An Introduction to Neural Networks

by Carola F Berger, PhD, CT
cberger@CFBtranslations.com

Neurons and Units

In the following, unless explicitly stated otherwise, the term “neural network” refers to artificial neural networks (ANNs), as opposed to biological neural networks.

ANNs are not a new idea. The idea has been floating around since the 1940s, when researchers first attempted to create artificial models of the human brain. However, back then, computers were the size of entire rooms and consisted of fragile vacuum tubes. Only in the last decade or so have computers become small and powerful enough to put these ideas into practice.

Like biological brains, which are composed of neurons, ANNs are composed of individual artificial neurons, called units. Their function is similar to biological neurons, as shown in the figures below. Fig. 1 shows a biological neuron, whose precise function is very complicated. Loosely speaking, the neuron consists of a cell body, dendrites, and an axon. The neuron receives input signals via the dendrites. When these input signals reach a certain threshold, an electrochemical process takes place in the nucleus, and the neuron transmits an output signal via the axon.

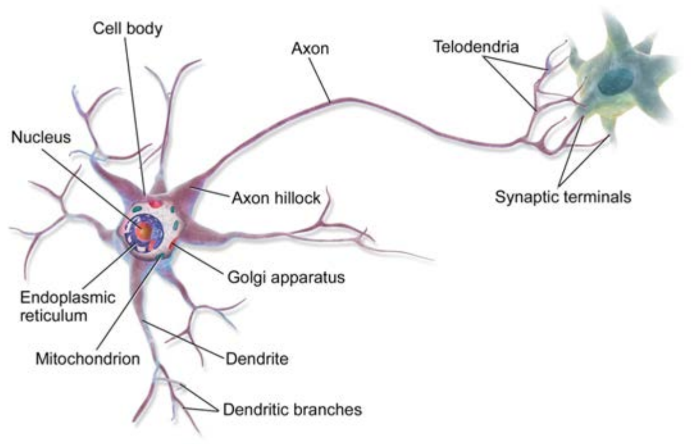


Fig. 1: Biological neuron. Source: Bruce Blaus, https://commons.wikimedia.org/wiki/File:Blausen_0657_MultipolarNeuron.png

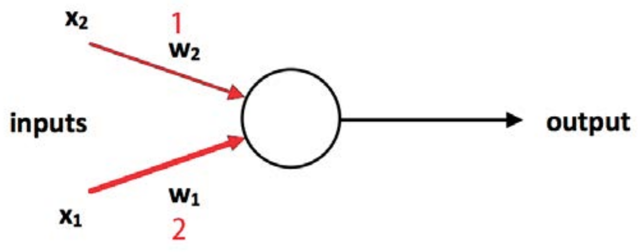


Fig. 2: Unit in artificial neural net

Fig. 2 shows a model of a very simple artificial unit. It also receives inputs, in this case two inputs (labeled x_1 and x_2). An activation function (the white blob in Fig. 2) transmits an output signal that depends on the inputs. The activation function can be a simple threshold function. This means that the unit is turned off until the sum of the input signals reaches a certain threshold, and when the sum of the inputs exceeds the threshold, it transmits an on signal. However, the activation function can also be much more complicated. As described, this artificial unit does not perform any particularly interesting functions. Slightly more interesting functions can be achieved by weighting the inputs differently according to their importance. An artificial neural net "learns" by adjusting the weights (labeled w_1 and w_2 in Fig. 2), or equivalently the importance, of the input signals into each unit according to some automated algorithm. In Fig. 2, input x_1 is twice as important as input x_2, as illustrated by the relative thickness of the input arrows.

Layers and Networks

Nevertheless, despite the weights, one unit by itself does not perform particularly complex functions. However, similar to biological brains, these units are assembled into a neural network, as shown in Fig. 3. More precisely, the figure shows a so-called feed-forward neural net. Once assembled into a neural net, the units can perform remarkable tasks.

