Recurrent Neural Networks

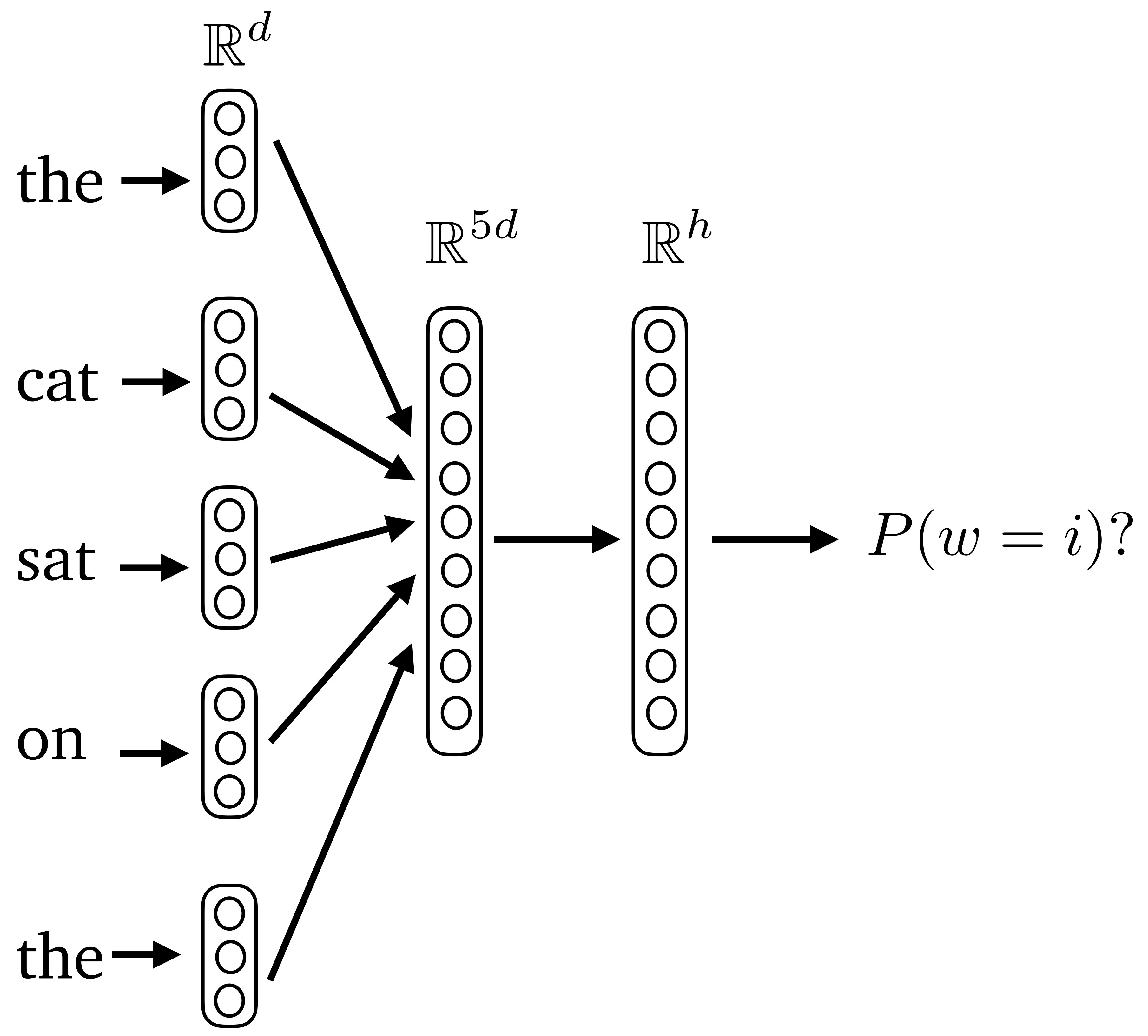
Antoine Bosselut

Section Outline

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

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)

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

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

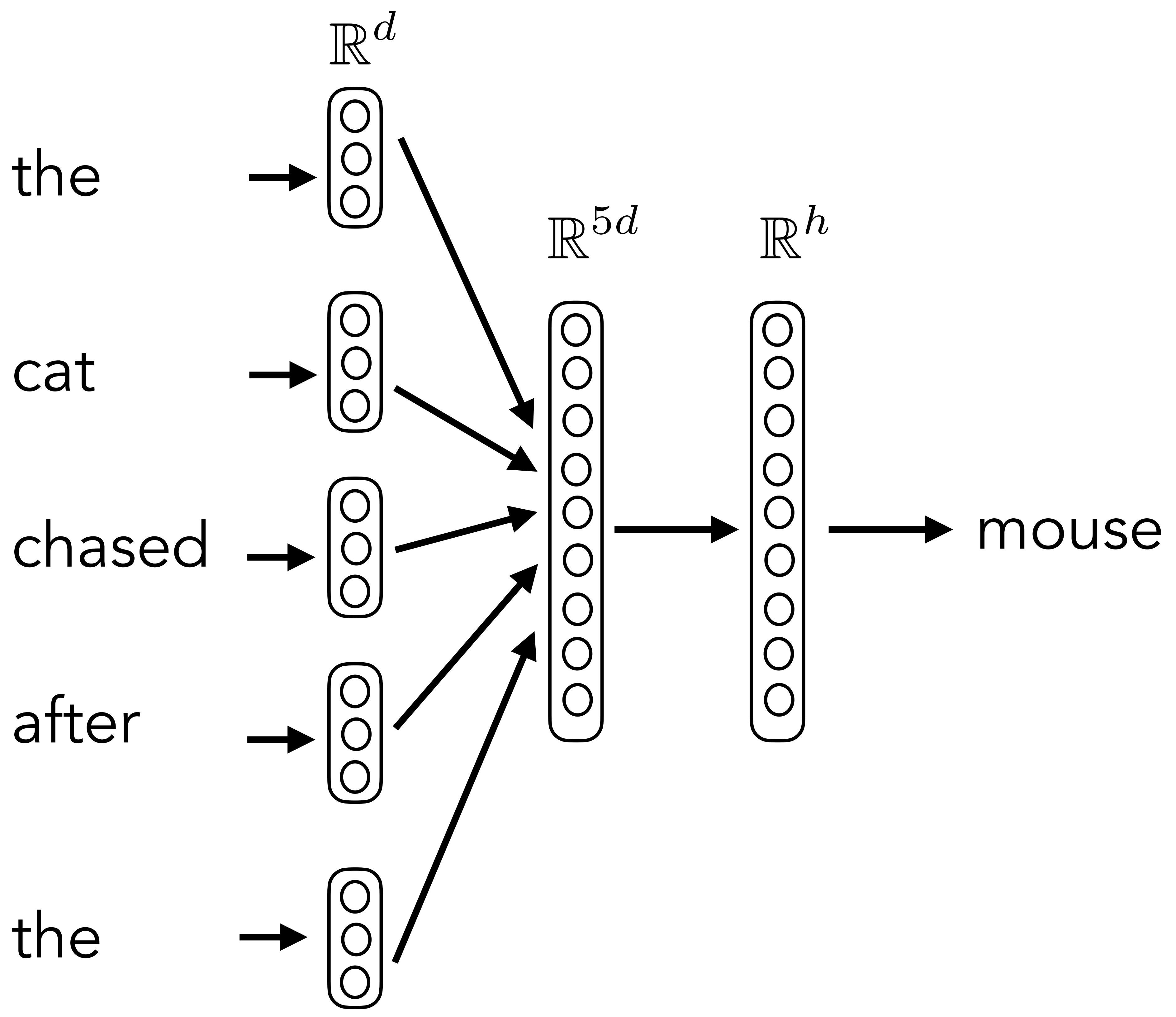
(Bengio et al., 2003): A Neural Probabilistic Language Model

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**

Fixed-context Neural Language Models



P(mouse | the cat chased 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

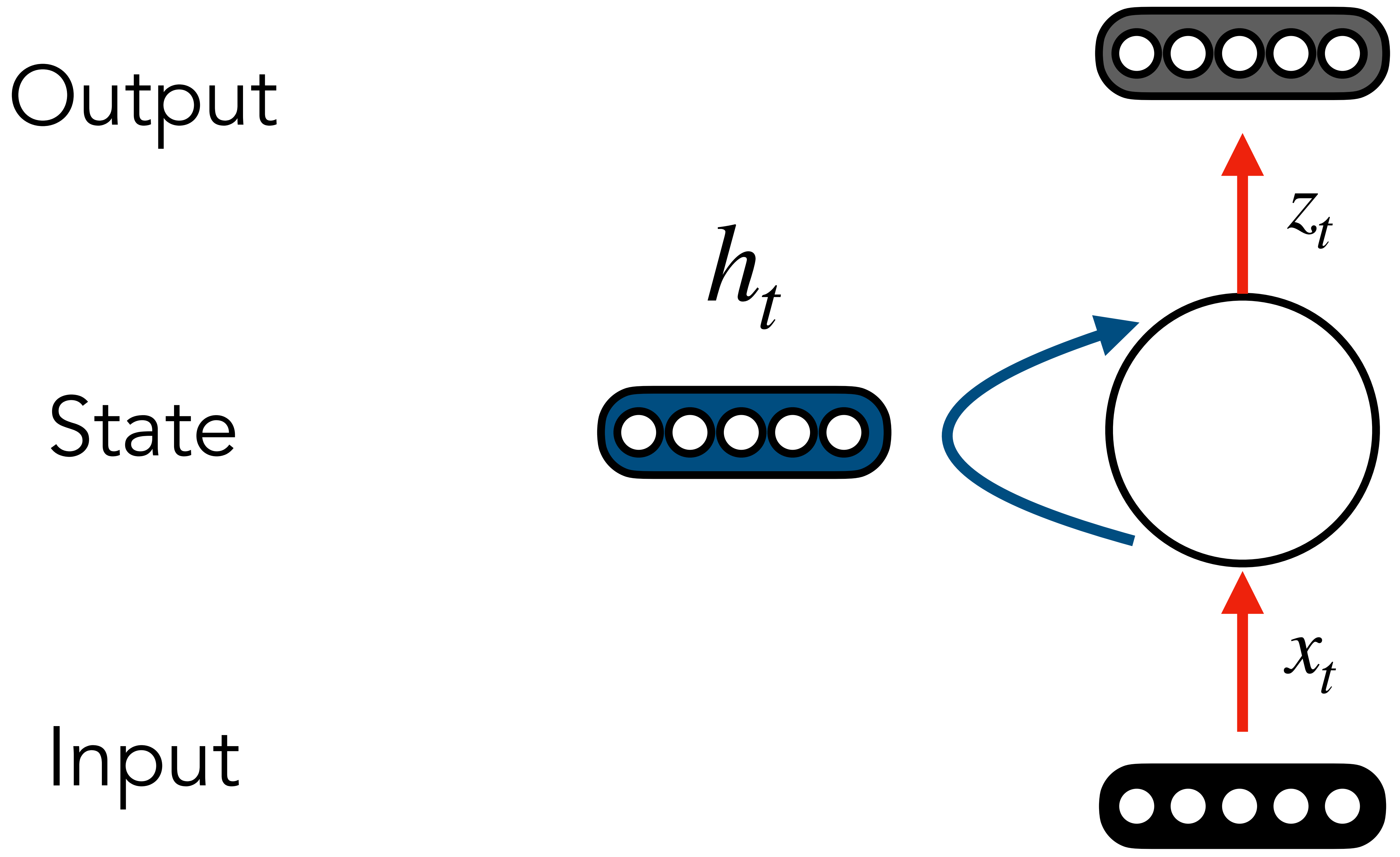
Advantages vs. Disadvantages

- 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**
- Weights in **W** aren't shared across embeddings in the window (computationally inefficient!)
- Enlarging the window size makes the weight matrix **W** larger (more computationally expensive!)

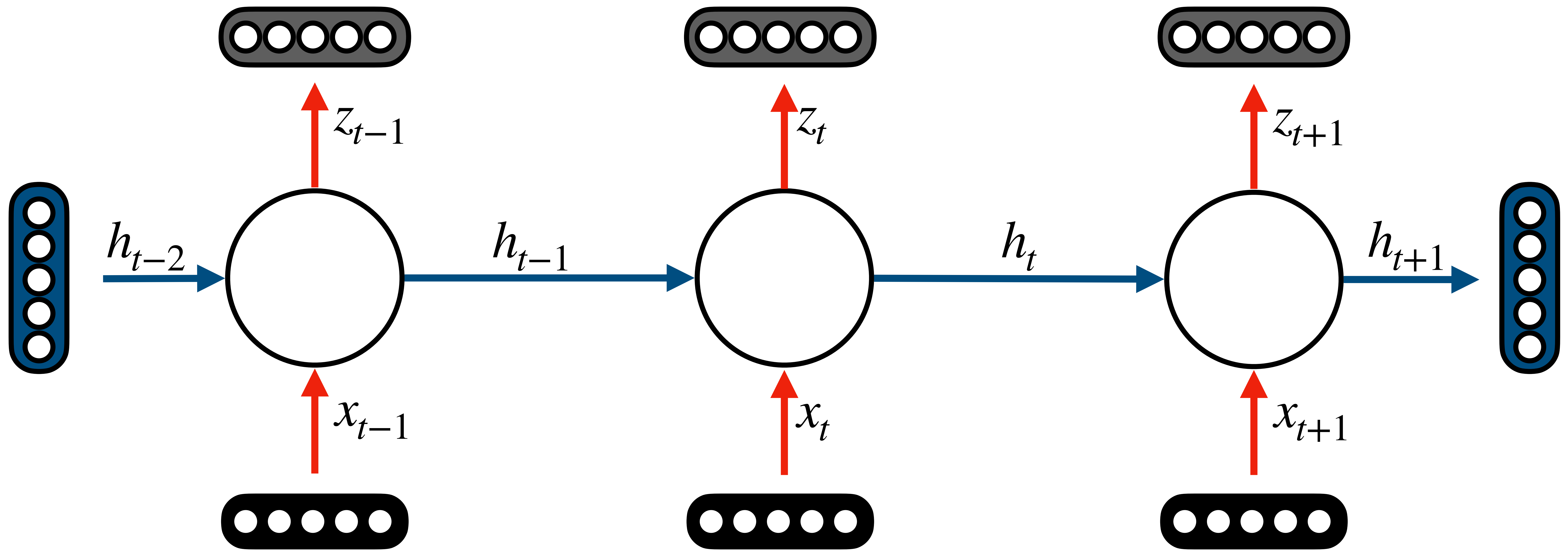
Recurrent Neural Networks

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



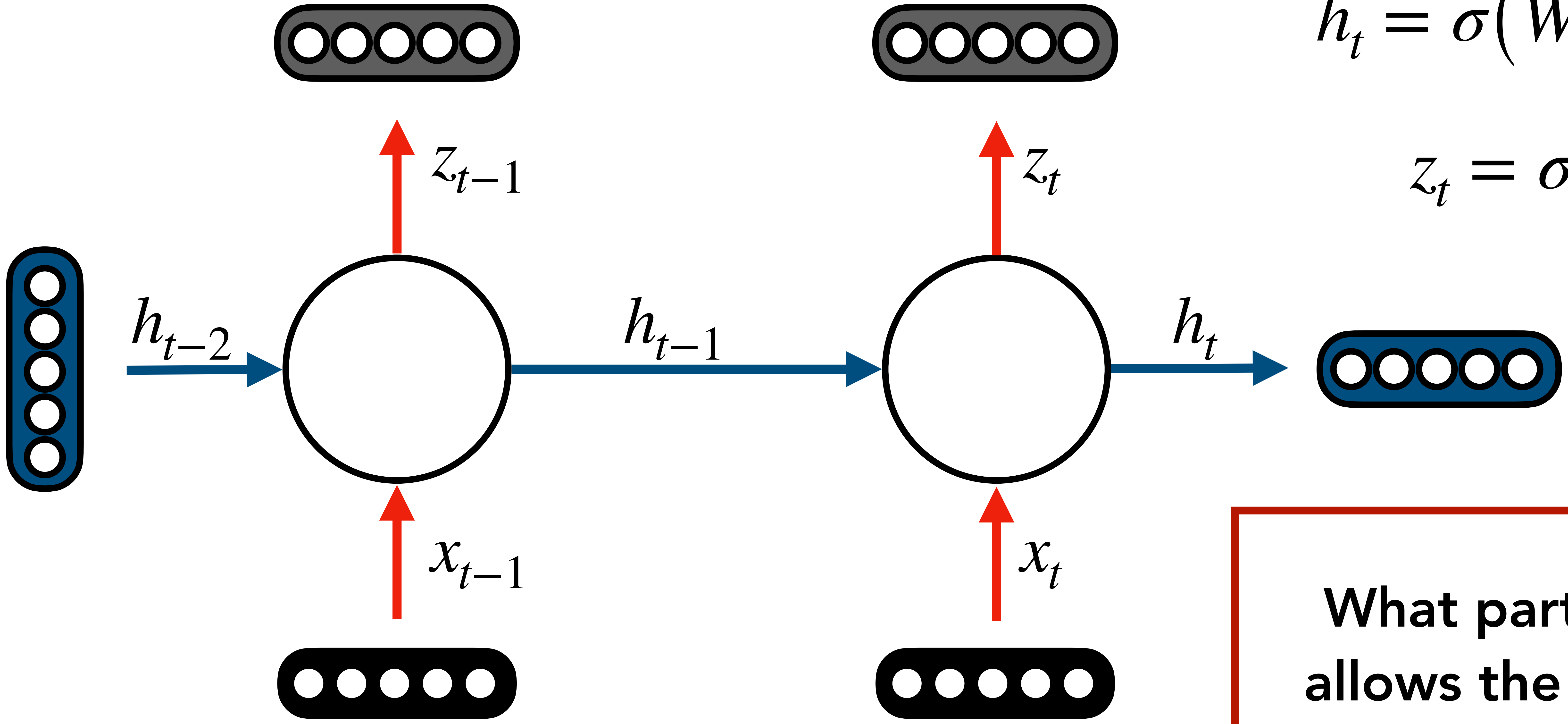
Recurrent Neural Networks

Unrolling the RNN across all time steps gives full computation graph



Allows for learning from entire sequence history, regardless of length

Classical RNN: Elman Network



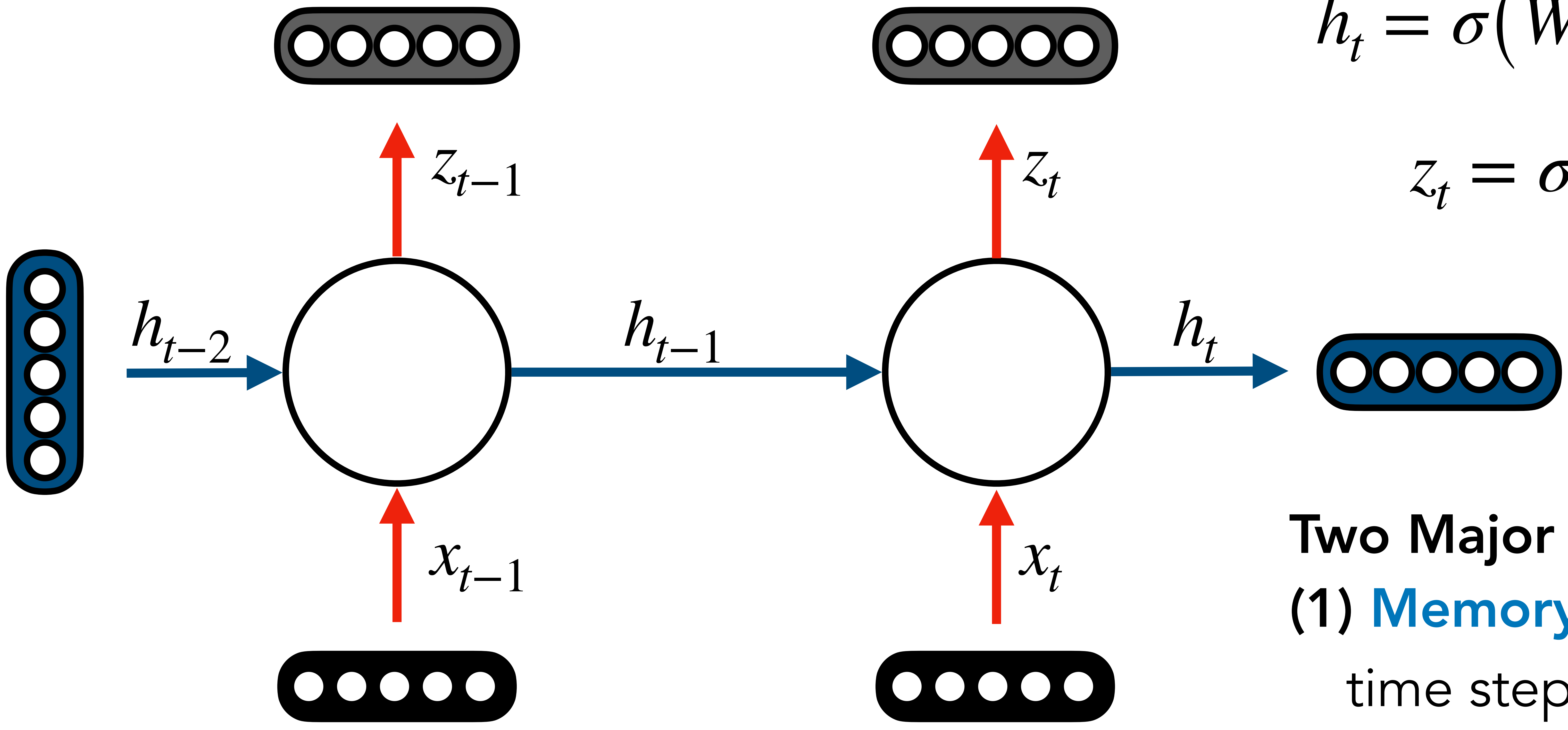
$$h_t = \sigma\left(W_{hx}x_t + W_{hh}h_{t-1} + b_h\right)$$

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

What part of this formulation allows the model to maintain a state of previously input tokens ?

(Elman, 1990)

Classical RNN: Elman Network



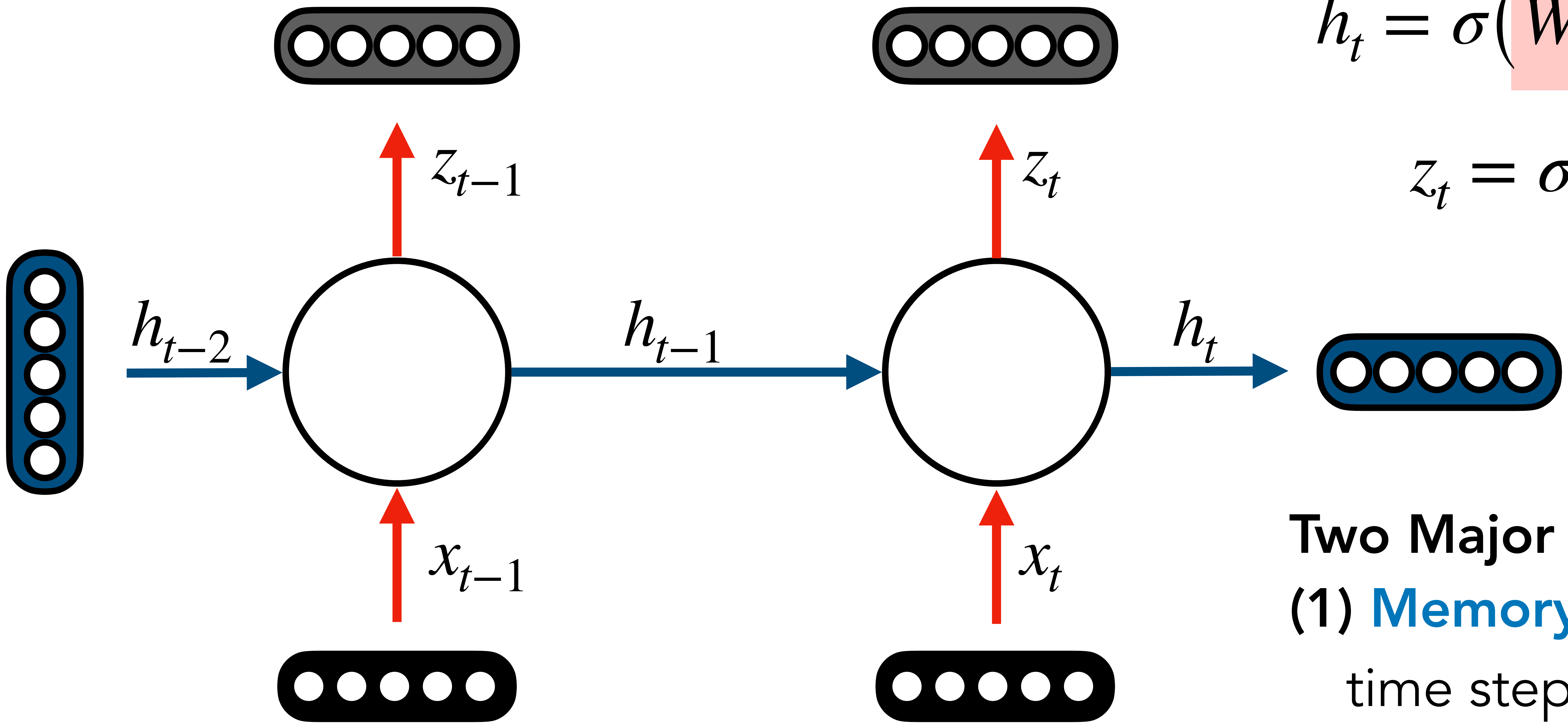
$$h_t = \sigma\left(W_{hx}x_t + W_{hh}h_{t-1} + b_h\right)$$

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

Two Major Components:

(1) **Memory** is maintained across time steps by the state h.

