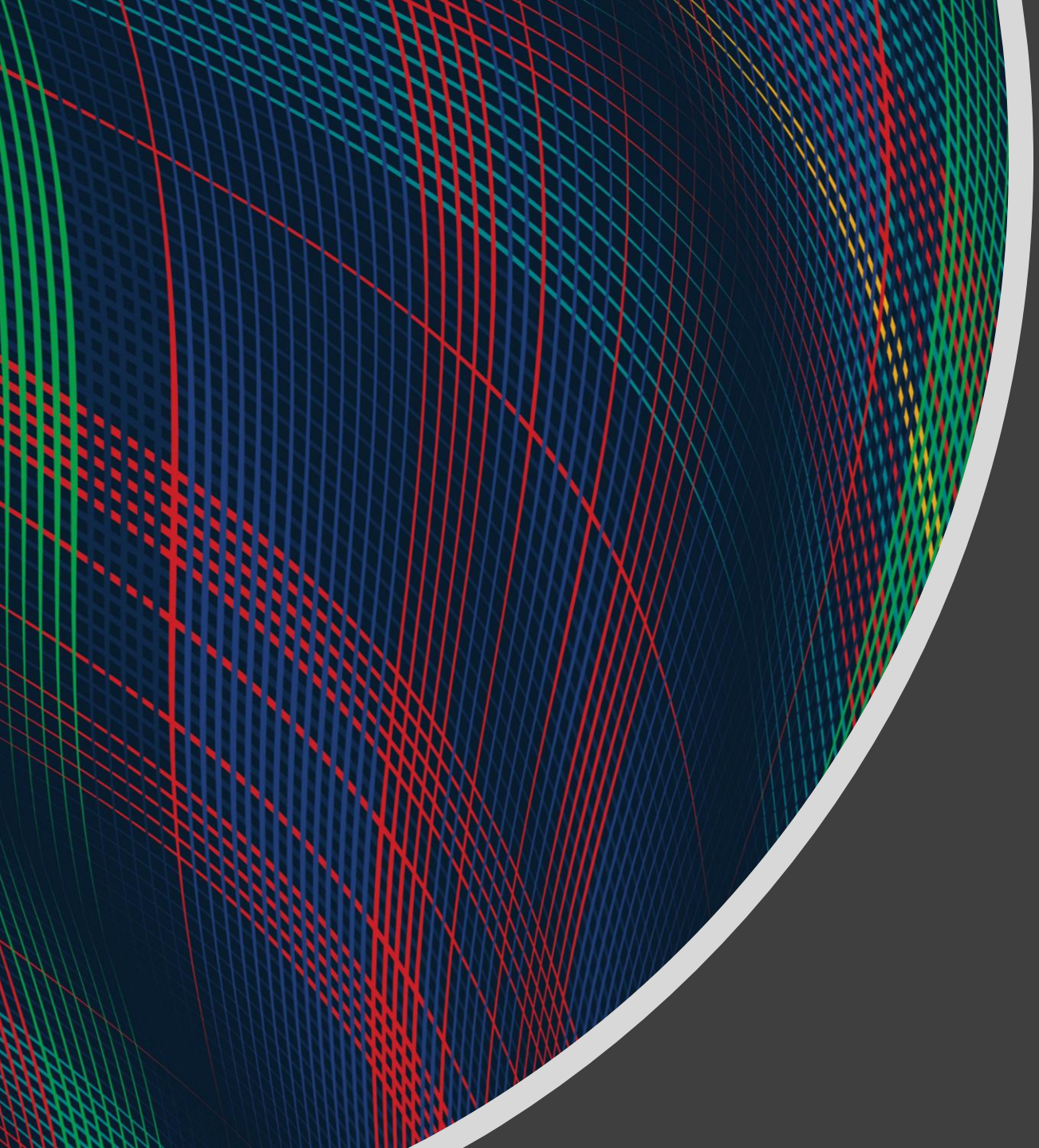


10-315 Introduction to ML

Neural Networks

Instructor: Pat Virtue

Poll 4

Logistic regression for 28x28=784 pixel hand-written digit images into 10 classes:

How many parameters (including ~~bias terms~~)?

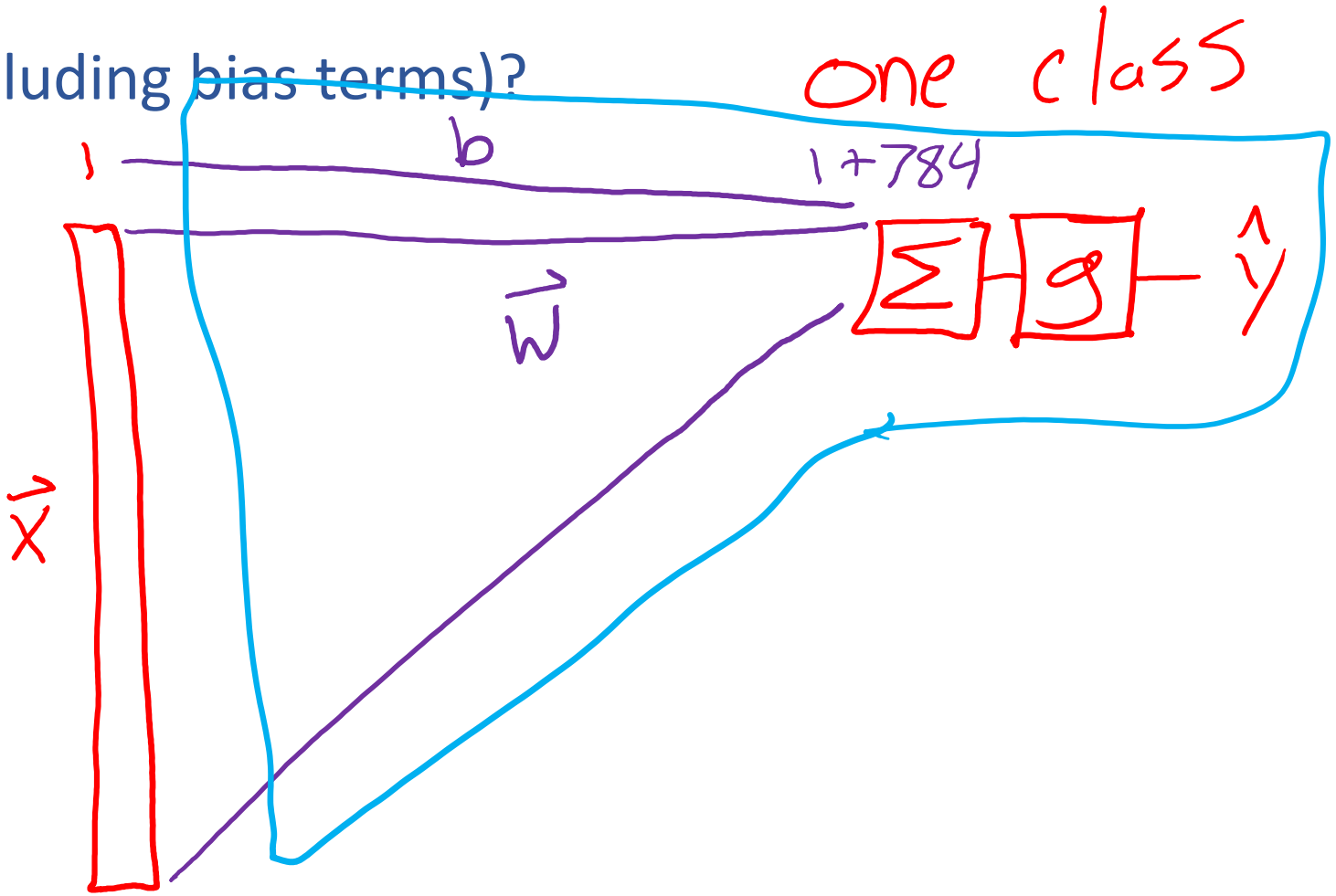
A. 10

B. 10+784

C. 10*784

D. 10*784 + 10

E. 10*784 + 784



Poll 4

Logistic regression for 28x28=784 pixel hand-written digit images into 10 classes:

How many parameters (including bias terms)?

A. 10

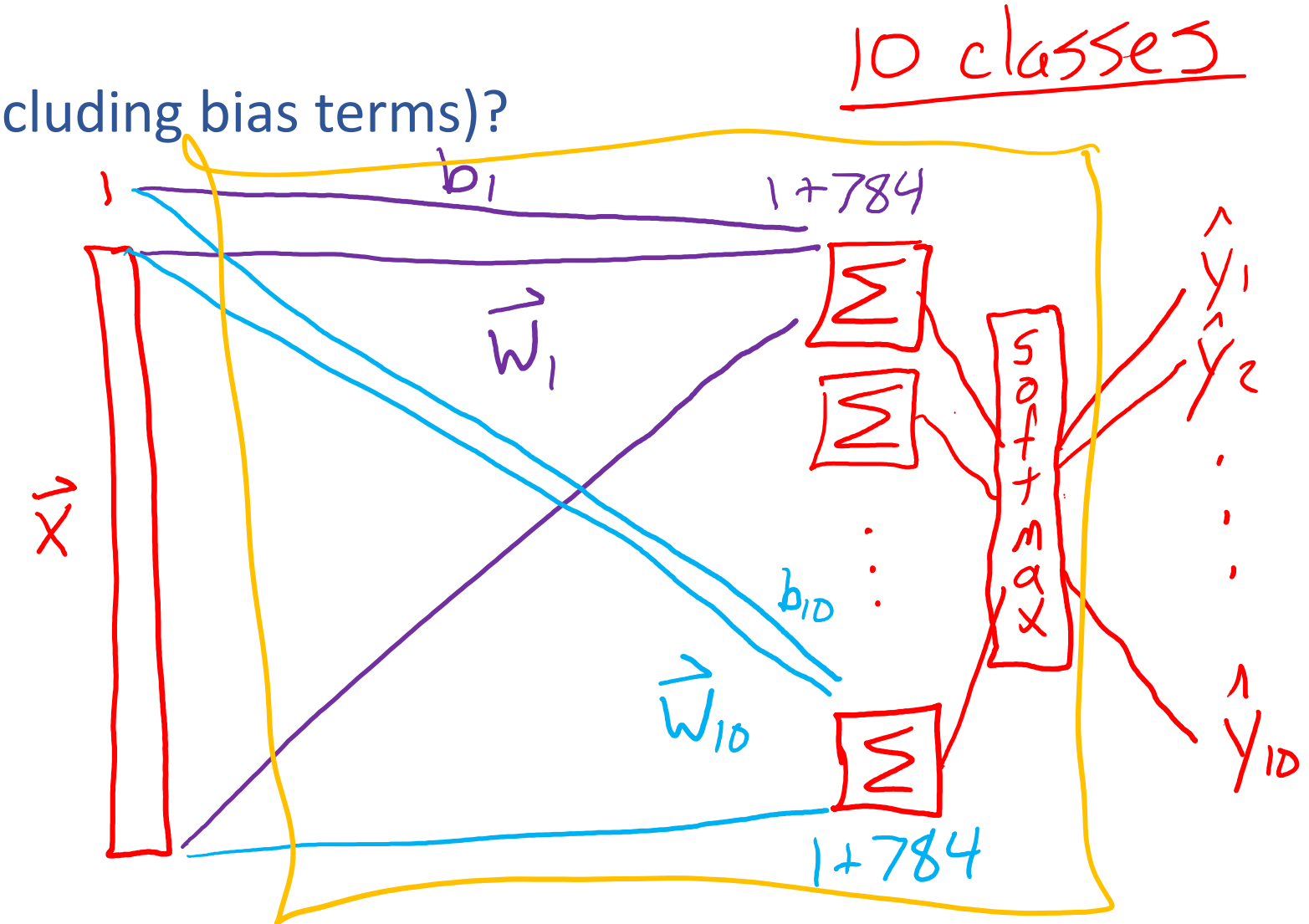
B. 10+784

C. 10*784

D. 10*784 + 10

E. 10*784 + 784

$g(W\vec{x} + \vec{b})$



Poll 4

$$\hat{\vec{y}} = g_{softmax}(W\vec{x})$$

Logistic regression for 28x28=784 pixel hand-written digit images into 10 classes:

How many parameters (including bias terms)?

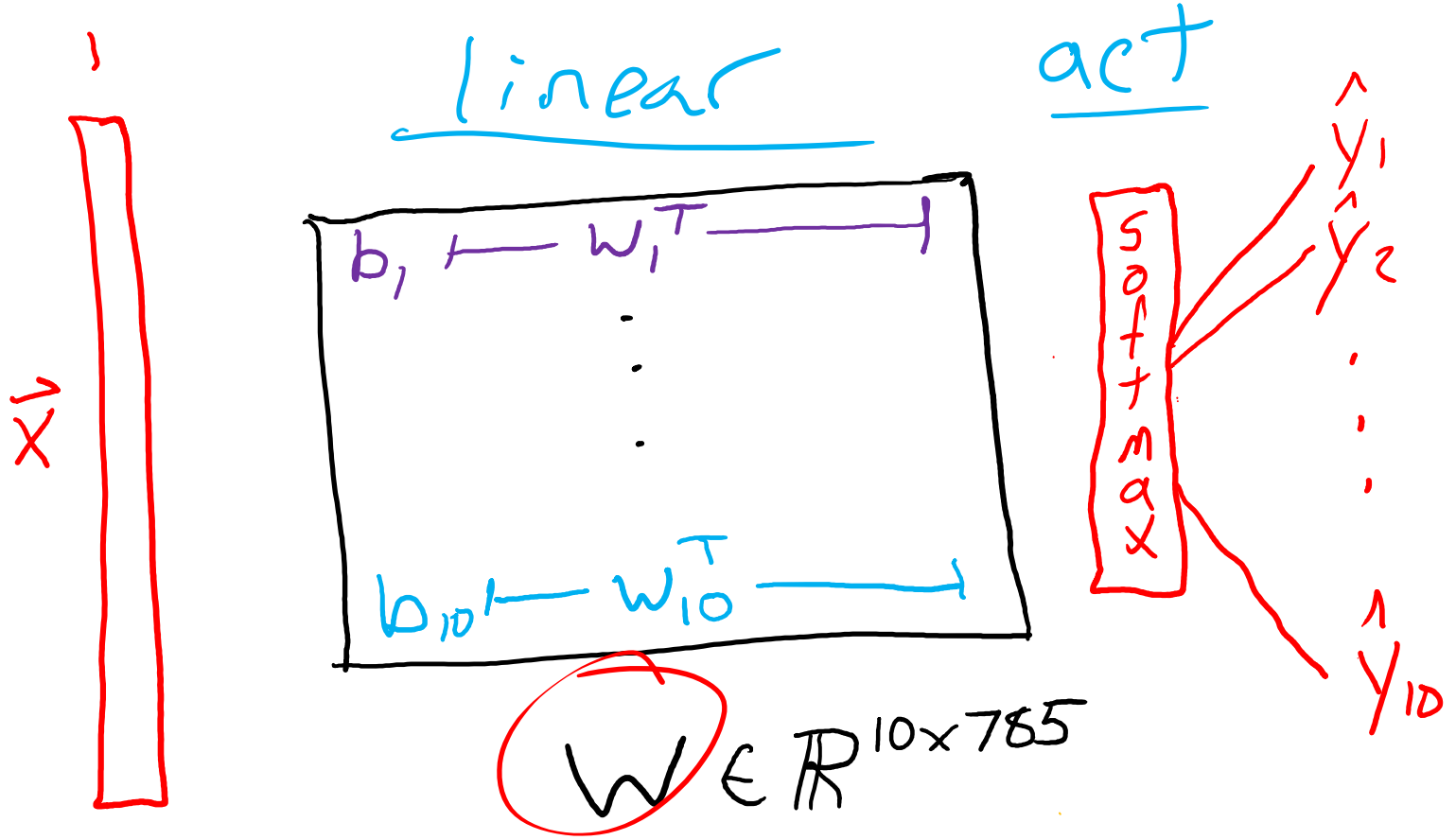
A. 10

B. 10+784

C. 10*784

D. 10*784 + 10

E. 10*784 + 784



Outline

Pre-reading: Neural Networks

- Network diagrams for linear and logistic regression
- Neuron and activation functions
- Three-neuron network

Today: Neural Networks

- Three-neuron network + optimization
- Neural network structure (adding more neurons)
- Forward pass and backpropagation (scalar version)

Network Diagrams for Linear and Logistic Regression

Pre-reading

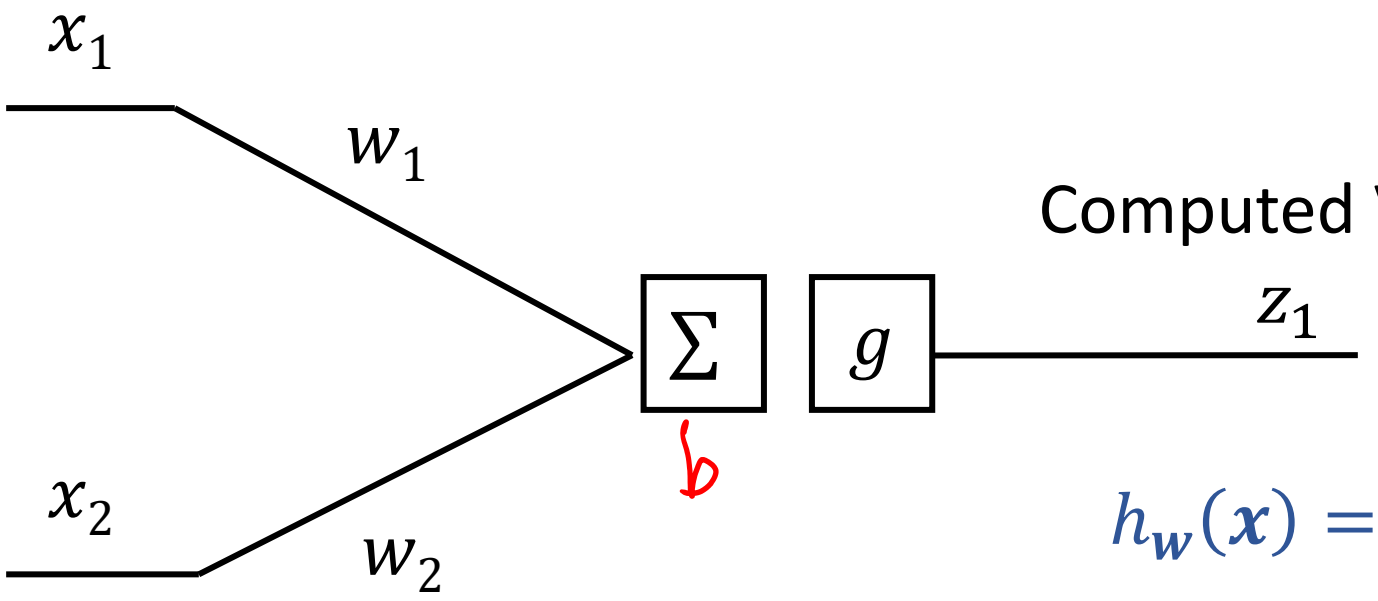
Neuron

Pre-reading

Single Neuron

Single neuron system

- Perceptron (if g is step function)
- Logistic regression (if g is sigmoid)
- Linear regression (if g is nothing)



