Recurrent Neural Networks

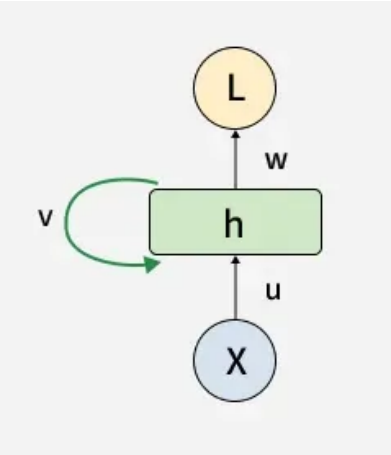
Recurrent Neural Networks (RNNs)—a class of deep learning models specifically designed to process **sequential data** (such as text, speech, time series, or video) where order and context matter. The core feature of a Recurrent Neural Network is its **recurrent loop**. Instead of passing information strictly forward in a single pass, an RNN feeds the network's state back into itself at every time step.

Key Components of RNNs

There are mainly two components of RNNs

1. Recurrent Neurons

The fundamental processing unit in RNN is a Recurrent Unit**.** They hold a hidden state that maintains information about previous inputs in a sequence. Recurrent units can "remember" information from prior steps by feeding back their hidden state, allowing them to capture dependencies across time.

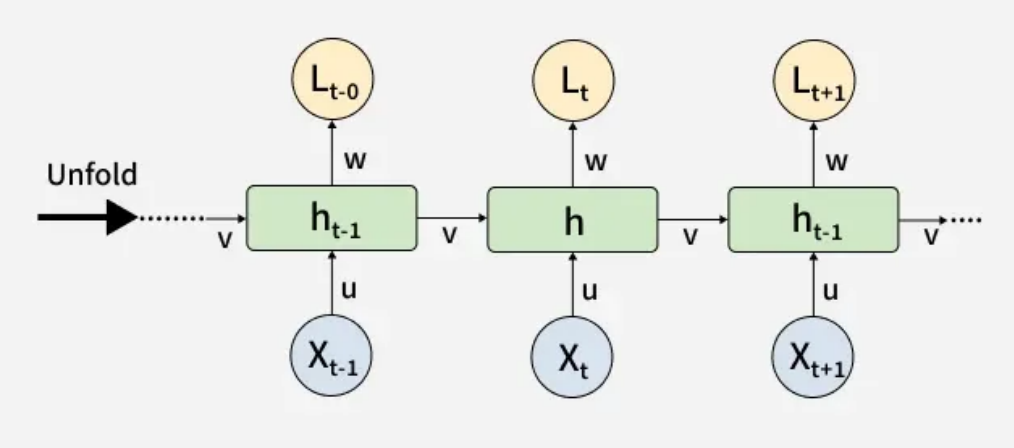


2. RNN Unfolding

RNN unfolding or unrolling is the process of expanding the recurrent structure over time steps. During unfolding each step of the sequence is represented as a separate layer in a series illustrating how information flows across each time step.

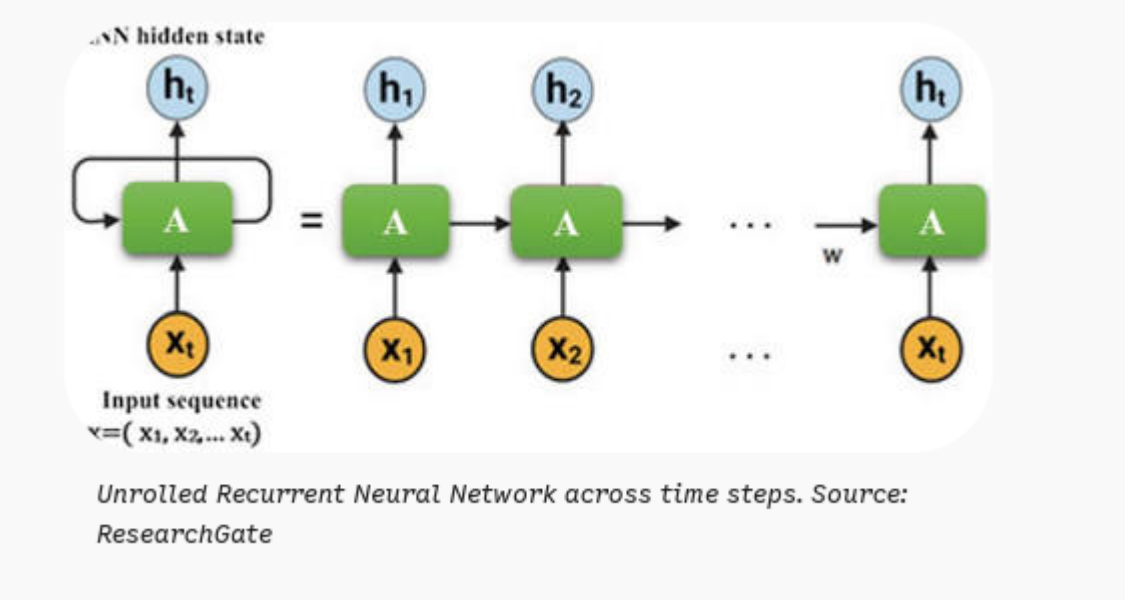
This unrolling enables backpropagation through time (BPTT) a learning process where errors are propagated

