Deep Learning with Keras and TensorFlow

Recurrent Neural Networks (RNN)

Learning Objectives

By the end of this lesson, you will be able to:

- ◉ Analyze and interpret sequential data to understand temporal dependencies using RNN architectures

- ◉ Examine different RNN architectures to identify their suitability for various types of sequential modeling tasks.

- ◉ Apply LSTM models to perform text classification, demonstrating the ability to capture and utilize linguistic patterns effectively

- ◉ Implement a hybrid RNN model to classify video data, integrating knowledge of both spatial and temporal feature extraction

Business Scenario

ABC Corporation is a financial company that requires a predictive model to analyze financial market trends, analyze time series data, and forecast future outcomes.

To achieve this, the company employs a sequence modeling program that can handle and predict sequential data. They use a recurrent neural network to analyze financial data from the past and use it to forecast future market trends.

The company implements a hybrid model by combining the one-to-many architecture of the RNN with a convolutional neural network to predict a series of sequential events. The hybrid model enables ABC Corporation to make more accurate predictions and gain a competitive edge in the financial market.

Sequential Modeling

Sequential Modeling

Sequential modeling is a task or problem domain that involves predicting or modeling patterns in sequential data.
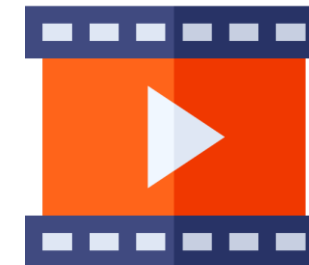
Different kinds of sequential data can be modeled, interpreted, and predicted by a sequential modeling tool, including:

Text

Audio

Video

Sequential Modeling: Advantages

The advantages of sequential modeling are listed below:

- It captures temporal dependencies in data.

- It can handle variable-length input.

- It is advantageous in time series prediction tasks.

- It is widely used in natural language processing.

- It is essential in speech and audio processing.

- It enables sequential decision-making in various domains.
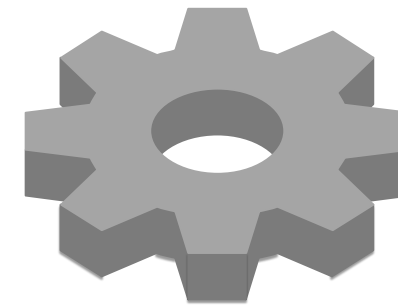
Sequential Modeling

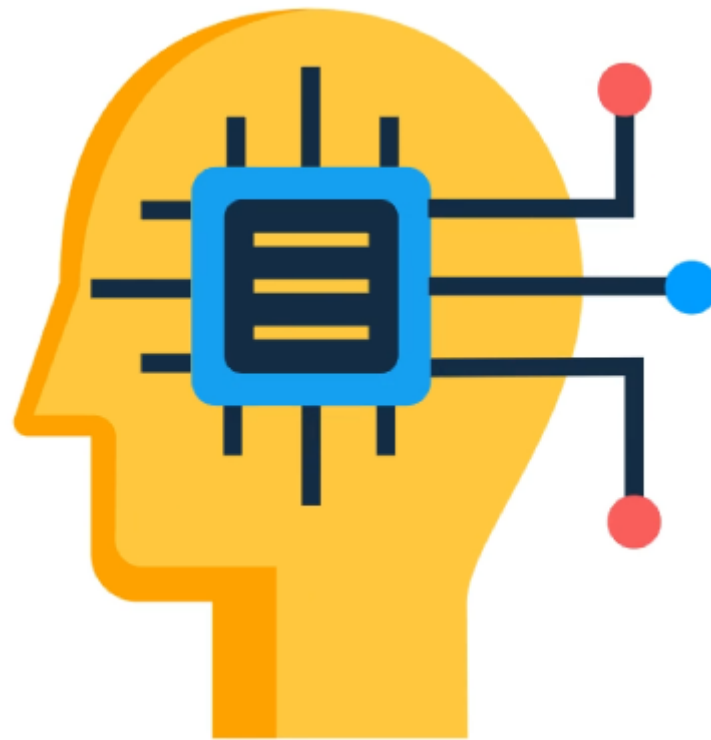The different types of sequential models are:
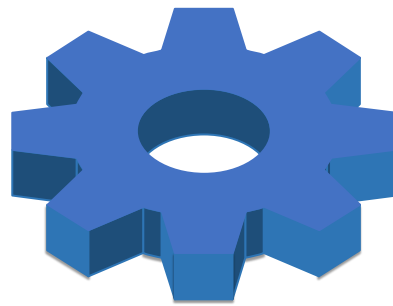
RNNs
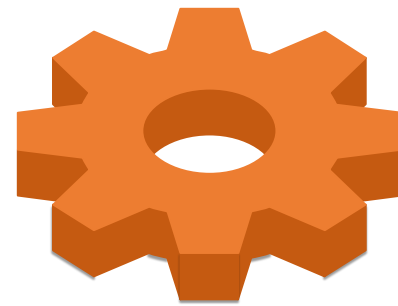
Autoencoders

Seq2Seq

Sequential Modeling

To achieve memory-based learning, a unique artificial intelligence algorithm known as recurrent neural network (RNN) is used.

The 'memory' aspect in RNNs refers to their ability to use information from previous steps (or time points) in the sequence to inform their current output. This is particularly useful in applications where the context is essential.
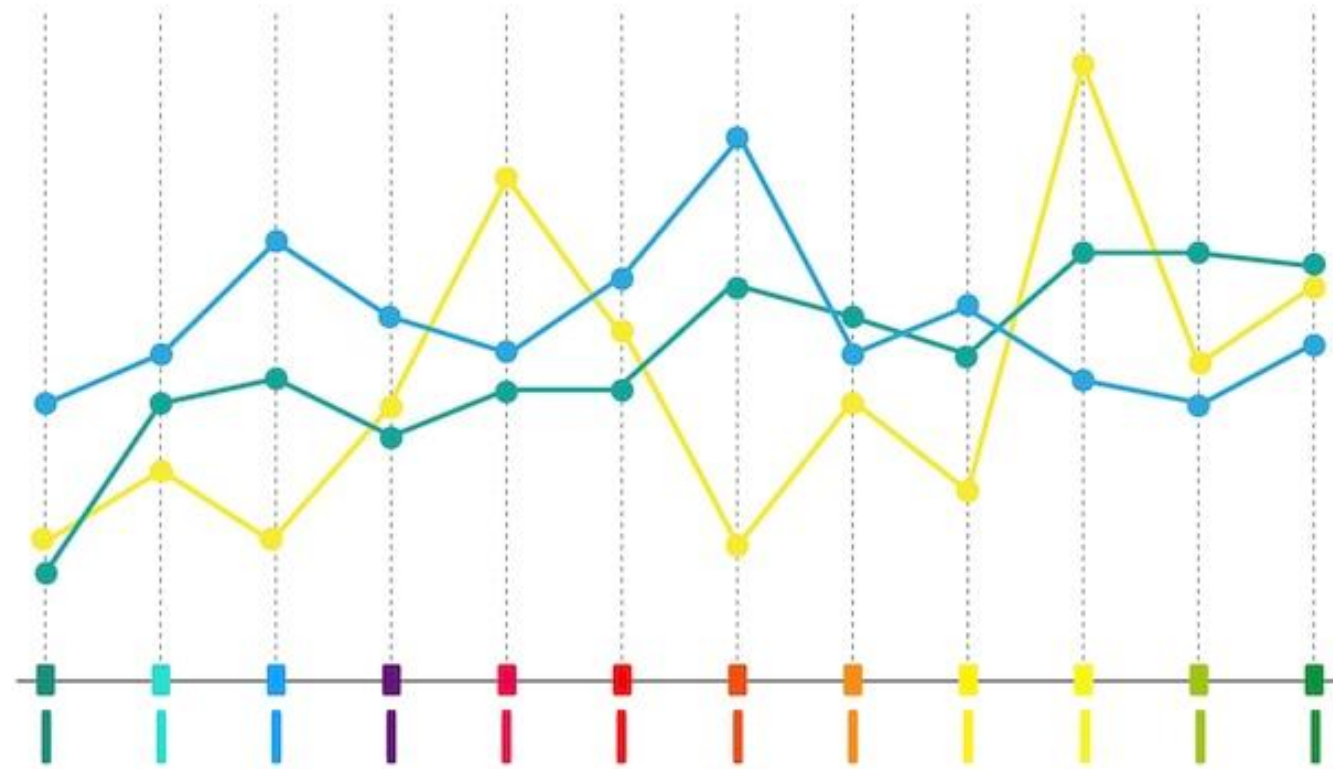
Sequential Modeling: Working

The following image shows the working of sequence modeling.
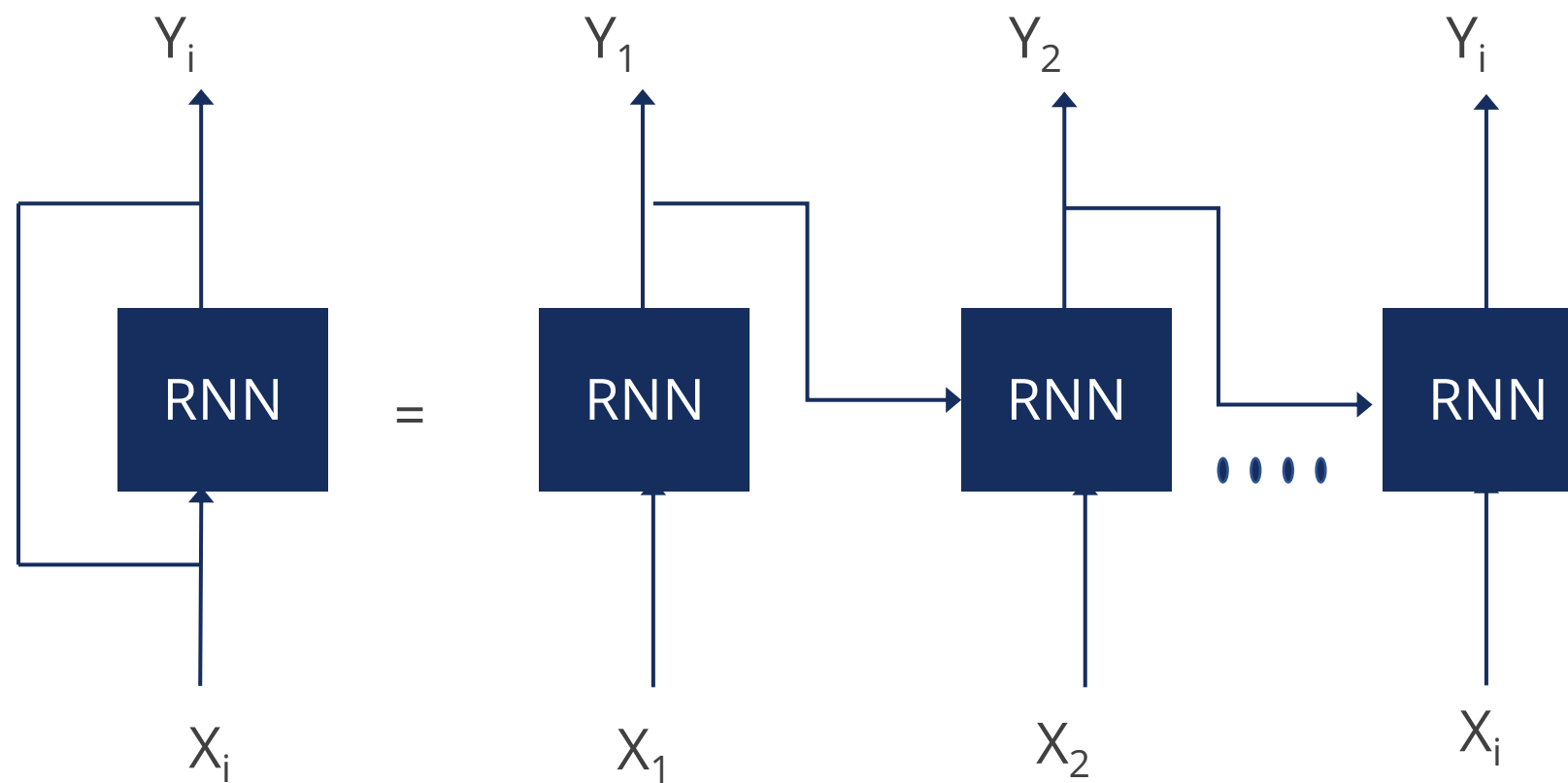
It is used to generate a sequence of values by analyzing a series of input values.

Sequential Modeling: Working

The input values may be a time-series data where variables vary over time.