Classical RNN: Elman Network



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

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

Two Major Components:

(1) **Memory** is maintained across time steps by the state h.

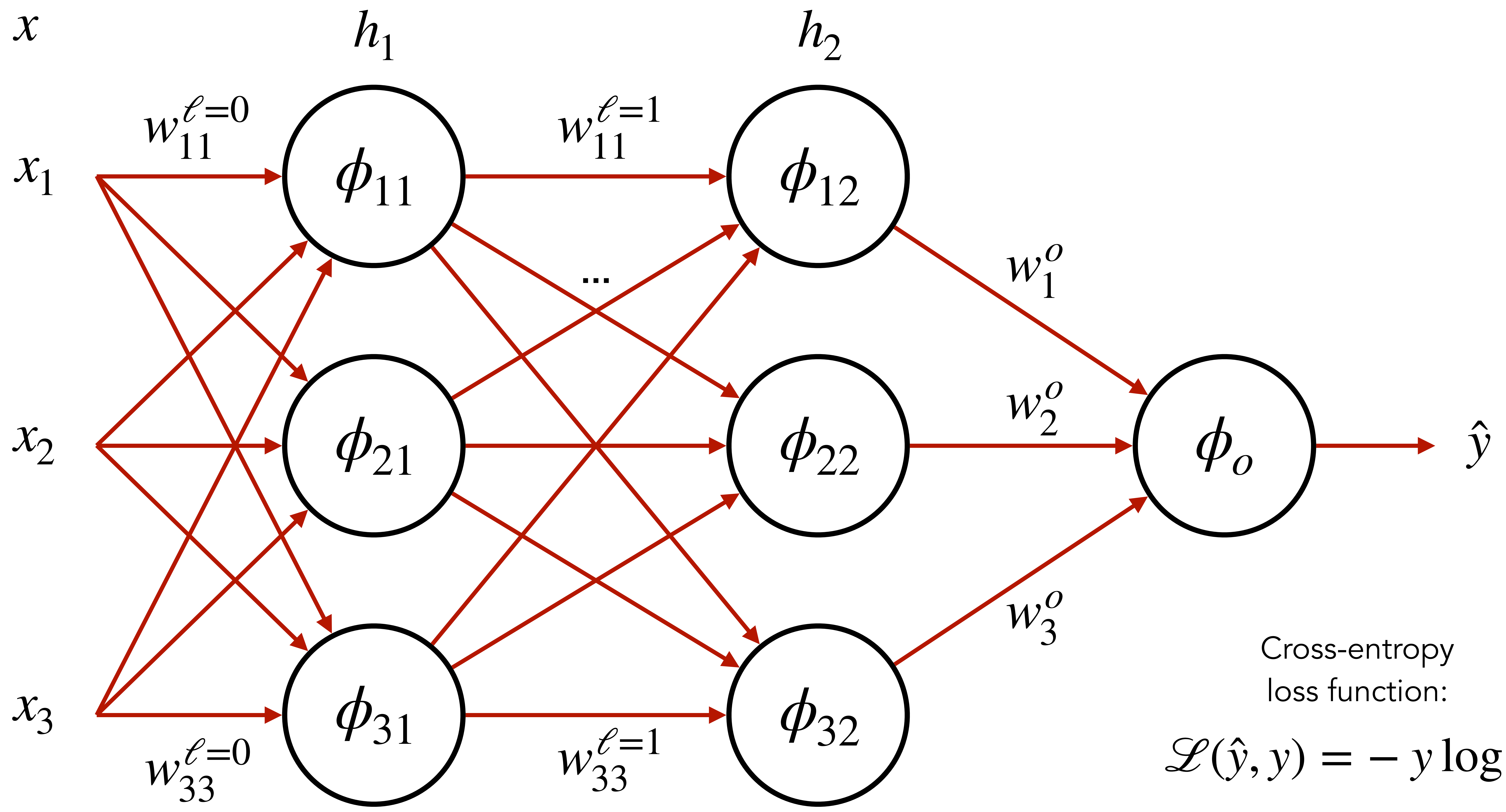
(2) **Parameter matrices** are shared across all time steps.

(Elman, 1990)

Objective

Learn parameter matrices W_{hh}, W_{hx}, W_{zh} in the Elman network such that the RNN can represent textual sequences

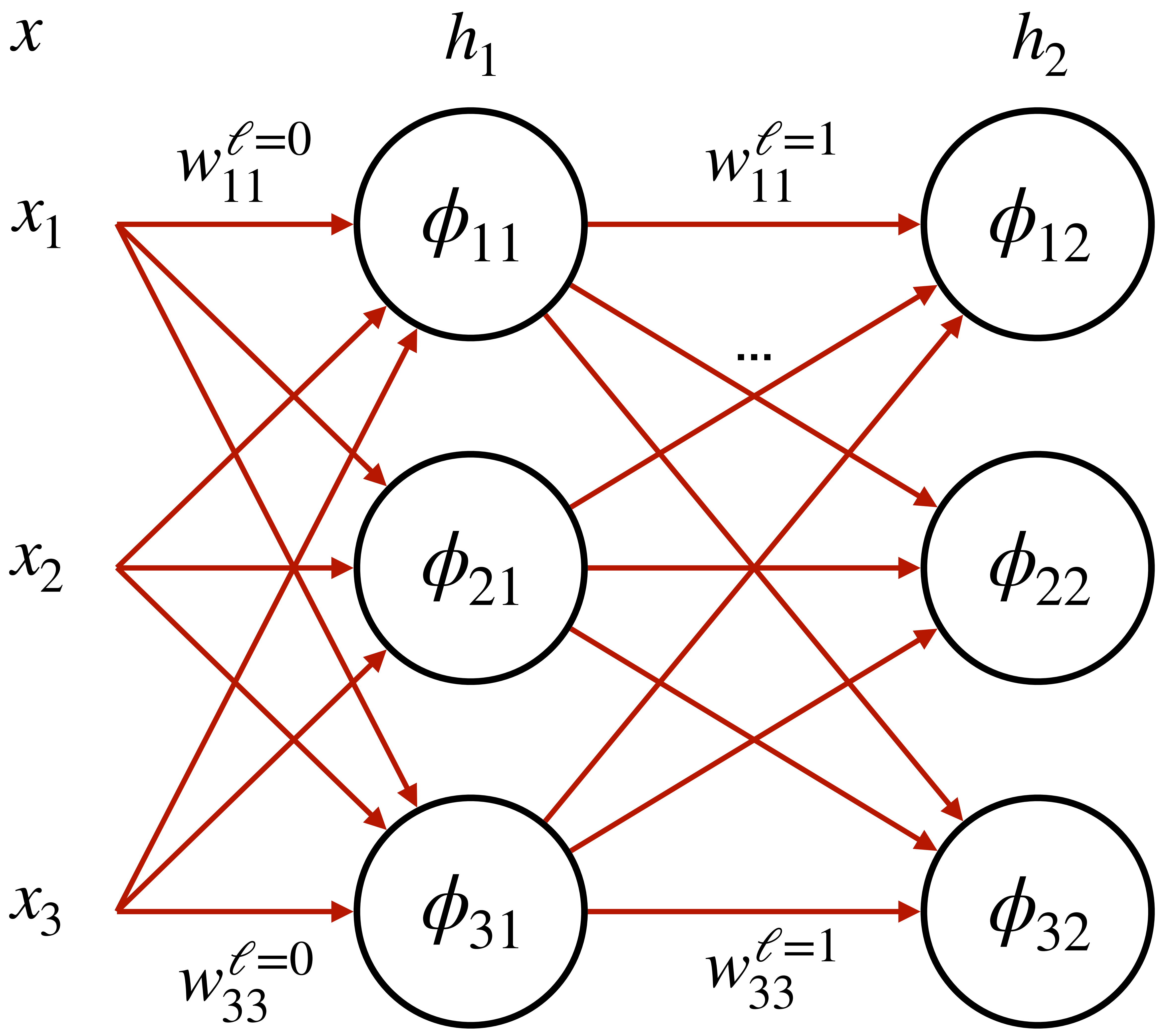
Backpropagation Review: FFNs



Cross-entropy loss function:

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

Backpropagation Review: FFNs



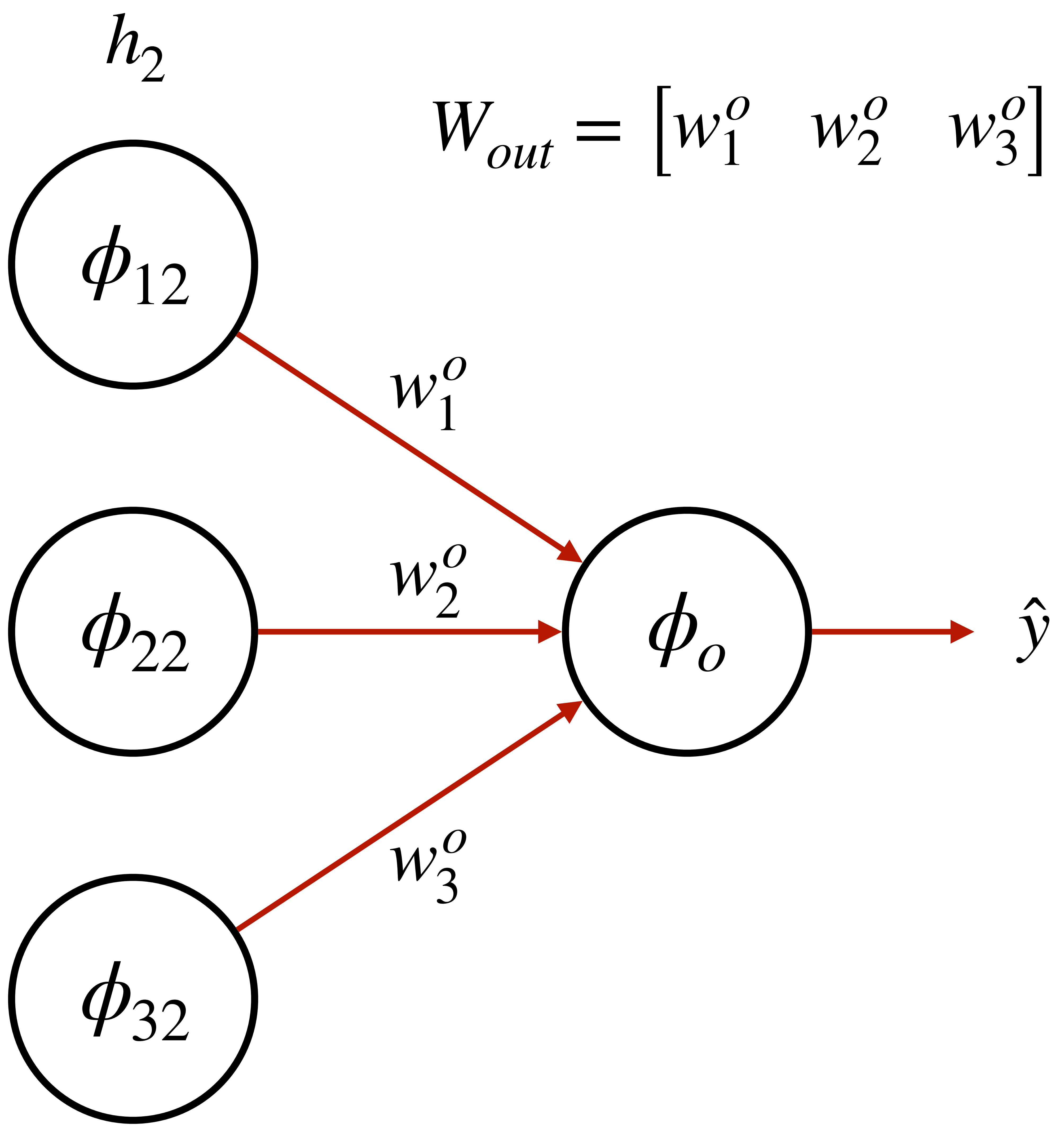
$$W_0 = \begin{bmatrix} w_{11}^{\ell=0} & w_{21}^{\ell=0} & w_{31}^{\ell=0} \\ w_{12}^{\ell=0} & w_{22}^{\ell=0} & w_{32}^{\ell=0} \\ w_{13}^{\ell=0} & w_{23}^{\ell=0} & w_{33}^{\ell=0} \end{bmatrix}$$

$$W_1 = \begin{bmatrix} w_{11}^{\ell=1} & w_{21}^{\ell=1} & w_{31}^{\ell=1} \\ w_{12}^{\ell=1} & w_{22}^{\ell=1} & w_{32}^{\ell=1} \\ w_{13}^{\ell=1} & w_{23}^{\ell=1} & w_{33}^{\ell=1} \end{bmatrix}$$

$$h_1 = \phi_1(W_0 x)$$

$$h_2 = \phi_2(W_1 h_1)$$

Backpropagation Review: FFNs



$$W_{out} = \begin{bmatrix} w_1^o & w_2^o & w_3^o \end{bmatrix}$$

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

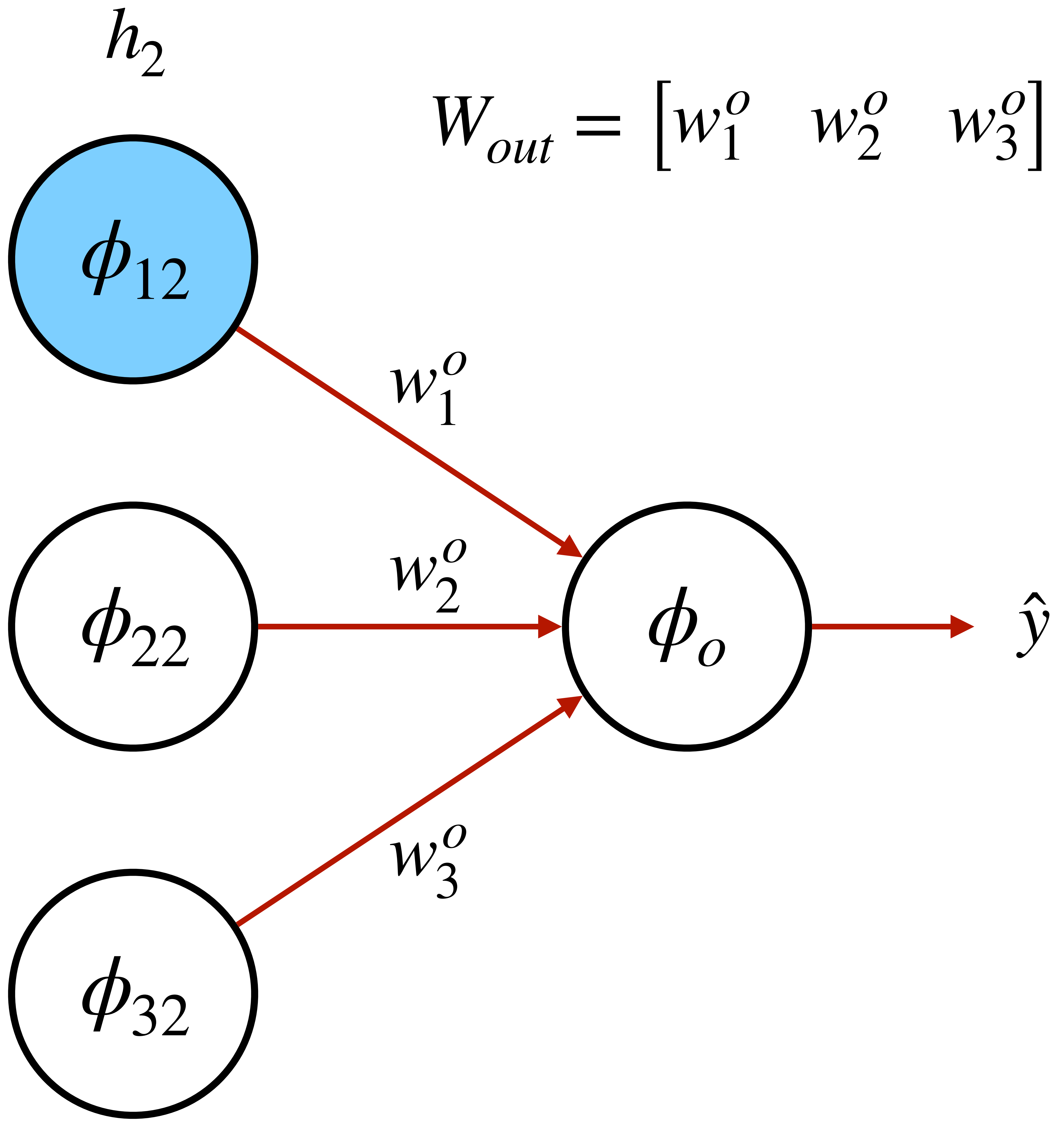
$$P(\hat{y}) = \phi_o(u)$$

$$u = W_{out}h_2 = w_1^o \times \phi_{12}(\,.\,) + w_2^o \times \phi_{22}(\,.\,) + w_3^o \times \phi_{32}(\,.\,)$$

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

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

Backpropagation Review: FFNs



$$W_{out} = \begin{bmatrix} w_1^o & w_2^o & w_3^o \end{bmatrix}$$

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

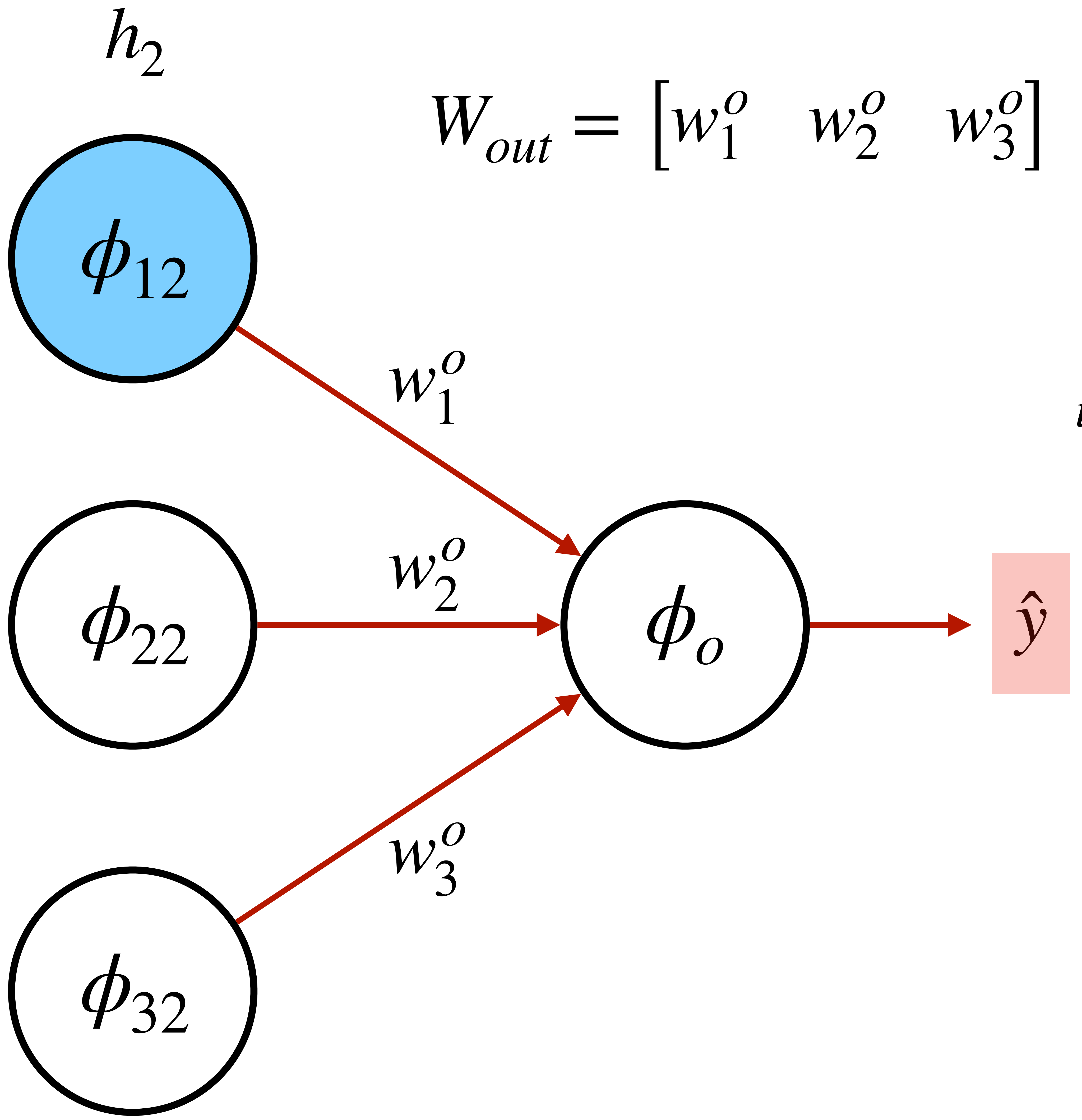
$$P(\hat{y}) = \phi_o(u)$$

$$u = W_{out}h_2 = w_1^o \times \phi_{12}(\,.\,) + w_2^o \times \phi_{22}(\,.\,) + w_3^o \times \phi_{32}(\,.\,)$$

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

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

Backpropagation Review: FFNs



$$W_{out} = \begin{bmatrix} w_1^o & w_2^o & w_3^o \end{bmatrix}$$

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

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

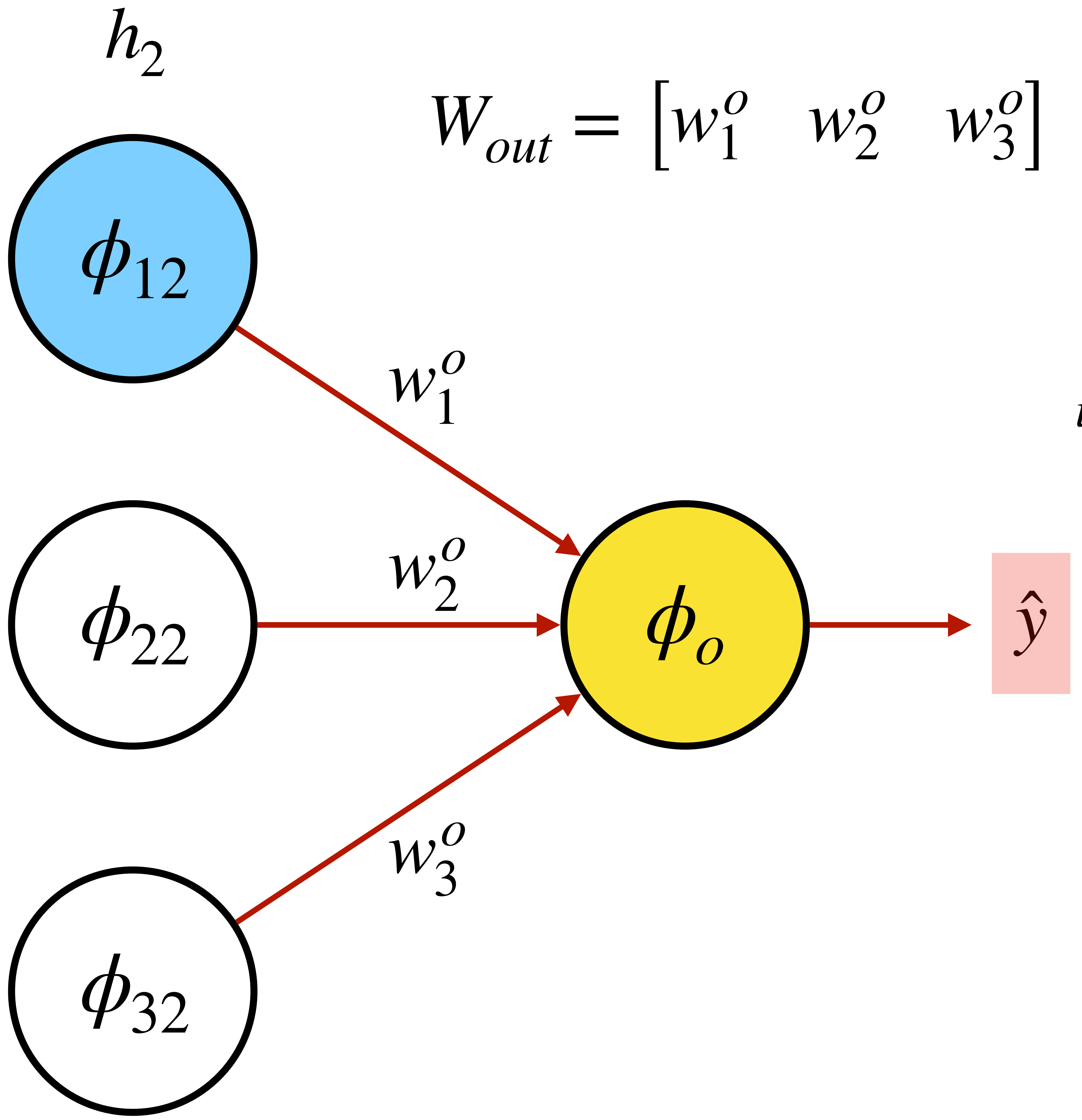
$$u = W_{out} h_2 = w_1^o \times \phi_{12}(\,.\,) + w_2^o \times \phi_{22}(\,.\,) + w_3^o \times \phi_{32}(\,.\,)$$

