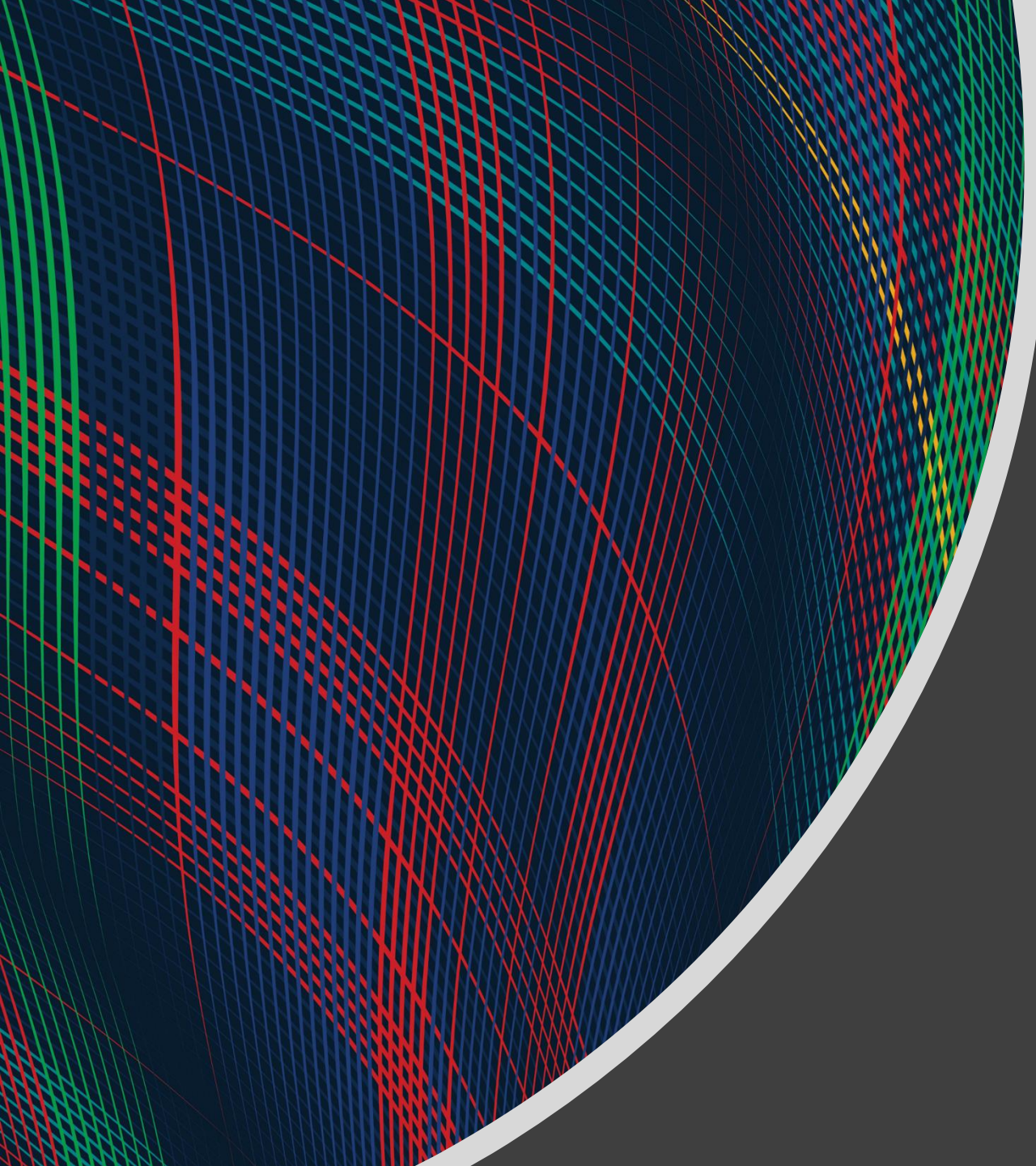


An abstract graphic on the left side of the slide, featuring a sphere-like shape composed of a dense grid of intersecting red, green, and blue lines. The lines are curved and follow the contour of the sphere, creating a complex, woven pattern. The sphere is set against a dark gray background.

07-280
AI & ML I

Neural Networks II

Instructor:
Aviral Kumar & 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
- 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?

