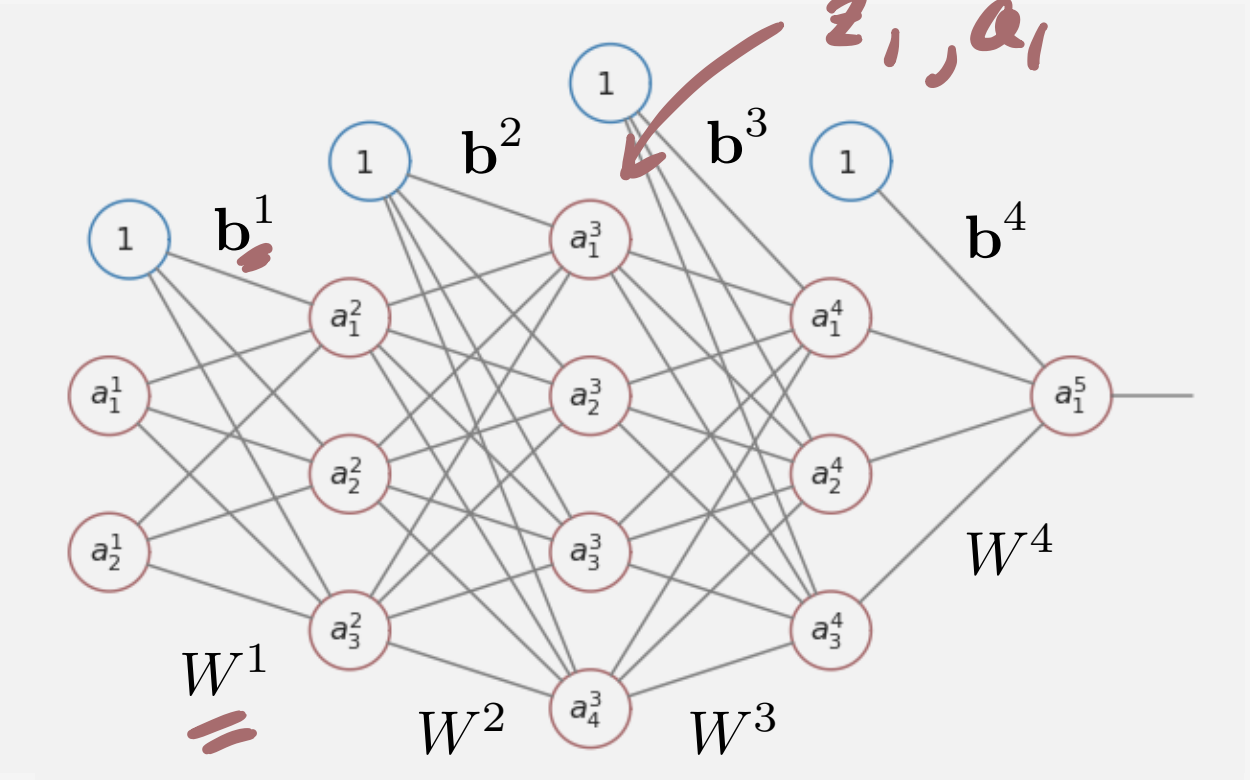
Neural Networks Part III

Back Propagation Part I

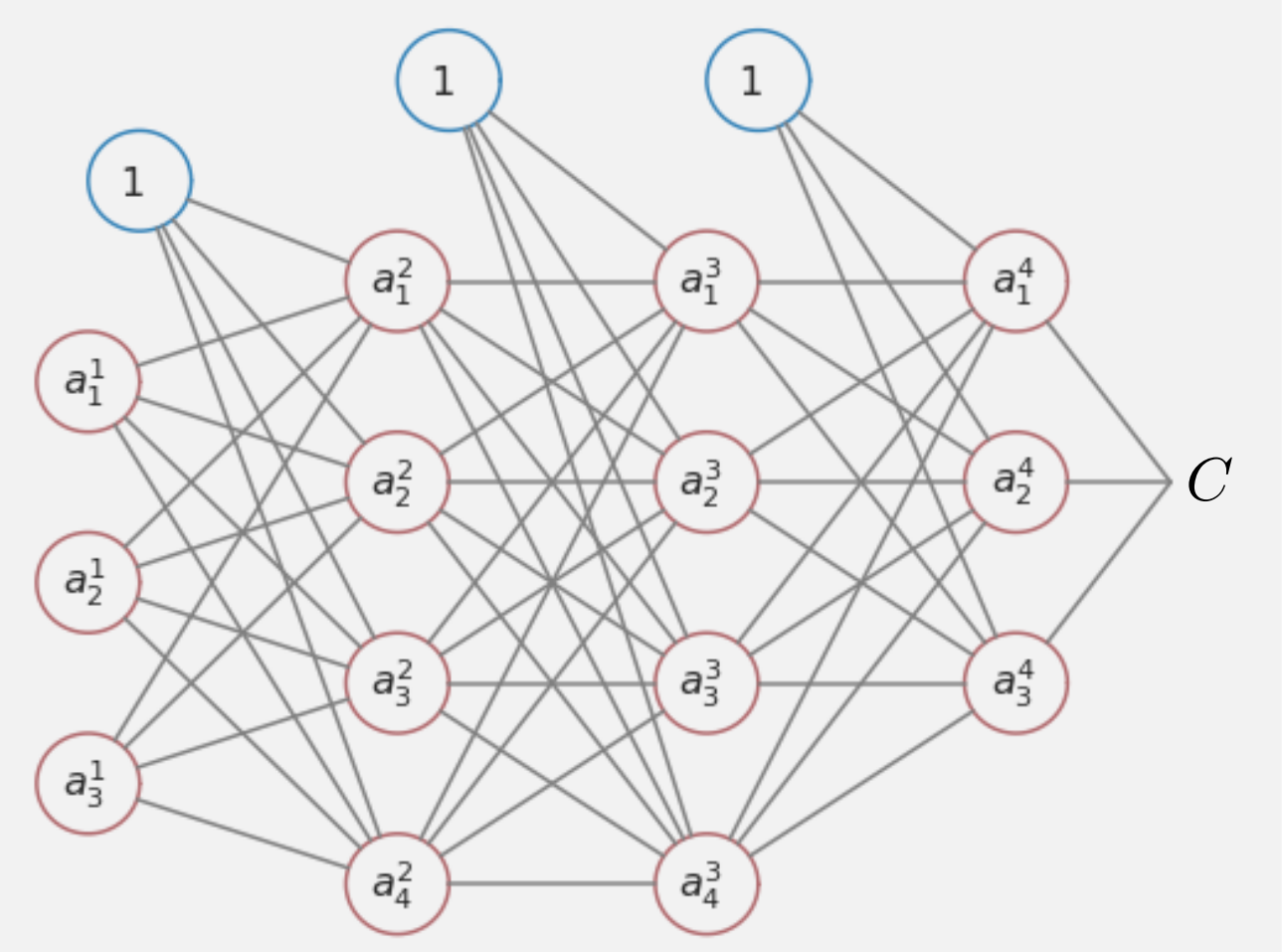
The Feed-Forward Neural Network

String together lots of neurons



Each transition between layers has associated set of weights an biases

Loss Functions



Regression or Classification:

$$\frac{1}{2}\sum_{i=1}^{n} \|y(x_i) - \mathbf{a}^L\|^2$$

Single-Output Classification:

$$-\sum_{i=1}^{n} \{y_i \log(a^L) + (1 - y_i) \log(1 - a^L)\}$$

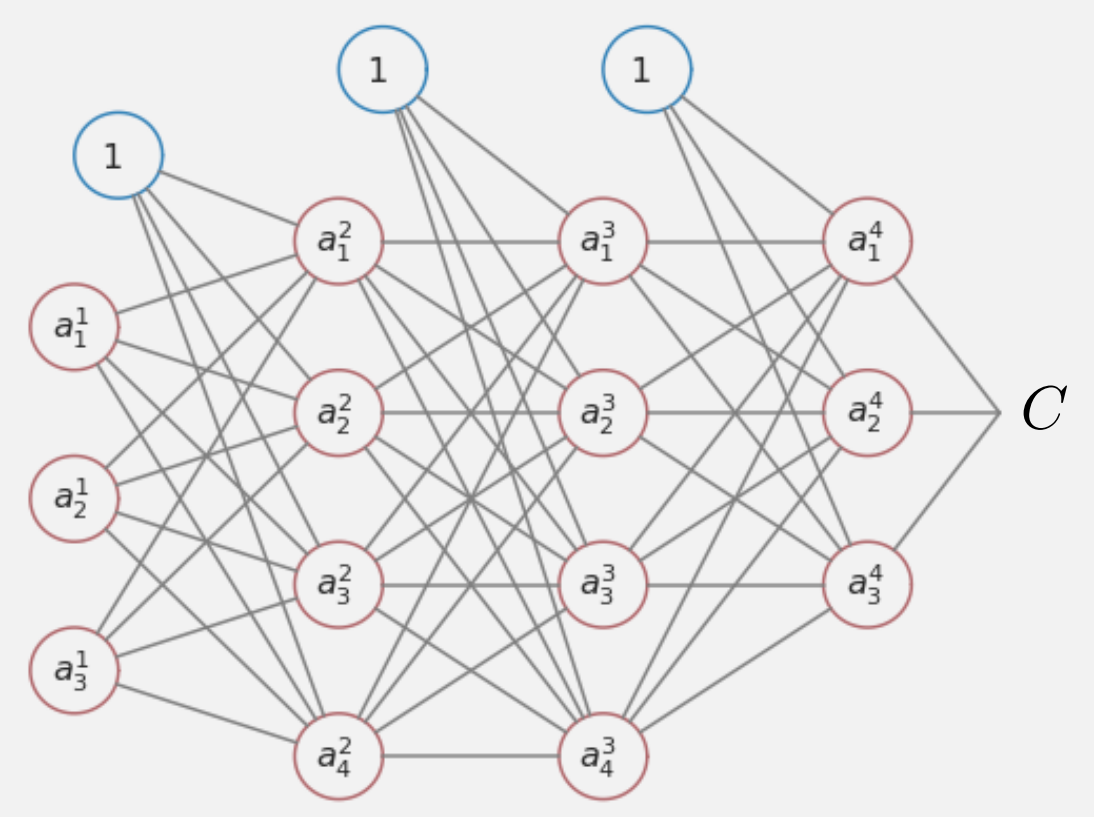
Multiple-Output Classification:

