Lecture 8: Sequence Modeling and Recurrent Neural Networks (RNN)

Olexandr Isayev

Department of Chemistry, CMU

olexandr@cmu.edu

Class Project

- Class project 2-minute topic intro by March 17

- Please use a template and fill in ASAP:

- https://docs.google.com/presentation/d/18MzCutGZ4InECsVgFEKUHarXbazgo3UPaXgpWj3brRQ/edit?usp=sharing

Project Title

Team:
Student Name
Student Name

A brief idea of the problem you are trying to solve

Dataset (type of data, size, etc.)

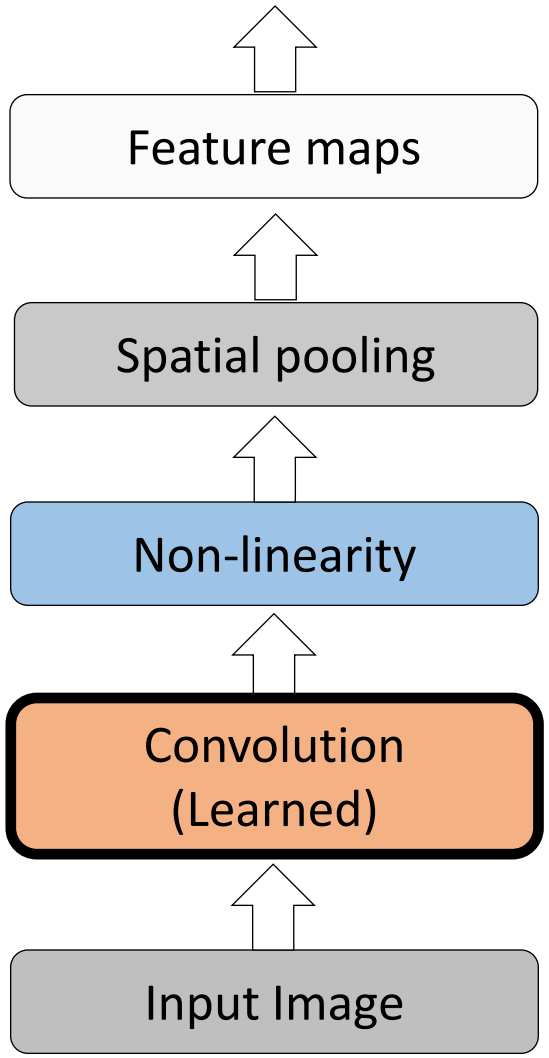
What are you planning to do?

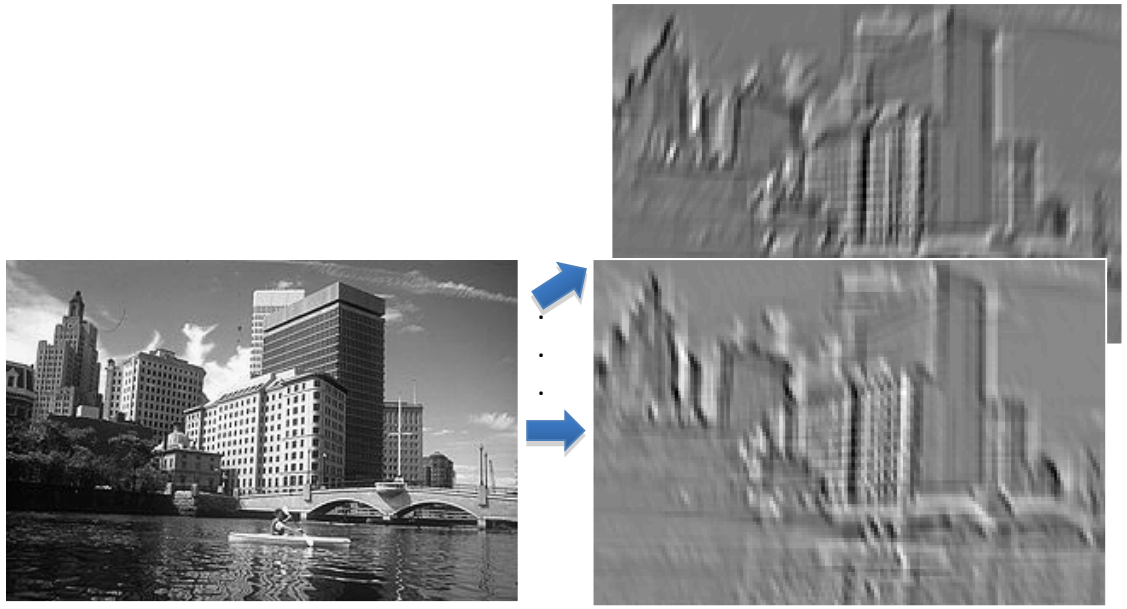
Please add your slides after this template.

HW2 Discussions

Wednesday, March 15

Summary: CNN pipeline





Source: R. Fergus, Y. LeCun

Architectures

1st generation (2012-2013): AlexNet

2nd generation (2014): VGGNet, GoogLeNet

3rd generation (2015): ResNet

4th generation (2016): Wide ResNet, ResNeXt, DenseNet

VGGNet vs. AlexNet

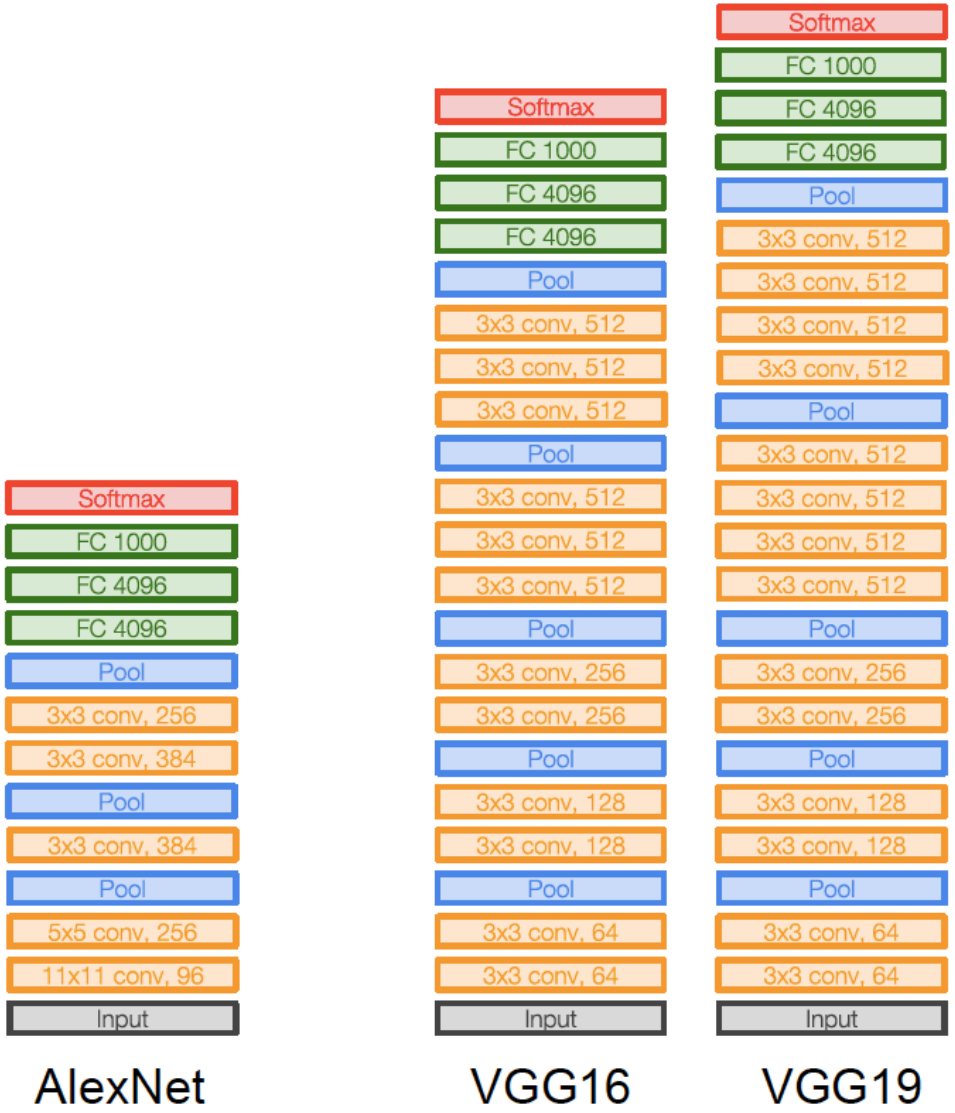
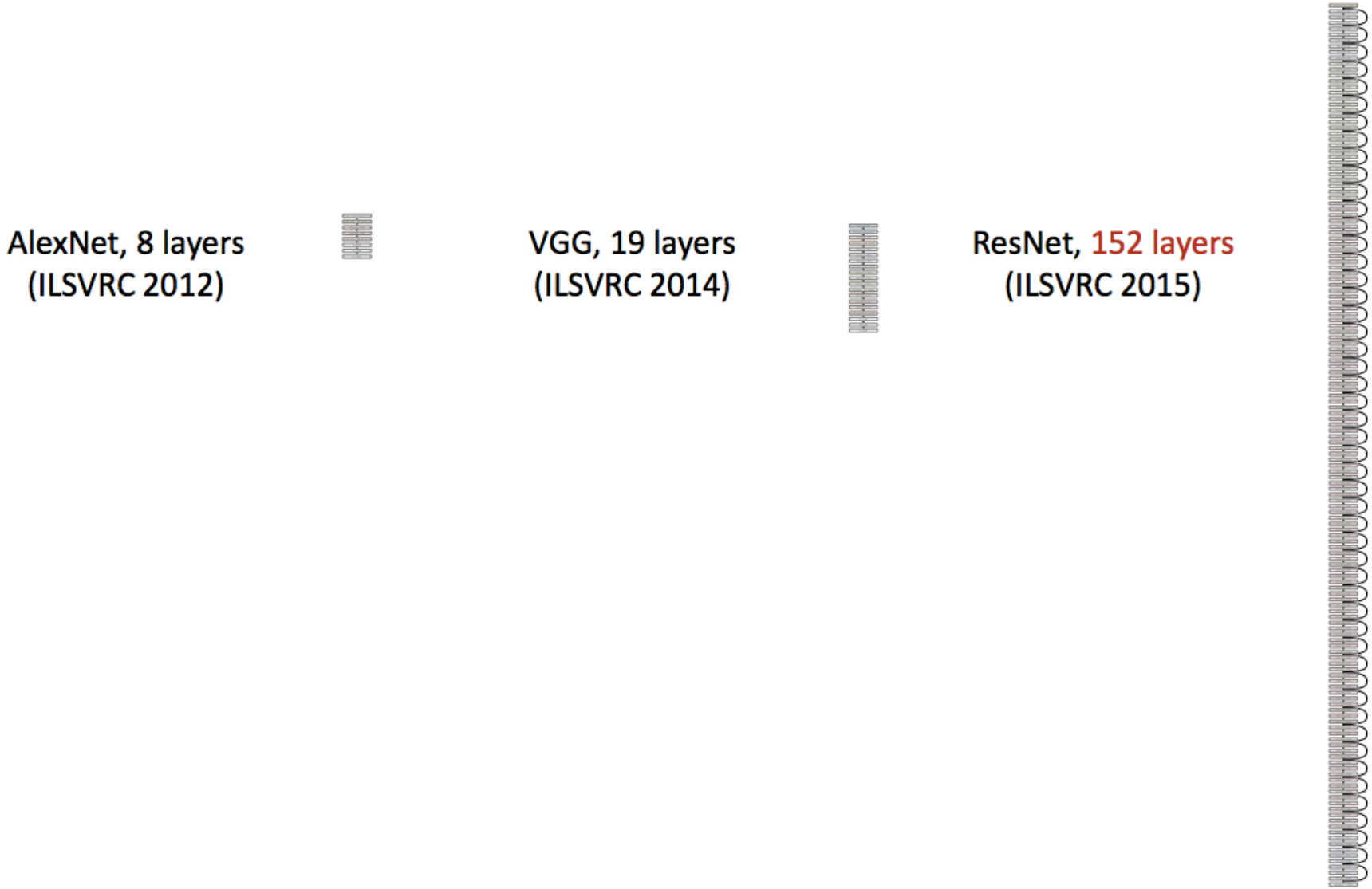


Image source

K. Simonyan and A. Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, ICLR 2015

ResNet: ImageNet 2015 winner



K. He, X. Zhang, S. Ren, and J. Sun, Deep Residual Learning for Image Recognition, CVPR 2016 (Best Paper)

A typical phenomenon

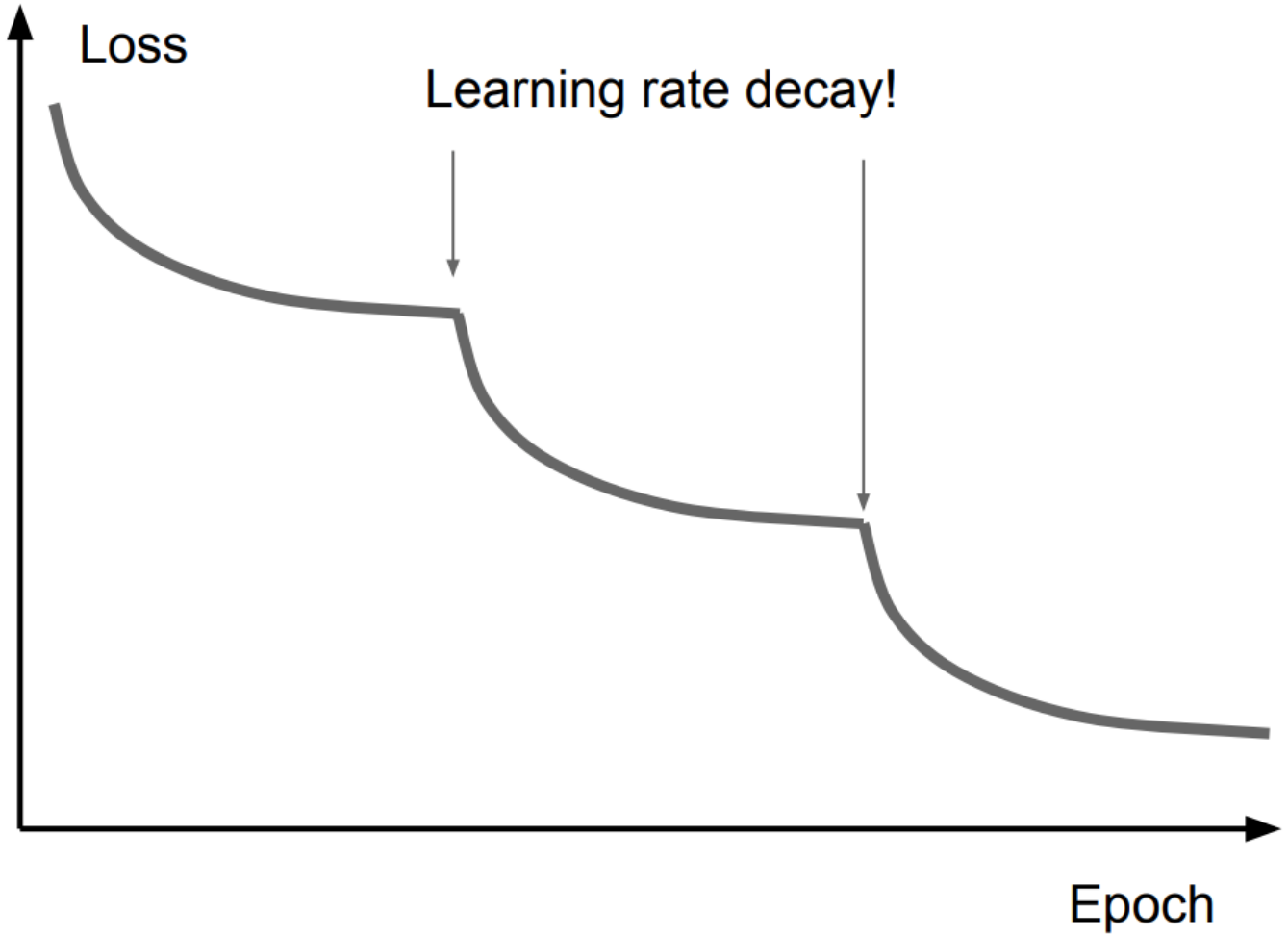


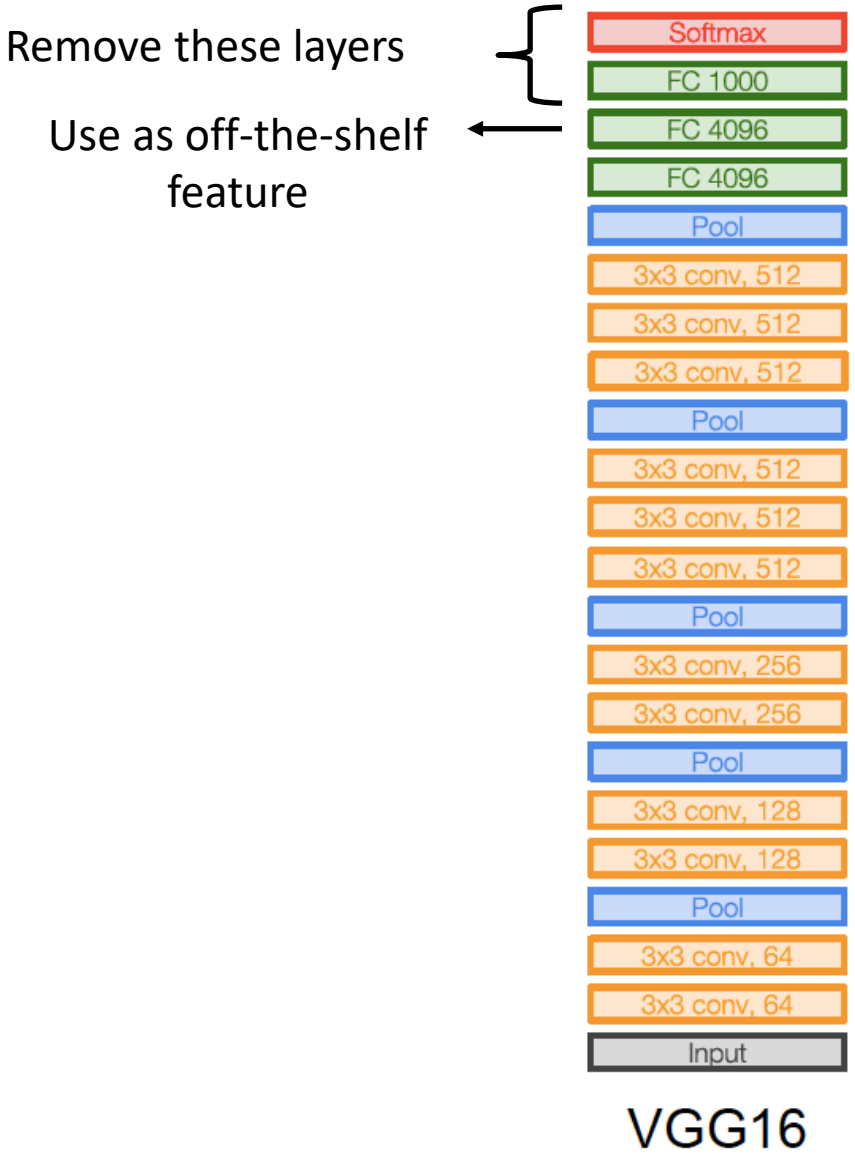
Image source: Stanford CS231n

Data preprocessing

- Zero centering
  - Subtract *mean image* – all input images need to have the same resolution
  - Subtract *per-channel means* – images don't need to have the same resolution
- Optional: rescaling – divide each value by (per-pixel or per-channel) standard deviation

