Danil Nagy Follow
Jan 23, 2017 · 7 min read

The problem of learning

This course will teach the concepts of generative design, specifically geared towards applications in architectural design. The goal of the course is to present both tools and concepts in a way that helps designers build intuition for how these tools work, how they can affect the design process, and what they mean for the future of design practice.

Before we dive into the subject of generative design and all the tools and technologies that enable it, let's first consider how generative design fits within a broader context of artificial intelligence. Let's start with a basic definition of generative design taken from promotional material published last year on Autodesk's website:

> *Generative design is a technology that mimics nature's evolutionary approach to design. It starts with your design goals and then explores all of the possible permutations of a solution to find the best option. Using cloud computing, generative design software quickly cycles through thousands— or even millions—of design choices, testing configurations and* ***learning*** *from each iteration what works and what doesn't. [This] process lets designers generate brand new options, beyond what a human alone could create, to arrive at the most effective design. (emphasis added)*

Perhaps the most important concept from this description is that this technology can learn from designs it has analyzed, and apply that knowledge to generate new, better performing designs. This ability of the system to learn over time places generative design within a larger framework of artificial intelligence (AI), which describes any computer system that can reason and make decisions that are not explicitly programmed by a human. Although true "strong" AI is still not technically possible and is more in the realm of science fiction than hard science, the last decade has seen impressive gains in several domains. For example, Google alone has created systems that can automatically learn to categorize images, translate between languages, and play highly complex games like Go better than human experts.

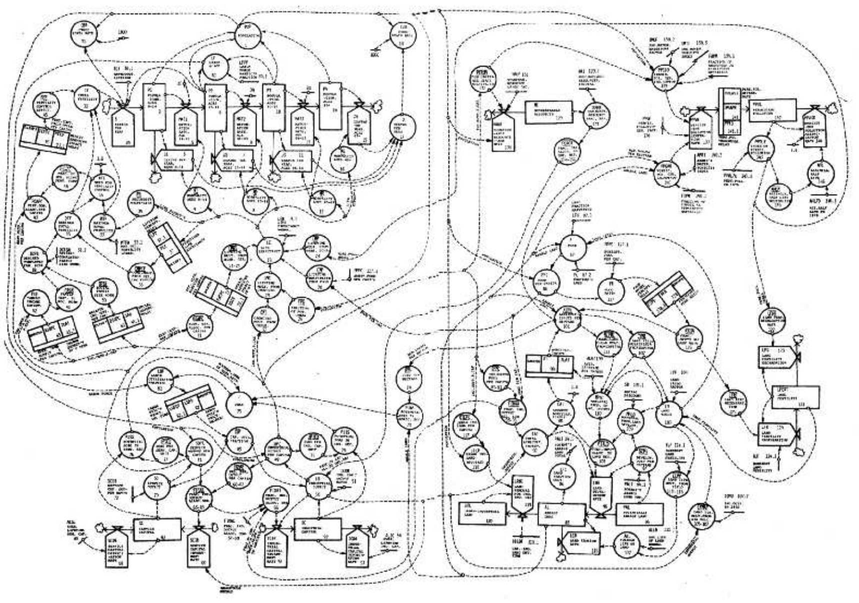
The key to these developments was the realization that we as humans are simply not capable of creating an artificial intelligence from scratch.

In the early days of computer development, it was assumed that once computers were powerful enough, we would be able to program them to have intelligences similar to our own. Considering the great progress that was being made, there was no reason *not* to believe it—anything seemed possible.

However, as the hardware capabilities of computers continued to advance rapidly, it became apparent that artificial intelligence was not as inevitable as it was once thought. There was something missing, a crucial limitation that could not be solved by rapidly increasing processing speeds or memory capacities. The key—which has since been exploited in almost every form of modern artificial intelligence—was to not directly program the intelligence at all. Instead of dictating exactly how the intelligence would work, we could create an abstract and flexible model which can develop and learn to become intelligent *on its own*.

