

<https://introml.mit.edu>

6.390 Intro to Machine Learning

Lecture 6: Neural Networks II

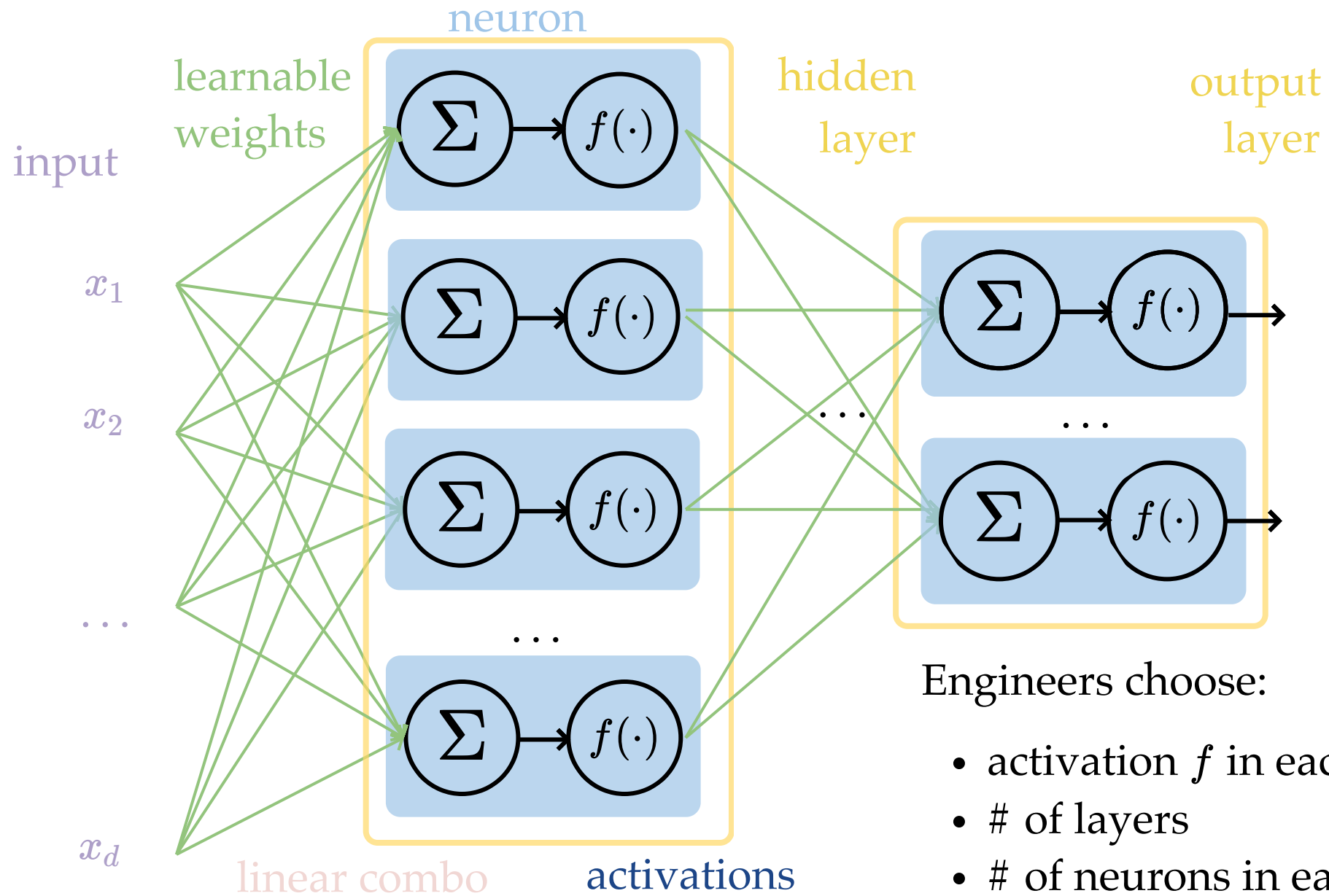
Shen Shen

Oct 13, 2026

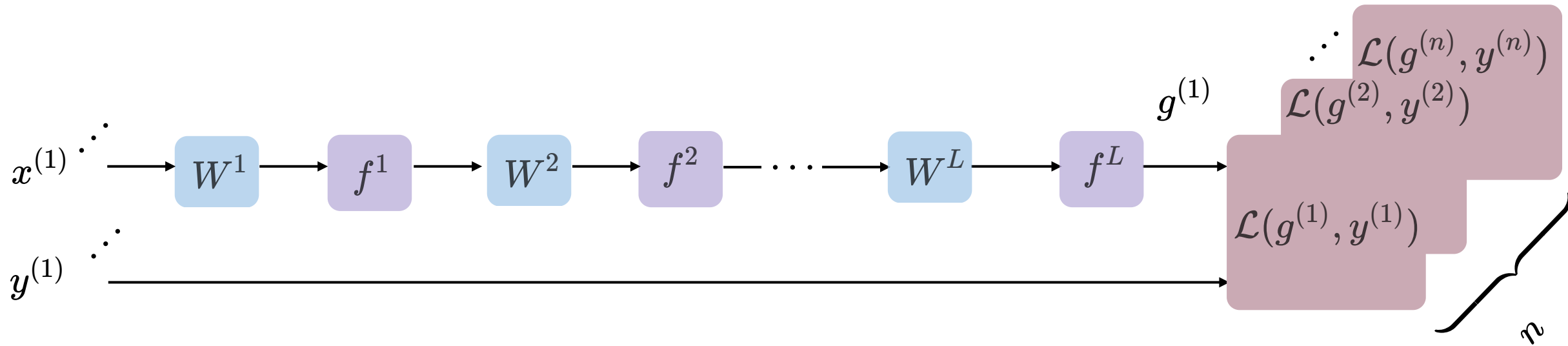
2:30pm, Room 10-250

[Slides and Lecture Recording](#)

Recall: a (fully-connected, feed-forward) neural network (aka, multi-layer perceptrons)



Recall: Forward pass: evaluate, *given* the current parameters



- the model outputs $g^{(i)} = f^L(\dots f^2(f^1(\mathbf{x}^{(i)}; \mathbf{W}^1); \mathbf{W}^2); \dots \mathbf{W}^L)$
- the loss incurred on the current data $\mathcal{L}(g^{(i)}, y^{(i)})$
- the training error $J = \frac{1}{n} \sum_{i=1}^n \mathcal{L}(g^{(i)}, y^{(i)})$

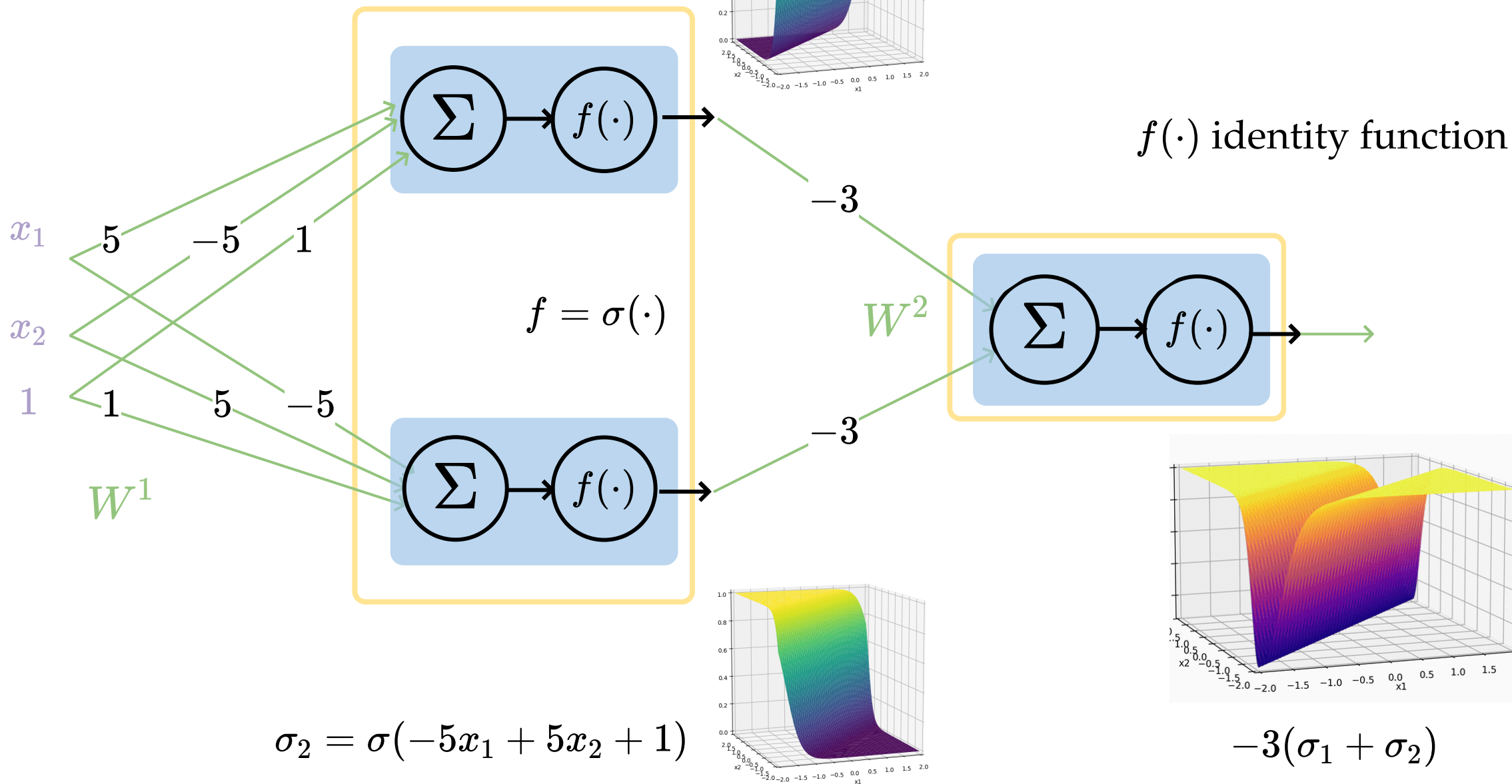
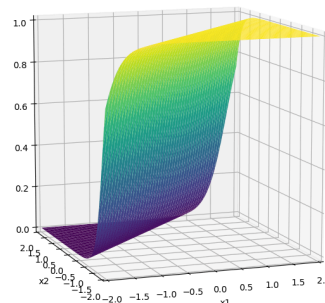
linear combination

