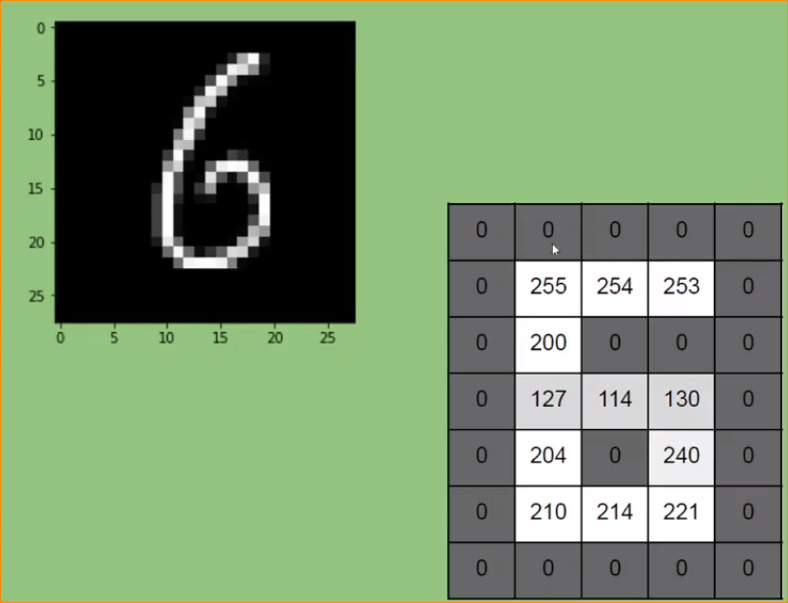
* **Taking Convolution Neural Network**

1. Computer Vision – Can we use for

* Image Classification – try to separate images
* Object Detection – Identify the object in the image
* Semantic Segmentation- Identify the reigns of the image (pixel wise recognition)
* Action Recognition- If you give the video it can recognize the action (cooking, Eating)
* Clustering- Grouping (Base on image similarity)
* 3D reconstruction – Construct missing part of images (Incomplete image)

1. What is an image?

An image is a visual representation or depiction of an object, scene, person, or concept. In the context of computing and digital technology, an image is often a two-dimensional array of pixels, where each pixel represents a tiny element of the picture. These pixels contain information about color and intensity, allowing the image to be displayed on a screen or printed on paper.Images can take various forms, including photographs, illustrations, paintings, charts, graphs, or any other visual content.



This is a blur image of number 6. In the pixel from can represent as a Metrix.

Black – 0

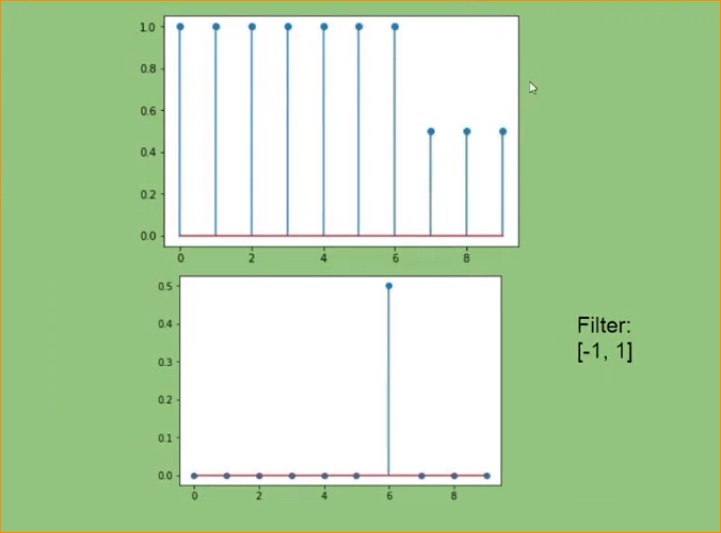
Other – light value

1. What is Convolution?

Convolution is a mathematical operation that combines two functions to produce a third function, representing how one function modifies the other. In the context of signal processing and image analysis, convolution is often used to filter or process data. Convolution plays a significant role in convolutional neural networks (CNNs), a type of deep learning architecture commonly used for image-related tasks.

Simple Represent of Convolution –

Signal 01



Input one dimension signal

Another signal call Filter.

So We can Addition or multiplication operation.

Signal 02

How its happen?

1 x 1 + (-1) x 1 = 0

…………

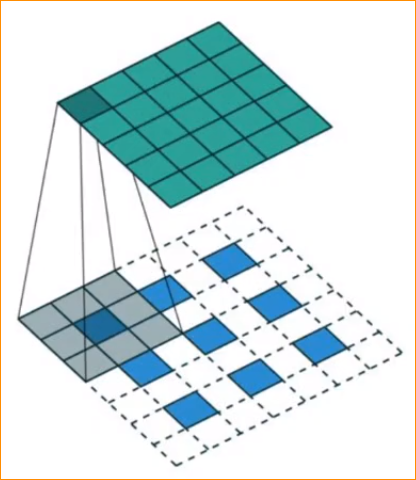
1 x 1 + (-1) x 0.5 = 0.5

In here If there we have changing the input signal, that we get a spick.

That is important property that we can use in image property. Because image is also signal.

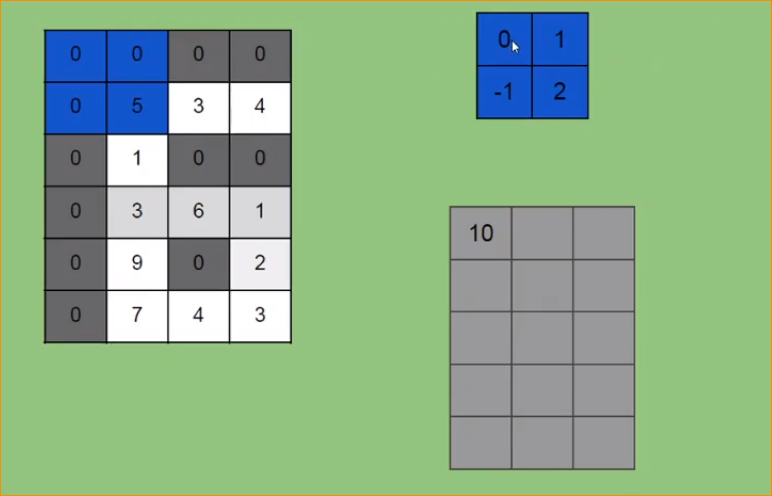
1. Apply Same principle to the 2D- (Like image) Apply Convolution filter to that.

