MODULE 7: INTRODUCTION TO RNN

BA713 - Machine Learning & AI

CONTENTS

SEQUENCE MODELING

APPLICATIONS OF SEQUENCE MODELING

INTRODUCTION TO RNN & ITS VARIANTS

SEQUENCE MODELING

- task of predicting what comes next in the sequence

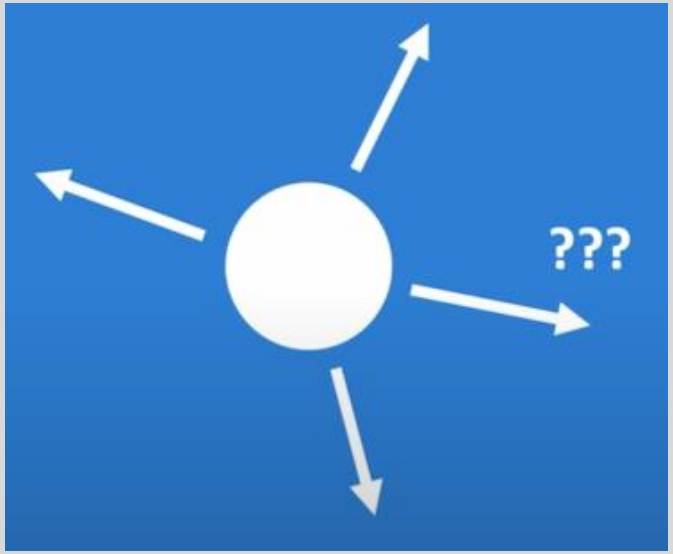
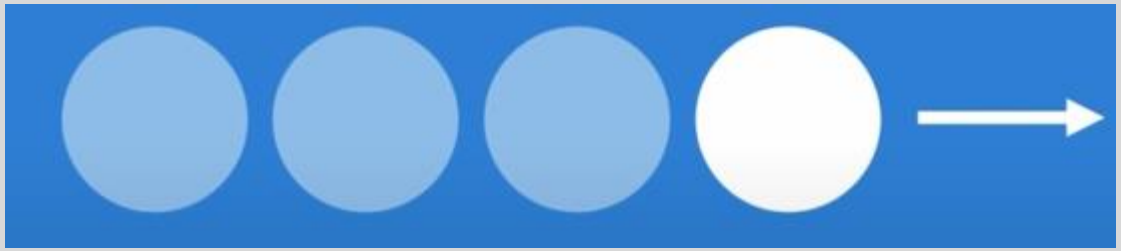


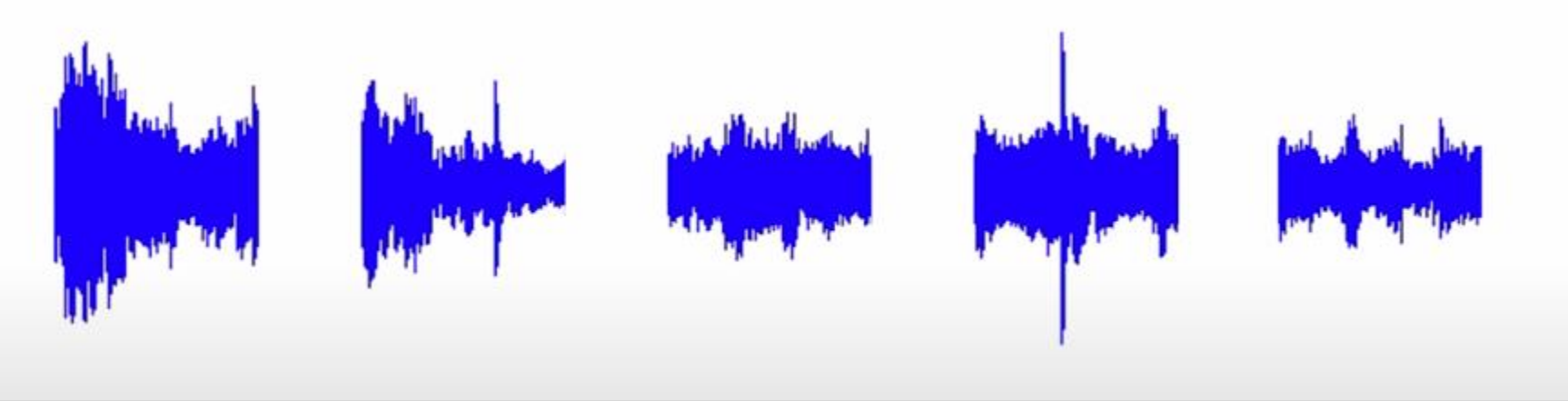
Image of a ball, predict the direction it will go next



With past information about the sequence of events, you can predict the future

Image Source: introtodeeplearning.com

EXAMPLE OF SEQUENTIAL DATA-AUDIO



Audio/speech is an example of sequential data. In this image, it is divided into multiple parts or sequences. We can use AI to learn from the past sequence and predict the next sequence

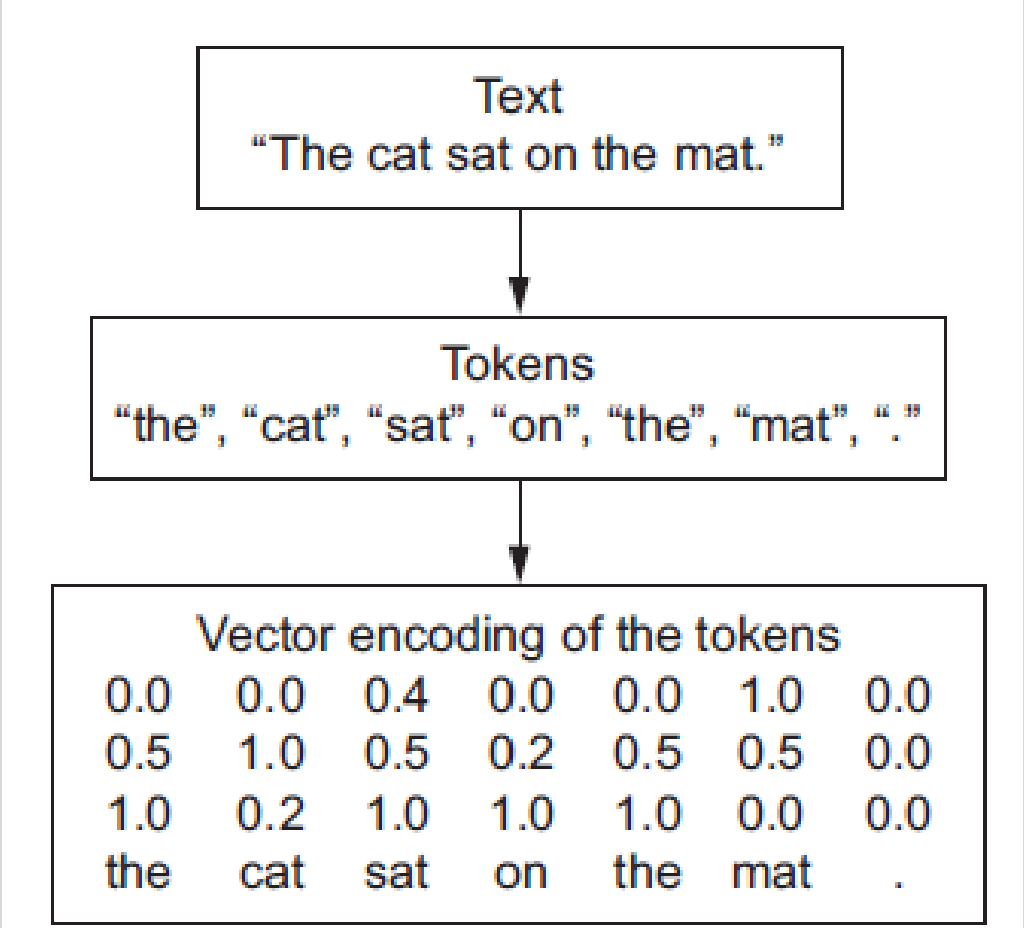
Image Source: introtodeeplearning.com

EXAMPLE OF TEXT SEQUENCES

- Convert text to numeric tensors or tokens
- Tokenization process
- One-hot encoding

