Classical Language Models

Antoine Bosselut

Announcements

- **Assignment 1 Released:** Due **March 17, 2024**
  - Will go online in the early afternoon. No need to check during class!
  - **Q&A Sessions:**
    - Wednesday, March 6th, 2024 - 1 PM (STCC)
    - Thursday, March 14th, 2024 - 1 PM (CE 1 6)
- **Quick Poll:** How many of you are first-year Masters students ?
  - Plan for lecture on March 13th

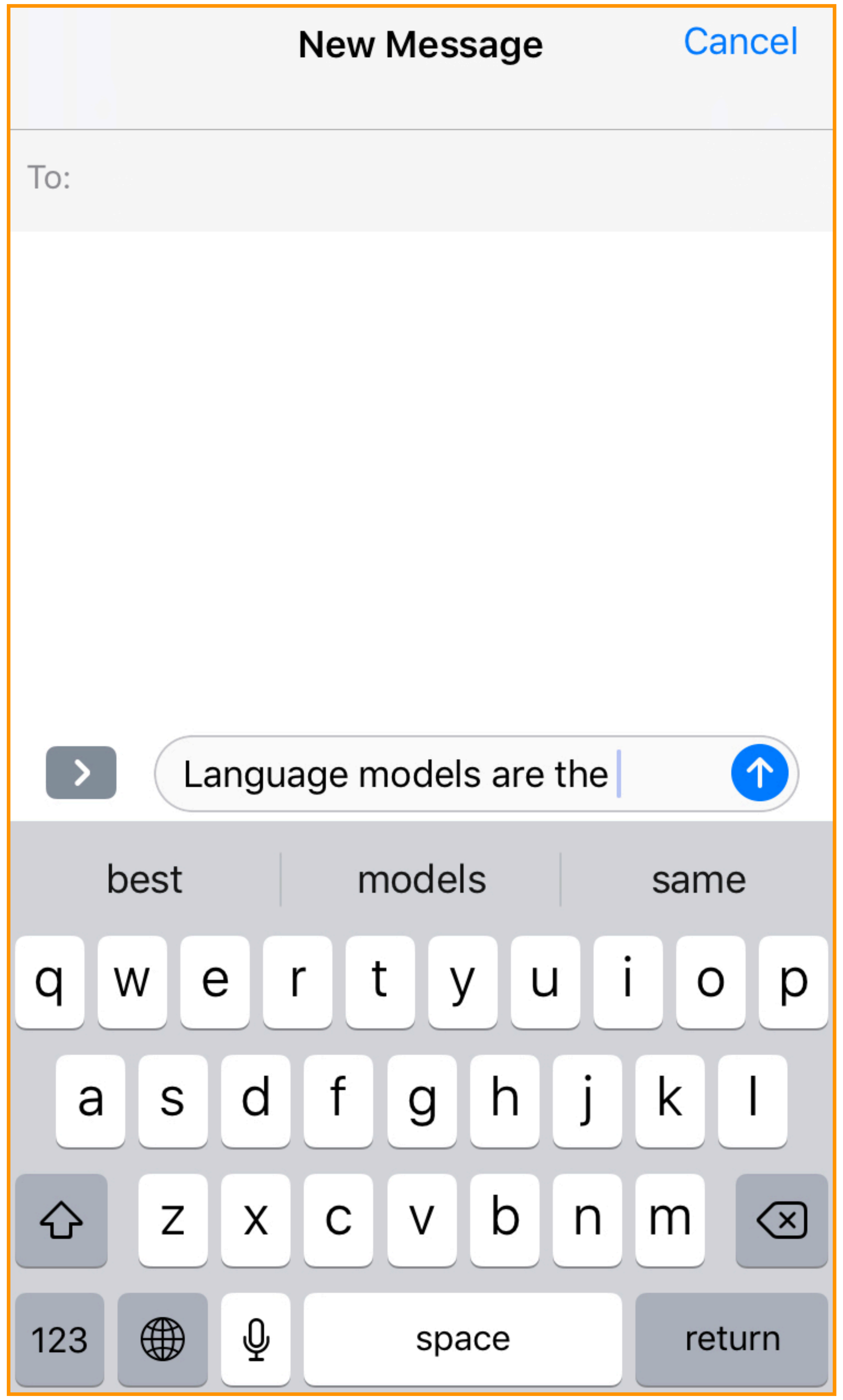
Some slides adapted from Danqi Chen, Mohit Iyyer, Dan Jurafsky

Today's Outline

- **Part 1:** Language models introduction, count-based language models, evaluating language models, smoothing
- **Part 2:** Fixed-context Language Models, Recurrent Neural Networks 1

Some slides adapted from Danqi Chen, Mohit Iyyer, Dan Jurafsky

Language models are ubiquitous



Applications of LMs

- Predicting words is important in many situations
  - **Classical:** Machine translation, text generation

$$P(\text{a } \textbf{smooth} \text{ finish}) > P(\text{a } \textbf{flat} \text{ finish})$$

  - **Classical:** Speech recognition/Spell checking

$$P(\text{high school } \textbf{principal}) > P(\text{high school } \textbf{principle})$$

  - **Modern:** Information extraction, Question answering, Classification, etc.

What is a language model?

- A language model is a **probabilistic model of a sequence of tokens**
  - How likely is a given phrase/sentence/paragraph/document?

- A sequence is modelled as a joint distribution of tokens $w_1, w_2, w_3, \ldots, w_n$:

$$P(w_1, w_2, w_3, ..., w_n)$$

- Language models estimate this probability!

Chain rule

- How should we compute this joint probability over words in a sequence?

$$P(w_1, w_2, w_3, \ldots, w_N) = P(w_1) \times P(w_2 \mid w_1) \times P(w_3 \mid w_1, w_2) \times \ldots \times P(w_N \mid w_1, w_2, \ldots, w_{N-1})$$

$$P(w_1, w_2, w_3, \ldots, w_N) = \prod_i P(w_i \mid \boxed{w_1, w_2, \ldots, w_{i-1}}) \Rightarrow \text{“prefix”}$$

Chain rule

- How should we compute this joint probability over words in a sequence?

$$P(w_1, w_2, w_3, \ldots, w_N) = P(w_1) \times P(w_2 \mid w_1) \times P(w_3 \mid w_1, w_2) \\ \times \ldots \times P(w_N \mid w_1, w_2, \ldots, w_{N-1})$$

$$P(w_1, w_2, w_3, \ldots, w_N) = \prod_i P(w_i \mid \boxed{w_1, w_2, \ldots, w_{i-1}}) \rightarrow \text{“prefix”}$$

Sentence: “the cat sat on the mat”

$$P(\text{the cat sat on the mat}) = P(\text{the}) * P(\text{cat}|\text{the}) * P(\text{sat}|\text{the cat}) \\ *P(\text{on}|\text{the cat sat}) * P(\text{the}|\text{the cat sat on}) \\ *P(\text{mat}|\text{the cat sat on the})$$

Count-based Modeling

Maximum likelihood estimate (MLE)

$$P(\text{sat}|\text{the cat}) = \frac{\text{count}(\text{the cat sat})}{\text{count}(\text{the cat})}$$

$$P(\text{on}|\text{the cat sat}) = \frac{\text{count}(\text{the cat sat on})}{\text{count}(\text{the cat sat})}$$

$$\vdots$$

Note on Notation:
Word type: unique word in our vocabulary V
Token: instance of a word type in our corpus

- Estimate probabilities by counting occurrences of sequences in a **corpus** of text

Question

What's going to be a problem with counting sequences for any prefix?

Count-based Modeling

Maximum likelihood estimate (MLE)

$$P(\text{sat}|\text{the cat}) = \frac{\text{count}(\text{the cat sat})}{\text{count}(\text{the cat})}$$

$$P(\text{on}|\text{the cat sat}) = \frac{\text{count}(\text{the cat sat on})}{\text{count}(\text{the cat sat})}$$

$$\vdots$$

Note on Notation:
Word type: unique word in our vocabulary *V*
Token: instance of a word type in our corpus

- Estimate probabilities by counting occurrences of sequences in a **corpus** of text
- With a vocabulary of size *V*,
  - number of sequences of length n = V^n
- Typical vocabulary ≈ 40000 words
  - even sentences of length <= 11 results in more than $4 * 10^{50}$ sequences! *(more than the number of atoms in the earth)*

Markov assumption

- Use only the recent past to predict the next word

- **Effect:** reduce the number of estimated parameters in exchange for modeling capacity

- 1st order Markov

$$P(\mathrm{mat}|\mathrm{the\ cat\ sat\ on\ the}) \approx P(\mathrm{mat}|\mathrm{the})$$

- 2nd order Markov

$$P(\mathrm{mat}|\mathrm{the\ cat\ sat\ on\ the}) \approx P(\mathrm{mat}|\mathrm{on\ the})$$

kth order Markov

- Consider only the last *k* words for context

$$P(w_i \mid w_1 w_2 \ldots w_{i-1}) \approx P(w_i \mid w_{i-k} \ldots w_{i-1})$$

which implies the probability of a sequence is:

$$P(w_1 w_2 \ldots w_n) \approx \prod_i P(w_i \mid w_{i-k} \ldots w_{i-1})$$

(ignore $w_j \quad \forall j < 0$)

n-gram models

Unigram

$$P(w_1, w_2, ...w_n) = \prod_{i=1}^{n} P(w_i)$$

Bigram

$$P(w_1, w_2, ...w_n) = \prod_{i=1}^{n} P(w_i|w_{i-1})$$

and Trigram, 4-gram, and so on.

Larger n leads to more accurate and better language models*
(but also higher costs)

**Caveat: Assuming infinite data!*

Examples

Unigram	*release millions See ABC accurate President of Donald Will cheat them a CNN megynkelly experience @ these word out- the*
Bigram	*Thank you believe that @ ABC news, Mississippi tonight and the false editorial I think the great people Bill Clinton . ''*
Trigram	*We are going to MAKE AMERICA GREAT AGAIN! #MakeAmericaGreatAgain https: //t.co/DjkdAzT3WV*

$$\arg \max_{(w_1, w_2, \ldots, w_n)} P(w_1, w_2, \ldots, w_n) = \arg \max_{(w_1, w_2, \ldots, w_n)} \prod_{i=1}^{n} P(w_i \,|\, w_{i-k}, \ldots, w_{i-1})$$

Examples

Unigram	*release millions See ABC accurate President of Donald Will cheat them a CNN megynkelly experience @ these word out- the*
Bigram	*Thank you believe that @ ABC news, Mississippi tonight and the false editorial I think the great people Bill Clinton . ''*
Trigram	*We are going to MAKE AMERICA GREAT AGAIN! #MakeAmericaGreatAgain https: //t.co/DjkdAzT3WV*

Typical LMs are not sufficient to handle long-range dependencies

"Alice/Bob could not go to work that day because she/he had a doctor's appointment"

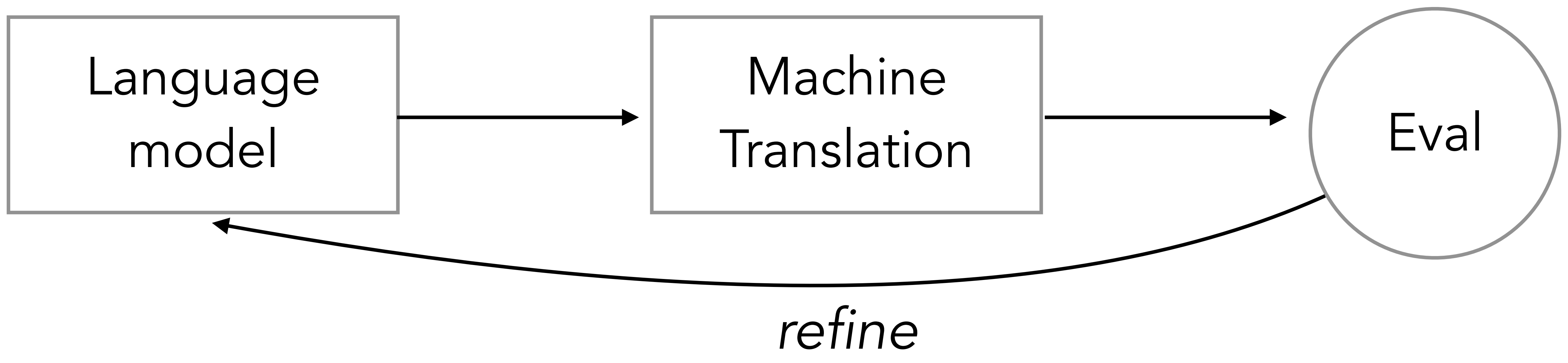
Evaluating language models

- A good language model should assign **higher probability** to typical, grammatically correct sentences
- Research process:
  - Train parameters on a suitable training corpus
    - *Assumption*: observed sentences are probably good sentences
  - Test on *different, unseen* corpus
    - **Training on any part of test set not acceptable! Why?**
  - Evaluation metric

Question

What evaluation metric should we use?

Extrinsic evaluation



- Train LM —> apply to task —> observe accuracy
- Directly optimized for downstream tasks
    - higher task accuracy —> better model
- Expensive, time consuming
- Hard to optimize downstream objective (indirect feedback)

Perplexity (ppl)

at time t, the prob of predicting x_{t+1}

$$ppl(\underset{\circ}{x}) = \prod_{t=1}^{T} \left(\frac{1}{\hat{y}^{(t)}_{x_{t+1}}} \right)^{1/T} = \exp\left(\underbrace{\frac{1}{T} \sum_{t=1}^{T} -\log \hat{y}^{(t)}_{x_{t+1}}}_{\text{cross entropy loss of the seq}} \right)$$

a sentence

- Measure of how well a probability distribution (or LM) **predicts** a sample

- For a corpus S with sentences $S^1, S^2, \ldots, S^n$

$$\text{ppl}(S) = 2^x \quad \text{where} \quad x = -\frac{1}{W}\sum_{i=1}^{n} \log_2 P(S^i)$$

→ Cross-Entropy

where W is the total number of words in test corpus

- Unigram model: $x = -\frac{1}{W}\sum_{i=1}^{n}\sum_{j=1}^{m} log_2 P(w^i_j)$ $= \log_2 P(S^i)$ (since $P(S) = \prod_j P(w_j)$)

m: # word in S^i

- Minimizing perplexity is the same as maximizing probability of corpus $P(S^1 S^2 \ldots S^n)$

Intuition on perplexity

- If our n-gram model (with vocabulary V) has following probability:

$$P(w_i|w_{i-n}, ...w_{i-1}) = \frac{1}{|V|} \quad \forall w_i$$

- What is the perplexity of the test corpus?

$$ppl = 2^{-\frac{1}{W}W \times \log(\frac{1}{|V|})} = |V|$$

$$\boxed{\text{ppl}(S) = 2^x \quad \text{where} \quad x = -\frac{1}{W}\sum_{i=1}^{n} \log_2 P(S^i)}$$

= cross-entropy

- **Any word is equally likely at next step!**

- ***Intuition***: *Perplexity measures a model's uncertainty about the next word*

Intuition on perplexity

- Perplexity is the exponentiated token-level negative log likelihood
  - **Why logs?**

- **Theory:** perplexity measured using a base of 2
  - **Practice**: doesn't matter what the base is, *e* used often
  - **Exercise**: derive why!

$$\text{ppl}(S) = 2^x \quad \text{where}$$
$$x = -\frac{1}{W}\sum_{i=1}^{n}\log_2 P(S^i)$$

Question: How good is a language model with ppl > |V| ?

Perplexity as a metric

Pros	Cons
Easy to compute	Requires domain match between train and test
standardized	might not correspond to end task optimization
directly useful, easy to use to correct sentences	log 0 undefined
nice theoretical interpretation - matching distributions	can be 'cheated' by predicting common tokens
	size of test set matters
	can be sensitive to low prob tokens/sentences

Question

What are major shortcomings of n-gram language models?

Exponential Decay of Counts

- What happens when we scale to longer contexts?

$P(w|\text{to})$ *to* occurs 1M times in corpus

$P(w|\text{go to})$ *go to* occurs 50,000 times in corpus

$P(w|\text{to go to})$ *go to* occurs 1500 times in corpus

$P(w|\text{want to go to})$ *want to go to*: only 100 occurrences

Exponential Decay of Counts

- What happens when we scale to longer contexts?

$P(w\|\text{to})$	*to* occurs 1M times in corpus
$P(w\|\text{go to})$	*go to* occurs 50,000 times in corpus
$P(w\|\text{to go to})$	*go to* occurs 1500 times in corpus
$P(w\|\text{want to go to})$	*want to go to*: only 100 occurrences

- Probability counts get very sparse, and we often want information from 5+ words away

Not all n-grams observed in training data

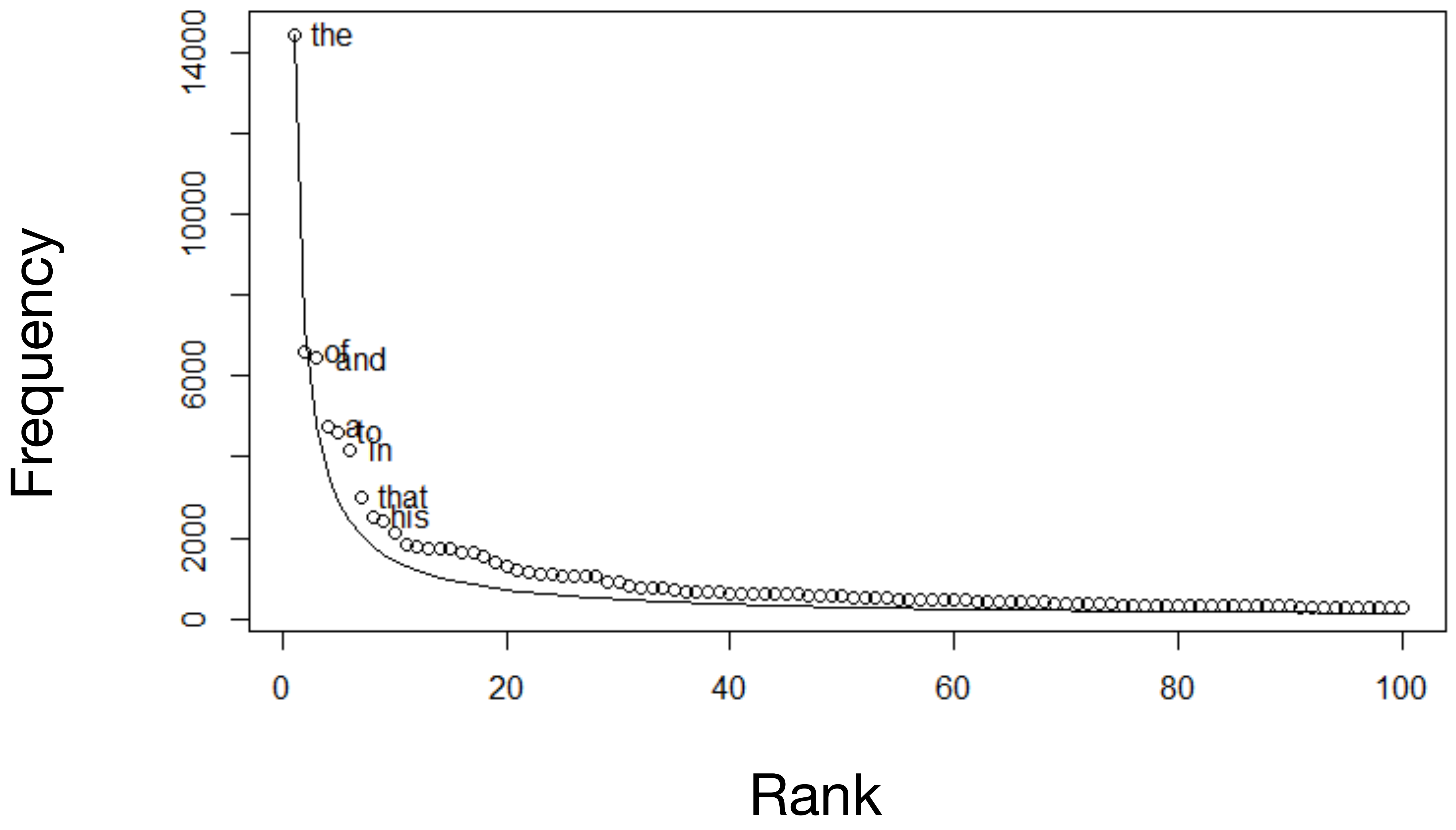
- Test corpus might have some that have zero probability under our model
  - Training set: *Google news*
  - Test set: *Shakespeare*
  - P (affray | voice doth us) = 0 → P(test corpus) = 0
  - **Undefined perplexity**

Fun Fact!

Shakespeare as corpus

- N=884,647 tokens, V=29,066
- Shakespeare produced 300,000 bigram types out of V^2= 844 million possible bigrams.
  - So 99.96% of the possible bigrams were never seen (have zero entries in the table)

Sparsity in language



$$freq \propto \frac{1}{rank}$$

Zipf's Law

- Long tail of infrequent words
- Most finite-size corpora will have this problem.

Question

How might we fix this ?