(nonlinear) activation

loss function

Recall:

$$\sigma_1 = \sigma(5x_1 - 5x_2 + 1)$$



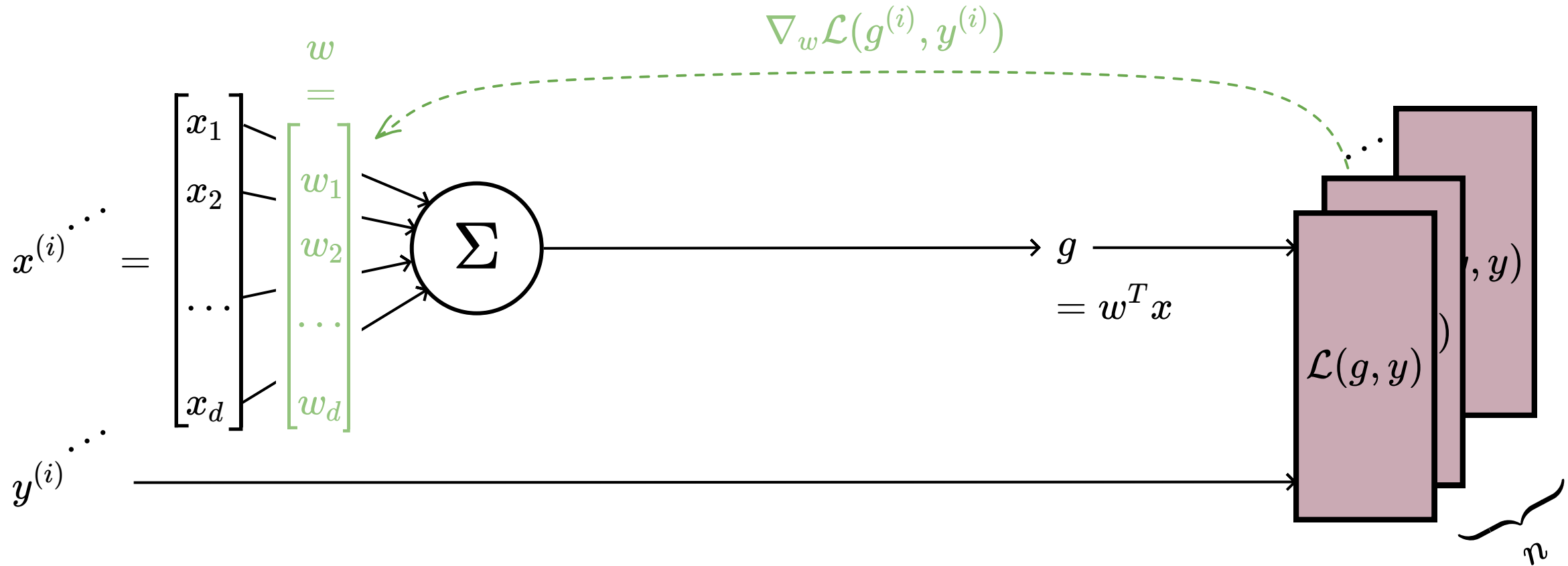
(The demo won't embed in PDF. The direct link below gets to the demo.)

<https://shenshen.mit.edu/demos/nn/playground/?embed>

Outline

- Backward pass (to learn parameters/weights)
 - Backpropagation (gradient descent & the chain rule)
 - Recursive reuse of computation
 - Practical gradient issues and remedies

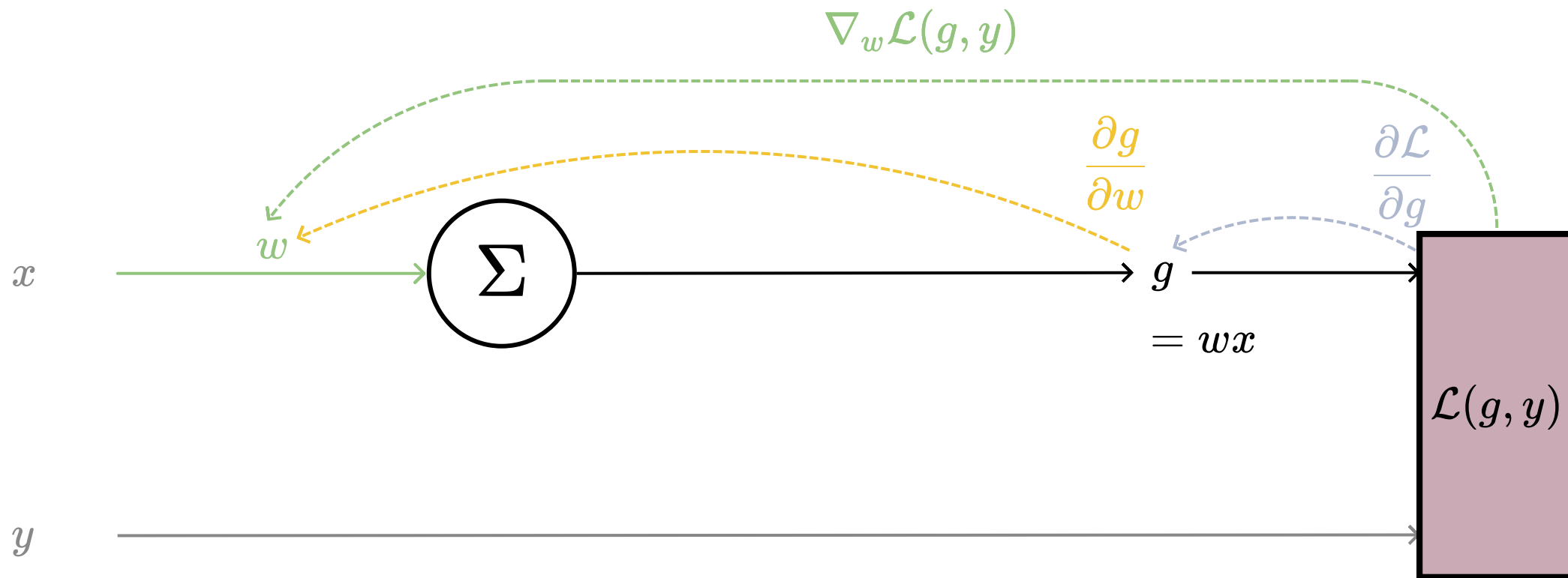
stochastic gradient descent to learn a linear regressor



- Randomly pick a data point $(x^{(i)}, y^{(i)})$
- Evaluate the gradient $\nabla_w \mathcal{L}(g^{(i)}, y^{(i)})$
- Update the weights $w \leftarrow w - \eta \nabla_w \mathcal{L}(g^{(i)}, y^{(i)})$

for simplicity, say training data set is just (x, y) and squared loss

1-d example:
 $x = 3, y = 6, w = 4$



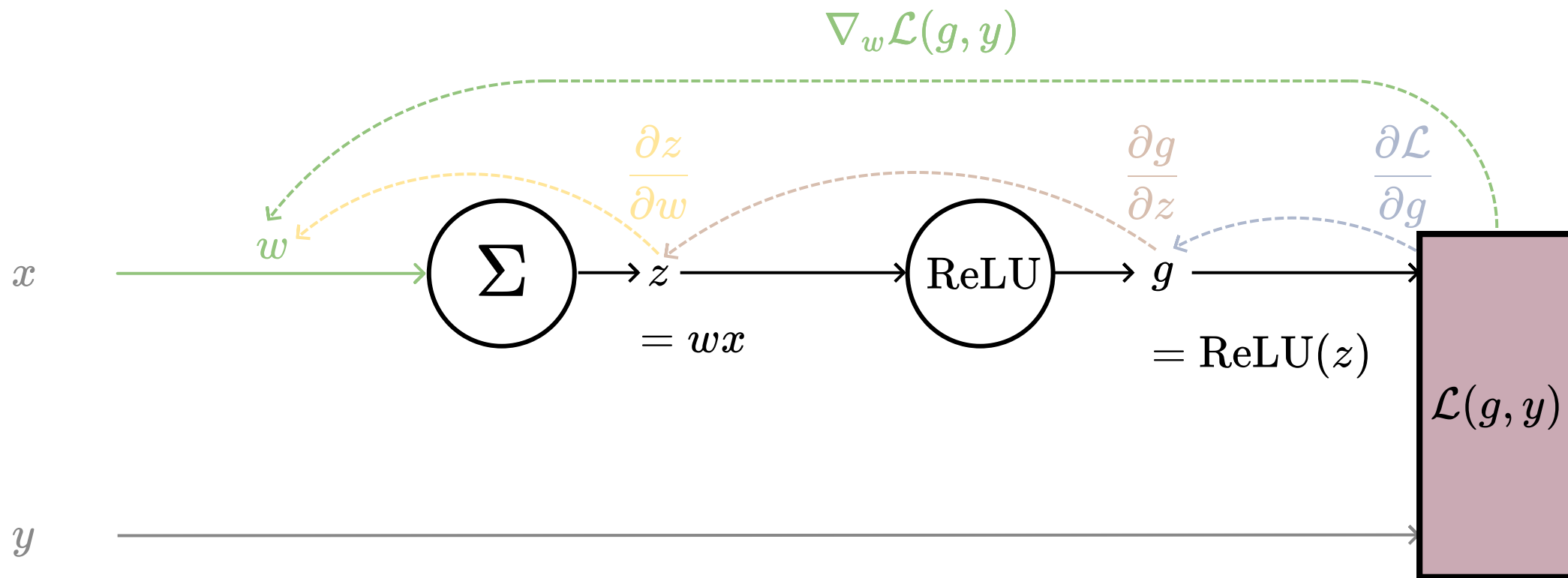
$$\nabla_w \mathcal{L}(g, y) = \frac{\partial \mathcal{L}(g, y)}{\partial w} = \frac{\partial g}{\partial w} \cdot \frac{\partial \mathcal{L}}{\partial g} = \frac{\partial g}{\partial w} \cdot \frac{\partial (g - y)^2}{\partial g} = \frac{\partial (wx)}{\partial w} \cdot 2(g - y) = x \cdot 2(g - y)$$

walk-through on board

Slightly more interesting:

