# 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

50

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

# 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 recipes:

```
      Title: CHOCOLATE RANCH BARBECUE
Categories: Game, Casseroles, Cookies, Cookies
      Yield: 6 Servings

      2 tb Parmesan cheese -- chopped
      1 c  Coconut milk
      3    Eggs, beaten

Place each pasta over layers of lumps. Shape mixture into the moderate oven and simmer
until firm. Serve hot in bodied fresh, mustard, orange and cheese.

Combine the cheese and salt together the dough in a large skillet; add the ingredients
and stir in the chocolate and pepper.
```

**Source:** https://gist.github.com/nylki/1efbaa36635956d35bcc

51

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

# 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:

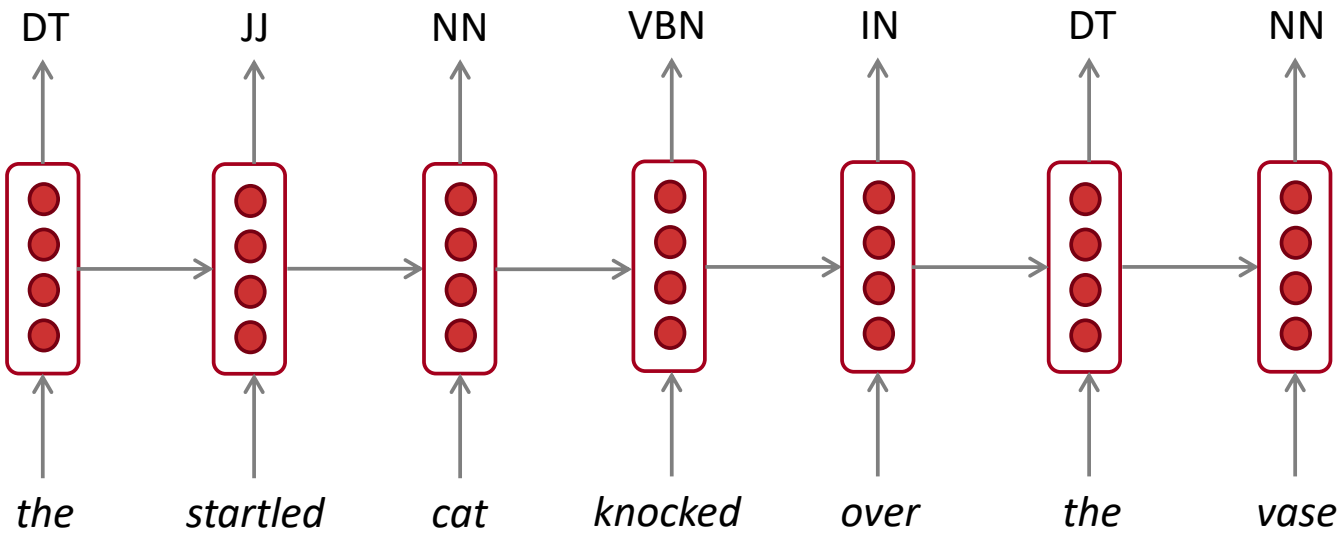
| | |
|---|---|
| Ghasty Pink 231 137 165 | Sand Dan 201 172 143 |
| Power Gray 151 124 112 | Grade Bat 48 94 83 |
| Navel Tan 199 173 140 | Light Of Blast 175 150 147 |
| Bock Coe White 221 215 236 | Grass Bat 176 99 108 |
| Horble Gray 178 181 196 | Sindis Poop 204 205 194 |
| Homestar Brown 133 104 85 | Dope 219 209 179 |
| Snader Brown 144 106 74 | Testing 156 101 106 |
| Golder Craam 237 217 177 | Stoner Blue 152 165 159 |
| Hurky White 232 223 215 | Burble Simp 226 181 132 |
| Burf Pink 223 173 179 | Stanky Bean 197 162 171 |
| Rose Hork 230 215 198 | 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