Smoothing

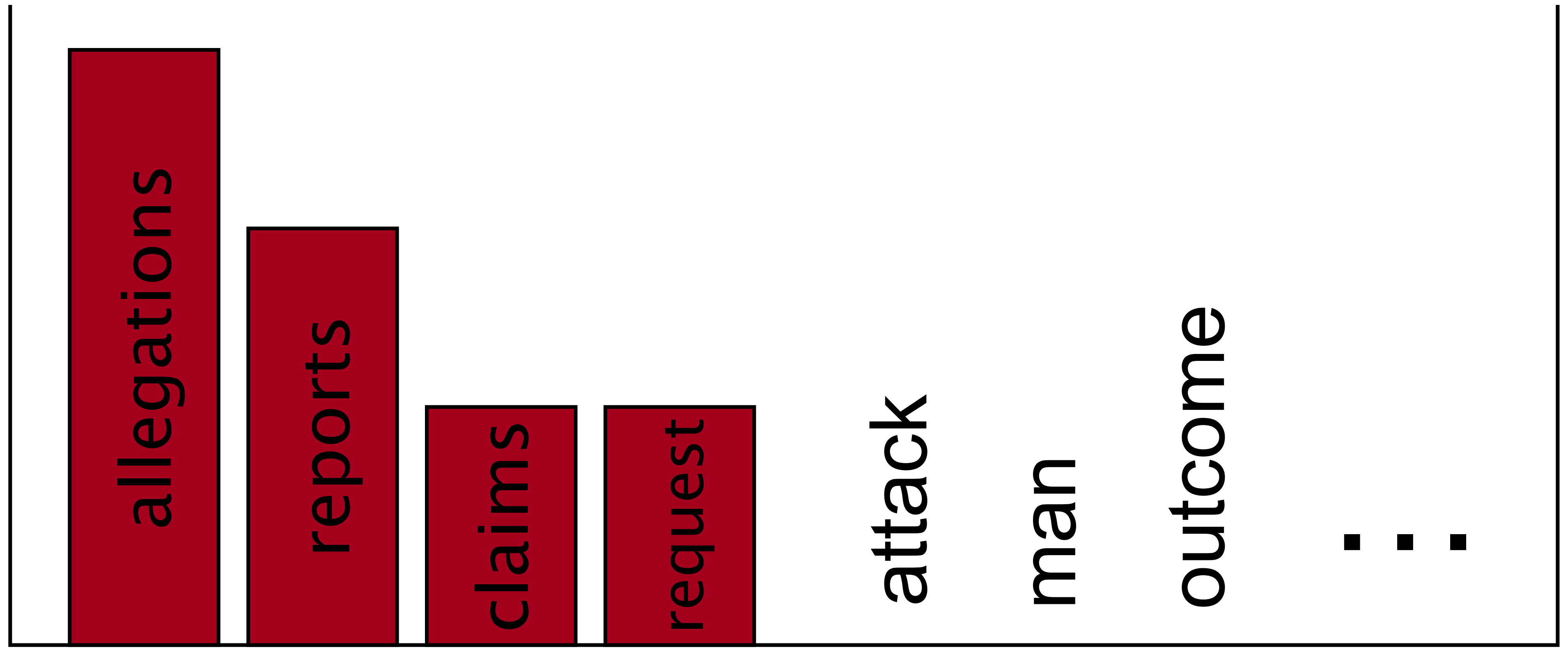
- Handle sparsity by making sure all probabilities are non-zero in our model
  - Additive: Add a small amount to all probabilities
  - Discounting: Redistribute probability mass from observed n-grams to unobserved ones
  - Back-off: Use lower order n-grams if higher ones are too sparse
  - Interpolation: Use a combination of different granularities of n-grams

Smoothing intuition

When we have sparse statistics:

P(w | denied the)
3 allegations
2 reports
1 claims
1 request

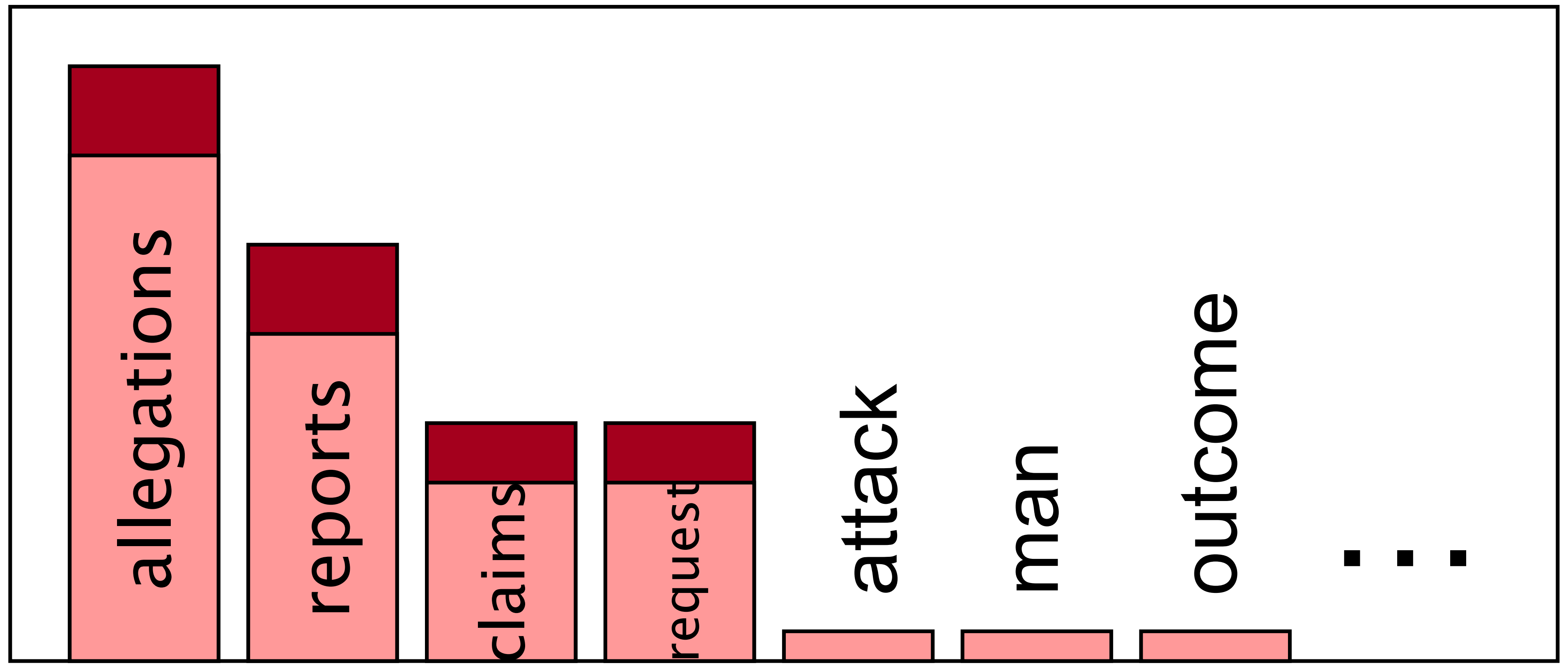
7 total



Steal probability mass to generalize better

P(w | denied the)
2.5 allegations
1.5 reports
0.5 claims
0.5 request
2 other

7 total



Laplace smoothing (add-α)

- **Simplest smoothing heuristic**: add α to all counts and renormalize!

- **Example**: max likelihood estimate for bigrams:

$$P(w_i|w_{i-1}) = \frac{C(w_{i-1}, w_i)}{C(w_{i-1})}$$

- After smoothing:

$$P(w_i|w_{i-1}) = \frac{C(w_{i-1}, w_i) + \alpha}{C(w_{i-1}) + \alpha|V|}$$

Raw bigram counts (Berkeley restaurant corpus)

- Out of 9222 sentences

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Smoothed bigram counts

	i	want	to	eat	chinese	food	lunch	spend
i	6	828	1	10	1	1	1	3
want	3	1	609	2	7	7	6	2
to	3	1	5	687	3	1	7	212
eat	1	1	3	1	17	3	43	1
chinese	2	1	1	1	1	83	2	1
food	16	1	16	1	2	5	1	1
lunch	3	1	1	1	1	2	1	1
spend	2	1	2	1	1	1	1	1

Add 1 to all the entries in the matrix

Smoothed bigram probabilities

$$P^*(w_n|w_{n-1}) = \frac{C(w_{n-1}w_n)+1}{C(w_{n-1})+V}$$

	i	want	to	eat	chinese	food	lunch	spend
i	0.0015	0.21	0.00025	0.0025	0.00025	0.00025	0.00025	0.00075
want	0.0013	0.00042	0.26	0.00084	0.0029	0.0029	0.0025	0.00084
to	0.00078	0.00026	0.0013	0.18	0.00078	0.00026	0.0018	0.055
eat	0.00046	0.00046	0.0014	0.00046	0.0078	0.0014	0.02	0.00046
chinese	0.0012	0.00062	0.00062	0.00062	0.00062	0.052	0.0012	0.00062
food	0.0063	0.00039	0.0063	0.00039	0.00079	0.002	0.00039	0.00039
lunch	0.0017	0.00056	0.00056	0.00056	0.00056	0.0011	0.00056	0.00056
spend	0.0012	0.00058	0.0012	0.00058	0.00058	0.00058	0.00058	0.00058

Discounting

Bigram count in training	Bigram count in heldout set
0	.0000270
1	0.448
2	1.25
3	2.24
4	3.23
5	4.21
6	5.23
7	6.21
8	7.21
9	8.26

- Determine some "mass" to remove from probability estimates
- Redistribute mass among unseen n-grams
- Just choose an absolute value to discount (usually <1)

$$P_{\text{abs\_discount}}(w_i \mid w_{i-1}) = \begin{cases} \dfrac{c(w_{i-1}, w_i) - d}{c(w_{i-1})} & \text{if } c(w_{i-1}, w_i) > 0 \\ \alpha(w_{i-1}) \dfrac{P(w_i)}{\sum_{w'} P(w')} & \text{for all } w' \text{ s.t. } c(w_{i-1}, w') = 0 \text{ if } c(w_{i-1}, w_i) = 0 \end{cases}$$

Unigram probabilities

Back-off

- Use n-gram if enough evidence, else back off to (n-1)-gram

$$P_{bo}(w_i \mid w_{i-n+1} \cdots w_{i-1})$$
$$= \begin{cases} d_{w_{i-n+1} \cdots w_i} \dfrac{C(w_{i-n+1} \cdots w_{i-1} w_i)}{C(w_{i-n+1} \cdots w_{i-1})} & \text{if } C(w_{i-n+1} \cdots w_i) > k \\ \alpha_{w_{i-n+1} \cdots w_{i-1}} P_{bo}(w_i \mid w_{i-n+2} \cdots w_{i-1}) & \text{otherwise} \end{cases}$$

(Katz back-off)

- d = amount of discounting
- α = back-off weight

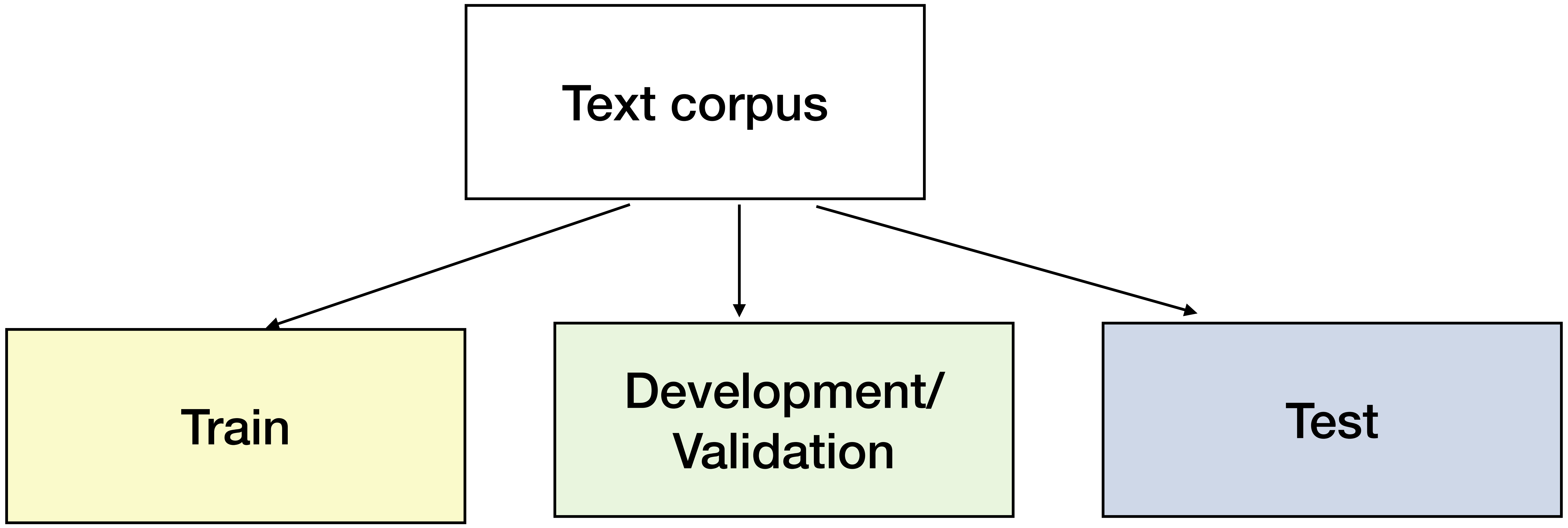
Linear Interpolation

$$\begin{aligned}\hat{P}(w_i|w_{i-1}, w_{i-2}) = \lambda_1 P(w_i|w_{i-1}, w_{i-2}) & \quad \text{Trigram} \\ +\lambda_2 P(w_i|w_{i-1}) & \quad \text{Bigram} \\ +\lambda_3 P(w_i) & \quad \text{Unigram}\end{aligned}$$

$$\sum_i \lambda_i = 1$$

- Use a combination of models to estimate probability
- Strong empirical performance

Choosing lambdas



- First, estimate n-gram prob. on training set
- Then, estimate lambdas (hyperparameters) to maximize probability on the held-out development/validation set
- Use best model from above to evaluate on test set

Recap

- *n*-gram language models are simple, effective methods for estimating the probability of sequences
- **Problem #1:** Number of modelable sequences grows exponentially with *n*
- **Problem #2:** Smoothing heuristics must be used to estimate the probability of sequences unseen during training

Language Models:
Fixed-context Neural Models

Antoine Bosselut

EPFL

nlp lab

Today's Outline

- **Part 1:** Language models introduction, count-based language models, evaluating language models, smoothing
- **Part 2:** Fixed-context Language Models
- **Additional Slides:** Neural networks recap

Some slides adapted from Danqi Chen, Mohit Iyyer

Question

What's another issue with n-gram language models?

Problem

With n-gram language models, there is no notion of similarity unless sequence overlaps!

We treat all words / prefixes as independent of one another!

students opened their ___

pupils opened their ___

scholars opened their ___

undergraduates opened their ___

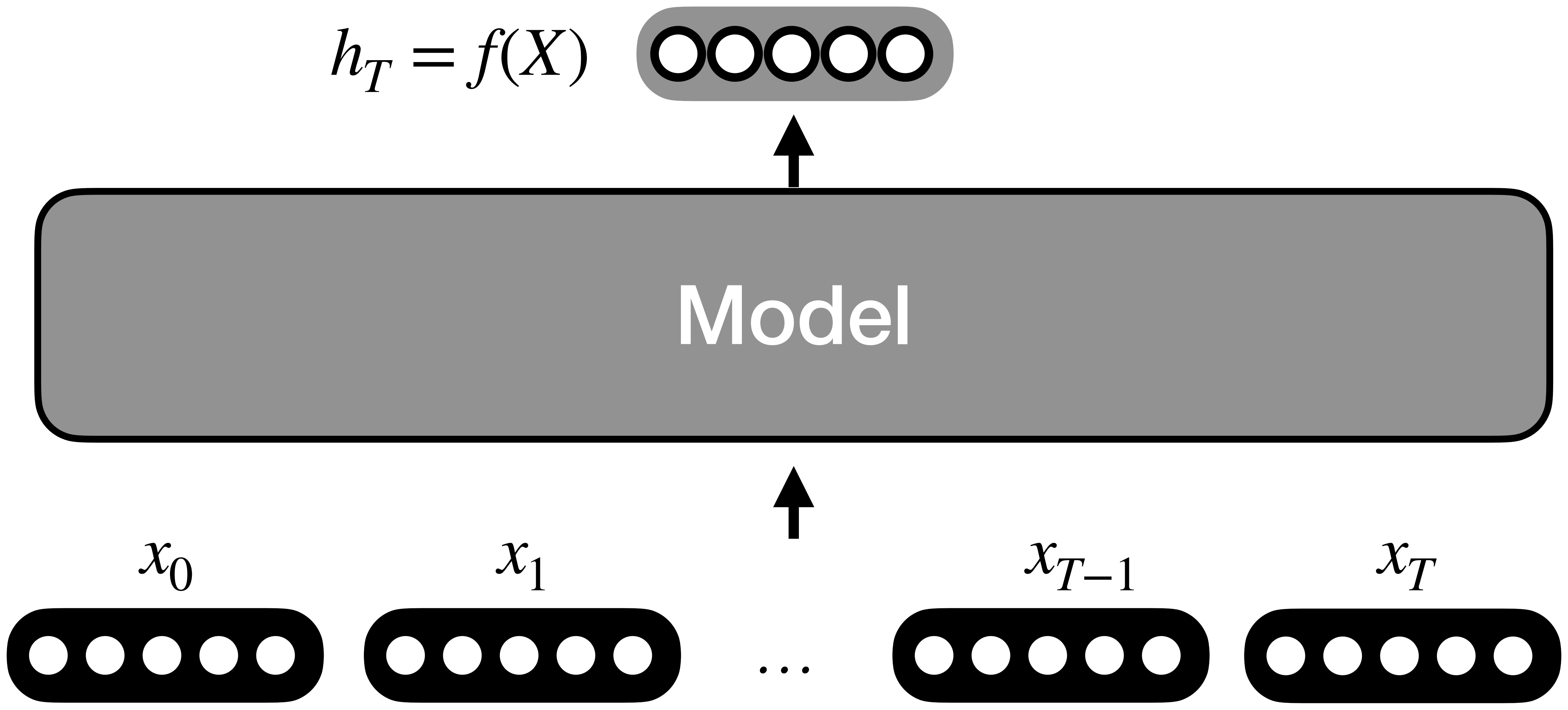
students turned the pages of their ___

students attentively perused their ___

Ideally, we should share information across these same prefixes!

Recall

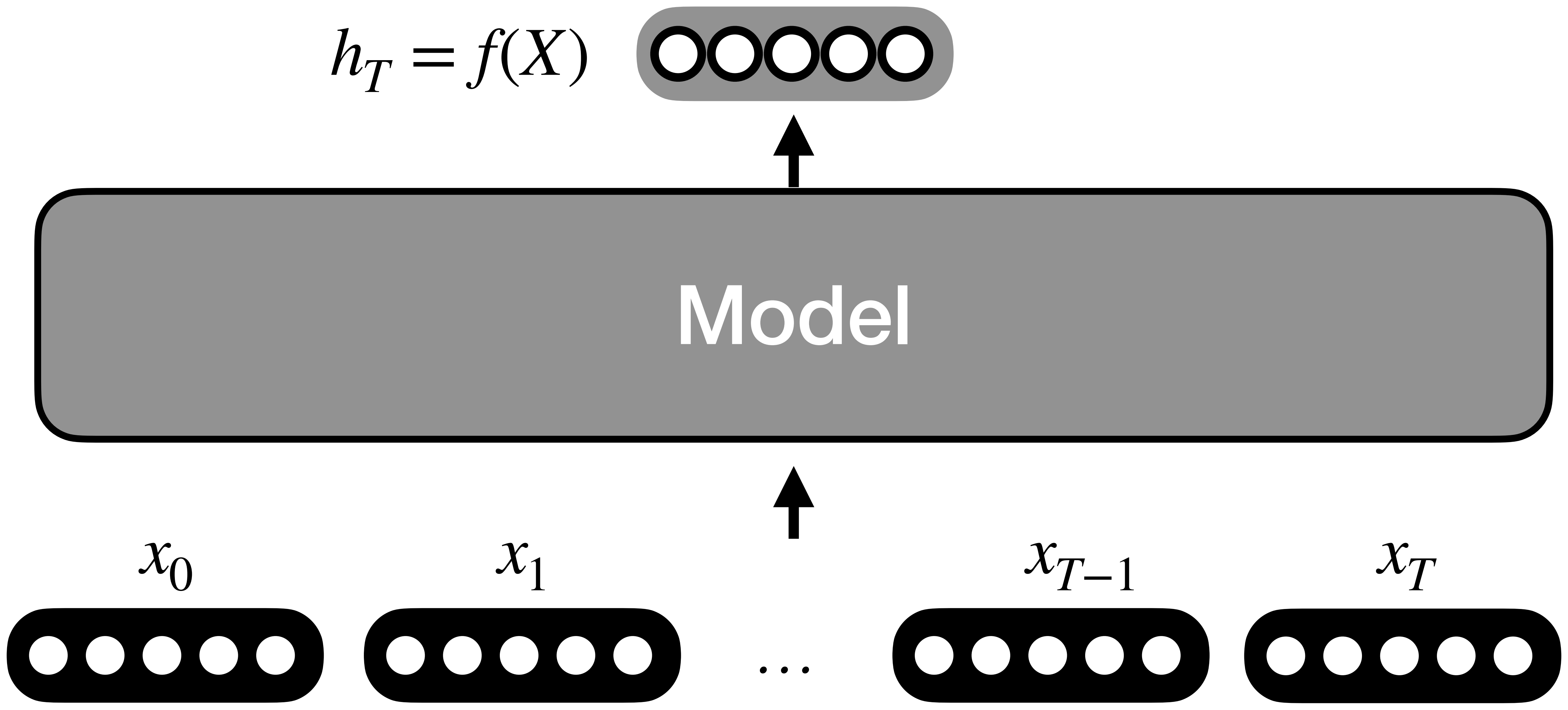
- neural networks **compose** word embeddings into vectors for natural language phrases, sentences, and documents



Recall

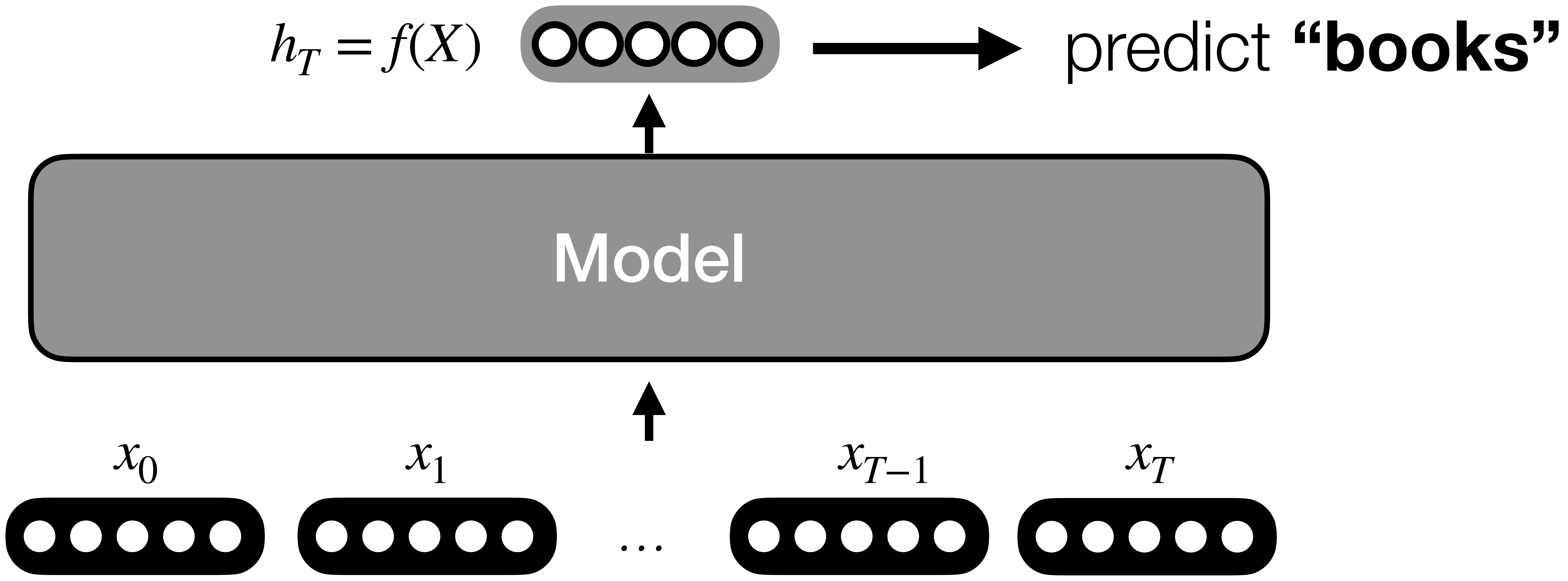
- neural networks **compose** word embeddings into vectors for natural language phrases, sentences, and documents

- Predict the next word from composed prefix representation



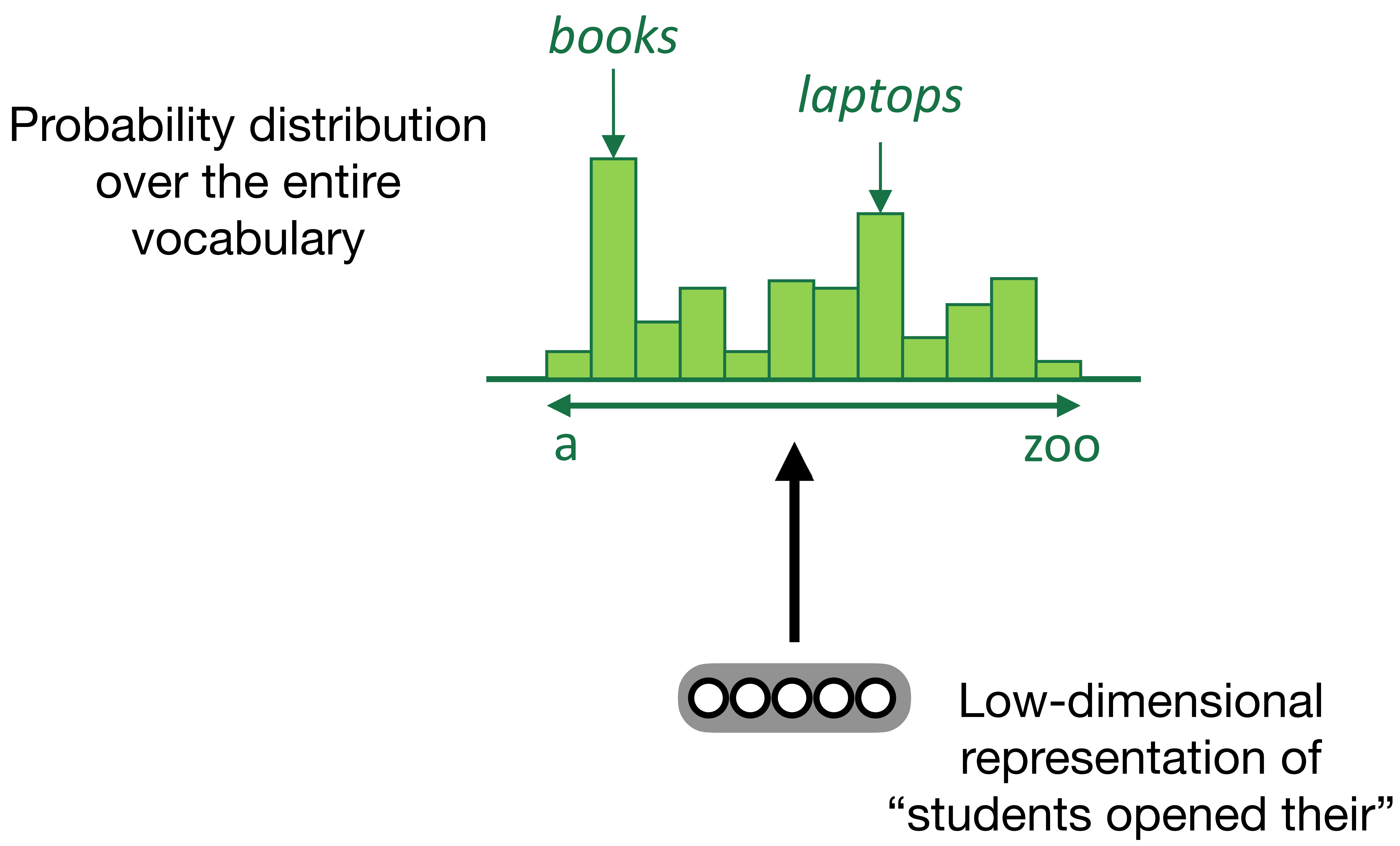
How does this happen?

