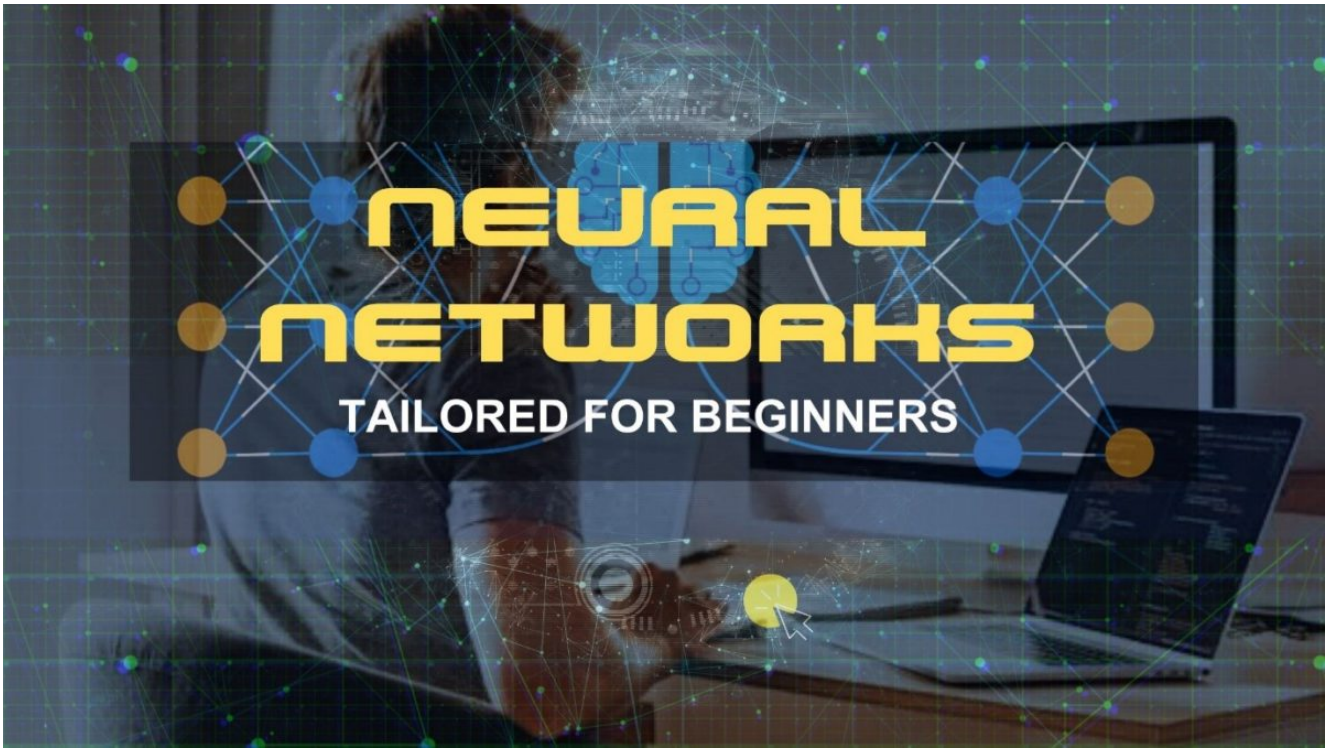
A Beginner’s Guide to Neural Networks



Machine Learning (ML) has moved past the domain of academia and research to become part of our daily lives. Well, almost. Self-driving cars, speech recognition, product recommendation, healthcare, and financial trading systems are just a few of the areas impacted by ML.

Beneath machine learning engines are neural networks, which provide all the brainpower to these powerful systems. This article is a first look at what neural networks are and how to understand their working in a much simpler way.

Let’s dive right in:

Anatomy of Neural Networks

To clarify, the machine learning system is the entire network that takes an input and provides an output. Within this system, there are artificial ‘neurons’ that perform functions such as weight adjustment, decision-making, and

transformation. Numerous neurons might contribute to one or more collective outputs.

This is a topic with a significant learning curve. If you're a college student working on a research paper on neural networks, expert writers like UK essay writing service can assist you. How? By providing the topic, you can get your research, citations, or even a draft crafted for you. That will reduce your working time.