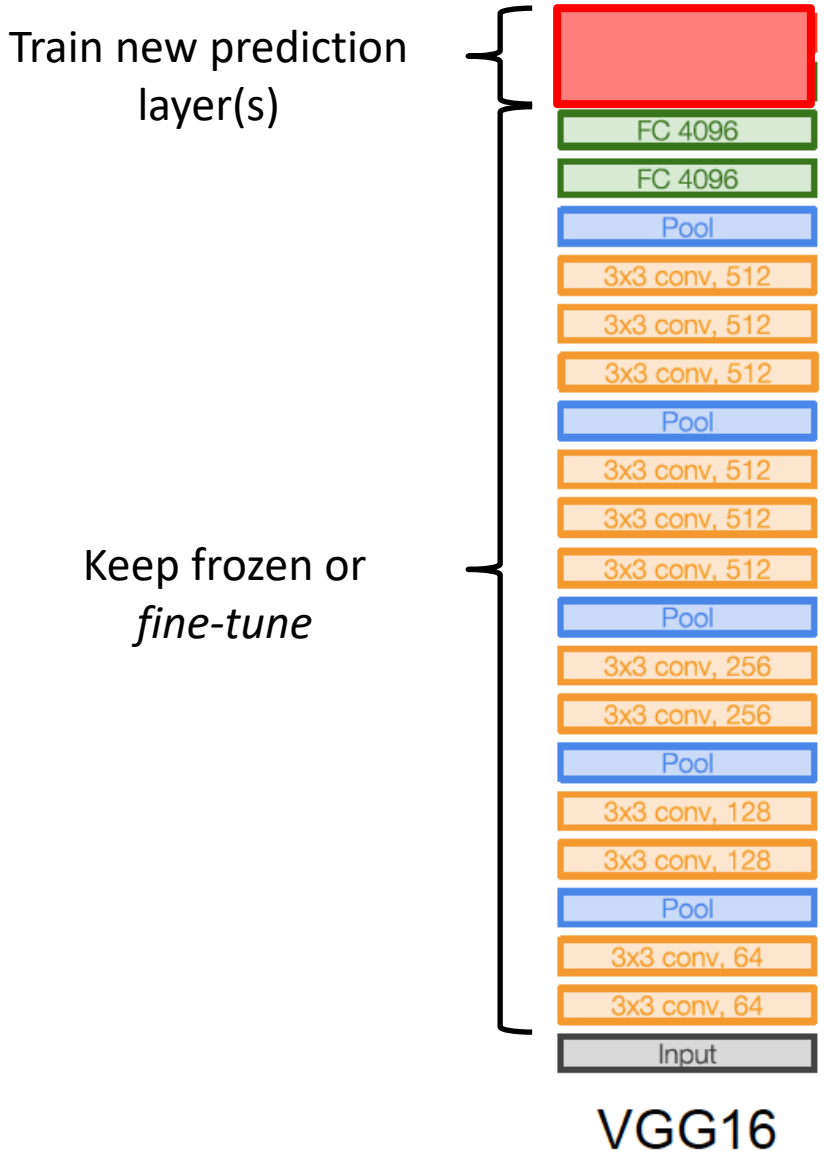
- Be sure to apply the same transformation at training and test time!
  - Save training set statistics and apply to test data

How to use a pre-trained network for a new task?



- Strategy 1: Use as feature extractor

How to use a pre-trained network for a new task?



- Strategy 2: Transfer learning

Sequence Modeling

Sequence Modeling

How to take a **variable length sequence** as **input**?

How to predict a **variable length sequence** as **output**?

Embeddings

An embedding is a mapping of a discrete — categorical — variable to a vector of continuous numbers.

In the context of neural networks, embeddings are *low-dimensional, learned* continuous vector representations of discrete variables.

Neural network embeddings are useful because they can *reduce the dimensionality* of categorical variables and *meaningfully represent* categories in the transformed space.

Embeddings

Neural network embeddings have 3 primary purposes:

- Finding nearest neighbors in the embedding space. These can be used to make recommendations based on user interests or cluster categories.
- As input to a machine learning model for a supervised task.
- For visualization of concepts and relations between categories.

Learning Embeddings

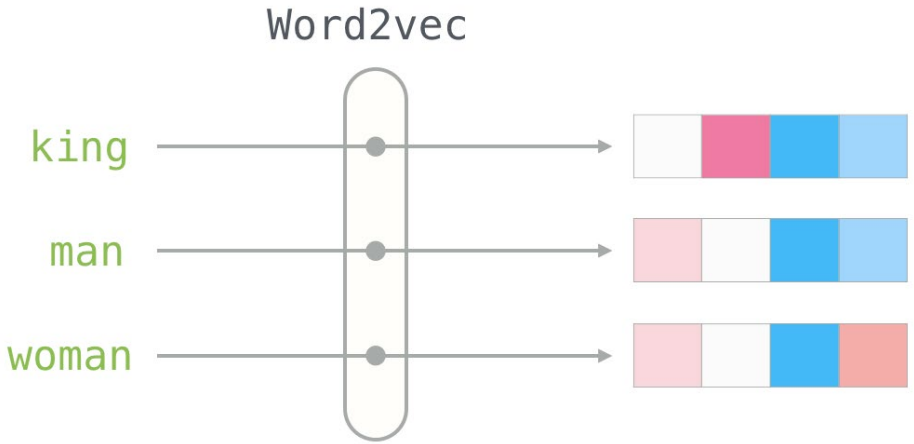
The main issue with one-hot encoding is that the transformation does not rely on any supervision.

We can greatly improve embeddings by ***learning*** them using a neural network on a supervised task.

The resulting embedded vectors are representations of categories where similar categories — relative to the task — are closer to one another.

Word2Vec

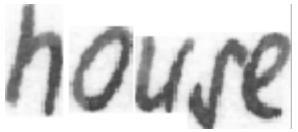
Distributed vector representation for words



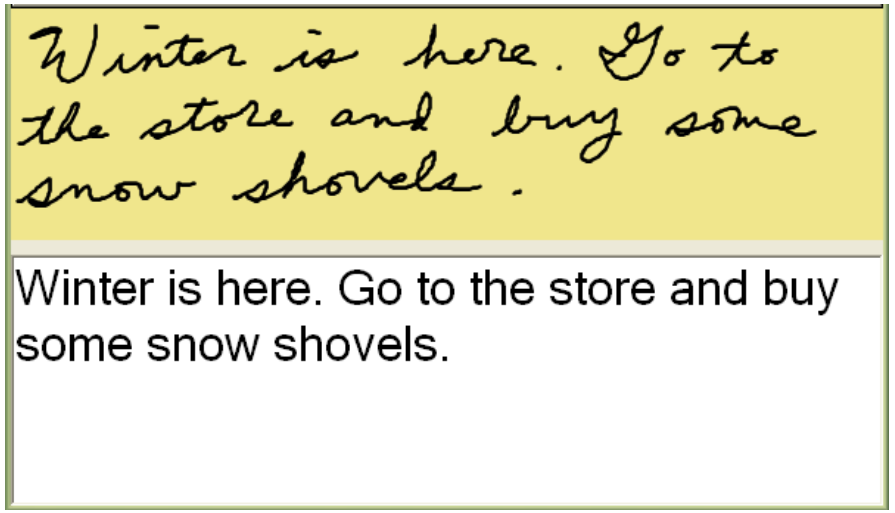
Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. "Distributed representations of words and phrases and their compositionality." In *Advances in neural information processing systems*, pp. 3111-3119. 2013.

Neural Nets for Sequence Modeling

Sequence Modeling: Handwritten Text Translation



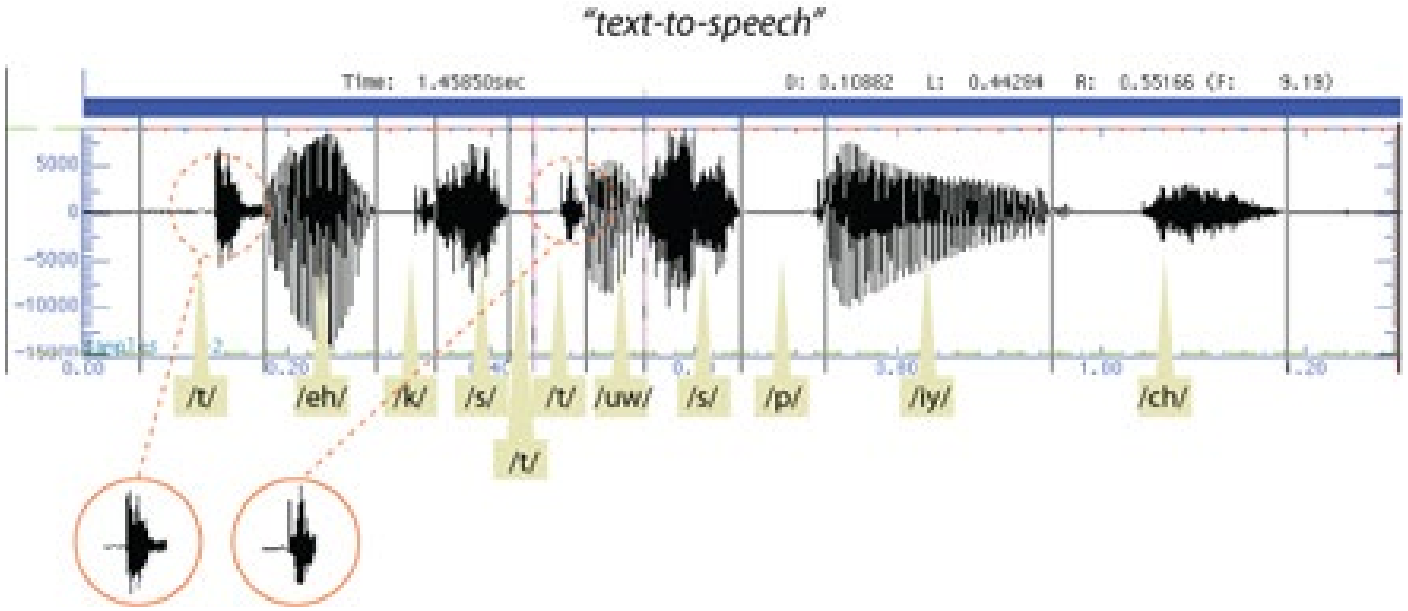
→ "house"



Input : Image
Output: Text

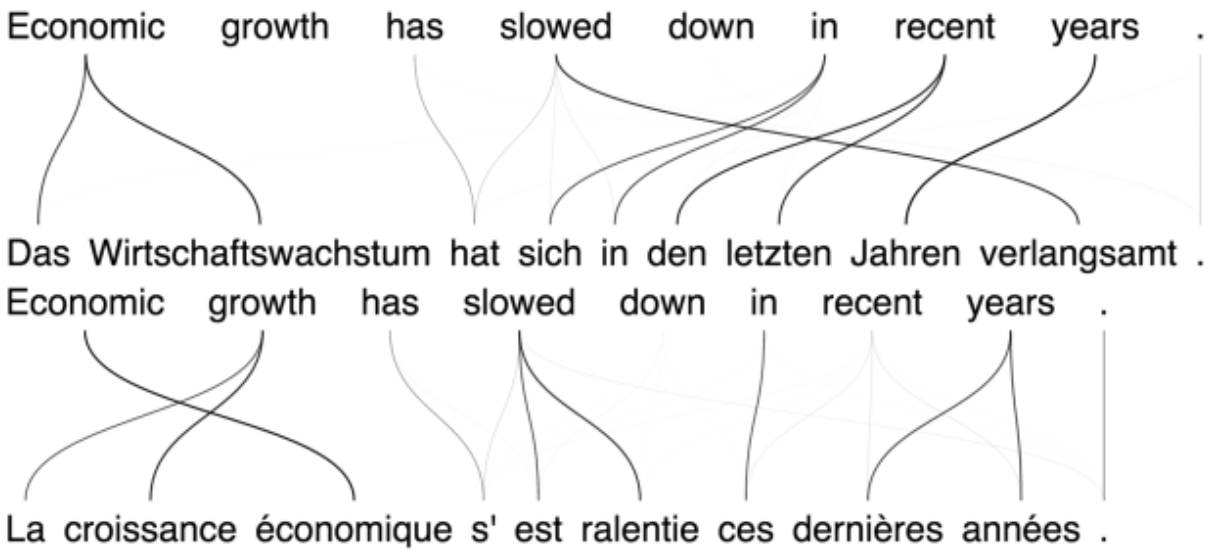
https://towardsdatascience.com/build-a-handwritten-text-recognition-system-using-tensorflow-2326a3487cd5

Sequence Modeling: Text-to-Speech



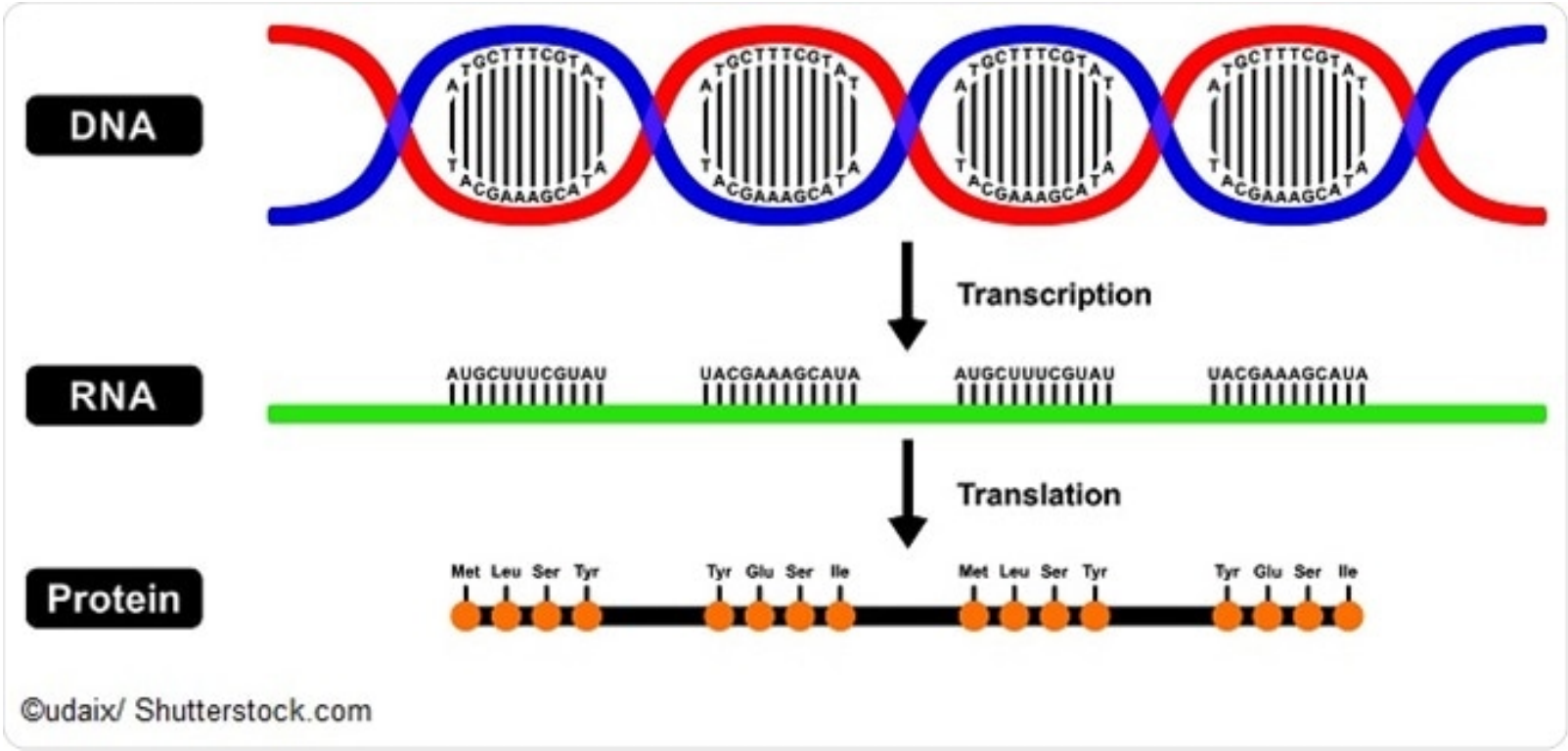
Input : Audio
Output: Text

Sequence Modeling: Machine Translation



Input : Text
Output: Translated Text

Biological sequences

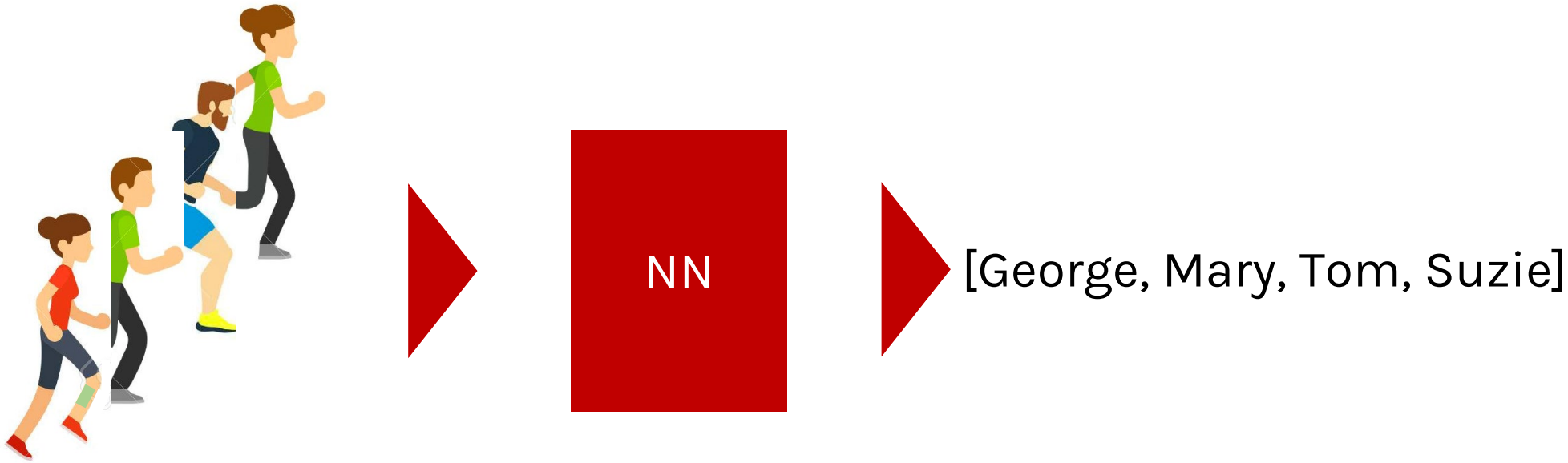


Input : Sequence
Output: biological function, site of methylation, site of translation, 3D structure, …

Recurrent Neural Networks (RNN)

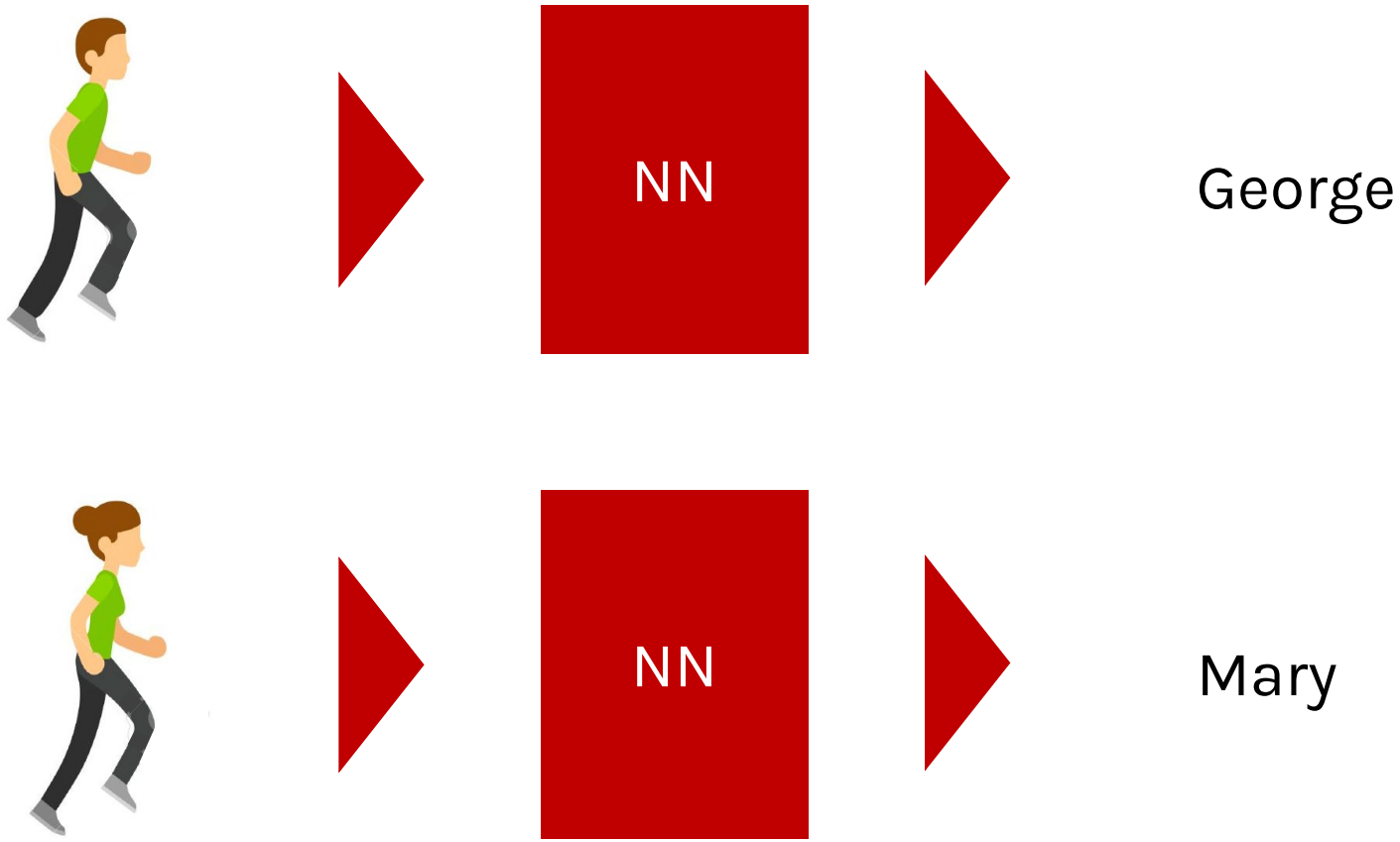
What can my NN do?

