

Recurrent Neural Networks

Prof. Adriana Kovashka
(with some slides from me)

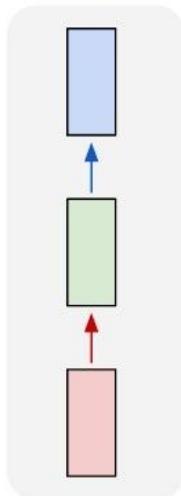
Plan for this lecture

- Recurrent neural networks
 - Basics
 - Training (backprop through time, vanishing gradient)
 - Recurrent networks with gates (GRU, LSTM)
- Applications in NLP and vision
 - Neural machine translation (beam search, attention)
 - Image/video captioning

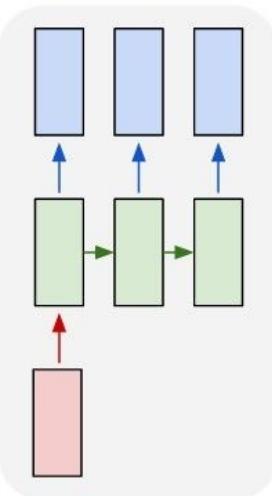
Recurrent neural networks

Recurrent Networks offer a lot of flexibility:

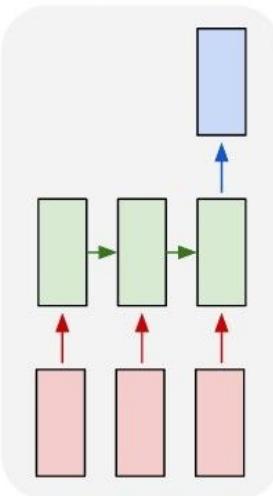
one to one



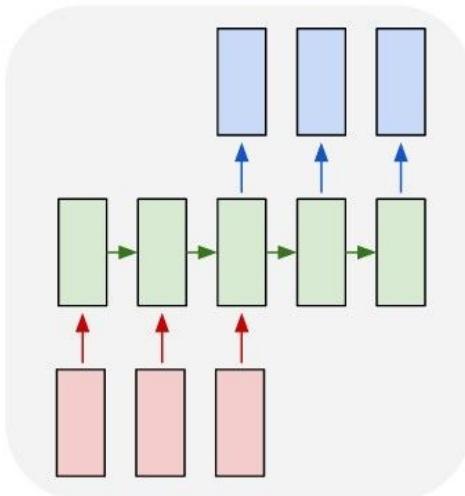
one to many



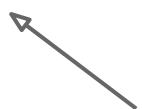
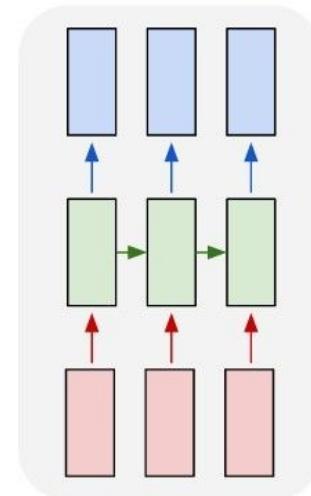
many to one



many to many



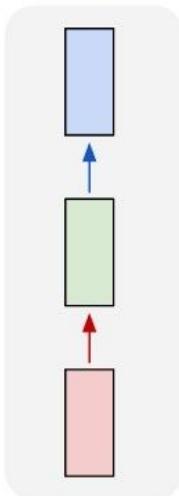
many to many



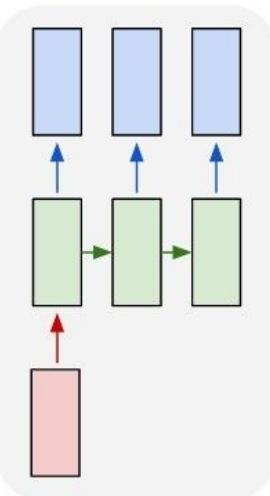
vanilla neural networks

Recurrent Networks offer a lot of flexibility:

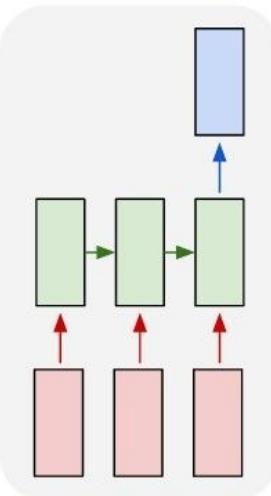
one to one



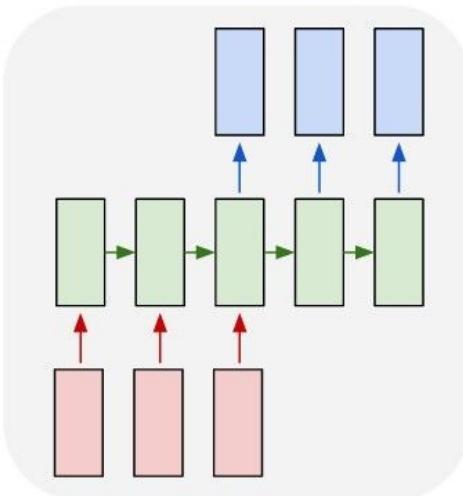
one to many



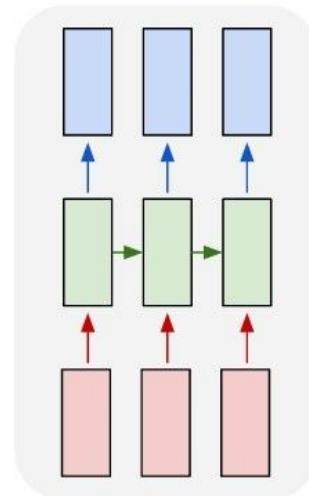
many to one



many to many



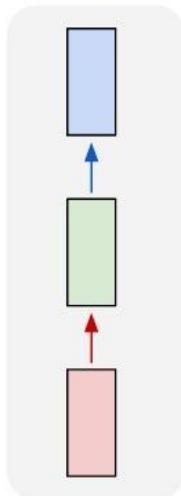
many to many



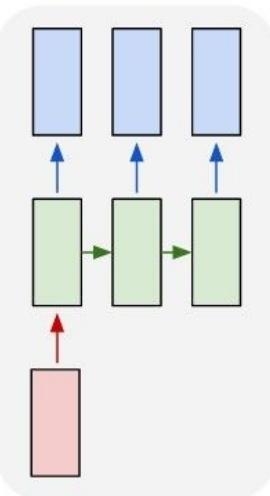
e.g. **image captioning**
image -> sequence of words

Recurrent Networks offer a lot of flexibility:

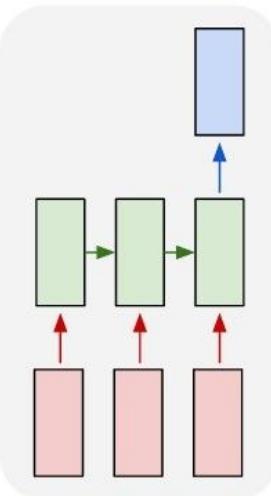
one to one



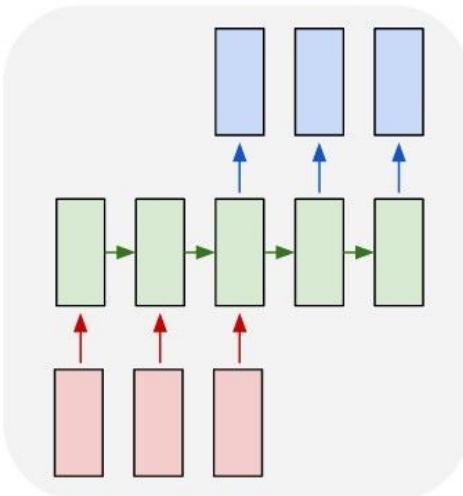
one to many



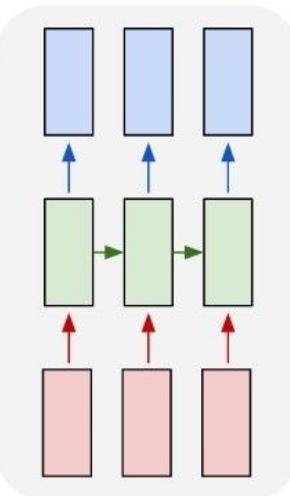
many to one



many to many



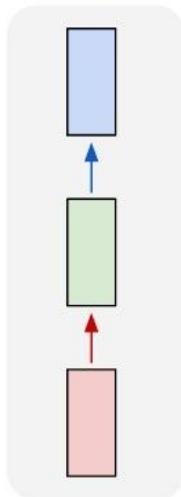
many to many



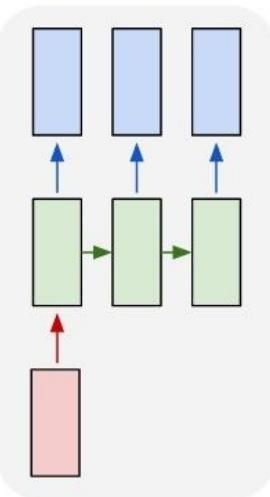
e.g. **sentiment classification**
sequence of words -> sentiment

Recurrent Networks offer a lot of flexibility:

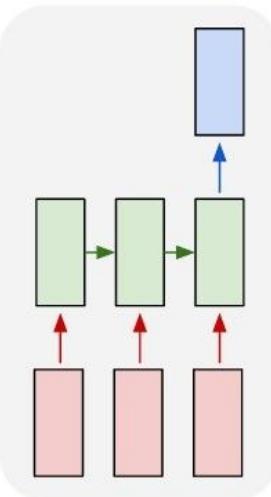
one to one



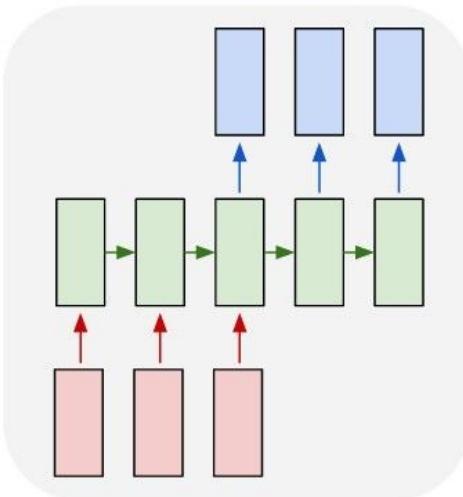
one to many



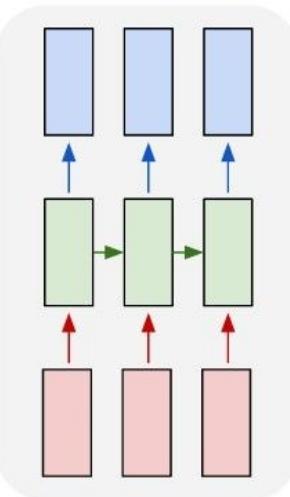
many to one



many to many



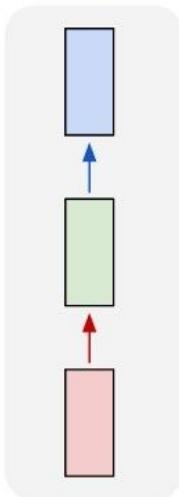
many to many



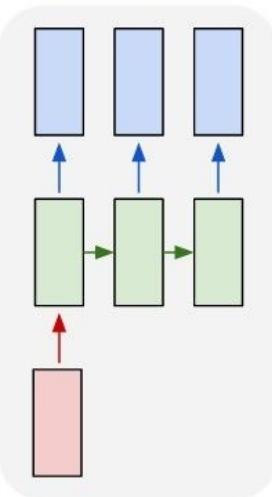
e.g. **machine translation**
seq of words -> seq of words

Recurrent Networks offer a lot of flexibility:

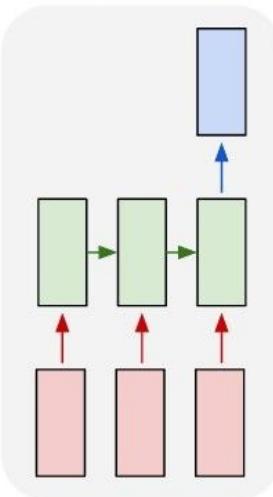
one to one



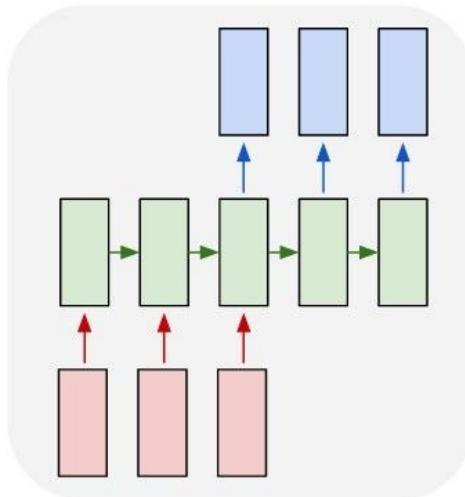
one to many



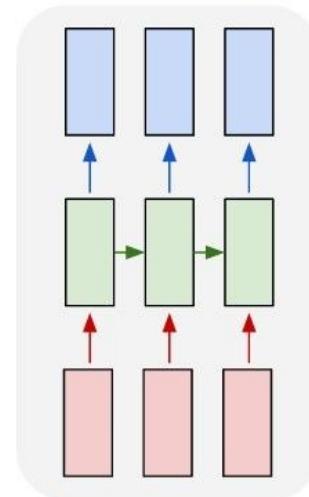
many to one



many to many



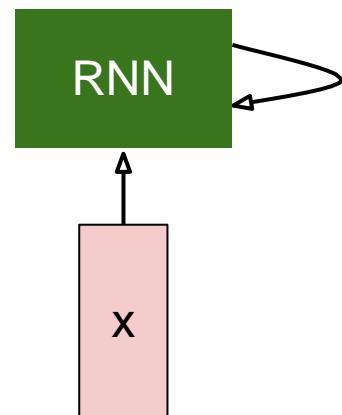
many to many



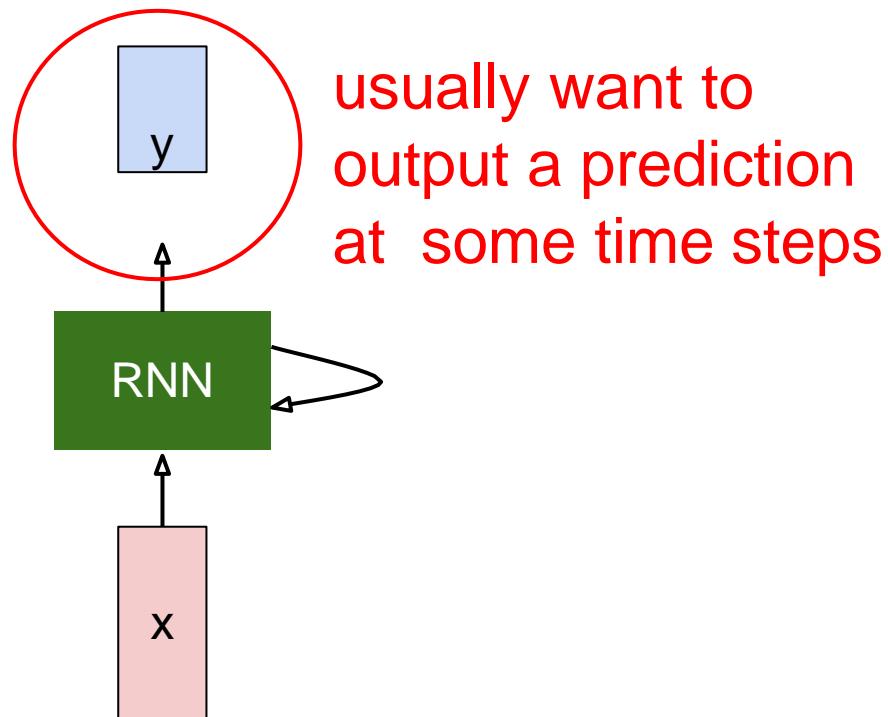
e.g. **video classification on frame level**



Recurrent Neural Network



Recurrent Neural Network

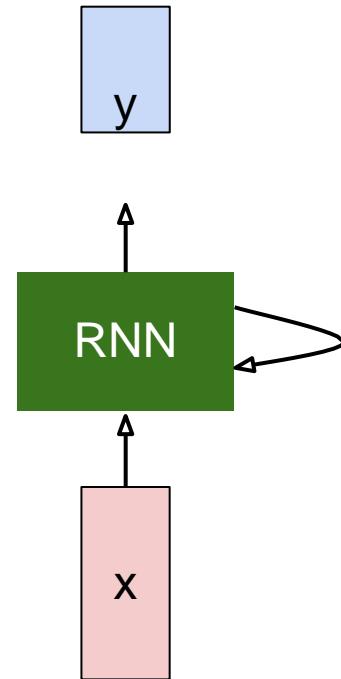


Recurrent Neural Network

We can process a sequence of vectors \mathbf{x} by applying a recurrence formula at every time step:

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

new state / old state input vector at
 \ some function some time step
 some function
 with parameters W

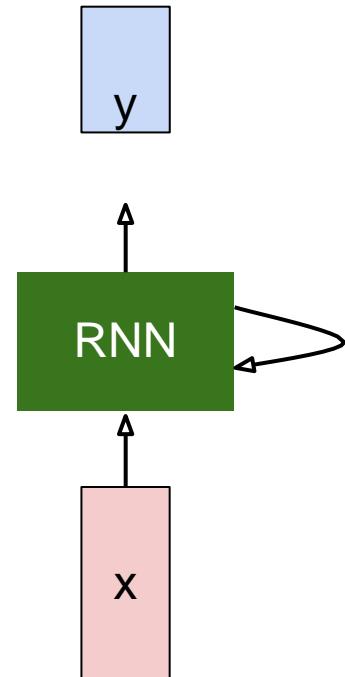


Recurrent Neural Network

We can process a sequence of vectors \mathbf{x} by applying a recurrence formula at every time step:

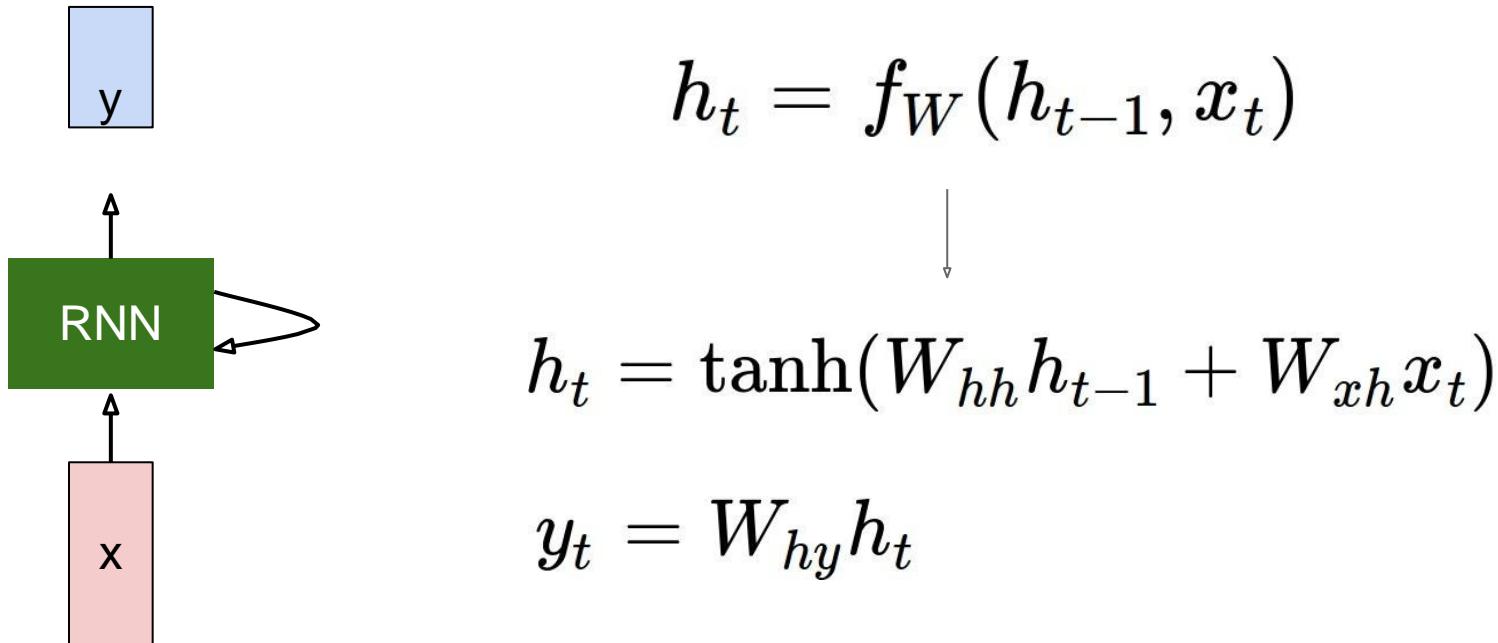
$$h_t = f_W(h_{t-1}, x_t)$$

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



(Vanilla) Recurrent Neural Network

The state consists of a single “hidden” vector \mathbf{h} :

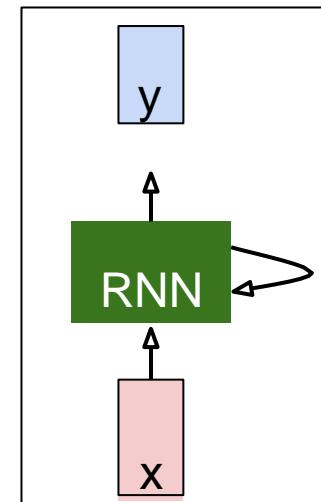


Example

**Character-level
language model
example**

Vocabulary:
[h,e,l,o]

Example training
sequence:
“hello”

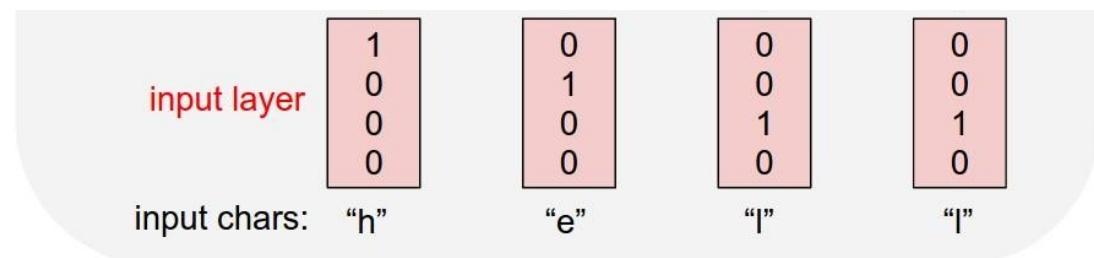


Example

Character-level language model example

Vocabulary:
[h,e,l,o]

Example training
sequence:
“hello”



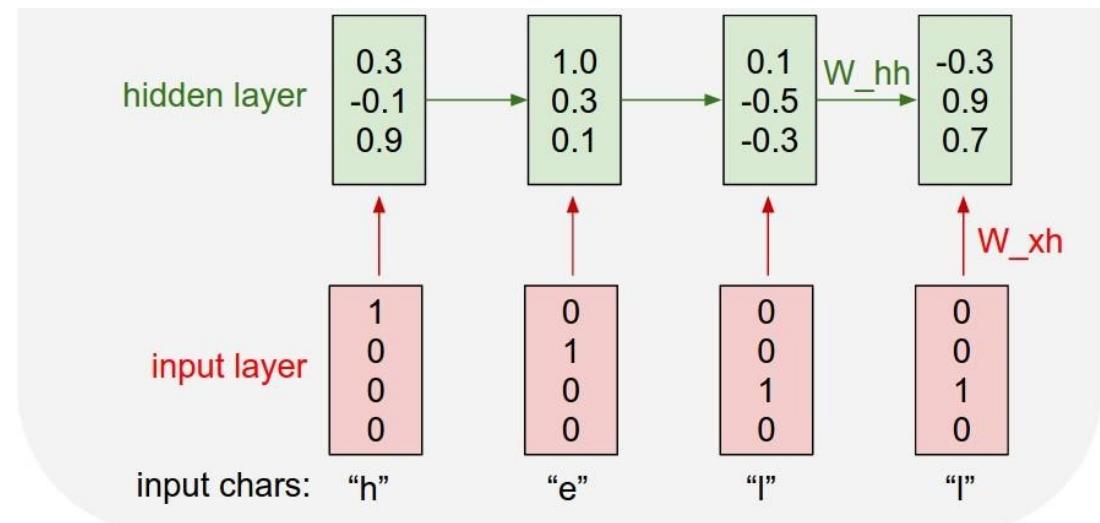
Example

Character-level language model example

Vocabulary:
[h,e,l,o]

Example training
sequence:
“hello”

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

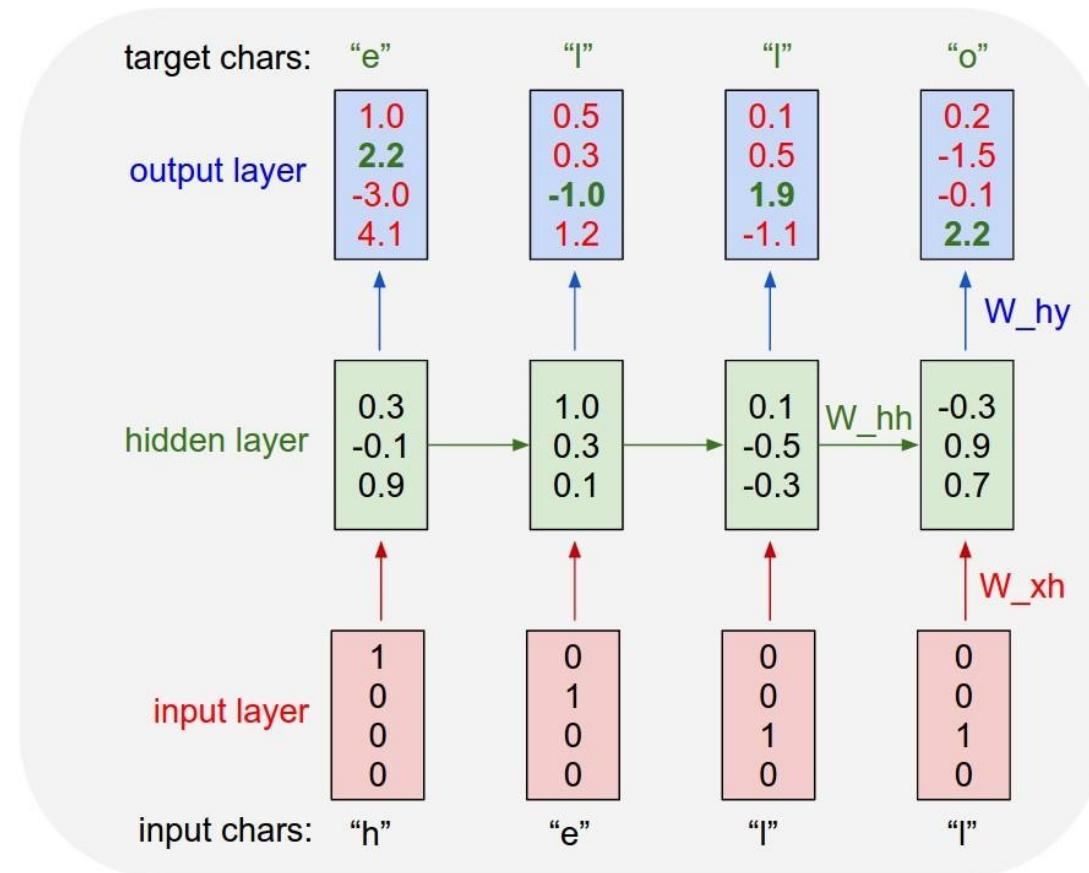


Example

Character-level language model example

Vocabulary:
[h,e,l,o]

Example training sequence:
“hello”



What do we still need to specify, for this to work?

What kind of loss can we formulate?

Training a Recurrent Neural Network

- Get a **big corpus of text** which is a sequence of words $\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(T)}$
- Feed into RNN; compute output distribution $\hat{\mathbf{y}}^{(t)}$ **for every step t .**
 - i.e. predict probability distribution of *every word*, given words so far
- **Loss function** on step t is **cross-entropy** between predicted probability distribution $\hat{\mathbf{y}}^{(t)}$, and true next word $\mathbf{y}^{(t)}$ (one-hot); V is vocabulary

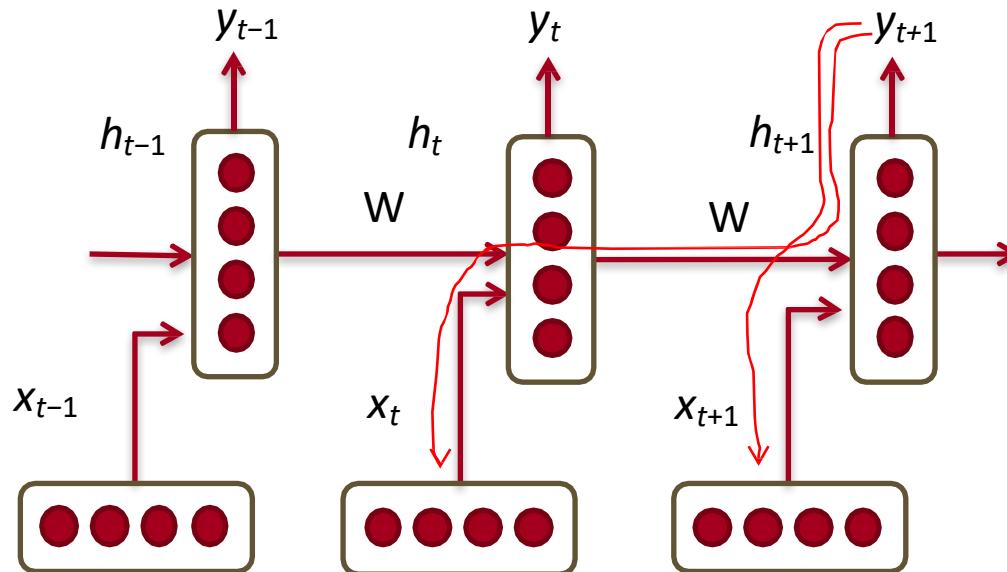
$$J^{(t)}(\theta) = CE(\mathbf{y}^{(t)}, \hat{\mathbf{y}}^{(t)}) = - \sum_{w \in V} \mathbf{y}_w^{(t)} \log \hat{\mathbf{y}}_w^{(t)} = - \log \hat{\mathbf{y}}_{\mathbf{x}_{t+1}}^{(t)}$$

- Average this to get **overall loss** for entire training set:

$$J(\theta) = \frac{1}{T} \sum_{t=1}^T J^{(t)}(\theta) = \frac{1}{T} \sum_{t=1}^T - \log \hat{\mathbf{y}}_{\mathbf{x}_{t+1}}^{(t)}$$

The vanishing/exploding gradient problem

- The error at a time step ideally can tell a previous time step from many steps away to change during backprop
- Multiply the same matrix at each time step during backprop



The vanishing gradient problem

- Total error is the sum of each error at time steps t

$$\frac{\partial E}{\partial W} = \sum_{t=1}^T \frac{\partial E_t}{\partial W}$$

- Chain rule:

$$\frac{\partial E_t}{\partial W} = \sum_{k=1}^t \frac{\partial E_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial W}$$

- More chain rule:

$$\frac{\partial h_t}{\partial h_k} = \prod_{j=k+1}^t \frac{\partial h_j}{\partial h_{j-1}}$$

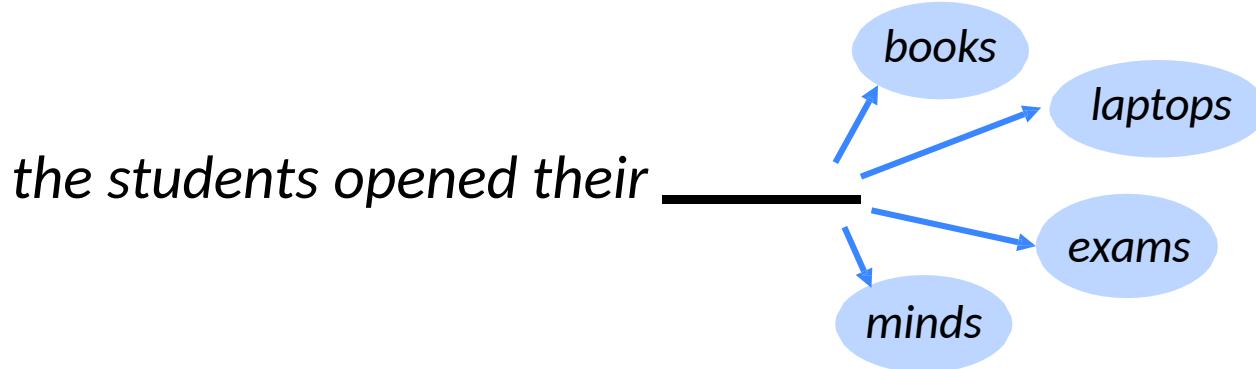
- Derivative of vector wrt vector is a Jacobian matrix of partial derivatives; norm of this matrix can become very small or very large quickly [Bengio et al 1994, Pascanu et al. 2013], leading to vanishing/exploding gradient

What uses of language models from everyday life can you think of?

Now in more detail...

Language Modeling

- **Language Modeling** is the task of predicting what word comes next.



- More formally: given a sequence of words $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(t)}$, compute the probability distribution of the next word $\mathbf{x}^{(t+1)}$:

$$P(\mathbf{x}^{(t+1)} | \mathbf{x}^{(t)}, \dots, \mathbf{x}^{(1)})$$

where $\mathbf{x}^{(t+1)}$ can be any word in the vocabulary $V = \{\mathbf{w}_1, \dots, \mathbf{w}_{|V|}\}$

- A system that does this is called a **Language Model**.

n-gram Language Models

- First we make a **simplifying assumption**: $\mathbf{x}^{(t+1)}$ depends only on the preceding $n-1$ words.

$$P(\mathbf{x}^{(t+1)} | \mathbf{x}^{(t)}, \dots, \mathbf{x}^{(1)}) = P(\mathbf{x}^{(t+1)} | \underbrace{\mathbf{x}^{(t)}, \dots, \mathbf{x}^{(t-n+2)}}_{n-1 \text{ words}}) \quad (\text{assumption})$$

prob of a n-gram $\rightarrow P(\mathbf{x}^{(t+1)}, \mathbf{x}^{(t)}, \dots, \mathbf{x}^{(t-n+2)})$ (definition of conditional prob)

prob of a (n-1)-gram $\rightarrow P(\mathbf{x}^{(t)}, \dots, \mathbf{x}^{(t-n+2)})$

- Question:** How do we get these n -gram and $(n-1)$ -gram probabilities?
- Answer:** By **counting** them in some large corpus of text!

$$\approx \frac{\text{count}(\mathbf{x}^{(t+1)}, \mathbf{x}^{(t)}, \dots, \mathbf{x}^{(t-n+2)})}{\text{count}(\mathbf{x}^{(t)}, \dots, \mathbf{x}^{(t-n+2)})} \quad (\text{statistical approximation})$$

Sparsity Problems with n-gram Language Models

Sparsity Problem 1

Problem: What if “students opened their w ” never occurred in data? Then w has probability 0!

(Partial) Solution: Add small δ to the count for every $w \in V$. This is called *smoothing*.

$$P(w|\text{students opened their}) = \frac{\text{count}(\text{students opened their } w)}{\text{count}(\text{students opened their})}$$

Sparsity Problem 2

Problem: What if “students opened their” never occurred in data? Then we can’t calculate probability for *any* w !

(Partial) Solution: Just condition on “opened their” instead. This is called *backoff*.

Note: Increasing n makes sparsity problems worse.
Typically we can’t have n bigger than 5.

A fixed-window neural Language Model

output distribution

$$\hat{y} = \text{softmax}(\mathbf{U}\mathbf{h} + \mathbf{b}_2) \in \mathbb{R}^{|V|}$$

hidden layer

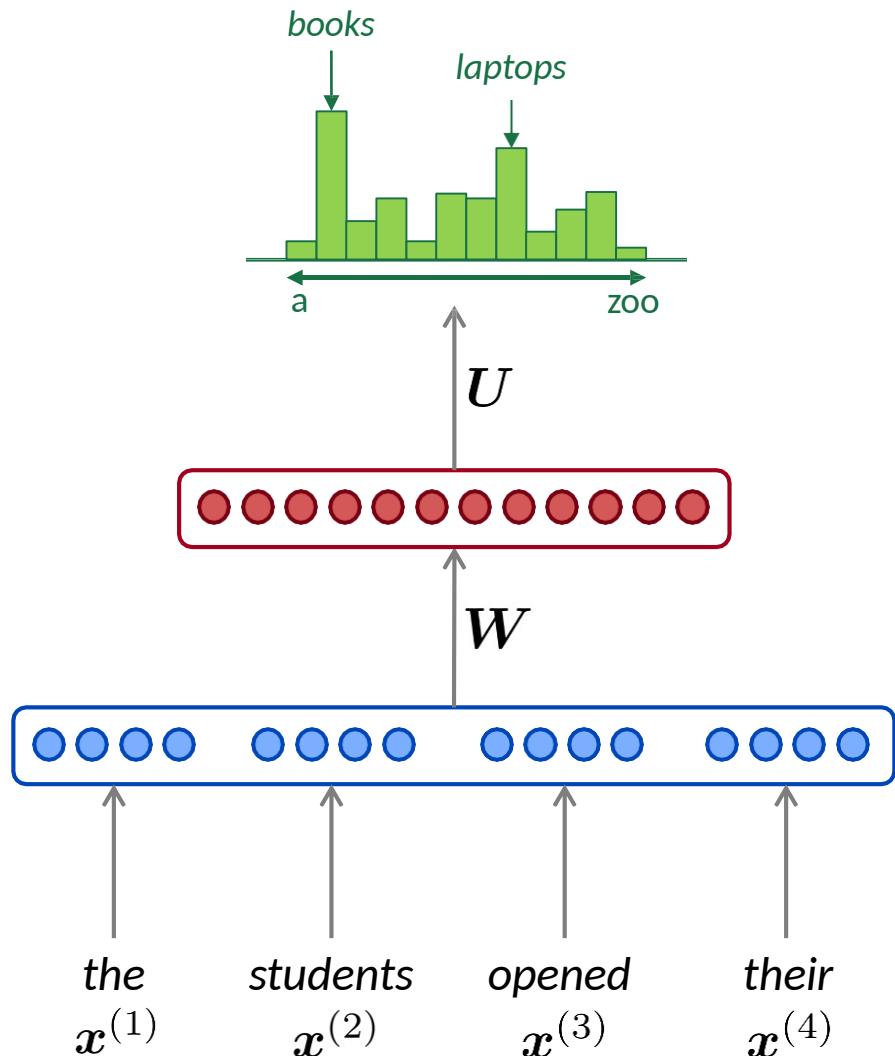
$$\mathbf{h} = f(\mathbf{W}\mathbf{e} + \mathbf{b}_1)$$

concatenated word embeddings

$$\mathbf{e} = [\mathbf{e}^{(1)}; \mathbf{e}^{(2)}; \mathbf{e}^{(3)}; \mathbf{e}^{(4)}]$$

words / one-hot vectors

$$\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \mathbf{x}^{(3)}, \mathbf{x}^{(4)}$$



A fixed-window neural Language Model

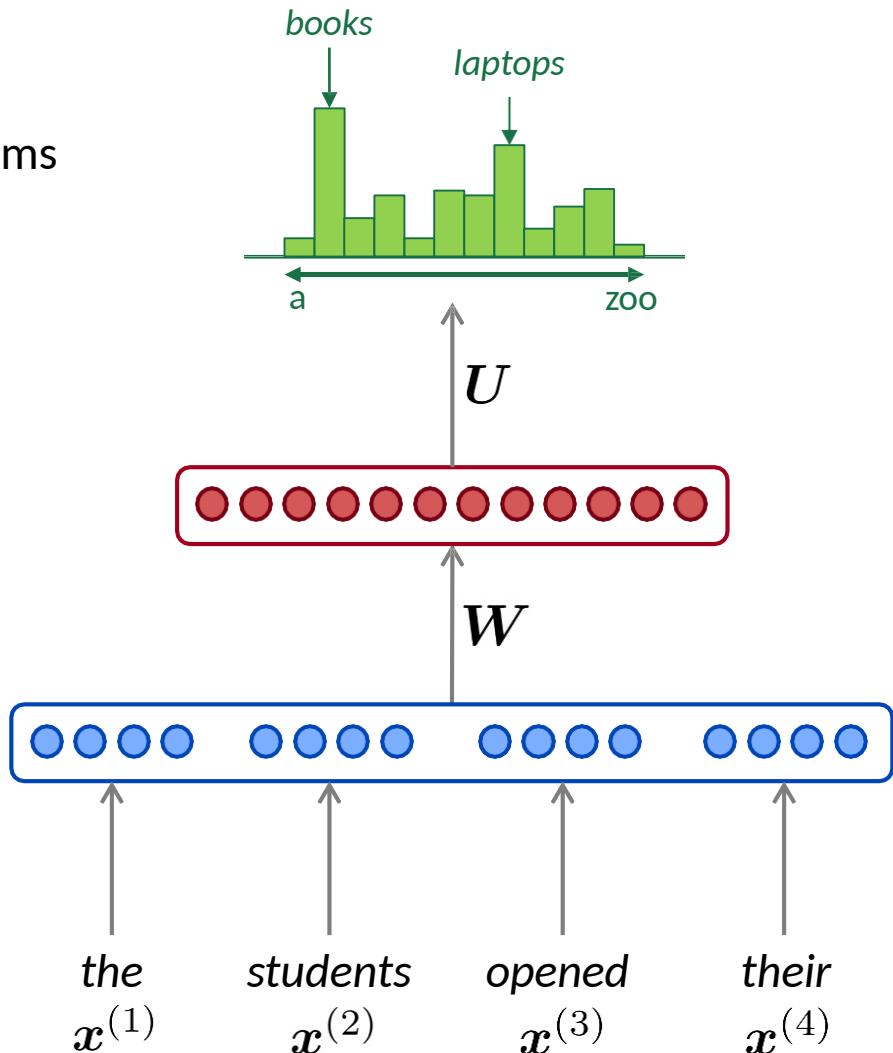
Improvements over n -gram LM:

- No sparsity problem
- Don't need to store all observed n -grams

Remaining **problems**:

- Fixed window is **too small**
- Enlarging window enlarges W
- Window can never be large enough!
- $x^{(1)}$ and $x^{(2)}$ are multiplied by completely different weights in W .
No symmetry in how the inputs are processed.

