R.M.K GROUP OF ENGINEERING INSTITUTIONS

R.M.K
GROUP OF
INSTITUTIONS

Digital Course Material
22AI005
Introduction to Generative AI

Department	: CSE(CYBER SECURITY)
Batch/Year	: 2022-2026/IV
Created by	: Ms. G.K. MONICA
	AP/CSE(CS)
Date	: 07-07-2025

TABLE OF CONTENTS

R.M.K GROUP OF INSTITUTIONS

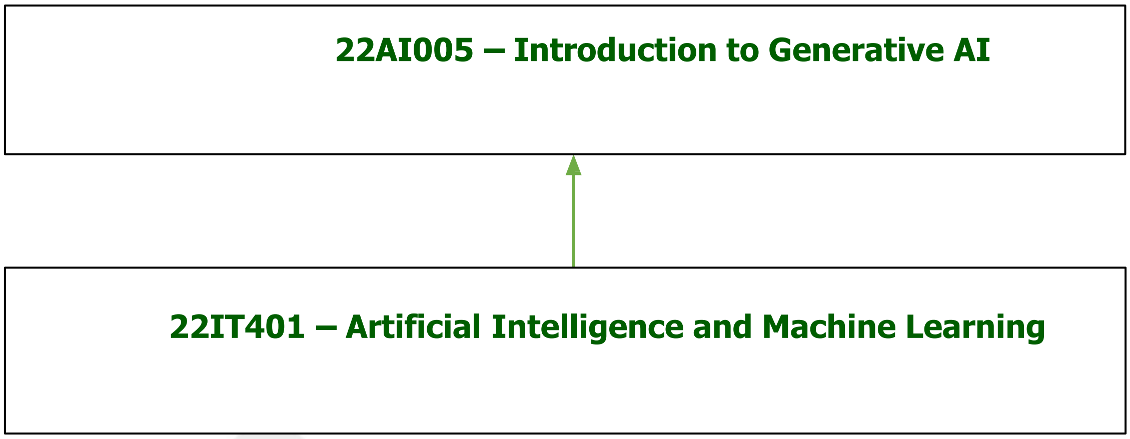
COURSE OBJECTIVES

The Course will enable learners to:

- To understand the basic concepts of Generative AI.
- To build Generative AI systems to generate images.
- To understand the concept used in Generative AI Models.
- To use various Generative AI models.
- To compare and use the various Large Language Models.
- To understand the basics of Prompt Engineering.

PRE REQUISITES

PRE-REQUISITE CHART


22AI005 – Introduction to Generative AI
22IT401 – Artificial Intelligence and Machine Learning

INTRODUCTION TO GENERATIVE AI

L T P C

3 0 0 3

OBJECTIVES:

- To understand the basic concepts of Generative AI.
- To build Generative AI systems to generate images.
- To understand the concept used in Generative AI Models.
- To use various Generative AI models.
- To compare and use the various Large Language Models.
- To understand the basics of Prompt Engineering.

UNIT I INTRODUCTION 9

Generative Models – Image transformation – Challenges - Deep Neural Networks – Perceptron – back propagation – CNN – RNN – Optimizer.

UNIT II IMAGE GENERATION 9

Creating encodings of images – variational objective – Inverse Autoregressive flow – Importing CIFAR – Creating the network from TensorFlow 2.

UNIT III GENERATIVE ADVERSARIAL NETWORKS 9

Generative Adversarial Networks – Vanilla GAN – Improved GANs – Progressive GAN – Challenges – Paired style transfer – Unpaired style transfer – Deepfakes – Modes of operation – key feature set – High level flow – Replacement – Re-enactment.

UNIT IV LARGE LANGUAGE MODELS 9

Overview of LLMs - Transformers – GPT – Types of LLMs – Key concepts – other Transformers – T5 – Generative Pre-Training Models – Multi-modal Models – DALL.E 2

UNIT V PROMPT ENGINEERING 9

Basics – In-Context Learning – In-Context Prompting – Techniques – Image Prompting – Prompt Hijacking – Challenges.

TOTAL: 45 PERIODS

COURSE OUTCOME

Course Code	Course Outcome Statement	Cognitive/ Affective Level of the Course Outcome	Expected Level of Attainment
Course Outcome Statements in Cognitive Domain			
C305.1	Elaborate the basic concepts of Generative AI	Understand K2	60%
C305.2	Build Generative AI systems to generate images	Analyse K4	60%
C305.3	Apply the concepts used in Generative AI Models	Apply K3	60%
C305.4	Use various Generative AI models.	Apply K3	60%
C305.5	Compare and use the various Large Language Models	Analyse K4	60%
C305.6	Analyze the basics of Prompt Engineering.	Apply K3	60%
Course Outcome Statements in Affective domain			
C305.7	Attend the classes regularly	Respond (A2)	95%
C305.8	Submit the Assignments regularly.	Respond (A2)	95%
C305.9	Participation in Seminar/Quiz/ Group Discussion/ Collaborative learning and content beyond syllabus	Valuing (A3)	95%

CO-PO/PSO MAPPING

Correlation Matrix of the Course Outcomes to Programme Outcomes and Programme Specific Outcomes Including Course Enrichment Activities

Course Outcomes (Cos)		Programme Outcomes (POs), Programme Specific Outcomes (PSOs)														
		PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
		K3	K4	K5	K5	K3/K5	A2	A3	A3	A3	A3	A3	A2	K3	K3	K3
C305.1	K3	3	2	1	1	3						3		3	3	3
C305.2	K4	3	3	2	2	3						3		3	3	3
C305.3	K2	2	1									2		3	3	3
C305.4	K3	3	2	1	1	3						3		3	3	3
C305.5	K4	3	3	2	2	3						3		3	3	3
C305.6	K3	3	2	1	1	3						3		2	2	2
C305.7	A2												3			
C305.8	A2								2	2	2		3			
C305.9	A3						3	3		3	3		3			
C305		3	3	2	2	3	1	1	1	3	3	3	3	3	3	3

UNIT I

INTRODUCTION

LECTURE PLAN – UNIT I

	UNIT I INTRODUCTION						
Sl. No	TOPIC	NO OF PERIODS	PROPOSED LECTURE	ACTUAL LECTURE	PERTAINING CO(s)	TAXONOMY LEVEL	MODE OF DELIVERY
			PERIOD	PERIOD			
1	Generative Models	1			CO1	K2	PPT
2	Image transformation	1			CO1	K2	PPT
3	Challenges	1			CO1	K2	PPT
4	Deep Neural Networks	1			CO1	K3	PPT
5	Perceptron	1			CO1	K3	PPT
6	Back propagation	1			CO1	K2	PPT
7	CNN	1			CO1	K2	PPT
8	RNN	1			CO1	K2	PPT
9	Optimizer.	1			CO1	K2	PPT

LECTURE PLAN – UNIT I

ASSESSMENT COMPONENTS	MODE OF DELEIVERY
AC 1. Unit Test	MD 1. Oral presentation
AC 2. Assignment	MD 2. Tutorial
AC 3. Course Seminar	MD 3. Seminar
AC 4. Course Quiz	MD 4 Hands On
AC 5. Case Study	MD 5. Videos
AC 6. Record Work	MD 6. Field Visit
AC 7. Lab / Mini Project	
AC 8. Lab Model Exam	
AC 9. Project Review	

ACTIVITY BASED LEARNING – UNIT I
QUIZ- LINK

Unit I:

Generative Models

https://custom.typingmind.com/tools/gen-ai-quiz

Deep Neural Networks

https://www.linkedin.com/advice/0/what-best-online-quizzes-testing-your-neural-network

CNN

https://quizizz.com/admin/quiz/5f2544e17c727f001b6b97c4/convolutional-neural-network

RNN

https://github.com/abhishektripathi24/Deep-Learning-Specialization-Coursera/blob/master/5.%20Sequence-Models/week1/Quiz%20-%20Recurrent%20Neural%20Networks.pdf

Perceptrons and Backpropagation

https://webcms3.cse.unsw.edu.au/COMP9444/19T3/resources/34909

TEST YOURSELF

1. What are some of the potential benefits of Generative AI?

A. Generative AI can be used to create new and innovative products and services.

B. Generative AI can be used to improve the quality of life for people with disabilities.

C. Generative AI can be used to solve complex problems that are currently beyond the reach of human intelligence.

D. All of the above

2. Which of the following is a subset of machine learning?

A. SciPy

B. NumPy

C. Deep learning

D. All of the above

3. A perceptron is a ___________

A. Backtracking algorithm

B. Backpropagation algorithm

C. Feed-forward neural network

D. Feed Forward-backward algorithm

4. What is a Convolutional Neural Network (CNN)?

A. A type of recurrent neural network

B. An unsupervised learning algorithm

C. A deep reinforcement learning model

D. A specialized neural network for image processing

5. What is the primary purpose of a Recurrent Neural Network (RNN)?

A) Image classification

B) Text generation

C) Reinforcement learning

D) Object detection

UNIT I
INTRODUCTION

1.1 Generative Models

A generative model is a type of machine learning model that aims to learn the underlying patterns or distributions of data in order to generate new, similar data. In essence, it's like teaching a computer to dream up its own data based on what it has seen before. The significance of this model lies in its ability to create, which has vast implications in various fields, from art to science.

What is a Generative Model?

Generative models use machine learning to discover patterns in data & generate new data. Learn about their significance & applications in AI.

Generative Models Explained

Generative models are a cornerstone in the world of artificial intelligence (AI). Their primary function is to understand and capture the underlying patterns or distributions from a given set of data. Once these patterns are learned, the model can then generate new data that shares similar characteristics with the original dataset.

Imagine you're teaching a child to draw animals. After showing them several pictures of different animals, the child begins to understand the general features of each animal. Given some time, the child might draw an animal they've never seen before, combining features they've learned. This is analogous to how a generative model operates: it learns from the data it's exposed to and then creates something new based on that knowledge.

The distinction between generative and discriminative models is fundamental in machine learning:

Generative models: These models focus on understanding how the data is generated. They aim to learn the distribution of the data itself. For instance, if we're looking at pictures of cats and dogs, a generative model would try to understand what makes a cat look like a cat and a dog look like a dog. It would then be able to generate new images that resemble either cats or dogs.

Discriminative models: These models, on the other hand, focus on distinguishing between different types of data. They don't necessarily learn or understand how the data is generated; instead, they learn the boundaries that separate one class of data from another. Using the same example of cats and dogs, a discriminative model would learn to tell the difference between the two, but it wouldn't necessarily be able to generate a new image of a cat or dog on its own.

In the realm of AI, generative models play a pivotal role in tasks that require the creation of new content. This could be in the form of synthesizing realistic human faces, composing music, or even generating textual content. Their ability to "dream up" new data makes them invaluable in scenarios where original content is needed, or where the augmentation of existing datasets is beneficial.

In essence, while discriminative models excel at classification tasks, generative models shine in their ability to create. This creative prowess, combined with their deep understanding of data distributions, positions generative models as a powerful tool in the AI toolkit.

