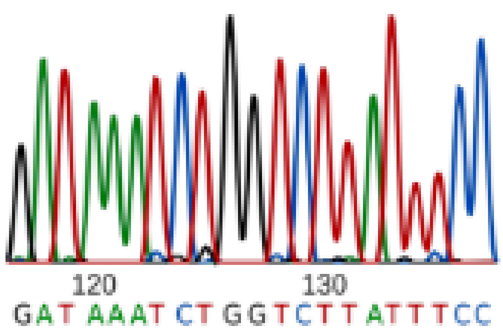
RNN & Regression : Stock Prices Prediction with LSTM

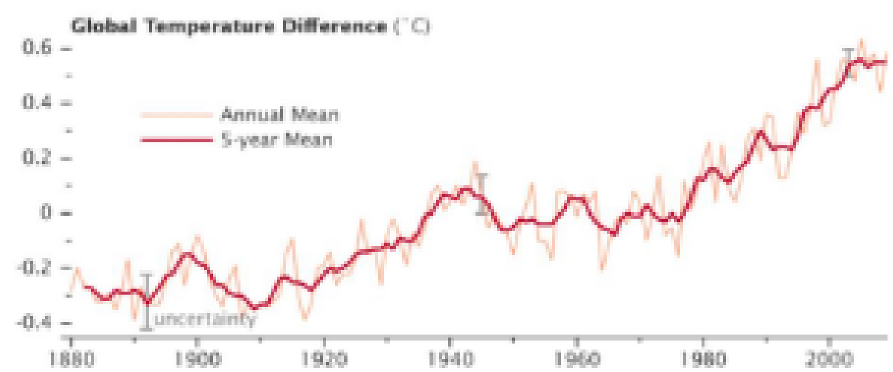
TA. Bogyeong Suh

Sequential data



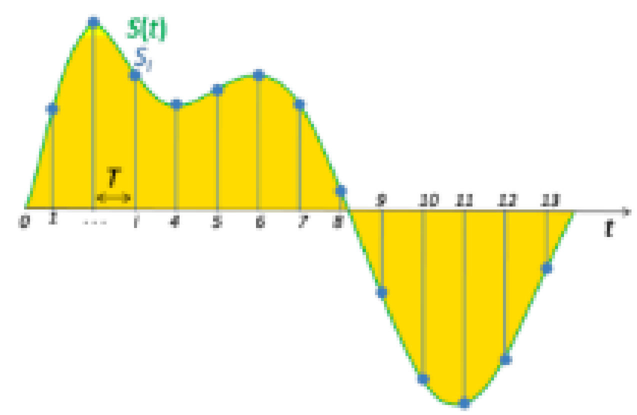
DNA 염기 서열

(Sequential Data)



세계 기온 변화

(Temporal Sequence)



샘플링된 소리 신호

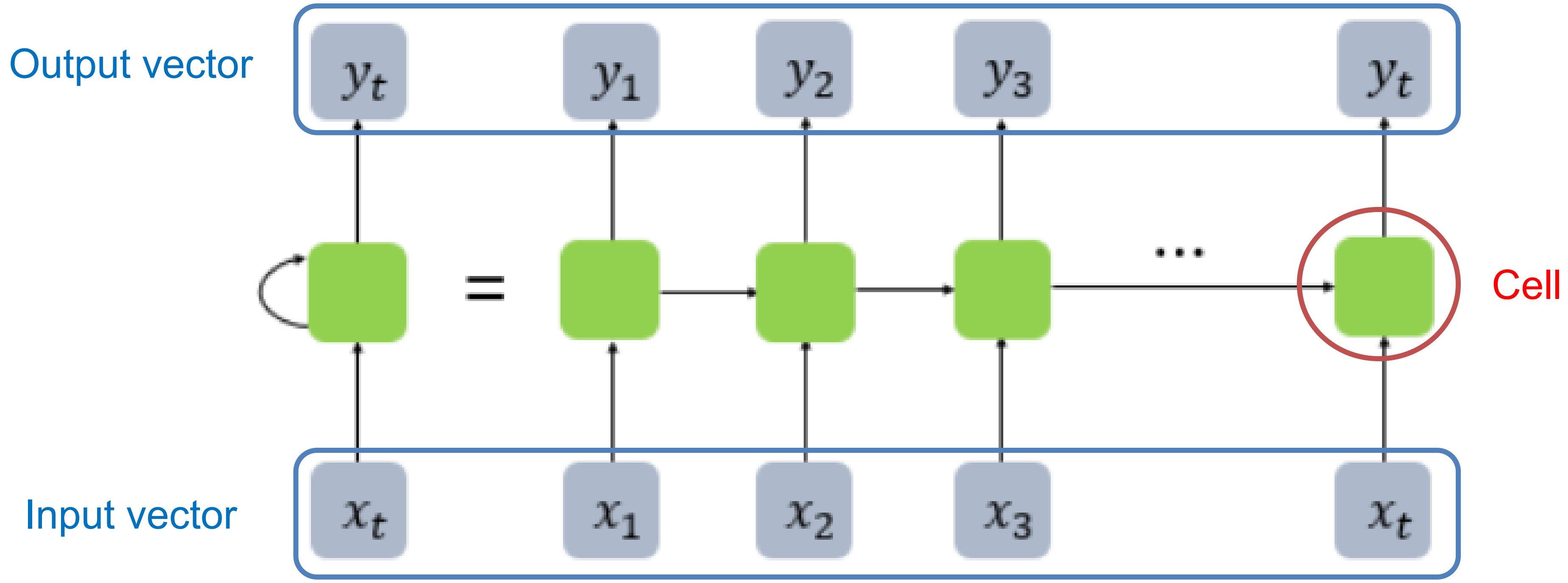
(Time Series)