# RNNs can be used for tagging

e.g. part-of-speech tagging, named entity recognition

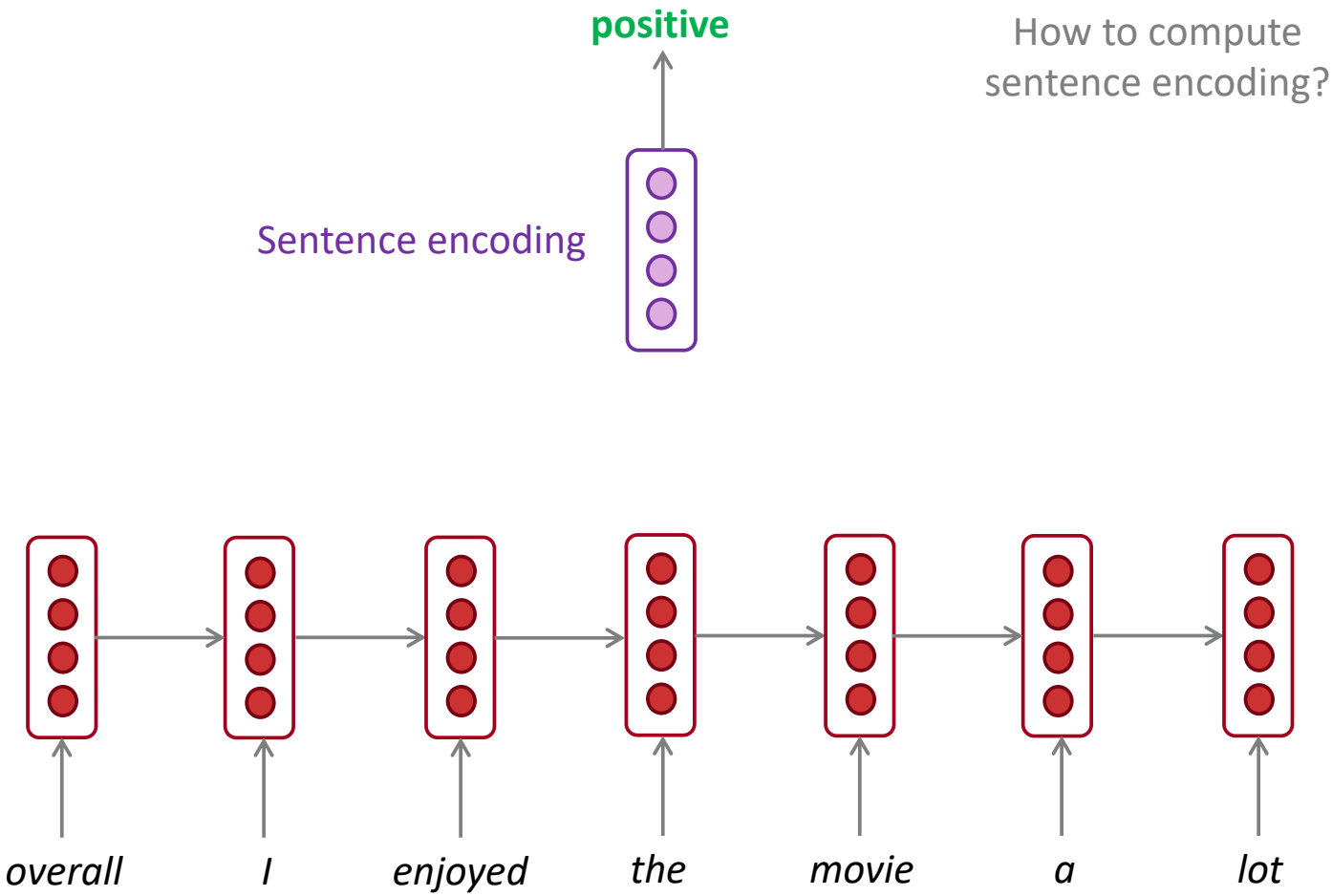



57

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

# RNNs can be used for sentence classification

e.g. sentiment classification

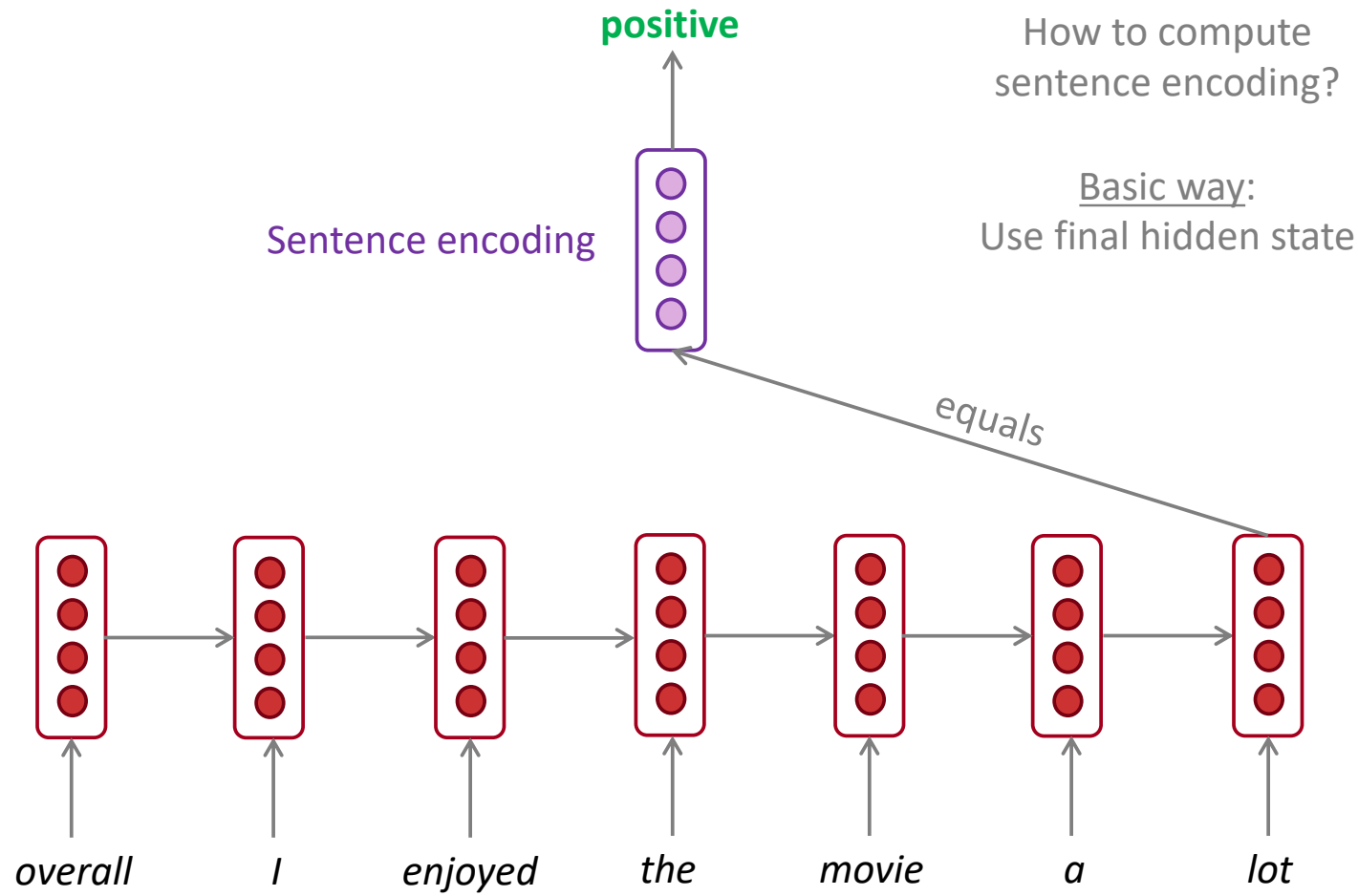



58

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

# RNNs can be used for sentence classification

e.g. sentiment classification

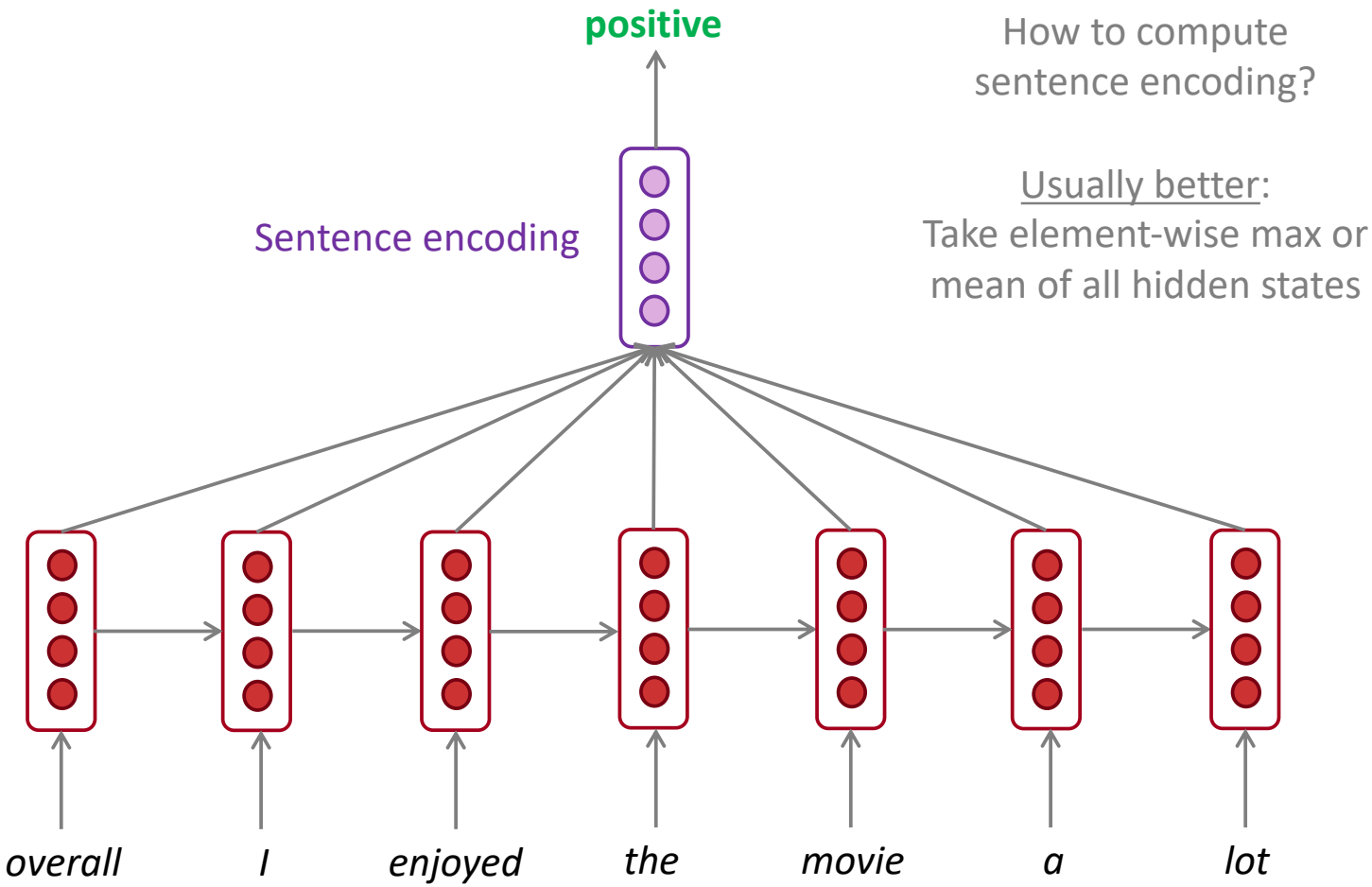



59

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

# RNNs can be used for sentence classification

e.g. sentiment classification

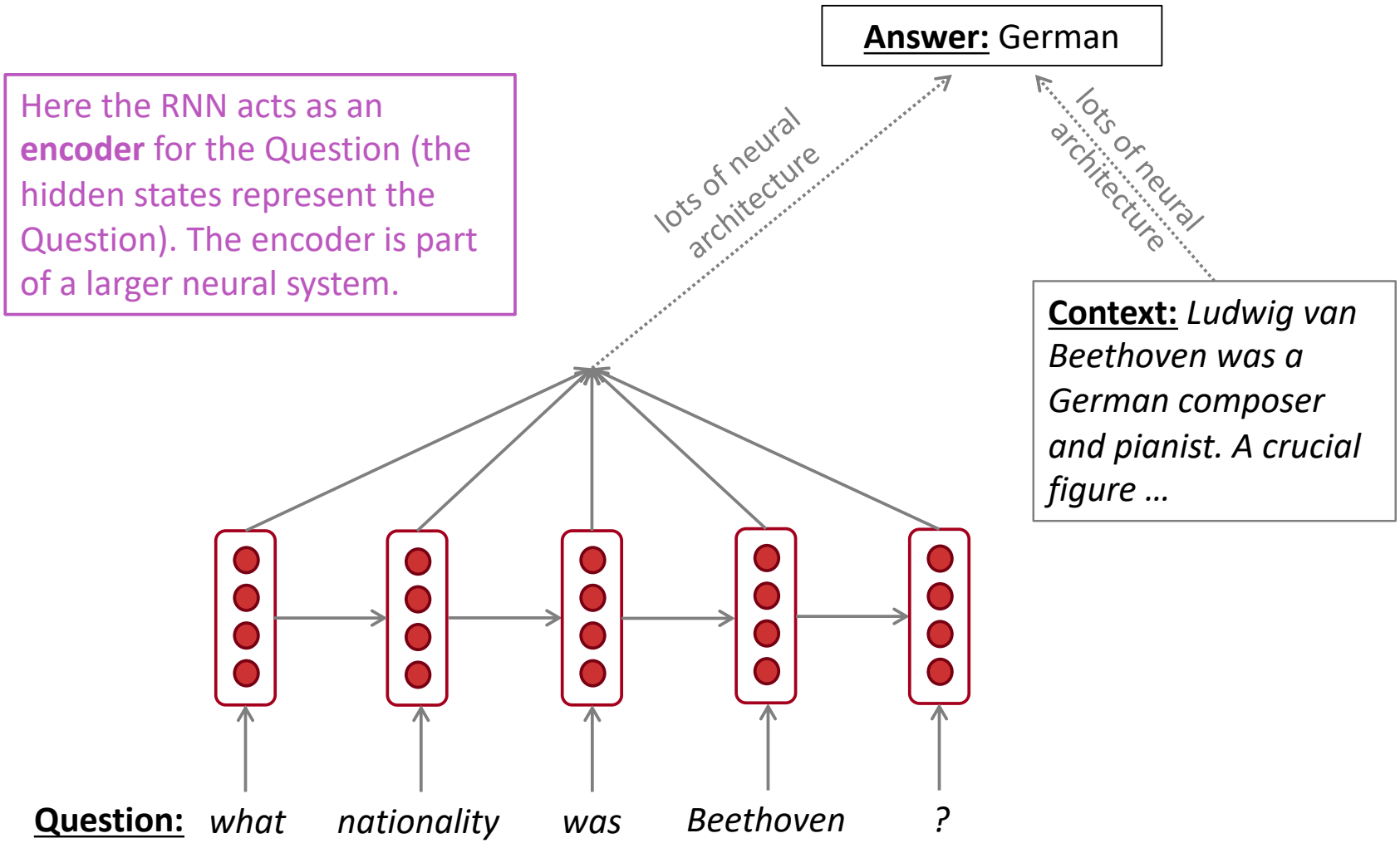



60

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

# RNNs can be used as an encoder module

e.g. question answering, machine translation, *many other tasks!*

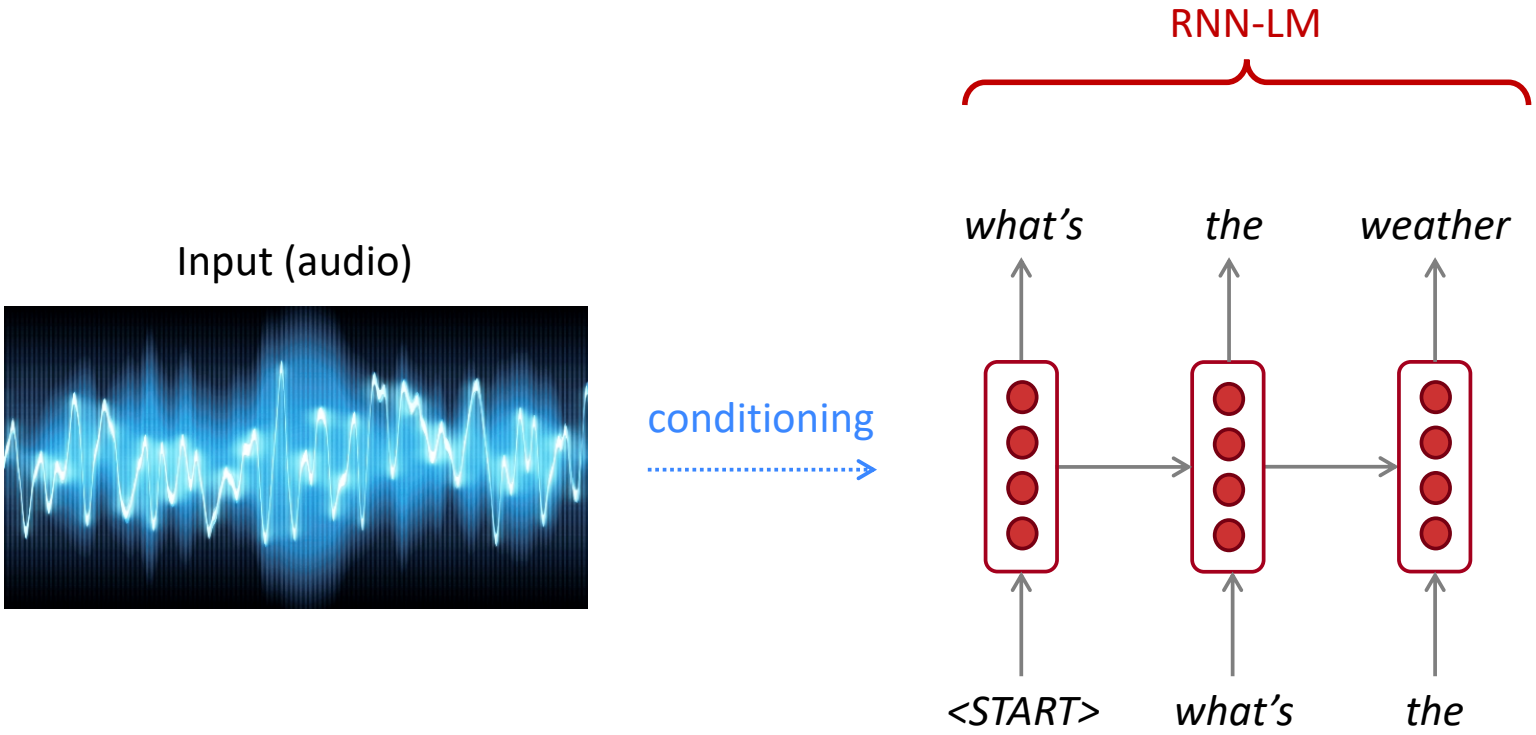



61

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

# RNN-LMs can be used to generate text

e.g. speech recognition, machine translation, summarization




This is an example of a *conditional language model*.
We'll see Machine Translation in much more detail later.

62

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

# A note on terminology

The RNN described in this lecture = simple/vanilla/Elman RNN

**Next lecture:** You will learn about other RNN flavors

like GRU  and LSTM  and multi-layer RNNs 