1-d example:

$x = 3, y = 6, w = 4$



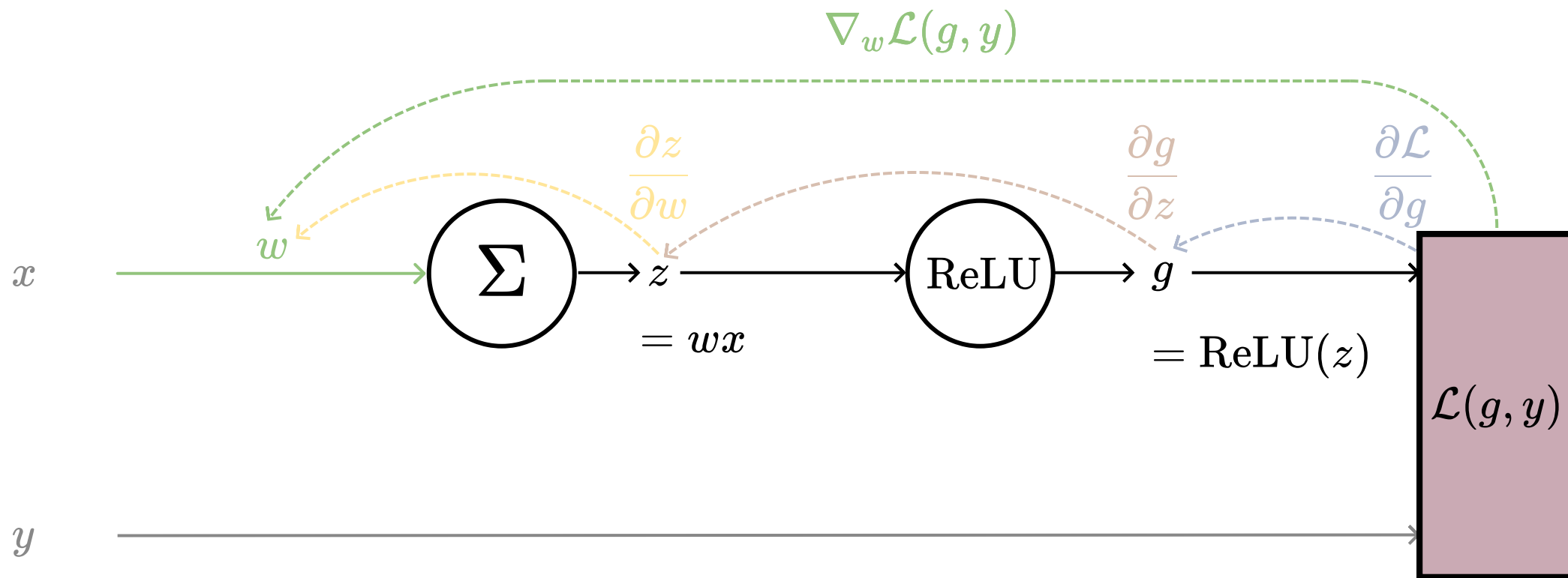
$$\nabla_w \mathcal{L}(g, y) = \frac{\partial \mathcal{L}(g, y)}{\partial w} = \frac{\partial z}{\partial w} \cdot \frac{\partial g}{\partial z} \cdot \frac{\partial \mathcal{L}}{\partial g} = \frac{\partial z}{\partial w} \cdot \frac{\partial g}{\partial z} \cdot \frac{\partial (g - y)^2}{\partial g} = x \cdot \frac{\partial[(\text{ReLU}(z))]}{\partial z} \cdot 2(g - y)$$

walk-through on board

Slightly more interesting:

1-d example:

$x = 3, y = 6, w = 4$

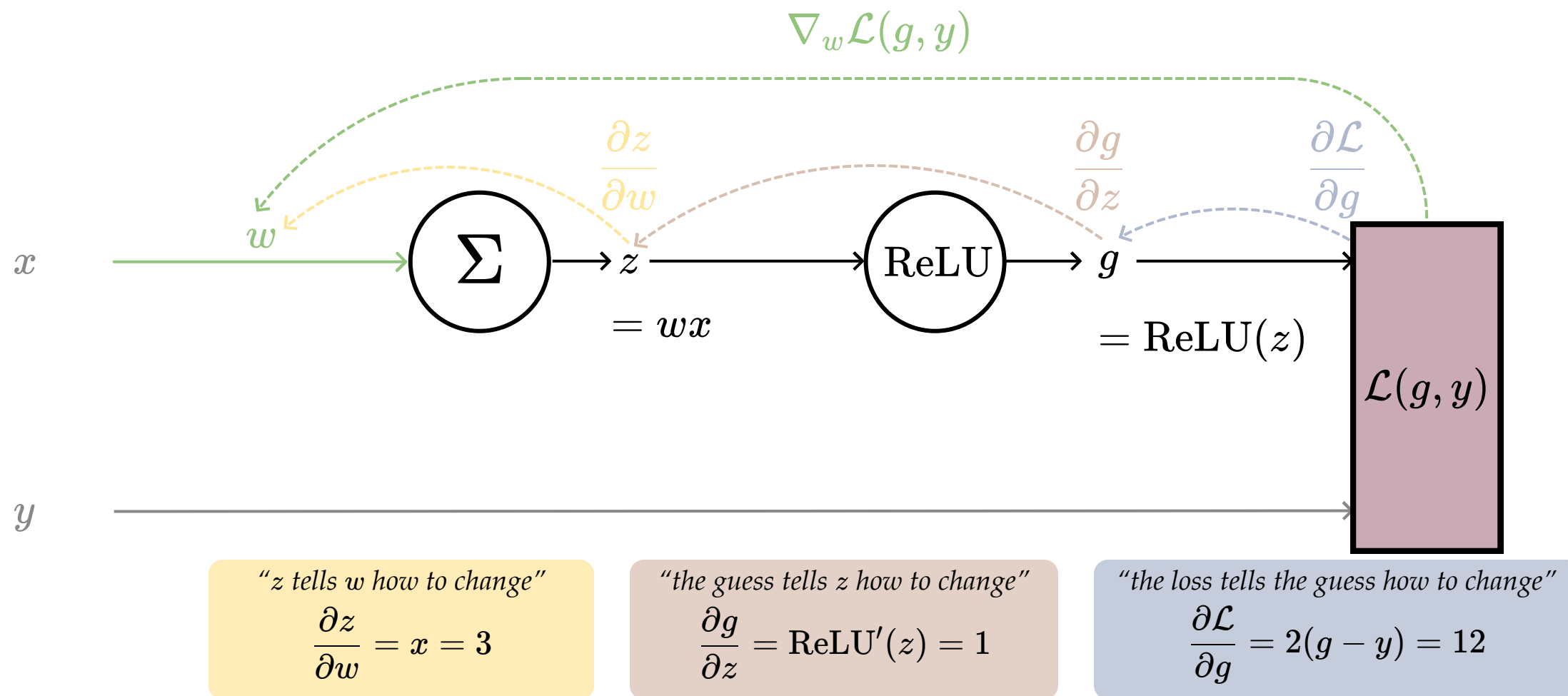


$$\nabla_w \mathcal{L}(g, y) = x \cdot \frac{\partial[(\text{ReLU}(z))]}{\partial z} \cdot 2(g - y) = \begin{cases} 0, & \text{if } z < 0 \\ 2x(g - y), & \text{otherwise} \end{cases}$$

$$= \begin{cases} 0, & \text{if } z < 0 \\ 1, & \text{otherwise} \end{cases}$$

walk-through on board

The middlemen



one partial derivative per hop,
traced backwards along the path, from the loss to the variable of interest

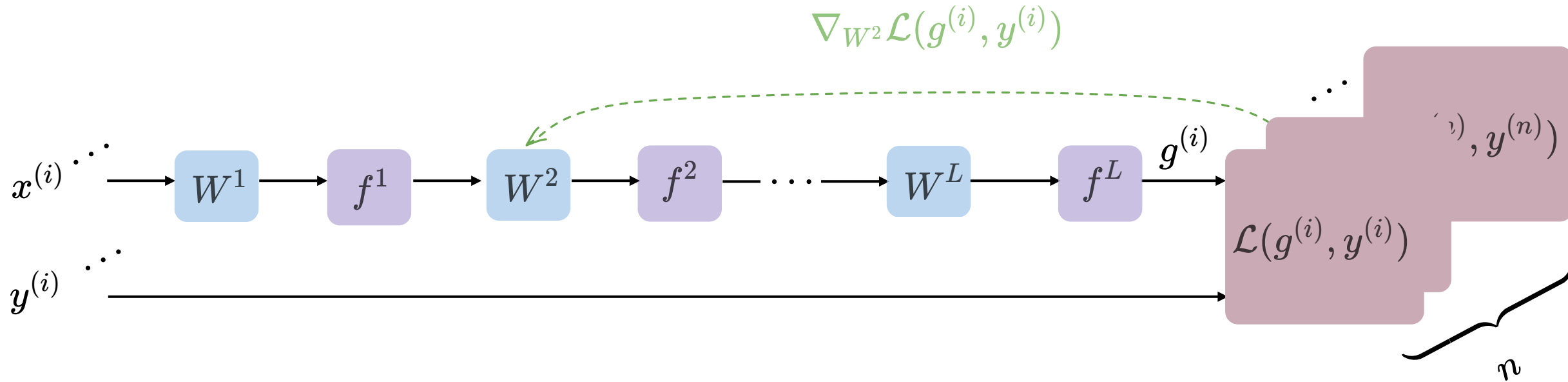
Outline

- Backward pass (to learn parameters/weights)
 - Backpropagation (gradient descent & the chain rule)
 - Recursive reuse of computation
 - Practical gradient issues and remedies

