Inspire…Educate…Transform.

Recurrent Neural Networks

Dr. Anand Narasimhamurthy

Senior Associate Professor
International School of Engineering

Outline

- Background and motivation for different architectures for sequence data
- Basics of Recurrent Neural Networks (RNN)
- Basics of Long Short Term Memory (LSTM) networks and Gated Recurrent Unit (GRU)
- A brief discussion on additional architectures

Learning outcomes

- An understanding of motivation of Recurrent Neural Networks (RNNS) for sequential data
- A basic understanding of Recurrent Neural Network architecture and their limitations
- A working understanding of Long Short Term Memory (LSTM) networks and Gated Recurrent Units (GRU)

Background and motivation

State of the art

Deep learning methods are the current state of the art in many application domains involving unstructured data. A few well known application domains include

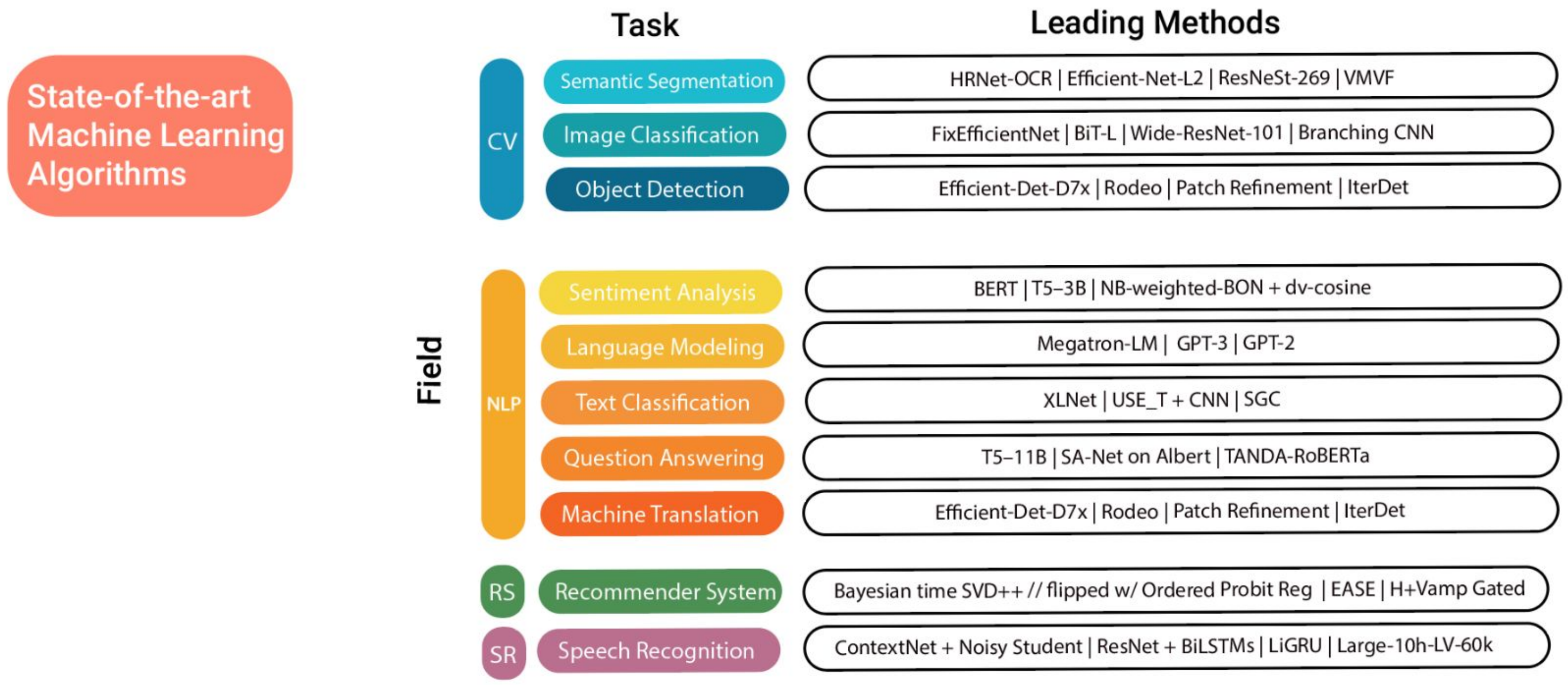
- **Computer vision**

Common tasks include image classification, object recognition etc.

- **NLP/Text analytics.**

Common applications include sentiment analysis,machine translation, question-answer

Overview: State-of-the-Art Machine Learning Algorithms per Discipline & per task, September 2020,



State-of-the-art Machine Learning Algorithms

Field	Task	Leading Methods
CV	Semantic Segmentation	HRNet-OCR | Efficient-Net-L2 | ResNeSt-269 | VMVF
CV	Image Classification	FixEfficientNet | BiT-L | Wide-ResNet-101 | Branching CNN
CV	Object Detection	Efficient-Det-D7x | Rodeo | Patch Refinement | IterDet
NLP	Sentiment Analysis	BERT | T5–3B | NB-weighted-BON + dv-cosine
NLP	Language Modeling	Megatron-LM | GPT-3 | GPT-2
NLP	Text Classification	XLNet | USE_T + CNN | SGC
NLP	Question Answering	T5–11B | SA-Net on Albert | TANDA-RoBERTa
NLP	Machine Translation	Efficient-Det-D7x | Rodeo | Patch Refinement | IterDet
RS	Recommender System	Bayesian time SVD++ // flipped w/ Ordered Probit Reg | EASE | H+Vamp Gated
SR	Speech Recognition	ContextNet + Noisy Student | ResNet + BiLSTMs | LiGRU | Large-10h-LV-60k

Image source :
https://towardsdatascience.com/overview-state-of-the-art-machine-learning-algorithms-per-discipline-per-task-c1a16a66b8bb

Motivation for sequential models

- A lot of applications involve sequential data
  Eg. Text, speech

  Consider for example, Natural language processing as the application domain.
  A few common NLP applications include

  - Machine Translation
  - Speech recognition
  - Conversation Modeling
  - Question Answering

NLP applications

Consider Natural Language Processing (NLP) applications.

- Most NLP tasks require knowing the entire set of previous words
- Local context, though useful is not sufficient
- Sentiment analysis :
  - Not unusually, it rained a lot in August.   (no surprise)
  - It is unusual that it did not rain a lot in August (surprise)

The set of words are almost the same, but the meaning is completely different !

NLP applications

- Option 1 : Train with entire sentences
  Disadvantages
  - Training data becomes too large
  - Cannot predict all possible sentences
  - Length of sentences differ
  - Only a few past words may matter

- Option 2 : Capture the sentence as “temporal weights”

Motivation for different architecture

- An essential aspect of understanding sequential data is **persistence.** For example, as we read a piece of text, we are not learning from scratch every time. We have stored the information in the most recent preceding words/sentences as well as that from paragraphs far back.

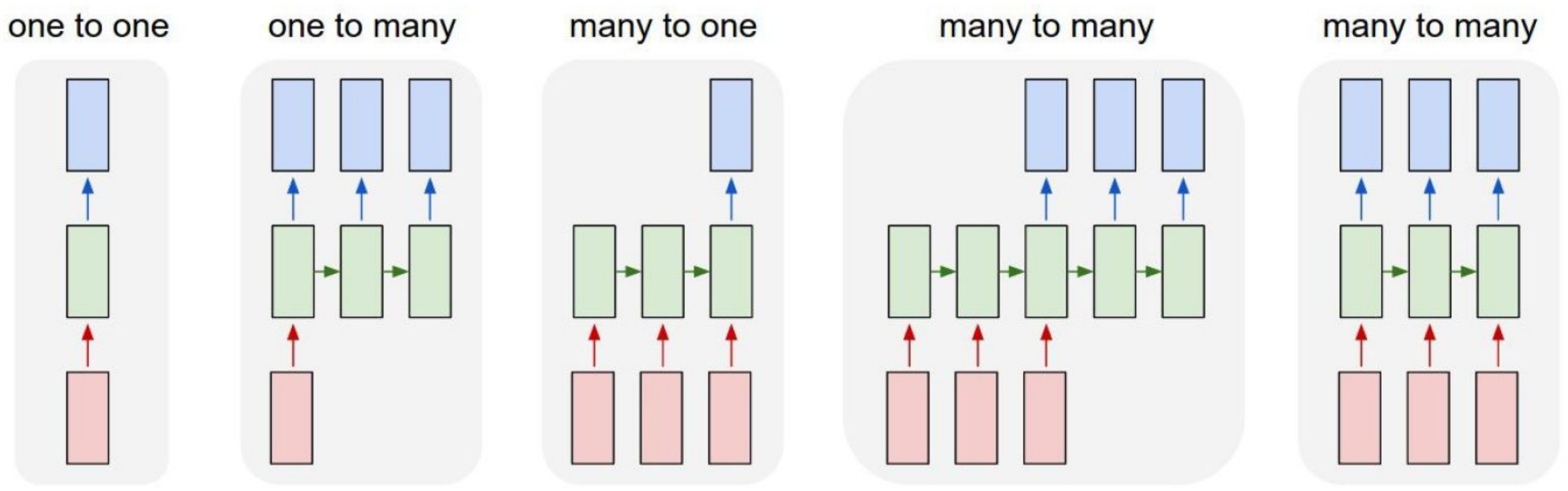
- A feedforward neural network does not have a natural capacity for persistence of information.

- Additionally, the number of inputs and outputs could be different depending on the application

- Also the length of the local context is variable and not known before hand

All the above considerations necessitate a different architecture.
One of the most commonly used architectures for sequence data is the **Recurrent Neural Network (RNN)**

Different types of inputs and outputs in sequences



e.g. **Video classification on frame level**

Vanilla Neural Networks

Different types of inputs and outputs in sequences

Input are in red, output vectors are in blue and green vectors hold the RNN's state. From left to right:

(1) Vanilla mode of processing without RNN, from fixed-sized input to fixed-sized output (e.g. image classification).

(2) Sequence output (e.g. image captioning takes an image and outputs a sentence of words).

(3) Sequence input (e.g. sentiment analysis where a given sentence is classified as expressing positive or negative sentiment).

(4) Sequence input and sequence output (e.g. Machine Translation: an RNN reads a sentence in English and then outputs a sentence in French).

(5) Synced sequence input and output (e.g. video classification where we wish to label each frame of the video).

Recurrent neural networks : the basics

Recurrent neural networks

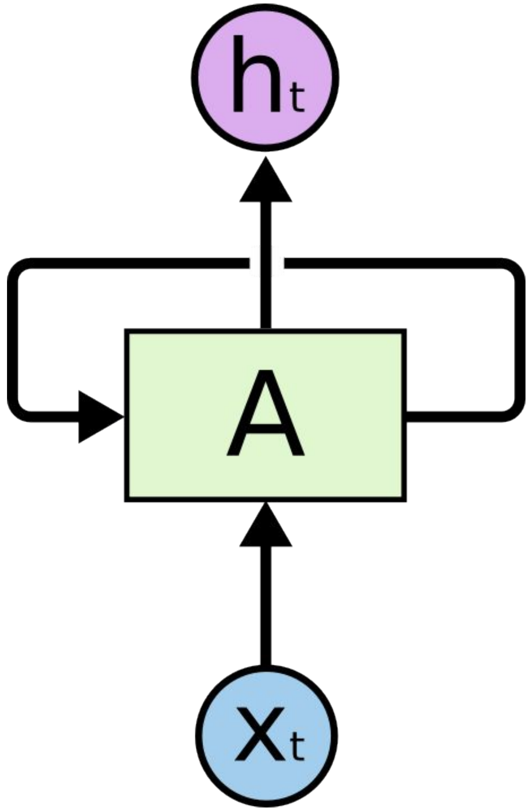
A Recurrent neural network is created by applying the same set of weights over a differentiable graph-like structure

Introduced first in 1986 (Rumelhart et al 1986)

Mostly used to handle sequential data of arbitrary input lengths.

