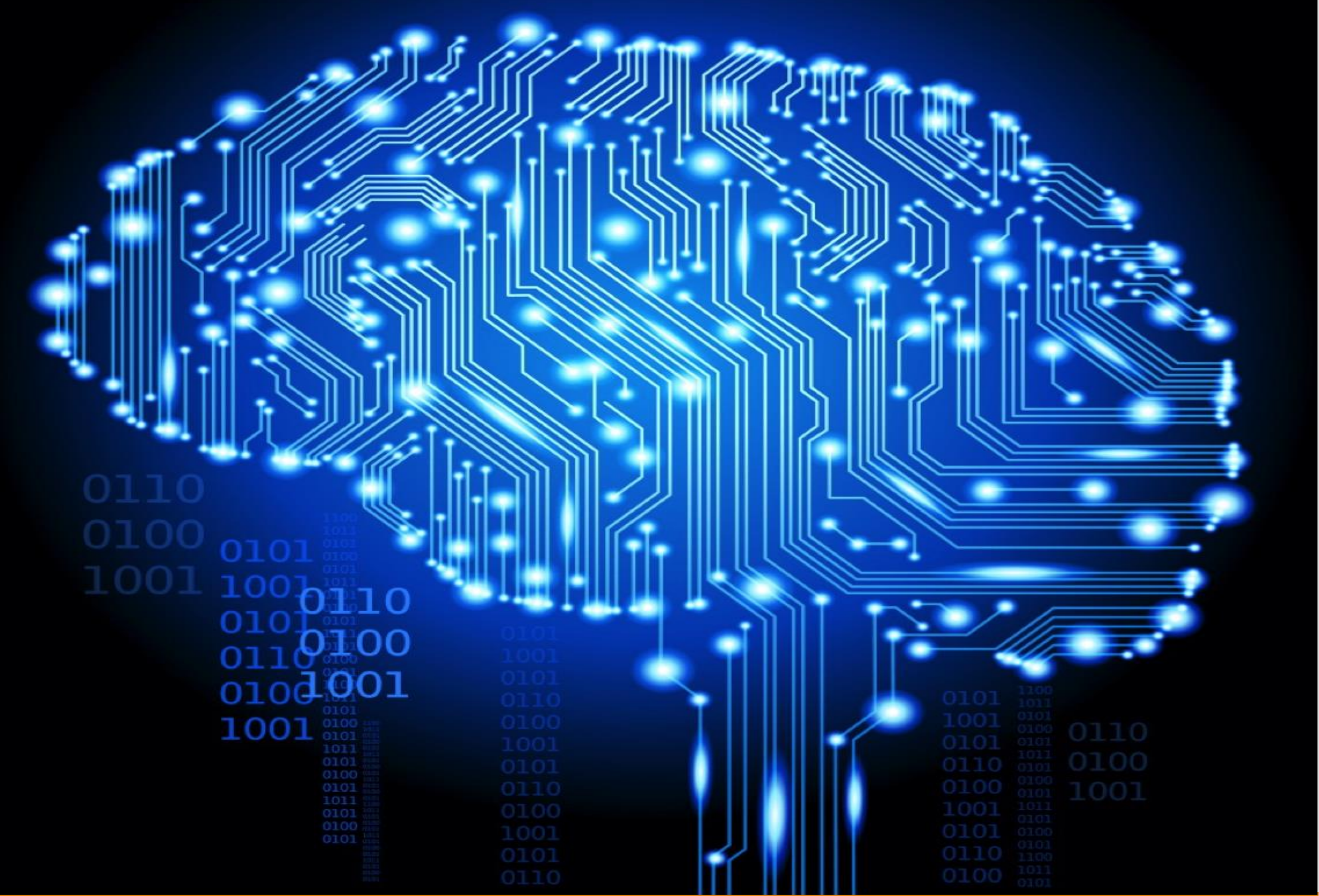


Lecture 9: Recurrent Neural Networks

Deep Learning @ UvA

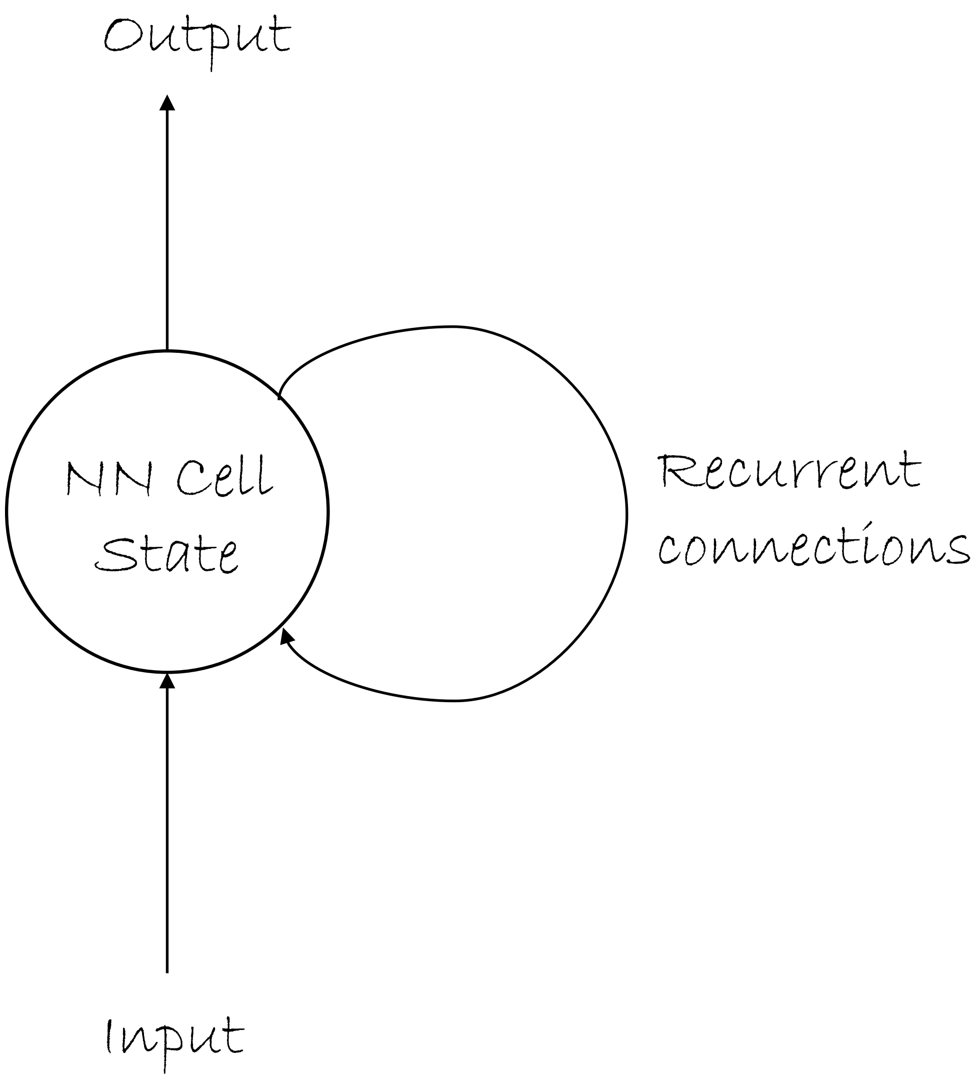
Previous Lecture

- Word and Language Representations
- From n-grams to Neural Networks
- Word2vec
- Skip-gram

Lecture Overview

- Recurrent Neural Networks (RNN) for sequences
- Backpropagation Through Time
- RNNs using Long Short-Term Memory (LSTM)
- Applications of Recurrent Neural Networks

Recurrent Neural Networks



Sequential vs. static data

- Abusing the terminology
  - static data come in a mini-batch of size D
  - dynamic data come in a mini-batch of size 1

Sequential vs. static data

- Abusing the terminology
  - static data come in a mini-batch of size D
  - dynamic data come in a mini-batch of size 1

What about inputs that appear in sequences, such as text? Could a neural network handle such modalities?

Modelling sequences is ...

- ... roughly equivalent to predicting what is going to happen next

$$\Pr(x) = \prod_i \Pr(x_i | x_1, \dots, x_{i-1})$$

- Easy to generalize to sequences of arbitrary length
- Considering small chunks x_i → fewer parameters, easier modelling
- Often is convenient to pick a "frame" T

$$\Pr(x) = \prod_i \Pr(x_i | x_{i-T}, \dots, x_{i-1})$$

Word representations

- One-hot vector
  - After one-hot vector add an embedding
- Instead of one-hot vector use directly a word representation
  - Word2vec
  - GloVE

Vocabulary	One-hot vectors							
I	I	1	I	0	I	0	I	0
am	am	0	am	1	am	0	am	0
Bond	Bond	0	Bond	0	Bond	1	Bond	0
James	James	0	James	0	James	0	James	1
tired	tired	0	tired	0	tired	0	tired	0
,	,	0	,	0	,	0	,	0
McGuire	McGuire	0	McGuire	0	McGuire	0	McGuire	0
!	!	0	!	0	!	0	!	0

What a sequence *really* is?

- Data inside a sequence are non i.i.d.
  - Identically, independently distributed
- The next "word" depends on the previous "words"
  - Ideally on all of them
- We need **context,** and we need **memory!**
- How to model context and memory ?

I am Bond , James $\boxed{}$

What a sequence *really* is?

- Data inside a sequence are non i.i.d.
  - Identically, independently distributed
- The next "word" depends on the previous "words"
  - Ideally on all of them
- We need **context,** and we need **memory!**
- How to model context and memory ?

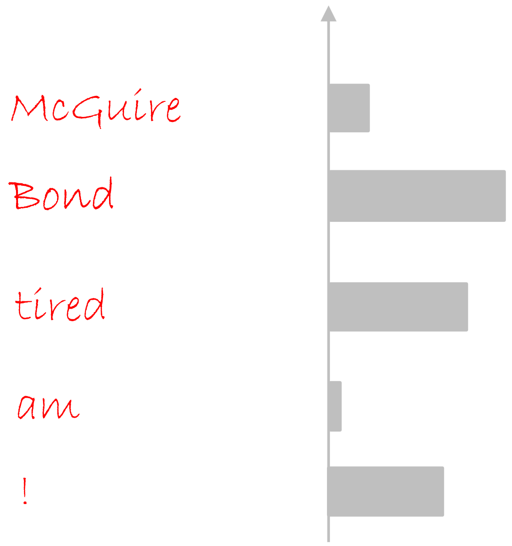
I am Bond , James $\boxed{}$

McGuire
Bond
tired
am
!

What a sequence *really* is?

- Data inside a sequence are non i.i.d.
  - Identically, independently distributed
- The next "word" depends on the previous "words"
  - Ideally on all of them
- We need **context,** and we need **memory!**
- How to model context and memory ?

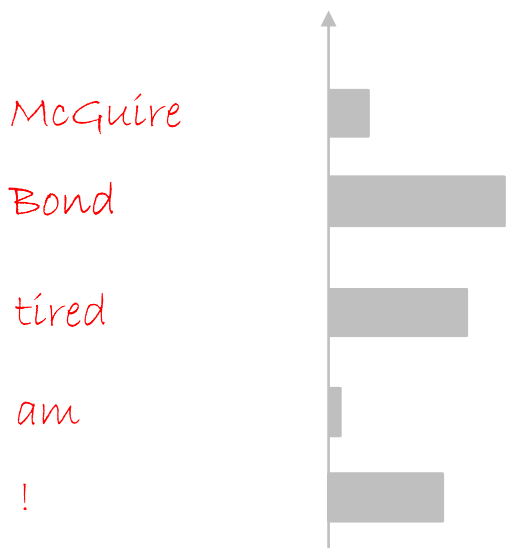
I am Bond , James $\boxed{}$



What a sequence *really* is?