Computed Value z_1

True Label y

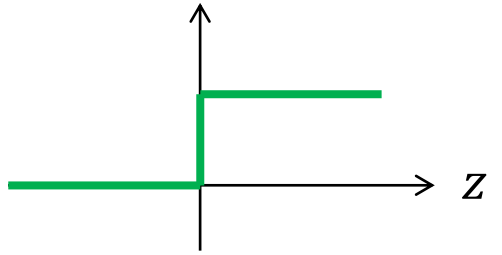
$$h_{\boldsymbol{w}}(\boldsymbol{x}) = z_1$$

$$h_{\boldsymbol{w}}(\boldsymbol{x}) = g\left(\sum_i w_i x_i\right)$$

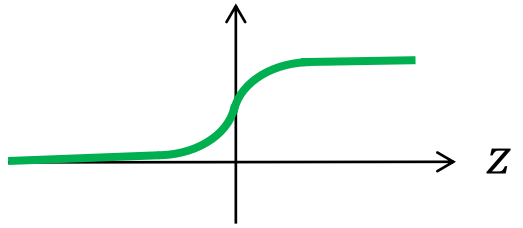
Activation Functions

It would be really helpful to have a g(z) that was nicely differentiable

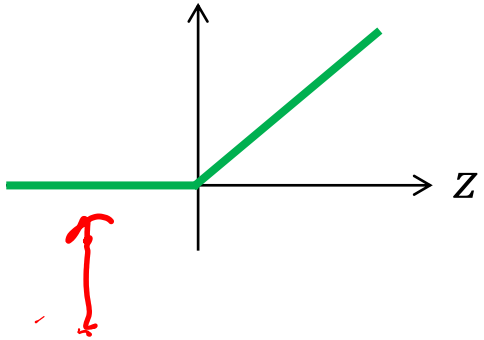
- Hard threshold: $g(z) = \begin{cases} 1 & z \geq 0 \\ 0 & z < 0 \end{cases}$ $\quad \frac{dg}{dz} = \begin{cases} 0 & z \geq 0 \\ 0 & z < 0 \end{cases}$



- Sigmoid: $g(z) = \frac{1}{1+e^{-z}}$ $\quad \frac{dg}{dz} = g(z)\big(1 - g(z)\big)$
- (Softmax)



- ReLU: $g(z) = max(0, z)$ $\quad \frac{dg}{dz} = \begin{cases} 1 & z \geq 0 \\ 0 & z < 0 \end{cases}$



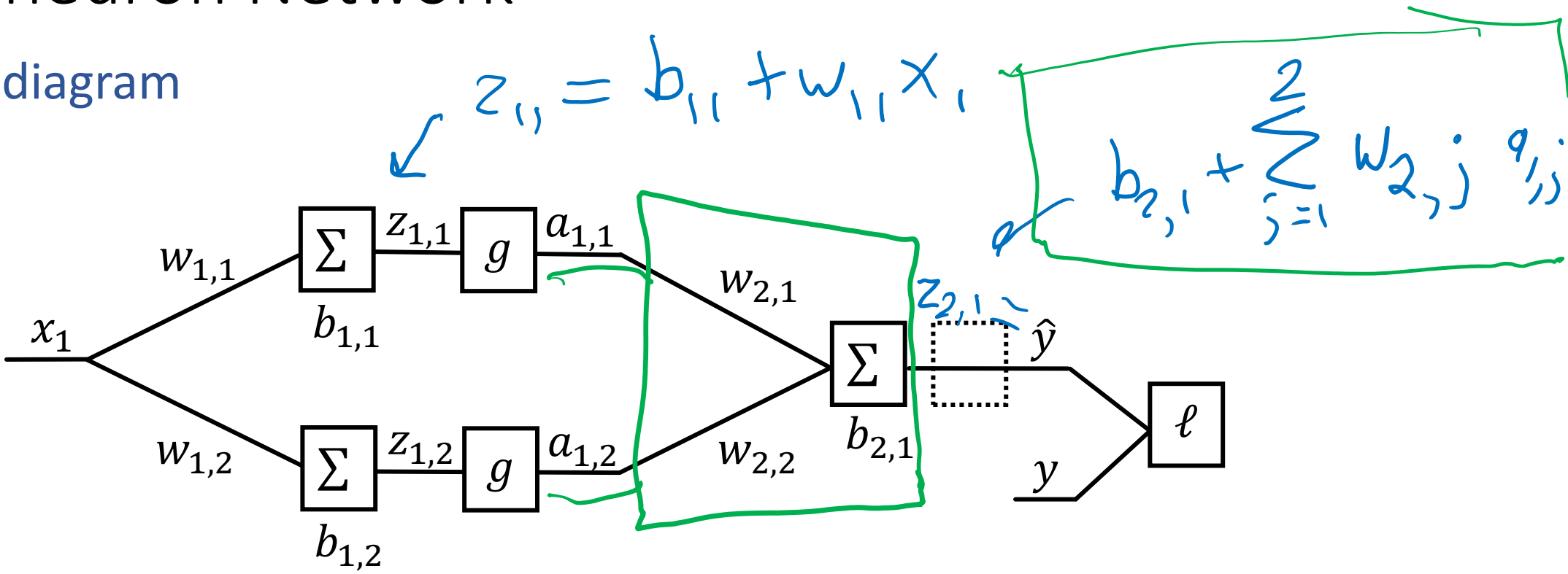
Activation Functions

Pre-reading

Three-neuron Network

Three-neuron Network

Network diagram



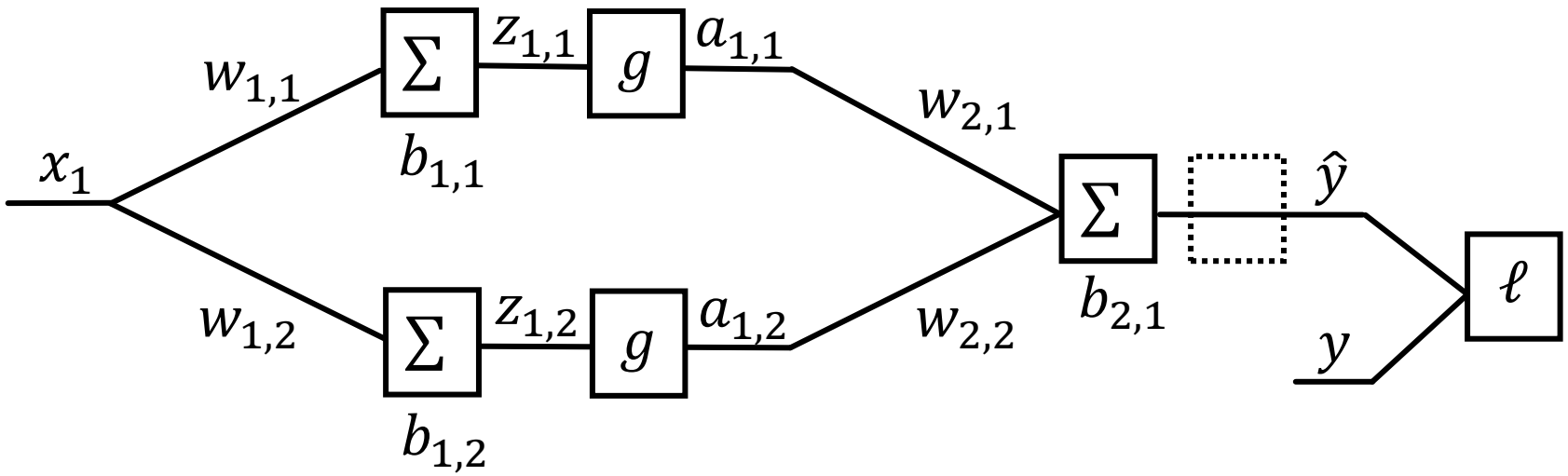
$$\hat{y} = h_{\boldsymbol{\theta}}(\mathbf{x}) = b_2 + \sum_j w_{2,j}\, a_{1,j}$$

$$a_{1,j} = g_{ReLU}\left(b_{1,j} + w_{1,j}\, x_1\right)$$

$a = g(z)$

Poll 1

How many total parameters?



Poll 2

Which of the following updates will we execute during gradient descent?

Select all that apply

A. $x \leftarrow x - \alpha \frac{\partial J}{\partial x}$

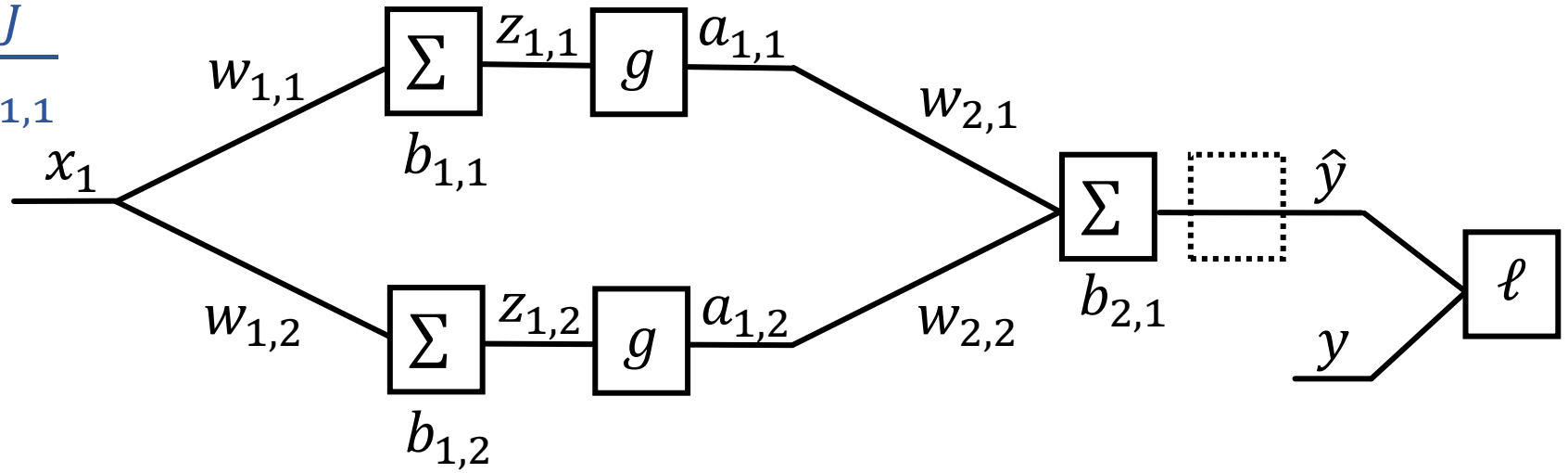
B. $w_{1,1} \leftarrow w_{1,1} - \alpha \frac{\partial J}{\partial w_{1,1}}$

C. $b_{1,1} \leftarrow b_{1,1} - \alpha \frac{\partial J}{\partial b_{1,1}}$

D. $a_{1,1} \leftarrow a_{1,1} - \alpha \frac{\partial J}{\partial a_{1,1}}$

E. $\hat{y} \leftarrow \hat{y} - \alpha \frac{\partial J}{\partial \hat{y}}$

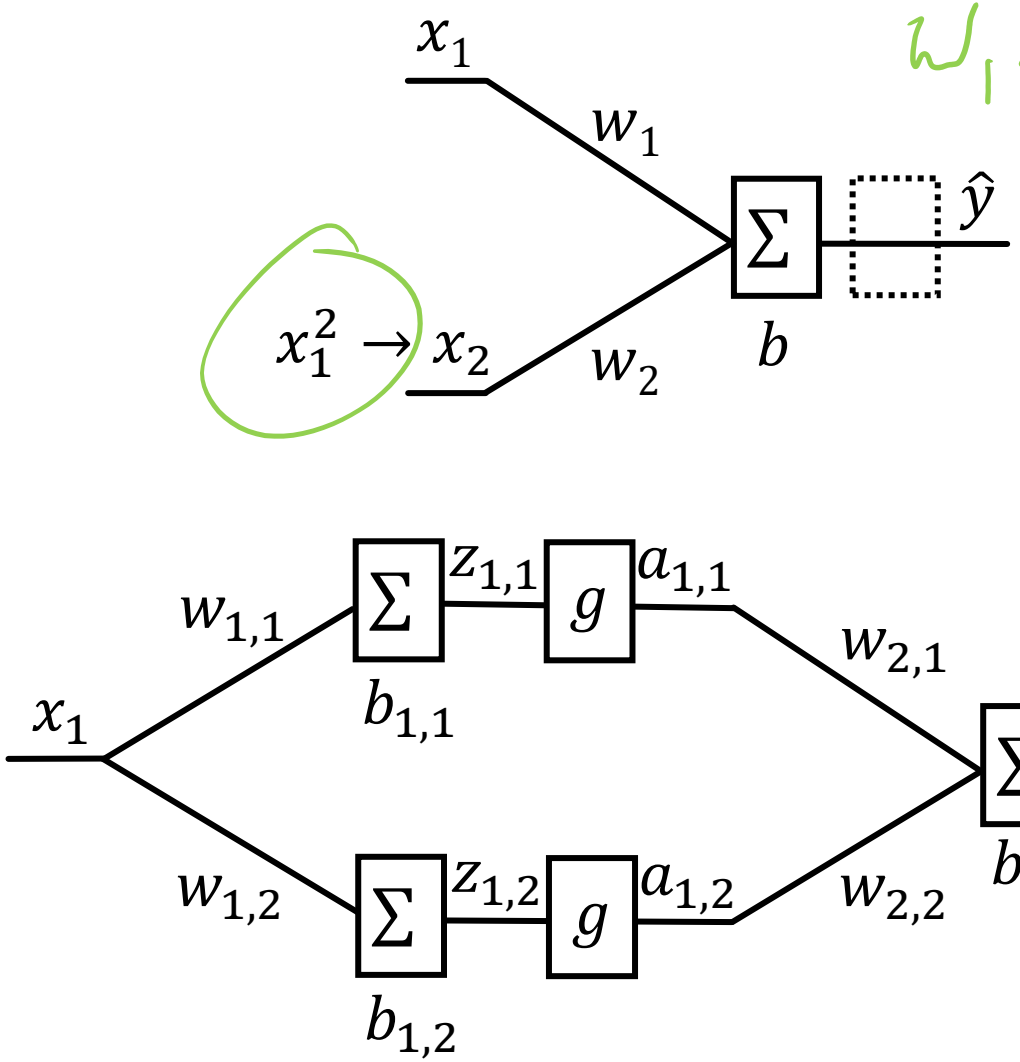
F. $y \leftarrow y - \alpha \frac{\partial J}{\partial y}$

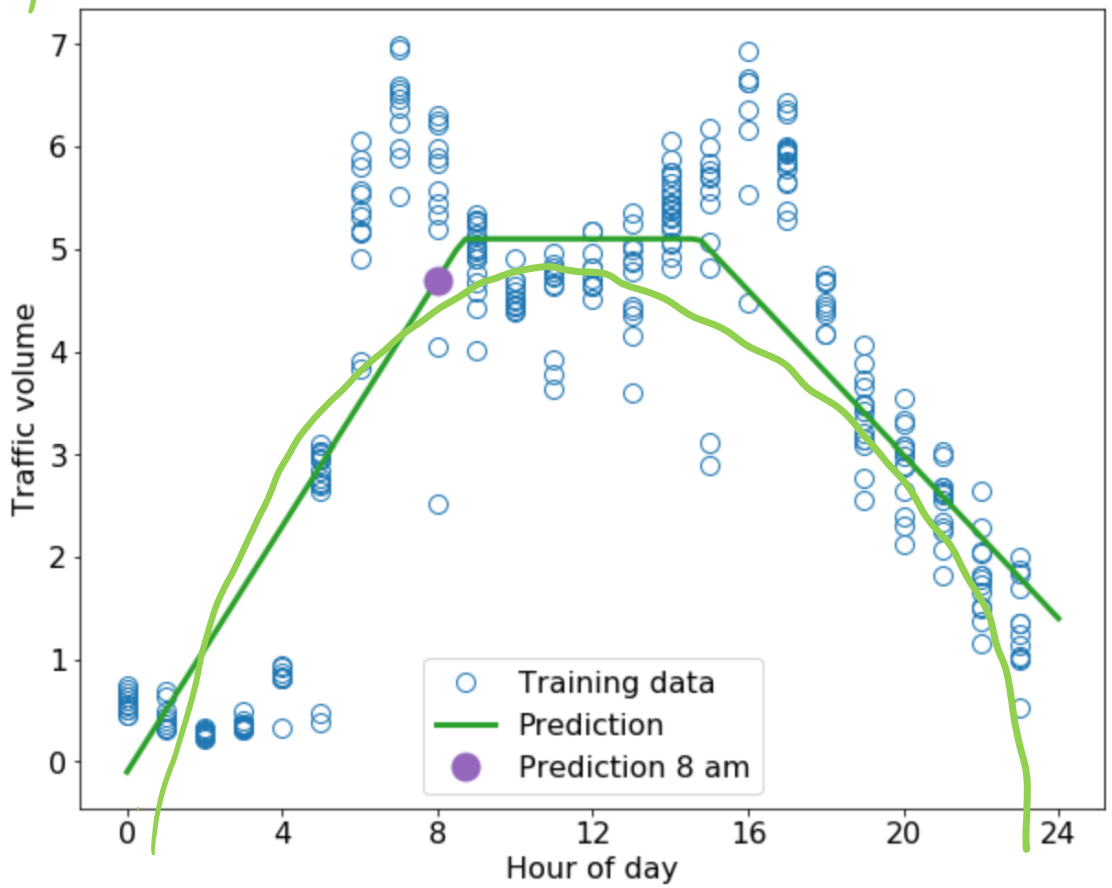


Traffic Data Example

Quadratic feature engineering vs three-neuron network

$w_1 x_1 + w_2 x_1^2 + b$

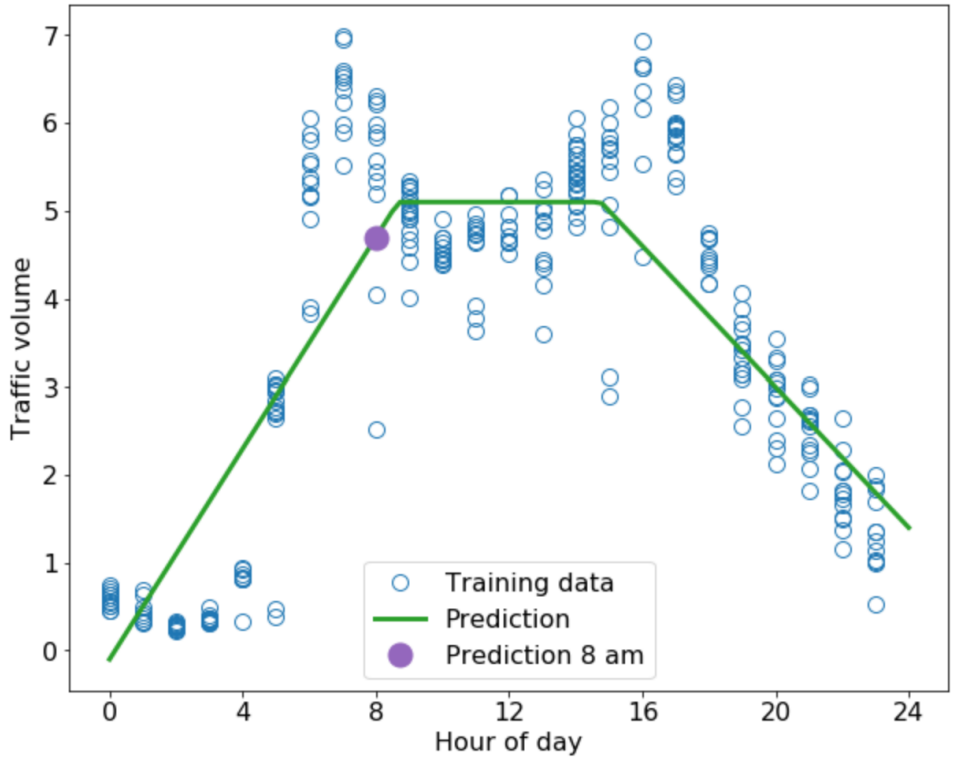


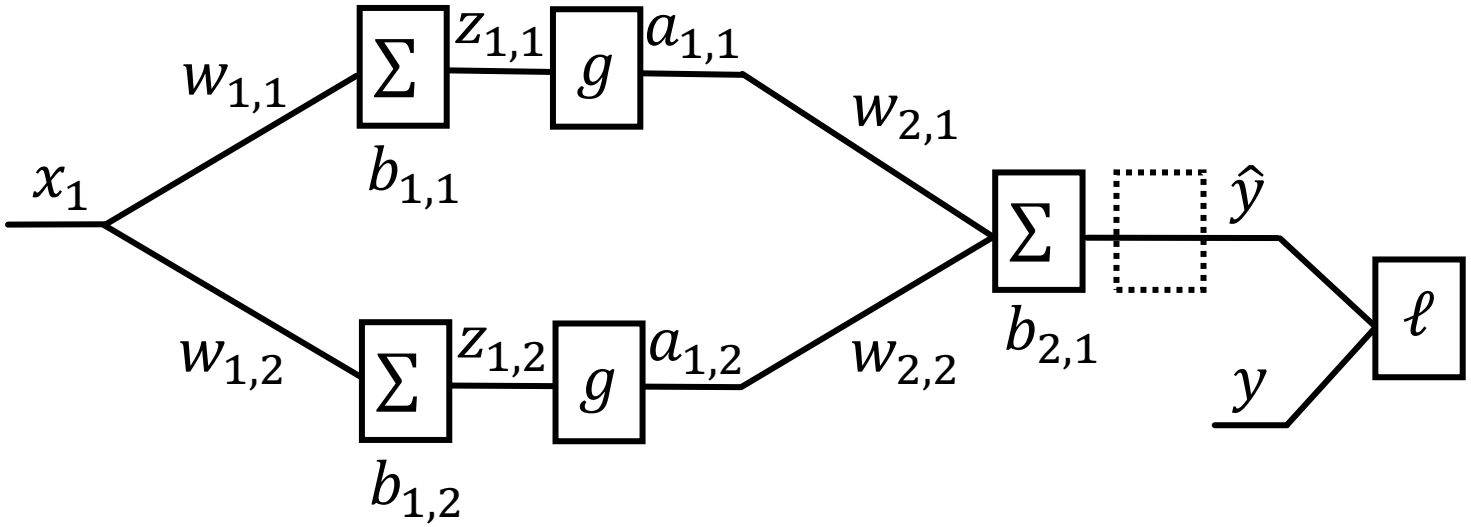


Traffic Data Example

Three-neuron network

Parameters			
$w_{1,1}$:	0.4	$b_{1,1}$:	-5.9
$w_{1,2}$:	-0.6	$b_{1,2}$:	5.2
$w_{2,1}$:	-1.0	$b_{2,1}$:	5.1
$w_{2,2}$:	-1.0		





