Lecture 9 - Networks for Sequential Data RNNs & LSTMs

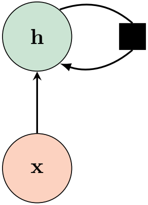
DD2424

April 27, 2017

Recurrent Neural Networks (RNNs)

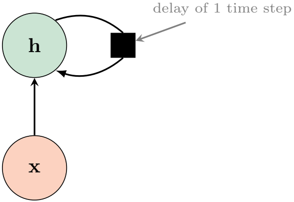
- RNNs are a family of networks for processing sequential data.
- A RNN applies the same function recursively when traversing network's graph structure.
- RNN encodes a sequence $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_\tau$ into fixed length hidden vector $\mathbf{h}_\tau$.
- The size of $\mathbf{h}_\tau$ is independent of τ.
- Amazingly flexible and powerful high-level architecture.

RNN with no outputs



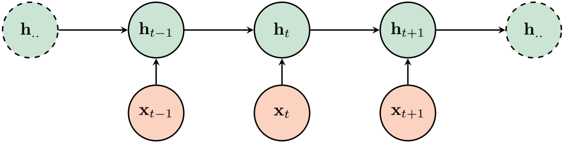
- Graph displays processing of information for each time step.
- Information from input $\mathbf{x}$ is incorporated into state $\mathbf{h}$.

RNN with no outputs



- Graph displays processing of information for each time step.
- Information from input $\mathbf{x}$ is incorporated into state $\mathbf{h}$.
- State $\mathbf{h}$ is passed forward in time.

RNN with no outputs



Unrolled visualization of the RNN

RNN: How hidden states generated

- Most **recurrent neural networks** have a function f

$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_t; \boldsymbol{\theta})$$

that defines their hidden state over time where

  - $\mathbf{h}_t$ is the hidden state at time t (a vector)
  - $\mathbf{x}_t$ is the input vector at time t
  - $\boldsymbol{\theta}$ is the parameters of f.

- $\boldsymbol{\theta}$ remains constant as t changes.

Apply the **same function** with the **same parameter** values at **each iteration**.

RNN: How hidden states generated

- Most **recurrent neural networks** have a function f

$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_t; \boldsymbol{\theta})$$

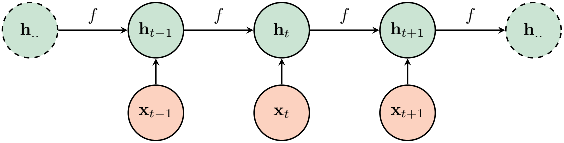
  that defines their hidden state over time where

  - $\mathbf{h}_t$ is the hidden state at time t (a vector)
  - $\mathbf{x}_t$ is the input vector at time t
  - $\boldsymbol{\theta}$ is the parameters of f.

- $\boldsymbol{\theta}$ remains constant as t changes.

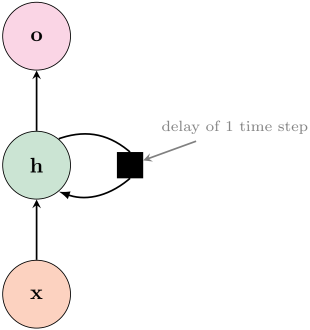
  Apply the **same function** with the **same parameter** values at **each iteration**.

RNN with no outputs



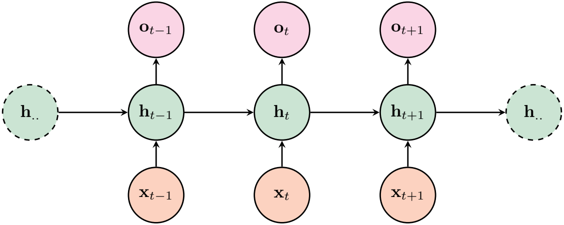
Unrolled visualization of the RNN

RNN with outputs



- Usually also predict an output vector at each time step

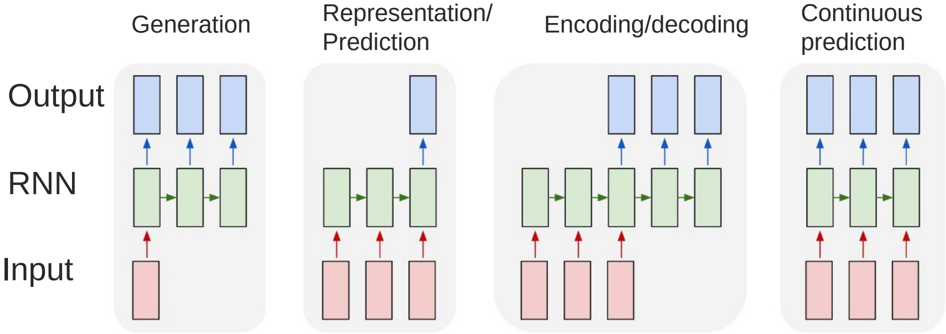
RNN with outputs



Unrolled visualization of the RNN

- Usually also predict an output vector at each time step

Use cases of RNNs



[http://karpathy.github.io/2015/05/21/rnn-effectiveness/]

Back to Vanilla RNN

- The state consists of a single **hidden** vector **h**:
- Initial hidden state $\mathbf{h}_0$ is assumed given.
- For $t = 1, \ldots, T$ the RNN equations are

$$\begin{aligned}
\mathbf{a}_t &= W\mathbf{h}_{t-1} + U\mathbf{x}_t + \mathbf{b} \\
\mathbf{h}_t &= \tanh(\mathbf{a}_t) \\
\mathbf{o}_t &= V\mathbf{h}_t + \mathbf{c} \\
\mathbf{p}_t &= \text{softmax}(\mathbf{o}_t)
\end{aligned}$$

Network's input

- $\mathbf{h}_0$ initial hidden state has size $m \times 1$
- $\mathbf{x}_t$ input vector at time t has size $d \times 1$

Vanilla RNN equations

- The state consists of a single **hidden** vector **h**:
- Initial hidden state $\mathbf{h}_0$ is assumed given.
- For $t = 1, \ldots, T$ the RNN equations are

$$
\begin{aligned}
\mathbf{a}_t &= W\mathbf{h}_{t-1} + U\mathbf{x}_t + \mathbf{b} \\
\mathbf{h}_t &= \tanh(\mathbf{a}_t) \\
\mathbf{o}_t &= V\mathbf{h}_t + \mathbf{c} \\
\mathbf{p}_t &= \text{softmax}(\mathbf{o}_t)
\end{aligned}
$$

Network's output and hidden vectors

- $\mathbf{a}_t$ hidden state at time t of size $m \times 1$ before non-linearity
- $\mathbf{h}_t$ hidden state at time t of size $m \times 1$
- $\mathbf{o}_t$ output vector (of unnormalized log probabilities for each class) at time t of size $C \times 1$
- $\mathbf{p}_t$ output probability vector at time t of size $C \times 1$

Vanilla RNN equations

- The state consists of a single **hidden** vector **h**:
- Initial hidden state $\mathbf{h}_0$ is assumed given.
- For $t = 1, \ldots, T$ the RNN equations are

$$
\begin{aligned}
\mathbf{a}_t &= W\mathbf{h}_{t-1} + U\mathbf{x}_t + \mathbf{b} \\
\mathbf{h}_t &= \tanh(\mathbf{a}_t) \\
\mathbf{o}_t &= V\mathbf{h}_t + \mathbf{c} \\
\mathbf{p}_t &= \text{softmax}(\mathbf{o}_t)
\end{aligned}
$$

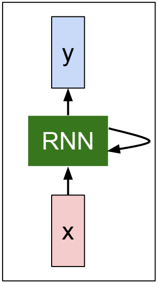
Parameters of the network

- W weight matrix of size $m \times m$ applied to $\mathbf{h}_{t-1}$ (hidden-to-hidden connection)
- U weight matrix of size $m \times d$ applied to $\mathbf{x}_t$ (input-to-hidden connection)
- $\mathbf{b}$ bias vector of size $m \times 1$ in equation for $\mathbf{a}_t$
- V weight matrix of size $C \times m$ applied to $\mathbf{a}_t$ (hidden-to-output connection)
- $\mathbf{c}$ bias vector of size $C \times 1$ in equation for $\mathbf{o}_t$

Character-level language model example

Vocabulary:
[h,e,l,o]

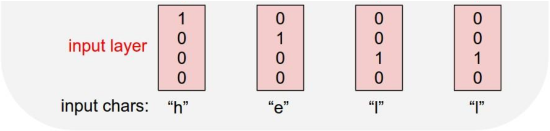
Example training sequence:
“hello”



Character-level language model example

Vocabulary:
[h,e,l,o]

Example training sequence:
“hello”

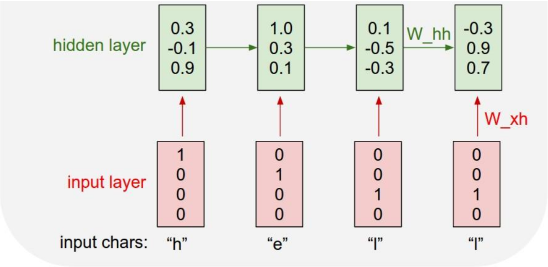


Character-level language model example

Vocabulary:
[h,e,l,o]

Example training sequence:
“hello”

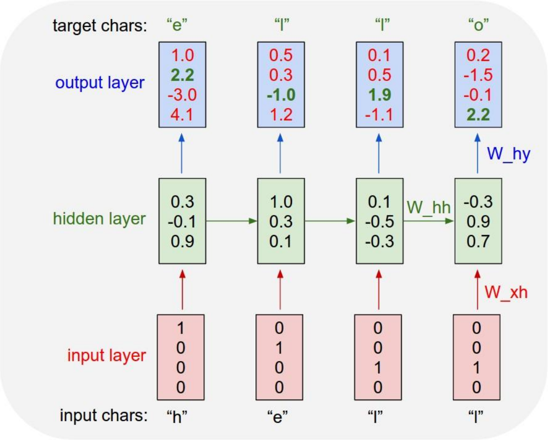
$$\boxed{h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)}$$



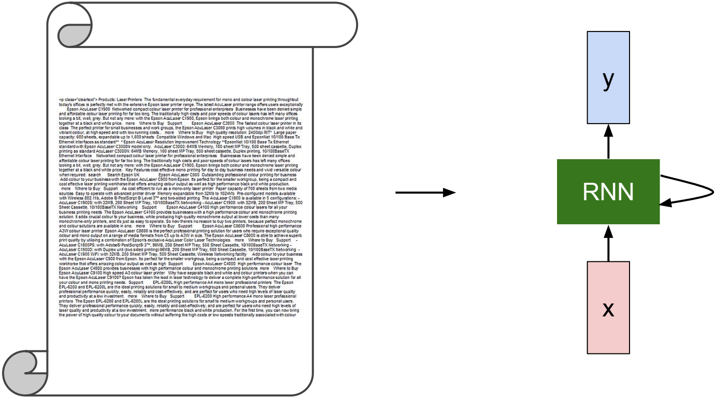
Character-level language model example

Vocabulary:
[h,e,l,o]