**By the end of the course:** You will understand phrases like
*"stacked bidirectional LSTM with residual connections and self-attention"*

63

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

# Content

# Evaluating Language Models

- The standard evaluation metric for Language Models is perplexity.

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

Normalized by number of words

Inverse probability of corpus, according to Language Model

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

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

**Lower** perplexity is better!

53

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

# RNNs have greatly improved perplexity

*n*-gram model →

Increasingly complex RNNs ↓

| Model | Perplexity |
|---|---|
| Interpolated Kneser-Ney 5-gram (Chelba et al., 2013) | 67.6 |
| RNN-1024 + MaxEnt 9-gram (Chelba et al., 2013) | 51.3 |
| RNN-2048 + BlackOut sampling (Ji et al., 2015) | 68.3 |
| Sparse Non-negative Matrix factorization (Shazeer et al., 2015) | 52.9 |
| LSTM-2048 (Jozefowicz et al., 2016) | 43.7 |
| 2-layer LSTM-8192 (Jozefowicz et al., 2016) | 30 |
| **Ours small** (LSTM-2048) | 43.9 |
| **Ours large** (2-layer LSTM-2048) | 39.8 |

↓ Perplexity improves (lower is better)

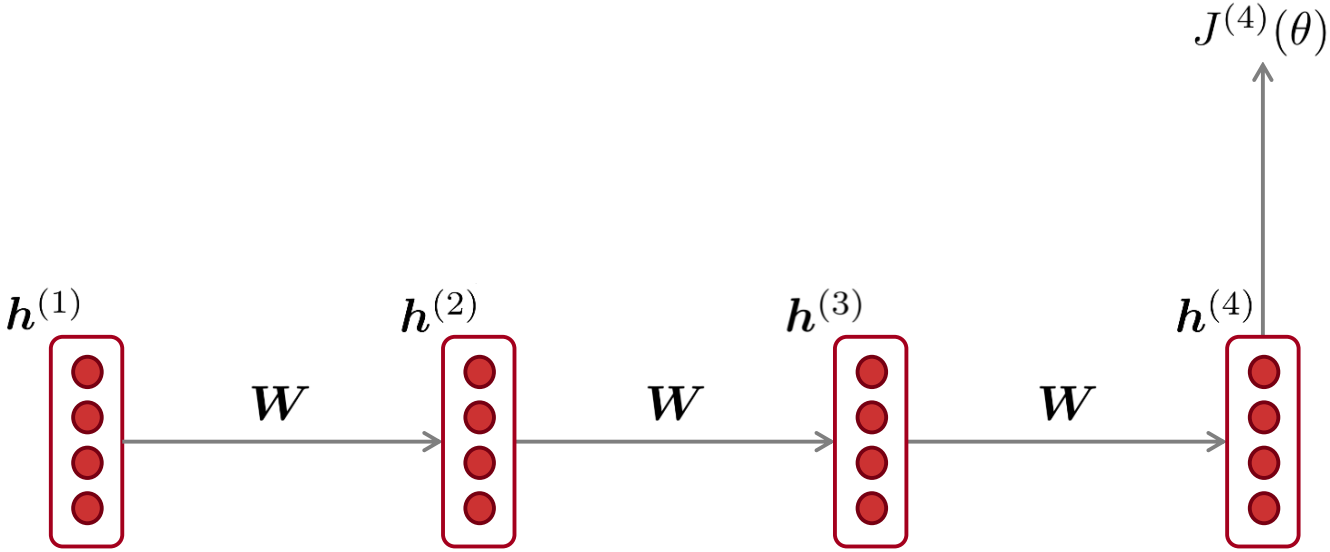
**Source:** https://research.fb.com/building-an-efficient-neural-language-model-over-a-billion-words/



Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

# Content

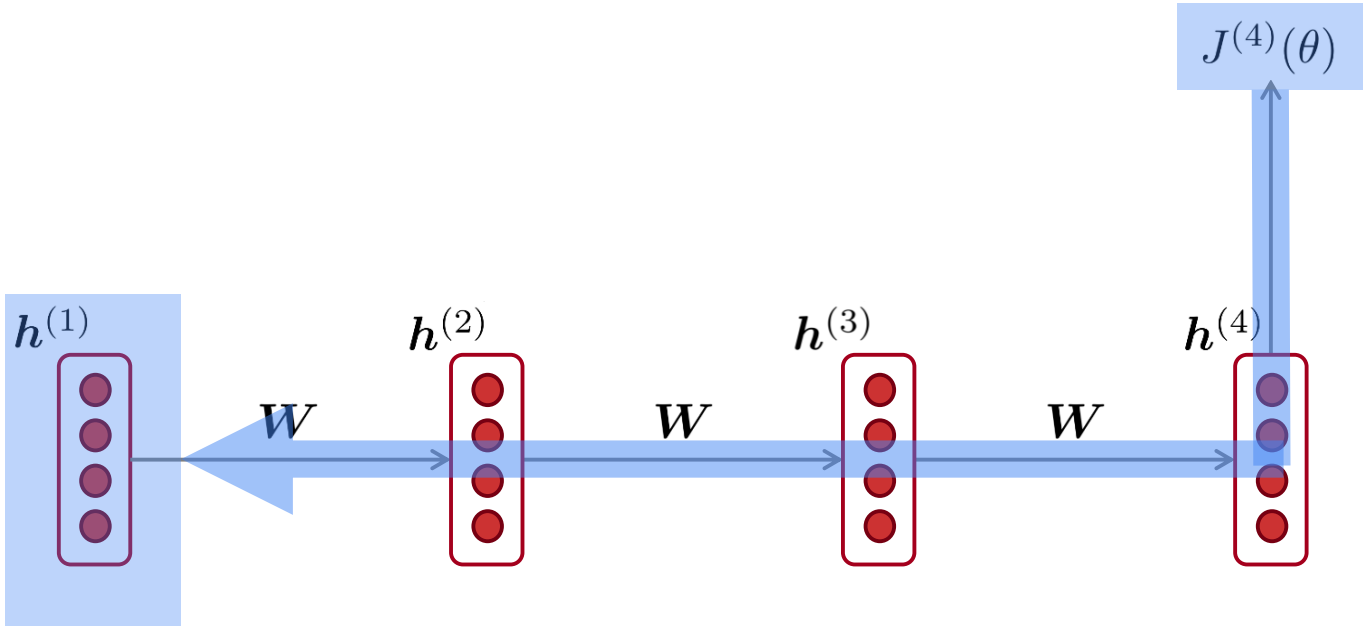
# Vanishing gradient intuition




6

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

# Vanishing gradient intuition

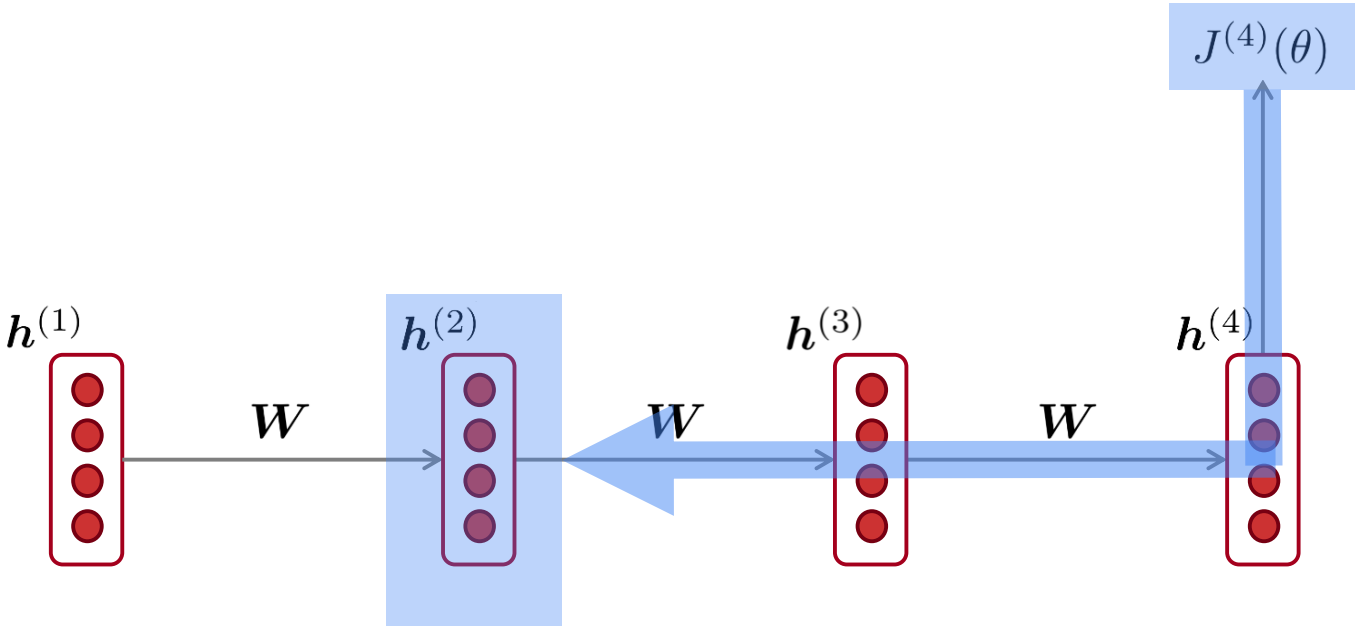



$$\frac{\partial J^{(4)}}{\partial h^{(1)}} = \ ?$$

7

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

# Vanishing gradient intuition



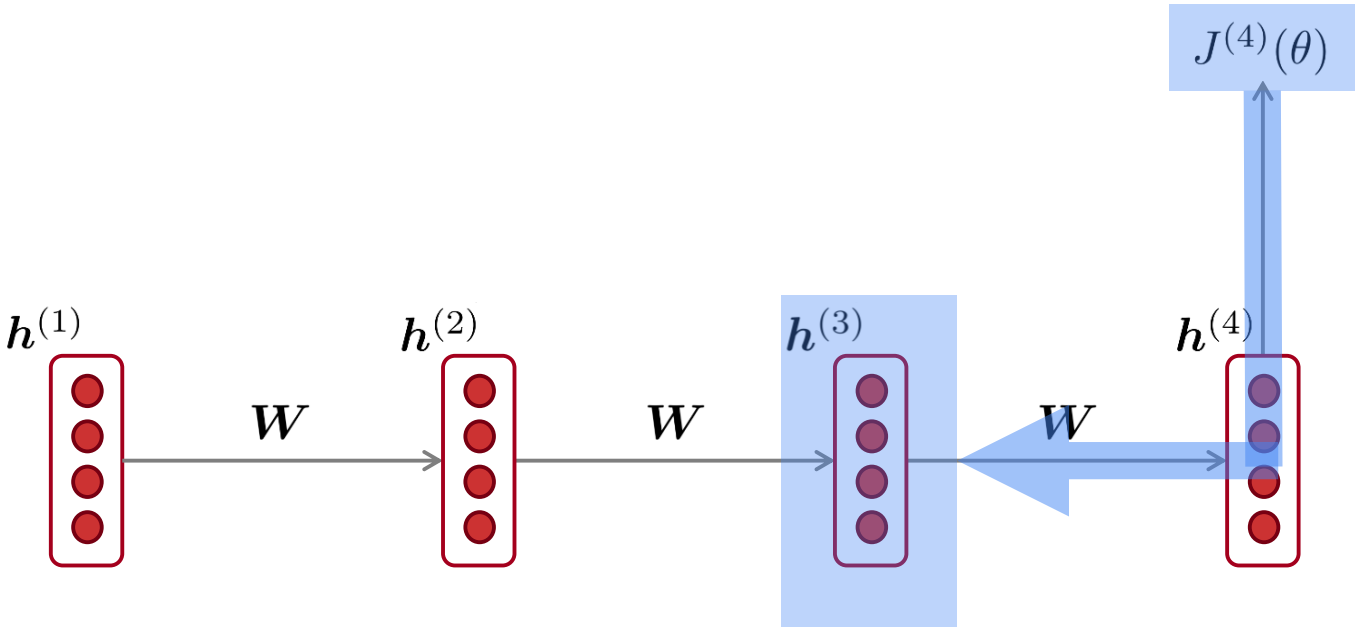

$$\frac{\partial J^{(4)}}{\partial h^{(1)}} = \frac{\partial h^{(2)}}{\partial h^{(1)}} \times \frac{\partial J^{(4)}}{\partial h^{(2)}}$$

chain rule!

