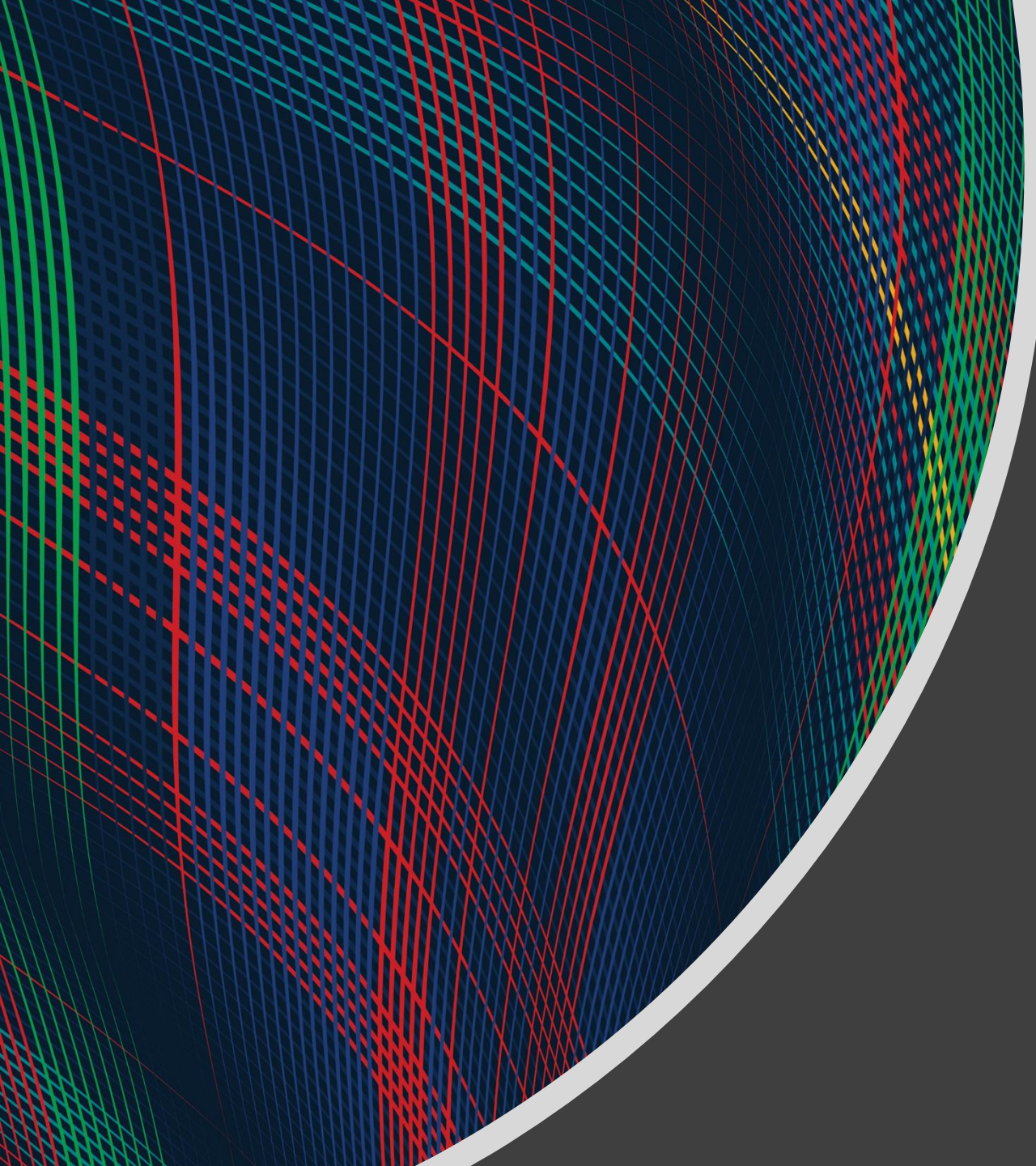




07-280

Introduction to ML

Neural Networks

Instructor:

Aviral Kumar & Nihar Shah

Outline

Pre-reading: Neural Networks

- Network diagrams for linear and logistic regression
- Neuron and activation functions
- Three-neuron network
- Neural network structure (adding more neurons)

Today: Neural Networks

- Neural Network Regression
- Scalar Optimization (Backpropagation)

Next time: Neural Network

- Vector/Matrix Calculus Optimization (Backpropagation)
- Properties and Intuition

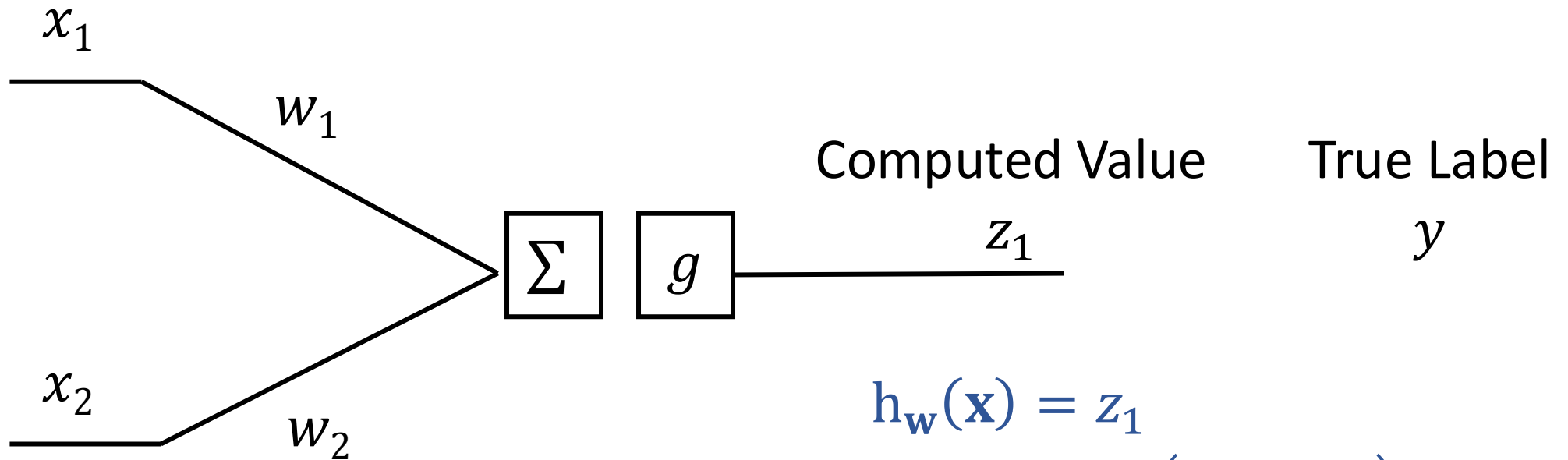
Neuron

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)



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

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

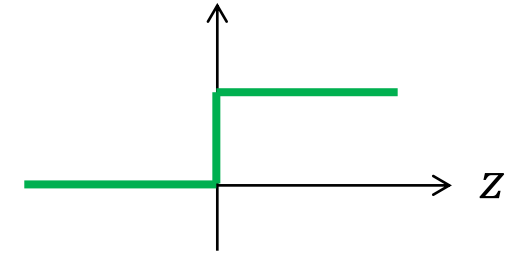
Activation Functions

Pre-reading

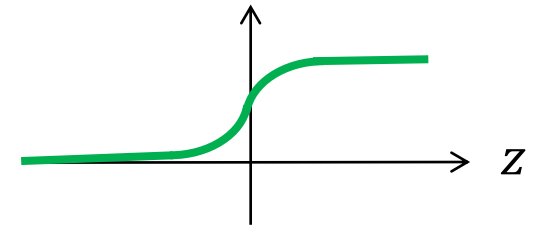
Activation Functions

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

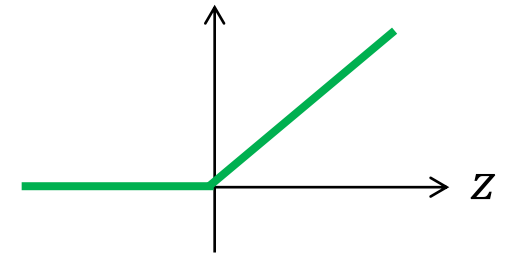
- Hard threshold: $g(z) = \begin{cases} 1 & z \geq 0 \\ 0 & z < 0 \end{cases} \quad \frac{dg}{dz} = \begin{cases} 0 & z \geq 0 \\ 0 & z < 0 \end{cases}$