Traffic Data Example

Three-neuron network

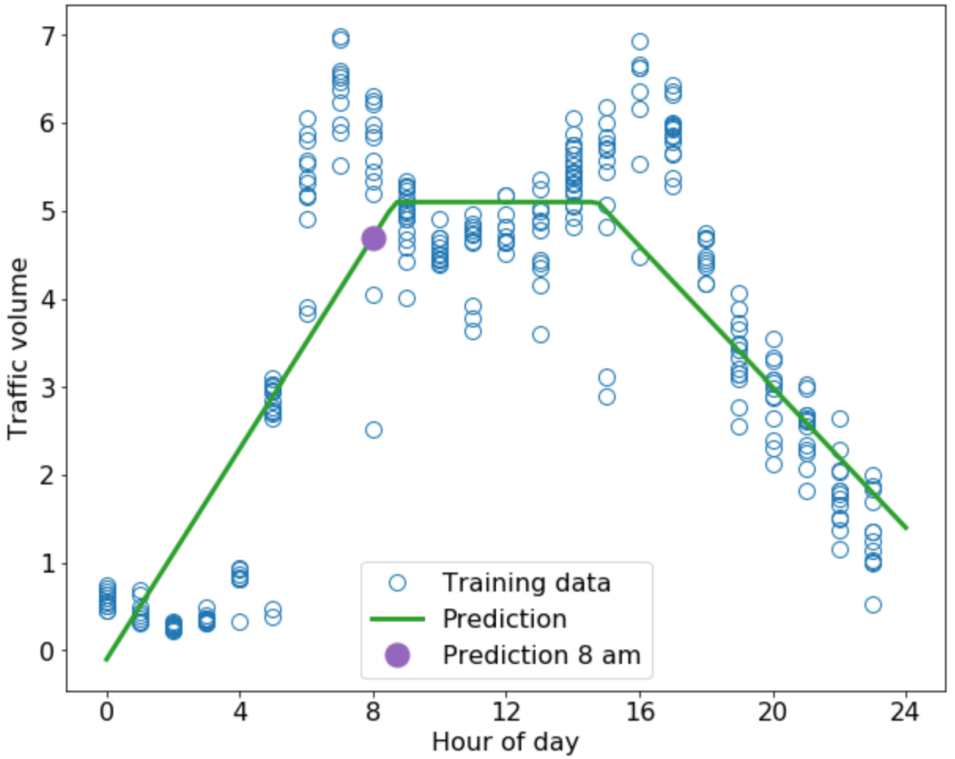
Parameters

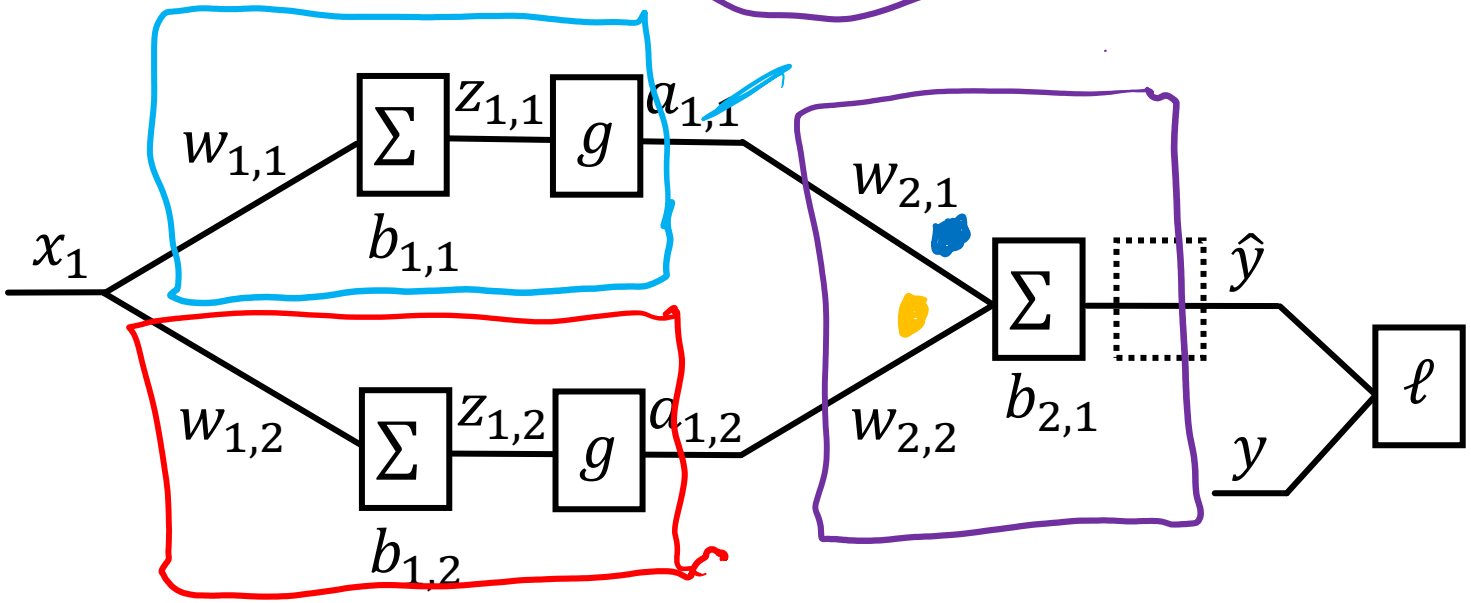
$w_{1,1}$: 0.4 $b_{1,1}$: -5.9

$w_{1,2}$: -0.6 $b_{1,2}$: 5.2

$w_{2,1}$: -1.0 $b_{2,1}$: 5.1

$w_{2,2}$: -1.0

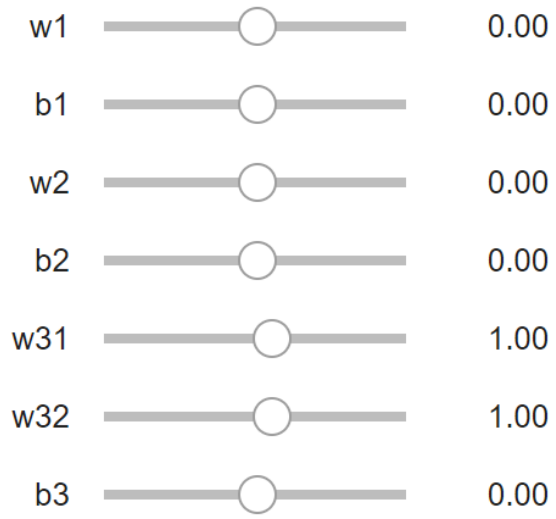


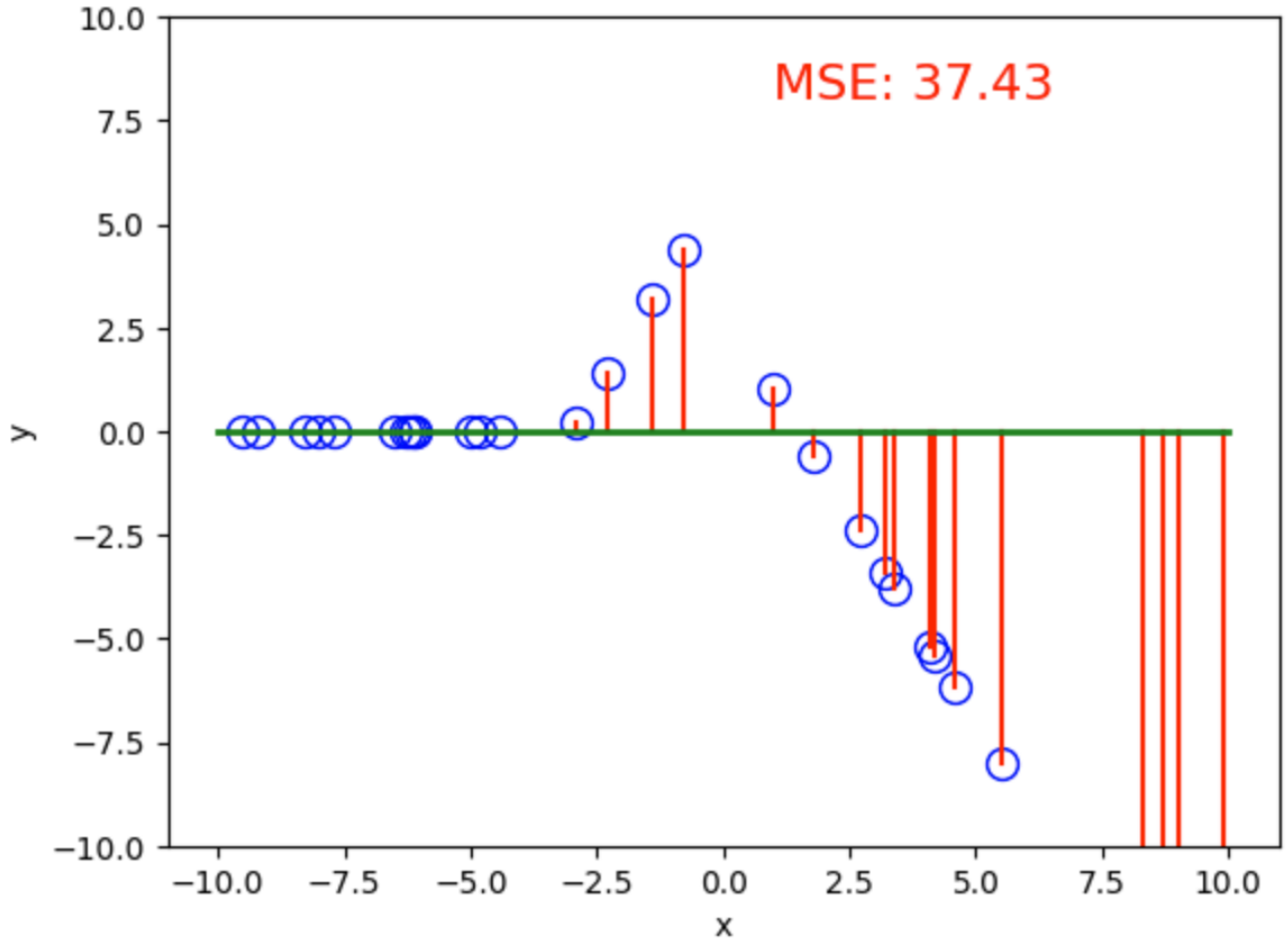


Neural Network Optimization

Three-neuron network

Interactive demo: three_neuron_interactive.ipynb on course website

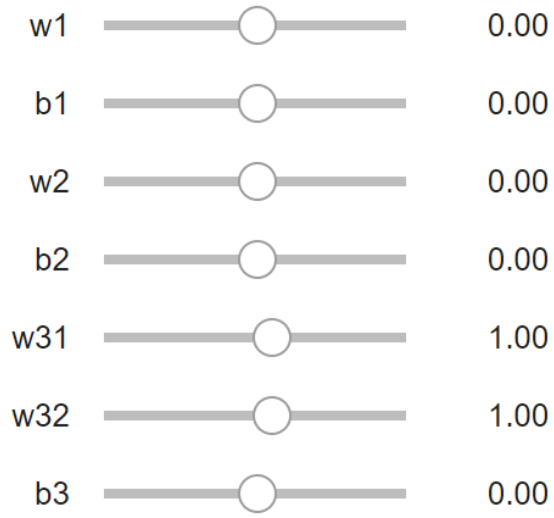


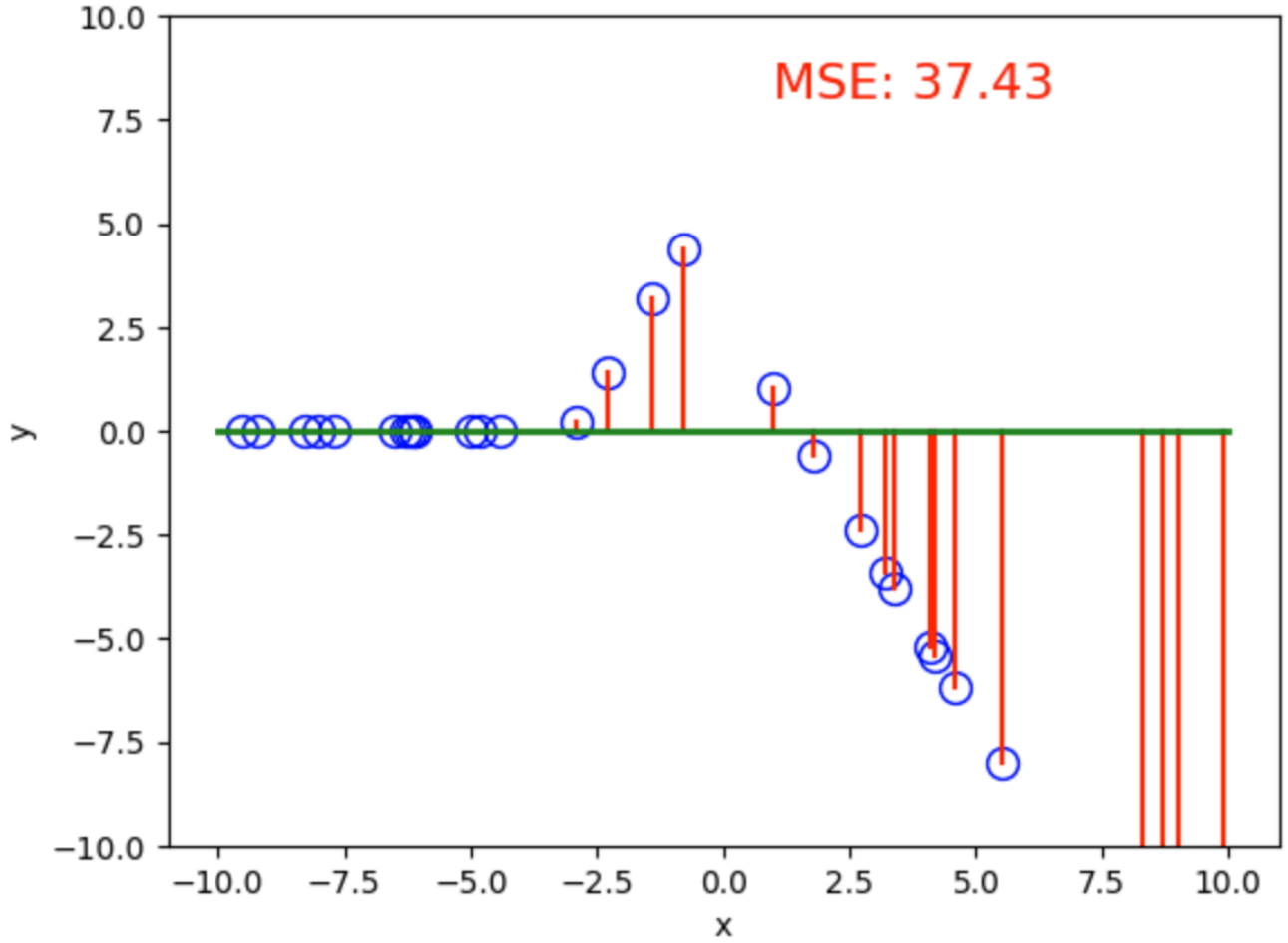


Poll 3

Interactive demo: three_neuron_interactive.ipynb on course website

What's the smallest MSE that you can find?





