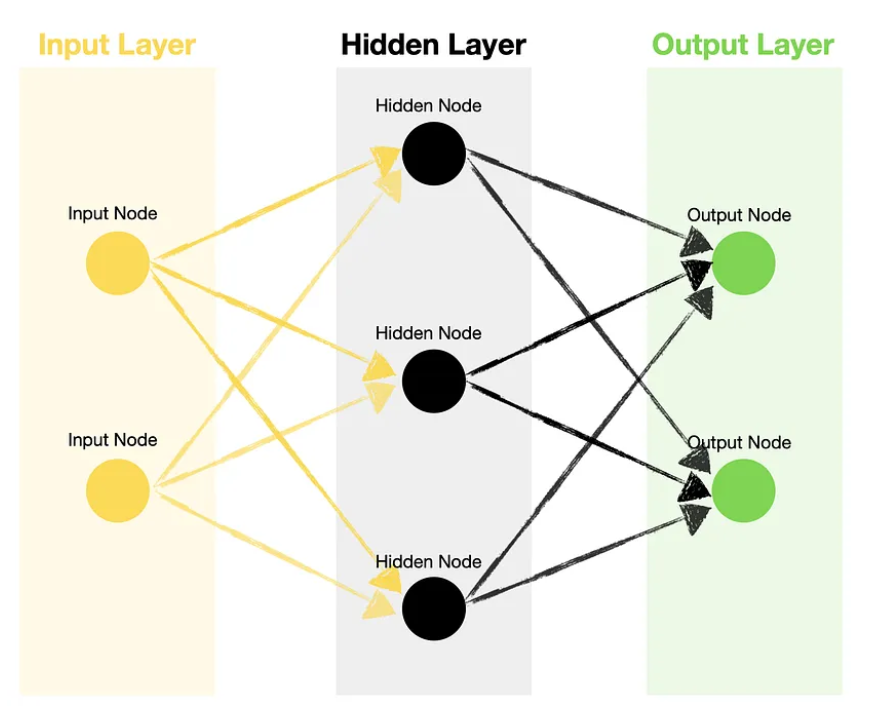
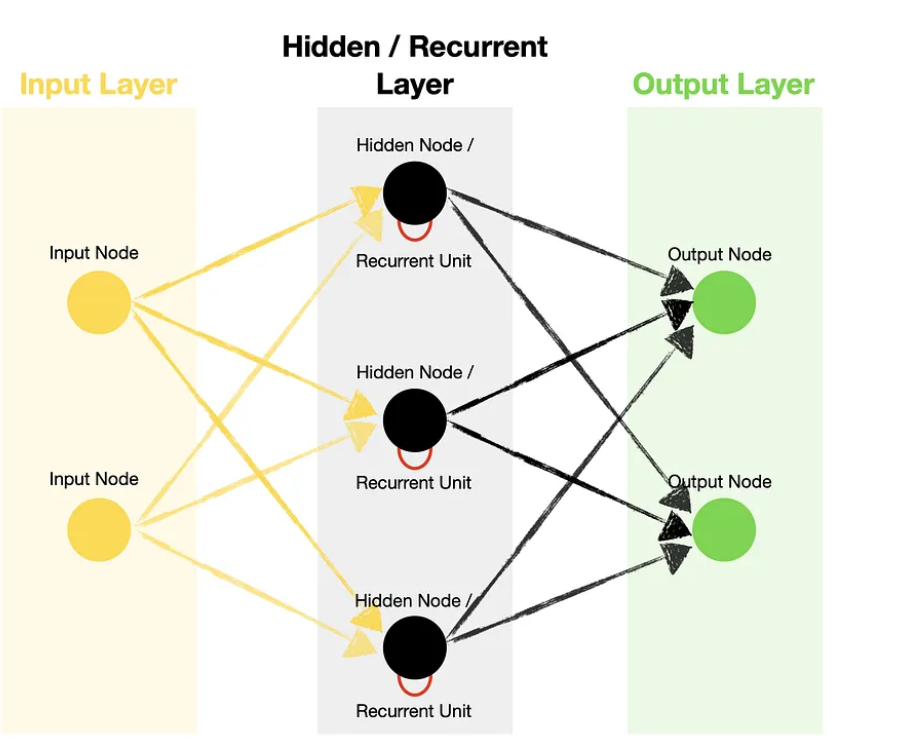
**The structure of Recurrent Neural Networks (RNNs)**

