Recurrent Neural Networks

Learning Objectives:

- Define notation for building sequence models.
- Describe the architecture of a basic RNN.
- Identify the main components of an LSTM.
- Implement backpropagation through time for a basic RNN and an LSTM.
- Give examples of several types of RNN.
- Build a character-level text generation model using an RNN.
- Store text data for processing using an RNN.
- Sample novel sequences in an RNN.

Recurrent Neural Networks

Learning Objectives:

- Explain the vanishing/exploding gradient problem in RNNs.
- Apply gradient clipping as a solution for exploding gradients.
- Describe the architecture of a GRU.
- Use a bidirectional RNN to take information from two points of a sequence.
- Stack multiple RNNs on top of each other to create a deep RNN.
- Use the flexible Functional API to create complex models.
- Generate your own jazz music with deep learning.
- Apply an LSTM to a music generation task.

Recurrent Neural Networks

1 Why sequence models?
2 Notation
3 Recurrent Neural Network Model
4 Backpropagation through time
5 Different types of RNNs
6 Language model and sequence generation
7 Sampling novel sequences
8 Vanishing gradients with RNNs
9 Gated Recurrent Unit (GRU)
10 LSTM (long short term memory) unit
11 Bidirectional RNN
12 Deep RNNs

FPT Education

FPT UNIVERSITY

Recurrent Neural Networks

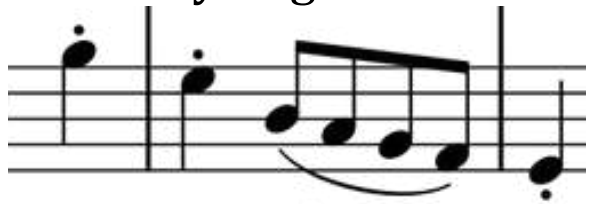
Why sequence models?

Why sequence models?

- Sequence models are useful for handling tasks involving sequential data, such as natural language processing, speech recognition, music generation, and more.
- Traditional neural networks do not have the ability to capture temporal dependencies in data, but sequence models, such as recurrent neural networks (RNNs) and long short-term memory (LSTM) networks, can process input data sequentially while maintaining an internal state that allows them to retain information from previous inputs.
- This makes sequence models well-suited to handle tasks where context and temporal relationships are important.

Examples of sequence data

FPT Education
FPT UNIVERSITY

Speech recognition		→	"The quick brown fox jumped over the lazy dog."
Music generation	∅	→	
Sentiment classification	"There is nothing to like in this movie."	→	★☆☆☆☆
DNA sequence analysis	AGCCCCTGTGAGGAACTAG	→	AGCCCCTGTGAGGAACTAG
Machine translation	Voulez-vous chanter avec moi?	→	Do you want to sing with me?
Video activity recognition		→	Running
Name entity recognition	Yesterday, Harry Potter met Hermione Granger.	→	Yesterday, Harry Potter met Hermione Granger.

FPT Education

FPT UNIVERSITY

Recurrent Neural Networks

Notation

Notation

- In sequence models, it's common to use notations to represent input and output sequences along with their indices.
- **X** represents the input sequence of words.
- **Y** represents the output sequence.
- $\mathbf{X}^{<t>}$ represents a specific word at position **t** in the input sequence.
- $\mathbf{Y}^{<t>}$ represents the corresponding label in the output sequence.
- $\mathbf{T}_x$ denotes the length of the input sequence.
- $\mathbf{T}_y$ denotes the length of the output sequence.

Movating Example

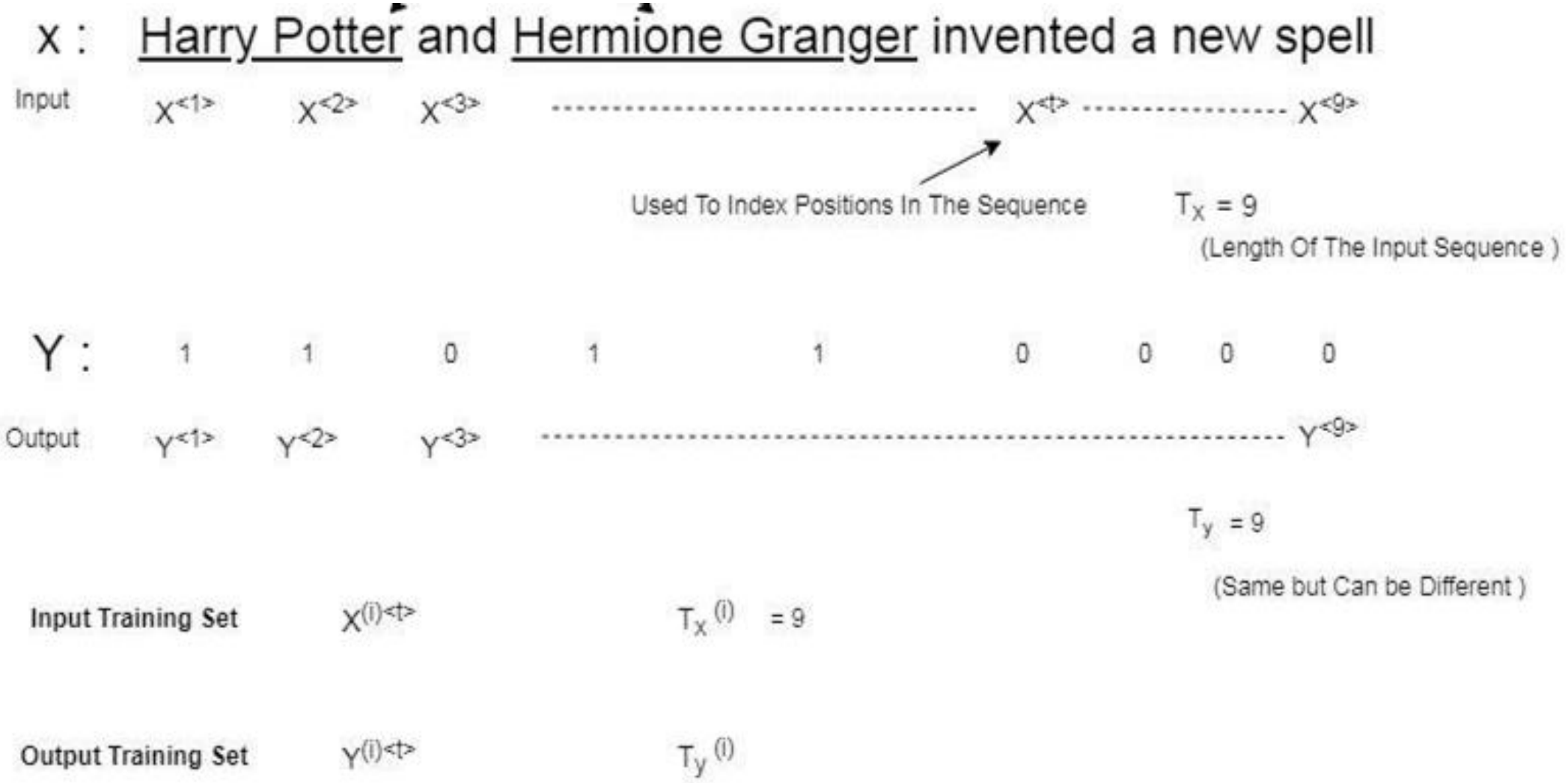
- Let's consider an example sentence: "**Harry Potter and Hermione Granger invented a new spell**“. Let say you want a sequence model to automatically tell you where are the peoples names in this sentence. So, this is a problem called Named-entity recognition (NER).

- X={"Harry", " Potter", "and", "Hermione", "Granger", "invented", “a“, “new“, “spell"}
- T_x = 9 (the length of the input sequence)
- Perform named-entity recognition where each word is classified as either a person's name (1) or other (0).
- Y = {1, 1, 0, 1, 1, 0, 0, 0, 0}
- T_y = 9 (the length of the output sequence)
- Using the provided notation:
- $X = (X^{<1>}, X^{<2>}, X^{<3>}, X^{<4>}, X^{<5>}, X^{<6>}, X^{<7>}, X^{<8>}, X^{<9>})$
- $Y = (Y^{<1>}, Y^{<2>}, Y^{<3>}, Y^{<4>}, Y^{<5>}, Y^{<6>}, Y^{<7>}, Y^{<8>}, Y^{<9>})$

Movating Example

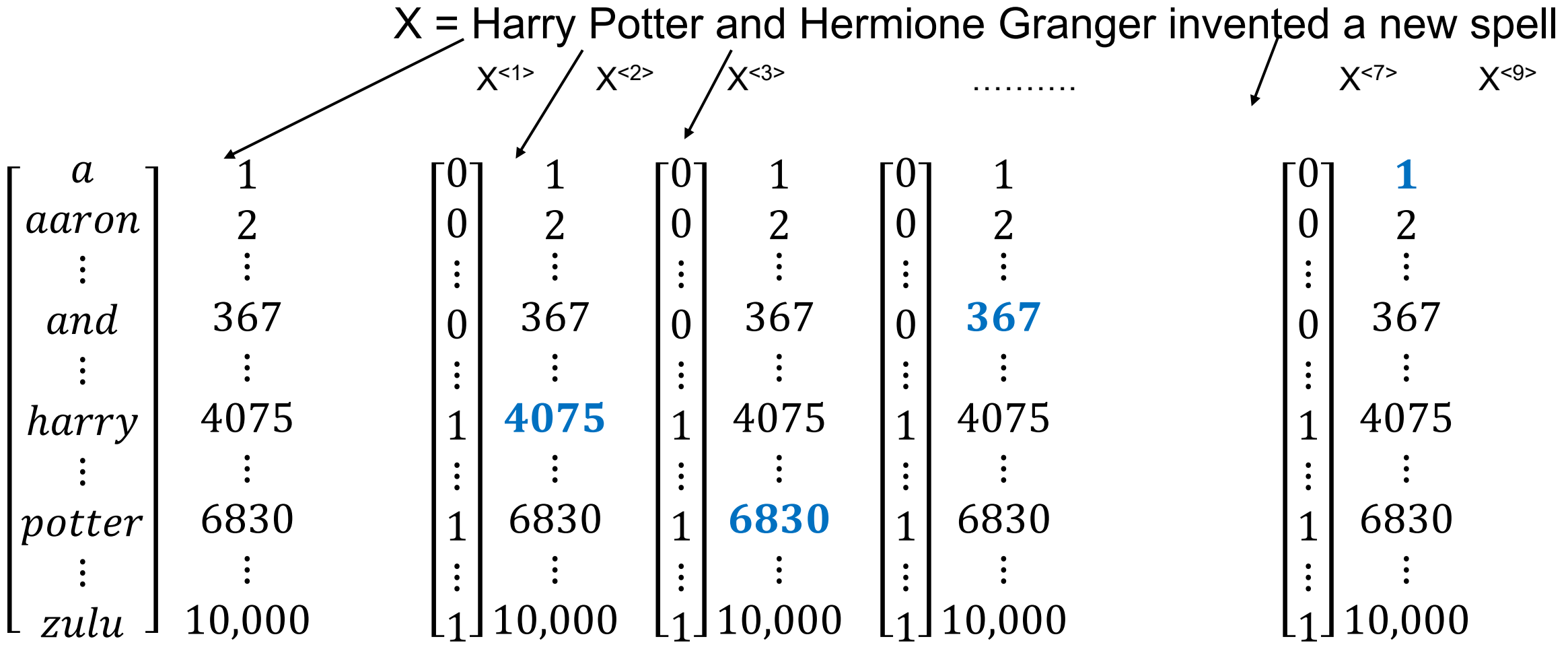
- For a specific word at position t:
- $X^{<t>}$ represents a word in the input sequence, e.g., $X^{<3>}$ = "and".
- $Y^{<t>}$ represents the corresponding label in the output sequence, e.g., $Y^{<4>}$ = 1.
- So, in this example:
- $X^{<3>}$ = “and"
- $X^{<4>}$ = "Hermione "
- $Y^{<3>}$ = 0
- $Y^{<4>}$ = 1

