In cases of extremely basic binary images, the method might show an average precision score while performing prediction of classes but would have little to no accuracy when it comes to complex images having pixel dependencies throughout.

A ConvNet is able to **successfully capture the Spatial and Temporal dependencies** in an image through the application of relevant filters. The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved and reusability of weights.

The role of the ConvNet is to reduce the images into a form which is easier to process, without losing features which are critical for getting a good prediction. This is important when we are to design an architecture which is not only good at learning features but also is scalable to massive datasets.

Convolution Layer — The Kernel

The element involved in carrying out the convolution operation in the first part of a Convolutional Layer is called the **Kernel/Filter**

The objective of the Convolution Operation is to extract the high-level features such as edges, from the input image. ConvNets need not be limited to only one Convolutional Layer. Conventionally, the first ConvLayer is responsible for capturing the Low-Level features such as edges, color, gradient orientation, etc. With added layers, the architecture adapts to the High-Level features as well, giving us a network which has the wholesome understanding of images in the dataset, similar to how we would.

Pooling Layer

Pooling layer is responsible for reducing the spatial size of the Convolved Feature. This is to decrease the computational power required to process the data through dimensionality reduction. Furthermore, it is useful for extracting dominant features which are rotational and positional invariant, thus maintaining the process of effectively training the model.

Classification — Fully Connected Layer (FC Layer)

Adding a Fully-Connected layer is a (usually) cheap way of learning non-linear combinations of the high-level features as represented by the output of the convolutional layer. The Fully-Connected layer is learning a possibly non-linear function in that space.

we shall flatten the image into a column vector. The flattened output is fed to a feed-forward neural network and backpropagation applied to every iteration of training. Over a series of epochs, the model is able to distinguish between dominating and certain low-level features in images and classify them using the Softmax Classification technique.

VGGNet

Why? VGGNet was born out of the need to reduce the # of parameters in the CONV layers and improve on training time.

VGG16 has a total of 138 million parameters. The important point to note here is that all the conv kernels are of size 3x3 and maxpool kernels are of size 2x2 with a stride of two.

How? The idea behind having fixed size kernels is that all the variable size convolutional kernels used in Alexnet (11x11, 5x5, 3x3) can be replicated by making use of multiple 3x3 kernels as building blocks.

Now let's look at the number of variables needed to be trained. For a 5x5 conv layer filter, the number of variables is 25. On the other hand, two conv layers of kernel size 3x3 have a total of 3x3x2=18 variables (a reduction of 28%).

A reduced number of trainable variables means faster learning and more robust to overfitting.

RESNet

Why? Neural Networks are notorious for not being able to find a simpler mapping when it exists.

As per what we have seen so far, increasing the depth should increase the accuracy of the network, as long as overfitting is taken care of. But the problem with increased depth is that the signal required to change the weights, which arises from the end of the network by comparing ground-truth and prediction becomes very small at the earlier layers, because of increased depth. It essentially means that earlier layers are almost negligible learned. This is called the vanishing gradient. The second problem with training the deeper networks is, performing the optimization on huge parameter space and therefore naively adding the layers leading to higher training error. Residual networks allow training of such deep networks by constructing the network through modules called residual models as shown in the figure. This is called the degradation problem.

To solve this, the module shown above creates a direct path between the input and output to the module implying an identity mapping and the added layer-C just need to learn the features on top of already available input. Since C is learning only the residual, the whole module is called residual module.

This method of bypassing the data from one layer to another is called as shortcut connections or skip connections. This approach allows the data to flow easily between the layers without hampering the learning ability of the deep learning model. The advantage of adding this type of skip connection is that if any layer hurts the performance of the model, it will be skipped.

The intuition behind the skip connection is that it is easier for the network to learn to convert the value of f(x) to zero so that it behaves like an identity function rather than learning to behave like an identity function altogether on its own by trying to find the right set of values that would give you the result.

Inception:

In an image classification task, the size of the salient feature can considerably vary within the image frame. Hence, deciding on a fixed kernel size is rather difficult. Lager kernels are preferred for more global features that are distributed over a large area of the image, on the other hand, smaller kernels provide good results in detecting area-specific features that are distributed across the image frame. For effective recognition of such a variable-sized feature, we need kernels of different sizes. That is what Inception does. Instead of simply going deeper in terms of the number of layers, it goes wider. Multiple kernels of different sizes are implemented within the same layer.

Decision Tree:

It is a tree-structured classifier, where **internal nodes represent the features of a dataset, branches represent the decision rules** and **each leaf node represents the outcome.**

In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.

how to select the best attribute for the root node and for sub-nodes?

Entropy,

Information gain,

Gini index

Entropy

Entropy is a measure of the randomness in the information being processed. The higher the entropy, the harder it is to draw any conclusions from that information.

Information Gain

Information gain is a statistical property that measures how well a given attribute separates the training examples according to their target classification. Constructing a decision tree is all about finding an attribute that returns the highest information gain and the smallest entropy.

Higher value of Gini index implies higher inequality, higher heterogeneity.

1-sigma(pi^2)

drawbacks of the Decision Tree algorithm

- Small changes to training data can result in a significantly different tree structure.
- It may have the problem of overfitting

So, instead of training a single decision tree, it is better to train a group of decision trees which together make a random forest.

A random forest consists of a group (an ensemble) of individual decision trees. Therefore, the technique is called *Ensemble Learning*. A large group of *uncorrelated* decision trees can produce more accurate and stable results than any of individual decision trees.

When you train a random forest for a classification task, you actually train a group of decision trees. Then you obtain the predictions of all the individual trees and predict the class that gets the most votes. Although some individual trees produce wrong predictions, many can produce accurate predictions. As a group, they can move towards accurate predictions.

In a normal decision tree, the algorithm searches the very best feature out of *all the features* when it wants to split a node. In contrast, each tree in a random forest searches for the very best feature out of a *random subset of features*. This creates extra randomness when growing the trees inside a random forest.

In a random forest, each decision tree is trained on a different random sample of the training set. When sampling is done with replacement, the method is called bagging. The bootstrap method reduces the correlation between decision trees.

We make a few assumptions when we use linear regression to model the relationship between a response and a predictor. These assumptions are essentially conditions that should be met before we draw inferences regarding the model estimates or before we use a model to make a prediction.

- The true relationship is linear: relations between the independent and dependent variables must be linear.
- Errors are normally distributed
- Homoscedasticity of errors (or, equal variance around the line).
- Independence of the observations