UNIT 5

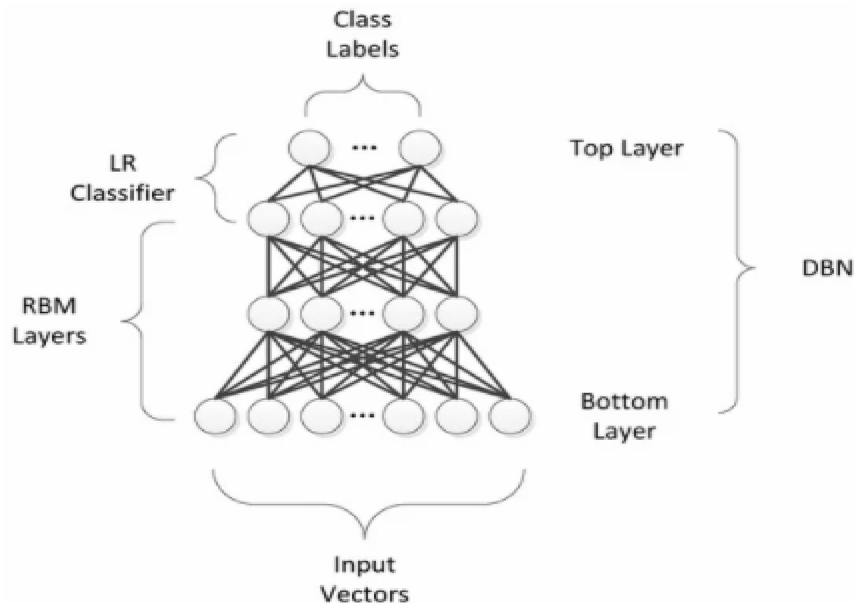
DEEP BELIEF NETWORKS

1. What is a Deep Belief Network?

We create ***Deep Belief Networks (DBNs)*** to address issues with classic neural networks in deep layered networks.

For example – slow learning, becoming stuck in local minima owing to poor parameter selection, and requiring a large number of training datasets.

- Several layers of stochastic latent variables make a DBN. Binary latent variables that are often known as feature detectors or hidden units are binary variables.
- DBN is a hybrid generative graphical model. The top two layers have no direction. The layers above have directed links to lower layers.
- DBN is an algorithm for unsupervised probabilistic deep learning.

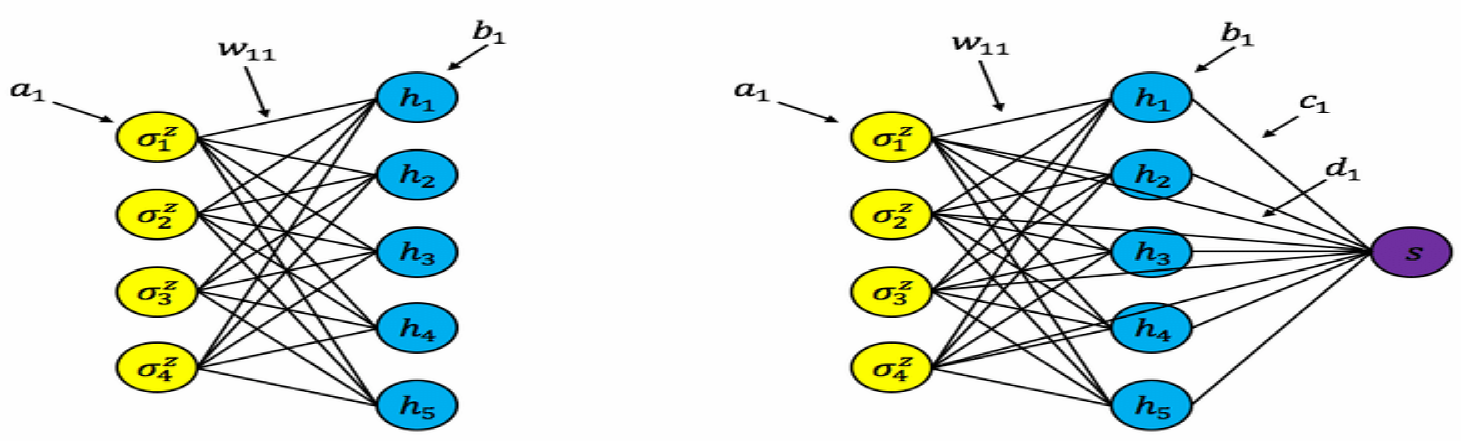


Source: Mdpi.com

Deep Belief Networks are machine learning algorithm that resembles the deep neural network but are not the same. These are feedforward neural networks with a deep architecture, i.e., having many hidden layers. Simple, unsupervised networks like restricted Boltzmann machines- RBMs or autoencoders make DBNs, with the hidden layer of each sub-network serving as the visible layer for the next layer.

How did Deep Belief Neural Networks Evolve?

We employ Perceptrons in the First Generation of neural networks to identify a certain object or anything else by considering the weight. However, Perceptrons may be beneficial for basic technology only, but not for sophisticated technology. To address these problems, the Second Generation of Neural Networks introduced the notion of Backpropagation, which compares the received output to the desired output and reduces the error value to zero. Then came directed acyclic graphs known as belief networks, which aided in the solution of inference and learning problems. Then, we'll use Deep Belief Networks to help construct unbiased values that we can store in leaf nodes.

