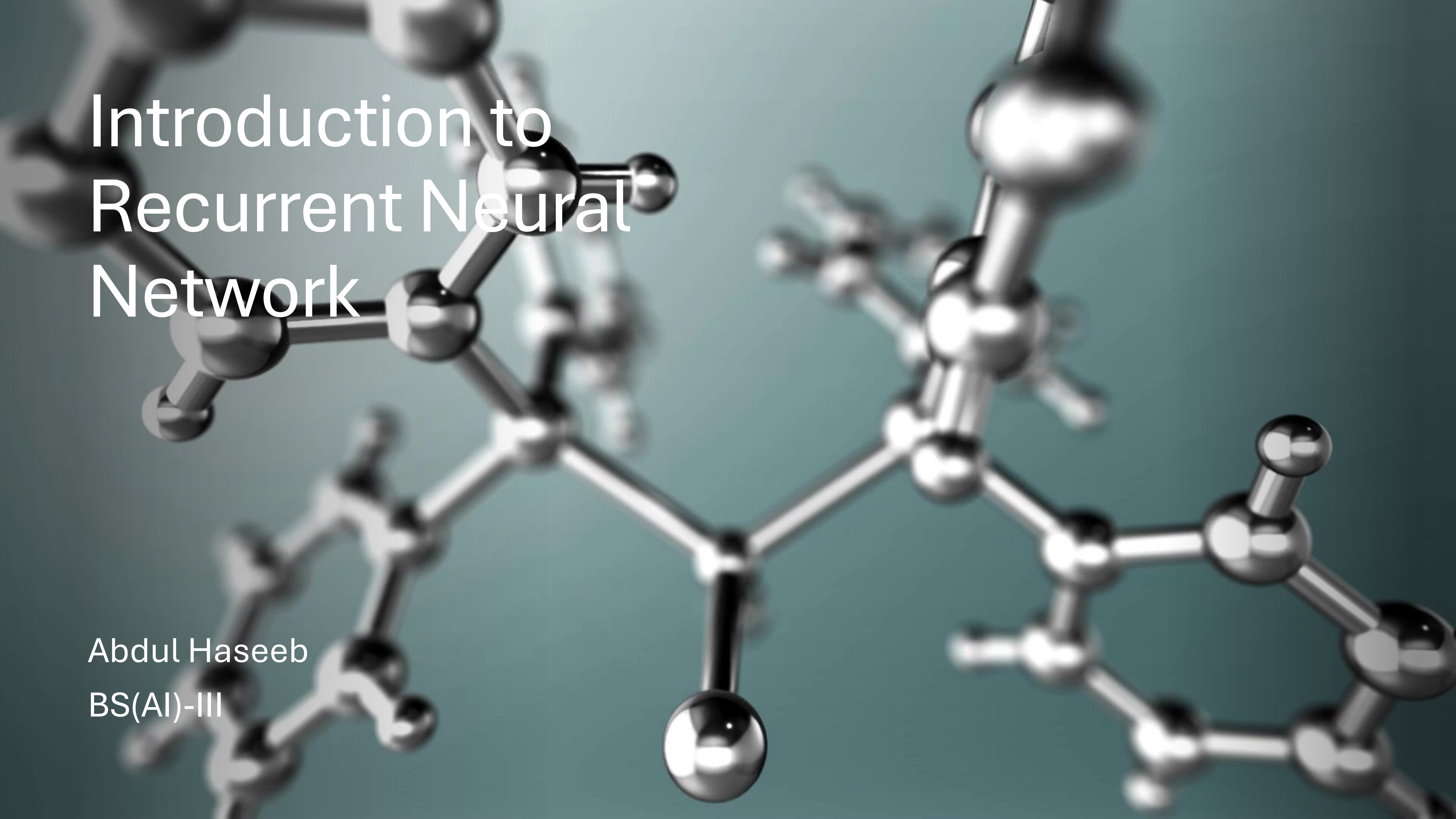


# Introduction to Recurrent Neural Network

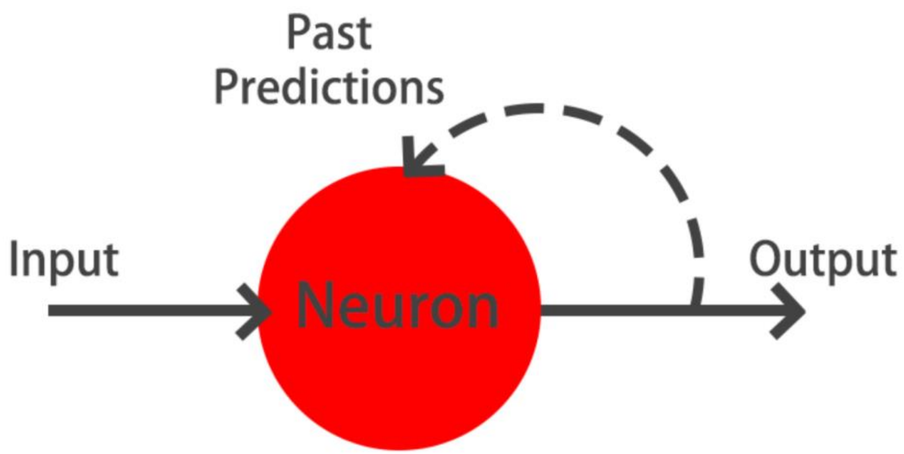
Abdul Haseeb

BS(AI)-III

# What is RNN?

A type of Neural Network that uses past prediction in order to infer new ones

This allows us to solve problems where there is dependence on past input



# Some examples of Dependence

- **Speech Recognition:**
  - Exp: Convert spoken voice to text
  - I want to give it to you, to understand what it means in this sentence requires previous input

# Some examples of Dependence

- **Machine Translation/Generation**
  - To convert the language from English to French
  - To generate a sequence of text

# Some examples of Dependence

- **Video Analysis**
  - Recognizing actions in videos, such as detecting whether someone is running, jumping, or sitting.
  - Recognizing actions in a video requires the model to understand the sequence of frames. For example, identifying a person as "running" requires understanding the movement over a series of frames, not just a single frame.

# Characteristics of RNN

## Sequential Data Processing:

- RNNs are designed to process sequences of data, such as time series, text, or speech, where the order of the data points matters.

## Hidden State:

- RNNs maintain a "hidden state" that acts as a memory of previous inputs.

# When to use RNN?

RNNs are best for tasks that involve sequences, such as:

Time series prediction

NLP tasks (translation, sentiment analysis)

speech recognition, video analysis, and sequence-to-sequence problems.

