Code: 20CS4701A

PVP-20

PRASAD V POTLURI SIDDHARTHA INSTITUTE OF TECHNOLOGY

(AUTONOMOUS)

IV B.Tech- I Semester- Regular Examinations- OCTOBER 2024

DEEP LEARNING

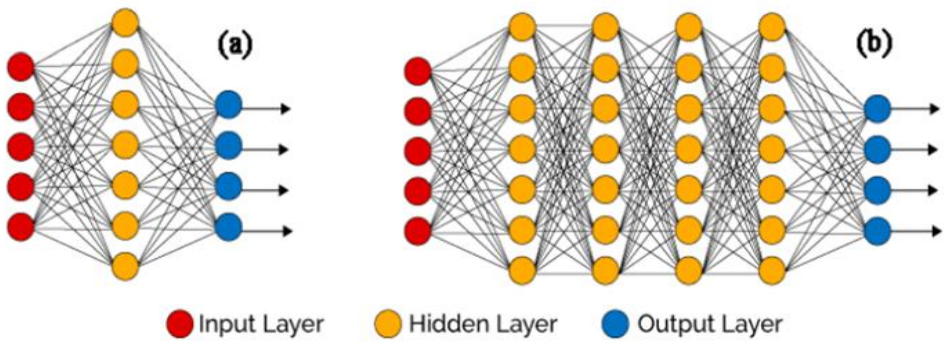
Duration: 3 Hours **Max. Marks :** 70

UNIT-I

1 A) Describe the concept of Deep Learning. Explain the Historical evolution and how it differs from Traditional Machine Learning Algorithms [CO1-L2] [7M]

Ans: Deep learning is a **subset of Machine Learning** that uses **multi-layered neural networks**, called **deep neural networks. The** term **"deep"** usually refers to the **number of hidden layers in the neural network.** Models are trained by using a **large set of labeled data** and **neural network architectures** that **contain many layers. Deep Learning Models** can **recognize complex patterns** in **pictures, text, sounds, and other data** to produce accurate **insights and prediction**. It improves the ability to **classify, recognize, detect and describe using data.**

For Example, we have to find out the sentences- the first layer of nodes might learn to identify the letters, the second layer might learn to identify the words, and the third layer might learn to identify sentences etc. Like this we may increase the hidden layers in the Artificial Neural Network.



(a) Simple neural network architecture; (b) Simple architecture of deep neural network (DNN)

Historical Evolution of Deep Learning:

- **1957 - Frank Rosenblatt** submitted a paper titled **'The Perceptron: A Perceiving and Recognizing Automaton'**, which consisted of an **algorithm or a method for pattern recognition** using a **two-layer neural network.**

- **1965 - Alexey Ivakhnenko and V.G. Lapa** developed the **first working neural network** and **Alexey Ivakhnenko created an 8-layer deep neural network in 1971** which was demonstrated in the **computer identification system, Alpha**. This was the **actual introduction to deep learning**

- **1980 - Kunihiko Fukushima** developed the '**Neocognitron'**, an **Artificial deep neural network** with **multiple and convolutional layers to recognize visual patterns.**

- **1985 - Terry Sejnowski** created **NETtalk**, a **program which learnt how to pronounce English words.**

- **1989 - Yann LeCun**, using **convolution deep neural network**, developed a system which could **read handwritten digits.**

- **Mid-2000s:** The term **"deep learning"** begins to gain popularity after a paper by **Geoffrey Hinton and Ruslan Salakhutdinov** showed **how a many-layered neural network could be pre-trained one layer at a time.**

- **2009 -** As deep learning models require a **tremendous amount of labelled data** to **train themselves in supervised learning**, **Fei-Fei Li** launched **ImageNet**, which is a **large database of labelled images.**

- **2012 -** The results of '**The Cat Experiment**' conducted by **Google Brain** were released. This experiment was based on **unsupervised learning** in which the **deep neural network** worked with **unlabelled data to recognize patterns and features in the images of cats**. However, it could only recognize 15% of images correctly.

- **2014 -** Facebook puts **deep learning technology** – called **DeepFace** – into operations to **automatically tag and identify Facebook users in photographs.**

Difference between Traditional Machine Learning Algorithms:

S. No	Machine Learning	Deep Learning
1	ML is invented in the Year 1950	ML is invented in the Year 1970
2	Apply statistical algorithms to learn the hidden patterns and relationships in the dataset.	Uses artificial neural network architecture to learn the hidden patterns and relationships in the dataset.
3	Can work on the smaller amount of dataset	Requires the larger volume of dataset compared to machine learning
4	Better for the low-label task.	Better for complex task like image processing, natural language processing, etc.
5	Takes less time to train the model.	Takes more time to train the model.
6	Automatically Extracted Features from data using Algorithms	Need not having separate feature extraction. Here given input to Networks
7	Machines can take decisions on their own, based on Past Data	Machines can take decisions with the help of Artificial Neural Networks.
8	Less complex and easy to interpret the result.	More complex, it works like the black box interpretations of the result are not easy.
9	It can work on the CPU or requires less computing power as compared to deep learning.	It requires a high-performance computer with GPU.
10	Data is depends on Labeled and Unlabeled Data for Training and Testing	Requires Large amount of Labeled Data to train complex models

1 B) Compare and Contrast any three common Activation Functions [CO1-L2][7M]

Ans: An **Activation Function** is a **mathematical operation** applied to the **output of a neuron** in a neural network. It determines **whether a neuron should be activated or not**, based on the **weighted sum of its inputs.** Activation functions provide **non-linear properties** to the neural network**.**

There are different types of Non Linear Activation Functions:

(a) Sigmoid Activation Function

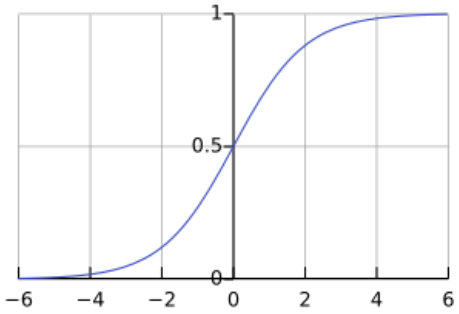
(b) Tanh Activation Function

(c) ReLU Activation Function

(d) Softmax Activation Function

A) Sigmoid Activation Function: This function is a probabilistic approach towards decision making and output range is between 0 and 1. The Sigmoid function, often represented as **(x),** features an **S-shaped curve.** Sigmoid finds the binary classification problems, like logistic regression, where outputs represent probabilities. The sigmoid function, also known as the **squashing function**, takes the **input from the previously hidden layer** and **squeezes it between 0 and 1.** Such as "**yes" or "no", or 1 or 0.** The sigmoid function can **output a probability between 0 and 1,** which can be used to predict the likelihood of a particular class.

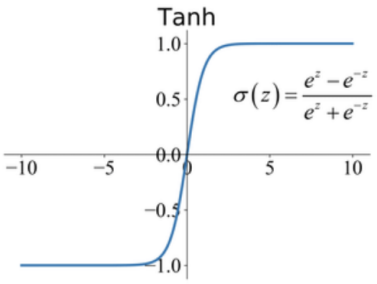
$$f(x) = \frac{1}{1 + e^{-x}}$$

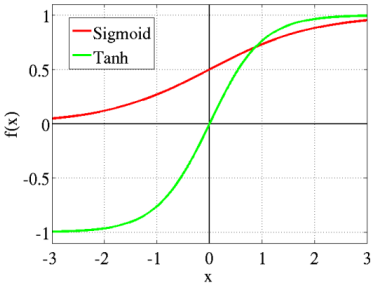


B) Hyperbolic Tangent Activation Function(Tanh): It overcomes the disadvantage of the sigmoid activation function by *extending the range to include -1 to 1*. It is also having the same **S-shape** with the difference in **output range of -1 to 1.** In Tanh, the larger the input (more positive), the closer the output value will be to 1.0, here as the smaller the input (more negative), the closer the output will be to -1.0.

$$Tanh$$

$$f(x) = \frac{(e^{x} - e^{-x})}{(e^{x} + e^{-x})}$$



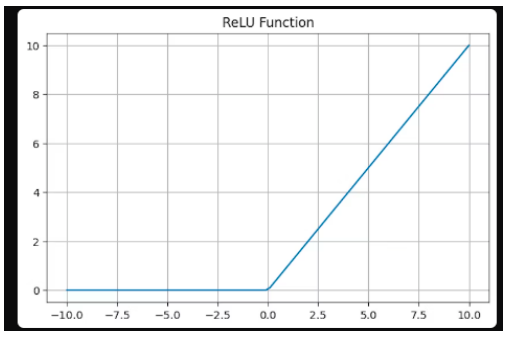


Tanh function can be used when the input data has a range between **negative and positive values.** For instance, it can be used in Neural Networks which is made for sentiment analysis, it can be used to model the sentiment of text data, where negative sentiment is represented by negative values, positive sentiment is represented by positive values, and Neutral can be represented by zero or close to zero.

C) ReLu Activation Function: ReLU is nothing **but Rectified Linear Unit**. ReLU is a **piecewise linear function** that results in the **input directly if it is positive**. Otherwise**, it outputs zero.**

Equation :- It gives an output x if x is positive and 0 otherwise.

$$f(x) = \max(0, x)$$



- **ReLU is most commonly** used in **Large Deep Neural Networks** with **lots and lots of hidden layers**. Since ReLU only **requires some threshold operations**, it is computationally **efficient** even though the **dataset and the network are large.** Also, ReLU is a great choice to **avoid the Vanishing Gradient Problem**

- This is the most frequently used activation unit in deep learning. **R(x) = max(0, x) . Thereby, if x < 0, R(x) = 0 and if x ≥ 0, R(x) = x**.

- Its notable efficiency makes it especially advantageous in **Convolutional Neural Networks (CNNs)** and other deep learning models, leading to its **adoption in advanced object detection frameworks such as YOLO** (You Only Look Once) **etc.**

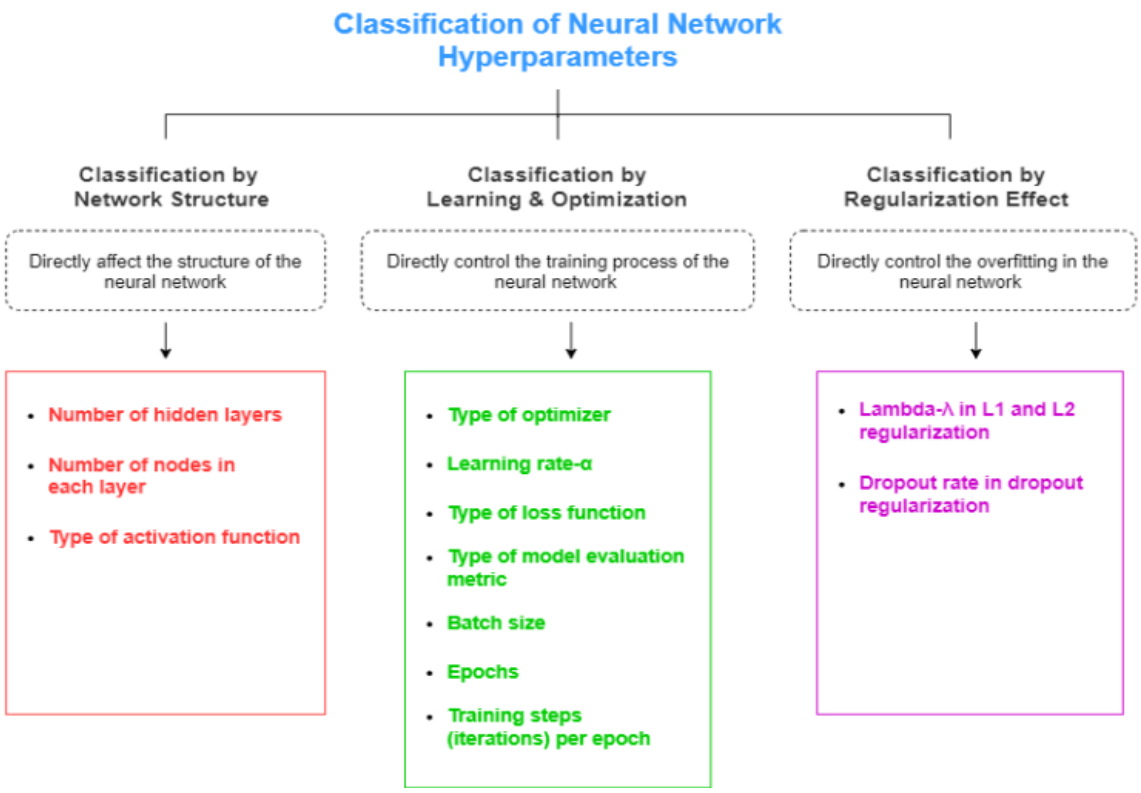
- **Nature :- It is non-linear**, which means we can easily **back propagate the errors** and have **multiple layers of neurons** being activated by the **ReLU function. The main catch here is that the ReLU function does not activate all the neurons at the same time.**

(OR)

2 A) Discuss the significance of Hyper Parameters in performance of the model training [CO1-L2][7M]

Ans: Hyperparameters are those parameters that are explicitly defined by the user to control the learning process. **Hyperparameters** determine **key features** such as **model architecture**, **learning rate**, and **model complexity.** Examples of hyperparameters include the **number of nodes** and **layers in a neural network** and the **number of branches in a decision tree**.

