# Recurrent Neural Networks

Antoine Bosselut

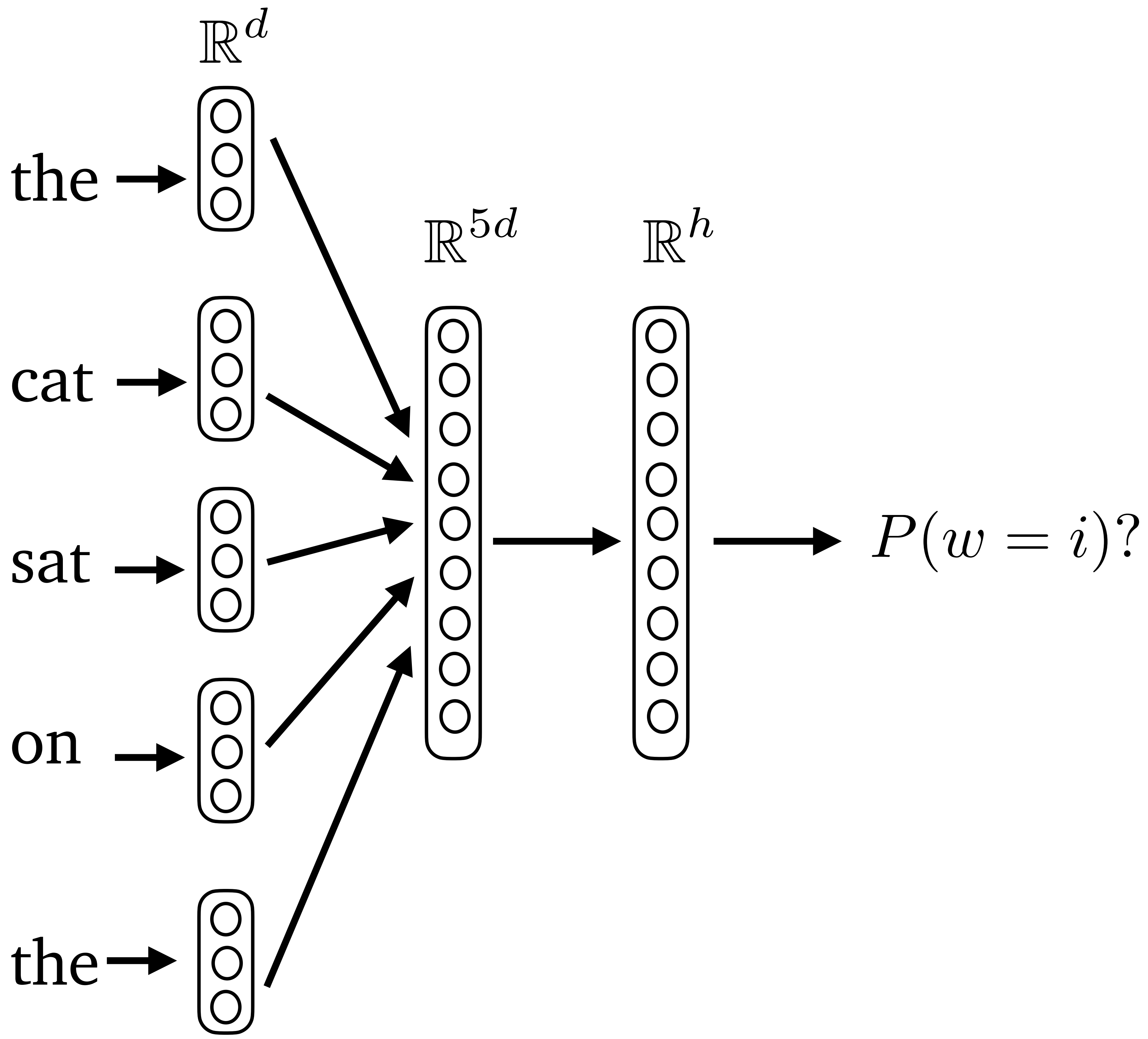
EPFL

# Section Outline

- **Fixing the context bottleneck:** recurrent neural networks
- **Training recurrent neural networks:** backpropagation through time
- **Training Challenges:** Vanishing Gradients

# 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)

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

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



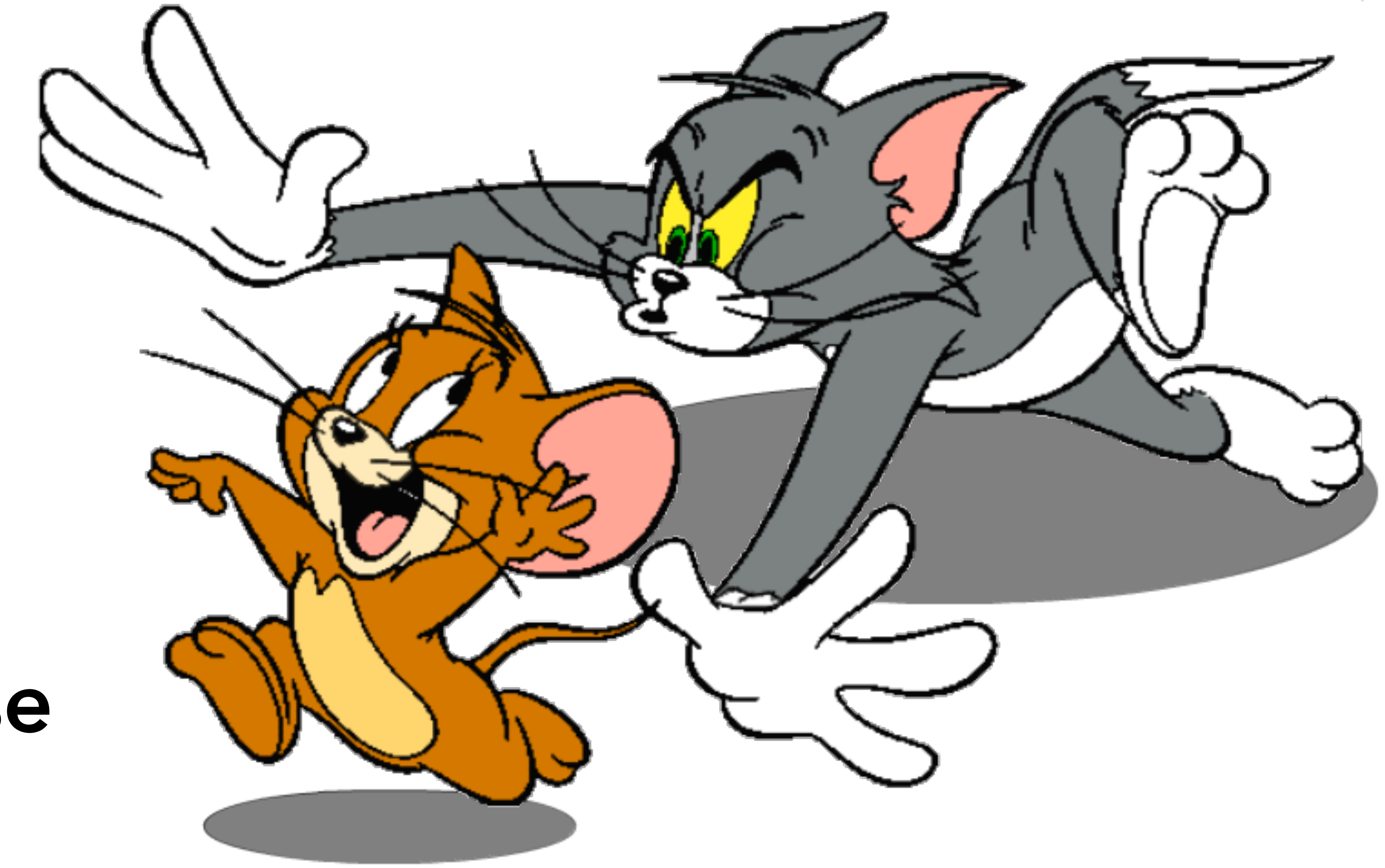
(Bengio et al., 2003): A Neural Probabilistic Language Model

# 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**

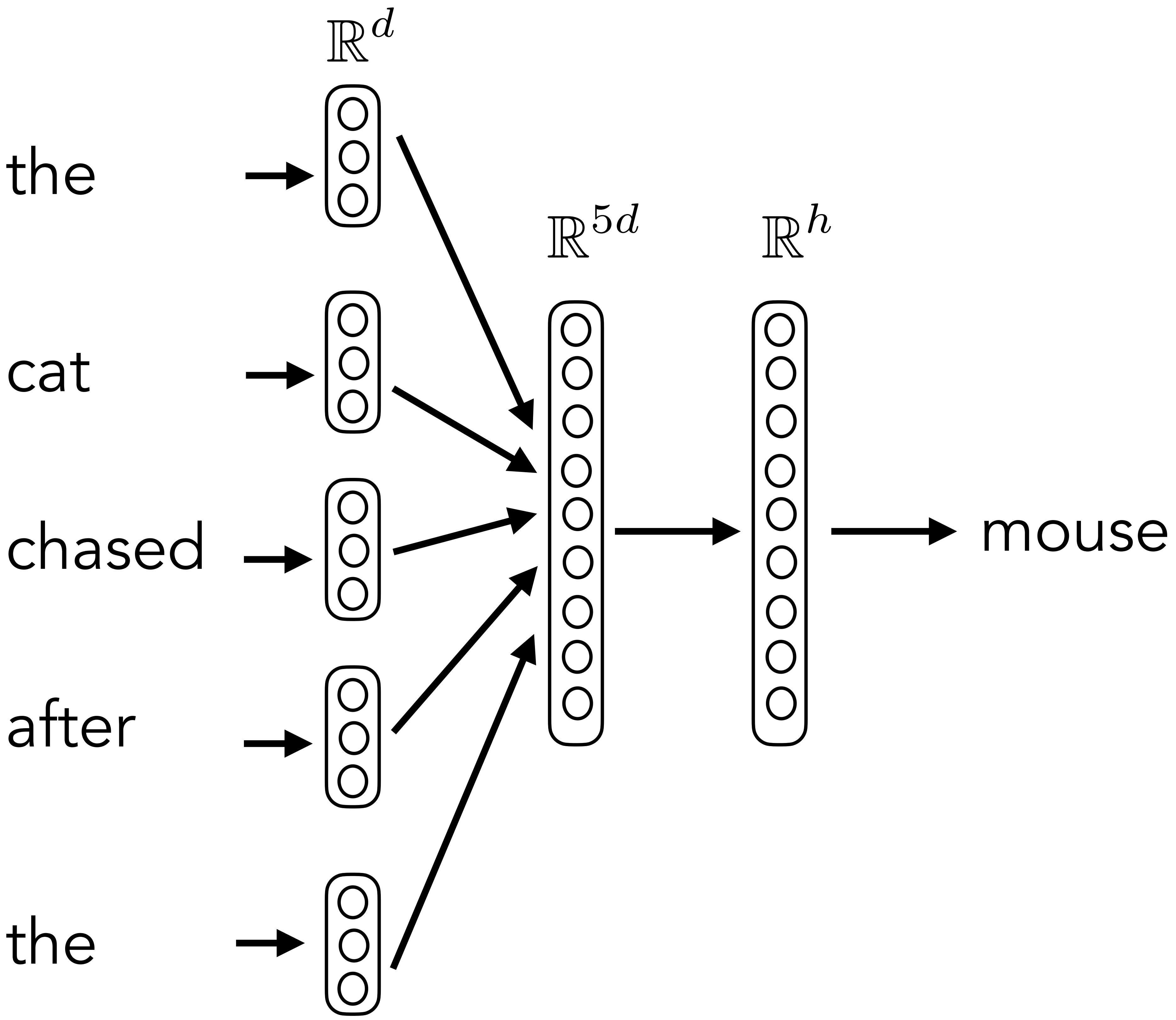
The starving **cat** chased the **mouse**

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) ✓

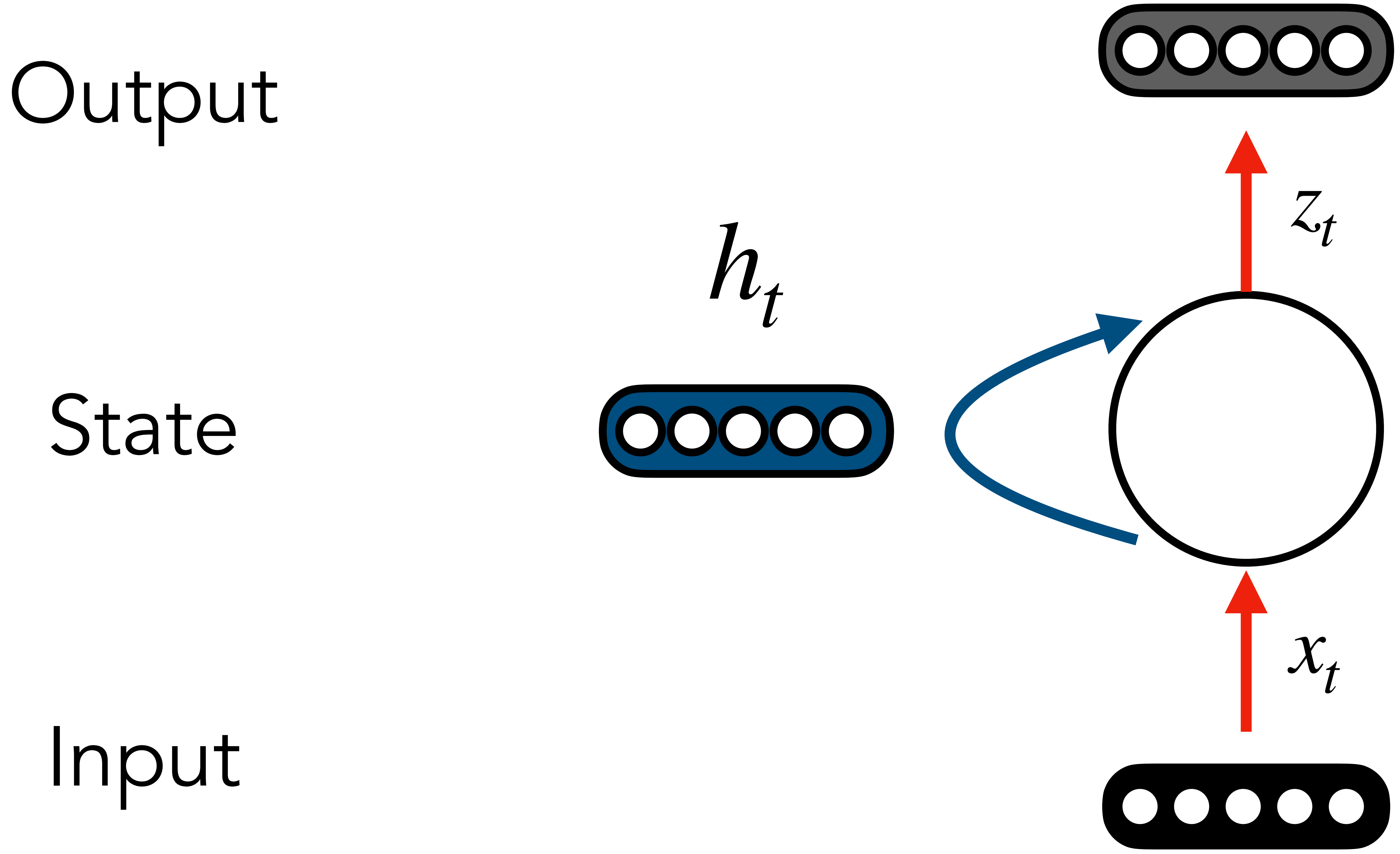
P(mouse | fanatically chased after the elusive) ✗