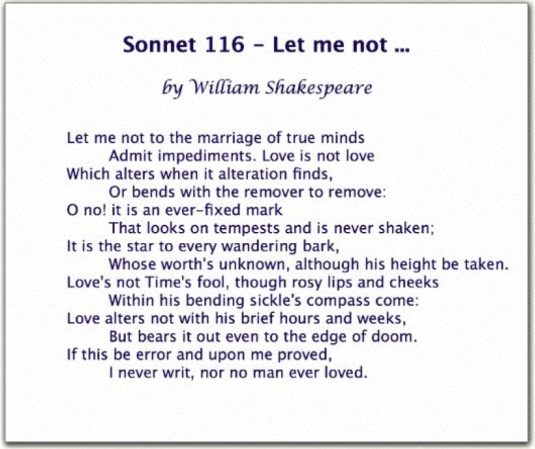
Example training sequence:
“hello”



Extend this simple approach to full alphabet and punctuation characters



<p class="cleartext"> Products: Laser Printers The fundamental everyday requirement for mono and colour laser printing throughout today's offices is perfectly met with the extensive Epson laser printer range. The latest AcuLaser printer range offers users exceptionally Epson AcuLaser C1900 Networked compact colour laser printer for professional enterprises Businesses have been denied simple and affordable colour laser printing for far too long. The traditionally high costs and poor speeds of colour lasers has left many offices looking a bit, well, grey. But not any more: with the Epson AcuLaser C1900, Epson brings both colour and monochrome laser printing together at a black and white price. more Where to Buy Support Epson AcuLaser C3000 The fastest colour laser printer in its class The perfect printer for small businesses and work groups, the Epson AcuLaser C3000 prints high volumes in black and white and vibrant colour, at high speed and with low running costs... more Where to Buy High quality resolution: 2400dpi RIT* Large paper capacity: 600 sheets, expandable up to 1,600 sheets Compatible Windows and Mac High speed USB and EpsonNet 10/100 Base Tx Ethernet interfaces as standard** * Epson AcuLaser Resolution Improvement Technology **EpsonNet 10/100 Base Tx Ethernet standard with Epson AcuLaser C3000N model only. AcuLaser C3000: 64MB Memory, 100 sheet MP Tray, 500 sheet cassette, Duplex printing as standard AcuLaser C3000N: 64MB Memory, 100 sheet MP Tray, 500 sheet cassette, Duplex printing, 10/100BaseTX Ethernet Interface Networked compact colour laser printer for professional enterprises Businesses have been denied simple and affordable colour laser printing for far too long. The traditionally high costs and poor speeds of colour lasers has left many offices looking a bit, well, grey. But not any more: with the Epson AcuLaser C1900, Epson brings both colour and monochrome laser printing together at a black and white price. Key Features cost effective mono printing for day to day business needs and vivid versatile colour when required. search Search Epson UK Epson AcuLaser C900 Outstanding professional colour printing for business Add colour to your business with the Epson AcuLaser C900 from Epson. Its perfect for the smaller workgroup, being a compact and cost effective laser printing workhorse that offers amazing colour output as well as high performance black and white production. more Where to Buy Support As cost efficient to run as a mono-only laser printer Paper capacity of 700 sheets from two media sources Easy to operate with advanced printer driver Memory expandable from 32Mb to 1024Mb Pre-configured models available with Wireless 802.11b, Adobe ® PostScript ® Level 3™ and two-sided printing The AcuLaser C1900 is available in 5 configurations: - AcuLaser C1900S: with 32MB, 200 Sheet MP Tray, 10/100BaseTX Networking - AcuLaser C1900: with 32MB, 200 Sheet MP Tray, 500 Sheet Cassette, 10/100BaseTX Networking Support Epson AcuLaser C4100 High performance colour lasers for all your business printing needs The Epson AcuLaser C4100 provides businesses with a high performance colour and monochrome printing solution. It adds crucial colour to your business, while producing high quality monochrome output at lower costs than many monochrome-only printers, and it's just as easy to operate. So now there's no reason to buy two printers, because perfect monochrome and colour solutions are available in one. more Where to Buy Support Epson AcuLaser C8600 Professional high performance A3W colour laser printer Epson AcuLaser C8600 is the perfect professional printing solution for users who require exceptional quality colour and mono output on a range of media formats from C5 up to A3W in size. The Epson AcuLaser C8600 is able to achieve superb print quality by utilising a combination of Epson's exclusive AcuLaser Color Laser Technologies. more Where to Buy Support - AcuLaser C1900PS: with Adobe® PostScript® 3™, 96MB, 200 Sheet MP Tray, 500 Sheet Cassette, 10/100BaseTX Networking - AcuLaser C1900D: with Duplex unit (two sided printing) 96MB, 200 Sheet MP Tray, 500 Sheet Cassette, 10/100BaseTX Networking - AcuLaser C1900 WiFi: with 32MB, 200 Sheet MP Tray, 500 Sheet Cassette, Wireless Networking facility Add colour to your business with the Epson AcuLaser C900 from Epson. Its perfect for the smaller workgroup, being a compact and cost effective laser printing workhorse that offers amazing colour output as well as high Support Epson AcuLaser C4000 High performance colour laser The Epson AcuLaser C4000 provides businesses with high performance colour and monochrome printing solutions more Where to Buy Epson AcuLaser C9100 High speed A3 colour laser printer Why have separate black and white and colour printers when you can have the Epson AcuLaser C9100? Epson has taken the lead in laser technology to deliver a complete high-performance solution for all your colour and mono printing needs. Support EPL-6200L High performance A4 mono laser professional printers The Epson EPL-6200 and EPL-6200L are the ideal printing solutions for small to medium workgroups and personal users. They deliver professional performance quickly, easily, reliably and cost-effectively, and are perfect for users who need high levels of laser quality and productivity at a low investment. more Where to Buy Support EPL-6200 High performance A4 mono laser professional printers The Epson EPL-6200 and EPL-6200L are the ideal printing solutions for small to medium workgroups and personal users. They deliver professional performance quickly, easily, reliably and cost-effectively, and are perfect for users who need high levels of laser quality and productivity at a low investment. more performance black and white production. For the first time, you can now bring the power of high quality colour to your documents without suffering the high costs or low speeds traditionally associated with colour

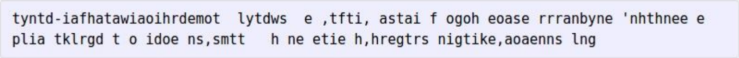


Sonnet 116 – Let me not ...

by William Shakespeare

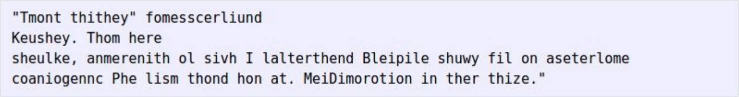
Let me not to the marriage of true minds
Admit impediments. Love is not love
Which alters when it alteration finds,
Or bends with the remover to remove:
O no! it is an ever-fixed mark
That looks on tempests and is never shaken;
It is the star to every wandering bark,
Whose worth's unknown, although his height be taken.
Love's not Time's fool, though rosy lips and cheeks
Within his bending sickle's compass come:
Love alters not with his brief hours and weeks,
But bears it out even to the edge of doom.
If this be error and upon me proved,
I never writ, nor no man ever loved.

at first:



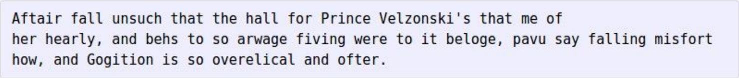
```
tyntd-iafhatawiaoihrdemot  lytdws  e ,tfti, astai f ogoh eoase rrranbyne 'nhthnee e
plia tklrgd t o idoe ns,smtt   h ne etie h,hregtrs nigtike,aoaenns lng
```

train more



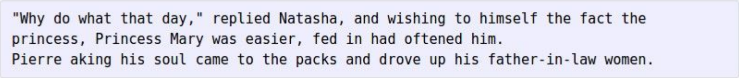
```
"Tmont thithey" fomesscerliund
Keushey. Thom here
sheulke, anmerenith ol sivh I lalterthend Bleipile shuwy fil on aseterlome
coaniogennc Phe lism thond hon at. MeiDimorotion in ther thize."
```

train more

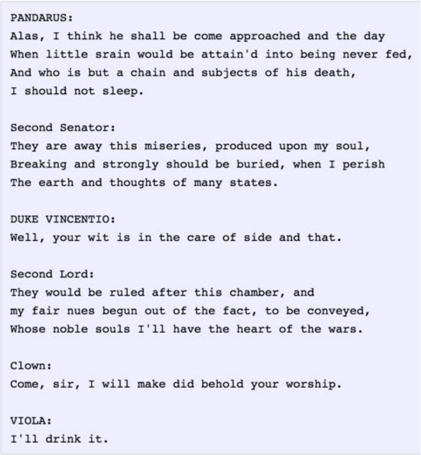


```
Aftair fall unsuch that the hall for Prince Velzonski's that me of
her hearly, and behs to so arwage fiving were to it beloge, pavu say falling misfort
how, and Gogition is so overelical and ofter.
```

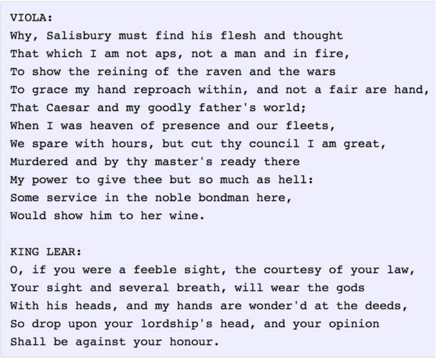
train more



```
"Why do what that day," replied Natasha, and wishing to himself the fact the
princess, Princess Mary was easier, fed in had oftened him.
Pierre aking his soul came to the packs and drove up his father-in-law women.
```



```
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.
```



```
VIOLA:
Why, Salisbury must find his flesh and thought
That which I am not aps, not a man and in fire,
To show the reining of the raven and the wars
To grace my hand reproach within, and not a fair are hand,
That Caesar and my goodly father's world;
When I was heaven of presence and our fleets,
We spare with hours, but cut thy council I am great,
Murdered and by thy master's ready there
My power to give thee but so much as hell:
Some service in the noble bondman here,
Would show him to her wine.

KING LEAR:
O, if you were a feeble sight, the courtesy of your law,
Your sight and several breath, will wear the gods
With his heads, and my hands are wonder'd at the deeds,
So drop upon your lordship's head, and your opinion
Shall be against your honour.
```

How do we train a vanilla RNN?

Supervised learning via a loss function & mini-batch gradient descent.

Loss defined for one training sequence.

- Have a sequence $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_\tau$ of input vectors.
- For each $\mathbf{x}_t$ in sequence have a target output $\mathbf{y}_t$.
- Define loss l_t between the y_t and $\mathbf{p}_t$ for each t.
- Sum the loss over all time-steps

$$L(\mathbf{x}_1, \ldots, \mathbf{x}_\tau, y_1, \ldots, y_\tau, W, U, V, \mathbf{b}, \mathbf{c}) = \sum_{t=1}^{\tau} l_t$$

Loss function for a vanilla RNN

Common to use the cross-entropy loss:

$$l_t = -\log(\mathbf{y}_t^T \mathbf{p}_t)$$

thus

$$L(\mathbf{x}_{1:\tau}, y_{1:\tau}, W, U, V, \mathbf{b}, \mathbf{c}) = -\sum_{t=1}^{\tau} \log(\mathbf{y}_t^T \mathbf{p}_t)$$

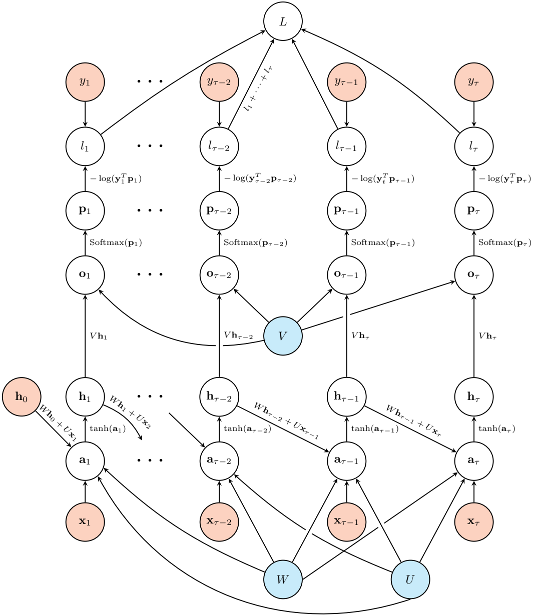
where $\mathbf{x}_{1:\tau} = \{\mathbf{x}_1, \ldots, \mathbf{x}_\tau\}$ and $\mathbf{y}_{1:\tau} = \{y_1, \ldots, y_\tau\}$.