Sequential data

- Data that contain elements ordered into sequences

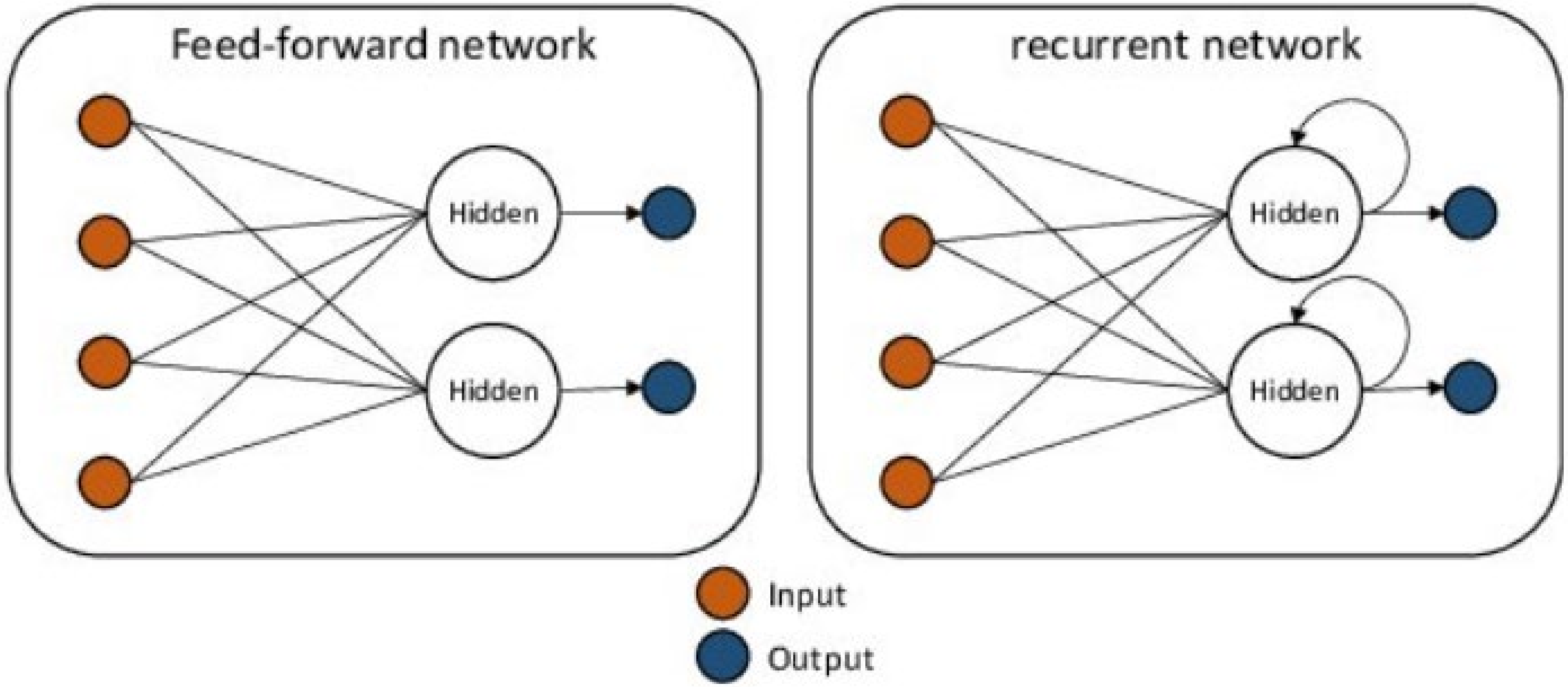
  e.g. Collection of observations obtained through repeated measurements over time

RNN (Recurrent Neural Network)



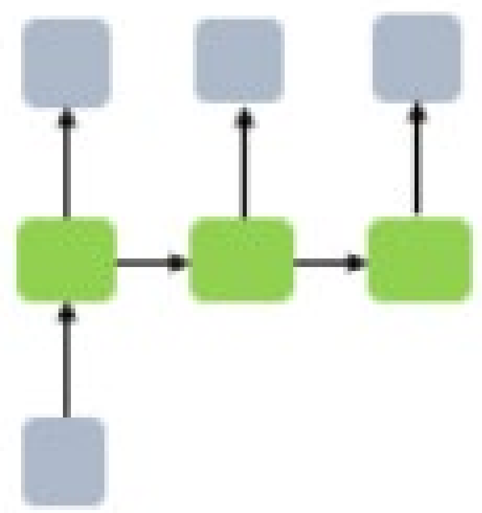
- RNN can recognize data's sequential characteristics and use patterns to predict the next likely scenario
- Connections between nodes can create a cycle, allowing output from some nodes to affect subsequent input to the same nodes
- Hidden state from x_{t-1} is used for calculating x_t (current hidden state)

Feed-forward Neural Network vs RNN

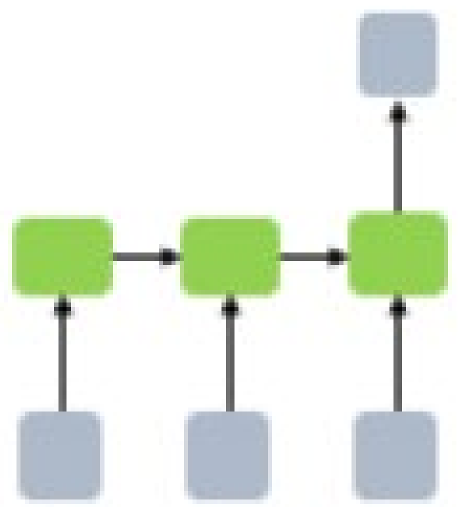


- In feed-forward neural network, information moves in <u>only one direction forward</u>, from the input nodes through the hidden nodes and to the output nodes
- In RNN, networks can have signals traveling in <u>both directions</u> by introducing loops in the network

