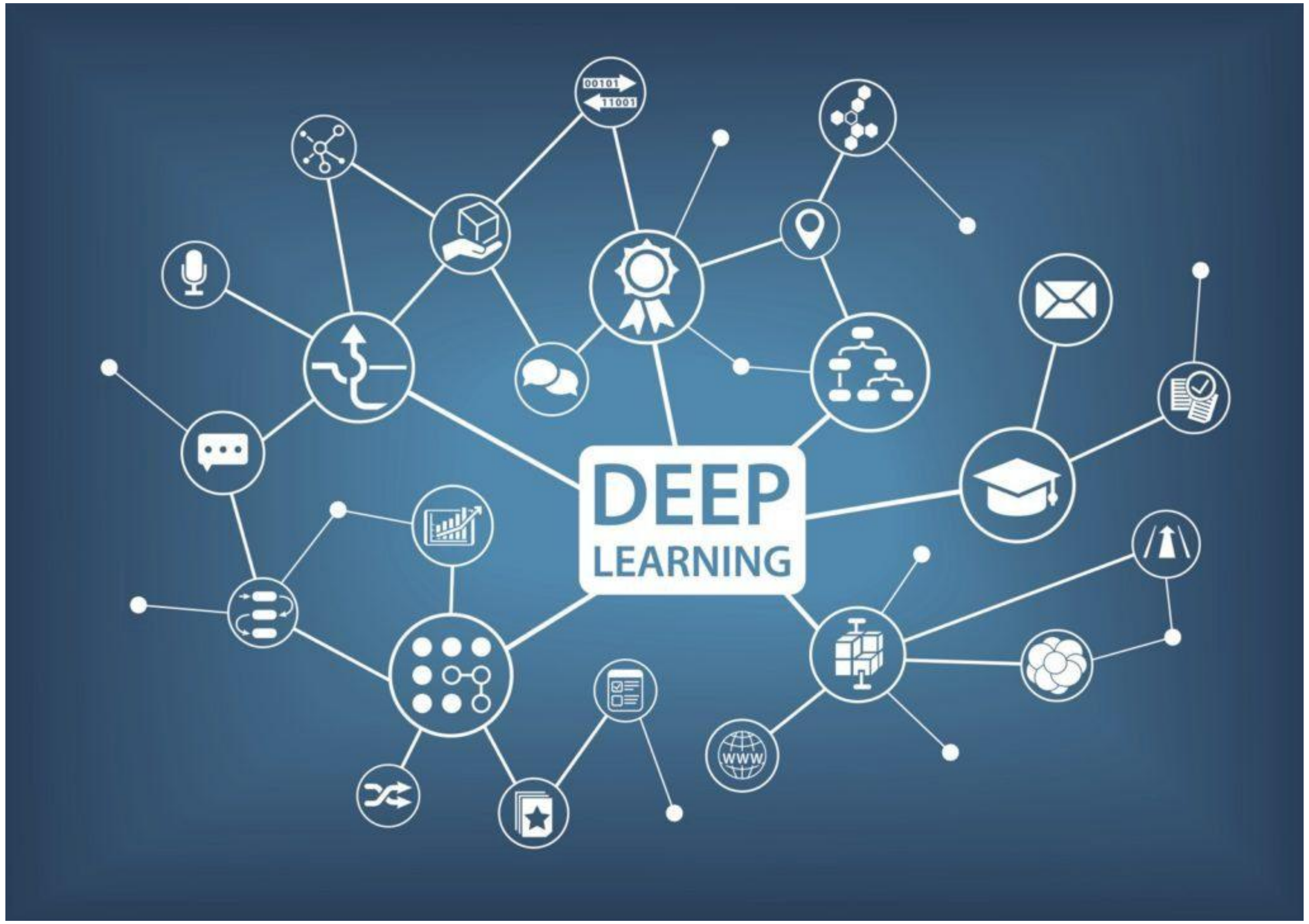
Recurrent Neural Networks(RNNs)

Presented by

NAME
Anamika Banwari
Victor Eghujovbo
Manikanta Teja Gullapalli

October 27, 2023

Presented to: Dr Yasser Alginahi



Source: https://tinyurl.com/34k38nut

Engineering
University of Windsor

Table of contents

source: https://tinyurl.com/2p8mcjep

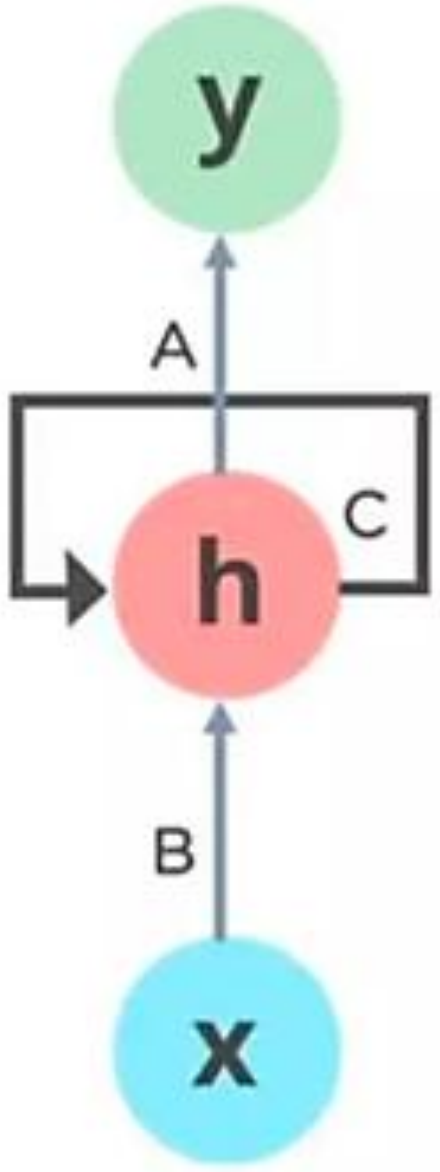
Sequence Modeling (RNN): Design Criteria

To model sequences, we need to:

1. Handle **variable-length** sequences
2. Maintain information about **order**
3. **Share parameters** across the sequences
4. Track **long-term** dependencies

was

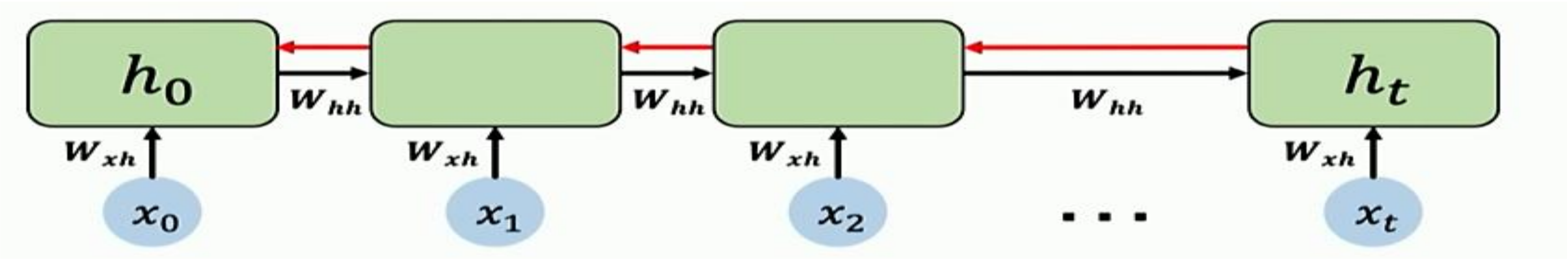
Ex: "The person who took my bike, ………………… a thief "



source: https://tinyurl.com/tsfcmx75

Recurrent Neural Networks (RNNs) meet these sequences modeling design criteria

Standard RNN Gradient Flow: Vanishing Gradients



Where h_o is the prediction from backward propagation
h_t is the hidden layer prediction at time t
x_0,x_1 *and* x_2 are the input vectors.
W_{hh} and W_{xh} are weights at the hidden layer and from the input vectors

Reference: https://tinyurl.com/3xyrn62w

Computing the gradient with respect to $\boldsymbol{h_o}$ involves **many factors of $\boldsymbol{W_{hh}}$** + **repeated gradient computation!**