Vocab Projection + Softmax layers:
Converts a vector representation
into a probability distribution over
the entire vocabulary



Vocabulary Space Projection

$P(w_i \mid \text{vector for "students opened their"})$



Recap

- Given a d-dimensional vector representation $\boldsymbol{x}$ of a prefix, we do the following to predict the next word:

1. Project it to a V-dimensional vector using a matrix-vector product (a.k.a. a "linear layer", or a "feedforward layer"), where V is the size of the vocabulary

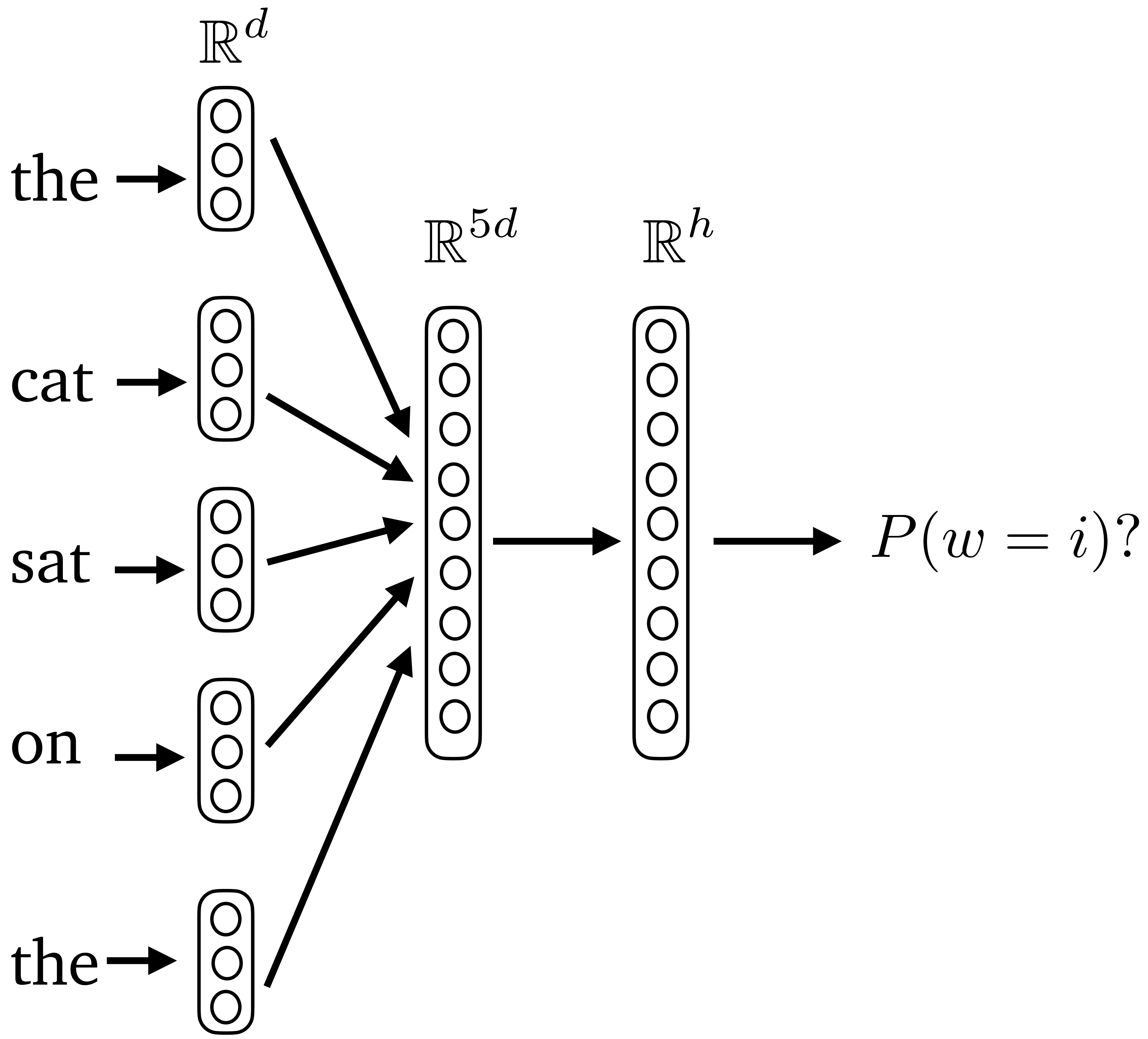
2. Apply the softmax function to transform the resulting vector into a probability distribution

Question

How should we represent the history of the sequence?

Fixed-context Neural Language Models

- P(mat | the cat sat on the) = ?



- Input layer (n = 5):

$$\mathbf{x} = [\mathbf{e}_{\text{the}}; \mathbf{e}_{\text{cat}}; \mathbf{e}_{\text{sat}}; \mathbf{e}_{\text{on}}; \mathbf{e}_{\text{the}}] \in \mathbb{R}^{dn}$$

- Hidden layer

$$\mathbf{h} = \tanh(\mathbf{W}\mathbf{x} + \mathbf{b}) \in \mathbb{R}^h$$

- Output layer (softmax)

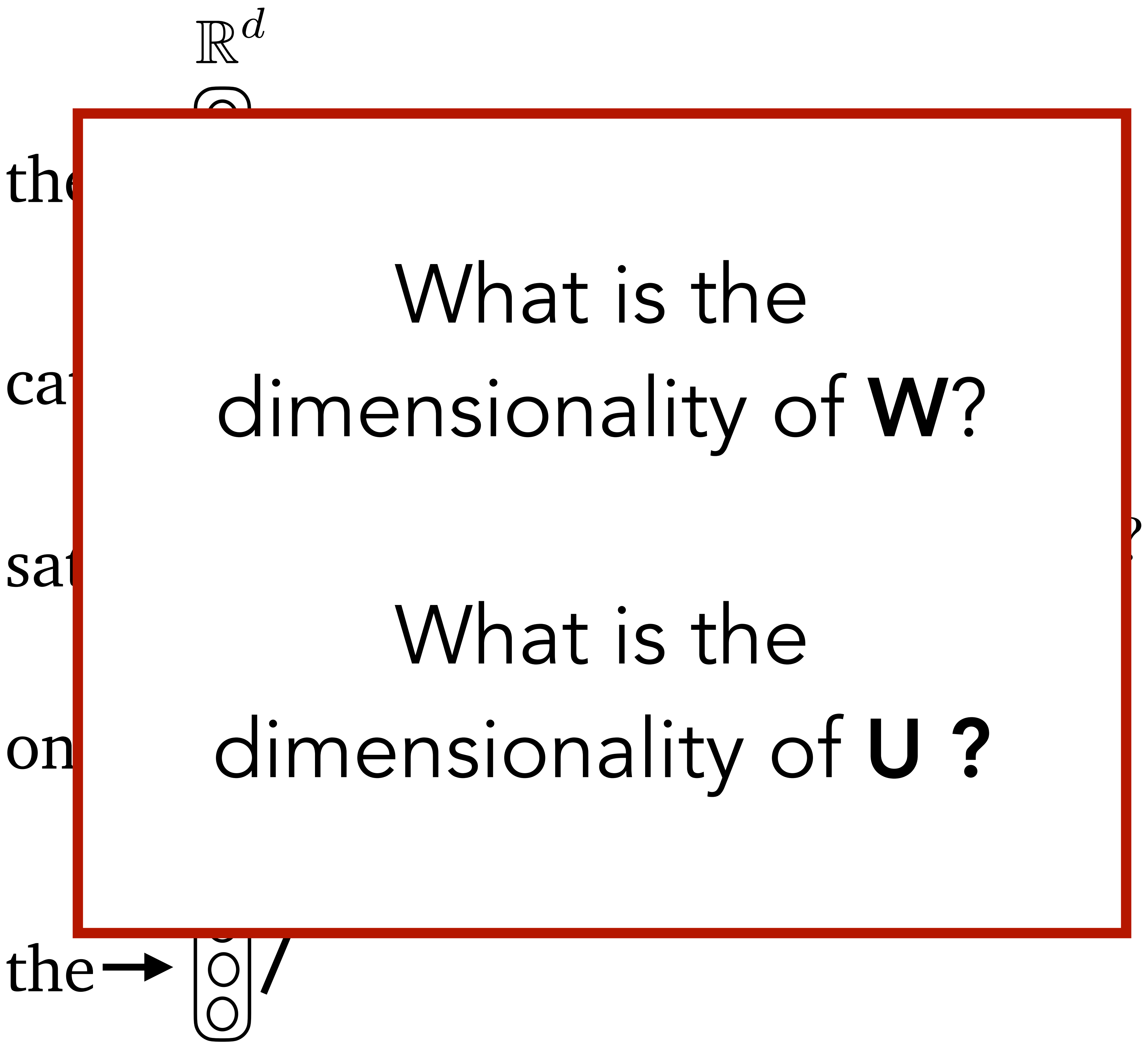
$$\mathbf{z} = \mathbf{U}\mathbf{h} \in \mathbb{R}^{|V|}$$

$$P(w = i \mid \text{the cat sat on the})$$
$$= \text{softmax}_i(\mathbf{z}) = \frac{e^{z_i}}{\sum_k e^{z_k}}$$

(Bengio et al., 2003): A Neural Probabilistic Language Model

Fixed-context Neural Language Models

- P(mat | the cat sat on the) = ?



What is the dimensionality of **W**?

What is the dimensionality of **U** ?

- Input layer (n = 5):

$$\mathbf{x} = [\mathbf{e}_{\text{the}}; \mathbf{e}_{\text{cat}}; \mathbf{e}_{\text{sat}}; \mathbf{e}_{\text{on}}; \mathbf{e}_{\text{the}}] \in \mathbb{R}^{dn}$$

- Hidden layer

$$\mathbf{h} = \tanh(\mathbf{W}\mathbf{x} + \mathbf{b}) \in \mathbb{R}^h$$

- Output layer (softmax)

$$\mathbf{z} = \mathbf{U}\mathbf{h} \in \mathbb{R}^{|V|}$$

$$P(w = i \mid \text{the cat sat on the}) = \text{softmax}_i(\mathbf{z}) = \frac{e^{z_i}}{\sum_k e^{z_k}}$$

(Bengio et al., 2003): A Neural Probabilistic Language Model

Question

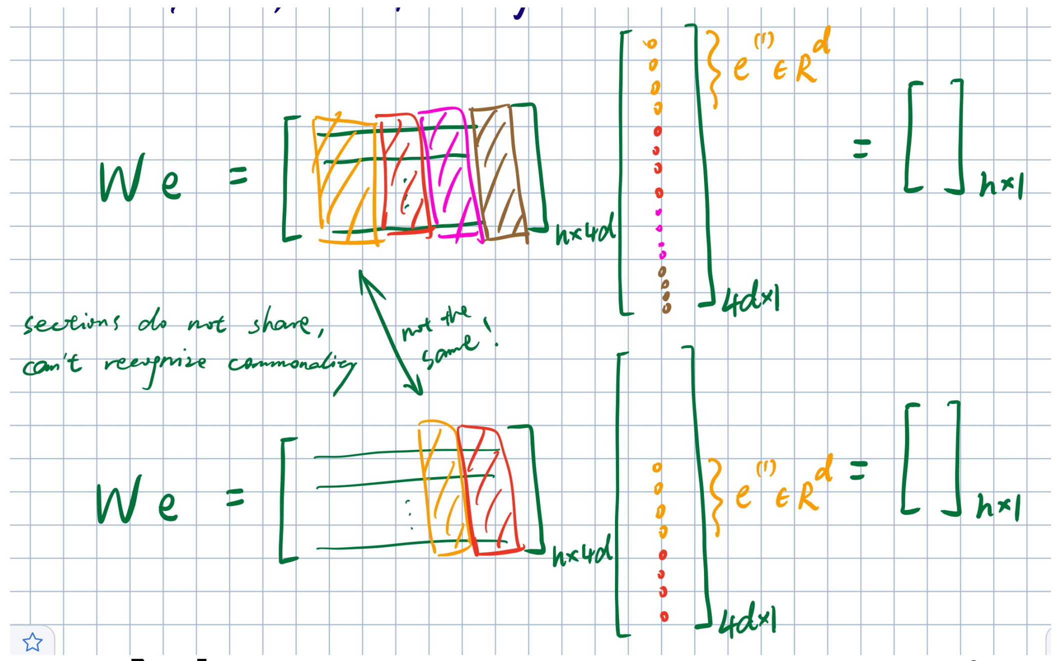
What algorithm did we see last week do fixed context language models resemble the most?

Question

What algorithm did we see last week do fixed context language models resemble the most?

CBOW!

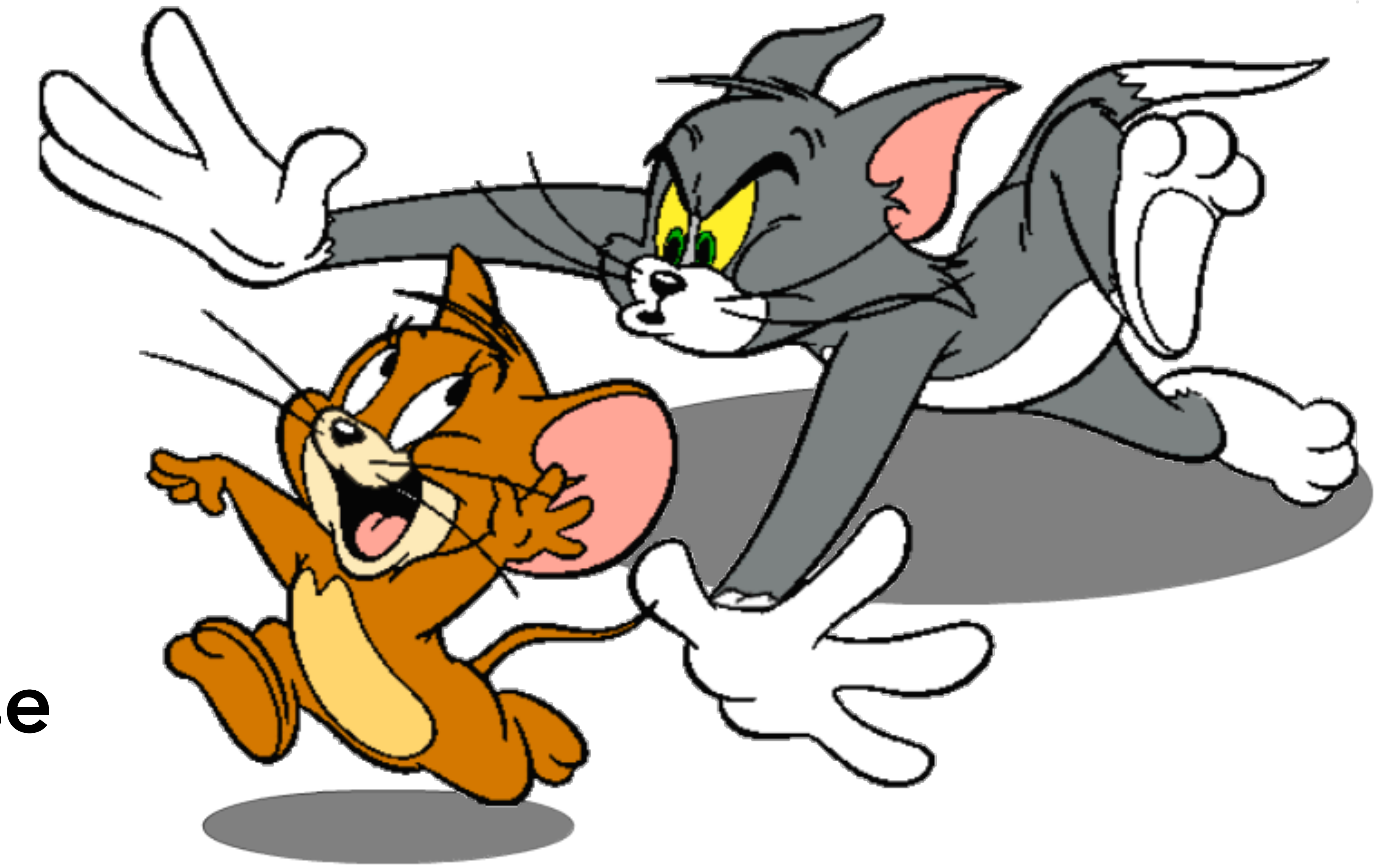
Advantages vs. Disadvantages



- No more sparsity problem
  - All sequences can be estimated with non-zero probability ! **Why?**
- Model size is much smaller!
  - Depends on number of weights in model, not number of sequences!

- Fixed windows are still too small to encode **long-range dependencies**
- Enlarging the window size makes the weight matrix **W** larger (more computationally expensive!)
- Weights in **W** aren't shared across embeddings in the window (computationally inefficient!)

Sequences are not of consistent length



The **cat** chased the **mouse**

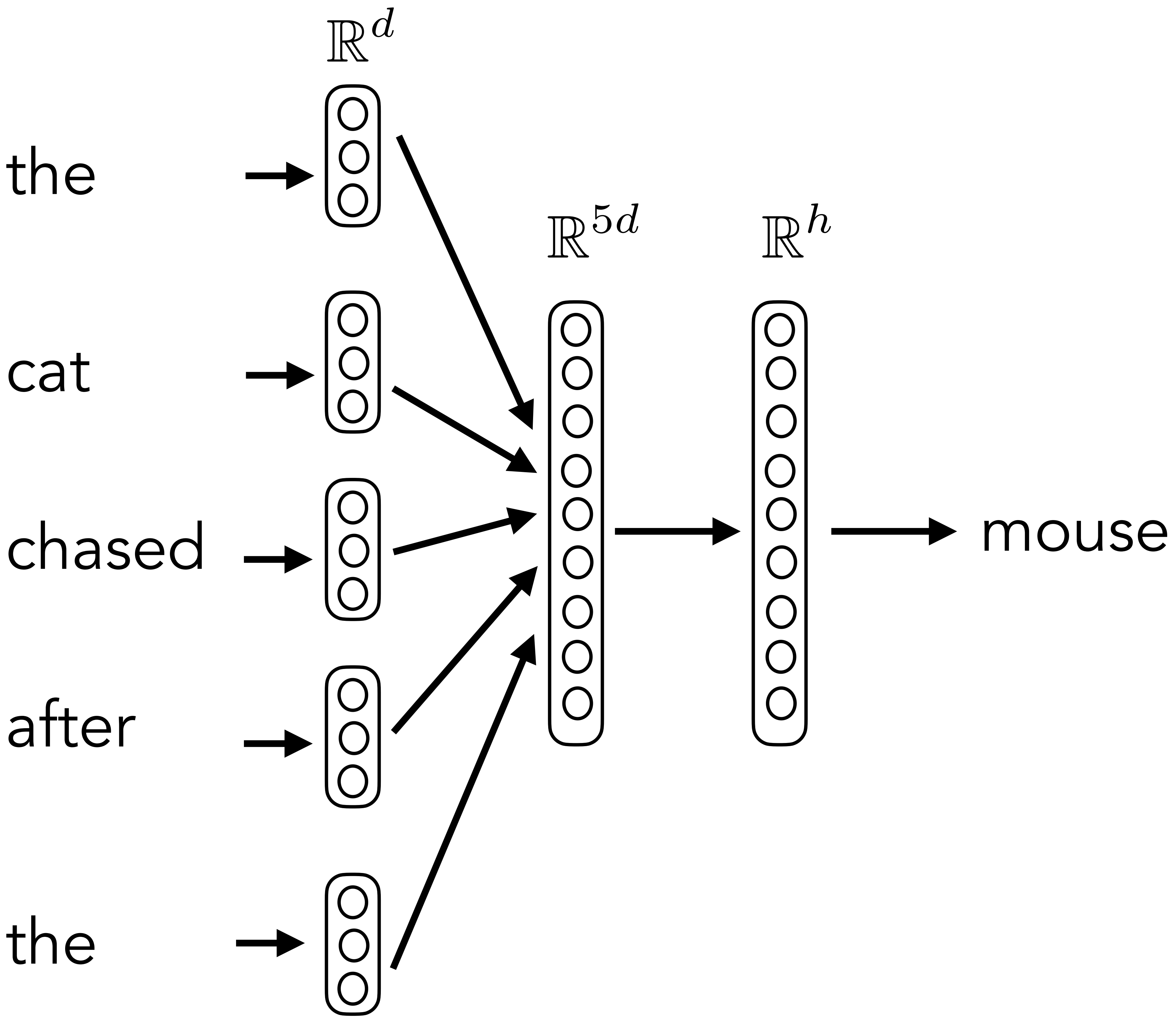
The starving **cat** chased the **mouse**

The starving **cat** fanatically chased the **mouse**

The starving **cat** fanatically chased the elusive **mouse**

The starving **cat**, who had not eaten in six days, fanatically chased the elusive **mouse**

Fixed-context Neural Language Models



P(mouse | the cat chase the)

P(mouse | the starving cat chased the)

P(mouse | starving cat chased after the)

P(mouse | cat fanatically chased after the)

P(mouse | fanatically chased after the elusive)

