

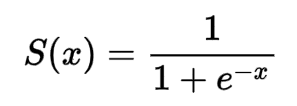
First neural n/w- perceptron.

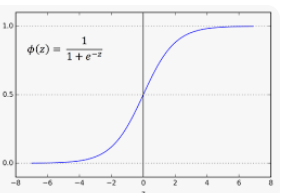
What is Activation function

* activates the neurons. Eg: when a hot cup is touched to human skin, the neurons get activated.
* Z=Act(y)

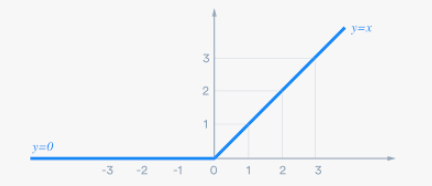
Its Types:

1. Tanh (Hyperbolic Tangent): The Tanh function maps input values to a range between (-1, 1), making it a popular choice for the output layer in neural networks.
2. Sigmoid: Used in the outer layers of a network





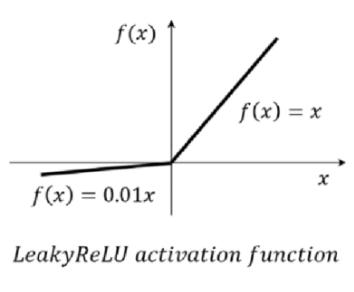
1. Relu : ReLU could also accelerate the convergence of gradient descent and solves the problem of vanishing gradient.



1. Leaky relu: The ELU activation function can address the issue of dying neurone problem in ReLU, as its gradient is non-zero for all negative values.

F(x) = 0.01x, x<0

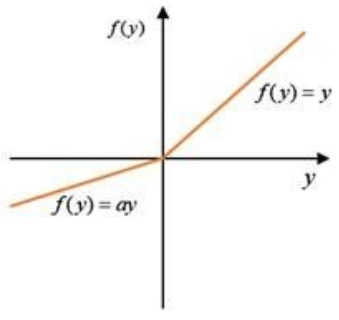
F(x) = x, x>=0



1. PRelu (Parametric Relu): it is called parametric bec, for a=0, it becomes ReLU and for a=0.01, it is leaky ReLU.

F(x) = ax, x<0

F(x) = x, x>=0



Training n/w with single layer

O/P

ACT(Y)

W4

X3

X2

X1

W3

W1

W2

* O/P=Y’
* LOSS= (Y’-Y)
* Optimizer (gradient descent) is used to reduce loss so that actual o/p = correct o/p.
* Back propagation is done to update the weights of the neural network in order to minimise the error between predicted and actual output. This is done by calculating the gradient of the loss value w.r.t to the weights of the NN and updating the value of weights with this value, thus converging towards minimum error function.
* *Wi-new =Wi-old -alpha\**
* Alpha- learning rate, should be small only otherwise the convergence will not happen properly.

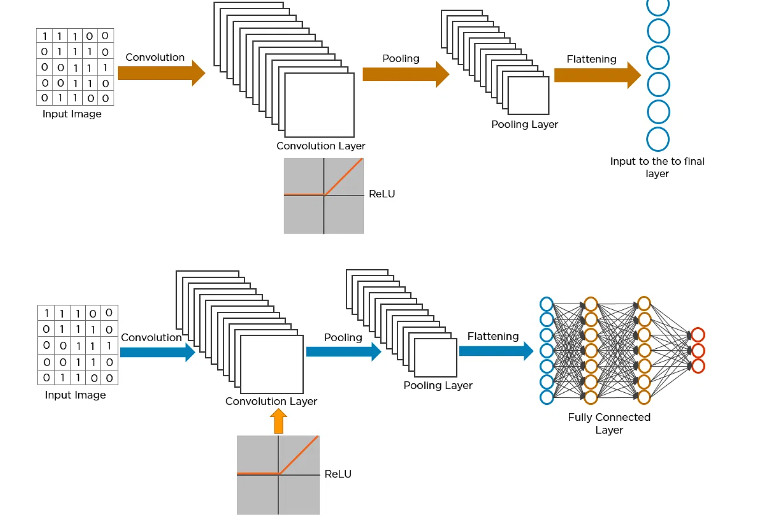
**Multi layer neural network**

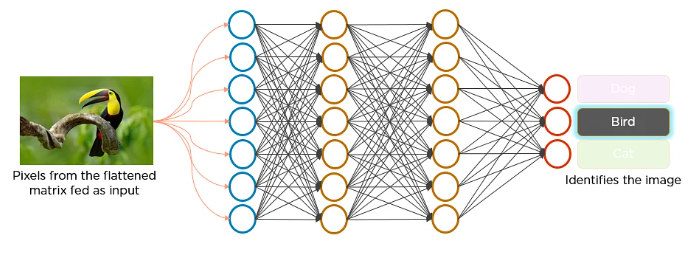
* Single layer NN are prone to under-fitting.
* Multi-layer NN are mostly prone to over fitting.

