Applied Deep Learning

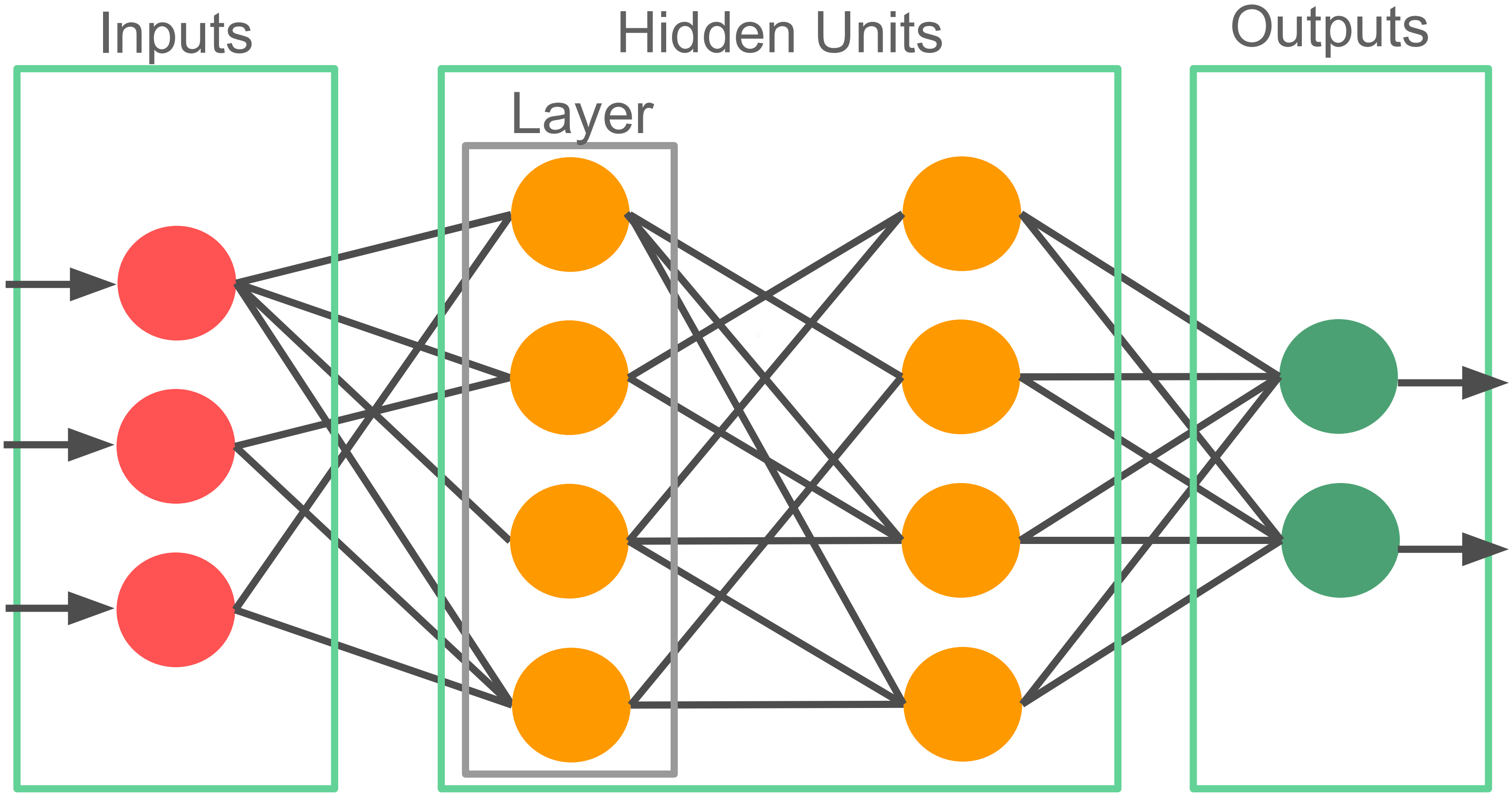
Recurrent Neural Networks

Alexander Pacha - TU Wien

Recap

- What do we mean when we say Multi Layer Perceptron (MLP)?
- What is the key insight that justifies Convolutional Layers?
- Why do we use parameter sharing and what are feature maps?

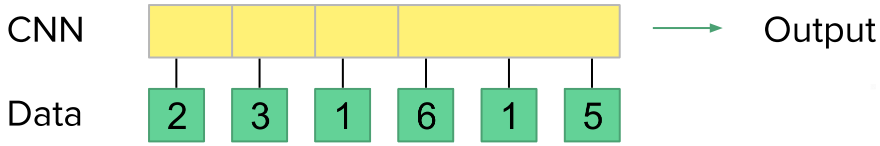
Recap - Neural Network



Modelling sequences with CNNs

CNNs expect grid-like structure and can operate on sequences

- Audio waveform, Color image data, Volumetric data, Color video data, ...



But:

- Fixed-size input
- Fixed-size output
- Fixed-size number of computations
- No understanding of previous information

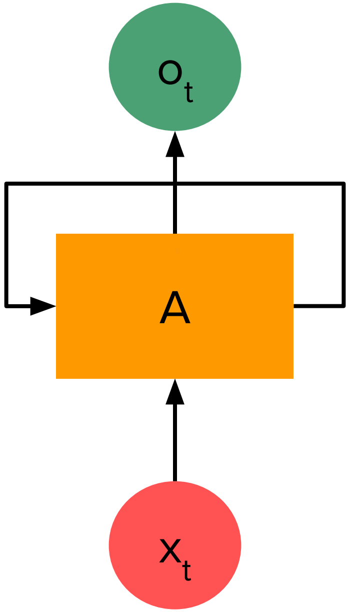
We need to persist information

Recurrent Neural Network Model

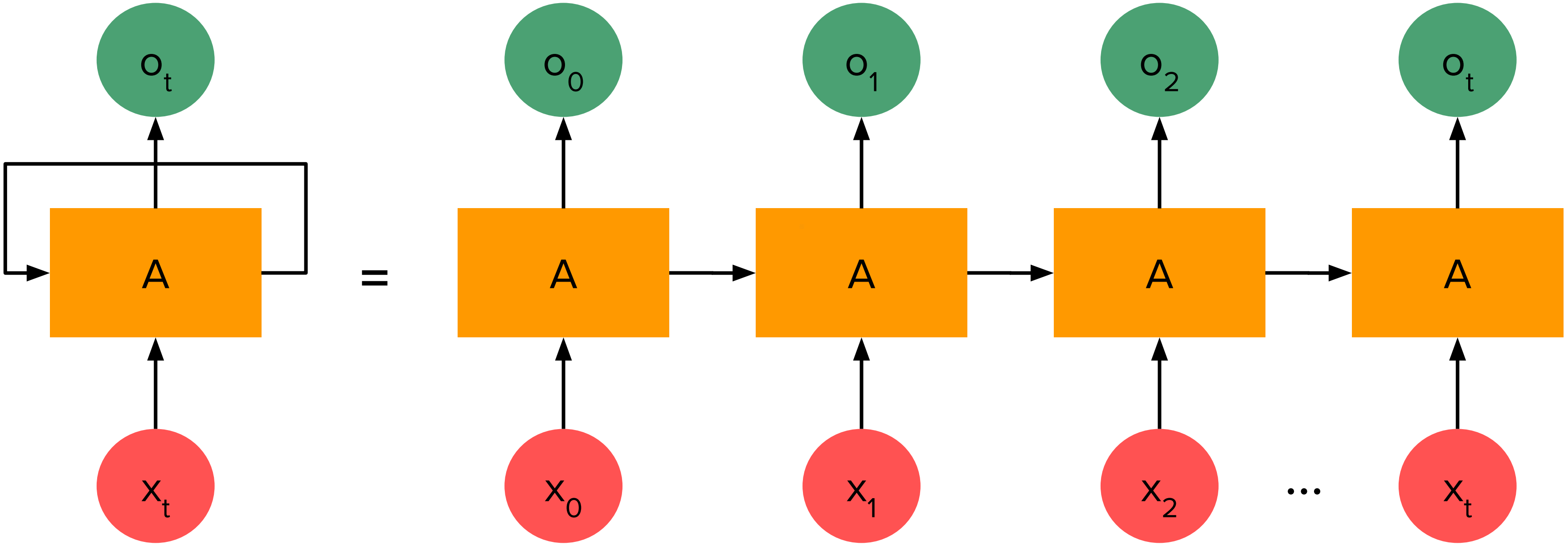
A = a chunk of neural network

x_t = Input at time step t

o_t = Output at time step t



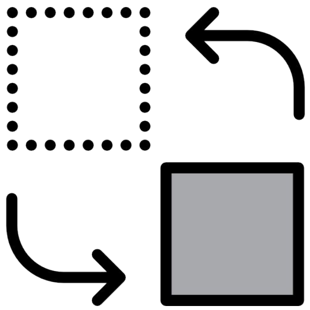
Unfolding Computational Graph



Parameter sharing

RNNs are specified in terms of *transition from one state to another*, with shared weights across all time steps, thus allowing to

- Generalize to sequence lengths not seen during training
- Share statistical strength across different positions in time
    - “I moved to Berlin in 2020” vs. “In 2020, I moved to Berlin”



Backpropagation through time (BPTT)

- Simply apply back-propagation to unfolded graph
- Can not parallelize because computation depends on previous state
- Must step through entire graph

