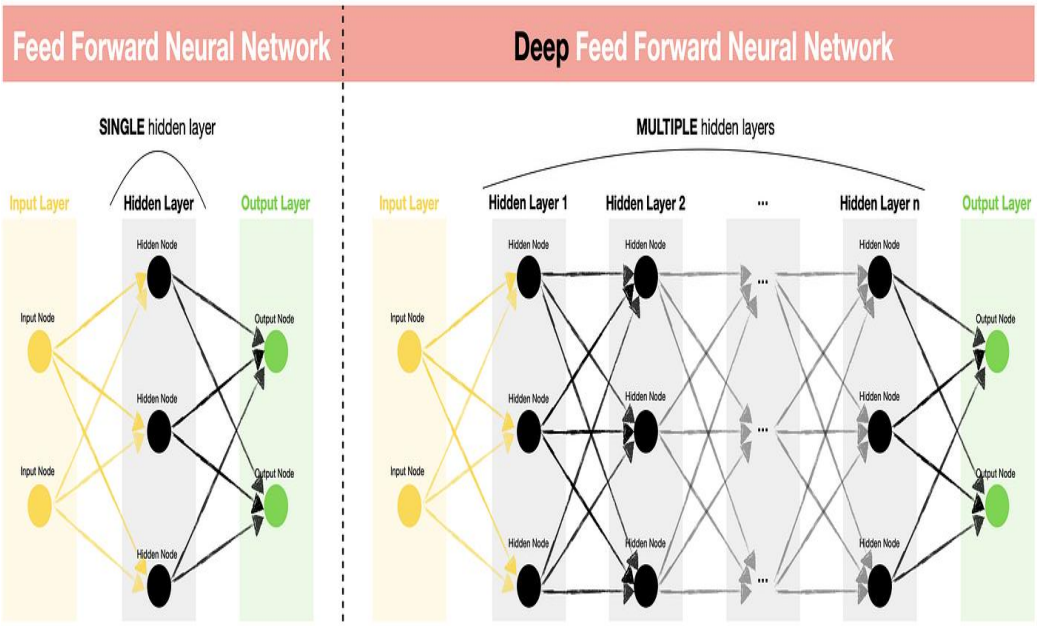
UNIT - III: Deep Feed Forward network

Syllabus:

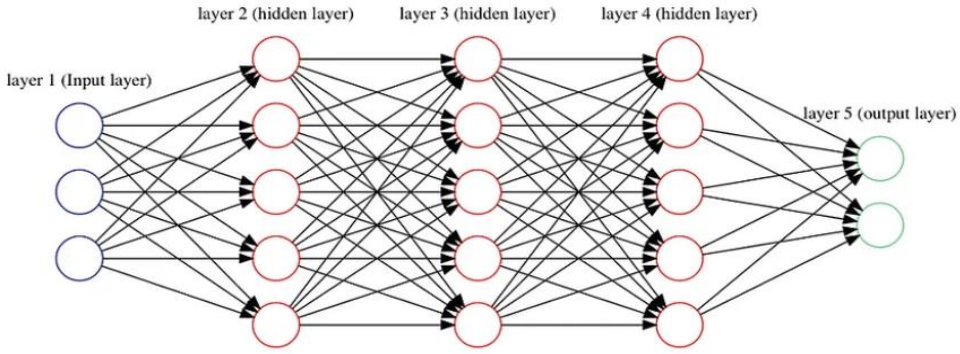
Deep Feed Forward network, regularizations, training deep models, dropouts, Convolution Neural Network, Recurrent Neural Network, and Deep Belief Network.

A deep feed-forward network (DFN) is a type of neural network that processes data by passing it forward through layers, without using feedback. DFNs are also known as multi-layer perceptron's (MLPs) or simply neural networks.

- The structure of a **Deep Feed Forward (**DFF) is very similar to that of a **Feed Forward (**FF).
- The major difference between them is the number of hidden layers.
- Currently, Neural Network with one hidden layer is a "shallow" network or simply a Feed-Forward network.