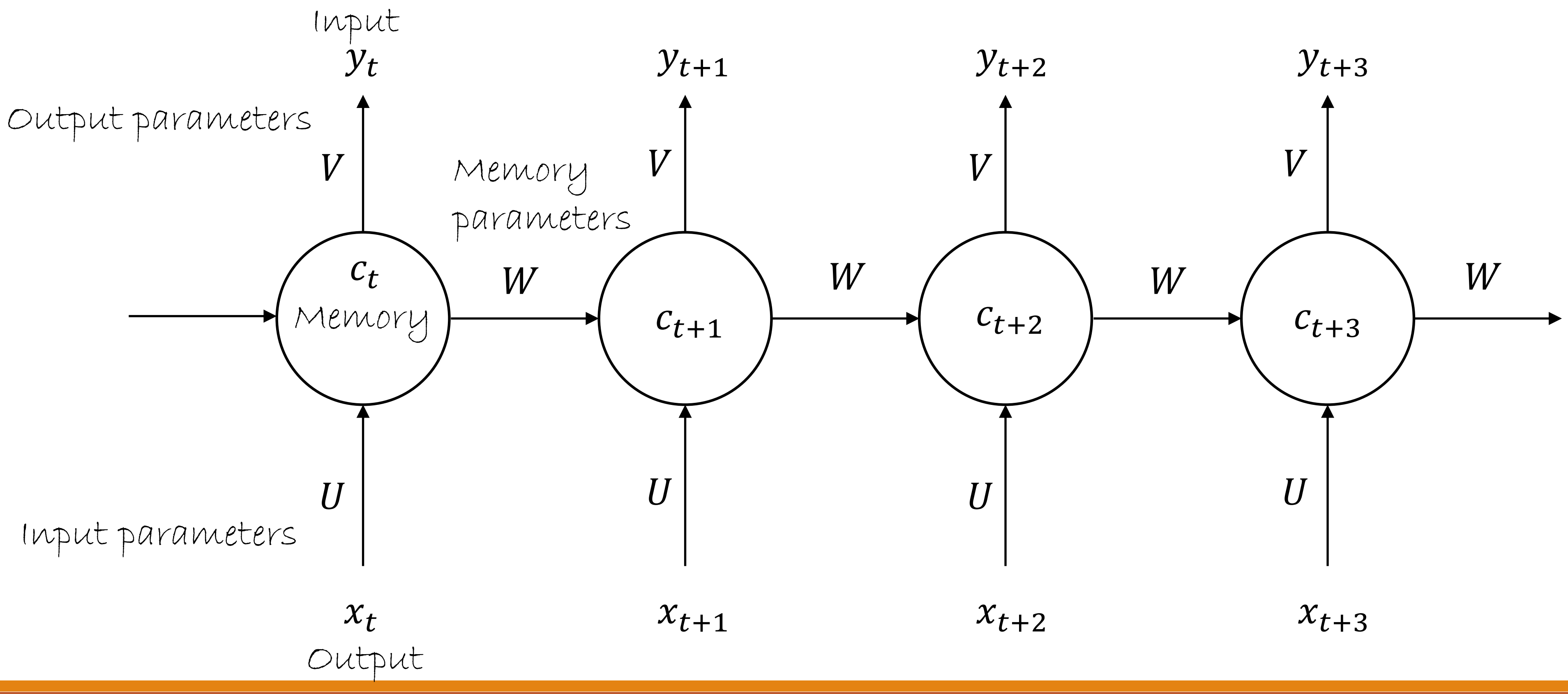
- Data inside a sequence are non i.i.d.
  - Identically, independently distributed
- The next "word" depends on the previous "words"
  - Ideally on all of them
- We need **context,** and we need **memory!**
- How to model context and memory ?

I am Bond , James [Bond]

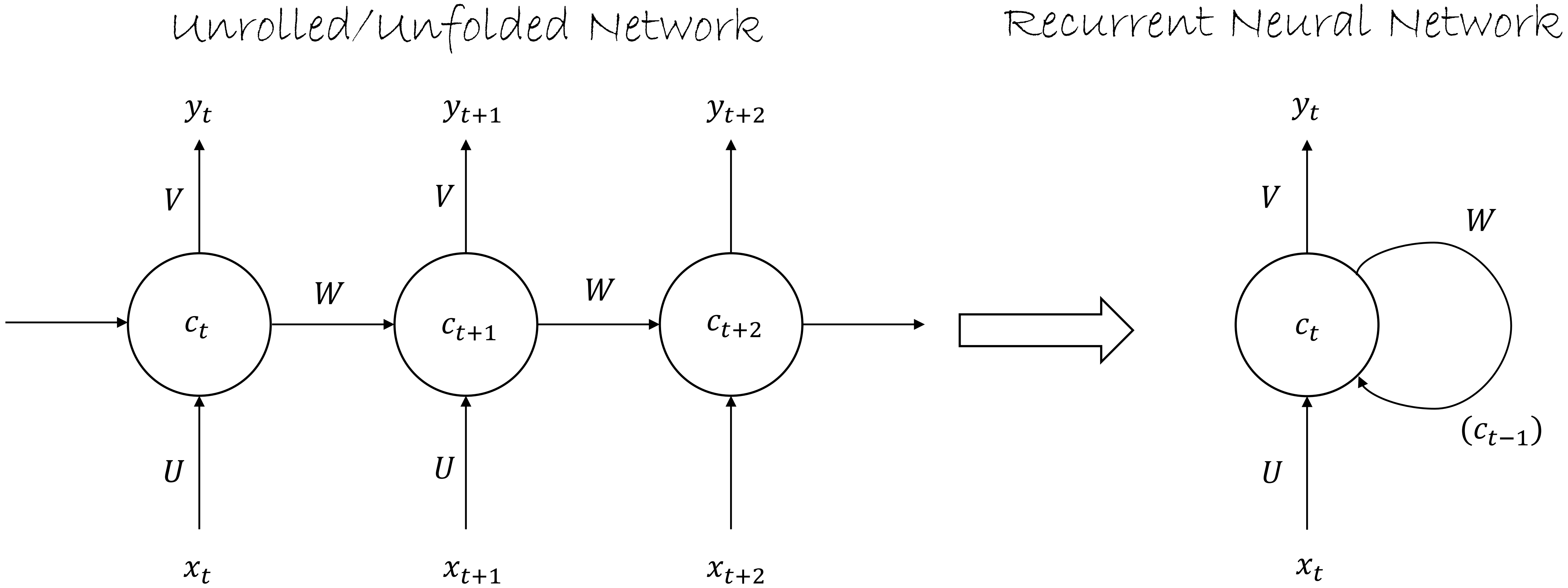


Modelling-wise, what is memory?

- A representation (variable) of the past
- How to adapt a simple Neural Network to include a memory variable?

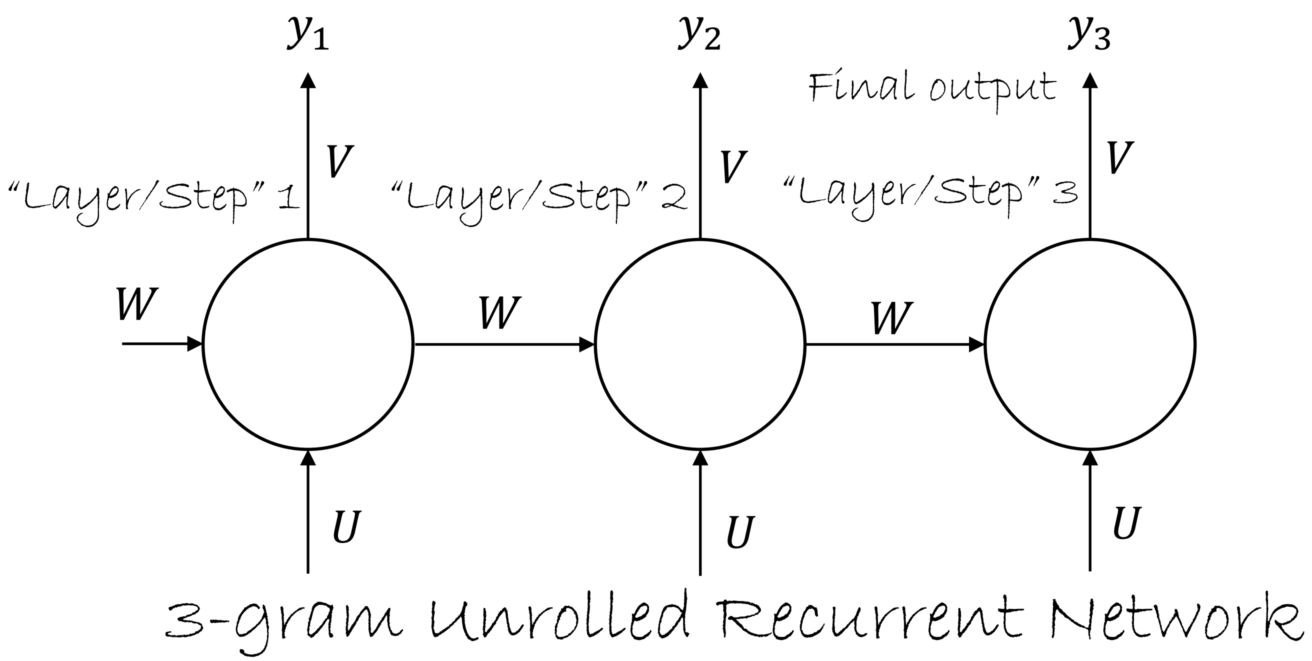


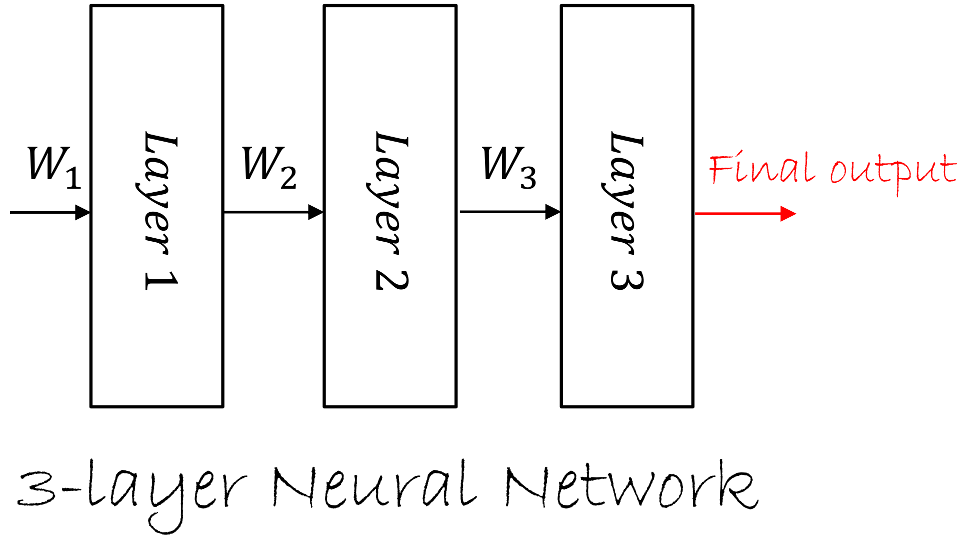
Recurrent Neural Network (RNN)



Why unrolled?

- Imagine we care only for 3-grams
- Are the two networks that different?
  - Steps instead of layers
  - Step parameters are same (shared parameters in a NN)

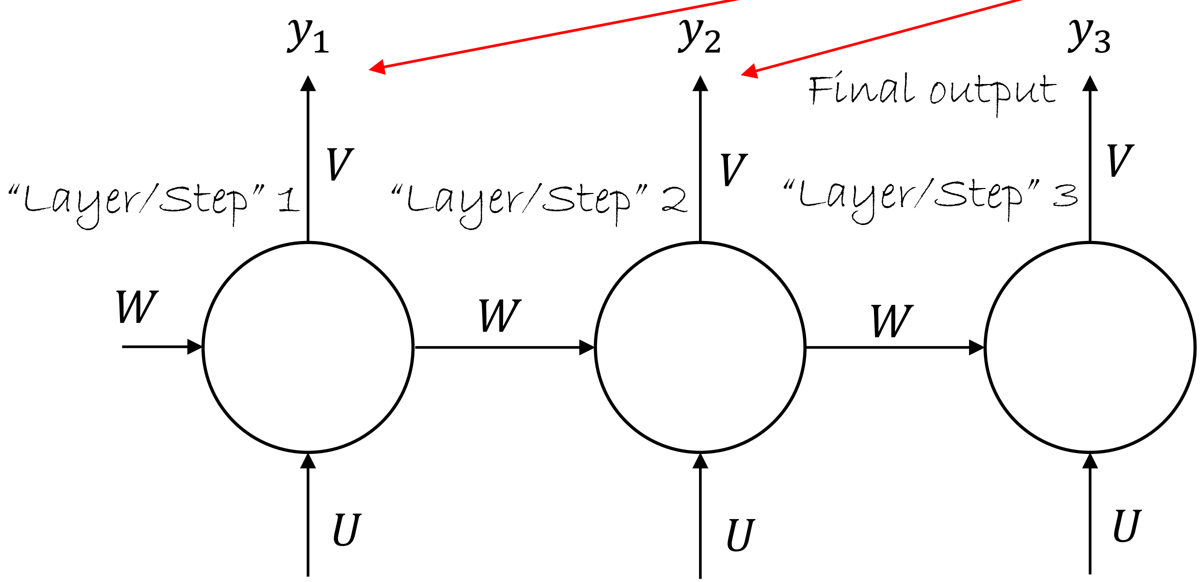




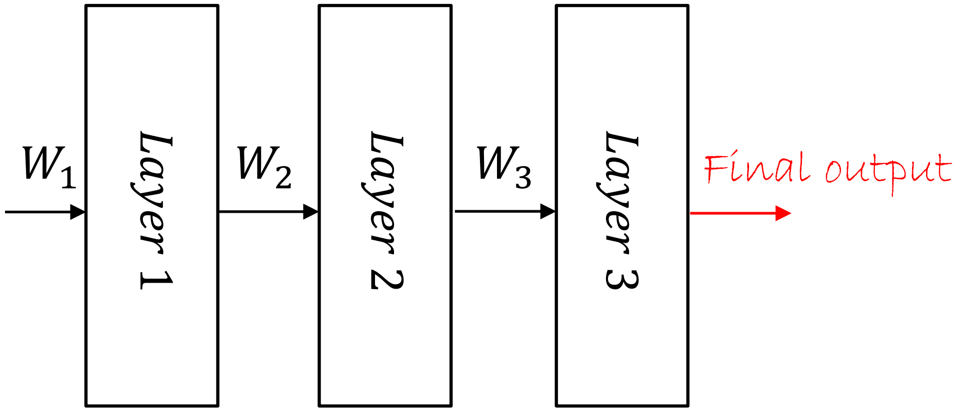
Why unrolled?