Training: Present to the NN examples and learn from them.

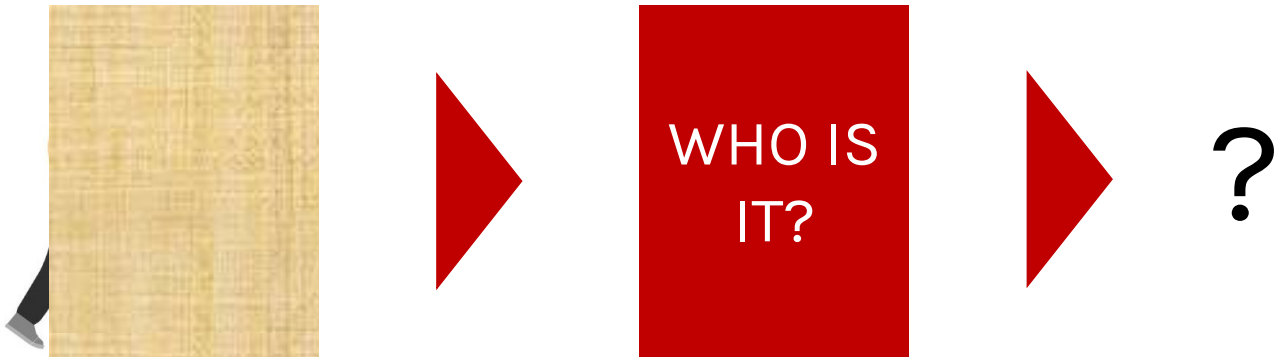


What can my NN do?

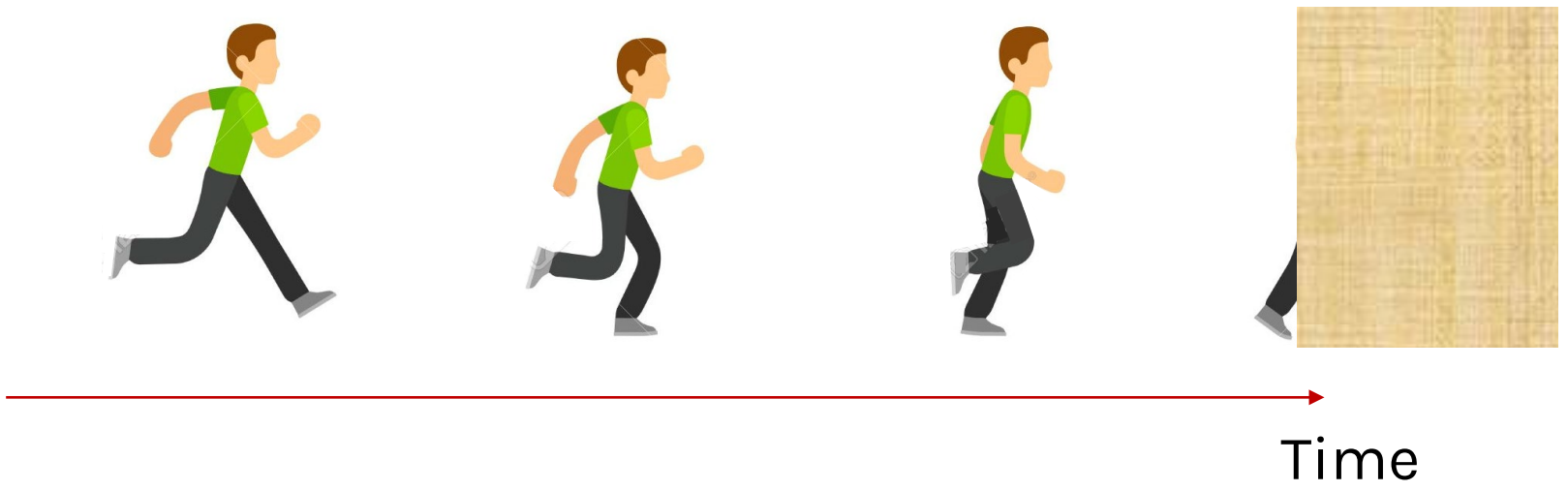
Prediction: Given an example



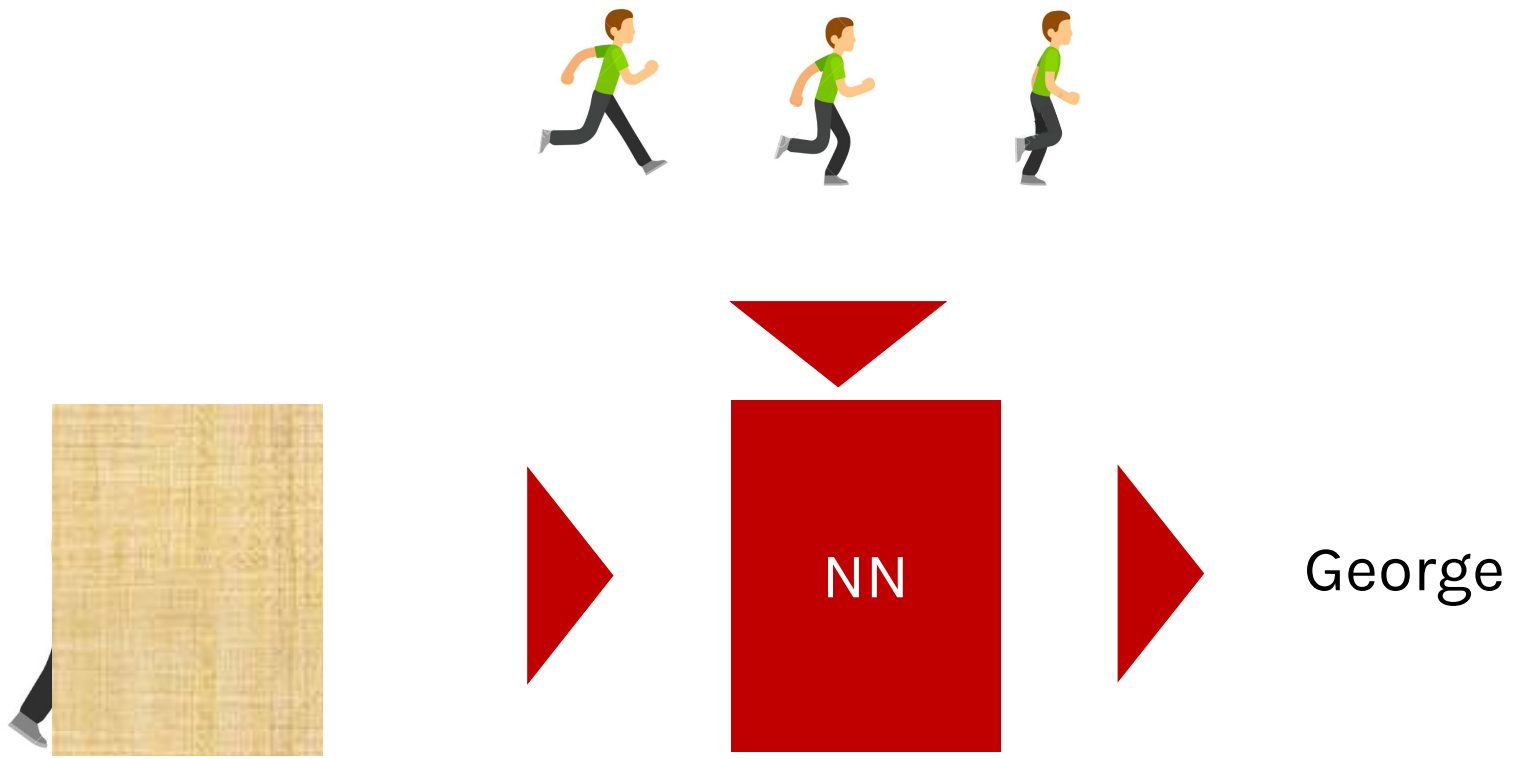
What my NN can NOT do?



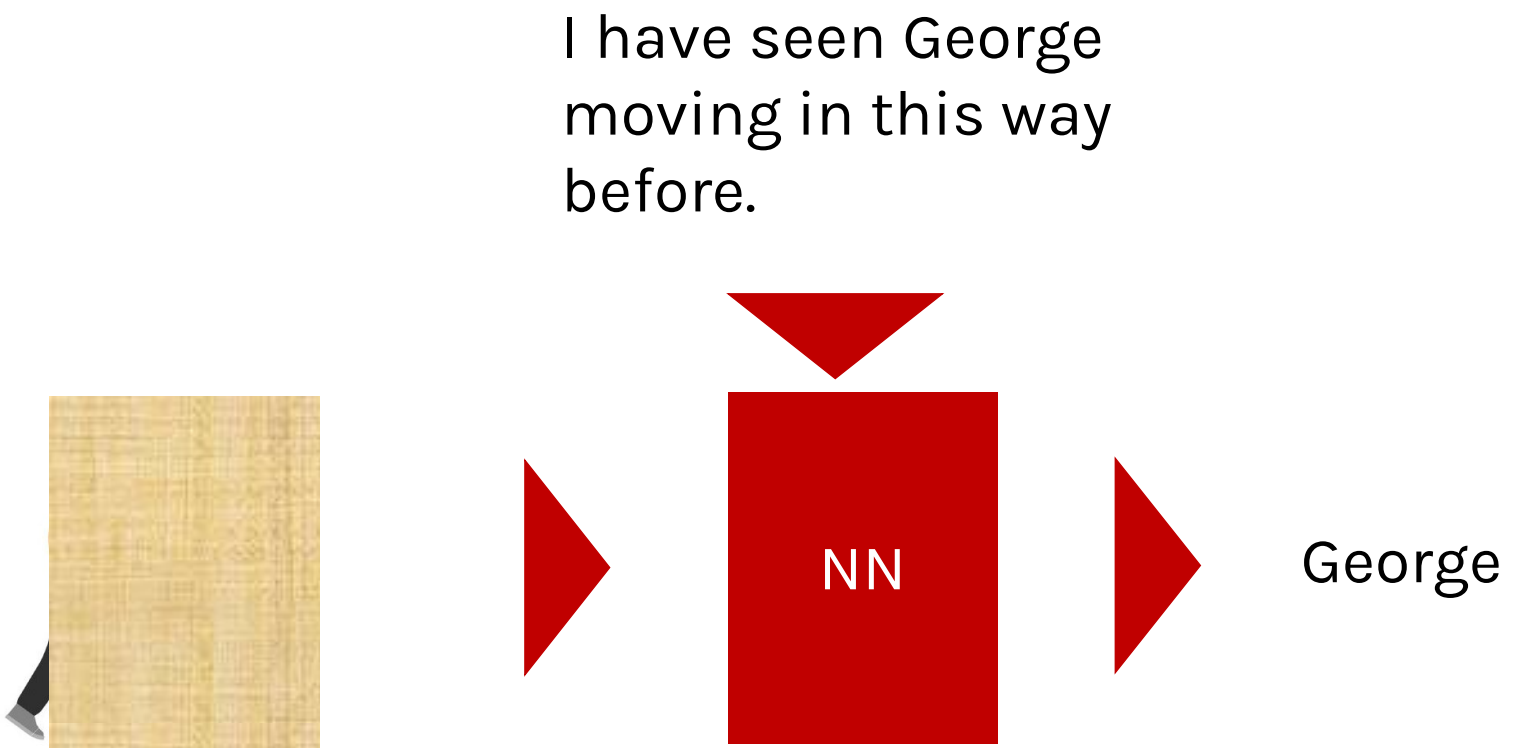
Learn from previous examples



Recurrent Neural Network (RNN)



Recurrent Neural Network (RNN)



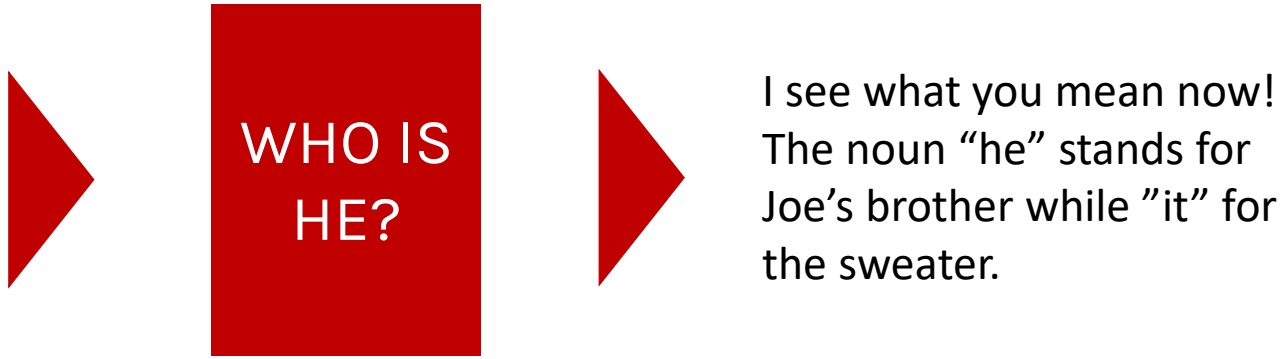
- RNNs recognize the data's sequential characteristics and use patterns to predict the next likely scenario.

Recurrent Neural Network (RNN)

- Our model requires context - or contextual information - to understand the subject (he) and the direct object (it) in the sentence.

RNN – Another Example with Text

- Hellen: Nice sweater Joe.
- Joe: Thanks, Hellen. It used to belong to my brother and ***he told me I could have it****.*

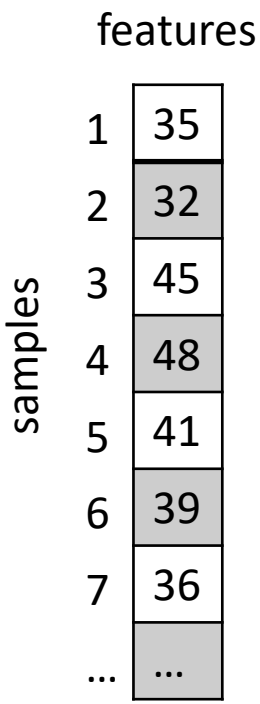


I see what you mean now! The noun “he” stands for Joe’s brother while ”it” for the sweater.

- After providing sequential information, the model understood the subject (Joe’s brother) and the direct object (sweater) in the sentence .

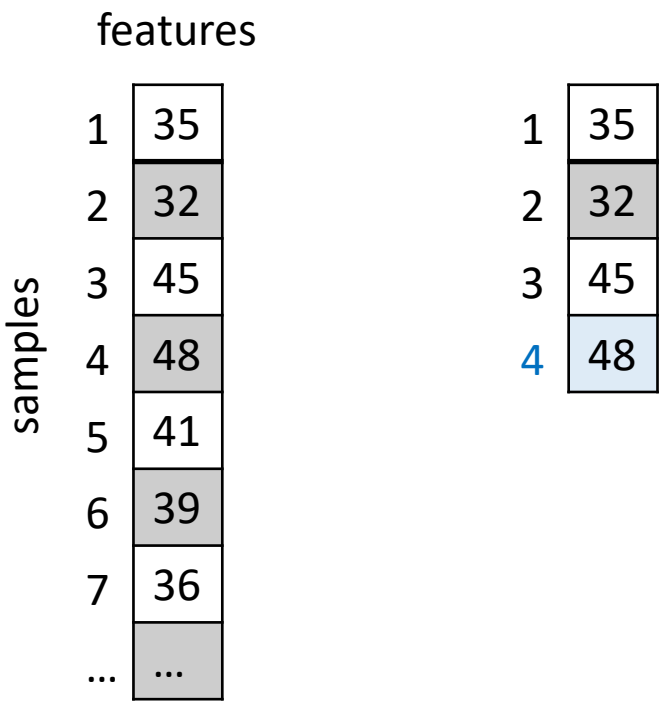
Sequences

- We want a machine learning model to understand sequences, not isolated samples.
- Can MLP do this?
- Assume we have a sequence of temperature measurements and we want to take 3 sequential measurements and predict the next one



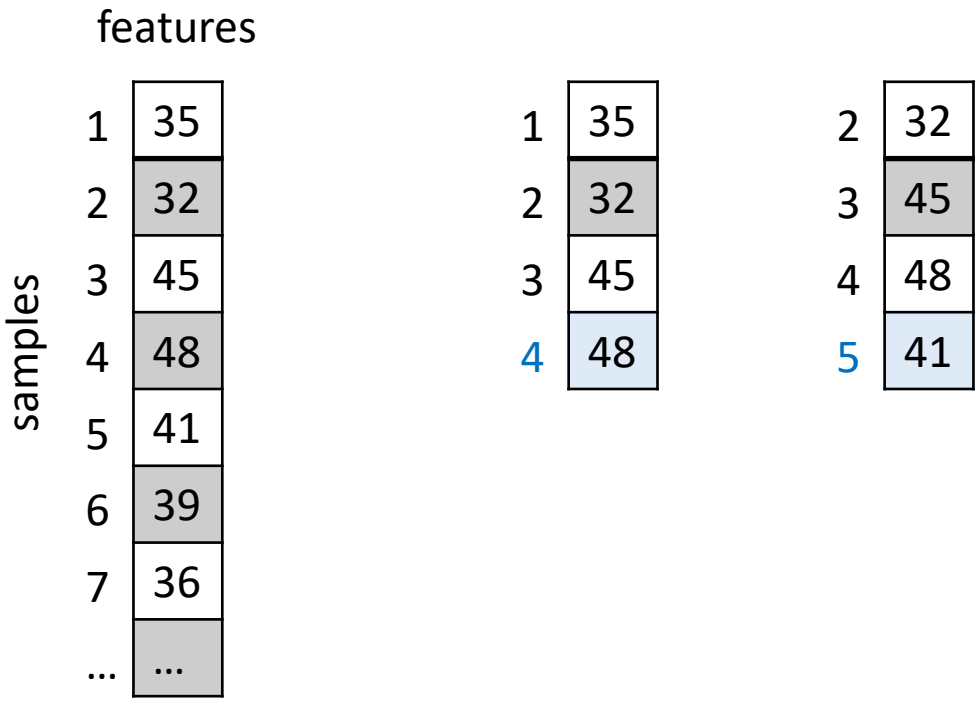
Sequences

- We want a machine learning model to understand sequences, not isolated samples.
- Can MLP do this?
- Assume we have a sequence of temperature measurements and we want to take 3 sequential measurements and predict the next one