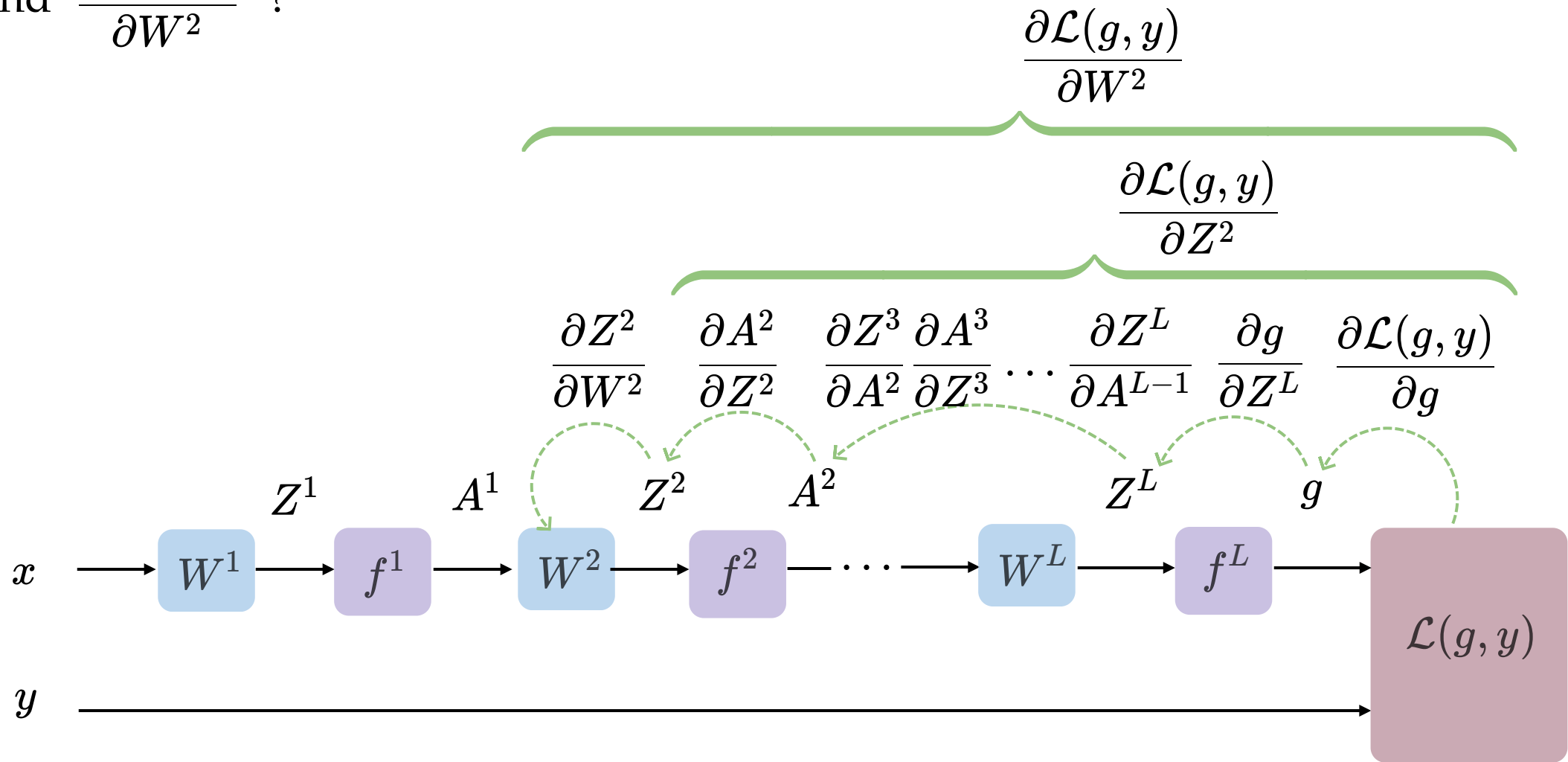
Backward pass: run SGD to update all parameters

e.g. to update W^2

- Randomly pick a data point $(x^{(i)}, y^{(i)})$
- Evaluate the gradient $\nabla_{W^2} \mathcal{L}(g^{(i)}, y^{(i)})$
- Update the weights $W^2 \leftarrow W^2 - \eta \nabla_{W^2} \mathcal{L}(g^{(i)}, y^{(i)})$

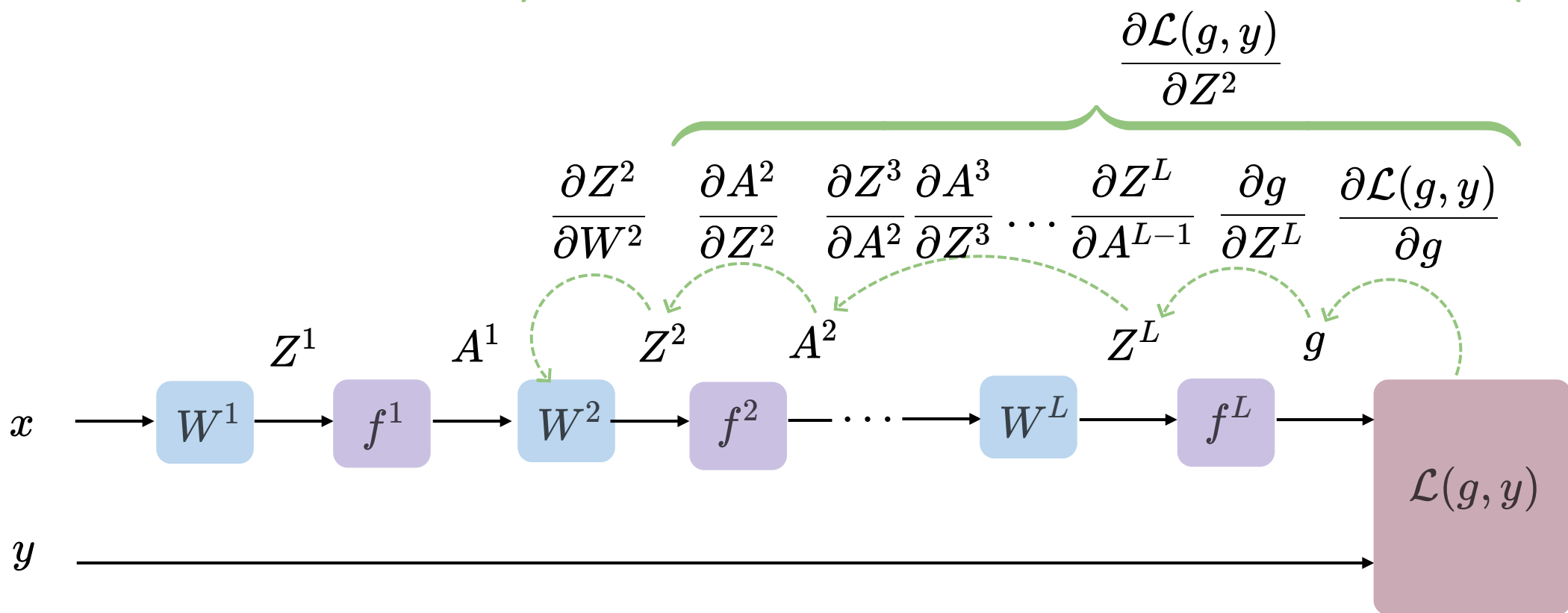


how to find $\frac{\partial \mathcal{L}(g, y)}{\partial W^2}$?