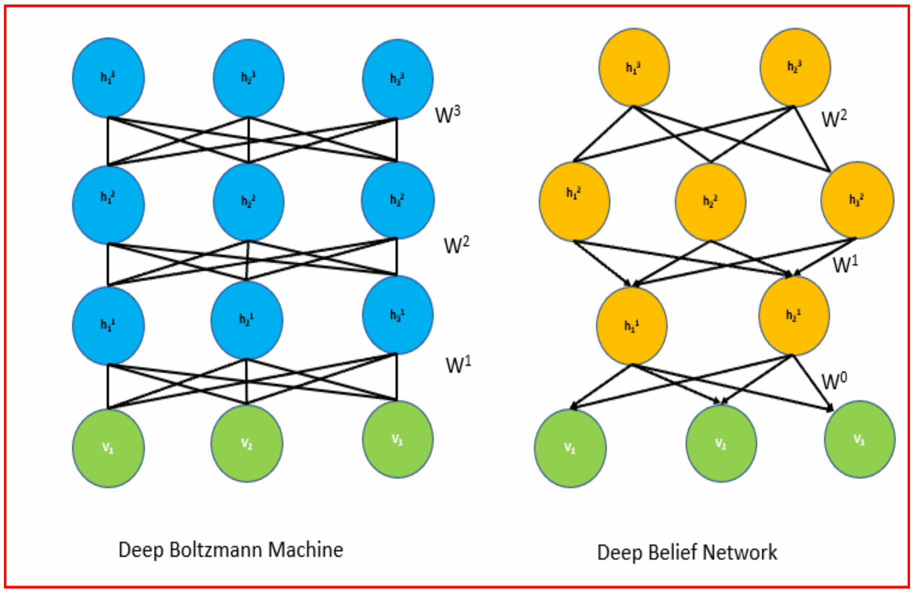


Restricted Boltzmann Machines

A *Restricted Boltzmann Machine (RBM)* is a type of generative stochastic artificial neural network that can learn a probability distribution from its inputs. Deep learning networks can also use RBM. Deep belief networks, in particular, can be created by "stacking" RBMs and fine-tuning the resulting deep network via gradient descent and backpropagation.

The Architecture of DBN

A series of constrained Boltzmann machines connected in a specific order make a ***Deep Belief Network****. We supplement the result of the "output" layer of the Boltzmann machine as input to the next Boltzmann machine consecutively. Then we'll train it until its convergence and apply the same until the completion of the whole network.*

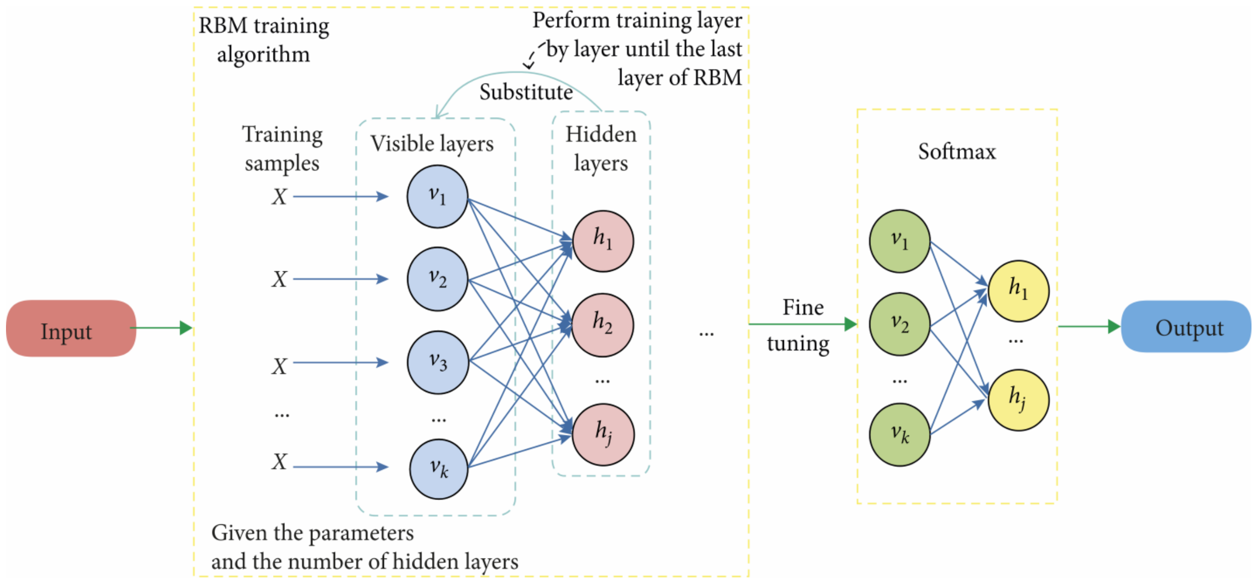


Source: Medium.com

The undirected and symmetric connections between the top two levels of DBN form associative memory. The arrows pointing towards the layer closest to the data point to the relationships between all lower layers. Directed acyclic connections in the lower layers translate associative memory to observable variables. The lowest layer of visible units receives the input data. We can use Binary or actual data as input. Like *Restricted Boltzmann Machine (RBM)*, there are no intralayer connections. The hidden units represent features that encapsulate the data's correlations. A matrix of symmetrical weights W connects two layers. We'll link every unit in each layer to every other unit in the layer above it.

How does DBN work?

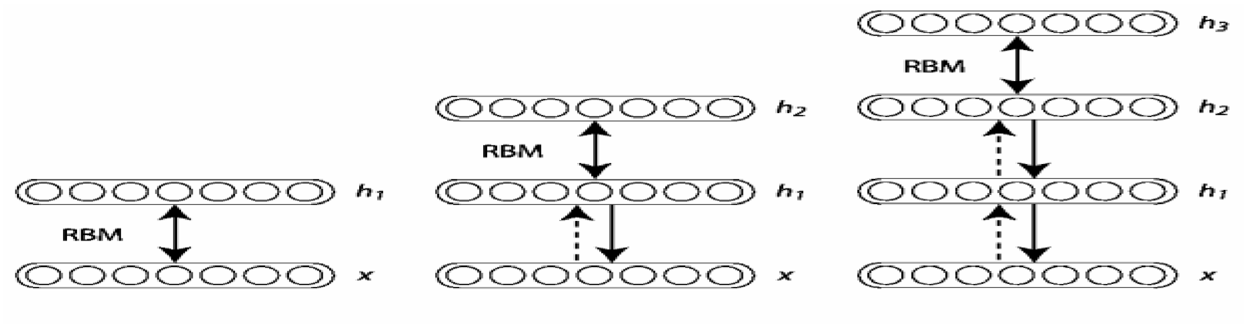
The first stage is to train a property layer that can directly gain input signals from pixels. In an alternate retired subcaste, learn the features of the preliminarily attained features by treating the values of this subcaste as pixels. The lower bound on the log-liability of the training data set improves every time a fresh subcaste of parcels or features that we add to the network.