Many Values < 1: (**vanishing gradients)**

1. Activation functions
2. Weight initialization
3. **Network architecture**

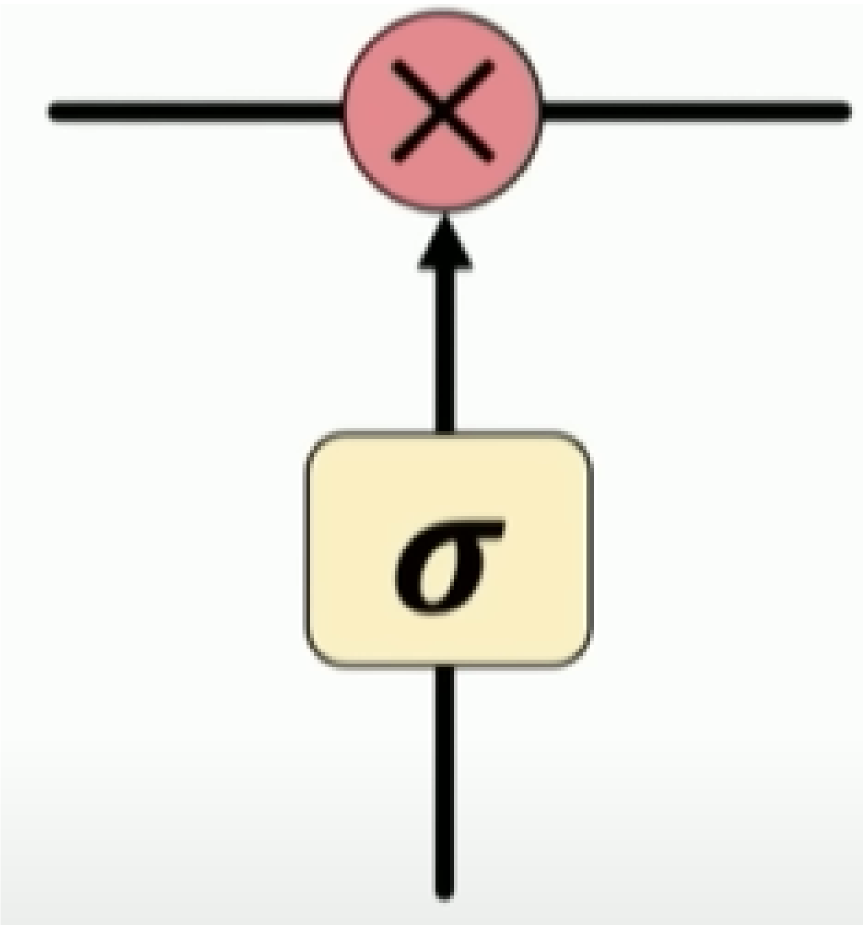
Gated Cells

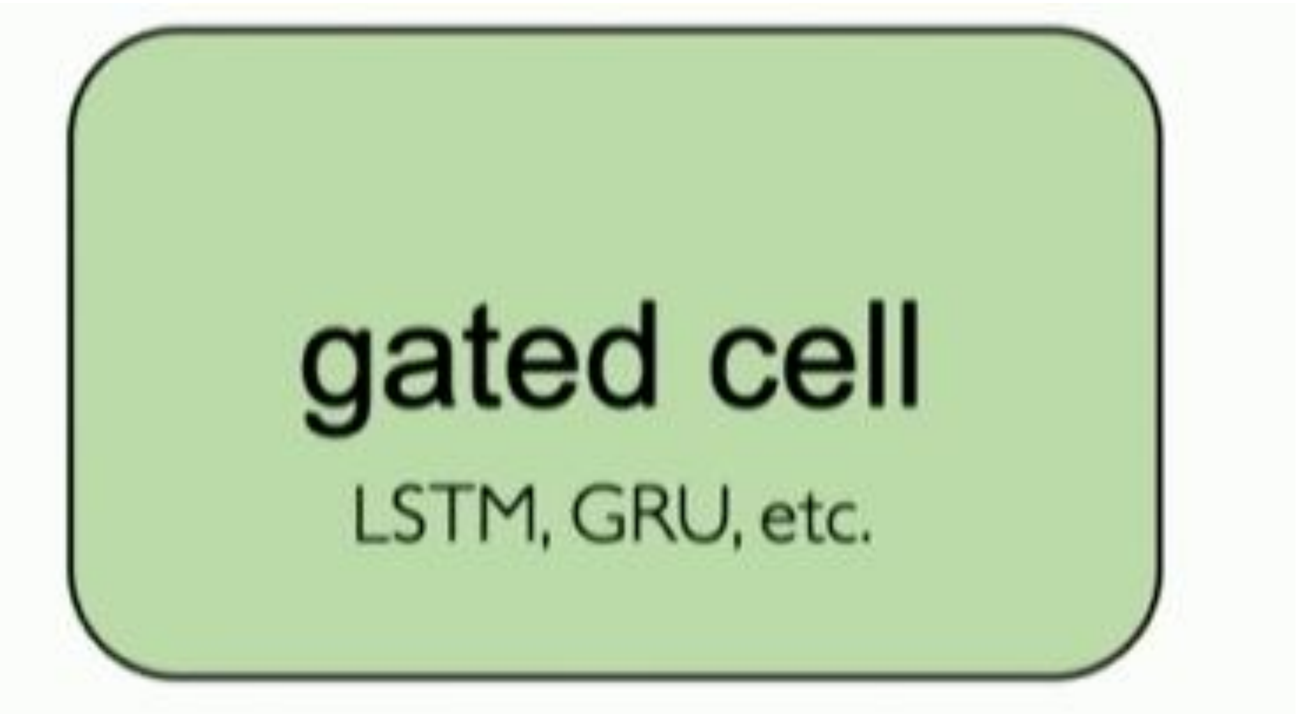
Idea: use **gates** to selectively **add** or **remove** information within

each recurrent unit with

Point wise multiplication

Sigmoid neural net layer





Gates optionally let information through the cell

Reference: https://tinyurl.com/3xyrn62w

Long Short-Term Memory (LSTM) networks rely on a gated cell to track information throughout many time steps.

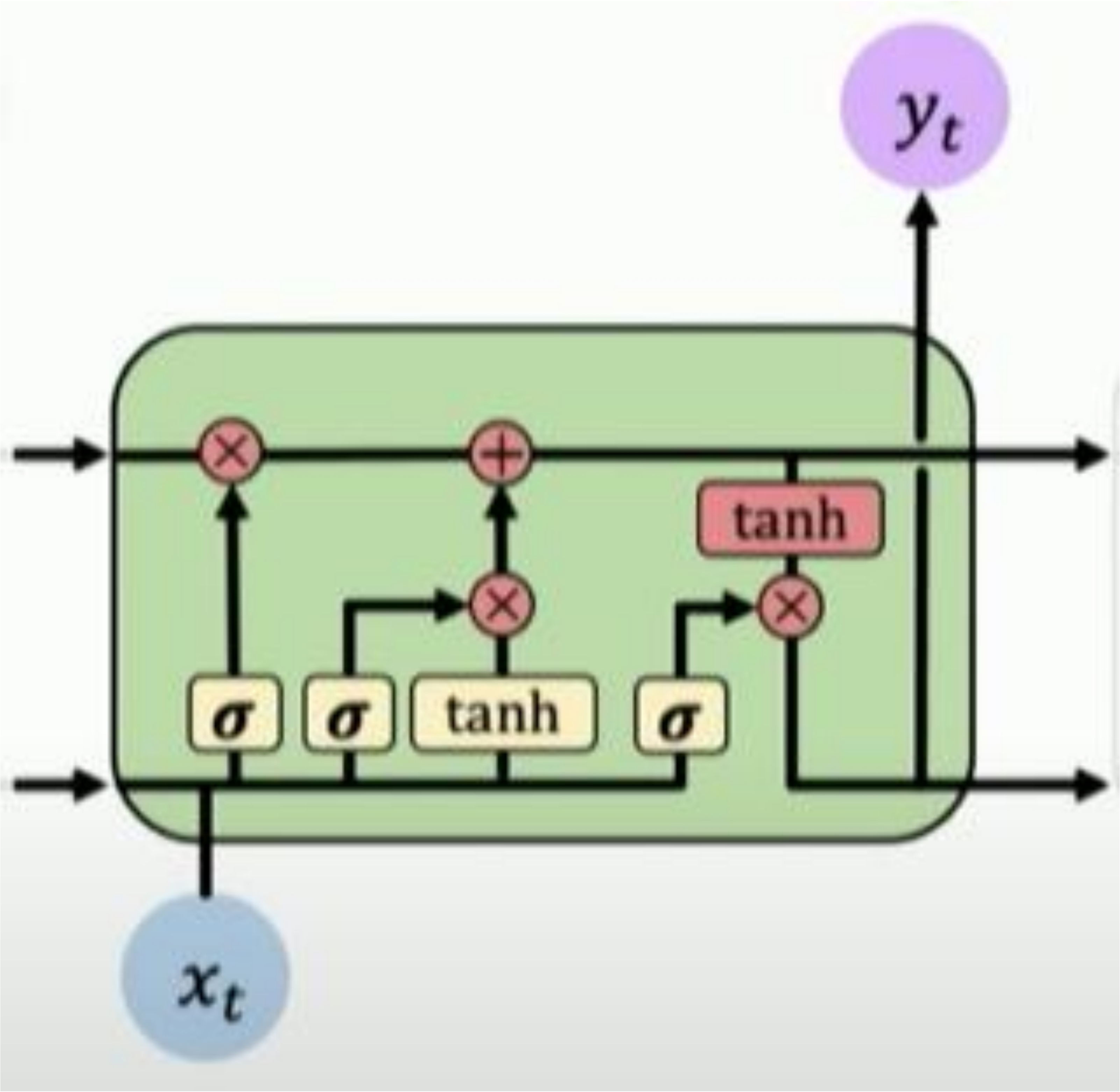
Long Short-Term Memory (LSTM)

Gated LSTM cells control information flow:

1) **Forget**
2) **Store**
3) **Update**
4) **Output**

LSTM is capable of learning long-term dependencies.

Remembering information for long periods of time

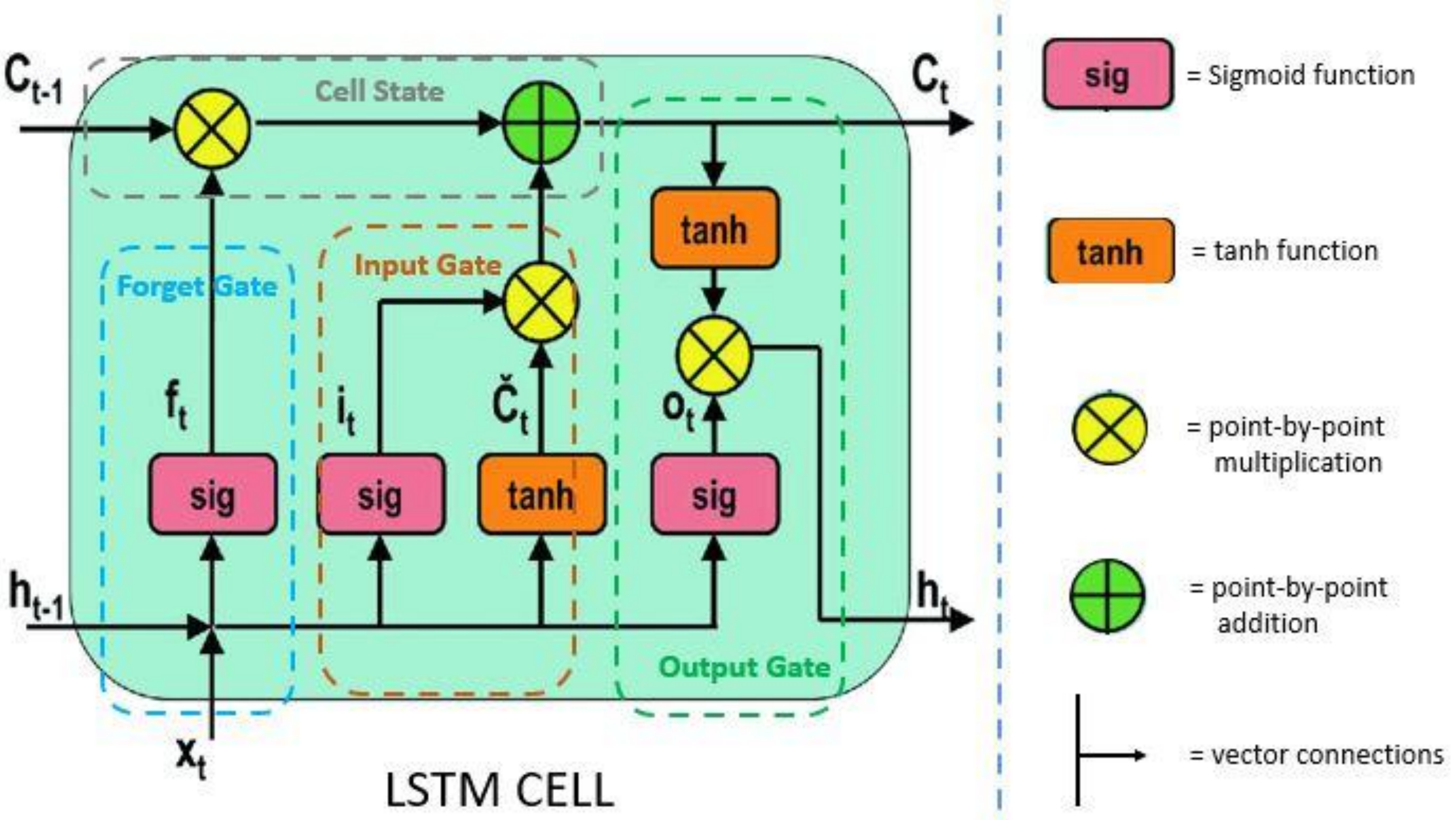


Reference: https://tinyurl.com/3xyrn62w

Structure of LSTM

It consists of:

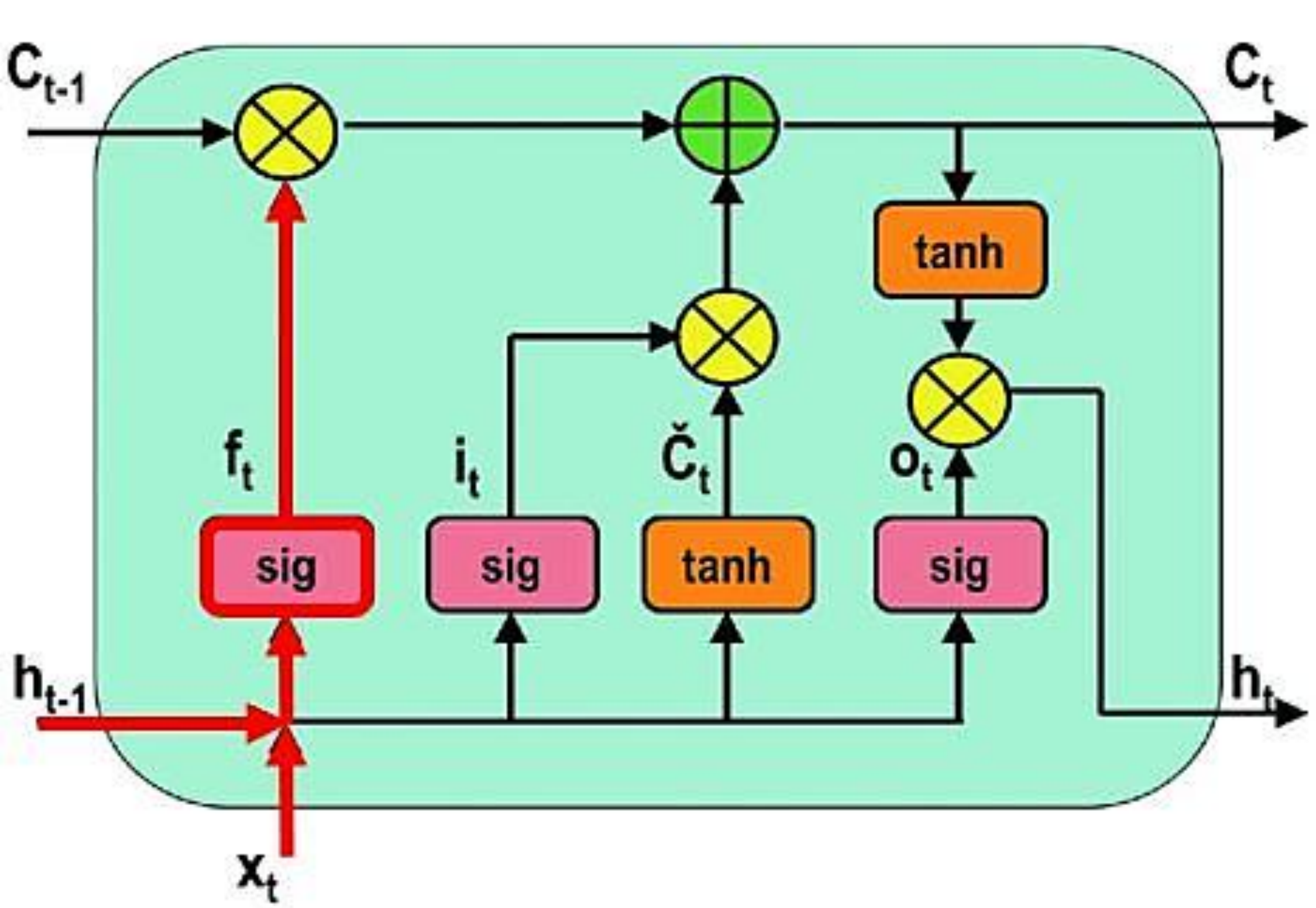
1. Four interactive layers
2. Gated unit
   - 3 logistic sigmoid gates
   - 1 cell state



Reference : https://tinyurl.com/3xyrn62w

3 logistic sigmoid gates

FORGET GATE



Forget Gate Operation

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

$t = timestep$

$f_t = forget\ gate\ at\ t$

$x_t = input$

$h_{t-1} = Previous\ hidden\ state$

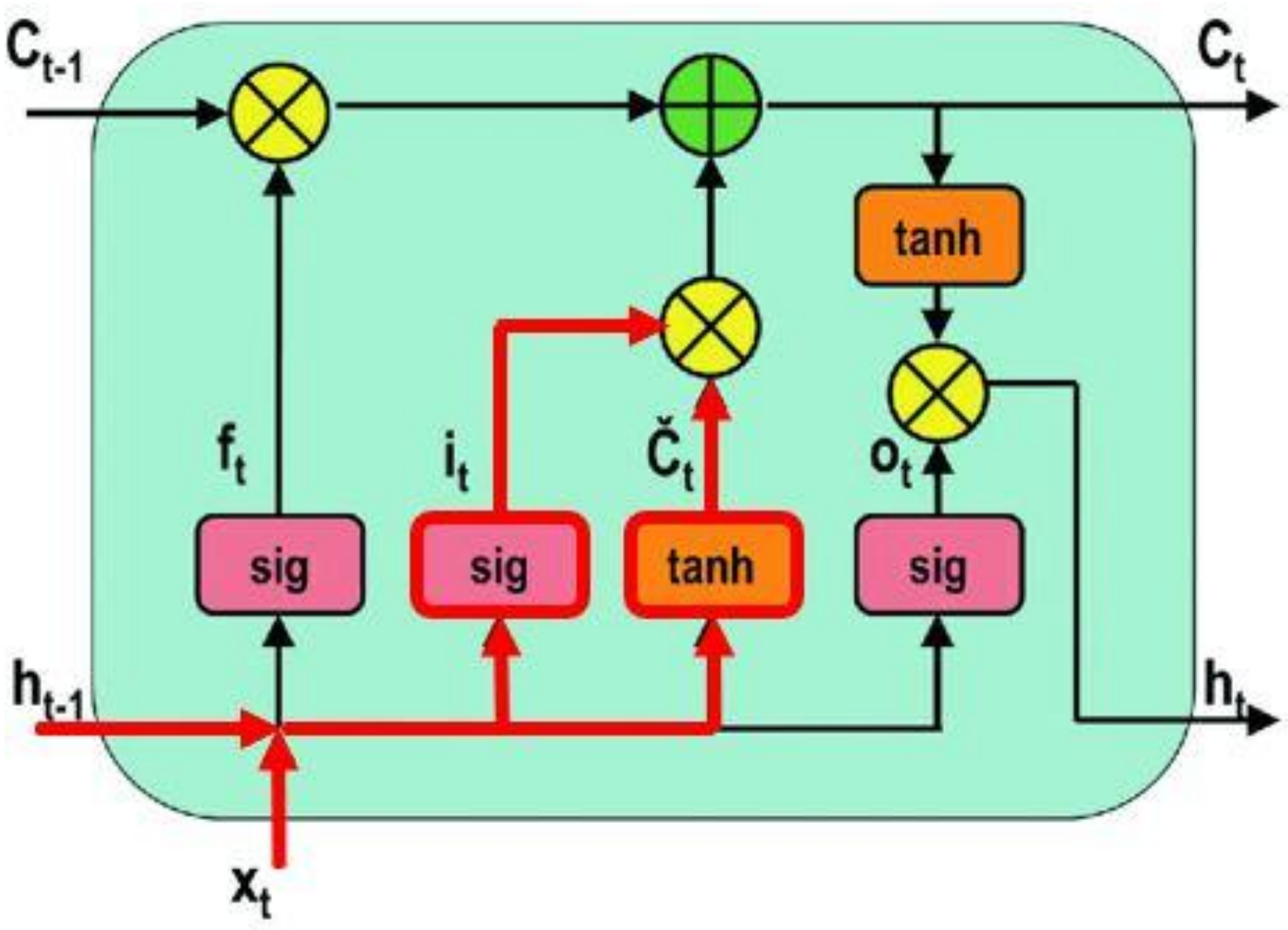
$W_f = Weight\ matrix\ between$
$forget\ gate\ and\ input\ gate$

$b_t = connection\ bias\ at\ t$

Reference : https://tinyurl.com/3xyrn62w

Engineering
University of Windsor

INPUT GATE



Input Gate Operation

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

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

t = *timestep*

i_t = *input gate at t*

W_i = *Weight matrix of sigmoid operator between input gate and output gate*

b_t = *bias vector at t*

$C{\sim}_t$ = *value genrated by tanh*

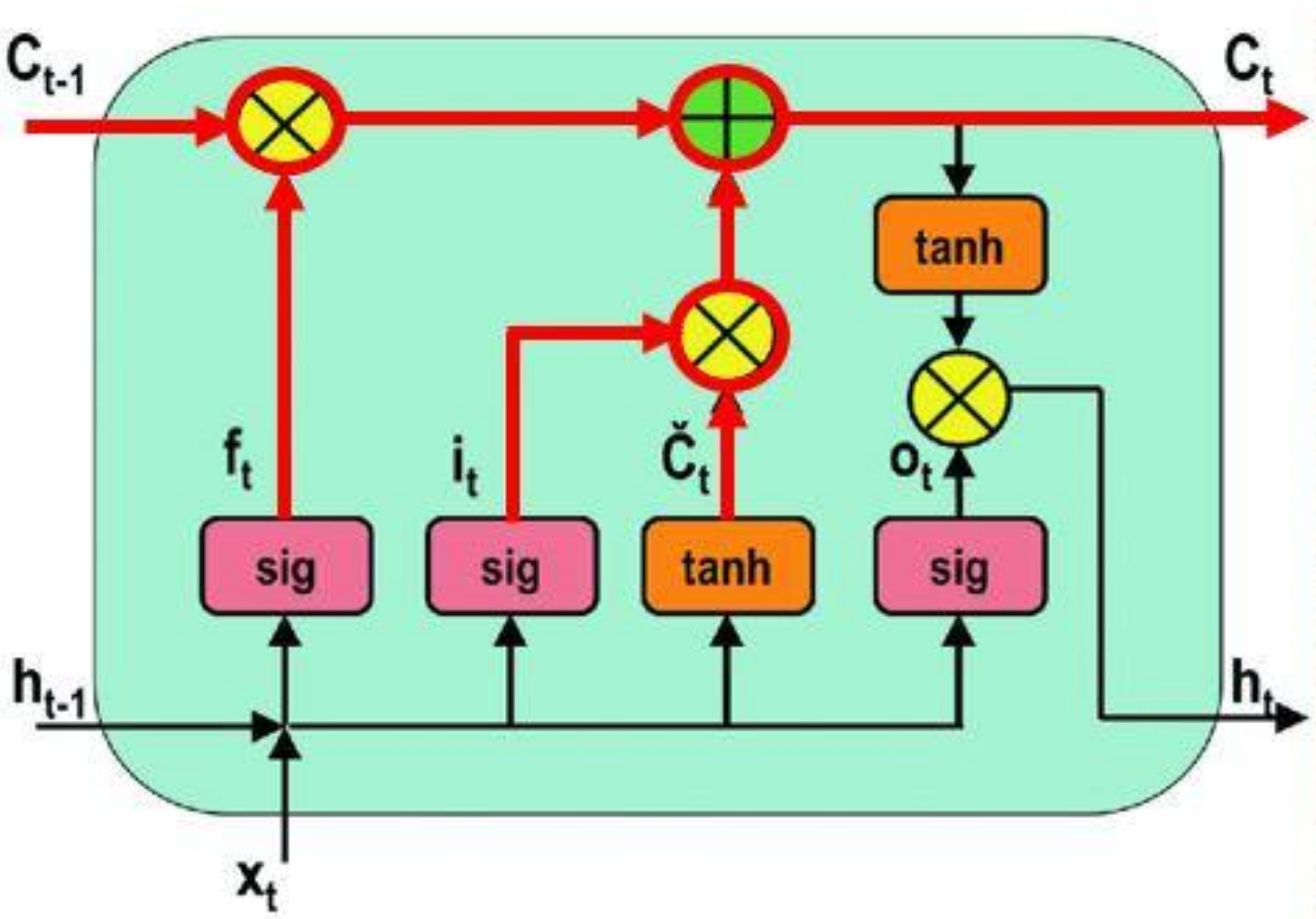
W_c = *Weight matrix of tanh operator between cell state information and network output*