It gives Spikes – ages o0f image



In here there color separation, intensity.

Example –



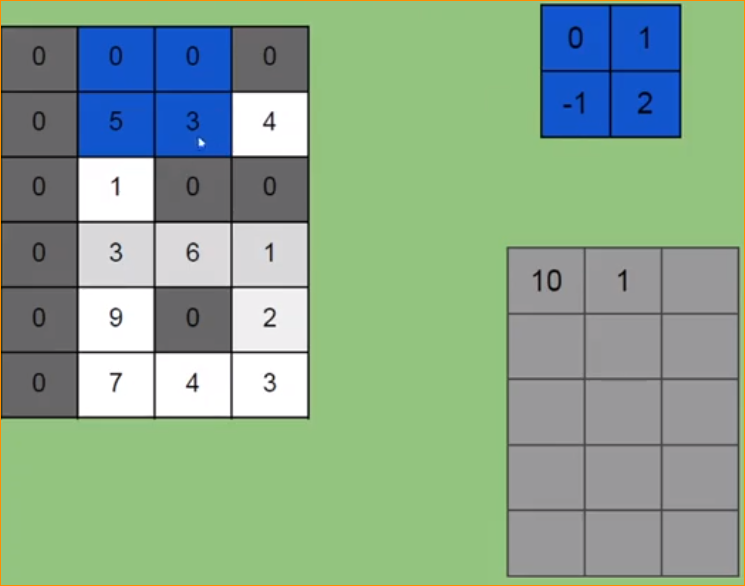
Input image

Filter

In here,

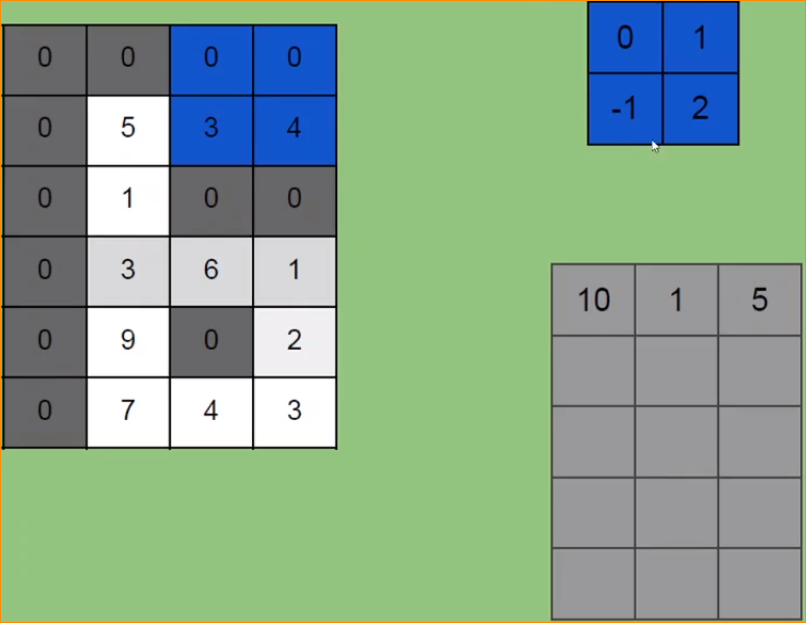
0x0 + 1x0 + (-1)x0 + 2x5 = 10

Move One step –



0x0 + 1x0 +(-1)x5 +2x3 = 1

Another one –

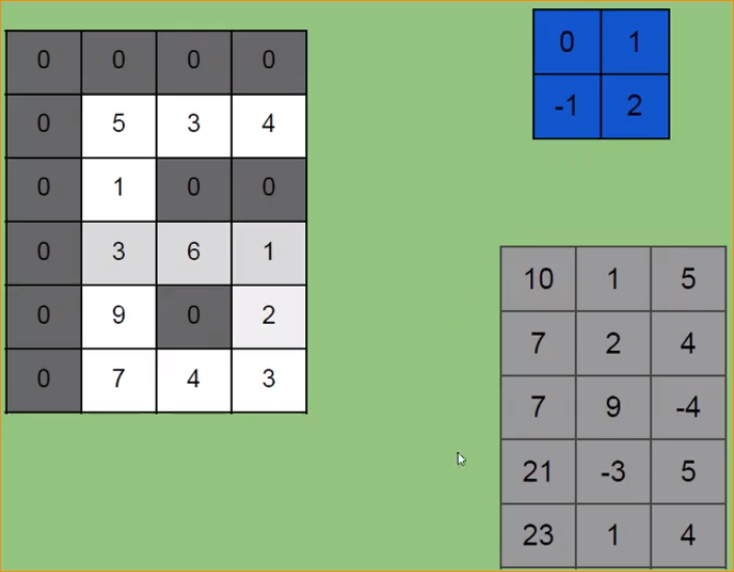


0x0 + 1x0 +(-1)x3 +2x4 = (-3) + 8 = 5

If want we want to add Additional zeros that’s call **padding**

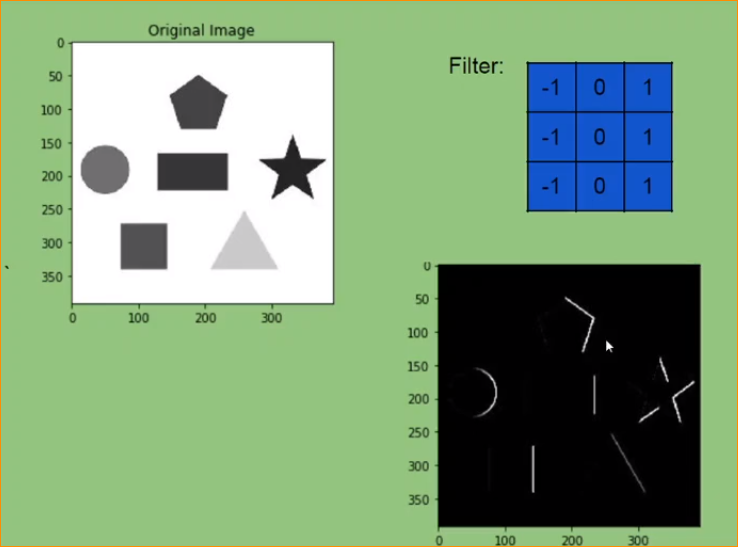
.