8

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

# Vanishing gradient intuition



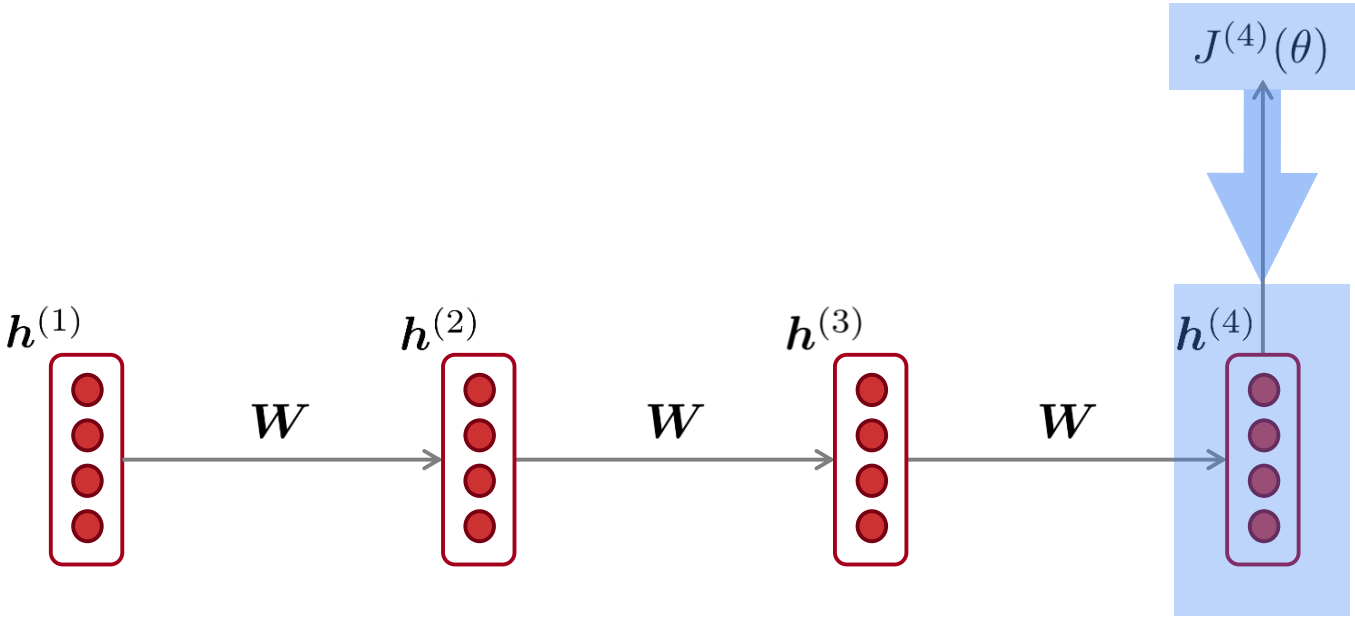

$$\frac{\partial J^{(4)}}{\partial h^{(1)}} = \frac{\partial h^{(2)}}{\partial h^{(1)}} \times \frac{\partial h^{(3)}}{\partial h^{(2)}} \times \frac{\partial J^{(4)}}{\partial h^{(3)}}$$

chain rule!

9

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

# Vanishing gradient intuition



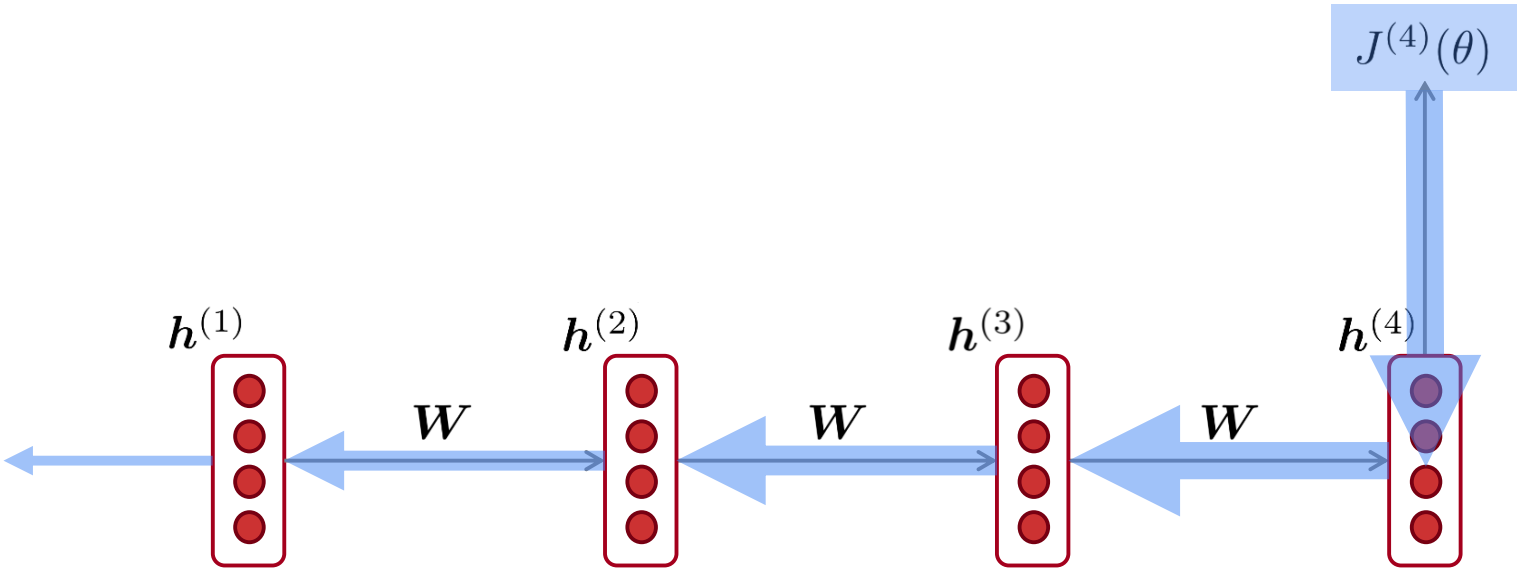

$$\frac{\partial J^{(4)}}{\partial h^{(1)}} = \frac{\partial h^{(2)}}{\partial h^{(1)}} \times \frac{\partial h^{(3)}}{\partial h^{(2)}} \times \frac{\partial h^{(4)}}{\partial h^{(3)}} \times \frac{\partial J^{(4)}}{\partial h^{(4)}}$$

chain rule!

10

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

# Vanishing gradient intuition




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

What happens if these are small?

Vanishing gradient problem: When these are small, the gradient signal gets smaller and smaller as it backpropagates further

11

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

# Vanishing gradient proof sketch

- Recall: $$\boldsymbol{h}^{(t)} = \sigma\left(\boldsymbol{W}_h \boldsymbol{h}^{(t-1)} + \boldsymbol{W}_x \boldsymbol{x}^{(t)} + \boldsymbol{b}_1\right)$$

- Therefore: $$\frac{\partial \boldsymbol{h}^{(t)}}{\partial \boldsymbol{h}^{(t-1)}} = \operatorname{diag}\left(\sigma'\left(\boldsymbol{W}_h \boldsymbol{h}^{(t-1)} + \boldsymbol{W}_x \boldsymbol{x}^{(t)} + \boldsymbol{b}_1\right)\right) \boldsymbol{W}_h$$ (chain rule)

- Consider the gradient of the loss $J^{(i)}(\theta)$ on step *i*, with respect to the hidden state $\boldsymbol{h}^{(j)}$ on some previous step *j*.

$$\frac{\partial J^{(i)}(\theta)}{\partial \boldsymbol{h}^{(j)}} = \frac{\partial J^{(i)}(\theta)}{\partial \boldsymbol{h}^{(i)}} \prod_{j<t\leq i} \frac{\partial \boldsymbol{h}^{(t)}}{\partial \boldsymbol{h}^{(t-1)}}$$ (chain rule)

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

If $W_h$ is small, then this term gets vanishingly small as *i* and *j* get further apart

12

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

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

# Vanishing gradient proof sketch

- Consider matrix L2 norms:

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

- Pascanu et al showed that that if the largest eigenvalue of $W_h$ is less than 1, then the gradient $\left\|\frac{\partial J^{(i)}(\theta)}{\partial \boldsymbol{h}^{(j)}}\right\|$ will shrink exponentially
  - Here the bound is 1 because we have sigmoid nonlinearity

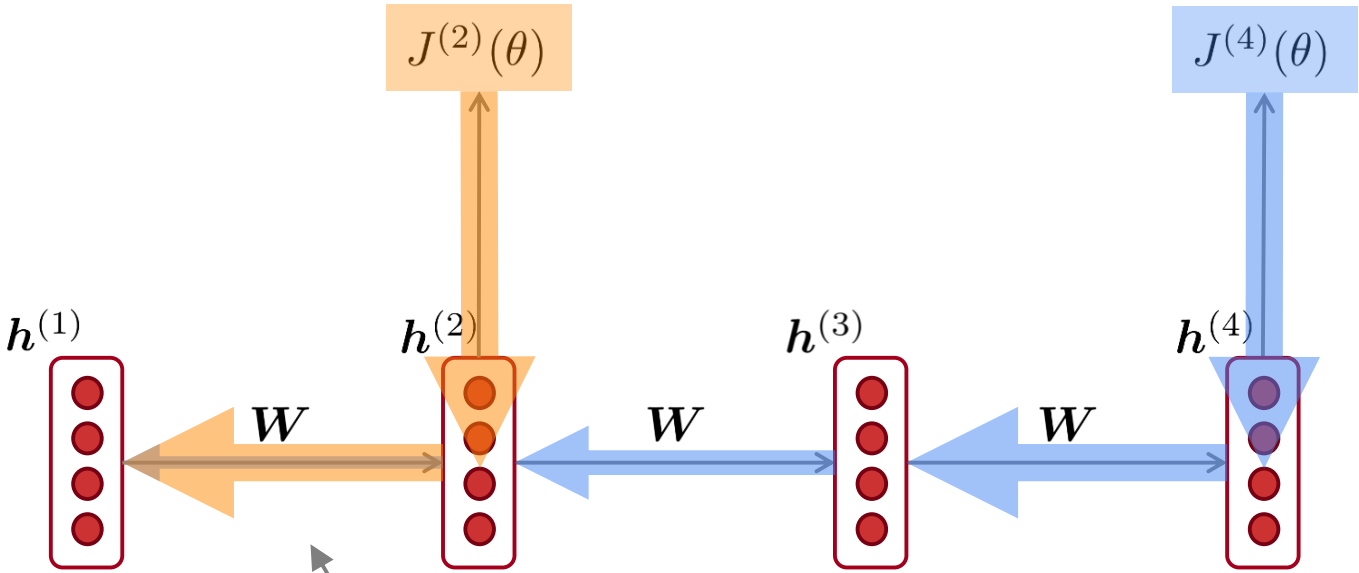
- There's a similar proof relating a largest eigenvalue >1 to exploding gradients

13

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

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

# 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.

14

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

# Why is vanishing gradient a problem?

- Another explanation: Gradient can be viewed as a measure of *the effect of the past on the future*

- If the gradient becomes vanishingly small over longer distances (step *t* to step *t+n*), then we can't tell whether:
  1. There's no dependency between step *t* and *t+n* in the data
  2. We have wrong parameters to capture the true dependency between *t* and *t+n*

15

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

# Effect of vanishing gradient on RNN-LM

- **LM task:** *When she tried to print her tickets, 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 _____*

- To learn from this training example, the RNN-LM needs to model the dependency between *"tickets"* on the 7$^{th}$ step and the target word *"tickets"* at the end.

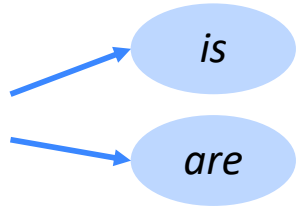
- But if gradient is small, the model can't learn this dependency
  - So the model is unable to predict similar long-distance dependencies at test time

16

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

# 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]

17

"Assessing the Ability of LSTMs to Learn Syntax-Sensitive Dependencies", Linzen et al, 2016. https://arxiv.org/pdf/1611.01368.pdf

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

# Why is exploding gradient a problem?

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

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

- 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)

18

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

# 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** $\|\hat{\mathbf{g}}\| \geq threshold$ **then**
  $\hat{\mathbf{g}} \leftarrow \frac{threshold}{\|\hat{\mathbf{g}}\|}\hat{\mathbf{g}}$
**end if**

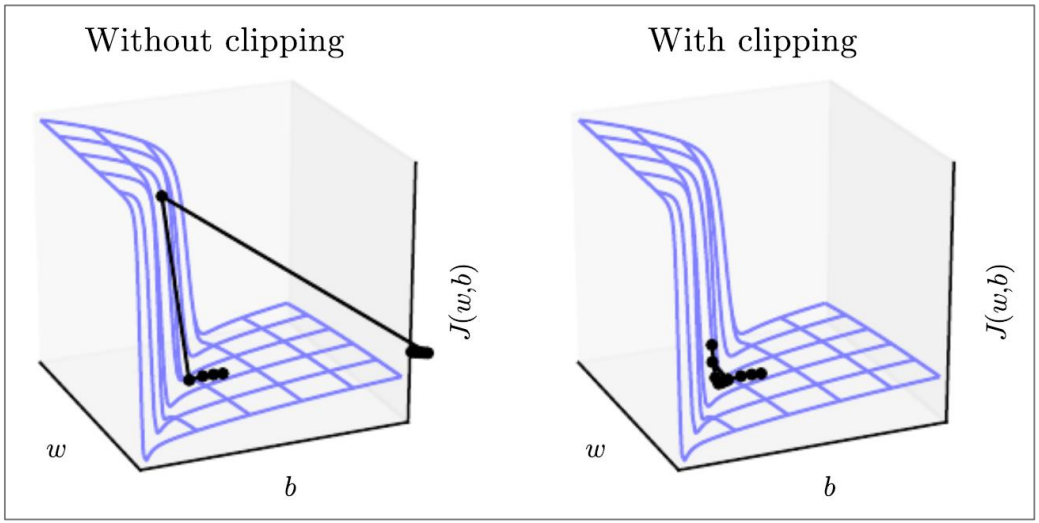
- Intuition: take a step in the same direction, but a smaller step

19

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

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

# Gradient clipping: solution for exploding gradient




- This shows the loss surface of a simple RNN (hidden state is a scalar not a vector)
- The "cliff" is dangerous because it has steep gradient
- On the left, gradient descent takes two very big steps due to steep gradient, resulting in climbing the cliff then shooting off to the right (both bad updates)
- On the right, gradient clipping reduces the size of those steps, so effect is less drastic

20

**Source:** "Deep Learning", Goodfellow, Bengio and Courville, 2016. Chapter 10.11.1. https://www.deeplearningbook.org/contents/rnn.html

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

# 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

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

- How about a RNN with separate memory?

21

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

# Content

# Long Short-Term Memory (LSTM)

- A type of RNN proposed by Hochreiter and Schmidhuber in 1997 as a solution to the vanishing gradients problem.

- On step *t*, there is a hidden state $\boldsymbol{h}^{(t)}$ *and* a cell state $\boldsymbol{c}^{(t)}$
  - Both are vectors length *n*
  - The cell stores long-term information
  - The LSTM can erase, write and read information from the cell

- The selection of which information is erased/written/read is controlled by three corresponding gates
  - The gates are also vectors length *n*
  - On each timestep, each element of the gates can be open (1), closed (0), or somewhere in-between.
  - The gates are dynamic: their value is computed based on the current context

22

"Long short-term memory", Hochreiter and Schmidhuber, 1997. https://www.bioinf.jku.at/publications/older/2604.pdf