b_c = *bias vector at t, w.r.t* W_c

Reference : https://tinyurl.com/3xyrn62w

Engineering
University of Windsor

Cell State



Cell State Operation

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

$t = timestep$

$C_t = cell\ state\ information$

$f_t = forget\ gate\ at\ t$

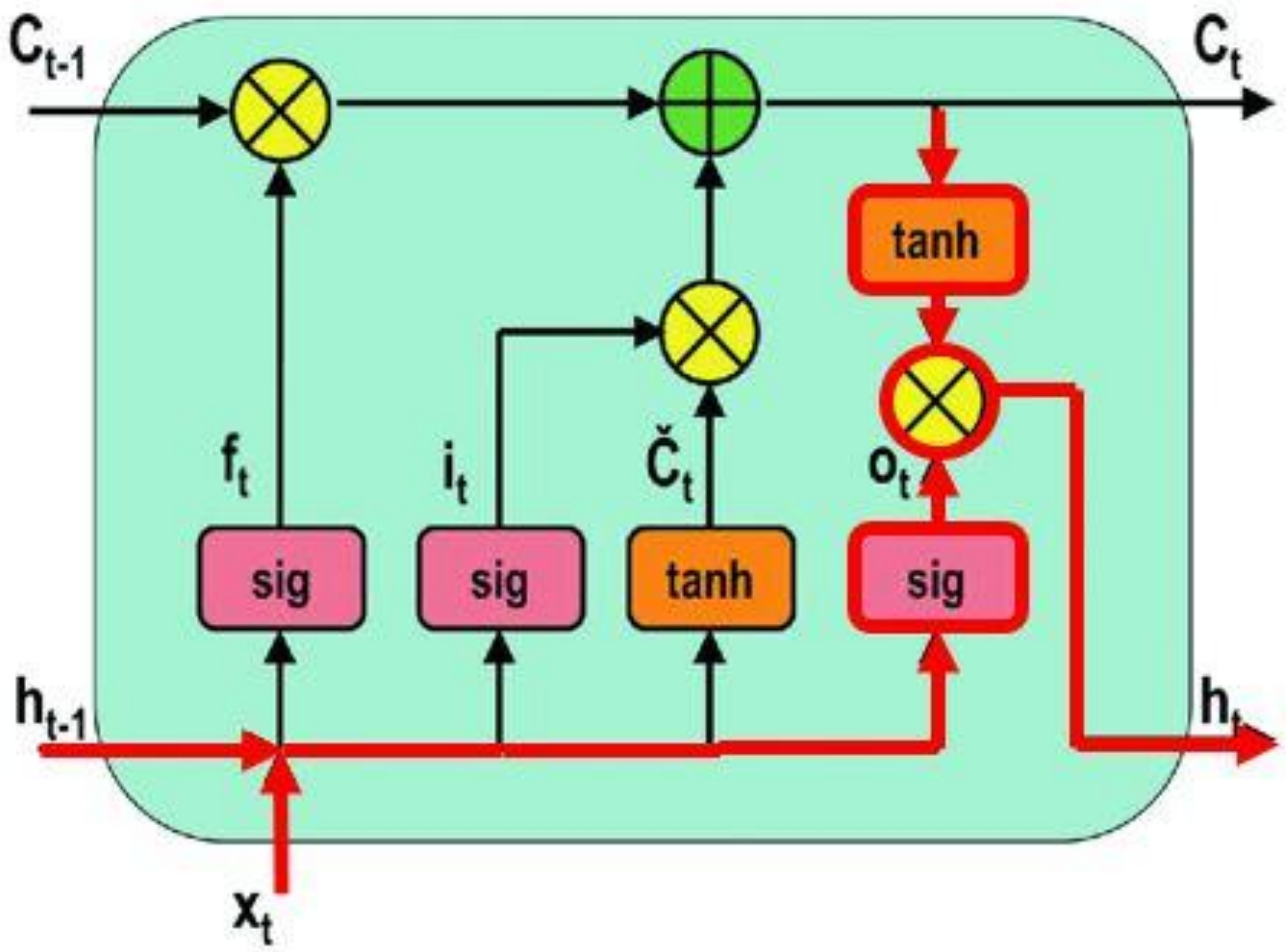
$i_t = input\ gate\ at\ t$

$C_{t-1} = Previous\ timestemp$

$C\sim_t = value\ genrated\ by\ tanh$

Reference : https://tinyurl.com/3xyrn62w

OUTPUT GATE



Output Gate Operation

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

$t = timestep$

$O_t = output\ gate\ at\ t$

$W_o = Weight\ matrix\ of\ output\ gate$

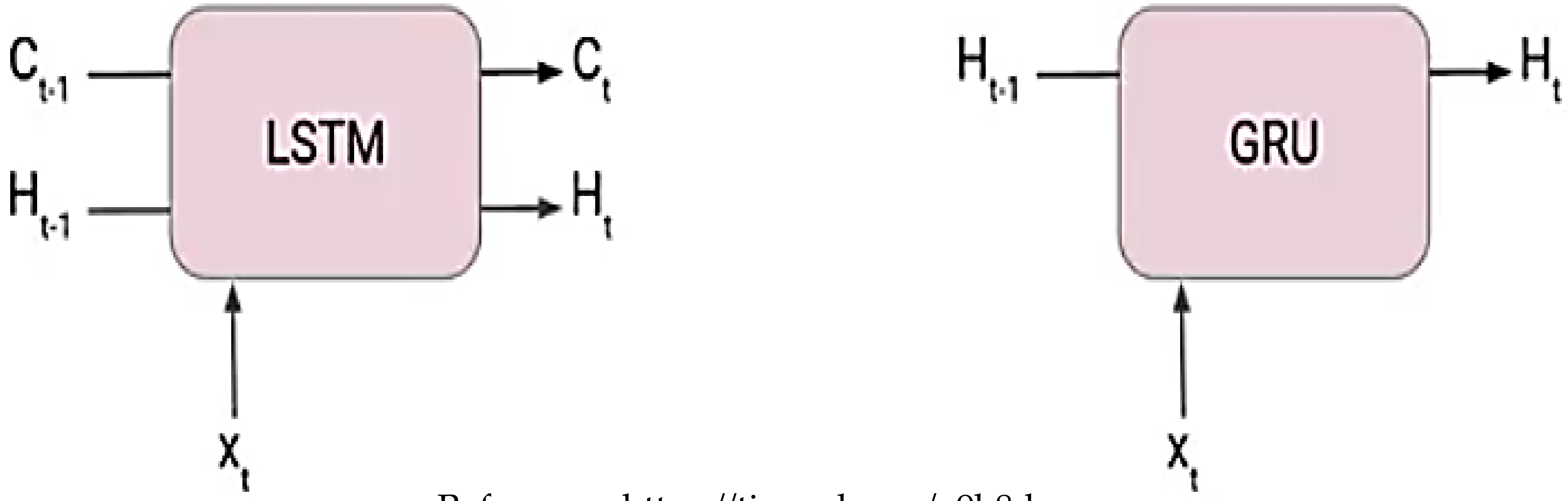
$b_o = bias\ vector, w.r.t\ W_o$

$h_t = LSTM\ output$

Reference: https://tinyurl.com/3xyrn62w

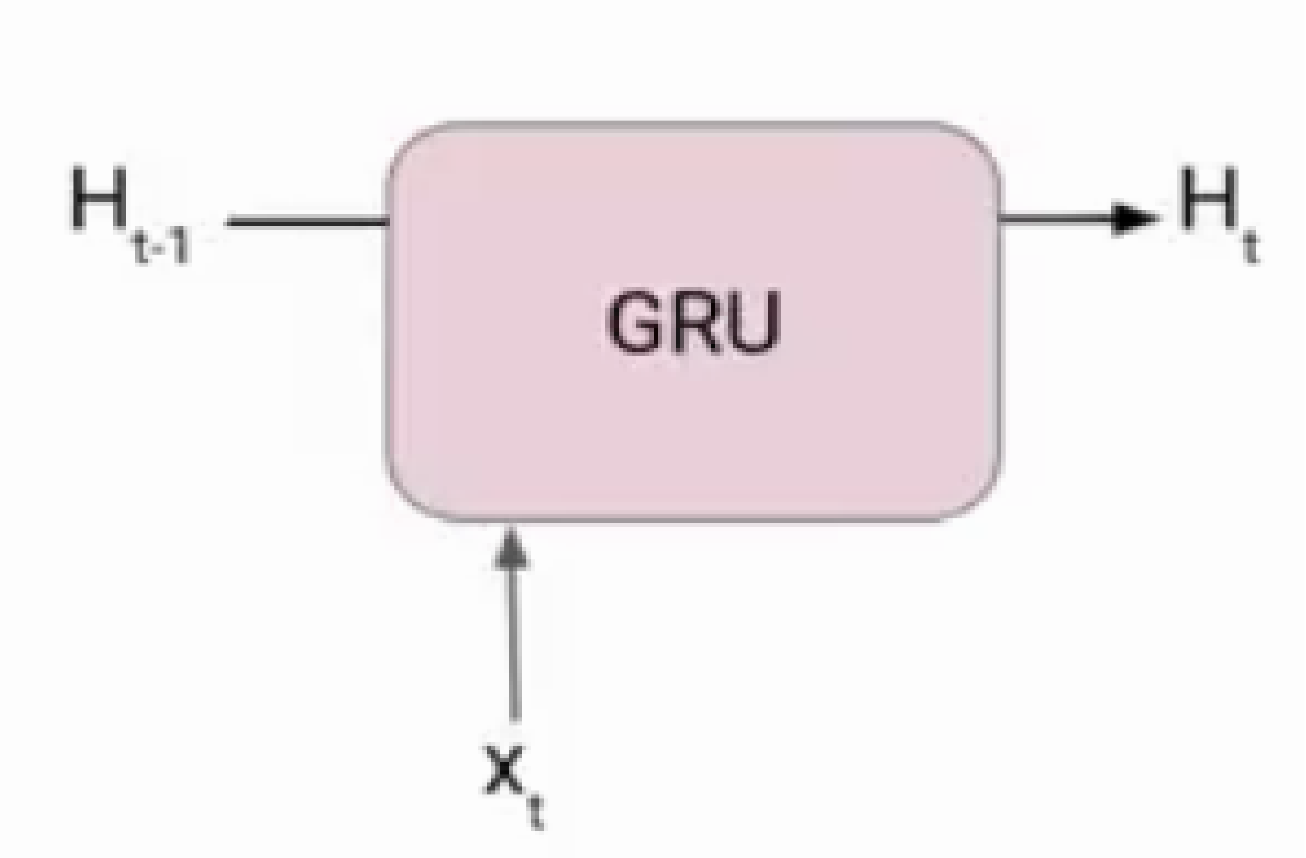
Optimized RNN!

- **Gradient clipping**
  - Avoid the **vanishing/exploding gradient problem** by looking at a threshold and clip the gradient.
  - Simple to address the issue but might **hamper the performance**.
  - For system optimization **LSTM** was introduced and much recently **GRU.**



Reference : https://tinyurl.com/y9b8desy

The Gated Recurrent Unit (GRU) architecture.



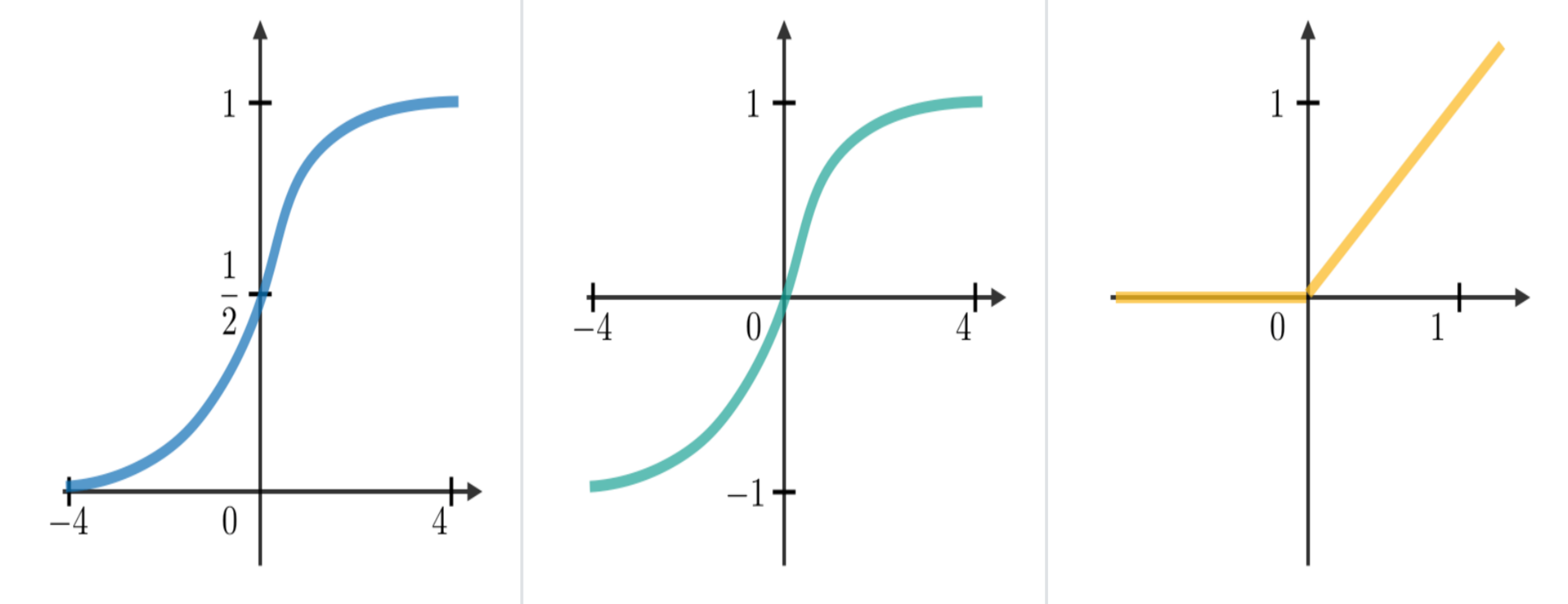
Reference : https://tinyurl.com/n7djd7ax

- At each timestamp t, it takes an input X_t and the hidden state H_{t-1} from the previous timestamp t-1.
- Later it outputs a new hidden state H_t which again passed to the next timestamp.

