Back-propagation Through Time (BPTT)

- A variant of BP specifically designed for **recurrent neural networks (RNNs)**, which handle **sequential** data.
- Since RNNs process inputs sequentially over time, BPTT unrolls the network across multiple time steps and then applies standard back-propagation to compute gradients (essentially the error due to the weights at each connection layer).
- One way to think about this is that as each word is encountered, it keeps a copy of the new state of the network that results from that word combining with the prior state of the network, accumulating as many such states as there are words in the sequence, and at each new word, it back-propagates through/across all the prior states.
- The number of time steps that **Back-propagation Through Time (BPTT)** unrolls the network depends on the implementation:
-
    1. **Full BPTT**
        - The network is unrolled **for all time steps** in the sequence.
        - This captures long-term dependencies but is computationally expensive and memory-intensive.
    2. **Truncated BPTT**
        - The network is unrolled for a fixed number of **k** time steps (e.g., 5, 10, or 20).
        - This makes training more efficient and helps prevent exploding/vanishing gradients.
        - Commonly used in practice because full BPTT is impractical for long sequences.

In **truncated BPTT**, gradients are typically only back-propagated for k time steps, even if dependencies extend further back in time. The choice of k is a trade-off between efficiency and capturing long-term dependencies.