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

# Long Short-Term Memory (LSTM)

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

**Sigmoid function**: all gate values are between 0 and 1

**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

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

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

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

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

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

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

All these are vectors of same length $n$

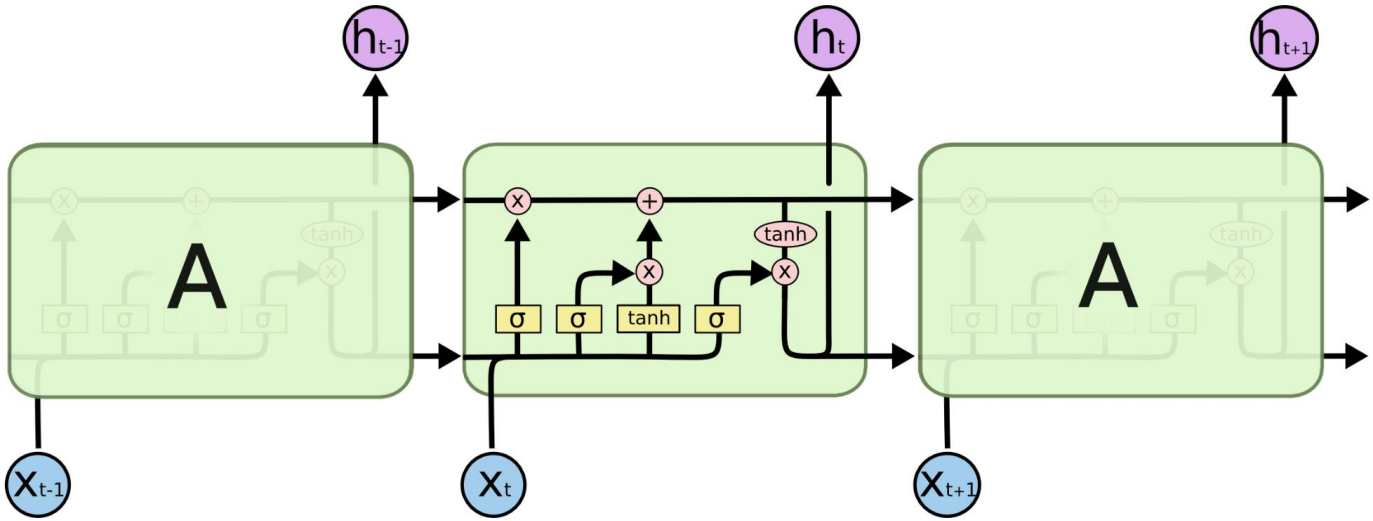
Gates are applied using element-wise product

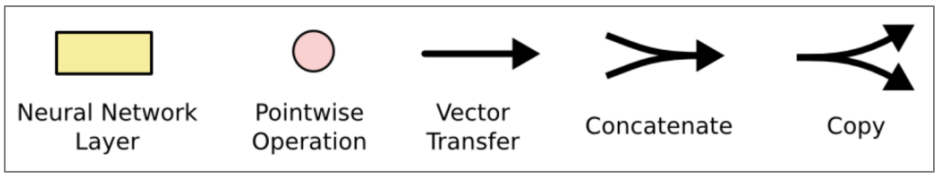
23

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

# Long Short-Term Memory (LSTM)

You can think of the LSTM equations visually like this:



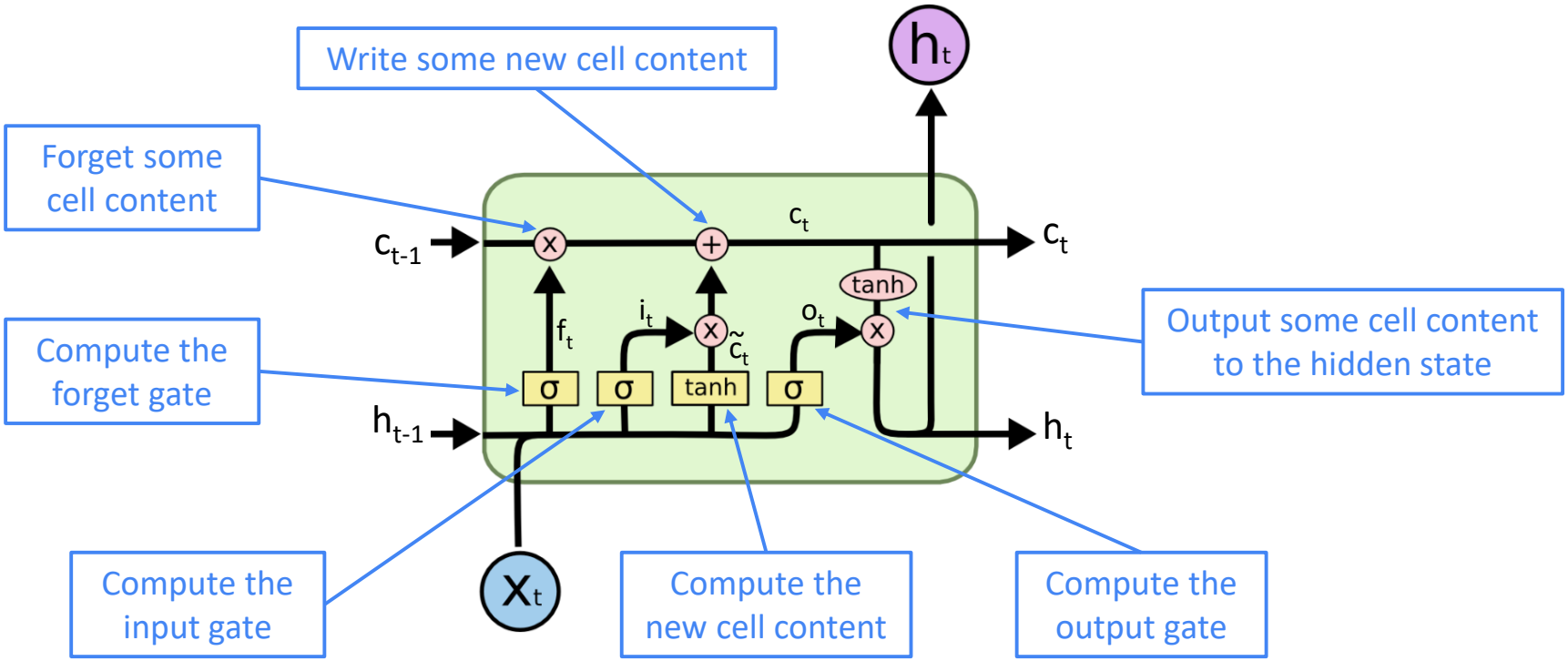

24

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

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

# Long Short-Term Memory (LSTM)

You can think of the LSTM equations visually like this:




25

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

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

# How does LSTM solve vanishing gradients?

- The LSTM architecture makes it easier for the RNN to preserve information over many timesteps
  - e.g. if the forget gate is set to remember everything on every timestep, then the info in the cell is preserved indefinitely
  - By contrast, it's harder for vanilla RNN to learn a recurrent weight matrix $W_h$ that preserves info in hidden state

- LSTM doesn't *guarantee* that there is no vanishing/exploding gradient, but it does provide an easier way for the model to learn long-distance dependencies

26

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

# LSTMs: real-world success

- In 2013-2015, LSTMs started achieving state-of-the-art results
  - Successful tasks include: handwriting recognition, speech recognition, machine translation, parsing, image captioning
  - LSTM became the dominant approach

- Now (2019), other approaches (e.g. Transformers) have become more dominant for certain tasks.
  - For example in **WMT** (a MT conference + 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

27

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

# Content

# 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 $\boldsymbol{x}^{(t)}$ and hidden state $\boldsymbol{h}^{(t)}$ (no cell state).

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

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

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

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

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

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

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

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

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

28 "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation", Cho et al. 2014, https://arxiv.org/pdf/1406.1078v3.pdf

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

# 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

29

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

# Content

# 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

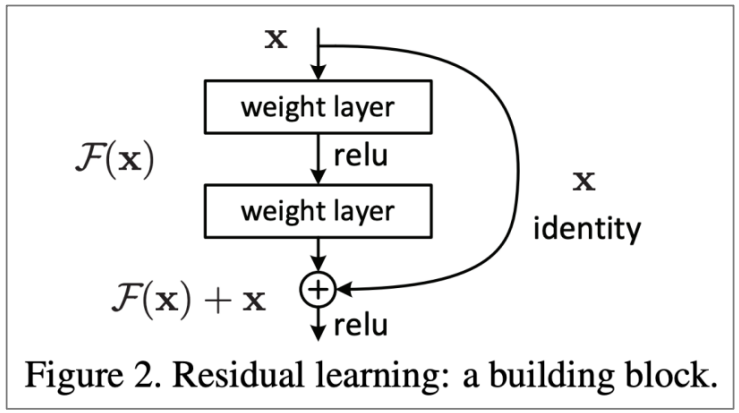


Figure 2. Residual learning: a building block.