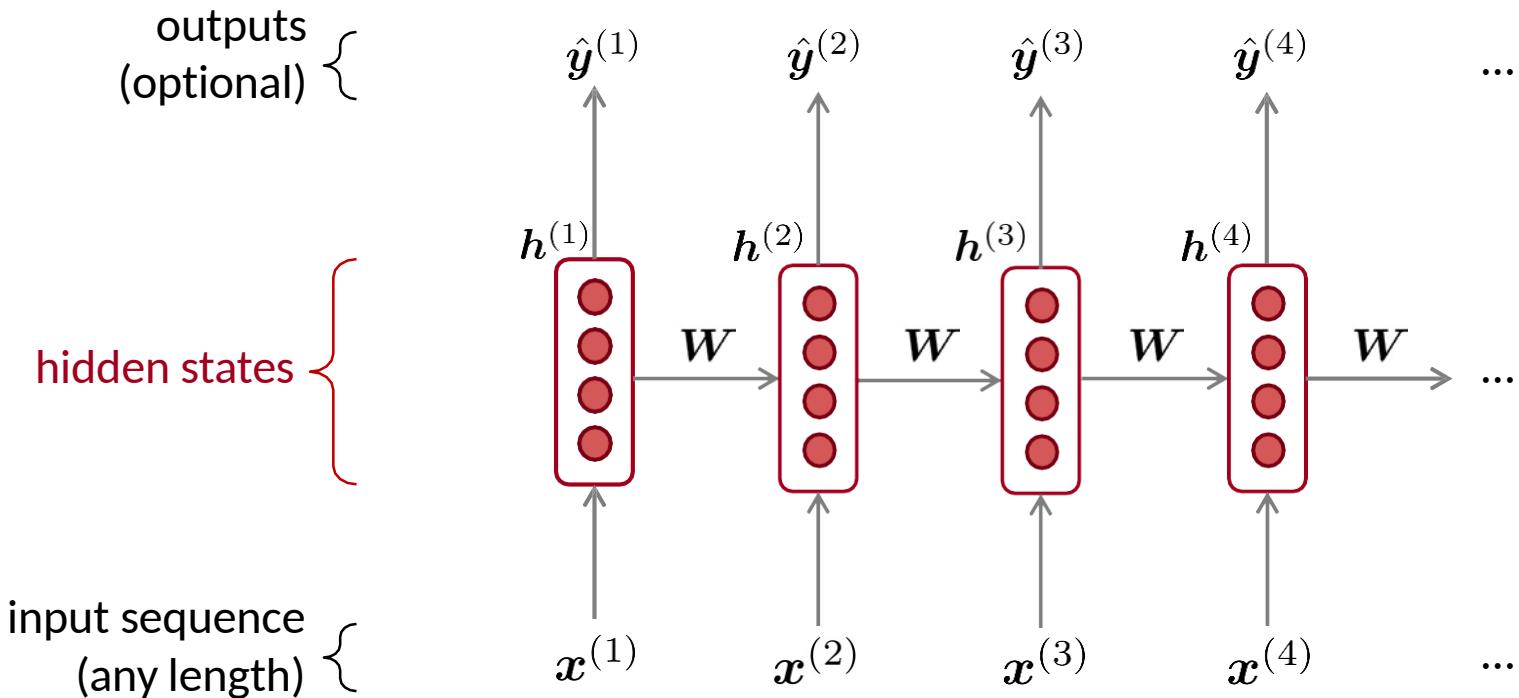
We need a neural architecture that can process *any length input*



Recurrent Neural Networks (RNN)

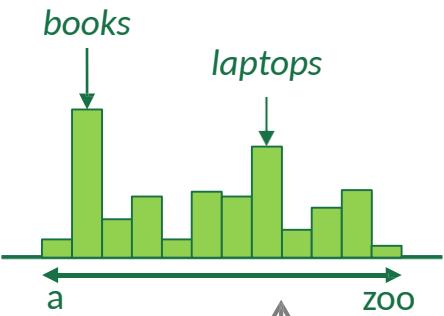
A family of neural architectures

Core idea: Apply the same weights W repeatedly



A RNN Language Model

$\hat{y}^{(4)} = P(\mathbf{x}^{(5)} | \text{the students opened their}$



output distribution

$$\hat{y}^{(t)} = \text{softmax}(\mathbf{U}\mathbf{h}^{(t)} + \mathbf{b}_2) \in \mathbb{R}^{|V|}$$

hidden states

$$\mathbf{h}^{(t)} = \sigma(\mathbf{W}_h \mathbf{h}^{(t-1)} + \mathbf{W}_e \mathbf{e}^{(t)} + \mathbf{b}_1)$$

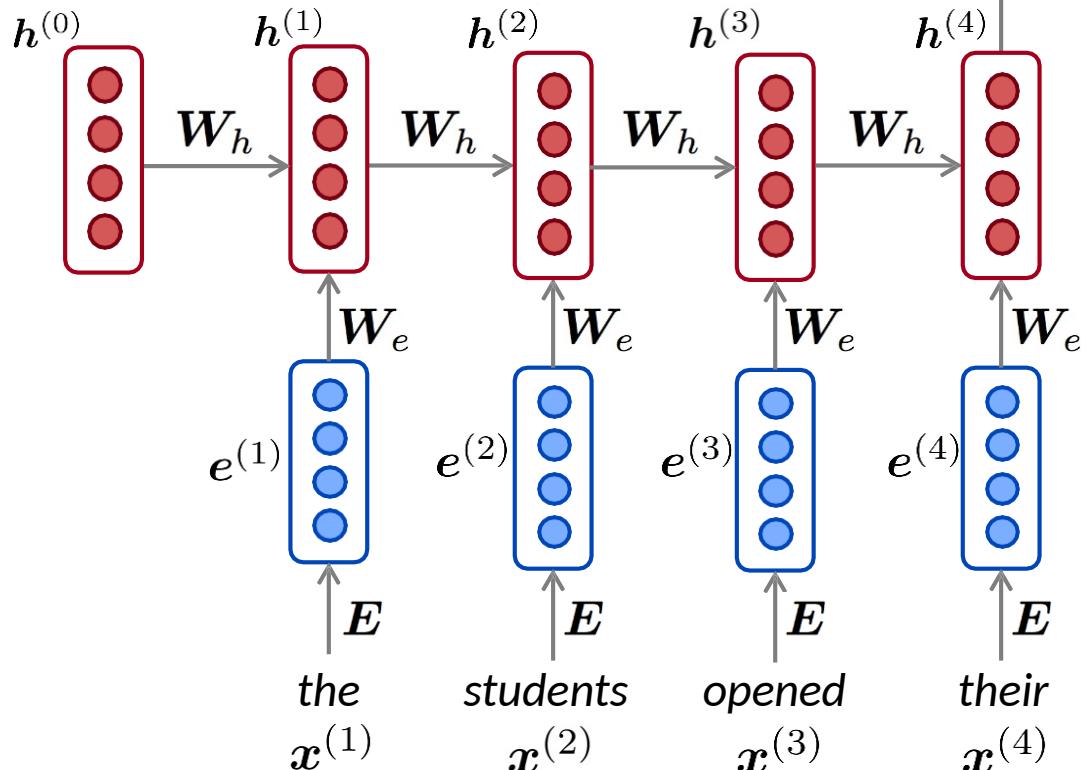
$\mathbf{h}^{(0)}$ is the initial hidden state

word embeddings

$$\mathbf{e}^{(t)} = \mathbf{E}\mathbf{x}^{(t)}$$

words / one-hot vectors

$$\mathbf{x}^{(t)} \in \mathbb{R}^{|V|}$$



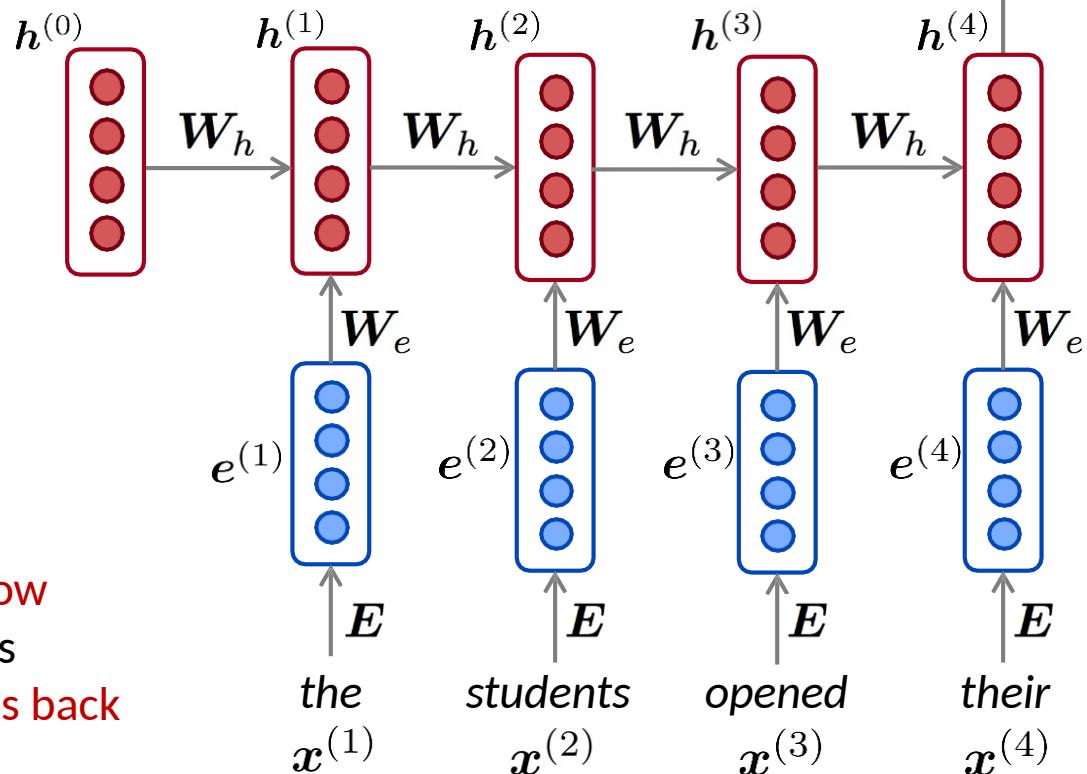
Note: this input sequence could be much longer, but this slide doesn't have space!

A RNN Language Model

$\hat{y}^{(4)} = P(\mathbf{x}^{(5)} | \text{the students opened their}$

RNN Advantages:

- Can process **any length** input
- Computation for step t can (in theory) use information from **many steps back**
- Model size **doesn't increase** for longer input
- Same weights applied on every timestep, so there is **symmetry** in how inputs are processed



RNN Disadvantages:

- Recurrent computation is **slow**
- In practice, difficult to access information from **many steps back**

Recall: Training a RNN Language Model

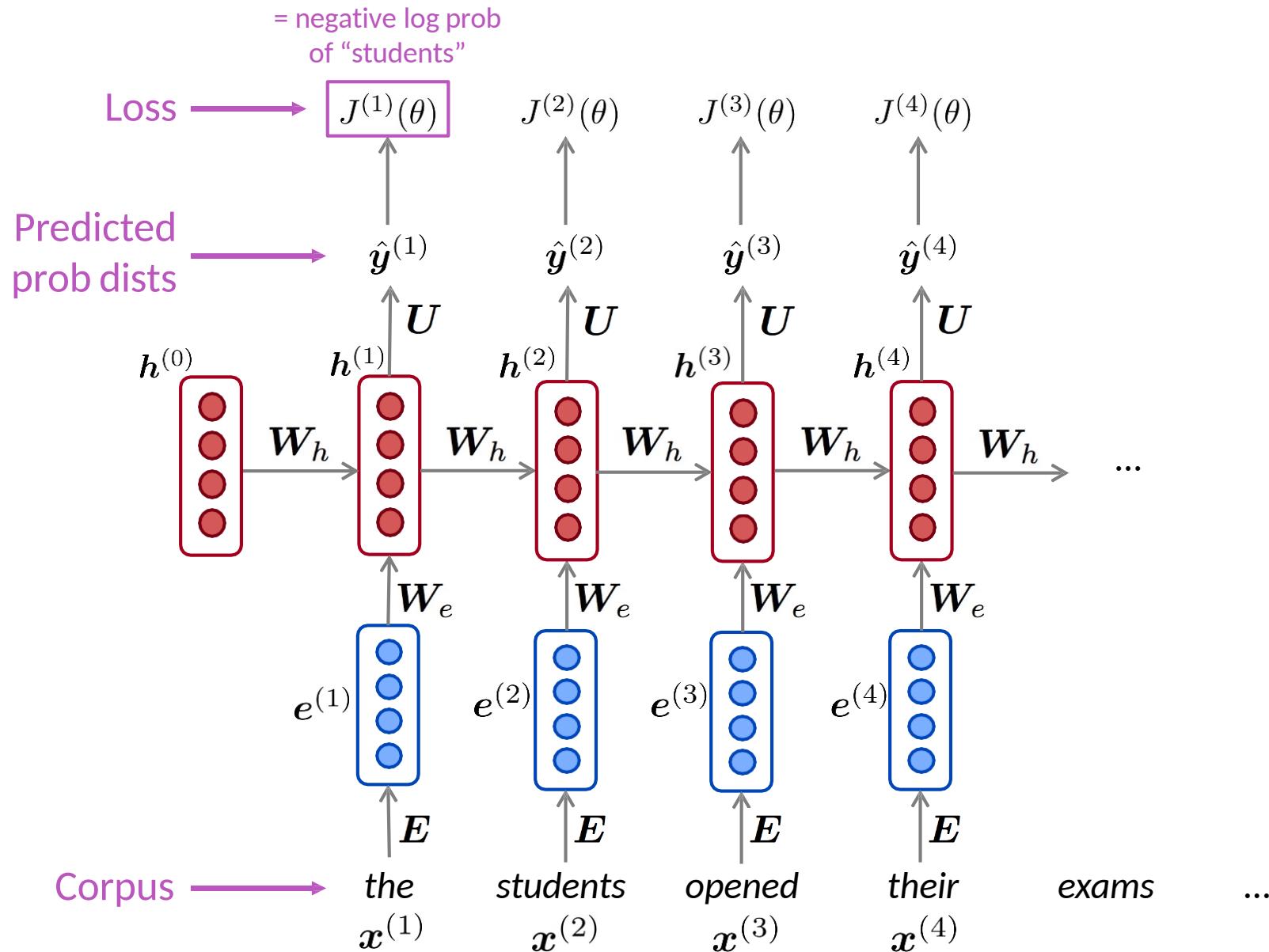
- Get a **big corpus of text** which is a sequence of words $\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(T)}$
- Feed into RNN-LM; compute output distribution $\hat{\mathbf{y}}^{(t)}$ **for every step t .**
 - i.e. predict probability distribution of *every word*, given words so far
- **Loss function** on step t is **cross-entropy** between predicted probability distribution $\hat{\mathbf{y}}^{(t)}$, and the true next word $\mathbf{y}^{(t)}$ (one-hot for $\mathbf{x}^{(t+1)}$):

$$J^{(t)}(\theta) = CE(\mathbf{y}^{(t)}, \hat{\mathbf{y}}^{(t)}) = - \sum_{w \in V} \mathbf{y}_w^{(t)} \log \hat{\mathbf{y}}_w^{(t)} = - \log \hat{\mathbf{y}}_{\mathbf{x}_{t+1}}^{(t)}$$

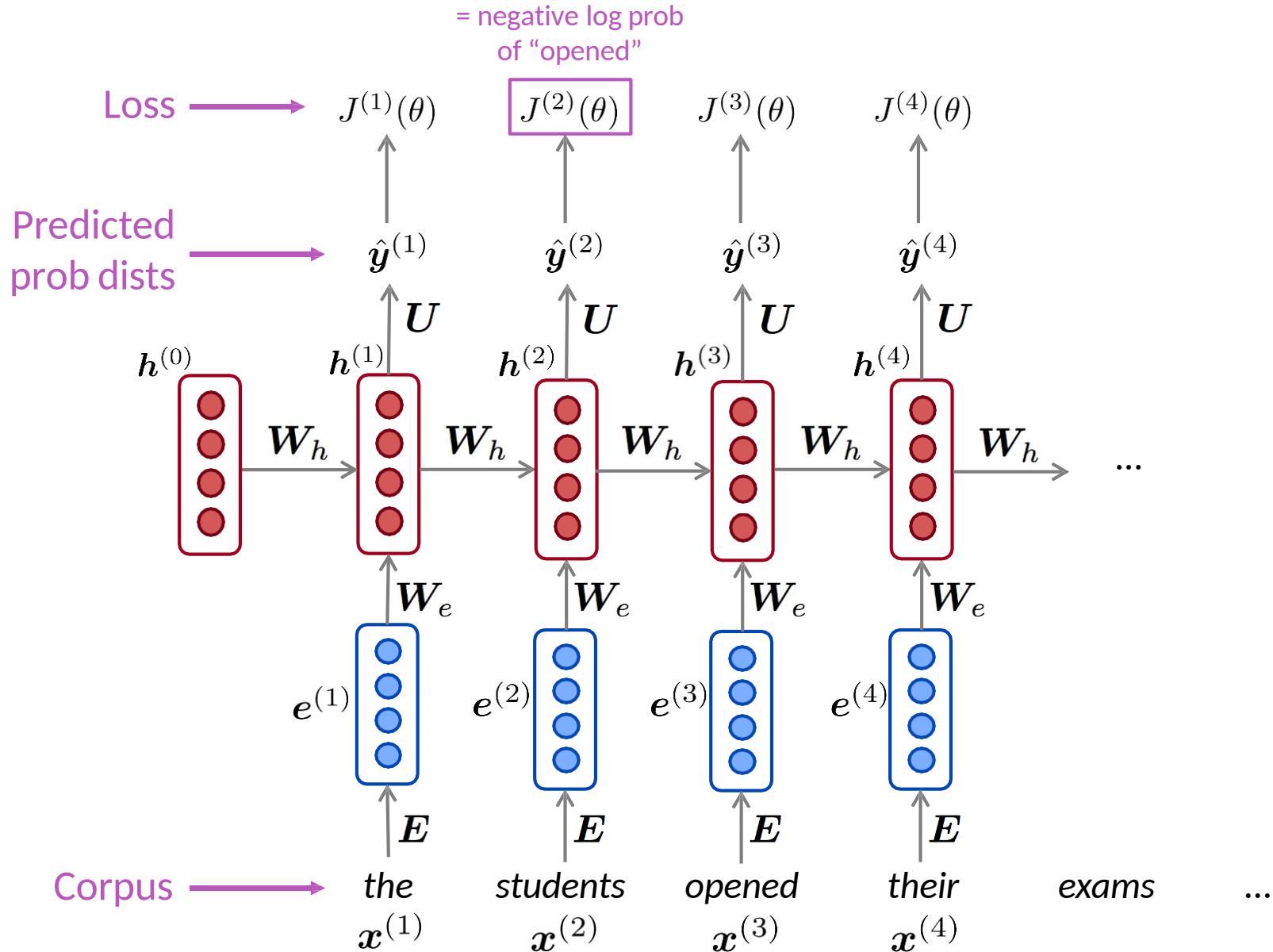
- Average this to get **overall loss** for entire training set:

$$J(\theta) = \frac{1}{T} \sum_{t=1}^T J^{(t)}(\theta) = \frac{1}{T} \sum_{t=1}^T - \log \hat{\mathbf{y}}_{\mathbf{x}_{t+1}}^{(t)}$$

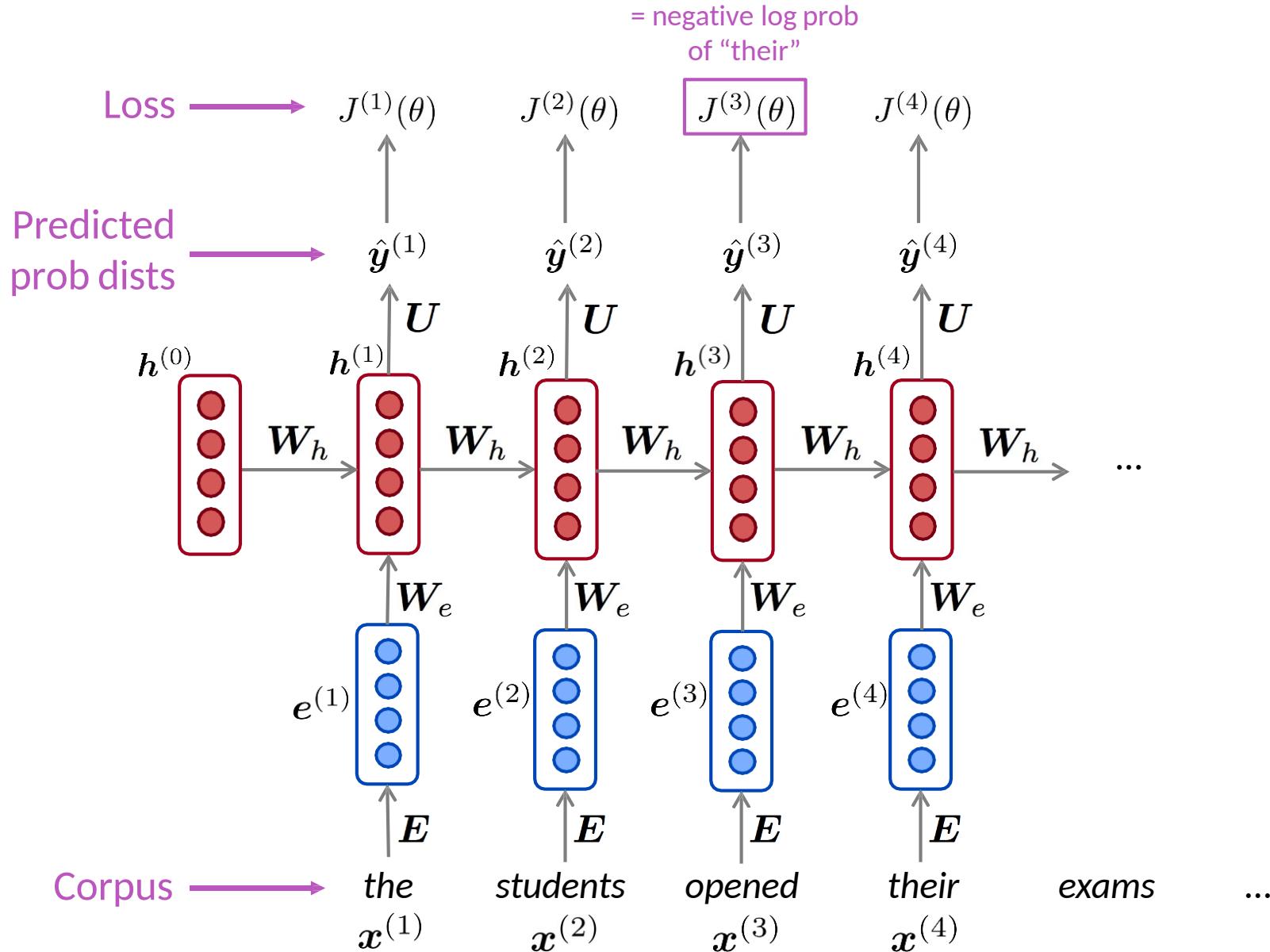
Training a RNN Language Model



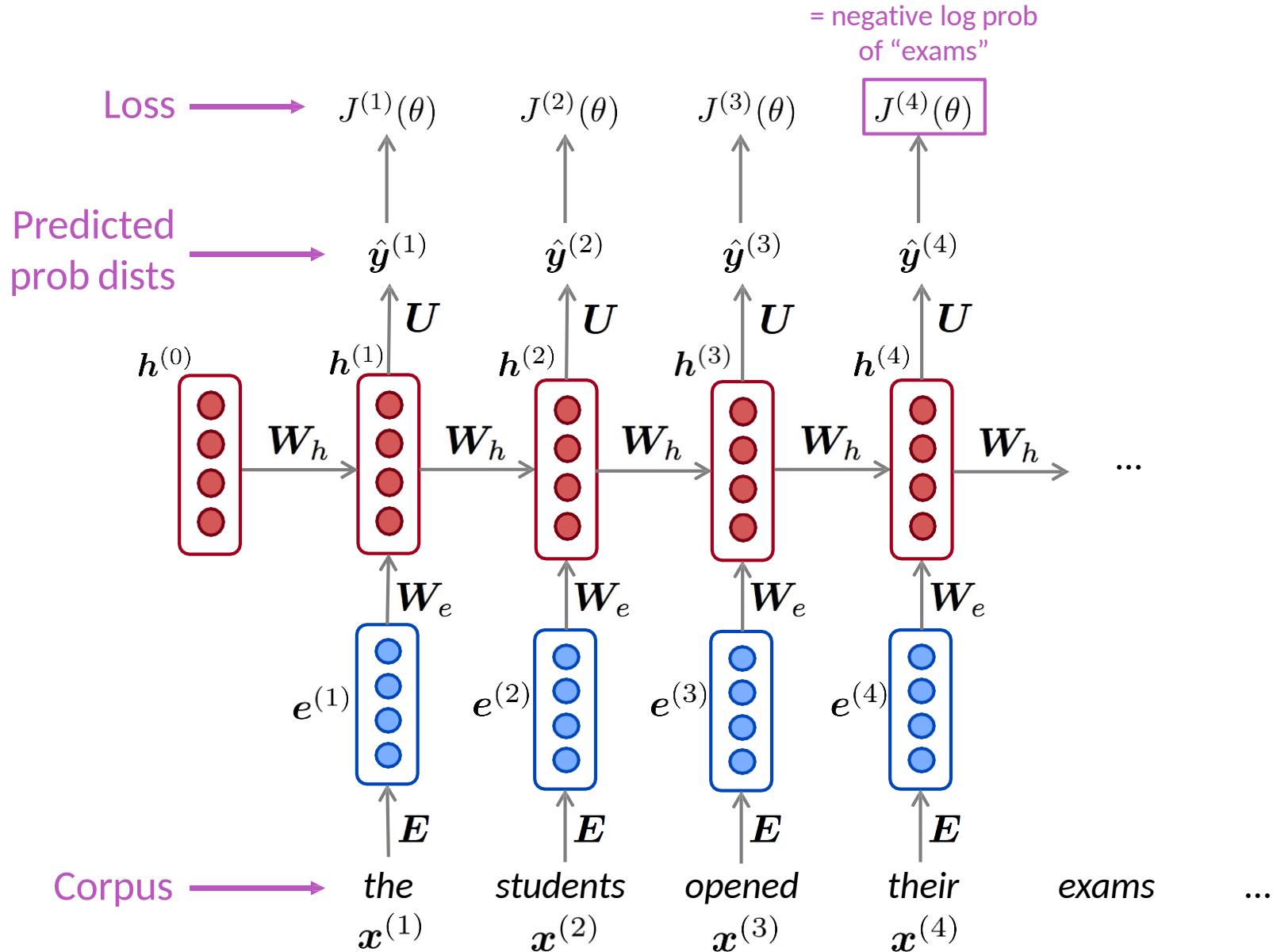
Training a RNN Language Model



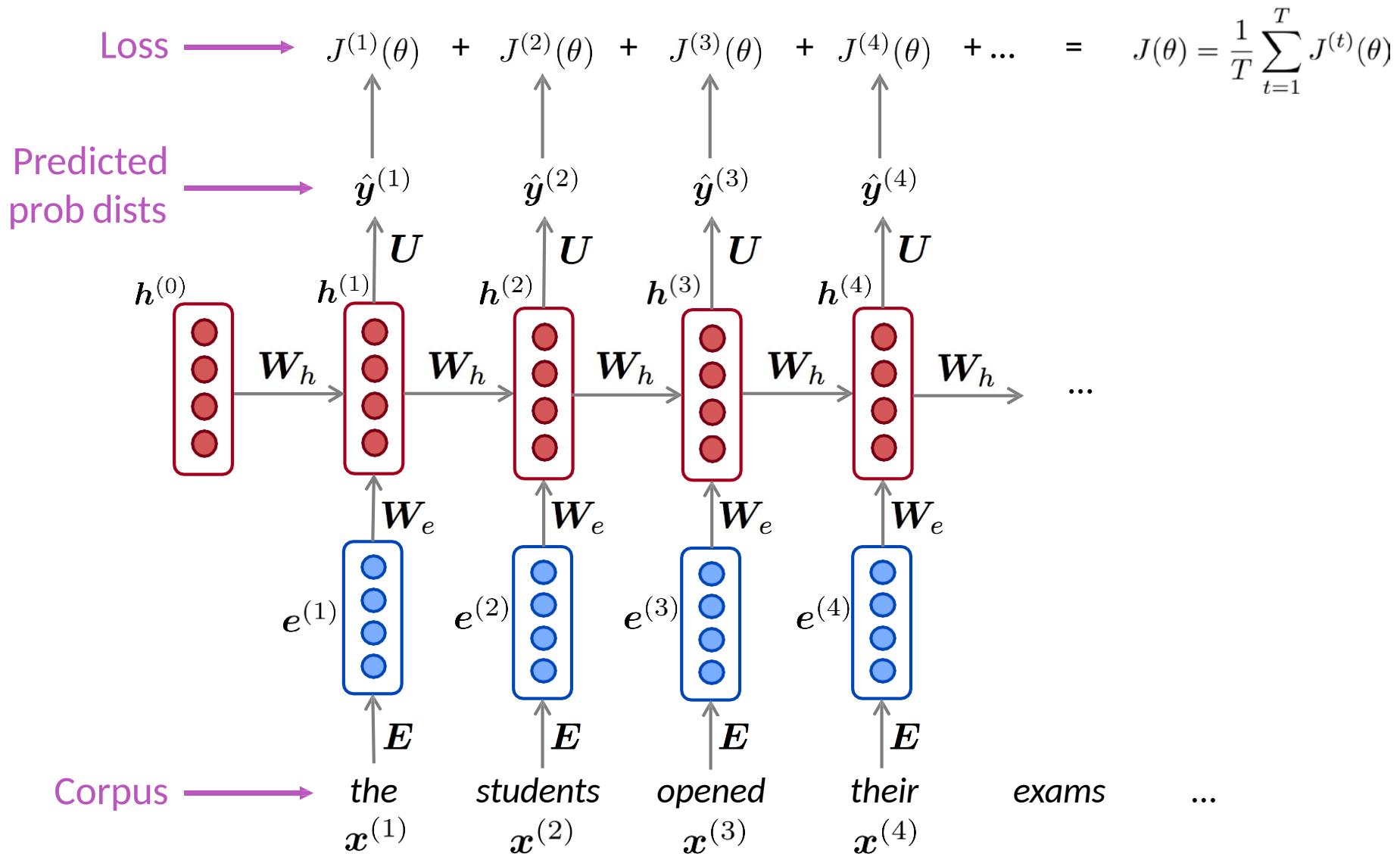
Training a RNN Language Model



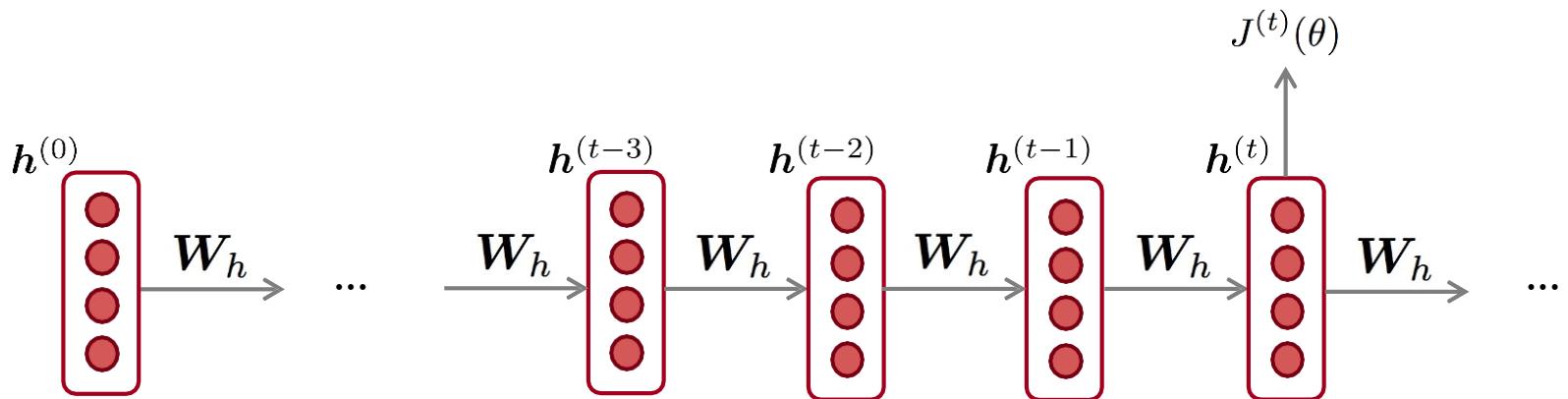
Training a RNN Language Model



Training a RNN Language Model



Backpropagation for RNNs



Question: What's the derivative of $J^{(t)}(\theta)$ w.r.t. the **repeated** weight matrix W_h ?

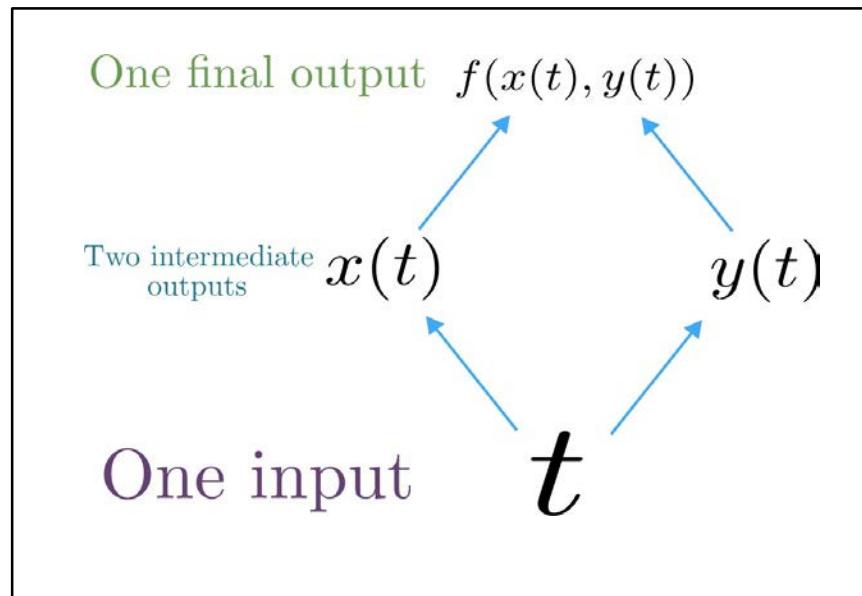
Answer:
$$\frac{\partial J^{(t)}}{\partial W_h} = \sum_{i=1}^t \frac{\partial J^{(t)}}{\partial W_h} \Big|_{(i)}$$

“The gradient w.r.t. a repeated weight
is the sum of the gradient
w.r.t. each time it appears”

Multivariable Chain Rule

- Given a multivariable function $f(x, y)$, and two single variable functions $x(t)$ and $y(t)$, here's what the multivariable chain rule says:

$$\underbrace{\frac{d}{dt} f(\textcolor{teal}{x}(t), \textcolor{red}{y}(t))}_{\text{Derivative of composition function}} = \frac{\partial f}{\partial \textcolor{teal}{x}} \frac{dx}{dt} + \frac{\partial f}{\partial \textcolor{red}{y}} \frac{dy}{dt}$$



Source:

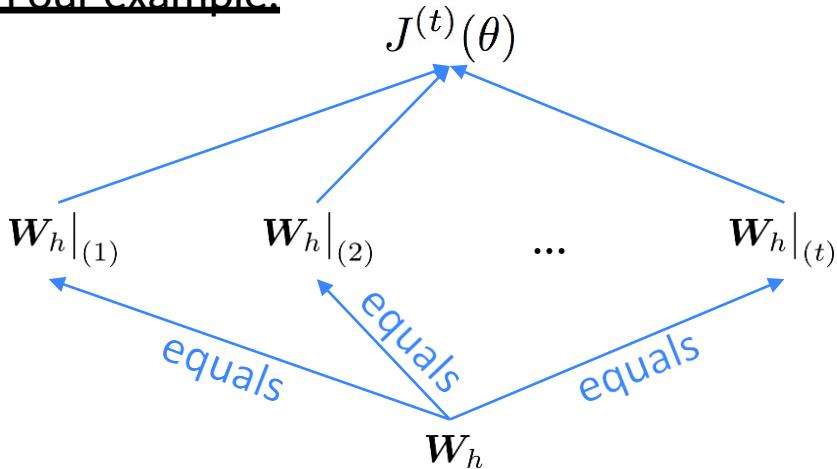
<https://www.khanacademy.org/math/multivariable-calculus/multivariable-derivatives/differentiating-vector-valued-functions/a/multivariable-chain-rule-simple-version>

Backpropagation for RNNs: Proof sketch

- Given a multivariable function $f(x, y)$, and two single variable functions $x(t)$ and $y(t)$, here's what the multivariable chain rule says:

$$\underbrace{\frac{d}{dt} f(\textcolor{teal}{x}(t), \textcolor{red}{y}(t))}_{\text{Derivative of composition function}} = \frac{\partial f}{\partial \textcolor{teal}{x}} \frac{d\textcolor{teal}{x}}{dt} + \frac{\partial f}{\partial \textcolor{red}{y}} \frac{d\textcolor{red}{y}}{dt}$$

In our example:



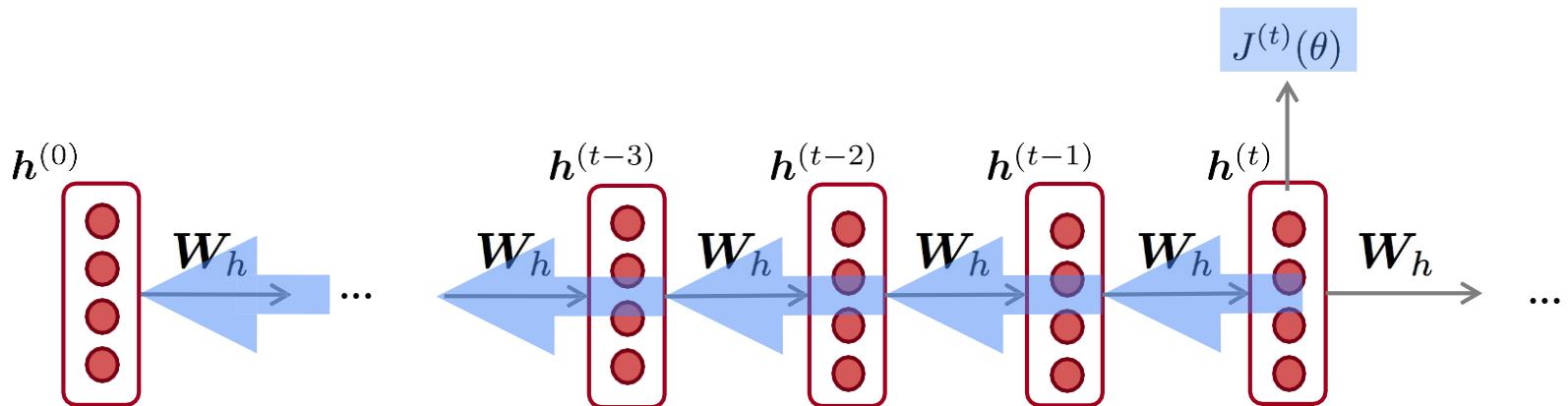
Apply the multivariable chain rule:

$$\begin{aligned}\frac{\partial J^{(t)}}{\partial \mathbf{W}_h} &= \sum_{i=1}^t \frac{\partial J^{(t)}}{\partial \mathbf{W}_h} \Big|_{(i)} \boxed{\frac{\partial \mathbf{W}_h|_{(i)}}{\partial \mathbf{W}_h}} \\ &= \sum_{i=1}^t \frac{\partial J^{(t)}}{\partial \mathbf{W}_h} \Big|_{(i)}\end{aligned}$$

Source:

<https://www.khanacademy.org/math/multivariable-calculus/multivariable-derivatives/differentiating-vector-valued-functions/a/multivariable-chain-rule-simple-version>

Backpropagation for RNNs

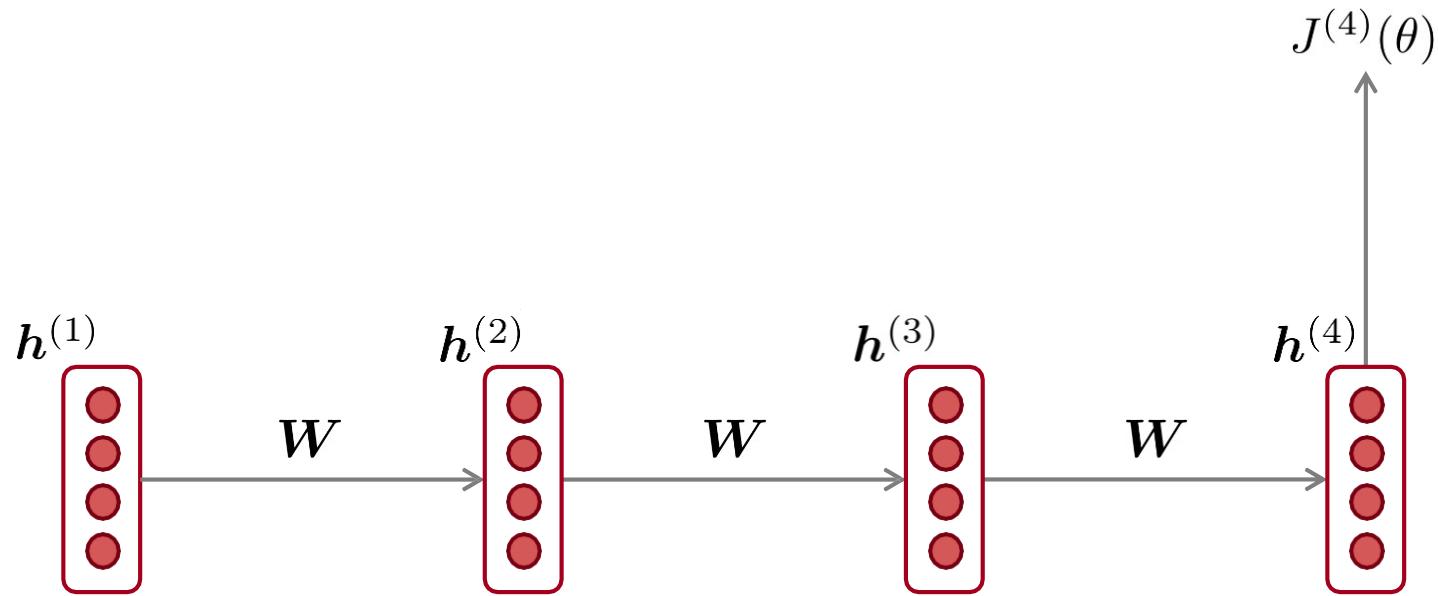


$$\frac{\partial J^{(t)}}{\partial \mathbf{W}_h} = \sum_{i=1}^t \frac{\partial J^{(t)}}{\partial \mathbf{W}_h} \Big|_{(i)}$$

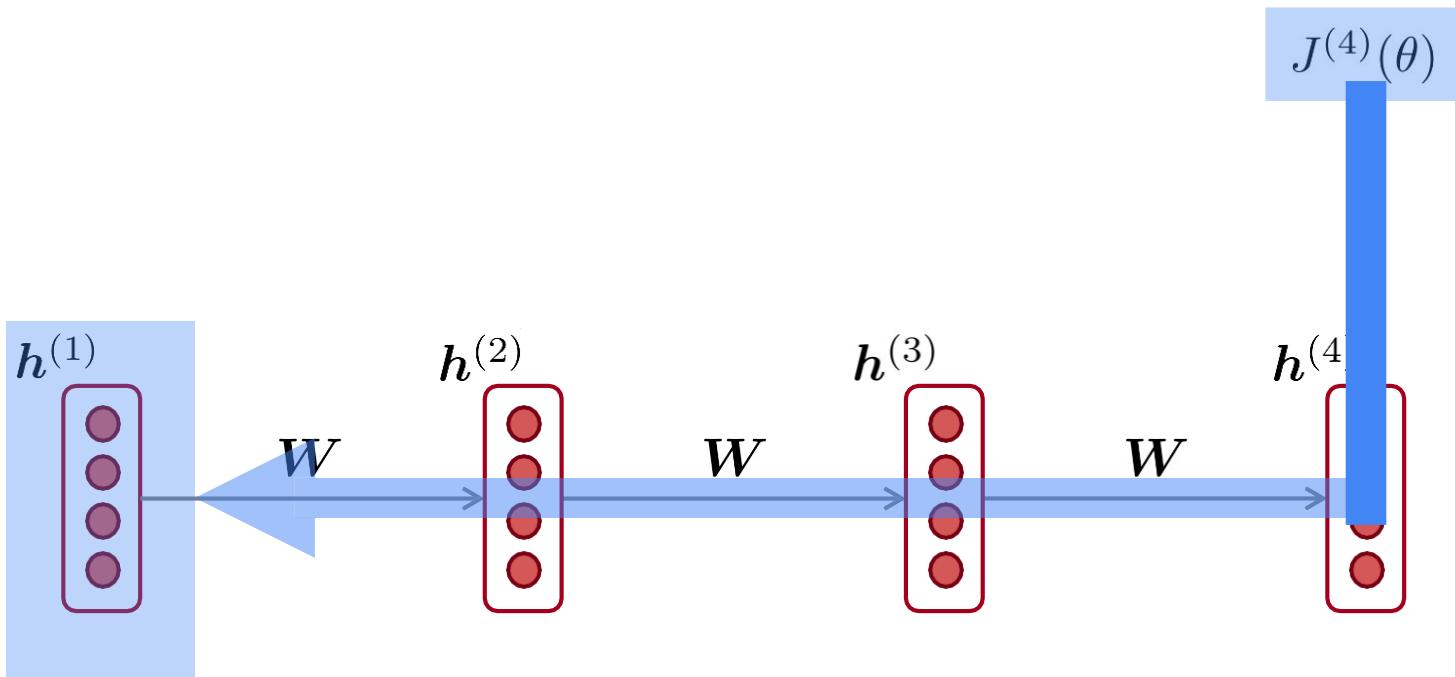
Question: How do we calculate this?