→ Expensive to train: States must be stored until being reused in backward pass

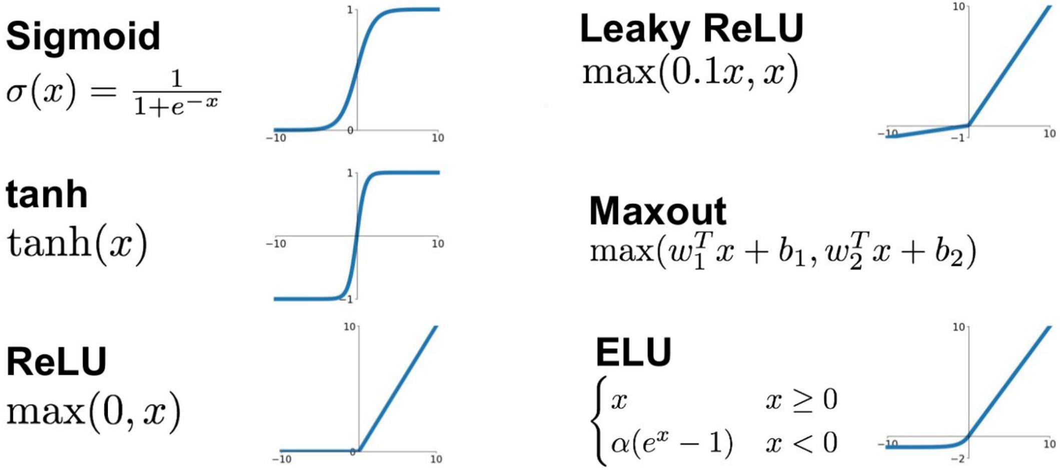
Use of shared parameters in different time steps assumes that conditional probability distribution is stationary

→ Relationship between previous time step and next time step is independent of *t*

Activations

Most frequently used hyperbolic tangent function (tanh)

- Bounded output [-1; 1]
- Easier to compute than sigmoid



Linear Units can have unbounded output and are likely to explode

→ Avoid in RNN

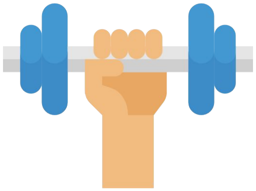
[11, 12]

Vanishing/Exploding Gradients with RNNs

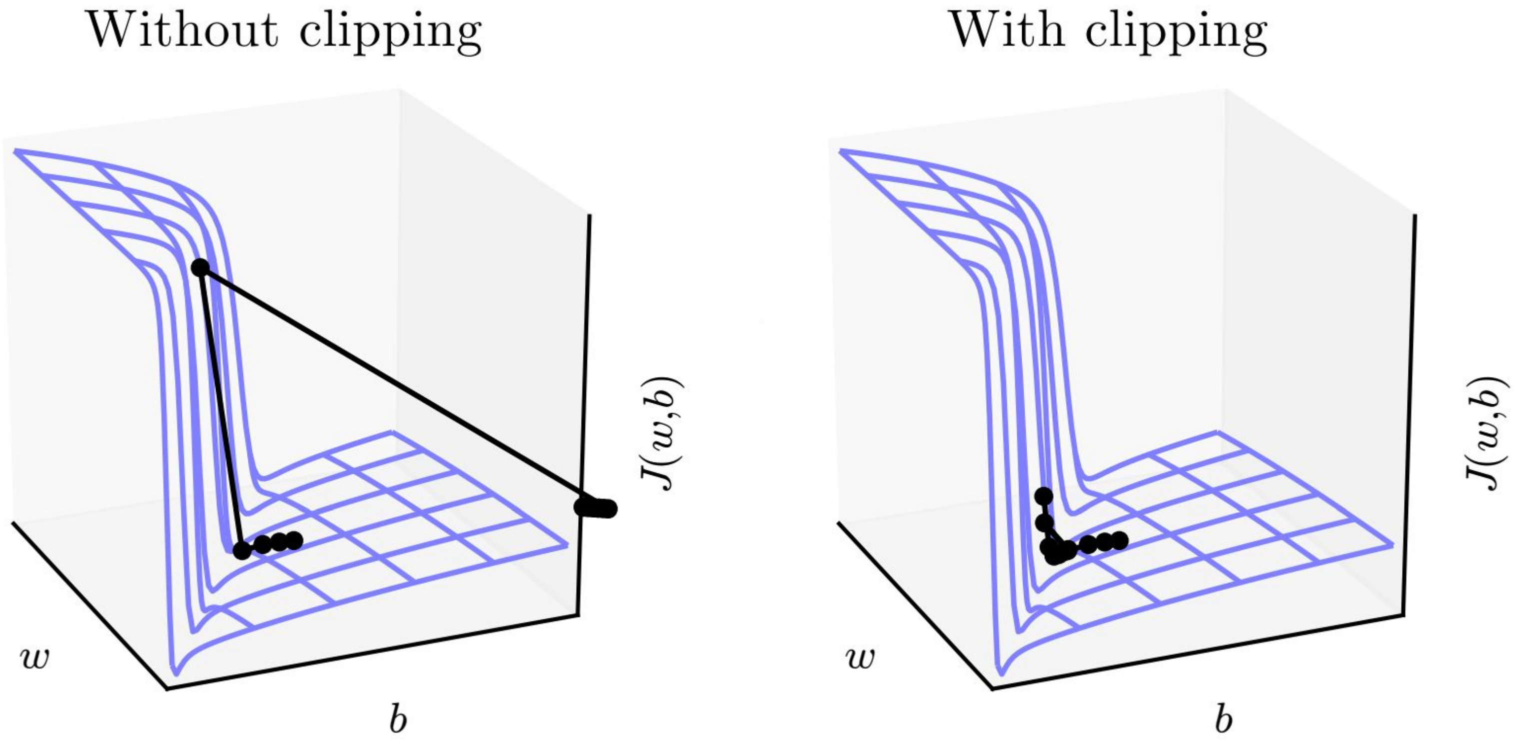
- Gradients propagated through many stages either vanish or explode
- Long-term dependencies must travel through many stages
  - Signal must be able to vanish
- Short-term dependencies have a much stronger signal
  - Small perturbations can interfere with signal from long-term dependencies

Therefore:

- The longer the span of dependencies that needs to be captured, the harder it is to train the network
- Special cells can mitigate this problem



Gradient Clipping to avoid Exploding Gradients



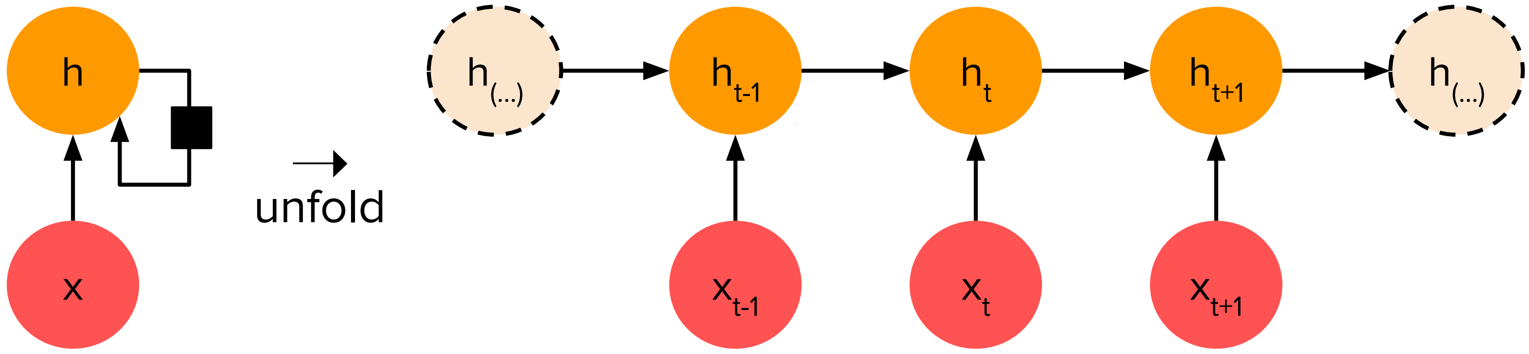
[1]

Variations of RNNs

RNN without outputs

Network just digests input into state **h**

■ = Delay of one time step



RNN with single output at the end

x = Input

h = Hidden state

y = Target

o = Output

L = Loss

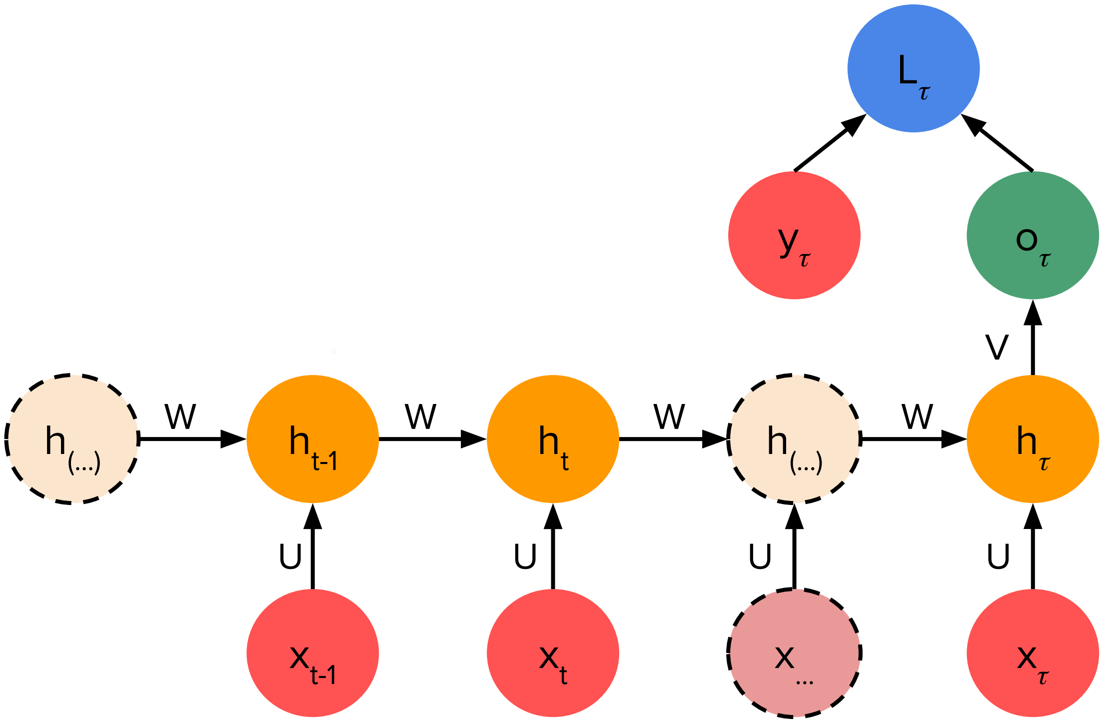
τ = Number of steps

Weight matrices:

U = input to hidden

W = hidden to hidden

V = hidden to output



RNN with output at each time step

x = Input

h = Hidden state

y = Target

o = Output

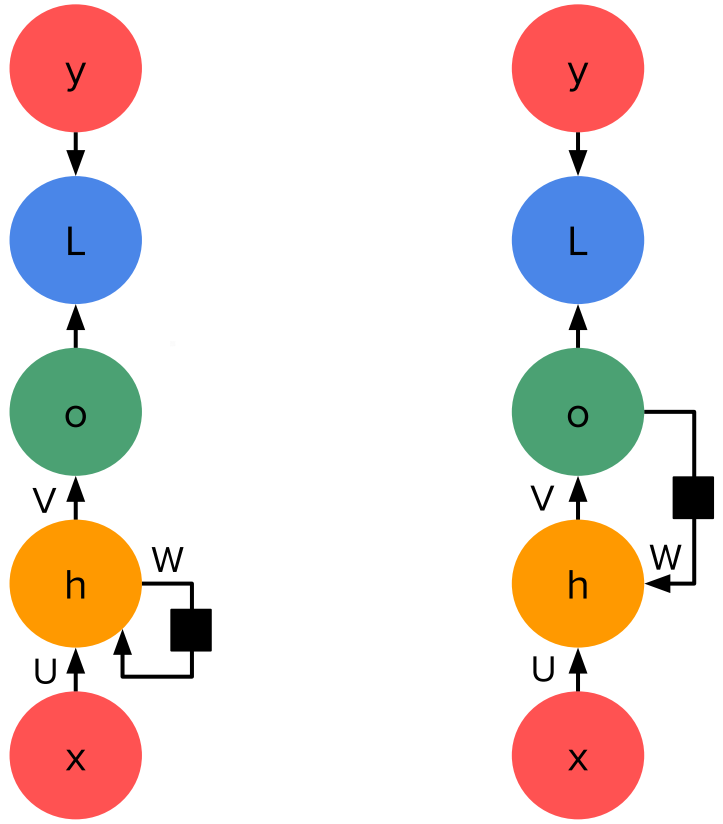
L = Loss

Weight matrices:

