Recurrent Neural Networks

VVIT@2025

November 8, 2025

Table of Contents

Introduction

- Overview of neural networks
- Brief explanation of recurrent neural networks

What are Recurrent Neural Networks?

- Neural network architecture designed for sequential data processing
- Suitable for tasks involving time series, speech recognition, natural language processing, etc.
- Process input data of arbitrary length and maintain internal state
- Share parameters across different time steps

Key Concepts

- Hidden State
- Activation Function
- Backpropagation Through Time (BPTT)
- Long Short-Term Memory (LSTM)
- Gated Recurrent Unit (GRU)

Applications of RNNs

- Language Modeling
- Machine Translation
- Speech Recognition
- Image Captioning
- Sentiment Analysis
- Time Series Prediction

Training and Architecture

- Training RNNs using Backpropagation Through Time (BPTT)
- Challenges: Vanishing and Exploding Gradients
- Architectural variations: LSTM, GRU
- Deep RNNs and Bidirectional RNNs

RNN Architecture

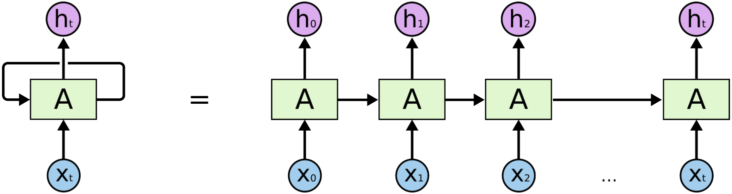


Figure: Illustration of an RNN architecture

Equations

- RNN forward pass equation: $h_t = \text{activation}(W_{hh}h_{t-1} + W_{xh}x_t + b_h)$
- RNN backward pass equation: $\frac{\partial L}{\partial h_t} = \frac{\partial L}{\partial h_{t+1}} \frac{\partial h_{t+1}}{\partial h_t}$

RNN Equations

- RNN forward pass equation: $h_t = \text{activation}(W_{hh}h_{t-1} + W_{xh}x_t + b_h)$
- RNN backward pass equation: $\frac{\partial L}{\partial h_t} = \frac{\partial L}{\partial h_{t+1}} \frac{\partial h_{t+1}}{\partial h_t}$

Types of RNN Architecture

- Simple RNN (Elman network)
- Long Short-Term Memory (LSTM)
- Gated Recurrent Unit (GRU)

Applications of RNNs

- Natural language processing
- Speech recognition
- Time series analysis

What Is a Neural Network?

A Neural Network consists of different layers connected to each other, working on the structure and function of a human brain. It learns from huge volumes of data and uses complex algorithms to train a neural net.

What Is a Recurrent Neural Network (RNN)?

A Recurrent Neural Network (RNN) works on the principle of saving the output of a particular layer and feeding this back to the input in order to predict the output of the layer.

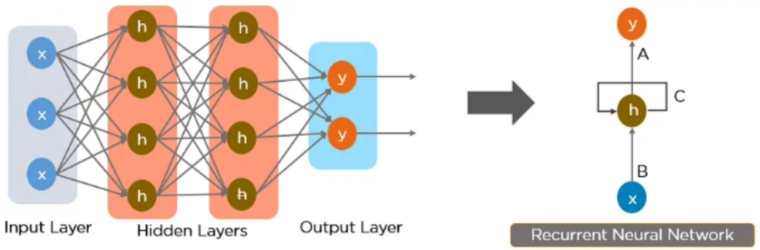


Figure: Recurrent Neural Network (RNN) Architecture

The nodes in different layers of the neural network are compressed to form a single layer of recurrent neural networks. A, B, and C are the parameters of the network.

Introduction

- Humans possess the ability to build upon prior knowledge for continuous understanding.
- Traditional neural networks lack this persistence, hindering sequential reasoning tasks.
- Recurrent neural networks (RNNs) provide a solution with their inherent looping structure.

Understanding RNNs

- RNNs consist of recurrent connections that allow information to persist across time steps.
- Each time step receives input and produces output, with hidden states acting as memory.
- Hidden states are updated at each step, incorporating current input and previous hidden state.

The Power of Sequential Reasoning

- Traditional neural networks lack the ability to leverage prior context for sequential tasks.
- RNNs excel in tasks requiring understanding of temporal dependencies and context.
- Example: Movie event classification based on previous events informs later predictions.

The Power of Sequential Reasoning

In sequential tasks, such as processing language, understanding time-series data, or analyzing events, context and temporal dependencies play a vital role.

