Neural Networks

Table of Contents

[Activation Functions 5](#_Toc138881709)

[Human Brain 5](#_Toc138881710)

[Space Warping 6](#_Toc138881711)

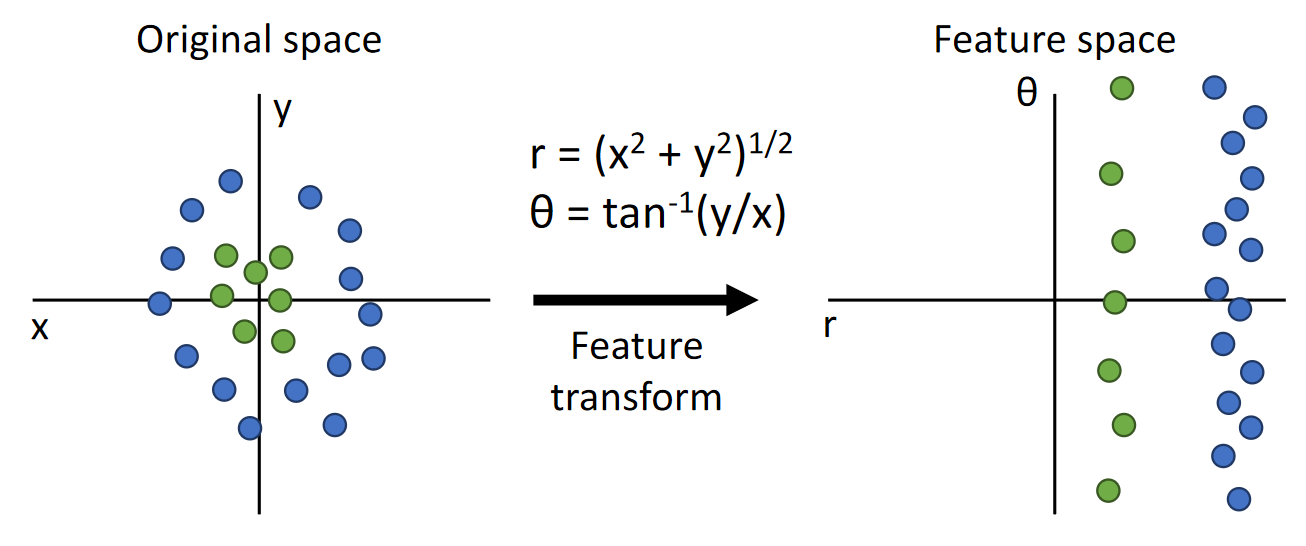
[Network Depth 7](#_Toc138881712)

[Universal Approximation 8](#_Toc138881713)

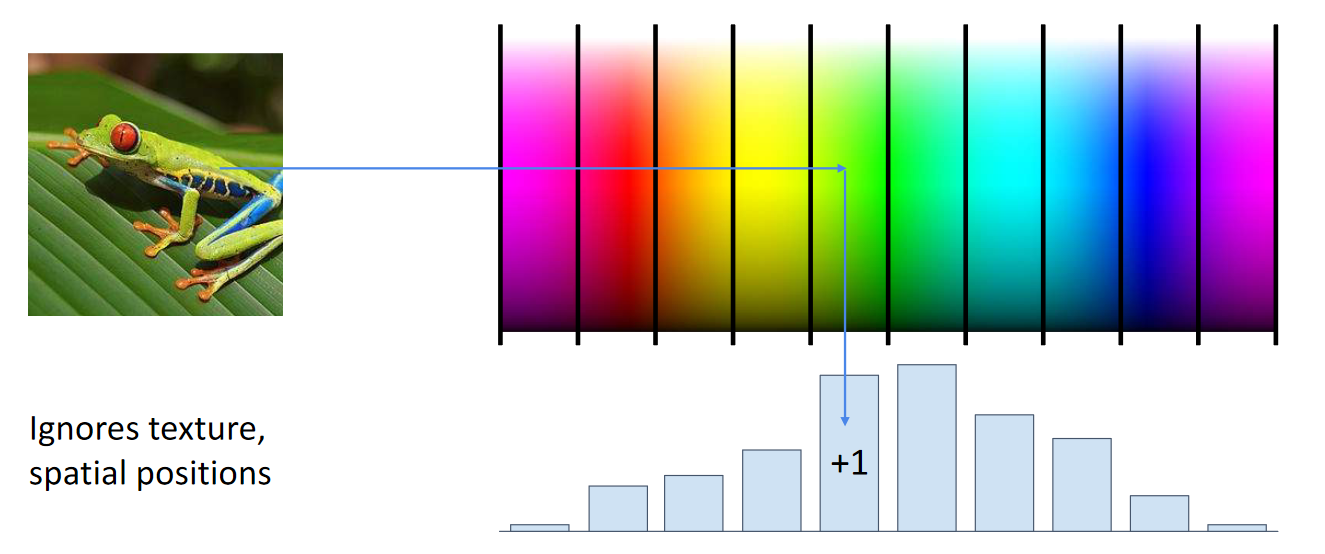
[Convex Functions 9](#_Toc138881714)