A single Recurrent Neural Network unit

- An RNN includes a “memory unit”

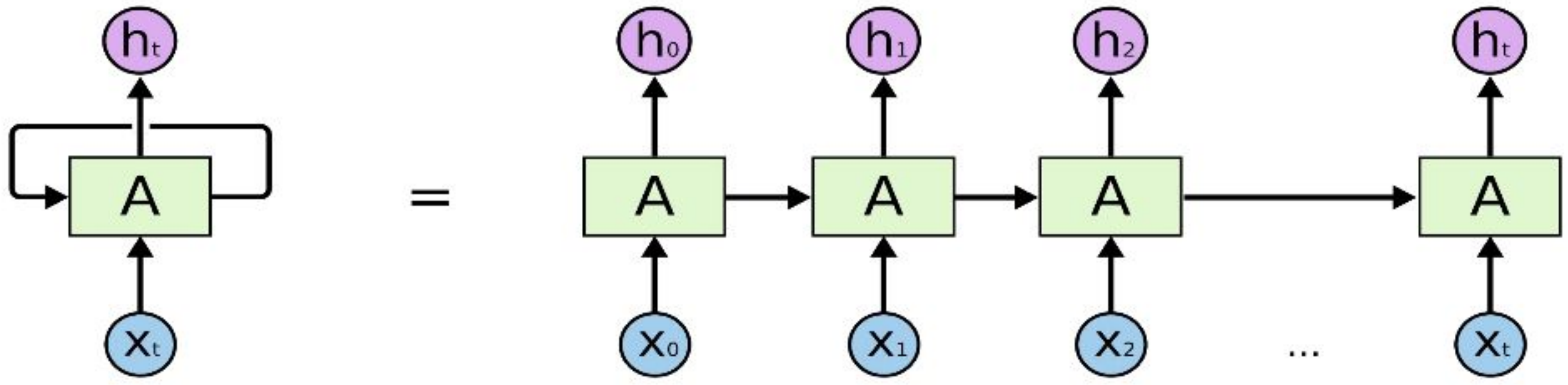


A chunk of neural network *A*, looks at some input x_t and outputs a value h_t

A loop allows information to be passed from one step of the network to the next.

Source : http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Unrolling a Recurrent Neural Network unit



Source : http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Recurrent Neural Network

We can process a sequence of vectors **x** by applying a recurrence formula at every time step:

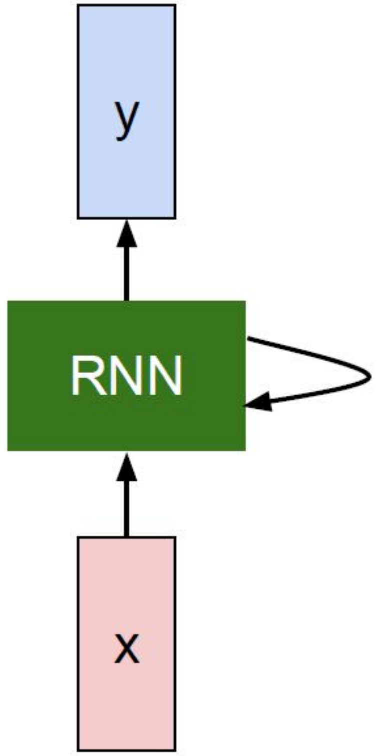
$$\boxed{h_t} = \boxed{f_W}(\boxed{h_{t-1}}, \boxed{x_t})$$

new state

some function with parameters W

old state

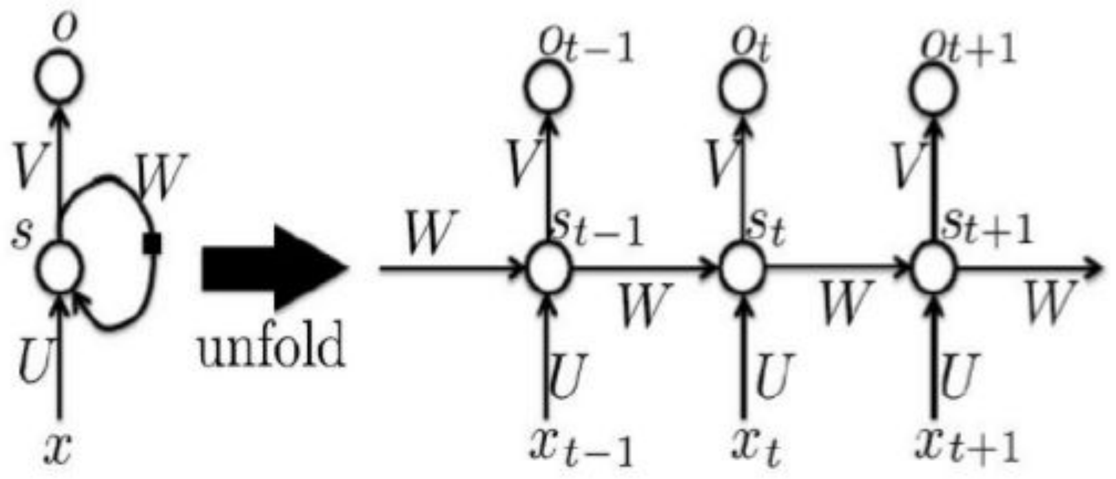
input vector at some time step

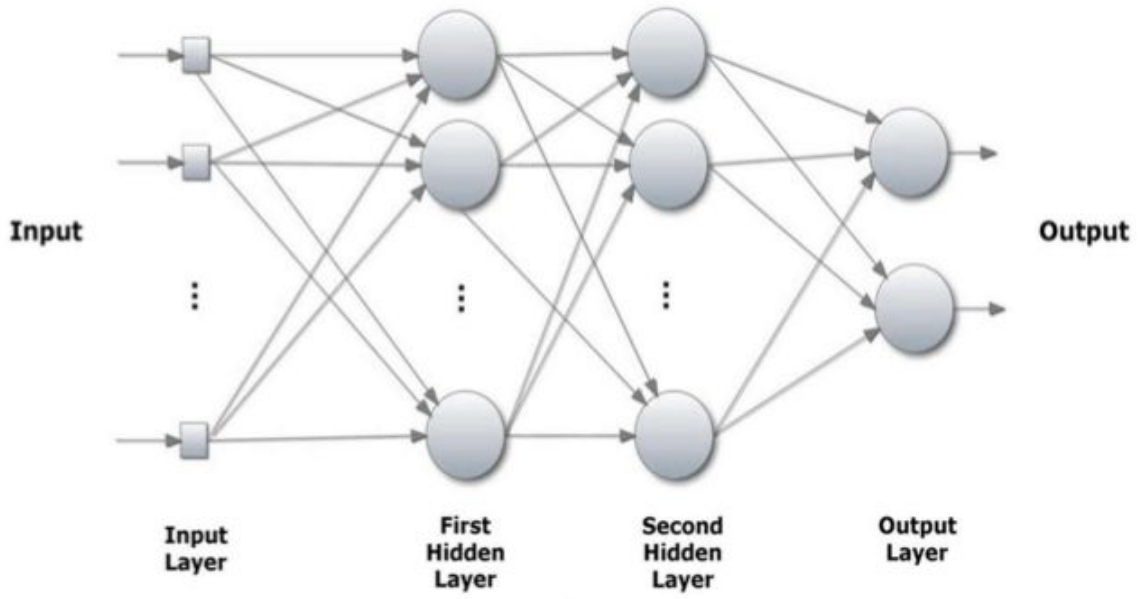


Notice: the same function and the same set of parameters are used at every time step.

Storing states can be achieved by having self-loops for the hidden layer units.

Comparison with feed forward networks





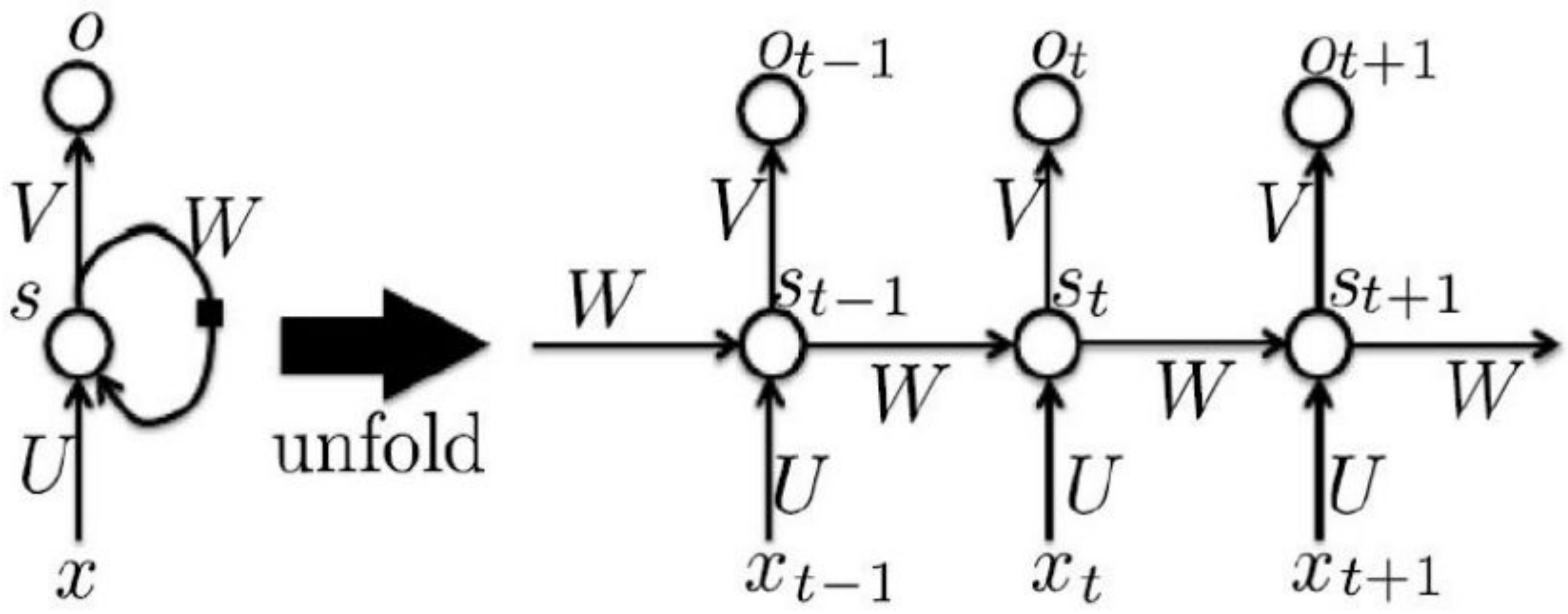
- Parameter sharing across sequence
- Multiple inputs at different time instances
- Self or recurrent connections

Training RNNs

- Backpropagation is the most commonly used method for training RNNs.
- However unlike in feedforward neural networks, the training begins by unfolding a recurrent neural network through time This is referred to as Backpropagation through time (BPTT)

- After each pattern is presented, weight updates are computed for time instance.

All weight updates are averaged together.