Types of Generative Models

Generative models come in various forms, each with its unique approach to understanding and generating data. Here's a more comprehensive list of some of the most prominent types:

Bayesian networks. These are graphical models that represent the probabilistic relationships among a set of variables. They're particularly useful in scenarios where understanding causal relationships is crucial. For example, in medical diagnosis, a Bayesian network might help determine the likelihood of a disease given a set of symptoms.

Diffusion models. These models describe how things spread or evolve over time. They're often used in scenarios like understanding how a rumor spreads in a network or predicting the spread of a virus in a population.

Generative Adversarial Networks (GANs). GANs consist of two neural networks, the generator and the discriminator, that are trained together. The generator tries to produce data, while the discriminator attempts to distinguish between real and generated data. Over time, the generator becomes so good that the discriminator can't tell the difference. GANs are popular in image generation tasks, such as creating realistic human faces or artworks.

Variational Autoencoders (VAEs). VAEs are a type of autoencoder that produces a compressed representation of input data, then decodes it to generate new data. They're often used in tasks like image denoising or generating new images that share characteristics with the input data.

Restricted Boltzmann Machines (RBMs). RBMs are neural networks with two layers that can learn a probability distribution over its set of inputs. They've been used in recommendation systems, like suggesting movies on streaming platforms based on user preferences.

Types of Generative Models

Pixel Recurrent Neural Networks (PixelRNNs). These models generate images pixel by pixel, using the context of previous pixels to predict the next one. They're particularly useful in tasks where the sequential generation of data is crucial, like drawing an image line by line.

Markov chains. These are models that predict future states based solely on the current state, without considering the states that preceded it. They're often used in text generation, where the next word in a sentence is predicted based on the current word.

Normalizing flows. These are a series of invertible transformations applied to simple probability distributions to produce more complex distributions. They're useful in tasks where understanding the transformation of data is crucial, like in financial modeling.

Real-World Use Cases of Generative Models

Generative models have penetrated mainstream consumption, revolutionizing the way we interact with technology and experience content, for example:

- **Art creation.** Artists and musicians are using generative models to create new pieces of art or compositions, based on styles they feed into the model. For example, **Midjourney** is a very popular tool that is used to generate artwork.
- **Drug discovery.** Scientists can use generative models to predict molecular structures for new potential drugs.
- **Content creation.** Website owners leverage generative models to speed up the content creation process. For example, **Hubspot's AI content writer** helps marketers generate blog posts, landing page copy and social media posts.
- **Video games.** Game designers use generative models to create diverse and unpredictable game environments or characters.

What are the Benefits of Generative Models?

Generative models, with their unique ability to create and innovate, offer a plethora of advantages that extend beyond mere data generation. Here's a deeper dive into the myriad benefits they bring to the table:

- **Data augmentation.** In domains where data is scarce or expensive to obtain, generative models can produce additional data to supplement the original set. For instance, in medical imaging, where obtaining large datasets can be challenging, these models can generate more images to aid in better training of diagnostic tools.
- **Anomaly detection.** By gaining a deep understanding of what constitutes "normal" data, generative models can efficiently identify anomalies or outliers. This is particularly useful in sectors like finance, where spotting fraudulent transactions quickly is paramount.
- **Flexibility.** Generative models are versatile and can be employed in a range of learning scenarios, including **unsupervised**, semi-supervised, and **supervised learning**. This adaptability makes them suitable for a wide array of tasks.
- **Personalization.** These models can be tailored to generate content based on specific user preferences or inputs. For example, in the entertainment industry, generative models can create personalized music playlists or movie recommendations, enhancing user experience.
- **Innovation in design.** In fields like architecture or product design, generative models can propose novel designs or structures, pushing the boundaries of creativity and innovation.
- **Cost efficiency.** By automating the creation of content or solutions, generative models can reduce the costs associated with manual production or research, leading to more efficient processes in industries like manufacturing or entertainment.

What are the Limitations of Generative Models?

While generative models are undeniably powerful and transformative, they are not without their challenges. Here's an exploration of some of the constraints and challenges associated with these models:

- **Training complexity.** Generative models, especially sophisticated ones like GANs, require significant computational resources and time. Training them demands powerful hardware and can be resource-intensive.
- **Quality control.** While they can produce vast amounts of data, ensuring the quality and realism of the generated content can be challenging. For instance, a model might generate an image that looks realistic at first glance but has subtle anomalies upon closer inspection.
- **Overfitting.** There's a risk that generative models can become too attuned to the training data, producing outputs that lack diversity or are too closely tied to the input they've seen.
- **Lack of interpretability.** Many generative models, particularly deep learning-based ones, are often seen as "black boxes." This means it can be challenging to understand how they make decisions or why they produce specific outputs, which can be a concern in critical applications like healthcare.
- **Ethical concerns.** The ability of generative models to produce realistic content raises ethical issues, especially in the creation of deep fakes or counterfeit content. Ensuring responsible use is paramount to prevent misuse or deception.
- **Data dependency.** The quality of the generated output is heavily dependent on the quality of the training data. If the training data is biased or unrepresentative, the model's outputs will reflect those biases.
- **Mode collapse.** Particularly in GANs, there's a phenomenon called mode collapse where the generator produces limited varieties of samples, reducing the diversity of the generated outputs.

Image transformation

Image transformation in the context of generating AI refers to the ability of artificial intelligence models, particularly generative models like Generative Adversarial Networks (GANs) or Variational Autoencoders (VAEs), to alter or enhance images in various ways. Here’s a detailed look at how image transformation is approached in generating AI:

Techniques and Approaches

1. Generative Adversarial Networks (GANs):

Overview: GANs consist of two neural networks — a generator and a discriminator — that are trained adversarial to generate realistic images.

Image Transformation: GANs can transform images in several ways:

1. **Style Transfer**: Altering the artistic style of an image while preserving its content.
2. **Image-to-Image Translation**: Converting images from one domain to another (e.g., day to night, sketches to photographs).
3. **Super-Resolution**: Increasing the resolution and quality of low-resolution images.
4. **Face Aging and Rejuvenation**: Predicting how faces might look older or younger based on learned features.
5. **Inpainting**: Filling in missing parts of an image based on surrounding context.

2. Variational Autoencoders (VAEs):

Overview: VAEs consist of an encoder network that maps input images into a latent space and a decoder network that reconstructs images from this latent space.

Image Transformation: VAEs can transform images by modifying their latent representations:

1. **Image Generation**: Generating new images from learned latent representations.
2. **Image Editing**: Interpolating between latent representations to edit specific attributes (e.g., changing colors, styles).
3. **Cross-Domain Transformation**: Learning shared latent spaces to translate images between different domains (e.g., maps to satellite images).

Challenges and Considerations

1. **Quality and Realism**:
   1. Ensuring that generated images are realistic and of high quality, without artifacts or distortions.
   2. Maintaining fidelity to the input while applying transformations.
2. **Semantic Understanding**:
   1. Preserving semantic meaning and context during transformations (e.g., not distorting essential features or objects).
   2. Handling multi-modal outputs where multiple plausible transformations exist for a single input.
3. **Control and Interactivity**:
   1. Providing user control over specific aspects of transformations (e.g., style, attributes).
   2. Enabling interactive manipulation of images in real-time applications.
4. **Generalization and Robustness**:
   1. Ensuring that models generalize well to unseen data and diverse input conditions.
   2. Addressing biases and ensuring fairness in transformations, particularly in sensitive applications like face manipulation.
5. **Computational Efficiency**:
   1. Optimizing models for fast inference and real-time applications.
   2. Managing computational resources required for training and deployment, especially for large-scale datasets.

Applications

Image transformation in generating AI has diverse applications across various fields:

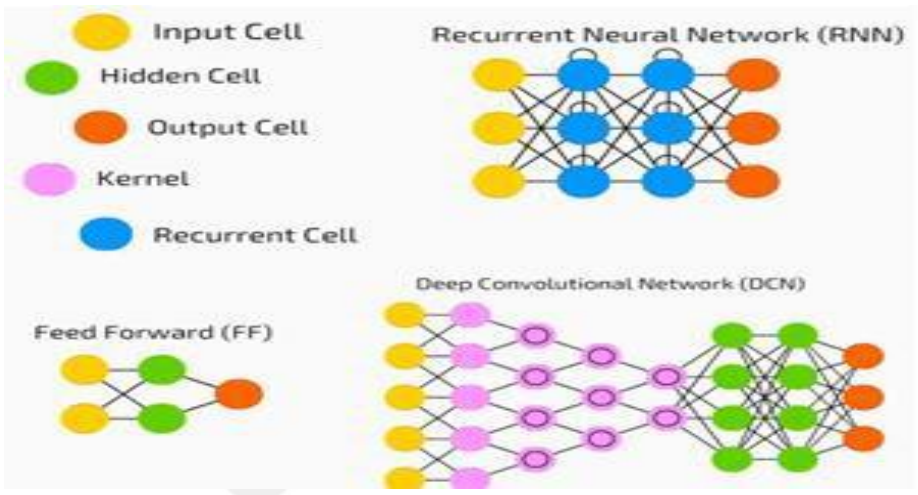
- **Creative Arts**: Enabling artists to explore new styles and create innovative artwork.
- **Entertainment**: Enhancing visual effects in movies and games.
- **Healthcare**: Generating synthetic medical images for training and diagnostics.
- **Design and Fashion**: Facilitating virtual try-ons and customization.
- **Education and Training**: Creating realistic simulations and educational materials.

Deep Neural Networks

A deep neural network (DNN) is an ANN with multiple hidden layers between the input and output layers. Similar to shallow ANNs, DNNs can model complex non-linear relationships.

The main purpose of a neural network is to receive a set of inputs, perform progressively complex calculations on them, and give output to solve real world problems like classification. We restrict ourselves to feed forward neural networks.

We have an input, an output, and a flow of sequential data in a deep network.



Neural networks are widely used in supervised learning and reinforcement learning problems. These networks are based on a set of layers connected to each other.

In deep learning, the number of hidden layers, mostly non-linear, can be large; say about 1000 layers.

DL models produce much better results than normal ML networks.

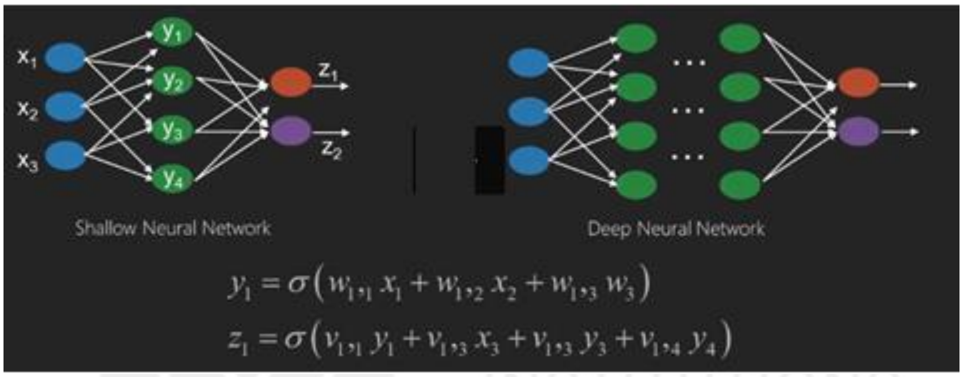
We mostly use the gradient descent method for optimizing the network and minimizing the loss function.