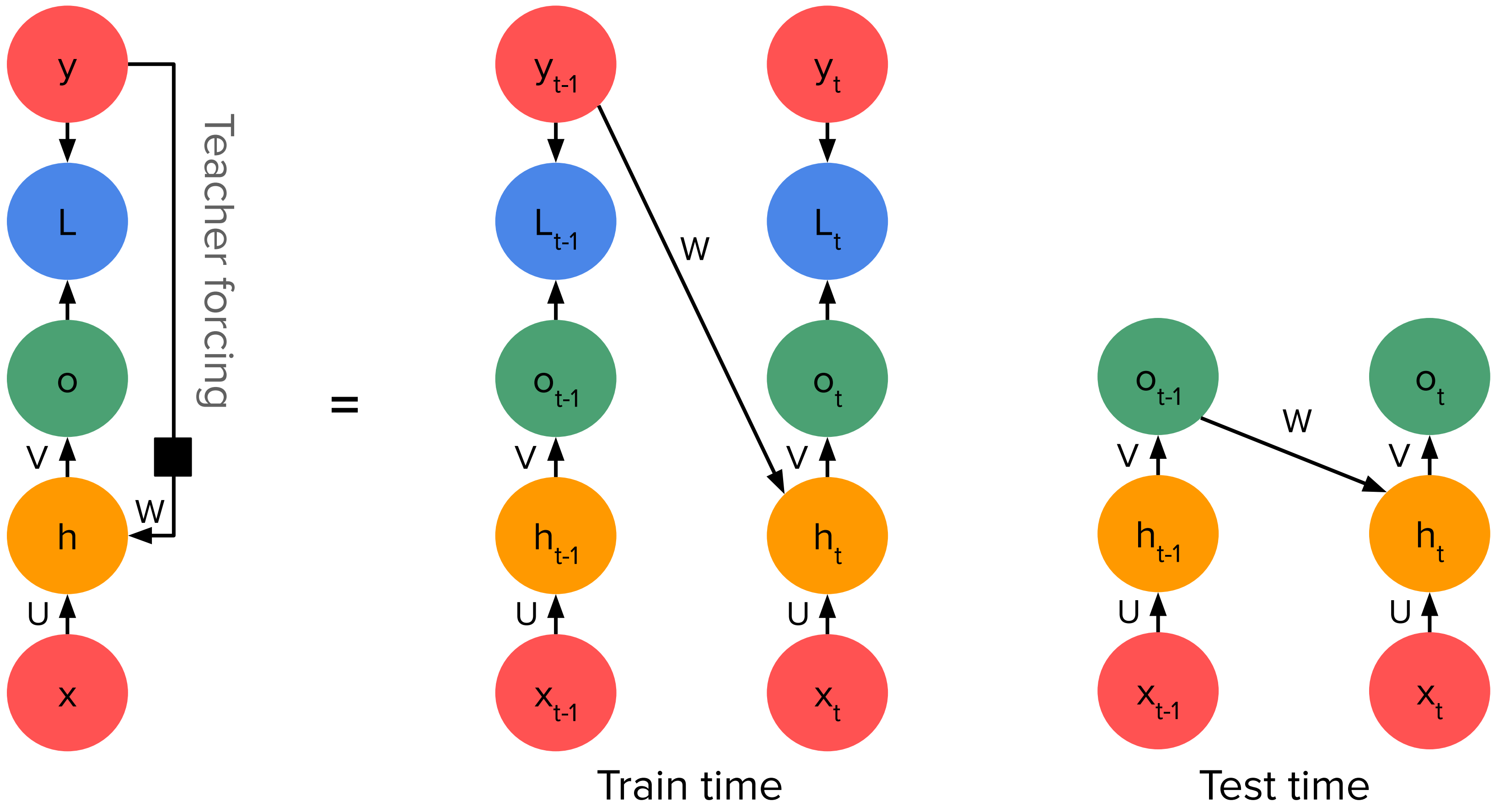
U = input to hidden

W = hidden to hidden

V = hidden to output



Teacher Forcing

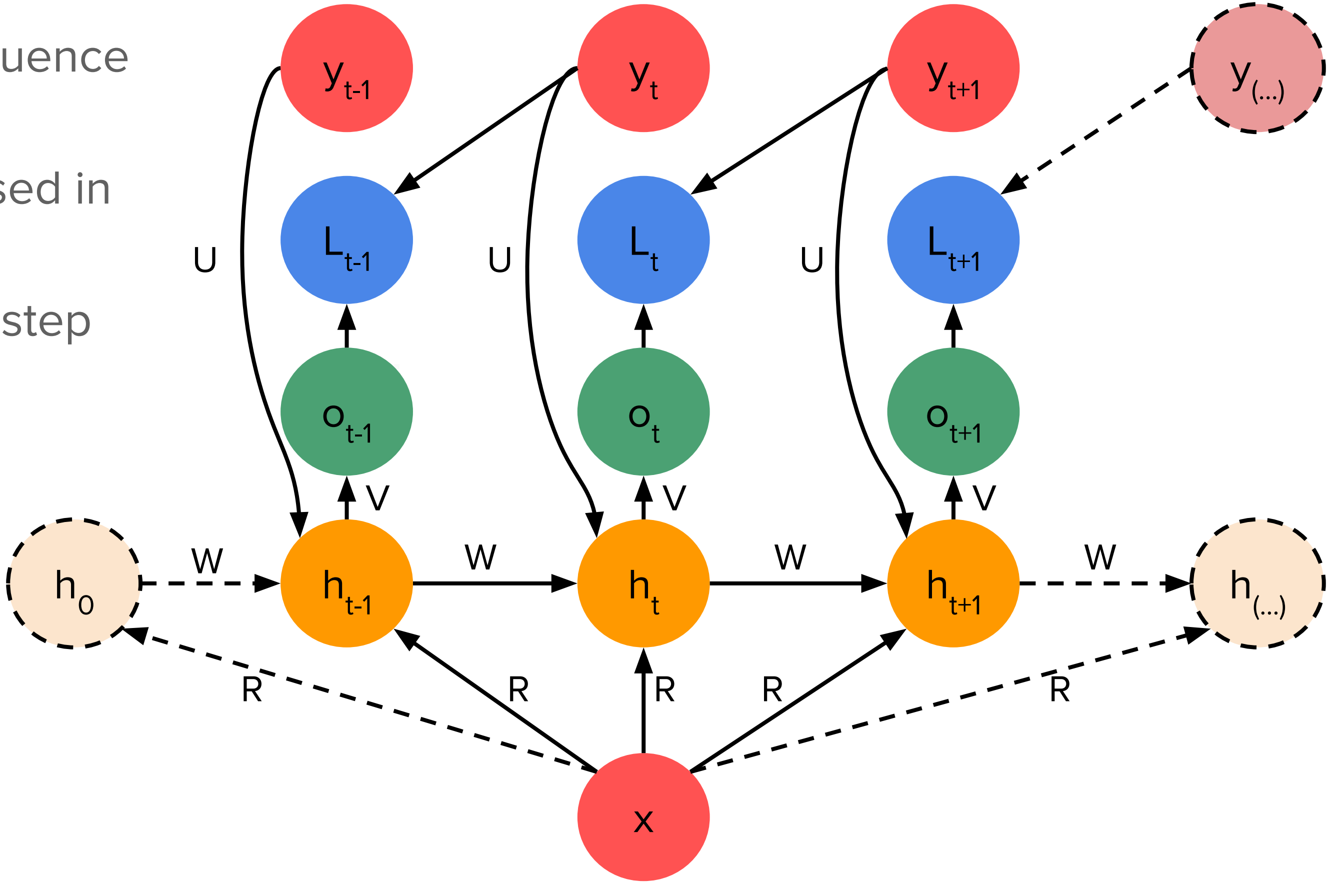


RNNs for Vector to Sequences

Unpacks input into sequence

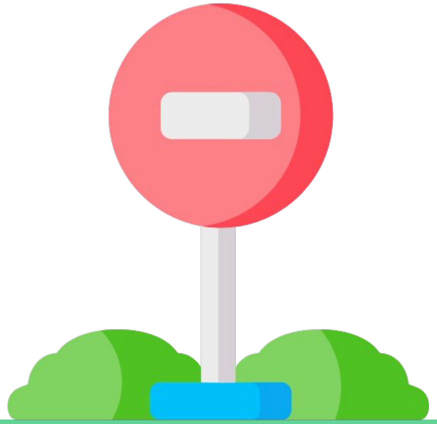
Single vector input **x** used in

- initial state $\mathbf{h_0}$
- input at each time step
- both



How to stop generation?

- Special symbol that can be generated to halt generation
    - Is added to each sequence from the training set
- Extra output (head) that predicts whether to stop or not
- Extra output that predicts sequence length τ
    - First predict τ then sample τ steps worth of data
    - Predict number of remaining steps $\tau - t$



RNN Sequence-to-Sequence of same length

x = Input

y = Target

o = Output

h = Hidden state

L = Loss

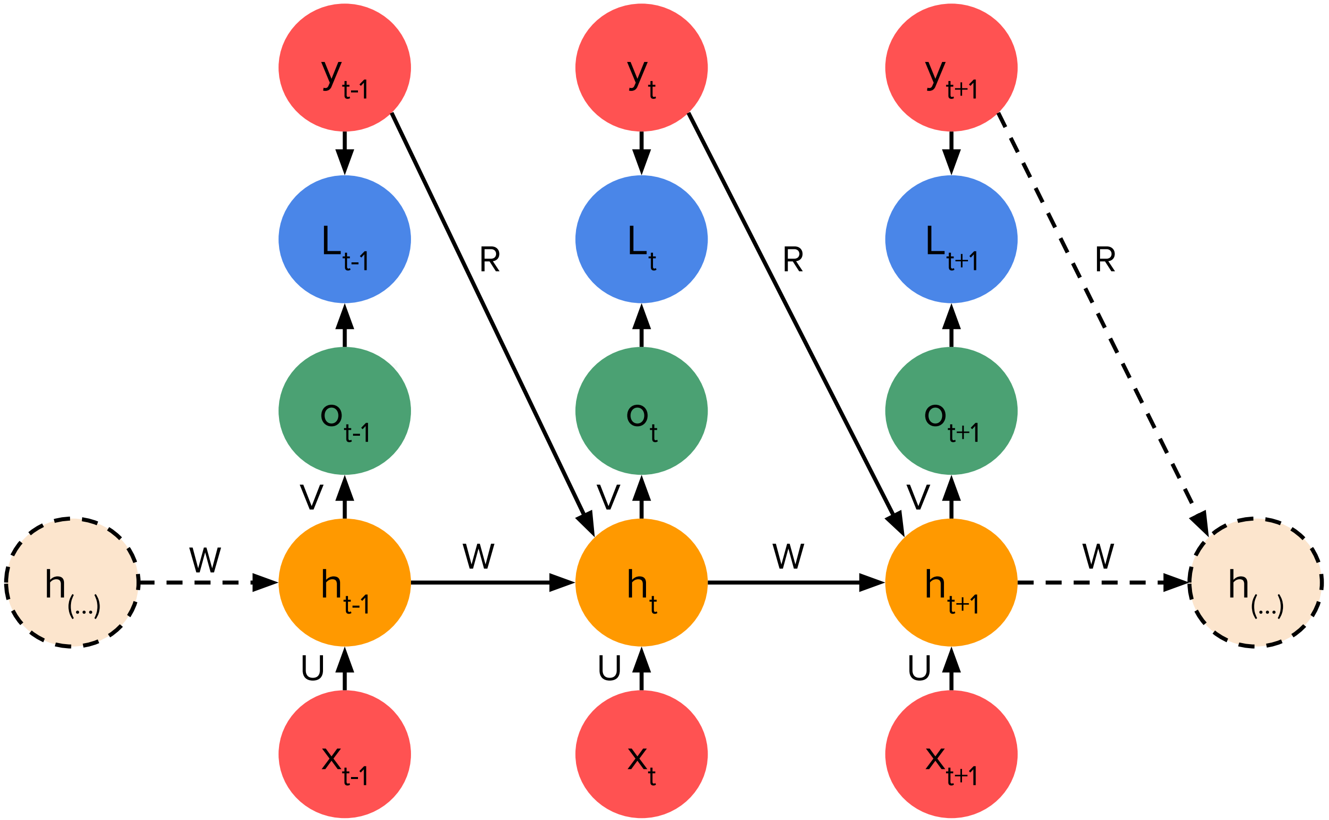
Weight matrices:

U = input to hidden

W = hidden to hidden

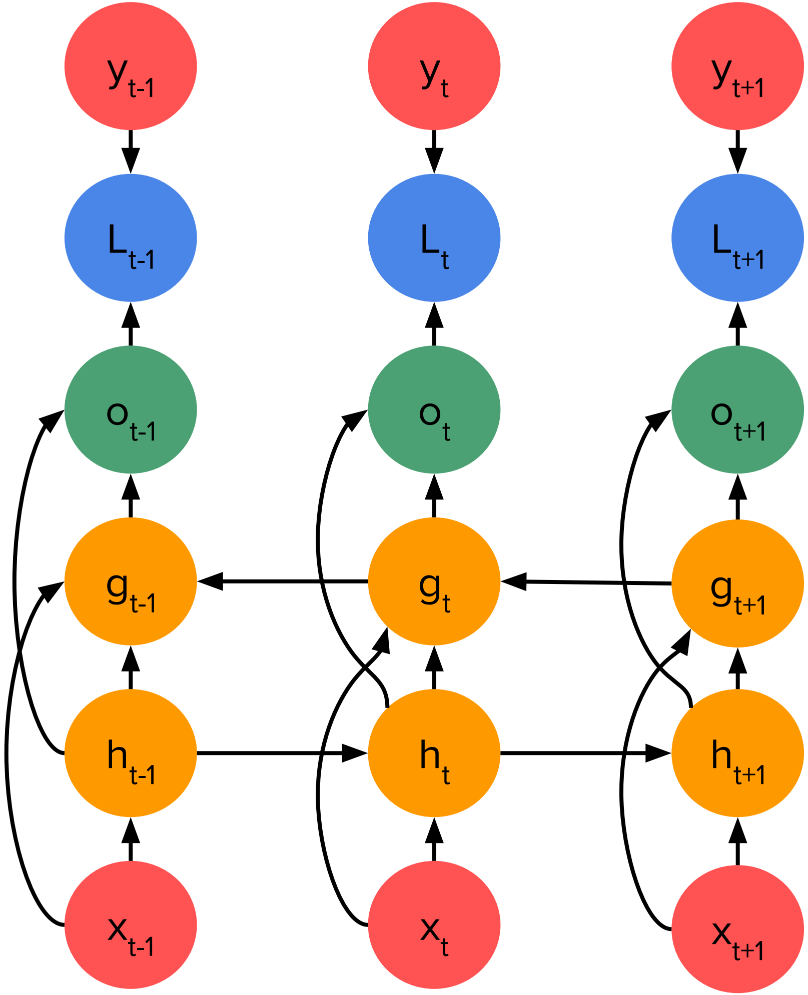
V = hidden to output

R = output to hidden



Bidirectional RNNs

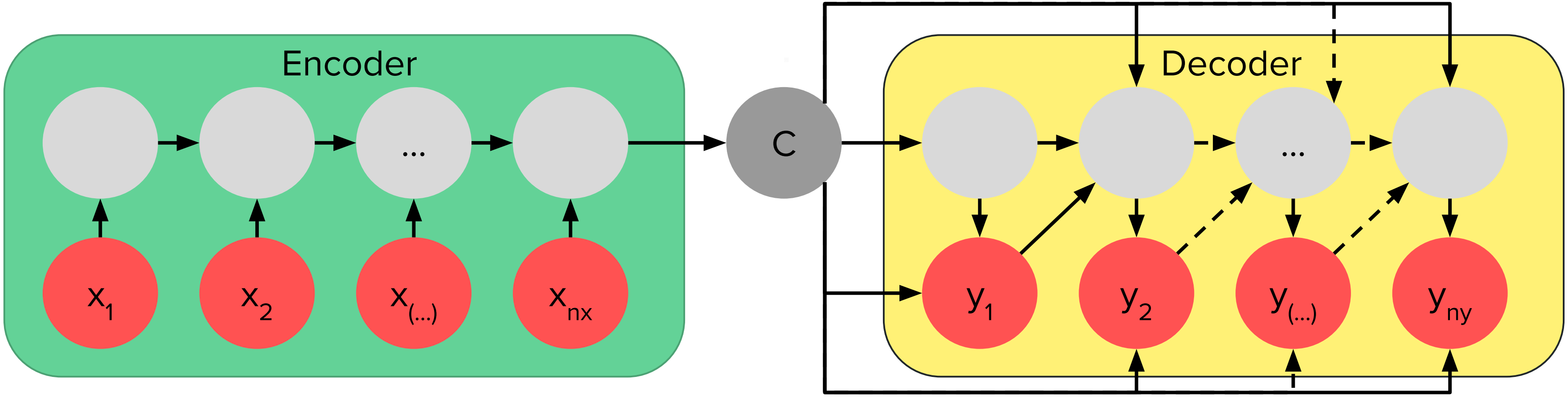
- So far: Only took information from the past
- In some situations, the output depends on the whole sequence (e.g., speech recognition)
- Output units can benefit from past and future information



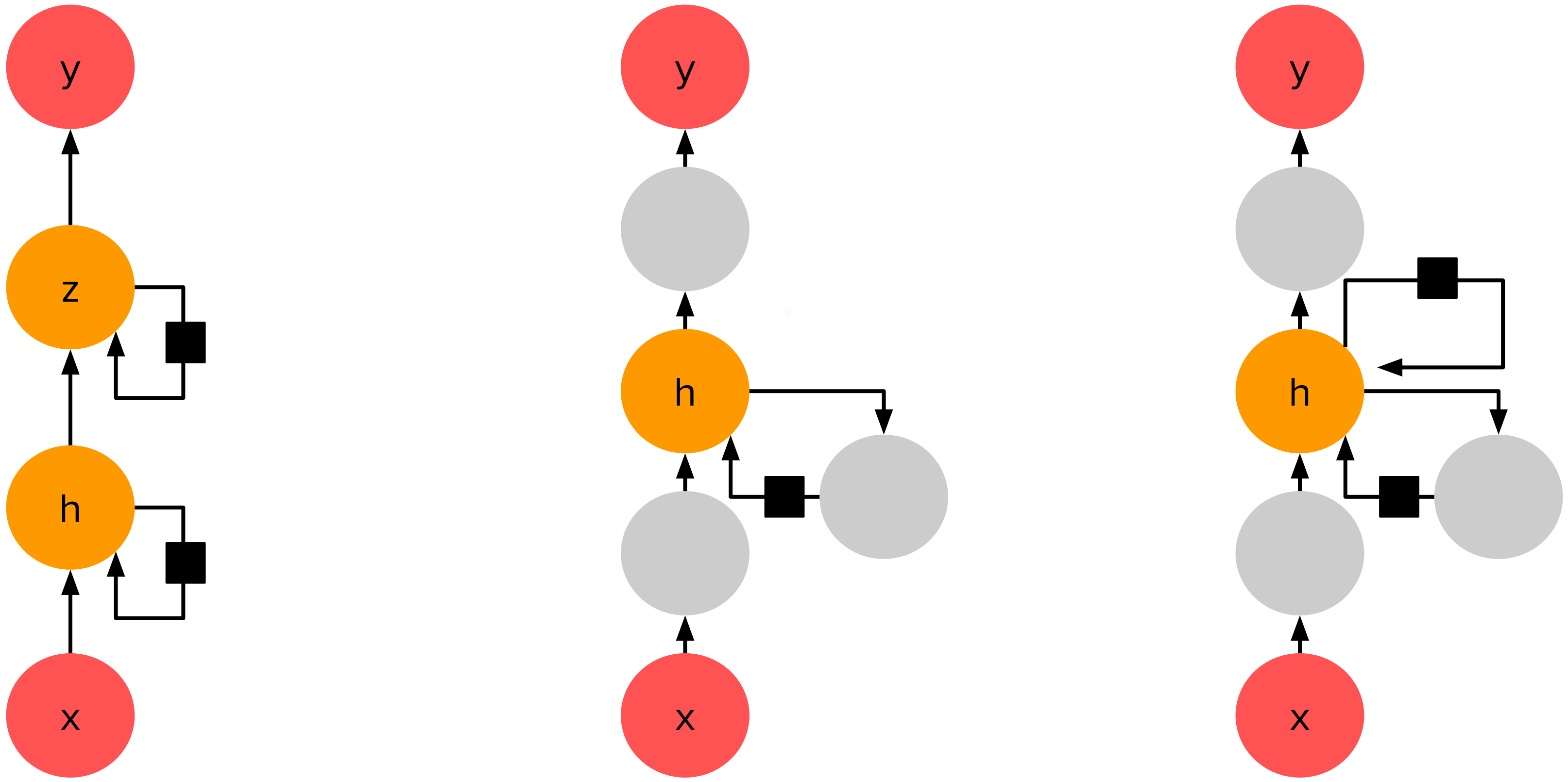
Encoder-Decoder Sequence-to-Sequence

Input and output sequences can have different lengths:

- Process input with encoder (reader) into context C
- Decoder (writer) generates output from context C



Deep RNNs

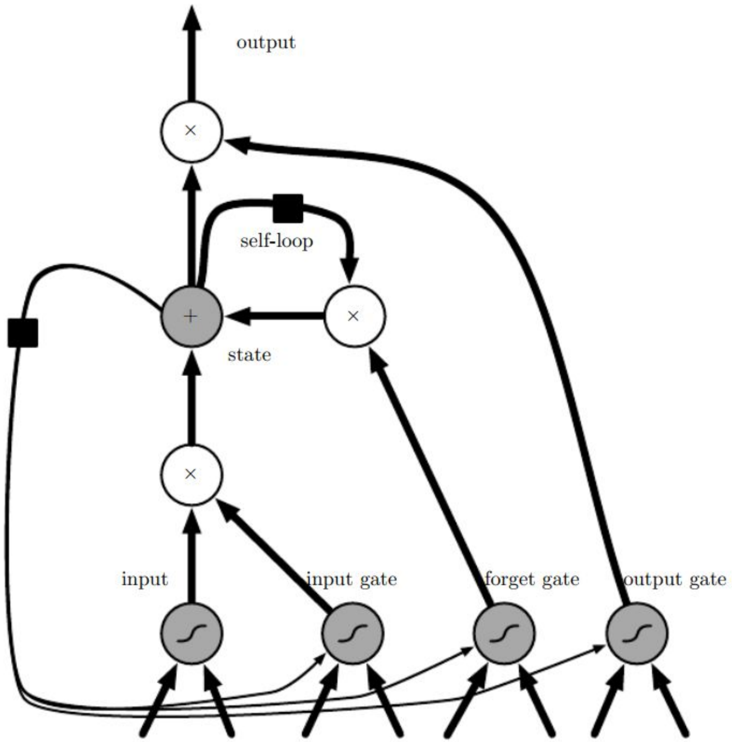


Long Short-Term Memory (LSTM)

Designed to allow learning long-term dependencies

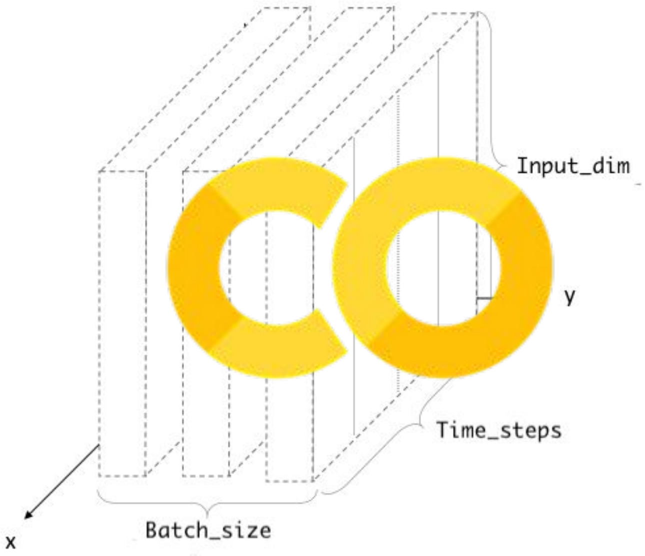
- Introduces self-loops to produce paths where gradient can flow for a long duration
- Introduces gates to change dynamically (controlled by another hidden unit)
- Use addition instead of multiplication

Gated Recurrent Unit (GRU) similar, but simpler (fewer gates)