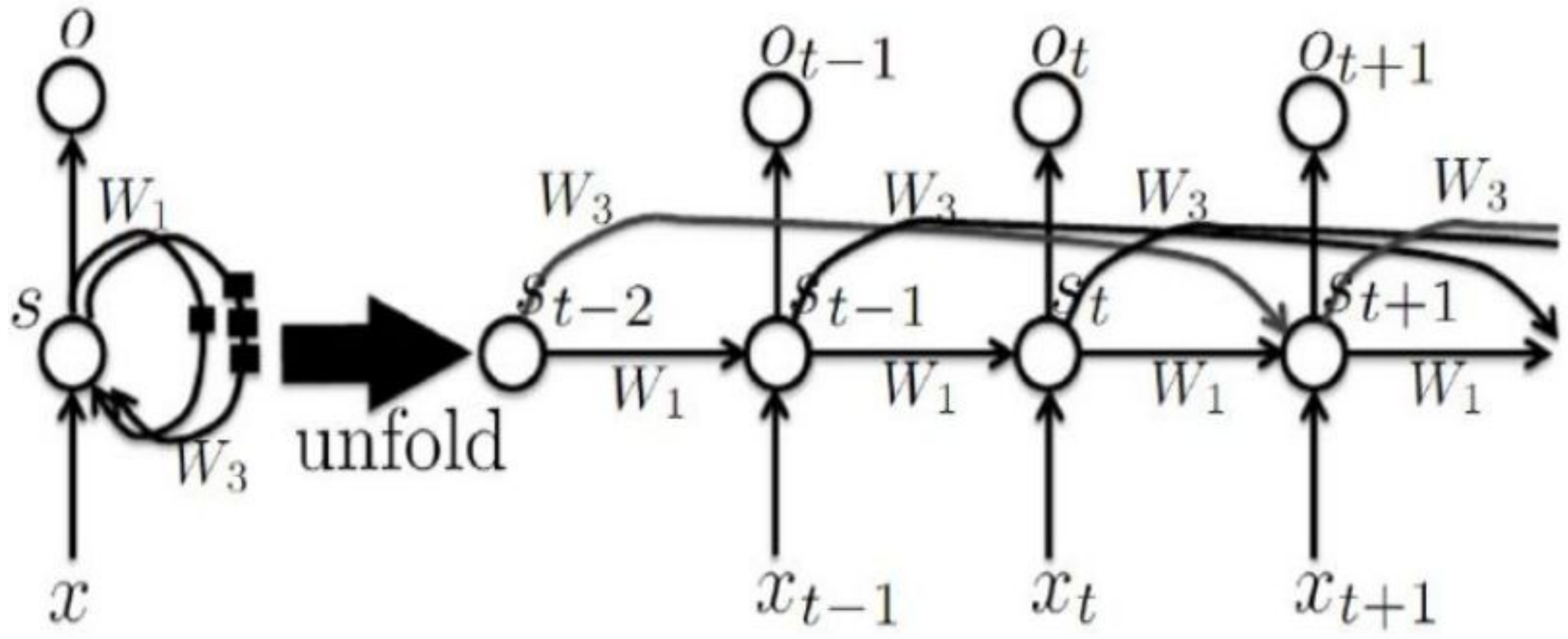


Issues with plain vanilla RNNs

- **Vanishing Gradient Problem**
  - Using BPTT results in a vanishing gradient problem. This is more severe than feedforward networks.
  - Error is propagated back in time where each time step is exactly equivalent to propagating through an additional layer of a feed forward network.
- **Exploding gradients**
  - The final output is a composition of a large number of non-linear transformations. The partial derivatives through the composition can be large (or small)

A partial mitigation for vanishing gradients

Use Recurrent Networks with long delays



A partial mitigation for exploding gradients

- Simple solution can be to clip the gradient
  - Clip the parameter gradient from a mini batch element-wise (Mikolov, 2012) just before the parameter update.
  - Clip the norm g of the gradient (Pascanu et al., 2013) just before the parameter update.

Language modeling with RNNs

Language models

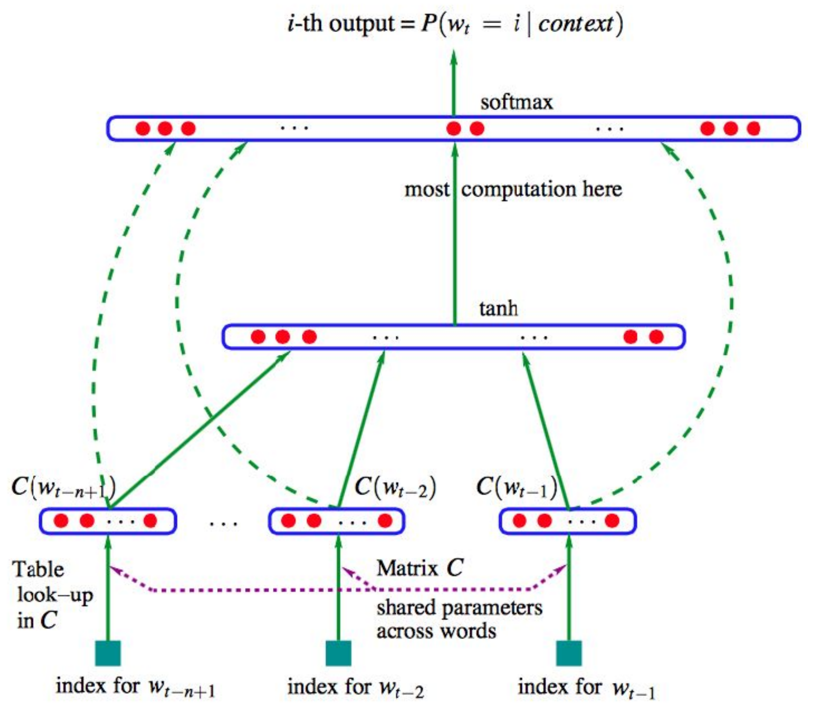
Predicting the next word (or words) has been a breakthrough approach in modern Natural Language Processing.

Language Modeling is the task of predicting what word comes next

More formally: given a sequence of words, $x^{(1),}, x^{(2)}, \ldots, x^{(t),}$
compute the probability distribution of the next word $x^{(t+1)}$
i.e. compute $P(x^{(t+1)} \mid x^{(1),}, x^{(2)}, \ldots, x^{(t),})$ where $x^{(t+1)}$ is any word in the vocabulary.

Original Neural Language Model using MLPs

- Predict the i^{th} word based on a fixed size context.
- In all conventional language models, the memory requirements of the system grows exponentially with the window size n making it nearly impossible to model large word windows without running out of memory.
- Problem: Fixed window of context (i.e., n)
- Can there be a model which can handle contexts of different length?



$$\hat{y} = softmax(W^{(2)}tanh(W^{(1)}x + b^{(1)}) + W^{(3)}x + b^{(3)})$$

RNN Language Model

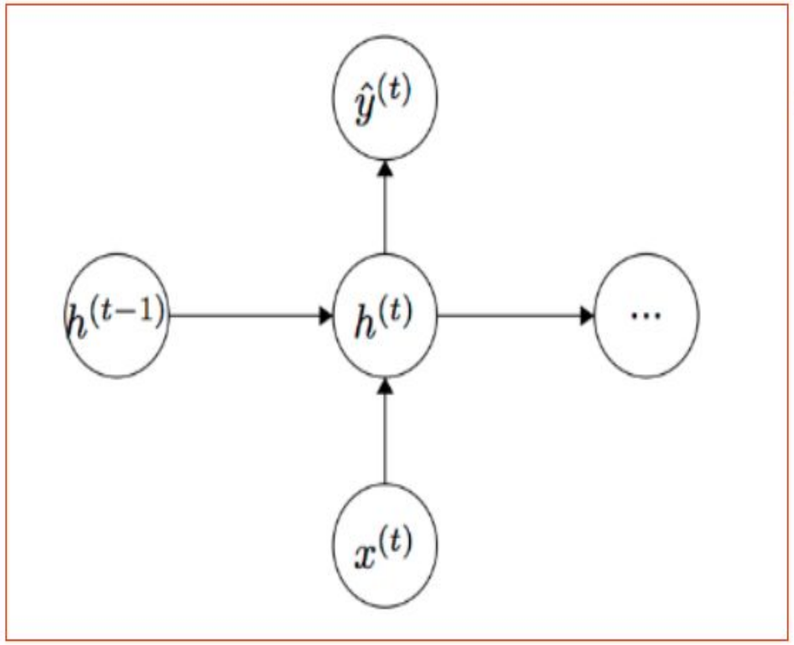
Given list of word **vectors:** $x_1, \ldots, x_{t-1}, x_t, x_{t+1}, \ldots, x_T$

At a single time step:

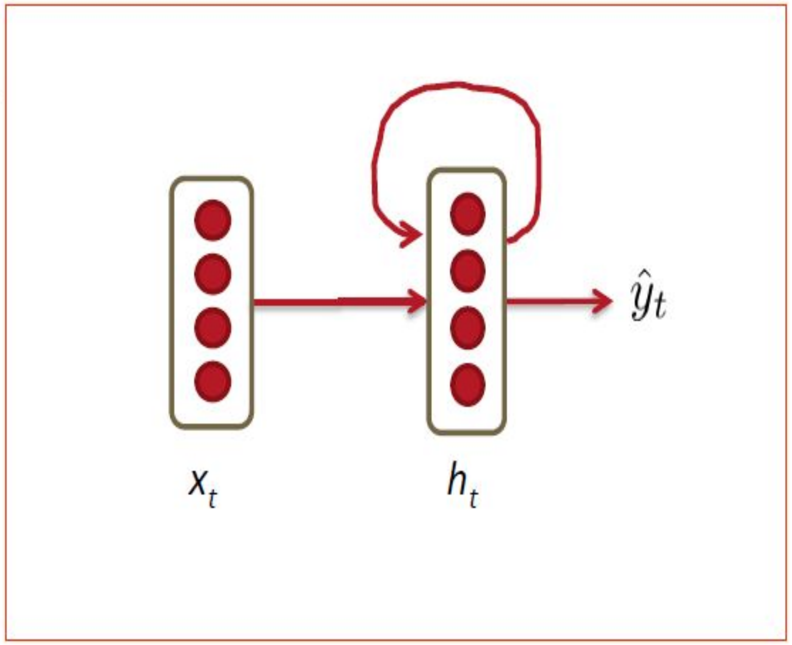
$$h_t = \sigma\left(W^{(hh)} h_{t-1} + W^{(hx)} x_{[t]}\right)$$

$$\hat{y}_t = \text{softmax}\left(W^{(S)} h_t\right)$$

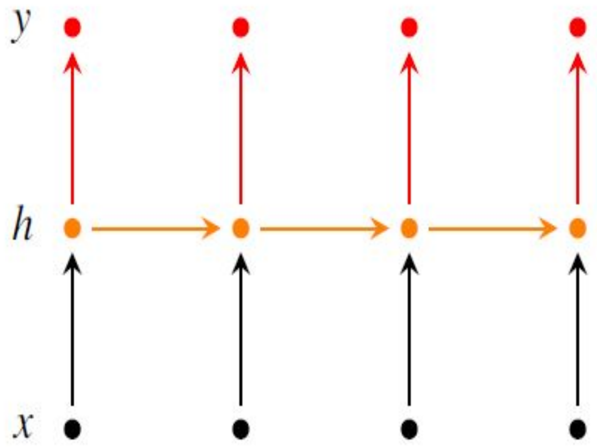
$$\hat{P}(x_{t+1} = v_j \mid x_t, \ldots, x_1) = \hat{y}_{t,j}$$



$\leftrightarrow$



Modeling using RNNs



$$h_t = f(Wx_t + Vh_{t-1} + b)$$
$$y_t = g(Uh_t + c)$$

- *x* represents a token (word) as a vector.
- *y* represents the output label (B, I or O) – g = softmax
- *h* is the memory, computed from the past memory and current word. It summarizes the sentence up to that time.

Retaining relevant information from the past

The importance of network state at times which lie far back in the past.
How much of the past is relevant?

Gap between relevant information and where it is needed is important.
Consider for example, predicting the last word in the text:
•“the clouds are in the sky”. Not much context is required as obvious. (small gap)

•“I grew up in Karnataka. ... I speak fluent kannada”. (large gap)
As the gap grows, RNNs unable to learn to connect information

Retaining relevant information from the past

Although in theory, a Recurrent Neural Network can retain information from arbitrarily long sequences, this is not feasible practically.

Typically, gradient based networks cannot reliably use information which lies more than about 10 time steps in the past.