It analyzes the recurring input values and sees them as discrete inputs for different problems or cases.

Real-World Applications

Sequence modeling can be used in natural language processing to process texts into valuable insights.

Example

An email spam classifier enhances user experience by identifying and filtering out unwanted or harmful emails, keeping the inbox organized and secure

Real-World Applications

Another use-case could be a computer program that:

Takes an audio
file as input

Recognizes its
possible genre

Suggests a sequence
of musical notes to
complete the song

This could aid musical composition, as several musicians face problems in decrypting
symbols early on in their careers.

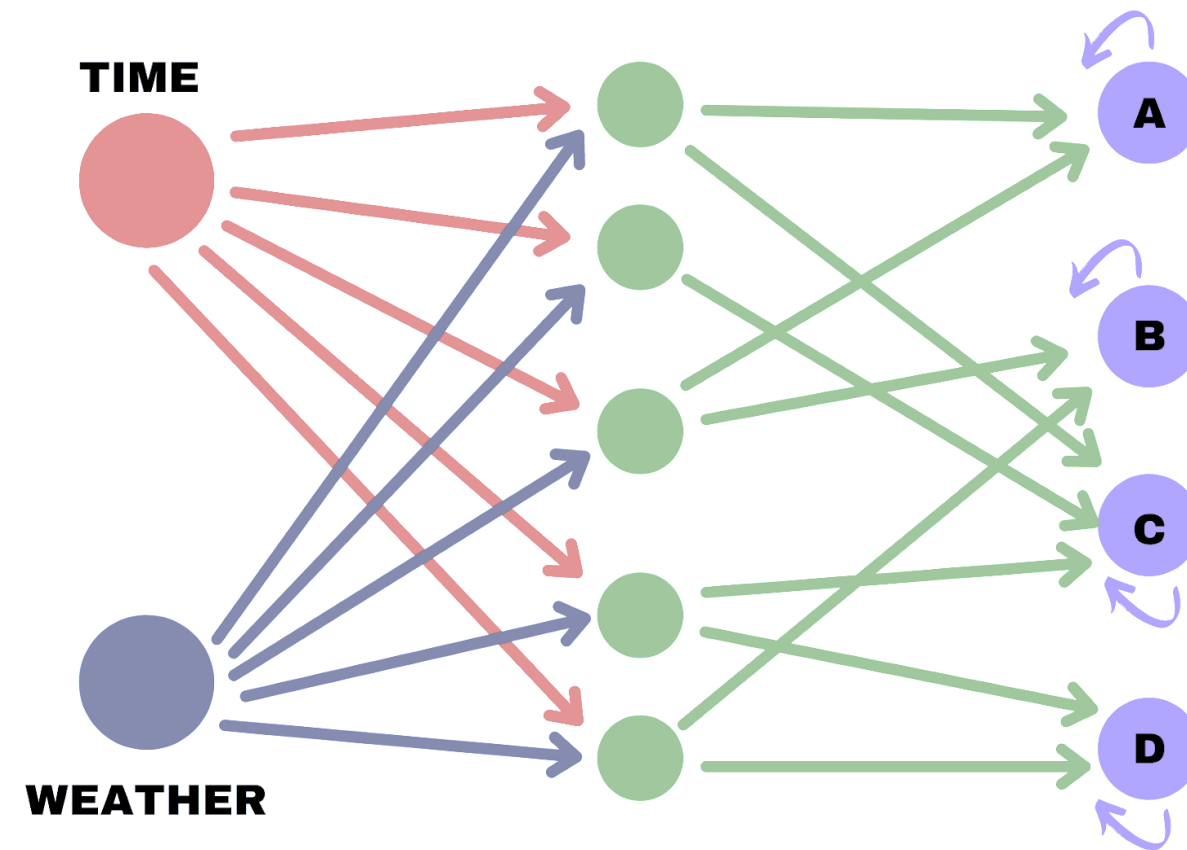# Introduction to Recurrent Neural Networks

Feedforward Neural Networks in RNN

RNNs were introduced after feed-forward neural networks (FNNs) to address the limitation of processing sequential data.
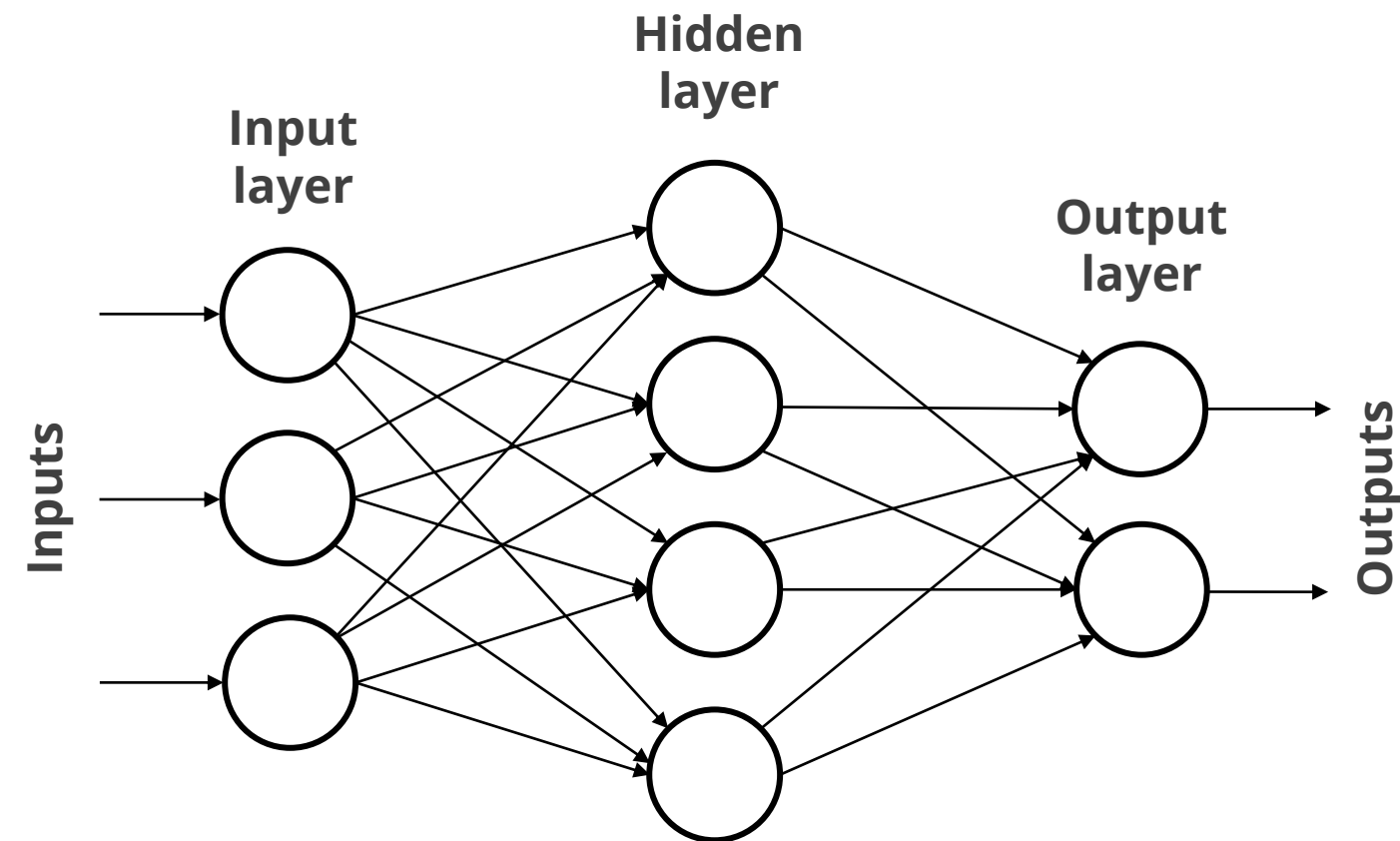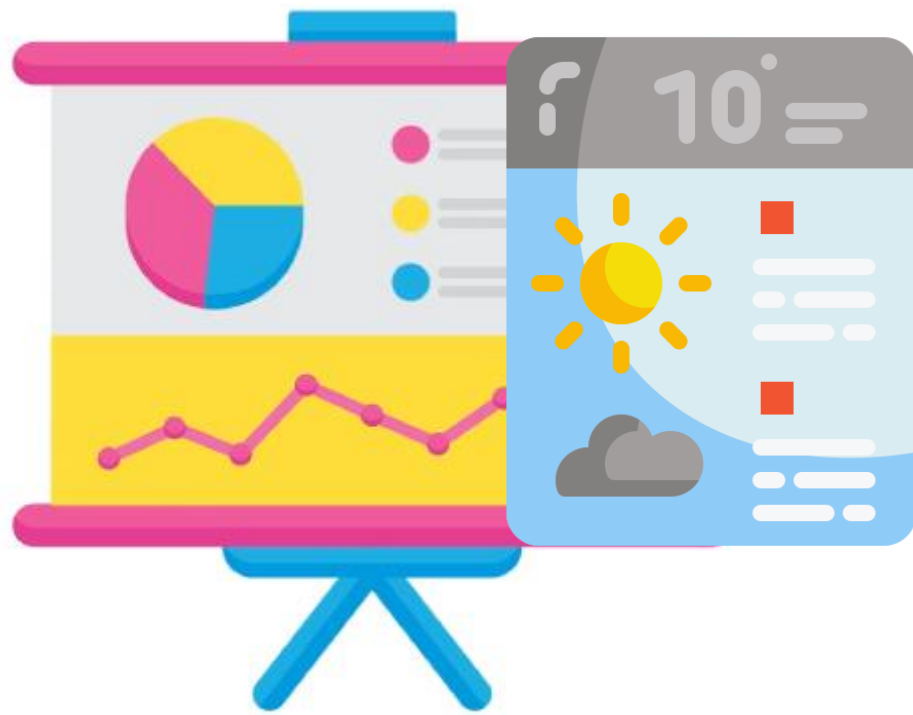
Example: FNN in RNN

Consider the following image and create a computer program that suggests a sport to be played based on the time and the weather:

It follows a particular pattern and makes appropriate decisions.

Example: FNN in RNN

It might suggest football in the evening in clear weather or badminton when there is no wind.
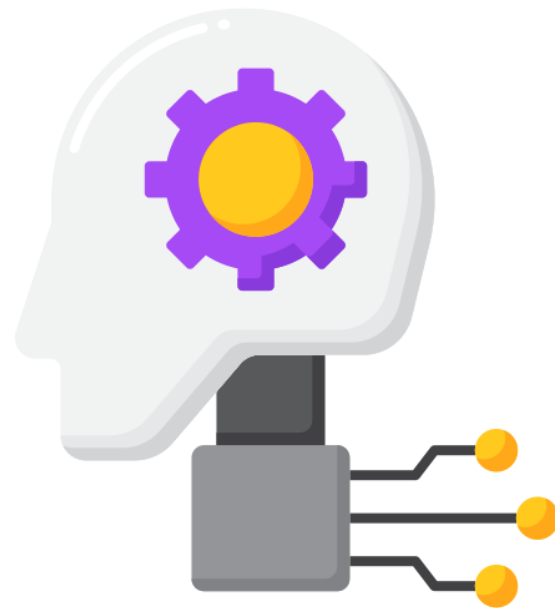
An FNN would suggest the same sport for similar weather conditions every day.

An RNN would remember the user's choices from the previous day and suggest something different the next day.

Recurrent Neural Networks

A recurrent neural network (RNN) is a state-of-the-art algorithm used by Apple's Siri and Google's voice search.

It is the first algorithm that remembers its input owing to internal memory.

The ability to add the recent past to the present allows it to make precise predictions.

It is ideal for machine learning problems that involve sequential data, like natural language processing and time series forecasting.

