



مجموعة جامعة  
الجميع الذكية  
للتعليم القابضة

MIDOCEAN  
UNIVERSITY  
BY EVERYONE'S SMART UNIVERSITY

# HCIA-AI Huawei

## Lecture 9

### Sequence Models

Prepared by  
**Eng. Eman Elgallad**

[www.midocean.ae](http://www.midocean.ae)

Lecture

**09**

## Content:

- Sequential Data
- Sequential Models
  - RNN
  - LSTM
  - HMM
- Transformers



# **Sequential Data**

Lecture

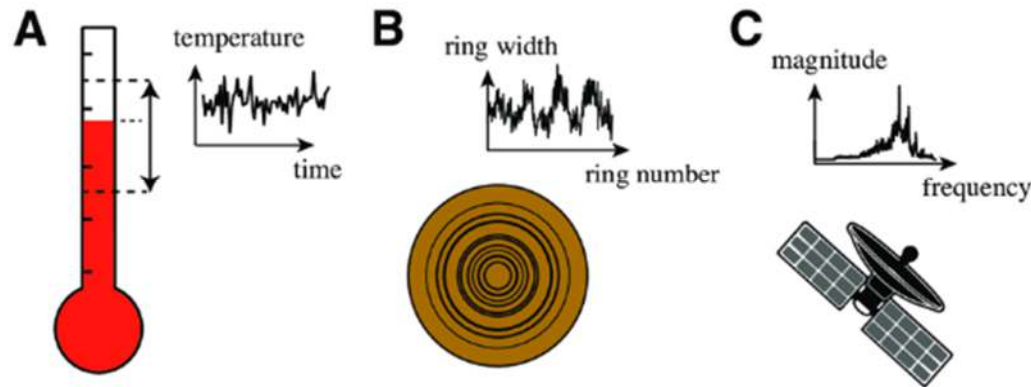
09

## Sequential Data

Sequential data refers to a type of data where the order or sequence of elements is important and carries meaningful information. In sequential data, each data point is dependent on the preceding ones, and the arrangement of these data points over time or position matters

# Characteristics of Sequential Data:

- **Order Matters:** The arrangement of data points in a sequence is significant. Changing the order can alter the meaning or interpretation of the data.
- **Temporal or Spatial Dependence:** Sequential data often exhibits a temporal or spatial dependency, meaning that each data point is influenced by previous ones in a systematic way.
- **Variable Length:** Sequential data can have variable lengths. The number of data points in a sequence is not fixed and can vary from one sequence to another.



## Examples of Sequential Data:

**1. Time Series Data:** Time series data is perhaps the most common example of sequential data. It represents data points collected or recorded at successive points in time. Examples include:

1. Financial Transactions: Transaction data, such as purchases and payments, form sequential records over time for financial analysis.
2. Temperature measurements recorded hourly.
3. Daily website traffic data.

**2. Text Data:** Text data is sequential in nature because the order of words and characters matters for the meaning of the text. Examples include:

1. Sentences and paragraphs in natural language text.
2. Genomic Data: Genomic sequences, such as DNA and RNA, are composed of sequences of nucleotides, and the order of these nucleotides is crucial for genetic information.
3. Code sequences in programming.

## Examples of Sequential Data:

**3.Music:** Musical notes and compositions are sequential data. The order of notes and their timing define the melody and rhythm.

**4.Clickstream Data:** Clickstream data from websites or applications records the sequence of user interactions, such as clicks, page views, and timestamps.

**5.Speech Data:** Speech data consists of sequences of sound waves, and the order of these waves conveys information in the form of spoken words and sentences.

**6.Video Data:** Video data is composed of sequential frames, where each frame represents a snapshot of the scene at a specific moment in time. The order of frames forms the video sequence.

**7.Sensor Data:** Sensor data collected from various sensors (e.g., IoT devices, accelerometers, gyroscopes) often represents sequential measurements taken over time.