Motivating Example



Notation

- Dictionary:



Notation

- What to do if an unknown word is encountered?
  - In this case, a new token called "Unknown Word" or "UNK" is created to represent words not in the vocabulary.
- Overall, the basic notation and vocabulary representation, which are used in sequence models, will be used to build recurrent neural networks in the next section.

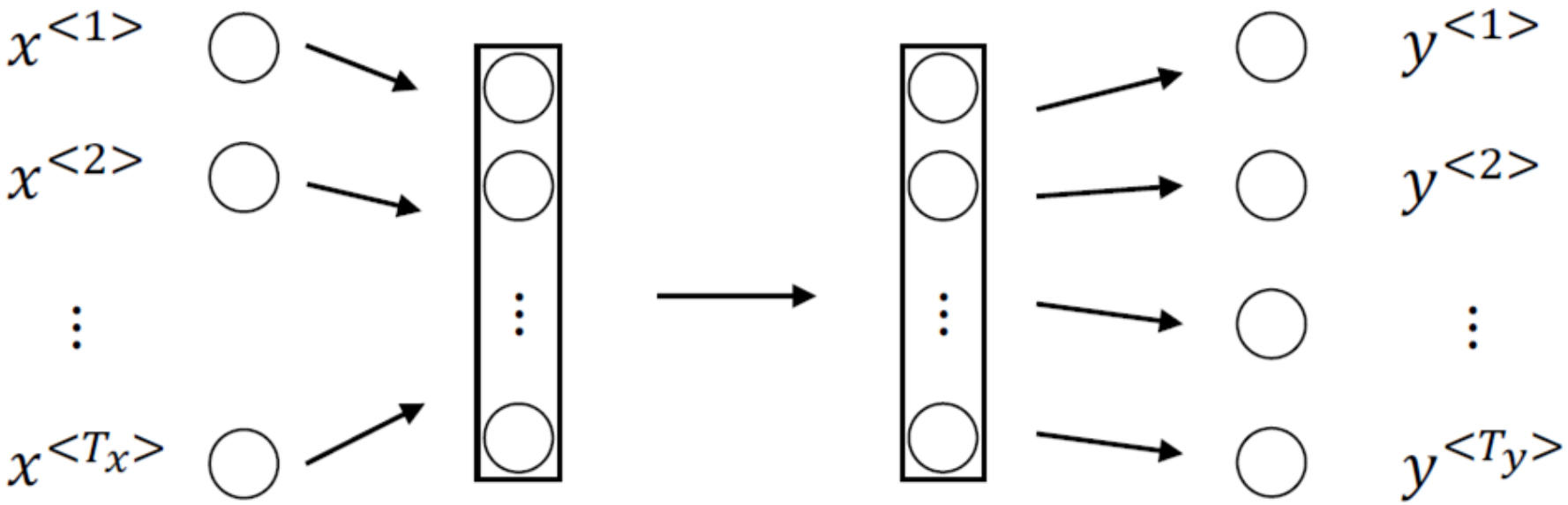
FPT UNIVERSITY

Recurrent Neural Network Model

Recurrent Neural Network Model

- Standard neural networks have limitations in handling varying input and output lengths and sharing learned features across text positions.
- RNNs address this with a looped structure, passing activations from previous to current time steps.
- Unidirectional RNNs use only past information, limiting some tasks. To overcome this, bidirectional RNNs are suggested.
- Forward propagation involves computing each time step's output using previous activations and current input. Notation is simplified for complex models.
- Backpropagation learns parameters by minimizing a loss function, detailed in a later section.

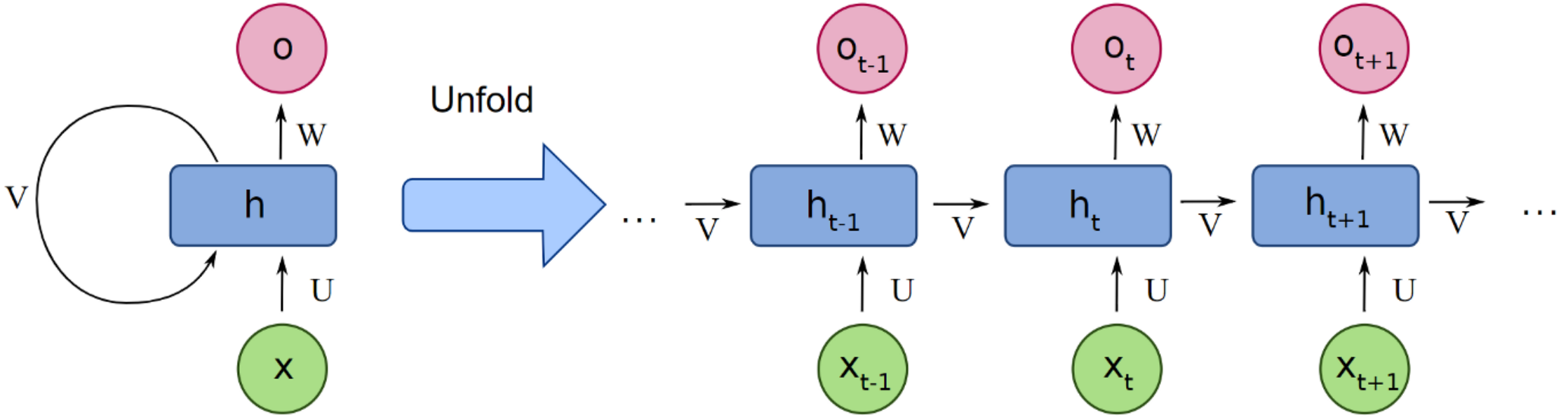
Why not a Standard Network



Problems:

- Inputs, outputs can be different lengths in different examples.
- Doesn't share features learned across different positions of text.

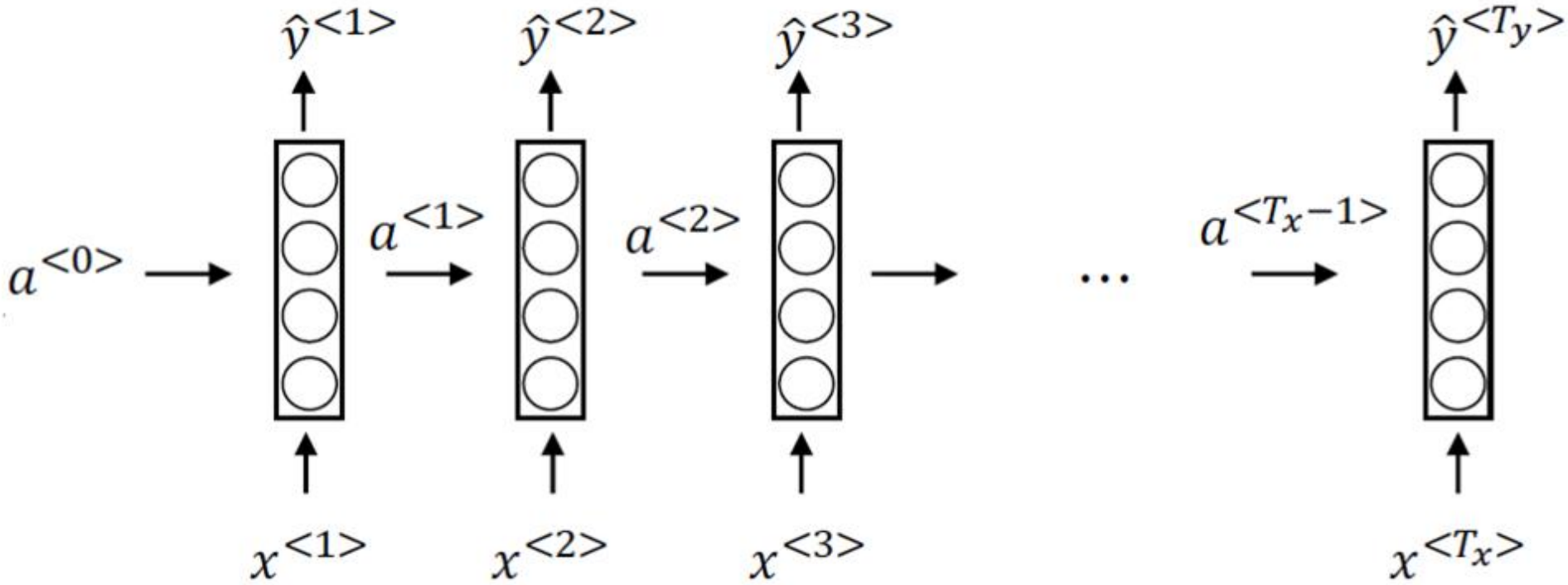
Recurrent Neural Networks



He said, “Teddy Roosevelt was a great President.”

