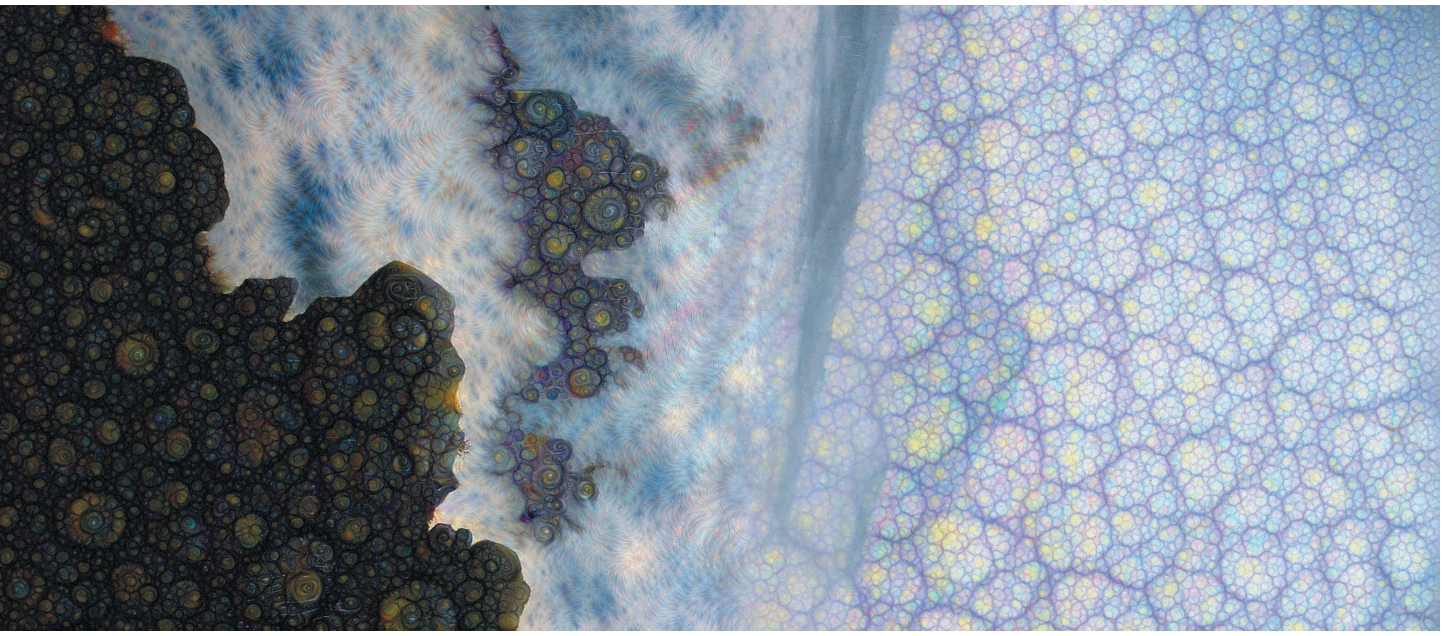
Kyle Steinfeld
University of California, Berkeley

Dreams May Come



1

ABSTRACT

This paper argues that prevailing approaches to CAD software have been fashioned to support modes of reasoning only of secondary importance to design activity, and that, due to some recent developments in computer vision, this state of affairs may be about to change. Surveying the current state of CAD tools, a critical position is developed based upon the best current understanding of the cognitive processes related to design. Following a high-level overview of some of the important developments in computer vision, and a curated set of examples of the applications these developments are finding in practices loosely related to architectural design, we draw out a number of parallels between machine learning (ML) and design thinking. We expect that this will serve as a guide to future research at the intersection of ML and architectural design tools.

1 An image generated using a convolutional neural network (CNN).

INTRODUCTION

The promise of computation as a partner in creative architectural design has not yet been realized. As seemingly complete as the "digital turn" in architecture has been in the past two decades, the prevailing approaches to CAD software have been fashioned to support a mode of reasoning that is only of secondary importance to design activity. The core of the problem is a simple disconnect: insofar as software presents impediments to a user seeing like a designer, software compels a designer to reason like a user. Put another way: *because computers cannot see the way we see, they cannot help us to reason the way we wish to reason*.

Implicit in this claim is the position that seeing is central to act of designing, and is intimately linked to a designerly way of thinking and working through a design problem. From this point of view, despite all the ways in which the use of computer software has supported and improved design activity—increasing the efficiency of design delivery, enabling a more collaborative design process, and opening up new frontiers of complex geometry—every piece of design software misses the mark in supporting the mode of reasoning that is most essential to our creative work: *seeing*. We may have reason to believe that this regrettable state of affairs is about to change, and that an entirely new paradigm of computation may be soon upon us. This optimism is motivated by an observation that moves slightly beyond the borders of the traditional design computation community, and draws from the broader digital culture: computers are being trained to see. In this text, we discuss how they're beginning to see (it's notable that they're being trained, not programmed), and how this new capacity might hold ramifications for how we approach software tools. It turns out that computers can see in ways that we cannot, and through mechanisms that even those who train them are only just beginning to understand.

Two caveats. First, it is clear that machine learning is a topic drawing intense interest, and many of the developments discussed below are quite recent. Some of the most exciting and relevant techniques have come to light within the last two years, and most haven't made their way to practical applications of any kind. We do not presume here to anticipate the precise nature of how these techniques will find their way to practical applications in architectural design software, and rather aim to offer a guide to the important contours of a transition in CAD that is bound to come. Second, we recognise that machine learning is a broad topic, and applies to a range of applications beyond computer vision that are highly relevant to architectural design, and likely have found their way into the pages of these proceedings. Our focus here will remain on creative architectural design, a highly visual domain, and on how new developments in computer vision might offer new opportunities.