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



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

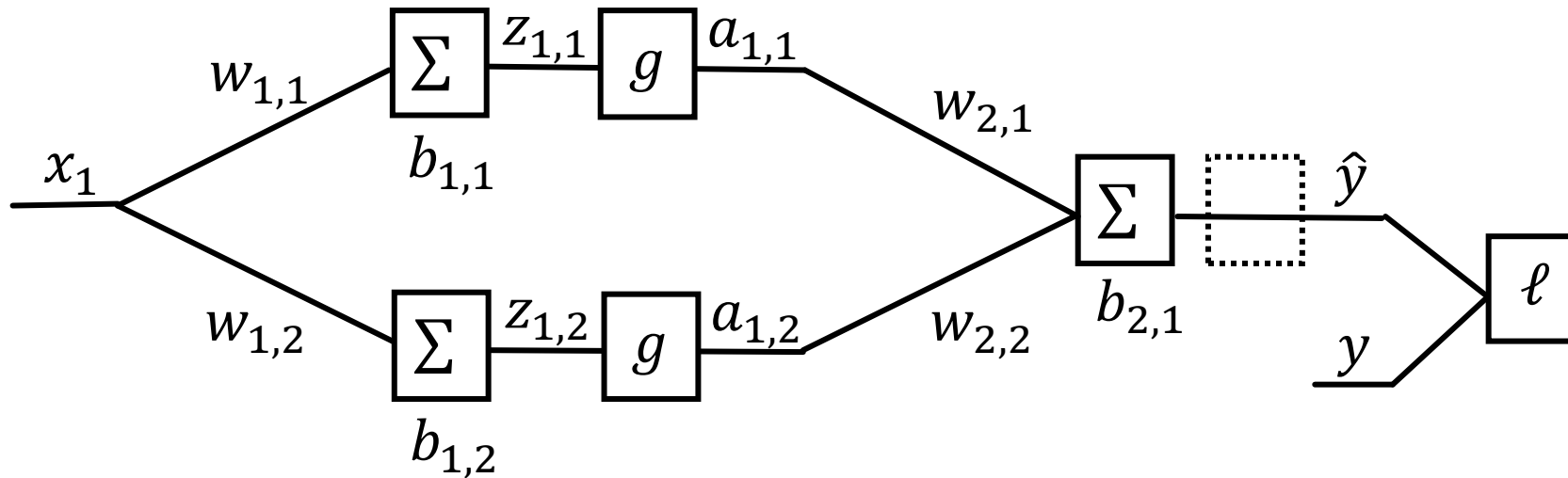


Three-neuron Network

Pre-reading

Three-neuron Network

Network diagram



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

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

Adding More Neurons

Pre-reading

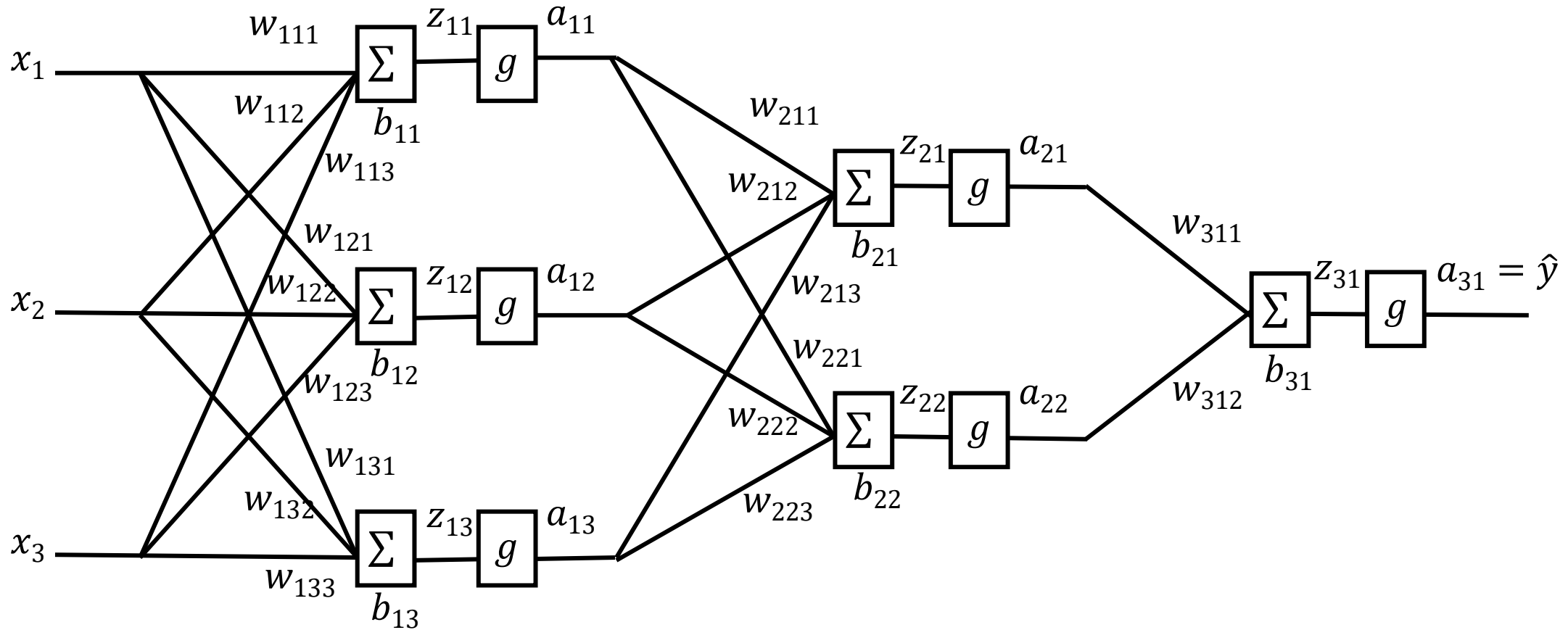
Three-layer Network

Parameter naming conventions

$w_{\text{layer output input}}$

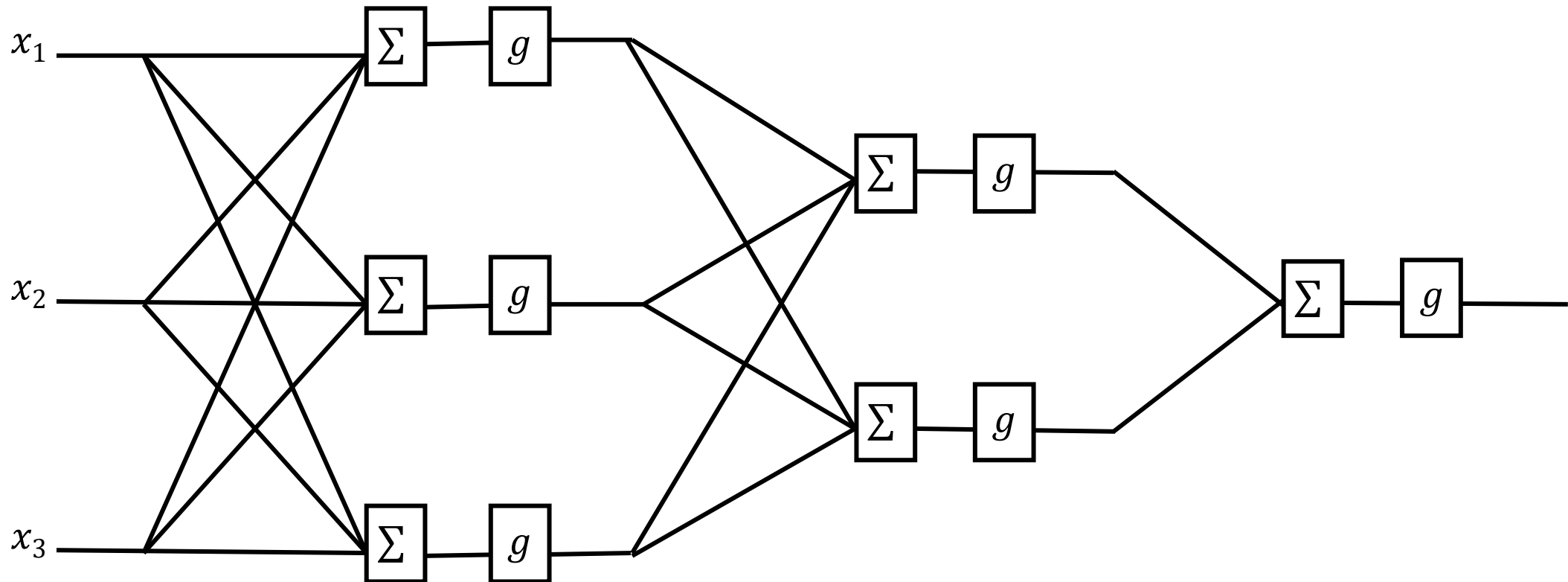
$b_{\text{layer output}}$

Network diagram



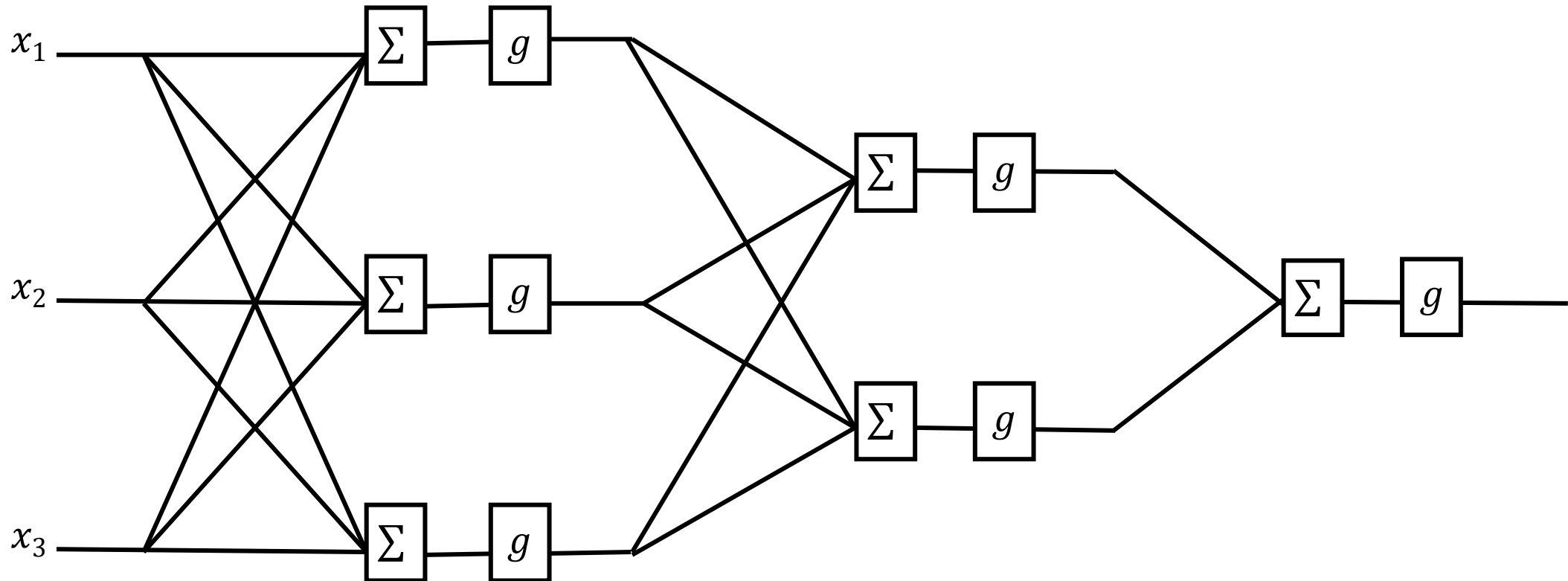
Three-layer Network

Different ways to define network layers: *Layers of functions*



Three-layer Network

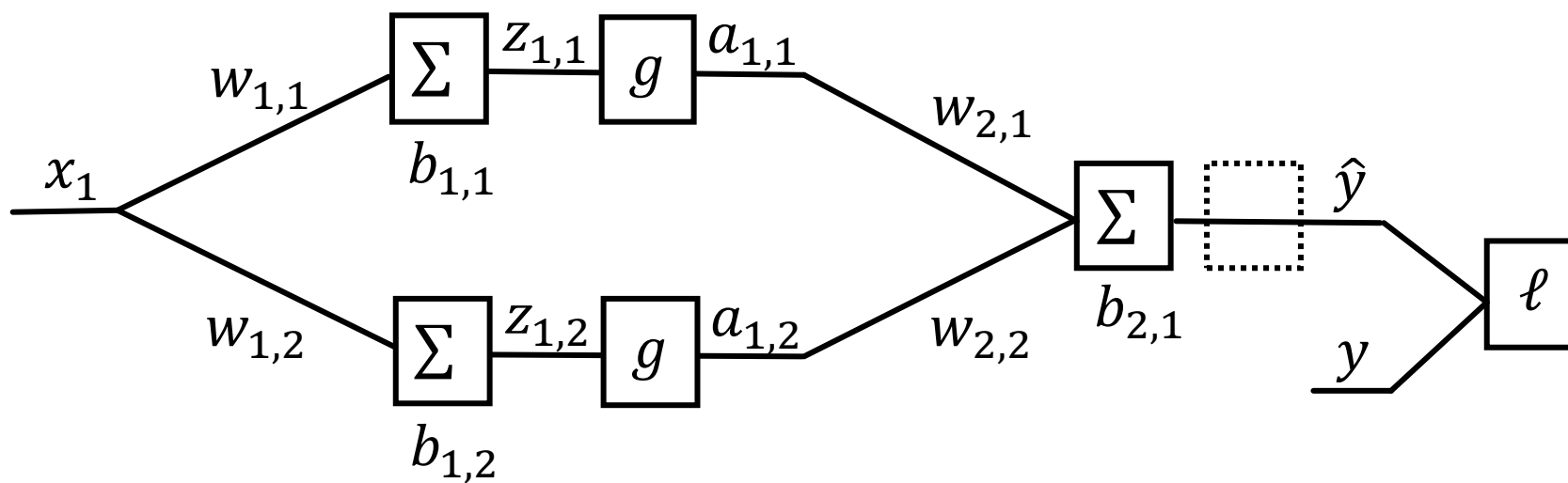
Different ways to define network layers: *Layers of neurons*