Back to neural networks. These networks are composed of a series of layers and weights, each serving a distinct purpose. The input layer receives the primary data and usually, no computation happens here, just data transformation. The hidden layers perform computation using specific algorithms. The output layer is the final layer that represents the final prediction of the neural network.

The following example will help us understand neural networks better.

Training a Neural Network for Ad Marketing

Let's imagine we're training a neural network for an ML ad model. The model that can be deployed in digital marketing networks predicts whether a lead will click on an ad or not based on their age and income.

Let's see what this network would look like:

1. Input layer: Two pieces of info are fed to the network, the customer's age, e.g., 25, and income, e.g., $50,000. Think of each piece as a separate neuron in the input layer.

2. Hidden layer: A neural network can contain multiple hidden layers. For this example, we'll only assume one hidden layer. Each neuron in this layer receives a

weighted combination of the age and income values from the inputs. Think of the weights as “importance” factors. Thus, a younger person with a high income may be more likely to click an ad than an older person with the same income.

3. Output layer: Has a single neuron that receives the output from the hidden layer and makes the final prediction. That output might be a binary of 0 or 1, with o representing a “no click” and 1 representing a “click”.

4. Learning: Why it’s called a ‘neural learning network’ is because the network learns from its mistakes and corrects them. It does this by adjusting the weight of the connections between the neurons. The more data the network gets, the more fine-tuning it does and adjusts its weights. Thus, it gets better at predicting whether the customer will click on the ad or not.

Neural Network Diagram:

Input Layer (2 Neurons)	**Hidden Layer (1 Neuron)**	**Output Layer (1 Neuron)**
[Age: 25] ——— [Income: $50,000] ——-	[Weight, w1] [Weight, w2] **(Activation Function)** ——	**Prediction**

Thus, the neural network takes in information, transforms it through weighted connections, and makes predictions based on the processed data.

Weights and Biases

Weights control the strength of the connections between nodes, therefore how much influence the input will have on the

output.

In neural networks, biases are correction factors that are usually assigned a random value at the start of the learning process. They ensure that even when there is a zero input the neuron will still be activated.

While bias units are not influenced by previous connections, the bias influences the weight in the outgoing connection.

Activation Functions, Feedforward and Backpropagation

Activation functions determine whether a neuron should be activated or not. The activation function does this by transforming the summed weighted input from the node into the output value to be fed into the next layer.

Thus, we can summarize the process of neural learning as looking like:

- Taking input and multiplying by the neuron's weight.
- Adding bias.
- Feeding the result, x, to the activation function.
- Taking the output and transmitting it to the next layer of neurons.

This type of movement is referred to as **feedforward propagation**, where the information flows going forward. Here the activation function acts as the "determiner" of what goes to the next neuron.

In backward propagation, the weights of the neural connections are repeatedly adjusted depending on different factors. This provides non-linearity to the neural network, without which the neural network would just be one giant linear regression model.

Examples of activation functions include:

- Linear activation functions – Also known as “no identity” functions. The function doesn’t transform the weighted sum of the input.
- Binary Step functions – These depend on threshold values to decide whether neurons should be activated or not. If lower than a certain threshold the neuron isn’t activated, and if it passes the threshold, it is activated.

Non-Linear Activation Functions

If you’re a college student performing research on this complex subject, you’d do well to pay for essay writing services. You can learn more about this subject, especially on the types of non-linear functions which won’t be covered in this article. Examples of non-linear activation functions are shown above, courtesy of v7 labs.

These allow the model to create complex mappings between the inputs and outputs. A derivative function is related to the input and the model can thus go back to find better predictors. This is the concept of backpropagation, where error information is transmitted through the network, thus facilitating weight adjustment.

Examples of non-linear functions include the Sigmoid/Logistic Activation, Tanh, the ReLU (Rectified Linear Unit), and the Softmax functions.

Conclusion

Machine learning running on neural networks is bound to change every facet of human society. Neural networks are capable of learning, adapting, and making predictions across diverse domains.

As a learner in the field of computer science, this article

represents a first step toward building a foundation for the subject. From theoretical constructs to real-world problems, take that monumental step towards a rewarding career in machine learning. Good luck!