Objective, Loss Functions, Gradient Descent

Objective

$$J(\theta) = \frac{1}{N}\sum_{i=1}^{N} J^{(i)}(\theta) \qquad J^{(i)}(\theta) = \ell\left(y^{(i)}, \hat{y}^{(i)}\right) \qquad \hat{y} = h(x;\theta)$$

Loss functions

- Regression → Squared error: $\ell(y, \hat{y}) = (y - \hat{y})^2$
- Classfiication → Cross entropy: $\ell(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -\sum_k y_k \log \hat{y}_k$

Gradient descent

while not converged

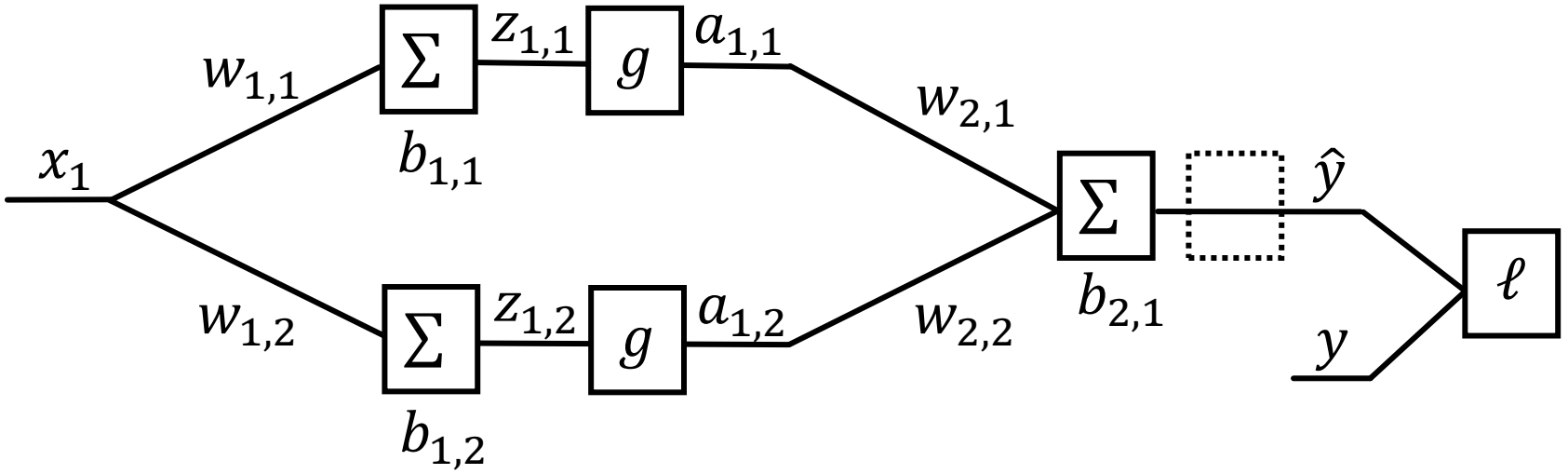
$$\theta \leftarrow \theta - \alpha\, \partial J / \partial \theta$$

Optimization

Three-neuron network

$$\hat{y} = h_{\boldsymbol{\theta}}(\mathbf{x}) = b_2 + \sum_j w_{2,j}\, a_{1,j}$$

$$a_{1,j} = g\left(b_{1,j} + w_{1,j}\, x_1\right)$$



$$\frac{\partial J}{\partial w_{2,1}} =$$

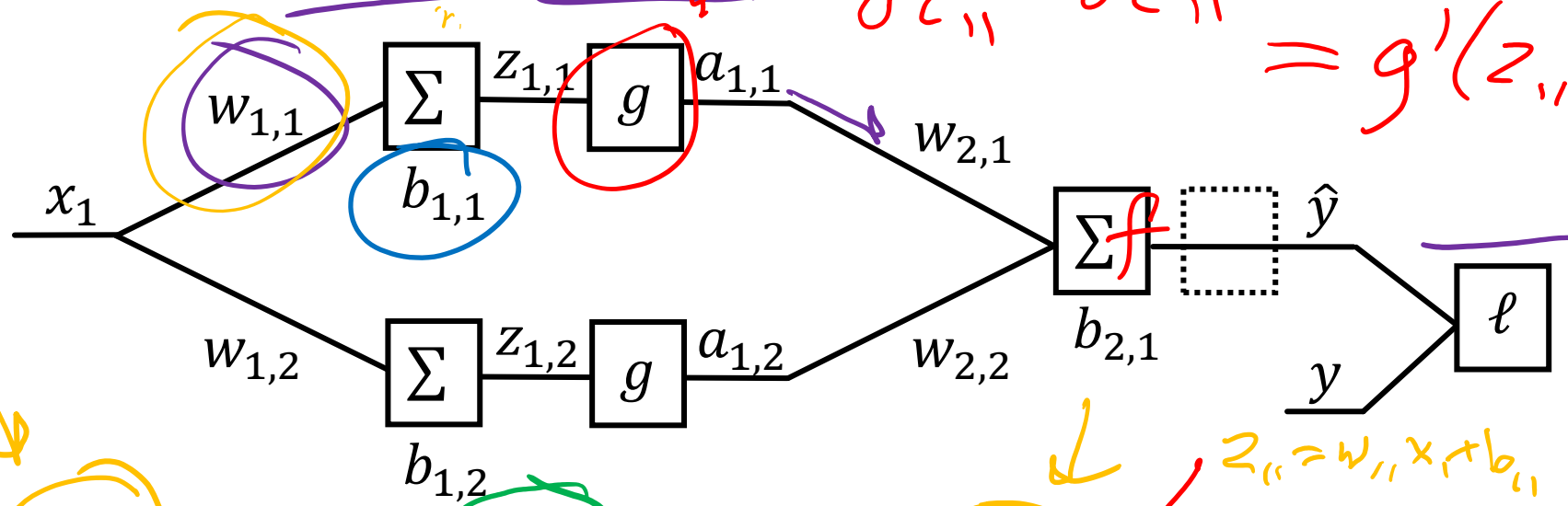
$$\frac{\partial J}{\partial b_{2,1}} =$$

Optimization

Three-neuron network

$$\hat{y} = h_{\boldsymbol{\theta}}(\mathbf{x}) = b_2 + \sum_j w_{2,j}\, a_{1,j}$$

$$a_{1,j} = g\left(b_{1,j} + w_{1,j}\, x_1\right)$$



$$\frac{\partial J}{\partial w_{1,1}} = \frac{\partial \ell}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_{1,1}} \frac{\partial a_{1,1}}{\partial z_{1,1}} \frac{\partial z_{1,1}}{\partial w_{1,1}}$$

$$\frac{\partial J}{\partial b_{1,1}} = \frac{\partial \ell}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_{1,1}} \frac{\partial a_{1,1}}{\partial z_{1,1}} \frac{\partial z_{1,1}}{\partial b_{1,1}}$$

$$\frac{\partial J}{\partial w_{2,1}} = \frac{\partial \ell}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial w_{2,1}}$$

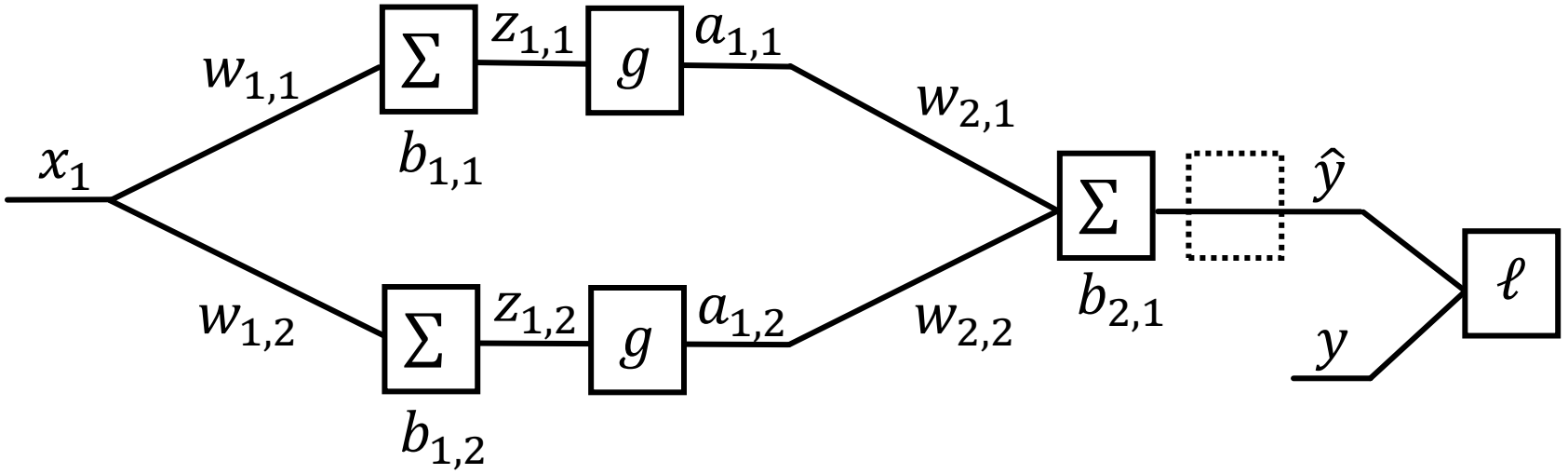
$$\frac{\partial J}{\partial b_{2,1}} = \frac{\partial \ell}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial b_{2,1}}$$

Optimization

Three-neuron network

$$\hat{y} = h_{\boldsymbol{\theta}}(\mathbf{x}) = b_2 + \sum_j w_{2,j}\, a_{1,j}$$

$$a_{1,j} = g\left(b_{1,j} + w_{1,j}\, x_1\right)$$



$$\frac{\partial J}{\partial w_{1,j}} = \frac{\partial \ell}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_{1,j}} \frac{\partial a_{1,j}}{\partial z_{1,j}} \frac{\partial z_{1,j}}{\partial w_{1,j}}$$

$$\frac{\partial J}{\partial w_{2,j}} = \frac{\partial \ell}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial w_{2,j}}$$

$$\frac{\partial J}{\partial b_{1,j}} = \frac{\partial \ell}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial a_{1,j}} \frac{\partial a_{1,j}}{\partial z_{1,j}} \frac{\partial z_{1,j}}{\partial b_{1,j}}$$

$$\frac{\partial J}{\partial b_{2,1}} = \frac{\partial \ell}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial b_{2,1}}$$

Adding More Neurons

Poll 4

Logistic regression for 28x28=784 pixel hand-written digit images into 10 classes:

How many parameters (including bias terms)?

A. 10
B. 10+784
C. 10*784
D. 10*784 + 10
E. 10*784 + 784
F. I have no idea

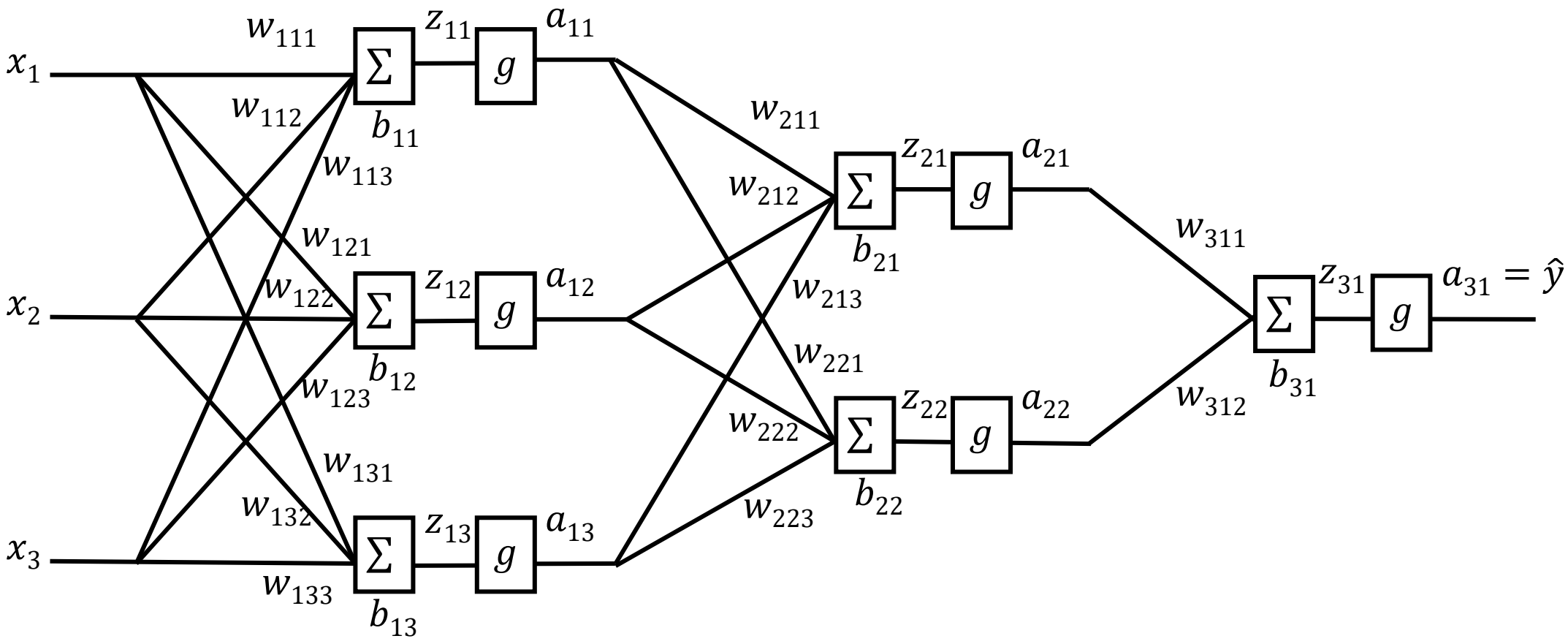
Three-layer Network

Parameter naming conventions

$w_{layer\ output\ input}$

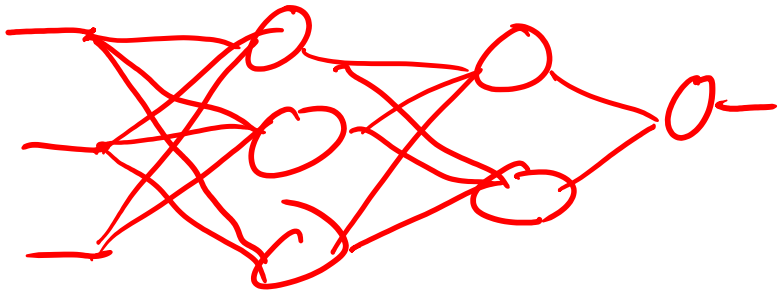
$b_{layer\ output}$

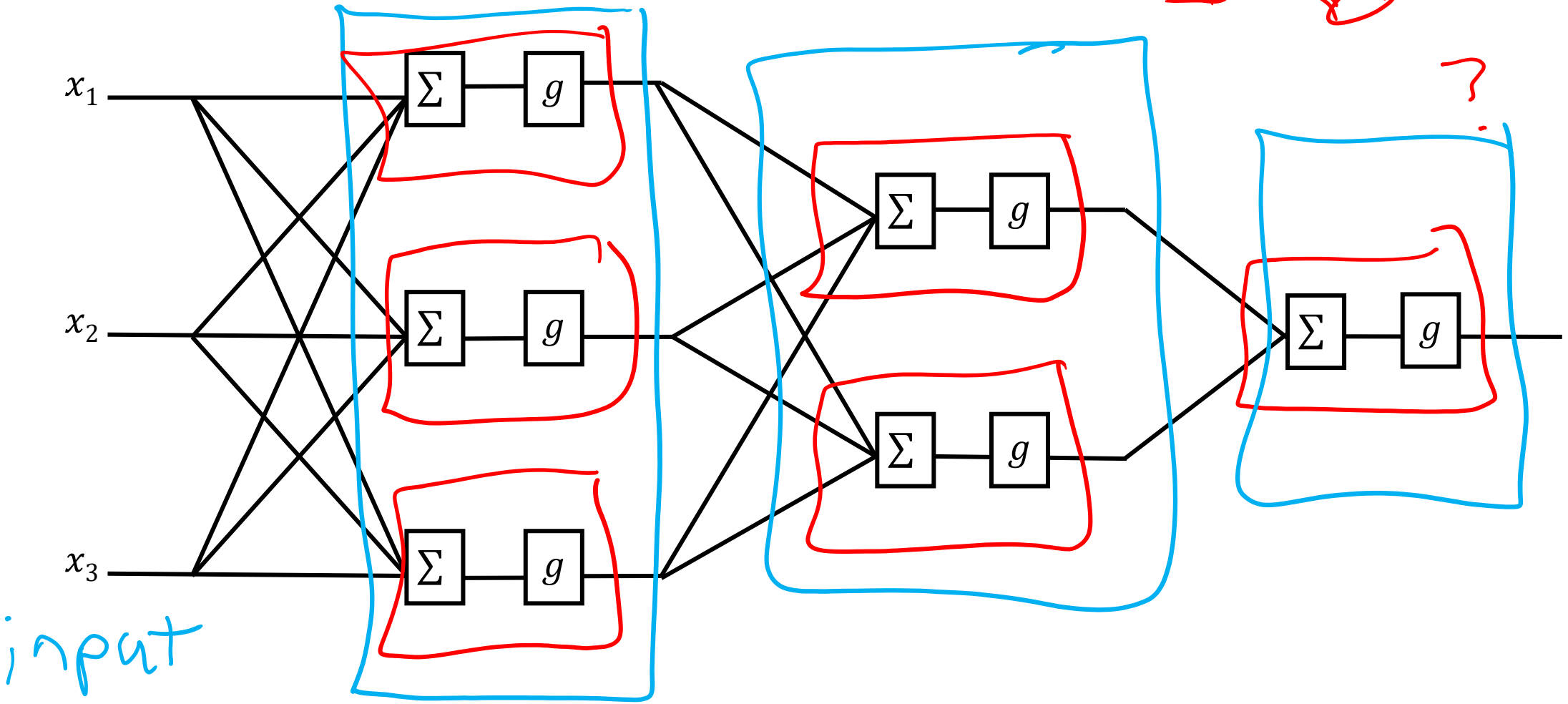
Network diagram



Three-layer Network

Different ways to define network layers





Three-layer Network

Different ways to define network layers

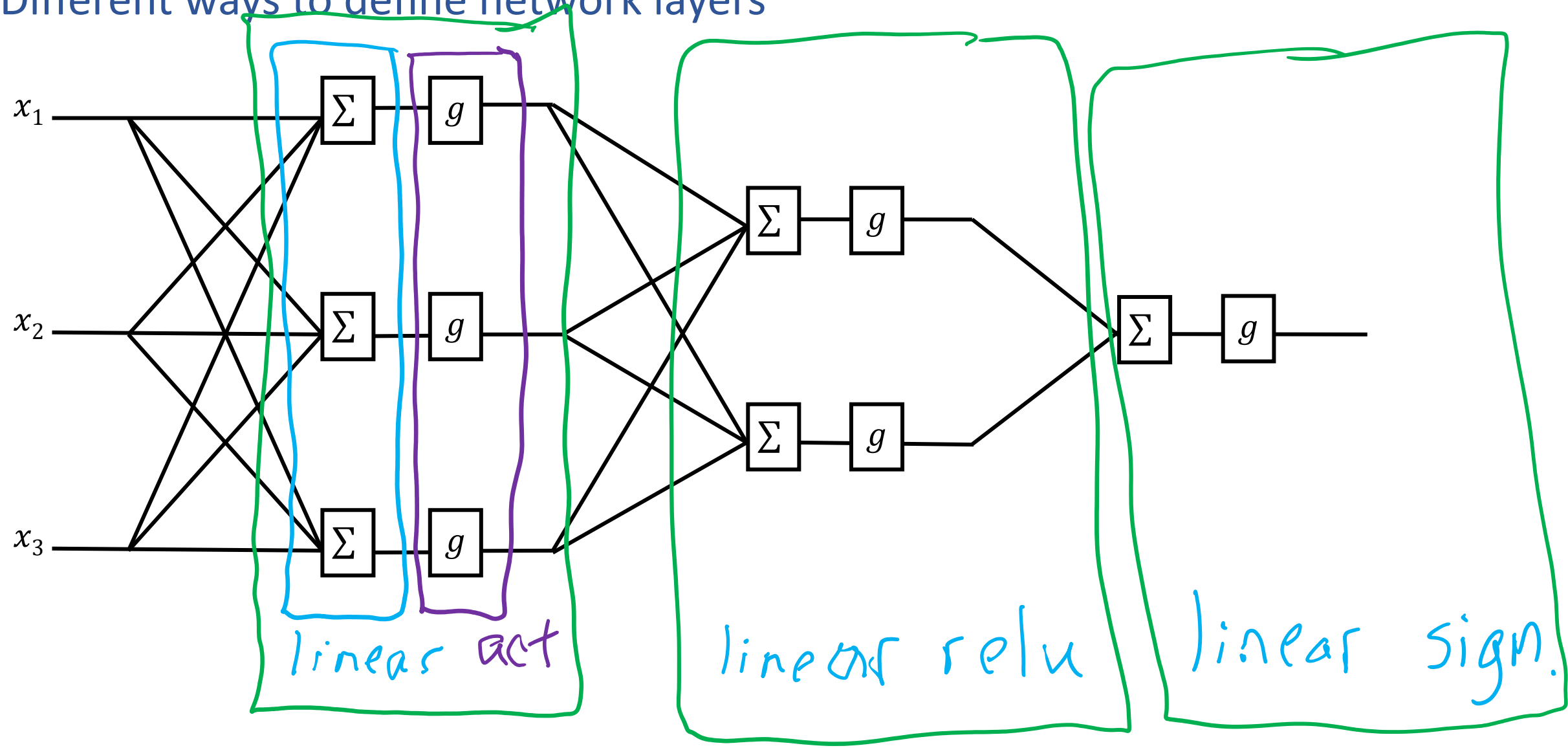
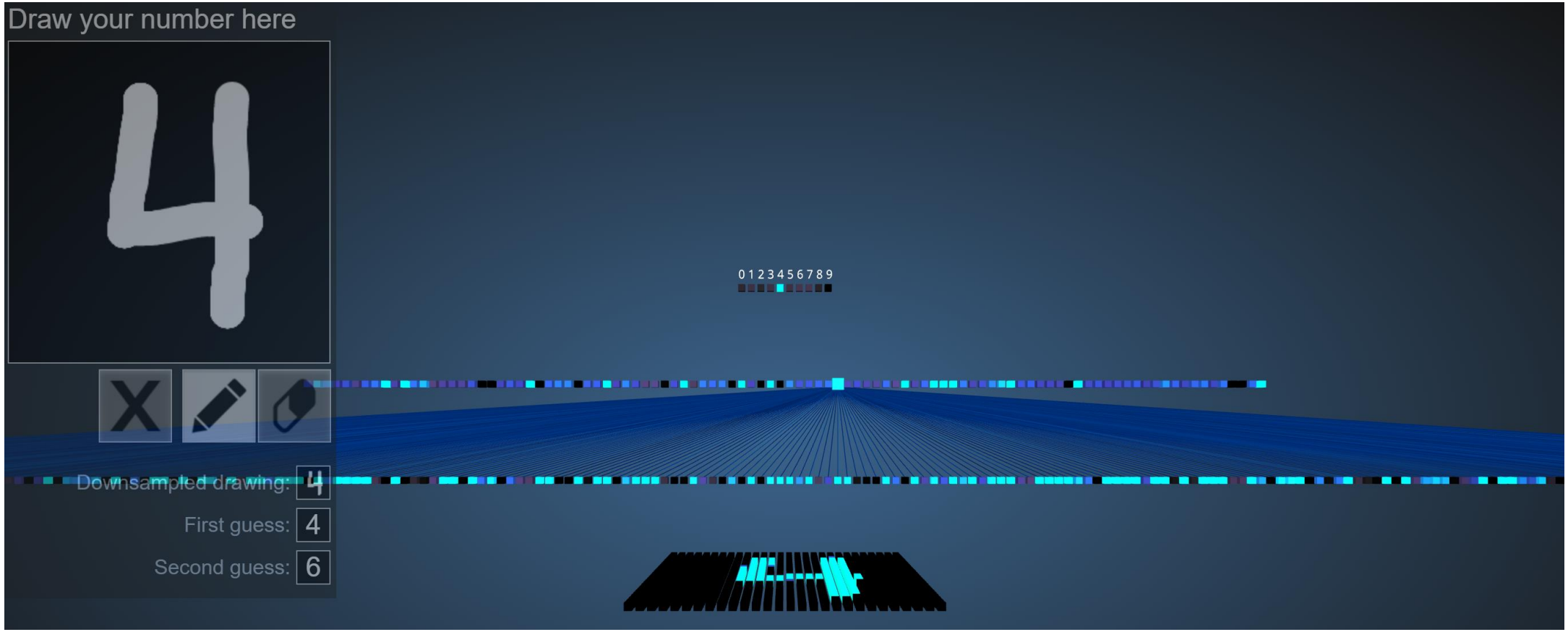


