BOSTON UNIVERSITY

Deep Learning for Data Science
DS 542

Lecture 14
Recurrent Neural Networks

Slides originally by Thomas Gardos.

Images from [Understanding Deep Learning] unless otherwise cited.

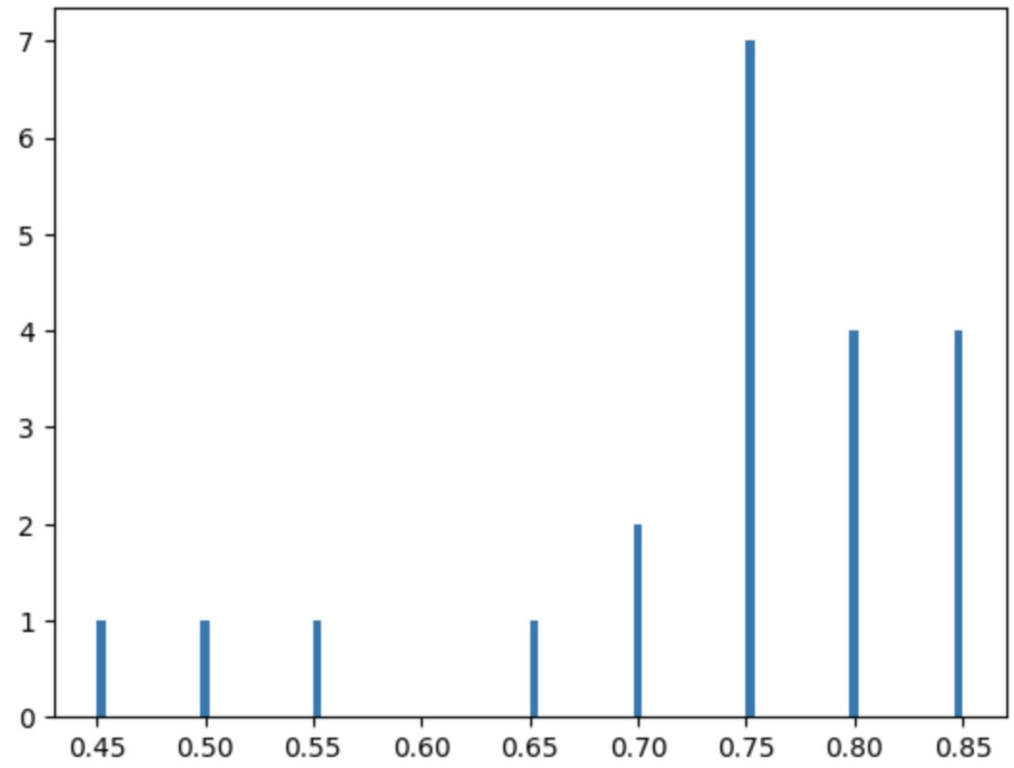
Today’s Agenda

- Midterm Recap
- Final Project
- Recurrent Neural Networks

Midterm

- Problem was designed to be challenging
  - Similar images with different target values
  - Bushes vs trees
  - Summer vs fall
  - Some odd (non-tree) images mixed in

Midterm - Test Accuracies (5% buckets)



Final Project - Project Categories

Choose a project that aligns with your interests and utilizes deep learning as part of the solution.

You may pick one of the three following categories of projects.

> **Application Project:** We expect most students will pick this category. Pick a problem or application that interests you. Consider whether there are suitable datasets available already or whether you will have to augment or create a dataset. Outcomes are expected to be implementation with an accompanying github repo and a report.
> **Algorithmic Project:** In this category, you will develop a new deep learning algorithm or substantively improve an existing one. One would typically benchmark against some well known dataset and show non-trivial improvement over prior work. Outcomes would typically be a short conference style article and an implementation with a github repo.
> **Theoretical Project:** Prove an interesting property of a new or existing learning algorithm. For a purely theoretical project, the output may only be a conference style report, but an implementation (and accompanying GitHub repo) may be appropriate as well.

It's possible that your project may blend more than one category.

Final Project - Ideas?

- Something new (to you)
- Something interesting to you
- Something where you know where to get data (not a lot of time here)

Last semester’s project presentations
https://mymedia.bu.edu/channel/channelid/340650712

Final Project - Timeline (updated since syllabus)

- 10/28 proposal draft due ← share in class for more feedback?
  - 10/30 proposal draft feedback
- 11/4 proposal final due
  - 11/6 proposal final feedback
- 11/20 midpoint check in
- 12/6 project due
- 12/9 project presentation in class

Topics

- Plain (vanilla) Recurrent Neural Network
- Problem of vanishing gradients
- Long Short-Term Memory
- Gradient Recurrent Unit
- Example applications
- Sequence to Sequence Learning

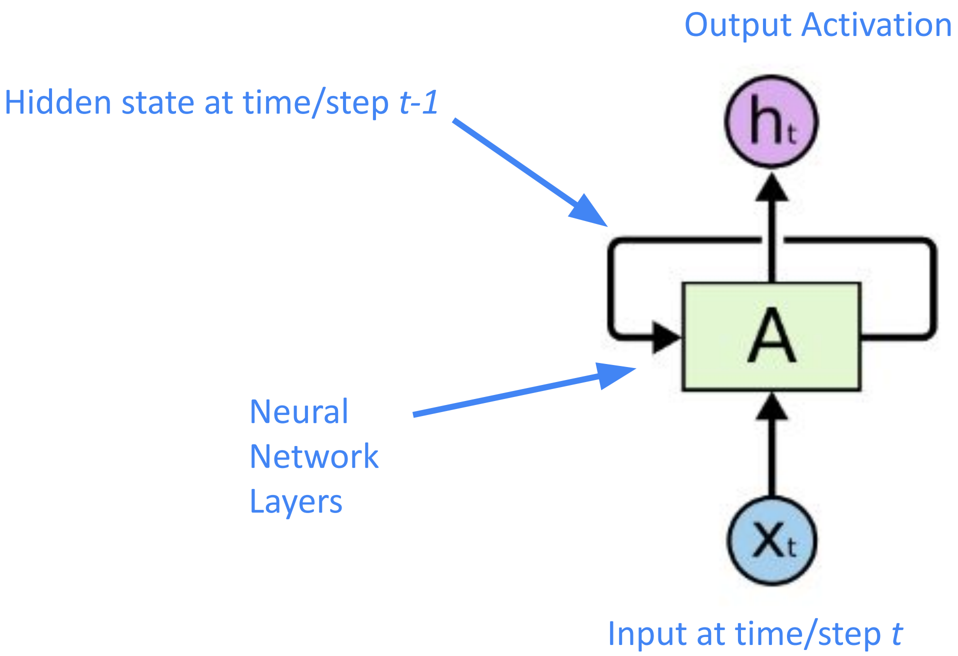
Motivation

- We want to process a sequence of data like text, digitized speech, video frames, etc.
- Want past samples to influence output from current sample

References

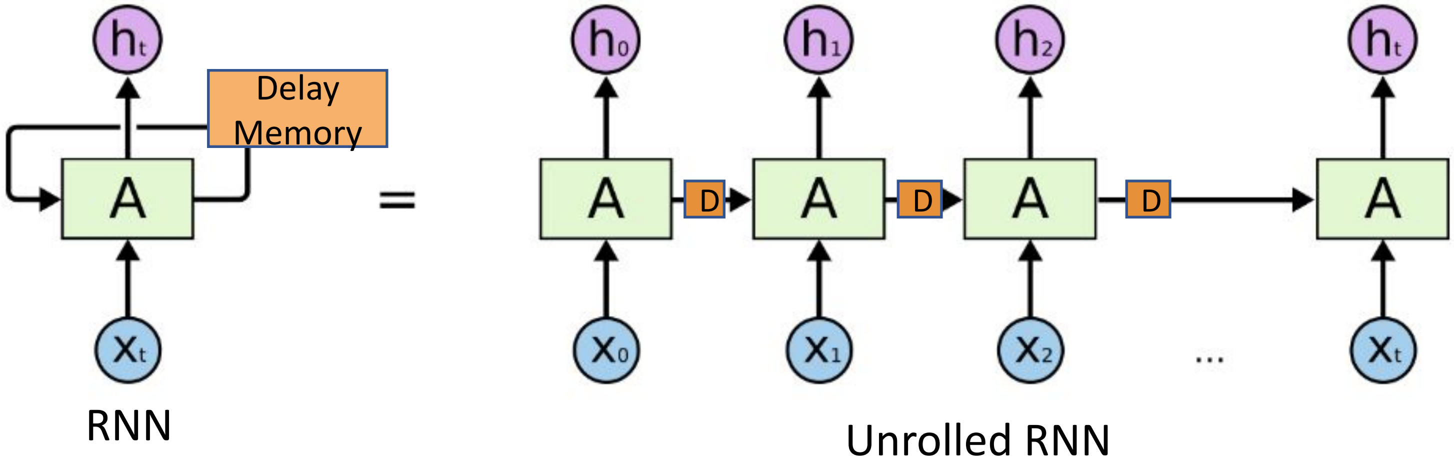
1. Understanding LSTMs, Colah's blog, 2015, https://colah.github.io/posts/2015-08-Understanding-LSTMs/
2. Speech and Language Processing. Daniel Jurafsky & James H. Martin. Draft of January 5, 2024. – Chapter 9, RNNs and LSTMs, https://web.stanford.edu/~jurafsky/slpdraft/9.pdf
3. The Unreasonable Effectiveness of LSTMs, Andrej Karpathy, 2015, https://karpathy.github.io/2015/05/21/rnn-effectiveness/

