IV - B.Tech I-Semester End Examinations (Regular) - Dec-2025

Deep Learning

(CSM)

Part - A

1. a) Give one example for each of Supervised and unsupervised learning algorithms

→ Supervised Learning ⇒ Linear Regression
⇒ Decision Trees

→ unsupervised Learning ⇒ K-Means clustering
⇒ Hierarchical clustering

b) what is feature engineering?

→ Feature engineering is the process of selecting, transforming, and creating new features from raw data to improve the performance of machine learning models.

c) How can norm Penalties be interpreted as a constrained optimization Problem?

→ Norm Penalties can be interpreted as a constrained optimization Problem by transforming the Problem of minimizing a function with a Penalty term into the equivalent Problem of minimizing the original function subject to a constraint on the norm of the Parameters.

d) Name one common Parameter initialization strategy

→ One common Parameter initialization technique is random initialization.

e) Give one real-world example where convolution is used in deep learning.

→ Image recognition.

f) How can unsupervised learning be used to learn convolutional features?

→ Autoencoders (or) discriminative tasks

g) what is meant by "unfolding" a recurrent neural network in time?

→ Expanding the recurrent loop over a sequence to show how it processes data at each time step, transforming the looped structure into a deep, feed forward - like network with layers for each step

h) what is explicit memory in neural network?

→ An architectural component that uses discrete memory cells to store and retrieve structured information separately from the networks implicit representations, enabling better reasoning.

i) why is it important to compare a new model against a baseline?

→ It provides a benchmark for performance, helps justify the use of complex models, and guides the iterative development process by

offering context for evaluation and identifying areas for improvement. ③

j) Give one example of deep learning applied in recommendation Systems.

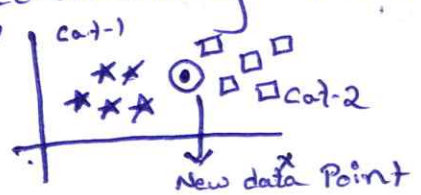
→ Using a two-tower model to balance Performance with Speed.

Part - B

2. Describe K-Nearest Neighbors (K-NN) algorithm and its applications in Supervised learning

→ • K-Nearest Neighbors (KNN) is a Supervised machine learning algorithm generally used for classification but can also be used for regression tasks.

- It works by finding the "K" closest data Points (neighbors) to a given input and makes a Predictions based on the majority class (for classification) or the average value (for regression).
- K-Nearest Neighbors is also called as a lazy learner algorithm because it doesnot learn from the training set immediately instead it stores the entire dataset and Performs computations only at the time of classification.
- For example consider the following graph of data Points containing two features.



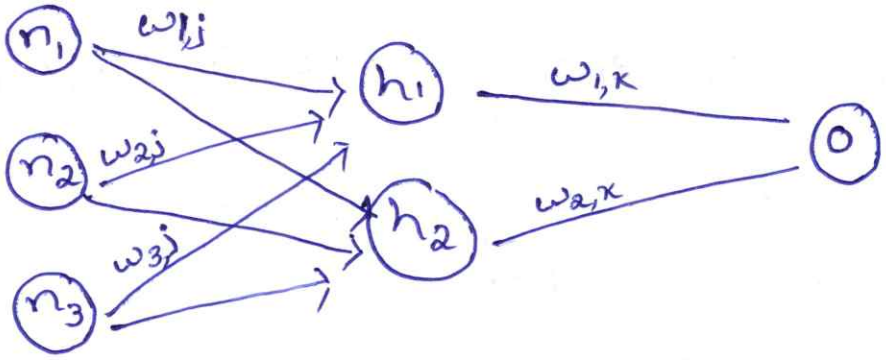
- The new point is classified as category 2 because most of its closest neighbors are squares. KNN assigns the category based on the majority of nearby points.
- In the K-Nearest neighbours algorithm 'K' is just a number that tells the algorithm how many nearby points or neighbors to look at when it makes a decision.
  - choosing the right 'K' is important for good results
  - If the data has lots of noise or outliers, using a larger 'K' can make the predictions more stable.
  - Statistical methods for selecting 'K' are cross-validation, Elbow method, odd values for K.
  - KNN uses distance metrics to identy nearest neighbor, these neighbors are used for classification and regression task. To identify nearest neighbor we use below distance metrics
    - Euclidean Distance. $d(x, x_i) = \sqrt{\sum_{j=1}^{d} \cdot (x_j - x_{ij})^2}$
    - Manhattan Distance $d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$
    - Minkowski Distance $d(x, y) = \left(\sum_{i=1}^{n} (x_i - y_i)^p\right)^{1/p}$
    - Applications are image and speech recognition, natural language Processing, Predictive analytics and healthcare.

