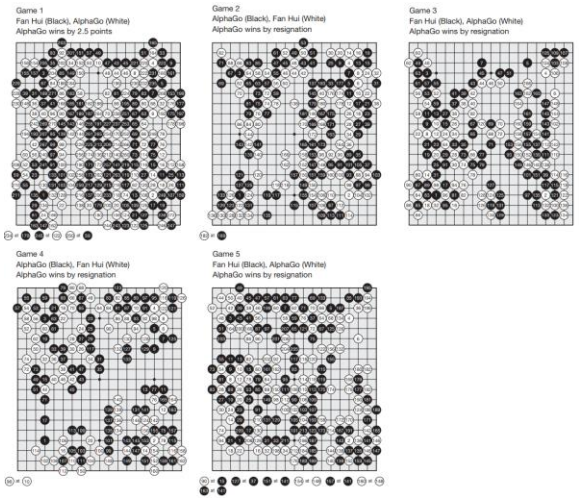


Figure 3. Final boards of all 5 games with Fan Hui

Dota 2 (June, 2018)

The final precedent in gaming design is a PC game known as DOTA2. This type of game is known as a MOBA (Multiplayer Online Battle Area). The game requires participation from 5 players versus 5 players in a large confined space or in other words “map”. This is coupled with a camera angle looking from the top right or top left. The complexity in a type of game like this is unrivaled, as coordination is a large portion to success and to have 5 A.I. agents collaborating is by itself an extremely demanding feat. In the environment, the goal to win is to take down enemy turrets in order to advance into the enemy base and to demolish their “nexus” (another way of saying main base / turret). The consequences of a character dying a single time in the game rewards the enemy with a somewhat massive lead, denoting the game’s fragile and sensitive nature. OpenAI was funded by entrepreneur Elon Musk to see if A.I. agents could in fact defeat humans at their own games.

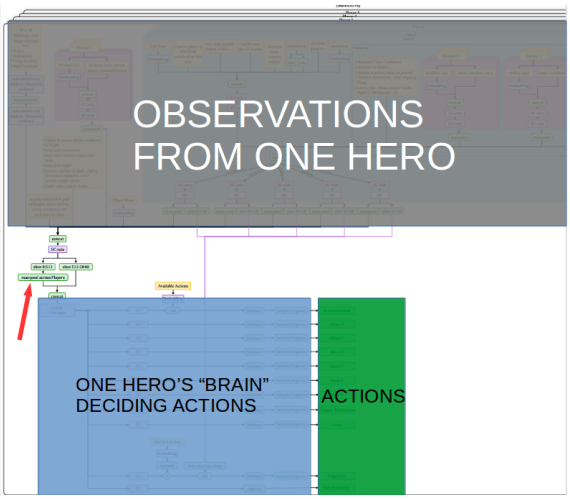


Figure 4. Diagram and captions for the overall structure for OpenAI

The diagram above (pulled from openAI's blog website) is a simplified version of the entire model architecture, consisting of a single-layer, 1024-unit LSTM (a type of neural network short for "Long-Short Term Memory", part of a recurrent neural network).

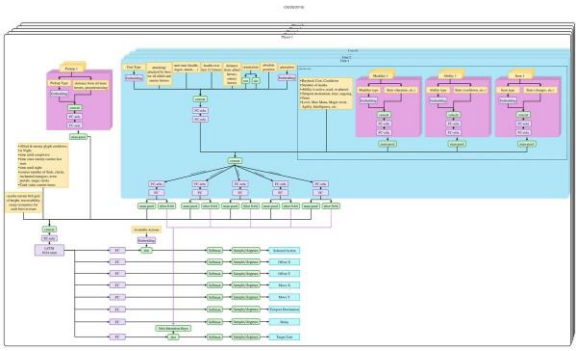


Figure 5. OpenAI full architecture reveal

The image directly above (also from the openAI blog website) is the full set of components assembled. There is a constant dialogue between states, the neural network and the observations received as input to be processed.

Architectural Design

In terms of Architectural design, the implementation of neural networks or an artificial intelligence agent has yet to make a substantial impact (Khean & others, 2018). There are a few reasons that stem from that statement. One, there is a conspicuous knowledge gap difference that

needs to be accentuated on. The subject of deep learning and neural networks is compounded and has yet to reach its prime, even in the Computer Science profession itself. Another problem is that architects do not fully comprehend the powerful predictive and self-learning quality of a neural network, let alone where and how to apply it (problem domain).

In design computation, a relatively new area that bridges between Architecture and Computer Science, a few tools have been created that attempt to implement deep neural networks and solve specific problems. A set of digital architectural components will be introduced in order to situate the reader appropriately.

"Grasshopper" is a visual programming extension which enables designers create scripts through several predefined nodes that may be connected and stitched together to generate a model. Grasshopper follows the same logical approach needed in programming, a linear, step by step procedure. For example, if a designer were to draw a line, they would do so in a CAD type environment by clicking one point in space, followed by clicking on another point in space and the software would create the line, however, in Grasshopper one would need to define the points in space through their XYZ coordinates, in other words, through numbers. This is followed by adding another node called "line" which requires two points to create the line. The 3D modelling environment is more intuitive than the visual programming approach. Grasshopper is an add-on to "Rhinoceros", which is mainly a 3D computer-aided design (CAD) modelling software used to create detailed models that may be rendered into photorealistic images.

Grasshopper is needed to operate machine learning plugins and there are, according to a paper written by Lorenzo Greco (2015), four documented plugins that enable designers to interact with and tune neural networks visually. The four plugins are listed as follows: 1. "Dodo", 2. "Crow", 3. "Owl" and 4. "LunchBox".

"Dodo" (2015), pioneered by Lorenzo Greco, provides a set of "nodes" for Grasshopper to allow for machine learning capabilities. There are also nodes that deal with tensor and matrix conversions. A tensor, to put it in simple terms, is a collection of vectors, required for a neural

network to understand the data received. The difference between these plugins is influenced by the .NET framework that the extension has been built on, as each framework is tuned dissimilarly. One of the main features of Dodo is the ability to perform non-linear and constrained optimization, as well as incorporate artificial neural networks (Doherty & others, 2018). Another plugin by the name of "Crow" (2016), is also capable of learning, however the expertise it provides is largely to do with SOM's (Self-Organizing Maps) and Crow provides the necessary nodes for Grasshopper to enable its functions. SOM's are a type of unsupervised learning approach and was developed in the 1980's by Tuevo Kohonen (Kohonen, 1999). The form of architecture in SOM's are disparate in behavior and goal. SOM's also utilize the K-means cluster algorithm, essentially to quantize vectors. The "centroid" of a supposed k-cluster ultimately becomes the new mean, and from that, the system **adapts** to these new positions the process is then repeated. Below is an image that displays a mesh, stretching, conforming and adapting to a set of points floating in space. The concept of SOM's is fundamentally what differentiates this plugin from the other three mentioned plugins.

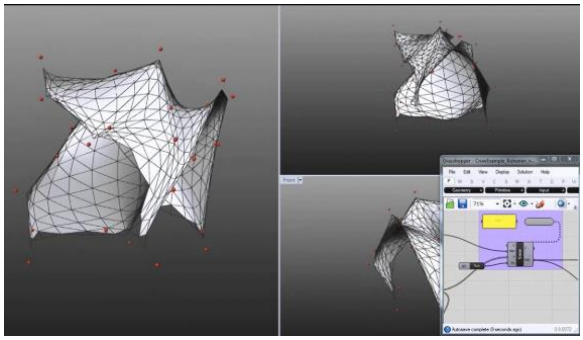


Figure 6. Image displaying SOM conforming to points in space

"Owl**"** (2017) and "LunchBox" (2017) also provide a set of predefined components for machine learning and neural network application. LunchBox is a slightly more comprehensive toolset, running the gamut from predefined panel creation tools, as well as fundamental and basic tools for generating neural networks. And finally, Owl provides a comprehensive toolset for creating neural networks, however it only deals with tensors as the major data type and provides

nodes to convert any type of data into a tensor set. Any data type (number, string or GUIobject (e.g. point in space)) is converted into a tensor, and then run through the neural network to compute information. The output would then be “detensored”, which is the conversion from tensor, back to data. Each one of these plugins cater to the same topic, however their approach and frameworks are various. One problem for all of these plugins is the fact that their frameworks are slightly outdated, with “Accord.Net” being the latest framework for the implementation of machine learning (2014).

Optimization (Galapagos)

There are numerous tools that have become accessible for designers seeking to incorporate genetic algorithms in the architectural design process, prominently, Galapagos, which runs on the previously mentioned Rhinoceros + Grasshopper combo (Rutten 2013). Galapagos allows the designer utilizing visual scripting platforms to use genetic algorithms without the need to program the complete process, which requires a vast amount of time.

CHAPTER 4: METHODOLOGY

This experiment goes through three main phases. The first phase involves two designers. Two distinct designs with a few static elements are constructed, however they were self-simulated in the case of this experiment. The second phase is to collect a 2D based dataset consisting of small resolution images of 150x150 pixels. This phase will be expanded on in the next subsection. The third phase is to use the collected dataset of images and feed it through a GAN (Generative Adversarial Networks), train the network and output new AI generated images based on what the model has learned.

1. Constructing two designs

In this initial experiment, the designs were loosely inspired from one-sided tall building elevations in New York city. The base 3D models were generated in the 3D modelling software "Rhinoceros", and those models in the experiment are 6'X 6' X 100' (10 story building). The two designs are to exhibit static and dynamic features, as well as contrasting styles.

