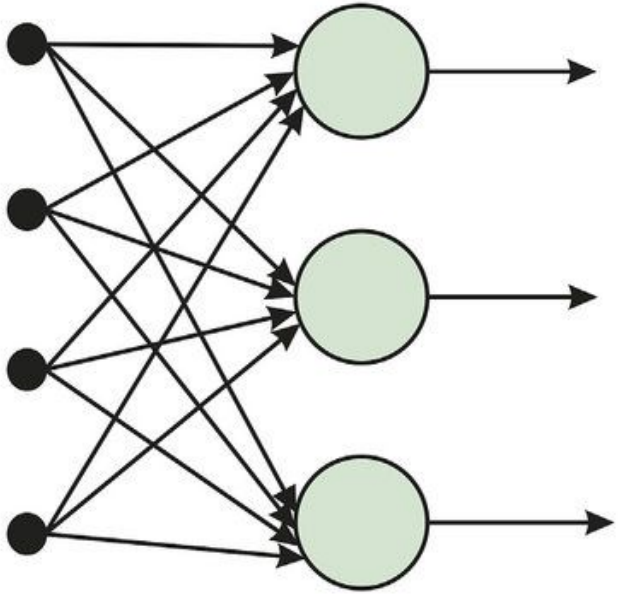
Recurrent Neural Networks (RNNs)

Recurrent Neural Networks (RNN)

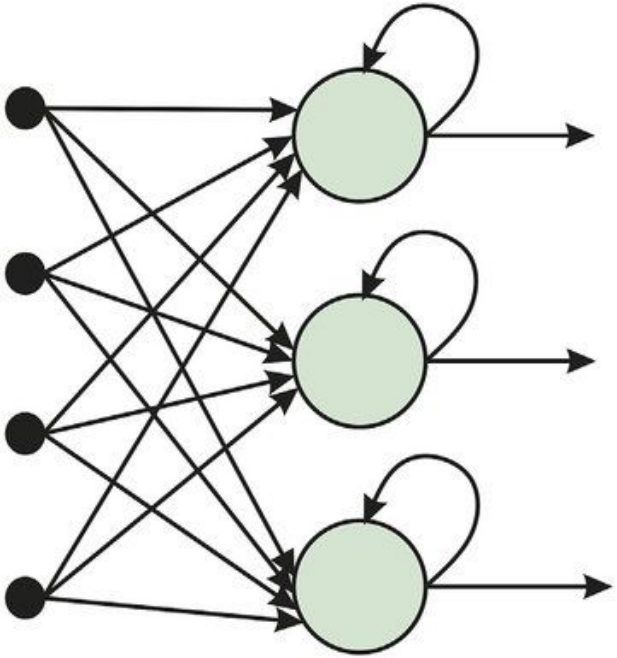
- Used for handling sequences of data.
  - Time series.
  - Text.
  - Speech.