THE ABDUCTIVE LOGIC OF DESIGN

Before detailing some of the innovations in computer vision that may be worthy of the attention of architectural designers, we first offer a diagnosis that may help ground the discussion. In this section, buttressed by a brief and high-level overview of the social science of CAD, we seek to flesh out a disconnect between the prevailing approaches to software tool development adopted by engineers, and the most current theories of design articulated by social scientists.

It may be self-evident that software developed to assist designers in our work follows from a set of implicit assumptions and explicit theories regarding the nature of design activity. For example, we may observe that thinking through drawings and images is fundamentally different than thinking through words or symbols (Knight and Stiny 2003), and that our tools might support one mode of thinking better than the other. Certain types of activities—we might, in unfairly broad strokes, call to mind the work of an engineer or scientist—predominantly employ linguistic or symbolic thinking. Other sorts of activities—here we might invoke the stereotypical image of a graphic designer or painter—predominantly employ visual thinking. While no activity draws exclusively from one domain, it is widely agreed that designers employ a uniquely balanced combination of both linguistic and imagistic thinking in our work. Despite the clear link between design tools and models of design, and despite the prominence of visual thinking in design, there persists a disconnect between the prevailing approaches to CAD tool development adopted by software engineers, and the most current theories of design articulated by social scientists. This disconnect hinges on the particular modes of reasoning that a software tool anticipates and is intended to support—inductive reasoning (that of science) and deductive reasoning (that of formal logic) find their origins in classical times. A third mode, abductive reasoning, has been identified more recently (Peirce 1974), and it relies upon the direct experience of the author and the context surrounding a problem. This third form has been called "the logic of design" (Cross 2011).

While the consensus view among design researchers is that the central form of reasoning employed in creative design is indeed abductive (March 1976), CAD tools have remained focused on supporting the inductive and deductive methods of reasoning more appropriate to technical design. As a result, contemporary architectural design tools tend to better support the later stages of design (in which well-bounded technical problems that benefit from inductive and deductive thinking prevail), while neglecting the needs of early-stage design (Visser 2006b). While the causes of this disconnect are varied, and likely include important social components, at least one technical component is clear: the most

prevalent, accessible, and well-developed computational techniques we have today are based on and directed towards the classical models of reasoning.

The dominant view in software development for the past 60 years, largely influenced by the work of Herbert Simon, is that design can be effectively characterized as an especially challenging form of problem solving (Simon 1973). From this point of view, design activity is a combinatorial search within a constrained space of all possible solutions; a process that may be effectively supported through the manipulation of logical representations decomposable into a fixed set of unambiguous primitives. Here, the classical modes of reasoning are emphasized, and design activity is seen to benefit primarily from deductive and inductive modes of thinking.

A minority view in software development is actually more widely accepted by cognitive scientists who study design. In this view, first articulated by Donald Schön, design is not search or problem-solving, but is itself a kind of "making"; designers make representations of a potential world and then construct the moves required to define solutions (Visser 2006a). This contrasts with the combinatorial search approach, in that problems are not given, but rather must be "constructed from the materials of problematic situations which are puzzling, troubling, and uncertain" (Schön 1983, 39). The representations that best support design from this point of view are inherently visual, as the ambiguity of visual material plays an important role in the construction of new worlds. Here, the central action of design involves abduction, wherein new potential orders are recognised based upon prior experience, and then acted upon.

Given this disconnect between the approach taken by developers of CAD software and the way designers tend to regard the nature of their work, we can see that while the most recent and most prominently discussed innovations in software (namely parametric modeling, optimization, simulation) enable their users to apply formal models of reasoning about design in much more powerful ways, they do nothing to support abductive reasoning or visual thinking. *So, how might alternative models of computation open up another path?* Recent developments suggest that the disconnect we have detailed here may be in flux, as a model of computation that facilitates abductive reasoning, machine learning, is rapidly emerging, and has proven increasingly viable for practical applications. Insofar as design tools are most effective when they are able to model and reflect the cognitive processes of the designers that employ them, such a development holds ramifications for how we approach software tools. Given that, as is detailed in the section below, machine learning is abductive by nature, we may consider that a better alignment is possible between our understanding of the nature of creative design and the development of tools that support this activity. Such an alignment suggests nothing less than an entirely novel approach to assisting design through computation.

LEARNING FROM EXPERIENCE

We return then to the central observation of this paper, drawn from the broader digital culture: *computers are being trained to see, and this new capacity matters to design*. To better appreciate the new developments in computer vision, and to consider what ramifications they may hold for architectural design, we must hold a rudimentary understanding of the underlying technology. As such, following an abundance of such guides (Kogan and Tseng 2017; Nielsen 2015; Hinton 2017), we offer here a brief overview of machine learning for a lay audience, with an emphasis on how it differs from other forms of programming. These details are important, as they demonstrate the link between vision, recognition, and abductive reasoning, both in humans and machines.

Machine learning (ML) is a subfield of artificial intelligence that employs processes of knowledge discovery (Kohavi and Provost 1998) that rely upon the preexistence of large datasets. ML has been broadly defined as learning through observation, in which patterns are mapped onto other patterns without any intervening representations (DeLanda 2012; Bechtel and Abrahamsen 2002). It is unique, and uniquely successful, in that it does not define deductive or inductive chains of reasoning in advance, but rather allows the machine to learn through its own experience. Critically, under this mode of artificial intelligence, *we do not program computers, we train them*. For example, let's imagine that we wanted a computer to be able to recognize images of hand-written numbers. This is a classification problem: we want the computer to recognize a digit and label it correctly. We may be tempted to approach such a task by describing the most important features of each digit, to explain to the computer how to recognise loops, circles, and lines, and how these go together to form digits. But this is not the ML way. Instead, the most successful models for handwriting recognition do not begin with explicit instructions, but rather with a very large database of right answers: images of digits written by people, each of which is labeled with the name of the digit it represents. This dataset represents the so-called "training set," which forms the body of experience offered to a computer model that allows it to make predictions about new handwritten characters it may encounter.