**Convolutional Neural Network (CNN)**

* Filter/Kernel: it is a matrix with which the convolution operation is done in CNN. It is done to extract features from the input then called as feature map. After each convolution operation, we have the application of a Rectified Linear Unit (ReLU) function, which transforms the feature map and introduces nonlinearity.
* Filter size=fxf
* Input/ Image size=nxn
* Padding: Padding is done to increase the size of the input/image matrix since the filter does not fit the input image matrix. ‘p’ number of rows or columns are added on both sides of the matrix so the total padding becomes 2p. There are three different kinds of padding:
* Valid padding: Also known as no padding. In this specific case, the last convolution is dropped if the dimensions do not align.
* Same padding: This padding ensures that the output layer has the exact same size as the input layer.
* Full padding: This kind of padding increases the size of the output by adding zeros to the borders of the input matrix.
* Stride is by how many steps the filter is shifting on the image. If it shift by 1 step, stride value is 1.
* Resultant image size is then calculated as follows: n-f+1 +2p
* Example: When filter of 3x3 applied on 6x6 image. Resultant image is of 4x4 size which is reducing the size of the picture. So padding p=1 done to increase size to 6+2=8 to get resultant picture of 6x6 size only.
* Pooling is also one operation done in CNN. It is done to down sample the input size of the feature map. There are two types of pooling:
* Max pooling: where a window of 2x2 or 3x3 is taken and in that window, maximum value of the input is taken and the input matrix of that window size is replaced by that max value. This reduces the size of the input and also represents it by the max value , preserving the most prominent feature value also. Used more often.
* Mean pooling: in this the mean value is used for replacing the entire input of the given window size.
* Fully connected layers: connects all flattened layers to form convolved NN.

Example of how bird is recognised with CNN:





Here’s how exactly CNN recognizes a bird:

1. The pixels from the image are fed to the convolutional layer that performs the convolution operation. It results in a convolved map
2. The convolved map is applied to a ReLU function to generate a rectified feature map .
3. The image is processed with multiple convolutions and ReLU layers for locating the features .
4. Different pooling layers with various filters are used to identify specific parts of the image.
5. The pooled feature map is flattened and fed to a fully connected layer to get the final output.

**Loss functions**

1. Mean squared error: It is used to measure the average squared deviation of predicted values and actual values. It gives more weightage to large errors since we square the error.

½(y-y’)2 this is the loss function calculated for 1 sample. When we calculate for n samples it becomes cost function : 2

Advantages: it is differentiable. It has only 1 local or global minima. It converges.

Disadvantage: not robust to outliers. Changes the model a lot due to outliers.



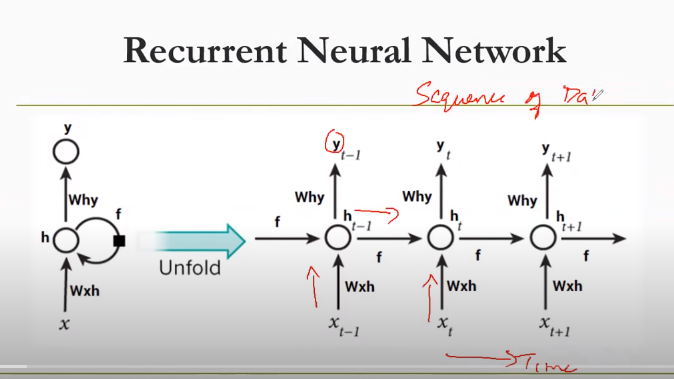
1. Mean absolute error: ½ |y-y’|

Cost function:

It is robust to outliers since it doesn’t square the error.

There will be a minor shift in the regression line/model due to outliers since the error is not squared and only absolute value is taken.

