Delving into Deep Learning

The latest neural networks learn to see and hear, and maybe even dream.

Brian Hayes

Endowing a computer with human perceptual skills, such as understanding spoken language or recognizing faces, has been on the agenda of computer science since the era of vacuum tubes and punch cards. For decades, progress was slow and successes were few. But now you can have a conversation with your cell phone about tomorrow's weather or last night's baseball scores. And Facebook and Google+ recognize faces well enough to suggest that you "tag" your friends in photos.

What accounts for these sudden breakthroughs in pattern recognition by machines? There is no single answer. A variety of algorithmic and statistical ideas have played a part, along with more powerful hardware. But one suite of techniques merits special mention. A scheme called *deep learning* has achieved impressive performance on several pattern-analysis tasks. Programs for deep learning are also known as *deep neural networks,* because they are constructed as multiple layers of processing elements analogous to the neurons of the nervous system.

The deep methods have a role in some well-known speech-recognition systems, such as Apple's Siri and the Google search-by-voice service. They are also leading the way in aspects of computer vision. Apart from perceptual tasks, deep learning is proving adept at data mining: extracting meaningful patterns from large data sets.

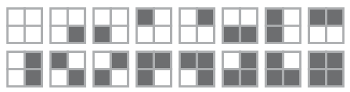
How do the deep programs work? Oddly enough, no one can answer this question in full detail. A distinctive feature of neural networks is that the designer or programmer does not directly specify all the particulars of a computation. Instead, the neural net is "trained" by exposure to thousands of examples, and it adjusts its internal parameters to maximize its own success. When the training is complete, we have a machine that can answer questions, but we don't necessarily know how it computes the answers. I find this situation mildly frustrating. On the other hand, it's a predicament I am familiar with at the most intimate level. I, too, understand speech and recognize faces—and I can't explain how I do it.

Brian Hayes is senior writer for American Scientist. *Additional material related to the* Computing Science *column can be found online at http://bit-player.org. E-mail: brian@bit-player.org*

A Network for Stripes

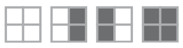
Before diving into deep neural networks, let's play in the shallow end of the pool, with a network that recognizes simple geometric patterns. The patterns come from an image sensor like the one in a digital camera, only simpler: It ignores color and shades of gray, merely reporting whether a pixel is light or dark. Signals from the pixels go to the input layer of the neural network. Each neuron in this layer receives signals from four pixels arranged in a 2×2 square; every such square patch in the sensor array reports to its own neuron. (The patches overlap.)

A 2×2 window of black or white pixels can display 16 different motifs:



The neurons of the input layer are programmed to recognize a specific subset of these patterns. If a neuron sees one of the designated motifs in its square patch, the neuron "fires," producing a positive output; otherwise it stays silent. Signals from the input layer are passed on to a single output neuron, which fires only if all the input neurons fire. Thus the output neuron reports a unanimous yea-or-nay judgment.

This 2×2, black-or-white, all-or-nothing device is just about the simplest conceivable neural network, and yet it can do some interesting things. Let the input neurons respond positively to these four motifs (and no others):



Such a network recognizes any pattern of vertical stripes, regardless of the stripes' width. It's a wallpaper sensor. Other subsets of the 16 motifs yield networks triggered by horizontal stripes or by diagonals. The ability to detect stripes in various orientations is intriguing in that the primary visual cortex of the mammalian brain is full of stripe-sensitive neurons.

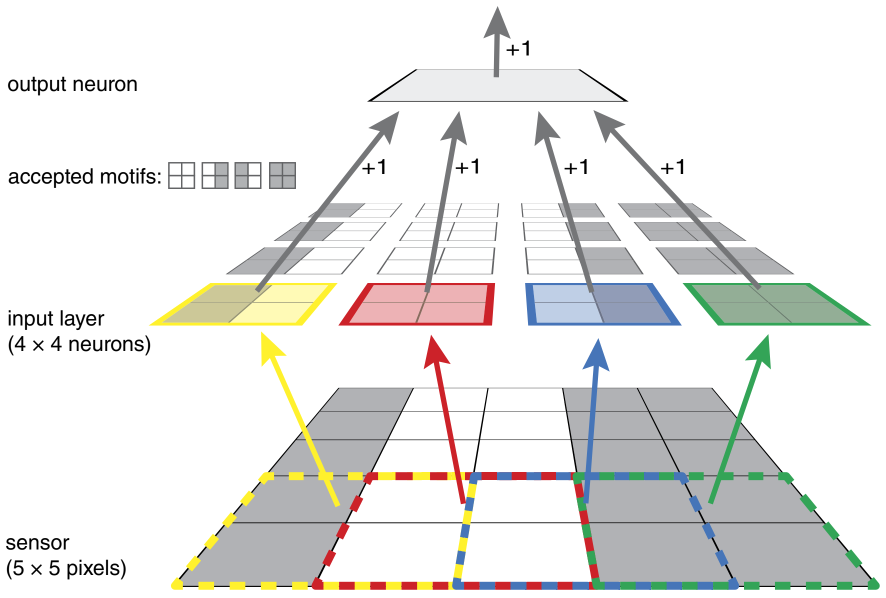
This rudimentary neural network can also be programmed to detect patterns other than stripes. You might try to find a set of motifs that picks out rectangles. But there are also tasks the system *cannot* perform. For example, no network of this kind can distinguish squares from other rectangles.