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

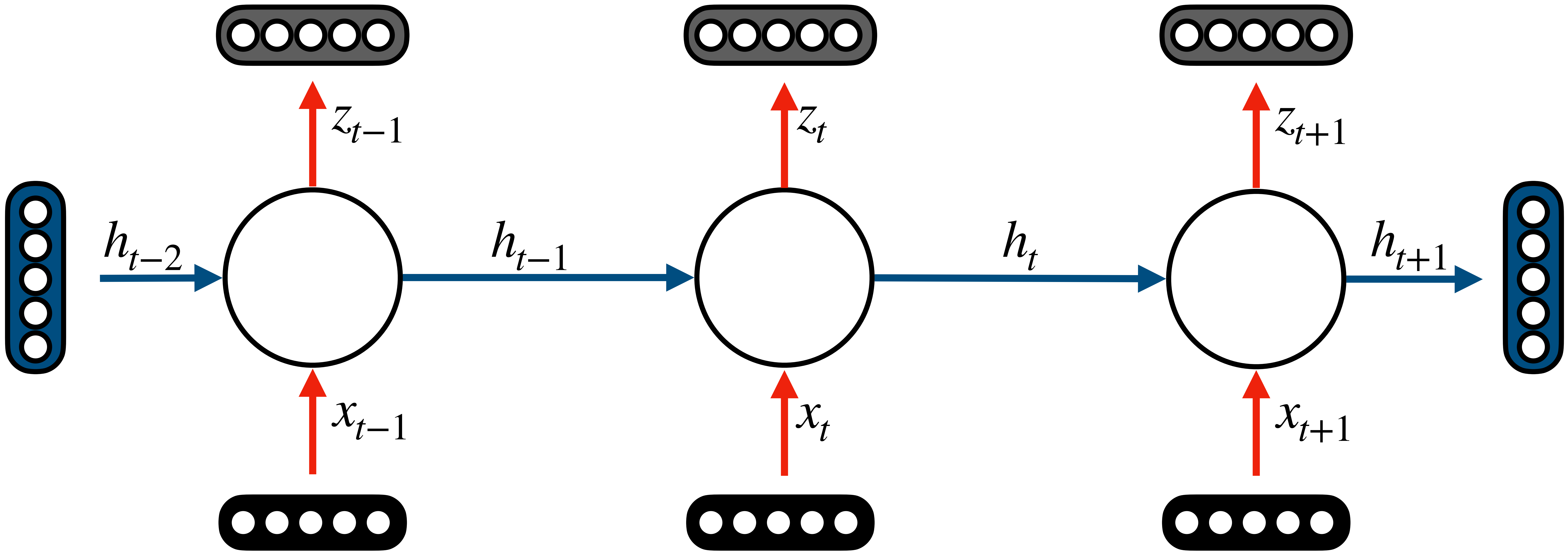
# Recurrent Neural Networks

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



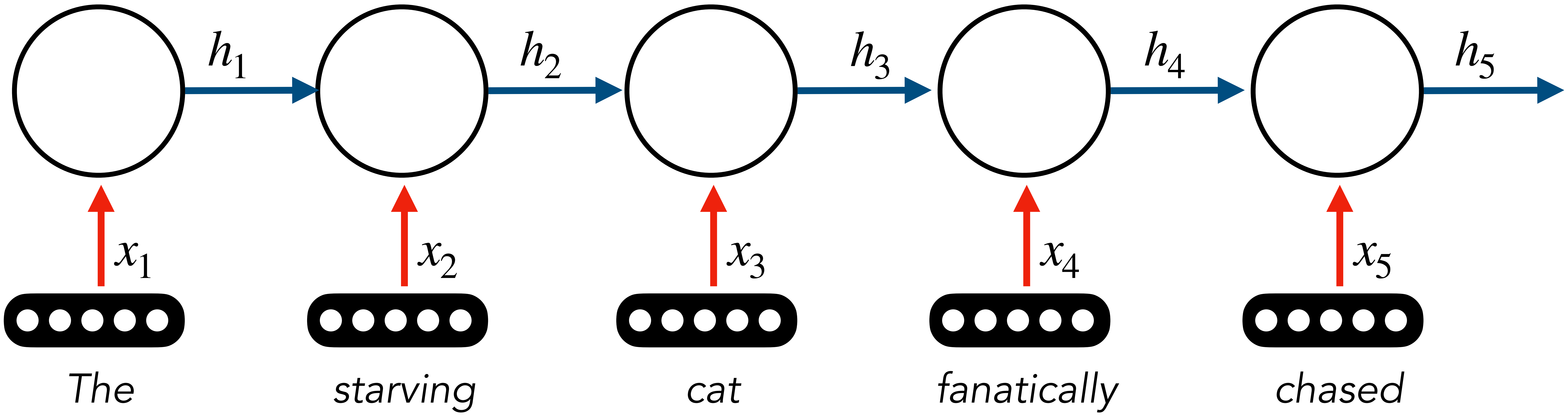
# Unrolling the RNN

**Unrolling the RNN across all time steps gives full computation graph**

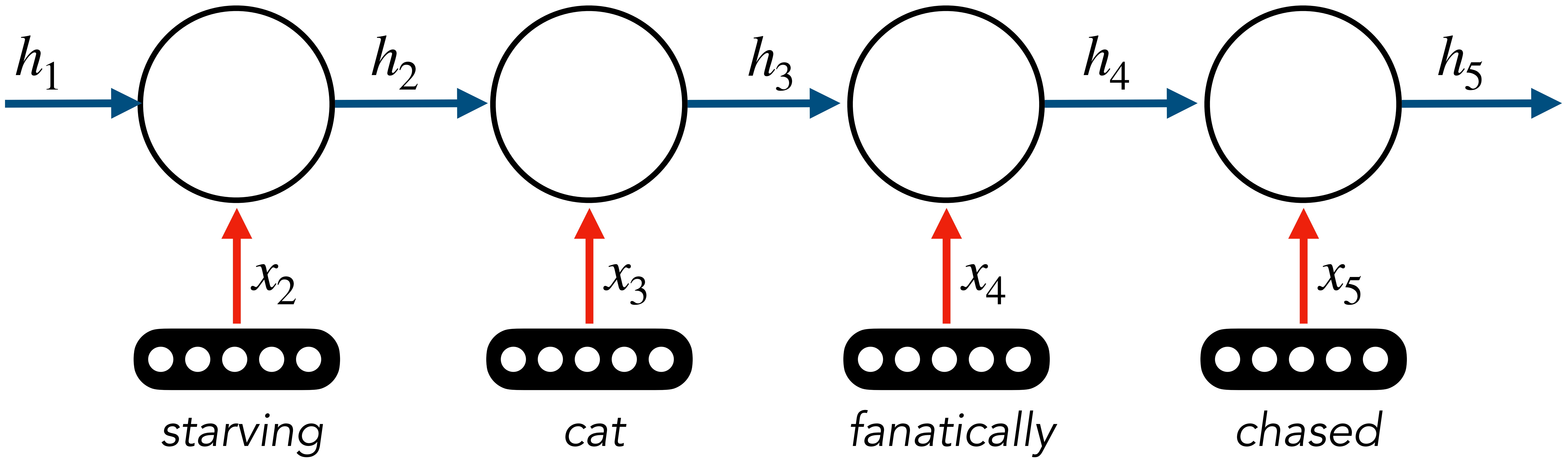



**Allows for learning from entire sequence history, regardless of length**

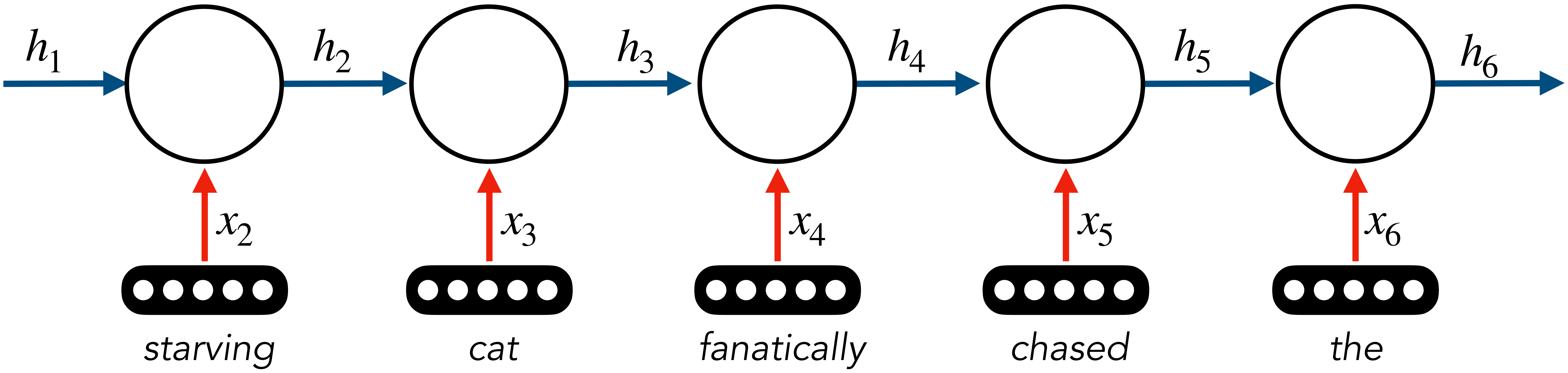
# Unrolling the RNN



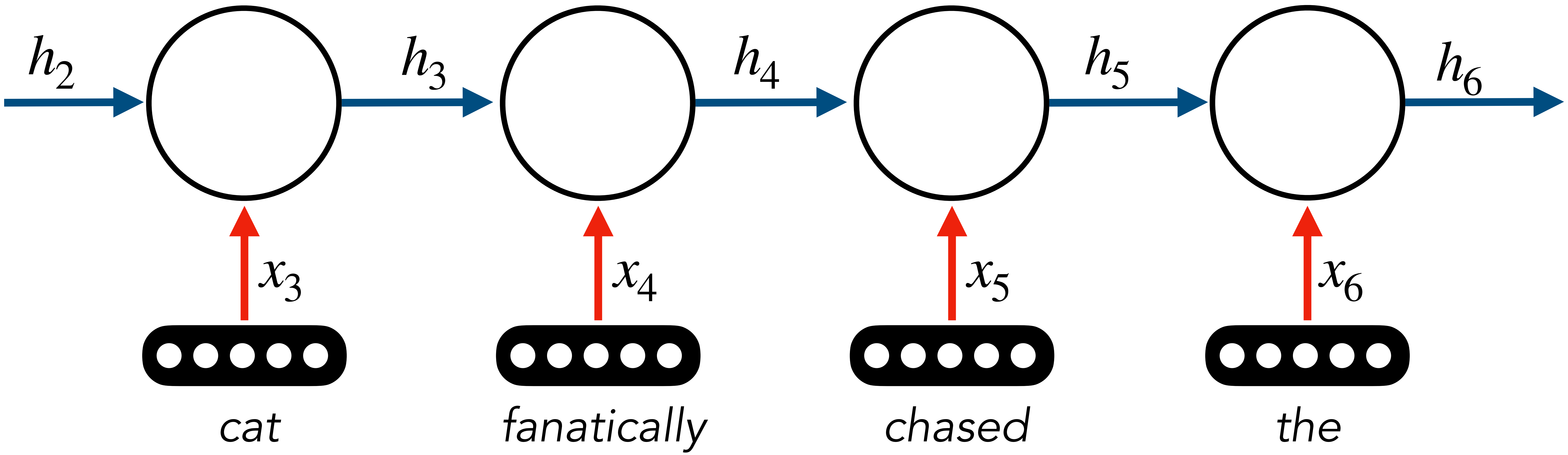
# Unrolling the RNN



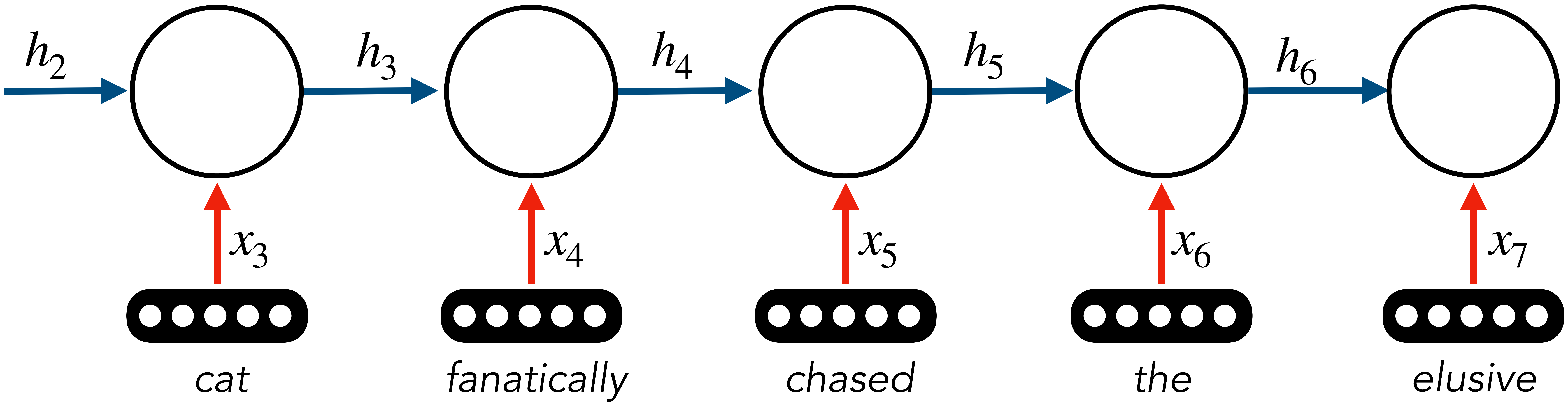
# Unrolling the RNN



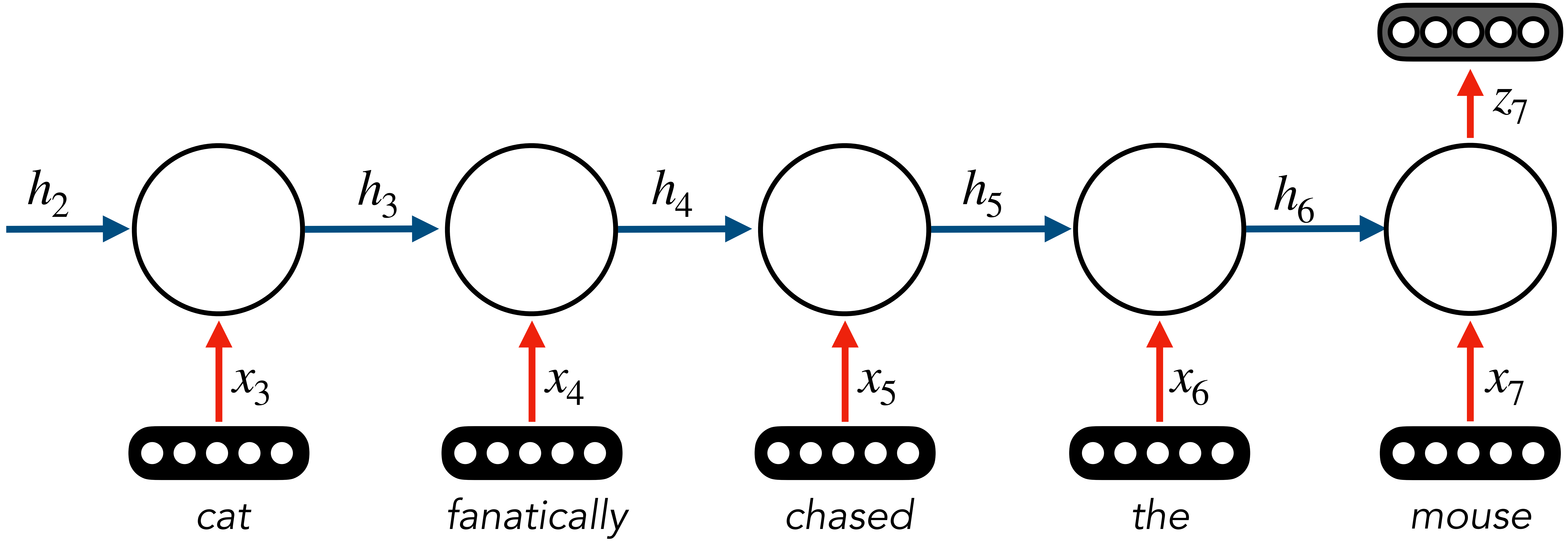