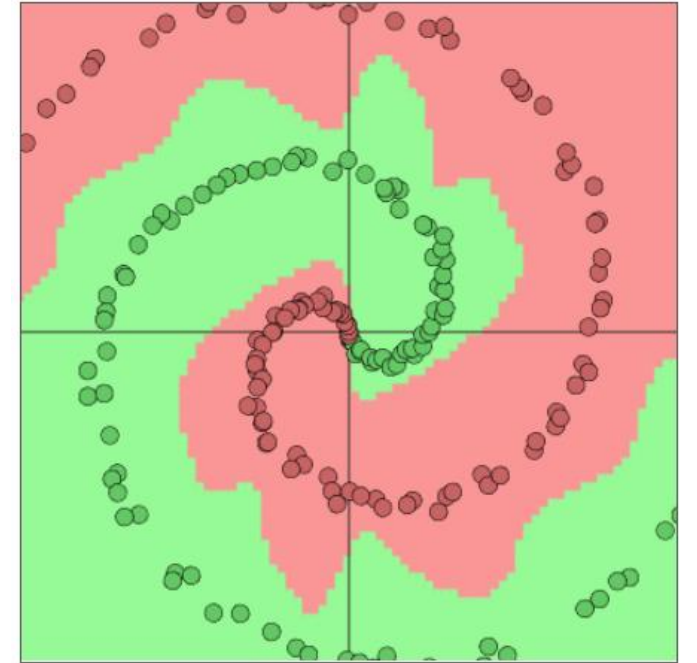
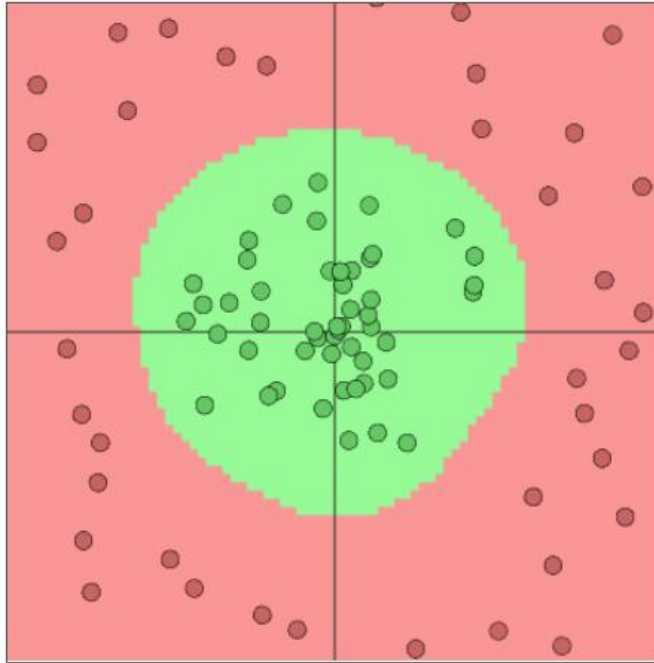
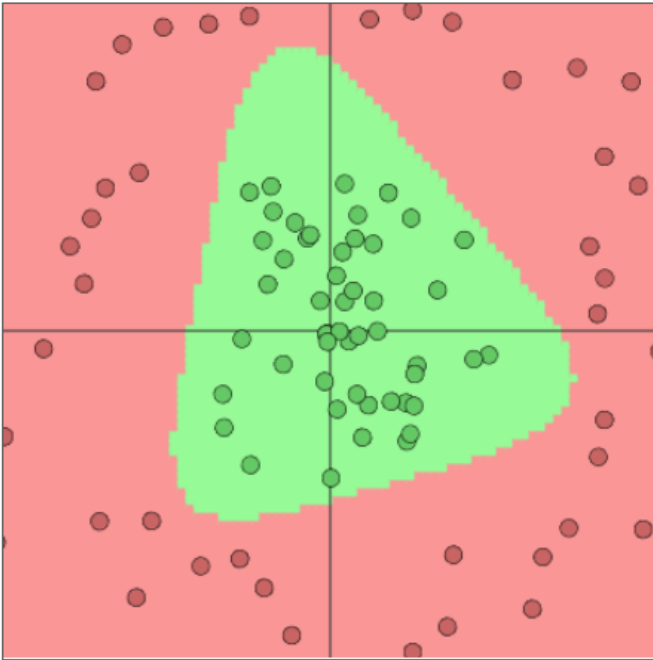
What if there are no non-linear activations?

How many neurons/layers should we add?

Please, tell me more about calculus

Network to Approximate Binary Classification

Approximate arbitrary decision boundary



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??

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:

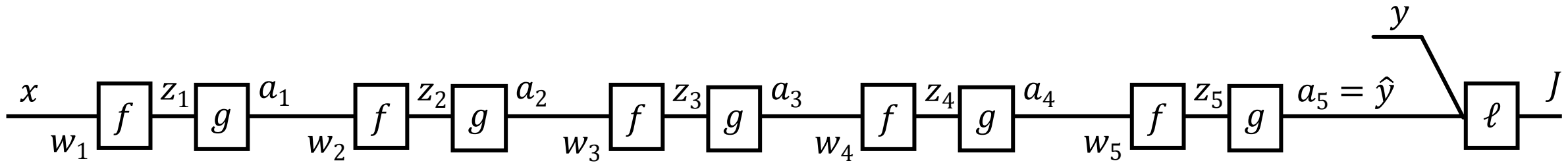
- A two-layer neural network with a sufficient number of neurons can approximate any continuous function to any desired accuracy.

Forward Pass and Backpropagation

Scalar's version

Forward Pass

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



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

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

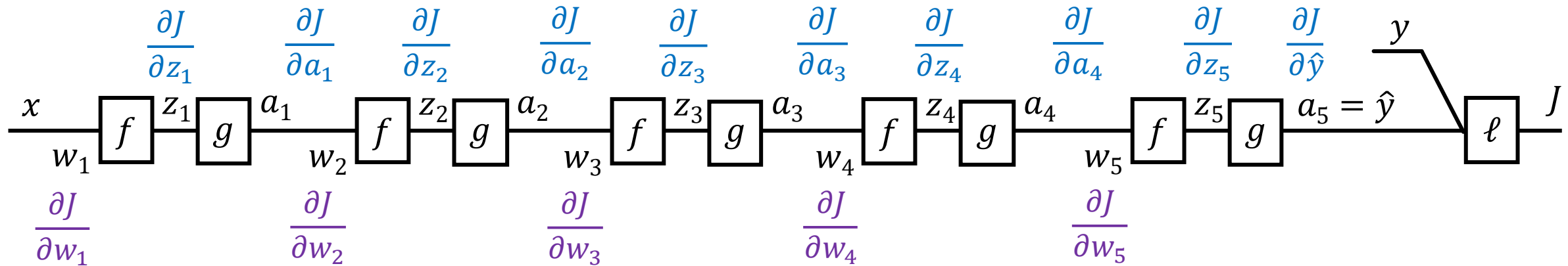
$$\hat{y} = h_{\theta}(\mathbf{x}) = g\left(w_5 \cdot g\left(w_4 \cdot g\left(w_3 \cdot g\left(w_2 \cdot g(w_1 \cdot x)\right)\right)\right)\right)$$

