Towards Hallucinating Machines

Designing with Computational Vision

Matias del Campo
Taubman College of Architecture and Urban Planning,
University of Michigan

Alexandra Carlson
Michigan Robotics
University of Michigan

Sandra Manninger
Taubman College of Architecture and Urban Planning,
University of Michigan

Keywords: Artificial Intelligence, Design Agency, Neural Networks, Machine Learning, Machine Vision

Abstract

How many thousand images does it take an architect to learn what Gothic is, or Baroque or Modern? How many more to differentiate between good and bad architectural solutions? This article strives to de-mystify the nature of design choice in architecture by interrogating the underlying processes of Neural Networks and thus the extent of their ability to inform architectural design. The presented approach strives to explore the design problem not only through the lens of expediency, but also by considering the cultural transformation that comes along with the possibilities of a technology that profoundly asks about the nature of agency in a posthuman environment.

1 Introduction

In his famous paper *Computing Machinery and Intelligence*, Alan Turing posited the question, "Can machines do what we, as thinking entities, do?"[1]. The goal of this article is to ask this same question in the context of architectural design to gain an understanding and appreciation of an artificial intelligence's abilities to generate novel design solutions. These solutions can reach from very pragmatic plan analysis to surprisingly creative and innovative architectural solutions[2]. This article describes the motivation to explore design methodologies embedded in a posthuman and computational architecture design ecology, although the technique itself can be expanded to any area of creativity such as art, writing and music. See for example the painting *Portrait of Edmond Belamy*, by the Paris based art collective *Obvious*[3]*, Poemportraits* on Google's Artsexperiments platform[4] –a collaboration between coder Ross Goodwin and artist Es Devlin- and artists such as Holly Herndon[5], YACHT[6] and Dadabots[7] creating their music with the help of machine learning algorithms. The paper examines the meaning of agency in a world where decision making processes are defined by human/machine collaborations. The main aim of this article is to demonstrate and interrogate a design technique based on Deep Learning, a branch of the research on Artificial Intelligence.

Taking cues from the language and methods used by experts in Deep visual learning, such as Hallucinations, Dreaming, Style Transfer and Vision, this article strives to clarify the position and role of Artificial Intelligence, namely neural networks, in the discipline of Architecture. Neural Networks have become ubiquitous across disciplines due to their ability to accurately mimic human behavior and capability to perform complex tasks and model the real world. At their most core level, neural networks are purely mathematical functions that are structured and trained to extract salient features from their input. However, because neural network structure is based upon how the human brain functions, we can, in a sense, use them to **study our own sensibilities and design choices**. Thus, while they are tools that have no self-awareness or can make conscious design/aesthetic choices, we as designers can use them as tools to see the world in a different way and leverage this new sight to inform the design process.

This article's primary focus is to discuss aspects of 2D to 3D style and shape transfer techniques. The presented method can be described as a computational design approach that uses internal representations of our visual world. This condition allows a deep vision neural networks to learn from observing hundreds of thousands of images in order to invoke stylistic edits in both 2D objects (images) and 3D objects (meshes).

There are two main paths of inquiry: first, the interrogation of the technical expertise necessary to train neural networks to generate successful solutions for pragmatic problems. This can be plan optimization, structural optimization and the analysis of the consumption of material. All these problems reside primarily within realm of engineering. The second path to be explored is the wicked part of architectural design pertaining to aspects of morphological studies, Style, mood and creativity. This problem was acutely described by Margaret Boden, the grande dame of AI research, like this: *Creativity is a fundamental feature of human intelligence, and an inescapable challenge for AI. Creativity is not a special "faculty", nor a psychological property confined to a tiny elite. Rather, it is a feature of human intelligence in general. It is grounded in everyday capacities such as the association of ideas, reminding, perception, analogical thinking, searching a structured problem-space, and reflective self-criticism. It involves not only a cognitive dimension (the generation of new ideas) but also motivation and emotion and is closely linked to cultural context and personality factors. Current AI models of creativity focus primarily on the cognitive dimension. A creative idea is one which is novel, surprising, and valuable (interesting, useful, beautiful..). But "novel" has two importantly different senses*

here. The idea may be novel with respect only to the mind of the individual (or AI-system) concerned or, so far as we know, to the whole of previous history.[8]

1.1 Architectures empathy towards images and representation

Traditionally, the architecture education contains a large portion of learning through images. Architecture students browse through thousands of pictures during their education, from books, lectures, blogs and Instagram accounts, learning the nature of architecture in the process. Similar to other fields, such as diagnostics in medicine[9] or the memorizing of text in the law education[10], the architecture discipline relies on learning through the absorption of visual stimuli[11]. In a similar fashion to the beforementioned fields, Architecture is experiencing a profound change in the nature of analyzing and assessing the respective approaches for design tasks[12]. The efficacy of neural networks to emulate and surpass the performance of biological vision systems on certain tasks makes them a unique computational tool for generating novel design techniques. For example, neural network architectures can achieve higher-than-human performance on image classification tasks[13], suggesting that these algorithms extract visual representations of objects that are better for classification in comparison to the ones extracted by their human counterparts. Thus, it may be possible for a Neural Network to understand, for example, a specific architectural style such as Gothic, Baroque or Renaissance in a far more sophisticated way, and potentially with higher expertise, than a human could – provided the database of images and tags the network is shown is large enough.

1.2 It's a question of Style

How can a human differentiate a buttress from a column? Each of these objects has a specific set of visual features that can be used to identify it; these sets can be thought of as a representation of the given object. These features can be geometric and structural elements, such as shapes and forms like vertical edges or corners, or material elements, e.g., color, texture, reflections, etc.

In deep learning, the 'style' of an image is synonymous with 'texture', which describes the underlying 2D patterns that result from how the shape and material of objects in the given scene interacts with the light from the environment and camera to form the image.