# Unrolling the RNN



# Unrolling the RNN



# Unrolling the RNN



# Classification

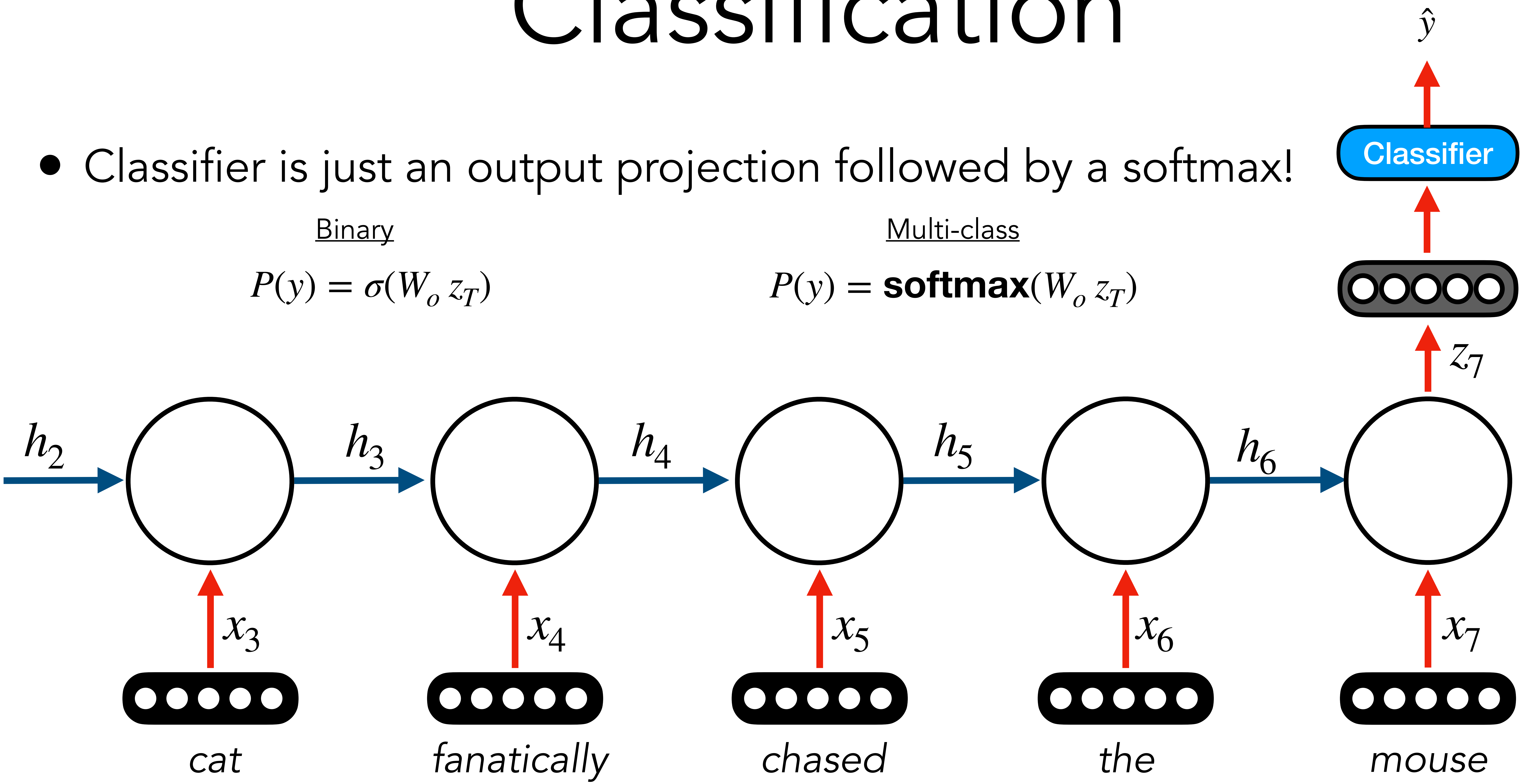
- Classifier is just an output projection followed by a softmax!

<u>Binary</u>

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

<u>Multi-class</u>

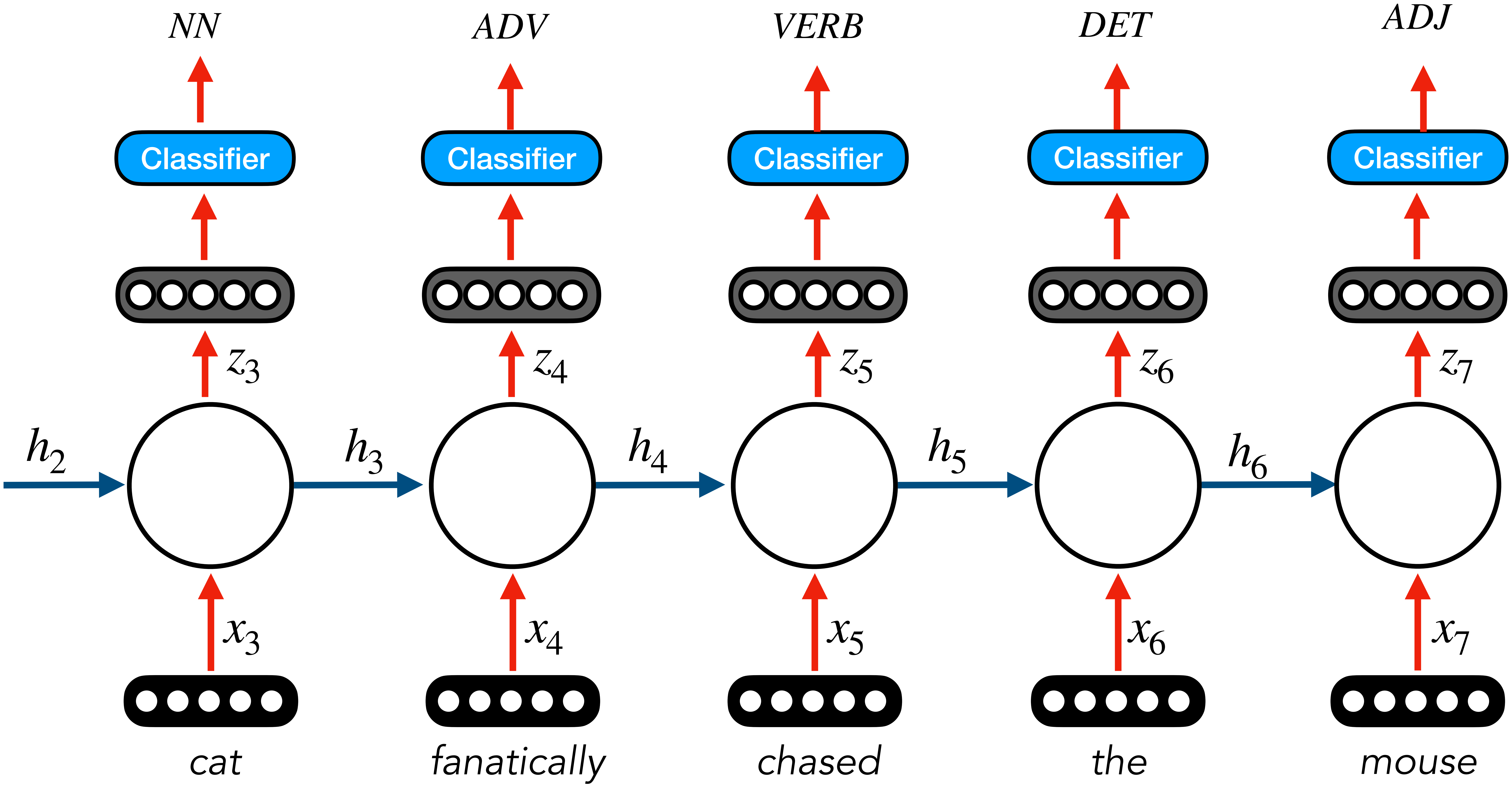
$$P(y) = \textbf{softmax}(W_o\, z_T)$$



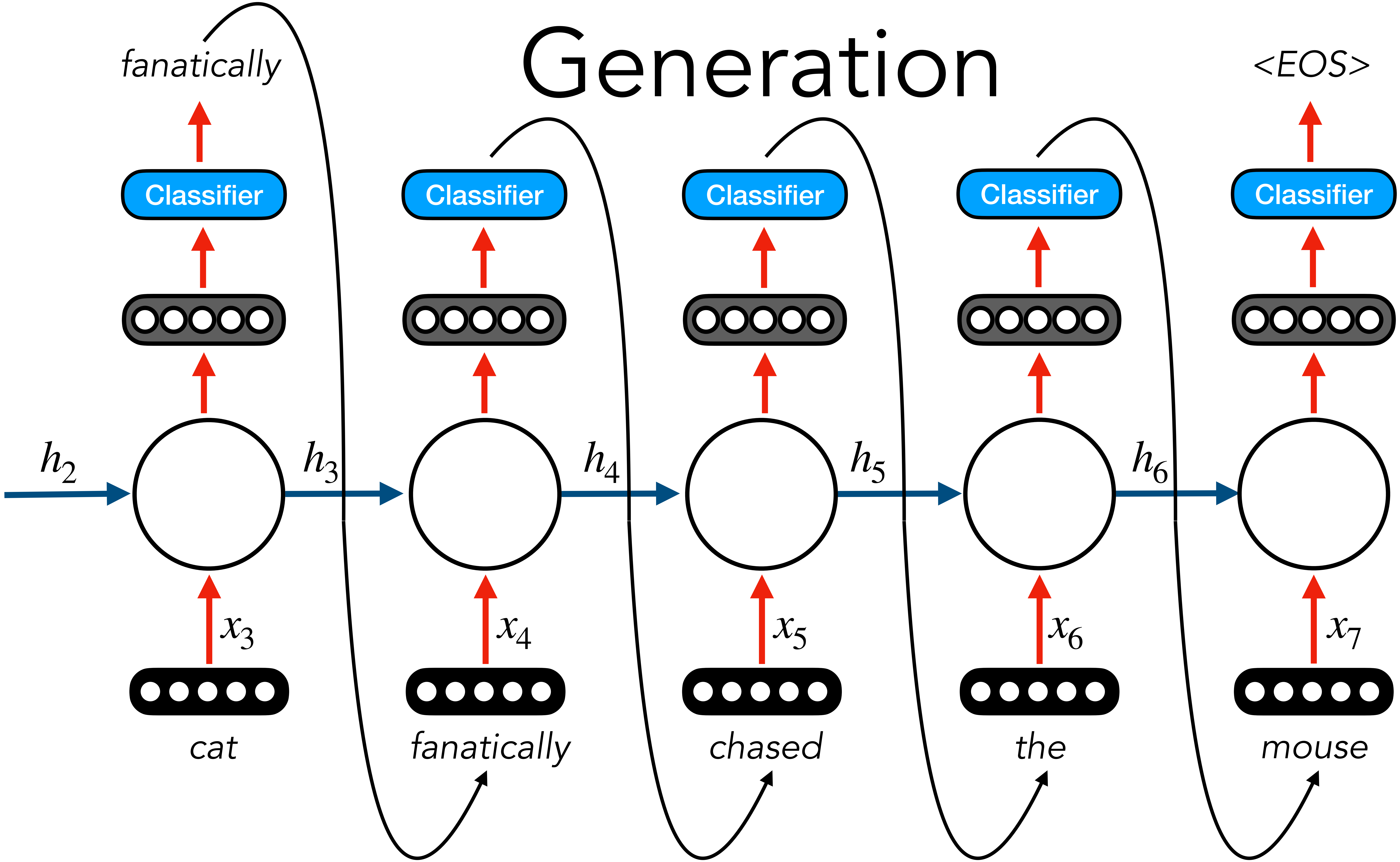
# Question

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

# Sequence Labeling

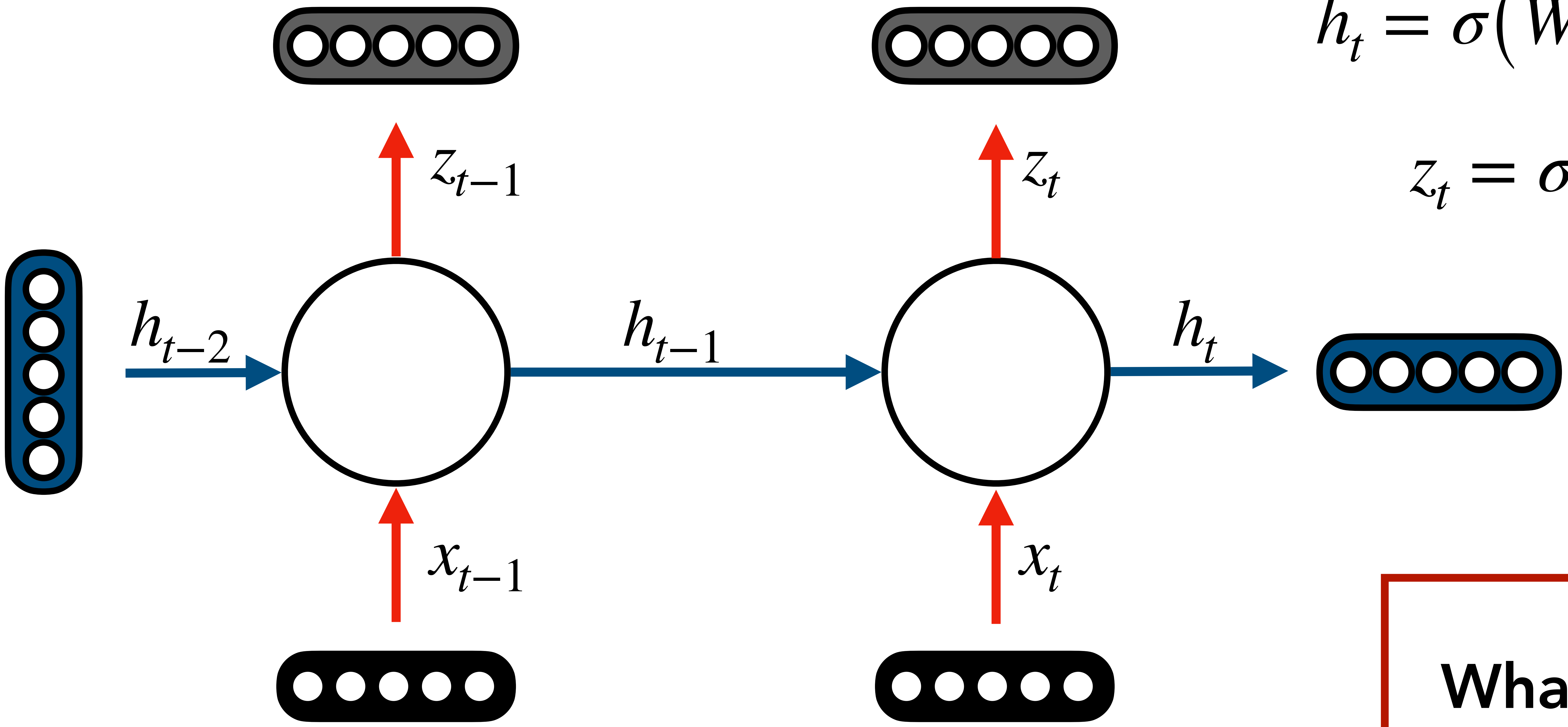


# Generation



fanatically
<EOS>
Classifier
Classifier
Classifier
Classifier
Classifier
$h_2$
$h_3$
$h_4$
$h_5$
$h_6$
$x_3$
$x_4$
$x_5$
$x_6$
$x_7$
cat
fanatically
chased
the
mouse

