By Paul Naidoo, EOLAS

Deep Neural Networks

A **Neural Network (NN)** is a computational model inspired by the human brain, consisting of interconnected nodes ("neurons") organized in layers. It processes input data through these layers, using weights and activation functions to transform the data and produce an output. Neural networks are used for tasks like classification, regression, and pattern recognition.

A **Deep Neural Network (DNN)** is an NN that has multiple layers ("hidden layers") between the input and output layers, allowing it to learn and represent complex hierarchical features in the data.

Building a simple neural network

The layers in an NN are represent linear relationships between data variables ("features"). Consider a simple linear model, as illustrated in Figure 1. This can, of course, be modelled linearly with the formula: $y = mx + c$; or in the NN-vernacular: $F(x) = wx + b$, where w is weight, and b is bias. Figure 2 illustrates this equation as the building block of an NN.

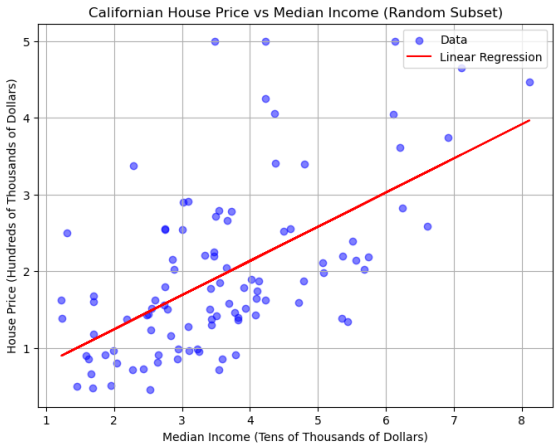

Figure 1: A simple linear relationship

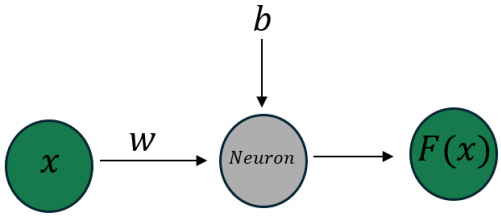

Figure 2: Illustration of a single neuron representing the relationship $F(x) = wx + b$

Adding complexity, what if we want to model a function of three variables? This would be described by the equation: $F(x) = w_1x_1 + w_2x_2 + w_3x_3 + b$, illustrated in Figure 3.

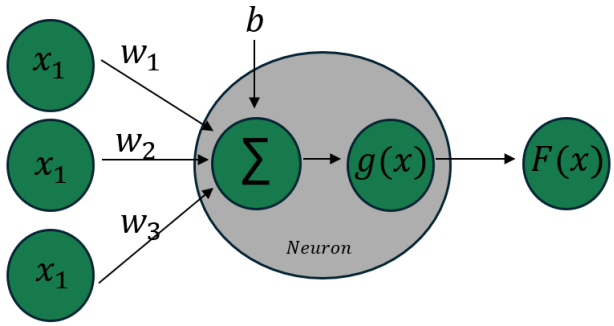

Figure 3: Illustration of the equation $F(x) = w_1x_1 + w_2x_2 + w_3x_3 + b$, with the activation function $g(x)$ also depicted.

Figure 3 also shows the presence of the "activation function" within the neuron. Not only are the products of the weights and variables summed with the bias, but an activation function, $g(x)$, is also what then "simulates" the activation of a neuron in the brain, where a sufficiently large signal would lead to activation in the connected neurons. Figure 4 shows various functions frequently used as activation functions. Note that these vary from the identity function, which simply passes forward the value unperturbed, to non-linear functions which empower these linear equations to model non-linear relationships.

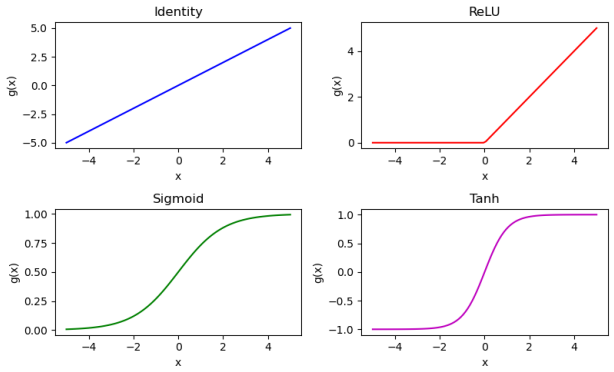

Figure 4: Various activation functions which can be used (among others) in a Neural Network.

The final step in building our neural network is to include more than one neuron. A layer can have any number of neurons, which would all be connected to our input later. The combination of all of these would then be connected to the output layer, or indeed another hidden layer. Figure 5 illustrates an NN with one layer, containing four neurons. Note that each connecting line between variable and neuron represents a weight.