- To implement mini-batch gradient descent need to compute

$$\frac{\partial L(\mathbf{x}_{1:\tau}, y_{1:\tau}, W, U, V, \mathbf{b}, \mathbf{c})}{\partial W}, \frac{\partial L(\mathbf{x}_{1:\tau}, y_{1:\tau}, W, U, V, \mathbf{b}, \mathbf{c})}{\partial U}, \ldots$$

- You've guessed it, use back-prop...

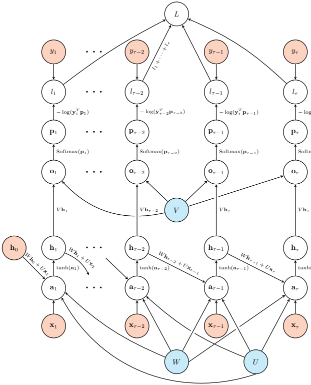
Computational Graph for vanilla RNN loss



- Loss for one labelled training sequence $\mathbf{x}_1, \ldots \mathbf{x}_\tau$
- Bias vectors have been omitted for clarity.

Back-prop for a vanilla RNN

Gradient of loss for the cross-entropy & softmax layers



Know from prior dealings with cross-entropy loss:

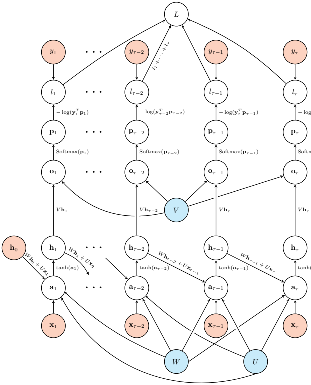
for $t = 1, \ldots \tau$

$$\frac{\partial L}{\partial l_t} = 1$$

$$\frac{\partial L}{\partial \mathbf{p}_t} = \frac{\partial L}{\partial l_t}\frac{\partial l_t}{\partial \mathbf{p}_t} = -\frac{\mathbf{y}_t^T}{\mathbf{y}_t^T\mathbf{p}_t}$$

$$\frac{\partial L}{\partial \mathbf{o}_t} = \frac{\partial L}{\partial \mathbf{p}_t}\frac{\partial \mathbf{p}_t}{\partial \mathbf{o}_t} = -\frac{\mathbf{y}_t^T}{\mathbf{y}_t^T\mathbf{p}_t}\left(\mathrm{diag}(\mathbf{p}_t) - \mathbf{p}_t\mathbf{p}_t^T\right)$$

Gradient of loss w.r.t. V



Children of node V are $\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_\tau$. Thus

$$\frac{\partial L}{\partial \text{vec}(V)} = \sum_{t=1}^{\tau} \frac{\partial L}{\partial \mathbf{o}_t} \frac{\partial \mathbf{o}_t}{\partial \text{vec}(V)}$$

Know

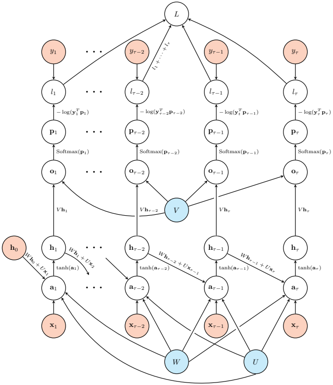
$$\begin{aligned} \mathbf{o}_t = V\mathbf{h}_t \implies \mathbf{o}_t &= (I_C \otimes \mathbf{h}_t)\,\text{vec}(V) \\ \implies \frac{\partial \mathbf{o}_t}{\partial \text{vec}(V)} &= I_C \otimes \mathbf{h}_t \end{aligned}$$

From prior reshapings know:

$$\frac{\partial L}{\partial V} = \sum_{t=1}^{\tau} \mathbf{g}_t^T \mathbf{h}_t^T$$

where $\mathbf{g}_t = \frac{\partial L}{\partial \mathbf{o}_t}$.

Gradient of loss w.r.t. V



Children of node V are $\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_\tau$. Thus

$$\frac{\partial L}{\partial \text{vec}(V)} = \sum_{t=1}^{\tau} \frac{\partial L}{\partial \mathbf{o}_t} \frac{\partial \mathbf{o}_t}{\partial \text{vec}(V)}$$

Know

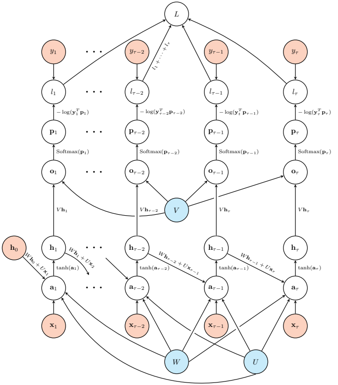
$$\mathbf{o}_t = V\mathbf{h}_t \implies \mathbf{o}_t = (I_C \otimes \mathbf{h}_t)\,\text{vec}(V)$$
$$\implies \frac{\partial \mathbf{o}_t}{\partial \text{vec}(V)} = I_C \otimes \mathbf{h}_t$$

From prior reshapings know:

$$\frac{\partial L}{\partial V} = \sum_{t=1}^{\tau} \mathbf{g}_t^T \mathbf{h}_t^T \leftarrow \text{gradient needed for training network}$$

where $\mathbf{g}_t = \frac{\partial L}{\partial \mathbf{o}_t}$.

Gradient of loss w.r.t. $\mathbf{h}_\tau$



$\mathbf{h}_\tau$ (last hidden state) has one child $\mathbf{o}_\tau$ thus

$$\frac{\partial L}{\partial \mathbf{h}_\tau} = \frac{\partial L}{\partial \mathbf{o}_\tau} \frac{\partial \mathbf{o}_\tau}{\partial \mathbf{h}_\tau}$$

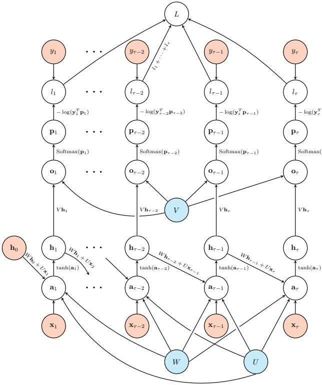
Know

$$\mathbf{o}_\tau = V\mathbf{h}_\tau \implies \frac{\partial \mathbf{o}_\tau}{\partial \mathbf{h}_\tau} = V$$

Thus

$$\frac{\partial L}{\partial \mathbf{h}_\tau} = \frac{\partial L}{\partial \mathbf{o}_\tau} V$$

Gradient of loss w.r.t. $\mathbf{h}_t$



If $1 \le t \le \tau - 1$ then $\mathbf{h}_t$ has children $\mathbf{o}_t$ and $\mathbf{a}_{t+1}$

$$\frac{\partial L}{\partial \mathbf{h}_t} = \frac{\partial L}{\partial \mathbf{o}_t}\frac{\partial \mathbf{o}_t}{\partial \mathbf{h}_t} + \frac{\partial L}{\partial \mathbf{a}_{t+1}}\frac{\partial \mathbf{a}_{t+1}}{\partial \mathbf{h}_t}$$

Know

$$\mathbf{o}_t = V\mathbf{h}_t \implies \frac{\partial \mathbf{o}_t}{\partial \mathbf{h}_t} = V$$

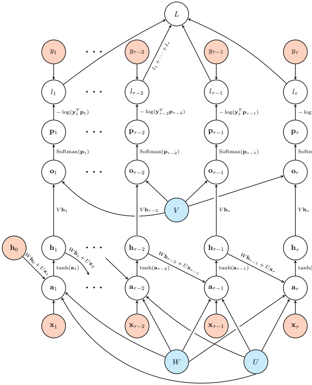
and

$$\mathbf{a}_{t+1} = W\mathbf{h}_t + U\mathbf{x}_{t+1} \implies \frac{\partial \mathbf{a}_{t+1}}{\partial \mathbf{h}_t} = W$$

Thus

$$\frac{\partial L}{\partial \mathbf{h}_t} = \frac{\partial L}{\partial \mathbf{o}_t}V + \frac{\partial L}{\partial \mathbf{a}_{t+1}}W$$

Gradient of loss w.r.t. $\mathbf{h}_t$



If $1 \le t \le \tau - 1$ then $\mathbf{h}_t$ has children $\mathbf{o}_t$ and $\mathbf{a}_{t+1}$

$$\frac{\partial L}{\partial \mathbf{h}_t} = \frac{\partial L}{\partial \mathbf{o}_t}\frac{\partial \mathbf{o}_t}{\partial \mathbf{h}_t} + \frac{\partial L}{\partial \mathbf{a}_{t+1}}\frac{\partial \mathbf{a}_{t+1}}{\partial \mathbf{h}_t}$$

Know

$$\mathbf{o}_t = V\mathbf{h}_t \implies \frac{\partial \mathbf{o}_t}{\partial \mathbf{h}_t} = V$$

and

$$\mathbf{a}_{t+1} = W\mathbf{h}_t + U\mathbf{x}_{t+1} \implies \frac{\partial \mathbf{a}_{t+1}}{\partial \mathbf{h}_t} = W$$

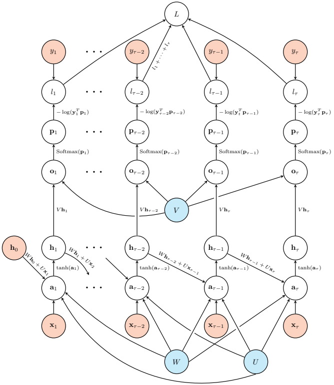
Thus

$$\frac{\partial L}{\partial \mathbf{h}_t} = \frac{\partial L}{\partial \mathbf{o}_t}V + \frac{\partial L}{\partial \mathbf{a}_{t+1}}W$$

Have two different time steps in expression $\implies$

must iterate backwards in time to compute all $\frac{\partial L}{\partial \mathbf{h}_t}$

Gradient of loss w.r.t. $\mathbf{a}_t$



The gradient w.r.t. $\mathbf{a}_t$

$$\frac{\partial L}{\partial \mathbf{a}_t} = \frac{\partial L}{\partial \mathbf{h}_t} \frac{\partial \mathbf{h}_t}{\partial \mathbf{a}_t}$$