## Why is sequential data an issue for the traditional neural network?

**1.Fixed Input Size:** FNNs have a fixed input size, meaning they expect input data to have a predetermined and constant dimensionality. Sequential data, however, often has varying lengths, and it's challenging to adapt FNNs to handle sequences of different lengths.

**2.Lack of Memory:** FNNs lack the ability to retain information about previous inputs or states. They treat each input independently and do not consider the order or context of the data points. This is a significant limitation for tasks that involve sequences where past information is essential for understanding the current state.

**3.Inability to Capture Temporal Dependencies:** In many sequential data problems, there are temporal dependencies where the current data point depends on previous ones. FNNs, with their fixed architecture, cannot capture these dependencies effectively.

# Example

Let's consider a sequential problem to see how well suited basic neural network might be. Suppose we have a sequence of data that contains temperature and humidity value every day.

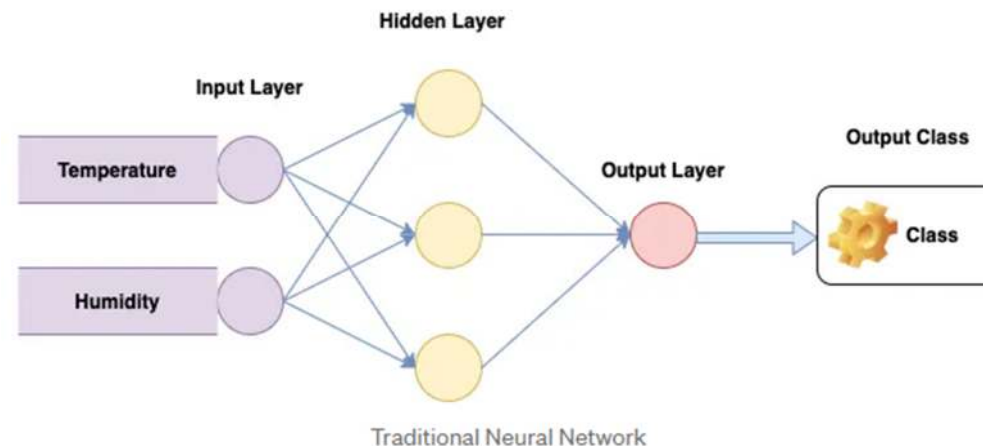
Our goal is to build a neural network that imports the temperature and humidity values of a given day as input and output. For instance, to predict the weather for that day is sunny or rainy. This is a straightforward task for a traditional feedforward neural network.

| timeseries | temperature | humidity    |
|------------|-------------|-------------|
| 2019-11-01 | 30.49789813 | 68.43105695 |
| 2019-11-02 | 34.17009898 | 55.73184495 |
| 2019-11-03 | 38.72460807 | 57.63061612 |
| 2019-11-04 | 32.59569289 | 66.4557475  |
| 2019-11-05 | 30.04958949 | 59.94638509 |
| 2019-11-06 | 33.35118282 | 58.42986496 |
| 2019-11-07 | 32.42705438 | 68.35190681 |
| 2019-11-08 | 35.35606403 | 51.16645954 |
| 2019-11-09 | 35.97581918 | 54.28133089 |
| 2019-11-10 | 32.54127693 | 60.67027843 |

# Example Cont.

Using our dataset, we first feed a data point into the input layer. The data then flows into the hidden layers where the weights and biases are applied. Then the output layer classifies the result from the hidden layer, which produces the output of sunny or cloudy or rainy weather.

Of course, we can repeat the steps for the second day and get the results. However, it is important to know that the model is stateless, **it does not remember the data that it just analyzed**. As a result, the inputs are analyzed in isolation, which can cause problems in case there are dependencies in the data.



# Example Cont.

As you can imagine when examining weather, there is often a strong correlation between the weather on one day having a strong influence on the weather on the subsequent days. That is if it was sunny on one day in the middle of the summer, it's not unreasonable to presume that it'll also be sunny the following day. A traditional neural network model does make use of this information, however, so we'd have to turn to a [Sequential Modelling](#) .