The Learning Algorithm

The automaton we have just assembled is not a learning machine. The patterns it detects are predetermined, like the hard-wired behaviors of a primitive organism. Learning requires some form of corrective feedback.

Suppose you want to build a stripe detector not by hand-picking the motifs to accept but by letting the system learn them. You project a series of images onto the sensor, some of which should be recognized as vertical-stripe patterns and some not. If the network gives the correct answer for an image, do nothing. If the system makes the wrong choice, there must be at least one neuron in the input layer that responded incorrectly, either accepting a motif it should have rejected, or vice versa. Find all such errant neurons, and instruct them to reverse their classification of the current motif.

With this feedback mechanism, we have a machine that improves with practice—but we are still a long ways from a device that can learn to recognize human faces. Given this network architecture—an input layer and a single output neuron—the repertory of recognized patterns can never extend beyond rather simple geometric figures. Moreover, the system's all-or-nothing logic makes it brittle and inflexible. A single stray pixel can alter the machine's verdict, and it can be taught to recognize just one set of patterns at a time. More useful would be a *classifier* that could look at a variety of patterns and assign them to groups.



A rudimentary neural network called a *perceptron* recognizes oriented stripes in an image. The sensor for this example is a 5 × 5 array of pixels. Each 2 × 2 subarray communicates with a neuron in the input layer; four of these overlapping 2 × 2 squares are distinguished by color. An input neuron responds positively (with a +1 output) if and only if the underlying pixel array matches one of four motifs. The output neuron fires only if the input neurons are unanimous.

Going Deeper

Our little neural network can be augmented in several ways. To begin with, the simple yes-or-no choices can be extended to a continuous range of values. Signals passing from one neuron to another are multiplied by a coefficient, called a weight, with a value between –1 and +1. The receiving neuron adds up all the weighted inputs, and calculates an output based on the sum. Signals with a positive weight are excitatory; those with a negative weight are inhibitory. Heavily weighted signals (whether positive or negative) count for more than those with weights near zero. Learning becomes a matter of adjusting the weights in response to corrective feedback.

The geometric scope of the input neurons can also be enlarged. Each neuron might collect inputs from a larger patch, or the inputs might come from regions scattered across the sensor. In the limiting case, every input neuron receives signals from every sensor element.

Finally, the two-layer architecture of the network can be expanded. Inserting intermediate layers of neurons—known as hidden layers because they don't directly communicate with the outside world—lifts many restrictions on the computational capabilities of neural networks. Indeed, this is what makes the networks "deep."

Deep networks are more versatile and potentially more powerful. They are also more complex and computationally demanding. It's not just that there are more neurons and more connections between them. The big challenge is organizing the learning process. Suppose a certain hidden-layer neuron has sent the wrong signal to the output layer, causing the system to mix up Elvis Presley and Elmer Fudd. You might "punish" that behavior by decreasing the weight of the Elvis connection. But maybe the error should really be attributed to the input neurons that feed information to the hidden neuron. It's not obvious how to apportion blame in this situation.