Final Result-



When You apply convolution operation, can get ages, images.

Use Special filters- and result

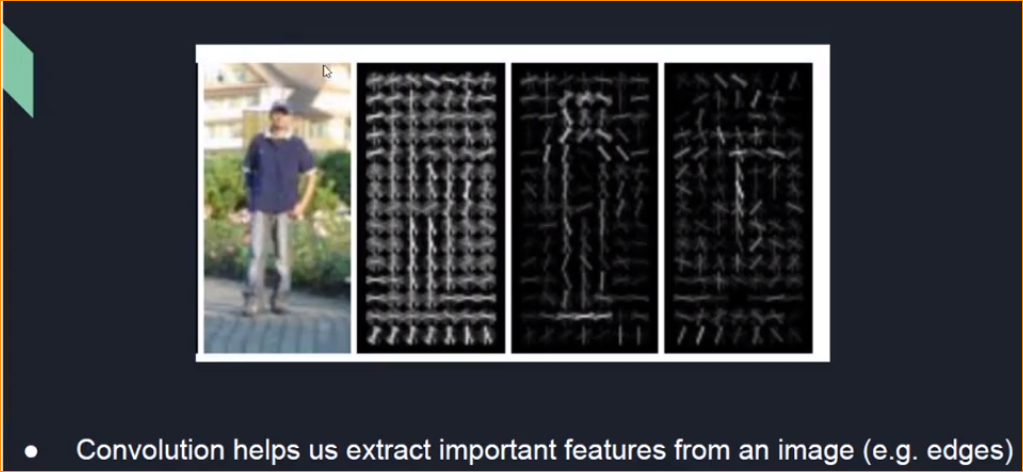
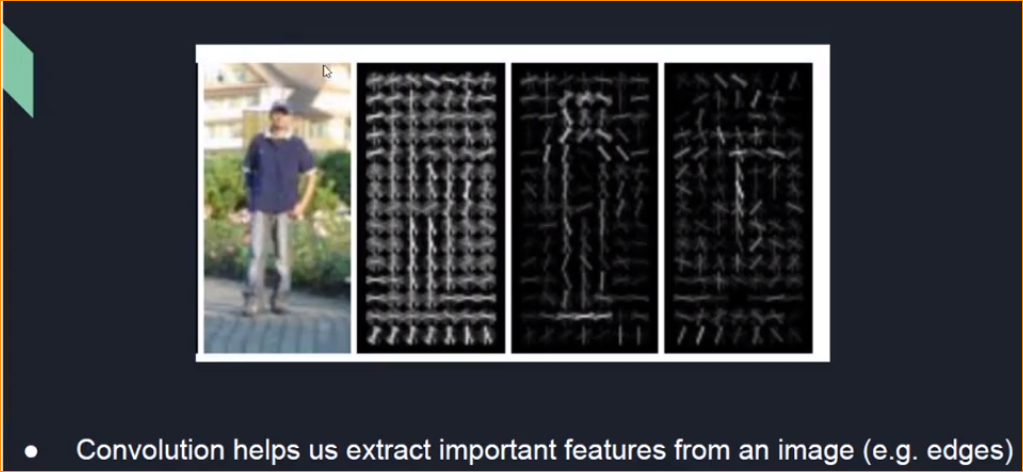
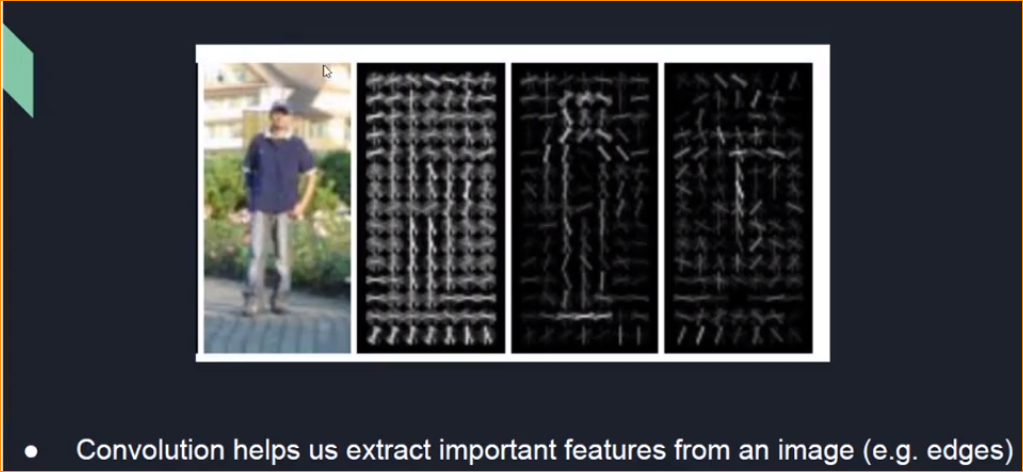


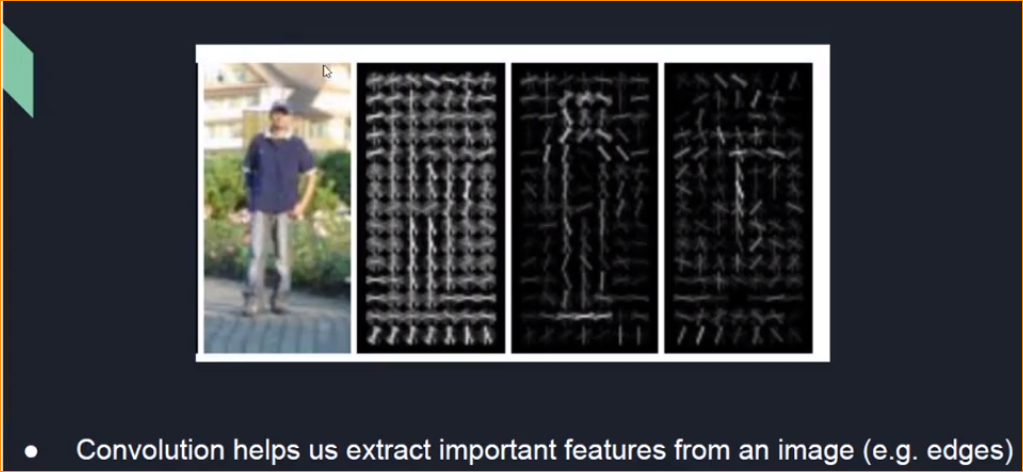
Result -

Apply this kind of filter- Can get this kind of images-

1. Complex images- (Can recognize that way)

With Ages we can Classified Images

Apply CNN Filter with set of value



1. CNN – Key Idea

Different types of filters -  


Even more powerful and Flexible feature.