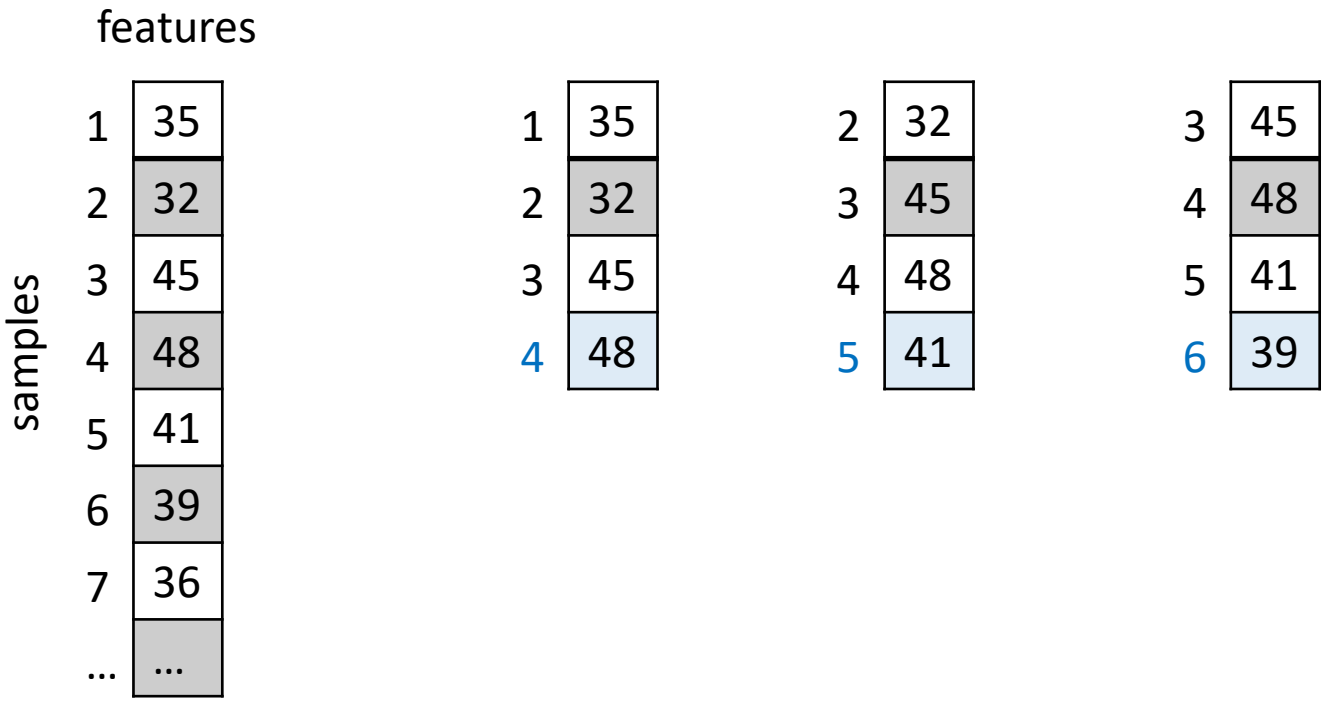
Sequences

- We want a machine learning model to understand sequences, not isolated samples.
- Can MLP do this?
- Assume we have a sequence of temperature measurements and we want to take 3 sequential measurements and predict the next one



Sequences

- We want a machine learning model to understand sequences, not isolated samples.
- Can MLP do this?
- Assume we have a sequence of temperature measurements and we want to take 3 sequential measurements and predict the next one

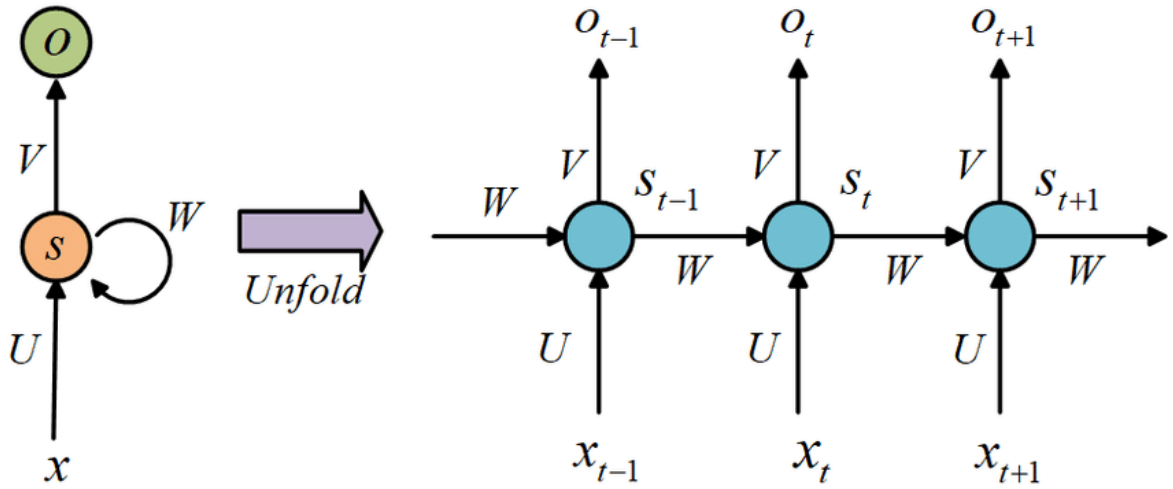


Why not CNNs or MLPs?

1. MLPs/CNNs require fixed input and output size
2. MLPs/CNNs can’t classify inputs in multiple places

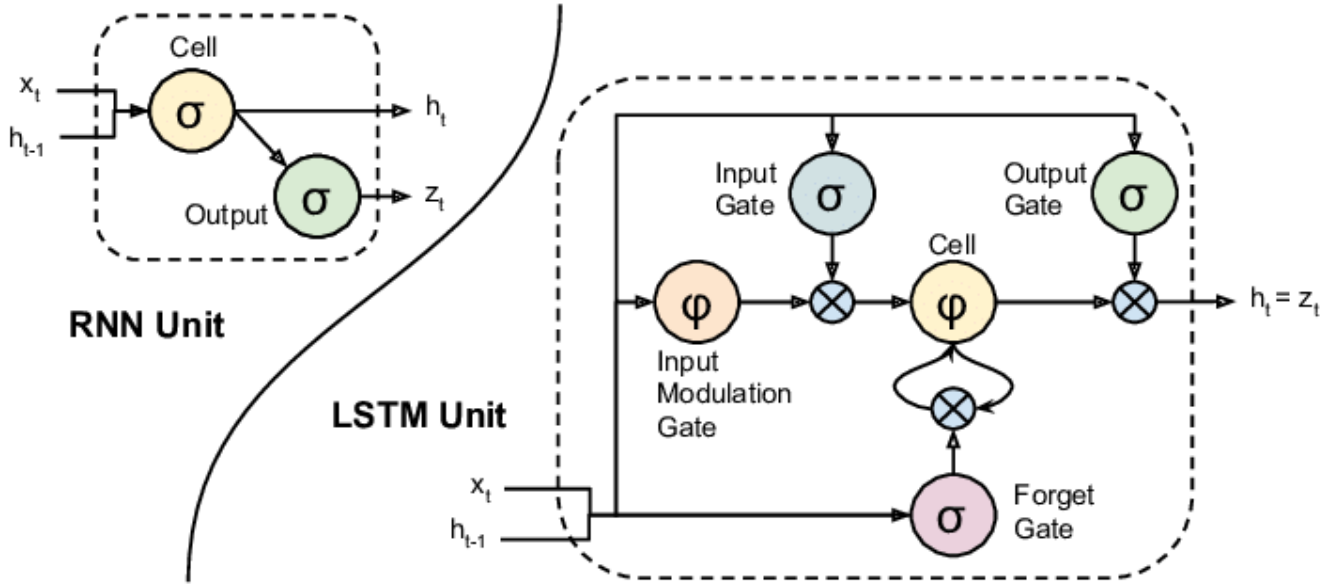
Main Concept of RNNs

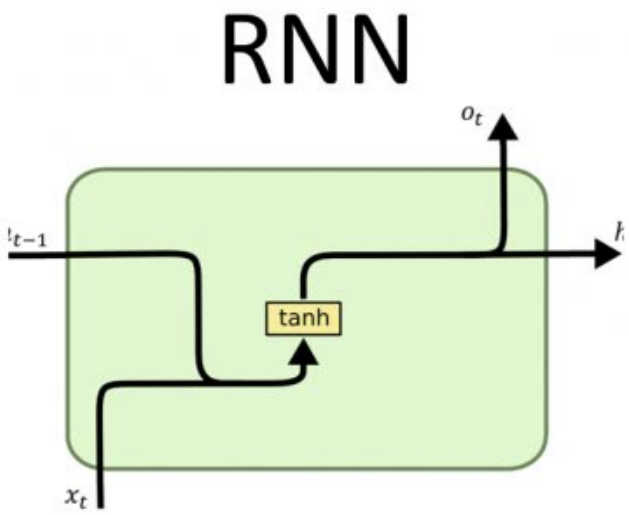
RNN cell

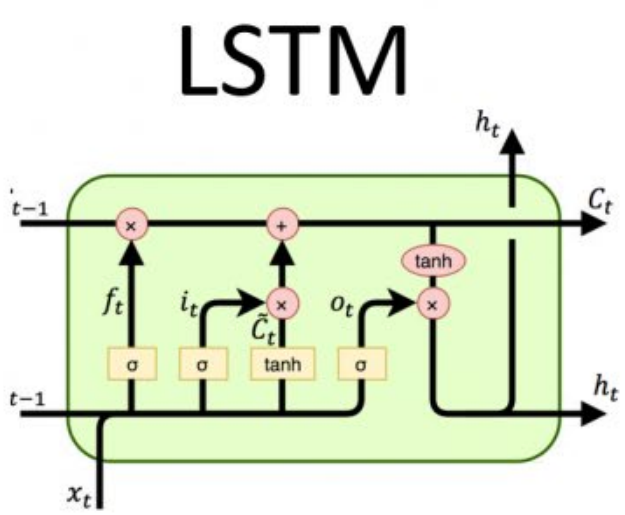


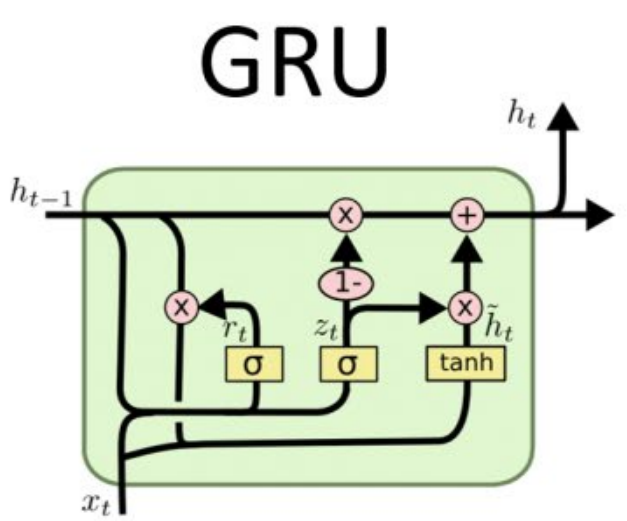
Time

RNN Cells


Cell
x_t
h_{t-1}
σ
h_t
Output
σ
z_t
RNN Unit
LSTM Unit
Input Gate
σ
Output Gate
σ
Cell
φ
φ
$h_t = z_t$
Input Modulation Gate
x_t
h_{t-1}
σ
Forget Gate

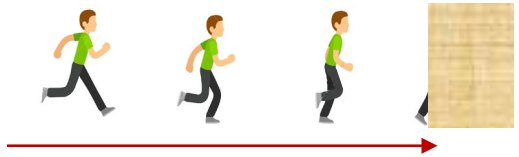

RNN
o_t
h_{t-1}
tanh
x_t

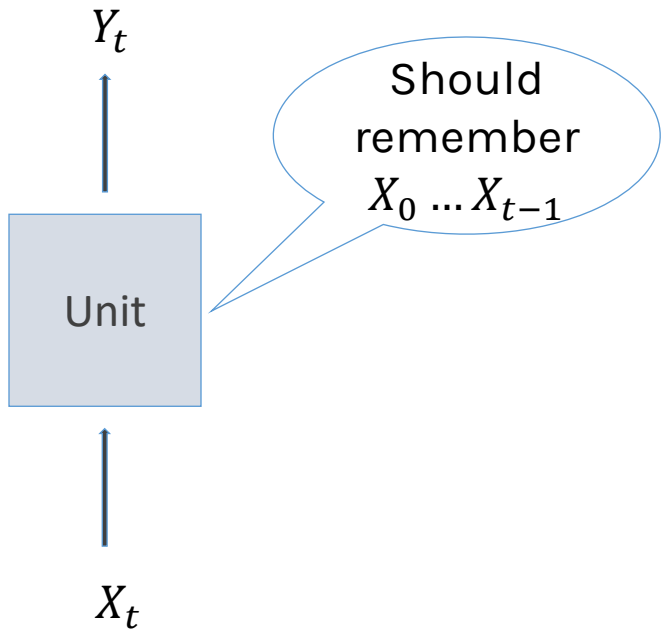

LSTM
h_t
C_t
h_{t-1}
f_t
i_t
$\tilde{C}_t$
o_t
tanh
σ
σ
tanh
σ
h_{t-1}
h_t
x_t


GRU
h_t
h_{t-1}
1-
r_t
z_t
$\tilde{h}_t$
σ
σ
tanh
x_t

Memory

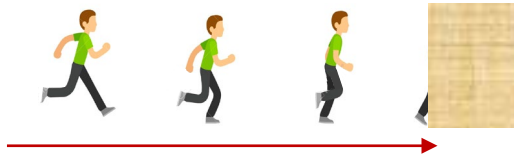
- Somehow the computational unit should remember what it has seen before.

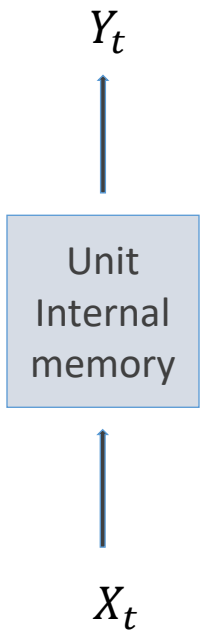




Memory

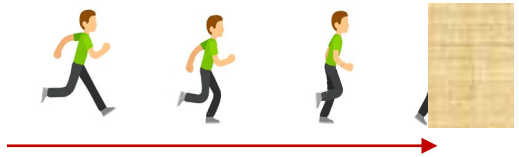
- Somehow the computational unit should remember what it has seen before.

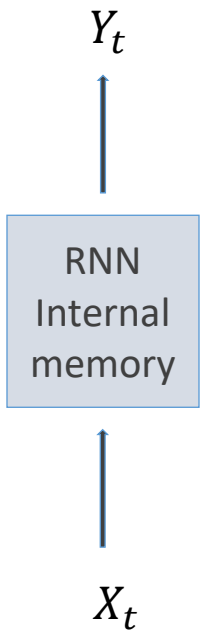




Memory

- Somehow the computational unit should remember what it has seen before.
- We'll call the information the unit's **state**.