Design 1

The **key feature** of the first design is to display a sense of rigidity. This entails using a grid like pattern with repeating frames. Another feature is presenting an overall vertical proportionality. The base of the generated building is to be amalgamated of simple rectangular repeating elements which represent a glass wall with mullions. The rule for the base is for it to be the static element in both designs. The only dynamic element which is to be optimized by Galapagos using "Grasshopper" is the frame extrusions. The extrusions are set to a max length and at random for each frame. This is to allow for variation and to exercise the GAN network as the training is performed on it.

Below is the image of the proposed design:

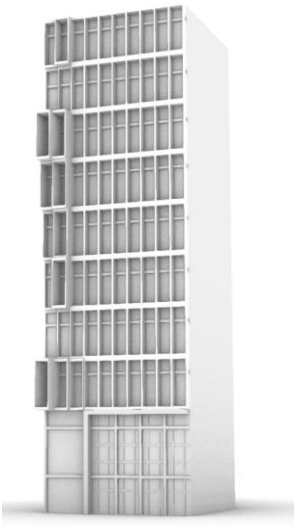


Figure 7. Design1 with rigid features

Design 2

The **key features** for the second design exhibit smoother, organic and a freeform quality to it. A large organic surface is generated and contoured in a striped manner to highlight the horizontality in proportions. Again, as mentioned before, the base is static and exactly similar to the Design 1. The reasoning is to test if the networks can actually study the positioning of the base element and can reconstruct it in both designs. The dynamic elements of the whole design are the culling of the stripes at random. This reveals some basic structure behind the culled pieces.

Below is the image of the proposed second design:

Figure 8. Design2 with organic features

This set of designs are simulated to test if the GAN network can learn specific features, understand the patterning behind the designs and attempt to combine features together to form a hybrid style building from the domain.

2. Creating the dataset

The second phase is comprised of three main steps:

1. "Optimize" the model to produce iterations.
2. Record these iterations through a spreadsheet in Microsoft Excel.
3. Use Anemone (Grasshopper extension) to loop through iterations and save each elevation to a folder

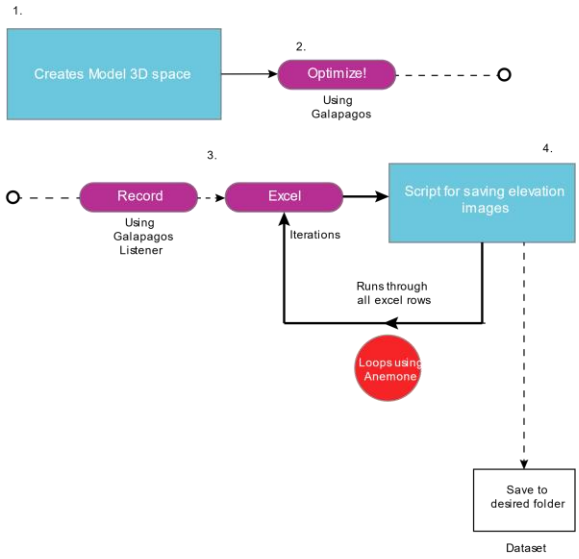


Figure 9. Diagram of dataset collection process

a. Given the generated model from both designers- "optimize" the model using the Galapagos genetic solver. The solver node is connected to a set of genomes, or in other terms, the variables that are to be changed and optimized. The random component from Grasshopper is used on the dynamic elements from both designs. Below displays an image directly from the Grasshopper environment and the connections made from Galapagos (pink node) and random. The genetic solver will produce iterations of each model by modifying the genome numbers. This is needed to reinforce the GAN network later.

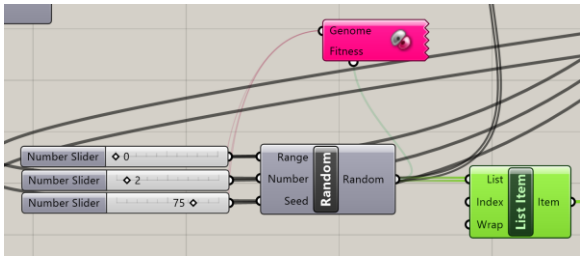


Figure 10. Part of Grasshopper script with random component

b. To record all the iterations, another Grasshopper extension is used, called "Galapagos Listener". This extension is a tool from computational design firm "Core Studio" under the "TT Toolbox". Listener is a component that simply records the values as the genomes are being optimized, in a data tree. The values recorded are then connected to an output Microsoft Excel

component from the same extension, “TT Toolbox”. Below is an of a set of recorded numbers with column A representing the fitness of each iteration and the other three columns representing the values from the random component.

	A	B	C	D
49	0.389524	35	35	1
50	0.434041	39	35	1
51	0.478558	43	35	5
52	0.503819	9	37	7
53	0.511318	19	58	9
54	0.533894	2	91	17
55	0.607474	1	17	9
56	0.615779	11	37	13
57	0.762127	4	43	7
58	0.914991	34	58	1
59	0.914991	34	58	3
60	0.932902	8	62	13
61	0.93721	1	54	10
62	0.941902	35	58	6
63	0.941902	35	58	4
64	0.941902	35	58	2

Figure 11. Excel sheet with fitness and genome values

c. “Anemone” is another Grasshopper extension with the aim to perform a loop function using a “start loop” and “end loop” nodes within the Grasshopper environment. This is utilized in node form instead of the native code written form. What Anemone does in this case is iterating through each row in the spreadsheet. This is coupled with a small script written in VB (visual basic) which contain Rhinoceros commands in code format. The script opens a new Rhino viewport, sets the elevation to “Front”, saves and crops a screenshot. This is set and done with a count which can be set along using the Anemone components.

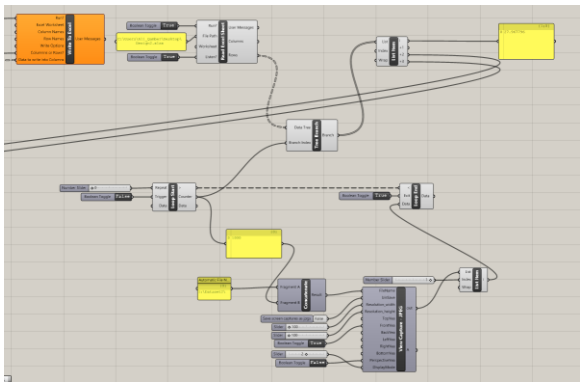


Figure 12. Grasshopper script that takes snapshots of model space

Once all three steps are completed, images of 500x500 are stored in the designated folder. The initial experiment had roughly 5000 images. Below is an example of the images generated:

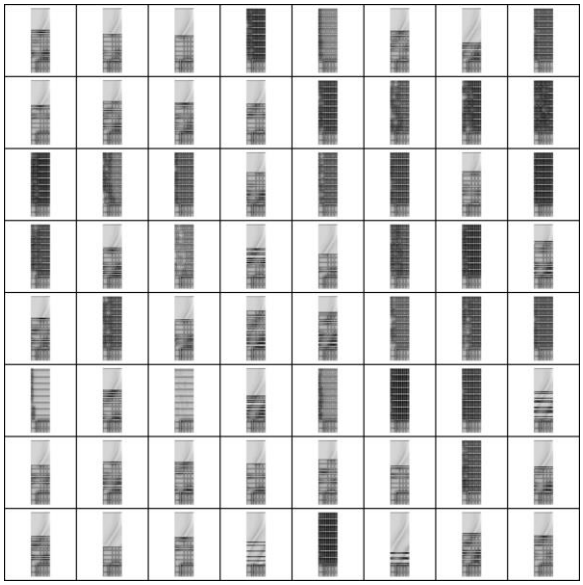


Figure 13. dataset created

3. Generative Adversarial Network

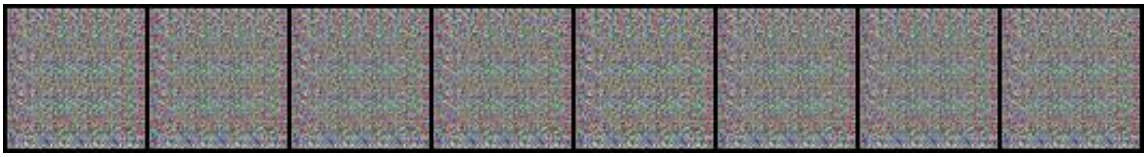
The third and final phase is to run the dataset through a DCGAN (Deep Convolutional Generative Adversarial Network). The DCGAN architecture convolves and breaks down the images into smaller pieces to work with. GANS, in general contain two major pieces. One is a neural network called a “Generator” and the other a “Discriminator”, and both networks are needed for a GAN to work. The generator takes in noise as input and generates samples, while the discriminator determines whether these generated images are A.I generated or not.

The loss function used and is necessary for GANs to work is a BCE Loss which stands for Binary Cross-Entropy loss. It is a Sigmoid activation as well as Cross-Entropy loss together. It returns the 0 or 1 classification needed for the discriminator part of the model to determine if an image is fake or real. But the real reason behind GAN networks is their generative ability, not the discriminator’s 0 or 1 classification.

CHAPTER 5: RESULTS & DISCUSSION

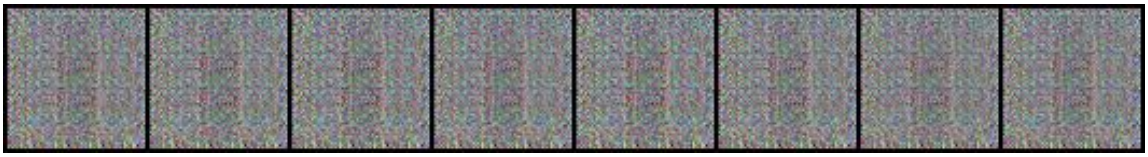
Initial Results

Below are the preliminary results and images of resolution 64x64 pixels. The original images were at 500x500 pixels.