Image Classification

Demo of (Fully-connected) neural network to classify images of hand-written digits

https://adamharley.com/nn_vis/mlp/3d.html



Reminder: Image Classification

Demo of (Fully-connected) neural network to classify images of hand-written digits

https://adamharley.com/nn_vis/mlp/3d.html

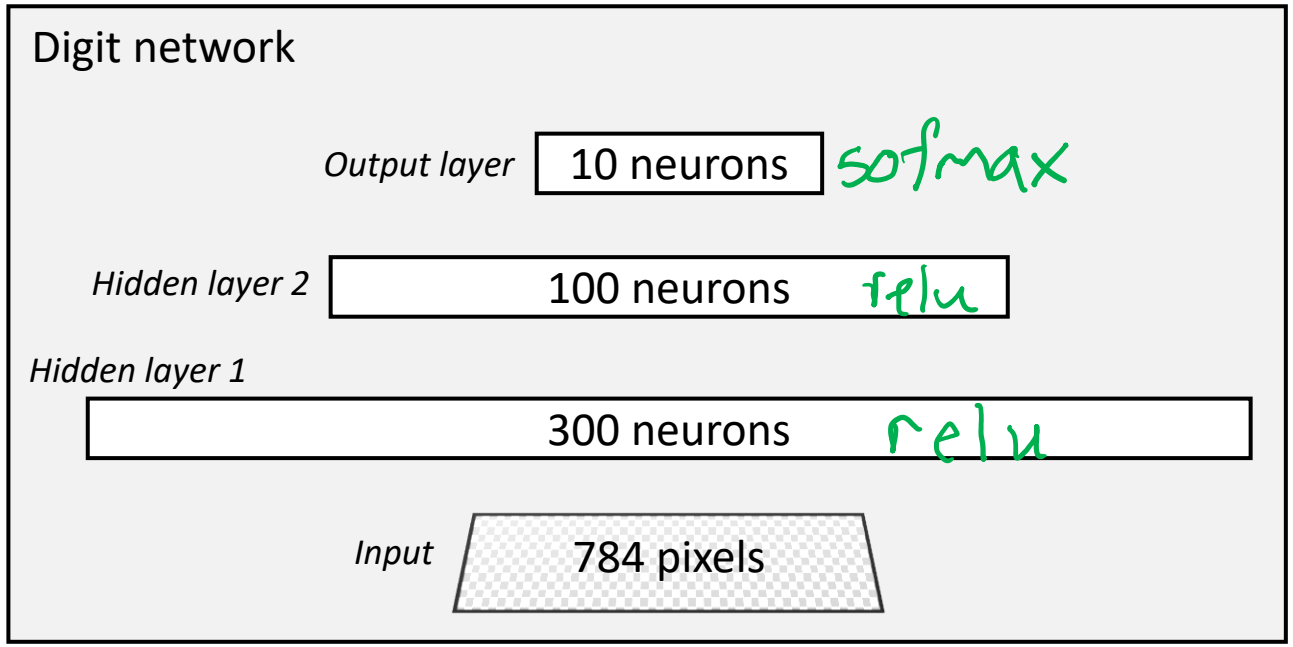
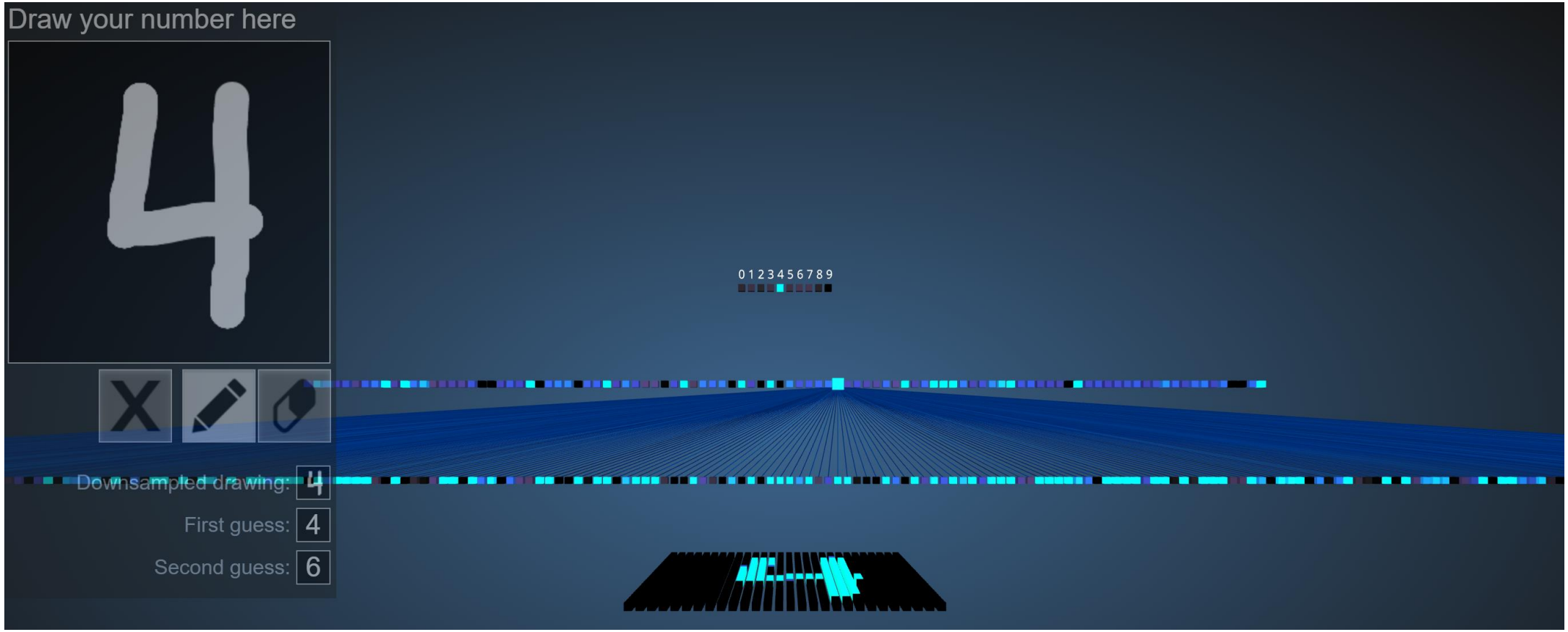


Image Classification

Demo of (Fully-connected) neural network to classify images of hand-written digits

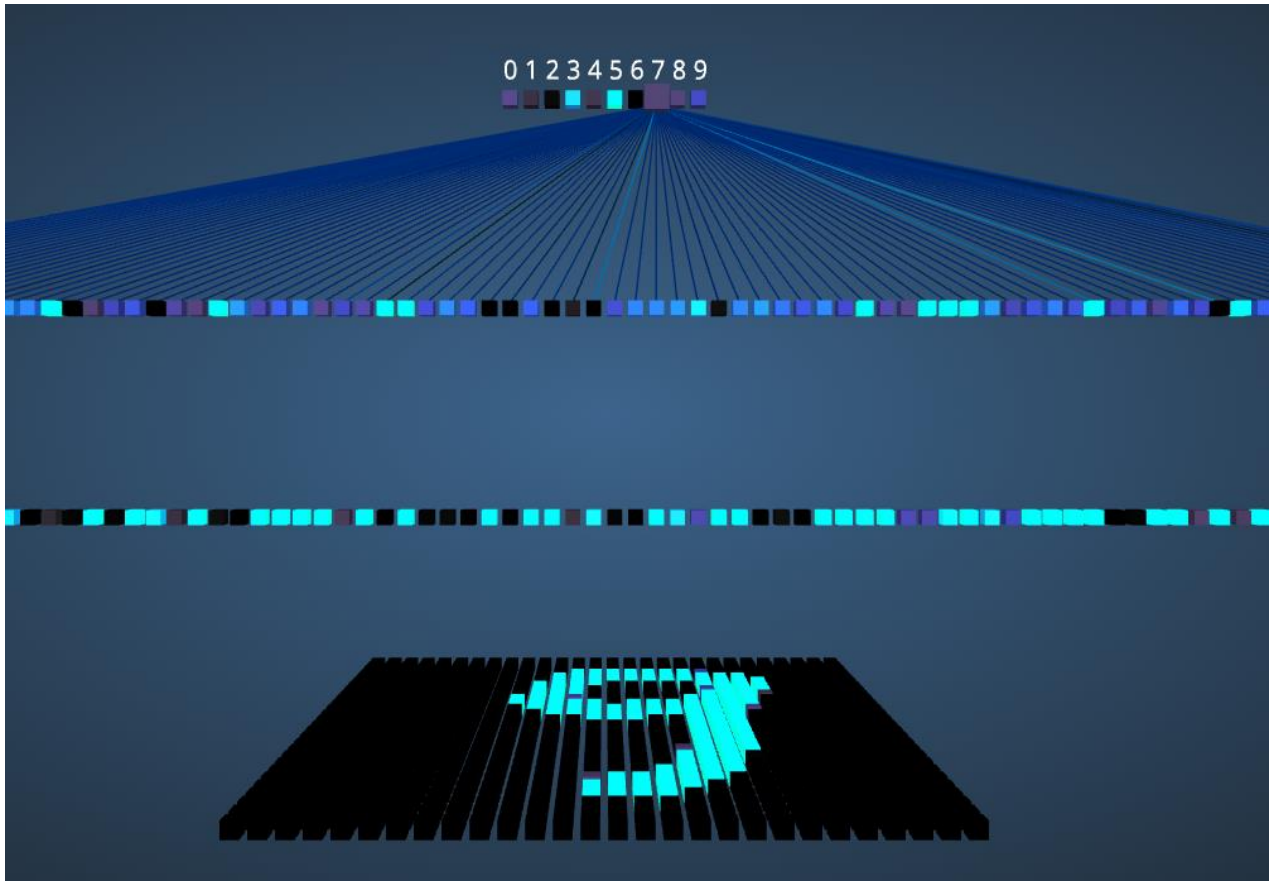
https://adamharley.com/nn_vis/mlp/3d.html



Poll 5

How many parameters does one neuron in the output layer have?

A. 1
B. 2
C. 10
D. 11
E. 100
F. 101



https://adamharley.com/nn_vis/mlp/3d.html

Poll 6

How many parameters does one neuron in the first hidden layer have?

A. 1

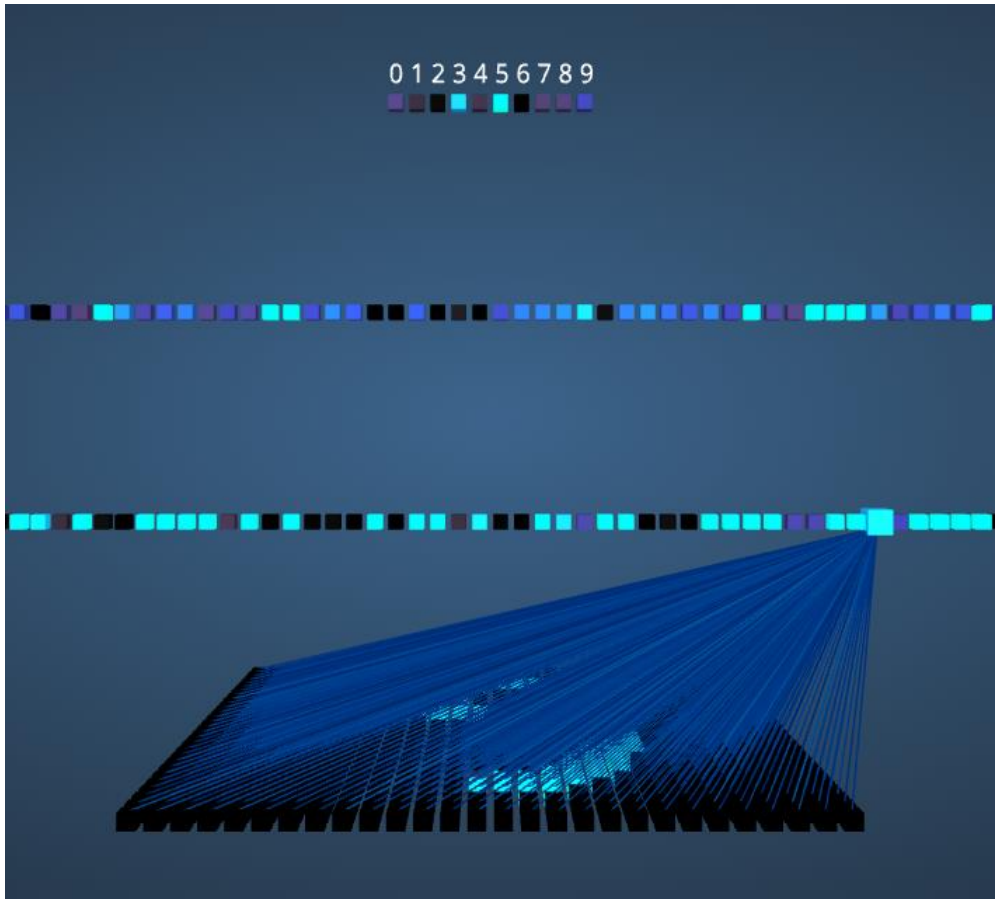
B. 2

C. 300

D. 301

E. 784

F. 785



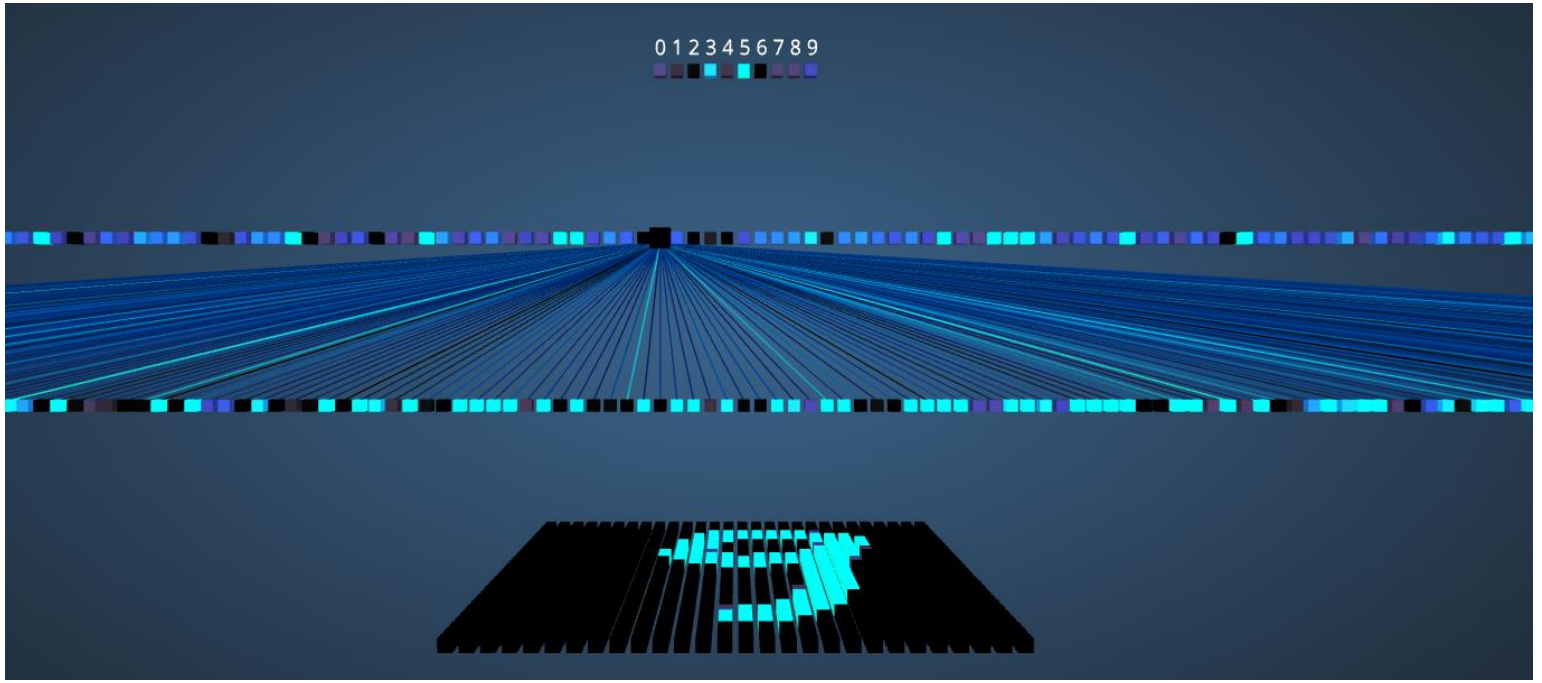
https://adamharley.com/nn_vis/mlp/3d.html

Poll 7

What do the colors of the lines represent?