"Deep Residual Learning for Image Recognition", He et al, 2015. https://arxiv.org/pdf/1512.03385.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:

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

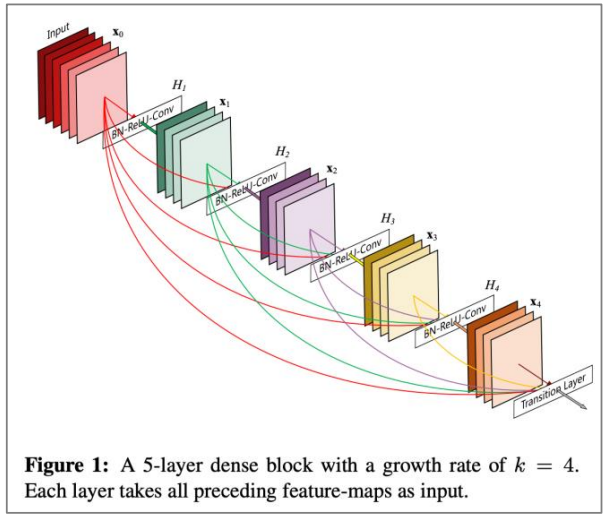



**Figure 1:** A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

”Densely Connected Convolutional Networks", Huang et al, 2017. https://arxiv.org/pdf/1608.06993.pdf

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

# 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:

- Highway connections aka “HighwayNet”
- Similar to residual connections, but the identity connection vs the transformation layer is controlled by a dynamic gate
- Inspired by LSTMs, but applied to deep feedforward/convolutional networks

32

”Highway Networks", Srivastava et al, 2015. https://arxiv.org/pdf/1505.00387.pdf

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

# 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)

33

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

# 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)

- Conclusion: Though vanishing/exploding gradients are a general problem, RNNs are particularly unstable due to the repeated multiplication by the same weight matrix [Bengio et al, 1994]

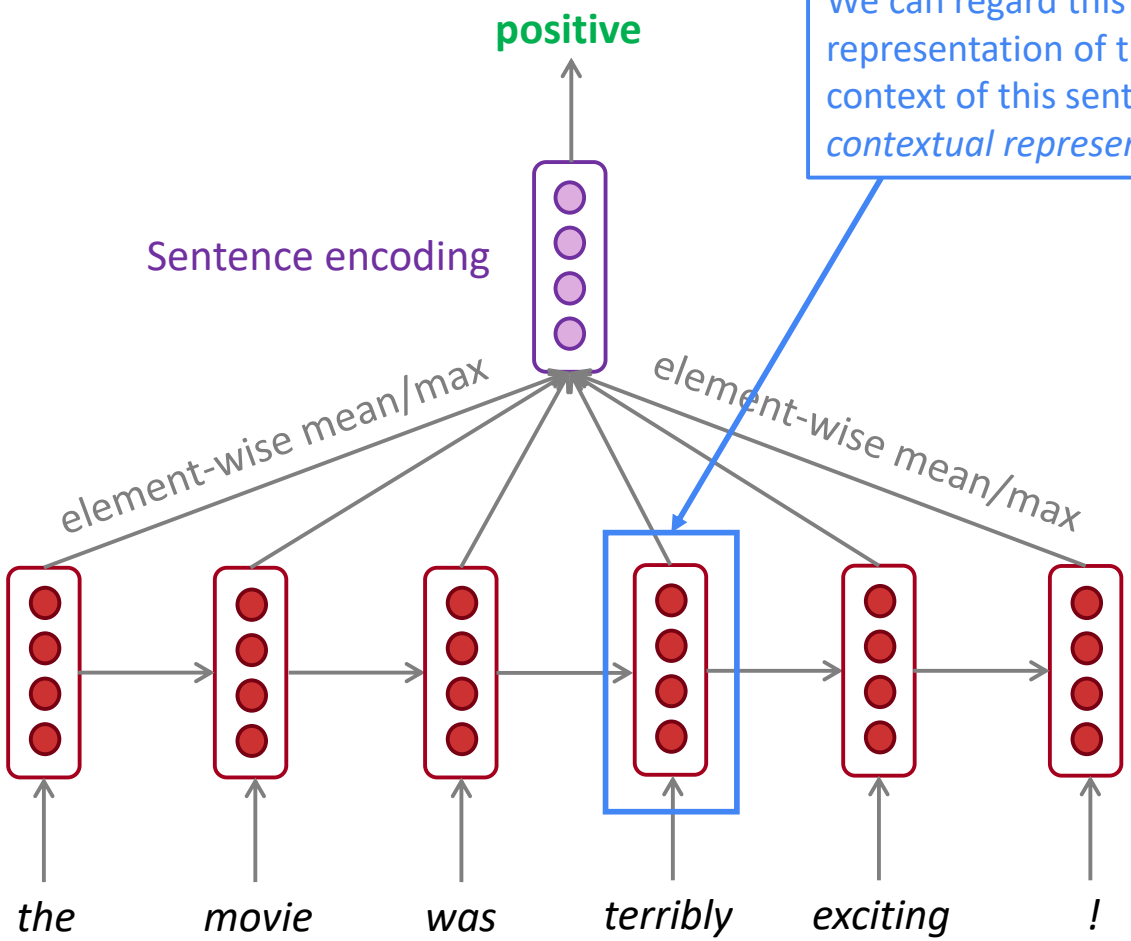
34 "Learning Long-Term Dependencies with Gradient Descent is Difficult", Bengio et al. 1994, http://ai.dinfo.unifi.it/paolo//ps/tnn-94-gradient.pdf

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

# Content

# 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"*).

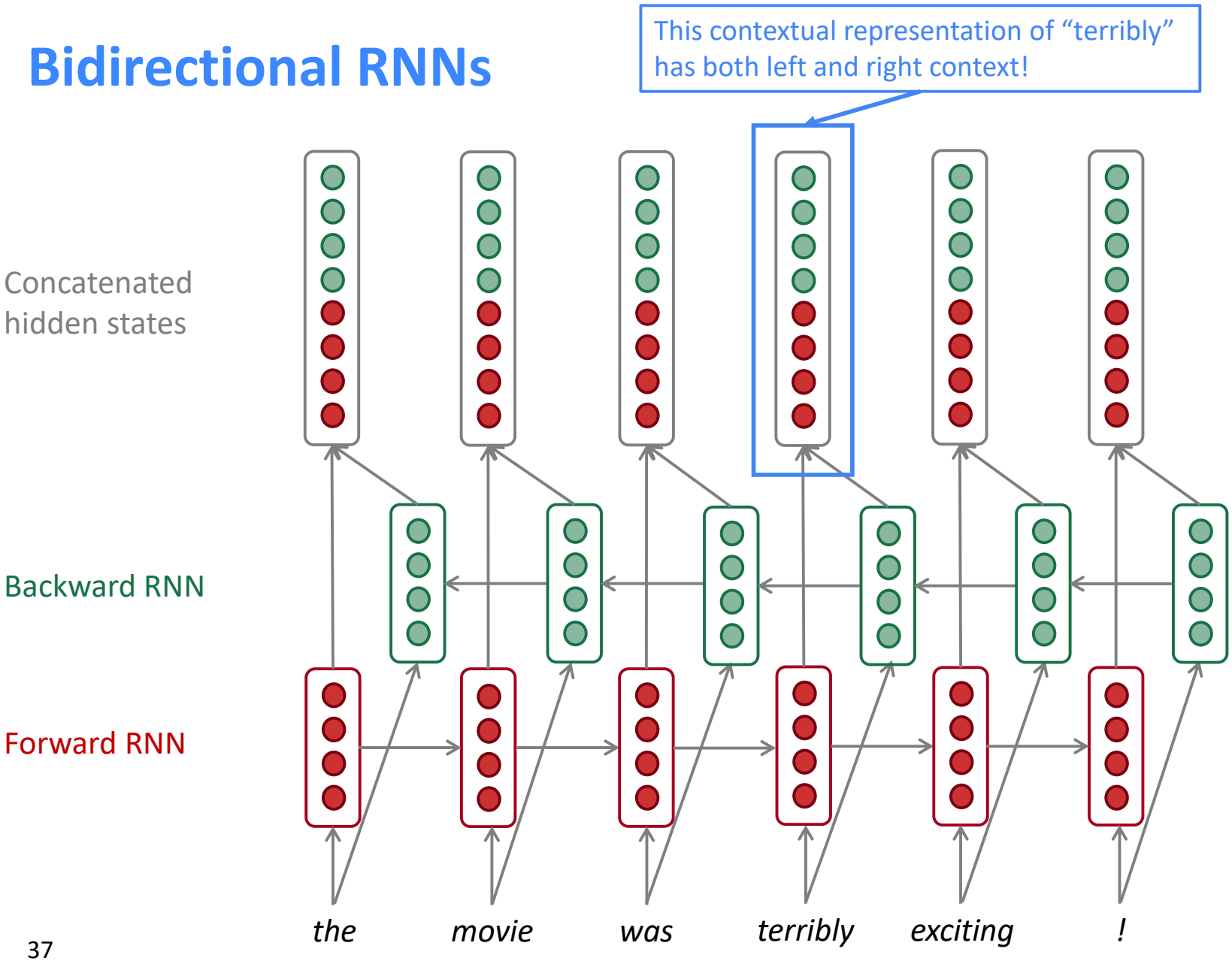
**What about *right* context?**

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



Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

# Bidirectional RNNs




Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n

# 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 $\overrightarrow{\boldsymbol{h}}^{(t)} = \mathrm{RNN}_{\mathrm{FW}}(\overrightarrow{\boldsymbol{h}}^{(t-1)}, \boldsymbol{x}^{(t)})$

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

Generally, these two RNNs have separate weights

Concatenated hidden states $\boldsymbol{h}^{(t)} = [\overrightarrow{\boldsymbol{h}}^{(t)}; \overleftarrow{\boldsymbol{h}}^{(t)}]$

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

38

Christopher Manning, Natural Language Processing with Deep Learning, Standford U. CS224n