Answer: Backpropagate over timesteps $i=t, \dots, 0$, summing gradients as you go.
This algorithm is called “backpropagation through time”

Vanishing gradient intuition

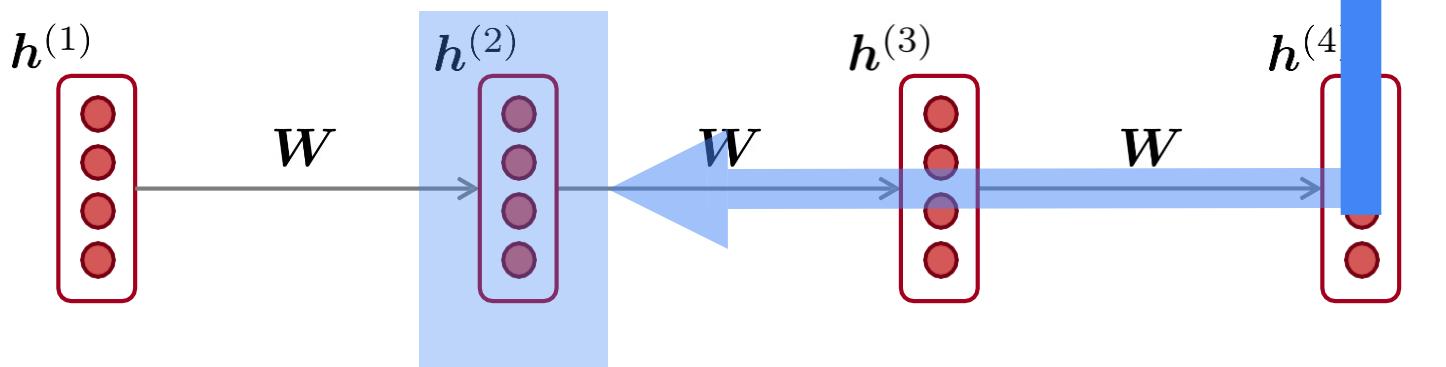


Vanishing gradient intuition



$$\frac{\partial J^{(4)}}{\partial h^{(1)}} = ?$$

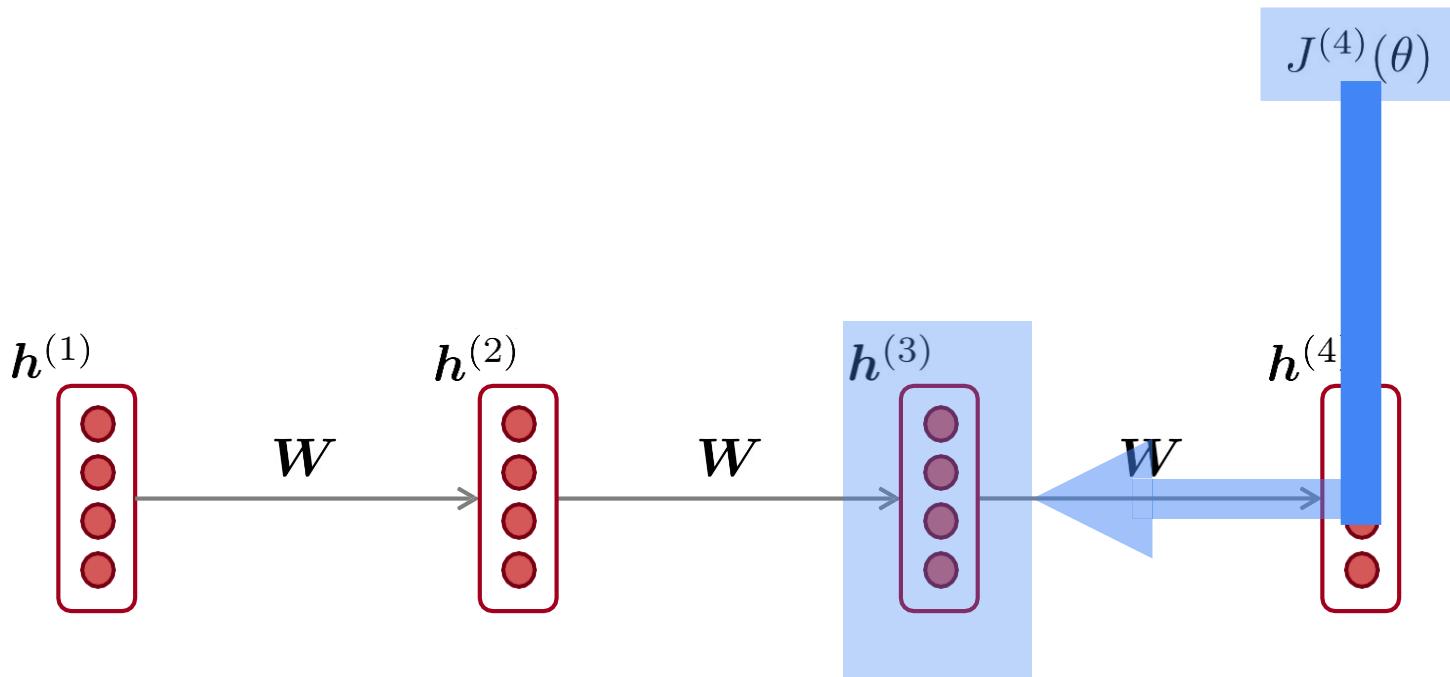
Vanishing gradient intuition



$$\frac{\partial J^{(4)}}{\partial h^{(1)}} = \frac{\partial h^{(2)}}{\partial h^{(1)}} \times \frac{\partial J^{(4)}}{\partial h^{(2)}}$$

chain rule!

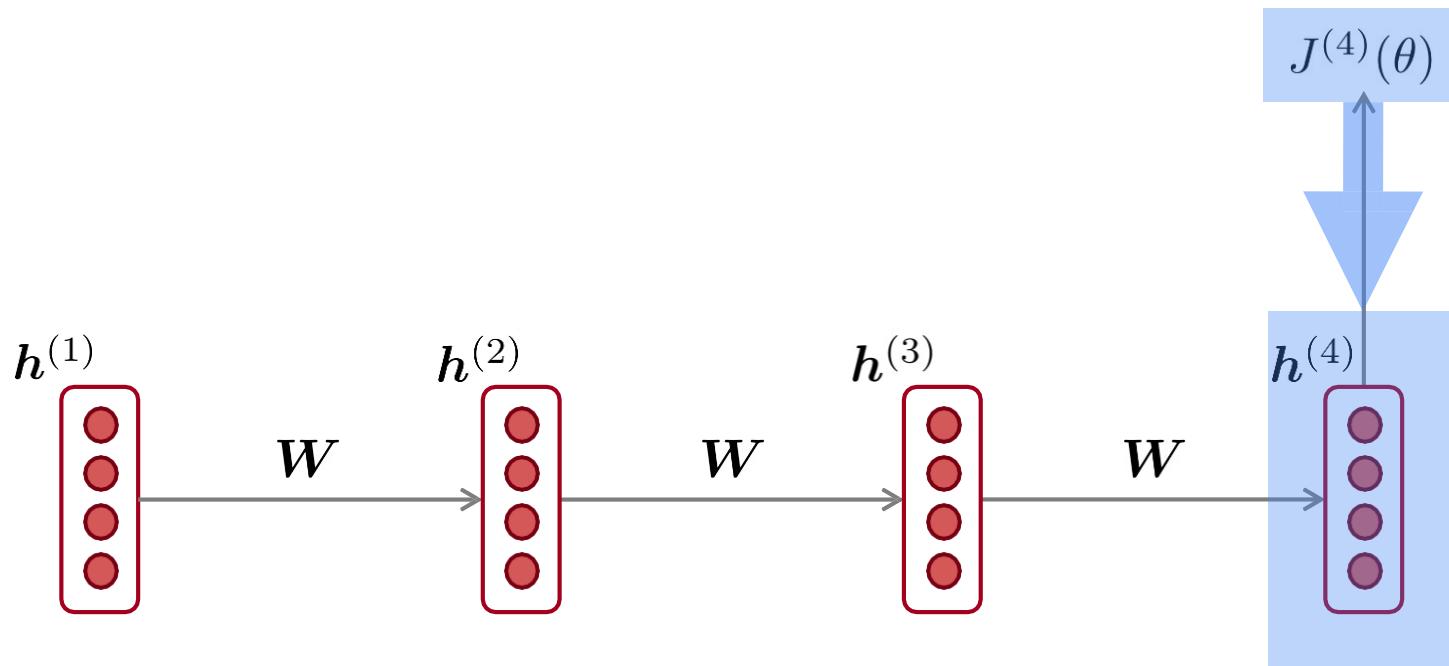
Vanishing gradient intuition



$$\frac{\partial J^{(4)}}{\partial h^{(1)}} = \frac{\partial h^{(2)}}{\partial h^{(1)}} \times \dots \frac{\partial h^{(3)}}{\partial h^{(2)}} \times \frac{\partial J^{(4)}}{\partial h^{(3)}}$$

chain rule!

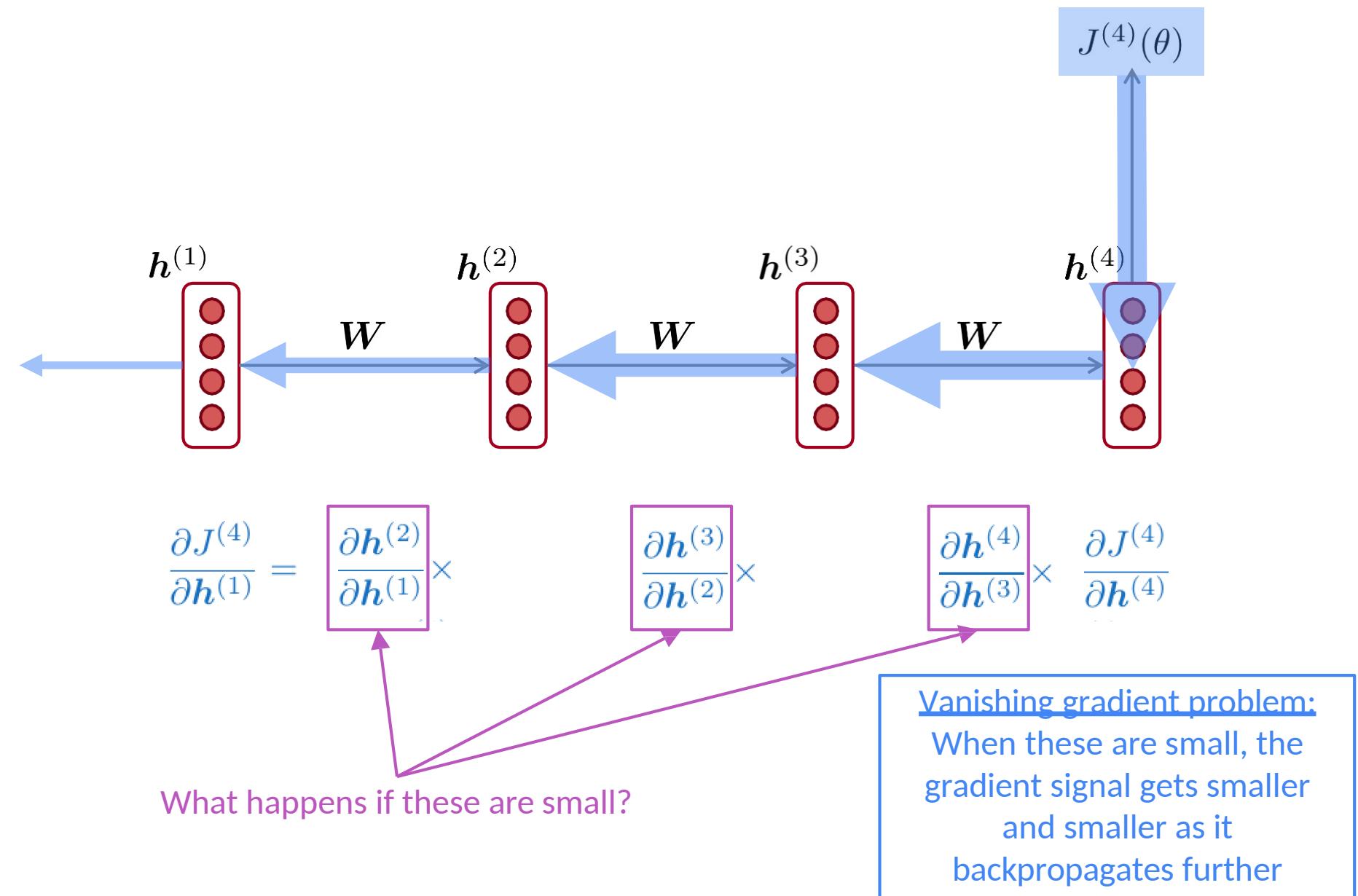
Vanishing gradient intuition



$$\frac{\partial J^{(4)}}{\partial h^{(1)}} = \frac{\partial h^{(2)}}{\partial h^{(1)}} \times \dots \quad \frac{\partial h^{(3)}}{\partial h^{(2)}} \times \quad \frac{\partial h^{(4)}}{\partial h^{(3)}} \times \frac{\partial J^{(4)}}{\partial h^{(4)}}$$

chain rule!

Vanishing gradient intuition



Vanishing gradient proof sketch

- Recall: $\mathbf{h}^{(t)} = \sigma(\mathbf{W}_h \mathbf{h}^{(t-1)} + \mathbf{W}_x \mathbf{x}^{(t)} + \mathbf{b}_1)$
- Therefore: $\frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{h}^{(t-1)}} = \text{diag}\left(\sigma'\left(\mathbf{W}_h \mathbf{h}^{(t-1)} + \mathbf{W}_x \mathbf{x}^{(t)} + \mathbf{b}_1\right)\right) \mathbf{W}_h$ (chain rule)
- Consider the gradient of the loss $J^{(i)}(\theta)$ on step i , with respect to the hidden state $\mathbf{h}^{(j)}$ on some previous step j .

$$\begin{aligned}\frac{\partial J^{(i)}(\theta)}{\partial \mathbf{h}^{(j)}} &= \frac{\partial J^{(i)}(\theta)}{\partial \mathbf{h}^{(i)}} \prod_{j < t \leq i} \frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{h}^{(t-1)}} && \text{(chain rule)} \\ &= \frac{\partial J^{(i)}(\theta)}{\partial \mathbf{h}^{(i)}} \boxed{\mathbf{W}_h^{(i-j)}} \prod_{j < t \leq i} \text{diag}\left(\sigma'\left(\mathbf{W}_h \mathbf{h}^{(t-1)} + \mathbf{W}_x \mathbf{x}^{(t)} + \mathbf{b}_1\right)\right) && \text{(value of } \frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{h}^{(t-1)}} \text{)}\end{aligned}$$

If \mathbf{W}_h is small, then this term gets vanishingly small as i and j get further apart

Source: "On the difficulty of training recurrent neural networks", Pascanu et al, 2013. <http://proceedings.mlr.press/v28/pascanu13.pdf>

Vanishing gradient proof sketch

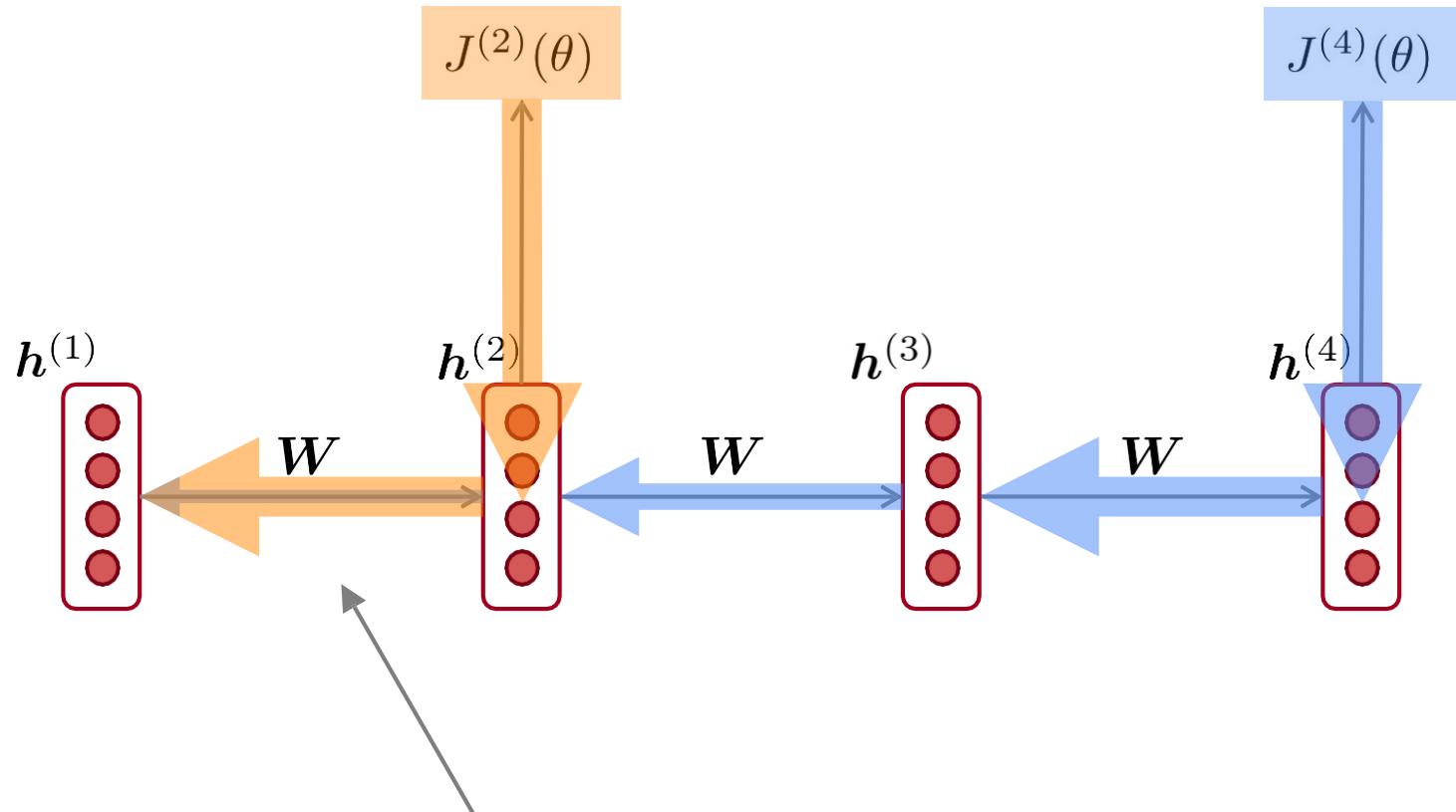
- Consider matrix L2 norms:

$$\left\| \frac{\partial J^{(i)}(\theta)}{\partial \mathbf{h}^{(j)}} \right\| \leq \left\| \frac{\partial J^{(i)}(\theta)}{\partial \mathbf{h}^{(i)}} \right\| \|\mathbf{W}_h\|^{(i-j)} \prod_{j < t \leq i} \left\| \text{diag} \left(\sigma' \left(\mathbf{W}_h \mathbf{h}^{(t-1)} + \mathbf{W}_x \mathbf{x}^{(t)} + \mathbf{b}_1 \right) \right) \right\|$$

- Pascanu et al showed that if the **largest eigenvalue** of \mathbf{W}_h is less than 1, then the gradient $\left\| \frac{\partial J^{(i)}(\theta)}{\partial \mathbf{h}^{(j)}} \right\|$ will **shrink** exponentially
- There's a similar proof relating a **largest eigenvalue > 1** to **exploding gradients**

Source: "On the difficulty of training recurrent neural networks", Pascanu et al, 2013. <http://proceedings.mlr.press/v28/pascanu13.pdf>

Why is vanishing gradient a problem?

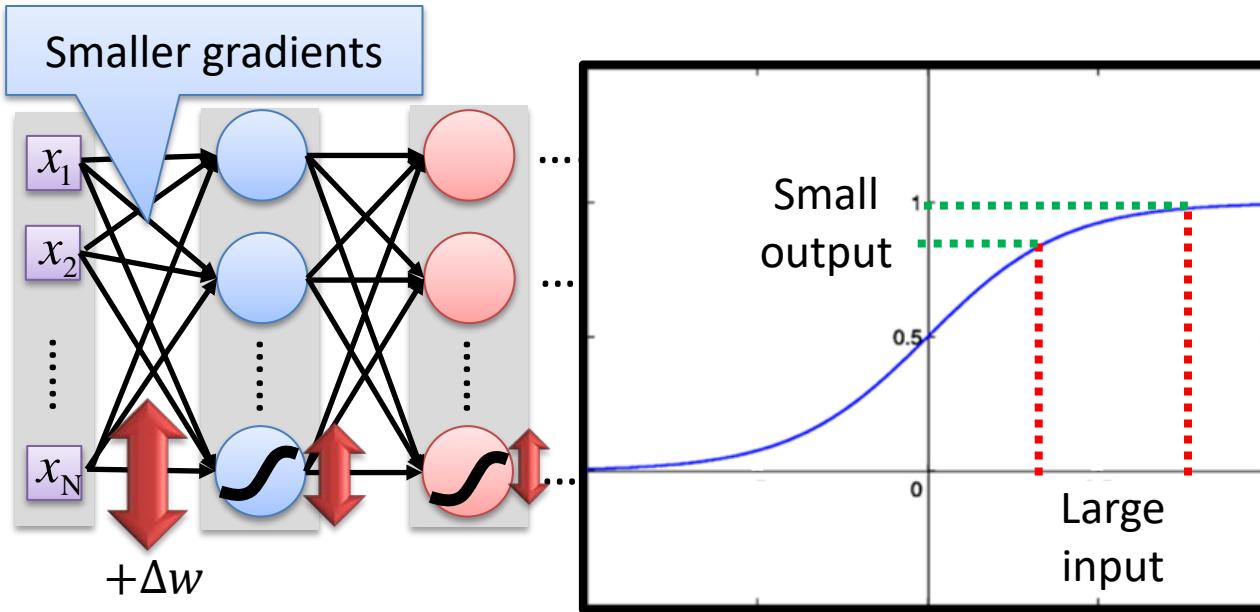


Gradient signal from faraway is lost because it's much smaller than gradient signal from close-by.

So model weights are only updated only with respect to near effects, not long-term effects.

Recipe of Deep Learning

- *Vanishing Gradient Problem*

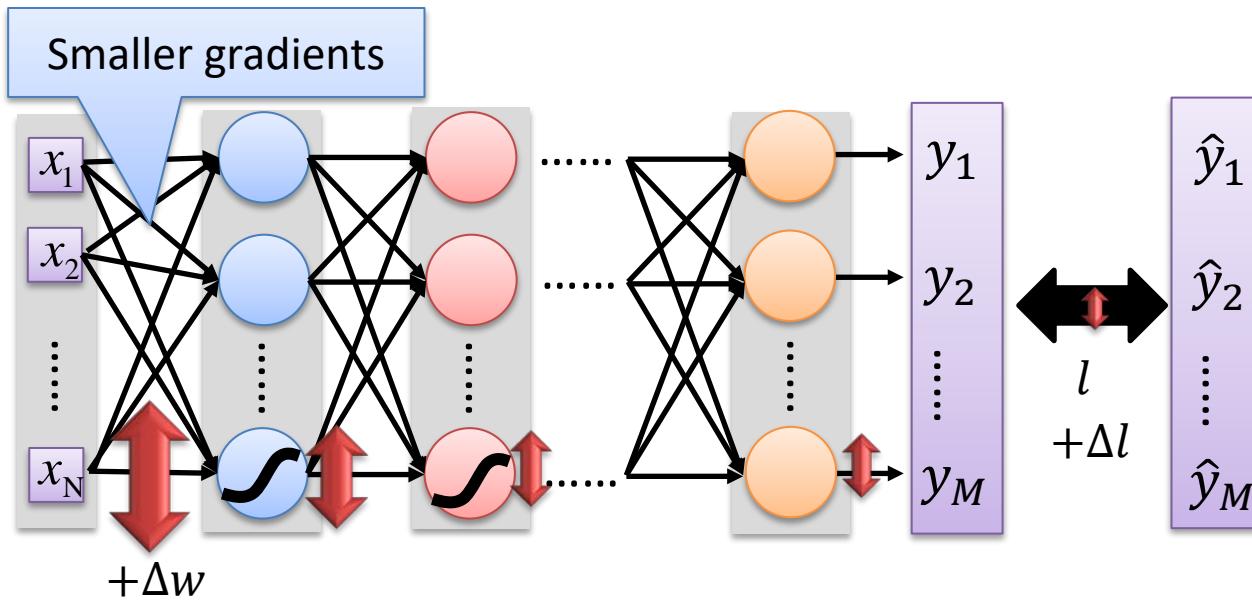


Intuitive way to compute the derivatives ...

$$\frac{\partial l}{\partial w} = ? \quad \frac{\Delta l}{\Delta w}$$

Recipe of Deep Learning

- *Vanishing Gradient Problem*

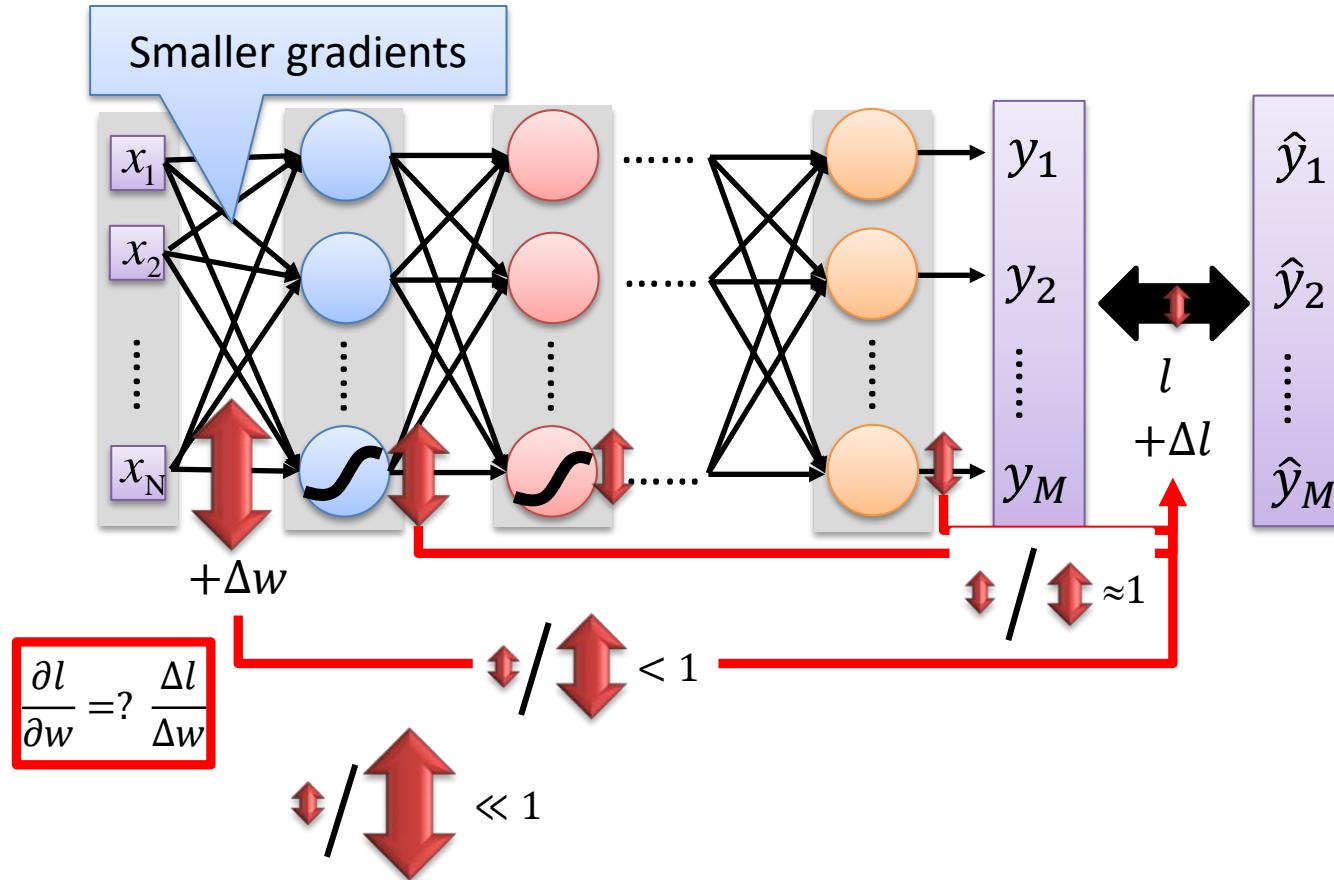


Intuitive way to compute the derivatives ...

$$\frac{\partial l}{\partial w} = ? \quad \frac{\Delta l}{\Delta w}$$

Recipe of Deep Learning

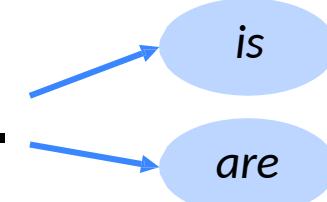
- *Vanishing Gradient Problem*



Effect of vanishing gradient on RNN-LM

- **LM task:** When she tried to print her _____, she found that the printer was out of toner. She went to the stationery store to buy more toner. It was very overpriced. After installing the toner into the printer, she finally printed her tickets.
- To learn from this training example, the RNN-LM needs to model the dependency between “tickets” on the 7th step and the target word “tickets” at the end.
- But if gradient is small, the model can't learn this dependency
 - So the model is unable to predict similar long-distance dependencies at test time

Effect of vanishing gradient on RNN-LM

- LM task: *The writer of the books* 
- Correct answer: *The writer of the books is planning a sequel*
- Syntactic recency: *The writer of the books is* (correct)
- Sequential recency: *The writer of the books are* (incorrect)
- Due to vanishing gradient, RNN-LMs are better at learning from sequential recency than syntactic recency, so they make this type of error more often than we'd like [Linzen et al 2016]

Why is exploding gradient a problem?

- If the gradient becomes too big, then the SGD update step becomes too big:

$$\theta^{new} = \theta^{old} - \underbrace{\alpha \nabla_{\theta} J(\theta)}_{\text{gradient}}$$

learning rate

- This can cause **bad updates**: we take too large a step and reach a bad parameter configuration (with large loss)
- In the worst case, this will result in **Inf** or **NaN** in your network (then you have to restart training from an earlier checkpoint)

Gradient clipping: solution for exploding gradient

- Gradient clipping: if the norm of the gradient is greater than some threshold, scale it down before applying SGD update

Algorithm 1 Pseudo-code for norm clipping

```
 $\hat{\mathbf{g}} \leftarrow \frac{\partial \mathcal{E}}{\partial \theta}$ 
if  $\|\hat{\mathbf{g}}\| \geq \text{threshold}$  then
     $\hat{\mathbf{g}} \leftarrow \frac{\text{threshold}}{\|\hat{\mathbf{g}}\|} \hat{\mathbf{g}}$ 
end if
```

- Intuition: take a step in the same direction, but a smaller step

RNNs with Gates

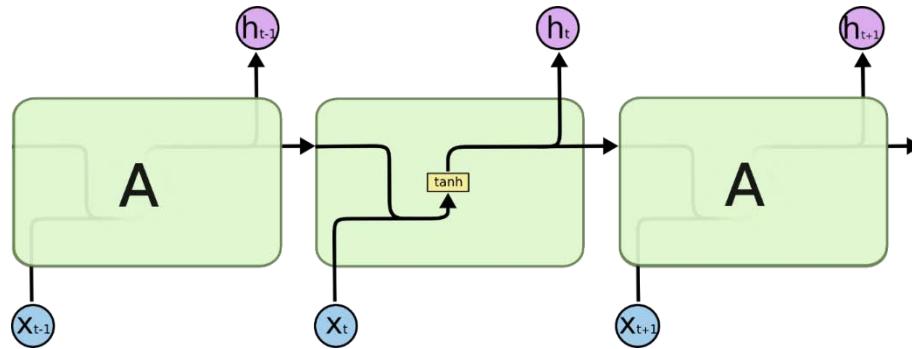
How to fix vanishing gradient problem?

- The main problem is that *it's too difficult for the RNN to learn to preserve information over many timesteps.*
- In a vanilla RNN, the hidden state is constantly being rewritten

$$\mathbf{h}^{(t)} = \sigma \left(\mathbf{W}_h \mathbf{h}^{(t-1)} + \mathbf{W}_x \mathbf{x}^{(t)} \right)$$

- How about a RNN with separate memory?

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



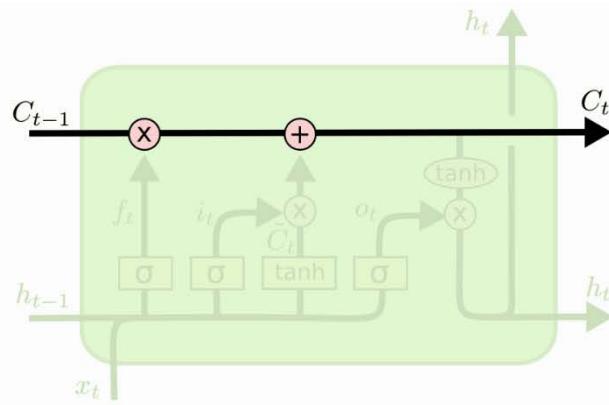
Unfortunately, as that gap grows, RNNs become unable to learn to connect the information (cf. vanishing gradients)

The problem was explored in depth by Hochreiter (1991) and Bengio, et al. (1994), who found some pretty fundamental reasons why it might be difficult.

Thankfully, LSTMs do not have this problem! (Hochreiter & Schmidhuber, 1997)

The Core Idea Behind LSTMs and GRUs

The cell state is kind of like a conveyor belt. It runs straight down the entire chain, with only some minor linear interactions. It's very easy for information to just flow along it unchanged.



The LSTM does have the ability to remove or add information to the cell state, carefully regulated by structures called gates.

Gated Recurrent Units (GRUs)

- More complex hidden unit computation in recurrence!
- Introduced by Cho et al. 2014
- Main ideas:
 - keep around memories to capture long distance dependencies
 - allow error messages to flow at different strengths depending on the inputs

Gated Recurrent Units (GRUs)

- Standard RNN computes hidden layer at next time step directly:
$$h_t = f \left(W^{(hh)} h_{t-1} + W^{(hx)} x_t \right)$$
- GRU first computes an update **gate** (another layer) based on current input word vector and hidden state

$$z_t = \sigma \left(W^{(z)} x_t + U^{(z)} h_{t-1} \right)$$

- Compute reset gate similarly but with different weights

$$r_t = \sigma \left(W^{(r)} x_t + U^{(r)} h_{t-1} \right)$$

Gated Recurrent Units (GRUs)

- Update gate

$$z_t = \sigma \left(W^{(z)} x_t + U^{(z)} h_{t-1} \right)$$

- Reset gate

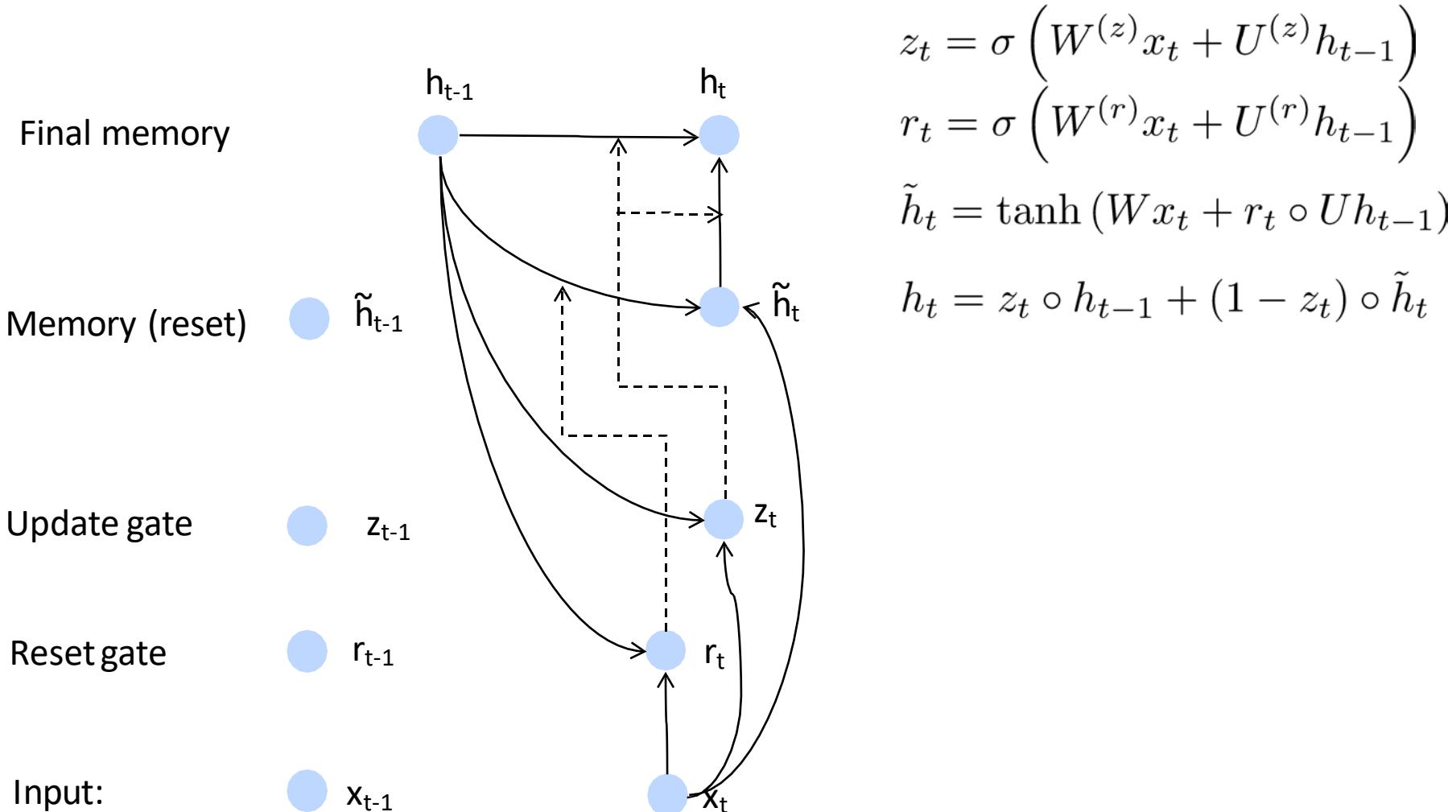
$$r_t = \sigma \left(W^{(r)} x_t + U^{(r)} h_{t-1} \right)$$

- New memory content: $\tilde{h}_t = \tanh (W x_t + r_t \circ U h_{t-1})$

If reset gate unit is ~ 0 , then this ignores previous memory and only stores the new word information

- Final memory at time step combines current and previous time steps: $h_t = z_t \circ h_{t-1} + (1 - z_t) \circ \tilde{h}_t$

Gated Recurrent Units (GRUs)



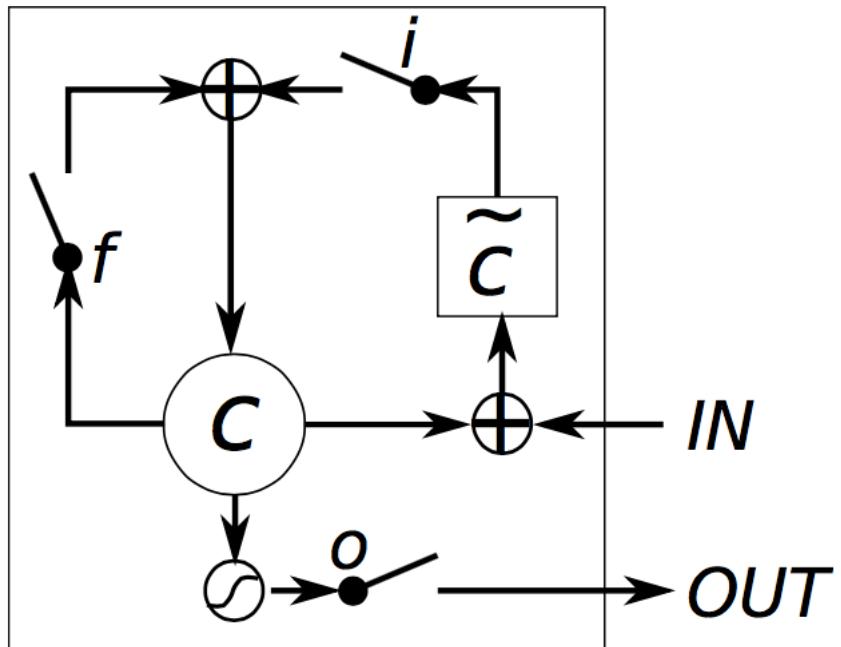
Gated Recurrent Units (GRUs)

- If reset r is close to 0, ignore previous hidden state: Allows model to drop information that is irrelevant in the future
 - If update z is close to 1, can copy information through many time steps, i.e. copy-paste state: Less vanishing gradient!
 - Units with short-term dependencies often have reset gates (r) very active; ones with long-term dependencies have active update gates (z)
- $$z_t = \sigma \left(W^{(z)} x_t + U^{(z)} h_{t-1} \right)$$
- $$r_t = \sigma \left(W^{(r)} x_t + U^{(r)} h_{t-1} \right)$$
- $$\tilde{h}_t = \tanh (W x_t + r_t \circ U h_{t-1})$$
- $$h_t = z_t \circ h_{t-1} + (1 - z_t) \circ \tilde{h}_t$$

Long-short-term-memories (LSTMs)

- Proposed by Hochreiter and Schmidhuber in 1997
- We can make the units even more complex
- Allow each time step to modify
 - Input gate (current cell matters) $i_t = \sigma(W^{(i)}x_t + U^{(i)}h_{t-1})$
 - Forget (gate 0, forget past) $f_t = \sigma(W^{(f)}x_t + U^{(f)}h_{t-1})$
 - Output (how much cell is exposed) $o_t = \sigma(W^{(o)}x_t + U^{(o)}h_{t-1})$
 - New memory cell $\tilde{c}_t = \tanh(W^{(c)}x_t + U^{(c)}h_{t-1})$
- Final memory cell: $c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t$
- Final hidden state: $h_t = o_t \circ \tanh(c_t)$

Long-short-term-memories (LSTMs)



$$i_t = \sigma(W^{(i)}x_t + U^{(i)}h_{t-1})$$
$$f_t = \sigma(W^{(f)}x_t + U^{(f)}h_{t-1})$$
$$o_t = \sigma(W^{(o)}x_t + U^{(o)}h_{t-1})$$
$$\tilde{c}_t = \tanh(W^{(c)}x_t + U^{(c)}h_{t-1})$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t$$

$$h_t = o_t \circ \tanh(c_t)$$

Intuition: memory cells can keep information intact, unless inputs makes them forget it or overwrite it with new input

Cell can decide to output this information or just store it

Review on your own: Gated Recurrent Units (GRU)

- Proposed by Cho et al. in 2014 as a simpler alternative to the LSTM.
- On each timestep t we have input $\mathbf{x}^{(t)}$ and hidden state $\mathbf{h}^{(t)}$ (no cell state).