- Imagine we care only for 3-grams
- Are the two networks that different?
  - Steps instead of layers
  - Step parameters are same (shared parameters in a NN)

- Sometimes intermediate outputs are not even needed



3-gram Unrolled Recurrent Network

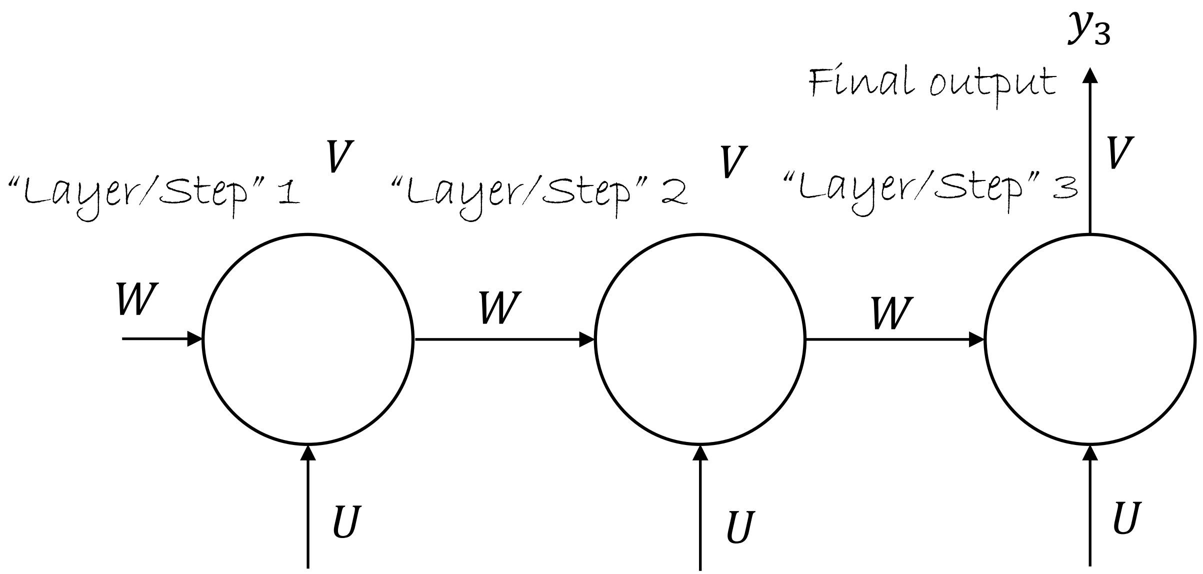


3-layer Neural Network

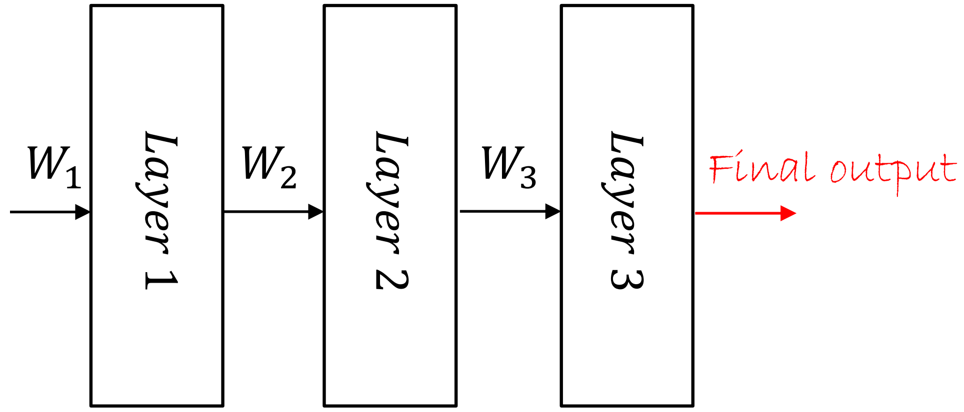
Why unrolled?

- Imagine we care only for 3-grams
- Are the two networks that different?
  - Steps instead of layers
  - Step parameters are same (shared parameters in a NN)

- *Sometimes intermediate outputs are not even needed*
- *Removing them, we almost end up to a standard Neural Network*



3-gram Unrolled Recurrent Network

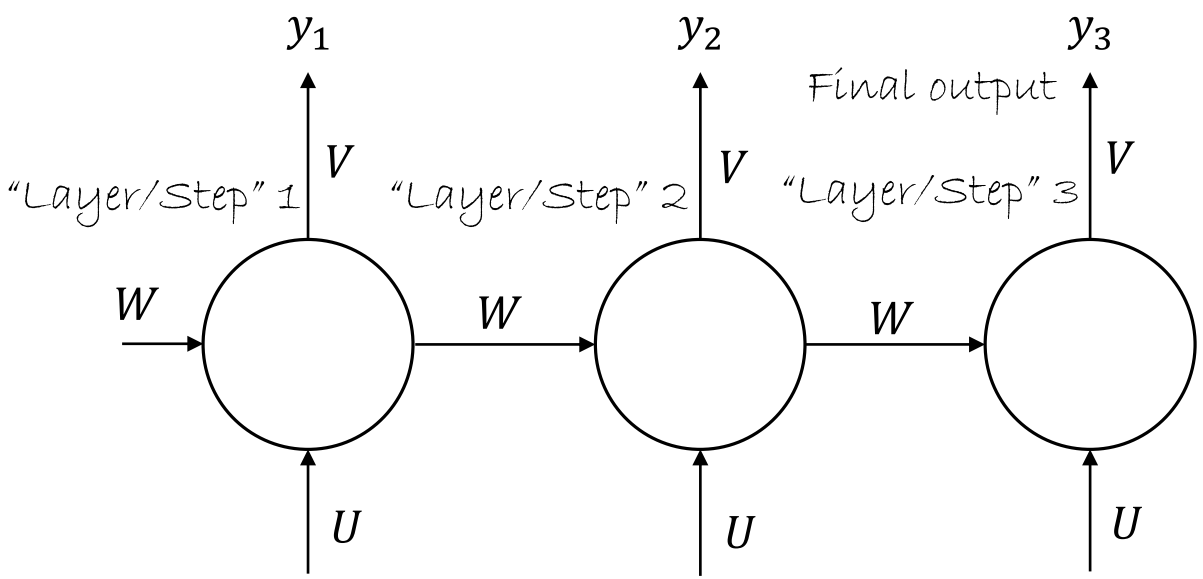


3-layer Neural Network

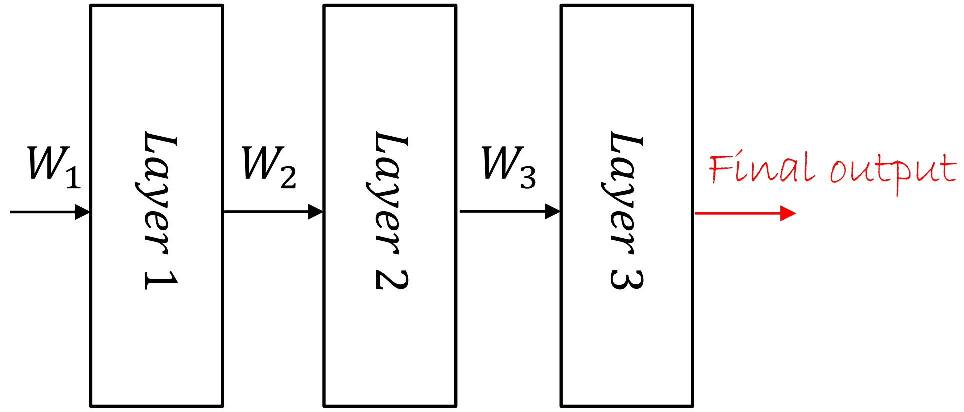
Why unrolled?

- Imagine we care only for 3-grams
- Are the two networks that different?
  - Steps instead of layers
  - Step parameters are same (shared parameters in a NN)

- Sometimes intermediate outputs are not even needed
- Removing them, we almost end up to a standard Neural Network



3-gram Unrolled Recurrent Network

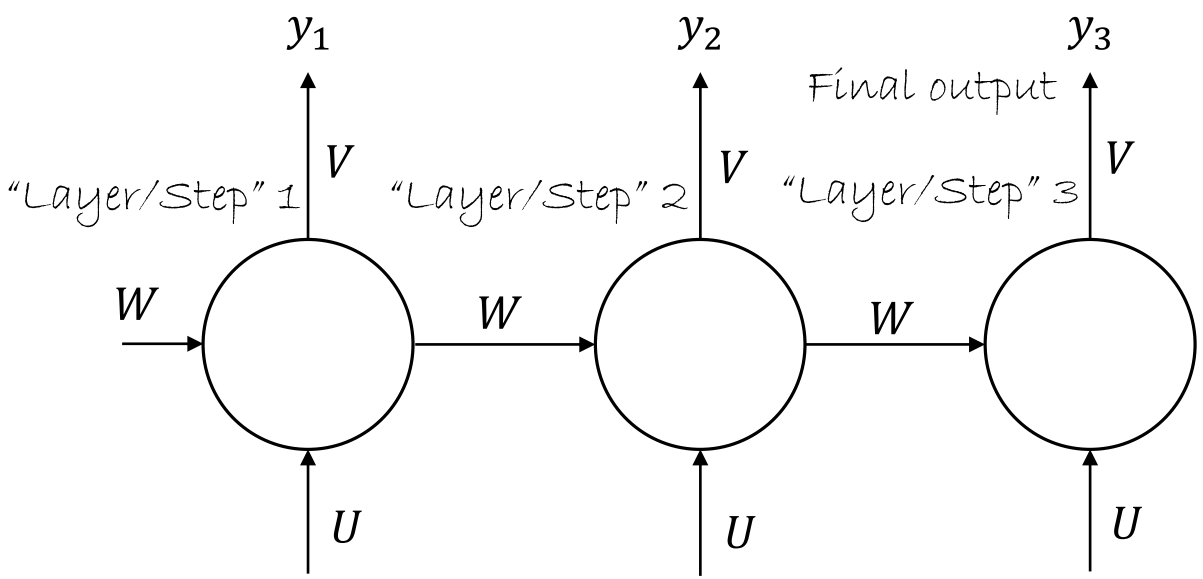


3-layer Neural Network

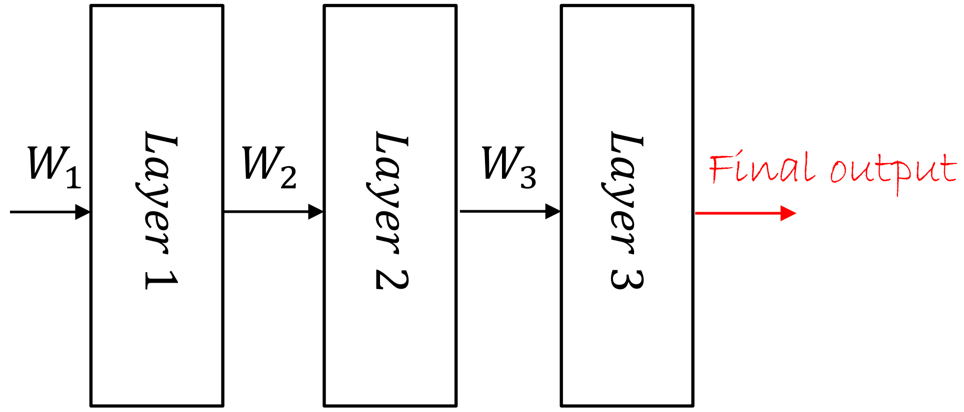
Why unrolled?