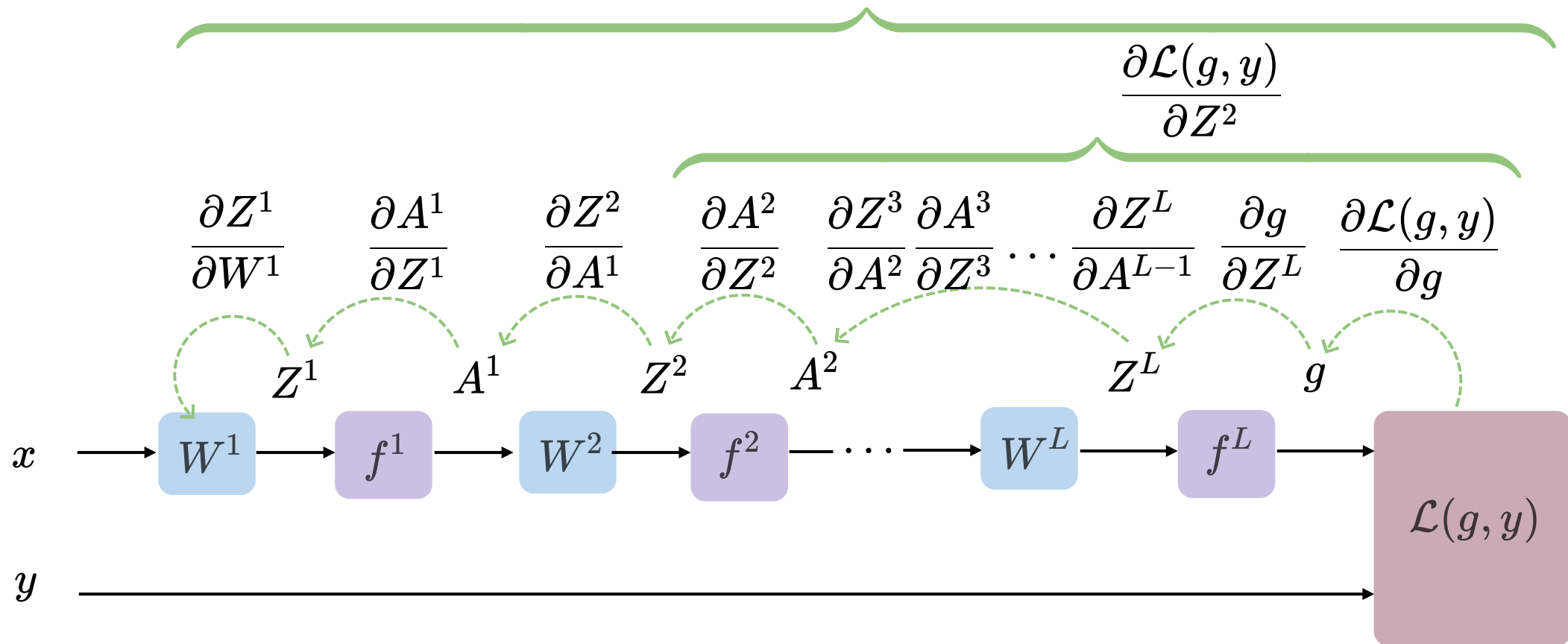
how to find $\frac{\partial \mathcal{L}(g, y)}{\partial W^1}$?

Previously, we found $\frac{\partial \mathcal{L}(g, y)}{\partial W^2}$

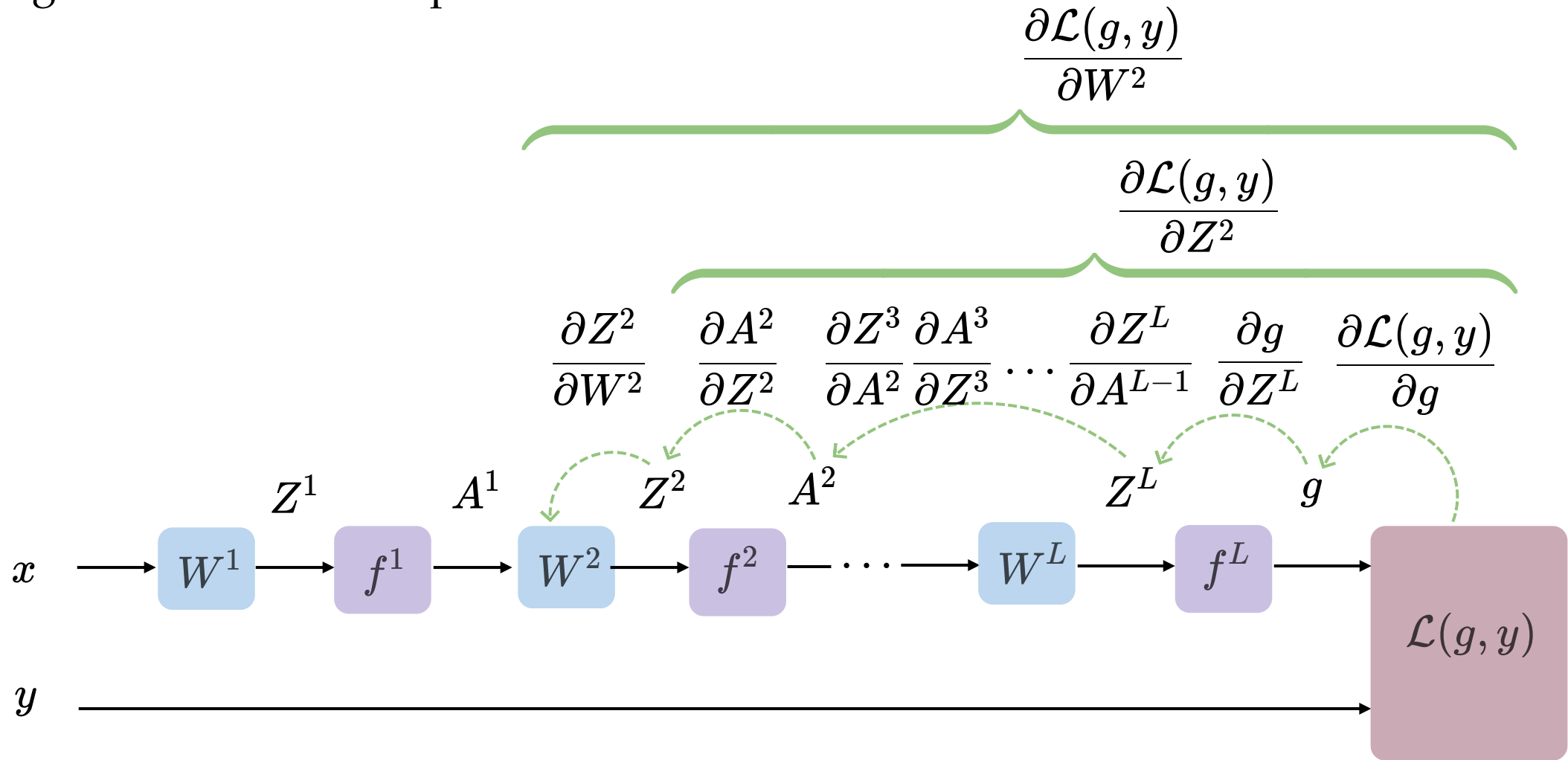


how to find $\frac{\partial \mathcal{L}(g, y)}{\partial W^1}$

Now $\frac{\partial \mathcal{L}(g, y)}{\partial W^1}$



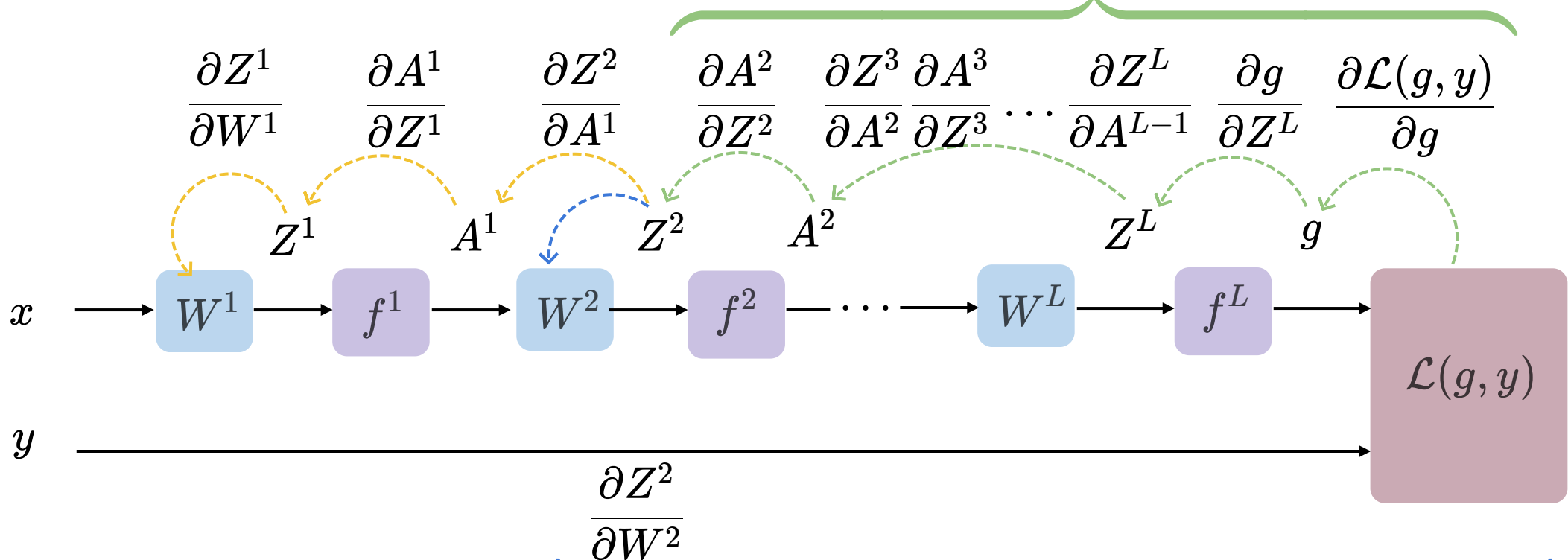
backpropagation: reuse of computation



backpropagation: reuse of computation

$$\frac{\partial \mathcal{L}}{\partial W^1} = \frac{\partial Z^1}{\partial W^1} \cdot \frac{\partial A^1}{\partial Z^1} \cdot \frac{\partial Z^2}{\partial A^1} \cdot \frac{\partial \mathcal{L}}{\partial Z^2}$$

$$\frac{\partial \mathcal{L}(g, y)}{\partial Z^2}$$



$$\frac{\partial \mathcal{L}}{\partial W^2} = \frac{\partial Z^2}{\partial W^2} \cdot \frac{\partial \mathcal{L}}{\partial Z^2}$$

Training a neural network: the full loop

Initialize all weights $W^1, W^2, \dots, W^L$ randomly

Repeat until stopping criterion:

1. **Forward pass:** pick a data point $(x^{(i)}, y^{(i)})$, compute $Z^1, A^1, Z^2, A^2, \dots, g^{(i)}$
2. **Evaluate loss:** compute $\mathcal{L}(g^{(i)}, y^{(i)})$
3. **One full backward pass:** compute $\nabla_{W^\ell} \mathcal{L}(g^{(i)}, y^{(i)})$ for every $\ell = L, \dots, 1$, reusing cached results

e.g., three layers:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial W^3} &= \frac{\partial Z^3}{\partial W^3} \cdot \frac{\partial g}{\partial Z^3} \cdot \frac{\partial \mathcal{L}}{\partial g} \\ \frac{\partial \mathcal{L}}{\partial W^2} &= \frac{\partial Z^2}{\partial W^2} \cdot \frac{\partial A^2}{\partial Z^2} \cdot \frac{\partial Z^3}{\partial A^2} \cdot \frac{\partial g}{\partial Z^3} \cdot \frac{\partial \mathcal{L}}{\partial g} \\ \frac{\partial \mathcal{L}}{\partial W^1} &= \frac{\partial Z^1}{\partial W^1} \cdot \frac{\partial A^1}{\partial Z^1} \cdot \frac{\partial Z^2}{\partial A^1} \cdot \frac{\partial A^2}{\partial Z^2} \cdot \frac{\partial Z^3}{\partial A^2} \cdot \frac{\partial g}{\partial Z^3} \cdot \frac{\partial \mathcal{L}}{\partial g}\end{aligned}$$

4. **Update all weights at once:** $W^\ell \leftarrow W^\ell - \eta \nabla_{W^\ell} \mathcal{L}$ for all ℓ

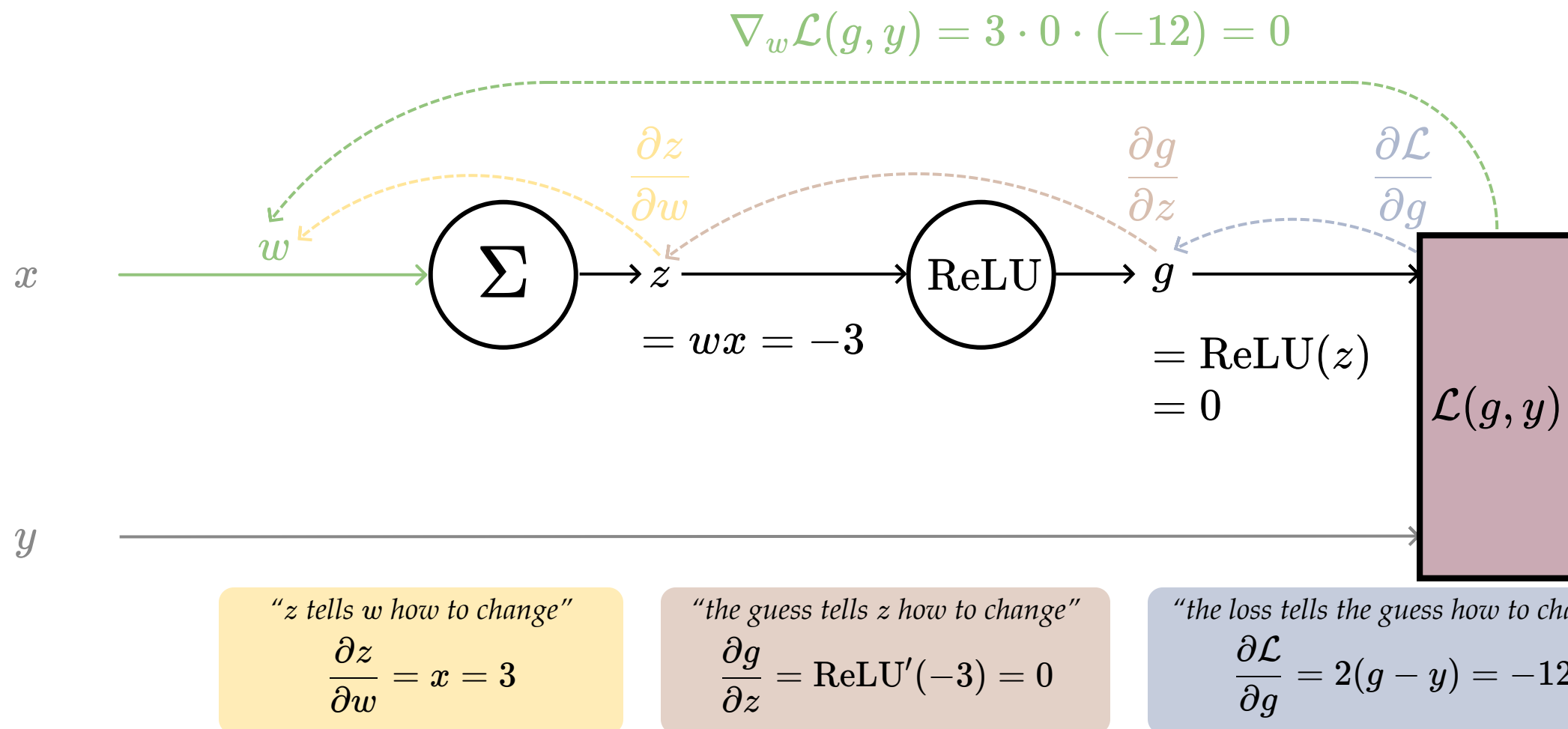
Outline

- Backward pass (to learn parameters/weights)
 - Backpropagation (gradient descent & the chain rule)
 - Recursive reuse of computation
 - Practical gradient issues and remedies

The flat branch

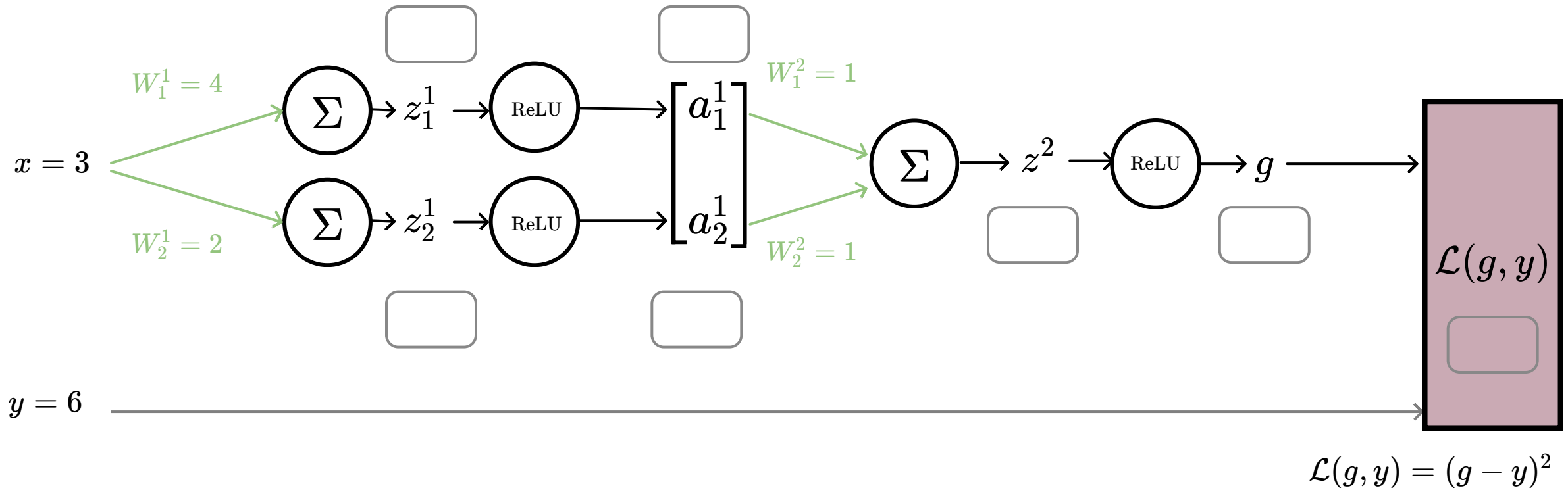
1-d example:

$$x = 3, y = 6, w = -1$$



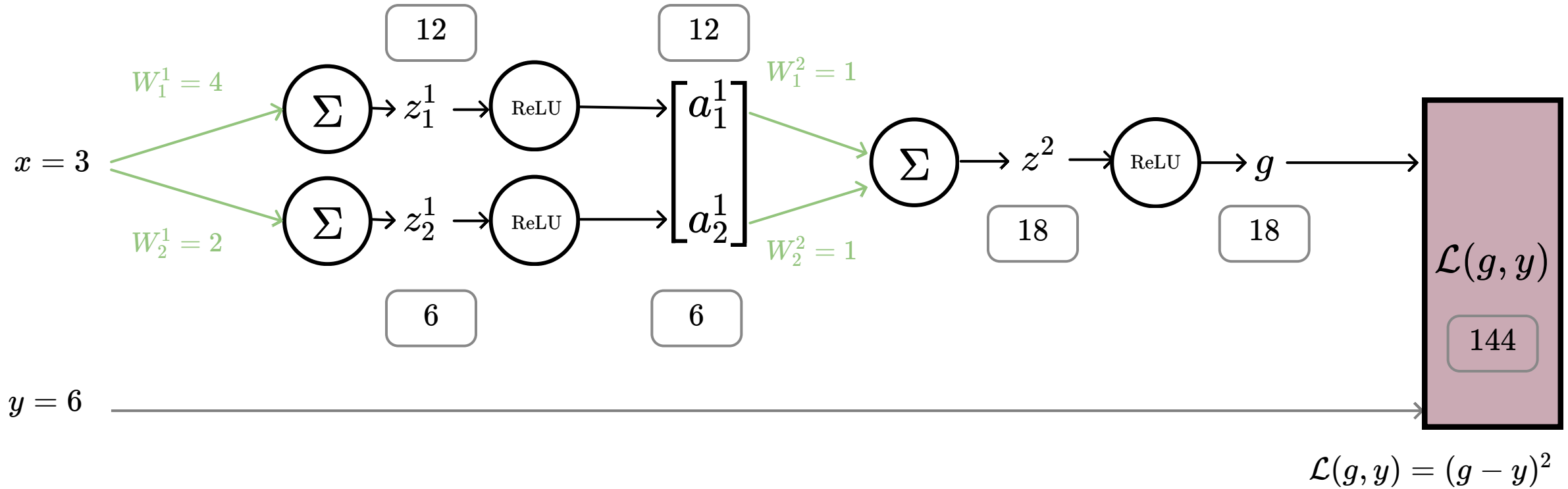
on the flat branch the ReLU's slope is 0, and one zero factor zeroes the whole product
at $w = 4$ (The middlemen): $3 \cdot 1 \cdot 12 = 36$