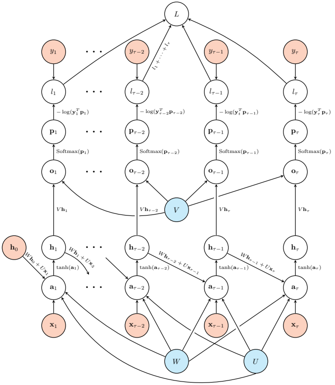
Know

$$\mathbf{h}_t = \tanh(\mathbf{a}_t) \implies \frac{\partial \mathbf{h}_t}{\partial \mathbf{a}_t} = \text{diag}\left(\tanh'(\mathbf{a}_t)\right)$$
$$= \text{diag}\left(1 - \tanh^2(\mathbf{a}_t)\right)$$

Thus

$$\frac{\partial L}{\partial \mathbf{a}_t} = \frac{\partial L}{\partial \mathbf{h}_t} \text{diag}\left(1 - \tanh^2(\mathbf{a}_t)\right)$$

Recursively compute gradients for all $\mathbf{a}_t$ and $\mathbf{h}_t$



- Assume $\frac{\partial L}{\partial \mathbf{o}_t}$ calculated for $1 \leq t \leq \tau$.
- Calculate

$$\frac{\partial L}{\partial \mathbf{h}_\tau} = \frac{\partial L}{\partial \mathbf{o}_\tau} V \quad \& \quad \frac{\partial L}{\partial \mathbf{a}_\tau} = \frac{\partial L}{\partial \mathbf{h}_\tau} \text{diag}\left(1 - \tanh^2(\mathbf{a}_\tau)\right)$$

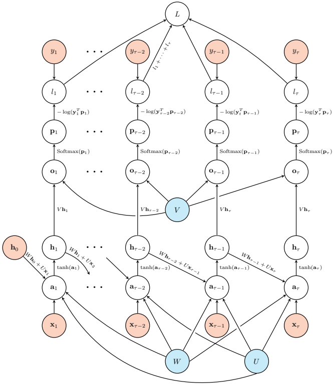
- for $t = \tau - 1, \tau - 2, \ldots, 1$
  - Compute

$$\frac{\partial L}{\partial \mathbf{h}_t} = \frac{\partial L}{\partial \mathbf{o}_t} V + \frac{\partial L}{\partial \mathbf{a}_{t+1}} W$$

  - Compute

$$\frac{\partial L}{\partial \mathbf{a}_t} = \frac{\partial L}{\partial \mathbf{h}_t} \text{diag}\left(1 - \tanh^2(\mathbf{a}_t)\right)$$

Gradient of loss w.r.t. W



The gradient of the loss w.r.t. node W.

Children of W are $\mathbf{a}_1, \ldots \mathbf{a}_\tau$ thus

$$\frac{\partial L}{\partial \text{vec}(W)} = \sum_{t=1}^{\tau} \frac{\partial L}{\partial \mathbf{a}_t} \frac{\partial \mathbf{a}_t}{\partial \text{vec}(W)}$$

Know

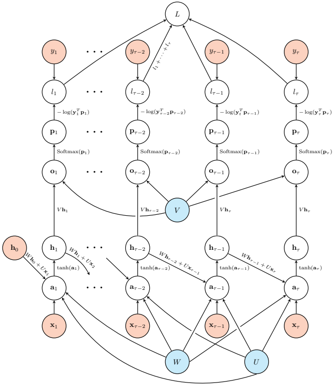
$$\mathbf{a}_t = W\mathbf{h}_{t-1} + U\mathbf{x}_t \implies \mathbf{a}_t = (I_m \otimes \mathbf{h}_{t-1})\text{vec}(W) + U\mathbf{x}_t$$
$$\implies \frac{\partial \mathbf{a}_t}{\partial \text{vec}(W)} = I_m \otimes \mathbf{h}_{t-1}$$

From prior reshapings know:

$$\frac{\partial L}{\partial W} = \sum_{t=1}^{\tau} \mathbf{g}_t^T \mathbf{h}_{t-1}^T$$

where $\mathbf{g}_t = \frac{\partial L}{\partial \mathbf{a}_t}$.

Gradient of loss w.r.t. W



The gradient of the loss w.r.t. node W.

Children of W are $\mathbf{a}_1, \ldots \mathbf{a}_\tau$ thus

$$\frac{\partial L}{\partial \text{vec}(W)} = \sum_{t=1}^{\tau} \frac{\partial L}{\partial \mathbf{a}_t} \frac{\partial \mathbf{a}_t}{\partial \text{vec}(W)}$$

Know

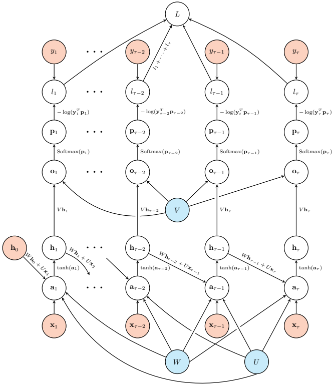
$$\mathbf{a}_t = W\mathbf{h}_{t-1} + U\mathbf{x}_t \implies \mathbf{a}_t = (I_m \otimes \mathbf{h}_{t-1})\text{vec}(W) + U\mathbf{x}_t$$
$$\implies \frac{\partial \mathbf{a}_t}{\partial \text{vec}(W)} = I_m \otimes \mathbf{h}_{t-1}$$

From prior reshapings know:

$$\frac{\partial L}{\partial W} = \sum_{t=1}^{\tau} \mathbf{g}_t^T \mathbf{h}_{t-1}^T \leftarrow \text{gradient needed for training network}$$

where $\mathbf{g}_t = \frac{\partial L}{\partial \mathbf{a}_t}$.

Gradient of loss w.r.t. U



The gradient of the loss w.r.t. node V.

Children of V are $\mathbf{a}_1, \ldots \mathbf{a}_\tau$ thus

$$\frac{\partial L}{\partial \text{vec}(U)} = \sum_{t=1}^{\tau} \frac{\partial L}{\partial \mathbf{a}_t} \frac{\partial \mathbf{a}_t}{\partial \text{vec}(U)}$$

Know

$$\mathbf{a}_t = W\mathbf{h}_{t-1} + U\mathbf{x}_t \implies \mathbf{a}_t = W\mathbf{h}_{t-1} + (I_m \otimes \mathbf{x}_t)\text{vec}(U)$$

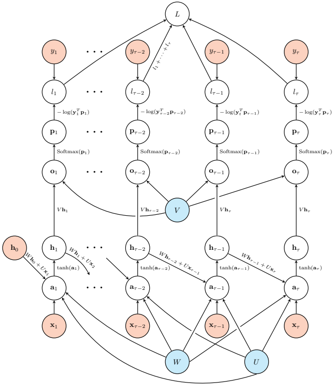
$$\implies \frac{\partial \mathbf{a}_t}{\partial \text{vec}(U)} = I_m \otimes \mathbf{x}_t$$

From prior reshapings know:

$$\frac{\partial L}{\partial U} = \sum_{t=1}^{\tau} \mathbf{g}_t^T \mathbf{x}_t^T$$

where $\mathbf{g}_t = \frac{\partial L}{\partial \mathbf{a}_t}$.

Gradient of loss w.r.t. U



The gradient of the loss w.r.t. node V.

Children of V are $\mathbf{a}_1, \ldots \mathbf{a}_\tau$ thus

$$\frac{\partial L}{\partial \text{vec}(U)} = \sum_{t=1}^{\tau} \frac{\partial L}{\partial \mathbf{a}_t} \frac{\partial \mathbf{a}_t}{\partial \text{vec}(U)}$$

Know

$$\mathbf{a}_t = W\mathbf{h}_{t-1} + U\mathbf{x}_t \implies \mathbf{a}_t = W\mathbf{h}_{t-1} + (I_m \otimes \mathbf{x}_t)\text{vec}(U)$$
$$\implies \frac{\partial \mathbf{a}_t}{\partial \text{vec}(U)} = I_m \otimes \mathbf{x}_t$$

From prior reshapings know:

$$\frac{\partial L}{\partial U} = \sum_{t=1}^{\tau} \mathbf{g}_t^T \mathbf{x}_t^T \leftarrow \text{gradient needed for training network}$$

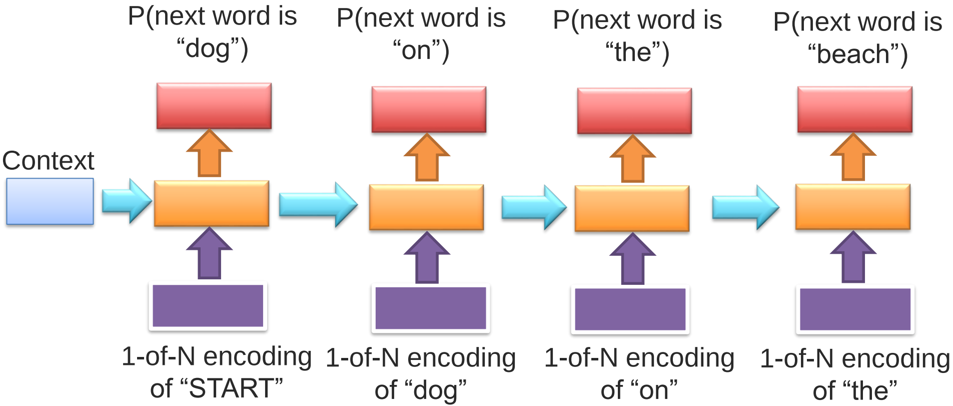
where $\mathbf{g}_t = \frac{\partial L}{\partial \mathbf{a}_t}$.

RNNs in Translation Applications

Language translation

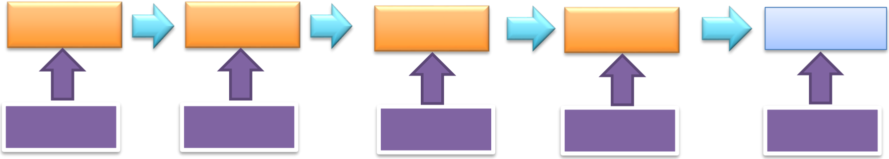
- Given a sentence in on language translate it to another language
- le chien sur la plage → Dog on the beach

RNN-based Sentence Generation (Decoder)



- Model long-term information

RNN-based Sentence Representation (Encoder)



1-of-N encoding of “le”

1-of-N encoding of “chien”

1-of-N encoding of “sur”

1-of-N encoding of “la”

1-of-N encoding of “plage”

Encoder-Decoder Architecture

[Cho et al., “Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation”, EMNLP 2014]