Use Weights

Extracts vertical edges

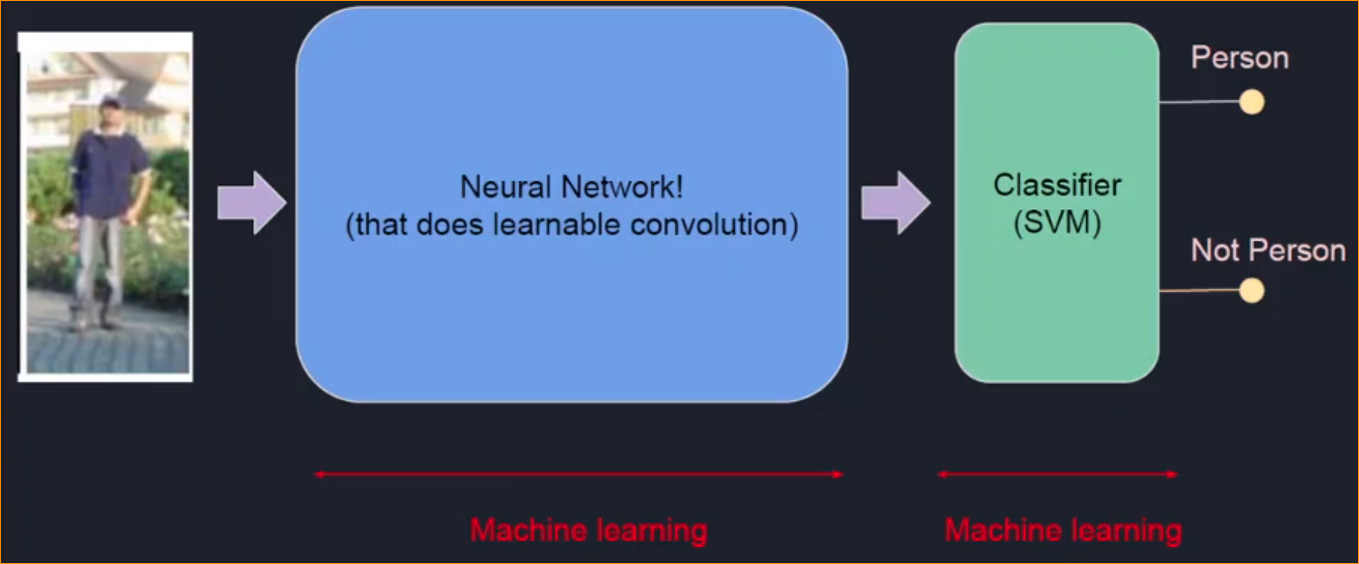
Extracts horizontal edges

If we can understand about weight, we can extract the corresponding image.

Using ML to image classification -

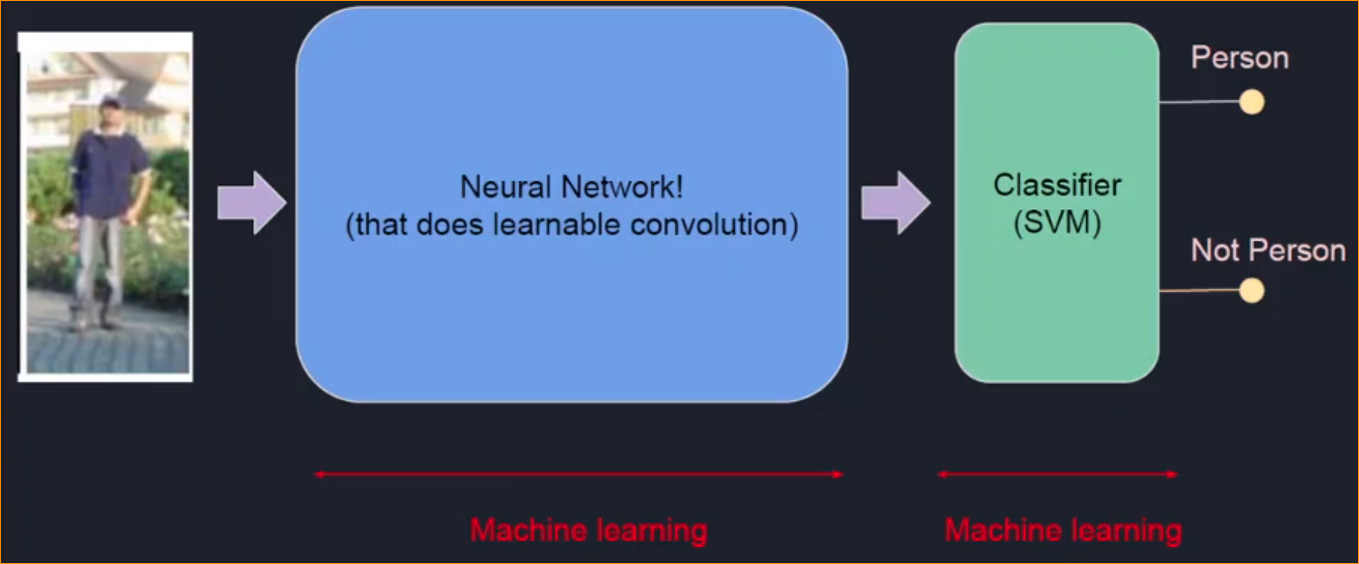


ML

Feature extraction

In here image extract using filters and using old ML part like support vector machine input those features. And Do the classification..

CNN mean – that feature extraction part doing automatically. For that it adjusts that weights.