Update gate: controls what parts of hidden state are updated vs preserved

Reset gate: controls what parts of previous hidden state are used to compute new content

New hidden state content: reset gate selects useful parts of prev hidden state. Use this and current input to compute new hidden content.

Hidden state: update gate simultaneously controls what is kept from previous hidden state, and what is updated to new hidden state content

$$\mathbf{u}^{(t)} = \sigma(\mathbf{W}_u \mathbf{h}^{(t-1)} + \mathbf{U}_u \mathbf{x}^{(t)} + \mathbf{b}_u)$$

$$\mathbf{r}^{(t)} = \sigma(\mathbf{W}_r \mathbf{h}^{(t-1)} + \mathbf{U}_r \mathbf{x}^{(t)} + \mathbf{b}_r)$$

$$\tilde{\mathbf{h}}^{(t)} = \tanh(\mathbf{W}_h (\mathbf{r}^{(t)} \circ \mathbf{h}^{(t-1)}) + \mathbf{U}_h \mathbf{x}^{(t)} + \mathbf{b}_h)$$

$$\mathbf{h}^{(t)} = (1 - \mathbf{u}^{(t)}) \circ \mathbf{h}^{(t-1)} + \mathbf{u}^{(t)} \circ \tilde{\mathbf{h}}^{(t)}$$

How does this solve vanishing gradient?
GRU makes it easier to retain info long-term (e.g. by setting update gate to 0)

Review on your own: Long Short-Term Memory (LSTM)

We have a sequence of inputs $\mathbf{x}^{(t)}$, and we will compute a sequence of hidden states $\mathbf{h}^{(t)}$ and cell states $\mathbf{c}^{(t)}$. On timestep t :

Forget gate: controls what is kept vs forgotten, from previous cell state

Input gate: controls what parts of the new cell content are written to cell

Output gate: controls what parts of cell are output to hidden state

New cell content: this is the new content to be written to the cell

Cell state: erase (“forget”) some content from last cell state, and write (“input”) some new cell content

Hidden state: read (“output”) some content from the cell

Sigmoid function: all gate values are between 0 and 1

$$\mathbf{f}^{(t)} = \sigma(\mathbf{W}_f \mathbf{h}^{(t-1)} + \mathbf{U}_f \mathbf{x}^{(t)} + \mathbf{b}_f)$$

$$\mathbf{i}^{(t)} = \sigma(\mathbf{W}_i \mathbf{h}^{(t-1)} + \mathbf{U}_i \mathbf{x}^{(t)} + \mathbf{b}_i)$$

$$\mathbf{o}^{(t)} = \sigma(\mathbf{W}_o \mathbf{h}^{(t-1)} + \mathbf{U}_o \mathbf{x}^{(t)} + \mathbf{b}_o)$$

$$\tilde{\mathbf{c}}^{(t)} = \tanh(\mathbf{W}_c \mathbf{h}^{(t-1)} + \mathbf{U}_c \mathbf{x}^{(t)} + \mathbf{b}_c)$$

$$\mathbf{c}^{(t)} = \mathbf{f}^{(t)} \circ \mathbf{c}^{(t-1)} + \mathbf{i}^{(t)} \circ \tilde{\mathbf{c}}^{(t)}$$

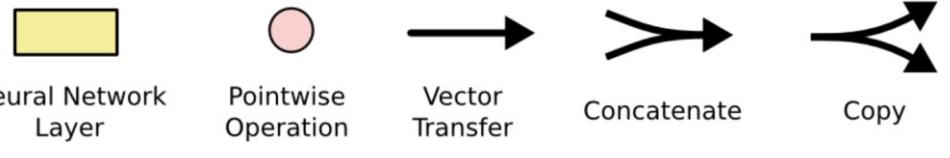
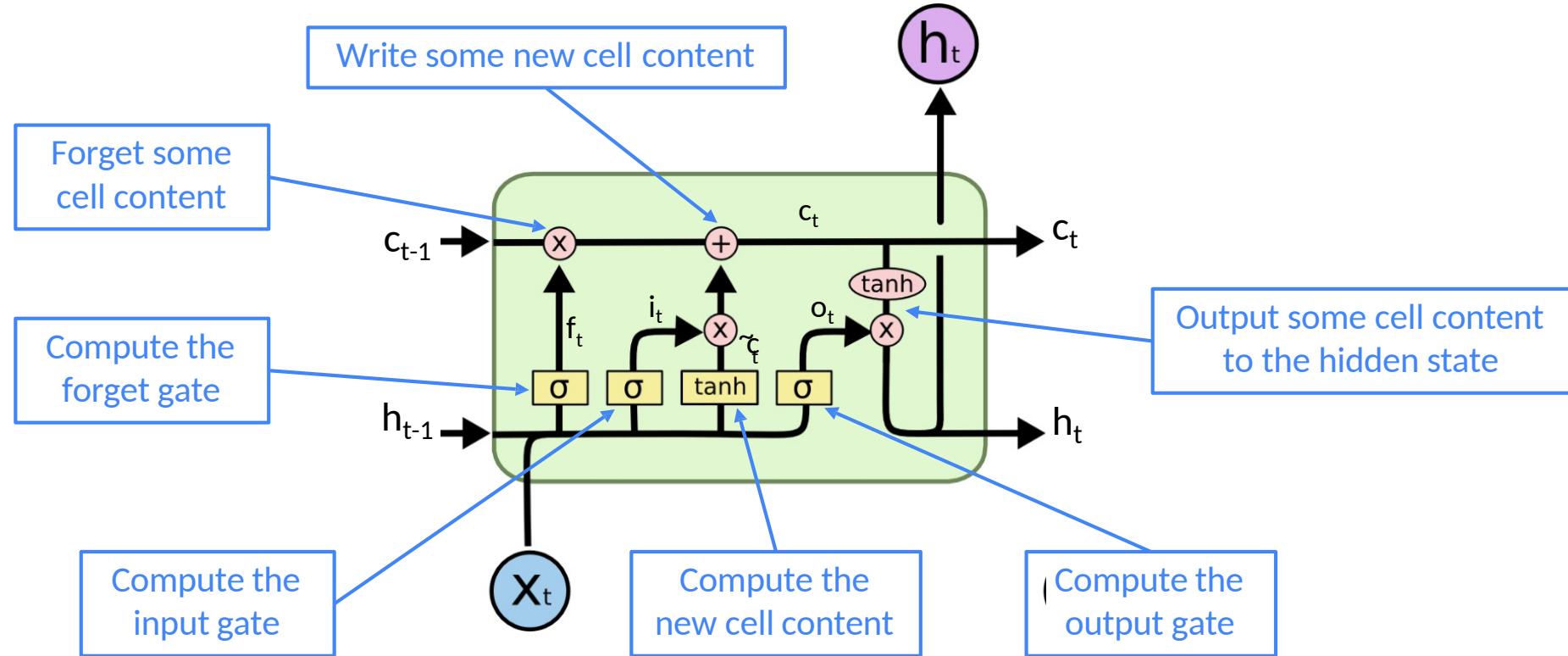
$$\mathbf{h}^{(t)} = \mathbf{o}^{(t)} \circ \tanh \mathbf{c}^{(t)}$$

Gates are applied using element-wise product

All these are vectors of same length n

Review on your own: Long Short-Term Memory (LSTM)

You can think of the LSTM equations visually like this:



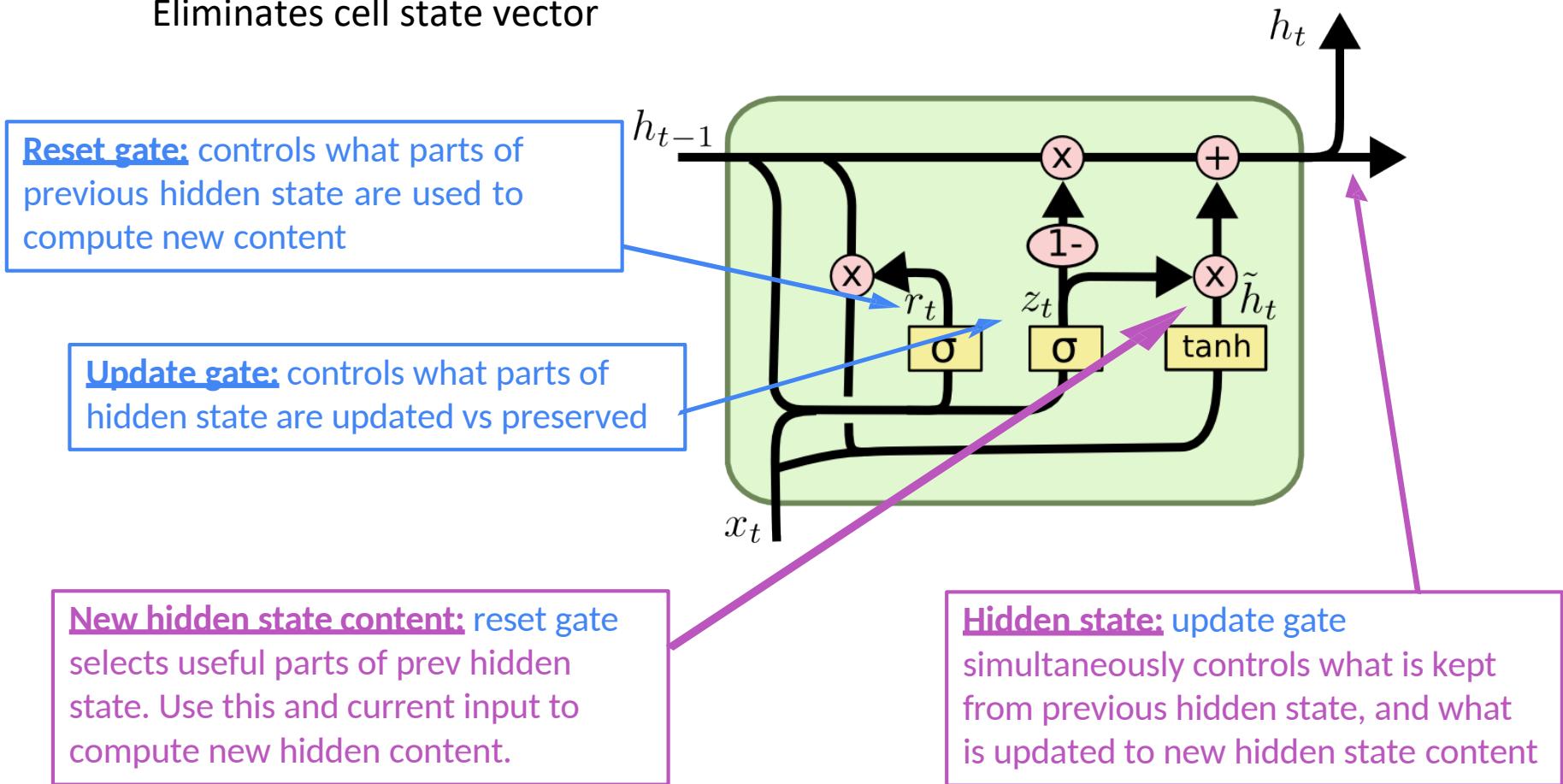
Source: <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>

Gated Recurrent Unit (GRU)

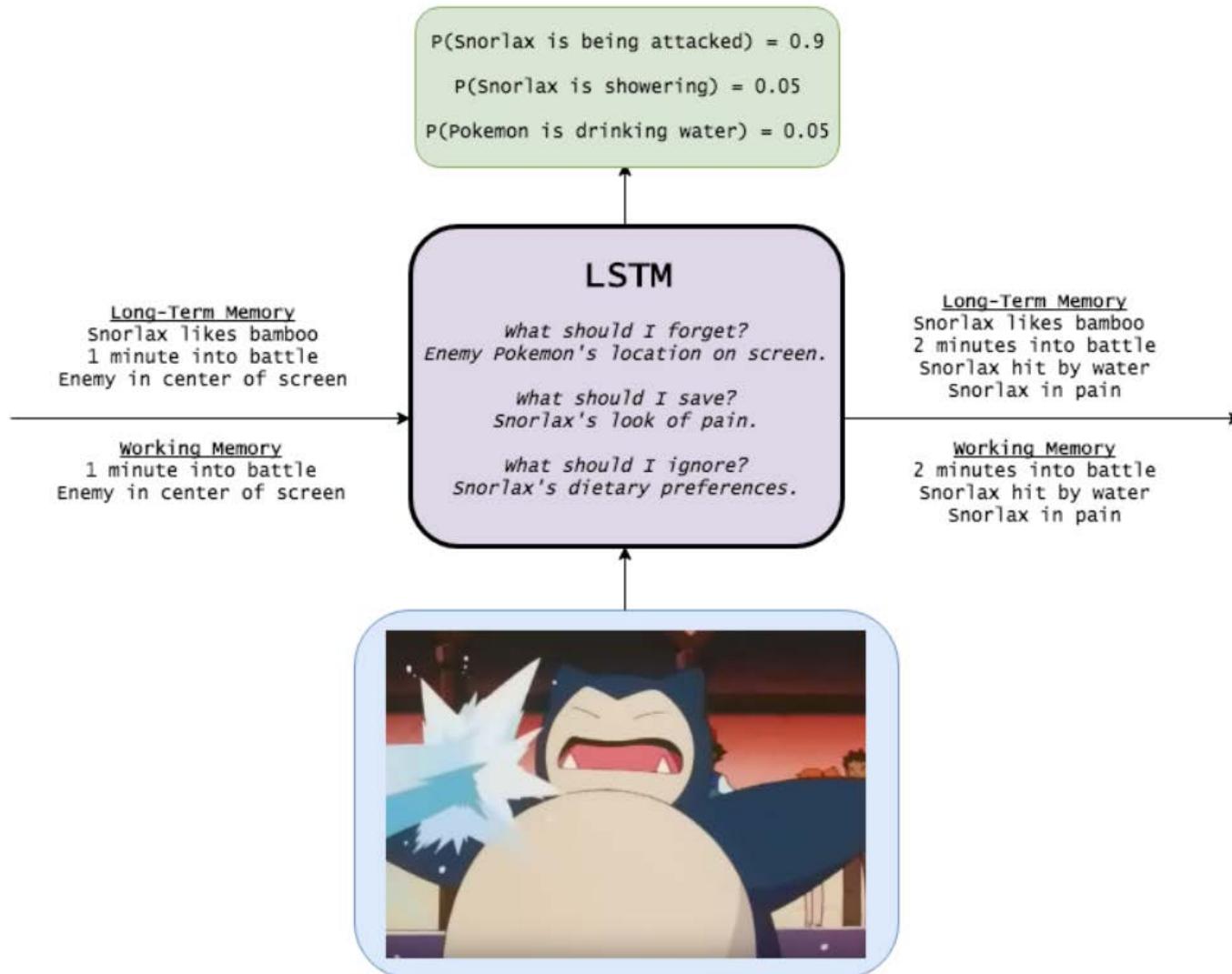
Alternative RNN to LSTM that uses fewer gates ([Cho, et al., 2014](#))

Combines forget and input gates into “update” gate.

Eliminates cell state vector



Cell State vs Hidden State



Activity

<http://blog.echen.me/2017/05/30/exploring-lstms/>

LSTM vs GRU

- Researchers have proposed many gated RNN variants, but LSTM and GRU are the most widely-used
- The biggest difference is that GRU is quicker to compute and has fewer parameters
- There is no conclusive evidence that one consistently performs better than the other
- LSTM is a good default choice (especially if your data has particularly long dependencies, or you have lots of training data)
- Rule of thumb: start with LSTM, but switch to GRU if you want something more efficient

LSTMs: real-world success

- In 2013-2015, LSTMs started achieving state-of-the-art results for sequence modeling
 - Successful tasks include: handwriting recognition, speech recognition, machine translation, parsing, image captioning
 - LSTM became the dominant approach
- Starting in 2019, other approaches (e.g. Transformers) became more dominant for certain NLP tasks (will discuss next lecture)
 - For example in WMT (machine translation competition):
 - In WMT 2016, the summary report contains "RNN" 44 times
 - In WMT 2018, the report contains "RNN" 9 times and "Transformer" 63 times

Source: "Findings of the 2016 Conference on Machine Translation (WMT16)", Bojar et al. 2016, <http://www.statmt.org/wmt16/pdf/W16-2301.pdf>

Source: "Findings of the 2018 Conference on Machine Translation (WMT18)", Bojar et al. 2018, <http://www.statmt.org/wmt18/pdf/WMT028.pdf>

Is vanishing/exploding gradient just a RNN problem?

- No! It can be a problem for all neural architectures (including **feed-forward** and **convolutional**), especially **deep** ones.
 - Due to chain rule / choice of nonlinearity function, gradient can become vanishingly small as it backpropagates
 - Thus lower layers are learnt very slowly (hard to train)
 - Solution: lots of new deep feedforward/convolutional architectures that **add more direct connections** (thus allowing the gradient to flow)

For example:

- **Residual connections** aka “ResNet”
- Also known as **skip-connections**
- The **identity connection** **preserves information** by default
- This makes **deep** networks much easier to train

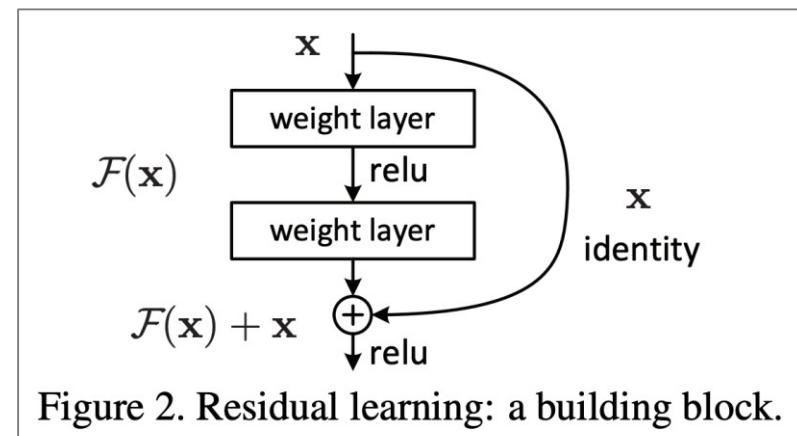


Figure 2. Residual learning: a building block.

Is vanishing/exploding gradient just a RNN problem?

- No! It can be a problem for all neural architectures (including **feed-forward** and **convolutional**), especially **deep** ones.
 - Due to chain rule / choice of nonlinearity function, gradient can become vanishingly small as it backpropagates
 - Thus lower layers are learnt very slowly (hard to train)
 - Solution: lots of new deep feedforward/convolutional architectures that **add more direct connections** (thus allowing the gradient to flow)

For example:

- **Dense connections** aka “DenseNet”
- Directly connect everything to everything!

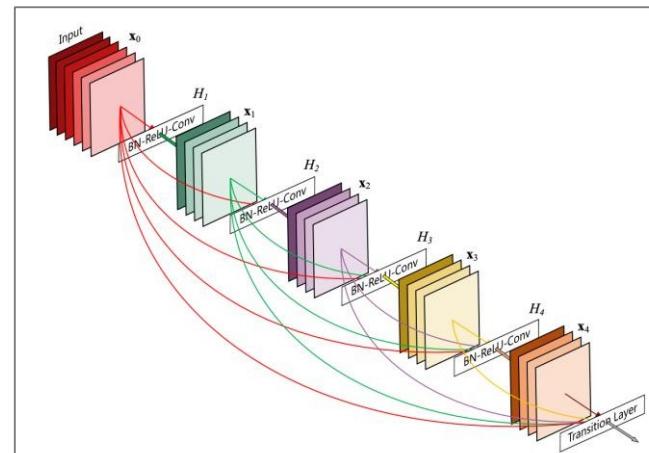
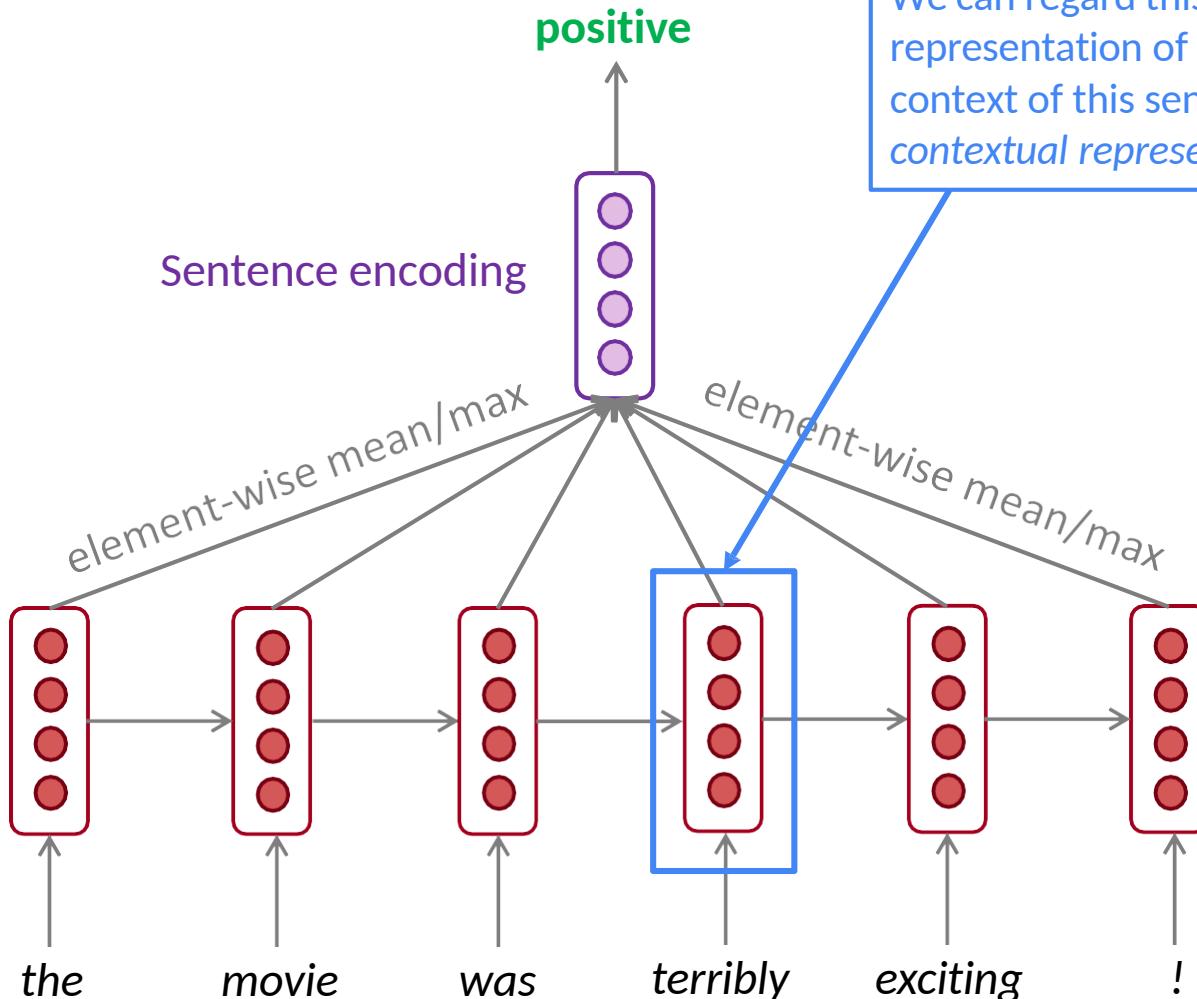


Figure 1: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

Bidirectional RNNs: motivation

Task: Sentiment Classification



We can regard this hidden state as a representation of the word “*terribly*” in the context of this sentence. We call this a *contextual representation*.

These contextual representations only contain information about the *left context* (e.g. “*the movie was*”).

What about *right context*?

In this example, “*exciting*” is in the right context and this modifies the meaning of “*terribly*” (from negative to positive)

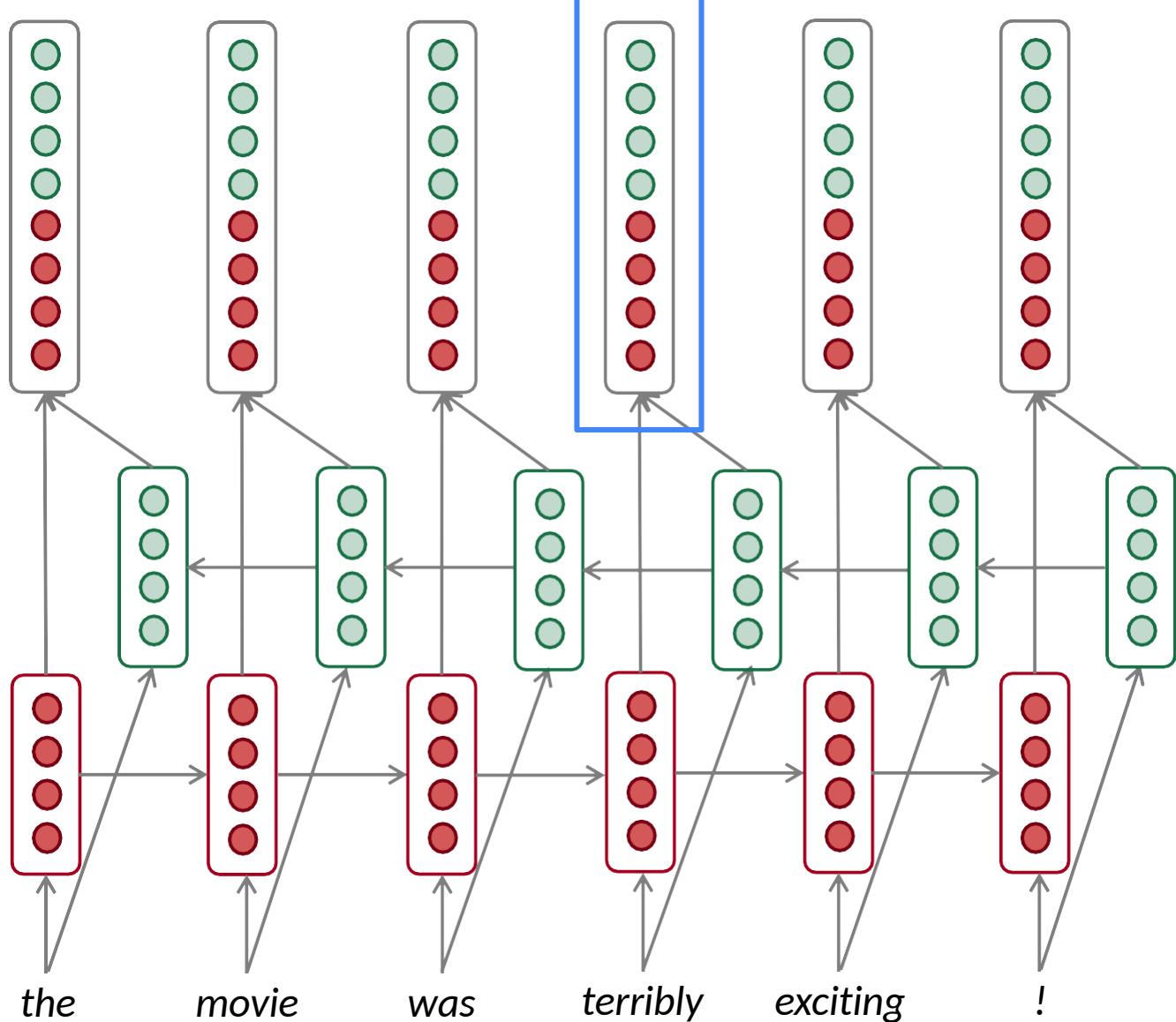
Bidirectional RNNs

This contextual representation of “terribly” has both left and right context!

Concatenated hidden states

Backward RNN

Forward RNN



Bidirectional RNNs

On timestep t :

This is a general notation to mean “compute one forward step of the RNN” – it could be a vanilla, LSTM or GRU computation.

Forward RNN $\vec{h}^{(t)} = \text{RNN}_{\text{FW}}(\vec{h}^{(t-1)}, \mathbf{x}^{(t)})$

Backward RNN $\overleftarrow{h}^{(t)} = \text{RNN}_{\text{BW}}(\overleftarrow{h}^{(t+1)}, \mathbf{x}^{(t)})$

Concatenated hidden states

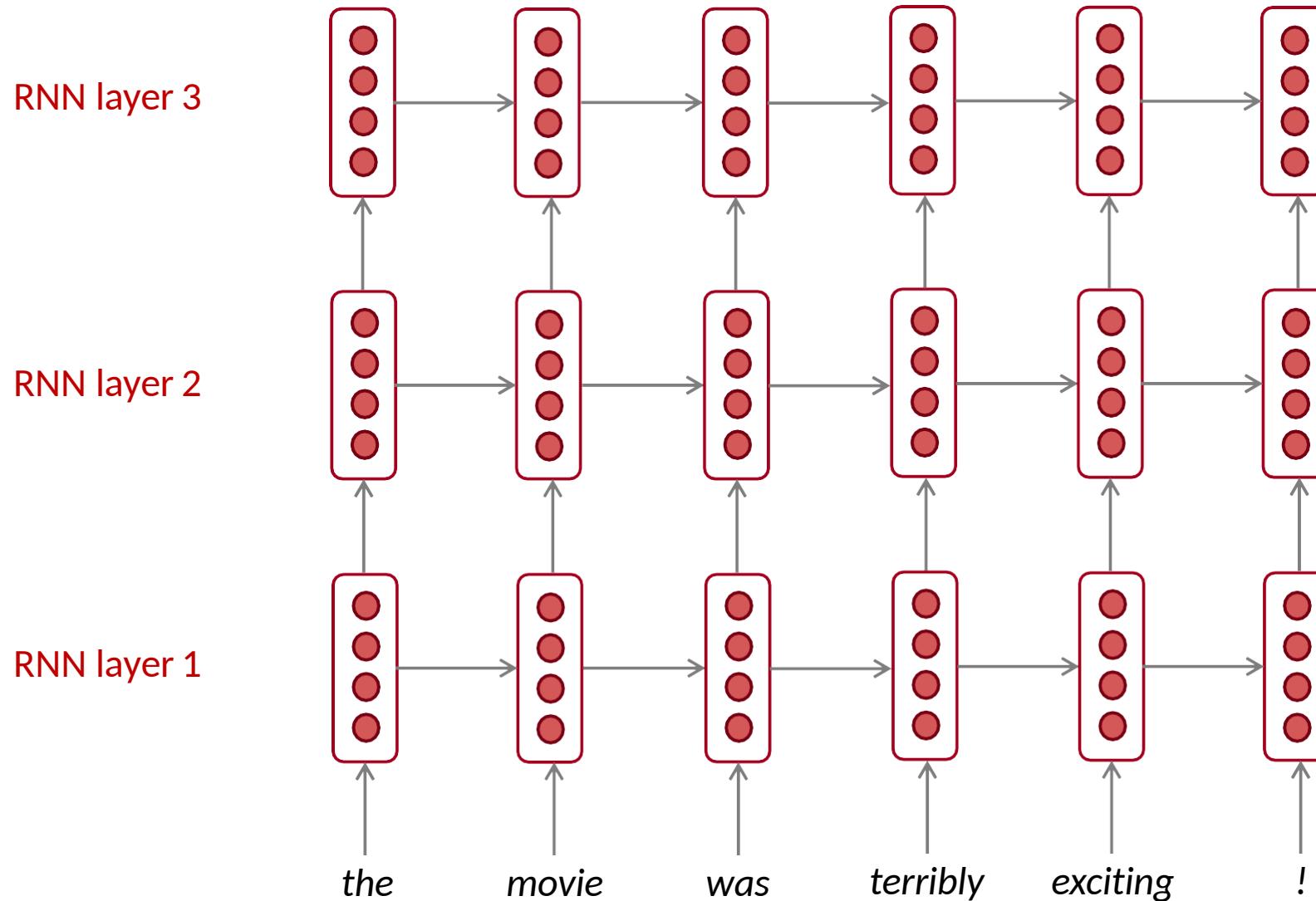
$$\mathbf{h}^{(t)} = [\vec{h}^{(t)}; \overleftarrow{h}^{(t)}]$$

Generally, these two RNNs have separate weights

We regard this as “the hidden state” of a bidirectional RNN. This is what we pass on to the next parts of the network.

Multi-layer RNNs

The hidden states from RNN layer i are the inputs to RNN layer $i+1$



Evaluating Language Models

- The standard **evaluation metric** for Language Models is **perplexity**.

$$\text{perplexity} = \prod_{t=1}^T \left(\frac{1}{P_{\text{LM}}(\mathbf{x}^{(t+1)} | \mathbf{x}^{(t)}, \dots, \mathbf{x}^{(1)})} \right)^{1/T}$$

Normalized by
number of words

Inverse probability of corpus, according to Language Model

- This is equal to the exponential of the cross-entropy loss $J(\theta)$:

$$= \prod_{t=1}^T \left(\frac{1}{\hat{\mathbf{y}}_{\mathbf{x}_{t+1}}^{(t)}} \right)^{1/T} = \exp \left(\frac{1}{T} \sum_{t=1}^T -\log \hat{\mathbf{y}}_{\mathbf{x}_{t+1}}^{(t)} \right) = \exp(J(\theta))$$

Lower perplexity is better!

Recap thus far

- **Language Model**: A system that predicts the next word
- **Recurrent Neural Network**: A family of neural networks that:
 - Take sequential input of any length
 - Apply the same weights on each step
 - Can optionally produce output on each step
- **Vanishing gradient problem**: what it is, why it happens, and why it's bad for RNNs
- **LSTMs and GRUs**: more complicated RNNs that use gates to control information flow; more resilient to vanishing gradients

Plan for this lecture

- Recurrent neural networks
 - Basics
 - Training (backprop through time, vanishing gradient)
 - Recurrent networks with gates (GRU, LSTM)
- Applications in NLP and vision
 - Neural machine translation (beam search, attention)
 - Image/video captioning

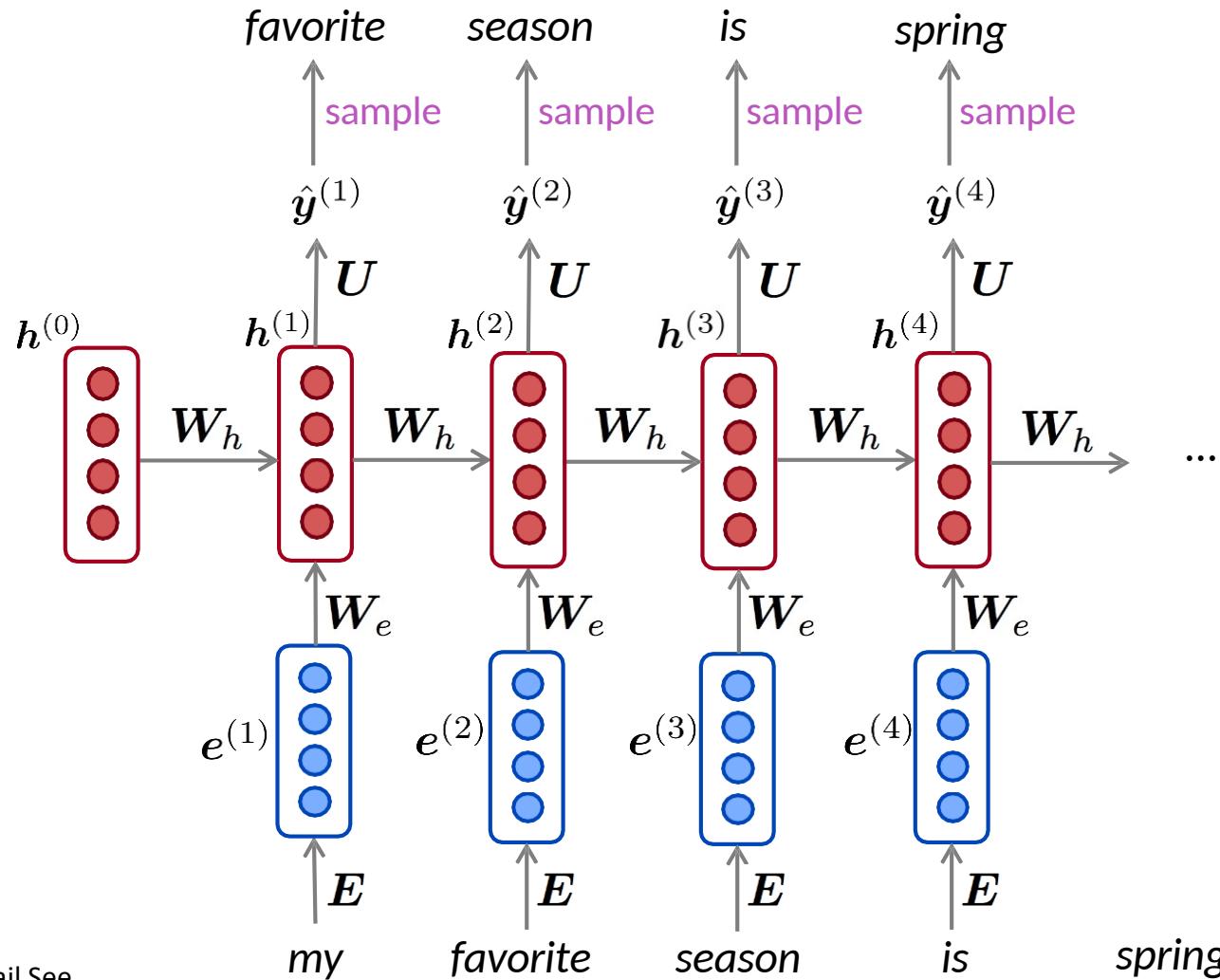
Applications

Why should we care about Language Modeling?

- Language Modeling is a **benchmark task** that helps us measure our progress on understanding language
- Language Modeling is a **subcomponent** of many NLP tasks, especially those involving **generating text** or **estimating the probability of text**:
 - Predictive typing
 - Speech recognition
 - Handwriting recognition
 - Spelling/grammar correction
 - Authorship identification
 - Machine translation
 - Summarization
 - Dialogue
 - etc.

Generating text with a RNN Language Model

You can use a RNN Language Model to **generate text** by **repeated sampling**.
Sampled output is next step's input.



Generating text with a RNN Language Model

- Let's have some fun!
- You can train a RNN-LM on any kind of text, then generate text in that style.
- RNN-LM trained on Obama speeches:



The United States will step up to the cost of a new challenges of the American people that will share the fact that we created the problem. They were attacked and so that they have to say that all the task of the final days of war that I will not be able to get this done.

Source: <https://medium.com/@samim/obama-rnn-machine-generated-political-speeches-c8abd18a2ea0>

Generating text with a RNN Language Model

- Let's have some fun!
- You can train a RNN-LM on any kind of text, then generate text in that style.
- RNN-LM trained on *Harry Potter*:



“Sorry,” Harry shouted, panicking—“I’ll leave those brooms in London, are they?”

“No idea,” said Nearly Headless Nick, casting low close by Cedric, carrying the last bit of treacle Charms, from Harry’s shoulder, and to answer him the common room perched upon it, four arms held a shining knob from when the spider hadn’t felt it seemed. He reached the teams too.

Source: <https://medium.com/deep-writing/harry-potter-written-by-artificial-intelligence-8a9431803da6>

Generating text with a RNN Language Model

- Let's have some fun!
- You can train a RNN-LM on any kind of text, then generate text in that style.
- RNN-LM trained on paint color names:

Ghasty Pink	231	137	165
Power Gray	151	124	112
Navel Tan	199	173	140
Bock Coe White	221	215	236
Horble Gray	178	181	196
Homestar Brown	133	104	85
Snader Brown	144	106	74
Golder Craam	237	217	177
Hurky White	232	223	215
Burf Pink	223	173	179
Rose Hork	230	215	198

Sand Dan	201	172	143
Grade Bat	48	94	83
Light Of Blast	175	150	147
Grass Bat	176	99	108
Sindis Poop	204	205	194
Dope	219	209	179
Testing	156	101	106
Stoner Blue	152	165	159
Burble Simp	226	181	132
Stanky Bean	197	162	171
Turdly	190	164	116

This is an example of a character-level RNN-LM (predicts what character comes next)

Source: <http://aiweirdness.com/post/160776374467/new-paint-colors-invented-by-neural-network>

Generating poetry with RNNs

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.

Generating poetry with RNNs

at first:

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

↓ train more

```
"Tmont thithey" fomescerliund  
Keushey. Thom here  
sheulke, anmerenith ol sivh I lalterthend Bleipile shuwyl 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.
```

More info: <http://karpathy.github.io/2015/05/21/rnn-effectiveness/>

Generating poetry with RNNs

PANDARUS:

Alas, I think he shall be come approached and the day
When little strain 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.

Generating textbooks with RNNs

For $\bigoplus_{n=1,\dots,m} \mathcal{L}_{m_n} = 0$, hence we can find a closed subset \mathcal{H} in \mathcal{H} and any sets \mathcal{F} on X , U is a closed immersion of S , then $U \rightarrow T$ is a separated algebraic space.

Proof. Proof of (1). It also start we get

$$S = \text{Spec}(R) = U \times_X U \times_X U$$

and the comparicoly in the fibre product covering we have to prove the lemma generated by $\coprod Z \times_U U \rightarrow V$. Consider the maps M along the set of points Sch_{fppf} and $U \rightarrow U$ is the fibre category of S in U in Section, ?? and the fact that any U affine, see Morphisms, Lemma ???. Hence we obtain a scheme S and any open subset $W \subset U$ in $\text{Sh}(G)$ such that $\text{Spec}(R') \rightarrow S$ is smooth or an

$$U = \bigcup U_i \times_{S_i} U_i$$

which has a nonzero morphism we may assume that f_i is of finite presentation over S . We claim that $\mathcal{O}_{X,x}$ is a scheme where $x, x', s'' \in S'$ such that $\mathcal{O}_{X,x'} \rightarrow \mathcal{O}'_{X',x'}$ is separated. By Algebra, Lemma ?? we can define a map of complexes $\text{GL}_{S'}(x'/S'')$ and we win. \square

To prove study we see that $\mathcal{F}|_U$ is a covering of \mathcal{X}' , and \mathcal{T}_i is an object of $\mathcal{F}_{X/S}$ for $i > 0$ and \mathcal{F}_p exists and let \mathcal{F}_i be a presheaf of \mathcal{O}_X -modules on \mathcal{C} as a \mathcal{F} -module. In particular $\mathcal{F} = U/\mathcal{F}$ we have to show that

$$\widetilde{M}^\bullet = \mathcal{I}^\bullet \otimes_{\text{Spec}(k)} \mathcal{O}_{S,s} - i_X^{-1} \mathcal{F}$$

is a unique morphism of algebraic stacks. Note that

$$\text{Arrows} = (\text{Sch}/S)^{opp}_{fppf}, (\text{Sch}/S)_{fppf}$$

and

$$V = \Gamma(S, \mathcal{O}) \longrightarrow (U, \text{Spec}(A))$$

is an open subset of X . Thus U is affine. This is a continuous map of X is the inverse, the groupoid scheme S .

Proof. See discussion of sheaves of sets. \square

The result for prove any open covering follows from the less of Example ???. It may replace S by $X_{\text{spaces},\text{étale}}$ which gives an open subspace of X and T equal to S_{Zar} , see Descent, Lemma ???. Namely, by Lemma ?? we see that R is geometrically regular over S .

Lemma 0.1. Assume (3) and (3) by the construction in the description.