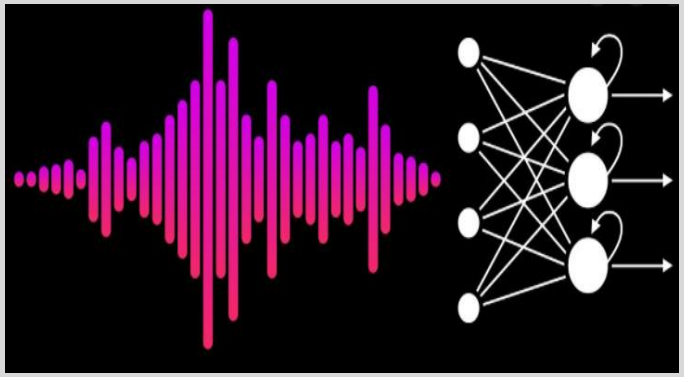
- Multiple ways→

✓ Segment text into words, and transform each word into a vector
✓ Segment text into characters and transform each character into a vector.
✓ Extract overlapping groups of multiple consecutive words or characters



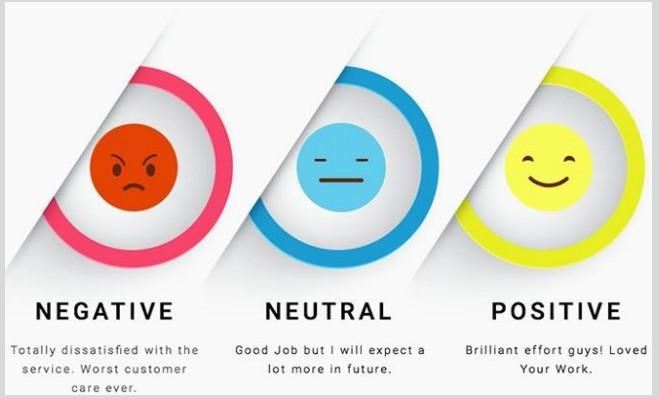
Source : Deep Learning with Python by François Chollet

SEQUENCE MODELING APPLICATIONS



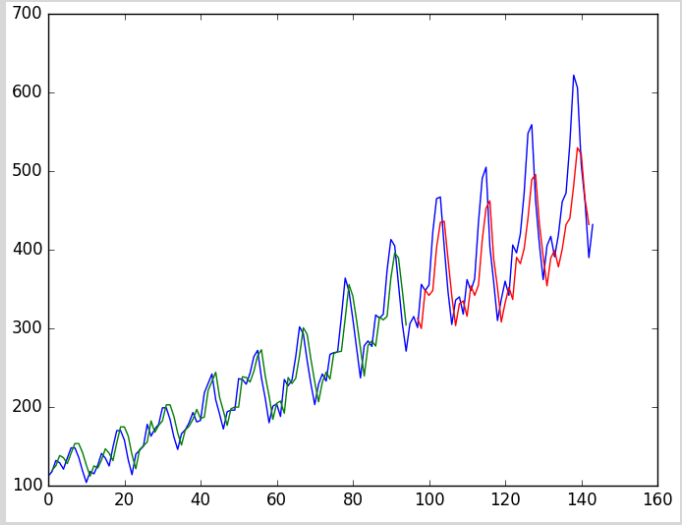
Speech Recognition

Natural Language Processing



Sentiment Classification

Image Captioning



Sequence Prediction (Weather Forecasting, Stock Prices, etc.)

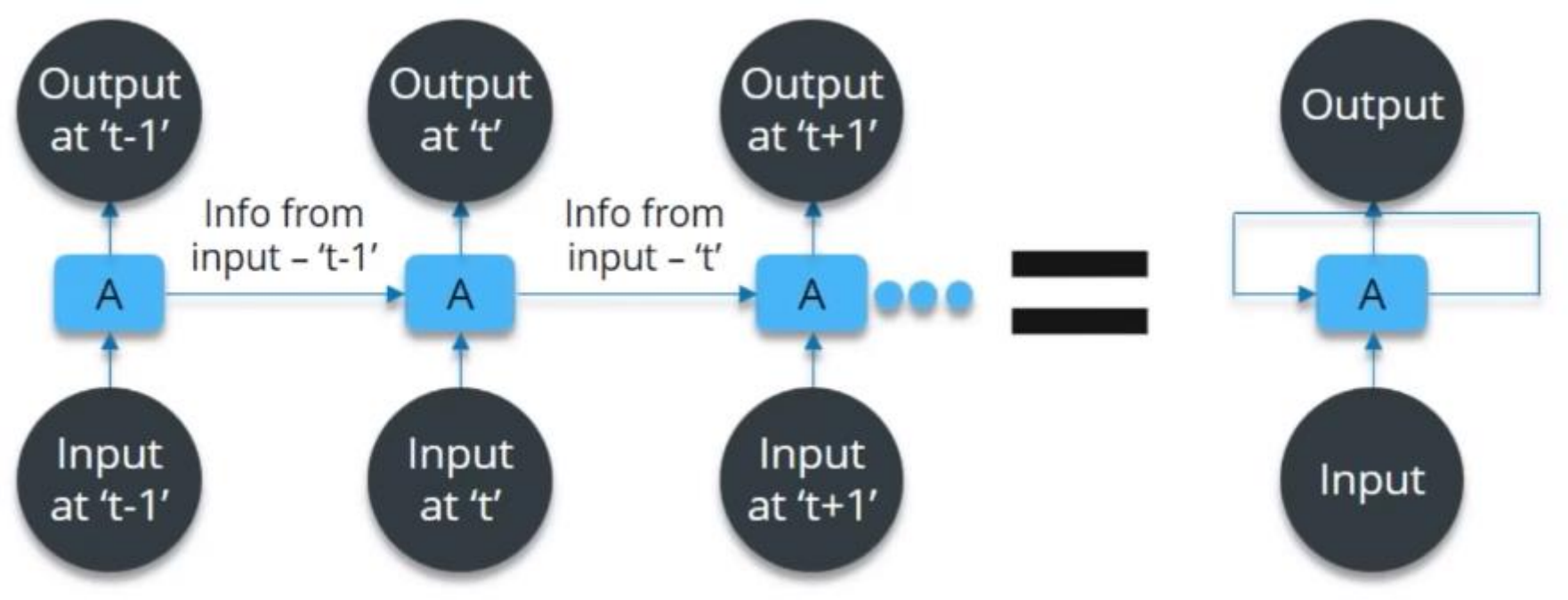
Product Recommendation

RECURRENT NEURAL NETWORKS (RNNs)