$$-\sum_{i=1}^{n}\sum_{c=1}^{C} I(y_i = c) \log\left(\frac{\exp[a_c^L]}{\sum_{k=1}^{C} \exp[a_k^L]}\right)$$

Predictions: Forward Propagation

Suppose we've learned the weights in a NN with sigmoid neurons.

Given an input $\mathbf{x}$, make predictions with Forward Prop



Re: $\mathbf{a}^{\ell+1} = g(\mathbf{z}^{\ell+1})$ where $\mathbf{z}^{\ell+1} = W^\ell \mathbf{a}^\ell + \mathbf{b}^\ell$

Suppose the network has L layers

Initialize $\mathbf{a}^1 = \mathbf{x}$

for $\ell = 1, \ldots, L-1$

$$\mathbf{z}^{\ell+1} = W^\ell \mathbf{a}^\ell + \mathbf{b}^\ell$$

$$\mathbf{a}^{\ell+1} = g(\mathbf{z}^{\ell+1})$$

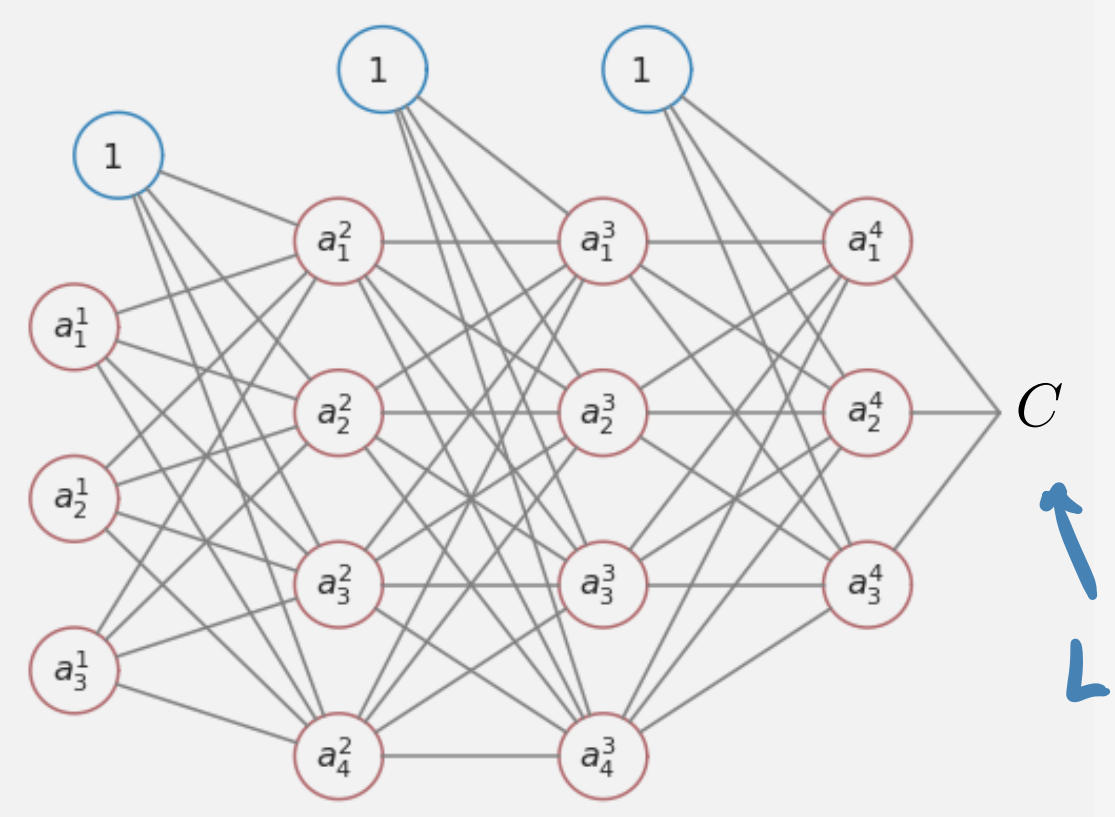
Activation function

If classification and sigmoid neuron output

then predict based on $\mathbf{a}^L$

Learning Weights and Biases

Given training data, we want to use Stochastic Gradient Descent to learn weights and biases



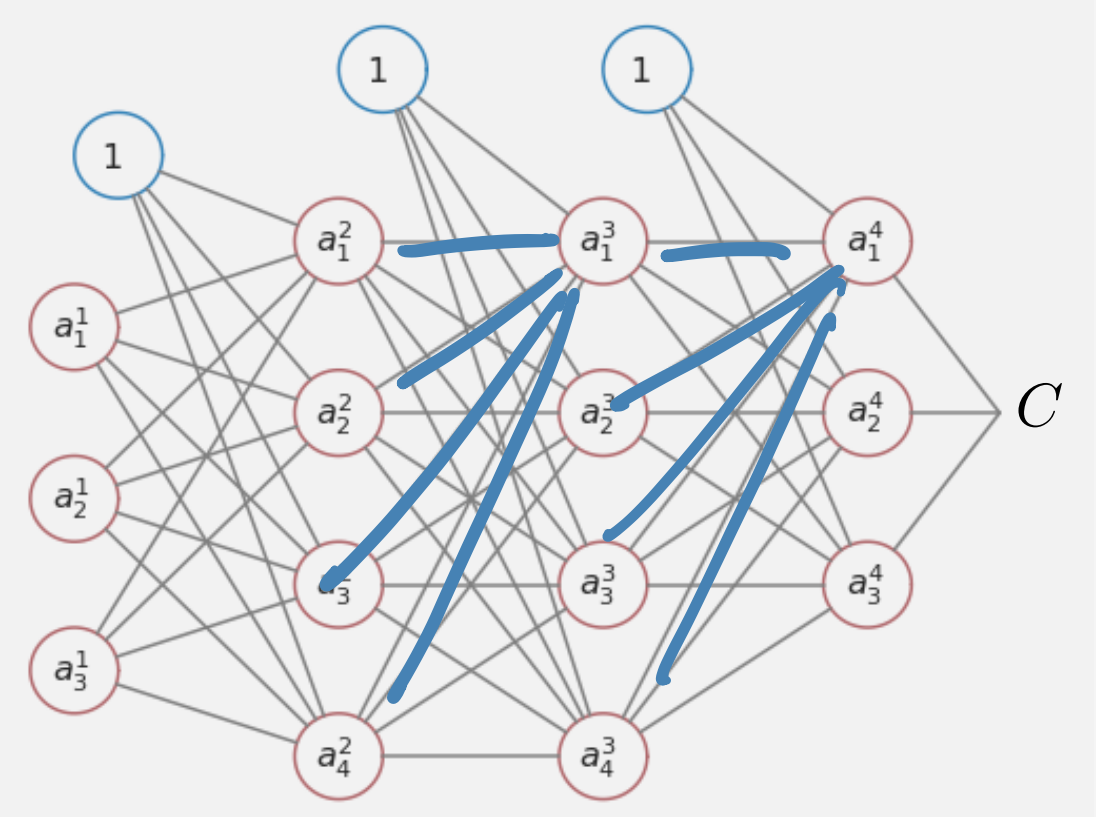
Usual SGD update:

$$w_{ij}^{\ell} \leftarrow w_{ij}^{\ell} - \eta \frac{\partial C}{\partial w_{ij}^{\ell}}$$

$$b_i^{\ell} \leftarrow b_i^{\ell} - \eta \frac{\partial C}{\partial b_i^{\ell}}$$

Learning Weights and Biases

Given training data, we want to use Stochastic Gradient Descent to learn weights and biases



Usual SGD update:

$$w_{ij}^{\ell} \leftarrow w_{ij}^{\ell} - \eta \frac{\partial C}{\partial w_{ij}^{\ell}}$$

$$b_i^{\ell} \leftarrow b_i^{\ell} - \eta \frac{\partial C}{\partial b_i^{\ell}}$$