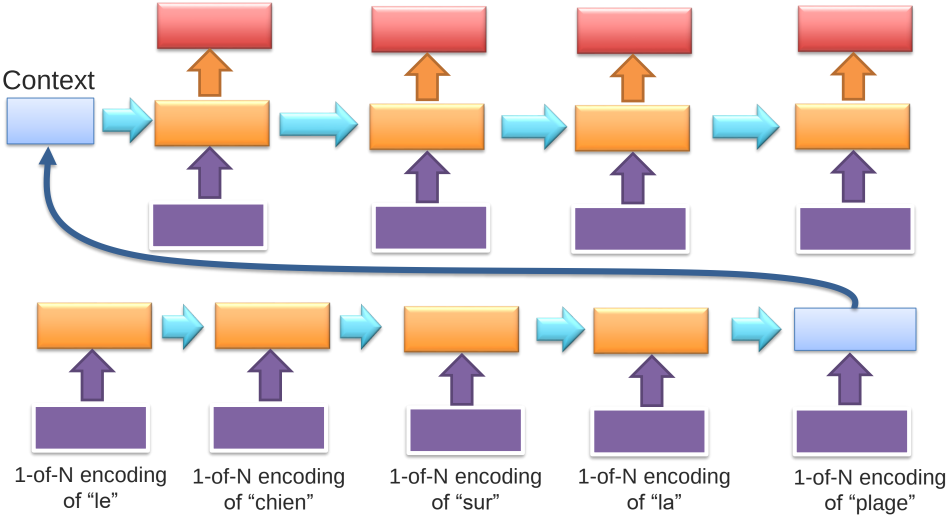
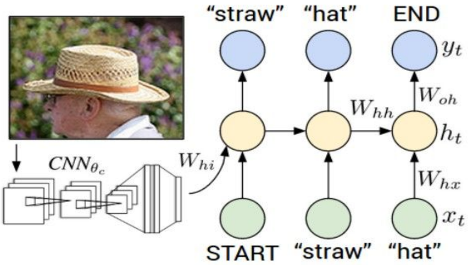
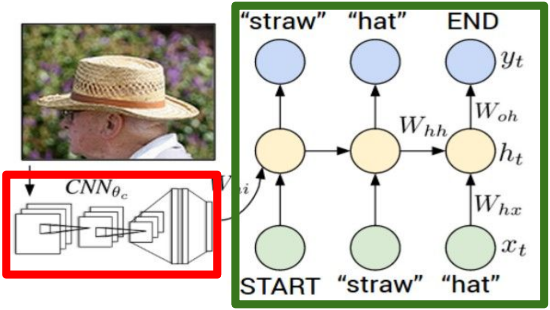


Image Captioning



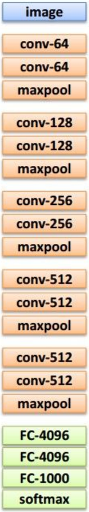
Explain Images with Multimodal Recurrent Neural Networks, Mao et al.
Deep Visual-Semantic Alignments for Generating Image Descriptions, Karpathy and Fei-Fei
Show and Tell: A Neural Image Caption Generator, Vinyals et al.
Long-term Recurrent Convolutional Networks for Visual Recognition and Description, Donahue et al.
Learning a Recurrent Visual Representation for Image Caption Generation, Chen and Zitnick

Recurrent Neural Network



Convolutional Neural Network

test image



image

conv-64

conv-64

maxpool

conv-128

conv-128

maxpool

conv-256

conv-256

maxpool

conv-512

conv-512

maxpool

conv-512

conv-512

maxpool

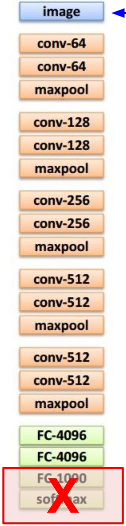
FC-4096

FC-4096

FC-1000

softmax

test image



image

conv-64

conv-64

maxpool

conv-128

conv-128

maxpool

conv-256

conv-256

maxpool

conv-512

conv-512

maxpool

conv-512

conv-512

maxpool

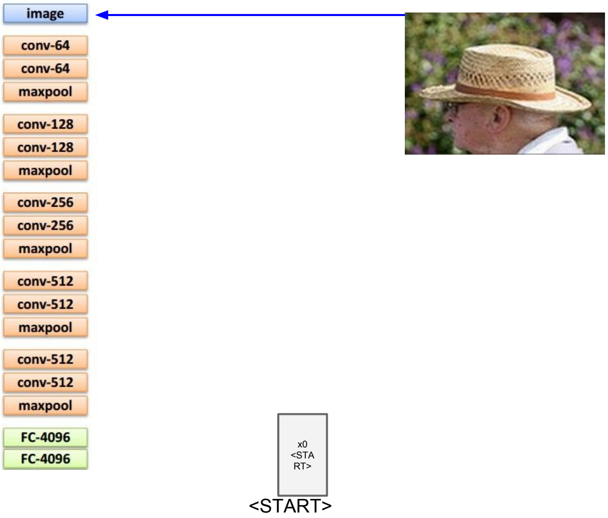
FC-4096

FC-4096

~~FC-1000~~

~~softmax~~

test image



image

conv-64
conv-64
maxpool

conv-128
conv-128
maxpool

conv-256
conv-256
maxpool

conv-512
conv-512
maxpool

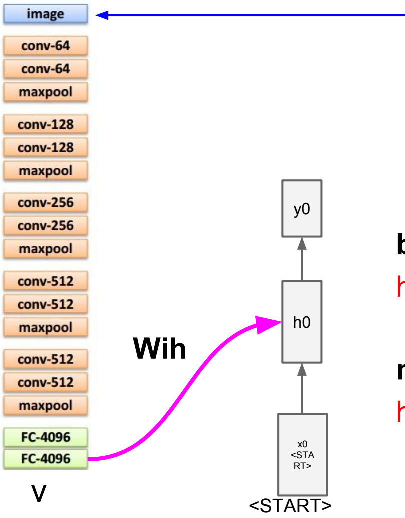
conv-512
conv-512
maxpool

FC-4096
FC-4096

test image

x0
<START>

<START>



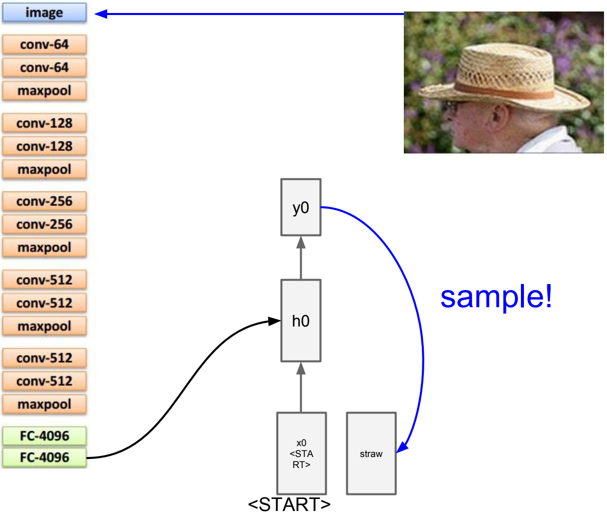
test image

before:

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

now:

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



image

conv-64

conv-64

maxpool

conv-128

conv-128

maxpool

conv-256

conv-256

maxpool

conv-512

conv-512

maxpool

conv-512

conv-512

maxpool

FC-4096

FC-4096

y0

h0

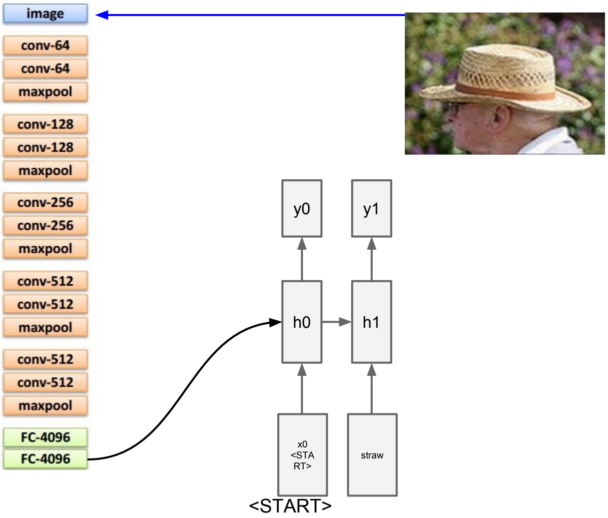
x0 <STA RT>

straw

<START>

test image

sample!



test image

image

conv-64

conv-64

maxpool

conv-128

conv-128

maxpool

conv-256

conv-256

maxpool

conv-512

conv-512

maxpool

conv-512

conv-512

maxpool

FC-4096

FC-4096

y0

y1

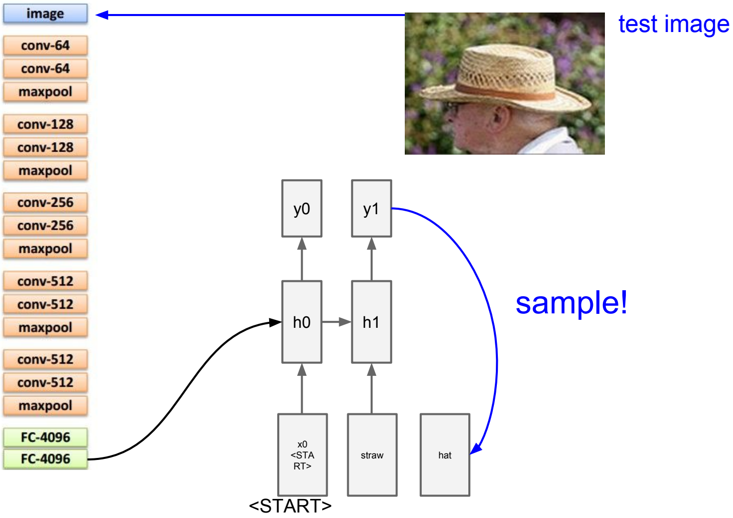
h0

h1

x0 <STA RT>

straw

<START>



test image

image

conv-64

conv-64

maxpool

conv-128

conv-128

maxpool

conv-256

conv-256

maxpool

conv-512

conv-512

maxpool

conv-512

conv-512

maxpool

FC-4096

FC-4096

y0

y1

h0

h1

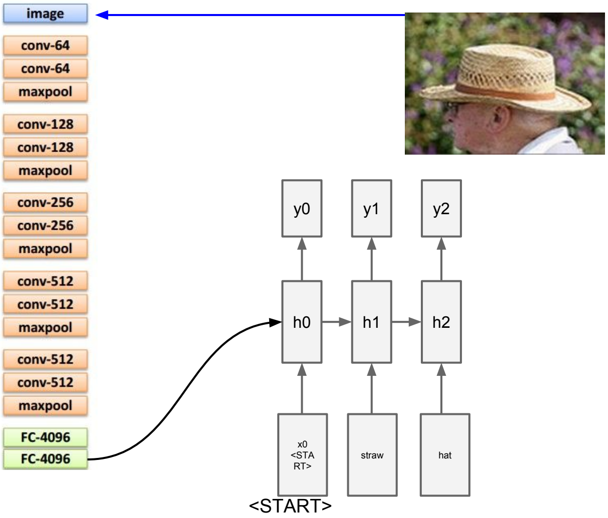
x0
<STA
RT>

straw

hat

<START>

sample!



test image

image

conv-64

conv-64

maxpool

conv-128

conv-128

maxpool

conv-256

conv-256

maxpool

conv-512

conv-512

maxpool

conv-512

conv-512

maxpool

FC-4096

FC-4096

y0

y1

y2

h0

h1

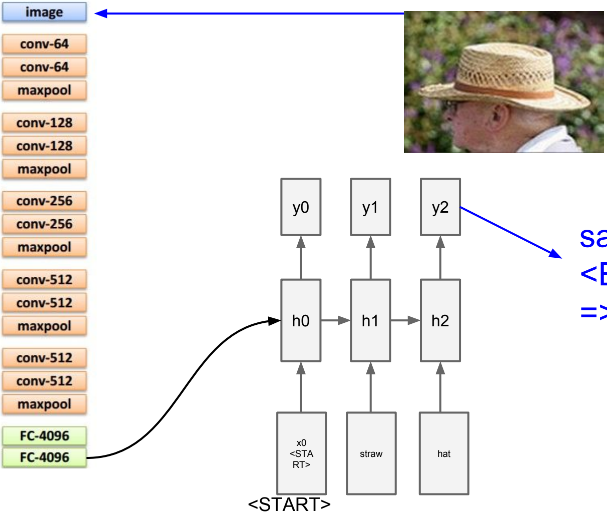
h2

x0 <START>

straw

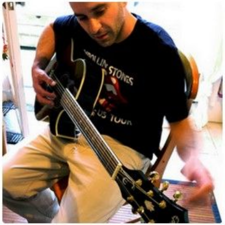
hat

<START>

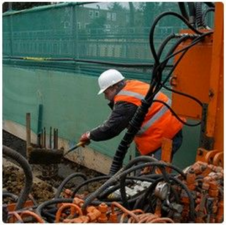


test image

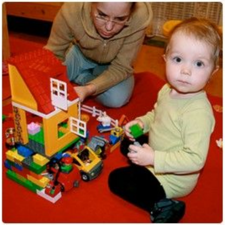
sample <END> token => finish.



