Date:- / /
Page No.:-

DL Assignment : 3

Q1 Compare Feed-Forward Neural Network and Recurrent Neural Network ?

→

Features	Feed-Forward Neural Network	Recurrent Neural Network.
① Architecture	Layers are arranged in a sequence without cycles.	Contain cycles within the n/w. allowing feedback loops.
② Input Handling	Processes fixed-size input vectors independently.	Handles input sequence of variables lengths.
③ Memory.	Lacks explicit memory of past inputs	Maintains memory of previous inputs through hidden states.
④ Data Dependency	Each input processed independently of others.	Output depends on previous inputs due to recurrent nature.
⑤ Output Handling	Output depend solely on the current input.	Outputs can depend on previous inputs and hidden states.
⑥ Complexity	Typically simpler due to lack of temporal dependencies	More complex due to handling sequential dependencies.
⑦ Memory Efficiency	More memory-efficient for fixed-size inputs.	Less memory-efficient due to storing hidden states.
⑧ Example	Image classification using static features.	Language translation, speech recognition using sequences.

Q2. Describe the types of RNNs?

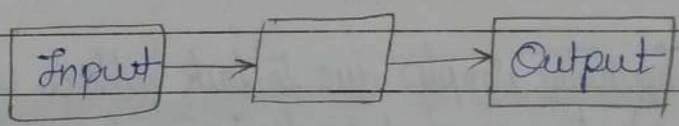
→

1. One-to-One
2. One-to-Many
3. Many-to-One
4. Many-to-Many

① One-to-One :-

- Plain Neural Network is another name for this.
- The simplest type of RNN is One-to-One.
- It allows a single input and a single output.
- It works with fixed-size input to fixed-size output, where they are unrelated to earlier data or output.
- The one-to-One application can be found in Image Classification
- It is a simplest form of RNN.

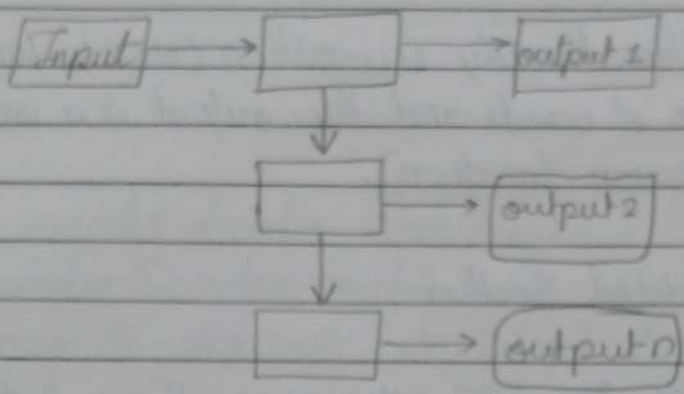
• Use Case :- Image classification, where the input is a ~~sig~~ single image, and the output is a single label.



② One-to-Many :-
- Takes a single input and produces a sequence of outputs over time.
- It takes a fixed input size and gives a sequence of data outputs.

- It applications can be found in Music Generation and Image Captioning.
- Example : Image Captioning takes the images as ~~an~~ input (Single Input) and outputs a sentence of words. (many output)

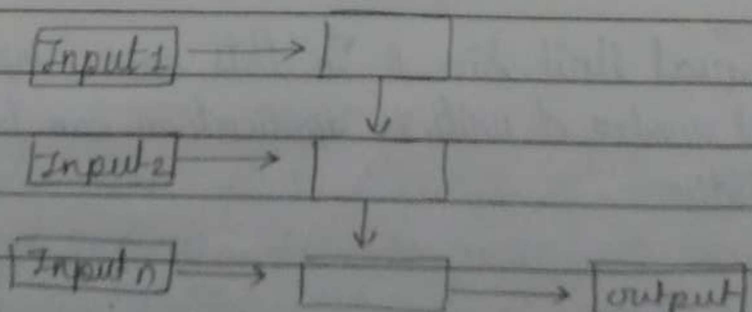
③ Many - to - One :



③ Many - to - One :

- It takes sequence of input and produce a single output.
- Useful when the ~~taks~~ task requires understanding the entire sequence to make a prediction.

Use case :- Sentiment analysis, where the input is a sequence of words (sentences) and the output is a sentiment label (positive/negative).