Challenge: Computing partial derivatives

The Chain Rule

The Chain Rule allows us to take derivatives of nested functions

There are two forms of the Chain Rule

The Baby Chain Rule:

$$\frac{d}{dx}f(g(x)) = f'(g(x))g'(x) = \frac{df}{dg}\frac{dg}{dx}$$

OUTSIDE

INSIDE

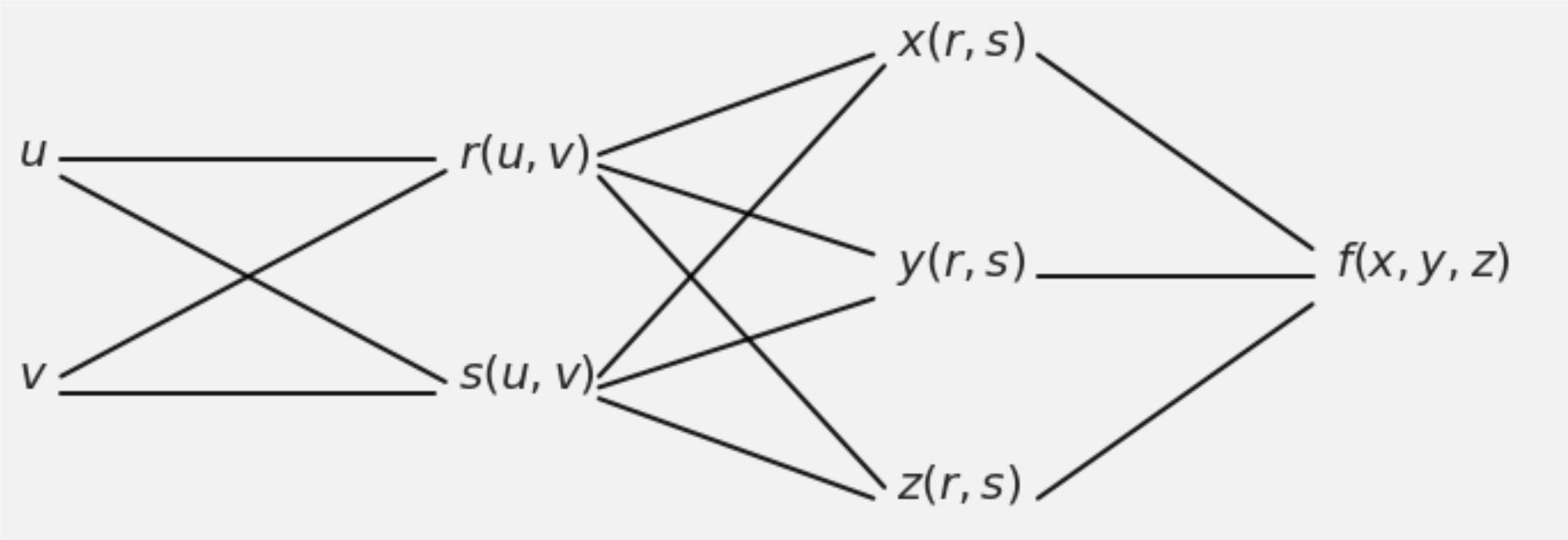
Example: Compute $\frac{d}{dx}sin(x^2) =$

$$\cos(x^2)\frac{d}{dx}(x^2)$$

$$= \cos(x^2)(2x)$$

The Chain Rule

The Full-Grown Adult Chain Rule:

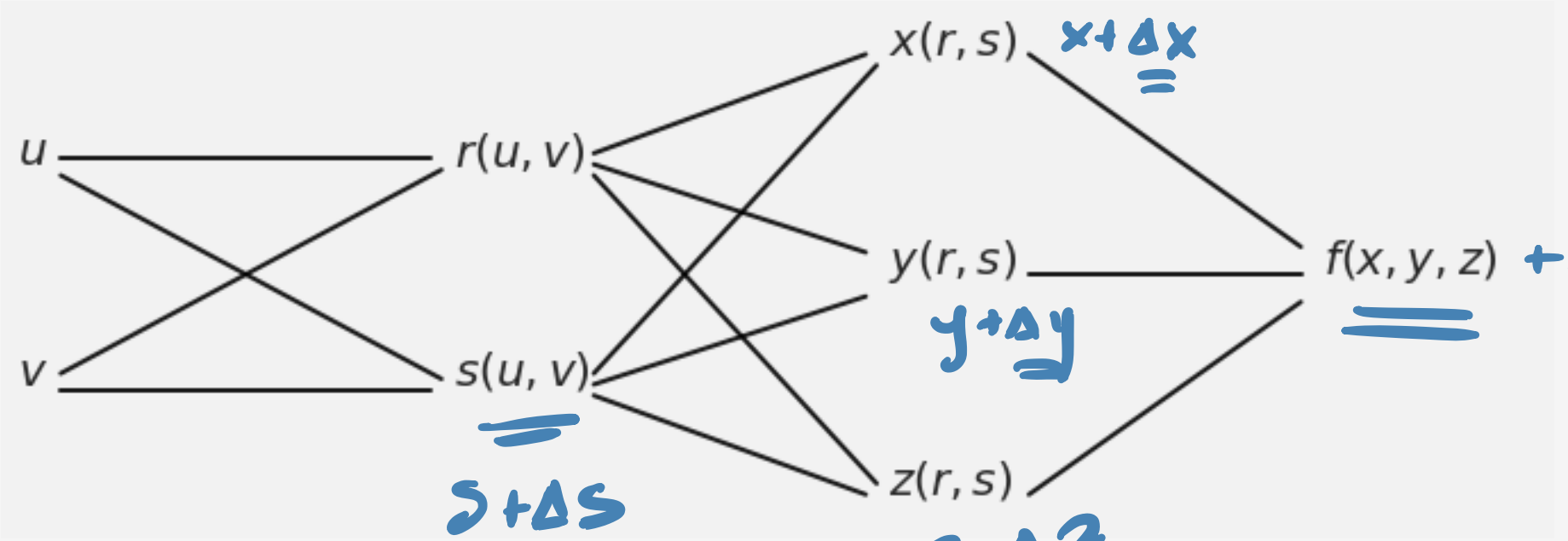


The derivative of f with respect to x is $\frac{\partial f}{\partial x}$

The Chain Rule

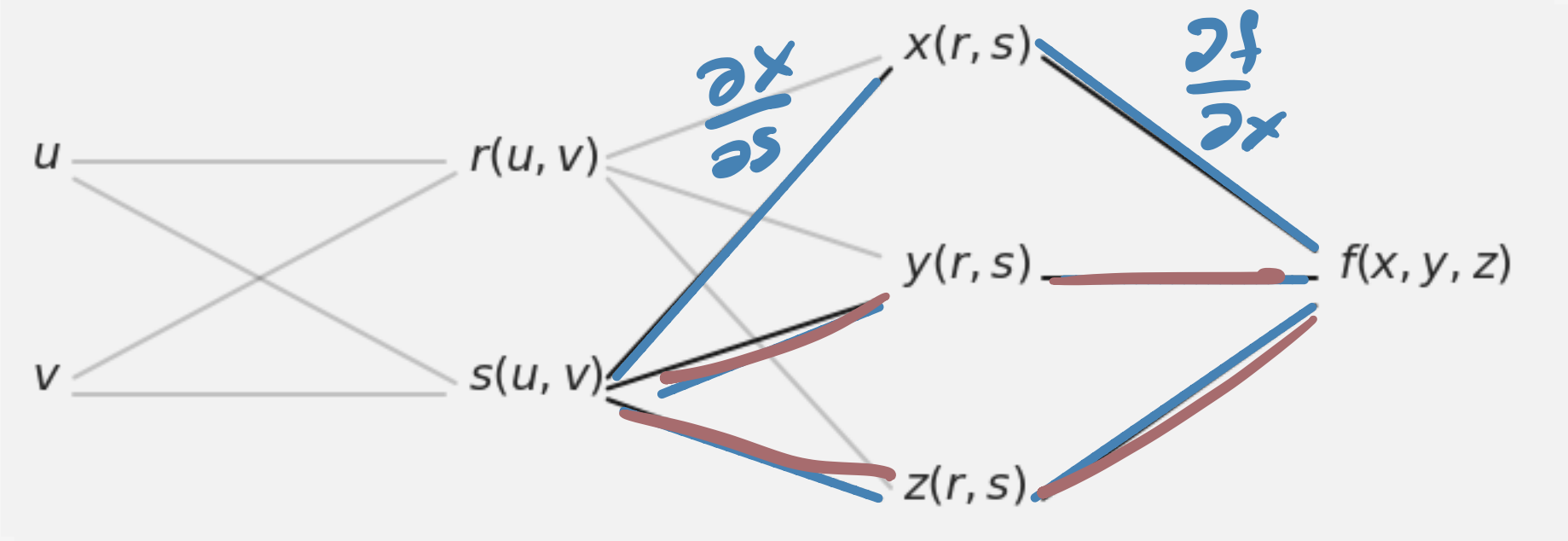
What is the derivative of f with respect to s?