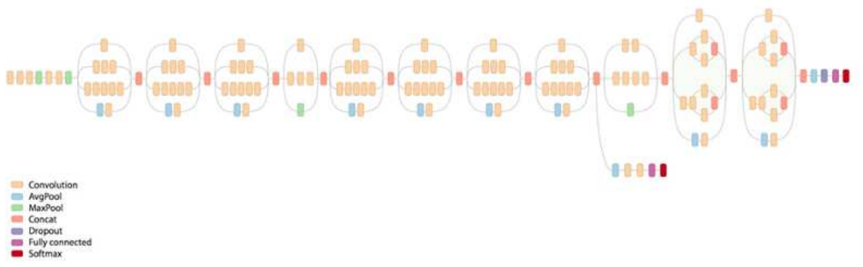
Consider two very different approaches to modelling a complex system:

The first comes from the Limits to Growth model, one of the first comprehensive computer models created to simulate the entire world. Developed by a group of researchers at MIT in 1972—the early days of computer development —the model imagines the world as a series of resource flows, operating in a variety of feedback loops which affect each other in dynamic ways. While this model provides many insights and should be commended for its early contribution to computer modeling and simulation (not to mention its apparent accuracy), it possessed one critical flaw. Though the model was exquisitely complex in how it modelled the world's dynamics, the model itself was inherently static. Since the design of the system relied solely on the expertise and intuition of its human designers, it could never adapt to changing conditions or incorporate new data over time. In effect, in could not *learn*. It was inevitable that the model would reproduce results much along the lines of what the researchers were already predicting.



Meadows, et al. The Limits to Growth (1972)

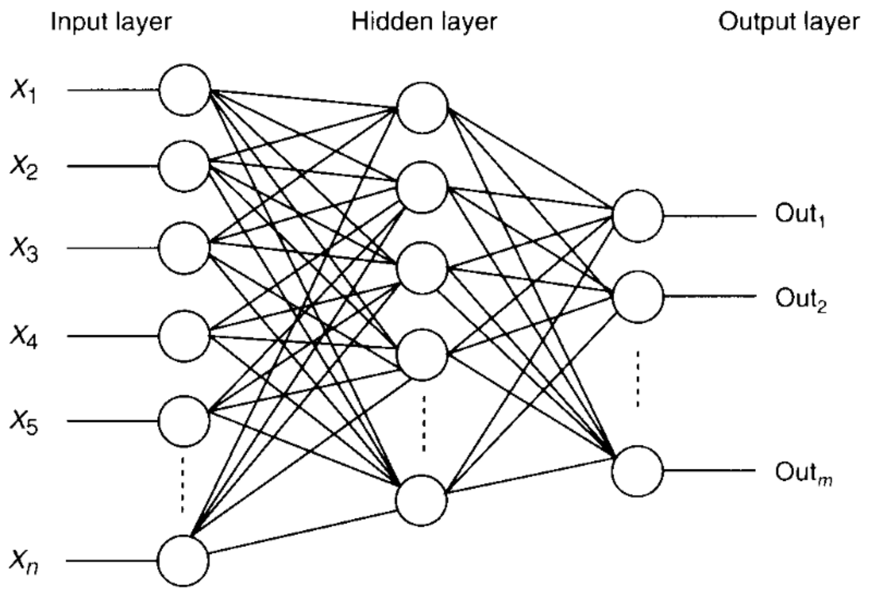
Compare this to another approach to modelling a complex system—GoogLeNet, Google's state of the art Machine Learning system that won first prize in the prestigious ILSVRC competition for image recognition in 2014. The two models at first appear similar. They both have nodes which process information, and wires that carry the information between the nodes. However, the way they work could not be more different. While the nodes and connections within the Limits to Growth model were directly designed by the researchers, Google's model is based on a Neural Network, an abstract computational system where the nodes represent arbitrary nonlinear functions, which are fully connected to a collection of other nodes also processing arbitrary nonlinear functions. The reason it works is that the model can be *trained* to adjust its functions and the weights (relative importance) of their interconnections *automatically*, based on data fed into the network. Thus, the model's human designers don't need to painstakingly specify every detail of a fixed model that will inevitably be superseded by reality—the model can *learn* and *adapt* on its own over time.



GoogLeNet's network architecture, 2014

This idea of learning and adaptation has been the key to most modern technologies driving advances in practical artificial intelligence. One of the most prominent of these technologies are Artificial Neural Networks, such as the one described above. Although the idea for such networks was first proposed by Warren McCulloch and Walter Pitts back in 1943, only very recent advances in computing power have allowed them to scale up to the level necessary to solve a variety of complex problems that were once thought to be beyond the abilities of computer systems.