Suppose $X = \lim |X|$ (by the formal open covering X and a single map $\underline{\text{Proj}}_X(\mathcal{A}) = \text{Spec}(B)$ over U compatible with the complex

$$\text{Set}(\mathcal{A}) = \Gamma(X, \mathcal{O}_{X,\mathcal{O}_X}).$$

When in this case of to show that $\mathcal{Q} \rightarrow \mathcal{C}_{Z/X}$ is stable under the following result in the second conditions of (1), and (3). This finishes the proof. By Definition ?? (without element is when the closed subschemes are catenary. If T is surjective we may assume that T is connected with residue fields of S . Moreover there exists a closed subspace $Z \subset X$ of X where U in X' is proper (some defining as a closed subset of the uniqueness it suffices to check the fact that the following theorem

(1) f is locally of finite type. Since $S = \text{Spec}(R)$ and $Y = \text{Spec}(R)$.

Proof. This is form all sheaves of sheaves on X . But given a scheme U and a surjective étale morphism $U \rightarrow X$. Let $U \cap U = \coprod_{i=1,\dots,n} U_i$ be the scheme X over S at the schemes $X_i \rightarrow X$ and $U = \lim_i X_i$. \square

The following lemma surjective restrocomposes of this implies that $\mathcal{F}_{x_0} = \mathcal{F}_{x_0} = \mathcal{F}_{x,\dots,x}$.

Lemma 0.2. Let X be a locally Noetherian scheme over S , $E = \mathcal{F}_{X/S}$. Set $\mathcal{I} = \mathcal{J}_1 \subset \mathcal{I}'_n$. Since $\mathcal{I}'_n \subset \mathcal{I}^n$ are nonzero over $i_0 \leq p$ is a subset of $\mathcal{J}_{n,0} \circ \overline{A}_2$ works.

Lemma 0.3. In Situation ???. Hence we may assume $q' = 0$.

Proof. We will use the property we see that p is the next functor (??). On the other hand, by Lemma ?? we see that

$$D(\mathcal{O}_{X'}) = \mathcal{O}_X(D)$$

where K is an F -algebra where δ_{n+1} is a scheme over S . \square

Generating code with RNNs

```
static void do_command(struct seq_file *m, void *v)
{
    int column = 32 << (cmd[2] & 0x80);
    if (state)
        cmd = (int)(int_state ^ (in_8(&ch->ch_flags) & Cmd) ? 2 : 1);
    else
        seq = 1;
    for (i = 0; i < 16; i++) {
        if (k & (1 << i))
            pipe = (in_use & UMXTHREAD_UNCCA) +
                ((count & 0x00000000fffffff8) & 0x000000f) << 8;
        if (count == 0)
            sub(pid, ppc_md.kexec_handle, 0x20000000);
        pipe_set_bytes(i, 0);
    }
    /* Free our user pages pointer to place camera if all dash */
    subsystem_info = &of_changes[PAGE_SIZE];
    rek_controls(offset, idx, &soffset);
    /* Now we want to deliberately put it to device */
    control_check_polarity(&context, val, 0);
    for (i = 0; i < COUNTER; i++)
        seq_puts(s, "policy ");
}
```

Generated
C code

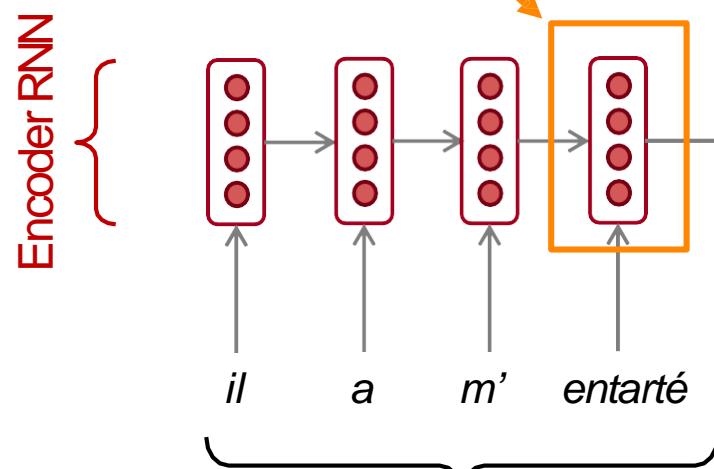
Neural Machine Translation

- Neural Machine Translation (NMT) is a way to do Machine Translation with a *single neural network*
- The neural network architecture is called sequence-to-sequence (aka seq2seq) and it involves *two RNNs*.

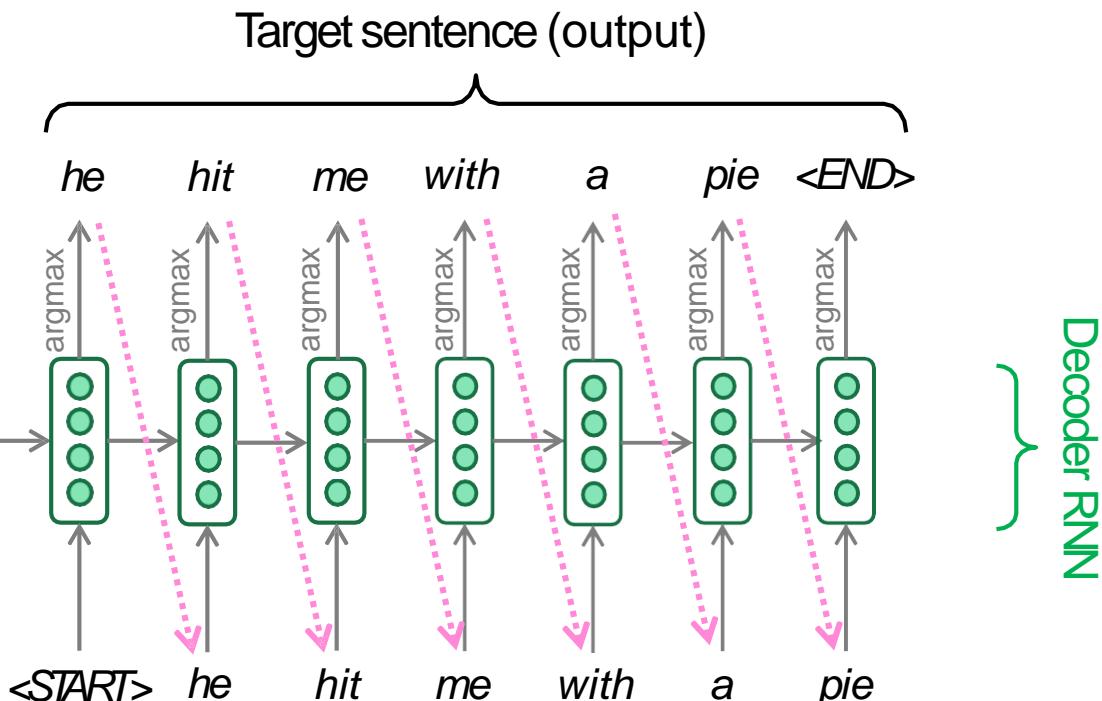
Neural Machine Translation (NMT)

The sequence-to-sequence model

Encoding of the source sentence.
Provides initial hidden state
for Decoder RNN.



Encoder RNN produces
an encoding of the
source sentence.

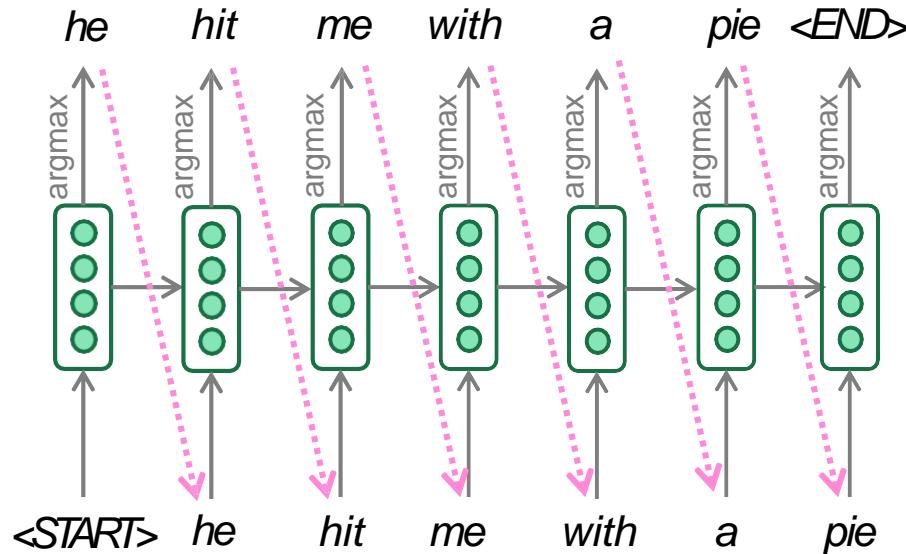


Decoder RNN is a Language Model that generates target sentence, conditioned on encoding.

Note: This diagram shows test time behavior:
decoder output is fed in → as next step's input

Greedy decoding

- We saw how to generate (or “decode”) the target sentence by taking argmax on each step of the decoder



- This is **greedy decoding** (take most probable word on each step)
- Problems with this method?**

Problems with greedy decoding

- Greedy decoding has no way to undo decisions!
 - Input: *il a m'entarté* (*he hit me with a pie*)
 - → *he* __
 - → *he hit* __
 - → *he hit a* __ (whoops! no going back now...)
- How to fix this?

Exhaustive search decoding

- Ideally we want to find a (length T) translation y that maximizes

$$\begin{aligned} P(y|x) &= P(y_1|x) P(y_2|y_1, x) P(y_3|y_1, y_2, x) \dots, P(y_T|y_1, \dots, y_{T-1}, x) \\ &= \prod_{t=1}^T P(y_t|y_1, \dots, y_{t-1}, x) \end{aligned}$$

- We could try computing **all possible sequences** y
 - This means that on each step t of the decoder, we're tracking V^T possible partial translations, where V is vocabulary size
 - This $O(V^T)$ complexity is **far too expensive!**

Beam search decoding

- Core idea: On each step of decoder, keep track of the k most probable partial translations (*hypotheses*)
 - k is the beam size (in practice around 5 to 10)
- A hypothesis y_1, \dots, y_t has a **score** which is its log probability:
$$\text{score}(y_1, \dots, y_t) = \log P_{\text{LM}}(y_1, \dots, y_t | x) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$$
 - Scores are all negative, and higher score is better
 - We search for high-scoring hypotheses, tracking top k on each step
- Beam search is **not guaranteed** to find optimal solution
- But **much more efficient** than exhaustive search!

Beam search decoding: example

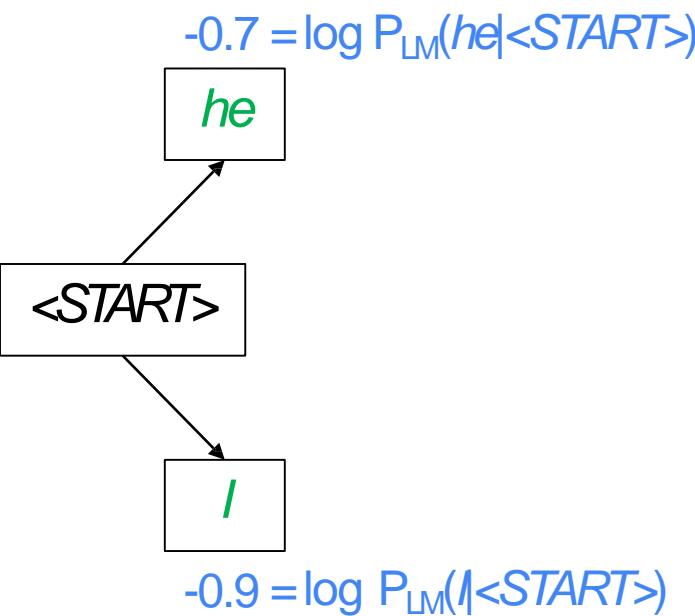
Beam size = $k = 2$. Blue numbers = $\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$

<START>

Calculate prob
dist of next word

Beam search decoding: example

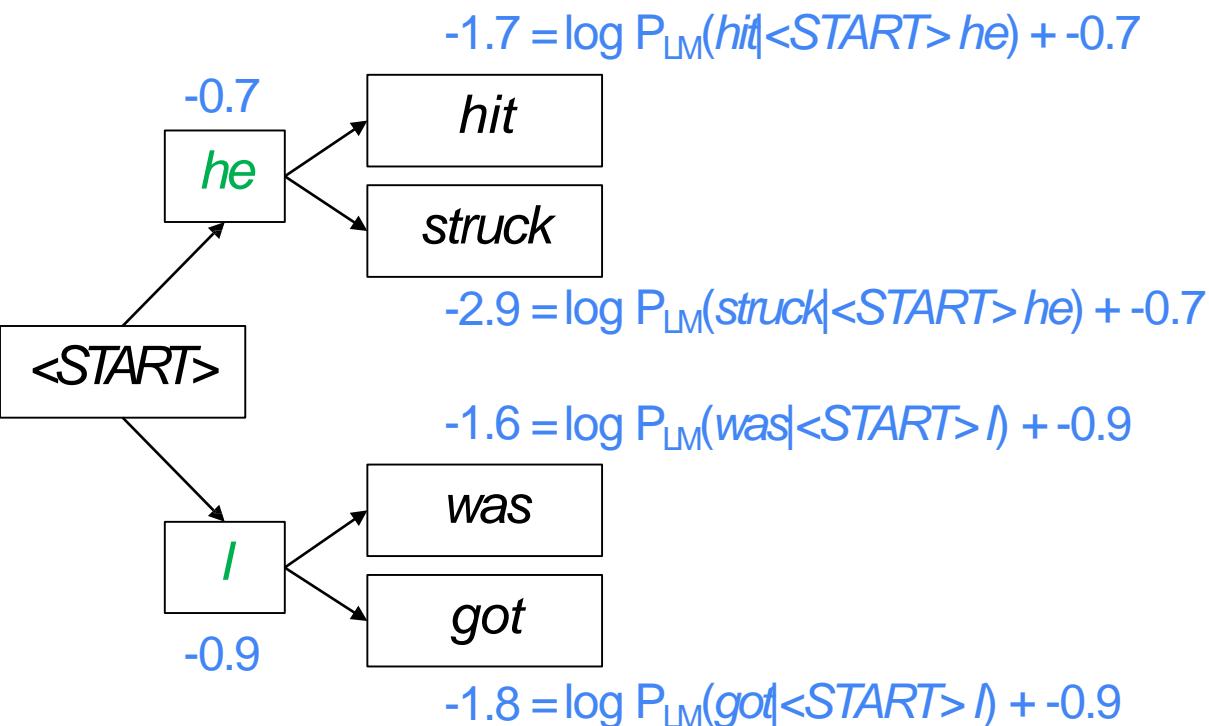
Beam size = $k = 2$. Blue numbers = $\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$



Take top k words
and compute scores

Beam search decoding: example

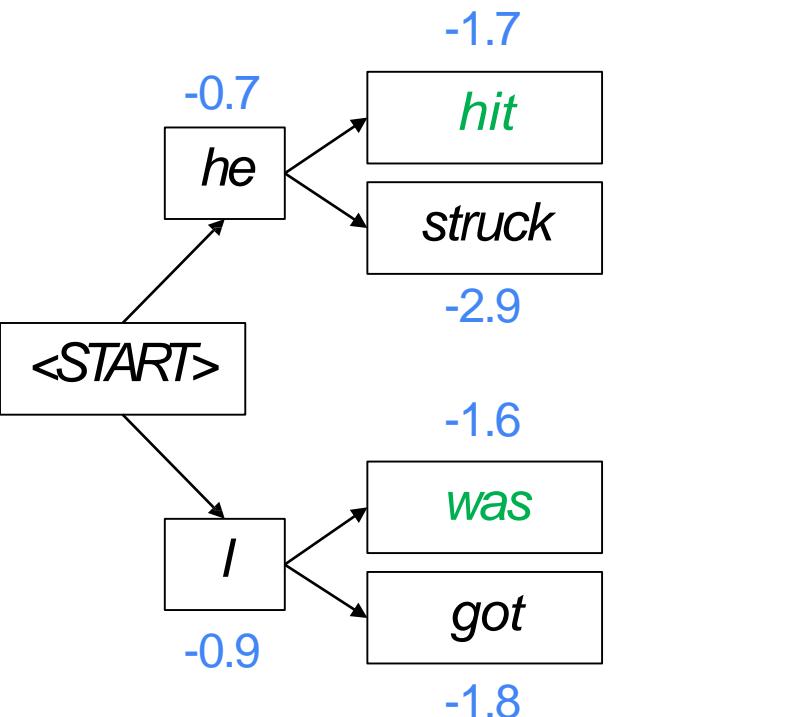
Beam size = $k = 2$. Blue numbers = $\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$



For each of the k hypotheses, find top k next words and calculate scores

Beam search decoding: example

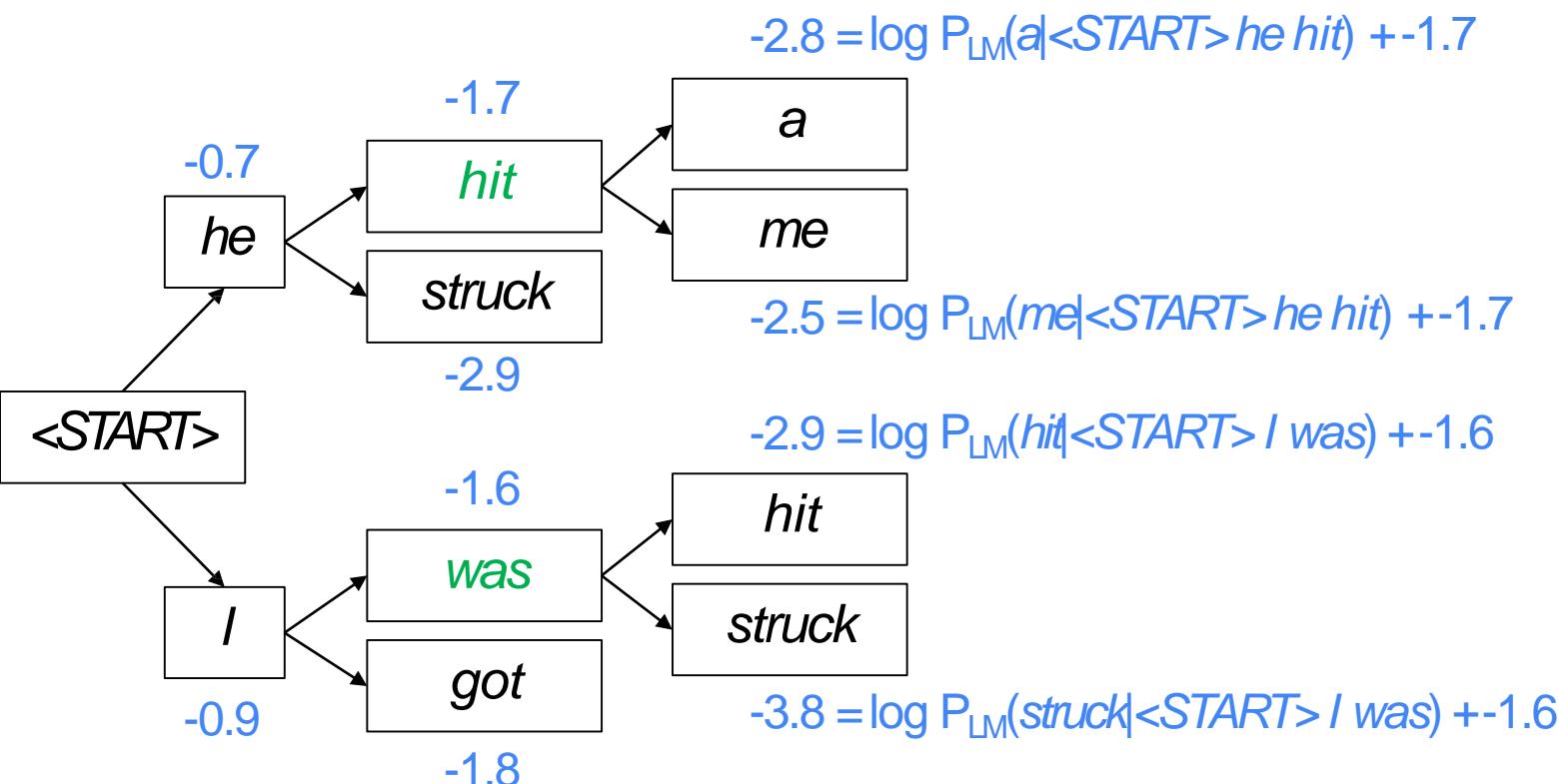
Beam size = $k = 2$. Blue numbers = $\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$



Of these k^2 hypotheses,
just keep k with highest scores

Beam search decoding: example

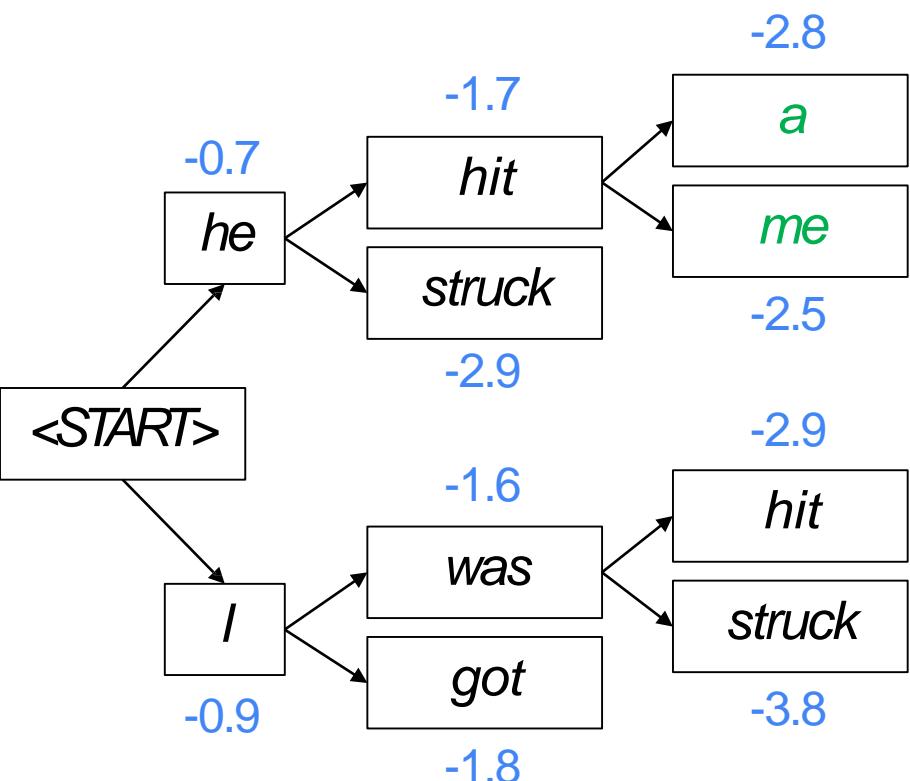
Beam size = $k = 2$. Blue numbers = $\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$



For each of the k hypotheses, find top k next words and calculate scores

Beam search decoding: example

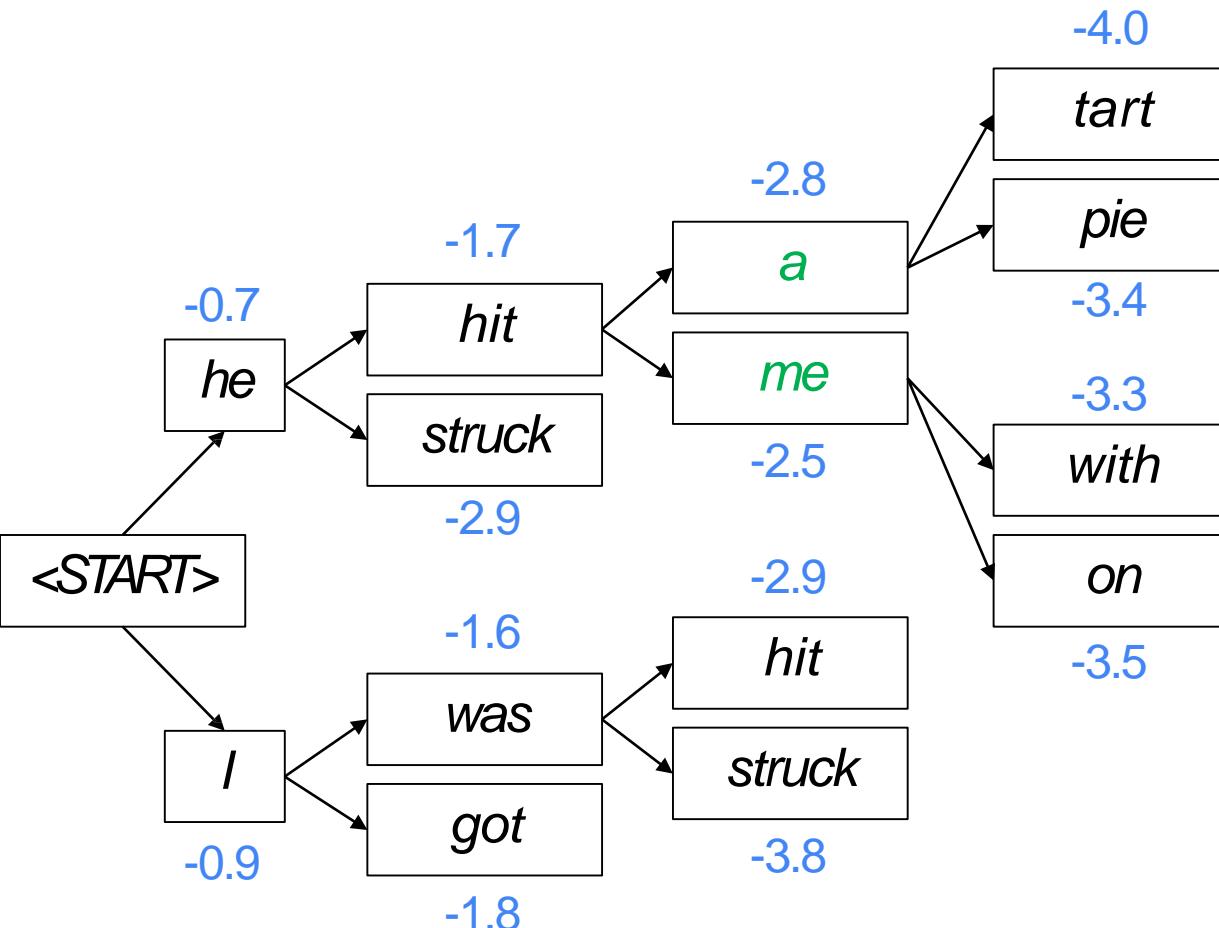
Beam size = $k = 2$. Blue numbers = $\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$



Of these k^2 hypotheses,
just keep k with highest scores

Beam search decoding: example

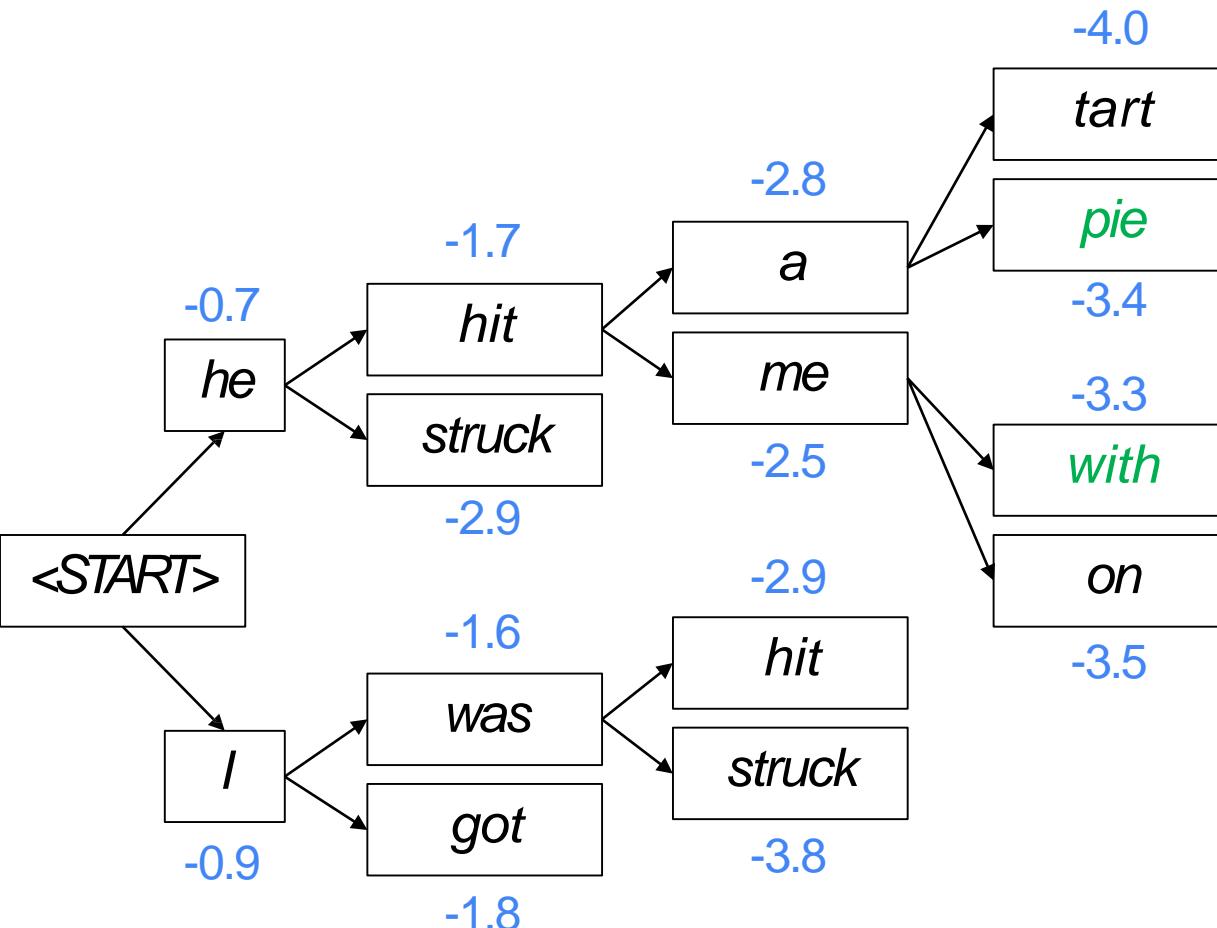
Beam size = $k = 2$. Blue numbers = $\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$



For each of the k hypotheses, find top k next words and calculate scores

Beam search decoding: example

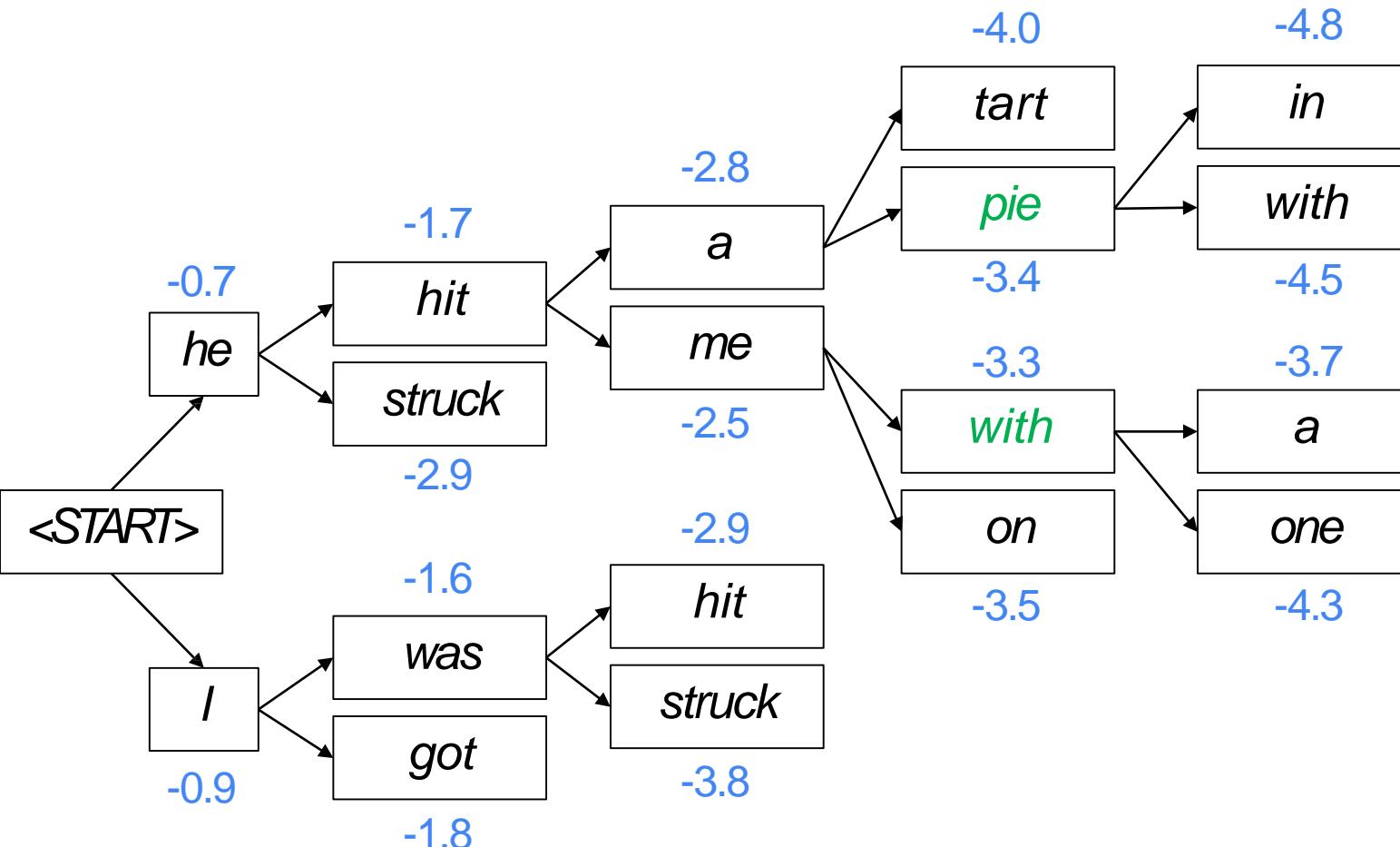
Beam size = $k = 2$. Blue numbers = $\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$



Of these k^2 hypotheses,
just keep k with highest scores

Beam search decoding: example

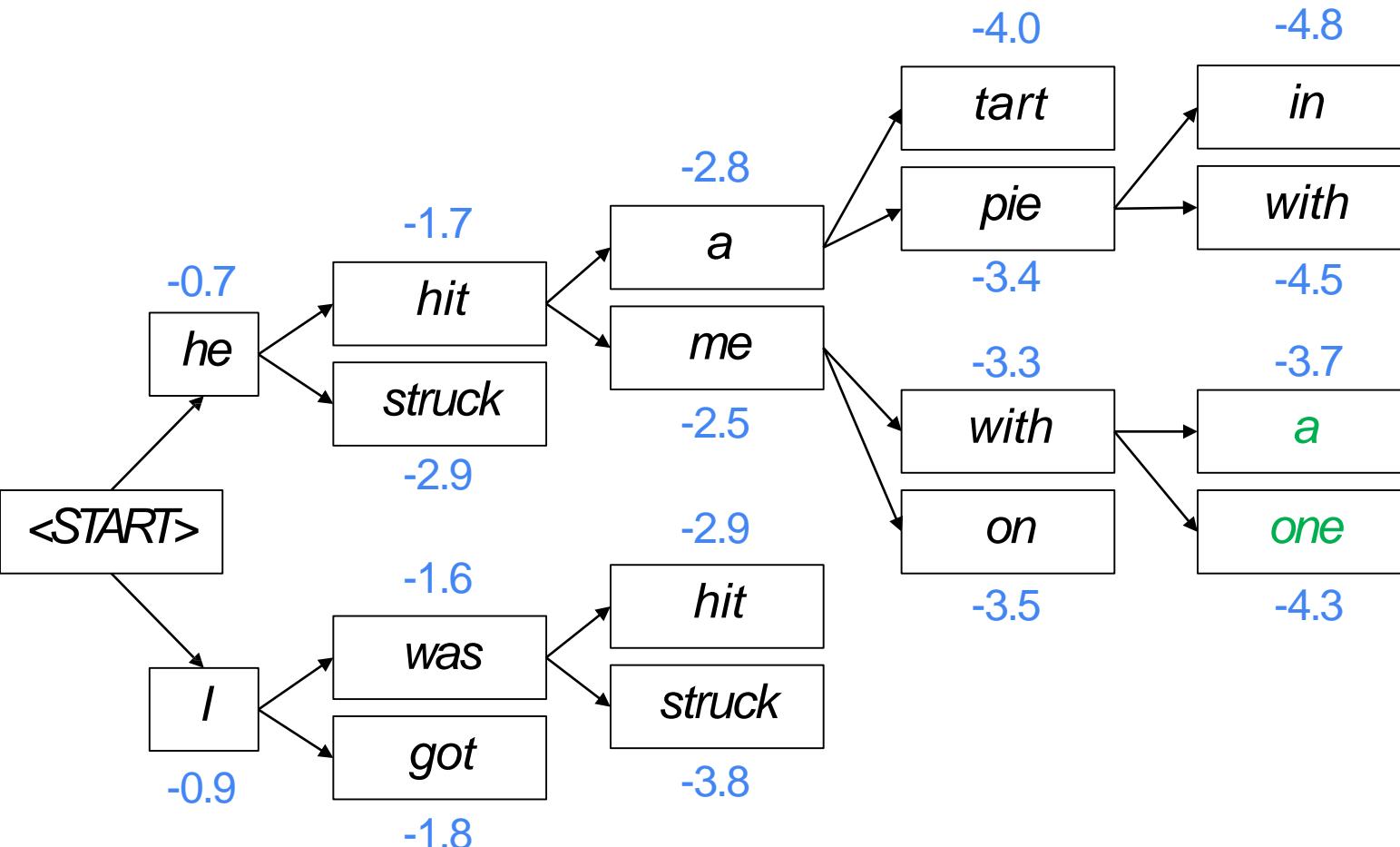
Beam size = $k = 2$. Blue numbers = $\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$



For each of the k hypotheses, find top k next words and calculate scores

Beam search decoding: example

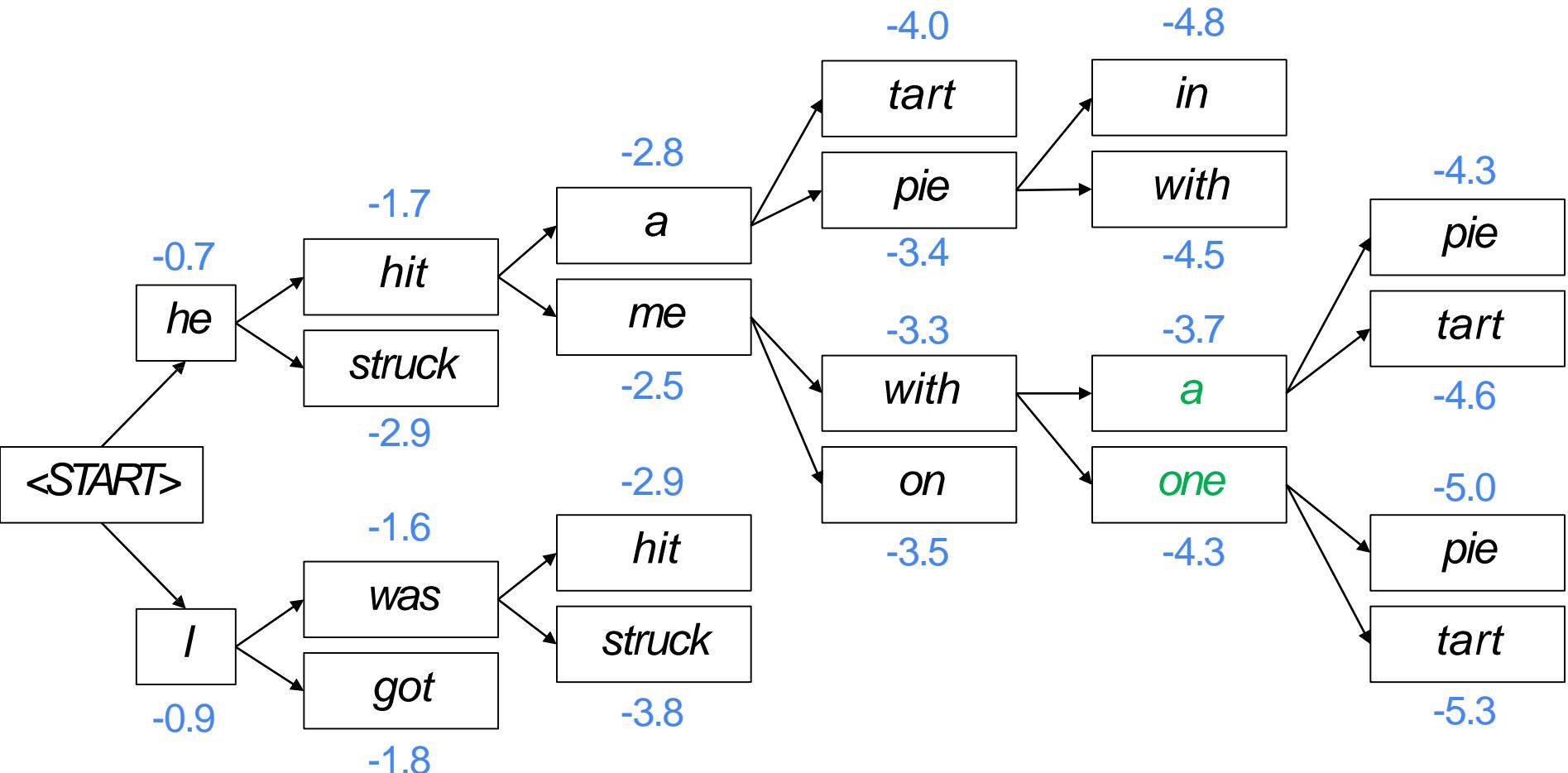
Beam size = $k = 2$. Blue numbers = $\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$



Of these k^2 hypotheses,
just keep k with highest scores

Beam search decoding: example

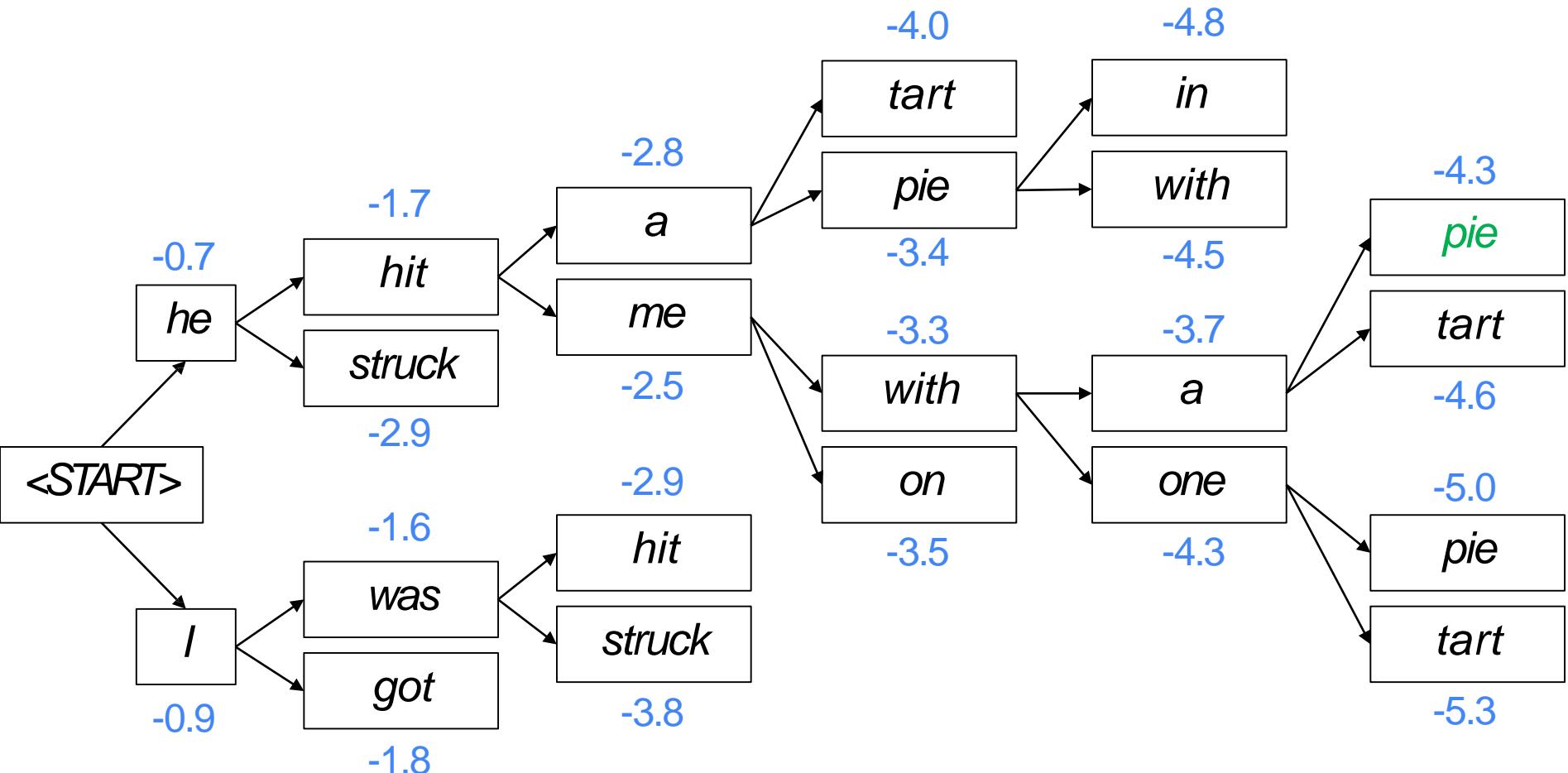
Beam size = $k = 2$. Blue numbers = $\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$



For each of the k hypotheses, find top k next words and calculate scores

Beam search decoding: example

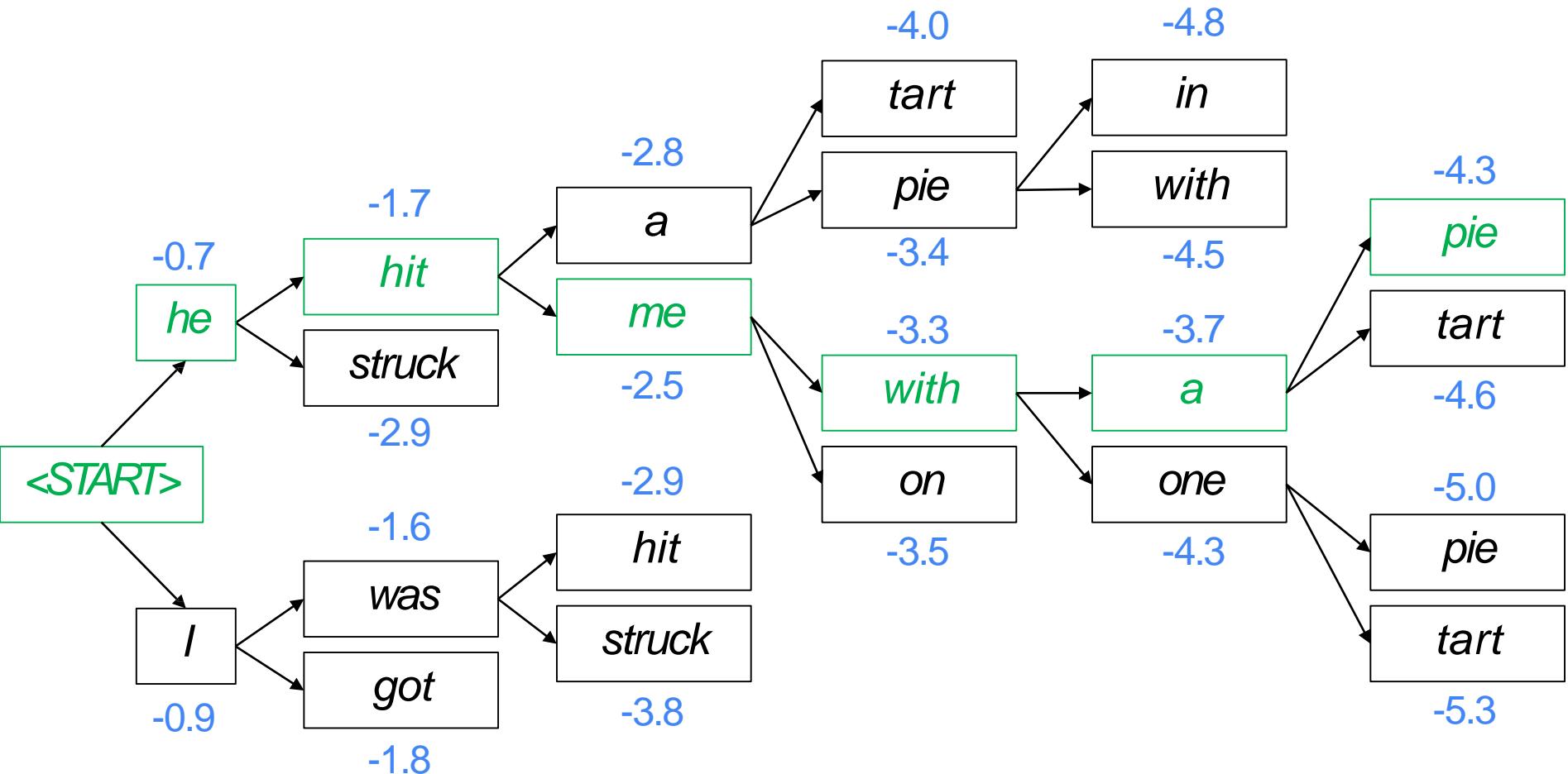
Beam size = $k = 2$. Blue numbers = $\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$



This is the top-scoring hypothesis!