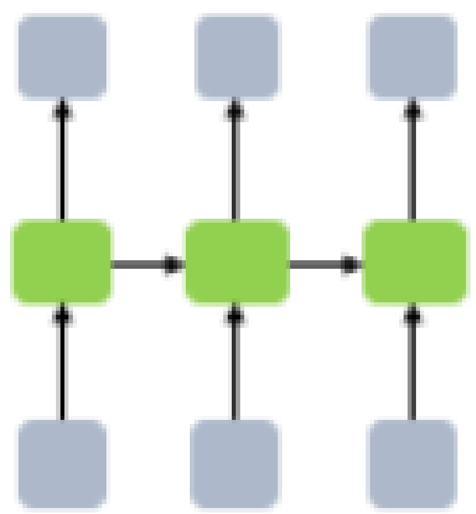
Various types of RNN structure



one-to-many



many-to-one



many-to-many

e.g. Image captioning

- Input: image
- Output: sequences of words

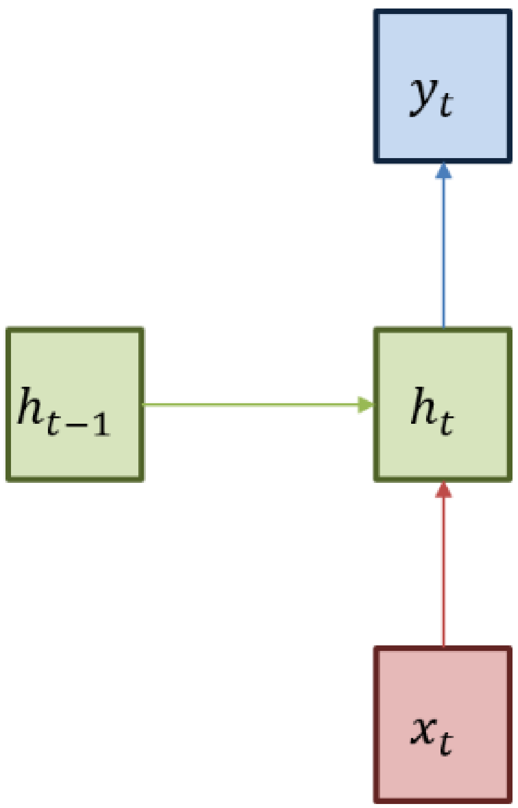
e.g. Spam detection

- Input: sequences of words
- Output: binary classification

e.g. Translation

- Input: sequences of words
- Output: sequences of words

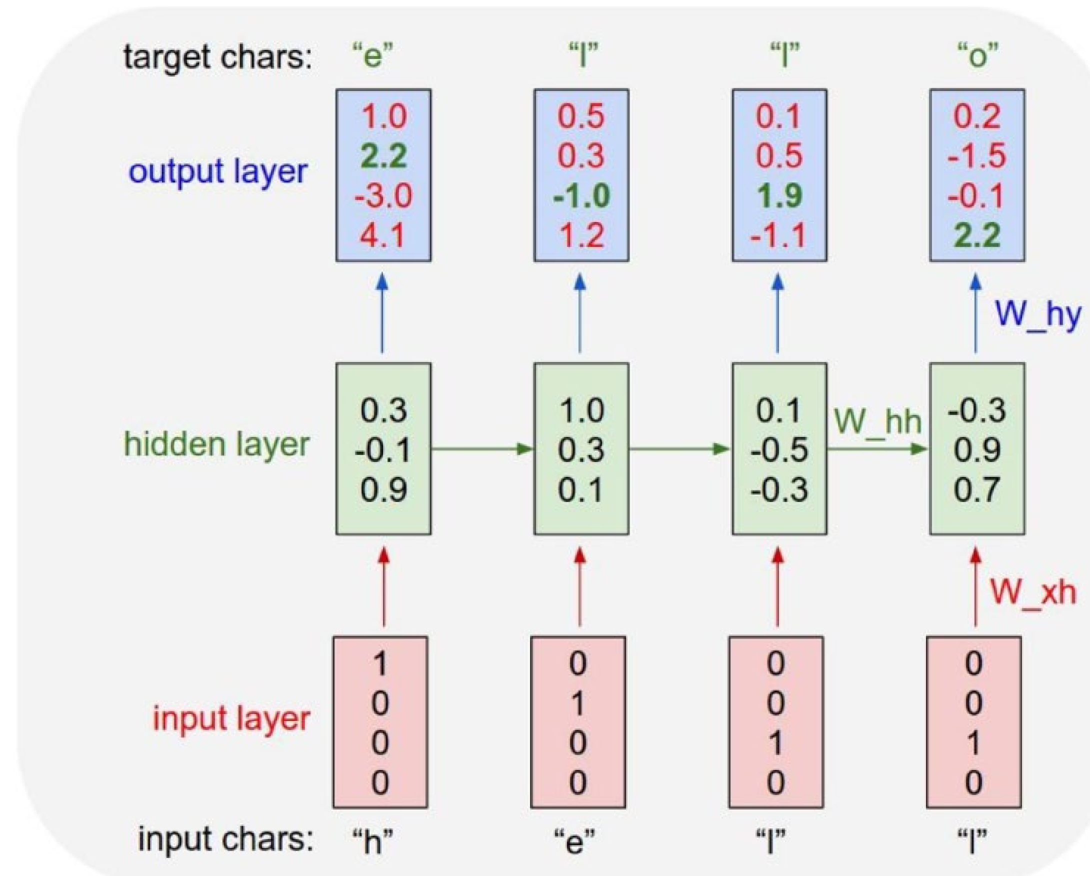
How does RNN work?



$$y_t = f(W_{hy} h_t + by)$$

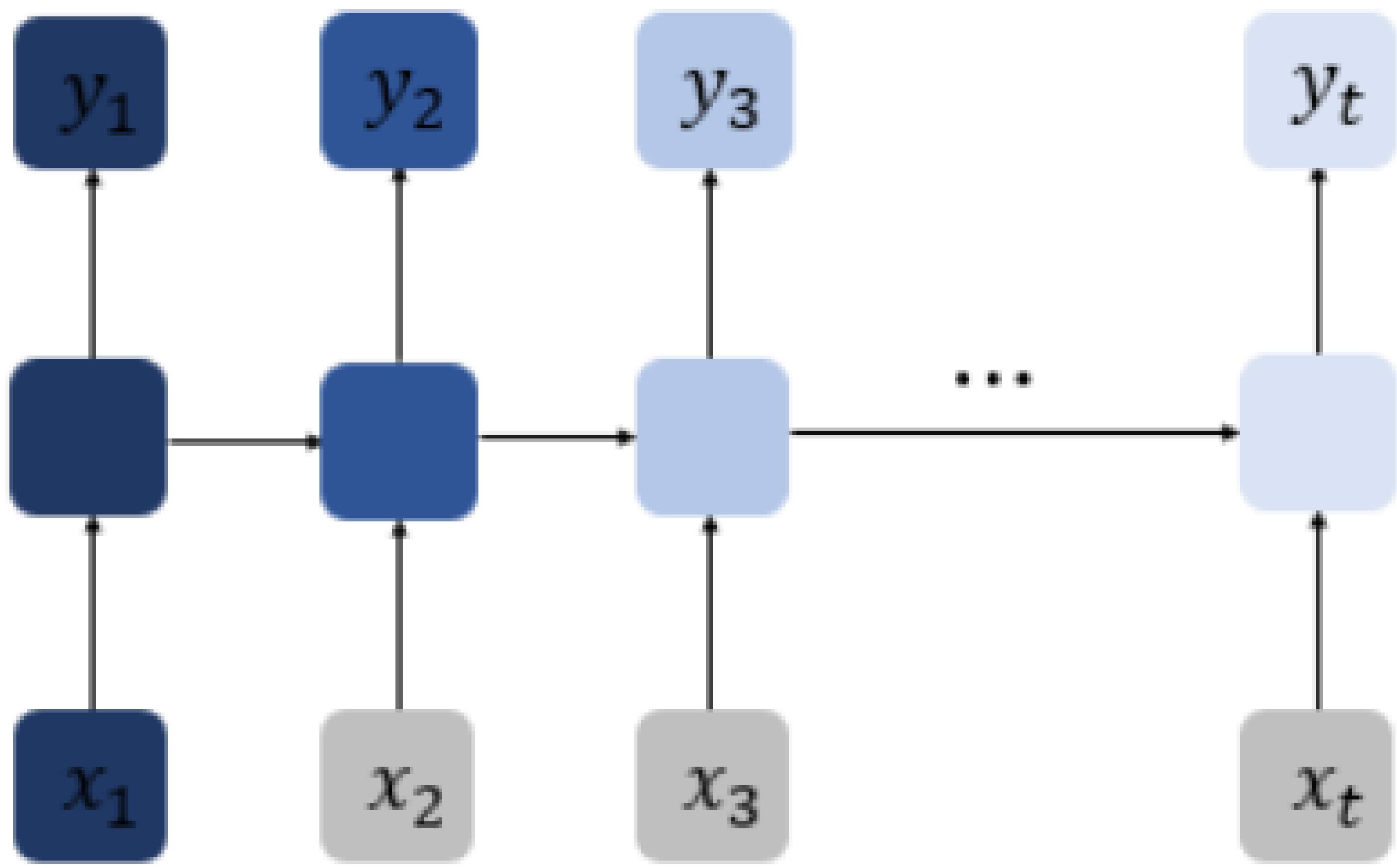
$$h_t = \tanh(W_{hh} h_{t-1} + W_{xh} x_t + b_h)$$