That those ML features as one- End to End ML

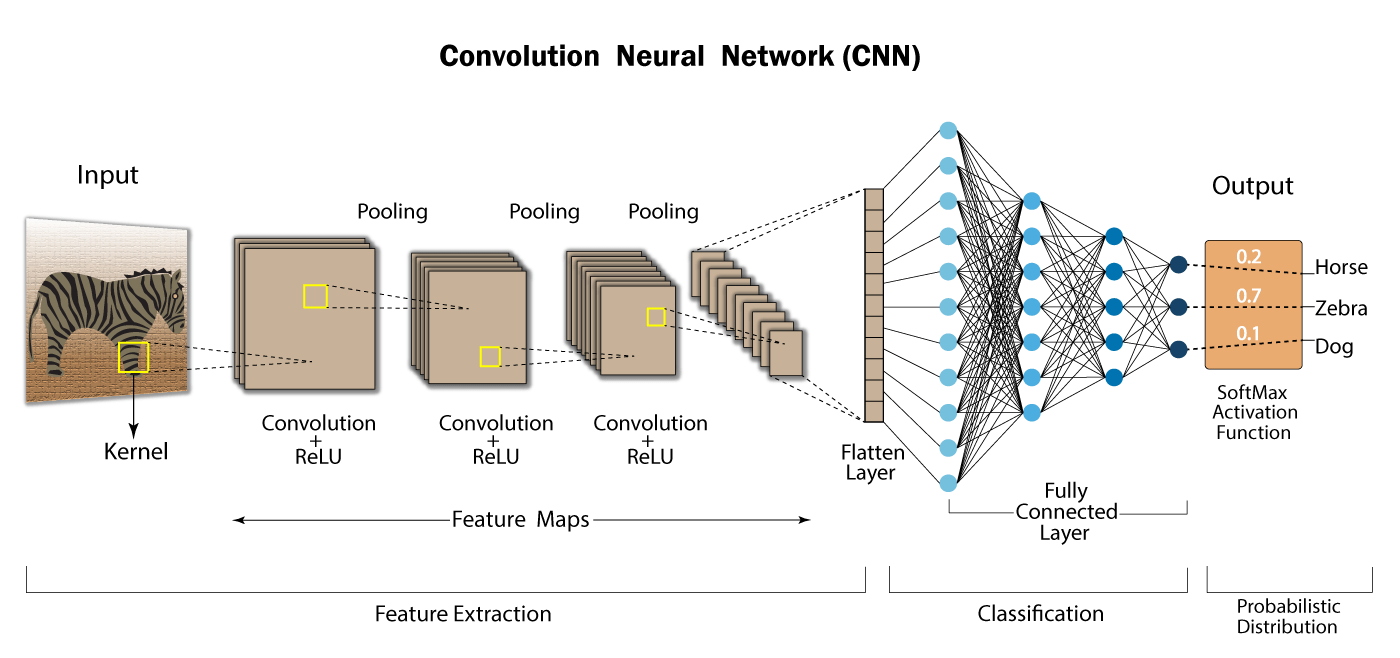


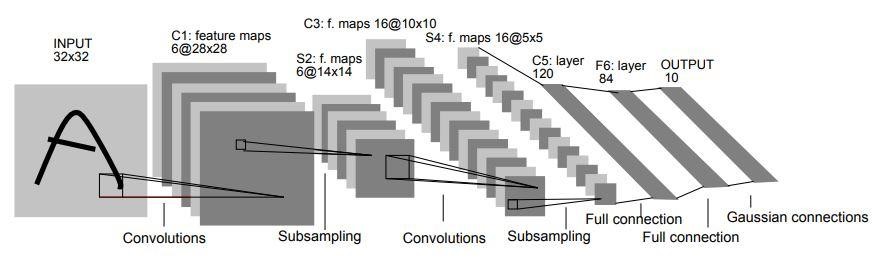
**Neural Network = feature extraction + Classification**

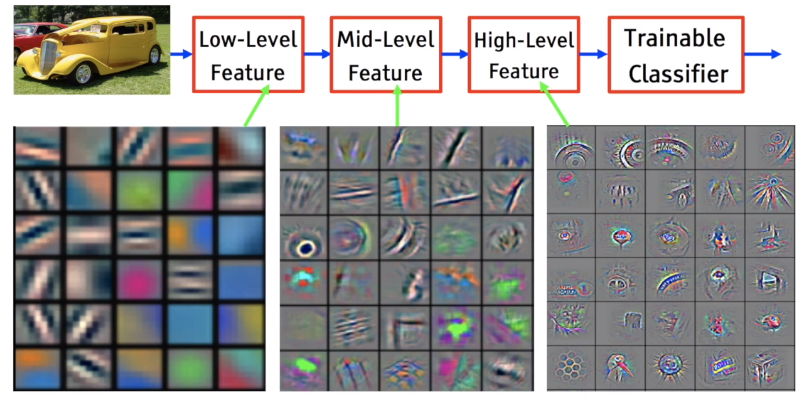
Both these happen in one Neural Network

Deep Learning – Feature Extraction is automated.

1. Anatomy of CNN : LeNet







1. What Happen in CNN ?

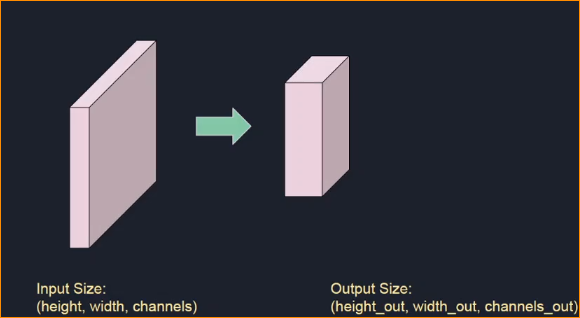
It learn features in hierarchy faction,

Low level Features- Identify basic edgers.

Midlevel , Highlevel features-

That we call hierarchy learning model

1. **Close Look CNN -**

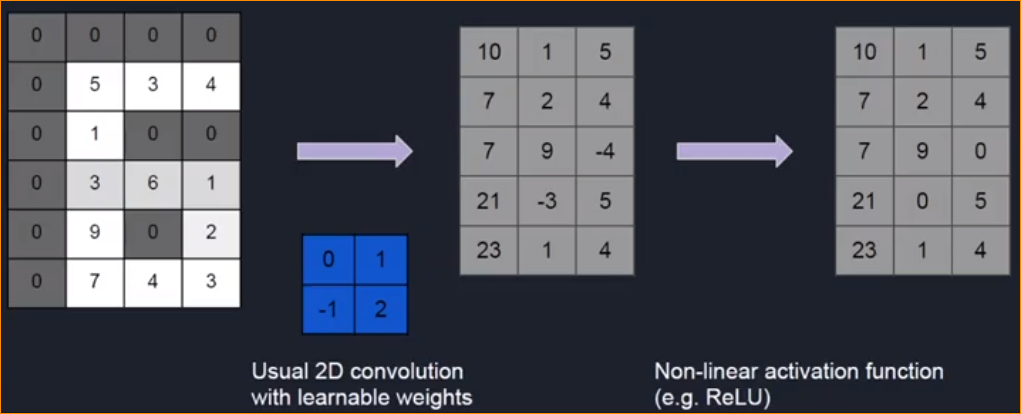


Convolution Layers –

Input Image –

In here multiple Convolutional neural network adding to the image. Generate multiple output images.