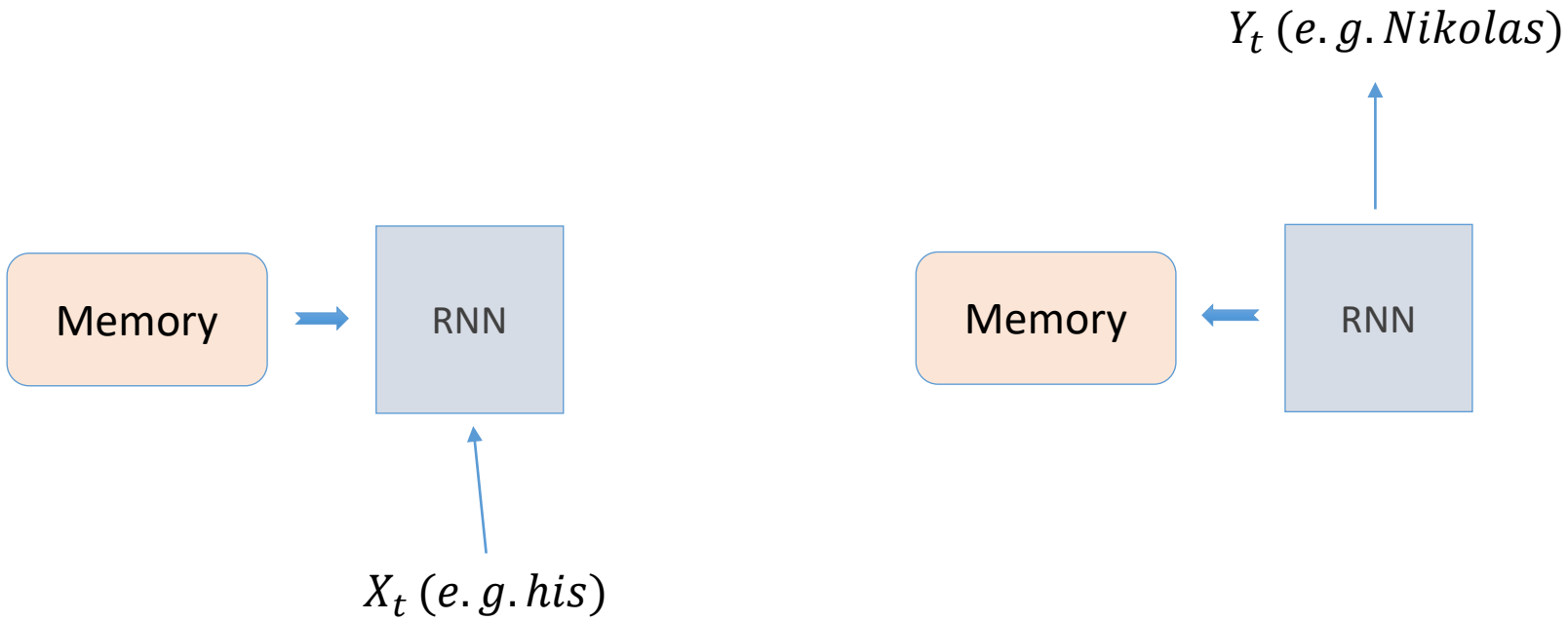


Memory

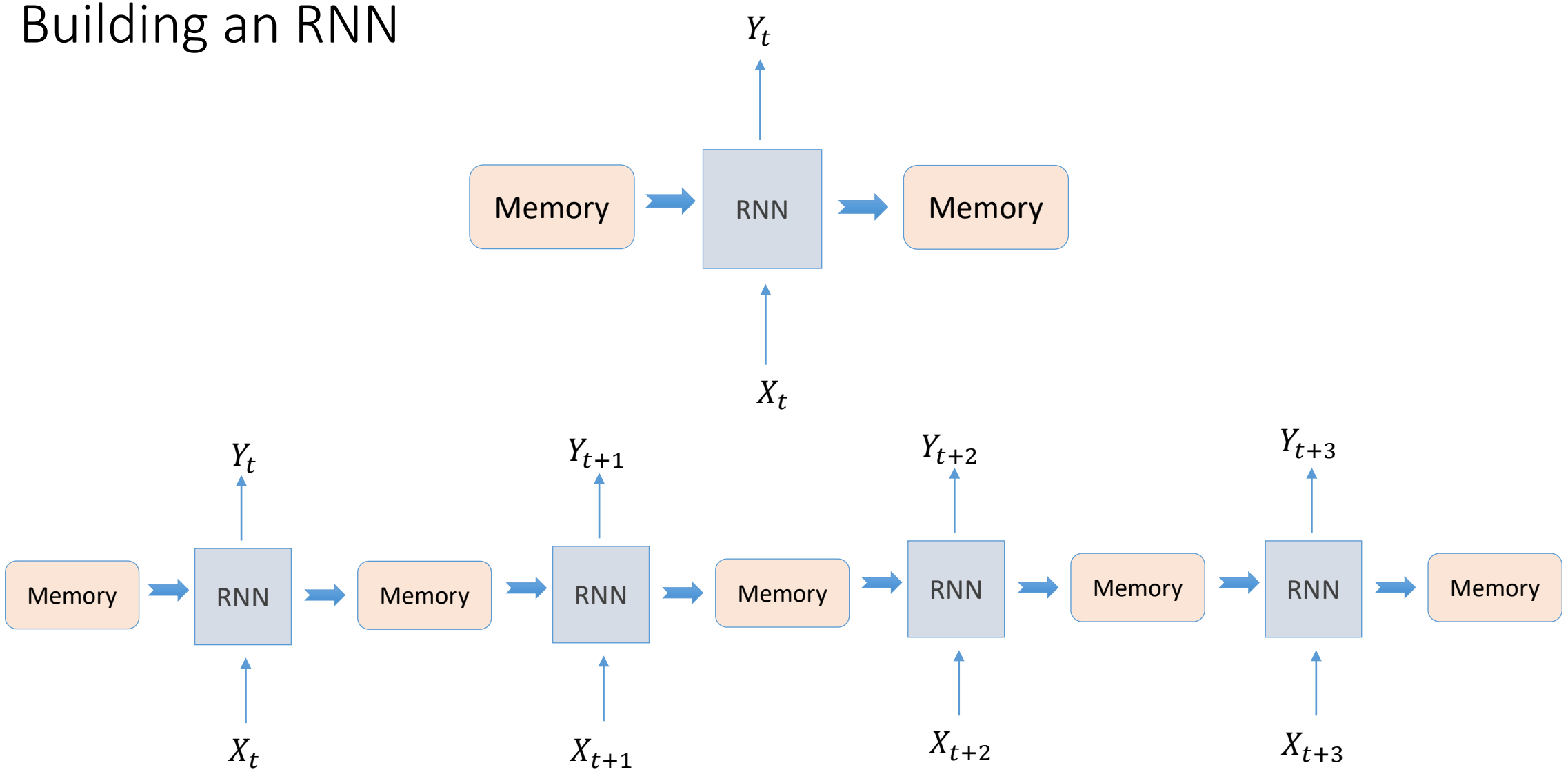
- In neural networks, once training is over, the weights do not change. This means that the network is done learning and done changing.
- Then, we feed in values, and it simply applies the operations that make up the network, using the values it has learned.
- But the RNN units are able to remember new information after training has completed.
- **That is, they're able to keep changing after training is over.**

Memory

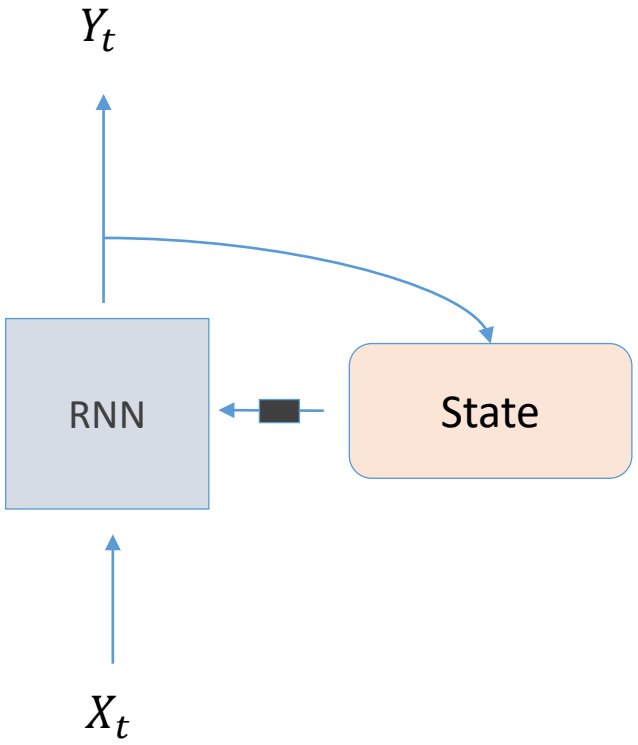
- Question: How can we do this? How can build a unit that remembers the past?
- The memory or **state** can be written to a file but in RNNs, we keep it inside the recurrent unit.
- In an array or in a vector!

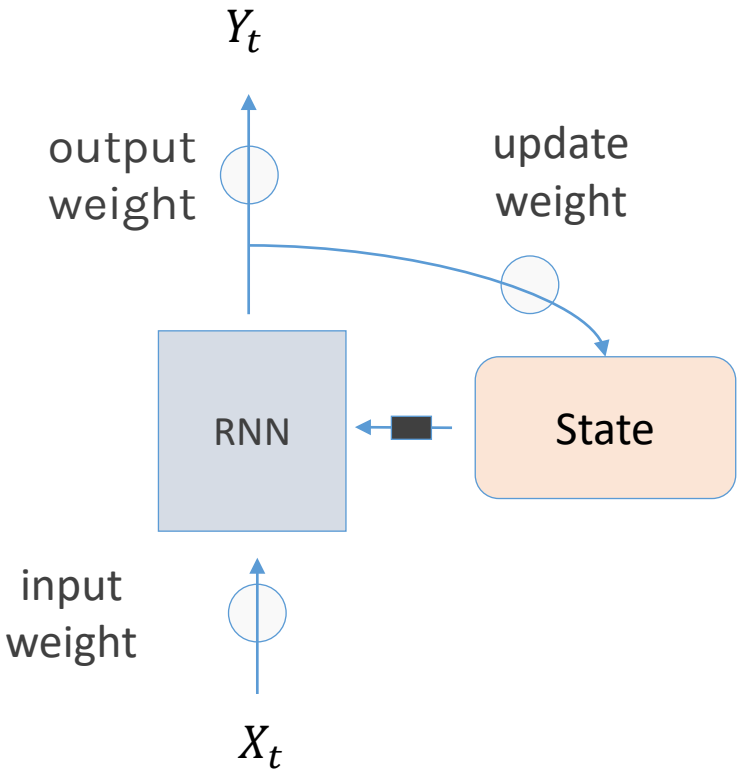


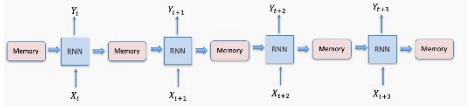
Building an RNN



Structure of an RNN cell



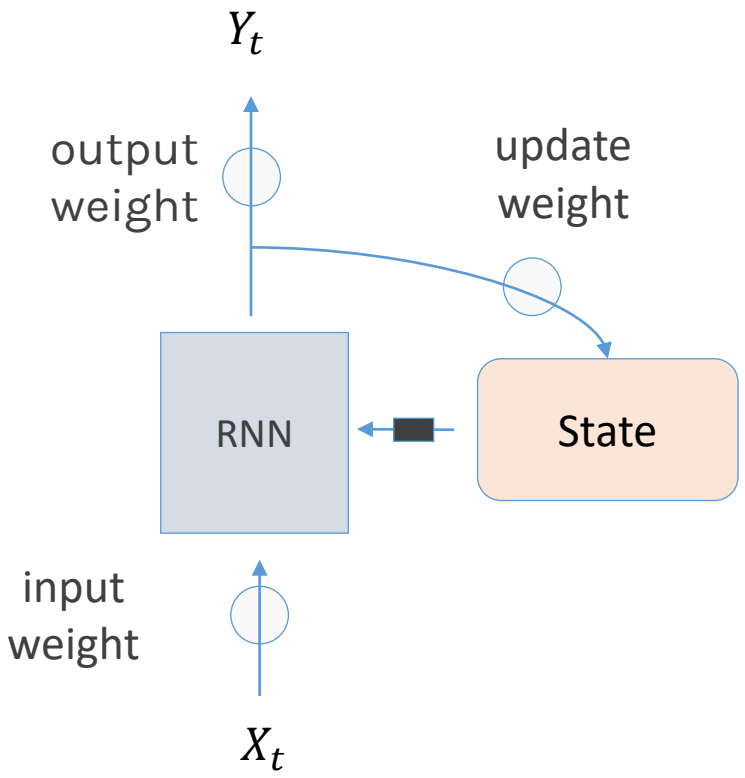




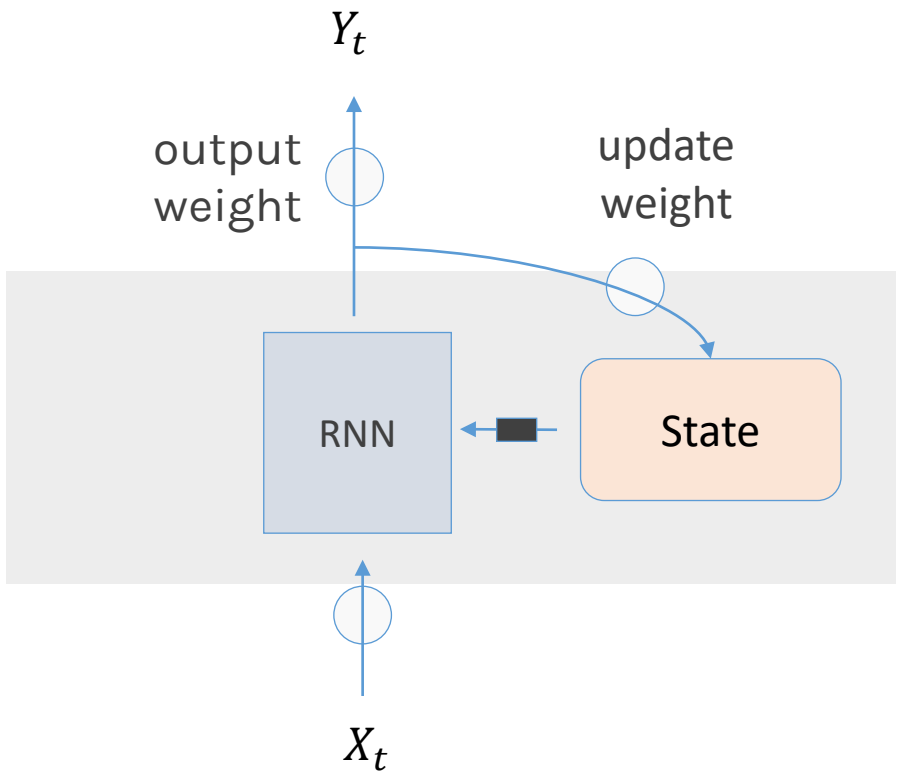
Backprop Through Time

- For each input, unfold network for the sequence length T
- Back-propagation: apply forward and backward pass on unfolded network
- Memory cost: $O(T)$

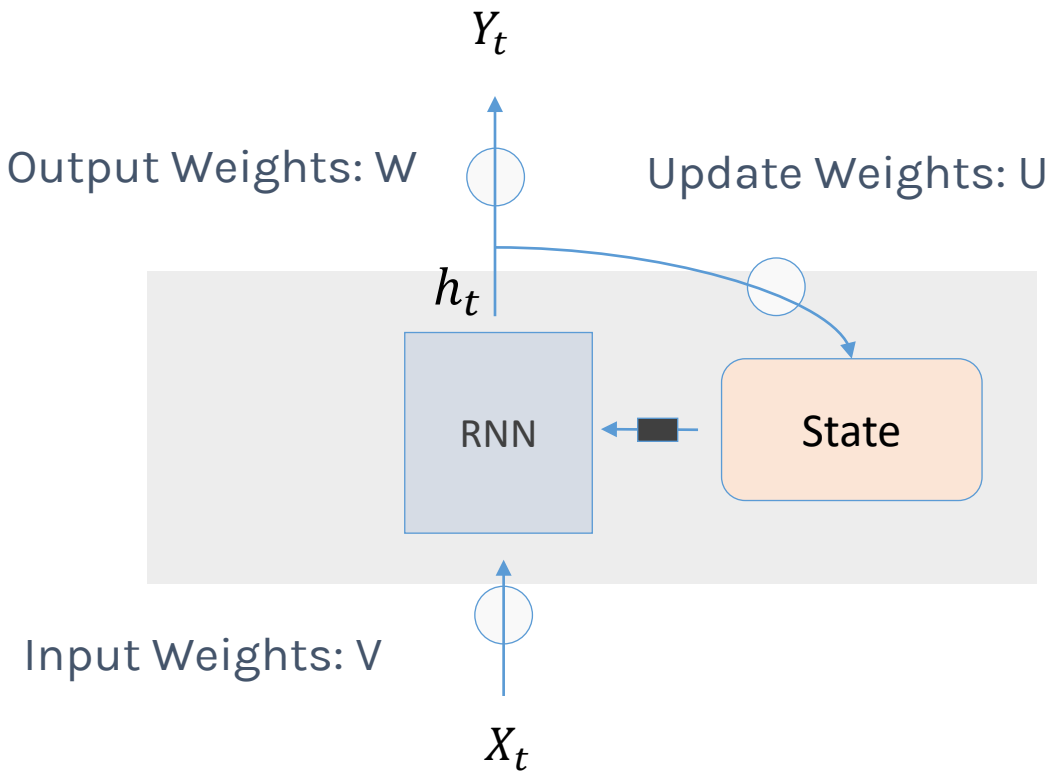
Backprop Through Time



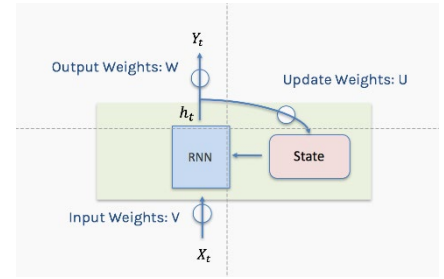
Backprop Through Time

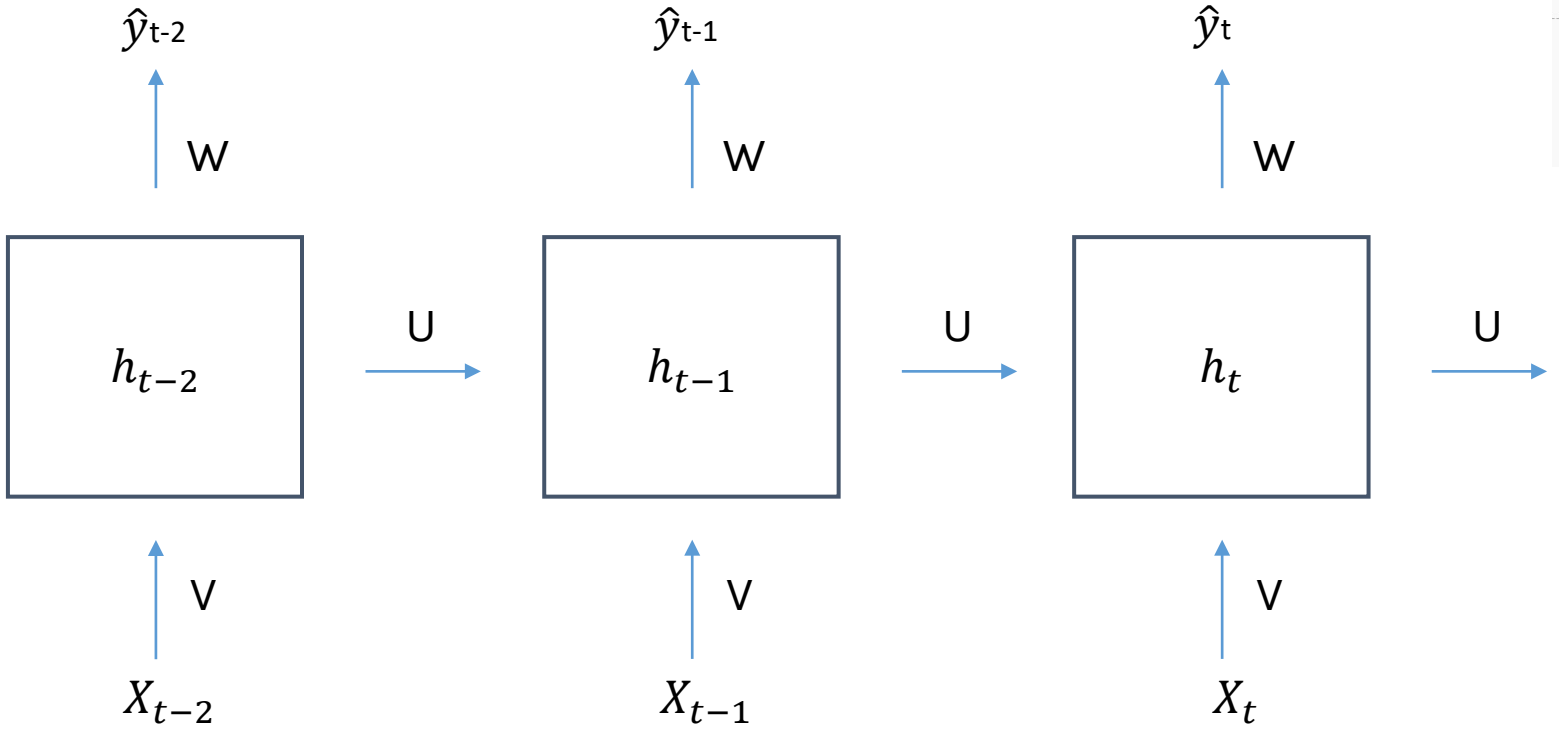


Backprop Through Time

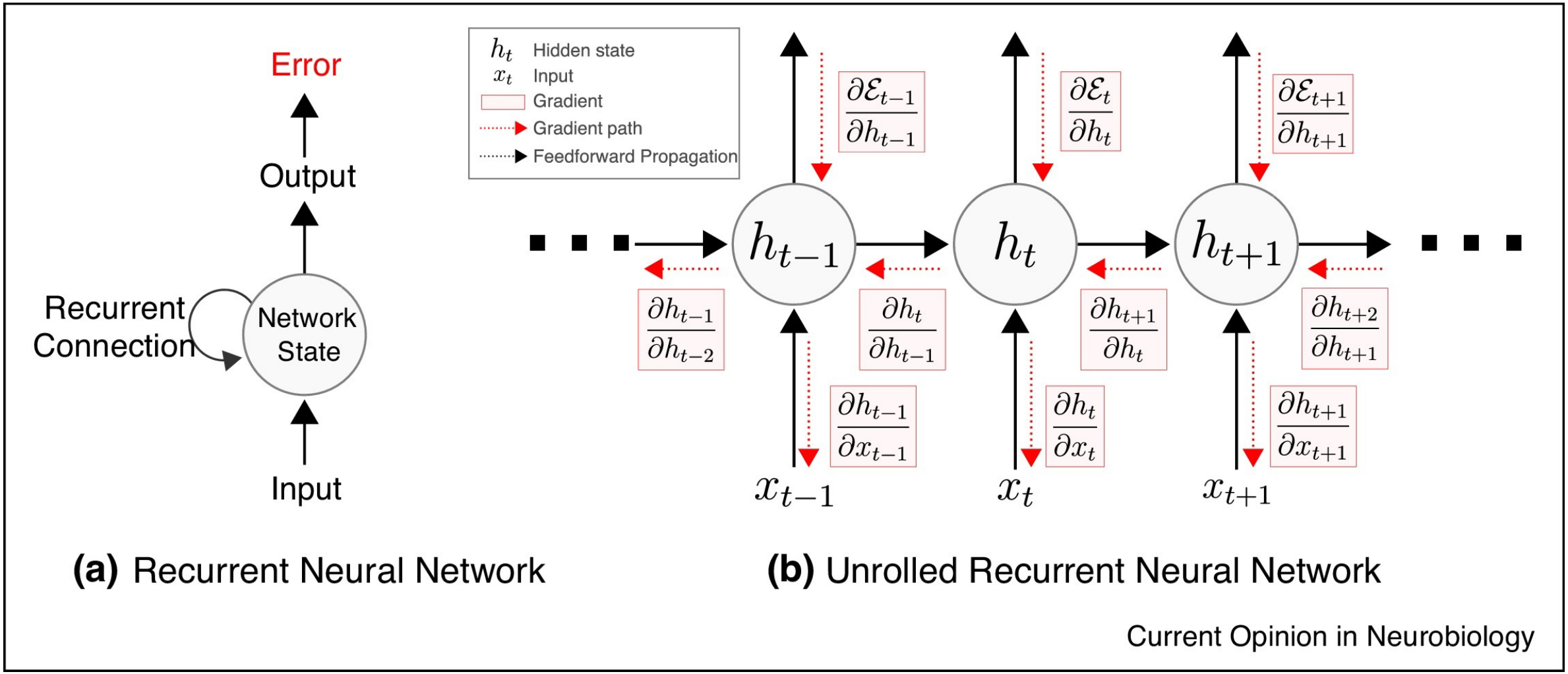


Backprop Through Time





You have two activation functions g_h which serves as the activation for the hidden state and g_y which is the activation of the output. In the example shown before g_y was the identity.



