Deep Learning – the heart of the current wave of Artificial Intelligence

By Lalit Pant

Artificial intelligence. That's a word that has been gaining prominence (and hype) over the past few years. We keep hearing of a world in the not too distant future where personal virtual assitants will help us with our every desire, self-driving cars will drive us around, and smart cities will provide environments that will adjust to our needs. This will be a world where most of the manufacturing and services driving the economy will be run by robots and Artificial Intelligence (AI) agents. This upcoming world is sometimes painted as a utopia, and at other times looked at with fear related to how the machines might take over.

Given this context, it is important for us to see exactly what AI is, and what it is made of. It is important for us to try to dig beneath the hype and find out what lies underneath. Armed with this knowledge, we will be better prepared to make sense of the flood of AI related information that will inevitably come our way, and to make sensible choices around AI.

So what exactly is Artificial Intelligence?

First, let's look at a simple definition of intelligence:
Intelligence is the ability to acquire and apply knowledge in useful ways.

Artificial Intelligence is the automation of the above. It's the effort to capture in hardware and software the ability – for machines – to acquire knowledge and to apply it in useful ways, in ways that a human would.

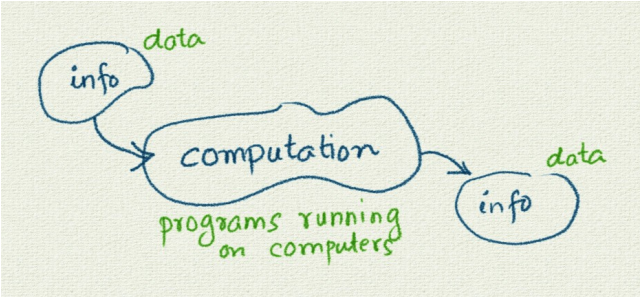
Work on AI started many decades ago. The first wave of AI peaked in the 1970s and 1980s, and was based on handcrafted knowledge. Rules related to a probem domain were fed into the system by human beings, and the system made decisions based on incoming data and higher-level general rules of reasoning. These systems worked great on narrow (but important) problems like planning, routing, scheduling, process-control, etc. But these systems had a big limitation. They stumbled on natural data – like images, sounds, and text.

The second wave of AI started around 2015, and does an excellent job at recognizing patterns in natural data and making predictions based on these patterns. This wave is based on a technology called deep learning. Over the past five years, it has taken the world by storm. Some of the big applications of deep learning have been (a) in computer vision – in recognizing the contents of images and videos – in areas like medical imaging, security, social networks, and education, (b) in understanding spoken words – in areas like voice recognition (with devices like Alexa, etc), and (c) in understanding written language – in areas like language translation, chatbots, etc.

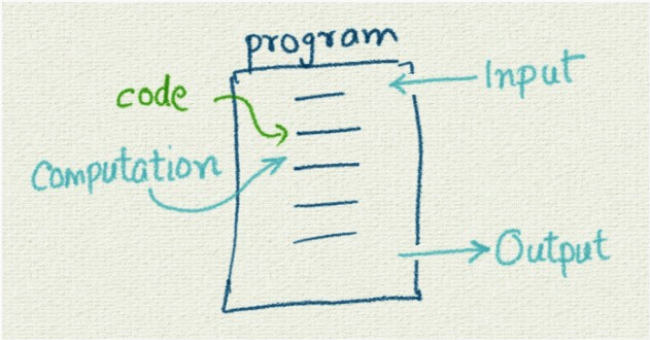
But what exactly is deep learning?

Let's take a step back and look at our world. Over the last couple of decades, we have surrounded our physical world with a digital cloud. We have created this cloud to provide ourselves with information, communication, productivity, entertainment, and more. Services that run on the digital cloud include information search, maps, online shopping, travel planning, meetings, games, educational courses, movies, shows, etc., etc.

The life energy of this digital cloud is computation, and it's important for us know a little bit about computation to make sense of deep learning.



A computation runs on a computer, and its basic job is to take in information and spit out new information. A computation is defined via coding or programming – a human sits down and writes some code and creates a program. When this program runs, it takes in some inputs, carries out the computation defined within it, and produces some outputs.



Let's look at an example to make all of this more concrete. Think of a mapping application like Google Maps. What kind of computation do you think it carries out? One of the things this App can do for you is to tell you directions from one location to another. You give it two locations – a source and a destination – and it gives you back the route between the two locations. The computation for this, running on the mapping App, takes in the two locations as input, and outputs (or returns) the route between these locations.