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

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

Depends on label *y*

Backpropagation Review: FFNs



$$W_{out} = \begin{bmatrix} w_1^o & w_2^o & w_3^o \end{bmatrix}$$

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

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

$$u = W_{out} h_2 = w_1^o \times \phi_{12}(\,.\,) + w_2^o \times \phi_{22}(\,.\,) + w_3^o \times \phi_{32}(\,.\,)$$

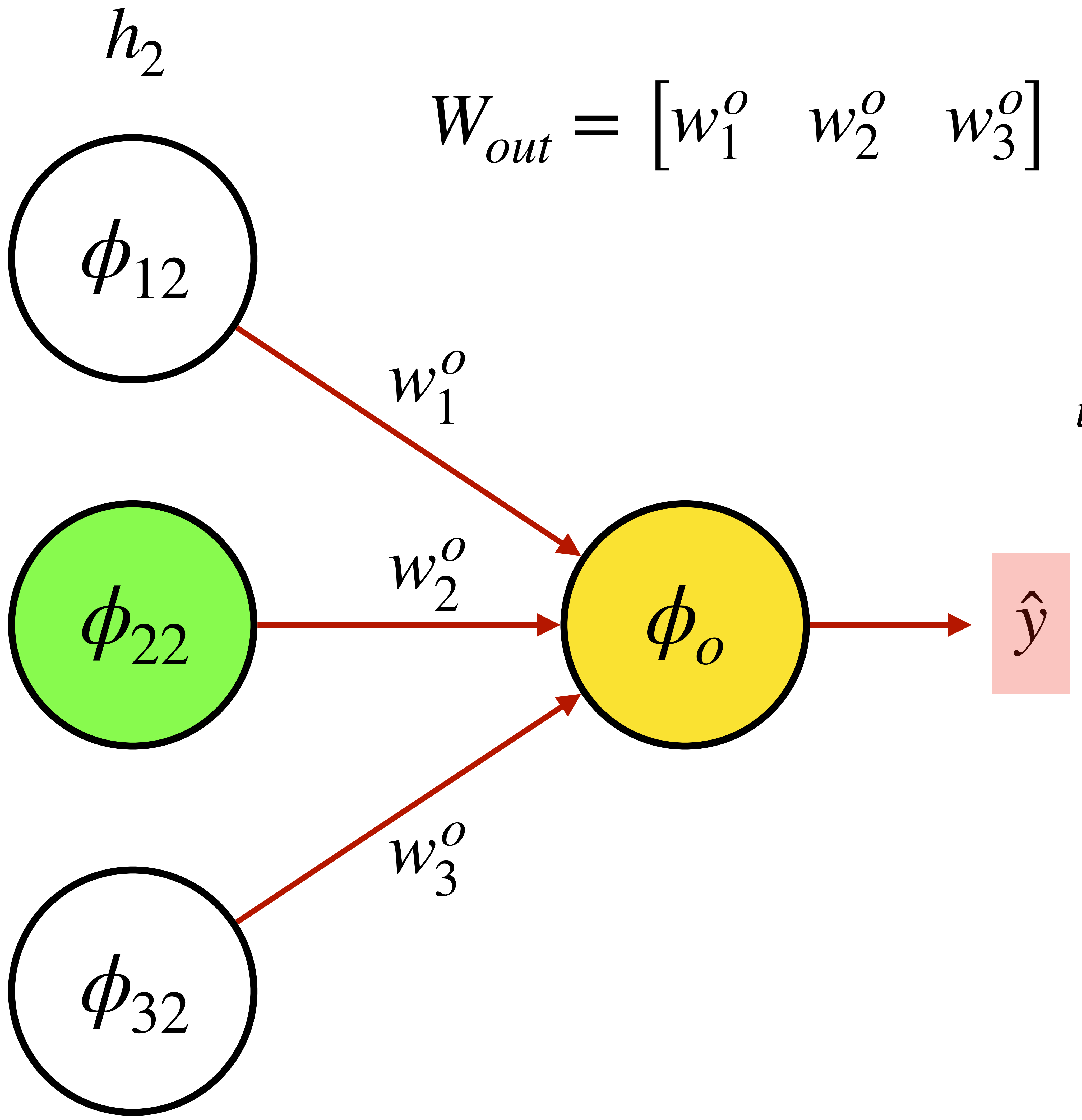
$$\frac{\partial \mathscr{L}(\hat{y}, y)}{\partial \phi_{12}(\,.\,)} = \frac{\partial \mathscr{L}(\hat{y}, y)}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial u} \frac{\partial u}{\partial \phi_{12}(\,.\,)}$$

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

Depends on label y

Depends on ϕ_o

Backpropagation Review: FFNs



$$W_{out} = \begin{bmatrix} w_1^o & w_2^o & w_3^o \end{bmatrix}$$

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

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

$$u = W_{out} h_2 = w_1^o \times \phi_{12}(\,.\,) + w_2^o \times \phi_{22}(\,.\,) + w_3^o \times \phi_{32}(\,.\,)$$

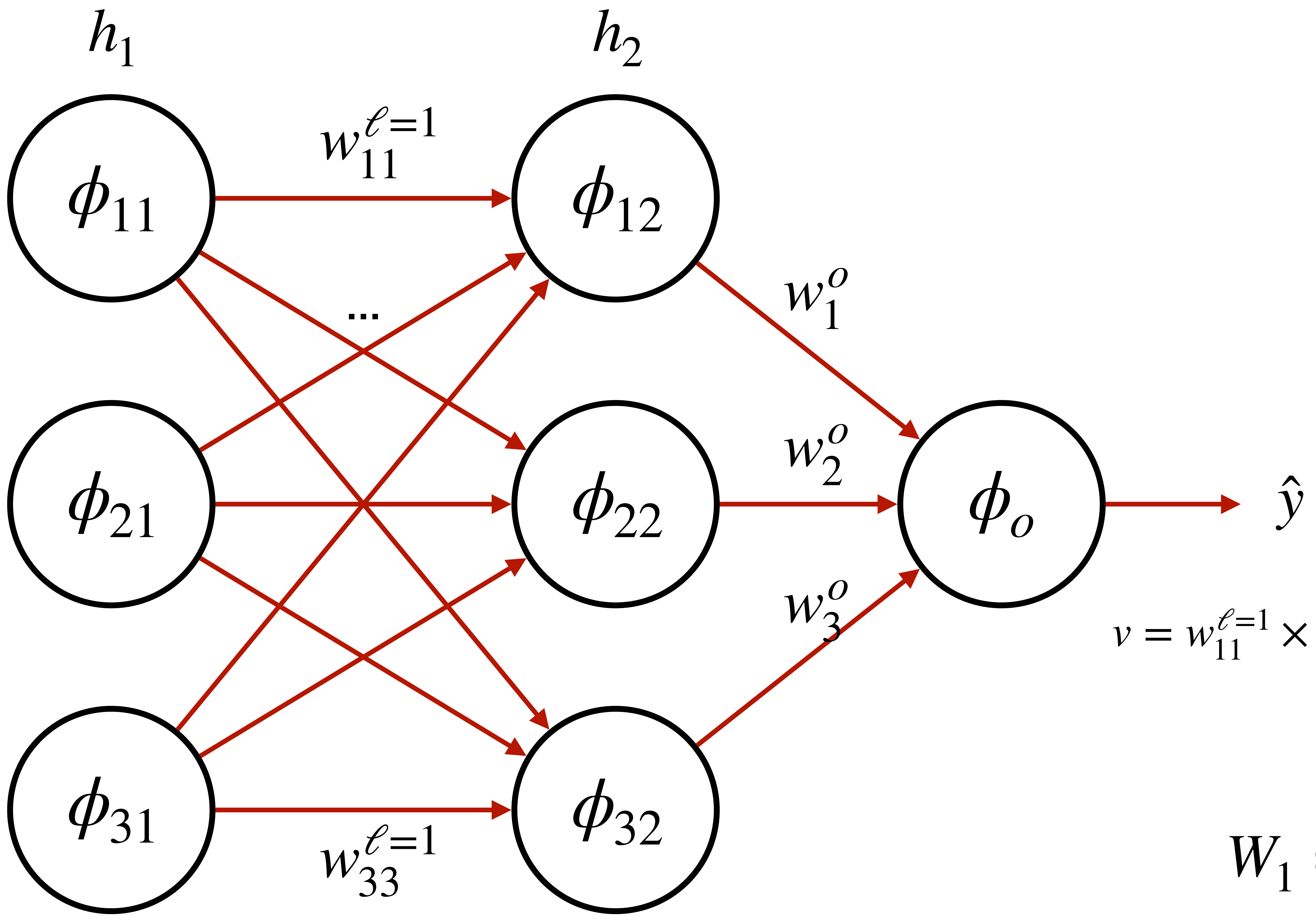
$$\frac{\partial \mathcal{L}(\hat{y}, y)}{\partial \phi_{22}(\,.\,)} = \frac{\partial \mathcal{L}(\hat{y}, y)}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial u} \frac{\partial u}{\partial \phi_{22}(\,.\,)}$$

$$= \frac{\partial \mathcal{L}(\hat{y}, y)}{\partial \hat{y}} \frac{\partial \phi_o(u)}{\partial u} w_2^0$$

Depends on label y

Depends on ϕ_o

Backpropagation Review: FFNs



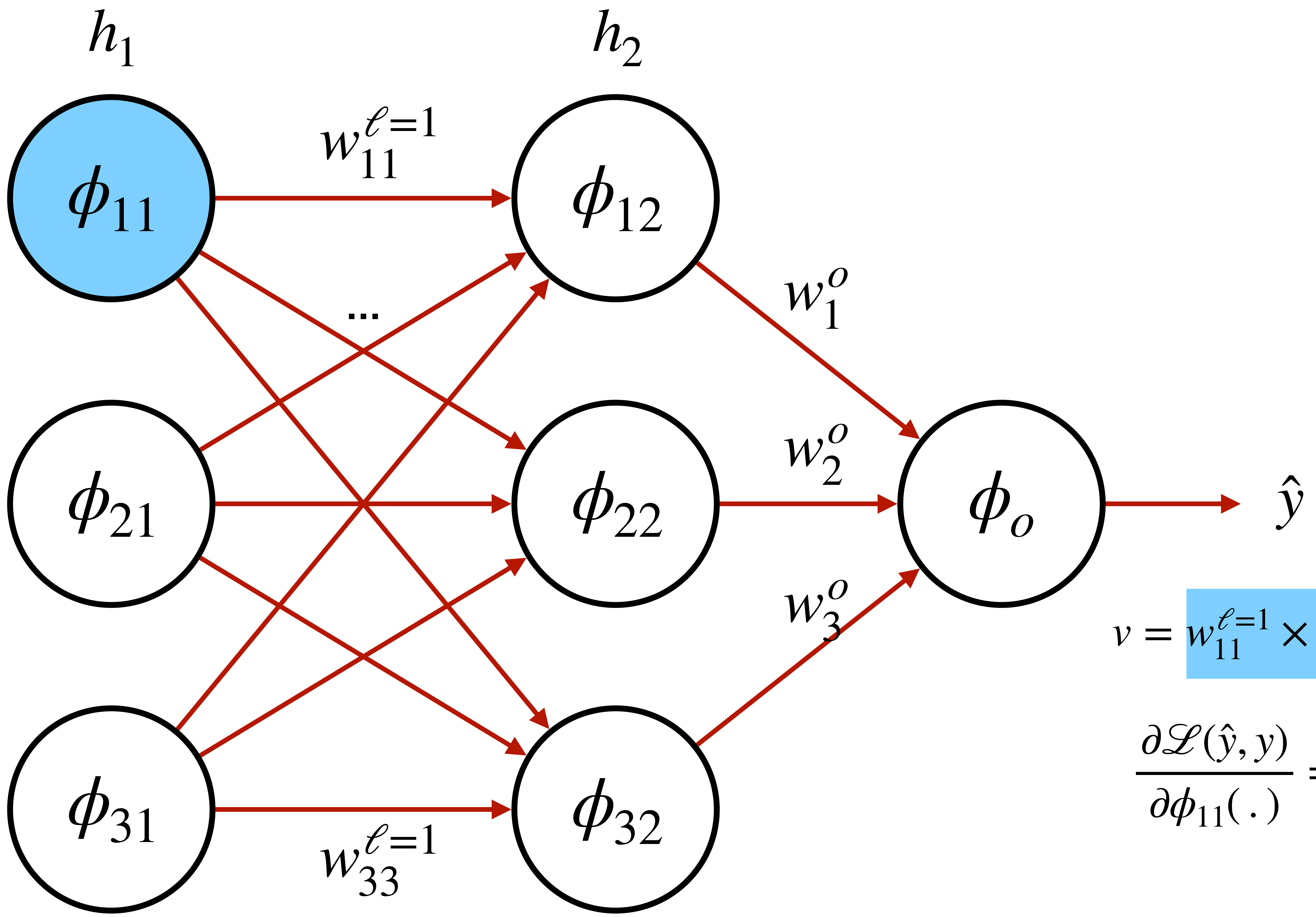
$$\frac{\partial \mathscr{L}(\hat{y}, y)}{\partial \phi_{12}(\,.\,)} = \frac{\partial \mathscr{L}(\hat{y}, y)}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial u} \frac{\partial u}{\partial \phi_{12}(\,.\,)}$$

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

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

$$W_1 = \begin{bmatrix} w_{11}^{\ell=1} & w_{21}^{\ell=1} & w_{31}^{\ell=1} \\ w_{12}^{\ell=1} & w_{22}^{\ell=1} & w_{32}^{\ell=1} \\ w_{13}^{\ell=1} & w_{23}^{\ell=1} & w_{33}^{\ell=1} \end{bmatrix}$$

Backpropagation Review: FFNs



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

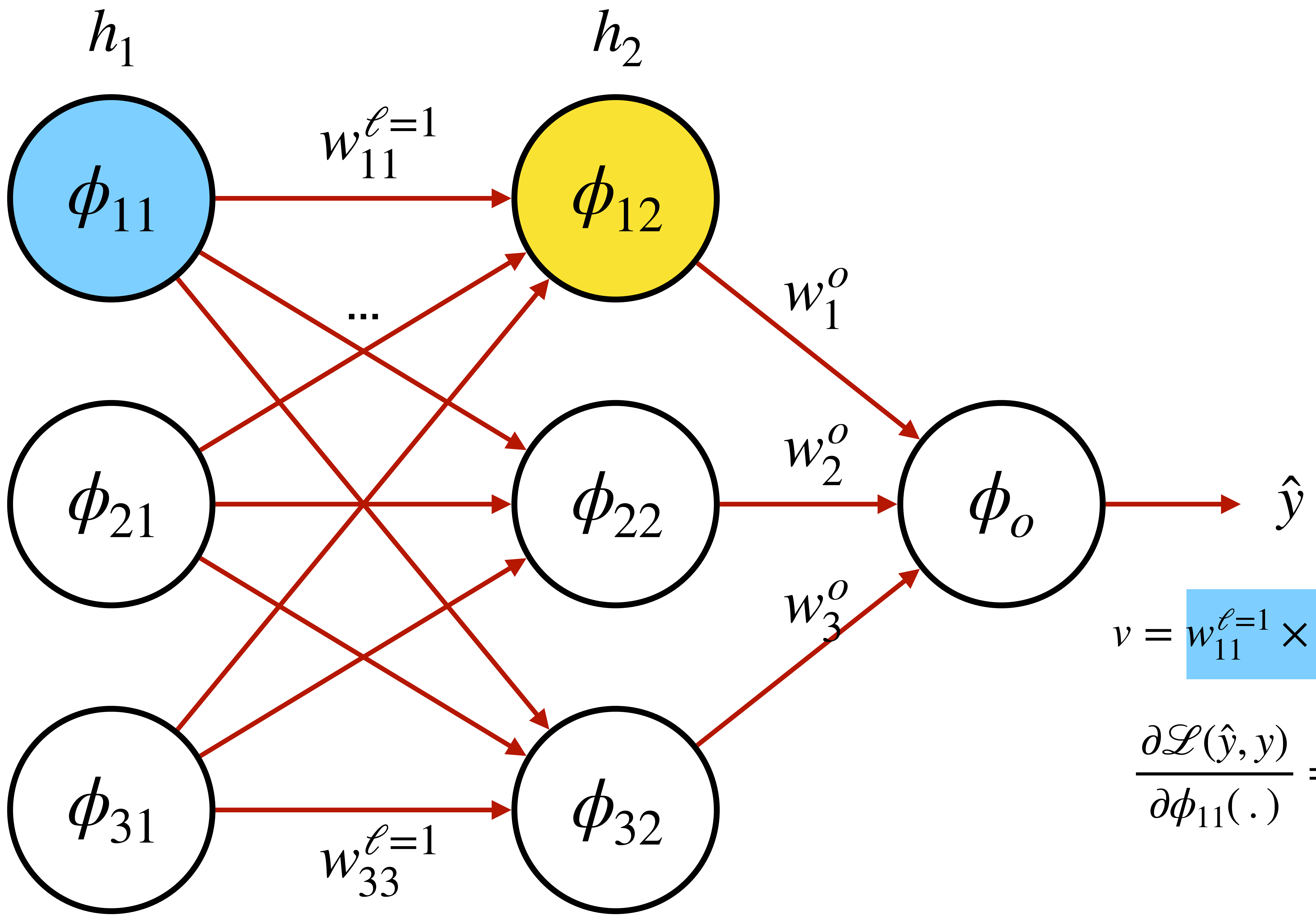
$$= \frac{\partial \mathcal{L}(\hat{y}, y)}{\partial \hat{y}} \frac{\partial \phi_o(u)}{\partial u} w_1^0$$

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

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

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

Backpropagation Review: FFNs



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

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

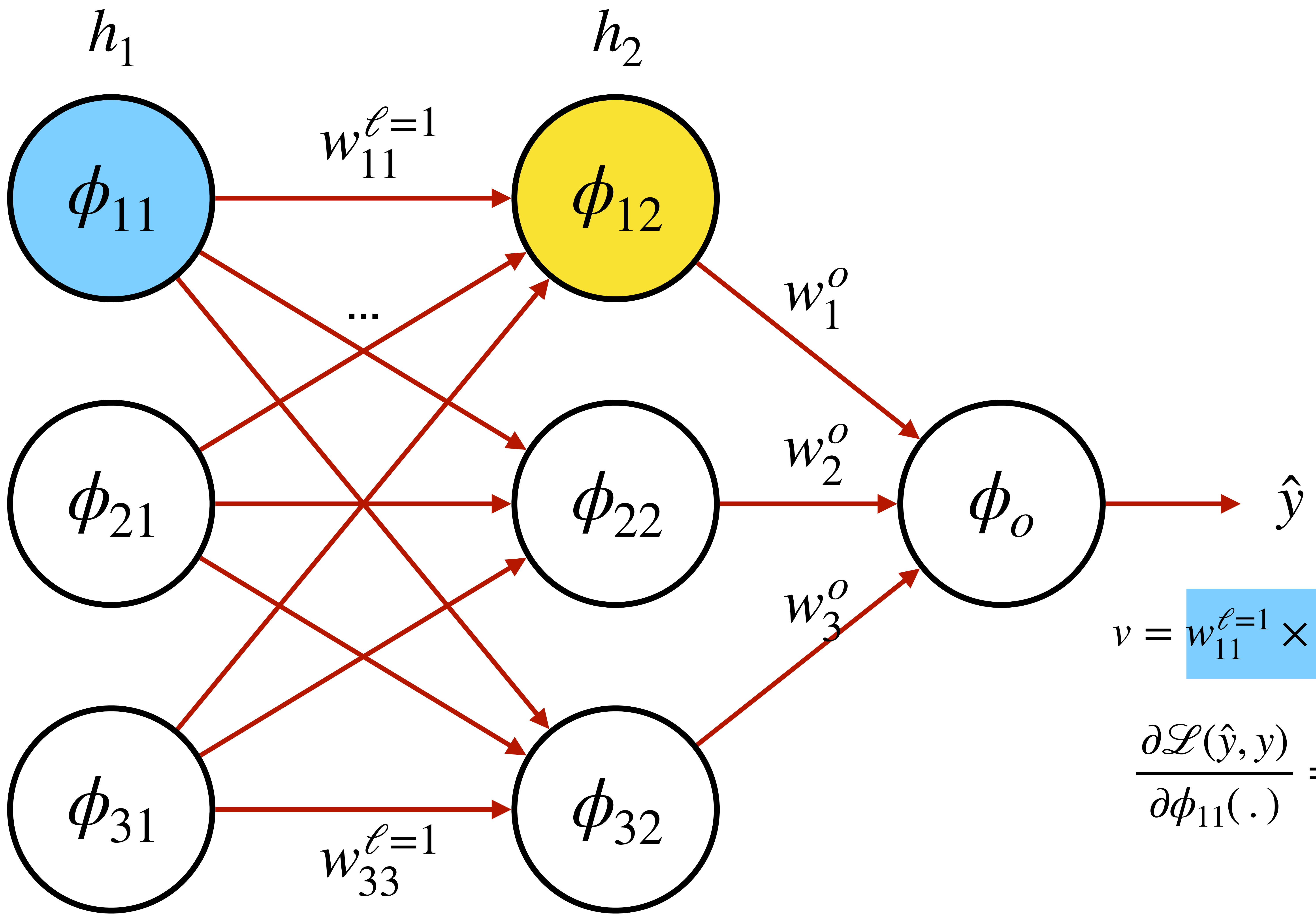
Depends on ϕ_{12}

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

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

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

Backpropagation Review: FFNs



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

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

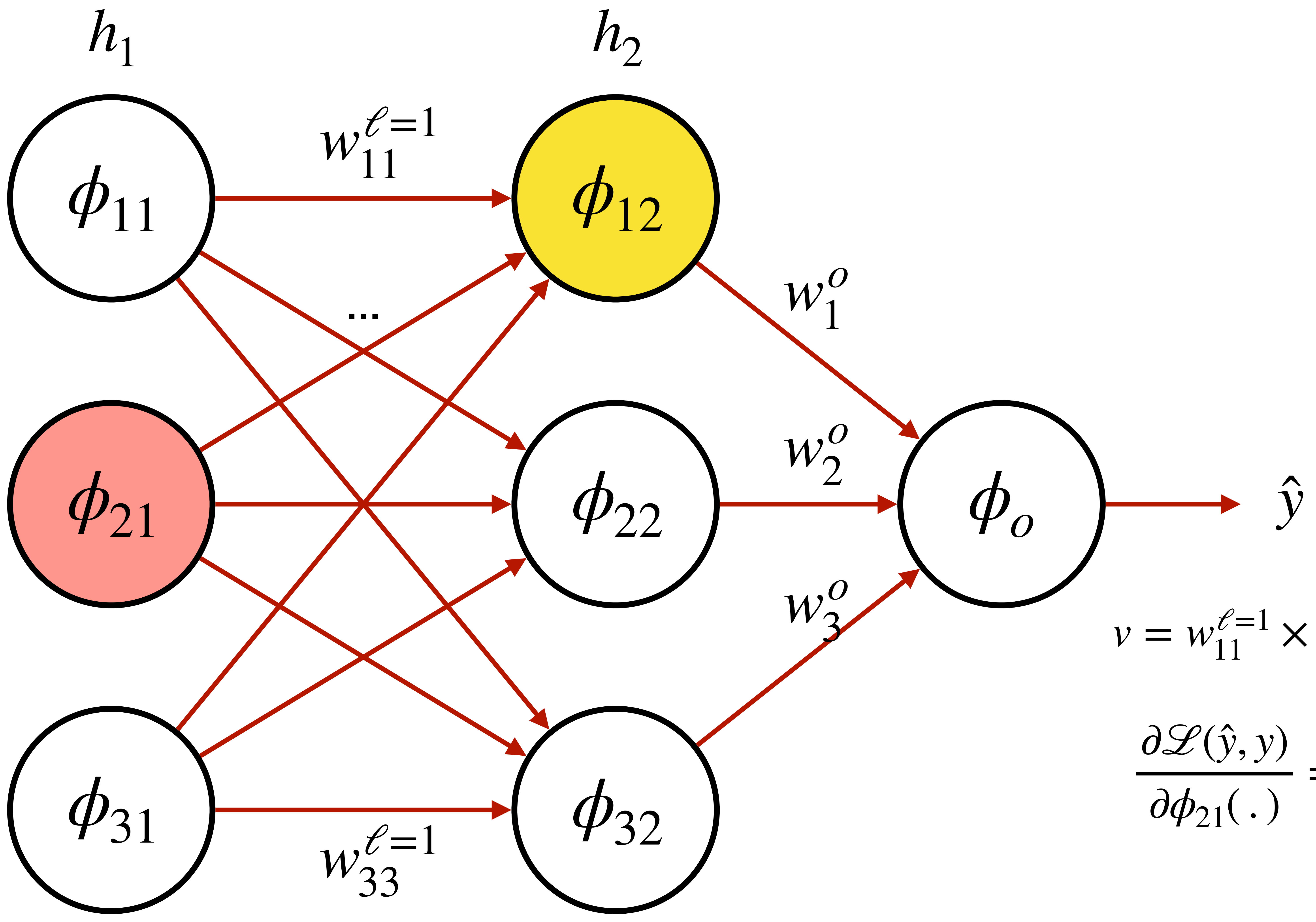
Depends on ϕ_{12}

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

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

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

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}

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

$$\frac{\partial \mathcal{L}(\hat{y},y)}{\partial \phi_{21}(\,.\,)} = \frac{\partial \mathcal{L}(\hat{y},y)}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial u} \frac{\partial u}{\partial \phi_{12}(v)} \frac{\partial \phi_{12}(v)}{\partial v} \frac{\partial v}{\partial \phi_{21}(\,.\,)} + \ldots$$