f: nonlinear activation function



- Current hidden state h_t can be updated with former information from h_{t-1} and input x_t

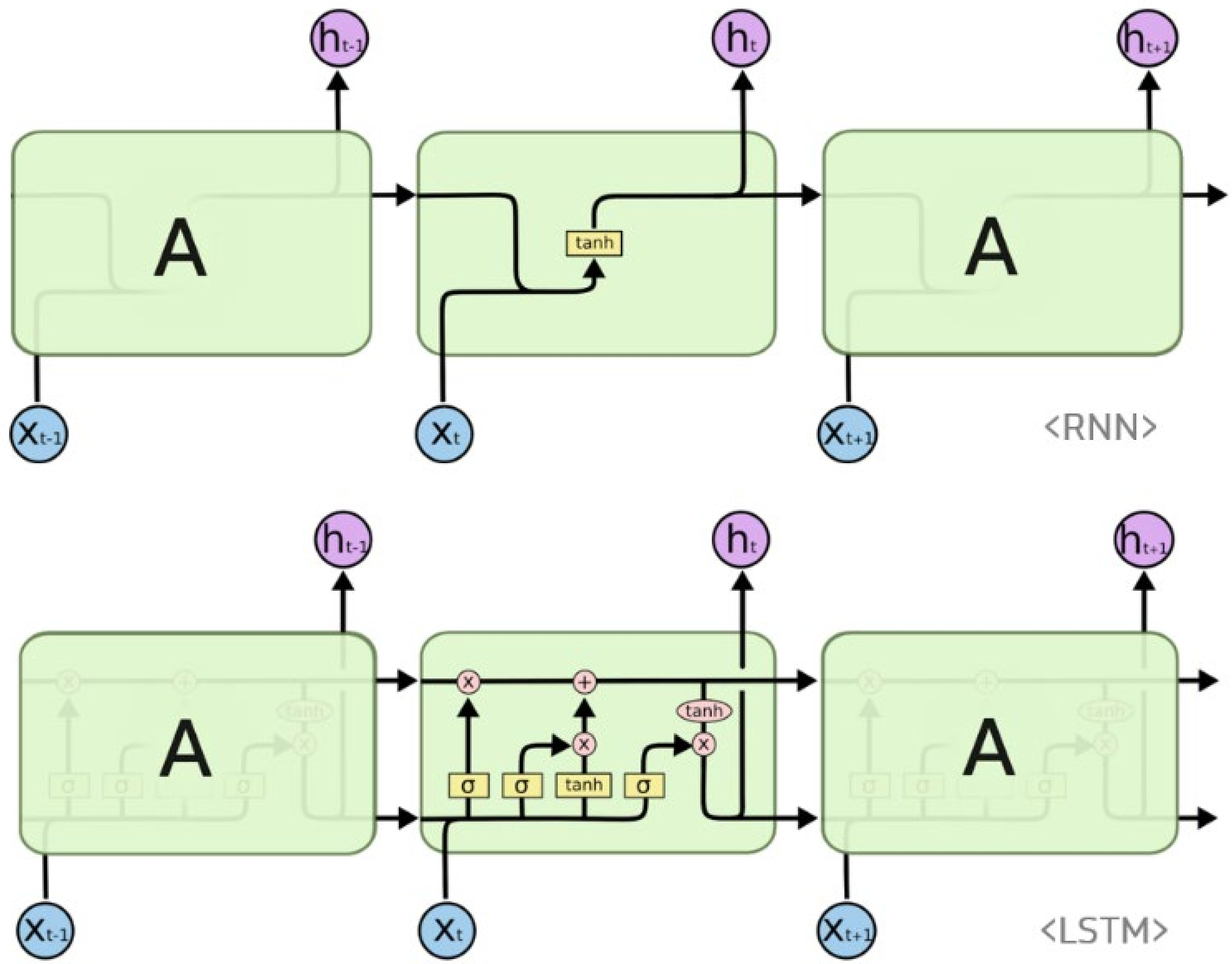
Limitation



Problem of Long-Term Dependencies : vanishing gradients

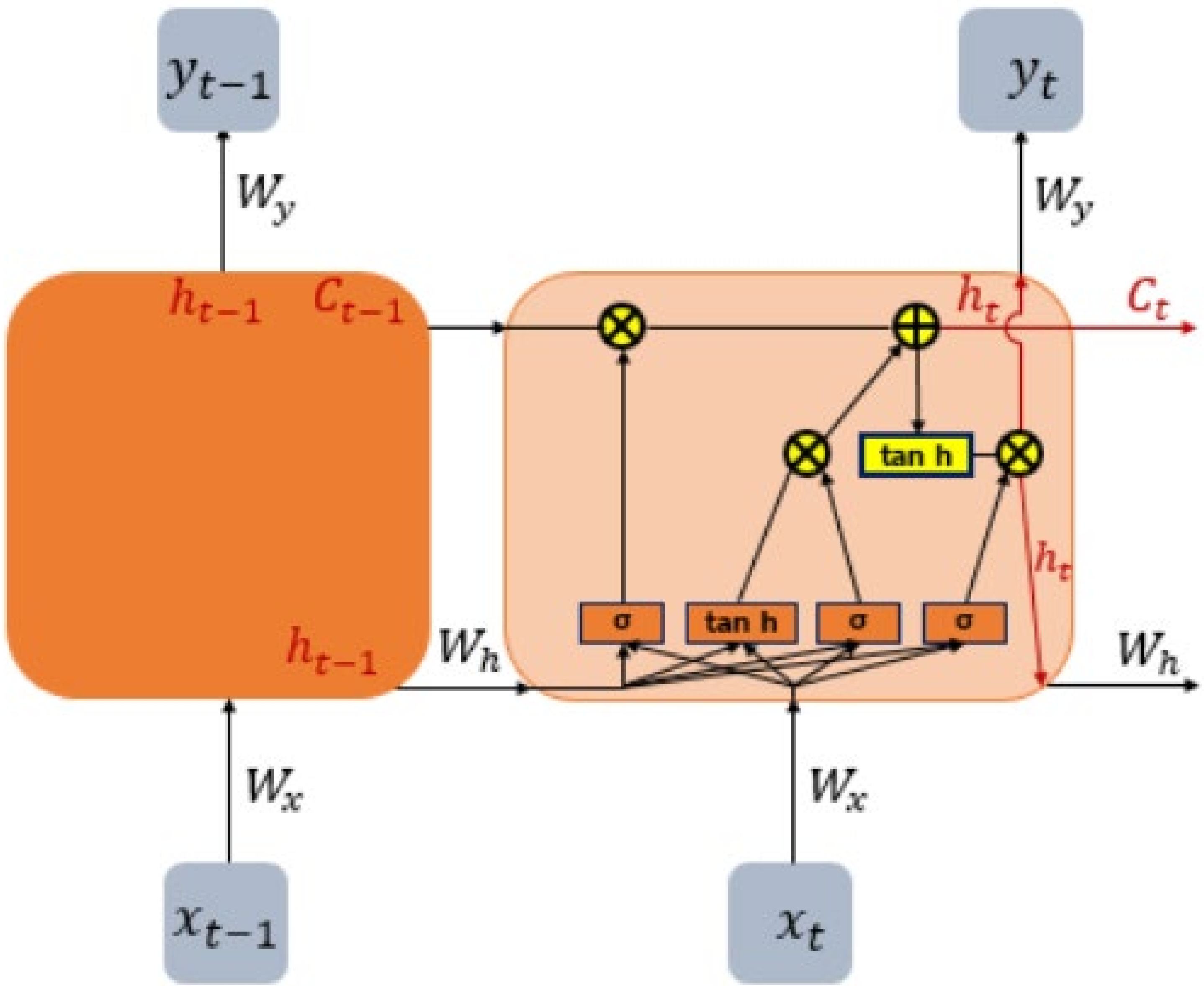
- As time steps increase, gradient become smaller and smaller, thus leading former information to lose impact on hidden state
- What if important information places in former time steps?