In the same way that an architecture student must observe instances of different architecture styles to learn which elements are unique to specific architectures, a neural network must be trained by observing millions of images to learn which features, both stylistic and geometric, of an image are relevant to the task it is trying to solve. However, given the nature of how neural networks learn (which is discussed in detail in the following section), we cannot easily determine which visual features form the representation that a neural network learns for a specific object. We can use these learned feature sets from networks to achieve novel representations of form for standard architectural motifs. For example, if given the task of image or object classification, what is the set of features learned for 'arches' by a neural network? (middle panel, fig.1) Are there specific patterns to arches that the human visual system disregards when it performs classification that a neural network utilizes? The learned visual representations of neural networks could capture new ways of 'seeing' and understanding building structure and aesthetics. However, the discussion as of how neural networks would classify and understand aesthetics is beyond the scope of this article, and is part of a larger conversation to be addressed another time. The method presented in this article, uses the learned representations of Convolutional Neural Networks (CNNs) to invoke edits to both 2D objects (images) and 3D objects (polymeshes) in an effort to transform and potentially remove the constraints of the human visual system on the design process.

Figure 1: In this example, a neural network is trained to dream or 'hallucinate' the features of arches on a given depth rendering to impose novel geometric structure into the input, and then to perform a style transfer between a rendered texture image from a student's 3D model upon the result of the dreaming process. Image:

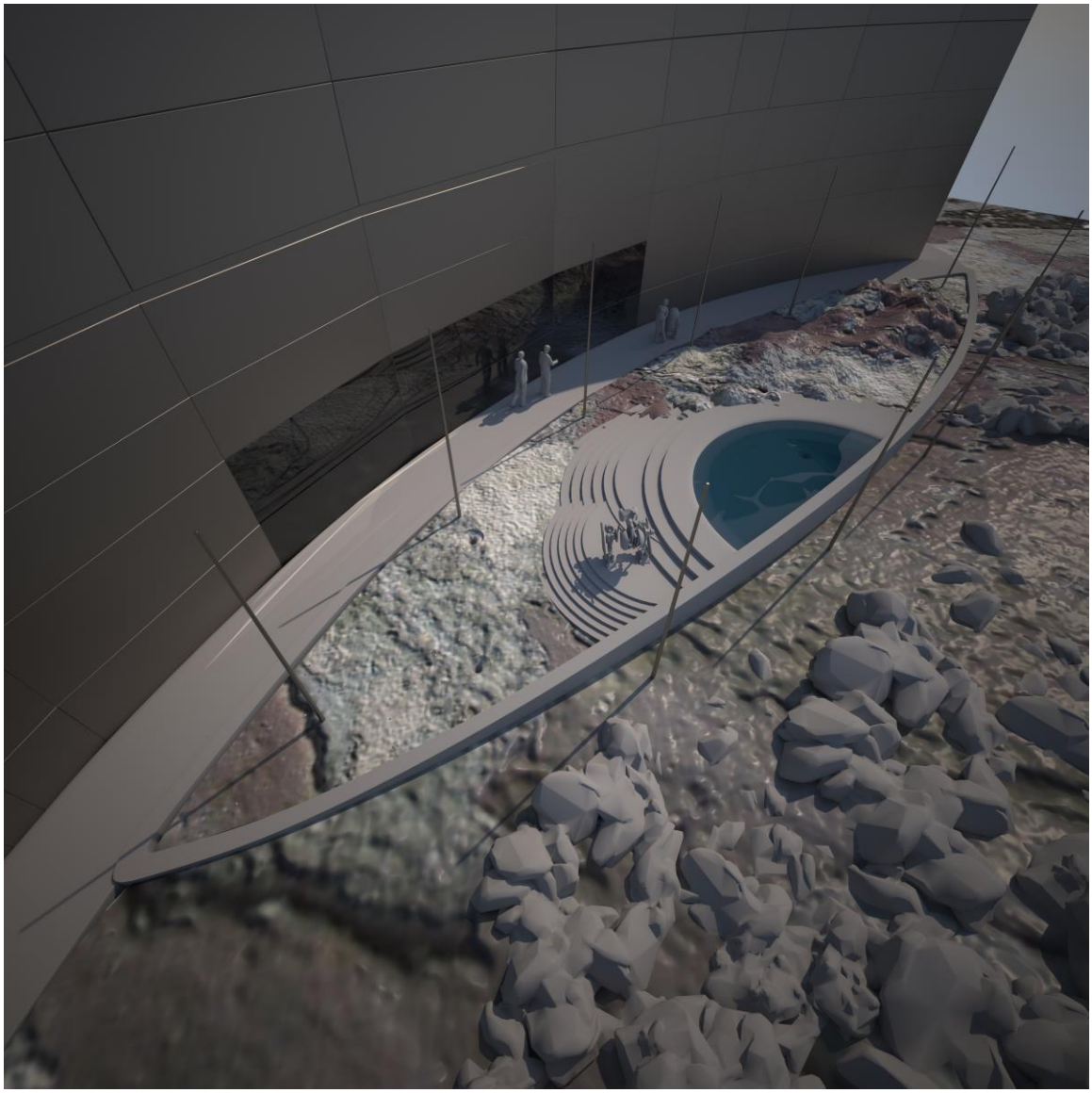

Figure 1**: Robot Garden**

The proof of concept project - presented for the first time in this article- is the Robot Garden (Fig.2) currently under construction. The Robot Garden is designed as a future test ground for bipedal and quadrupedal robots. Part of the challenge consisted in designing various ground conditions and complexities that robots have to overcome. In order to achieve this, and to include a Neural Network component in the design process we designed a methodology that would involve satellite images of the site and a particular training set to dream features on this image. These features consisted of architectural

elements derived from a sprawling dataset trained to recognize architectural features such as arches, architraves, columns, mullions, moldings, rustications and many more. This allows to generate a synthetic ecology present somewhere between the natural and the artificial. The resulting digital 3D model serves on the one side as the template for the construction, on the other it serves the purpose of simulating and preparing the test of robots in the garden. The garden itself is executed in natural material providing various different terrains such as grass, gravel, stone, sand, water and has topographical features such as waves, inclinations, pits etc. to emulate difficult terrain. Though the garden is made of natural materials, there is a contrast between the natural landscape features adjacent to the Robot Garden, and the Robot Garden itself which flirts with an intentional artificiality – operating within the realm of a synthetic ecology, befitting the origin of robots as an artificial progeny. In the following the authors would like to explain the background of this approach, in particular explaining the computational methods used in Neural Networks.