He said, “Teddy bears are on sale!”

Forward Propagation



$$a^{<t>} = g(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a)$$

$$\hat{y}^{<t>} = g(W_{ya}a^{<t>} + b_y)$$

FPT Education

FPT UNIVERSITY

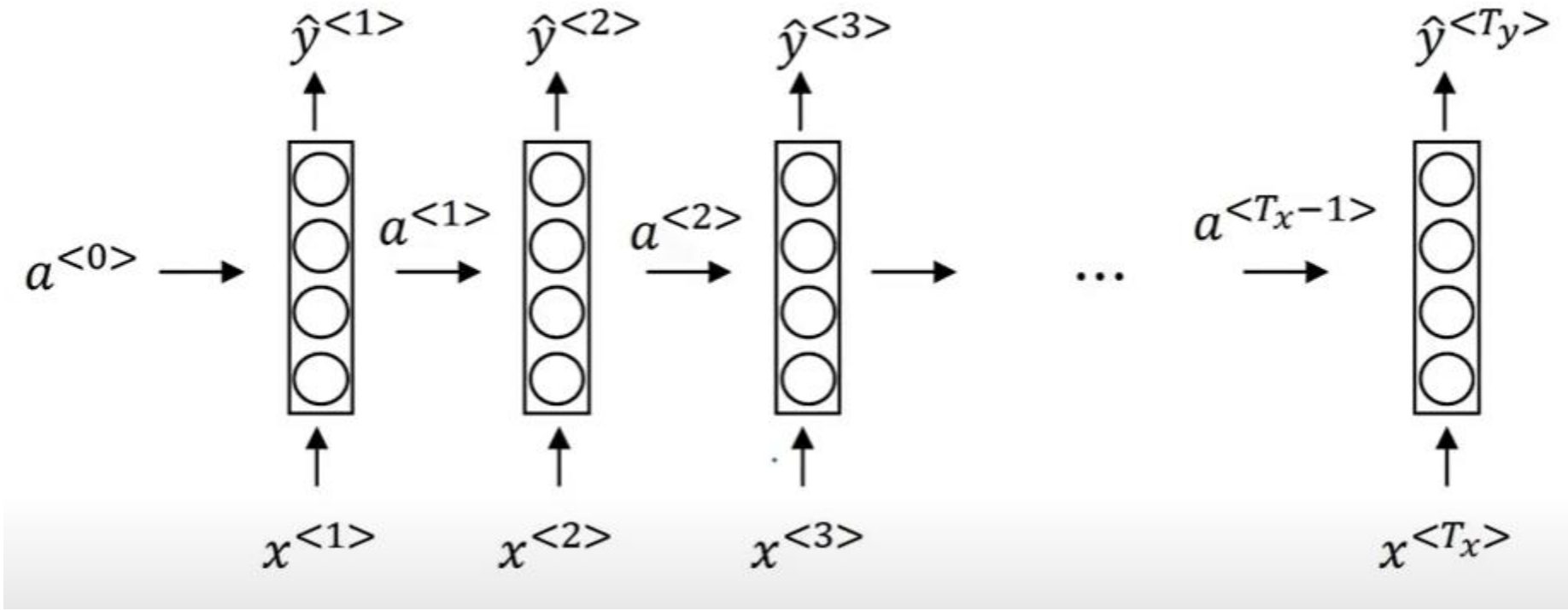
Recurrent Neural Networks

Backpropagation through time

Backpropagation through time

- How does backpropagation work in a recurrent neural network (RNN)?
  - The RNN has a basic structure where it takes an input sequence $x^{<1>}$, $x^{<2>}$, $x^{<3>}$,..., $x^{<Tx>}$ and computes activations $a^{<0>}$, $a^{<1>}$, $a^{<2>}$, $a^{<3>}$,..., $a^{<Tx-1>}$.
  - These activations are then used to compute predictions $\hat{y}^{<1>}$, $\hat{y}^{<2>}$, $\hat{y}^{<3>}$,..., $\hat{y}^{<Ty>}$. The parameters used to compute activations and predictions are W_a, b_a, W_y, and b_y.

Backpropagation through time



Backpropagation through time

- To compute backpropagation, a loss function is defined, which is the cross-entropy loss for a single prediction at a single timestep. The overall loss for the entire sequence is defined as the sum of the losses for all timesteps.
- Backpropagation in RNN works by carrying out computations in the opposite direction of the forward propagation arrows. This is called backpropagation through time (BPTT), as the algorithm goes from right to left, or backwards in time, to calculate the recursive calculations. The final step is to update the parameters using gradient descent.

- $\mathcal{L}^{\langle t \rangle}\left(\hat{y}^{\langle t \rangle}, y^{\langle t \rangle}\right) = -y^{\langle t \rangle} log \hat{y}^{<t>} + \left(1 - y^{\langle t \rangle}\right) \log\left(1 - \hat{y}^{\langle t \rangle}\right)$

- $\mathcal{L}(\hat{y}, y) = \sum_{t=1}^{Ty} \mathcal{L}^{\langle t \rangle}\left(\hat{y}^{\langle t \rangle}, y^{\langle t \rangle}\right)$

Backpropagation through time

- **Forward Pass:**
  - Input Sequence: At each time step t, the RNN receives an input x_t and the hidden state h_{t-1} from the previous time step.
  - Hidden State Update: The RNN computes a new hidden state h_{t-1} based on the input x_t and the previous hidden state h_{t-1}.
  - Output: The RNN produces an output y_t based on the current hidden state h_t.
- **Backward Pass:**
  - Loss Calculation: The output y_t is compared to the target output, and a loss is calculated using a loss function (e.g., mean squared error or cross-entropy).
  - Gradient Calculation: Gradients of the loss with respect to the parameters (weights and biases) are calculated using backpropagation. This involves calculating the gradients at each time step.
  - Weight Update: The model parameters are updated using an optimization algorithm (e.g., stochastic gradient descent) to minimize the loss.

Backpropagation through time

- Backpropagation Through Time:
  - Unrolling: BPTT involves unrolling the network through time, treating the sequence as a series of connected neural networks. This creates a computational graph where each time step corresponds to a layer.
  - Gradient Propagation: Gradients are propagated backward through the unrolled network, updating the parameters at each time step. The gradients are accumulated across time steps.
  - Truncation: To manage the exploding or vanishing gradient problem, the sequence is often truncated after a certain number of time steps. This helps in stabilizing the training process.
  - Parameter Update:
  - After completing the backward pass through the unrolled sequence, the accumulated gradients are used to update the model parameters in the optimization step.
- It's important to note that BPTT requires maintaining hidden states for each time step during the forward pass, which allows the model to capture information from the entire sequence.
- However, this also makes BPTT computationally expensive and can lead to issues like vanishing or exploding gradients, which may require additional techniques such as gradient clipping or using more advanced RNN architectures (e.g., LSTM or GRU) to address.

FPT Education

FPT UNIVERSITY

Recurrent Neural Networks

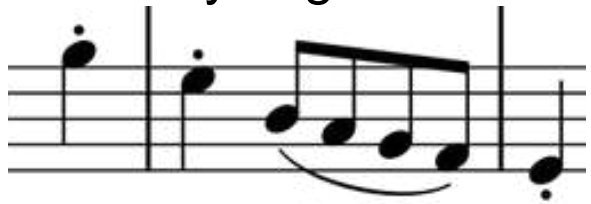
Different types of RNNs

Different types of RNNs

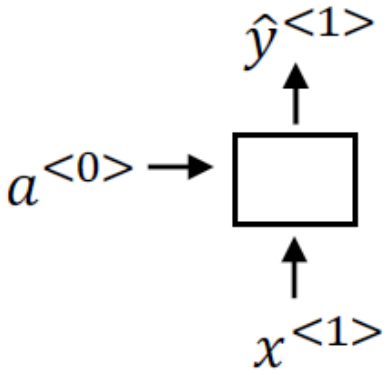
- Various RNN architectures serve specific purposes: one-to-one, one-to-many (e.g., music generation), many-to-one (e.g., sentiment classification), and many-to-many (e.g., machine translation with varying sequence lengths).
- RNNs' basic blocks enable diverse model creation, with sequence generation subtleties discussed later.

Examples of sequence data

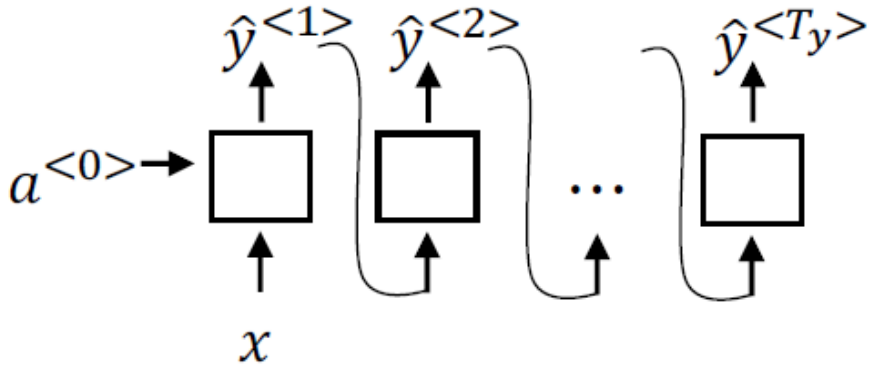
FPT Education
FPT UNIVERSITY

Speech recognition		→	"The quick brown fox jumped over the lazy dog."
Music generation	∅	→	
Sentiment classification	"There is nothing to like in this movie."	→	★☆☆☆☆
DNA sequence analysis	AGCCCCTGTGAGGAACTAG	→	AGCCCCTGTGAGGAACTAG
Machine translation	Voulez-vous chanter avec moi?	→	Do you want to sing with me?
Video activity recognition		→	Running
Name entity recognition	Yesterday, Harry Potter met Hermione Granger.	→	Yesterday, Harry Potter met Hermione Granger.