Recurrent Neural Network



Understanding LSTM Networks, C. Colah Blog Post

Unrolled view over time

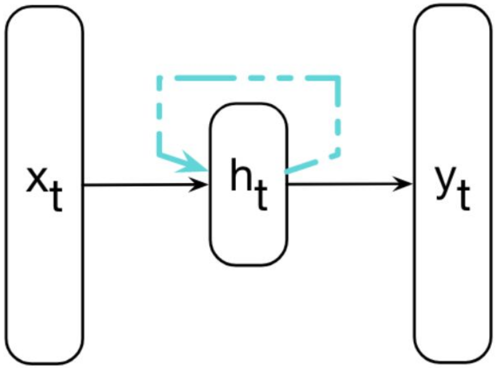


In this case you are emitting an output for every input token

Unrolled network is fed sequentially (not all at once)

Understanding LSTM Networks, C. Colah Blog Post

Recurrent Neural Network – Weight Matrices



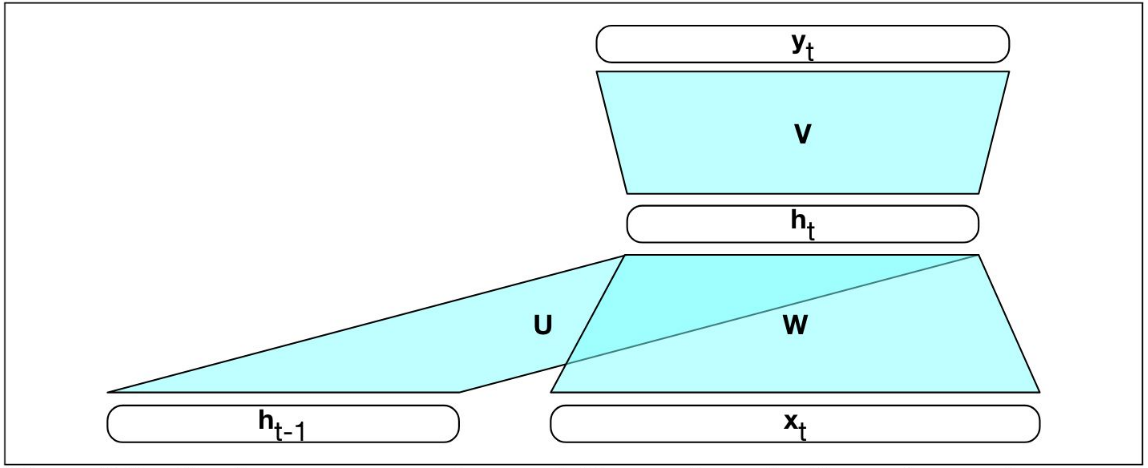


Figure 9.2 Simple recurrent neural network illustrated as a feedforward network.

$$
\begin{aligned}
\mathbf{h}_t &= g(\mathbf{U}\mathbf{h}_{t-1} + \mathbf{W}\mathbf{x}_t) \\
\mathbf{y}_t &= f(\mathbf{V}\mathbf{h}_t)
\end{aligned}
$$

S&LP, Jurafsky & Martin

Unrolled Network

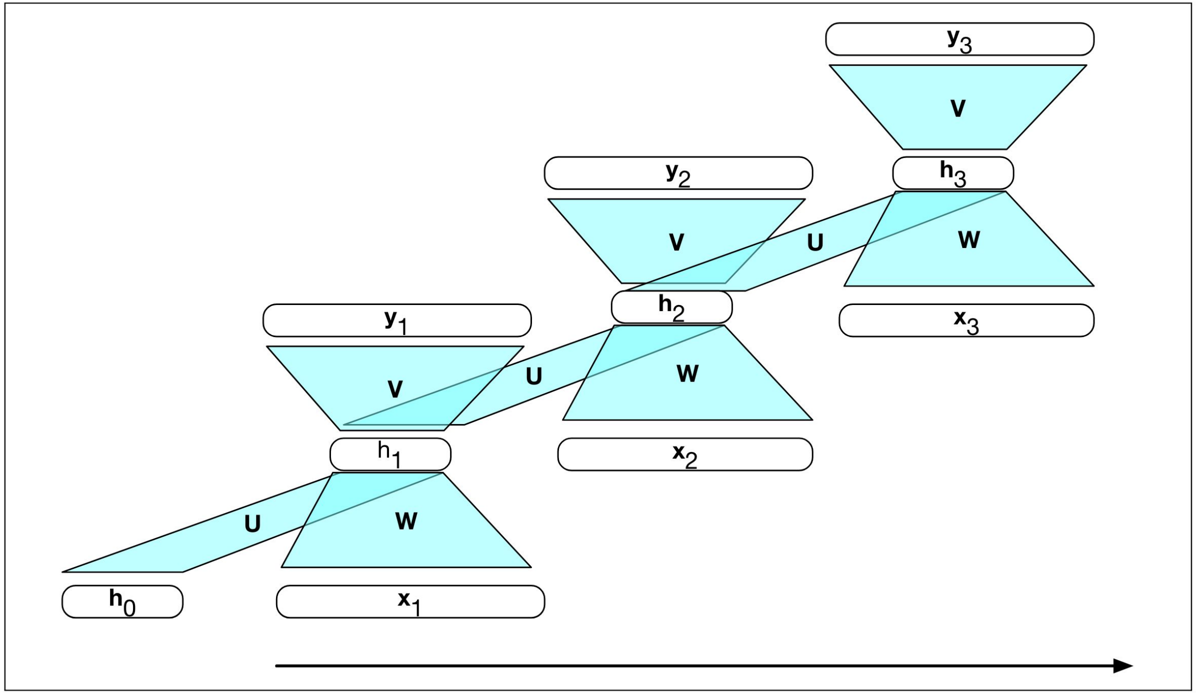
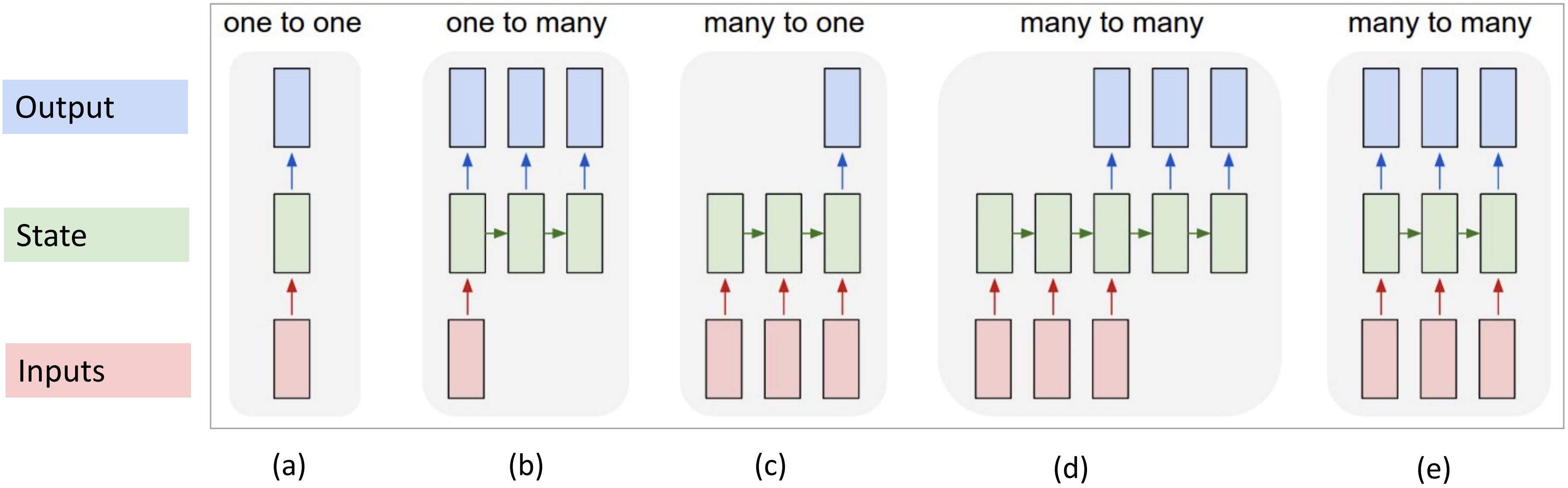