# Limitation of RNN

RNNs can not maintain long range dependency, because of vanishing gradient problem.

# LSTMs and GRU

Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) are both types of Recurrent Neural Networks (RNNs).

Designed to address the vanishing gradient problem and capture long-term dependencies in sequential data.

These use a memory cell to do that.

# LSTM vs GRU

| Feature | LSTM (Long Short-Term Memory) | GRU (Gated Recurrent Unit) |
|---|---|---|
| **Number of Gates** | 3 gates: Forget gate, Input gate, Output gate | 2 gates: Update gate, Reset gate |
| **Memory Cell** | Yes, separate memory cell and hidden state | No, combines memory and hidden state into one unit |
| **Complexity** | More complex due to 3 gates and a separate memory cell | Simpler with fewer gates (2 gates) |
| **Training Time** | Takes longer to train due to more parameters | Faster to train due to simpler structure |
| **Performance** | Better for tasks needing long-term memory (complex tasks) | Can perform similarly in many tasks, but with fewer parameters |
| **Ability to Remember** | Better at capturing long-term dependencies | Also captures long-term dependencies but might be slightly less effective |
| **Computation Cost** | Higher computation cost due to additional parameters | Lower computation cost due to fewer parameters |
| **Default Choice** | Used when the problem needs complex memory management | Often preferred when simplicity and speed are important |
| **Popular Use Cases** | Language modeling, speech recognition, time-series pred[illegible]n | Similar tasks but with faster results or less complex data |