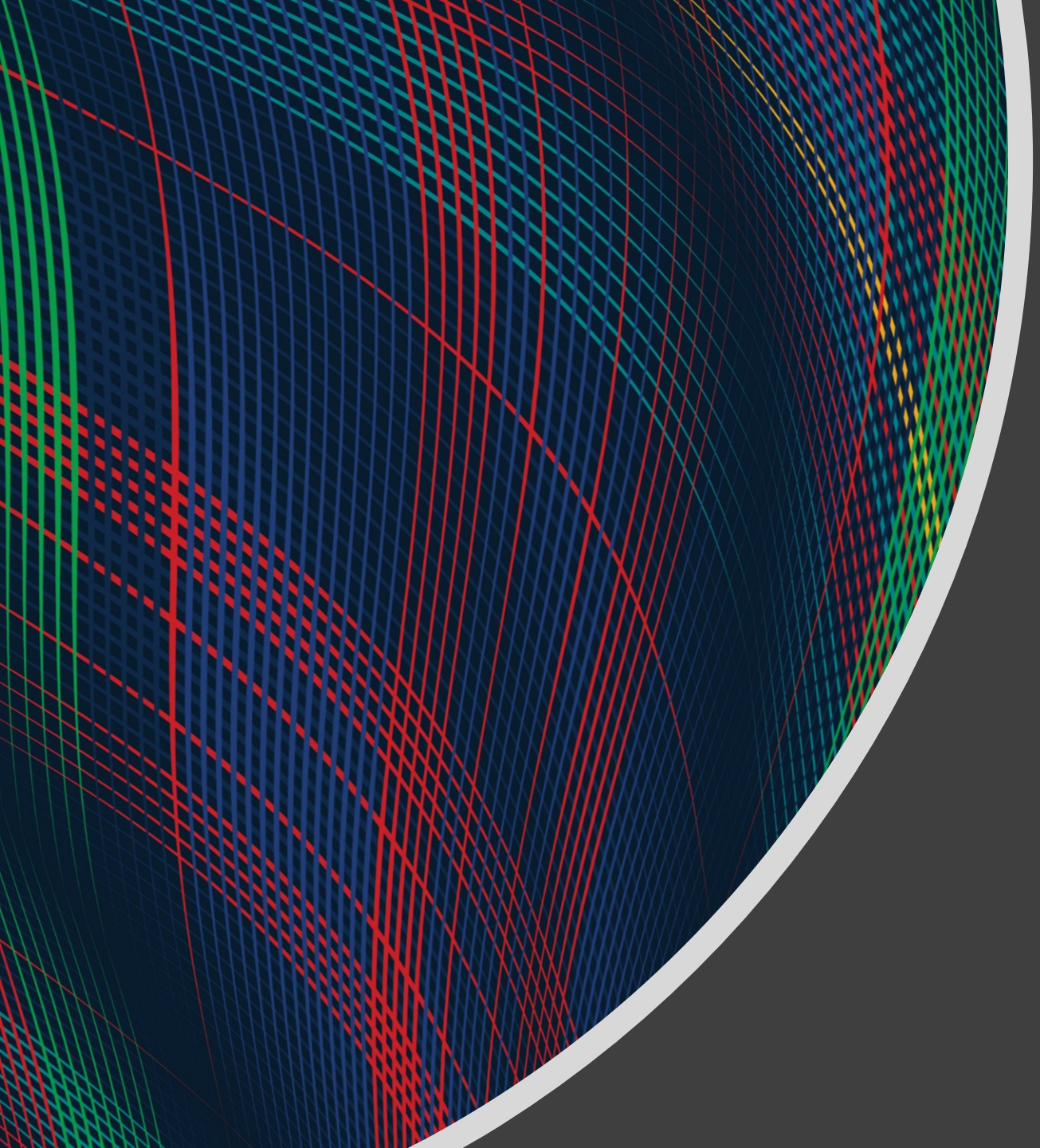


# 07-280
# AI & ML I

## Neural Networks II

Instructor:
Pat Virtue & Nihar Shah

# Outline

## Last time: Neural Networks

- Three-neuron Network
  - Optimization
  - Neural Network Intuition
- Forward pass and Backward pass (scalar version)

## Today: Neural Networks

- Neural Network Properties and Intuition
- Backpropagation (scalar version)
- Calculus: multi-variate chain rule
- Backpropagation (vector version)

# Neural Net
# Properties and Intuition

# Neural Networks Questions

Can we fit any function?

Non-convex? What about local minima?

What if there are no non-linear activations?

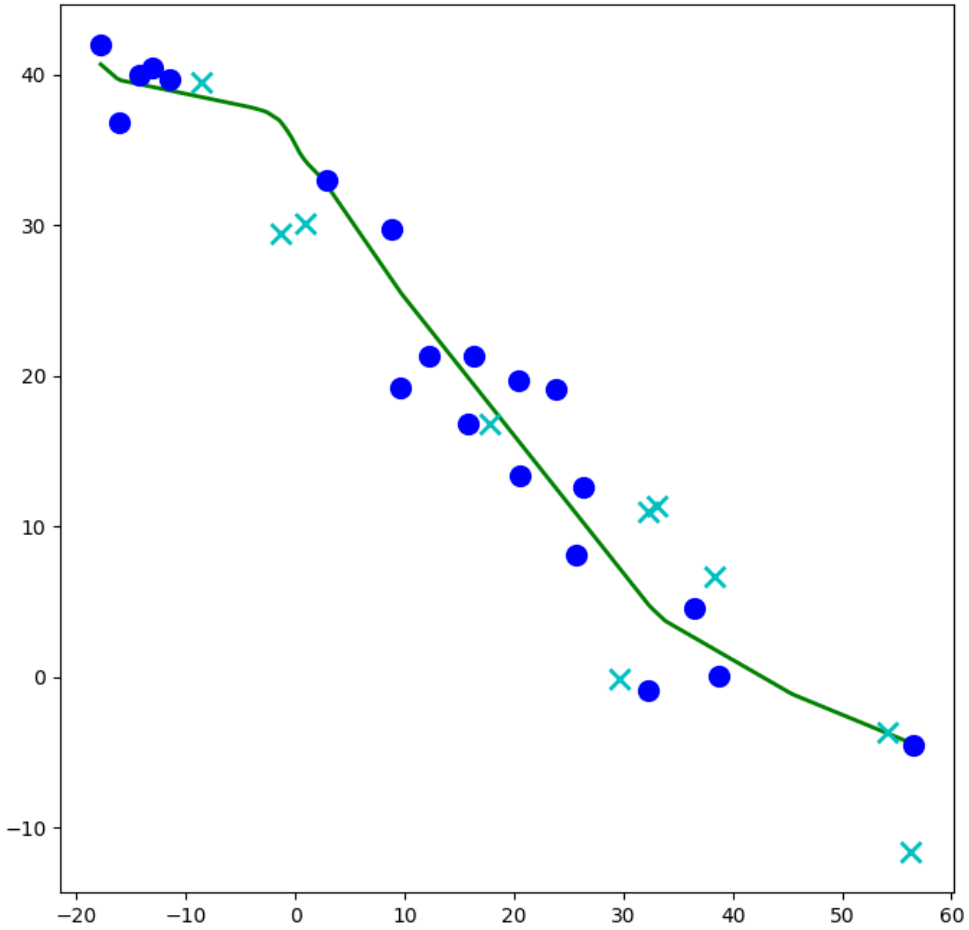
How many neurons/layers should we add?

Please, tell me more about calculus

# Neural Network Properties

## Fitting complex functions (with 1-D input)

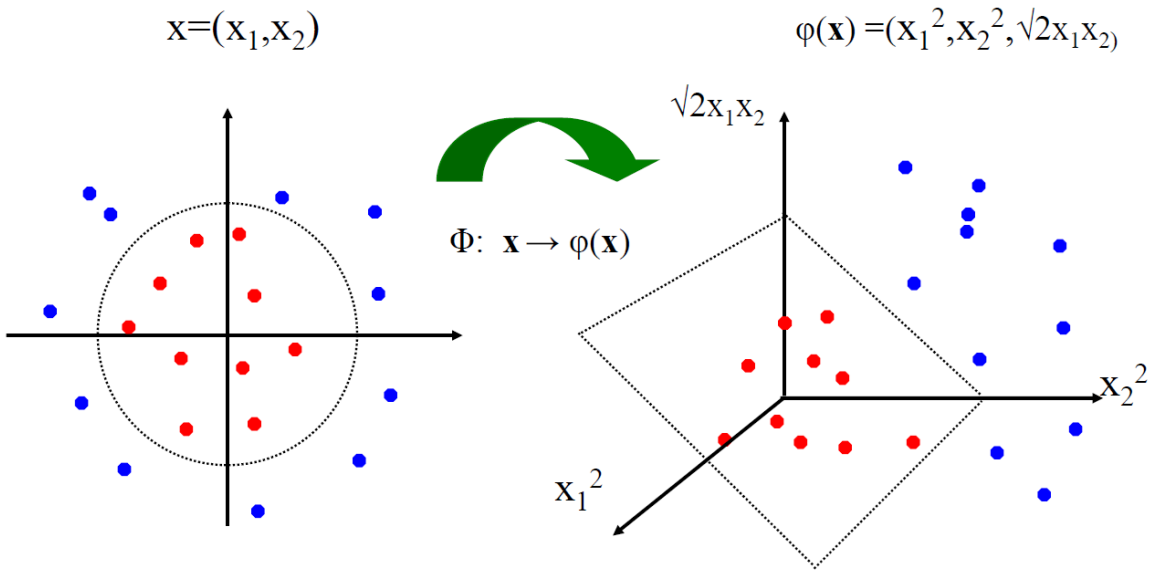
# Network to Approximate a 1-D Function



# Linear Classifiers for Nonlinear Data

Linear classifiers have linear decision boundaries

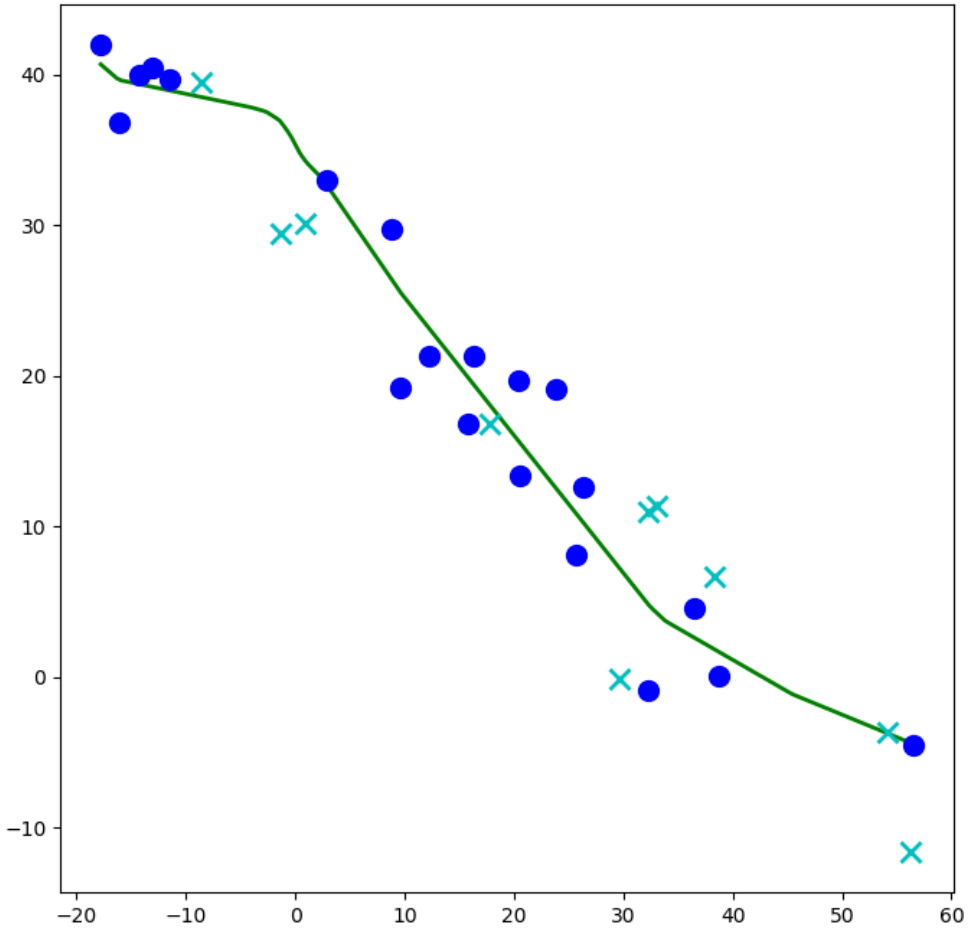
Feature mapping can convert nonlinear data to higher dimensions