- Traditional neural networks process inputs independently without considering the sequence or context.
- RNNs overcome this limitation by introducing recurrent connections that allow information to flow across time steps.
- At each time step, an RNN takes an input, produces an output, and updates its hidden state.
- The hidden state acts as memory, capturing information from previous steps and incorporating it with the current input.
- This enables RNNs to reason sequentially and leverage prior context, making them effective in tasks involving sequential data.
- For example, in movie event classification, an RNN can learn to recognize patterns and dependencies among events to make accurate predictions.

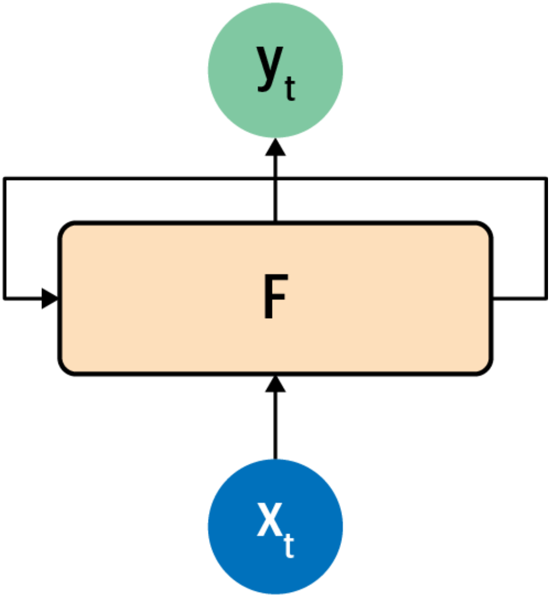


Figure: A recurrent neuron

Recurrent Neural Networks (RNNs)

- RNNs process sequential data, such as time series or text.
- A value x_t is fed into the function F at a time step t.
- The output y_t is generated based on x_t.
- RNNs have a feedback loop, allowing information to flow from one step to another.
- This feedback enables capturing temporal dependencies in the data.

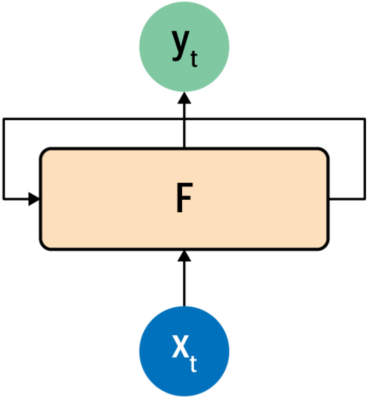


Figure: Recurrent Neural Network (RNN) Architecture

Recurrent Neural Networks (RNNs)

- RNNs process sequential data.
- At each time step, an RNN takes an input x_t and produces an output y_t.
- The RNN maintains a hidden state that captures information from previous steps.
- The hidden state is passed forward to the next time step.