An unaddressed issue of linear classifiers is the fact that they cannot have **non-linear decision boundaries**. This issue has been addressed in the past by several mechanisms:

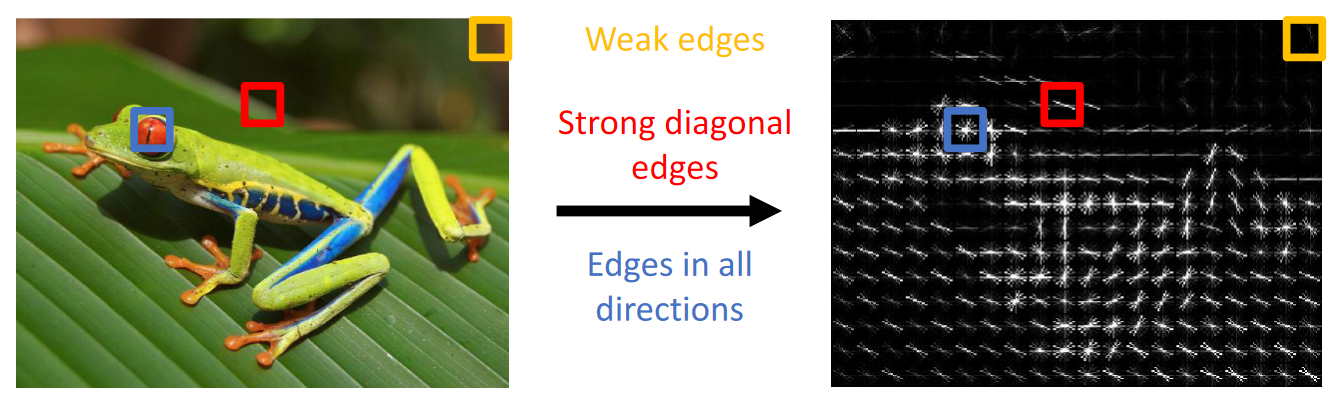
* Using an alternative feature space. This method can only be used for simple cases.