In terms of matching patterns to patterns, in this case the input pattern is contained within the pixels in an image of a hand-written number. The output pattern is a prediction—one that

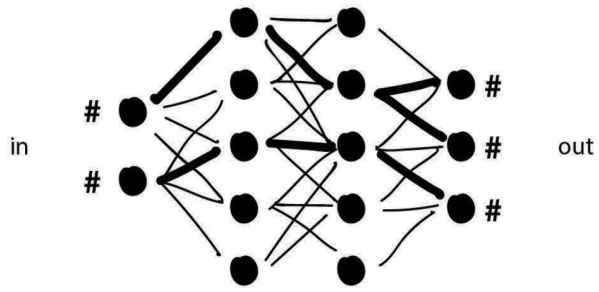
can be described as ten percentages, each of which is a measure of the likelihood that the given image depicts one of the integer numbers 0–10. As the details of this process have been well documented elsewhere, here we offer some key points regarding the general approach:

First, as we have discussed, *there exists no pre-defined internal representation*. In ML, *this emerges only through experience*. The lack of an internal representation offers an odd side-effect that we'll soon get to: since the computer has effectively programmed itself, it can be difficult for humans to understand how these models perform the task they were trained for. Next, we have introduced the idea of *training*, which is about allowing a model to build experience. Broadly speaking, the more samples a model can draw from, the better it is able to perform. Finally, we can see that this sort of process can work well with *loosely defined problems for which there are large unstructured datasets*. For this reason, ML is a popular approach to big data problems.

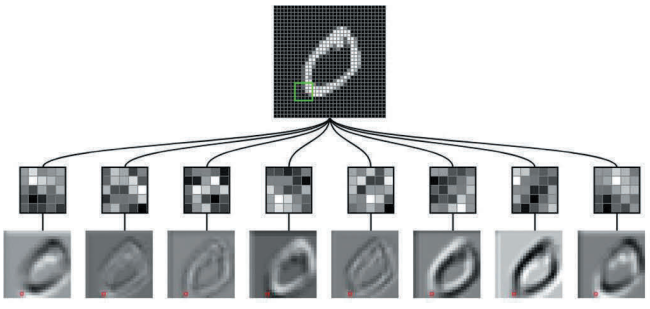
We may note that the process by which a trained model is able to classify novel observations is based not upon any predefined formal chain of deductions, but rather by recognising new orders based upon prior experience. This is not unlike the way that social scientists might characterize the abductive nature of creative design. As such, it is perhaps not a coincidence that models such as this are at the heart of recent advances in computer vision. These recent advances, especially those that have gained widespread attention in the past five years, are largely due to a particular approach to ML called a neural net. Neural nets are actually nothing new, and trace their origins back to the early days of AI (Widrow and Lehr 1990). Based loosely on an abstract model of the human brain, they were first formalized in the late 1940s and 1950s, but fell out of favor in the time since. Having made quiet gains throughout the 1990s and early 2000s, neural networks have recently become the dominant strain of ML algorithms.

The basic setup is this: a data-processing network is established for matching patterns of input data to patterns of output data. For those familiar with parametric modeling, think of this as a Grasshopper model: each node of this network is a container for or operation upon a piece of data. Input data comes in from the left of the nearby diagram, and the job of the network is to produce certain patterns on the right by processing it through layers of neurons in between. Crucially, this network is established in a generic and reconfigurable way. While the connections between nodes are not typically rewired, the weights assigned to them are. Different configurations of weights result in different manipulations of the input data, and therefore different output patterns. This is how a neural net can "learn": by iteratively adjusting the configuration of weights in the network to better match input data, the model can be gradually nudged from producing random and incorrect results to ones that perform as desired. This iterative reconfiguration of edge weights based on a known dataset is called training.

