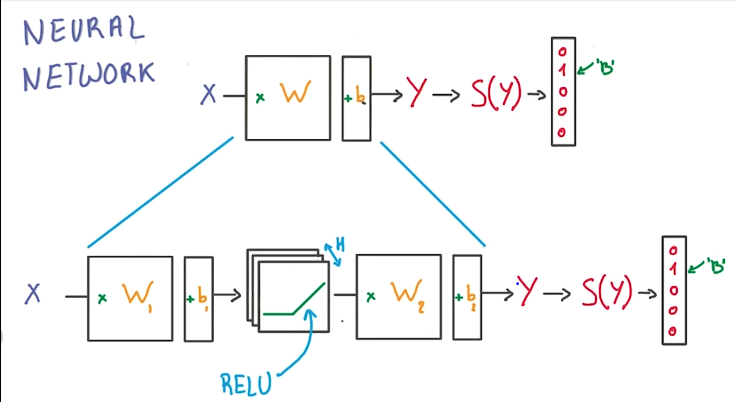


This model is relatively limited. You might want to use many more parameters than used here so that you can represent more relationships. This representation can only be used when representing inputs that have a linear relationship with the output, so an additive relationship.

If we wanted to represent a model where 2 inputs get multiplied, you need to use a non-linear model.

Linear models have the bonus of being easier to compute, due to being able to use matrixes in calculations, and use GPUs, which are designed for things like matrix multiplication.



This model is much more flexible due to having the RELU (the simplest non-linear function). You have more paramters you can use (H) and can represent non-linear relationships.

H represents the number of RELU units that you have in the classifier.

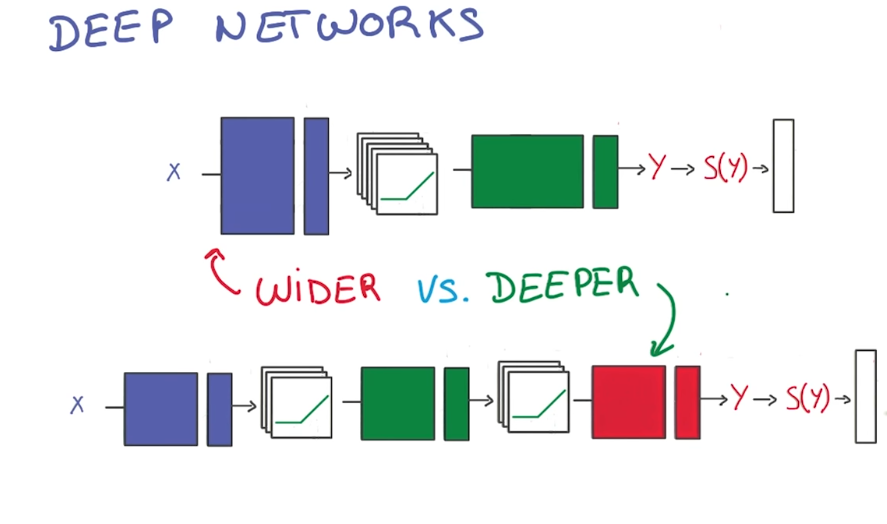
1. The first layer effectively consists of the set of weights and biases applied to X and passed through ReLUs. The output of this layer is fed to the next one, but is not observable outside the network, hence it is known as a *hidden layer*.
2. The second layer consists of the weights and biases applied to these intermediate outputs, followed by the softmax function to generate probabilities.

When an activation function (in this case the RELU) is non-linear, then a two-layer neural network can be proven to be a universal function approximator.

If you had a multi-layer neural network with linear activation functions, the entire network is technically the equivalent to a single-layer model.

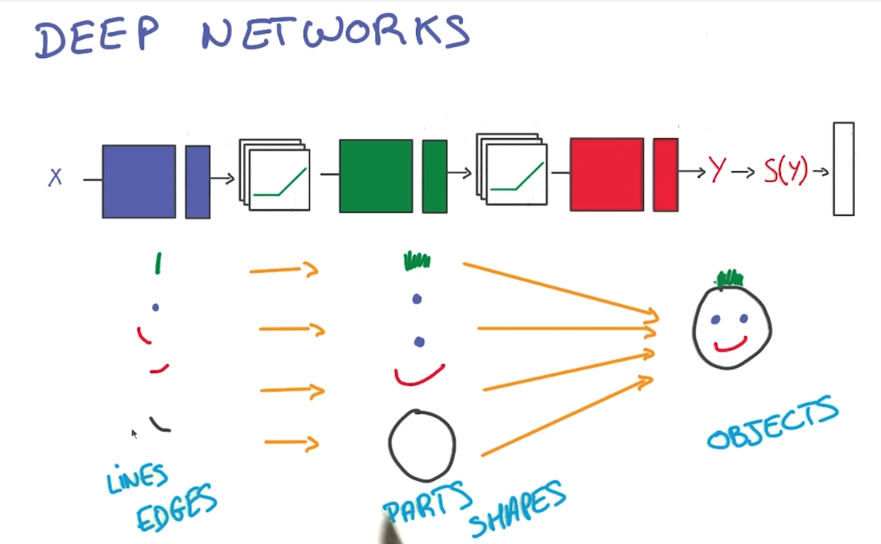
The H parameter (size of the hidden layer) is chosen to allow for much more parameters, making your model more flexible. The size can always be increased to add more flexibility, but it turns out that isn’t necessarily efficient due to the slower it becomes to train.

It is much better to add more layers, and to make your model deeper rather than to increase the size of your hidden layer and increase the width.



You typically have much greater performance with much less parameters when you have a deeper neural network rather than a bigger hidden layer (wide).

Therefore going deeper is much better for parameter optimization.



A lot of interesting natural phenomena that need to be observed tend to have a hierarchical structure that deep models capture well.

Building deep neural network to recognize things in an image for example tends to look for simple things at the lowest layers, then move up to more complicated shapes.

The model structure matches the data abstractions we expect in our data, meaning the model has an easier time learning them.