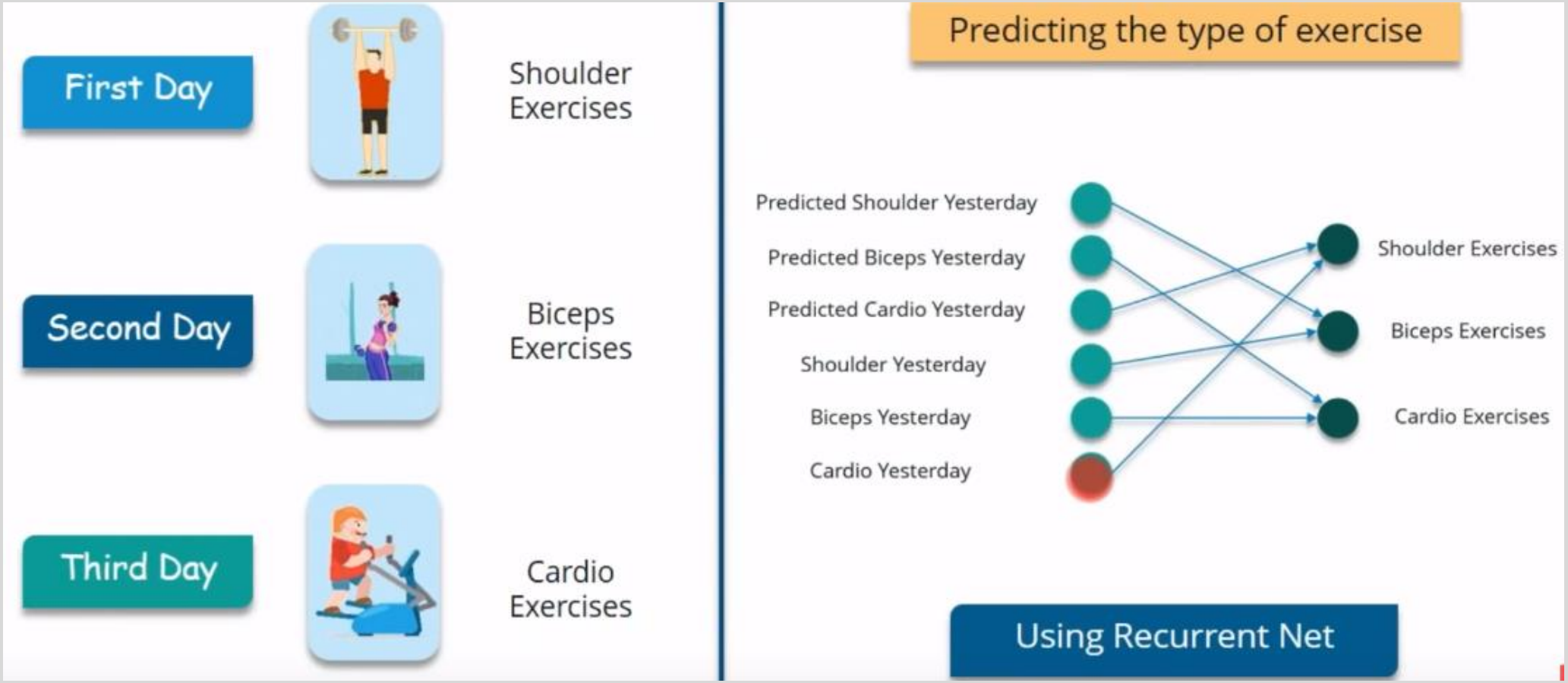
Definition: Advanced type of artificial neural network with directed cycle between nodes

- commonly used in sequence prediction
- Based on **David Rumelhart's** work in 1986
- designed to recognize sequential characteristics and use patterns to predict the next scenario
- Internal state-memory-sequence of events
- Different from other neural networks-use feedback loops
- **Backpropagation Through Time (BPTT)**→ loop information back into the network
- Connect inputs together→ sequential or temporal data

RNN

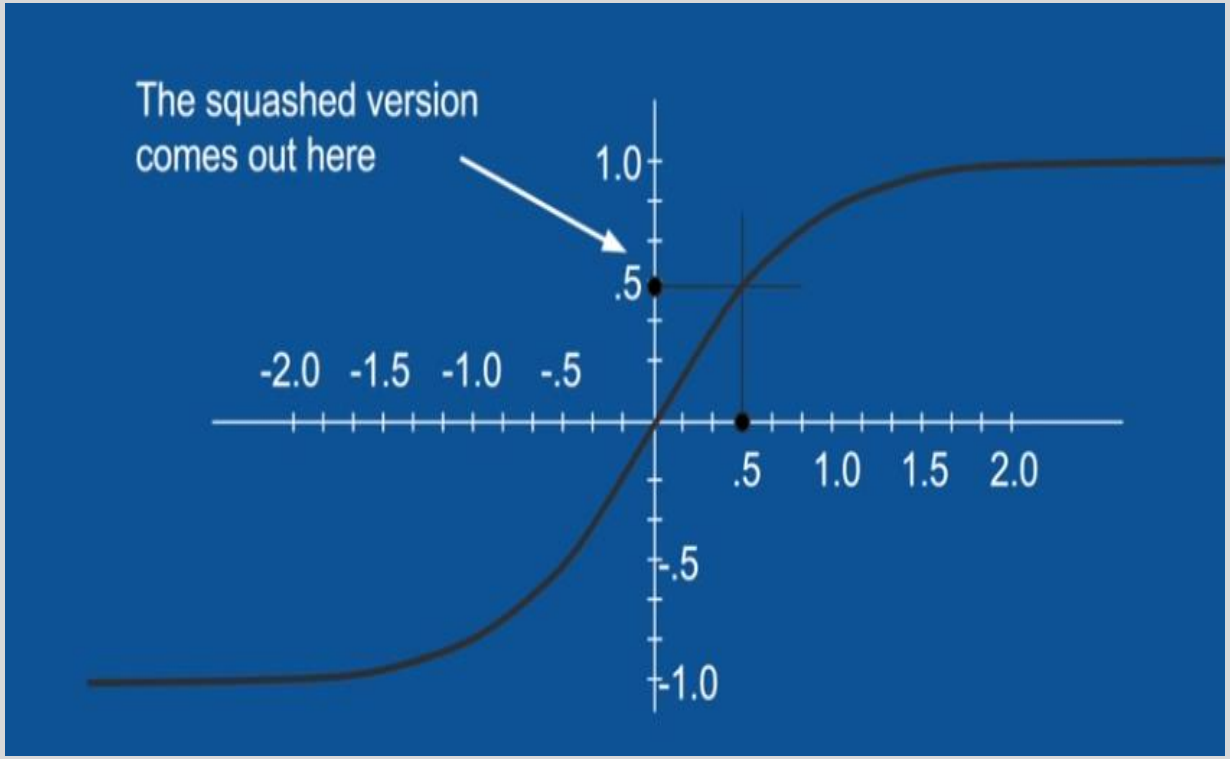

Output at 't-1'
Output at 't'
Output at 't+1'
Info from input – 't-1'
Info from input – 't'
A
A
A
Input at 't-1'
Input at 't'
Input at 't+1'
Output
A
Input