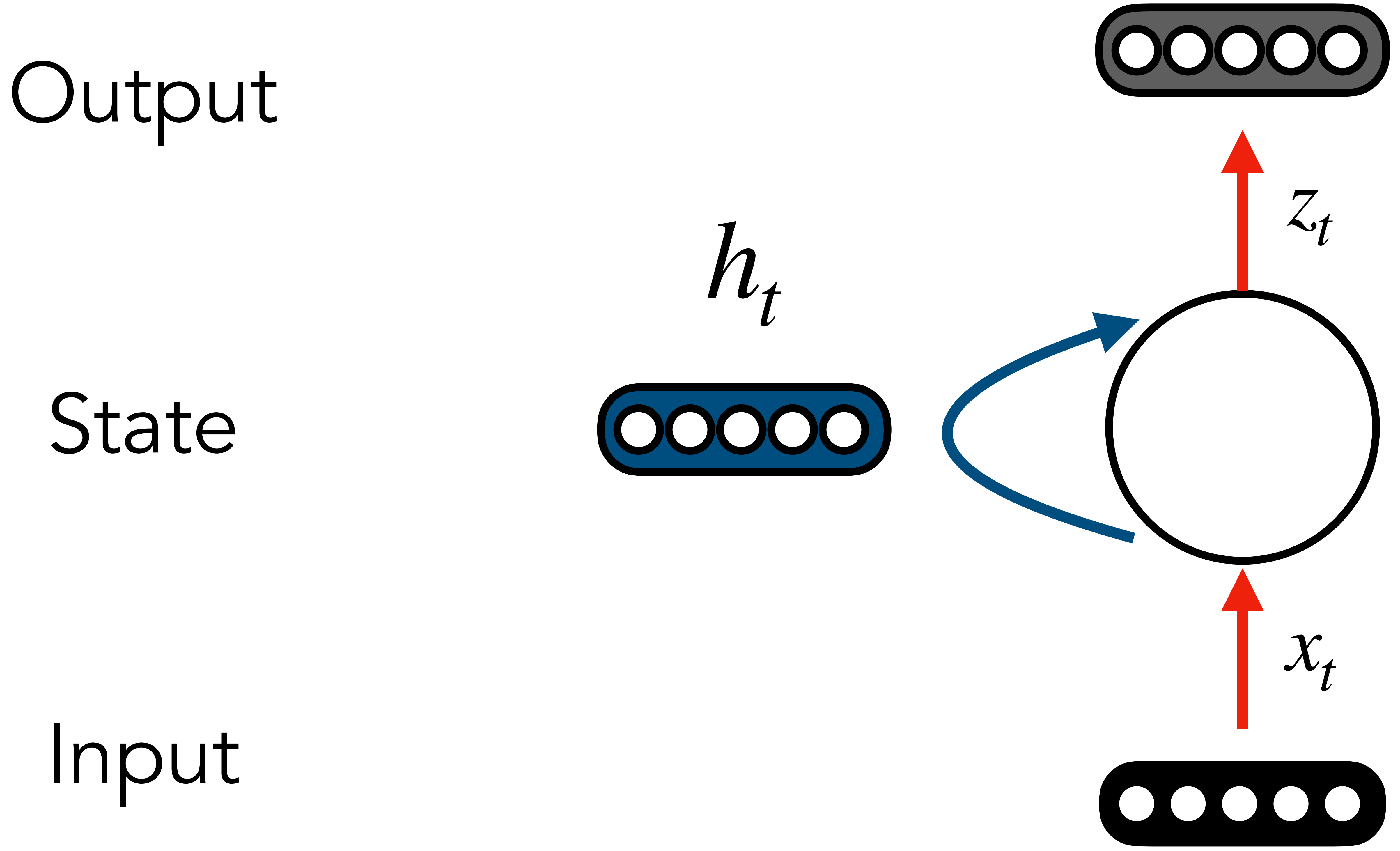
(Bengio et al., 2003): A Neural Probabilistic Language Model

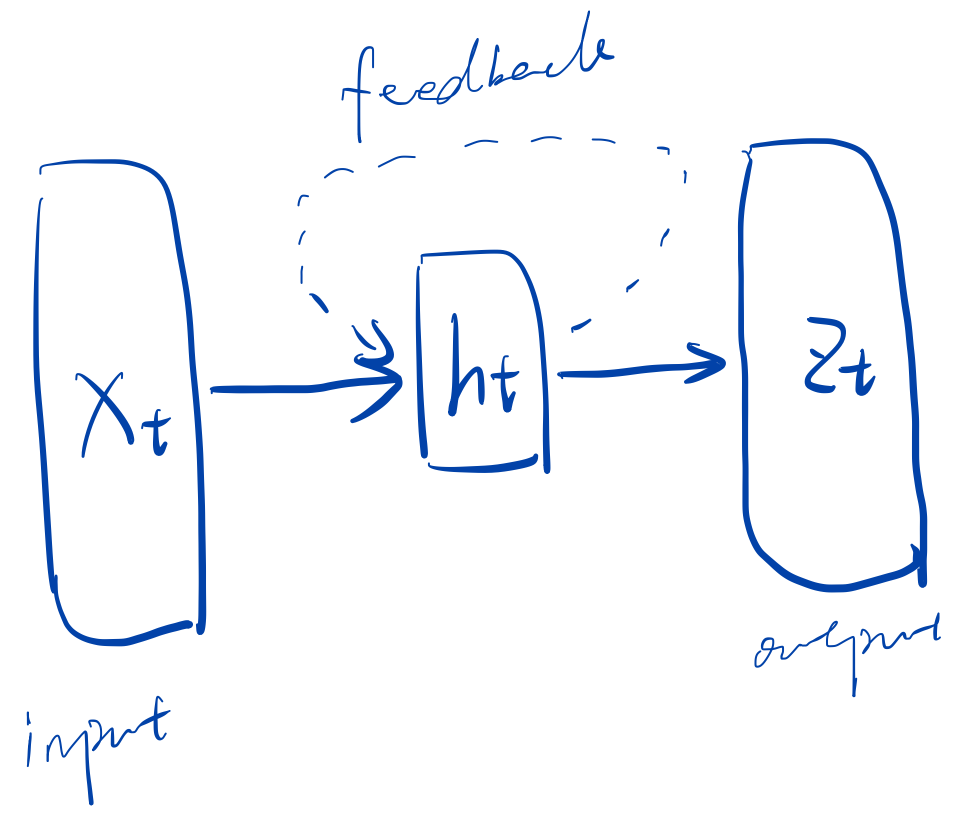
Language Models: Recurrent Neural Networks

Antoine Bosselut

Recurrent Neural Networks

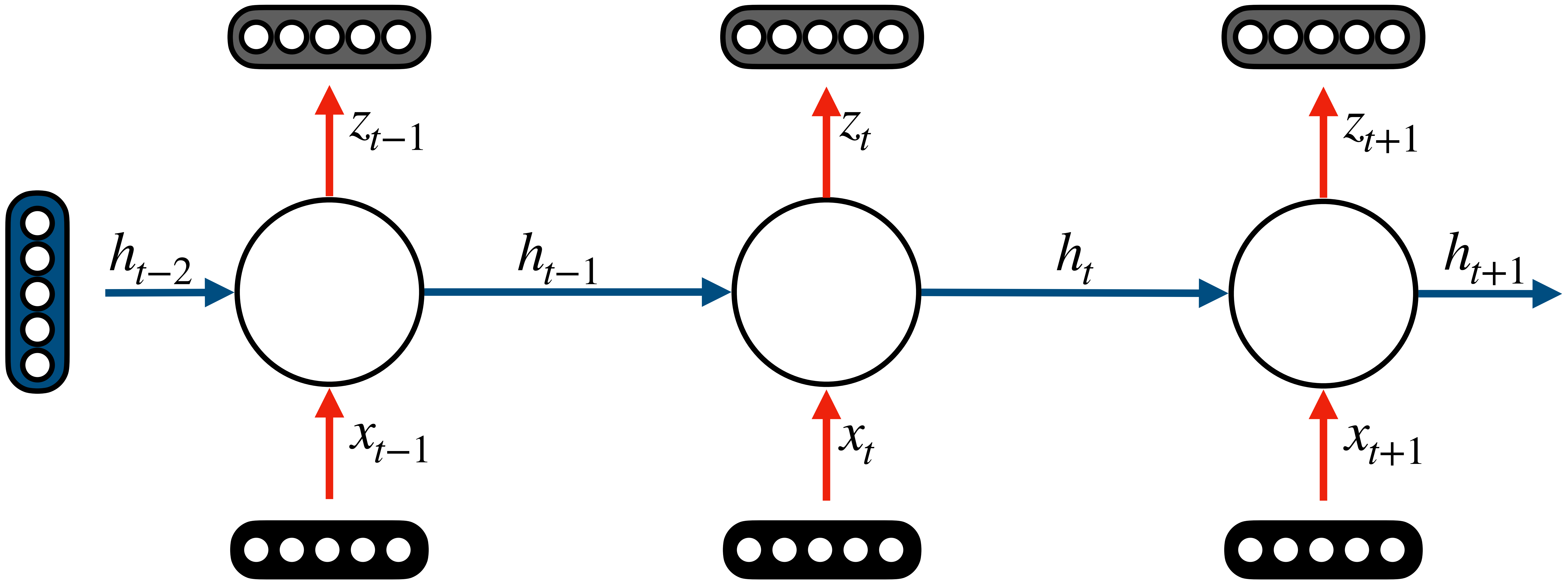
- **Solution:** Recurrent neural networks — NNs with feedback loops





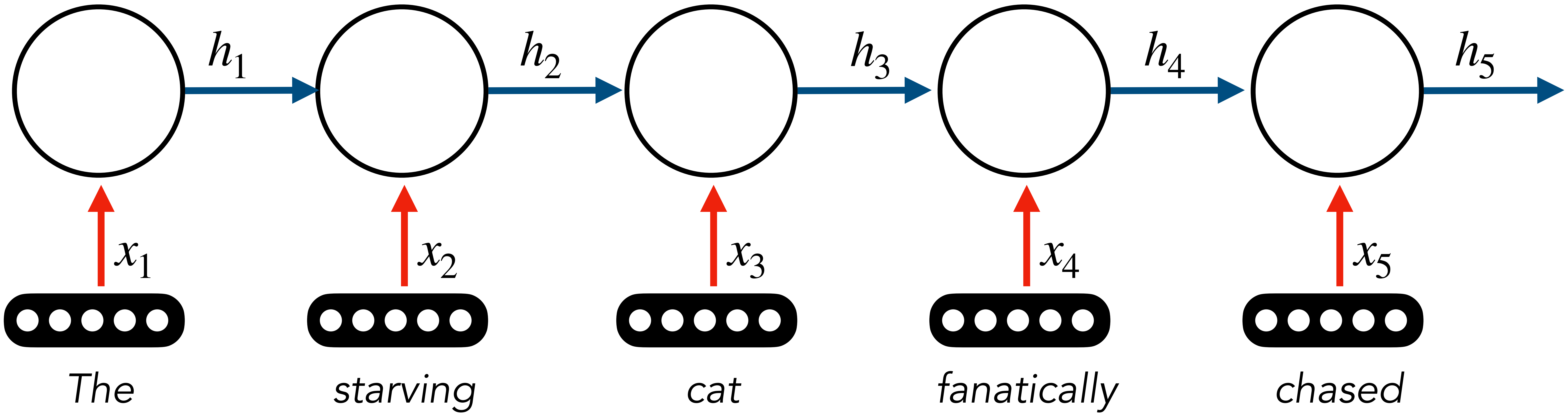
Unrolling the RNN

Unrolling the RNN across all time steps gives full computation graph

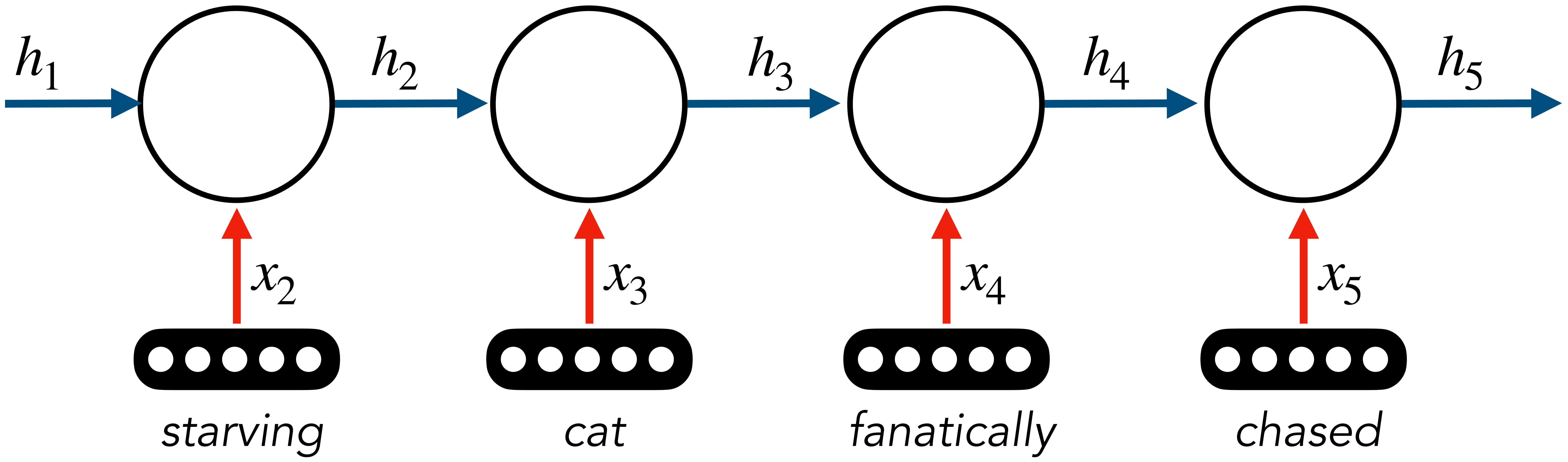


Allows for learning from entire sequence history, regardless of length

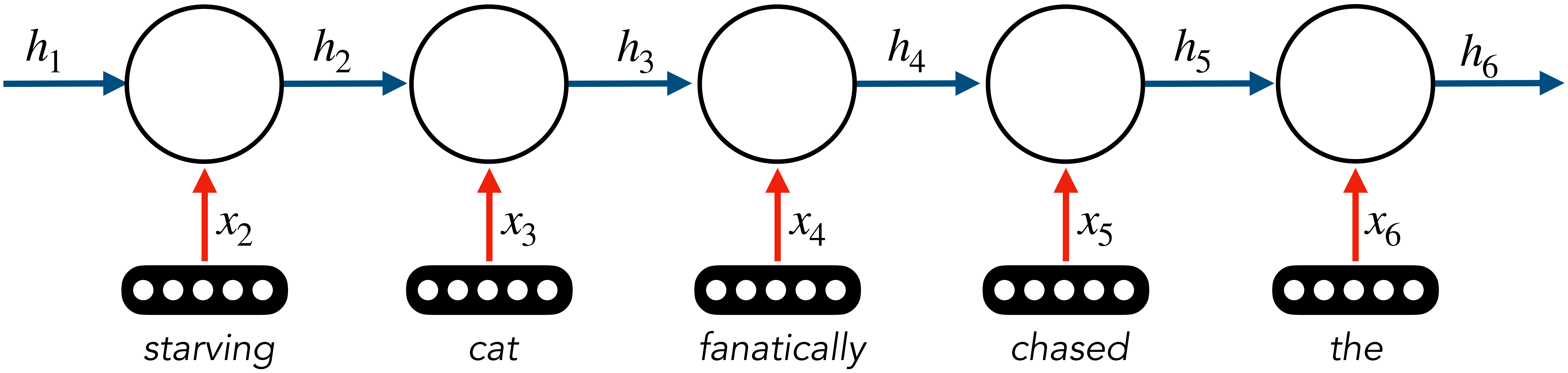
Unrolling the RNN



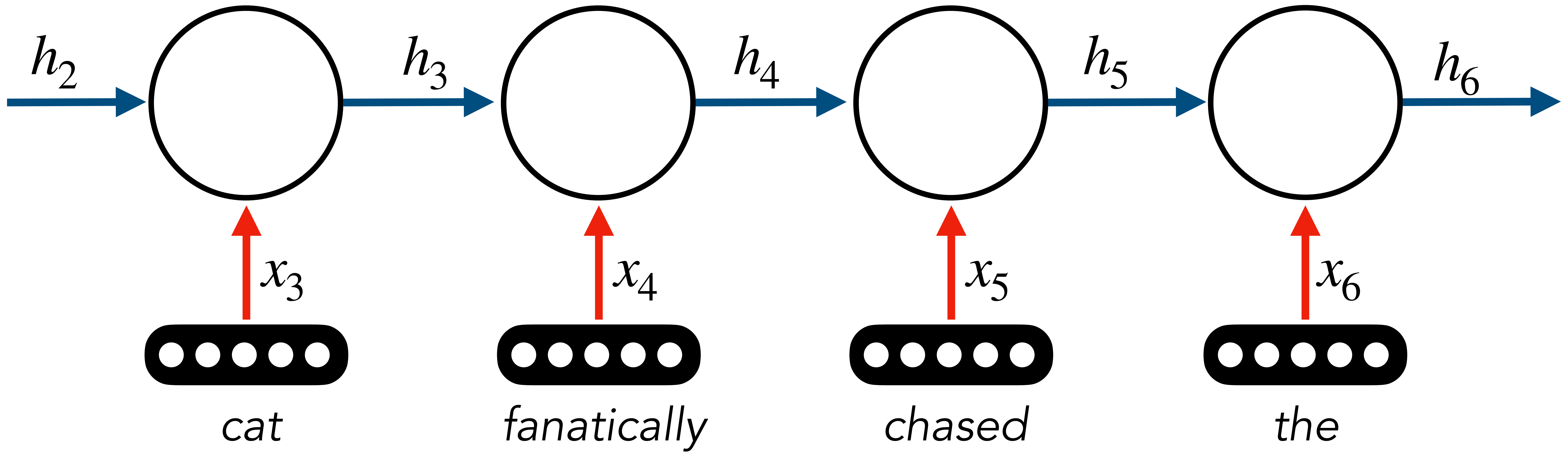
Unrolling the RNN



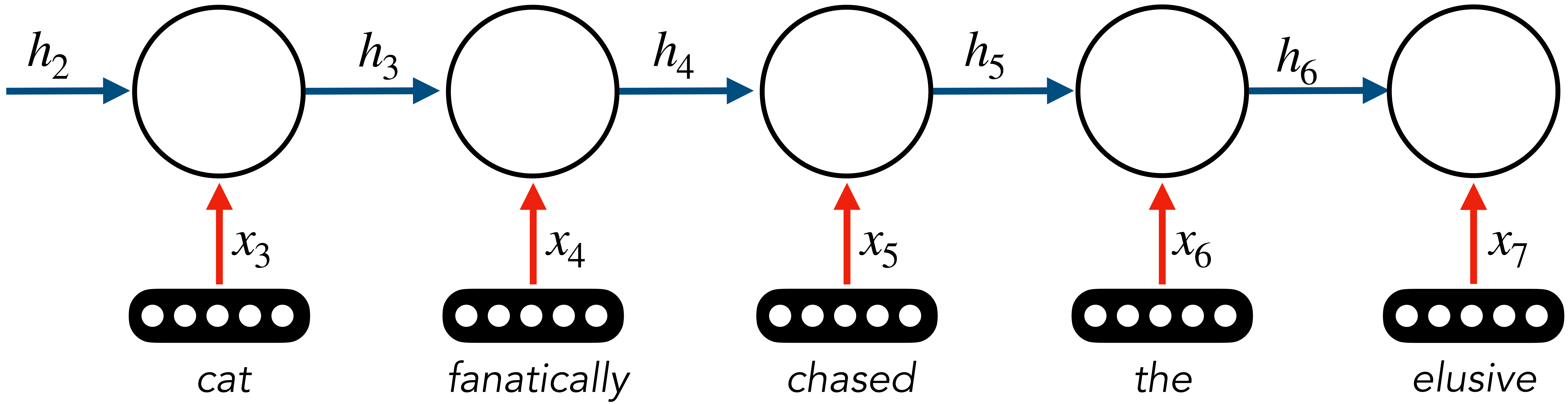
Unrolling the RNN



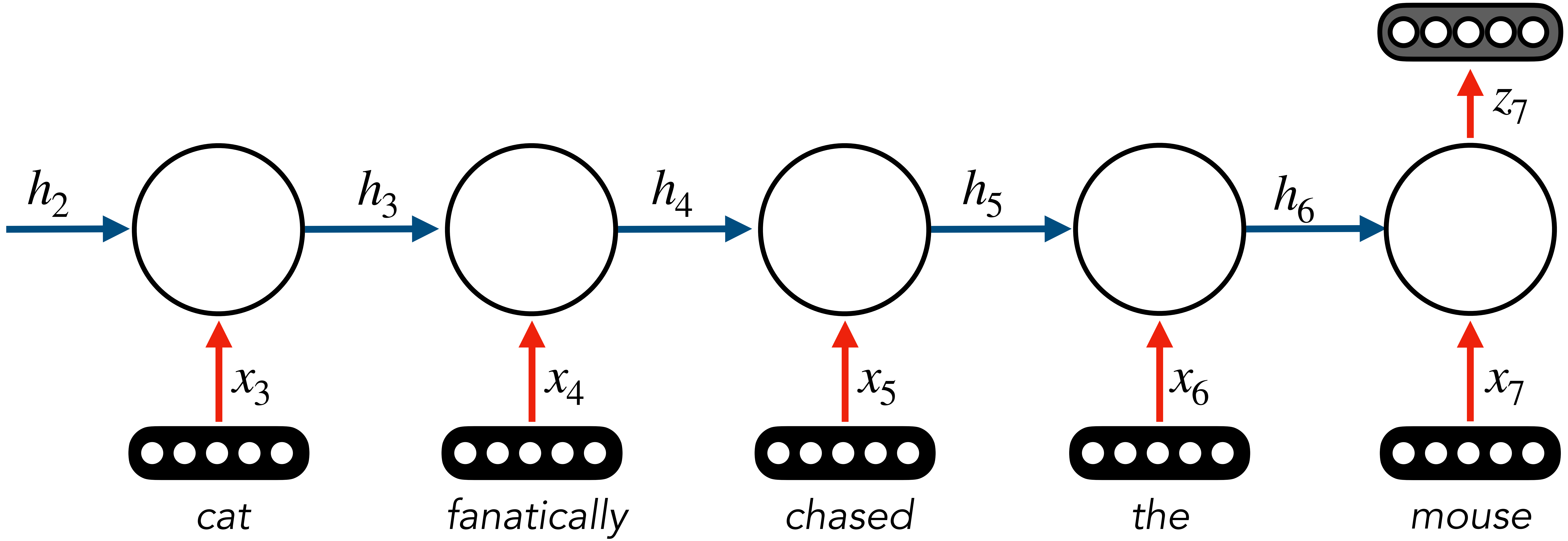
Unrolling the RNN



Unrolling the RNN



Unrolling the RNN



Classification

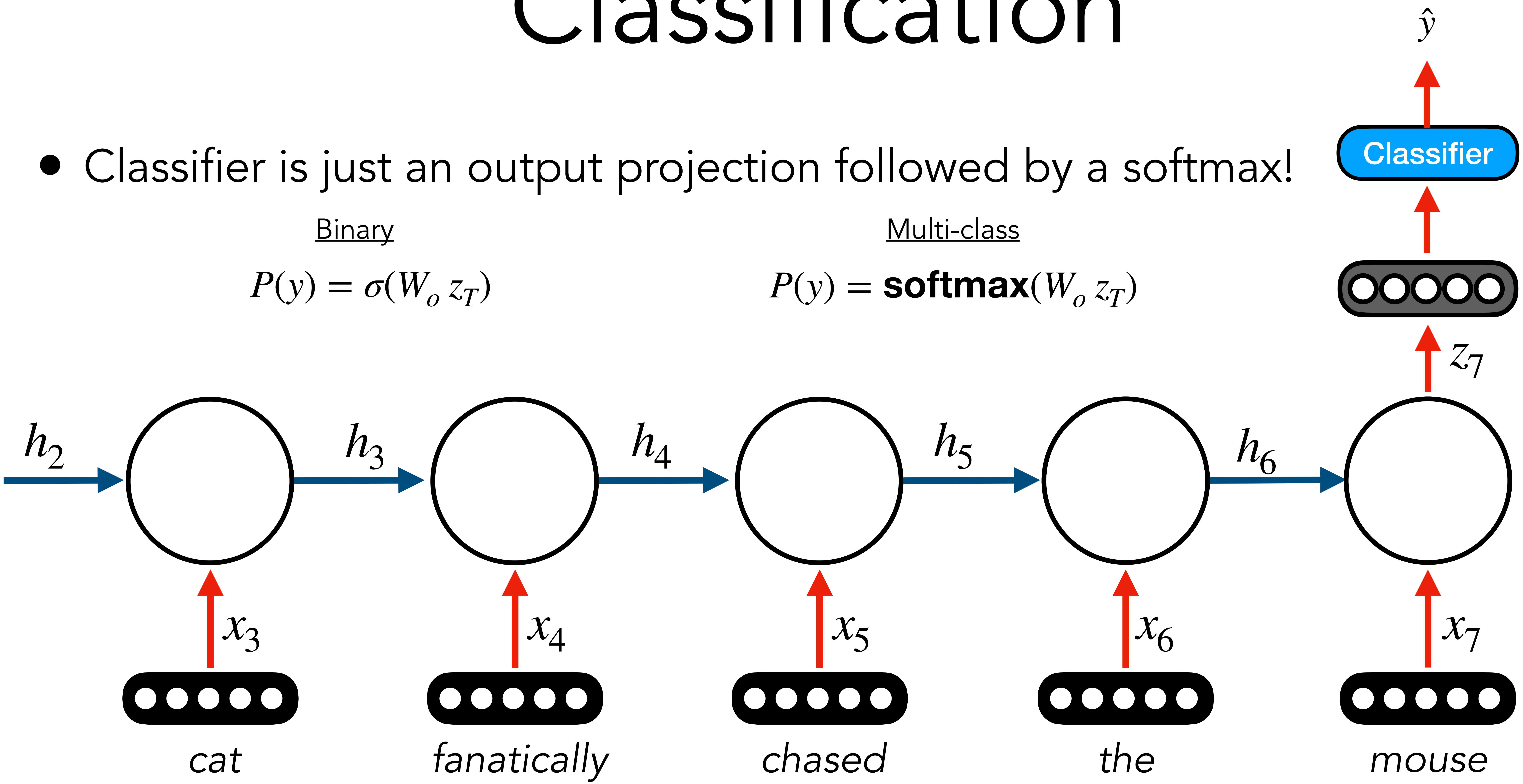
- Classifier is just an output projection followed by a softmax!