$$\frac{\partial f}{\partial s} \approx \frac{\Delta f}{\Delta s}$$



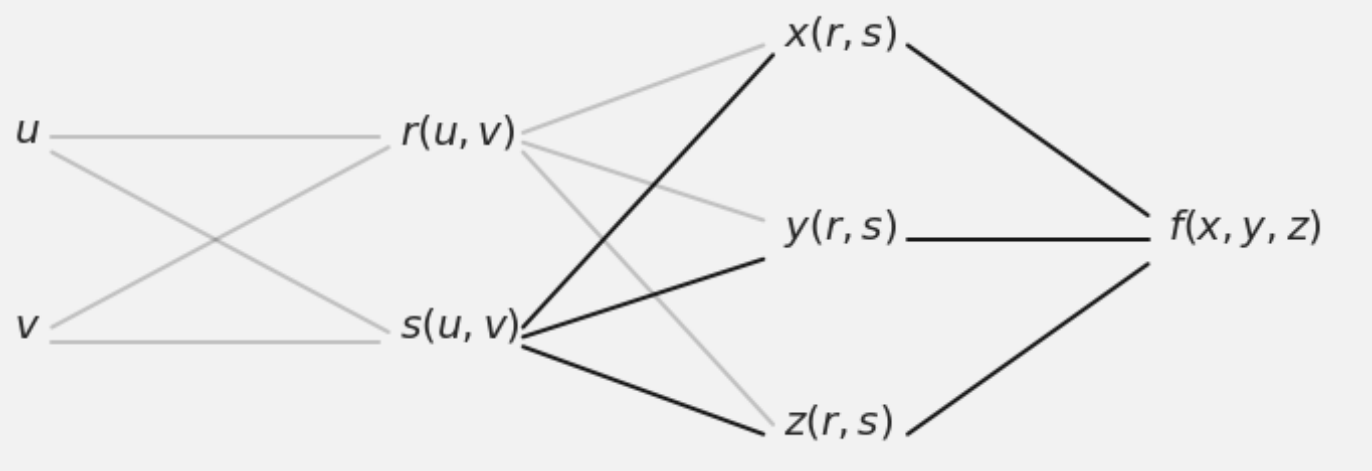
The Chain Rule

What is the derivative of f with respect to s?



$$\frac{\partial f}{\partial s} = \frac{\partial f}{\partial x}\frac{\partial x}{\partial s} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial s} + \frac{\partial f}{\partial z}\frac{\partial z}{\partial s}$$

The Chain Rule



$$\frac{\partial f}{\partial s} = \frac{\partial f}{\partial x}\frac{\partial x}{\partial s} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial s} + \frac{\partial f}{\partial z}\frac{\partial z}{\partial s}$$

$yz \cdot 0 \quad xz \cdot 1 \quad xy \cdot 1$

$= yz \cdot 0 + xz \cdot r + xy \cdot 1$

$= xzr + xy$

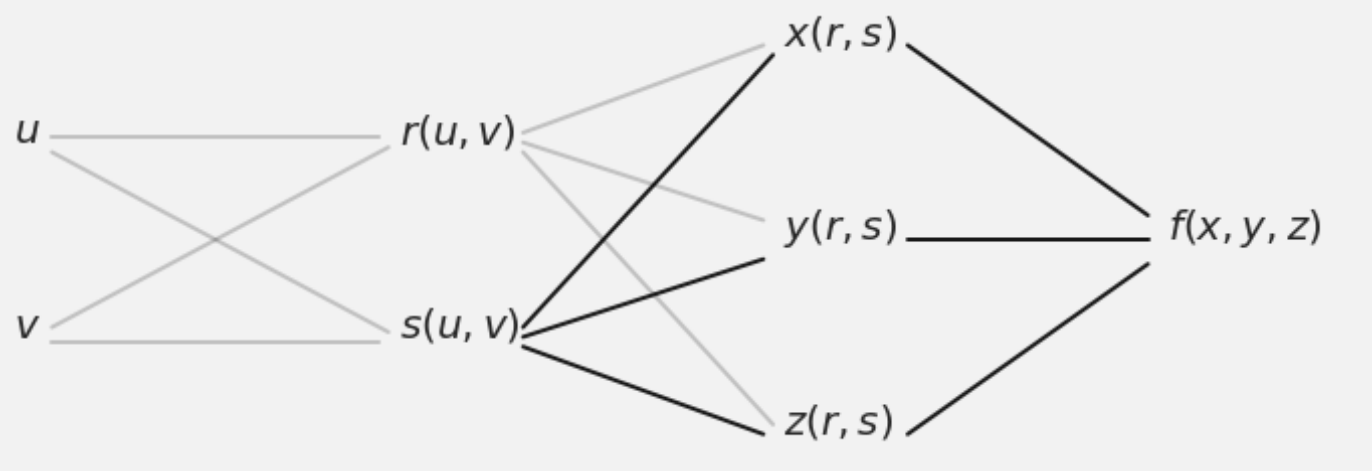
$= r^2 s + r^2 s$

$= 2r^2 s$

Example: Let. $f = xyz$, $x = r$, $y = rs$, and $z = s$. Find $\frac{\partial f}{\partial s}$

$\frac{\partial f}{\partial x} = \frac{\partial}{\partial x}(xyz) = 1 \cdot yz = yz$

The Chain Rule



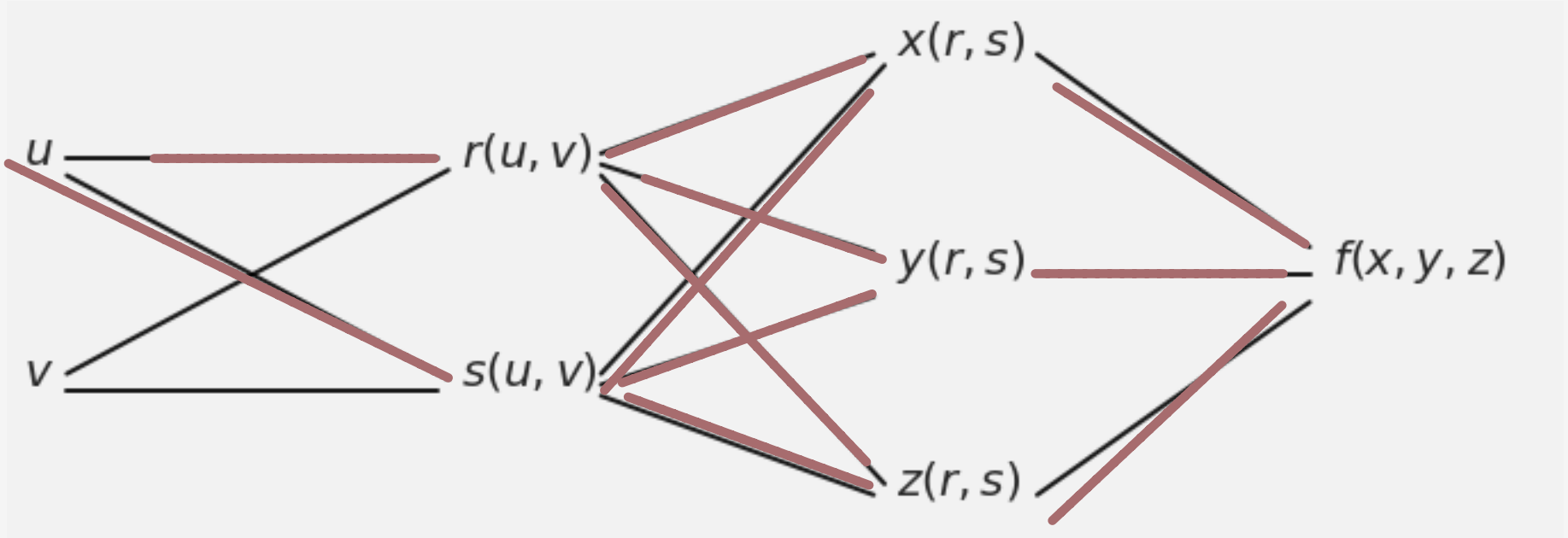
$$\frac{\partial f}{\partial s} = \frac{\partial f}{\partial x}\frac{\partial x}{\partial s} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial s} + \frac{\partial f}{\partial z}\frac{\partial z}{\partial s}$$

Example: Let. $f = xyz$, $x = r$, $y = rs$, and $z = s$. Find $\frac{\partial f}{\partial s}$