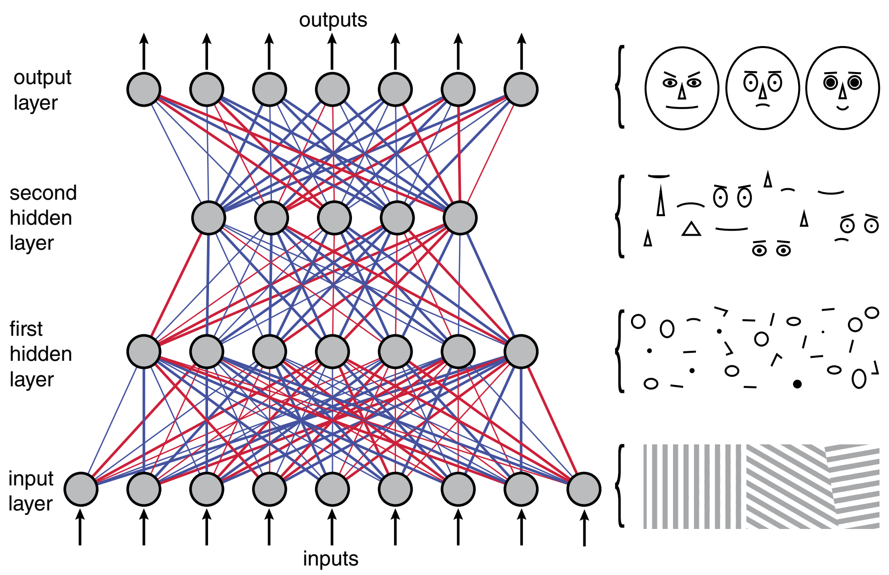
From Perceptrons to Connectionism

Artificial neural networks have had a roller coaster history. In the 1950s Frank Rosenblatt described a class of devices he called *perceptrons*. To show how they work, he built an electromechanical contraption with 400 photocells as sensors and motor-driven potentiometers to adjust the weights. The stripe-detecting network described above is a particularly simple instance of a perceptron. (This specific model was introduced in 1984 by A. K. Dewdney.)

In 1969 Marvin Minsky and Seymour Papert published a critique of two-layer perceptrons, giving mathematical proofs of their limitations. For example, they showed that no network without hidden layers can distinguish connected geometric figures from those made up of two or more disconnected pieces. Beyond the proofs, Minsky and Papert offered a harsh assessment of the entire neural network field, remarking that much writing on the subject was "without scientific value."

The Minsky-Papert impossibility proofs applied only to networks without hidden layers. Rosenblatt and others were already experimenting with multilayer devices, but they had trouble finding efficient learning rules. In the aftermath of these setbacks, neural network research languished for a decade. And there was a further misfortune: Rosenblatt died in a boating accident in 1971, on his 43rd birthday.

Interest in neural networks revived in the 1980s under the new brand name of *connectionism.* A key event was the discovery of an algorithm known as back-propagation, which allowed efficient training of neural networks with three layers: an input layer, an output layer, and a single hidden layer. The technique was first formulated by Paul J. Werbos and was popularized by David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams, who demonstrated its successful application in 1986.



Multilayer neural networks can learn to recognize features at several levels of abstraction, from geometric motifs such as oriented stripes, through simple shapes, to complex objects such as human faces. Connections between layers of neurons can be either excitatory or inhibitory and also differ in their "weight," or influence on the receiving neuron. In this schematic diagram, the properties of the connections are encoded in the color and thickness of lines.

As the name suggests, back-propagation reverses the flow of information through the network. Error-correcting signals travel back from the output layer to the hidden layer, then continue on to the input layer. Within each layer, the corrective adjustment is determined by the principle of steepest descent: Incorrect weights are nudged in the direction that causes the greatest change in the output. The process is not guaranteed to find the best possible assignment of weights—it can get trapped in a local optimum—but experiments suggested that densely connected networks seldom succumb to this hazard.

In principle, back-propagation can be applied to networks of any depth, but with multiple hidden layers the procedure tends to bog down. There is also the risk of "overfitting," where the network learns the training cases too well, responding to irrelevant details that are not present outside the training set.

The neural network roller coaster did not plunge steeply after climbing to the connectionist peak in the 1980s. Nevertheless, 20 years passed before the current frenzy over deep learning began.

Multilayer Networks

The new deep networks are not just deeper; they are larger in all dimensions, with more neurons, more layers, and more connections. A project called Google Brain, begun in 2011, had a million neurons and a billion connections. (After perusing 10 million images from YouTube, Google Brain concluded that the Internet is full of cats.)

A prerequisite for bigger networks is more hardware. Google Brain ran on a cluster of 16,000 computers. But hardware alone is not enough to tame the complexities of training multilayer networks. Another contribution is the concept of *pretraining*, which addresses the problem of how to set the initial weights in a network. If you start by making all the weights very small, it takes forever to reach a state where anything interesting happens. On the other hand, large initial weights raise the likelihood of getting trapped prematurely in a local optimum. Pretraining primes the system to follow a more fruitful path.

