A Recurrent Neural Network (RNN) is a type of neural network designed to process sequential data by using feedback connections that allow information to persist across time steps, giving the network a form of memory.

RNN

Problem

- Vanishing Gradient
- Exploding Gradient

Solution

- GRU
- LSTM

Vanishing Gradient

The vanishing gradient problem occurs when gradients become extremely small during backpropagation in deep or recurrent networks, causing earlier layers to learn very slowly or not at all.

- **Happens mainly with sigmoid or tanh activations, where derivatives are small and repeatedly shrink during backpropagation.**
- **Deep networks and RNNs are most affected because gradients must pass through many layers or time steps.**
- **Leads to slow or stalled learning in early layers — network can't capture long-term dependencies.**
- **Causes models to favor short-term patterns, ignoring distant information.**
- **Solved or reduced using ReLU, LSTM/GRU gates, batch normalization, residual connections, and proper initialization.**

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

If each term

$$\left\| \frac{\partial h_k}{\partial h_{k-1}} \right\| = \alpha$$

then the gradient magnitude becomes:

$$\left\| \frac{\partial L}{\partial h_t} \right\| = \alpha^{(T-t)} \left\| \frac{\partial L}{\partial h_T} \right\|$$

Exploding Gradient

The exploding gradient problem occurs when gradients become excessively large during backpropagation, causing unstable learning where model weights grow uncontrollably and training diverges.

- **Most common in deep networks and RNNs where repeated multiplication of gradients causes them to blow up.**
- **Leads to numerical instability, causing loss values to become NaN or extremely large.**
- **Results in very large weight updates, making the model oscillate or fail to converge.**
- **Often caused by poor weight initialization or very long sequences in recurrent networks.**
- **Common solutions include gradient clipping, proper initialization (Xavier/He), using LSTM/GRU, and normalization techniques.**

- If **α < 1**, then

$$\alpha^{(T-t)} \to 0 \quad \Rightarrow \quad \text{Vanishing gradient}$$

- If **α > 1**, then

$$\alpha^{(T-t)} \to \infty \quad \Rightarrow \quad \text{Exploding gradient}$$

- h_k is the **hidden state** of an RNN at time step k.
- It represents the memory or internal representation at that moment.

Example in RNN:

$$h_k = f(W_h h_{k-1} + W_x x_k)$$

- L is the **loss function** (e.g., cross-entropy, MSE).
- During backpropagation, we compute how L changes with respect to each hidden state h_t.

Example:

$$L = \text{CrossEntropy}(y_{\text{true}}, y_{\text{pred}})$$

A Comparative Analysis of GRU & LSTM

GRU

A GRU (Gated Recurrent Unit) is a type of recurrent neural network that uses gates to control information flow, allowing it to learn long-term dependencies while avoiding vanishing gradients, similar to LSTM but with a simpler structure.

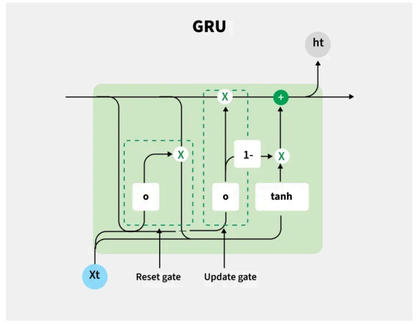
1. **Has two gates**:
   - **Update gate (z_t)** → decides how much of the past to keep
   - **Reset gate (r_t)** → decides how much of the past to forget
2. **No separate cell state**; the hidden state itself carries memory (simpler than LSTM).
3. **Faster and more efficient** than LSTM because it has fewer parameters.
4. Handles **long-term dependencies** better than simple RNNs.
5. Works well for **text, speech, time series**, and any sequential task requiring memory.

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

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

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

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



LSTM

A Long Short-Term Memory (LSTM) network is a type of recurrent neural network designed to learn long-term dependencies using a memory cell and three gates that control information flow, preventing vanishing and exploding gradients.

1. **Three gates**:
   - **Forget gate** f_t → decides what to erase from memory
   - **Input gate** i_t → decides what new information to store
   - **Output gate** o_t → controls what part of memory becomes output
2. **Has a separate cell state** C_t that carries long-term memory across many time steps.
3. Designed to **solve vanishing gradient problem**, enabling learning of long sequences.
4. More powerful but **heavier and slower** than GRU (more parameters).
5. Used in **NLP, speech recognition, time-series forecasting**, and tasks requiring long-range context.

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

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

$$\tilde{C}_t = \tanh(W_C x_t + U_C h_{t-1})$$

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

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

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