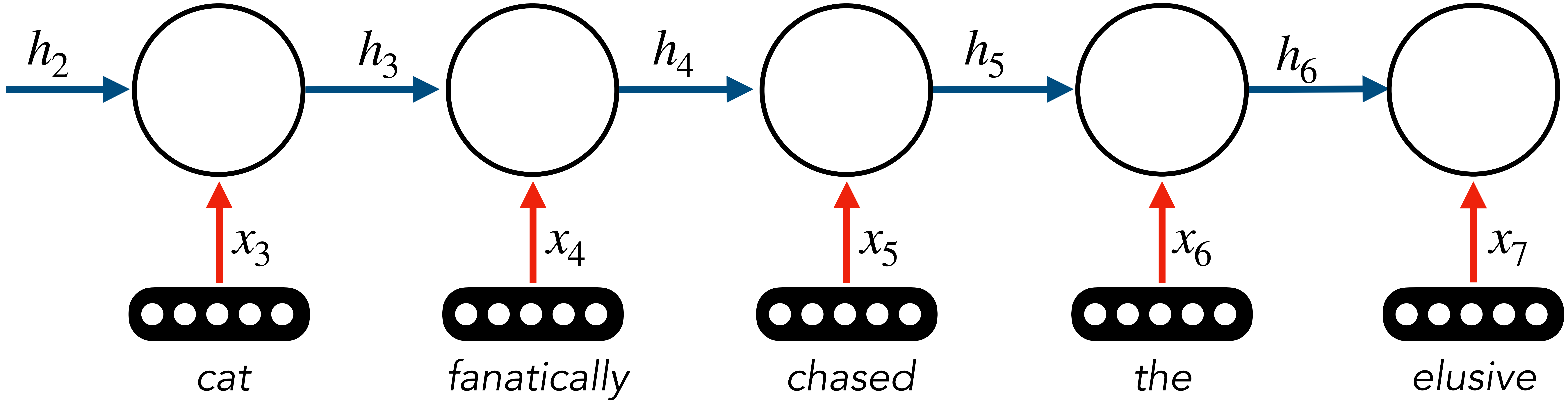
$$= \frac{\partial \mathcal{L}(\hat{y},y)}{\partial \hat{y}} \frac{\partial \phi_o(u)}{\partial u} w_1^0 \frac{\partial \phi_{12}(v)}{\partial v} w_{21}^{\ell=1} + \ldots$$

Question

How would we extend backpropagation to a recurrent neural network?

Recall

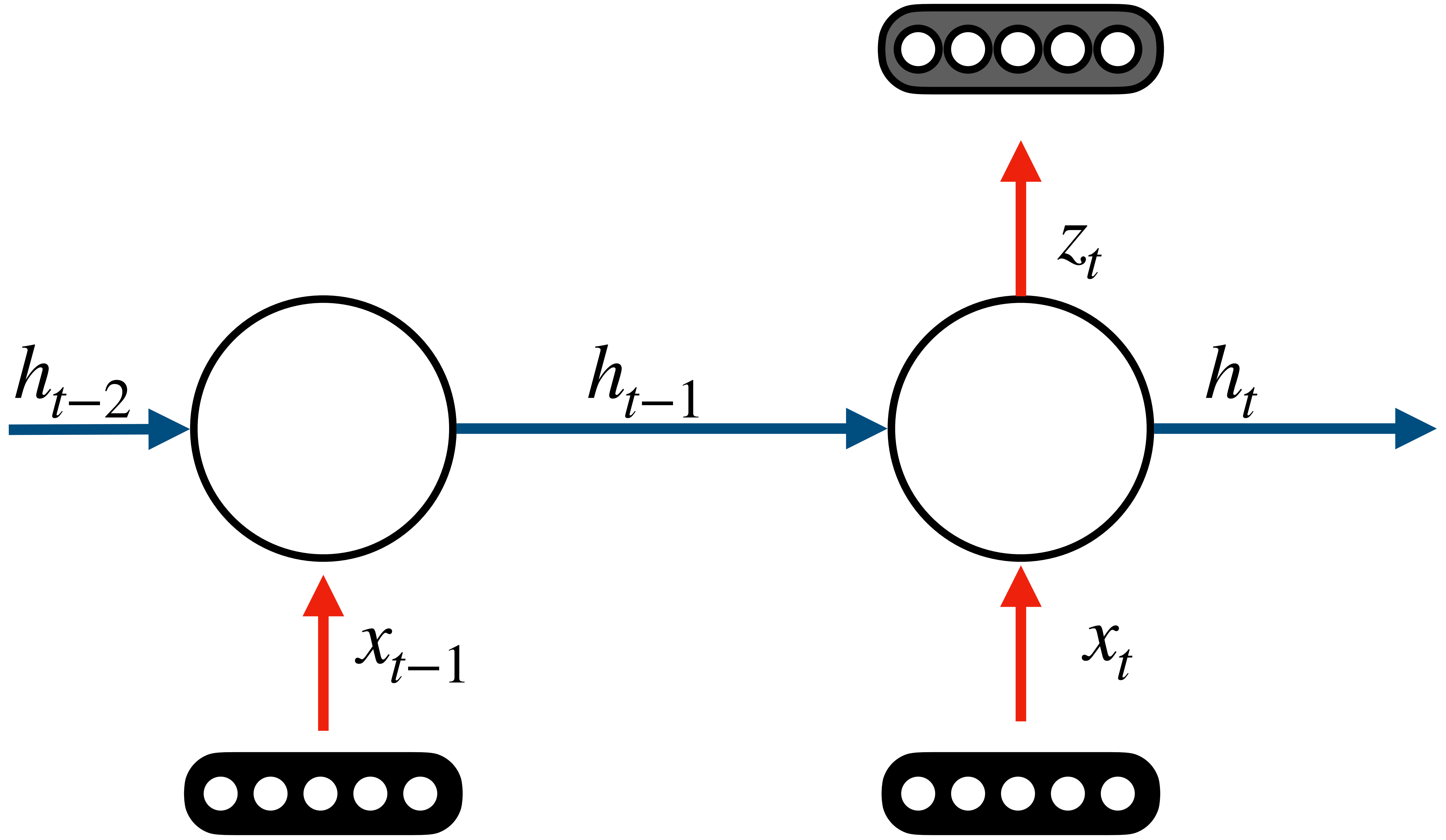
- RNN can be unrolled to a feedforward neural network
- Depth of feedforward neural network depends on length of the sequence



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

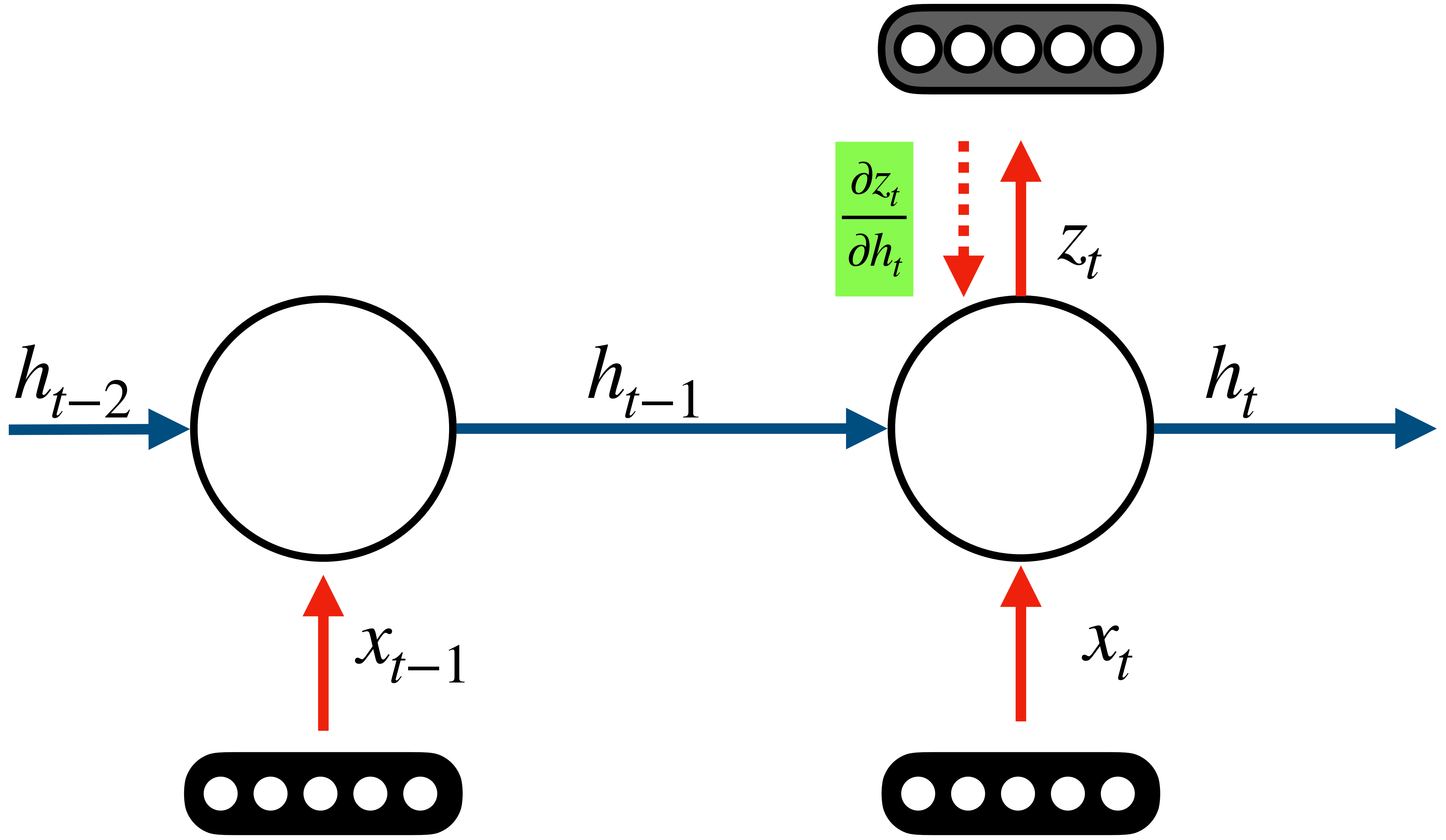
$$z_t = \sigma(W_{zh}h_t + b_z)$$

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

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

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

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



Backpropagation through Time

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

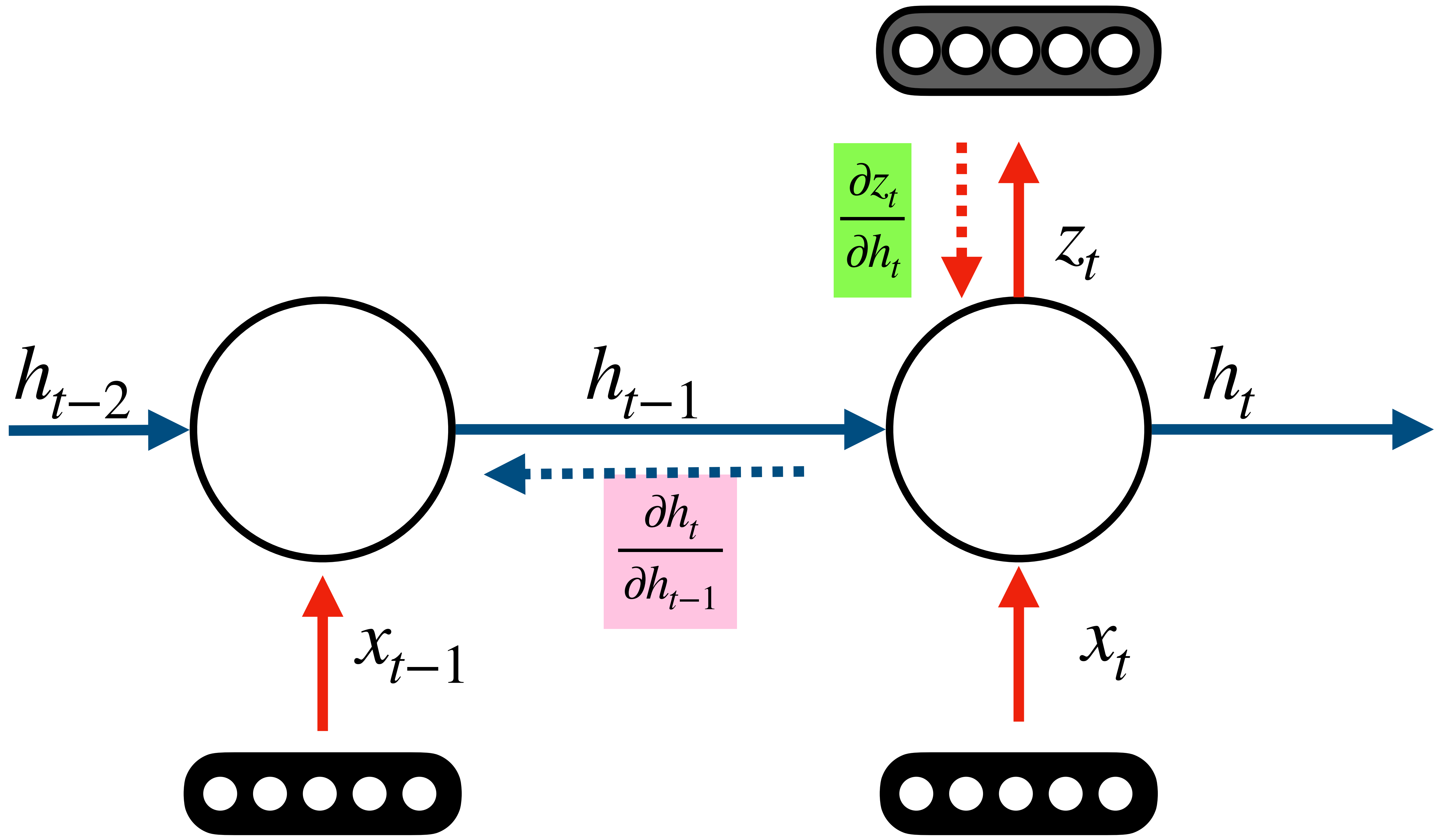
$$h_t = \sigma\left(W_{hx} x_t + W_{hh} h_{t-1} + b_h\right)$$

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

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

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

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


$\frac{\partial z_t}{\partial h_t}$
z_t
h_{t-2}
h_{t-1}
h_t
$\frac{\partial h_t}{\partial h_{t-1}}$
x_{t-1}
x_t

Backpropagation through Time

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

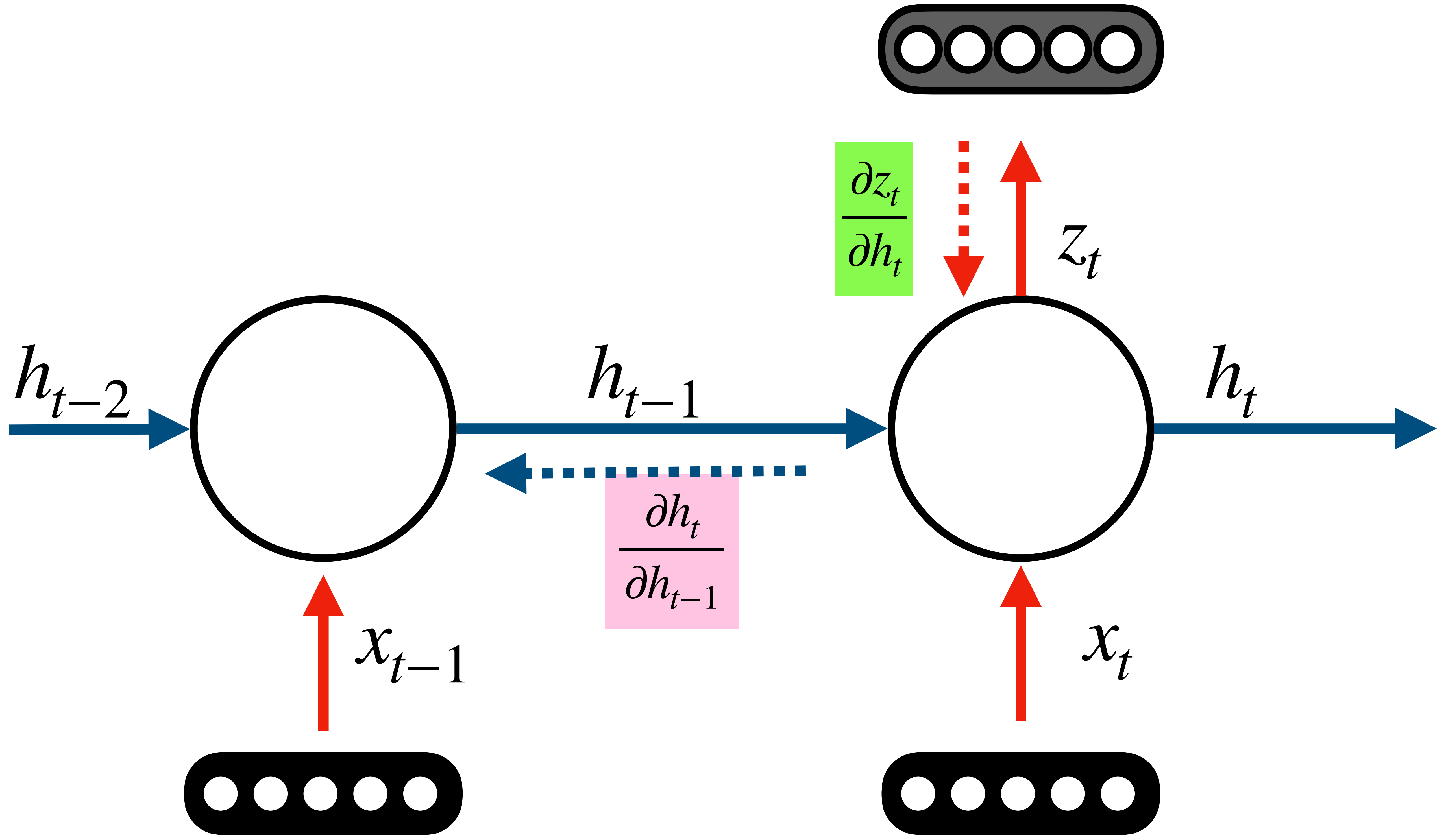
$$h_t = \sigma\left(W_{hx}x_t + W_{hh}h_{t-1} + b_h\right)$$

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

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

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

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


$\frac{\partial z_t}{\partial h_t}$
z_t
h_{t-2}
h_{t-1}
h_t
$\frac{\partial h_t}{\partial h_{t-1}}$
x_{t-1}
x_t

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

Backpropagation through Time

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

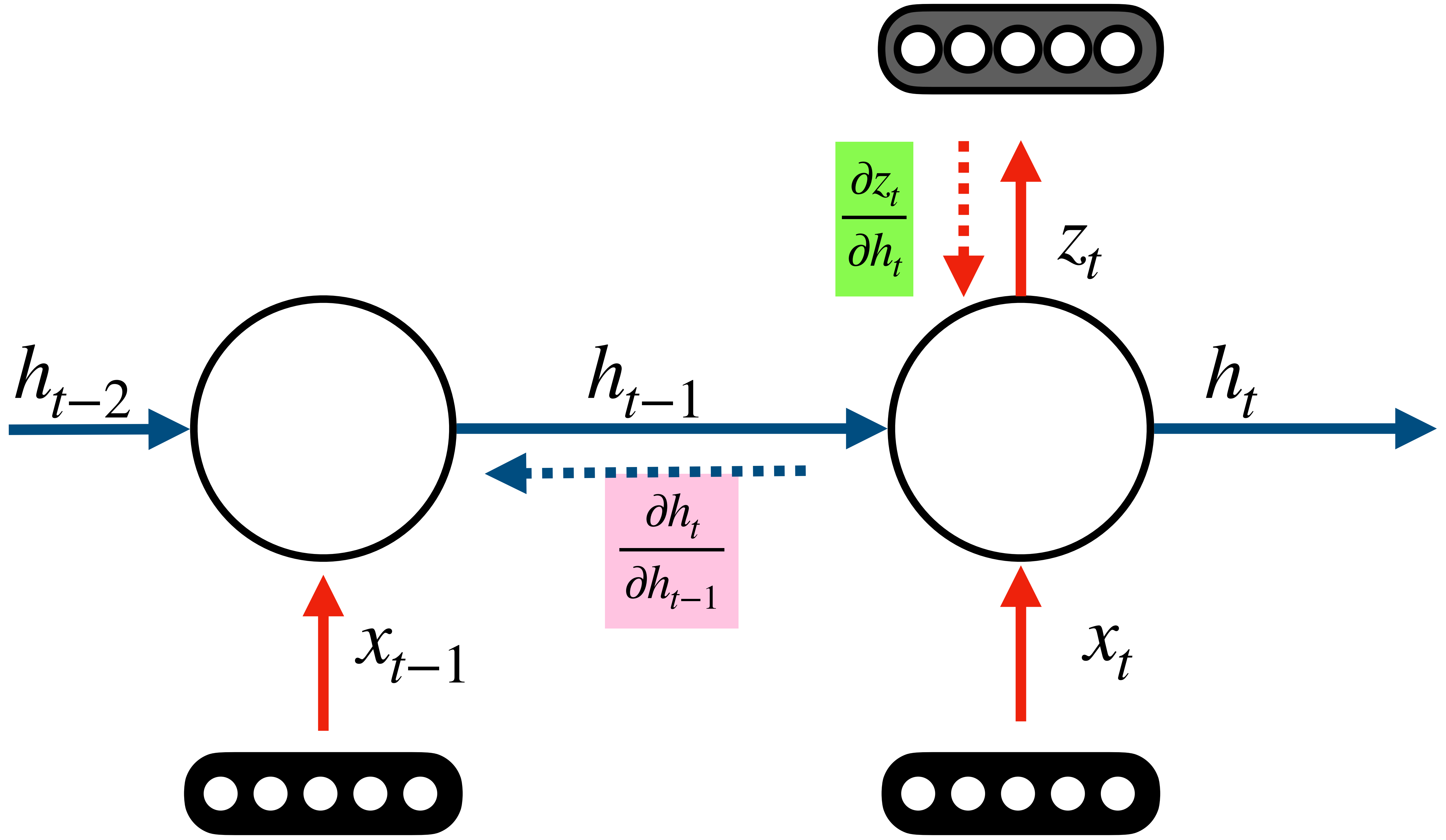
$$h_t = \sigma\left(W_{hx}x_t + W_{hh}h_{t-1} + b_h\right)$$

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

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

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

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



$\frac{\partial z_t}{\partial h_t}$
z_t
h_{t-2}
h_{t-1}
h_t
$\frac{\partial h_t}{\partial h_{t-1}}$
x_{t-1}
x_t