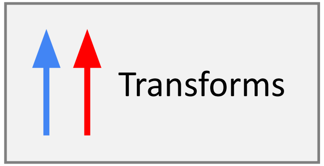
Figure 9.4 A simple recurrent neural network shown unrolled in time. Network layers are recalculated for each time step, while the weights **U**, **V** and **W** are shared across all time steps.

The weights, U, W and V, are the same at each time step. Only the inputs (x_t, h_{t-1}) change.

S&LP, Jurafsky & Martin

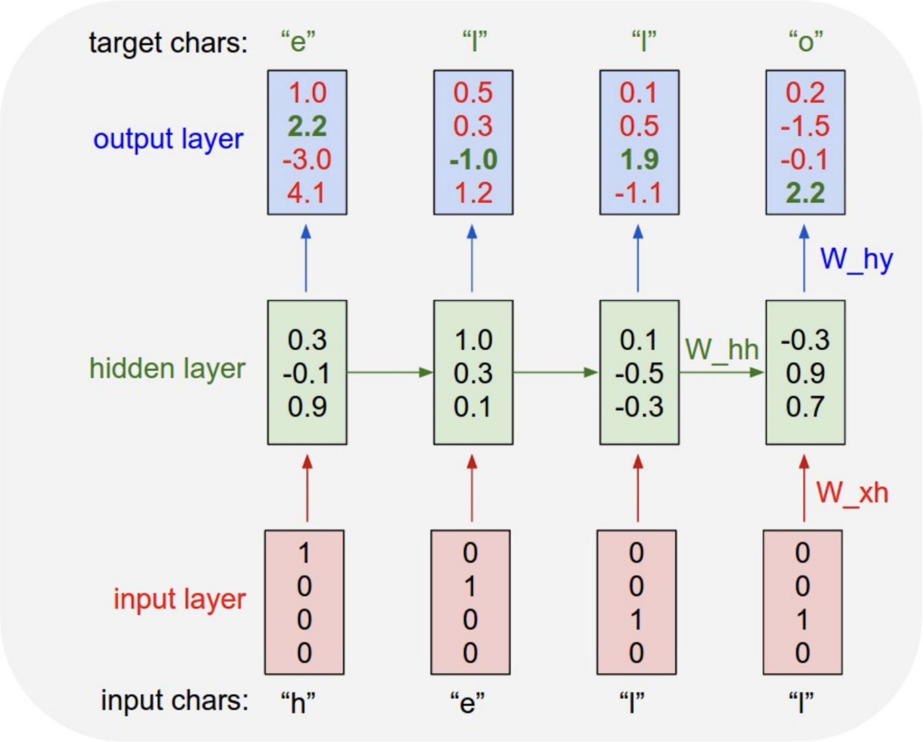
Different RNN configurations





(a) Regular Feed Forward Network
(b) E.g. image captioning – input 1 image, outputs sequence of words
(c) E.g. sentiment analysis from string of words or characters
(d) E.g. machine translation such as English to French
(e) Synced sequence input and output, e.g. video frame-by-frame action classification or text generation

RNN next letter prediction example



Output is probability or likelihood over the vocabulary

Hidden layer encodes history, here e.g. length 3.

One-hot encoded input of vocabulary length, e.g. ('h', 'e', 'l', 'o')

Training an RNN

```
import torch.nn as nn

class RNN(nn.Module):
    def __init__(self, input_size, hidden_size, output_size):
        super().__init__()

        self.hidden_size = hidden_size

        self.i2h = nn.Linear(input_size + hidden_size, hidden_size)
        self.h2o = nn.Linear(hidden_size, output_size)
        self.softmax = nn.LogSoftmax(dim=1)

    def forward(self, input, hidden):
        combined = torch.cat((input, hidden), 1)
        hidden = self.i2h(combined)
        output = self.h2o(hidden)
        output = self.softmax(output)
        return output, hidden

    def initHidden(self):
        return torch.zeros(1, self.hidden_size)
```

Simple feed forward network.

History and recurrence managed outside the model

https://pytorch.org/tutorials/intermediate/char_rnn_classification_tutorial.html

Training an RNN – Same Backprop as FFN

```
# If you set this too high, it might explode. If too low, it might not learn
learning_rate = 0.005

def train(category_tensor, line_tensor):
    hidden = rnn.initHidden()

    rnn.zero_grad()

    for i in range(line_tensor.size()[0]):
        output, hidden = rnn(line_tensor[i], hidden)

    loss = criterion(output, category_tensor)
    loss.backward()

    # Add parameters' gradients to their values, multiplied by learning rate
    for p in rnn.parameters():
        p.data.add_(p.grad.data, alpha=-learning_rate) # in-place addition

    return output, loss.item()
```

Managing recurrence.

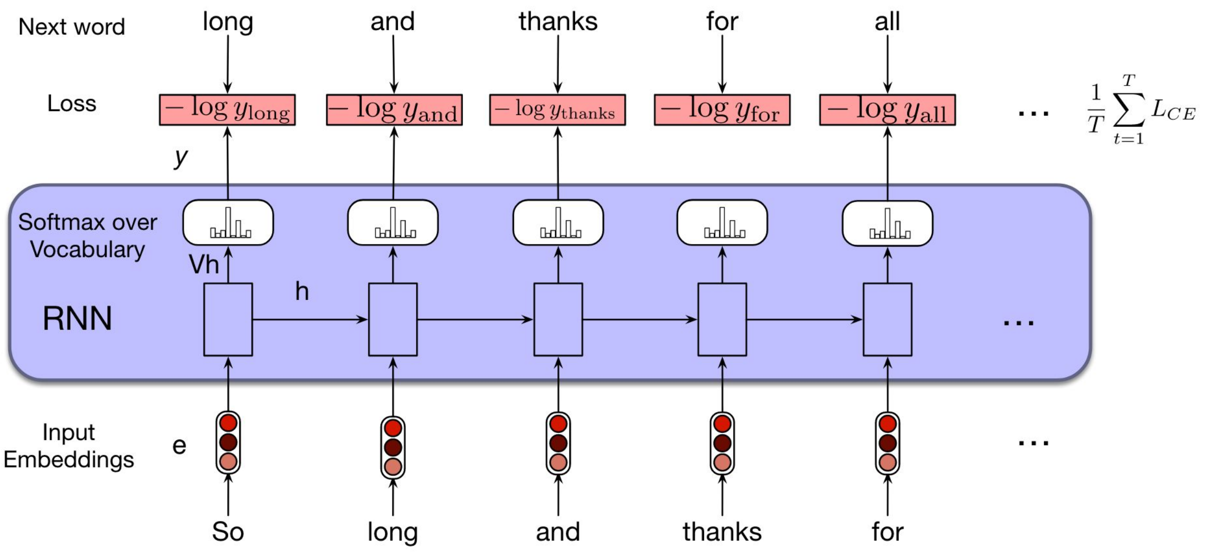
Single output for classification in this case.

Needs to save a lot for backpropagation.