# Classical RNN: Elman Network



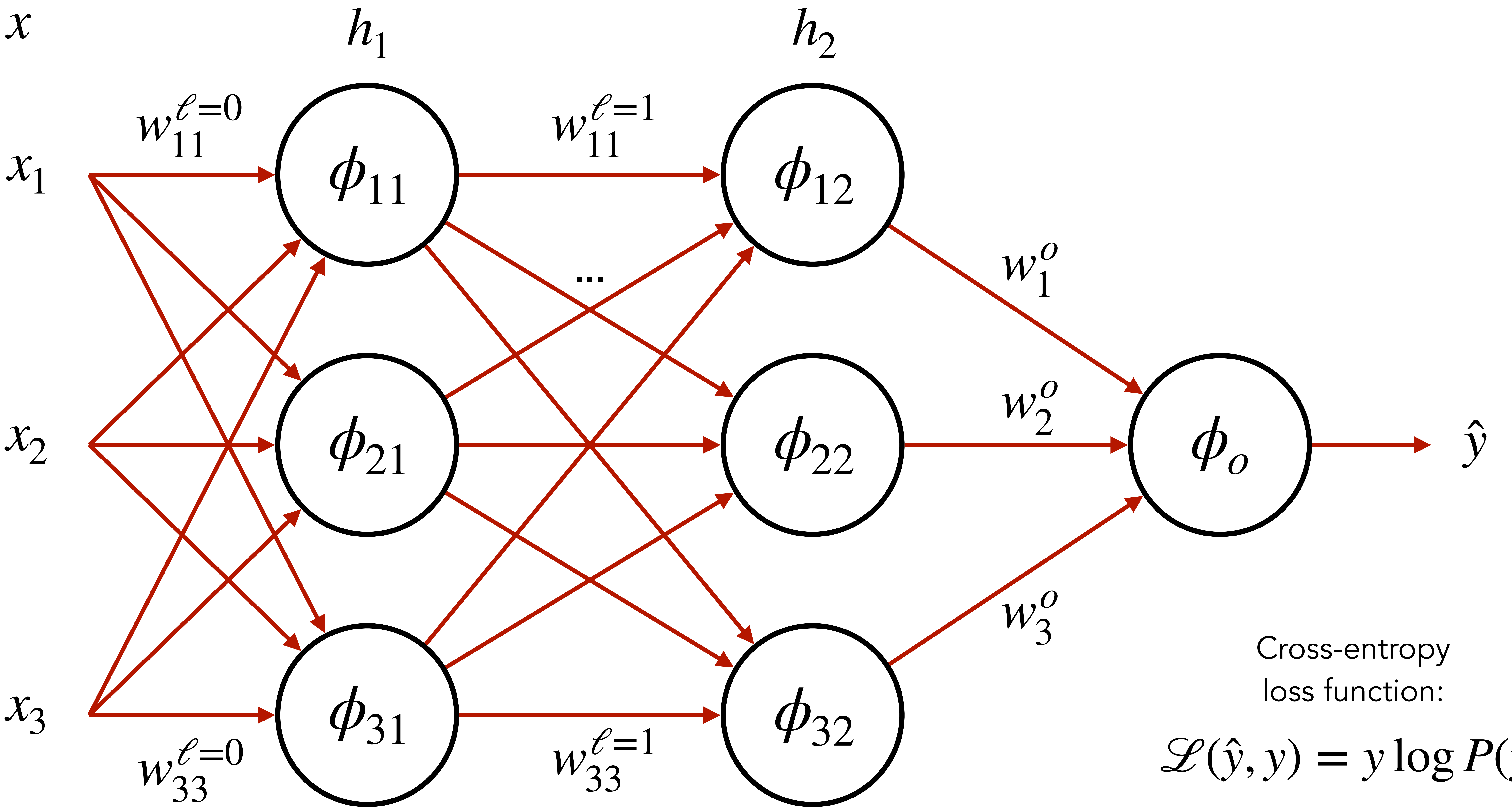

$$h_t = \sigma\left(W_{hx}x_t + W_{hh}h_{t-1} + b_h\right)$$

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

**What should $h_0$ be?**

(Elman, 1990)

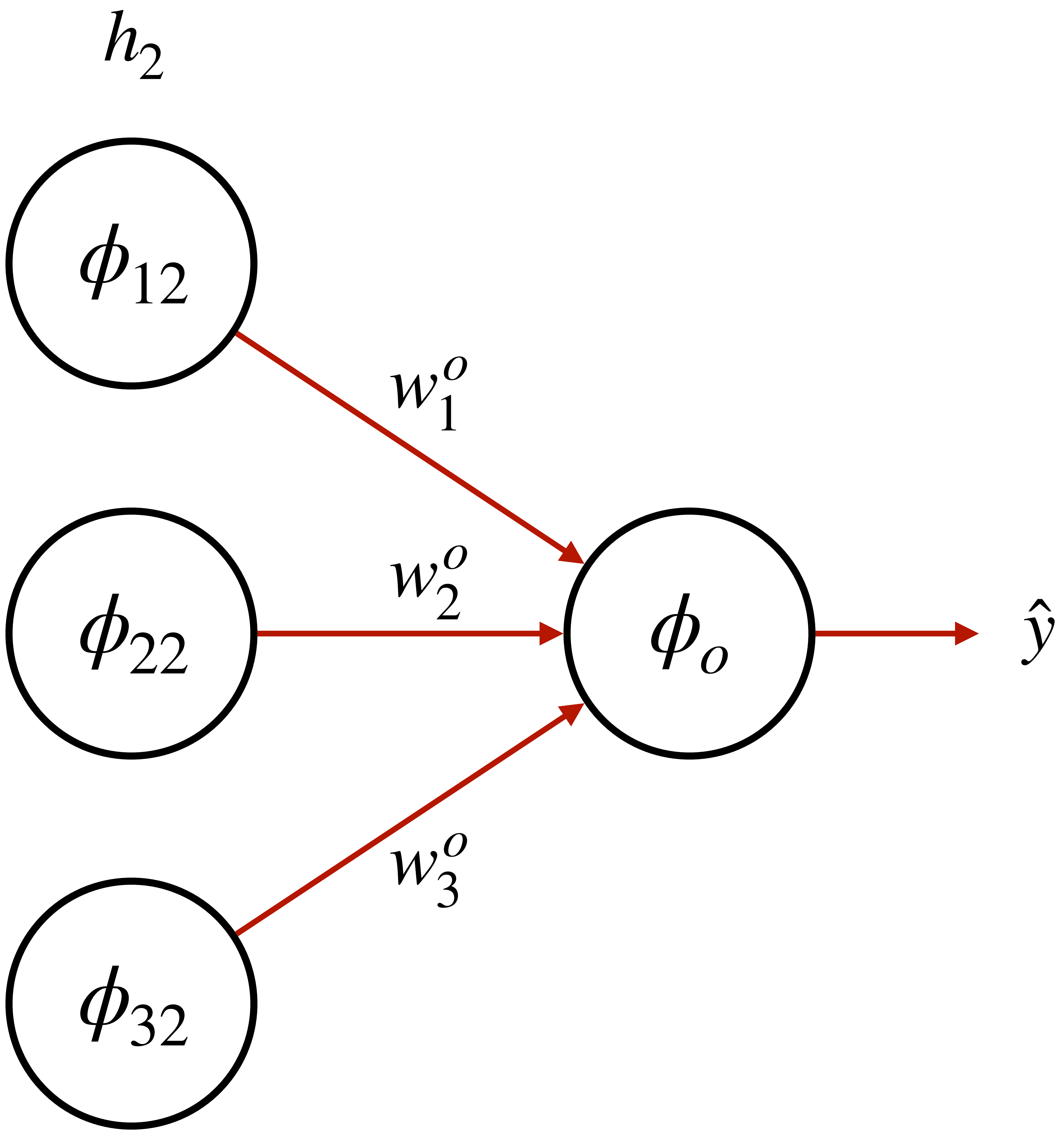
# Backpropagation Review: FFNs




Cross-entropy loss function:

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

# Backpropagation Review: FFNs




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

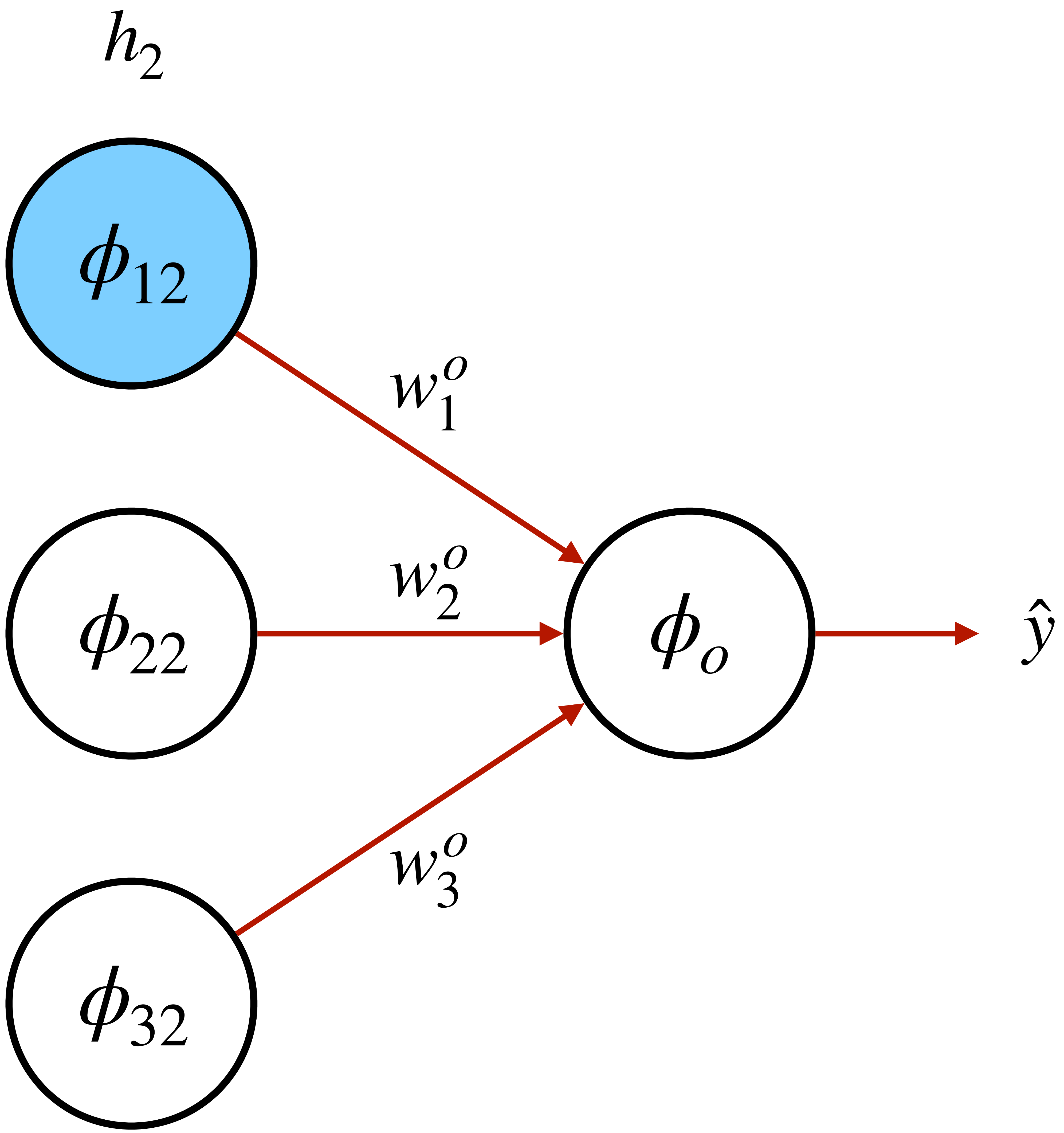
$$\hat{y} = \phi_o(u)$$

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

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

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

# Backpropagation Review: FFNs




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

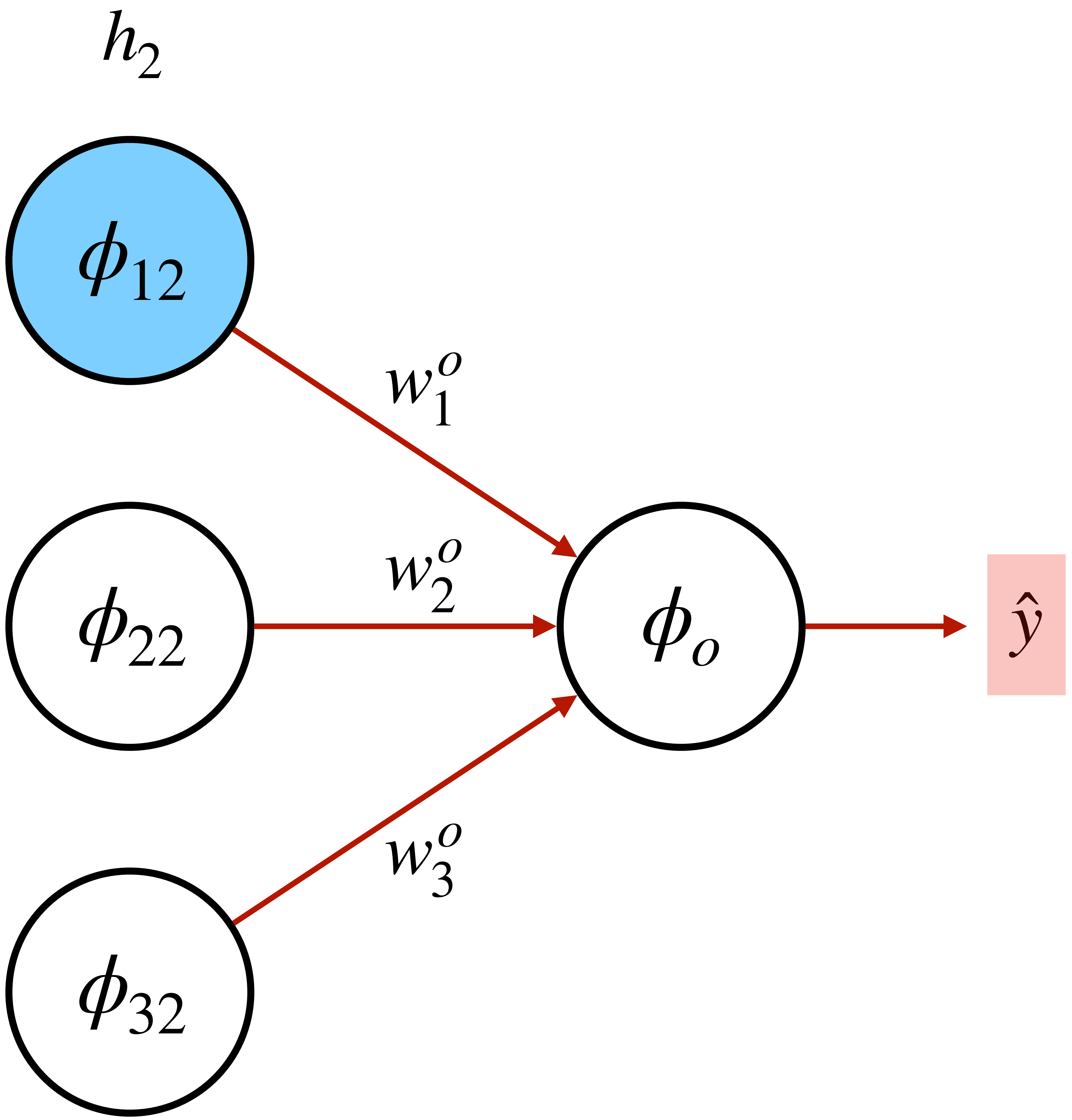
$$\hat{y} = \phi_o(u)$$

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

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

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

# Backpropagation Review: FFNs




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

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

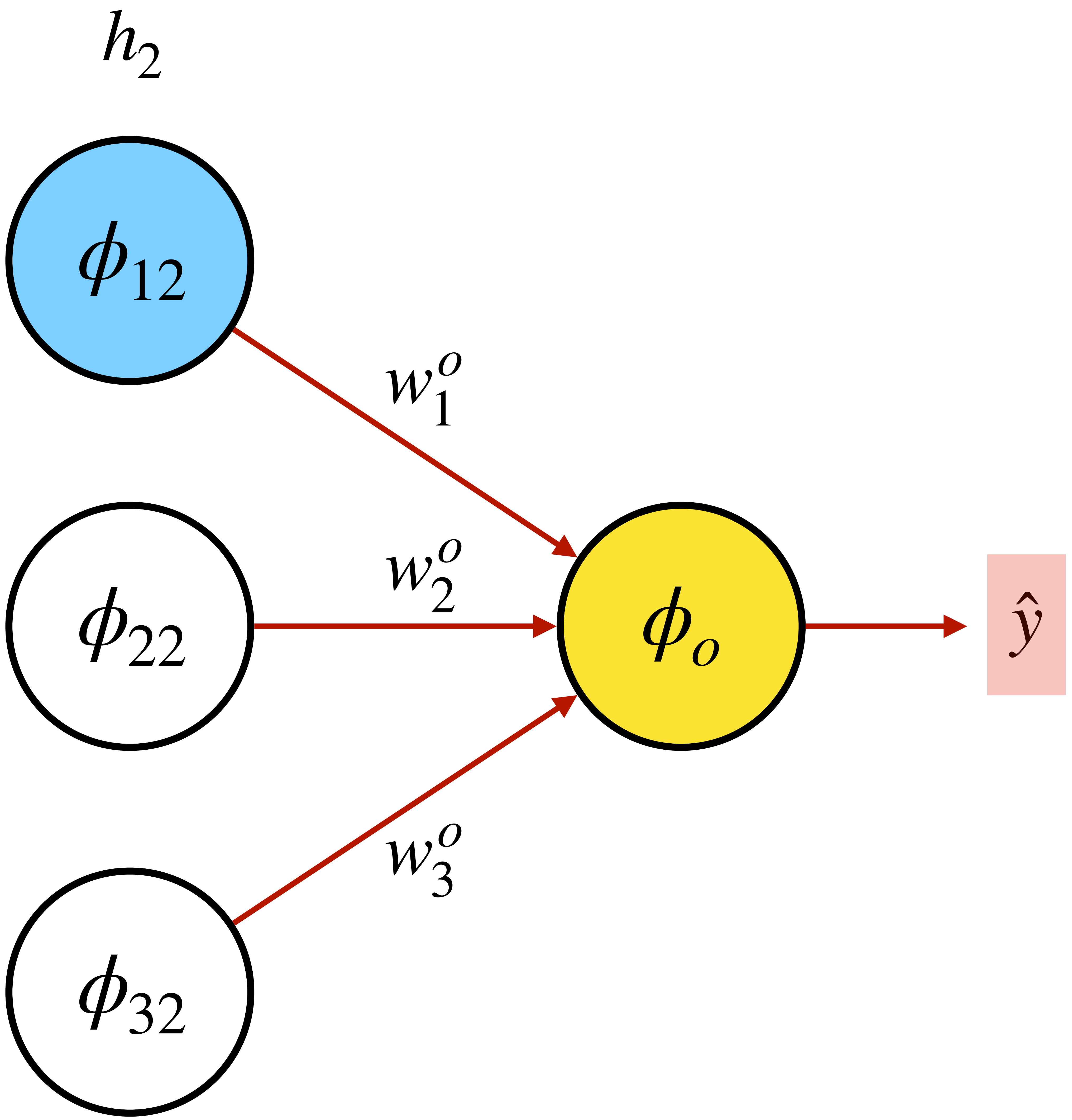
$$u = w_1^o \times \phi_{12}(\,.\,) + w_2^o \times \phi_{22}(\,.\,) + w_3^o \times \phi_{32}(\,.\,)$$

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

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

Depends on label $y$

# Backpropagation Review: FFNs




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

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

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

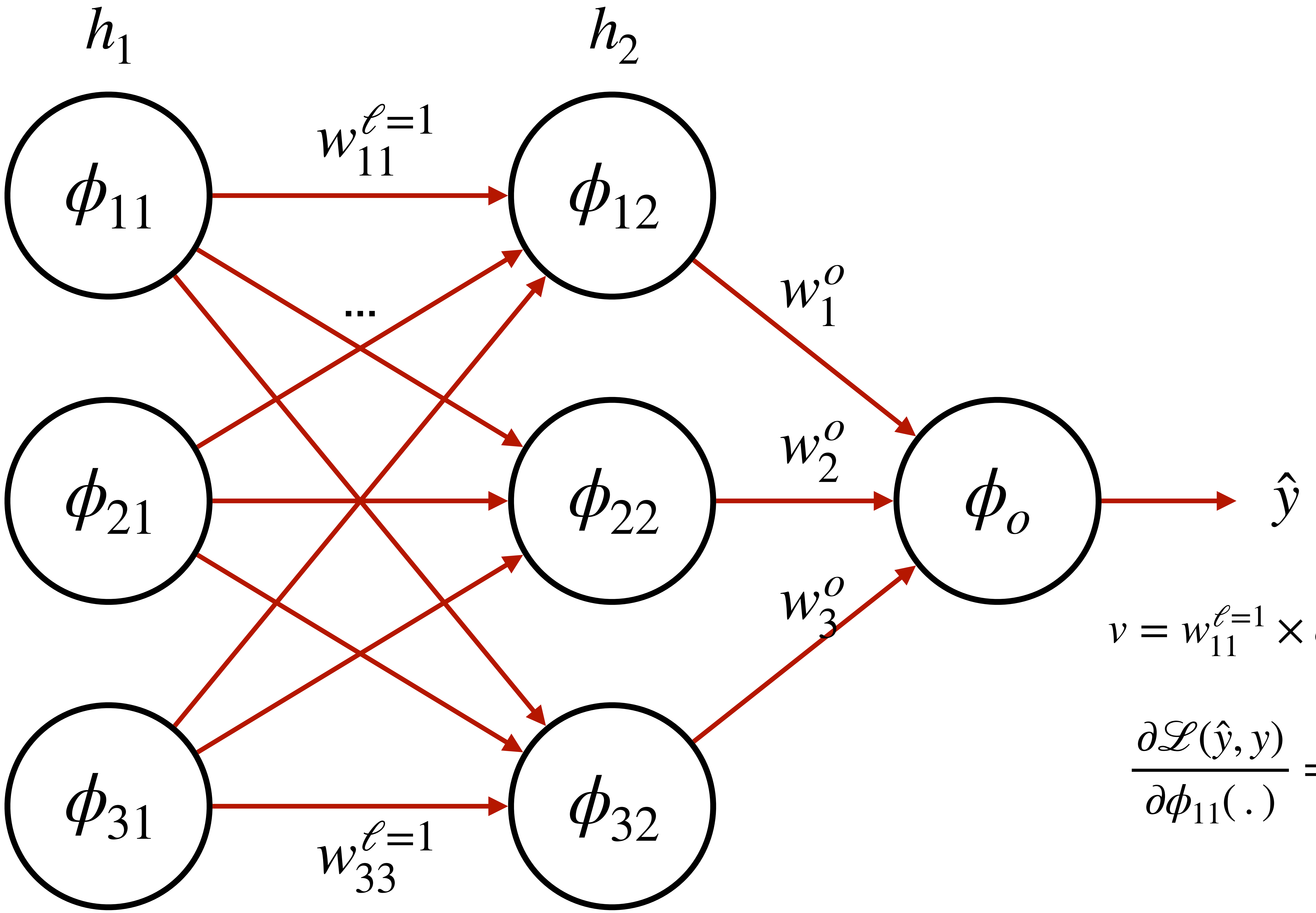
$$\frac{\partial \mathscr{L}(\hat{y}, y)}{\partial \phi_{12}(\,.\,)} = \frac{\partial \mathscr{L}(\hat{y}, y)}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial u} \frac{\partial u}{\partial \phi_{12}(\,.\,)}$$

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

Depends on label $y$

Depends on $\phi_o$

# Backpropagation Review: FFNs




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

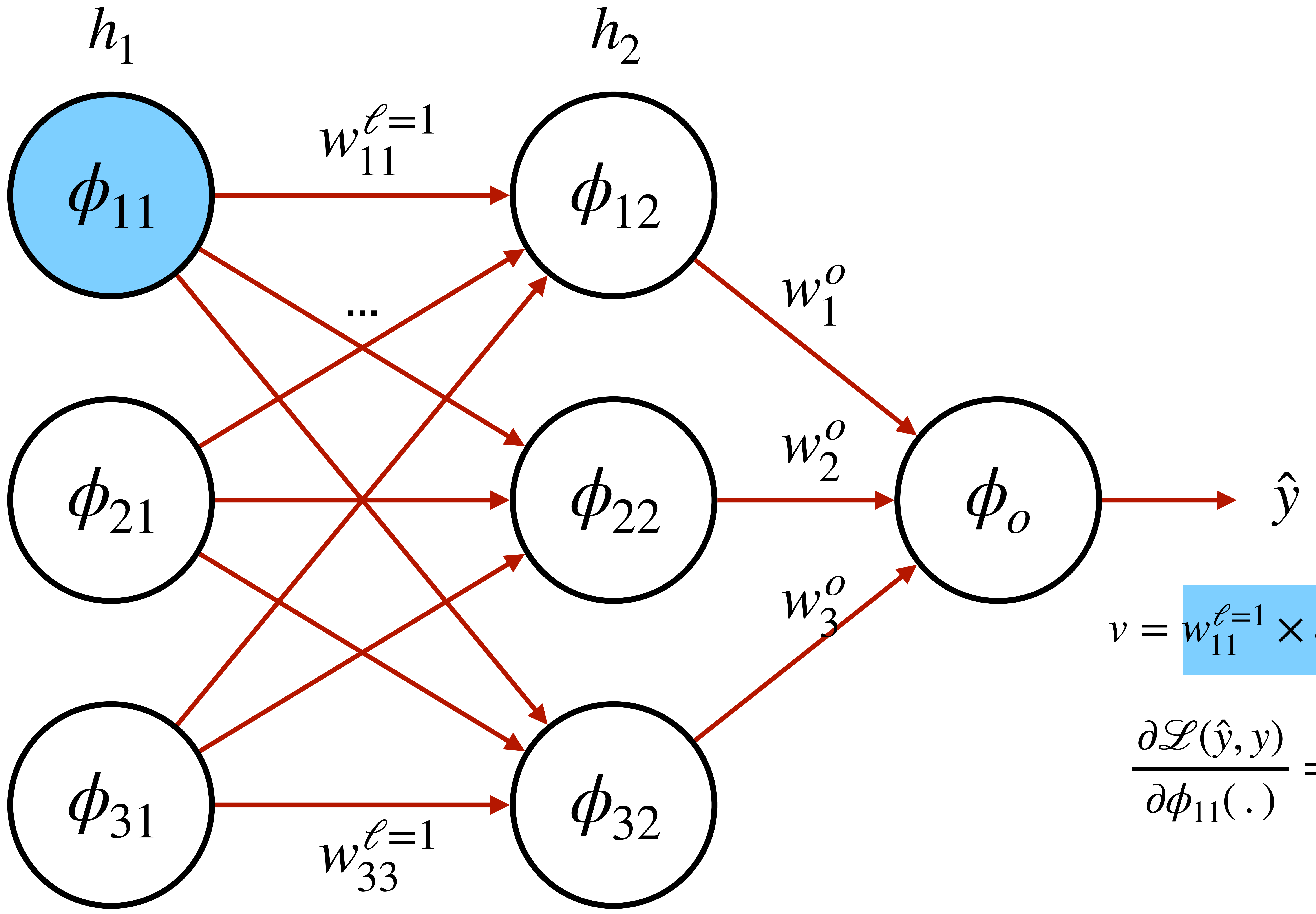
$$= \frac{\partial \mathcal{L}(\hat{y}, y)}{\partial \hat{y}} \frac{\partial \phi_o(u)}{\partial u} w_1^0$$

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

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

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

# Backpropagation Review: FFNs




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

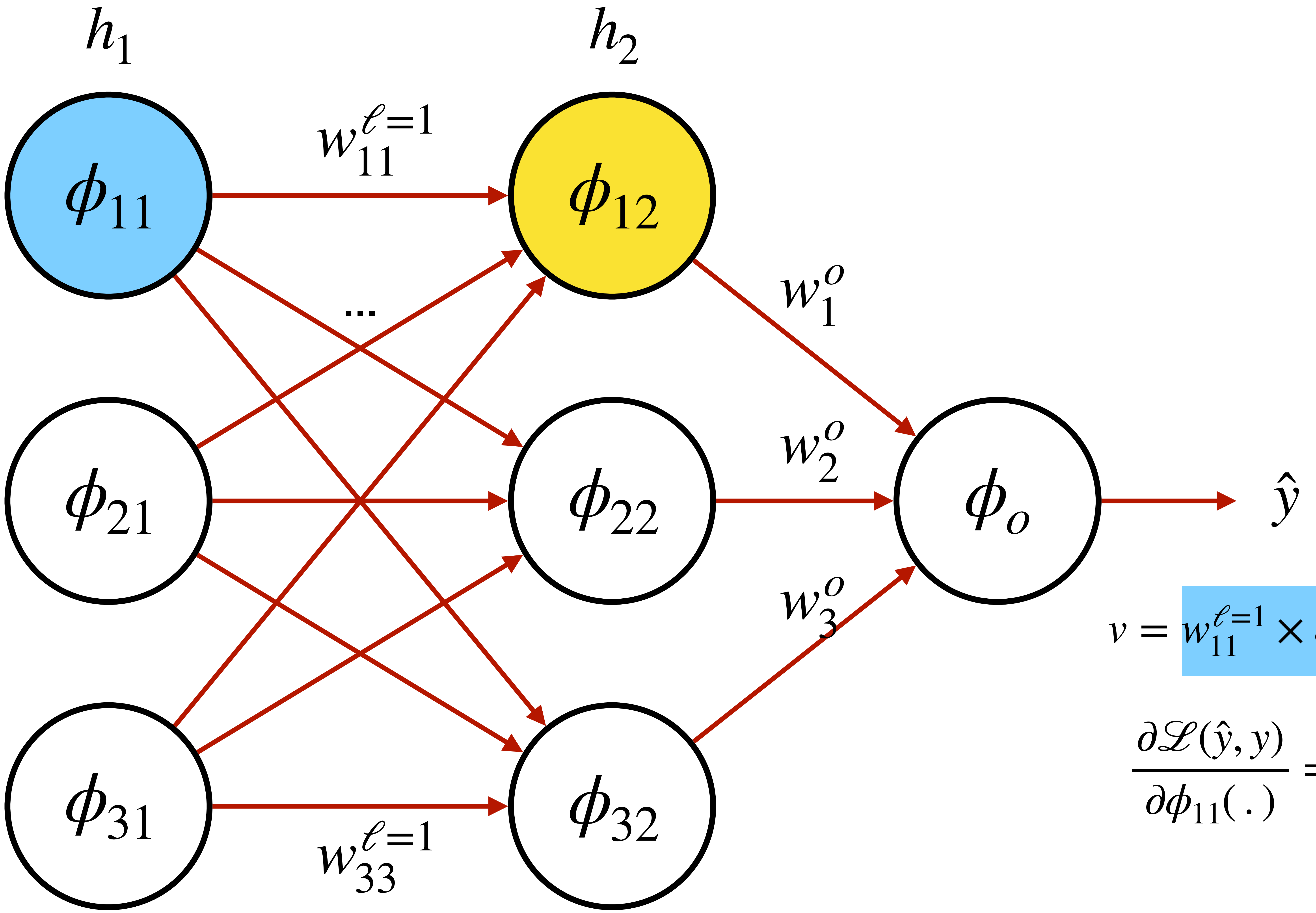
$$= \frac{\partial \mathcal{L}(\hat{y}, y)}{\partial \hat{y}} \frac{\partial \phi_o(u)}{\partial u} w_1^0$$