Beam search decoding: example

Beam size = $k = 2$. Blue numbers = $\text{score}(y_1, \dots, y_t) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$



Backtrack to obtain the full hypothesis

Beam search decoding: stopping criterion

- In greedy decoding, usually we decode until the model produces an <END> token
 - For example: <START> he hit me with a pie <END>
- In beam search decoding, different hypotheses may produce <END> tokens on different timesteps
 - When a hypothesis produces <END>, that hypothesis is complete.
 - Place it aside and continue exploring other hypotheses via beam search.
- Usually we continue beam search until:
 - We reach timestep T (where T is some pre-defined cutoff), or
 - We have at least n completed hypotheses (where n is pre-defined cutoff)

Beam search decoding: finishing up

- We have our list of completed hypotheses.
- How to select top one with highest score?
- Each hypothesis y_1, \dots, y_t on our list has a score

$$\text{score}(y_1, \dots, y_t) = \log P_{\text{LM}}(y_1, \dots, y_t | x) = \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$$

- Problem with this: longer hypotheses have lower scores
- Fix: Normalize by length. Use this to select top one instead:

$$\frac{1}{t} \sum_{i=1}^t \log P_{\text{LM}}(y_i | y_1, \dots, y_{i-1}, x)$$

How do we evaluate Machine Translation?

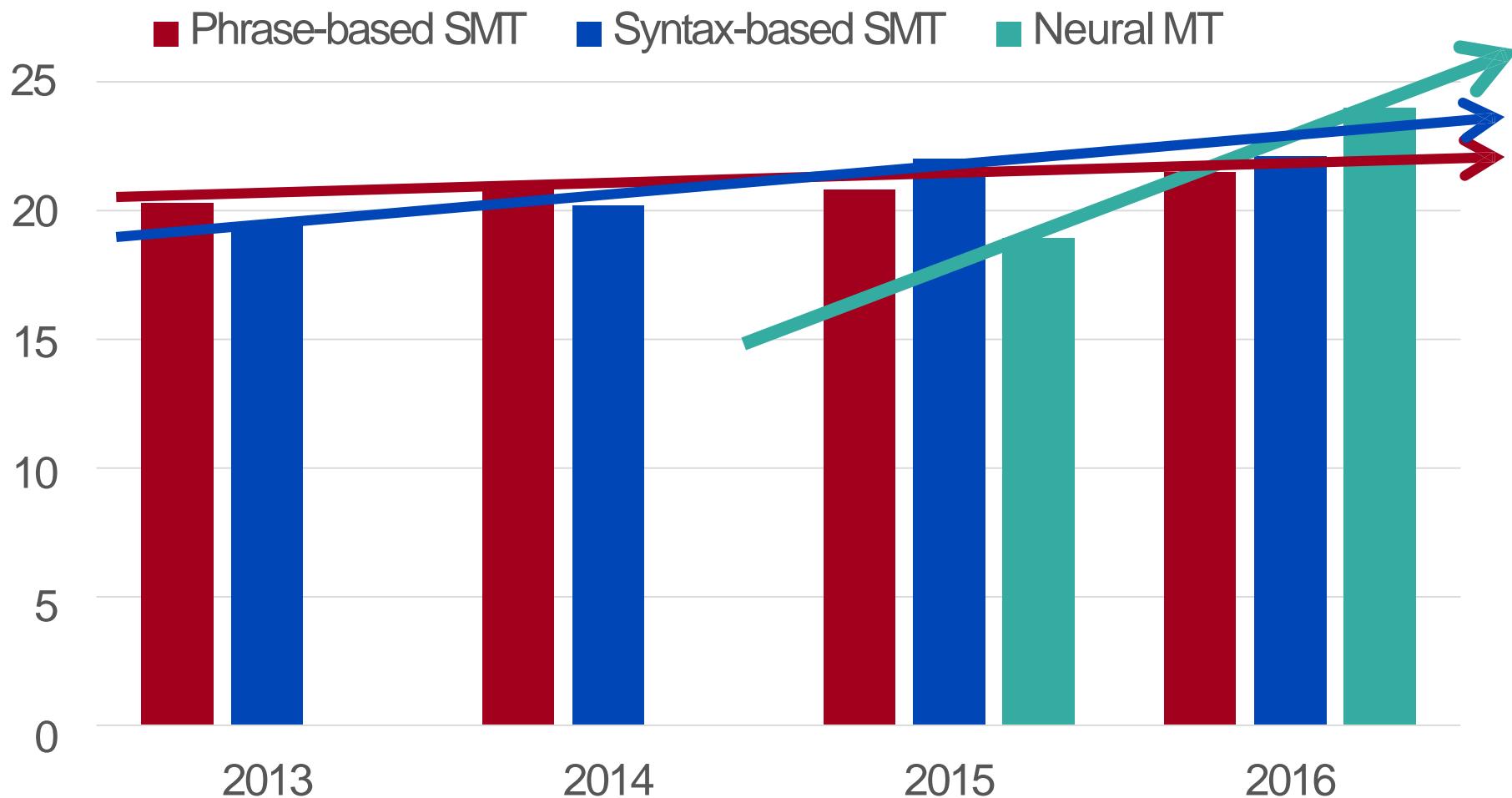
BLEU (Bilingual Evaluation Understudy)

- BLEU compares the machine-written translation to one or several human-written translation(s), and computes a **similarity score** based on:
 - *n*-gram precision (usually for 1, 2, 3 and 4-grams)
 - Plus a penalty for too-short system translations
- BLEU is **useful** but **imperfect**
 - There are many valid ways to translate a sentence
 - So a **good** translation can get a **poor** BLEU score because it has low *n*-gram overlap with the human translation ☹

Source: "BLEU: a Method for Automatic Evaluation of Machine Translation", Papineni et al, 2002.

MT progress overtime

[Edinburgh En-De WMT newstest2013 Cased BLEU; NMT 2015 from U. Montréal]



Source: http://www.meta-net.eu/events/meta-forum-2016/slides/09_sennrich.pdf

NMT: the biggest success story of NLP Deep Learning

Neural Machine Translation went from a fringe research activity in 2014 to the leading standard method in 2016

- 2014: First seq2seq paper published
- 2016: Google Translate switches from SMT to NMT
- This is amazing!
 - SMT systems, built by hundreds of engineers over many years, outperformed by NMT systems trained by a handful of engineers in a few months

So is Machine Translation solved?

- **Nope!**
- Many difficulties remain:
 - Out-of-vocabulary words
 - Domain mismatch between train and test data
 - Maintaining context over longer text
 - Low-resource language pairs

Further reading: “Has AI surpassed humans at translation? Not even close!”
https://www.skynettoday.com/editorials/state_of_nmt

So is Machine Translation solved?

- Nope!
- Using common sense is still hard

The image shows a screenshot of the Google Translate web interface. On the left, under "English", the text "paper jam" is entered, with an "Edit" link below it. On the right, under "Spanish", the translation "Mermelada de papel" is displayed. The interface includes language selection dropdowns, microphone and speaker icons, and a copy/paste button. At the bottom, there are links for "Open in Google Translate" and "Feedback".



So is Machine Translation solved?

- Nope!
- NMT picks up **biases** in training data

Malay - detected ▾

English ▾

Dia bekerja sebagai jururawat.

Dia bekerja sebagai pengaturcara. Edit

She works as a nurse.

He works as a programmer.

Didn't specify gender

Source: <https://hackernoon.com/bias-sexist-or-this-is-the-way-it-should-be-ce1f7c8c683c>

Remember

Man is to Computer Programmer as Woman is to Homemaker?

Debiasing Word Embeddings

Tolga Bolukbasi¹, Kai-Wei Chang², James Zou², Venkatesh Saligrama^{1,2}, Adam Kalai²

¹Boston University, 8 Saint Mary's Street, Boston, MA

²Microsoft Research New England, 1 Memorial Drive, Cambridge, MA

tolgab@bu.edu, kw@kwchang.net, jamesyzou@gmail.com, srv@bu.edu, adam.kalai@microsoft.com

Abstract

NMT research continues

NMT is the **flagship** task for NLP Deep Learning

- NMT research has **pioneered** many of the recent **innovations** of NLP Deep Learning
- In 2019: NMT research continues to **thrive**
 - Researchers have found **many, many** improvements to the “vanilla” seq2seq NMT system
 - But **one improvement** is so integral that it is the new vanilla...

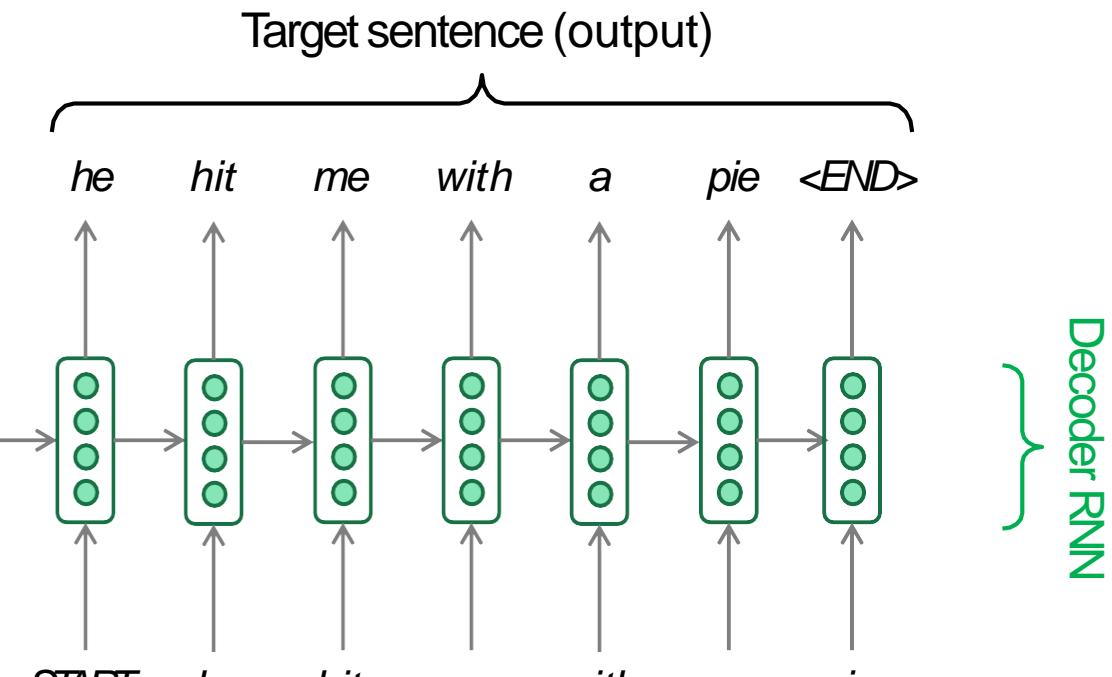
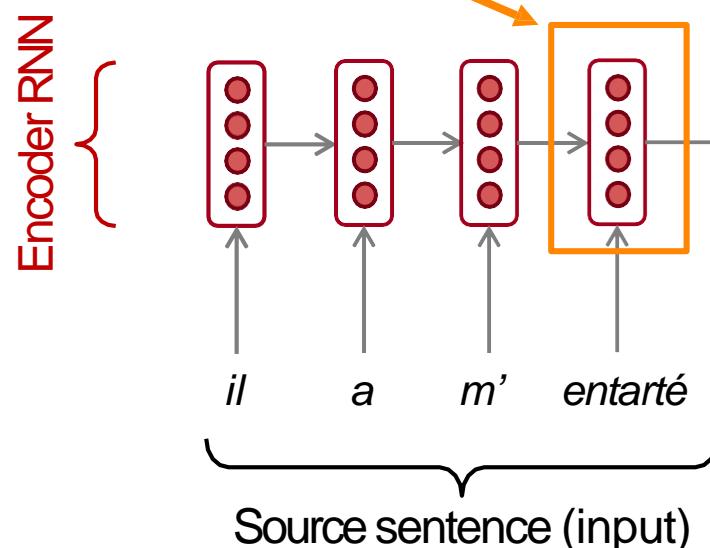
ATTENTION

Sequence-to-sequence: the bottleneck problem

Encoding of the source sentence.

This needs to capture *all information* about the source sentence.

Information bottleneck!



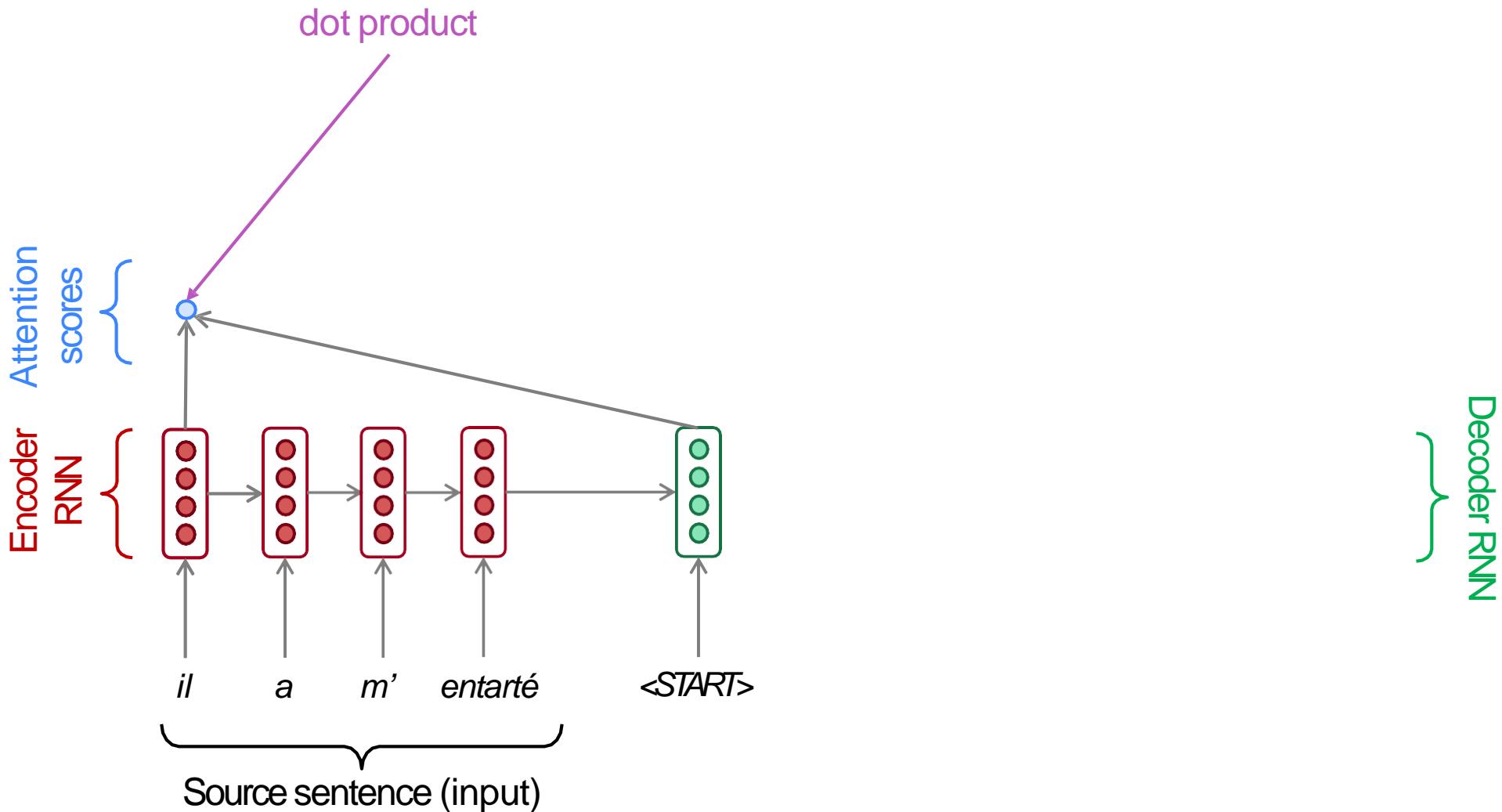
Attention

- Attention provides a solution to the bottleneck problem.
- Core idea: on each step of the decoder, use *direct connection to the encoder* to *focus on a particular part* of the source sequence

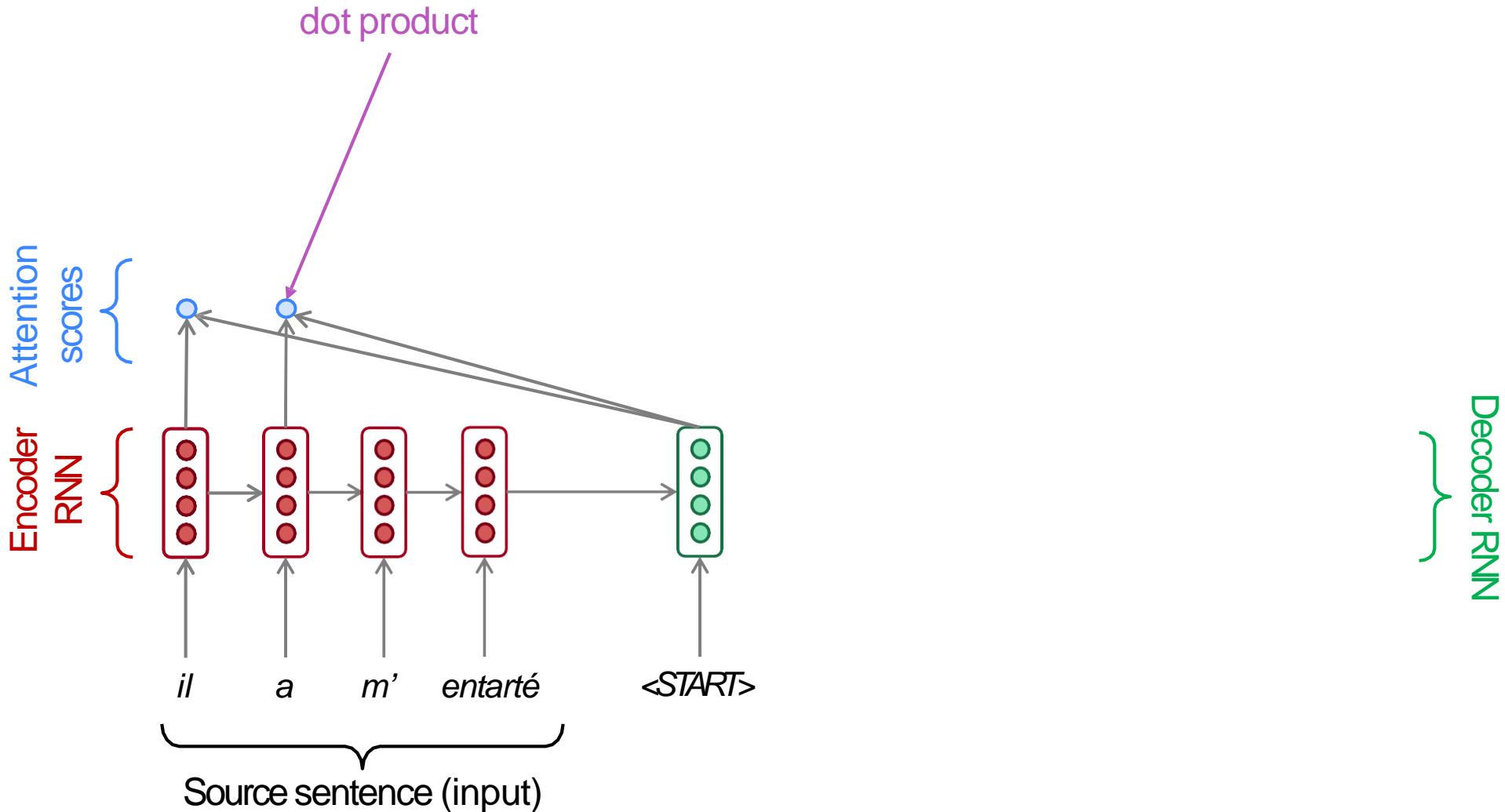


- First we will show via diagram (no equations), then we will show with equations

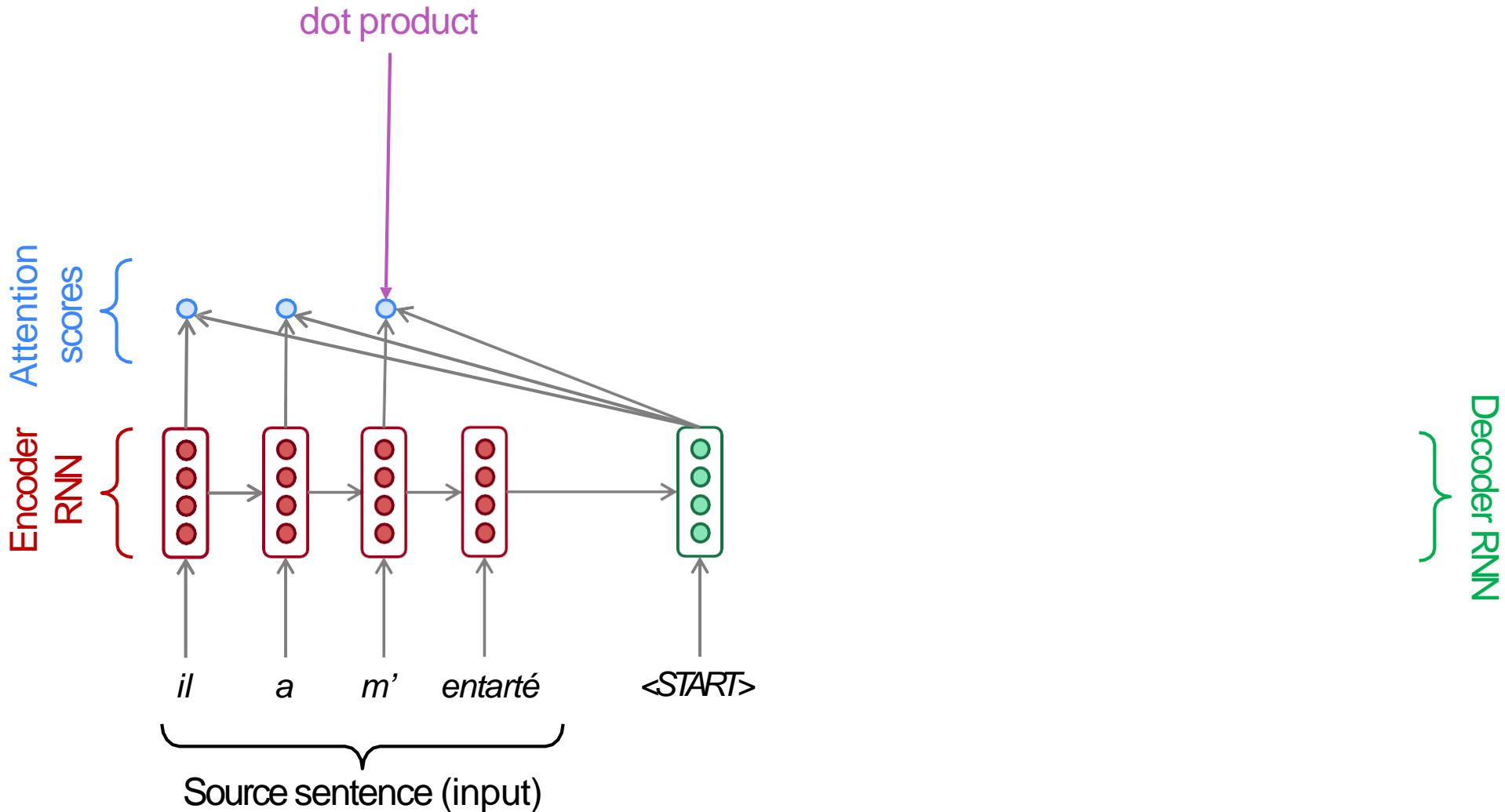
Sequence-to-sequence with attention



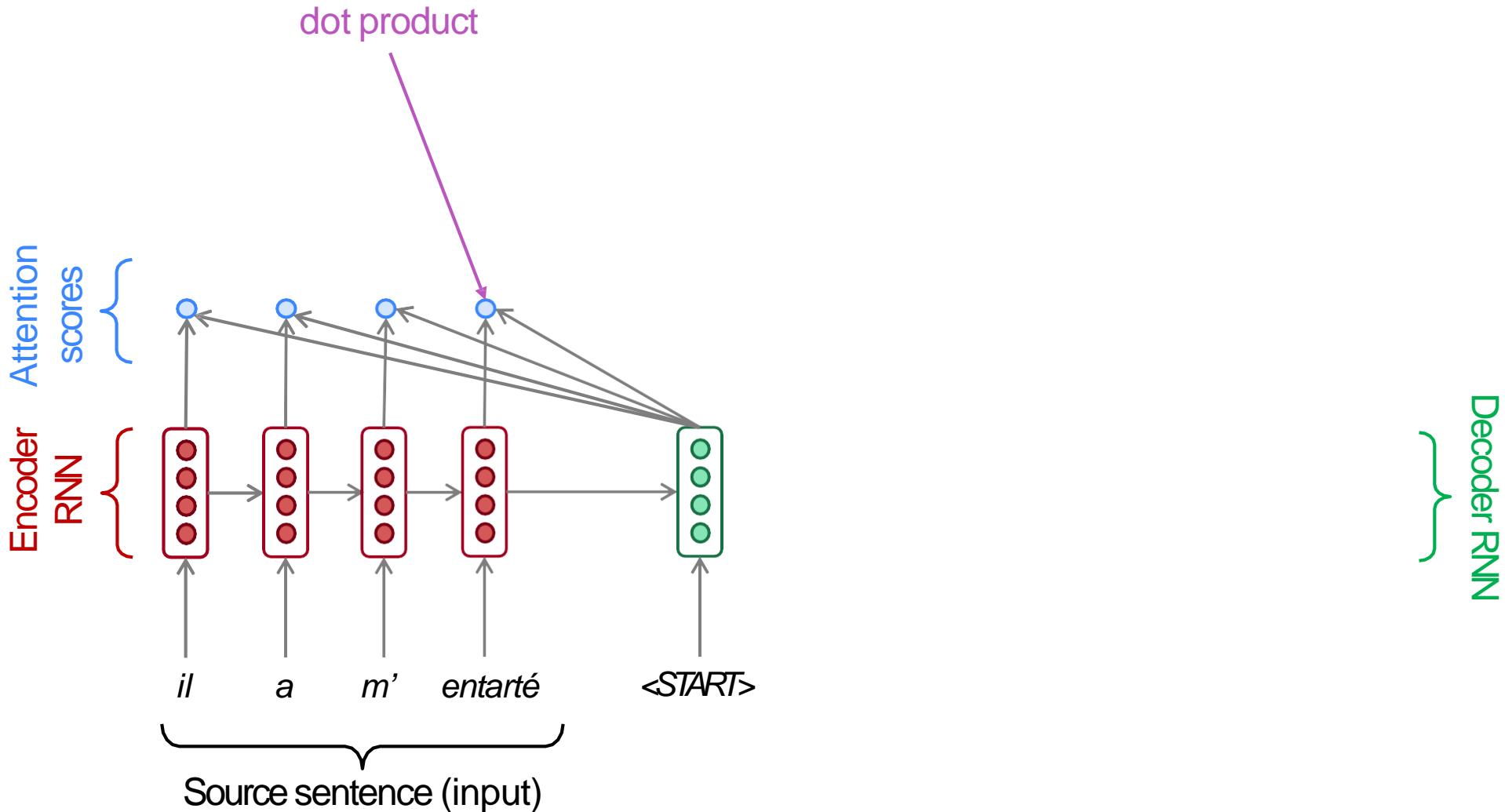
Sequence-to-sequence with attention



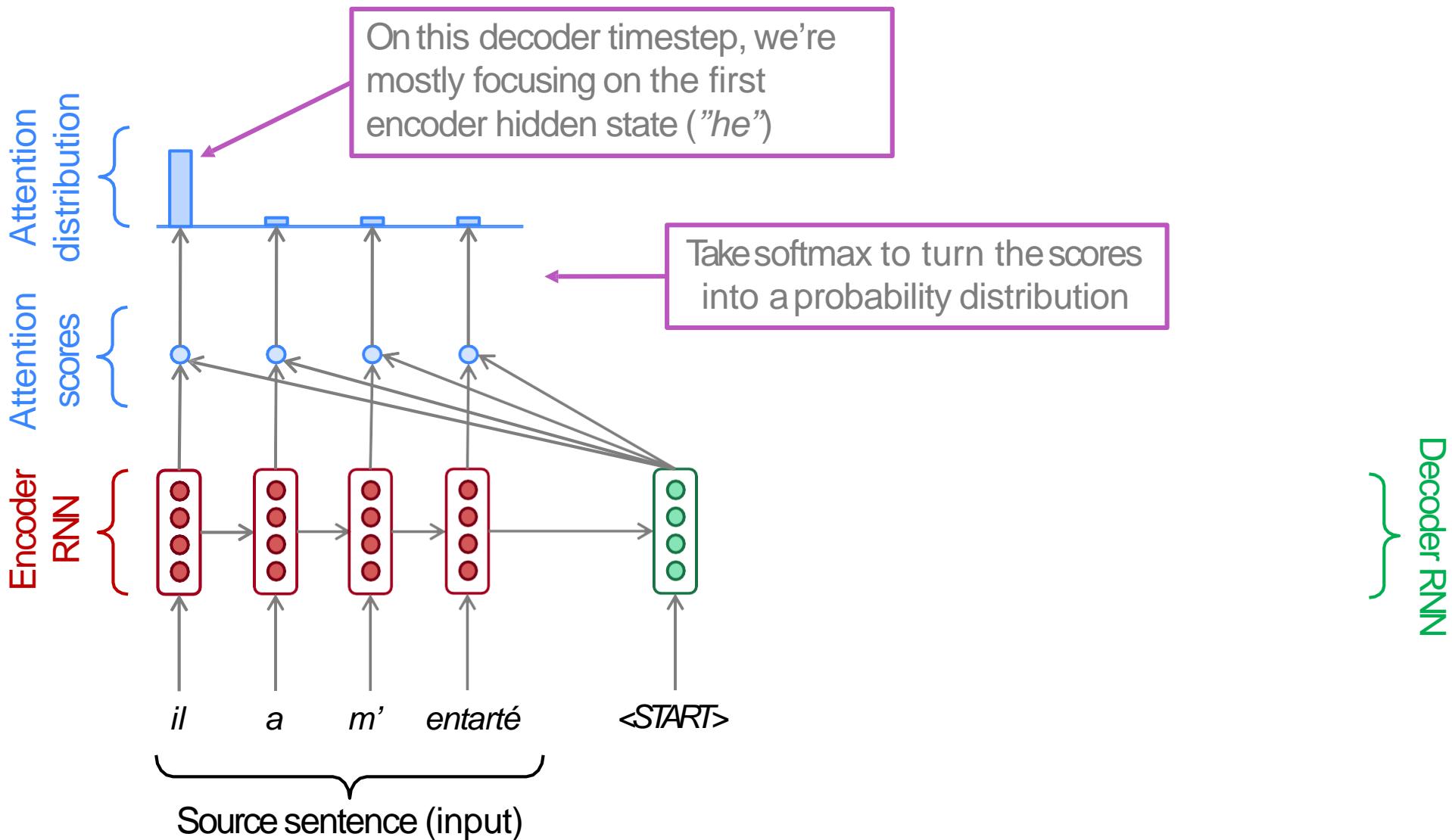
Sequence-to-sequence with attention



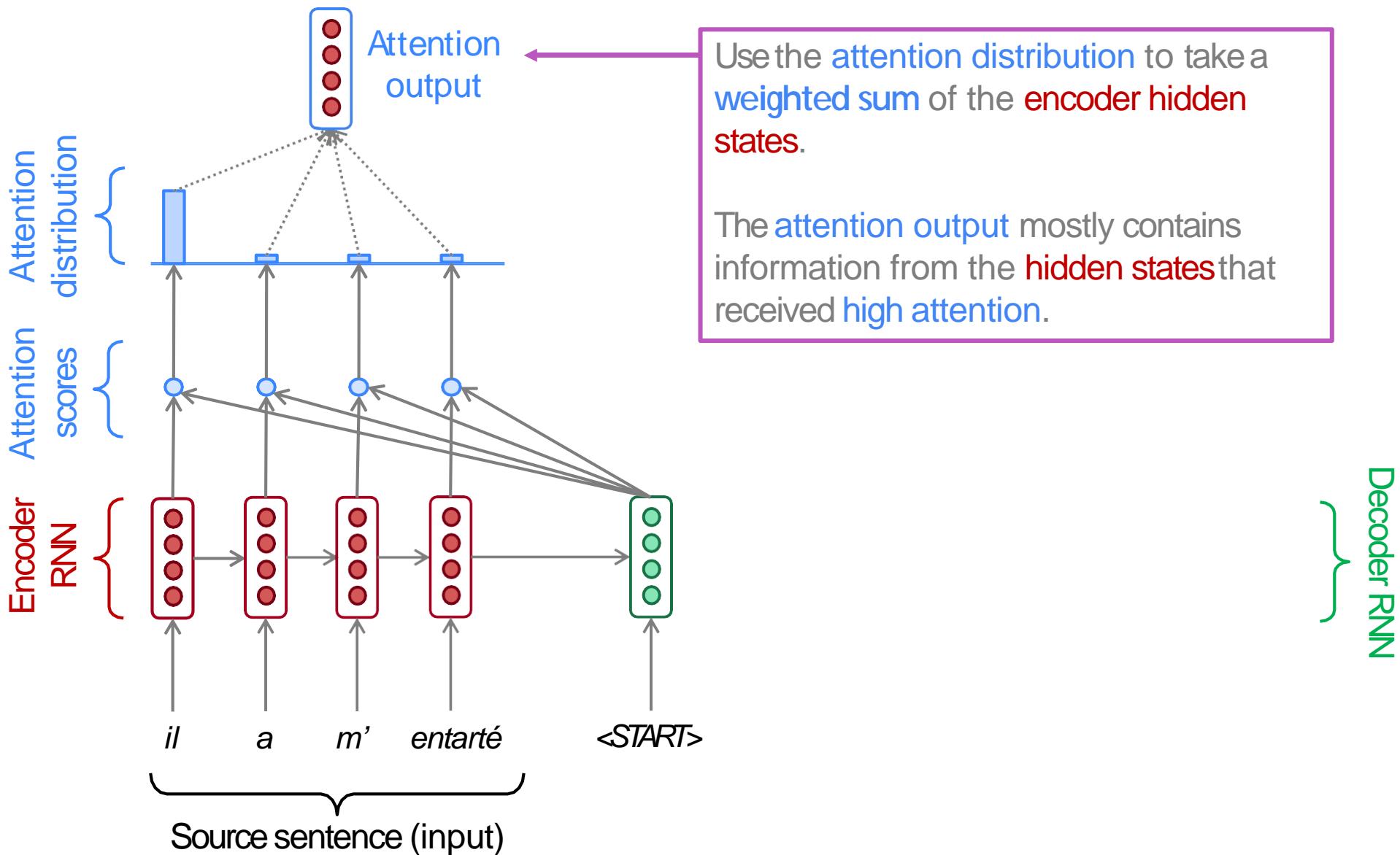
Sequence-to-sequence with attention



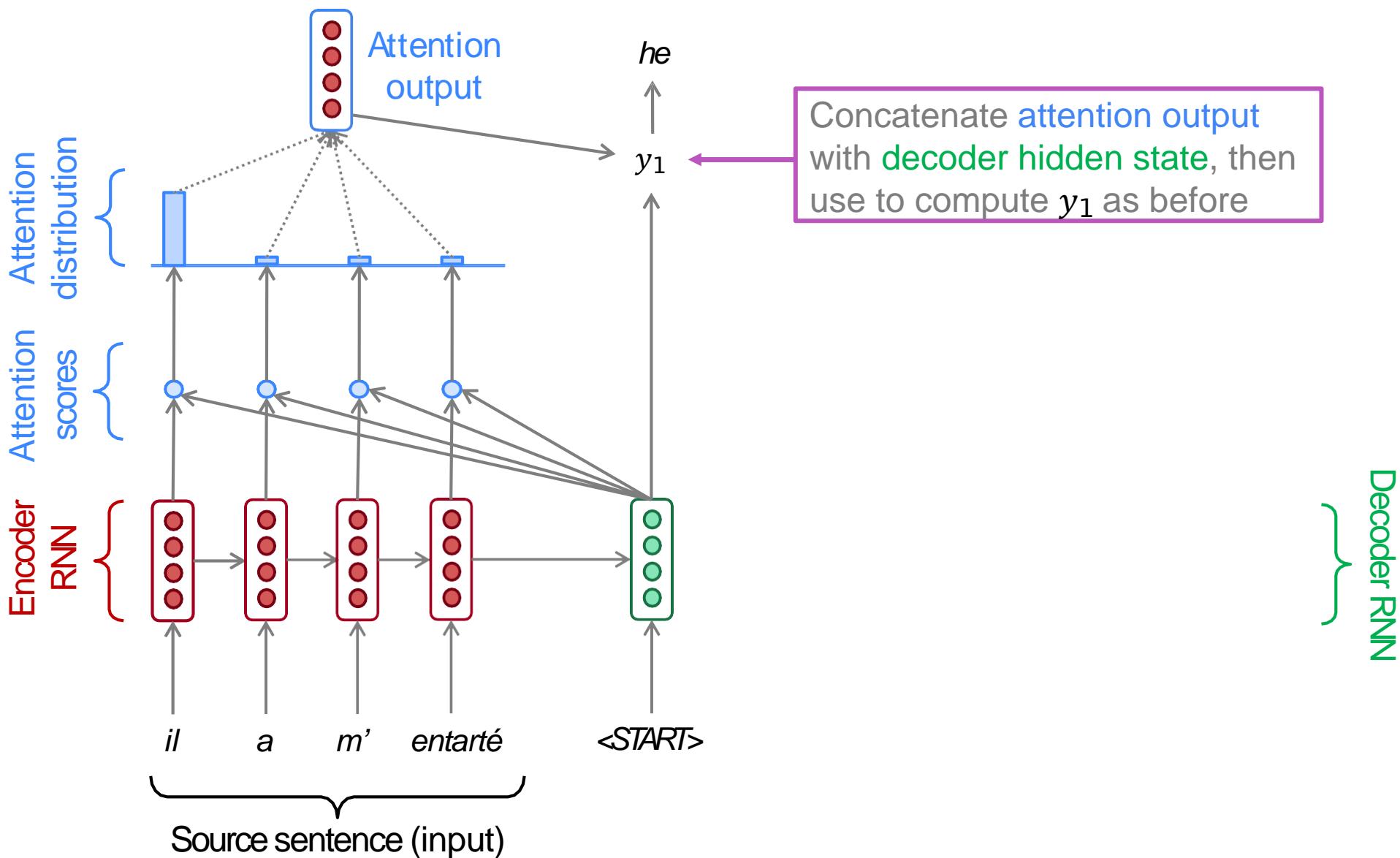
Sequence-to-sequence with attention



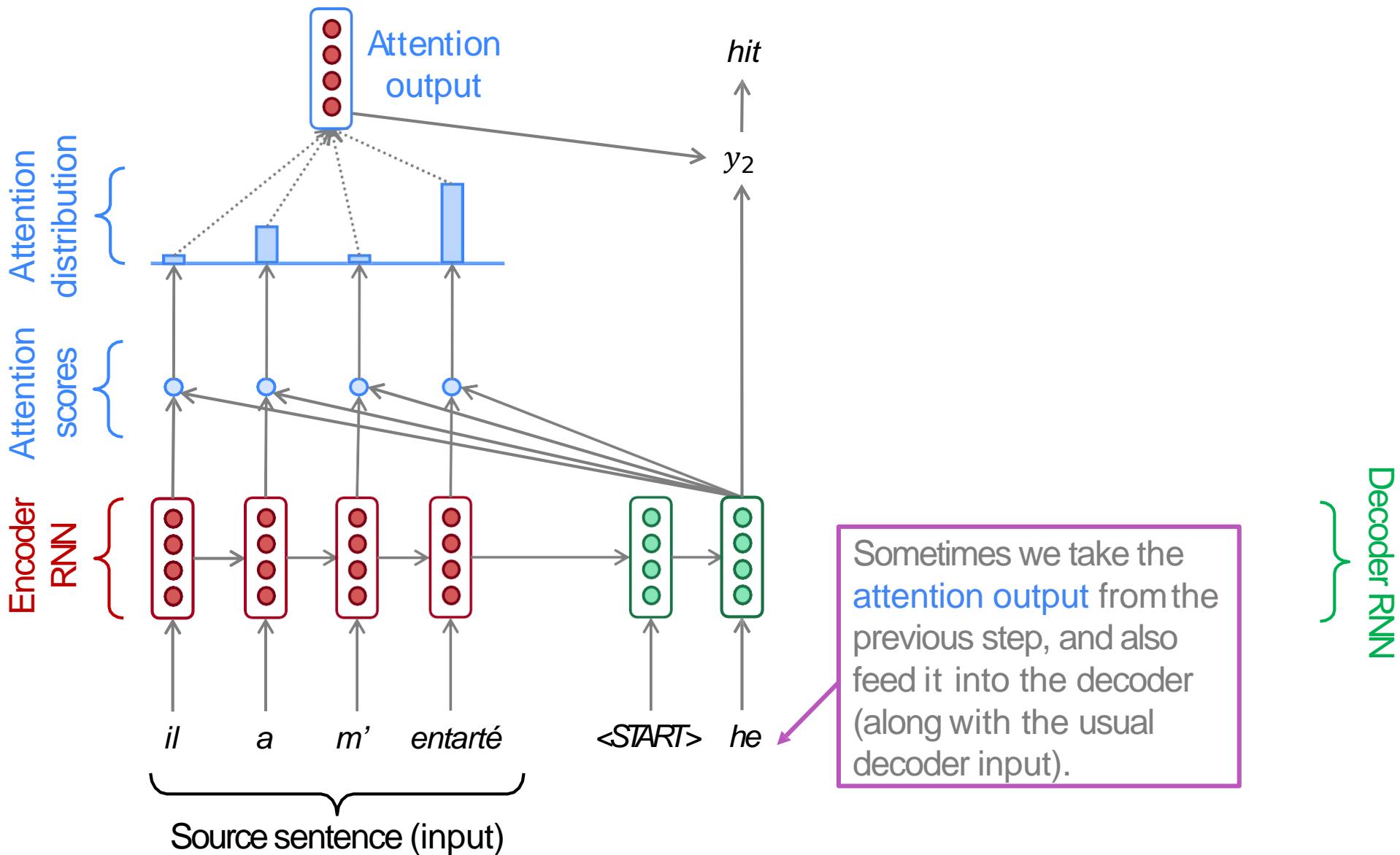
Sequence-to-sequence with attention



Sequence-to-sequence with attention



Sequence-to-sequence with attention



Attention: in equations

- We have encoder hidden states $h_1, \dots, h_N \in \mathbb{R}^h$
- On timestep t , we have decoder hidden state $s_t \in \mathbb{R}^h$
- We get the attention score e^t for this step:

$$e^t = [s_t^T h_1, \dots, s_t^T h_N] \in \mathbb{R}^N$$

- We take softmax to get the attention distribution α^t for this step (this is a probability distribution and sums to 1)

$$\alpha^t = \text{softmax}(e^t) \in \mathbb{R}^N$$

- We use α^t to take a weighted sum of the encoder hidden states to get the attention output a_t