DOI:10.1016/j.conb.2019.01.011

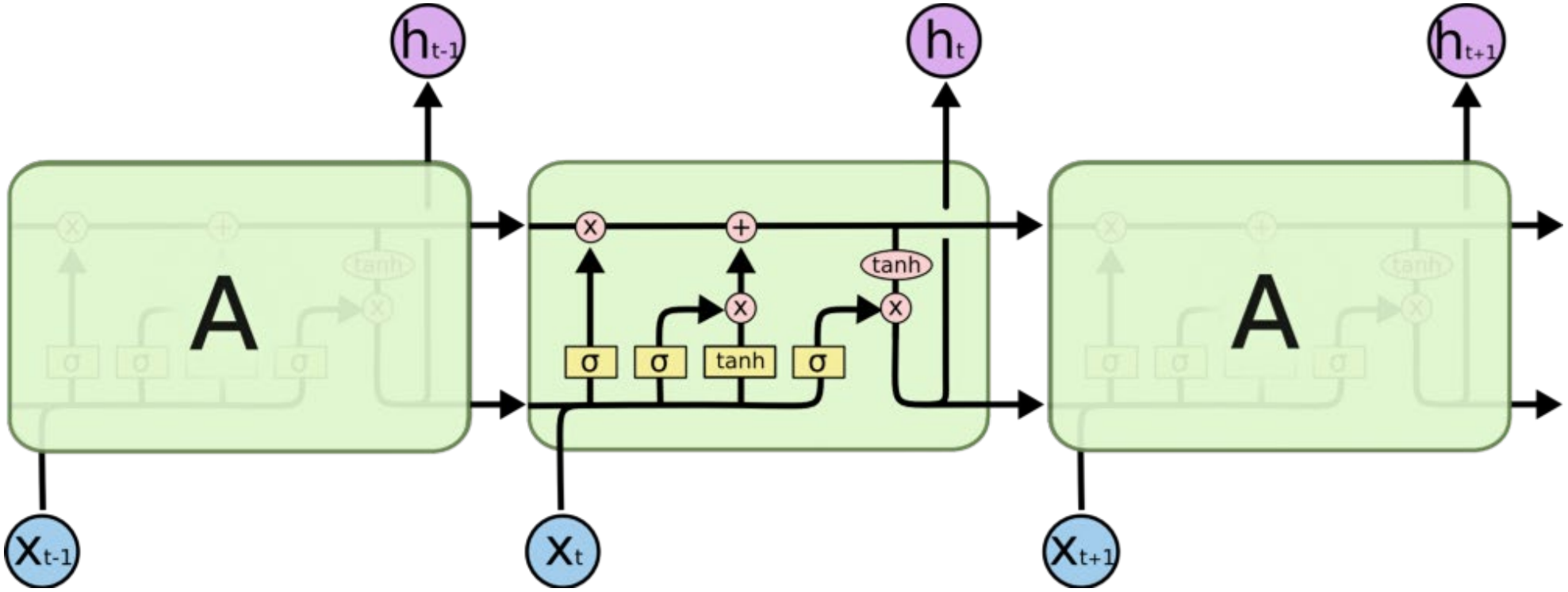
Long-term Dependencies

- Unfolded networks can be very deep
- Long-term interactions are given exponentially smaller weights than small-term interactions
- Gradients tend to either *vanish* or *explode*

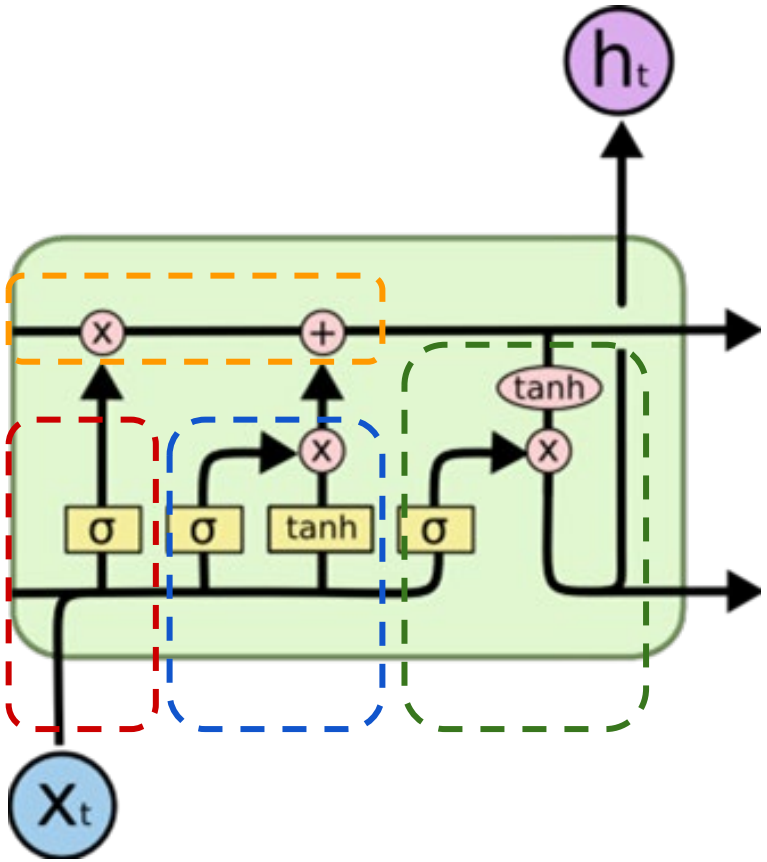
Long Short-Term Memory

- Handles long-term dependencies
- Leaky units where weight on self-loop α is context-dependent
- Allow network to decide whether to accumulate or forget past info

LSTM: Long short term memory



LSTM big picture …



Cell State

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

Output Gate

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

Forget Gate

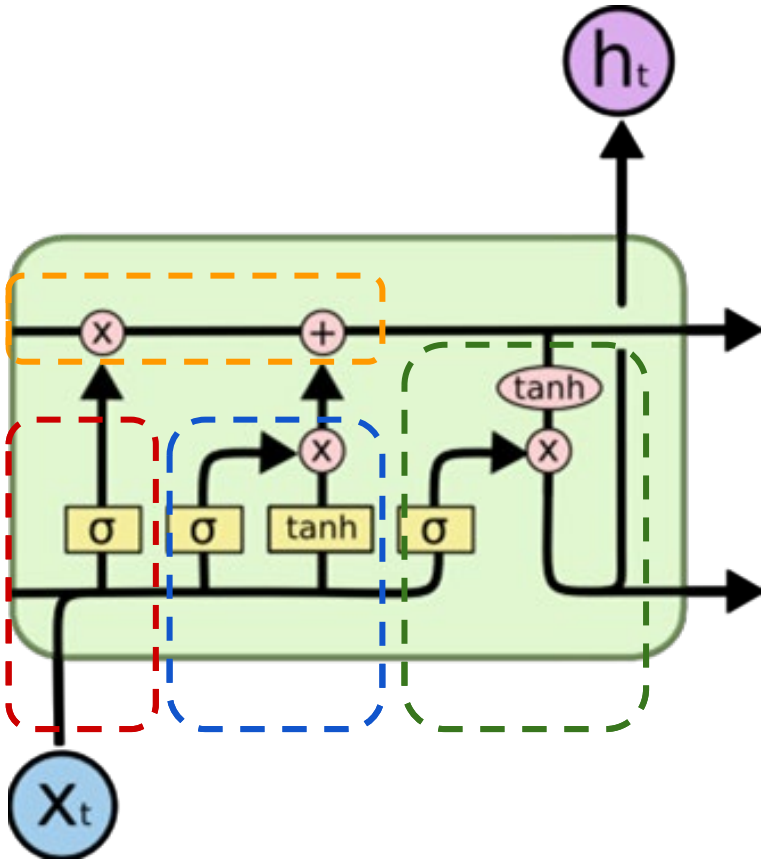
$$f_t = \sigma(W_f \cdot [h_{t-1}, x_{t-1}] + b_f)$$

Input Gate

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_f)$$

LSTM big picture …

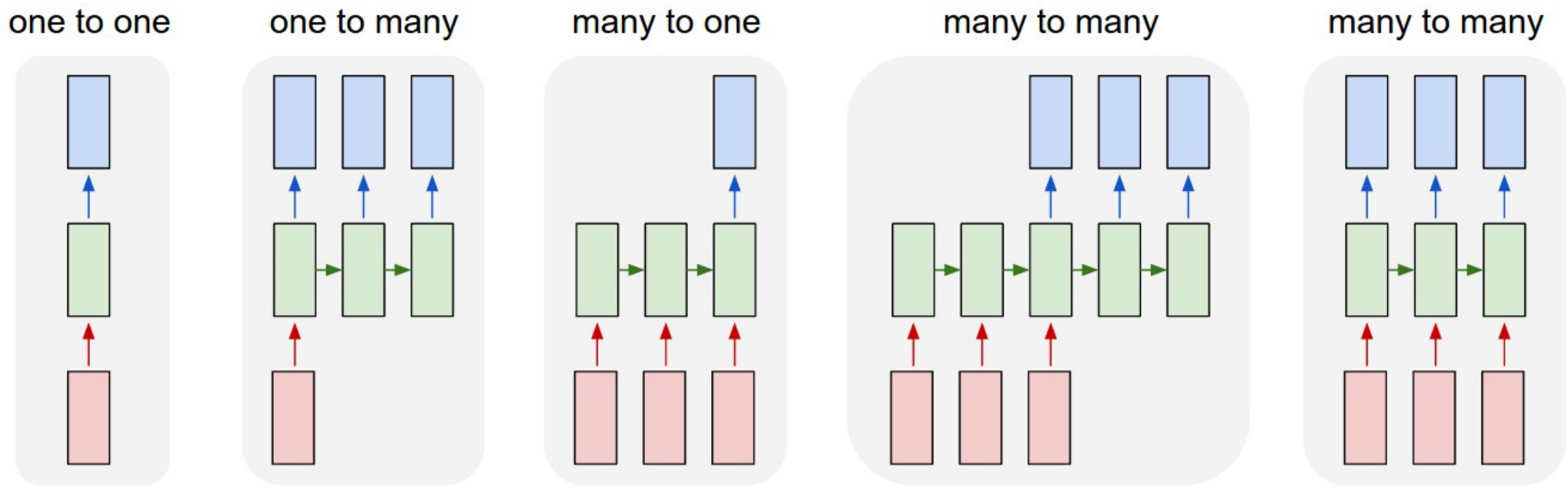


1. LSTM are recurrent neural network with a cell and a hidden state, boths of these are updated in each step and can be thought as memories.

2. Cell states work as a long term memory and the updates depends on the relation between the hidden state in t -1 and the input.

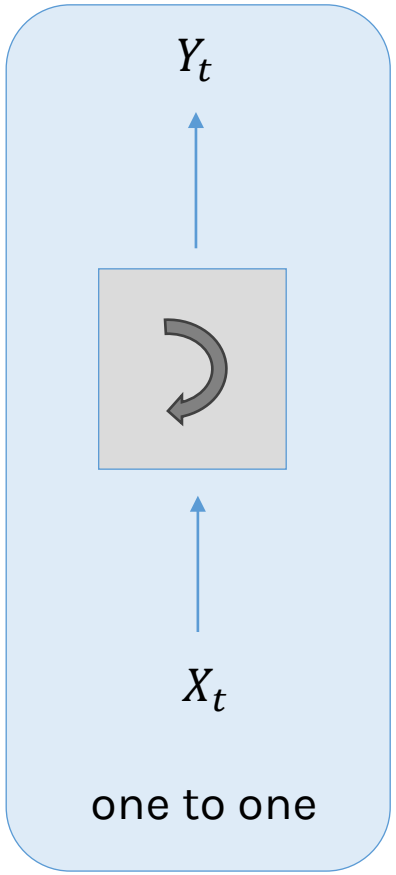
3. The hidden state of the next step is a transformation of the cell state and the output (which is the section that is in general used to calculate our loss, ie information that we want in a short memory).

What makes Recurrent Networks so special?



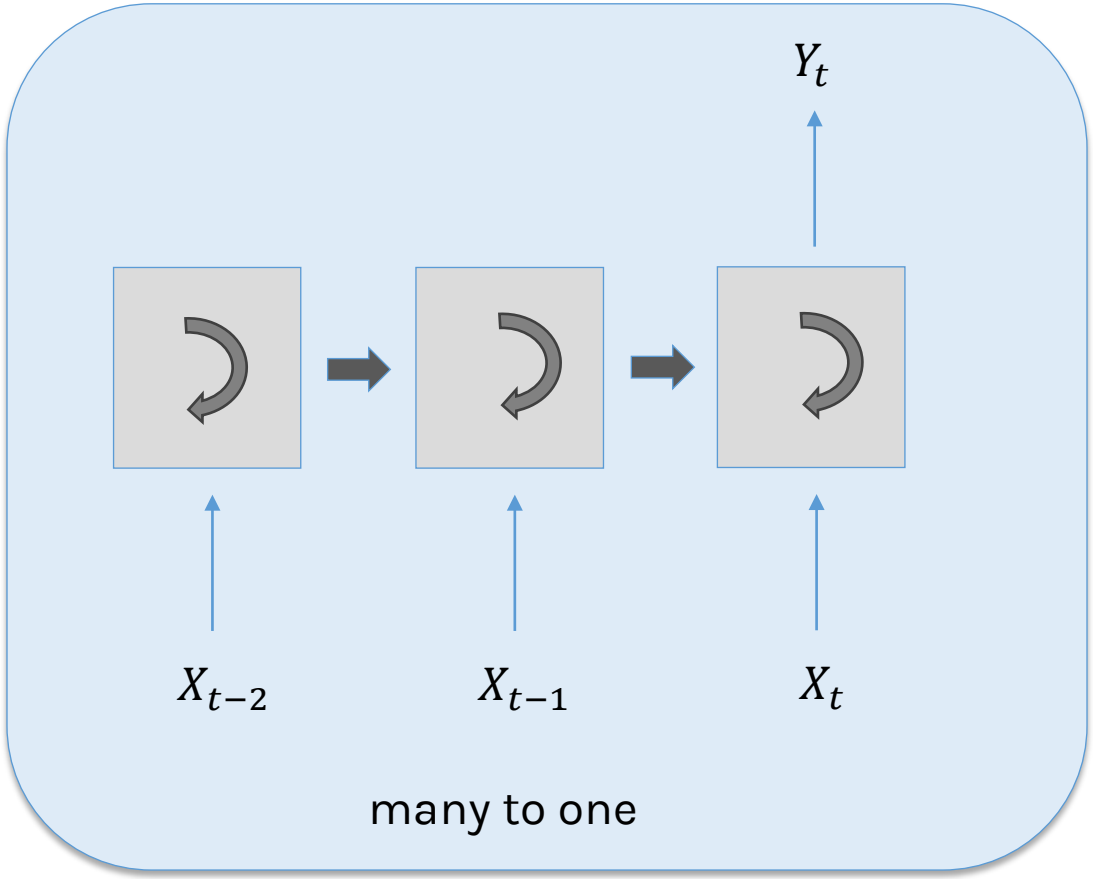
From: http://karpathy.github.io/

RNN Structures



- The **one to one** structure is useless.
- It takes a single input and it produces a single output.
- Not useful because the RNN cell is making little use of its unique ability to remember things about its input sequence

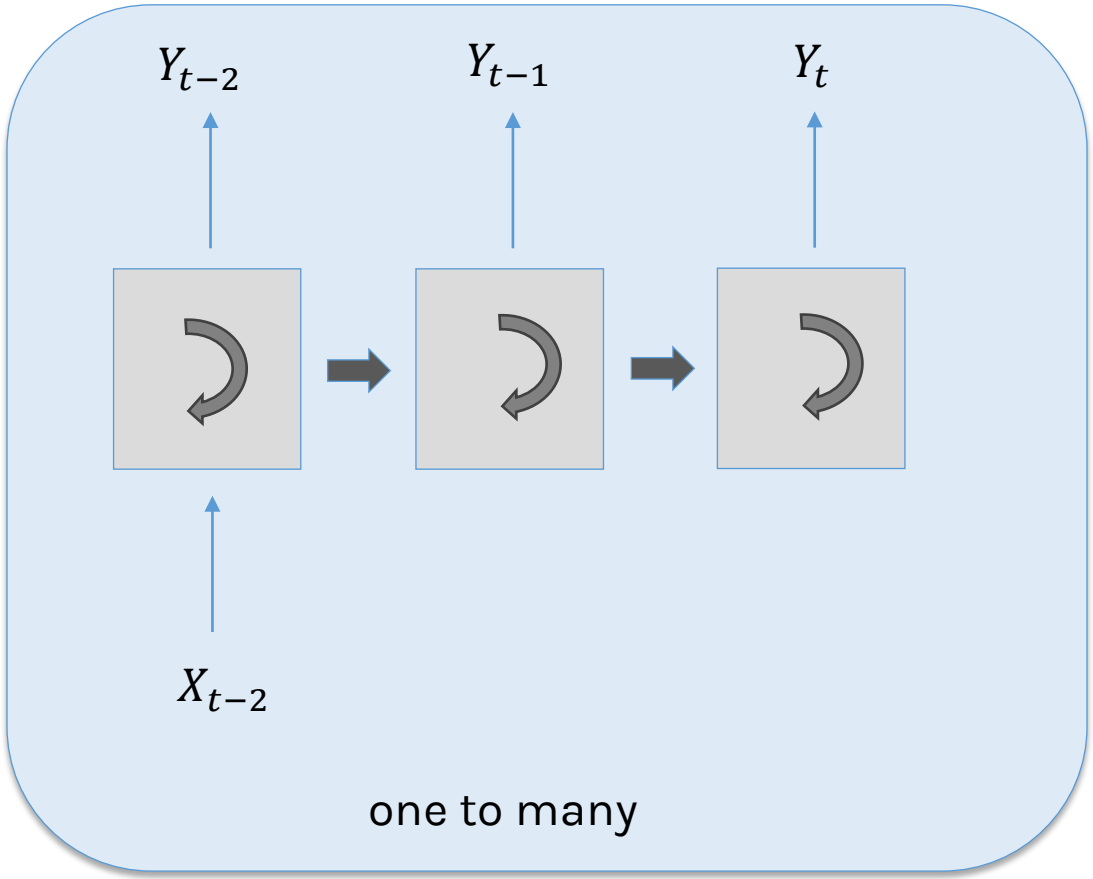
RNN Structures (cont)



The **many to one** structure reads in a sequence and gives us back a single value.