"Back propagation Through Time", e.g. BPTT

https://pytorch.org/tutorials/intermediate/char_rnn_classification_tutorial.html

Loss Calculation for Sequence



$$L_{CE} = -\sum_{w \in V} \mathbf{y}_t[w] \log \hat{\mathbf{y}}_t[w]$$

$$L_{CE}(\hat{\mathbf{y}}_t, \mathbf{y}_t) = -\log \hat{\mathbf{y}}_t[w_{t+1}]$$

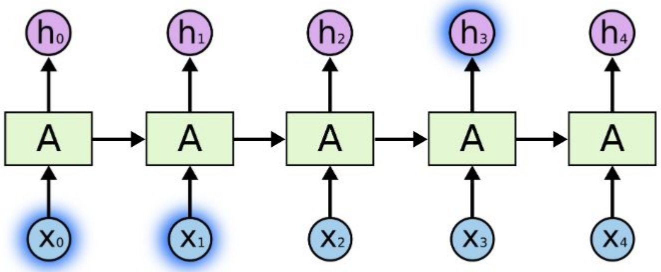
The probability that the model assigns to the next word in the training sequence.

S&LP, Jurafsky & Martin

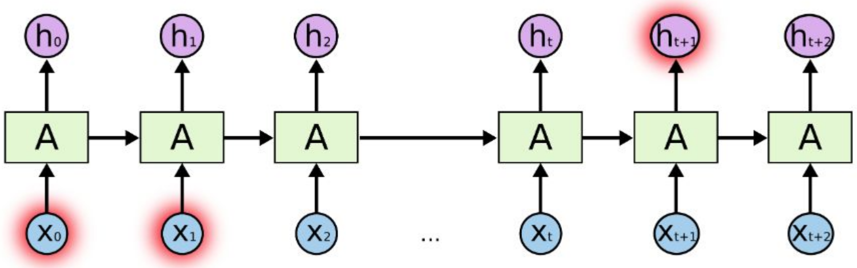
Topics

- Plain (vanilla) Recurrent Neural Network
- Problem of vanishing gradients
- Long Short-Term Memory
- Gradient Recurrent Unit
- Example applications
- Sequence to Sequence Learning

Problem of vanishing gradients



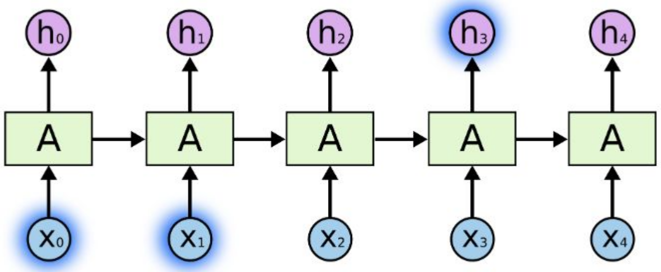
Tokens from earlier in the sequence can influence the current output



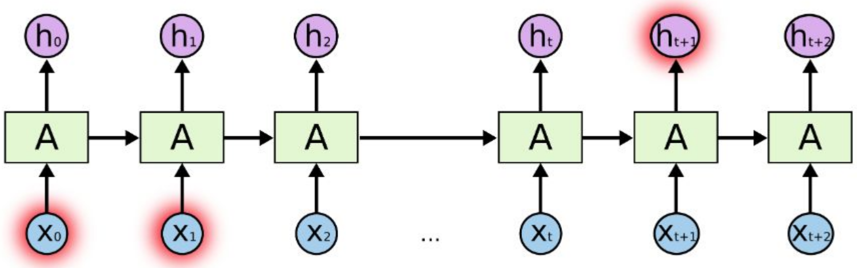
But for plain RNNs, the influence can reduce rapidly the further the sequence difference

Understanding LSTM Networks, C. Colah Blog Post

Why not exploding gradients?



Tokens from earlier in the sequence can influence the current output



But for plain RNNs, the influence can reduce rapidly the further the sequence difference

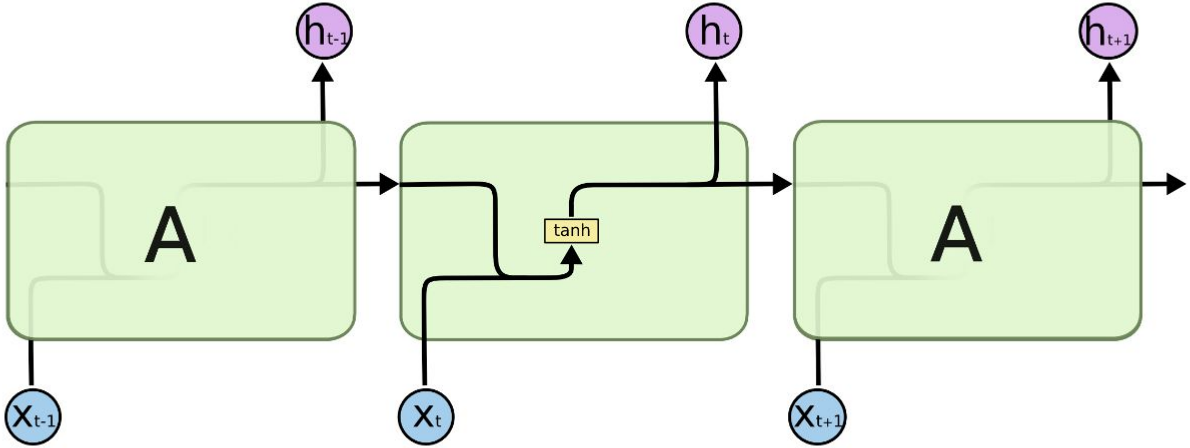
Understanding LSTM Networks, C. Colah Blog Post

Topics

- Plain (vanilla) Recurrent Neural Network
- Problem of vanish gradients
- Long Short-Term Memory
- Gradient Recurrent Unit
- Example applications
- Sequence to Sequence Learning

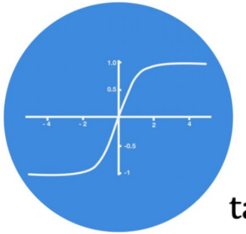
Redrawing RNN

Redraw the RNN in slightly more detail



<u>Neural Network Layer</u>:

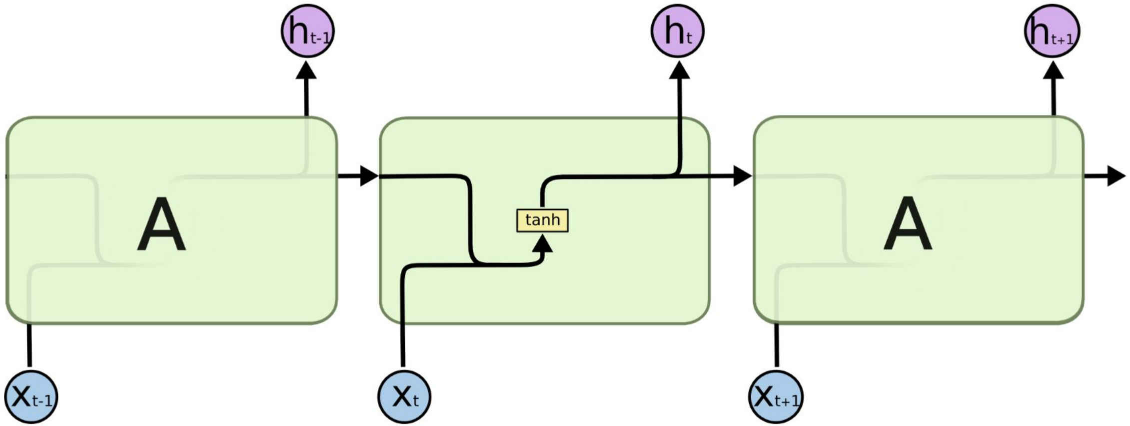
$$h_t = tanh(W \cdot [h_{t-1}, x_t] + b)$$