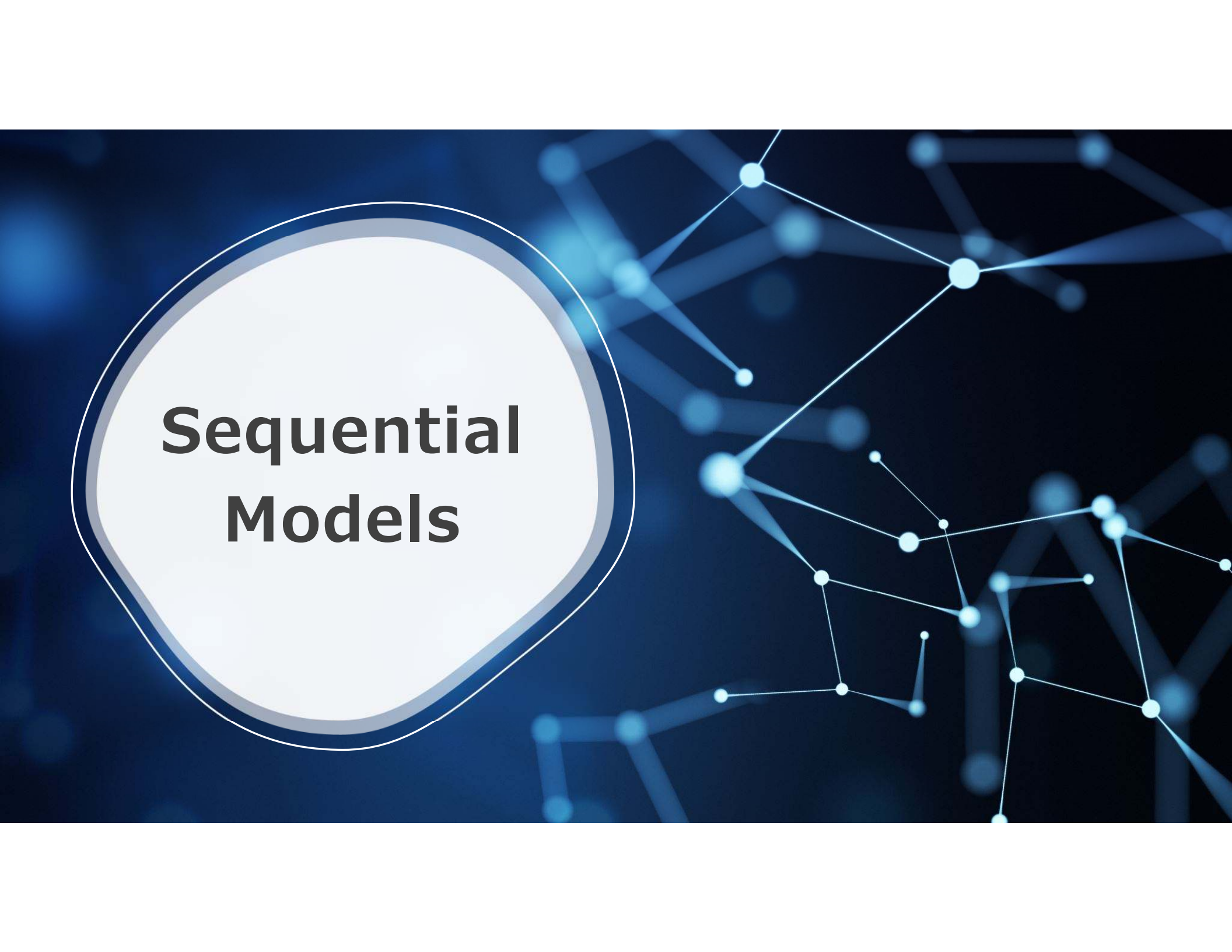
# sequence modelling

sequence modelling is the process of producing a sequence of values from a set of input values.