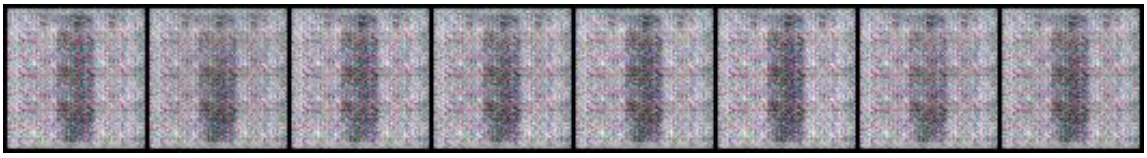
Epoch1

The first cycle of GAN network training on the dataset being fed into it. It is only natural to see noise on the first epoch since it is cycling through all the images for the first time. Since the outcomes are at 64x64 pixels, training goes by quickly.



Epoch2

While the second epoch is for the most part noise, there is a slightly visible figure in the middle. This gives hope that something might appear, however all that is seen right now is just the outline of the figure as the network is still training and is in the early phase of the operation.

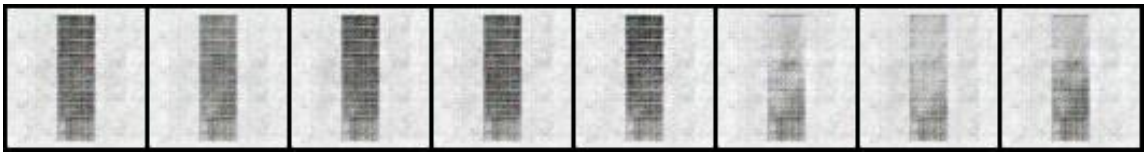


Epoch3

The noise in the third epoch is clearing up and the elevations are beginning to become visible, however there is no telling how the features might turn out just yet. For the initial results to become visible by epoch3 is good news. Again, the reason this is network is producing results early in epochs is because of the image size of 64x64.

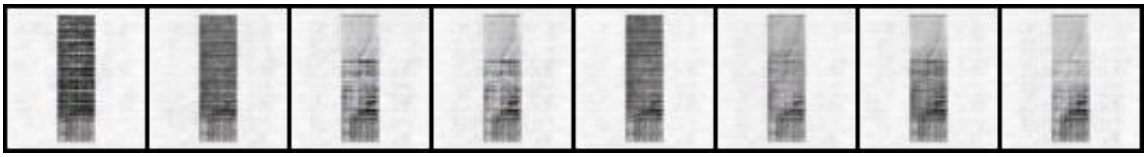
Epoch4

Getting to this point, epoch4 is displaying promise that results are appearing. As one might tell, the network has not yet reached a state of stability just yet. But again, the fact that there exhibit results at just epoch4 is good to see. Also, as opposed to epoch3, the variety of designs in epoch4 are now discernable.



Epoch5

After epoch4, the networks are attempting to fully stabilize the image. The noise is clearing up and the images are starting to become worthy of comparison.



Epoch6

At epoch6, clearly the networks have stabilized. The white background is brightening up and now even the features of each design are visible. As much as it exciting to see these results, there are a few pros and cons to them.

Pros:

What is fascinating about this is witnessing for the first time how the images are formed from pure noise. From looking at the images, there are early signs of hybrid design. Another great aspect of the initial results is that the resulted images were appearing quickly and were run on the CPU. By the fourth epoch we can see results and by the fifth, stability is displayed from the

network in generating images. If the network is not tuned correctly, only noise will be visible. This is a great starting set.

Cons:

One main con is that the features are barely seen. A person would have zoom in and look extremely deep for clues. Another con is the fact that these initial results are not comparable just yet to the original images, just due to the image size. Compression is playing a big role in generating these images and thus, many features are lost in the transition. However, as mentioned before, as an exploratory initial set, these results serve as a good starting point in the entire experiment.

Challenges & Adjustments needed after the preliminary set:

1. Dataset size needs to be increased for better results
2. Modification of 3D models to be rendered with less shadows & more variation
3. Improvement in hyper-tuning of GAN networks (conv layer numbers etc.)
4. Train with slower learning rate and more epochs
5. Switch between CPU to GPU computing for faster results
6. Current generated images are at 64x64 working on 128x128 which will yield finer results
7. Training GAN networks is tricky & takes a while to train

The major next step was to generate larger resolution images to see how much finer the results would be. An attempt is made to move from 64x64 → 128x128

Second set of results:

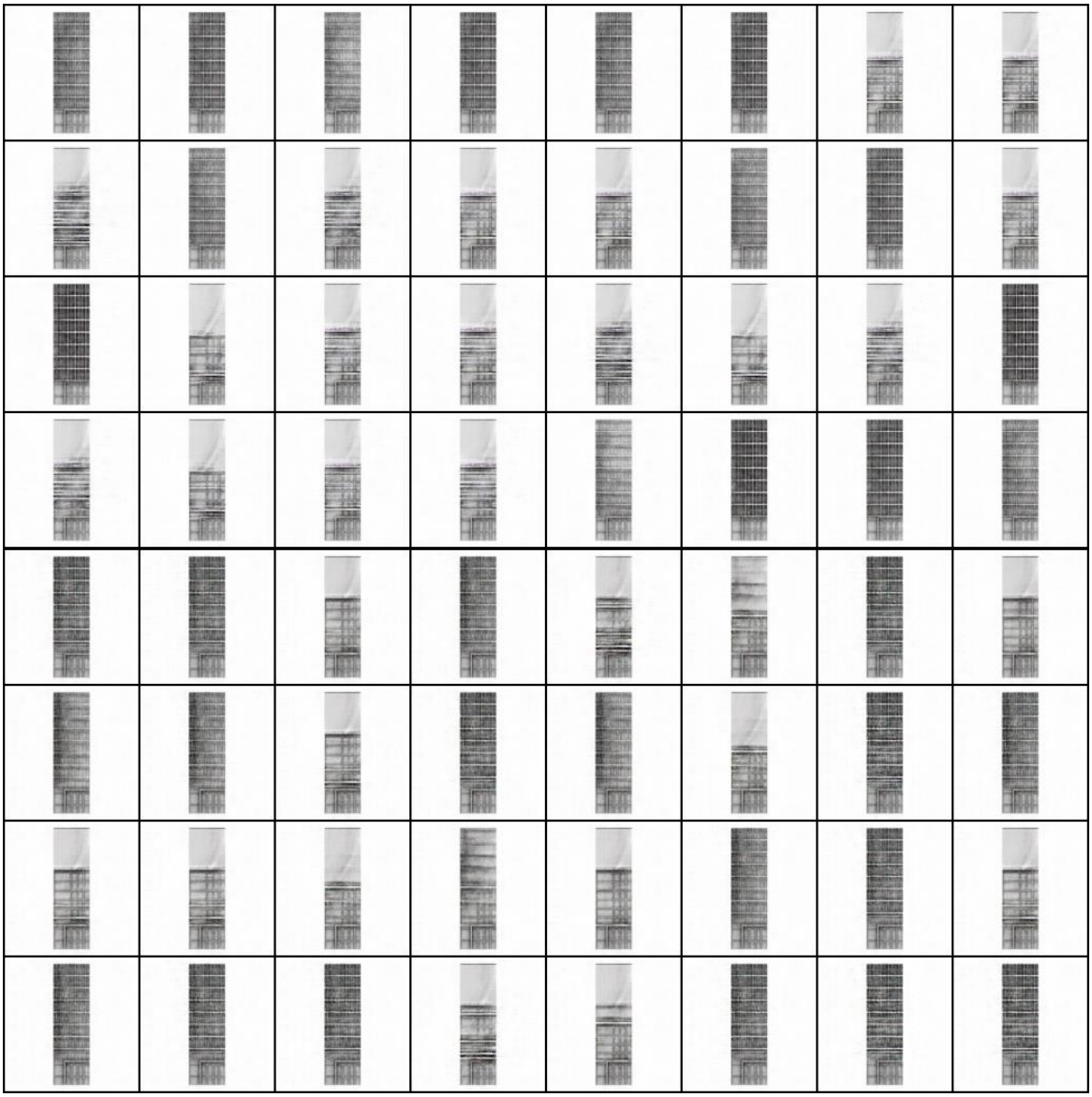


FIGURE 14. A.I. Generated images at 128x128

Here are some of the A.I. generated images at 128x128.

First time attempted to generate these images; the network would only yield noise. There was some realization that the network needed to be tuned once again, with changes in specifically:

1. Batchsize no. (initial results had a batch of 8 but this one is 32)
2. Imagesize Output
3. Learning rate

4. Adding more layers to both neural networks

5. Switching to GPU

After a series of long period trial and errors with these factors, we finally begin to see the results for the 128x128. A problem with GAN networks (experience and reading other relative research papers) is that there is difficulty when the model converges, leaving the generated images at noise level even after training the networks for more than 100 epochs. A GAN network's discriminator is constantly trying to reduce its loss, however on the other end, the generator is attempting to maximize the loss. This is the reason why training GANs is a tricky task, since the model can collapse and just produce noise. It is quite a sensitive model. The image below was towards end of a training set that been going on for a while. Towards the end, the generator network is being dominated by the discriminator who is close to reaching zero loss, thus it knew a while back that the images were fake images. This leads to a model collapse.