Within a computer program, a computation is defined in something called a function. Let's look at an example:

```
def multiply(n1: Int, n2: Int) = {
    var answer = 0
    repeat(n1) {
        answer = answer + n2
    }
    return answer
}
```

Here, we have defined a multiplication function in terms of repeated addition. The multiply function takes in two numeric (integer) inputs and returns their product.

From the perspective of a function, the earlier routing computation that we talked about will look something like this:

```
def calculateRoute(source: Location, destination: Location) = {
    val route = // some complex calculation
    return route
}
```

So, we have this digital cloud running around us. It's driven by computations/functions. And these computations/functions *are written by human beings*.

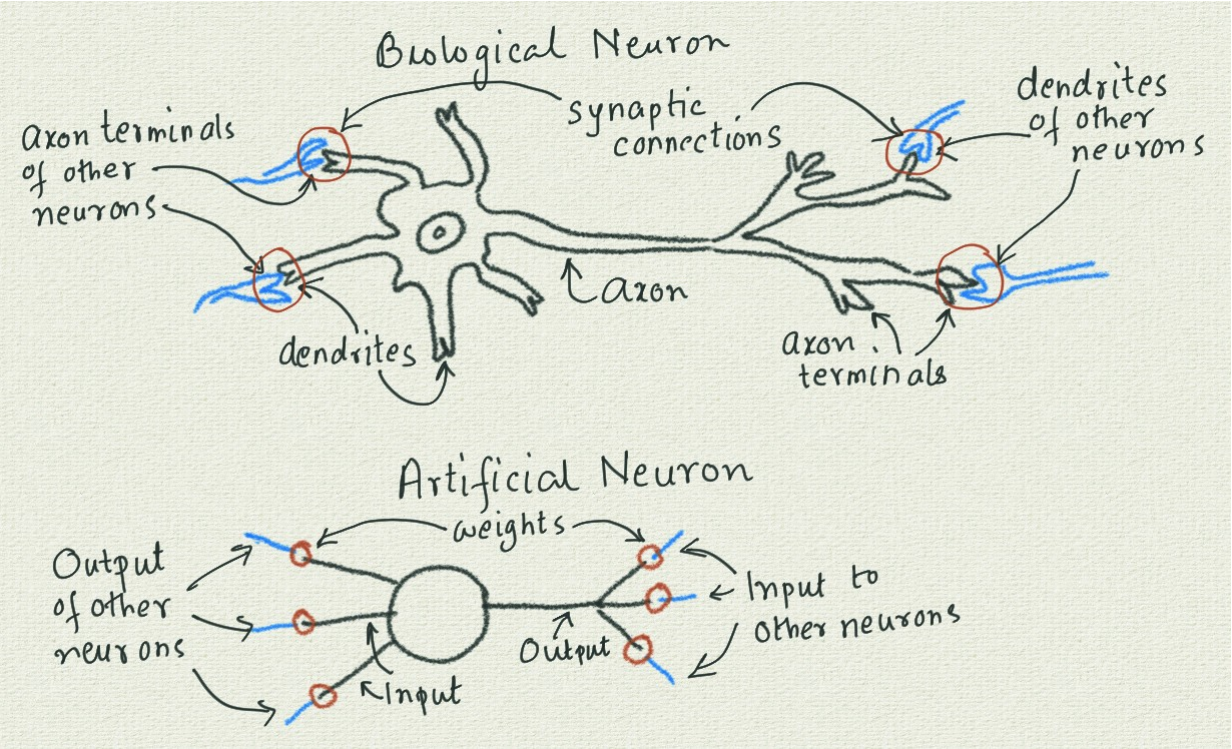
Deep learning brings a new idea into the mix. With deep learning, *the computer learns functions based on data*. Why is this important? Because for many types of problems, it is almost impossible for a human being to write the desired function. For example, it is really difficult to hand-write a function that can take an image as an input and determine if there is a dog present in the image.

How does deep learning learn functions? Let's find out.

Deep learning is based on artificial neural networks (ANNs). These networks are made out of layers of artificial neurons, whose working is inspired by biological neurons in the human brain.

A biological neuron has the following important components:

- dendrites – to carry impulses from other neurons to the cell body.
- cell body – where all the action happens.
- an axon – to carry an impulse from the cell body towards other neurons.
- axon terminals – to pass an impulse to the dendrites of other neurons.
- synaptic connections – between axon terminals and dendrites – to carry an impulse across neurons.