**For Example,** text prediction, in which the sequence modelling algorithm predicts the next word based on the sequence of the previous phrase and a set of pre-loaded conditions and rules.

## Types of Sequence Models:

- 1.Recurrent Neural Networks (RNNs)
- 2.Long Short-Term Memory (LSTM) Networks
- 3.Gated Recurrent Unit (GRU) Networks
- 4.Transformer Models
- 5.Seq2Seq (Sequence-to-Sequence) Models



# **Sequential Models**

# RNN:

RNN stands for Recurrent Neural Network and is a Deep Learning and Artificial Neural Network design that is suited for sequential data processing.

## Components of an RNN:

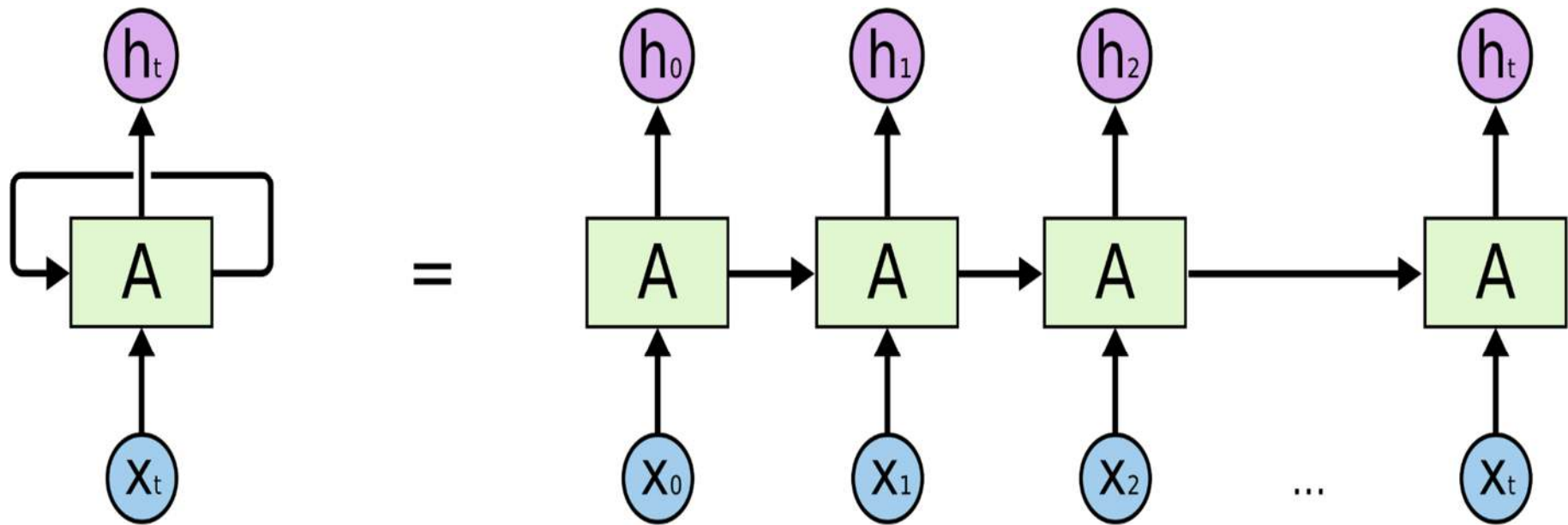
**Input Sequence:** An RNN takes a sequence of inputs where each element in the sequence corresponds to a time step. For example, in natural language processing, each word in a sentence can be an input element.

**Hidden State:** The hidden state, often denoted as "h" or "hidden," is a vector that captures information from previous time steps. It serves as a form of memory for the network and encodes information about the sequence's history.