We can use the Imagenet, a repository of millions of digital images to classify a dataset into categories like cats and dogs. DL nets are increasingly used for dynamic images apart from static ones and for time series and text analysis.

Training the data sets forms an important part of Deep Learning models. In addition, Backpropagation is the main algorithm in training DL models.

DL deals with training large neural networks with complex input output transformations.

One example of DL is the mapping of a photo to the name of the person(s) in photo as they do on social networks and describing a picture with a phrase is another recent application of DL.

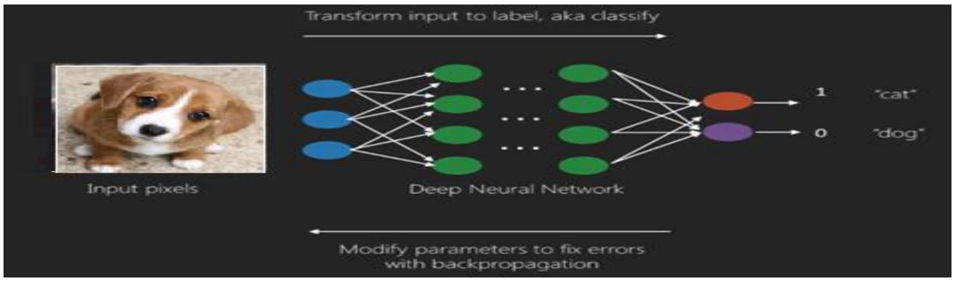


$$y_1 = \sigma\left(w_{1,1}\, x_1 + w_{1,2}\, x_2 + w_{1,3}\, w_3\right)$$
$$z_1 = \sigma\left(v_{1,1}\, y_1 + v_{1,3}\, x_3 + v_{1,3}\, y_3 + v_{1,4}\, y_4\right)$$

Neural networks are functions that have inputs like x1,x2,x3…that are transformed to outputs like z1,z2,z3 and so on in two (shallow networks) or several intermediate operations also called layers (deep networks).

The weights and biases change from layer to layer. 'w' and 'v' are the weights or synapses of layers of the neural networks.

The best use case of deep learning is the supervised learning problem.Here,we have large set of data inputs with a desired set of outputs.


Transform input to label, aka classify
Input pixels
Deep Neural Network
1
0
"cat"
"dog"
Modify parameters to fix errors with backpropagation

Here we apply back propagation algorithm to get correct output prediction.

The most basic data set of deep learning is the MNIST, a dataset of handwritten digits.

We can train deep a Convolutional Neural Network with Keras to classify images of handwritten digits from this dataset.

The firing or activation of a neural net classifier produces a score. For example,to classify patients as sick and healthy,we consider parameters such as height, weight and body temperature, blood pressure etc.

A high score means patient is sick and a low score means he is healthy.

Each node in output and hidden layers has its own classifiers. The input layer takes inputs and passes on its scores to the next hidden layer for further activation and this goes on till the output is reached.

This progress from input to output from left to right in the forward direction is called forward propagation.

Credit assignment path (CAP) in a neural network is the series of transformations starting from the input to the output. CAPs elaborate probable causal connections between the input and the output.

CAP depth for a given feed forward neural network or the CAP depth is the number of hidden layers plus one as the output layer is included. For recurrent neural networks, where a signal may propagate through a layer several times, the CAP depth can be potentially limitless.

Perceptron

A single-layer feedforward neural network was introduced in the late 1950s by Frank Rosenblatt. It was the starting phase of Deep Learning and Artificial neural networks. During that time for prediction, Statistical machine learning, or Traditional code Programming is used. Perceptron is one of the first and most straightforward models of artificial neural networks. Despite being a straightforward model, the perceptron has been proven to be successful in solving specific categorization issues.

What is Perceptron?

Perceptron is one of the simplest Artificial neural network architectures. It was introduced by Frank Rosenblatt in 1957s. It is the simplest type of feedforward neural network, consisting of a single layer of input nodes that are fully connected to a layer of output nodes. It can learn the linearly separable patterns. it uses slightly different types of artificial neurons known as threshold logic units (TLU). it was first introduced by McCulloch and Walter Pitts in the 1940s.

Types of Perceptron

Single-Layer Perceptron: This type of perceptron is limited to learning linearly separable patterns. effective for tasks where the data can be divided into distinct categories through a straight line.

Multilayer Perceptron: Multilayer perceptrons possess enhanced processing capabilities as they consist of two or more layers, adept at handling more complex patterns and relationships within the data.

Basic Components of Perceptron

A perceptron, the basic unit of a neural network, comprises essential components that collaborate in information processing.

Input Features: The perceptron takes multiple input features, each input feature represents a characteristic or attribute of the input data.

Weights: Each input feature is associated with a weight, determining the significance of each input feature in influencing the perceptron's output. During training, these weights are adjusted to learn the optimal values.

Summation Function: The perceptron calculates the weighted sum of its inputs using the summation function. The summation function combines the inputs with their respective weights to produce a weighted sum.

Activation Function: The weighted sum is then passed through an activation function. Perceptron uses Heaviside step function functions. which take the summed values as input and compare with the threshold and provide the output as 0 or 1.

Output: The final output of the perceptron, is determined by the activation function's result. For example, in binary classification problems, the output might represent a predicted class (0 or 1).

Bias: A bias term is often included in the perceptron model. The bias allows the model to make adjustments that are independent of the input. It is an additional parameter that is learned during training.

Learning Algorithm (Weight Update Rule): During training, the perceptron learns by adjusting its weights and bias based on a learning algorithm. A common approach is the perceptron learning algorithm, which updates weights based on the difference between the predicted output and the true output.

These components work together to enable a perceptron to learn and make predictions. While a single perceptron can perform binary classification, more complex tasks require the use of multiple perceptrons organized into layers, forming a neural network.

How does Perceptron work?

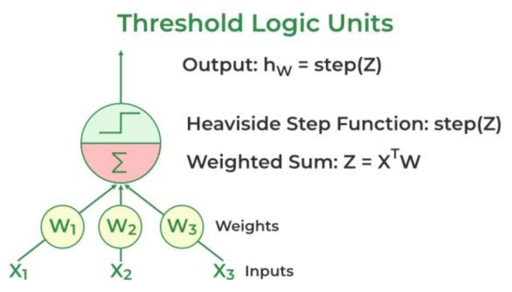
A weight is assigned to each input node of a perceptron, indicating the significance of that input to the output. The perceptron's output is a weighted sum of the inputs that have been run through an activation function to decide whether or not the perceptron will fire. it computes the weighted sum of its inputs as:

$$z = w_1x_1 + w_1x_2 + \ldots + w_nx_n = X^TW$$

The step function compares this weighted sum to the threshold, which outputs 1 if the input is larger than a threshold value and 0 otherwise, is the activation function that perceptrons utilize the most frequently. The most common step function used in perceptron is the Heaviside step function:

$$h(z) = \begin{cases} 0 & \text{if } z < Threshold \\ 1 & \text{if } z \geq Threshold \end{cases}$$

A perceptron has a single layer of threshold logic units with each TLU connected to all inputs.



When all the neurons in a layer are connected to every neuron of the previous layer, it is known as a fully connected layer or dense layer.

The output of the fully connected layer can be:

f_{W,b}(X) = h(XW + b)

where X is the input W is the weight for each inputs neurons and b is the bias and h is the step function.

During training, The perceptron's weights are adjusted to minimize the difference between the predicted output and the actual output. Usually, supervised learning algorithms like the delta rule or the perceptron learning rule are used for this.

w_{i,j} = w_{i,j} +\eta (y_j -\hat y_j)x_i

Here wi,j is the weight between the ith input and jth output neuron, xi is the ith input value, and yj and \hat y_j is the jth actual and predicted value is \eta the learning rate.

Backpropagation

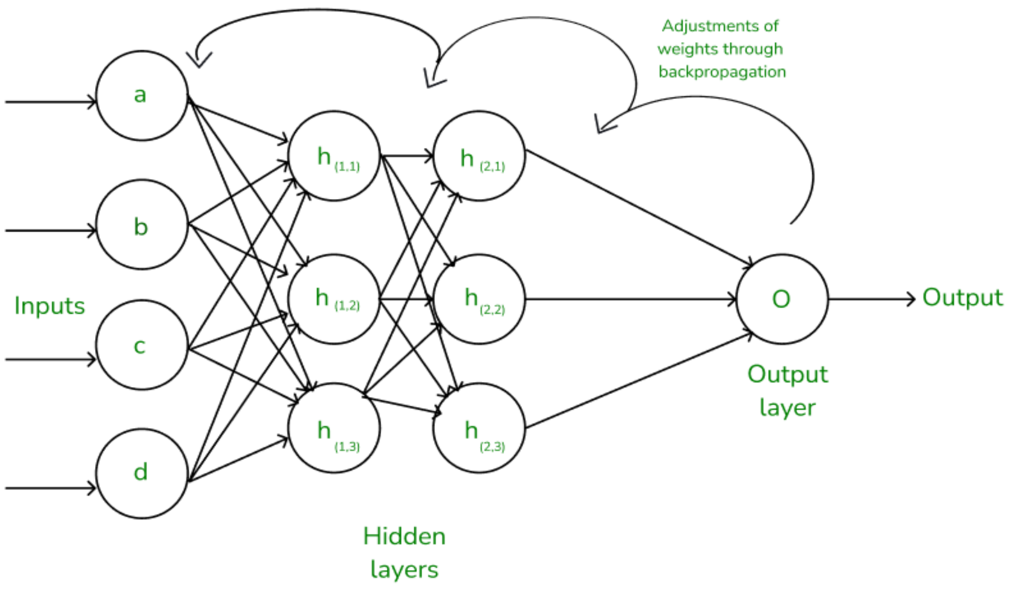
Machine learning models learn from data and make predictions. One of the fundamental concepts behind training these models is backpropagation. In this article, we will explore what backpropagation is, why it is crucial in machine learning, and how it works.

What is backpropagation?

In machine learning, backpropagation is an effective algorithm used to train artificial neural networks, especially in feed-forward neural networks.

Backpropagation is an iterative algorithm, that helps to minimize the cost function by determining which weights and biases should be adjusted. During every epoch, the model learns by adapting the weights and biases to minimize the loss by moving down toward the gradient of the error. Thus, it involves the two most popular optimization algorithms, such as gradient descent or stochastic gradient descent.

Computing the gradient in the backpropagation algorithm helps to minimize the cost function and it can be implemented by using the mathematical rule called chain rule from calculus to navigate through complex layers of the neural network.



Advantages of Using the Backpropagation Algorithm in Neural Networks

Backpropagation, a fundamental algorithm in training neural networks, offers several advantages that make it a preferred choice for many machine learning tasks. Here, we discuss some key advantages of using the backpropagation algorithm:

Ease of Implementation: Backpropagation does not require prior knowledge of neural networks, making it accessible to beginners. Its straightforward nature simplifies the programming process, as it primarily involves adjusting weights based on error derivatives.