across time steps to adjust the network's weights enhancing the RNN's ability to learn dependencies within sequential data.



Recurrent Neural Network Architecture

Unlike standard Feedforward Neural Networks (FNNs) that process each input independently, RNNs possess an **internal memory** (hidden state, *ht*). When predicting the next step, the network considers both the **current input** (*xt*) and the **previous hidden state** (*ht*−1).



Unrolled Recurrent Neural Network across time steps. Source: ResearchGate

- **Hidden State Update:**

  $ht = \tanh(Whh \cdot ht-1 + Wxh \cdot xt + bh)$
- **Output Calculation:**

  $yt = Why \cdot ht + by$
- **Shared Weights:** RNNs share the same weight matrices (Whh, Wxh, Why) across all time steps, allowing them to generalize across sequences of varying lengths.

2. Training: Backpropagation Through Time (BPTT)

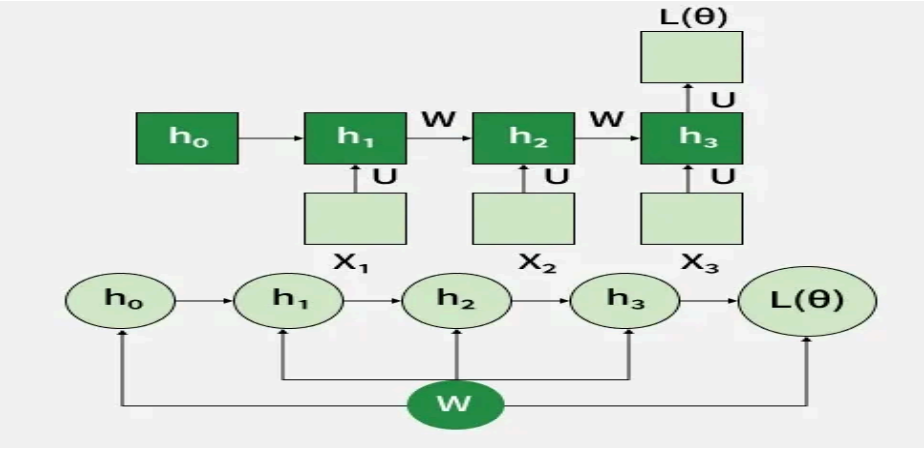
To train an RNN, the recurrent loop is "unrolled" over each time step into a sequence of layers. Gradients are then calculated and propagated backward through time step by step to update the shared weights.

Unrolling/Unfolding an RNN

To understand how data flows, we "unroll" or "unfold" the network across time. In an unrolled network:

- X_t: The input vector at time step t (e.g., word t in a sentence).
- h_t: The hidden state (memory) at time step t.
- y_t: The output at time step t.
- A: The shared recurrent cell/layer.

At each step t, the cell A takes two inputs: the current sequence item X_t and the previous hidden state h_{t-1}. It outputs a new hidden state h_t and optionally an output prediction y_t.



3. RNN Architectures by Input/Output Structure

- **One-to-One:** Standard classification with single input and single output (e.g., standard image classification).
- **One-to-Many:** Single input generates a sequence of outputs (e.g., Image Captioning).
- **Many-to-One:** Sequence of inputs maps to a single output (e.g., Sentiment Analysis).
- **Many-to-Many:** Sequence of inputs generates a sequence of outputs (e.g., Machine Translation, video frame tagging).

4. Common Variants & Advanced RNNs

1. **Vanilla (Simple) RNN:** The basic structure with a single hidden state using tanh. Susceptible to the **vanishing/exploding gradient problem** on long sequences.
2. **Bidirectional RNN (BiRNN):** Processes sequences in both forward and backward directions to leverage past and future context.
3. **LSTM (Long Short-Term Memory):** Uses internal gates (Input, Forget, Output) and a cell state to regulate information flow and solve the vanishing gradient issue.
4. **GRU (Gated Recurrent Unit):** A streamlined, faster variation of LSTM combining input and forget gates into an Update Gate.