LSTM (Long Short-Term Memory)



- In the hidden layer of a LSTM, flow of information into and out of the cell is regulated by gates
- These gates decide what <u>information to discard from a previous state, or to store in the current state</u>

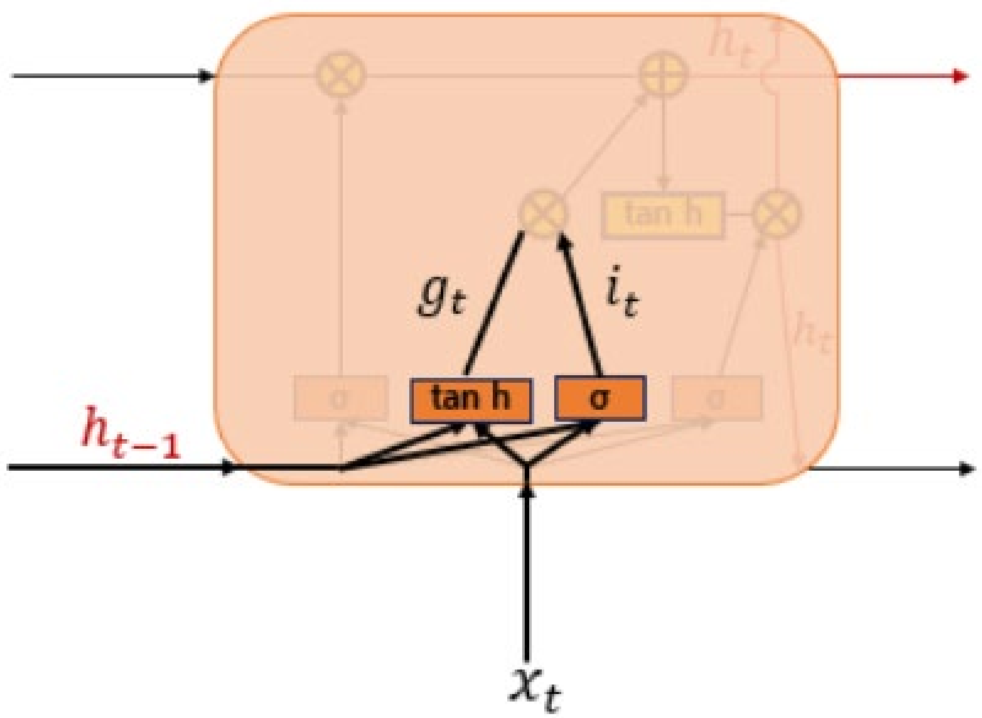
LSTM (Long Short-Term Memory)



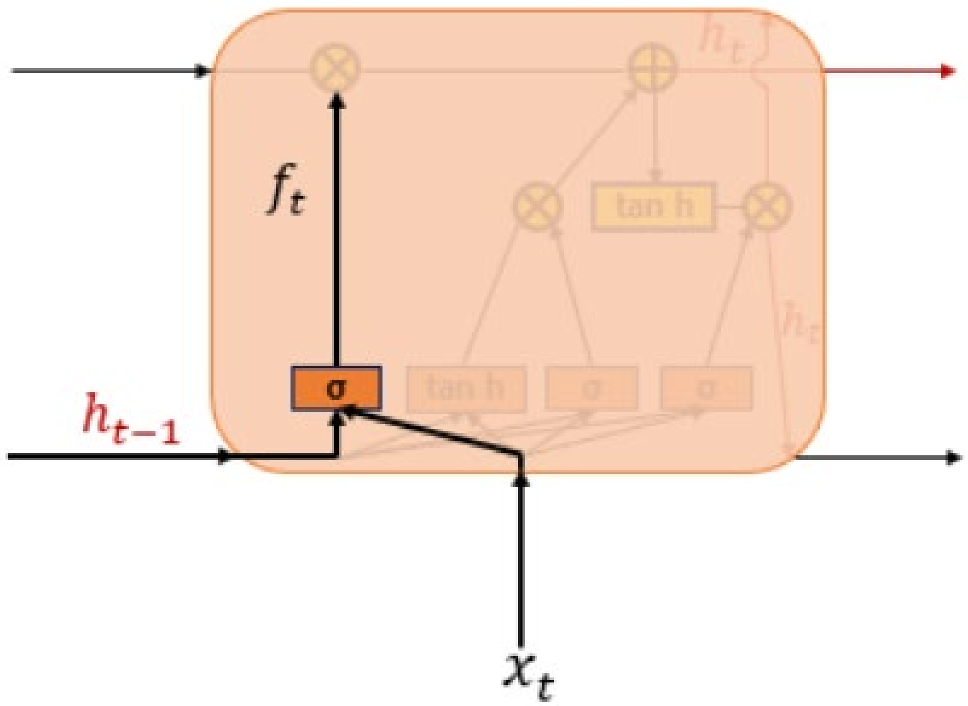
- LSTM is composed of **input gate, output gate, forget gate, and cell state**