There are only two gates in a GRU as opposed to three gates in an LSTM cell. The first gate is the Reset gate and the other one is the update gate.

Common Activation Functions

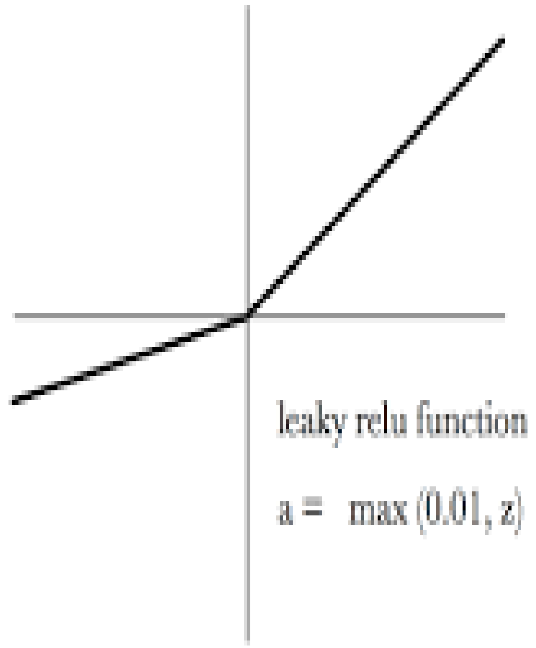
Sigmoid	Tanh	RELU
$g(z) = \dfrac{1}{1+e^{-z}}$	$g(z) = \dfrac{e^z - e^{-z}}{e^z + e^{-z}}$	$g(z) = \max(0, z)$



Reference : https://tinyurl.com/n7djd7ax

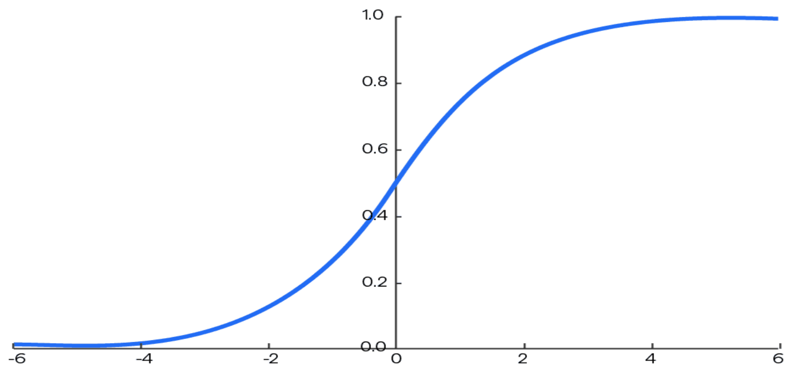
Leaky Relu Function

Leaky ReLU(x) = max(0.01x, x).



SoftMax Function

SoftMax(x) = e^x / ∑(e^x)



GRU Gate and memory allocation

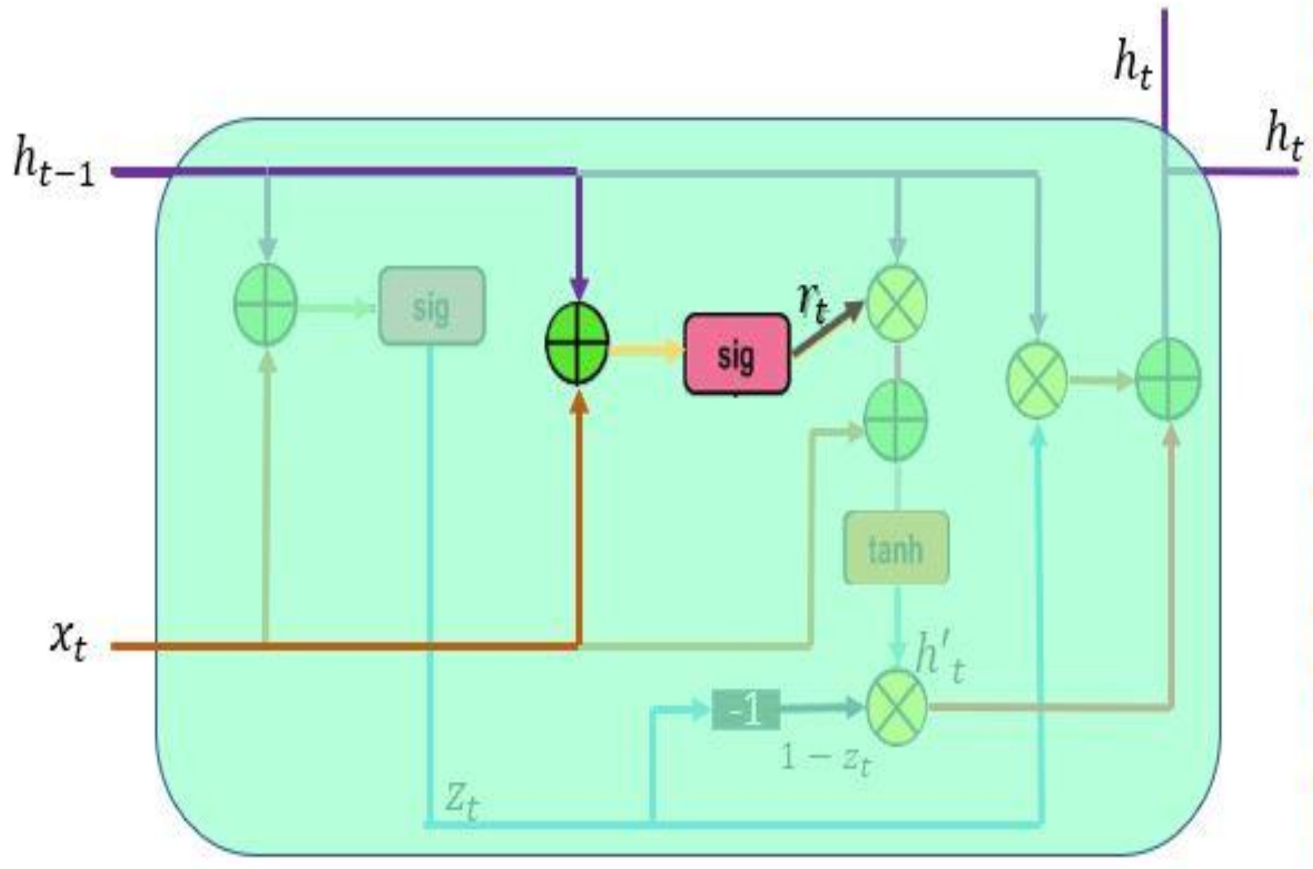
- **The Reset Gate (Short Term memory)**

This is responsible for the short-term memory of the network in the hidden state (H_t).

Equation of the Reset gate is:

$$\boldsymbol{r_t = \sigma(X_t * U_r + H_{t-1} * W_r) + b_r}$$

U_r and W_r are weight matrices for the reset gate.



Reset Gate Operation

Reference: https://tinyurl.com/5n933r7c

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

t = *timestep*

r_t = *Reset gate at t*

x_t = *input vector*

h_{t-1} = *Previous hidden state*

W_r = *Reset gate weight matrix*

b_r = *connection bias at t*

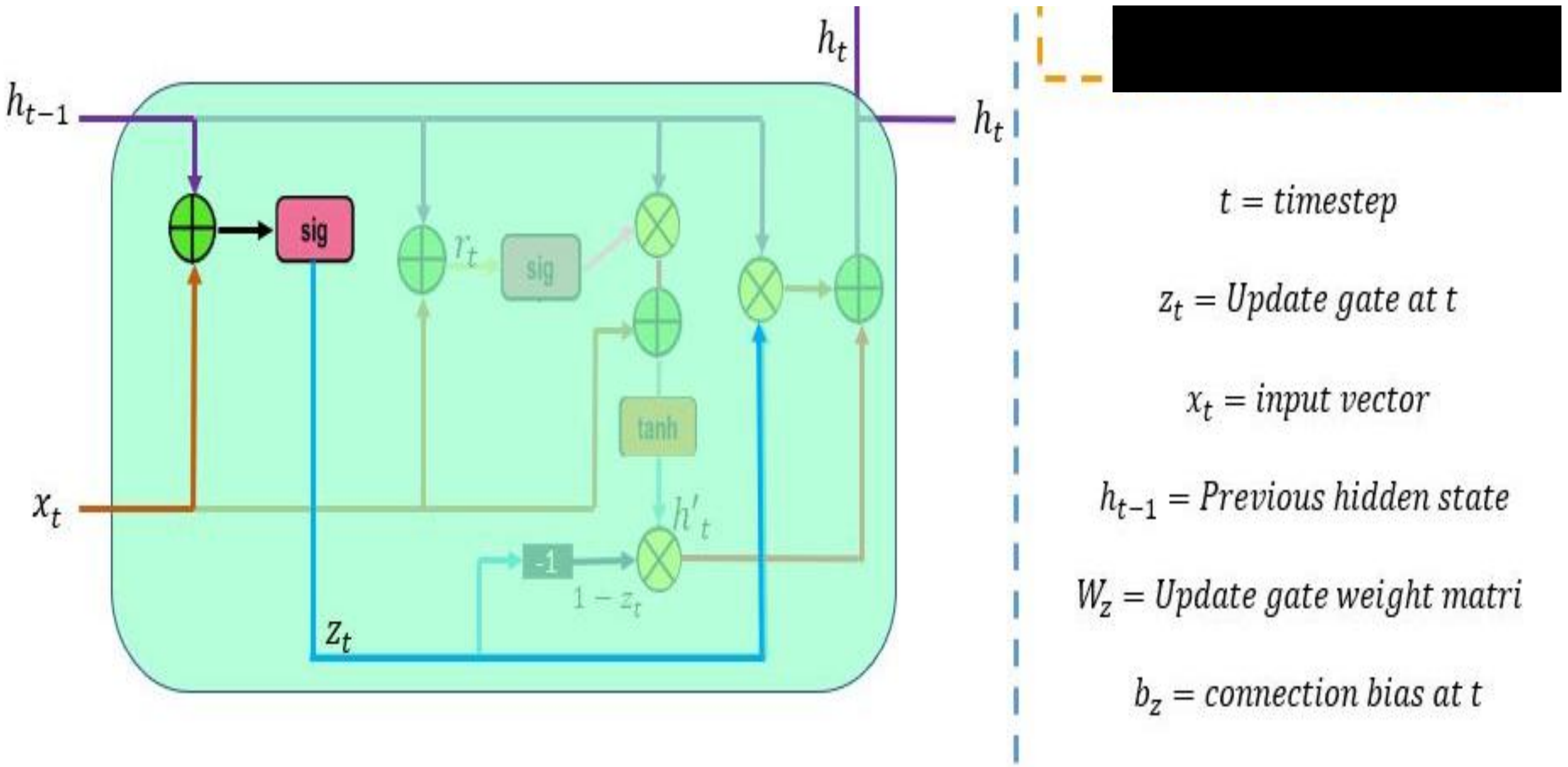
GRU Gate and memory allocation

- **The Update Gate (Long Term memory)**

  This is for long-term memory in the hidden state (H_t).