Binary

$$P(y) = \sigma(W_o\, z_T)$$

Multi-class

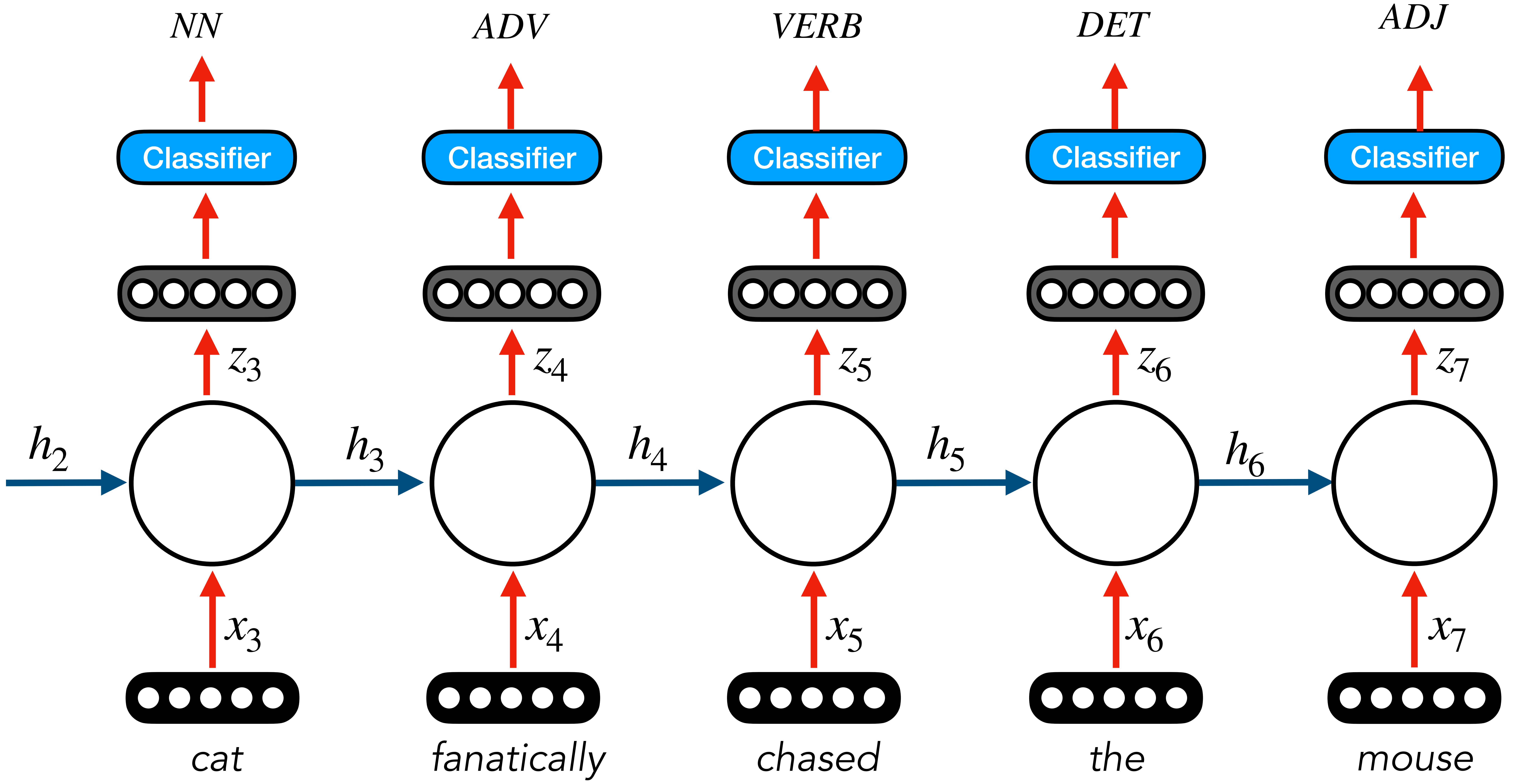
$$P(y) = \mathbf{softmax}(W_o\, z_T)$$



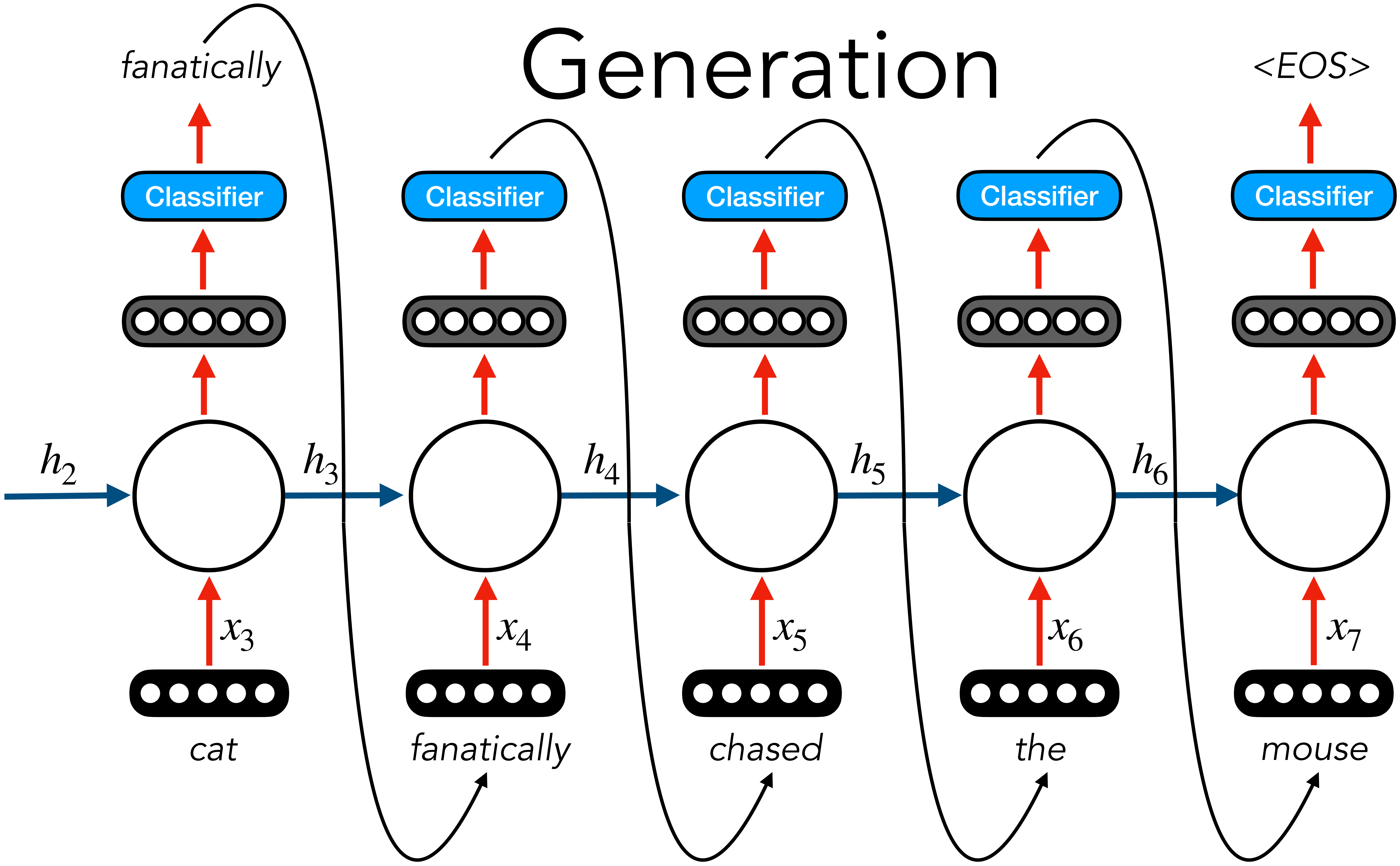
Question

Why would you use the output of the last recurrent unit as the one to predict a label?

Sequence Labeling



Generation


fanatically
<EOS>
Classifier
Classifier
Classifier
Classifier
Classifier
h2
h3
h4
h5
h6
x3
x4
x5
x6
x7
cat
fanatically
chased
the
mouse