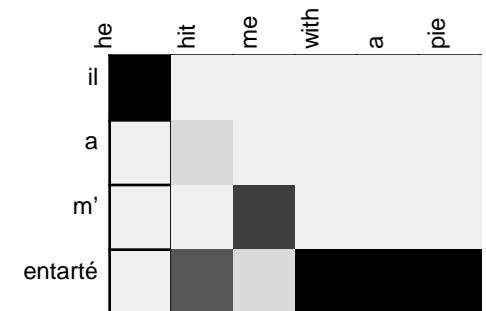
$$a_t = \sum_{i=1}^N \alpha_i^t h_i \in \mathbb{R}^h$$

- Finally we concatenate the attention output a_t with the decoder hidden state s_t and proceed as in the non-attention seq2seq model

$$[a_t; s_t] \in \mathbb{R}^{2h}$$

Attention is great

- Attention significantly improves NMT performance
 - It's very useful to allow decoder to focus on certain parts of the source
- Attention solves the bottleneck problem
 - Attention allows decoder to look directly at source; bypass bottleneck
- Attention helps with vanishing gradient problem
 - Provides shortcut to faraway states
- Attention provides some interpretability
 - By inspecting attention distribution, we can see what the decoder was focusing on
 - We get (soft) alignment for free!
 - This is cool because we never explicitly trained an alignment system
 - The network just learned alignment by itself

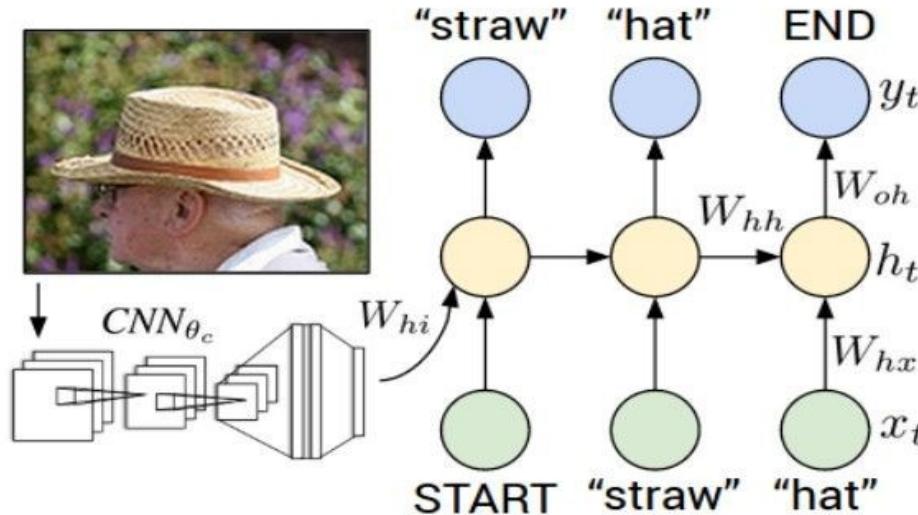


Attention is a *general* Deep Learning technique

- We've seen that attention is a great way to improve the sequence-to-sequence model for Machine Translation.
- However: You can use attention in **many architectures** (not just seq2seq) and **many tasks** (not just MT)

- More general definition of attention:
 - Given a set of vector *values*, and a vector *query*, attention is a technique to compute a weighted sum of the values, dependent on the query.
- We sometimes say that the *query attends to the values*.
- For example, in seq2seq + attention model, each decoder hidden state (query) *attends to* all encoder hidden states (values).

Image Captioning



CVPR 2015:

Deep Visual-Semantic Alignments for Generating Image Descriptions, Karpathy and Fei-Fei

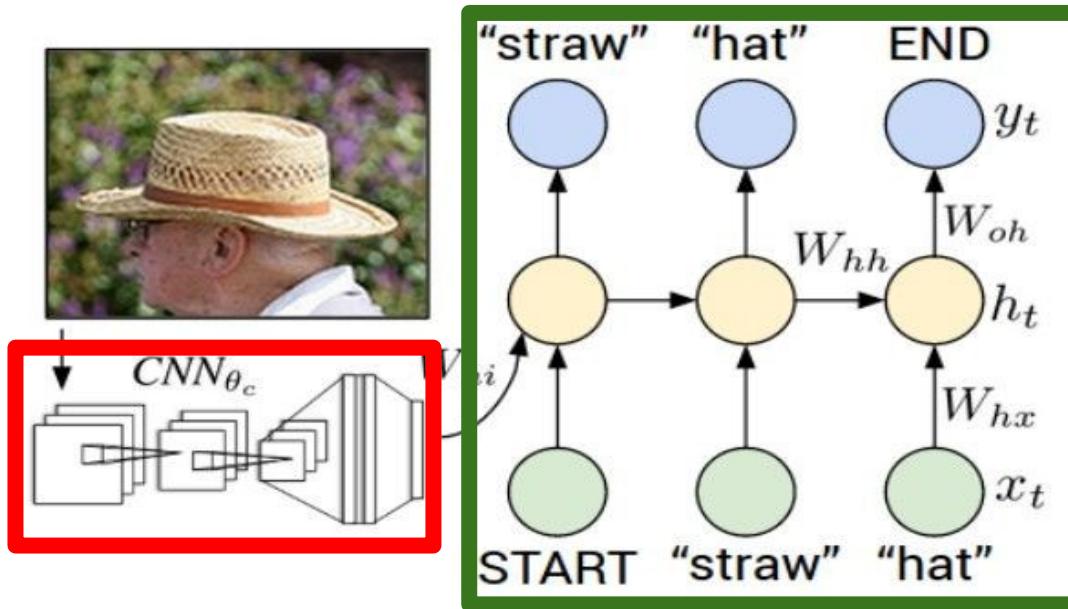
Show and Tell: A Neural Image Caption Generator, Vinyals et al.

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

Image Captioning

Recurrent Neural Network



Convolutional Neural Network

Image Captioning

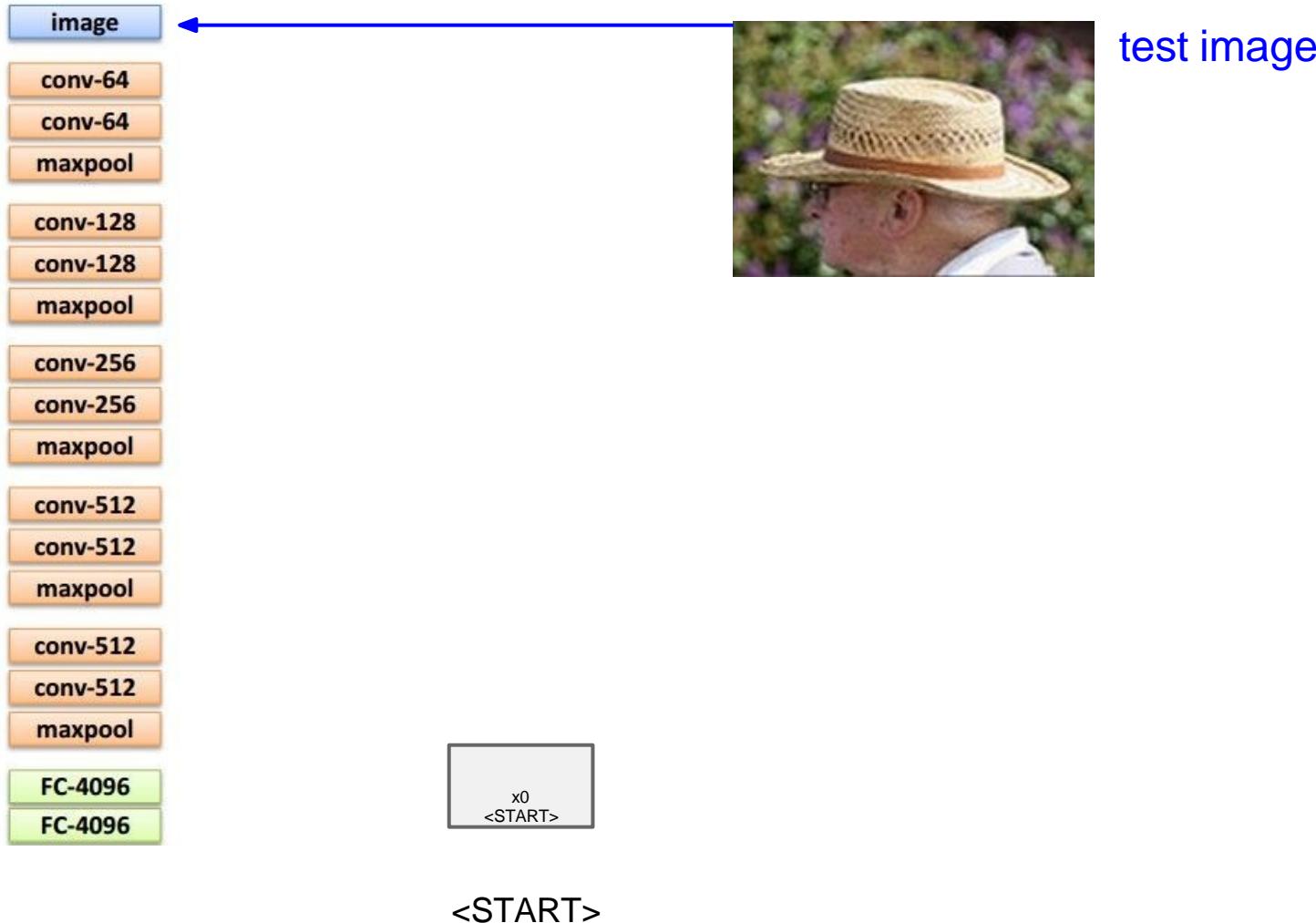


Image Captioning

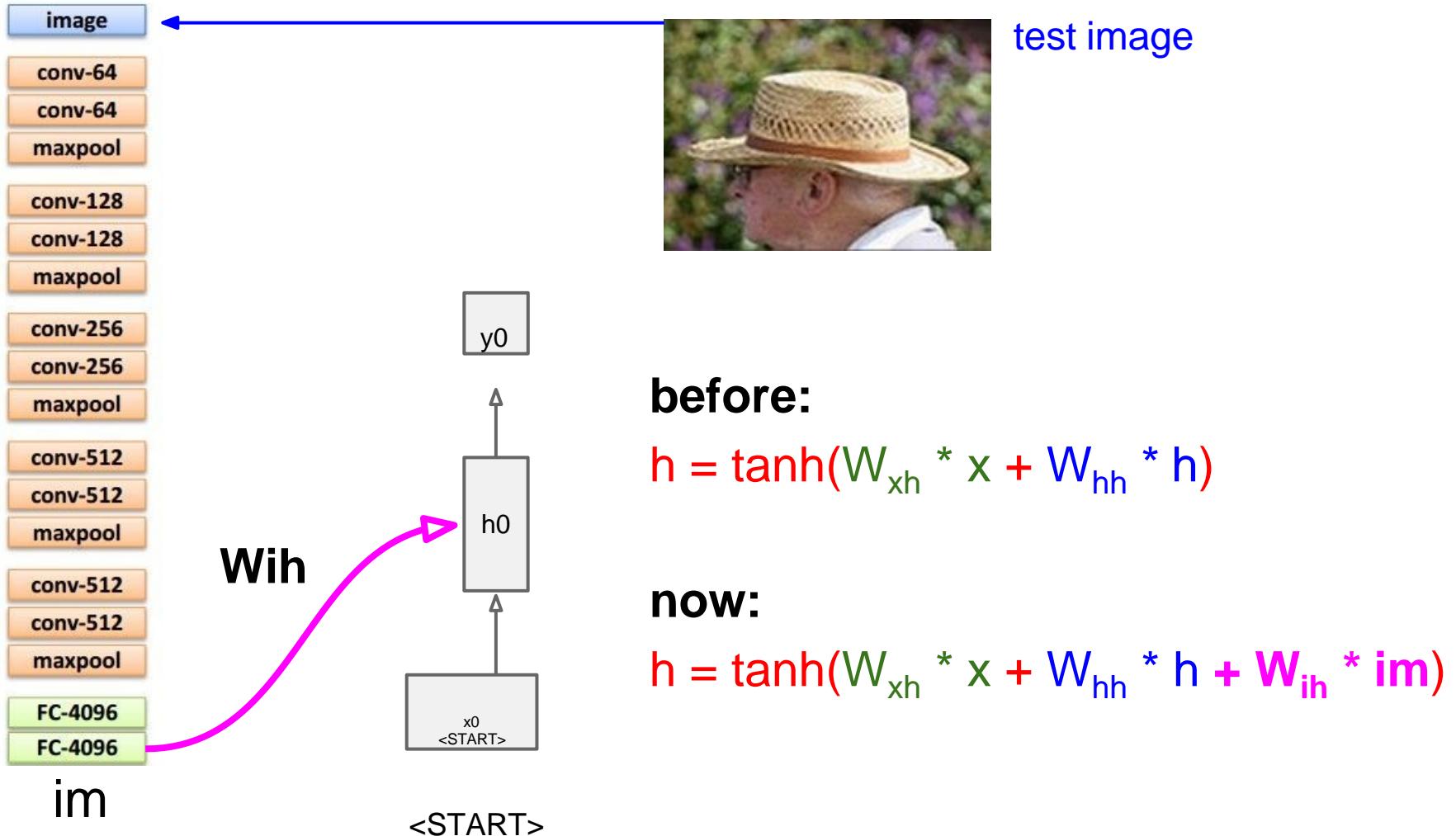


Image Captioning

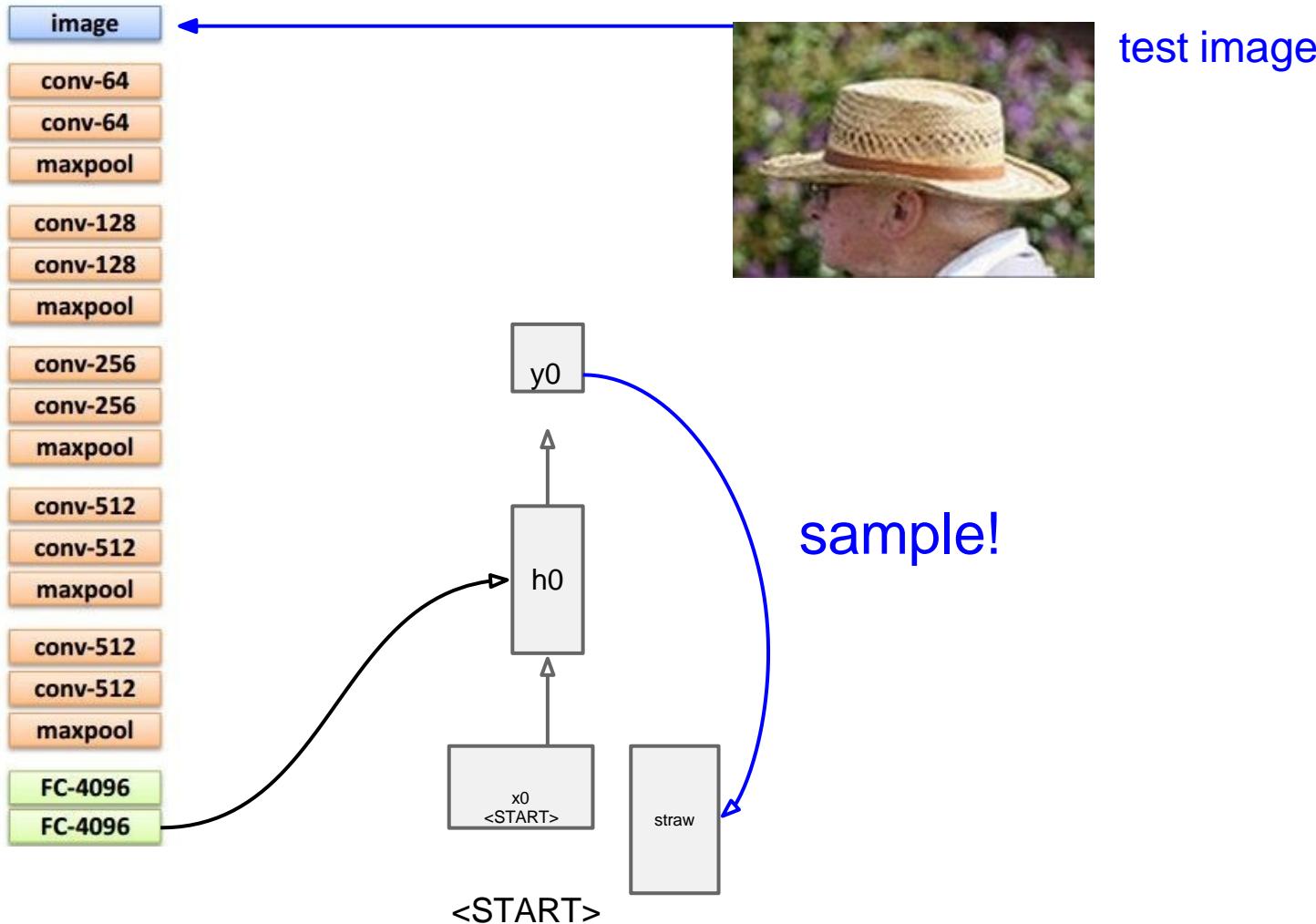


Image Captioning

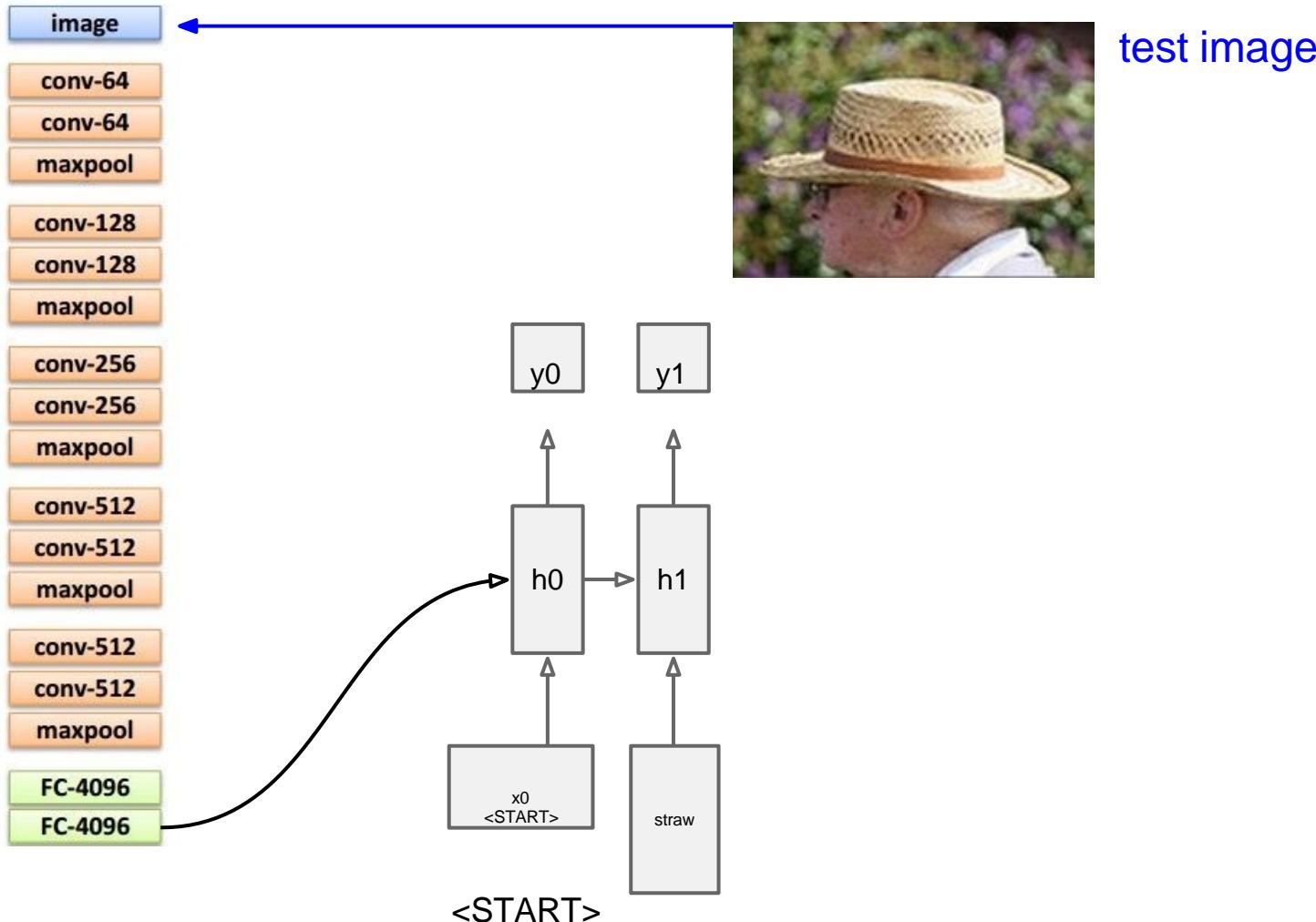


Image Captioning

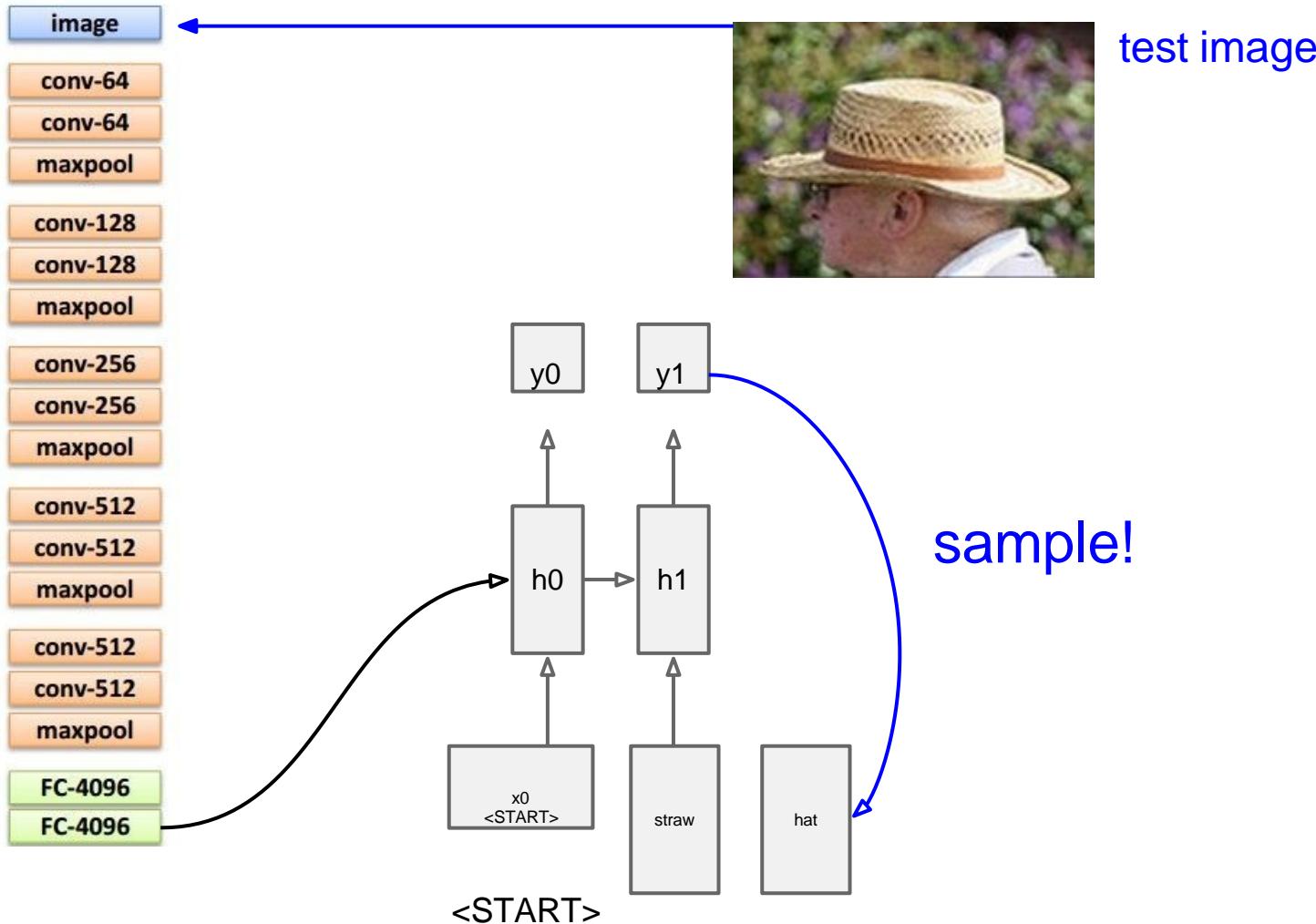


Image Captioning

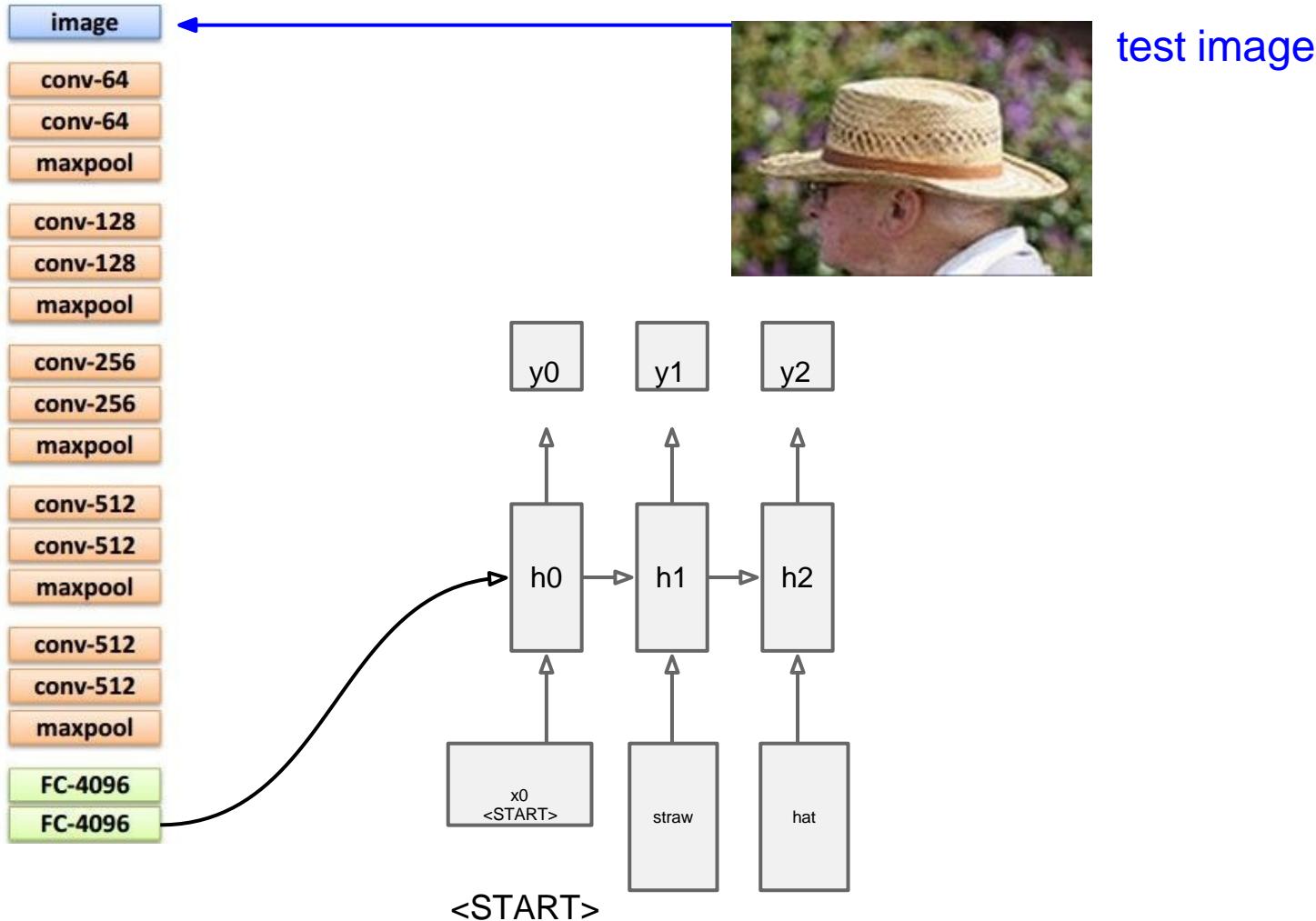


Image Captioning

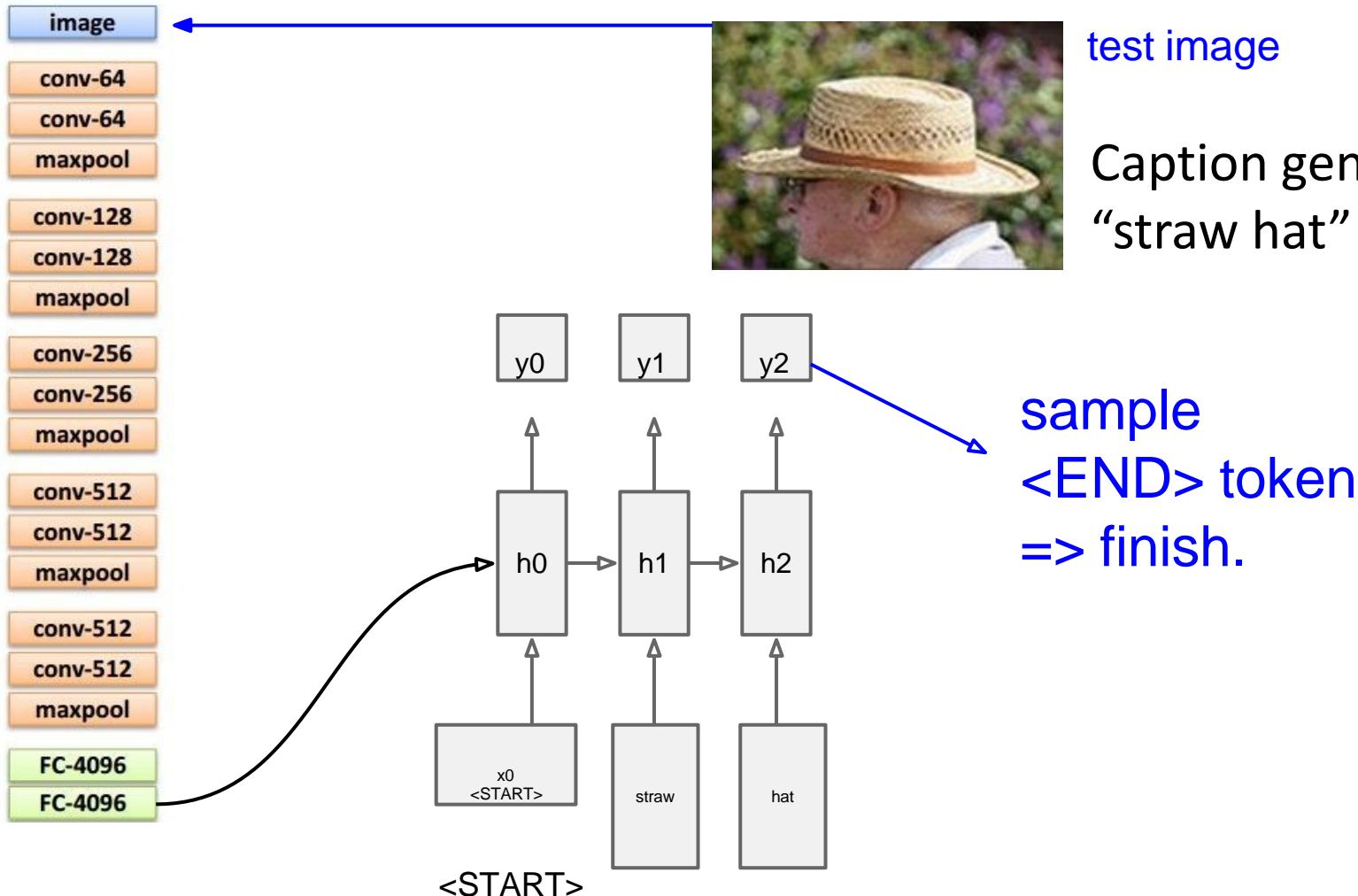


Image Captioning



"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."

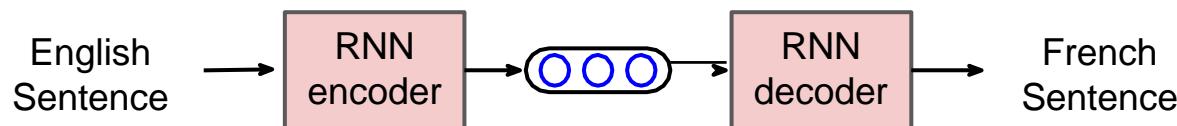


"a woman holding a teddy bear in front of a mirror."

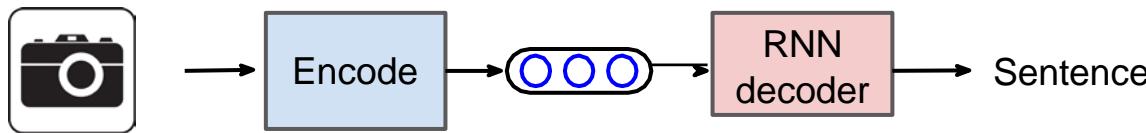


"a horse is standing in the middle of a road."

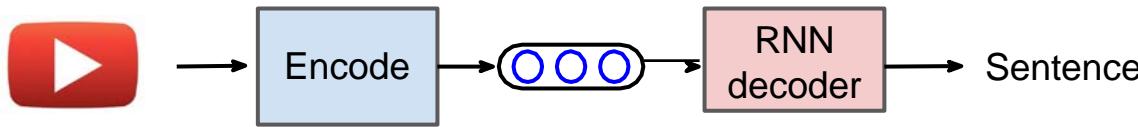
Video Captioning



[Sutskever et al. NIPS'14]



[Donahue et al. CVPR'15]
[Vinyals et al. CVPR'15]

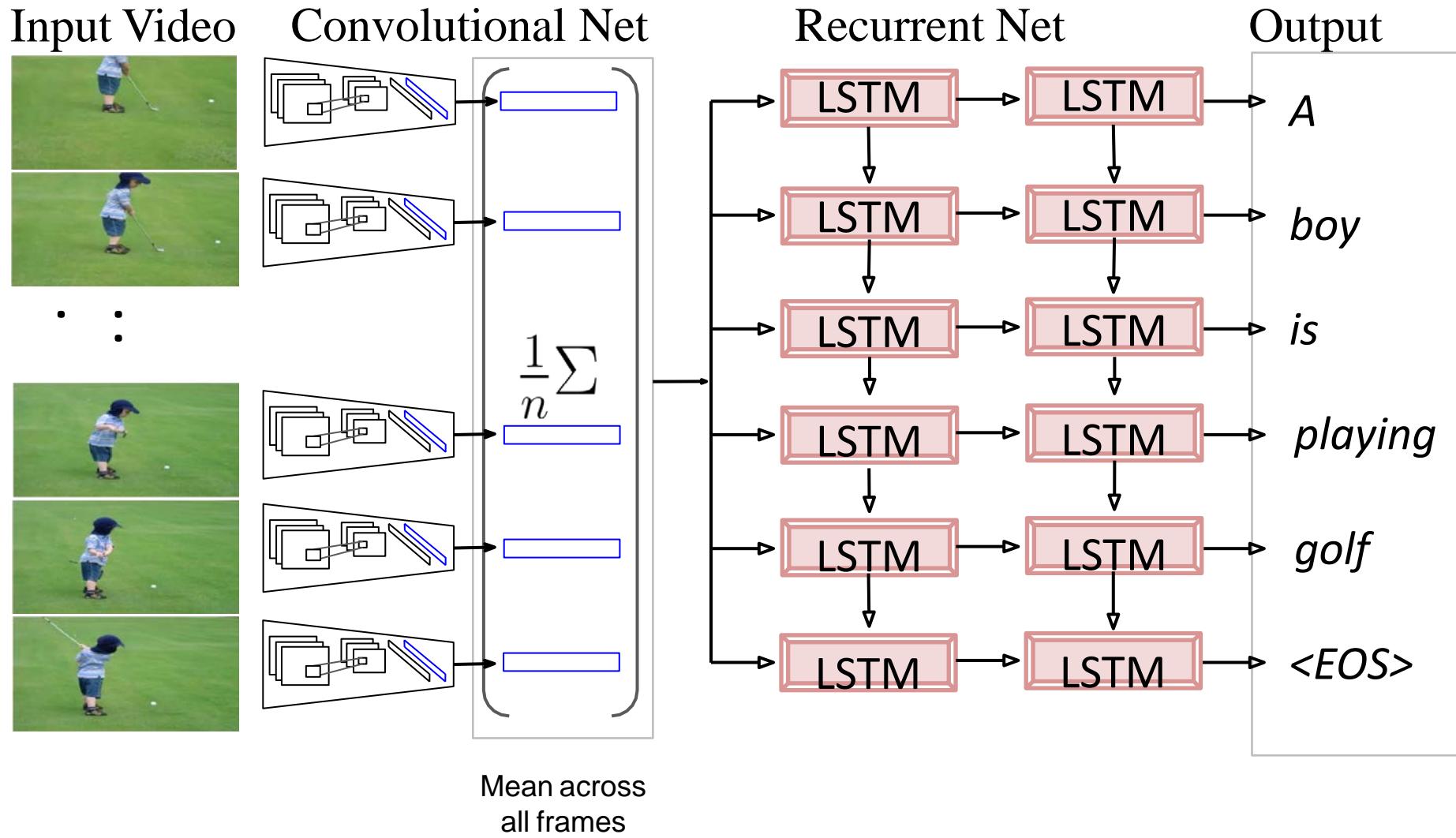


[Venugopalan et. al.
NAACL'15] (this work)

Key Insight:

Generate feature representation of the video and “decode” it to a sentence

Video Captioning



Video Captioning



FGM: A person is dancing with the person on the stage.

YT: A group of men are riding the forest.

I+V: **A group of people are dancing.**

GT: Many men and women are dancing in the street.



FGM: A person is cutting a potato in the kitchen.

YT: A man is slicing a tomato.

I+V: **A man is slicing a carrot.**

GT: A man is slicing carrots.



FGM: A person is walking with a person in the forest.

YT: A monkey is walking.