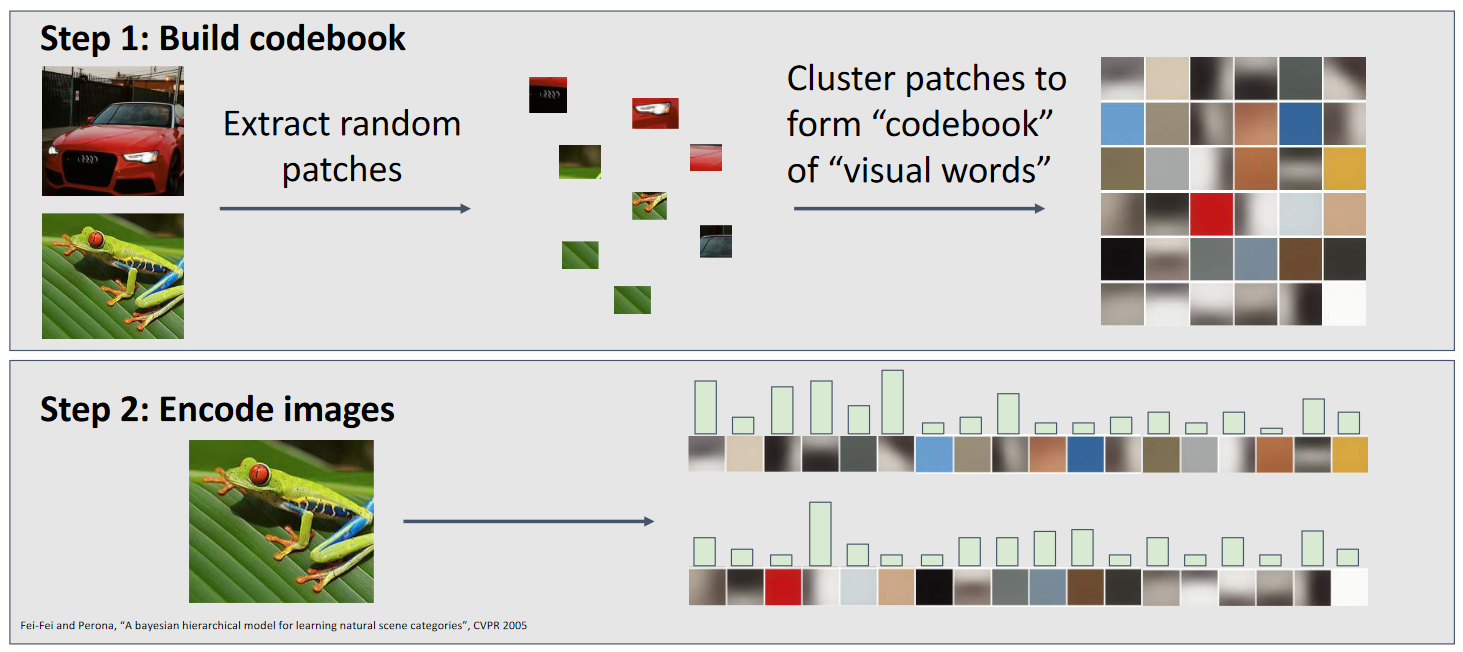
* Using colours as features. This fails in cases where images have similar colours.



* Histogram of Oriented Gradients (HOG). This divides the image into regions and calculates the histogram of edge directions weighted by edge strength in each region.

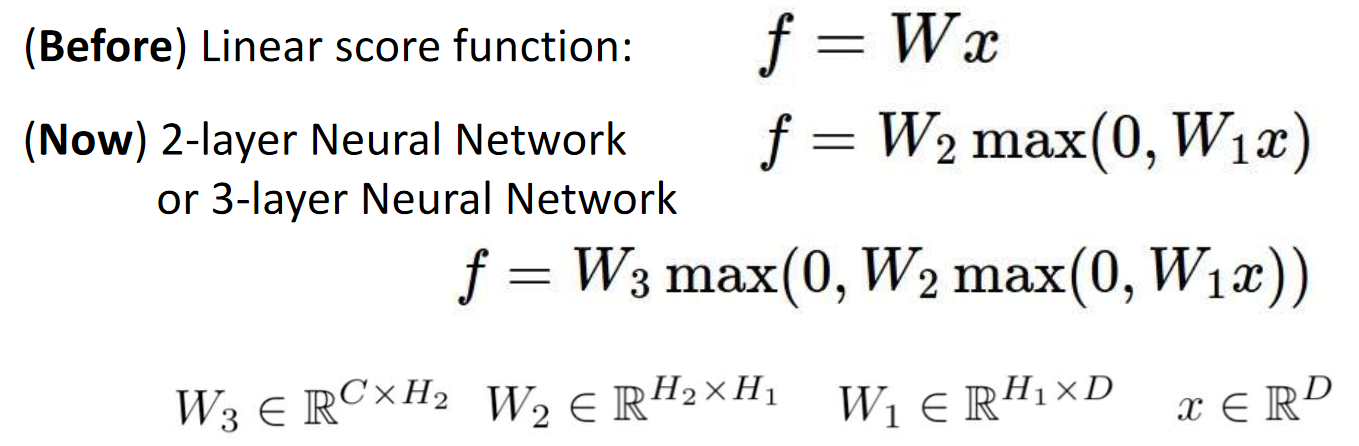


* Bag of Words, one of the first data driven methods. This method extracts patches from the training images and then compares the patches to test images.