Example

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

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

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

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

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

VIOLA:
I'll drink it.
```

Not bad for generating Shakespeare!

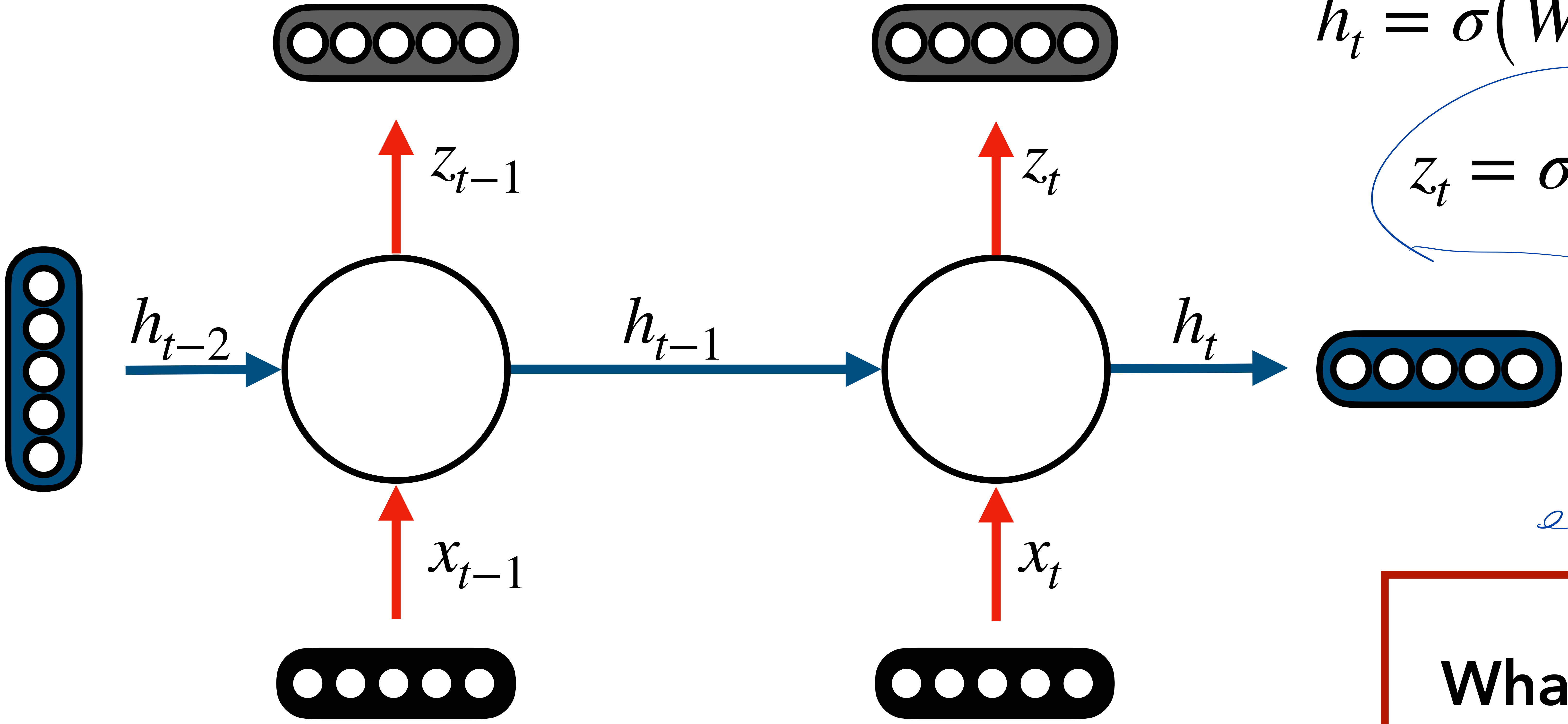
https://karpathy.github.io/2015/05/21/rnn-effectiveness/

Wikipedia works too!

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


https://karpathy.github.io/2015/05/21/rnn-effectiveness/

Classical RNN: Elman Network



$$h_t = \sigma\left(W_{hx}x_t + W_{hh}h_{t-1} + b_h\right)$$

$$z_t = \sigma\left(W_{zh}h_t + b_z\right)$$

for prediction

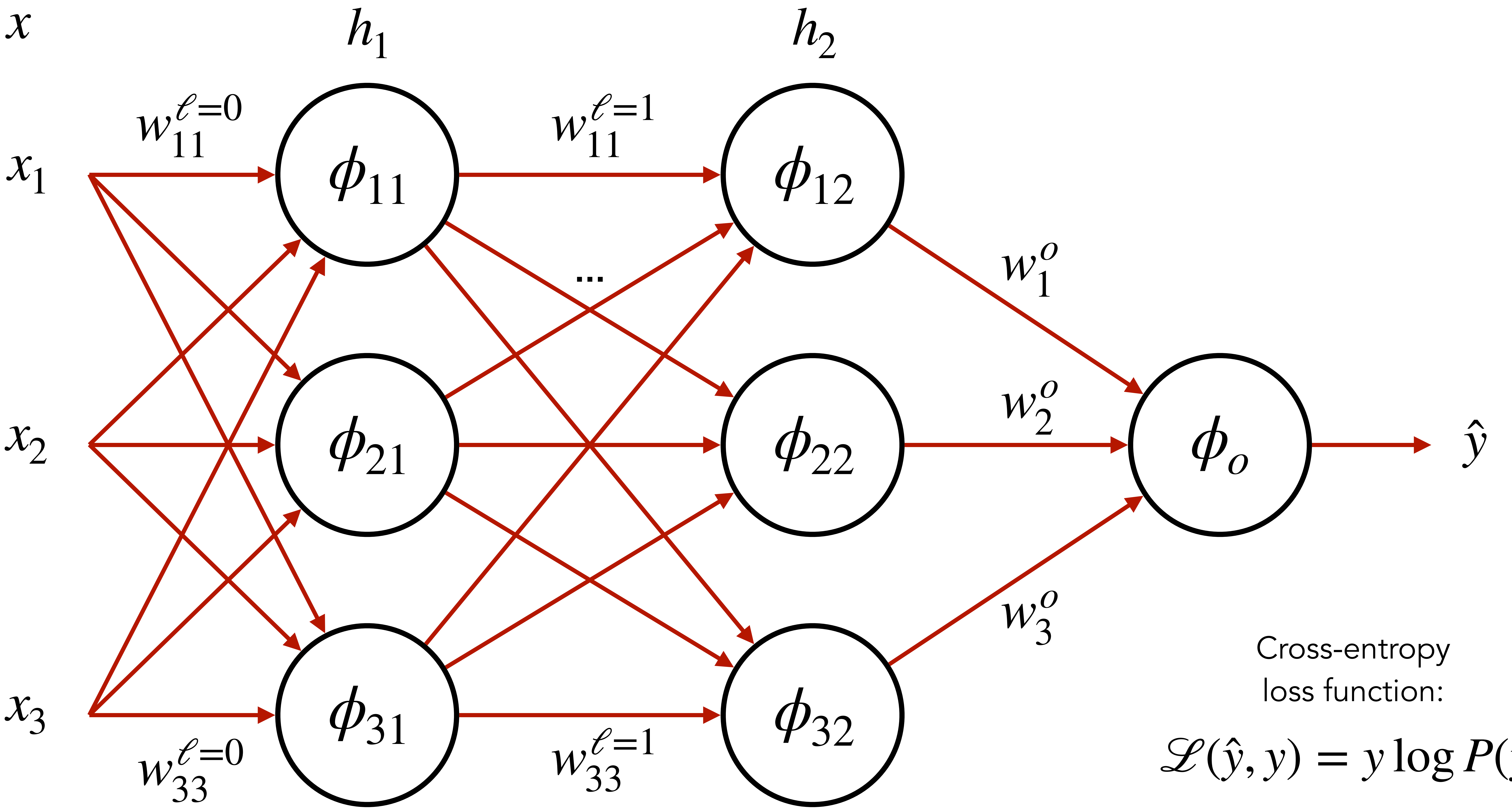
eg. pos, generation ...

What should h_0 be?

all zeros

(Elman, 1990)

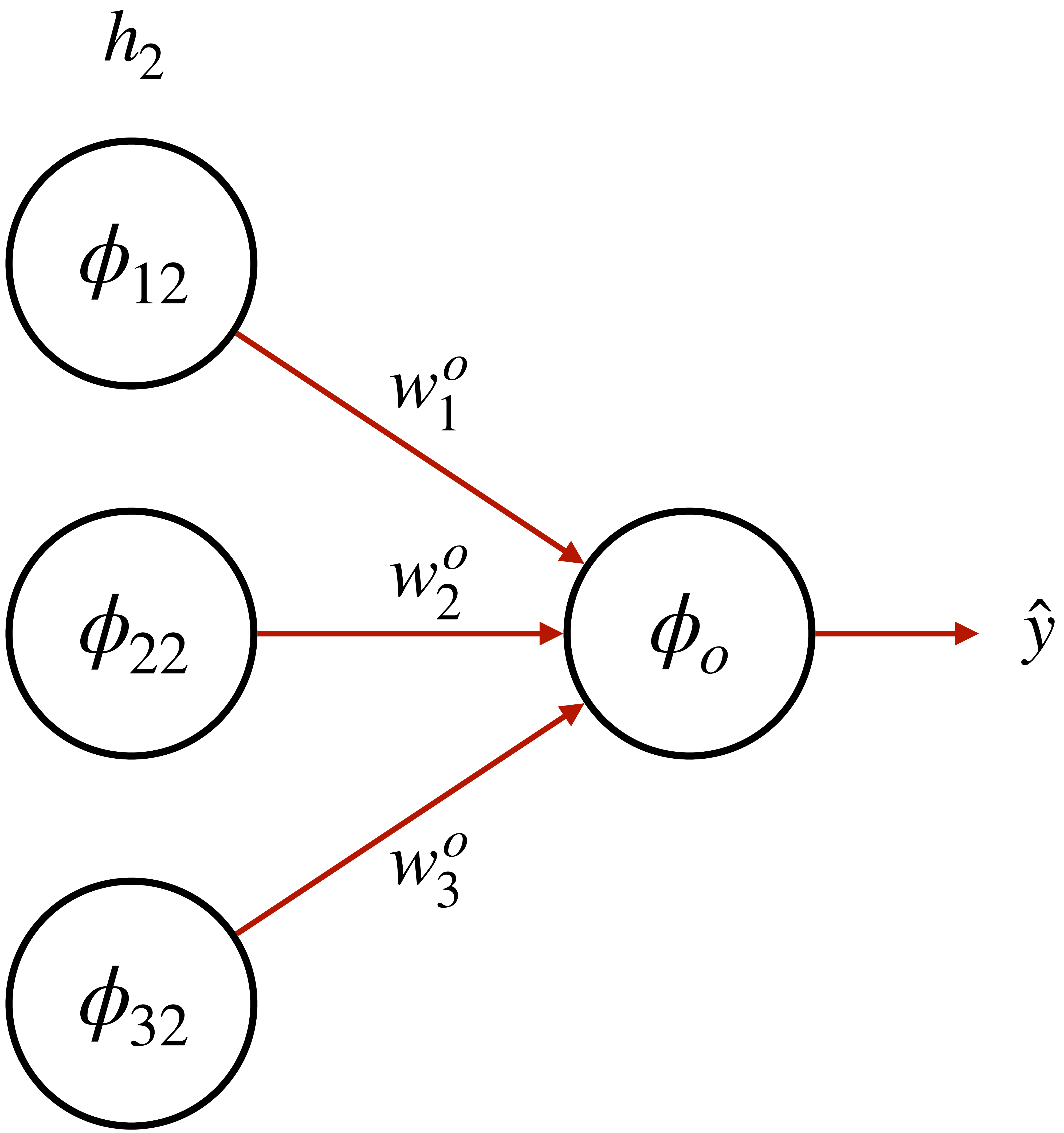
Backpropagation Review: FFNs



Cross-entropy loss function:

$$\mathscr{L}(\hat{y}, y) = y \log P(\hat{y})$$

Backpropagation Review: FFNs



$$\mathscr{L}(\hat{y}, y) = y \log P(\hat{y})$$

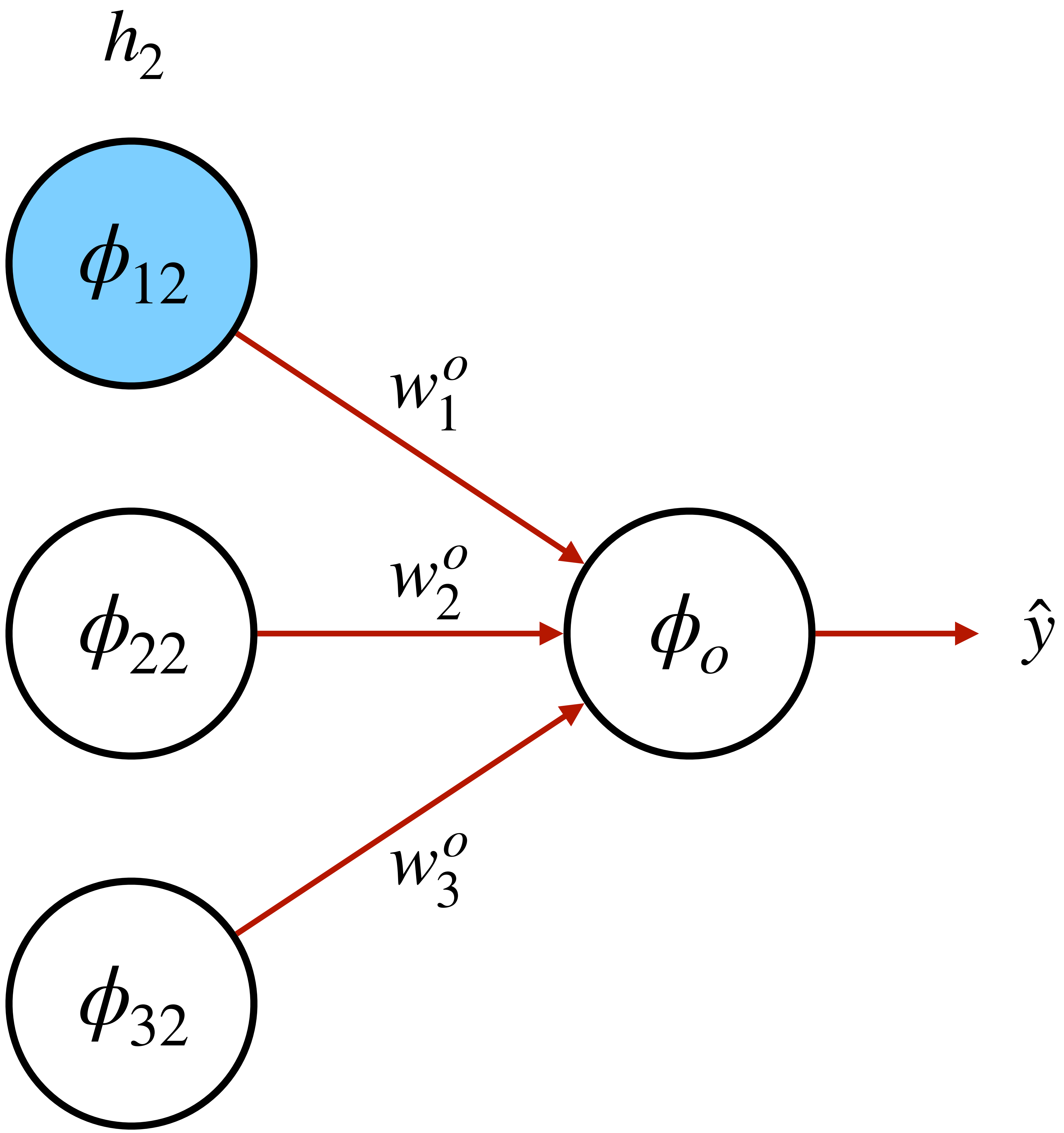
$$\hat{y} = \phi_o(u)$$

$$u = w_1^o \times \phi_{12}(\,.\,) + w_2^o \times \phi_{22}(\,.\,) + w_3^o \times \phi_{32}(\,.\,)$$

$$\frac{\partial \mathscr{L}(\hat{y}, y)}{\partial \phi_{12}(\,.\,)} = \frac{\partial \mathscr{L}(\hat{y}, y)}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial u} \frac{\partial u}{\partial \phi_{12}(\,.\,)}$$

$$= \frac{\partial \mathscr{L}(\hat{y}, y)}{\partial \hat{y}} \frac{\partial \phi_o(u)}{\partial u} w_1^0$$

Backpropagation Review: FFNs



$$\mathscr{L}(\hat{y}, y) = y \log P(\hat{y})$$

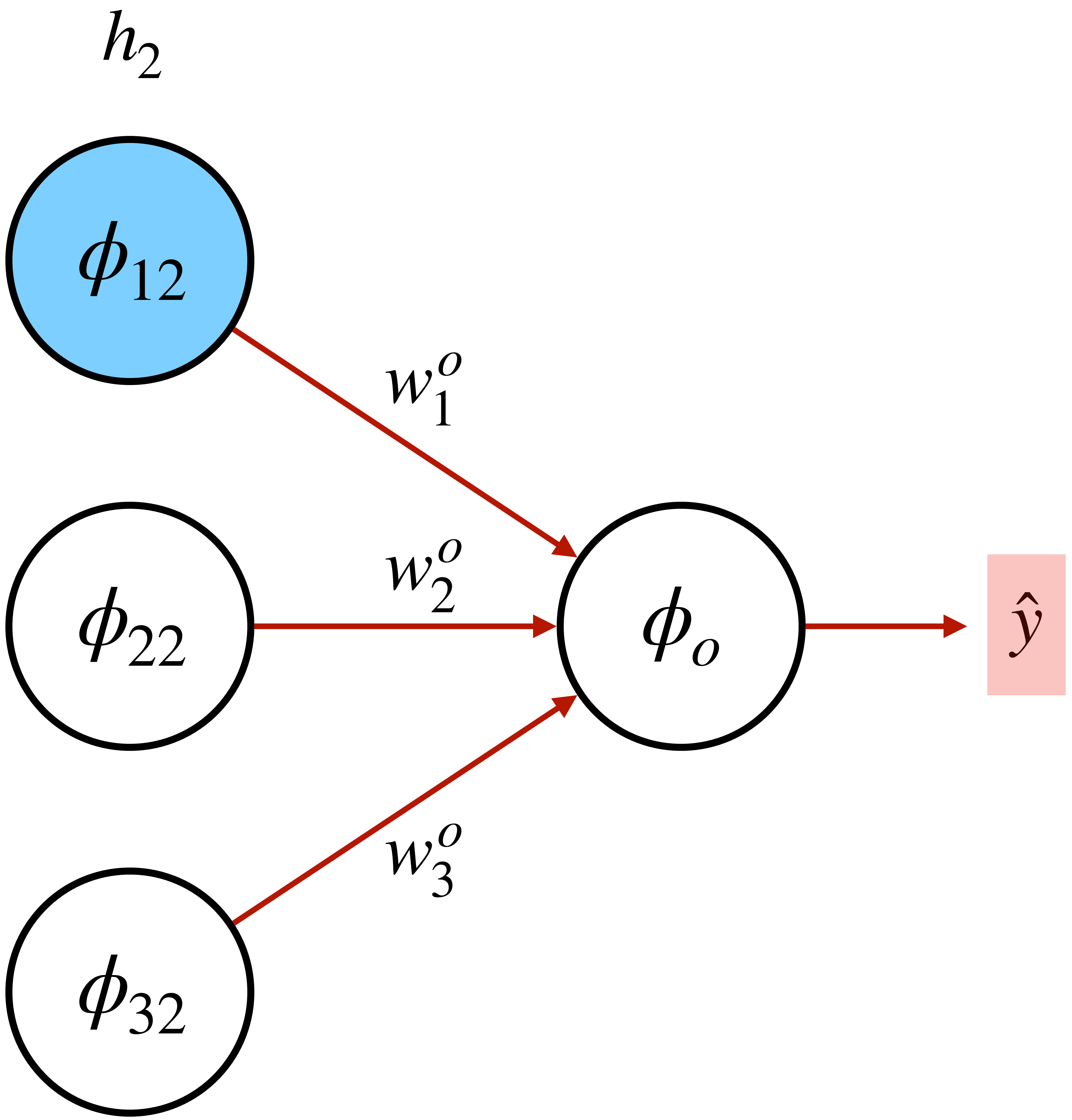
$$\hat{y} = \phi_o(u)$$

$$u = w_1^o \times \phi_{12}(\,.\,) + w_2^o \times \phi_{22}(\,.\,) + w_3^o \times \phi_{32}(\,.\,)$$

$$\frac{\partial \mathscr{L}(\hat{y}, y)}{\partial \phi_{12}(\,.\,)} = \frac{\partial \mathscr{L}(\hat{y}, y)}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial u} \frac{\partial u}{\partial \phi_{12}(\,.\,)}$$

$$= \frac{\partial \mathscr{L}(\hat{y}, y)}{\partial \hat{y}} \frac{\partial \phi_o(u)}{\partial u} w_1^0$$

Backpropagation Review: FFNs



$$\mathscr{L}(\hat{y}, y) = y \log P(\hat{y})$$

$$\hat{y} = \phi_o(u)$$

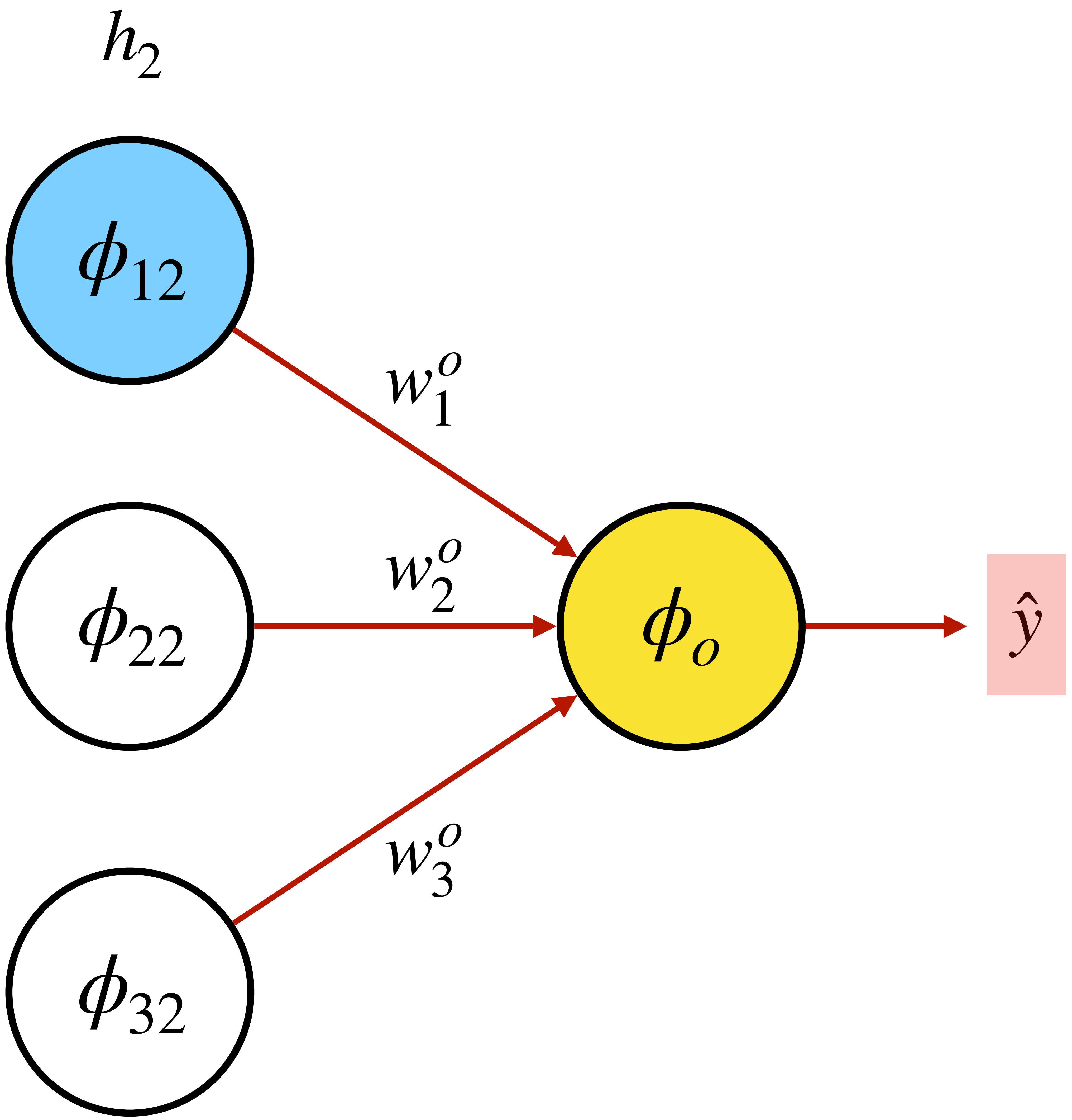
$$u = w_1^o \times \phi_{12}(\,.\,) + w_2^o \times \phi_{22}(\,.\,) + w_3^o \times \phi_{32}(\,.\,)$$

$$\frac{\partial \mathscr{L}(\hat{y}, y)}{\partial \phi_{12}(\,.\,)} = \frac{\partial \mathscr{L}(\hat{y}, y)}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial u} \frac{\partial u}{\partial \phi_{12}(\,.\,)}$$

$$= \frac{\partial \mathscr{L}(\hat{y}, y)}{\partial \hat{y}} \frac{\partial \phi_o(u)}{\partial u} w_1^0$$

Depends on label y

Backpropagation Review: FFNs



$$\mathscr{L}(\hat{y}, y) = y \log P(\hat{y}) + (1-y)\log P(1-\hat{y})$$

$$\hat{y} = \phi_o(u)$$

$$u = w_1^o \times \phi_{12}(\,.\,) + w_2^o \times \phi_{22}(\,.\,) + w_3^o \times \phi_{32}(\,.\,)$$

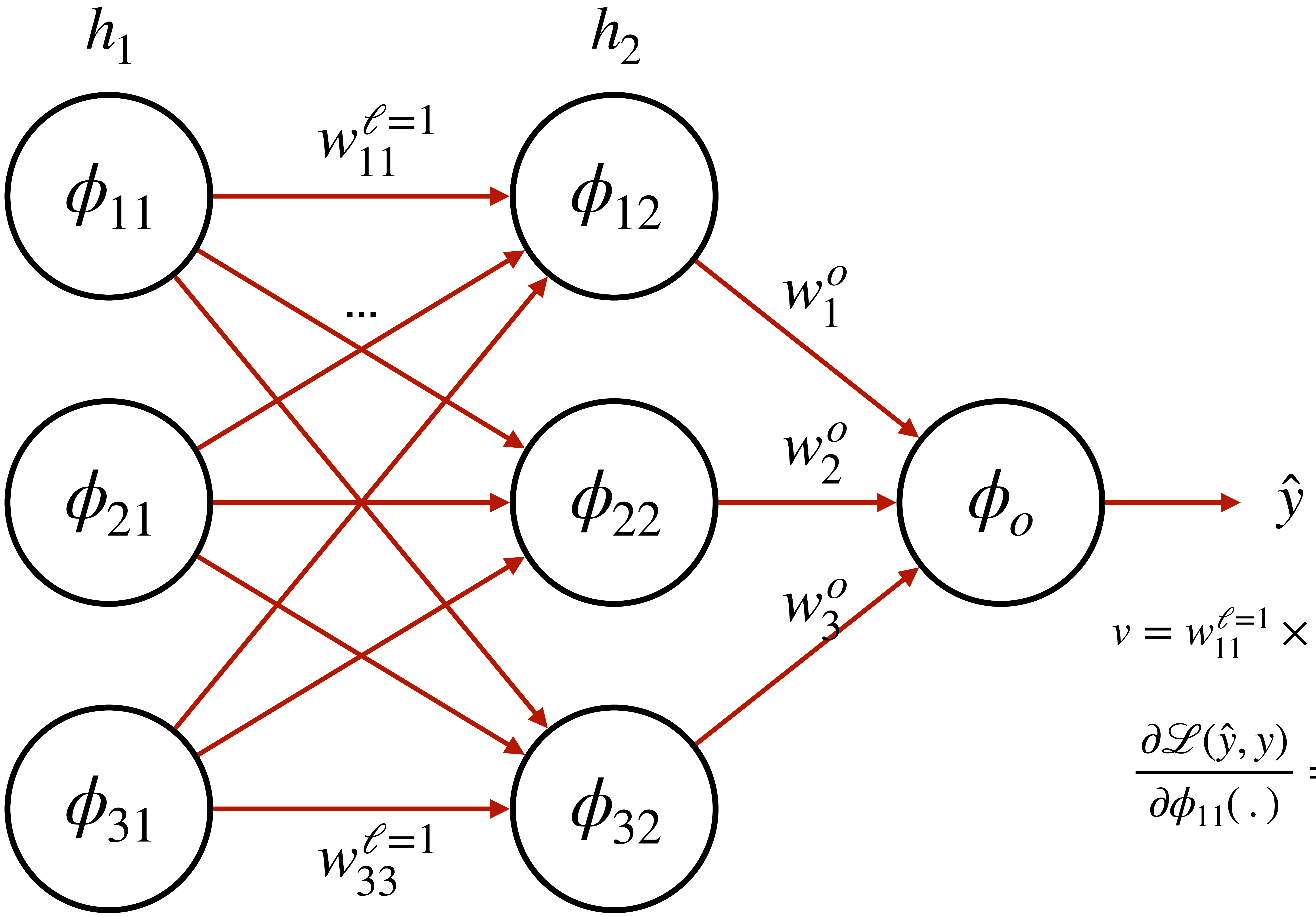
$$\frac{\partial \mathscr{L}(\hat{y}, y)}{\partial \phi_{12}(\,.\,)} = \frac{\partial \mathscr{L}(\hat{y}, y)}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial u} \frac{\partial u}{\partial \phi_{12}(\,.\,)}$$

$$= \frac{\partial \mathscr{L}(\hat{y}, y)}{\partial \hat{y}} \frac{\partial \phi_o(u)}{\partial u} w_1^0$$

Depends on label y

Depends on ϕ_o

Backpropagation Review: FFNs



$$\frac{\partial \mathcal{L}(\hat{y}, y)}{\partial \phi_{12}(\,.\,)} = \frac{\partial \mathcal{L}(\hat{y}, y)}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial u} \frac{\partial u}{\partial \phi_{12}(\,.\,)}$$

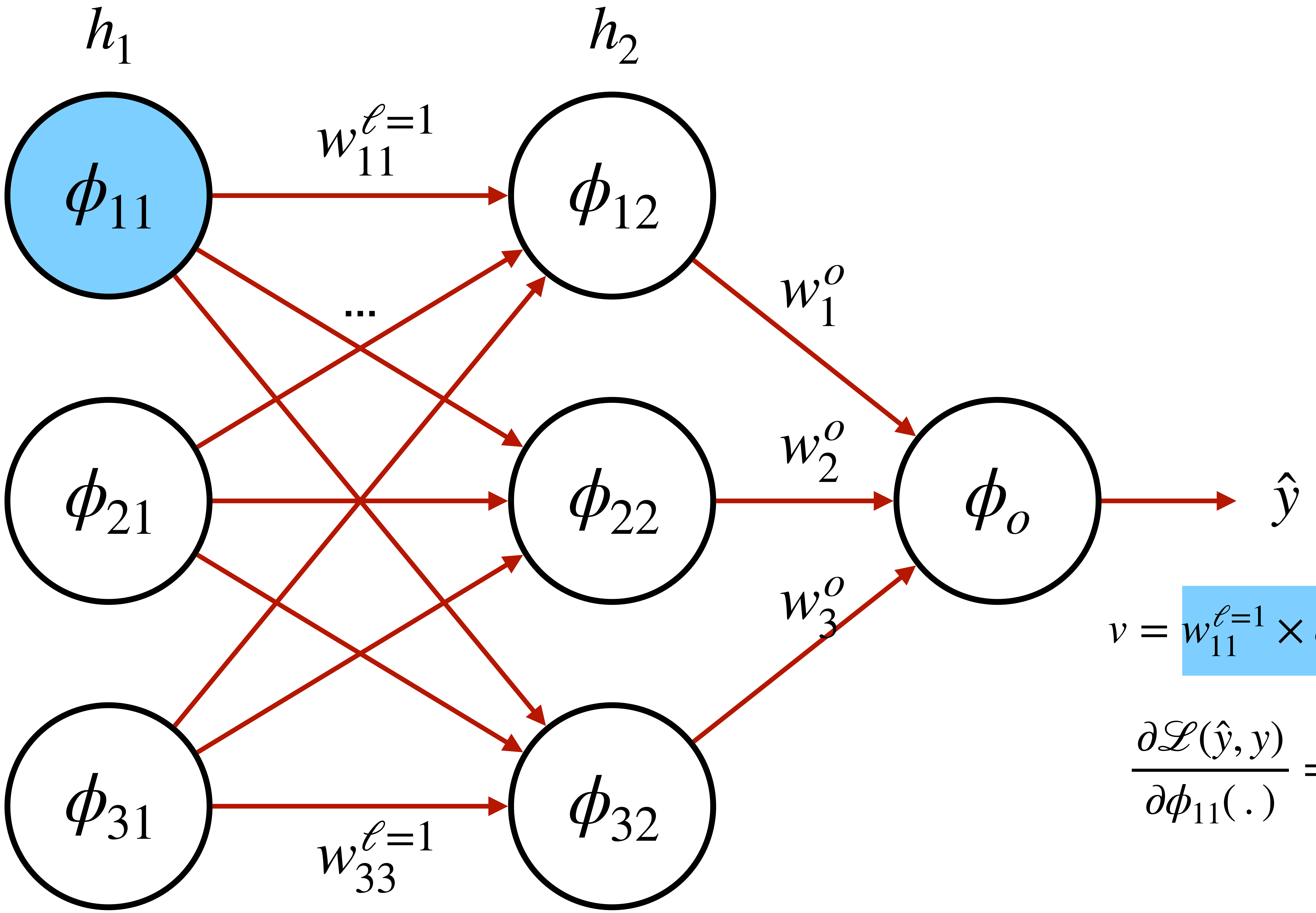
$$= \frac{\partial \mathcal{L}(\hat{y}, y)}{\partial \hat{y}} \frac{\partial \phi_o(u)}{\partial u} w_1^0$$

$$v = w_{11}^{\ell=1} \times \phi_{11}(\,.\,) + w_{21}^{\ell=1} \times \phi_{21}(\,.\,) + w_{31}^{\ell=1} \times \phi_{31}(\,.\,)$$

$$\frac{\partial \mathcal{L}(\hat{y}, y)}{\partial \phi_{11}(\,.\,)} = \frac{\partial \mathcal{L}(\hat{y}, y)}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial u} \frac{\partial u}{\partial \phi_{12}(v)} \frac{\partial \phi_{12}(v)}{\partial v} \frac{\partial v}{\partial \phi_{11}(\,.\,)}$$

$$= \frac{\partial \mathcal{L}(\hat{y}, y)}{\partial \hat{y}} \frac{\partial \phi_o(u)}{\partial u} w_1^0 \frac{\partial \phi_{12}(v)}{\partial v} w_{11}^{\ell=1}$$

Backpropagation Review: FFNs