I+V: **A bear is eating a tree.**

GT: Two bear cubs are digging into dirt and plant matter at the base of a tree.



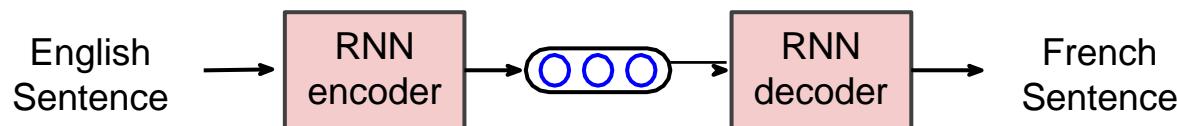
FGM: A person is riding a horse on the stage.

YT: A group of playing are playing in the ball.

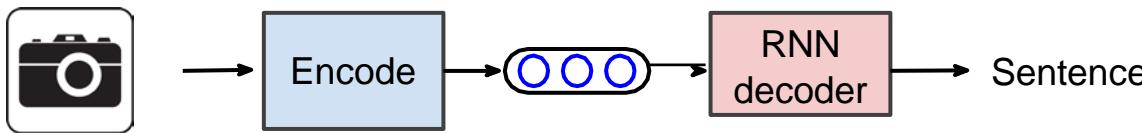
I+V: **A basketball player is playing.**

GT: Dwayne wade does a fancy layup in an allstar game.

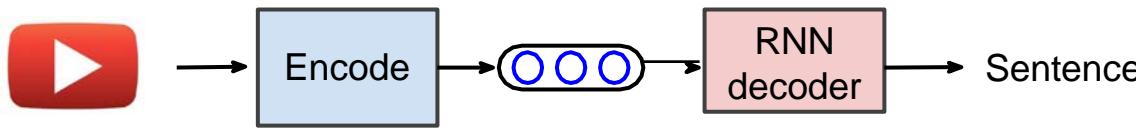
Video Captioning



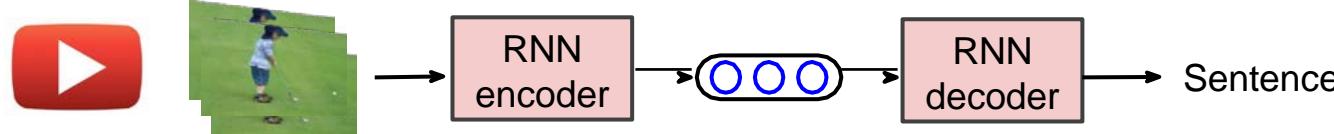
[Sutskever et al. NIPS'14]



[Donahue et al. CVPR'15]
[Vinyals et al. CVPR'15]

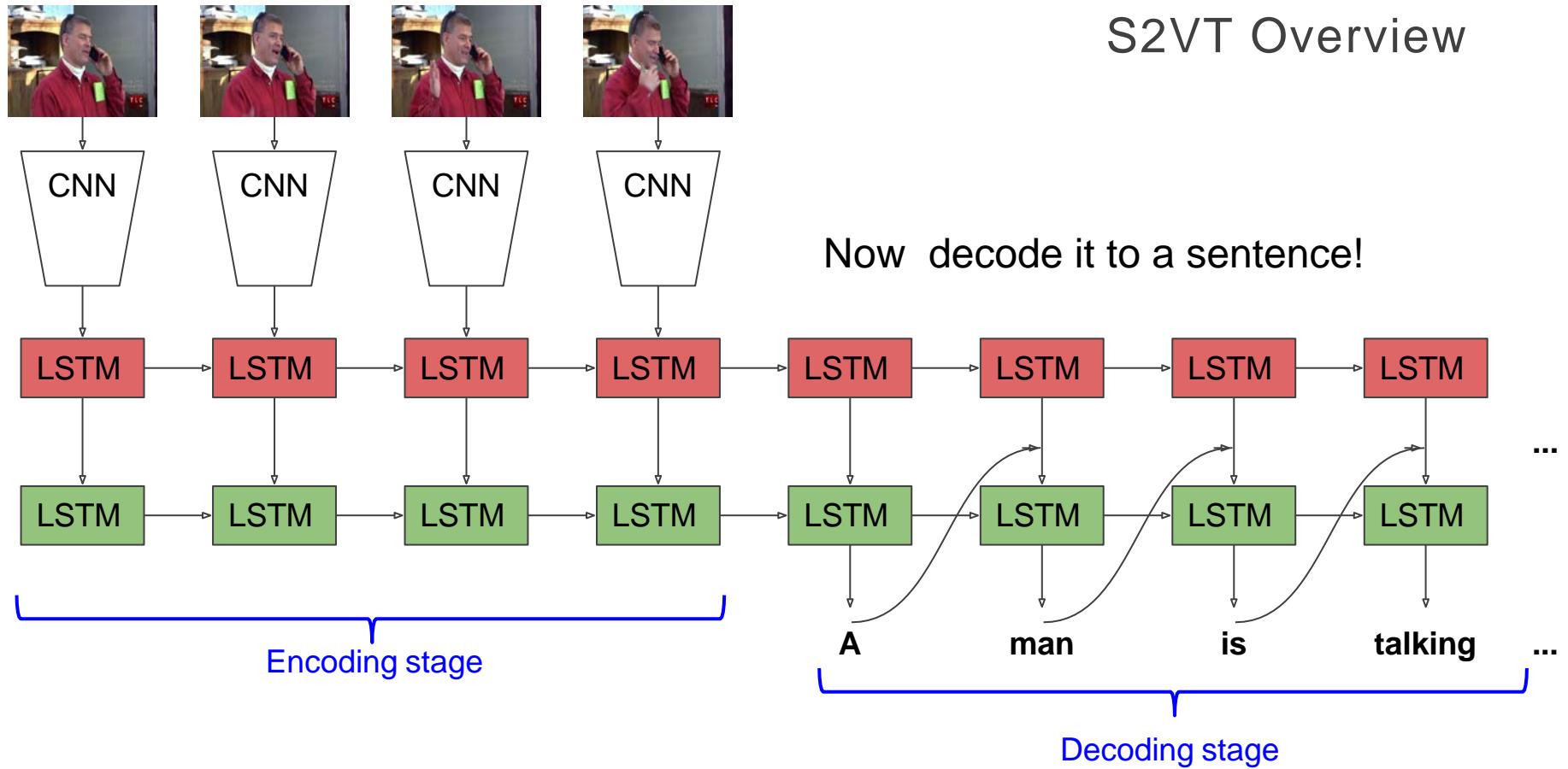


[Venugopalan et. al.
NAACL'15]



[Venugopalan et. al. ICCV'
15] (this work)

Video Captioning





Conversation AI

Bias in the Vision and Language of Artificial Intelligence



Margaret Mitchell
Senior Research Scientist
Google AI



Andrew Zaldivar

Me

Simone Wu

Parker Barnes

Lucy Vasserman

Ben Hutchinson

Elena Spitzer

Deb Raji

Timnit Gebru



Adrian Benton

Brian Zhang

Dirk Hovy

Josh Lovejoy

Alex Beutel

Blake Lemoine

Hee Jung Ryu

Hartwig Adam

Blaise Agüera y Arcas

What do you see?

- Bananas
- Stickers
- Dole Bananas
- Bananas at a store
- Bananas on shelves
- Bunches of bananas
- Bananas with stickers on them
- Bunches of bananas with stickers on them on shelves in a store

...We don't tend to say

Yellow Bananas



What do you see?

Green Bananas

Unripe Bananas



What do you see?

Ripe Bananas

Bananas with spots

Bananas good for
banana bread



What do you see?

Yellow Bananas?

Yellow is prototypical
for bananas



Prototype Theory

One purpose of categorization is to **reduce the infinite differences** among stimuli **to behaviourally and cognitively usable proportions**

There may be some central, prototypical notions of items that arise from stored typical properties for an object category (Rosch, 1975)

May also store exemplars (Wu & Barsalou, 2009)



Fruit



Bananas
“Basic Level”



Unripe Bananas,
Cavendish Bananas

A man and his son are in a terrible accident and are rushed to the hospital in critical care.

The doctor looks at the boy and exclaims "I can't operate on this boy, he's my son!"

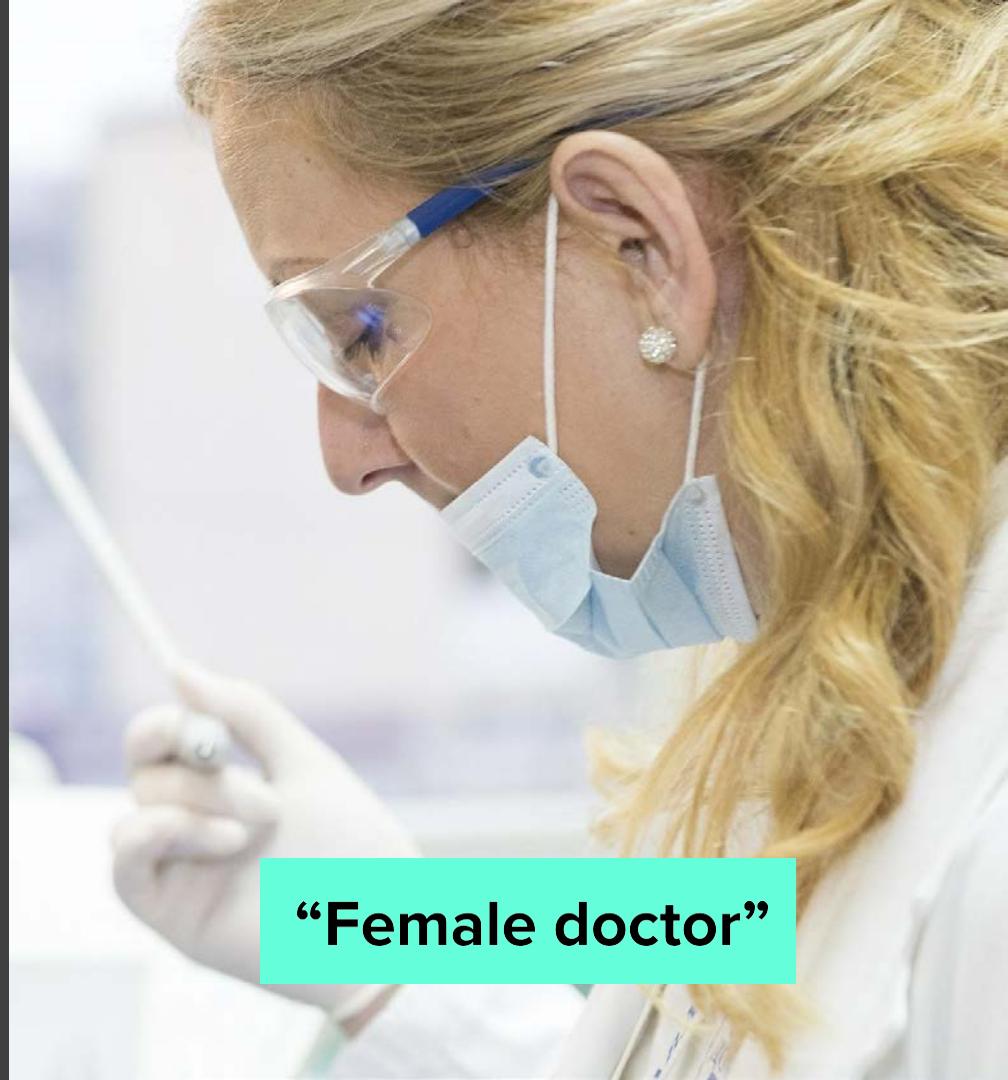
How could this be?



A man and his son are in a terrible accident and are rushed to the hospital in critical care.

The doctor looks at the boy and exclaims "I can't operate on this boy, he's my son!"

How could this be?



“Female doctor”



“Doctor”



“Female doctor”

The majority of test subjects overlooked the possibility that the doctor is a she - including men, women, and self-described feminists.

[Wapman & Belle, Boston University](#)

Human Reporting Bias

The **frequency** with which **people write** about actions, outcomes, or properties is **not a reflection of real-world frequencies** or the degree to which a property is characteristic of a class of individuals

Bias in Language

Extreme <i>she</i> occupations		
1. homemaker	2. nurse	3. receptionist
4. librarian	5. socialite	6. hairdresser
7. nanny	8. bookkeeper	9. stylist
10. housekeeper	11. interior designer	12. guidance counselor

Extreme <i>he</i> occupations		
1. maestro	2. skipper	3. protege
4. philosopher	5. captain	6. architect
7. financier	8. warrior	9. broadcaster
10. magician	11. fighter pilot	12. boss

Figure 1: The most extreme occupations as projected on to the *she-he* gender direction on g2vNEWS. Occupations such as *businesswoman*, where gender is suggested by the orthography, were excluded.

Gender stereotype <i>she-he</i> analogies.		
sewing-carpentry	register-nurse-physician	housewife-shopkeeper
nurse-surgeon	interior designer-architect	softball-baseball
blond-burly	feminism-conservatism	cosmetics-pharmaceuticals
giggle-chuckle	vocalist-guitarist	petite-lanky
sassy-snappy	diva-superstar	charming-affable
volleyball-football	cupcakes-pizzas	hairdresser-barber

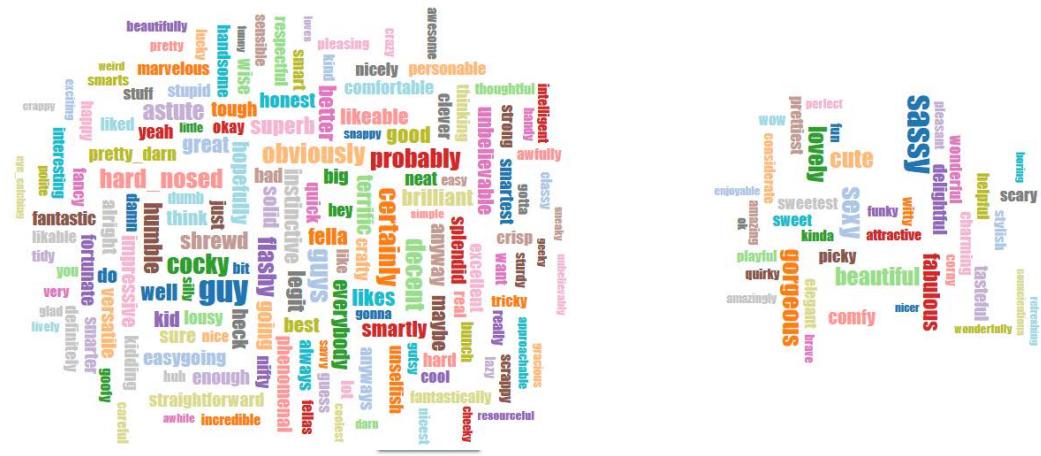
Gender appropriate <i>she-he</i> analogies.		
queen-king	sister-brother	mother-father
waitress-waiter	ovarian cancer-prostate cancer	convent-monastery

Figure 2: **Analogy examples.** Examples of automatically generated analogies for the pair *she-he* using the procedure described in text. For example, the first analogy is interpreted as *she:sewing :: he:carpentry* in the original w2vNEWS embedding. Each automatically generated analogy is evaluated by 10 crowd-workers to whether or not it reflects gender stereotype. Top: illustrative gender stereotypic analogies automatically generated from w2vNEWS, as rated by at least 5 of the 10 crowd-workers. Bottom: illustrative generated gender-appropriate analogies.

Bias in Language

he (158)

she (42)



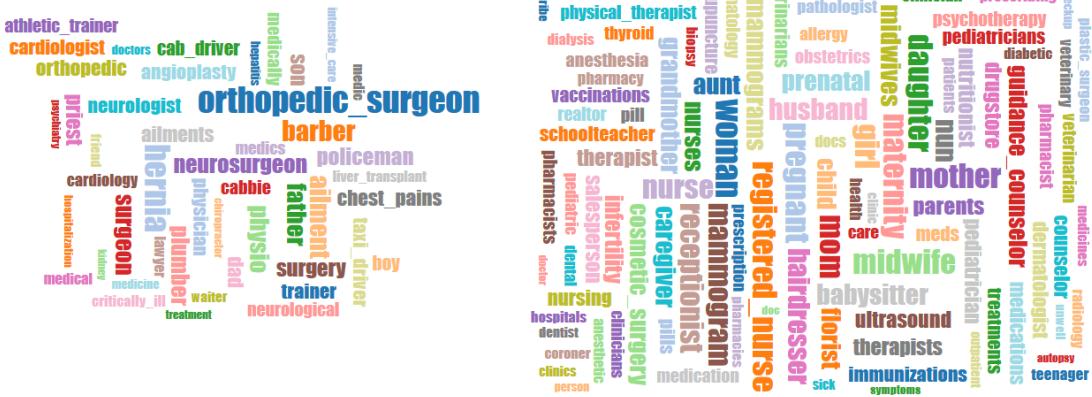
Adjectives

Or type your own words...

doctor

he (47)

she (153)



But...don't forget the biases in the initial data

Man is to Computer Programmer as Woman is to Homemaker?
Debiasing Word Embeddings

Tolga Bolukbasi¹, Kai-Wei Chang², James Zou², Venkatesh Saligrama^{1,2}, Adam Kalai²

¹Boston University, 8 Saint Mary's Street, Boston, MA

²Microsoft Research New England, 1 Memorial Drive, Cambridge, MA

tolgab@bu.edu, kw@kwchang.net, jamesyzou@gmail.com, srv@bu.edu, adam.kalai@microsoft.com

Bias in Vision

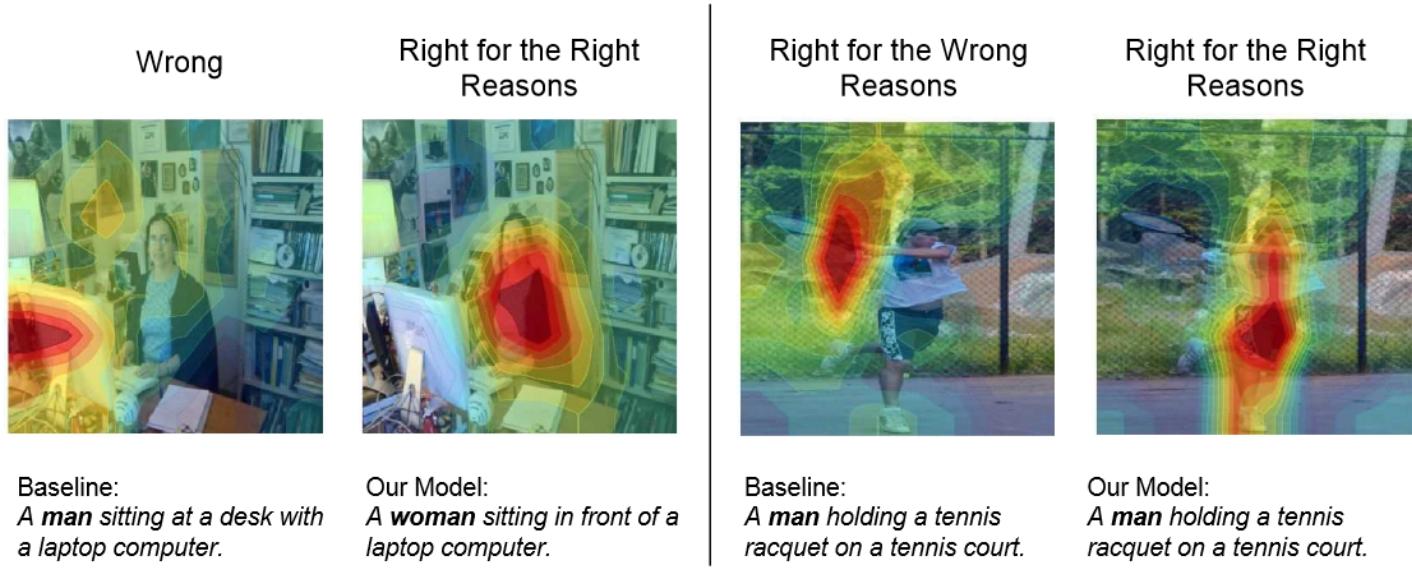


Fig. 1: Examples where our proposed model (Equalizer) corrects bias in image captions. The overlaid heatmap indicates which image regions are most important for predicting the gender word. On the left, the baseline predicts gender incorrectly, presumably because it looks at the laptop (not the person). On the right, the baseline predicts the gender correctly but it does not look at the person when predicting gender and is thus not acceptable. In contrast, our model predicts the correct gender word and correctly considers the person when predicting gender.

Bias in Vision

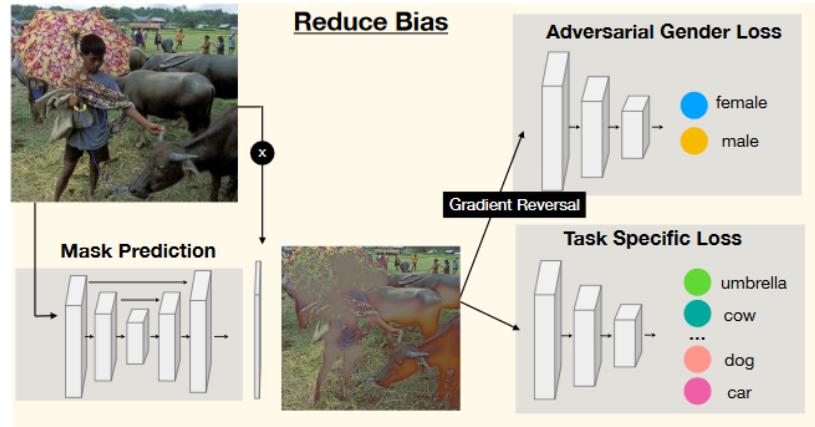


Figure 2. In our bias mitigation approach, we learn a task-specific model with an adversarial loss that removes features corresponding to a protected variable from an intermediate representation in the model – here we illustrate our pipeline to visualize the removal of features in image space through an auto-encoder network.



Figure 3. Images after adversarial removal of gender when applied to the image space. The objective was to preserve information about objects and verbs, e.g. scissors, banana (COCO) or vaulting, lifting (imSitu) while removing gender correlated features.





Biases in Data

Biases in Data

Selection Bias: Selection does not reflect a random sample



CREDIT

[© 2013–2016 Michael Yoshitaka Erlewine and Hadas Kotek](#)

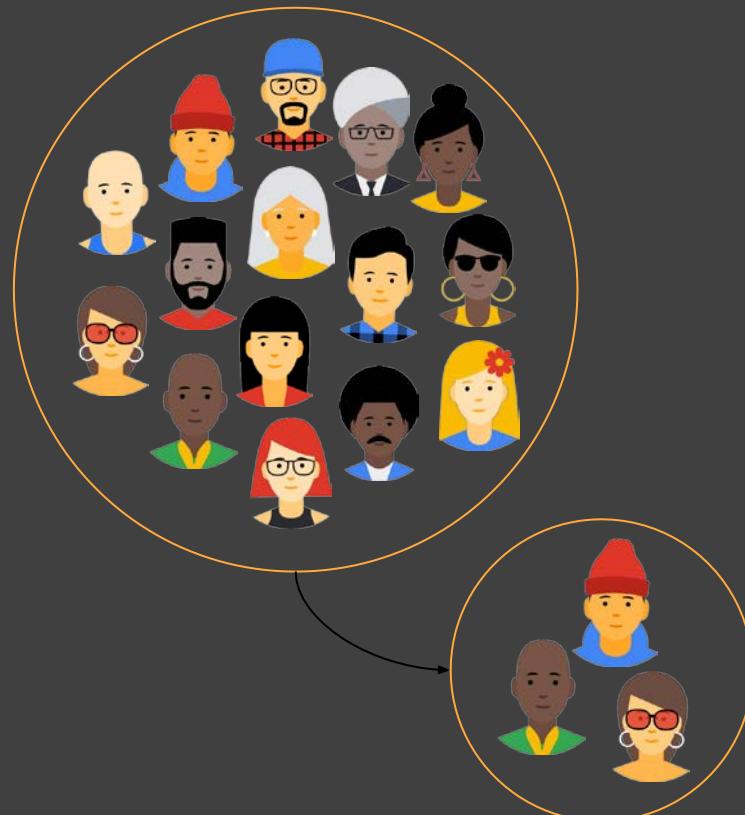
Biases in Data

Out-group homogeneity bias: Tendency to see outgroup members as more alike than ingroup members



Biases in Data → Biased Data Representation

It's possible that you have an appropriate amount of data for every group you can think of but that some groups are represented less positively than others.



Biases in Data → Biased Labels

Annotations in your dataset will reflect the worldviews of your annotators.



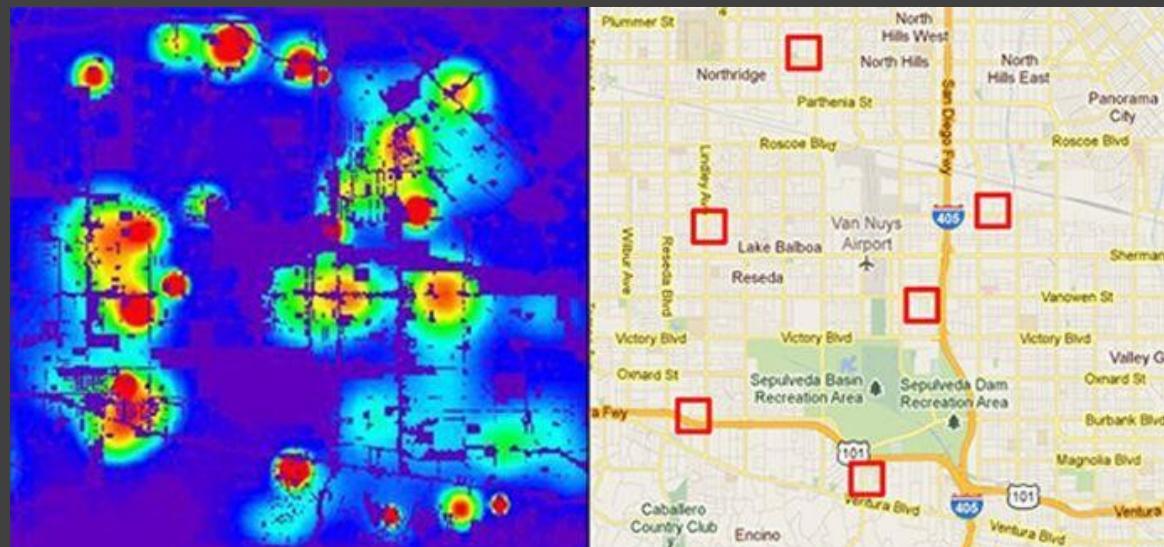
<https://ai.googleblog.com/2018/09/introducing-inclusive-images-competition.html>



Predicting Future Criminal Behavior

Predicting Policing

- Algorithms identify potential crime hot-spots
- Based on where crime is previously reported, not where it is known to have occurred
- Predicts future events from past



CREDIT

[Smithsonian. Artificial Intelligence Is Now Used to Predict Crime. But Is It Biased? 2018](#)

Predicting Sentencing

- Prater (who is white) rated **low risk** after shoplifting, despite two armed robberies; one attempted armed robbery.
- Borden (who is black) rated **high risk** after she and a friend took (but returned before police arrived) a bike and scooter sitting outside.
- Two years later, Borden has not been charged with any new crimes. Prater serving 8-year prison term for grand theft.

CREDIT

[ProPublica. Northpointe: Risk in Criminal Sentencing. 2016.](#)

Predicting Criminality

Israeli startup, [Faception](#)

“Faception is first-to-technology and first-to-market with proprietary computer vision and machine learning technology for profiling people and revealing their personality based only on their facial image.”

Offering specialized engines for recognizing “High IQ”, “White-Collar Offender”, “Pedophile”, and “Terrorist” from a face image.

Main clients are in homeland security and public safety.

Predicting Criminality

[“Automated Inference on Criminality using Face Images”](#) Wu and Zhang, 2016.

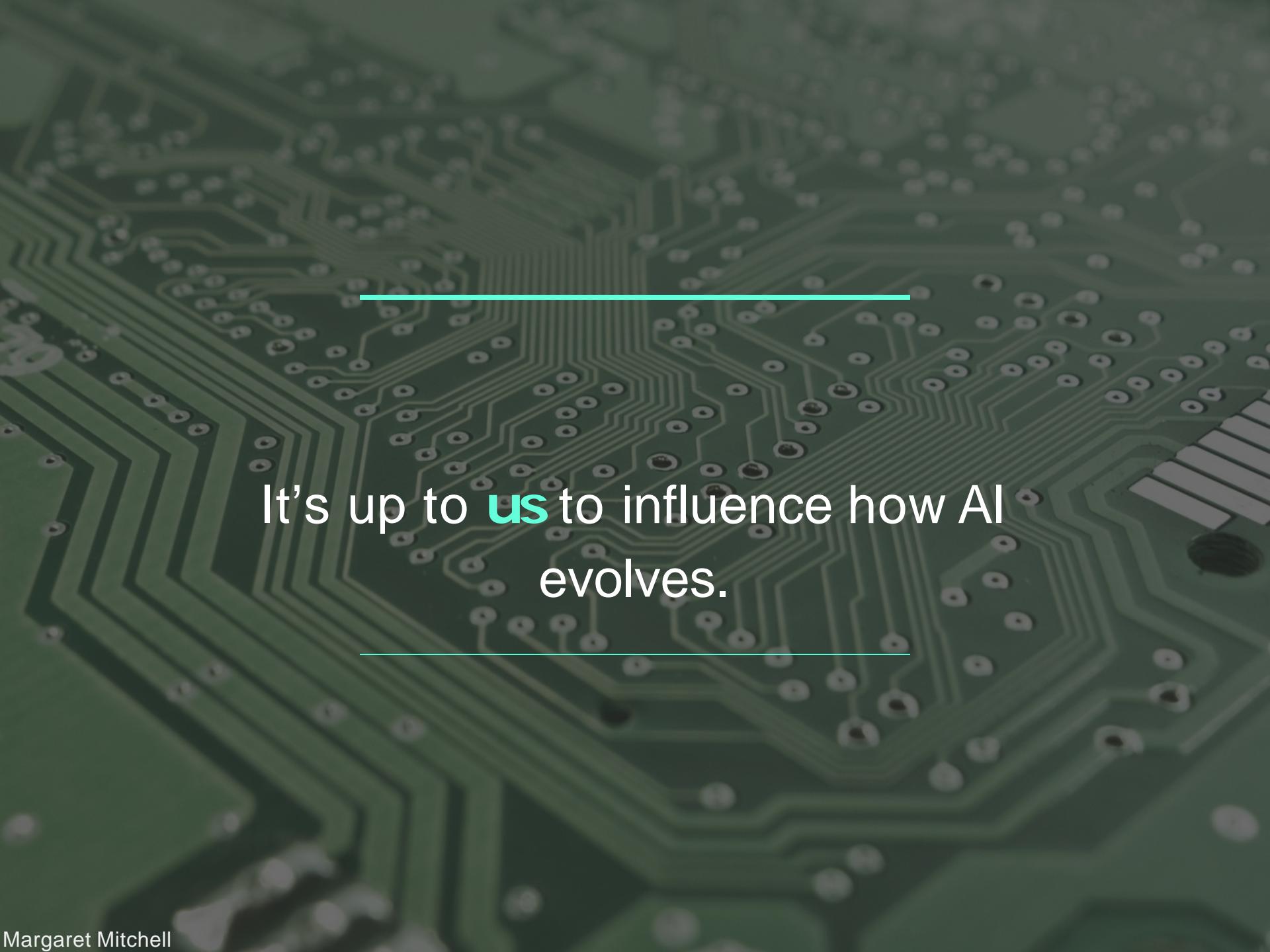
arXiv

1,856 closely cropped images of faces;
Includes “wanted suspect” ID pictures
from specific regions.

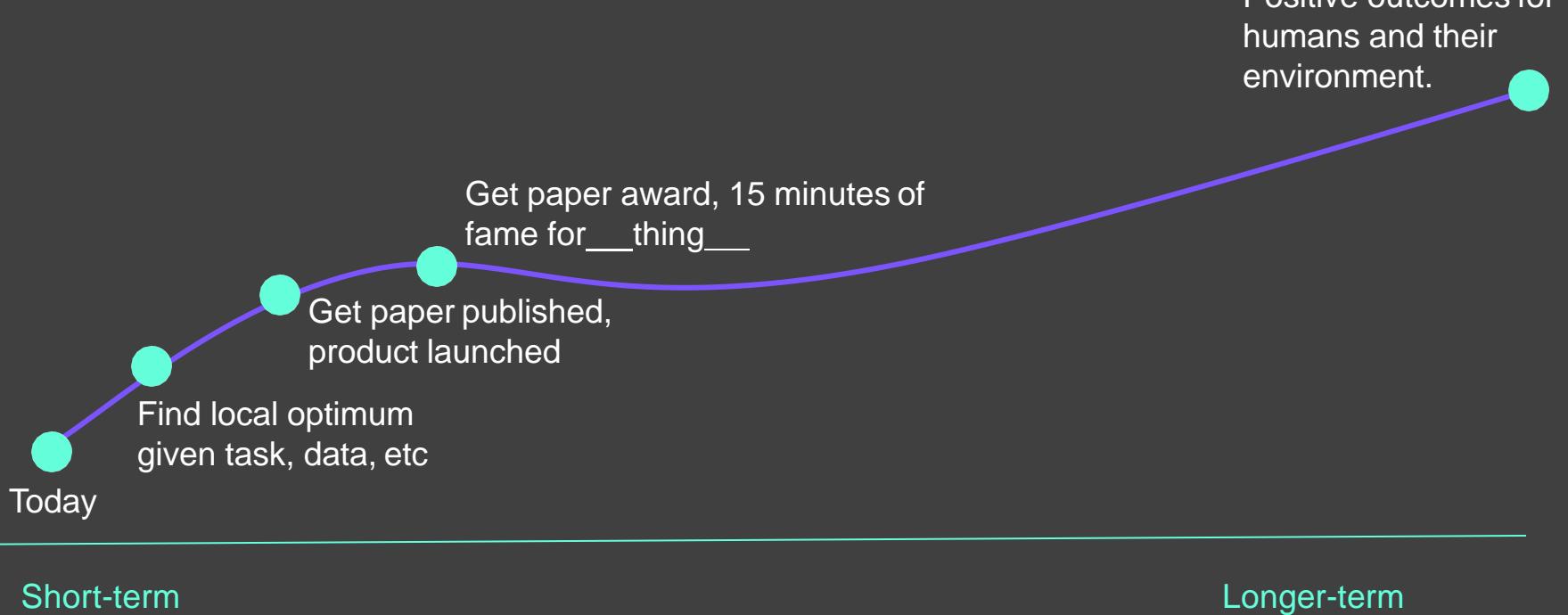
*[...] angle θ from nose tip to two
mouth corners is on average 19.6%
smaller for criminals than for
non-criminals ...”*



See our longer piece on Medium, “[Physiognomy’s New Clothes](#)”



It's up to **us** to influence how AI evolves.



Beware the biases!

Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings

Tolga Bolukbasi¹, Kai-Wei Chang², James Zou², Venkatesh Saligrama^{1,2}, Adam Kalai²

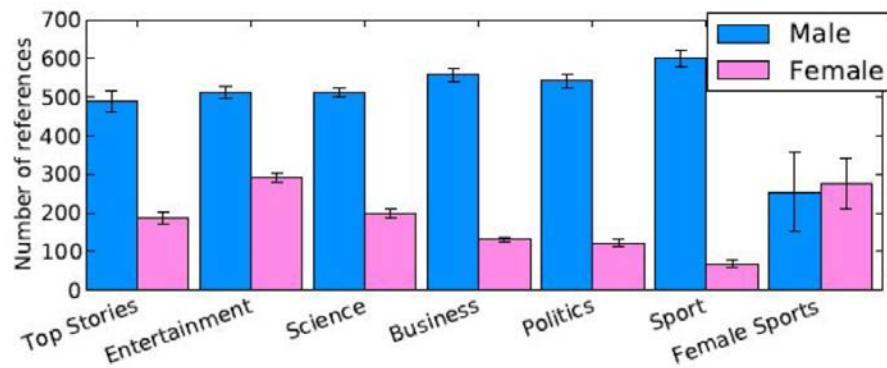
¹Boston University, 8 Saint Mary's Street, Boston, MA

²Microsoft Research New England, 1 Memorial Drive, Cambridge, MA

tolgab@bu.edu, kw@kwchang.net, jamesyzou@gmail.com, srv@bu.edu, adam.kalai@microsoft.com

A brief abstract

Beware the biases!



<https://translate.google.com>
L

TABLE I: List of the top 10 occupations per gender by their association with gender.

Gender	Occupations most associated with a gender
Male	Manager, Engineer, Coach, Executive, Surveyor, Secretary, Architect, Driver, Police, Caretaker, Director
Female	Housekeeper, Nurse, Therapist, Bartender, Psychologist, Designer, Pharmacist, Supervisor, Radiographer, Underwriter

From Nello Cristianini, at at *Frontier Research and Artificial Intelligence Conference*:

https://erc.europa.eu/sites/default/files/events/docs/Nello_Cristianini-ThinkBIG-Patterns-in-Big-Data.pdf

Beware the biases!

BUSINESS NEWS OCTOBER 10, 2018 / 5:12 AM / 7 MONTHS AGO

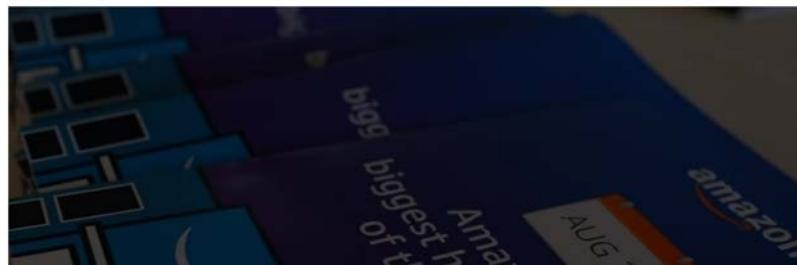
Amazon scraps secret AI recruiting tool that showed bias against women

Jeffrey Dastin

8 MIN READ



SAN FRANCISCO (Reuters) - Amazon.com Inc's (AMZN.O) machine-learning specialists uncovered a big problem: their new recruiting engine did not like women.



Forget Killer Robots—Bias Is the Real AI Danger

John Giannandrea,
GETTY

Artificial Intelligence / Robots

Forget Killer Robots— Bias Is the Real AI Danger

John Giannandrea, who leads AI at Google, is worried about intelligent systems learning human prejudices.

by Will Knight

Oct 3, 2017

Google's AI chief isn't fretting about super-intelligent killer robots. Instead, John Giannandrea is concerned about the danger that may be lurking inside the machine-learning algorithms used to make millions of decisions every minute.

"The real safety question, if you want to call it that, is that if we give these systems biased data, they will be biased," Giannandrea said before a recent Google conference on the relationship between humans and AI systems.

The problem of bias in machine learning is likely to become more significant as the technology spreads to critical areas like medicine and law, and as more people without a deep technical understanding are tasked with deploying it.

Beware the biases!



Joy Buolamwini

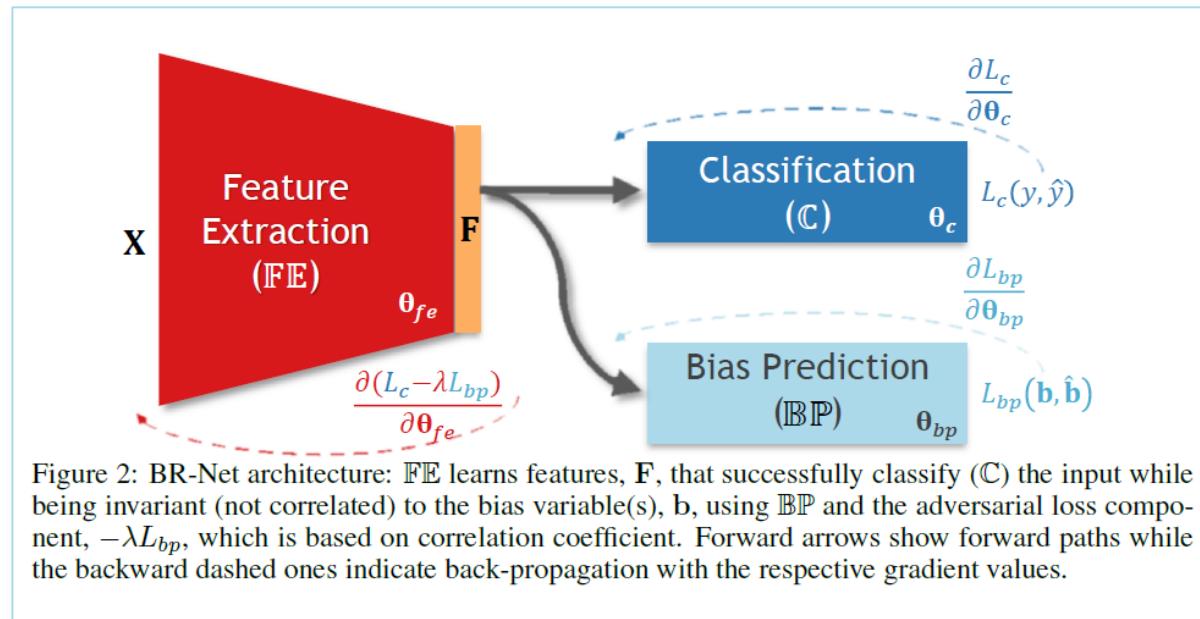
<http://gendershades.org/>

<https://www.youtube.com/watch?v=T2oZvzgrill>



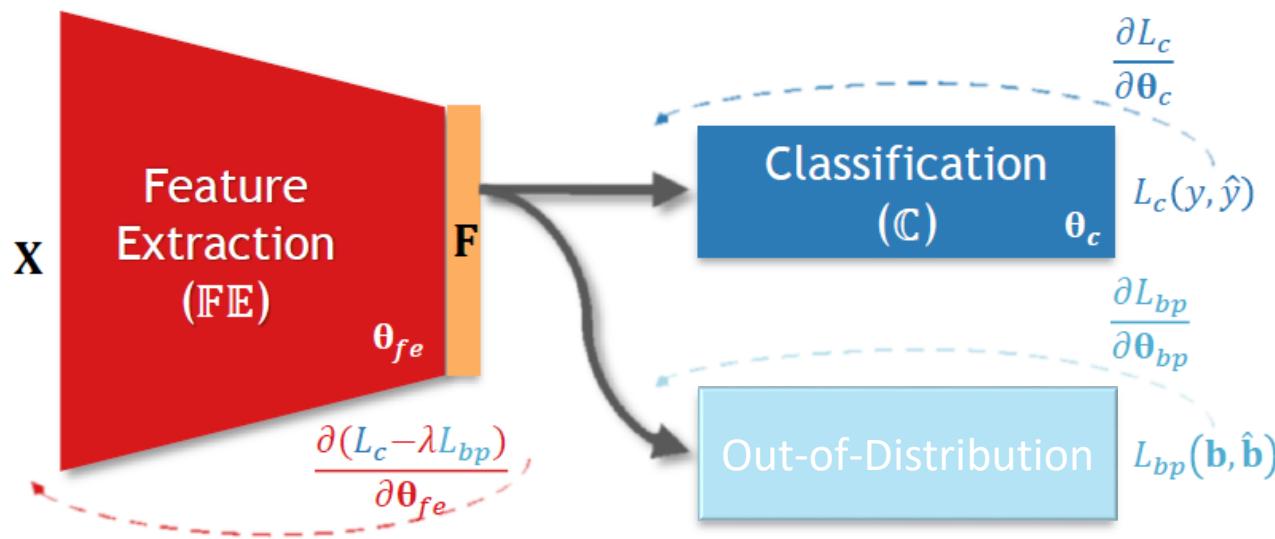
Timnit Gebru

Beware the biases!



<https://arxiv.org/pdf/1910.03676.pdf>

Available Internship



You have to like Maths (at least statistics) and possibly
Quebec...