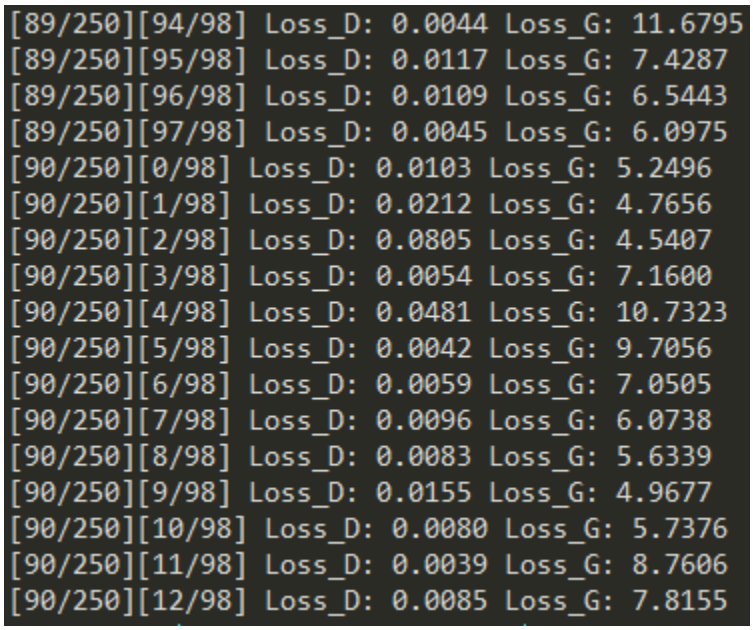


FIGURE 15. Loss function D & G

Pros:

This a much better rendition than the previous results and we can clearly see features this time. More early evidence of hybridization.

Cons:

The increase in resolution is great, however training and tuning is time consuming. As opposed to the first set of results (images appeared within a mere 4 epoch), features became visible at the 7th epoch and stabilized around the 9th epoch.

Before analyzing the current set of results, a conscious decision was made to increase the generated image resolution. The process needs repetition so we can achieve a finer illustration. The next step is to move from 128x128 → 256x256 size. The image resolution problem needed to be solved first before moving on trying out new and other features.

The results for 256x256:

FIGURE 16: A.I. generated images at 256x256

Now this is an image resolution that works, and features are large enough for anyone to witness. The images above and until recently have been a combination of blacks, greys and whites for color and reason is for simplicity's sake. Below are a few samples selected from the batches to get a closer look at their features in detail.

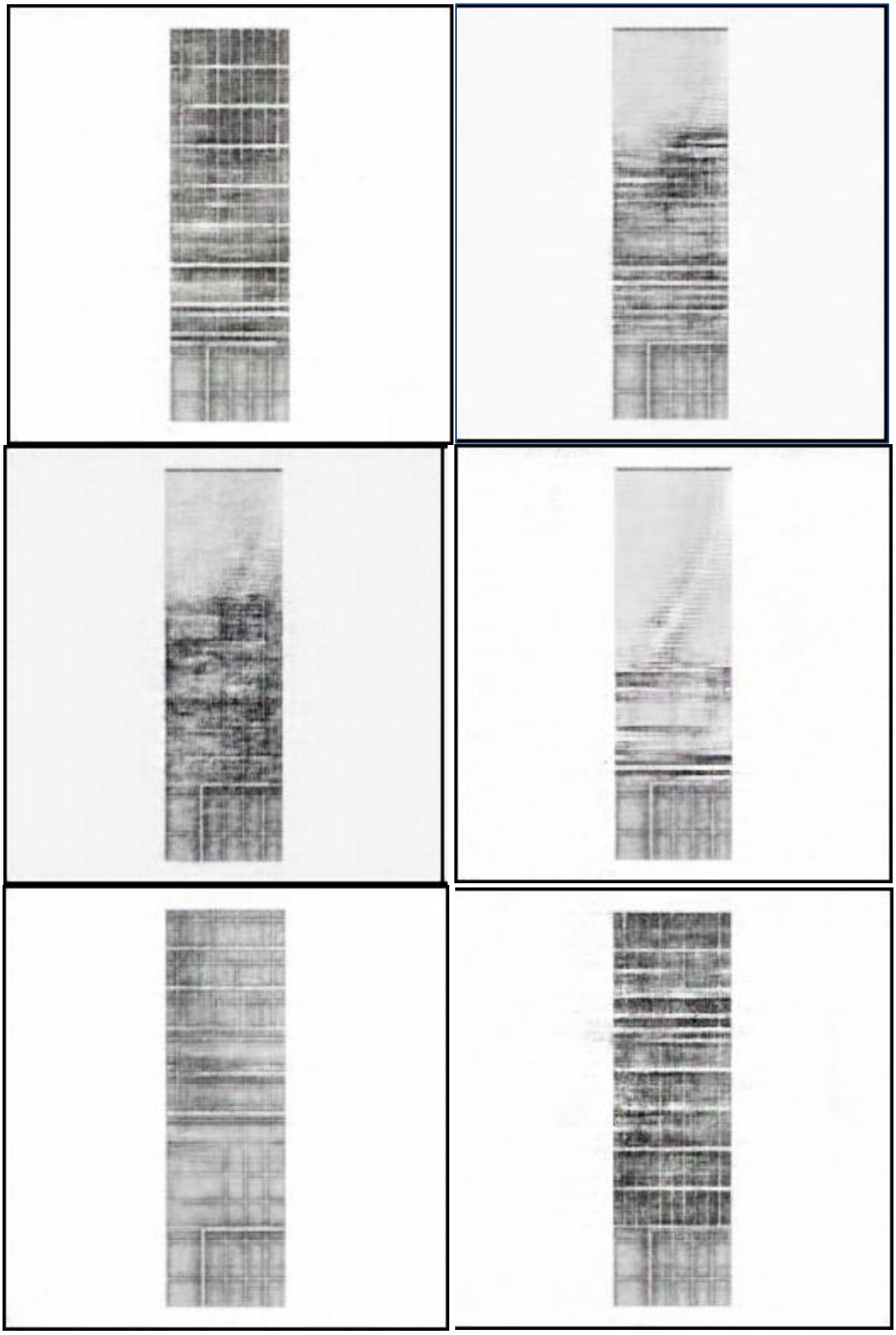


FIGURE 17. Samples selected at 256x256

In the set above, we can clearly see that the GAN network does exhibit some complexity, and all three images were from a set in the late epochs between 200 – 220. This potentially means that the network could understand the underlying structure and prominent features of each design in the late epochs, then learns to play and hybridize some of the features together. Since both the generator and discriminator are in adversarial mode and are trying to out-wit each other, then the generator would have learned enough to carry out possible hybrid variations.

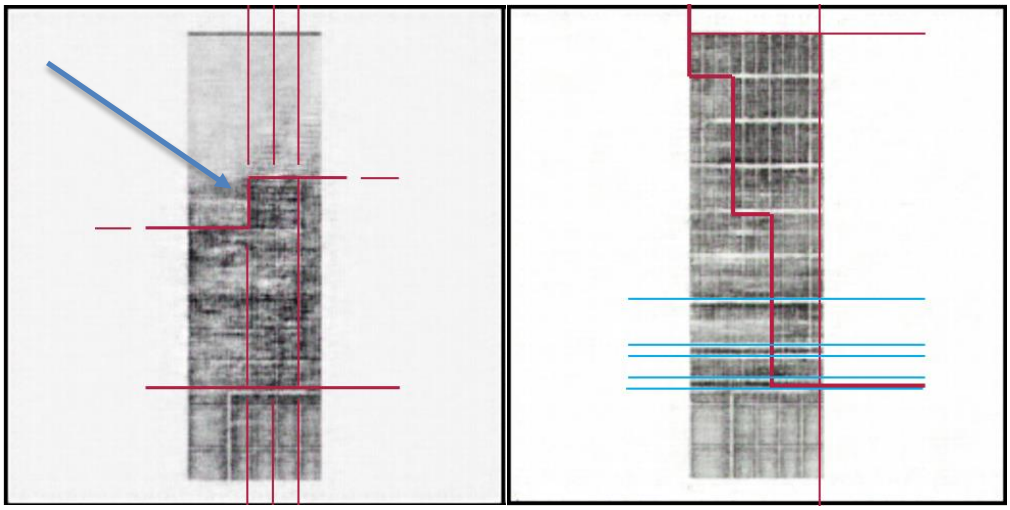


FIGURE 18. Observational analysis at set 1

Before moving on to the comparison section (compares A.I. generated images with the real dataset's images), let's look at a quick analysis made by observation at first glance. The colored lines in the images above represent edge and boundary lines. The image to the left displays a jagged red line where the blue arrow is pointing to, and that is an anomaly, a hard edge that the A.I. has chosen to stop at. This hard stop is originating from the mullions that are extruding from the base, as displayed by the red lines. There is clear evidence of understanding some of the underlying framework of the images. However, these are just observations at first sight and can be evaluated at the comparison section with the real images from the original dataset.

The image to the right is concerned with evidence of hybridization, as the GAN network has produced an image with features from both designs. The red outline in this case is the rigid design features and the blue lines represent elements (stripes) from the second design (organic).

The images below introduce color at 256x256 resolution size. The pink red color represents glass and the experiment was set up to see how GANs understands materiality (buildings), edges and boundaries (lines within patterns).