Simplicity and Flexibility: The algorithm's simplicity allows it to be applied to a wide range of problems and network architectures. Its flexibility makes it suitable for various scenarios, from simple feedforward networks to complex recurrent or convolutional neural networks.

Efficiency: Backpropagation accelerates the learning process by directly updating weights based on the calculated error derivatives. This efficiency is particularly advantageous in training deep neural networks, where learning features of a function can be time-consuming.

Generalization: Backpropagation enables neural networks to generalize well to unseen data by iteratively adjusting weights during training. This generalization ability is crucial for developing models that can make accurate predictions on new, unseen examples.

Scalability: Backpropagation scales well with the size of the dataset and the complexity of the network. This scalability makes it suitable for large-scale machine learning tasks, where training data and network size are significant factors.

In conclusion, the backpropagation algorithm offers several advantages that contribute to its widespread use in training neural networks. Its ease of implementation, simplicity, efficiency, generalization ability, and scalability make it a valuable tool for developing and training neural network models for various machine learning applications.

Working of Backpropagation Algorithm

The Backpropagation algorithm works by two different passes, they are:

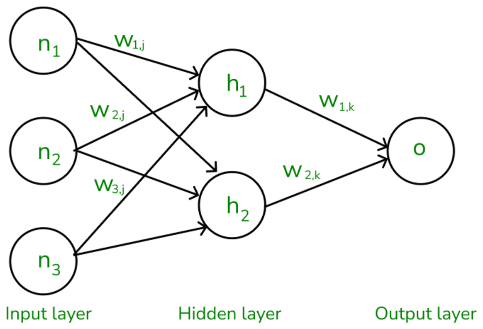
- **Forward pass**
- **Backward pass**

How does Forward pass work?

In forward pass, initially the input is fed into the input layer. Since the inputs are raw data, they can be used for training our neural network.

The inputs and their corresponding weights are passed to the hidden layer. The hidden layer performs the computation on the data it receives. If there are two hidden layers in the neural network, for instance, consider the illustration fig(a), h1 and h2 are the two hidden layers, and the output of h1 can be used as an input of h2. Before applying it to the activation function, the bias is added.

To the weighted sum of inputs, the activation function is applied in the hidden layer to each of its neurons. One such activation function that is commonly used is ReLU can also be used, which is responsible for returning the input if it is positive otherwise it returns zero. By doing this so, it introduces the non-linearity to our model, which enables the network to learn the complex relationships in the data. And finally, the weighted outputs from the last hidden layer are fed into the output to compute the final prediction, this layer can also use the activation function called the softmax function which is responsible for converting the weighted outputs into probabilities for each class.



How does backward pass work?

In the backward pass process shows, the error is transmitted back to the network which helps the network, to improve its performance by learning and adjusting the internal weights.

To find the error generated through the process of forward pass, we can use one of the most commonly used methods called mean squared error which calculates the difference between the predicted output and desired output. The formula for mean squared error is:

$Meansquarederror=(predictedoutput–actualoutput)2$

Once we have done the calculation at the output layer, we then propagate the error backward through the network, layer by layer.

The key calculation during the backward pass is determining the gradients for each weight and bias in the network. This gradient is responsible for telling us how much each weight/bias should be adjusted to minimize the error in the next forward pass. The chain rule is used iteratively to calculate this gradient efficiently.

In addition to gradient calculation, the activation function also plays a crucial role in backpropagation, it works by calculating the gradients with the help of the derivative of the activation function.

Convolution Neural Network

A Convolutional Neural Network (CNN) is a type of Deep Learning neural network architecture commonly used in Computer Vision. Computer vision is a field of Artificial Intelligence that enables a computer to understand and interpret the image or visual data.

When it comes to Machine Learning, Artificial Neural Networks perform really well. Neural Networks are used in various datasets like images, audio, and text. Different types of Neural Networks are used for different purposes, for example for predicting the sequence of words we use Recurrent Neural Networks more precisely an LSTM, similarly for image classification we use Convolution Neural networks. In this blog, we are going to build a basic building block for CNN.

In a regular Neural Network there are three types of layers:

1.**Input Layer**s: It's the layer in which we give input to our model. The number of neurons in this layer is equal to the total number of features in our data (number of pixels in the case of an image).

2.**Hidden Layer**: The input from the Input layer is then fed into the hidden layer. There can be many hidden layers depending on our model and data size. Each hidden layer can have different numbers of neurons which are generally greater than the number of features. The output from each layer is computed by matrix multiplication of the output of the previous layer with learnable weights of that layer and then by the addition of learnable biases followed by activation function which makes the network nonlinear.

3**.Output Layer**: The output from the hidden layer is then fed into a logistic function like sigmoid or softmax which converts the output of each class into the probability score of each class.

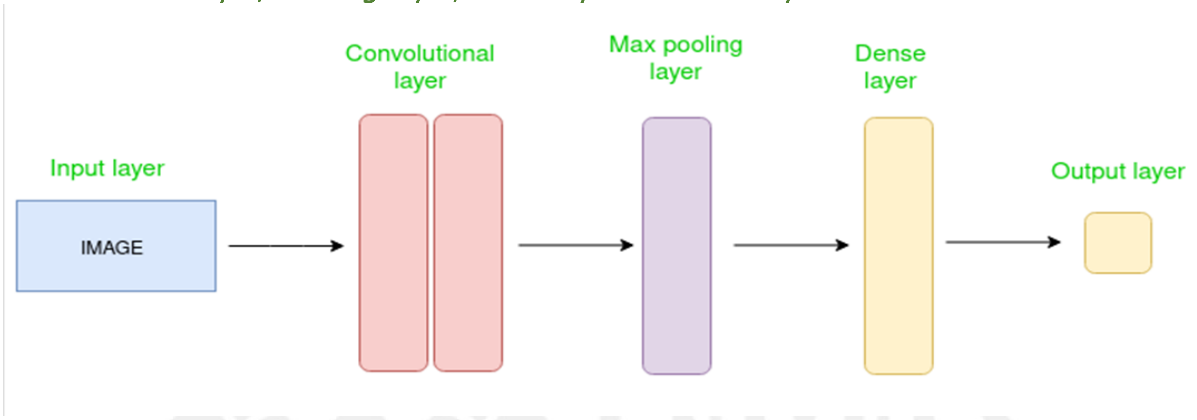
The data is fed into the model and output from each layer is obtained from the above step is called feedforward, we then calculate the error using an error function, some common error functions are cross-entropy, square loss error, etc. The error function measures how well the network is performing. After that, we backpropagate into the model by calculating the derivatives. This step is called Backpropagation which basically is used to minimize the loss.

Convolution Neural Network

Convolutional Neural Network (CNN) is the extended version of artificial neural networks (ANN) which is predominantly used to extract the feature from the grid-like matrix dataset. For example visual datasets like images or videos where data patterns play an extensive role.

CNN architecture

Convolutional Neural Network consists of multiple layers like the input layer, Convolutional layer, Pooling layer, and fully connected layers.

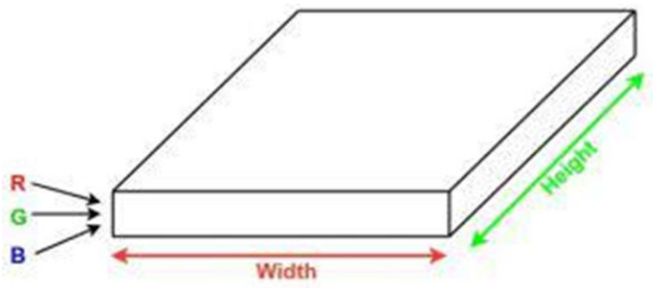


The Convolutional layer applies filters to the input image to extract features, the Pooling layer downsamples the image to reduce computation, and the fully

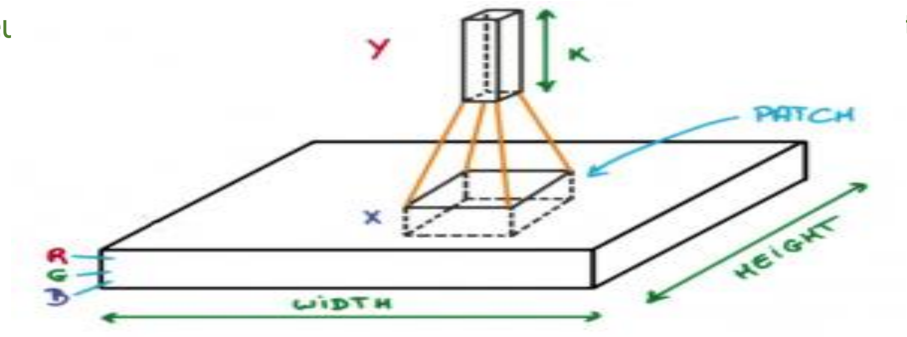
connected layer makes the final prediction. The network learns the optimal filters through backpropagation and gradient descent.

How Convolutional Layers works

Convolution Neural Networks or covnets are neural networks that share their parameters. Imagine you have an image. It can be represented as a cuboid having its length, width (dimension of the image), and height (i.e the channel as images generally have red, green, and blue channels).



Now imagine taking a small patch of this image and running a small neural network, called a filter or kernel on it, with say, K outputs and representing them vertically. Now slide that neural network across the whole image, as a result, we will get another image with different widths, heights, and depths. Instead of just R, G, and B channels now we have more channels but lesser width and height. This operation is called Convolution. If the patch size is the same as that of the image it will be a regular neu ts.



Now let's talk about a bit of mathematics that is involved in the whole convolution process.

•Convolution layers consist of a set of learnable filters (or kernels) having small widths and heights and the same depth as that of input volume (3 if the input layer is image input).

•For example, if we have to run convolution on an image with dimensions 34x34x3. The possible size of filters can be axax3, where 'a' can be anything like 3, 5, or 7 but smaller as compared to the image dimension.

•During the forward pass, we slide each filter across the whole input volume step by step where each step is called stride (which can have a value of 2, 3, or even 4 for high-dimensional images) and compute the dot product between the kernel weights and patch from input volume.

•As we slide our filters we'll get a 2-D output for each filter and we'll stack them together as a result, we'll get output volume having a depth equal to the number of filters. The network will learn all the filters.

Layers used to build ConvNets

A complete Convolution Neural Networks architecture is also known as covnets. A covnets is a sequence of layers, and every layer transforms one volume to another through a differentiable function.

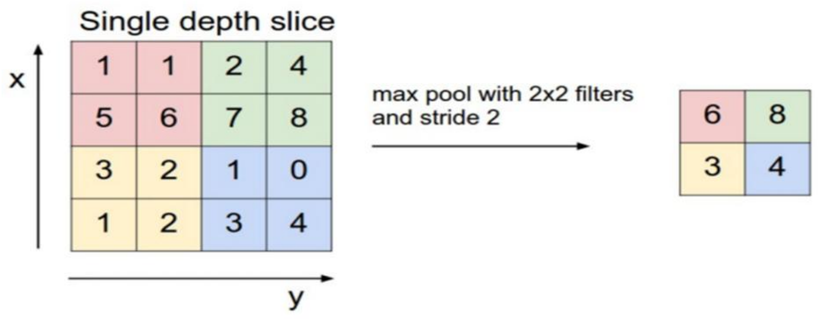
Types of layers: datasets

Let's take an example by running a covnets on of image of dimension 32 x 32 x 3.