  Equation of the gate is shown below.

$$\boldsymbol{u_t = \sigma(X_t{}^*U_u + H_{t-1}{}^*W_u) + b_z}$$



Update Gate Operation

Reference: https://tinyurl.com/5n933r7c

Engineering
University of Windsor

How GRU works

- To find the Hidden state $\boldsymbol{H_t}$, it follows a two-step process. The first step is to generate what is known as the candidate hidden state.

The Candidate Hidden state ($\hat{H}_t$)

$$\hat{H}_t = \tanh(\boldsymbol{X_t}*\boldsymbol{U_g} + (\boldsymbol{r_t}*\boldsymbol{H_{t-1}})*\boldsymbol{W_g})$$

(for $r_t = 0$ and $r_t = 1$)

- The input and the hidden state from the previous timestamp t-1 is multiplied by the reset gate output $\boldsymbol{r_t}$
- Later passed this entire information to the **tanh function**, the resultant value is the candidate's hidden state $\hat{H}_t$

The Hidden state (H_t)

$$\boldsymbol{H_t} = \boldsymbol{u_t}*\boldsymbol{H_{t-1}} + (1-\boldsymbol{u_t})*\hat{H}_t$$

(for $u_t = 0$ and $u_t = 1$)

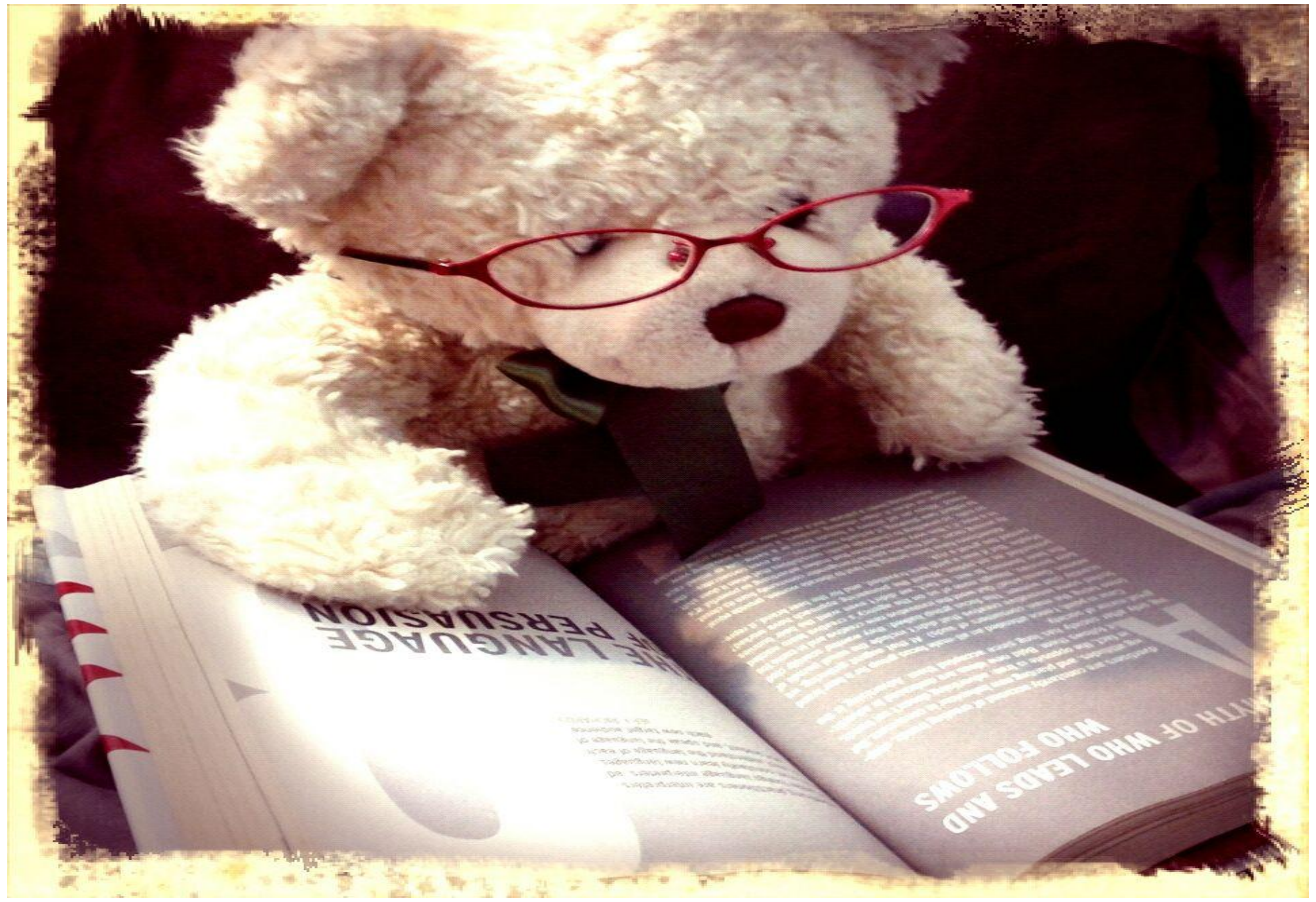
- This is where we implement the Update gate.
- We use a single update gate to control both the historical information $\boldsymbol{H_{t-1}}$ as well as the new information which comes from the candidate state $\hat{H}_t$

Reference: https://tinyurl.com/5n933r7c

Text prediction

- Neural network itself is not equipped to handle language explicitly, they are just functional / mathematical operators [9].
- We need to define a way to translate this text language into numerical encoding of a vector which then fed to a neural network generating a numerical vector.
- But how??

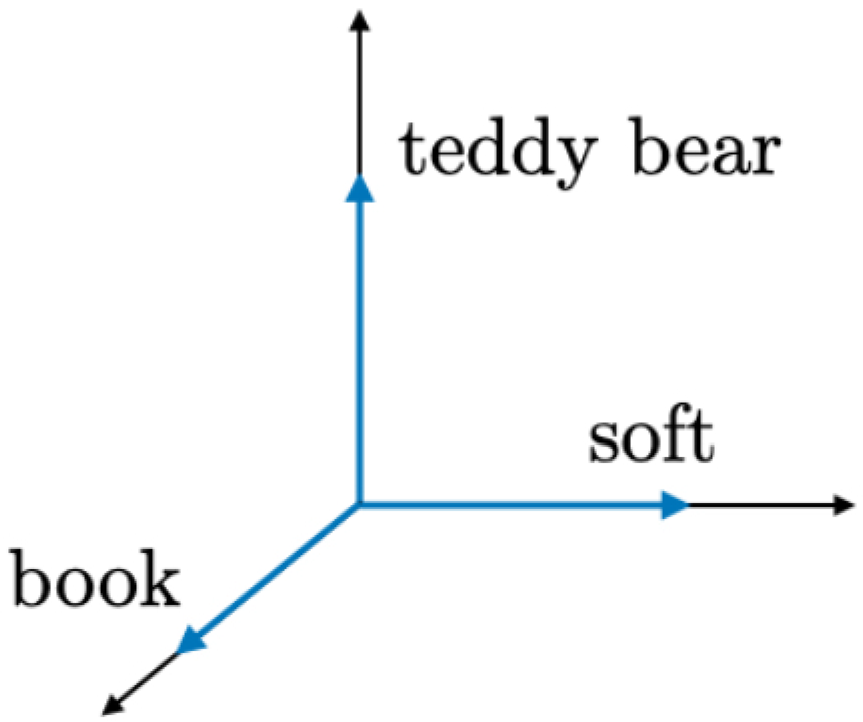
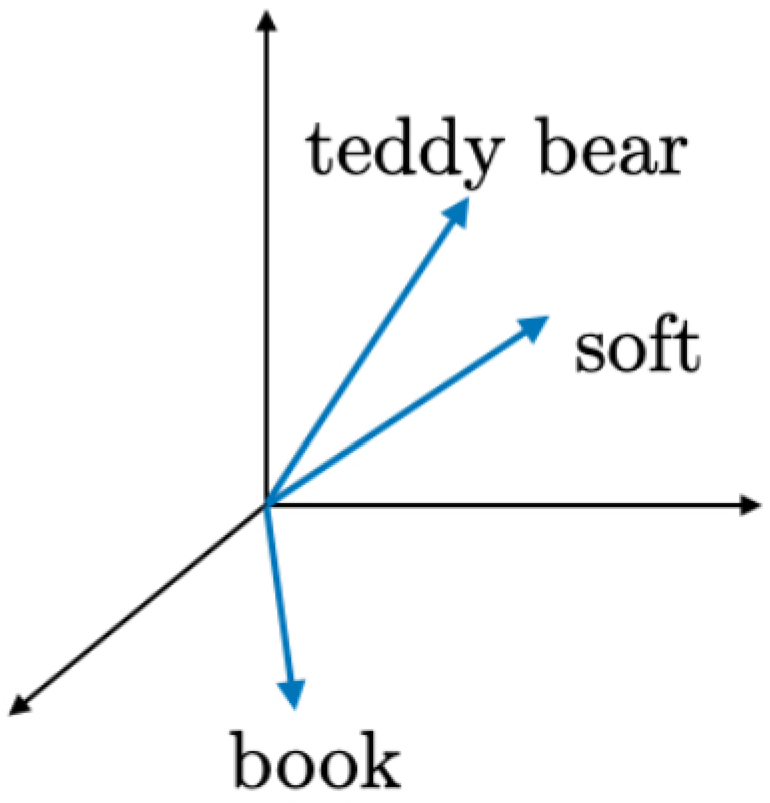
example : A cute teddy bear is reading a book



Source: https://tinyurl.com/3k9mhrjr

Engineering
University of Windsor

Embedding

1-hot representation	Word embedding
	
• Noted o_w • Naive approach, no similarity information	• Noted e_w • Takes into account words similarity