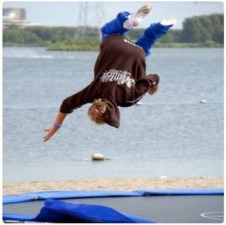
"man in black shirt is playing guitar."



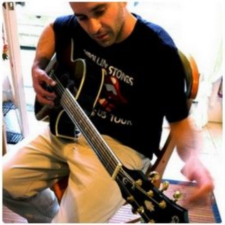
"construction worker in orange safety vest is working on road."



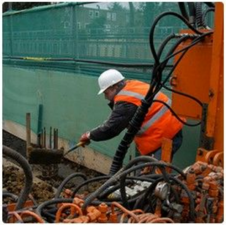
"two young girls are playing with lego toy."



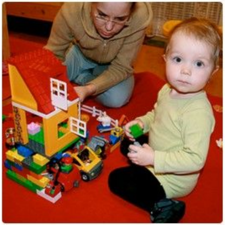
"boy is doing backflip on wakeboard."



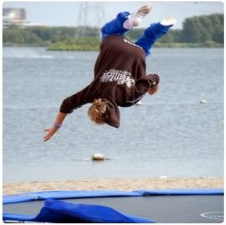
"man in black shirt is playing guitar."



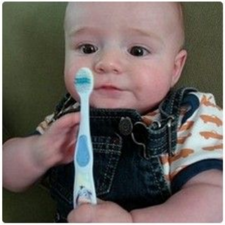
"construction worker in orange safety vest is working on road."



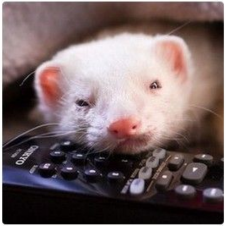
"two young girls are playing with lego toy."



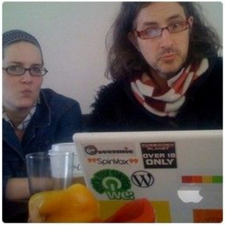
"boy is doing backflip on wakeboard."



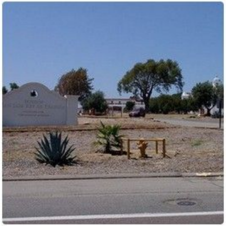
"a young boy is holding a baseball bat."



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



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



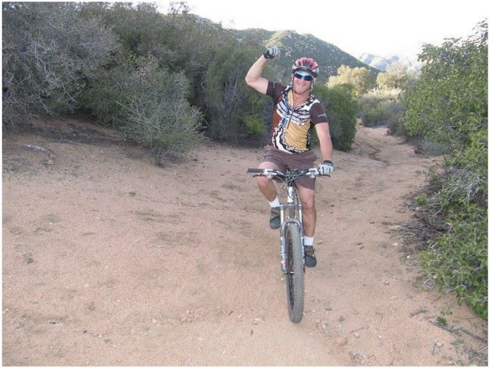
"a horse is standing in the middle of a road."

Evaluation of text translation results

- Tricky to do automatically!
- Ideally want humans to evaluate
    - What do you ask?
    - Can't use human evaluation for validating models - too slow and expensive.
- Use standard machine translation metrics instead
    - BLEU
    - ROUGE CIDER
    - Meteor

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

Problem of exploding and vanishing gradients in an RNN

Focus on gradient of loss w.r.t. W

- Take a closer look at

$$\frac{\partial L}{\partial W} = \sum_{t=1}^{\tau} \mathbf{g}_t^T \mathbf{h}_{t-1}^T \quad \text{where } \mathbf{g}_t = \frac{\partial L}{\partial \mathbf{a}_t}$$

- $\implies$ $\frac{\partial L}{\partial W}$ depends on $\mathbf{h}_{t-1}$ and $\frac{\partial L}{\partial \mathbf{a}_t}$ for $t = 1, \ldots, \tau$.

- Let's take a closer look at $\frac{\partial L}{\partial \mathbf{a}_t}$....

Focus on $\frac{\partial L}{\partial \mathbf{a}_t}$

- Denote

$$\mathbf{g}_{\mathbf{o}_t} = \frac{\partial L}{\partial \mathbf{o}_t} \quad \text{and} \quad D(\mathbf{a}_t) = \text{diag}(1 - \tanh^2(\mathbf{a}_t))$$

- Remember

$$\frac{\partial L}{\partial \mathbf{h}_\tau} = \mathbf{g}_{\mathbf{o}_\tau} V \implies \frac{\partial L}{\partial \mathbf{a}_\tau} = \mathbf{g}_{\mathbf{o}_\tau} V D(\mathbf{a}_\tau)$$

and for $t = \tau - 1, \tau - 2, \ldots, 1$:

$$\frac{\partial L}{\partial \mathbf{h}_t} = \mathbf{g}_{\mathbf{o}_t} V + \frac{\partial L}{\partial \mathbf{a}_{t+1}} W \quad \text{and} \quad \frac{\partial L}{\partial \mathbf{a}_t} = \frac{\partial L}{\partial \mathbf{h}_t} D(\mathbf{a}_t)$$

- Then you can show by recursive substitution that

$$\frac{\partial L}{\partial \mathbf{h}_t} = \sum_{j=t}^{\tau} \mathbf{g}_{\mathbf{o}_j} V \left(\prod_{k=1}^{j-t} D(\mathbf{a}_{t+k}) \right) W^{j-t}$$

and

$$\frac{\partial L}{\partial \mathbf{a}_t} = \sum_{j=t}^{\tau} \mathbf{g}_{\mathbf{o}_j} V \left(\prod_{k=0}^{j-t} D(\mathbf{a}_{t+k}) \right) W^{j-t}$$

Focus on $\frac{\partial L}{\partial \mathbf{a}_t}$

- Denote

$$\mathbf{g}_{\mathbf{o}_t} = \frac{\partial L}{\partial \mathbf{o}_t} \quad \text{and} \quad D(\mathbf{a}_t) = \text{diag}(1 - \tanh^2(\mathbf{a}_t))$$

- Then you can show by recursive substitution that

$$\frac{\partial L}{\partial \mathbf{h}_t} = \sum_{j=t}^{\tau} \mathbf{g}_{\mathbf{o}_j} V \left(\prod_{k=1}^{j-t} D(\mathbf{a}_{t+k}) \right) W^{j-t}$$

and

$$\frac{\partial L}{\partial \mathbf{a}_t} = \sum_{j=t}^{\tau} \mathbf{g}_{\mathbf{o}_j} V \quad \underbrace{\left(\prod_{k=0}^{j-t} D(\mathbf{a}_{t+k}) \right)}_{\text{likely has small values on diagonal}} \quad W^{j-t}$$

Why? Each matrix $D(\mathbf{a}_{t+k})$ has $\tanh'(\mathbf{a}_{t+k})$ on its diagonal and $0 \le \tanh'(a) \le 1$. Thus $(\tanh'(a))^{j-t+1}$ is highly likely to have a small value even for not too large $j - t + 1$.

$\implies$ $\frac{\partial L}{\partial \mathbf{a}_t}$ only depends on first few entries in the sum.

Focus on $\frac{\partial L}{\partial \mathbf{a}_t}$

- Denote

$$\mathbf{g}_{\mathbf{o}_t} = \frac{\partial L}{\partial \mathbf{o}_t} \quad \text{and} \quad D(\mathbf{a}_t) = \text{diag}(1 - \tanh^2(\mathbf{a}_t))$$

- Then you can show by recursive substitution that

$$\frac{\partial L}{\partial \mathbf{h}_t} = \sum_{j=t}^{\tau} \mathbf{g}_{\mathbf{o}_j} V \left(\prod_{k=1}^{j-t} D(\mathbf{a}_{t+k}) \right) W^{j-t}$$

and

$$\frac{\partial L}{\partial \mathbf{a}_t} = \sum_{j=t}^{\tau} \mathbf{g}_{\mathbf{o}_j} V \left(\prod_{k=0}^{j-t} D(\mathbf{a}_{t+k}) \right) \underbrace{W^{j-t}}_{\text{potentially has very large or small values}}$$

Why? ......

Focus on W^n

- Remember W has size $m \times m$.
- Assume W is diagonalizable.
- Let its eigen-decomposition be

  $$W = Q\Lambda Q^T$$

  where Q is orthogonal and Λ is a diagonal matrix containing the eigenvalues of W.
- Then

  $$W^n = Q\Lambda^n Q^T$$

- Let $\lambda_1, \ldots, \lambda_m$ be the e-values of W. Thus
  - If $\lambda_i > 1 \implies \lambda_i^n$ will explode as n increases.
  - If $\lambda_i < 1 \implies \lambda_i^n \to 0$ as n increases.

Focus on $\frac{\partial L}{\partial \mathbf{a}_t}$

- Denote

$$\mathbf{g}_{\mathbf{o}_t} = \frac{\partial L}{\partial \mathbf{o}_t} \quad \text{and} \quad D(\mathbf{a}_t) = \text{diag}(1 - \tanh^2(\mathbf{a}_t))$$

- Then you can show by recursive substitution that

$$\frac{\partial L}{\partial \mathbf{h}_t} = \sum_{j=t}^{\tau} \mathbf{g}_{\mathbf{o}_j} V \left(\prod_{k=1}^{j-t} D(\mathbf{a}_{t+k}) \right) W^{j-t}$$

and

$$\frac{\partial L}{\partial \mathbf{a}_t} = \sum_{j=t}^{\tau} \mathbf{g}_{\mathbf{o}_j} V \left(\prod_{k=0}^{j-t} D(\mathbf{a}_{t+k}) \right) \underbrace{W^{j-t}}_{\text{potentially has very large or small values}}$$

Thus for sufficiently large $j - t$ either entries in W^{j-t} can explode or vanish.

Summary on $\frac{\partial L}{\partial \mathbf{a}_t}$

- Denote

$$\mathbf{g}_{\mathbf{o}_t} = \frac{\partial L}{\partial \mathbf{o}_t} \quad \text{and} \quad D(\mathbf{a}_t) = \text{diag}(1 - \tanh^2(\mathbf{a}_t))$$

- Then you can show by recursive substitution that

$$\frac{\partial L}{\partial \mathbf{h}_t} = \sum_{j=t}^{\tau} \mathbf{g}_{\mathbf{o}_j} V \left(\prod_{k=1}^{j-t} D(\mathbf{a}_{t+k}) \right) W^{j-t}$$

and

$$\frac{\partial L}{\partial \mathbf{a}_t} = \sum_{j=t}^{\tau} \mathbf{g}_{\mathbf{o}_j} V \left(\prod_{k=0}^{j-t} D(\mathbf{a}_{t+k}) \right) W^{j-t}$$

- If W^{j-t} explodes for $j - t > N \implies \frac{\partial L}{\partial \mathbf{a}_t}$ explodes $\implies \frac{\partial L}{\partial W}$ explodes.