2 The Nature of Neural Networks

In this section we provide the foundational/background information necessary to understand what a neural network is and how it operates on visual information; it is based upon the material presented in ref. 19. A neural network is a neurobiologically-inspired computing system comprised of groups, called layers, of simple, highly interconnected processing elements, called neurons. Input information flows through a neural network in a feed-forward, hierarchical manner: each node in a neural network receives input from neurons in the preceding layer, but not neurons within its own layer. It transforms this input into a new representation, and then passes it to the neurons it is connected to in the proceeding layer.

Mathematically, this transformation is defined as follows: a neuron calculates a weighted sum of its inputs and applies a threshold onto this value via a nonlinear function to determine if it has been ‘activated’ given its input. This output, i.e. the neuron’s state of activation, is then sent to neurons it is connected to in the following layer. The weights

used to calculate the neuron state, paired with the thresholding operation, effectively filter out specific information from being sent to the next layer, allowing for different features in the input to be extracted by each neuron. Since each layer operates on the activations of the previous layer, neurons can also extract/detect groups of input features. This weighting-then-thresholding transformation allows for pattern recognition; it allows a given neuron to select specific elements and information to be transmitted to its downstream neighbors while suppressing others, ultimately refining the way in which the image is 'seen' or represented in the network.

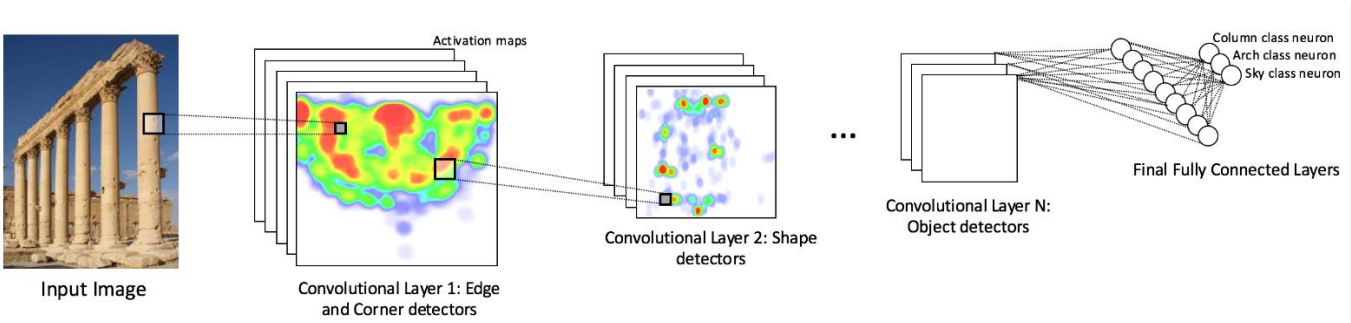

Figure 3. A pictorial example of a CNN architecture. Note that for convolutional layer 1, we show an activation map for a single kernel (gray square), and similarly one for the second convolutional layer. We also show the window within the layer's input that the kernel is operating on (white square). The activation maps can be thought of as heat maps, where a high value indicates the possible presence of a feature the kernel is trained to pick out, such as vertical edges. Each convolutional layer can have multiple kernels, which means multiple activation maps. Each kernel in the proceeding layer operates on each of the input activation maps to detect co-occurring simple features, so they can activate to complex shapes. The ellipsis indicates additional convolutional layers within the network that are not shown. The final layers of the network are dependent upon the task; in the above network, the task is object classification, and thus uses fully connected layers to transform the 2D activation map features into a vector of predicted class probabilities.

The way in which neurons are connected between layers controls how input information is transformed through the network. This connectivity structure varies based upon the application. For example, a fully connected neural network means that all of the neurons in one layer are connected to all the neurons in the proceeding layer and is designed to

process 1D input information into 1D output. The most widely used neural network structure visual information processing and tasks (including facial recognition, pedestrian detection, and image generation) is the convolutional neural network (CNN). An example of a CNN is given in Figure 3.

A CNN is designed to operate on the image; the compressed, 2D pixel representation of our 3D world. The neuron in a CNN processes the 2D information about a location/pixel (x,y) in its input image. It calculates its activation by applying a 2D convolutional kernel to the pixel information at its spatial location, and thresholds the value with a nonlinear function. The convolution kernel essentially performs pattern recognition on the neighborhood of pixels around (x,y); it weights the groups of pixels that represent salient visual information, like edges (think about the examples of columns and arches mentioned before). This convolution kernel is slid over the entire 2D input, like a sliding window, to create a 2D activation map. The value of an element within the activation map indicates the presence and magnitude of a particular visual pattern, which is a 'feature', at a specific location in the input. Depending on the values of the weights of the convolution kernel, different 2D spatial patterns of pixels can be detected and thus extracted by the CNN. Each layer can have multiple convolutional kernels, so the output of a single CNN layer is a 3D activation volume of stacked activation maps, each representing the locations where a specific visual feature is present in its input. Conceptually, a CNN layer transforms its input by decomposing it into the set of spatially-varying visual features captured by the layer's convolutional kernels.