Example: Sentiment analysis, where the network is given a piece of text and then reports on some quality inherent in the writing. A common example is to look at a movie review and determine if it was positive or negative.

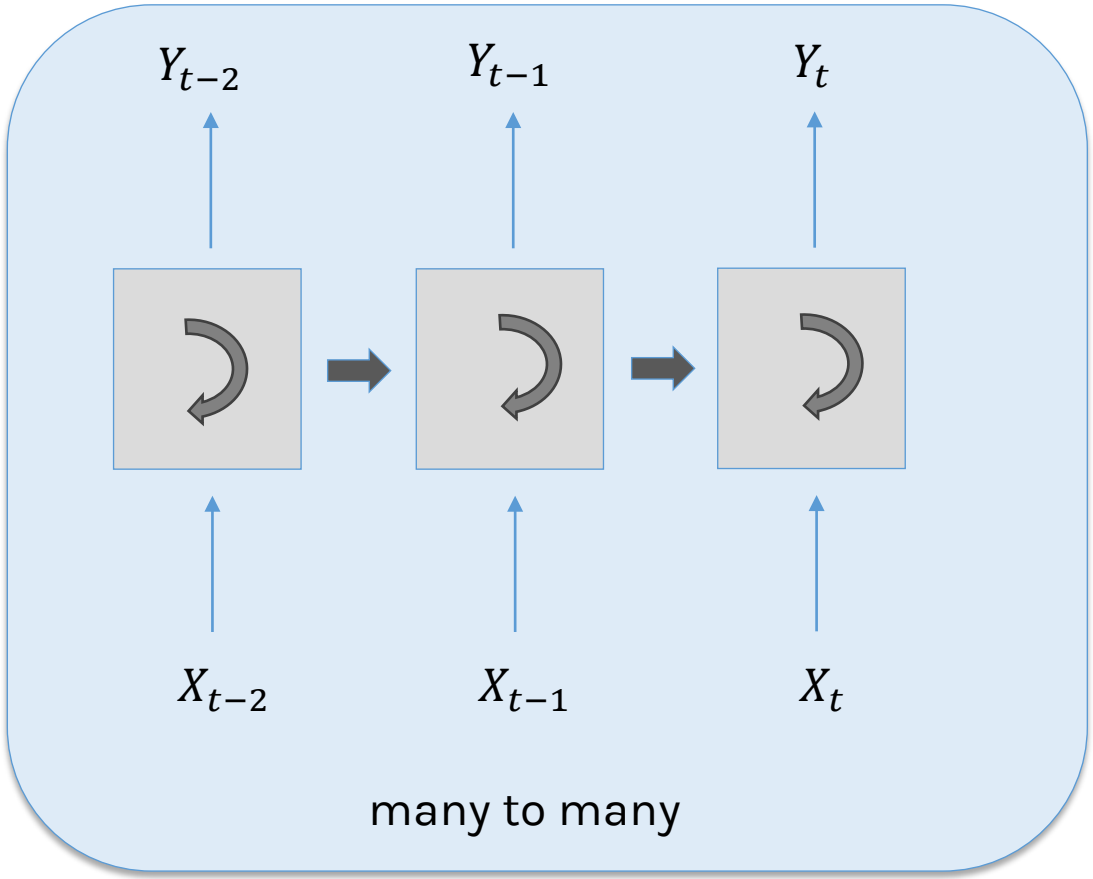
RNN Structures (cont)



The **one to many** takes in a single piece of data and produces a sequence.
For example we give it the starting note for a song, and the network produces the rest of the melody for us.

Sequence generative models

RNN Structures (cont)



The **many to many** structures are in some ways the most interesting. used for machine translation.

Seq2seq models

Machine Translation

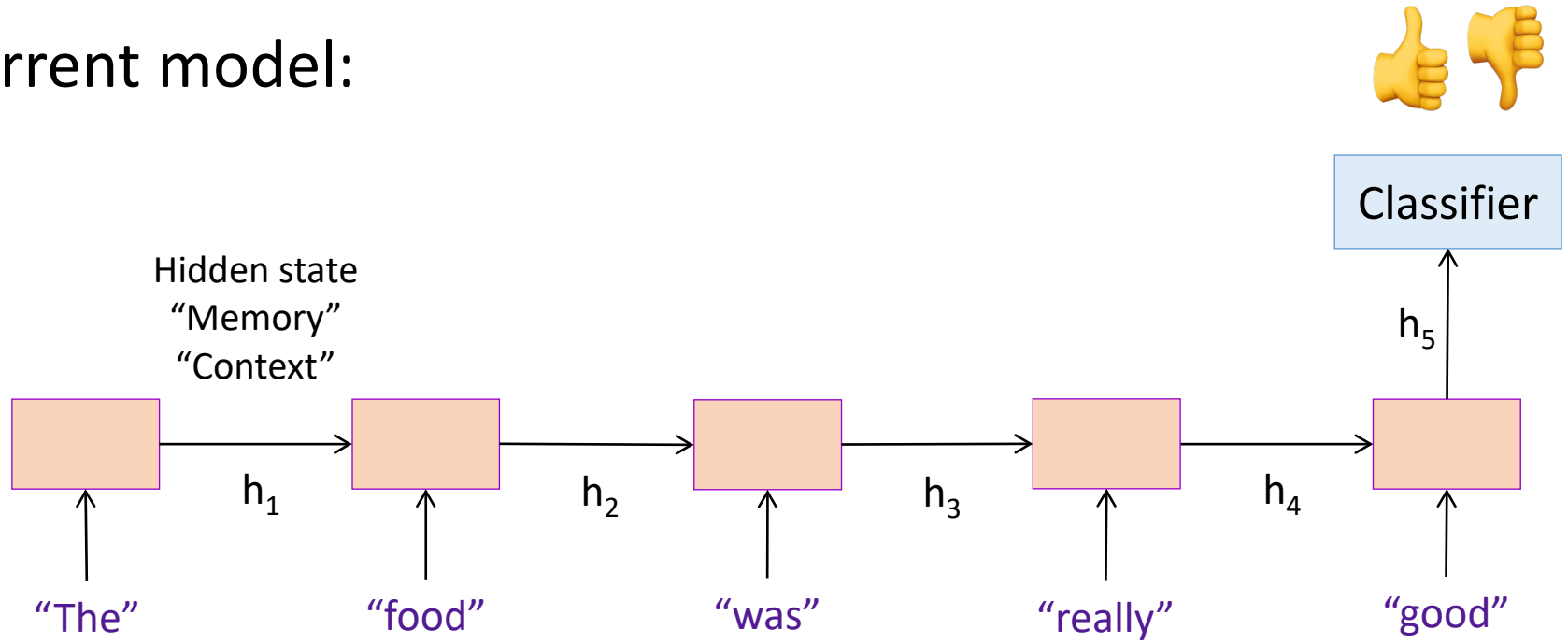
…

Ref

- [Chen17b] Qiming Chen, Ren Wu, “CNN Is All You Need”, arXiv 1712.09662, 2017. https://arxiv.org/abs/1712.09662
- [Chu17] Hang Chu, Raquel Urtasun, Sanja Fidler, “Song From PI: A Musically Plausible Network for Pop Music Generation”, arXiv preprint, 2017. https://arxiv.org/abs/1611.03477
- [Johnson17] Daniel Johnson, “Composing Music with Recurrent Neural Networks”, Heahedria, 2017. http://www.hexahedria.com/2015/08/03/ composing-music-with-recurrent-neural-networks/
- [Deutsch16b] Max Deutsch, “Silicon Valley: A New Episode Written by AI”, Deep Writing blog post, 2017. https://medium.com/deep-writing/ silicon-valley-a-new-episode-written-by-ai-a8f832645bc2
- [Fan16] Bo Fan, Lijuan Wang, Frank K. Soong, Lei Xie “Photo-Real Talking Head with Deep Bidirectional LSTM”, Multimedia Tools and Applications, 75(9), 2016. https://www.microsoft.com/en-us/research/wp-content/uploads/2015/04/icassp2015_fanbo_1009.pdf

Example 1: Sentiment classification

- Recurrent model:



Example 2: Text generation

Example 2: Text generation

- Sample from the distribution of a given text corpus (also known as language modeling)
- Can be done one character or one word at a time:

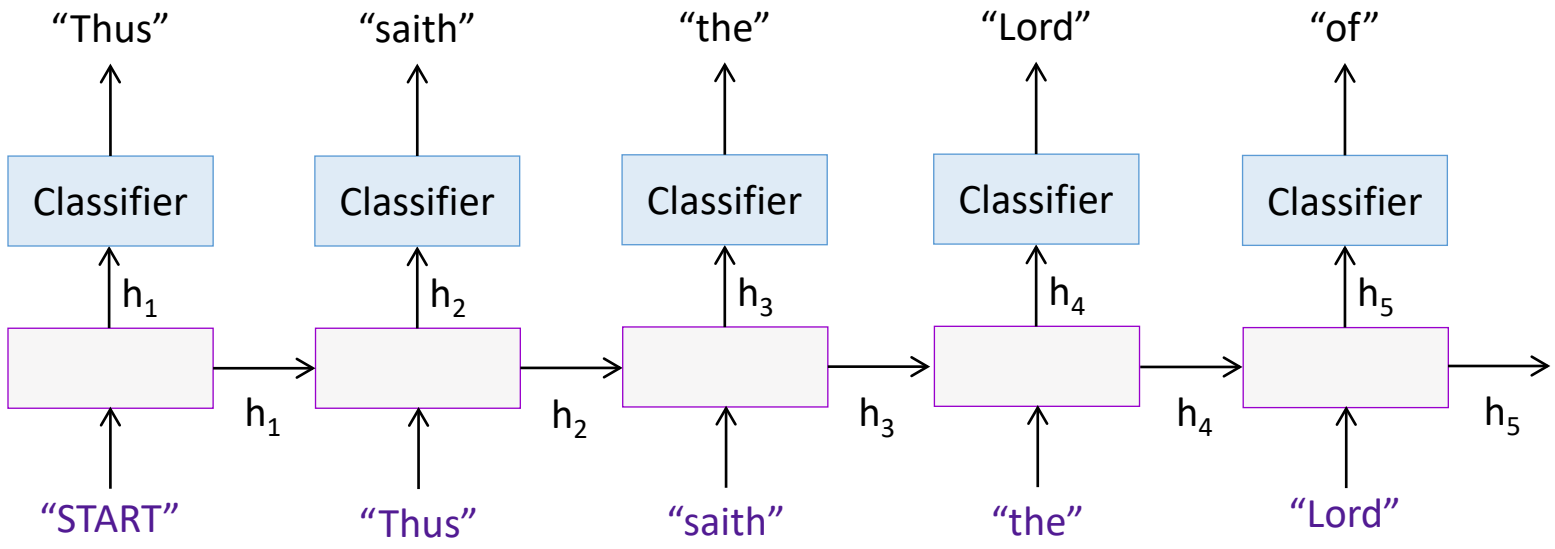


Image source

Example 3: Image caption generation

A cat sitting on a suitcase on the floor

A cat is sitting on a tree branch

A dog is running in the grass with a frisbee

A white teddy bear sitting in the grass

Two people walking on the beach with surfboards

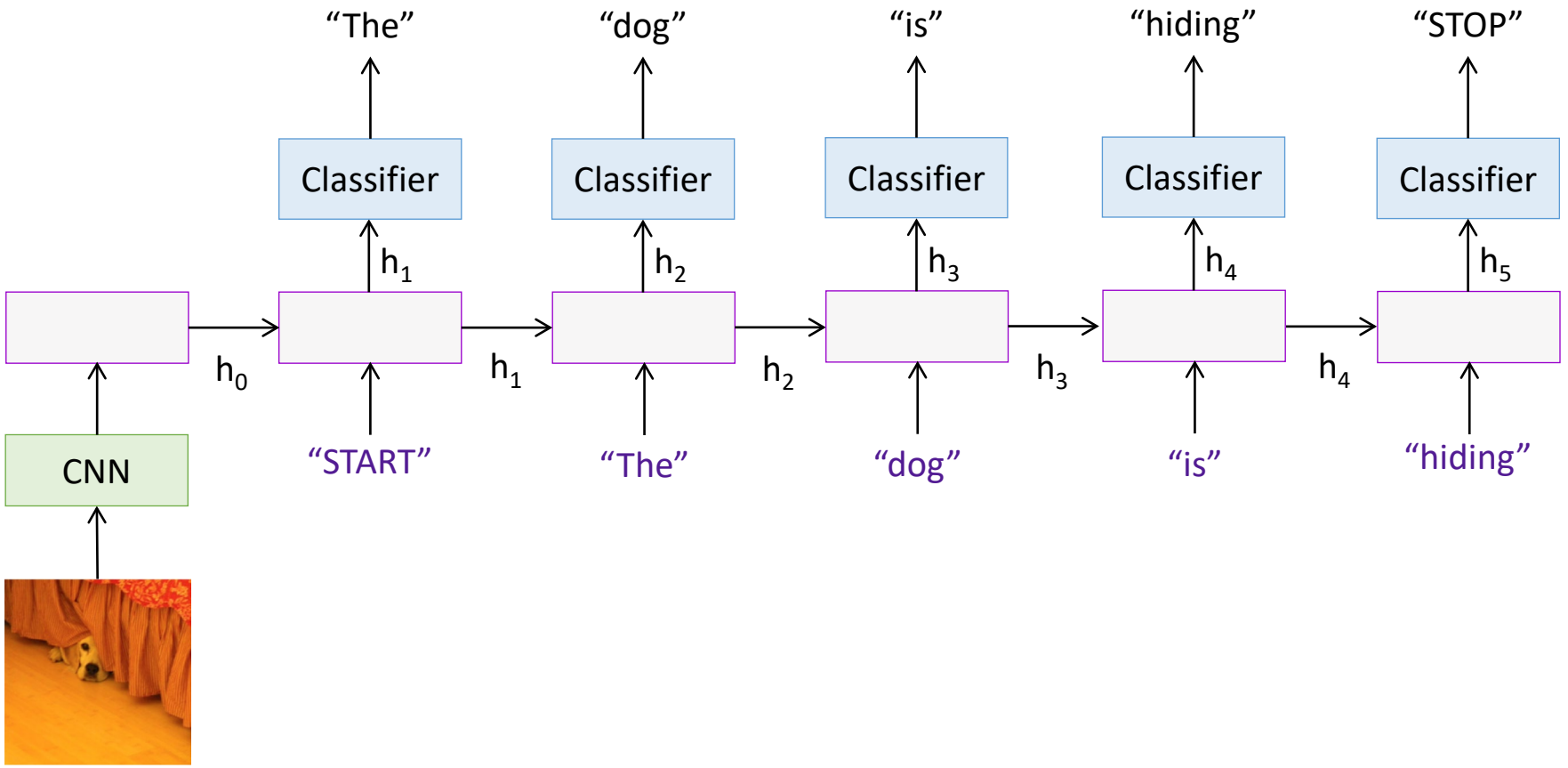
A tennis player in action on the court

Two giraffes standing in a grassy field

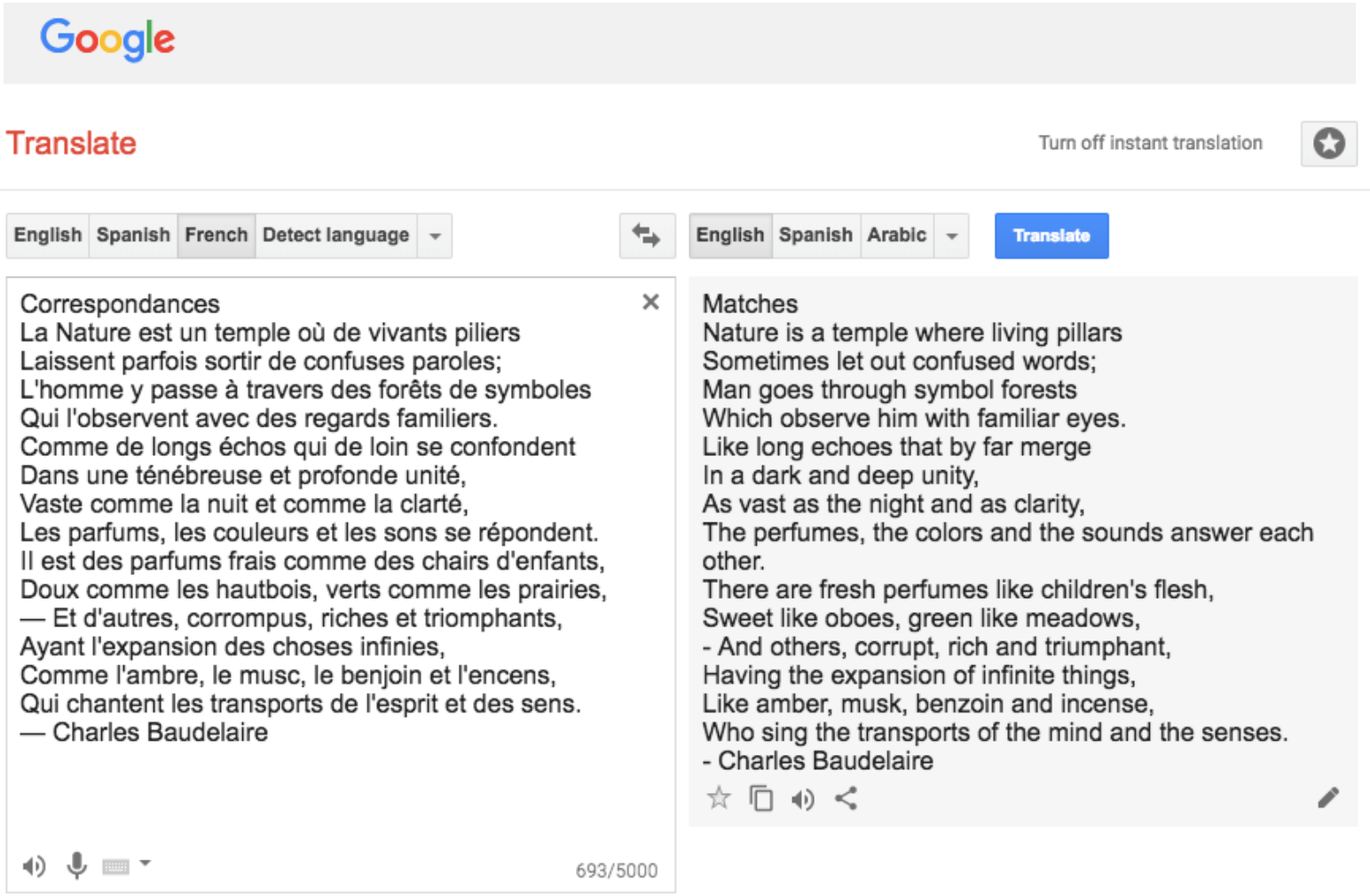
A man riding a dirt bike on a dirt track