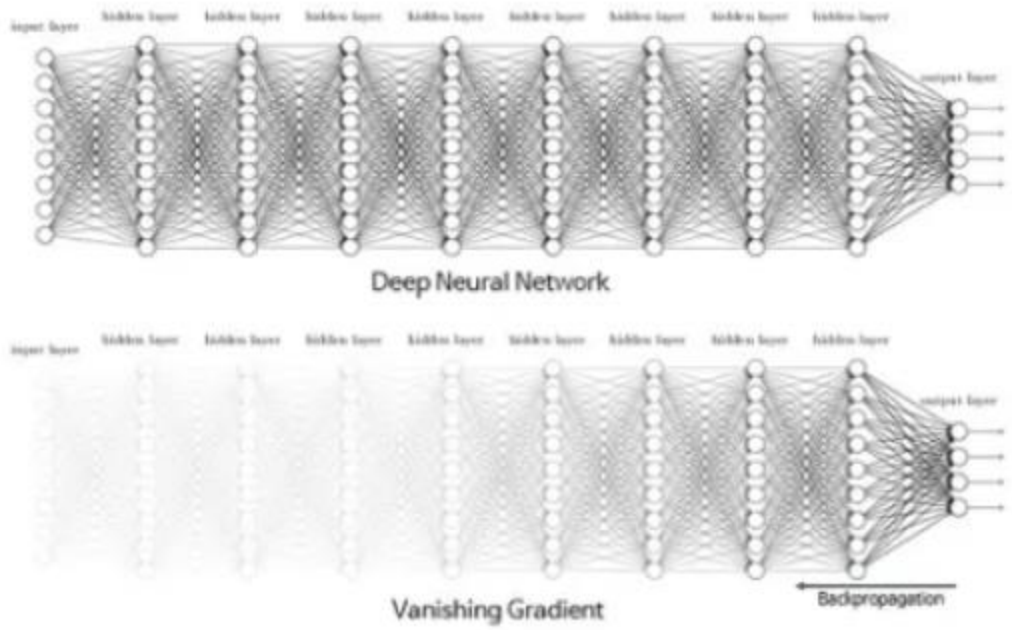
- Meanwhile, a Neural Network with multiple hidden layers (2+) is called a Deep network.



- Typically, you will find that deep NNs perform better than shallow ones.
- However, it is not always necessary to use a deep network, the choice will largely depend on the data.
- If you are working with many inputs, such as image data, then using a Deep Feed Forward (DFF) or a Convolutional Neural Network (CNN) would likely yield better results than a simple Feed Forward network.

Challenges presented by deep networks

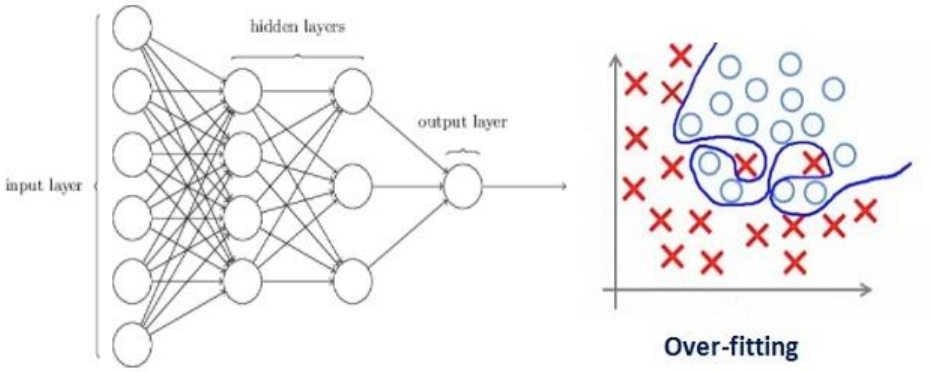
- A Neural Network architecture contains **activation functions** inside the hidden nodes and output nodes.
- Traditionally, the two most widely used nonlinear activation functions were the sigmoid and the hyperbolic tangent (tanh).
- However, using these activation functions with Deep Neural Networks presented a problem of **vanishing gradient**.



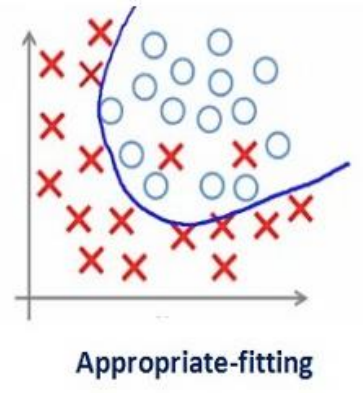
- The error is backpropagated through the network during the training process and is used to update the weights.
- Unfortunately, sigmoid and tanh activation functions tend to **saturate**.
- It means that large negative and large positive values are transformed to 0 and 1 by sigmoid and -1 and 1 by tanh.
- As the functions saturate, the derivate becomes close to zero.
- Hence, there is essentially **no gradient** left to propagate back through the network, making it challenging for the learning algorithm to continue adapting the weights.
- **ReLU** has been introduced as a solution to a vanishing gradient problem and quickly became a default option for most Deep Feed Forward (DFF) and Convolutional Neural Networks (CNN).
- It has a very simple function that sets all negative values to 0 while returning the **same value** for all positive inputs.