$$\frac{\partial \mathcal{L}(\hat{y}, y)}{\partial \phi_{12}(\,.\,)} = \frac{\partial \mathcal{L}(\hat{y}, y)}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial u} \frac{\partial u}{\partial \phi_{12}(\,.\,)}$$

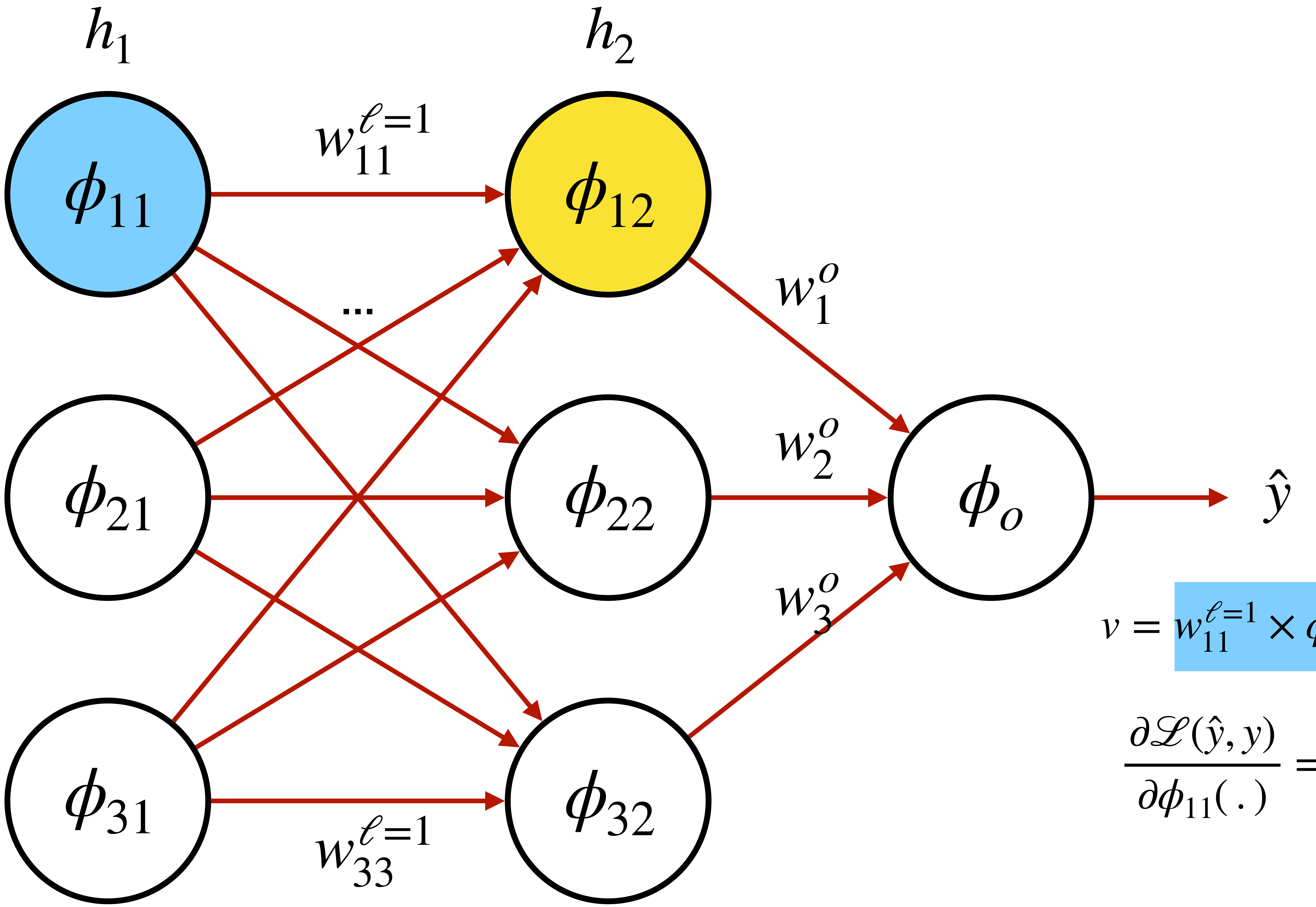
$$= \frac{\partial \mathcal{L}(\hat{y}, y)}{\partial \hat{y}} \frac{\partial \phi_o(u)}{\partial u} w_1^0$$

$$v = \boxed{w_{11}^{\ell=1} \times \phi_{11}(\,.\,)} + w_{21}^{\ell=1} \times \phi_{21}(\,.\,) + w_{31}^{\ell=1} \times \phi_{31}(\,.\,)$$

$$\frac{\partial \mathcal{L}(\hat{y}, y)}{\partial \phi_{11}(\,.\,)} = \frac{\partial \mathcal{L}(\hat{y}, y)}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial u} \frac{\partial u}{\partial \phi_{12}(v)} \frac{\partial \phi_{12}(v)}{\partial v} \boxed{\frac{\partial v}{\partial \phi_{11}(\,.\,)}}$$

$$= \frac{\partial \mathcal{L}(\hat{y}, y)}{\partial \hat{y}} \frac{\partial \phi_o(u)}{\partial u} w_1^0 \frac{\partial \phi_{12}(v)}{\partial v} \boxed{w_{11}^{\ell=1}}$$

Backpropagation Review: FFNs



$$\frac{\partial \mathcal{L}(\hat{y}, y)}{\partial \phi_{12}(\,.\,)} = \frac{\partial \mathcal{L}(\hat{y}, y)}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial u} \frac{\partial u}{\partial \phi_{12}(\,.\,)}$$

$$= \frac{\partial \mathcal{L}(\hat{y}, y)}{\partial \hat{y}} \frac{\partial \phi_o(u)}{\partial u} w_1^0$$

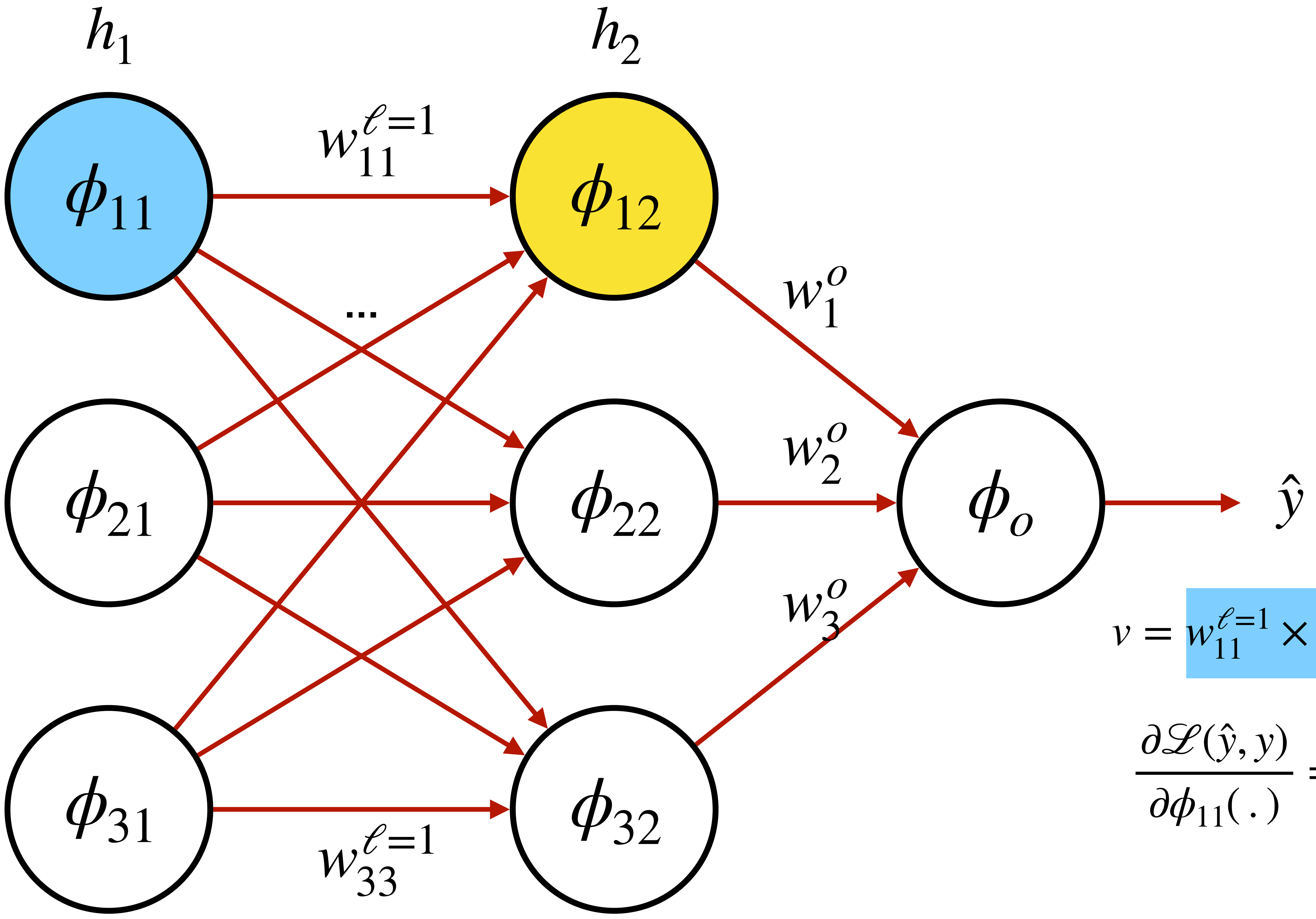
Depends on ϕ_{12}

$$v = w_{11}^{\ell=1} \times \phi_{11}(\,.\,) + w_{21}^{\ell=1} \times \phi_{21}(\,.\,) + w_{31}^{\ell=1} \times \phi_{31}(\,.\,)$$

$$\frac{\partial \mathcal{L}(\hat{y}, y)}{\partial \phi_{11}(\,.\,)} = \frac{\partial \mathcal{L}(\hat{y}, y)}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial u} \frac{\partial u}{\partial \phi_{12}(v)} \frac{\partial \phi_{12}(v)}{\partial v} \frac{\partial v}{\partial \phi_{11}(\,.\,)}$$

$$= \frac{\partial \mathcal{L}(\hat{y}, y)}{\partial \hat{y}} \frac{\partial \phi_o(u)}{\partial u} w_1^0 \frac{\partial \phi_{12}(v)}{\partial v} w_{11}^{\ell=1}$$

Backpropagation Review: FFNs



$$\frac{\partial \mathscr{L}(\hat{y}, y)}{\partial \phi_{12}(\,.\,)} = \frac{\partial \mathscr{L}(\hat{y}, y)}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial u} \frac{\partial u}{\partial \phi_{12}(\,.\,)}$$

$$= \frac{\partial \mathscr{L}(\hat{y}, y)}{\partial \hat{y}} \frac{\partial \phi_o(u)}{\partial u} w_1^0$$

Depends on ϕ_{12}

$$v = w_{11}^{\ell=1} \times \phi_{11}(\,.\,) + w_{21}^{\ell=1} \times \phi_{21}(\,.\,) + w_{31}^{\ell=1} \times \phi_{31}(\,.\,)$$

$$\frac{\partial \mathscr{L}(\hat{y}, y)}{\partial \phi_{11}(\,.\,)} = \frac{\partial \mathscr{L}(\hat{y}, y)}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial u} \frac{\partial u}{\partial \phi_{12}(v)} \frac{\partial \phi_{12}(v)}{\partial v} \frac{\partial v}{\partial \phi_{11}(\,.\,)}$$

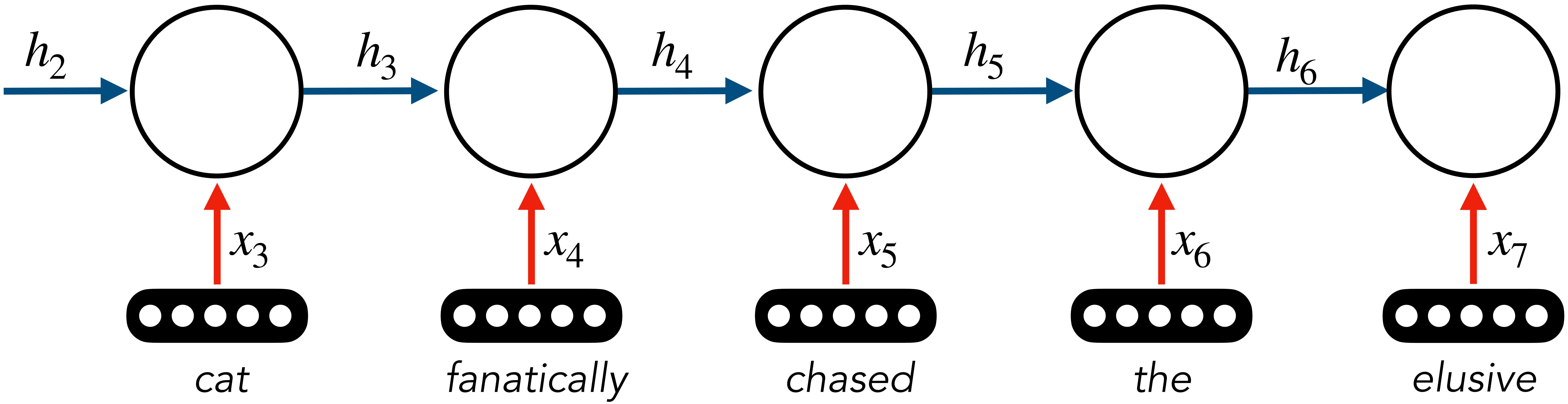
$$= \frac{\partial \mathscr{L}(\hat{y}, y)}{\partial \hat{y}} \frac{\partial \phi_o(u)}{\partial u} w_1^0 \frac{\partial \phi_{12}(v)}{\partial v} w_{11}^{\ell=1}$$

Question

How would we extend backpropagation to a recurrent neural network?

Recall

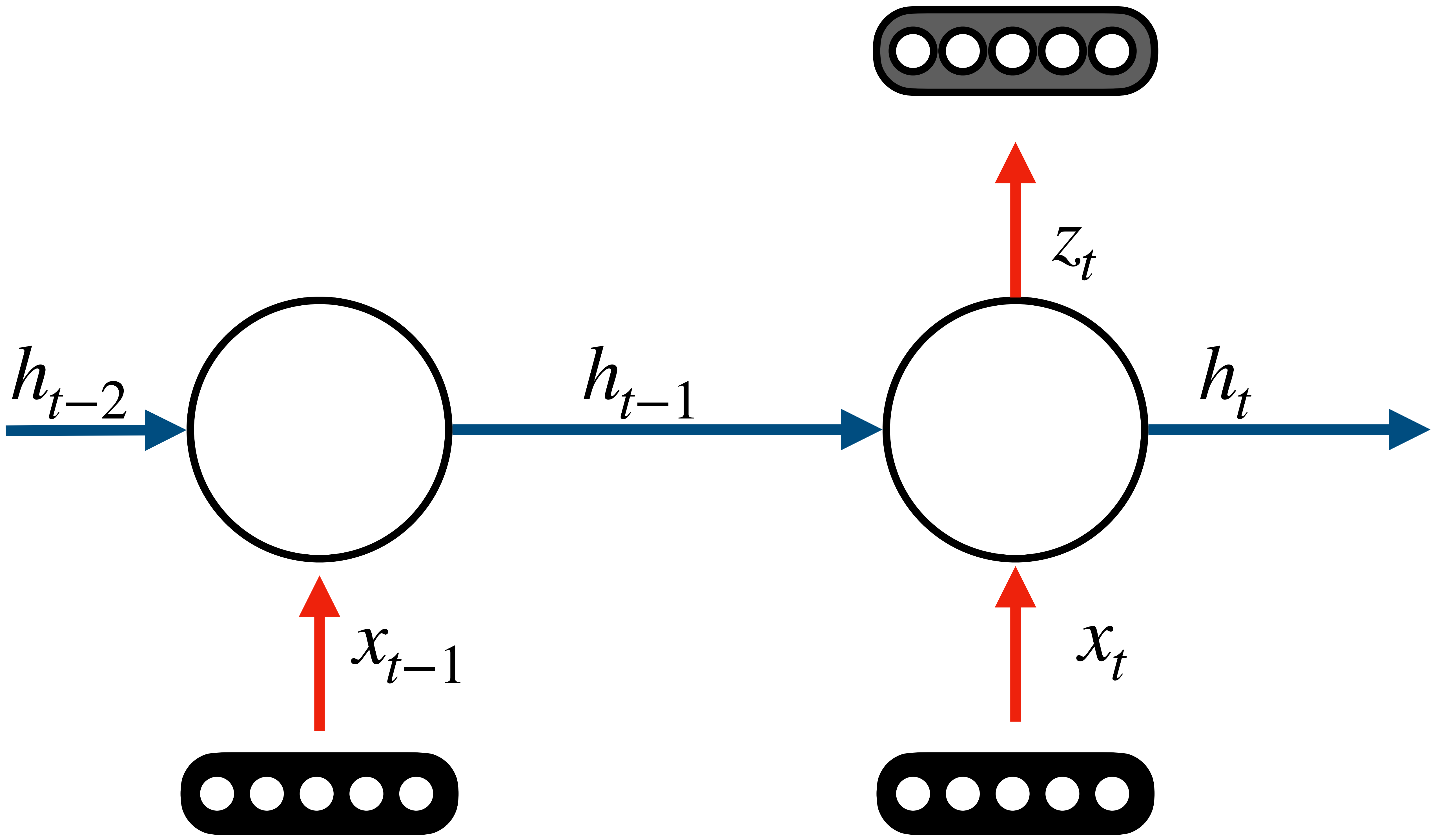
- RNN can be unrolled to a feedforward neural network
- Depth of feedforward neural network depends on length of the sequence



Backpropagation through Time

$$z_t = \sigma\left(W_{zh}h_t + b_z\right)$$

$$h_t = \sigma\left(W_{hx}x_t + W_{hh}h_{t-1} + b_h\right)$$



Backpropagation through Time

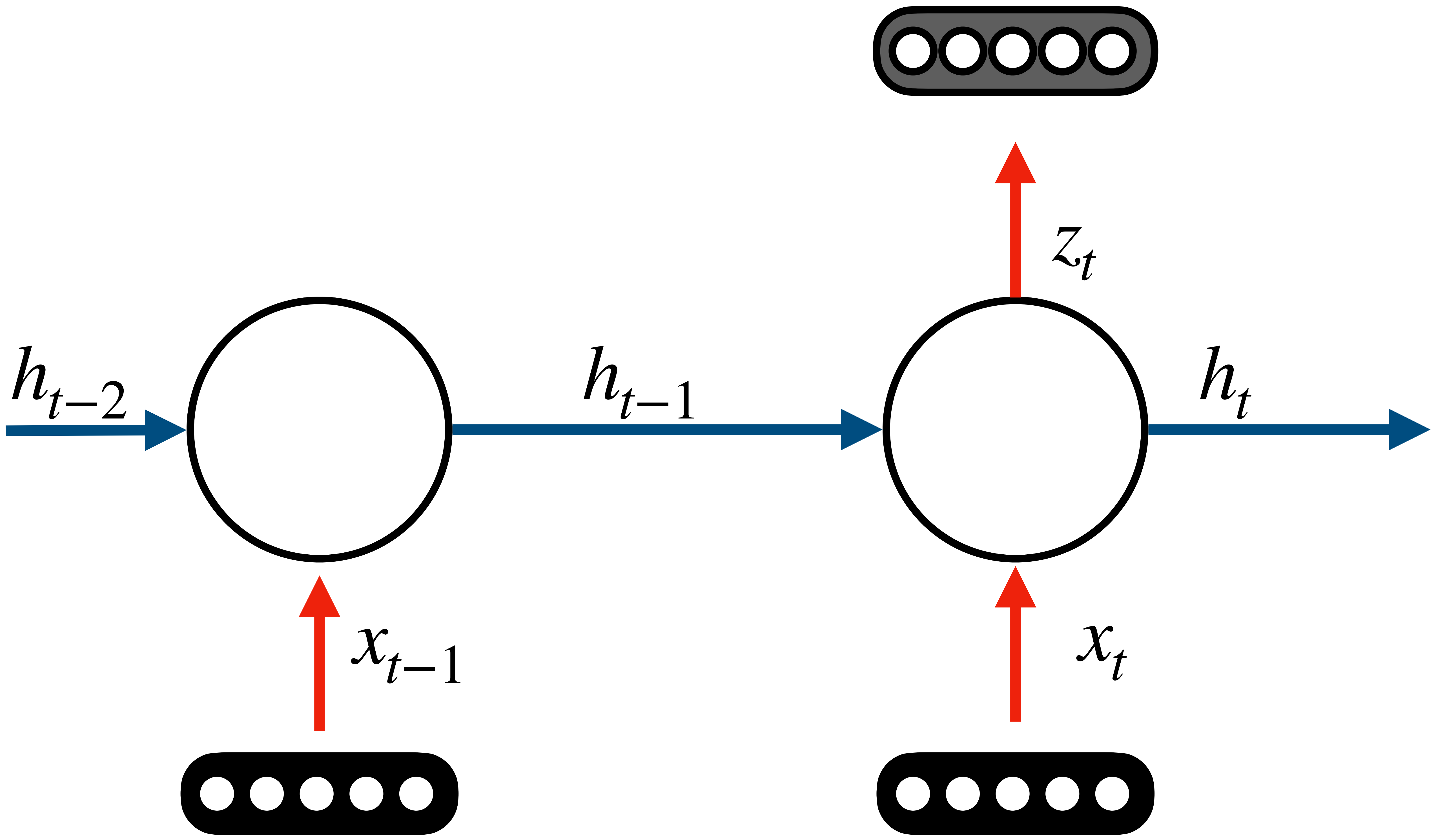
$$z_t = \sigma\left(W_{zh}h_t + b_z\right)$$

$$h_t = \sigma\left(W_{hx}x_t + W_{hh}h_{t-1} + b_h\right)$$

$$v = W_{zh}h_t + b_z \qquad z_t = \sigma(v)$$

$$u = W_{hx}x_t + W_{hh}h_{t-1} + b_h \qquad h_t = \sigma(u)$$

$$\frac{\partial z_t}{\partial h_t} = \frac{\partial \sigma(v)}{\partial v}\frac{\partial v}{\partial h_t} = \frac{\partial \sigma(v)}{\partial v}W_{zh}$$



Backpropagation through Time

$$z_t = \sigma\left(W_{zh}h_t + b_z\right)$$

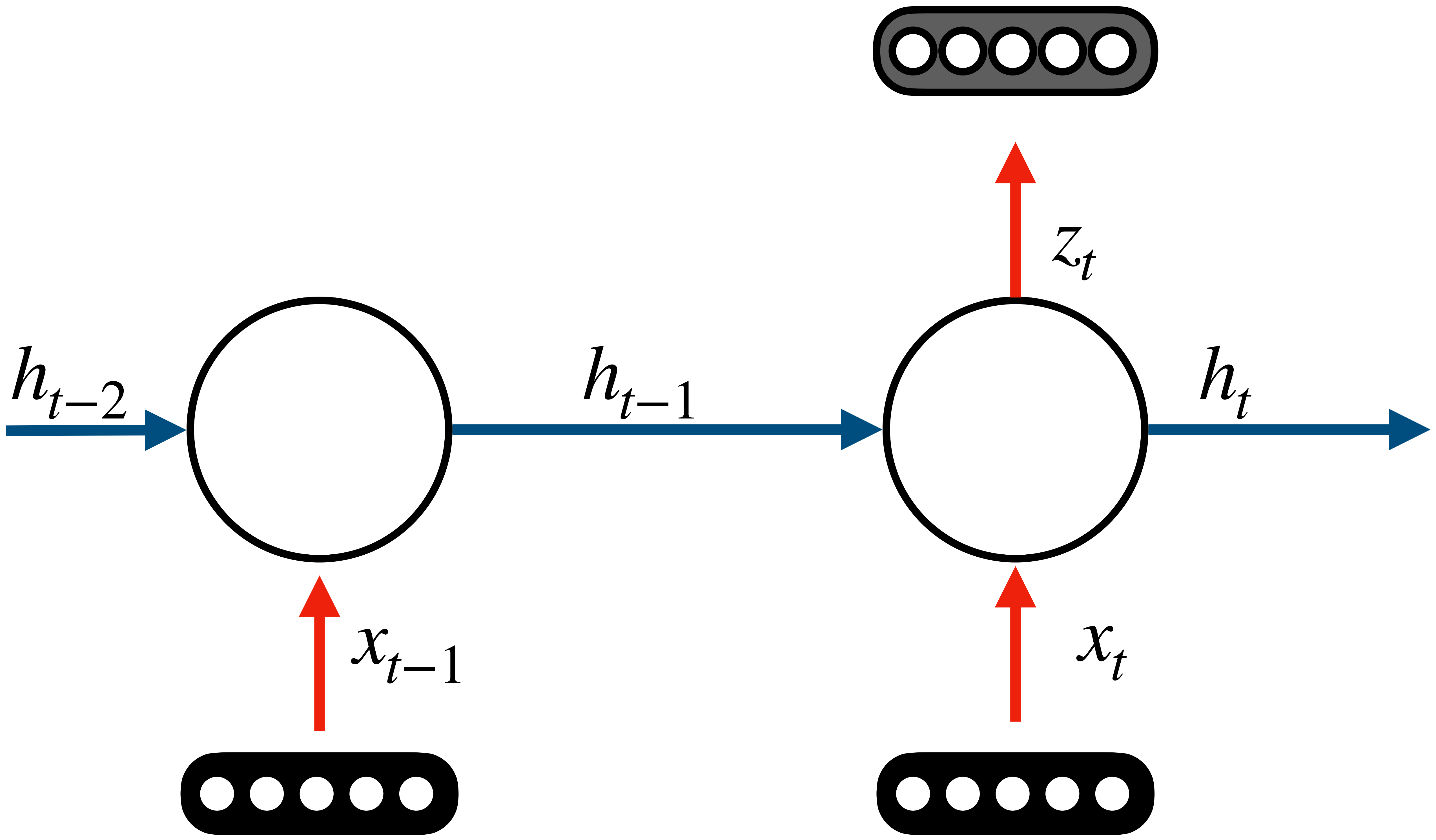
$$h_t = \sigma\left(W_{hx}x_t + W_{hh}h_{t-1} + b_h\right)$$

$$v = W_{zh}h_t + b_z \qquad z_t = \sigma(v)$$

$$u = W_{hx}x_t + W_{hh}h_{t-1} + b_h \qquad h_t = \sigma(u)$$

$$\frac{\partial z_t}{\partial h_t} = \frac{\partial \sigma(v)}{\partial v}\frac{\partial v}{\partial h_t} = \frac{\partial \sigma(v)}{\partial v}W_{zh}$$

$$\frac{\partial h_t}{\partial h_{t-1}} = \frac{\partial \sigma(u)}{\partial u}\frac{\partial u}{\partial h_{t-1}} = \frac{\partial \sigma(u)}{\partial u}W_{hh}$$



Backpropagation through Time

$$z_t = \sigma\left(W_{zh}h_t + b_z\right)$$

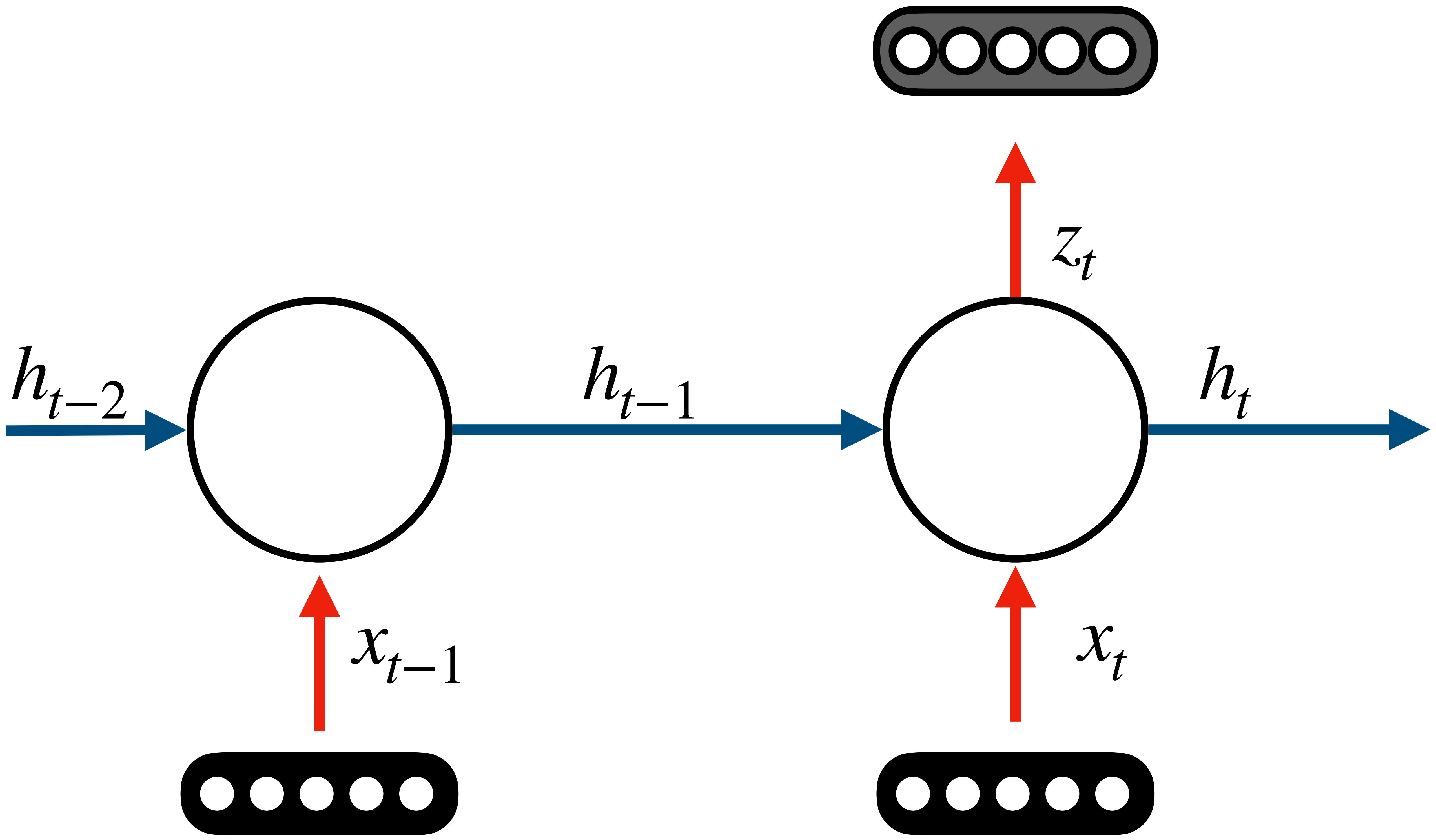