A. Weight

B. Value from previous neuron

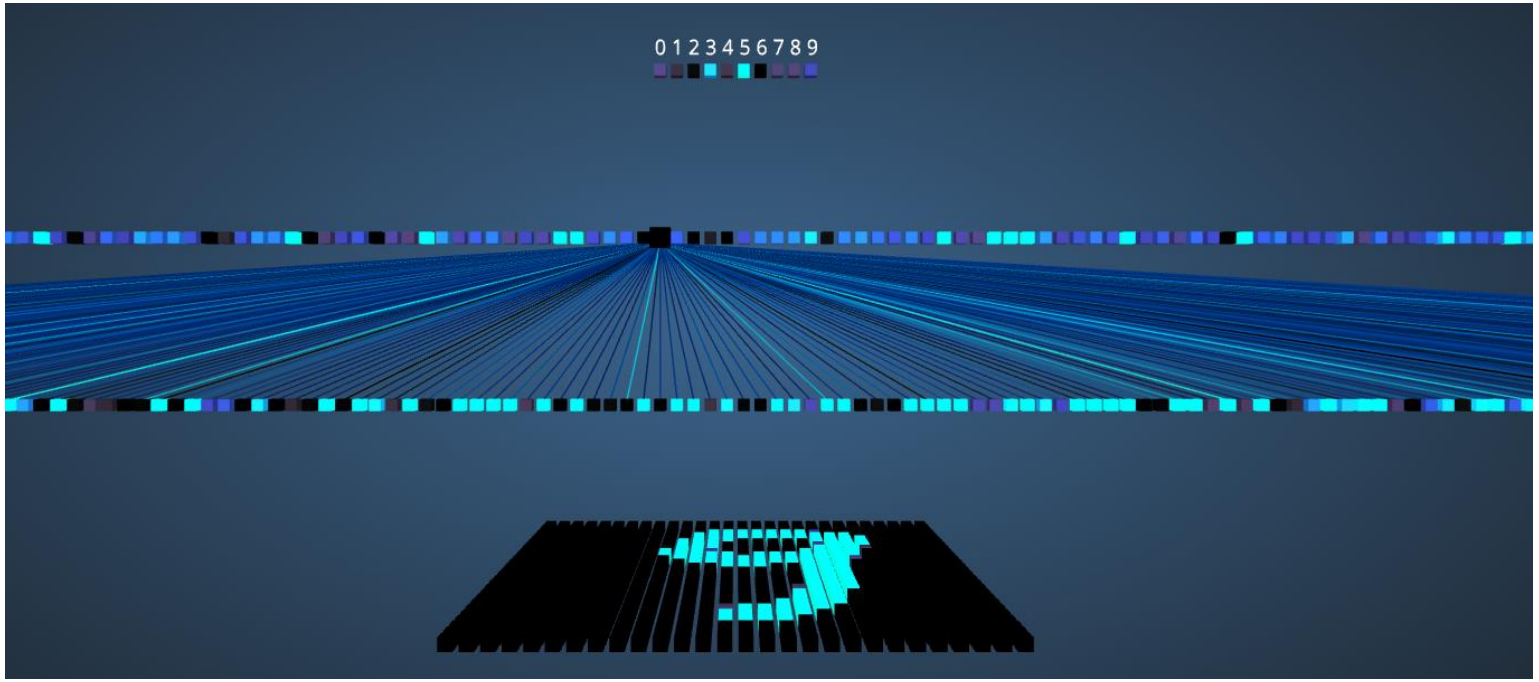


https://adamharley.com/nn_vis/mlp/3d.html

Poll 8

What do the colors of the neuron squares represent?

A. Weight
B. Output value of the neuron
C. Input value of the neuron



https://adamharley.com/nn_vis/mlp/3d.html

Neural Net Forward Pass and Backpropagation

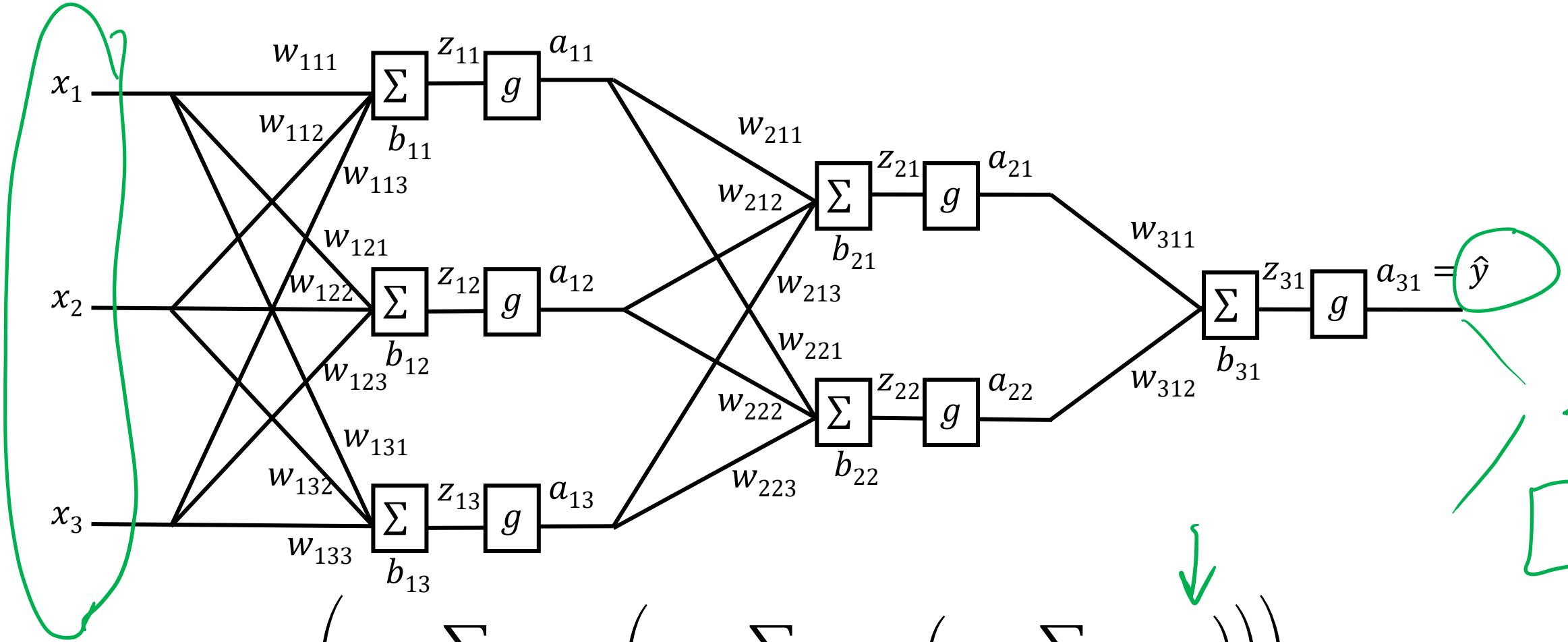
Forward pass

Parameter naming conventions

$w_{layer\ output\ input}$

$b_{layer\ output}$

Predicting output from input



$$\hat{y} = h_{\boldsymbol{\theta}}(\mathbf{x}) = g\left(b_{3,1} + \sum_k w_{3,1,k}\ g\left(b_{2,k} + \sum_i w_{2,k,i}\ g\left(b_{1,i} + \sum_j w_{1,i,j}\ x_j\right)\right)\right)$$

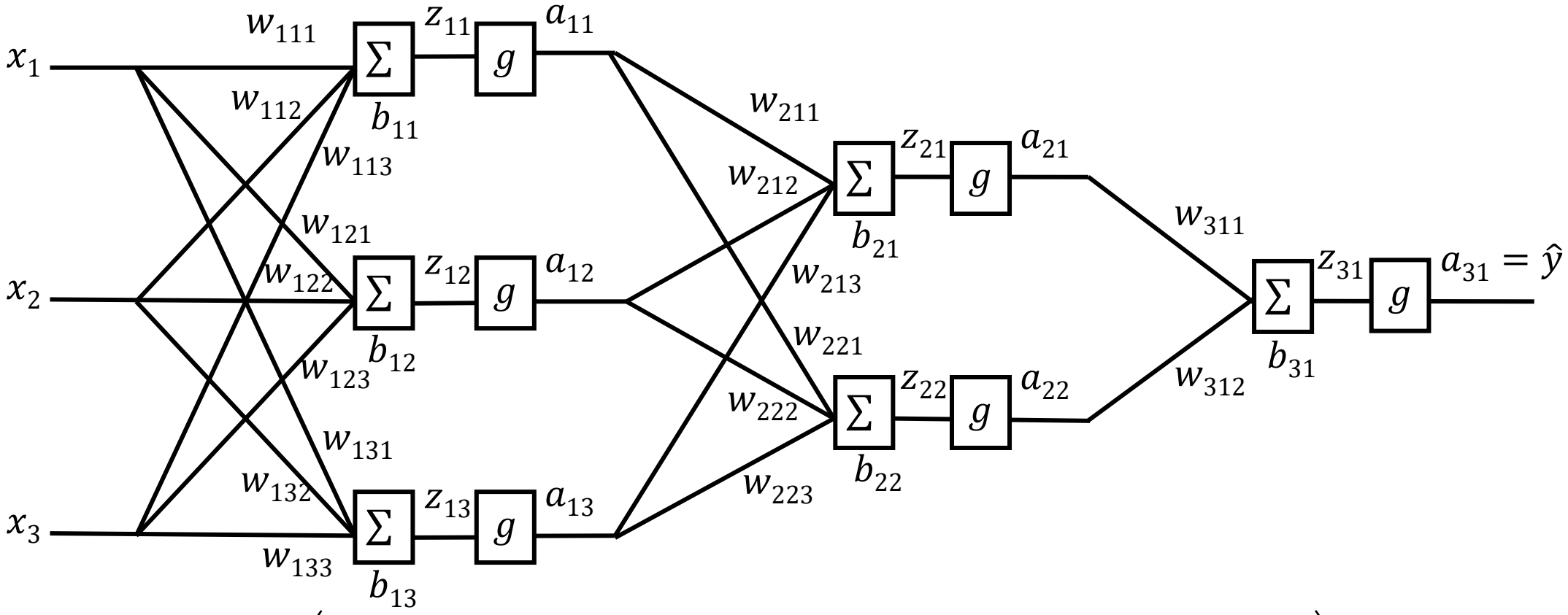
Three-layer Network

Parameter naming conventions

$w_{layer\ output\ input}$

$b_{layer\ output}$

Network diagram

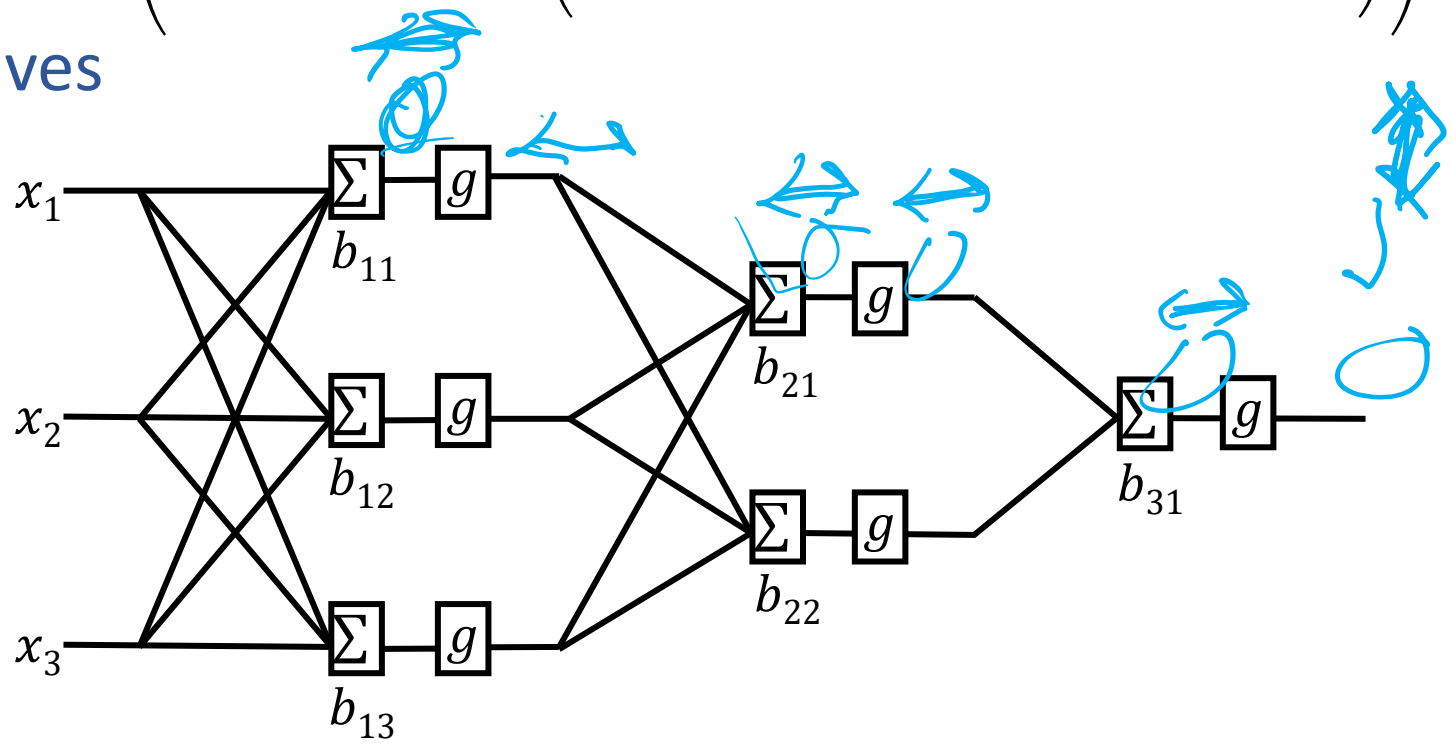


$$\hat{y} = h_{\boldsymbol{\theta}}(\mathbf{x}) = g\left(b_{3,1} + \sum_k w_{3,1,k}\ g\left(b_{2,k} + \sum_i w_{2,k,i}\ g\left(b_{1,i} + \sum_j w_{1,i,j}\ x_j\right)\right)\right)$$

Optimization

$$\hat{y} = h_{\boldsymbol{\theta}}(\mathbf{x}) = g\left(b_{3,1} + \sum_k w_{3,1,k}\; g\left(b_{2,k} + \sum_i w_{2,k,i}\; g\left(b_{1,i} + \sum_j w_{1,i,j}\; x_j\right)\right)\right)$$

Tons of repeated partial derivatives



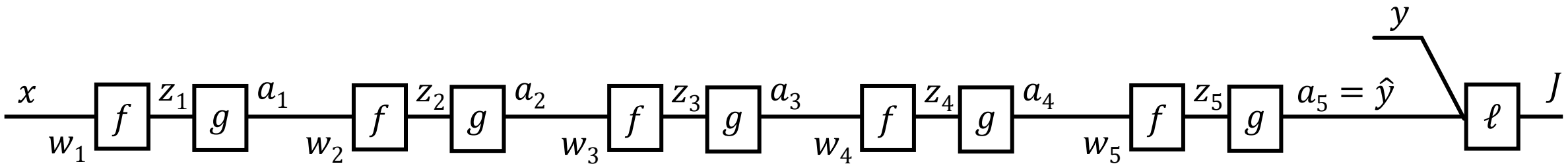
$$\frac{\partial J}{\partial b_{3,1}} = \frac{\partial \ell}{\partial \hat{y}} \frac{\partial a_{3,1}}{\partial z_{3,1}} \frac{\partial z_{3,1}}{\partial b_{3,1}}$$

$$\frac{\partial J}{\partial b_{2,i}} = \frac{\partial \ell}{\partial \hat{y}} \frac{\partial a_{3,1}}{\partial z_{3,1}} \frac{\partial z_{3,1}}{\partial a_{2,i}} \frac{\partial a_{2,i}}{\partial z_{2,i}} \frac{\partial z_{2,i}}{\partial b_{2,i}}$$

$$\frac{\partial J}{\partial b_{1,i}} = \frac{\partial \ell}{\partial \hat{y}} \frac{\partial a_{3,1}}{\partial z_{3,1}} \frac{\partial z_{3,1}}{\partial \mathbf{a}_2} \frac{\partial \mathbf{a}_2}{\partial \mathbf{z}_2} \frac{\partial \mathbf{z}_2}{\partial a_{1,i}} \frac{\partial a_{1,i}}{\partial z_{1,i}} \frac{\partial z_{1,i}}{\partial b_{1,i}}$$

Forward Pass

Width 1 deep network (no bias) (dumb but will help with calculus)



$$z_\ell = f(w_\ell, a_{\ell-1}) = w_\ell \cdot a_{\ell-1}$$

$$a_\ell = g(z_\ell)$$

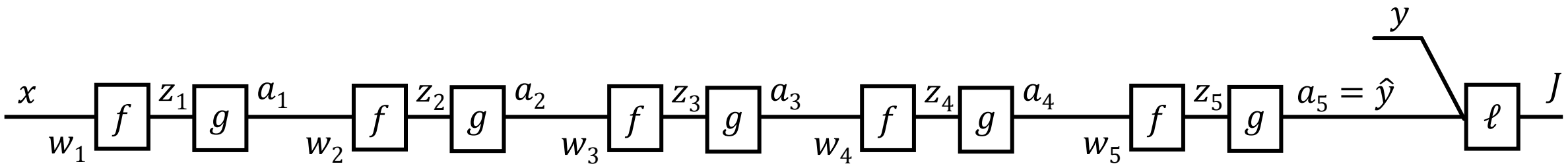
$$\hat{y} = h_{\boldsymbol{\theta}}(\mathbf{x}) = g\left(w_5 \cdot g\left(w_4 \cdot g\left(w_3 \cdot g\left(w_2 \cdot g(w_1 \cdot x)\right)\right)\right)\right)$$

$$= g\left(f\left(g\left(f\left(g\left(f\left(g\left(f\left(g(f(x))\right)\right)\right)\right)\right)\right)\right)\right)$$

Forward Pass

$$\hat{y} = h_{\boldsymbol{\theta}}(\mathbf{x}) = g\left(w_5 \cdot g\left(w_4 \cdot g\left(w_3 \cdot g\left(w_2 \cdot g(w_1 \cdot x)\right)\right)\right)\right)$$

Width 1 deep network (no bias) (dumb but will help with calculus)



With a new data point (x, y), we have our current weight values

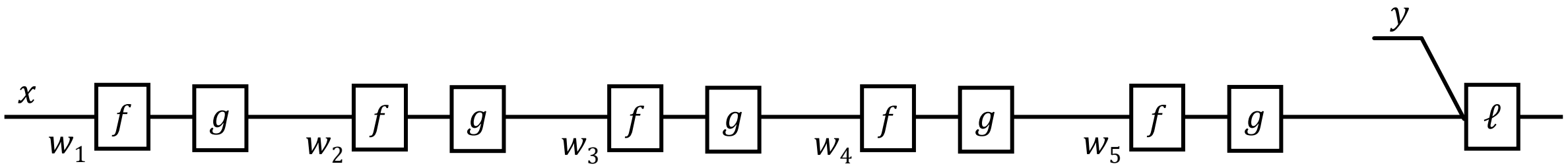
but we don't $\hat{y}$ (or any of the intermediate values z_ℓ and a_ℓ)

or the value of our object function J

Forward Pass

$$\hat{y} = h_{\boldsymbol{\theta}}(\mathbf{x}) = g\left(w_5 \cdot g\left(w_4 \cdot g\left(w_3 \cdot g\left(w_2 \cdot g(w_1 \cdot x)\right)\right)\right)\right)$$