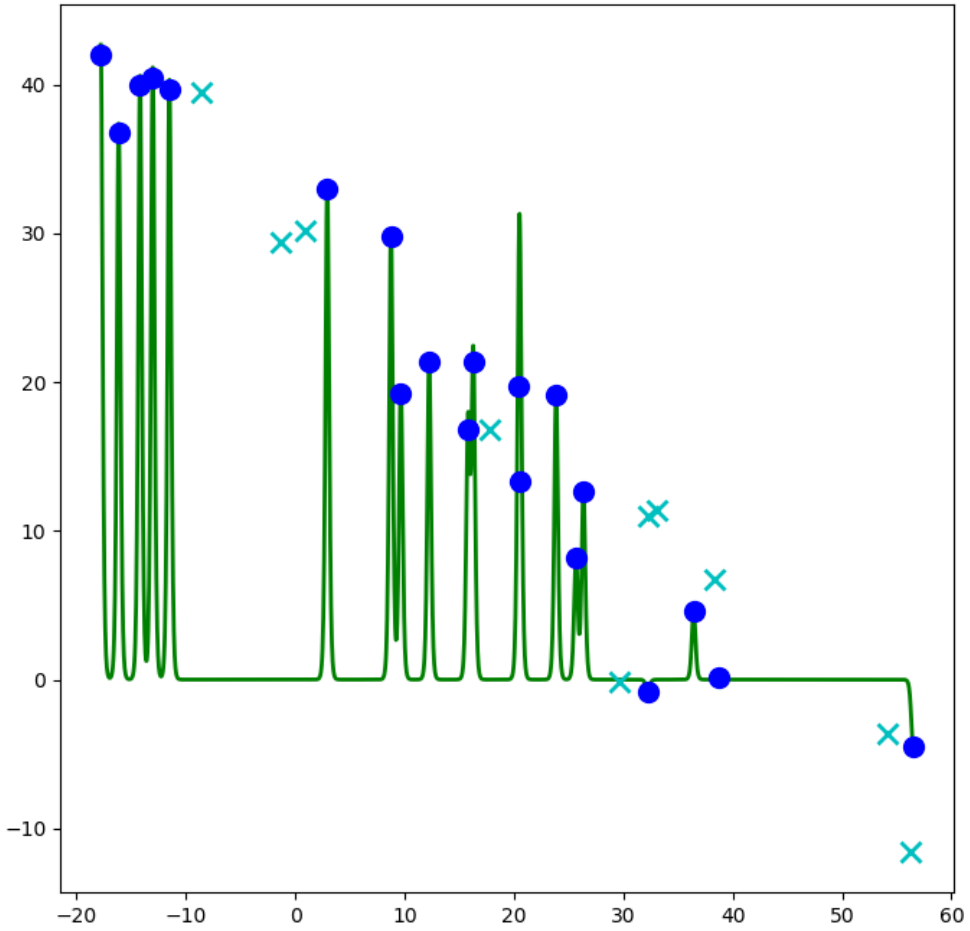



This slide is courtesy of *www.iro.umontreal.ca/~pift6080/documents/papers/**svm**_tutorial.**ppt***

Today, instead of choosing a feature mapping function, we'll use neural networks to learn nonlinear decision boundaries and regression functions

# Network to Approximate a 1-D Function



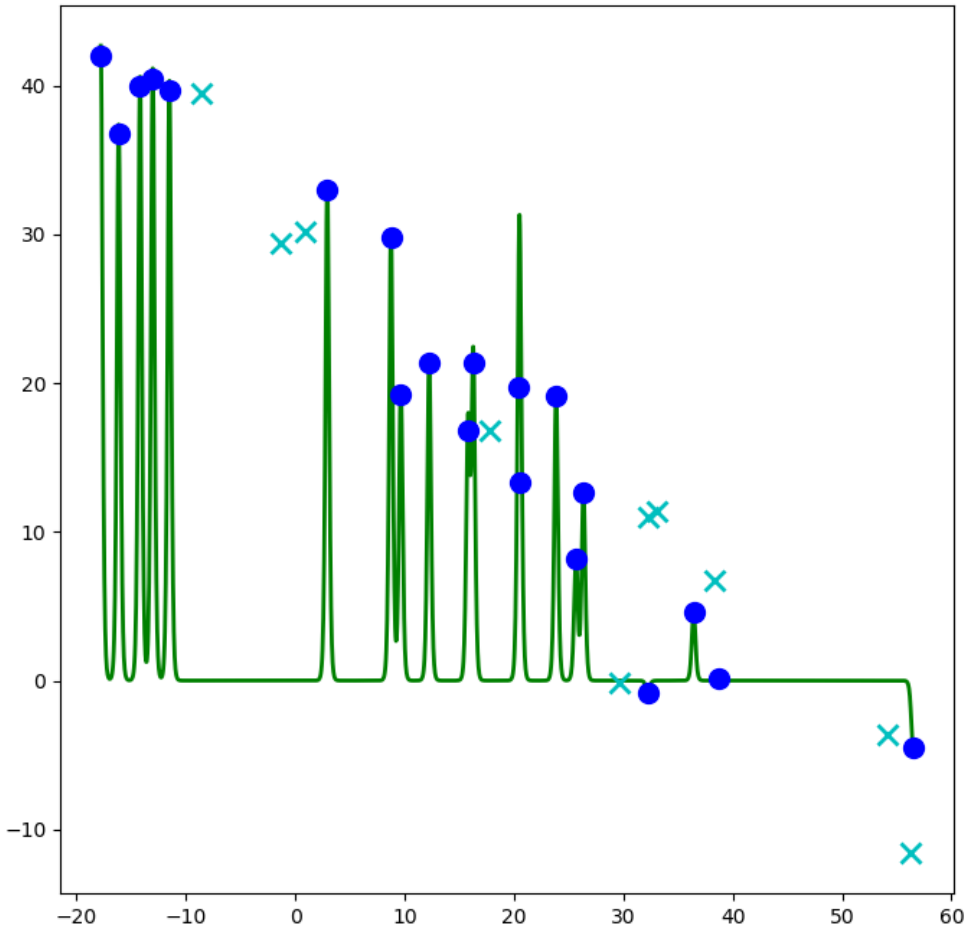



# Network to Approximate a 1-D Function

Design a network to approximate this function using:

Linear, Sigmoid, Step, or ReLU

# Network to Approximate a 1-D Function

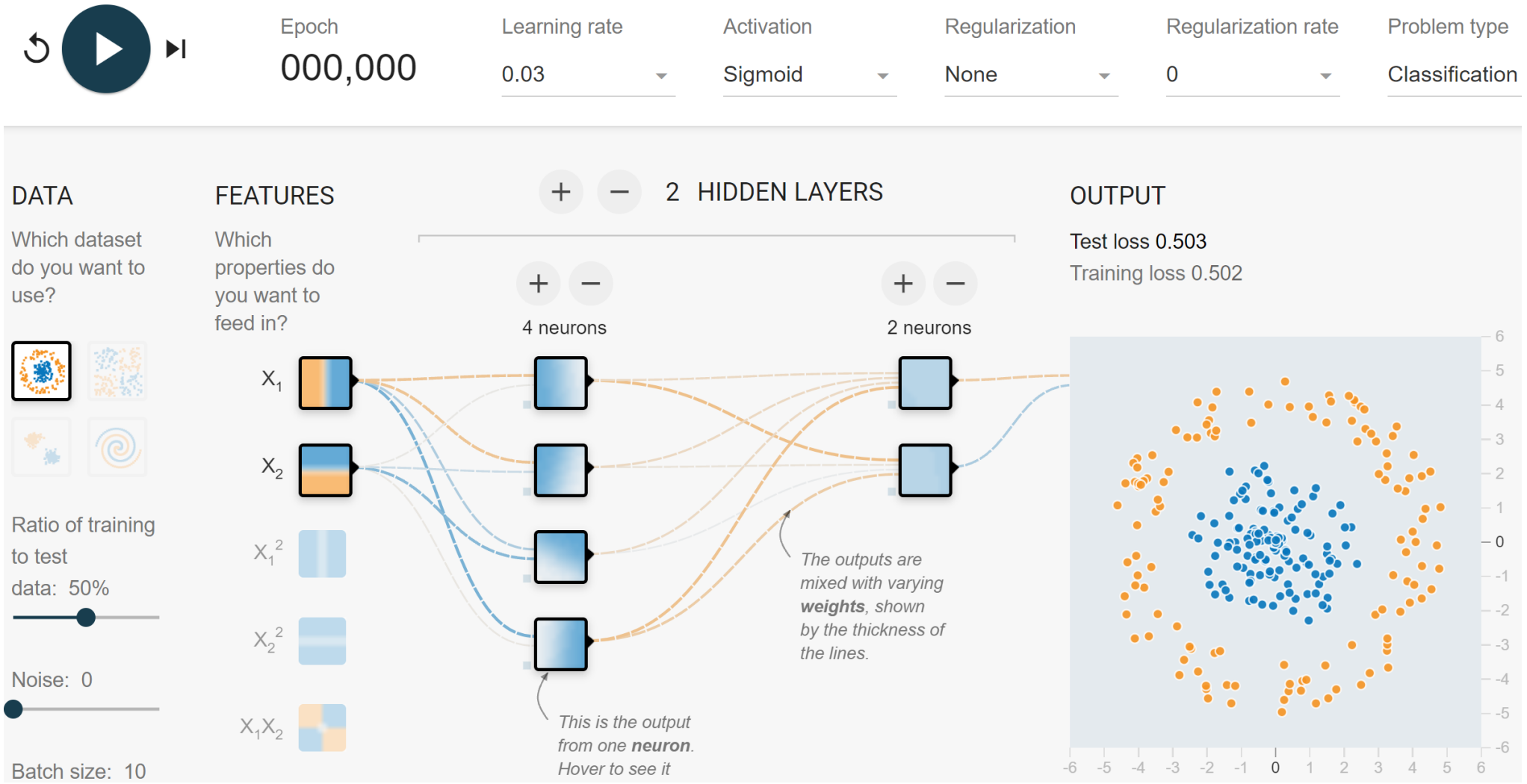


# Neural Network Properties

## Fitting complicated functions (with 2-D input)

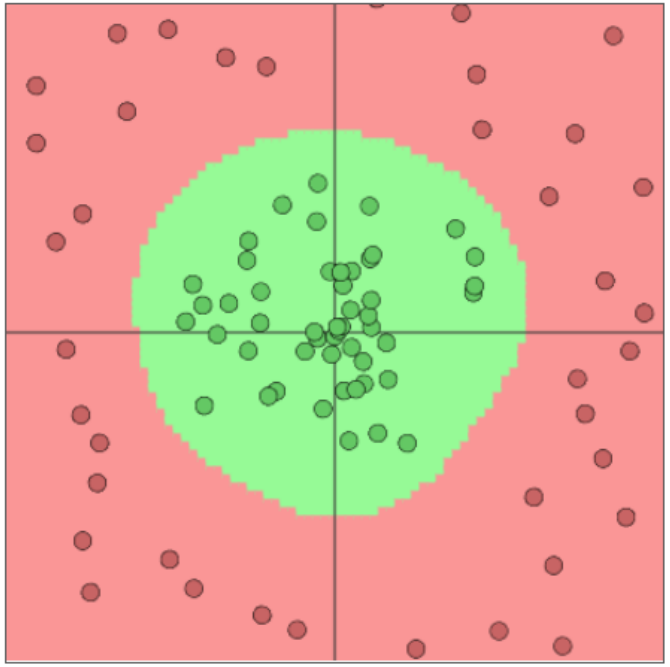
# Network to Approximate Binary Classification

https://playground.tensorflow.org/#activation=sigmoid



# Network to Approximate Binary Classification

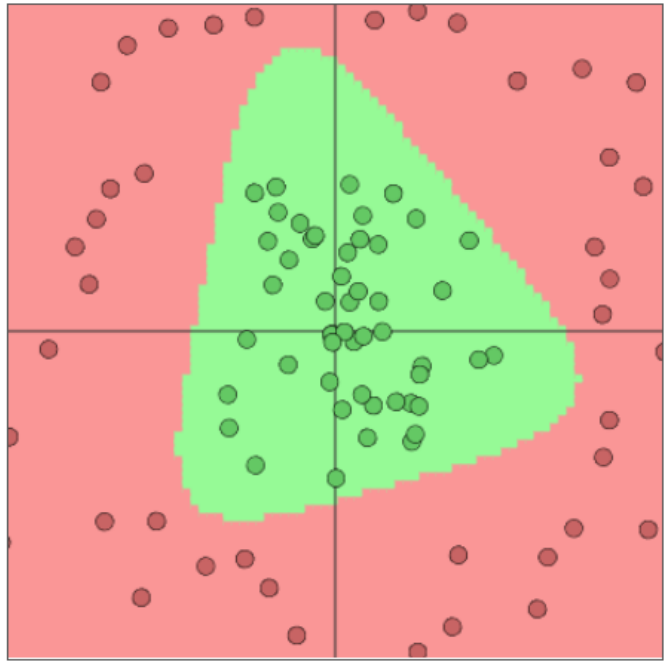
## Approximate arbitrary decision boundary

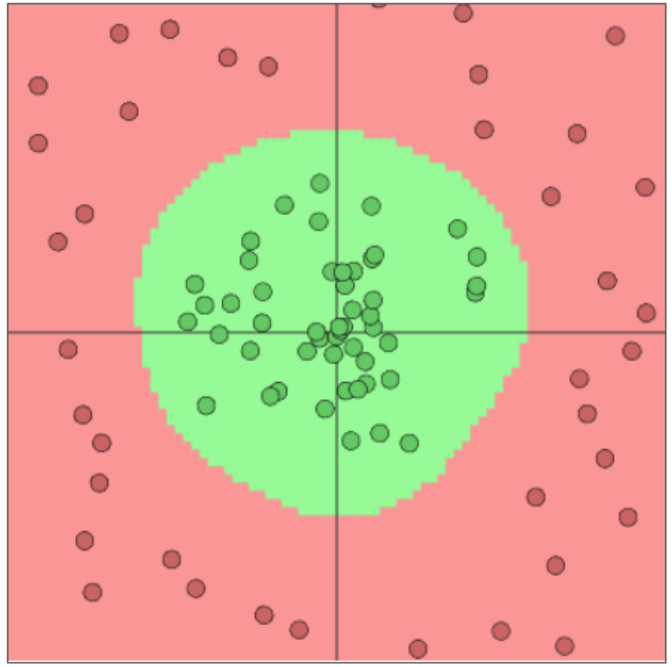


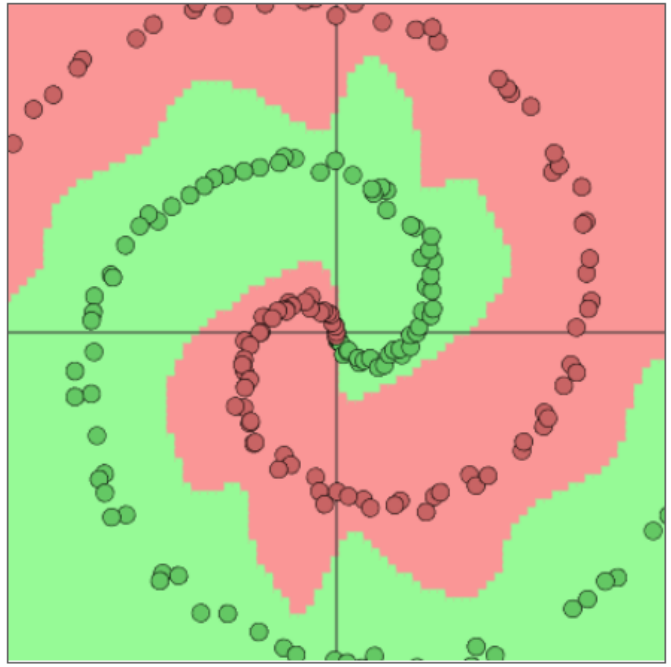
https://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html