Neural Network Regression

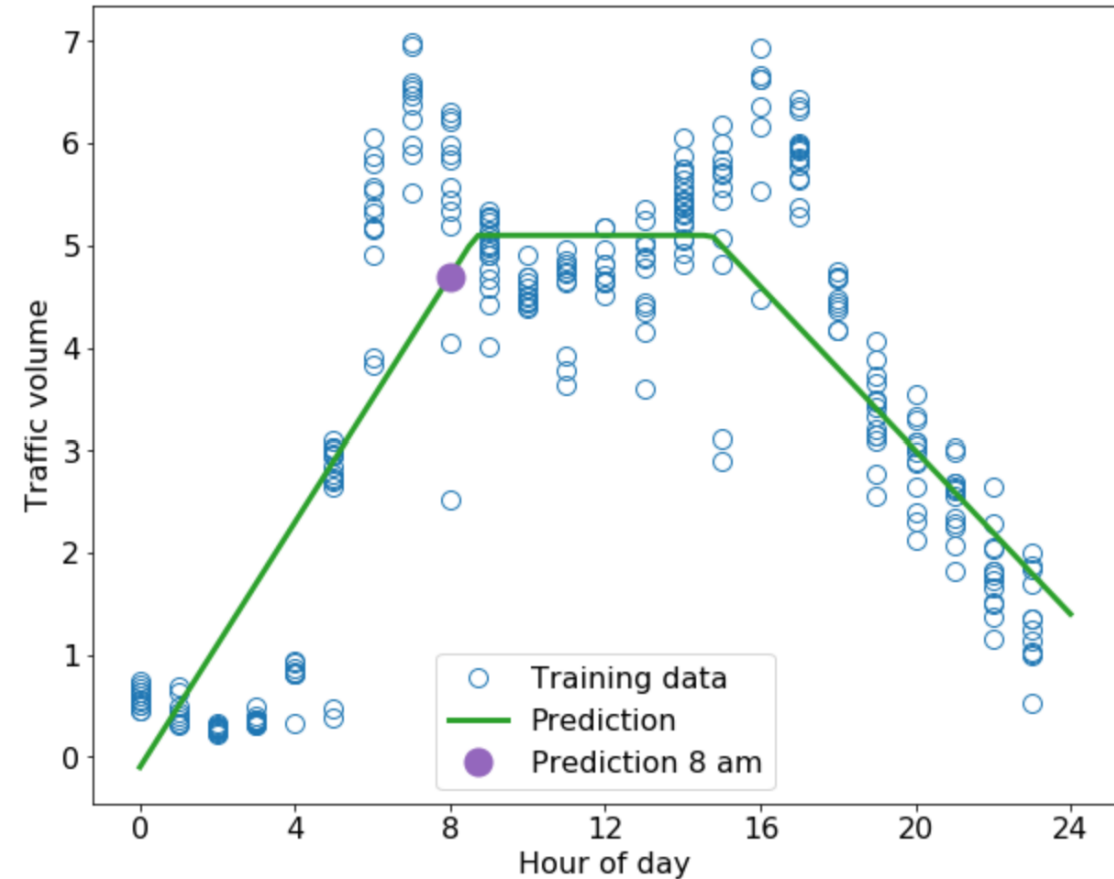
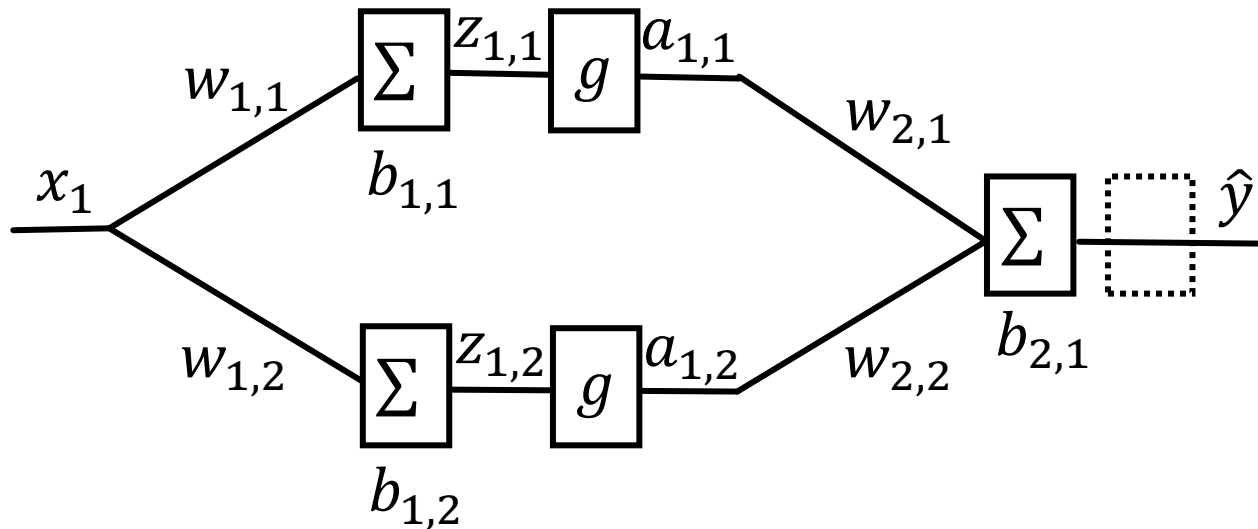
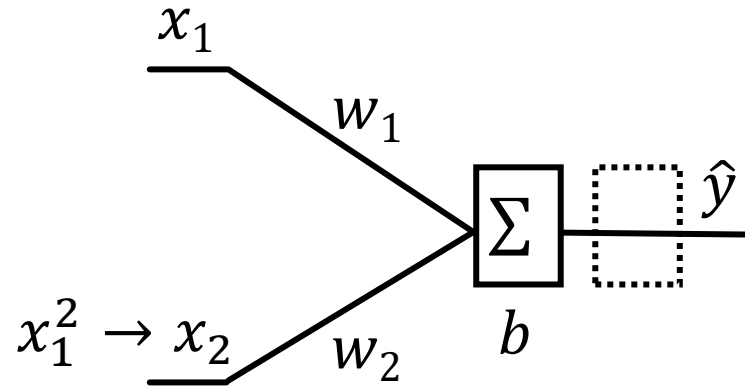
Poll 1

How many total parameters?



Traffic Data Example

Quadratic feature engineering vs three-neuron network



Traffic Data Example

Three-neuron network

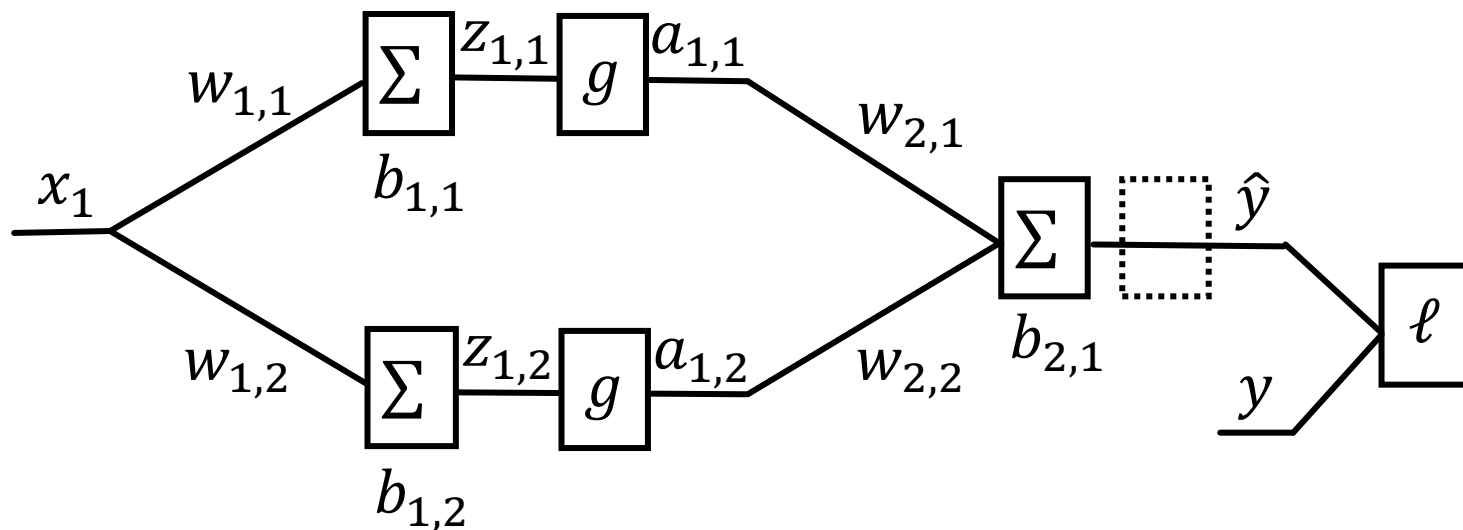
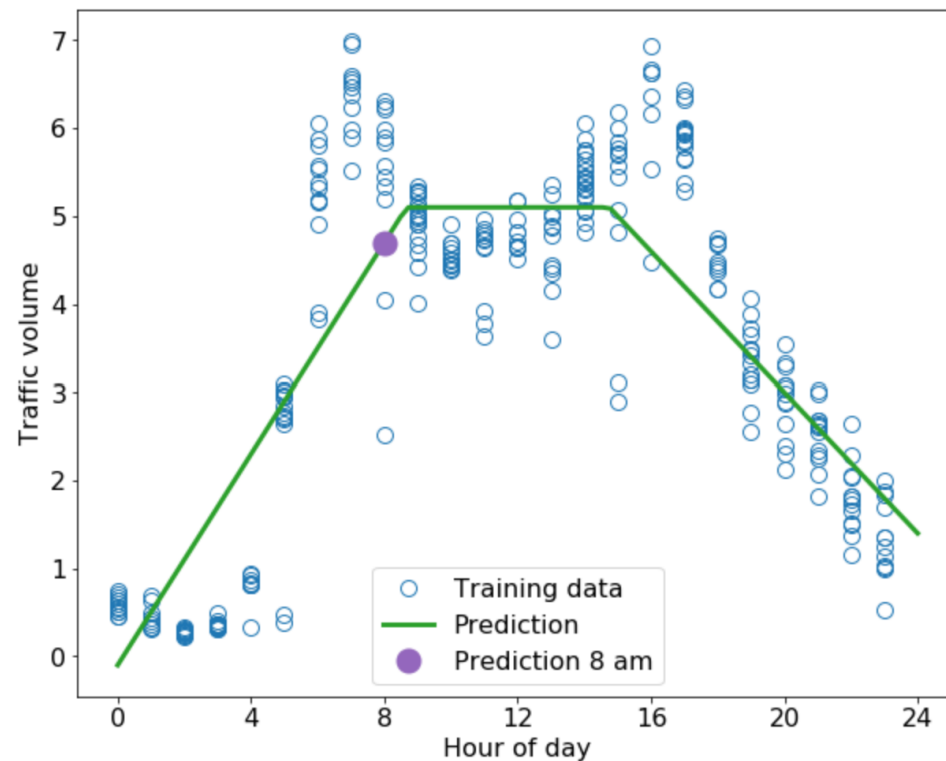
Parameters