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

Backpropagation through Time

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

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

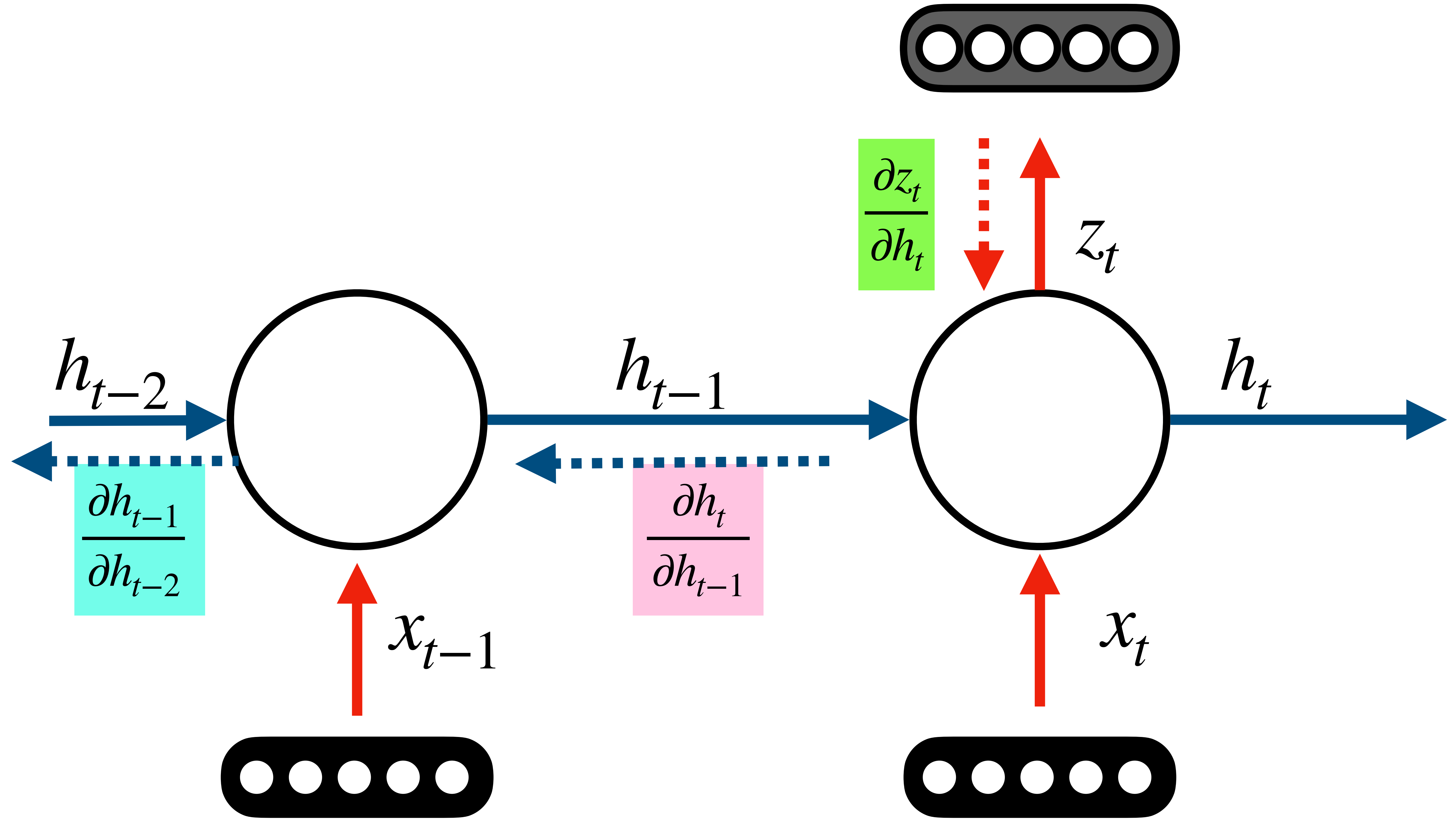
$$v_t = W_{zh}h_t + b_z \qquad z_t = \sigma(v_t)$$

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

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

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

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



$\frac{\partial z_t}{\partial h_t}$
z_t
h_{t-2}
h_{t-1}
h_t
$\frac{\partial h_{t-1}}{\partial h_{t-2}}$
$\frac{\partial h_t}{\partial h_{t-1}}$
x_{t-1}
x_t

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

Backpropagation through Time

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

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

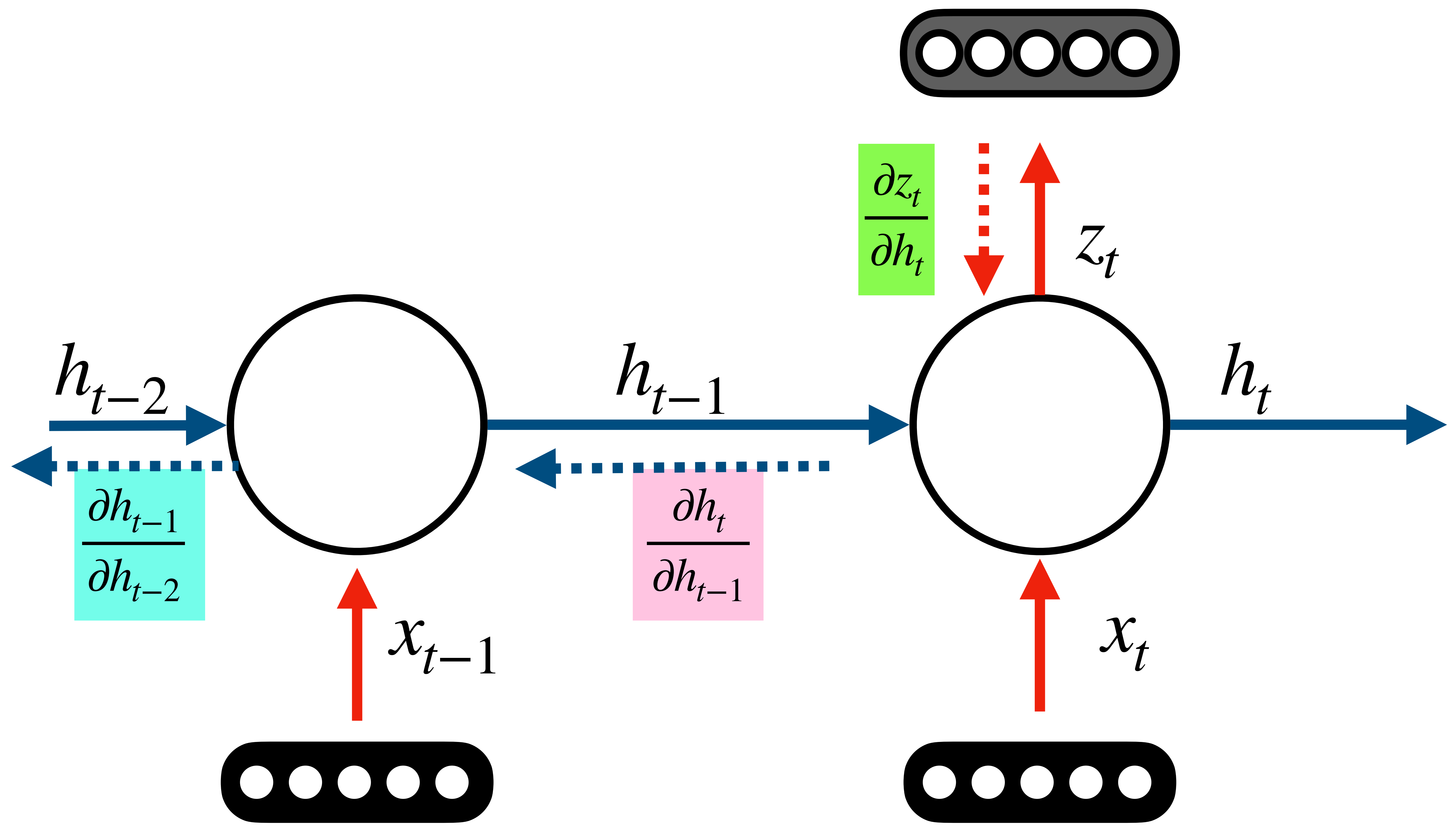
$$v_t = W_{zh}h_t + b_z \qquad z_t = \sigma(v_t)$$

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

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

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

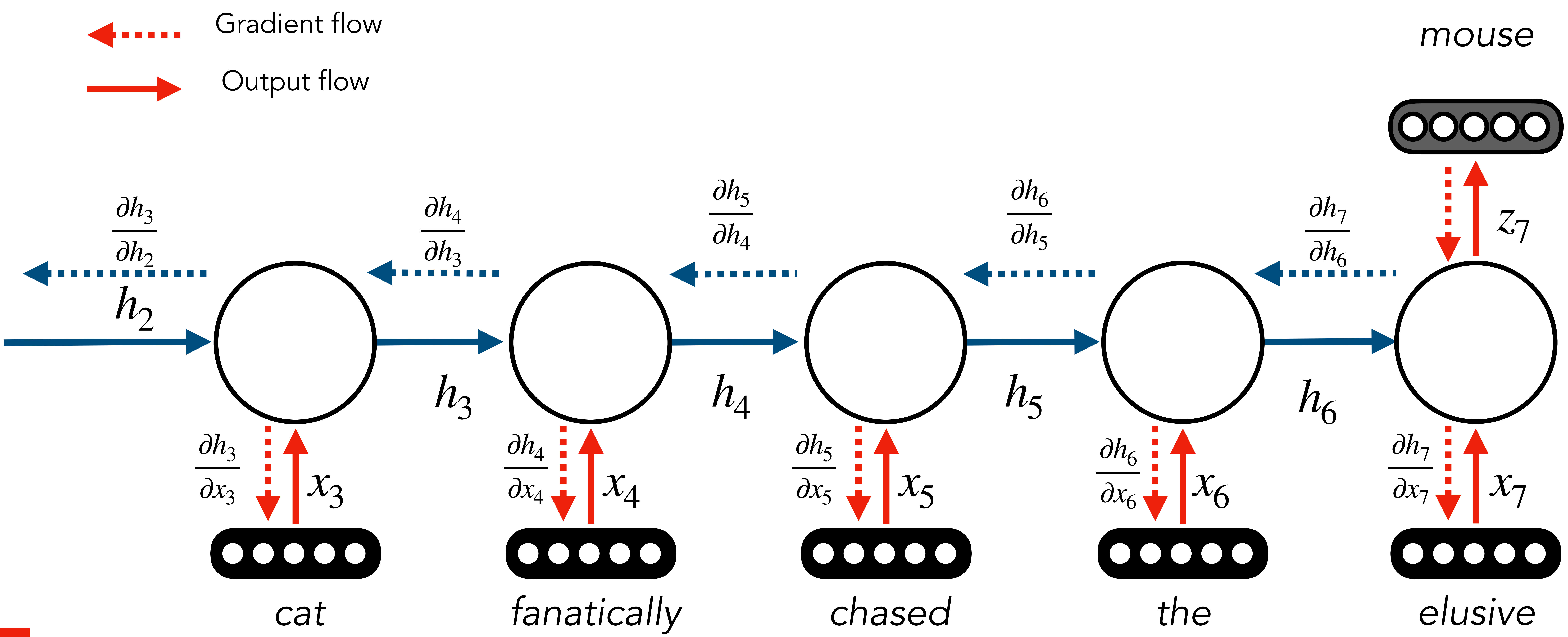
$$\frac{\partial h_{t-1}}{\partial h_{t-2}} = \frac{\partial \sigma(u_{t-1})}{\partial u_{t-1}}\frac{\partial u_{t-1}}{\partial h_{t-2}} = \frac{\partial \sigma(u_{t-1})}{\partial u_{t-1}}W_{hh}$$


$\frac{\partial z_t}{\partial h_t}$
z_t
h_{t-2}
h_{t-1}
h_t
$\frac{\partial h_{t-1}}{\partial h_{t-2}}$
$\frac{\partial h_t}{\partial h_{t-1}}$
x_{t-1}
x_t

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

Note that these are
actually the same matrix

Backpropagation through time



Gradient flow
Output flow
mouse
z_7
$\frac{\partial h_3}{\partial h_2}$
$\frac{\partial h_4}{\partial h_3}$
$\frac{\partial h_5}{\partial h_4}$
$\frac{\partial h_6}{\partial h_5}$
$\frac{\partial h_7}{\partial h_6}$
h_2
h_3
h_4
h_5
h_6
$\frac{\partial h_3}{\partial x_3}$
x_3
$\frac{\partial h_4}{\partial x_4}$
x_4
$\frac{\partial h_5}{\partial x_5}$
x_5
$\frac{\partial h_6}{\partial x_6}$
x_6
$\frac{\partial h_7}{\partial x_7}$
x_7
cat
fanatically
chased
the
elusive

RNNs 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! You won't have to compute gradients by hand :)

Multiple Layers

z_3^2
z_4^2
z_5^2
z_6^2
z_7^2
h_2^2
h_3^2
h_4^2
h_5^2
h_6^2
h_7^2
z_3^1
z_4^1
z_5^1
z_6^1
z_7^1
h_2^1
h_3^1
h_4^1
h_5^1
h_6^1
h_7^1
x_3
x_4
x_5
x_6
x_7
cat
fanatically
chased
the
elusive

z_3^3
z_4^3
z_5^3
z_6^3
z_7^3
h_2^3
h_3^3
h_4^3
h_5^3
h_6^3
h_7^3
z_3^2
z_4^2
z_5^2
z_6^2
z_7^2
h_2^2
h_3^2
h_4^2
h_5^2
h_6^2
h_7^2
z_3^1
z_4^1
z_5^1
z_6^1
z_7^1
h_2^1
h_3^1
h_4^1
h_5^1
h_6^1
h_7^1
x_3
x_4
x_5
x_6
x_7
cat
fanatically
chased
the
elusive

z_3^3
z_4^3
z_5^3
z_6^3
z_7^3
h_2^3
h_3^3
h_4^3
h_5^3
h_6^3
h_7^3
z_3^2
z_4^2
z_5^2
z_6^2
z_7^2
h_2^2
h_3^2
h_4^2
h_5^2
h_6^2
h_7^2
z_3^1
z_4^1
z_5^1
z_6^1
z_7^1
h_2^1
h_3^1
h_4^1
h_5^1
h_6^1
h_7^1
x_3
x_4
x_5
x_6
x_7
cat
fanatically
chased
the
elusive

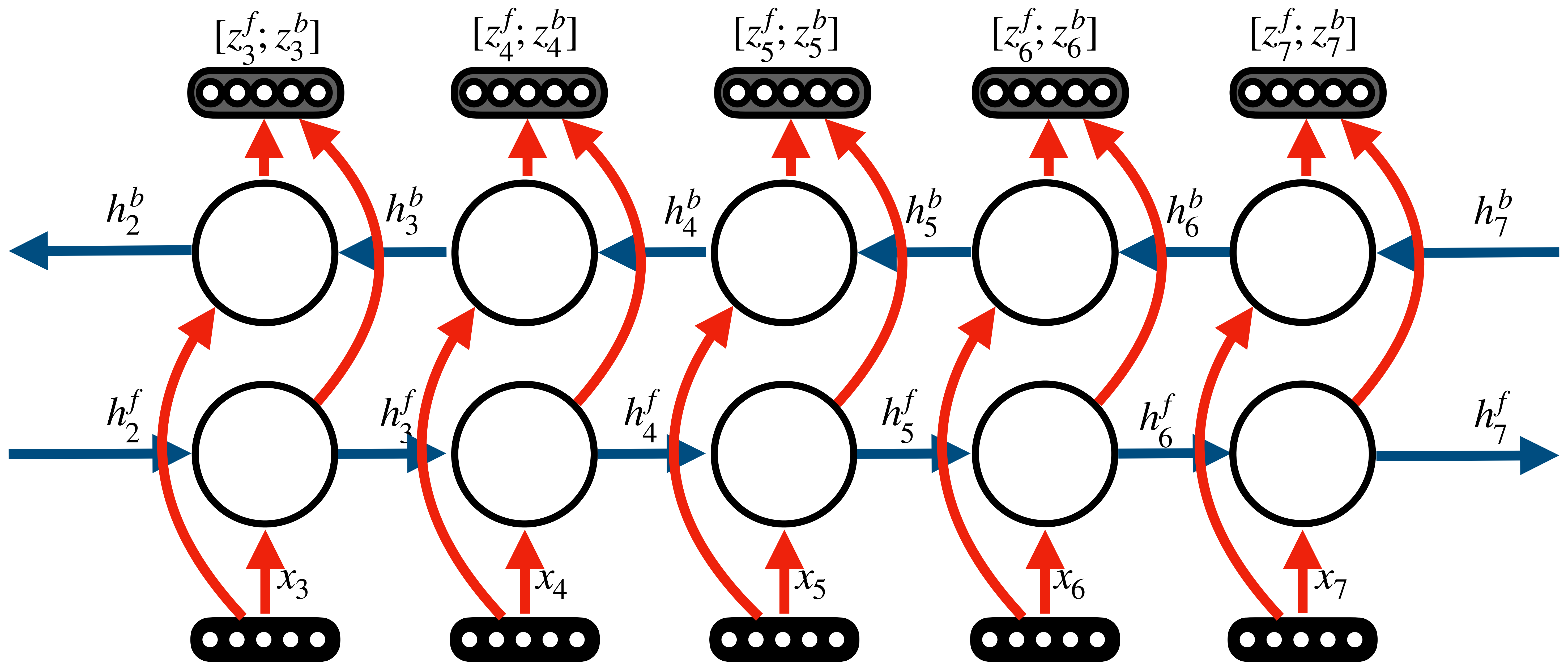
...
z_3^3
z_4^3
z_5^3
z_6^3
z_7^3
h_2^3
h_3^3
h_4^3
h_5^3
h_6^3
h_7^3
z_3^2
z_4^2
z_5^2
z_6^2
z_7^2
h_2^2
h_7^2
z_3^1
z_4^1
z_5^1
z_6^1
z_7^1
h_2^1
h_3^1
h_4^1
h_5^1
h_6^1
h_7^1
x_3
x_4
x_5
x_6
x_7
cat
fanatically
chased
the
elusive

In practice, typically use 3-4 layers of RNN cells

Bidirectionality

$[z_3^f; z_3^b]$
$[z_4^f; z_4^b]$
$[z_5^f; z_5^b]$
$[z_6^f; z_6^b]$
$[z_7^f; z_7^b]$
h_2^b
h_3^b
h_4^b
h_5^b
h_6^b
h_7^b
h_2^f
h_3^f
h_4^f
h_5^f
h_6^f
h_7^f
x_3
x_4
x_5
x_6
x_7
cat
fanatically
chased
the
elusive

Bidirectionality



- **Concatenate** the output states for final representation at each step
    - Can also do mean / max
- Separate parameters for forward and backward RNNs
- If you can use the future text for the task, then you should use **bidirectionality**

Question

For which of the following task types can we use a bidirectional RNN?

(a) Classification
(b) Sequence labelling
(c) Text Generation

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**

Issue with Recurrent Models

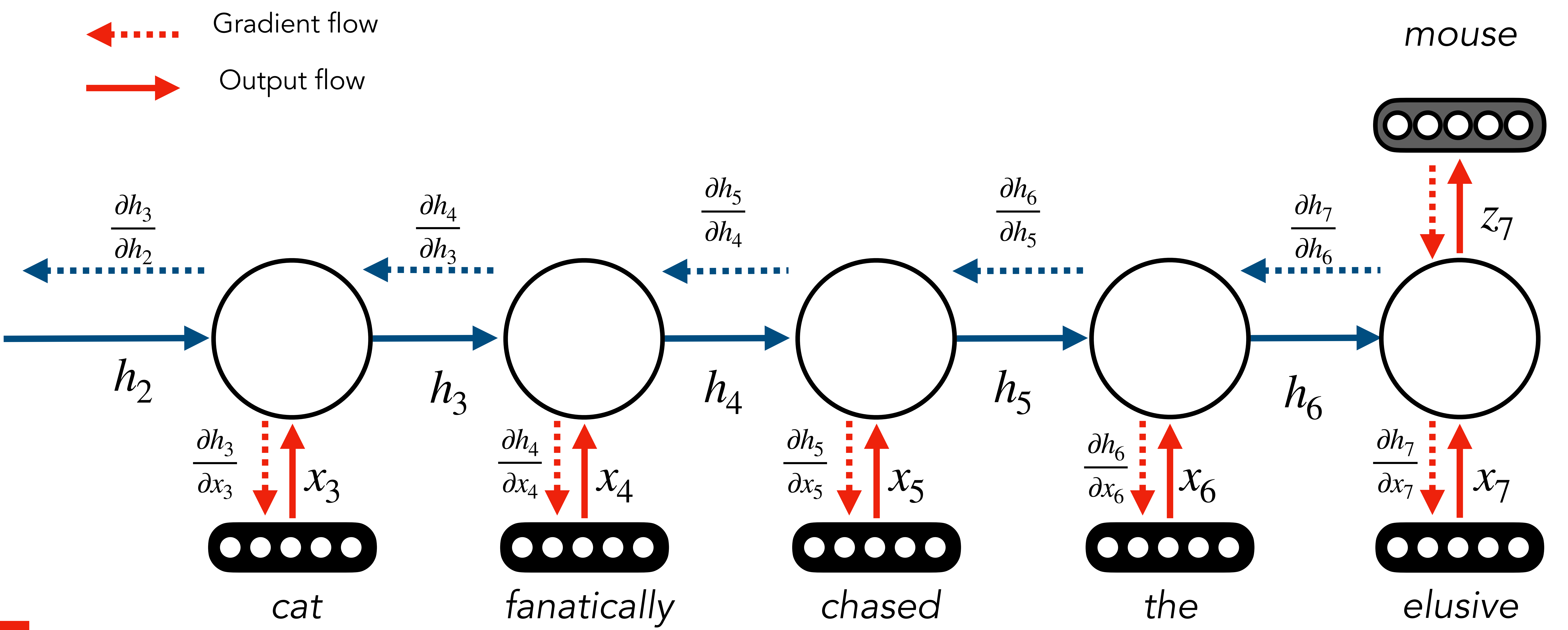
- Multiple steps of state overwriting makes it challenging to learn long-range dependencies.

