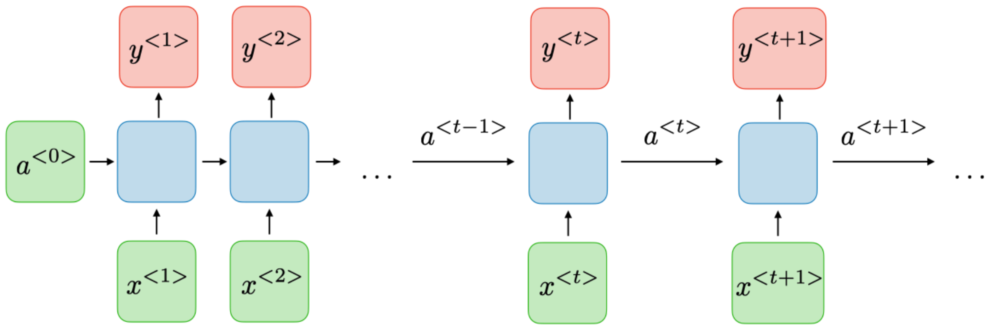
Recurrent Neural Network

Recurrent neural networks or RNNs are a family of neural networks for processing sequential data.
A recurrent neural network is a neural network that is specialized for processing a sequence of values $x^{(1)}$, …. $x^{(n)}$. RNN were first developed in the 1980s where the idea was sharing parameters across different parts of a model. Parameter sharing makes it possible to extend and apply the model to examples of different forms (different lengths, here) and generalize across them. RNNs were the standard suggestion for working with sequential data before the advent of attention models. RNNs have a Memory which stores all information about the calculations. It employs the same settings for each input since it produces the same outcome by performing the same task on all inputs or hidden layers.



An RNN processes the sequence one element at a time, in the so-called time steps.

For each timestep t, the activation $a^{<t>}$ and the output $y^{<t>}$ are expressed as follows:

$$\boxed{a^{<t>} = g_1(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a)} \quad \text{and} \quad \boxed{y^{<t>} = g_2(W_{ya}a^{<t>} + b_y)}$$

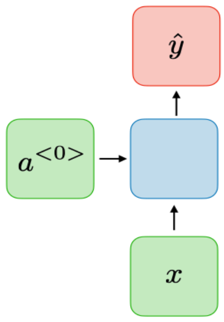
where $W_{ax}, W_{aa}, W_{ya}, b_a, b_y$ are coefficients that are shared temporally and g_1, g_2 activation functions.

RNN models are mostly used in the fields of natural language processing and speech recognition. The different applications are summed up in the table below:

Type of RNN:

1. **One-to-One**

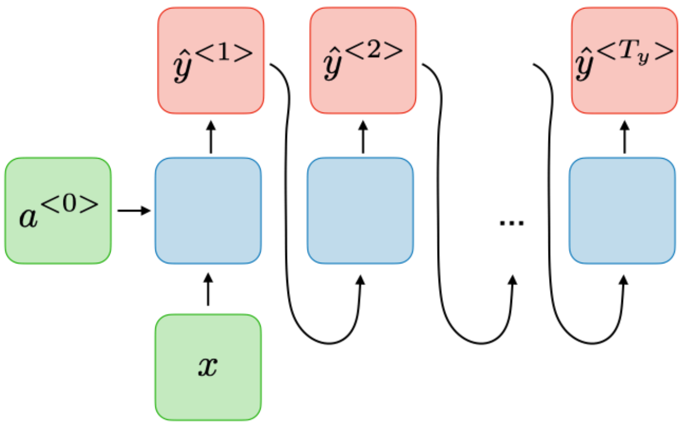
$$T_x = T_y = 1$$



Example: Traditional neural network

2. **One-to-many**

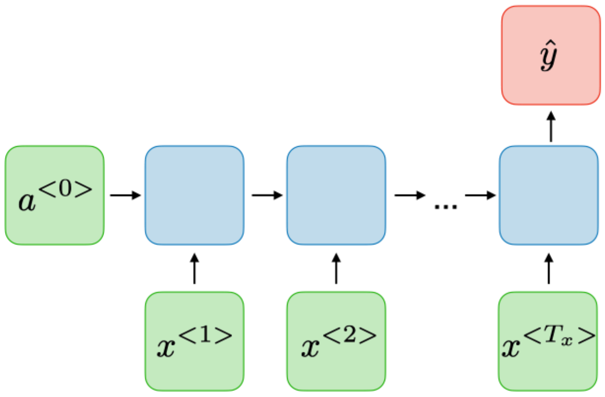
$$T_x = 1, T_y > 1$$



Example: Music generation

3. Many-to-one

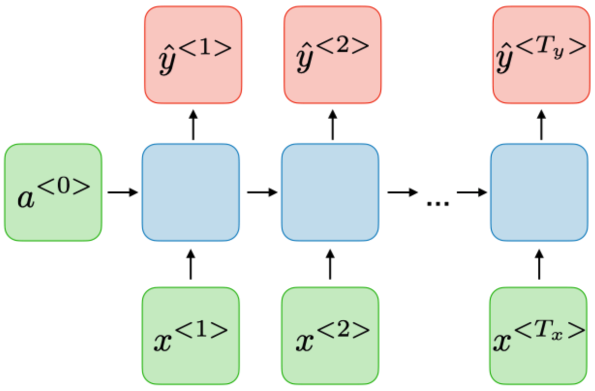
$T_x > 1, T_y = 1$



Example: Sentiment classification

4. Many-to-many

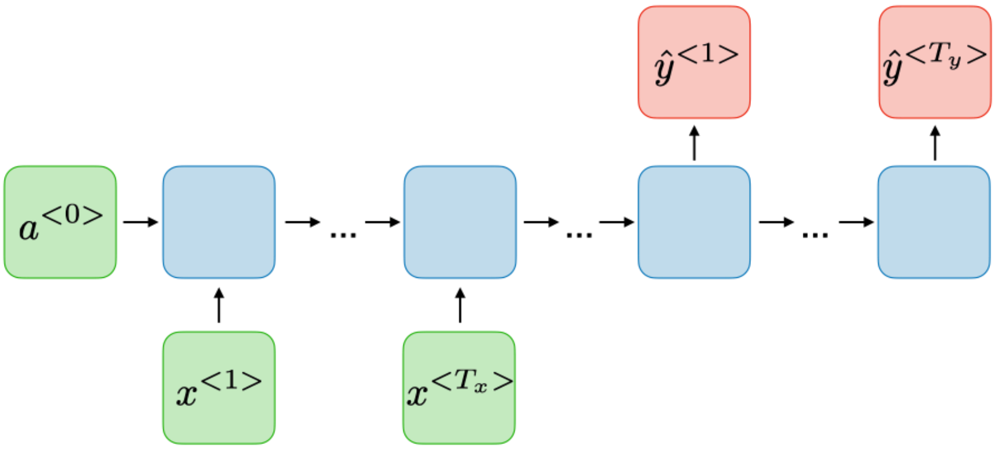
$T_x = T_y$



Example: Name entity recognition

5. Many-to-Many

$T_x \neq T_y$

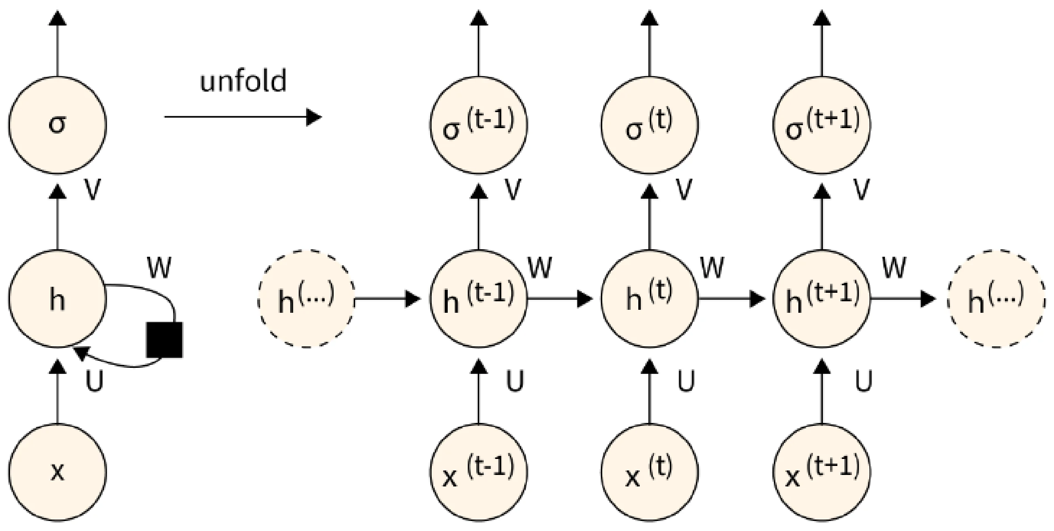


Example: Machine translation

Summary of types:

Type of RNN	One-to-one	One-to-many	Many-to-one	Many-to-many	Many-to-many
Number of I/O	Tx = Ty = 1	Tx = 1 and Ty > 1	Tx > 1 and Ty = 1	Tx = Ty	Tx ≠ Ty
Application	Traditional NN	Music generation	Sentiment classification	Name entity recognition	Machine translation

The Architecture of a Traditional RNN



The above diagram displays an RNN neural network in notation on the left and an RNN becoming unrolled (or unfolded) into a complete network on the right. Unrolling refers to writing out the network for the entire sequence. The network will be unrolled into a 3-layer neural network, one layer for each word, for instance, if the sequence we are interested in is a sentence of three words.

Input:- At time step t, $x^{(t)}$ is used as the network's input.

Weights:- The Recurrent neural network in deep learning has hidden input connections that are parameterized by a weight matrix U, connections for hidden-to-hidden recurrent connections that are parameterized by a weight matrix W, and connections for hidden-to-output connections that are parameterized by a weight matrix V. These weights (U,V,W) are shared over time.

Input to Output layer propagation:

$t = 1$ to $t = \tau$, we apply the following update equations:

$$\begin{aligned} \boldsymbol{a}^{(t)} &= \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)} \\ \boldsymbol{h}^{(t)} &= \tanh(\boldsymbol{a}^{(t)}) \\ \boldsymbol{o}^{(t)} &= \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)} \\ \hat{\boldsymbol{y}}^{(t)} &= \operatorname{softmax}(\boldsymbol{o}^{(t)}) \end{aligned}$$

Advantages of RNNs:

- Handle sequential data effectively, including text, speech, and time series.
- Process inputs of any length, unlike feedforward neural networks.
- Share weights across time steps, enhancing training efficiency.

Disadvantages of RNNs:

- Prone to vanishing and exploding gradient problems, hindering learning.
- Training can be challenging, especially for long sequences.
- Computationally slower than other neural network architectures.