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

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

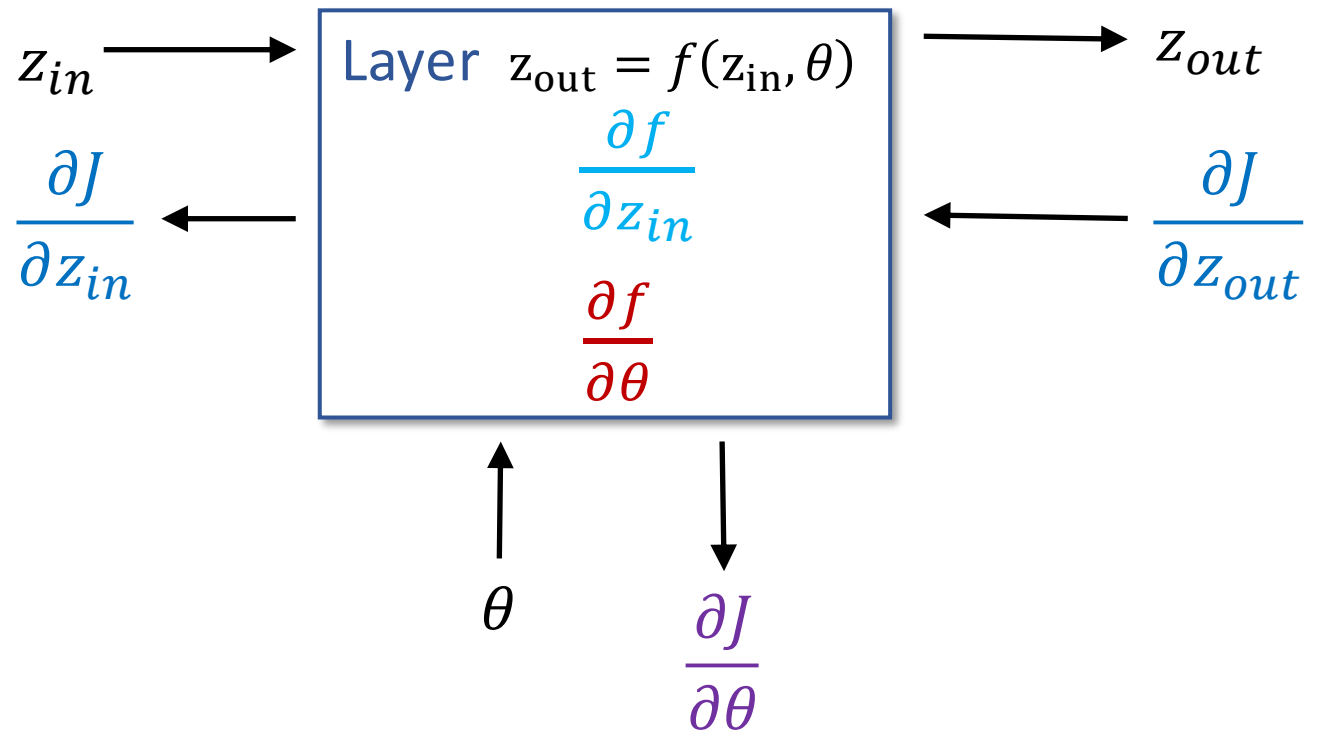
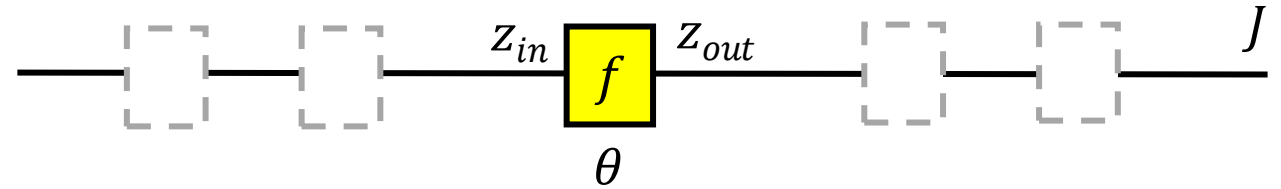
Objective: $J(\theta)$

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

Need:

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

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



Multivariate-chain rule

Multivariable Chain Rule

$$g_1(x) = 3x$$

$$g_2(x) = 5x$$

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

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

Exercise: Multivariable Chain Rule

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

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

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

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

Exercise: Multivariable Chain Rule

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

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

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

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

$$dg_1/dx = \cos(x)$$

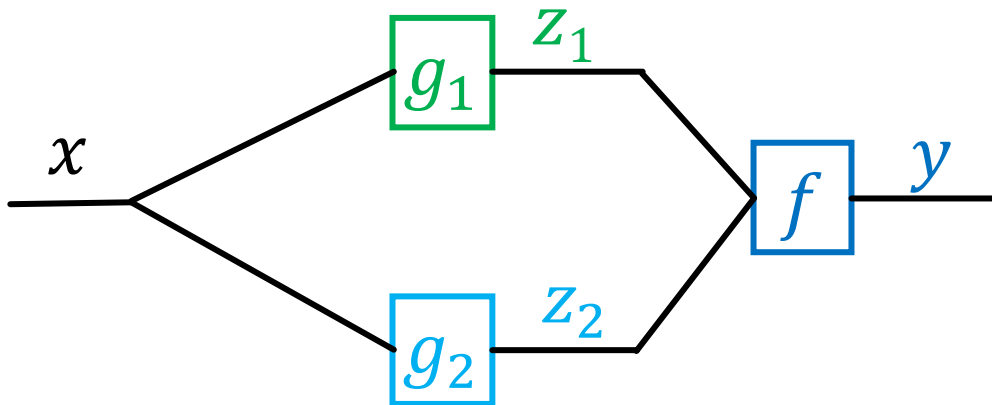
$$dg_2/dx = 3x^2$$

$$\partial f / \partial z_1 = 4z_1^3 e^{z_2} + 5$$

$$\partial f / \partial z_2 = z_1^4 e^{z_2} + 7$$

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

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



Calculus Chain Rule

Scalar:

$$y = f(z)$$

$$z = g(x)$$

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

Multivariate:

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

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

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

Multivariate:

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

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

$$\frac{dy_i}{dx_k} = \sum_j \frac{\partial y_i}{\partial z_j} \frac{\partial z_j}{\partial x_k}$$

Calculus Chain Rule

Scalar:

$$y = f(z)$$

$$z = g(x)$$

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

Multivariate:

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

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

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

Multivariate:

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

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

$$\frac{dy_i}{dx_k} = \sum_j \frac{\partial y_i}{\partial z_j} \frac{\partial z_j}{\partial x_k}$$

Multivariable Chain Rule

	Numerator layout	Denominator layout	Notes
$\frac{d}{dt} f(g(t), h(t))$	$\frac{df}{dg} \frac{dg}{dt} + \frac{df}{dh} \frac{dh}{dt}$	Same	$f : (\mathbb{R} \times \mathbb{R}) \rightarrow \mathbb{R}, t \in \mathbb{R}$ $g : \mathbb{R} \rightarrow \mathbb{R}, h : \mathbb{R} \rightarrow \mathbb{R}$
$\frac{d}{dt} f(g_1(t), \dots, g_N(t))$	$\sum_{i=1}^N \frac{df}{dg_i} \frac{dg_i}{dt}$	Same	$h : (\mathbb{R} \times \dots \times \mathbb{R}) \rightarrow \mathbb{R}, t \in \mathbb{R}$ $f_i : \mathbb{R} \rightarrow \mathbb{R} \quad \forall i \in \{1, \dots, N\}$
$\frac{d}{dt} f(\mathbf{g}(t))$	$\frac{\partial f}{\partial \mathbf{g}} \frac{\partial \mathbf{g}}{\partial t}$	$\frac{\partial \mathbf{g}}{\partial t} \frac{\partial f}{\partial \mathbf{g}}$	$f : \mathbb{R}^N \rightarrow \mathbb{R}, \mathbf{g} : \mathbb{R} \rightarrow \mathbb{R}^N$ $t \in \mathbb{R}$
$\frac{\partial}{\partial \mathbf{v}} f(\mathbf{g}(\mathbf{v}))$	$\frac{\partial f}{\partial \mathbf{g}} \frac{\partial \mathbf{g}}{\partial \mathbf{v}}$	$\frac{\partial \mathbf{g}}{\partial \mathbf{v}} \frac{\partial f}{\partial \mathbf{g}}$	$f : \mathbb{R}^N \rightarrow \mathbb{R}, \mathbf{g} : \mathbb{R}^M \rightarrow \mathbb{R}^N$ $\mathbf{v} \in \mathbb{R}^M$
$\frac{\partial}{\partial \mathbf{v}} f(\mathbf{g}(\mathbf{v}), \mathbf{h}(\mathbf{v}))$	$\frac{\partial f}{\partial \mathbf{g}} \frac{\partial \mathbf{g}}{\partial \mathbf{v}} + \frac{\partial f}{\partial \mathbf{h}} \frac{\partial \mathbf{h}}{\partial \mathbf{v}}$	$\frac{\partial \mathbf{g}}{\partial \mathbf{v}} \frac{\partial f}{\partial \mathbf{g}} + \frac{\partial \mathbf{h}}{\partial \mathbf{v}} \frac{\partial f}{\partial \mathbf{h}}$	$h : \mathbb{R}^K \times \mathbb{R}^N \rightarrow \mathbb{R}, \mathbf{v} \in \mathbb{R}^M$ $f : \mathbb{R}^M \rightarrow \mathbb{R}^K, \mathbf{g} : \mathbb{R}^M \rightarrow \mathbb{R}^N$

Backpropagation (vector version)

Generic Layer Implementation (updated)

Compute derivatives per layer, utilizing previous derivatives

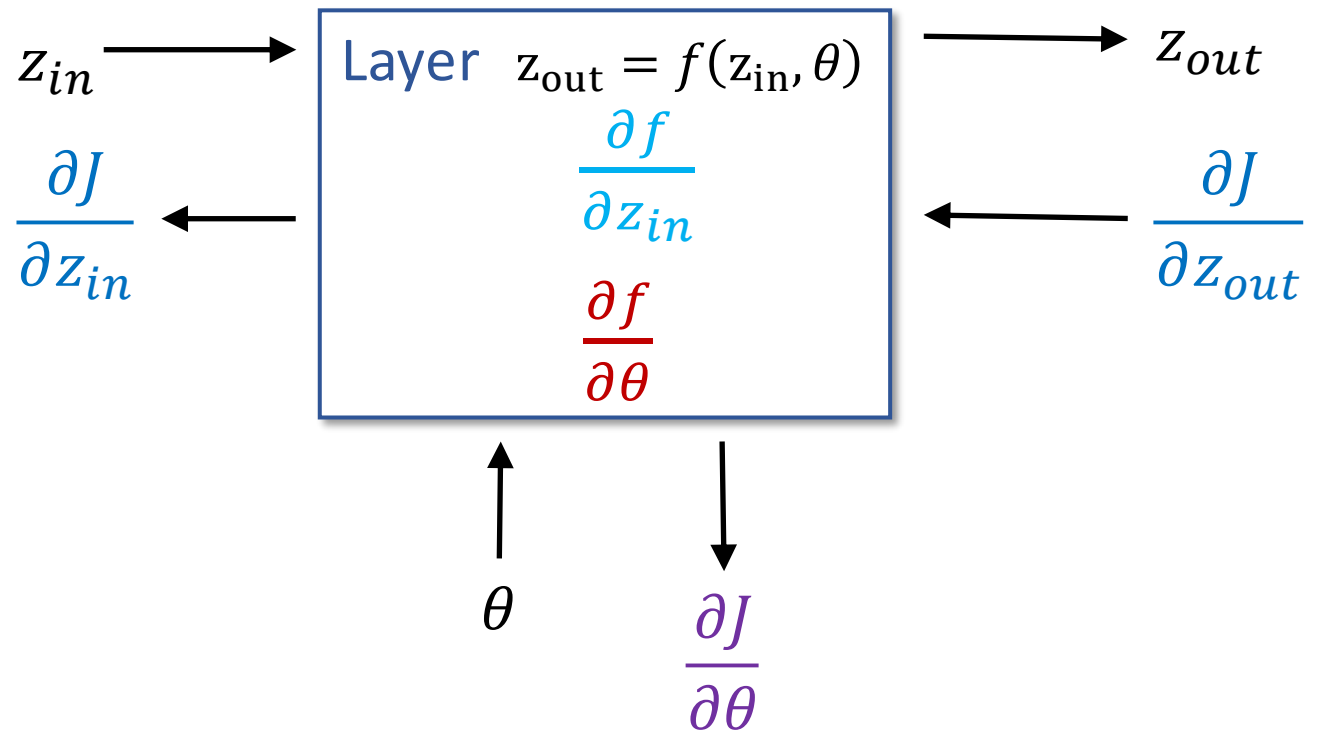
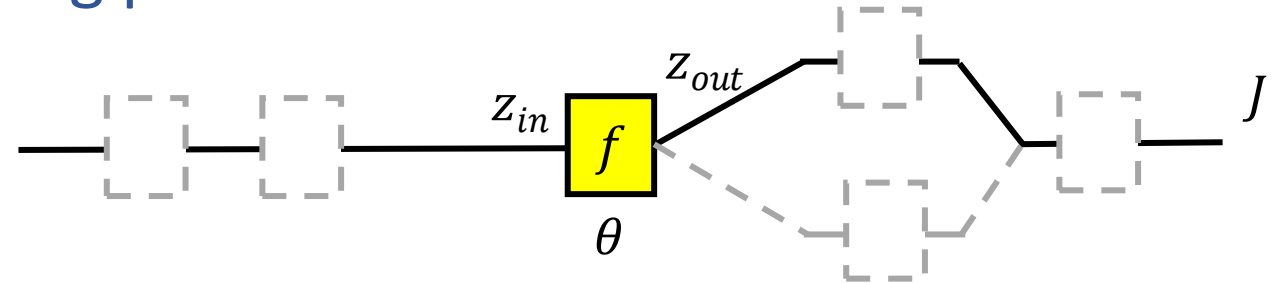
Objective: $J(\theta)$