Increasingly complex spatial patterns/features can be detected and extracted by each proceeding layer. For example, a convolutional kernel A in the first layer of a CNN may have weight values that cause it to detect the presence of patterns that resemble diagonal lines within the input image, and a second convolutional kernel B in the same layer may detect groups of pixels that look like horizontal lines. A convolutional kernel C in layer 2 operates on the activation maps that are output from convolutional kernel A and kernel B from the first layer, so it could activate in the presence of patterns that resemble line intersections, which would require the activation of both convolutional kernel A and convolutional kernel B in the same location in the activation maps from the first convolutional layer. As a result, at the final layers of the network, neurons will have weights that activate to the semantic content of the image (as opposed to raw pixel values), e.g., complex structures such as buildings. Fundamentally, this is the same

process any architect would go through when evaluating and assessing a building's style; the architect would break down the building into its basic components, looking for features such as columns, arches, curves, colors, materials, that are associated with specific time periods, schools of thought, or that can differentiate the image content in a unique and meaningful way. An example of this is given in Figure 4.

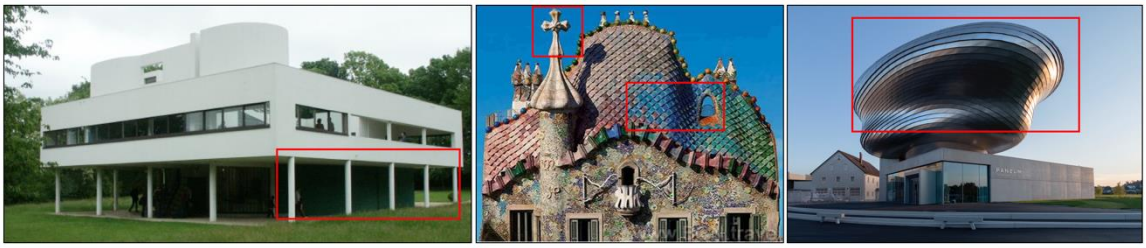

Figure 4 Consider the visual task of building recognition/classification with the above images. At first glance, the panel on the far left of Le Corbusier's *Villa Savoy* is easily separable from the two on the right; it differs significantly from the other two images in terms of shape (of the three it is the only building with columns and simple geometric edges/corners), in terms of color, and in terms of background. The right-hand panels, the *Casa Batllo* by Antoni Gaudi (middle image) and the Coop Himmelb(l)au's *Haus des Brotes* (far right image), have similar bluish backgrounds and have curved roofs. Therefore, more complex visual features, like the texture and color of the *Casa Batllo* roof must be used to differentiate the two. If presented this small dataset from which to learn building classification, a neural network would only need to learn the features 'white' and 'columns' to separate the Villa Savoy from the others. In contrast, the network would need to learn more complex, dense visual feature sets to separate the *Casa Batllo* image from the Coop Himmelb(l)au image, e.g., the different roof shapes as spatial locations of curves in the image, the different colors in the roofs, and perhaps the locations of the roof ornaments or windows in the *Casa Batllo* image. These example features are highlighted in red in the figure.

2.1 Learning Architectural Features

The weights of each convolutional kernel, and thus which visual features are extracted from the input, need to be learned from data. A widely-used learning paradigm is a training procedure called *supervised learning*. In this training scheme, a large number of different (image, label) pairs are presented repeatedly to the network with the goal of teaching the network to perform a specific visual task. The label varies depending upon the desired task. For example, in image classification, an image would be paired with a

label that describes elements of its semantic content, such as *'arch', 'fountain'* or *'column'.*

For each image input to the CNN, the network uses its current set of weights to process relevant visual information, which is transformed at the final layer into a prediction of the input image label. For example, in the image classification task, a neuron in the output layer would activate/detect if all the visual features associated with a particular class are present in the input image. In a winner-takes-all fashion, the neuron in the output layer with the highest activation would be considered as the class prediction of the network.

An error (also referred to as loss) is then calculated on this predicted label based on how close it is to the ground truth label of the input image. Multivariate differentiation is used to calculate the gradient of this error with respect to the network's weights. Conceptually, the gradient of the error represents how much each weight parameter contributed to the network's prediction, and thus its contribution to the prediction error. The values of the weights are then changed based upon the magnitude of their gradient, with the objective that on the next iteration, the updated weight values will yield a more accurate prediction. This process is referred to as backpropagation, because the error is literally being passed backwards through the network layers to learn better visual features.

Revisiting the metaphor of the architecture student, in order to differentiate and identify the nuances between classes of architecture styles, a student must utilize a similar training procedure. They must decompose many input examples/images into potential candidate features, and use these to make predictions on these images based on his or her past knowledge of architectural features and their associated classification. The student must then evaluate their prediction, and if an error is made, update their internal mental model of the given class with either additional, novel visual features, or by removing current visual features that do not actually help differentiate that class from any other class.

The learning process for neural networks is conceptually the same: the network takes the inputs (images) and the desired outputs (labels) and updates its internal state/mental model of the world (i.e., its weights) according to the error in the network's predicted label such that this prediction becomes as close as possible to the desired label. The task or error calculation is akin to 'schools of thought' or architecture styles; it is the lens and

bias through which the network learns about how to process the subset of the 'world' that is captured by the training images.

This training/learning procedure is performed for many iterations, repeatedly feeding in randomly sampled images from the dataset and adjusting the weights until the network achieves the highest possible prediction accuracy. The labels thus act as a supervisory signal that guides the network to process, extract and transform visual information into a representation that maximizes the network performance on the desired task. For example, in the instance of image classification, the network transforms and decomposes images into features that are different/distinct between classes, making grouping images into a class easy. The class *'column'* may have features associated with vertical lines and/or ones that capture standard column material, like marble texture. Examples of learned feature sets for *'fountain'* are shown in Fig.5. It is important to note that a neural network is learning how to represent objects from scratch, i.e., from the raw pixels in the given input images. Its only bias is to maximize its performance on the desired task. It is not influenced by any information that exists outside its training images or labels.