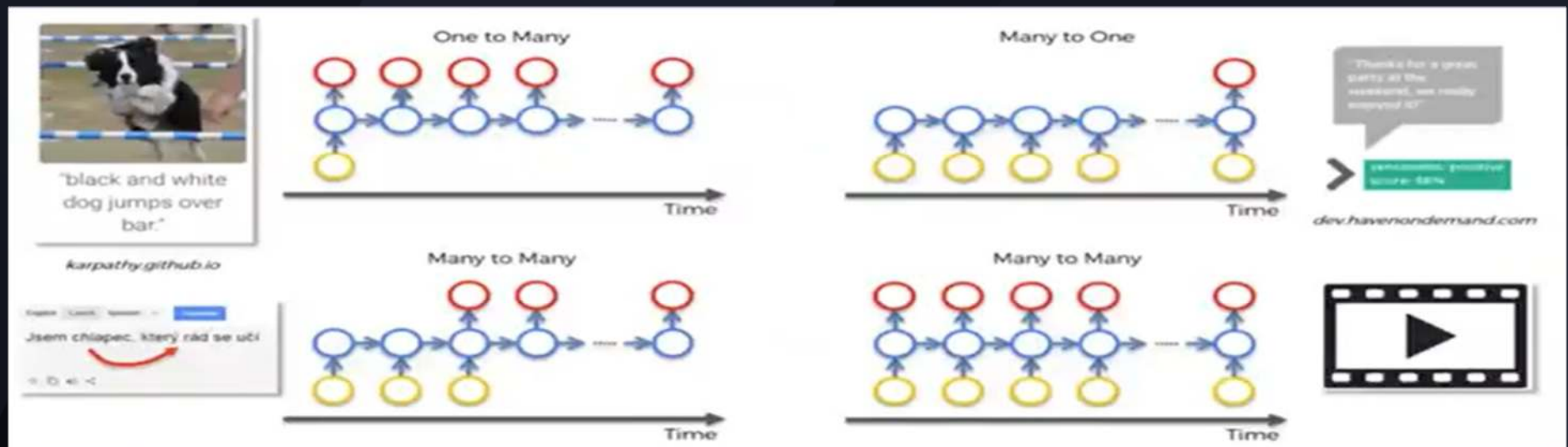
**Weight Matrices:** Like other neural networks, an RNN consists of weight matrices that parameterize the transformations between inputs and hidden states. There are weights for the input-to-hidden and hidden-to-hidden transitions.

**Output:** Depending on the task, an RNN can produce an output at each time step, a single output at the end of the sequence, or both. The output can be a prediction, classification, or any other relevant information.

# RNN Cont.



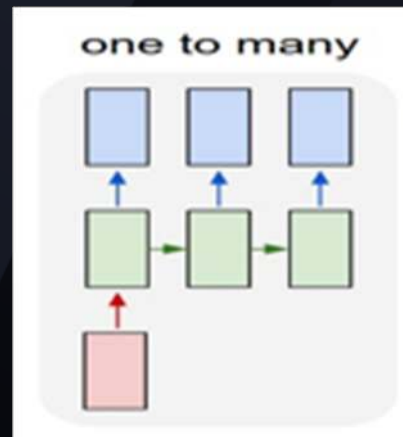
# Types of Recurrent Neural Networks



# Types of Recurrent Neural Networks

## 1. One-to-Many:

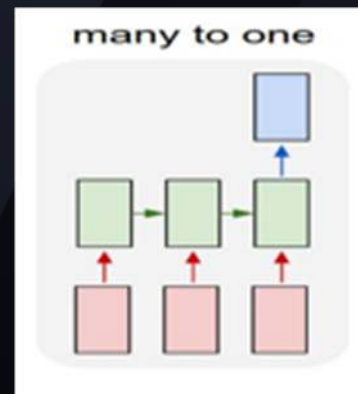
- **Input:** A single input is provided, and multiple outputs are expected. This setup is used when a single input is used to generate a sequence of outputs.
- **Use Case:** An example of one-to-many is image captioning, where a single image is input, and the model generates a sequence of words describing the image.