$f = (r)(rs)(s) = r^2s^2 \Rightarrow \frac{\partial f}{\partial s} = 2r^2s$

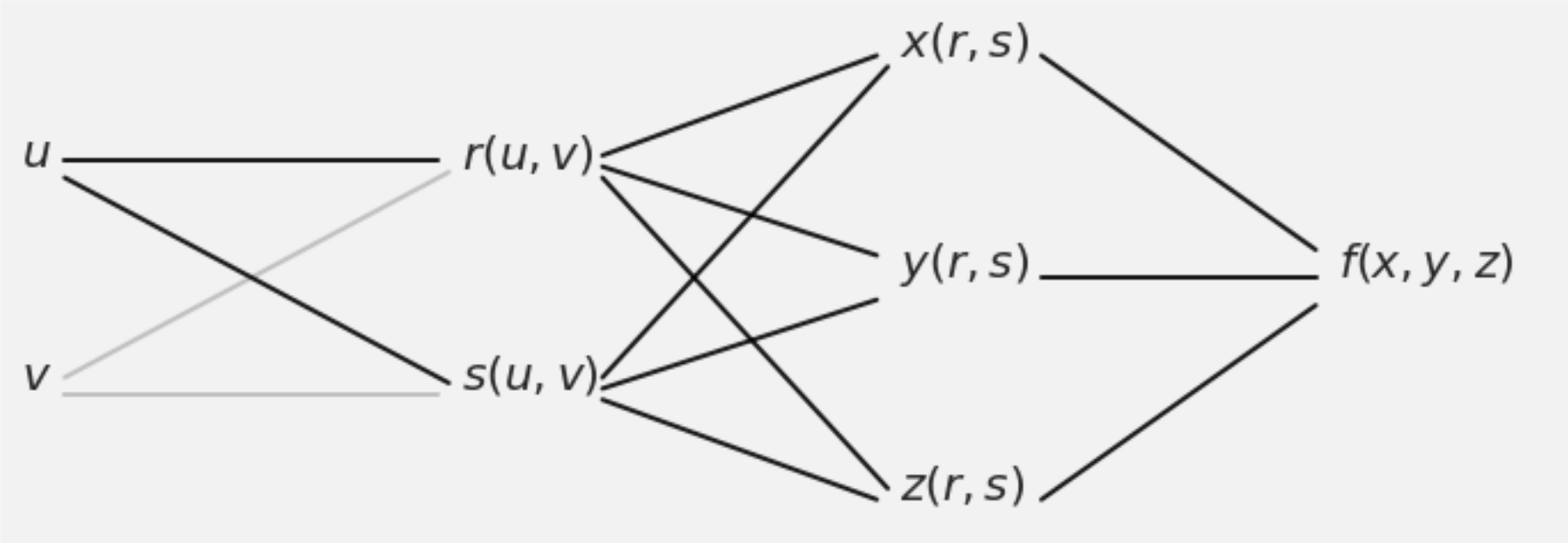
The Chain Rule

What is the derivative of f with respect to u?



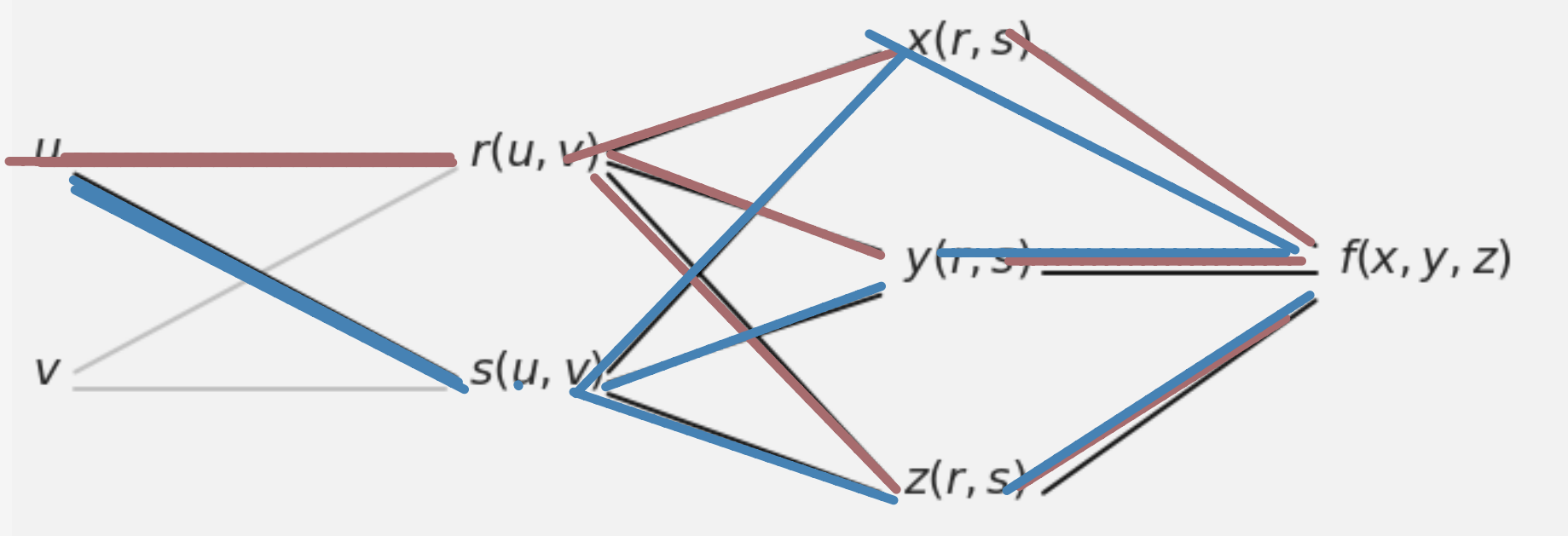
The Chain Rule

What is the derivative of f with respect to u?



The Chain Rule

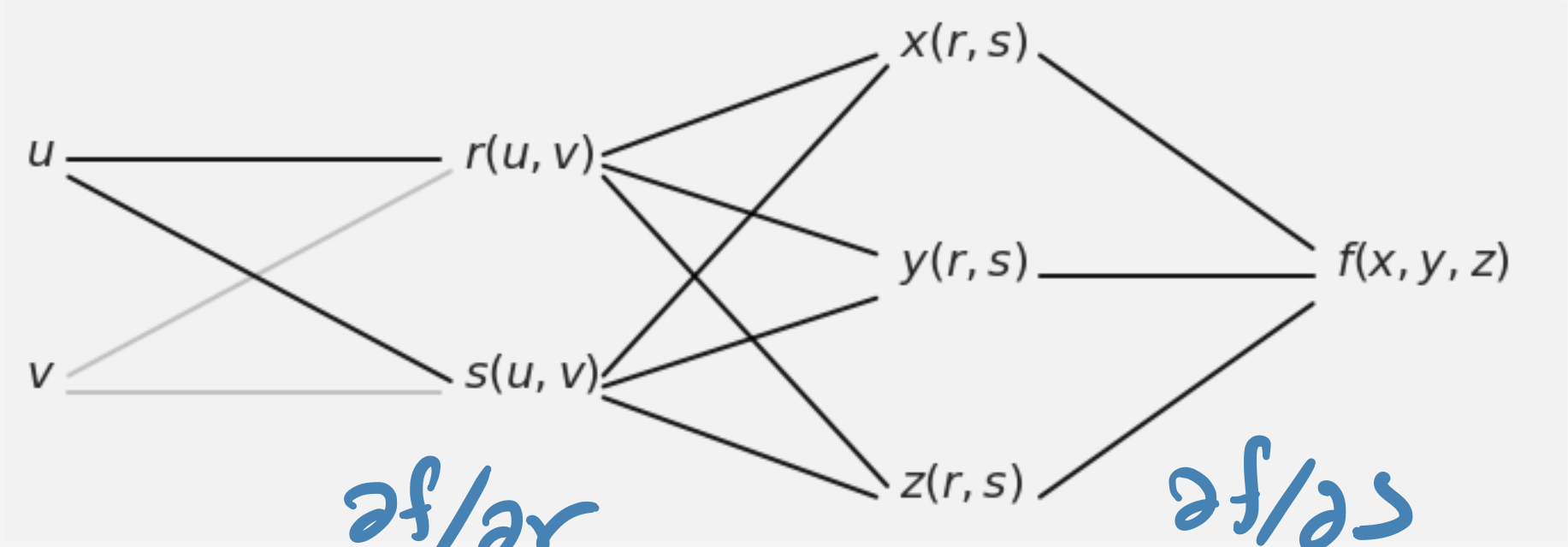
What is the derivative of f with respect to u?



$$\frac{\partial f}{\partial u} = \left(\frac{\partial f}{\partial x}\frac{\partial x}{\partial r}\frac{\partial r}{\partial u} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial r}\frac{\partial r}{\partial u} + \frac{\partial f}{\partial z}\frac{\partial z}{\partial r}\frac{\partial r}{\partial u} \right) + \left(\frac{\partial f}{\partial x}\frac{\partial x}{\partial s}\frac{\partial s}{\partial u} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial s}\frac{\partial s}{\partial u} + \frac{\partial f}{\partial z}\frac{\partial z}{\partial s}\frac{\partial s}{\partial u} \right)$$