A human carefully analyzes the passage and picks the right features to put in the memory i.e. a natural architecture that remembers the relevant past information.

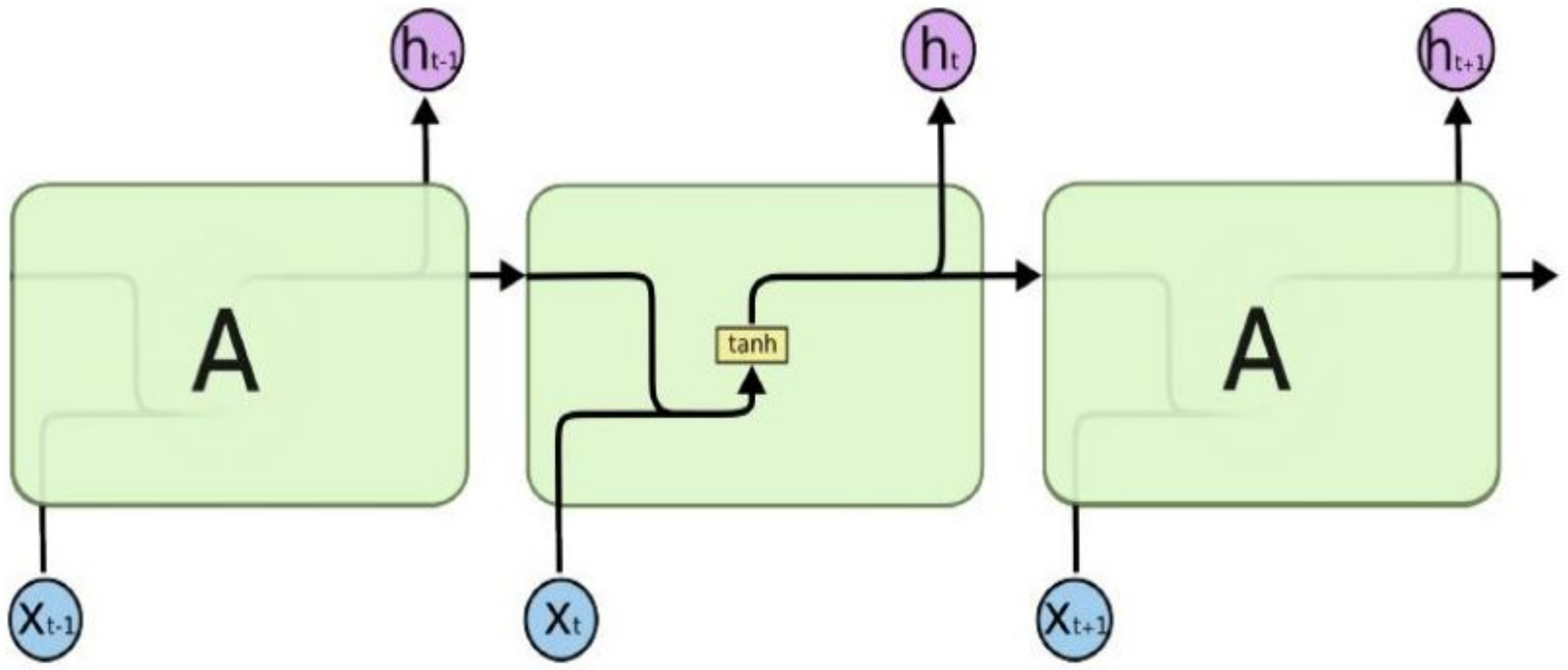
In an artificial system we need a mechanism which can facilitate update of memory only when required.

Gated RNNs

Repeating units

All recurrent neural networks have the form of a chain of repeating modules as shown in the figure below.

In standard RNNs, the repeating module has a simple structure
eg. a single tanh layer



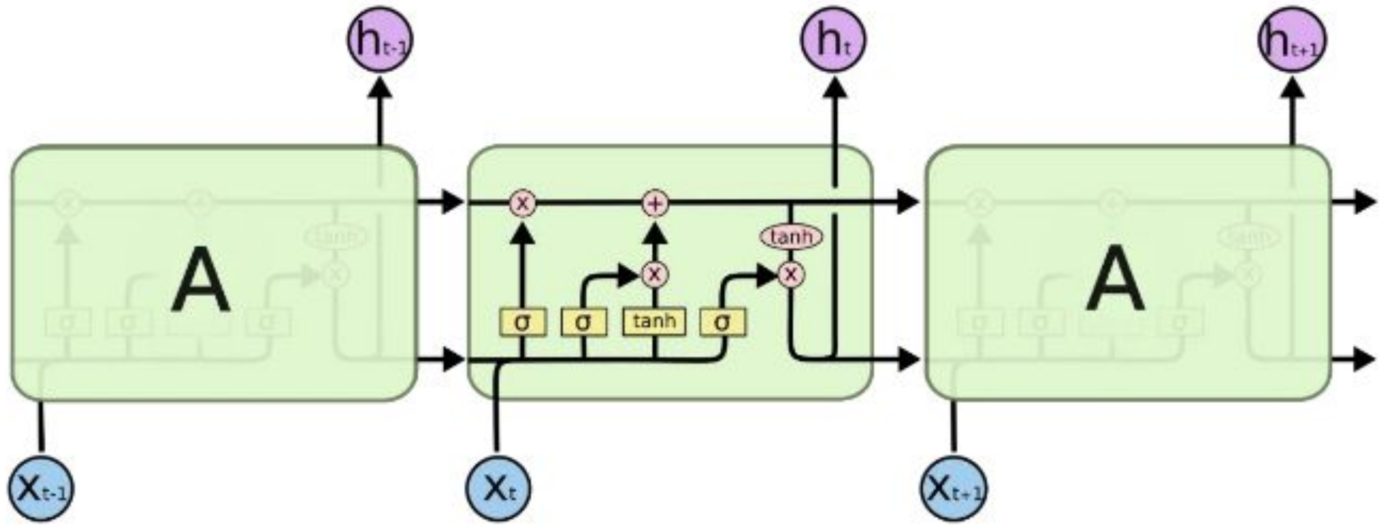
By introducing additional elements in the repeating module, more sophisticated sub tasks can be performed.

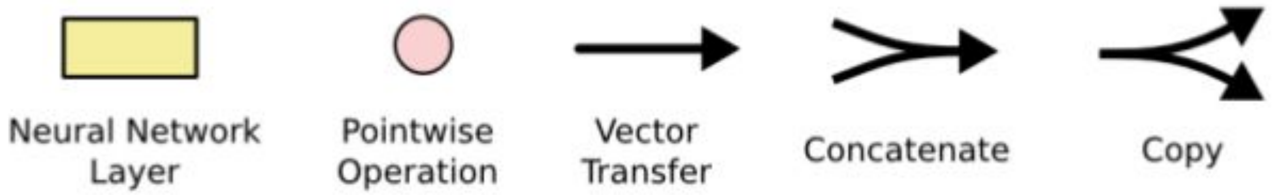
The resulting network is called a **Long Short Term Memory (LSTM)** network.

- LSTMs help preserve the error that can be backpropagated through time and layers.

- They allow recurrent nets to continue to learn over many time steps (over 1000)
  This opens a channel to link causes and effects remotely.

LSTMs

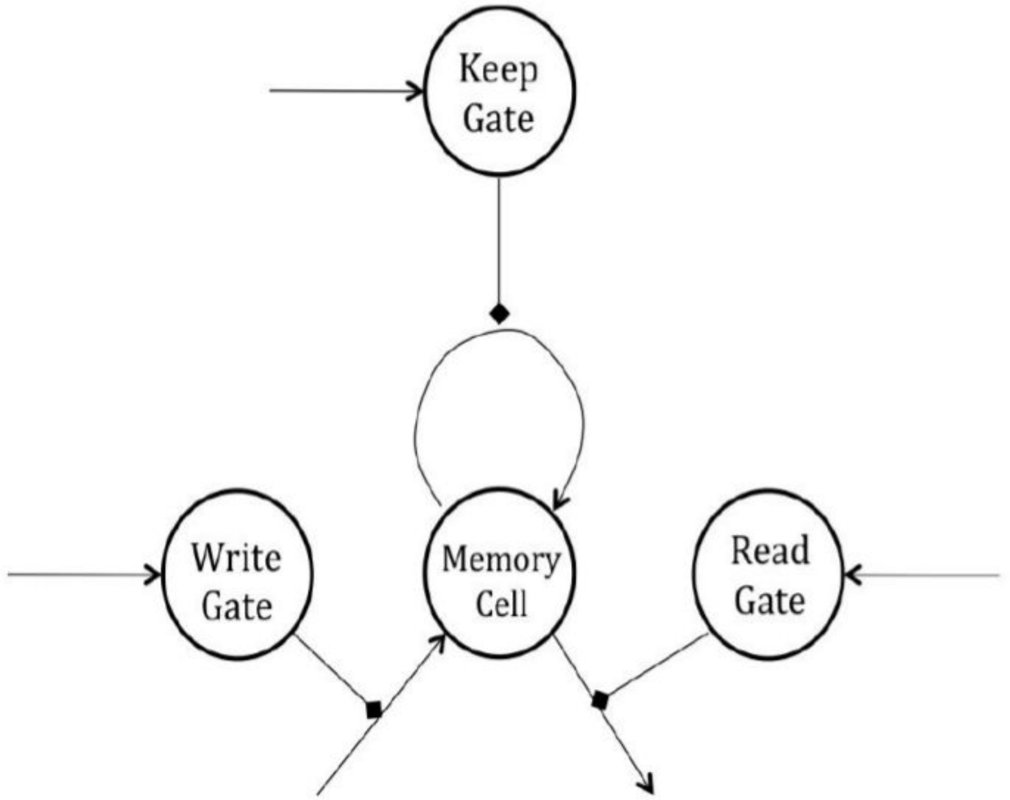




Each line carries an entire vector, from the output of one node to the inputs of others.

- The pink circles are point-wise operations, like vector addition
- The yellow boxes are learned neural network layers.
- Lines merging denote concatenation
- Line forking denote its content being copied and the copies going to different locations.

- LSTMs contain information outside the normal flow of the recurrent network in a gated cell.
    - Information can be stored in, written to, or read from a cell, much like data in a computer's memory.
    - The cell makes decisions about what to store, and when to allow reads, writes and erases, via gates that open and close.

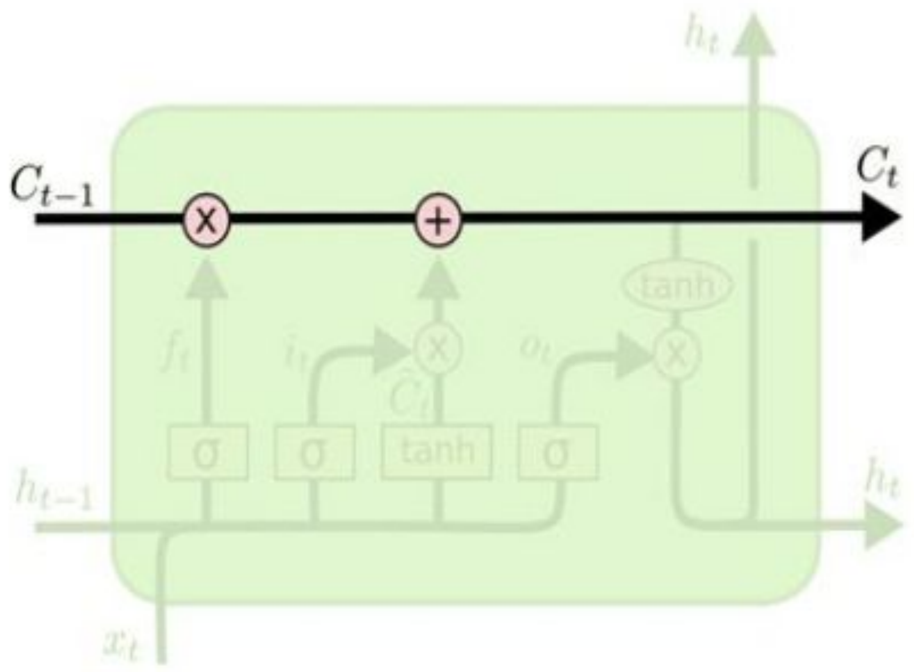


An LSTM typically has three gates, to protect and control the cell state.