# Types of Recurrent Neural Networks

## 2. Many-to-One:

- **Input:** Multiple inputs are provided as a sequence, and a single output is expected. This mode is suitable for sequence classification tasks where the model takes a sequence as input and produces a single output.
- **Use Case:** Sentiment analysis is a common example of many-to-one, where a model analyzes a sequence of words (a sentence or document) and predicts the sentiment (positive/negative).

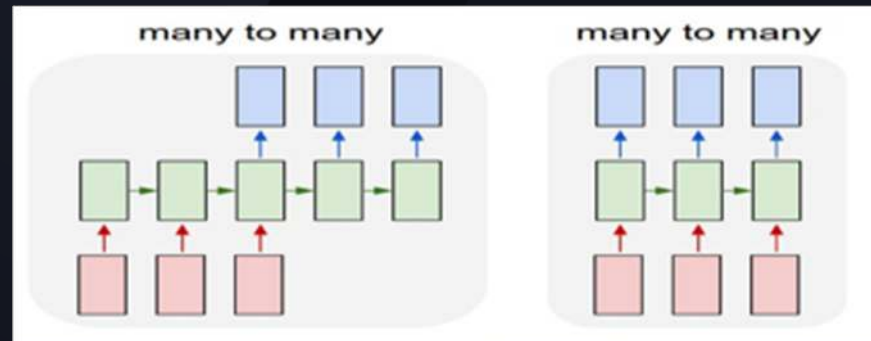


# Types of Recurrent Neural Networks

## 4. Many-to-Many :

- **Input:** Multiple inputs are provided as a sequence, and multiple outputs are expected. In this configuration, the input and output sequences have the same length.
- **Use Case:** Machine translation is a classic many-to-many (same length) task, where the model translates a sequence of words in one language into another language.

**The last many to many model** can be used for video classification on frame level. Feed every frame of a video into the neural network and expect an output right away. However, since frames are generally dependent on each other, it is necessary for the network to propagate its hidden state from the before the next. Thus, we need recurrent neural network for this kind of task.



## Advantages of RNN :

- RNN may represent a set of data in such a way that each sample is assumed to be reliant on the previous one.
- To extend the active pixel neighbourhood, a Recurrent Neural Network is combined with convolutional layers.

## Disadvantages of RNN :

- RNN training is a difficult process.
- If it is using tanh or ReLU like activation function, it wouldn't be able to handle very lengthy sequences.
- The Vanishing or Exploding Gradient problem in RNN

# LSTM

Long ShortTerm Memory is a kind of recurrent neural network. In RNN output from the last step is fed as input in the current step.

It tackled the problem of long-term dependencies of RNN in which the RNN cannot predict the word stored in the long-term memory but can give more accurate predictions from the recent information.

As the gap length increases RNN does not give an efficient performance. LSTM can by default retain the information for a long period of time. It is used for processing, predicting, and classifying on the basis of time-series data.

# LSTM Cont.

A traditional RNN has a single hidden state that is passed through time, which can make it difficult for the network to learn long-term dependencies. LSTMs address this problem by **introducing a memory cell**, which is a container that can hold information for an extended period of time.

**The memory cell is controlled by three gates:**

- the input gate

- the forget gate

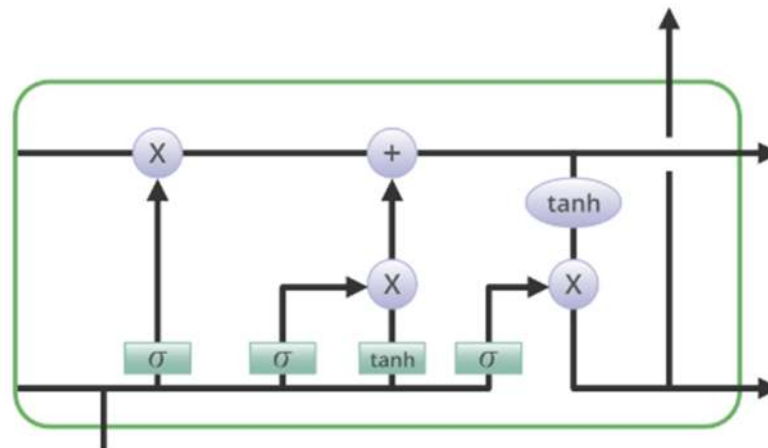
- the output gate.