$$h_t = \sigma\left(W_{hx}x_t + W_{hh}h_{t-1} + b_h\right)$$

$$v = W_{zh}h_t + b_z \qquad z_t = \sigma(v)$$

$$u = W_{hx}x_t + W_{hh}h_{t-1} + b_h \qquad h_t = \sigma(u)$$

$$\frac{\partial z_t}{\partial h_t} = \frac{\partial \sigma(v)}{\partial v}\frac{\partial v}{\partial h_t} = \frac{\partial \sigma(v)}{\partial v}W_{zh}$$

$$\frac{\partial h_t}{\partial h_{t-1}} = \frac{\partial \sigma(u)}{\partial u}\frac{\partial u}{\partial h_{t-1}} = \frac{\partial \sigma(u)}{\partial u}W_{hh}$$



$$\frac{\partial z_t}{\partial h_{t-1}} = \frac{\partial z_t}{\partial h_t}\frac{\partial h_t}{\partial h_{t-1}}$$

Backpropagation through Time

$$z_t = \sigma\left(W_{zh}h_t + b_z\right)$$

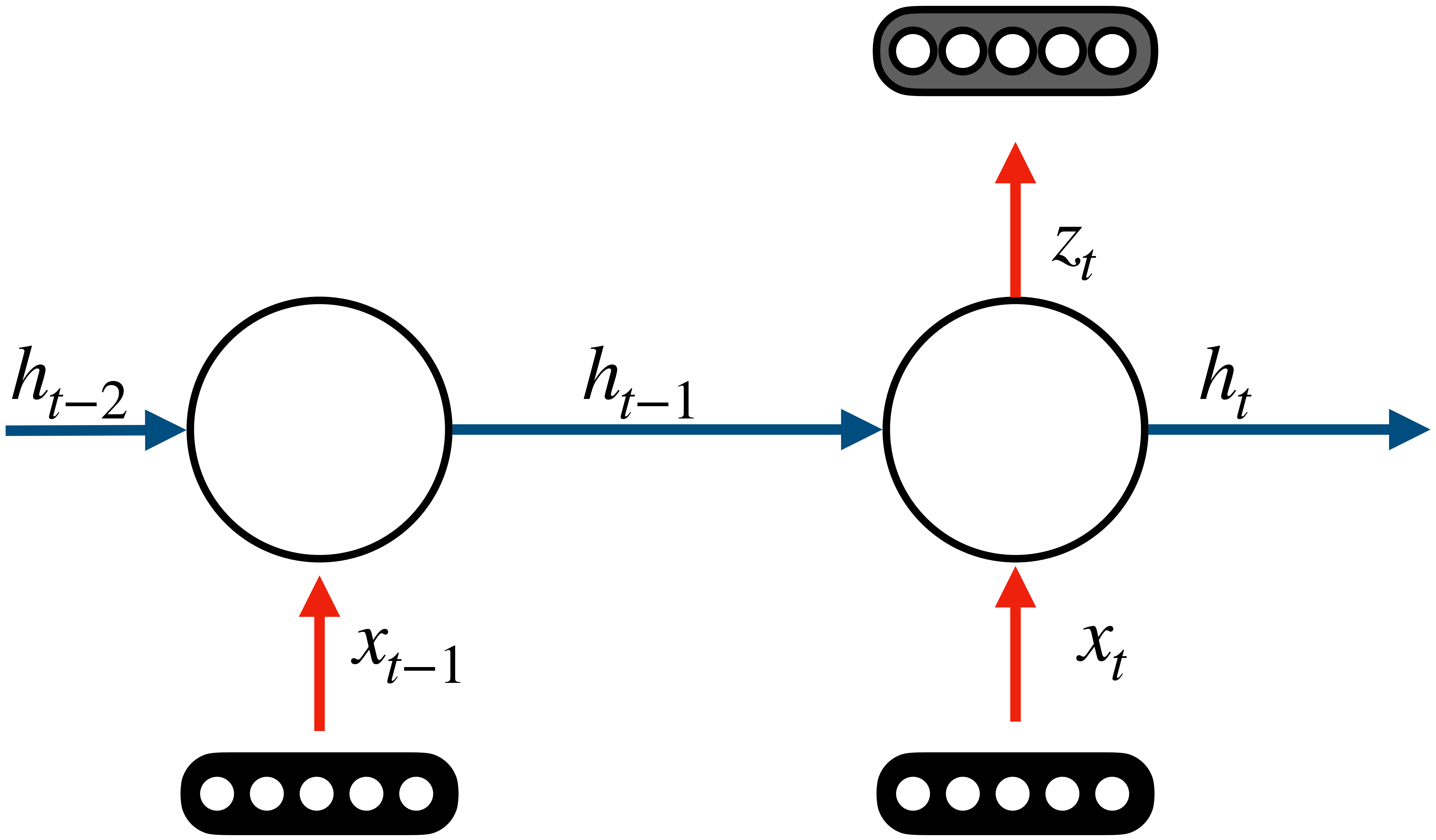
$$h_t = \sigma\left(W_{hx}x_t + W_{hh}h_{t-1} + b_h\right)$$

$$v = W_{zh}h_t + b_z \qquad z_t = \sigma(v)$$

$$u = W_{hx}x_t + W_{hh}h_{t-1} + b_h \qquad h_t = \sigma(u)$$

$$\frac{\partial z_t}{\partial h_t} = \frac{\partial \sigma(v)}{\partial v}\frac{\partial v}{\partial h_t} = \frac{\partial \sigma(v)}{\partial v}W_{zh}$$

$$\frac{\partial h_t}{\partial h_{t-1}} = \frac{\partial \sigma(u)}{\partial u}\frac{\partial u}{\partial h_{t-1}} = \frac{\partial \sigma(u)}{\partial u}W_{hh}$$



$$\frac{\partial z_t}{\partial h_{t-1}} = \frac{\partial z_t}{\partial h_t}\frac{\partial h_t}{\partial h_{t-1}} = \frac{\partial \sigma(v)}{\partial v}\frac{\partial v}{\partial h_t}\frac{\partial \sigma(u)}{\partial u}\frac{\partial u}{\partial h_{t-1}} = \frac{\partial \sigma(v)}{\partial v}W_{zh}\frac{\partial \sigma(u)}{\partial u}W_{hh}$$

Backpropagation through Time

$$z_t = \sigma\left(W_{zh}h_t + b_z\right)$$

$$h_t = \sigma\left(W_{hx}x_t + W_{hh}h_{t-1} + b_h\right)$$

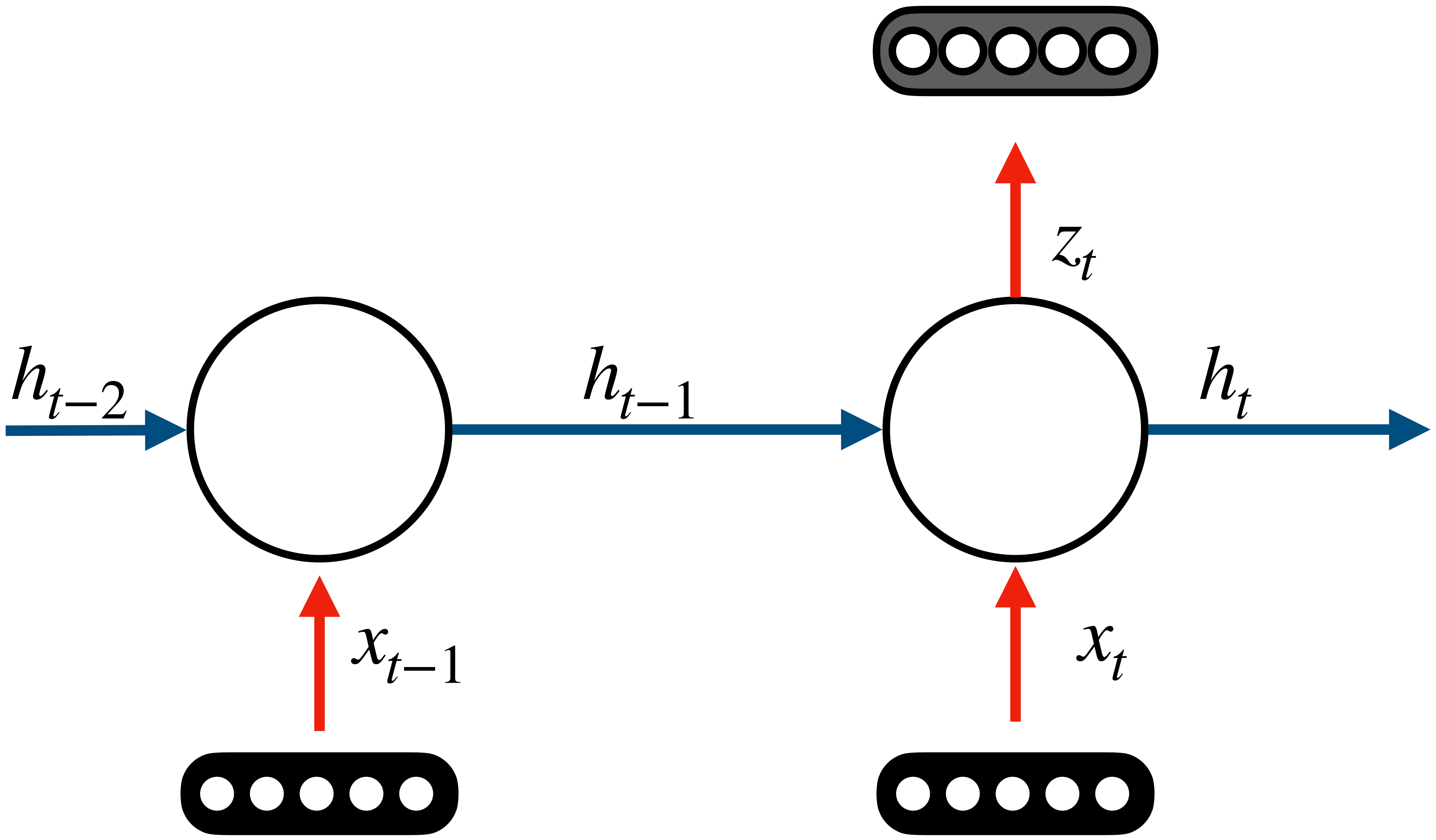
$$v_t = W_{zh}h_t + b_z \qquad z_t = \sigma(v_t)$$

$$u_t = W_{hx}x_t + W_{hh}h_{t-1} + b_h \qquad h_t = \sigma(u_t)$$

$$\frac{\partial z_t}{\partial h_t} = \frac{\partial \sigma(v_t)}{\partial v_t}\frac{\partial v_t}{\partial h_t} = \frac{\partial \sigma(v_t)}{\partial v_t}W_{zh}$$

$$\frac{\partial h_t}{\partial h_{t-1}} = \frac{\partial \sigma(u_t)}{\partial u_t}\frac{\partial u_t}{\partial h_{t-1}} = \frac{\partial \sigma(u_t)}{\partial u_t}W_{hh}$$

$$\frac{\partial h_{t-1}}{\partial h_{t-2}} = \frac{\partial \sigma(u_{t-1})}{\partial u_{t-1}}\frac{\partial u_{t-1}}{\partial h_{t-2}} = \frac{\partial \sigma(u_{t-1})}{\partial u_{t-1}}W_{hh}$$



$$\frac{\partial z_t}{\partial h_{t-1}} = \frac{\partial z_t}{\partial h_t}\frac{\partial h_t}{\partial h_{t-1}}\frac{\partial h_{t-1}}{\partial h_{t-2}} = \frac{\partial \sigma(v_t)}{\partial v_t}W_{zh}\frac{\partial \sigma(u_t)}{\partial u_t}W_{hh}\frac{\partial \sigma(u_{t-1})}{\partial u_{t-1}}W_{hh}$$

Backpropagation through Time

$$z_t = \sigma\left(W_{zh}h_t + b_z\right)$$

$$h_t = \sigma\left(W_{hx}x_t + W_{hh}h_{t-1} + b_h\right)$$

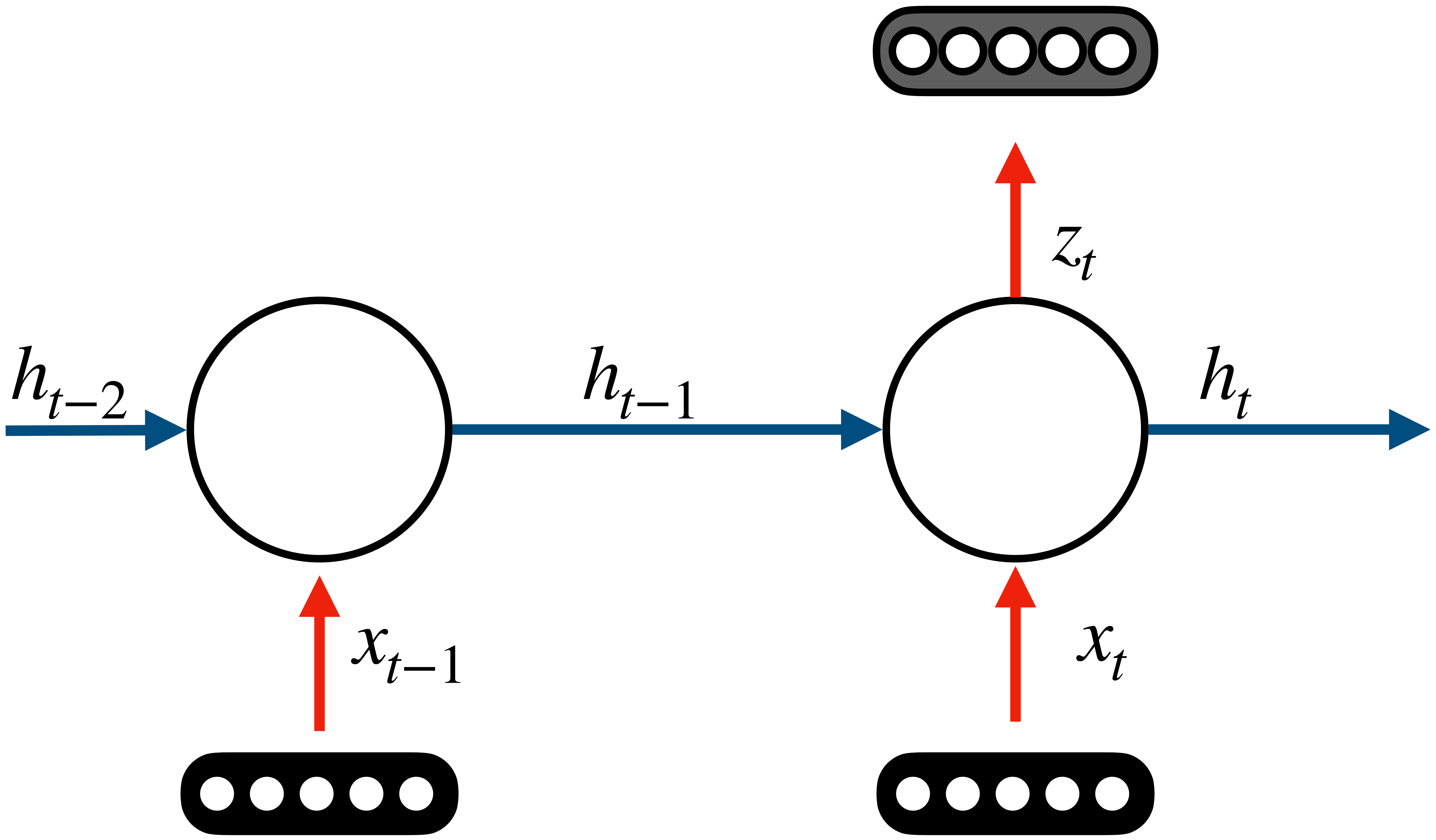
$$v_t = W_{zh}h_t + b_z \qquad z_t = \sigma(v_t)$$

$$u_t = W_{hx}x_t + W_{hh}h_{t-1} + b_h \qquad h_t = \sigma(u_t)$$

$$\frac{\partial z_t}{\partial h_t} = \frac{\partial \sigma(v_t)}{\partial v_t}\frac{\partial v_t}{\partial h_t} = \frac{\partial \sigma(v_t)}{\partial v_t}W_{zh}$$

$$\frac{\partial h_t}{\partial h_{t-1}} = \frac{\partial \sigma(u_t)}{\partial u_t}\frac{\partial u_t}{\partial h_{t-1}} = \frac{\partial \sigma(u_t)}{\partial u_t}W_{hh}$$

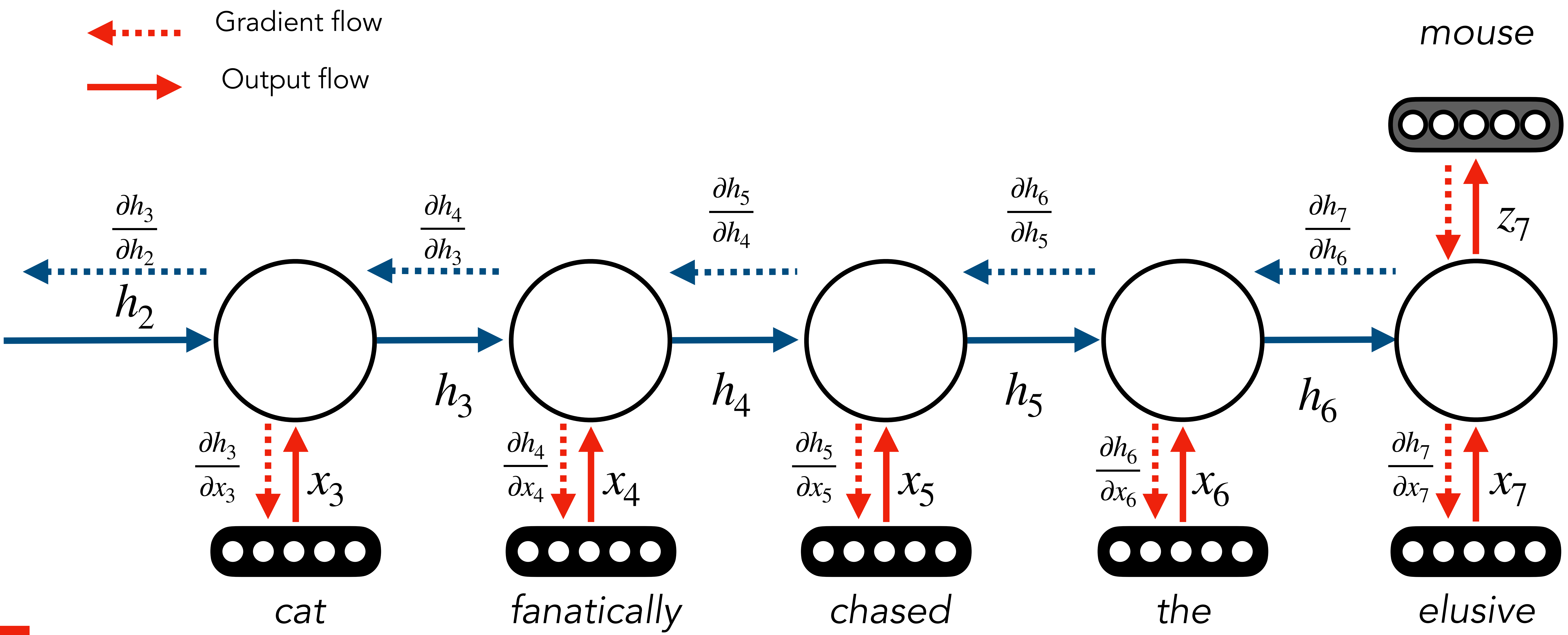
$$\frac{\partial h_{t-1}}{\partial h_{t-2}} = \frac{\partial \sigma(u_{t-1})}{\partial u_{t-1}}\frac{\partial u_{t-1}}{\partial h_{t-2}} = \frac{\partial \sigma(u_{t-1})}{\partial u_{t-1}}W_{hh}$$



$$\frac{\partial z_t}{\partial h_{t-1}} = \frac{\partial z_t}{\partial h_t}\frac{\partial h_t}{\partial h_{t-1}}\frac{\partial h_{t-1}}{\partial h_{t-2}} = \frac{\partial \sigma(v_t)}{\partial v_t}W_{zh}\frac{\partial \sigma(u_t)}{\partial u_t}W_{hh}\frac{\partial \sigma(u_{t-1})}{\partial u_{t-1}}W_{hh}$$

Note that these are
actually the same matrix

Backpropagation through time



Recap

- Neural language models allow us to *share information* among similar sequences by learning neural representations that similarly represent them

- **Problem:** Fixed context language models can only process a limited window of the word history at a time

- **Solution:** recurrent neural networks can **theoretically** learn to model an **unbounded context length**

- **Next Class:** Training recurrent neural networks, associated challenges and mitigations through gated recurrent networks