3. Explain back Propagation algorithm for training a neural network. (5)

→ • In Simpler terms, backpropagation uses the chain rule to calculate the rate at which loss changes in response to any change to a Specific weight (or bias) in the network.

- The four main Steps in the backPropagation are
  - Forward Pass
  - Errors calculation
  - Backward Pass
  - weights update.
- There are two main types of back Propagation
  - Static
  - Recurrent.
- Static used in feedforward networks for tasks with static, independent inputs and outputs.
- Recurrent used for networks that Process Sequential or temporal data.
- Forward Pass work - In forward pass the i/p data is fed into the i/p layer. These inputs combined with their respective weights are passed to hidden layers. For example in a networ with two hidden layers (h1 and h2) the o/p from h1 serves as the i/p to h2. Before applying an activation fn, a bias is added to the weighted i/ps.

Each hidden layer computes the weighted sum (a) of the i/p's then applies an activation function like ReLU (Rectified linear unit) to obtain the o/p (o'). The o/p is passed to the next layer where an activation function such as softmax converts the weighted o/p's into probabilities for classification.



- Backward Pass work - In the backward Pass the error (the difference between the predicted and actual output) is Propagated back through the network to adjust the weights and biases. One common method for error calculation is the Mean Squared Error $MSE = (\text{Predicted output} - \text{Actual output})^2$

once the error is calculated the network adjusts weights using gradients which are computed with the chain rule. These gradients indicate how much each weight and bias should be adjusted to minimize the error in the next iteration. The backward Pass continues layer by layer ensuring that the network learns and improves its Performance The activation fn through its derivative plays a crucial role in computing these gradients during Back Propagation.

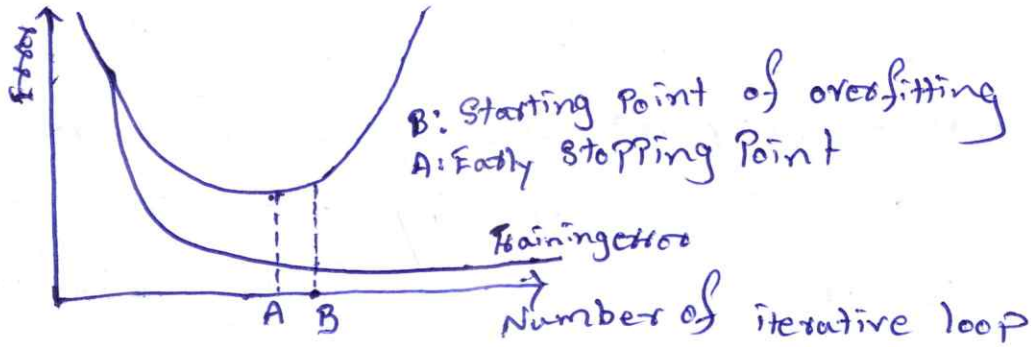
(4) Explain how early stopping can prevent overfitting and improve generalization with a training graph.

→ • Early Stopping is a regularization technique used in machine learning to prevent overfitting and improve the generalization ability of a model. It works by monitoring the model's performance on a seperate validation dataset during training and halting the training process when the performance on this validation set begins to degrade, even if the model is still improving on the training data.