•**Input Layers:** It's the layer in which we give input to our model. In CNN, Generally, the input will be an image or a sequence of images. This layer holds the raw input of the image with width 32, height 32, and depth 3.

•**Convolutional Layers**: This is the layer, which is used to extract the feature from the input dataset. It applies a set of learnable filters known as the kernels to the input images. The filters/kernels are smaller matrices usually 2×2, 3×3, or 5×5 shape. it slides over the input image data and computes the dot product between kernel weight and the corresponding input image patch. The output of this layer is referred as feature maps. Suppose we use a total of 12 filters for this layer we'll get an output volume of dimension 32 x 32 x 12.

•**Activation Layer:** By adding an activation function to the output of the preceding layer, activation layers add nonlinearity to the network. it will apply an element-wise activation function to the output of the convolution layer. Some common activation functions are RELU: max(0, x), Tanh, Leaky RELU, etc. The volume remains unchanged hence output volume will have dimensions 32 x 32 x 12.

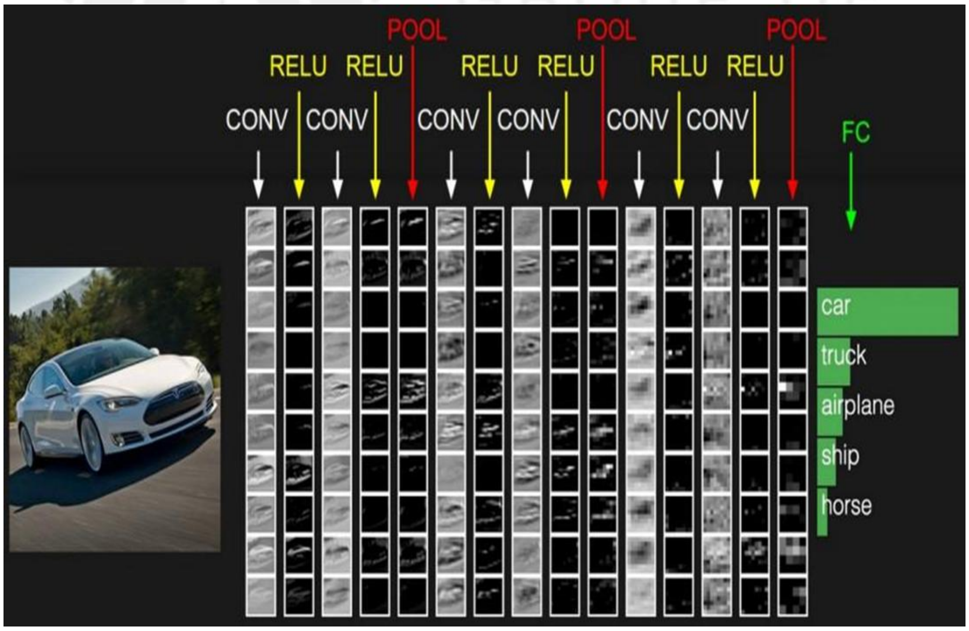


• put of each class into the probability score of each class. **Pooling layer**: This layer is periodically inserted in the covnets and its main function is to reduce the size of volume which makes the computation fast reduces memory and also prevents overfitting. Two common types of pooling layers are max pooling and average pooling. If we use a max pool with 2 x 2 filters and stride 2, the resultant volume will be of dimension 16x16x12.

•**Flattening:** The resulting feature maps are flattened into a one-dimensional vector after the convolution and pooling layers so they can be passed into a completely linked layer for categorization or regression.

•**Fully Connected Layers:** It takes the input from the previous layer and computes the final classification or regression task.

•**Output Layer:** The output from the fully connected layers is then fed into a logistic function for classification tasks like sigmoid or softmax which converts the out

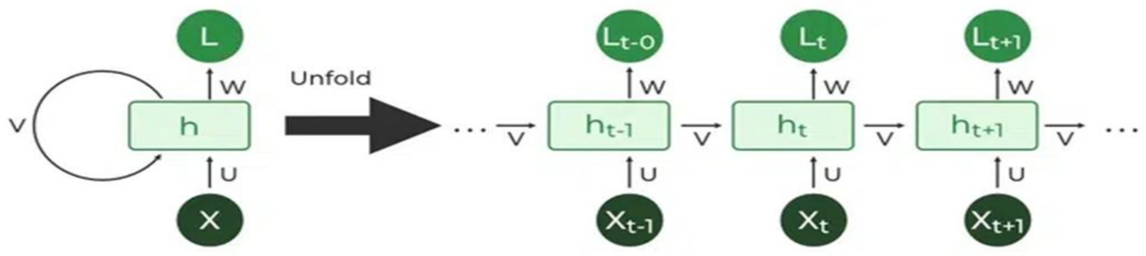


Recurrent Neural Network

we will introduce a new variation of neural network which is the Recurrent Neural Network also known as (RNN) that works better than a simple neural network when data is sequential like Time-Series data and text data.

What is Recurrent Neural Network (RNN)?

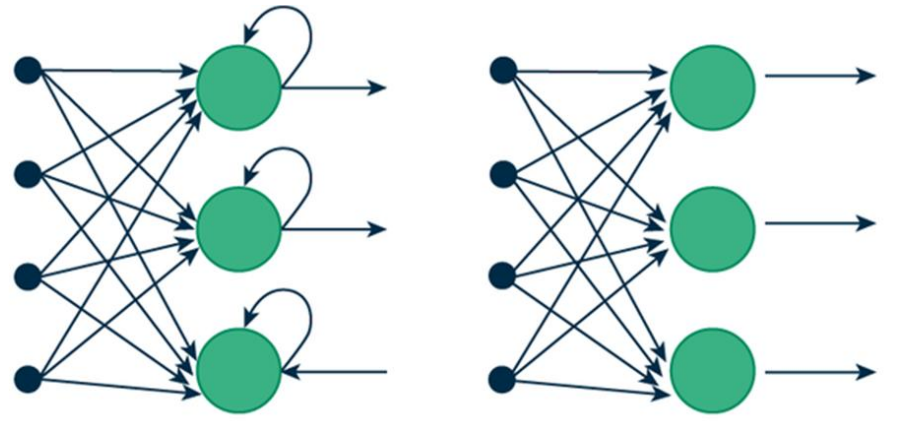
Recurrent Neural Network(RNN) is a type of Neural Network where the output from the previous step is fed as input to the current step. In traditional neural networks, all the inputs and outputs are independent of each other. Still, in cases when it is required to predict the next word of a sentence, the previous words are required and hence there is a need to remember the previous words. Thus RNN came into existence, which solved this issue with the help of a Hidden Layer. The main and most important feature of RNN is its Hidden state, which remembers some information about a sequence. The state is also referred to as Memory State since it remembers the previous input to the network. It uses the same parameters for each input as it performs the same task on all the inputs or hidden layers to produce the output. This reduces the complexity of parameters, unlike other neural networks.



How RNN differs from Feedforward Neural Network?

Artificial neural networks that do not have looping nodes are called feed forward neural networks. Because all information is only passed forward, this kind of neural network is also referred to as a multi-layer neural network.

Information moves from the input layer to the output layer – if any hidden layers are present – unidirectionally in a feedforward neural network. These networks are appropriate for image classification tasks, for example, where input and output are independent. Nevertheless, their inability to retain previous inputs automatically renders them less useful for sequential data analysis.

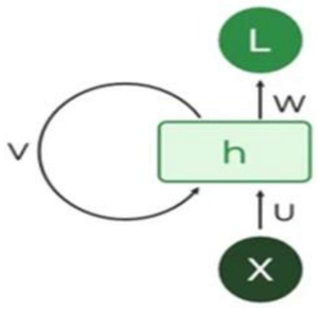


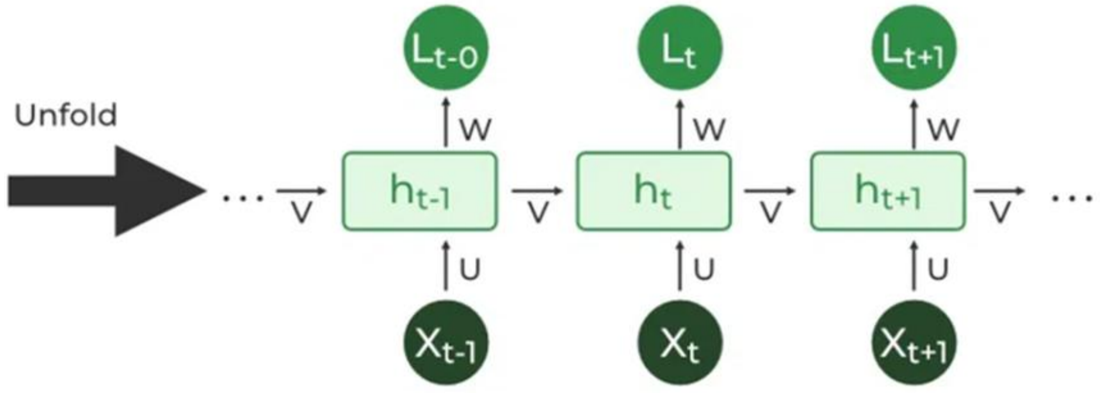
(a) Recurrent Neural Network (b) Feed-Forward Neural Network

Recurrent Neuron and RNN Unfolding

The fundamental processing unit in a Recurrent Neural Network (RNN) is a Recurrent Unit, which is not explicitly called a "Recurrent Neuron." This unit has the unique ability to maintain a hidden state, allowing the network to capture sequential dependencies by remembering previous inputs while processing. Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) versions improve the RNN's ability to handle long-term dependencies.

Recurrent Neuron





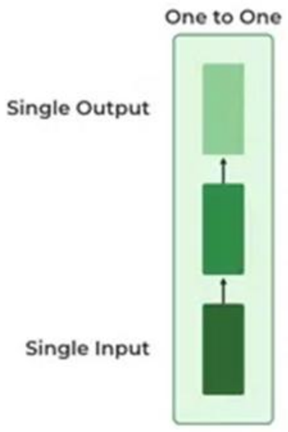
Types Of RNN

There are four types of RNNs based on the number of inputs and outputs in the network.

2. One to One
3. One to Many
4. Many to One
5. Many to Many

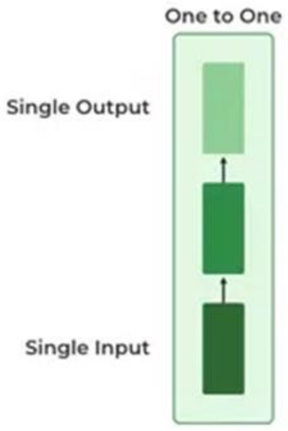
One to One

This type of RNN behaves the same as any simple Neural network it is also known as Vanilla Neural Network. In this Neural network, there is only one input and one output.