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

Need:

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

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



Generic Layer Implementation (vector output version)

Compute derivatives per layer, utilizing previous derivatives

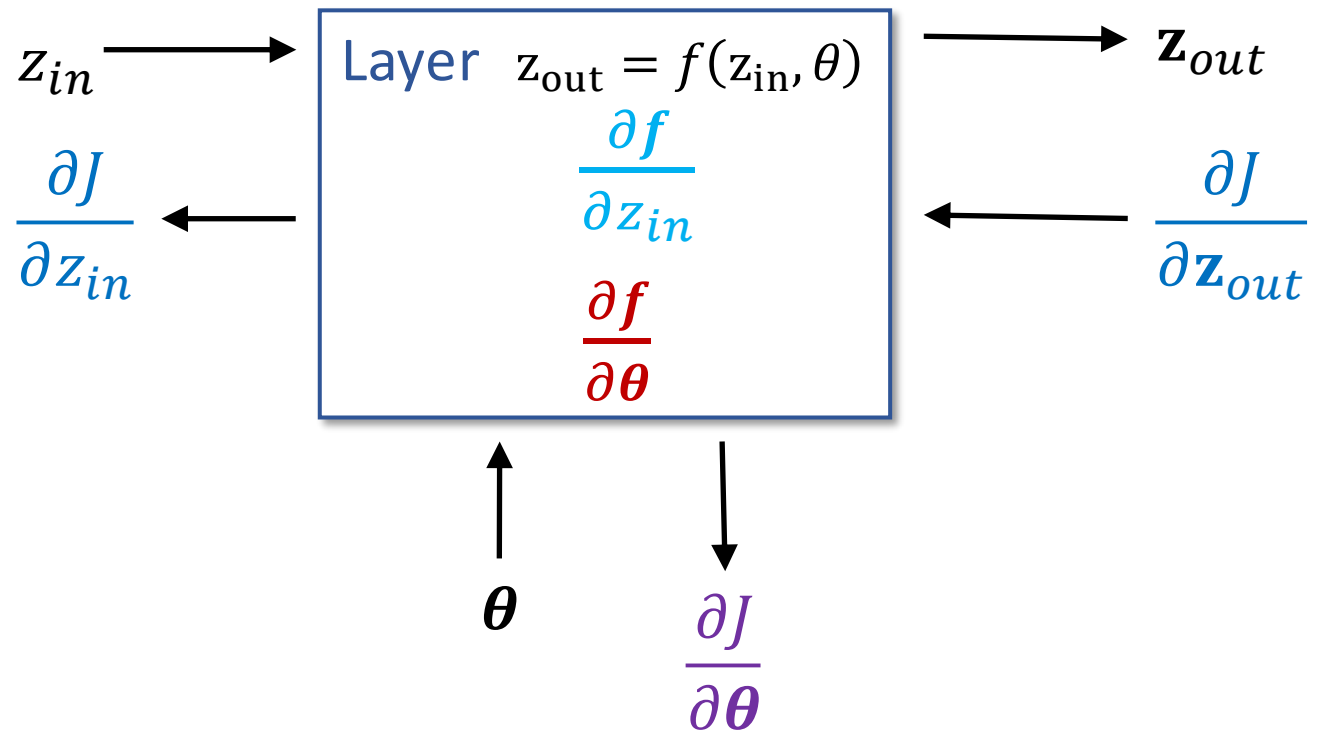
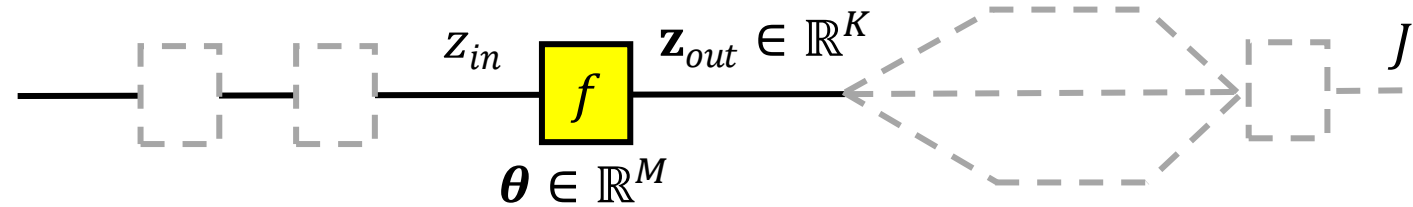
Objective: $J(\theta)$