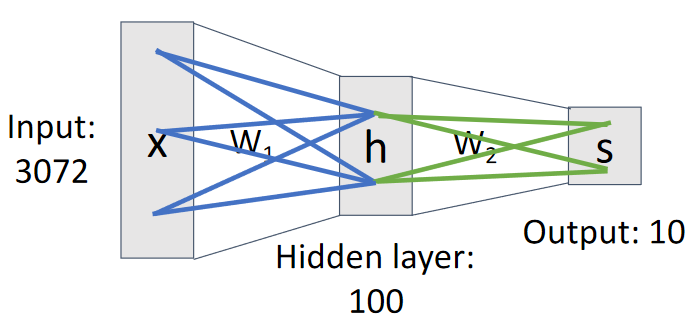


The mechanisms above can also be combined. All of these methods are called **feature extraction** methods. The idea of a **neural network** is to create a model that can extract these features, thus removing the burden of having to handcraft features from humans.

A neural network is just multiple linear layers stacked together. The number of layers being used is called the **depth** of the network.

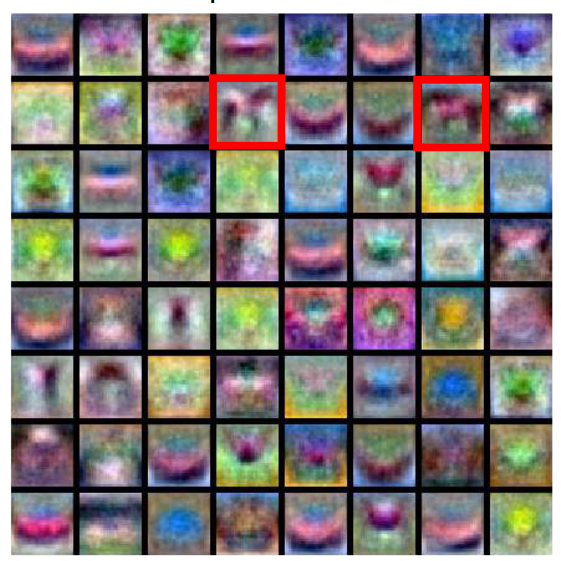
****

The layers in between are called **hidden layers** because in practice, their values are never seen. However, they do have an effect. In the diagram below, all elements of the input affect all elements of the hidden layer, which in turn affects all elements of the output.



Because of these, the hidden layers are also called **fully connected layers**.

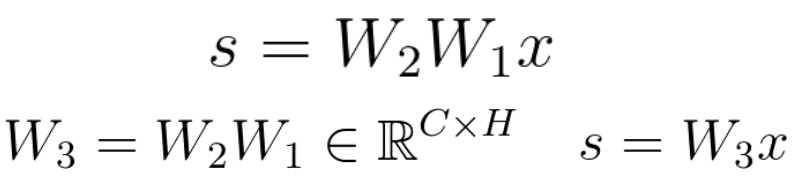
The use of multiple layers essentially allows the model to create **multiple templates** for each class. This gives it the opportunity to learn alternative representations of the same data. Notice that the templates below have successfully separated templates for horses facing in two directions. We no longer have two-headed horses!



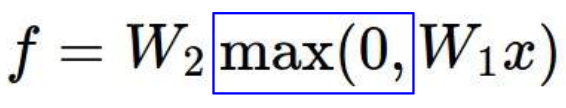
Most of the templates, however, represent partial information about the classes. Thus, they cannot easily be interpreted.

## Activation Functions

Notice that the equation shown earlier contains a **max** value. There is an important reason for this. If we just keep stacking multiple linear layers together, all of the layers can mathematically be shown to be equivalent to just a **single layer**. This means it won’t solve our original problem.



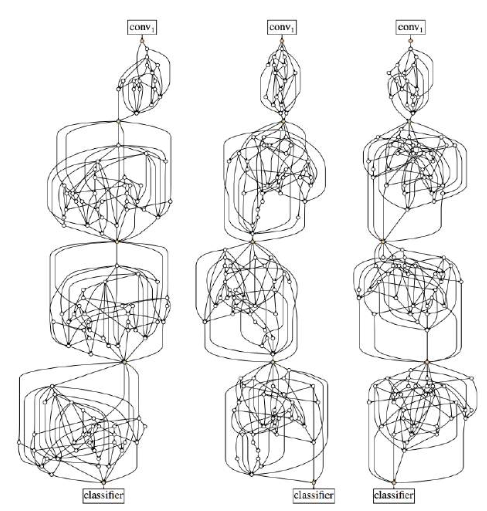
To get around this, we introduce **non-linearity** between the layers. This is done with the help of **activation functions**. The max value in the equation is one such popular activation function called a **ReLU function**. We will be studying various activation functions in depth later on.