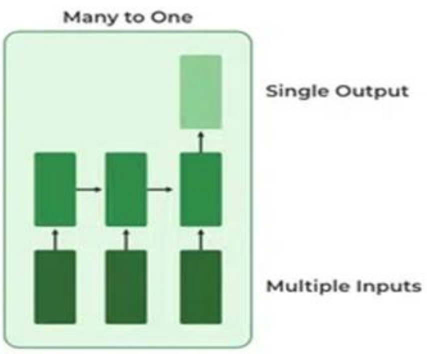
One To Many

In this type of RNN, there is one input and many outputs associated with it. One of the most used examples of this network is Image captioning where given an image we predict a sentence having Multiple words.



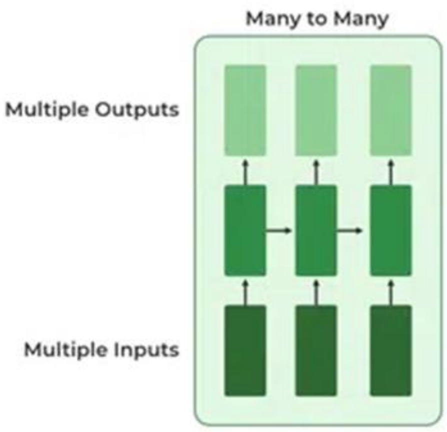
Many to One

In this type of network, Many inputs are fed to the network at several states of the network generating only one output. This type of network is used in the problems like sentimental analysis. Where we give multiple words as input and predict only the sentiment of the sentence as output.



Many to Many

In this type of neural network, there are multiple inputs and multiple outputs corresponding to a problem. One Example of this Problem will be language translation. In language translation, we provide multiple words from one language as input and predict multiple words from the second language as output.



Recurrent Neural Network Architecture

RNNs have the same input and output architecture as any other deep neural architecture. However, differences arise in the way information flows from input to output. Unlike Deep neural networks where we have different weight matrices for each Dense network in RNN, the weight across the network remains the same. It calculates state hidden state Hi for every input Xi . By using the following formulas:

$h= \sigma(UX + Wh-1 + B)$

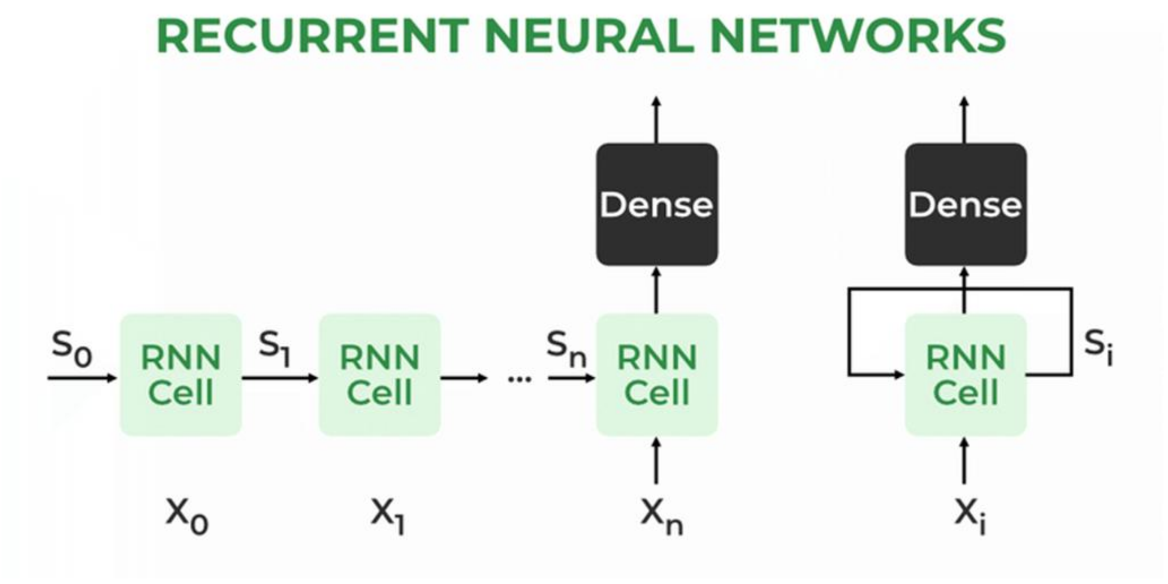
$Y = O(Vh + C)$

Hence

$Y = f (X, h , W, U, V, B, C)$

Here S is the State matrix which has element si as the state of the network at timestep i

The parameters in the network are W, U, V, c, b which are shared across timestep



How does RNN work?

The Recurrent Neural Network consists of multiple fixed activation function units, one for each time step. Each unit has an internal state which is called the hidden state of the unit. This hidden state signifies the past knowledge that the network currently holds at a given time step. This hidden state is updated at every time step to signify the change in the knowledge of the network about the past. The hidden state is updated using the following recurrence relation:-

The formula for calculating the current state:

$ht=f(ht-1,xt)$ht=f(ht−1,xt)

where,

- ht -> current state
- ht-1 -> previous state
- xt -> input state

Formula for applying Activation function(tanh)

$ht=tanh(Whhht-1+Wxhxt)$ht=tanh(Whhht−1+Wxhxt)

where,

- whh -> weight at recurrent neuron
- wxh -> weight at input neuron

The formula for calculating output:

$yt=Whyht$yt=Whyht

- Yt -> output
- Why -> weight at output layer

These parameters are updated using Backpropagation. However, since RNN works on sequential data here we use an updated backpropagation which is known as Backpropagation through time.

Optimizers

Deep learning, a subset of machine learning, tackles intricate tasks like speech recognition and text classification. Comprising components such as activation functions, input, output, hidden layers, and loss functions, deep learning models aim to generalize data and make predictions on unseen data. To optimize these models, various algorithms, known as optimizers, are employed. Optimizers adjust model parameters iteratively during training to minimize a loss function, enabling neural networks to learn from data. This guide delves into different optimizers used in deep learning, discussing their advantages, drawbacks, and factors influencing the selection of one optimizer over another for specific applications. Common optimizers include Stochastic Gradient Descent (SGD), Adam, and RMSprop, each employing specific update rules, learning rates, and momentum for refining model parameters. Optimizers play a pivotal role in enhancing accuracy and speeding up the training process, shaping the overall performance of deep learning models.

Need for Optimizers in Deep Learning

Choosing an appropriate optimizer for a deep learning model is important as it can greatly impact its performance. Optimization algorithms have different strengths and weaknesses and are better suited for certain problems and architectures.

For example, stochastic gradient descent is a simple and efficient optimizer that is widely used, but it may need help to converge on problems with complex, non-convex loss functions. On the other hand, Adam is a more sophisticated optimizer that combines the ideas of momentum and adaptive learning rates and is often considered one of the most effective optimizers in deep learning.

Here are a few pointers to keep in mind when choosing an optimizer:

•Understand the problem and model architecture, as this will help you determine which optimizer is most suitable

•Experiment with different optimizers in deep learning to see which one works best for your problem

•Adjust the hyperparameters of the optimizer, such as the learning rate, to see if it improves performance

•Remember that the optimizer's choice is not the only factor affecting model performance.

•Other important factors include the choice of architecture, the quality of the data, and the amount of data available.

Types of Optimizers

Many types of optimizers are available for training machine learning models, each with its own strengths and weaknesses. Some optimizers are better suited for certain types of models or data, while others are more general-purpose.

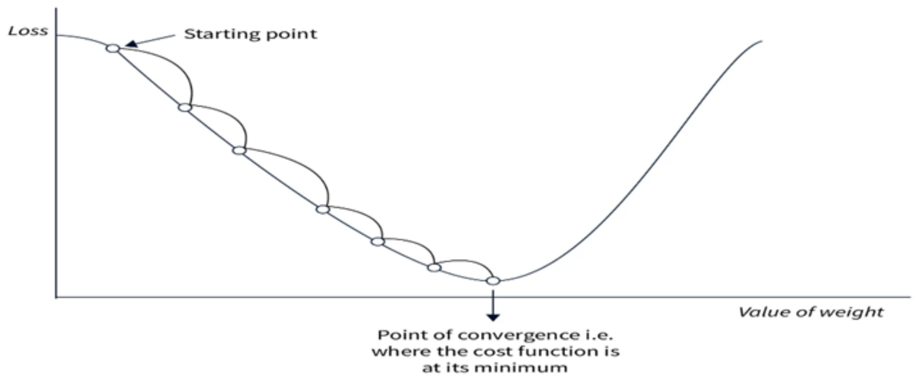
This section will briefly overview the most commonly used optimizers, starting with the simpler ones and progressing to the more complex ones.

Gradient Descent

Gradient descent is a simple optimization algorithm that updates the model's parameters to minimize the loss function. We can write the basic form of the algorithm as follows:

$\theta=\theta-\alpha\cdot\nabla\theta L(\theta)$

where θ is the model parameter, $L(\theta)$ is the loss function, and α is the learning rate.


Loss
Starting point
Value of weight
Point of convergence i.e. where the cost function is at its minimum

Pros:

- •Simple to implement.
- •Can work well with a well-tuned learning rate.

Cons:

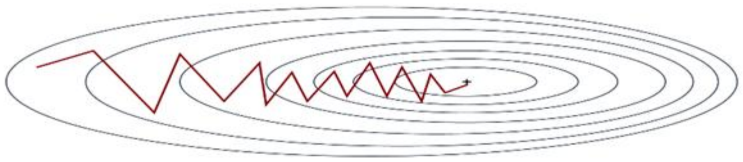
- •It can converge slowly, especially for complex models or large datasets.
- •Sensitive to the choice of learning rate.

Stochastic Gradient Descent

Stochastic gradient descent (SGD) is a variant of gradient descent that involves updating the parameters based on a small, randomly-selected subset of the data (i.e., a "mini-batch") rather than the full dataset. We can write the basic form of the algorithm as follows:

$\theta = \theta - \alpha \cdot \nabla \theta L(\theta; x(i); y(i))$

where $(x(i), y(i)$is a mini-batch of data.



Pros:

•It can be faster than standard gradient descent, especially for large datasets.

•Can escape local minima more easily.

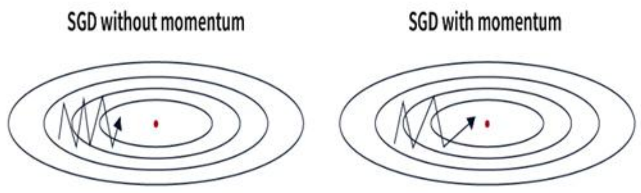
Cons:

•It can be noisy, leading to less stability.

•It may require more hyperparameter tuning to get good performance.

Stochastic Gradient Descent with Momentum

SGD with momentum is a variant of SGD that adds a "momentum" term to the update rule, whi the same direction even if the local ically set to a value between 0 and 1.



$v = \beta \cdot v + (1 - \beta) \cdot \nabla \theta$

$\theta = \theta - \alpha \cdot v$

where v is the momentum vector and β is the momentum hyperparameter

Pros:

•It can help the optimizer to move more efficiently through "flat" regions of the loss function.

•It can help to reduce oscillations and improve convergence.