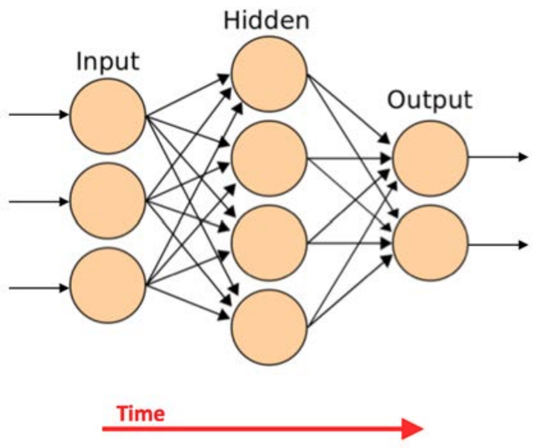


Fig. 3: Artificial neural network. Adapted from: Cburnett, https://commons.wikimedia.org/wiki/File:Artificial_neural_network.svg

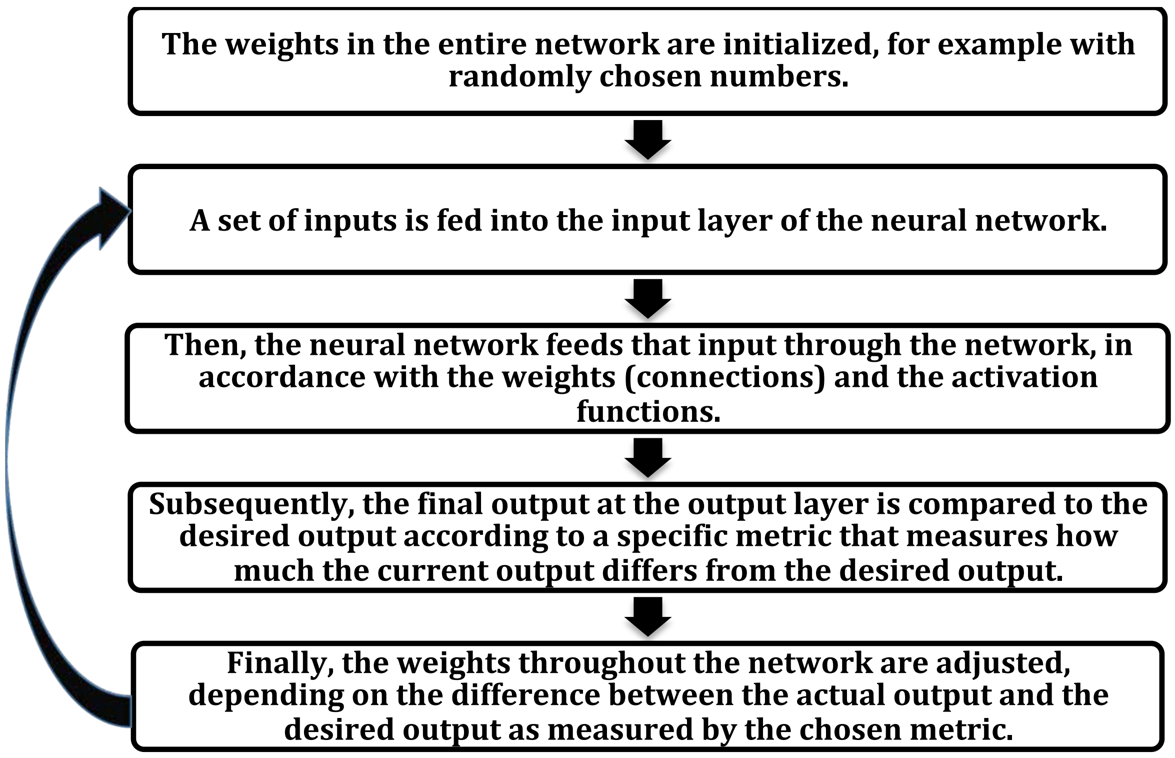
ANNs consist generally of an input layer, one or more hidden layers, and an output layer. Each layer consists of one or more of the units described above. The input layer processes the input from the external world such that it can be fed further into the ANN. Similarly, the output layer processes the output of the ANN for further consumption by the external world. The hidden layer(s) perform(s) the actual processing. Neural networks with more than one hidden layer are called "deep" neural nets. Each unit is connected with one or more other units (indicated by the arrows in Fig. 3), and each connection is assigned more or less importance through an associated weight.

In a feed-forward neural net, as shown in Fig. 3, a unit in a specific layer is only connected to units in the next layer, not with units within the same layer or in a previous layer, whereby the terms "next" and "previous" refer to sequences in time. In Fig. 3, the arrow of time flows from left to right. There are also so-called recurrent and convolutional neural networks, where the connections are more complicated. However, the main idea is the same. The middle layer in Fig. 3 is hidden, because it does not have direct connections to inputs or outputs, whereas the input and output layers communicate directly with the external world.