Width 1 deep network (no bias) (dumb but will help with calculus)



With a new data point (x, y), we have our current weight values

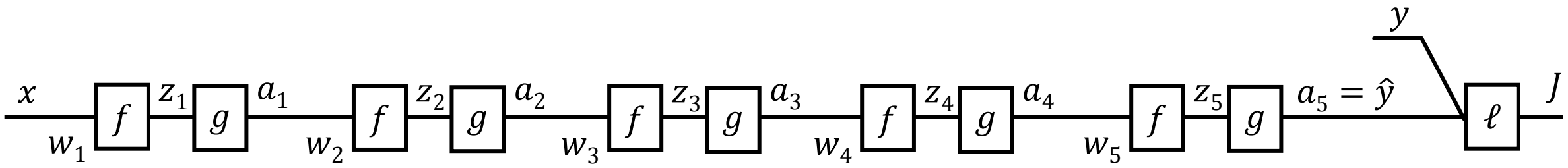
but we don't $\hat{y}$ (or any of the intermediate values z_ℓ and a_ℓ)

or the value of our object function J

Forward Pass

$$\hat{y} = h_{\boldsymbol{\theta}}(\mathbf{x}) = g\left(w_5 \cdot g\left(w_4 \cdot g\left(w_3 \cdot g\left(w_2 \cdot g(w_1 \cdot x)\right)\right)\right)\right)$$

Width 1 deep network (no bias) (dumb but will help with calculus)



With a new data point (x, y), we have our current weight values

but we don't have $\hat{y}$ (or any of the intermediate values z_ℓ and a_ℓ)

or the value of our object function J

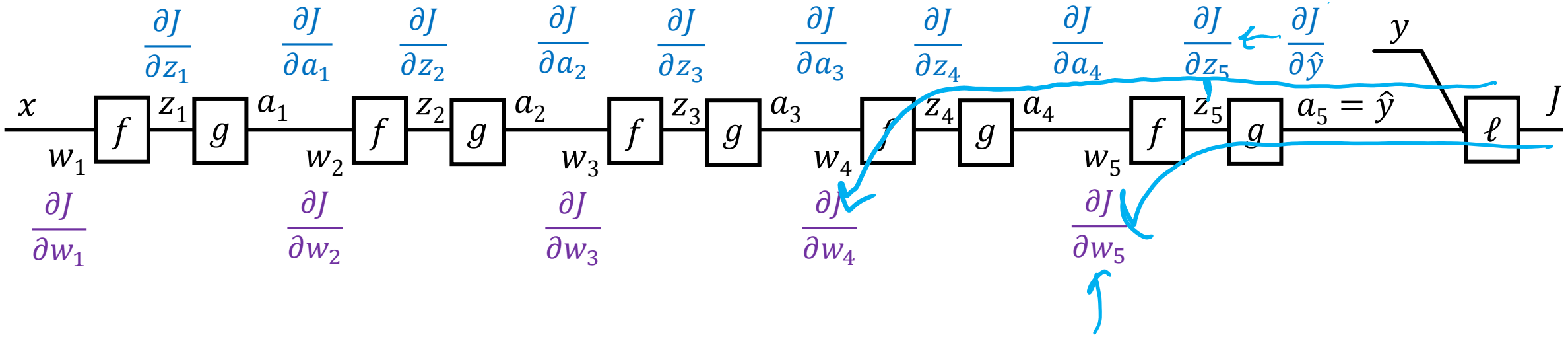
The forward pass propagates x (forward) through the network

to give us these values

Backpropagation

$$\hat{y} = h_{\boldsymbol{\theta}}(\mathbf{x}) = g\left(w_5 \cdot g\left(w_4 \cdot g\left(w_3 \cdot g\left(w_2 \cdot g\left(w_1 \cdot x\right)\right)\right)\right)\right)$$

Width 1 deep network (no bias) (dumb but will help with calculus)



To do gradient descent we need the partial derivative of the objective with respect to each parameter, $w_\ell \leftarrow w_\ell - \alpha\, \partial J / \partial w_\ell$

The backward pass propagates the change in the objective with respect to intermediate values ($\partial J / \partial z_\ell$ and $\partial J / \partial a_\ell$) back through the network to produce each $\partial J / \partial w_\ell$

Reminder: Calculus Chain Rule (scalar version)

Composite functions → chain rule

$$y = f(\mathrm{g}(\mathrm{x}))$$

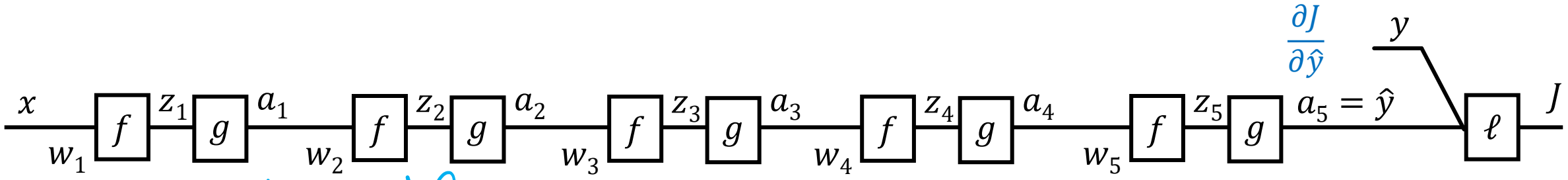
$$y = f(z)$$
$$z = g(x)$$

$$\frac{df}{dx} = \frac{df}{dz}\frac{dg}{dx}$$

Why backwards?

$$\hat{y} = h_{\boldsymbol{\theta}}(\mathbf{x}) = g\left(w_5 \cdot g\left(w_4 \cdot g\left(w_3 \cdot g\left(w_2 \cdot g(w_1 \cdot x)\right)\right)\right)\right)$$

Width 1 deep network (no bias) (dumb but will help with calculus)

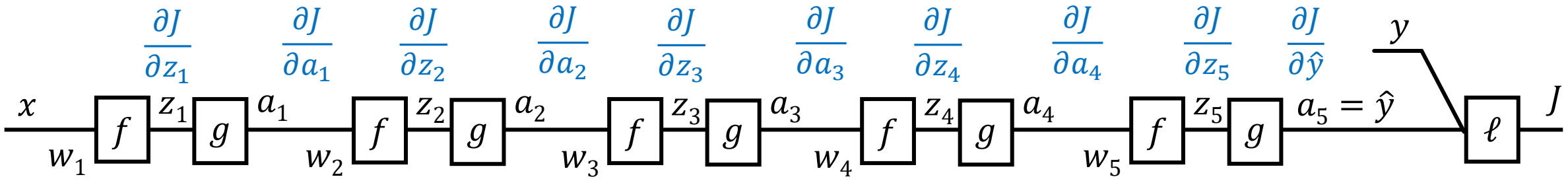


$$\frac{\partial J}{\partial w_1} = \frac{\partial \ell}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z_5} \frac{\partial f}{a_4} \cdots$$

Why backwards?

$$\hat{y} = h_{\boldsymbol{\theta}}(\mathbf{x}) = g\left(w_5 \cdot g\left(w_4 \cdot g\left(w_3 \cdot g\left(w_2 \cdot g(w_1 \cdot x)\right)\right)\right)\right)$$

Width 1 deep network (no bias) (dumb but will help with calculus)



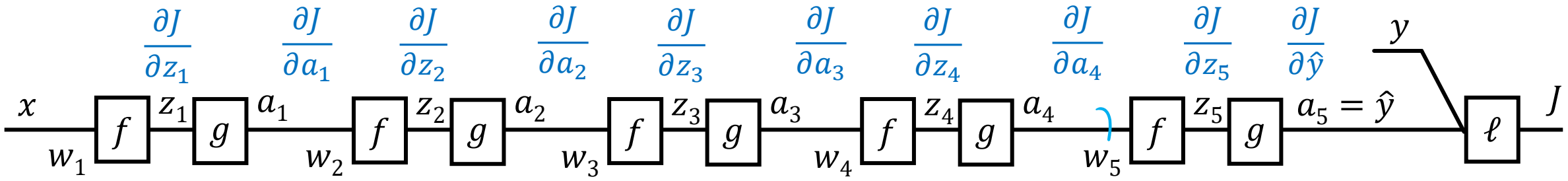
$$\frac{\partial J}{\partial w_1} = \frac{\partial J}{\partial \hat{y}} \frac{\partial g}{\partial z_5} \frac{\partial f}{\partial a_4} \frac{\partial g}{\partial z_4} \frac{\partial f}{\partial a_3} \frac{\partial g}{\partial z_3} \frac{\partial f}{\partial a_2} \frac{\partial g}{\partial z_2} \frac{\partial f}{\partial a_1} \frac{\partial g}{\partial z_1} \frac{\partial f}{\partial w_1}$$

$$\frac{\partial J}{\partial w_2} = \frac{\partial \ell}{\partial \hat{y}} \frac{\partial g}{\partial z_5} \frac{\partial f}{\partial a_4} \cdots$$

Why backwards?

$$\hat{y} = h_{\boldsymbol{\theta}}(\mathbf{x}) = g\left(w_5 \cdot g\left(w_4 \cdot g\left(w_3 \cdot g\left(w_2 \cdot g(w_1 \cdot x)\right)\right)\right)\right)$$

Width 1 deep network (no bias) (dumb but will help with calculus)



$$\frac{\partial J}{\partial w_1} = \frac{\partial J}{\partial \hat{y}}\frac{\partial g}{\partial z_5}\frac{\partial f}{\partial a_4}\frac{\partial g}{\partial z_4}\frac{\partial f}{\partial a_3}\frac{\partial g}{\partial z_3}\frac{\partial f}{\partial a_2}\frac{\partial g}{\partial z_2}\frac{\partial f}{\partial a_1}\frac{\partial g}{\partial z_1}\frac{\partial f}{\partial w_1}$$

$$\frac{\partial J}{\partial w_2} = \frac{\partial J}{\partial \hat{y}}\frac{\partial g}{\partial z_5}\frac{\partial f}{\partial a_4}\frac{\partial g}{\partial z_4}\frac{\partial f}{\partial a_3}\frac{\partial g}{\partial z_3}\frac{\partial f}{\partial a_2}\frac{\partial g}{\partial z_2}\frac{\partial f}{\partial w_2}$$

$$\frac{\partial J}{\partial w_3} = \frac{\partial J}{\partial \hat{y}}\frac{\partial g}{\partial z_5}\frac{\partial f}{\partial a_4}\frac{\partial g}{\partial z_4}\frac{\partial f}{\partial a_3}\frac{\partial g}{\partial z_3}\frac{\partial f}{\partial w_3}$$

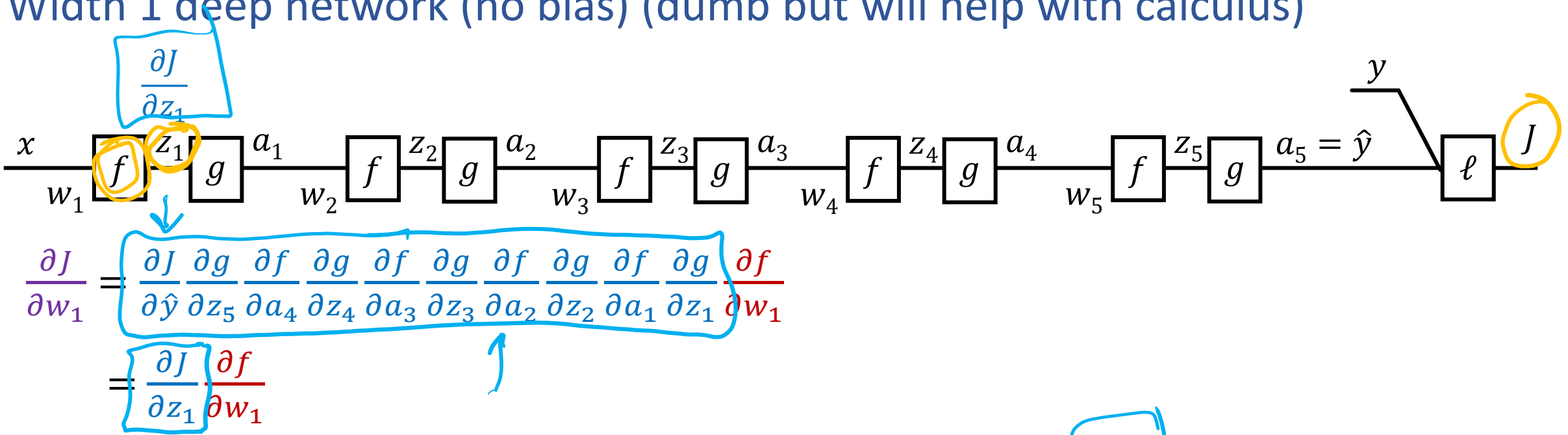
$$\frac{\partial J}{\partial w_4} = \frac{\partial J}{\partial \hat{y}}\frac{\partial g}{\partial z_5}\frac{\partial f}{\partial a_4}\frac{\partial g}{\partial z_4}\frac{\partial f}{\partial w_4}$$

$$\frac{\partial J}{\partial w_5} = \frac{\partial J}{\partial \hat{y}}\frac{\partial g}{\partial z_5}\frac{\partial f}{\partial w_5}$$

Why backwards?

$$\hat{y} = h_{\boldsymbol{\theta}}(\mathbf{x}) = g\left(w_5 \cdot g\left(w_4 \cdot g\left(w_3 \cdot g\left(w_2 \cdot g(w_1 \cdot x)\right)\right)\right)\right)$$

Width 1 deep network (no bias) (dumb but will help with calculus)



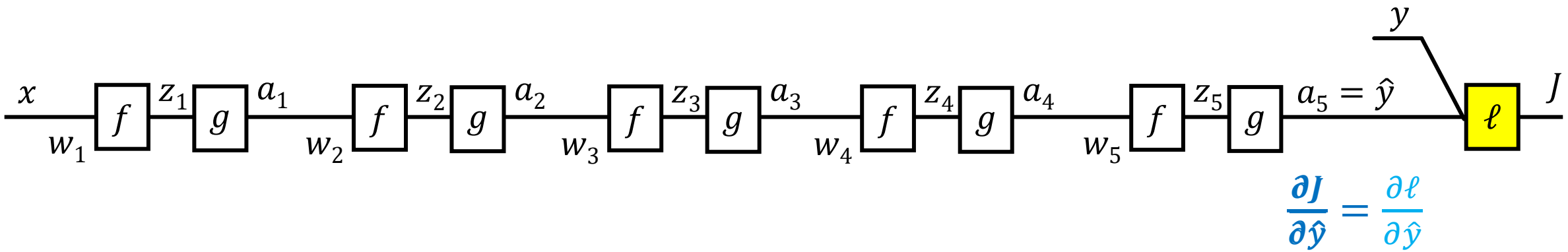
What if someone had already done all of the work to compute $\frac{\partial J}{\partial z_1}$?

Then we just need to do the local partial derivative for f with respect to the parameter w and then use that to compute the partial derivative for J with respect to the parameter w

Backpropagation

$$\hat{y} = h_{\boldsymbol{\theta}}(\mathbf{x}) = g\left(w_5 \cdot g\left(w_4 \cdot g\left(w_3 \cdot g\left(w_2 \cdot g(w_1 \cdot x)\right)\right)\right)\right)$$

More efficient to start at the end and reuse values!

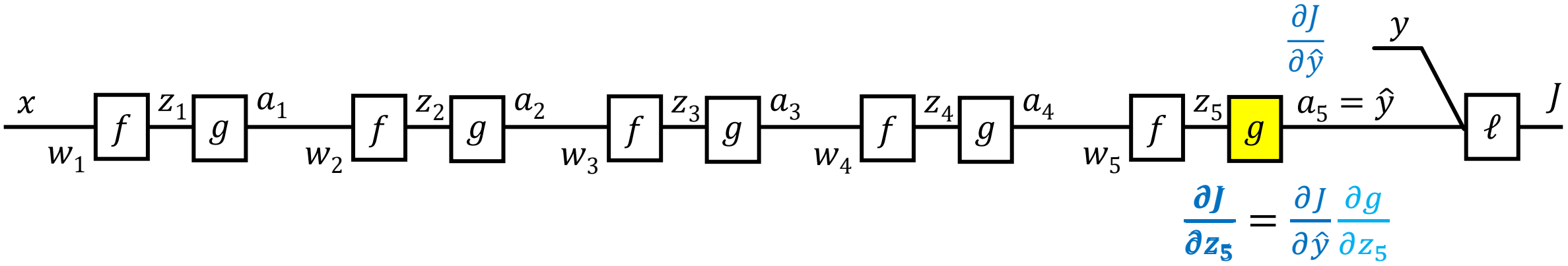


$$\frac{\partial J}{\partial \hat{y}} = \frac{\partial \ell}{\partial \hat{y}}$$

Backpropagation

$$\hat{y} = h_{\boldsymbol{\theta}}(\mathbf{x}) = g\left(w_5 \cdot g\left(w_4 \cdot g\left(w_3 \cdot g\left(w_2 \cdot g(w_1 \cdot x)\right)\right)\right)\right)$$

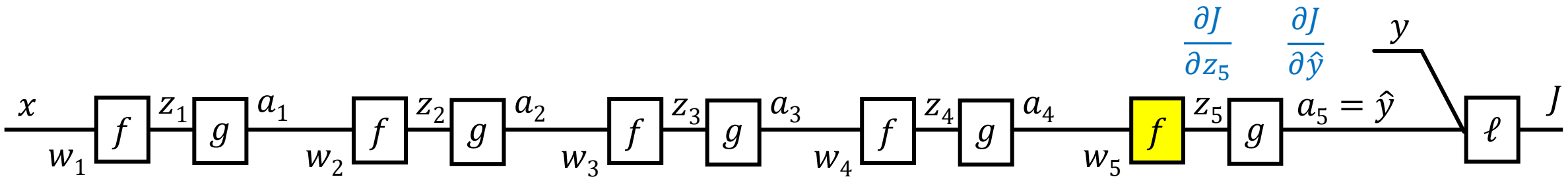
More efficient to start at the end and reuse values!



Backpropagation

$$\hat{y} = h_{\boldsymbol{\theta}}(\mathbf{x}) = g\left(w_5 \cdot g\left(w_4 \cdot g\left(w_3 \cdot g\left(w_2 \cdot g(w_1 \cdot x)\right)\right)\right)\right)$$

More efficient to start at the end and reuse values!



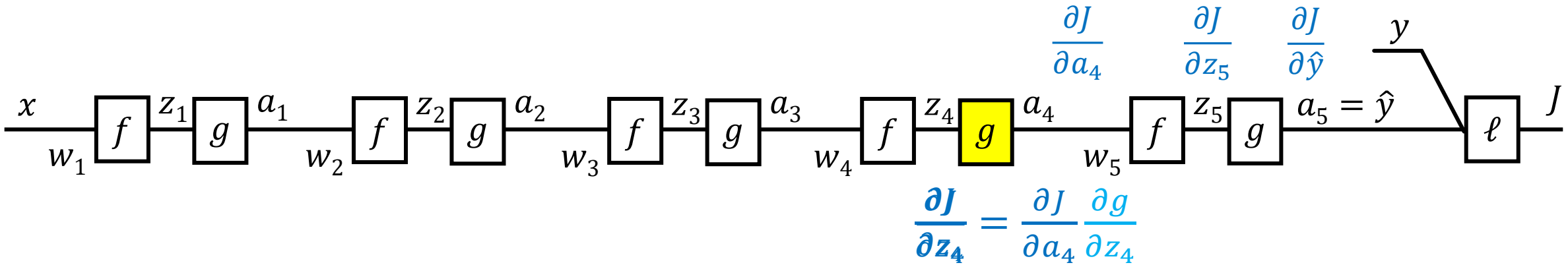
$$\frac{\partial J}{\partial a_4} = \frac{\partial J}{\partial z_5}\frac{\partial f}{\partial a_4}$$

$$\frac{\partial J}{\partial w_5} = \frac{\partial J}{\partial z_5}\frac{\partial f}{\partial w_5}$$

Backpropagation

$$\hat{y} = h_{\boldsymbol{\theta}}(\mathbf{x}) = g\left(w_5 \cdot g\left(w_4 \cdot g\left(w_3 \cdot g\left(w_2 \cdot g(w_1 \cdot x)\right)\right)\right)\right)$$

More efficient to start at the end and reuse values!