Source: Hindawi.com

Deep Belief Network's operational pipeline is as follows:

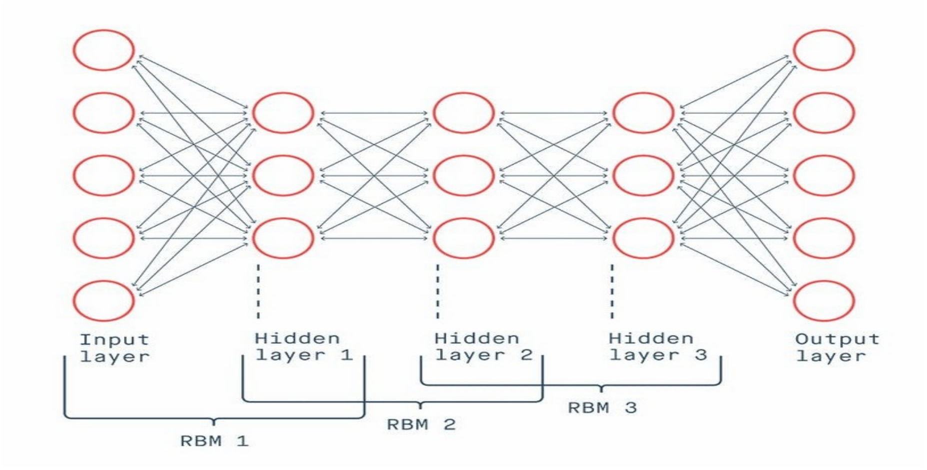
- We'll use the Greedy learning algorithm to pre-train DBN. For learning the top-down generative weights-the greedy learning method that employs a layer-by-layer approach. These generative weights determine the relationship between variables in one layer and variables in the layer above.
- On the top two hidden layers, we run numerous steps of Gibbs sampling in DBN. The top two hidden layers define the RBM thus, this stage is effectively extracting a sample from it.
- Then generate a sample from the visible units using a single pass of ancestral sampling through the rest of the model.
- We'll use a single bottom-up pass to infer the values of the latent variables in each layer. In the bottom layer, greedy pretraining begins with an observed data vector. It then oppositely fine-tunes the generative weights.



It's necessary to remember that constructing a Deep Belief Network necessitates training each RBM layer. Initially, we'll initiate the units and parameters for this purpose. In the Contrastive Divergence algorithm, there are two phases: positive and negative. We'll calculate the binary states of the hidden layers in the positive phase by computing the probabilities of weights and visible units. It is known as the positive phase since it enhances the likelihood of the training data set. The negative phase reduces the likelihood of the model producing samples. To train a complete Deep Belief Network, we'll employ the greedy learning technique. The greedy learning algorithm trains one RBM at a time until all of the RBMs are trained.

Creating a Deep Belief Network

Several RBMs together make a Deep Belief Networks, which reflects in their Clojure record structure. We use a fully unsupervised form of DBN to initialize a Deep Neural Network, whereas we use a classification DBN (CDBN) as a classification model on its own.



Each record type includes the RBMs that make up the network's layers, as well as a vector- indicating the layer size and, in the case of the classification DBN- the number of classes in the representative dataset. The DBN record represents a model made up entirely of stacked RBMs, but the CDBN record contains the top-level associative memory, which is a CRBM record. This enables the top layer to be trained to create class labels for input data vectors and to classify unknown data vectors.

Learning a Deep Belief Network

Factually, we learned that RBMs are the ones that make DBN unsupervised making it much easier to train. The RBM training duration is longer than the whole DBN training period, but the code is simpler. Because CDBNs require the observation labels to be available during top-layer training, a training session entails first training the bottom layer, then propagating the dataset via the learned RBM and using the newly altered dataset as the training data for the next RBM. We'll repeat this process until the passage of the dataset through the penultimate trained RBM, and then all the labels are concatenated with the altered dataset and used to train the top-layer associative memory. The DBN has hyperparameters to set, similar to the RBM model, and offers sensible default values.

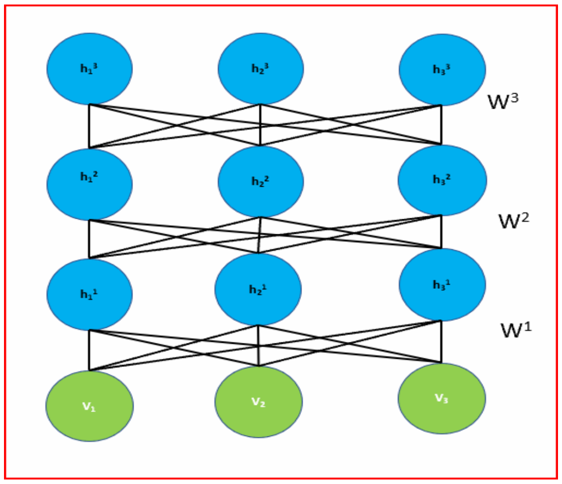
Applications

We employ deep belief networks in place of deep feedforward networks or even convolutional neural networks in more sophisticated setups. They have the benefit of being less computationally costly. computational complexity grows linearly with the number of layers, rather than exponentially as with feedforward neural networks) and is less susceptible to the vanishing gradients problem.

Applications of DBN are as follows:

- Recognition of images.
- Sequences of video.
- Data on mocap.
- Speech recognition.