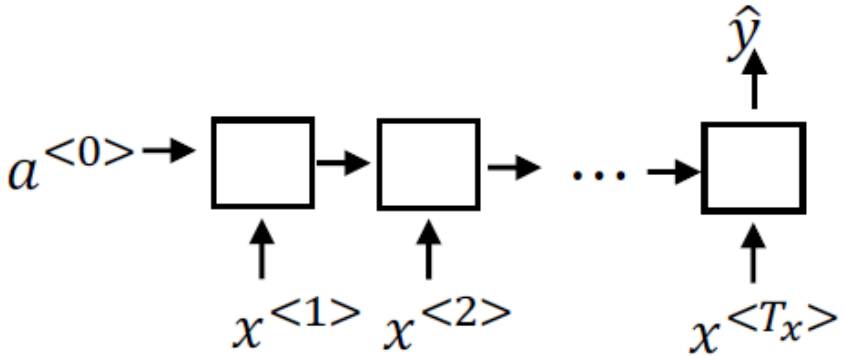
Examples of RNN architectures



One to one

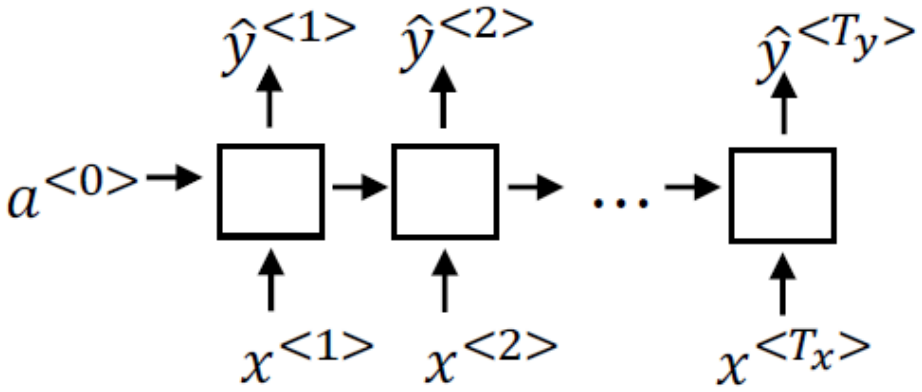


One to many

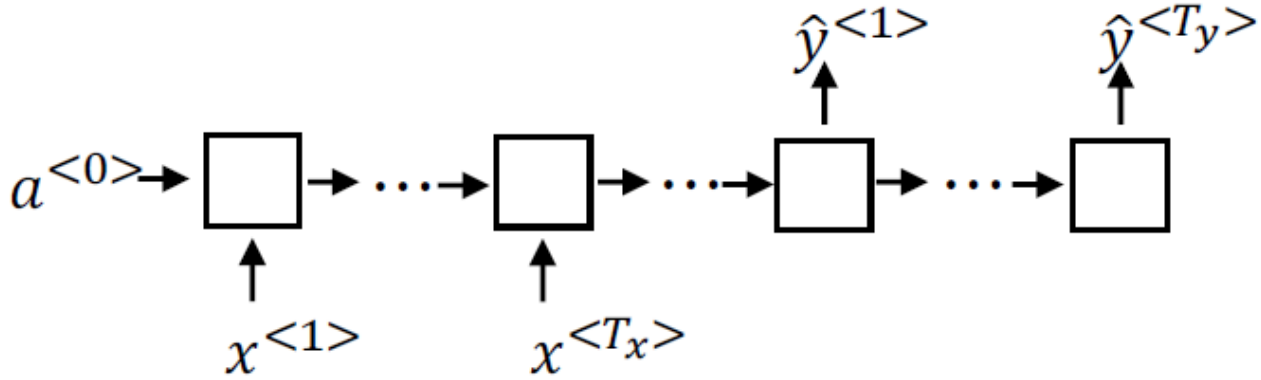


Many to one

Examples of RNN architectures

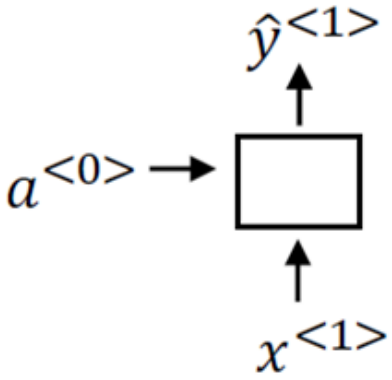


Many to many

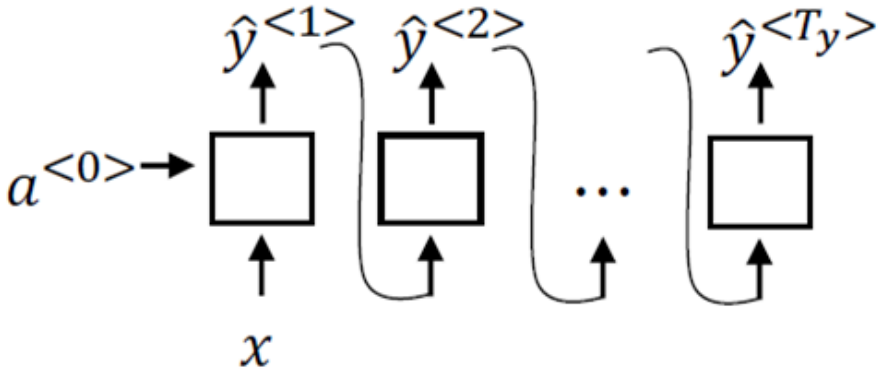


Many to many

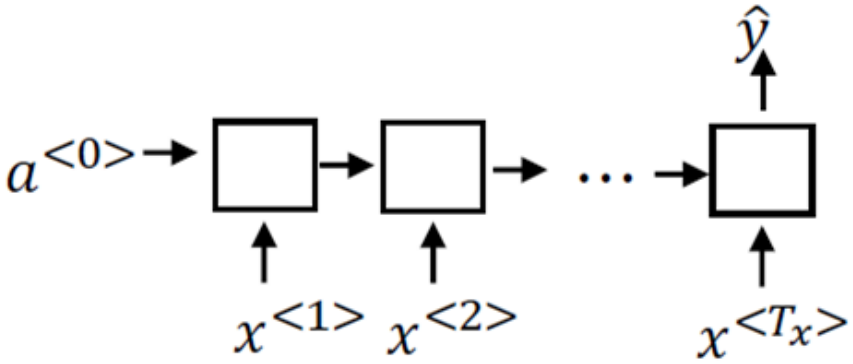
Summary of RNN types



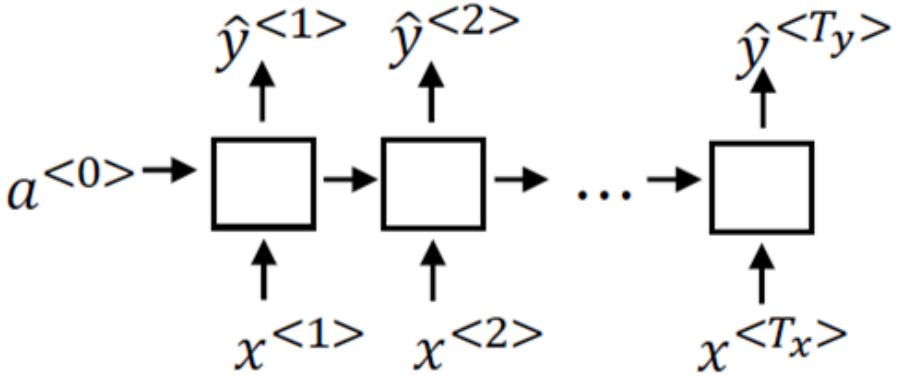
One to one



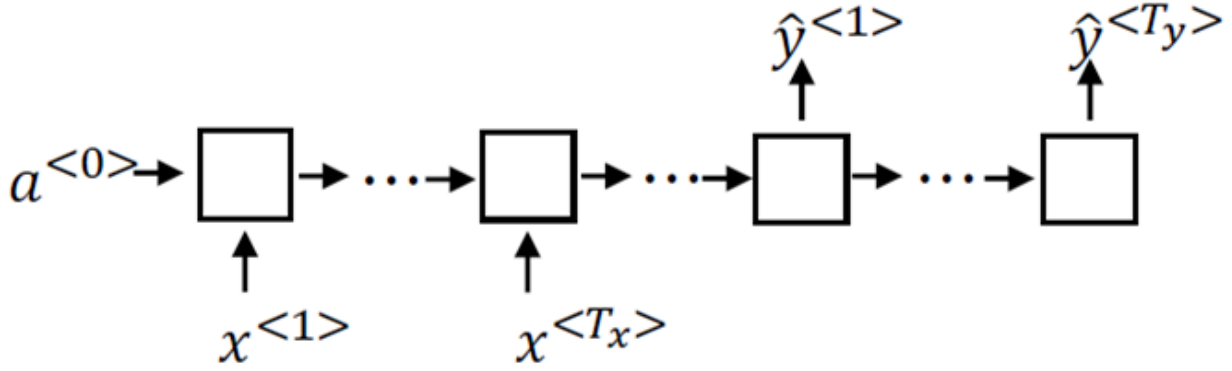
One to many



Many to one



Many to many



Many to many

FPT Education

FPT UNIVERSITY

Recurrent Neural Networks

Language model and sequence generation

Language model and sequence generation

- Language modeling predicts word sequence probabilities and is vital in tasks like speech recognition and machine translation. RNNs excel in building language models by capturing the sequential nature of language.
- Training involves tokenizing a text corpus, mapping words to vectors or indices.
- The RNN predicts the next word, learning sequentially from left to right. A cost function measures the difference during training.
- The trained model can generate new text by sampling the most probable next word at each step. This allows the model to produce text resembling the training set or generate creative content.

What is languge modelling

FPT Education
FPT UNIVERSITY

Speech recognition

The apple and pair salad.

The apple and pear salad.