These gates decide what information to add to, remove from, and output from the memory cell.

# LSTM Cont.

The input gate controls what information is added to the memory cell.  
The forget gate controls what information is removed from the memory cell. And the output gate controls what information is output from the memory cell.  
This allows LSTM networks to selectively retain or discard information as it flows through the network, which allows them to learn long-term dependencies.

## Structure Of LSTM:



# LSTM Structure

**1. Forget Gate:** The information that is no longer useful in the cell state is removed with the forget gate.

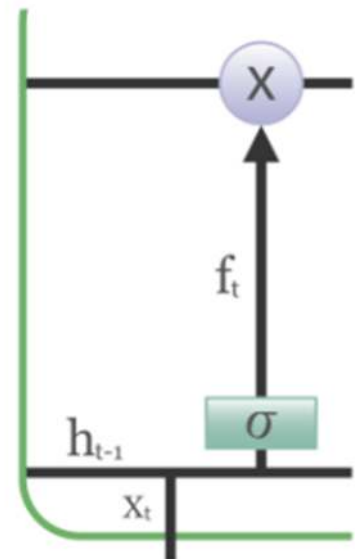
Two inputs  $x_t$  (input at the particular time) and  $h_{t-1}$  (previous cell output) are fed to the gate and multiplied with weight matrices followed by the addition of bias.

The resultant is passed through an activation function which gives a binary output. If for a particular cell state the output is 0, the piece of information is forgotten and for output 1, the information is retained for future use. The equation for the forget gate is:

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

where:

- $W_f$  represents the weight matrix associated with the forget gate.
- $[h_{t-1}, x_t]$  denotes the concatenation of the current input and the previous hidden state.
- $b_f$  is the bias with the forget gate.
- $\sigma$  is the sigmoid activation function.



# LSTM Structure Cont

**2. Input gate:** The addition of useful information to the cell state is done by the input gate.

First, the information is regulated using the sigmoid function and filter the values to be remembered similar to the forget gate using inputs  $h_{t-1}$  and  $x_t$ .

Then, a vector is created using tanh function that gives an output from -1 to +1, which contains all the possible values from  $h_{t-1}$  and  $x_t$ .

At last, the values of the vector and the regulated values are multiplied to obtain the useful information. The equation for the input gate is:

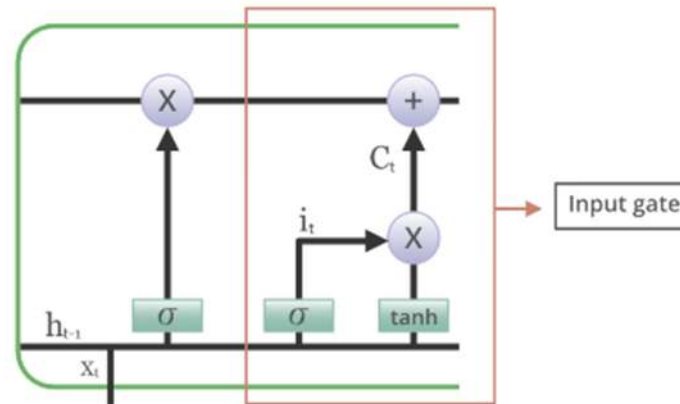
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\hat{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \hat{C}_t$$

where

- $\odot$  denotes element-wise multiplication
- tanh is tanh activation function



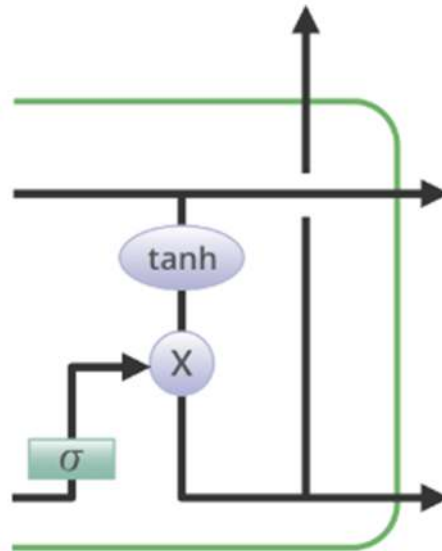
# LSTM Structure Cont

**3. Output gate:** The task of extracting useful information from the current cell state to be presented as output is done by the output gate.