2. Deep Boltzmann Machine (DBM)



Deep Boltzmann Machine

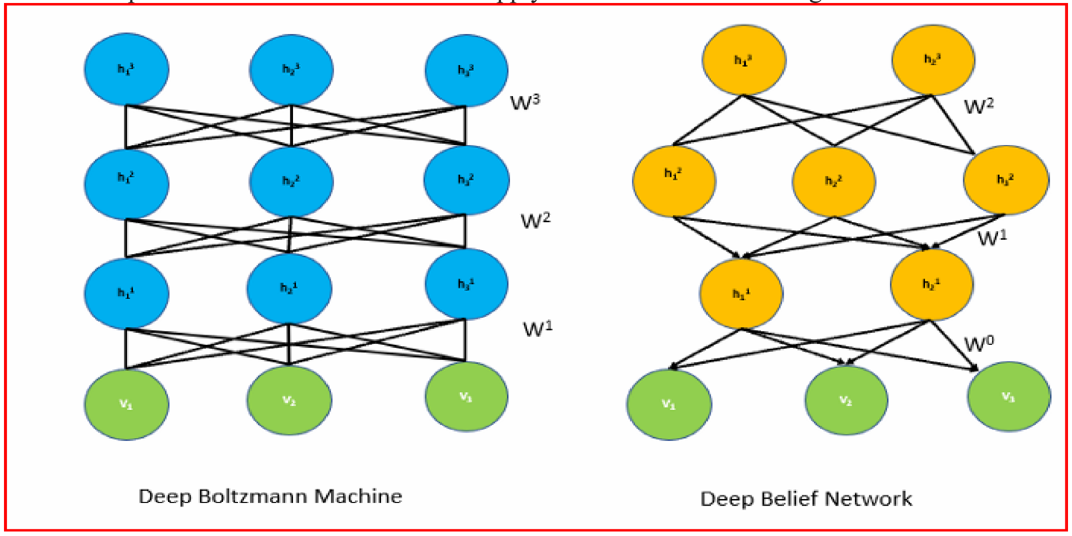
- Unsupervised, probabilistic, generative model with entirely undirected connections between different layers
- Contains visible units and multiple layers of hidden units
- Like RBM, no intralayer connection exists in DBM. Connections exists only between units of the neighboring layers
- Network of symmetrically connected stochastic binary units
- DBM can be organized as bipartite graph with odd layers on one side and even layers on one side
- Units within the layers are independent of each other but are dependent on neighboring layers
- Learning is made efficient by layer by layer pre training — Greedy layer wise pre training slightly different than done in DBM
- After learning the binary features in each layer, DBM is fine tuned by back propagation.

Sounds similar to DBN so what is the difference between Deep Belief networks(DBN) and Deep Boltzmann Machine(DBM)?

Let's talk first about similarity between DBN and DBM and then difference between DBN and DBM

Similarity between *Deep Belief networks(DBN) and Deep Boltzmann Machine(DBM)*

- Both DBN and DBM are unsupervised, probabilistic, generative, graphical model consisting of stacked layers of RBM.
- DBN and DBM both are used to identify latent feature present in the data.
- Both DBN and DBM performs inference and parameter learning efficiently using greedy layer–wise training.
- Both DBN and DBM apply discriminative fine tuning after greedy layer wise pre training.
- Both DBN and DBM use a large set of unlabeled data for pre training in an unsupervised manner to find good set of parameter for the model and then apply discriminative fine tuning on a small labelled dataset.



Difference between Deep Belief networks(DBN) and Deep Boltzmann Machine(DBM)

- Deep Belief Network(DBN) have top two layers with undirected connections and lower layers have directed connections
- Deep Boltzmann Machine(DBM) have entirely undirected connections.
- Approximate inference procedure for DBM uses a top-down feedback in addition to the usual bottom-up pass, allowing Deep Boltzmann Machines to better incorporate uncertainty about ambiguous inputs.
- A disadvantage of DBN is the approximate inference based on mean field approach is slower compared to a single bottom-up pass as in Deep Belief Networks. Mean field inference needs to be performed for every new test input.

What is Mean Field or Variational Approximation?

Explaining mean field or variational approximation intuitively here

Computing posterior distribution is known as an inference problem.

We have a data distribution P(x) and computing posterior distribution is often intractable.

We can approximate intractable inference with simpler tractable inference by introducing a distribution Q(x) which is the best approximation of P(x).

Q(x) becomes the mean field approximation where variables in Q distribution is independent of variable x.

Our goal is to minimize KL divergence between the approximate distribution and the actual distribution

How is DBM trained ?

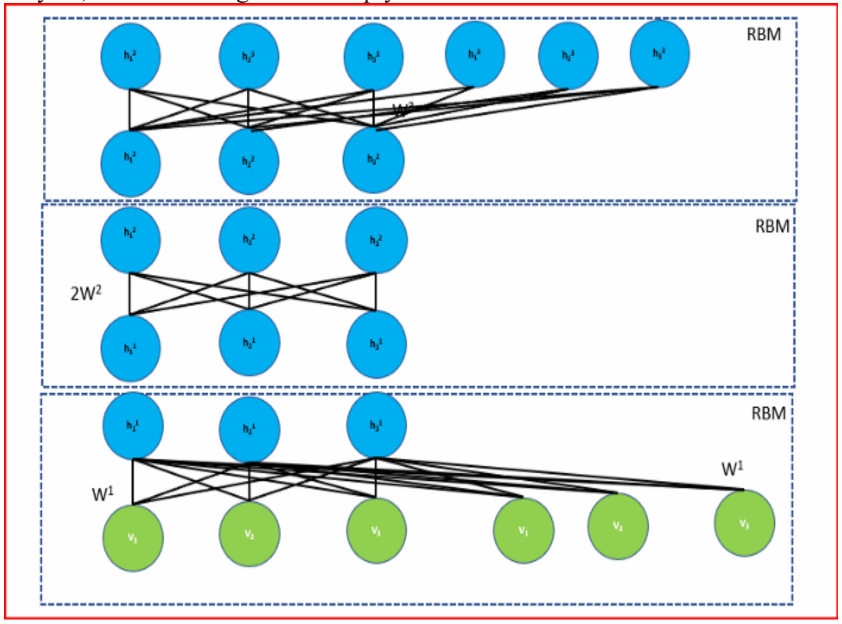
Boltzmann machine uses randomly initialized Markov chains to approximate the gradient of the likelihood function which is too slow to be practical.

DBM uses greedy layer by layer pre training to speed up learning the weights. It relies on learning stacks of Restricted Boltzmann Machine with a small modification using contrastive divergence.

The key intuition for greedy layer wise training for DBM is that we double the input for the lower-level RBM and the top level RBM.

Lower level RBM inputs are doubled to compensate for the lack of top-down input into first hidden layer.Similarly for top-level RBM, we double the hidden units to compensate for the lack of bottom-up input.

