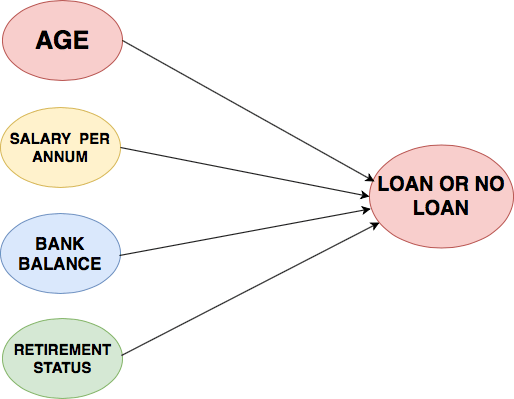
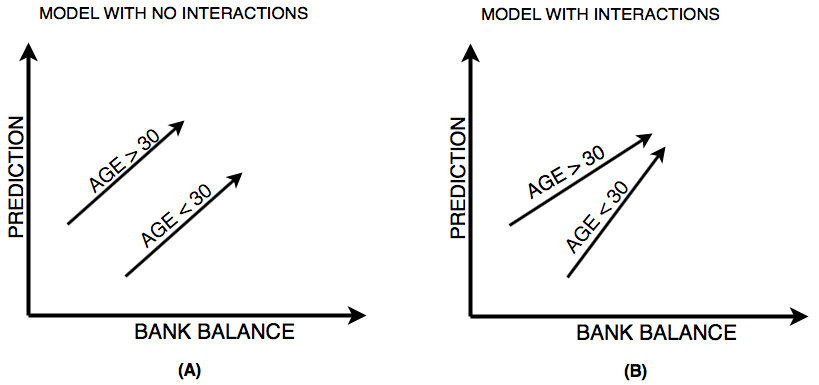
Common Assignment for Regression and Deep Learning

Imagine you work for a loan company, and you need to build a model for predicting, whether a user (borrower) should get a loan or not? You have the features for each customer like age, bank balance, salary per annum, whether retired or not and so on.

Consider if you want to solve this problem using a linear regression model, then the linear regression will assume that the outcome (whether a customer's loan should be sanctioned or not) will be the sum of all the features. It will take into account the effect of age, salary, bank balance, retirement status and so. So the linear regression model is not taking into account the interaction between these features or how they affect the overall loan process.



The above figure left (A) shows prediction from a linear regression model with absolutely no interactions in which it simply adds up the effect of age (30 > age > 30) and bank balance, you can observe from figure (A) that the lack of interaction is reflected by both lines being parallel that is what the linear regression model assumes.

On the other hand, figure right (B) shows predictions from a model that allows interactions in which the lines do not have to parallel. **Neural Networks** is a pretty good modeling approach that allows interactions like the one in figure (B) very well and from these neural networks evolves a term known as **Deep Learning** which uses these powerful neural networks. Because the neural network takes into account these type of interactions so well it can perform quite well on a plethora of prediction problems you have seen till now or possibly not heard.

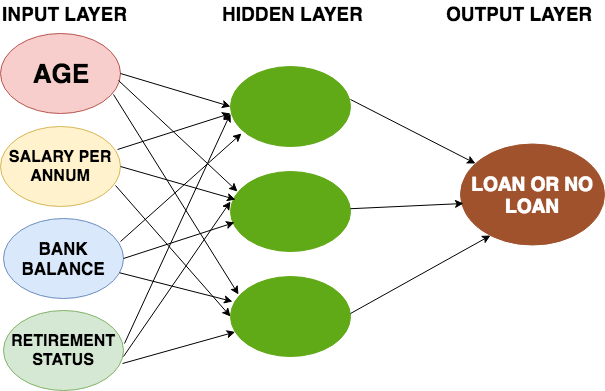
Since neural networks are capable of handling such complex interactions gives them the power to solve challenging problems and do amazing things with

* Image
* Text
* Audio
* Video

This list is merely a subset of what neural networks are capable of solving, almost anything you can think of in data science field can be solved with neural networks.

Deep learning can even learn to write a code for you. Well isn't that super amazing?

**Interactions in Neural Network**



The neural network architecture looks something similar to the above figure. On the far left you have the **input layer** that consists of the features like age, salary per annum, bank balance, etc. and on the far right, you have the **output layer** that outputs the prediction from the model which in your case is whether a customer should get a loan or not.

The layers apart from the input and the output layers are called the **hidden layers**.

Now the question is why they are called hidden layers?

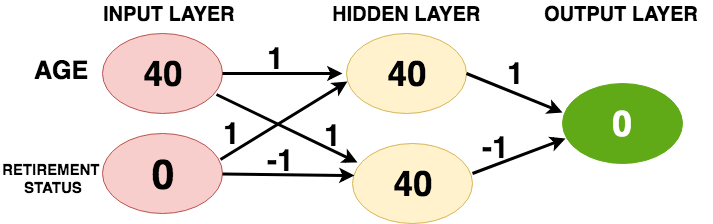
Well, one good reason is while the input and output layers correspond to apparent things that occur or are present in the world and can be stored as data but the values in the hidden layers are not something that relates to the real world or something for which have data.

Technically, each node in the hidden layer represents an aggregation of information from the input data; hence each node adds to the model's capability to capture interactions between the data. The more the nodes, the more interactions can be achieved from the data.

**Forward Propagation**

Let's start by seeing how neural networks use data to make predictions which is taken care by the forward propagation algorithm.

To understand the concept of forward propagation let's revisit the example of a loan company. For simplification, let's consider only two features as an input namely age and retirement status, the retirement status being a binary ( 0 - not retired and 1 - retired) number based on which you will make predictions.



The above figure shows a customer with age 40 and is not retired. The forward propagation algorithm will pass this information through the network/model to predict the output layer. The lines connect each node of the input to every other node of the hidden layer. Each line has a weight associated with it which indicates how strongly that feature affects the hidden node connected to that specific line.   
  
There are total four weights between input and hidden layer. The first set of weights are connected from the top node of the input layer to the first and second node of the hidden layer; likewise, the second set of weight are connected from the bottom node of the input to the first and second node of the hidden layer.

Remember these weights are the key in deep learning which you train or update when you fit a neural network to the data. These weights are commonly known as **parameters**.

To make a prediction for the top node of the hidden layer, you consider each node in the input layer multiply it by the weights connected to that top node and finally sum up all the values resulting in a value 40 (40 \* 1 + 0 \* 1 = 40) as shown in above figure. You repeat the same process for the bottom node of the hidden layer resulting in a value 40. Finally, for the output layer you follow the same process and obtain a value 0 (40 \* 1 + 40 \* (-1) = 0). This output layer predicts a value zero.  
  
Now you might wonder what the relevance of value zero is, well you consider the loan problem as binary classification in which an output of zero indicates a loan sanction and an output of one indicates a loan prohibition.

That's pretty much what happens in forward propagation. You start from the input layer move to the hidden layer and then to the output layer which then gives you a prediction score. You pretty much always use the multiple-add process, in linear algebra this operation is a dot product operation. In general, a forward propagation is done for a single data point at a time.

Reference: https://www.datacamp.com/community/tutorials/introduction-deep-learning