## Human Brain

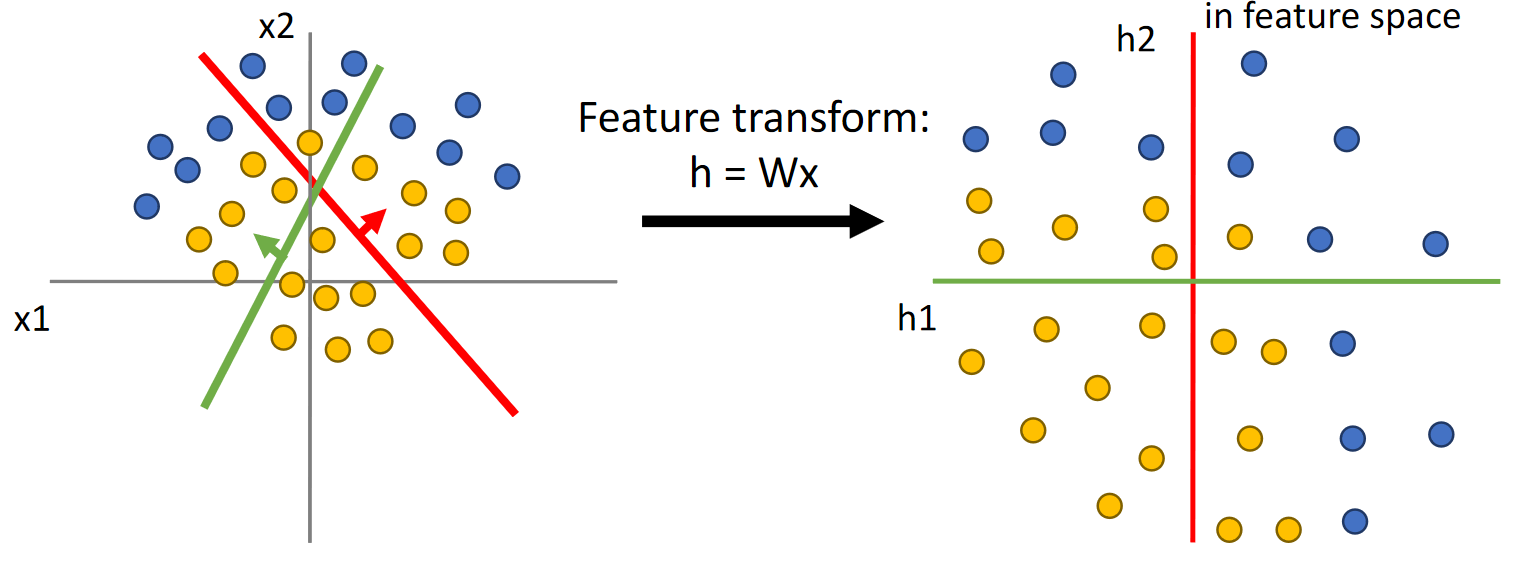
The concept of a neural network was created in an attempt to mimic the human brain. The human brain has some similarities, with neurons taking inputs from previous ‘layers’ and creating outputs to pass to future ‘layers’. However, unlike neural networks, the human brain is not arranged in a step-by-step fashion. It is a mess with random connections.

Alternative forms of neural networks that connect layers in a random manner have also been explored in past literature.