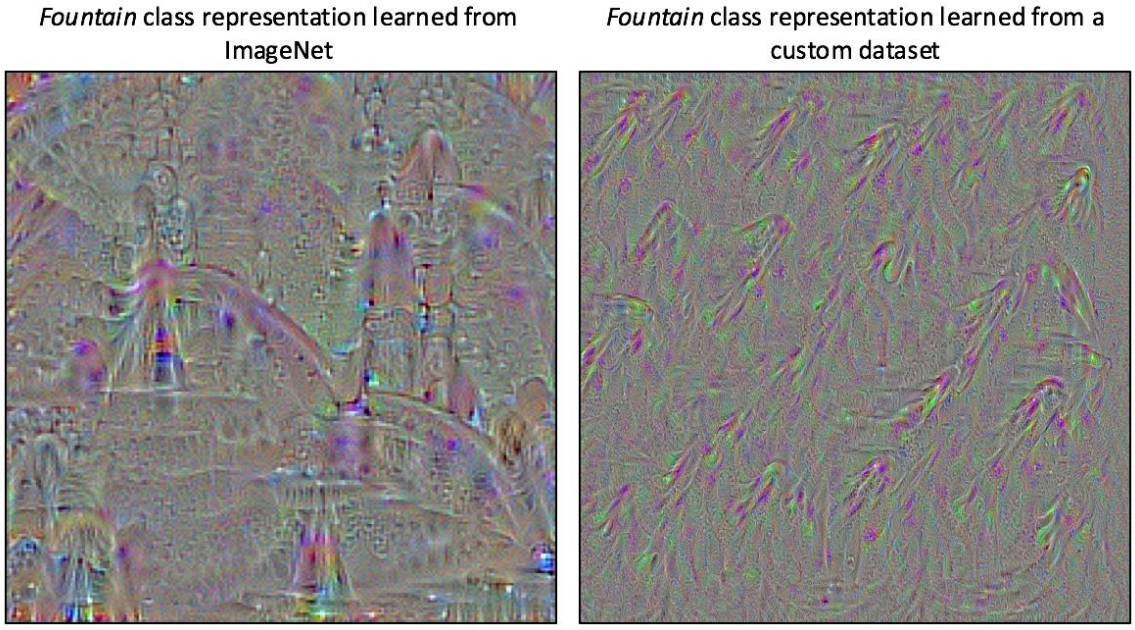

Figure 5: Applying a visualization technique similar to google deep dream, which is discussed in detail in the methods and supplement, we can visualize the feature set learned for a particular class by a CNN that has been trained to perform image classification. The above left-hand image is the feature representation for the 'fountain class' generated from a CNN trained upon the ImageNet dataset, which contains over a

million images and is labeled for over 1600 classes. It is the dataset that 'best approximates' the real world due to its sheer size. The right-hand image also shows the fountain class representation learned by the same CNN, but trained on a much smaller, custom dataset with only 8 classes, each representing an architectural element, specifically, stepping stone, fountain, ditch, brick, stair, arch, pedestal, and boulder. We see that the primary feature learned for both of the fountain class representations appears to be based on water, which suggests that either (1) both datasets had few to no examples of dry fountains, and/or (2) that the other classes captured in each dataset did not have water features, so this was the primary differentiating visual pattern between 'fountain' and the other classes, and thus the only pattern the CNN needed to learn to successfully identify it. However, note that in the ImageNet Fountain representation, there is more spatial and geometric information than just a 'vertical spout' that the CNN has learned is necessary to separate a 'fountain' image from other classes. Compare this to the fountain representation for the custom architectural elements dataset, in which only a water-like feature is necessary to differentiate fountain from the other 7 classes in the dataset.

2.2 Fountains, Figures and Features – or how to confuse an AI

As previously mentioned, outside of the error function, there are no constraints upon what visual features the network actually learns from the data. There is no way to know beforehand what the weights end up being at the end of the training procedure, and these values can change depending on a variety of factors, including the how the weights are initialized at the start of training, the order in which the images are shown to the network, how many training iterations are completed, how many layers are in the network, how many neurons per layer, how many convolutional kernels per layer, the amount of image data and variability of images. The salient feature information captured by the weights is also hugely dependent upon the visual information that is contained within the training data, as seen in Figures 4 and 5. The training dataset defines the CNN's notion of the 'world', and encounter difficulties when given information that falls outside of it.

In this sense, the 'design intelligence' of a neural network is determined entirely by the size and variety captured its training dataset. For example, if the CNN from Figure 5 was only shown images of tiered, marble fountains with vertical water spouts to learn the 'fountain' class, it may not be able to accurately identify dry fountains, metal fountains, or wall fountains. This is because it has associated the vertical 'spout', 'marble feature'

and/or ‘water feature’ as being necessary for a fountain to be within the image. As can be seen in the figure, there is more spatial information than just a ‘vertical spout’ that the CNN has learned as features for the fountain class.

The visual features learned by the network are also dependent upon the chosen task. For example, in the instance of image classification, visual features for a given class would be learned that differentiate it from other chosen classes, which can be seen upon comparing the two representations presented in Figure 5.

Ultimately, while it is understood how a single neuron processes information, given the large number of parameters and nonlinearities that comprise a single network as well as its dependency upon the training data, it is difficult to tease apart how the neurons collectively function upon images to achieve a prediction. This results in the ‘black box’ nature of neural networks; we cannot guarantee what features are learned or how its learned. Consider the representations shown in Figure 5; while some of the features learned by the networks align with our own feature set, e.g., spouts, others are unintelligible to us, but still yield high classification accuracy. This is in direct contrast, for example, to the generalizability that a human architect who can rely on his/her education, and the many images ingested throughout his/her career as well as an inherent sensibility towards design. If and how sensibility, for example, can be part of this process is yet to be seen. This means that, in the specification/definition of both task and dataset, practitioners of neural networks can guide what and how the neural network algorithm processes information about the world.