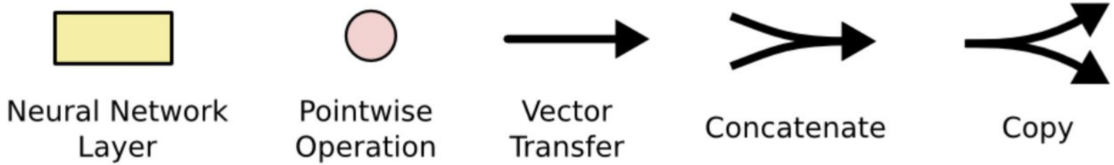
tanh(), $\mathbb{R} \rightarrow [-1,1]$

Understanding LSTM Networks, C. Colah Blog Post

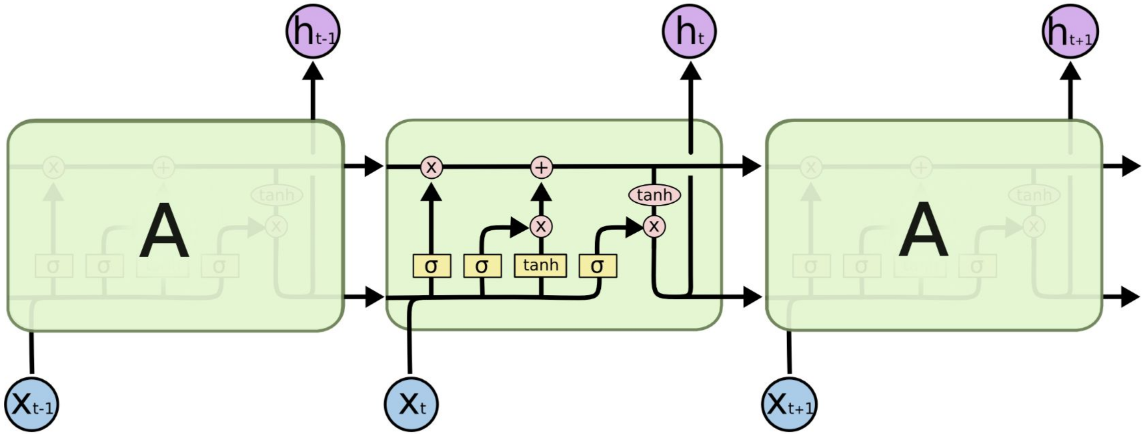
First redraw RNN



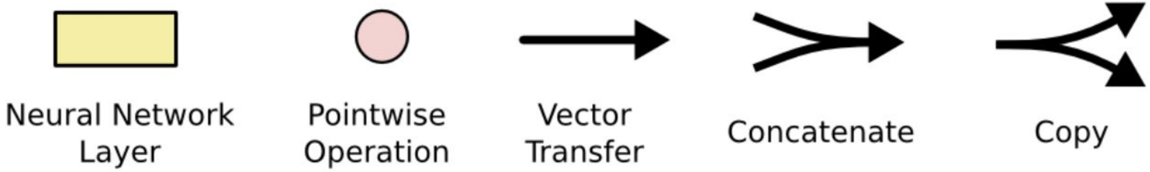
The repeating module in a standard RNN contains a single layer.



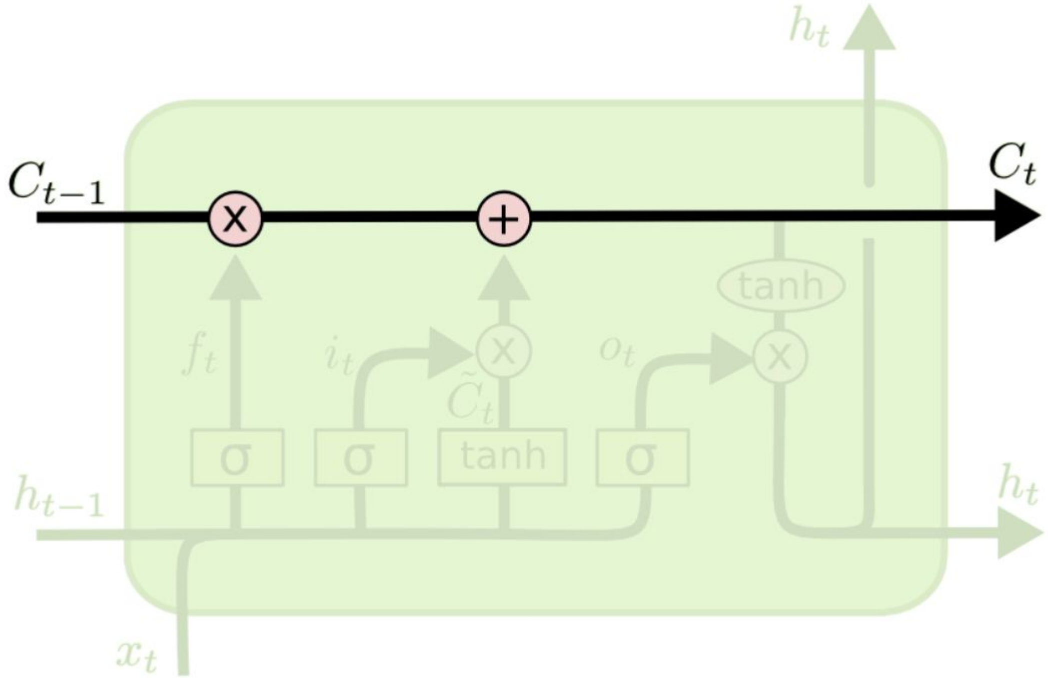
Long Short Term Memory (LSTM)

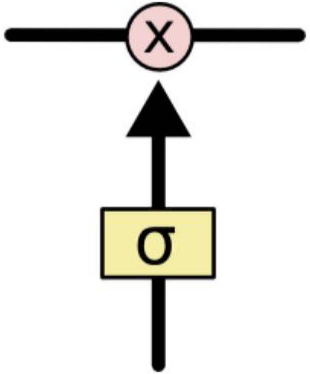


The repeating module in an LSTM contains four interacting layers.