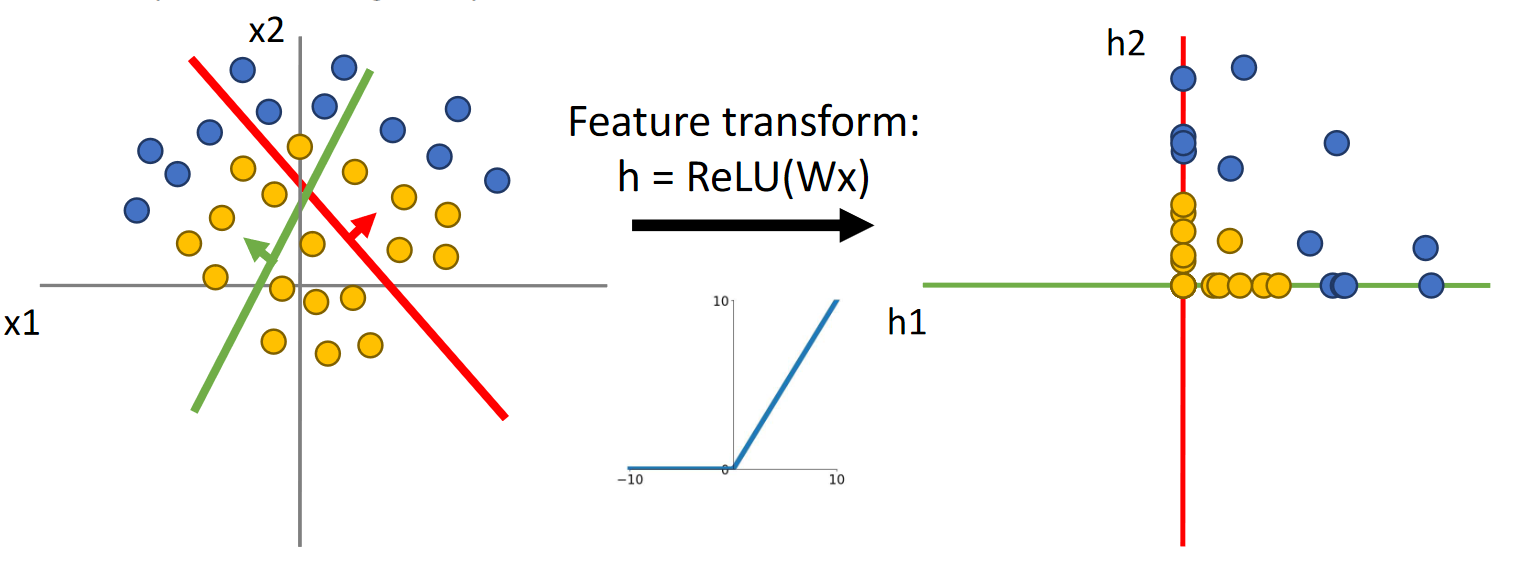


## Space Warping

The benefits of neural networks and the way in which the activation functions create non-linearity can be easily visualized. Suppose we have a situation where the data points cannot be linearly separated. Undergoing a **feature transformation** will create a new feature space, but it will not change the fact that the data points cannot be linearly separated.