- Imagine we care only for 3-grams
- Are the two networks that different?
  - Steps instead of layers
  - Step parameters are same (shared parameters in a NN)

- Sometimes intermediate outputs are not even needed
- Removing them, we almost end up to a standard Neural Network



3-gram Unrolled Recurrent Network

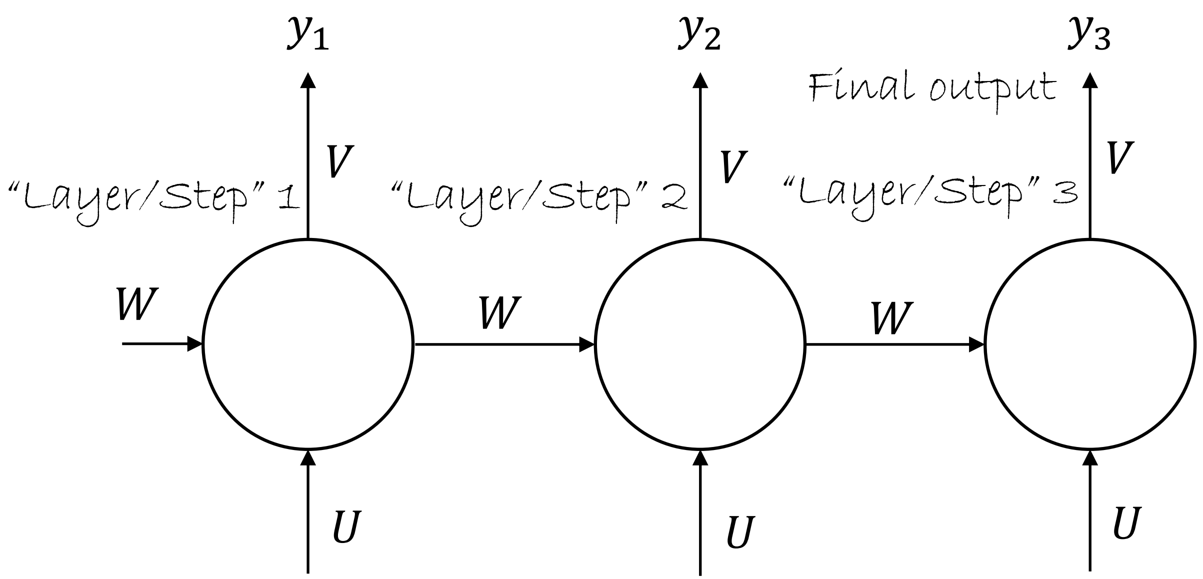


3-layer Neural Network

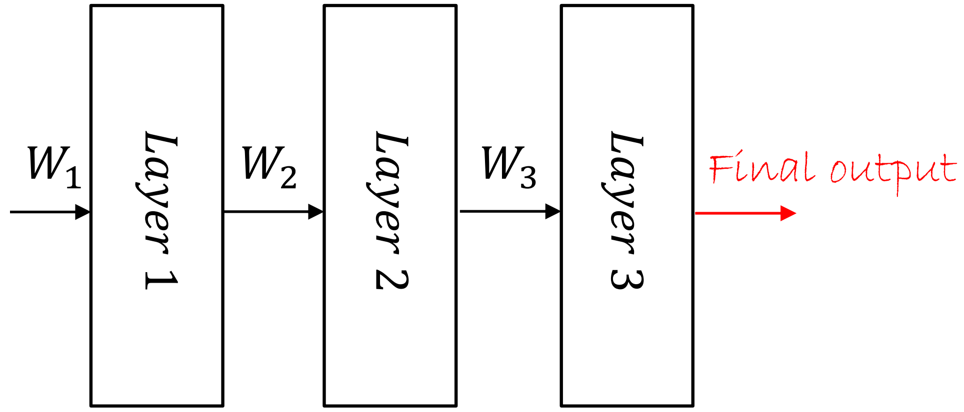
Why unrolled?

- Imagine we care only for 3-grams
- Are the two networks that different?
  - Steps instead of layers
  - Step parameters are same (shared parameters in a NN)

- Sometimes intermediate outputs are not even needed
- Removing them, we almost end up to a standard Neural Network



3-gram Unrolled Recurrent Network



3-layer Neural Network

RNN equations

- At time step t

One-hot vector

$$\boxed{\begin{aligned} c_t &= \tanh(U\,x_t + W c_{t-1}) \\ y_t &= \text{softmax}(V c_t) \end{aligned}}$$

$$\begin{array}{cc} U & x_t \\ \begin{bmatrix} 0.1 & -0.3 & 1.2 & 0.6 & -0.8 \\ -0.2 & 0.4 & 0.5 & 0.9 & -0.1 \\ -0.1 & 0.2 & -0.7 & -0.8 & 0.3 \end{bmatrix} \cdot & \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \end{array}$$

$$= \begin{bmatrix} 0.6 \\ 0.9 \\ -0.8 \end{bmatrix} = U^{(4)}$$

- Example
  - Vocabulary of 500 words
  - An input projection of 50 dimensions (U: $[50 \times 500]$)
  - A memory of 128 units (c_t: $[128 \times 1]$, W: $[128 \times 128]$)
  - An output projections of 500 dimensions (V: $[500 \times 128]$)

Loss function

- Cross entropy loss

$$P = \prod_t \prod_k y_{tk}^{l_{tk}} \quad \Rightarrow \quad \mathcal{L} = -\log P = -\frac{1}{T}\sum_t l_t \log y_t \,,$$

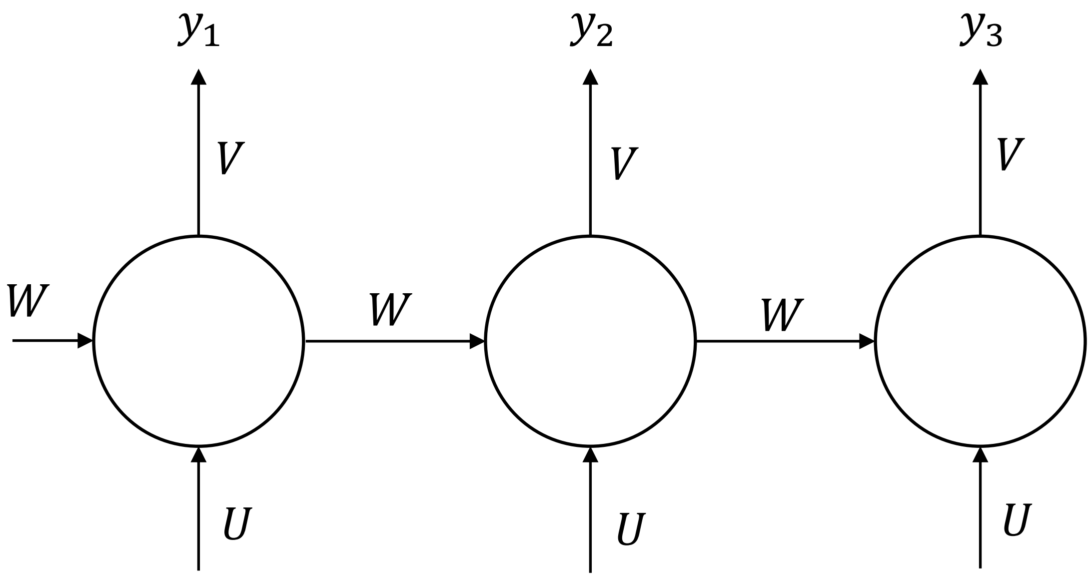
  - non-target words $l_t = 0 \Rightarrow$ Compute loss only for target words
  - Random loss $=? \Rightarrow \mathcal{L} = \log K$

- Usually loss is measured w.r.t. **perplexity**

$$\boxed{J = 2^{\mathcal{L}}}$$

Training an RNN

- Backpropagation Through Time (BPTT)

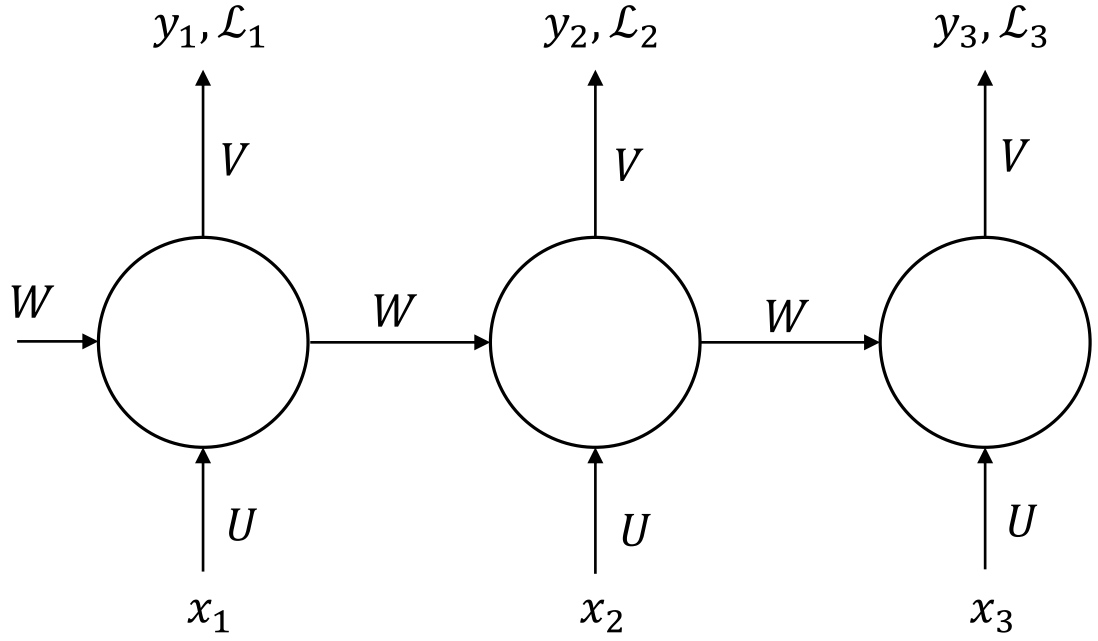


Backpropagation Through Time

- $\frac{\partial \mathcal{L}}{\partial V}, \frac{\partial \mathcal{L}}{\partial W}, \frac{\partial \mathcal{L}}{\partial U}$
- To make it simpler let's focus on step 3

$$\frac{\partial \mathcal{L}_3}{\partial V}, \frac{\partial \mathcal{L}_3}{\partial W}, \frac{\partial \mathcal{L}_3}{\partial U}$$

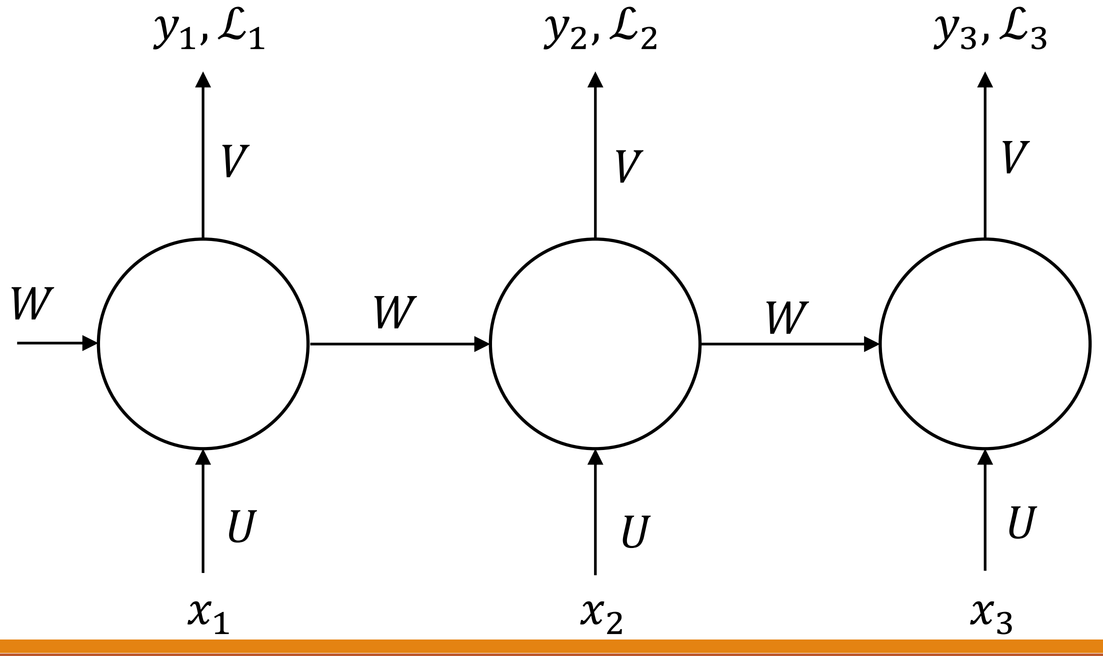
$$c_t = \tanh(U\,x_t + W c_{t-1})$$
$$y_t = \text{softmax}(V c_t)$$
$$\mathcal{L} = -\sum_t l_t \log y_t = \sum_t \mathcal{L}_t$$



Backpropagation Through Time

$$\frac{\partial \mathcal{L}_3}{\partial V} = \frac{\partial \mathcal{L}_3}{\partial y_3}\frac{\partial y_3}{\partial V} = (y_3 - l_3) \times c_3$$

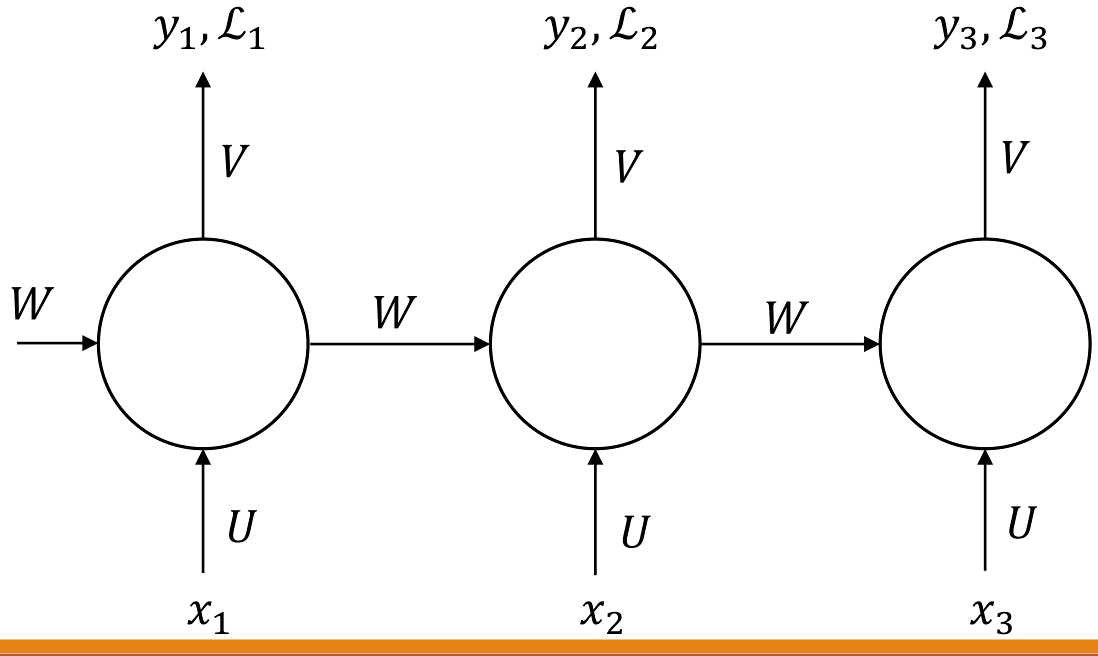
$$c_t = \tanh(U\,x_t + W c_{t-1})$$
$$y_t = \text{softmax}(V c_t)$$
$$\mathcal{L} = -\sum_t l_t \log y_t = \sum_t \mathcal{L}_t$$