For the intermediate layers, the RBM weights are simply doubled.



Double inputs for the input and the top level hidden layer. Double the weights for the intermediate layers

The 3 RBM’s are then combined to form a single model.

Greedily pretraining the weights of a DBM initializes the weights to reasonable values helping subsequent joint learning of all layers.

This is expensive compared to a single bottom up inference used in DBN. In order to learn using large dataset we need to accelerate inference in a DBM.

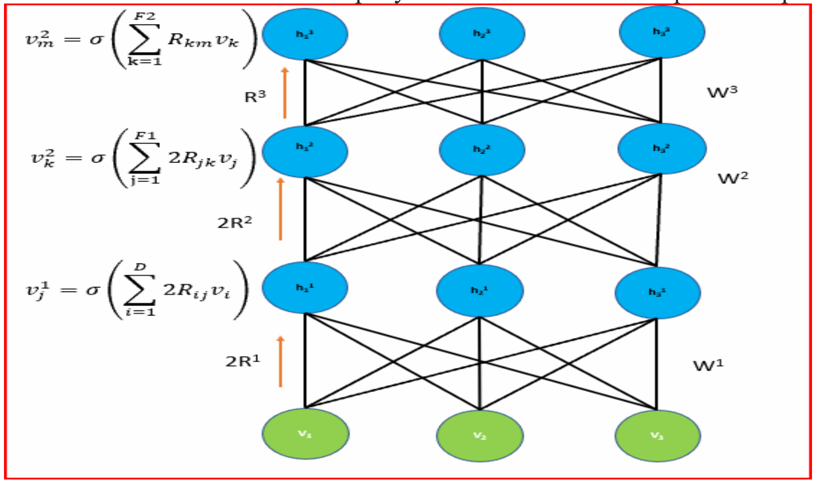
In order to accelerate inference of DBM, we use a set of recognition weights, which are initialized to the weights found by the greedy pre training.

We take an input vector and apply the recognition weights to reconstruct the input *v* of fully factorized approximation posterior distribution.

Each layer of hidden units is activated in a single deterministic bottom-up pass as shown in figure below.

We double the weights of the recognition model at each layer to compensate for the lack of top-down feedback.

However we do not double the top layer as it does not have a top-down input.

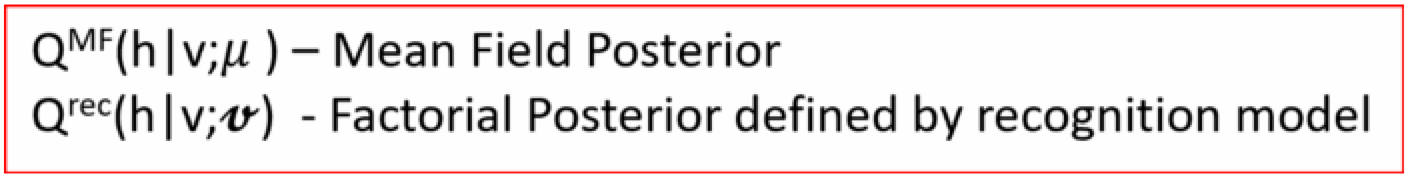

$v_m^2 = \sigma\left(\sum_{k=1}^{F2} R_{km} v_k\right)$
$v_k^2 = \sigma\left(\sum_{j=1}^{F1} 2R_{jk} v_j\right)$
$v_j^1 = \sigma\left(\sum_{i=1}^{D} 2R_{ij} v_i\right)$
R^3
$2R^2$
$2R^1$
W^3
W^2
W^1
h_1^3
h_2^3
h_3^3
h_1^2
h_2^2
h_3^2
h_1^1
h_2^1
h_3^1
v_1
v_2
v_3

{R1 , R2 , R3} denotes set of recognition weights

We apply K iterations of mean-field to obtain the mean-field parameters that will be used in the training update for DBM’s.

Finally we update the recognition weights for an initial guess of the input ν close to the result μ. μ is the result of the mean field inference which is ur target

By updating the recognition weights we want to minimize the KL divergence between the mean-field posterior (h|v; μ) and the recognition model.


$Q^{MF}(h|v;\mu)$ – Mean Field Posterior
$Q^{rec}(h|v;\upsilon)$ - Factorial Posterior defined by recognition model

Objective is to minimize KL divergence between mean field posterior and factorial posterior of the recognition model

Discriminative Fine Tuning of DBM

To perform classification, we need a separate multi layer perceptrons(MLP) on top of the hidden features extracted

from greedy layer pre training just as fine tuning is performed in DBN

3. Generative Adversarial Network (GAN)

Generative Adversarial Networks (GANs) are a powerful class of neural networks that are used for unsupervised learning. It was developed and introduced by Ian J. Goodfellow in 2014. GANs are basically made up of a system of two competing neural network models which compete with each other and are able to analyze, capture and copy the variations within a dataset.

Why were GANs developed in the first place?

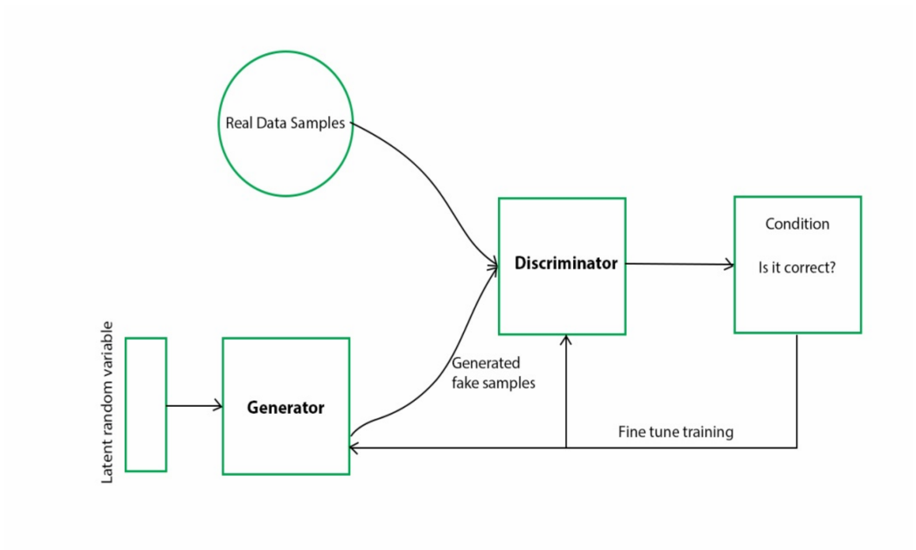
It has been noticed most of the mainstream neural nets can be easily fooled into misclassifying things by adding only a small amount of noise into the original data. Surprisingly, the model after adding noise has higher confidence in the wrong prediction than when it predicted correctly. The reason for such adversary is that most machine learning models learn from a limited amount of data, which is a huge drawback, as it is prone to overfitting. Also, the mapping between the input and the output is almost linear. Although, it may seem that the boundaries of separation between the various classes are linear, but in reality, they are composed of linearities and even a small change in a point in the feature space might lead to misclassification of data.