Regularizations

- Regularization is a technique which makes slight modifications to the learning algorithm such that the model generalizes better.
- Regularization is a technique used in machine learning and deep learning to prevent **overfitting** and improve the generalization performance of a model.
- Let's consider a neural network which is overfitting on the training data.



- We need to optimize the value of regularization coefficient in order to obtain a well-fitted model



- It involves adding a penalty term to the loss function during training.
- This penalty discourages the model from becoming too complex or having large parameter values, which helps in controlling the model's ability to fit noise in the training data.
- By applying regularization, models become more robust and better at making accurate predictions on unseen data.

Different Regularization Techniques in Deep Learning:

- L2 & L1 regularization
- Dropout
- Data Augmentation
- Early stopping

L^1 Parameter Regularization:

- L^1 regularization is a method of doing regularization

$$Loss = Error(y, \hat{y}) + \lambda \sum_{i=1}^{N} |w_i|$$

- The lambda parameter **controls the amount of regularization applied to the model.**

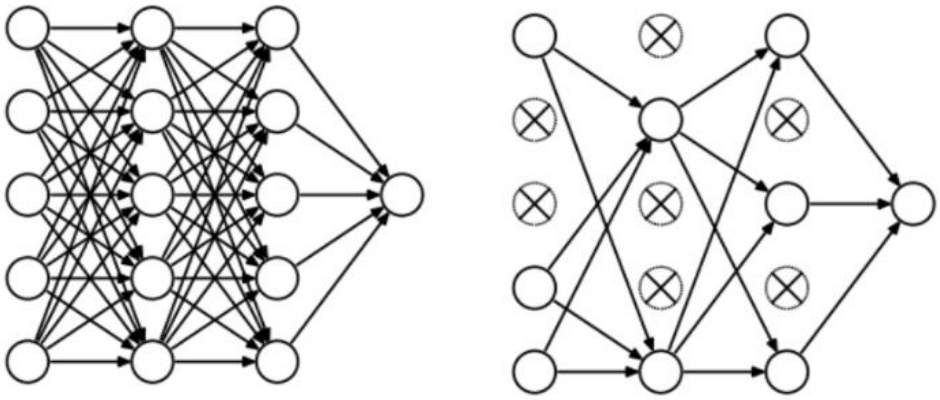
L^2regularization: Here, **lambda** is the regularization parameter. It is the hyperparameter whose value is optimized for better results.

- L^2 regularization is also known as *weight decay* as it forces the weights to decay towards zero (but not exactly zero).

$$Cost\,function = Loss + \frac{\lambda}{2m} * \sum \|w\|^2$$

Dropout

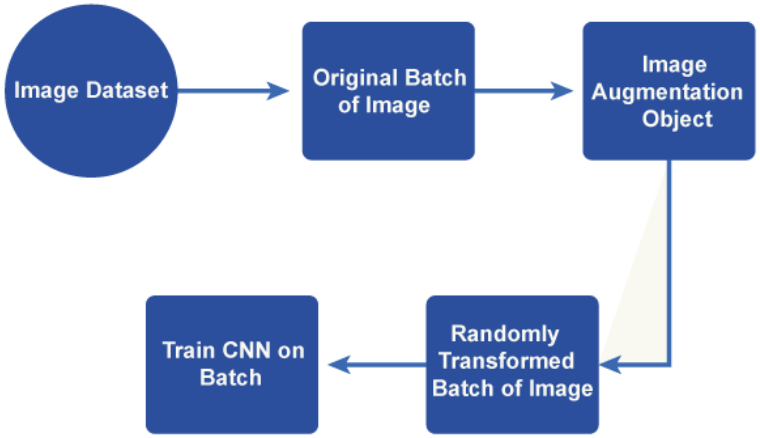
- It also produces very good results and is consequently the most frequently used regularization technique in the field of deep learning.
- At every iteration, it randomly selects some nodes and removes them along with all of their incoming and outgoing connections as shown below.



- So, each iteration has a different set of nodes and this results in a different set of outputs.
- This probability of choosing how many nodes should be dropped is the hyperparameter of the dropout function.

Data augmentation:

- It is the process of increasing the amount of data.
- Data augmentation can be especially beneficial when the original set of data is small as it enables the system to learn from a larger and more varied group of samples.
- We do not collect new data, rather we transform the already present data.