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

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

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

$$w_{2,2}: -1.0$$



Traffic Data Example

Three-neuron network

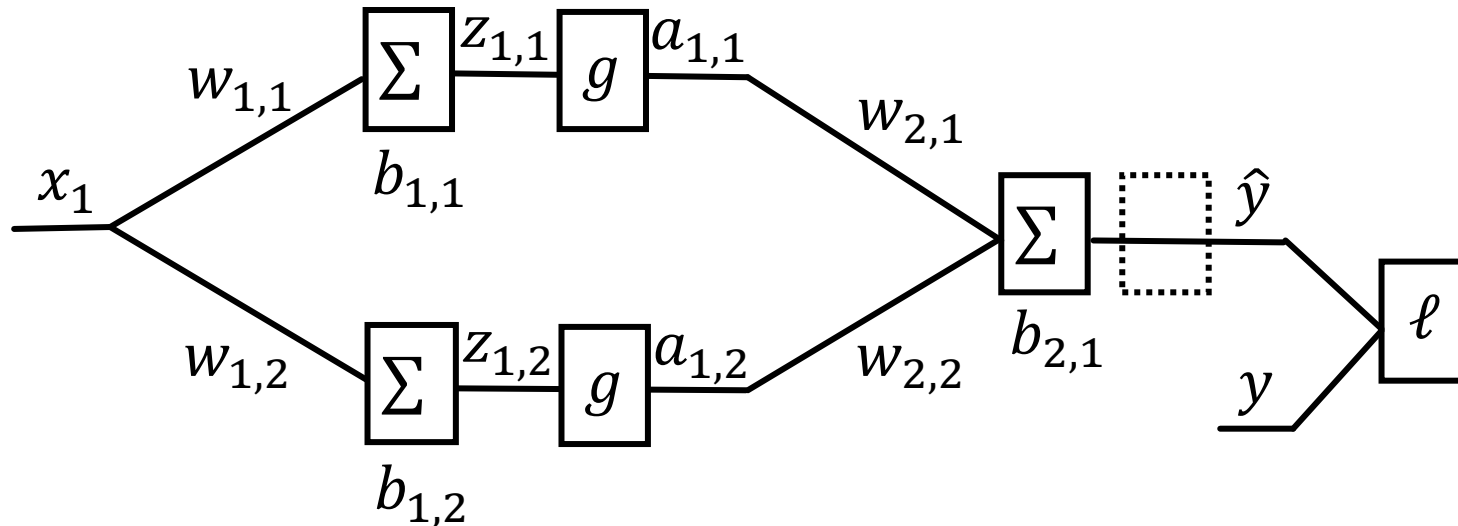
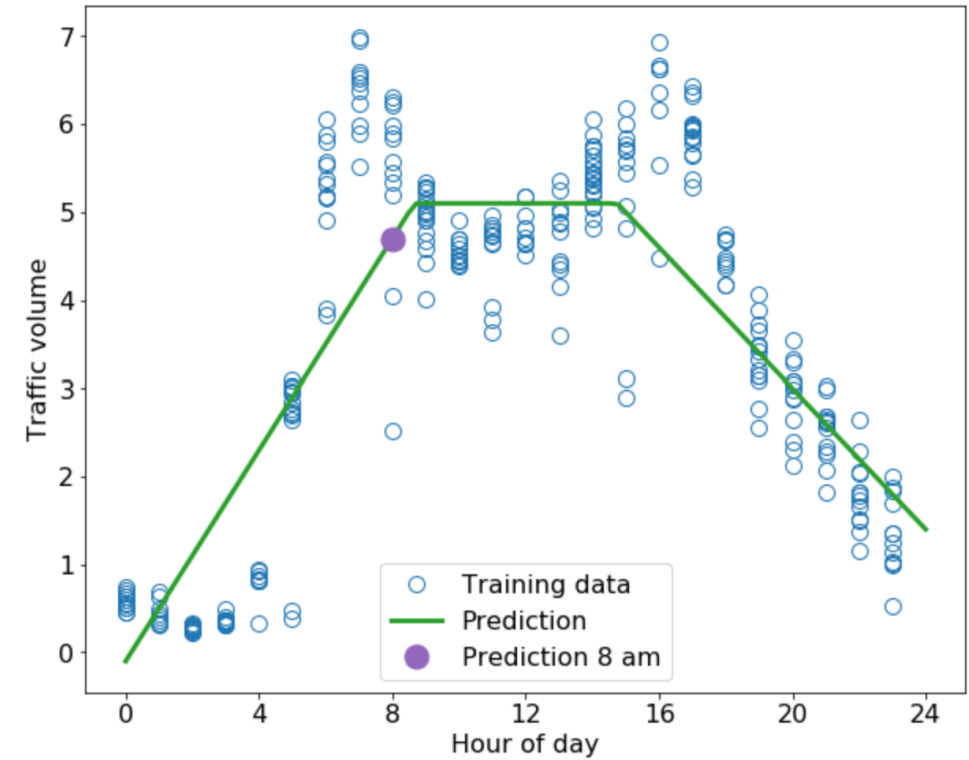
Parameters

$w_{1,1}$: 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

Objective, Loss Functions, Gradient Descent

Objective

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

Loss functions

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

Gradient descent

while not converged

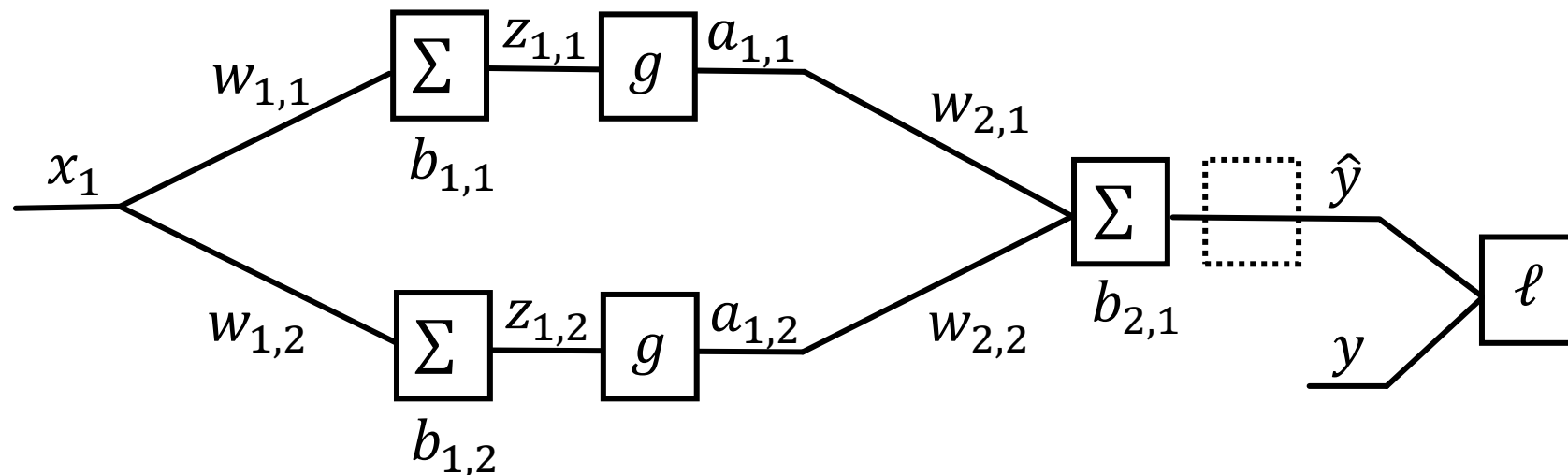
$$\theta \leftarrow \theta - \alpha \partial J / \partial \theta$$

Optimization

Three-neuron network

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

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



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

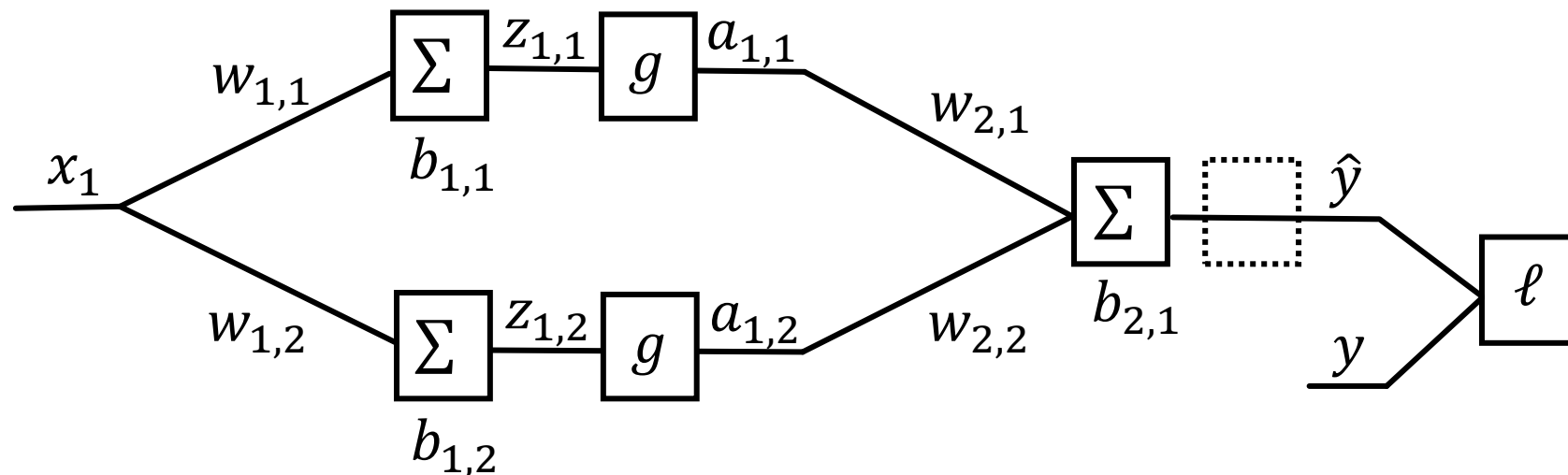
$$\frac{\partial J}{\partial b_{2,1}} =$$

Optimization

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

Three-neuron network

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



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

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

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

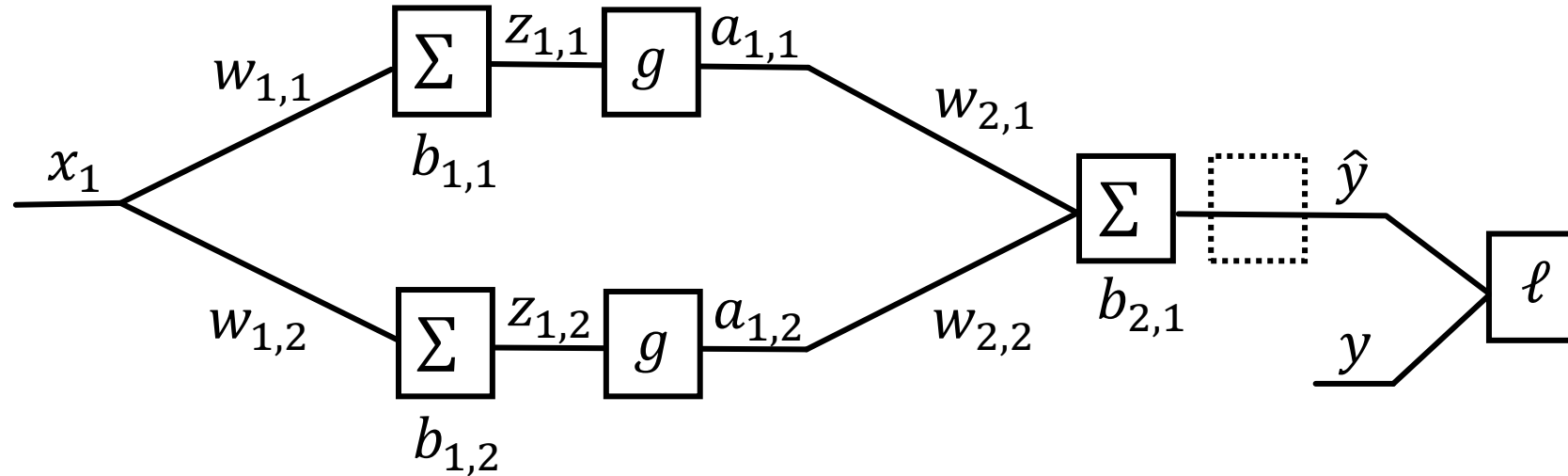
$$\frac{\partial J}{\partial b_{2,1}} = \frac{\partial \ell}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial b_{2,1}}$$