- **Hyperparameters control many aspects** of **DL algorithms.**
  - They can decide the **time and computational cost** of **running the algorithm.**
  - They can **define the structure of the neural network model**
- They affect the **model's prediction accuracy. In other words, hyperparameters control the behavior and structure of the neural network models.**
- **The Main Key points used in Hyperparameters are:**
  - Main parameters of the NN is **W and b**
  - Hyper parameters (parameters that control the algorithm) are like:
    - **Learning rate.**
    - Number of **iterations.**
    - Number of **hidden layers L.**
    - Number of **hidden units n**.
    - Choice of ***activation functions***.
    - Check the Training and Testing



- **Learning Rate:** It is a Hyperparameter that provides the model a scale of how much model **weights should be updated, to minimize the Loss Function.**
- **Optimizer:** In deep learning, optimizers are crucial as algorithms that **dynamically adjust the model's parameters** to **minimize the loss function**.
- There are **various optimization techniques** to **change model weights and learning rates,** like **AdaGrad, RMSProp, AdaDelta, and Adam.**
- **Adagrad** stands for **Adaptive Gradient Optimizer.** It is used to **reduce the loss function** with **respect to the weights.** The **weight updating formula is as follows**:

$$(\mathrm{w})_{\mathrm{new}} = (\mathrm{w})_{\mathrm{old}} - \eta \frac{\partial \mathrm{L}}{\mathrm{dw}(\ \mathrm{old}\ )}$$

Based on **iterations, this formula can be written as:**

$$w_t = w_{t-1} - \eta \frac{\partial L}{\partial w(t-1)}$$

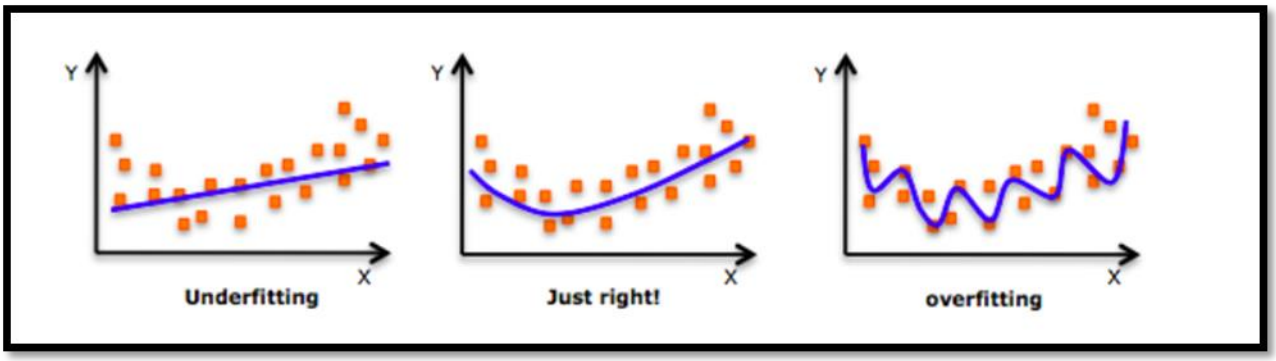
where

w(t) = value of w at **current iteration,**

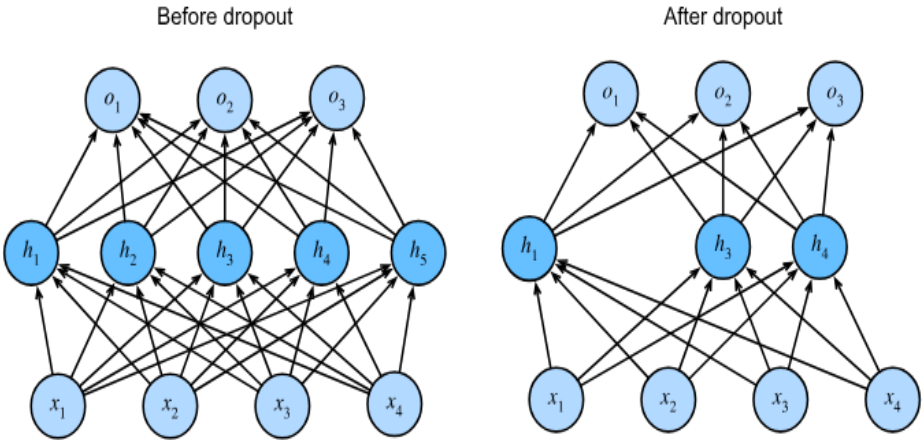
w(t-1) = value of w at **previous iteration and**

η = **learning rate.**

Regularization: Regularization is a set of methods for **reducing overfitting in Deep Learning**



- **Just have a look at the above figure,** and we can immediately predict that once we try to **cover every minutest feature of the input data,** there can be **irregularities in the extracted features,** which can **introduce noise in the output.** This is referred to as **"Overfitting".**
- This may also happen with the **lesser number of features extracted** as some of the **important details might be missed out.** This will **leave an effect on the accuracy of the outputs produced**. This is referred to as **"Underfitting".**
- To eliminate this, **regularization is used**, in which we have to make the **slightest modification in the design of the neural network**, and we can **get better outcomes.**
- **Dropout:** Dropout was **introduced by "Hinton et al"** and this method is now very popular. It consists of **setting to zero** the **output of each hidden neuron in chosen layer** with **some probability** and is proven to be **very effective in reducing overfitting.**

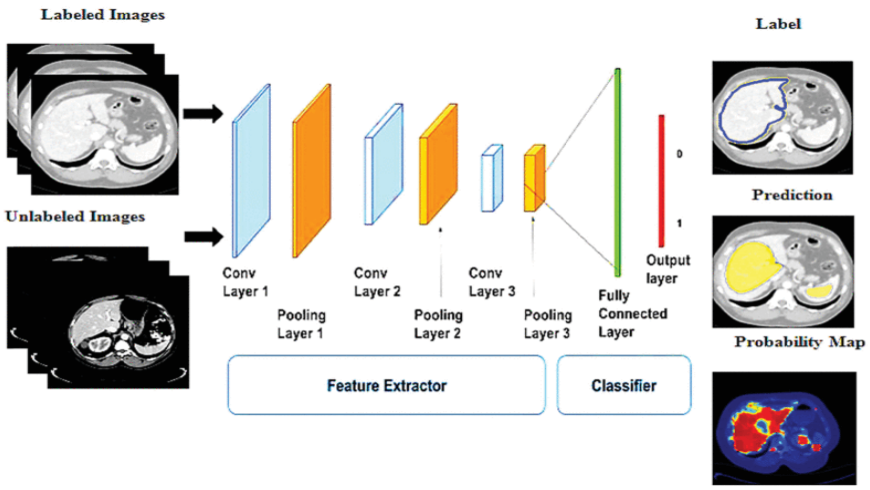


- To achieve **dropout regularization**, some **neurons in the artificial neural network are randomly disabled.** That prevents them from being **too dependent on one another as they learn the correlations.**

---

2 B) Explain at least 3 real world applications where deep learning has shown significant process. [CO1-L2][7M]

Ans: **1. Medical Care:** Deep learning helps in detecting cancer cells and analyzing the MRI images to give elaborative results. Google has made **Google AI eye doctor software**. It examines **retina scans and identifies diabetic retinopathy**, which can **cause blindness**. Suppose we have **taken Kidney Disease Prediction it will follow the structure.**



This image, **we can take Convolution Neural Networks are used for Feature Extraction and Classification**.

2. Visual Translation: With deep learning, **identification of text on the images is possible**. Once identification completes, it **translates the text immediately and recreates the image with translated text**.

For example, Suppose you **visit an unknown country** whose **local language is not known to you**. An app like **google translator converts the text of an image in an understandable** language.

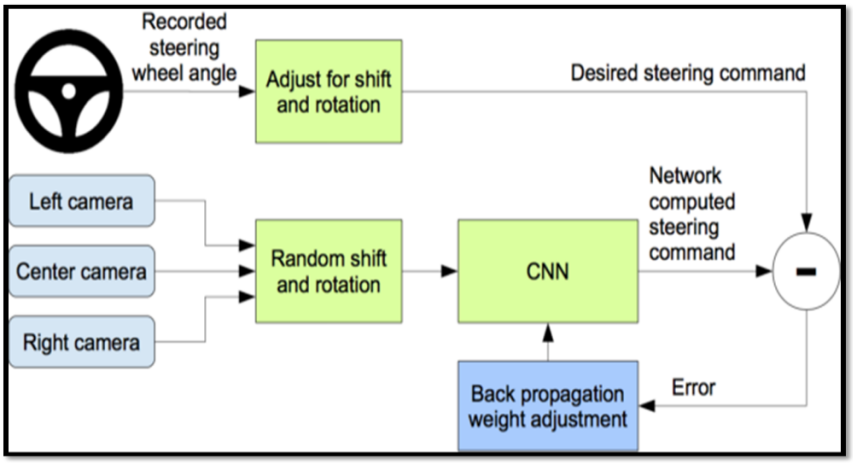
Figure 1: Translation from English to Spanish of the English sentence "the cat likes to eat pizza"

With the power of deep learning, **Neural Machine Translation (NMT)** has arisen as the **most powerful algorithm** to perform this task. This state-of-the-art algorithm is an application of deep learning in which **datasets of translated sentences** are used to **train a model capable of translating between any two languages. This conversion is composed by using Two Recurrent Neural Networks(RNN)**

3. Self Driving Car: Google has made an **amazing self-driven car**. This car operates on a **combination of sensors and software**.

Car can classify objects, people, traffic signs and signals. It also detects the road works. It uses **Lidar Technology.**

- A self-driving car **(sometimes called an autonomous car or driverless car)**.
- **Tesla**, the most **popular car manufacturing company** is **working on self-driving car**.



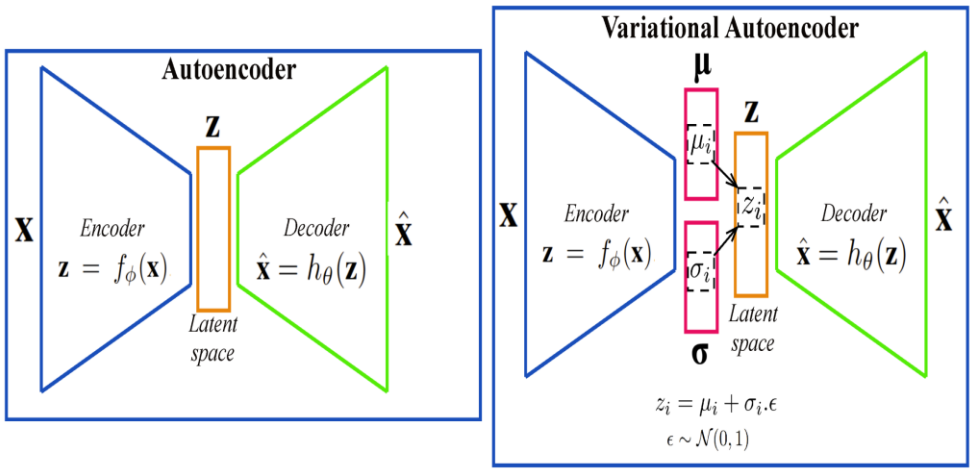
UNIT-II

3 A) Distinguish the key differences between a Variational Auto Encoder (VAE) and a Traditional Auto Encoder (TAE)? How does a VAE enable generating new data samples. [CO2-L4] [7M]

Ans:

Autoencoder	Variational Autoencoder
Used to **generate a compressed transformation** of **input in a latent space**	Enforces conditions on the latent variable to be the **unit norm**
The latent variable is **not regularized**	The latent variable in the **compressed form** is **mean and variance**
Picking a random latent variable will generate **garbage output**	A random value of latent variable generates **meaningful output at the decoder**
Latent variable has **deterministic values**	The input of the decoder is **stochastic and is sampled from a Gaussian with mean and variance** of the output of the encoder.
The **latent space lacks the generative capability**	The **latent space has generative capabilities.**