More neurons, more layers



forward pass: evaluation

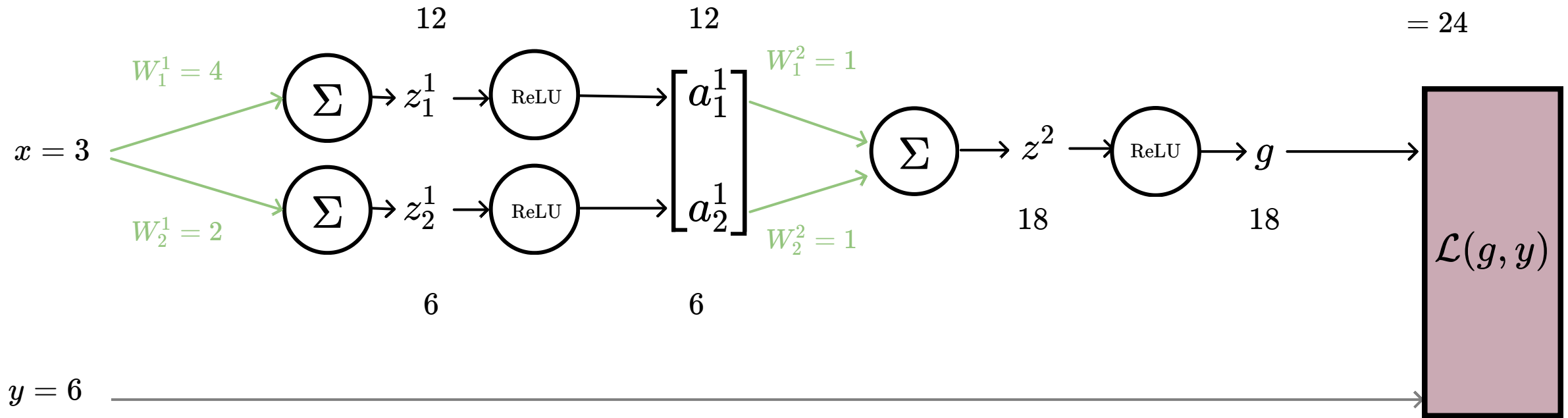
More neurons, more layers



forward pass: evaluation

The backward pass

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial g} &= 2(g - y) \\ &= 2(18 - 6) \\ &= 24\end{aligned}$$



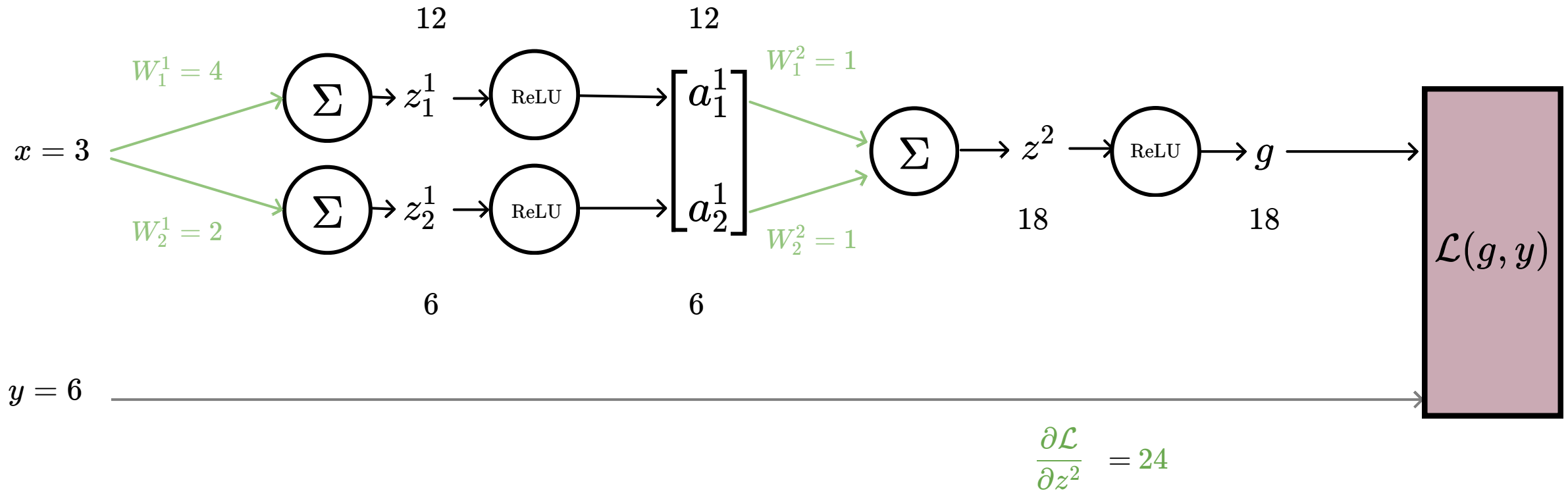
$$\frac{\partial \mathcal{L}}{\partial z^2} = \frac{\partial g}{\partial z^2} \cdot \frac{\partial \mathcal{L}}{\partial g} = 1 \cdot 24 = 24$$

gradient w.r.t. 2nd layer weights:

$$\frac{\partial \mathcal{L}}{\partial W_1^2} = \frac{\partial z^2}{\partial W_1^2} \cdot \frac{\partial \mathcal{L}}{\partial z^2} = 12 \cdot 24 = 288$$

$$\frac{\partial \mathcal{L}}{\partial W_2^2} = \frac{\partial z^2}{\partial W_2^2} \cdot \frac{\partial \mathcal{L}}{\partial z^2} = 6 \cdot 24 = 144$$

The backward pass



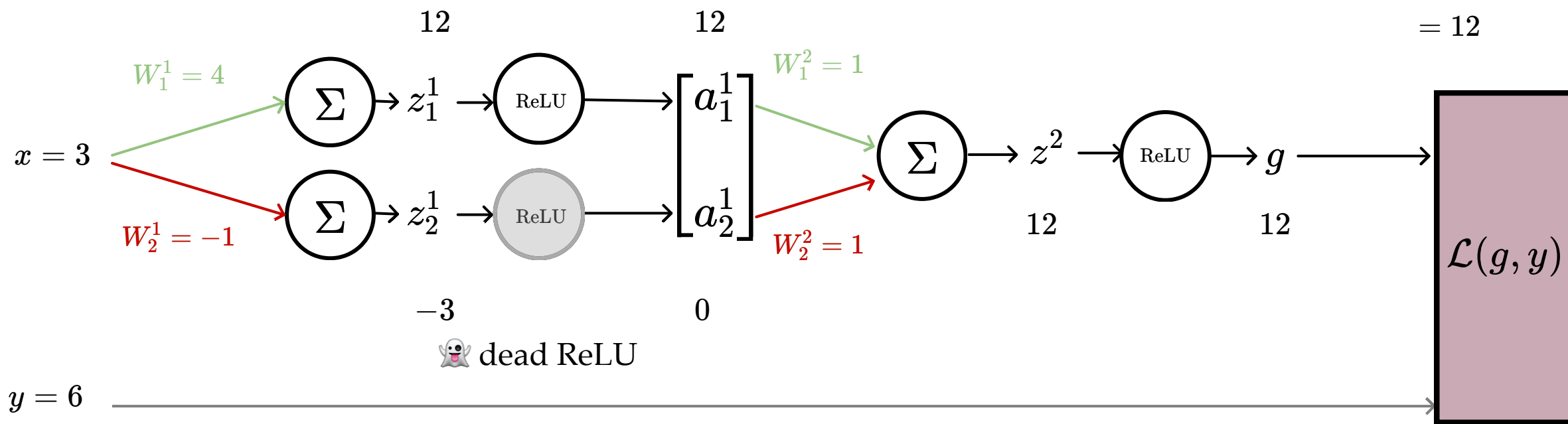
gradient w.r.t. 1st layer weights:

$$\frac{\partial \mathcal{L}}{\partial W_1^1} = \frac{\partial z_1^1}{\partial W_1^1} \cdot \frac{\partial a_1^1}{\partial z_1^1} \cdot \frac{\partial z^2}{\partial a_1^1} \cdot \frac{\partial \mathcal{L}}{\partial z^2} = 3 \cdot 1 \cdot 1 \cdot 24 = 72$$

$$\frac{\partial \mathcal{L}}{\partial W_2^1} = \frac{\partial z_2^1}{\partial W_2^1} \cdot \frac{\partial a_2^1}{\partial z_2^1} \cdot \frac{\partial z^2}{\partial a_2^1} \cdot \frac{\partial \mathcal{L}}{\partial z^2} = 3 \cdot 1 \cdot 1 \cdot 24 = 72$$

if ReLU input is negative, some weights won't update

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial g} &= 2(g - y) \\ &= 2(12 - 6) \\ &= 12\end{aligned}$$