Optimization

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

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

Three-neuron network



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

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

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

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

Neural Net Forward Pass and Backpropagation

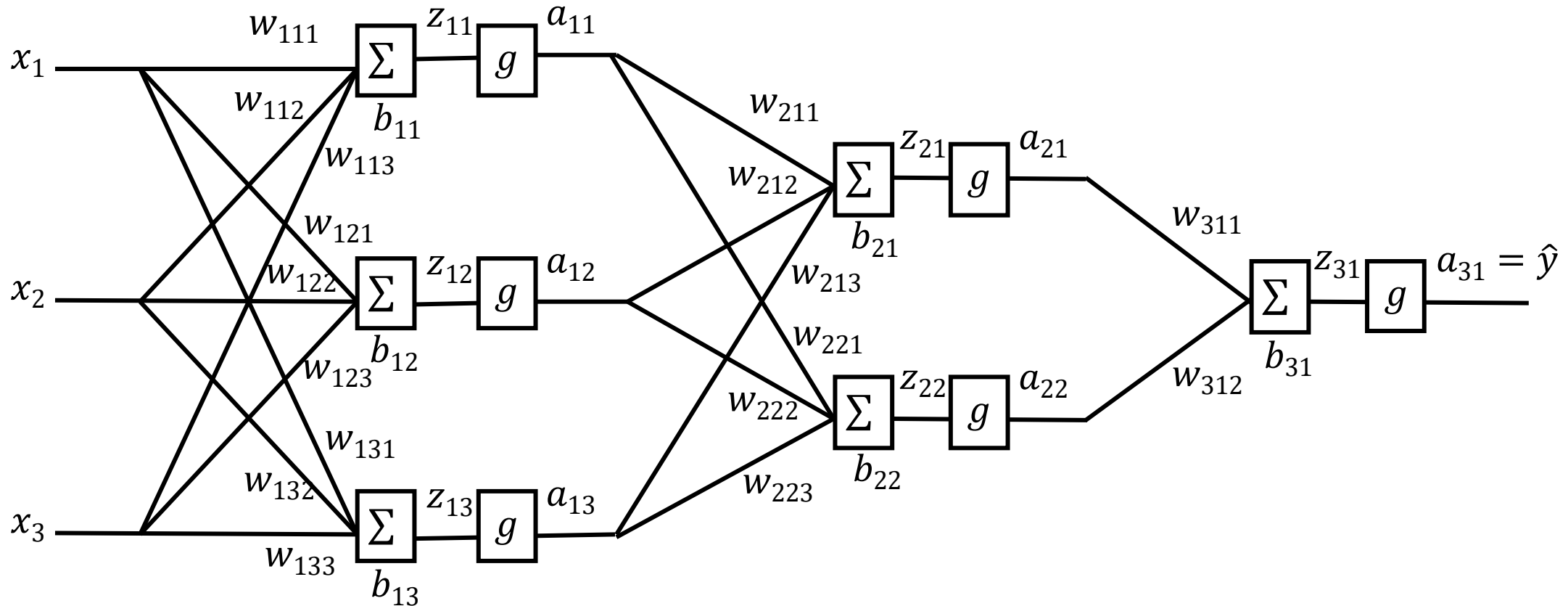
Forward pass

Parameter naming conventions

$w_{\text{layer output input}}$

$b_{\text{layer output}}$

Predicting output from input

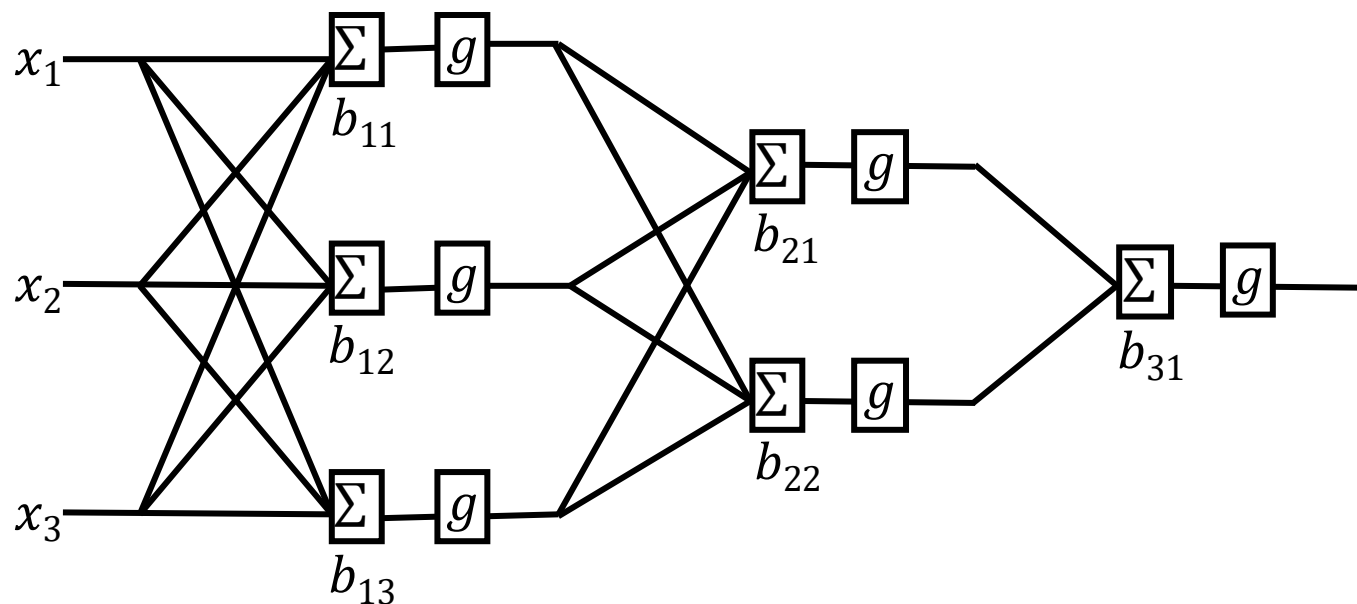


$$\hat{y} = h_{\theta}(\mathbf{x}) = g \left(b_{3,1} + \sum_k w_{3,1,k} g \left(b_{2,k} + \sum_i w_{2,k,i} g \left(b_{1,i} + \sum_j w_{1,i,j} x_j \right) \right) \right)$$

Optimization

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

Tons of repeated partial derivatives



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

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

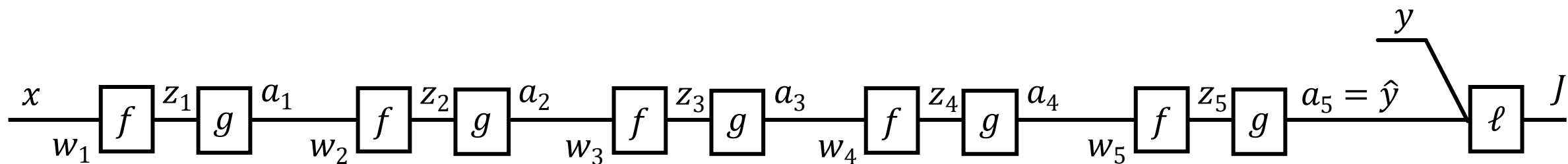
$$\frac{\partial J}{\partial b_{1,i}} = \frac{\partial \ell}{\partial \hat{y}} \frac{\partial a_{3,1}}{\partial z_{3,1}} \frac{\partial z_{3,1}}{\partial \mathbf{a}_2} \frac{\partial \mathbf{a}_2}{\partial \mathbf{z}_2} \frac{\partial \mathbf{z}_2}{\partial a_{1,i}} \frac{\partial a_{1,i}}{\partial z_{1,i}}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

Forward Pass and Backpropagation

Scalar's version

Forward Pass

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

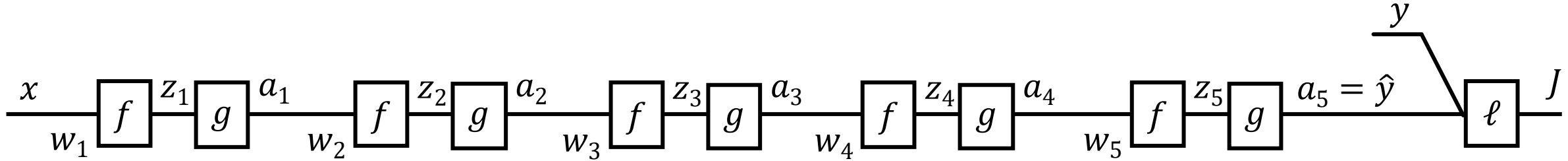


$$\begin{aligned} z_\ell &= f(w_\ell, a_{\ell-1}) = w_\ell \cdot a_{\ell-1} \\ a_\ell &= g(z_\ell) \end{aligned} \qquad \begin{aligned} \hat{y} &= h_{\boldsymbol{\theta}}(\mathbf{x}) = g\left(w_5 \cdot g\left(w_4 \cdot g\left(w_3 \cdot g\left(w_2 \cdot g(w_1 \cdot x)\right)\right)\right)\right) \\ &= g\left(f\left(g\left(f\left(g\left(f\left(g\left(f\left(g(f(x))\right)\right)\right)\right)\right)\right)\right)\right) \end{aligned}$$

Forward Pass

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

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

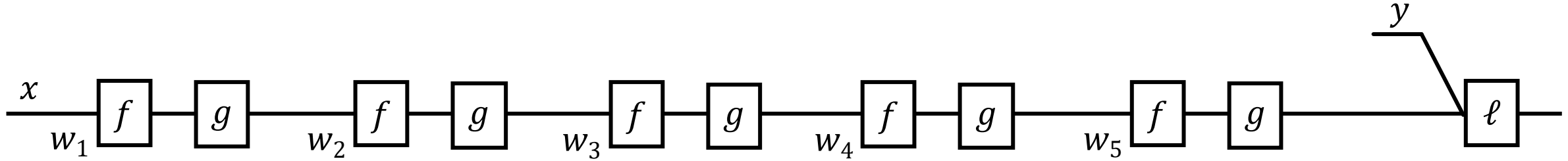


With a new data point (x, y) , we have our current weight values
but we don't $\hat{y}$ (or any of the intermediate values z_ℓ and a_ℓ)
or the value of our object function J