- Operations in data augmentation are-

❖ Rotation

❖ Shearing(similar to concept of shear modulus in physics)

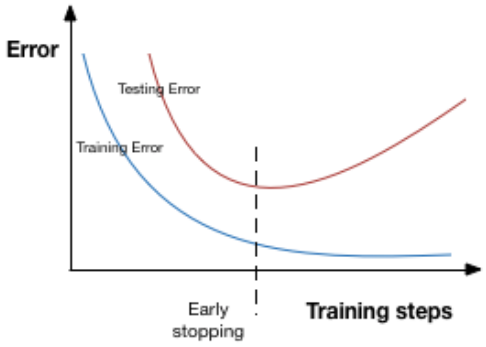
❖ Zooming

❖ Cropping

❖ Changing the brightness level

Early stopping

- Early stopping is a kind of cross-validation strategy where we keep one part of the training set as the validation set.
- When we see that the performance on the validation set is getting worse, we immediately stop the training on the model.
- This is known as early stopping.



Training Deep Neural Networks

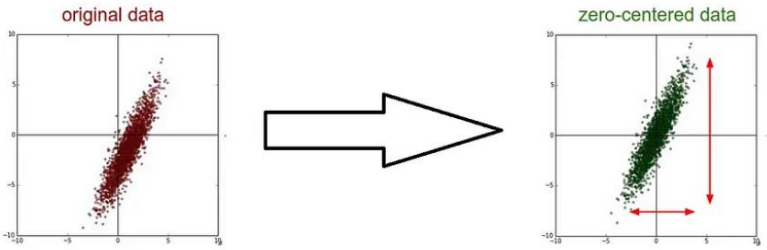
- Deep neural networks are key in the field of computer vision and speech recognition.
- To achieve high level of accuracy, huge amount of data and computing power is needed to train these networks.
- However, we can follow certain guidelines to reduce the time for training and improve model accuracy.

Data Pre processing

- If important data inputs are missing, neural network may not be able to achieve desired level of accuracy.

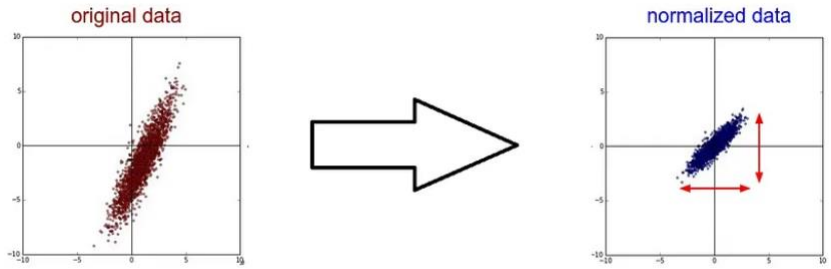
Mean subtraction (Zero centering)

- It's the process of subtracting mean from each of the data point to make it zero-centered.



Data Normalization

- *Normalization* refers to normalizing the data to make it of same scale across all dimensions.
- Common way to do that is to divide the data across each dimension by its **standard deviation.**



Parameter Initialization

- *Deep neural networks* are strangers to millions or billions of parameters.
- The way these parameters are initialized can determine how fast our learning algorithm would converge and how accurate it might end up.
- we can do better by initializing the weights with some *small random numbers*.