(4) Many - to - Many :

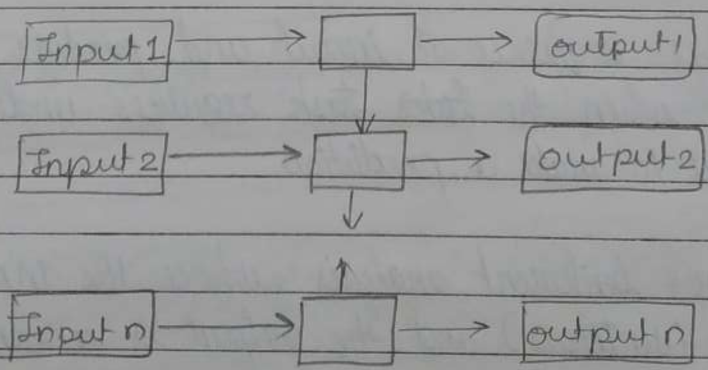
- It is used to generate a sequence of output data from a sequence of input units.

- The outputs are typically alligned with the inputs at each time step.

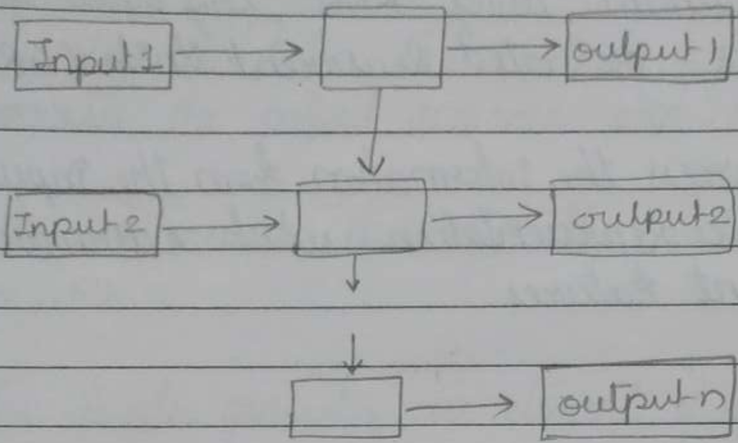
Use Case : Named entity recognition (NER), where the input is a sequence of words and the output is a sequence of entity labels (e.g. person, location).

- this is divided further.

(1) Equal Unit Size : In this case, the number of both the input and output units is the same. A common application can be found in Name-Enity Recognition.



(2) Unequal Unit Size : In this case, inputs and outputs have different numbers of units. Its application can be found in Machine Translation.



Q4. Explain encoder decoder architecture?

→ - The Encoder-Decoder architecture is a widely used framework in sequence-to-sequence (seq2seq) tasks, where the input and output are sequence of different lengths.

- This architecture is particularly useful for applications like machine translation, text summarization and speech recognition.

- this architecture allows the model to handle variable-length inputs and outputs, making it versatile for various sequence-based tasks.

Component of Encoder-Decoder Architecture.

① Encoder

- The Encoder takes in a sequence of inputs (eg., words in a sentence) and processes them to generate a context vector (also called the hidden state or thought vector).

- It can be implement using RNN, Long Short-term Memory (LSTM) networks, or Gated Recurrent Units (GRUs).

• Purpose : Compress the information from the input sequence into a fixed-size representation, which summarizes the inputs important features.

• Steps :
  - At each time step t, the encoder processess an input element x_i, and updates its hidden state h_i.

  - The final hidden state (or a combination of all hidden states) is passed as the context vector to the decoder.

② Context Vector
- The context vector is a fixed-length representation that encodes all the relevant information from the input sequence.

- It serves as the starting point for the ~~de~~ decoder, which uses this vector to generate the output sequence.

- The challenges with using a single context vector is that it may not capture all the input information in long sequences, which is why Attention Mechanisms are often added to improve performance.

Date:- / /
Page No.:-

(3) Decoder :

- The Decoder takes the context vector from the encoder and generates the output sequence step-by-step.

- Like the encoder, it is typically implemented with RNNs, LSTMS, or GRUs.

- The decoder generates one element of the output sequence at each time step. It uses the previous hidden state and the current input (often the previous output) to generate the next output.

