Name: Pranab Baro

Roll no:22b2224

SOS Mid Term Report

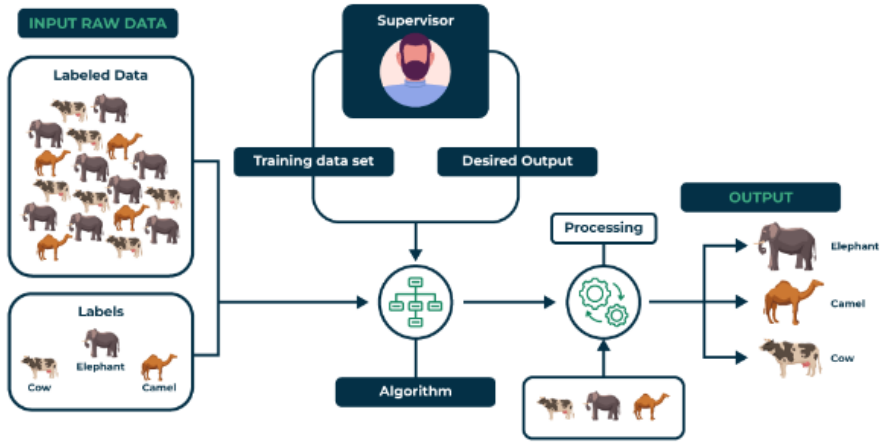
Topic:- Generative AI

Mentor: Piyush Raj

Introduction to Machine Learning

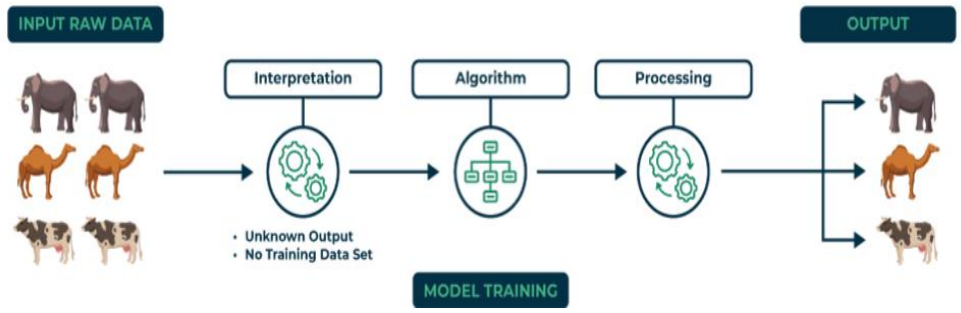
In simple words machine learning is the area of artificial intelligence in which computers are given the ability to learn from data. Instead of providing explicit instructions on how we want it to complete a task, we provide examples and justifications so that they can construct mental models that they can use in a variety of contexts. Several fundamental ideas in machine learning are:

Supervised Learning: It uses labelled data to learn.



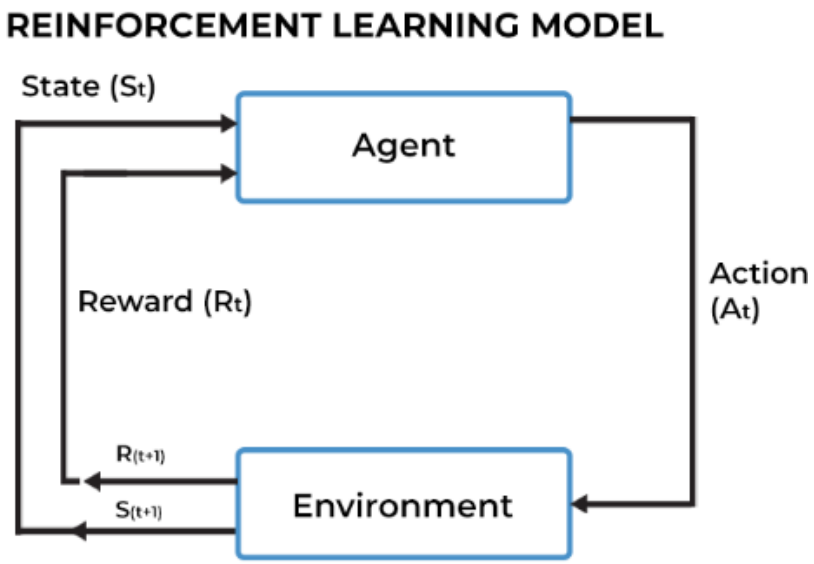
-The above figure explains mechanism of supervised learning which involves training from labelled data so that it can make predictions on fresh and new unseen data.