$P(\text{The apple and pair salad}) =$

$P(\text{The apple and pear salad}) =$

Language modelling with an RNN

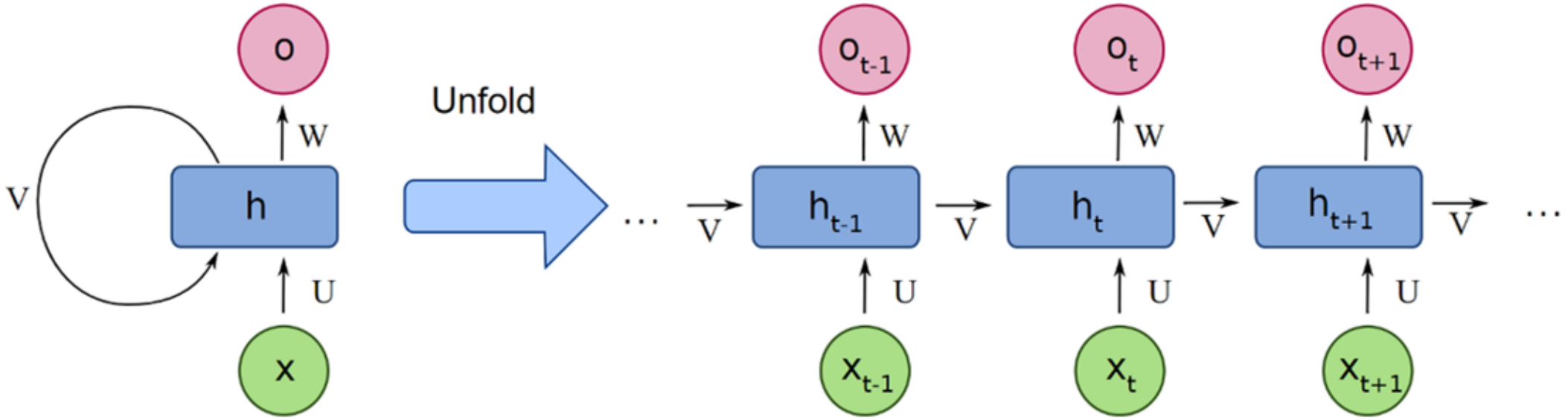
Training set: large corpus of english text.

Cats average 15 hours of sleep a day.

The Egyptian Mau is a bread of cat. <EOS>

< UNK >

RNN model



Cats average 15 hours of sleep a day. <EOS>

$$\mathcal{L}(\hat{y}^{<t>}, y^{<t>}) = -\sum_i y_i^{<t>} \log \hat{y}_i^{<t>}$$

$$\mathcal{L} = \sum_t \mathcal{L}^{<t>}(\hat{y}^{<t>}, y^{<t>})$$

FPT Education

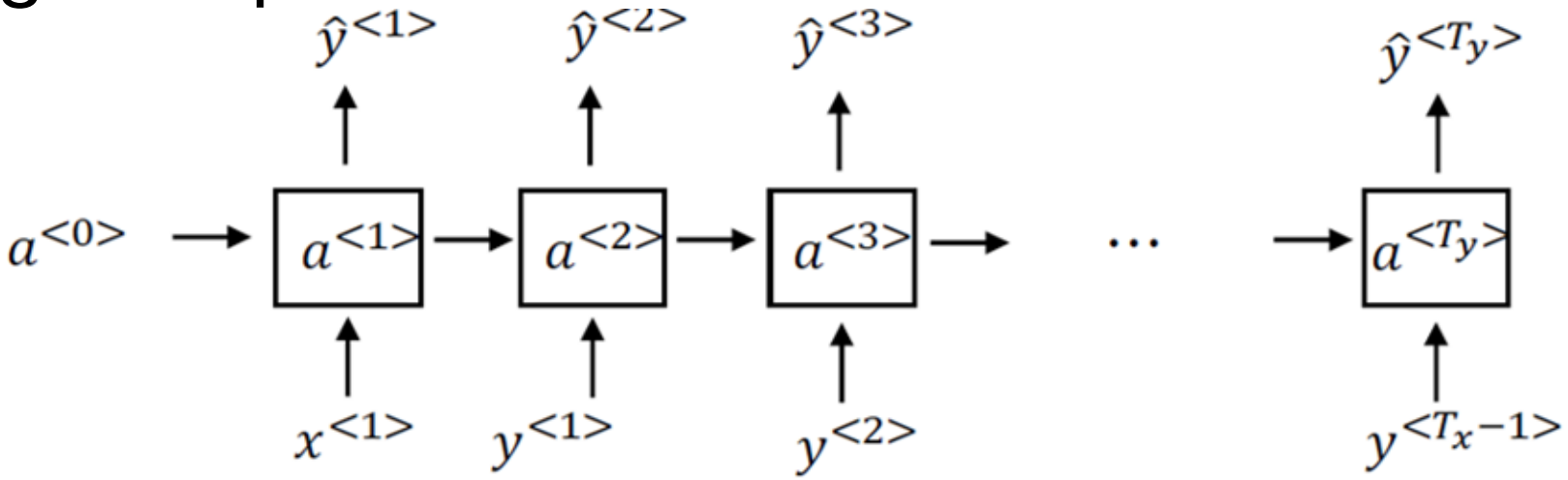
FPT UNIVERSITY

Sampling novel sequences

Sampling novel sequences

- After training, sequence models reveal learning by generating novel sequences through sampling predictions. This involves sampling the first word, passing it to subsequent steps, and repeating until sequence completion.
- Word-level models use word vocabularies, while character-level models use individual characters, avoiding unknown word tokens. However, they handle longer sequences and are computationally intensive.
- Powerful models like GRUs and LSTMs address training challenges, overcoming issues like vanishing gradients.

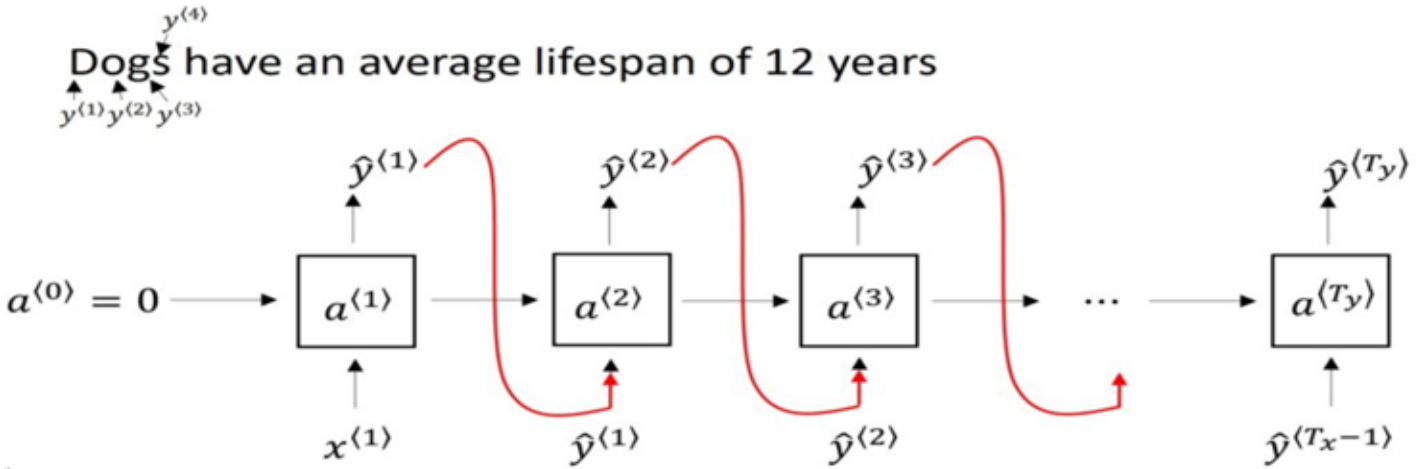
Sampling a sequence from a trained RNN



Vocabulary = [a, Aaron, ... zulu, < UNK]

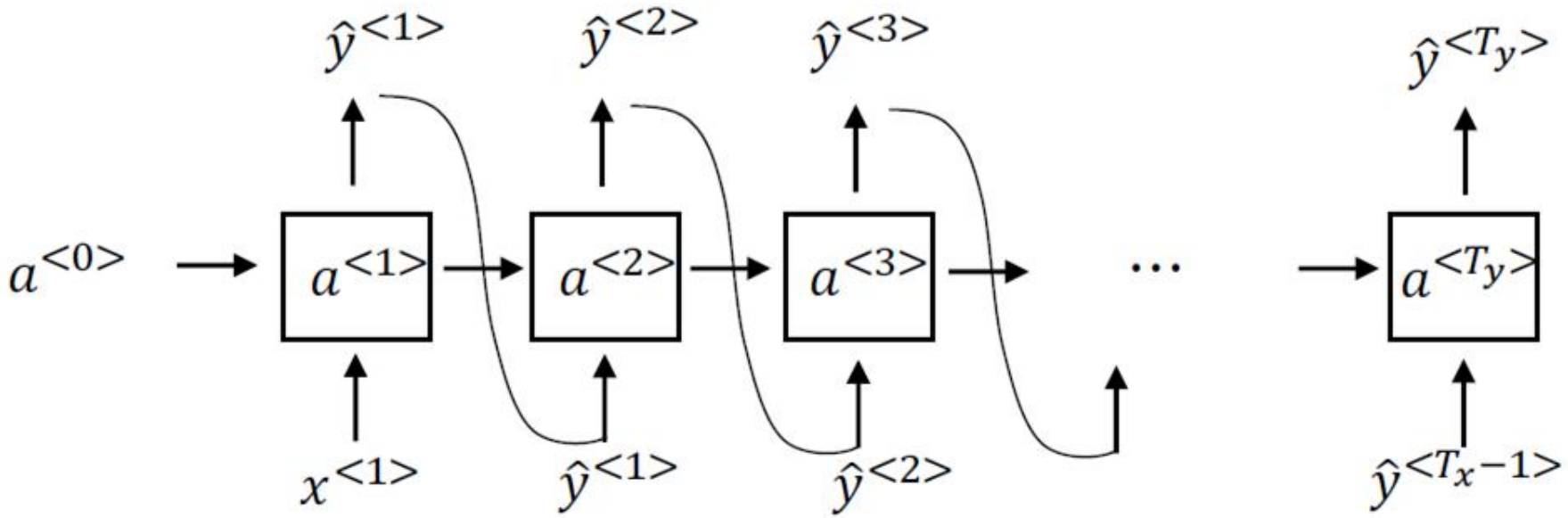
Vocabulary = [a, b, c, ... z, _, . , , , 0, ... , 9, A, ... , Z]

$y^{\langle 1 \rangle}$ $y^{\langle 2 \rangle}$ $y^{\langle 3 \rangle}$ -individual characters



Character-level language model

Vocabulary = [a, aaron, …, zulu, <UNK>]



Sequence generation

FPT Education
FPT UNIVERSITY

News

President enrique peña nieto, announced sench's sulk former coming football langston paring.