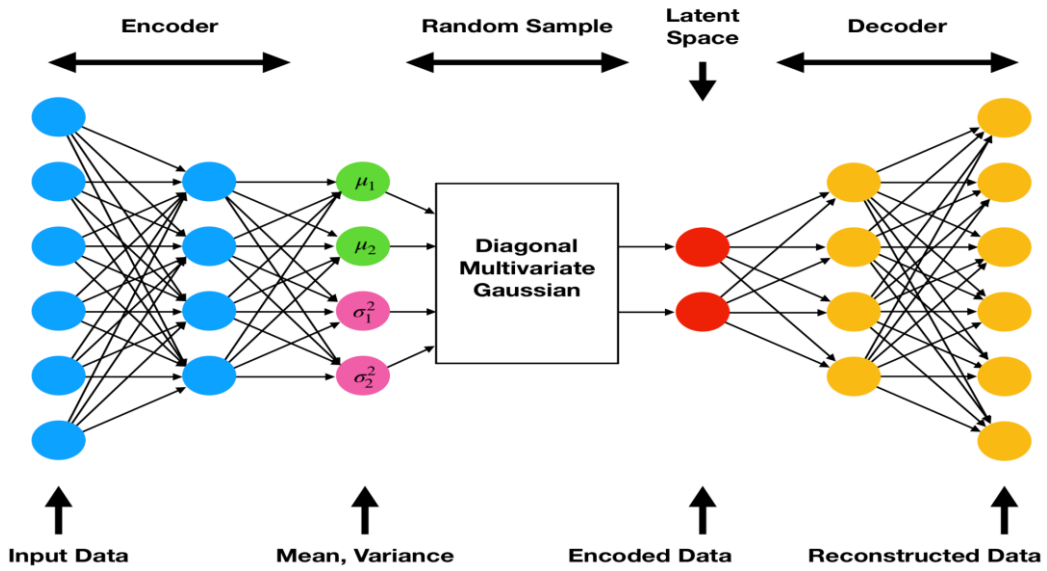
Mathematically,



The VAE is similar to compression and Denoising autoencoders in that they are all trained in an unsupervised manner to reconstruct inputs. Variational Autoencoders (VAEs) are **generative models explicitly designed to capture the underlying probability distribution** of a **given dataset** and **generate novel samples.**

- However, the mechanisms that the V**AEs use to perform training are quite different.** In a **compression/denoising autoencoder**, **activations are mapped to activations throughout the layers**, as **in a standard neural network;** comparatively, a ***VAE uses a probabilistic approach for the forward pass.***

- **First,** we map **each point x in our dataset to a low-dimensional vector** of **means μ(x) and variances σ(x)2** for a **diagonal multivariate Gaussian distribution.**



The **encoder-decoder architecture** lies at the heart of **Variational Autoencoders (VAEs),** distinguishing them from **traditional autoencoders.** The **encoder network takes raw input data and transforms it into a probability distribution within the latent space**.

i.e, the **latent code generated by the encoder** is a **probabilistic encoding,** allowing the **VAE to express not just a single point in the latent space** but a **distribution of potential representations**.

---

3 B) Demonstrate the fundamental idea behind Deep Belief Networks (DBNs)?

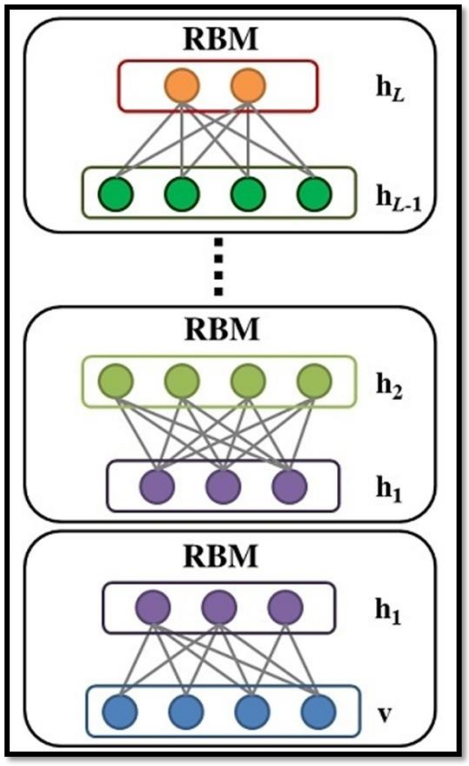
[CO2-L3] [7M]

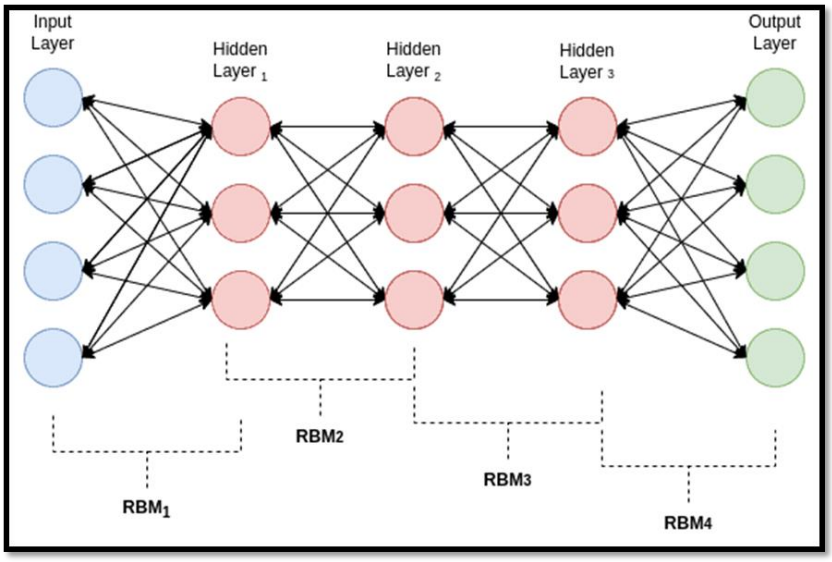
Ans: We create *Deep Belief Networks (DBNs)* to address issues with classic neural networks in deep layered networks.

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

- A DBN is a deep-learning architecture introduced by Geoffrey Hinton in 2006. Deep Belief Networks (DBNs) are a type of deep learning architecture combining unsupervised learning principles and neural networks.
- Deep Belief Networks (DBNs) are sophisticated artificial neural networks used in the field of deep learning.
- 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. They are **composed of layers** of **Restricted Boltzmann Machines (RBMs),** which are **trained one at a time in an unsupervised manner.**
- **Several Restricted Boltzmann Machines (RBM)** can be **stacked** and **trained in a greedy manner** to **form Deep Belief Network architecture.**
- i.e, It is a **composition of Stack** of **Unsupervised Neural Network** such as **Restricted Bolzmann Machine (RBM). Each RBM** has a **Visible Layer (Input)** and **Hidden Layer (Output)** . Here the **Hidden Layer in Stack1** is the **Visible Layer for the Stack2.etc.**
- DBNs work similarly to **traditional multi-layer perceptrons (MLPs)** and offer certain benefits over them, including **faster training and better weight initialization.**

Structure of DBNs:





- In the DBN, we have a hierarchy of layers. The top two layers are the associative memory, and the bottom layer is the visible units. The arrows pointing towards the layer closest to the data point to relationships between all lower layers.

(OR)

4 A) Construct the architecture of Generative Adversarial Networks (GANs) with an example [CO2-L2][7M]

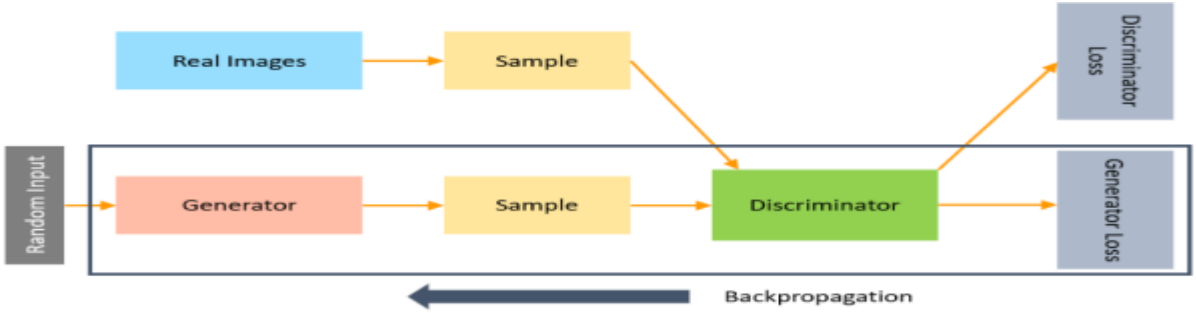
Ans: **Generative Adversarial Networks** : Generative Adversarial Networks (GANs) are a powerful class of neural networks that are used for unsupervised learning. It was developed and introduced by Ian J. Goodfellow in 2014.

- 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. To understand the term GAN let's break it into separate three parts •
- **Generative –** To learn a generative model, which describes how data is generated in terms of a probabilistic model. In simple words, it explains how data is generated visually.
- **Adversarial –** The training of the model is done in an adversarial setting.
- **Networks –** use deep neural networks for training purposes. GAN consists of 2 models that automatically discover and learn the patterns in input data.
- The two models are known as Generator and Discriminator. They compete with each other to scrutinize, capture, and replicate the variations within a dataset.

What is a Generator? A Generator in GANs is a neural network that creates fake data to be trained on the discriminator. It learns to generate plausible data. The generated examples/instances become negative training examples for the discriminator. It takes a fixed-length random vector carrying noise as input and generates a sample.

The main aim of the Generator is to make the discriminator classify its output as real. The part of the GAN that trains the Generator includes:

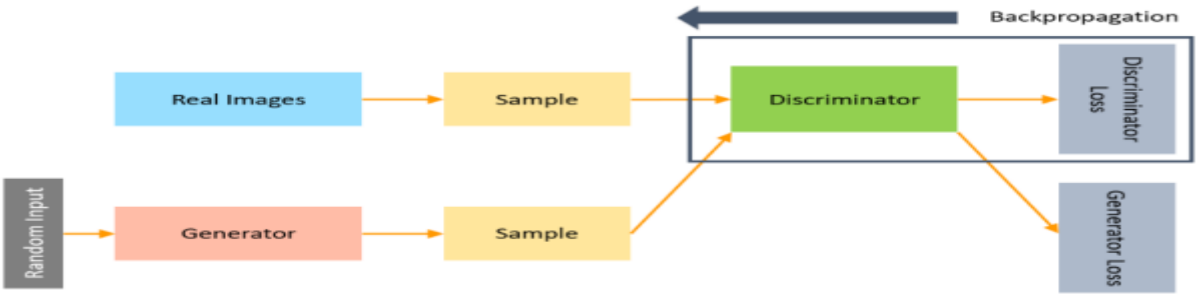
- noisy input vector
- generator network, which transforms the random input into a data instance
- discriminator network, which classifies the generated data
- generator loss, which penalizes the Generator for failing to dolt the discriminator



What is a Discriminator? The Discriminator is a neural network that identifies real data from the fake data created by the Generator. The discriminator's training data comes from **different two sources**:

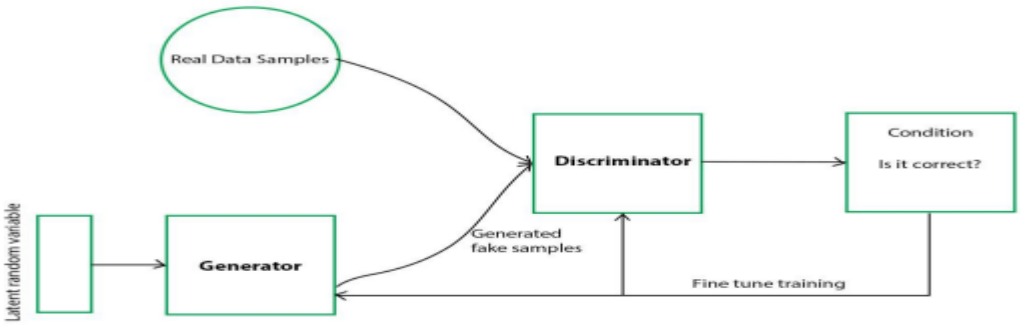
1. The real data instances, such as real pictures of birds, humans, currency notes, etc., are used by the Discriminator as positive samples during training.

2. The fake data instances created by the Generator are used as negative examples during the training process.



While training the discriminator, it connects to two loss functions. During discriminator training, the discriminator ignores the generator loss and just uses the discriminator loss. In the process of training the discriminator, the discriminator classifies both real data and fake data from the generator. The discriminator loss penalizes the discriminator for misclassifying a real data instance as fake or a fake data instance as real. The discriminator updates its weights through backpropagation from the discriminator loss through the discriminator network.

Working of GAN:



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 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:

It can be mathematically described by the formula below:

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

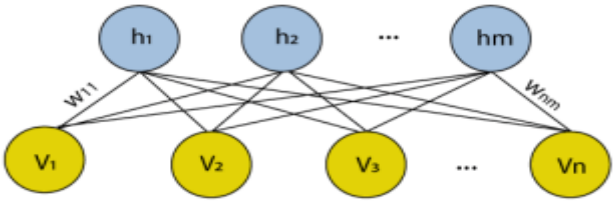
$$V(D,G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z))]$$

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

---

4 B) Describe the architecture and Training process of a Restricted Boltzmann Machine (RBM) and provide one example [CO2-L2][7M]

Ans: Restricted Boltzmann Machine is an undirected graphical model that plays a major role in Deep Learning Framework in recent times. It is an algorithm which is useful for dimensionality reduction, classification, regression, collaborative filtering, feature learning, and topic modeling. The "restricted" part of the name "Restricted Boltzmann Machines" means that connections between nodes of the same layer are prohibited (e.g., there are no visible-visible or hidden connections along which signal passes).