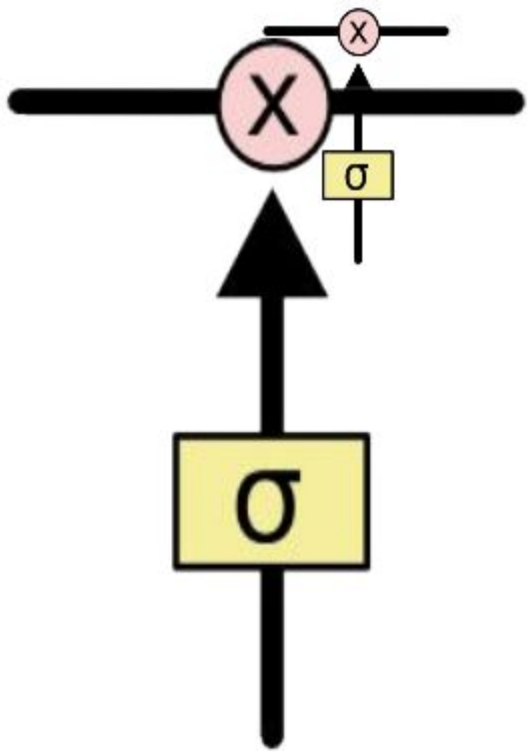
Elements of an LSTM

Cell state :

- The cell state, depicted by the horizontal line running through the top of the diagram below is the key to LSTMs.
- The cell state is kind of like a conveyor belt. It runs straight down the entire chain, with only some minor linear interactions. It is very easy for information to just flow along it unchanged.



Add/remove information to cell state in a regulated manner



- An LSTM has the ability to remove or add information to the cell state, carefully regulated by structures called gates.

- Gates are a mechanism to optionally let information through.

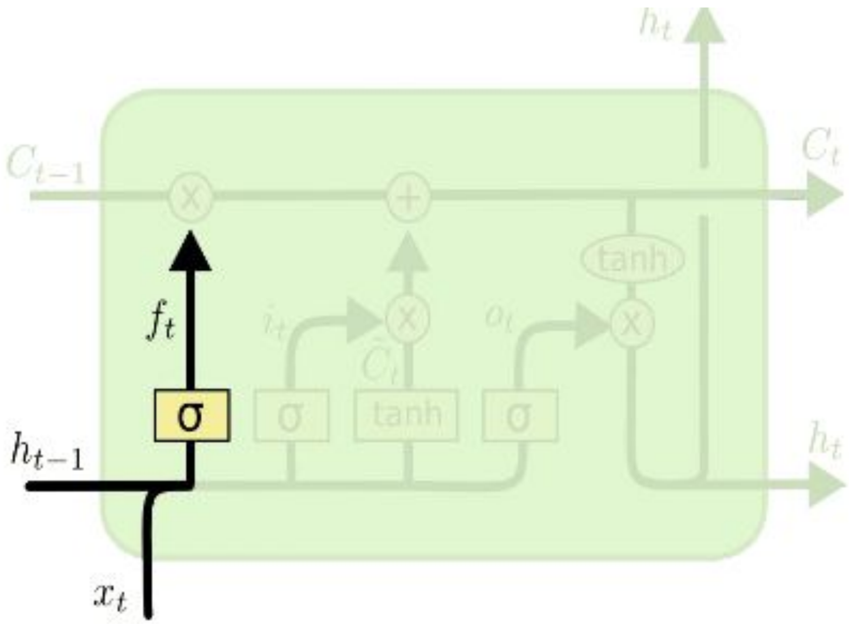
- Gates are composed out of a sigmoid neural net layer and a point-wise multiplication operation.

Add/remove information to cell state in a regulated manner

The first step in our LSTM is to decide what information we’re going to throw away from the cell state.

This decision is made by a **sigmoid layer** called the “**forget gate layer**.”

The sigmoid layer outputs numbers between zero and one, describing how much of each component should be let through.
0 : means “let nothing through,”
1 : “let everything through!”



Add/remove information to cell state in a regulated manner

The next step in our LSTM is to decide what information is going to be stored in the cell state. This is usually done in two parts.

- A sigmoid layer called the “input gate layer” decides which values will be updated.

- Next, a tanh layer creates a vector of new candidate values, C_t, that could be added to the state.

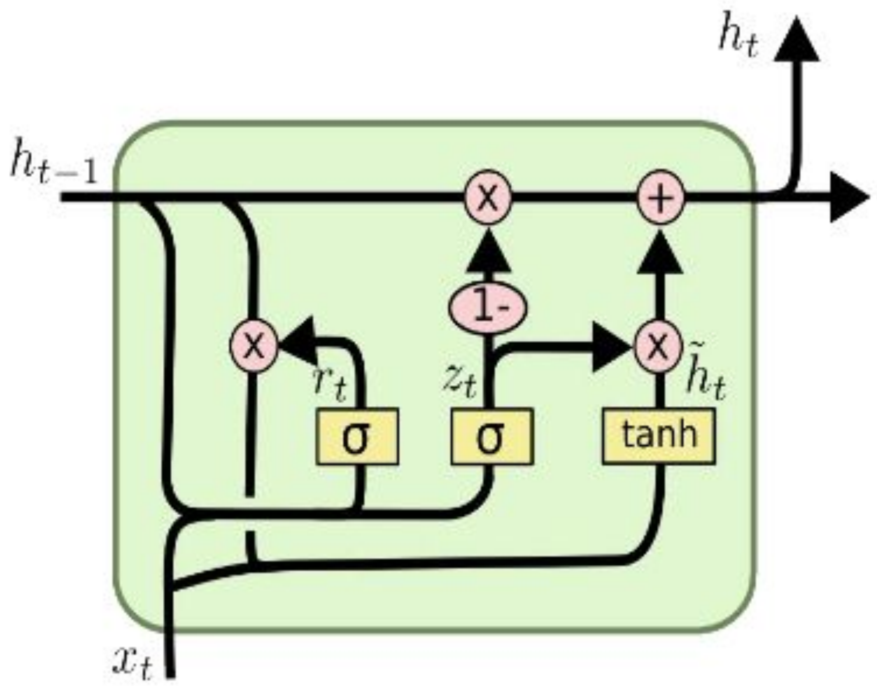
The final step involves updating the cell state based on the previous two steps i.e. taking into account what information is retained and what is discarded.

Additional architectures

Gated Recurrent Unit (GRU)

- A slightly more dramatic variation on the LSTM is the Gated Recurrent Unit, or GRU, introduced by Cho, et al. (2014).
- It combines the forget and input gates into a single “update gate.”
- It also merges the cell state and hidden state, and makes some other changes.
- The resulting model is simpler than standard LSTM models, and has been growing increasingly popular.

Gated Recurrent Unit (GRU)



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

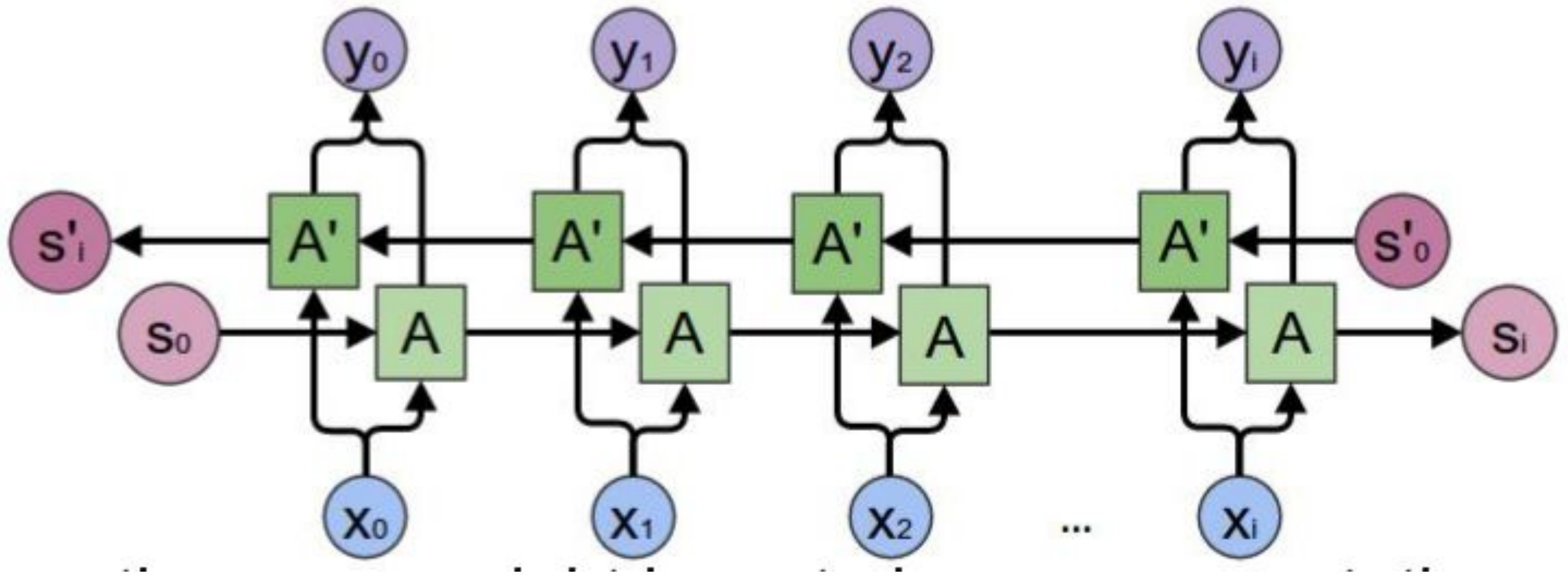
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Bi-directional models

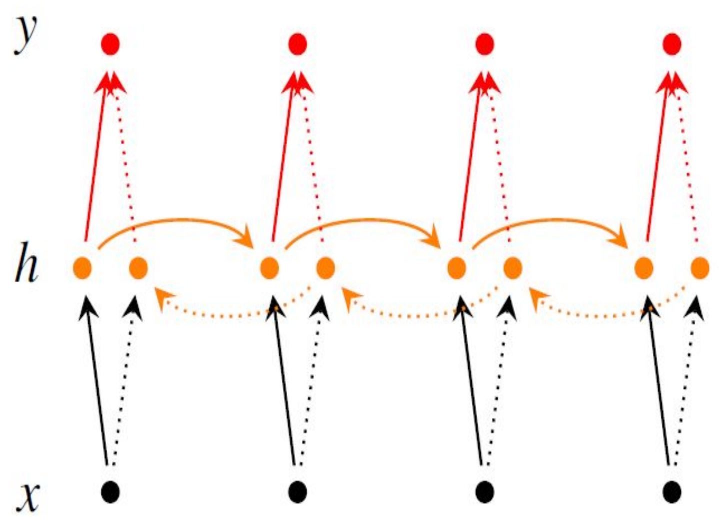
- Sometimes, you might have to learn representations from future time steps to better understand the context and eliminate ambiguity.
  An NLP example :
  “He said, Teddy bears are on sale”
  “He said, Teddy Roosevelt was a great President”

- This can be achieved by having connections in both directions.