**Recurrent neural network**



* Let an Input be: <x1x2x3x4x5x6>
* Suppose at t1-> x1 is given, the output for x1 is y1, is again fed to same n/w along with x2 at time t2.
* Y1= fn (wt\*x1)
* Y2= fn (wt\*x2+ ht-1\*y1)
* Since for all the outputs, are taken into consideration for the next time stamp, RNN is used for long sequence data and data prediction. Since it takes the output of all time stamps into consideration. It is called recurrent since the output is again fed to the same network again.

**Problem with RNN:**

1. It faces vanishing and exponential gradient problem. Due to which it is difficult to solve long term dependencies. Since if in between the gradient becomes zero (vanishing gradient) , the values of the weights will stop updating. Also if we want to see the dependency between x1 and x4 and in between gradient becomes zero we will not be able to solve long sequence dependencies.
2. Context doesn’t get captured in RNN. They normally learn the context from the words coming before it, whereas to understand the context we need to check words coming after that as well. Even bidirectional RNNs (BiLSTM) learn left to right and right to left separately and **then concatenate them, which doesn’t capture the contextual meaning truly**. Whereas transformers take i/p in parallel and gives output also in parallel way.
3. Rnns are slow to train since the inputs get processed one at time. More long / large data requires more memory.
4. Inputs to RNN/LTSM are given at by one and outputs are also generated one by one (sequentially).

**Why does vanishing exploding gradients happen and how to prevent it?**

* Vanishing or exploding gradients in CNNs can be caused by several factors, such as the activation functions used. Sigmoid or tanh activation functions have a narrow range of output values, and their derivatives approach zero for large or small inputs, which can cause gradients to shrink or vanish. ReLU activation functions have a large range of output values, and their derivatives are either zero or one for any input, which can cause the gradients to explode or vanish depending on the sign of the input. The weight initialization can also affect the magnitude and variance of the gradients. If the weights are too small, the outputs of the neurons will be close to zero and the gradients will vanish. If the weights are too large, the outputs of the neurons will be far from zero and the gradients will explode. The deeper the network, the more layers the gradients have to pass through, making them more likely to vanish or explode due to multiplication of small or large values.
* Keep track of the loss function values during training. Sudden increases or decreases in loss values may indicate exploding or vanishing gradients' presence.
* Consider using activation functions like ReLU or variants such as Leaky ReLU, which are less prone to vanishing gradients compared to traditional sigmoid or tanh functions. Advanced optimization algorithms like Adam or RMSprop, which adaptively adjust learning rates based on the magnitude of gradients, facilitating more stable and efficient training.

**CNN Architectures:**

**VGG16:**

VGG16 is object detection and classification algorithm which is able to classify 1000 images of 1000 different categories with 92.7% accuracy. It is one of the popular algorithms for image classification and is easy to use with transfer learning.

**Architecture:**

* The 16 in VGG16 refers to 16 layers that have weights. In VGG16 there are thirteen convolutional layers, five Max Pooling layers, and three Dense layers which sum up to 21 layers but it has only sixteen weight layers i.e., learnable parameters layer.
* VGG16 takes input tensor size as 224, 244 with 3 RGB channel
* Most unique thing about VGG16 is that instead of having a large number of hyper-parameters they focused on having convolution layers of 3x3 filter with stride 1 and always used the same padding and maxpool layer of 2x2 filter of stride 2.
* The convolution and max pool layers are consistently arranged throughout the whole architecture
* Conv-1 Layer has 64 number of filters, Conv-2 has 128 filters, Conv-3 has 256 filters, Conv 4 and Conv 5 has 512 filters.
* Three Fully-Connected (FC) layers follow a stack of convolutional layers: the first two have 4096 channels each, the third performs 1000-way ILSVRC classification and thus contains 1000 channels (one for each class). The final layer is the soft-max layer.

All the previous models used deep neural networks in which they stacked many convolution layers one after the other. It was learnt that deeper networks are performing better. However, it turned out that this is not really true. Following are the problems with deeper networks:

* Network becomes difficult to optimize: with a lot of convolution layers, the convergence does not happen. Problems like Vanishing / Exploding Gradients happen due to which weights do not converge.
* Vanishing / Exploding Gradients: due to large no. of layers, the value of gradient becomes very small and weights do not get updated during back propagation. Or the value of gradient explodes due to which it has to be trimmed or scaled down.
* Degradation Problem (accuracy first saturates and then degrades)