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

Need:

- $\frac{\partial J}{\partial \mathbf{z}_{\text{in}}} =$

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



Generic Layer Implementation (vector output version)

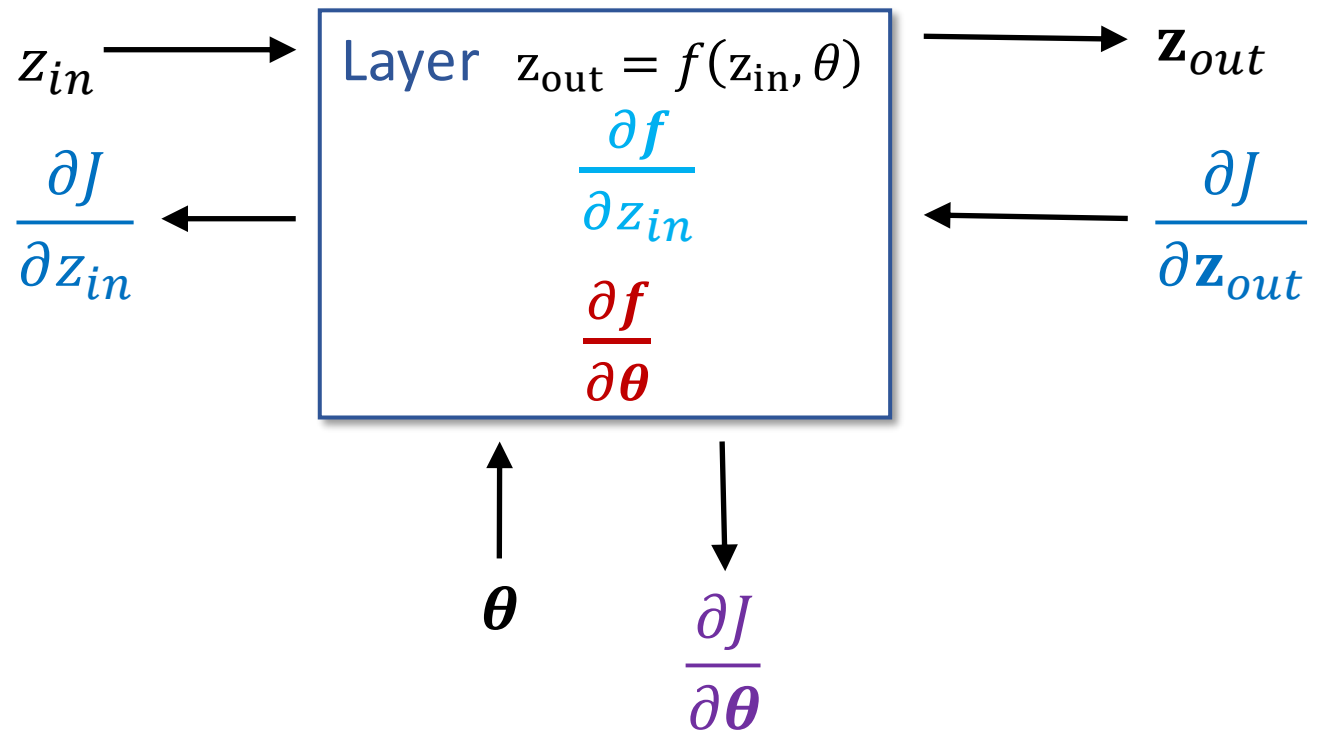
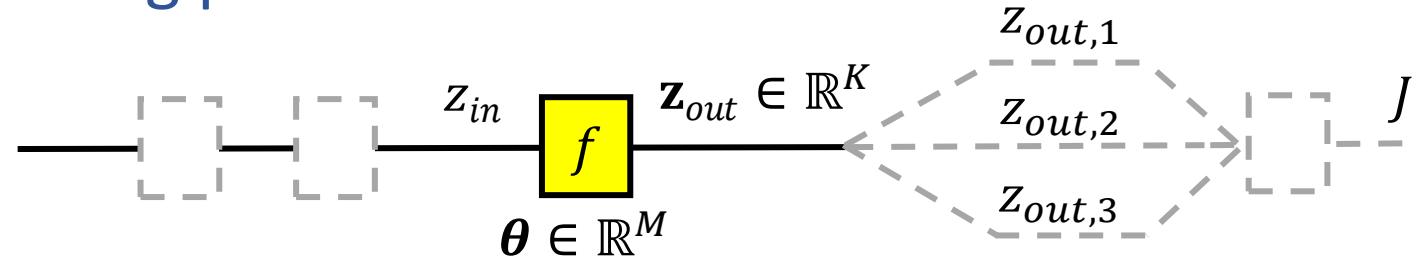
Compute derivatives per layer, utilizing previous derivatives

Objective: $J(\theta)$

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

Need:

- $\frac{\partial J}{\partial \mathbf{z}_{\text{in}}} = \sum_k \frac{\partial J}{\partial \mathbf{z}_{\text{out},k}} \frac{\partial \mathbf{z}_{\text{out},k}}{\partial \mathbf{z}_{\text{in}}}$
- $\frac{\partial J}{\partial \theta_j} = \sum_k \frac{\partial J}{\partial \mathbf{z}_{\text{out},k}} \frac{\partial f}{\partial \theta_j}$