Network layout There are five main parts of a basic RBM:

• Visible units

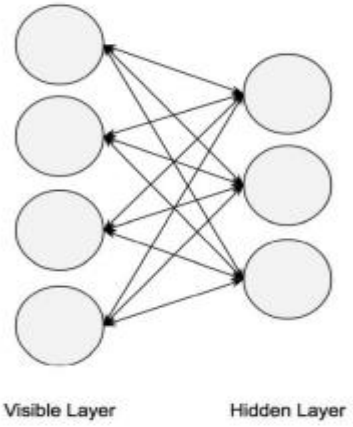
• Hidden units

• Weights

• Visible bias units

Hidden Bias Units

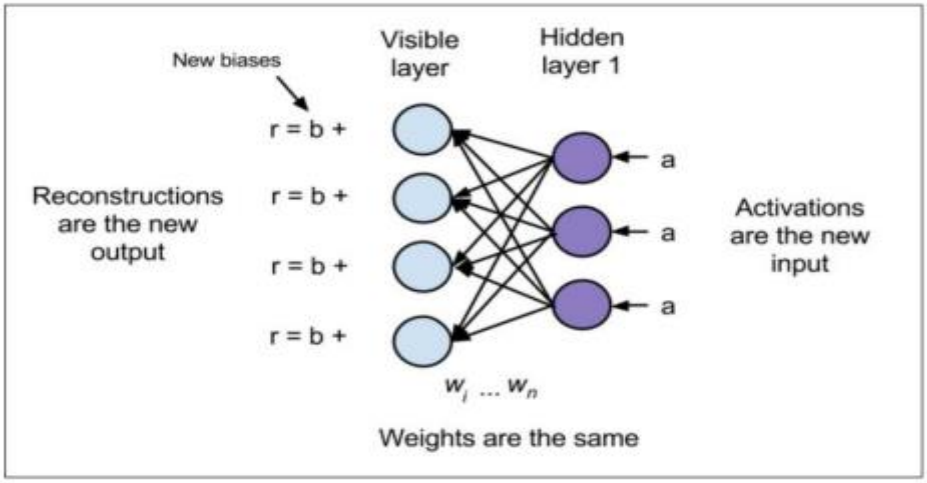
1. Layers: Restricted Boltzmann Machines are shallow, two-layer neural nets that constitute the building blocks of deep-belief networks. The first layer of the RBM is called the visible, or input layer, and the second is the hidden layer. Each circle represents a neuron-like unit called a node. The nodes are connected to each other across layers, but no two nodes of the same layer are linked.



2. Visible and hidden layers: In an RBM, every single node of the input (visible) layer is connected by weights to every single node of the hidden layer, but no two nodes of the same layer are connected. The second layer is known as the "hidden" layer. Hid- den units are feature detectors, learning features from the input data. Each node performs computation based on the input to the node and outputs a result based on a stochastic decision whether or not to transmit data through an activation.

3. Connections and Weights: All connections are visible-hidden; none are visible-visible or hidden-hidden. The edges represent connections along which signals are passed. Loosely speaking, those circles, or nodes, act like human neurons. They make decisions about whether to be on or off through acts of computation. "On" means that they pass a signal further through the net; "off" means that they don't. Usually, being "on" means the data passing through the node is valuable; it contains information that will help the network make a decision. Being "off" means the net- work thinks that particular input is irrelevant noise.

1. **Biases:** There is a set of bias weights ("parameters") connecting the bias unit for each layer to every unit in the layer. Bias nodes help the network better triage and model cases in which an input node is always on or always off.
2. **Training:** The technique known as pre training using RBMs means teaching it to reconstruct the original data from a limited sample of that data. That is, given a chin, a trained net- work could approximate (or "reconstruct") a face. RBMs learn to reconstruct the input dataset.
3. **Reconstruction:** Deep neural networks with unsupervised pretraining (RBMs, autoencoders) perform feature engineering from unlabeled data through reconstruction.



Example: We can visually explain reconstruction in RBMs by looking at the MNIST dataset. The MNIST dataset is a collection of images representing the handwritten numerals 0 through 9.

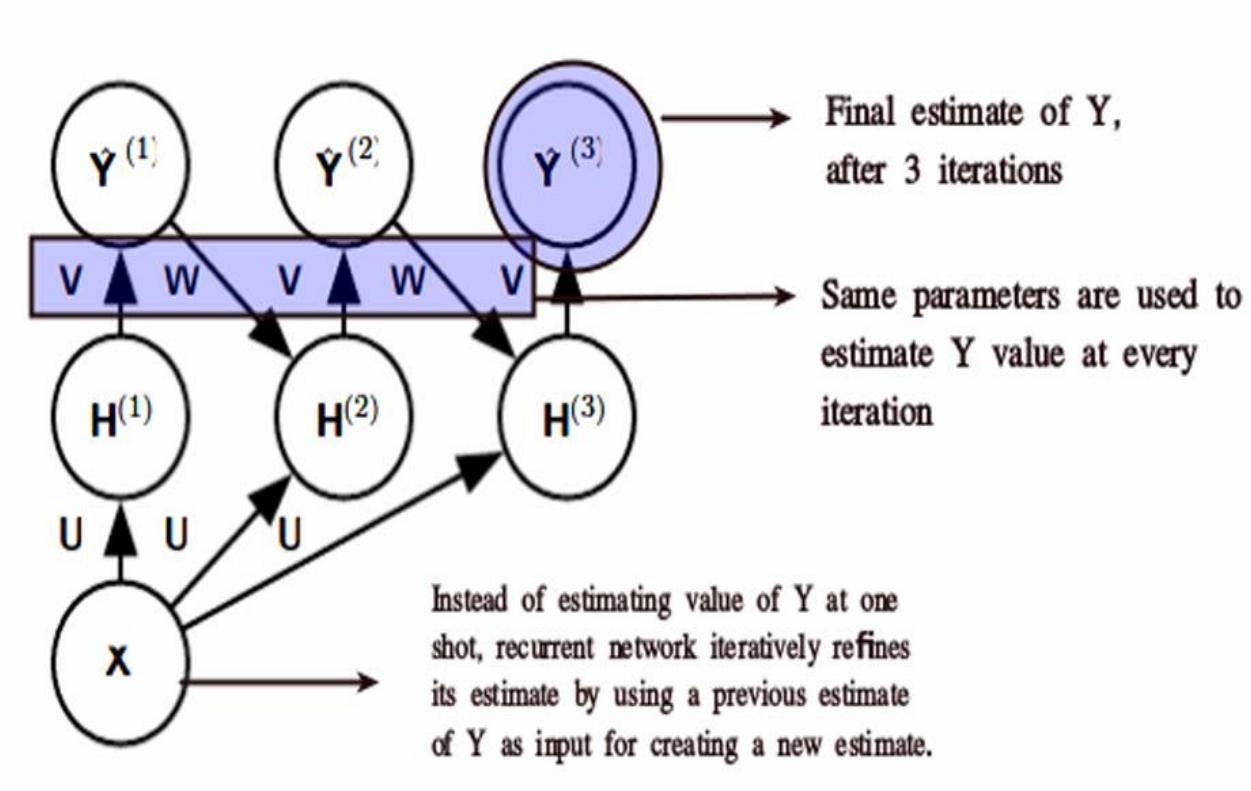
The training dataset in MNIST has 60,000 records and the test dataset has 10,000 records. If we use a RBM to learn the MNIST dataset, we can sample from the trained network to see how well it can reconstruct the digits. If the training data has a normal distribution, most of them cluster around a central mean, or average, and become scarcer the further you stray from that average.

UNIT-III

5 A) What are structured output in the context of CNNs and why are they important in tasks like image segmentation or object detection? Explain with an Example [CO3-L4] [7M]

Ans: Convolutional networks can be trained to output high-dimensional structured output rather than just a classification score.

- To produce an output map as same size as input map, only same-padded convolutions can be stacked.
- The output of the first labelling stage can be refined successively by another convolutional model.
- If the models use tied parameters, this gives rise to a type of recursive model as shown below. (H^1, H^2, H^3 share parameters or Hidden Layers)



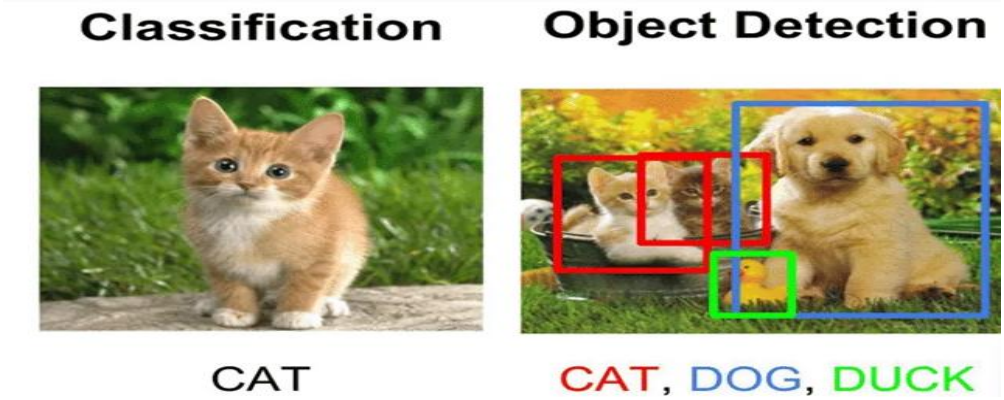
Variable	Description
X	Input image tensor
Y	Output of Classification or Object Detection
H	Hidden representation
U	Kernel
V	Produce Estimated Labels
W	Output Y to provide the input to H

- The **input layer receives the raw data,** such as **an image, and serves as the starting point for processing.**
- Then the so-called **Hidden Layers:** here the Hidden Layers are **Convolutional Layer, Activation Function, Pooling Layer.**
- The term "**hidden layers**" in a CNN refers to the **intermediate layers** between **the input and output layers**. Although the layers **are not directly observable from the input or output**, you have the **authority to shape their structure and functionality .**
- By defining **the convolutional layers, activation functions**, **pooling layers**, you **guide the network's ability** to **extract meaningful features** and **make accurate predictions.**
- CNNs are especially useful for computer vision tasks such as image recognition and classification because they are designed to learn the spatial hierarchies of features by capturing essential features in early layers and complex patterns in deeper layers. One of the most significant advantages of CNNs is their ability to perform automatic feature extraction or

feature learning. This eliminates the need to extract features manually, historically a labor-intensive and complex process.

- Object detection is the process of identifying and locating objects in an image. The task is to determine the object's class and obtain the exact coordinates of its location.

- CNN-based neural networks play an important role in object detection. They identify features in an image that help identify and locate objects. Convolutional layers in CNNs effectively detect various features, leading to accurate object detection.



---

5 B) Describe the concept of Random or unsupervised features in CNNs

[CO3-L2] [7M]

Ans: Typically, the most expensive part of conv network training is learning the features. There are 3 basic strategies for obtaining convolution kernels without supervised training.

1. Simply initialize randomly: random filters work well in **convolutional networks**. Inexpensive way to **choose the architecture of a convolutional network:**

Evaluate the performance of several convolutional network architecture by **training only the last layer.** Take the **best of these architectures** and **train the entire architecture** using a **more expensive approach**.

2. Design them by hand

3. Unsupervised training of kernels may be performed;

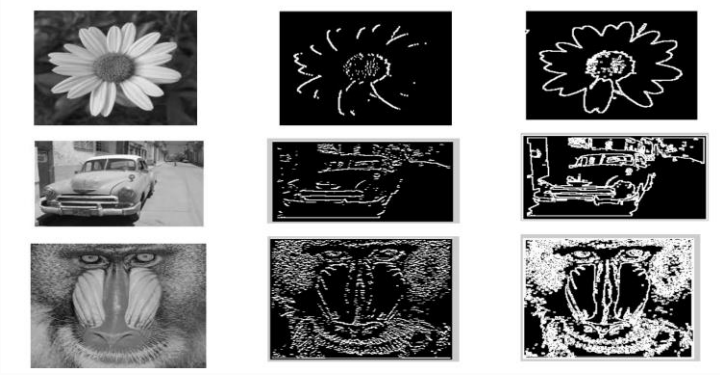
- **Eg: applying k-means clustering** to **image patches** and **using the centroids as convolutional kernels**.
- **Unsupervised pre training may offer regularization effect** (not well established). **It may also allow for training of larger CNNs** because of **reduced computation cost.**

- Learning the features from unsupervised criterion allows them to be determined **separately from the classifier layer** at the **top of the architecture.**
- Another approach for CNN training is **greedy layer-wise pretraining** most **notably used in convolutional deep belief network**.

- **<u>For example</u>**, In the case of **multi-layer perceptrons**, starting with the **first layer**, **each layer is trained in isolation**. Once the **first layer is trained, its output is stored** and **used as input for training the next layer, and so on. Instead of training an entire convolutional layer at a time, we can train a model of small patch, we can use the parameters from this patch-based to define the kernels of a convolutional layer.**
- To **reduce the computational cost of training the CNN**, we can use **features not learned by supervised training**.
- **1. <u>Random initialization</u>** has been shown to **create filters** that **features are selective and translation invariant**. This can be used to **inexpensively select the model architecture**.