They tuned, discussed for a moment, then struck up a lively ***jig****. Everyone joined in, turning the courtyard into an even more chaotic scene, people now* ***dancing*** *in circles,* ***swinging*** *and* ***spinning*** *in circles, everyone making up their own* ***dance steps****. I felt my feet tapping, my body wanting to move. Aside from writing, I 've always loved* <u>**dancing**</u> *.*

- Nearby words should affect each other more than farther ones, but RNNs make it challenging to learn <u>**any**</u> long-range interactions

LAMBADA dataset, 2016

Backpropagation through time



Gradient flow
Output flow
mouse
z_7
$\frac{\partial h_3}{\partial h_2}$
$\frac{\partial h_4}{\partial h_3}$
$\frac{\partial h_5}{\partial h_4}$
$\frac{\partial h_6}{\partial h_5}$
$\frac{\partial h_7}{\partial h_6}$
h_2
h_3
h_4
h_5
h_6
$\frac{\partial h_3}{\partial x_3}$
x_3
$\frac{\partial h_4}{\partial x_4}$
x_4
$\frac{\partial h_5}{\partial x_5}$
x_5
$\frac{\partial h_6}{\partial x_6}$
x_6
$\frac{\partial h_7}{\partial x_7}$
x_7
cat
fanatically
chased
the
elusive

Vanishing Gradients

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

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

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

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

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

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

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

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

Generalising this:

$$\frac{\partial h_t}{\partial h_{t-T}} = \prod_{i=t-T}^{i=t}\frac{\partial h_i}{\partial h_{i-1}} = \prod_{i=t-T}^{i=t}\frac{\partial \sigma(u_i)}{\partial u_i}W_{hh}$$

Vanishing Gradients

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

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

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

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

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

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

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

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

Generalising this:

$$\frac{\partial h_t}{\partial h_{t-T}} = \prod_{i=t-T}^{i=t}\frac{\partial h_i}{\partial h_{i-1}} = \prod_{i=t-T}^{i=t}\frac{\partial \sigma(u_i)}{\partial u_i}W_{hh}$$

< 1 for many activation fxns

Typically small (Regularisation)

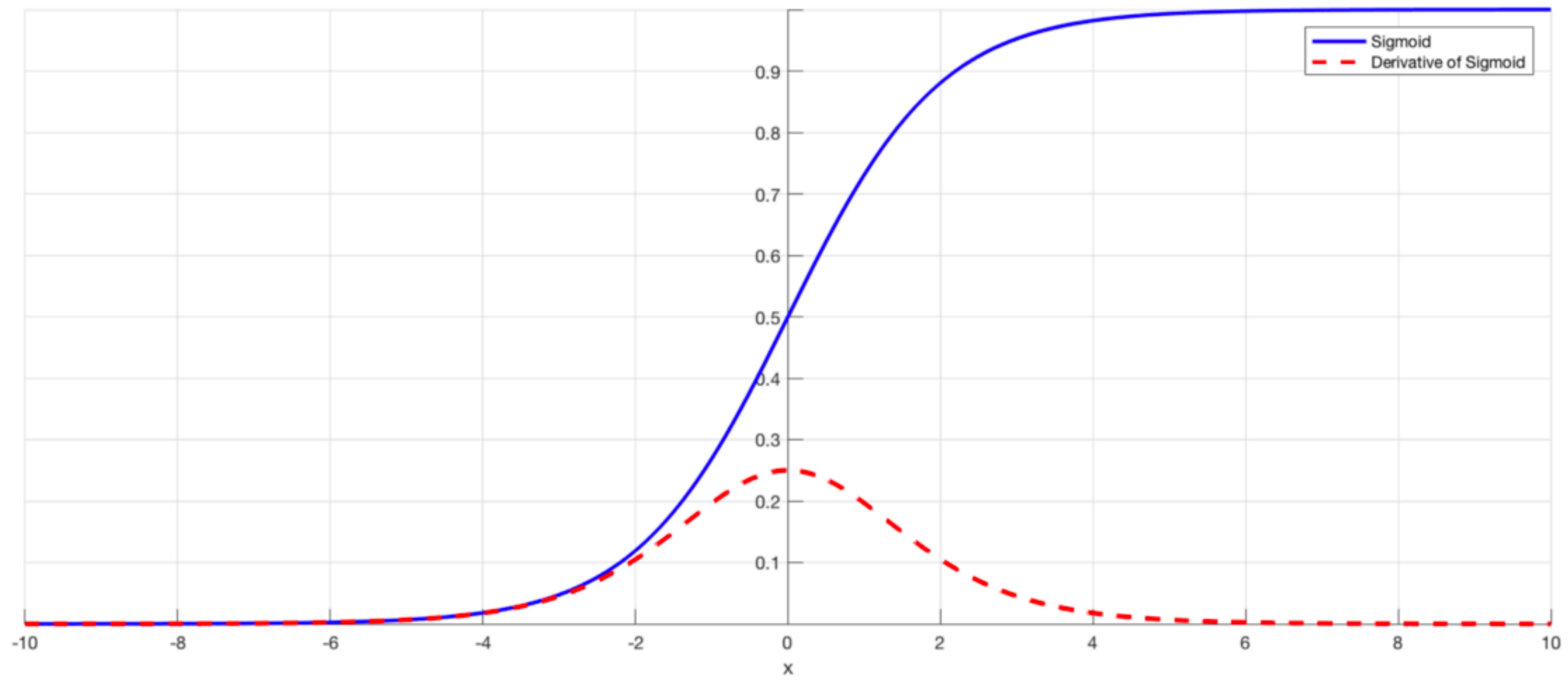
Vanishing Gradients

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

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

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

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

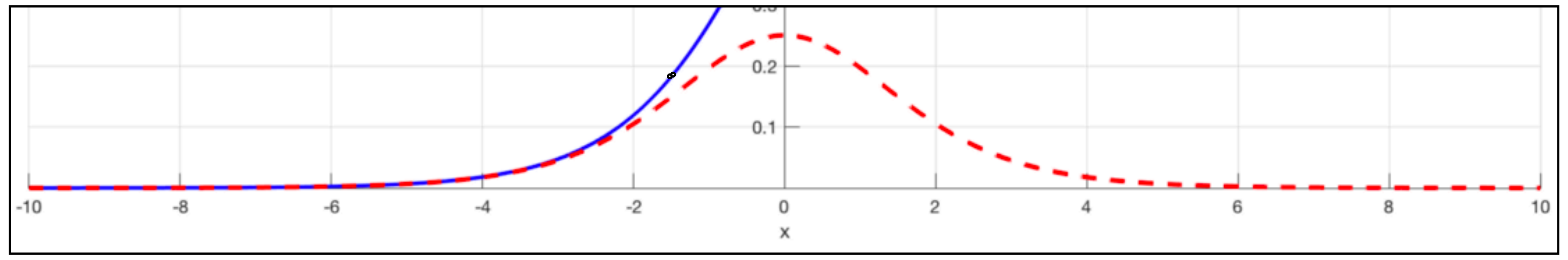


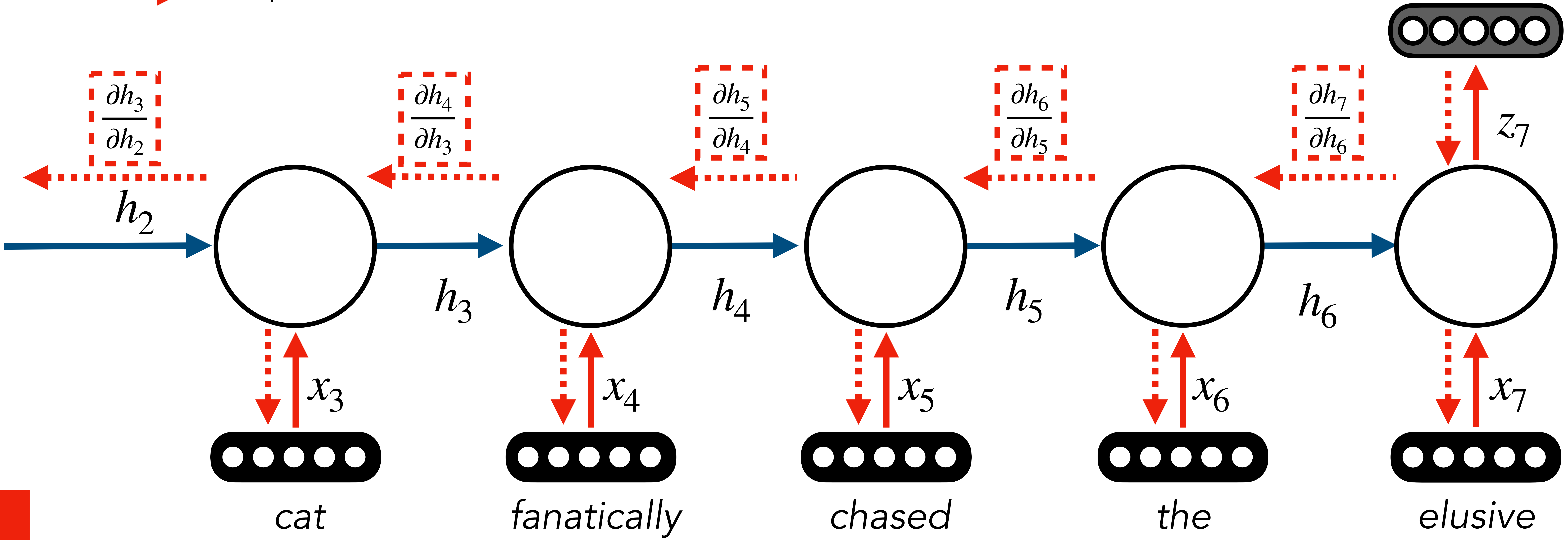
Vanishing Gradients

Gradient flow

Output flow

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

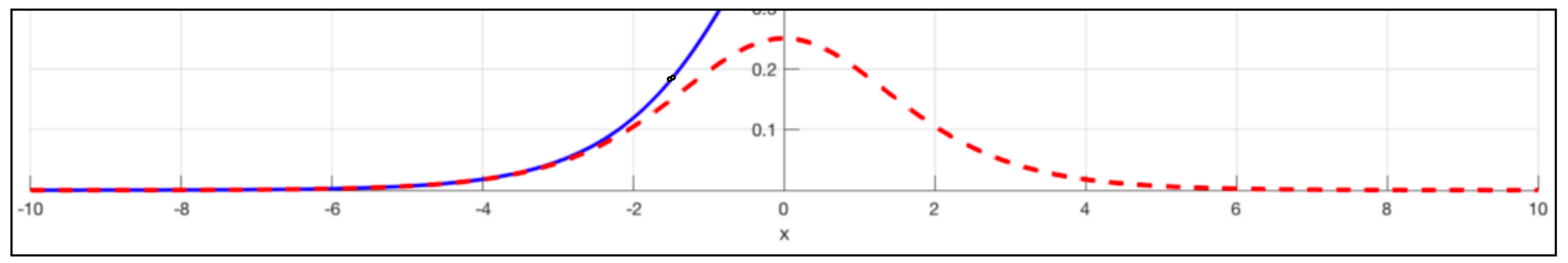


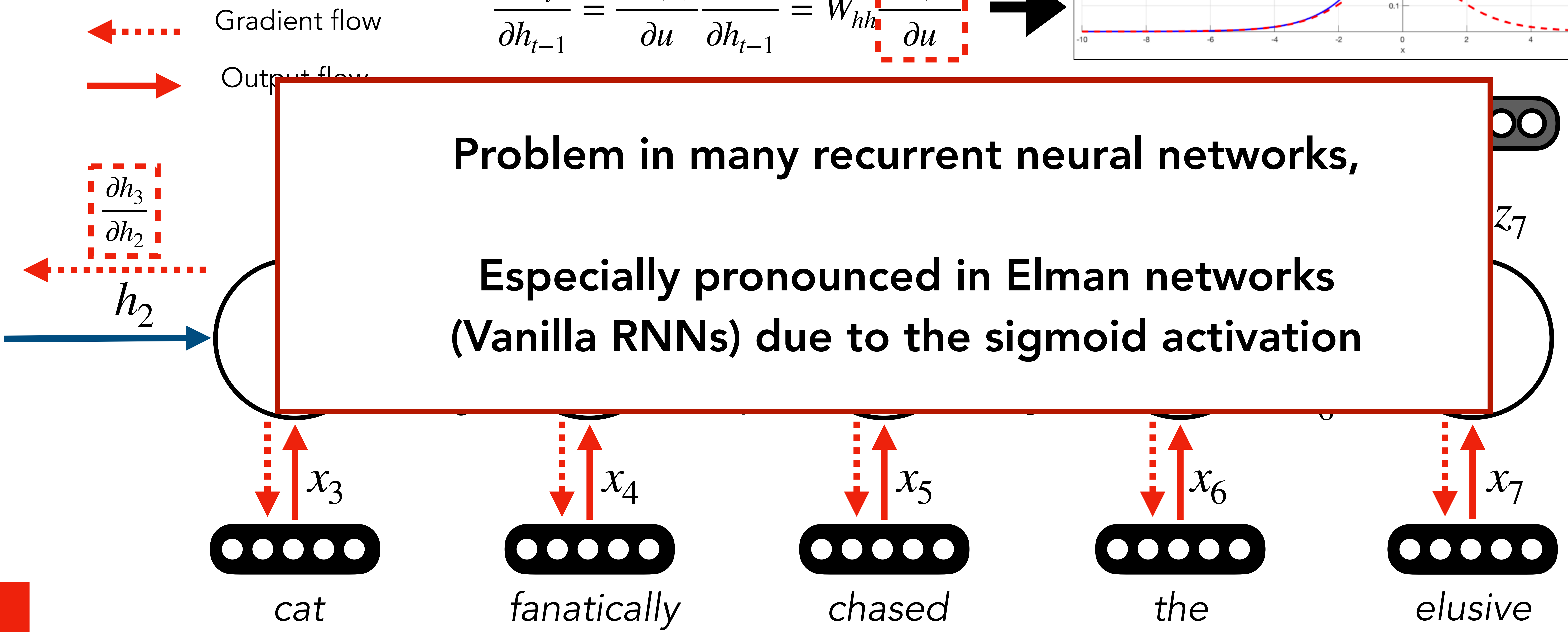


$\frac{\partial h_3}{\partial h_2}$
$\frac{\partial h_4}{\partial h_3}$
$\frac{\partial h_5}{\partial h_4}$
$\frac{\partial h_6}{\partial h_5}$
$\frac{\partial h_7}{\partial h_6}$
z_7
h_2
h_3
h_4
h_5
h_6
x_3
x_4
x_5
x_6
x_7
cat
fanatically
chased
the
elusive

Vanishing Gradients

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





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?

$$\frac{\partial h_t}{\partial h_{t-T}} = \prod_{i=t-T}^{i=t} \frac{\partial h_i}{\partial h_{i-1}} = \prod_{i=t-T}^{i=t} \underbrace{\frac{\partial \sigma(u_i)}{\partial u_i}}_{\text{Typically less than one}} \underbrace{W_{hh}}_{\text{Typically small (Regularisation)}}$$

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**

Gated Recurrent Neural Networks

- Use gates to avoid dampening gradient signal every time step

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

Elman Network

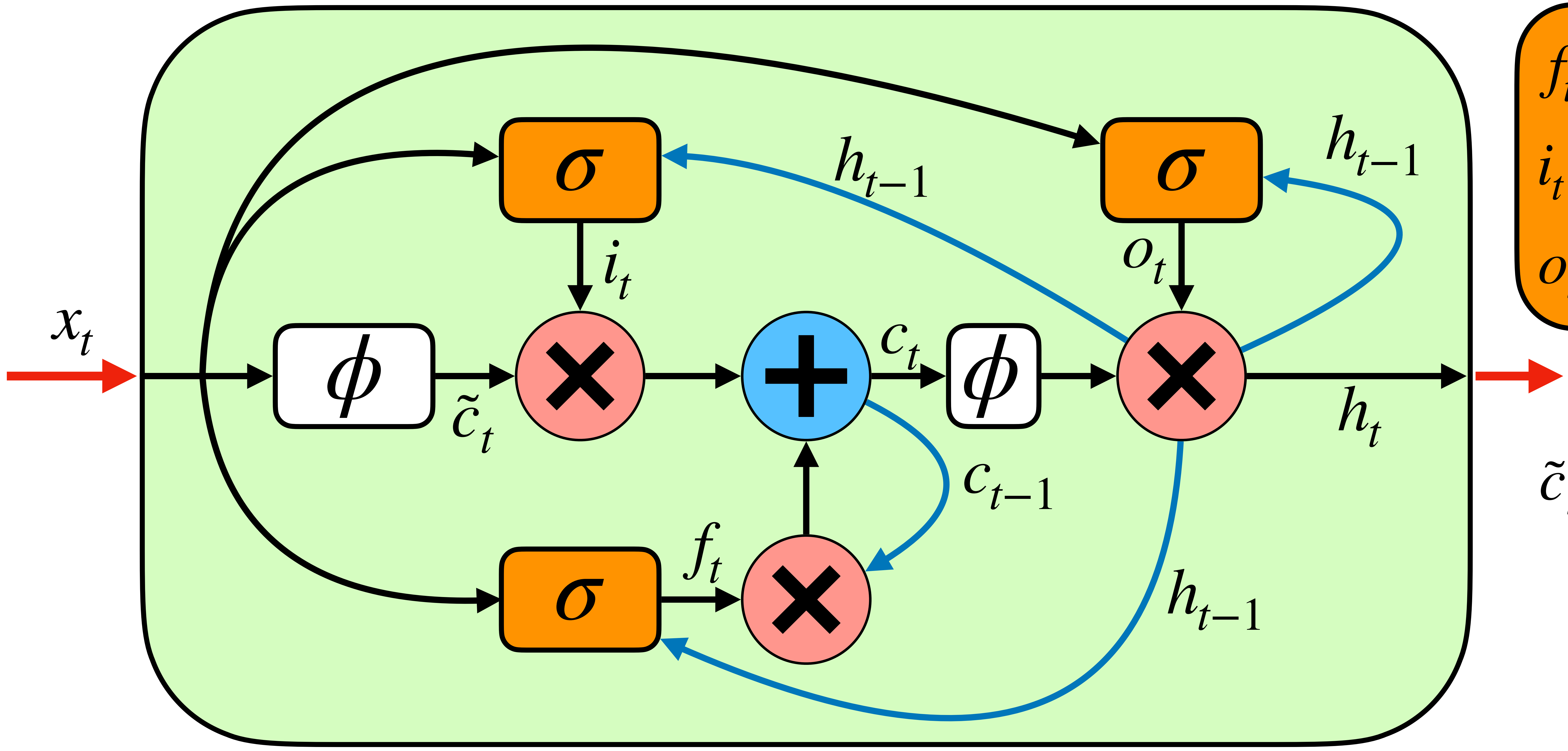
$$h_t = h_{t-1} \odot \mathbf{f} + \mathbf{func}(x_t)$$

Gated Network Abstraction

- Gate value **f** computes how much information from previous hidden state moves to the next time step —> 0 < **f** < 1

- Because h_{t-1} is no longer inside the activation function, it is not automatically constrained, reducing vanishing gradients!

Long Short Term Memory (LSTM)



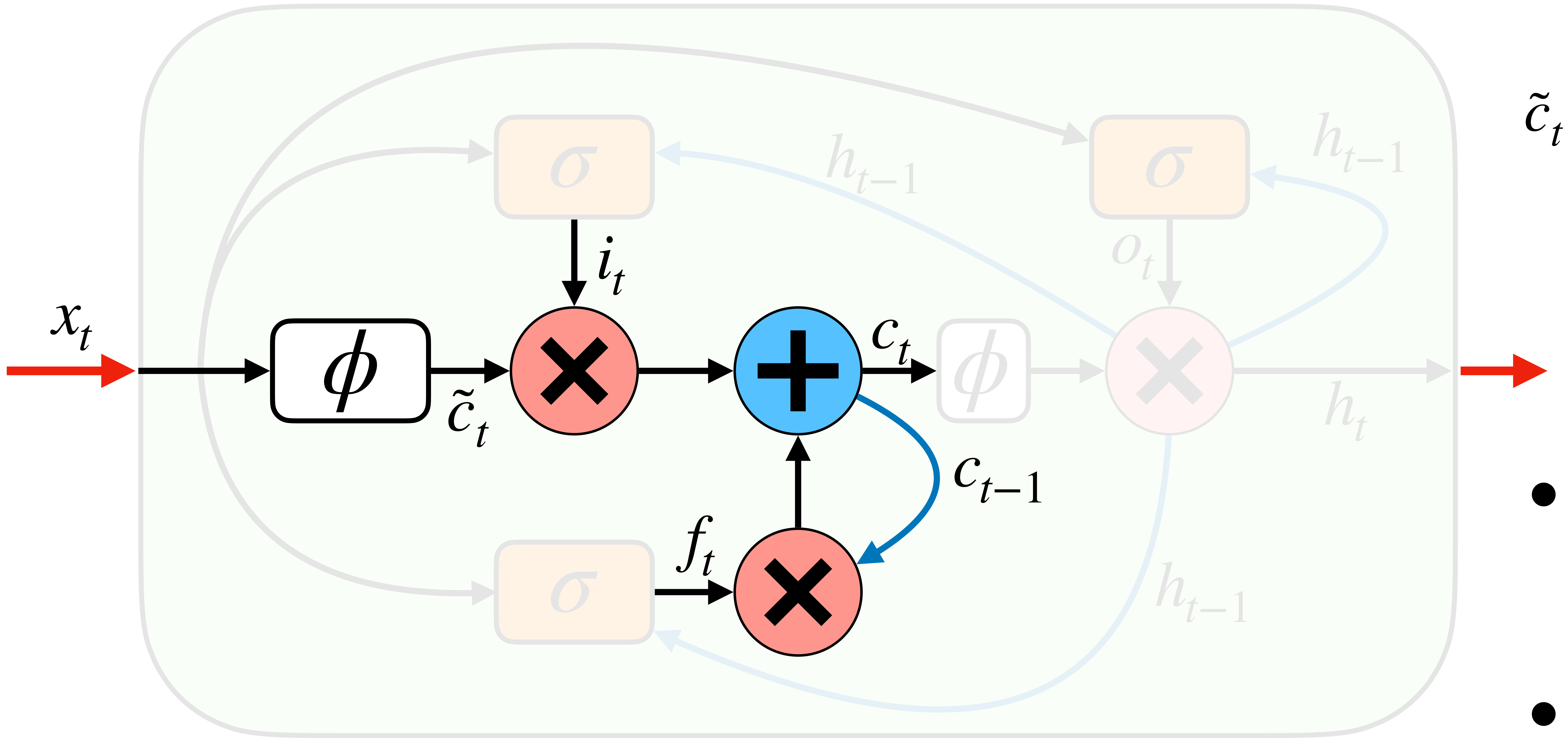
Gates:

$$f_t = \sigma\left(W_{fx}x_t + W_{fh}h_{t-1} + b_f\right)$$
$$i_t = \sigma\left(W_{ix}x_t + W_{ih}h_{t-1} + b_i\right)$$
$$o_t = \sigma\left(W_{ox}x_t + W_{oh}h_{t-1} + b_o\right)$$

$$\tilde{c}_t = \phi\left(W_{cx}x_t + W_{ch}h_{t-1} + b_c\right)$$
$$c_t = i_t \times \tilde{c}_t + f_t \times c_{t-1}$$
$$h_t = o_t \times \phi(c_t)$$

(Hochreiter and Schmidhuber, 1997)

Cell State



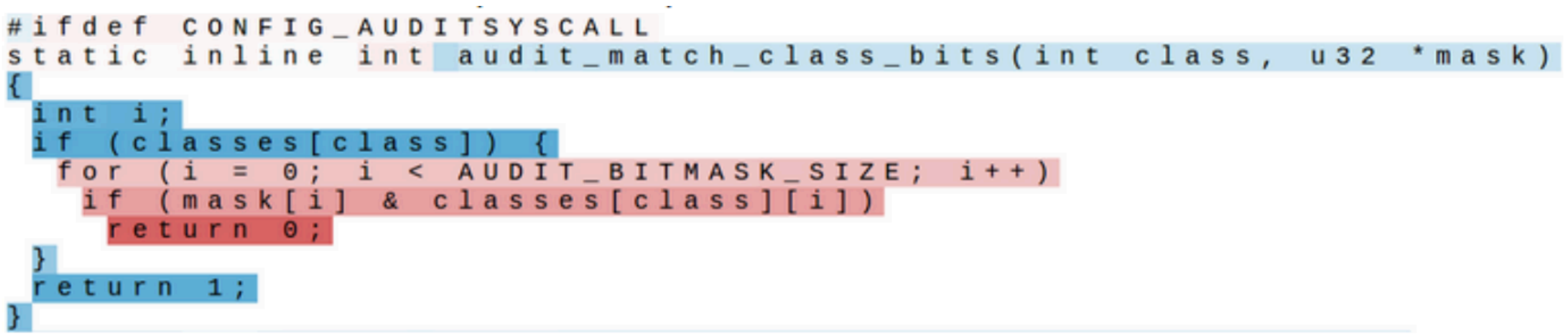
$$\tilde{c}_t = \phi\left(W_{cx}x_t + W_{ch}h_{t-1} + b_c\right)$$

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

- Hidden state h_{t-1} is now short-term memory
- Cell state c_t tracks longer-term dependencies

What does the cell state track?

- Can visualise the activations of cell state (i.e., dimensions of **c**) and find semantic behaviour!
- Stack Overflow example:

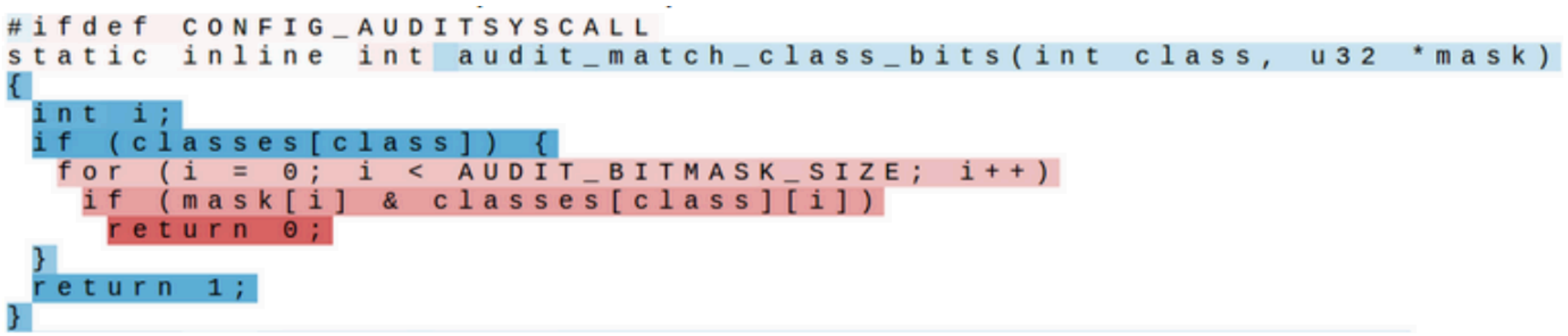


```
#ifdef CONFIG_AUDITSYSCALL
static inline int audit_match_class_bits(int class, u32 *mask)
{
 int i;
 if (classes[class]) {
  for (i = 0; i < AUDIT_BITMASK_SIZE; i++)
   if (mask[i] & classes[class][i])
    return 0;
 }
 return 1;
}
```

Karpathy et al. (2015)

What does the cell state track?