Unsupervised Learning: It works on unlabelled data to find any pattern or understanding.



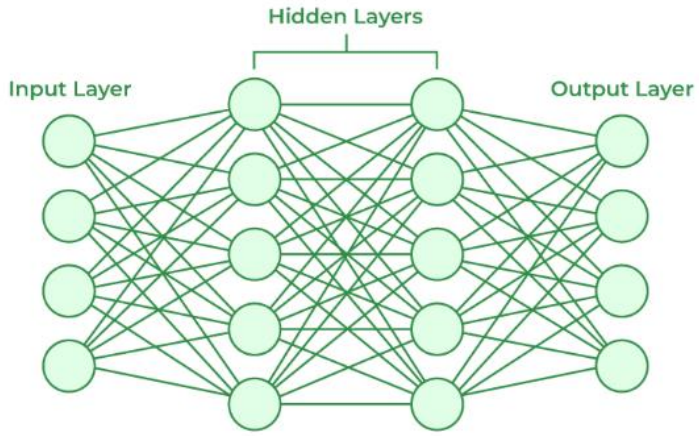
- Unsupervised machine learning uses various machine learning techniques to analyze as well as group unlabelled sets of data. To identify patterns or groups on its own, the training model needs the values of the input parameters.

Reinforcement Learning: To achieve the best outcomes reinforcement learning technique which imitates the trial-and-error learning process is used.



-Clearly from above figure we can conclude that as the model interacts with its surroundings it is rewarded for its desired behaviours and penalized for doing any error. Thus this way the model learns to maximize rewards.

Neural Networks



Neural networks is a method in AI which teaches computers to process data in such a way our human brain functions. In this process it uses interconnected nodes/ neurons in a layered structure. There are 3 types of layers:

Input Layer: This layer receives the input data and then process it and then send data to hidden layer.

Hidden Layers: This is an intermediate layer between input and output layer. It is used for learning complex pattern and change the input data.

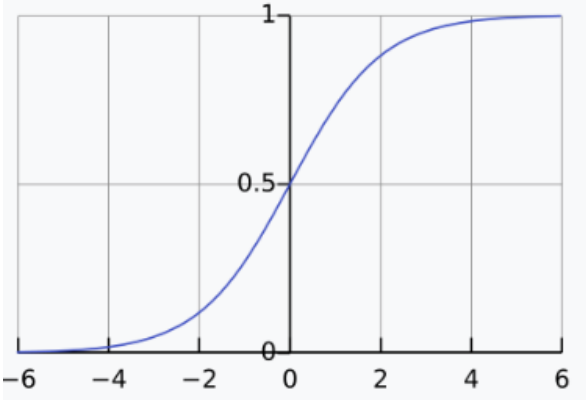
Output Layer: Output layer analyse the data from hidden layers, then makes the final prediction of output.

Activation Functions

The use of the activation function is to introduce non-linearity which allows it to identify different patterns in the input. Types of activation functions are written below:

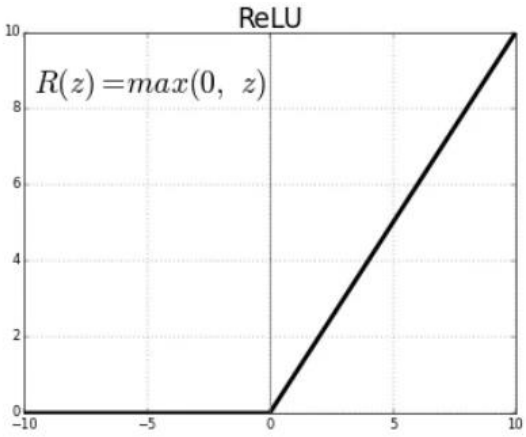
Sigmoid: σ(x)=1 / (1+e^{-x})

- The binary classification is the most fundamental kind of classification which is used when the variable we want to explain or the dependant variable has only two values. Its output range is limited to 0 to 1.