Training and Learning

The thus assembled neural network "learns" by adjusting the various weights, which are, for example, numbers between -1.0 and +1.0, but other values are of course possible. The weights are adjusted according to a specific training algorithm.

The training of a neural network typically proceeds as shown in the following flow chart on the next page:


The weights in the entire network are initialized, for example with randomly chosen numbers.
A set of inputs is fed into the input layer of the neural network.
Then, the neural network feeds that input through the network, in accordance with the weights (connections) and the activation functions.
Subsequently, the final output at the output layer is compared to the desired output according to a specific metric that measures how much the current output differs from the desired output.
Finally, the weights throughout the network are adjusted, depending on the difference between the actual output and the desired output as measured by the chosen metric.

The entire process is repeated, usually many thousands or millions of times, until the output is satisfactory. There are many possible algorithms to adjust the weights, but a description of these algorithms goes beyond the scope of this article.

An Example

As a concrete example, let's look at a fairly simple neural network that recognizes handwritten digits. A sample of the inputs is shown in Fig. 4. I programmed this simple feed-forward neural net for Andrew Ng's excellent introductory course on Machine Learning[1], which I highly recommend.

[1] https://www.coursera.org/learn/machine-learning

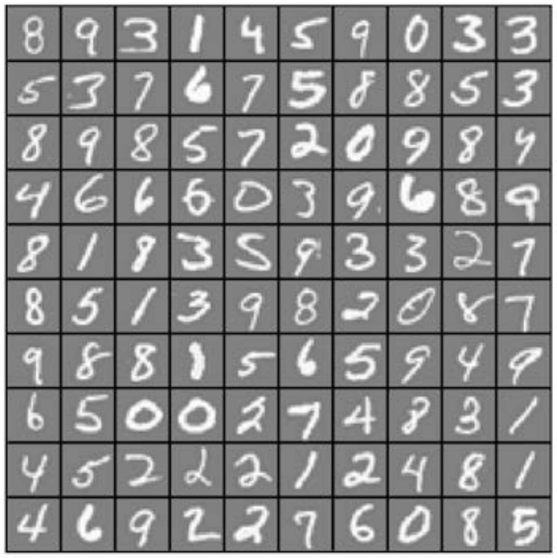


Fig. 4: Sample handwritten digits

The architecture of the neural network is exactly as shown in Fig. 3, with 400 input units, since the input picture files have a size of 20 x 20 grayscale pixels (=400 pixels). There are 25 units in the hidden layer, and 10 output units, one for each digit from 0 to 9. This means that there are 10,000 connections (weights) between the input layer and the hidden layer (400 x 25) and 250 connections between the hidden layer and the output layer (25 x 10). In other words, we have 10,250 total parameters! For the technically interested, the activation function is here a simple sigmoid.

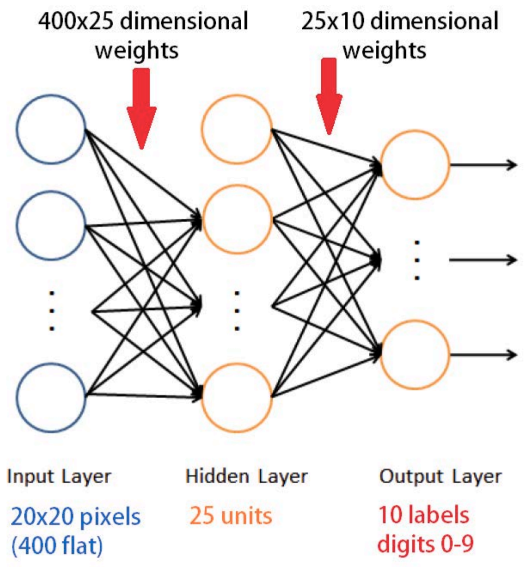


Fig. 5: ANN for handwritten digit recognition

The training proceeded exactly as described above. I fed in batches of several thousand 20×20 labeled grayscale images as shown in Fig. 4 and trained the net by an algorithm called backpropagation, which adjusted the weights according to how far the output was from the desired label from 0 to 9. The result was remarkable, especially considering that there were only a couple dozen lines of code.