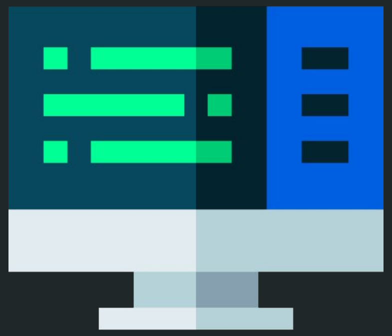


RNNs in Code

https://colab.research.google.com/drive/1NXNcbzezSgIkSUsG3pqv7yBwLAHG3dR0

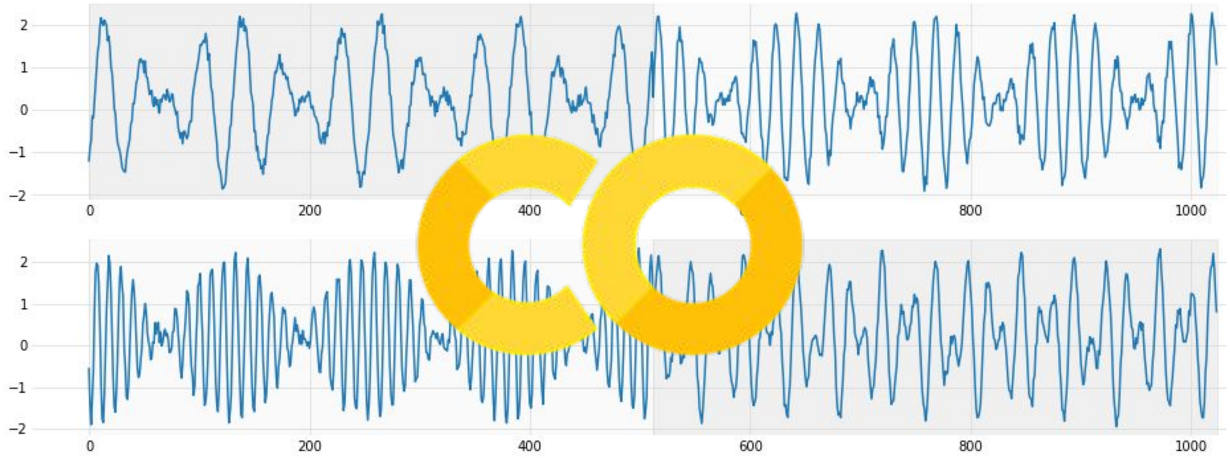


Applications



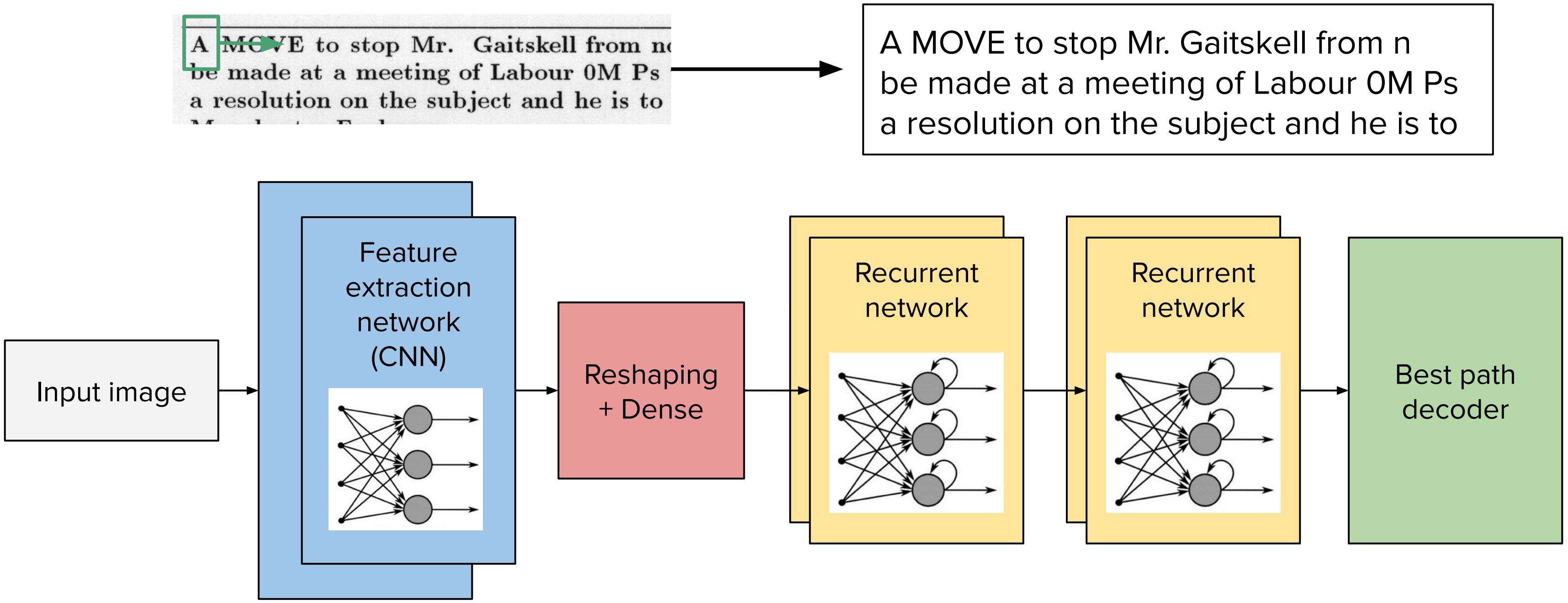
Short-Term Predictions

https://colab.research.google.com/drive/1TIW_emPcv4YjjArDsgapZygOI_j3SFl3



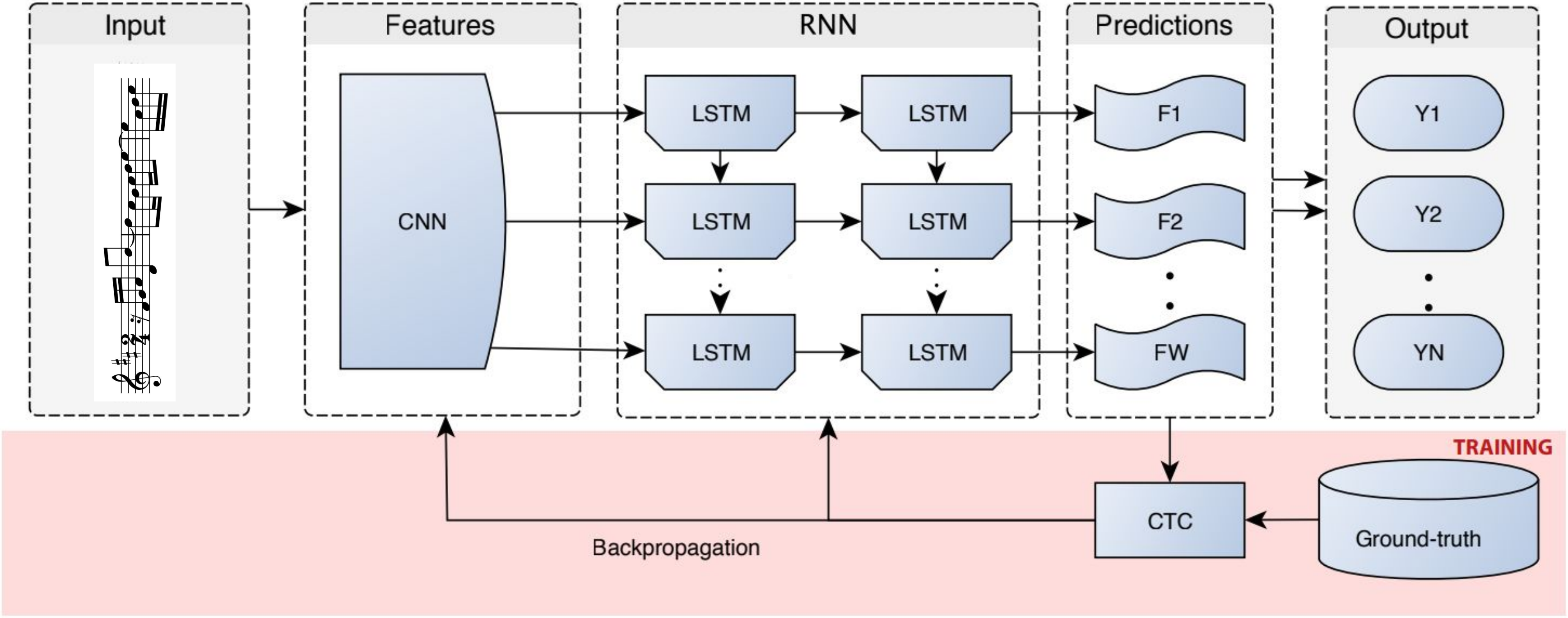
Source: https://github.com/GoogleCloudPlatform/tensorflow-without-a-phd/tree/master/tensorflow-rnn-tutorial