Randomly initialize several CNN architectures and **just train the last classification layer.** Once a **Features are extracted**, that model can be **fully trained in a supervised manner.**

2. Hand designed kernels may be used; **e.g.** to **detect edges** at **different orientations and intensities.**



3. Unsupervised training of kernels may be performed;

Eg: applying k-means clustering to image patches and using the centroids as convolutional kernels. Unsupervised pre training may offer regularization effect (not well established). It may also allow for training of larger CNNs because of reduced computation cost.

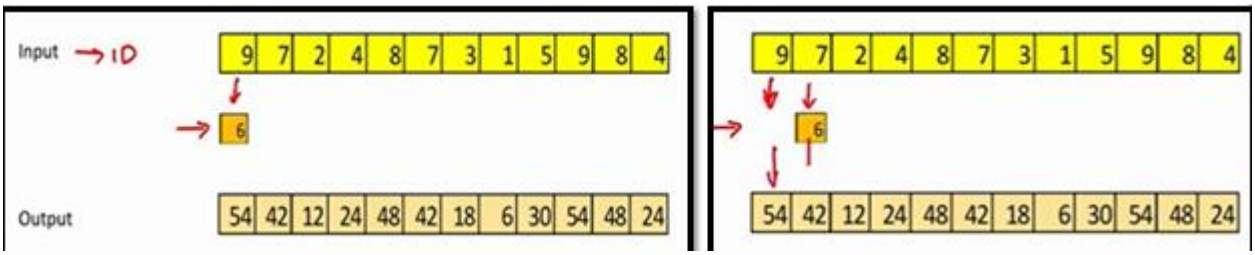
(OR)

6 A) Explain Two efficient convolution algorithms used in CNNs. [CO3-L4] [7M]

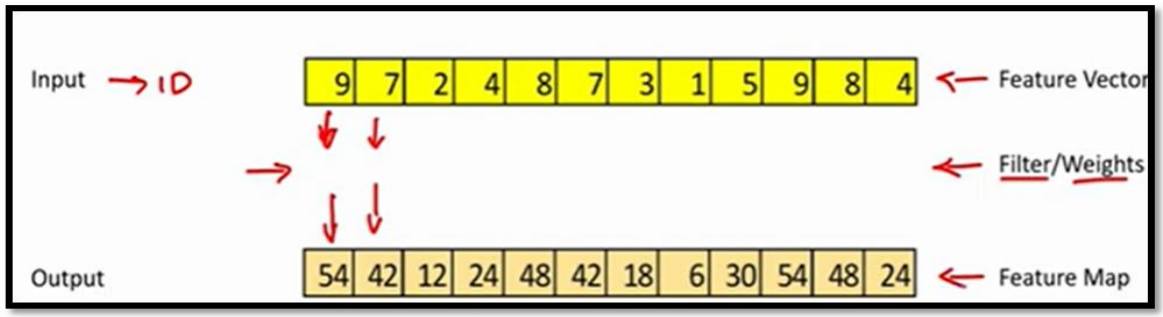
Ans: In the CNN Architecture, There are many Efficient Algorithms are used based on Research occur in Object Detection and Image Classification.

- Here we will show the basic Convolution Algorithms are:
  - 1. One Dimensional (1D) Convolution Algorithm
  - 2. Two Dimensional(2D) Convolution Algorithm
  - 3. Fourier Transform

1. One Dimensional Convolution(ID): 1D CNN can perform **activity recognition task** from **accelerometer data,** such as **if the person is standing, walking, jumping etc.** In **1D convolution example** has **one input channel** and **one output channel. In ID our Input is One Dimensional Array. In the given Example**, we have **taken the Input Vector**, and **simply multiplied with the Number.**

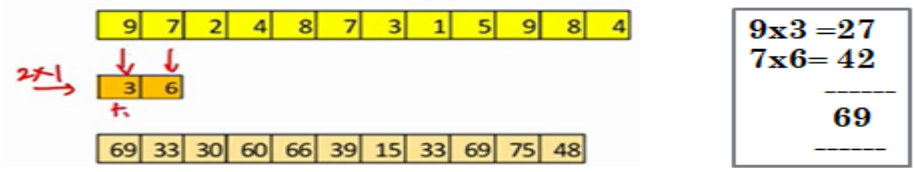


- **(This is the simplest case of 1D Convolution). But we can used this in Deep Learning , we have aware of Feature Vector, Filter or Weights, Feature Map.**

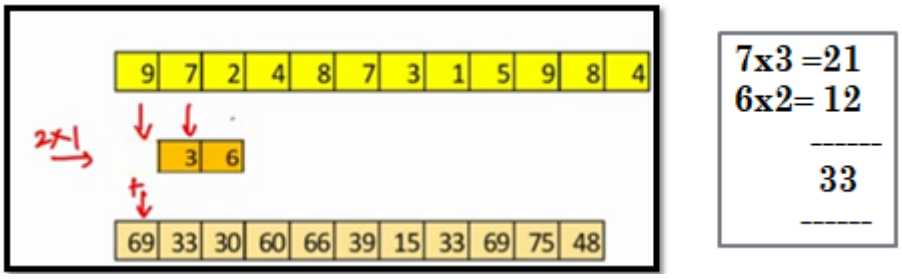


- **Generally, Here the Input size is 12 and Filter size is 1 and Output Size is 12.**
- **But in Deep Learning all can represent by using Matrix Form. So, Input Size is 12 x1 and Filter Size 1x1 and Output Size 12x1.**

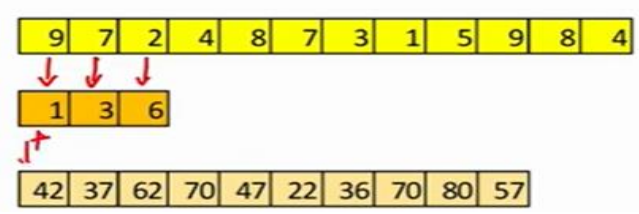
Eg-3: Suppose here we can take the Filter Size 3x 1.



- **Next, we can move this filter to next position (i.e, Right Direction)**

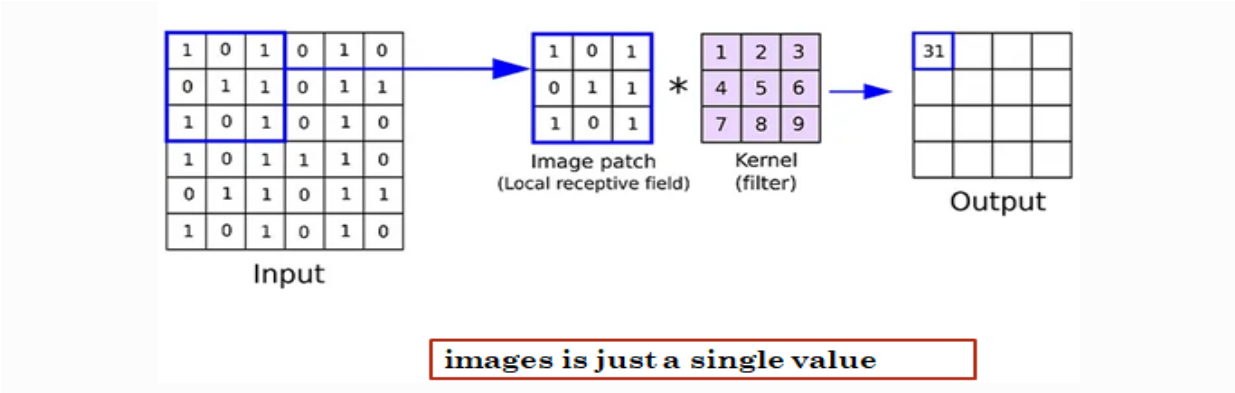


- **<u>Eg-2:</u> Suppose here we can take the Filter Size 3x 1. So here, we can use padding due to loss of information**

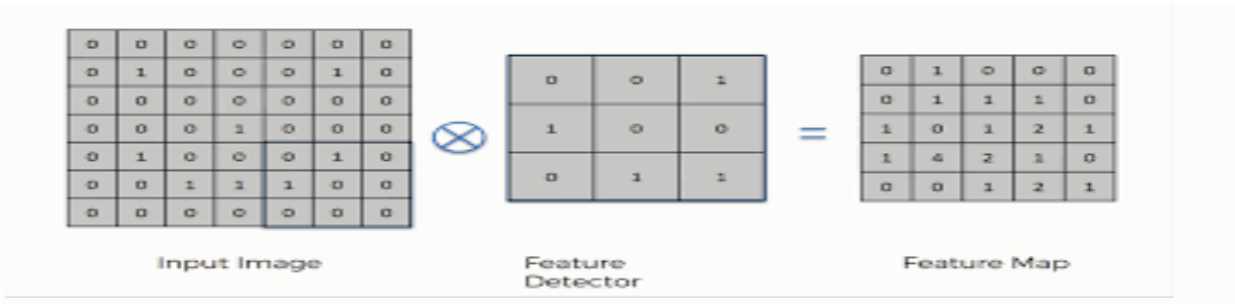


<u>2. Two Dimensional Convolution(2D):</u> In 2D CNN, **kernel moves in 2 directions. Input and output data of 2D CNN is 3 dimensional**. Mostly used **on Image data.** Here **filter comes to play** which select **some part of images and applies dot product on that.**

- A <u>dot product</u> is the **element-wise multiplication between the filter-sized patch** of the input and filter, which is then summed, always resulting in a single value. This filter has to be design to detect specific type of feature in image.



- **After filter iterate through entire image(Input), we get feature map.**



We can change the kernel or filter to detect horizontal or vertical lines as per needed. This feature map help us to identify edges, vertical lines, horizontal lines, bends, etc.

---

6 B) Compare and Contrast Max Pooling and Average Pooling, highlighting Strengths and Weaknesses. [CO3-L4] [7M]

Ans: Pooling layers are one of the building blocks of Convolutional Neural Networks. Where Convolutional layers extract features from images, Pooling layers consolidate the features learned by CNNs.

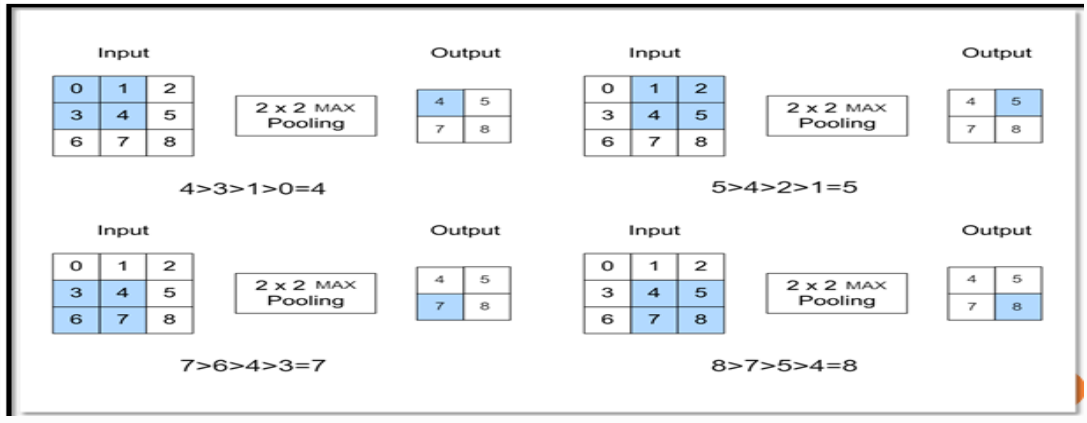
- we use a pooling function to modify the output of the layer further. Its purpose is to gradually shrink the representation's spatial dimension to minimize the number of parameters and computations in the network.
- Pooling layers are used to reduce the dimensions of the feature maps.
- The **size of the pooling operation or filter** is s**maller than the size of the feature map.** This means that the **pooling layer** will **always reduce the size of each feature map by a factor of 2**.

1. Max Pooling: Finally, we can say it selects the maximum valued element from the region captured by the filter in any feature map.

This works by selecting the maximum value from every pool. Max Pooling retains the most prominent features of the feature map, and the returned image is sharper than the original image. i.e, if we talk about image processing the max-pooling helps to extract the sharpest features on the image.

- **Kernel/filter is the size of a matrix** that is **applied over the complete input data.** As the **filter moves across the input**, it **selects the pixel** with the **maximum value to send to the output array**.