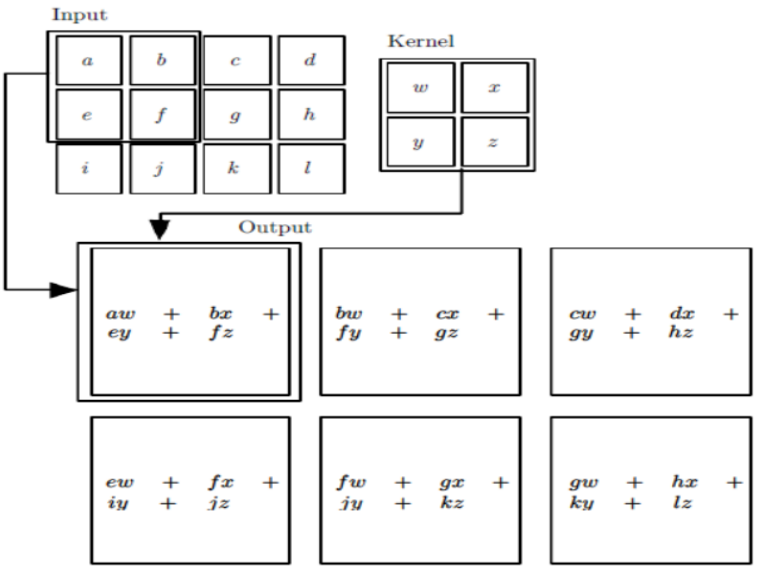
Batch normalization

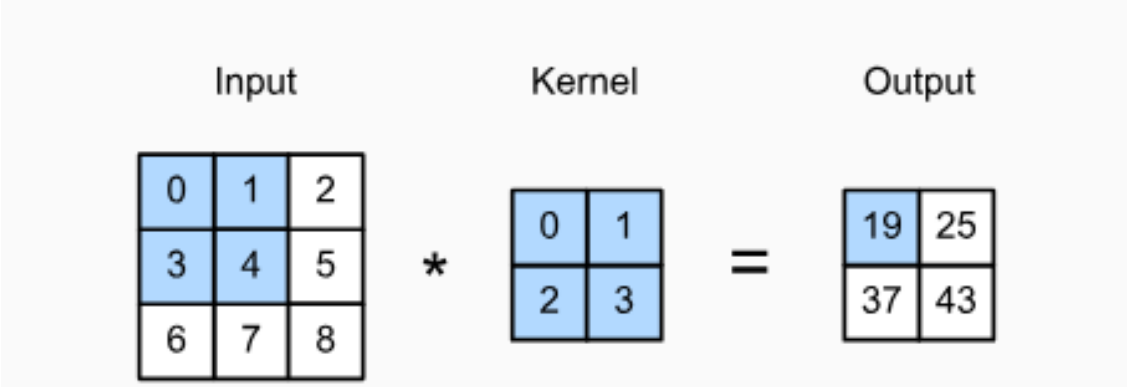
- Batch normalization (batch norm) is a method used to make training of NN faster and more stable through normalization.
- Activation at each layer could result in different data distribution.
- Hence, to increase the stability of deep neural networks we need to normalize the data fed at each layer *by subtracting the mean and dividing by the standard deviation*.

Convolutional Neural Network

- A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various objects in the image and be able to differentiate one from the other.
- 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.
- The name "Convolutional neural network" indicates that the network employs a mathematical operation called Convolution.
- Convolution is a specialized kind of linear operation.
- Convnets are simply neural networks that use convolution in place of general matrix multiplication.
- The convolutional layer is the core building block of a CNN, and it is where the majority of computation occurs.
- It requires a few components, which are input data, a filter, and a feature map.
- This filter is also called a kernel, or feature detector, and its dimensions can be, for example, 3x3.
- A filter acts as a pattern, which, when convolved across the input, finds similarities between the ***stored template*** *&* different locations/regions in **the input image.**
- To perform convolution, the kernel goes over the input image, doing matrix multiplication element after element.
- The movement of a kernel is always from left to right and top to bottom.

Example: Matrix multiplication for 2D Convolution



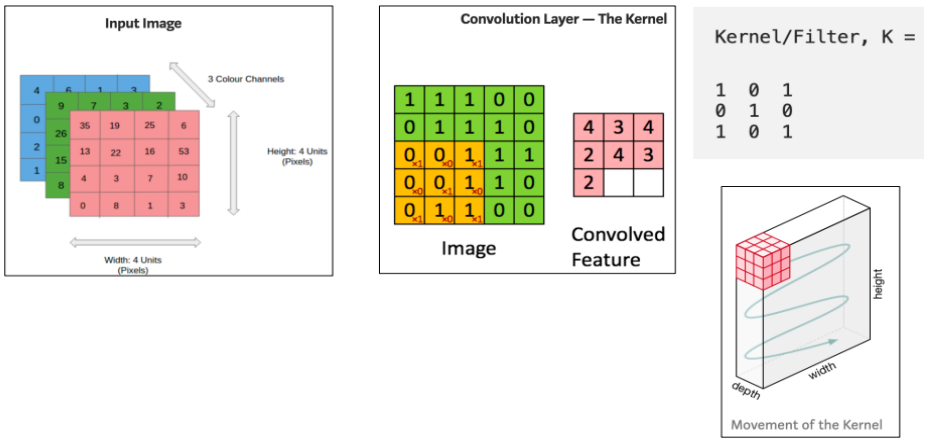


Padding and striding

- These techniques are often used in CNNs
- Padding. Padding expands the input matrix by adding fake pixels to the borders of the matrix.
- This is done because convolution reduces the size of the matrix.
- For example, a 5x5 matrix turns into a 3x3 matrix when a filter goes over it.
- Striding. It often happens that when working with a convolutional layer, you need to get an output that is smaller than the input.