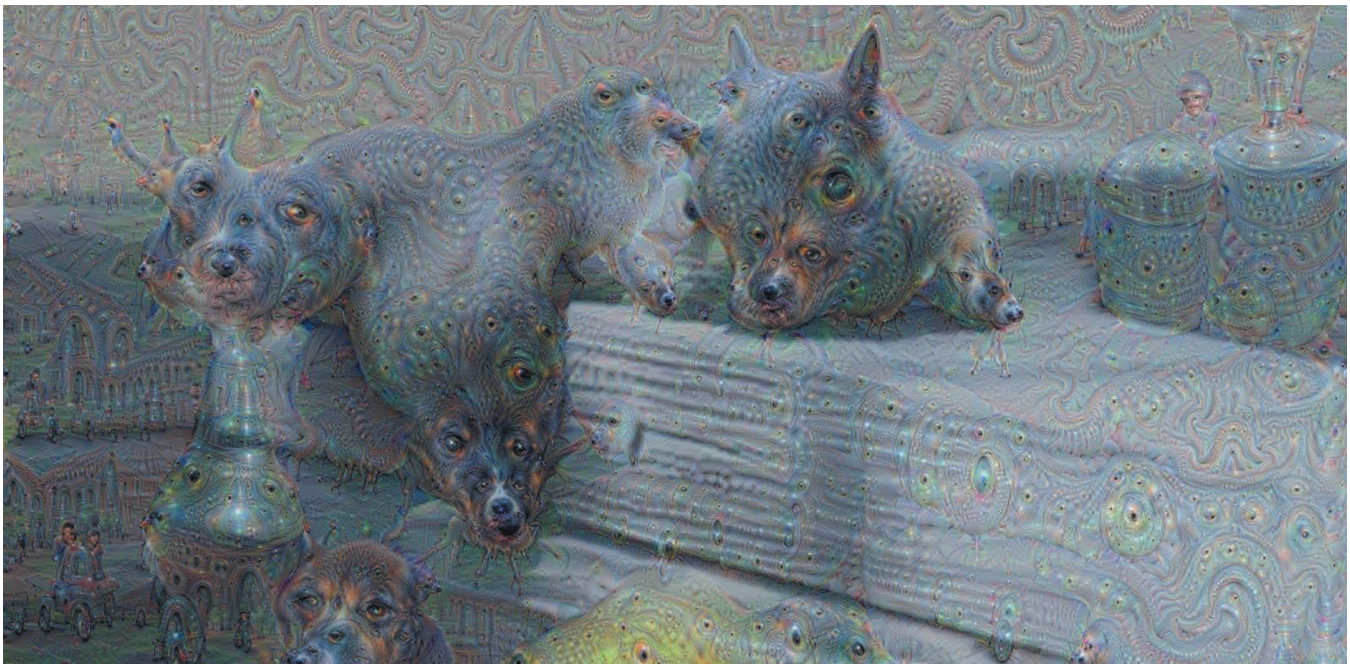
This, in summary, is how a general neural net operates. Through a series of recent innovations too technical to get into here, this model of ML has enabled a rush of innovations in the past five years, and holds widely recognised promise for a range of applications. To understand how this process pertains to computer vision, as we saw in our handwriting example, we'll require a way of transforming images in a way that can play a meaningful role in the network. Such is the rationale behind a process called "convolution," a relatively recent innovation in the definition of neural nets, which acts as something like a filter for detecting features of an image. At each node in a convolutional neural network (CNN), this filter can be applied and tuned to highlight



2 A diagram of the data-processing network that constitutes a neural net. Data flows from left to right.



3 A demonstration of convolutional filters applied to an image of a hand-drawn digit.



4 "Image Generated by a Convolutional Network - r/MachineLearning." Reddit, June 16, 2015

certain specific features. For example, the nearby figure shows a hand-drawn digit with different filters applied to it. Some of these call out vertical lines, some horizontal, and others find corners. Convolution is designed to be applied in layers: each layer of neurons transforms the given data in anticipation of a higher-level layer, meaning one that contains a more compact and salient representation of the given data. In this way, layers trained to find low-level features—such as edges, corners, and patterns—lay the groundwork for higher-level layers that can identify wrinkles, eyes, and faces. The use of convolutional filters allows us to construct a neural network capable of transforming image data, and to train neural nets to recognize patterns based on information found in large datasets of images.

Research has shown the CNN to be a remarkably resilient approach for image-recognition tasks. CNNs have been developed that can successfully recognise handwritten digits (LeCun et al. 1989); species of flowers (Choi 2015), dogs (@hartator 2016), and birds (Branson et al. 2014); discerning hot dogs from things that are not hot dogs (Yang 2017); and, as described below, queries of satellite imagery that operate not by textual searches, but by example (Levin et al. 2016). This approach underlies the most successful large-scale visual recognition systems, such as GoogLeNet (Szegedy et al. 2014).

It is worth reiterating that in each of these cases—since the configuration of weights was not explicitly laid out by a person, but rather was the product of the training of the model—no human programmer has any special insight as to how a specific neural net is able to function. Nobody really knows how such a model is able to distinguish between hand-written sevens and nines, or calicos and himalayas. In this sense, a trained neural net is the ultimate black box. It may be useful to think of these programs as just that: recognition machines.

Going back to our definition of ML, we may recall that we distinguished this approach from other forms of AI as one that maps patterns to patterns with no intervening representations (i.e., no representations crafted by a human). It turns out that it's not entirely true that nobody knows what's happening inside one of these models. In the past few years, techniques have been developed to better understand and visualize the inner workings of CNNs. These visualizations are the key to lead us out of the analytical applications of ML to generative ones. Before discussing this shift, we present here a couple of examples of analytical application of CNNs that will help us make our way back to how ML holds ramifications for creative architectural design.

Quick Draw

One project demonstrates that ML can perform classification tasks not just on photographic images, as described above, but on a format clearly relevant in architectural design: sketches. Quickdraw (Jongejan et al. 2017) is a web-based project by Google Creative Lab and Google Data Arts that functions like a game of charades played in partnership between a human and a computer. Given what has already been presented, the approach taken here is clear: a neural net has been trained to classify

quickly drawn sketches of a limited set of concepts. As a player of the game draws, this trained model tries to guess the concept being conveyed. Right or wrong, the final human-drawn sketch is fed back into the system for training, thereby allowing the model to improve over time. The creators of this project claim to have already assembled the world's largest dataset of hand-drawn doodles, and aim to demonstrate the applicability of machine vision in the visual arts.