For Eg:

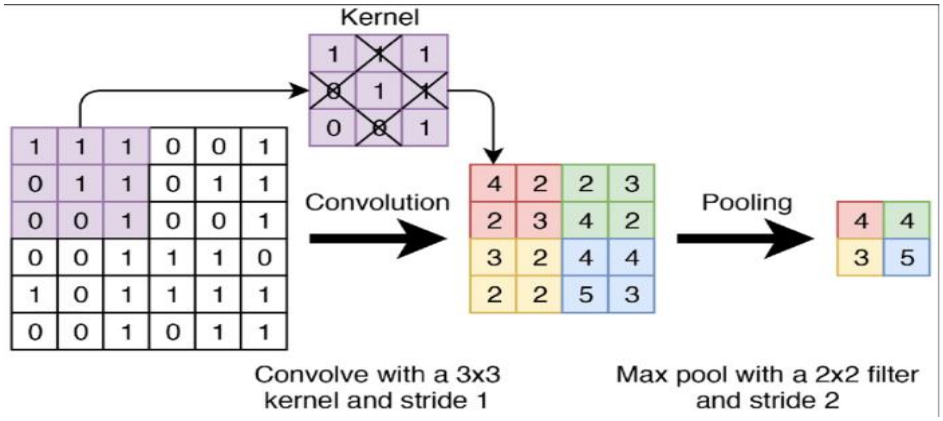


Mathematical Formula: The pooling layer requires 2 hyperparameters

1. Kernel/filter size F and

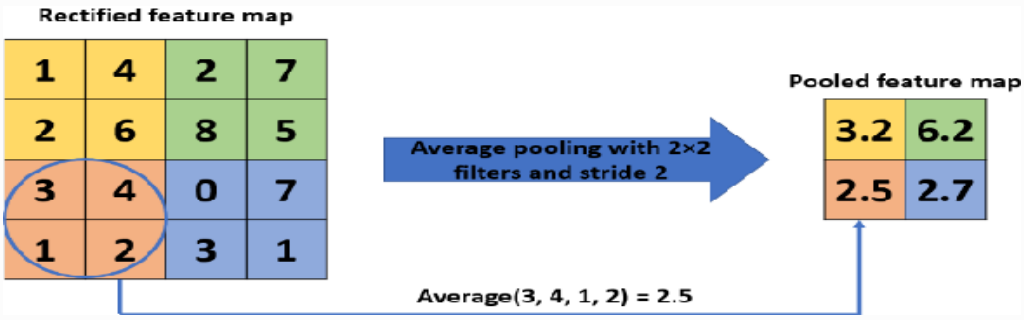
2. Stride S.

On applying the pooling layer over the input volume, output dimensions of output volume will be ((n-f)/s +1) x ((n-f)/s +1)

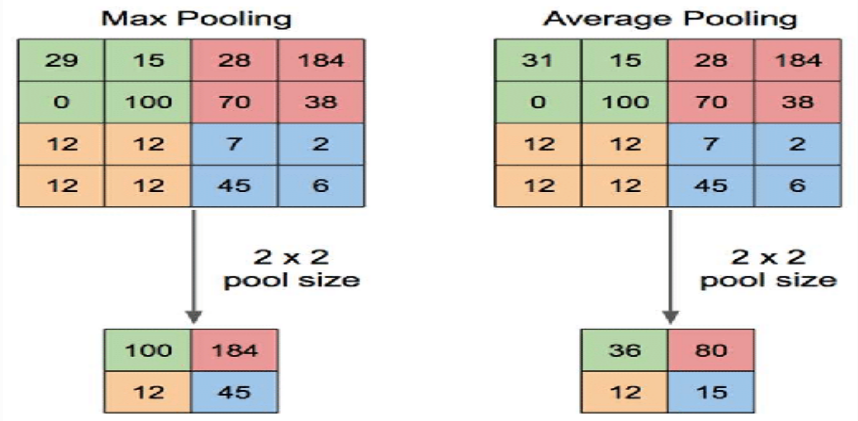


2. Average Pooling: As the filter moves across the input, it calculates the average value within the receptive field to send to the output array. This pooling layer works by getting the average of the pool. Average pooling retains the average values of features of the feature map. Average pooling method smooth's out the image and hence the sharp features may not be identified when this pooling method is used.

With average pooling, the **harsh edges of a picture** are **smoothened,** and this type of pooling layer can used **when harsh edges can be ignored.**



Comparison of Max Pooling and Average Pooling :



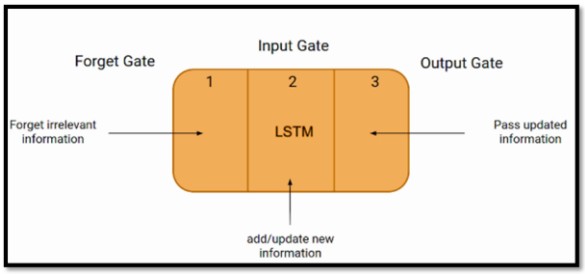
Max pooling selects the brighter pixels from the image. It is useful when the background of the image is dark and we are interested in only the lighter pixels of the image. For example: in MNIST dataset, the digits are represented in white color and the background is black. So, max pooling is used. Similarly, min pooling is used in the other way round.

Average pooling method smoothes out the image and hence the sharp features may not be identified when this pooling method is used.

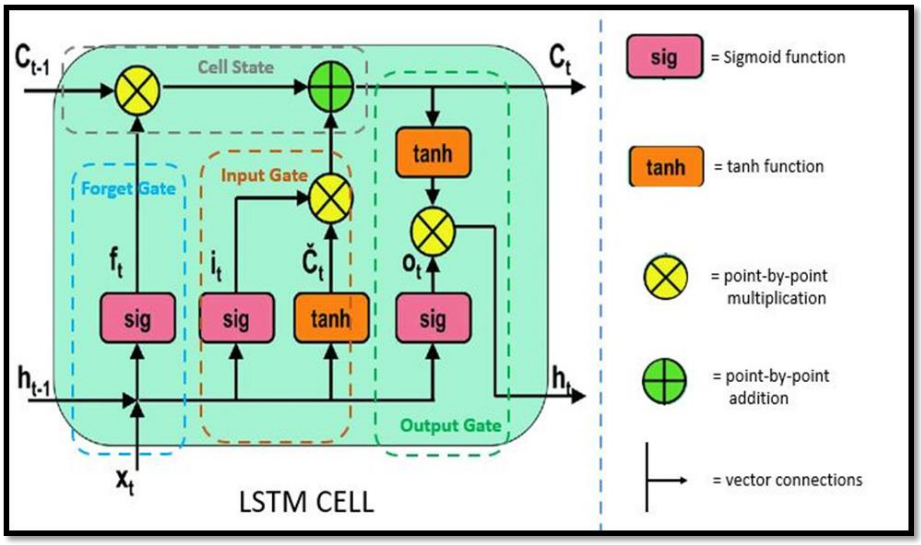
UNIT-IV

7 A) Construct the Architecture of an LSTM cell and how it retains and updates information over time [CO3-L3] [7M]

Ans: **The architecture of LSTM:** The LSTM architectures involves the memory cell which is controlled by three gates:



- **These gates decide** what **information to add to**, **remove from**, and **output from the memory cell**.
- **The input gate controls what information** is **added to the memory cell.**
- **The forget gate controls** what information is **removed from the memory cell.**
- **The output gate controls** what information is **output from the memory cell.**
- **Cell (the memory part of LSTM):** The **cell stores the state of a sequence**, so it **has the ability to either keep** or **forget certain information.**
- The **LSTM maintains a hidden state**, which acts as the **short-term memory of the network**. The **hidden state is updated based on the input,** the **previous hidden state**, and **the memory cell's current state**.
- Here **Information is retained by the cells** and **the memory manipulations** are **done by the gates.**
- Let's assume we have a **sequence of words (w1, w2, w3, ..., wn)** and we are **processing the sequence one word at a time**. Let's denote the **state of the LSTM at time step t** as **(ht, ct),**
- where **ht** is the **hidden state** and **ct** is the **cell state**.
- **ct-1** stands for the **input from a memory cell in time point t;**
  Xt is an **input in time point t;**
- **ht** is an **output in time point t** that **goes to both the output layer** and the **hidden layer in the next time point.**



❖ Here Information is retained by the cells and the memory manipulations are done by the gates.

❖ Let's assume we have a sequence of words (w1, w2, w3, ..., wn) and we are processing the sequence one word at a time. Let's denote the state of the LSTM at time step t as (ht, ct),

❖ where ht is the hidden state and ct is the cell state.

❖ ct-1 stands for the input from a memory cell in time point t;

Xt is an input in time point t;

❖ ht is an output in time point t that goes to both the output layer and the hidden layer in the next time point.

The equation for the forget gate is:

$$f_t = \sigma\left(W_f \cdot \left[h_{(t-1)}, x_t \right] + b_f \right)$$

- where:
  - **Wf** represents the **weight matrix** associated with the **forget gate.**
  - **[ht-1, Xt]** denotes the **concatenation** of the **current input** and **the previous hidden state.**
  - **bf** is **the bias** with the **forget gate.**
  - **σ** is the **sigmoid activation function**.

Input Gate: The **input gate (it)** decides what **new information to store in the cell state**.

It has two parts. A **sigmoid Function** called the **"input gate layer"** decides **which values we'll update**, and **a tanh Function** creates a **vector of new candidate values (Ct~)** that **could be added to the state**. Ie, **First, the information is regulated** using the **sigmoid function** and **filter the values to be remembered similar to the forget gate** using **inputs h_{t-1} and x_t. .**

➢ Then, **a vector** is **created using *tanh* function** that **gives an output from -1 to +1,** which **contains all the possible** values **from h_{t-1} and x_t. At last, the values of the vector** and the **regulated values** are **multiplied to obtain the useful information.**

➢ **The equation for the input gate is:**

$$i_t = \sigma\left(W_i \cdot \left[h_{(t-1)}, x_t \right] + b_i \right)$$

❖ **Candidate Values(Cell State Update):**

$$\tilde{C_t} = \tanh\left(W_c \cdot \left[h_{(t-1)}, x_t \right] + b_c \right)$$

Output Gate: The task of extracting useful information from the **current cell state** to be **presented as output is done by the output gate.** First, **a vector is generated** by **applying tanh function** on **the cell**. Then, the information is regulated **using the sigmoid function** . At last, the **values of the vector** and the **regulated values are multiplied to be sent** as **an output** and **input to the next cell. The equation for the output gate is:**

$$o_t = \sigma\left(W_o \cdot \left[h_{(t-1)}, x_t \right] + b_o \right)$$

Hidden State:

$$h_t = o_t \times \tanh\left(C_t \right)$$

---

7 B) What is the primary role of an Encoder-Decoder Architecture in Sequence- to – Sequence tasks? Discuss with an Example. [CO3-L2] [7M]

Ans: Encoder – Decoder Model:

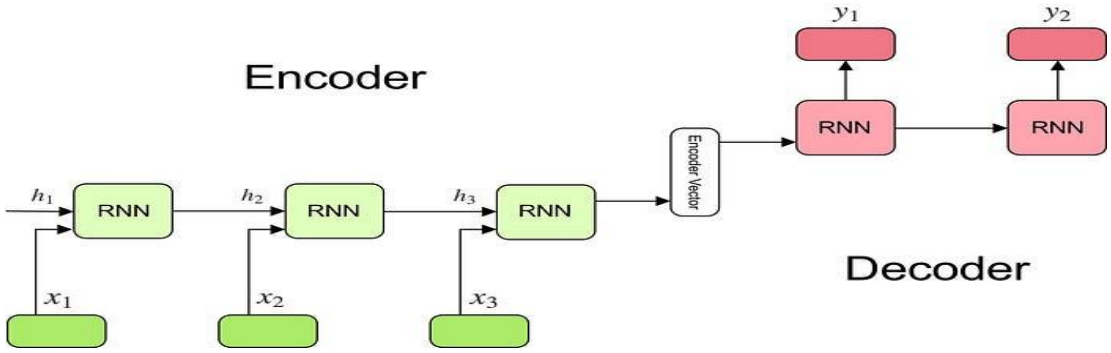
There are three main blocks in the encoder-decoder model,

- Encoder
- Hidden Vector
- Decoder

- The Encoder will convert the input sequence into a single-dimensional vector (hidden vector). The decoder will convert the hidden vector into the output sequence.

- Encoder-Decoder models are jointly trained to maximize the conditional probabilities of the target sequence given the input sequence.

Encoder-Decoder Architecture in Sequence- to – Sequence tasks : In order to fully understand the model's underlying logic, we will go over the below illustration:



1. **Encoder :** Multiple RNN cells can be stacked together to form the encoder. RNN reads each inputs sequentially
   - For every timestep (each input) t, the hidden state (hidden vector) h is updated according to the input at that timestep X[i].
   - After all the inputs are read by encoder model, the final hidden state of the model represents the context/summary of the whole input sequence.
   - Example: Consider the input sequence "I am a Student" to be encoded. There will be totally 4 timesteps (4 tokens) for the Encoder model. At each time step, the hidden state h will be updated using the previous hidden state and the current input.

2. **Encoder Vector:** This is the final hidden state produced from the encoder part of the model. It is calculated using the formula above.This vector aims to encapsulate the information for all input elements in order to help the decoder make accurate predictions.it acts as the initial hidden state of the decoder part of the model.