- Stride defines by what step does kernel move, for example stride of 1 makes kernel slide by one row/column at a time and stride of 2 moves kernel by 2 rows/columns.
- Pooling layers are used to reduce the dimensions of the feature maps.
- Thus, it reduces the number of parameters to learn and the amount of computation performed in the network.
- The pooling layer summarises the features present in a region of the feature map generated by a convolution layer.

Convolutional Layer for Image Recognition



There are 2 types of Pooling Layers, they are

Max Pooling

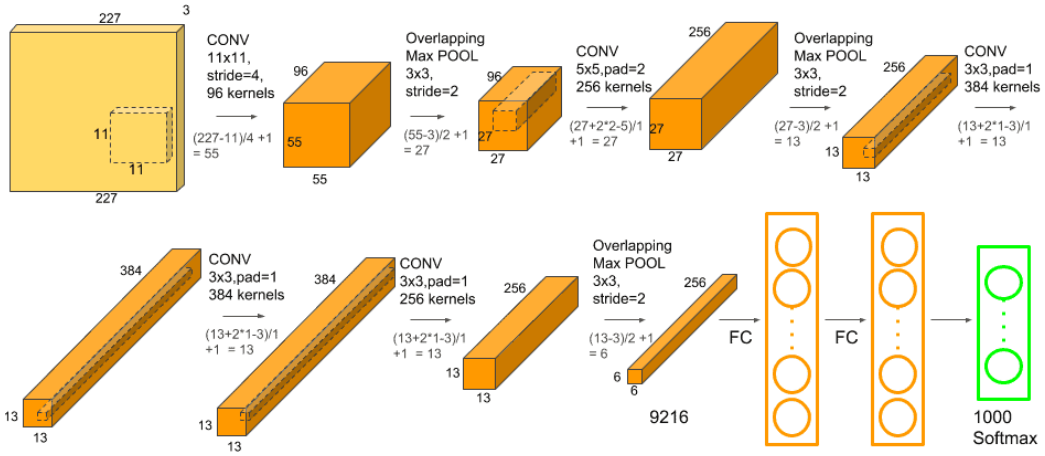
- Max pooling is a pooling operation that selects the maximum element from the region of the feature map covered by the filter.

Average Pooling

- Average pooling computes the average of the elements present in the region of feature map covered by the filter.

- Thus, while max pooling gives the most prominent feature in a particular patch of the feature map, average pooling gives the average of features present in a patch.

Example::



$$W_{out} = \frac{W - F + 2P}{S} + 1$$

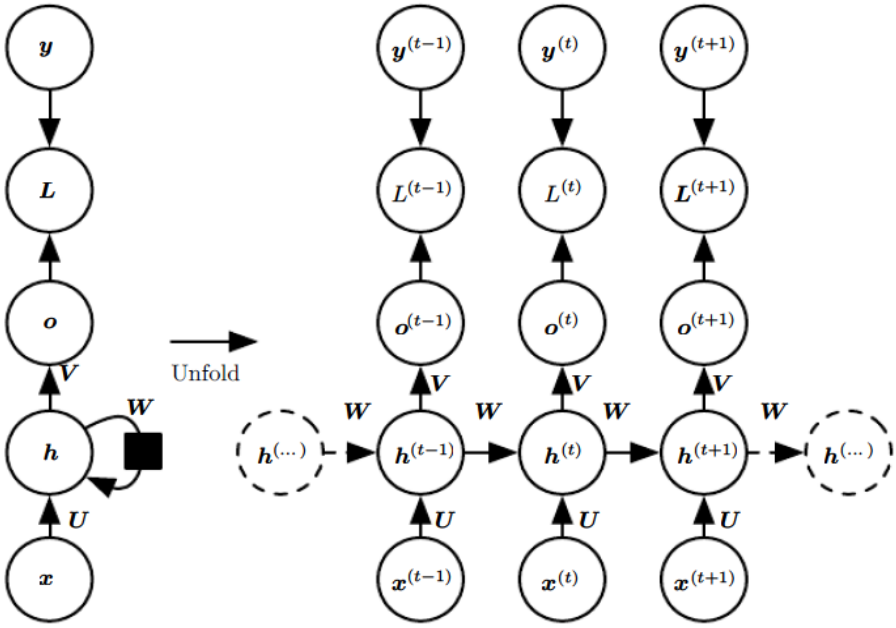
Formula for Convolution Layer

$$W_{out} = \frac{W - F}{S} + 1$$

Recurrent Neural Network

- Recurrent Neural Network(RNN) is a type of Neural Network where the output from the previous step is fed as input to the current step.
- In traditional neural networks, all the inputs and outputs are independent of each other.

