Unit - 3

Deep Learning

* It is a sub field of machine learning that focuses on training ANN to perform complex tasks by ~~levere~~ learning from large amount of data.

* It is inspired by the structure of human brain, mainly how the neurons are connected.

* Deep learning models consist of multiple layers of artificial neurons.

* It is highly effective for tasks like image recognition, natural language processing, speech recognition.

→ Neural Networks

↳ neural networks are ~~connected~~ composed of layers of interconnected nodes (neurons). Each neuron processes input data and passes its output to the next layer

→ Deep Architecture

→ Activation functions

→ Back Propagation

→ Loss Function

same topics again.

Machine learning

input → feature exteration → classification → output

Deep learning

input → feature extraction + classification → output.

Applications

→ Computer vision.

→ natural language processing

→ healthcare

→ autonomous systems.

Deep learning Architectures

1. CNN (Convolutional Neural Networks)

2. RNN (Recurrent Neural Networks)

3. LSTM (Long short term memory.

4. Autoencoders

5. GAN (Generative adversarial network)

6. Transformer. models

Transformer

↳ models combine an encoder and decoder architecture with a text-processing mechanism and have revolutionized how language models are trained

input
↓
encoder
↓
Self-Attention
↓
Feed Forward
↓
Decoder
↓
output

Applications

→ NLP
  ↳ google translate, GPT models

→ computer vision

→ multimodel tasks.

Historical Trends in Deep learning

→ **2000's**

Deep learning took off in 2000's with improved neural network algorithms and better computer.

→ **2012**

ImageNet showed the power in 2012 in CNN in image tasks.

→ **2014**

In 2014 GAN's (Generative Adversarial Networks) changed the game for creating synthetic data.

2016

AlphaGo in 2016 showcased complex games & impacting robotics.

2017 - present

→ Transformed Natural language process -ing tasks.

→ BERT and GPT models achieved breakthrough results

→ DeepSeek R1 model changed the way of training and using few amount of resources and implemented the Reinforce -ment learning etc.

→ Image generation models came into picture

Dall-e, ~~jeanne~~ Midjourney, Stable Diffusion.

GAN

Generative Adversarial Networks are class of deep learning models introduced by Goodfellow and colleagues in 2014. They are designed to generate new, realistic data by learning the underlying distribution of a given dataset.

components

(1) Generator

- A neural network that generates new data from random noise.
- The goal is to produce ~~th~~ data that is indistinguishable from real data

② Discriminator

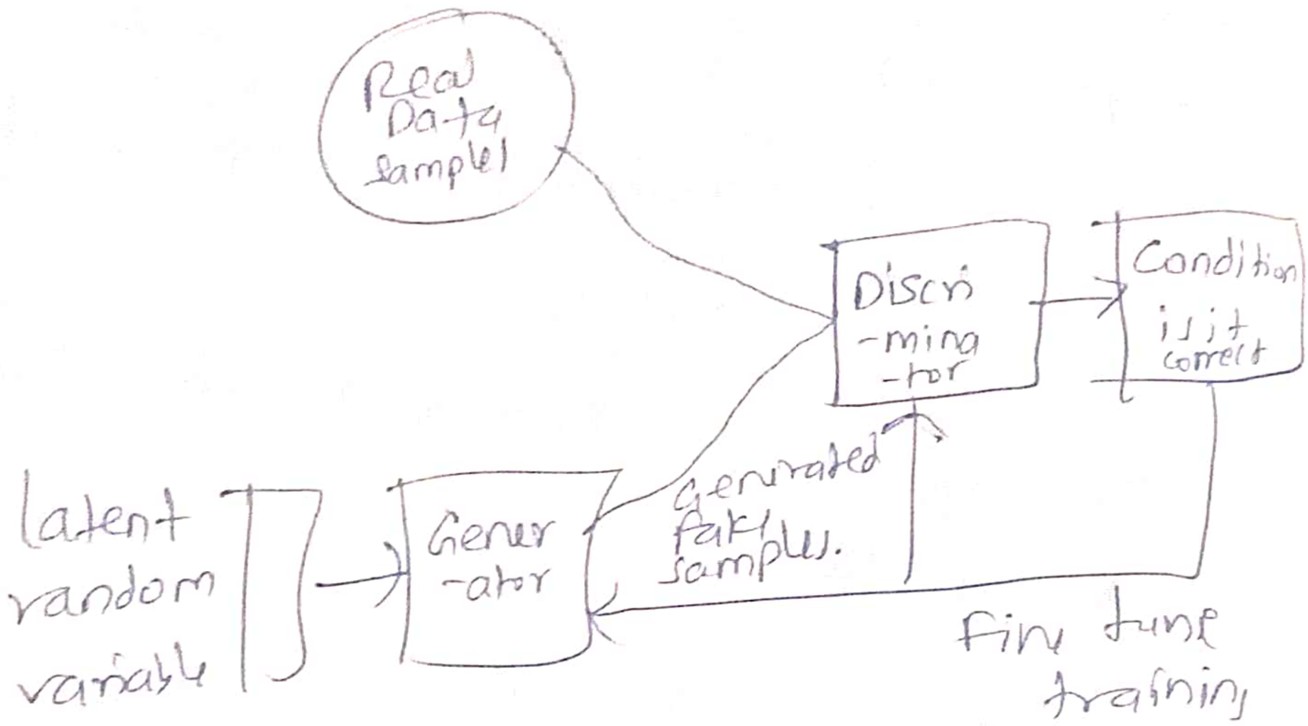
↳ A neural network that acts as a classifier, distinguishing between real data and fake data.

