NATIONAL INSTITUTE OF TECHNOLOGY, TIRUCHIRAPPALLI – 620015
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
B.Tech – CYCLE TEST – II
CSPE72 - DEEP LEARNING TECHNIQUES

Date: 15.10.2025 **Time: 4.00 pm to 5.00 pm** **Max.Marks: 20**

Answer all questions

1. Why are Convolutional Neural Networks (CNNs) used in NLP, and how are they applied? How can you compare CNN within the attention paradigm? (4)
   - A convolution is a window that slides over a larger input data with an emphasis on a subset of the input matrix.
   - Filters are slid over these matrices to detect local patterns (similar to n-gram extraction). Each filter identifies phrases or linguistic features by combining neighboring word vectors.
   - Raw text is first converted into numerical representations (usually word embeddings like Word2Vec or GloVe), transforming sentences into matrices where each row is a dense vector for a word
   - Max or average pooling aggregations follow, reducing the feature map dimensionality and retaining the most relevant detected features. This step helps with computational efficiency and reduces the risk of overfitting.
   - Fully Connected Layers: Features from previous layers are integrated to produce final output predictions, such as sentiment labels or text categories.

   CNNs are applied in NLP by treating text data as structured sequences, typically using word or character embeddings as input, and then learning local feature patterns through convolutional and pooling operations. This enables automatic extraction of relevant linguistic structures, such as n-grams or key phrases, which are useful for various NLP tasks like sentiment analysis, classification, and named entity recognition.

   Steps of CNN Application in NLP
   1. Embedding Layer
   2. Convolutional Layer
   3. Activation and Pooling - ReLU
   4. Fully Connected Layer

   - CNNs use filters (kernels) that slide over the input (usually word or character embeddings) to extract local feature patterns such as n-grams or short word sequences.
   - They are efficient due to parameter sharing and local connectivity, but are inherently limited by the fixed size of their receptive field, which means they may struggle to capture long-range dependencies or relationships between distant words.
   - Attention, especially self-attention, enables models to dynamically determine the importance of each word in the context of every other word in a sequence.
   - It overcomes the fixed-context window limitation by allowing the model to directly connect distant positions, making it highly effective for capturing long-range dependencies, ambiguity, and complex structure in text, as seen in Transformer architectures

2. How do RNNs handle variable-length inputs? (3)

RNNs (Recurrent Neural Networks) handle variable-length inputs by processing each element in a sequence step by step, allowing the model to naturally accommodate sequences of different lengths without requiring the input size to be fixed.

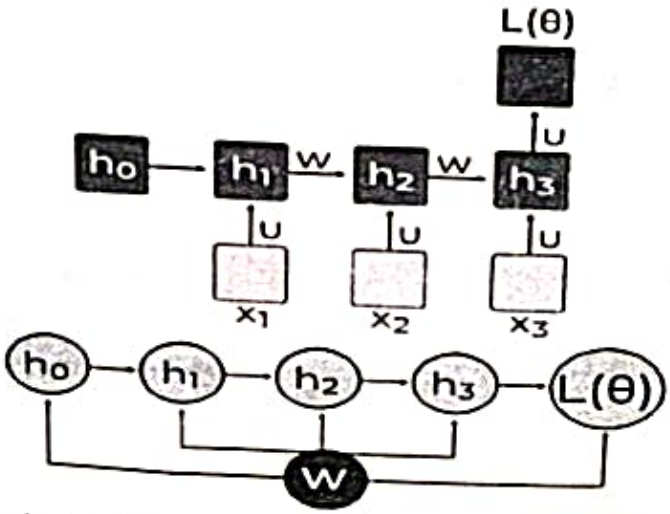
i.e., Sequential processing

RNNs can flexibly work with variable-length inputs by sequentially reading them and using padding or packed representations for efficient batch processing—a core advantage for NLP tasks involving sentences or texts of arbitrary length.