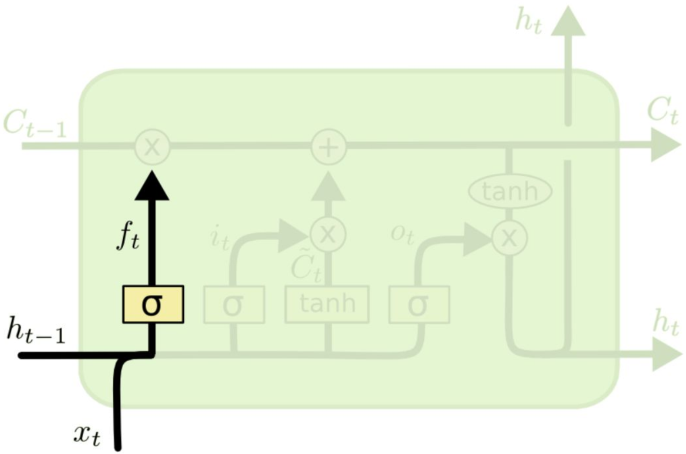


LSTM – Cell State





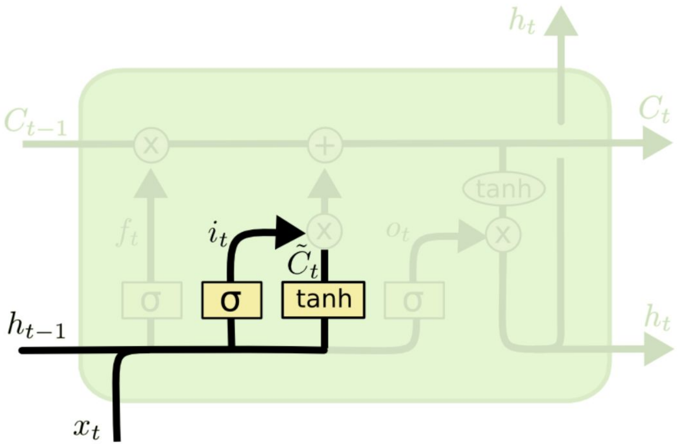
LSTM -- Forgetting Gate



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

Decides what part of cell state to suppress

LSTM – Cell state update



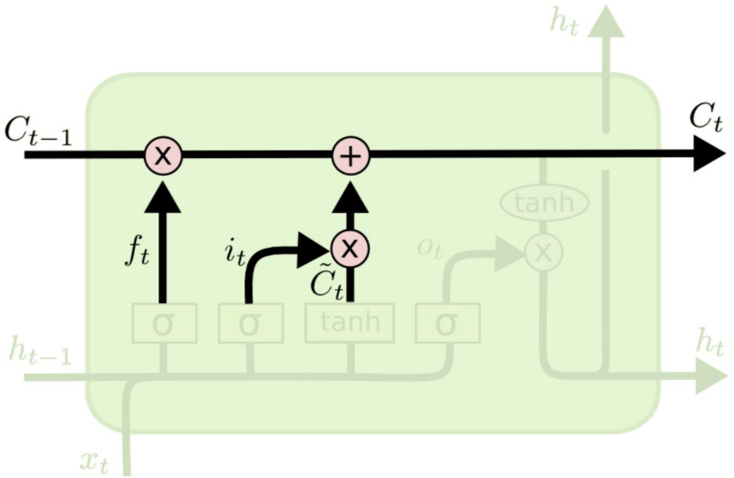
Input Gate Layer

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

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

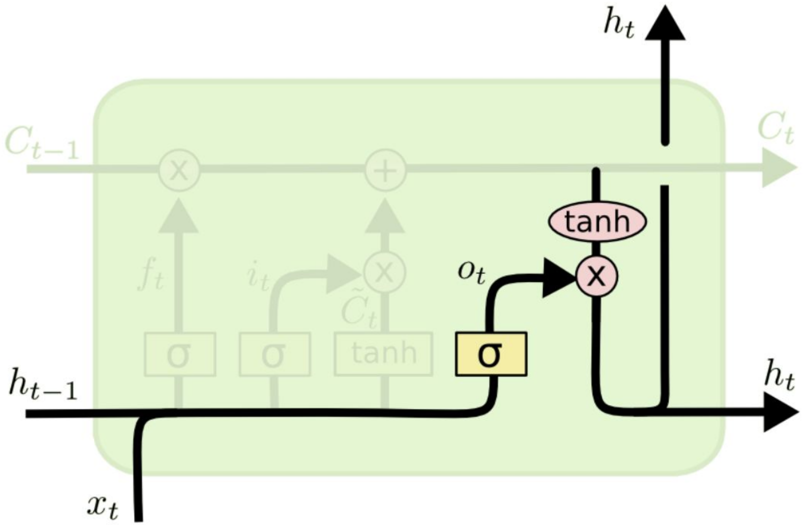
Candidate Cell State

LSTM – Apply changes to cell state



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

LSTM – Output and Hidden State Update



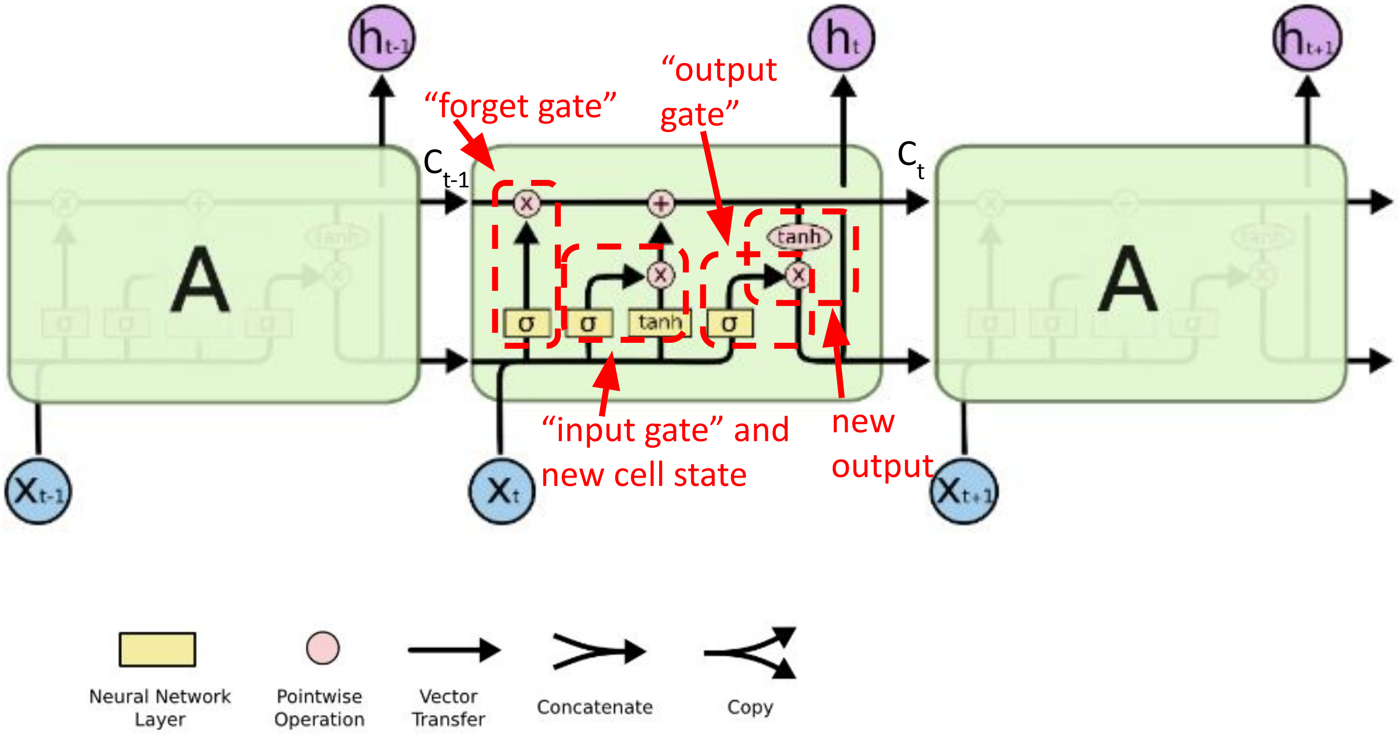
Output Gate

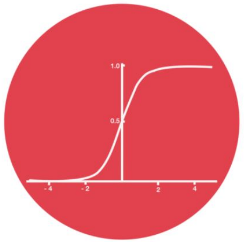
$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

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

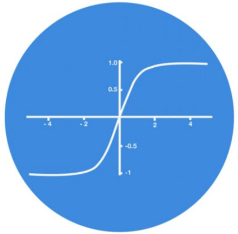
Next hidden state and output

Long Short Term Memory (LSTM)





σ – Sigmoid, $\mathbb{R} \rightarrow [0,1]$



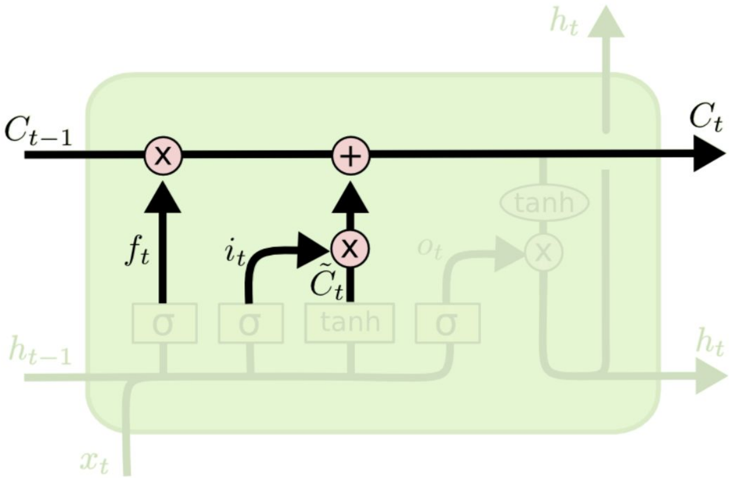
$\tanh()$, $\mathbb{R} \rightarrow [-1,1]$

Neural Network Layer:

$$out_t = activation(W \cdot [h_{t-1}, x_t] + b)$$

Understanding LSTM Networks, C. Colah Blog Post
Illustrated Guide to LSTM's and GRU's, M. Phi Blog Post

LSTM – What does this look like?



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

A little bit like a residual network? Similar motivation to have clear gradient path…

Topics

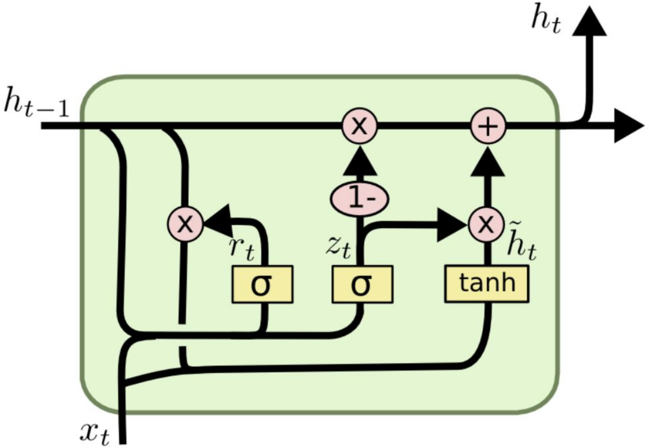
- Plain (vanilla) Recurrent Neural Network
- Problem of vanish gradients
- Long Short-Term Memory
- Gradient Recurrent Unit
- Example applications
- Sequence to Sequence Learning

Gradient Recurrent Unit

- Combines the forget and input gates into a single “update gate.”
- Merges the cell state and hidden state

The resulting model is simpler than standard LSTM models.