• Early Stopping Prevents overfitting
  - Monitoring validation Performance
  - Detecting overfitting
  - Halting Training

• Training Graph illustration - Imagine a graph with the x-axis representing the number of training epochs and the y-axis representing the loss (or error). You would typically observe two curves:
  - Training Loss curve: This curve steadily decreases as the model learns to fit the training data better.
  - validation Loss curve: This curve initially decreases alongside the training loss,

indicating that the model is learning generalizable Patterns. However, at a certain point, the validation loss curve will reach a minimum and then start to increase, while the training loss may continue to decrease



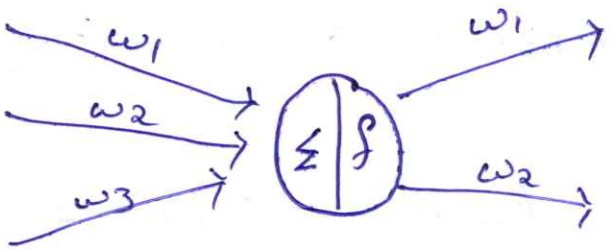
- In this graph the optimal stop point is where the validation loss reaches its minimum before starting to rise. Early stopping would halt the training at or shortly after this point (depending on the Patience setting), effectively preventing the model from clustering to train into the overfitting region where the training loss decreases but generalization suffers.

- Improving Generalization - By stopping training at the point where the model performs best on the validation data, early stopping ensures that the chosen model has learned robust, generalizable Patterns from the training data rather than memorizing specific examples. This lead to a model that can perform well on new, unseen data thus improving its generalization ability.

5. Discuss the importance of Parameter initialization and describe common initialization strategies.

→ • Parameter initialization is critical in training neural networks because it directly influences the model's ability to converge to an optimal solution efficiently and effectively.

- while building and training neural networks, it is crucial to initialize the weights appropriately to ensure a model with high accuracy.
- If the weights are not correctly initialized, it may give rise to the vanishing gradient Problem or the exploding gradient Problem.
- Consider the following neuron as a part of a deep neural network



- Common initialization strategies are
  - zero initialization
  - Random initialization
  - xavier/Glorot initialization
  - Normalized xavier/Glorot initialization
  - He initialization

- Orthogonal initialization

- zero initialization - All the weights are assigned zero as the initial value is zero initilization. This kind of initialization is highly ineffective as neurons learn the same feature during each iteration.
- Random initialization - Random initialization assigns random values except for zeros as weights to neuron paths.
- xavier / Glorot initialization - The weights are assigned from values of a uniform distribution as follows:

$$w_i \sim U\left[-\sqrt{\frac{\sigma}{fan\_in + fan\_out}}, \sqrt{\frac{\sigma}{fan\_in + fan\_out}}\right]$$

- He initilization - He initialization is similar to xavier but better suited for layers with ReLU activations. It scales the weights by the square root of the number of incoming neurons multiplied by 2.
- orthogonal Initialization - Initializes weight matrices to be orthogonal, helping to preserve the norm of the input vectors during backpropagation, which is especially useful for recurrent neural networks (RNN's).

6. Discuss the implications of this Prior on model design and generalization Performance in image recognition tasks.

→ 1. In Image recognition, the most significant Prior (Inductive bias) is the assumption of translation invariance/equivariance and local spatial correlation.

- These priors are inherently embedded in the design of convolutional Neural Networks (CNNs) and have design and generalization Performance.

- Implications for model design - These Priors directly lead to the fundamental architecture components of CNNs
  - Convolutional Layers
  - Local Receptive Fields
  - Pooling Layers (e.g., Max Pooling)
  - Hierarchical Architecture.

- Implications for Generalization Performance - The incorporation of this Prior significantly impacts a model's ability to generalize to unseen data.
  - Improved Generalization with less data
  - Robustness to Positional shifts.
  - Risk of information loss
  - Vulnerability to specific shifts

7. Explain methods for efficient computation (12) of convolutions in deep learning.

→ Efficient computation of convolutions in deep learning relies on minimizing arithmetic complexity, optimizing memory access patterns, and leveraging specialized hardware features. Key methods include.

- Data transformation into matrix multiplication
- using fast algorithms like winograd & FFT
- Architectural innovations such as seperable convolutions

- The Im2col + GEMM approach - The image to column (Im2col) method is a widely used technique in deep learning frameworks (e.g., NVIDIA's cuDNN library) to transform the convolution operation into a highly optimized general matrix multiplication (GEMM) problem.

