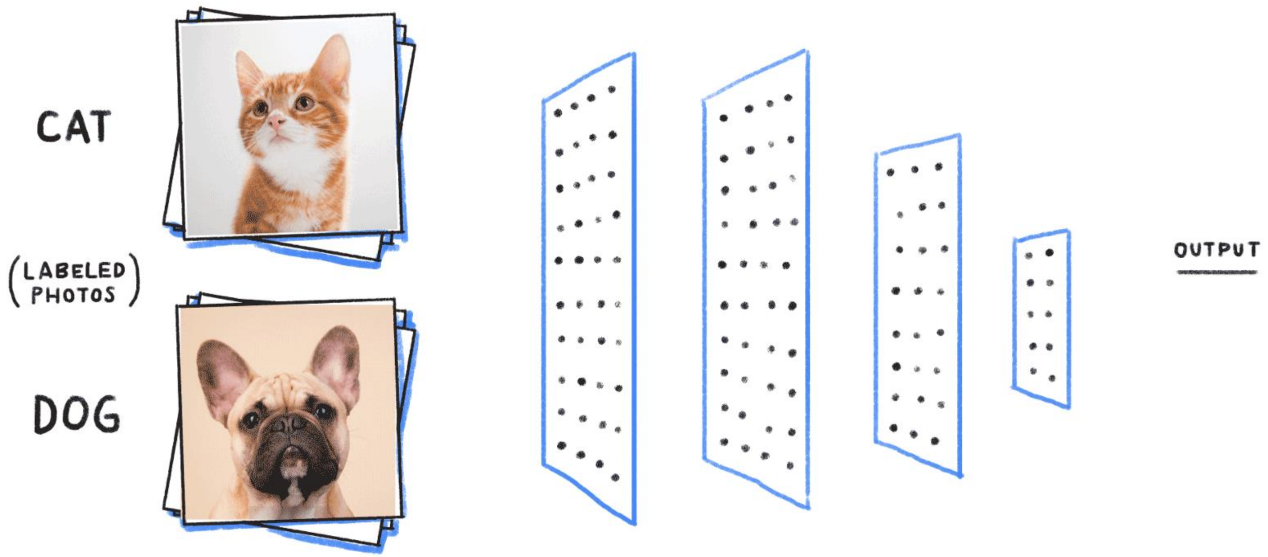
진동신호 분석을 위한 순환 신경망 (RNN)

Prof. Seungchul Lee
Industrial AI Lab.

So Far

- Regression, Classification, Dimension Reduction,
- Based on snapshot-type data



Sequence Matters

(Deterministic) Time Series Data

- For example

$$y[0] = 1, \quad y[1] = \frac{1}{2}, \quad y[2] = \frac{1}{4}, \quad \cdots$$

- Closed-form

$$y[n] = \left(\frac{1}{2}\right)^n, \quad n \geq 0$$

- Linear difference equation (LDE) and initial condition

$$y[n] = \frac{1}{2} y[n-1], \quad y[0] = 1$$

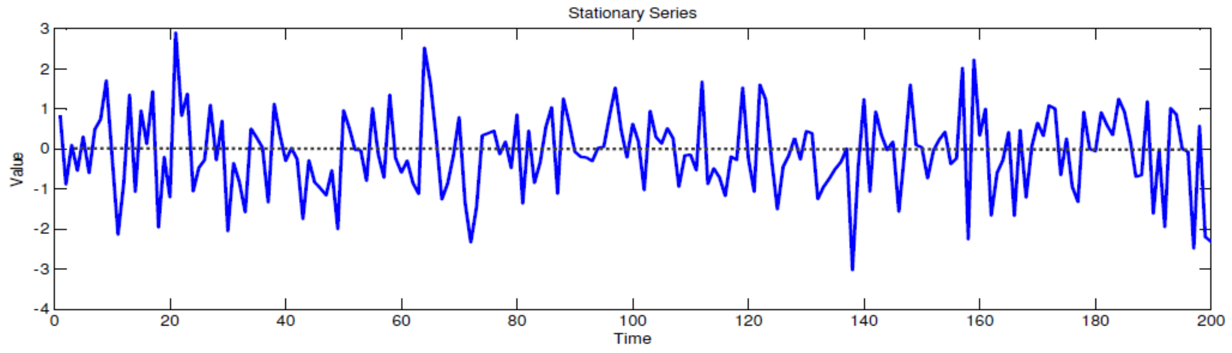
- High order LDEs

$$y[n] = \alpha_1 y[n-1] + \alpha_2 y[n-2]$$

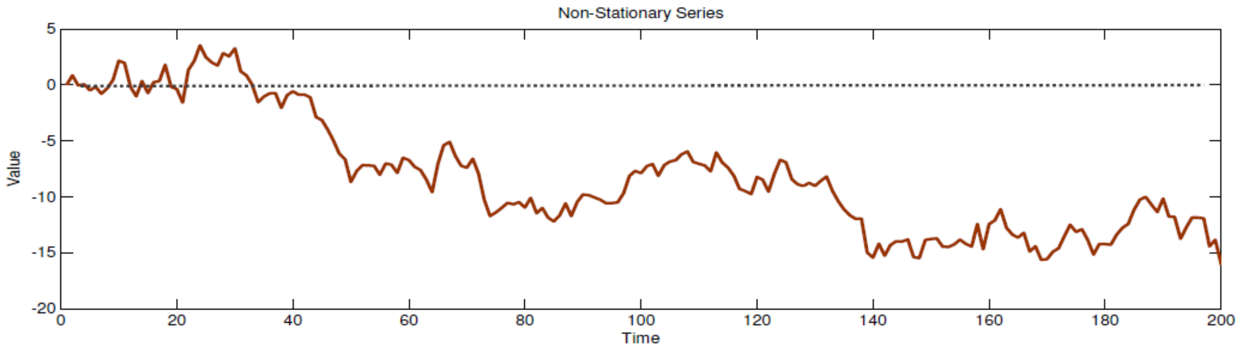
$$y[n] = \alpha_1 y[n-1] + \alpha_2 y[n-2] + \cdots + \alpha_k y[n-k]$$

(Stochastic) Time Series Data

- Stationary



- Non-stationary
  - Mean and variance change over time



Dealing with Non-Stationarity

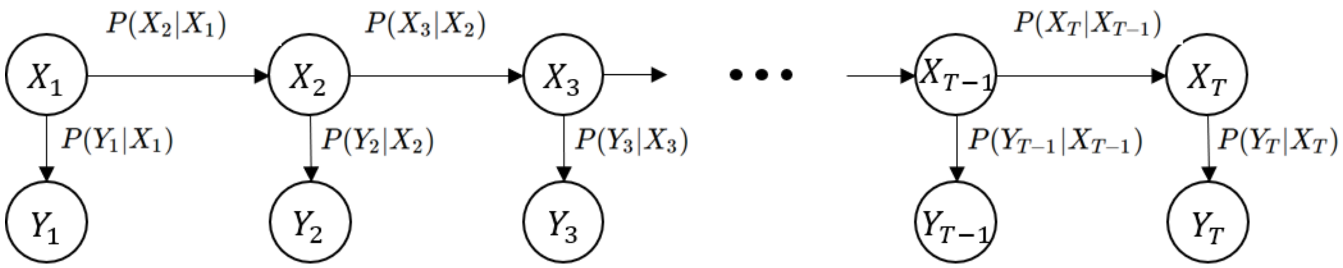
- Model assumption

$$
\begin{aligned}
Y_t = {} & \beta_1 + \beta_2 Y_{t-1} \\
& + \beta_3 t + \beta_4 t^{\beta_5} \\
& + \beta_6 \sin \frac{2\pi}{s} t + \beta_7 \cos \frac{2\pi}{s} t \\
& + u_t
\end{aligned}
$$

Hidden Markov Model

Hidden Markov Models

- Discrete state-space model
  - Used in speech recognition
  - State representation is simple
  - Hard to scale-up the training