3. Decoder : The Decoder generates the output sequence by predicting the next output Yt given the hidden state ht.

- The input for the decoder is the final hidden vector obtained at the end of encoder model.
- Each layer will have three inputs, hidden vector from previous layer ht-1 and the previous layer output yt-1, original hidden vector h.
- At the first layer, the output vector of encoder and the random symbol START, empty hidden state ht-1 will be given as input, the outputs obtained will be y1 and updated hidden state h1 (the information of the output will be subtracted from the hidden vector).
- The second layer will have the updated hidden state h1 and the previous output y1 and original hidden vector h as current inputs, produces the hidden vector h2 and output y2.

- The outputs occurred at each timestep of decoder is the actual output. The model will predict the output until the END symbol occurs.
- A stack of several recurrent units where each predicts an output y_t at a time step t.

(OR)

8 A) Explain the concept of Deep Recurrent Networks (DRN) and how it enables the modeling of complex sequential dependencies [CO3-L4] [7M]

Ans: **Deep Recurrent Networks:** Defining networks consisting of a sequence input, a single hidden RNN layer, and an output layer. Despite having just one hidden layer between the input at any time step and the corresponding output, there is a sense in which these networks are deep. Inputs from the first time step can influence the outputs at the final time step T (often 100s or 1000s of steps later). These inputs pass through T applications of the recurrent layer before reaching the final output. However, we often also wish to retain the ability to express complex relationships between the inputs at a given time step and the outputs at that same time step. Thus we often construct RNNs that are deep not only in the time direction but also in the input-to-output direction. This is precisely the notion of depth that we have already encountered in our development of MLPs and deep CNNs.

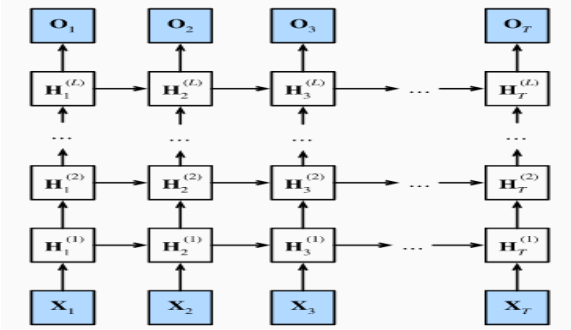


Fig. Architecture of a deep RNN.

Formally, suppose that we have a minibatch input $\mathbf{X}_t \in \mathbb{R}^{n \times d}$ (number of examples: n, number of inputs in each example: d) at time step t. At the same time step, let the hidden state of the l^{th} hidden layer ($l = 1, \ldots, L$) be $\mathbf{H}_t^{(l)} \in \mathbb{R}^{n \times h}$ (number of hidden units: h) and the output layer variable be $\mathbf{O}_t \in \mathbb{R}^{n \times q}$ (number of outputs: q). Setting $\mathbf{H}_t^{(0)} = \mathbf{X}_t$, the hidden state of the l^{th} hidden layer that uses the activation function ϕ_l is calculated as follows:

$$\mathbf{H}_t^{(l)} = \phi_l(\mathbf{H}_t^{(l-1)}\mathbf{W}_{xh}^{(l)} + \mathbf{H}_{t-1}^{(l)}\mathbf{W}_{hh}^{(l)} + \mathbf{b}_h^{(l)}),$$

where the weights $\mathbf{W}_{xh}^{(l)} \in \mathbb{R}^{h \times h}$ and $\mathbf{W}_{hh}^{(l)} \in \mathbb{R}^{h \times h}$, together with the bias $\mathbf{b}_h^{(l)} \in \mathbb{R}^{1 \times h}$, are the model parameters of the l^{th} hidden layer.

In the end, the calculation of the output layer is only based on the hidden state of the final L^{th} hidden layer:

$$\mathbf{O}_t = \mathbf{H}_t^{(L)}\mathbf{W}_{hq} + \mathbf{b}_q,$$

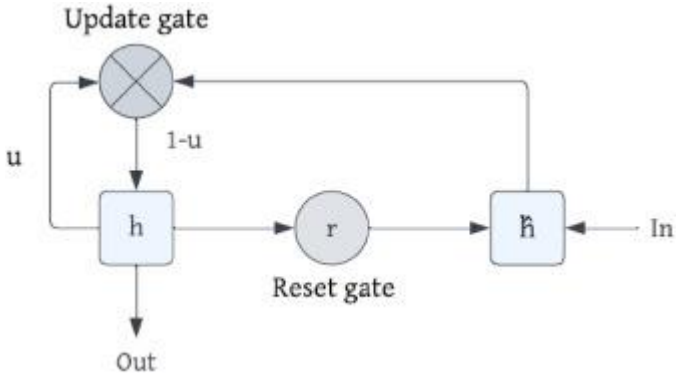
where the weight $\mathbf{W}_{hq} \in \mathbb{R}^{h \times q}$ and the bias $\mathbf{b}_q \in \mathbb{R}^{1 \times q}$ are the model parameters of the output layer.

8 B) Illustrate the core idea behind Gated Recurrent Unit (GRU) and how they extend the capabilities of standard RNNs. [CO3-L3] [7M]

Ans: The basic idea behind GRU is to use gating mechanisms to selectively update the hidden state of the network at each time step. The gating mechanisms are used to control the flow of information in and out of the network. The GRU has two gating mechanisms, called the reset gate and the update gate.

The reset gate determines how much of the previous hidden state should be forgotten, while the update gate determines how much of the new input should be used to update the hidden state. The output of the GRU is calculated based on the updated hidden state.

Architecture:



The equations used to calculate the reset gate, update gate, and hidden state of a GRU are as follows:

Reset gate: r_t = **sigmoid(W_r * [h_{t-1}, x_t])**
Update gate: z_t = **sigmoid(W_z * [h_{t-1}, x_t])**
Candidate hidden state: **h_t' = tanh(W_h * [r_t * h_{t-1}, x_t])**
Hidden state: **h_t = (1 – z_t) * h_{t-1} + z_t * h_t'**
where W_r, W_z, and W_h are learnable weight matrices, x_t is the input at time step t, h_{t-1} is the previous hidden state, and h_t is the current hidden state.

GRU networks are a type of RNN that use gating mechanisms to selectively update the hidden state at each time step, allowing them to effectively model sequential data. They have been shown to be effective in various natural language processing tasks, such as language modeling, machine translation, and speech recognition.

A Gated Recurrent Unit (GRU) is a type of recurrent neural network (RNN) that enhances the speed performance of LSTM networks by simplifying the structure with only two gates: the update gate and the reset gate. It is used when speed is crucial in processing large amounts of data.

UNIT-V

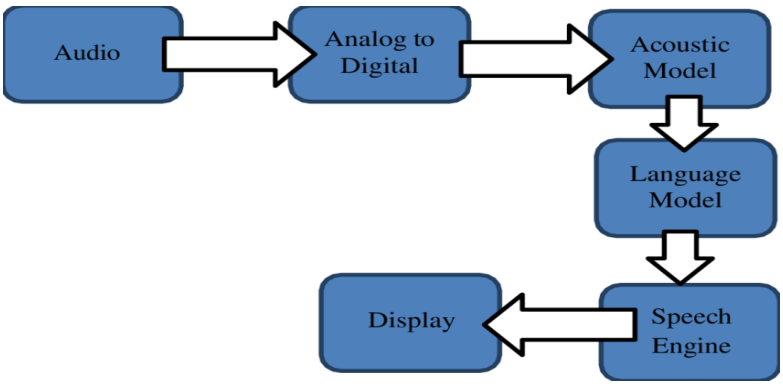
9 A) Discuss the significance of Deep Learning in the field of Speech Recognition [CO4-L2]

[7M]

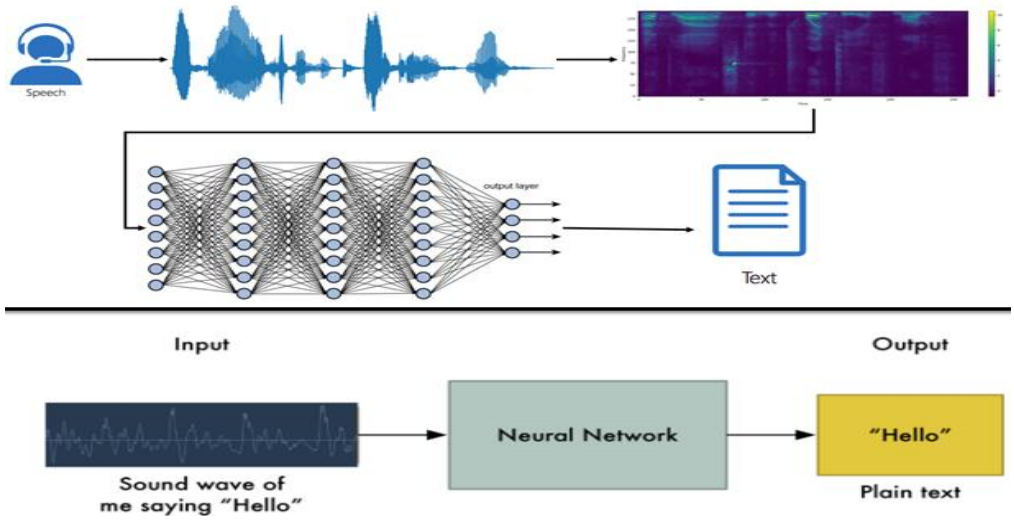
Ans: Speech recognition is the ability of a machine or program to identify and understand human speech. It has a wide range of applications, from virtual assistants like Siri and Alexa, to transcription of audio tracks, like generating subtitles for YouTube videos. Early speech recognition systems relied on hand-crafted algorithms and acoustic models. However, with the rise of deep learning, speech recognition systems have become far more accurate. Deep neural networks are able to learn speech representations directly from data, surpassing previous state-of-the-art models.

- The primary goal of speech recognition systems is to accurately and efficiently transcribe spoken words into a format that can be processed, stored, or used for various applications. This technology relies on sophisticated algorithms and Deep Learning techniques to interpret and understand human speech patterns.
- Automatic speech recognition (ASR) refers to the task of recognizing human speech and translating it into text.

Architecture:

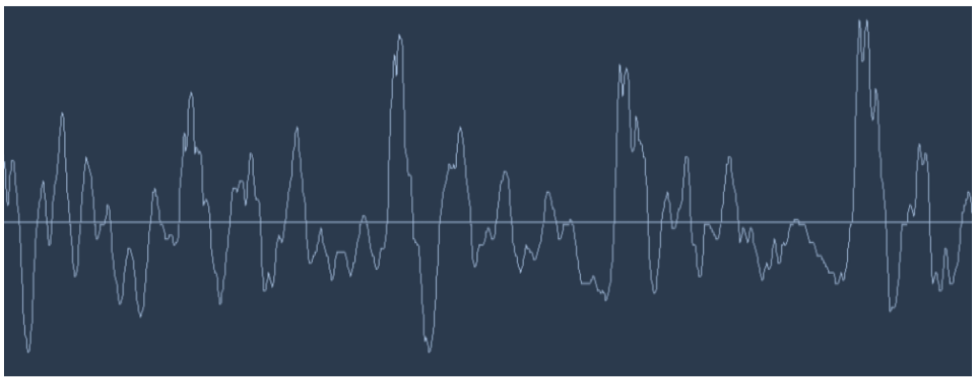


Working Procedure:



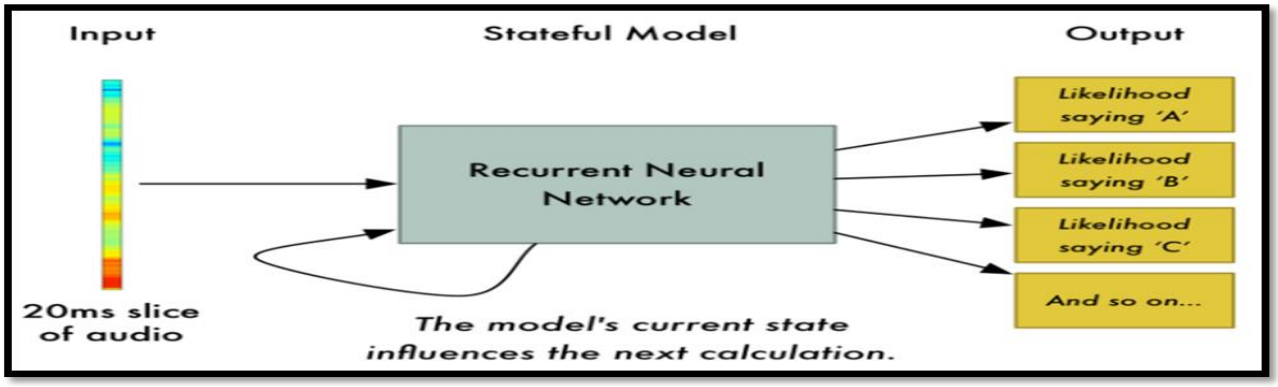
- The big problem is that **speech varies in speed.**

- One person might say **“hello!” very quickly** and another person might say **“heeeellllllllllllooooo!” very slowly**, producing a much **longer sound file with much more data.**
- Both both sound files should be recognized as exactly the **same text — “hello!”**
- Automatically aligning **audio files of various lengths** to a fixed-length piece of text turns out to be pretty hard.
- **Step-1: Turning Sounds into Bits:** The first step in speech recognition is obvious — we need to feed sound waves into a computer.
- **For Eg:** how to take an image and treat it as an array of numbers so that we can feed directly into a neural network for image recognition:
- But sound is transmitted as **waves.** How do we turn sound waves into numbers? Let’s use this sound clip of me saying **“Hello”:**



A waveform of me saying “Hello”

- Sound waves are **one-dimensional**. At every moment in time, they have a **single value based on the height of the wave**. Let’s zoom in on one tiny part of the sound wave and take a look**(These are Analog Signals)**
- **Step-2: Pre-processing the Sample sound data**
- We could feed these numbers right into a neural network. But trying to recognize speech patterns by processing these samples directly is difficult. Instead, we can make the problem easier by doing some pre-processing on the audio data.
- To make this data easier for a neural network to process, we are going to break apart this complex sound wave into it’s component parts. We’ll break out the low-pitched parts, the next-lowest-pitched-parts, and so on. Then by adding up how much energy is in each of those frequency bands (from low to high).
- **Step-3: Recognizing Characters from Short Sounds**
- Here we have to **apply deep neural network**. The **input to the neural network will be 20 millisecond audio chunks**. For each little audio slice, it will try to figure out the letter that corresponds the sound currently being spoken.