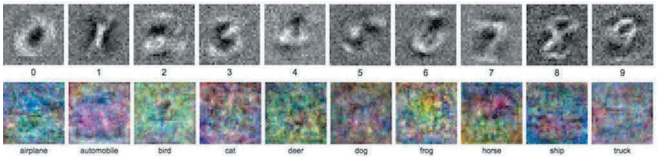
Terrapattern

Recalling that neural nets work best on large unstructured datasets, of note here is how the existing research has taken advantage of one of the most easily accessible datasets related to environmental design: satellite imagery. Terrapattern (Levin et al. 2016) is a web tool that allows a user to perform geographic searches in an entirely new way: by performing queries by example. Rather than enter a textual search term, users may simply point to the thing that looks like what they are interested in, and the tool returns more examples of that thing alongside the relevant locations. This project employs a deep convolutional neural network, similar to the handwriting recognition model discussed earlier. Hundreds of thousands of labeled satellite images were employed to train the CNN to distinguish one place from another. In the process, high-level visual features important for classification were modeled. By stripping off the final layer of neurons, the ones that predict category labels, a kind of fingerprint for each satellite image is produced. This allows for any given image to be matched with others that exhibit similar fingerprints.

THE GENERATIVE TURN

As compelling as it may be to have the ability to train computers to find loosely defined patterns in large datasets of images, as we saw in the Terrapattern project, or to recognise concepts in hand-drawn sketches, as we saw in Quick Draw, these abilities do not yet meet our aim of supporting abductive thinking in design. Abduction is reasoning not about what is, but what is possible. Unlike conventional logic, a design solution cannot be derived directly from the problem, but can only be matched to it. In other words, "deduction proves that something must be; induction shows that something actually is operative; abduction suggests that something may be" (Peirce 1974). The projects discussed above are analytical, and nothing new has truly been created or discovered by the computer. The key moment for these processes taking a generative turn came about two years prior to the time of writing. It was not intended as a generative process; rather, it was a byproduct of computer vision researchers attempting to better visualize what was happening inside a neural network.

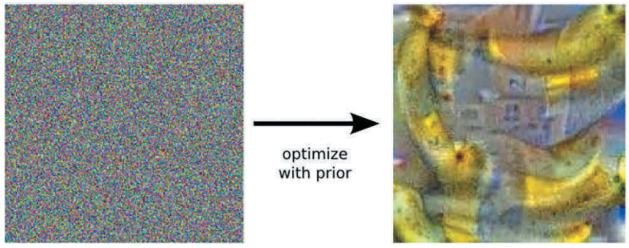
On June 16th of 2015 the image seen on the adjacent page was posted to Reddit ML group (swifty8883 2015), paired only with the caption "Image generated by a Convolutional Network." The Reddit group responded in disbelief, speculating that the image was fraudulent and must have been the work of a human artist, or have been some sort of composite or manipulation of an existing image. Twenty-four hours later, a follow-up post to Google's research blog (Mordvintsev, Olah, and Tyka 2015) verified that the image was indeed primarily computer-generated. It was "authentically artificial." It's difficult to overstate the importance of this moment, and the shockwaves it sent first through the machine learning research community, and then through communities as disparate as fashion (Darby 2015), fine art (CubeBreaker 2015; Wayne 2015; Galperina 2015; Cheung 2015; McDonald 2016), and oneirology (Turner 2017). As suspected by the commenters on Reddit, this image did begin as a photograph, but the forms and details that give it its character were generated by a machine. The slug-like creature at the center, which appears to be comprised entirely of puppies and eyes, the pagoda-like building in the lower left, the tiny image of a child on the right, all of these were conjured up by a CNN based on structures in the base image that it incorrectly recognized based on its prior experience and training. What we see here is a hallucination: the dream of an artificial mind. *When the history of the creative application of artificial intelligence is written, this image will surely figure prominently*. In their blog post, the research scientists at Google included several other similar images, and also detailed the method by which these were created, and termed the the script that generated them "Deepdream," which visualized the GoogLeNet model. To understand this process, we must recall an important property of neural networks: that nobody knows exactly how they work. Since they are trained, not explicitly programmed, no human programmer has any special insight into how they do what they do. Because of this property, visualizing what's happening inside—which nodes and weights are important, for example, or which features of an image trigger recognition—is a topic of interest in computer vision circles, and a number of techniques for doing just this have been developed.



5 A visualization of the importance played by pixel position in a model for recognizing handwritten digits.

Returning to the classic example of handwriting recognition, we can understand one such technique. Imagine a simple 1-layer neural network trained to recognise handwritten digits. Input neurons each represent a specific pixel in the given image, while output neurons represent the likelihood that this image contains a particular digit. The nearby images (Figure 5), which seem to show barely recognizable digits, were produced by visualizing which pixels play the most important role in making a determination for a given digit. This is quite remarkable. Recall that no programmer has explained to the network anything about digits or what they mean, and yet the configuration of weights has come to resemble these classes of forms regardless. Each of these represents not any particular form of a digit, but rather embodies some important characteristics shared by all forms of this digit that have been encountered by the model. What we see here is a representation of the experience of the neural net. Some may go so far as to term this a platonic digit.

The researchers at Google held a similar aim, as demonstrated by this simple example—to better understand the role of particular nodes, edges, and layers in a neural net—but took a slightly different approach that employed a well-established process for visualizing features of a deep neural network (Erhan et al. 2009).



6 A visualization of higher-layer features of a deep network. Starting with an image of static, researchers iteratively made small changes in this image until a particular recognition neuron (in this case, a neuron that recognizes bananas) was satisfied.