How does GANs work?

Generative Adversarial Networks (GANs) can be broken down into three parts:

- **Generative:** To learn a generative model, which describes how data is generated in terms of a probabilistic model.
- **Adversarial:** The training of a model is done in an adversarial setting.
- **Networks:** Use deep neural networks as the artificial intelligence (AI) algorithms for training purpose.

In GANs, there is a **generator** and a **discriminator**. The Generator generates fake samples of data(be it an image, audio, etc.) and tries to fool the Discriminator. The Discriminator, on the other hand, tries to distinguish between the real and fake samples. The Generator and the Discriminator are both Neural Networks and they both run in competition with each other in the training phase. The steps are repeated several times and in this, the Generator and Discriminator get better and better in their respective jobs after each repetition. The working can be visualized by the diagram given below:



Here, the generative model captures the distribution of data and is trained in such a manner that it tries to maximize the probability of the Discriminator in making a mistake. The Discriminator, on the other hand, is based on a model that estimates the probability that the sample that it got is received from the training data and not from the Generator.
The GANs are formulated as a minimax game, where the Discriminator is trying to minimize its reward **V(D, G)** and the Generator is trying to minimize the Discriminator's reward or in other words, maximize its loss. It can be mathematically described by the formula below:

$$\min_G \max_D V(D,G)$$

$$V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))]$$

where,
G = Generator
D = Discriminator
Pdata(x) = distribution of real data
P(z) = distribution of generator
x = sample from Pdata(x)
z = sample from P(z)
D(x) = Discriminator network
G(z) = Generator network

So, basically, training a GAN has two parts:

- **Part 1:** The Discriminator is trained while the Generator is idle. In this phase, the network is only forward propagated and no back-propagation is done. The Discriminator is trained on real data for n epochs, and see if it can correctly predict them as real. Also, in this phase, the Discriminator is also trained on the fake generated data from the Generator and see if it can correctly predict them as fake.
- **Part 2:** The Generator is trained while the Discriminator is idle. After the Discriminator is trained by the generated fake data of the Generator, we can get its predictions and use the

results for training the Generator and get better from the previous state to try and fool the Discriminator.

The above method is repeated for a few epochs and then manually check the fake data if it seems genuine. If it seems acceptable, then the training is stopped, otherwise, its allowed to continue for few more epochs.

Different types of GANs:

GANs are now a very active topic of research and there have been many different types of GAN implementation. Some of the important ones that are actively being used currently are described below:

1. **Vanilla GAN:** This is the simplest type GAN. Here, the Generator and the Discriminator are simple multi-layer perceptrons. In vanilla GAN, the algorithm is really simple, it tries to optimize the mathematical equation using stochastic gradient descent.
2. **Conditional GAN (CGAN):** CGAN can be described as a deep learning method in which some conditional parameters are put into place. In CGAN, an additional parameter 'y' is added to the Generator for generating the corresponding data. Labels are also put into the input to the Discriminator in order for the Discriminator to help distinguish the real data from the fake generated data.
3. **Deep Convolutional GAN (DCGAN):** DCGAN is one of the most popular also the most successful implementation of GAN. It is composed of ConvNets in place of multi-layer perceptrons. The ConvNets are implemented without max pooling, which is in fact replaced by convolutional stride. Also, the layers are not fully connected.
4. **Laplacian Pyramid GAN (LAPGAN):** The Laplacian pyramid is a linear invertible image representation consisting of a set of band-pass images, spaced an octave apart, plus a low-frequency residual. This approach uses multiple numbers of Generator and Discriminator networks and different levels of the Laplacian Pyramid. This approach is mainly used because it produces very high-quality images. The image is down-sampled at first at each layer of the pyramid and then it is again up-scaled at each layer in a backward pass where the image acquires some noise from the Conditional GAN at these layers until it reaches its original size.
5. **Super Resolution GAN (SRGAN):** SRGAN as the name suggests is a way of designing a GAN in which a deep neural network is used along with an adversarial network in order to produce higher resolution images. This type of GAN is particularly useful in optimally up-scaling native low-resolution images to enhance its details minimizing errors while doing so.

Applications of Generative Adversarial Networks (GANs)

Reading about GANs is too exciting and when you will read their application then I hope that excitement

will reach the sky and then study the working of GANs creates a different impact on learning.

1. Generate new data from available data – It means generating new samples from an available sample that is not similar to a real one.
2. Generate realistic pictures of people that have never existed.
3. Gans is not limited to Images, It can generate text, articles, songs, poems, etc.
4. Generate Music by using some clone Voice – If you provide some voice then GANs can generate a similar clone feature of it. In this research paper, researchers from NIT in Tokyo proposed a

system that is able to generate melodies from lyrics with help of learned relationships between notes and subjects.

5. Text to Image Generation (Object GAN and Object Driven GAN)
6. Creation of anime characters in Game Development and animation production.
7. Image to Image Translation – We can translate one Image to another without changing the background of the source image. For example, Gans can replace a dog with a cat.
8. Low resolution to High resolution – If you pass a low-resolution Image or video, GAN can produce a high-resolution Image version of the same.
9. Prediction of Next Frame in the video – By training a neural network on small frames of video, GANs are capable to generate or predict a small next frame of video. For example, you can have a look at below GIF
10. Interactive Image Generation – It means that GANs are capable to generate images and video footage in an art form if they are trained on the right real dataset.
11. Speech – Researchers from the College of London recently published a system called GAN-TTS that learns to generate raw audio through training on 567 corpora of speech data.