Backpropagation Through Time

- $\frac{\partial \mathcal{L}_3}{\partial W} = \frac{\partial \mathcal{L}_3}{\partial y_3} \frac{\partial y_3}{\partial c_3} \frac{\partial c_3}{\partial W}$
- What is the relation between c_3 and W?
  - Two-fold: $c_t = \tanh(U\, x_t + W c_{t-1})$
- $\frac{\partial\, f(\varphi(x), \psi(x))}{\partial x} = \frac{\partial f}{\partial \varphi} \frac{\partial \varphi}{\partial x} + \frac{\partial f}{\partial \psi} \frac{\partial \psi}{\partial x}$
- $\frac{\partial c_3}{\partial W} \propto c_2 + \frac{\partial c_2}{\partial W} \quad \left(\frac{\partial W}{\partial W} = 1\right)$

$$c_t = \tanh(U\, x_t + W c_{t-1})$$
$$y_t = \text{softmax}(V c_t)$$
$$\mathcal{L} = -\sum_t l_t \log y_t = \sum_t \mathcal{L}_t$$

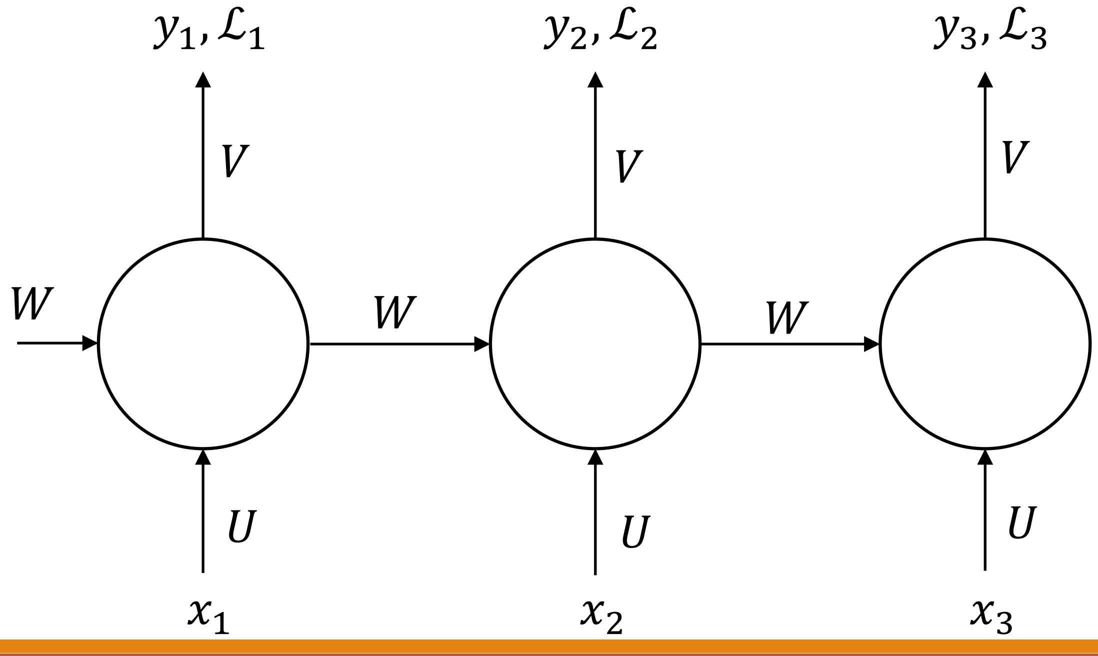


Recursively

- $\frac{\partial c_3}{\partial W} = c_2 + \frac{\partial c_2}{\partial W}$
- $\frac{\partial c_2}{\partial W} = c_1 + \frac{\partial c_1}{\partial W}$
- $\frac{\partial c_1}{\partial W} = c_0 + \frac{\partial c_0}{\partial W}$

$$\frac{\partial c_3}{\partial W} = \sum_{t=1}^{3} \frac{\partial c_3}{\partial c_t} \frac{\partial c_t}{\partial W} \Rightarrow \boxed{\frac{\partial \mathcal{L}_3}{\partial W} = \sum_{t=1}^{3} \frac{\partial \mathcal{L}_3}{\partial y_3} \frac{\partial y_3}{\partial c_3} \frac{\partial c_3}{\partial c_t} \frac{\partial c_t}{\partial W}}$$

$$c_t = \tanh(U\, x_t + W c_{t-1})$$
$$y_t = \mathrm{softmax}(V c_t)$$
$$\mathcal{L} = -\sum_t l_t \log y_t = \sum_t \mathcal{L}_t$$



Are RNNs perfect?

- NO
  - Although in theory yes!
- Vanishing gradient
  - After a few time steps the gradients become almost 0
- Exploding gradient
  - After a few time steps the gradients become huge
- Can’t really capture long-term dependencies

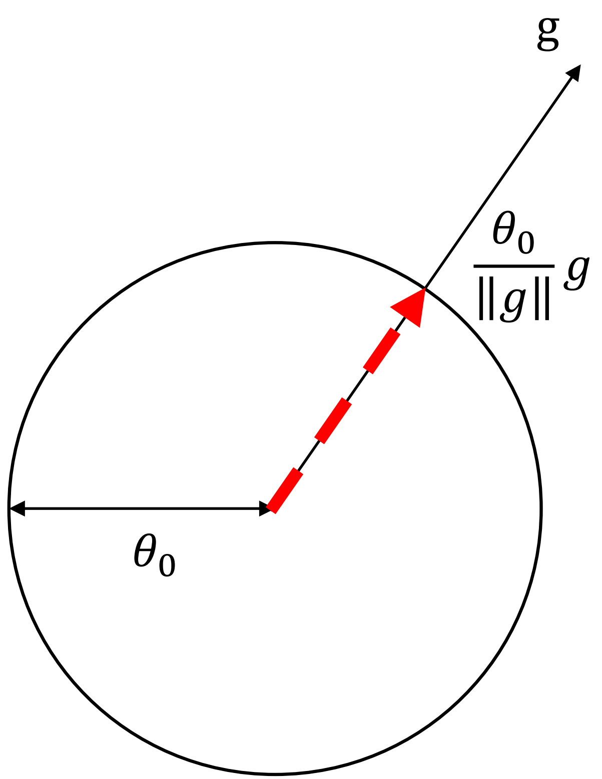
Exploding gradients

- $\frac{\partial \mathcal{L}_\mathrm{r}}{\partial W} = \sum_{t=1}^{\mathrm{r}} \frac{\partial \mathcal{L}_r}{\partial y_r} \frac{\partial y_r}{\partial c_r} \frac{\partial c_r}{\partial c_t} \frac{\partial c_t}{\partial W}$
- $\frac{\partial c_r}{\partial c_t}$ can be decomposed further based on the chain rule
- $$\frac{\partial c_r}{\partial c_t} = \underbrace{\frac{\partial c_r}{\partial c_{r-1}} \cdot \frac{\partial c_{r-1}}{\partial c_{r-2}}}_{Rest \;\rightarrow\; \text{long-term factors}} \cdot \ldots \cdot \underbrace{\frac{\partial c_{t+1}}{\partial c_t}}_{\tau \gg r \;\rightarrow\; \text{short-term factors}}$$
- When many long-term factors, for many of which $\frac{\partial c_i}{\partial c_{i-1}} \gg 1$
  - then $\left\| \frac{\partial c_r}{\partial c_t} \right\| \gg 1$
  - then $\left\| \frac{\partial \mathcal{L}_\mathrm{r}}{\partial W} \right\| \gg 1$

Gradient explodes

Remedy for exploding gradients

- Scale the gradients to a threshold
- Step 1. $\mathrm{g} \leftarrow \frac{\partial \mathcal{L}}{\partial \theta}$
- Step 2. Is $||g|| > \theta_0$?
  - Step 3a. If yes $\mathrm{g} \leftarrow \frac{\theta_0}{||g||} g$
  - Step 3b. If no, then do nothing
- Simple, but works!

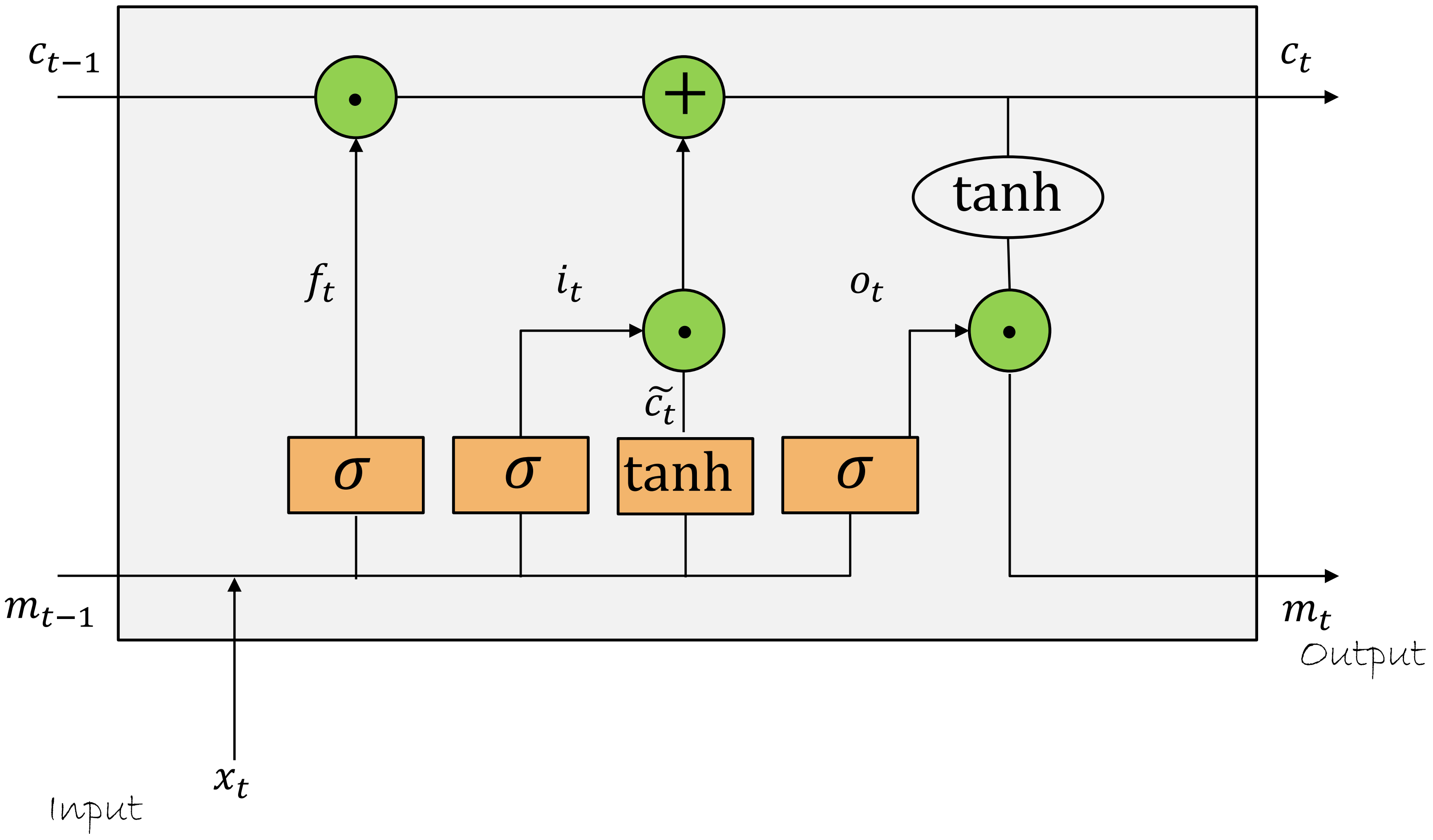


Vanishing gradients

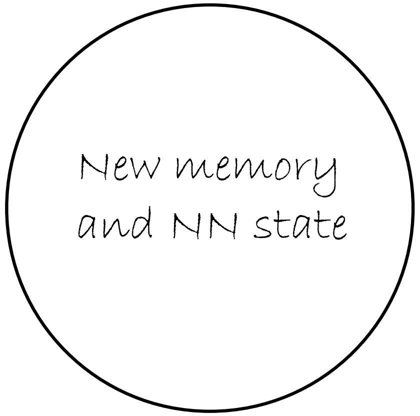
- $\frac{\partial \mathcal{L}_r}{\partial W} = \sum_{t=1}^{r} \frac{\partial \mathcal{L}_r}{\partial y_r} \frac{\partial y_r}{\partial c_r} \frac{\partial c_r}{\partial c_t} \frac{\partial c_t}{\partial W}$
- $\frac{\partial c_r}{\partial c_t}$ can be decomposed further based on the chain rule
- $\frac{\partial c_r}{\partial c_t} = \underbrace{\frac{\partial c_r}{\partial c_{r-1}} \cdot \frac{\partial c_{r-1}}{\partial c_{r-2}}}_{Rest \;\rightarrow\; \text{long-term factors}} \cdot \ldots \cdot \underbrace{\frac{\partial c_{t+1}}{\partial c_t}}_{\tau \gg r \;\rightarrow\; \text{short-term factors}}$
- When many $\frac{\partial c_i}{\partial c_{i-1}} \rightarrow 0$ (e.g. with sigmoids),
  - then $\frac{\partial c_r}{\partial c_t} \rightarrow 0$
  - then $\frac{\partial \mathcal{L}_r}{\partial W} \rightarrow 0$