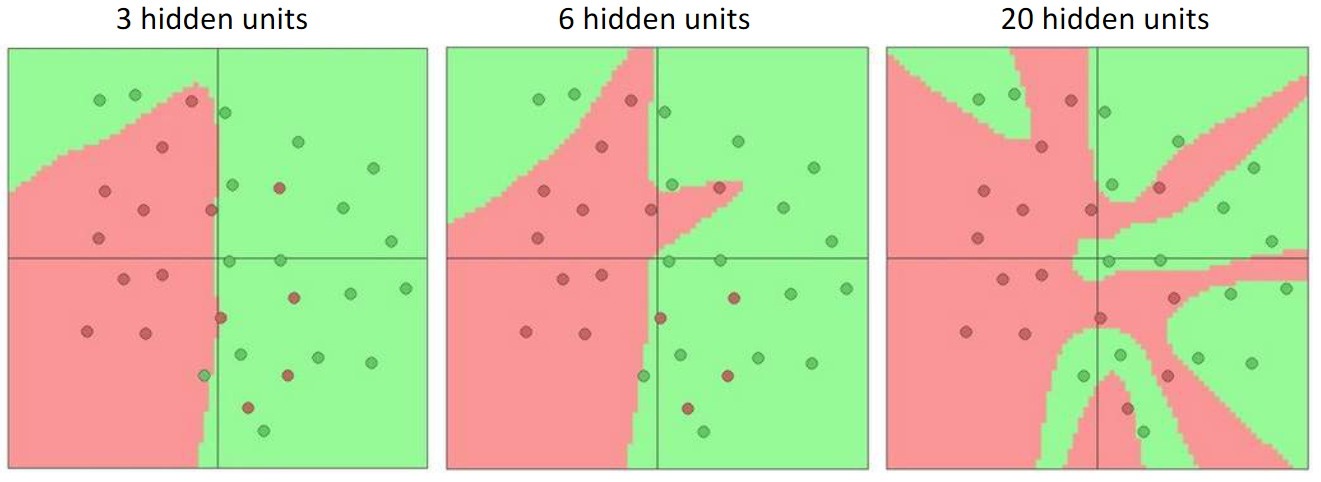


However, an activation function like the ReLU function can force the data to be separable linearly.

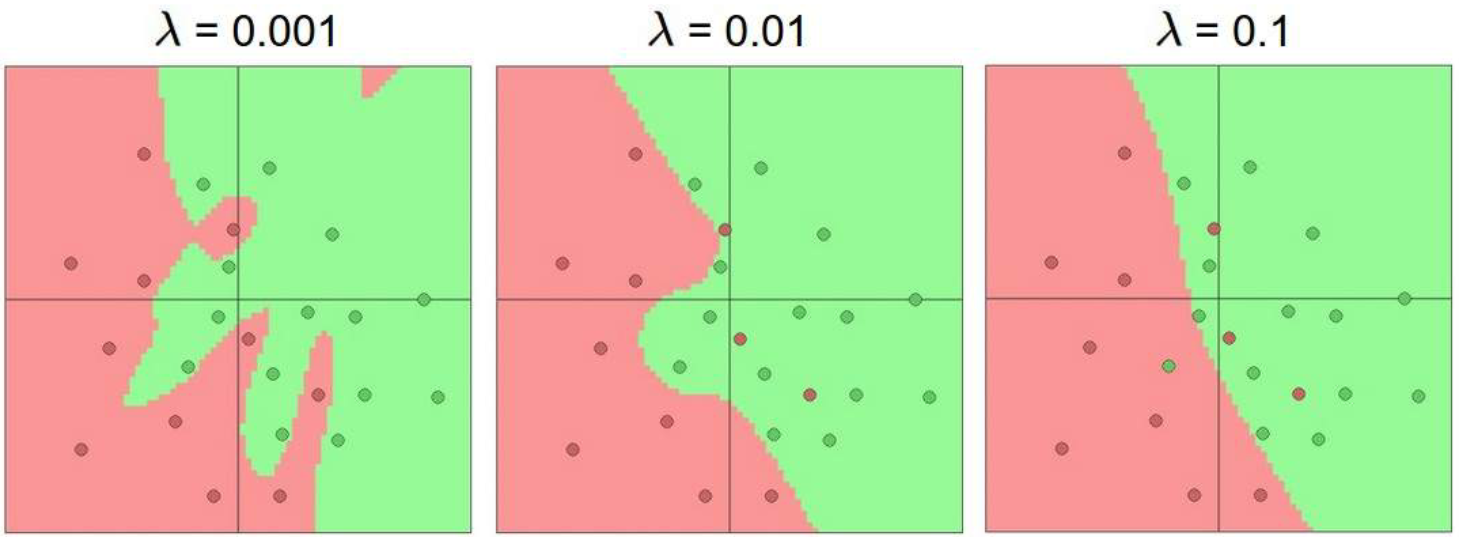


## Network Depth

Increasing the number of hidden units in a network increases the ability of the network to adapt to more difficult data. However, this is a two-edged sword since it also increases the likelihood that the network will overfit.

