Architecture of Recurrent Neural Networks

1 Introduction

Recurrent Neural Networks (RNNs) are a class of neural networks designed to handle sequential data, such as time series, text, and speech. Unlike feedforward neural networks, RNNs maintain a hidden state that allows information from previous time steps to influence the current output.

Over time, standard RNNs were found to suffer from serious training issues, most notably the **vanishing gradient problem**. This led to the development of more advanced architectures such as **Gated Recurrent Units (GRUs)** and **Long Short-Term Memory (LSTM)** networks.

This report explains the evolution from RNNs to GRUs to LSTMs, the mathematical reason behind vanishing gradients, and the internal working of an LSTM cell with its gates.

2 Basic Recurrent Neural Network

A simple RNN processes a sequence one element at a time. At time step t, the hidden state h_t is computed as:

$$h_t = \tanh(W_h h_{t-1} + W_x x_t + b)$$

where:

- x_t is the input at time t
- h_{t-1} is the previous hidden state
- W_h and W_x are weight matrices
- b is a bias vector

The output y_t is typically computed as:

$$y_t = W_y h_t + c$$

This recurrence allows the network to model temporal dependencies. However, learning long-term dependencies is difficult due to gradient instability.

3 Vanishing Gradient Problem

Training RNNs uses backpropagation through time (BPTT). Gradients are propagated backward across many time steps. Mathematically, the gradient of the loss L with respect to an earlier hidden state involves repeated multiplication:

$$\frac{\partial L}{\partial h_{t-k}} = \left(\prod_{i=t-k+1}^{t} \frac{\partial h_i}{\partial h_{i-1}} \right) \frac{\partial L}{\partial h_t}$$

Each term $\frac{\partial h_i}{\partial h_{i-1}}$ contains derivatives of activation functions like tanh or σ, whose values lie in $(0, 1)$.

Repeated multiplication of values less than 1 causes the gradient magnitude to shrink exponentially as k increases. As a result:

- Early layers receive near-zero gradients
- Long-term dependencies are not learned
- Training becomes unstable

This issue motivated the design of gated architectures.

4 Gated Recurrent Unit (GRU)

GRUs introduce gating mechanisms to control information flow. A GRU has two gates:

- Update gate z_t
- Reset gate r_t

The equations are:

$$z_t = \sigma(W_z x_t + U_z h_{t-1})$$

$$r_t = \sigma(W_r x_t + U_r h_{t-1})$$

$$\tilde{h}_t = \tanh(W x_t + U(r_t \odot h_{t-1}))$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t$$

The update gate controls how much past information is retained, helping reduce gradient decay. GRUs are simpler than LSTMs and often perform similarly with fewer parameters.

5 Long Short-Term Memory (LSTM)

LSTMs further improve gradient flow by introducing a dedicated memory cell c_t and three gates:

- Forget gate
- Input gate
- Output gate

This structure allows gradients to flow almost unchanged across time steps.

5.1 Forget Gate

The forget gate decides what information to discard from the previous cell state:

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f)$$

Values close to 1 keep information, while values close to 0 erase it.

5.2 Input Gate

The input gate determines what new information should be stored:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$$

$$\tilde{c}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c)$$

5.3 Cell State Update

The cell state is updated as:

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$

This additive update is the key reason LSTMs avoid vanishing gradients.

5.4 Output Gate

The output gate controls what part of the cell state is exposed as hidden state:

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o)$$

$$h_t = o_t \odot \tanh(c_t)$$

6 Why LSTMs Solve Vanishing Gradients

Unlike standard RNNs, LSTMs use additive state updates instead of purely multiplicative ones. This allows gradients to propagate backward with minimal decay:

$$\frac{\partial c_t}{\partial c_{t-1}} = f_t$$

When $f_t \approx 1$, gradients flow almost unchanged across many time steps, enabling long-term memory learning.

7 Conclusion

The progression from RNNs to GRUs to LSTMs represents a steady improvement in handling long-term dependencies in sequential data. Standard RNNs are simple but unstable for long sequences. GRUs introduce gating to improve gradient flow with fewer parameters. LSTMs provide the most robust solution by explicitly separating memory storage and control mechanisms through multiple gates.

Due to their stability and expressive power, LSTMs remain widely used in sequence modeling tasks despite the rise of transformer-based architectures.