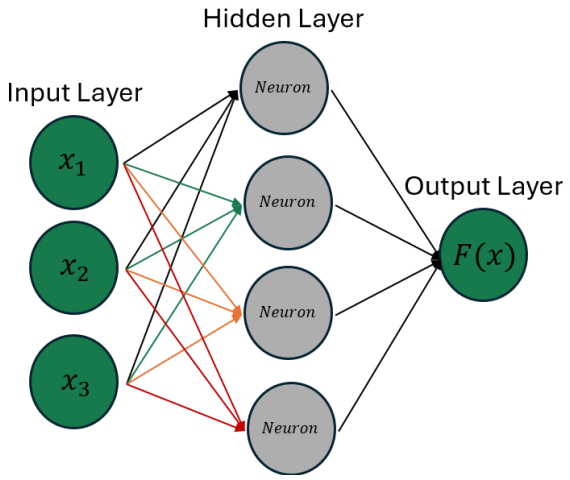

Figure 5: A Neural Net containing one hidden layer, with four neurons.

DNNs: Rapidly Increasing Complexity

We have demonstrated how a simple neural network can be constructed. However, in practice the most performant models are Deep Neural Networks (DNNs), an example of which is illustrated in Figure 6. With DNNs, the sky is the limit when it comes to complexity. A DNN can have several hidden layers. Each layer can have an arbitrary number of neurons and can invoke different activation functions.

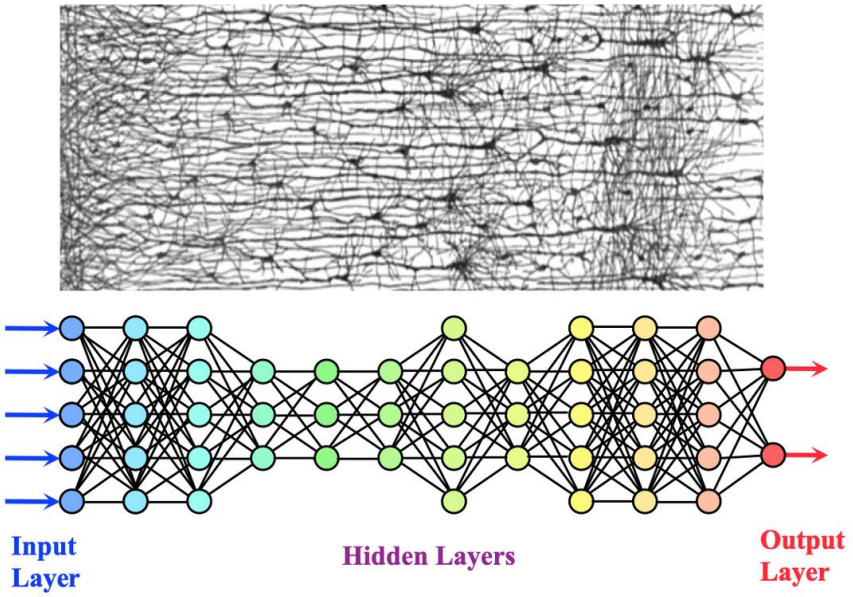

Figure 6: Depiction of how a Deep Neural Network replicates the interconnectivity of neurons in the brain.

The complexity and scale of these models is what leads to the “black box” phenomenon, where a model can represent a function of hundreds or even thousands of variables and weights, meaning that interpreting what is causing the model to have made one prediction or another is seemingly impossible to glean. Conversely, this complexity is what allows these models to isolate subtle, multi-variate signal in data that empower them to perform analyses and predictions that otherwise would not be possible with a “classical” analysis.

How DNNs Work

DNNs are “trained” by iteratively ingesting data, and tuning the weights and biases via various statistical methods. A *very* high-level view of some of these are given below.

- **Forward Propagation**: Input data is fed into the network, passing through each layer. Neurons in each layer process the data using weights, biases, and activation functions, transforming it step-by-step until it reaches the output layer.
- **Loss Function**: The output is compared to the actual target using a loss function, which measures the error. Common loss functions include mean squared error for regression and cross-entropy for classification.
- **Backpropagation**: To minimize the error, DNNs use backpropagation. This involves calculating gradients of the loss function with respect to weights and biases, and updating them using optimization algorithms like gradient descent.

Advantages of DNNs

- **Complex Pattern Recognition**: DNNs excel at recognizing complex patterns and relationships in data, making them suitable for tasks like image and speech recognition.
- **Feature Learning**: They automatically learn hierarchical features from raw data, reducing the need for manual feature extraction.

Limitations

- **Computationally Intensive**: Training DNNs requires significant computational resources and time, especially with large datasets.
- **Overfitting**: DNNs can overfit to training data, especially if the model is too complex or the dataset is small. Techniques like regularization and dropout are used to mitigate this.