→ The goal is to ~~to~~ correctly identify whether the input is real or fake.

③ Adversarial Training

↳ The generator and discriminator are trained simutaneously in a competitive manner.

↳ The generator tries to fool the discriminator, while the discriminator tries to improve its ability to detect fake data.



Applications

→ Image generation
→ Image-to-Image Translation
→ Data Augumentation,
→ Art and creativity.

Deep Feedforward Network

It is known as feedforward neural networks (FNNs) or multilayer perceptrons (MLP), are the simplest and most fundamental type of

ANN. They are called feedforward because information flows in one direction.

Components

input layer
Hidden layer
output layer
Activation functions.

How it works

(1) forward propagation
(2) Loss function
(3) Back Propagation
(4) training.

Applications

(1) classification
(2) Regression.
(3) pattern Recognition

Gradient - Based Learning

It is a fundamental optimization technique used in training ML models, particularly neural networks. It involves using the gradient of a loss function with respect to model's parameters to iteratively update the parameters and minimize the loss.

Concepts

① Loss function

↳ it measures the difference between the model's predictions of the actual target values.

Ex:- Mean Squared Error for regression

Cross-entropy loss for classification

② Gradient

→ The gradient is a vector of partial derivatives of the loss function with respect to each parameter in the model.

→ It indicates the direction and magnitude of the steepest increase in the loss function.

③ Gradient Descent

→ An optimization alg. that uses the gradient to update the model's parameters.

→ The parameters are adjusted in the opposite direction of the gradient to minimize the loss.

→ update rule

↳ $\theta_{new} = \theta_{old} - \eta \cdot \nabla_\theta L(\theta)$

θ → model parameters

η → learning rate

$\nabla_\theta L(\theta)$ → Gradient of the loss function.

④ Learning Rate

→ A hyperparameter that controls the size of the steps taken during parameter updates.

Types

① Batch Gradient Descent

↳ computes the gradient using the entire training dataset.

② Stochastic Gradient Descent (SGD)

↳ computes the gradient using a single training example at a time

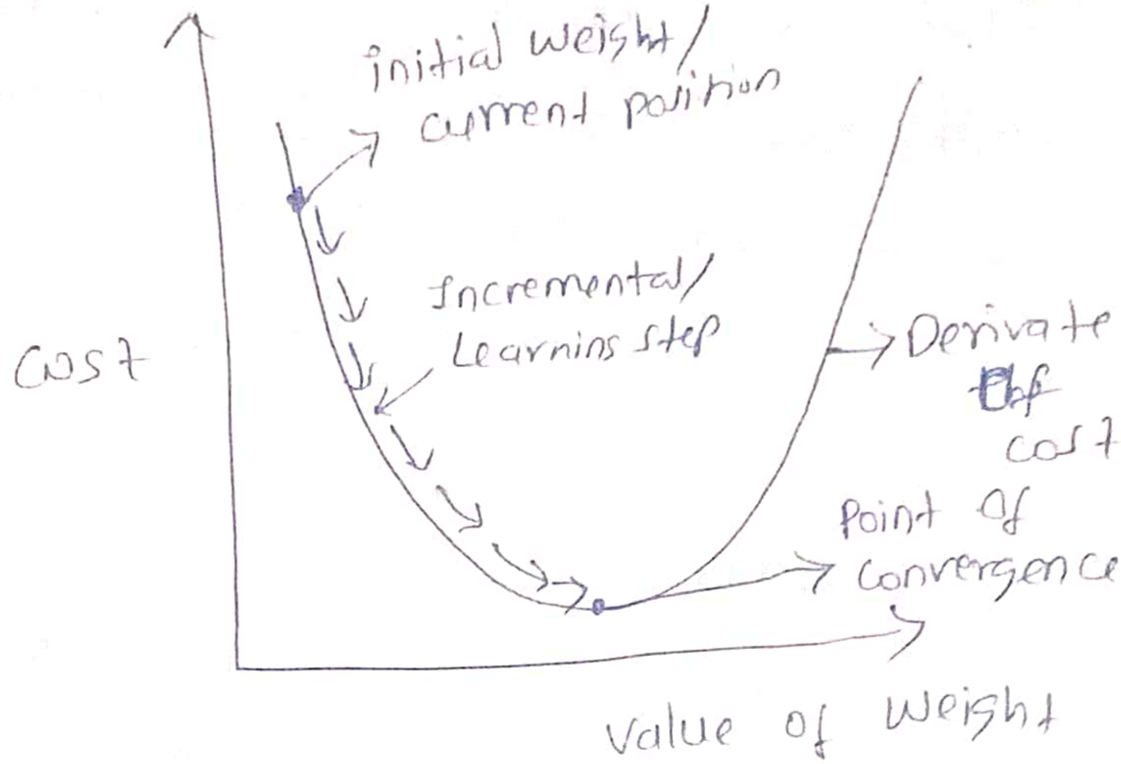
③ Mini-Batch Gradient Descent

↳ computes the gradient using a small batch of training examples

④ Advanced optimizers

→ momentum.