Feed-Forward NN

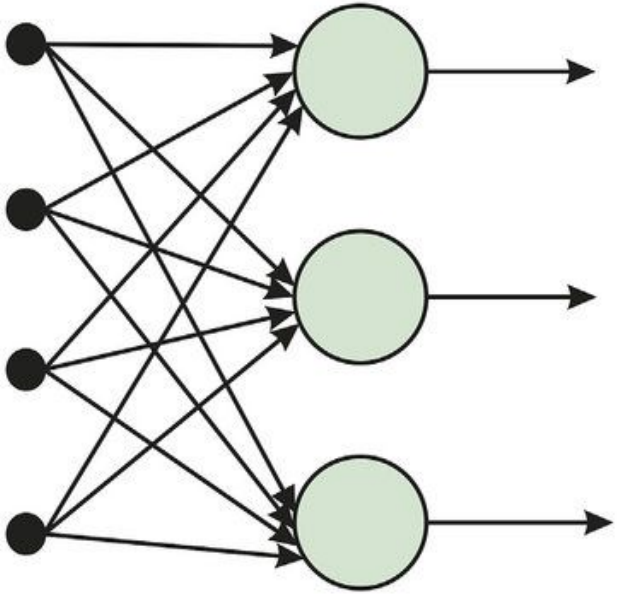


(b) Feed-Forward Neural Network

Recurrent NN

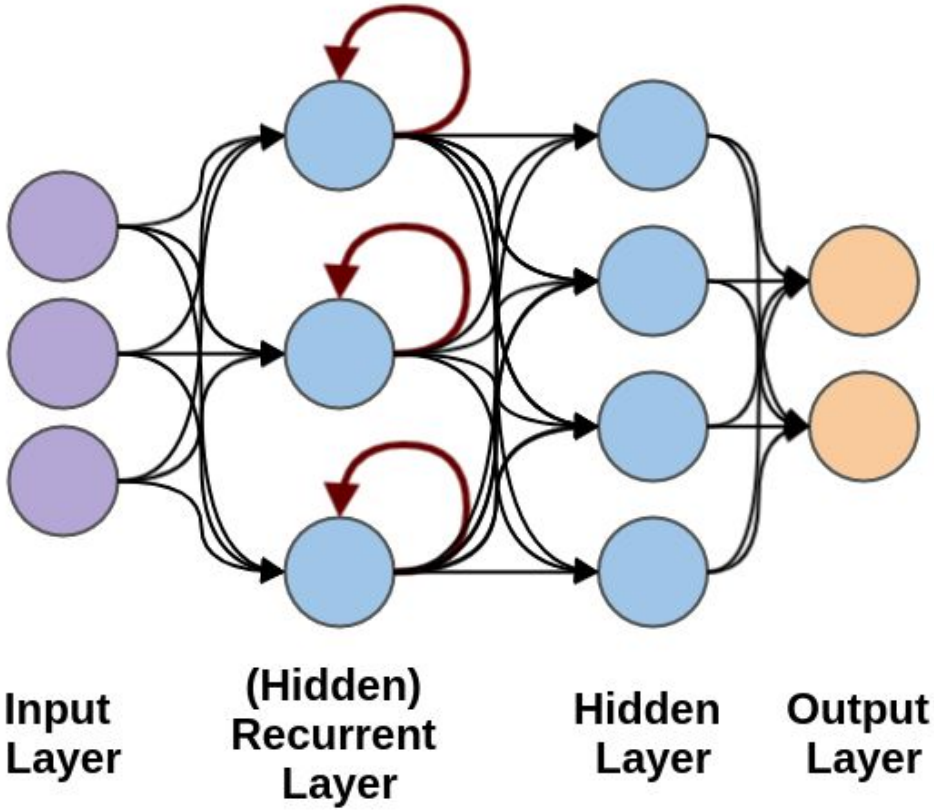


(a) Recurrent Neural Network

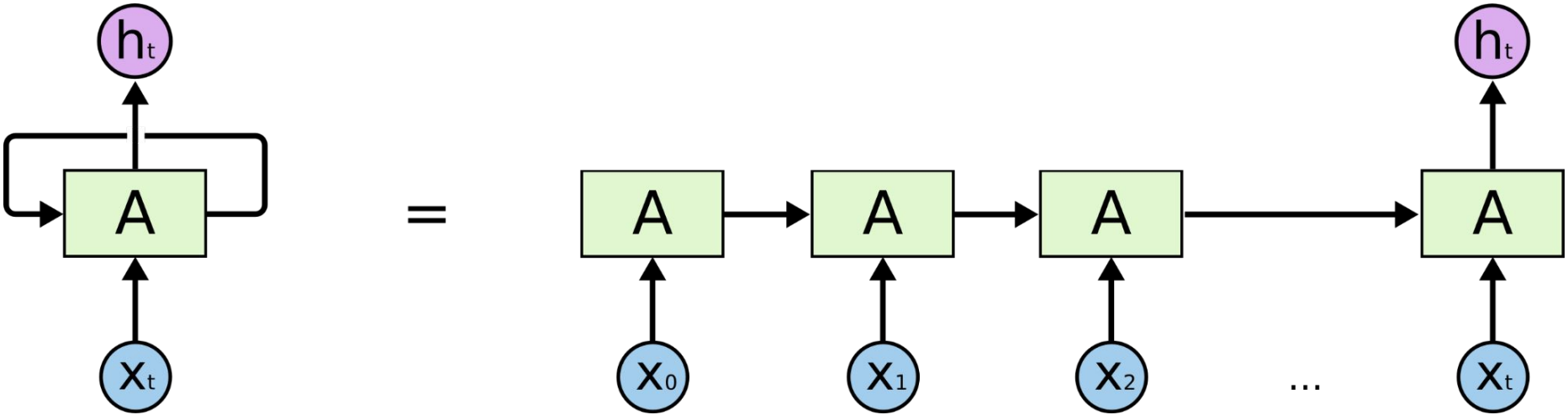


(b) Feed-Forward Neural Network

Recurrent NN



Recurrent NN

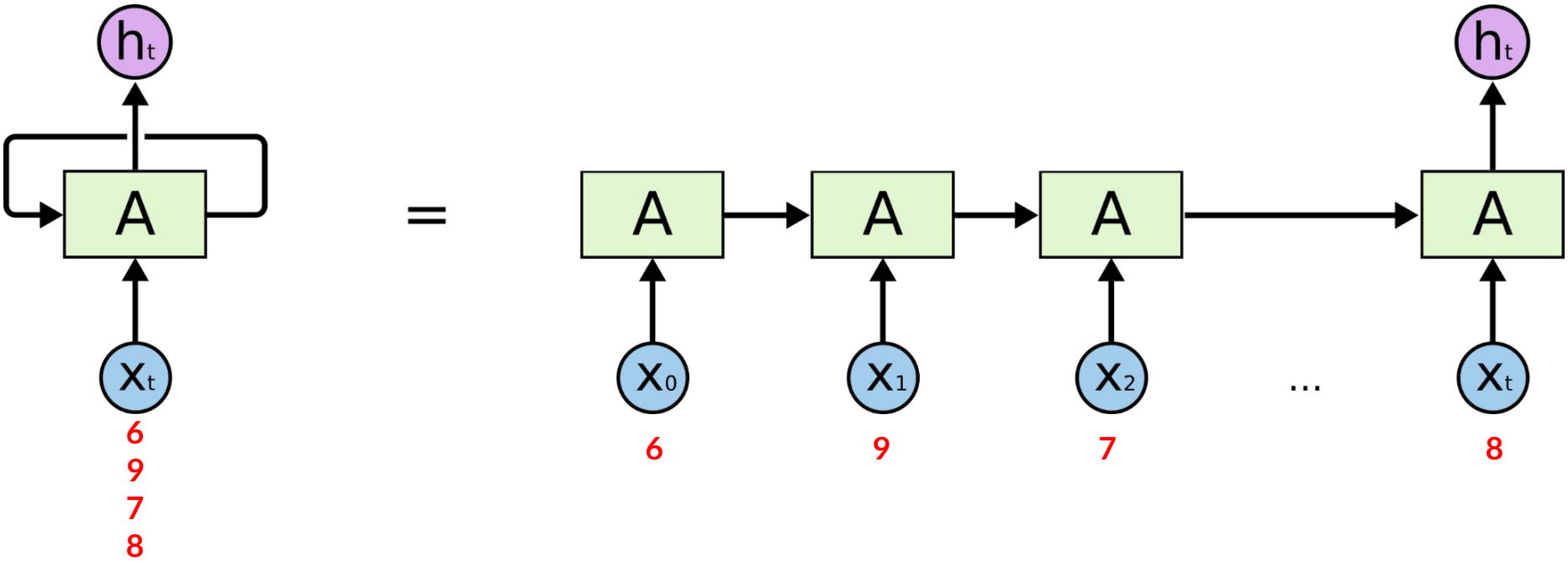


Recurrent NN

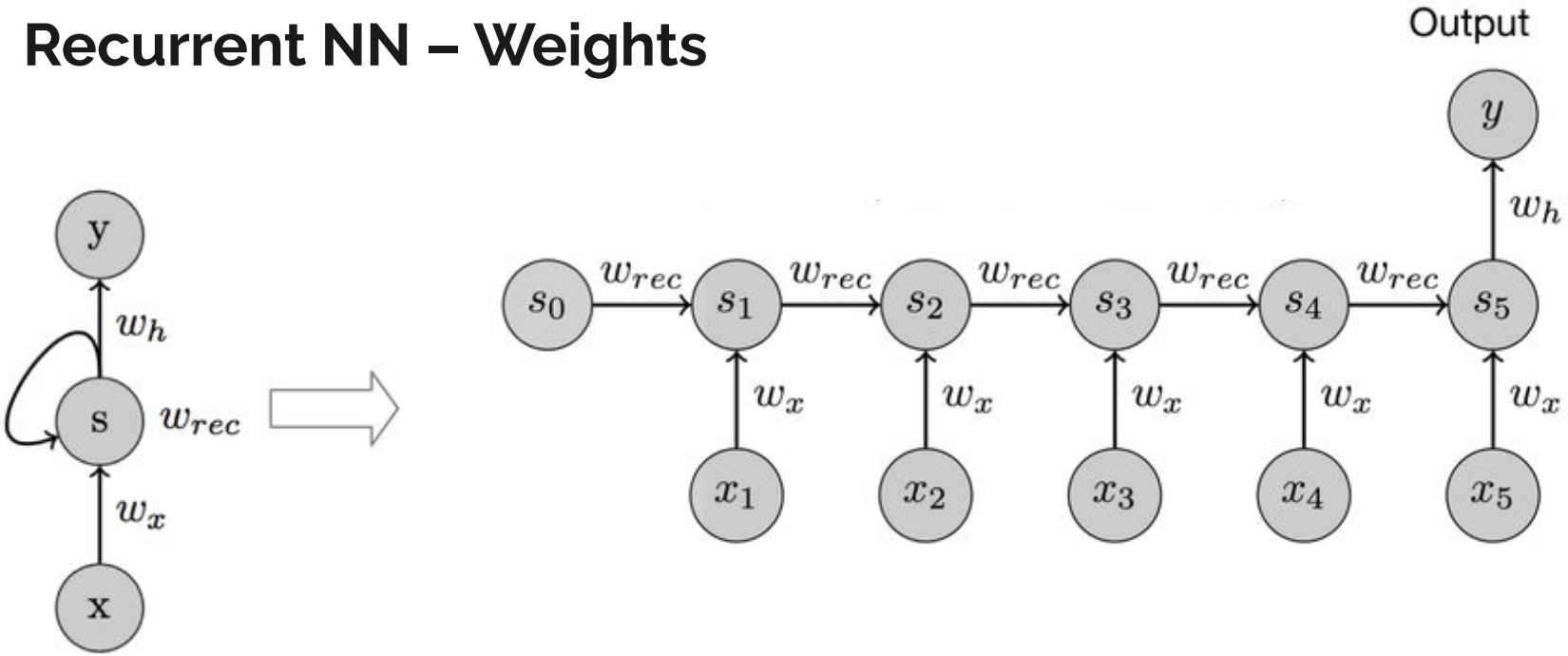