ReLU (Rectified Linear Unit):

ReLU(x)=max(0,x)



ReLU is the most used activation function right now. we can see that the ReLU is partially rectified (from the bottom) in the above figure. When z is less than zero, the function value (along y axis) become zero, when z is more than or equal to zero, function value is equals to z. It ease the training of deep learning models, such as neural networks, with deep structures.

Tanh or Hyperbolic Activation Function

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

- Its value varies from -1 to 1, which makes the model centred on the data utilized in the hidden layers. In the past, the tanh function was chosen over the sigmoid because it performed better in multi-layer neural networks.

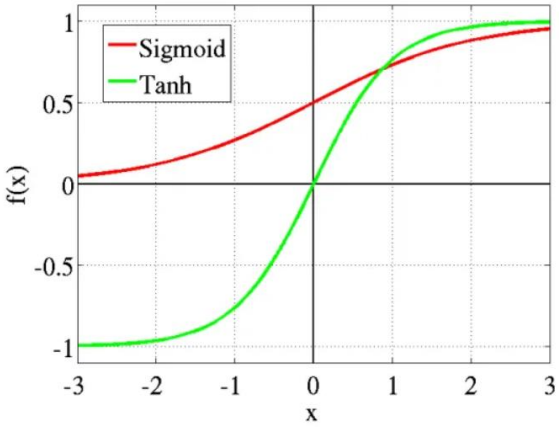


Fig: tanh v/s Logistic Sigmoid

Loss Functions

An essential component of neural network training is the loss function. The accuracy with which the predicted model fits the actual data is determined by mean squared error. The following list of typical loss functions is written:

- **Mean Squared Error (MSE)**:

  It is calculated using formula**: MSE=** $\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$ where n denotes no of data points, y_i denotes actual value of i-th data point and $\hat{y}_i$ denotes predicted value of i-th data point.

  - MSE function is used in regression tasks. It can be used in predicting house prices.

- **Cross-Entropy Loss**:

  Formula used to calculate loss**: Loss=** $-\sum_{i=1}^{n} y_i \log(\hat{y}_i)$

  - It is used for classification tasks. It can be used to predict category of an image.

Forward Propagation:- The method by which the data moves from input layer to output layer(left to right) is called forward propagation. Steps of how it's done are written below:

1) The input data is sent to the neural network's input layer.
2) The input data is processed through one or more hidden layers. Each neuron in a hidden layer receives inputs from the layer before it. It then applies an activation function to the weighted sum of these inputs and forwards the result to the layer after it.

3) Finally the processed data moves through the output layer then the network's output is produced.

Backward Propagation:- Backpropagation is an iterative algorithm that helps to minimize the cost function by determining which weights and biases should be adjusted. Feedforward neural networks are trained by using backward propagation technique.

The error can be calculated using formula :

Backpropagation error = Actual Output- Desired Output

Below are the specific steps in the process:

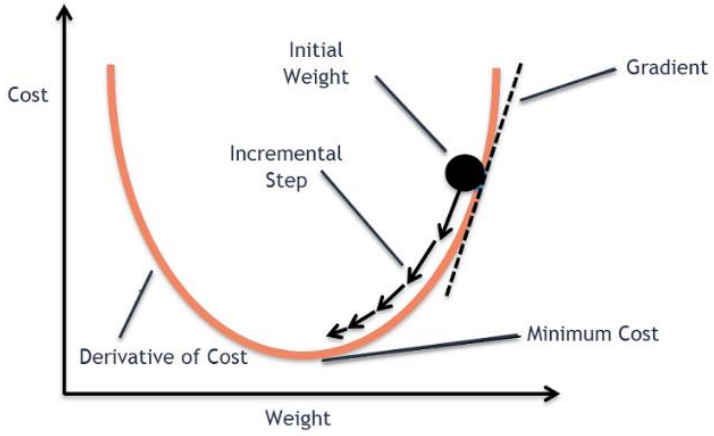
1) The deviation from actual output from the predicted output is calculated by using the loss function.
2) The error is then passed backward through each layers.
3) Now the gradients which are the loss function's slopes wrt each parameters (weights and biases) are calculated.
4) The gradients show the relative contributions of each parameter in the error.