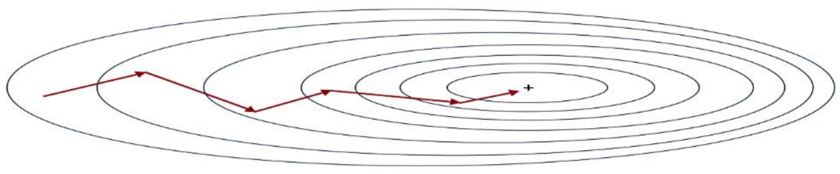
Cons:

•Can overshoot good solutions and settle for suboptimal ones if the momentum is too high.

•Requires tuning of the momentum hyperparameter.

Mini-Batch Gradient Descent

Mini-batch gradient descent is similar to SGD, but instead of using a single sample to compute the gradient, it uses a small, fixed-size "mini-batch" of samples. The update rule is the same as for SGD, except that the gradient is averaged over the mini-batch. This can reduce noise in the updates and improve convergence.



Pros:

•It can be faster than standard gradient descent, especially for large datasets.

•Can escape local minima more easily.

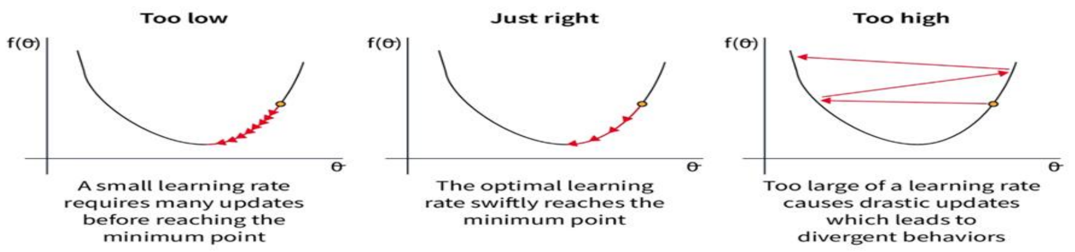
•Can reduce noise in updates, leading to more stable convergence.

Cons:

•Can be sensitive to the choice of mini-batch size.

How Do Optimizers Work in Deep Learning?

Optimizers in deep learning adjust the model's parameters to minimize the loss function. The loss function measures how well the model can make predictions on a given dataset, and the goal of training a model is to find the set of model parameters that yields the lowest possible loss.


Too low
f(θ)
θ
A small learning rate requires many updates before reaching the minimum point
Just right
f(θ)
θ
The optimal learning rate swiftly reaches the minimum point
Too high
f(θ)
θ
Too large of a learning rate causes drastic updates which leads to divergent behaviors

The optimizer uses an optimization algorithm to search for the parameters that minimize the loss function. The optimization algorithm uses the gradients of the loss function to the model parameters to determine the direction in which we should adjust the parameters.

The gradients are computed using backpropagation, which involves applying the chain rule to compute the gradients of the loss function to each of the model parameters.

The optimization algorithm then adjusts the model parameters to minimize the loss function. This process is repeated until the loss function reaches a minimum or the optimizer reaches the maximum number of allowed iterations.

Conclusion

Optimizers in deep learning are essential, as they adjust the model's parameters to minimize the loss function.

In general, the choice of which optimization algorithm to use will depend on the specific characteristics of the problem, such as the dataset's size and the model's complexity.

It is important to consider each algorithm's pros and cons carefully and tune any relevant hyperparameters to achieve the best possible performance.

Overall, understanding the role of optimizers in deep learning and the various available algorithms is essential for anyone looking to build and train effective machine learning models.

ASSIGNMENT – UNIT I

1. Design a CNN architecture for detecting objects in images.(K6)
2. Evaluate the trade-offs between convergence speed and stability for different optimizers(K5)
3. Analyze the impact of learning rate on the effectiveness of backpropagation.(K4)
4. Design a deep neural network architecture for image recognition(K6)
5. How would you apply an RNN to a time series prediction task?(K3)

PART A- UNIT-1

1.Define what a generative model is in the context of machine learning.

Generative" describes a class of statistical models that contrasts with discriminative models. Informally: Generative models can generate new data instances. Discriminative models discriminate between different kinds of data instances.

2. What are the challenges associated with training deep neural networks compared to shallow networks?

One challenge is that they can be more difficult to train than shallow networks, due to the increased number of parameters and the risk of overfitting. Another challenge is that deep networks can be computationally expensive to train and require a lot of data to achieve good performance.

3. What do you mean by DNN?

Deep neural networks (DNN) can be defined as ANNs with additional depth, that is, an increased number of hidden layers between the input and the output layers.

4. What is a perceptron?

A machine-based algorithm used for supervised learning of various binary sorting tasks is called Perceptron. Furthermore, Perceptron also has an essential role as an Artificial Neuron or Neural link in detecting certain input data computations in business intelligence

5. Define backpropagation in the context of neural networks.

Backpropagation is an algorithm used in artificial intelligence and machine learning to train artificial neural networks through error correction. The computer learns by calculating the loss function, or the difference between the input you provided and the output it produced.

6. How many convolutional layers are in CNN?

A CNN typically has three layers: a convolutional layer, a pooling layer, and a fully connected layer.

7. What are convolutional neural networks?

A convolutional neural network (CNN or ConvNet) is a network architecture for deep learning that learns directly from data. CNNs are particularly useful for finding patterns in images to recognize objects, classes, and categories. They can also be quite effective for classifying audio, time-series, and signal data.

8. How would you apply an RNN to a time series prediction task?

A typical structure for time series prediction comprises an input layer, one or more hidden layers with LSTM or GRU cells and an output layer. Selecting the optimal number of layers and neurons: This is a critical step in building the RNN model.

9. Define recurrent neural networks (RNNs)

A recurrent neural network (RNN) is a deep learning model that is trained to process and convert a sequential data input into a specific sequential data output.

10. What is an optimizer in the context of neural networks?

Optimizers are algorithms or methods used to change the attributes of the neural network such as weights and learning rate to reduce the losses. Optimizers are used to solve optimization problems by minimizing the function

PART B- UNIT 1

1. Apply a CNN to a standard image classification taskK3

2.Explain how RNNs process sequential data.K2

3.Compare the performance of RNNs on different types of sequential data.K4

4.Explain the difference between gradient descent and stochastic gradient descent.K2

5.Analyze the convergence behavior of different optimizers.K4

6.Evaluate the trade-offs between speed and accuracy for various optimizers.K5

7.Design a multi-layer perceptron to solve a complex classification task.K6

8.Analyze the impact of network depth on the training processK4

9.Apply image transformation techniques to preprocess a dataset.K3

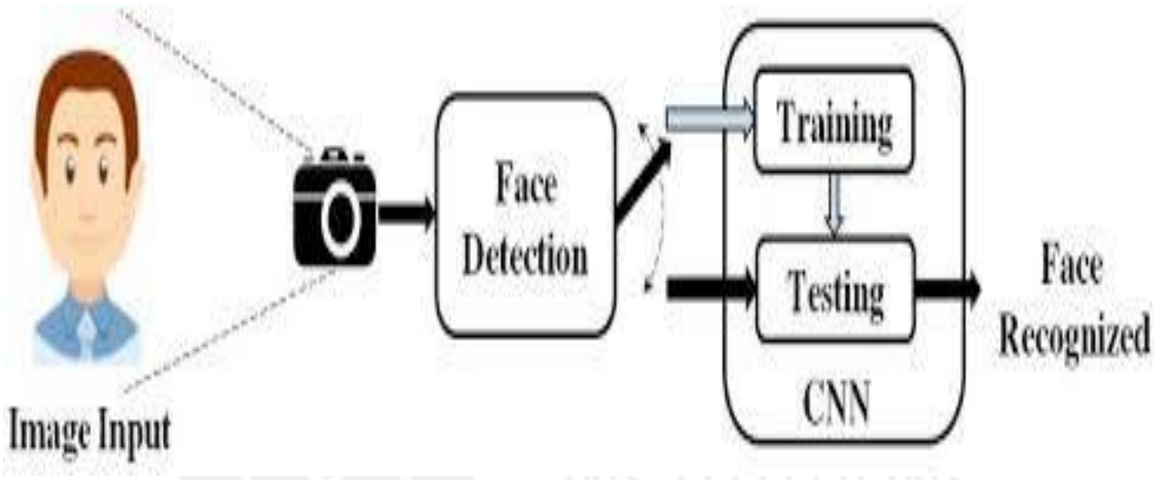
10.Demonstrate how to evaluate the performance of a generative model.K3

SUPPORTIVE ONLINE COURSES – UNIT I

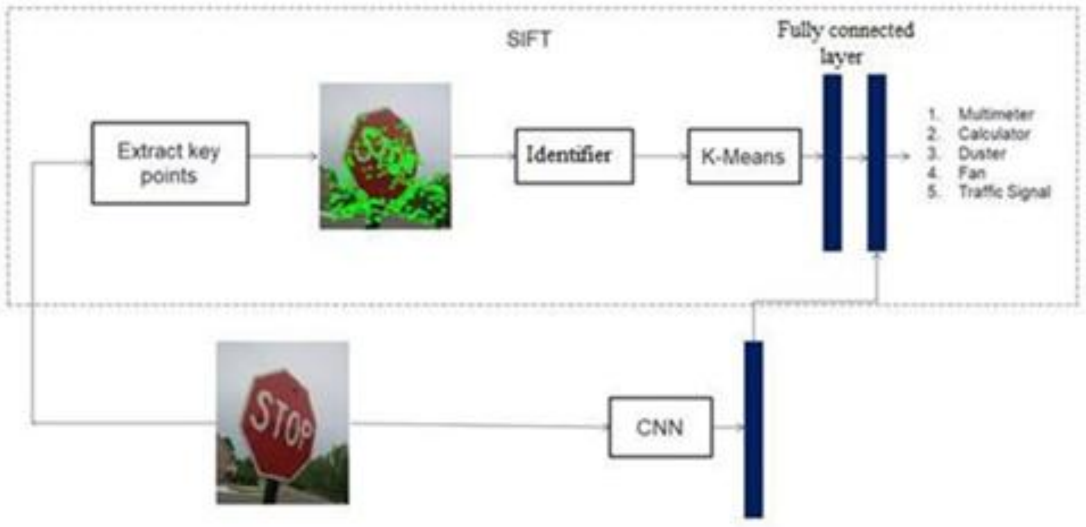
- https://onlinecourses.swayam2.ac.in/imb24_mg116/preview
- Introduction to Generative AI Course by Google Cloud | Coursera
- https://www.coursera.org/learn/generative-ai-for-everyone
- Generative AI Fundamentals | Coursera

Real Time Applications in Day to Day life and to Industry

1.Face detection: CNNs have been used to detect faces within images. The network takes an image as the input and produces a set of values that represent characteristics of faces or facial features at different parts of the image. CNN has shown improved accuracy over previous algorithms, identifying faces 97% of the time according to various research papers. CNN can also be used to help reduce the amount of distortion in a face. CNNs can detect features such as eyes, nose, and mouth with great accuracy reducing distortions from things like angles or shadows on faces.