3. What is the role of Backpropagation Through Time (BPTT) in training RNNs, and how does it work? (3)

Backpropagation Through Time (BPTT)

- In RNN the neural network is in an ordered fashion and since in the ordered network each variable is computed one at a time in a specified order like first h1 then h2 then h3 so on.
- Hence we will apply backpropagation throughout all these hidden time states sequentially.
- Recurrent neural networks leverage backpropagation through time (BPTT) algorithm to determine the gradients, which is slightly different from traditional backpropagation as it is specific to sequence data.
- The principles of BPTT are the same as traditional backpropagation, where the model trains itself by calculating errors from its output layer to its input layer.
- These calculations allow us to adjust and fit the parameters of the model appropriately.
- BPTT differs from the traditional approach in that BPTT sums errors at each time step whereas feedforward networks do not need to sum errors as they do not share parameters across each layer



- L(θ)(loss function) depends on h3
- h3 in turn depends on h2 and W
- h2 in turn depends on h1 and W
- h1 in turn depends on h0 and W
- where h0 is a constant starting state.

$$\frac{\partial L(\theta)}{\partial W} = \sum_{t=1}^{T} \frac{\partial L(\theta)}{\partial W}$$

4. Under what conditions does a linear autoencoder act like Principal Component Analysis (PCA)? (2)

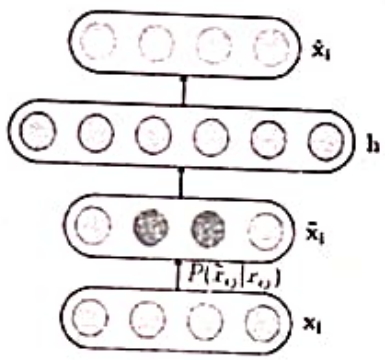
The encoder of a linear autoencoder is equivalent to PCA if we

- use a linear encoder
- use a linear decoder
- use a squared error loss function
- and normalize the inputs

5. Explain the difference between GRU and LSTM? (2)

Aspect	LSTM	GRU
Gates	Input, output, forget	Update, reset
States	Hidden state and cell state	Hidden state only
Memory/Long-term retention	Excellent for long-term dependencies	Good, but sometimes less effective for very long sequences
Training speed/complexity	Slower, more complex due to multiple gates and states	Faster, simpler structure
Use cases	Tasks with complex long-term structure (e.g., nuanced sentiment analysis, translation)	Real-time applications or shorter sequence tasks

6. Explain the concept of a denoising autoencoder and describe its working mechanism. (4)



7. Describe the typical steps involved in greedy layer-wise pretraining using unsupervised learning for a deep architecture. (2)

Greedy layer-wise pretraining with unsupervised learning for deep architectures involves training each layer of the network individually and sequentially before fine-tuning the entire model. The typical steps are:

Train the first layer unsupervised: Use an unsupervised model like an autoencoder or Restricted Boltzmann Machine (RBM) to learn features from raw input data. This layer learns a representation of the input without supervision.

Freeze the first layer weights: Once trained, the first layer's weights are fixed to preserve the learned features.

Train the next layer on transformed data: The output (features) of the first layer serves as input to train the second layer, again using unsupervised learning. This step allows the second layer to learn a more abstract representation based on the first layer's features.

Repeat for deeper layers: Continue this greedy process layer-by-layer, training each layer unsupervised on the output of the previous layer and freezing weights afterwards.

Fine-tune the entire network: After pretraining all layers, combine them into a deep network and fine-tune jointly using supervised learning (if labeled data is available), optimizing all weights end-to-end for the specific task.

This approach helps initialize deep networks effectively, addressing the vanishing gradient problem and improving generalization by learning hierarchical feature representations layer by layer before supervised fine-tuning.

Step 1: (Greedy) unsupervised pre-training

- Deep Belief Networks: Contrastive Divergence (CD-k) or SML/PCD
- Stacked Denoising Autoencoders: Back-propagation w/ cross-entropy loss

Step 2: Supervised fine-tuning

1) Toss old model, dump parameters into MLP

2) (Gentle) back-propagation fine-tuning

8. Explain the concept of Zero-Shot Learning. (2)

Zero-shot learning

- No labeled examples
- Ex: A learner reads a large collection of text and then solves object recognition problems
  - Having read that a cat has four legs and pointed ears, learner guesses that an image is a cat without having seen a cat before

Zero-data learning explained

- Possible because additional data exploited
- Zero-data learning scenario includes three random variables
  1. Traditional inputs x
     - Unlabeled text data containing sentences such as "cats have four legs", "cats have pointy ears')
  2. Traditional outputs y (y=1 indicating yes, y=0 for no)
  3. Description of task T (represents questions to be answered)
     - Is there a cat in this image?
- Model trained to determine conditional $p(y|x, T)$

Zero-Shot Learning allows AI systems to make intelligent predictions about unseen data by transferring knowledge through semantic relationships rather than relying on direct training examples.

***** END *****