First, let’s remind ourselves what a typical Feed Forward Neural Network looks like. Note that it can contain any number of input nodes, hidden nodes, and output nodes. The below 2–3–2 structure is purely for illustration:



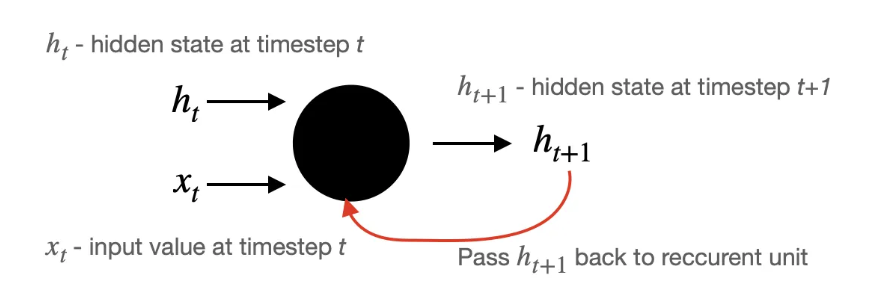
Next, if we look at RNN, we notice a slight difference. The hidden units inside RNN have a built-in feedback loop, enabling the information to be passed back to the same node multiple times. These hidden units are commonly called **recurrent units**:



A recurrent unit processes information for a predefined number of **timesteps**, each time passing a hidden state and an input for that specific timestep through an activation function.

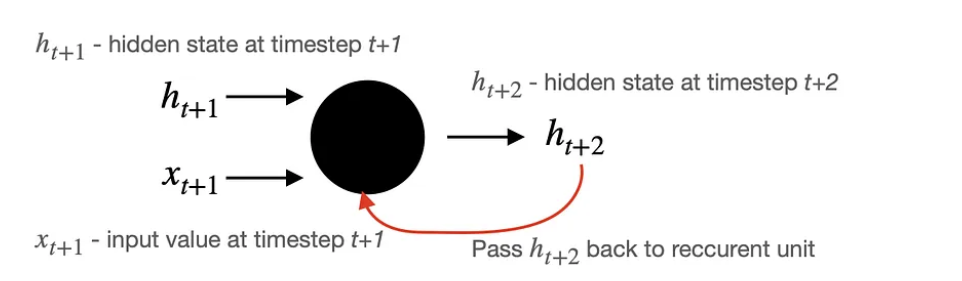
**The hidden state of an RNN can capture historical information of the sequence up to the current time step.**

**Timestep**— single processing of the inputs through the recurrent unit. E.g., if you have only one timestep, then your inputs will only be processed once (equivalent to a regular hidden node). If you have seven timesteps, then your inputs will be processed seven times.



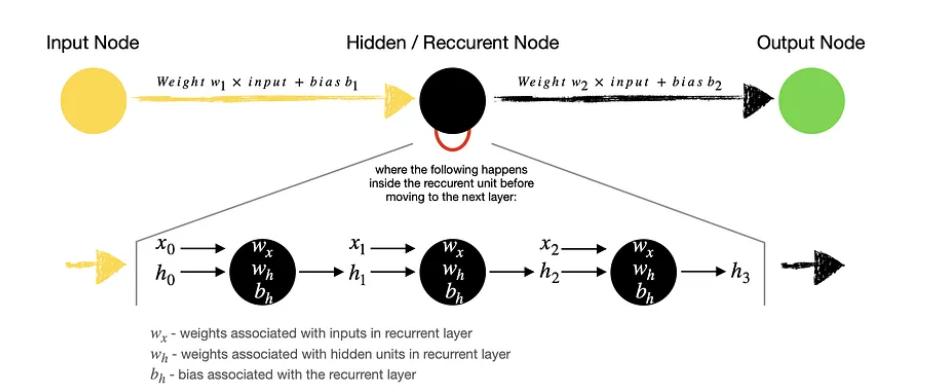
Remember that the hidden state captures all the previous inputs of the sequence Xt-1, Xt-2,…,Xt-n **up to the current time step (t).**