"I was not at all surprised," said hich langston.

"Concussion epidemic", to be examined.

The gray football the told some and this has on the uefa icon, should money as.

Shakespeare

The mortal moon hath her eclipse in love.

And subject of this thou art another this fold.

When besser be my love to me see sabl's.

For whose are ruse of mine eyes heaves.

FPT UNIVERSITY

Recurrent Neural Networks

Vanishing gradients with RNNs

Vanishing gradients with RNNs

- In deep neural network training, the vanishing gradient problem hinders backward propagation, especially in RNNs where forward propagation is left to right and backpropagation is right to left.
- This challenge makes it hard for errors in later time steps to influence earlier computations, limiting the model's ability to capture long-range dependencies.
- In very deep networks, exploding gradients can occur, but vanishing gradients pose a more significant issue in RNN training.
- While exploding gradients are noticeable, vanishing gradients are harder to solve.
- Advanced RNN models like GRUs and LSTMs address the vanishing gradient problem, enabling the capture of longer-range dependencies.

FPT UNIVERSITY

Recurrent Neural Networks

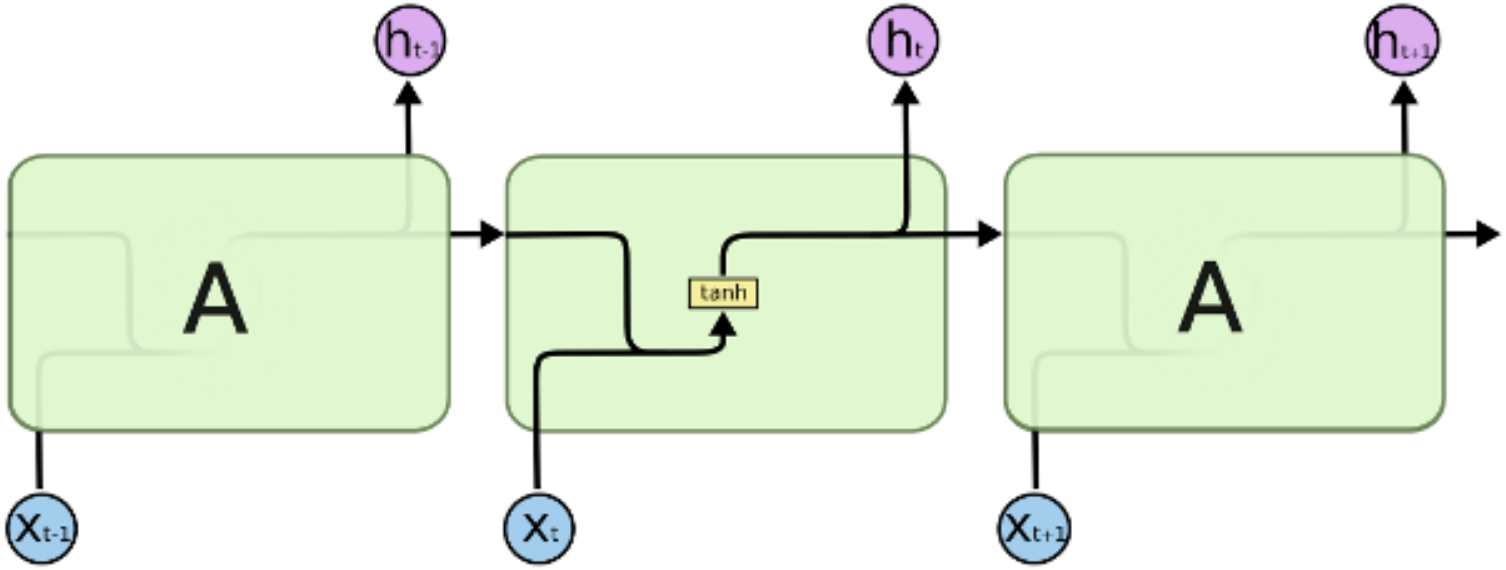
Gated Recurrent Unit (GRU)

Gated Recurrent Unit (GRU)

FPT Education
FPT UNIVERSITY

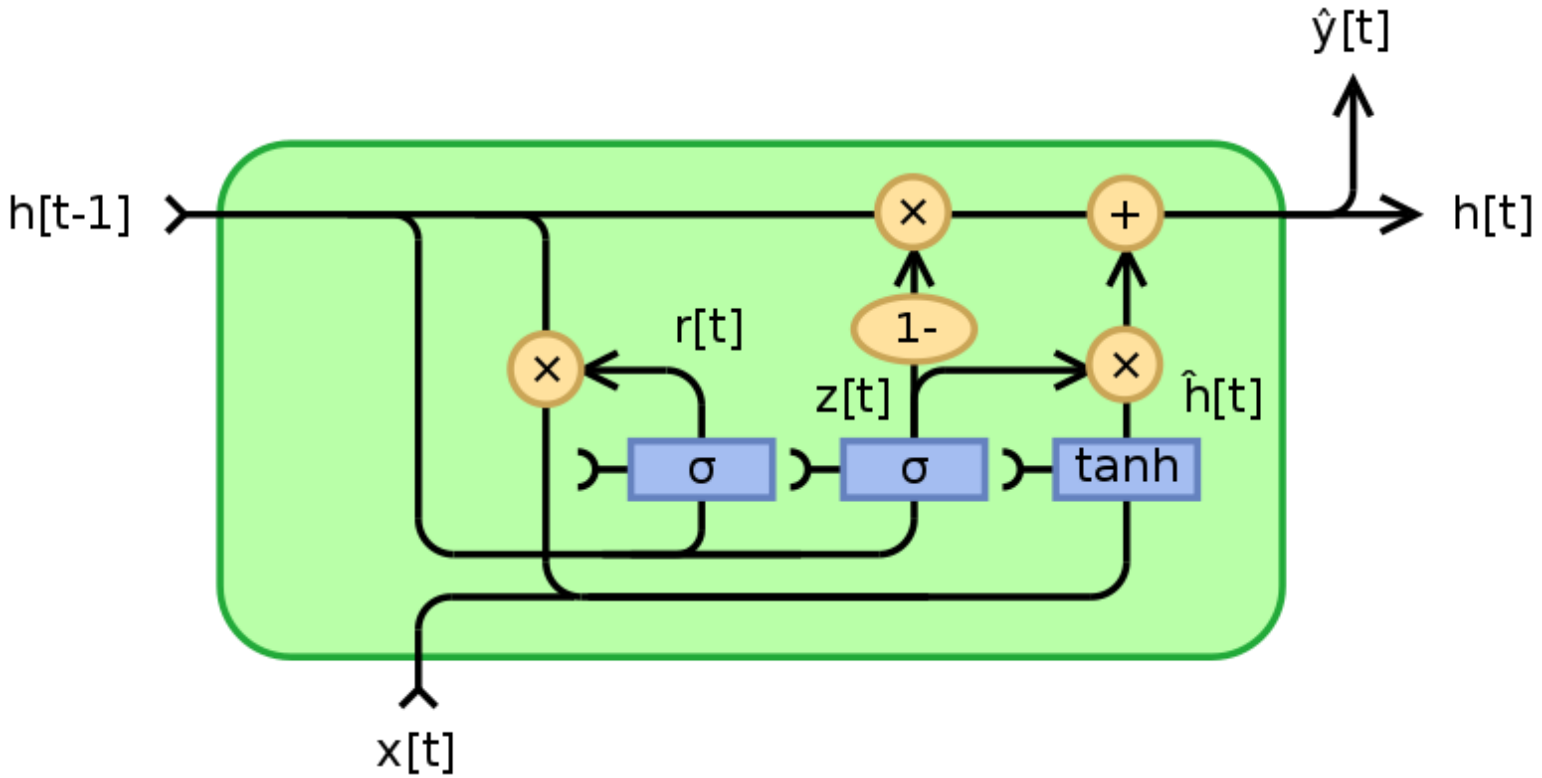
- The Gated Recurrent Unit (GRU) enhances the standard Recurrent Neural Network (RNN) hidden layer to address vanishing gradient problems and capture long-range connections.
- It introduces a memory cell (C) to retain essential information from previous timesteps. A gate, ranging from 0 to 1, decides whether to update the memory cell. The update gate determines when to update, and the relevance gate gauges the importance of the previous memory cell. The memory cell's actual value is computed using the gates and candidate new and old values.
- GRUs and LSTMs, both derived from these concepts, are widely used in neural network design for their effectiveness.

RNN unit



$$a^{<t>} = \tanh g(W_a[a^{<t-1>}, x^{<t>}] + b_a)$$

GRU (simplified)



The cat, which already ate …, was full.

Full GRU

$$\tilde{c}^{<t>} = \tanh(W_c[\, c^{<t-1>}, x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_u[\, c^{<t-1>}, x^{<t>}] + b_u)$$

$$\Gamma_r = \sigma(W_r[\, c^{<t-1>}, x^{<t>}] + b_r)$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) + c^{<t-1>}$$

The cat, which ate already, was full.

FPT UNIVERSITY

LSTM (long short term memory) unit

LSTM (long short term memory) unit

- The Long Short-Term Memory (LSTM) unit is another type of recurrent neural network (RNN) that allows for learning long-range dependencies in a sequence. Compared to the Gated Recurrent Unit (GRU), the LSTM is even more powerful and flexible due to its three gates: an **input** gate, an **output** gate, and a **forget** gate.
- **Input Gate**: The input gate determines which values from the current input and the previous hidden state should be updated and added to the cell state.
- **Forget Gate**: The forget gate determines which information from the cell state should be discarded or kept for the current time step.
- **Output Gate**: The output gate determines the next hidden state and the output of the LSTM cell.

LSTM (long short term memory) unit

- There are also variations on the LSTM, such as the peephole connection, which allows the gate values to depend on the previous memory cell value in addition to the input and previous hidden state.
- Both the GRU and LSTM have been tried on many different problems, and the choice between them depends on the specific problem and the trade-off between simplicity and power.
- The GRU is simpler and faster, making it easier to build bigger models, while the LSTM is more powerful and flexible. However, the LSTM has been the historically more proven choice and is often used as the default first thing to try.