The Chain Rule

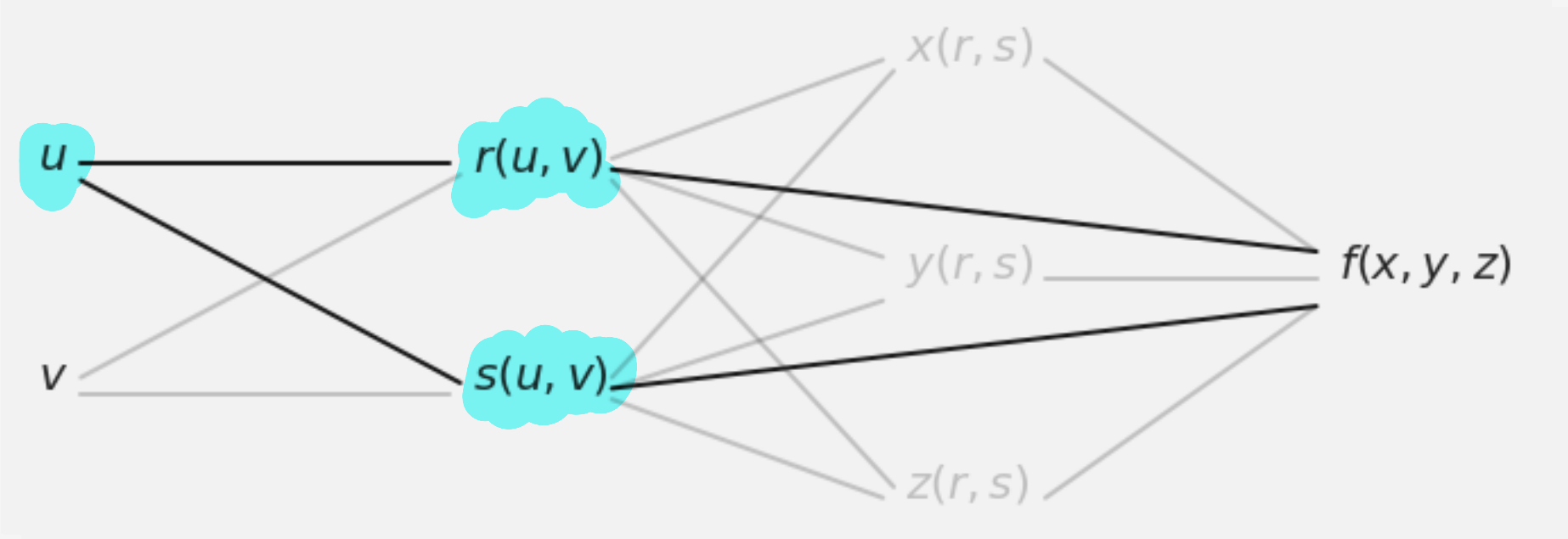
What is the derivative of f with respect to u?



$$\frac{\partial f}{\partial u} = \left(\frac{\partial f}{\partial x}\frac{\partial x}{\partial r} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial r} + \frac{\partial f}{\partial z}\frac{\partial z}{\partial r}\right)\frac{\partial r}{\partial u} + \left(\frac{\partial f}{\partial x}\frac{\partial x}{\partial s} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial s} + \frac{\partial f}{\partial z}\frac{\partial z}{\partial s}\right)\frac{\partial s}{\partial u}$$

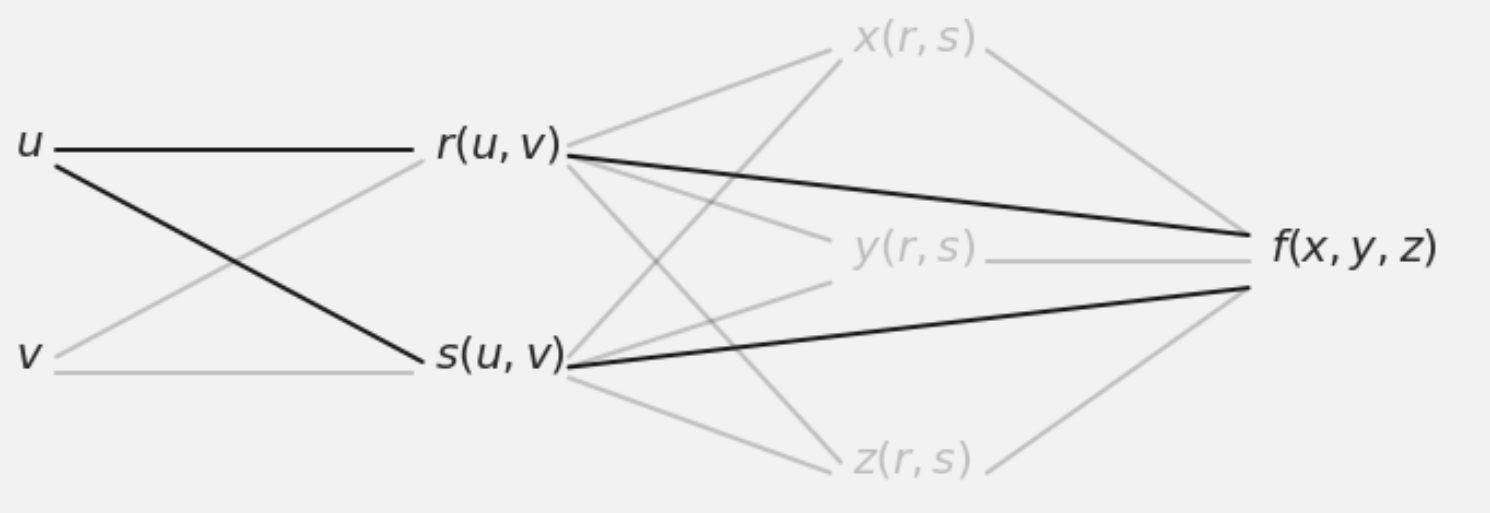
The Chain Rule

What is the derivative of f with respect to u?



$$\frac{\partial f}{\partial u} = \frac{\partial f}{\partial r}\frac{\partial r}{\partial u} + \frac{\partial f}{\partial s}\frac{\partial s}{\partial u}$$

The Chain Rule



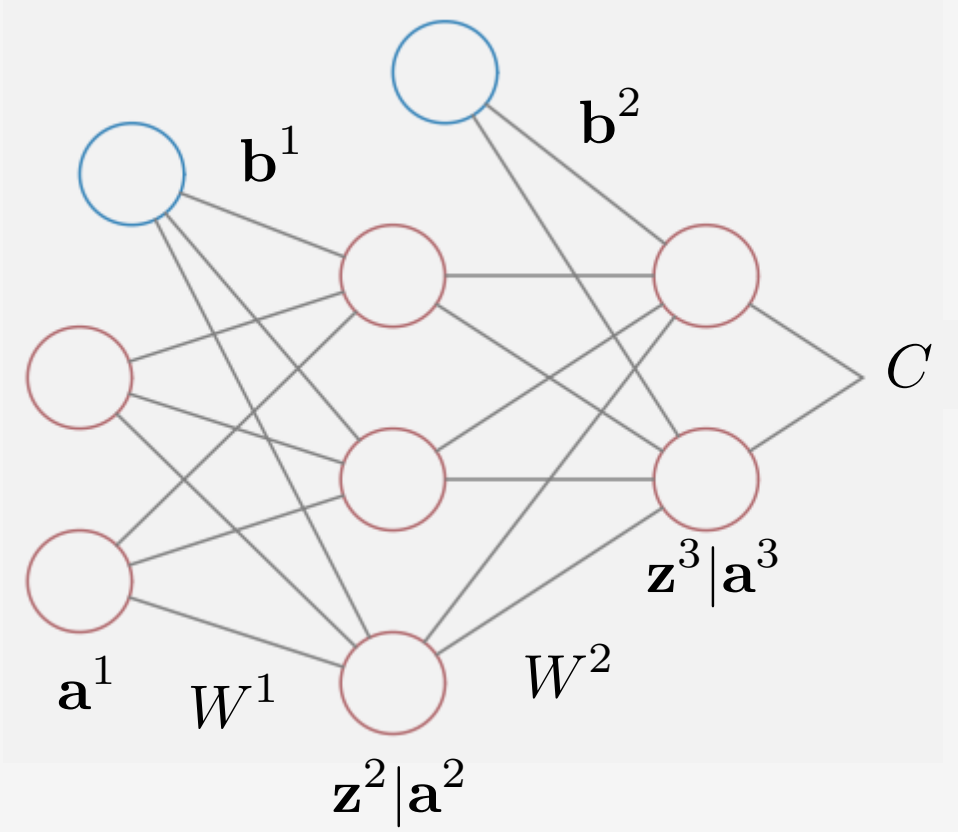
$$\frac{\partial f}{\partial u} = \frac{\partial f}{\partial r}\frac{\partial r}{\partial u} + \frac{\partial f}{\partial s}\frac{\partial s}{\partial u}$$

Crux: If you know the derivative with respect to an intermediate variable in the chain, can eliminate everything that lives in between.