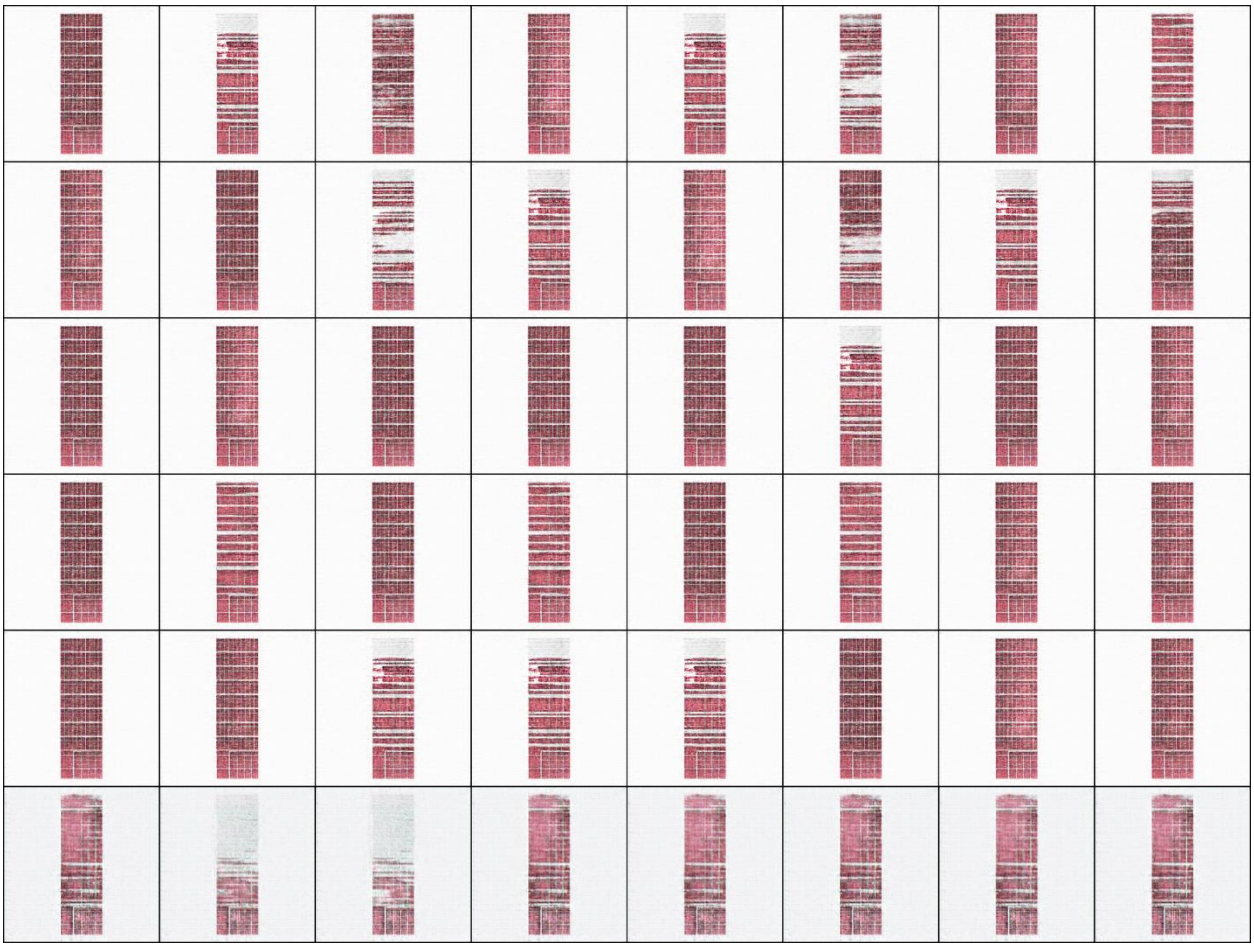


FIGURE 19. A.I generated images at 256x256 set 2

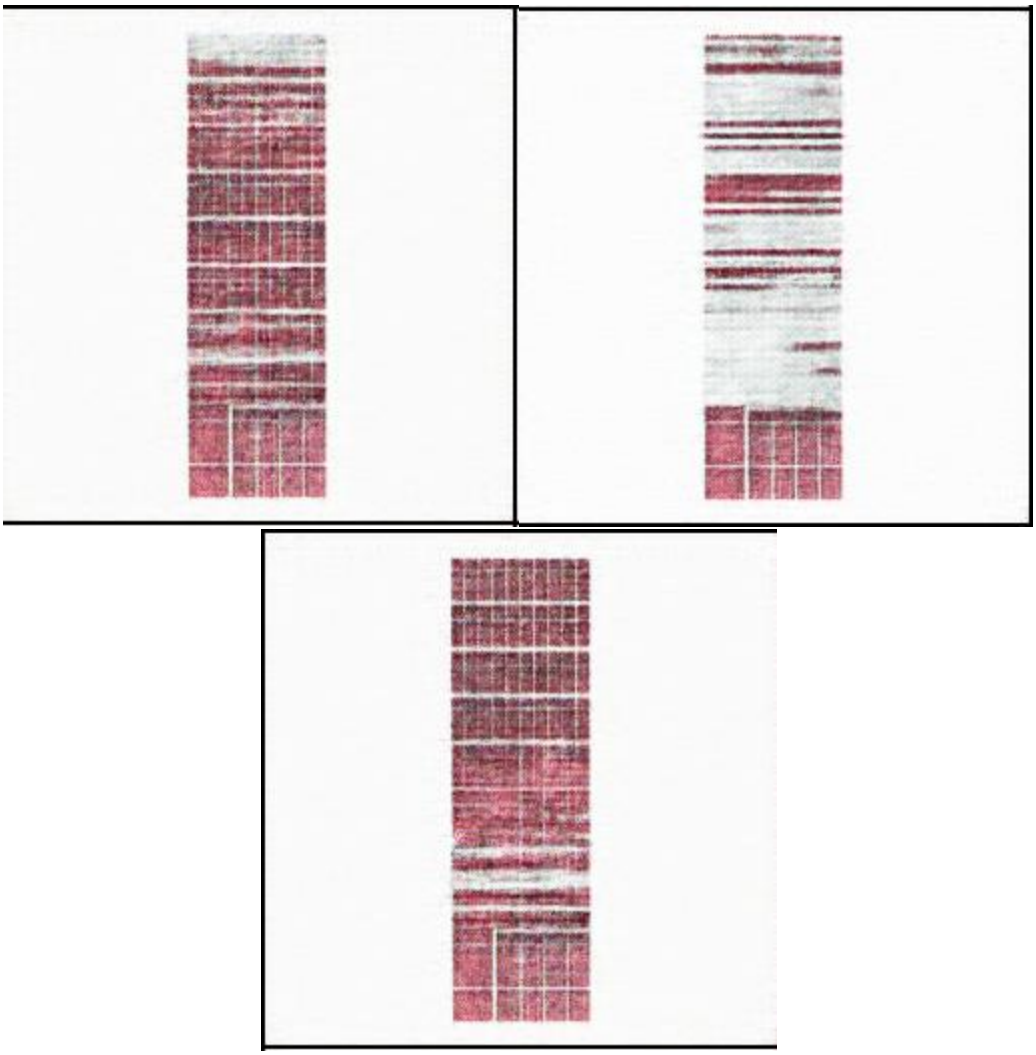


FIGURE 20. Selected samples from set 2

As opposed to the grayscale set of results, the coloration in these images allows for insight into how the network handles more tensors. Does it make sense of shadows and colors? Do the new formations still respond to the underlying framework of the images it had trained on? And does it still hybridize some of the features of both designs? Below is another set of simple observational analysis at first glance. As mentioned previously, a comparison will be made after the experiments between the preliminary observation and images from the real dataset.

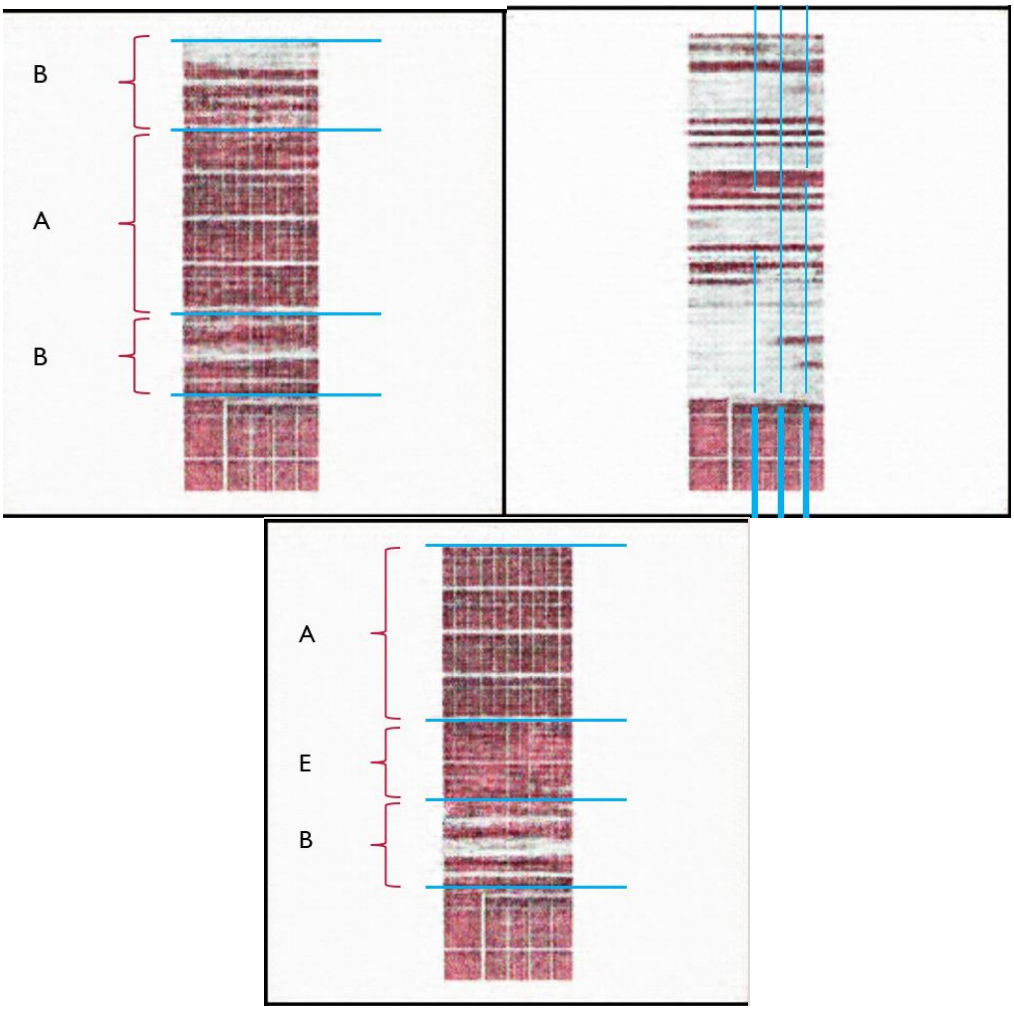


FIGURE 21. Observational analysis set 2

Here is another quick observational analysis performed on the second set. Note that all these images are pulled from the 100+ epoch sets. The first image is associated with feature hybrids, and the letters A & B correspond to which design these features are from. ‘A’ represents the rigid design whilst ‘B’ represents the organic design, thus the first design exhibits a 50/50 feature mixture. The image to the right utilizes lines from the three mullions at the base as guidelines to halt stripes crossing the whole way at intersecting points, thus creating novel patterns that have not been seen before. And finally, the third image is another feature mixture of about 50/50 from each design. Aside from these observations, there is a plethora of other designs with hybrid features and small anomalies. The main question that arises from the current

observations is can GAN networks understand the meaningful variation necessary for human interaction? In other words, are these somewhat subtle variations enough to move designers to pay attention to them and make changes to their architectural design?