Input gate / Forget gate

Input gate



Forget gate

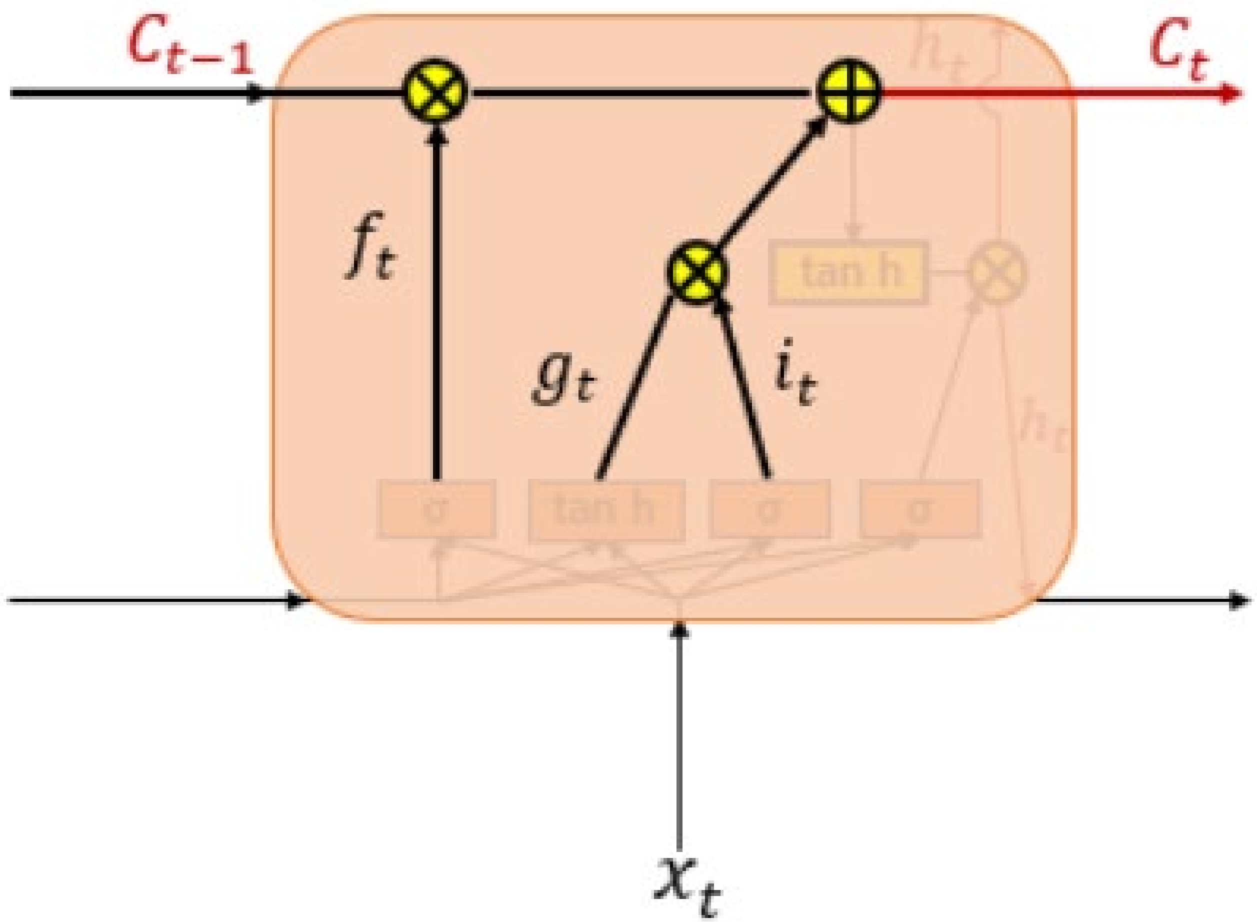


$$g_t = \tanh(W_{xg}x_t + W_{hg}h_{t-1} + b_g)$$
$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i)$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f)$$

- Input gates decide which pieces of new information to store in the current state
- Forget gates decide what information to discard from a previous state
- Both gates make decision by assigning a value between 0 and 1 or -1 and 1