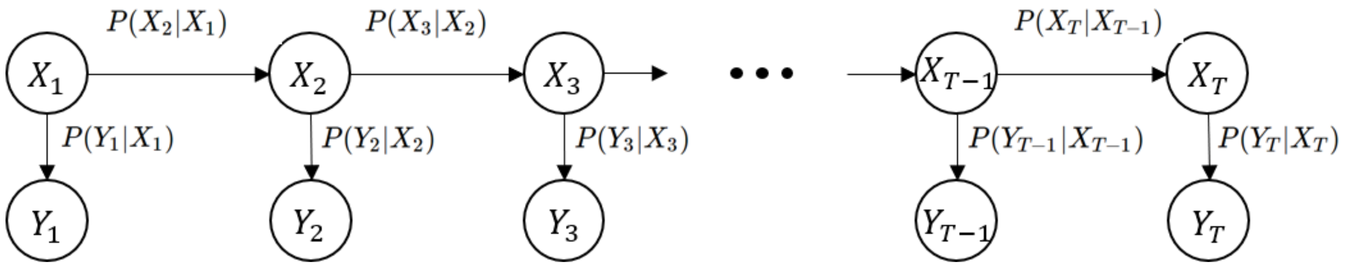
- Assumption
  - We can observe something that's affected by the true state
  - Natural way of thinking

- Limited sensors (incomplete state information)
  - But still partially related

- Noisy sensors
  - Unreliable

Hidden Markov Model (HMM)

- True state (or hidden variable) follows Markov chain
- Observation emitted from state
  - Y_t is noisily determined depending on the current state X_t



- Forward: sequence of observations can be generated
- Question: state estimation

$$P(X_T = s_i \mid Y_1 Y_2 \cdots Y_T)$$

- HMM can do this, but with many difficulties

Neural Network Architectures for Time-Series: Recurrent Neural Network (RNN)

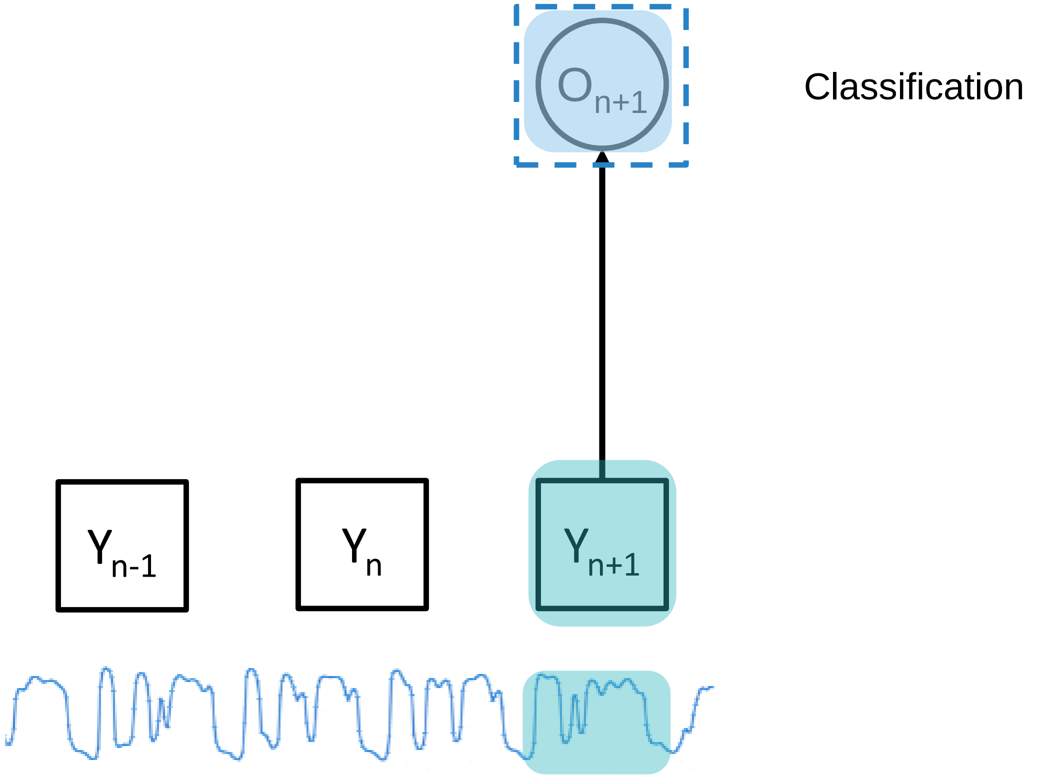
Prof. Seungchul Lee

Industrial AI Lab.

Most slides from CMU 11-785 Intro. to Deep Learning by Prof. Bhiksha Raj

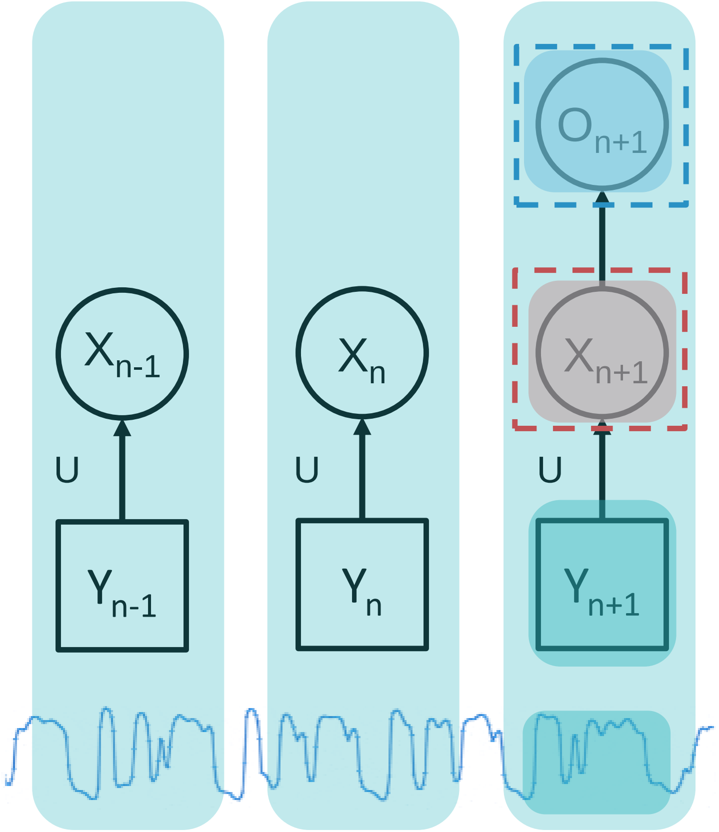
Recurrent NN (RNN)

- Hidden state extraction and transformation



Recurrent NN (RNN)

- Hidden state extraction and transformation

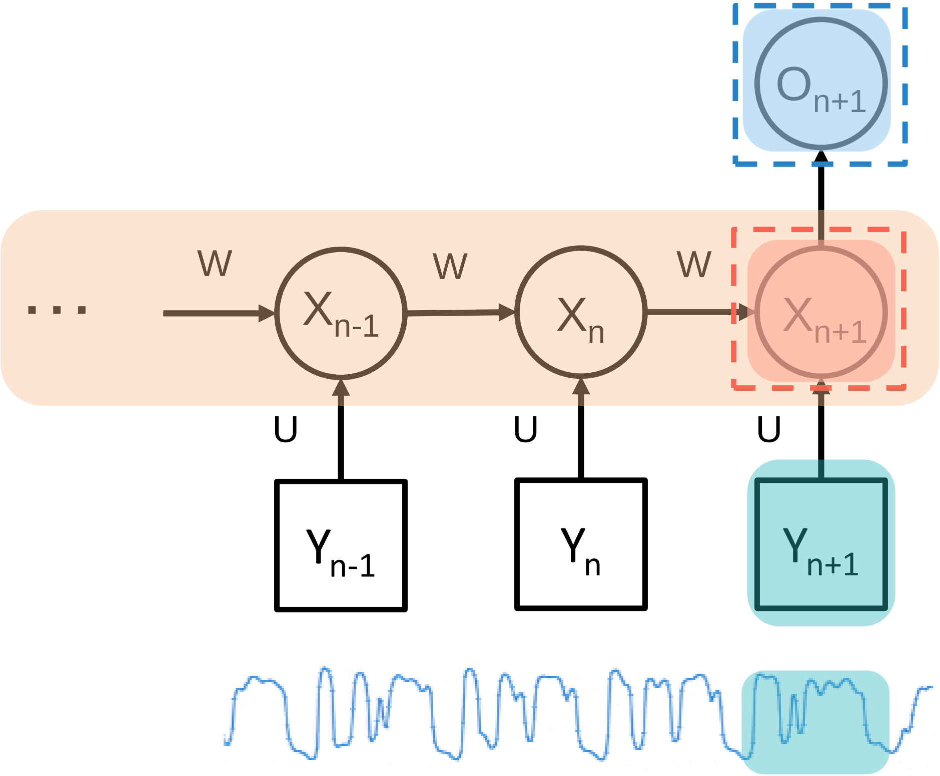


Classification based on states

Learned latent state

Recurrent NN (RNN)