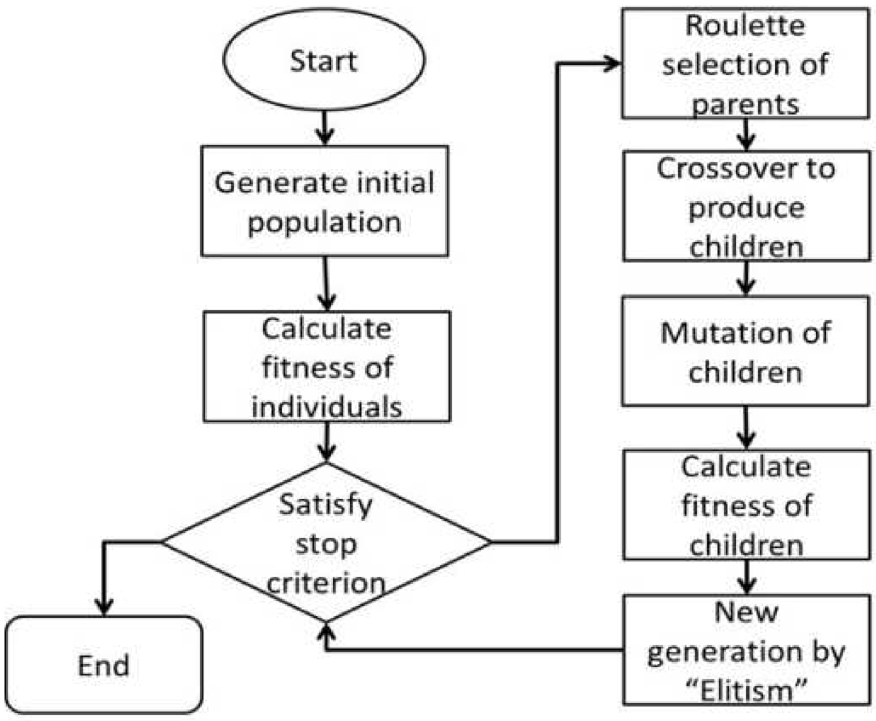
In generative design, however, we focus on a slightly different form of artificial intelligence—a class of algorithms technically called “metaheuristics”, but more often referred to simply as “search algorithms”. There is one key reason why search algorithms are more appropriate for solving design problems than neural networks. Under the hood, an ANN is composed of a huge number of simple calculations. The aggregation of these simple calculations in a dense network has the ability to create highly complex nonlinear functions which have been successful at modelling many real-world systems. However, the underlying computation needed to train these models is fairly straight forward, making their learning process more or less direct and deterministic. To put it more concretely, these networks are composed entirely of differentiable functions, meaning it is possible to compute their gradients, and deterministically follow the gradient’s path to the best solution.



Basic model for artificial neural network (ANN)

Most design models, however, are not composed of differentiable functions. Parametric models used in design practice are composed of a variety of modules that combine computation with geometric operations, none of which are easily differentiable. Such models can be thought of as 'black box' systems, which can not be directly trained to achieve an optimal solution. Instead, we need a separate system which can learn how to work with the model without a direct understanding of how it works.

This is where search algorithms come in. Rather than the learning system being embedded within the model, search algorithms use an external system that tests a given model to try to learn how it works from the outside in. You can think of this as a more 'top-down' strategy as opposed to the neural network's 'bottom-up' approach. The crucial advantage is that the search algorithm does not need to know anything about the internal workings of the model in order to figure out how to make it work. Because these algorithms rely on repetitive testing of a model rather than direct solving, they are referred to as 'stochastic' algorithms, which implies that at least a portion of their functionality can be attributed to random and unpredictable events.


Start
Generate initial population
Calculate fitness of individuals
Satisfy stop criterion
End
Roulette selection of parents
Crossover to produce children
Mutation of children
Calculate fitness of children
New generation by "Elitism"

Basic model for Genetic Algorithm (GA)

The specific type of search algorithm we will use in our generative design applications is the Genetic Algorithm (GA), which borrows concepts from Darwinian evolution such as selection, breeding, and mutation to iteratively search through the possibilities of a given design model and 'evolve' better performing designs. In the next few articles I will demonstrate why such tools may be useful additions to the traditional intuitive human design process.

As designers, we have much to learn from the way that form is designed in nature. But unfortunately, we cannot operate directly at the time scales allotted to natural evolution. Luckily, modern computation tools such as Genetic Algorithms provide a bridge, allowing us to enact similar mechanisms within our own design processes. In the best case, these artificially intelligent tools will augment our own human capabilities as designers—accentuating our strengths while making up for our weaknesses. By allowing us to more effectively think through the inherent complexities behind any design problem, these technologies have the potential to make us not only more efficient, but more thoughtful human designers.