Forward Pass

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

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

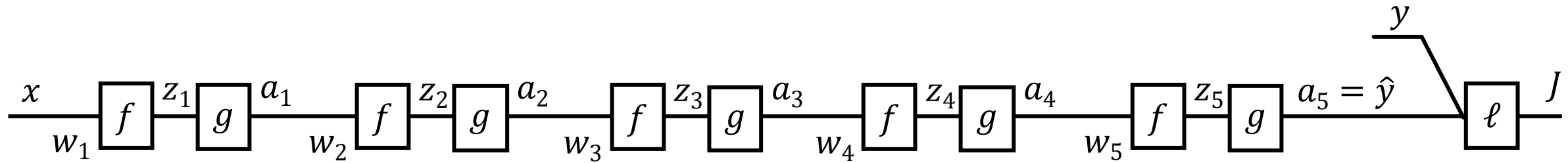


With a new data point (x, y) , we have our current weight values but we don't $\hat{y}$ (or any of the intermediate values z_ℓ and a_ℓ) or the value of our object function J

Forward Pass

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

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



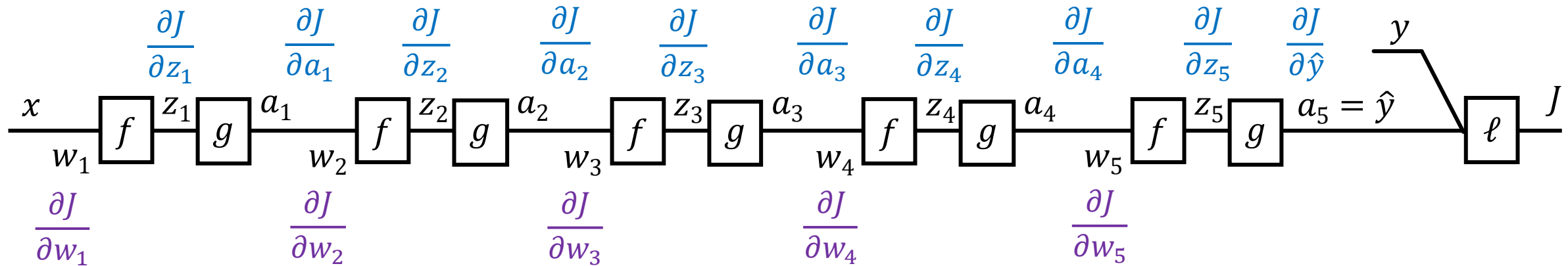
With a new data point (x, y) , we have our current weight values but we don't have $\hat{y}$ (or any of the intermediate values z_ℓ and a_ℓ) or the value of our object function J

The forward pass propagates x (forward) through the network to give us these values

Backpropagation

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

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



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

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

Reminder: Calculus Chain Rule (scalar version)

Composite functions → chain rule

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

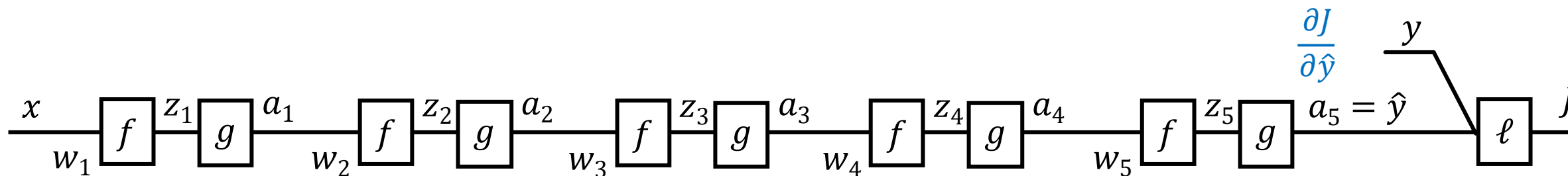
$$\begin{aligned} y &= f(z) \\ z &= g(x) \end{aligned}$$

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

Why backwards?

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

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

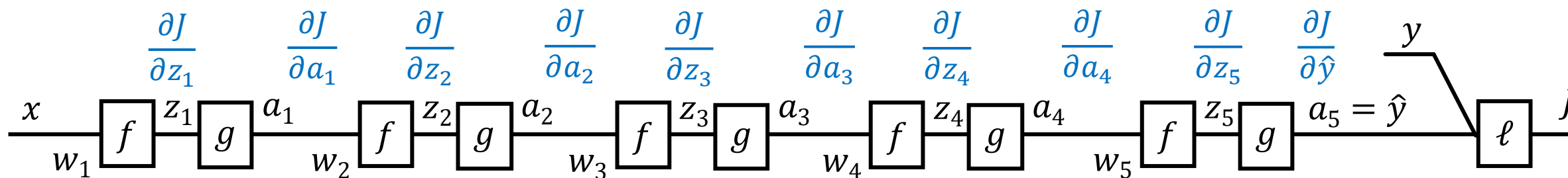


$$\frac{\partial J}{\partial w_1} =$$

Why backwards?

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

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



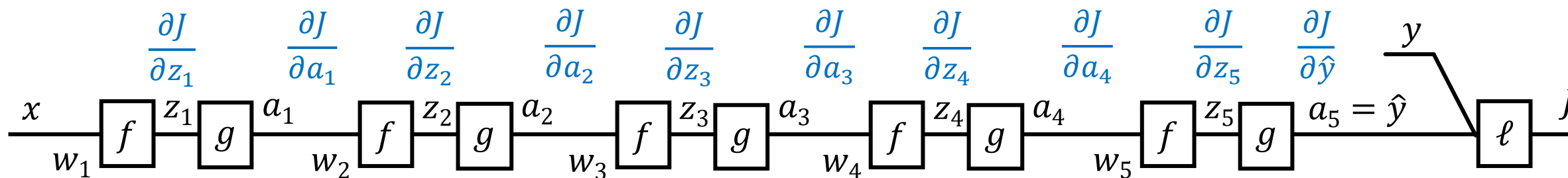
$$\frac{\partial J}{\partial w_1} = \frac{\partial J}{\partial \hat{y}} \frac{\partial g}{\partial z_5} \frac{\partial f}{\partial a_4} \frac{\partial g}{\partial z_4} \frac{\partial f}{\partial a_3} \frac{\partial g}{\partial z_3} \frac{\partial f}{\partial a_2} \frac{\partial g}{\partial z_2} \frac{\partial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

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

Why backwards?

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

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



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

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

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

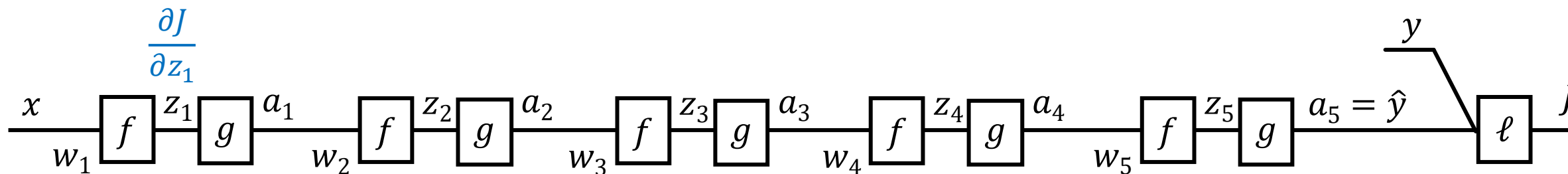
$$\frac{\partial J}{\partial w_4} = \frac{\partial J}{\partial \hat{y}} \frac{\partial g}{\partial z_5} \frac{\partial f}{\partial a_4} \frac{\partial g}{\partial z_4} \frac{\partial f}{\partial w_4}$$

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

Why backwards?

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

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



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

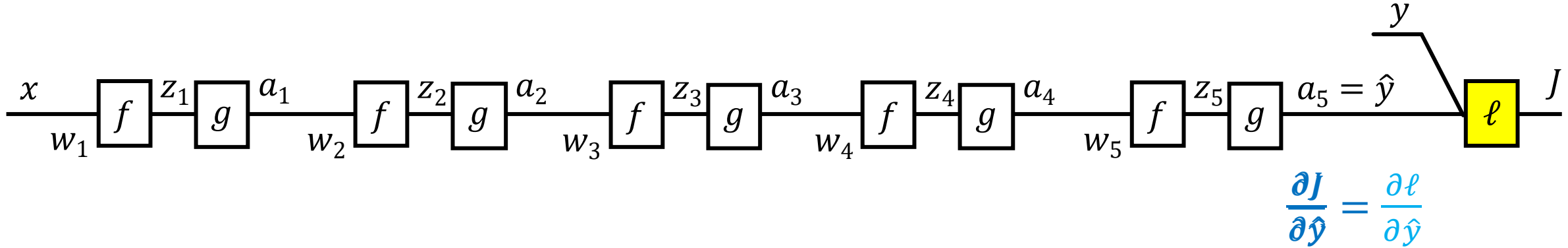
What if someone had already done all of the work to compute $\frac{\partial J}{\partial z_1}$?

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

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

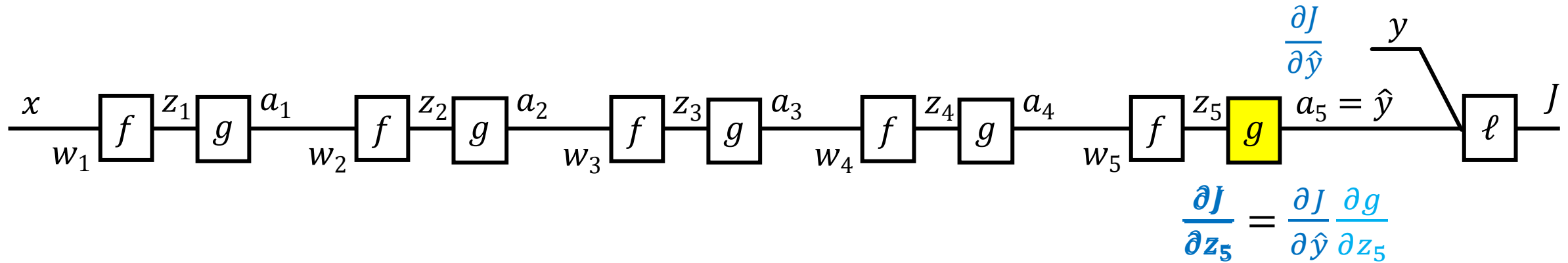
More efficient to start at the end and reuse values!



Backpropagation

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

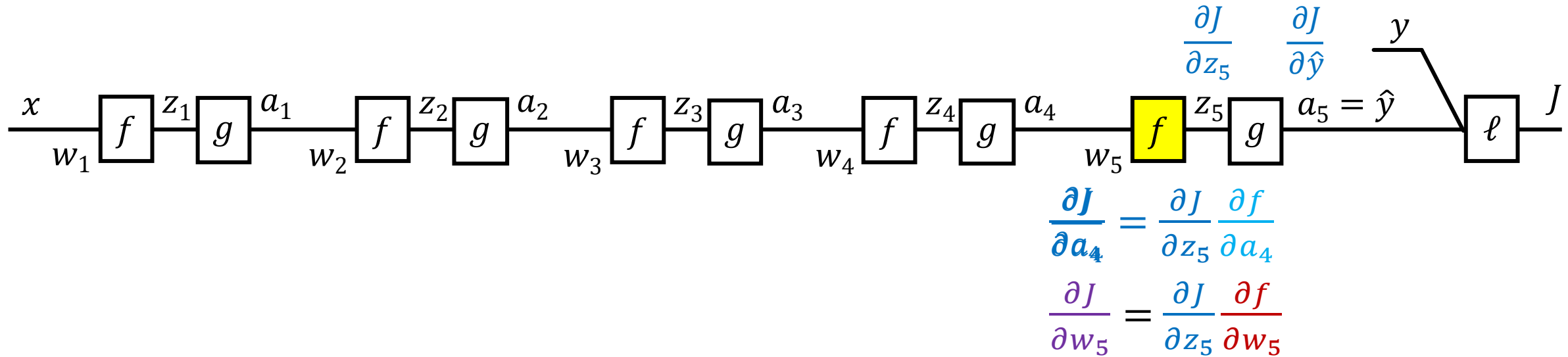
More efficient to start at the end and reuse values!