Recurrent Neural Networks

RNNs are mainly used for:

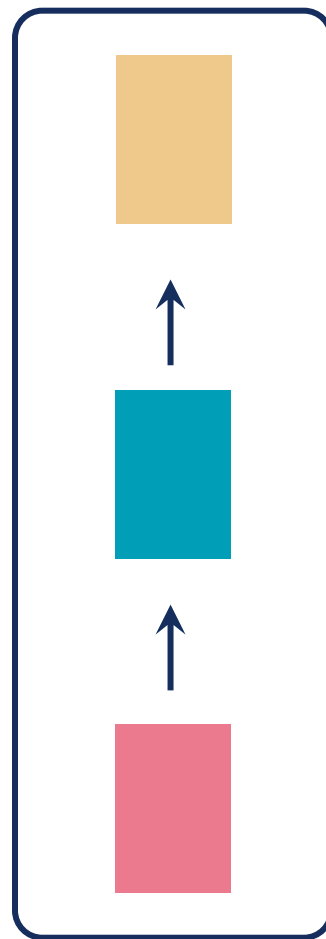| Sequence classification	Sequence labeling
Sentiment classification	Part of speech tagging
Video classification	Named entity recognition

Recurrent Neural Networks

The different types of RNNs are:
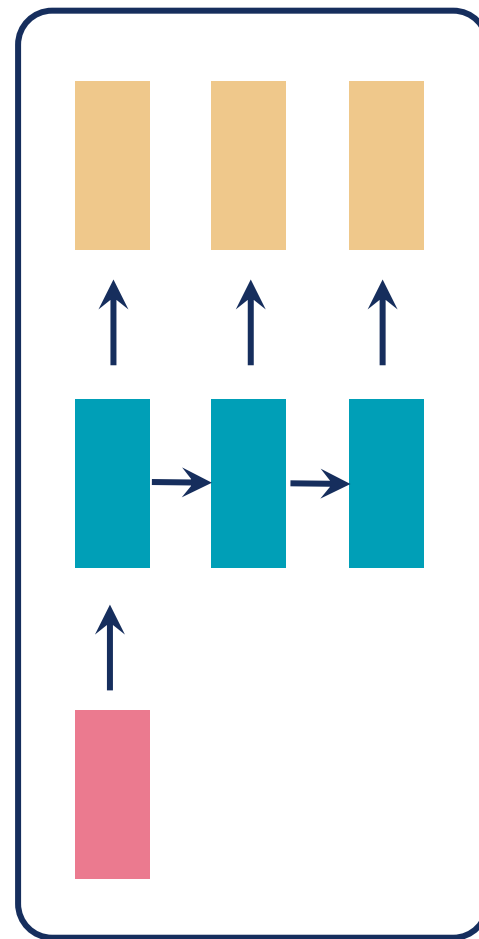
One-to-one   One-to-many   Many-to-one   Many-to-many
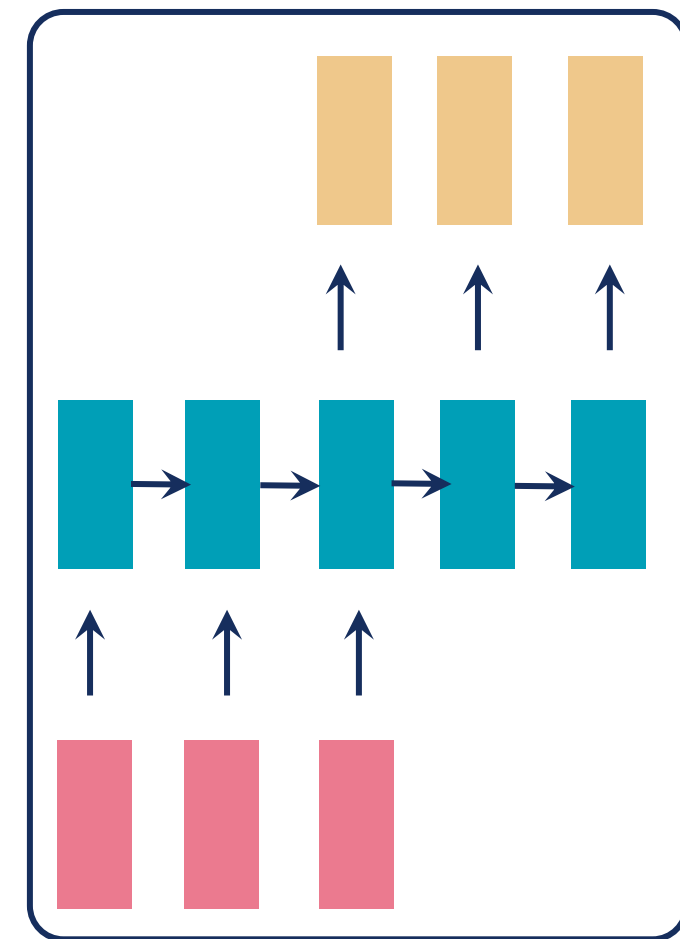
One-to-One

One-to-one architecture forms the basis of feedforward neural networks.

$$O^{(t)}$$

$$h^{(t)}$$

$$x^{(t)}$$
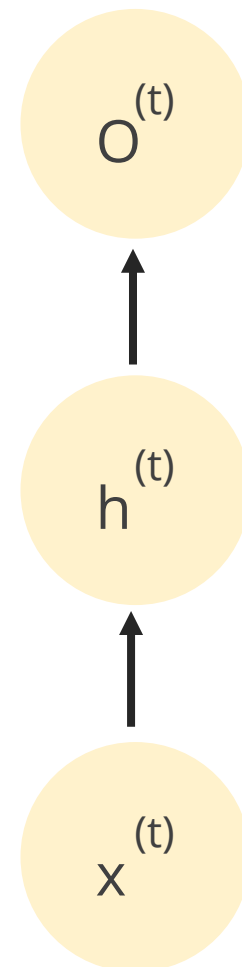
Example

An architecture that is used to classify the image of a musical symbol

Activation values are not needed as this is a straightforward scenario of a single input and a single output as seen in the figure.

One-to-Many

A single input drives the entire network.
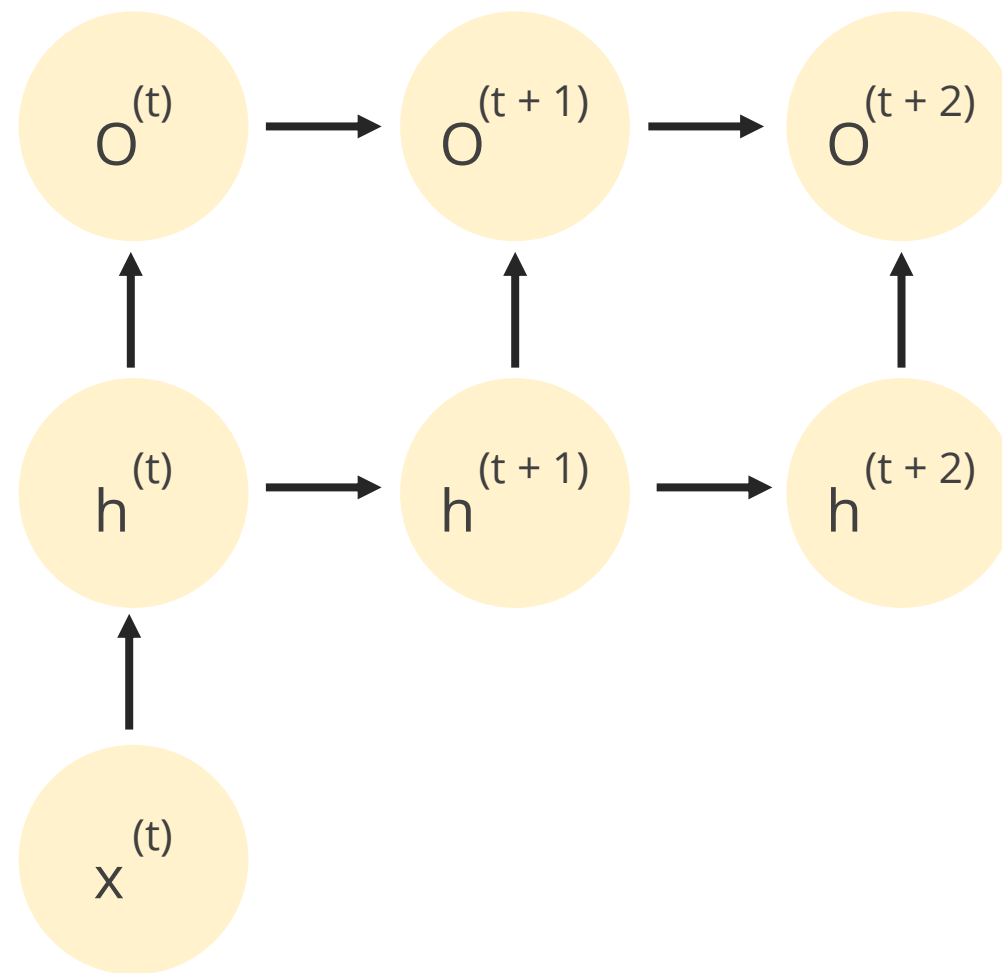
Example

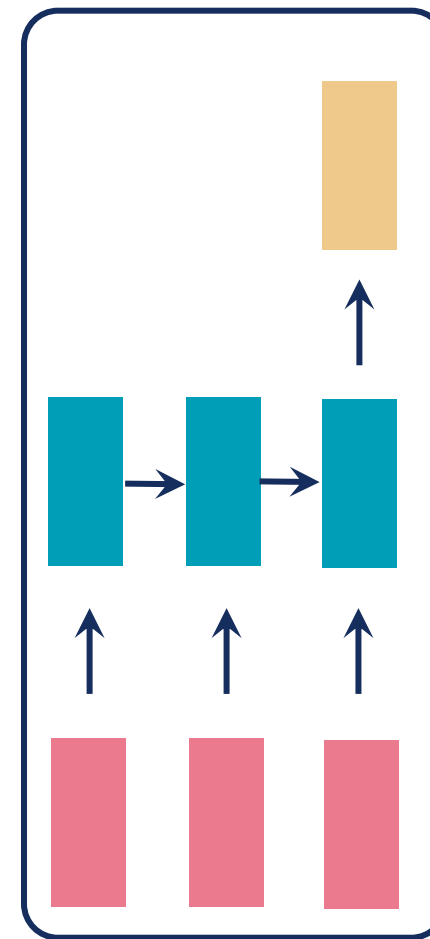Generating a sequence of musical notes based on the first note or the genre of music given as input

Once the input is given, The network produces a series of outputs, one for each time step.

Many-to-One

A sequence of multiple inputs is given to the network, which is instructed to predict a single output.

$$O^{(t)}$$

$$h^{(t-2)} \rightarrow h^{(t-1)} \rightarrow h^{(t)}$$

$$x^{(t-2)} \quad x^{(t-1)} \quad x^{(t)}$$
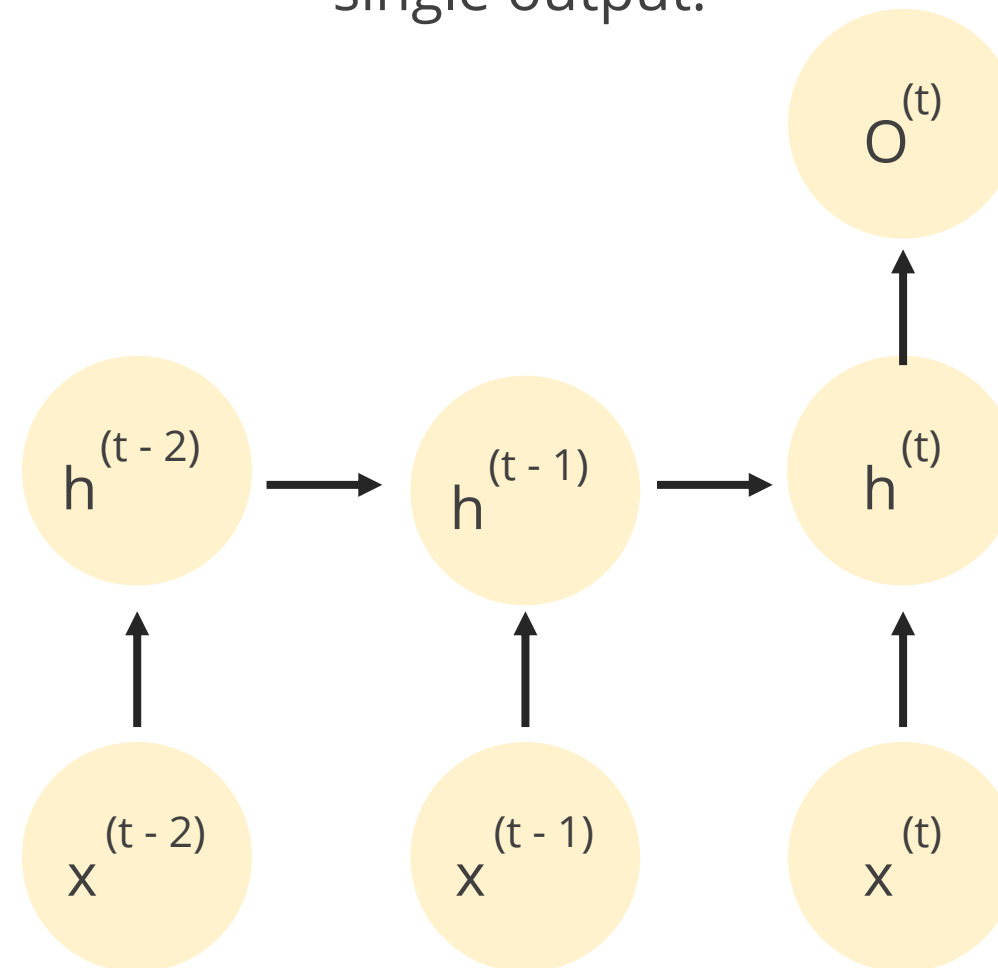
Example