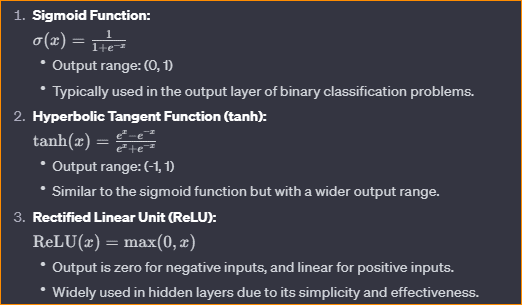
1. Activation Function

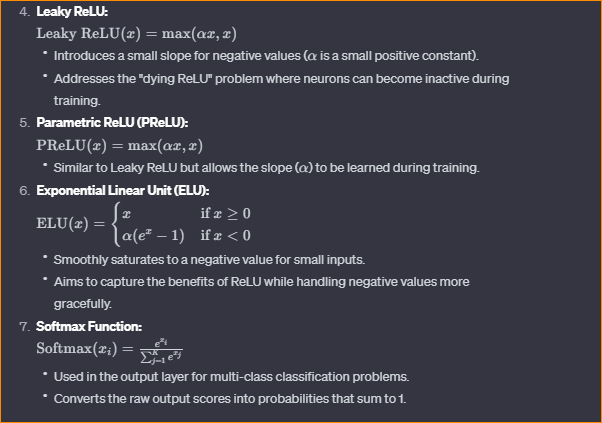


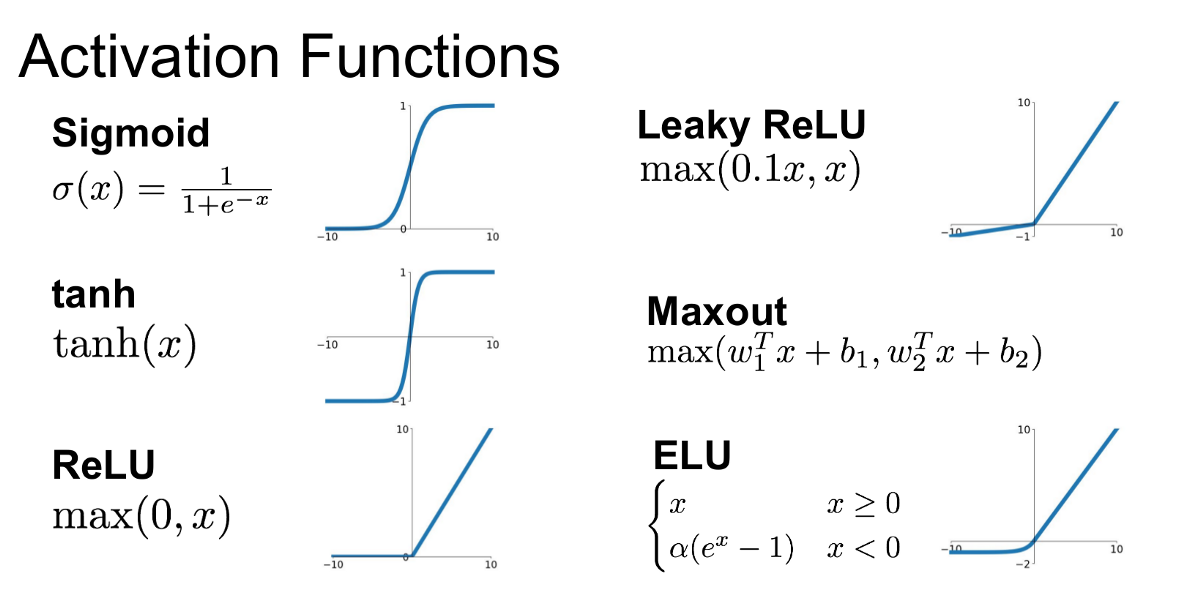
1. Step – 2D convolution
2. Step – Activation function – in here should add non-leaner functions

Note –

Activation functions are mathematical operations applied to the output of a neuron in a neural network. They introduce non-linearity to the network, allowing it to learn complex patterns and relationships in the data. Here are some common activation functions used in neural networks:

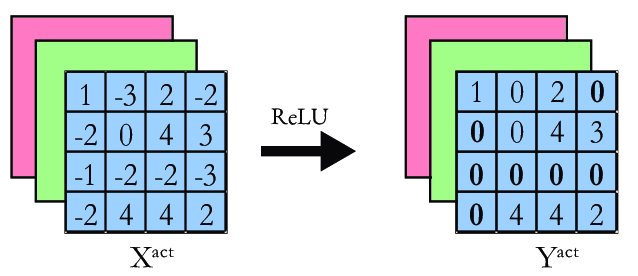






In here We use RELU Activation function,

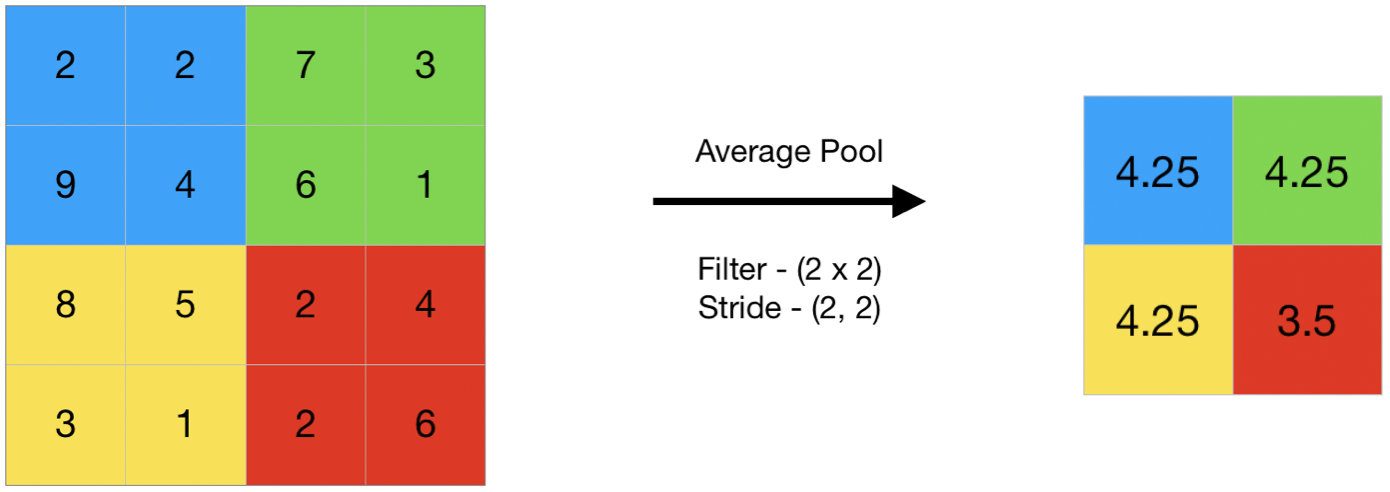
In that, if we have “ – “ values that convert to **0**