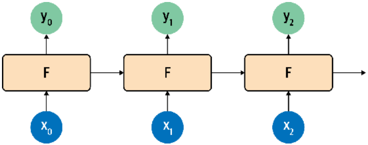


Figure: Recurrent Neural Network (RNN) Architecture

One-to-One

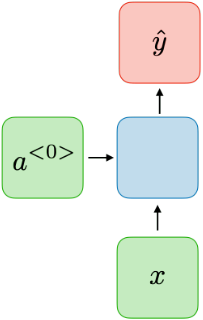


Figure: Illustration of an RNN architecture

Example: Traditional Neural Network

One-to-Many

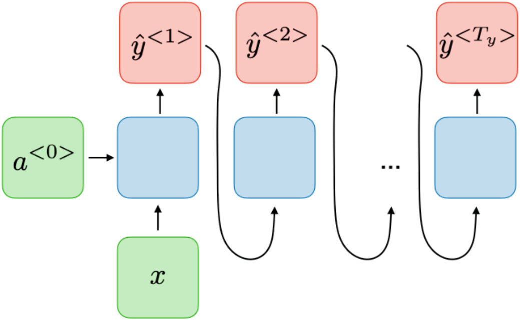


Figure: Illustration of an RNN architecture

Example: Language Modelling

Many-to-One

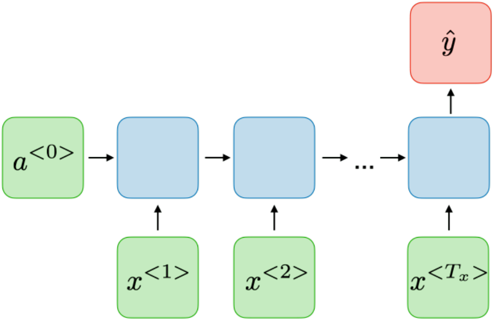


Figure: Illustration of an RNN architecture

Example: Sentiment classification

Many-to-Many

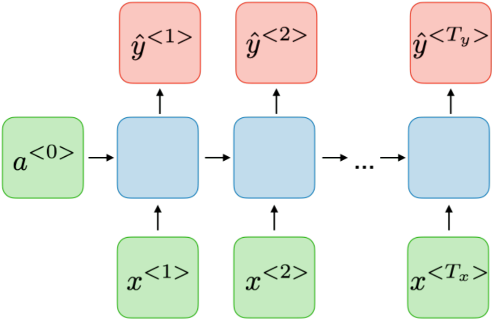


Figure: Illustration of an RNN architecture

Example: Name entity recognition

Recurrent Neural Networks (RNNs)

- RNNs process sequential data.
- At each time step, an RNN takes an input x_t and produces an output y_t.
- The RNN maintains a hidden state that captures information from previous steps.
- The hidden state is passed forward to the next time step.

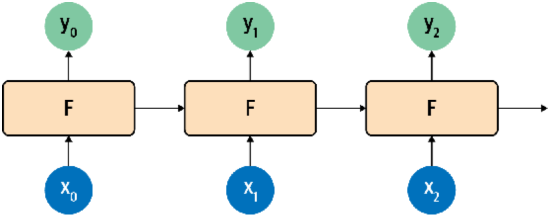


Figure: Recurrent Neural Network (RNN) Architecture

Why Recurrent Neural Networks?

- Feed-forward neural networks have limitations when it comes to handling sequential data.
- Sequential data, such as time series, text, and speech, cannot be effectively processed by traditional feed-forward networks.
- Feed-forward networks consider only the current input and lack the ability to capture temporal dependencies.
- They do not have memory to retain information about previous inputs.
- RNNs were designed to address these limitations and enable processing of sequential data.
- RNNs can capture dependencies over time and maintain a memory of past inputs.
- This makes RNNs suitable for tasks like speech recognition, language translation, and sentiment analysis.

Advantages of Recurrent Neural Network (RNN)

- Ability To Handle Variable-Length Sequences
  - RNNs can handle input sequences of variable length.
  - Well-suited for speech recognition, natural language processing, and time series analysis.
- Memory Of Past Inputs
  - RNNs have a memory of past inputs.
  - Captures information about the context of the input sequence.
  - Useful for tasks such as language modeling.
- Parameter Sharing
  - RNNs share the same set of parameters across all time steps.
  - Reduces the number of parameters to be learned.
  - Can lead to better generalization.
- Non-Linear Mapping
  - RNNs use non-linear activation functions.
  - Allows them to learn complex, non-linear mappings between inputs and outputs.

Advantages of Recurrent Neural Network (RNN) (contd.)

- Sequential Processing
  - RNNs process input sequences sequentially.
  - Computationally efficient and easy to parallelize.
- Flexibility
  - RNNs can be adapted to a wide range of tasks and input types.
  - Suitable for text, speech, and image sequences.
- Improved Accuracy
  - RNNs achieve state-of-the-art performance on various sequence modeling tasks.
  - Language modeling, speech recognition, and machine translation, among others.

Disadvantages of Recurrent Neural Network (RNN)

- Vanishing And Exploding Gradients
  - RNNs can suffer from the problem of vanishing or exploding gradients during training.
  - It becomes difficult to effectively train the network when gradients become too small or too large.
- Computational Complexity
  - RNNs can be computationally expensive to train, especially with long sequences.
  - Each input needs to be processed sequentially, leading to slow training times.
- Difficulty In Capturing Long-Term Dependencies
  - RNNs may struggle to capture long-term dependencies in the input sequence.
  - Gradients can become very small, causing the network to forget important information.