- If W^{j-t} vanishes for $j - t > N$
  $\implies \frac{\partial L}{\partial \mathbf{a}_t}$ only has contributions from nearby $\mathbf{g}_{\mathbf{o}_{t'}}$ where $t \leq t' \leq t + N$
  $\implies \frac{\partial L}{\partial W}$ is based on aggregation of gradients from subsets of temporally nearby states.

Summary on $\frac{\partial L}{\partial \mathbf{a}_t}$

- Denote

$$\mathbf{g}_{\mathbf{o}_t} = \frac{\partial L}{\partial \mathbf{o}_t} \quad \text{and} \quad D(\mathbf{a}_t) = \text{diag}(1 - \tanh^2(\mathbf{a}_t))$$

- Then you can show by recursive substitution that

$$\frac{\partial L}{\partial \mathbf{h}_t} = \sum_{j=t}^{\tau} \mathbf{g}_{\mathbf{o}_j} V \left(\prod_{k=1}^{j-t} D(\mathbf{a}_{t+k}) \right) W^{j-t}$$

and

$$\frac{\partial L}{\partial \mathbf{a}_t} = \sum_{j=t}^{\tau} \mathbf{g}_{\mathbf{o}_j} V \left(\prod_{k=0}^{j-t} D(\mathbf{a}_{t+k}) \right) W^{j-t}$$

- If W^{j-t} explodes for $j - t > N \implies \frac{\partial L}{\partial \mathbf{a}_t}$ explodes $\implies \frac{\partial L}{\partial W}$ explodes.

- If W^{j-t} vanishes for $j - t > N$
  $\implies \frac{\partial L}{\partial \mathbf{a}_t}$ only has contributions from nearby $\mathbf{g}_{\mathbf{o}_{t'}}$ where $t \leq t' \leq t + N$
  $\implies \frac{\partial L}{\partial W}$ is based on aggregation of gradients from subsets of temporally nearby states.
  $\implies$ Cannot learn long-range dependencies between states.

Solution to Exploding & Vanishing Gradients

Easy solution to exploding gradients

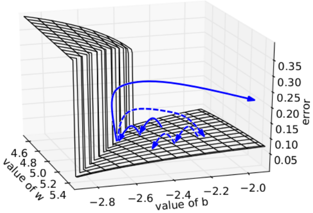
- **Gradient clipping**

  Let $G = \frac{\partial L}{\partial W}$ then

$$G = \begin{cases} \frac{\theta}{\|G\|} G & \text{if } \|G\| \geq \theta \\ G & \text{otherwise} \end{cases}$$

  where θ is some sensible threshold.

- A simple heuristic first introduced by Thomas Mikolov.



Dashed arrow shows what happens when the gradient is rescaled to a fixed size when its norm is above a threshold.

Easy partial solutions to vanishing gradients

- **Solution 1**: Initialize W as the identity matrix as opposed a random initialization.
- **Solution 2**: Use ReLU instead of $\tanh$ as the non-linear activation function.

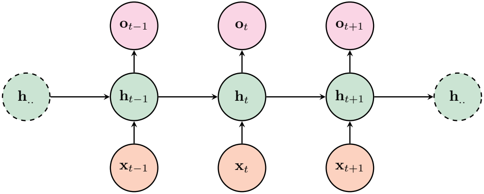
Easy partial solutions to vanishing gradients

- **Solution 1**: Initialize W as the identity matrix as opposed a random initialization.
- **Solution 2**: Use ReLU instead of $\tanh$ as the non-linear activation function.

Still hard for an RNN to capture long-term dependencies.

Long-Short-Term-Memories (LSTMs) - capturing long-range dependencies

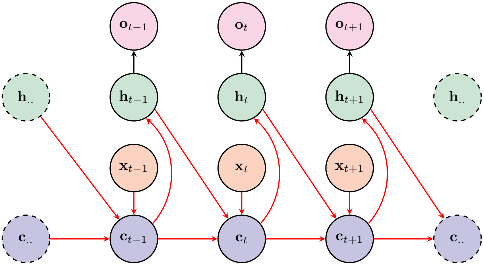
LSTMs Core Idea: Introduce a memory cell



High-level graphic of an RNN

- LSTMs similar to RNN but they introduce a memory cell state $\mathbf{c}_t$.

LSTMs Core Idea: Introduce a memory cell



High-level graphic of a LSTM

- LSTMs similar to RNN but they introduce a memory cell state $\mathbf{c}_t$.
- LSTMs have the ability to remove or add information to $\mathbf{c}_t$ regulated by structures called gates based on context.
- Update of $\mathbf{c}_t$ designed so gradients flows these nodes backward in time easily.
- $\mathbf{c}_t$ then controls what information from $\mathbf{h}_{t-1}$ and $\mathbf{x}_t$ and $\mathbf{c}_{t-1}$ should be used to generate $\mathbf{h}_t$.

LSTMs formal details

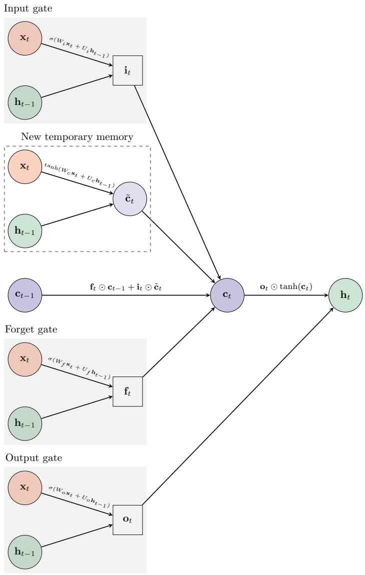
- LSTMs (Hochreiter & Schmidhuer, 1997) better at capturing long term dependencies.
- Introduces gates to calculate $\mathbf{h}_t, \mathbf{c}_t$ from $\mathbf{c}_{t-1}, \mathbf{h}_{t-1}$ and $\mathbf{x}_t$.
- Formal description of a LSTM unit:

$$\begin{aligned}
\mathbf{i}_t &= \sigma(W_i \mathbf{x}_t + U_i \mathbf{h}_{t-1}) && \text{Input gate} \\
\mathbf{f}_t &= \sigma(W_f \mathbf{x}_t + U_f \mathbf{h}_{t-1}) && \text{Forget gate} \\
\mathbf{o}_t &= \sigma(W_o \mathbf{x}_t + U_o \mathbf{h}_{t-1}) && \text{Output/Exposure gate} \\
\tilde{\mathbf{c}}_t &= \tanh(W_c \mathbf{x}_t + U_c \mathbf{h}_{t-1}) && \text{New memory cell} \\
\mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t && \text{Final memory cell} \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t)
\end{aligned}$$

where

- $\sigma(\cdot)$ is the sigmoid function and
- $\odot$ denotes element by element multiplication.

LSTMs basic unit



Input gate

$\mathbf{x}_t$

$\sigma(W_i\mathbf{x}_t + U_i\mathbf{h}_{t-1})$

$\mathbf{i}_t$

$\mathbf{h}_{t-1}$

New temporary memory

$\mathbf{x}_t$

$\tanh(W_c\mathbf{x}_t + U_c\mathbf{h}_{t-1})$

$\tilde{\mathbf{c}}_t$

$\mathbf{h}_{t-1}$

$\mathbf{c}_{t-1}$

$\mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t$

$\mathbf{c}_t$

$\mathbf{o}_t \odot \tanh(\mathbf{c}_t)$

$\mathbf{h}_t$

Forget gate

$\mathbf{x}_t$

$\sigma(W_f\mathbf{x}_t + U_f\mathbf{h}_{t-1})$

$\mathbf{f}_t$

$\mathbf{h}_{t-1}$

Output gate

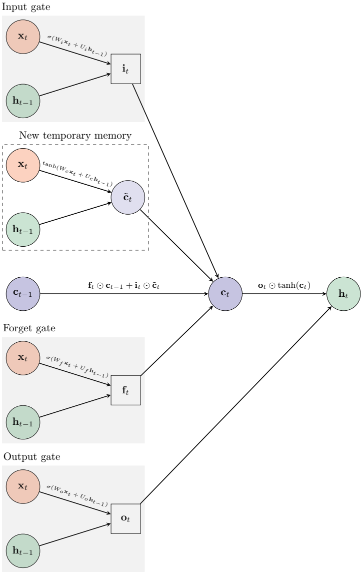
$\mathbf{x}_t$

$\sigma(W_o\mathbf{x}_t + U_o\mathbf{h}_{t-1})$

$\mathbf{o}_t$

$\mathbf{h}_{t-1}$

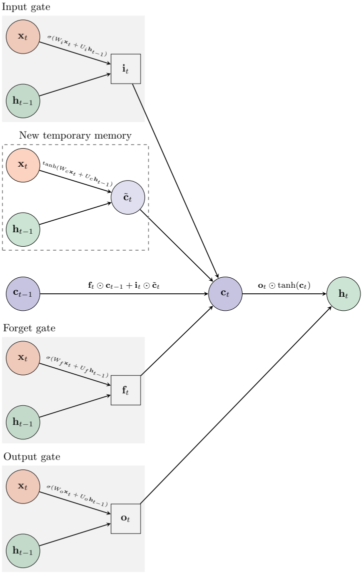
LSTMs basic unit



Input gate
$\mathbf{x}_t$
$\sigma(W_i\mathbf{x}_t + U_i\mathbf{h}_{t-1})$
$\mathbf{i}_t$
$\mathbf{h}_{t-1}$
New temporary memory
$\mathbf{x}_t$
$\tanh(W_c\mathbf{x}_t + U_c\mathbf{h}_{t-1})$
$\tilde{\mathbf{c}}_t$
$\mathbf{h}_{t-1}$
$\mathbf{c}_{t-1}$
$\mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t$
$\mathbf{c}_t$
$\mathbf{o}_t \odot \tanh(\mathbf{c}_t)$
$\mathbf{h}_t$
Forget gate
$\mathbf{x}_t$
$\sigma(W_f\mathbf{x}_t + U_f\mathbf{h}_{t-1})$
$\mathbf{f}_t$
$\mathbf{h}_{t-1}$
Output gate
$\mathbf{x}_t$
$\sigma(W_o\mathbf{x}_t + U_o\mathbf{h}_{t-1})$
$\mathbf{o}_t$
$\mathbf{h}_{t-1}$

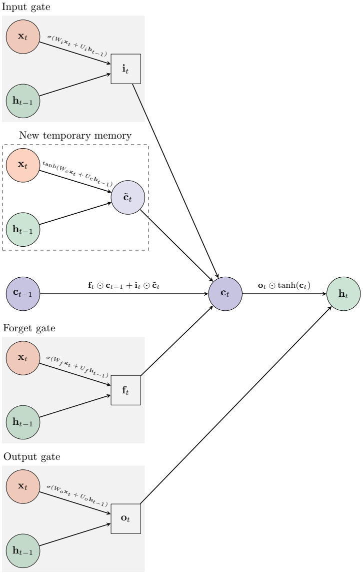
- **New temporary memory**: Use $\mathbf{x}_t$ and $\mathbf{h}_{t-1}$ to generate new memory that includes aspects of $\mathbf{x}_t$.
- **Input gate**: Use $\mathbf{x}_t$ and $\mathbf{h}_{t-1}$ to determine whether the temporary memory $\tilde{\mathbf{c}}_t$ is worth preserving.
- **Forget gate**: Assess whether the past memory cell $\mathbf{c}_{t-1}$ should be included in $\mathbf{c}_t$.
- **Updated memory state**: Use the forget and input gates to combine the new temporary memory and the current memory cell state to get $\mathbf{c}_t$.
- **Output gate**: Decides which part of $\mathbf{c}_t$ should be exposed to $\mathbf{h}_t$.