- Fast Algorithms - These method reduce the fundamental arithmetic complexity (number of multiplications and additions) of the convolution operation itself.

- winograd Transform - This algorithm minimizes the number of multiplications required for convolutions, often by a factor of 2x or more for small kernel sizes. which are common in modern CNNs.

- Fast Fourier Transform (FFT) - The Convolution theorem states that convolution in the spatial domain is equivalent to point-wise multiplication in the frequency (fourier) domain.
- Architectural Optimizations - These approaches involve modifying the neural network architecture itself to use less intensive operations.
- Depthwise Separable Convolutions - This technique, notably used in Mobilenet architectures, splits a standard convolution into two smaller operations.
- Seperable Kernels (factorization): If a large 2D kernel can be mathematically decomposed into smaller 1D kernels (one horizontal and one vertical), applying two 1D convolutions sequentially is significantly faster than a single 2D convolution.

8. Discuss applications such as machine translation and speech recognition and explain how variable length input and output sequences are handled.

→ 1. Machine translation converts text or speech between languages, while speech recognition transcribes spoken language into text.

- Machine translation uses AI to automatically translate text or speech from one language to another.

- ~~Speech Recognition~~

- Speech recognition converts spoken language into written text. also known as Automatic Speech Recognition (ASR).

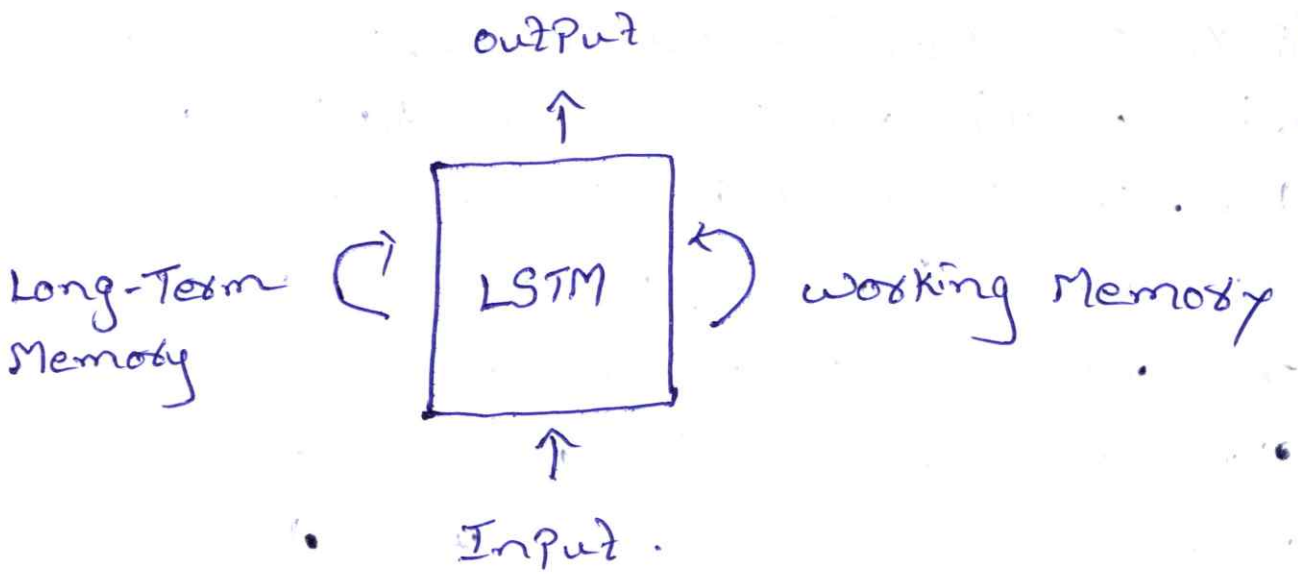
- variable length i/p and o/p sequences in machine translation and Speech recognition are primarily handled by Seq-to-Seq models.