First, a vector is generated by applying tanh function on the cell.

Then, the information is regulated using the sigmoid function and filter by the values to be remembered using inputs  $h_{t-1}$  and  $x_t$ . At last, the values of the vector and the regulated values are multiplied to be sent as an output and input to the next cell. The equation for the output gate is:

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



## Advantages of LSTM

- Long-term dependencies can be captured by LSTM networks. They have a memory cell that is capable of long-term information storage.
- In traditional RNNs, there is a problem of vanishing and exploding gradients when models are trained over long sequences. By using a gating mechanism that selectively recalls or forgets information, LSTM networks deal with this problem.
- LSTM enables the model to capture and remember the important context, even when there is a significant time gap between relevant events in the sequence. So where understanding context is important, LSTMS are used. eg. machine translation.

## Disadvantages of LSTM

- Compared to simpler architectures like feed-forward neural networks LSTM networks are computationally more expensive. This can limit their scalability for large-scale datasets or constrained environments.
- Training LSTM networks can be more time-consuming compared to simpler models due to their computational complexity. So, training LSTMs often requires more data and longer training times to achieve high performance.
- Since it is processed word by word in a sequential manner, it is hard to parallelize the work of processing the sentences.

Lecture

09

**HMMs:** Hidden Markov Models are probabilistic graphical models used for modeling sequences of observations. They are particularly well-suited for problems where you have a sequence of observable states and want to infer the underlying hidden states that generated those observations.

HMMs use a discrete state space and probabilistic transitions between states.

## Markov Models Cont.

- Next word in a sentence depends on previous symbols already written (history = one, two, or more words)

The lecture leaves me/ bumfuzzled

- ▶ Similar, next character in a word depends on previous characters already written

Bum/fuzzled

- ▶ We can model such kth order dependences between symbols with Markov Models

► Let  $w \in V$  denote the set of possible words/symbols that

Includes

- an UNK symbol for any unknown word (out of vocabulary)
- <beg> symbol for specifying the start of a sentence
- <end> symbol for specifying the end of the sentence

<beg> The lecture leaves me UNK <end>

► In a first order Markov model (bigram model), the next symbol only depends on the previous one

# A first order Markov model

- Each symbol (except ) in the sequence is predicted using the same conditional probability table until a symbol is seen

|           |        | $w_i$ |        |     |     |       |
|-----------|--------|-------|--------|-----|-----|-------|
|           |        | ML    | course | is  | UNK | <end> |
| $w_{i-1}$ | <beg>  | 0.7   | 0.1    | 0.1 | 0.1 | 0.0   |
|           | ML     | 0.1   | 0.5    | 0.2 | 0.1 | 0.1   |
|           | course | 0.0   | 0.0    | 0.7 | 0.1 | 0.2   |
|           | is     | 0.1   | 0.3    | 0.0 | 0.6 | 0.0   |
|           | UNK    | 0.1   | 0.2    | 0.2 | 0.3 | 0.2   |

- The goal is to maximize the probability that the model can generate all the observed sentences (corpus S)

$$s \in S, \quad s = \{w_1^s, w_2^s, \dots, w_{|s|}^s\}$$



مجموعة جامعة  
الجميع الذكية  
للتعليم القابضة

MIDOCEAN  
UNIVERSITY  
BY EVERYONE'S SMART UNIVERSITY



Thank You

[www.midocean.ae](http://www.midocean.ae)