Sequence size = 4

	Today's temperature	Tomorrow's temperature
25/11/22	6	9
26/11/22	9	7
27/11/22	7	8
28/11/22	8	NaN

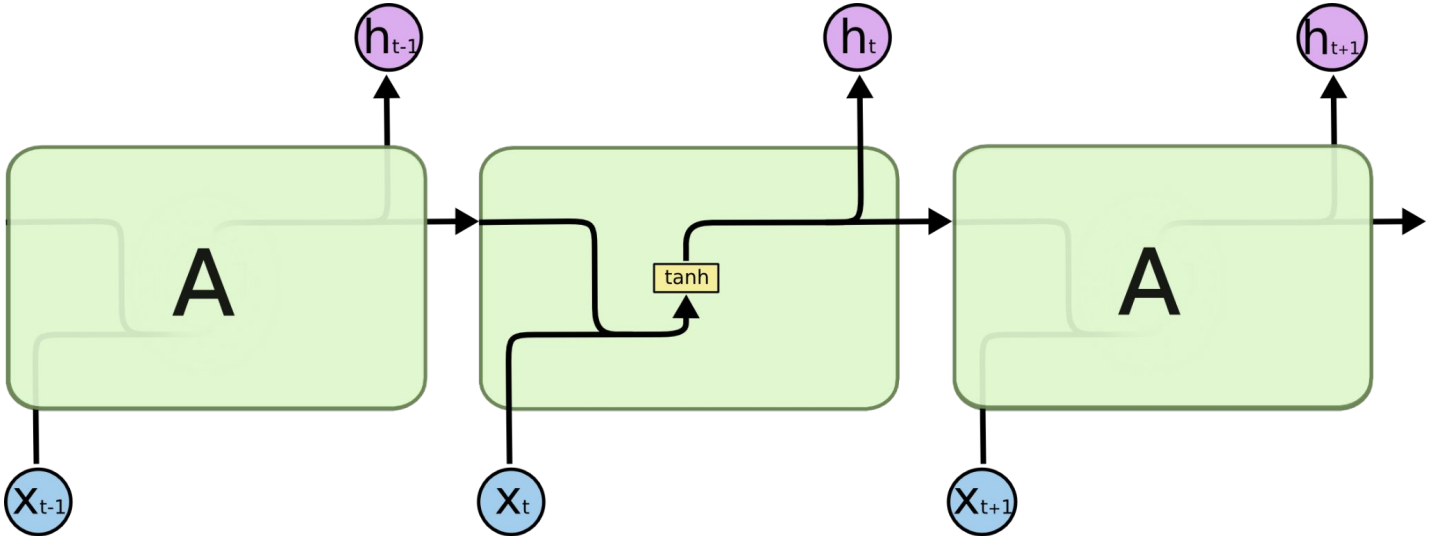
Recurrent NN – One output every t timesteps



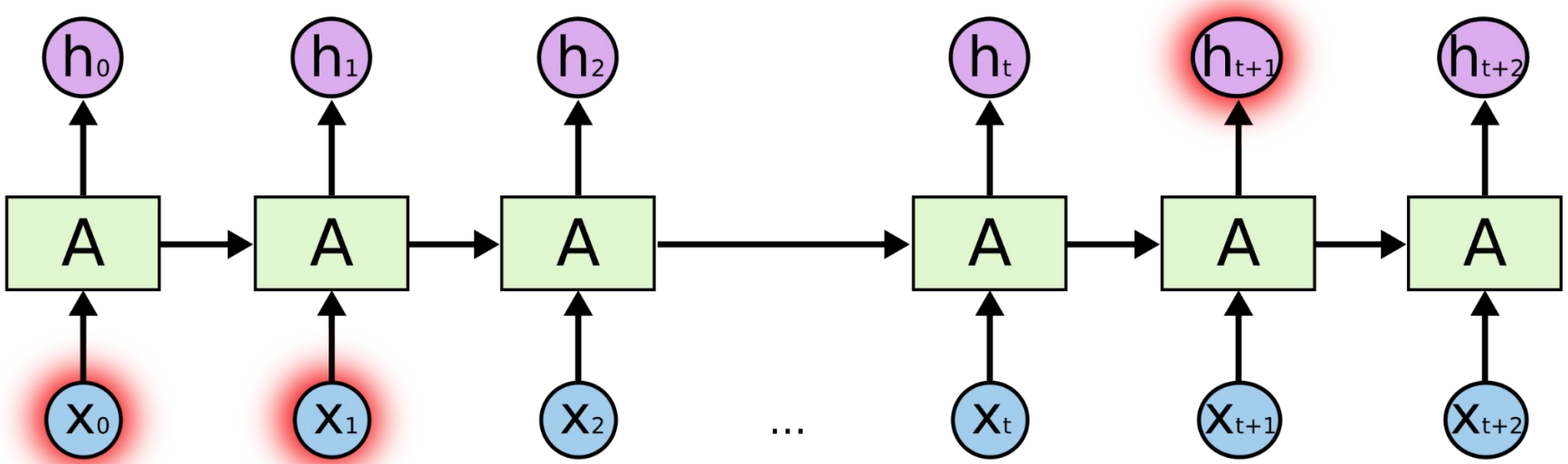
Recurrent NN – Weights



Recurrent NN – One output per timestep



Recurrent NN



Recurrent NN

- Use Backpropagation Through Time (BPTT).