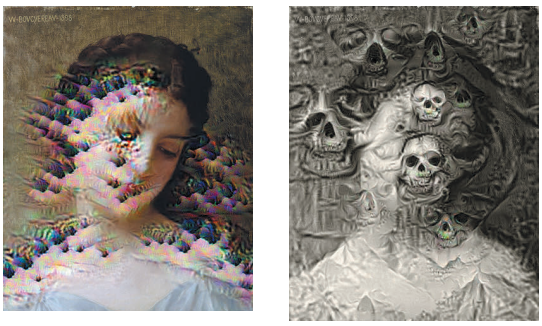
Investigating a model that was trained to recognize different categories of images, they recursively fed this model images, tweaking them a bit each time toward images that better satisfy what the model considers to be a given category. If we were interested in what sort of image would result in "banana," for example, we could start with an image of static and gradually nudge it toward one that better depicts the idea of a banana, from the neural net's point of view. It's worth noting that, while the image seen nearby that shows the result of this process is not of any particular banana, it does embody a certain "banananess" that we can all recognize.

From this process of visualizing the inner workings of a CNN, it is a small but significant step to producing the sort of hallucinations we saw earlier: rather than optimizing for the activation of one particular category of neuron, it is possible to optimize for patterns that activate any category of neuron. Rather than directing their model to recognize and act upon things that look like bananas, the researchers directed it to identify and enhance anything it recognized from its previous experiences. If a bit of an image is recognizable, or bears a resemblance to a pattern the model has encountered, it responds by enhancing those qualities. This basic approach—defining transformations of image data that rely on a trained CNN model—has borne fruit in a surprising range of applications, the first crop of which have already found their way to market. Some of these are detailed below.

More than any current or proximate practical software products, generative applications of machine learning also suggest a fulfillment of the charge laid down in the introduction of this text: that design tools should both support and reflect the cognitive processes of the designers that employ them. The parallels between ML and a designerly approach to problems are remarkable. Both designer and machine do not rely on preconceived representations that link problem and solution. Instead, both draw heavily from past experience. Similarly, in the sense that both rely on the recognition of and response to similar patterns, both proceed through what might be seen as allegory. We will expand on these parallels, and the potential they hold for design tools, in a section below. First, it may be illuminating to detail how generative machine learning is currently finding utility. The examples of applications of ML and computer vision described below represent a limited selection, intended to both illustrate the salient approaches and to draw some still-emerging distinctions among the varied applications of these techniques in practice. Each of the examples detailed below operate upon raster images, a format which holds immediate application in creative disciplines that trade in images as their primary currency. As architectural design finds value in a broader range of formats than those that have been addressed in these early days of creative and generative ML, we ask our readers to join us in speculating on how the underlying approaches exemplified by these projects might find their way into architecture.

Deepdream

The GoogLeNet model and the deepdream script were posted publicly and quickly reapplied by an enthusiastic group that dabbles at the intersection of visual art and technology. The images seen nearby are, of course, not photographic manipulations, but rather were generated by the deepdream script and the GoogLeNet model trained to recognize certain categories of forms. These tools in the hands of early adopters generated a wealth of material, but one that was seen as limited in its aesthetic variation (Galperina 2017): a limitation that is perhaps more a function of the particular tastes and fascinations of the



7, 8 Two pieces of visual art from Mario Klingemann's Deep Album series, produced by running the deepdream algorithm with a model trained on classifying album covers. This work demonstrates a new domain of artistic agency: the selection and training of recognition models.

initial user group (Pieters 2017) than inherent in the script itself. However, variants of this basic approach in more skillful hands demonstrate the multiple opportunities for authorship presented. Whereas the GoogLeNet model was trained for general use, a set of experiences that somehow lent this model a proclivity for images of eyes and small dogs, training a custom model with a particular application in mind opens up entirely new ground for crafting the set of forms available for recognition.

As an illustration, consider a piece by artist Mario Klingemann, who often uses ML in his work, in which the deepdream script was run using a model trained only on images of album covers. Because its experience had been carefully curated, like a designer on a grand tour of European classical buildings, images of faces, titles, and musicians formed the bulk of the experience of this model, and these images loomed large in the patterns it tended to recognize.

The Next Rembrandt

A collaboration between ING, advertising agency J. Walter Thompson, and researchers from Microsoft and TU Delft, the Next Rembrandt project (Korsten and Flores 2016) demonstrates the capacity of ML, given a sufficient corpus of work to draw from, to capture and replicate artistic style. Working from the considerable body of work of the Dutch master, researchers paired a facial recognition algorithm capable of identifying elements of human figures and an ML model to classify the most typical geometric patterns to render these figures in paint. In a manner similar to the Deepdream project, having internalized the style of Rembrandt, this model could then be employed to generate entirely new instances of his work. The resulting work succeeded more in provoking discussion than in accurately reproducing the work of the artist. While convincing to the layperson, Rembrandt scholars were not as impressed. What ramifications might an analogous development hold in architectural design, if the style of some great masters such as Palladio (Stiny and Mitchell 1978) or Siza (Duarte 2001) were able to be captured and reproduced in such a manner?

Style Transfer

Following up on the previous example, the term "style transfer" brings together the related work of a number of researchers in deep learning for capturing and transferring artistic style more generally. Unlike the example above, which attempts to reproduce the style of a single great master, and toward this end relies on a deep corpus of primary material, this area of research attempts to map the style of a single image onto another. Some incarnations of this work (Zhu et al. 2017) operate semantically—changing every horse it encounters to a zebra, for example—while others (Luan et al. 2017) operate on the basic elements of the image itself—he hue, saturation, value, and texture—to alter the quality of a photograph. Basic research in this area has quickly found its way to market, with a host of applications. Prisma (Moiseenkov 2017), for example, offers functionality similar to that of a Photoshop or Instagram filter, but one that is easily customizable, provided a single exemplar that embodies the style of image desired to guide the process. Here we find the clearest case for design by example. Extrapolating such an approach to architectural design, style transfer suggests perhaps the most stunning ramifications: starting with some number of precedents for the formal effects we desire, we would no longer need to define explicit routines to achieve these in our own work.