CNNs and RNNs for Optical Character Recognition



Source: https://github.com/apacha/document-analysis/blob/master/assignment2

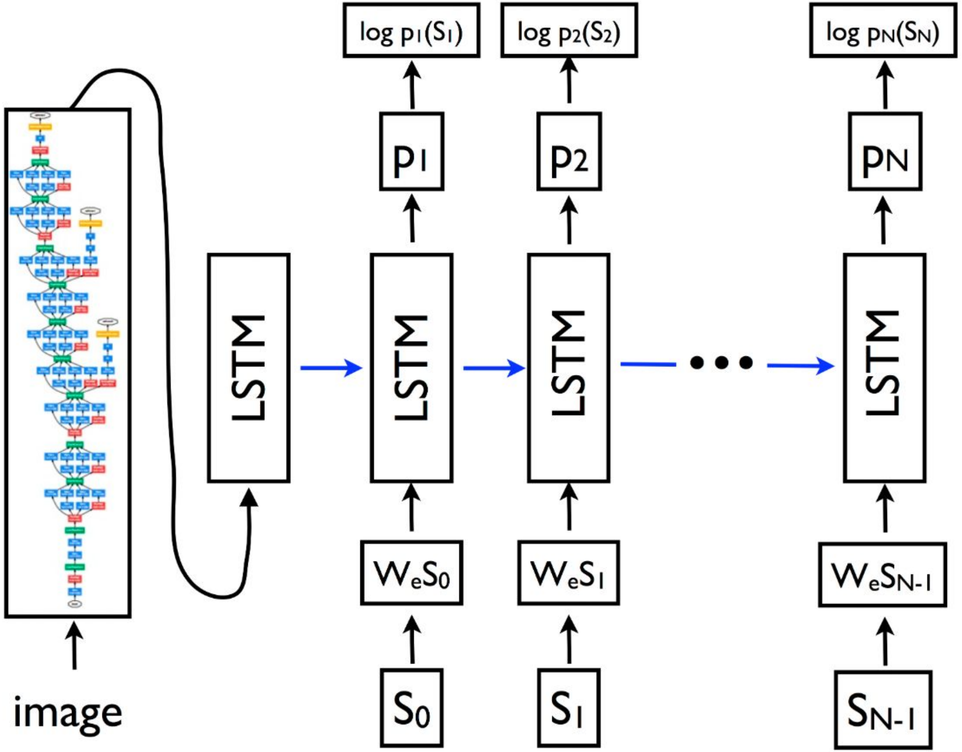
Recognizing Music Scores



Output: `clef-G2, keySignature-DM, timeSignature-2/4, rest-sixteenth, note-F#4_sixteenth, ...`

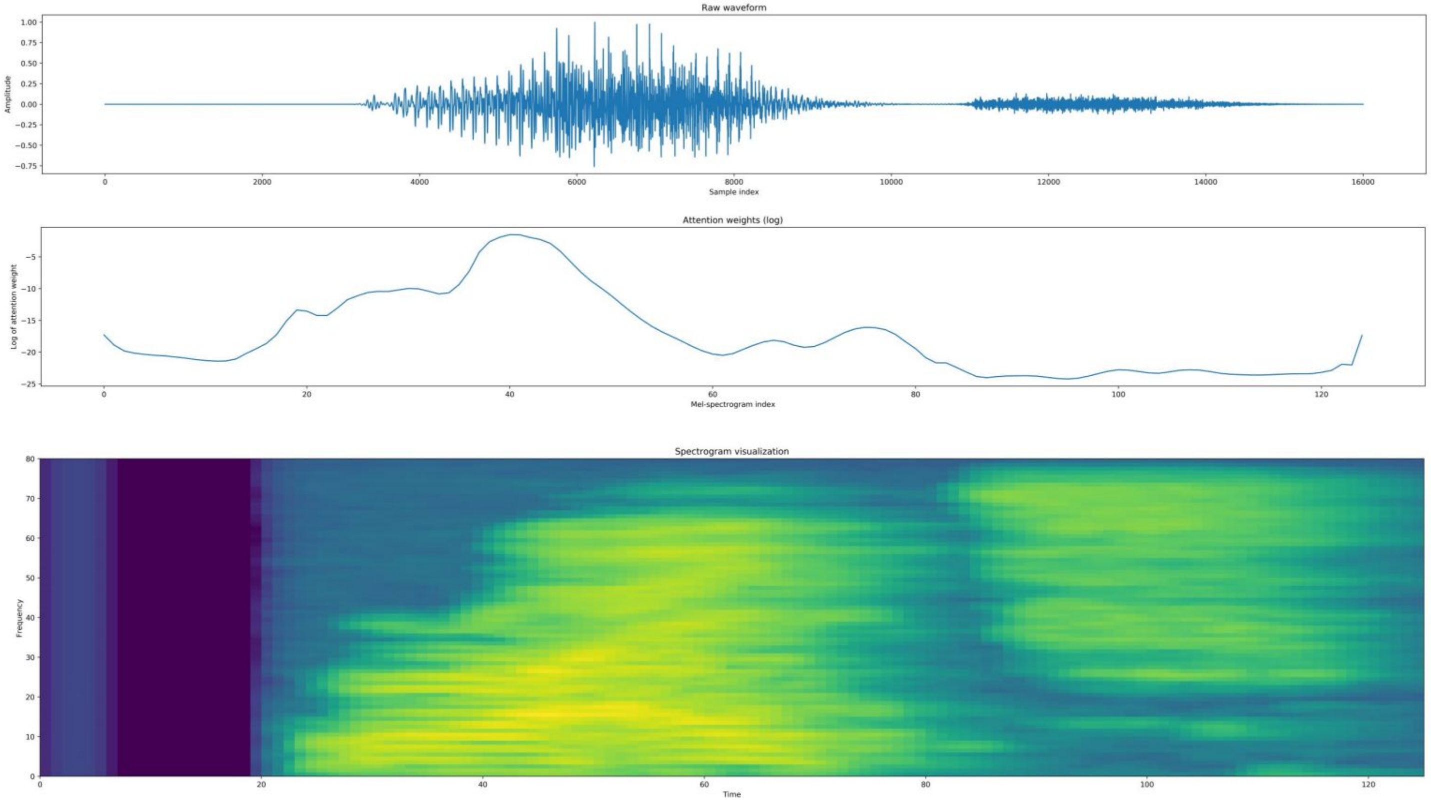
Source [6]

Image Captioning



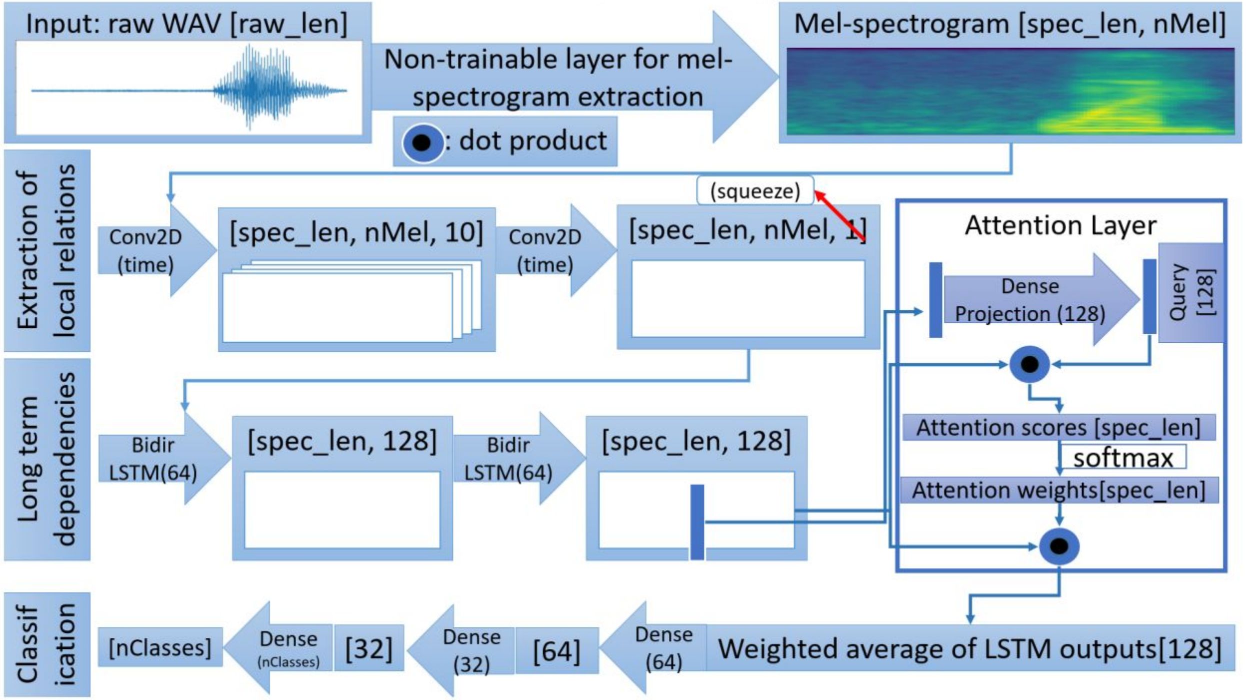
Source: https://ai.googleblog.com/2016/09/show-and-tell-image-captioning-open.html

Speech Recognition



Source: [7]