- Hidden state extraction and transformation
- Good for sequential data (dynamic behavior)



Classification based on states

Learned latent state and its dynamics

POSTECH

Recurrent NN

- Recurrence
  - Consider the classical form of a dynamical system:

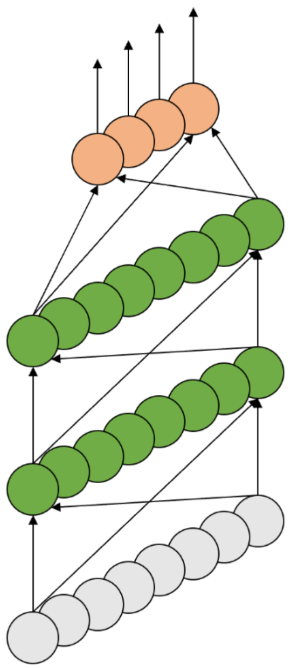
$$s^{(t)} = f(s^{(t-1)}; \theta)$$

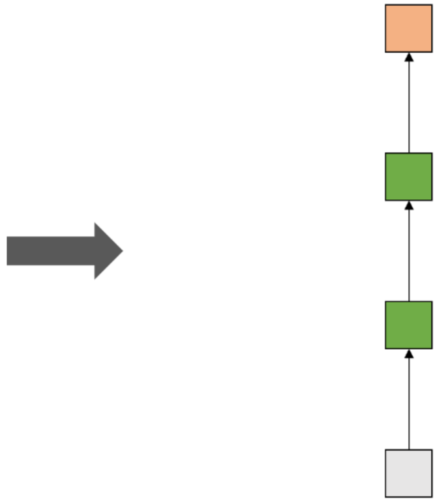
  - This is recurrent because the definition of s at time t refers back to the same definition at time $t-1$

- Hidden state representation

- Learn both from sequential data

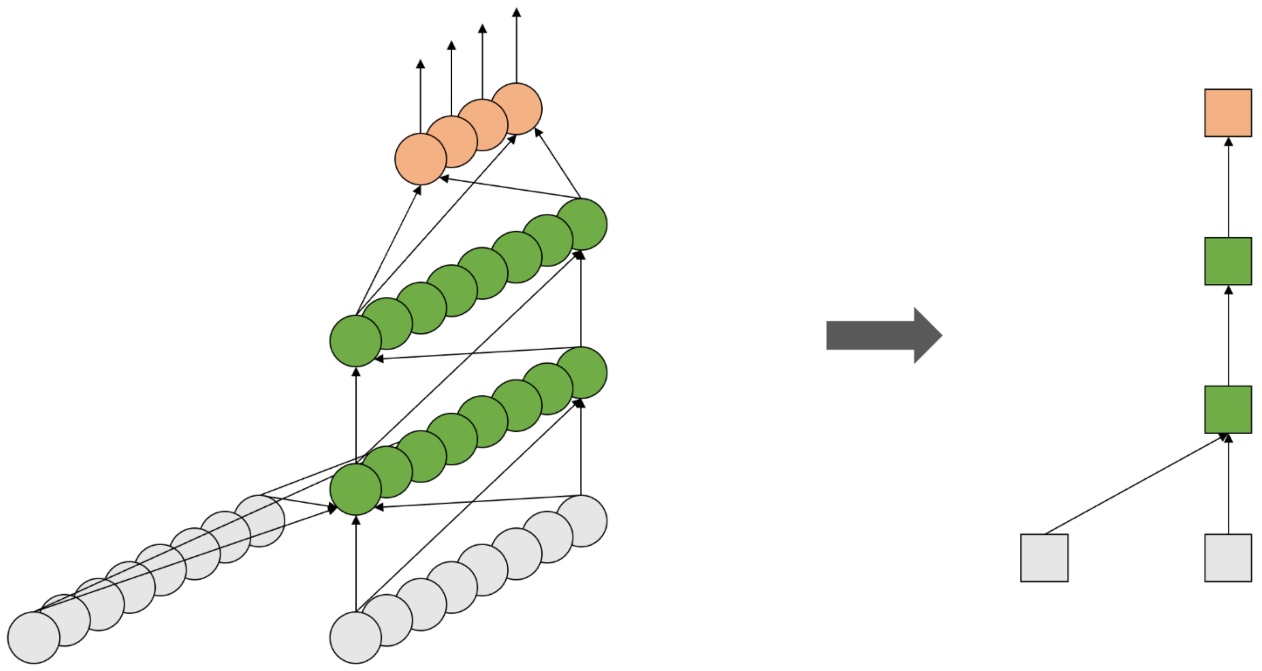
Representation Shortcut





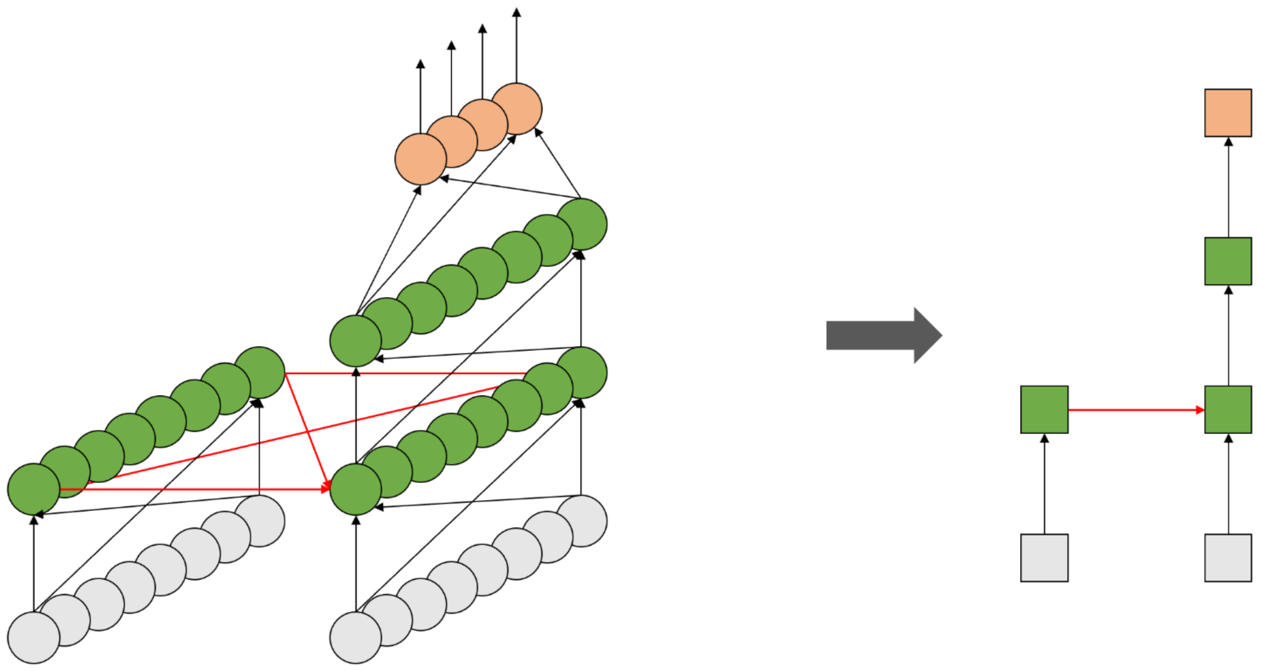
- Input at each time is a vector
- Each layer has many neurons
  - Output layer too may have many neurons
- But will represent everything simple boxes
  - Each box actually represents an entire layer with many units