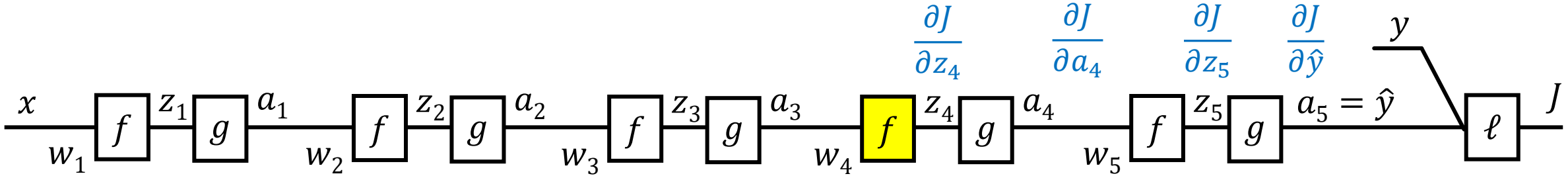


$$\frac{\partial J}{\partial z_4} = \frac{\partial J}{\partial a_4}\frac{\partial g}{\partial z_4}$$

Backpropagation

$$\hat{y} = h_{\boldsymbol{\theta}}(\mathbf{x}) = g\left(w_5 \cdot g\left(w_4 \cdot g\left(w_3 \cdot g\left(w_2 \cdot g(w_1 \cdot x)\right)\right)\right)\right)$$

More efficient to start at the end and reuse values!



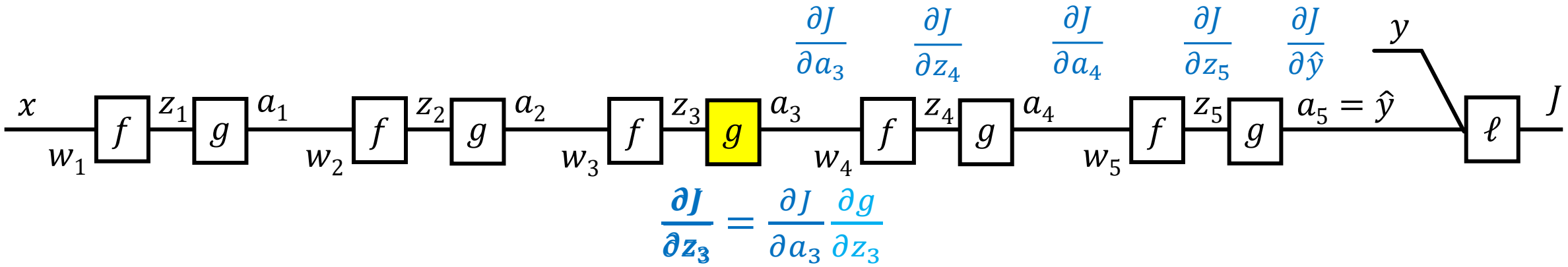
$$\frac{\partial J}{\partial a_3} = \frac{\partial J}{\partial z_4}\frac{\partial f}{\partial a_3}$$

$$\frac{\partial J}{\partial w_4} = \frac{\partial J}{\partial z_4}\frac{\partial f}{\partial w_4}$$

Backpropagation

$$\hat{y} = h_{\boldsymbol{\theta}}(\mathbf{x}) = g\left(w_5 \cdot g\left(w_4 \cdot g\left(w_3 \cdot g\left(w_2 \cdot g(w_1 \cdot x)\right)\right)\right)\right)$$

More efficient to start at the end and reuse values!

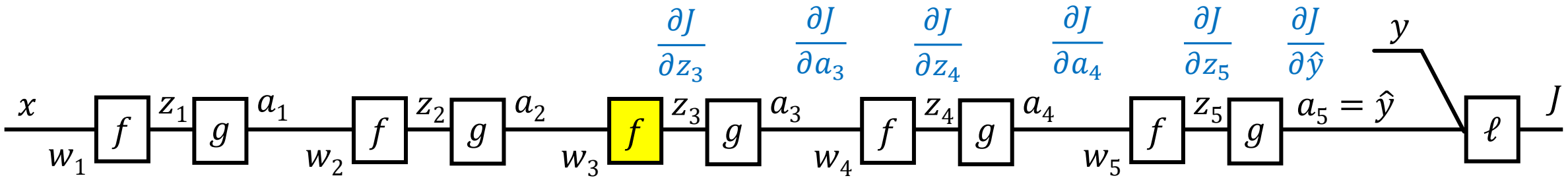


$$\frac{\partial J}{\partial z_3} = \frac{\partial J}{\partial a_3}\frac{\partial g}{\partial z_3}$$

Backpropagation

$$\hat{y} = h_{\boldsymbol{\theta}}(\mathbf{x}) = g\left(w_5 \cdot g\left(w_4 \cdot g\left(w_3 \cdot g\left(w_2 \cdot g(w_1 \cdot x)\right)\right)\right)\right)$$

More efficient to start at the end and reuse values!



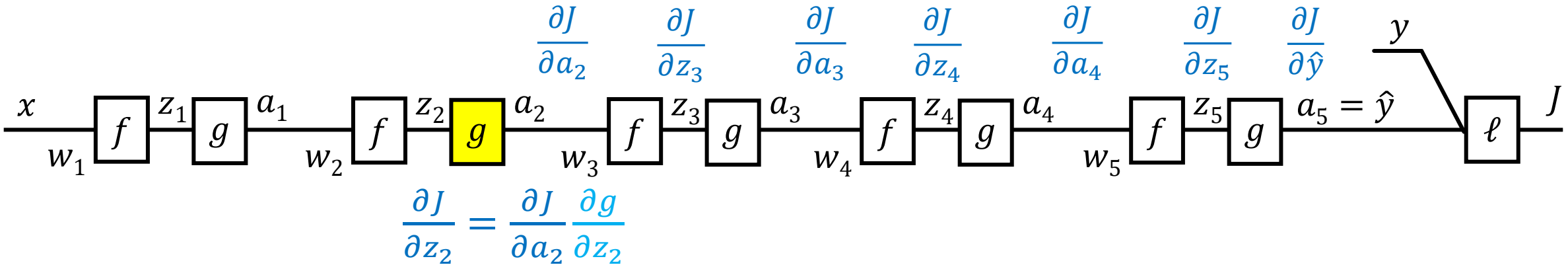
$$\frac{\partial J}{\partial a_2} = \frac{\partial J}{\partial z_3}\frac{\partial f}{\partial a_2}$$

$$\frac{\partial J}{\partial w_3} = \frac{\partial J}{\partial z_3}\frac{\partial f}{\partial w_3}$$

Backpropagation

$$\hat{y} = h_{\boldsymbol{\theta}}(\mathbf{x}) = g\left(w_5 \cdot g\left(w_4 \cdot g\left(w_3 \cdot g\left(w_2 \cdot g(w_1 \cdot x)\right)\right)\right)\right)$$

More efficient to start at the end and reuse values!

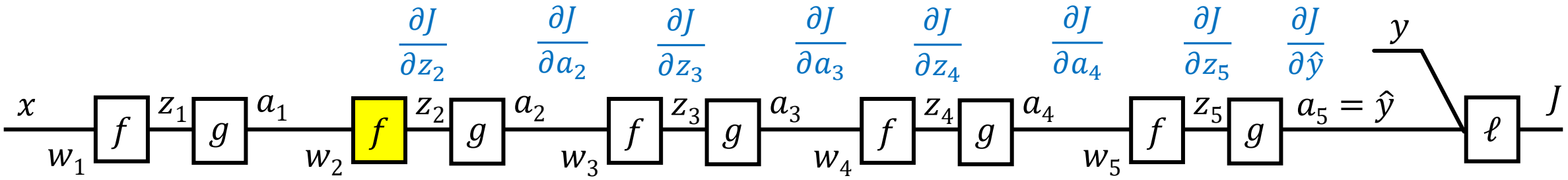


$$\frac{\partial J}{\partial z_2} = \frac{\partial J}{\partial a_2}\frac{\partial g}{\partial z_2}$$

Backpropagation

$$\hat{y} = h_{\boldsymbol{\theta}}(\mathbf{x}) = g\left(w_5 \cdot g\left(w_4 \cdot g\left(w_3 \cdot g\left(w_2 \cdot g(w_1 \cdot x)\right)\right)\right)\right)$$

More efficient to start at the end and reuse values!



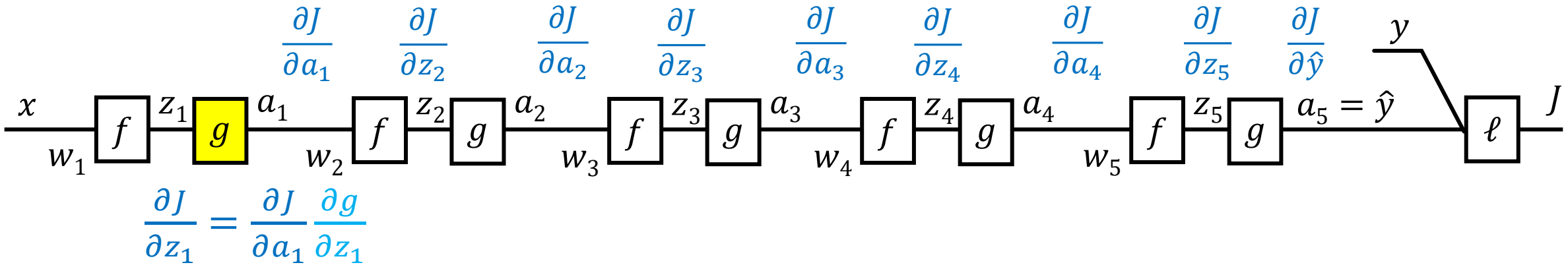
$$\frac{\partial J}{\partial a_1} = \frac{\partial J}{\partial z_2}\frac{\partial f}{\partial a_1}$$

$$\frac{\partial J}{\partial w_2} = \frac{\partial J}{\partial z_2}\frac{\partial f}{\partial w_2}$$

Backpropagation

$$\hat{y} = h_{\boldsymbol{\theta}}(\mathbf{x}) = g\left(w_5 \cdot g\left(w_4 \cdot g\left(w_3 \cdot g\left(w_2 \cdot g(w_1 \cdot x)\right)\right)\right)\right)$$

More efficient to start at the end and reuse values!

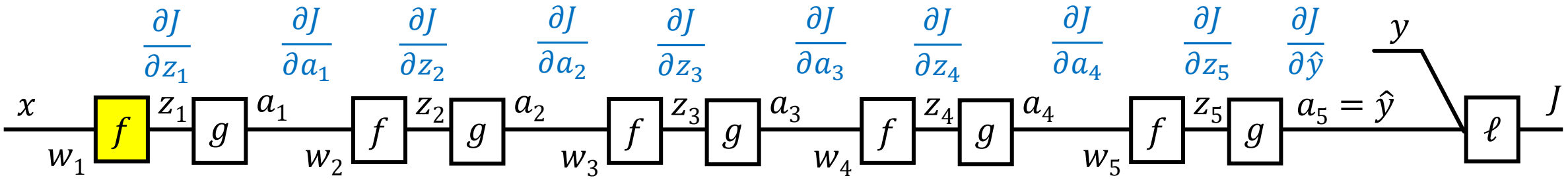


$$\frac{\partial J}{\partial z_1} = \frac{\partial J}{\partial a_1}\frac{\partial g}{\partial z_1}$$

Backpropagation

$$\hat{y} = h_{\boldsymbol{\theta}}(\mathbf{x}) = g\left(w_5 \cdot g\left(w_4 \cdot g\left(w_3 \cdot g\left(w_2 \cdot g(w_1 \cdot x)\right)\right)\right)\right)$$

More efficient to start at the end and reuse values!

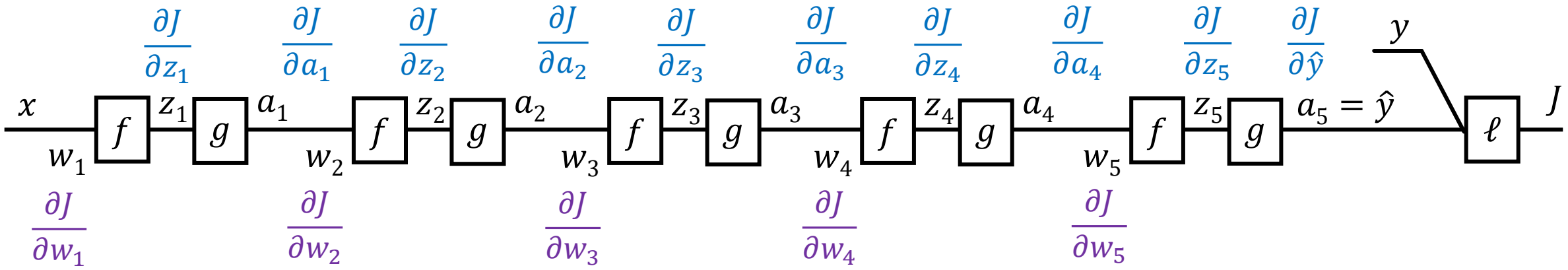


$$\frac{\partial J}{\partial w_1} = \frac{\partial J}{\partial z_1}\frac{\partial f}{\partial w_1}$$

Backpropagation

$$\hat{y} = h_{\boldsymbol{\theta}}(\mathbf{x}) = g\left(w_5 \cdot g\left(w_4 \cdot g\left(w_3 \cdot g\left(w_2 \cdot g\left(w_1 \cdot x\right)\right)\right)\right)\right)$$

More efficient to start at the end and reuse values!



To do gradient descent we need the partial derivative of the objective with respect to each parameter, $w_\ell \leftarrow w_\ell - \alpha\, \partial J / \partial w_\ell$

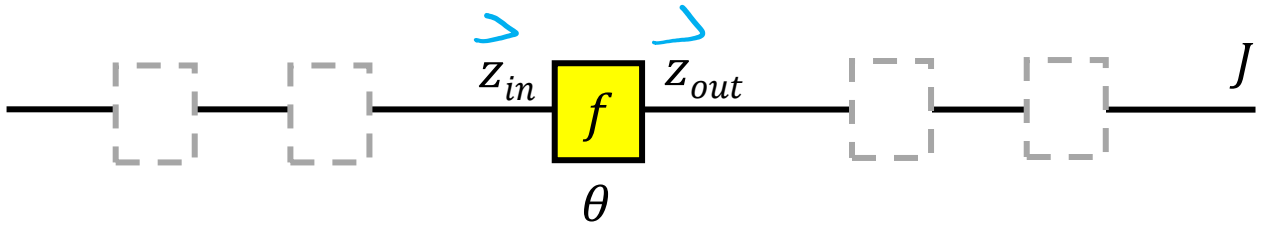
The backward pass propagates the change in the objective with respect to intermediate values ($\partial J / \partial z_\ell$ and $\partial J / \partial a_\ell$) back through the network to produce each $\partial J / \partial w_\ell$

Generic Layer Implementation (so-far)

Compute derivatives per layer, utilizing previous derivatives

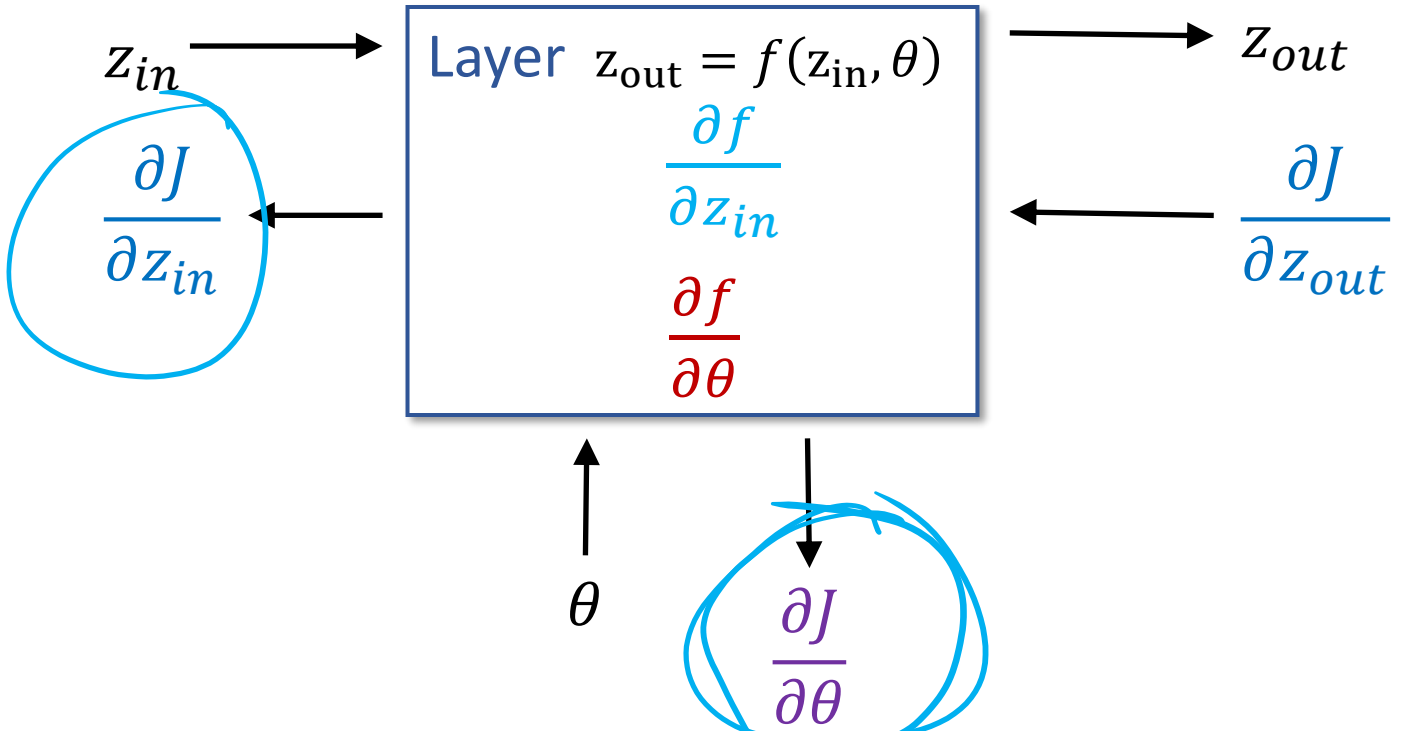
Objective: $J(\theta)$

Arbitrary layer: $\mathrm{z_{out}} = f(\mathrm{z_{in}}, \theta)$



Need:

- $\frac{\partial J}{\partial z_{in}} = \frac{\partial J}{\partial z_{out}} \frac{\partial f}{\partial z_{in}}$
- $\frac{\partial J}{\partial \theta} = \frac{\partial J}{\partial z_{out}} \frac{\partial f}{\partial \theta}$



Outline

Today: Neural Networks

- Three-neuron network + optimization
- Neural network structure (adding more neurons)
- Forward pass and backpropagation (scalar version)

Next time: Neural Networks

- Calculus: multi-variate chain rule
- Backpropagation (vector version)
- Optimization intuition (sliders)
- Neural network properties and intuition