Bidirectional RNNs

Problem: For classification you want to incorporate information from words both preceding and following



$$\overrightarrow{h}_t = f(\overrightarrow{W} x_t + \overrightarrow{V}\overrightarrow{h}_{t-1} + \overrightarrow{b})$$

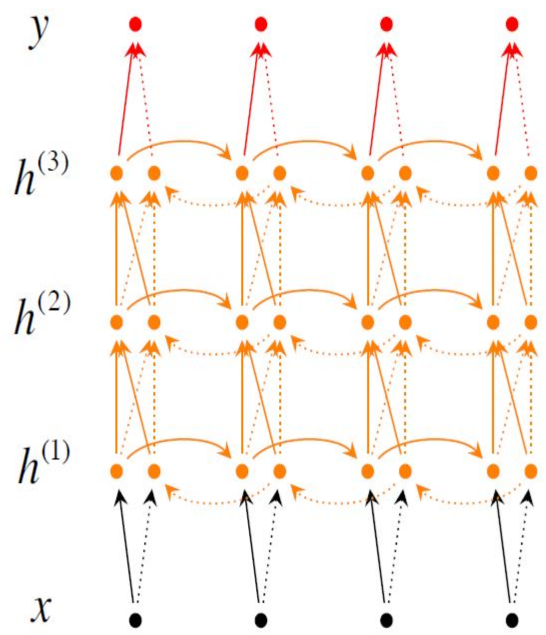
$$\overleftarrow{h}_t = f(\overleftarrow{W} x_t + \overleftarrow{V}\overleftarrow{h}_{t+1} + \overleftarrow{b})$$

$$y_t = g(U[\overrightarrow{h}_t;\overleftarrow{h}_t] + c)$$

$h = [\overrightarrow{h};\overleftarrow{h}]$ now represents (summarizes) the past and future around a single token.

To maintain two hidden layers at any time, this network consumes twice as much memory space for its weight and bias parameters.

Deep Bidirectional RNNs



$$\overrightarrow{h}_t^{(i)} = f(\overrightarrow{W}^{(i)} h_t^{(i-1)} + \overrightarrow{V}^{(i)} \overrightarrow{h}_{t-1}^{(i)} + \overrightarrow{b}^{(i)})$$

$$\overleftarrow{h}_t^{(i)} = f(\overleftarrow{W}^{(i)} h_t^{(i-1)} + \overleftarrow{V}^{(i)} \overleftarrow{h}_{t+1}^{(i)} + \overleftarrow{b}^{(i)})$$

$$y_t = g(U[\overrightarrow{h}_t^{(L)}; \overleftarrow{h}_t^{(L)}] + c)$$

Each memory layer passes an intermediate sequential representation to the next.

At time-step t each intermediate neuron receives one set of parameters from the previous time-step (in the same RNN layer), and two sets of parameters from the previous RNN hidden layer; one input comes from the left-to-right RNN and the other from the right-to-left RNN.

Encoder decoder models

- The encoder-decoder model is a recurrent neural network architecture for sequence-to-sequence prediction problems.

- Initially developed for **machine translation problems**, it has proven successful at related sequence-to-sequence prediction problems such as **text summarization** and **question answering**.

- The approach involves two recurrent neural networks, one to encode the input sequence, called the encoder, and a second to decode the encoded input sequence into the target sequence called the decoder.

Encoder decoder models

- **Encoder :** It reads the input sequence and summarizes the information in **internal state vectors** or **context vector**

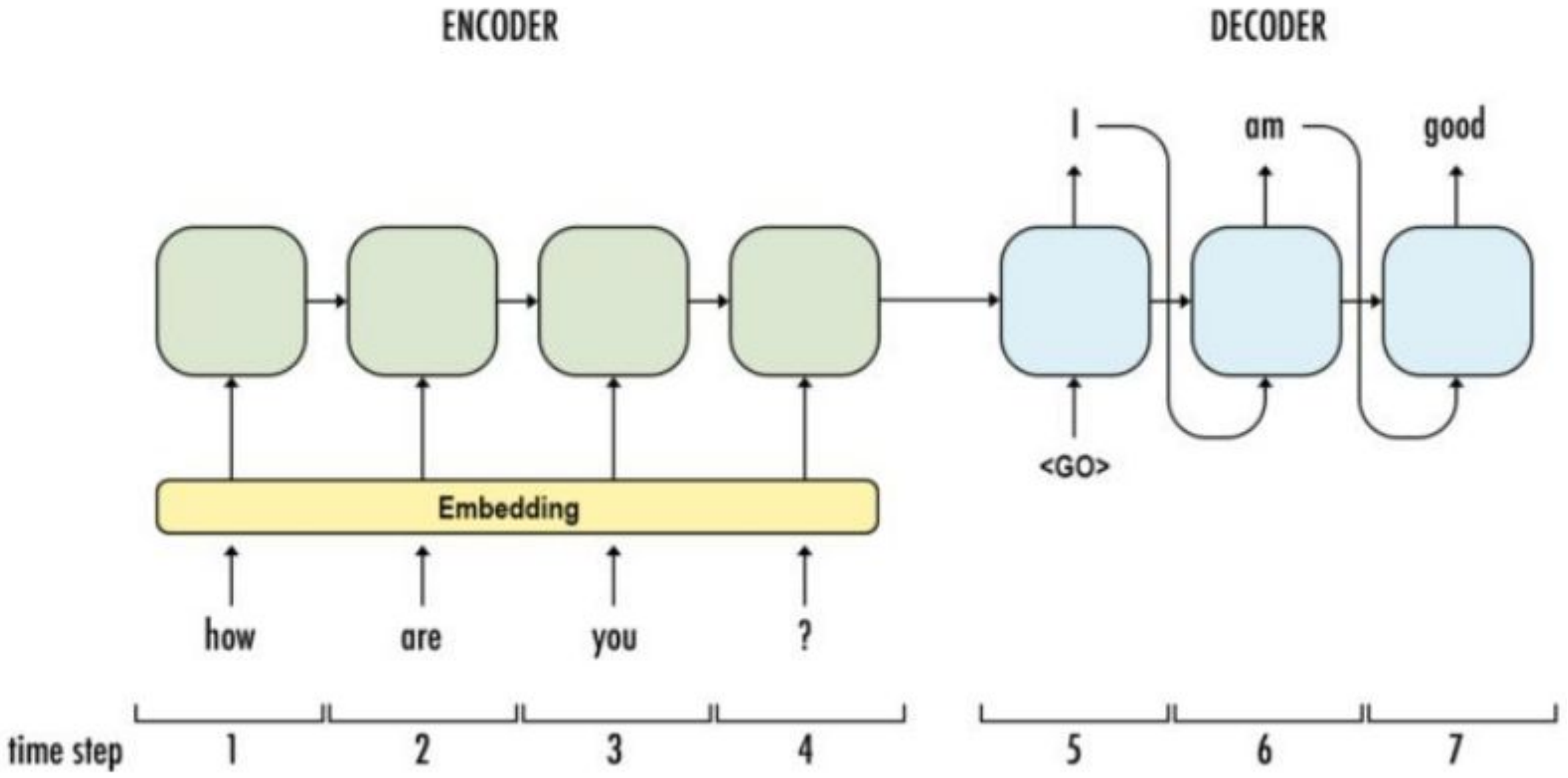
If an LSTM network is used as the encoder, these are called the hidden state and cell state vectors
Typically the hidden and cell state of the network are passed along to the decoder as input.

Decoder : A decoder interprets the context vector obtained from the encoder.

- The context vector of the encoder's final cell is input to the first cell of the decoder network.
- Using these initial states, the decoder starts generating the output sequence, and these outputs are also taken into consideration for future predictions.

https://medium.com/data-science-community-srm/understanding-encoders-decoders-with-attention-based-mechanism-c1eb7164c581

Encoder decoder models



An encoder decoder network used for conversation modeling

Image source : https://medium.com/syncedreview/a-brief-overview-of-attention-mechanism-13c578ba9129

Limitations of vanilla encoder decoder models

Although LSTMs can retain some relevant context information, an encoder decoder architecture would be unable to extract strong contextual relations from long semantic sentences.

This is partly because the information learned is compacted into a fixed length vector.

This leads to loss of information and accuracy especially when dealing with long sentences.

Attention mechanism

The drawback can be addressed via **Attention mechanism.**

Basic idea
To provide a more signified context from the encoder to the decoder and a learning mechanism where the decoder can interpret where to actually give more ‘attention’ to the subsequent encoding network when predicting outputs at each time step in the output sequence.

Attention is simply a vector, often the outputs of dense layer using softmax function.

Attention mechanism

The attention mechanism allows the network to learn where to pay attention in the input sequence for each item in the output sequence.

The key benefits of the approach are the ability to train a single end-to-end model directly on source and target sentences and the ability to handle variable length input and output sequences of text.

The common sequence-to-sequence learning framework with attention was the core of Google Translation service
Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation, 2016