Representation Shortcut



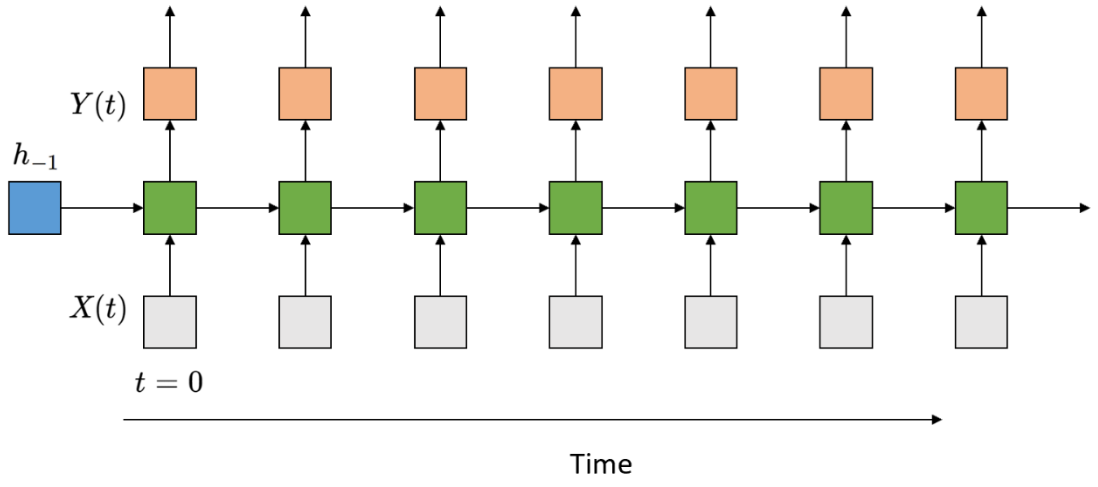
- Input at each time is a vector
- Each layer has many neurons
  – Output layer too may have many neurons
- But will represent everything simple boxes
  – Each box actually represents an entire layer with many units

Representation Shortcut



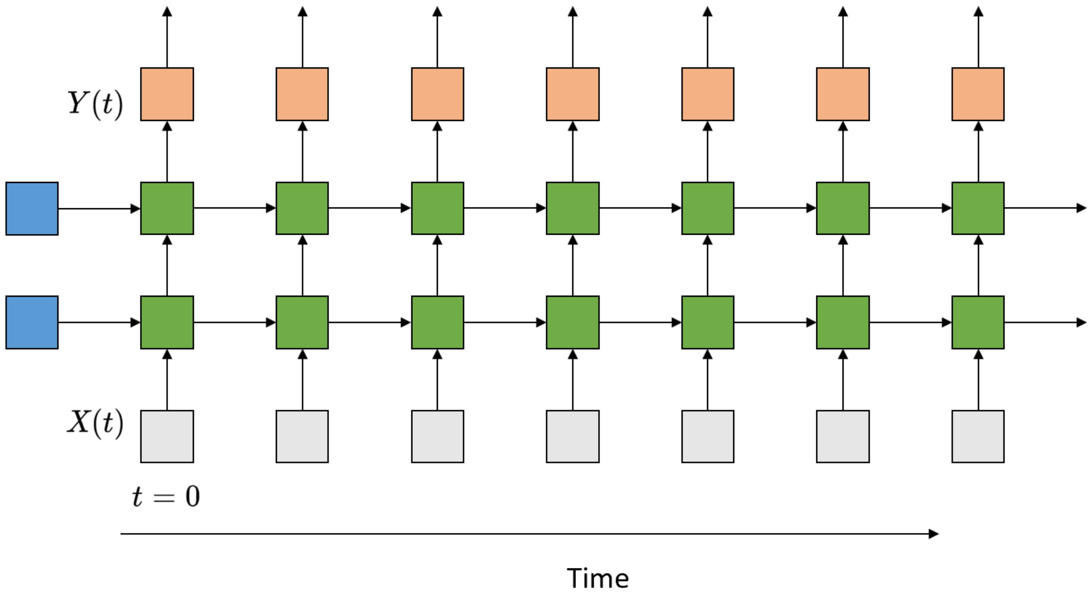
- Input at each time is a vector
- Each layer has many neurons
  - Output layer too may have many neurons
- But will represent everything simple boxes
  - Each box actually represents an entire layer with many units

Single Hidden Layer RNN (Simplest State-Space Model)



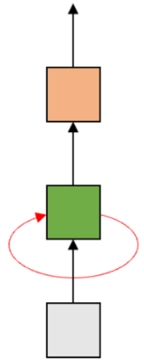
- The state (green) at any time is determined by the input at that time, and the state at the previous time
- All columns are identical
- An input at $t = 0$ affects outputs forever
- Also known as a recurrent neural net