Gradient vanishes

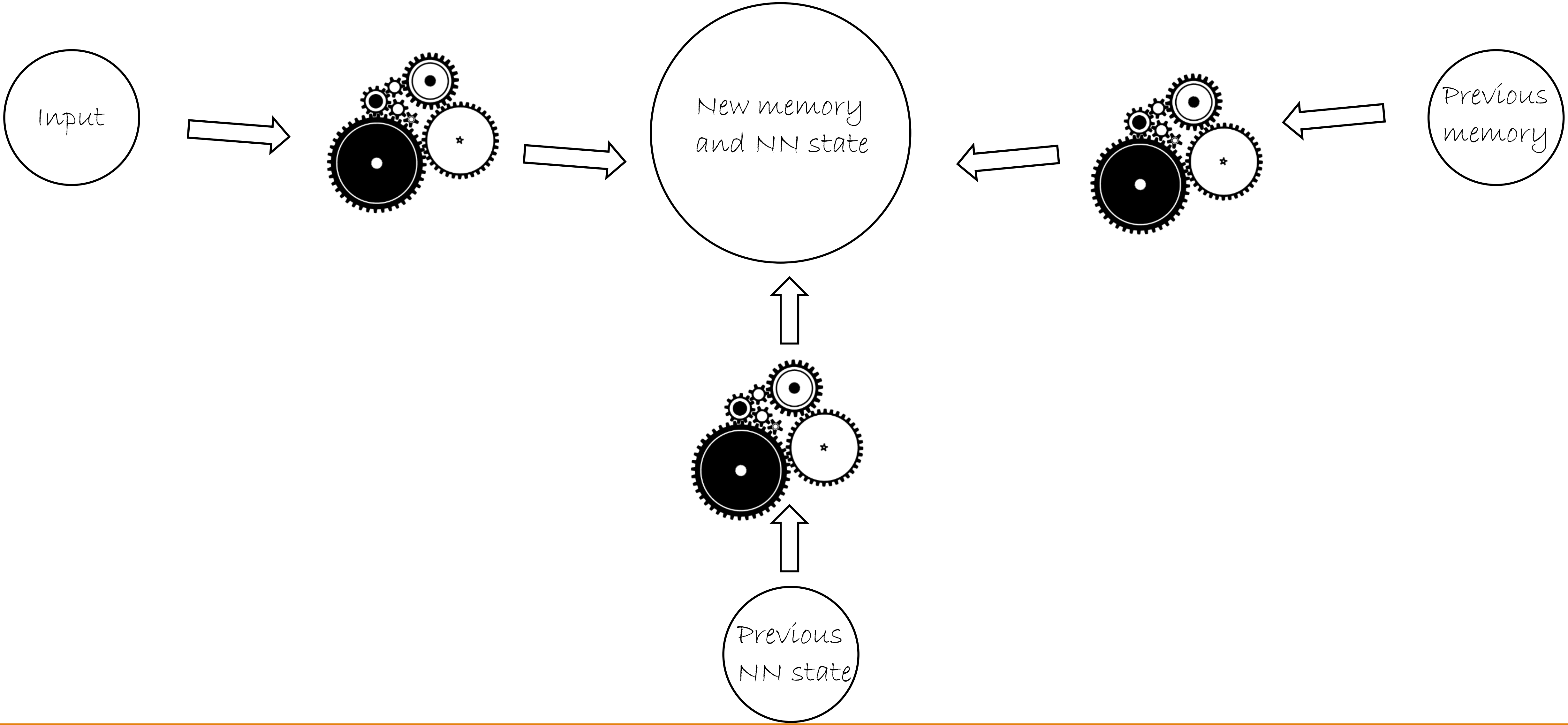
Advanced RNNs



A more realistic memory unit needs what?



A more realistic memory unit needs what?



Long Short-Termn Memory (LSTM: Beefed up RNN)

$$i = \sigma\left(x_t U^{(i)} + m_{t-1} W^{(i)}\right)$$

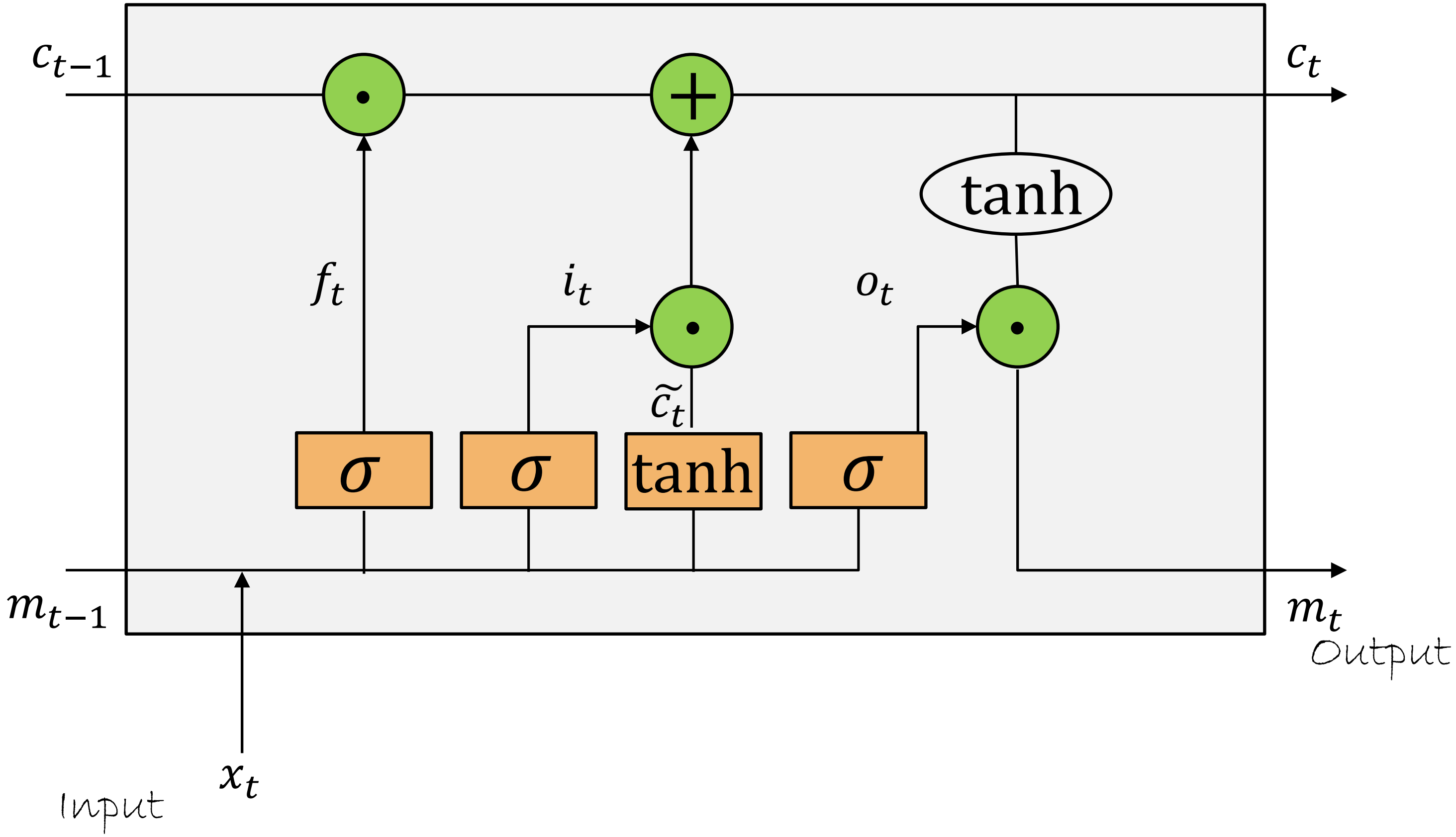
$$f = \sigma\left(x_t U^{(f)} + m_{t-1} W^{(f)}\right)$$

$$o = \sigma\left(x_t U^{(o)} + m_{t-1} W^{(o)}\right)$$

$$\widetilde{c_t} = \tanh(x_t U^{(g)} + m_{t-1} W^{(g)})$$

$$c_t = c_{t-1} \odot f + \widetilde{c_t} \odot i$$

$$m_t = \tanh(c_t) \odot o$$



LSTM Step-by-Step: Step (1)

- E.g. Model the sentence “Yesterday she slapped me. Today she loves me.”
- Decide what to forget and what to remember for the new memory
  - Sigmoid 1 → Remember everything
  - Sigmoid 0 → Forget everything

$$i_t = \sigma\left(x_t U^{(i)} + m_{t-1} W^{(i)}\right)$$

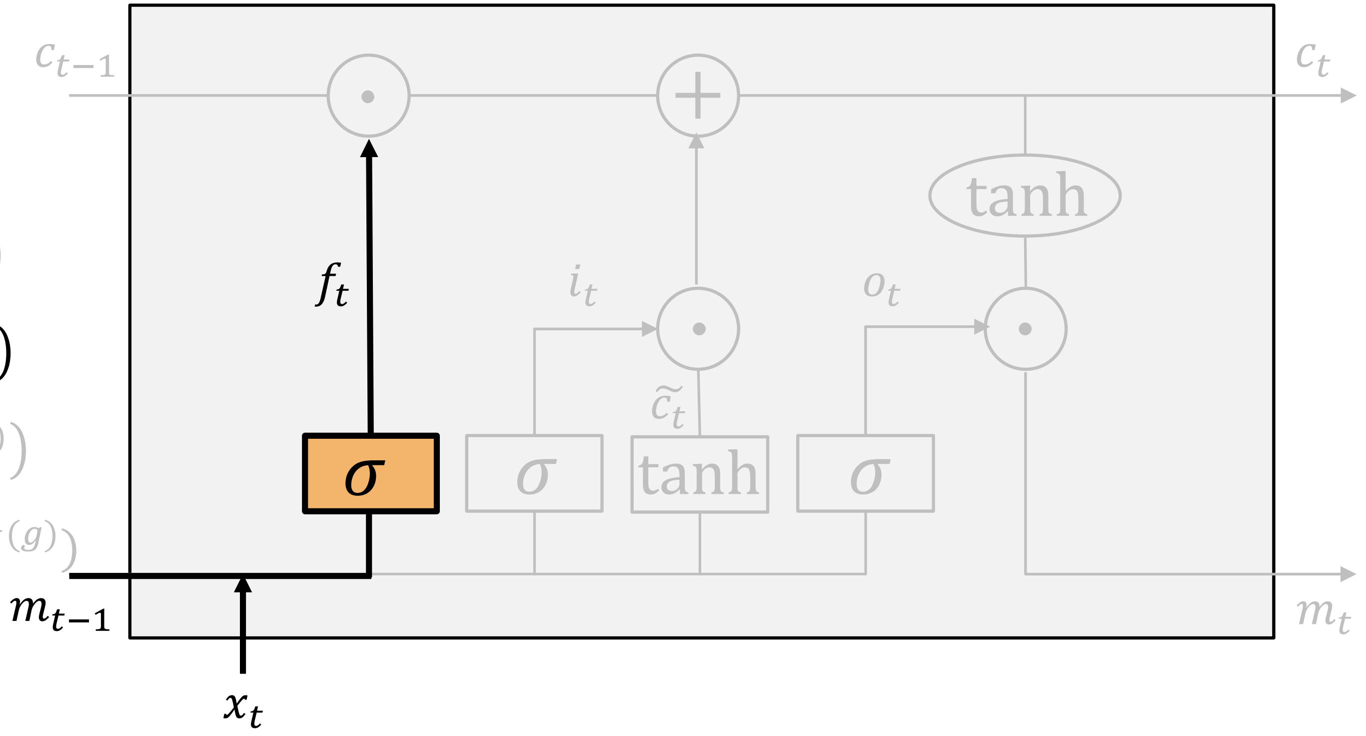
$$f_t = \sigma\left(x_t U^{(f)} + m_{t-1} W^{(f)}\right)$$

$$o_t = \sigma\left(x_t U^{(o)} + m_{t-1} W^{(o)}\right)$$

$$\tilde{c}_t = \tanh(x_t U^{(g)} + m_{t-1} W^{(g)})$$

$$c_t = c_{t-1} \odot f + \tilde{c}_t \odot i$$

$$m_t = \tanh(c_t) \odot o$$



LSTM Step-by-Step: Step (2)