RNN EXAMPLE

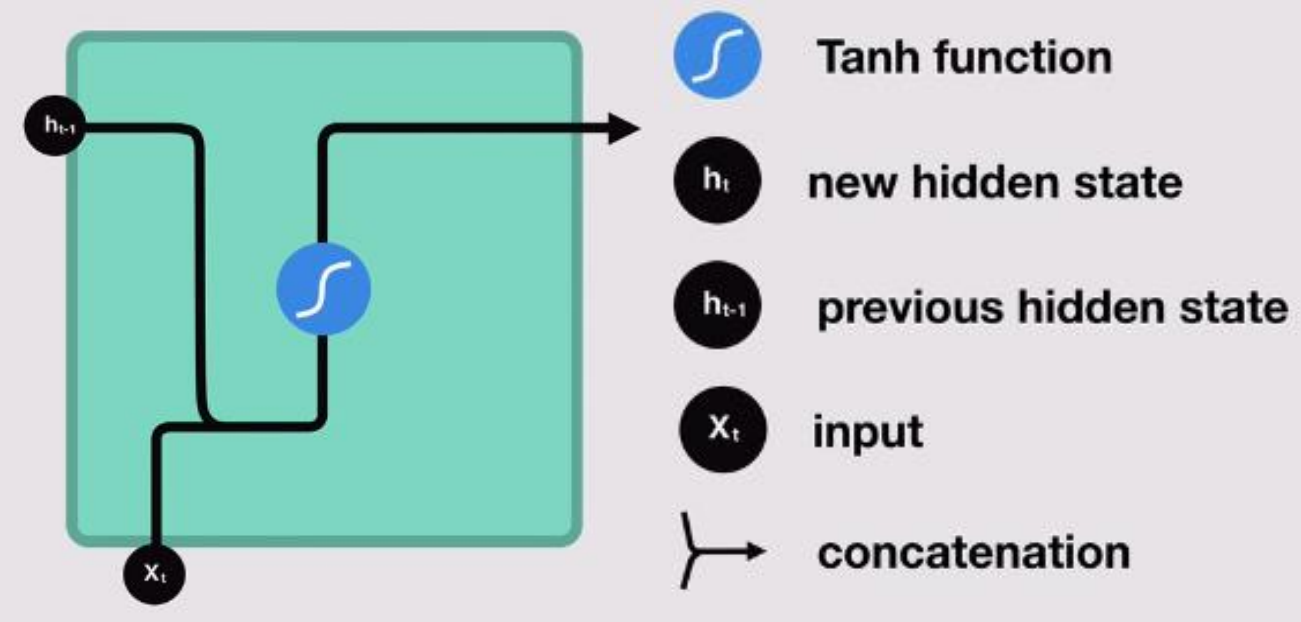

First Day
Shoulder Exercises
Second Day
Biceps Exercises
Third Day
Cardio Exercises
Predicting the type of exercise
Predicted Shoulder Yesterday
Predicted Biceps Yesterday
Predicted Cardio Yesterday
Shoulder Yesterday
Biceps Yesterday
Cardio Yesterday
Shoulder Exercises
Biceps Exercises
Cardio Exercises
Using Recurrent Net

SQUASHING FUNCTION (tanh) FOR RNN

- Activation/Transfer function
- Classification of output Y/N
- Non-linear activation function
- Range -1 to 1
- Not letting input vectors explode
- Regulates NN



RNN CONTROL FLOW



1) Input +prev hidden state= vector current input and previous input

2) Vector → tanh → output is new hidden state

3) tanh → regulates values through network between -1 to 1

Image source: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

TRAINING RECURRENT NEURAL NETWORKS

- Done through Backpropagation applied to every time stamp

- **Back Propagation Through Time (BPTT)**

✓ the errors are calculated and accumulated for each timestep
✓ network is then rolled back to update the weights
✓ when the number of time steps increases the computation also increases

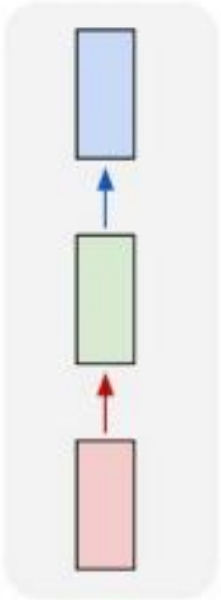
- **Truncated Backpropagation (TBPTT):**

✓ the sequence is processed one timestep at a time
✓ update is performed for a fixed number of time steps.

Source: https://machinelearningmastery.com/gentle-introduction-backpropagation-time/

RNN PARADIGMS

ONE TO ONE



Single input to single output

ONE TO MANY

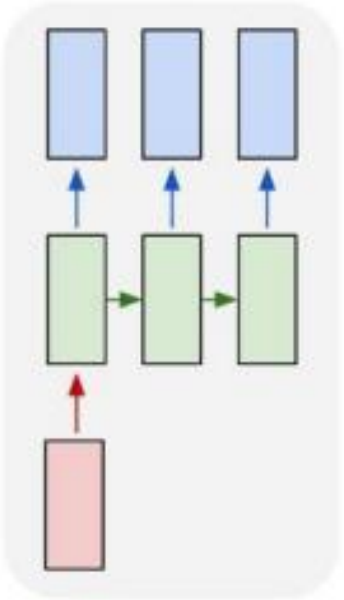
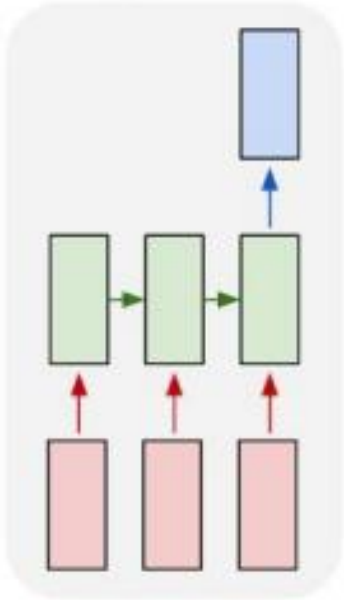