The task of classifying the genre of music by giving a song as input

Many-to-Many

A sequence of multiple inputs is fed to the network to predict a sequence of outputs that may or may not be of the same length.
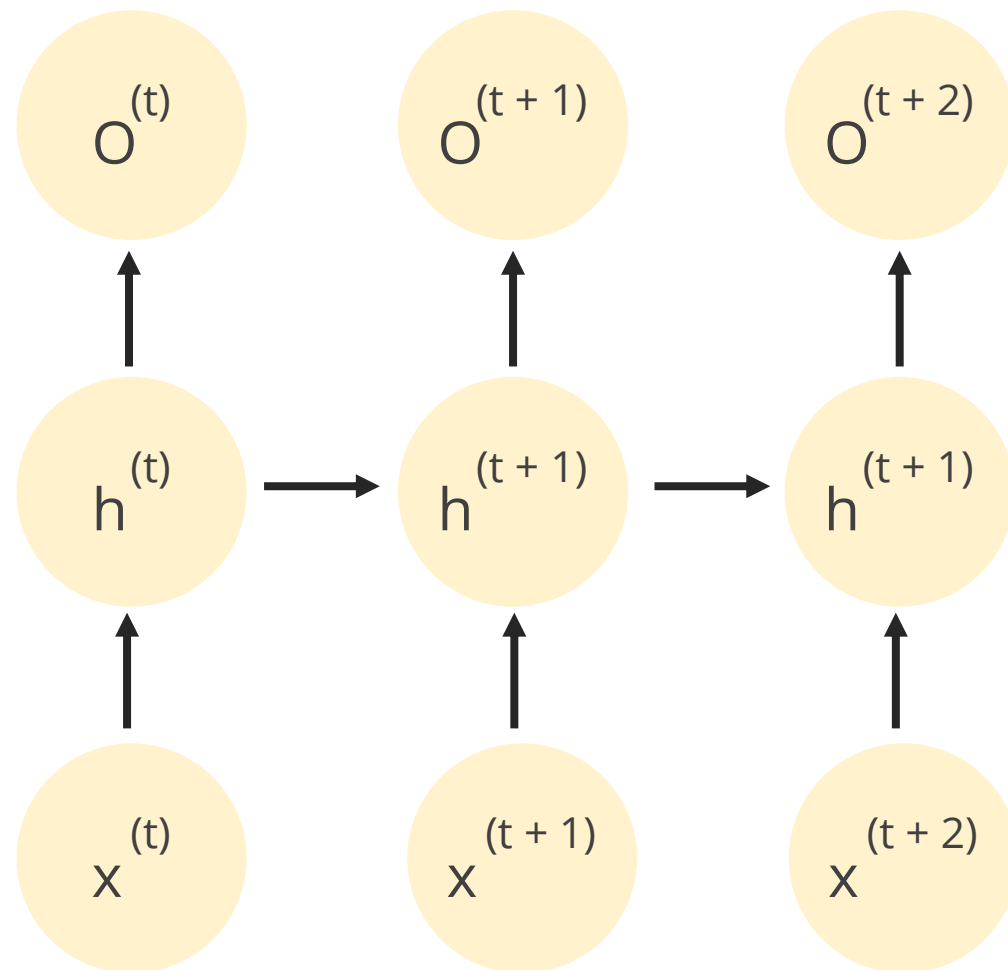
Example

An application for generating a string of musical symbols by giving the musical notes or chords as input

Such neural networks have two components: the encoder and the decoder.

Recurrent Neural Networks: Advantages

Some of the many advantages of RNN include:

The weights can be shared across time steps.

The model can process inputs of any length.

The model size isn't affected by the input size.

The model is designed to remember each piece of information and is helpful in any time-series predictor.

Recurrent Neural Networks: Disadvantages

Some of the disadvantages of RNN are:

Difficulty in training
RNN models

2

Computation slow due to its
recurrent nature

1

3

Issues with exploding and
vanishing gradients
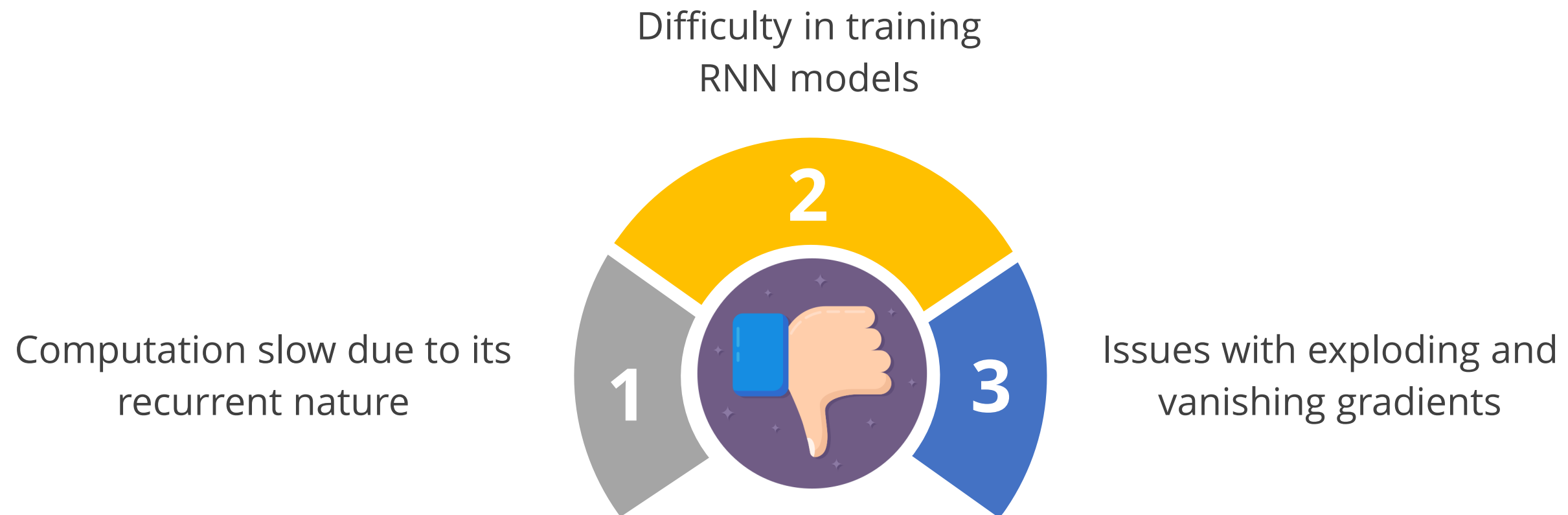
Architecture and Working of RNN

Architecture of RNN

The recurrent neural network (RNN) architecture is designed to process sequential data by capturing temporal dependencies and retaining the memory of previous inputs.
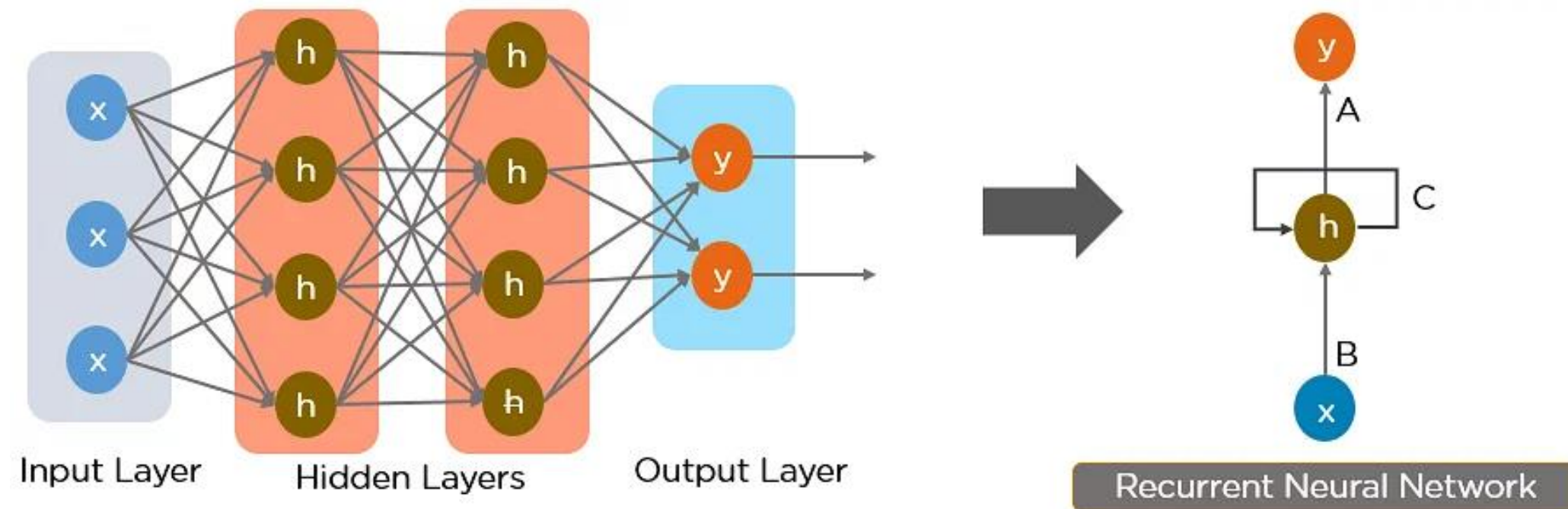
Architecture of RNN

It consists of three main components: an input layer, a hidden layer with recurrent connections, and an output layer.

Output layers   **y**

A

Hidden layers   **h**   C

B

Input layers   **x**

A, B and C are the parameters
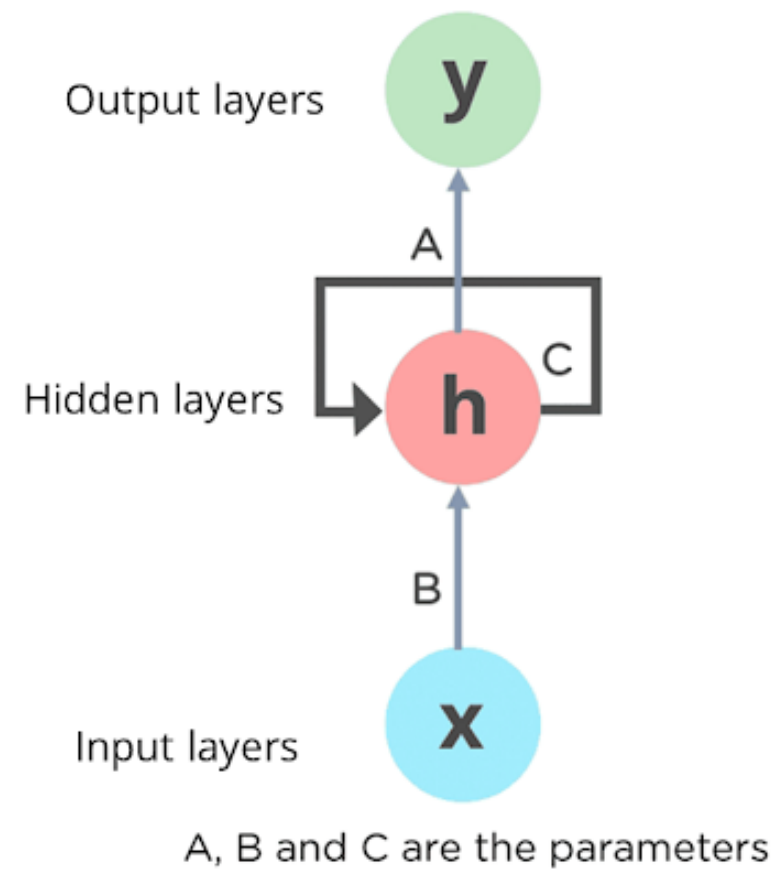
- x is an input layer; h is the hidden layer, and y is the output layer.

- Parameters A, B, and C are utilized to enhance the model's output.

- The output at each time step is fed back into the network to improve the subsequent outputs.