3 Methods - or: What it means to be a pixel

Visual tasks can be extended to 3D by redefining what it means to be a pixel. Naively, a third dimension can be added to an image pixel to create a voxel to represent a discretized 3D space. However, the additional spatial dimension creates a huge computational burden as a result of the increased input size and intermediate network representation sizes, which ultimately renders 3D vision/perception infeasible for many vision problems. This explains also the low resolution in some of the images presented in this article. Due to its GPU hungry nature, images must be kept to a low resolution in order to be able to process the thousands of images necessary to train a Neural Network. Other commonly used representations for modeling objects in the 3D world are depth maps, point clouds

and polygon meshes (see Fig.6). The choice of input data representation (specifically, the information it contains) will directly impact the features learned by the neural network. There are pros and cons of each type of data format in this regard. For example, while point clouds are much sparser and thus take up much less computer memory than voxels, they do not capture a sense of a 'face' or 'surface', and therefore editing techniques involving lighting or texture manipulation, such as style transfer, cannot be easily applied to this input[14]. In addition, it makes the materialization of these results more complex as it cannot be 3D printed or rationalized into components for fabrication. A watertight mesh or Nurbs surface is necessary to continue processing the file for digital fabrication.

While the majority of neural networks have been developed to process 2D visual information in the form of images, their structure can be extended to operate upon these different 3D representations of the world. These deep learning approaches fall into two general categories or algorithms: (1) project 3D objects into 2D representations, such as depth images, and then have 2D neural networks, i.e., CNNS, operate on this 2D representation of 3D data, or (2) use neural networks whose connectivity structure has been modified directly to operate on 3D information.

Figure 2: One of the first attempts of a 2D to 3D Style transfer on the Robot Garden Project. In order to create the high resolution model, the result from the style transfer was transformed into a depth map shader for the polygon model. A noisy, albeit interesting, result.

Applying the techniques of the first group of algorithms, 2D neural network-based image editing methods/techniques can be used to make aesthetic changes to 3D objects based upon the learned representations of neural networks trained on 2D image data. Two of these image editing methods, which are used in this paper, are 2D deep dreaming, a method popularized by Google, and neural style transfer[15]. Deep dreaming is an image editing algorithm that was originally developed as a tool to visualize the image features learned by a neuron, a layer of neurons, or the features that are associated with a particular class (e.g., dogs). Neural style transfer is an image editing technique whose objective is to alter a given input image so that it captures the style of a second, 'style guide' image without altering the original image's semantic and geometric content. It achieves this by leveraging the model of style and shape that the CNN has learned from its training dataset. For a more detailed description of each method, please refer to our supplement.

As described above, point clouds and voxels have no sense of connected structure or smooth surfaces. These factors however are necessary in order to be able to create continuously closed polygon meshes. These are crucial when it comes to elaborate an architectural project, be it for representational purposes (animation, Rendering, plan etc.) or construction purposes (creating fabrication files, rationalization of components and parts etc.). Object meshes are a happy medium between point clouds and voxels: they are computationally more efficient than voxels but have enough visual information about surfaces that can be manipulated with interesting aesthetic edits.

Currently, the Neural 3D Mesh Renderer[16] is the only method that allows for easy 3D object mesh editing based upon 2D neural style transfer and dreaming. It is a differentiable rendering/rasterizer algorithm that can be used as an input layer with CNNs and other forms of 2D neural networks. This framework effectively yields a mapping function (the fusion of the neural renderer algorithm and neural network) between the 3D polygon mesh and 2D image representation of objects. We can use this mapping function to associate any change within the pixels of an image to a corresponding change in the vertices and surfaces of the input mesh. Thus, the same principles/formulations guiding the aforementioned 2D dreaming and 2D style transfer techniques on images can be used to perform image editing on the surfaces of 3D objects. Please refer to our supplement for more details of this process.

4- Transferring 2D style onto the 3D nature of the Robot Garden

The project ***Robot Garden (Fig.8)***, makes extensive use of the described technique. The provided site was analyzed using a set of satellite images as a basis. The given shape of the site was cut out of the satellite images to create a set of pictures that are used to transfer the 2D style of the satellite representation of the site onto a 3D model of the robot garden. In an attempt to have a Neural Network dream or hallucinate architectural features on the site, it was trained using an extensive library of images of features such as boulders, stairs, stepping-stones etc. Surprisingly the resulting images represent a novel view to these archaic architectural features. The hybrid nature of the resulting meshes do not show the features in full clarity but are rather the hallucinogenic dream of a machine trying to see these features in the landscape.

FIGURE 3: THE LENS-SHAPED SITE OF THE ROBOT GARDEN IN A CURRENT SATELLITE IMAGE. VARIOUS SATELLITE IMAGES, OF DIFFERENT AGE, WERE USED IN THE DESIGN PROCESS AS BASIS OF A 2D TO 3D STYLE TRANSFER.