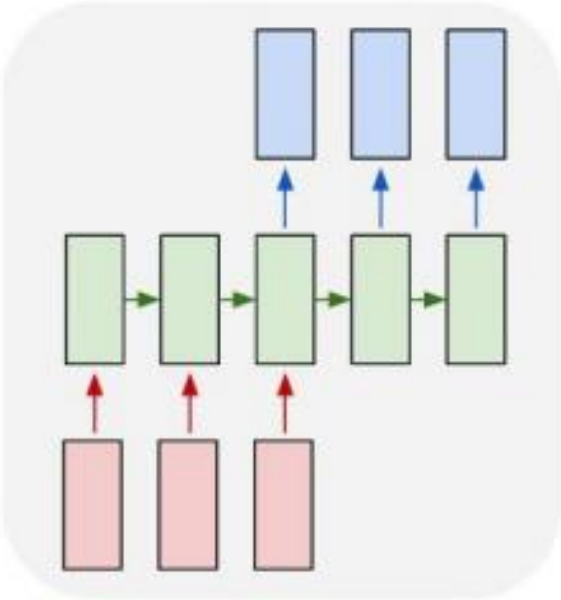
Image captioning

MANY TO ONE



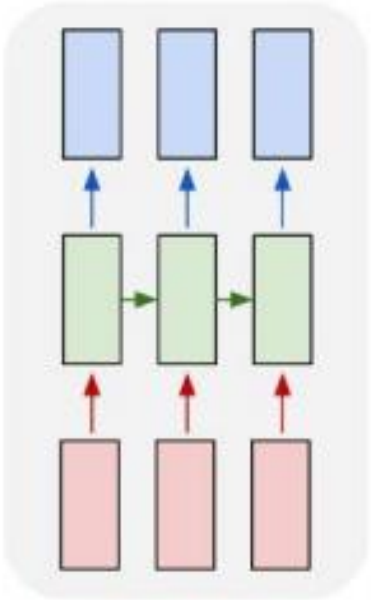
Sentiment Analysis

MANY TO MANY



Machine Translation

MANY TO MANY

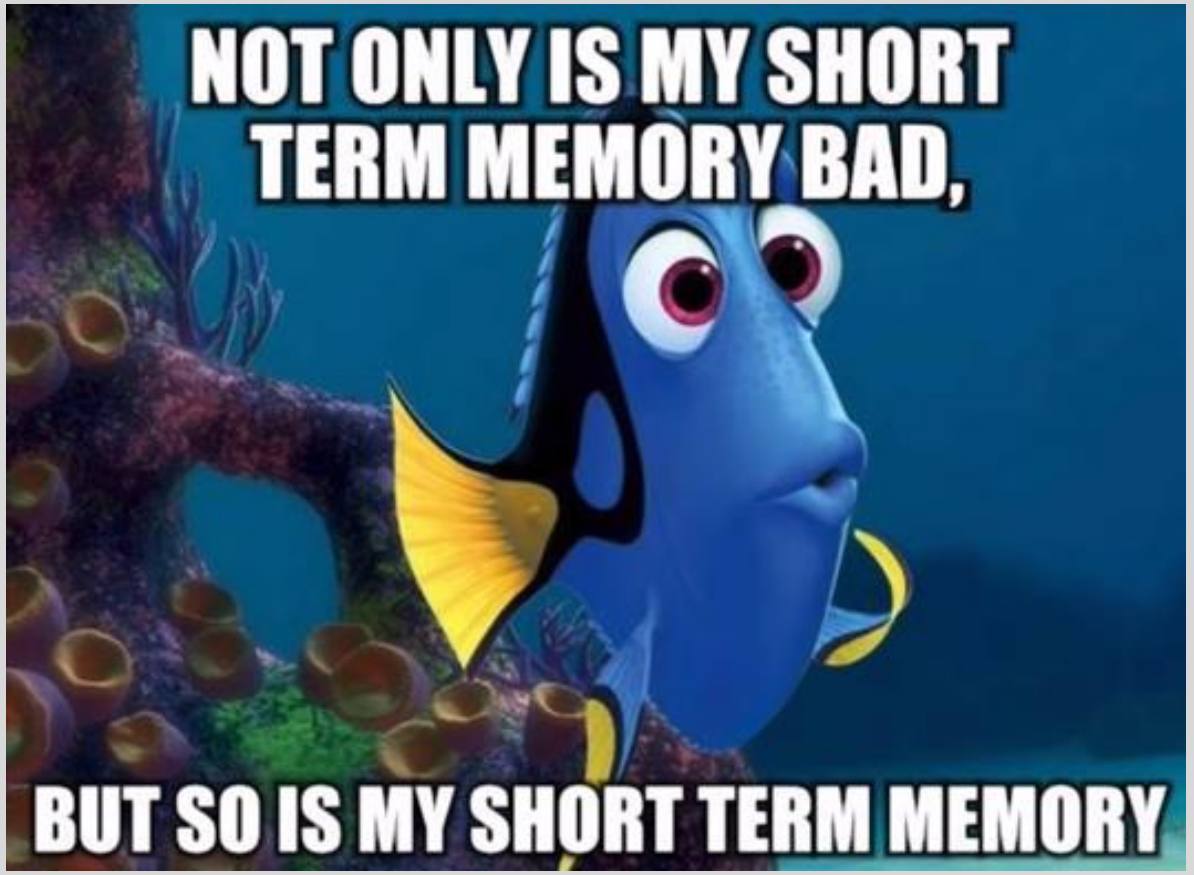


Speech Tagging

Image source: http://www.cs.cmu.edu/

PROBLEMS IN RNN

- Overfitting
- Cannot handle long term dependencies
- Vanishing gradient
- Exploding gradient



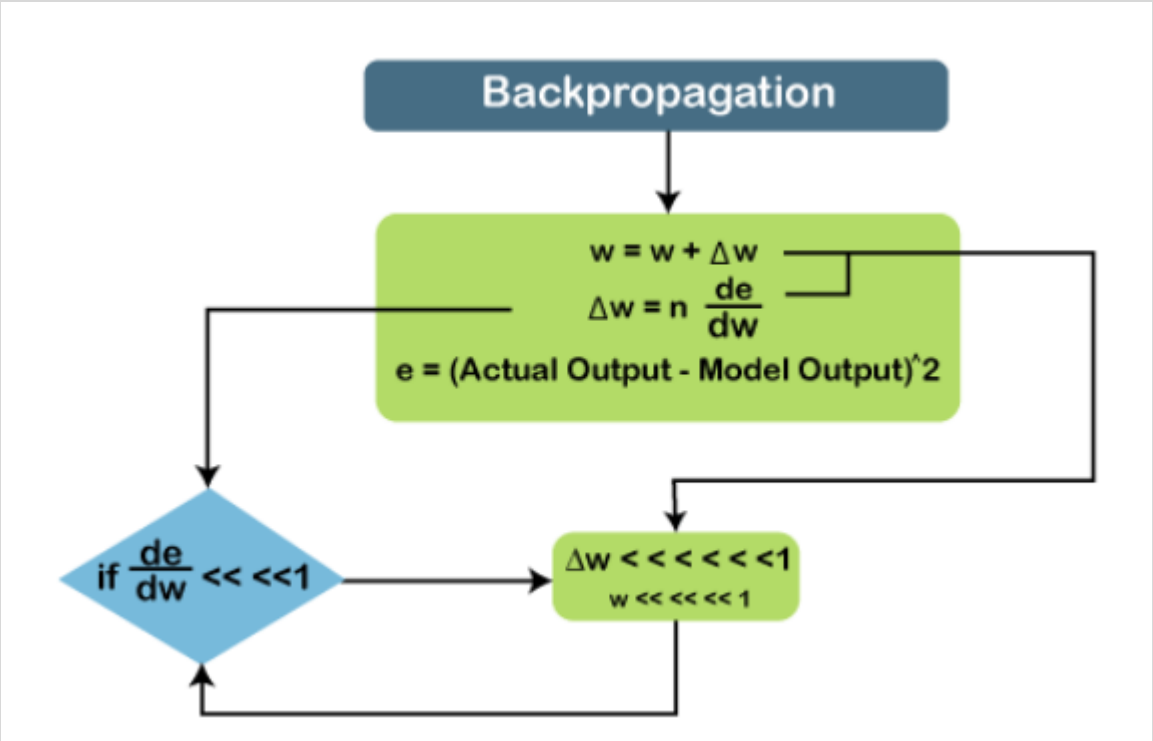
RNN cannot handle long term dependencies

Scenario 1: Suppose RNN is used to find next word in the text, say, **The moon is in the….. (sky),** easy to predict next word from previous word, recent information to perform a task, gap between prediction and recent information is less, RNN can use past info without vanishing gradient problem

Scenario 2: I went to Germany……..I learnt to speak……,*long term dependency*, previous context, recent information suggests next word is a language, but unless we know the previous context, wont be able to get it. It needs many iterations and difficult to find next word. As gap grows, RNNs are unable to learn to connect information.