Linear Layer Implementation

Compute derivatives per layer

Objective: $J(\theta)$

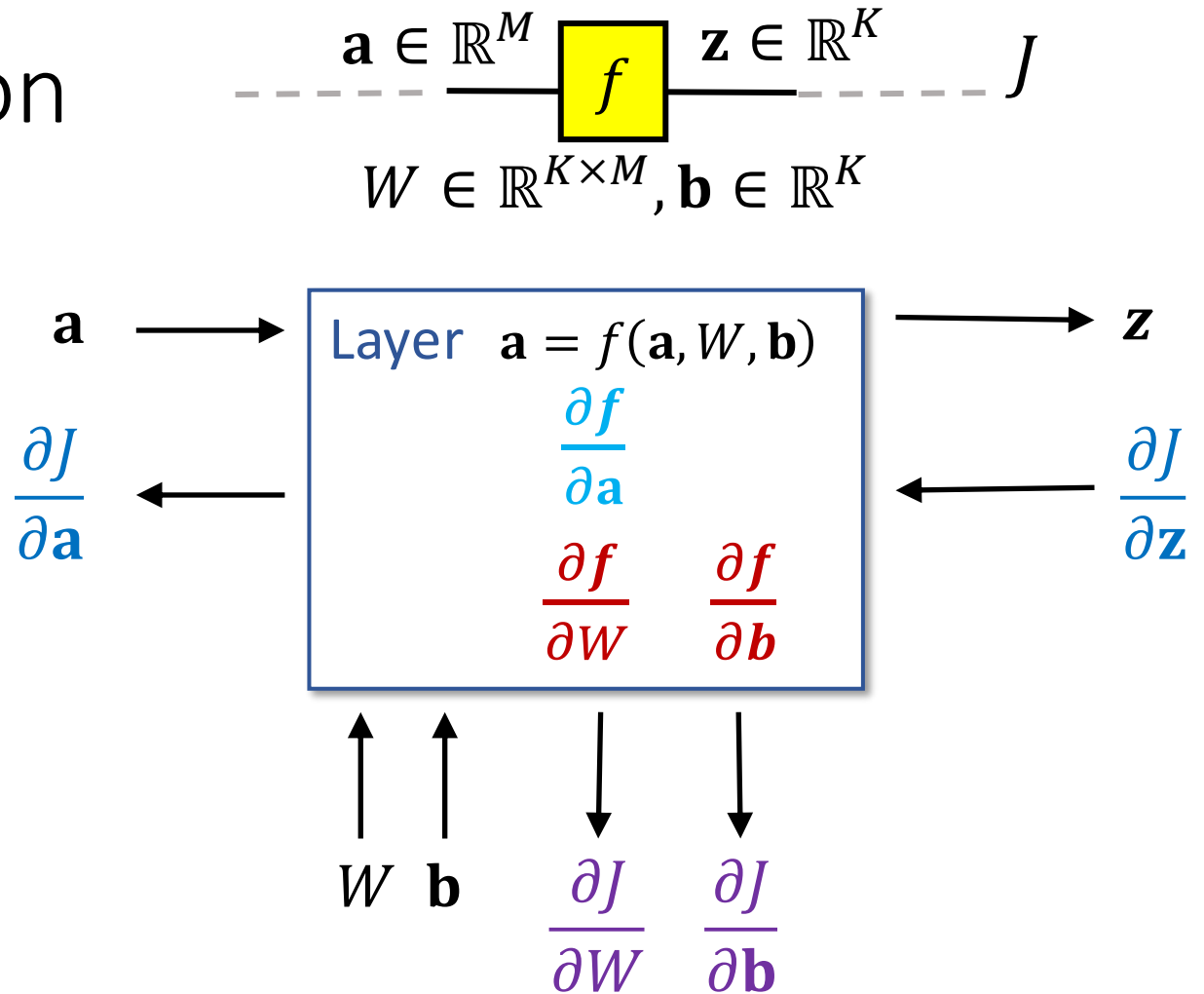
Layer: $\mathbf{z} = f(\mathbf{a}, W, \mathbf{b}) = W\mathbf{a} + \mathbf{b}$

Scalar version (no formatting)