Optimization Algorithms

This algorithm is used to find best possible solution of a problem. The main goal is to find a solution which minimizes the loss function. Different types of Optimization algorithms are written below:

Gradient Descent: Gradient descent is one of the optimization algorithm which are used to minimize the cost function by iteratively(repetitively) adjusting the parameters in negative gradient direction to find the ideal set of parameters.

-Formula: $\theta := \theta - \alpha \nabla_j J(\theta)$



Adam (Adaptive Moment Estimation): It combine advantages of 2 methods namely Adaptive Gradient algorithm (AdaGrad) and root mean square propagation (RMSProp)

Formula:

$$m_t=\beta_1 m_{t-1}+(1-\beta_1)\, g_t$$

$$v_t= \beta_2\, v_{t-1} + (1-\beta_2)\, g_t^2$$

$$\theta_t= \theta_{t-1} - \frac{\alpha}{\sqrt{v_t}} m_t$$

where m_t is first moment estimate, β_1 is decay rate of first moment estimate, m_{t-1} is previous moment estimate and g_t is gradient of loss function wrt parameters at time step t.

Regularization Techniques

Overfitting can be prevented with the help of regularization which adds an extra term to the loss function which penalizes values of the parameters that exceed certain limits. Common techniques are listed below:

1) L2 Regularization (Ridge Regression): It includes a penalty function which is the sum of the square of the parameter estimates.

-Formula: Loss=Loss + $\lambda \sum_{j=1}^{n} \theta_j^{\,2}$ where λ is regularization strength parameter, θ_j is the j-th values of model parameters,

2) Dropout: During the training time, it will set a proportion of input units to zero randomly in each update to reject over-fitting.

TensorFlow/PyTorch

TensorFlow and PyTorch, which are both machine learning libraries used in the development and training of neural networks.

TensorFlow: It is developed by Google Brain team and it can be used to develop models for various tasks like natural language processing, image recognition, handwriting recognition and many more. It particularly focuses on training and inference of deep neural networks.

PyTorch: It is used for applications such as computer vision and natural language processing, it was originally developed by Meta AI.

Gaussian Mixture Models (GMMs)

GMMs are used in data clustering and approximating the probability density functions.The data was generated from a mixture of Gaussian distributions.

Formula: Probability density function associated to any observed data point x.

$$\mathbf{p(x)=\sum_{k=1}^{K} \pi_k N(x|\mu_k \Sigma_k)}$$

where π_k are the mixture weights, μ_k are the means, $N(x|\mu_k \Sigma_k)$ denotes a Gaussian distribution, Σ_k are the covariances of the Gaussian components, K is the number of Gaussian components in the mixture model.

Hidden Markov Models (HMMs)

HMM is a statistical model. It is an assumption of a state process which is hidden and develops with time to generate the outcome data.

Components:

States: These are the variables which are hidden.

Observations: Data related to them that which are related with the states.

Transition Probabilities The number of state changes in a Markov chain or transitions to a different state are known as the transition probabilities

Emission Probabilities: The probability distribution of the corresponding given state.

Variational Inferences

Variational Inference is a way of approximating the distribution which is difficult to sample from or draw directly. Thus which results in conversion of problem into an optimisation problem.

Objective: The goal is to minimize the Kullback-Leibler (KL) divergence, which is a measure of the separation between the approximated true distribution and the generated true distribution; the closer to zero the better it is.

Formula:

KL(q(z | x)||p(z | x))=$E_{q(z|x)}$ [$log \frac{q(z|x)}{p(z|x)}$] where q(z |x) is approximate posterior distribution of latent variable latent variable z given observed data x, p(z |x) is true posterior distribution of latent variable z, $E_{q(z|x)}$ is the expectation wrt the distribution q(z |x).

Variational Autoencoders (VAEs): Variational Autoencoders (VAEs) are a specific kind of generative neural network. It has the capacity to learn representations in a latent space. They are based on neural networks known as autoencoders, which are taught to identify and replicate input data. The idea of a latent space which is lower dimensional than the original data set and aim to capture the most significant characteristics of the data.

Encoder-Decoder Architecture: This is a neural network architecture in which the encoder maps the input data features into a compact format and then used by the decoder to replicate the input data.