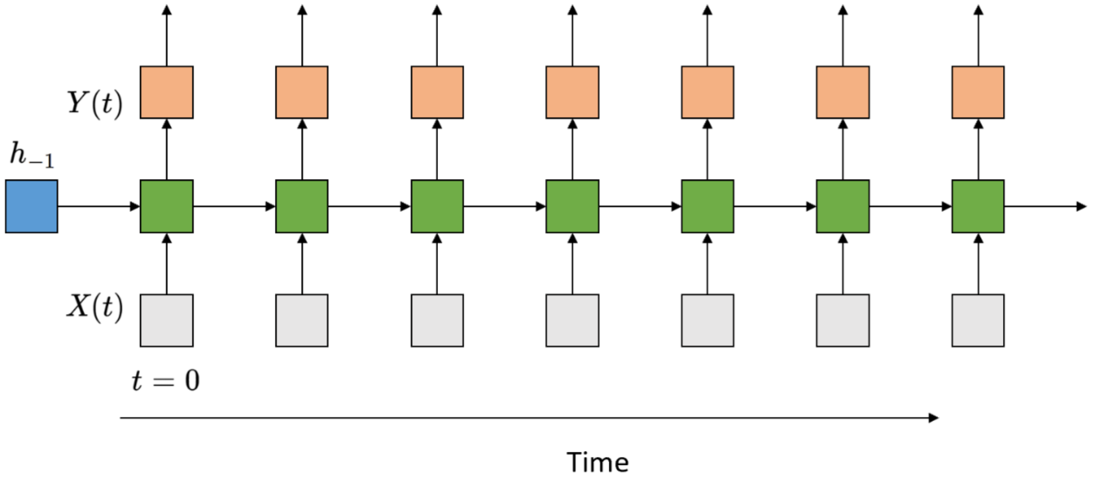
Multiple Recurrent Layer RNN



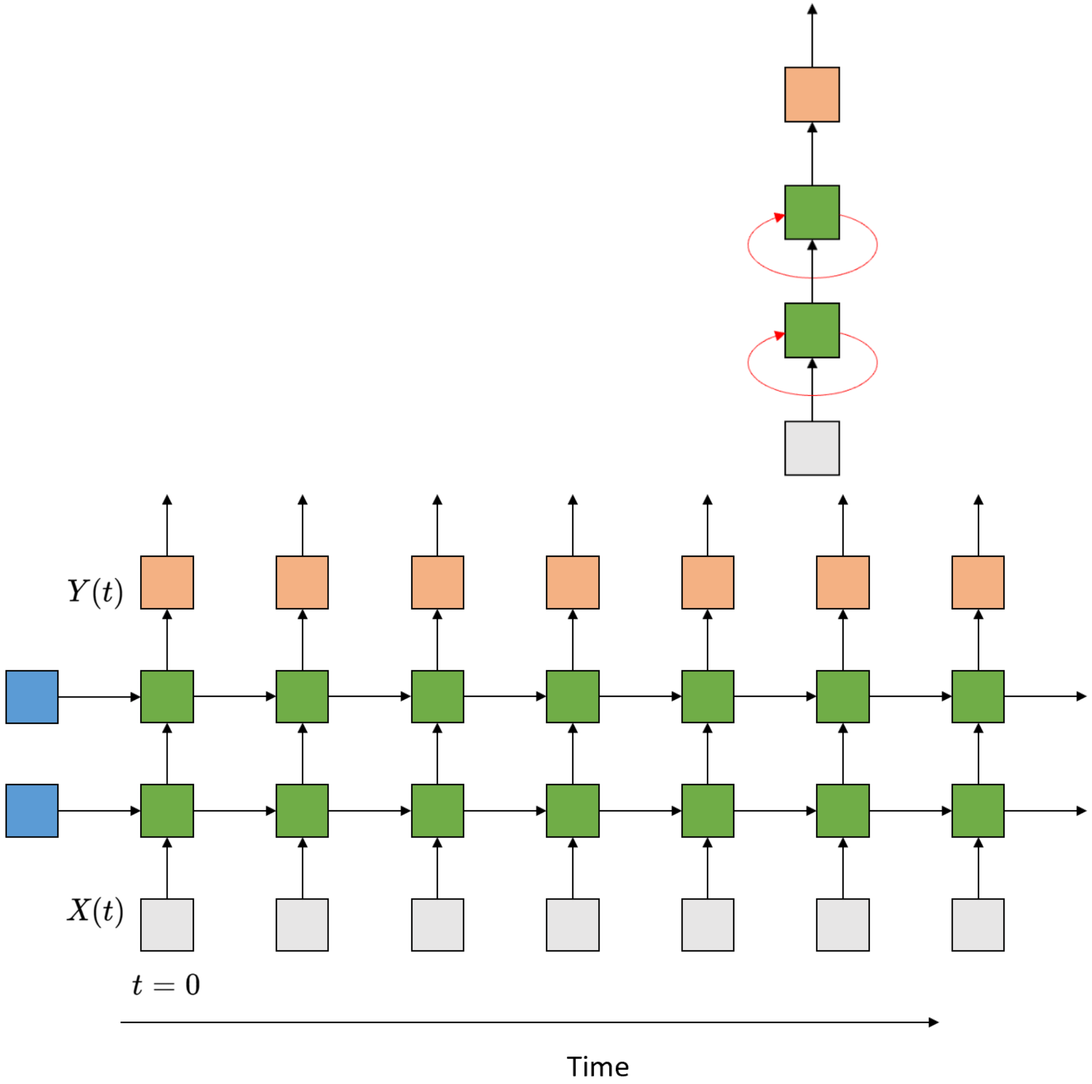
- The state (green) at any time is determined by the input at that time, and the state at the previous time
- All columns are identical
- An input at $t = 0$ affects outputs forever
- Also known as a recurrent neural net

The Folded Version of RNN

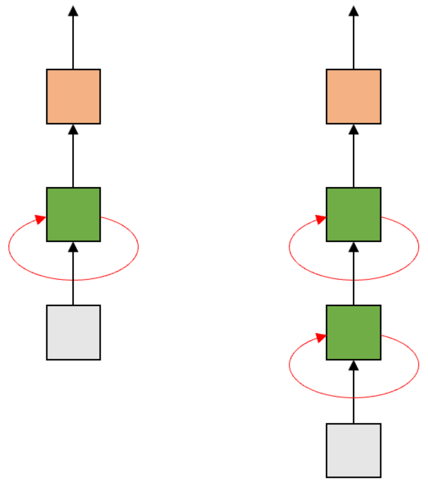




The Folded Version of RNN



Recurrent Neural Network



- Simplified models often drawn
- The loops imply recurrence

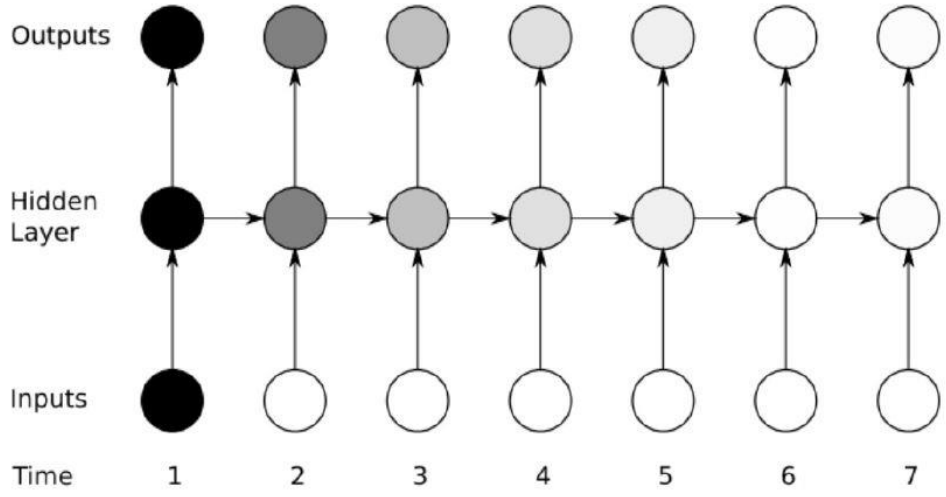
RNN Applications

- Machine translation
- Speech recognition
- Text-to-speech
- Image captioning
- Video analysis/understanding

Long Short-Term Memory (LSTM)

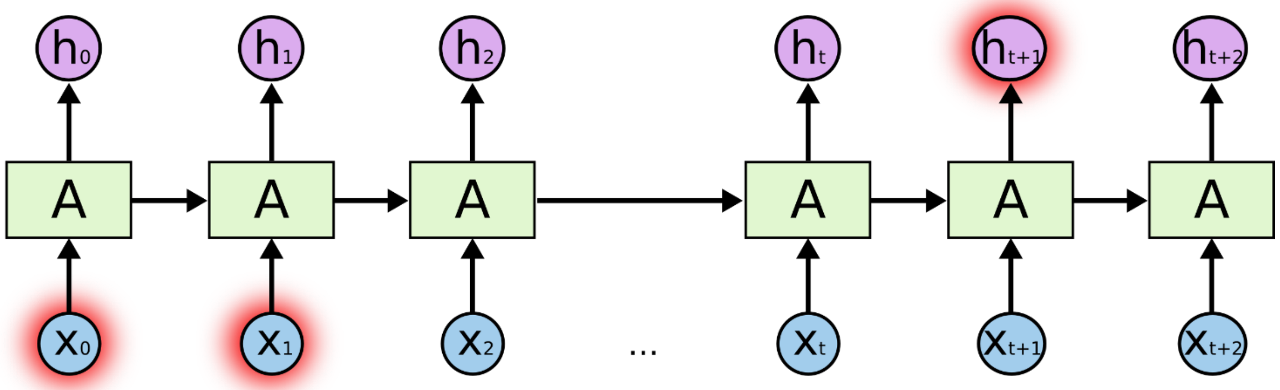
Long Short-Term Memory (LSTM)

- Long-Term Dependencies
  - Gradients propagated over many stages tend to either **vanish** or **explode**
  - Difficulty with long-term dependencies arises from the exponentially smaller weights given to long-term interactions
  - Introduce a memory state that runs through only linear operators
  - Use gating units to control the updates of the state



Example

- "I grew up in France… I speak fluent *French*."