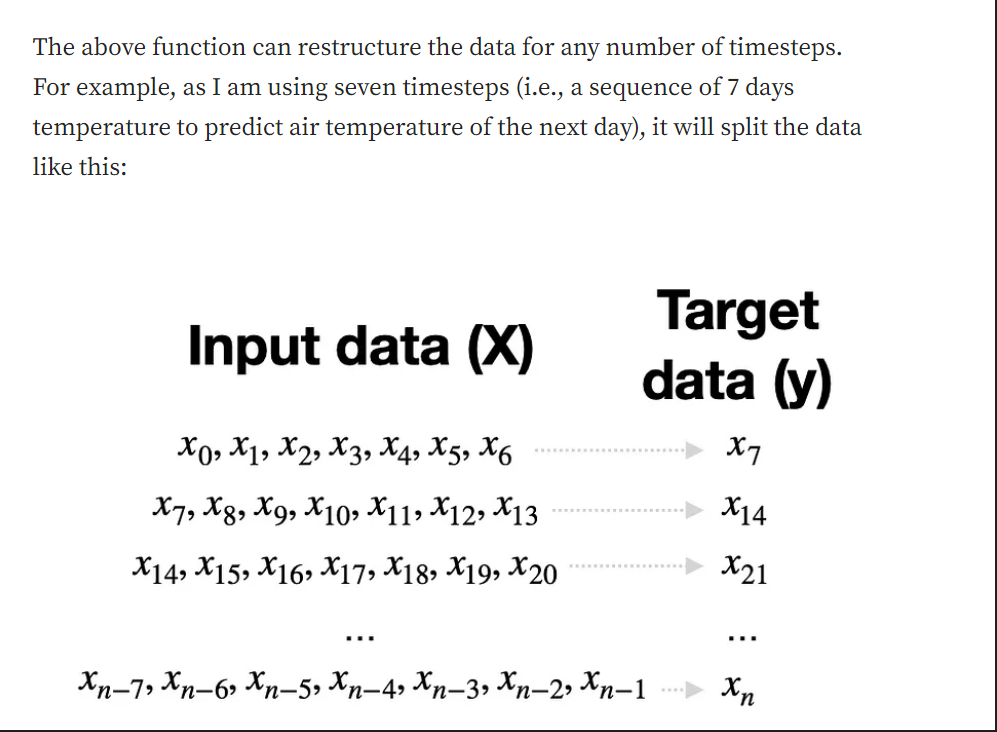
Note that at the initial timestep, the hidden state h0 is initialized to 0. Next, the output **(a hidden state h at t+1)** is passed back to a recurrent unit and processed again together with the following input:



The process repeats until the specified number of timesteps is reached.

Let’s tie all of it together and see what a simple RNN with one input, one hidden node (containing three timesteps), and one output would look like.





Assume you want to predict tomorrow’s air temperature based on the sequence of air temperatures from the last three days. Then:

* **Inputs** — while you may have only one input node, you would have to pass sequences of three numbers as your input because that is what’s required by the recurrent layer, i.e. [x0, x1, x2], …, [x\_{n-2}, x\_{n-1}, x\_{n}].
* **Recurrent layer** — in a typical feed-forward neural network, the hidden node would have two parameters: weight and bias. However, a recurrent layer has **three parameters** to optimize: weight for the input, weight for the hidden unit, and bias. Note that it would still be three parameters even if you had ten timesteps.
* **Training** — a typical feed-forward neural network is trained using a backpropagation algorithm. Meanwhile, training an RNN uses a slightly modified version of backpropagation, which includes the unfolding in time to train the weights of the network. The algorithm is based on computing the gradient vector and is called **backpropagation in time** or **BPTT** for short.

**Conceptually, BPTT works by unrolling all input timesteps. Each timestep has one input timestep, one copy of the network, and one output. Errors are then calculated and accumulated for each timestep. The network is rolled back up and the weights are updated.**