Among all the ideas that animate the deep learning movement, the one I find most evocative comes from Hinton. He suggests that the networks must not only perceive and reason but also sleep and dream. The dreaming allows the system to augment its own training set.

Underlying this metaphor is the idea that the layers of a neural network represent information at progressively higher levels of abstraction. In face recognition the bottom level holds the raw input data—an array of pixels. The lower neural layers capture simple, local features of the image, such as oriented edges or corners. Activity in the higher levels represents larger and more complex features. At some point we encounter eyes and noses, and establish spatial relations between them. At the top is the concept of the face itself. In the artificial network as in the human mind, something suddenly clicks and the identification tumbles out: Aunt Em.

In people, this process also works in reverse. The mere thought of Aunt Em conjures up a vision of her face. Hinton devised a mechanism by which neural networks could also have visions and fantasies. All it requires is making the connections between layers bidirectional. In the conventional phase of the training process, information moves from bottom to top, assembling higher-level abstractions out of the bits and pieces found in the lower layers. In dreaming, the higher-level representations are projected downward through the layers, creating lower-level realizations of each concept. Connection weights are interpreted as probabilities to guide the process. At the bottom of the stack is an imaginary portrait. Generating such faux images contributes to learning in the same way that analyzing real images does. Hinton refers to the two-phase training regime as the sleep-wake cycle.

A perennial technology of tomorrow, neural networks have arrived in the here and now.

Deep Networks and Social Networks

After decades as one of those perennial technologies of tomorrow, neural networks have suddenly arrived in the here and now. Speech recognition was the first area where they attracted notice. The networks are applied to the acoustic part of speech processing, where a continuous sound wave is dissected into a sequence of discrete phonemes. (The phonemes are assembled into words and sentences by another software module, which is not based on neural networks.) In 2009 Hinton and his student George Dahl set an accuracy record for the transcription of a standard corpus of recorded speech.

A current focus is object recognition in still and video images. The computer vision community holds an annual contest for this task, asking contestants to classify about a million images in 1,000 categories. In 2012 Hinton and two colleagues entered the contest with an eight-layer neural net having 650,000 neurons and 60 million adjustable parameters. They won with an error rate of about 15 percent; the runner up scored 26 percent.

These successes have attracted attention outside the academic world. As noted, speech recognition networks are already at work in voice-input devices and services. Google has adapted the object recognition techniques for image searches. A number of data-mining tools for tasks such as recommending products to customers are built on deep networks. There will doubtless be more such applications in the near future. Several of the senior research figures in deep learning (including Hinton) are working with Google, Facebook, and other companies ready to make large investments in the technology.

Neural Nets and Neurology

These triumphs of neural networks might seem to be the definitive answer to the Minsky-Papert critique of perceptrons. Yet some of the questions raised 50 years ago have not gone away.

The foundation of the neural network methods is almost entirely empirical; there's not much deep theory to direct deep learning. At best we have heuristic guidelines for choosing the number of layers, the number of neurons, the initial weights, the learning protocol. The impressive series of contests won by Hinton and his colleagues testifies to the effectiveness of their methods, but it also suggests that newcomers may have a hard time mastering those methods.

An immense space of network architectures remains to be explored, with a multitude of variations in topology, circuitry, and learning rules. Trial and error is not a promising tactic for finding the best of those alternatives.

Or is it? Trial and error certainly had a major role in building the most successful of all neural networks—those in our heads. And the long dialogue between biological and engineered approaches has been fascinating if not always fruitful. The biological model suggests ways to build better connectionist computers; the successes and failures of computational models inform our efforts to understand the brain.

In both of these projects, we have a ways to go. A machine that learns to distinguish cows from camels and cats from canines is truly a marvel. Yet any toddler can do the same without a training set of a million images.

Bibliography

Dewdney, A. K. 1984. Computer recreations. *Scientific American* 251:22–34.

Hinton, G. 2006. To recognize shapes, first learn to generate images. https://www.cs.toronto.edu/~hinton/absps/montrealTR.pdf

Hinton, G. E., and R. R. Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *Science* 313:504–507.

Hinton, G. E. 2007. Learning multiple layers of representation. *Trends in Cognitive Sciences* 11:428–434.

Minsky, M. L., and S. A. Papert. 1969. *Perceptrons: An Introduction to Computational Geometry*. Cambridge, MA: MIT Press.

Rosenblatt, F. 1962. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Washington, D.C.: Spartan Books.

Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams. 1986. Learning representations by back-propagating errors. *Nature* 323:533–536.