CONCLUSION

This paper has argued that prevailing approaches to CAD software have been fashioned to support modes of reasoning only of secondary importance to design activity, and that, due to some recent developments in computer vision, this state of affairs may be about to change. We began with a survey of the current state of CAD tools, and developed a critical position regarding these tools based upon our current understanding of the cognitive processes related to design. Following a high-level overview of some of the important developments in computer vision, we proposed a number of parallels between machine learning and design thinking:

- Both draw from past experiences.
- Neither rely on preconceived representations that link problem and solution.
- It is possible for each to proceed through allegory, as we have seen in our speculation on "design by example."

This text has told just half a story. The other half is currently being written, and we have not yet explored if it is possible to

move from the deductive and inductive models of software tools to an abductive one. The model of computer-assisted design in which we explicitly describe not only what we want but also precisely how to realize it may be giving way to a model in which we simply describe the qualities we're looking for, and allow the machine to offer up a range of approaches as to how to achieve it. This shift from explicit definitions of logic and geometry to design by example is profound. Soon, we may be paying less attention to composing elaborate chains of parametric logic and programming, and more to training artificial assistants; semi-autonomous agents able to act on our behalf. In time, we may look back on the computer-aided design tools we currently employ, and wonder how they really aided us at all.

As a closing thought, in a work otherwise content to speculate on long-term implications, we offer here just one short-term call to action. It has been said that many machine-learning breakthroughs are constrained, not by the limitations of algorithms, but by the availability of high-quality training datasets.

REFERENCES

Anonymous. 2015. "This Image Was Generated by a Computer on Its Own (from a Friend Working on AI) - R/Creepy." Reddit, June 10, 2015. https://www.reddit.com/r/creepy/comments/39c6ta/this_image_was_generated_by_a_computer_on_its_own/.

Bas Korsten, and Emmanuel Flores. 2016. "The Next Rembrandt." J. Walter Thompson Amsterdam. https://www.nextrembrandt.com.

Bechtel, William, and Adele Abrahamsen. 2002. *Connectionism and the Mind: Parallel Processing, Dynamics, and Evolution in Networks*. Malden, MA: Wiley-Blackwell, 2002.

Branson, Steve, Grant Van Horn, Serge Belongie, and Pietro Perona. 2014. "Bird Species Categorization Using Pose Normalized Deep Convolutional Nets." arXiv:1406.2952

Cheung, Ysabelle. 2015. "Was This Psychedelic Image Made by Man or Machine?" *Vice Creators*, June 18. https://creators.vice.com/en_us/article/was-this-psychedelic-image-made-by-man-or-machine.

Choi, Sungbin. 2015. "Plant Identification with Deep Convolutional Neural Network: SNUMedinfo at LifeCLEF Plant Identification Task 2015." Working Notes of International Conference of the Cross-Language Evaluation Forum for European Languages, September 8–11. Conference. Toulouse, France: CLEF.

Cross, Nigel. 2011. *Design Thinking: Understanding How Designers Think and Work*. Oxford, UK: Bloomsbury Academic.

CubeBreaker. "Inside the Mind of a Computer – Take a Look at These Images Created by Google's Artificial Neural Networks." Cube Breaker, June 23, 2015. http://www.cubebreaker.com/inside-the-mind-of-a-computer-take-a-look-at-these-images-created-by-googles-artificial-neural-networks/.

Darby, Luke. 2015. "Time to Find a Real Job? Google Just Made an A.I. That Makes Gorgeous, Trippy Artwork." *GQ*, June 21, 2015. http://www.gq.com/story/google-artist-ai.

DeLanda, Manuel. 2002. "Deleuze and the Use of the Genetic Algorithm in Architecture." *Architectural Design* 71 (7): 9–12.

———. "The Use of Genetic Algorithms in Art." In *Synthetic Digital Ecologies: Proceedings of the 32nd Annual Conference of the Association for Computer Aided Design in Architecture*, edited by Jason Kelly Johnson, Mark Cabrinha, and Kyle Steinfeld, 25–31. San Francisco: ACADIA.

Duarte, José Pinto. 2001. "Customizing Mass Housing : A Discursive Grammar for Siza's Malagueira Houses." Ph.D. Thesis, Massachusetts Institute of Technology. http://dspace.mit.edu/handle/1721.1/8189.

Erhan, Dumitru, Aaron Courville, Yoshua Bengio, and Pascal Vincent. 2009. "Visualizing Higher Layer Features of a Deep Network." *Technical Report 1341, DIRO, Université de Montréal*.

Galperina, Marina. 2015. "Is Google's Deep Dream Art?" *Hopes&Fears*, July 14, 2015. http://www.hopesandfears.com/hopes/culture/is-this-art/215039-deep-dream-google-art.

Gatys, Leon A., Alexander S. Ecker, and Matthias Bethge. 2016. "Image Style Transfer Using Convolutional Neural Networks." In 2016 IEEE Conference on Computer Vision and Pattern Recognition, 2414–23. Las Vegas, NV: CVPR. doi:10.1109/CVPR.2016.265.

Gedenryd, Henrik. 1998. "How Designers Work: Making Sense of Authentic Cognitive Activities." Ph.D. Thesis, Lund University. https://lup.lub.lu.se/search/publication/d88efa51-c2f9-4551-a259-00bd36fe8d03.

Golan Levin, David Newbury, Kyle McDonald, Irene Alvarado, Aman Tiwari, and Manzil Zaheer. 2016. "Terrapattern: Similar-Image Search for Satellite Photos." Terrapattern. http://www.terrapattern.com.