Biological and artificial neurons.

In a biological neuron, dendrites carry inputs signals from other neurons to the cell body. If the sum of the impulses arriving at the cell body exceed a threshold, the neuron fires and sends an impulse down its axon. This impluse then branches out via axon terminals into the dendrites of other neurons – via synaptic connections.

The synaptic connections between biological neurons are very important. They define how impulses flow across the brain. If two neurons have a strong synaptic connection, the impulse flow between them will be strong. On the other hand, if the synaptic connection between two neurons is weak or non-existant, then the impulse flow between them will be weak or zero.

In other words, *the strength of the synaptic connections in a brain defines the learning that is persent in the brain, in terms of the memories and the skills available in that brain.*

In an artificial neural network, just like in the human brain, input connections carry information into an artificial neuron. If the sum of these inputs is over a certain threshold, the artifical neuron fires, and sends information to the next neuron connected to it.

In an artificial neuron, the entity that corresponds to the strength of a synaptic connection (between biological neurons) is called a weight (or parameter). The information going from one neuron to another is multiplied by a weight before it is received by the second neuron.

So, in a manner similar to how learning exists in a human brain, *the weights of an artificial neural network define the learning of the network*, and determine how the network behaves when it is presented with some input.

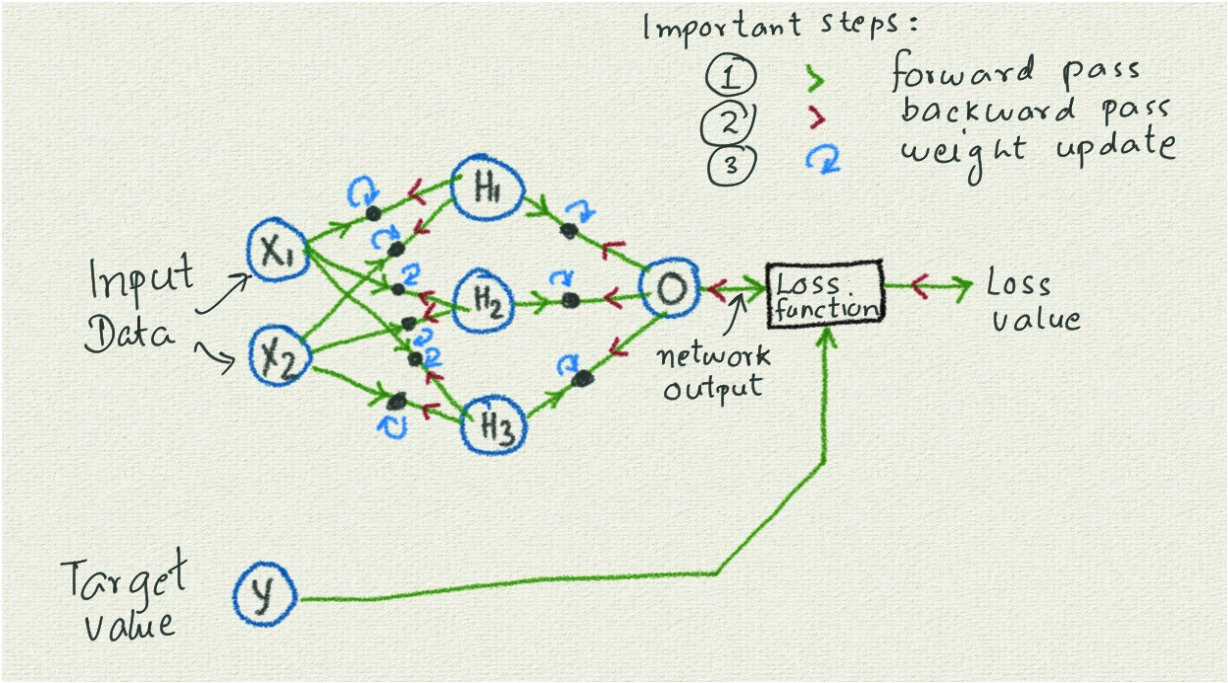
If you think of an artificial neural network as a function, the weights of the network determine what the function does. So – to learn a desired function, the network needs to learn the weights that correspond to that function.

How is this done?