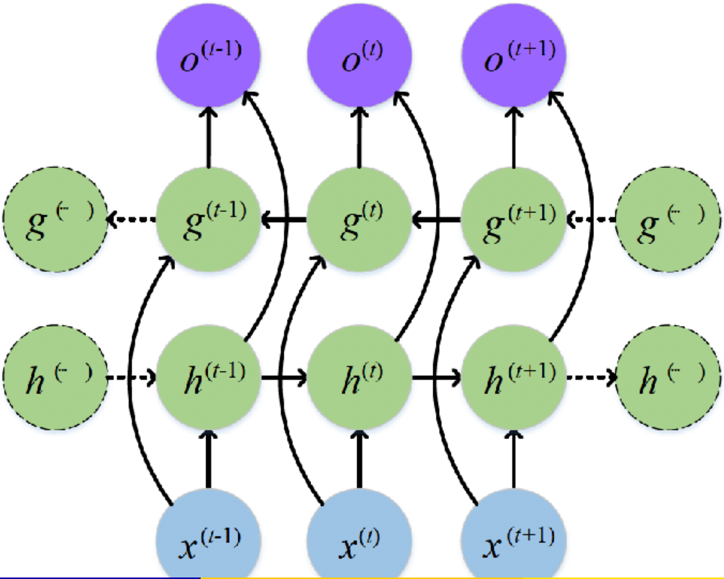
Disadvantages of Recurrent Neural Network (RNN) (contd.)

- Lack Of Parallelism
  - RNNs are inherently sequential, limiting parallelism in computation.
  - This can impact the speed and scalability of the network.
- Difficulty In Choosing The Right Architecture
  - There are many variants of RNNs with different pros and cons.
  - Selecting the appropriate architecture for a specific task can be challenging.
- Difficulty In Interpreting The Output
  - The output of an RNN can be challenging to interpret, especially for complex inputs.
  - Understanding how the network makes predictions can be difficult.

Bidirectional Recurrent Neural Networks (RNNs)

- Bidirectional RNNs process input sequences in both forward and backward directions.
- They consist of two separate hidden layers: one processing the sequence in the forward direction and the other in the backward direction.
- Each hidden layer maintains its own hidden state, capturing information from the past and future contexts.
- The outputs of both hidden layers are combined to produce the final prediction or representation.

Bidirectional Recurrent Neural Networks (RNNs)



Introduction to Long Short-Term Memory (LSTM) Networks

VVIT@2025

November 8, 2025

The Problem of Long-Term Dependencies

In certain tasks, recent information is sufficient for the present task. RNNs can effectively use past information when the gap between relevant information and its application is small.

- : In certain tasks, recent information is sufficient for the present task.
- : RNNs can effectively use past information when the gap between relevant information and its application is small.
- Language Modeling Example:
  - Language models predict the next word based on previous words.
  - Example: In the phrase "the sun rises in the," predicting the next word is straightforward.
  - The context of "the sun rises in" is sufficient to conclude that the next word will be "morning."
  - RNNs can successfully utilize the recent context to make accurate predictions.

Short term dependancies in RNN

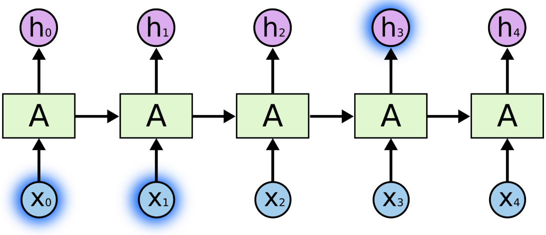


Figure: Example of a Recurrent Neural Network

Long term dependancies in RNN

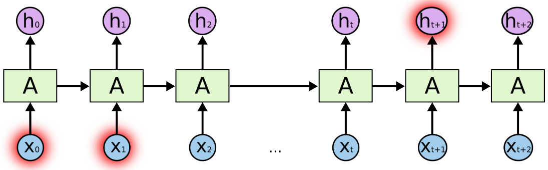


Figure: Long term dependancies in RNN

- I Grew up in Delhi.I did my schooling and college studies there. I speak fluent Hindi.

Long Short-Term Memory Networks (LSTMs)

In standard RNNs, the repeating module will have a very simple structure, such as a single tanh layer.

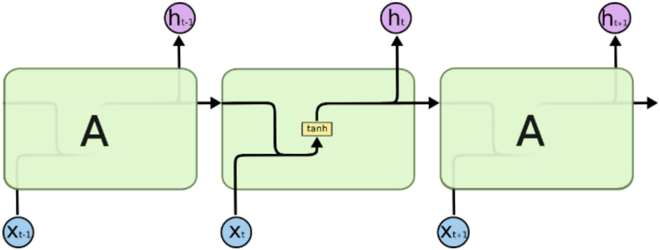


Figure: Long Short-Term Memory (LSTM) Structure

Long Short-Term Memory (LSTM)

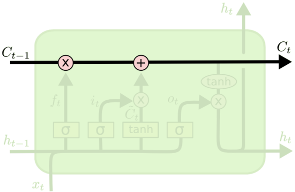


Figure: LSTM Cell Diagram

- The key to LSTMs is the cell state, represented by the horizontal line running through the top of the diagram.
- The cell state acts like a conveyor belt, running straight down the entire chain with minor linear interactions.
- Information can flow along the cell state largely unchanged.
- LSTMs have gates that regulate the removal or addition of information to the cell state.