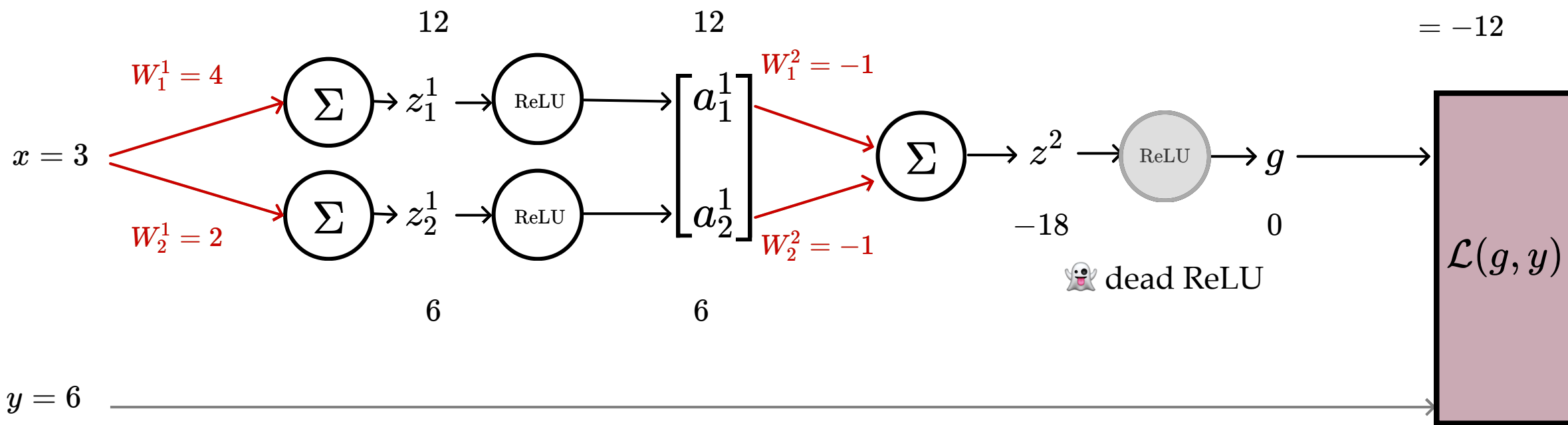
$$\frac{\partial \mathcal{L}}{\partial z^2} = \frac{\partial g}{\partial z^2} \cdot \frac{\partial \mathcal{L}}{\partial g} = 1 \cdot 12 = 12$$

gradient w.r.t. red weights:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial W_2^2} &= \frac{\partial z^2}{\partial W_2^2} \cdot \frac{\partial \mathcal{L}}{\partial z^2} = 0 \cdot 12 = 0 \\ \frac{\partial \mathcal{L}}{\partial W_2^1} &= \frac{\partial z_2^1}{\partial W_2^1} \cdot \frac{\partial a_2^1}{\partial z_2^1} \cdot \frac{\partial z^2}{\partial a_2^1} \cdot \frac{\partial \mathcal{L}}{\partial z^2} = 3 \cdot 0 \cdot 1 \cdot 12 = 0\end{aligned}$$

if $z^2 < 0$, no weights get updated

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial g} &= 2(g - y) \\ &= 2(0 - 6) \\ &= -12\end{aligned}$$



$y = 6$

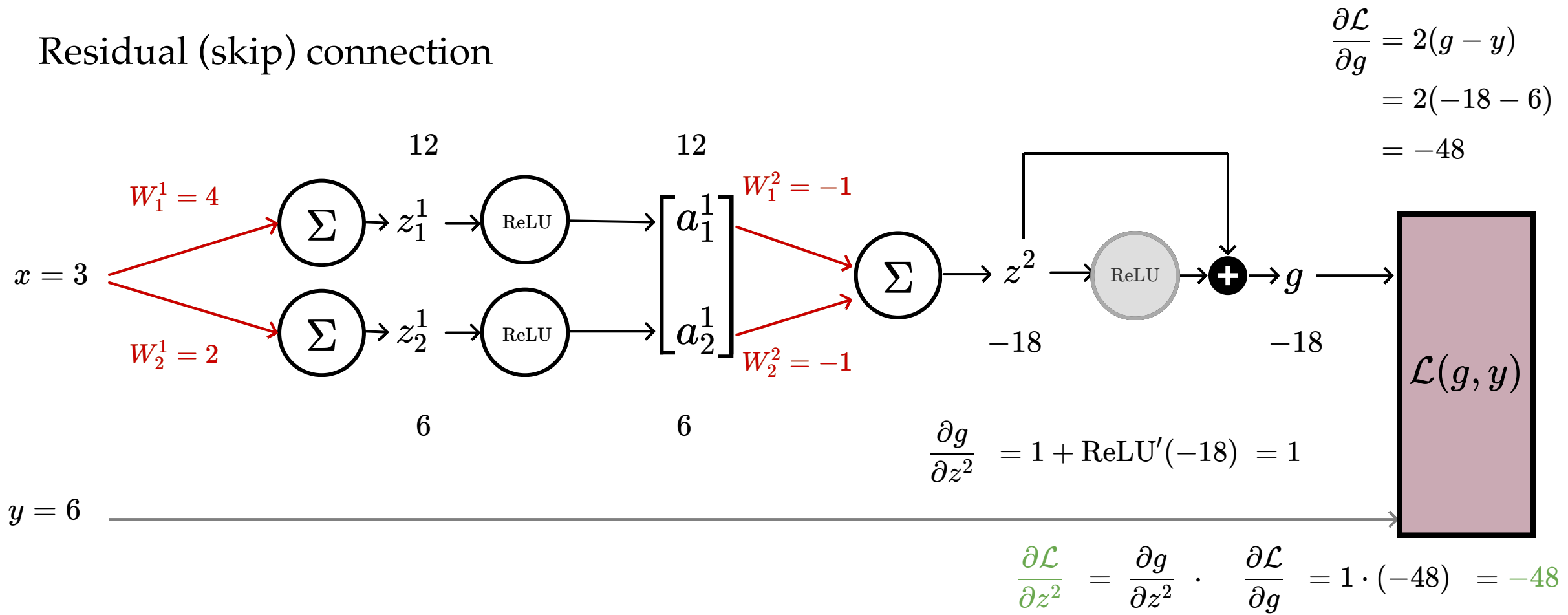
$$\frac{\partial \mathcal{L}}{\partial z^2} = \frac{\partial g}{\partial z^2} \cdot \frac{\partial \mathcal{L}}{\partial g} = 0 \cdot (-12) = 0$$

gradient w.r.t. all weights:

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial W_k^1} &= \frac{\partial z_k^1}{\partial W_k^1} \cdot \frac{\partial a_k^1}{\partial z_k^1} \cdot \frac{\partial z^2}{\partial a_k^1} \cdot \frac{\partial \mathcal{L}}{\partial z^2} = 3 \cdot 1 \cdot (-1) \cdot 0 = 0 \\ \frac{\partial \mathcal{L}}{\partial W_k^2} &= \frac{\partial z_k^2}{\partial W_k^2} \cdot \frac{\partial \mathcal{L}}{\partial z^2} = a_k^1 \cdot 0 = 0\end{aligned}$$

$k = 1$ or 2 , the hidden neuron the weight connects to, so each row covers both weights of its layer

Residual (skip) connection



gradient w.r.t. all weights:

$$\frac{\partial \mathcal{L}}{\partial W_k^1} = \frac{\partial z_k^1}{\partial W_k^1} \cdot \frac{\partial a_k^1}{\partial z_k^1} \cdot \frac{\partial z^2}{\partial a_k^1} \cdot \frac{\partial \mathcal{L}}{\partial z^2} = 3 \cdot 1 \cdot (-1) \cdot (-48) = 144$$

$$\frac{\partial \mathcal{L}}{\partial W_k^2} = \frac{\partial z_k^2}{\partial W_k^2} \cdot \frac{\partial \mathcal{L}}{\partial z^2} = a_k^1 \cdot (-48)$$

even if $z^2 < 0$, with skip connection, weights in earlier layers can still get updated

Going deep: vanishing and exploding gradients

backpropagation computes gradients via the chain rule — a product of many factors:

$$\text{e.g. } \frac{\partial \mathcal{L}}{\partial W^1} = \frac{\partial Z^1}{\partial W^1} \cdot \underbrace{\frac{\partial A^1}{\partial Z^1} \cdot \frac{\partial Z^2}{\partial A^1} \cdots \frac{\partial A^{L-1}}{\partial Z^{L-1}} \cdot \frac{\partial Z^L}{\partial A^{L-1}}}_{L-1 \text{ layers of multiplicative factors}} \cdot \frac{\partial g}{\partial Z^L} \cdot \frac{\partial \mathcal{L}}{\partial g}$$

Factors below 1: the product shrinks fast with depth, so gradients **vanish**

Factors above 1: the product grows fast with depth, so gradients **explode**

Common remedies:

- residual connections
- gradient clipping (if $\|\nabla\| > \tau$, rescale $\nabla \leftarrow \tau \frac{\nabla}{\|\nabla\|}$, keeping the direction)
- careful initialization, normalization: both keep factors near 1

Summary

backpropagation backward pass backpropagation chain rule

practical issues dead ReLU skip connection vanishing / exploding gradient

- Backpropagation is gradient descent plus the chain rule: each “tells” is one partial derivative, and a weight’s gradient multiplies the factors between it and the loss.
- One backward sweep caches $\partial\mathcal{L}/\partial Z$ at each layer, and every earlier weight reuses it.
- One training step: forward pass, loss, backward pass, then all weights update at once (as in PyTorch).
- On its flat branch a ReLU passes zero gradient: a dead neuron freezes its edges, and a dead output freezes the whole network.
- Skip connections keep a slope-1 path for the gradient; long products of factors can still vanish or explode (clipping and careful initialization help).