- Encoder- Decoder Architecture (Seq 2 Seq) Models -
  - Encoder: This component processes the variable-length input sequence (e.g., a source language sentence in machine translation, or an audio waveform in Speech recognition). It converts this seq into fixed-length numerical representation called a context vector or thought vector.
  - Decoder: This component takes the context vector generated by the encoder as its initial input. It then generates the variable-length output sequence (e.g. a target lang sentence or a text transcription)

- Handling variable lengths in Practice.
  - Padding
  - End of Sequence (EOS) Tokens.
  - Connectionist Temporal classification (CTC)

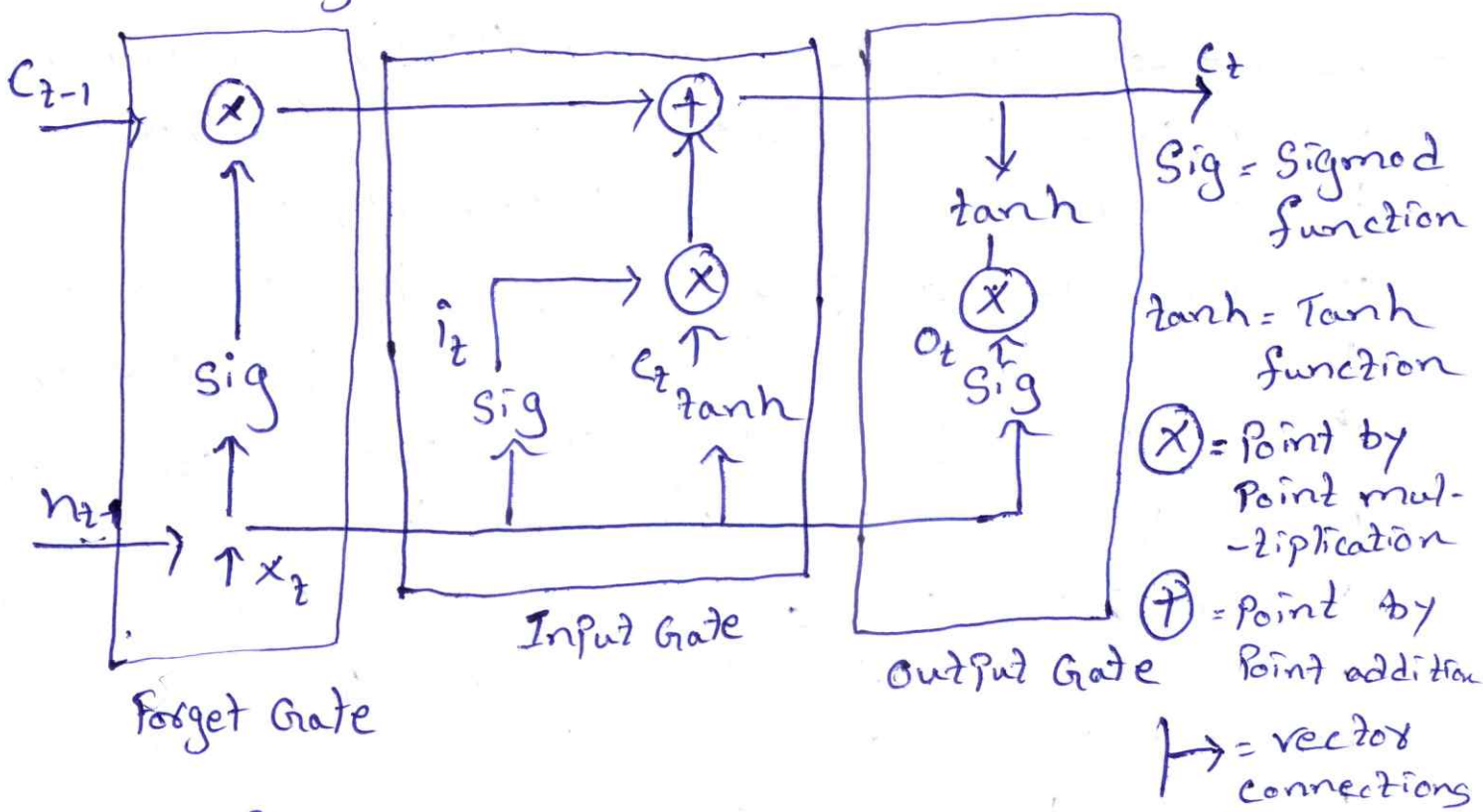
9. Explain the architecture of an LSTM cell with a diagram

→ LSTM - Long Short-Term Memory (LSTM) is an enhanced version of the Recurrent Neural Network (RNN) designed by Hochreiter and Schmidhuber. LSTMs can capture long term dependencies in sequential data making them ideal for tasks like language translation speech recognition and time series forecasting. Unlike traditional RNNs which ~~can~~ use a single hidden state passed through time LSTMs introduce a memory cell that holds information over extended periods addressing the challenge of learning long-term dependencies.