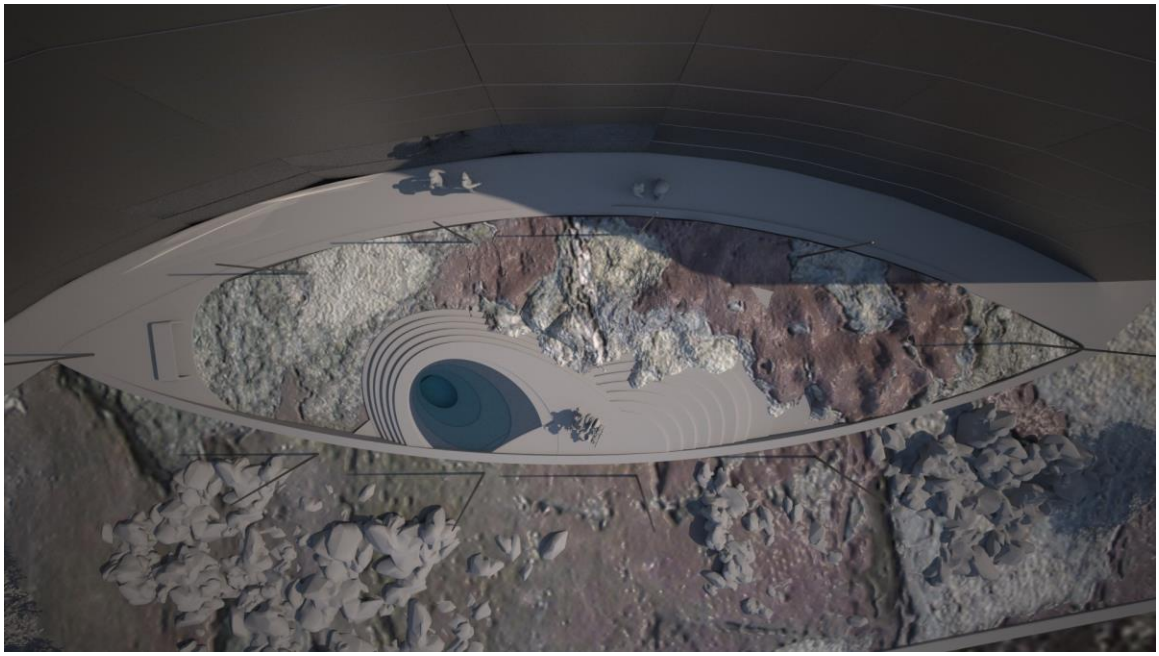

FIGURE 4: CURRENT TOP VIEW OF THE ROBOT GARDEN, WITH THE CURRENT VERSION OF THE FEATURES DREAMED ON THE SITE. MINOR THINGS WERE IMPLEMENTED MANUALLY, SUCH AS THE POSITION OF THE POLES HOLDING SENSORS FOR TESTS WITH ROBOTS, AND A PLATFORM (LEFT OF IMAGE) TO HOLD A CONTROL DESK.

4 Conclusion – Machines hallucinating Architecture

In conclusion it can be stated that the Project Robot Garden serves as a first successful attempt to use machine hallucinations, based on architectural imagery, as the basis for a building project. Construction for this project began in May 2019. The main task of the design process presented in this paper was to analyze and explore how current techniques in AI applications can be utilized in architecture design. On a technical level it can be stated that there are shading/lighting/perspective/geometric cues that exist in images that contain 3D information, however it is unclear if 2D CNNs have a sense of a 3D model captured in their weights/learned representations. Therefore, using 2D image editing methods in 3D are not easily interpretable in terms of what features are being transferred and how 2D features are projected into the 3D space. Future work would be to generate fully 3D pipelines that allow us to explore design modifications in the purely 3D realm. On the other side this project only barely skims the surface of the possibilities,

in terms of speculating about possible applications in the discipline. Provided proper training CNN's and Generative **Adversarial Networks could learn how to dream urban textures in landscapes – serving as the basis for urban design. Or they can be used in projects of cultural preservation, in that they dream how to complete the restoration of historic buildings, or they can be used to optimize the planning of housing projects by learning to compare thousands and thousands of housing plans[17].**The opportunities are remarkable, and possibly will generate a completely new paradigm as of how to approach architecture design. In terms of disciplinary implementations, the project contributes to the discussion of style in the 21st century. Borrowing from the conversations of Gottfried Semper[18] about the nature of style it can be stated that style has turned into a posthuman quality, where artificial players contribute to the discussion by analyzing and proposing ideas for a cultural discussion with an ever-increasing speed. If robots can dream of gothic cathedrals, humans need to renegotiate their position in a contemporary design ecology.

5 Glossary – Mathematical Supplementary

We provide the mathematical formulation for the different 2D to 3D image editing methods in this section. Each of the following descriptions treats a neural network as a function whose parameters/weights map the input space of images to the output space of labels, or into the activation/feature space of a given network substructure, e.g. a layer or neuron. Additionally, when the term *error* is used, we are referring to calculating the Euclidean distance between two quantities. This is denoted by $| \dots |^2$ in the following equations and definitions.

Both image editing techniques, deep dream and neural style transfer, rely on backpropagation. Recall that, during the training of a neural network, changes in the weights of a neural network can be associated to changes in the prediction error by calculating the network gradient.

In contrast, if we fix weights of the neural network to be constant, we can calculate a gradient that describes how any change within the pixels of an input image can be associated with the change in activation of a particular neuron layer, or output class

within the network. Thus, if we increase the activation value of a neuron or layer, we can backpropagate that change to the input image pixels and change their intensity values accordingly. In effect, this process emphasizes the visual patterns/features in the input image that a neuron, layer, has learned are salient. This allows us to generate images shown in Figures 4 and 5 in the paper.

In the case of 3D input, we can calculate a gradient that associates the changes in a rasterized image to a corresponding change in the vertices and surfaces of the input mesh (via the 3D neural renderer), and then calculate a gradient associating the rasterized image with the activation of network substructures. This ultimately allows us to associate changes in the activation of a neuron or layer with a change in the input mesh.

2D to 3D Style Transfer

Using the learned representations of a pretrained image classification neural network, VGG-16, we can define a training objective that is based upon the learned spatial features, or 'content features' of the input 3D mesh, which we assign the name m^c and the 2D 'style features' of the second, guide image, which we assign x_s.

To make the shape of the generated mesh, m, similar to that of m^c, the 3D content loss can be defined as:

$$\boldsymbol{\ell}(m|m^c) = \sum_{v_i, v_i^c \in m, m^c} |v_i - v_i^c|^2$$

Where v_i is the set of vertices for the manipulated mesh *m* and v_i^c is the set of vertices for the original content mesh m^c.