The last experiment is set up to have no shadows and green color for glass. The reason for this is to see if the network understands hard edges and if it can learn and recreate some of those features in various ways. The images are scaled up to 512x512 resolution for even finer results.

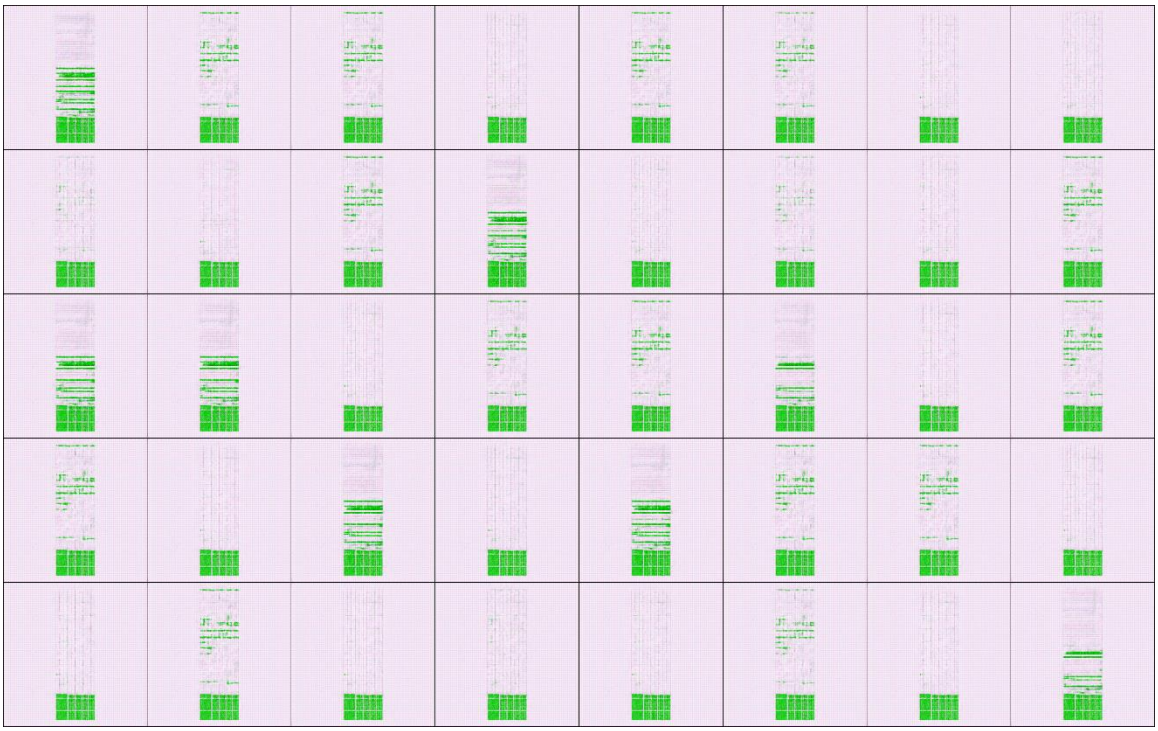


FIGURE 22. A.I. generated images at 512x512

Pros:

Many features are visible and clearly the network understands some hard edges and can recreate them. The test revealed that the network was able to generate the flat shade of green representing glass without any additional hues or other shades of green, denoting consistency.

Cons:

Even with a GPU, this takes a long time to train. The results are great, but the process is extremely time-consuming and can take a long time to fine tune the model each time. Another issue is one cannot tell if a form is emerging out of noise until a few epochs in. Each epoch takes around 30 minutes depending on batch size. The images above started to emerge overnight after 100 epochs.

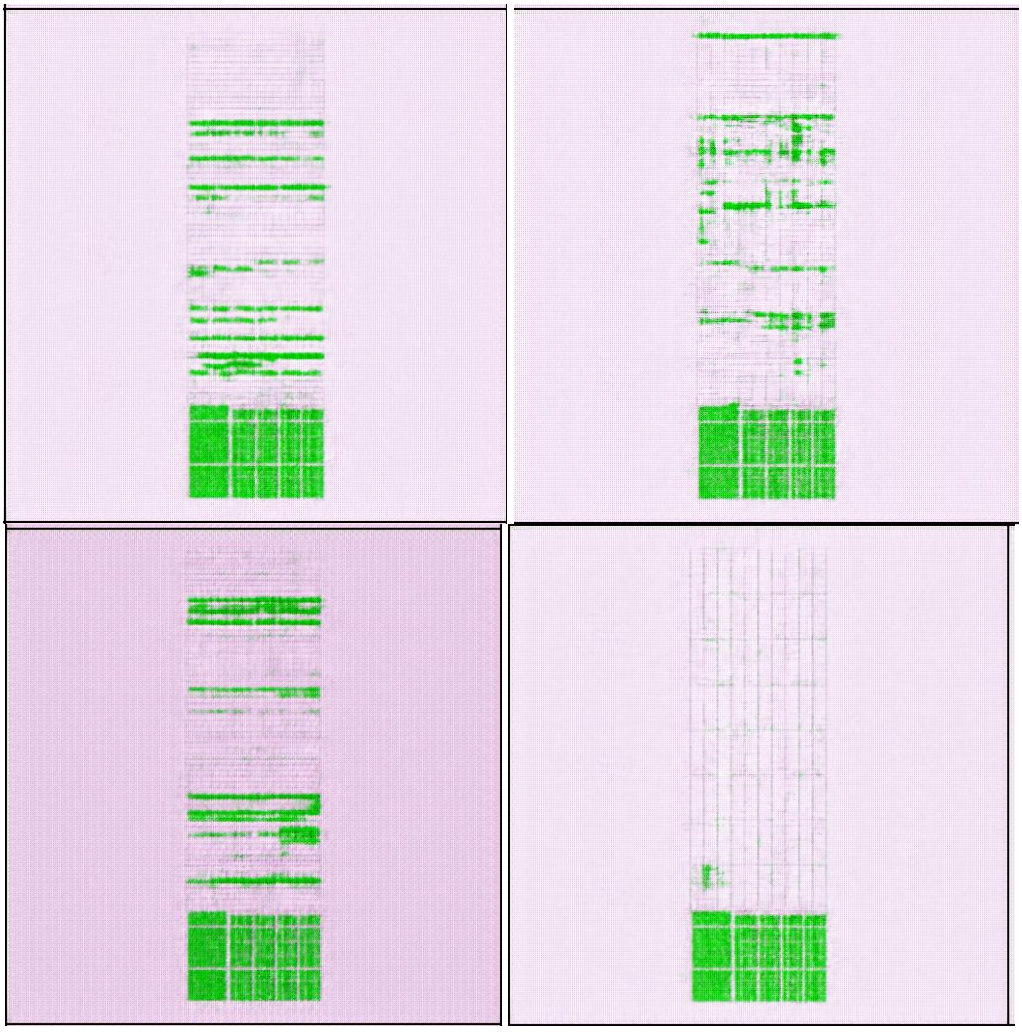


FIGURE 23. Selected samples from set 3

Looking at a few samples in detail, a few anomalies can be seen such as the stripes from design B do no cross all the way and much like previous examples, are halting at certain intersections. What slightly changed in design A is that the changing element is the window opening rather than the extrusion of frames. Hybrid features revealed when stripes are attempting to merge vertically with frame openings much like on the second image from the detail samples.

When zoomed in on the same second image, there is even an attempt made to amalgamate the grids from both designs together. Like all samples generated thus far, the static base element is well ingrained into the A.I. agent's head and has been overall quite consistent.