What is LSTM?

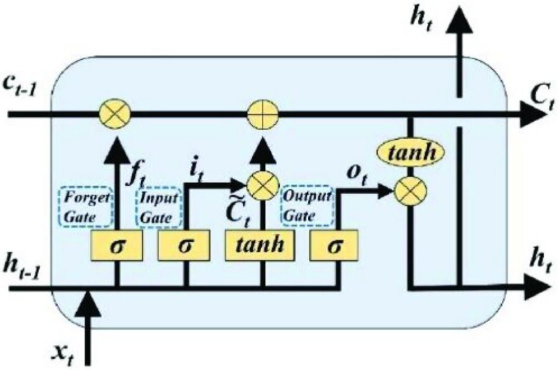
- LSTM stands for Long Short-Term Memory.
- A type of Recurrent Neural Network (RNN) that can learn long-term dependencies.
- Developed to overcome the vanishing gradient problem faced by traditional RNNs.
- Effective for tasks involving sequence prediction, language modeling, and more.

Why Use LSTM?

- Capable of learning from long sequences.
- Retains important information for longer periods.
- Controls the flow of information using gates.
- Widely used in speech recognition, machine translation, and time-series forecasting.

LSTM Architecture Overview

- LSTMs have a special architecture designed to remember and forget information.
- Composed of memory cells and gates (input, forget, output).
- Gates regulate the flow of information into and out of the cell state.



Forget Gate

Purpose

The forget gate decides what information from the cell state should be discarded or kept.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{1}$$

- f_t is the forget gate output.
- σ is the sigmoid function.
- W_f is the weight matrix for the forget gate.
- h_{t-1}: Previous hidden state.
- x_t: Current input.
- b_f: Bias term.

Input Gate

Purpose

The input gate decides which new information is stored in the cell state.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2}$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{3}$$

- i_t is the input gate output.
- $\tilde{C}_t$ is the candidate cell state.
- tanh: Hyperbolic tangent function.
- W_i and W_C are weight matrices.
- b_i and b_C are bias terms.

Output Gate

Purpose

The output gate decides what part of the cell state should be output to the next time step.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{4}$$

$$h_t = o_t \cdot \tanh(C_t) \tag{5}$$

- o_t is the output gate output.
- h_t is the new hidden state.
- W_o is the weight matrix for the output gate.
- b_o is the bias term.

Updating Cell State

Cell State Equation

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tilde{C}_t$$

- C_t: Current cell state.
- C_{t-1}: Previous cell state.
- $f_t \cdot C_{t-1}$: Forget gate's decision to discard or keep previous information.
- $i_t \cdot \tilde{C}_t$: Input gate's decision to add new information.

Applications of LSTM

- Language Modeling: Predicting the next word in a sentence.
- Machine Translation:Translating text from one language to another.
- Speech Recognition:Converting spoken language into text.
- Time Series Forecasting:Predicting future values based on past data.

GRU cell in practice

- Simplified LSTM
  - No cell state
  - Two gates (instead of three)
  - Fewer weights

- Update equations

  Reset gate: $r_t = \sigma(W^{(ir)}\bar{x}_t + W^{(hr)}h_{t-1})$

  Update gate: $z_t = \sigma(W^{(iz)}\bar{x}_t + W^{(hz)}h_{t-1})$

  Process input: $\tilde{h}_t = \tanh\left(W^{(i\tilde{h})}\bar{x}_t + r_t * \left(W^{(h\tilde{h})}h_{t-1}\right)\right)$

  Hidden state update: $h_t = (1 - z_t) * h_{t-1} + z_t \ * \tilde{h}_t$

  Output: $y_t = h_t$

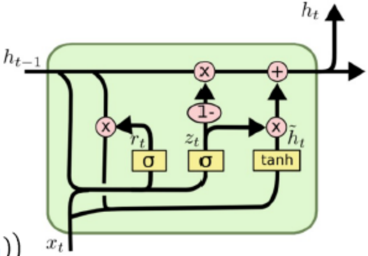


Figure: GRU Cell