Illustration of attention model used for machine translation

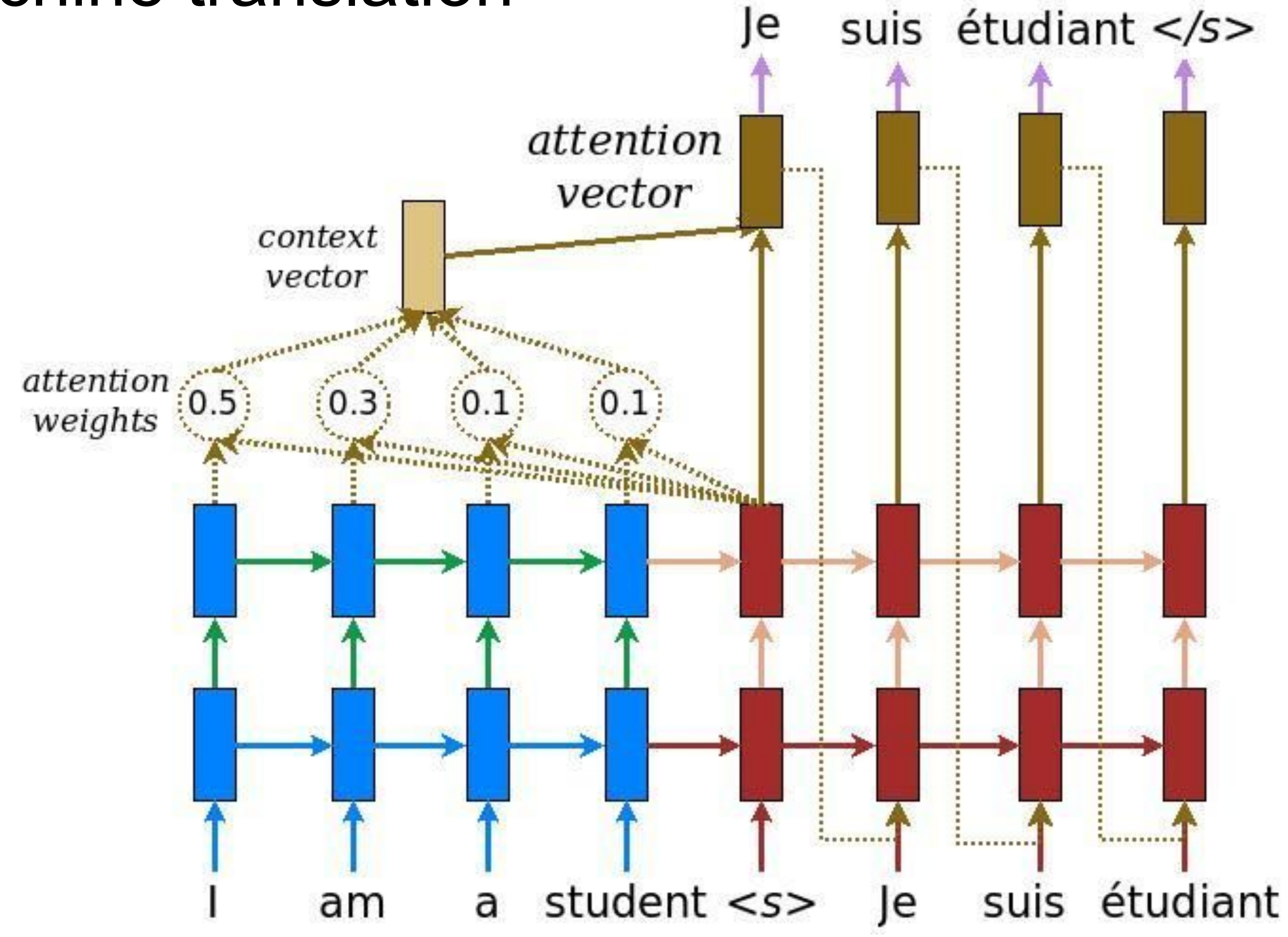
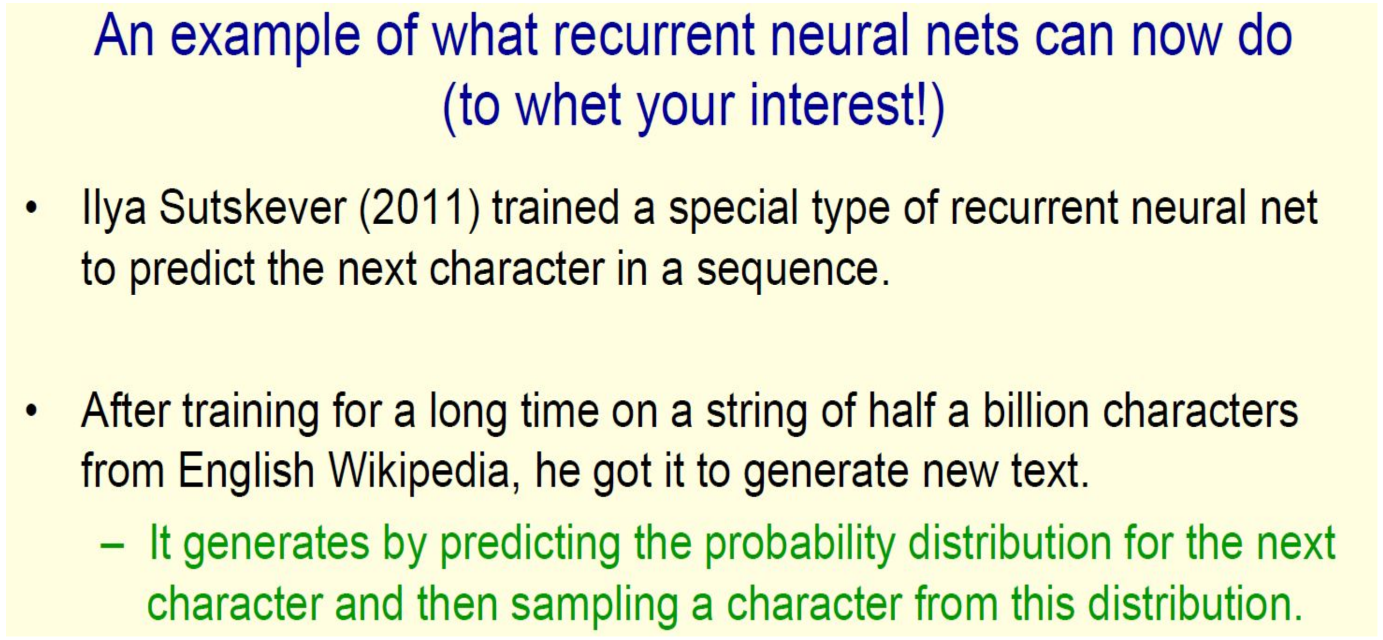


Image source : https://medium.com/syncedreview/a-brief-overview-of-attention-mechanism-13c578ba9129

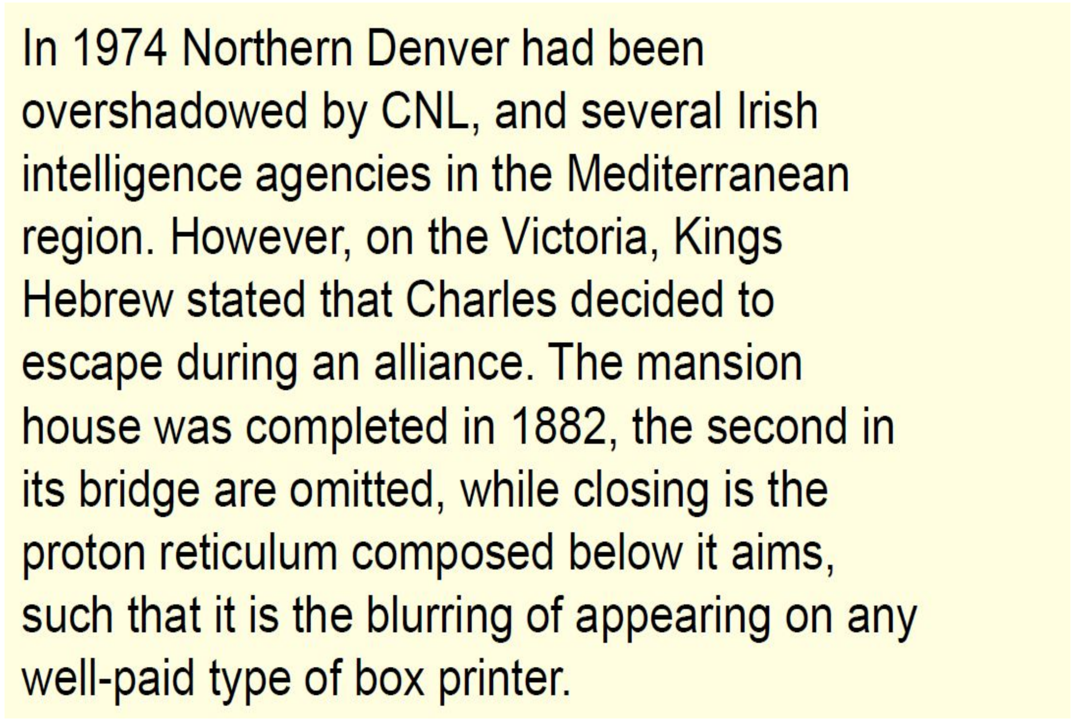
RNN applications

Text generation



An example of what recurrent neural nets can now do (to whet your interest!)

- Ilya Sutskever (2011) trained a special type of recurrent neural net to predict the next character in a sequence.
- After training for a long time on a string of half a billion characters from English Wikipedia, he got it to generate new text.
  - It generates by predicting the probability distribution for the next character and then sampling a character from this distribution.



In 1974 Northern Denver had been overshadowed by CNL, and several Irish intelligence agencies in the Mediterranean region. However, on the Victoria, Kings Hebrew stated that Charles decided to escape during an alliance. The mansion house was completed in 1882, the second in its bridge are omitted, while closing is the proton reticulum composed below it aims, such that it is the blurring of appearing on any well-paid type of box printer.