2.Object detection: CNN has been applied to object recognition across images by classifying objects based on shapes and patterns found within an image. CNN models have been created that can detect a wide range of objects from everyday items such as food, celebrities, or animals to more unusual ones including dollar bills and guns. Object detection is performed using techniques such as semantic or instance segmentation. CNNs have been used to localize and identify objects within images as well as create different views of those objects such as for use in drones or self-driving cars.



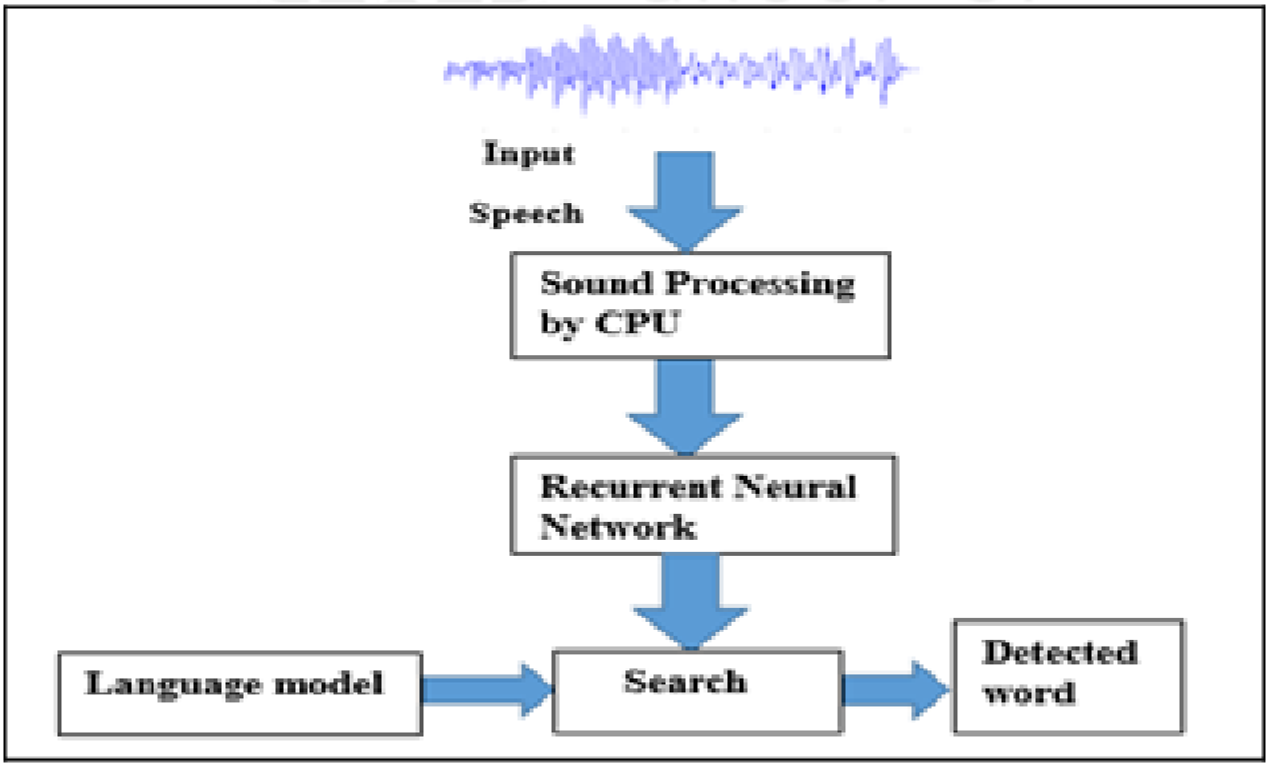
3.Speech Recognition

RNNs can be used for predicting phonetic segments considering sound waves from a medium as an input source .The set of inputs consists of phoneme or acoustic signals from an audio which are processed in a proper manner and taken as inputs. The RNN network will compute the phonemes and then produce a phonetic segment along with the likelihood of output.The steps used in speech recognition are as follows:-

The input data is first processed and recognized through a neural network. The result consists of a varied collection of input sound waves.

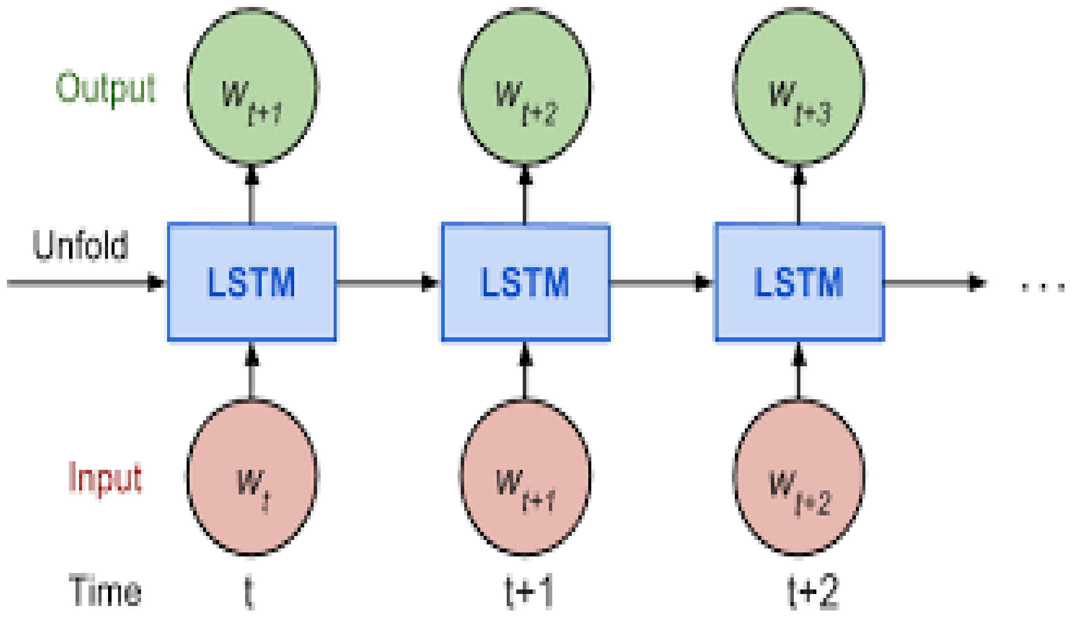
The information contained in the sound wave is further classified by intent and through keuwords related to the query.

Then input sound waves are classified into phonetic segments and are pieced together into cohesive words using a RNN application. The output consists of a pattern of phonetic segments put together into a singular whole in a logical manner.



4.Language Modelling and Generating Text

Taking a sequence of words as input, we try to predict the possibility of the next word. This can be considered to be one of the most useful approaches for translation since the most likely sentence would be the one that is correct.In this method, the probability of the output of a particular time-step is used to sample the words in the next iteration.



PRESCRIBED TEXT BOOKS AND REFERENCE BOOKS

TEXT BOOKS:

1.Ben Auffarth, Generative AI with LangChain, Packt Publishing, 2023.

2.Amit Bahree, Generative AI in Action, Manning Publication, First Edition, 2023.

REFERENCES:

1.David Foster, Generative Deep Learning, 2nd Edition, O'Reilly Media, 2023.

2.Numa Dhamani and Maggie Engler, Introduction to Generative AI, Manning Publication, First Edition, 2024.

3.Valentina Alto, Modern Generative AI with ChatGPT and OpenAI Models, Packt publications, 2024.

MINI PROJECT SUGGESTIONS

- **TEAM 1:** Neural Network Project on Automatic Music Generation System
- **TEAM 2:** Neural Network Project on Handwriting Recognition Tool using Autoencoders
- **TEAM 3:** Neural Network Project on Stock Market Value Prediction System using RNN

1. **TEAM 4:** Neural Network Project to Build a Vehicle Security System

- **TEAM 5:** Neural Network Project on Gender Recognition Systems

TOUGH QUESTIONS SIMILAR TO GATE

1. Which statement is false about GANs vs VAEs?

A) GANs minimize Jensen-Shannon divergence, VAEs minimize KL divergence.

B) VAEs provide explicit likelihood, GANs do not.

C) GANs are prone to mode collapse, VAEs generate blurry samples.

D) VAEs use adversarial training, GANs use reparameterization.

Ans: D

2. A 3×3 image undergoes 90° rotation followed by horizontal flipping.

The final pixel at (0,0) was originally at:

A) (0, 2)

B) (2, 0)

C) (2, 2)

D) (0, 0)

Ans: B.

3. Which is not a solution for mode collapse?

A) Mini-batch discrimination

B) Adding noise to discriminator inputs

C) Using Wasserstein loss

D) Increasing latent space dimension

Ans: D

TOUGH QUESTIONS SIMILAR TO GATE

4. What is the output of this perceptron?

```
import numpy as np

def perceptron(x, w, b):

    return 1 if np.dot(x, w) + b > 0 else 0

x = [1, -2]; w = [3, 4]; b = -5
```

A) 0

B) 1

C) Runtime error

D) -1

Ans: A

5. For a network with loss

$L = (y - \hat{y})^2$, $\hat{y} = \sigma(wx + b)$, what is $\frac{\partial L}{\partial w}$?

A) $2(y - \hat{y}) \cdot \sigma'(wx + b) \cdot x$

B) $-2(y - \hat{y}) \cdot \sigma(wx + b) \cdot x$

C) $2(y - \hat{y}) \cdot \sigma(wx + b) \cdot (1 - \sigma(wx + b)) \cdot x$

D) $-2(y - \hat{y}) \cdot x$

Ans: C (Chain rule: $\frac{\partial L}{\partial w} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z} \cdot \frac{\partial z}{\partial w}$, where $z = wx + b$.)

TOUGH QUESTIONS SIMILAR TO GATE

6. Gives the tensorflow code:

conv = tf.keras.layers.Conv2D(filters=32, kernel_size=3, strides=2,

padding='same')

What is the output shape for input (None, 64, 64, 3)?

A) (None, 32, 32, 32)

B) (None, 32, 32, 3)

C) (None, 64, 64, 32)

D) (None, 64, 64, 3)

Ans: A.

7. In an RNN, the hidden state

$$h_t = \tanh(W_{xh}x_t + W_{hh}h_{t-1} + b).$$

If h_{t-1} has 128 units and x_t has 64. What is

W_{hh}'s shape?

A) (64, 128)

B) (128, 128)

C) (64, 64)

D) (128, 64)

Ans : B8.

TOUGH QUESTIONS SIMILAR TO GATE

8. Which is true about Adam?

A) Combines momentum and RMSProp with bias correction.

B) Uses constant learning rate for all parameters

C) Does not track second moments of gradients.

D) Is equivalent to SGD with Nesterov acceleration.

Ans: A

9. Gradients vanish in a 10-layer ReLU network. Which change helps?

A) Replace ReLU with LeakyReLU.

B) Reduce learning rate.

C) Use SGD instead of Adam.

D) Remove batch normalization.

Ans: A

10. A 5×5 image convolved with a 3×3 kernel (stride=1, no padding) produces output of size:

A) 3×3

B) 5×5

C) 4×4

D) 2×2

Ans : A

THANK YOU