- Problem with Long-Term Dependencies in RNN
  - Vanishing Gradient.
  - Exploding Gradient.
- LSTM Architecture.
  - Input gate.

- Forget gate
- output gate



- The information that is no longer useful in the cell state is removed with the forget gate. Two i/p x_t and h_{t-1} are fed to the gate and multiplied with weight matrices follow by the addition of bias. The resultant is passed through sigmoid activation function which gives output.
- The addition of useful information to the cell state is done by the i/p gate. First the information is regulated using the sigmoid function and filter the values to be remembered Similar to the forget gate using i/p's h_{t-1} and x_t
- The output gate is responsible for deciding what part of the current cell state should be sent as the hidden state (o/p) for this time step

- Forget Gate.

$$f_t = \sigma(w_f \cdot (h_{t-1}, x_t) + b_f)$$

- Input Gate.

$$i_t = \sigma(w_i \cdot [h_{t-1}, x_i] + b_i)$$

$$C_t = \tanh(w_c \cdot [h_{t-1}, x_i] + b_c)$$

$$C_t = f_t \odot C_{t-1} + i_t \odot C_t$$

- output Gate.

$$O_t = \sigma(w_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = O_t \odot \tanh(C_t)$$

10. Discuss one method for systematically identifying and fixing Problems in a neural network's training Process

→ 7. One effective method for identifying and fixing Problems during neural network training is to monitor the training and validation loss/accuracy curves and use a systematic debugging strategy.

- Identification and Debugging Method:
  - Prepare a Small batch
  - Increase model capacity
  - Train and Monitor
  - Evaluate the result
    - Success (loss goes to near zero)
    - Failure (loss doesnot decrease or diverges)

- Loss goes up or explodes
- Loss oscillates
- Loss plateaus

- Fixing Problems with HyperParameter Tuning and regularization
  - overfitting
  - underfitting

- The other method of identifying and fixing problems in neural network's training process is Diagnosing with learning curves

11. Explain how deep learning is applied to Speech recognition. Include examples of architectures such as RNNs, LSTMs or Transformer-based models.

→ . Deep Learning is applied to Speech recognition by automatically Process audio, extract relevent features, and map them to text.

- The Process involves using a deep learning model, such as Recurrent Neural Network (RNN) or Convolution Neural Network (CNN), to Perform acoustic and language modeling, with the model learning to handle variations in accents, noise and dialects from vast amounts of data.

- Stages of deep learning in Speech recognition

- Pre Processing
- Feature Extraction
- Acoustic Modeling
- Language Modeling
- Decoding.

- Speech Recognition Systems leverage various neural network architectures to convert spoken language into text. RNNs, LSTM, and Tansformer-based model are prominent examples.
- RNNs are designed to process sequential data, making them suitable for speech signals. They maintain an internal "memory" that allows information from previous time steps to influence the process of current time steps
- LSTMs are a specialized type of RNN that address the vanishing gradient problem through the use of gates. These gates regulate the flow of information into and out of the cell state, enabling LSTMs to effectively learn and retain long term dependencies.
- Transformer-based models revolutionized Sequence-to-Sequence modeling with their self-attention mechanism, which allows them to weigh the importance of different parts of the input sequence when processing

each element. This enables them to capture long range dependencies more effectively than RNNs or LSTMs and in a more Parallelizable manner.

- RNN offer a foundational approach to Seq data but have limitations in handling long-term dependencies.
- LSTMs improve upon RNNs by mitigating the "vanishing gradient Problem and are effective for Capturing longer-range dependencies.