1. Pooling –

This is a Compression operation.

Compress that image gradually.



**Note** -

In Convolutional Neural Networks (CNNs), pooling is a downsampling operation applied to the feature maps produced by convolutional layers. Pooling is used to reduce the spatial dimensions of the input volume, decrease the computational complexity, and capture the most important information while maintaining the important features. The two most common types of pooling are Max Pooling and Average Pooling.

1. Max Pooling:

* Max pooling involves taking the maximum value from a group of neighboring pixels.
* It is defined by a window (pool size) and a stride.
* The window moves over the input feature map, and at each step, the maximum value within the window is selected.
* Example (2x2 max pooling with a stride of 2):

1. Average Pooling:

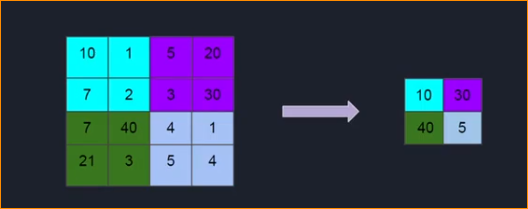
* Average pooling computes the average value within a window.
* Similar to max pooling, it is defined by a window size and a stride.
* Example (2x2 average pooling with a stride of 2):

Pooling helps to achieve several objectives in CNNs:

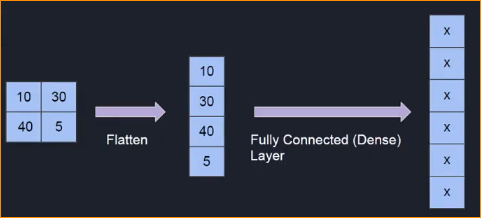
* Spatial Hierarchical Representation: By down sampling, higher-level features can capture more abstract and complex patterns.
* Translation Invariance: Pooling makes the network less sensitive to the precise spatial location of features.
* Reduced Computational Complexity: Pooling reduces the number of parameters and computations, making the network more computationally efficient.

Pooling is typically applied after convolutional layers in CNN architectures, and it is often followed by additional convolutional layers. The choice of pooling type, window size, and stride depends on the specific requirements of the task and the characteristics of the data. Max pooling is more commonly used, but the choice may vary depending on the problem and architecture.

Our CNN -



1. Fully Connected Layers



**Flatten** – After the convolutional and pooling layers, the data is in the form of a three-dimensional tensor (height, width, depth or channels). The flatten operation reshapes this tensor into a one-dimensional vector by stacking the values along a single dimension.

For example, if the tensor is of size (8,8,64)(8,8,64), the flatten operation would transform it into a vector of size (8×8×64)=4096(8×8×64)=4096.

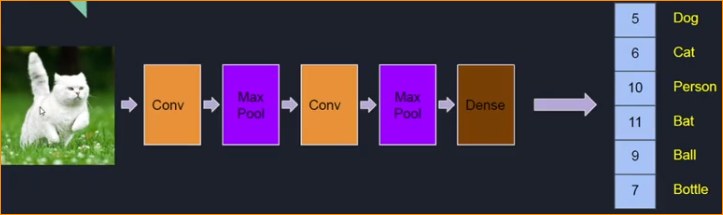
Fully Connected Dance Layer – (Similar to traditional neural network)

The flattened vector is then passed through one or more fully connected layers, which learn global patterns and relationships in the data.

This layer produce some output, some numeric value.

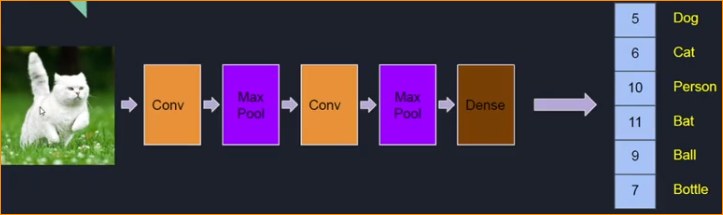
That number of numbers equal to number of output of the NN. Number of classes in the problem is equal to number of output to the layer.

Example –



Prediction

There layer doing feature extraction

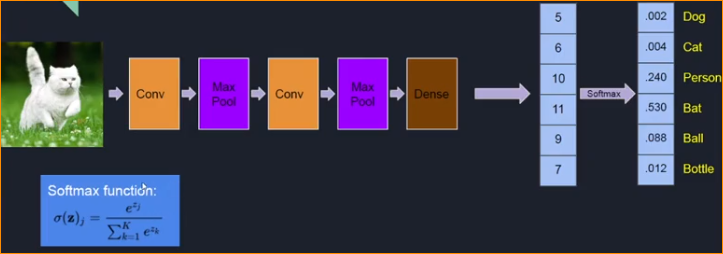


These are the classes that we have to classified into.

So Fully connected layer output the numbers base on the weigh or parameters of the convolution layer and the dense layer

In here there are two weights convolution and dense

Apply softMax function –



Using this Convert that Output in to Probability distribution.

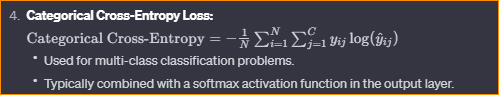
In here output is wrong that gives bat output = 0.530

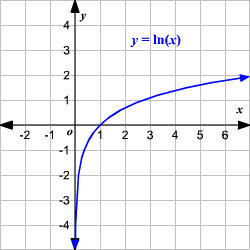
1. Loss Function

* High loss bad, Low Loss Good
* We need to encourage the network to give high probability for the cat class.
* In CNN **categorical cross entropy**

Note –

In machine learning, a loss function, also known as a cost function or objective function, is a measure of the difference between the predicted values of a model and the actual values (ground truth) in the training data. The goal during training is to minimize this loss function, which represents how well the model is performing on the task at hand.