Working of RNN

The working of a recurrent neural network (RNN) can be described in a step-wise manner as follows:

Initialization

↓

Input processing

↓

Hidden state update

↓

Output calculation

↓

Training
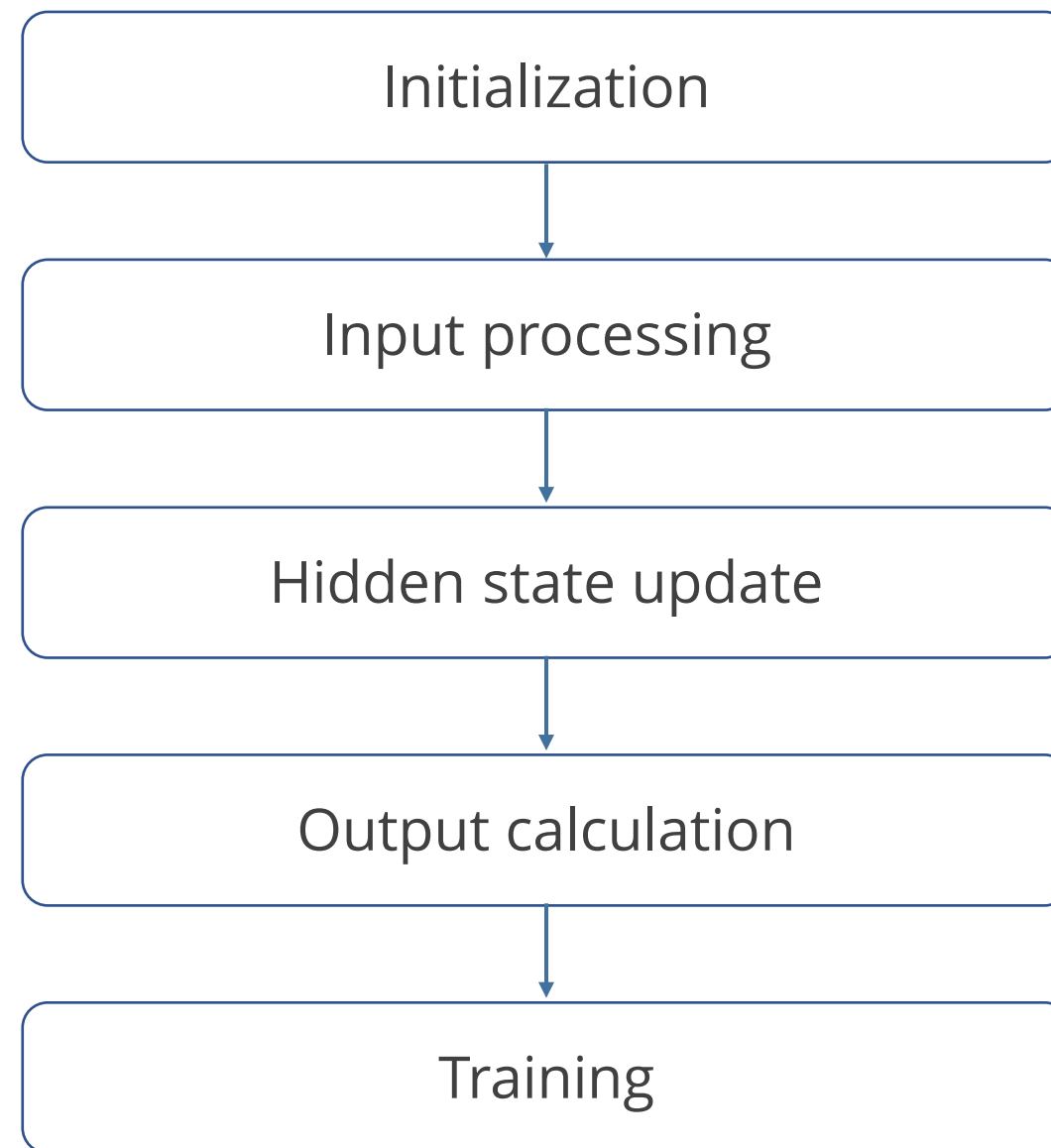
Initialization

It begins with initialization, where the hidden state is set to a vector of zeros or random values, and the RNN architecture is defined, including the number of hidden units, and activation functions.
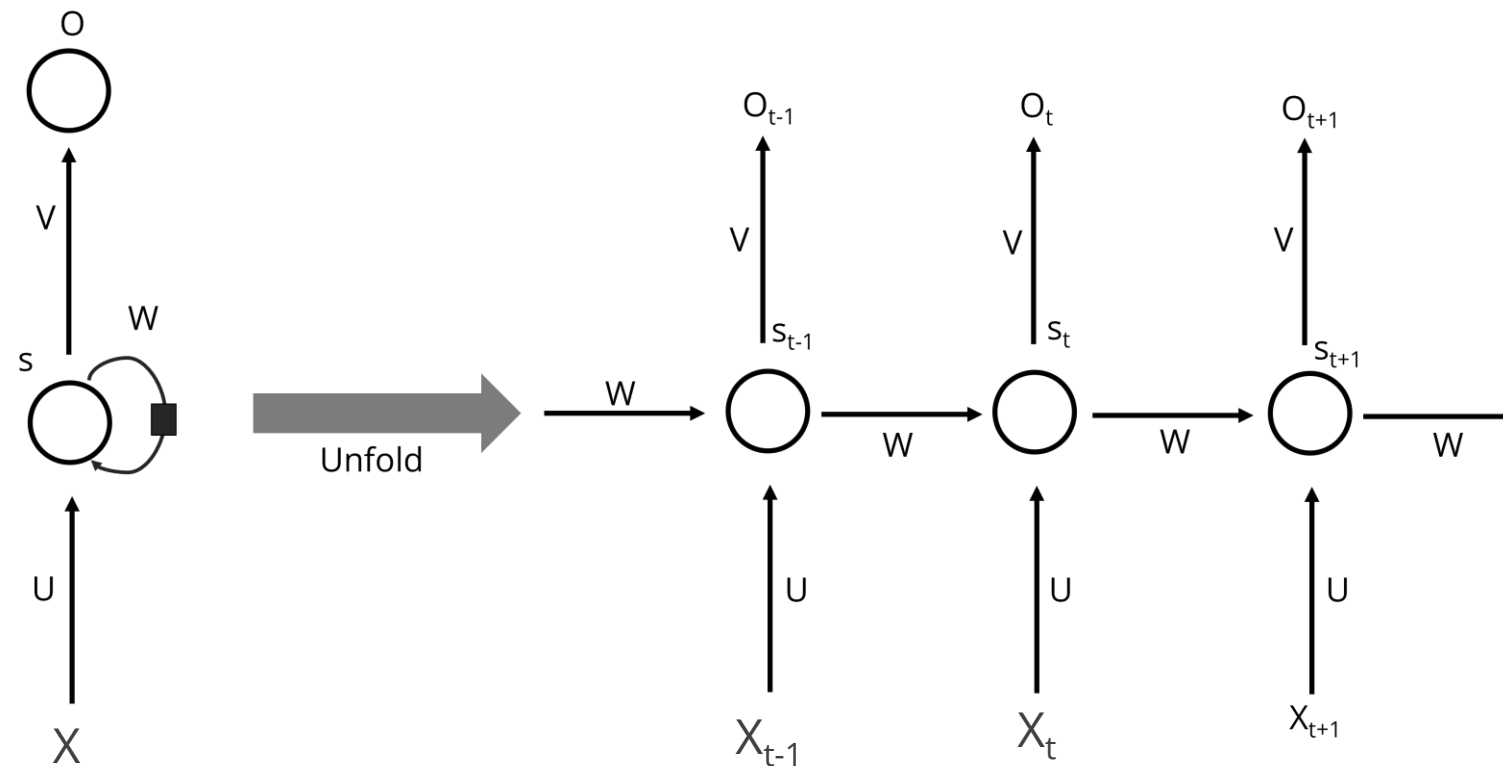
In the above diagram, X is the input state, s is the hidden state, and O is the output. The network's weights are U, V, and W.

Input Processing

During input processing, the input sequence is processed one element at a time.
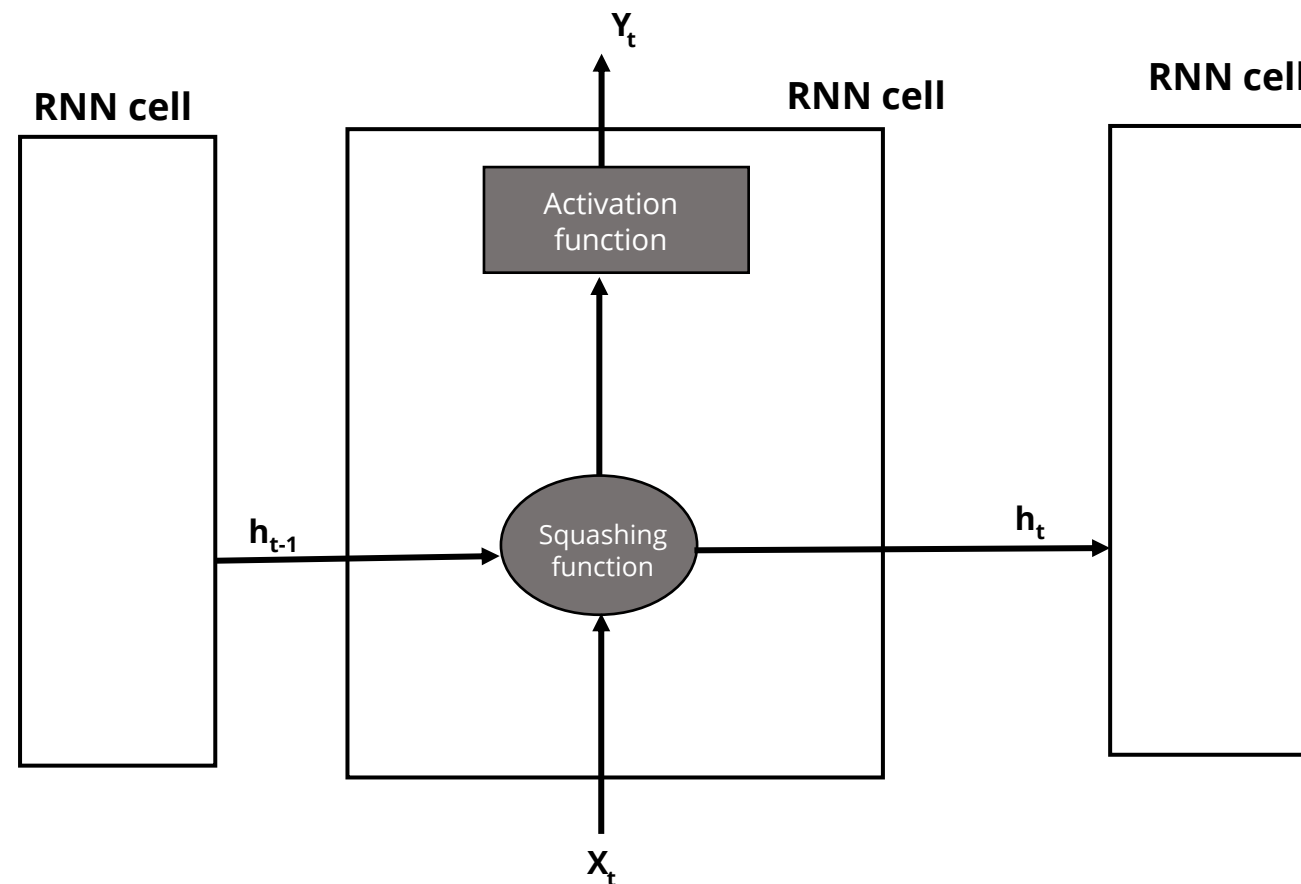
- X_t is the input
- h_t is the hidden state at time t
- h_{t-1} denotes hidden state from the previous time step t-1

The current input is combined with the previous hidden state to capture the context and dependencies, and the activation of the hidden layer is computed based on this combined input.

Input Processing

Use the following equation to combine the current input with the previous hidden state:

$$a_t = (W_{xh} * x_t + W_{hh} * h_{t-1}) + b_h$$

- W_{xh} represents the weight matrix for input-to-hidden connections.

- W_{hh} represents the weight matrix for hidden-to-hidden connections.

- x_t represents the current input.

- b_h represents the bias term for the hidden layer.

- a_t is the combined input to the activation function in the RNN cell, which determines the new hidden state $h_{t.}$.

Hidden State Update

The hidden state is then updated using the computed activation of the hidden layer, allowing information to flow across different time steps.

Compute the activation of the hidden unit using a hyperbolic tangent activation function.