VAEs are of 2 main parts:

1)Encoder: It is used to map the input data into a latent space.

2)Decoder: Here the data gets converted back from the latent space to the original data space.

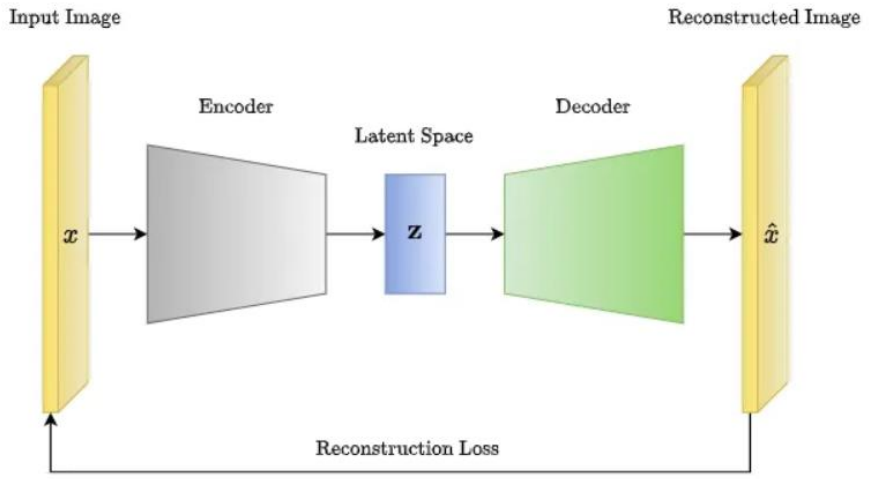


Fig: Autoencoder (AE)

Fully connected layers (MLP or CNN) make up the encoder and decoder in the most basic type of autoencoder. Minimizing the difference between the input data and the reconstructed output is what we aim for autoencoder training. Generally, a loss function such as binary cross-entropy or mean squared error is used to calculate the difference.

$$Loss_{MSE} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x}_i)^2$$

$$Loss_{BCE} = -\frac{1}{N}\sum_{n=1}^{N}\left[x_n \log \hat{x}_n + (1 - x_n)\log(1 - \hat{x}_n)\right]$$

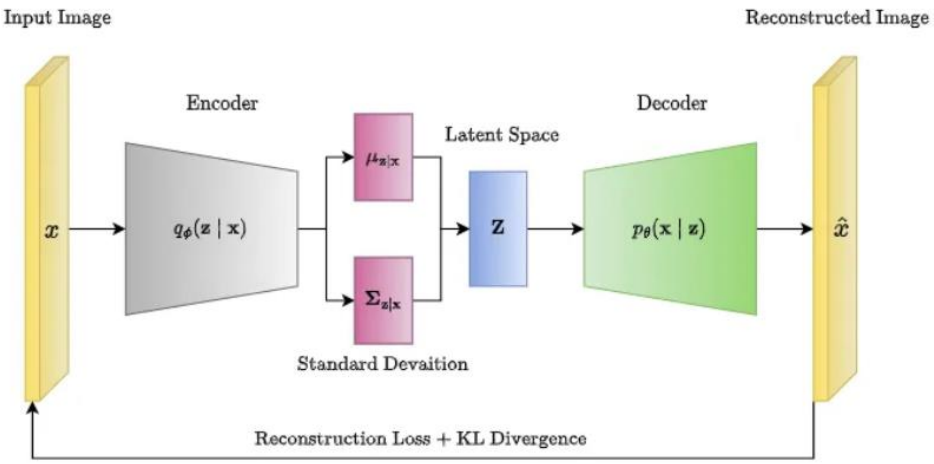


Fig: Variational Autoencoder (VAE)

Variational Autoencoders (VAEs) are a type of autoencoder which is introduced to overcome few limitations of traditional AE.