# Network to Approximate Binary Classification

## Approximate arbitrary decision boundary



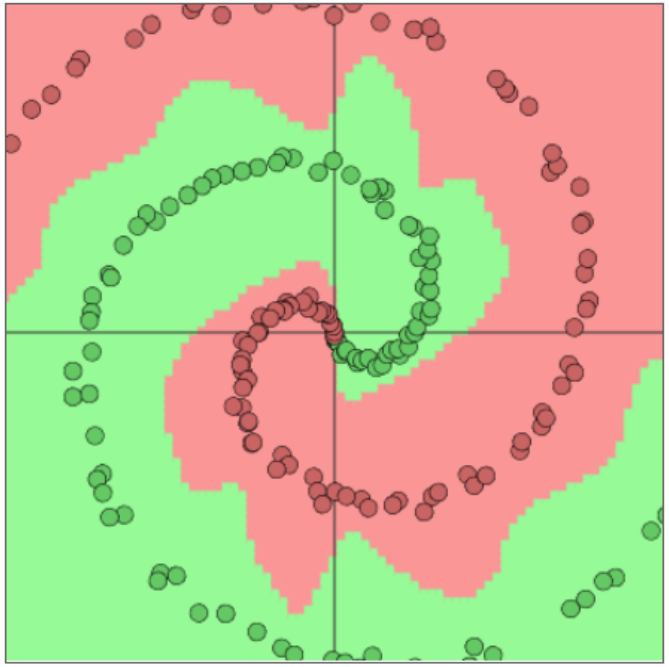




https://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html

# Network to Approximate Binary Classification

## Approximate arbitrary decision boundary



https://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html

# Neural Network Properties

## (Over?) Fitting any function

# Neural Networks Properties

## Universal Approximation Theorem

- A two-layer neural network with a sufficient number of neurons can approximate any continuous function to any desired accuracy

## Reminder

## Just because it can fit any function will it?

- Objective function is non-convex

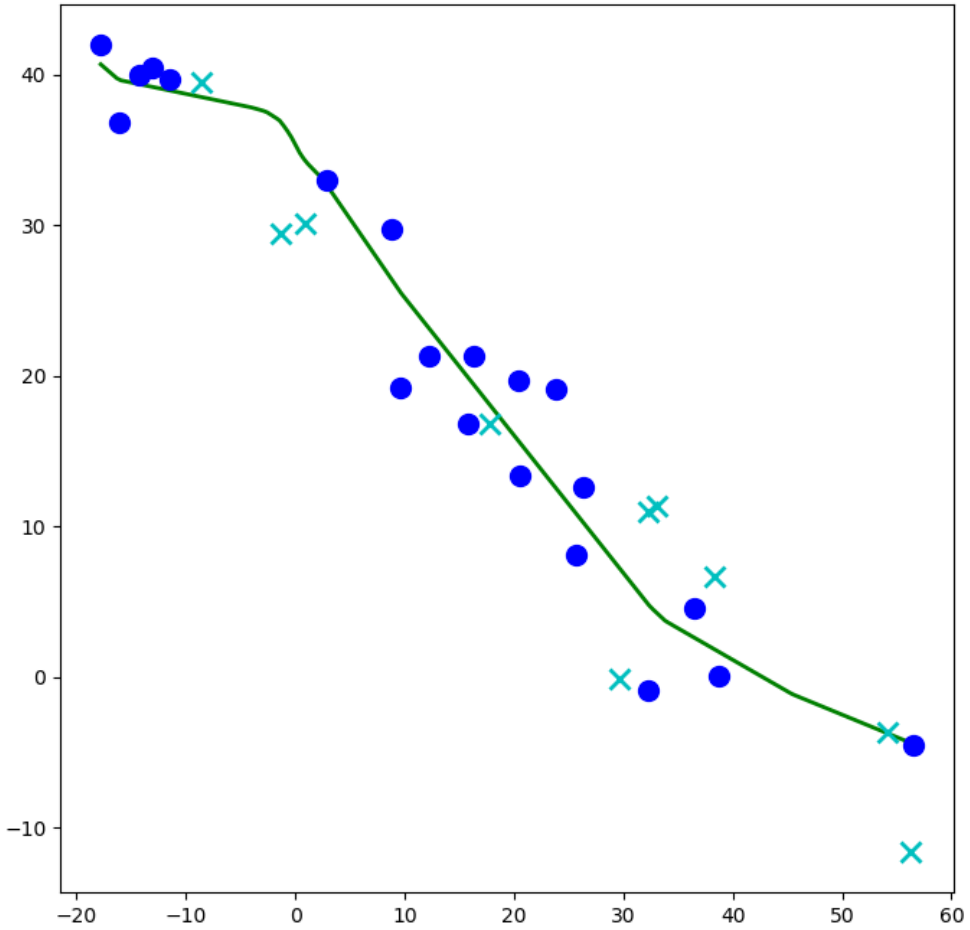
  → it will get stuck in local minima

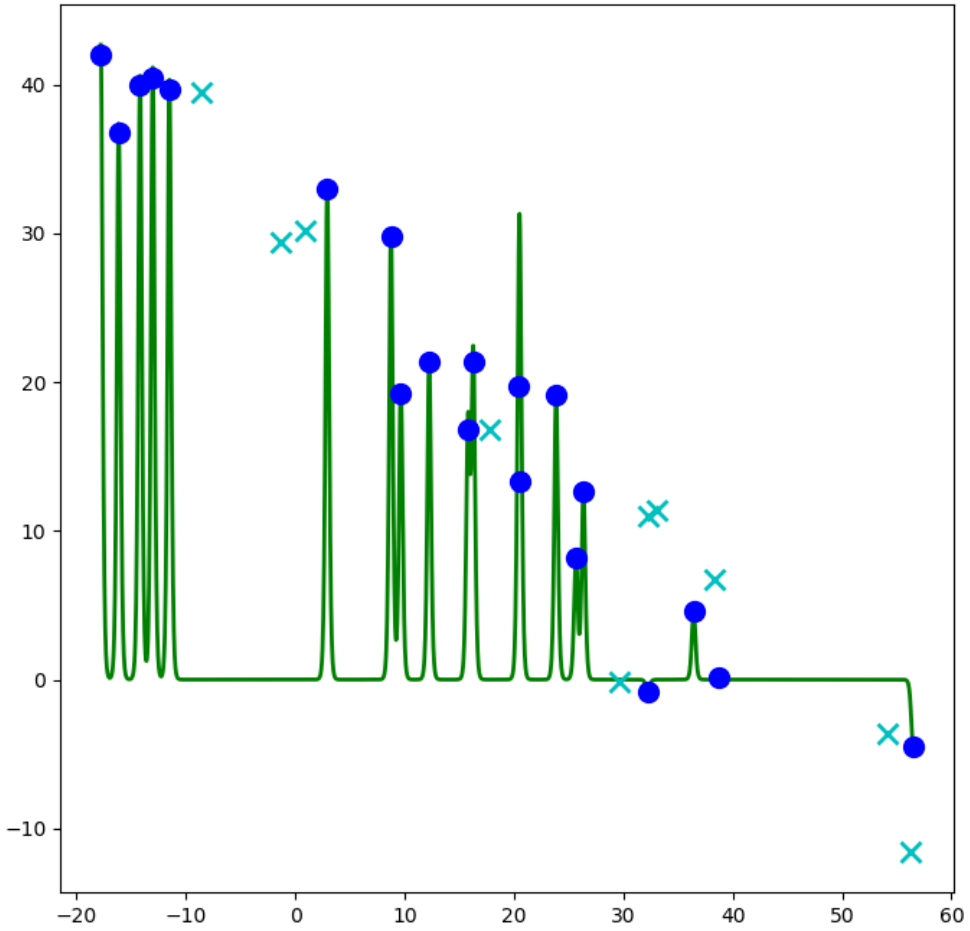
- *Stochastic* gradient decent take pseudorandom steps

  → Helps pop out of local minima

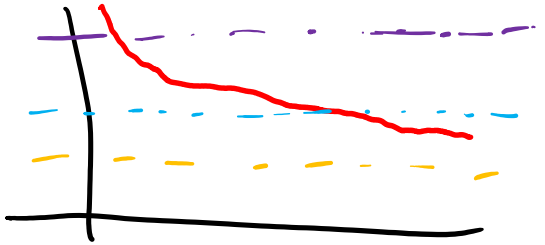
- Is getting stuck at a local minima necessarily a bad thing??

# Network to Approximate a 1-D Function



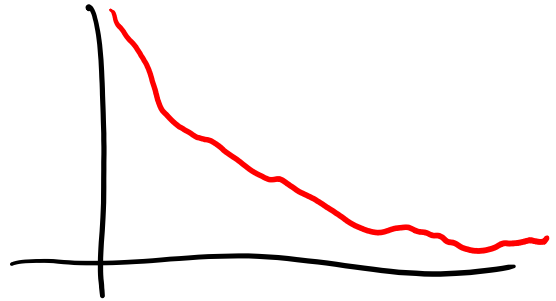



# Debugging Overfitting and Underfitting



## Underfitting (check this first!)

- Evidence: poor training loss (and poor validation loss)
  - Note: Compare with human performance as a baseline, i.e., make sure the task isn't impossible for a human (with unlimited resources)
- Try: Adding more capacity to network (wider or deeper)
  - But how do we choose a new network structure?



## Overfitting

- Evidence: good training loss, but poor validation loss
- Evidence: really large parameter values
- Try: Regularization (we'll learn about this soon!)
- Try: Adding more data

# Summary: Neural Networks Properties

## Practical considerations

- Large number of neurons
  - Danger for overfitting
- Modeling assumptions vs data assumptions trade-off
- Gradient descent can easily get stuck local optima

## What if there are no non-linear activations?

- A deep neural network with only linear layers can be reduced to an exactly equivalent single linear layer

## Universal Approximation Theorem:

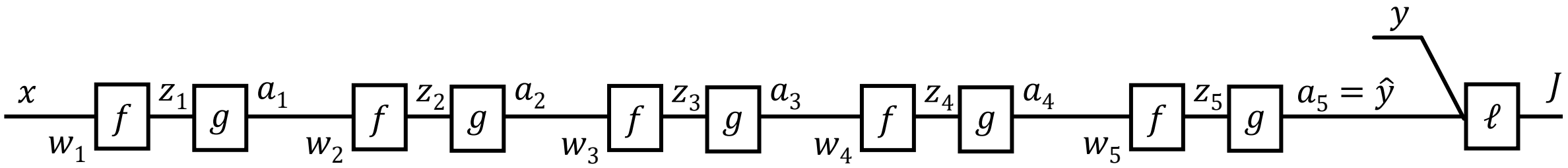
- A two-layer neural network with a sufficient number of neurons can approximate any continuous function to any desired accuracy.

# Forward Pass and Backpropagation

## Scalar's version

# Forward Pass

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




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

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

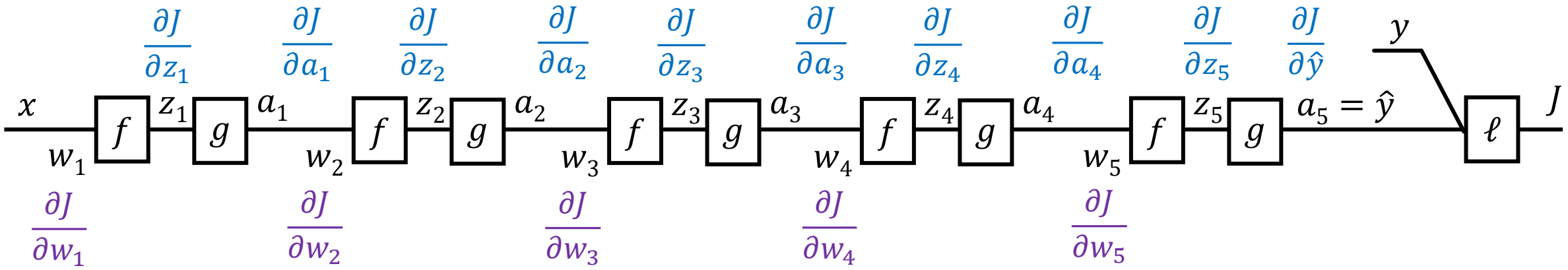
$$\hat{y} = h_{\boldsymbol{\theta}}(\mathbf{x}) = g\left(w_5 \cdot g\left(w_4 \cdot g\left(w_3 \cdot g\left(w_2 \cdot g(w_1 \cdot x)\right)\right)\right)\right)$$

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

# Backpropagation

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

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




Questions:

- Why do we need to compute all of the $\partial J/\partial z_\ell$ and $\partial J/\partial a_\ell$?
- Why "backwards"?
- How do we actually compute these?
- (What else do we need to learn to go beyond this useless network?)