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

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

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

# 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 $\phi_{12}$

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

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

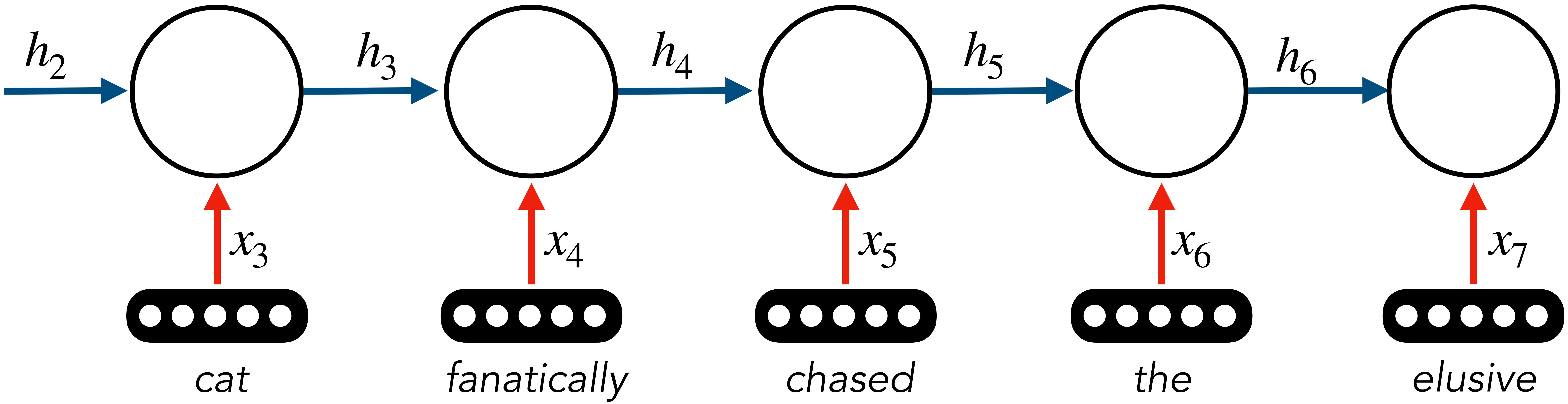
$$= \frac{\partial \mathcal{L}(\hat{y}, y)}{\partial \hat{y}} \frac{\partial \phi_o(u)}{\partial u} w_1^0 \frac{\partial \phi_{12}(v)}{\partial v} w_{11}^{\ell=1}$$

# Question

**How would we extend backpropagation to a recurrent neural network?**

# Recall

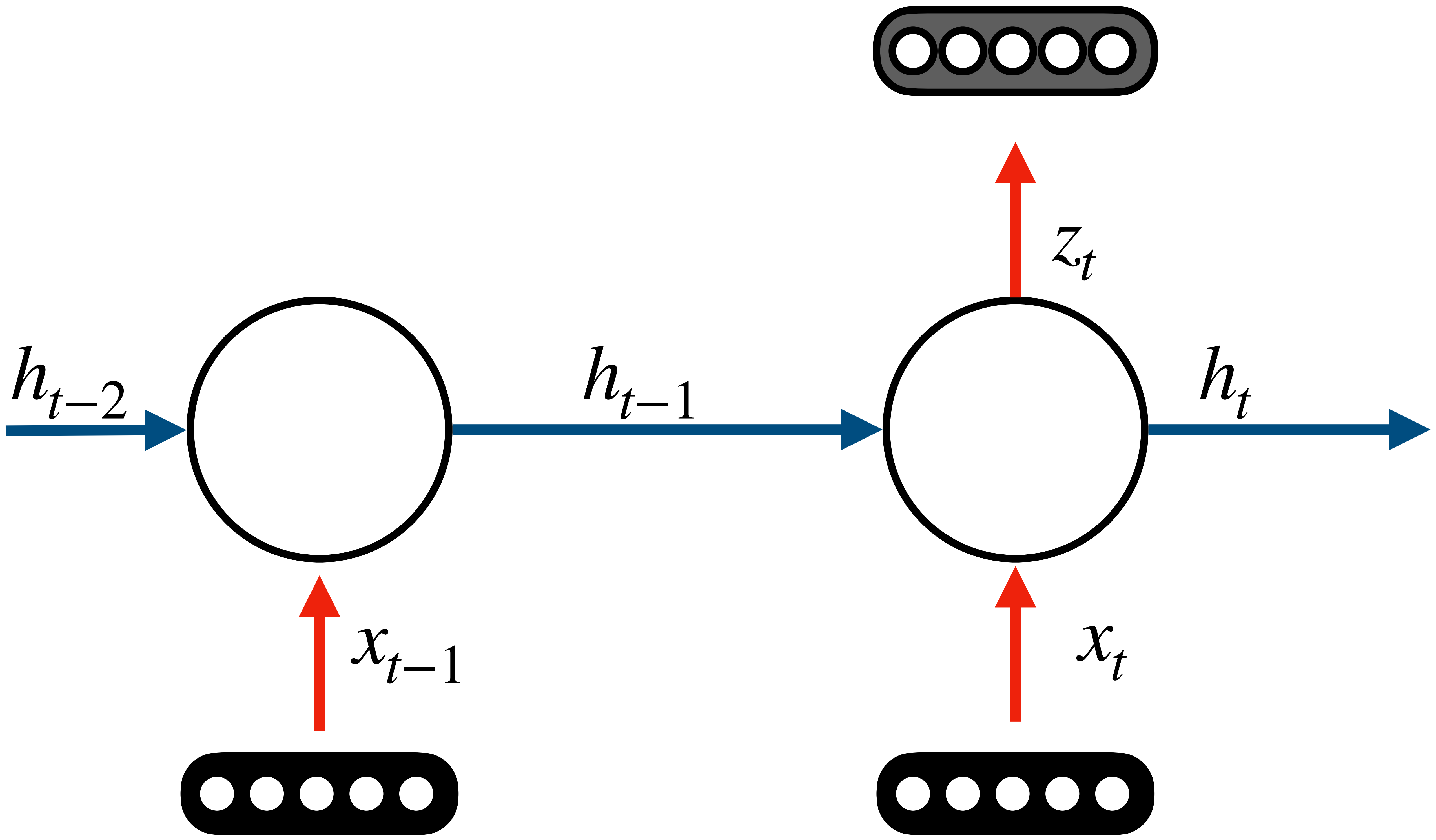
- RNN can be unrolled to a feedforward neural network
- Depth of feedforward neural network depends on length of the sequence



# Backpropagation through Time

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

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



# Backpropagation through Time

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

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

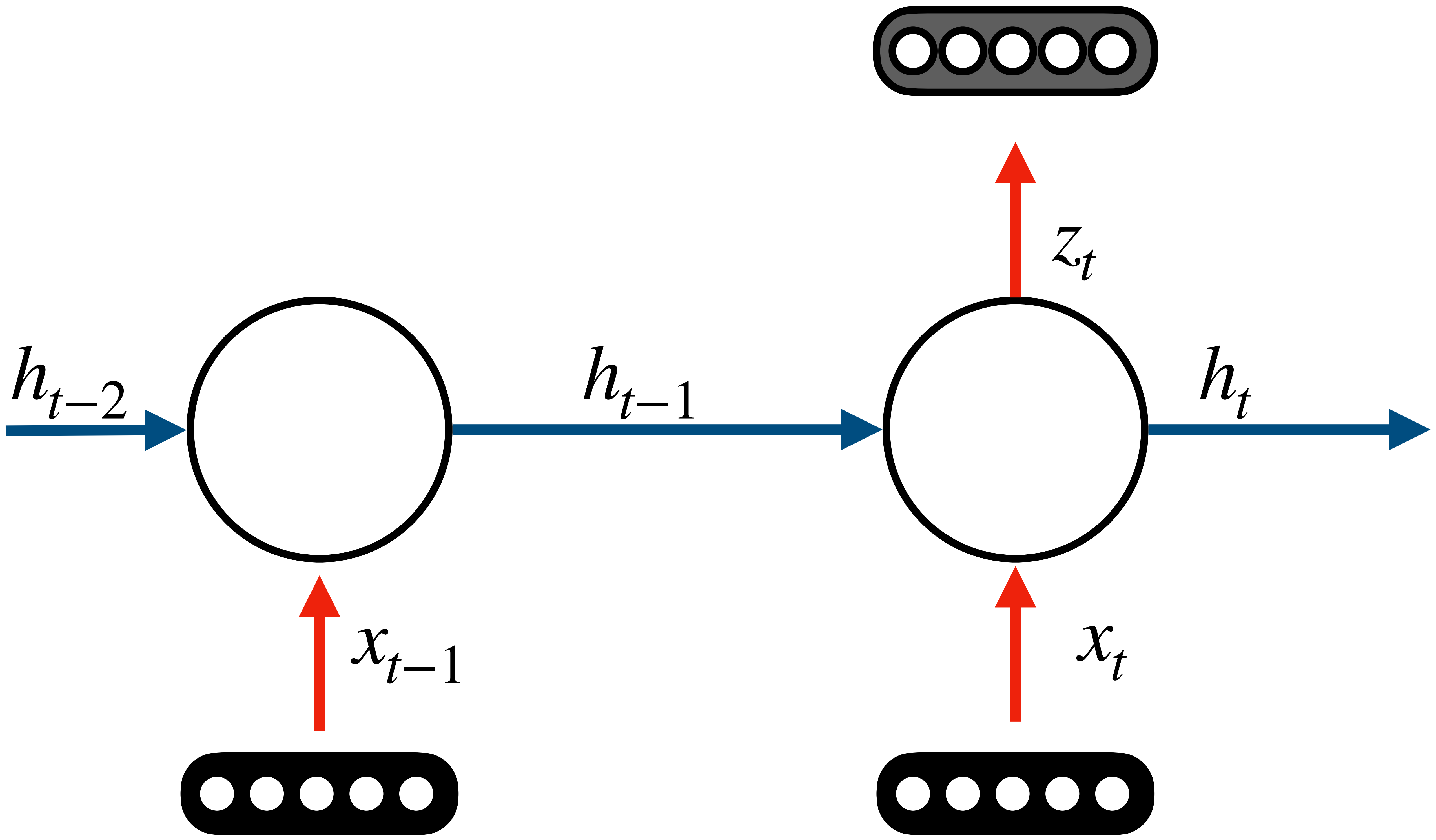
---

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

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

---

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



# Backpropagation through Time

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

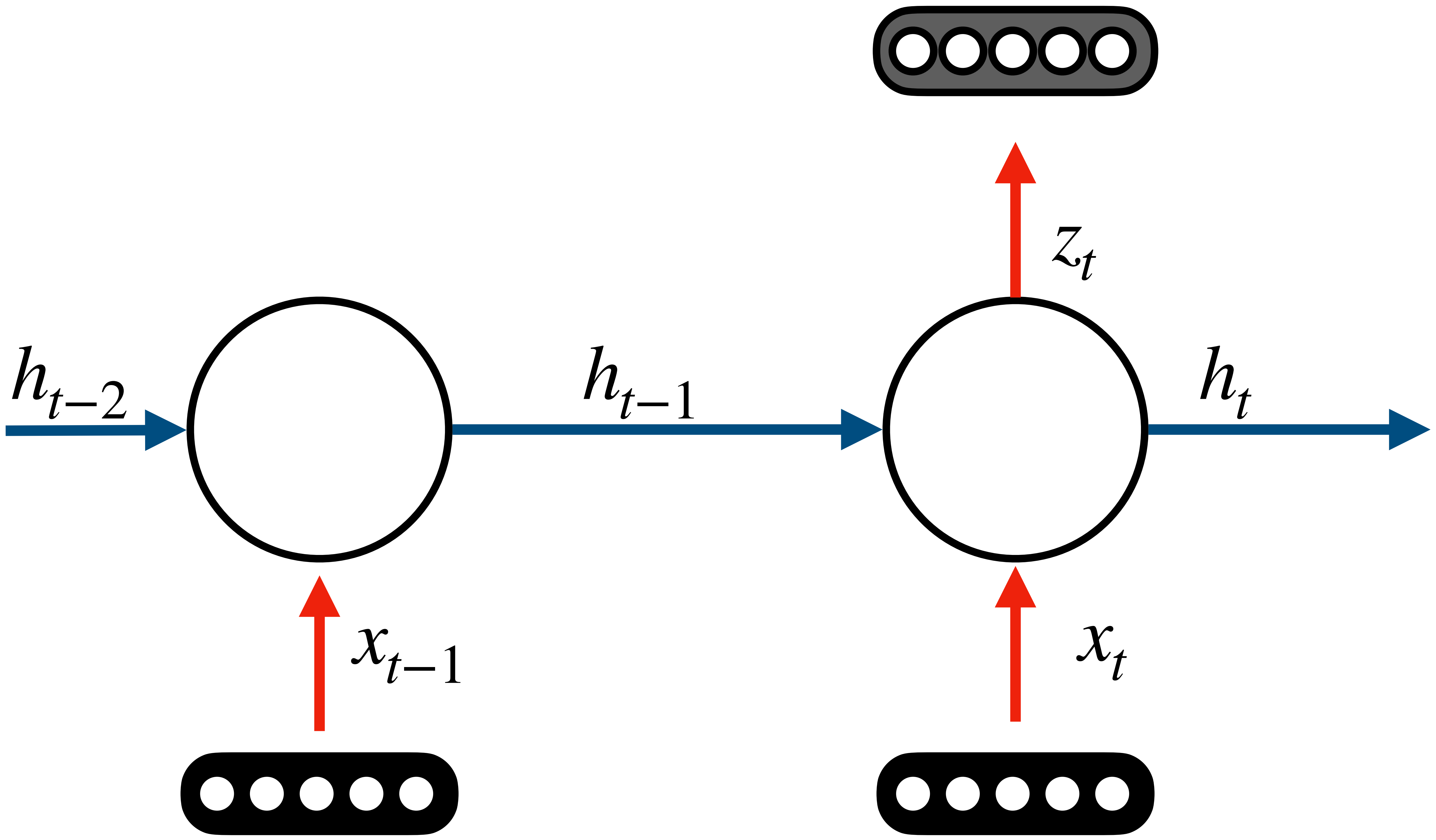
$$h_t = \sigma\left(W_{hx}x_t + W_{hh}h_{t-1} + b_h\right)$$