Recurrent NN

- Use Backpropagation Through Time (BPTT).
- Pros:
    - It can model non-linear sequential relationships.

Recurrent NN

- Use Backpropagation Through Time (BPTT).
- Pros:
  - It can model non-linear sequential relationships.
- Cons:
  - Exploding gradients (when w > 1).
  - Vanishing gradients (when w < 1).

  } Use `tanh` activation function.

  - Not well suited for long temporal data.

Long Short-Term Memory (LSTM)

- It is a RNN.
- Introduced in 1997.
- Deals with vanishing gradients.
- Can “store” information for long periods.

Long Short-Term Memory (LSTM)

- 3 gates regulate the flow of information through time:
  - **Forget gate**: Decides what information to discard or to keep in “long-term memory”.

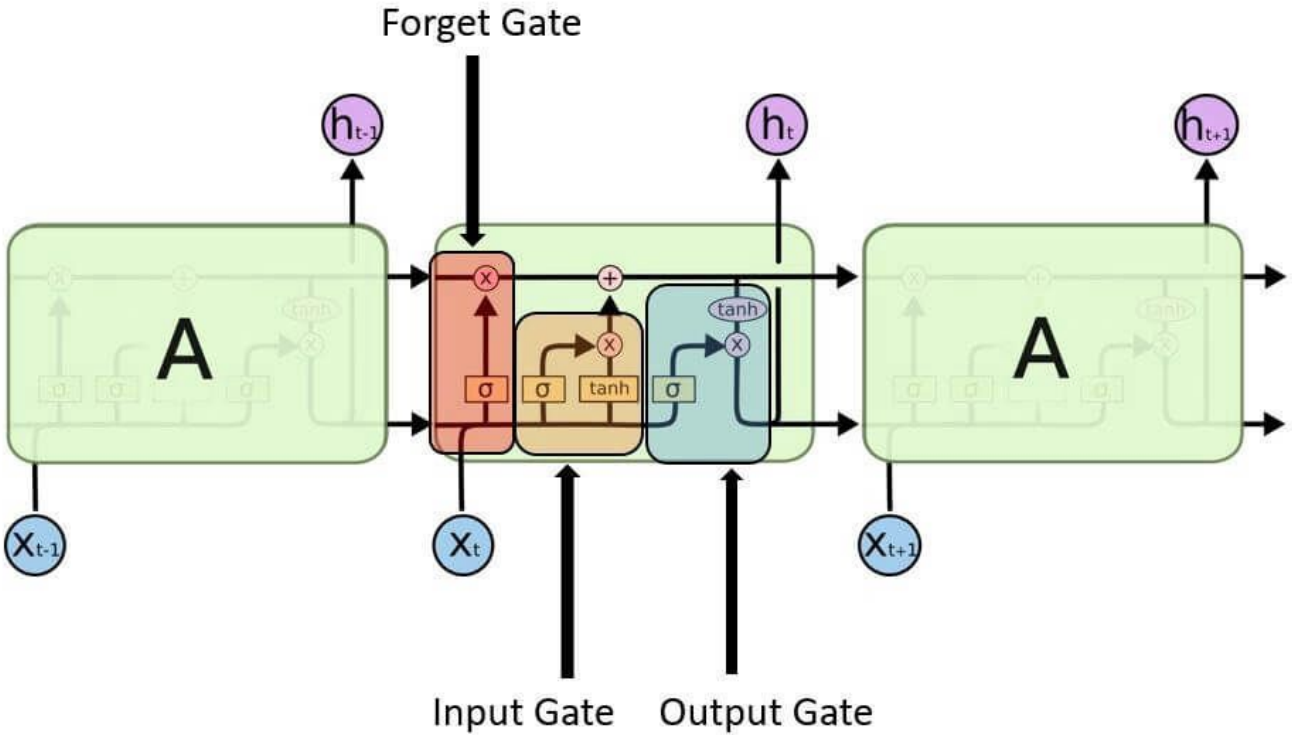
Long Short-Term Memory (LSTM)

- 3 gates regulate the flow of information through time:
  - **Forget gate**: Decides what information to discard or to keep in “long-term memory”.
  - **Input gate**: Decides what new information to store.

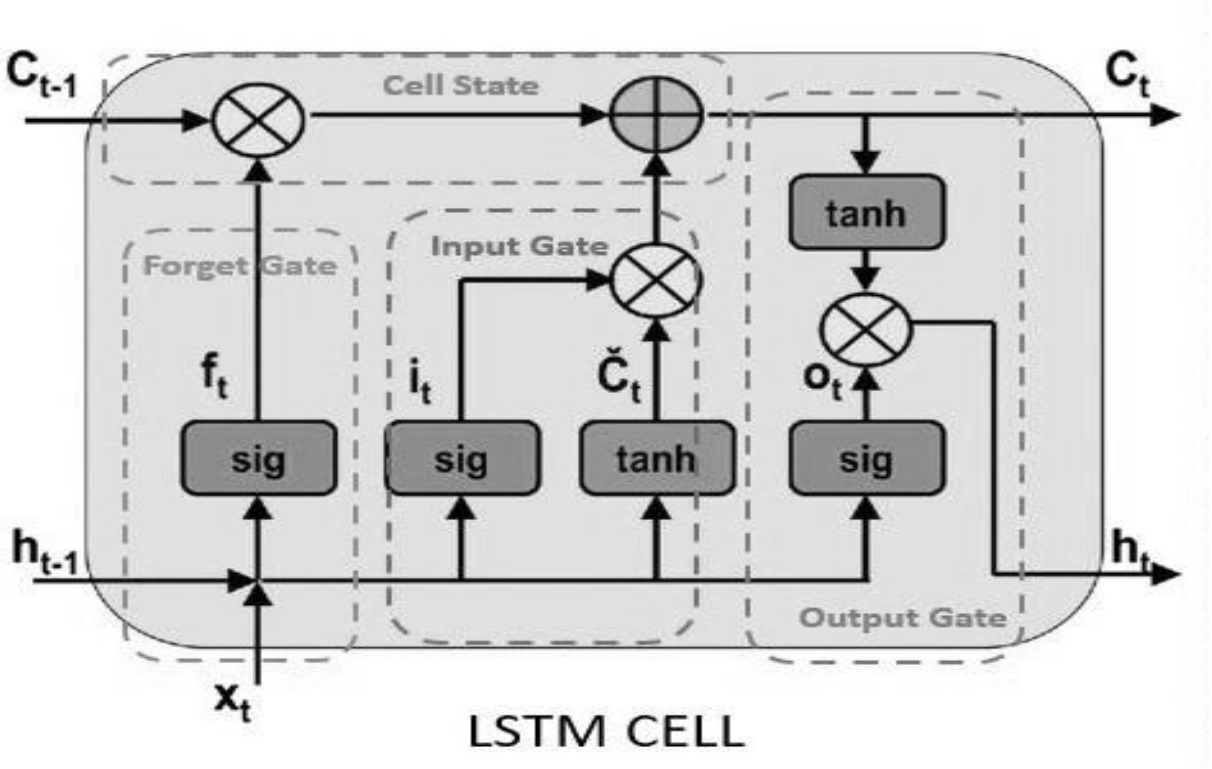
Long Short-Term Memory (LSTM)