- Can visualise the activations of cell state (i.e., dimensions of **c**) and find semantic behaviour!
- Stack Overflow example: **track indentation**

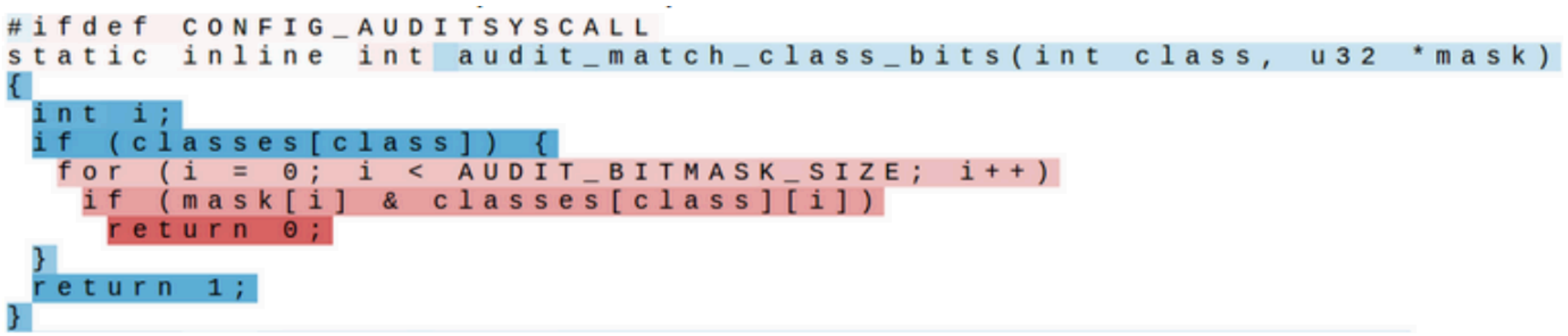


```
#ifdef CONFIG_AUDITSYSCALL
static inline int audit_match_class_bits(int class, u32 *mask)
{
 int i;
 if (classes[class]) {
  for (i = 0; i < AUDIT_BITMASK_SIZE; i++)
   if (mask[i] & classes[class][i])
    return 0;
 }
 return 1;
}
```

Karpathy et al. (2015)

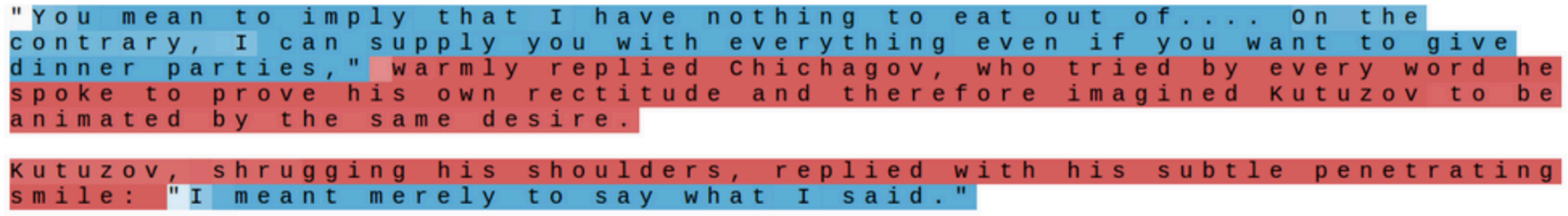
What does the cell state track?

- Can visualise the activations of cell state (i.e., dimensions of **c**) and find semantic behaviour!
- Stack Overflow example: track indentation



```
#ifdef CONFIG_AUDITSYSCALL
static inline int audit_match_class_bits(int class, u32 *mask)
{
 int i;
 if (classes[class]) {
  for (i = 0; i < AUDIT_BITMASK_SIZE; i++)
   if (mask[i] & classes[class][i])
    return 0;
 }
 return 1;
}
```

- War and Peace:



"You mean to imply that I have nothing to eat out of.... On the contrary, I can supply you with everything even if you want to give dinner parties," warmly replied Chichagov, who tried by every word he spoke to prove his own rectitude and therefore imagined Kutuzov to be animated by the same desire.

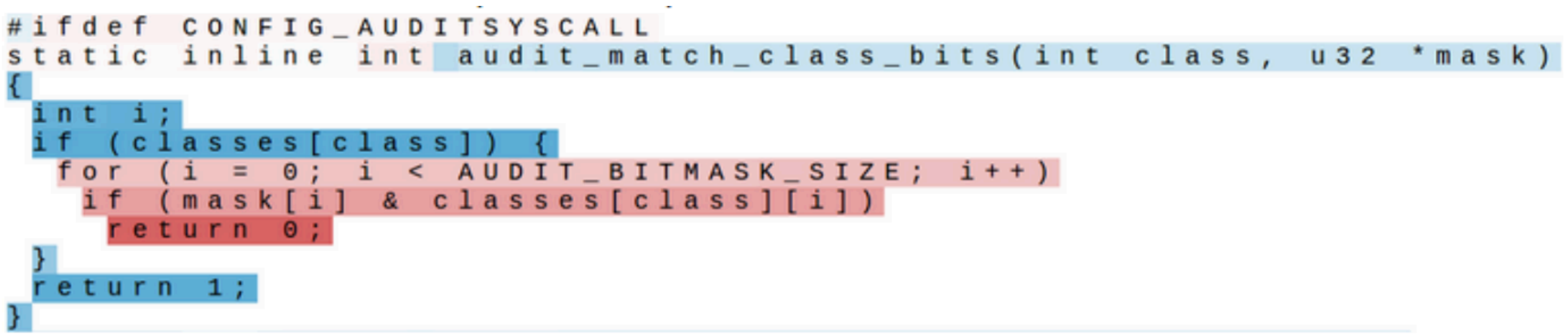
Kutuzov, shrugging his shoulders, replied with his subtle penetrating smile: "I meant merely to say what I said."

Karpathy et al. (2015)

What does the cell state track?

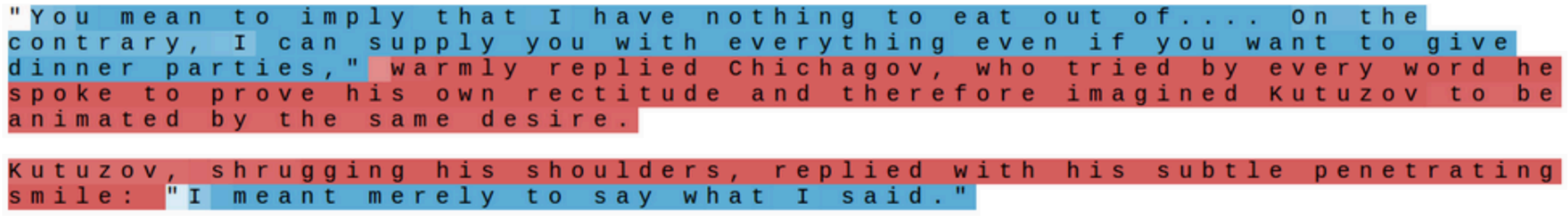
- Can visualise the activations of cell state (i.e., dimensions of **c**) and find semantic behaviour!

- Stack Overflow example: track indentation



```
#ifdef CONFIG_AUDITSYSCALL
static inline int audit_match_class_bits(int class, u32 *mask)
{
 int i;
 if (classes[class]) {
  for (i = 0; i < AUDIT_BITMASK_SIZE; i++)
   if (mask[i] & classes[class][i])
    return 0;
 }
 return 1;
}
```

- War and Peace: **are we in a quote or not?**

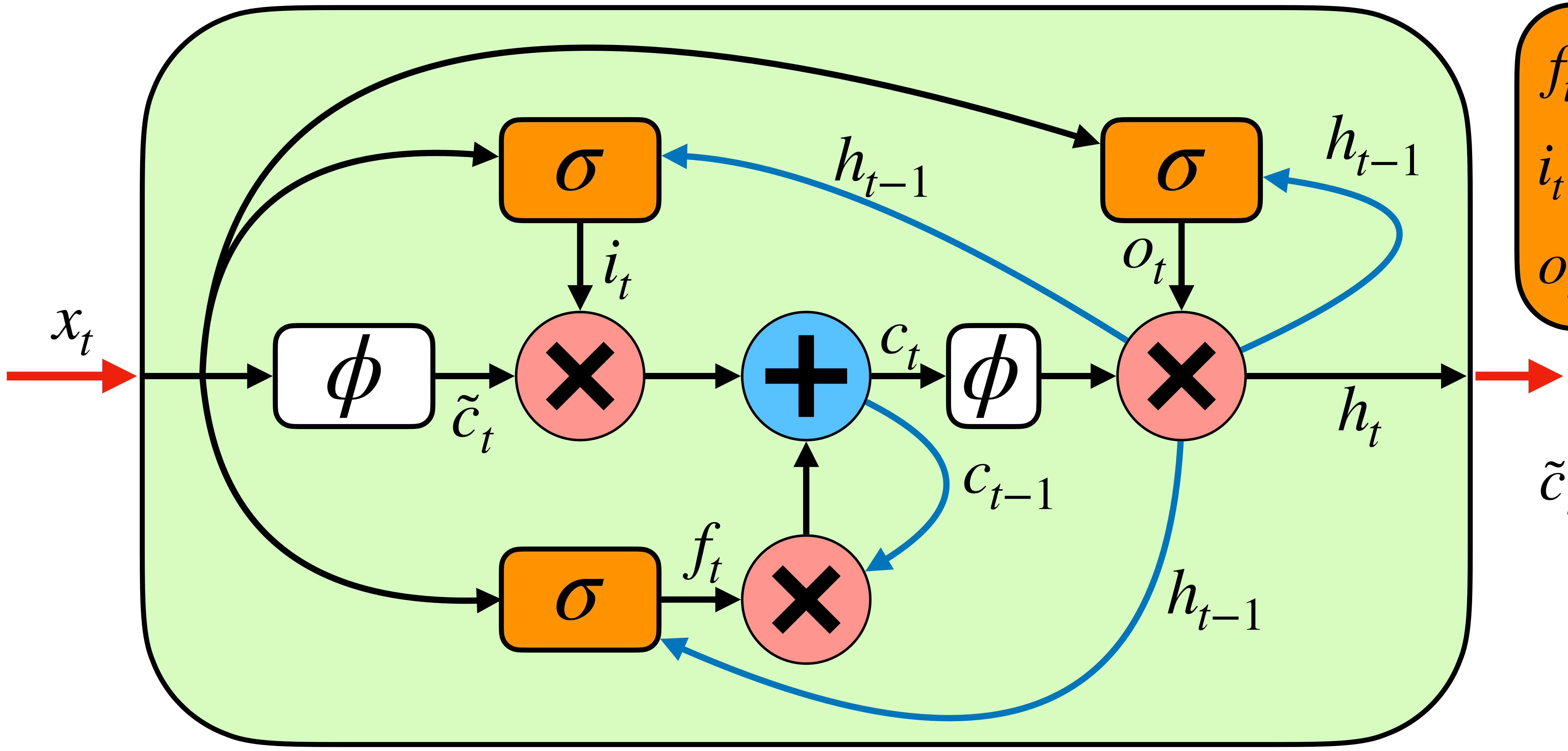


"You mean to imply that I have nothing to eat out of.... On the contrary, I can supply you with everything even if you want to give dinner parties," warmly replied Chichagov, who tried by every word he spoke to prove his own rectitude and therefore imagined Kutuzov to be animated by the same desire.

Kutuzov, shrugging his shoulders, replied with his subtle penetrating smile: "I meant merely to say what I said."

Karpathy et al. (2015)

Long Short Term Memory (LSTM)



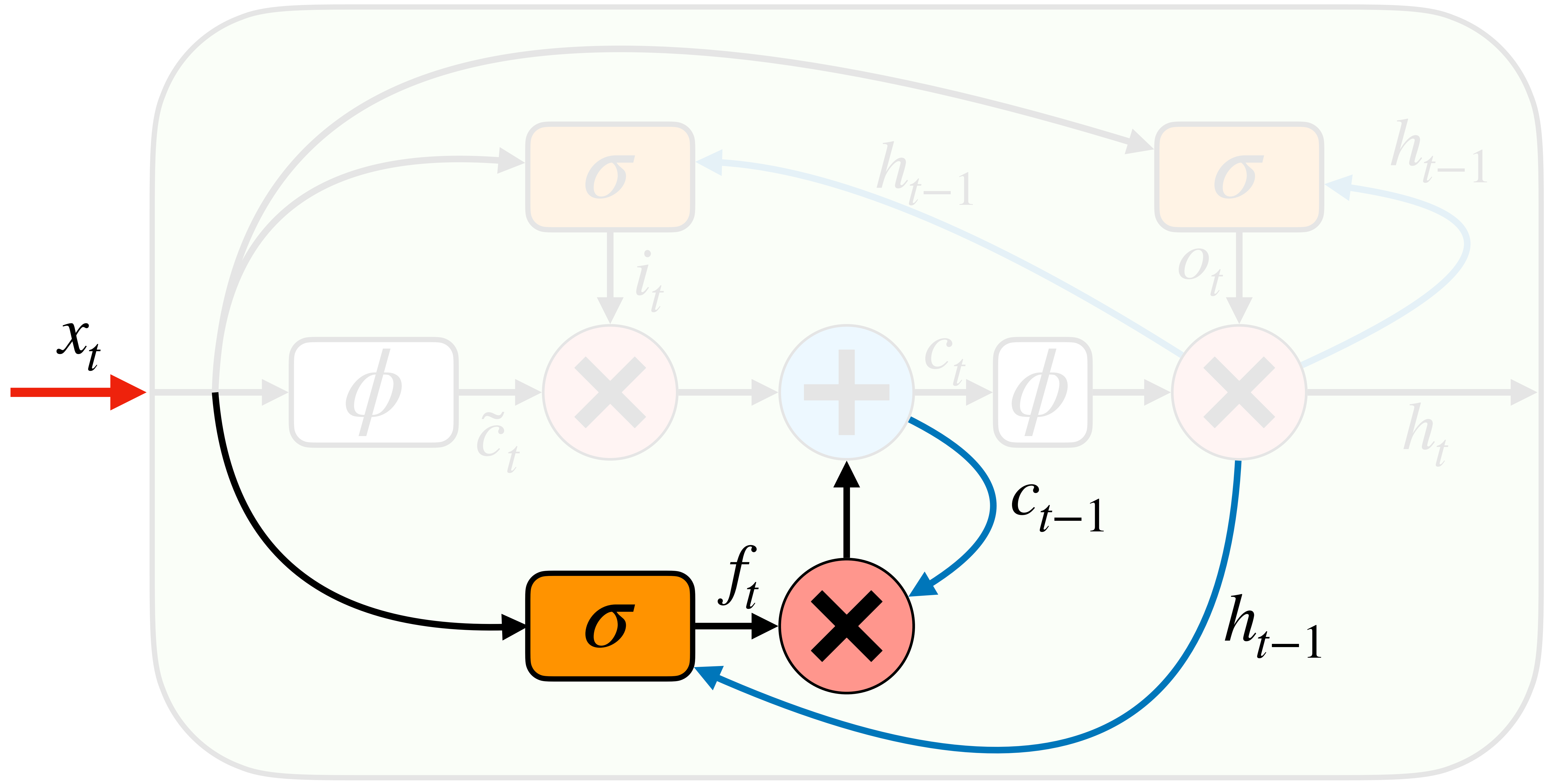
Gates:

$$f_t = \sigma\left(W_{fx}x_t + W_{fh}h_{t-1} + b_f\right)$$
$$i_t = \sigma\left(W_{ix}x_t + W_{ih}h_{t-1} + b_i\right)$$
$$o_t = \sigma\left(W_{ox}x_t + W_{oh}h_{t-1} + b_o\right)$$

$$\tilde{c}_t = \phi\left(W_{cx}x_t + W_{ch}h_{t-1} + b_c\right)$$
$$c_t = i_t \times \tilde{c}_t + f_t \times c_{t-1}$$
$$h_t = o_t \times \phi(c_t)$$

Forget Gate

I went to **the lecture**



$$\boxed{f_t = \sigma\left(W_{fx}x_t + W_{fh}h_{t-1} + b_f\right)}$$

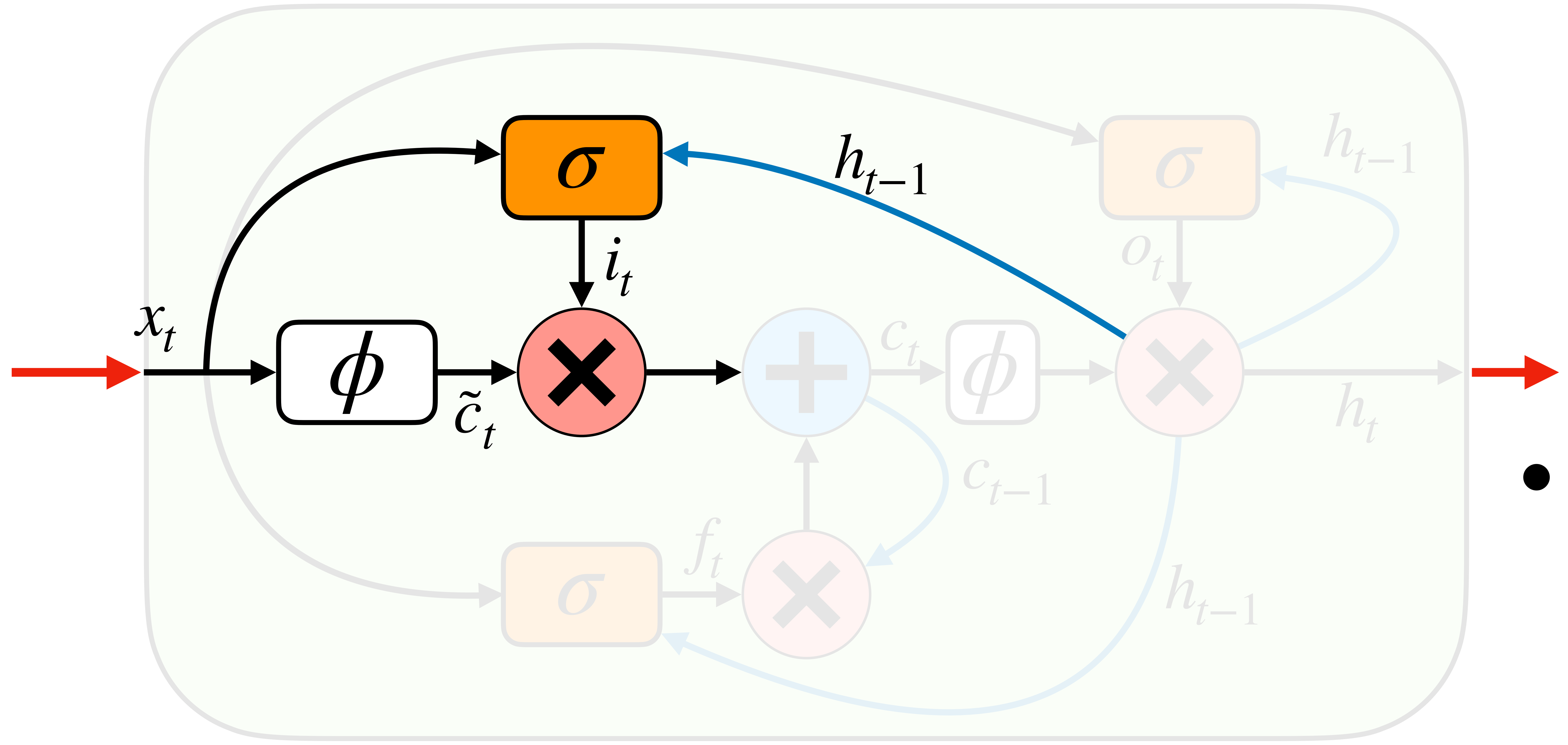
$$\tilde{c}_t = \phi\left(W_{cx}x_t + W_{ch}h_{t-1} + b_c\right)$$

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

- Forget gate controls how much memory is forgotten
    - 1 -> remember the past
    - 0 -> forget everything up to now

Input Gate

I went to **the** **lecture**



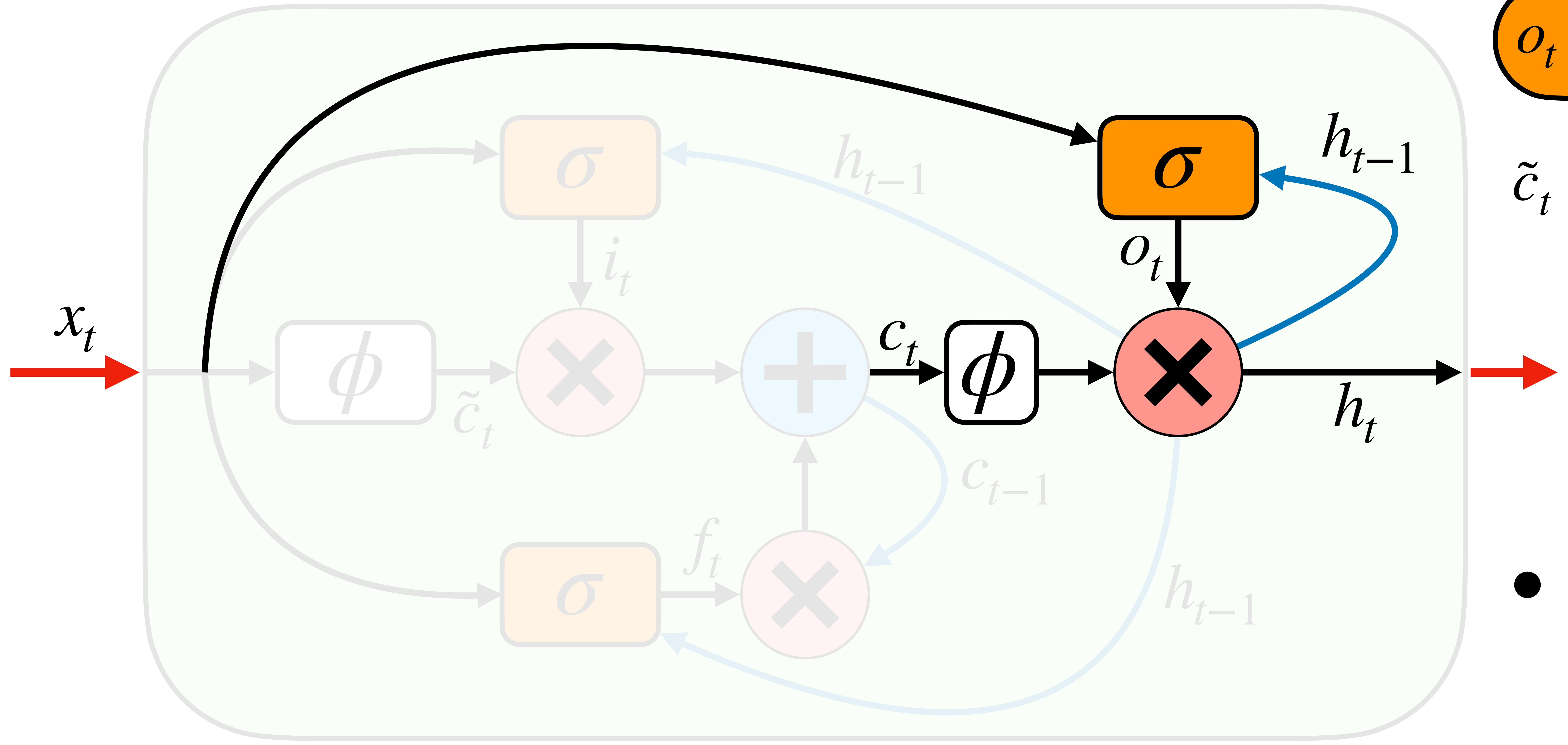
$$\boxed{i_t = \sigma\left(W_{ix}x_t + W_{ih}h_{t-1} + b_i\right)}$$

$$\tilde{c}_t = \phi\left(W_{cx}x_t + W_{ch}h_{t-1} + b_c\right)$$

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

- Input gate controls how new info is added to state memory
  - 0 -> ignore current time step
  - What might be ignored?