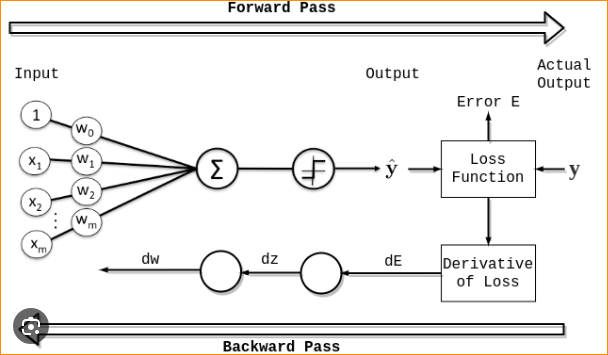


In here in INPUT(x) goes to 0 , OUTPUT goes to “-“ negative infinity .



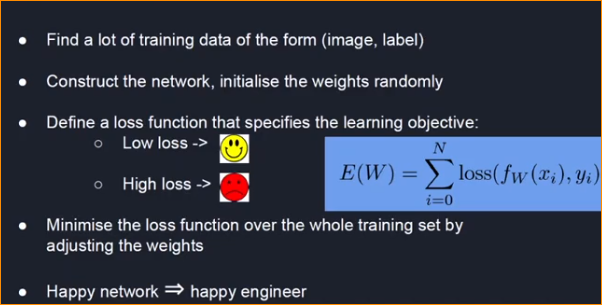
So if input near to one log going low.

Depend on loss, Can add back propagation to update weight in Convolution weight or Dense(Fully connected weight.)



1. Training A Convolution Neural network

* Find the lot of data (images or label data) – Example If you want to classified Cat or Dog images and You should Collect cat and dog images and you have to label it.

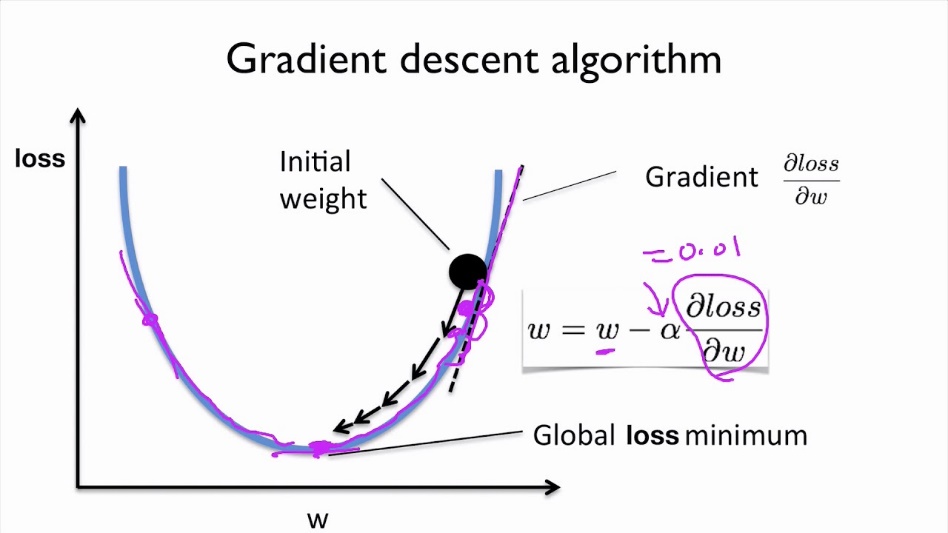


If you loss is high, need to optimize the Network until the loss is going to low. Adjust the weights and using loss and doing multiple iteration.

**Note** - **optimization algorithms**

Optimization algorithms are used to minimize or maximize an objective function by adjusting the parameters of a model. In the context of machine learning, optimization is crucial for training models, where the objective is to find the optimal set of parameters that minimizes the loss function on a given dataset. Here are some commonly used optimization algorithms:

1. Gradient Descent:



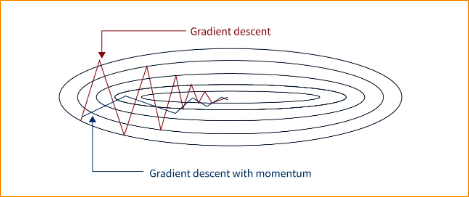
**Vanilla Gradient Descent**: Updates the parameters in the opposite direction of the gradient of the loss function with respect to the parameters.

**Stochastic Gradient Descent (SGD):** Computes the gradient and updates the parameters using a small randomly selected subset (mini-batch) of the training data at each iteration.

In here if I have 10000 images I am using, Random Images as 1000 mini batch and iteration.

**Mini-Batch Gradient Descent**: A compromise between vanilla gradient descent and SGD, where updates are made using a small, fixed-size batch of data.

1. Momentum:



Helps accelerate SGD by adding a fraction of the previous update to the current update.

It reduces oscillations and helps the optimizer navigate through shallow minima.

1. Adagrad:

Adapts the learning rates of each parameter based on the historical gradient information.

It performs smaller updates for frequently occurring parameters and larger updates for infrequent ones.

1. RMSprop:

Addresses the diminishing learning rate problem of Adagrad by using a moving average of squared gradients.

It adjusts the learning rates for each parameter independently.

1. Adam (Adaptive Moment Estimation):

Combines the ideas of momentum and RMSprop.

Maintains a moving average of gradients and squared gradients for each parameter.

Adaptive learning rates for each parameter.

1. Nadam:

An extension of Adam that incorporates Nesterov accelerated gradient (NAG) into its update rule.

NAG involves looking ahead of the current position before calculating the gradient.

1. Adadelta:

Similar to RMSprop but uses a more sophisticated update rule by maintaining a running average of parameter updates.

It eliminates the need for a global learning rate.

1. L-BFGS (Limited-memory Broyden-Fletcher-Goldfarb-Shanno):

An optimization algorithm that belongs to the family of quasi-Newton methods.

It approximates the inverse Hessian matrix to make updates to the parameters.

1. SGD with Nesterov Accelerated Gradient (NAG):

A modification of SGD that incorporates momentum with Nesterov's momentum term.

It reduces the oscillations observed in vanilla SGD.

**Summary –**