Results are mixed.



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

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

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

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

K. Cho *et al.*, “Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation.” arXiv, Sep. 02, 2014. doi: 10.48550/arXiv.1406.1078.

Topics

- Plain (vanilla) Recurrent Neural Network
- Problem of vanish gradients
- Long Short-Term Memory
- Gradient Recurrent Unit
- Example applications
- Sequence to Sequence Learning

Andrej Karpathy blog About

The Unreasonable Effectiveness of Recurrent Neural Networks

May 21, 2015

Trained on complete works of Shakespeare

3-layer RNN with 512 hidden nodes on each layer.

Trained for a few hours on a GPU

```
PANDARUS:
Alas, I think he shall be come approached and the day
When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:
They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:
Well, your wit is in the care of side and that.

Second Lord:
They would be ruled after this chamber, and
my fair nues begun out of the fact, to be conveyed,
Whose noble souls I'll have the heart of the wars.

Clown:
Come, sir, I will make did behold your worship.

VIOLA:
I'll drink it.
```

Andrej Karpathy blog

About

The Unreasonable Effectiveness of Recurrent Neural Networks

May 21, 2015

Trained and Generated Wikipedia Content

```
Naturalism and decision for the majority of Arab countries' capitalide was grounded
by the Irish language by [[John Clair]], [[An Imperial Japanese Revolt]], associated
with Guangzham's sovereignty. His generals were the powerful ruler of the Portugal
in the [[Protestant Immineners]], which could be said to be directly in Cantonese
Communication, which followed a ceremony and set inspired prison, training. The
emperor travelled back to [[Antioch, Perth, October 25|21]] to note, the Kingdom
of Costa Rica, unsuccessful fashioned the [[Thrales]], [[Cynth's Dajoard]], known
in western [[Scotland]], near Italy to the conquest of India with the conflict.
Copyright was the succession of independence in the slop of Syrian influence that
was a famous German movement based on a more popular servicious, non-doctrinal
and sexual power post. Many governments recognize the military housing of the
[[Civil Liberalization and Infantry Resolution 265 National Party in Hungary]],
that is sympathetic to be to the [[Punjab Resolution]]
(PJS)[http://www.humah.yahoo.com/guardian.
cfm/7754800786d17551963s89.htm Official economics Adjoi
was swear to advance to the resources for those Sociali
was starting to signing a major tripad of aid exile.]]
```

```
{ { cite journal | id=Cerling Nonforest Department|format=Newlymeslated|none } }
''www.e-complete''.

'''See also''': [[List of ethical consent processing]]

== See also ==
*[[Iender dome of the ED]]
*[[Anti-autism]]

===[[Religion|Religion]]===
*[[French Writings]]
*[[Maria]]
*[[Revelation]]
*[[Mount Agamul]]

== External links==
* [http://www.biblegateway.nih.gov/entrepre/ Website of the World Festival. The labour

nstitution of the Netherlands and Hispanic Competition
```

Structured Markdown

```
<page>
  <title>Antichrist</title>
  <id>865</id>
  <revision>
    <id>15900676</id>
    <timestamp>2002-08-03T18:14:12Z</timestamp>
    <contributor>
      <username>Paris</username>
      <id>23</id>
    </contributor>
    <minor />
    <comment>Automated conversion</comment>
    <text xml:space="preserve">#REDIRECT [[Christianity]]</text>
  </revision>
</page>
```

Valid XML

Counterpoint: N-gram stats are similarly good?

Replies:

- Yoav Goldberg compared these RNN results to n-gram maximum likelihood (counting) baseline

This was a good reply, but unfortunately the linked notebook is no longer available.

- Google made a dataset of “n-grams” available in 2006
  - https://research.google/blog/all-our-n-gram-are-belong-to-you/

Counterpoint: N-gram stats are similarly good?

“We processed 1,024,908,267,229 words of running text and are publishing the counts for all 1,176,470,663 five-word sequences that appear at least 40 times. There are 13,588,391 unique words, after discarding words that appear less than 200 times.”

- You can use longest prefix matches on this data to get similar quality results.
- Very limited pattern matching - does not have equivalent of find and replace.

Deep Visual-Semantic Alignments for Generating Image Descriptions

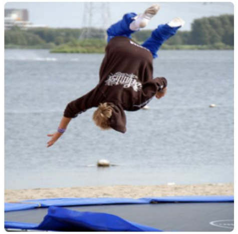
Multimodal Recurrent Neural Network

Our Multimodal Recurrent Neural Architecture generates sentence descriptions from images. Below are a few examples of generated sentences:

"man in black shirt is playing guitar."

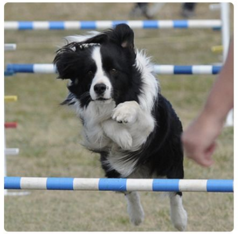
"construction worker in orange safety vest is working on road."

"two young girls are playing with lego toy."



"boy is doing backflip on wakeboard."

"girl in pink dress is jumping in air."



"black and white dog jumps over bar."

"young girl in pink shirt is swinging on swing."

"man in blue wetsuit is surfing on wave."

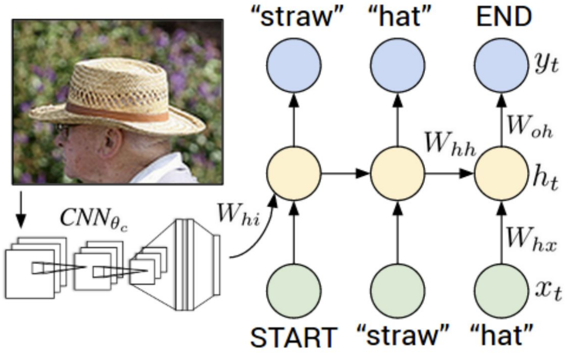


Figure 4. Diagram of our multimodal Recurrent Neural Network generative model. The RNN takes a word, the context from previous time steps and defines a distribution over the next word in the sentence. The RNN is conditioned on the image information at the first time step. START and END are special tokens.

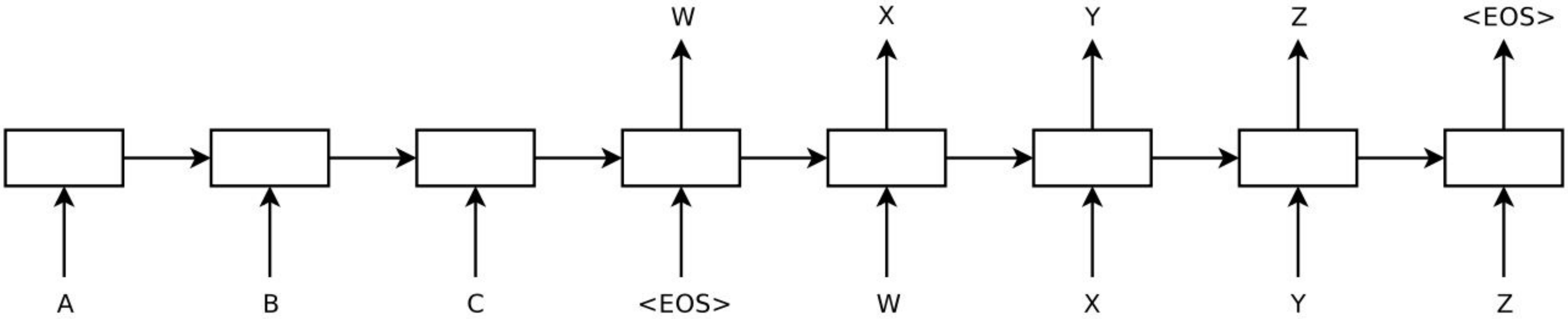
What else can be done with an RNN?

- Any kind of text analysis
  - Sentiment analysis was common
- Image analysis
  - Pixel by pixel or chunks at a time
  - Classification?