Backpropagation

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

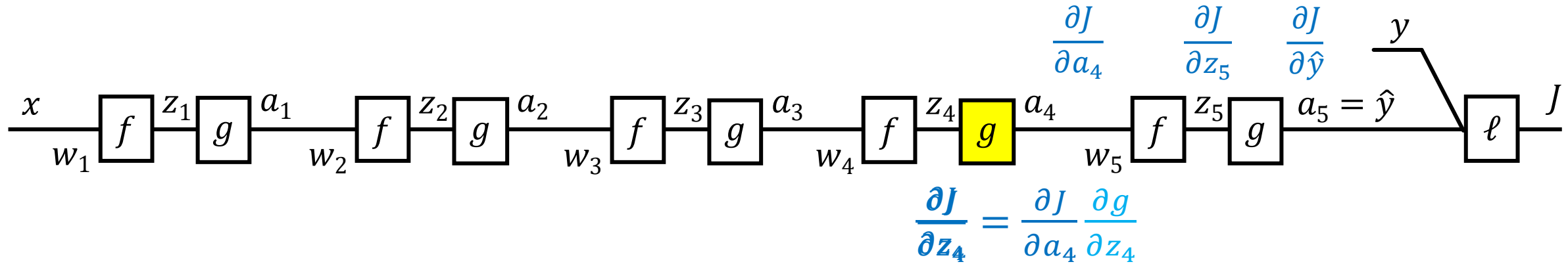
More efficient to start at the end and reuse values!



Backpropagation

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

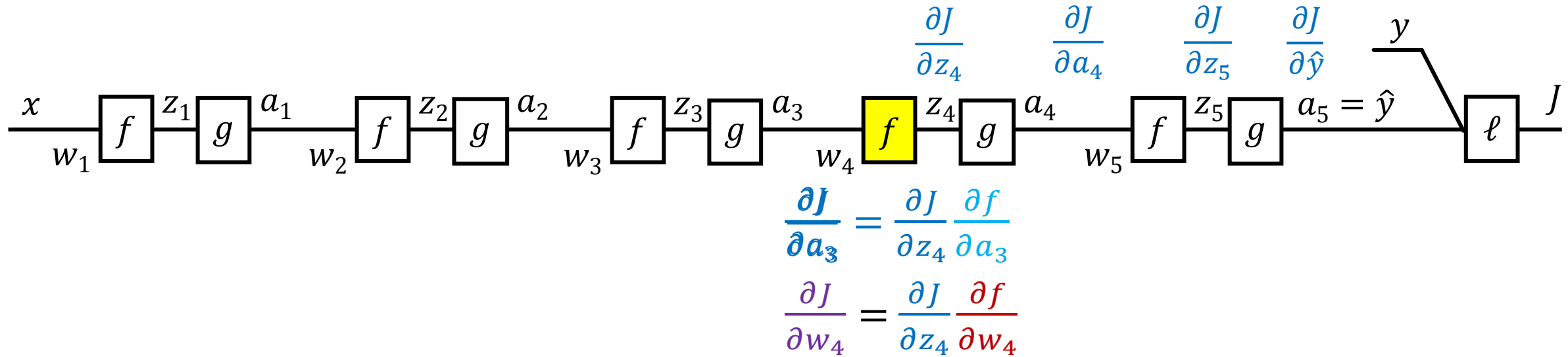
More efficient to start at the end and reuse values!



Backpropagation

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

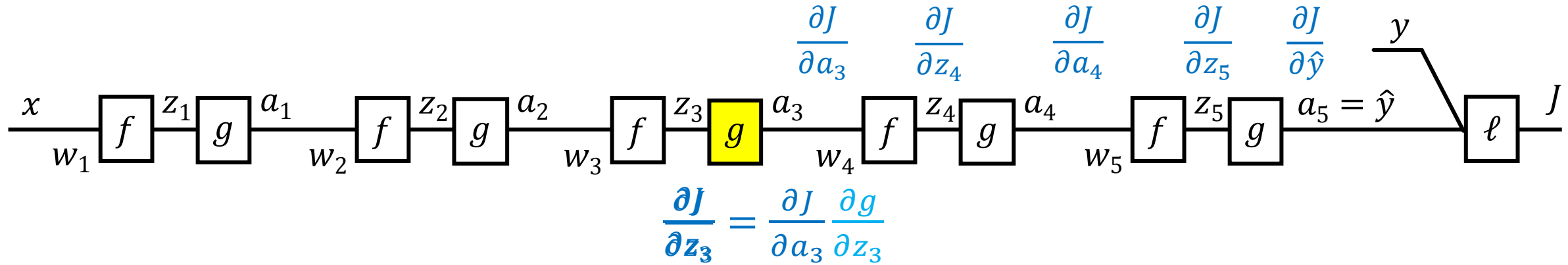
More efficient to start at the end and reuse values!



Backpropagation

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

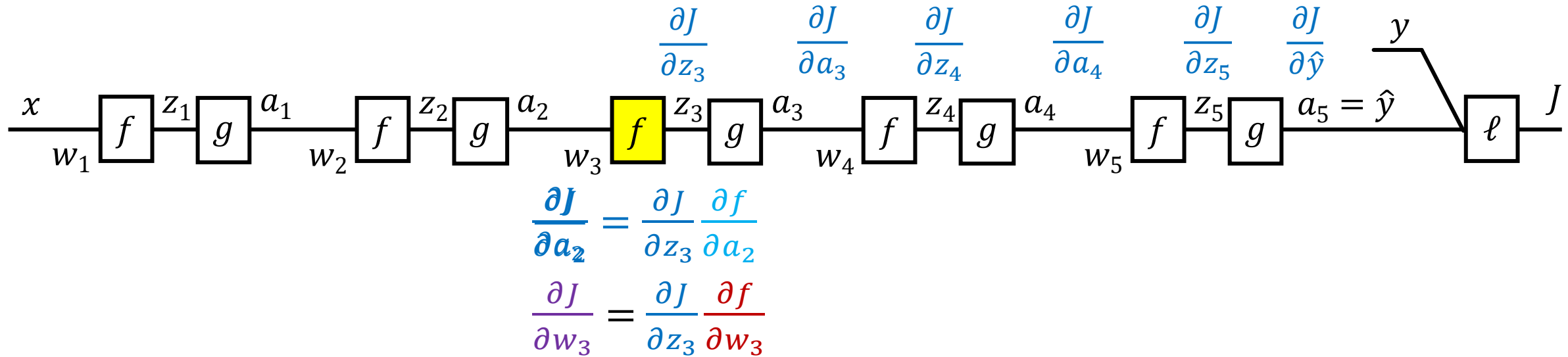
More efficient to start at the end and reuse values!



Backpropagation

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

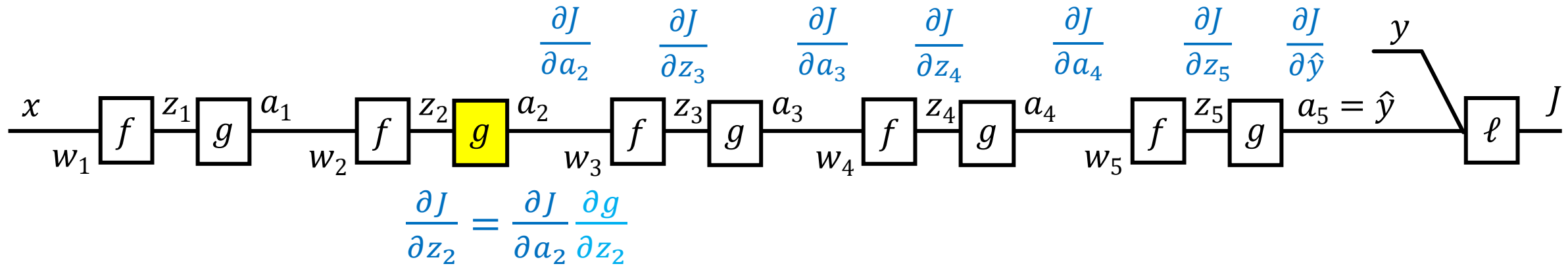
More efficient to start at the end and reuse values!



Backpropagation

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

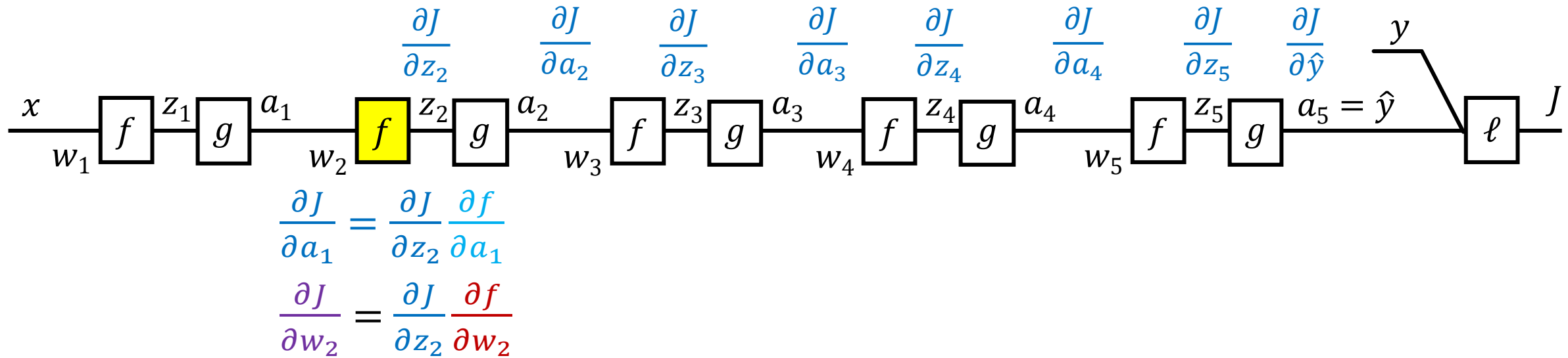
More efficient to start at the end and reuse values!



Backpropagation

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

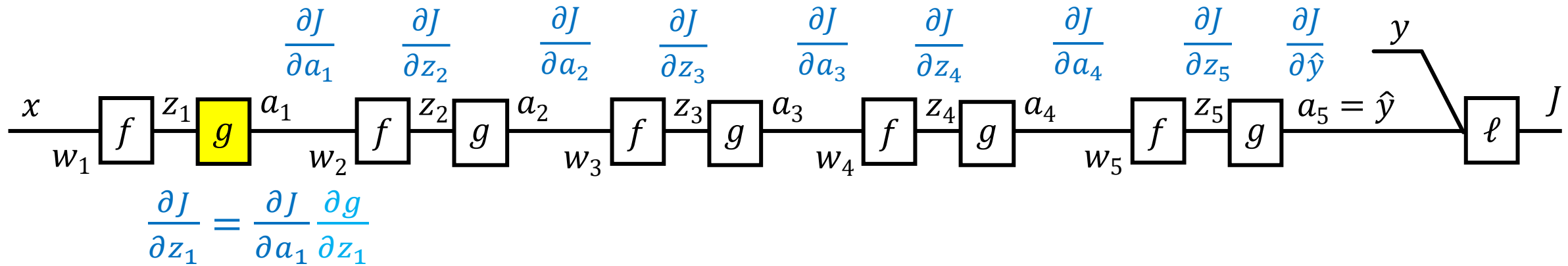
More efficient to start at the end and reuse values!