Tanh

tanh(x)

X
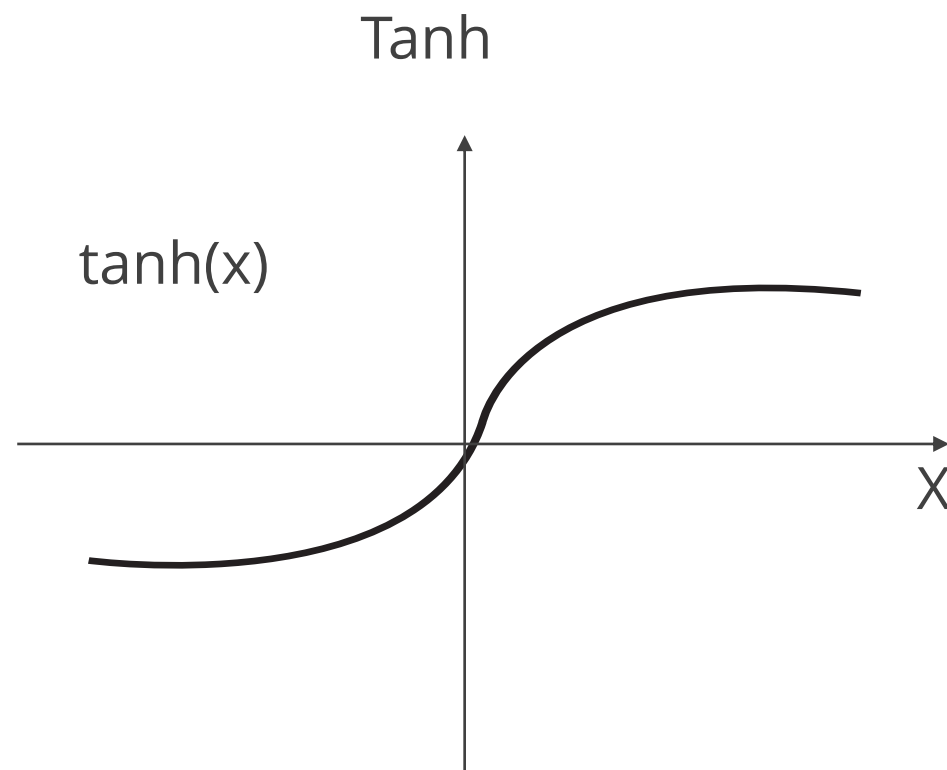
$$h_t = \tan h(W_{hh} \, h_{t-1} + W_{Xh} \, X_t + b_h + b_x)$$

h_t : Hidden state

W_{hh} : It is the weight matrix that determines the influence of the input X_t on the current hidden state h_t.

W_{Xh} : It is the weight matrix that determines the influence of the input X_t on the current hidden state h_t.

Output Calculation

The output of the network is calculated based on the updated hidden state, and any necessary transformations or activation functions are applied to obtain the desired output format.

$$Y_t = W_{hy} h_t + b_y$$

Here, Y_t denotes the output and W_{hy} denotes the output layer weights. b_y is the bias term for the output layer.

Training

During training, the RNN calculates the loss between the predicted output and the target output, and the weights and biases of the RNN are updated using backpropagation through time (BPTT) and an optimization algorithm.
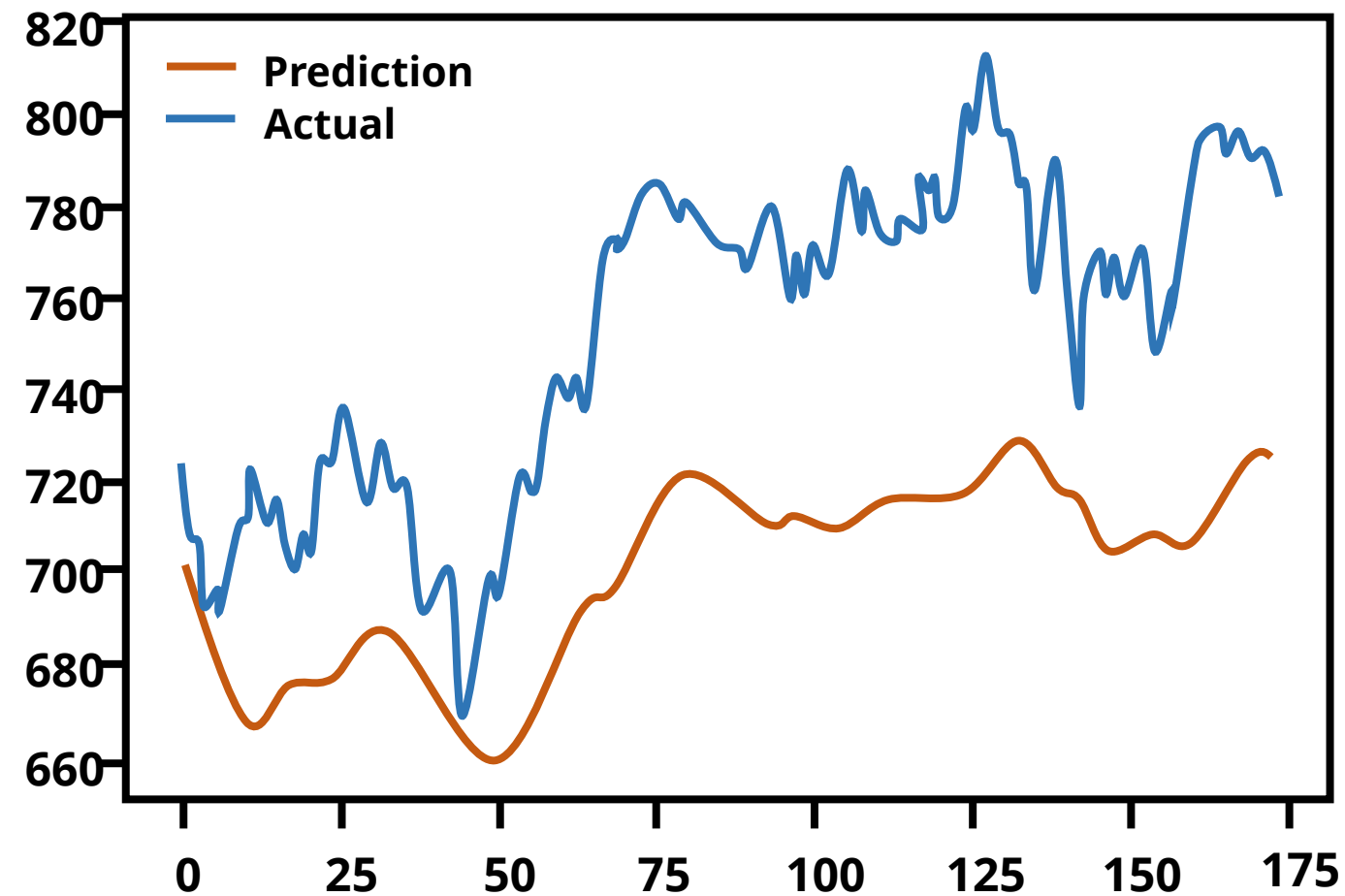
This process helps the RNN learn how to make more accurate predictions and improve its performance.

Assisted Practice

Let's understand the concept of text classification with RNN using Jupyter Notebooks.

- 11.05_Part_1_Text Classification Using RNN

Note: Please refer to the Reference Material section to download the notebook files corresponding to each mentioned topic

Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM)

When back-propagating, recurrent neural networks have issues with exploding and disappearing gradients.

Inputs

Input layer

Hidden layers

Output layer

Outputs
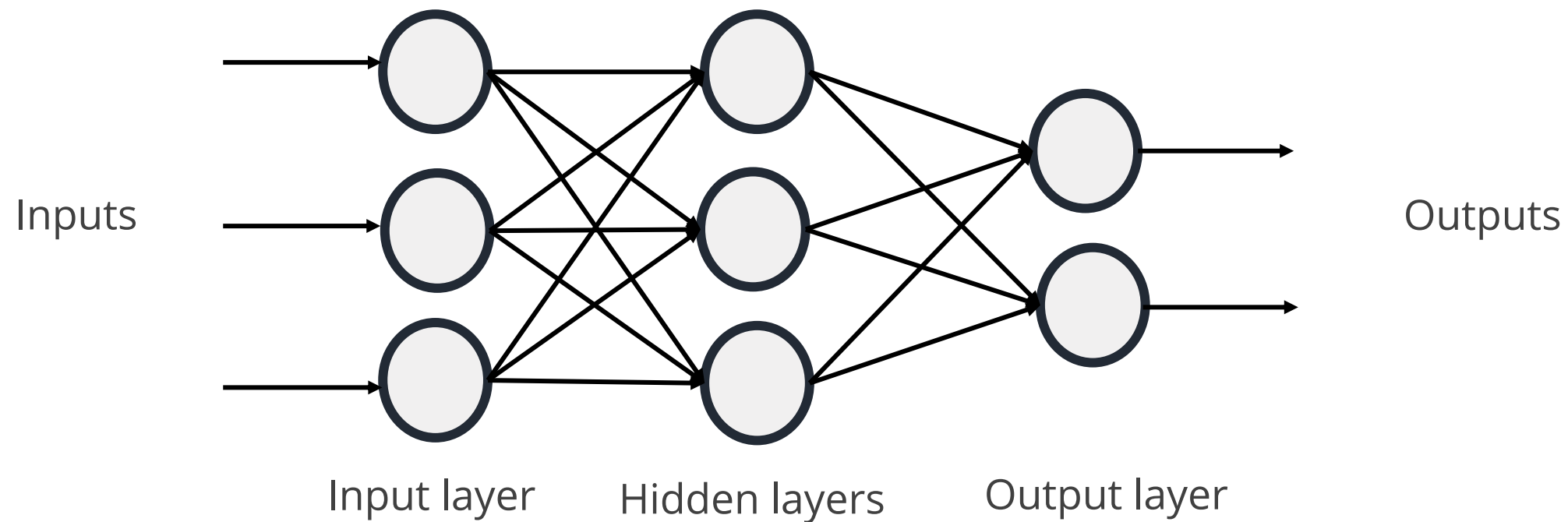
The modified versions of RNNs that help address the exploding and vanishing gradient problem are Long Short-Term Memory networks (LSTMs) and Gated Recurrent Units (GRUs).

Long Short-Term Memory (LSTM)

LSTM is an RNN architecture designed to capture and retain long-term dependencies in sequential data, overcoming the limitations of traditional RNNs.

Input gate

Forget gate

Output gate

Forget irrelevant
information

| 1 | 2 | 3 |

LSTM

Pass updated
information

Add or update new
information
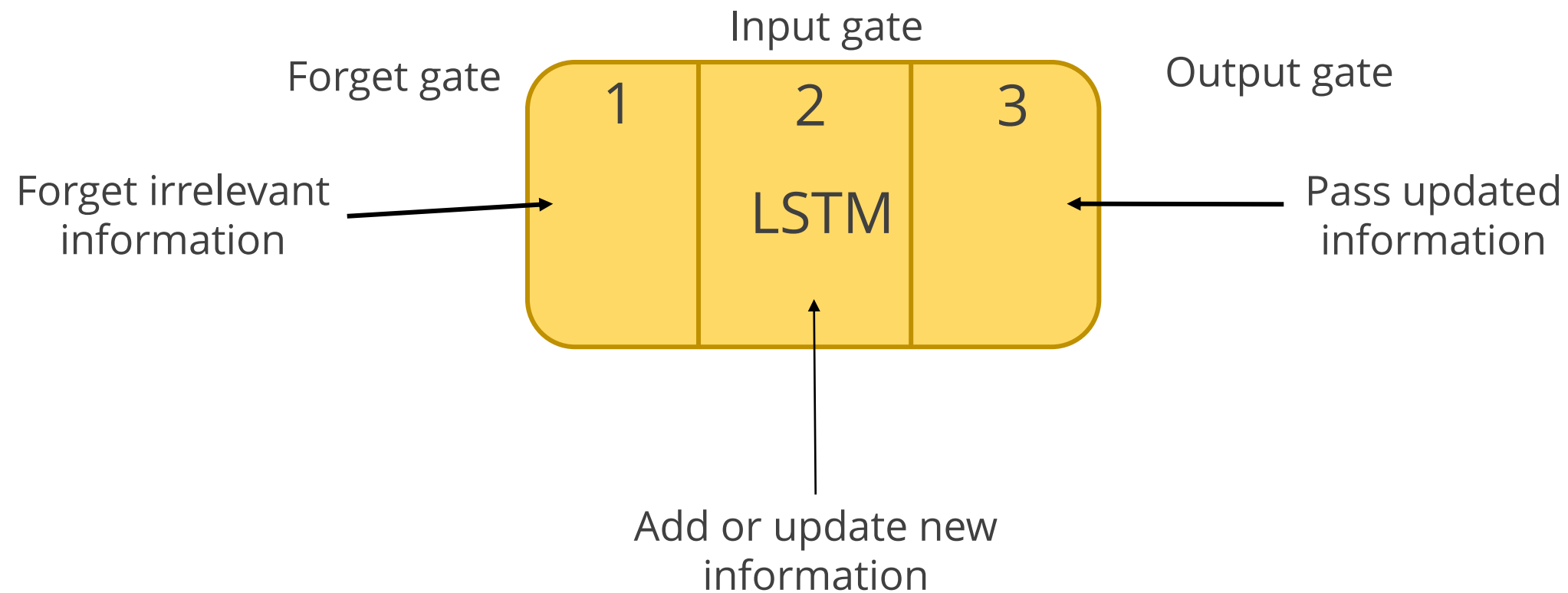
LSTM networks are widely used in tasks such as natural language processing, speech recognition, time series analysis, and more.