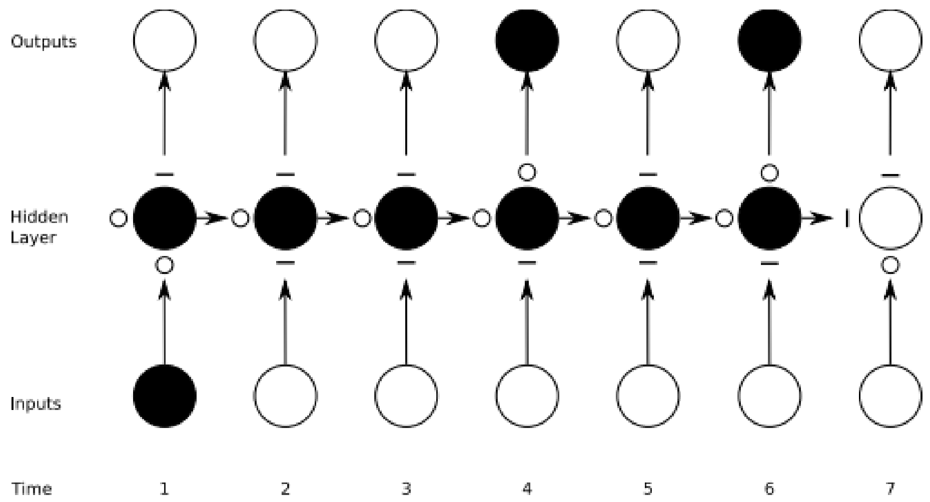


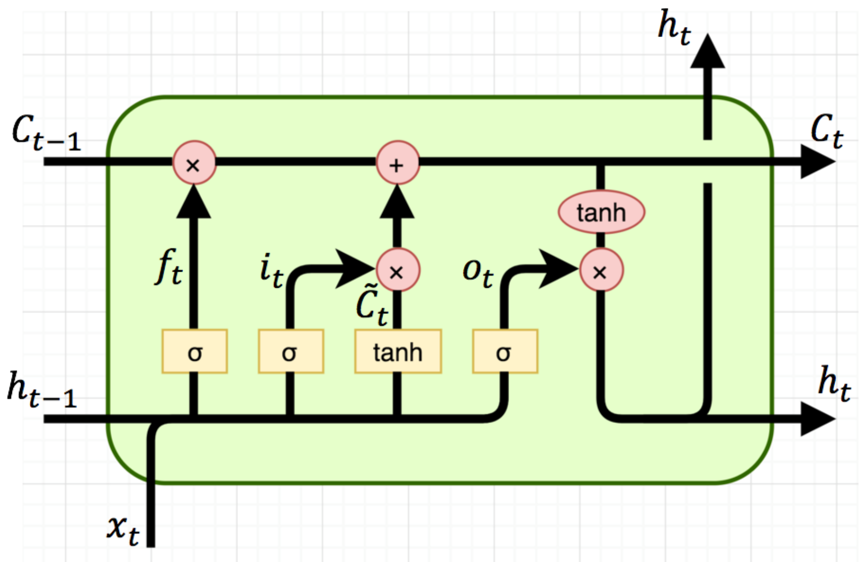
https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short-Term Memory (LSTM)

- Consists of a memory cell and a set of gating units
  - Memory cell is the context that carries over
  - Forget gate controls erase operation
  - Input gate controls write operation
  - Output gate controls the read operation

$$i_t = \sigma\left(x_t U^i + h_{t-1} W^i\right)$$
$$f_t = \sigma\left(x_t U^f + h_{t-1} W^f\right)$$
$$o_t = \sigma\left(x_t U^o + h_{t-1} W^o\right)$$
$$\tilde{C}_t = \tanh\left(x_t U^g + h_{t-1} W^g\right)$$
$$C_t = \sigma\left(f_t * C_{t-1} + i_t * \tilde{C}_t\right)$$
$$h_t = \tanh(C_t) * o_t$$

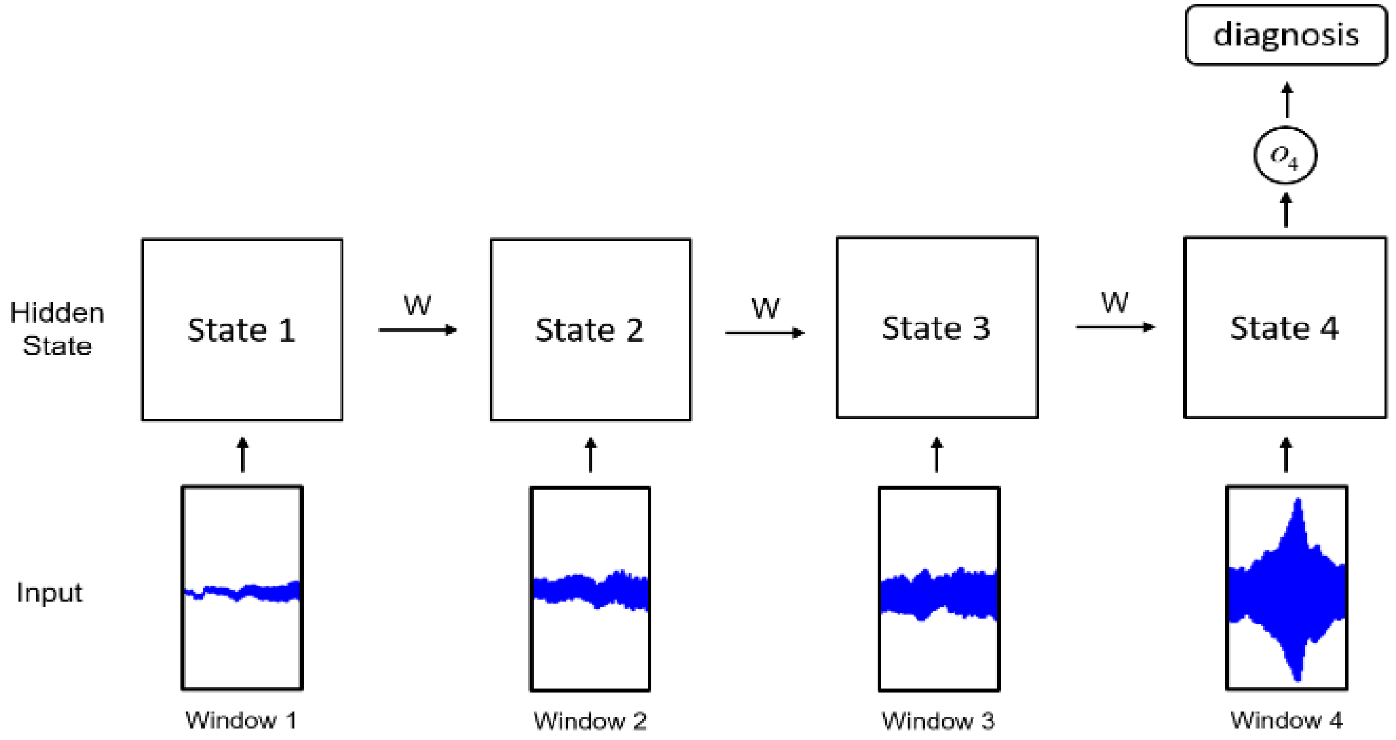




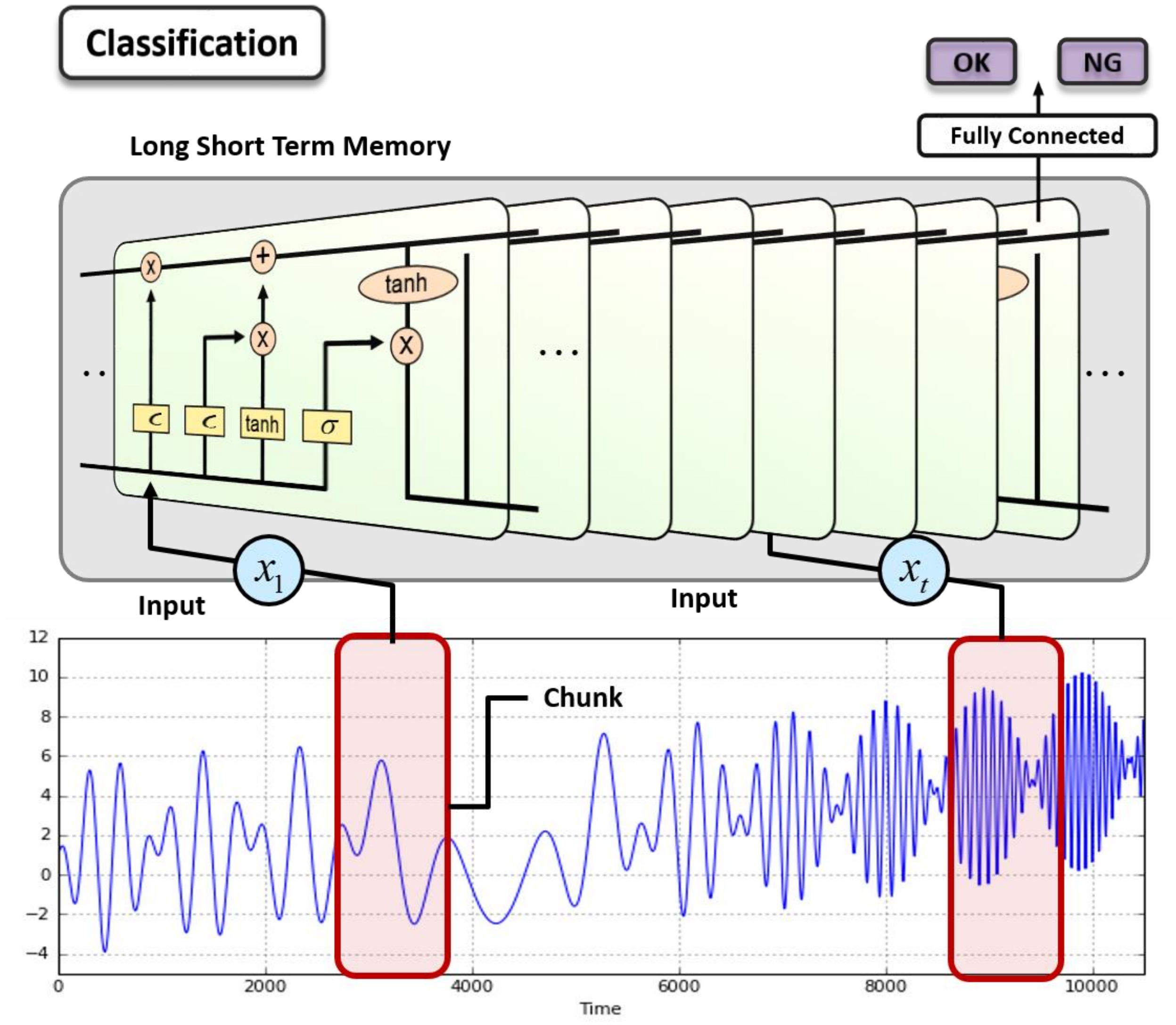
https://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM Implementation