- 3 gates regulate the flow of information through time:
  - **Forget gate**: Decides what information to discard or to keep in “long-term memory”.
  - **Input gate**: Decides what new information to store.
  - **Output gate**: Decides what information to discard or to keep in “short-term memory”.

LSTM Cell


Forget Gate
h_{t-1}
h_t
h_{t+1}
A
A
σ
σ
tanh
σ
tanh
X_{t-1}
X_t
X_{t+1}
Input Gate
Output Gate

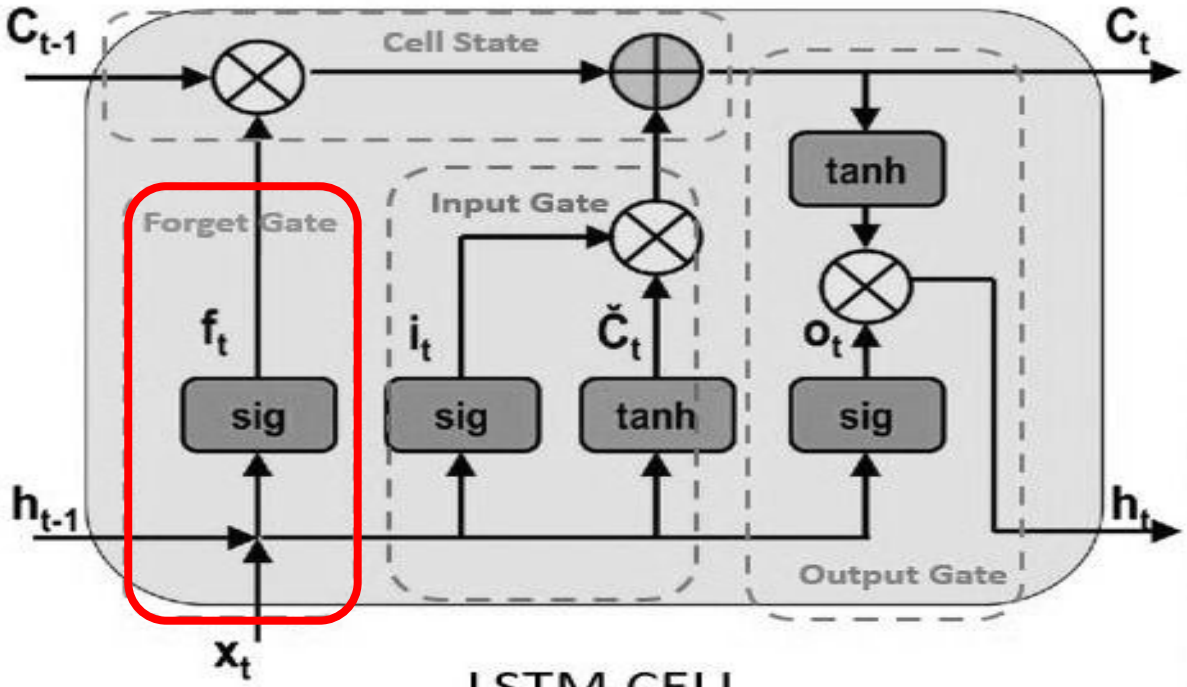
Long Short-Term Memory (LSTM)



LSTM

- Cell state (*C*) and hidden state (*h*) are vectors of *N* units (hyperparam.).
- These vectors are an abstraction of the “memory”.
- The `sigmoid` and `tanh` are applied element-wise to each vector element.
- Gates’ internal structure:
  - 1 NN for the Forget Gate.
  - 2 NN for the Input Gate (Input + Candidate).
  - 1 NN for the Output Gate.

