Architecture of Recurrent Neural Networks

Idant Srivastava

January, 2026

Contents

Chapter 1

Introduction to Sequence Models

1.1 What are Sequence Models?

Sequence models are a class of machine learning models designed to process sequential data where the order of inputs matters. Unlike traditional feedforward neural networks that treat each input independently, sequence models capture temporary dependencies and relationships between elements in a sequence.

1.2 Characteristics of Sequential Data

Sequential data appears in many domains:

- **Natural Language**: Words in sentences, characters in text
- **Time Series**: Stock prices, weather patterns, sensor readings
- **Audio**: Speech signals, music
- **Video**: Frame sequences
- **Biological Sequences**: DNA, protein structures

Chapter 2

Recurrent Neural Networks (RNNs)

2.1 Basic RNN Architecture

RNN is a type of artificial neural network designed to recognize patterns in sequential data, such as text, audio, or time series, by using internal memory to process inputs. Unlike traditional networks, RNNs have loops that allow information to persist, using previous inputs to influence current, context-dependent outputs.

- **Sequential Processing**: They process data step-by-step (e.g., word by word in a sentence), making them ideal for tasks where order matters.
- **Hidden State (Memory)**: RNNs maintain a "hidden state" that captures information about what has been seen in previous time steps.
- **Parameter Sharing**: In each time step, the network uses the same weights and biases, which helps the model generalize across different parts of a sequence

2.2 Backpropagation Through Time

BPTT is the algorithm used to train RNNs by calculating how to adjust the model's weights to reduce errors in its sequential predictions.

- **Unrolling**: The network is conceptually "unrolled" into a long chain, where each time step is treated like a separate layer in a deep feedforward network.
- **Forward Pass**: The sequence is fed through this unrolled network to generate outputs and calculate the total loss at each step.
- **Backward Pass**: Gradients are then sent backward from the final step toward the first. Because the same weights are reused at every time step, BPTT sums the gradients from all steps to determine the final weight update.

Chapter 3

The Vanishing Gradient Problem

3.1 Mathematical Analysis

The vanishing gradient problem is a fundamental challenge in training RNNs. It occurs when gradients become exponentially small as they are backpropagated through time.

3.2 Consequences

The vanishing gradient problem leads to:

- **Long-term dependency failure**: The network cannot learn relationships between distant time steps
- **Slow learning**: Updates to parameters affecting early time steps become negligible
- **Preference for recent information**: The network biases toward recent inputs

Conversely, gradients can also explode in some cases, causing numerical instability. This is addressed through gradient clipping, but the vanishing gradient problem requires architectural solutions.

Chapter 4

Long Short-Term Memory Networks

LSTMs addresses the vanishing gradient problem through a sophisticated gating mechanism that allows the network to selectively remember or forget information over long sequences. The key innovation is the **cell state** C_t, which runs throughout the chain with only minor linear interactions, making it easier to preserve information over long periods.

4.1 Anatomy of an LSTM Cell

An LSTM cell consists of four main components: the cell state, forget gate, input gate, and output gate.

4.1.1 Forget Gate

The forget gate decides what information from the previous cell state is no longer relevant and should be discarded.

- **Working**: It takes the previous hidden state h_{t-1} and current input x_t, passing them through a sigmoid function.
- **Output**: A value between 0 (forget) and 1 (retain) for each part of the cell state.

4.1.2 Input Gate

The input gate determines what new information from the current time step is worth storing in the long-term cell state.

- **Working**: Sigmoid layer that decides which values to update and a tanh layer which creates a vector of new "candidate" values that could be added to the state.
- **Update**: These two are multiplied and added to the cell state. Together, they determine what new information to store: $i_t \odot \tilde{C}_t$.

4.1.3 Output Gate

The output gate controls what information from the cell state to expose in the hidden state. It filters the cell state through tanh (to get values in $[-1, 1]$) and then multiplies by the sigmoid output to decide which parts to output.

Chapter 5

Gated Recurrent Units (GRUs)

5.1 Motivation

GRU is a streamlined version of the LSTM, introduced to solve the vanishing gradient problem with a more efficiency. Unlike LSTMs, which use three gates and two separate memory states, the GRU merges these into two gates and a single hidden state.

5.2 GRU Architecture

The GRU manages information flow using these specific mechanisms:

- **Update Gate (z_t):** This gate acts as a "long-term memory" controller. It decides how much of the previous hidden state should be kept and how much new information from the current step should be added.
  - **Value near 0:** Keeps the majority of the past state.
  - **Value near 1:** Replaces the past state with new information.
- **Reset Gate (r_t):** This gate handles "short-term memory" by deciding how much of the past hidden state is relevant to the current calculation.
  - **Value near 0:** Discards previous information, allowing the model to focus entirely on the new input (useful for sudden changes in a sequence).
  - **Value near 1:** Allows the previous hidden state to fully influence the current candidate calculation.

The final output, or hidden state (h_t), is calculated in two main phases:

- **Candidate Hidden State**: The model creates a "proposal" for the new state by combining the current input with a version of the previous hidden state.
- **Interpolation**: The Update Gate then performs a linear interpolation between the old hidden state and this new candidate. This direct path allows information to persist over many time steps, effectively fighting vanishing gradients.