- Decide what new information should you add to the new memory
  - Modulate the input i_t
  - Generate candidate memories $\widetilde{c_t}$

$$i_t = \sigma\left(x_t U^{(i)} + m_{t-1} W^{(i)}\right)$$

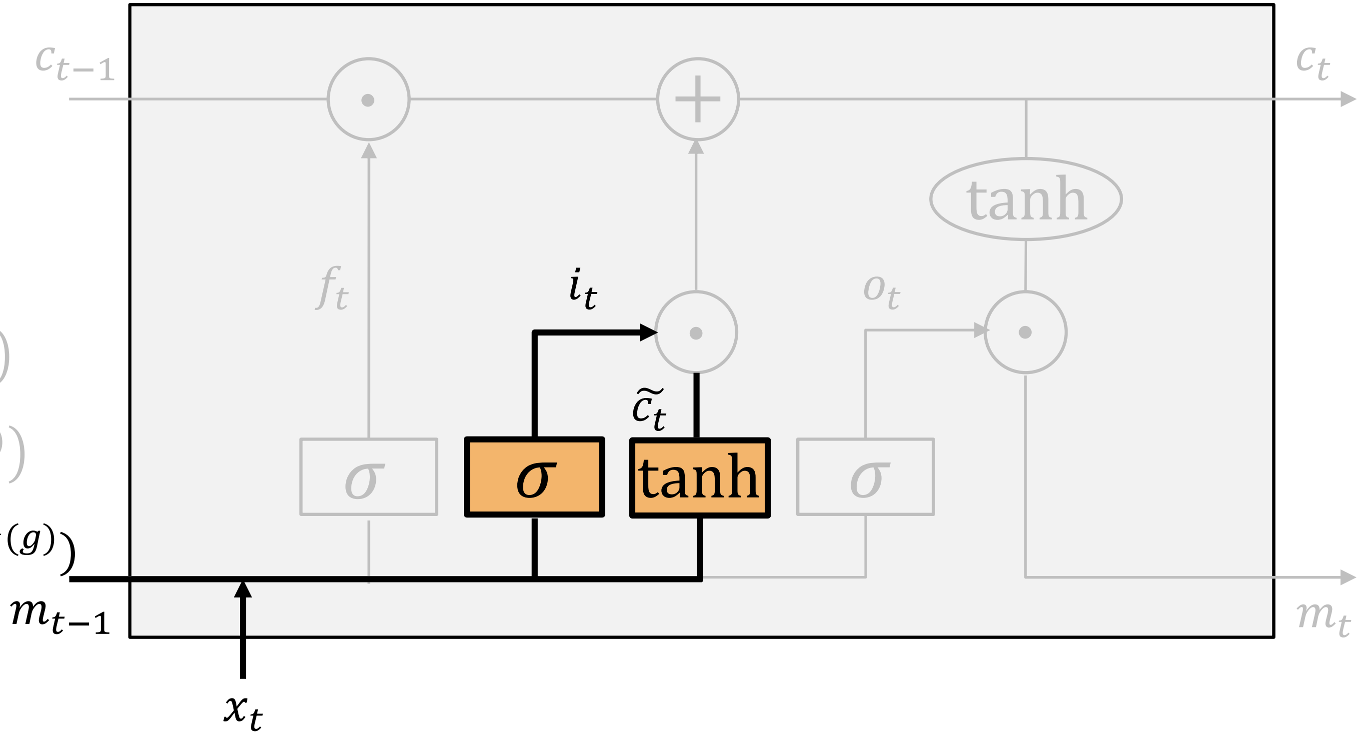
$$f_t = \sigma\left(x_t U^{(f)} + m_{t-1} W^{(f)}\right)$$

$$o_t = \sigma\left(x_t U^{(o)} + m_{t-1} W^{(o)}\right)$$

$$\widetilde{c_t} = \tanh(x_t U^{(g)} + m_{t-1} W^{(g)})$$

$$c_t = c_{t-1} \odot f + \widetilde{c_t} \odot i$$

$$m_t = \tanh(c_t) \odot o$$



LSTM Step-by-Step: Step (3)

- Compute and update the current cell state c_t
  - Depends on the previous cell state
  - What we decided to forget
  - What inputs we allowed
  - The candidate memories

$$i_t = \sigma\left(x_t U^{(i)} + m_{t-1} W^{(i)}\right)$$

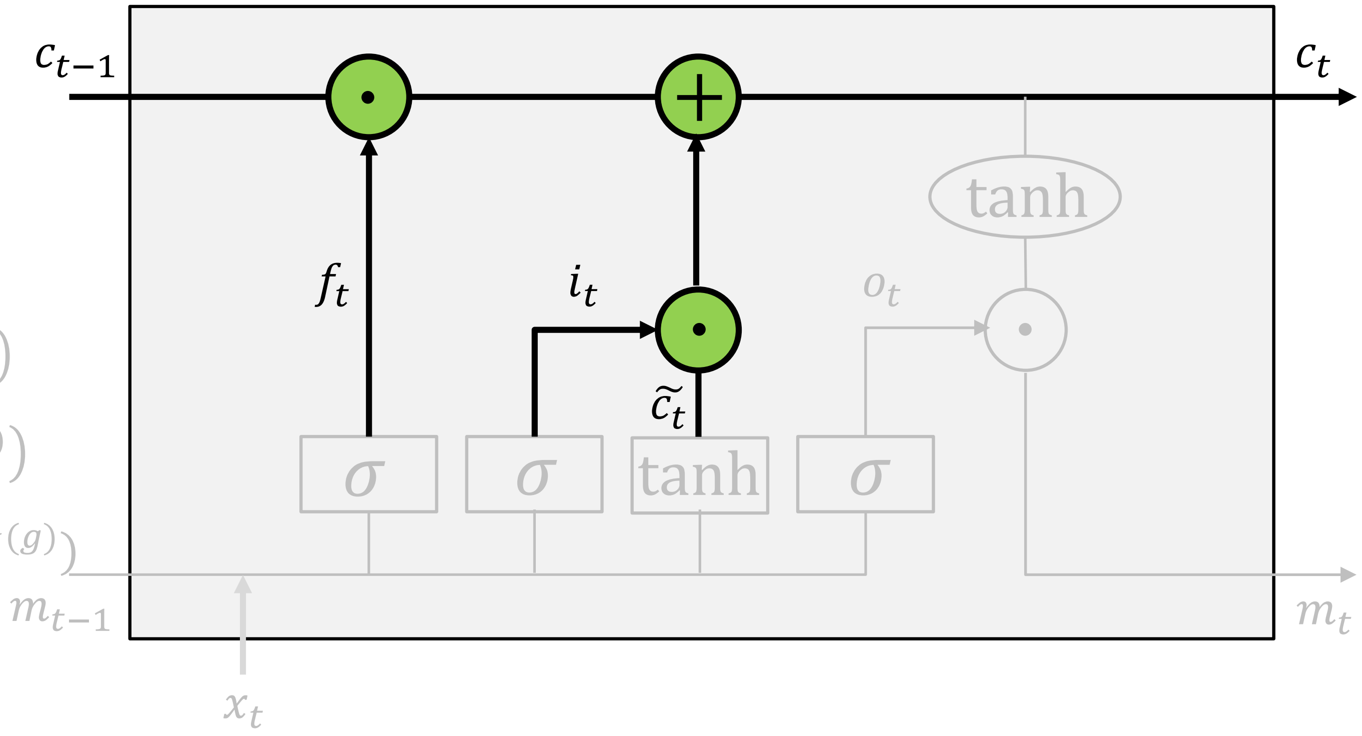
$$f_t = \sigma\left(x_t U^{(f)} + m_{t-1} W^{(f)}\right)$$

$$o_t = \sigma\left(x_t U^{(o)} + m_{t-1} W^{(o)}\right)$$

$$\widetilde{c_t} = \tanh(x_t U^{(g)} + m_{t-1} W^{(g)})$$

$$c_t = c_{t-1} \odot f + \widetilde{c_t} \odot i$$

$$m_t = \tanh(c_t) \odot o$$



LSTM Step-by-Step: Step (4)

- Modulate the output
  - Does the cell state contain something relevant? → Sigmoid 1
- Generate the new memory

$$i_t = \sigma\left(x_t U^{(i)} + m_{t-1} W^{(i)}\right)$$

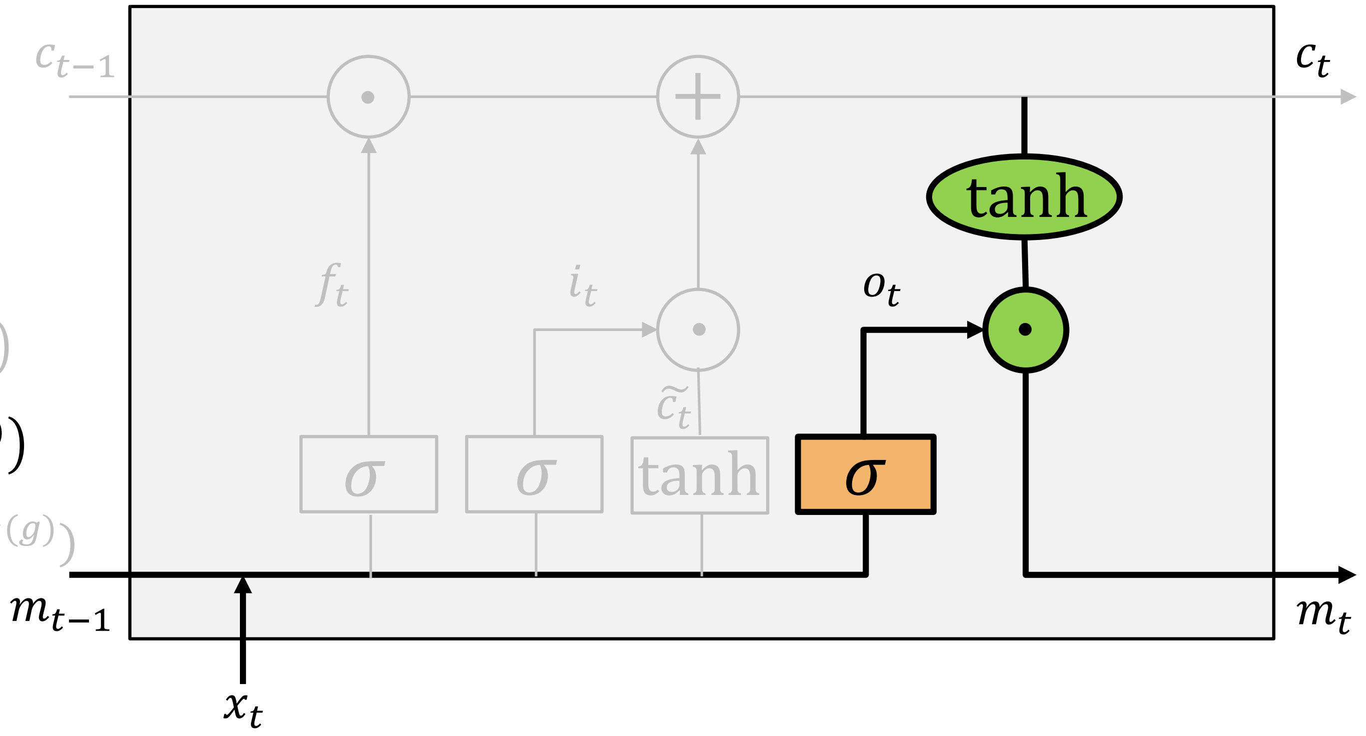
$$f_t = \sigma\left(x_t U^{(f)} + m_{t-1} W^{(f)}\right)$$

$$o_t = \sigma\left(x_t U^{(o)} + m_{t-1} W^{(o)}\right)$$

$$\widetilde{c_t} = \tanh(x_t U^{(g)} + m_{t-1} W^{(g)})$$

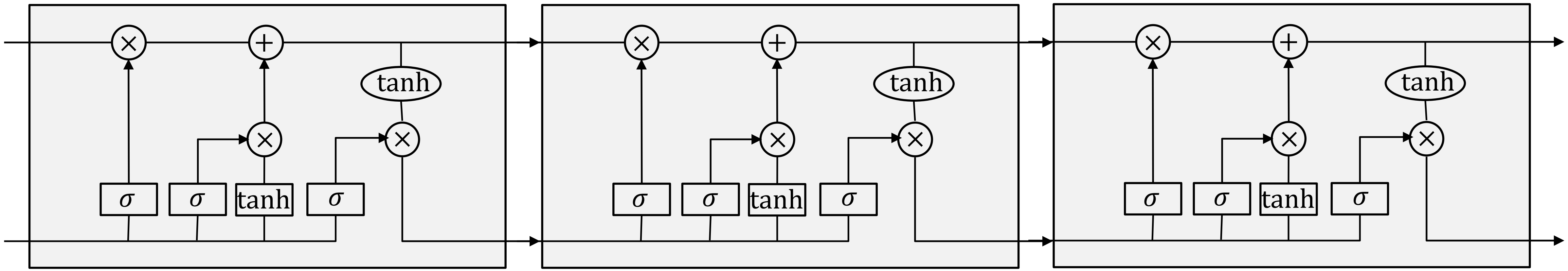
$$c_t = c_{t-1} \odot f + \widetilde{c_t} \odot i$$

$$m_t = \tanh(c_t) \odot o$$



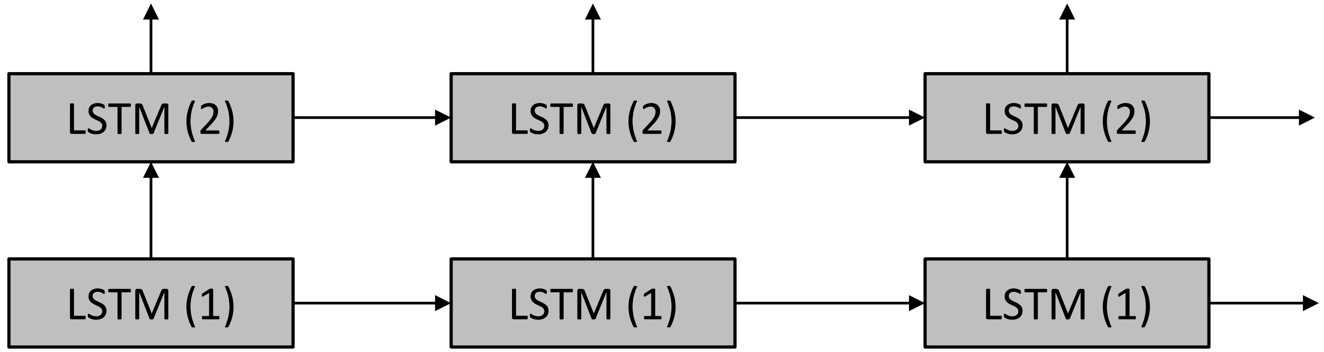
LSTM Unrolled Network

- Macroscopically very similar to standard RNNs
- The engine is a bit different (more complicated)
  - Because of their gates LSTMs capture long and short term dependencies