But these were weak / had frequent trouble…

Sequence to Sequence Learning

Powerful LSTMs (2014) and hints of their limits…



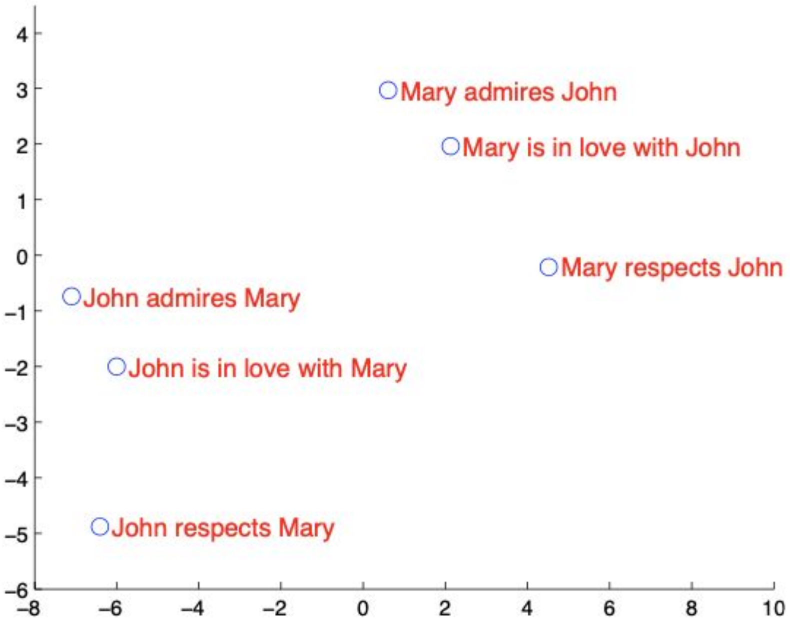
https://proceedings.neurips.cc/paper_files/paper/2014/hash/a14ac55a4f27472c5d894ec1c3c743d2-Abstract.html

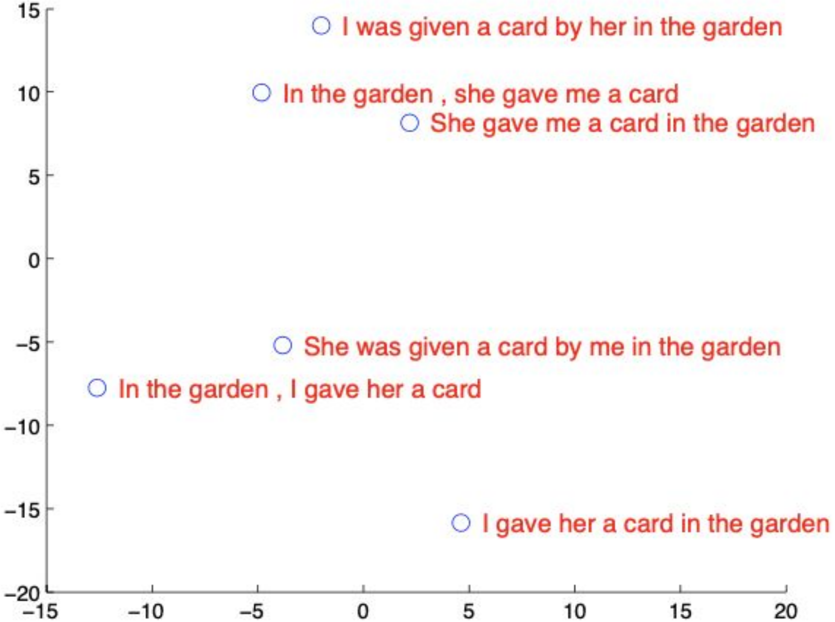
Sequence to Sequence Applications

- Text completion (generalizing previous examples)
- Machine translation
- Various other string / sentence manipulation
- Question answering

Sequence to Sequence State

2D PCA of Internal State





Sequence to Sequence Learning

“One of the attractive features of our model is its ability to turn a sequence of words into a vector of fixed dimensionality.”

Sequence to Sequence Learning

“Finally, we found that reversing the order of the words in all source sentences (but not target sentences) improved the LSTM's performance markedly, because doing so introduced many short term dependencies between the source and the target sentence which made the optimization problem easier.”

The Unreasonable Effectiveness of Recurrent Neural Networks

May 21, 2015

...

Inductive Reasoning, Memories and Attention.

...

The first convincing example of moving towards these directions was developed in DeepMind's Neural Turing Machines paper. This paper sketched a path towards models that can perform read/write operations between large, external memory arrays and a smaller set of memory registers (think of these as our working memory) where the computation happens. Crucially, the NTM paper also featured very interesting memory addressing mechanisms that were implemented with a (soft, and fully-differentiable) attention model. The concept of **soft attention** has turned out to be a powerful modeling feature and was also featured in Neural Machine Translation by Jointly Learning to Align and Translate for Machine Translation and Memory Networks for (toy) Question Answering. In fact, I'd go as far as to say that

> *The concept of **attention** is the most interesting recent architectural innovation in neural networks.*

A Brief History of Transformers

2000

Yoshua Bengio*

2014

Ilya Sutskever*

2014

Dzmitry Bahdanau*

2017

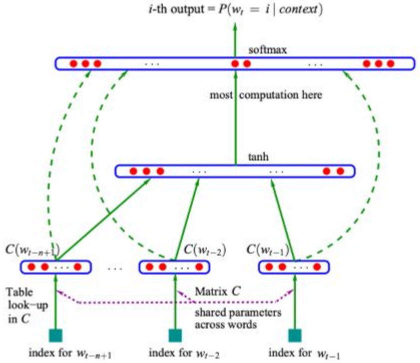
A Team at Google

Use LSTMs

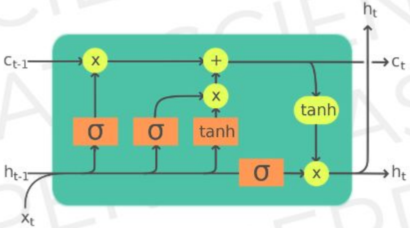
Add Attention

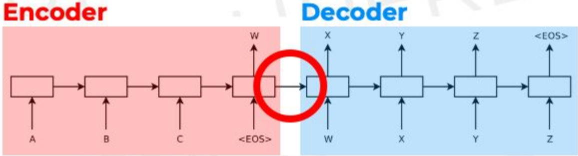
Remove LSTMs

A Neural Probabilistic Language Model

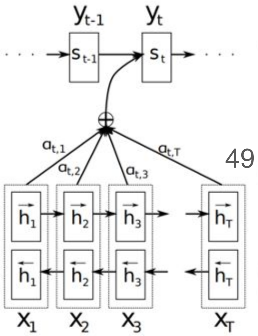


Seq-to-Seq Learning with Neural Networks

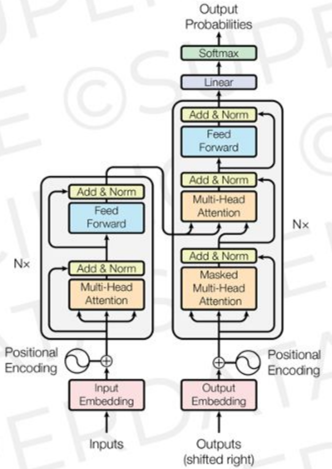




Neural Machine Translation by Jointly Learning to Align and Translate



Attention is all you need



*And others; Chronological analysis inspired by Andrej Karpathy's lecture, youtube.com/watch?v=XfpMkf4rD6E

Very Recent Update: Were RNNs All We Needed?

"The scalability limitations of Transformers regarding sequence length have renewed interest in recurrent sequence models that are parallelizable during training. As a result, many novel recurrent architectures, such as S4, Mamba, and Aaren, have been proposed that achieve comparable performance. In this work, we revisit traditional recurrent neural networks (RNNs) from over a decade ago: LSTMs (1997) and GRUs (2014). While these models were slow due to requiring to backpropagate through time (BPTT), we show that by removing their hidden state dependencies from their input, forget, and update gates, LSTMs and GRUs no longer need to BPTT and can be efficiently trained in parallel. Building on this, we introduce minimal versions (minLSTMs and minGRUs) that (1) use significantly fewer parameters than their traditional counterparts and (2) are fully parallelizable during training (175x faster for a sequence of length 512). Lastly, we show that these stripped-down versions of decade-old RNNs match the empirical performance of recent sequence models."

https://arxiv.org/abs/2410.01201

Feedback?