@hartator. 2016. "Dog Breed Identifier." https://hartator.github.io/dog-breed-identifier/.

Hinton, Geoffrey. 2017. "Neural Networks for Machine Learning." Coursera. https://www.coursera.org/learn/neural-networks.

Jonas Jongejan, Henry Rowley, Takashi Kawashima, Jongmin Kim, and Nick Fox-Gieg. 2017. Quick Draw. https://quickdraw.withgoogle.com/.

Knight, Terry, and George Stiny. 2003. "Classical and Non-Classical Computation." *arq: Architectural Research Quarterly* 6 (1): 5–10.

Kogan, Gene, and Francis Tseng. 2017. "Machine Learning for Artists." Machine Learning for Artists. https://ml4a.github.io/.

Kohavi, Ron, and Foster Provost. 1998. "Glossary of Terms." Machine Learning 30 (2–3): 271–274.

LeCun, Y., B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. 1989. "Backpropagation Applied to Handwritten Zip Code Recognition." *Neural Computation* 1 (4): 541–51. doi:10.1162/neco.1989.1.4.541.

Levy, Steven. 2015. "Inside Deep Dreams: How Google Made Its Computers Go Crazy." Medium, December 11, 2015. https://medium.com/backchannel/inside-deep-dreams-how-google-made-its-computers-go-crazy-83b9d24e66df#.uridzada1.

Luan, Fujun, Sylvain Paris, Eli Shechtman, and Kavita Bala. 2017. "Deep Photo Style Transfer." arXiv:1703.07511 [Cs], March 22, 2017.

March, Lionel. 1976. "The Logic of Design and the Question of Value." In *The Architecture of Form*, edited by Lionel March. Cambridge, UK: Cambridge University Press.

McDonald, Kyle. 2016. "A Return to Machine Learning." Medium, October 7, 2016. https://medium.com/@kcimc/a-return-to-machine-learning-2de3728558eb.

Moiseenkov, Alexey. Prisma. Accessed May 21, 2017. https://prisma-ai.com/.

Mordvintsev, Alexander, Christopher Olah, and Mike Tyka. 2015. "Inceptionism: Going Deeper into Neural Networks." *Google Research Blog*, June 17, 2015. https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html.

Nielsen, Michael A. 2015. *Neural Networks and Deep Learning*. Determination Press. eBook. http://neuralnetworksanddeeplearning.com.

Peirce, Charles Sanders. 1974. *Collected Papers of Charles Sanders Peirce*. Cambridge, MA: Harvard University Press.

Pieters, Roelof. "Deep Dreaming Fear & Loathing in Las Vegas: The Great San Francisco Acid Wave." YouTube video, uploaded July 3, 2015. Accessed May 22, 2017. https://www.youtube.com/watch?v=oyxSerkkP4o.

Schön, Donald. 1983. *The Reflective Practitioner: How Professionals Think in Action*. New York: Basic Books.

Simon, Herbert. 1973. "The Structure of Ill Structured Problems." *Artificial Intelligence* 4 (3–4): 181–201.

Stiny, George, and William J. Mitchell. 1978. "The Palladian Grammar." *Environment and Planning B: Planning and Design* 5 (1): 5–18. doi:10.1068/b050005.

swifty8883 [Unknown Author]. 2016. "Image Generated by a Convolutional Network - R/MachineLearning." Reddit, June 16, 2015. https://www.reddit.com/r/MachineLearning/comments/3a1ebc/image_generated_by_a_convolutional_network/.

Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2014. "Going Deeper with Convolutions." arXiv:1409.4842 [Cs], September 16, 2014.

Turner, Rebecca. "Inside Robot Dreams: What Google's DeepDream Bot Thinks About." World of Lucid Dreaming. Accessed May 19, 2017. http://www.world-of-lucid-dreaming.com/inside-robot-dreams-what-the-google-ai-bots-think-about.html.

Visser, Willemien. 2006a. "Designing as Construction of Representations: A Dynamic Viewpoint in Cognitive Design Research." *Human–Computer Interaction* 21 (1): 103–52.

———. 2006b. *The Cognitive Artifacts of Designing*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Wayne, Eric. 2015. "Google Deep Dream Getting Too Good." Art & Criticism, July 7, 2015. https://artofericwayne.com/2015/07/08/google-deep-dream-getting-too-good/.

Widrow, B., and M. A. Lehr. 1990. "30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation." *Proceedings of the IEEE* 78 (9): 1415–42. doi:10.1109/5.58323.

Yang, Jimmy O. 2017. Not Hotdog (version 1.0). OSX. SeeFood Technologies Inc.

Zhu, Jun-Yan, Taesung Park, Phillip Isola, and Alexei A. Efros. 2017. "Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks." arXiv:1703.10593 [Cs], March 30, 2017.

IMAGE CREDITS

Figures 3 and 5: From Machine Learning for Artists, Kogan, Tseng, Refsgaard, 2017. Reproduced with permission from artist.

Figure 4: swifty8883 [Unknown Author]. "Image Generated by a Convolutional Network - r/MachineLearning." Reddit, June 16, 2015.

Figure 6: Mordvintsev, Olah, and Tyka. "Inceptionism: Going Deeper into Neural Networks." Google Research Blog, June 17, 2015.

Figures 7 and 8: Mario Klingemann, 2015. Reproduced under a CC BY-NC 2.0 license. https://www.flickr.com/photos/quasimondo/23095678811/in/album-72157661320184201/

All other drawings and images by the authors.

Kyle Steinfeld is an Assistant Professor specializing in digital design technologies in the Department of Architecture at the University of California, Berkeley.