VANISHING GRADIENT

- W=w+Δ**w**= new adjusted weight (minimize errors)
- If **de/dw** or gradient→ very small, multiply with learning rate→value small
- gradient measures how much the output of a function changes, if you change the inputs a little bit
- **Change in weight is negligible**
- No learning
- Gradient vanishes



e=error

de/dw= change in error w.r.t weight

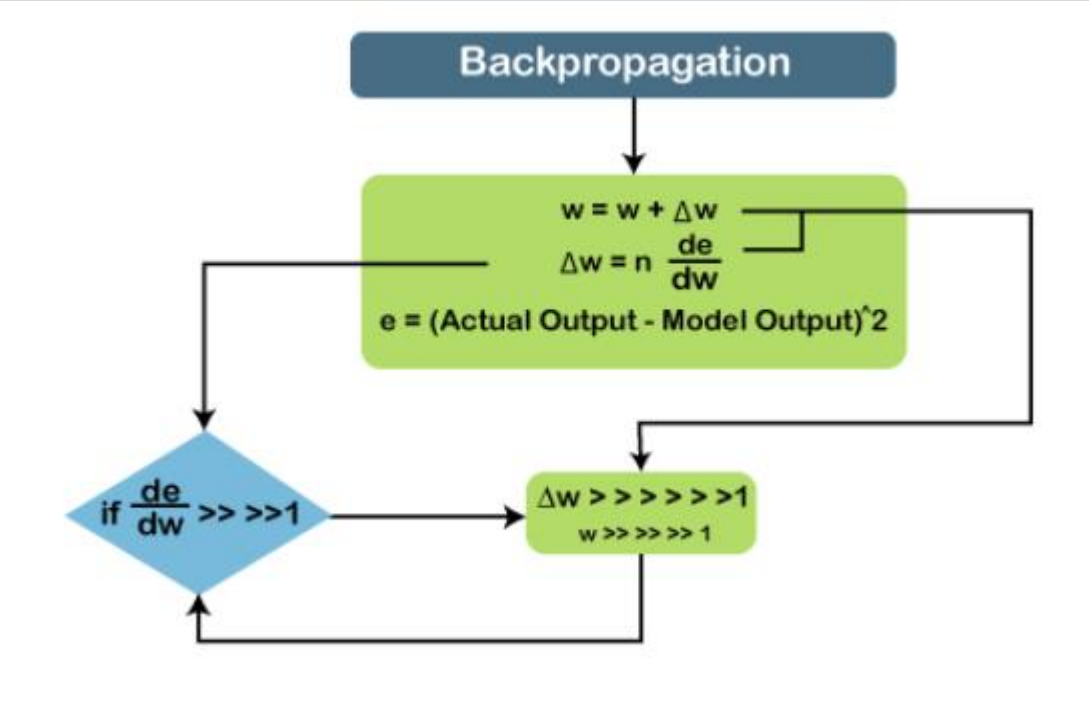
n= learning rate

Δw = change in weight

Image source: https://medium.com/@sjakki/vanishing-and-exploding-gradients-in-neural-networks-5b3ee108a568

EXPLODING GRADIENT

- W=w+Δ**w**= new adjusted weight (minimize errors)
- If **de/dw** or gradient→ very large, multiply with learning rate→ value large
- **Change in weight is large (explodes)**
- No learning, model unstable
- Large changes in loss→ NaN values



e=error

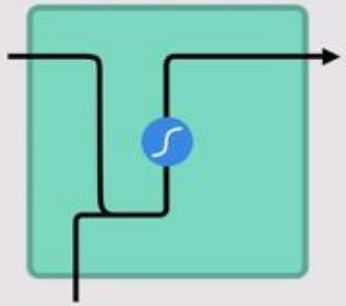
de/dw= change in error w.r.t weight

n= learning rate

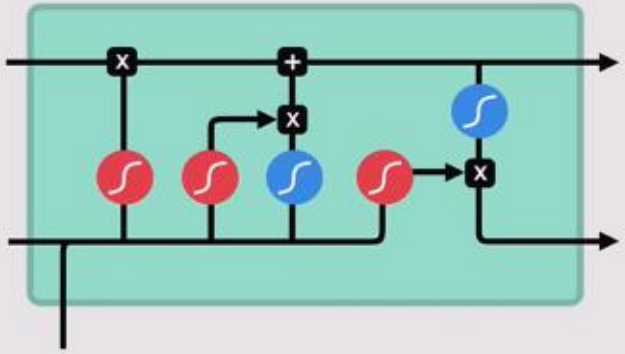
Δw = change in weight

Image source: https://medium.com/@sjakki/vanishing-and-exploding-gradients-in-neural-networks-5b3ee108a568

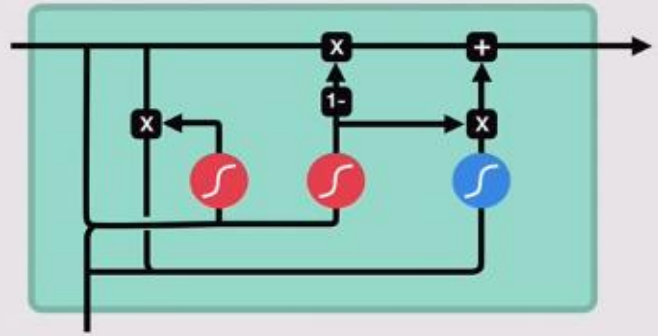
RNN AND ITS VARIANTS



Vanilla Recurrent Neural Networks



Long Short Term Memory (LSTM)



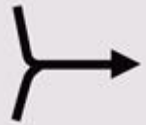
Gated Recurrent Units (GRU)

sigmoid

tanh

pointwise multiplication

pointwise addition



vector concatenation

Image source: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

LONG SHORT TERM MEMORY (LSTM) UNIT

LONG SHORT TERM MEMORY NETWORK (LSTM)

Definition: is a special kind of RNN capable of learning long term dependencies

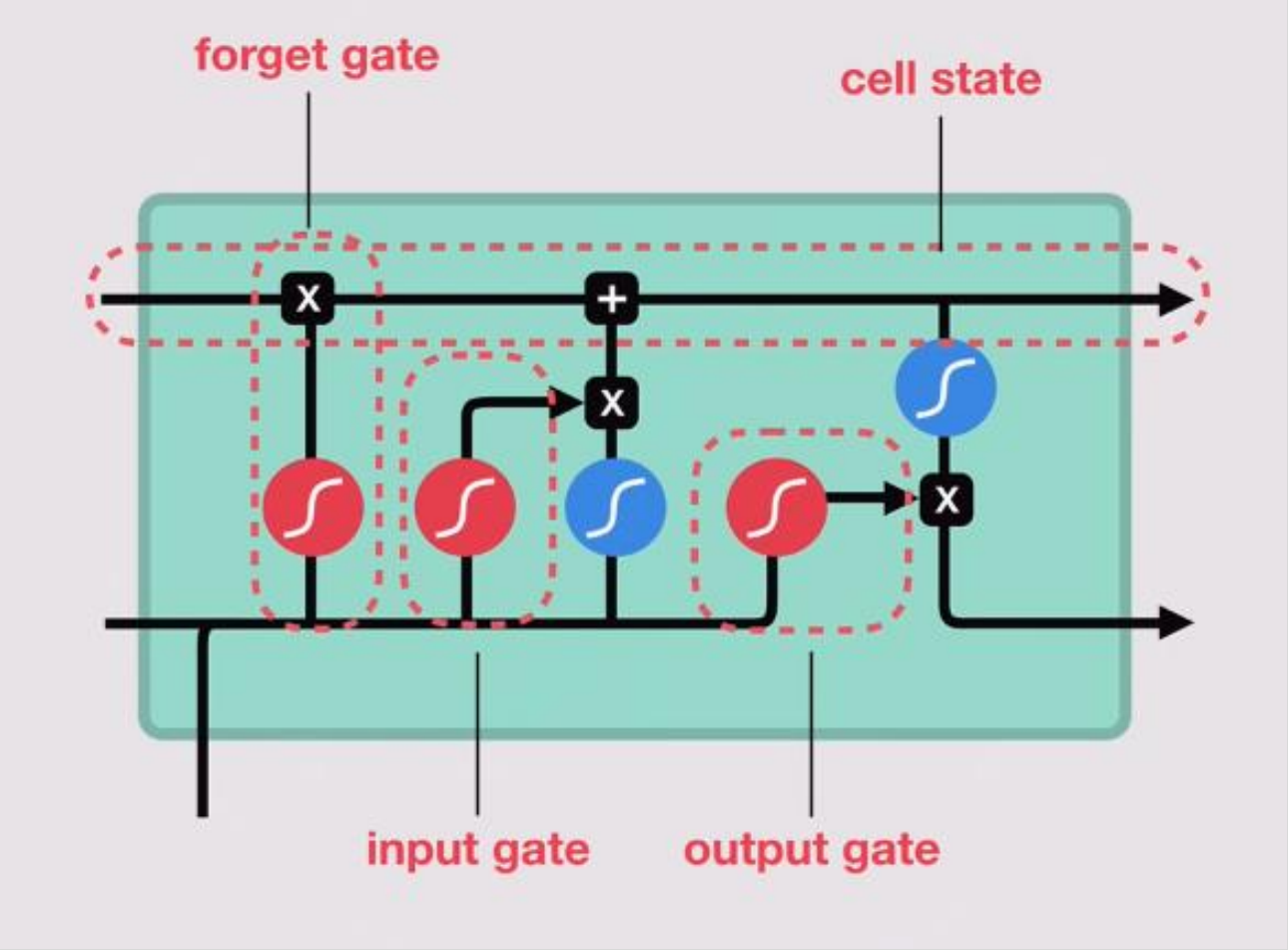
- LSTM block is part of a recurrent neural network structure
- RNN-multiple LSTM units forms LSTM network or LSTM
- **Hochreiter** and **Schmidhuber** in 1997
- well-suited to classify, process and predict time series given time lags of unknown duration
- LSTM selectively remembers data for a longer time, uses past information to make predictions for future
- **Applications:** Stock market prediction, text generation and sequence prediction, weather forecasting, time series analysis, natural language processing, handwriting recognition, voice recognition and so on.