# Reminder: Calculus Chain Rule (scalar version)

## Composite functions → chain rule

$$J = \ell\Big(g\big(f(x)\big)\Big)$$

$$\begin{aligned} J &= \ell(\hat{y}) \\ \hat{y} &= g(z) \\ z &= f(x, w) \end{aligned}$$

$$\frac{dJ}{dw} = \frac{d\ell}{d\hat{y}}\frac{dg}{dz}\frac{df}{dw}$$

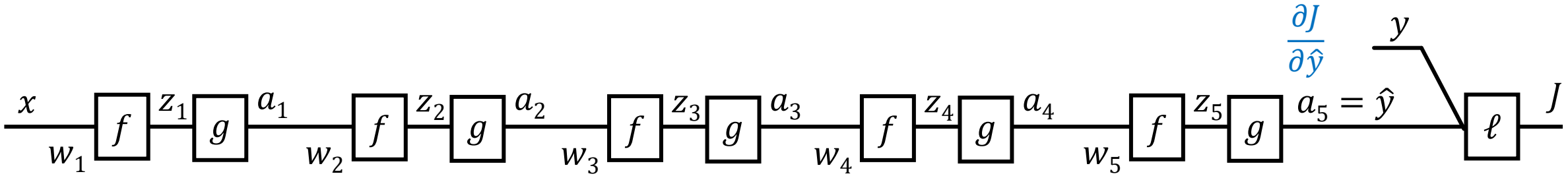
$$= \frac{\partial J}{\partial z}\frac{df}{dw}$$

$$f(x, w) = xw$$

# 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)

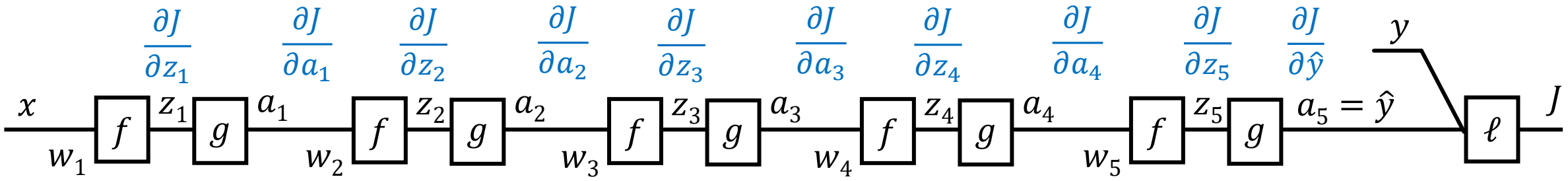



$$\frac{\partial J}{\partial w_1} =$$

# 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)



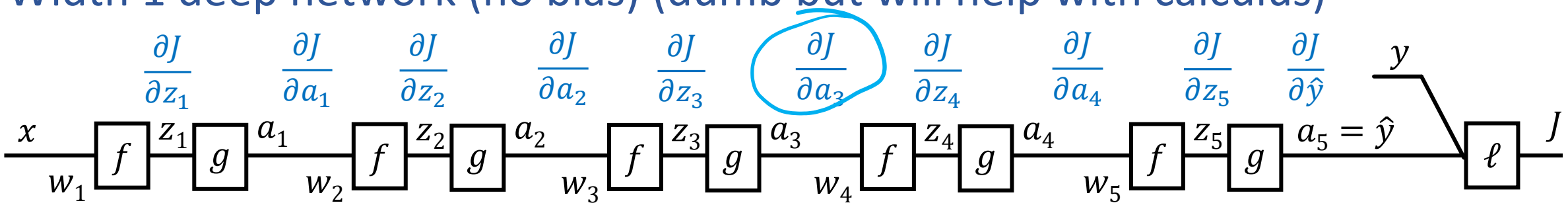

$$\frac{\partial J}{\partial w_1} = \frac{\partial J}{\partial \hat{y}} \frac{\partial g}{\partial z_5} \frac{\partial f}{\partial a_4} \frac{\partial g}{\partial z_4} \frac{\partial f}{\partial a_3} \frac{\partial g}{\partial z_3} \frac{\partial f}{\partial a_2} \frac{\partial g}{\partial z_2} \frac{\partial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

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

# 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)




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

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

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

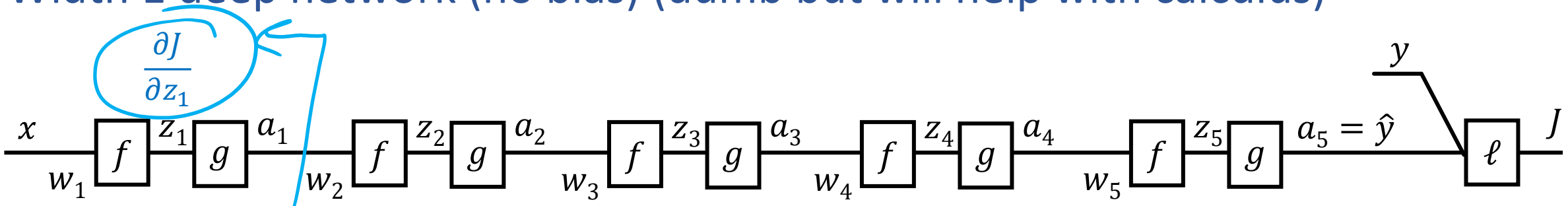
$$\frac{\partial J}{\partial w_4} = \frac{\partial J}{\partial \hat{y}}\frac{\partial g}{\partial z_5}\frac{\partial f}{\partial a_4}\frac{\partial g}{\partial z_4}\frac{\partial f}{\partial w_4}$$

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

# 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)




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

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

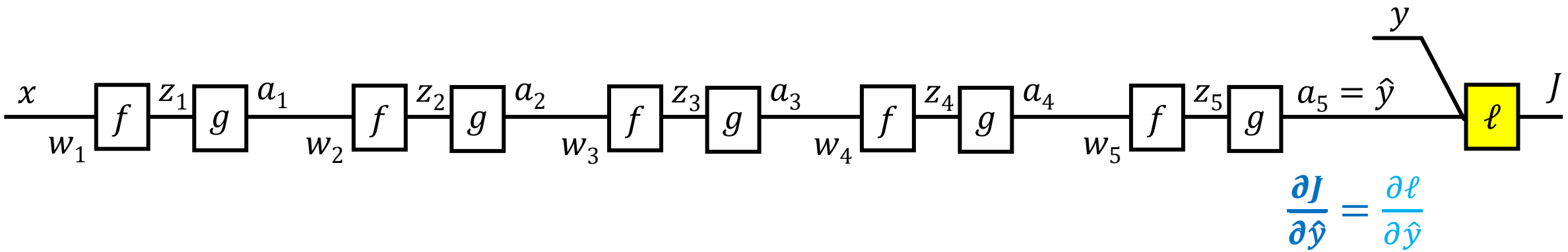
What if someone had already done all of the work to compute $\frac{\partial J}{\partial z_1}$ ?

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!

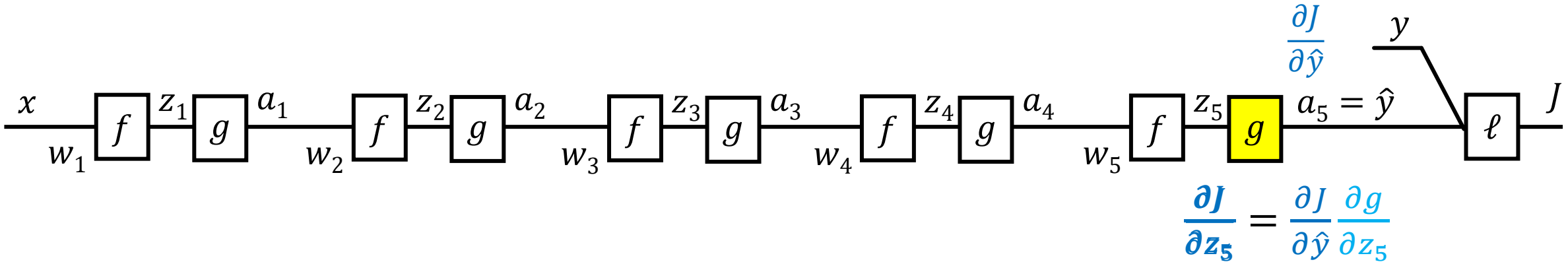



$$\frac{\partial J}{\partial \hat{y}} = \frac{\partial \ell}{\partial \hat{y}}$$

# Backpropagation

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

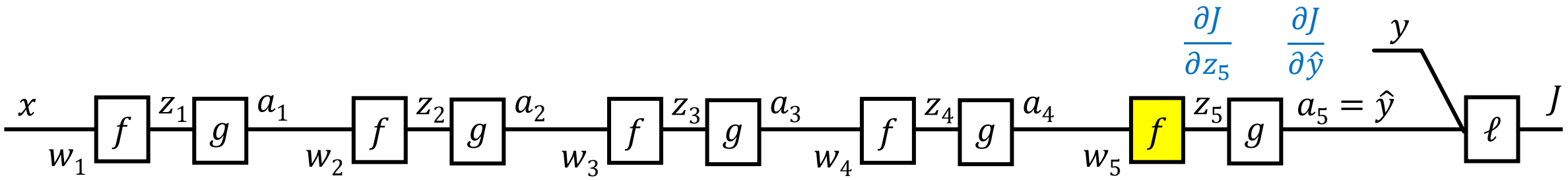
## More efficient to start at the end and reuse values!



# 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!



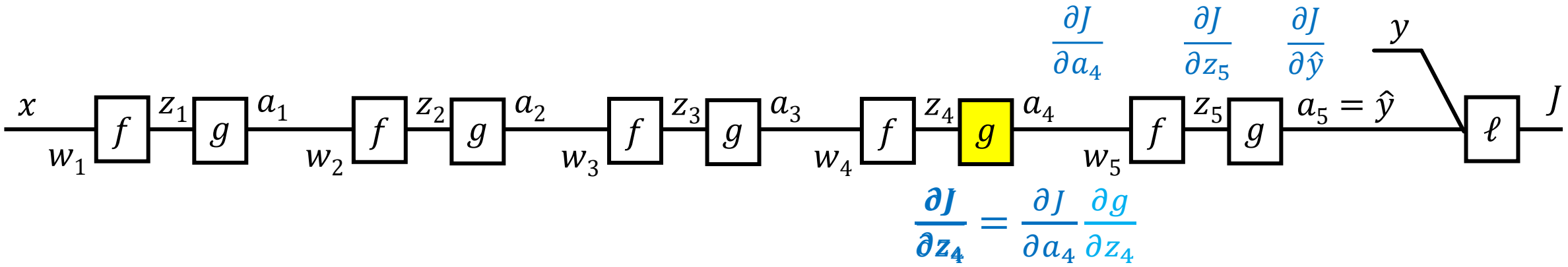

$$\frac{\partial J}{\partial a_4} = \frac{\partial J}{\partial z_5}\frac{\partial f}{\partial a_4}$$

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

# 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!

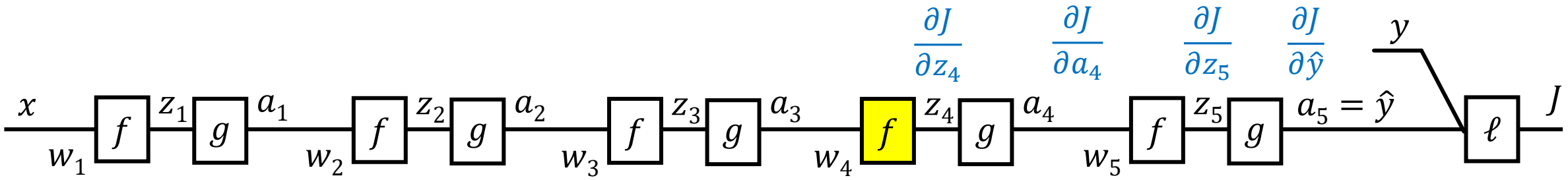



$$\frac{\partial J}{\partial z_4} = \frac{\partial J}{\partial a_4}\frac{\partial g}{\partial z_4}$$

# 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!



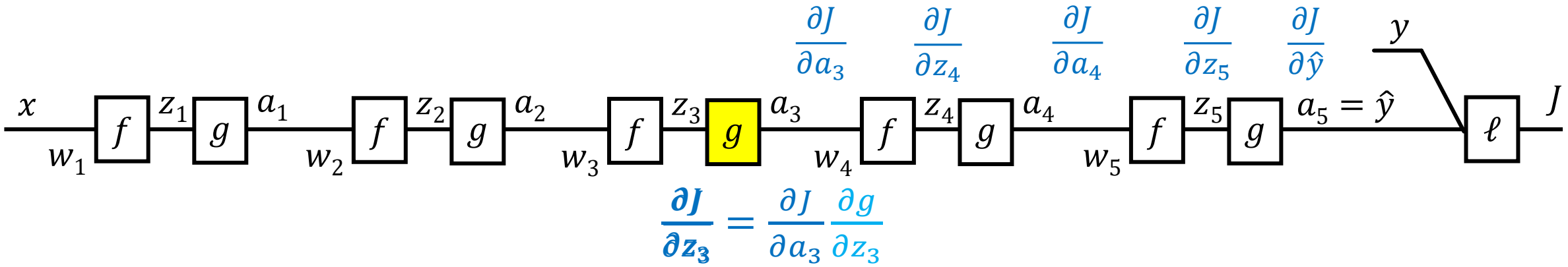

$$\frac{\partial J}{\partial a_3} = \frac{\partial J}{\partial z_4}\frac{\partial f}{\partial a_3}$$