The style loss is defined to be the same in the 2D image case using the rendered/rasterized image that is output from the 3D Neural mesh renderer:

$$\boldsymbol{\ell}(m|x_s, \phi) = |M(f_s(R(m, \phi))) - M(f_s(x_s))|_F^2$$

Where *R* is the 3D neural renderer function that projects the 3D mesh *m* to a rasterized 2D image, is the viewing angle at which to rasterize *m*, f_s is the pretrained VGG-16 network (used as a function) that projects the rasterized image into the feature space/representation of a specific network layer, and *M* is the Gram matrix function, which acts as a metric of style. The feature layers of the VGG-16 network used were *conv1_2*, *conv2_3*, *conv3_3*, and *conv4_3*. These two losses are summed to make the final objective, which is minimized via backpropagation to make the output mesh object.

2D to 3D vertex optimization

This method is similar to style transfer, but instead of using the learned feature representations to manipulate the mesh, the training method minimizes the error between the input mesh rasterized into a silhouette image and a guide silhouette image, $x_{silhouette}$. To rasterize a silhouette image from a mesh, the 3D neural mesh renderer is paired with a neural network that generates a silhouette image from the output of the mesh renderer. The training objective is to minimize the difference between the rendered silhouette image and the reference/guide silhouette image via backpropagation:

$$\boldsymbol{\ell}(m|x_{silhouette}) = |R(m|\phi) - x_{silhouette}|^2$$

3D Deep Dreaming

Deep dreaming, whether it is applied in 2D or 3D, is a visualization technique that allows us to qualitatively determine what visual features a given substructure of the network has learned. Conceptually, it makes assumptions similar to the grandmother cell hypothesis in Neuroscience: there is one neuron that is trained to be the detector for the face of a grandmother. Thus, this process of deep dreaming is more like hallucinating or pareidolia; the network is emphasizing vague pixel patterns in the image if those patterns resemble something that the neuron has learned to detect. In essence, we are seeing what the network is 'seeing' in the image.

To achieve deep dreaming, the training objective is to maximize the activation of a specific neuron, layer, or class by changing the values of the input image pixel intensities over many iterations. In 3D, the same objective is used to manipulated the vertices of a mesh object. Let *f(x)* be the GoogleLeNet pretrained neural network as a function that outputs an activation/feature map for the input image *x* at the specified neuron. The 3D neural mesh renderer is used to transform the mesh *m* into an image, which is then fed into GoogleLeNet to produce an activation map for the chosen neuron. A neuron in layer *inception_4* from GoogLeNet was used for all of the mesh manipulation.

The training objective for 3D deep dreaming to be optimized is

$$\ell(m) = -|f(R(m,\phi))|_F^2$$

where *R(m,)* is the rasterized image given the input mesh and viewing angle and *f* is the GoogleLeNet pretrained network that projects an image into the representation of the specified neuron.

References:

1: Turing A M. "Computing Machinery and Intelligence." Mind, Volume LIX, Issue 236, Oxford University Press, October 1950, P.433-460

2: See also:

3: *Obvious* is an art collective based in Paris, France that specializes on the use of AI applications for their works. Headed by Pierre Fautrel, Hugo Caselles-Dupre and Gauthier Vernier, the team regularly collaborates with artists around the globe.

4: Google's Artsexperiments platform is a repository for artworks that are primarily driven by the use of Chrome, Android, AI, Web VR and AR. https://experiments.withgoogle.com/ visited August 2nd 2019

5: Holly Herndon is an American composer who uses primarily the scripting language Max/MSP to create her music in a generative fashion

6: YACHT is a Los Angeles based dance music collective composed of Jona Bechtolt, Claire L. Evans and Rob Kieswetter. Their most recent single *(Downtown)Dancing* made heavily use of open source Artificial Intelligence and Machine Learning software.

7: Dadabots is the pseudonym of CJ Karr and Zack Zukovski who have developed several Machine Learning techniques to generate music. See for example their paper, *Curating Generative Raw Audio Music with D.O.M.E.* published at the *IUI Workshops'19*

8: Boden, M. A. In *Creativity and Artificial Intelligence,* Artificial Intelligence 103, Elsevier London 1998, p 347-356

9: See for example: Liang, Hui-Ying & Y. Tsui, Brian & Ni, Hao & C. S. Valentim, Carolina & Baxter, Sally & Liu, Guangjian & Cai, Wenjia & S. Kermany, Daniel & Sun, Xin & Chen, Jiancong & He, Liya & Zhu, Jie & Tian, Pin & Shao, Hua & Zheng, Lianghong & Hou, Rui & Hewett, Sierra & Li, Gen & Liang, Ping & Xia, Huimin. (2019). *Evaluation and accurate diagnoses of pediatric diseases using artificial intelligence*. Nature Medicine. 25. 10.1038/s41591-018-0335-9.

10: See also: Conrad, Jack G., Branting, L. Karl, Introduction to the special issue on legal text analytics, Artificial Intelligence and Law 2018 June 01 26 2 1572-8382 P 99-102

11: see also the section: *Learning Architecture features*

12: See for example Leach N., *Do Robots Dream of Digital Sheep?* Proceeding of the 2019 ACADIA Conference, Ubiquity and Autonomy

13: Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).

14: Simonyan, Karen, Andrea Vedaldi, and Andrew Zisserman. "Deep inside convolutional networks: Visualizing image classification models and saliency maps." arXiv preprint arXiv:1312.6034 (2013).

15: Gatys, Leon A., Alexander S. Ecker, and Matthias Bethge. "A neural algorithm of artistic style." arXiv preprint arXiv:1508.06576 (2015).

16: Kato, Hiroharu, Yoshitaka Ushiku, and Tatsuya Harada. "Neural 3d mesh renderer." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

17: See also:

18: Semper, Gottfried, Der Stil in den Technischen und Tektonischnen Künsten oder praktische Ästhetik: Ein Handbuch für techniker, Künstler und Kunstfreunde (Band 1): die textile Kunst für sich betrachtet und in Beziehung zur Baukunst. Verlag für Kunst und Wissenschaft Frankfurt am Main, 1860, P.13

19: Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. Deep Learning. The MIT Press.