The basic idea is simple:

- Randomly initialize the weights.
- Feed in data through the neural network (this is called the forward pass).
- Compare the output from the network to the target (desired) ouput. A special kind of a function called a loss function is used to do this comparision. The output from the loss function tells you how well the network is doing. The greater the loss, the worse the performance of the network.
- Determine how strongly each weight in the network contributes to the loss (this is called the backward pass, and is done via a technique called back-propogation).
- Update the weights based on how strongly they contribute to the loss, in a manner that reduces this loss.
- Repeat this process till a good enough solution is obtained.

The following figure shows this in action [Todo – simplify]:



Once a function is learned satisfactorily, new data can be fed into the network, and the output from the network represents the value of the learned function for that input data.

We have just seen at a high level how neural networks work (to learn functions). Deep learning is more of the same – it's the usage of neural networks with many many layers of neurons. The large number of layers make it easier to learn complex functions – because every layer in the network can extract useful features from the previous layer, and these multi-layer features have the ability to efficiently approximate *any* function.

Neural networks have been around since the 1980s. So how come these networks, in the form of deep learning, are only now starting have a big impact. It's because until a few years ago, deep networks were almost impossible to train. But then the following things came together:

- Massive amounts of data.
- Massive compute power.
- Algorithmic breakthroughs.

All of these factors have combined to give deep learning its day in the sun!

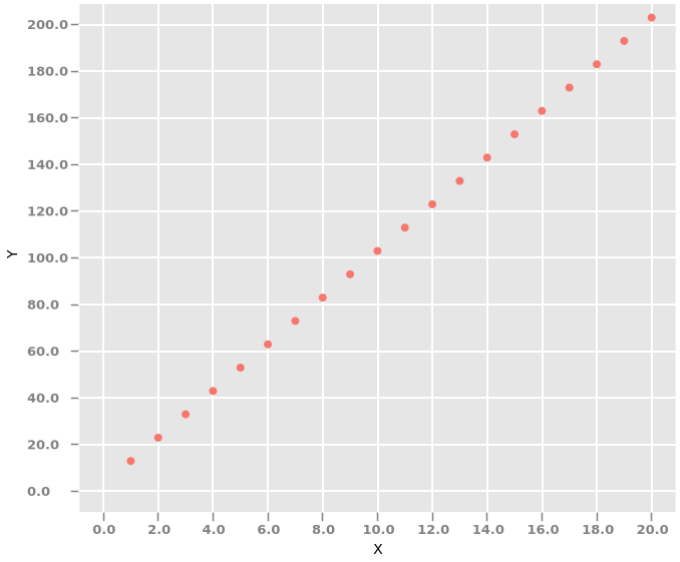
It's time for us to look at some concrete examples of neural networks to get a better idea of how they work.

The first function that we will look at is the following:

```
def linear(x: Double) = {
   10x + 3
}
```

Here's a table and a chart showing sample inputs and the corresponding outputs of this function.

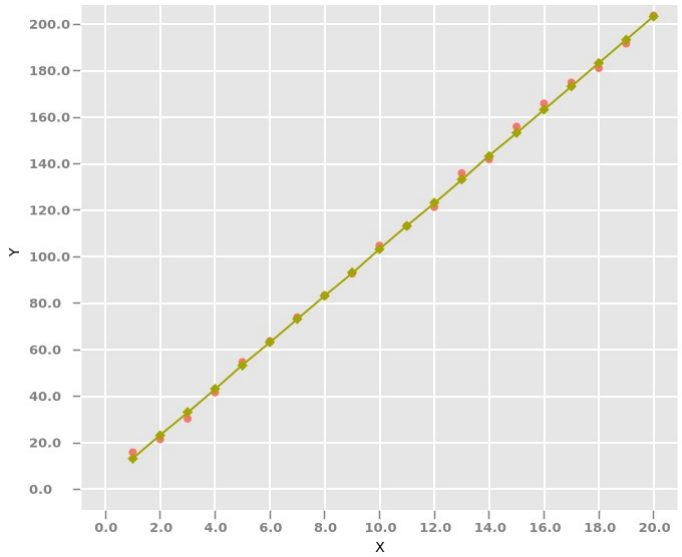
Input (X)	Output (Y)
1	13
2	23
3	33
...	
20	203



Now let's get a neural network to learn this function. For this, we will do a couple of things:

- Put a little bit of noise in the data – to simulate real world data (which is always noisy).
- Use two learnable parameters in the network, because we know that we want to learn a linear function (and two parameters are enough for that).

After learning, here's what the neural network comes up with. The red dots represent the original function sampled at regular intervals (with some noise added). The mustard-green line is the function that the neural network has learned.