We define a new term in the loss function called *Kullback-Leibler (KL) divergence*. This term measures the difference between the learned probability distribution over the latent space and a predefined prior distribution (usually a standard normal distribution).

$$\log p_\theta\left(x^{(i)}\right) \geq \mathcal{L}\left(x^{(i)}, \theta, \phi\right) = \underbrace{\mathbb{E}_z\left[\log p_\theta\left(x^{(i)} \mid z\right)\right]}_{\text{Reconstruct the Input Data}} - \underbrace{D_{KL}\left(q_\phi\left(z \mid x^{(i)}\right) \| p_\theta(z)\right)}_{\text{KL Divergence}}$$

$$\theta^*, \phi^* = \arg\max_{\theta,\phi} \sum_{i=1}^{N} \mathcal{L}\left(x^{(i)}, \theta, \phi\right)$$

Reparameterization Trick

The reparameterization trick allows conducting backpropagation through the stochastic portion of the model since it switches the random component of a stochastic variable with some noise.

Formula: $z=\mu+\sigma\cdot\epsilon$ where $\epsilon \sim N(0,1)$, **μ** is the mean which represent centre of probability distribution and **σ** is the standard deviation

VAE Training

Training of VAE is done using the ELBO (evidence lower bound), which is a trade-off between the reconstruction loss and the KL divergence loss.

- **ELBO**: $\mathbf{L(\theta, \phi; x)=E q_\phi (z|x)[\log p_\theta \theta(x|z)]-KL(q_\phi(z|x)||p(z))}$

where $L(\theta, \phi; x)$ is objective function, θ is parameter of decoder network, ϕ is parameter for encoder network, x is observed data point, $Eq_\phi (z|x)[\log p_\theta \theta(x|z)]$ is expected log likelihood of x given z, $q_\phi(z|x)$ is approximate posterior distribution of z given x, $KL(q_\phi(z|x)||p(z))$ is Kullback-Leibler divergence between $q_\phi(z|x)$ and $p(z)$.

For example let us have a dataset of handwritten digits. This means that if VAE trained in this dataset it can learn a representation of the digits and generate new similar digits by sampling from the learned representation and passing the sample through the decoder.

Comparison between Autoencoder (AEs) vs Variational Autoencoders (VAEs)

-To decide which model is better it depends on the task it perform. AEs are better for tasks like dimensionality reduction and feature extraction. VAEs are better for generative tasks like image and text generation where we want to generate new data points.
-In terms of disadvantages autoencoders overfitting is an issue and may not be able to generate new data points. VAEs may experience mode collapse where the model generates similar outputs for different inputs.

Till now I have covered all topics till week 4 content according to my Plan of Action. I will be starting with the week 5 topics from net week and I hope to finish it as soon as possible.

References used:

1) **"Deep Learning" by Ian Goodfellow, Yoshua Bengio, and Aaron Courville**
2) **https://towardsdatascience.com**
3) **https://www.geeksforgeeks.org/variational-autoencoders**
4) **Wikipedia**
5) **YouTube lectures by Andrew Ng's**

Mentee: Pranab Baro (22b2224)
Mentor: Piyush Raj

Summer Of Science 2024 - Generative AI

Plan of Actions

Timeline:

Week 1: Learn machine learning basics, neural networks, activation functions, and loss functions through Andrew Ng's and Stefano Ermon Standford Online courses.

Week 2: Study forward/backward propagation, optimization algorithms, regularization techniques, and explore TensorFlow/PyTorch basics.

Week 3: Understand Gaussian Mixture Models, Hidden Markov Models, and Variational Inference from Bishop's book and YouTube tutorials.

Week 4: Dive into Variational Autoencoders (VAEs), focusing on encoder-decoder architecture, reparameterization trick, and VAE training.

Week 5: Explore Generative Adversarial Networks (GANs), including generator and discriminator networks, training process, and challenges.

Week 6: Study advanced GAN architectures such as Conditional GANs, CycleGANs, and StyleGANs, and their practical applications.

Week 7: Learn about Normalizing Flows, Autoregressive Models, and Energy-Based Models through YouTube playlist and research papers.

Week 8: Implement VAEs and GANs in PyTorch/TensorFlow, experiment with architectures, optimize hyperparameters, and complete a capstone project.

Books & Resources:

1) Deep Generative Models by Jakub M. Tomczak
2) Hands-On Generative Adversarial Networks with PyTorch by John Hany
3) YouTube playlist of Standford Online (CS236) by Stefano Ermon
4) Course by Deeplearning.AI on Coursera