- In few cases it is required to predict the next word of a sentence, the previous words are required and hence there is a need to remember the previous words.
- The main and most important feature of RNN is its **Hidden state**, which remembers some information about a sequence.
- The state is also referred to as ***Memory State*** since it remembers the previous input to the network.
- It uses the same parameters for each input as it performs the **same task** on all the inputs or hidden layers to produce the output.
- This reduces the complexity of parameters.
- A recurrent neural network is a neural network that is specialized for processing a sequence of data **x(t)= x(1), . . . , x(τ)** with the time step index ***t*** ranging from **1 to τ**.
- For tasks that involve sequential inputs, such as speech and language, it is often better to use RNNs.
- RNNs are called ***recurrent*** because they perform the **same task** for every element of a sequence, with the output being depended on the previous computations.



- The computational graph to compute the training loss of a recurrent network that maps an input sequence of x values to a corresponding sequence of output o values.
- A loss L measures how far each o is from the corresponding training target y .
- When using softmax outputs, we assume o is the unnormalized log probabilities.
- The loss L internally computes $\hat{y} = \text{softmax}(o)$ and compares this to the target y.
- The RNN has input to hidden connections parametrized by a weight matrix U, hidden-to-hidden recurrent connections parametrized by a weight matrix W, and hidden-to-output connections parametrized by a weight matrix V .

Advantages of Recurrent Neural Network

- An RNN remembers each and every piece of information through time.

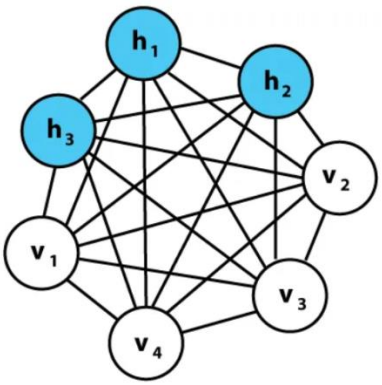
- It is useful in time series prediction only because of the feature to remember previous inputs as well.
- This is called Long Short Term Memory.

Applications of Recurrent Neural Network

- Language Modeling and Generating Text
- Speech Recognition
- Time series Forecasting

Deep Belief Network

- A Deep Belief Network(DBN) is a powerful generative model that use a deep architecture of multiple stacks of Restricted Boltzmann machines(RBM).
- Boltzmann Machines is an unsupervised DL model in which every node is connected to every other node.
- A Boltzmann Machine (BM) is a probabilistic generative undirected graph model.
- BMs learn the probability density from the input data to generating new samples from the same distribution.

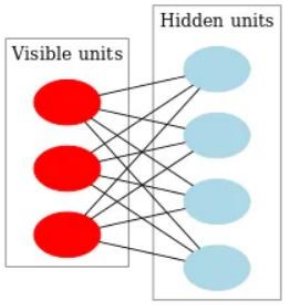


Boltzmann Machines

- A BM has an input or **visible layer** and one or several **hidden layers.** There is no output layer.
- When the input is provided, they are able to capture all the parameters, patterns and correlations among the data.

Restricted Boltzmann machine

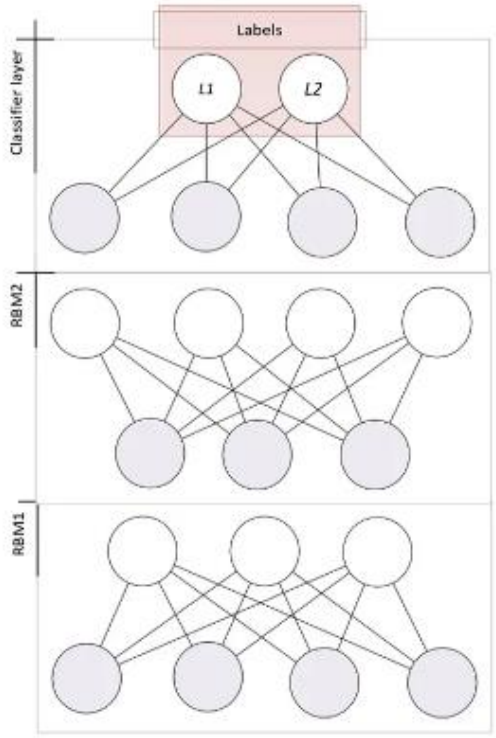
- RBMs are a two-layered generative stochastic building blocks that can learn a probability distribution over its set of inputs features(i.e. image pixels).
- As the name implies, RBMs are a variant of Boltzmann machines with a small difference, their neurons must form a bipartite graph, which means there are **no connections** between nodes within a group(**visible** and the **hidden**)
- In simpler terms, this means that we basically have fewer connections.