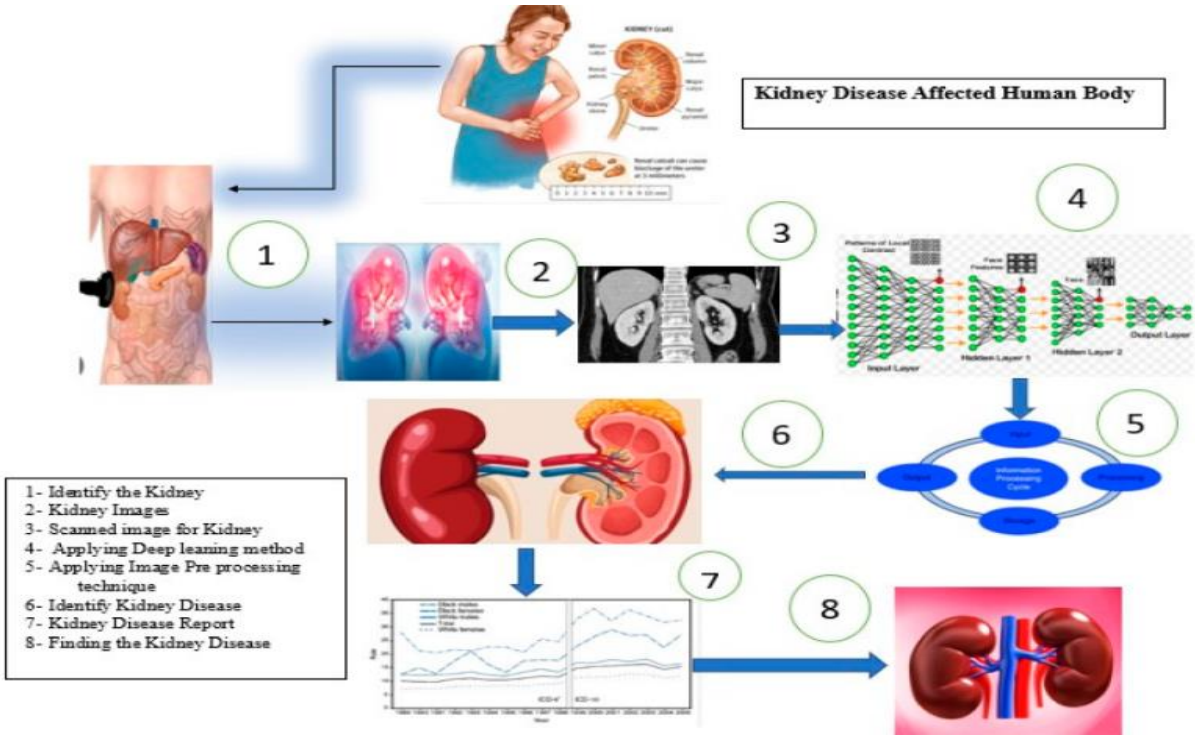


- We'll use a recurrent neural network — that is, a neural network that has a **memory that influences future predictions.**
- That's because each letter it predicts should affect the likelihood of the **next letter it will predict too.**
- **For example,** if we have said **"HEL"** so far, it's very likely we will say **"LO"** next to finish out the word **"Hello".**

---

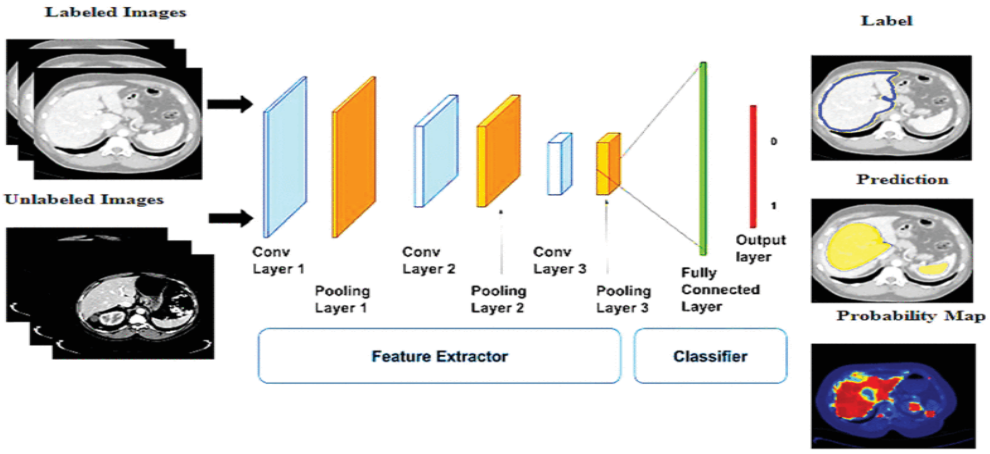
9 B) Illustrate the application of Deep Learning in the healthcare domain with an Example [CO4-L3] [7M]

Ans: Deep learning helps in detecting cancer cells and analyzing the MRI images to give elaborative results. Google has made Google AI eye doctor software. It examines retina scans and identifies diabetic retinopathy, which can cause blindness. Suppose we have taken Kidney Disease Prediction it will follow the structure.



Kidney Disease Classification is a project utilizing deep learning techniques to classify Kidney Tumor and Stone diseases from medical images dataset. This project leverages the power of Deep Learning,

Machine Learning Operations (MLOps) practices, Data Version Control (DVC). It integrates with DagsHub for collaboration and versioning.



This image , **we can take Convolution Neural Networks are used for Feature Extraction and Classification**

(OR)

10 A) Demonstrate the evolution of Deep Neural Networks in Computer Vision and their impact in image processing applications [CO4-L3] [7M]

Ans: Deep learning is a machine learning technique used to build artificial intelligence (AI) systems. It is based on the idea of artificial neural networks (ANN), designed to perform complex analysis of large amounts of data by passing it through multiple layers of neurons.

1. Image Classification: Convolutional Neural Networks (CNNs) have revolutionized image classification by their ability to automatically learn and extract features from images. CNNs process visual data through multiple layers, each layer extracting increasingly complex features from the image. This hierarchical feature extraction makes CNNs highly effective for classifying images into predefined categories.

Real-world Applications:

Facial Recognition: CNNs are widely used in facial recognition systems to identify and verify individuals based on their facial features. Applications include security systems, unlocking smartphones, and personalized user experiences.

Object Detection: In object detection tasks, CNNs not only classify images but also identify the location of objects within the images. This is essential for applications such as surveillance, autonomous driving, and robotics.

2. Image Segmentation:

Semantic Segmentation: This technique involves classifying each pixel in an image into a category, such as identifying different parts of an object or distinguishing between various objects within the same image. Semantic segmentation is crucial for understanding the structure and content of images.

Instance Segmentation: Building on semantic segmentation, instance segmentation identifies and segments each object instance separately. This allows for distinguishing between multiple objects of the same category in a single image.

- **Medical Imaging**: Image segmentation is vital in medical imaging for identifying and delineating anatomical structures, tumors, and other pathologies. It aids in accurate diagnosis and treatment planning.

- **Autonomous Driving**: Self-driving cars rely on image segmentation to understand and navigate their environment. By segmenting the road, pedestrians, vehicles, and obstacles, autonomous systems can make informed driving decisions.

3. Image Generation and Enhancement:

GANs for Creating Realistic Images :Generative Adversarial Networks (GANs) are used to generate highly realistic images by training two neural networks — the generator and the discriminator — in a competitive setting. The generator creates fake images, while the discriminator tries to distinguish between real and fake images. Over time, the generator becomes proficient at producing images that are indistinguishable from real ones.

---

10 B) Explain the impact of Deep Learning in improving Machine Translation, Sentiment Analysis [CO4-L4] [7M]

Ans: One popular type of deep learning model used in sentiment analysis is recurrent neural networks (RNNs). RNNs are designed to handle sequential data such as natural language by taking into account previous inputs when processing current inputs.

Sentiment analysis is a powerful tool in Natural Language Processing (NLP) that allows us to understand and interpret the emotions and sentiments expressed in text data. With the advancements in deep learning techniques, sentiment analysis has become even more accurate and efficient, leading to its adoption in various real-life applications.

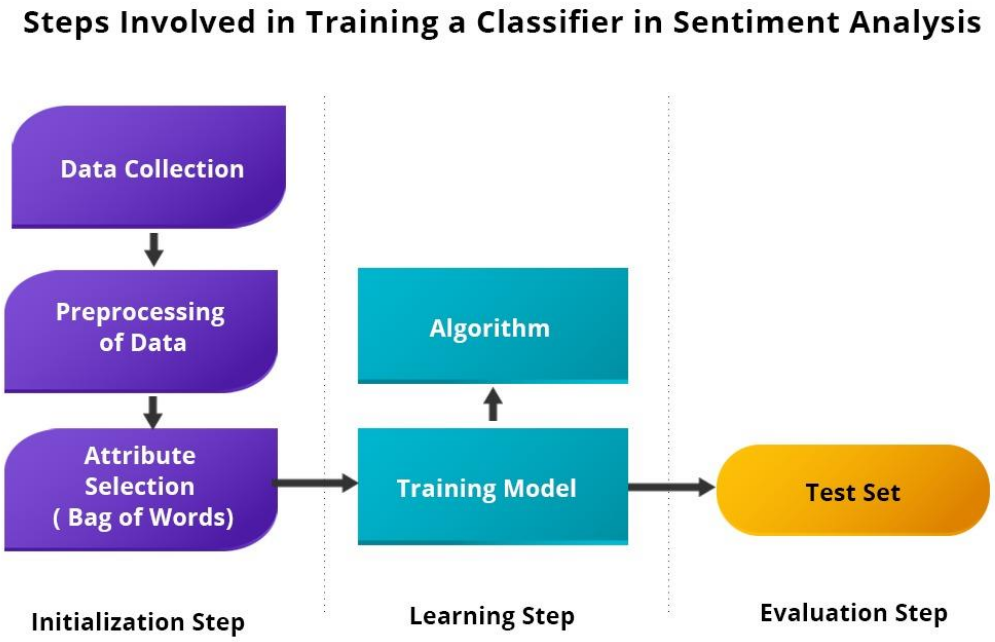
1. Customer feedback analysis: In today's competitive market, understanding customer sentiments is crucial for businesses to improve their products and services. Sentiment analysis using deep learning algorithms can help companies analyze large volumes of customer feedback from various sources such as social media reviews, surveys, and customer support interactions. This enables businesses to gain insights into customer satisfaction levels, identify areas for improvement, and make data-driven decisions.

2. Brand monitoring: With the rise of social media platforms, brands need to be aware of how their customers perceive them online. Sentiment analysis using deep learning techniques can help brands monitor their reputation by analyzing mentions on social media platforms, news articles or blog posts related to their brand. This allows companies to stay informed about any negative sentiment towards their brand and take necessary actions.

3. Stock market prediction: Sentiment analysis has found its use in predicting stock market trends by analyzing financial news articles or social media conversations related to stocks. Deep learning models

can analyze textual data from multiple sources and classify it as positive or negative sentiment towards specific stocks or the overall market trend. This information can be used by investors for making informed decisions about buying or selling stocks.

Social media is used to categorise products or services, but analysing vast comments is time-consuming. Researchers use sentiment analysis via natural language processing, evaluating methods and results conventionally through literature reviews and assessments. However, our approach diverges by offering a thorough analytical perspective with critical analysis, research findings, identified gaps, limitations, challenges and future prospects specific to deep learning-based sentiment analysis in recent times.



It all starts with building a sentiment library. Sentiment libraries are made of multiple dictionaries that have an exhaustive list of phrases and adjectives that have been manually scored beforehand. This is the same way we understand phrases. The first time we hear a phrase, we might not understand it but based on the context it is used in, we file away in our brain whether it has a positive, negative or neutral connotation.

**********THE END**********