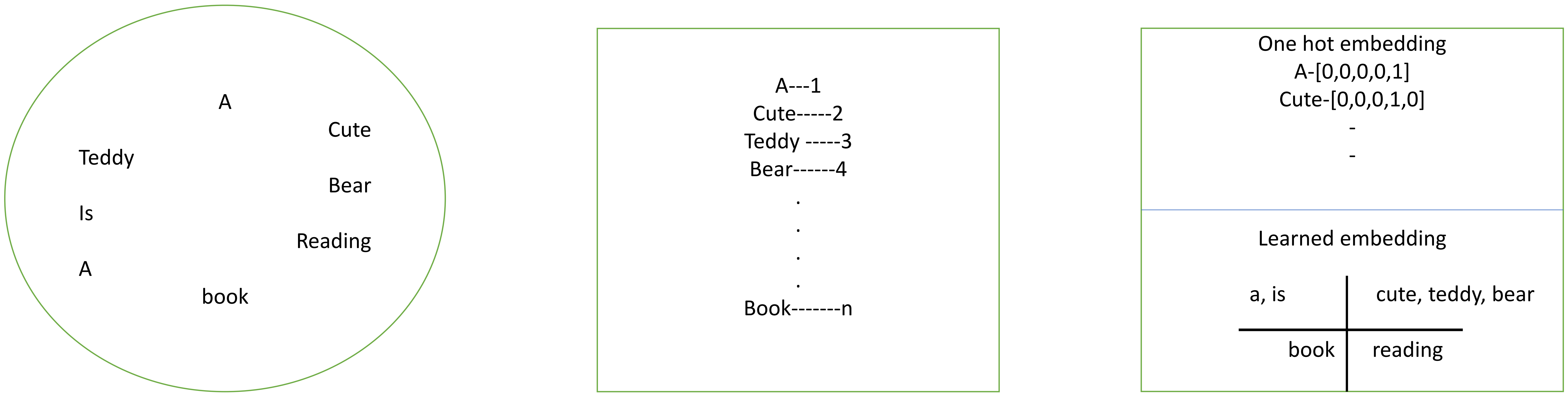
$$\boxed{e_w = E o_w}$$

E-embedding Matrix

Reference : https://tinyurl.com/n7djd7ax

Engineering
University of Windsor

Embedding

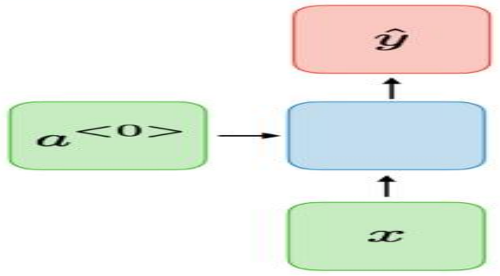
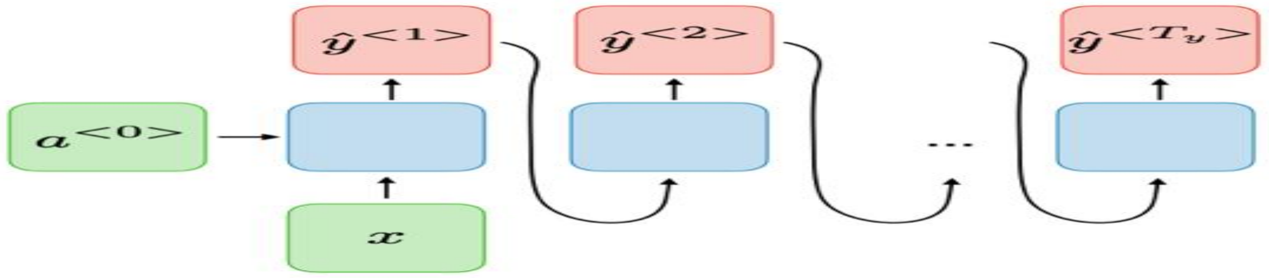
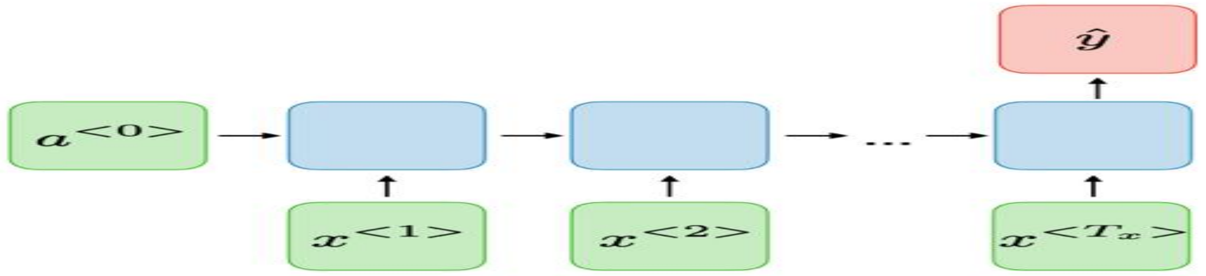
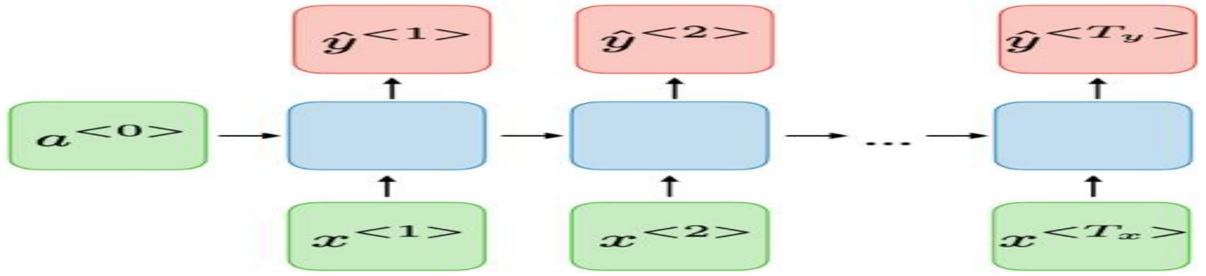
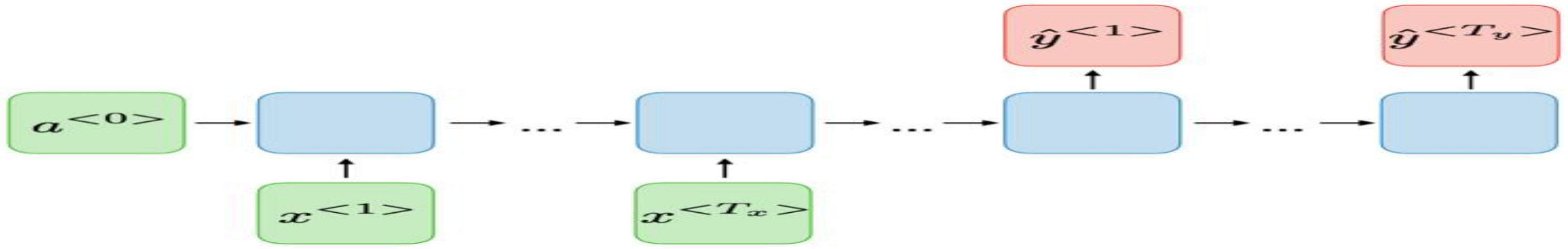


Bag of words/corpus of words

Indexing

Embedding

Types of RNN and their applications

Type of RNN	Illustration
One-to-one $T_x = T_y = 1$	
One-to-many $T_x = 1, T_y > 1$	
Many-to-one $T_x > 1, T_y = 1$	
Many-to-many $T_x = T_y$	
Many-to-many $T_x \neq T_y$	

Reference: https://tinyurl.com/n7djd7ax

Engineering
University of Windsor

Real-Life Applications of RNNs

Language Processing (LP)

- RNNs are commonly used in LP tasks such as translation, sentiment analysis, and speech recognition.

Time-Series Analysis

- RNNs can be used to analyze time-series data such as stock prices, weather patterns, and health records by studying trends and making future predictions

Image and Video Analysis

- RNNs can derive information and write text comments on the image/ video. This process is known as computer vision .

Advantages of RNNs

- Ability to process sequential data.
- Flexibility in handling variable-length inputs and outputs.
- Memory of previous inputs can be retained and used to inform future predictions.
- Can learn from past experiences and adjust its predictions accordingly

Limitation of RNNs

- Encoding bottleneck -- practically challenging as we are encoding time step by time step into a single output at the very last . We must ensure all this is done smoothly which might result in output
- Slow , no parallelization
- Not long memory

Attention

- The main goal of attention is to eliminate recurrence
- Identify which parts to attend most important features in input

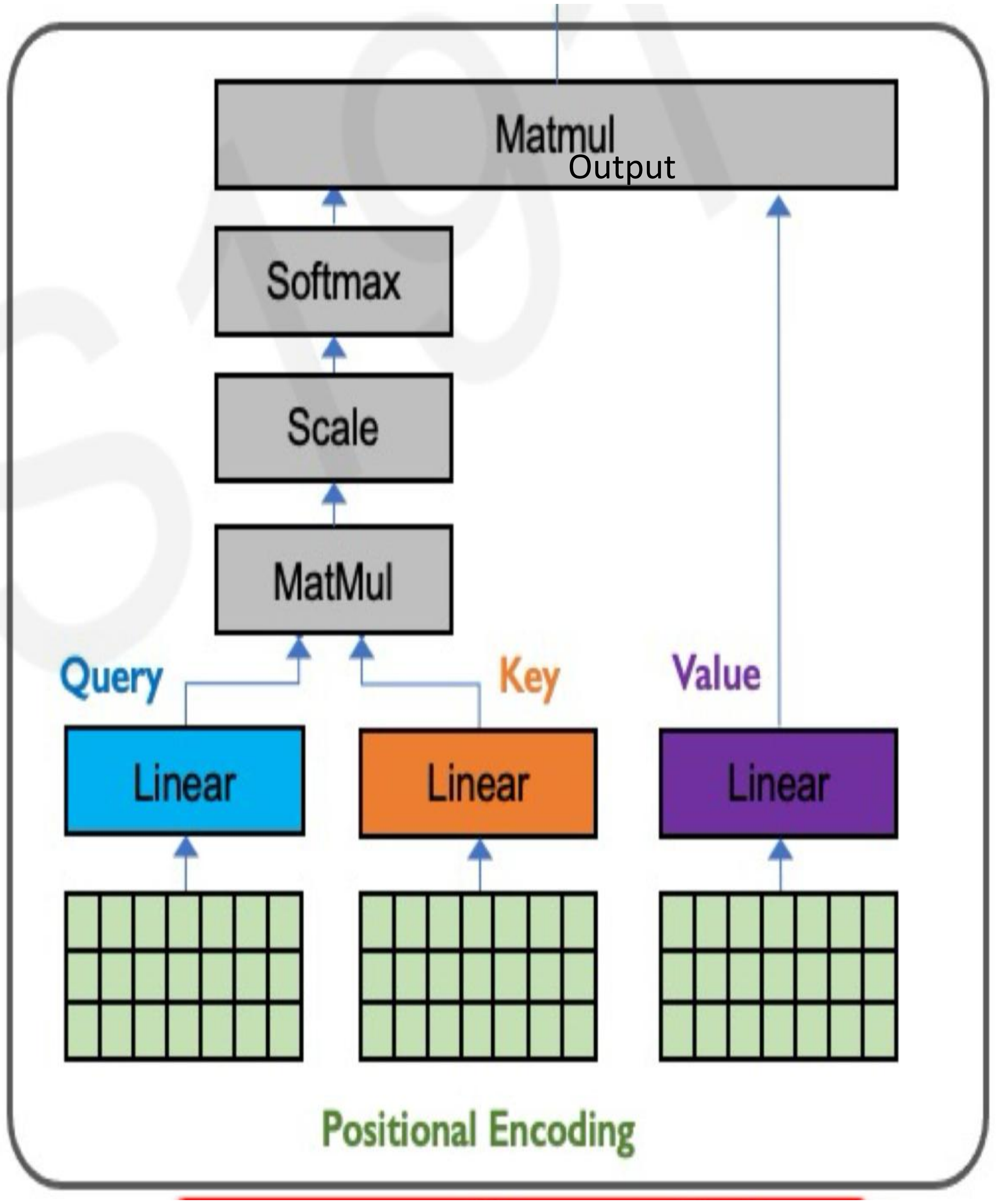
  that are relevant to the semantic meaning of sentence.
- Extract features with high attention.
- Parameters - Query(Q) , Key(k) , Value(v).

Source: https://tinyurl.com/5n6u7jny

Positional encoding

- MatMul- Matrix multiplication
- Scale-scaling
- Softmax-converts a vector of k values into probability distribution

of K probabilities proportional to exponents of input numbers



Reference : https://tinyurl.com/2bn84tkk

Transformers

- Transformers is a deep learning architecture works on parallel multi headed attention mechanisms.

- Common applications of transformers include

  -Natural language processing.

  -Computer vision.