Long Short-Term Memory (LSTM)

LSTM utilizes three types of gates to control the flow of information within the network:

Input gate

Forget gate

| 1 | 2 | 3 |

LSTM

Output gate

Forget irrelevant information

Pass updated information

Add or update new information
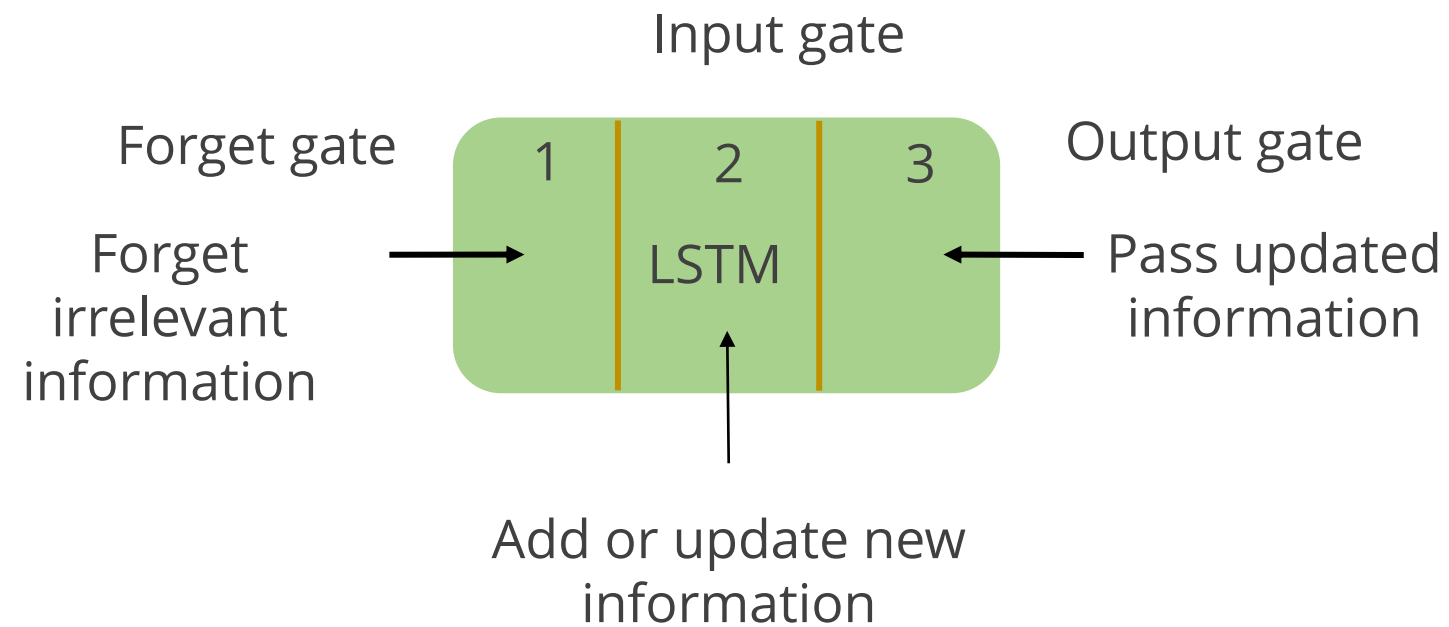
- **Forget gate:** Determines which information to discard from the previous memory cell state

- **Input gate:** Regulates which new information to incorporate into the current memory cell state

- **Output gate:** Controls the output generated by the memory cell

Hidden State

An LSTM, like a simple RNN, includes a hidden state.

Long Term Memory

C_{t-1}

Forget irrelevant Information

H_{t-1}

Short Term Memory

LSTM

1   2   3

add/update new Information

C_t

Pass updated Information

H_t
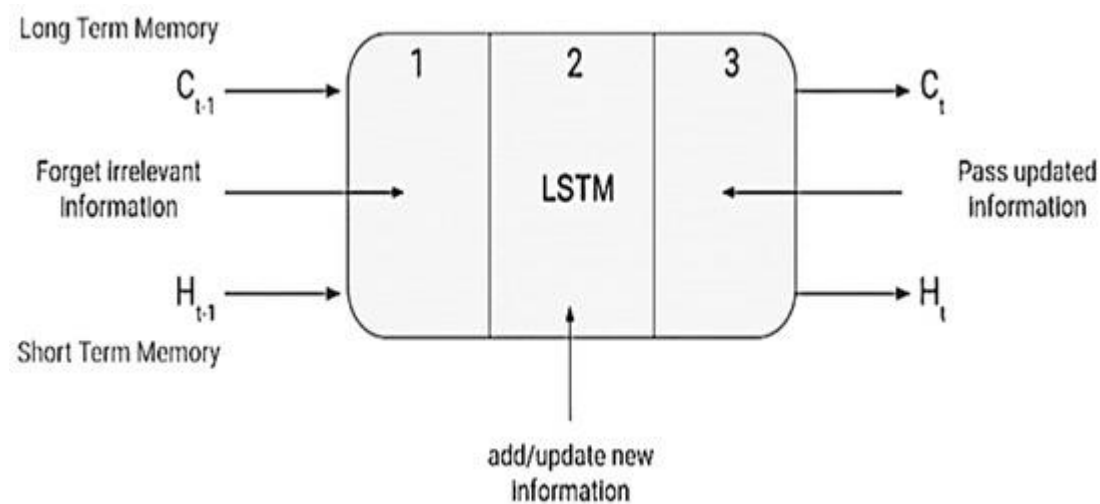
- The previous timestamp's hidden state is denoted as H_{t-1}, while the current timestamp's hidden state is denoted as H_t.

- In addition to the hidden state, LSTMs have a cell state.

- The cell state at the previous timestamp is represented as C_{t-1}, and at the current timestamp, it is represented as C_t.

- The cell state enables LSTMs to capture and retain long-term dependencies in sequential data.

Let's understand the concept of text classification with LSTM using Jupyter Notebooks.

- 11.07_Part_2 _Text Classification Using LSTM

Note: Please refer to the Reference Material section to download the notebook files corresponding to each mentioned topic

Gated Recurrent Network (GRU)

Gated Recurrent Network (GRU)

GRUs are a type of RNN architecture that incorporate gating mechanisms to regulate the flow of information between neural network cells.
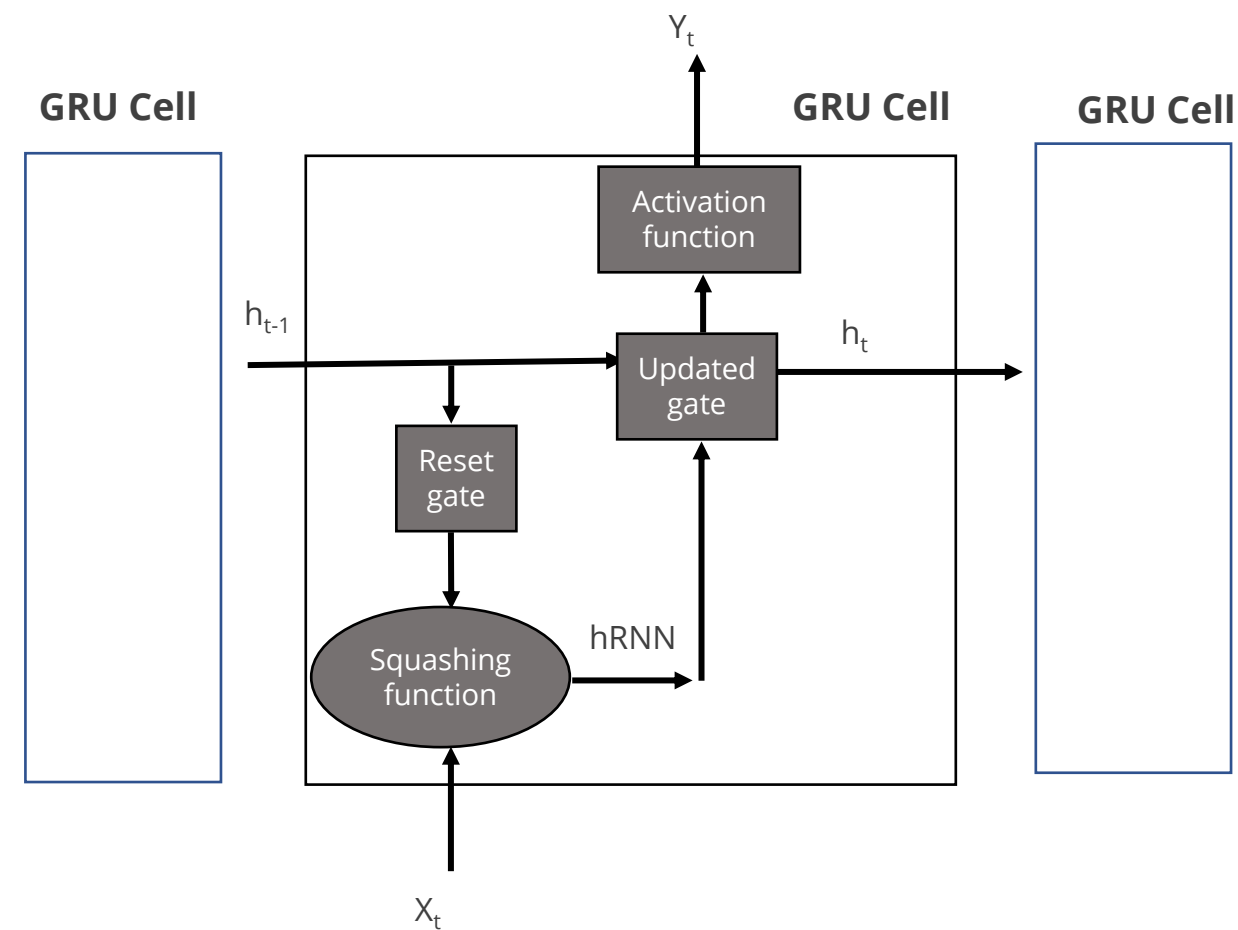
However, GRUs are relatively newer compared to LSTMs and have shown better performance with a simpler architecture.

Components of GRU

The update gate, like an LSTM's input and forget gate, determines whether the information is retained or discarded.

The reset gate allows you to control the relevance of the previous cell state by deciding the extent to which past data should be ignored.

The current hidden state (h_t) is computed by taking the Hadamard product of the update gate and the previously hidden state vector (h_{t-1}).

Introduction to Hybrid Modeling

Hybrid Modeling

It is the practice of employing two different neural network models and merging them to achieve a sequence of tasks.