5.4 **DEEP ASSOCIATIVE MEMORY**

These kinds of neural networks work on the basis of pattern association, which means they can store different patterns and at the time of giving an output they can produce one of the stored patterns by matching them with the given input pattern. These types of memories are also called **Content-Addressable Memory** CAMCAM. Associative memory makes a parallel search with the stored patterns as data files.

Following are the two types of associative memories we can observe −

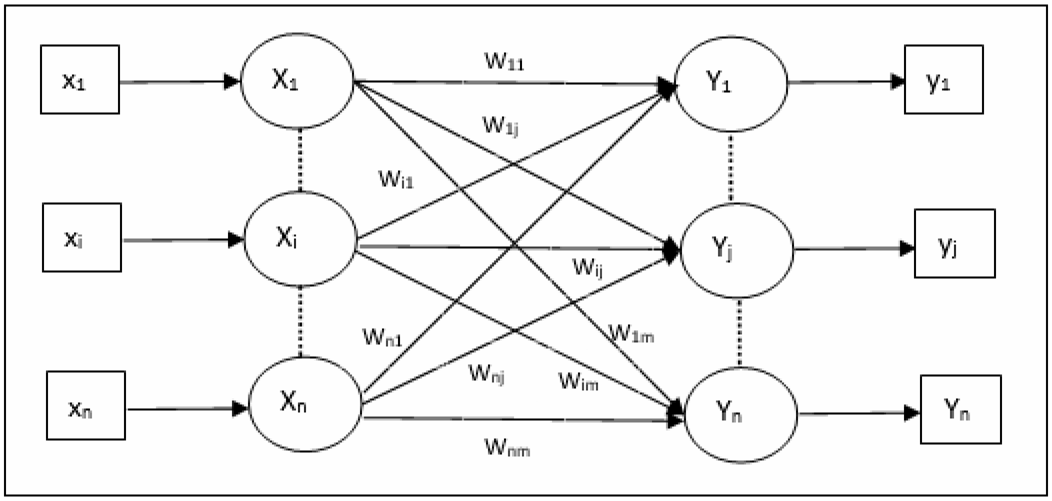
- Auto Associative Memory
- Hetero Associative memory

Auto Associative Memory

This is a single layer neural network in which the input training vector and the output target vectors are the same. The weights are determined so that the network stores a set of patterns.

Architecture

As shown in the following figure, the architecture of Auto Associative memory network has **‘n’** number of input training vectors and similar **‘n’** number of output target vectors.



Training Algorithm

For training, this network is using the Hebb or Delta learning rule.

Step 1 − Initialize all the weights to zero as $\mathbf{w_{ij} = 0}$ i=1ton,j=1toni=1ton,j=1ton

Step 2 − Perform steps 3-4 for each input vector.

Step 3 − Activate each input unit as follows –

$$x_i = s_i \; (i = 1 \; to \; n)$$

Step 4 – Activate each output unit as follows –

$$y_j = s_j \; (j = 1 \; to \; n)$$

Step 5 – Adjust the weights as follows –

$$w_{ij}(new) = w_{ij}(old) + x_i y_j$$

Testing Algorithm

Step 1 – Set the weights obtained during training for Hebb's rule.

Step 2 – Perform steps 3-5 for each input vector.

Step 3 – Set the activation of the input units equal to that of the input vector.

Step 4 – Calculate the net input to each output unit **j = 1 to n**

$$y_{inj} = \sum_{i=1}^{n} x_i w_{ij}$$

Step 5 – Apply the following activation function to calculate the output

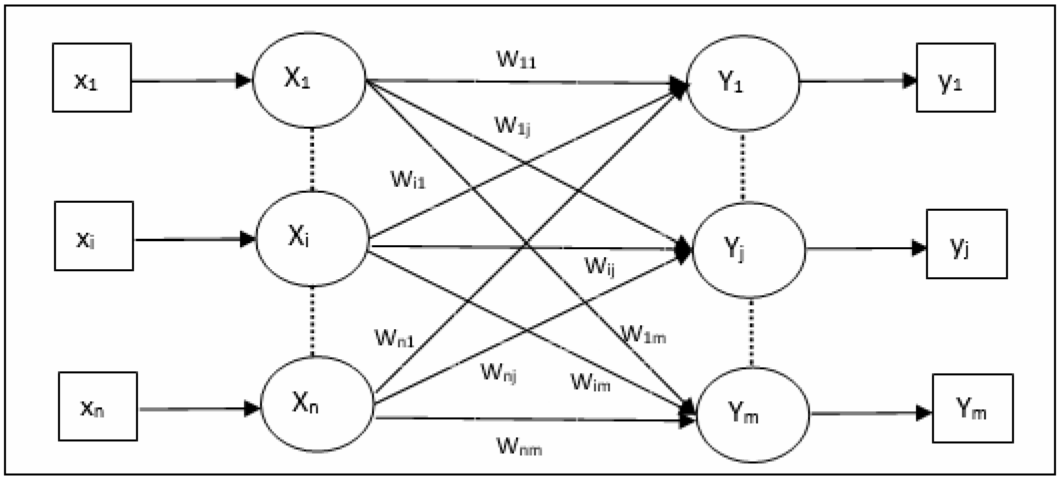
$$y_j = f(y_{inj}) = \begin{cases} +1 & if \; y_{inj} > 0 \\ -1 & if \; y_{inj} \leqslant 0 \end{cases}$$

Hetero Associative memory

Similar to Auto Associative Memory network, this is also a single layer neural network. However, in this network the input training vector and the output target vectors are not the same. The weights are determined so that the network stores a set of patterns. Hetero associative network is static in nature, hence, there would be no non-linear and delay operations.

Architecture

As shown in the following figure, the architecture of Hetero Associative Memory network has **'n'** number of input training vectors and **'m'** number of output target vectors.



Training Algorithm

For training, this network is using the Hebb or Delta learning rule.