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

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

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

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



# Backpropagation through Time

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

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

---

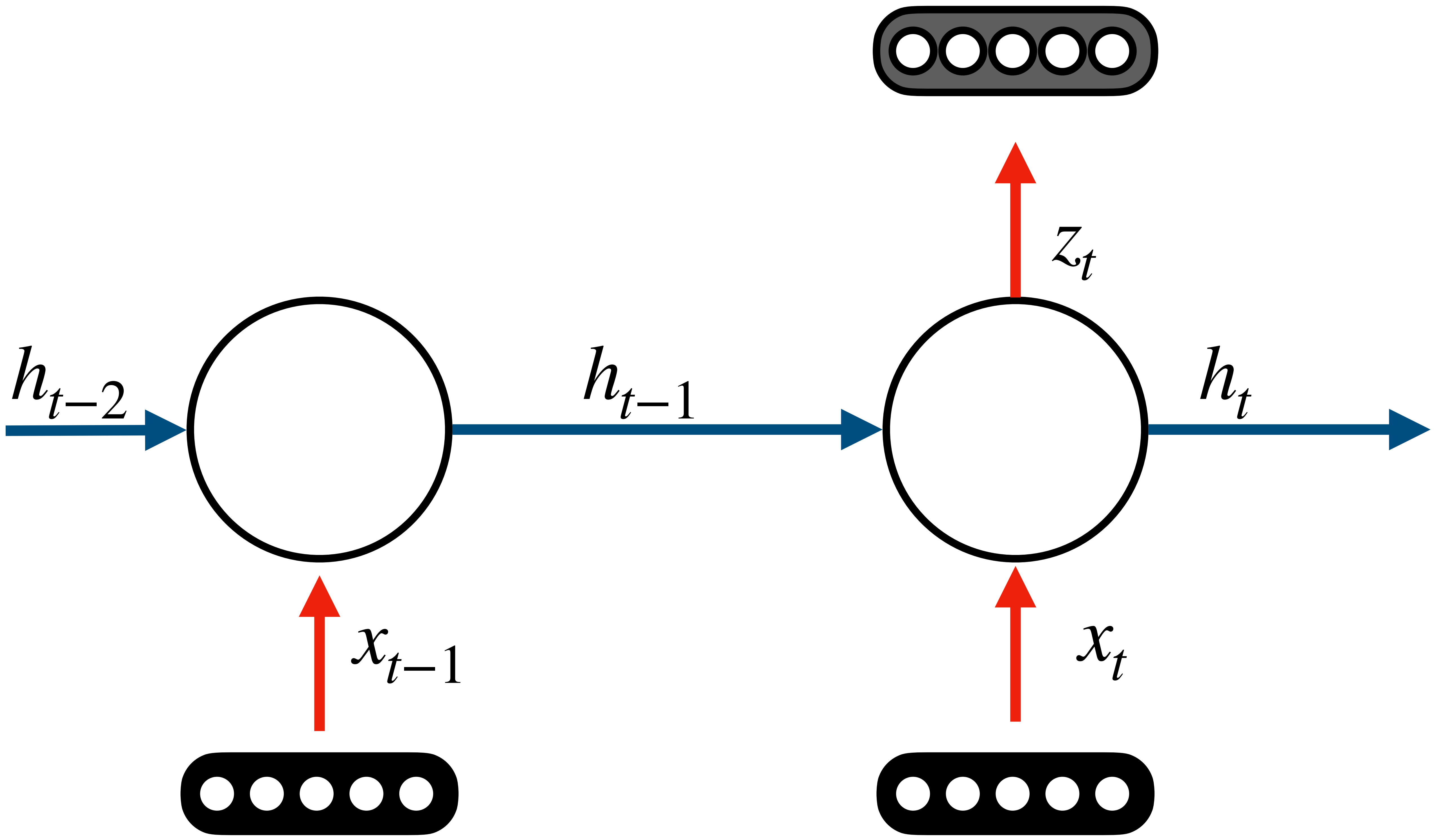
$$v = W_{zh} h_t + b_z \qquad z_t = \sigma(v)$$

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

---

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

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




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

# Backpropagation through Time

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

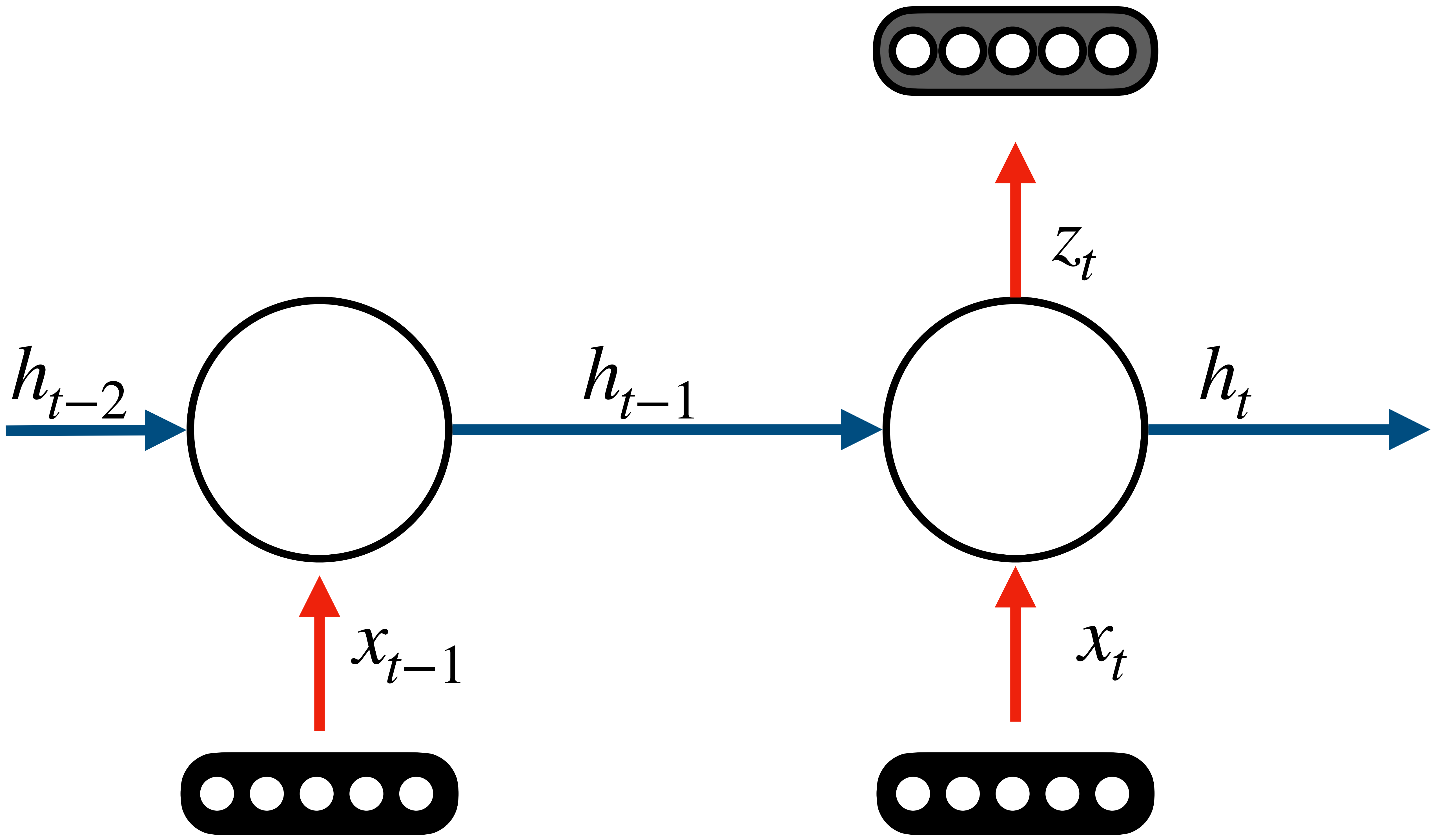
$$h_t = \sigma\left(W_{hx}x_t + W_{hh}h_{t-1} + b_h\right)$$

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

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

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

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




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

# Backpropagation through Time

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

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

---

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

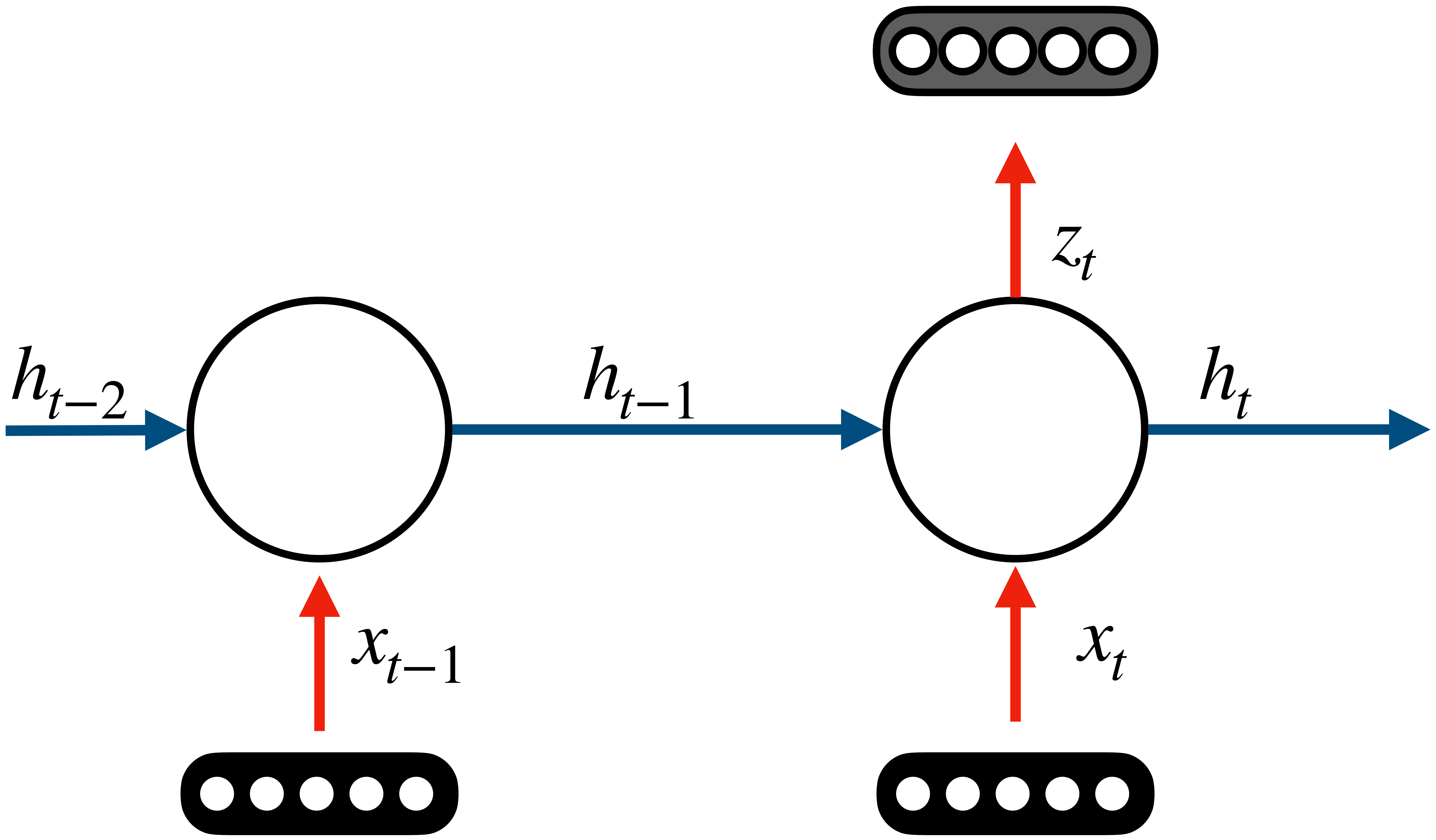
$$u_t = W_{hx}x_t + W_{hh}h_{t-1} + b_h \qquad h_t = \sigma(u_t)$$

---

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

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

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




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

# Backpropagation through Time

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

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

---

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

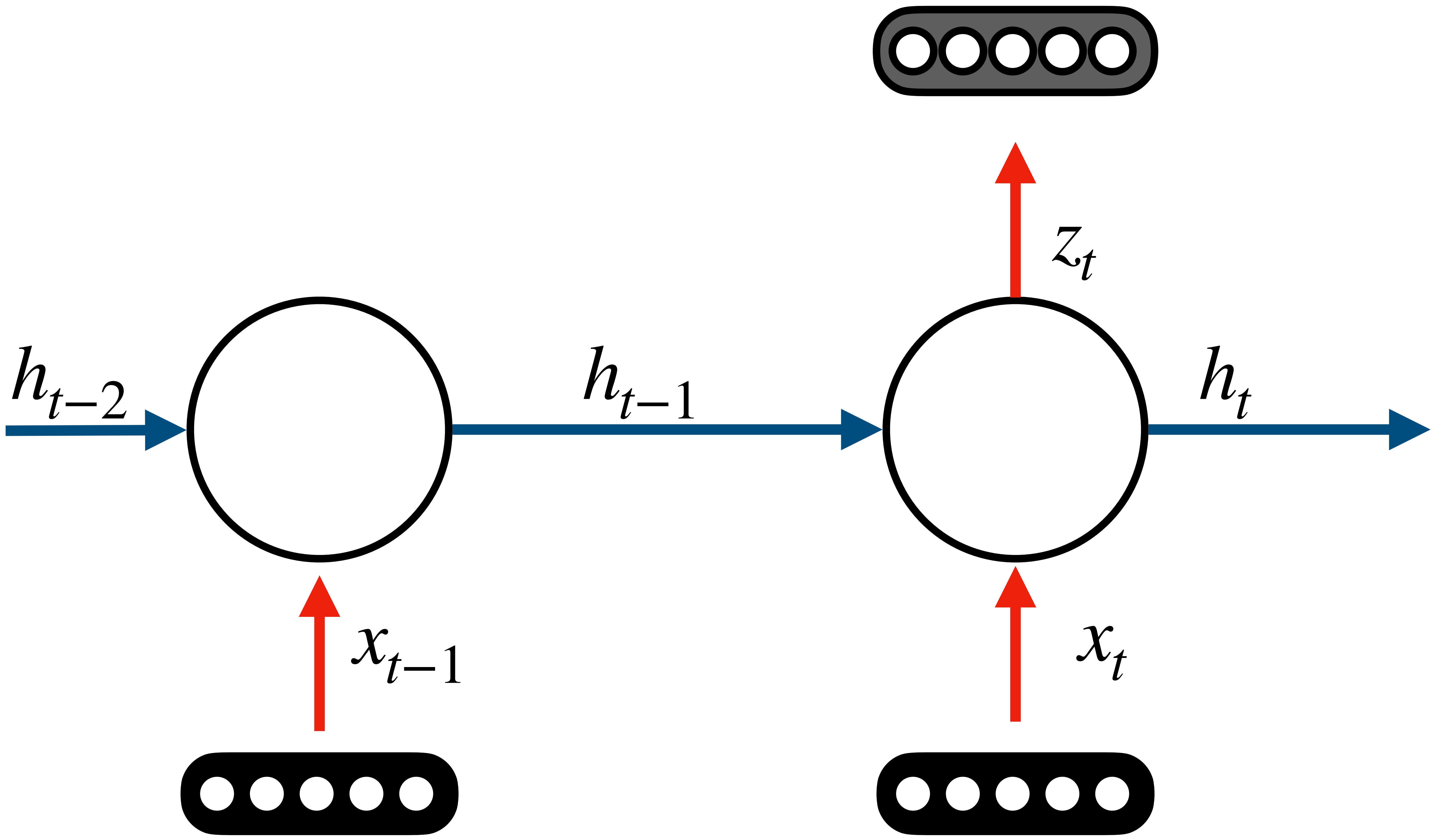
$$u_t = W_{hx}x_t + W_{hh}h_{t-1} + b_h \qquad h_t = \sigma(u_t)$$