  -GPT(Generative pre-trained transformers)

  -BERT(BI directional encoder representations from transformers

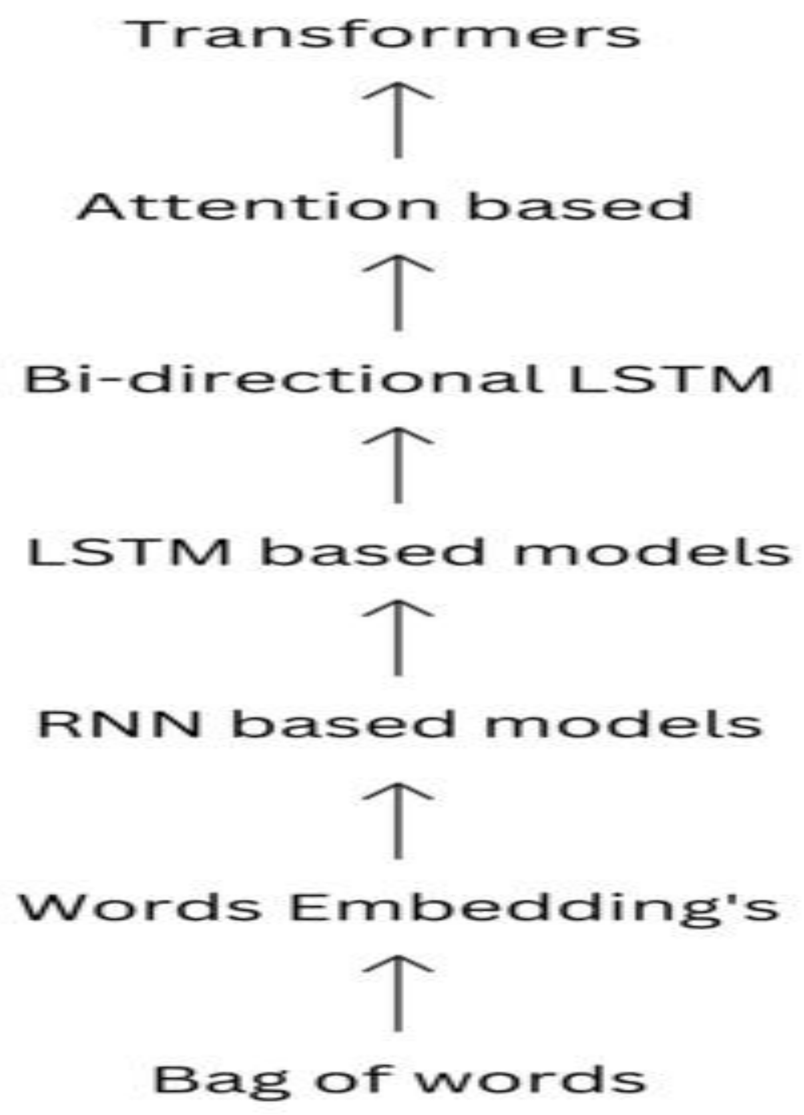
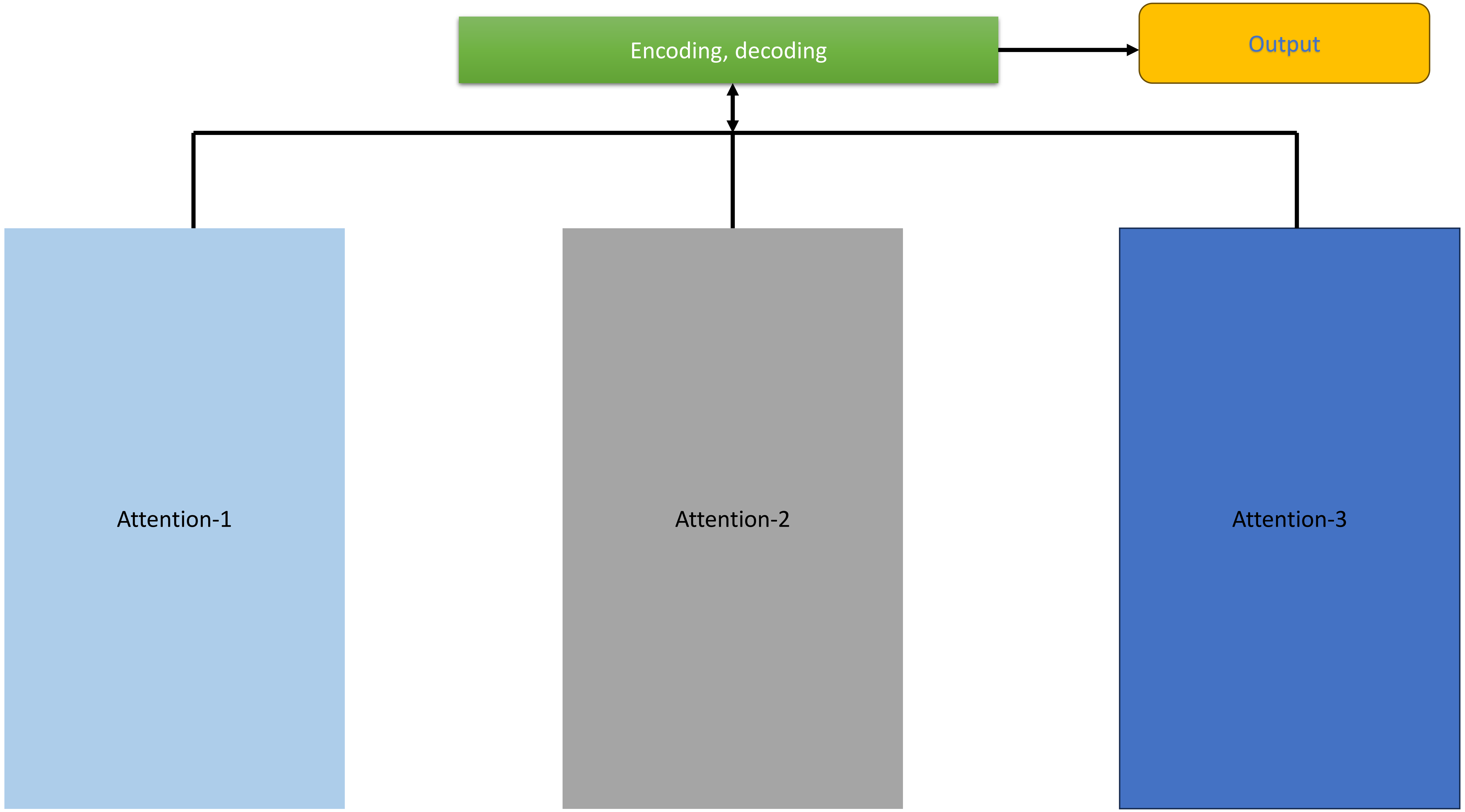


Fig: https://tinyurl.com/yrjhvwhd

Transformers



Reference : https://tinyurl.com/2bn84tkk

LSTM Code snippets

- Importing, training & fitting dataset

```
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout
```

```
regressor.add(LSTM(units=50, return_sequences=True, input_shape=(X_train.shape[1],1)))
regressor.add(Dropout(0.2))

regressor.add(LSTM(units=50, return_sequences=True))
regressor.add(Dropout(0.2))

regressor.add(LSTM(units=50, return_sequences=True))
regressor.add(Dropout(0.2))

regressor.add(LSTM(units=50))
regressor.add(Dropout(0.2))

regressor.add(Dense(units=1))
```

```
regressor.compile(optimizer='adam', loss='mean_squared_error')
regressor.fit(X_train, y_train, epochs=100,batch_size=32)
```

```
dataset_test=pd.read_csv('../input/google-stock-price-train/Google_Stock_Price_Train.csv', ind
```

```
real_stock_price=dataset_test.iloc[:, 1:2].values
```

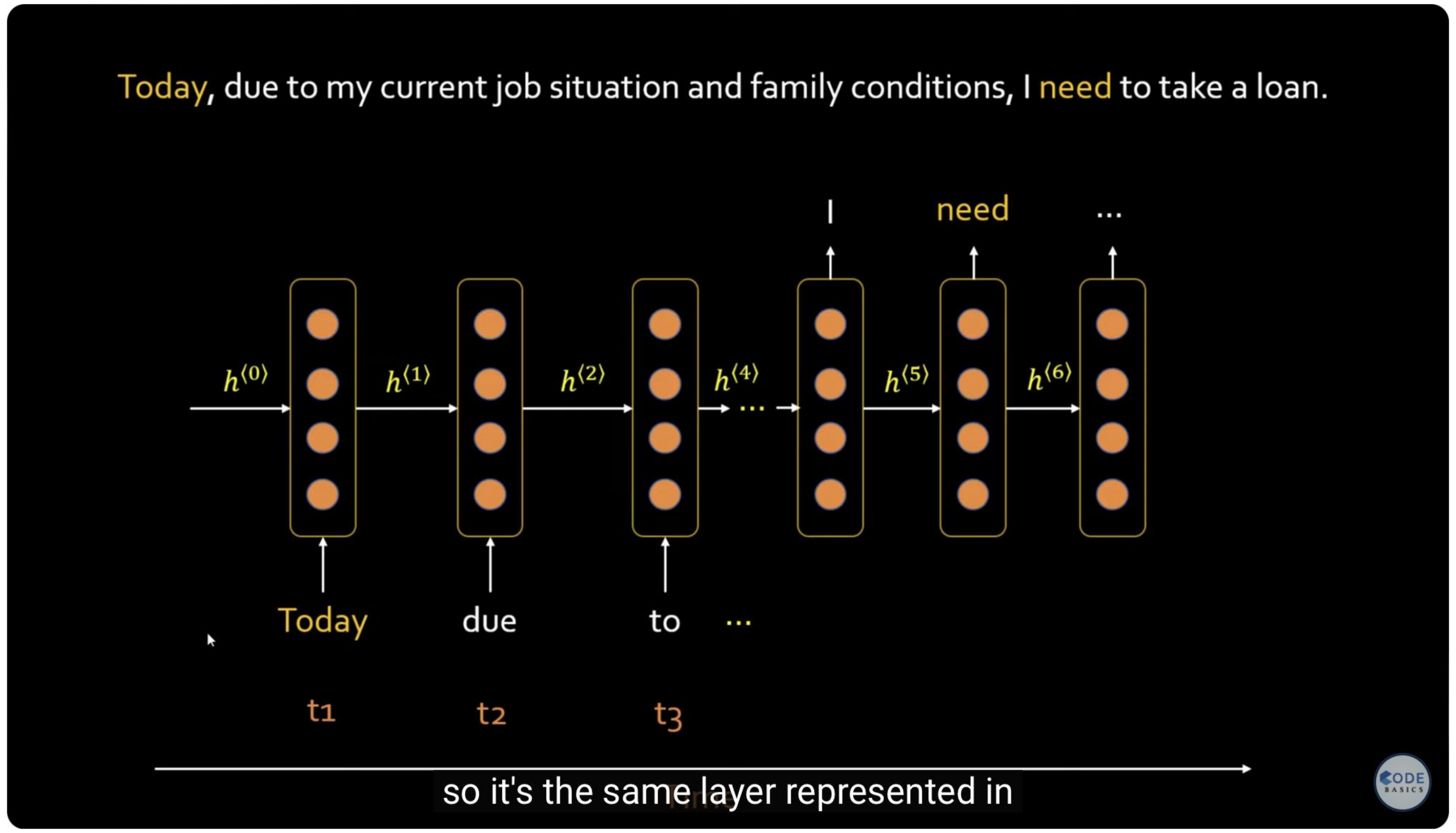
```
plt.plot(real_stock_price, color="red", label="Real stock Price")
plt.plot(pred_price, color="blue", label="Predicted stock price")
plt.title("Google stock price prediction")
plt.xlabel("Time")
plt.ylabel("Google stock price")
plt.legend()
plt.show()
```

LSTM Prediction and Graph of real stock vs predicted stock price

```
plt.plot(real_stock_price, color="red", label="Real stock Price")
plt.plot(pred_price, color="blue", label="Predicted stock price")
plt.title("Google stock price prediction")
plt.xlabel("Time")
plt.ylabel("Google stock price")
plt.legend()
plt.show()
```

Engineering
University of Windsor

A short YouTube tutorial video on RNN



Simple Explanation of LSTM | Deep Learning Tutorial 36 (Tensorflow, Keras & Python)

Reference: https://tinyurl.com/bdhv2npk

References

1. https://tinyurl.com/tsfcmx75
2. Zivkovic, S. (2020, November 14). # 005 RNN - Tackling Vanishing Gradients with GRU and LSTM. Master Data Science. https://tinyurl.com/rp8kfr6s
3. Recurrent Neural Network (RNN) | RNN LSTM Tutorial | Deep Learning Course | Simplilearn. (n.d.). PPT. https://tinyurl.com/3xyrn62w
4. https://tinyurl.com/dp3953za
5. Introduction to lstm -https://tinyurl.com/yc28s5pd
6. Understanding lstm- https://tinyurl.com/yc339ddr
7. Lstm - https://tinyurl.com/dp3953za
8. GRU- https://tinyurl.com/5n933r7c
9. MIT introduction to deep learning- https://tinyurl.com/2bn84tkk
10. Stanford. Edu RNN cheat sheet- https://tinyurl.com/n7djd7ax
11. Transforms Encyclopedia- https://tinyurl.com/yrjhvwhd
12. CODE: https://www.kaggle.com/code/faressayah/stock-market-analysis-prediction-using-lstm


Thank You
DesiComments.com