Step 1 − Initialize all the weights to zero as $\mathbf{w_{ij} = 0}$ i=1ton,j=1tomi=1ton,j=1tom

Step 2 − Perform steps 3-4 for each input vector.

Step 3 − Activate each input unit as follows −

$$x_i = s_i \; (i = 1 \; to \; n)$$

Step 4 − Activate each output unit as follows −

$$y_j = s_j \; (j = 1 \; to \; m)$$

Step 5 − Adjust the weights as follows −

$$w_{ij}(new) = w_{ij}(old) + x_i y_j$$

\

Testing Algorithm

Step 1 − Set the weights obtained during training for Hebb's rule.

Step 2 − Perform steps 3-5 for each input vector.

Step 3 − Set the activation of the input units equal to that of the input vector.

Step 4 − Calculate the net input to each output unit **j = 1 to m;**

$$y_{inj} = \sum_{i=1}^{n} x_i w_{ij}$$

Step 5 – Apply the following activation function to calculate the output

$$y_j = f(y_{inj}) = \begin{cases} +1 & if\, y_{inj} > 0 \\ 0 & if\, y_{inj} = 0 \\ -1 & if\, y_{inj} < 0 \end{cases}$$

5.5 DEEP FAKE

What is Deepfake AI?

Famously called AI that deceives, **deepfake technology takes its name from deep learning, which is a form of AI. In deepfake AI, deep learning algorithms that teach themselves how to solve problems with large data sets, are used to swap faces in videos, images, and other digital content to make the fake appear real.**

Deepfake content is created by **using two algorithms that compete with one another.** One is called a **generator** and the other one is called a **discriminator**. The generator creates the fake digital content and asks the discriminator to find out if the content is real or artificial. Each time the discriminator correctly identifies the content as real or fake, it passes on that information to the generator so as to improve the next deepfake.

When clubbed together, these two algorithms form a **generative adversarial network called GAN.** It uses a set of algorithms to train itself to recognize patterns which help it in learning the real characteristics needed to produce fake images.

What is the Use of a Deepfake?

Deepfake is majorly used for entertainment purposes. For example, the famous deepfake roundtable that featured some super-famous faces like Tom Cruise, George Lucas, Robert Downey Jr, Jeff Goldblum. This roundtable was the way of streaming services to grab your attention as it showed these superstars discussing streaming and the future of cinema in an amusing video. The main purpose of deepfake is to influence viewers and listeners to believe something that did not happen. This is why it is mostly used in movies for a creative effect when the entertainers are not available themselves. Another popular example is in the **Star Wars series when deepfake was used to show characters as they appeared in their youth or replace characters who had died**. In fact, there are times when **online retailers have used this technology to allow customers to try clothes and accessories virtually.**

But the examples given till now were solely used for entertainment. Things get nefarious when people use this technology to **spread false information from an otherwise reliable source, perform financial fraud, data breaching, phishing scams and automated disinformation attacks.**

How Can You Spot a Deepfake?

Now, here comes the big question. How do you spot deepfake videos?

Here are some of the things to look for while identifying deepfake videos:

- **Awkward facial positioning**

If the person's face is pointing one way and their nose is pointing the other way

- **Unnatural body movement**

When someone looks distorted or their movements are not smooth and disjointed

- **Unnatural coloring**

Discoloration, misplaced shadows and abnormal skin tone are signs that you are watching a deepfake

- **Misalignment**

When the visuals are misaligned or blurry

- **Images that look unnatural as you zoom in or slow down**

If you watch a video on a larger screen and zoom in, you can pay more heed to things like bad lip-syncing

- **Inconsistent audio**

Deepfakes spend more time on video images than fixing the audio. Look out for strange word pronunciation, digital background noise or even for that matter, the absence of noise

- **Absence of blinking**

You can see someone's face and tell if they are saying something without batting an eyelid. That's a very good sign to detect a deepfake

How to Combat Deepfakes with Technology?

Several organizations have come together to ensure that AI is used for good and deepfakes do not ruin lives. Here they are:

- **Google** works on text to speech conversion tools to verify speakers
- **Deeptrace** is a startup based out of Amsterdam that is developing deepfake AI detection tools: just like a deepake antivirus
- **US Defense Advanced Research Projects Agency (DARPA)** is funding research to create automated screening of deepfake using a program called MediFor or Medical Forensics
- **Adobe's** system allows you to attach some signature to your content to specify the details of creation

- **Twitter and Facebook** have officially banned the use of malicious deepfakes
- **Sensity** has developed a detection platform that alerts users through e-mail when they are watching something deepfake

Using AI Sensibly

In the vase of deepfakes, AI is being pitted against AI. But just like every coin has two sides, we should remember that the right use of technology is the only ethical practice. Now AI tools are being built to fight deepfake.

Awareness amongst individuals is extremely important to detect and demote false AI. But how can we know something is false if we don't know about it at all?

That's why **McCombs School of Business at The University of Texas at Austin in collaboration with Great Learning** brings you **The Post Graduate Program in Artificial Intelligence: Business Applications.** This is a 6-month online program that teaches you the fundamentals and advanced levels of AI concepts. It is one of the top AI courses in the USA being offered by one of the top universities in the world.

What Is the Difference between a Deepfake and a Shallowfake?

Creating shallowfakes does not require using deep learning systems. But just because shallowfake videos don't use AI, they don't differ much in terms of quality or quantity compared to deepfakes. The name merely indicates how the video was produced and what technologies (i.e., deep learning) were avoided to create it.

Are Shallowfakes Easily Identifiable?

While it's easier to tell if a video is a shallowfake because it is more crudely made than a deepfake, politicians, academicians, and other experts believe it can still cause a lot of damage to the subject. And even if the real video (i.e., before shallowfake alteration) is easy to locate on the Internet, the less discerning could still fall for and spread fake content without thinking twice.

Are Deepfakes and Shallowfakes Covered by Existing Cybercrime Laws?

California has made deepfake distribution illegal since 2019. But politicians did admit that enforcing the said law (i.e., AB 730), which makes circulating doctored videos, images, or audio files of politicians illegal within 60 days of an election, is hard to implement.