Output Gate



$$\boxed{o_t = \sigma\left(W_{ox}x_t + W_{oh}h_{t-1} + b_o\right)}$$

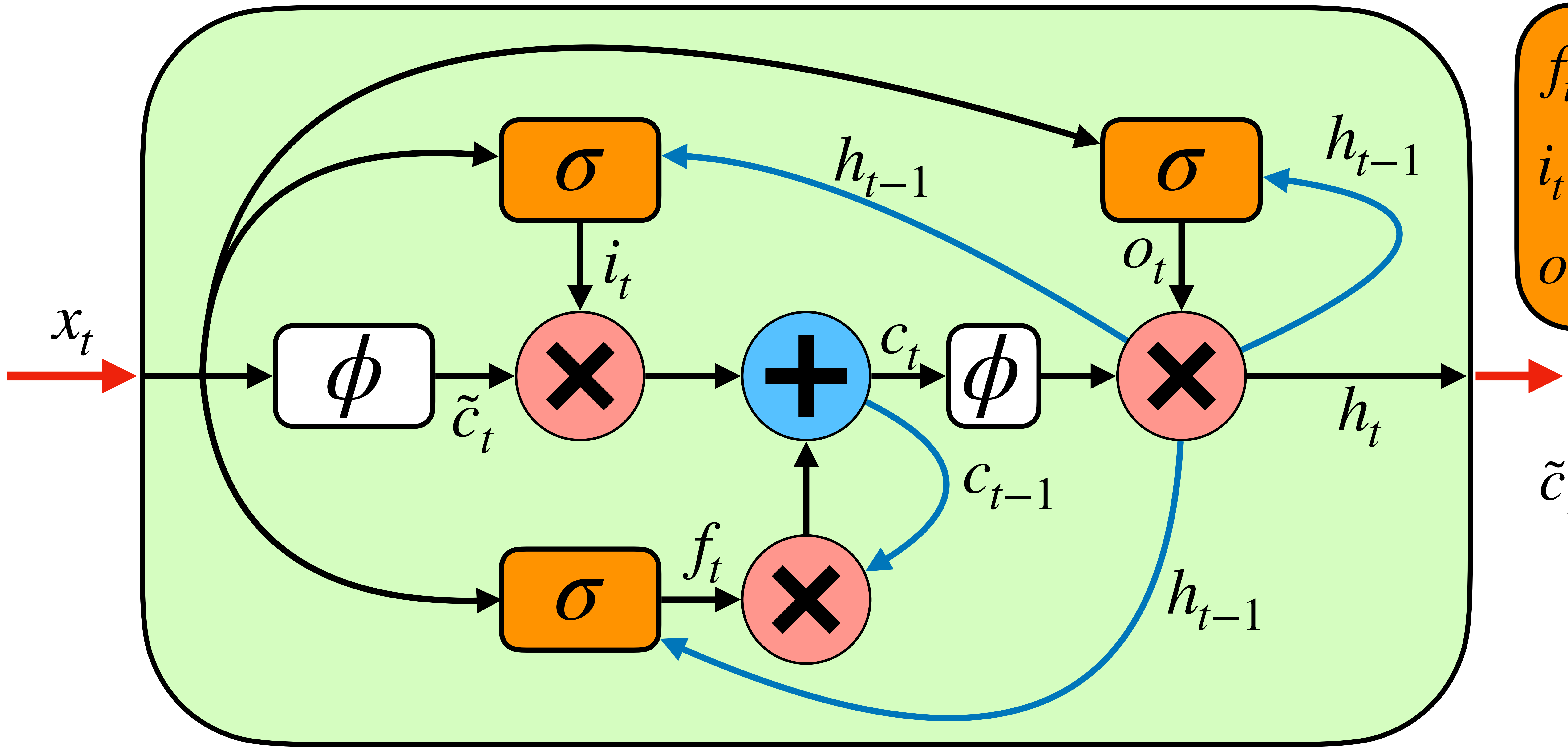
$$\tilde{c}_t = \phi\left(W_{cx}x_t + W_{ch}h_{t-1} + b_c\right)$$

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

$$h_t = \boxed{o_t \times \phi(c_t)}$$

- Output gate decides how the hidden state will influence gate values at next time step

Long Short Term Memory (LSTM)



Gates:

$$f_t = \sigma\left(W_{fx}x_t + W_{fh}h_{t-1} + b_f\right)$$
$$i_t = \sigma\left(W_{ix}x_t + W_{ih}h_{t-1} + b_i\right)$$
$$o_t = \sigma\left(W_{ox}x_t + W_{oh}h_{t-1} + b_o\right)$$

$$\tilde{c}_t = \phi\left(W_{cx}x_t + W_{ch}h_{t-1} + b_c\right)$$

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

$$h_t = o_t \times \phi(c_t)$$

Questions!

- For what type of input might the model learn to make the input gate be 0 ?
- What happens if the forget gate is 0?
- What happens if both the forget gate and input gate are 0 ?
- What happens if both the forget gate and input gate are 1 ?

Gates:

$$f_t = \sigma\left(W_{fx}x_t + W_{fh}h_{t-1} + b_f\right)$$
$$i_t = \sigma\left(W_{ix}x_t + W_{ih}h_{t-1} + b_i\right)$$
$$o_t = \sigma\left(W_{ox}x_t + W_{oh}h_{t-1} + b_o\right)$$

$$\tilde{c}_t = \phi\left(W_{cx}x_t + W_{ch}h_{t-1} + b_c\right)$$
$$c_t = i_t \times \tilde{c}_t + f_t \times c_{t-1}$$
$$h_t = o_t \times \phi(c_t)$$

Gated Recurrent Unit (GRU)

- Also uses gates to avoid dampening gradient signal every time step

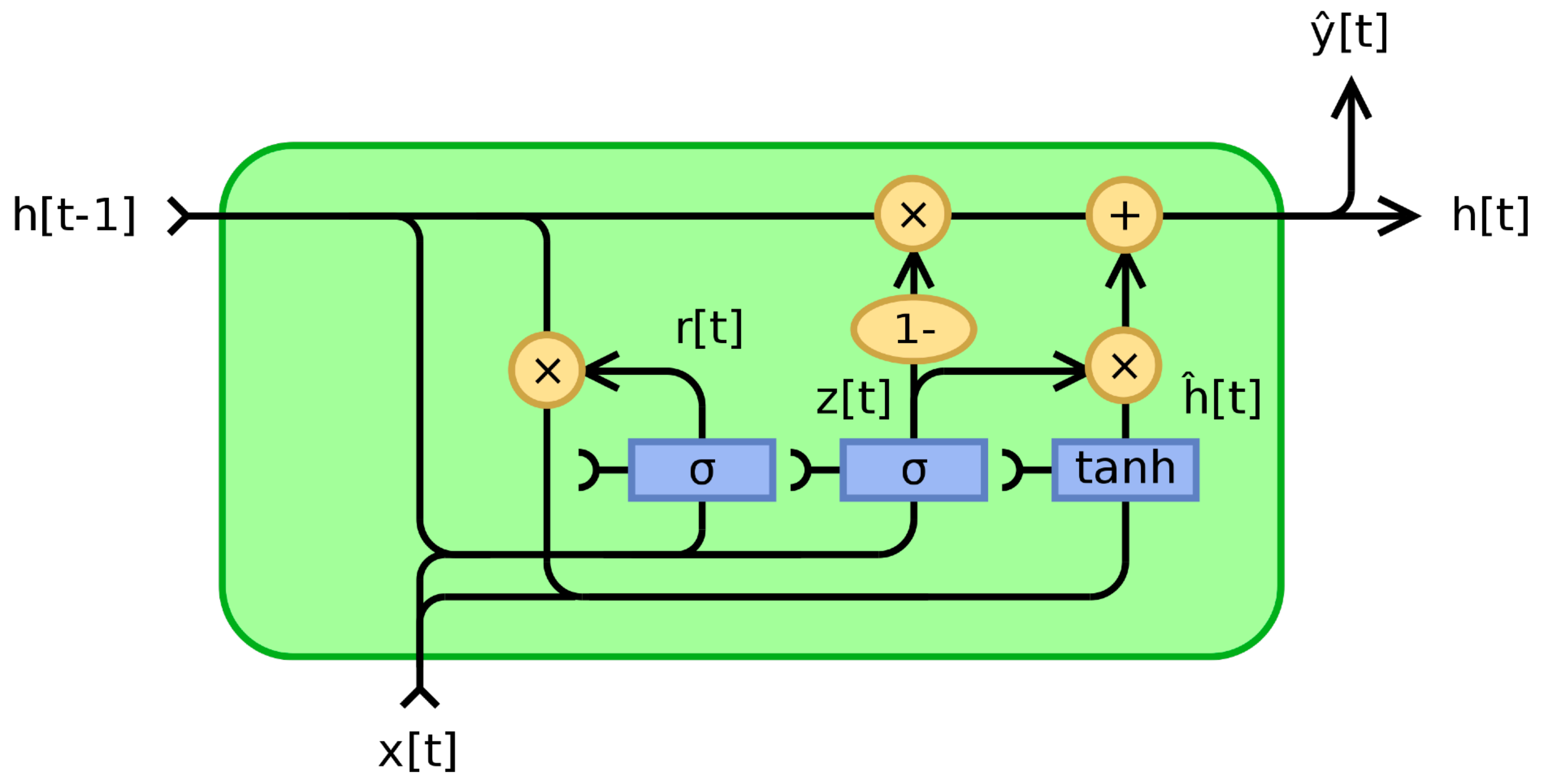
$$h_t = (1 - \mathbf{z}) \odot h_{t-1} + \mathbf{z} \odot \textbf{func}(x_t, h_{t-1}) \qquad h_t = h_{t-1} \odot \mathbf{f} + \textbf{func}(x_t)$$

GRU **LSTM**

- Works similarly to LSTM
    - Theoretically less powerful (find out why tomorrow!)
    - Typically faster to train and sometimes works better than LSTMs

Gated Recurrent Unit (GRU)

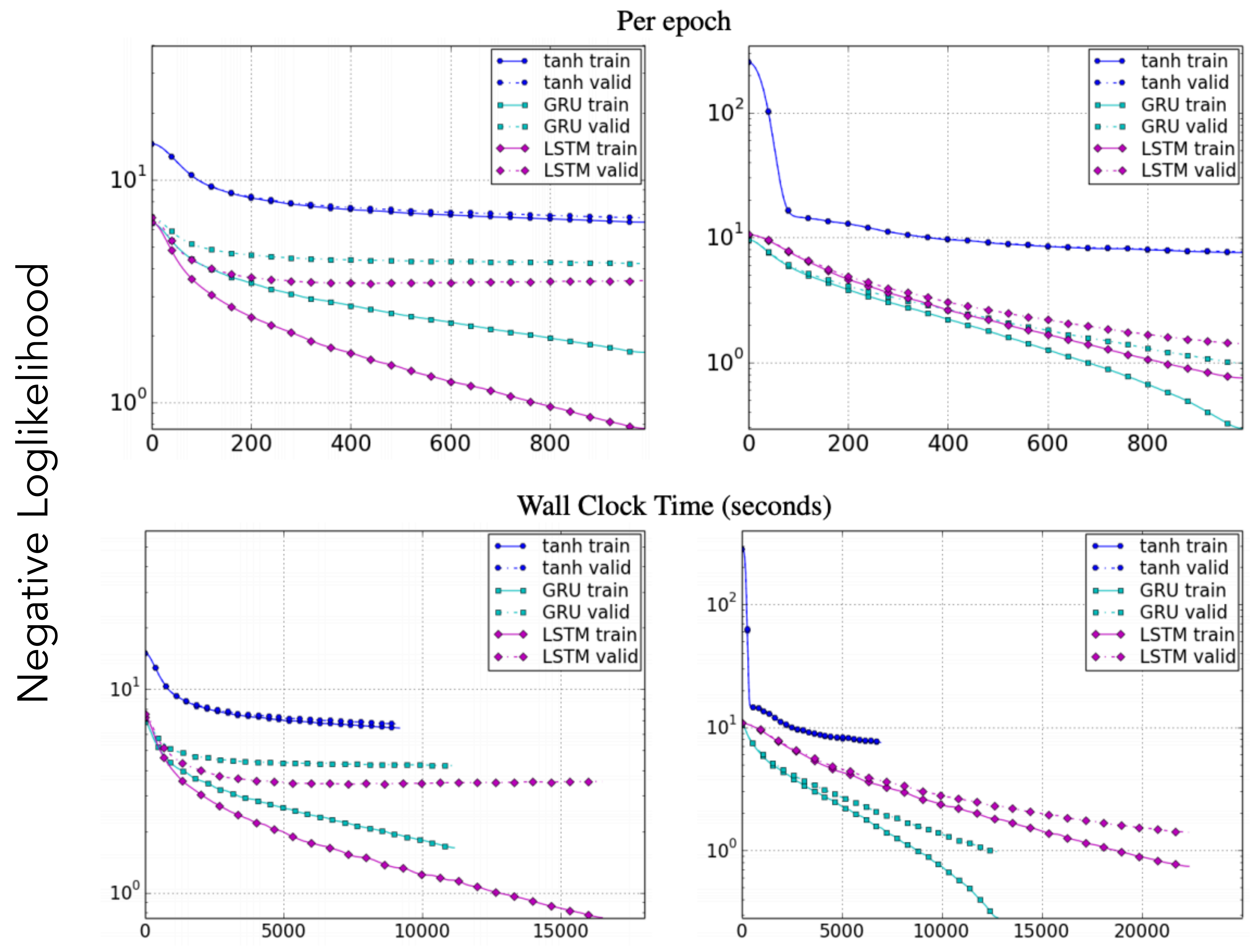
- **z** is update gate (used to update hidden state), **r** is reset gate (used to reset hidden state)
- The single hidden state and simpler update gate gives simpler mixing algorithm than in LSTMs



$$z_t = \sigma_g(W_z x_t + U_z h_{t-1} + b_z)$$
$$r_t = \sigma_g(W_r x_t + U_r h_{t-1} + b_r)$$
$$h_t = (1 - z_t) \circ h_{t-1} + z_t \circ \sigma_h(W_h x_t + U_h(r_t \circ h_{t-1}) + b_h)$$

Which is better?

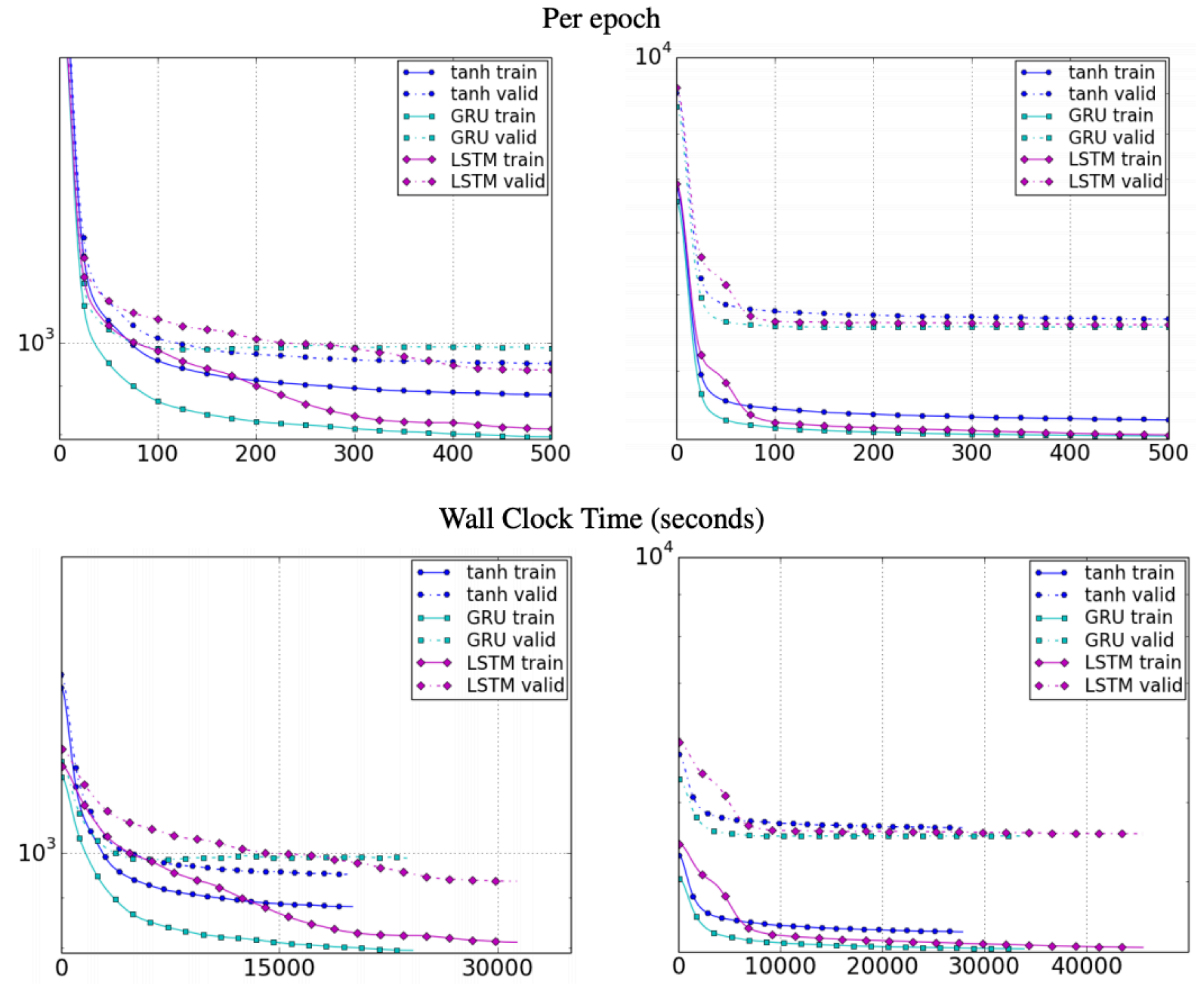
Speech Signal Modeling



(a) Ubisoft Dataset A

(b) Ubisoft Dataset B

Music Modeling



(a) Nottingham Dataset

(b) MuseData Dataset

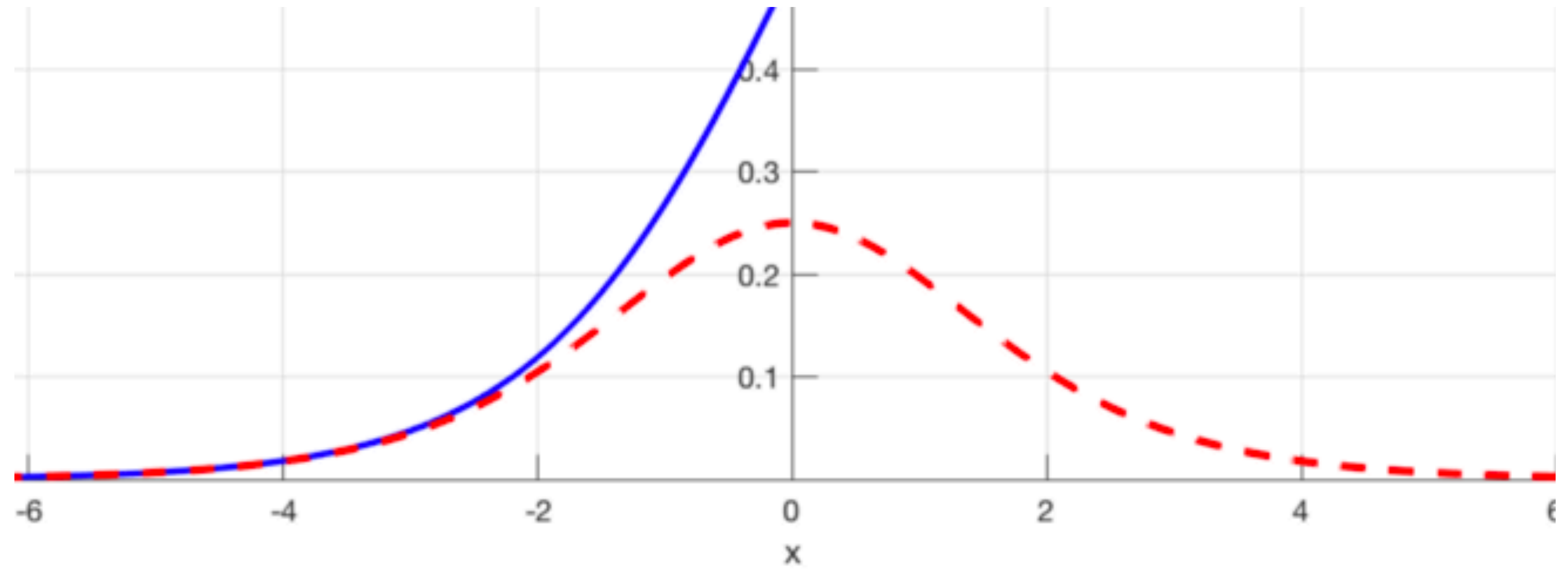
(Chung et al, 2014)

Question

What are the advantages of using LSTMs and GRUs?

Vanishing Gradients?

Recurrent Neural Networks	Long Short Term Memory
State maintained by hidden state feedback	State maintained by cell value
$h_t = \sigma\left(W_{hx}x_t + W_{hh}h_{t-1} + b_h\right)$	$c_t = i_t \times \tilde{c}_t + f_t \times c_{t-1}$
Gradient systemically squashed by sigmoid	Gradient set by value of forget gate



$$\frac{\partial c_t}{\partial c_{t-1}} = f_t$$

Can still vanish, but only if forget gate closes!

Question

What’s a disadvantage of using a LSTM or GRU?

Question

What's a disadvantage of using a LSTM or GRU?

More parameters!

$$f_t = \sigma\left(W_{fx}x_t + W_{fh}h_{t-1} + b_f\right)$$
$$i_t = \sigma\left(W_{ix}x_t + W_{ih}h_{t-1} + b_i\right)$$
$$o_t = \sigma\left(W_{ox}x_t + W_{oh}h_{t-1} + b_o\right)$$
$$\tilde{c}_t = \phi\left(W_{cx}x_t + W_{ch}h_{t-1} + b_c\right)$$
$$c_t = i_t \times \tilde{c}_t + f_t \times c_{t-1}$$
$$h_t = o_t \times \phi(c_t)$$

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

Question

Could there be better architectures than GRUs and LSTMs?

Optimal Architectures?

MUT1:

$$\begin{aligned} z &= \mathrm{sigm}(W_{\mathrm{xz}}x_t + b_{\mathrm{z}}) \\ r &= \mathrm{sigm}(W_{\mathrm{xr}}x_t + W_{\mathrm{hr}}h_t + b_{\mathrm{r}}) \\ h_{t+1} &= \tanh(W_{\mathrm{hh}}(r \odot h_t) + \tanh(x_t) + b_{\mathrm{h}}) \odot z \\ &+ h_t \odot (1 - z) \end{aligned}$$

MUT2:

$$\begin{aligned} z &= \mathrm{sigm}(W_{\mathrm{xz}}x_t + W_{\mathrm{hz}}h_t + b_{\mathrm{z}}) \\ r &= \mathrm{sigm}(x_t + W_{\mathrm{hr}}h_t + b_{\mathrm{r}}) \\ h_{t+1} &= \tanh(W_{\mathrm{hh}}(r \odot h_t) + W_{xh}x_t + b_{\mathrm{h}}) \odot z \\ &+ h_t \odot (1 - z) \end{aligned}$$

MUT3:

$$\begin{aligned} z &= \mathrm{sigm}(W_{\mathrm{xz}}x_t + W_{\mathrm{hz}}\tanh(h_t) + b_{\mathrm{z}}) \\ r &= \mathrm{sigm}(W_{\mathrm{xr}}x_t + W_{\mathrm{hr}}h_t + b_{\mathrm{r}}) \\ h_{t+1} &= \tanh(W_{\mathrm{hh}}(r \odot h_t) + W_{xh}x_t + b_{\mathrm{h}}) \odot z \\ &+ h_t \odot (1 - z) \end{aligned}$$

Arch.	Arith.	XML	PTB
Tanh	0.29493	0.32050	0.08782
LSTM	0.89228	0.42470	0.08912
LSTM-f	0.29292	0.23356	0.08808
LSTM-i	0.75109	0.41371	0.08662
LSTM-o	0.86747	0.42117	0.08933
LSTM-b	0.90163	0.44434	0.08952
GRU	0.89565	0.45963	0.09069
MUT1	**0.92135**	**0.47483**	0.08968
MUT2	0.89735	**0.47324**	0.09036
MUT3	0.90728	0.46478	**0.09161**

Arch.	5M-tst	10M-v	20M-v	20M-tst
Tanh	4.811	4.729	4.635	4.582 (97.7)
LSTM	4.699	4.511	4.437	4.399 (81.4)
LSTM-f	4.785	4.752	4.658	4.606 (100.8)
LSTM-i	4.755	4.558	4.480	4.444 (85.1)
LSTM-o	4.708	4.496	4.447	4.411 (82.3)
LSTM-b	4.698	4.437	4.423	**4.380 (79.83)**
GRU	4.684	4.554	4.559	4.519 (91.7)
MUT1	4.699	4.605	4.594	4.550 (94.6)
MUT2	4.707	4.539	4.538	4.503 (90.2)
MUT3	4.692	4.523	4.530	4.494 (89.47)

(Jozefowicz et al, 2015)

Recap

- Recurrent neural networks can **theoretically** learn to model an **unbounded context length**
  - no increase in model size because weights are shared across time steps
- Practically, however, **vanishing gradients** stop vanilla RNNs from learning useful **long-range dependencies**
- LSTMs and GRUs are variants of recurrent networks that mitigate the vanishing gradient problem
  - used for for **many sequence-to-sequence tasks (up next!)**

References

- Elman, J.L. (1990). Finding Structure in Time. *Cogn. Sci., 14*, 179-211.
- Schuster, M., & Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, *45*(11), 2673–2681.
- Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation, 9,* 1735-1780.
- Cho, K., Merrienboer, B.V., Gülçehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. *Conference on Empirical Methods in Natural Language Processing.*
- Greff, K., Srivastava, R.K., Koutník, J., Steunebrink, B.R., & Schmidhuber, J. (2015). LSTM: A Search Space Odyssey. *IEEE Transactions on Neural Networks and Learning Systems, 28,* 2222-2232.

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.