Cell state

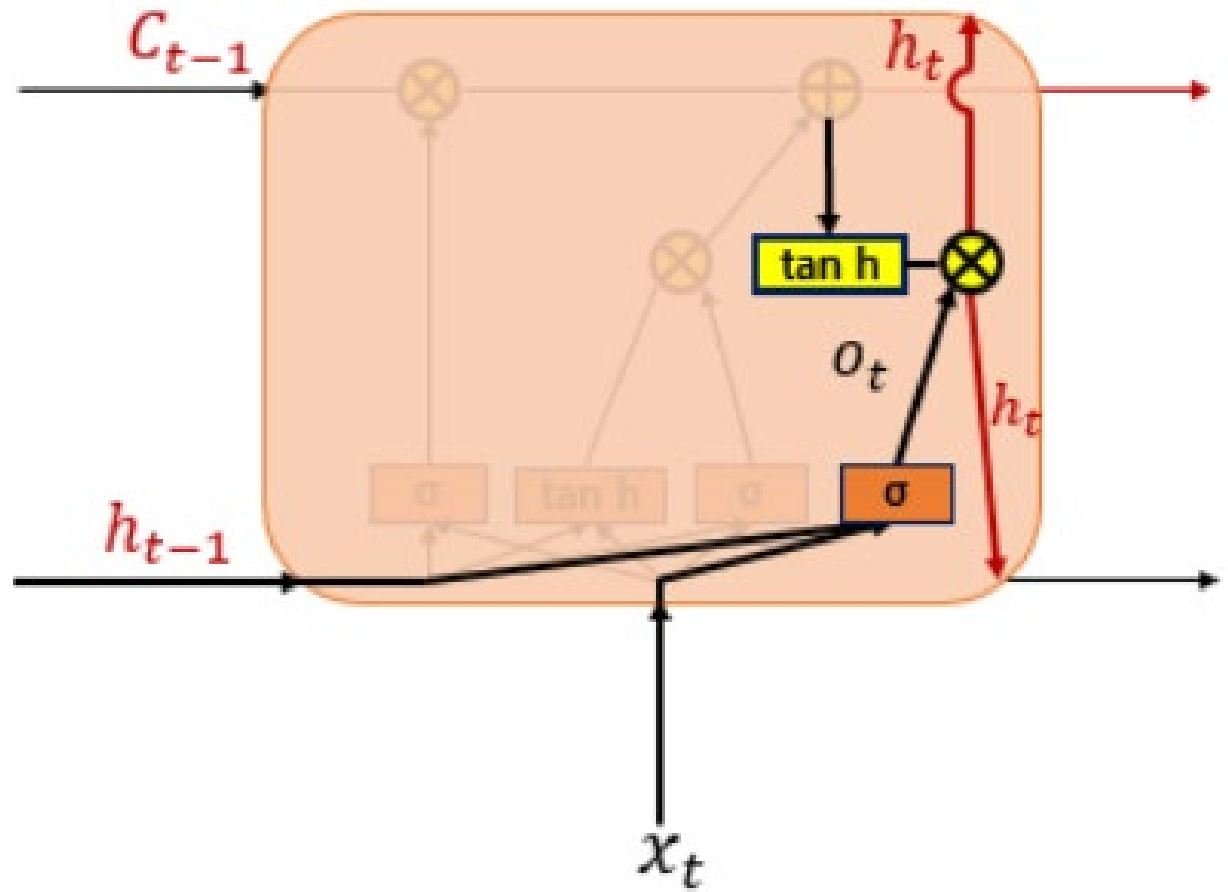


∘ : entrywise product

$$C_t = f_t \circ C_{t-1} + i_t \circ g_t$$

- Cell state can be calculated with previous cell state selected by forget gate, and new information selected by input gate
- If f_t becomes 0, C_{t-1} has no effect on deciding the current state C_t

Output gate



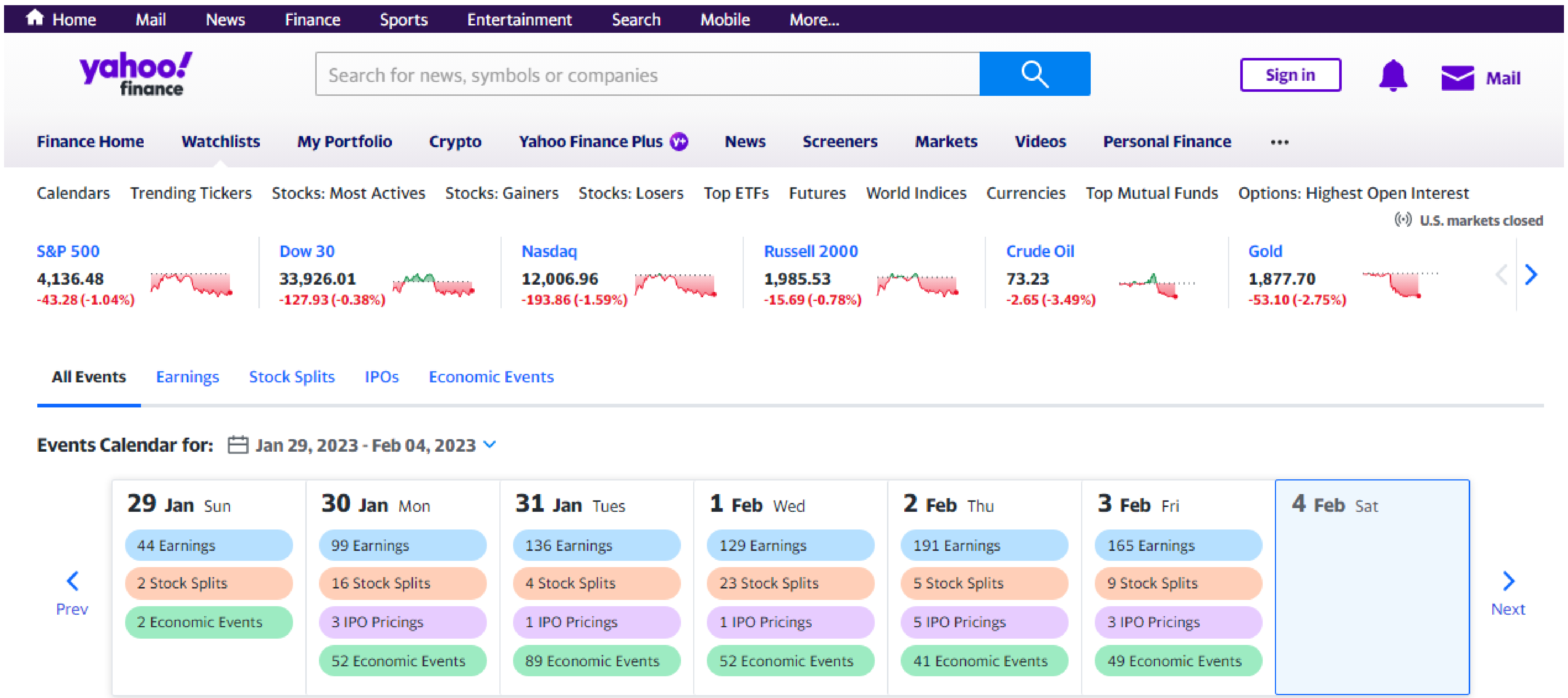
$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o)$$

$$h_t = o_t \circ \tanh(C_t)$$

- Output gates control which pieces of information in the current state to output by assigning a value from 0 to 1 to the information, considering the previous and current states

Stock prices prediction with LSTM

yFinance



- Open-source Python library that allows us to acuiqre stock data from Yahoo Finance
- Consists of stock prices, financial statements, historical data, news, etc
- Specific stocks can be found with ‘ticker’, explained in https://finance.yahoo.com/lookup

Stock prices prediction with LSTM

연세대학교
YONSEI UNIVERSITY

1. Load

- yFinance can be easily loaded with python library
- We can use 'pandas_datareader' from 'pandas' library to read a tabular data into a dataframe

```
import yfinance as yf
from pandas_datareader import data as pdr
yf.pdr_override()
df = pdr.get_data_yahoo('SBUX', start='2018-01-01', end='2022-12-31')
```

Ticker for Starbucks Corporation

Symbol	Name	Last Price	Change	% Change
SBUX	Starbucks Corporation	109.15	-0.84	-0.76%

2. Data preprocessing

- Make training and test data from the dataframe
- Sample code makes data based on '<u>many-to-one</u>', with one feature at each time step
- Normalize : MinMaxScaler, StandardScaler from Scikit-Learn