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

# 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!

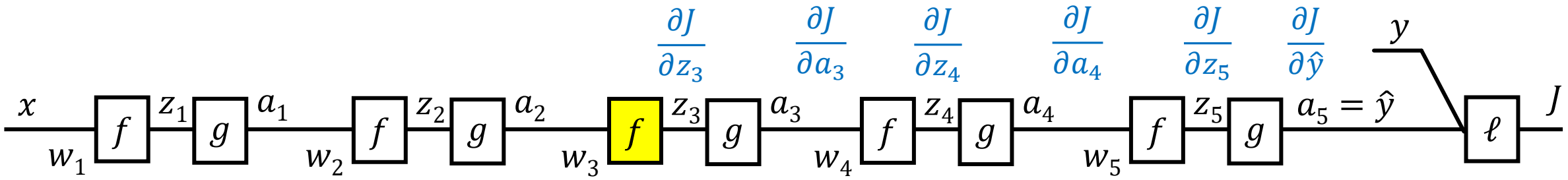



$$\frac{\partial J}{\partial z_3} = \frac{\partial J}{\partial a_3}\frac{\partial g}{\partial z_3}$$

# 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!



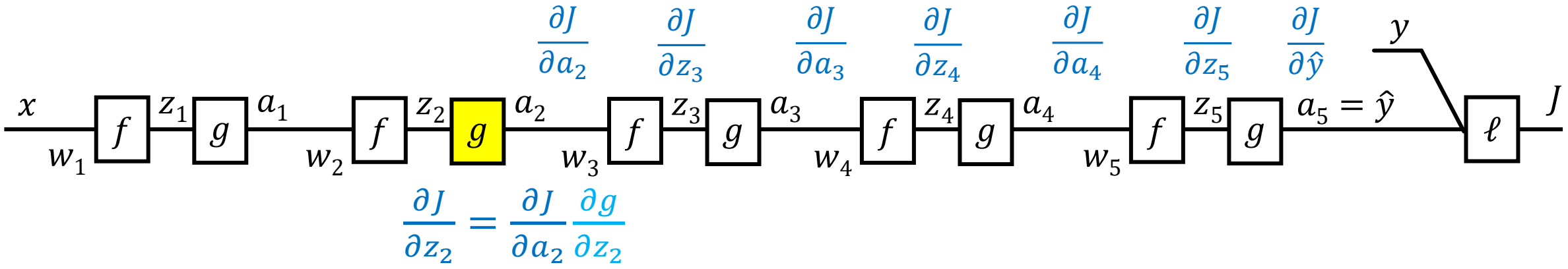

$$\frac{\partial J}{\partial a_2} = \frac{\partial J}{\partial z_3}\frac{\partial f}{\partial a_2}$$

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

# 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!

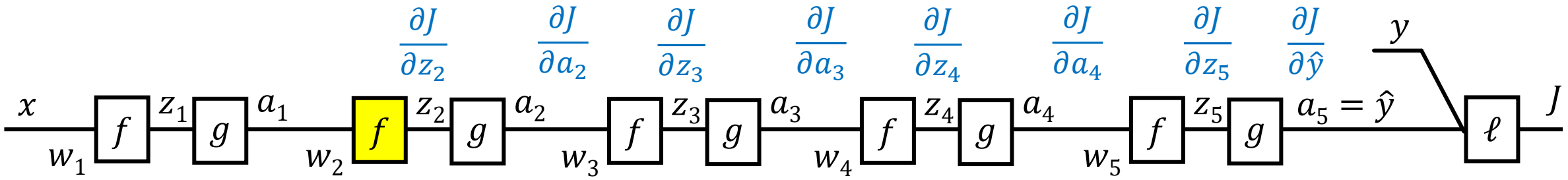



$$\frac{\partial J}{\partial z_2} = \frac{\partial J}{\partial a_2}\frac{\partial g}{\partial z_2}$$

# 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!



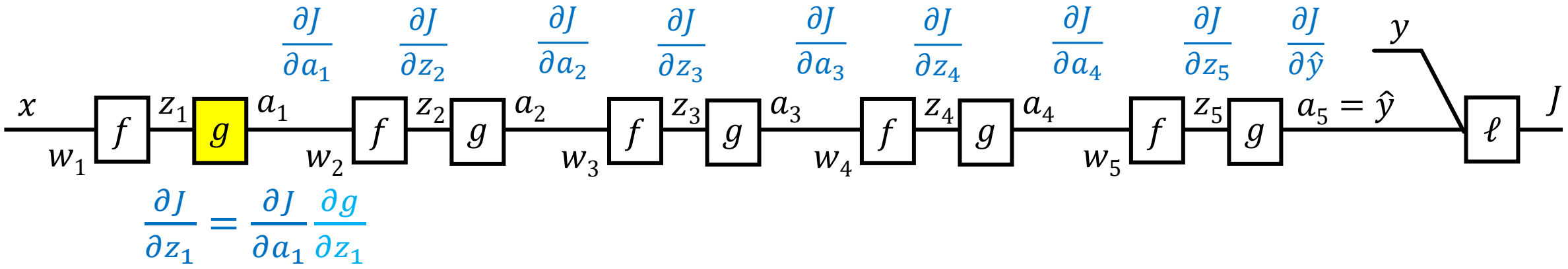

$$\frac{\partial J}{\partial a_1} = \frac{\partial J}{\partial z_2}\frac{\partial f}{\partial a_1}$$

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

# 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!

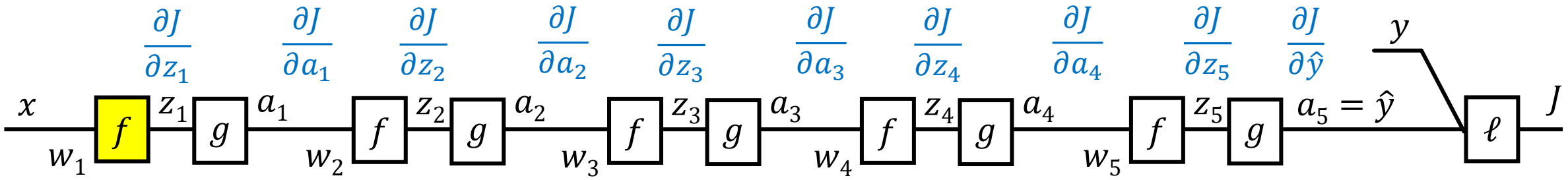



$$\frac{\partial J}{\partial z_1} = \frac{\partial J}{\partial a_1}\frac{\partial g}{\partial z_1}$$

# 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!

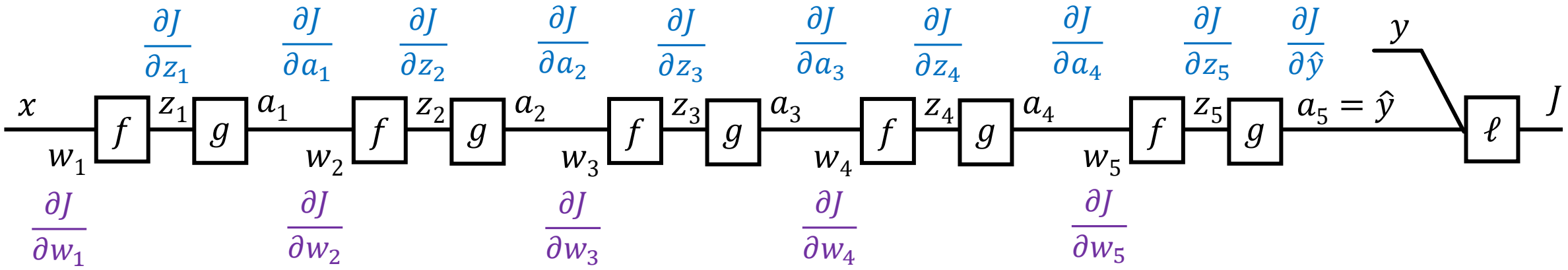



$$\frac{\partial J}{\partial w_1} = \frac{\partial J}{\partial z_1}\frac{\partial f}{\partial w_1}$$

# Backpropagation

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

## 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, $w_\ell \leftarrow w_\ell - \alpha \, \partial J / \partial w_\ell$

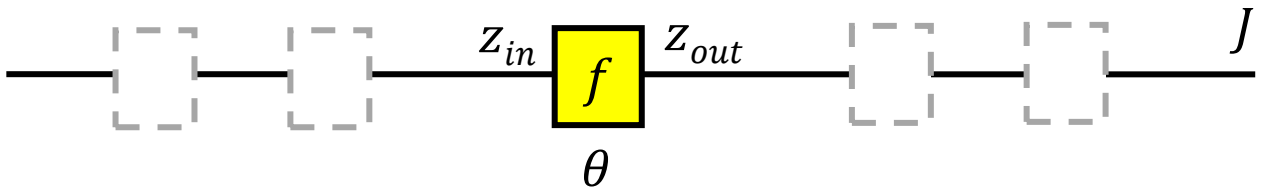
The backward pass propagates the change in the objective with respect to intermediate values ($\partial J / \partial z_\ell$ and $\partial J / \partial a_\ell$) back through the network to produce each $\partial J / \partial w_\ell$

# Generic Layer Implementation (so-far)

## Compute derivatives per layer, utilizing previous derivatives

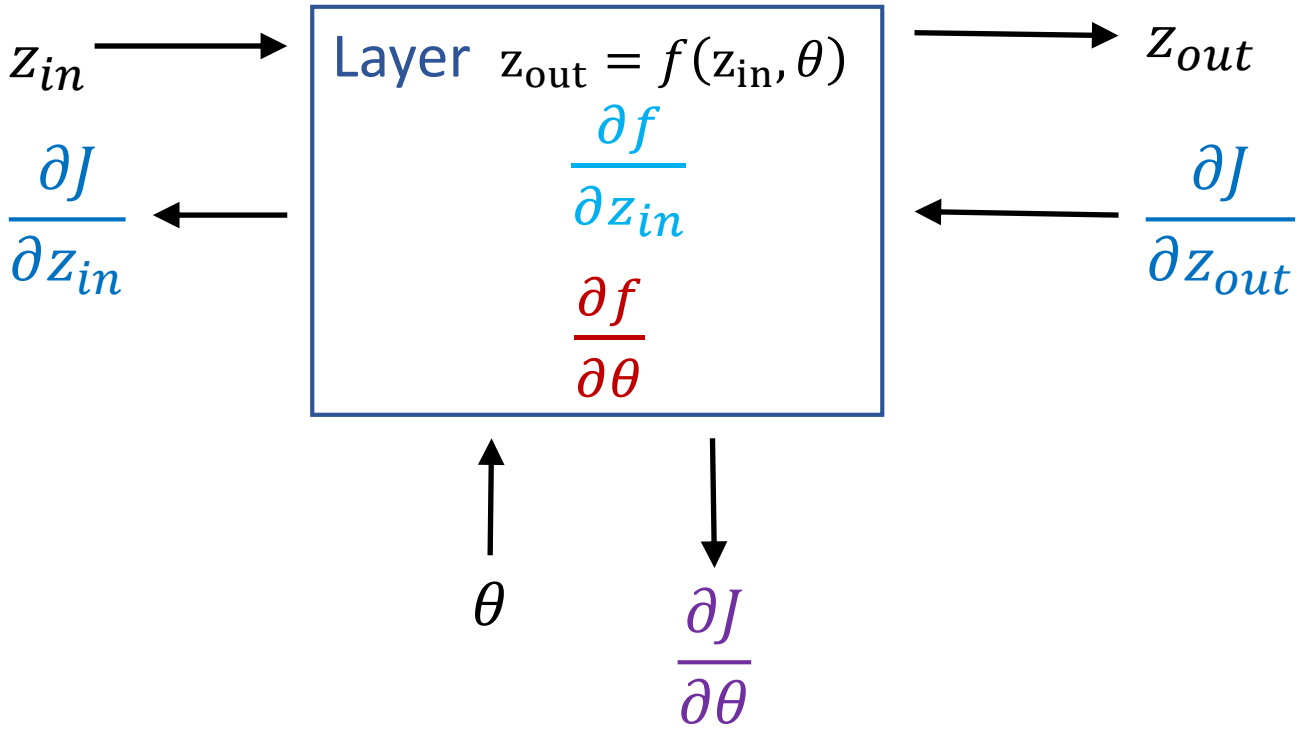
Objective: $J(\theta)$

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




Need:

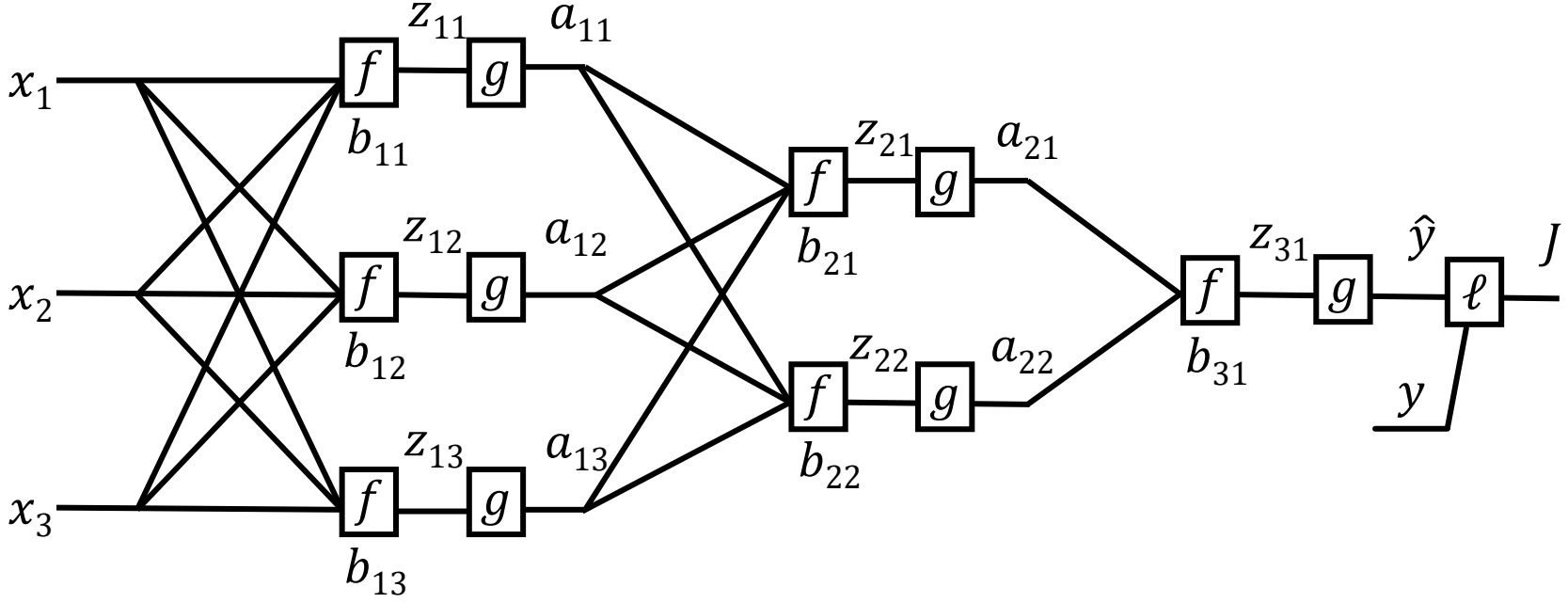
- $\frac{\partial J}{\partial z_{in}} = \frac{\partial J}{\partial z_{out}} \frac{\partial f}{\partial z_{in}}$
- $\frac{\partial J}{\partial \theta} = \frac{\partial J}{\partial z_{out}} \frac{\partial f}{\partial \theta}$



# 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




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

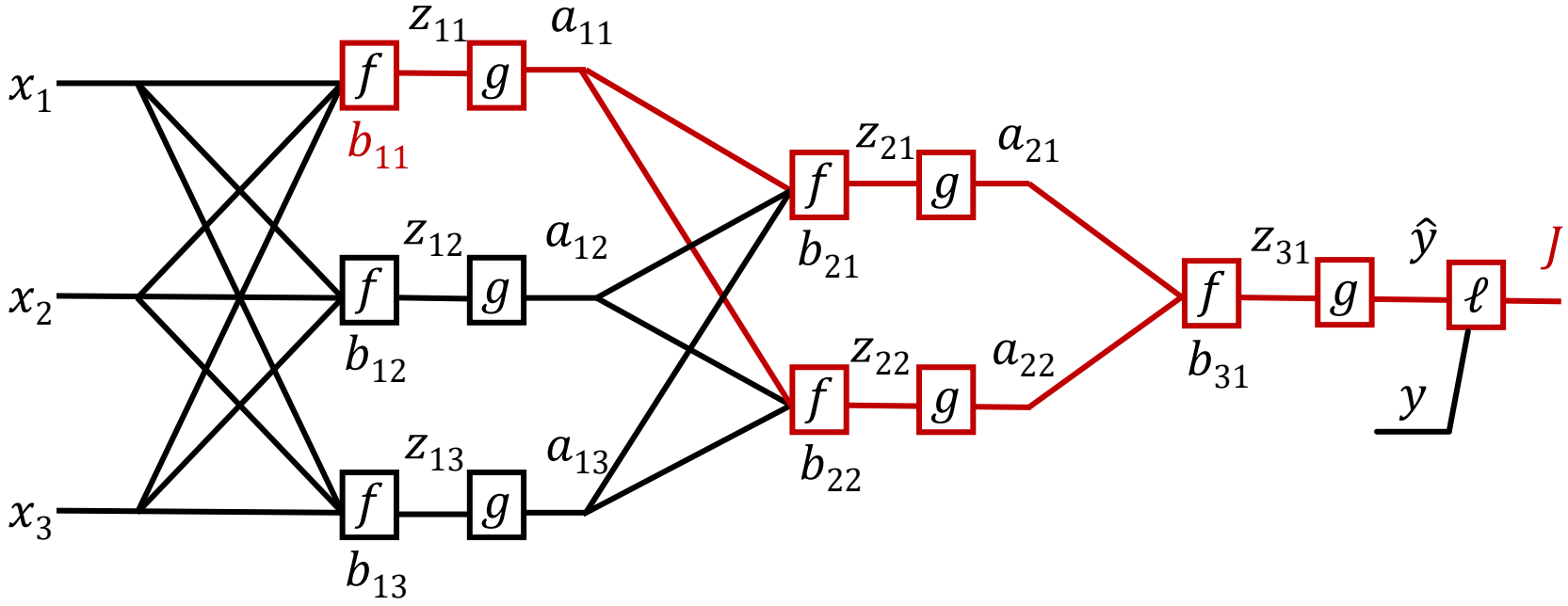
$$\frac{\partial J}{\partial b_{2,i}} = \frac{\partial \ell}{\partial \hat{y}} \frac{\partial a_{3,1}}{\partial z_{3,1}} \frac{\partial z_{3,1}}{\partial a_{2,i}} \frac{\partial a_{2,i}}{\partial z_{2,i}} \frac{\partial z_{2,i}}{\partial b_{2,i}}$$

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

# 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




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

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

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

# Calculus Time-out

# Multivariate-chain rule

# Multivariable Chain Rule

$g_1(x) = 3x$

$g_2(x) = 5x$

$f(z_1, z_2) = 2z_1 + 7z_2$

$y = f\big(g_1(x), g_2(x)\big)$

# Exercise: Multivariable Chain Rule

$$\frac{df}{dx} = \frac{\partial f}{\partial g_1}\frac{\partial g_1}{\partial x} + \frac{\partial f}{\partial g_2}\frac{\partial g_2}{\partial x}$$

$$z_1 = g_1(x) = \sin(x)$$

$$z_2 = g_2(x) = x^3$$

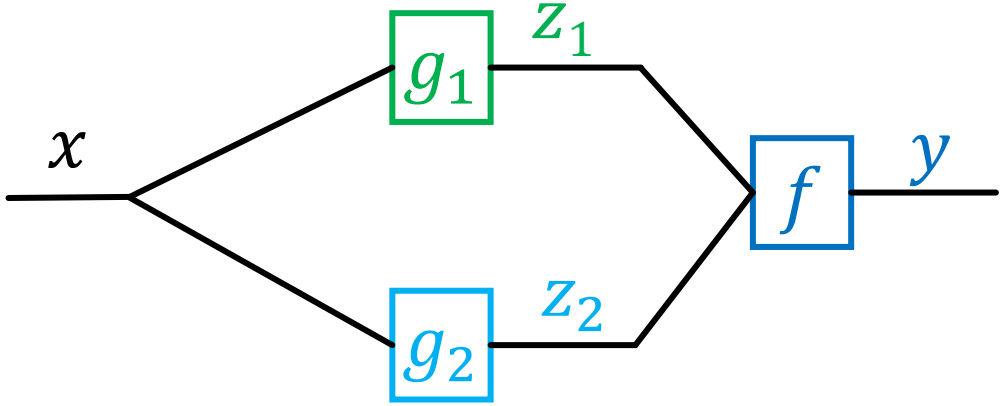
$$y = f(z_1, z_2) = z_1^4 e^{z_2} + 5z_1 + 7z_2$$

# Exercise: Multivariable Chain Rule

$$\frac{df}{dx} = \frac{\partial f}{\partial g_1}\frac{\partial g_1}{\partial x} + \frac{\partial f}{\partial g_2}\frac{\partial g_2}{\partial x}$$

$$z_1 = g_1(x) = \sin(x)$$
$$z_2 = g_2(x) = x^3$$
$$y = f(z_1, z_2) = z_1^4 e^{z_2} + 5z_1 + 7z_2$$

$$dg_1/dx = \cos(x)$$
$$dg_2/dx = 3x^2$$
$$\partial f/\partial z_1 = 4z_1^3 e^{z_2} + 5$$
$$\partial f/\partial z_2 = z_1^4 e^{z_2} + 7$$




$$\frac{df}{dx} = \frac{\partial f}{\partial g_1}\frac{\partial g_1}{\partial x} + \frac{\partial f}{\partial g_2}\frac{\partial g_2}{\partial x}$$

$$= \left(4z_1^3 e^{z_2} + 5\right)\cos(x) + \left(z_1^4 e^{z_2} + 7\right)3x^2$$

# Calculus Chain Rule

## Scalar:

$$y = f(z)$$

$$z = g(x)$$

$$\frac{dy}{dx} = \frac{dy}{dz}\frac{dz}{dx}$$

## Multivariate:

$$y = f(\boldsymbol{z})$$

$$\boldsymbol{z} = g(x)$$

$$\frac{dy}{dx} = \sum_j \frac{\partial y}{\partial z_j}\frac{\partial z_j}{\partial x}$$

## Multivariate:

$$\mathbf{y} = f(\boldsymbol{z})$$

$$\boldsymbol{z} = g(\mathbf{x})$$

$$\frac{dy_\mathrm{i}}{dx_\mathrm{k}} = \sum_j \frac{\partial y_i}{\partial z_j}\frac{\partial z_j}{\partial x_\mathrm{k}}$$

# Calculus Chain Rule

## Scalar:

$$y = f(z)$$

$$z = g(x)$$

$$\frac{dy}{dx} = \frac{dy}{dz}\frac{dz}{dx}$$

## Multivariate:

$$y = f(\boldsymbol{z})$$

$$\boldsymbol{z} = g(x)$$

$$\frac{dy}{dx} = \sum_j \frac{\partial y}{\partial z_j}\frac{\partial z_j}{\partial x}$$

## Multivariate:

$$\mathbf{y} = f(\boldsymbol{z})$$

$$\boldsymbol{z} = g(\mathbf{x})$$

$$\frac{dy_\mathrm{i}}{dx_\mathrm{k}} = \sum_j \frac{\partial y_i}{\partial z_j}\frac{\partial z_j}{\partial x_\mathrm{k}}$$

# Multivariable Chain Rule

| | Numerator layout | Denominator layout | Notes |
|---|---|---|---|
| $\frac{d}{dt}\ f\left(g(t), h(t)\right)$ | $\frac{df}{dg}\frac{dg}{dt} + \frac{df}{dh}\frac{dh}{dt}$ | Same | $f : (\mathbb{R} \times \mathbb{R}) \to \mathbb{R},\ t \in \mathbb{R}$<br>$g : \mathbb{R} \to \mathbb{R},\ h : \mathbb{R} \to \mathbb{R}$ |
| $\frac{d}{dt}\ f\left(g_1(t), \ldots, g_N(t)\right)$ | $\sum_{i=1}^{N} \frac{df}{dg_i}\frac{dg_i}{dt}$ | Same | $h : (\mathbb{R} \times \cdots \times \mathbb{R}) \to \mathbb{R},\ t \in \mathbb{R}$<br>$f_i : \mathbb{R} \to \mathbb{R}\ \ \forall i \in \{1, \ldots, N\}$ |
| $\frac{d}{dt}\ f(\mathbf{g}(t))$ | $\frac{\partial f}{\partial \mathbf{g}}\frac{\partial \mathbf{g}}{\partial t}$ | $\frac{\partial \mathbf{g}}{\partial t}\frac{\partial f}{\partial \mathbf{g}}$ | $f : \mathbb{R}^N \to \mathbb{R},\ \mathbf{g} : \mathbb{R} \to \mathbb{R}^N$<br>$t \in \mathbb{R}$ |
| $\frac{\partial}{\partial \mathbf{v}}\ f(\mathbf{g}(\mathbf{v}))$ | $\frac{\partial f}{\partial \mathbf{g}}\frac{\partial \mathbf{g}}{\partial \mathbf{v}}$ | $\frac{\partial \mathbf{g}}{\partial \mathbf{v}}\frac{\partial f}{\partial \mathbf{g}}$ | $f : \mathbb{R}^N \to \mathbb{R},\ \mathbf{g} : \mathbb{R}^M \to \mathbb{R}^N$<br>$\mathbf{v} \in \mathbb{R}^M$ |
| $\frac{\partial}{\partial \mathbf{v}}\ f(\mathbf{g}(\mathbf{v}), \mathbf{h}(\mathbf{v}))$ | $\frac{\partial f}{\partial \mathbf{g}}\frac{\partial \mathbf{g}}{\partial \mathbf{v}} + \frac{\partial f}{\partial \mathbf{h}}\frac{\partial \mathbf{h}}{\partial \mathbf{v}}$ | $\frac{\partial \mathbf{g}}{\partial \mathbf{v}}\frac{\partial f}{\partial \mathbf{g}} + \frac{\partial \mathbf{h}}{\partial \mathbf{v}}\frac{\partial f}{\partial \mathbf{h}}$ | $h : \mathbb{R}^K \times \mathbb{R}^N \to \mathbb{R},\ \mathbf{v} \in \mathbb{R}^M$<br>$f : \mathbb{R}^M \to \mathbb{R}^K,\ \mathbf{g} : \mathbb{R}^M \to \mathbb{R}^N$ |