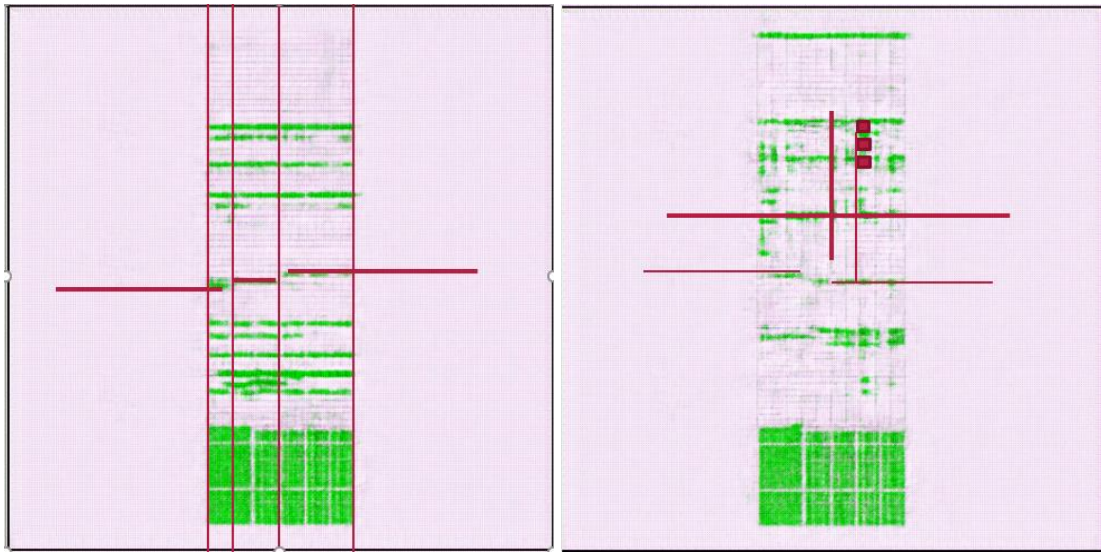


FIGURE 24. Observational analysis for set 3

These two samples taken from the batch of 4 above are analyzed to reveal that the network comprehends proportionality as well, as displayed by the red vertical lines on the image to the left (respectively 1 unit, 2 & 3 units far apart). The image to right displays feature mixture as previously discussed, where striped horizontal openings are attempted to merge with window openings from the first design, vertically. The purpose of the quick analysis was not only to display feature amalgamation, but also to point out some of the concepts talked about in general design (grid guidelines, proportionality etc.) and the GAN network is implementing these concepts. Again, the question still holds- Can GANs understand how important these variations are for human interaction? The upscaling of the image to 512x512 resolution bolsters the feature evidence presented in the A.I. generated images.

Comparisons

Since there exists a series of analysis created from observations, now it is time to pull the same samples from each set and compare them with images from the real dataset.

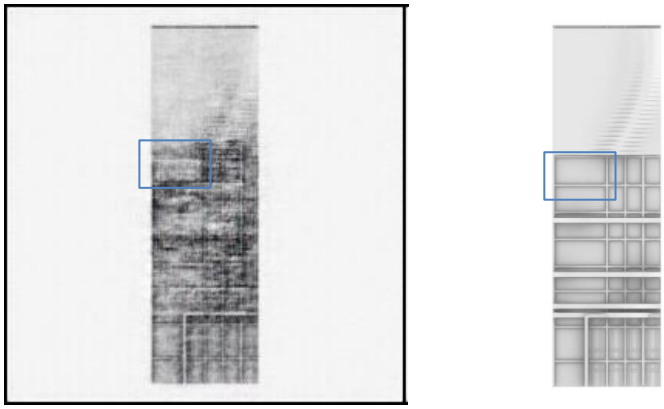
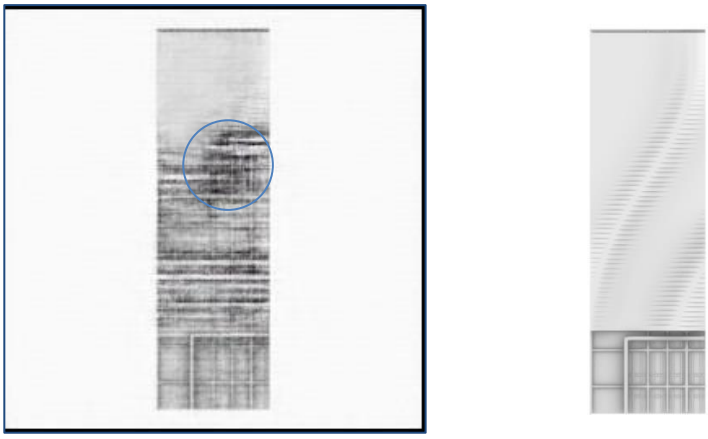


FIGURE 25. Comparison set 1 (A.I. vs real)

Note that the dataset images were at 500x500, so it is quite natural to sharper features than the 256x256 A.I. generated images (at least in the grayscale and pink window sets). In the first sample above the area marked with a blue square is an example of the network understanding a boundary created from the mullion lines and attempting to fill that area with stripes, even ending exactly at the boundary and revealing the prominently featured mullion. The reconstructions will never be as perfect as the images from the real dataset as it seems GAN's pick up on the shadow patterns as well.



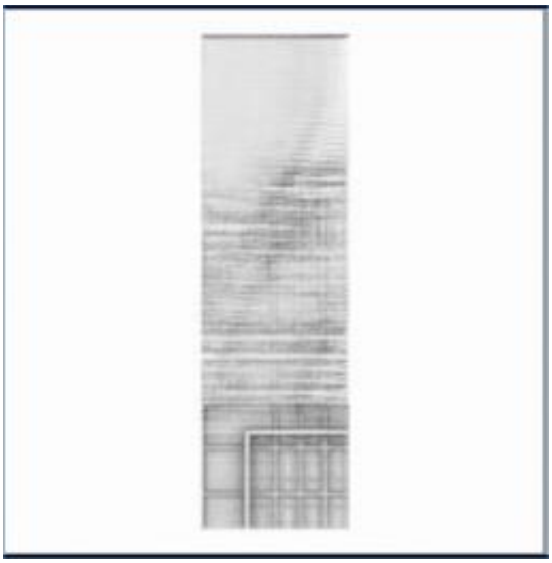


FIGURE 26. Comparison set 2 (A.I vs real)

In the set above, pay attention to the blue circled area. The A.I. had generated a reveal of the structure underneath the strips in a complex way. The comparison corroborates that the boundary lines from that reveal, flow along the organic curved lines. Just by observation, a person may not be able to see this directly without having a closer look. However, the two images (A.I. and real) have been overlaid on top of one another with the opacity reduced on the real image. This is to confirm and display where the A.I. had taken initiative from to produce some of that complexity in that circle cluster. In the first comparison, the variation may not be as high of importance for human interaction, however, the complexity revealed from the second comparison is not something an architect may simply reproduce or think of in that form. Designers can possibly learn from A.I. generated images.

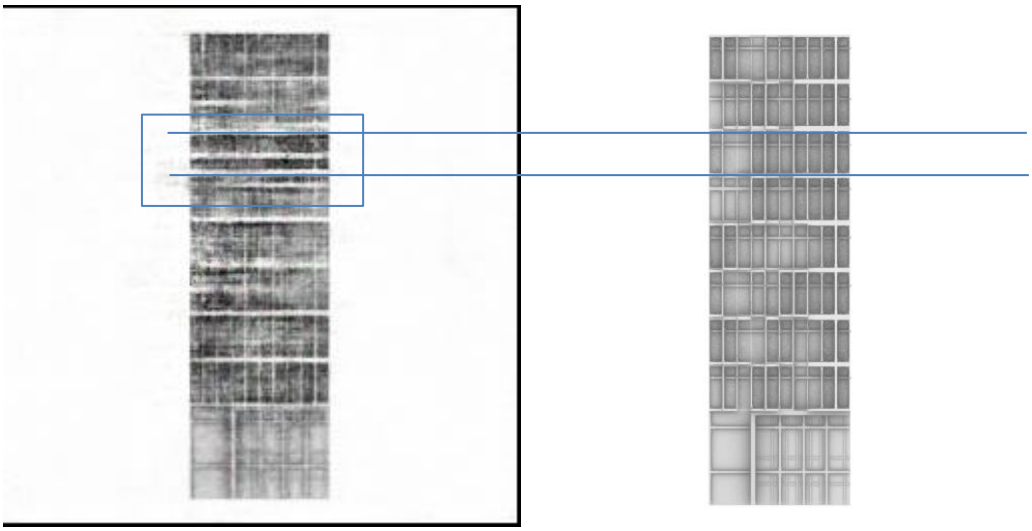


FIGURE 27. Comparison set 3 (A.I vs real)

Looking at some samples that are dominantly from the rigid design, the A.I. reconstruction highlighted a few stripes from the organic design and imposed them on top. The blue rectangle denotes where this feature mixture occurs. The blue lines show how the GAN network was playing around within the same height proportions as the frames. There is a discontinuity with the frame grid lines in that area as well.

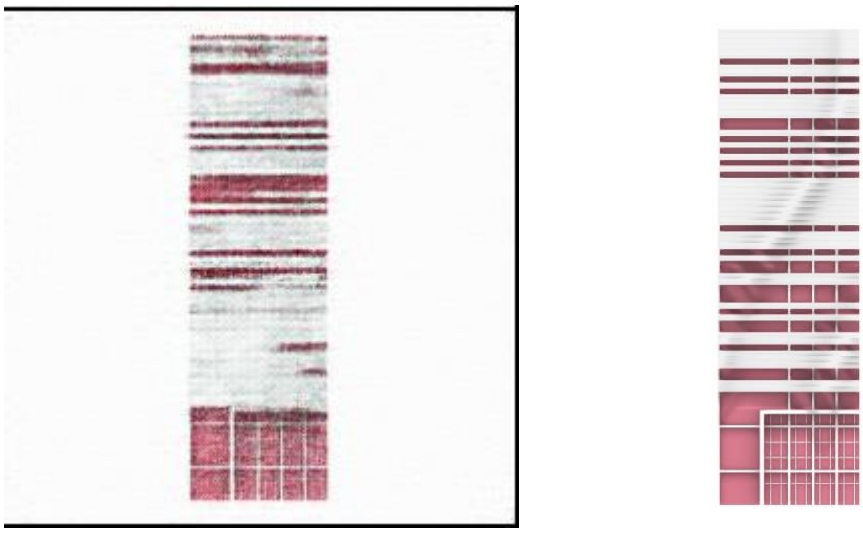


FIGURE 28. Comparison set 4 (A.I vs real)

Moving on to the second experiment's samples, the first obvious difference between the two designs is that the real sample's horizontal stripes cross all the way from left to right. However, in

the case of the A.I. generated images there are obvious stops on certain stripes. The analysis performed on this image confirms that the stripes are halting at precise intersections. Another note to mention is that mullions that go all the way up from the base are invisible or missing from the A.I. generated image.