Date	Adj Close
2020-01-05	257.31
2020-01-06	258.48
2020-01-07	266.38
2020-01-08	267.94
2020-01-09	272.73
2020-01-10	271.90
2020-01-11	265.55

$$x_{t-4} \quad \cdots \quad x_{t-2} \quad x_{t-1} \quad x_t \quad y_{t+1}$$

$$x_{t-4} \quad \cdots \quad x_{t-2} \quad x_{t-1} \quad x_t \quad y_{t+1}$$

$$x_{t-4} \quad \cdots \quad x_{t-2} \quad x_{t-1} \quad x_t \quad y_{t+1}$$

Inputs Output

Dataset

3. Train a LSTM model

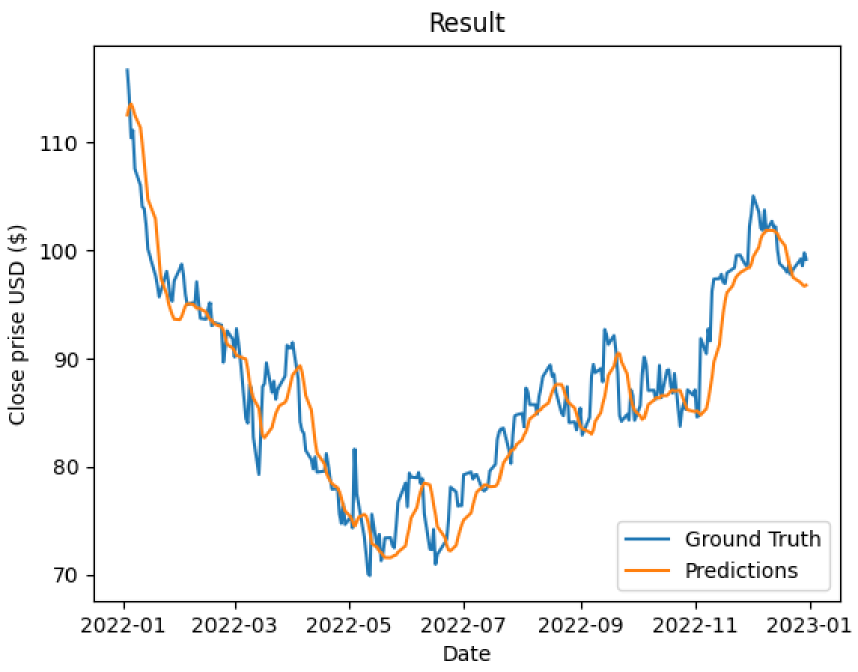
- Use tensorflow.keras
  1) Sequential model
  2) LSTM layer
  3) Dense layer
  4) Which optimizer?
  5) Performance of a regression model can be evaluated with RMSE(root mean square error)

4. Plot the result

- Use matplotlib.pyplot to plot the prediction result from the trained model

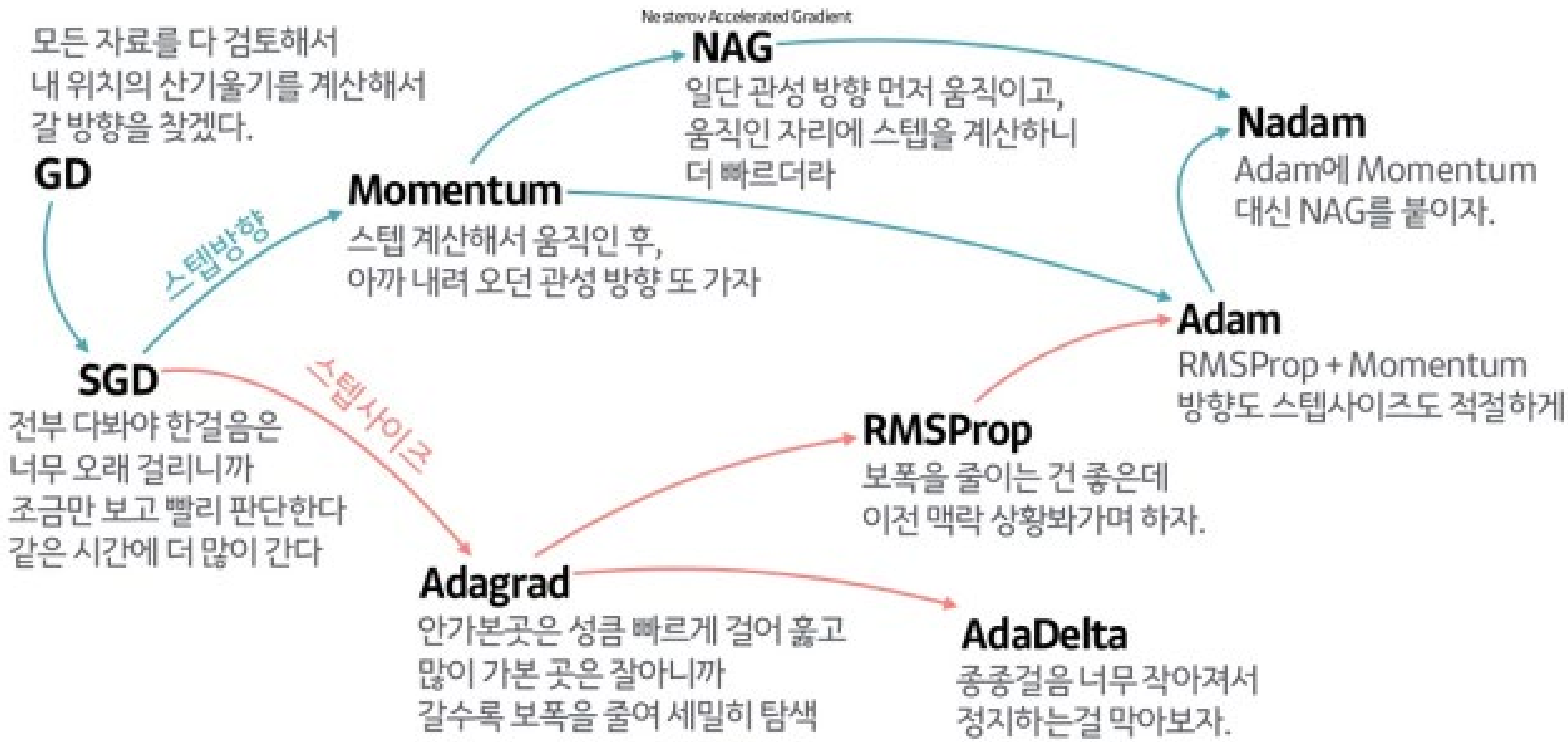
```python
model = Sequential()
model.add(LSTM(units=50, return_sequences=True, input_shape=(x_train.shape[1], 1)))
model.add(LSTM(50, return_sequences=False))
model.add(Dense(25))
model.add(Dense(1))
model.summary()
model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(x_train, y_train, batch_size=bs, epochs=epoch)

predictions = model.predict(x_test)
predictions = scaler.inverse_transform(predictions)
rmse = np.sqrt(np.mean(predictions - y_test)**2)
print(rmse)
```



Stock prices prediction with LSTM

Optimizer



- Optimizers are algorithms or methods used to change the attributes of the neural network such as weights and learning rate in order to reduce the losses