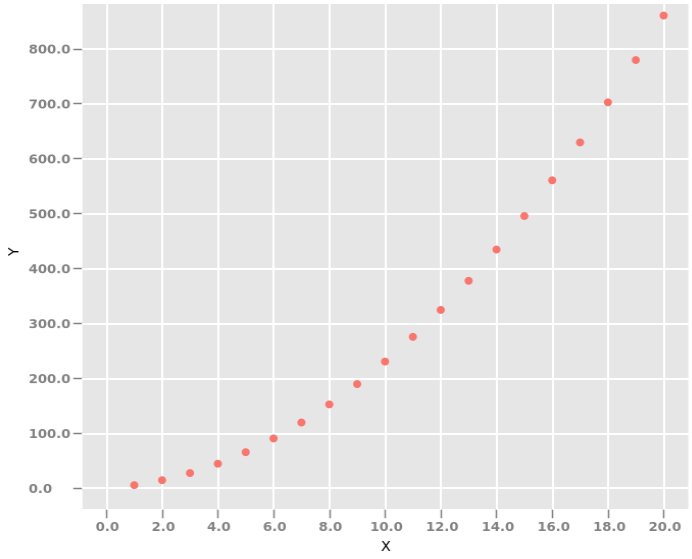
Looks pretty good, right?

Let's see if a neural network can learn a more complex function.

```
def nonLinear(x: Double) = {
   2x^2 + 3x + 1
}
```

Just like for the previous funtion, here's a table and a chart showing sample inputs and the corresponding outputs of this function.

Input (X)	Output (Y)
1	6
2	15
3	28
...	
20	861

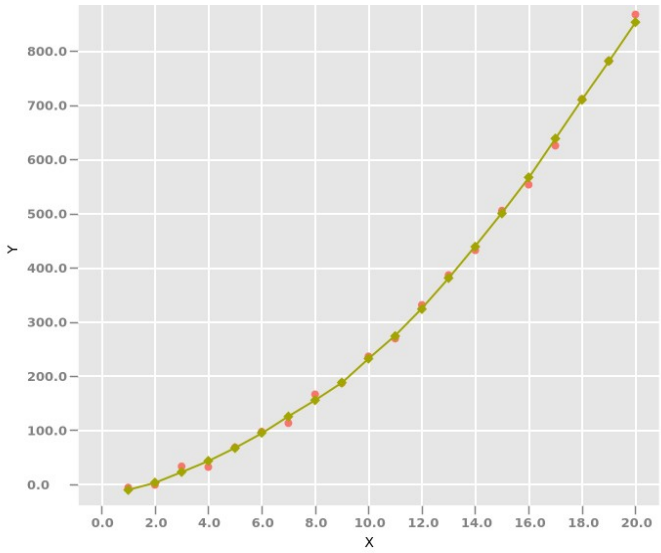


Again, let's get a neural network to learn this function.

- Like last time, let's put a little bit of noise in the data – to simulate real world data.
- Unlike last time (where we knew that we wanted to learn a linear function, and gave our network two parameters), we will not make any assumptions about the shape of the target

function. We will use a general neural network with sufficient power to learn a non-linear function.

After learning, here's what the neural network comes up with. The red dots represent (as before) the original function sampled at regular intervals (with some noise added). The mustard-green line is the function that the neural network has learned.



Once again this looks pretty good, right?

So, we have seen a neural network learn a linear function, and then another neural network learn a non-linear function. Neural networks, and especially deep neural networks, are capable of a lot more. They can do classification (identifying the category that an input belongs to), object detection (finding things of interest in images), unsupervised learning (identifying underlying structure and groupings in data), reinforcement learning (learning to achieve goals in environments that provides rewards and penalties), and more. But we don't have time to delve into these topics today. Your take-away from this article should be that at the heart of wherever deep learning is used, there is the idea of – learning some kind of a function through the learning of multiple layers of weights.

We have taken a whirlwind tour through the field of deep learning. Hopefully you now know that the core idea behind deep learning is pretty simple. The magic happens when you throw a lot of data and a lot of compute power at the this simple idea – to solve problems in many different domains. So now, when you see some App doing face recognition or disease identification or language translation or any of the other fancy things that deep learning can do, you will hopefully know the essence of what is going on underneath the covers.

For live experimentation with a lot of the ideas presented in this article, you can check out the Kojo-AI project (https://github.com/litan/kojo-ai-2).