LSTMs basic unit


Input gate
$\mathbf{x}_t$
$\sigma(W_i\mathbf{x}_t + U_i\mathbf{h}_{t-1})$
$\mathbf{i}_t$
$\mathbf{h}_{t-1}$
New temporary memory
$\mathbf{x}_t$
$\tanh(W_c\mathbf{x}_t + U_c\mathbf{h}_{t-1})$
$\tilde{\mathbf{c}}_t$
$\mathbf{h}_{t-1}$
$\mathbf{c}_{t-1}$
$\mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t$
$\mathbf{c}_t$
$\mathbf{o}_t \odot \tanh(\mathbf{c}_t)$
$\mathbf{h}_t$
Forget gate
$\mathbf{x}_t$
$\sigma(W_f\mathbf{x}_t + U_f\mathbf{h}_{t-1})$
$\mathbf{f}_t$
$\mathbf{h}_{t-1}$
Output gate
$\mathbf{x}_t$
$\sigma(W_o\mathbf{x}_t + U_o\mathbf{h}_{t-1})$
$\mathbf{o}_t$
$\mathbf{h}_{t-1}$

- **New temporary memory**: Use $\mathbf{x}_t$ and $\mathbf{h}_{t-1}$ to generate new memory that includes aspects of $\mathbf{x}_t$.
- **Input gate**: Use $\mathbf{x}_t$ and $\mathbf{h}_{t-1}$ to determine whether the temporary memory $\tilde{\mathbf{c}}_t$ is worth preserving.
- **Forget gate**: Assess whether the past memory cell $\mathbf{c}_{t-1}$ should be included in $\mathbf{c}_t$.
- **Updated memory state**: Use the forget and input gates to combine the new temporary memory and the current memory cell state to get $\mathbf{c}_t$.
- **Output gate**: Decides which part of $\mathbf{c}_t$ should be exposed to $\mathbf{h}_t$.

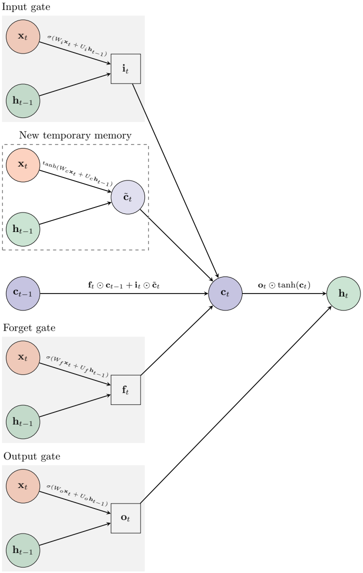
LSTMs basic unit



Input gate
$\mathbf{x}_t$
$\sigma(W_i\mathbf{x}_t + U_i\mathbf{h}_{t-1})$
$\mathbf{i}_t$
$\mathbf{h}_{t-1}$
New temporary memory
$\mathbf{x}_t$
$\tanh(W_c\mathbf{x}_t + U_c\mathbf{h}_{t-1})$
$\tilde{\mathbf{c}}_t$
$\mathbf{h}_{t-1}$
$\mathbf{c}_{t-1}$
$\mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t$
$\mathbf{c}_t$
$\mathbf{o}_t \odot \tanh(\mathbf{c}_t)$
$\mathbf{h}_t$
Forget gate
$\mathbf{x}_t$
$\sigma(W_f\mathbf{x}_t + U_f\mathbf{h}_{t-1})$
$\mathbf{f}_t$
$\mathbf{h}_{t-1}$
Output gate
$\mathbf{x}_t$
$\sigma(W_o\mathbf{x}_t + U_o\mathbf{h}_{t-1})$
$\mathbf{o}_t$
$\mathbf{h}_{t-1}$

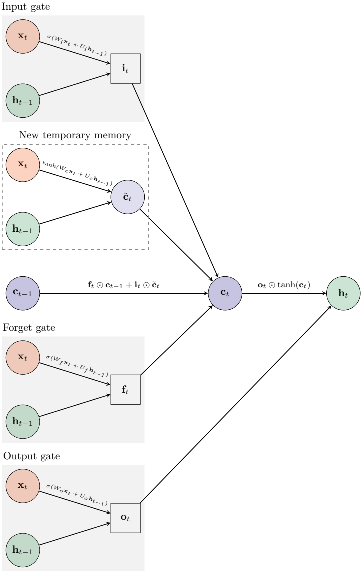
- **New temporary memory**: Use $\mathbf{x}_t$ and $\mathbf{h}_{t-1}$ to generate new memory that includes aspects of $\mathbf{x}_t$.
- **Input gate**: Use $\mathbf{x}_t$ and $\mathbf{h}_{t-1}$ to determine whether the temporary memory $\tilde{\mathbf{c}}_t$ is worth preserving.
- **Forget gate**: Assess whether the past memory cell $\mathbf{c}_{t-1}$ should be included in $\mathbf{c}_t$.
- **Updated memory state**: Use the forget and input gates to combine the new temporary memory and the current memory cell state to get $\mathbf{c}_t$.
- **Output gate**: Decides which part of $\mathbf{c}_t$ should be exposed to $\mathbf{h}_t$.

LSTMs basic unit


Input gate
$\mathbf{x}_t$
$\sigma(W_i\mathbf{x}_t + U_i\mathbf{h}_{t-1})$
$\mathbf{i}_t$
$\mathbf{h}_{t-1}$
New temporary memory
$\mathbf{x}_t$
$\tanh(W_c\mathbf{x}_t + U_c\mathbf{h}_{t-1})$
$\tilde{\mathbf{c}}_t$
$\mathbf{h}_{t-1}$
$\mathbf{c}_{t-1}$
$\mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t$
$\mathbf{c}_t$
$\mathbf{o}_t \odot \tanh(\mathbf{c}_t)$
$\mathbf{h}_t$
Forget gate
$\mathbf{x}_t$
$\sigma(W_f\mathbf{x}_t + U_f\mathbf{h}_{t-1})$
$\mathbf{f}_t$
$\mathbf{h}_{t-1}$
Output gate
$\mathbf{x}_t$
$\sigma(W_o\mathbf{x}_t + U_o\mathbf{h}_{t-1})$
$\mathbf{o}_t$
$\mathbf{h}_{t-1}$

- **New temporary memory**: Use $\mathbf{x}_t$ and $\mathbf{h}_{t-1}$ to generate new memory that includes aspects of $\mathbf{x}_t$.
- **Input gate**: Use $\mathbf{x}_t$ and $\mathbf{h}_{t-1}$ to determine whether the temporary memory $\tilde{\mathbf{c}}_t$ is worth preserving.
- **Forget gate**: Assess whether the past memory cell $\mathbf{c}_{t-1}$ should be included in $\mathbf{c}_t$.
- **Updated memory state**: Use the forget and input gates to combine the new temporary memory and the current memory cell state to get $\mathbf{c}_t$.
- **Output gate**: Decides which part of $\mathbf{c}_t$ should be exposed to $\mathbf{h}_t$.

LSTMs basic unit

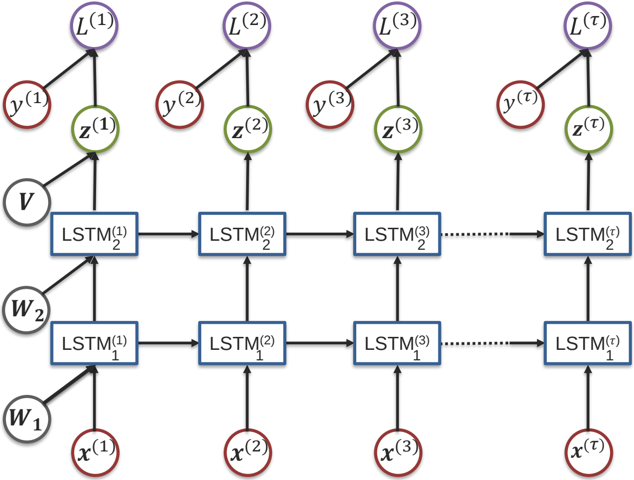


Input gate
$\mathbf{x}_t$
$\sigma(W_i\mathbf{x}_t + U_i\mathbf{h}_{t-1})$
$\mathbf{i}_t$
$\mathbf{h}_{t-1}$
New temporary memory
$\mathbf{x}_t$
$\tanh(W_c\mathbf{x}_t + U_c\mathbf{h}_{t-1})$
$\tilde{\mathbf{c}}_t$
$\mathbf{h}_{t-1}$
$\mathbf{c}_{t-1}$
$\mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t$
$\mathbf{c}_t$
$\mathbf{o}_t \odot \tanh(\mathbf{c}_t)$
$\mathbf{h}_t$
Forget gate
$\mathbf{x}_t$
$\sigma(W_f\mathbf{x}_t + U_f\mathbf{h}_{t-1})$
$\mathbf{f}_t$
$\mathbf{h}_{t-1}$
Output gate
$\mathbf{x}_t$
$\sigma(W_o\mathbf{x}_t + U_o\mathbf{h}_{t-1})$
$\mathbf{o}_t$
$\mathbf{h}_{t-1}$

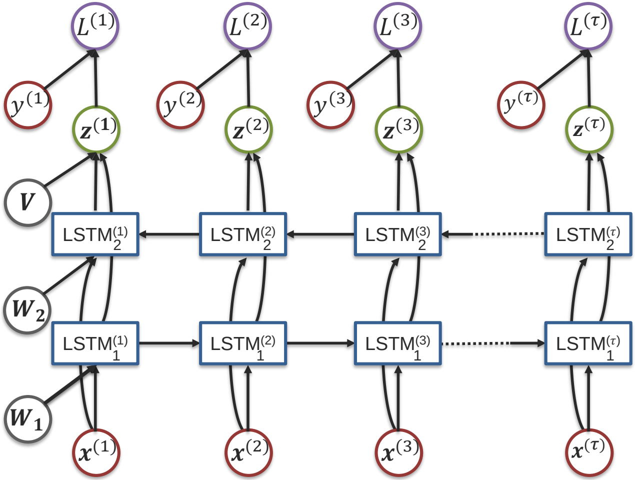
- **New temporary memory**: Use $\mathbf{x}_t$ and $\mathbf{h}_{t-1}$ to generate new memory that includes aspects of $\mathbf{x}_t$.
- **Input gate**: Use $\mathbf{x}_t$ and $\mathbf{h}_{t-1}$ to determine whether the temporary memory $\tilde{\mathbf{c}}_t$ is worth preserving.
- **Forget gate**: Assess whether the past memory cell $\mathbf{c}_{t-1}$ should be included in $\mathbf{c}_t$.
- **Updated memory state**: Use the forget and input gates to combine the new temporary memory and the current memory cell state to get $\mathbf{c}_t$.
- **Output gate**: Decides which part of $\mathbf{c}_t$ should be exposed to $\mathbf{h}_t$.

Can go deep with LSTMs

Deep LSTM Network



Bi-directional LSTM Network



Summary

- RNNs allow a lot of flexibility in architecture design
- Backward flow of gradients in RNN can explode or vanish.
- Vanilla RNNs are simple but find it hard to learn long-term dependencies.
- Exploding is controlled with gradient clipping. Vanishing is controlled with additive interactions (LSTM)
- Common to use LSTMs: their additive interactions improve gradient flow