LSTM COMPONENTS

LSTM has 4 components :-

1. **Forget gate-**decides which information should be discarded (between 1 and 0)
2. **Cell state-**memory of network, carry information throughout sequence processing
3. **Input gate-**updates cell state with input
4. **Output gate-**used to compute output activation, decides what next hidden state should be

LONG SHORT TERM MEMORY NETWORK



C_{t-1} previous cell state

f_t forget gate output

i_t input gate output

$\tilde{C}_t$ candidate

C_t new cell state

o_t output gate output

h_t hidden state

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Image source: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

LSTM CONTROL FLOW

1) **Forget gate-**information from previous hidden state and current input → sigmoid → values between 0 (forget) and 1 (keep)

2) **Input gate-**

- prev hidden state+current input→sigmoid function → which values updated-between 0 (not important) and 1 (important)
- Pass hidden state and current input into tanh function → squash values between -1 and 1 → regulate network
- Multiply tanh output with sigmoid output. Sigmoid decides which information to keep from tanh output

3) **Cell state-**

- Cell state multiplied by forget vector → drop values if multiplied by 0
- Output from input gate+cell state=new cell state

4) **Output gate-**

- Prev hidden state+current input → sigmoid function
- New cell state → tanh function
- Multiply tanh output with sigmoid output→hidden state
- New cell state and new hidden state carried→ next time step