Source: J. Johnson
Captions generated using neuraltalk2

Example 3: Image caption generation



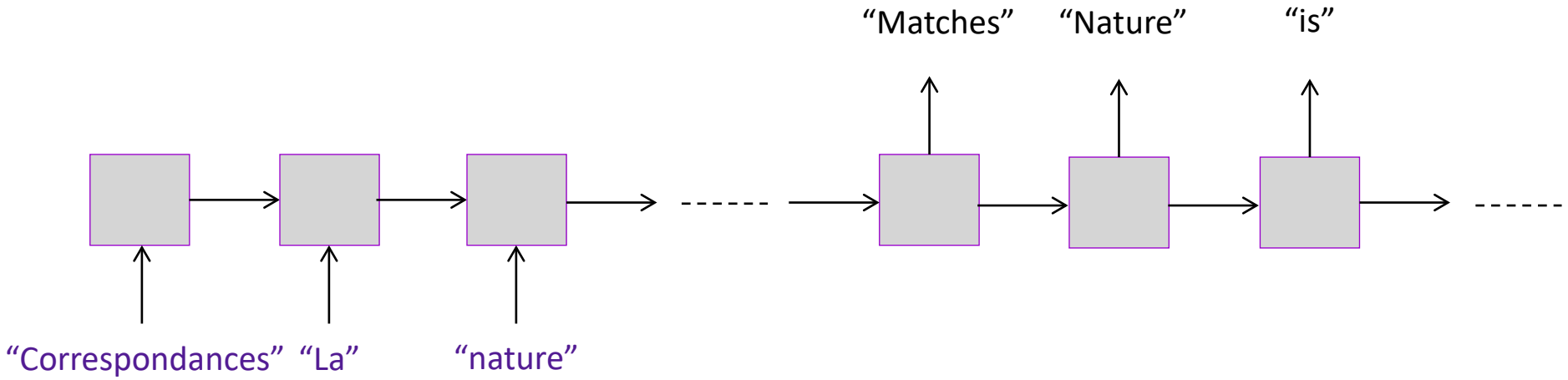
Example 4: Machine translation



https://translate.google.com/

Example 4: Machine translation

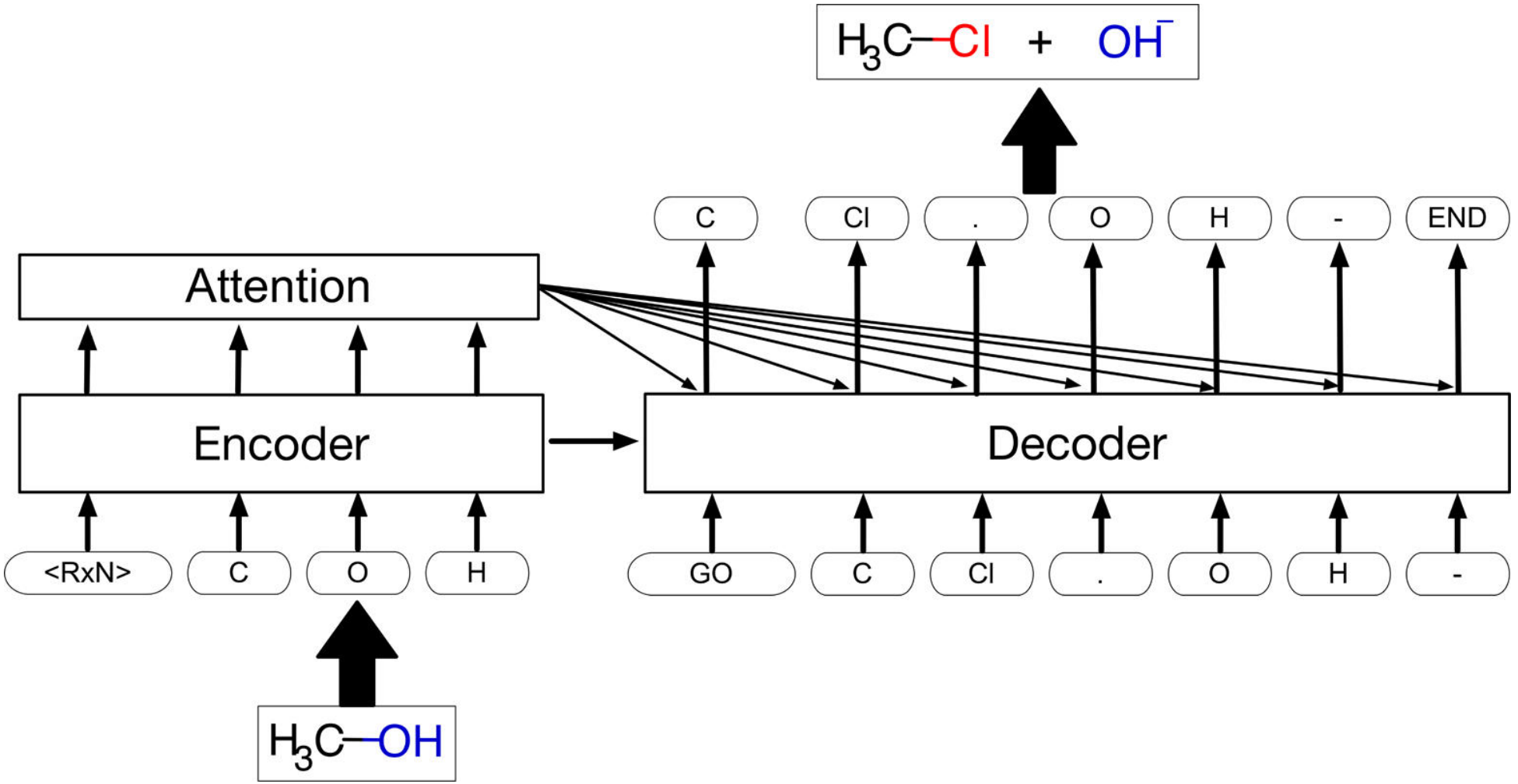
- Multiple input – multiple output (or sequence to sequence) scenario:



Synthesis prediction

(target)

ACS Cent. Sci. 2017, 3, 10, 1103-1113



ACS Cent. Sci. 2017, 3, 10, 1103-1113

Bidirectional

- LSTM and RNN are designed to analyze sequence of values.

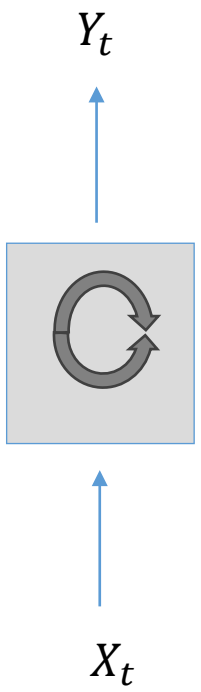
- For example: *Patrick said he needs a vacation.*
- *he* here means *Patrick* and we know this because *Patrick* was before the word *he.*

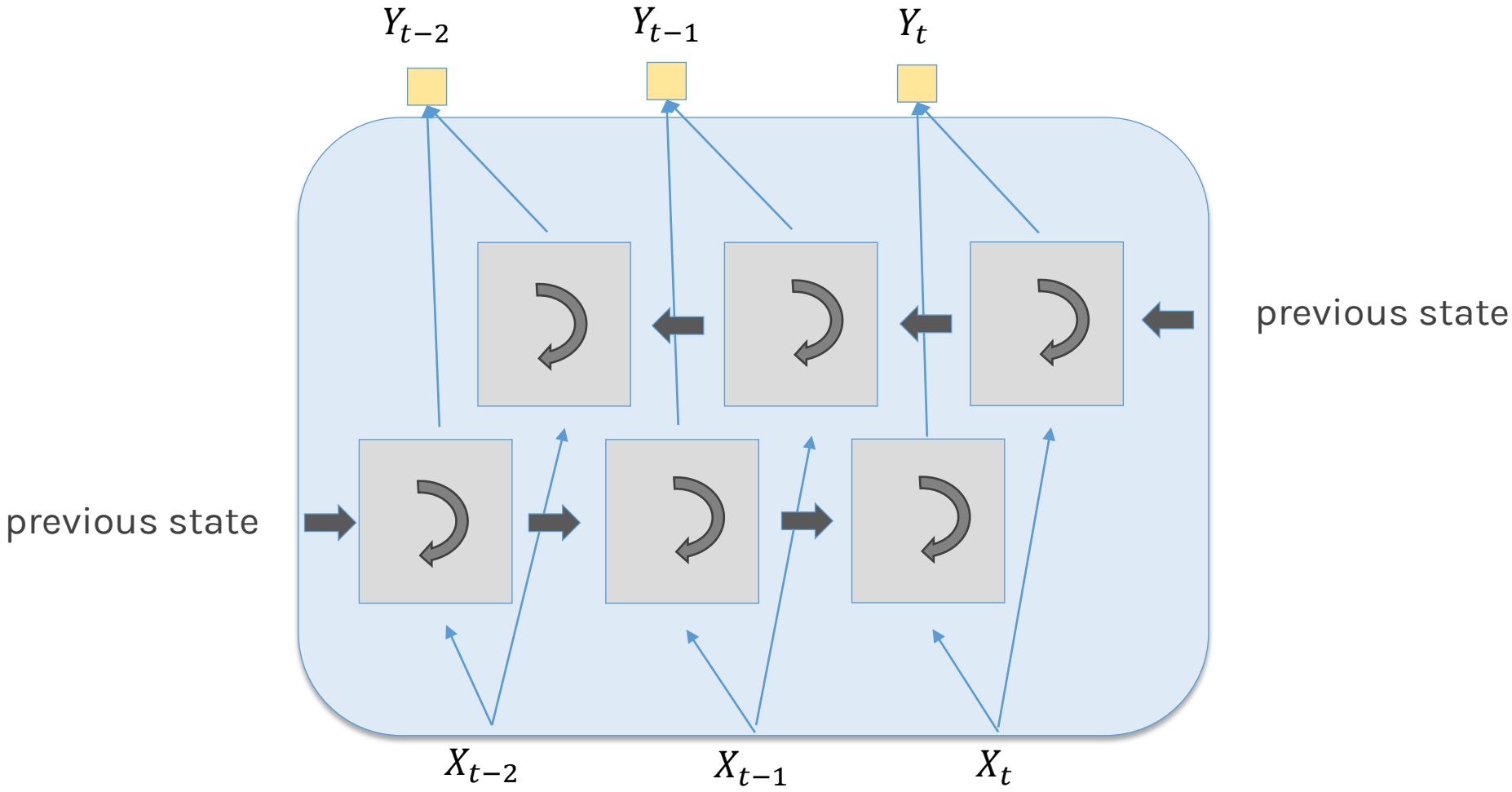
- However consider the following sentence:
- *He needs to work more, Peter said about Patrick.*

- Bidirectional RNN or BRNN or bidirectional LSTM or BLSTM when using LSTM units.

Bidirectional (cond)

symbol for a BRNN



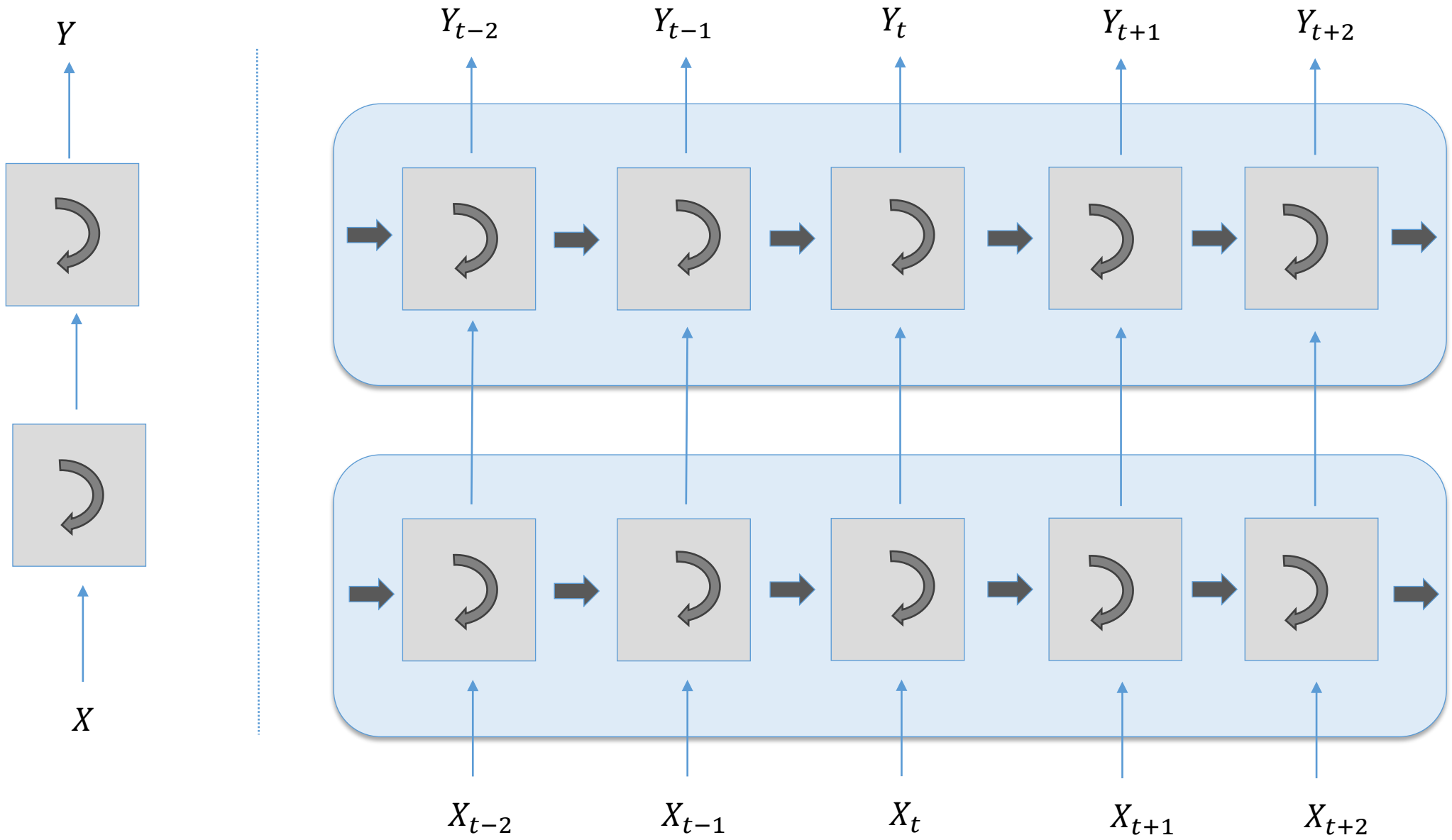


Deep RNN

- LSTM units can be arranged in layers, so that each the output of each unit is the input to the other units. This is called **a deep RNN**, where the adjective “deep” refers to these multiple layers.

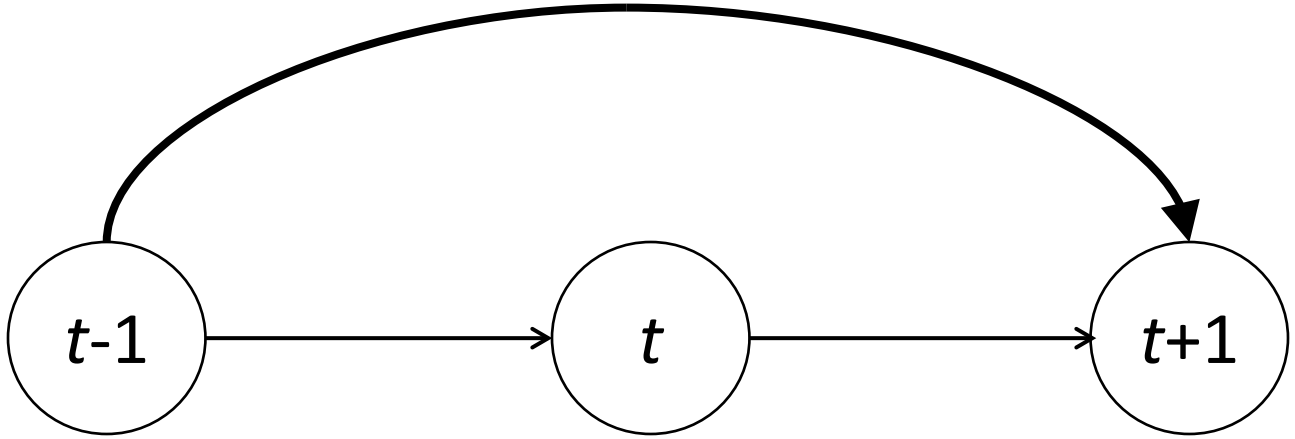
- Each layer feeds the LSTM on the next layer
- First time step of a feature is fed to the first LSTM, which processes that data and produces an output (and a new state for itself).
- That output is fed to the next LSTM, which does the same thing, and the next, and so on.
- Then the second time step arrives at the first LSTM, and the process repeats.

Deep RNN



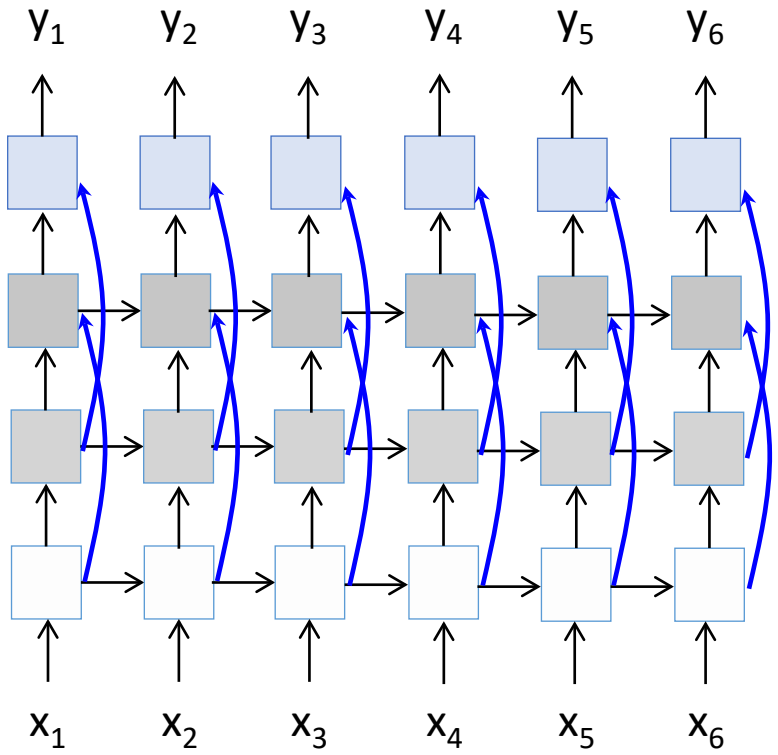
Skip Connections

- Add additional connections between units *d* time steps apart
- Creating paths through time where gradients neither vanish or explode



Multi-layer RNNs

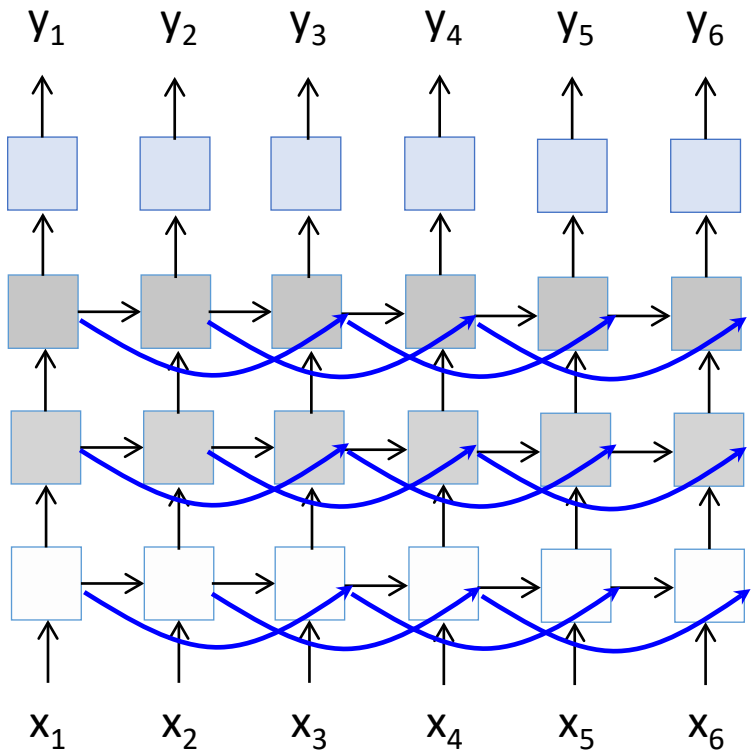
- We can of course design RNNs with multiple hidden layers



- Anything goes: skip connections across layers, across time, …

Multi-layer RNNs

- We can of course design RNNs with multiple hidden layers



- Anything goes: skip connections across layers, across time, …