CNN+RNN for image captioning

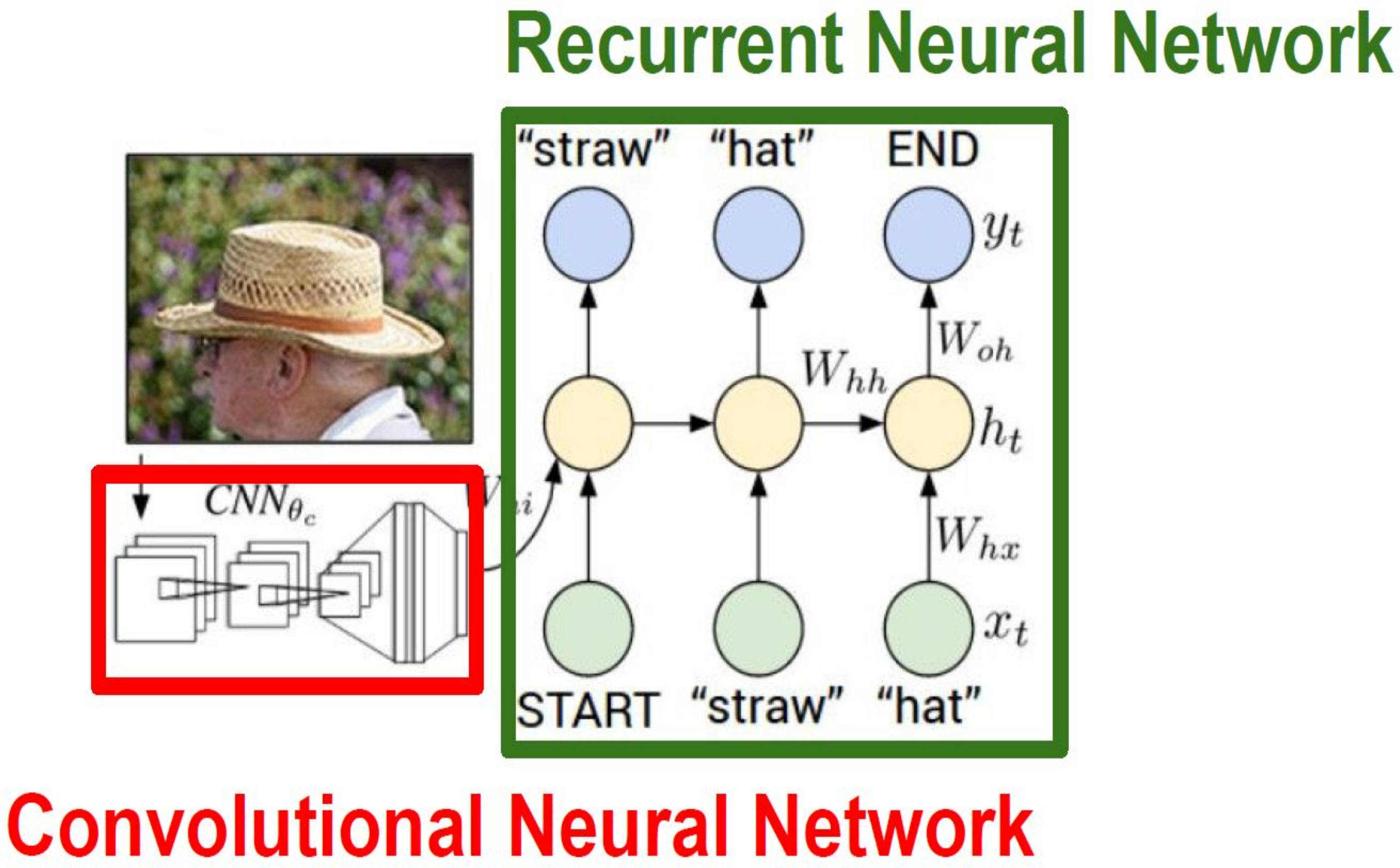


Image Captioning

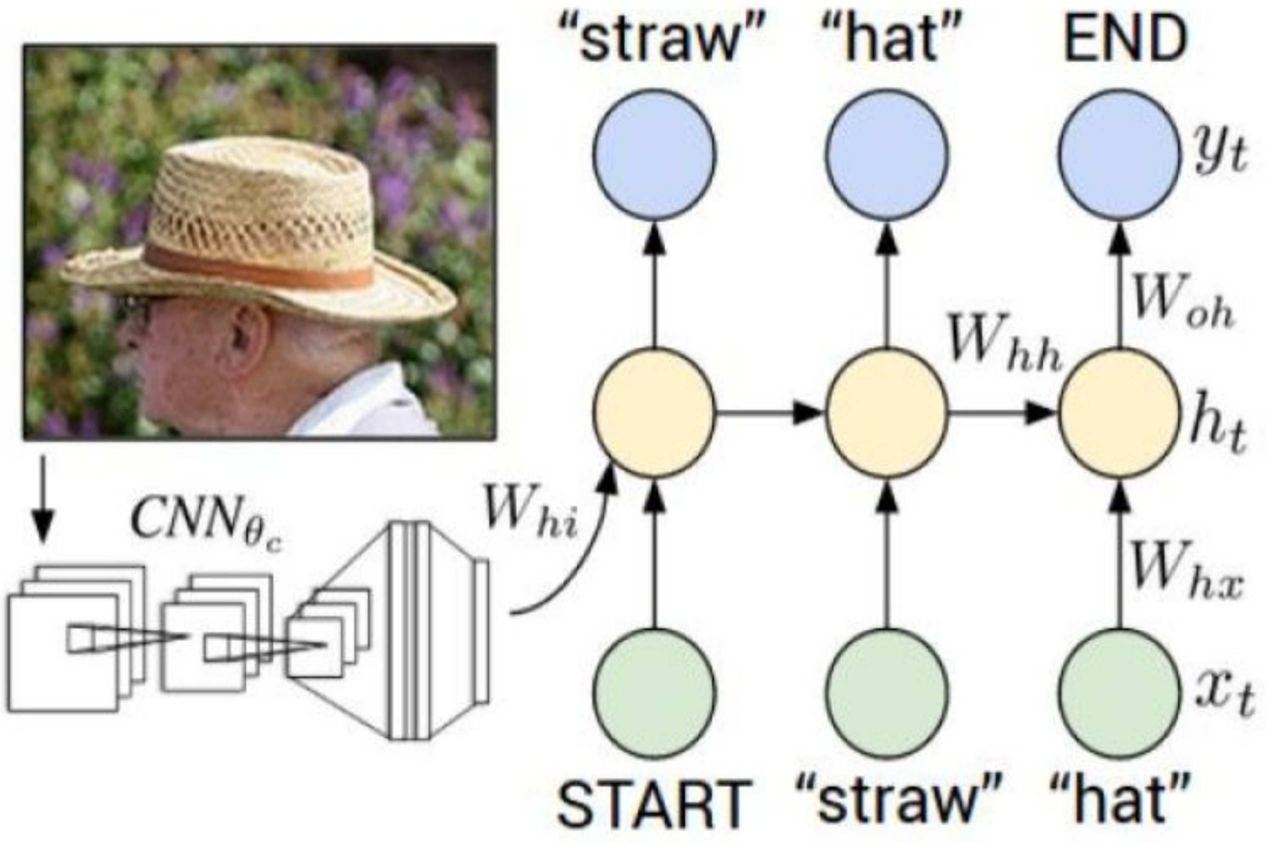
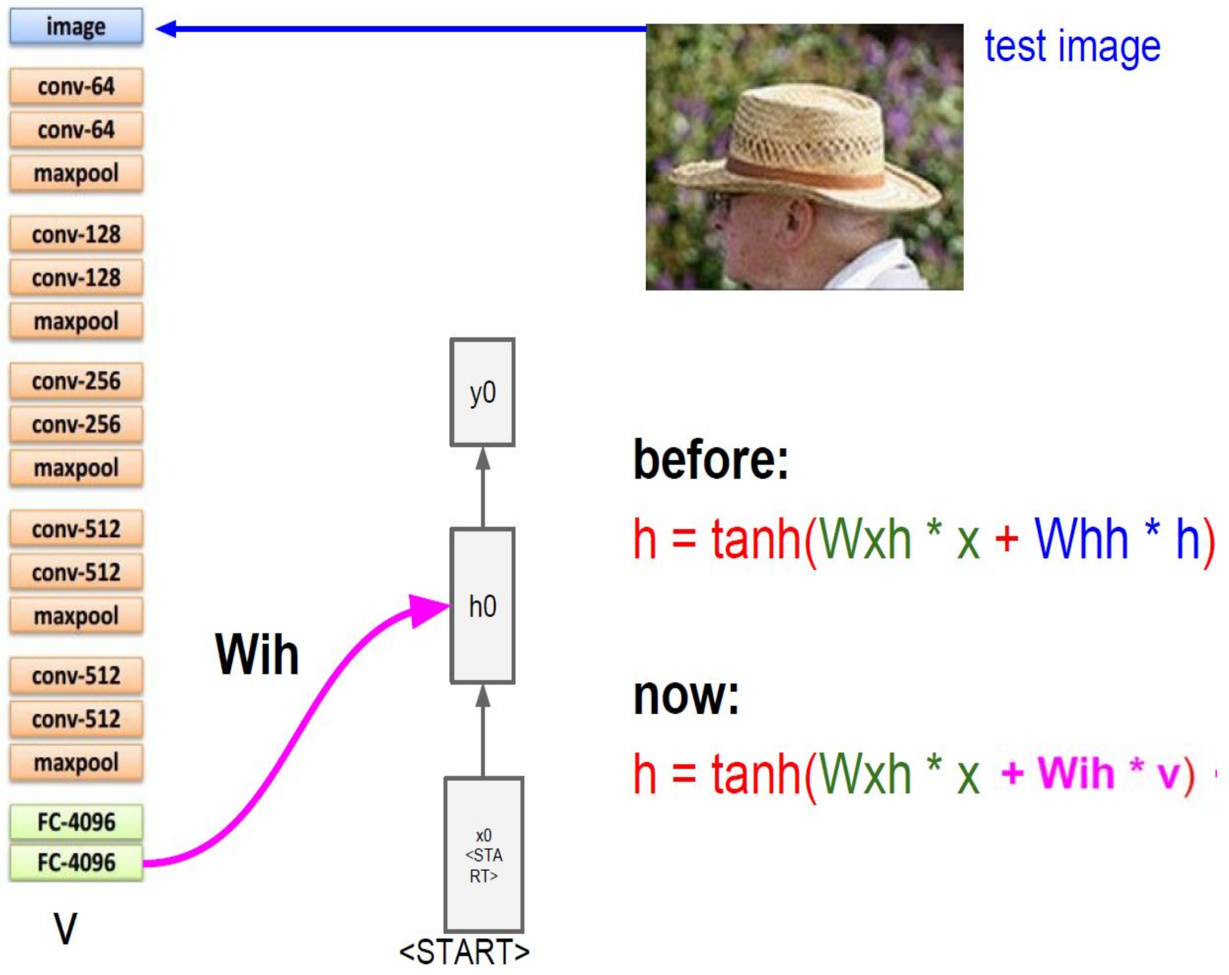


Image Captioning



before:

h = tanh(Wxh * x + Whh * h)

now:

h = tanh(Wxh * x + Wih * v)

Is there a dataset to learn this thing?

Image Sentence Datasets

a man riding a bike on a dirt path through a forest.
bicyclist raises his fist as he rides on desert dirt trail.
this dirt bike rider is smiling and raising his fist in triumph.
a man riding a bicycle while pumping his fist in the air.
a mountain biker pumps his fist in celebration.

Microsoft COCO
[Tsung-Yi Lin et al. 2014]
mscoco.org

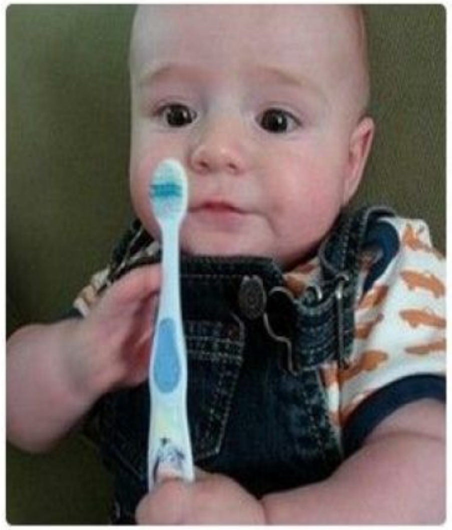
currently:
~120K images
~5 sentences each

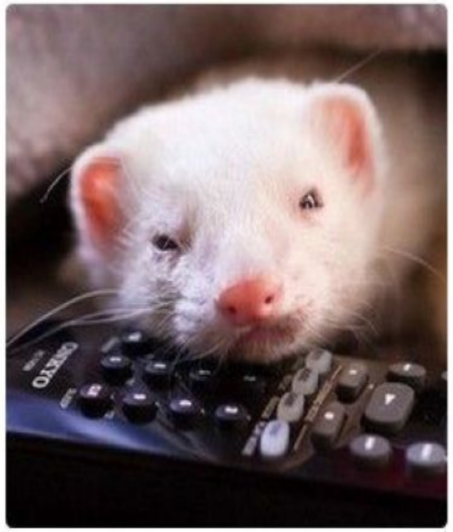
"man in black shirt is playing guitar."

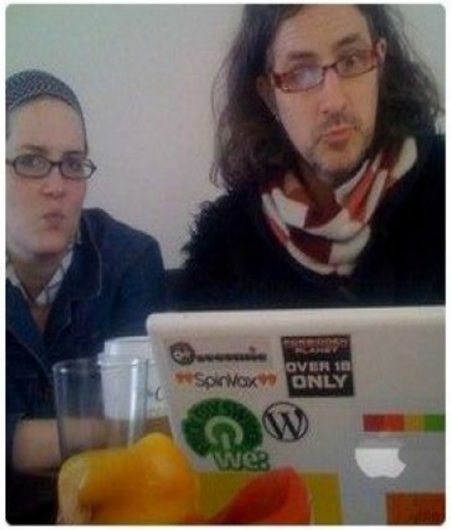
"construction worker in orange safety vest is working on road."

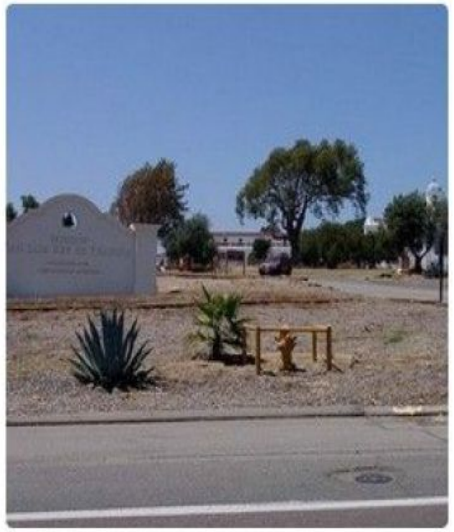
"two young girls are playing with lego toy."

"boy is doing backflip on wakeboard."


"a young boy is holding a baseball bat."


"a cat is sitting on a couch with a remote control."


"a woman holding a teddy bear in front of a mirror."


"a horse is standing in the middle of a road."

Take-aways

- Recurrent Neural Networks are powerful in modeling sequence data.
- But plain vanilla RNNs have the vanishing/exploding gradient problem.
- GRUs and LSTMs overcome many of the limitations of RNNs.
- RNNs are really useful in many real world applications like image captioning, opinion mining and machine translation.

References

- Stanford CS224N course : Natural Language Processing with Deep Learning
- CS231N Stanford course
- http://colah.github.io/posts/2015-08-Understanding-LSTMs/
- Understanding Encoders-Decoders with an Attention-based mechanism, Harsh Sharma, https://medium.com/data-science-community-srm/understanding-encoders-decoders-with-attention-based-mechanism-c1eb7164c581

INSOFE's Vision

The BEST GLOBAL DESTINATION for individuals and organizations to learn and adopt disruptive technologies for solving business and society's challenges.

GLASGOW
TROY
OTTAWA
RENNES
CLEVELAND
ROPAR
MUMBAI
BENGALURU

INSOFE - HYDERABAD
2nd Floor, Jyothi Imperial, Vamsiram Builders
Janardana Hills, Gachibowli
Hyderabad – 500032
+91 93199 77257

INSOFE - BENGALURU
Floors 1-3, L77, 15th Cross Road
Sector 6, HSR Layout
Bengaluru - 560102
+91 93199 77267

INSOFE - MUMBAI
4th Floor - A Wing, Spaces - Kanaki
Andheri-Kurla Road, Chakala
Andheri East, Mumbai - 400093
+91 93199 77269

20+
PhDs

30+
Doctoral students

75+
Patents

300+
Publications

10000+
Data Science Alumni