This is the cornerstone of Back Propagation for efficiently computing derivatives

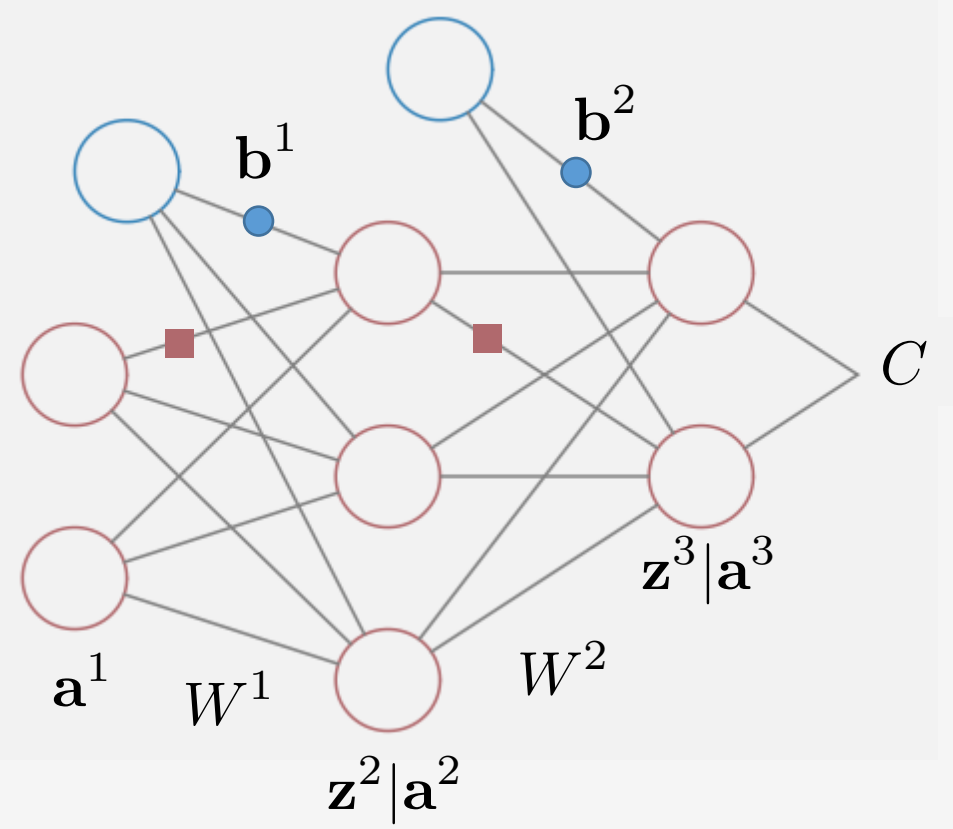
Learning Weights and Biases

For the derivation, we'll consider a very simple network



Learning Weights and Biases

For the derivation, we'll consider a very simple network

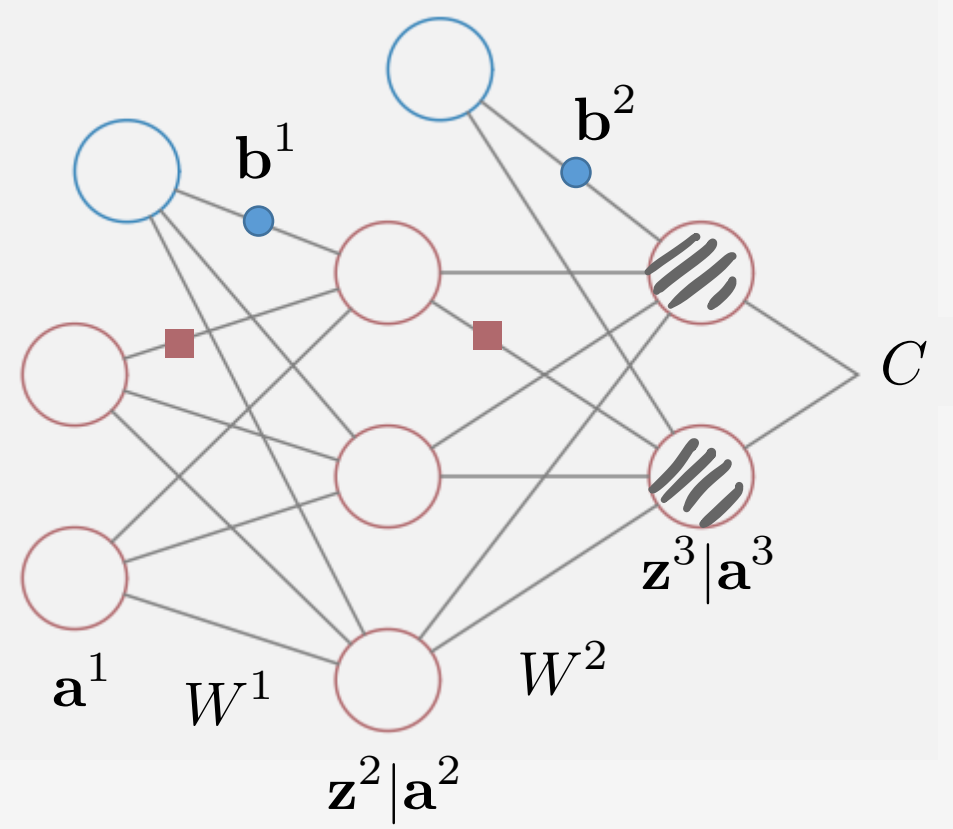


We want to use the Chain Rule to compute derivatives of the loss function w.r.t. weights and biases

$$\frac{\partial C}{\partial w_{ij}^{\ell}}, \quad \frac{\partial C}{\partial b_i^{\ell}} \qquad \text{for } \ell = 1, 2$$

Learning Weights and Biases

For the derivation, we'll consider a very simple network



We want to use the Chain Rule to compute derivatives of the loss function w.r.t. weights and biases

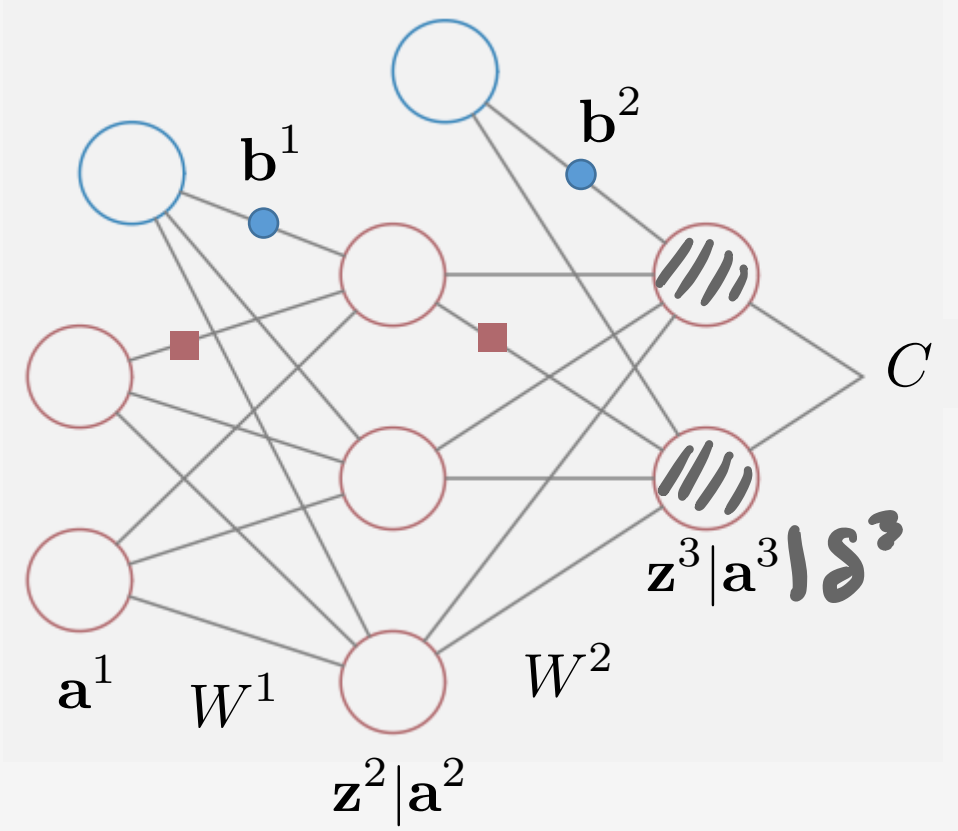
$$\frac{\partial C}{\partial w_{ij}^{\ell}}, \quad \frac{\partial C}{\partial b_i^{\ell}} \qquad \text{for } \ell = 1, 2$$

Need to choose an intermediate variable to that lives on the neurons, that we can easily compute derivatives with respect to