But How Does It Work?

The fact that it works is remarkable and also a somewhat unsettling, because all I did was program the activation function, specify how many units are in each layer and how the layers are connected, specify the metric and the backpropagation, and the neural net did all the rest. So, how does this really work? Let's "dissect" the above neural net layer by layer, and pixel by pixel. Here's what is actually happening to the input, *after training the neural net successfully*, that is after processing the weights and thousands of inputs according to the flowchart on page 4.

Autopsy of a neural network

The first set of weights between the input layer and the hidden layer can be thought of as a set of filters, to essentially recognize and filter out important patterns or features. If one plots only this first set of weights, one can visualize a set of 25 "filters," as shown in Fig. 6. These filters map the input onto the 25 hidden units in the hidden layer. Fig. 7 shows what happens if you map a specific input, in this case a handwritten "0," onto the hidden layer.

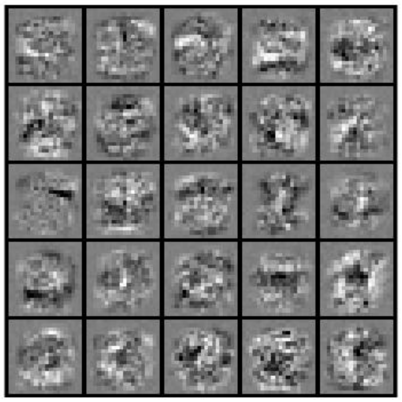


Fig. 6: First set of weights, acting as a "filter."

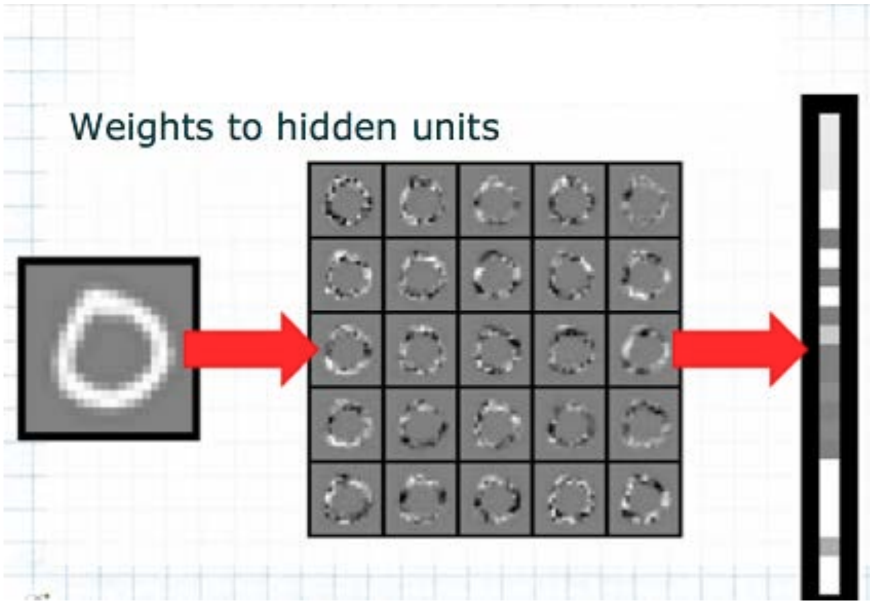


Fig. 7: Mapping of 0 to hidden units.

The output of the hidden layer is then piped through another filter, as shown in Figure 8, and mapped onto the final output layer via this filter/set of weights. Fig. 8 shows how the input picture with the digit "0" is correctly mapped onto the output unit for the digit "0" (at the bottom, because the program displays things vertically from 1 at the top to 9 and then to 0 at the bottom).