$$\blacksquare \frac{\partial J}{\partial a_j} =$$

$$\blacksquare \frac{\partial J}{\partial W_{i,j}} =$$

$$\blacksquare \frac{\partial J}{\partial b_i} =$$



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

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

Linear Layer Implementation

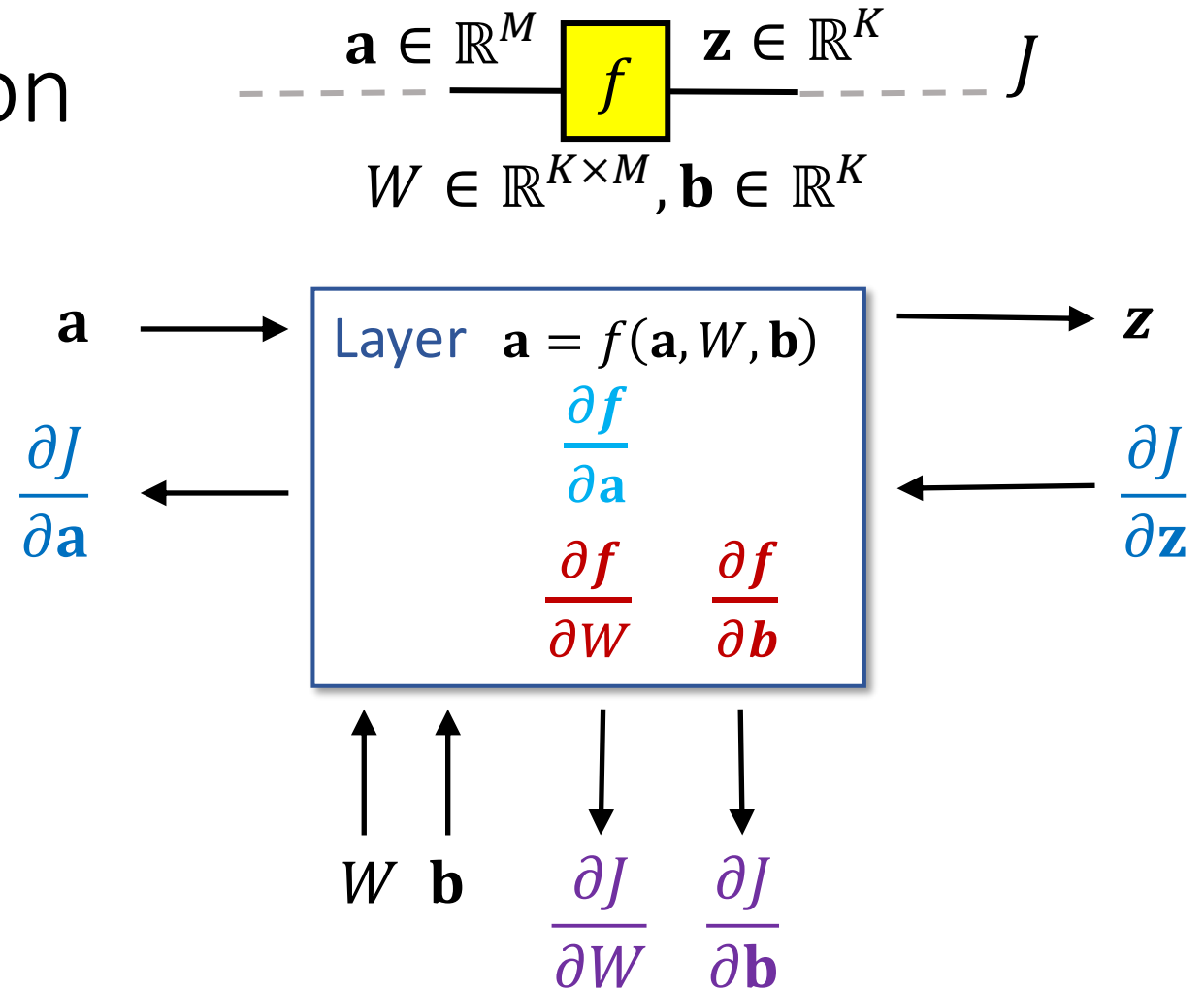
Compute derivatives per layer

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

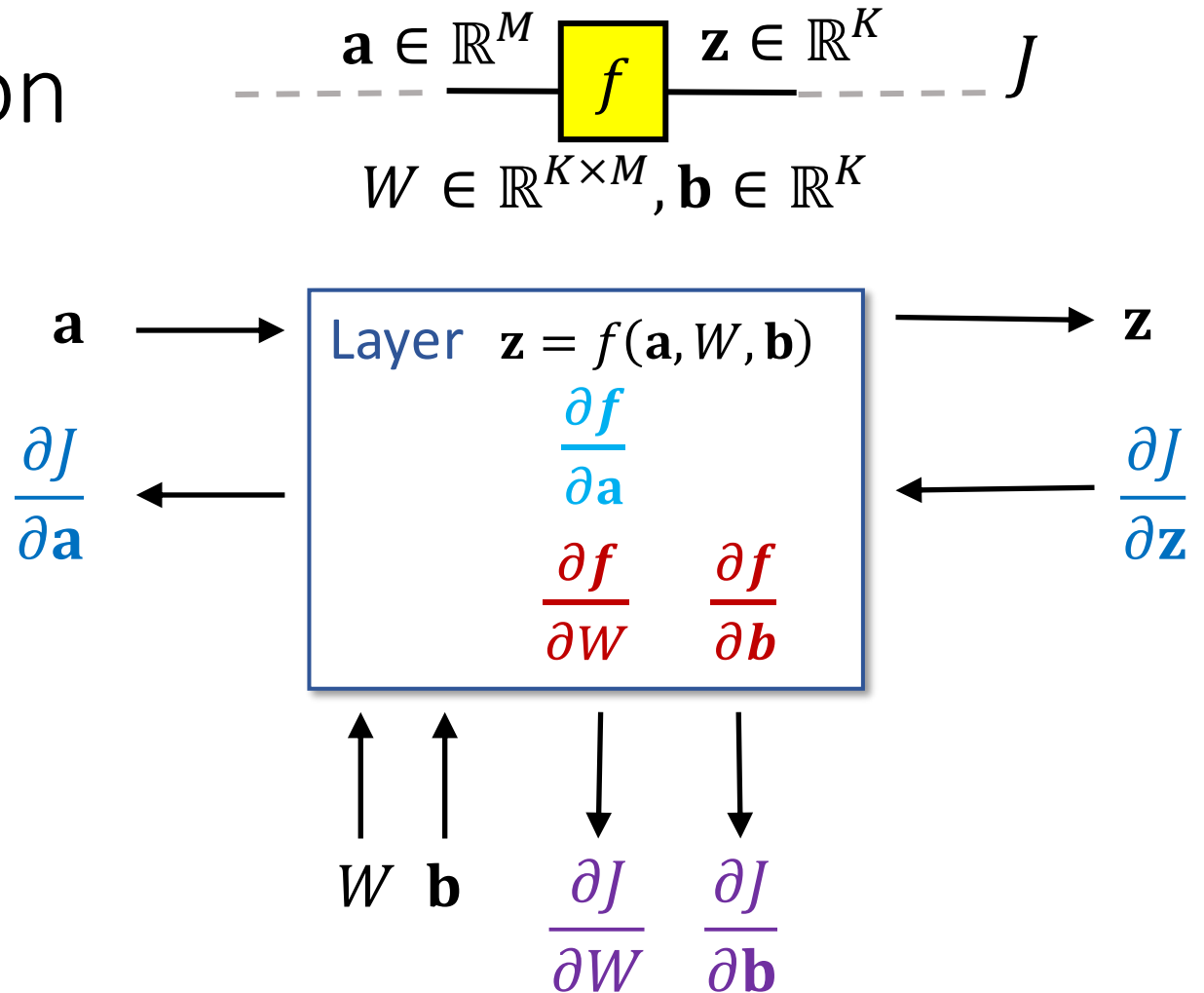
Objective: $J(\theta)$

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

- $\frac{\partial J}{\partial \mathbf{a}}$

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

- $\frac{\partial J}{\partial \mathbf{b}}$



Linear Layer Implementation

Compute derivatives per layer

Objective: $J(\theta)$

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

(Numerator format)

- $\frac{\partial J}{\partial \mathbf{a}}$

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

- $\frac{\partial J}{\partial \mathbf{b}}$