Even more beef to the RNNs

- LSTM with peephole connections
  - Gates have access also to the previous cell states c_{t-1} (not only memories)
  - Coupled forget and input gates, $c_t = f_t \odot c_{t-1} + (1 - f_t) \odot \widetilde{c}_t$
  - Bi-directional recurrent networks
- GRU
  - A bit simpler than LSTM
  - Performance similar to LSTM
- Deep LSTM

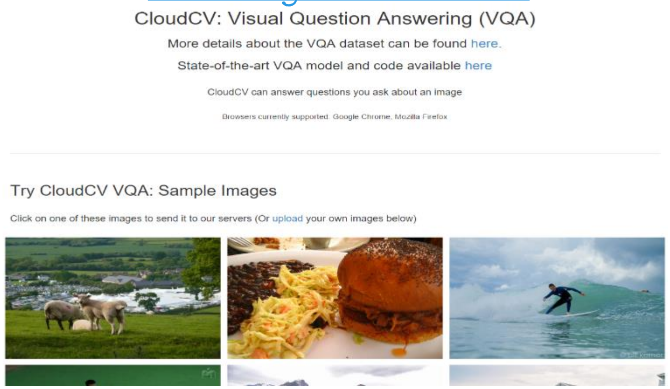


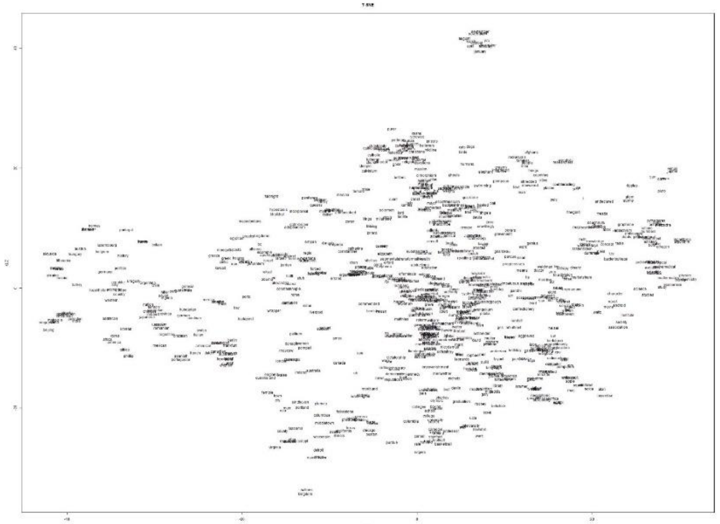
Applications of Recurrent Networks

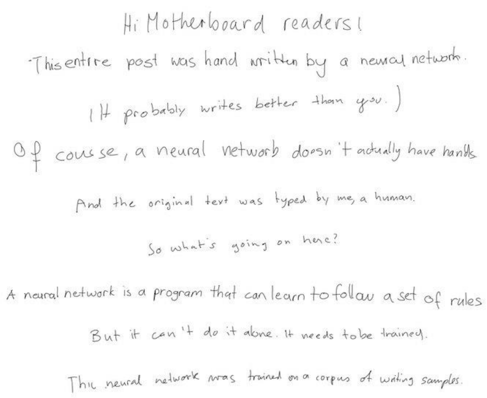
Click to go to the video in Youtube

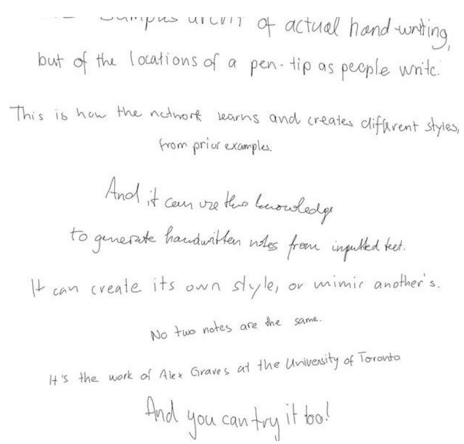
NeuralTalk and Walk, recognition, text description of the image while walking

Click to go to the website





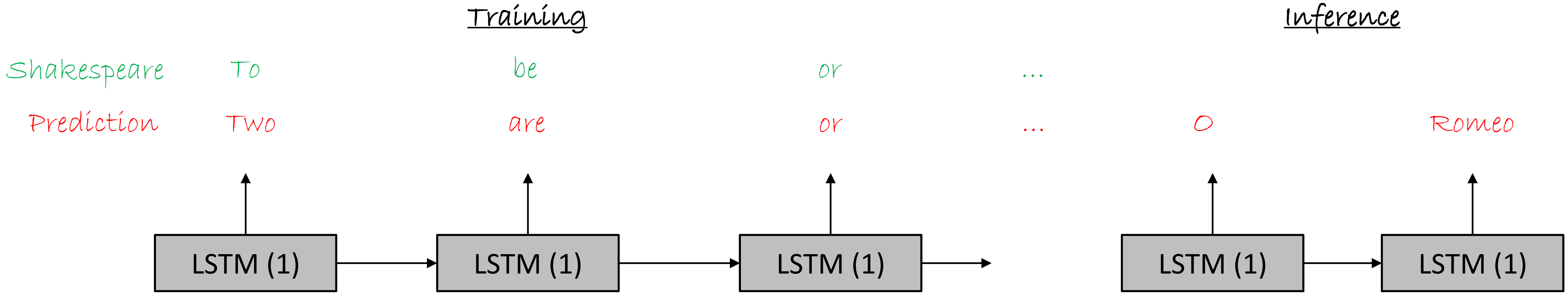




Text generation

- Generate text like Nietzsche, Shakespeare or Wikipedia
- Generate Linux kernel-like C++ code
- Or even generate a new website

$$\Pr(x) = \prod_i \Pr(x_i \mid x_{past}; \theta)$$, where θ are the LSTM parameters



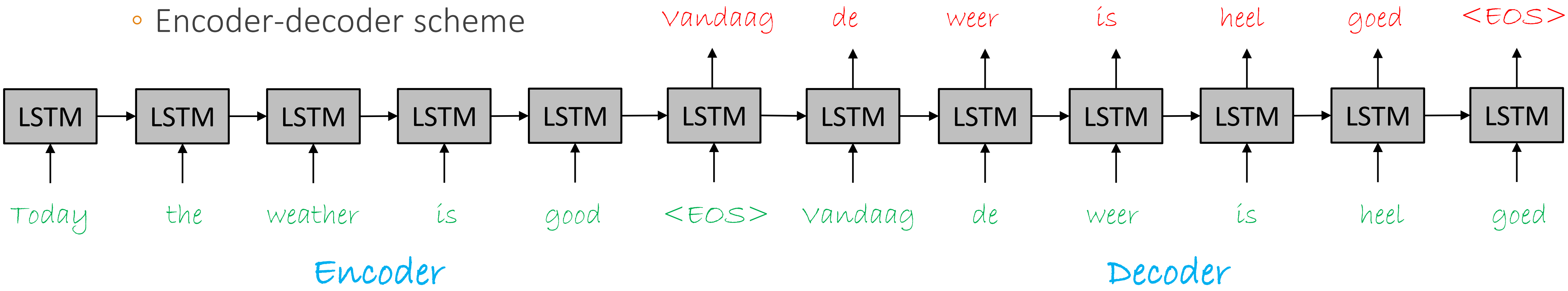
Handwriting generation

- Problem 1: Model real coordinates instead of one-hot vectors
- Recurrent Mixture Density Networks
- Have outputs follow a Gaussian Distribution
  - Output needs to be suitably squashed
- We don't just fit a Gaussian to the data
  - We also condition on the previous outputs

$$\Pr(o_t) = \sum_i w_i(x_{1:T}) N(o_t | \mu_i(x_{1:T}), \Sigma_i(x_{1:T}))$$

Machine Translation

- The phrase in the source language is one sequence
  - “Today the weather is very good”
- The phrase in the target language is also a sequence
  - “Vandaag de weer is heel goed”
- Problems
  - no perfect word alignment, sentence length might differ
- Solution
  - Encoder-decoder scheme



Better Machine Translation

- It might even pay off reversing the source sentence
  - The first target words will be closer to their respective source words
- The encoder and decoder parts can be modelled with different LSTMs
- Deep LSTM

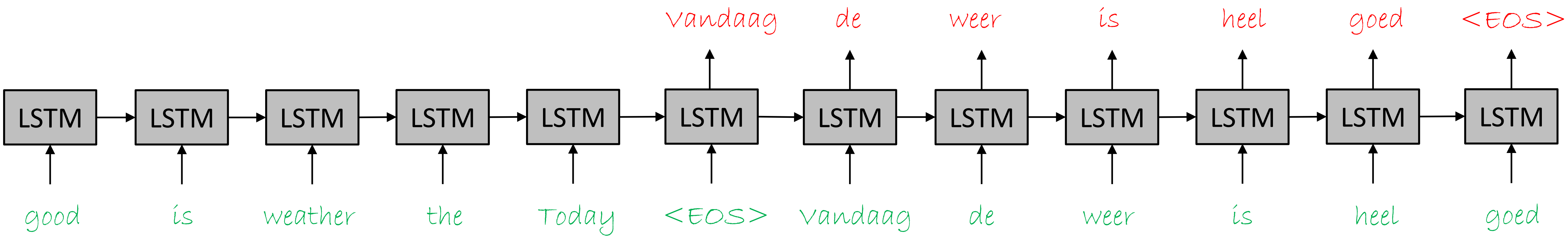
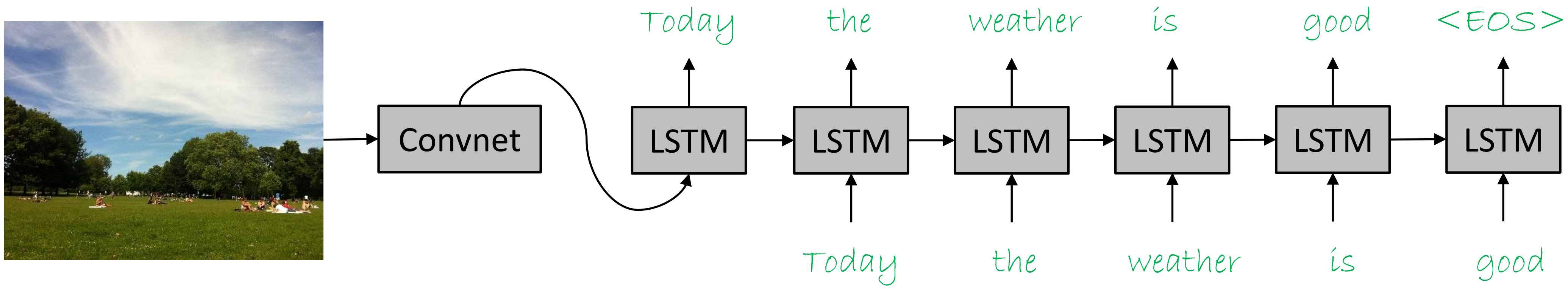


Image captioning

- An image is a thousand words, literally!
- Pretty much the same as machine transation
- Replace the encoder part with the output of a Convnet
  - E.g. use Alexnet or a VGG16 network
- Keep the decoder part to operate as a translator



Question answering

- Bleeding-edge research, no real consensus
  - Very interesting open, research problems
- Again, pretty much like machine translation
- Encoder-Decoder scheme
  - Insert the question to the encoder part
  - Model the answer at the decoder part
- You can also have question answering with images
  - Again, bleeding-edge research
  - How/where to add the image?
  - What has been working so far is to add the image only in the beginning

Q: John entered the living room, where he met Mary. She was drinking some wine and watching a movie. What room did John enter?

A: John entered the living room.

Q: what are the people playing?

A: They play beach football

Summary

- Recurrent Neural Networks (RNN) for sequences
- Backpropagation Through Time
- RNNs using Long Short-Term Memory (LSTM)
- Applications of Recurrent Neural Networks

Next lecture

- Memory networks
- Recursive networks