→ Adam



Applications

① Image recognition

② natural language processing

③ Speech recognition.

Hidden units

→ it is also known as hidden neurons or hidden nodes, are the building blocks of the hidden layers in a neural network. They play a critical role in enabling the network to learn complex patterns and representations from data. Hidden units transform the input data into a form that the output layer can use to make predictions.

Hidden units

↳ neurons located in the hidden layers of a neural network, between the input layer and the output layer.

Role of Hidden units

① Feature Extraction

↳ Hidden units automatically extract useful features from the input data, eliminating the need for manual feature engineering.

→ Each hidden unit learns to detect specific patterns or relationships in the data.

② Non-linerarity

→ Activation functions applied to hidden units introduce non-linerarity, enabling the network to model complex, non-linear relationships in the data.

Mathematical Representation

For single hidden unit in layer l:

$$z_j^{(l)} = \sum_{i=1}^{n} w_{ji}^{(l)} \cdot a_i^{(l-1)} + b_j^{(l)}$$

$$a_j^{(l)} = f(z_j^{(l)})$$

↳ weighted sum of inputs for the j-th hidden unit in layer l.

Activation of the i^{th} neuron in layer $l-1$

Activation functions

(1) ReLU
(2) sigmoid
(3) Tanh (Hyperbolic Tangent)
(4) Leaky ReLU.

Challenges

(1) overfitting
(2) vanishing/Exploding Gradients.
(3) computational cost.

Applications

(1) computer vision,
(2) Natural language processing
(3) General Machine learning.

Architecture Design

→ Designing the architecture of a neural network is a critical step in building effective ML models.

→ A well designed architecture blances model complexity, computational efficiency and performance

Key components

1. Input layer
2. Hidden layer
3. output layer
4. Activation functions
5. Loss functions
6. optimizer.

Step for Designing NN Architecture

1. Define the problem

→ Identify weather the task is regression, classification or another type.

→ understand the input data and desired output.

2. Choose the Number of layers

→ Start with a simple architecture and gradually increase complexity if need.

→ For deep learning tasks, use more layers to capture hierarchical features.

3. Choose the Number of Neurons per layer

4. Select Activation functions.

(5) Regularization

Add techniques like dropout or L2 regularization to prevent overfitting.

(6) Training strategy

→ Choose the optimizer and learning rate

→ Adjust the architecture based on results.

Common NN Architectures

(1) FNN
(2) RNN
(3) CNN
(4) Transformers.

Back Propagation and Other Differentiation Algorithms

- A feedward neural network accepts input x and produces output y, information flows forward through the network.
- Back-propagation algorithm allows the information from the cost to then flow backward through network in order to compute gradient.

Computational Graphs

- computational graphs are used to describe back-propagation algorithms more precisely.
- each node describes a variable.
- Graphs are accompaind by operations which is a simple function of one or more variables.

Chain Rule of Calculus

→ The chain rule is the backbone of back-propagation.

→ It helps us compute the derivative of a composite function.

Formula

$$\frac{dz}{dx} = \frac{dz}{dy} \cdot \frac{dy}{dx}$$

In neural networks:-

→ we use the chain rule to compute gradients layer by layer, starting from the output and moving backward to the input.

Recursively Applying the Chain Rule

→ Back-Propagation applies the chain rule recursively to compute gradients for all parameters in the network.

Steps

① compute the gradient of the output layer.

② use the chain rule to compute gradients for the previous layer.

③ Repeat until you reach the input layer.

→ This process is efficient because it avoid redundant calculations.

(or)

① Exact Differentiation Algorithms

These methods compute derivatives symbolically

1) Symbolic Differentiation

↳ computes derivatives using mathematical rules

b) <u>Automatic Differentiation (AD)</u>

↳ computes derivatives numerically using the chain rule but avoids expression swell.

<u>Two modes</u>

Forward mode.

↳ efficient for functions with few inputs.

Reverse mode

↳ Efficient for functions with many inputs. (used in BackProp)

c) <u>manual Differentiation</u>

↳ Deriving gradients by hand using calculus.

↳ often used for simple models or when performance is critical.

(2) Approximate Differentiation Algorithms

These methods estimate derivatives numerically, often used when exact derivatives are difficult to compute

a) Finite Differences

b) Stochastic Approximation

c) implicit Differentiation

d) Complex-step Differentiation.

(3) Hybrid methods

These combine exact and approximate techniques

a) Numerical Automatic Differentiation.

b) Differentiable Programming.