---

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

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

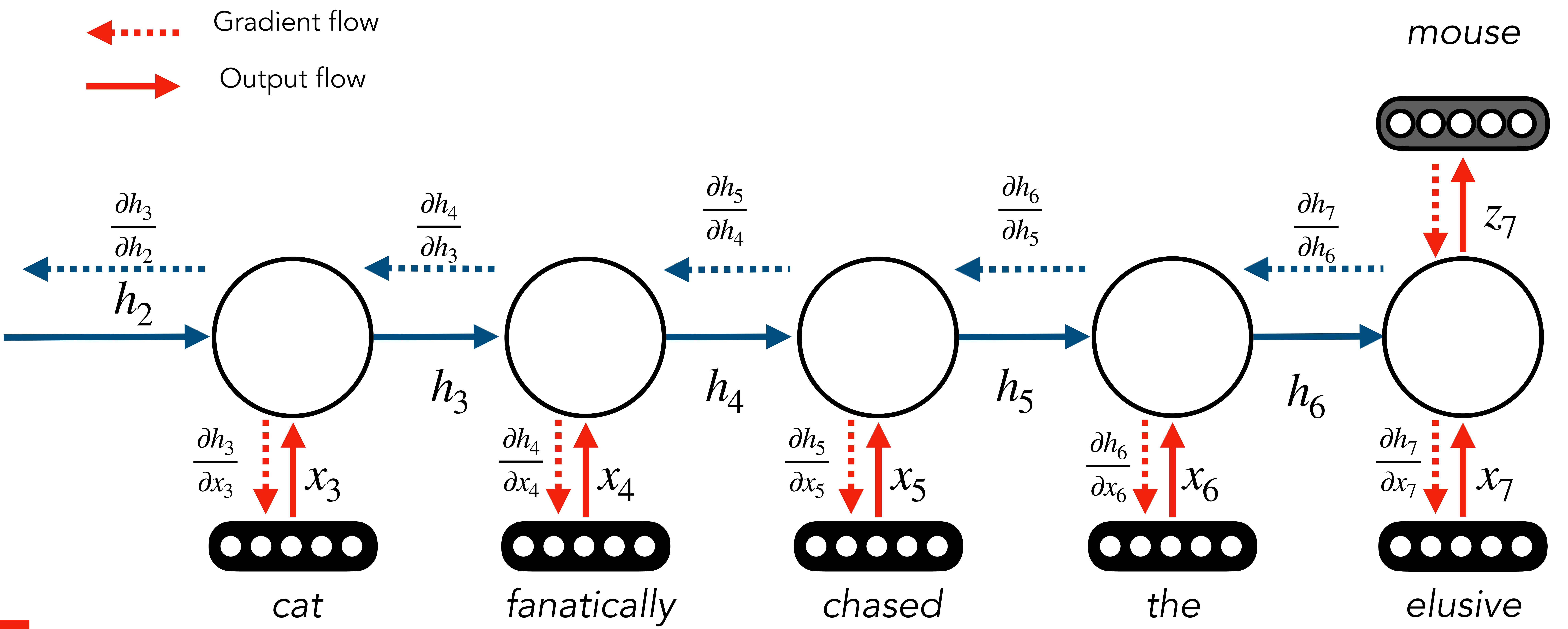
$$\frac{\partial h_{t-1}}{\partial h_{t-2}} = \frac{\partial \sigma(u_{t-1})}{\partial u_{t-1}}\frac{\partial u_{t-1}}{\partial h_{t-2}} = \frac{\partial \sigma(u_{t-1})}{\partial u_{t-1}}W_{hh}$$




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

Note that these are
actually the same matrix

# Backpropagation through time



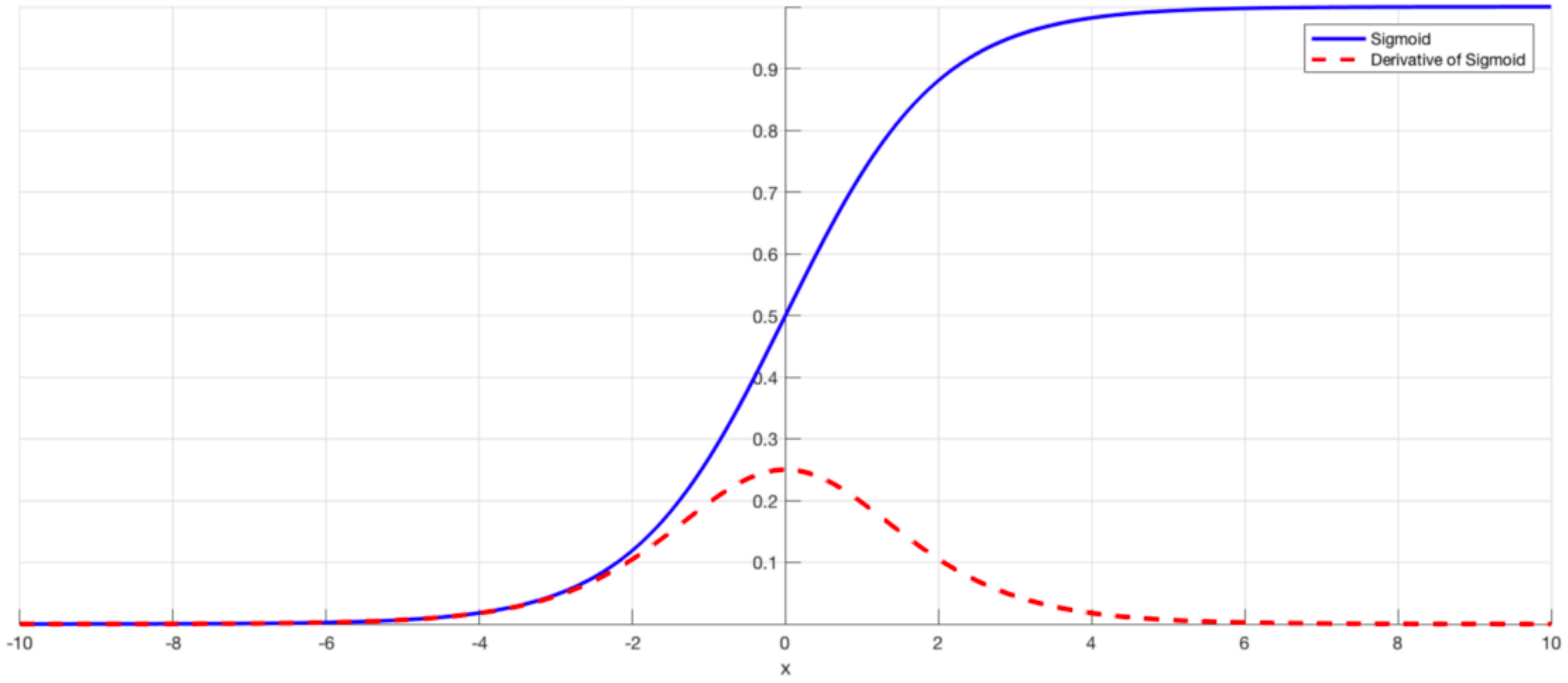
# Vanishing Gradients

- **Learning Problem:** Long unrolled networks will crush gradients that backpropagate to earlier time steps

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

$$u = W_{hx}x_t + W_{hh}h_{t-1} + b_h$$

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

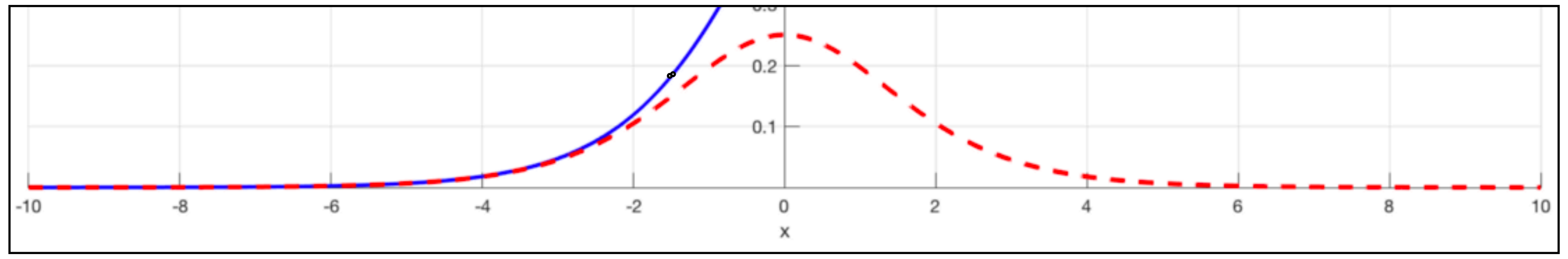


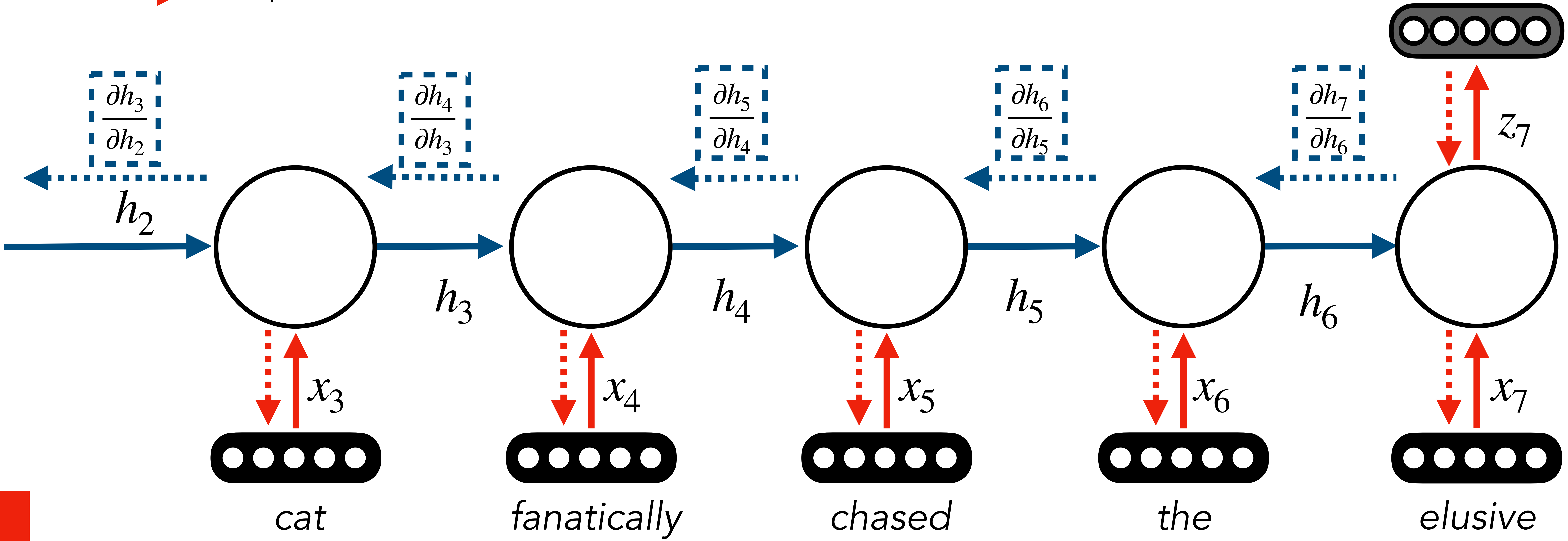
# Backpropagation through time

Gradient flow

Output flow

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



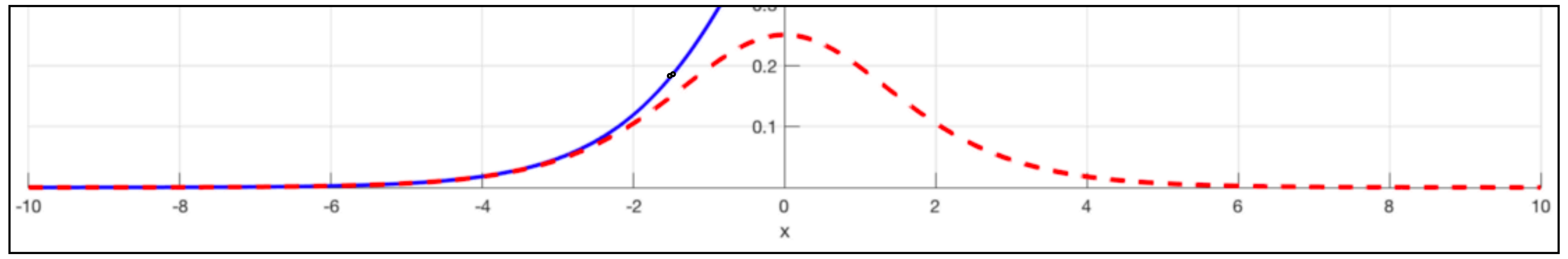


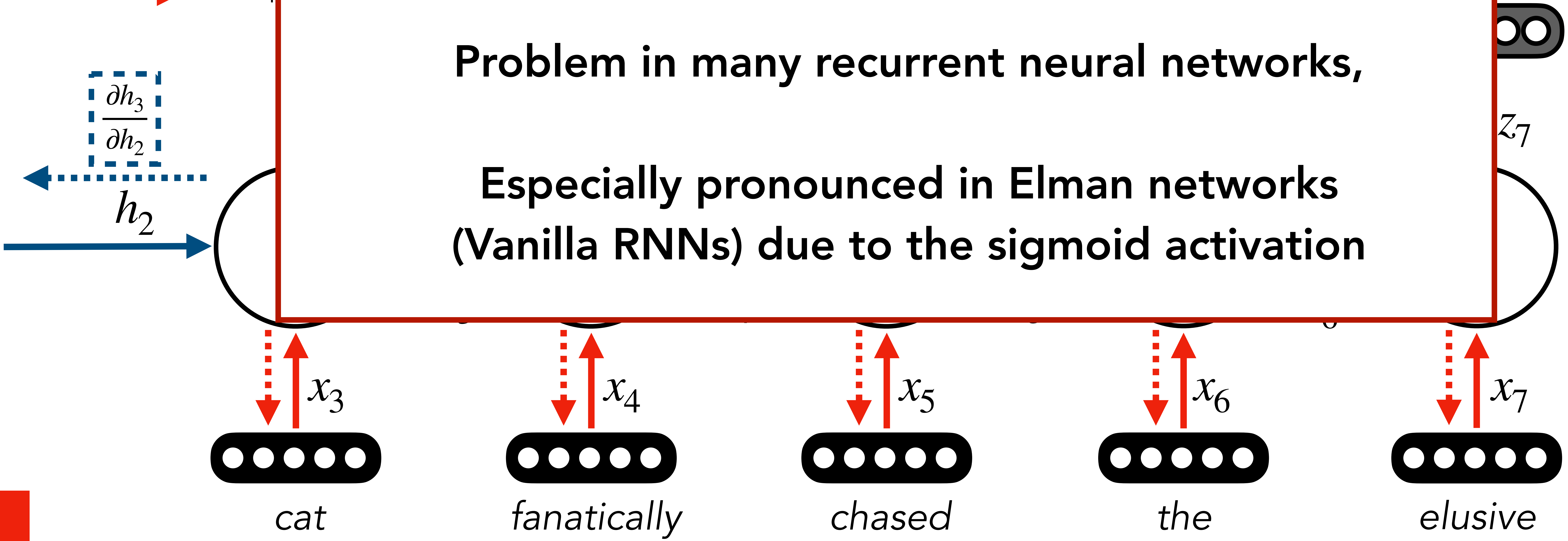
# Backpropagation through time

Gradient flow

Output flow

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






**Problem in many recurrent neural networks,**

**Especially pronounced in Elman networks (Vanilla RNNs) due to the sigmoid activation**

# Question

**How could we fix this vanishing gradient problem?**

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

# In Practice

- Computing gradients by hand is hard!
    - Though potentially a good way to sanity check whether you network behaves the way you expect
- Most modern software packages for deep learning use automatic differentiation to compute gradients automatically from the forward pass
- Only need to define the forward pass of your model (much easier!)
- You'll use PyTorch in this class

# Multiple Layers

# Recap

- Early neural language models (and n-gram models) suffer from **fixed context windows**
- Recurrent neural networks can **theoretically** learn to model an **unbounded context length** using back propagation through time (BPTT)
- Practically, **vanishing gradients** can still stop many RNNs from learning **long-range dependencies**
- RNNs (and modern variants) remain useful for **many sequence-to-sequence tasks**

# Next week

**More powerful recurrent neural network architectures**

**Encoder-decoder models**

**Guest Lecture: Gail Weiss!**

# References


- Bengio, Y., Ducharme, R., Vincent, P., & Janvin, C. (2003). A Neural Probabilistic Language Model. *Journal of machine learning research*.
- Elman, J.L. (1990). Finding Structure in Time. *Cogn. Sci.*, *14*, 179-211.