Speech Recognition

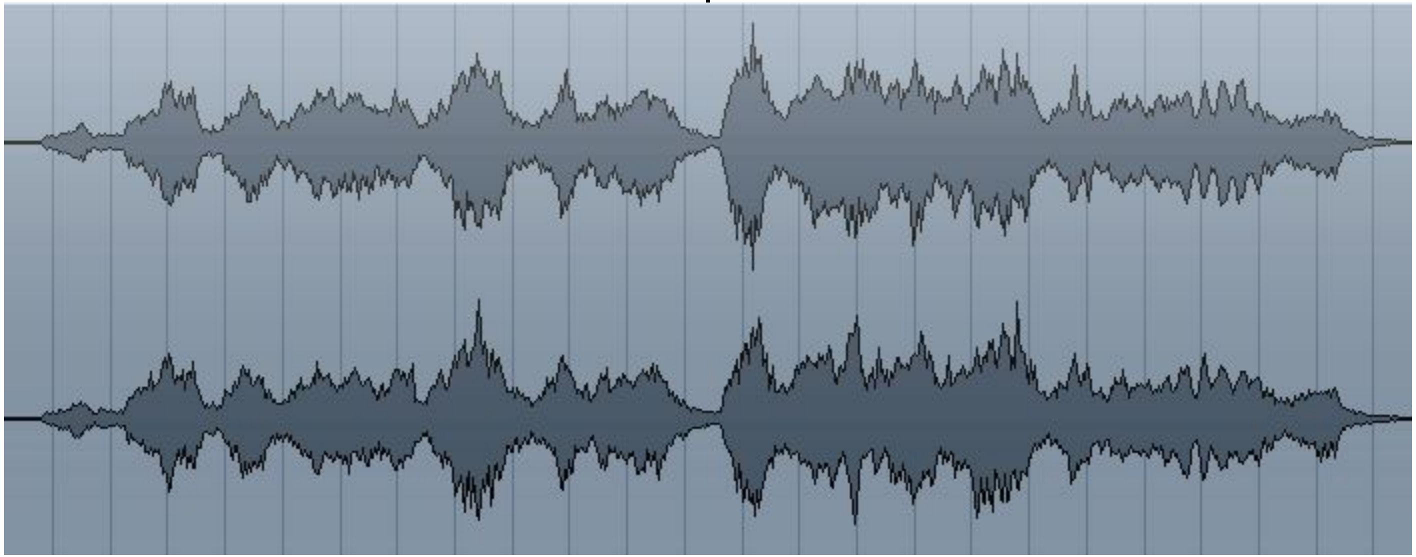


Source: [7]

Connectionist Temporal Classification (CTC)

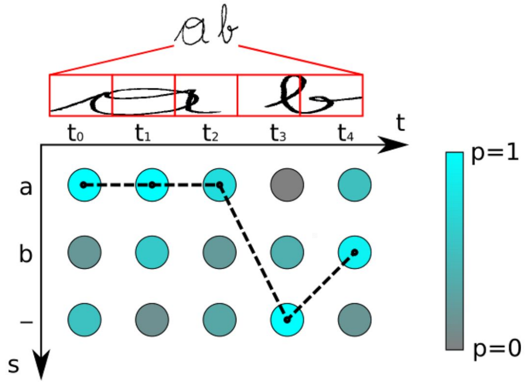
How to handle unaligned data?

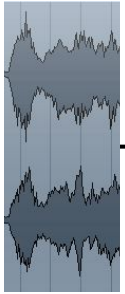
"My friend has too many cats"



Connectionist Temporal Classification (CTC)

Summing over probability of all possible alignments between input and the label.

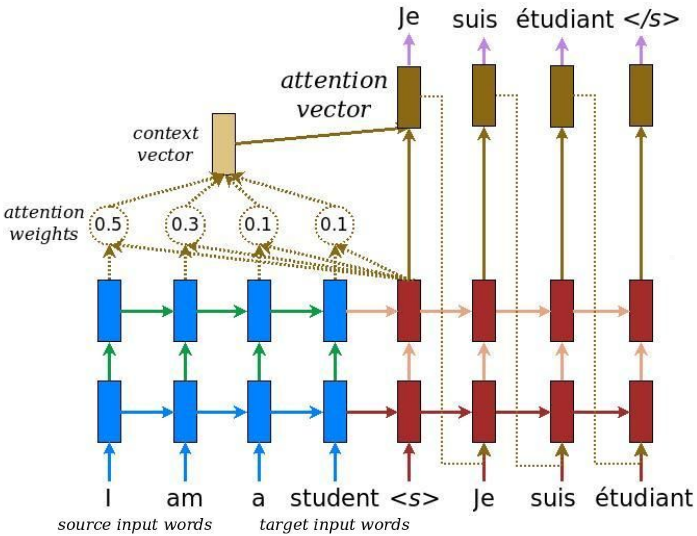


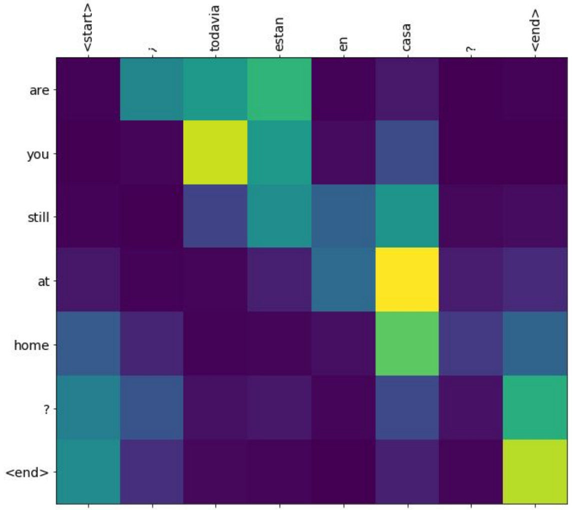


Input	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}
Alignment	h	h	a	s	_	t	t	o	ε	o	o	_
Output	h		a	s	_	t		o		o		_

Source: [9]

Text translation



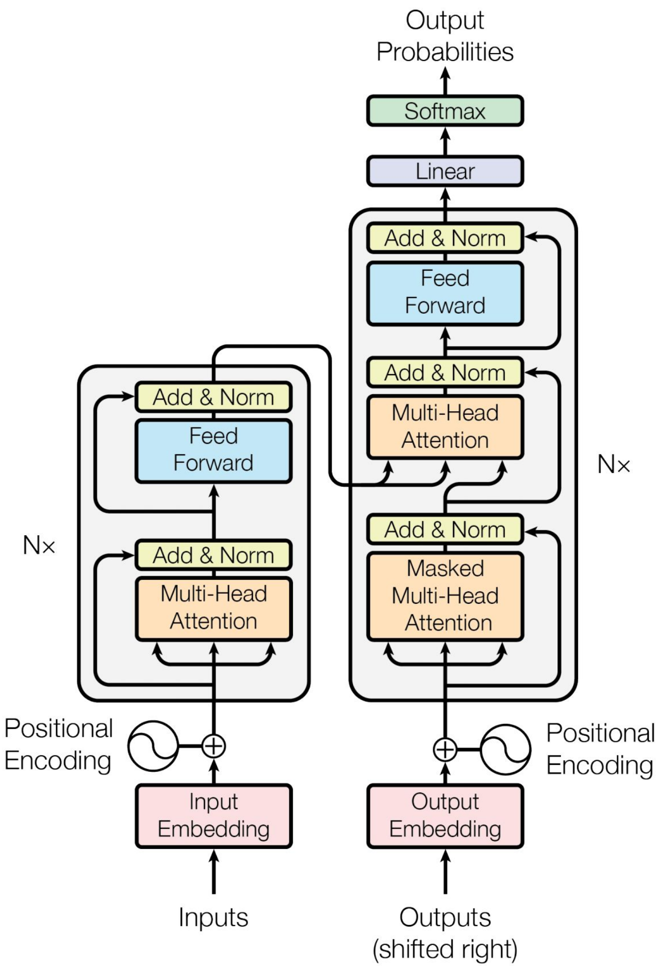


Spanish to English: https://www.tensorflow.org/text/tutorials/nmt_with_attention

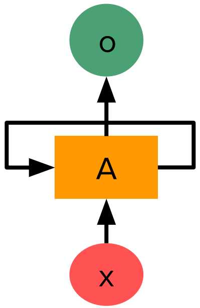
Modern Sequence Processing

- Due to the limitations of RNNs, sequence processing is nowadays mostly done with Attention and Transformers
- Fuels Large Language Models

→ Later lecture

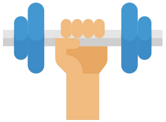


Summary



Recurrent Neural Networks

- have feedback loops and can save state information
- are ideal to process sequential data where a memory is beneficial
- can be trained with Backpropagation Through Time (BPTT)
- are hard to train (vanishing / exploding gradient)
- can process arbitrary input and output sequences



LSTMs / GRU

- can handle long-term dependencies
- replace multiplication with addition

CNNs and RNNs play really well together to solve challenging problems

Literature

1. Deep Learning: http://www.deeplearningbook.org/
2. Karpathy on RNNs: http://karpathy.github.io/2015/05/21/rnn-effectiveness/
3. Recurrent Neural Networks for Time Series Forecasting: Current Status and Future Directions: https://arxiv.org/abs/1909.00590v1
4. Simple OCR: http://cs231n.github.io/assignments2019/assignment3/
5. Understanding LSTMs (Blog): https://colah.github.io/posts/2015-08-Understanding-LSTMs/
6. RNN-Walkthrough: https://github.com/gabrielloye/RNN-walkthrough
7. Neural End-To-End OMR: http://ismir2018.ircam.fr/doc/pdfs/33_Paper.pdf
8. Recognizing Speech Commands: https://towardsdatascience.com/recognizing-speech-commands-using-recurrent-neural-networks-with-attention-c2b2ba17c837
9. Understanding CTC: https://towardsdatascience.com/intuitively-understanding-connectionist-temporal-classification-3797e43a86c
10. Neural Machine Translation with Attention mechanism: https://arxiv.org/abs/1409.0473
11. https://towardsdatascience.com/a-comprehensive-guide-on-activation-functions-b45ed37a4fa5
12. https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/

Icon credits

Free icons from Flaticon:

- https://www.flaticon.com/free-icon/asking_900415
- https://www.flaticon.com/free-icon/balloons_609624
- https://www.flaticon.com/free-icon/code_1383431
- https://www.flaticon.com/free-icon/crane_222566
- https://www.flaticon.com/free-icon/idea_427735
- https://www.flaticon.com/free-icon/transfer_179683
- https://www.flaticon.com/free-icon/broken-link_1201519
- https://www.flaticon.com/free-icon/gym_755336
- https://www.flaticon.com/free-icon/stop-sign_2168375