GRU and LSTM

FPT Education
FPT UNIVERSITY

GRU

$$\tilde{c}^{<t>} = \tanh(W_c[\Gamma_r * c^{<t-1>}, x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_u[\, c^{<t-1>}, x^{<t>}] + b_u)$$

$$\Gamma_r = \sigma(W_r[\, c^{<t-1>}, x^{<t>}] + b_r)$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) * c^{<t-1>}$$

$$a^{<t>} = c^{<t>}$$

LSTM

$$\tilde{c}^{<t>} = \tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_u[\, a^{<t-1>}, x^{<t>}] + b_u)$$

$$\Gamma_f = \sigma(W_f[\, a^{<t-1>}, x^{<t>}] + b_f)$$

$$\Gamma_o = \sigma(W_o[\, a^{<t-1>}, x^{<t>}] + b_o)$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t-1>}$$

$$a^{<t>} = \Gamma_o * \tanh c^{<t>}$$

LSTM in pictures

$$\tilde{c}^{<t>} = \tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c)$$

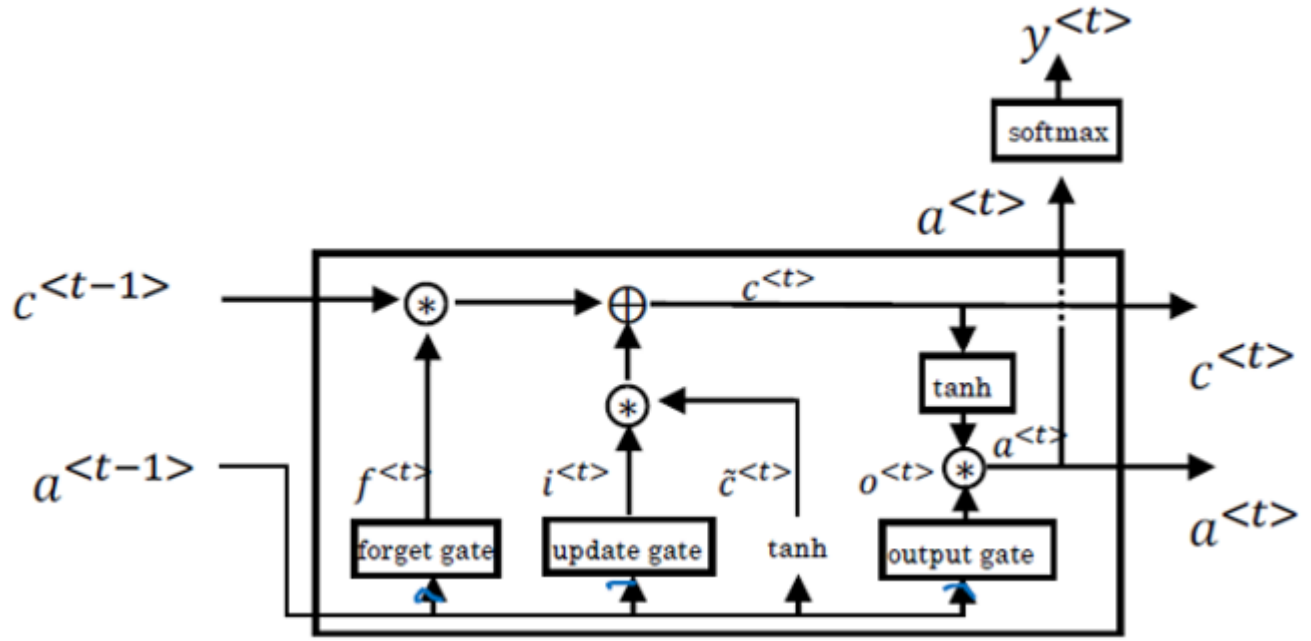
$$\Gamma_u = \sigma(W_u[a^{<t-1>}, x^{<t>}] + b_u)$$

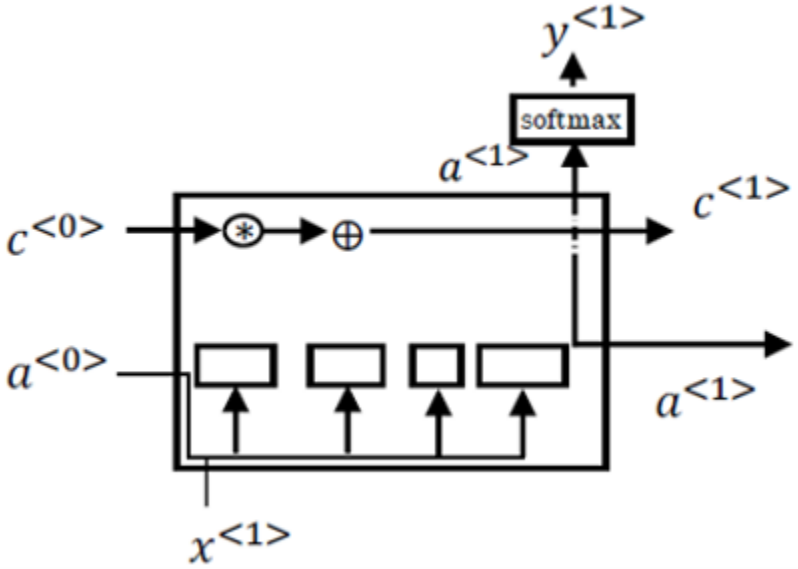
$$\Gamma_f = \sigma(W_f[a^{<t-1>}, x^{<t>}] + b_f)$$

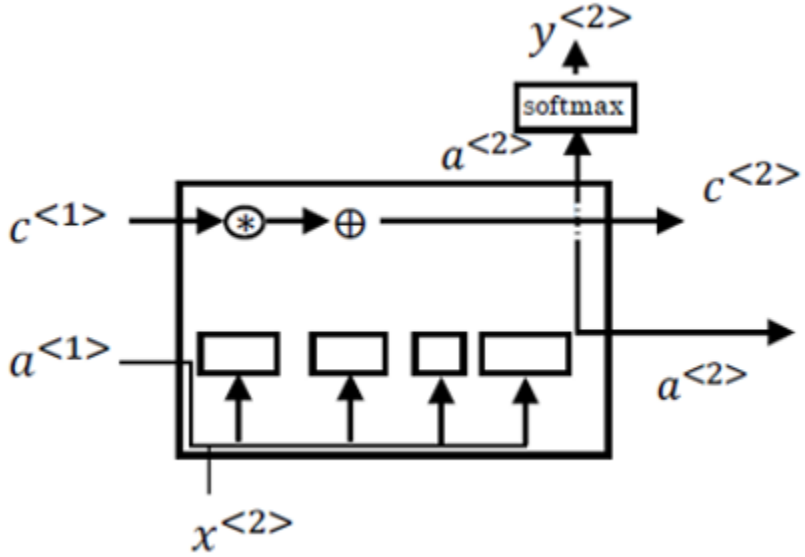
$$\Gamma_o = \sigma(W_o[a^{<t-1>}, x^{<t>}] + b_o)$$

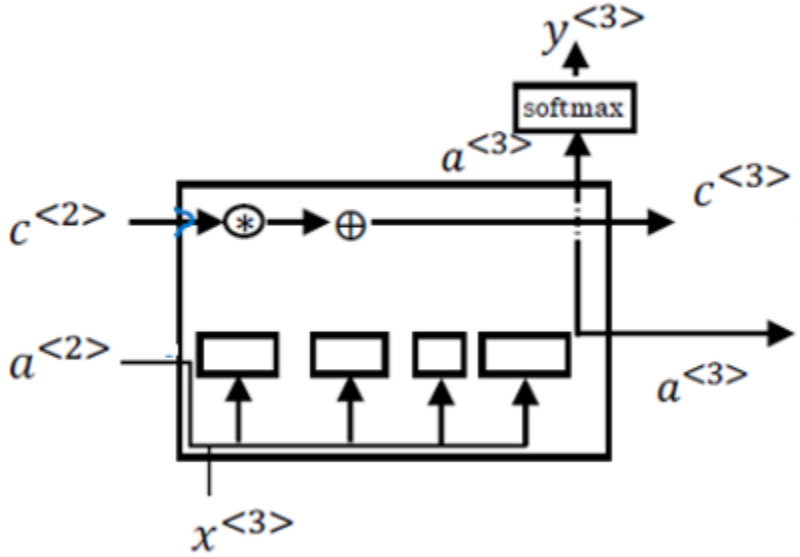
$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t-1>}$$

$$a^{<t>} = \Gamma_o * \tanh c^{<t>}$$









FPT Education

FPT UNIVERSITY

Recurrent Neural Networks

Bidirectional RNN

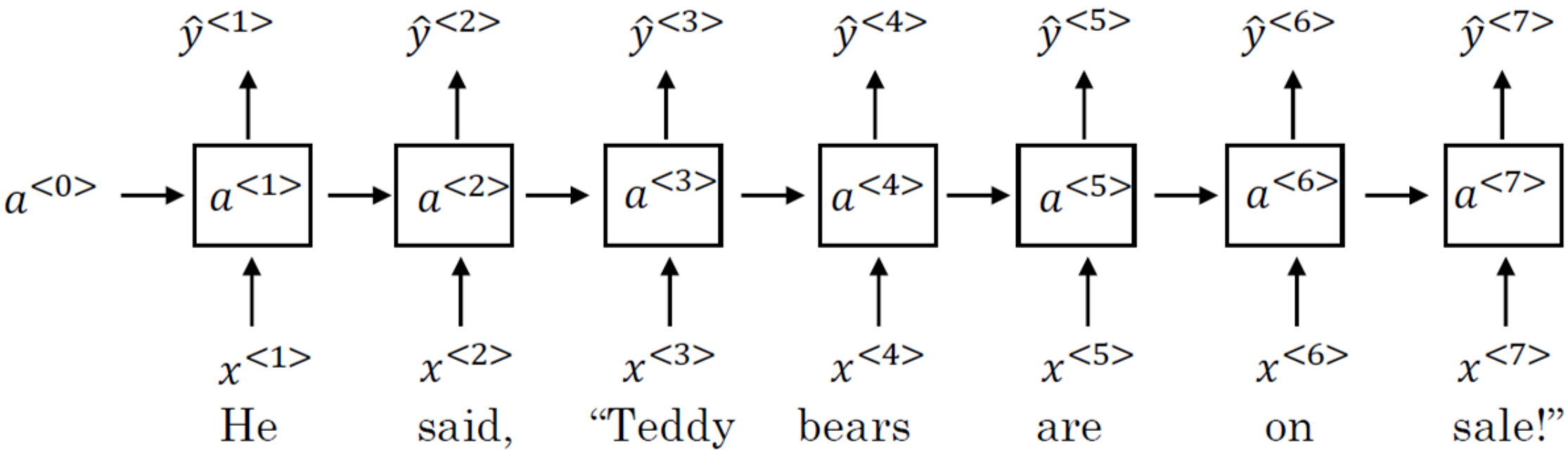
Bidirectional RNN

- A bidirectional RNN (BRNN) enhances the basic RNN, GRU, or LSTM by considering information from both the past and future at any given time, beneficial for tasks like named entity recognition or speech recognition.
- Constructed with two separate RNNs processing the sequence in opposite directions, their outputs are concatenated for predictions.
- While limited by requiring the entire sequence for predictions, it excels in tasks with full sentence data.
- Deep bidirectional RNNs (DBRNNs), featuring multiple stacked BRNN layers, further improve by capturing intricate patterns and dependencies for enhanced performance in various NLP tasks.