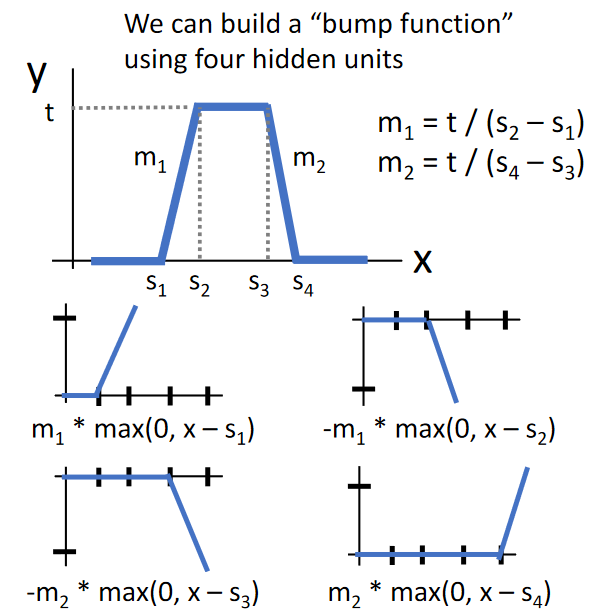


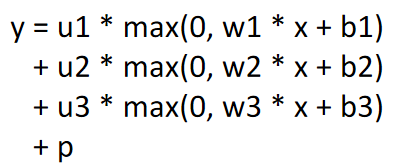
It is possible to reduce the overfitting without decreasing the network depth by increasing the **regularization strength**.



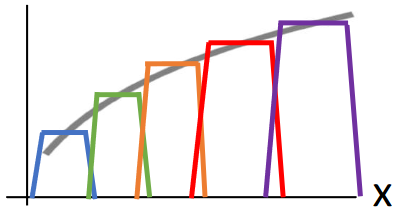
## Universal Approximation

The theory of **Universal Approximation** states that a neural network with just one hidden layer can approximate any function. A function with ‘bumps’ can theoretically be learnt by four separate units in a single layer.





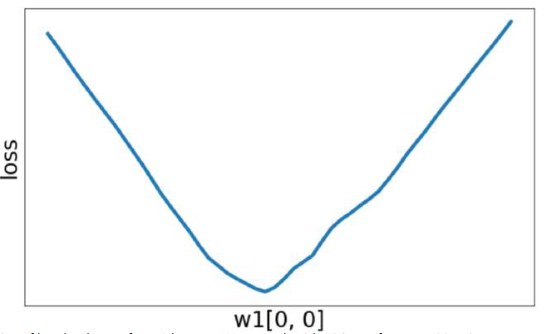
This can be expanded to make the neural network approximate any function using those bumps. The wider the network, the better the approximation since the bumps will become narrower.

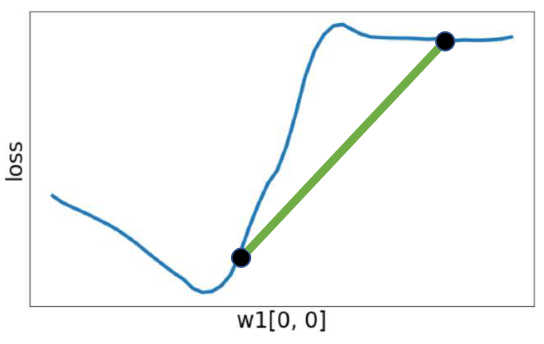


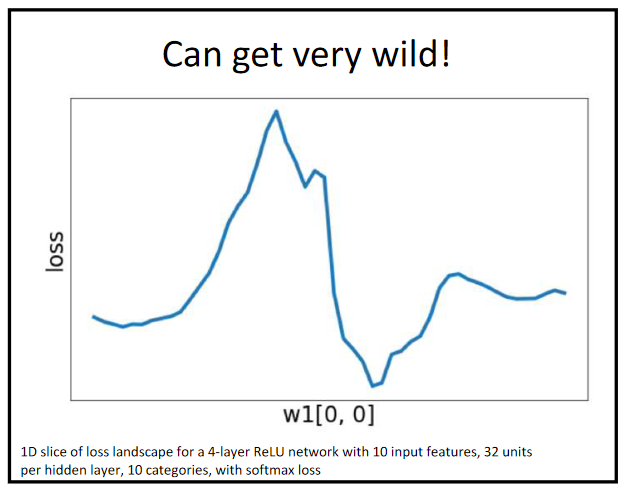
This theory, however, tells us nothing about whether SGD allows us to learn any function or how much data is required to do so. This theory does not actually take place in reality.

## Convex Functions

Generally speaking, convex functions are easy to optimize. Linear classifiers can optimize these. For neural networks, this can hold true but it can also become very weird.







We basically have no guarantee that neural networks will converge, but they seem to work **empirically**.