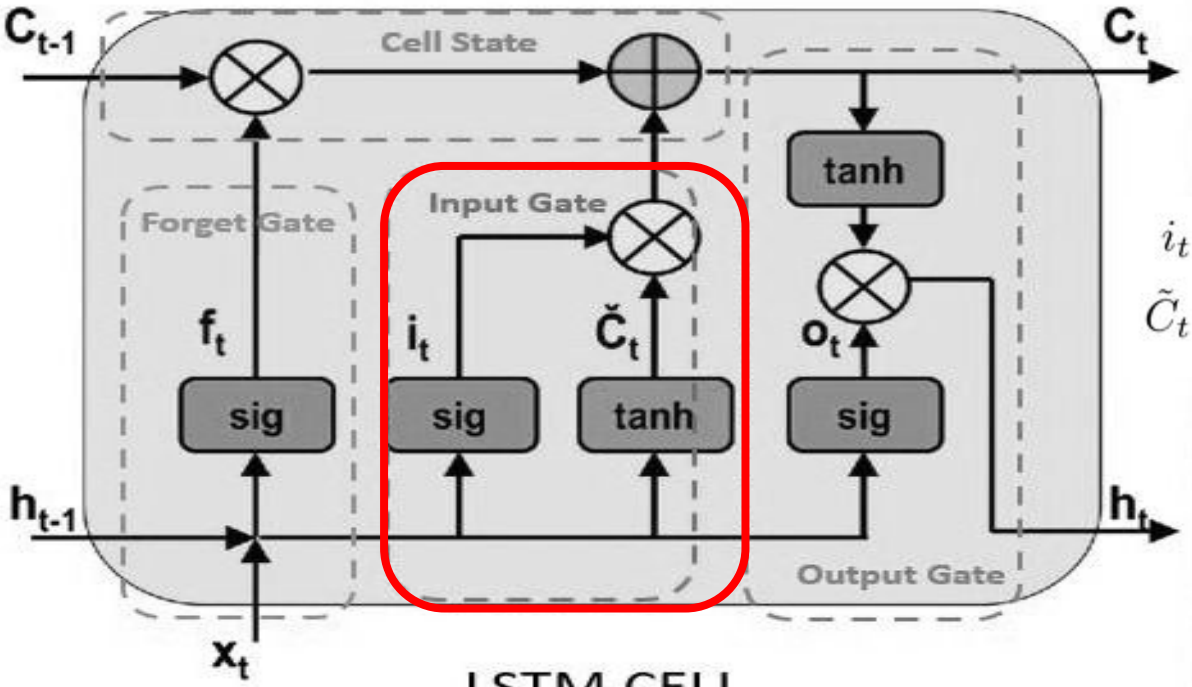
LSTM



LSTM CELL

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

LSTM

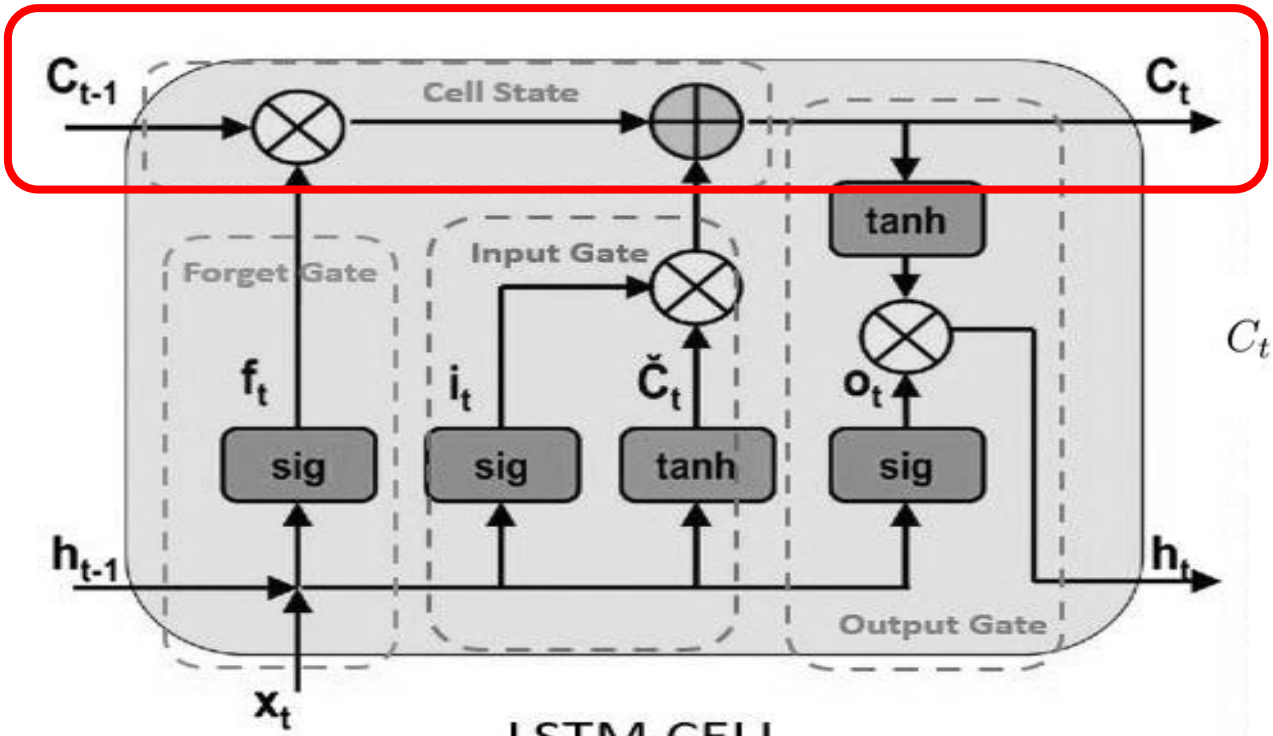


LSTM CELL

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

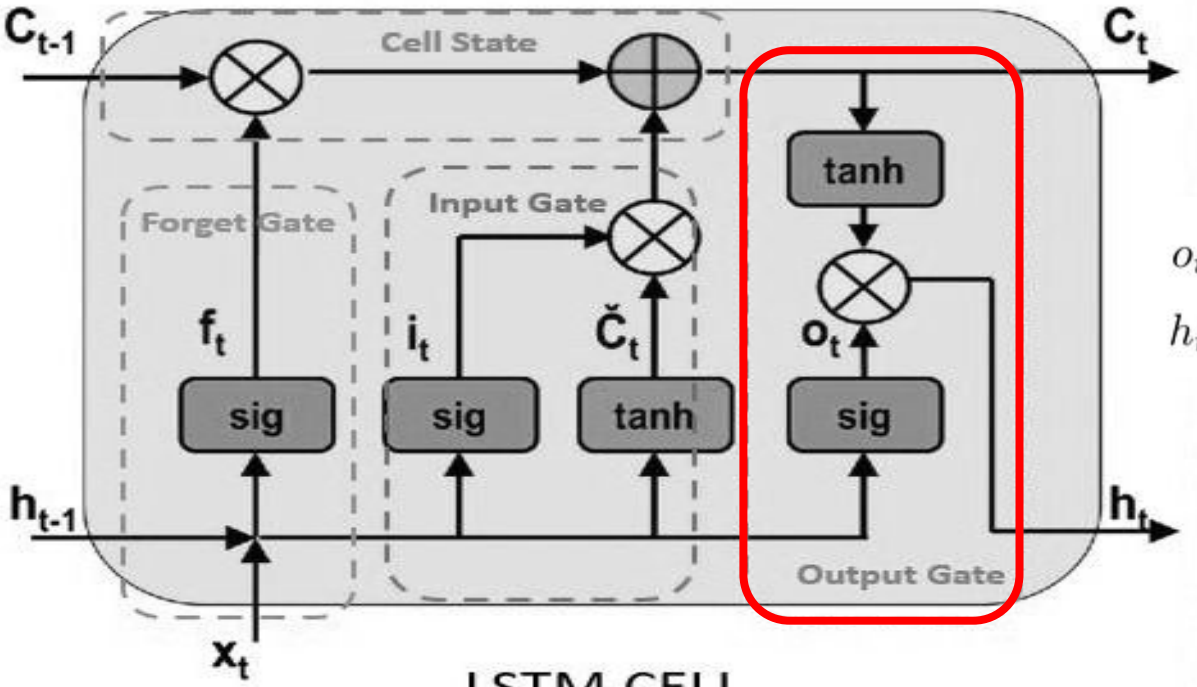
LSTM



LSTM CELL

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

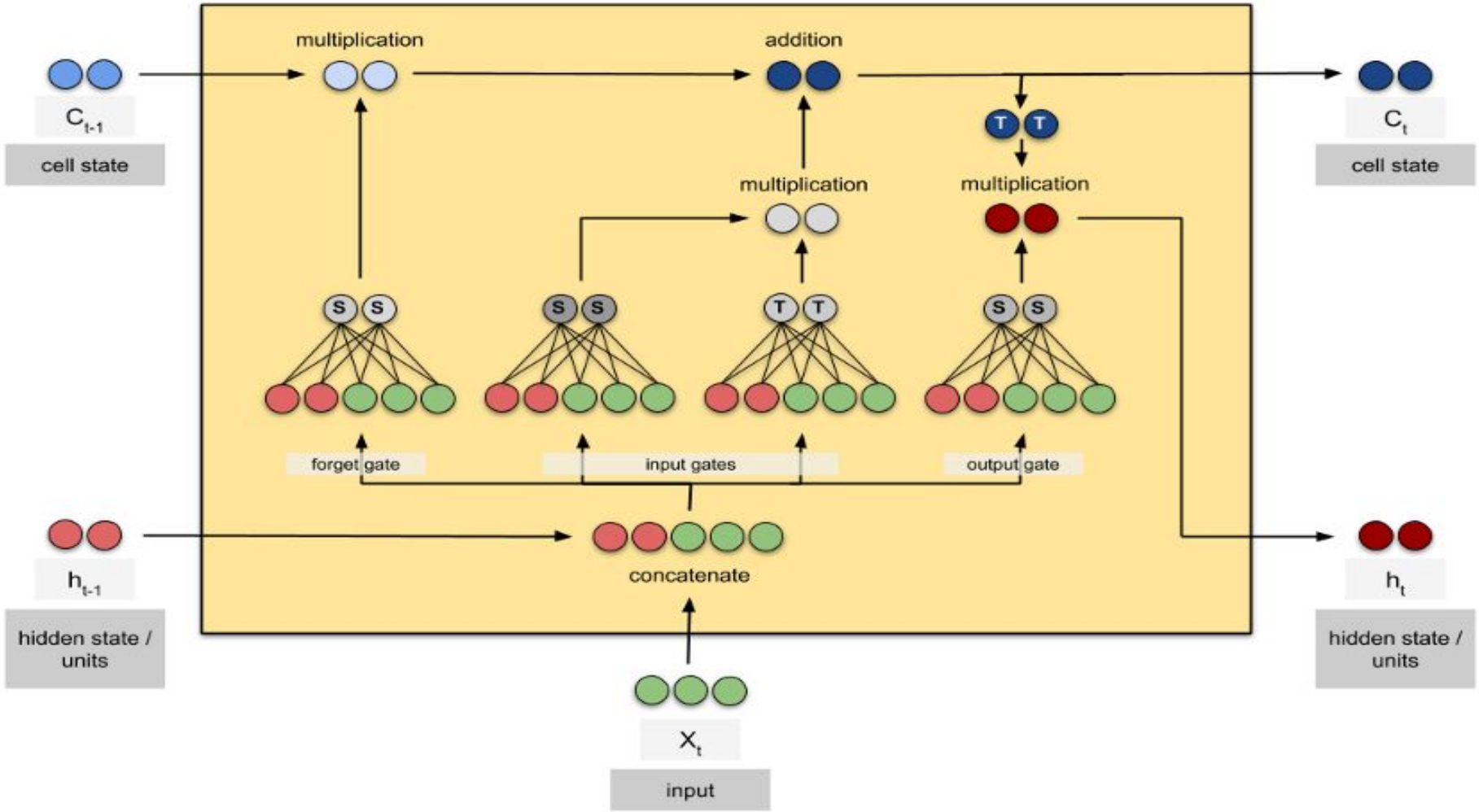