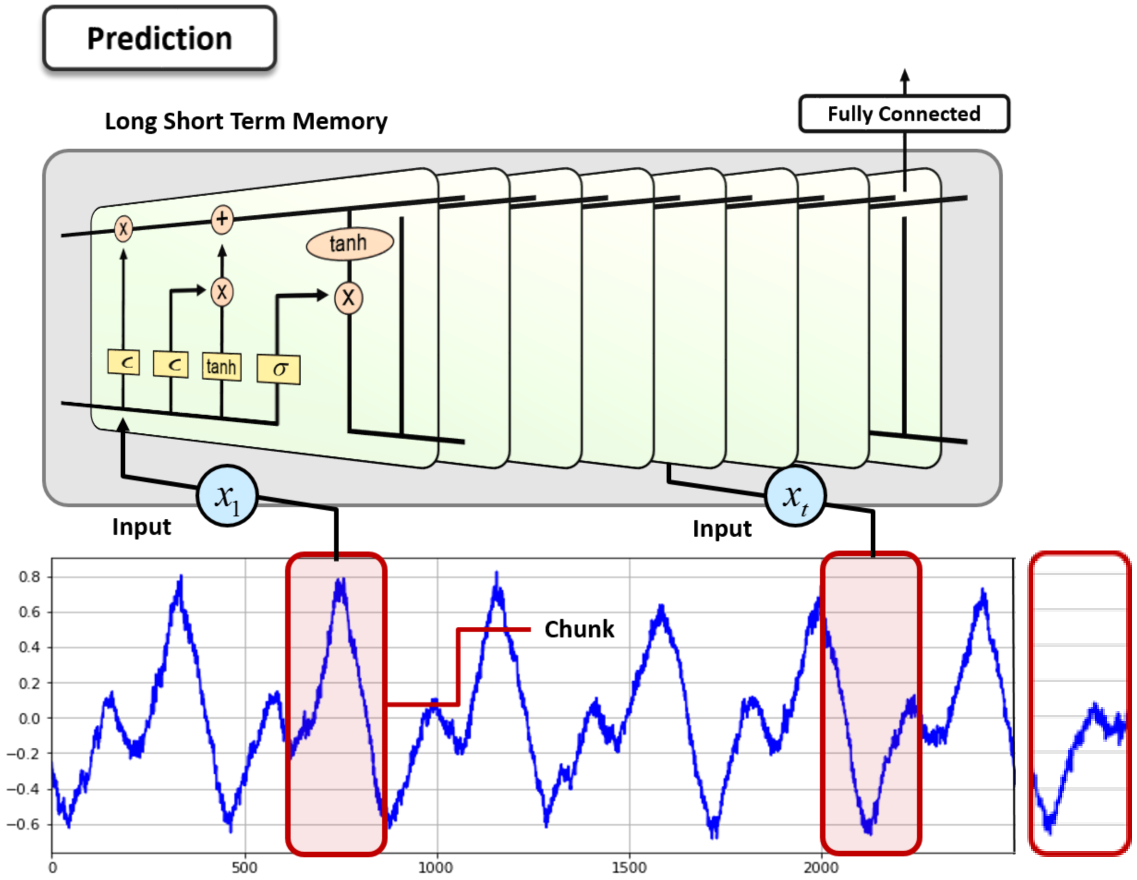
Time Series Data and LSTM



LSTM for Classification



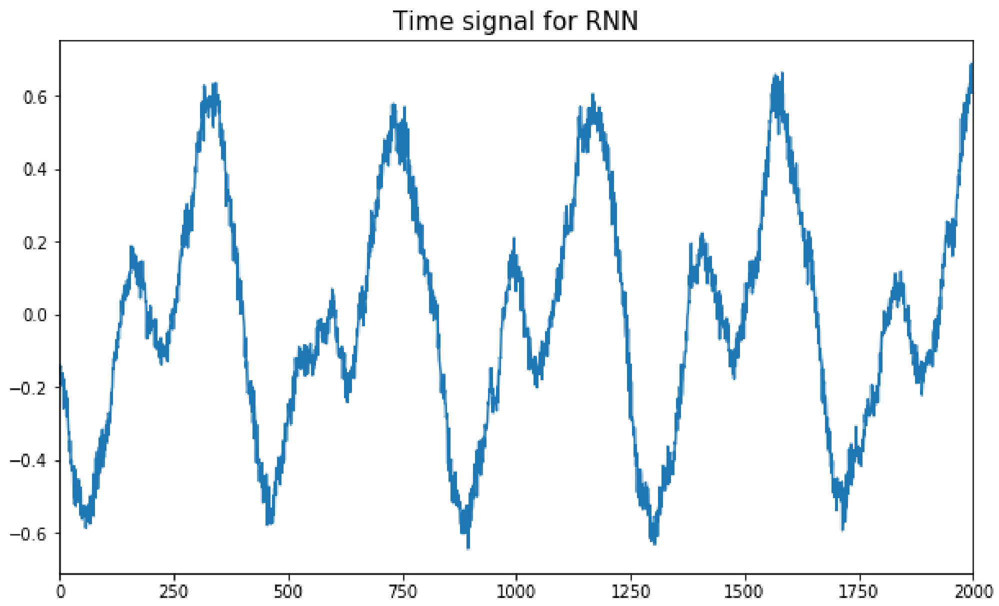
LSTM for Prediction



POSTECH

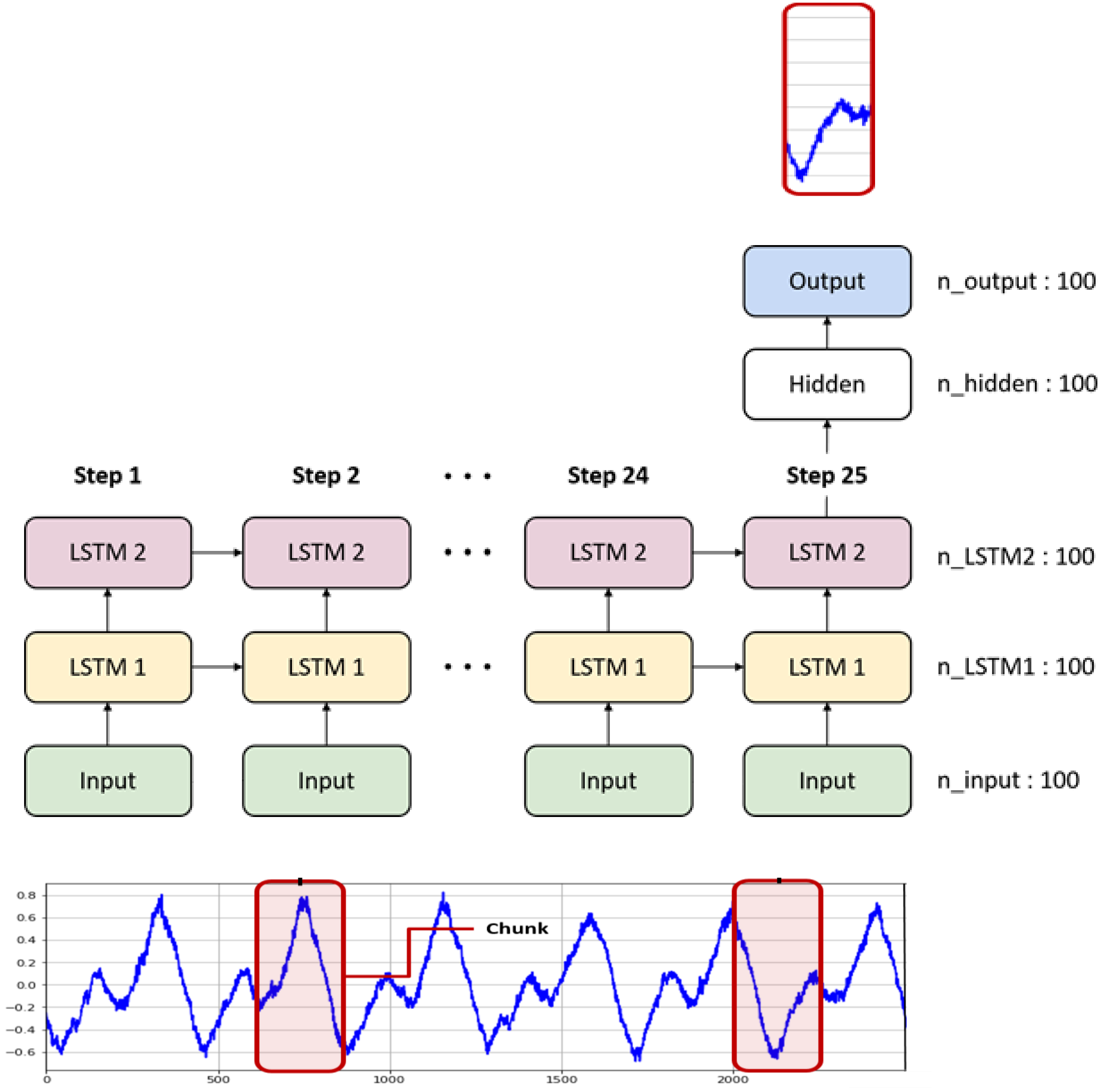
LSTM with TensorFlow

- An example for predicting a next piece of an acceleration signal
- Regression problem



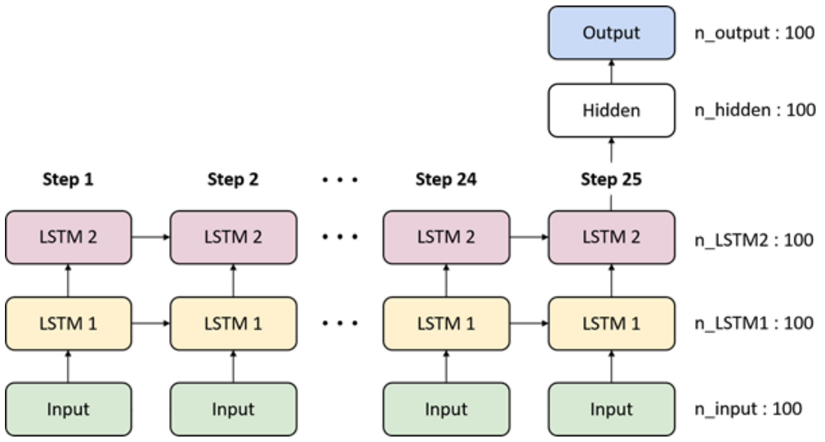
LSTM Structure

```
n_step = 25
n_input = 100

# LSTM shape
n_lstm1 = 100
n_lstm2 = 100

# fully connected
n_hidden = 100
n_output = 100
```



Build a Model

- Define the LSTM cells




```
lstm_network = tf.keras.models.Sequential([
    tf.keras.layers.Input(shape = (n_step, n_input)),