C_{t-1} previous cell state

f_t forget gate output

i_t input gate output

$\tilde{C}_t$ candidate

C_t new cell state

o_t output gate output

h_t hidden state

LSTM EXAMPLE

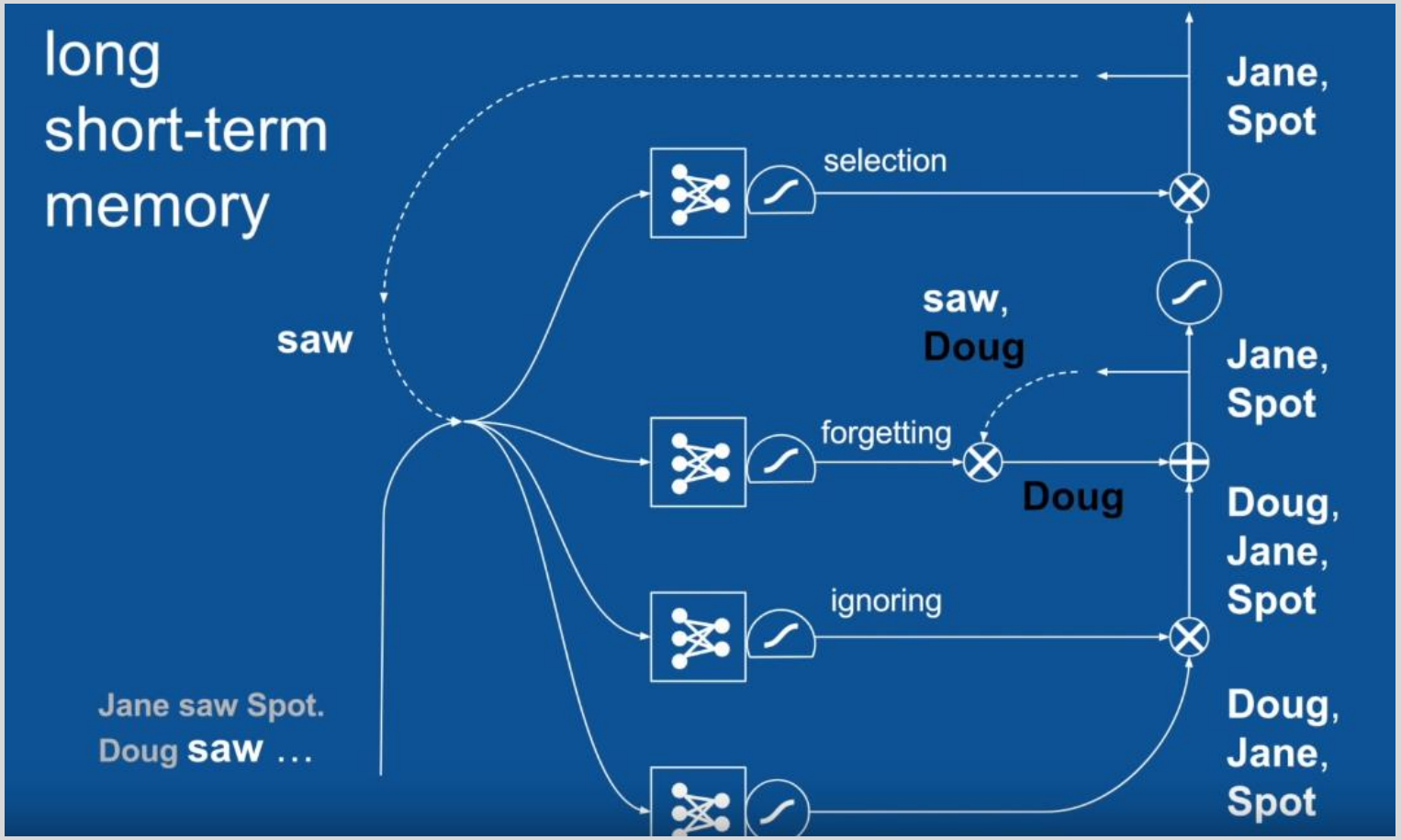


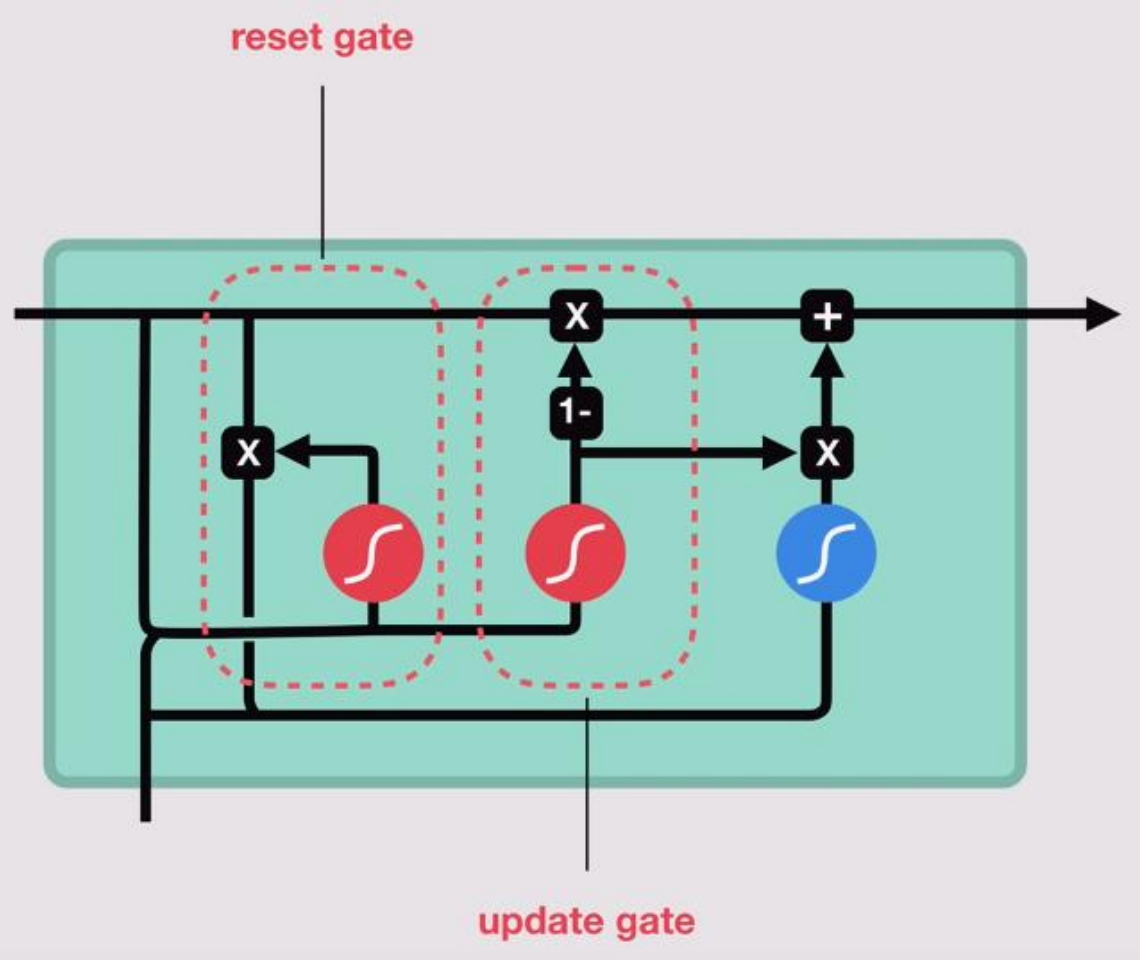
Image source: https://towardsdatascience.com/machine-learning-recurrent-neural-networks-and-long-short-term-memory-lstm-python-keras-example-86001ceaaebc

GATED RECURRENT UNIT (GRU)

GATED RECURRENT UNIT (GRU)

- **Definition-**Advanced RNN, Gating mechanism in RNNs
- **Kyunghyun Cho-2014**
- Variation of LSTM
- Solve vanishing gradient problem in RNN
- Computationally less expensive-less complex structure as compared to LSTM
- Performs better on smaller datasets.
- **Applications**: Speech signal modeling, music modeling

GATED RECURRENT UNIT ARCHITECTURE



- Only 2 gates-
- ✓ **Reset gate** -how much past info to forget
- ✓ **Update gate** -what info to discard or keep (forget gate+ input gate)
- cell state + hidden state=hidden state

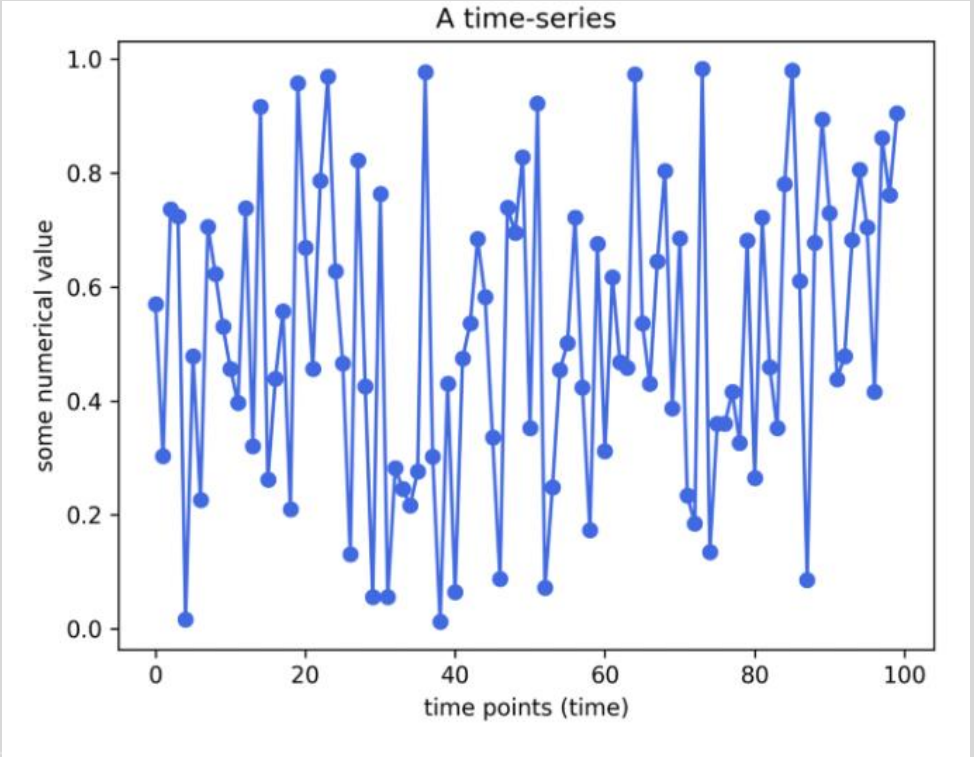
Image source: https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

LSTM VERSUS GRU

LSTM	GRU
Three gates	Two gates
Control the exposure of memory content (cell state)	Expose the entire cell state to other units in the network
Has separate input and forget gates	Performs both of these operations together via update gate
More parameters	Fewer parameters

TIME SERIES FORECASTING

- Traditional ML/DL models→ use input features and observations, no time dimension
- Time-series forecasting models→ capable of predicting future values from past or current information
- **Non-stationary data**→ statistical properties e.g., the mean and standard deviation change over time
- Sequence of observations taken sequentially in time
- **Time series data examples→ stock prices, house prices over time, temperature**



Source: https://towardsdatascience.com/lstm-time-series-forecasting-predicting-stock-prices-using-an-lstm-model-6223e9644a2f

LSTM HANDS-ON EXERCISE

(Time Series Forecasting)

REFERENCES

[1] http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[2] https://towardsdatascience.com/activation-functions-and-its-types-which-is-better-a9a5310cc8f

[3] http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

[4] https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be

[5] https://www.kaggle.com/kredy10/simple-lstm-for-text-classification/notebook

[6] http://www.wildml.com/2015/10/recurrent-neural-networks-tutorial-part-3-backpropagation-through-time-and-vanishing-gradients/

[7] https://www.researchgate.net/figure/Scheme-of-one-neuron-layer-with-the-application-of-the-activation-function_fig5_257548537

[8] https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

THANKS!

Do you have any questions?