LSTM



LSTM CELL

$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

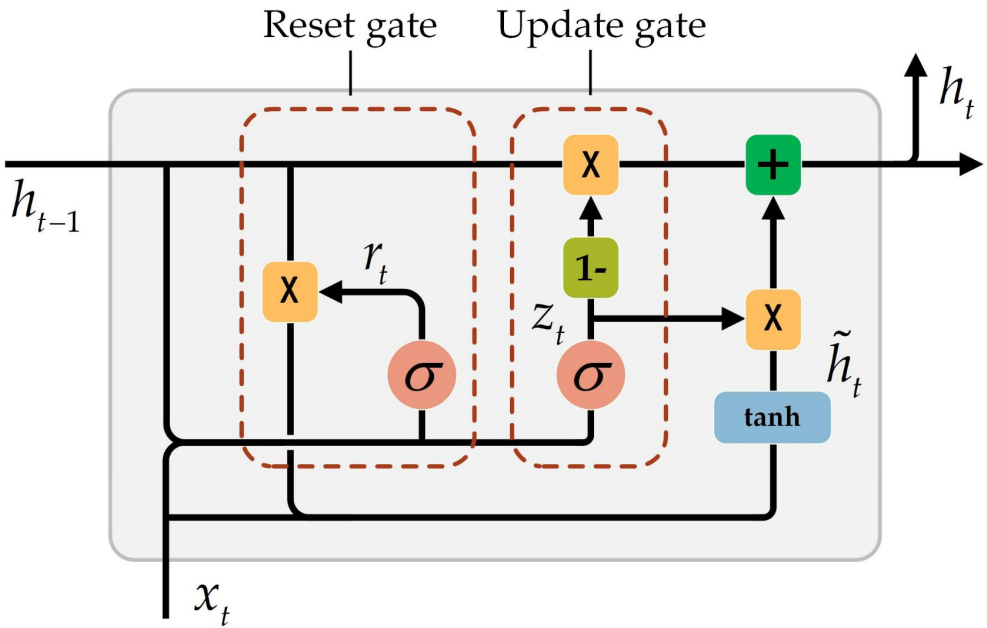
$$h_t = o_t * \tanh\left(C_t\right)$$


multiplication
addition
C_{t-1}
cell state
C_t
cell state
T T
multiplication
multiplication
S S
S S
T T
S S
forget gate
input gates
output gate
h_{t-1}
hidden state / units
concatenate
h_t
hidden state / units
X_t
input