- During training, the actual output from the previous time step is fed as input to the next time step. This is known as teacher forcing.

Steps :-

- At each time step t, the decoder generates an output y_t based on the context vector, the previous hidden state, and the previous output.

- The process continues until the entire output sequence is generated.

Detailed Workflow of the Encoder-Decoder Architecture :

1. Input Sequence :
   - A sequence of elements $(x_1, x_2, \ldots x_T)$ is fed into the encoder.

2. Encoding :
   - The encoder processes the sequence and compresses it into a context vector h_T (final hidden state or collection of hidden states.)

4. Decoding :
- The decoder uses the context vector to generate the output sequence $(y_1, y_2, \ldots y_t)$.

At each time step, the decoder uses the output from the previous step and the hidden state to predict the next output.

5. Final Output :
- The decoder continues generating the sequence until a specific stopping condition is met (such as generating an "end of ~~o~~ sequence" token).

Ex :- for ex, in English-to-French translation
- The encoder reads the English sentence : "How are you"? and encodes it into a context vector.
- The decoder uses the context vector to generate the French translation : "Comment ca va ?"

Q5 Short note on sequence-to-sequence model?
→ - A Sequence-to-Sequence (Seq2Seq) model is a neural network architecture designed to map an input sequence to an output sequence, often of different lengths. It is widely used in tasks where the input and output are sequence, such as machine translation, text summarization, and speech recognition.

Components :-
① Encoder : The encoder processes the input sequence (e.g., sentence in one language) and compresses it into a fixed-length

Date:- / /
Page No.:-

vector, also called the context vector. This vector summarize the information from the entire input sequence.

② Decoder : The decoder takes the context vector produced by the encoder and generates the output sequence (e.g., a sentence in another language). The decoder ~~work~~ works step-by-step, predicting ~~one another~~ ~~o~~ one output at a time based on its previous output and the context vector.

Architecture :-

- Seq2Seq model typically consist of two RNNs, LSTM n/w, or GRUs :
  - The Encoder RNN reads the i/p sequence
  - The Decoder RNN generates the o/p sequence.

Workflow :-

1. The encoder processes the input sequence and generates the context vector (final hidden state)

2. The decoder uses this context vector to produce the output sequence one element at a time.

Applications :-

- Machine Translation : Translating sentences from one language to another.

- Text Summarization : Generating shorter versions of long documents
- Speech Recognition : Converting speech into text.

Date:- / /
Page No.:-

Q 6. Where would you use sequence to sequence and why?

→ Sequence-to sequence (Seq2Seq) models are widely used in tasks where both the input and output are sequence of different lengths.

- here are key areas where seq2seq models are applied and the reasons for their effectiveness:

① Machine Translation :-

Use Case: Translating text from one language to another (e.g., English to French).

Why: Language have different sentence structures and lengths. Seq2seq models can handle variable-length sequence, making them ideal for translating sentence between languages.

② Text Summarization

Use Case: Generating a shorter summary from a long document.

Why: the input (long document) and output (short summary) have different lengths and seq2seq models can effectively capture and summarize key information.

③ Speech Recognition.

Use Case: Converting spoken language into written text.

Why: Speech (audio sequence) is transformed into a sequence of words. Seq2seq models are good at learnings the relationship between the spoken input and the text output.

Date:- / /
Page No.:-

④ Chatbots and Dialogue Systems
Use Case :- Generating responses in conversational agents.
Why :- Conversations involve variable-length sequence (questions & responses), and Seq2Seq models can map these sequence to provide coherent, context-aware responses.

⑤ Image Captioning :-
Use Case :- Automatically generating descriptive text for an image.
Why :- The image (processed as features) serves as input, and a sequence of words (description) is generated as output, a task well-suited for Seq2Seq models.

⑥ Time Series Prediction
Use Case :- Predicting future values in a time-series sequence.
Why :- Seq2Seq models can be used to learn patterns from historical data and predict sequence of future values, useful in financial forecasting and weather predictions.

Q.7 Explain Recursive Neural Network?
→ - A Recursive Neural Network (RvNN) is a neural network designed to handle data with hierarchical or tree-like structures, where smaller components combine to form larger, more complex structure.
- Unlike traditional (RNNs) that process sequential data, RvNNs work on inputs that can be represented as trees, like sentences in natural language or objects in images.