    tf.keras.layers.LSTM(n_lstm1, return_sequences = True),
    tf.keras.layers.LSTM(n_lstm2),
    tf.keras.layers.Dense(n_hidden),
    tf.keras.layers.Dense(n_output),
])

lstm_network.summary()
```

Cost, Initializer and Optimizer

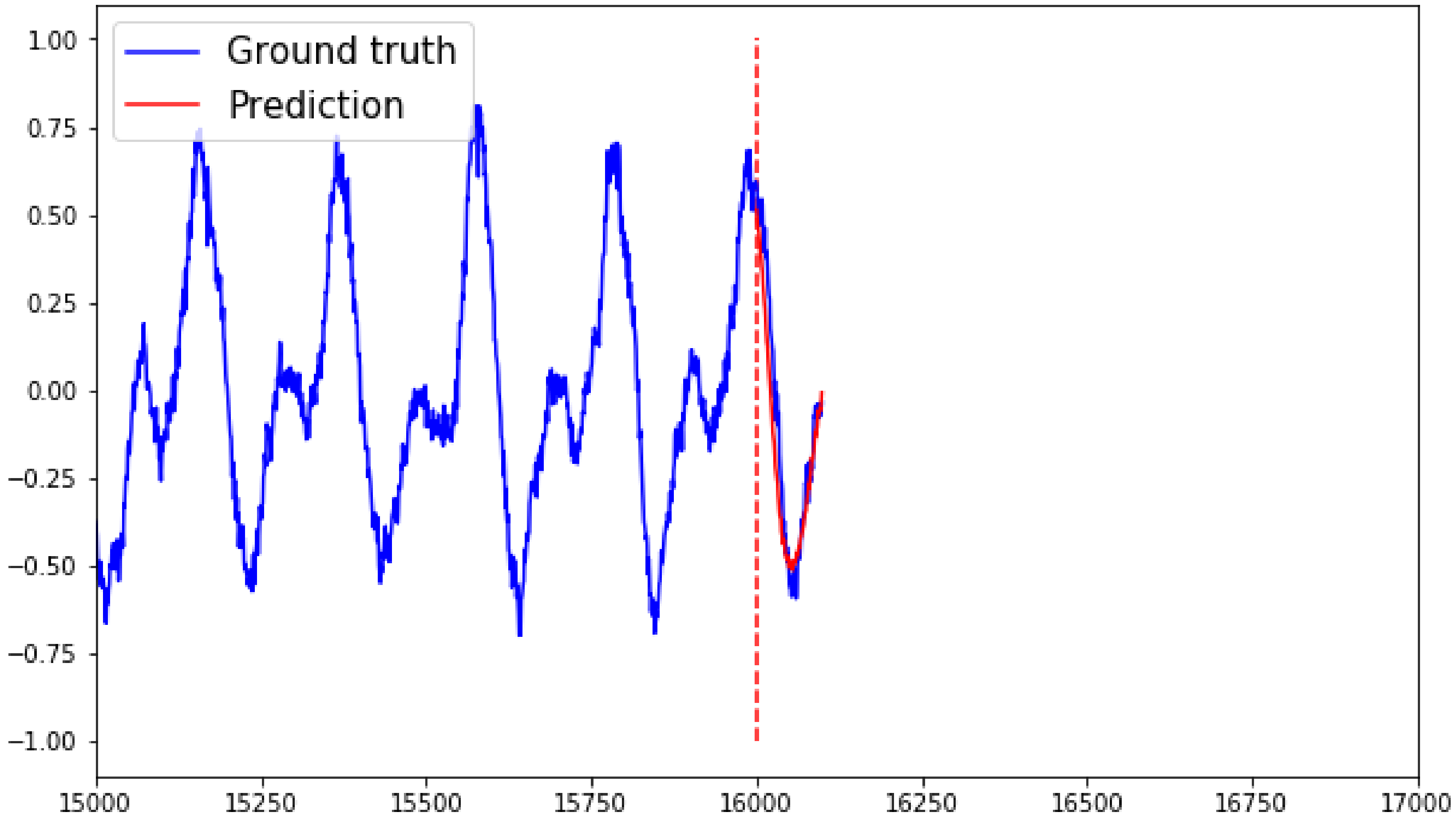
- Loss
  - Regression: Squared loss

$$\frac{1}{N}\sum_{i=1}^{N}(\hat{y}^{(i)} - y^{(i)})^2$$

- Optimizer
  - AdamOptimizer: the most popular optimize

```
lstm_network.compile(optimizer = 'adam',
                     loss = 'mean_squared_error',
                     metrics = ['mse'])
```

Prediction Example



Prediction Example