Recurrent Neural Networks (RNNs)

Muhammad Atef Elkaffas

Alexandria ITI AI 45

Recurrent Neural Networks (RNNs) process a sequence $x_1, x_2, \ldots, x_T$ by maintaining a hidden "memory" state h_t at each time step t. The recurrence equations are

$$h_t \;=\; f\big(W\,x_t \;+\; U\,h_{t-1} \;+\; b\big), \qquad y_t \;=\; g\big(V\,h_t \;+\; c\big),$$

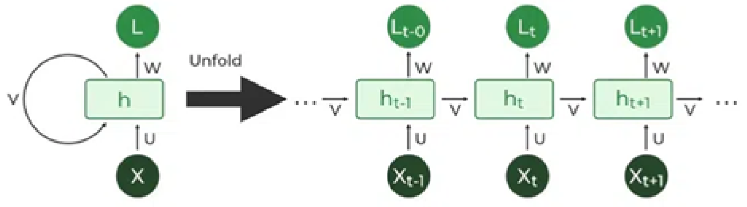
where:

- x_t is the input at time t,
- $h_t \in R^H$ is the hidden state at time t,
- y_t is the output at time t,
- W, U, V are learned weight matrices,
- b, c are learned bias vectors,
- f, g are activation functions (e.g. tanh, ReLU, softmax).

When we *unfold* the RNN through time, back-propagation requires computing gradients

$$\frac{\partial \mathcal{L}}{\partial h_{t-k}} \;=\; \prod_{i=1}^{k} \Big[\operatorname{diag}\big(f'(W x_{t-i} + U h_{t-i-1} + b)\big)\,U\Big] \;\times\; \frac{\partial \mathcal{L}}{\partial h_t}\,.$$

If the spectral radius of U is less than 1, these products *shrink* exponentially (the **vanishing-gradient** problem), causing the network to "forget" long-range dependencies; if it is greater than 1, they *grow* exponentially (the **exploding-gradient** problem), leading to unstable training. Vanilla RNNs lack any explicit gating mechanism to control information flow, so they cannot selectively forget irrelevant information or protect important signals.



Simplified Analogy

Imagine an RNN as a student trying to take perfect notes of a very long lecture. In theory, they jot down every word and carry those notes forward, but in practice they either smudge out early lines (*vanishing gradients*) or scribble so much that the page becomes illegible (*exploding gradients*). Without any way to decide which parts of the lecture to erase or highlight (no "forget" gates), the student ends up either drowning in faded details or overwhelmed by noise—so they struggle to remember the important lessons from the beginning by the time the lecture ends.

RNN Architectures

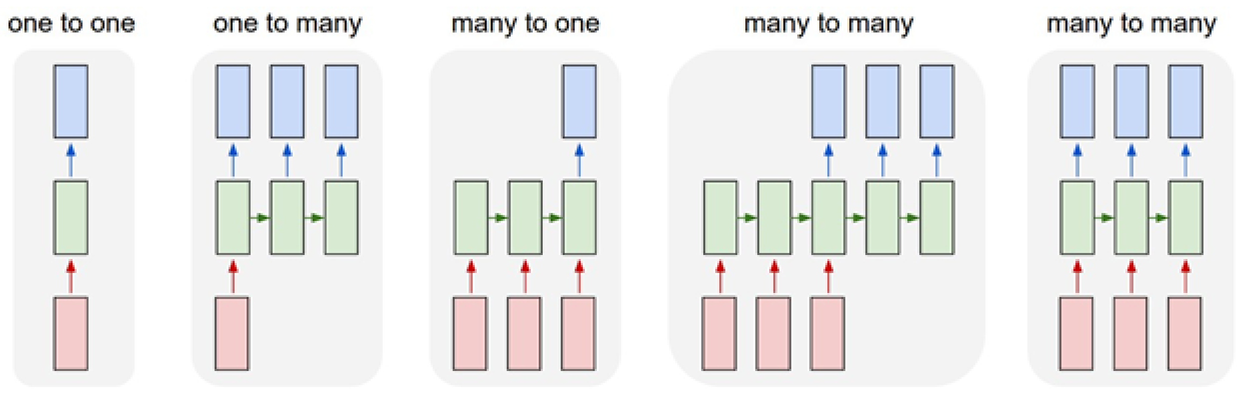
- *One-to-one:* standard feed-forward mapping.
- *One-to-many:* e.g. image caption generation.
- *Many-to-one:* e.g. sentiment classification from text.
- *Many-to-many (aligned):* e.g. sequence tagging or time series forecasting.
- *Many-to-many (transduction):* e.g. machine translation with differing input/output lengths.

Strengths of RNNs

- Can model *time dependencies* and *context* across sequential data.
- Handle *variable-length* inputs (e.g. sentences, audio) naturally.

Limitations of RNNs

- **Vanishing gradients:** makes it difficult to learn long-term dependencies.
- **Exploding gradients:** can cause instability during training.
- No built-in *forget* gate—cannot selectively preserve or discard information.
- Inherently *sequential* computation → slower on long sequences compared to parallelizable models (e.g. Transformers).


one to one
one to many
many to one
many to many
many to many