Concepts :-
1. Tree-Like Structure :-

Date:- / /
Page No.:-

- RvNNs process data structure as a hierarchy, where each nodes in the tree represents a combination of its childern. This makes it suitable for tasks like sentence parsing or hierarchical object recognition.

- The same neural function is applied recursively to the childern nodes of the tree, combining their information to form parent node representations. This recursive process continues until the root node is reached, which represents the entire inputs.

- The meaning or structure of a whole (eg. sentence or object) is determined by the meaning of its parts (words or components) and their relationships. This property makes RvNNs good at tasks where understanding the whole depends on the parts.

Architecture :-

- Leaf Node : The individual elements of the input (e.g., words or image components) are placed in the tree's leaf nodes.

- Recursive Function : At each non-leaf node, the network combines the information from its child nodes to form a new representation.

- Root Node : The final representation at the root node summarize the entire input sequences.

Date

Q. Explain Network architecture of Recursive Neural Network?

→ Network architecture of RNN operate on hierarchical or tree-~~combin~~ like structures, making them ideal for structured data, where smaller components recursively combine to form a comprehensive representation of the whole.

Components :-

① Tree Structure:

- Input is organized as a tree with atomic units (eg., words or images segments) at the leaves.
- Internal nodes combine child nodes into higher-level representations, using binary or n-ary trees depending on the application.

② Leaf Nodes (Input Embeddings):

- Each leaf node represents an atomic unit like a word or object part, typically as an embedding or feature vector.

③ Recursive Function (Internal Nodes):

- A neural function combines child nodes at each internal node.
- Mathematical Formula:

$$h_p = f(W|h_{1}; + h_2| + b)$$

- Combines hidden states h_1 and h_2 from child nodes using a weight matrix W, bias term b, and a non-linear activation function like tanh or ReLU.

④ Root Node:

- The final hidden state at the root node represents the entire input structure (eg., the meaning of a sentence).

⑤ Loss Function:

- At the root or intermediate notes, the output is compared to the target using a loss function.

- Backpropagation through Structure (BPTS) trains the model through the entire tree structure.

Example: Sentiment Analysis

- Leaf Nodes: Each word is represented by a word embedding.
- Internal Nodes: Words are combined into phrases like "not good".
- Root Node: The entire sentence is represented at the root.
- Output: A softmax layer classifies the sentence sentiment as positive or negative.

Q With diagram describe difference between Recursive Neural Network and Recurrent Neural Network?

→

Architecture	Network having hierarchical structure, Tree-like Structure	Chain-like structure known as Sequential structure.
Data Processing	It processes hierarchical data	It processes sequential and time series data.
Memory Handling	Limited context handling	Captures context through sequential memory.
Connections	Connections are based on hierarchical structure	Connections are based on sequential order.
Training Complexity	This network requires specific tree traversal algorithm for training.	It involves training back-propagation through time.
Dependency	Explicitly models dependencies in a tree structure.	Implicitly captures dependencies in sequences.
Use cases	Image parsing document structure analysis.	Language modeling, speech recognition.

Q.10 Describe types of RNN ? Explain application of RNN ?

→

① Standard Recursive Neural Networks :

- Combine child nodes in a tree-like structure to produce higher-level representations.
- Works on structured data like sentence parsing or hierarchical image representation.

② Tree-LSTM:

- Extends the LSTM model to work with tree structure.
- Has memory cells and gates (input, forget, output) at each node to manage information flow from child nodes.
- Ideal for complex hierarchical tasks where long-range dependencies need to be tracked.

③ Tree-GRU:

- Simplifies Tree-LSTM by using fewer gates (reset and update).
- Trades off some complexity for more efficient computations, still working on tree-structured data.

Application of RvNN:-

1. Natural Language Processing (NLP):

• Sentence Parsing: Breaking down sentences into grammatical commponents like subjects, verb and object.

• Sentiment Analysis: Understanding the sentiment (positive/negative) of sentences based on word combinations (e.g., "not good").

2. Computer Vision:

- Object Parshing: Decomposing an object into its parts (e.g., parsing an image of a car into wheels, doors, etc).

3. Hierarchical Data:

- Scene Understanding: Understanding a scene by combining objects & their relationships (e.g., person sitting on a chair in a room).