One of its well-known implementations is the concept of convolutional recurrent neural networks (CRNN).
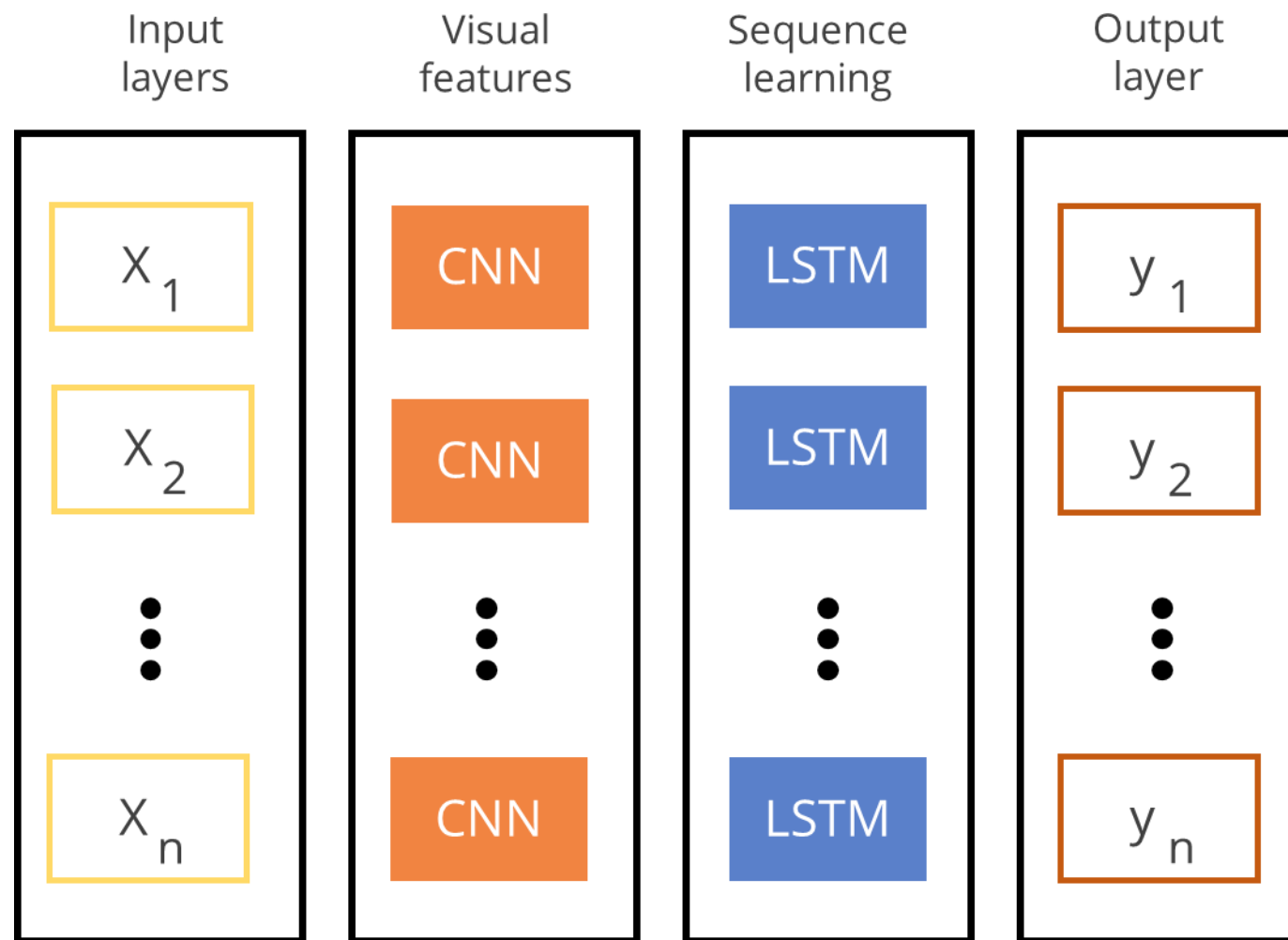
Hybrid Modeling

The recurrent network is used for temporal analysis, such as finding links between the extracted features that influence the output.
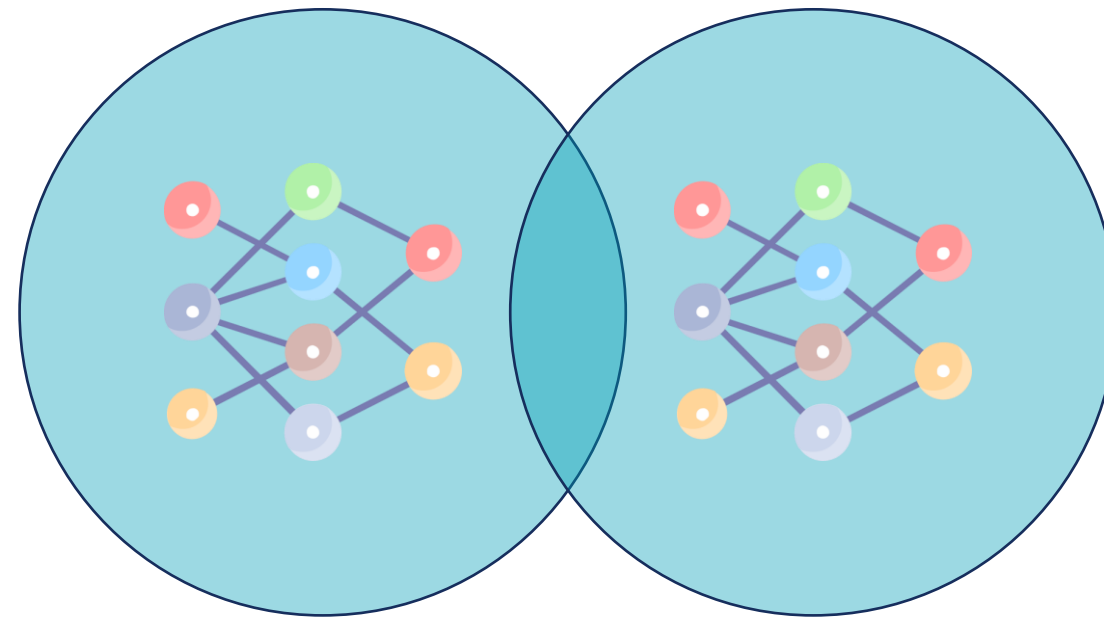
By merging the two networks, the machine can learn patterns in the sequential data provided and form appropriate predictions.

Hybrid Modeling: Example

A model predicts the next digit in a sequence of handwritten digits, and the input given is < 1, 2, 4, 8, ... >.

Input layers	Visual features	Sequence learning	Output layer
X_1	CNN	LSTM	y_1
X_2	CNN	LSTM	y_2
⋮	⋮	⋮	⋮
X_n	CNN	LSTM	y_n

The figure here shows it's working.

Hybrid Modeling: Example

It performs the following:

It slices the image of the digits into numerous segments.

The CNN first extracts the essential features of each segment, which are then passed into the RNN.

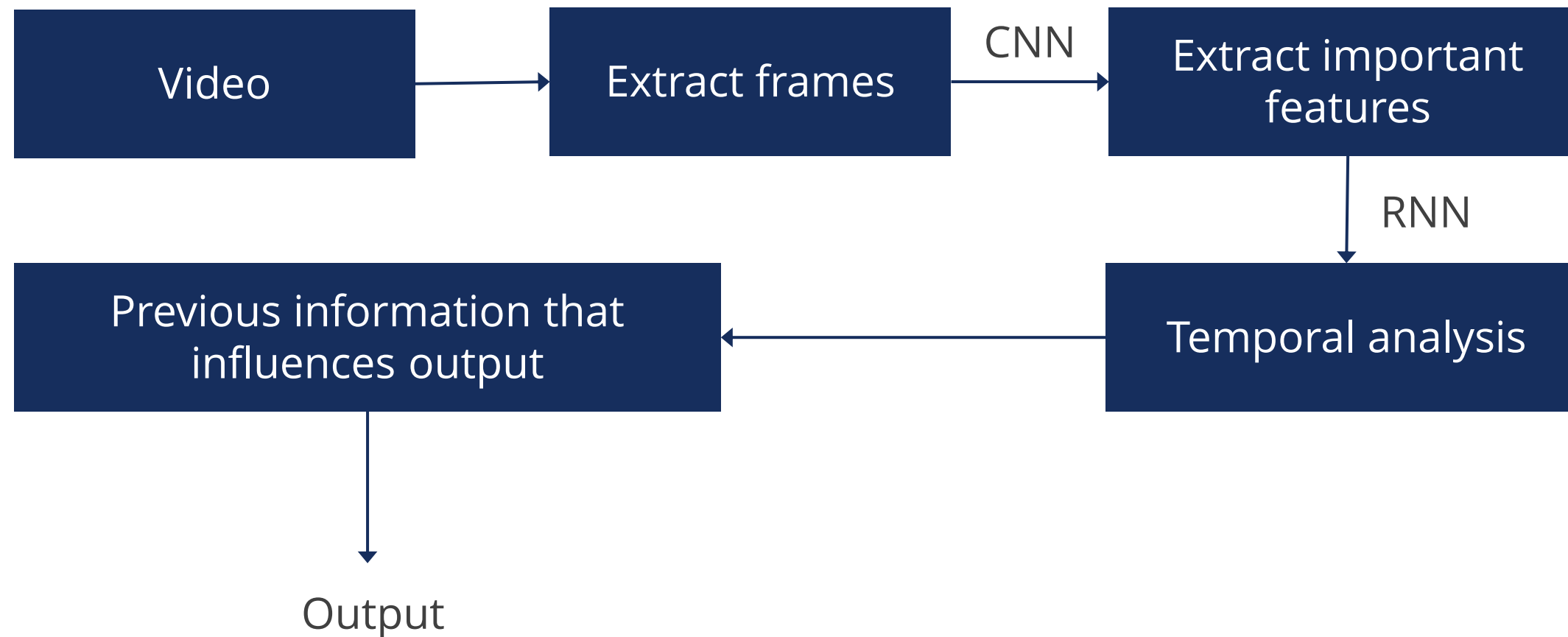The RNN recursively analyzes each feature, taking into consideration the previous input.

The algorithm decodes the output, and the model predicts 16 as the next digit in the sequence.

Applications of CRNN

Video Classification Using CRNN

Consider performing a video classification using the CRNN hybrid model:

A video is a series of images.

Video Classification Using CRNN

Extract frames and pass them to the CNN, which extracts the essential features in each frame.

Feed the features to the RNN, which analyzes them sequentially and considers the previous information to find links between them that influence the output.
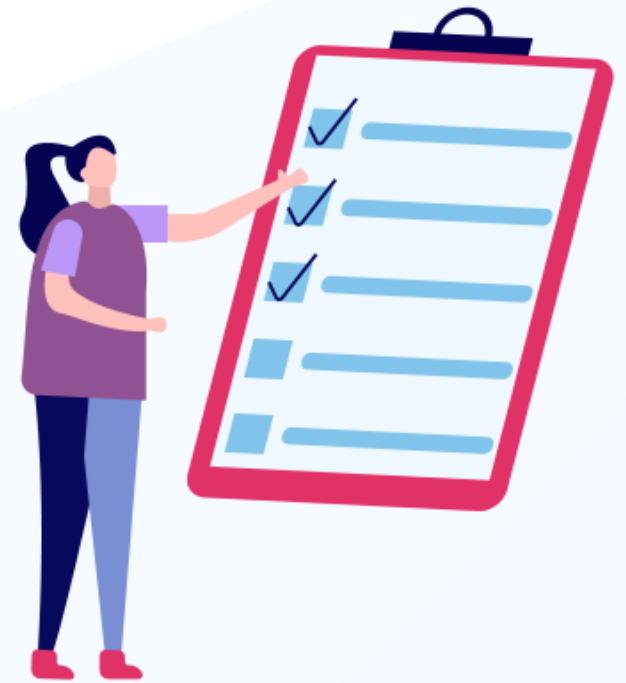
Let's understand the concept of video classification with the hybrid model using Jupyter Notebooks.

- 11.11_Video Classification Using Hybrid Model

Note: Please refer to the Reference Material section to download the notebook files corresponding to each mentioned topic

Key Takeaways

- A sequence modeling program can model, interpret, and predict various types of sequential data.

- The recurrent neural network is a state-of-the-art algorithm that adds the recent past to the present, allowing it to make precise predictions.

- A recurrent neural network has four types of architecture: one-to-one, one-to-many, many-to-one, and many-to-many.

- Hybrid modeling refers to employing two different neural network models and merging them to achieve a sequence of tasks.

Knowledge Check

What is sequential modeling?

A. A process used to analyze a series of input values to generate a sequence of values

B. A process used to analyze a series of output values to generate a sequence of input values

C. A process used to analyze a series of output values to generate a sequence of output values

D. A process used to analyze a series of input values to generate a sequence of input values

What is sequential modeling?

A.   A process used to analyze a series of input values to generate a sequence of values

B.   A process used to analyze a series of output values to generate a sequence of input values

C.   A process used to analyze a series of output values to generate a sequence of output values

D.   A process used to analyze a series of input values to generate a sequence of input values

The correct answer is **A**

Sequential modeling is a process used to generate a sequence of values by analyzing a series of input values.

Which neural network model is mostly used for spatial analysis, such as extracting essential features from an image?

A.   Recurrent neural network

B.   Convolutional neural network

C.   Dense neural network

D.   None of the above

Which neural network model is mostly used for spatial analysis, such as extracting essential features from an image?

A.   Recurrent neural network

B.   Convolutional neural network

C.   Dense neural network

D.   None of the above

The correct answer is **B**

The convolution network is mostly used for spatial analysis, such as extracting essential features from an image.

Thank You!