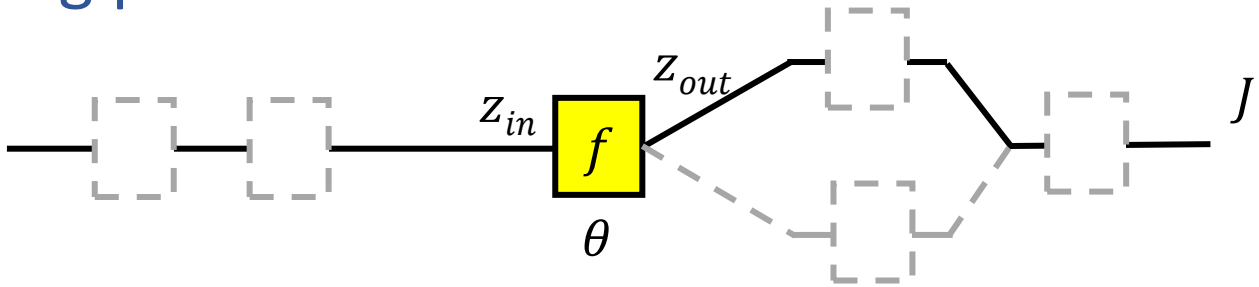
# Backpropagation (vector version)

# Generic Layer Implementation (updated)

## Compute derivatives per layer, utilizing previous derivatives

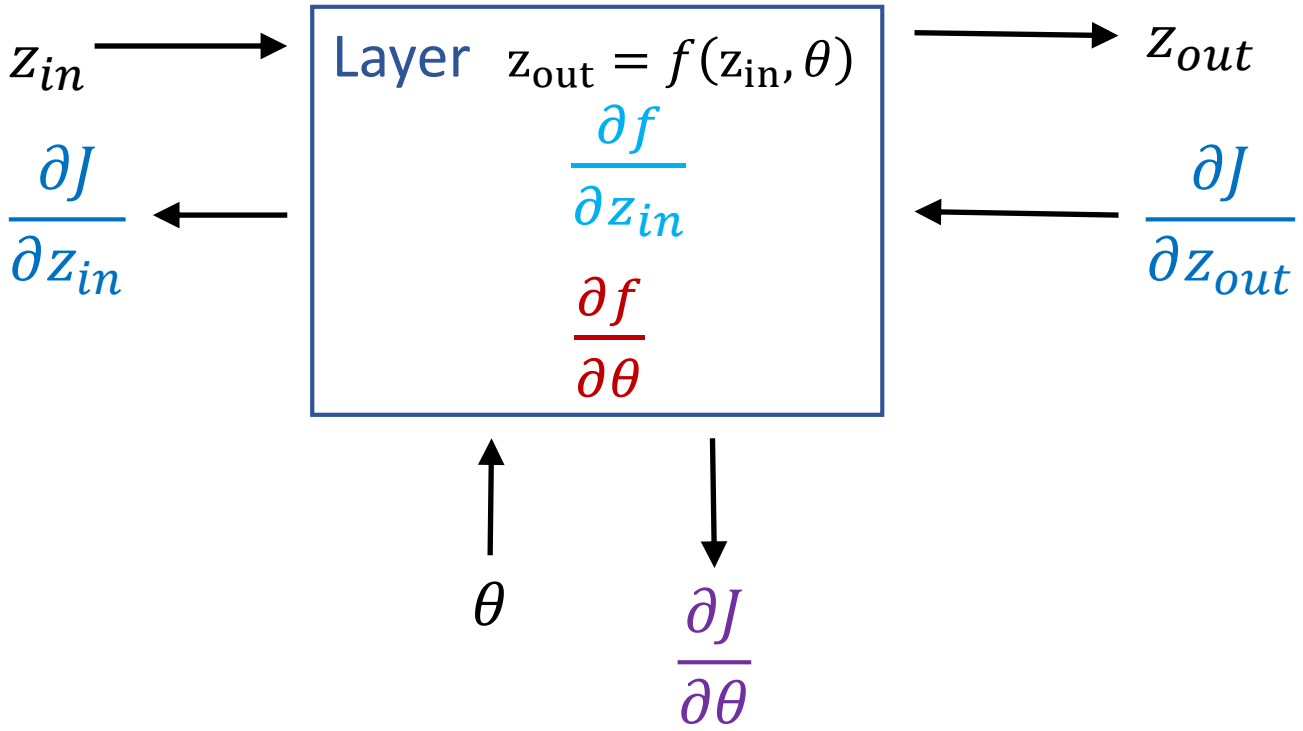
Objective: $J(\theta)$

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




## Need:

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

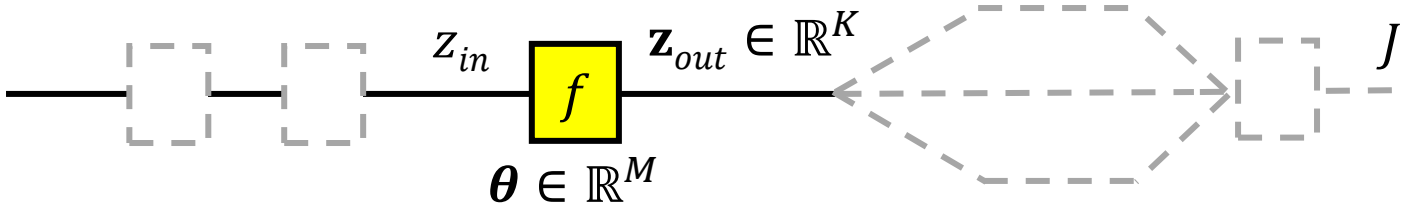


# Generic Layer Implementation (vector output version)

## Compute derivatives per layer, utilizing previous derivatives

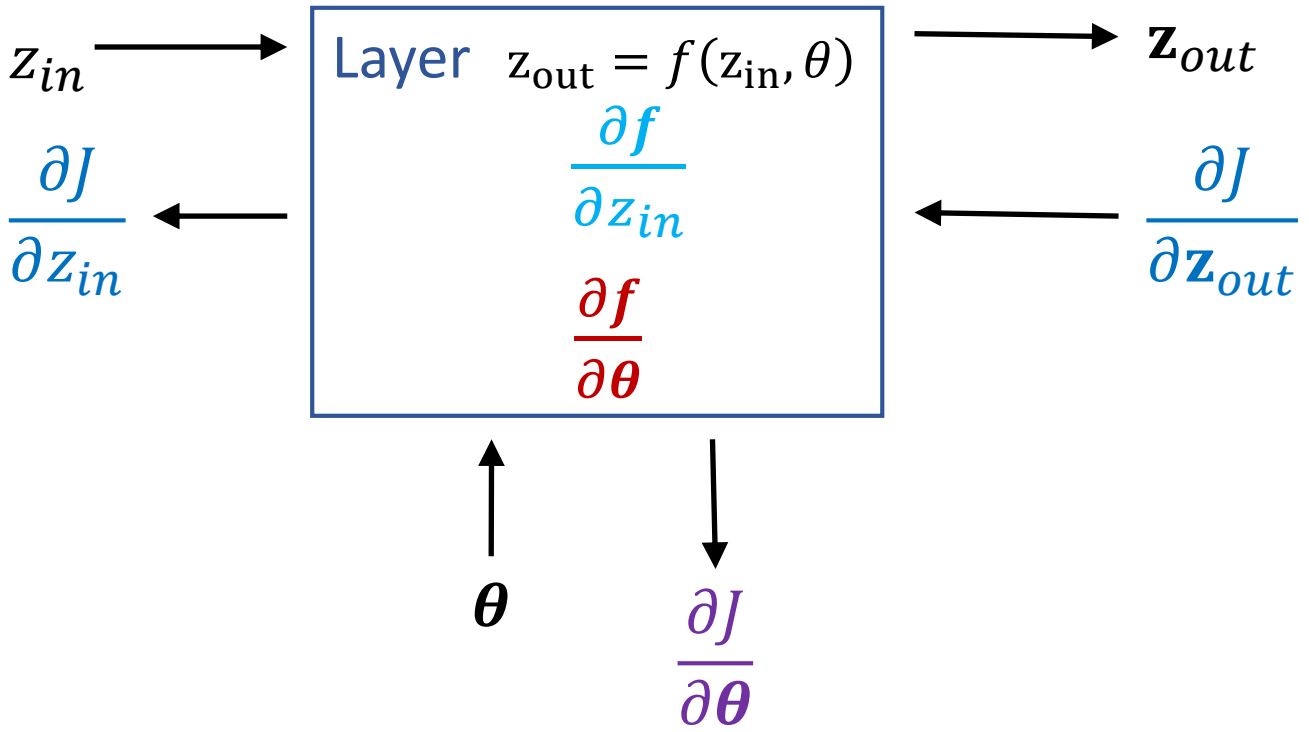
Objective: $J(\theta)$

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




Need:

- $\frac{\partial J}{\partial z_{in}} =$
- $\frac{\partial J}{\partial \theta_j} =$

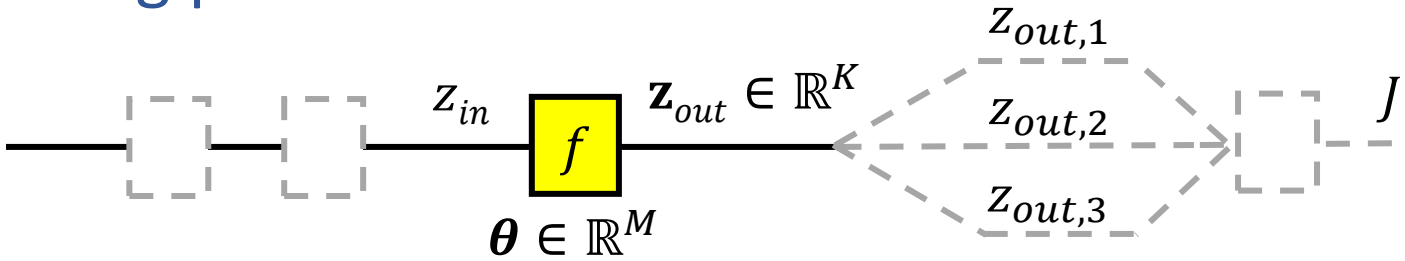


# Generic Layer Implementation (vector output version)

## Compute derivatives per layer, utilizing previous derivatives

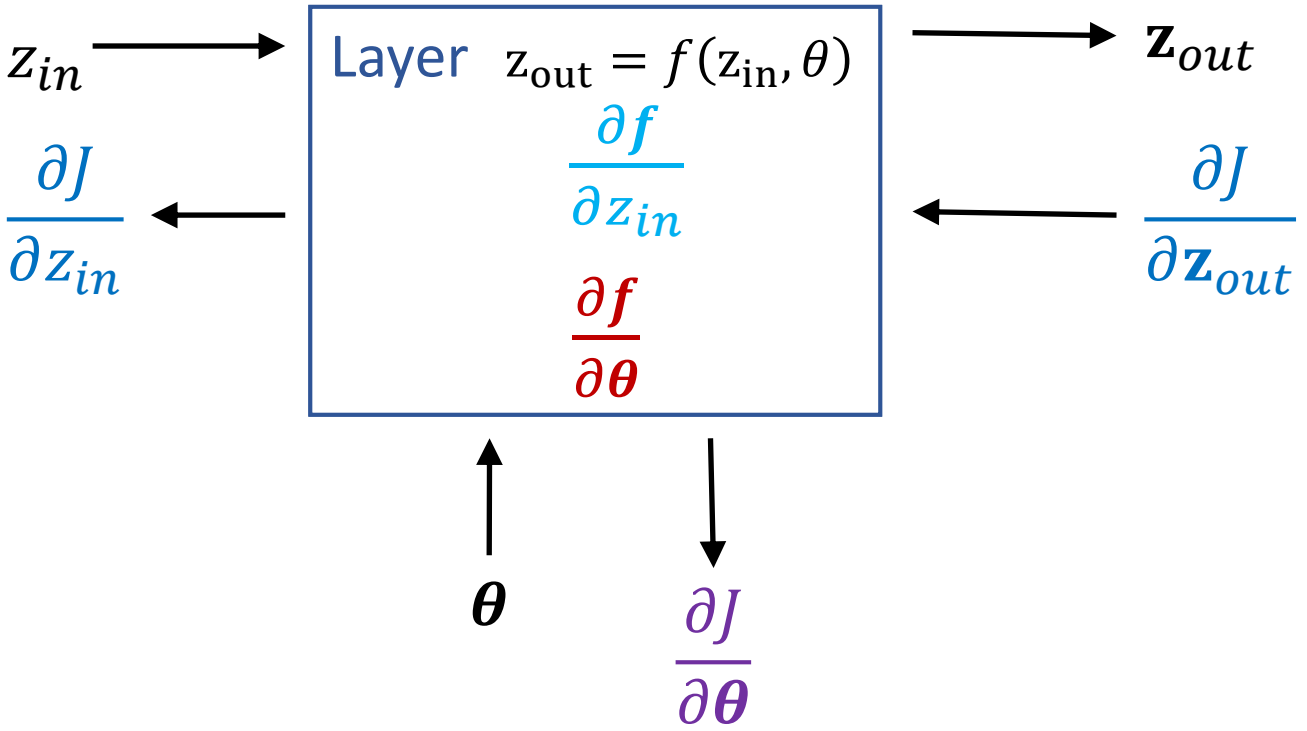
Objective: $J(\theta)$

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

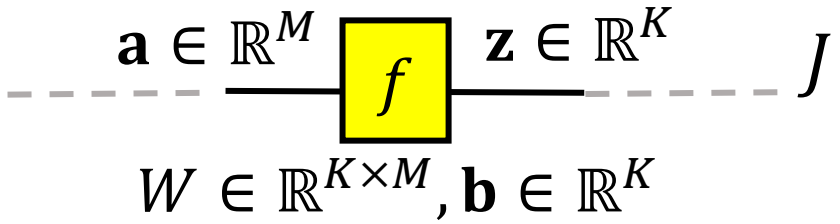



Need:

- $\frac{\partial J}{\partial z_{in}} = \sum_k \frac{\partial J}{\partial z_{out,k}} \frac{\partial z_{out,k}}{\partial z_{in}}$
- $\frac{\partial J}{\partial \theta_j} = \sum_k \frac{\partial J}{\partial z_{out,k}} \frac{\partial f}{\partial \theta_j}$



# Linear Layer Implementation



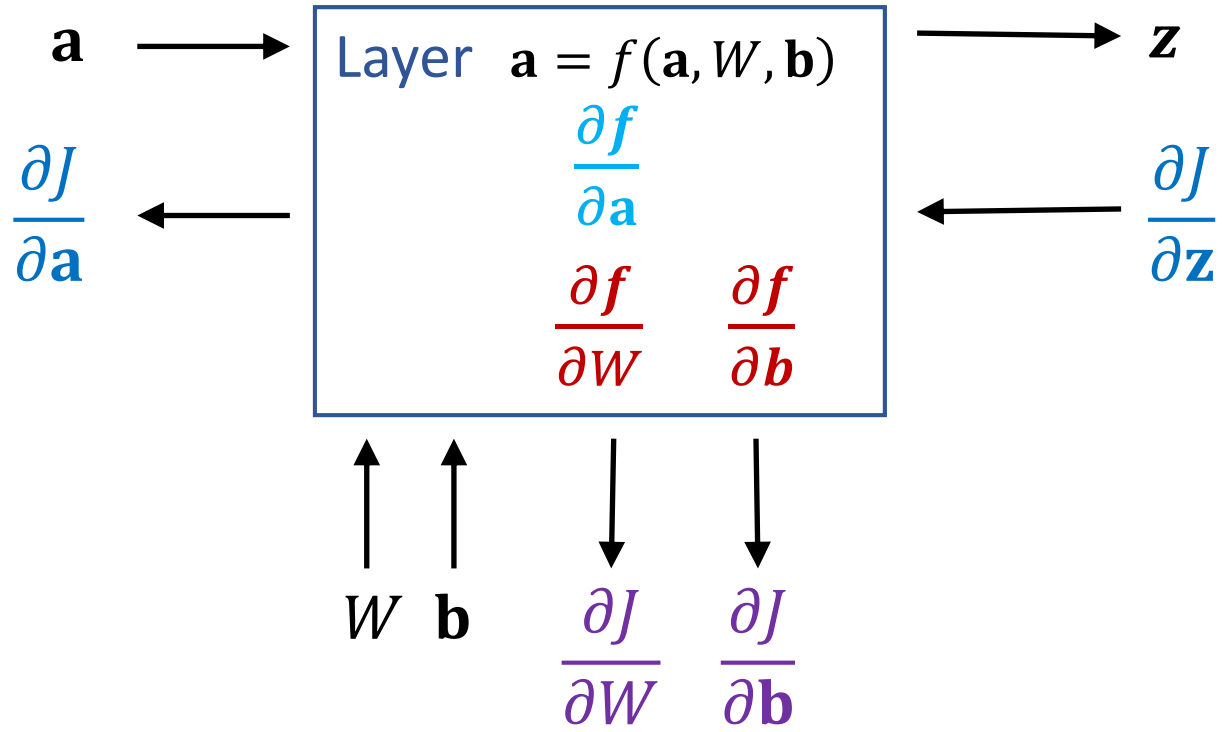

## Compute derivatives per layer

Objective: $J(\theta)$

Layer: $\mathbf{z} = f(\mathbf{a}, W, \mathbf{b}) = W\mathbf{a} + \mathbf{b}$

## Scalar version (no formatting)

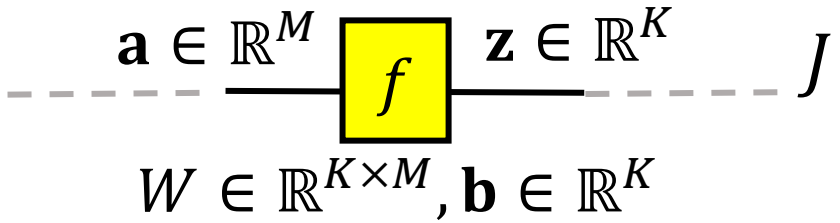
- $\frac{\partial J}{\partial a_j} =$
- $\frac{\partial J}{\partial W_{i,j}} =$
- $\frac{\partial J}{\partial b_i} =$




$$\mathbf{z} = W\mathbf{a} + \mathbf{b}$$

$$z_k = b_k + \sum_j W_{k,j} a_j$$

# Linear Layer Implementation



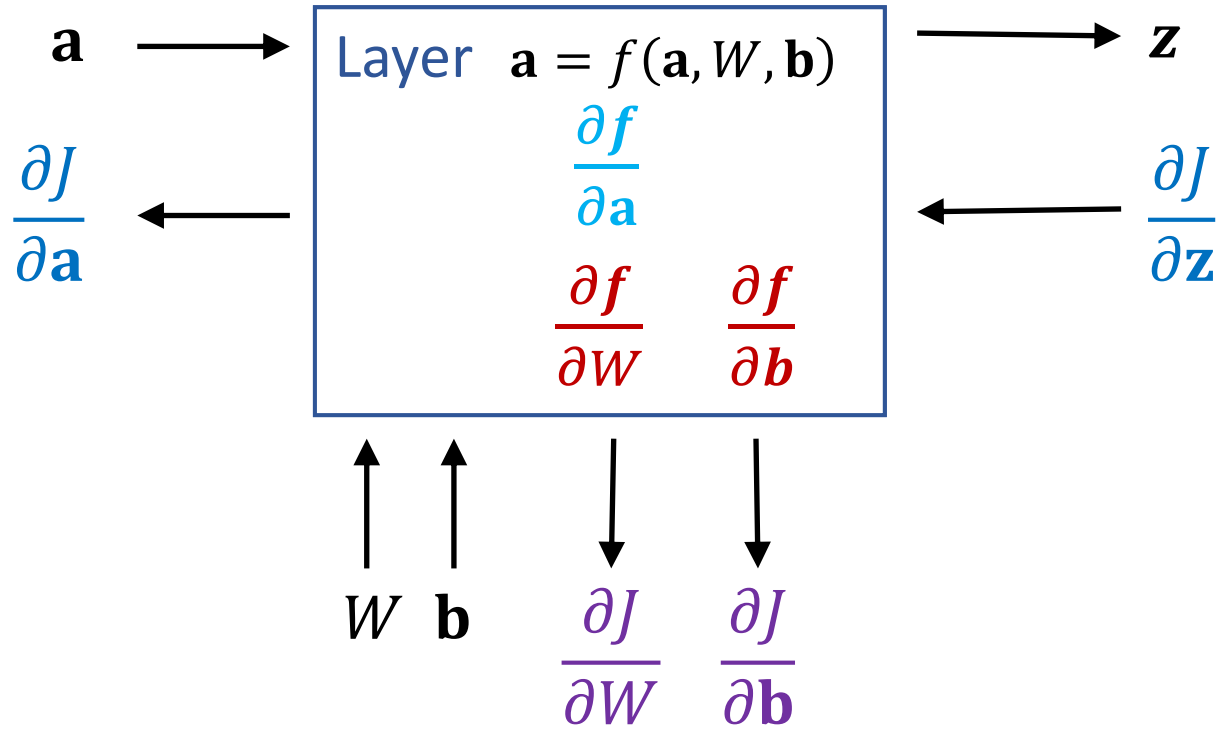

## Compute derivatives per layer

Objective: $J(\theta)$

Layer: $\mathbf{z} = f(\mathbf{a}, W, \mathbf{b}) = W\mathbf{a} + \mathbf{b}$

## Scalar version (no formatting)

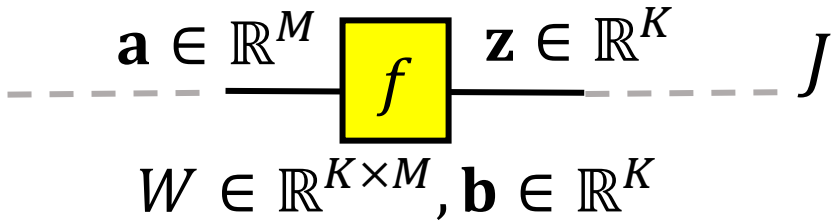
- $\frac{\partial J}{\partial a_j} = \sum_k \frac{\partial J}{\partial z_k} \frac{\partial z_k}{\partial a_j}$
- $\frac{\partial J}{\partial W_{i,j}} = \sum_k \frac{\partial J}{\partial z_k} \frac{\partial z_k}{\partial W_{i,j}}$
- $\frac{\partial J}{\partial b_i} = \sum_k \frac{\partial J}{\partial z_k} \frac{\partial z_k}{\partial b_i}$




$$\mathbf{z} = W\mathbf{a} + \mathbf{b}$$

$$z_k = b_k + \sum_j W_{k,j} a_j$$

# Linear Layer Implementation

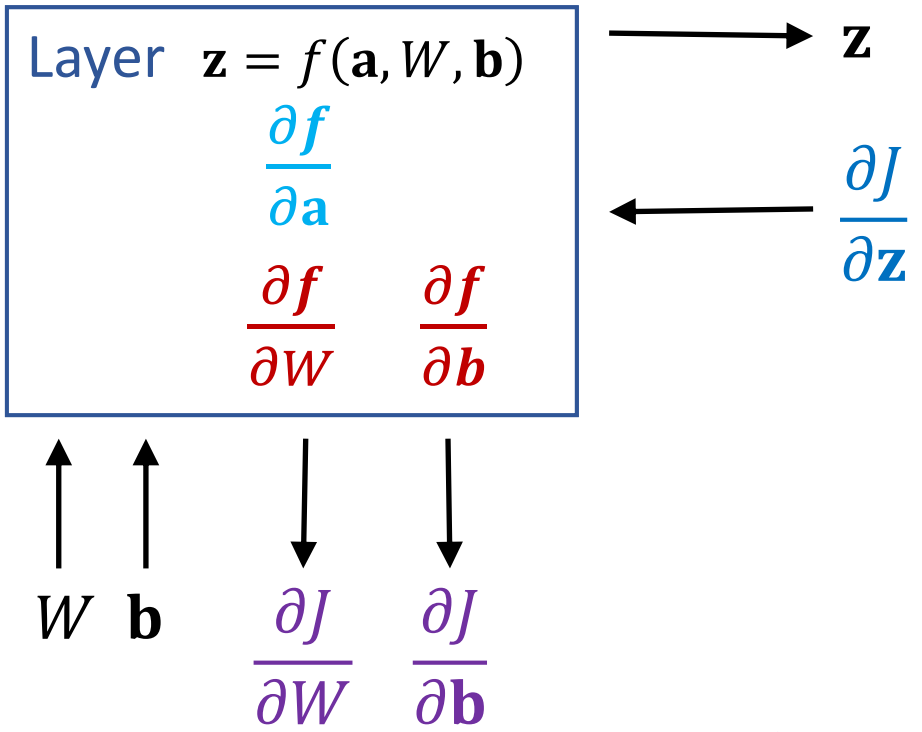



## Compute derivatives per layer

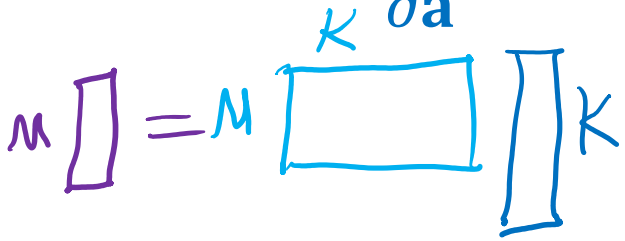
Objective: $J(\theta)$

Layer: $\mathbf{z} = f(\mathbf{a}, W, \mathbf{b}) = W\mathbf{a} + \mathbf{b}$

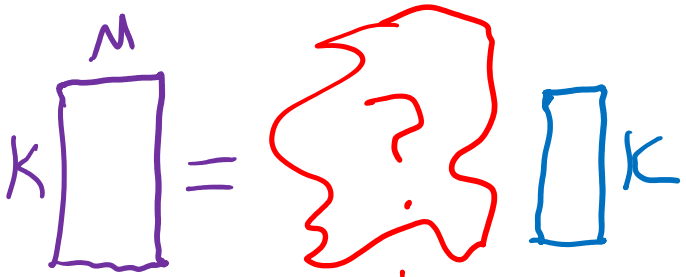
(Denominator format)



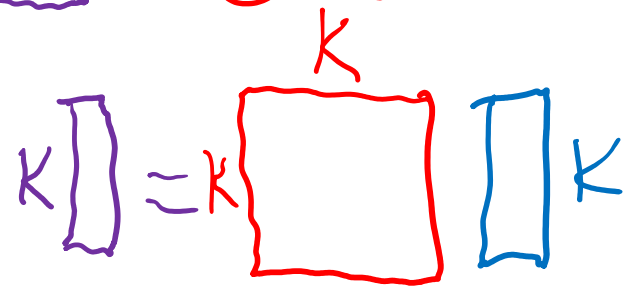

- $\frac{\partial J}{\partial \mathbf{a}} = \frac{\partial \vec{z}}{\partial \vec{a}} \frac{\partial J}{\partial \vec{z}}$




- $\frac{\partial J}{\partial W} = \frac{\partial \vec{z}}{\partial W} \frac{\partial J}{\partial \vec{z}}$




- $\frac{\partial J}{\partial \mathbf{b}} = \frac{\partial \vec{z}}{\partial \vec{b}} \frac{\partial J}{\partial \vec{z}}$




$$W \leftarrow W - \alpha \frac{\partial J}{\partial W}$$