Backpropagation

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

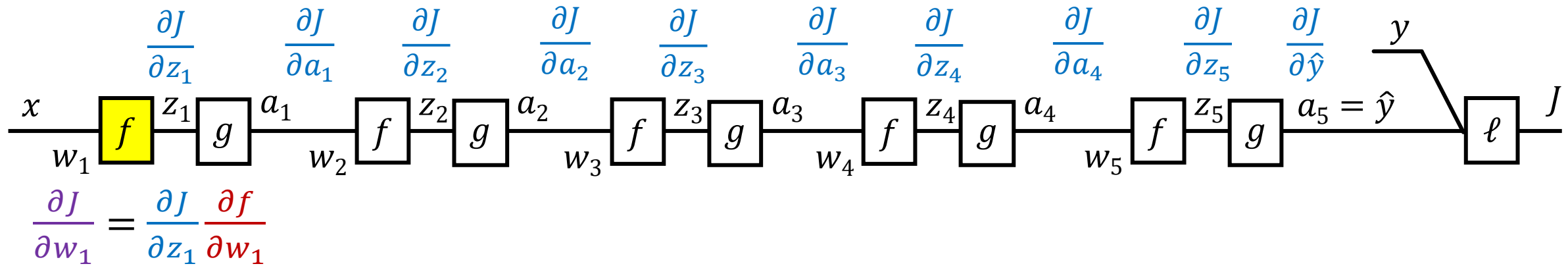
More efficient to start at the end and reuse values!



Backpropagation

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

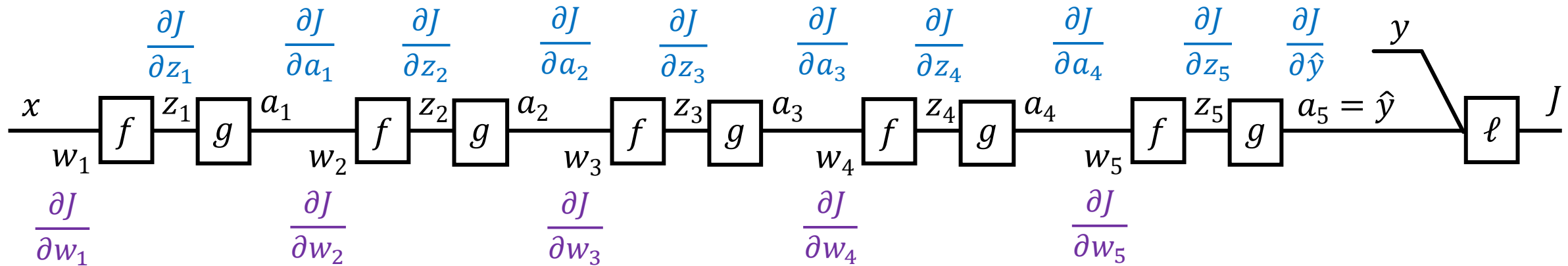
More efficient to start at the end and reuse values!



Backpropagation

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

More efficient to start at the end and reuse values!



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

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

Generic Layer Implementation (so-far)

Compute derivatives per layer, utilizing previous derivatives

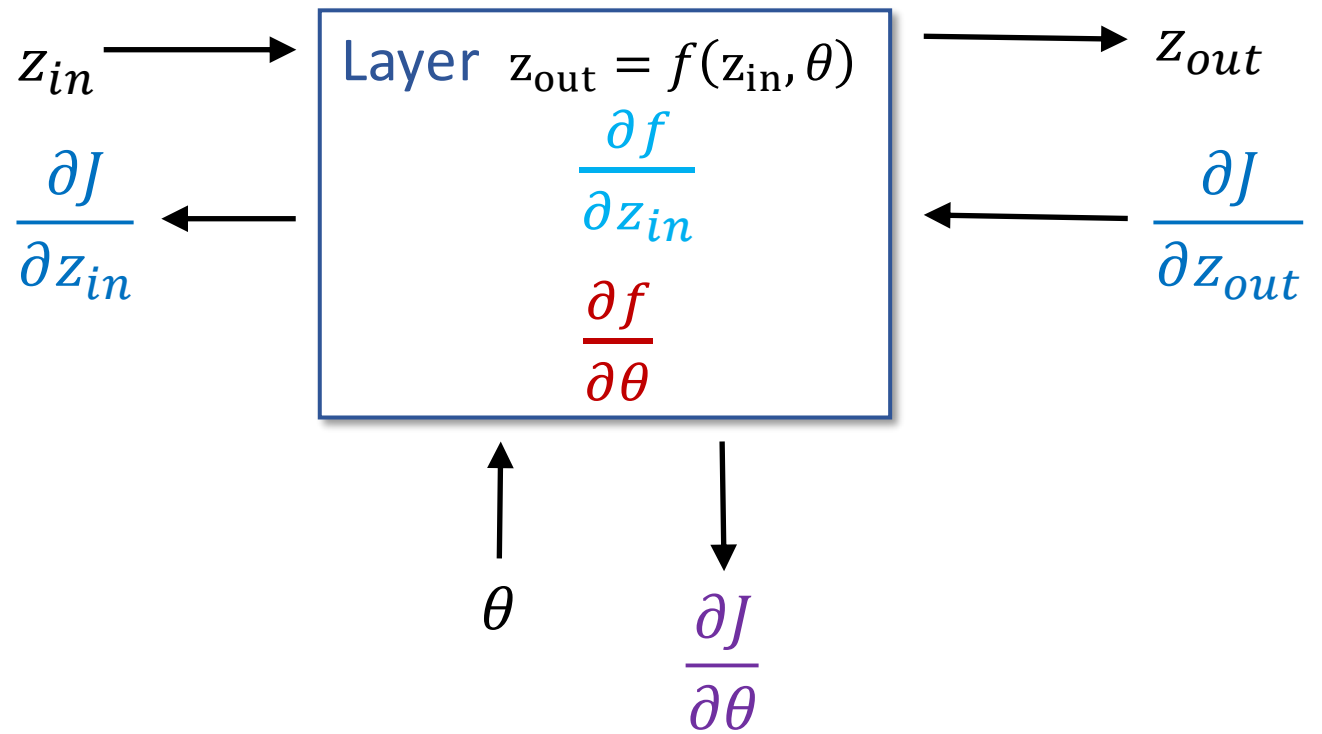
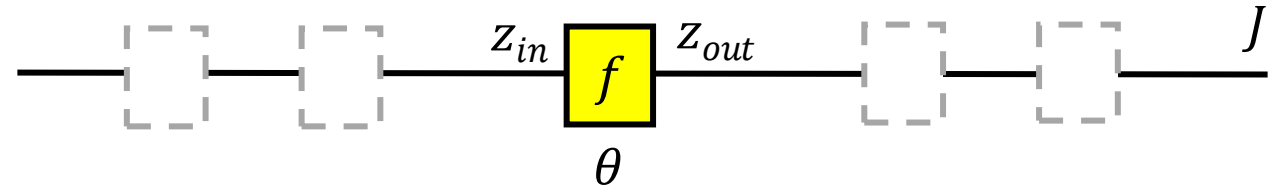
Objective: $J(\theta)$

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

Need:

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

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

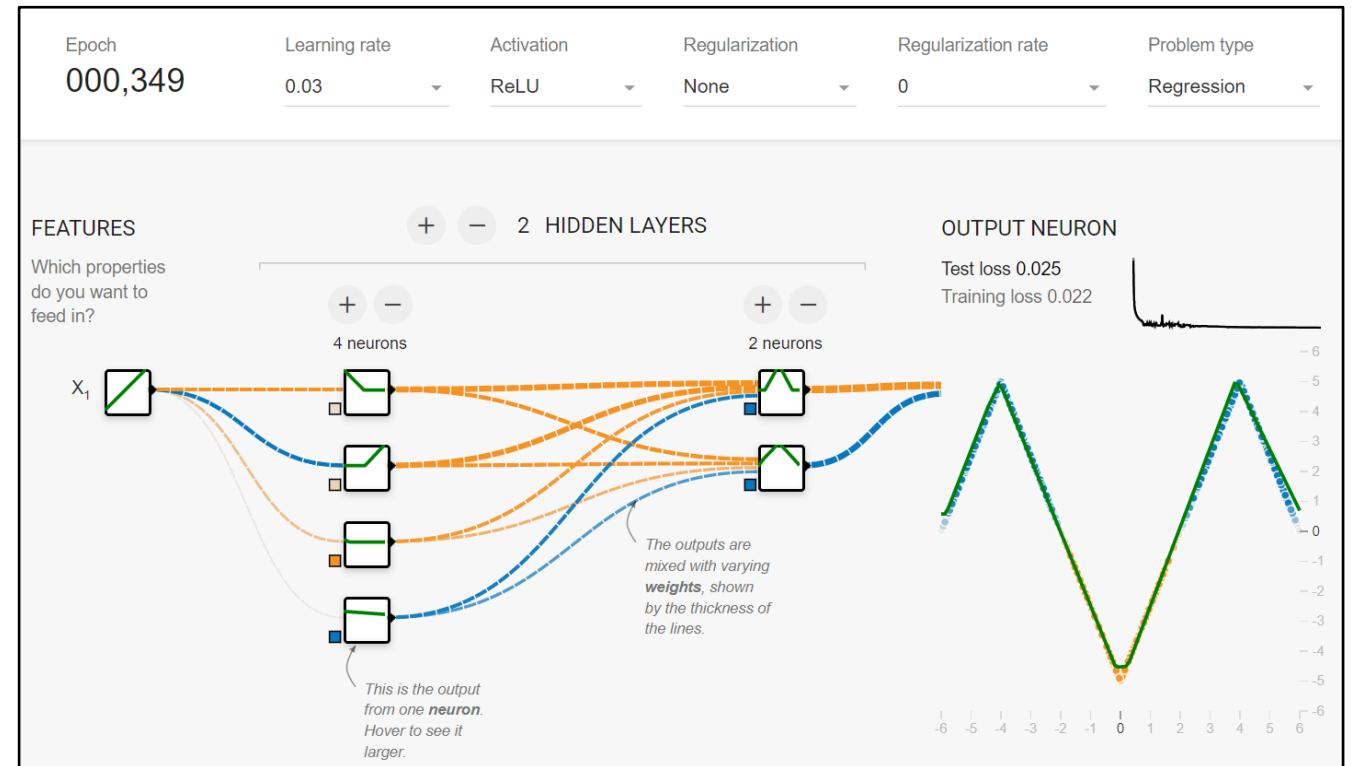


Additional Slides

Simple Network

What do the colors of the lines represent?

- A. Weight
- B. Value from previous neuron



<https://playground.tensorflow.org/>

<https://www.cs.cmu.edu/~pvirtue/tfp>

Image Classification

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

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