- RBMs hold two sets of random variables (also called neurons): one layer of **visible** variables/nodes(which is the layer where the inputs go) to represent observable data and one layer of **hidden** variables to capture dependencies(calculate the probability distribution of the features) of the visible variables.

- Deep Belief Networks (DBNs) are sophisticated artificial neural networks used in the field of deep learning, a subset of machine learning.
- They are designed to discover and learn patterns within large sets of data automatically.
- Imagine them as multi-layered networks, where each layer is capable of making sense of the information received from the previous one, gradually building up a complex understanding of the overall data.
- DBNs are composed of multiple layers of stochastic, or randomly determined, units.
- These units are known as Restricted Boltzmann Machines (RBMs) or other similar structures.
- Each layer in a DBN aims to extract different features from the input data, with lower layers identifying basic patterns and higher layers recognizing more abstract concepts.
- This structure allows DBNs to effectively learn complex representations of data, which makes them particularly useful for tasks like image and speech recognition, where the input data is high-dimensional and requires a deep level of understanding.
- The architecture of DBNs also makes them good at unsupervised learning, where the goal is to understand and label input data without explicit guidance.
- This characteristic is particularly useful in scenarios where labelled data is scarce or when the goal is to explore the structure of the data without any preconceived labels.
- Each RBM model performs a non-linear on its input vectors and produces as outputs vectors that will serve as input for the next RBM model in the sequence.

- DBNs have the ability to do feature learning/extraction and classification that are used in many applications.

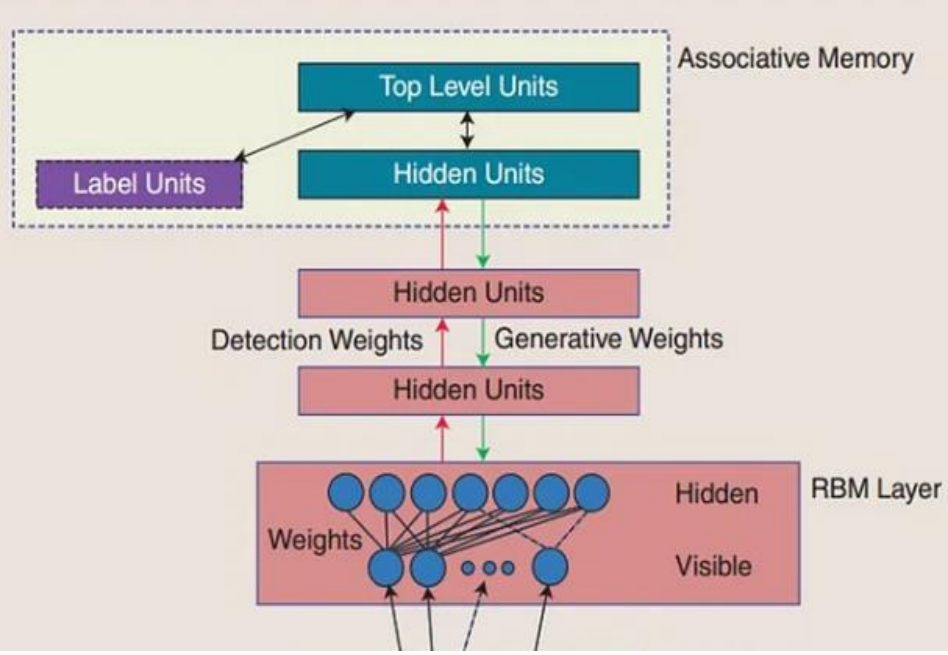


- Precisely, in feature learning we do layer-by-layer pre-training in an unsupervised manner on the different RBMs that form a DBN and we use back-propagation technique(i.e. gradient descent) to do classification and other tasks by fine-tuning on a small labelled dataset.

How Deep Belief Networks Work?

- DBNs work in two main phases: pre-training and fine-tuning.
- In the pre-training phase, the network learns to represent the input data layer by layer.
- Each layer is trained independently as an RBM, which allows the network to learn complex data representations efficiently.
- During this phase, the network learns the probability distribution of the inputs, which helps it understand the underlying structure of the data.
- In the fine-tuning phase, the DBN adjusts its parameters for a specific task, like classification or regression.

- This is typically done using a technique known as backpropagation, where the network's performance on a task is evaluated, and the errors are used to update the network's parameters.
- This phase often involves supervised learning, where the network is trained with labelled data.



*****