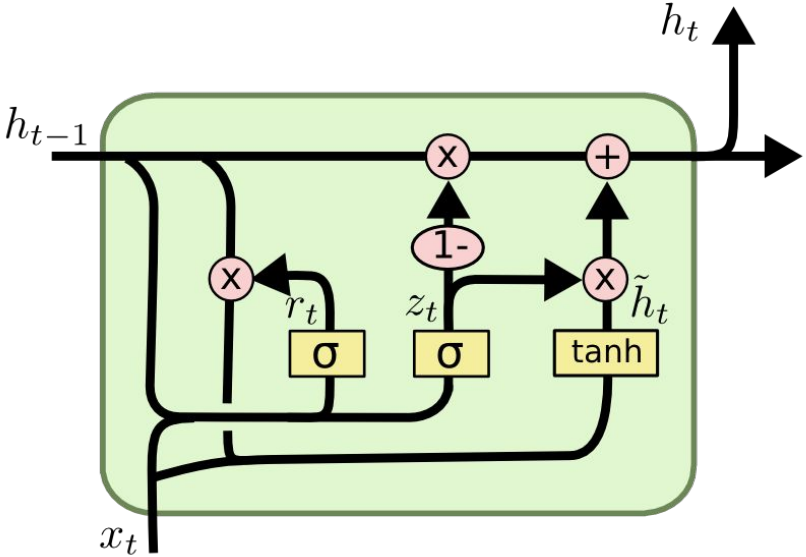
Gated Recurrent Unit (GRU)

- It is an RNN.
- Introduced in 2014.
- It is a “simplified LSTM”.
- Composed of 2 gates:
    - **Reset gate**: Decides what information to discard or to keep.
    - **Update gate**: Decides what new information to store.

GRU Cell


Reset gate
Update gate
h_t
h_{t-1}
X
+
X
r_t
1-
z_t
X
$\tilde{h}_t$
σ
σ
tanh
x_t

Gated Recurrent Unit (GRU)



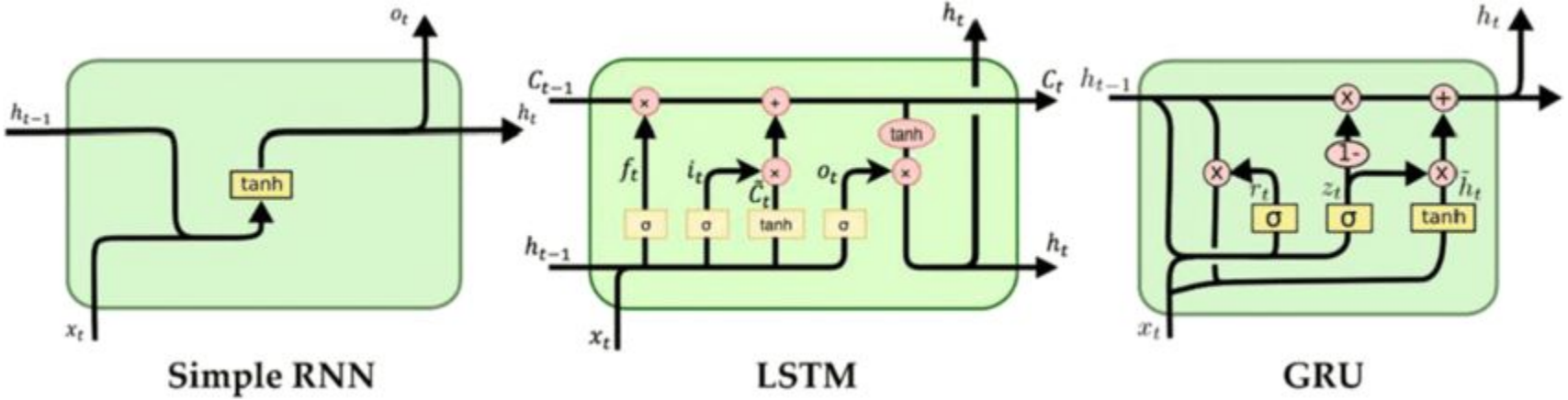
$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

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

RNN vs LSTM vs GRU



RNN vs LSTM vs GRU

- RNN
  - Generally not recommended for long sequences.
  - Vanishing/Exploding gradients.

RNN vs LSTM vs GRU

- LSTM
  - Usually performs better than simple RNN.
  - Good for long sequences.
  - Solves vanishing gradient problem.
  - Keeps exploding gradient problem.

RNN vs LSTM vs GRU

- GRU
  - Similar performance as LSTM.
  - Less parameters to train (i.e., easier and faster).
  - Less memory.
  - LSTMs may work better in very long sequences.

RNNs in Keras

- From tensorflow.keras.layers
  - SimpleRNN
  - LSTM
  - GRU

Typical Sklearn Input

	Temperature (X1)	Humidity (X2)
25/11/22	6	92
26/11/22	9	85
27/11/22	7	88
28/11/22	8	91
29/11/22	6	89

Shape: (5, 2)

```
X = [
    [6, 92],
    [9, 85],
    [7, 88],
    [8, 91],
    [6, 89],
]
```

Typical Sklearn Input

	Temperature (X1)	Humidity (X2)
25/11/22	6	92
26/11/22	9	85
27/11/22	7	88
28/11/22	8	91
29/11/22	6	89

```
X = [
    [6, 92],
    [9, 85],
    [7, 88],
    [8, 91],
    [6, 89],
]
```

Not working for keras RNNs

RNNs Keras Input

Decide: Sequence length.
E.g., 3

	Temperature (X1)	Humidity (X2)
25/11/22	6	92
26/11/22	9	85
27/11/22	7	88
28/11/22	8	91
29/11/22	6	89

Seq. 1

RNNs Keras Input

Decide: Sequence length.
E.g., 3

	Temperature (X1)	Humidity (X2)
25/11/22	6	92
26/11/22	9	85
27/11/22	7	88
28/11/22	8	91
29/11/22	6	89

Seq. 2

RNNs Keras Input

Decide: Sequence length.
E.g., 3

	Temperature (X1)	Humidity (X2)
25/11/22	6	92
26/11/22	9	85
27/11/22	7	88
28/11/22	8	91
29/11/22	6	89

Seq. 3

RNNs Keras Input

	Temperature (X1)	Humidity (X2)
25/11/22	6	92
26/11/22	9	85
27/11/22	7	88
28/11/22	8	91
29/11/22	6	89

Shape: (3, 3, 2)

```
X = [[[6, 92],
      [9, 85],
      [7, 88]],
     [[9, 85],
      [7, 88],
      [8, 91]],
     [[7, 88],
      [8, 91]
      [6, 89]]]
```

Resources

- https://colah.github.io/posts/2015-08-Understanding-LSTMs/
- https://www.kaggle.com/code/kmkarakaya/lstm-understanding-the-number-of-parameters
- https://www.youtube.com/watch?v=YCzL96nL7j0
- https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be