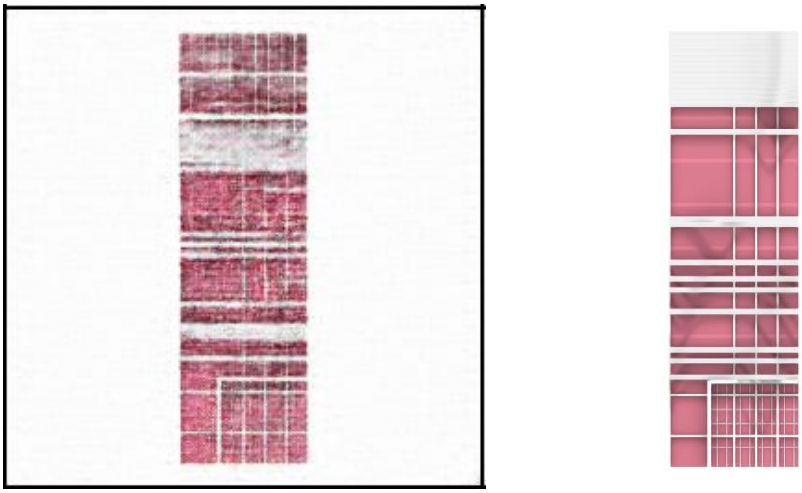


FIGURE 29. Comparison set 5 (A.I vs real)

Another clear example of hybrid features, however the quality of the A.I. generated image is not always the best. The real image (on the right), exhibits incessant features with stripes pulled all the way, along with the mullions. The image to the left (A.I. generated), decided to use rigid frame features from the first design and incorporate it at the top of the design. Another aspect of the A.I. generated image reveals two stripes that are intersecting with the mullions. This is outside of the norm, but promising since it could be understood that GAN networks are capable of using strong sets of features as guidelines.

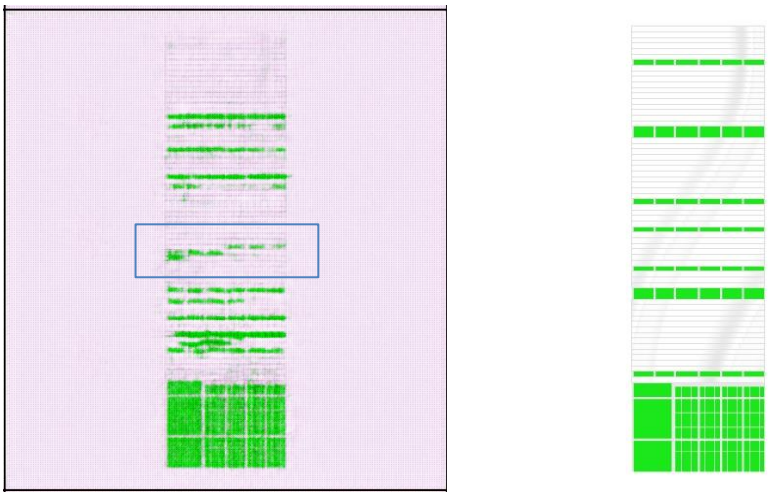


FIGURE 30. Comparison set 6 (A.I vs real)

Although this sample (A.I. generated image) is at 512x512, trumping the original 500x500 image from the dataset, however the original image is still much clearer overall. Still, there is still some credit to be given to the A.I generated image for the reconstruction. The main difference between the two images, which is highlighted with a blue rectangle, is that the network can understand unitization of elements. Aside from that, the network can also understand proportionality. The image from the dataset may only produce full stripes, just like the previous examples at 256x256, however the reconstruction comprehended some of the features from both datasets and learned to play around those features in an attempt to fool the discriminator. One feature that the GAN network seems to have trouble reproducing in most of the examples if the flow of the curve in the overall design.

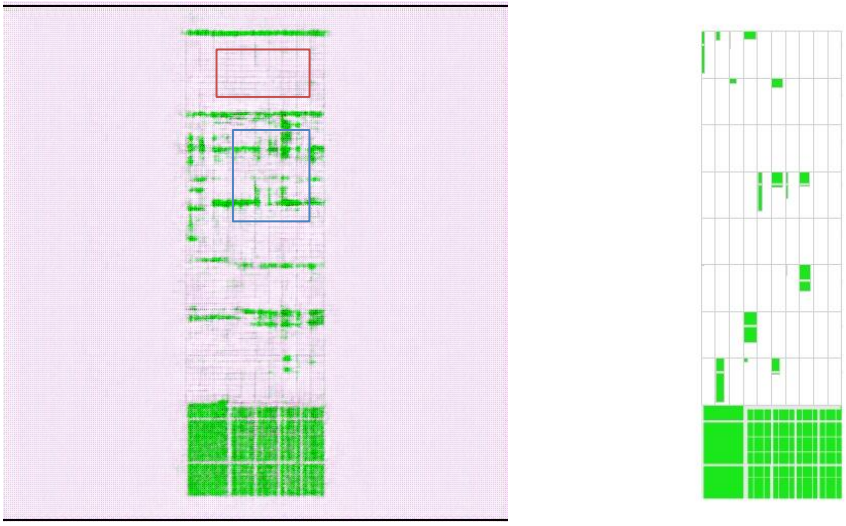


FIGURE 31. Comparison set 7 (A.I vs real)

The results for this A.I. generated image is intriguing. An attempt is made to create feature mixtures from both designs, however if zoomed in, one can see that the network even attempted to mix grid lines together as well (highlighted in red rectangle). The frame openings can also be seen in the A.I. generated image as small squares. There is a high combination of striped and frame openings, which allows the network to think of ways to not only go horizontally but also break the pattern and go vertically (because of frame openings) and an example of this is highlighted with a blue rectangle.

Improvements for the future:

1. Dataset size needs to be increased for better results
2. More tensors/features
3. Improvement in hyper-tuning of GAN networks (conv layer numbers etc.)
4. Train with various learning rates (highs and lows) and more epochs
5. Scale to 1K resolution
6. Training GAN networks is tricky since one network beats the other network quickly and can lead to a model collapse
7. Understand how to add more components to the model for better results

CHAPTER 6: CONCLUSION

GANs can understand the meaningful variation necessary for human interaction as seen from the results. There is enough to move architects and designers to able to incorporate A.I. influence into their designs. The combination of artificial intelligence, data and architects will prove to be extremely powerful and effective. The iterative search force of machines, when coupled with user-trained neural network models will provide users a way to achieve, and even explore designs that match their qualitative selections. This project will display the potential of future collaborations between a neural network and a designer working congruently.

REFERENCES

McCulloch, Warren S., and Walter Pitts. "A logical calculus of the ideas immanent in nervous activity." *The bulletin of mathematical biophysics* 5.4 (1943): 115-133.

Rittel, Horst WJ, and Melvin M. Webber. "Wicked problems." *Man-made Futures* 26.1 (1974): 272-280.

Samuel, Arthur L. "Some studies in machine learning using the game of checkers." *IBM Journal of research and development* 3.3 (1959): 210-229.

Coyne, Richard. "Wicked problems revisited." *Design studies*26.1 (2005): 5-17.

Gero, John S. "Creativity, emergence and evolution in design." *Knowledge-Based Systems* 9.7 (1996): 435-448.

Gero, John S., Sushil J. Louis, and Sourav Kundu. "Evolutionary learning of novel grammars for design improvement." *AI EDAM* 8.2 (1994): 83-94.

Rutten, David. "Galapagos: On the logic and limitations of generic solvers." *Architectural Design* 83.2 (2013): 132-135.

Hsu, Feng-hsiung. "IBM's deep blue chess grandmaster chips." *IEEE Micro* 19.2 (1999): 70-81.

Chen, Ying, JD Elenee Argentinis, and Griff Weber. "IBM Watson: how cognitive computing can be applied to big data challenges in life sciences research." *Clinical therapeutics* 38.4 (2016): 688-701.

DeLanda, Manuel. *Philosophy and simulation: the emergence of synthetic reason*. Bloomsbury Publishing, 2011.

De Landa, Manuel. "Virtual environments and the emergence of synthetic reason." *VIRTUAL FUTURES* (1993): 85.

DeLanda, Manuel. "Deleuzian social ontology and assemblage theory." *Deleuze and the Social* (2006): 250-266.

Kohonen, Teuvo. "The self-organizing map." *Proceedings of the IEEE* 78.9 (1990): 1464-1480.

PRZYBYLSKI, MAYA. "A FRAMEWORK TO ESTABLISH DATA QUALITY FOR SOFTWARE EMBEDDED DESIGN."

Silver, David, and Demis Hassabis. "AlphaGo: Mastering the ancient game of Go with Machine Learning." *Research Blog*(2016).

Fernandez, Jose Maria Font, and Tobias Mahlmann. "The Dota 2 Bot Competition." *IEEE Transactions on Games*(2018).

Sjoberg, C., C. Beorkrem, and J. Ellinger. "Emergent syntax: machine learning for curation of design solution space." *ACADIA 2017: DISCIPLINES & DISRUPTION (Proceedings of the 37th Annual Conference of the Association for Computer Aided Design in Architecture)*. 2017.