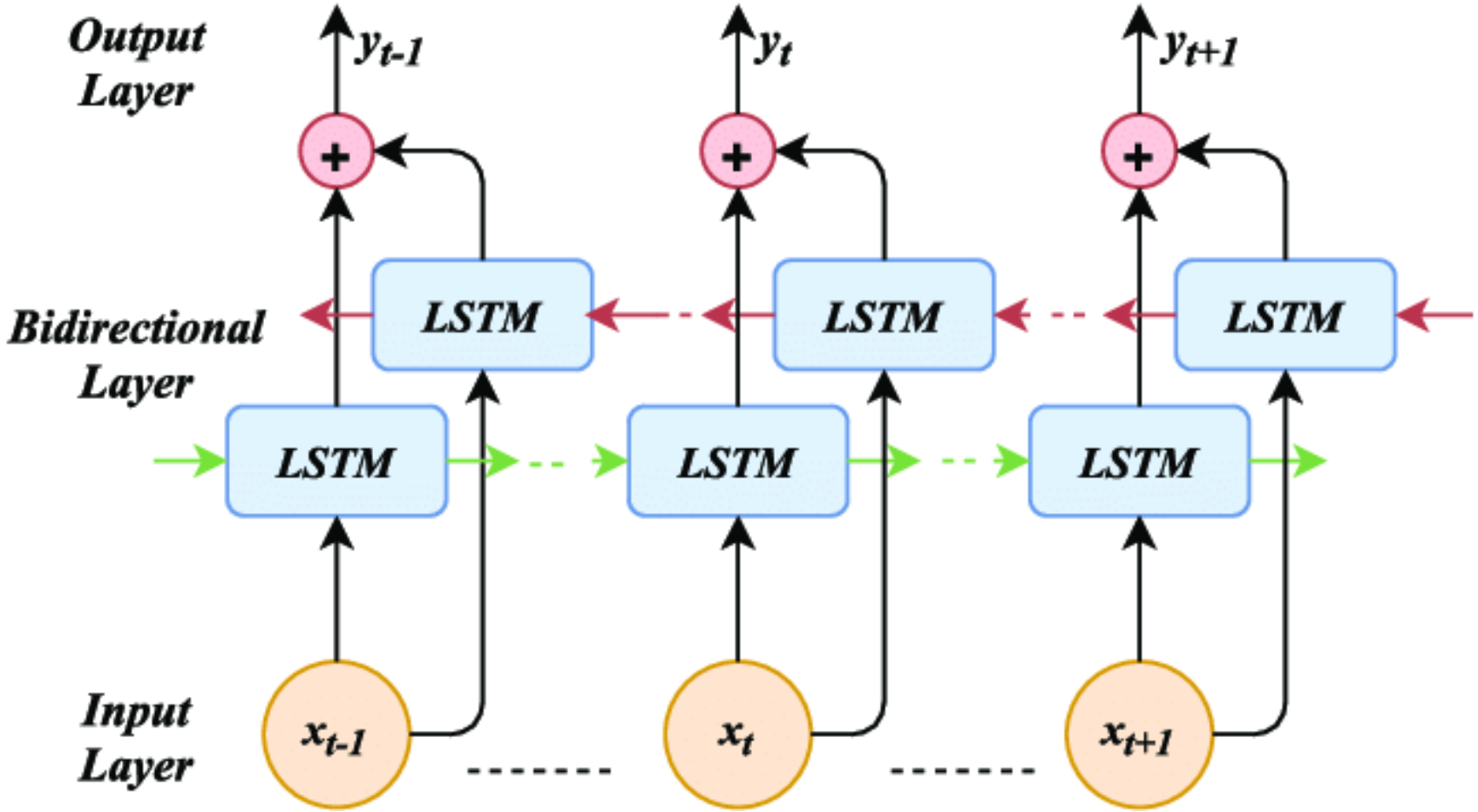
Getting information form future

He said, “Teddy bears are on sale!”

He said, “Teddy Roosevelt was a great President!”



Bidirectional RNN (BRNN)



FPT Education

FPT UNIVERSITY

Recurrent Neural Networks

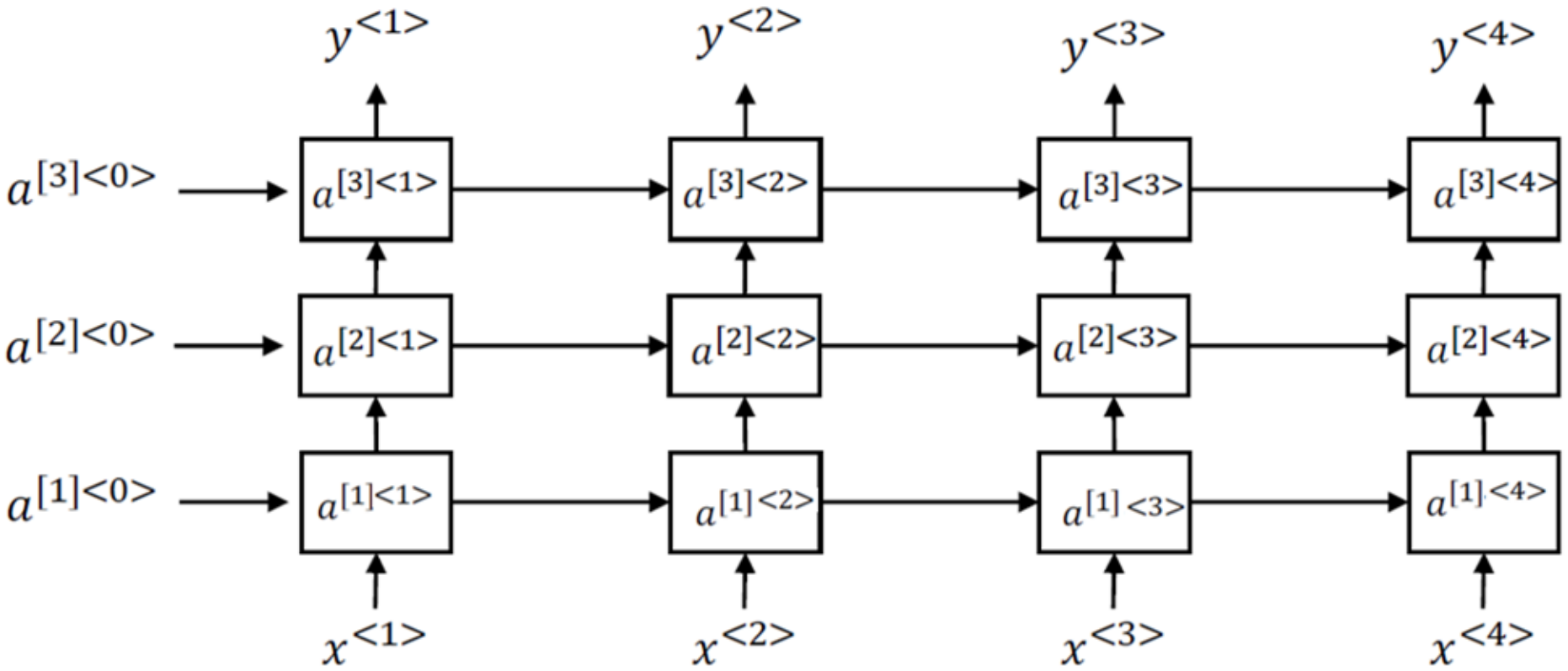
Deep RNNs

Deep RNNs

FPT Education
FPT UNIVERSITY

- In a deep RNN, each layer has its own activation value over time, computed using inputs from the previous layer and the previous time step.
- Despite the potential for large networks with even a few layers, having three layers is substantial for RNNs. Deep RNNs can utilize GRU and LSTM blocks, and deep bidirectional RNNs are feasible.
- Due to computational costs, deep RNNs with extensive temporal dimensions are less common. Typically, deep recurrent layers are stacked, followed by a deep network for output prediction.
- In summary, leveraging various RNN types allows the construction of potent sequence models.

Deep RNN exmample



Summarization

- Sequence models, crucial for tasks like NLP and time series analysis, process sequential data.
- Basic Recurrent Neural Networks (RNNs) maintain hidden states to capture information across time steps.
- Backpropagation Through Time trains RNNs by unfolding them through time.
- RNN variations include one-to-one, one-to-many, many-to-one, and many-to-many architectures for diverse tasks.
- Language models predict the next word, enabling text generation.
- GRU mitigates vanishing gradients in RNNs using gating mechanisms.
- LSTM incorporates memory cells to capture long-term dependencies.
- Bidirectional RNNs process sequences in both directions.
- Deep RNNs stack multiple layers for intricate pattern learning.

Questions

- How do you refer to the j-th word in the i-th training example?
- When is an RNN architecture appropriate with Tx = Ty?
- Which tasks use a many-to-one RNN architecture?
- What is the RNN estimating at time step t?
- How do you sample words from a trained language model RNN?
- What is the most likely cause of NaN values during RNN training?
- In an LSTM with 100-dimensional activations and 10,000-word vocab, what is the dimension of Γu?
- Which GRU simplification is less likely to suffer from vanishing gradients: removing Γu or Γr?
- In LSTM vs GRU, what are the equivalents of Update Gate and Forget Gate?
- For predicting a dog's mood from past weather, should you use a unidirectional or bidirectional RNN?