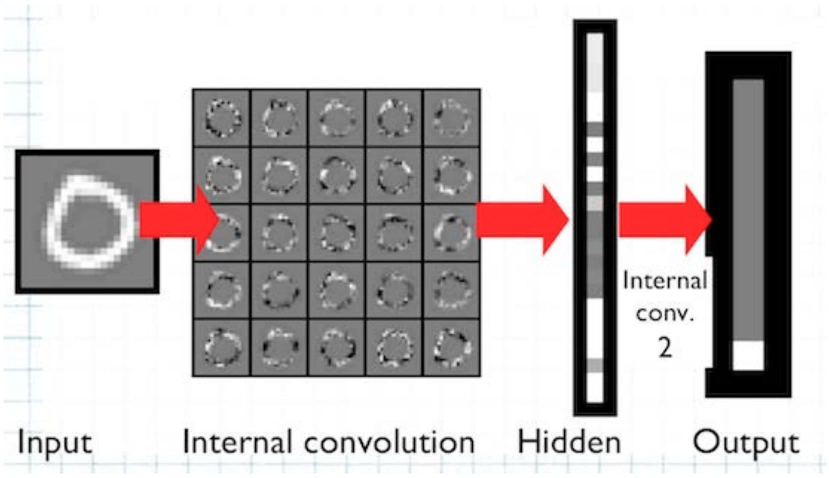


Fig. 8: Mapping of input, here a "0", to output via hidden layer

More examples of this filtering or mapping via the internal sets of weights are visualized in my slide set for ATA58[2] and also in Fig. 9.

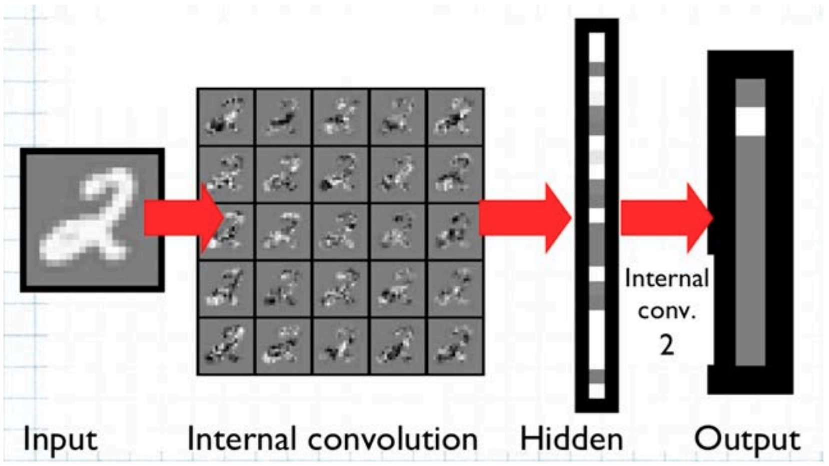


Fig. 9: Mapping of input "2" to output

Again, the internal weights act as a sort of filter to pick out the features of interest. Naively, I would have expected that these features or patterns of interest correspond to vertical and horizontal lines, for example for the digits 1, 4, or 7, or to various arcs and circles, for digits like 3 or 8 or 0. However, this is evidently not at all how the network picks out digits, as can be seen from the visualization of the first set of weights in Fig. 6. The patterns and structures that the neural net filters out are ostensibly much more complex than simple lines or arcs. *The intricate structure of these weights is necessary for the neural net to be able to recognize all 10 different digits with only one set of weights that are fixed once and for all after the original training.* This is also the reason for the "detour" via the hidden layer. A direct mapping from input to output, even with an internal convolution, would not be sufficient to pick out all the information that is

[2] https://www.cfbtranslations.com/wp-content/uploads/2015/03/AI_Berger_ST-8.pdf

necessary to distinguish one character from another. Similarly, for more complex tasks, more than one hidden layer will be needed. The number of hidden layers and units as well as their connections/weights grows with the complexity of the task.

Summary

This article explains the inner workings of a simple neural net by visualizing the internal process. Neural networks for other applications, including for machine translation, work pretty much the same way. Of course, most of these will have more than one hidden layer, possibly some pre- and post-processing of input and output data, more sophisticated activation functions, and a more complicated architecture, like recurrent neural nets or convolutional neural nets. However, the basic idea remains the same: ***The underlying function of an artificial neural net is pattern recognition.*** Not more, not less. While well-trained ANNs are extraordinary and unquestionably better than humans at the pattern recognition tasks they are trained for, because they don't get tired or have lapses of concentration, one should never forget that they are remarkably ill-suited for anything that goes beyond the tasks they are trained for. In such cases they can sometimes detect patterns that are not there, and sometimes the tasks simply cannot be cast into a pattern, however complicated. In other words, while ANNs certainly exceed their programming, they can never exceed their training. (At least until the so-called technological singularity[3] is upon us.)

[3] https://en.wikipedia.org/wiki/Technological_singularity