Could choose a's but we'll choose z's

Learning Weights and Biases

For the derivation, we'll consider a very simple network



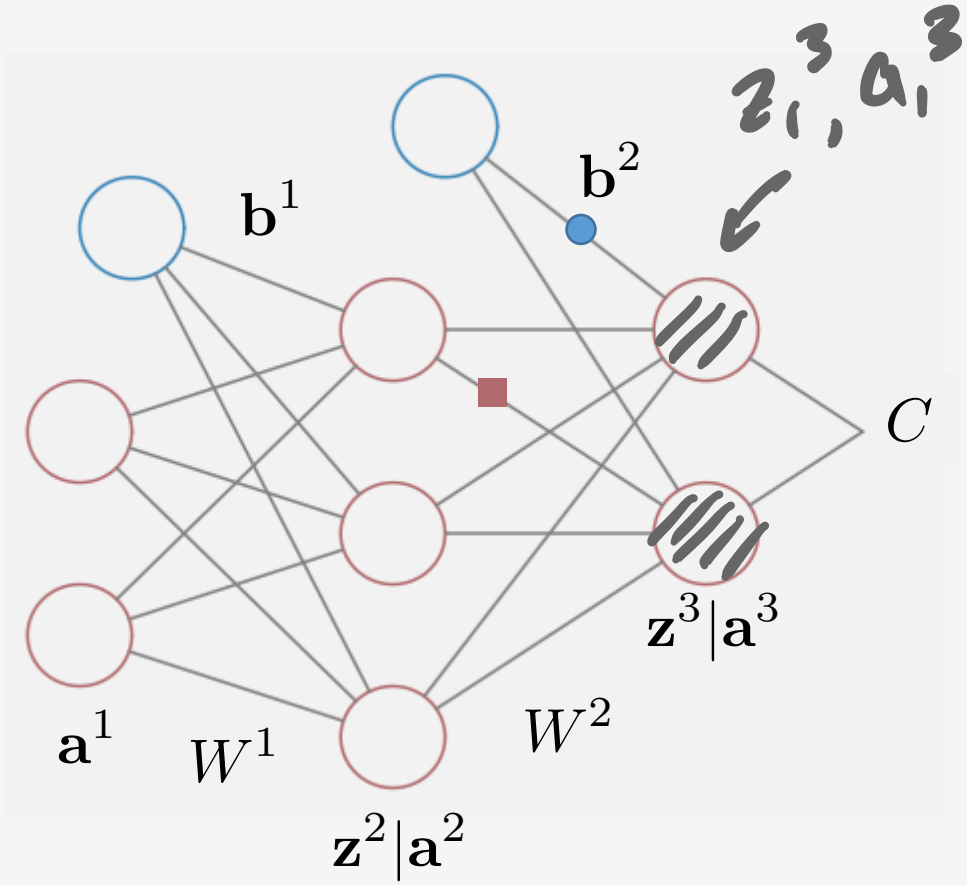
For easier notation, represent derivatives as

$$\delta_i^\ell = \frac{\partial C}{\partial z_i^\ell} \quad \text{(store them in } \delta^\ell \text{ vector)}$$

Note: Vector δ^ℓ lives on neurons

So δ^ℓ is same size as $\mathbf{z}^\ell$ and $\mathbf{a}^\ell$

Derivatives for Output Layer



Let's compute δ^3 for output layer $L = 3$

$$\delta_i^3 = \frac{\partial C}{\partial z_i^3} = (\text{using Chain Rule}) = \frac{\partial C}{\partial a_i^3}\frac{\partial a_i^3}{\partial z_i^3}$$

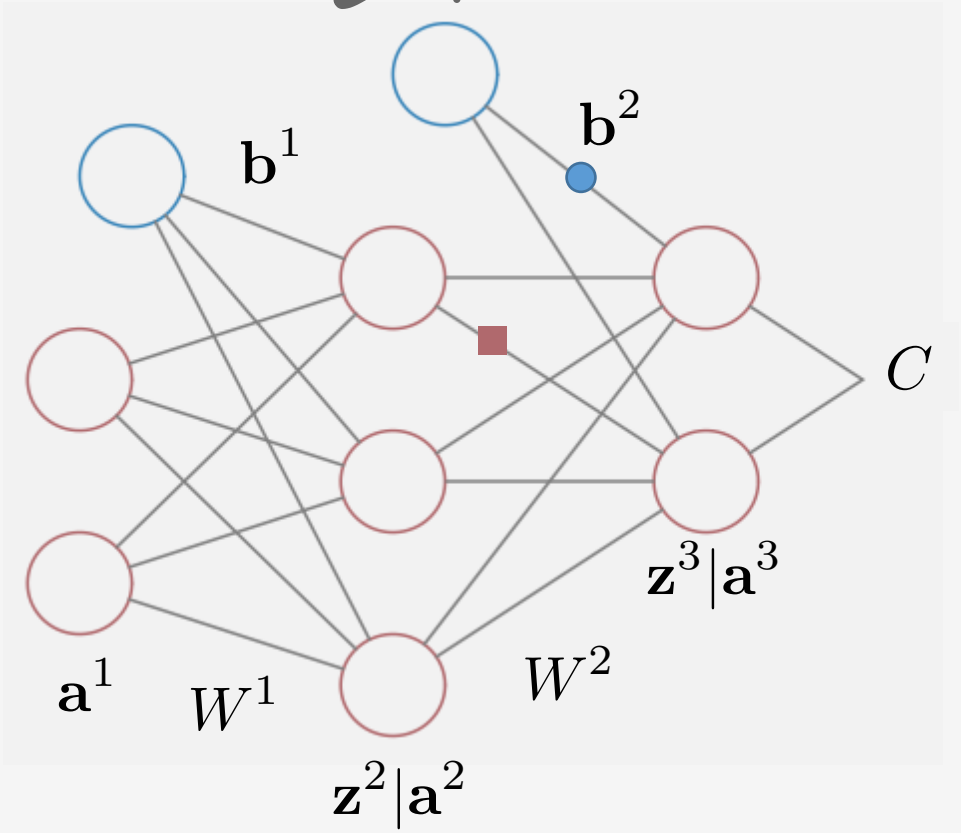
Let's find a nicer way to write this

$$a_i^3 = g(z_i^3)$$

$$\frac{\partial a_i^3}{\partial z_i^3} = g'(z_i^3)$$

Derivatives for Output Layer

$\delta_i^3 = \frac{\partial C}{\partial a_i^3} g'(z_i^3)$



Let's compute δ^3 for output layer $L = 3$

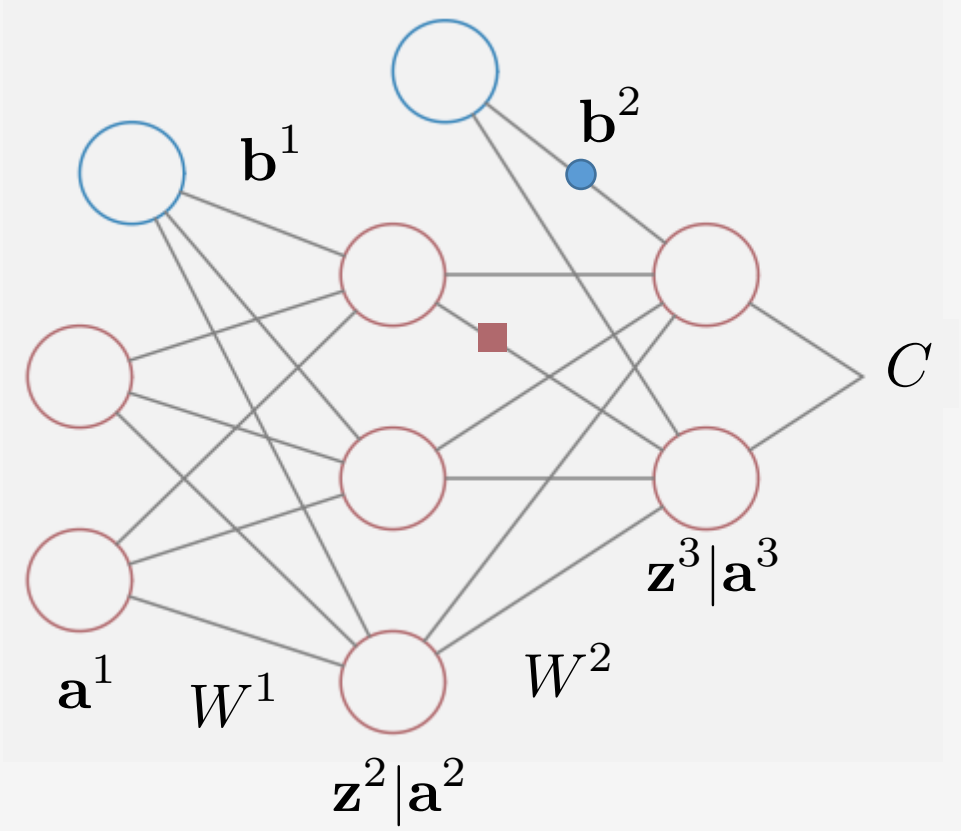
$$\delta_i^3 = \frac{\partial C}{\partial z_i^3} = (\text{using Chain Rule}) = \frac{\partial C}{\partial a_i^3}\frac{\partial a_i^3}{\partial z_i^3}$$

Let's find a nicer way to write this

We know activities and activations related by:

$$a_i^3 = g(z_i^3) \quad \Rightarrow \quad \frac{\partial a_i^3}{\partial z_i^3} = g'(z_i^3)$$

Derivatives for Output Layer



We've found δ^3 for output layer $L = 3$

$$\delta^3 = \nabla_{\mathbf{a}^3} C \odot g'(\mathbf{z}^3)$$

Important: Computing δ^3 requires knowing activities on output layer.

This means that before we can start Back Prop to get derivatives for SGD, we first need to do **Forward Prop** to compute activities and activations

Derivatives for Output Layer

Suppose we're using sigmoid neurons and the squared-loss function.

For a single training example, this loss function is:

$$C = \frac{1}{2}\|\mathbf{y} - \mathbf{a}^L\|^2 = \sum_{i=1}^{2}(y_i - a_i^L)^2$$

$y_i = 1 \Rightarrow \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

$y_i = 0 \Rightarrow \begin{bmatrix} 1 \\ 0 \end{bmatrix}$

Derivatives for Output Layer

Suppose we're using sigmoid neurons and the squared-loss function.

For a single training example, this loss function is:

$$C = \frac{1}{2}\|\mathbf{y} - \mathbf{a}^L\|^2 = \frac{1}{2}\sum_{i=1}^{2}(y_i - a_i^3)^2$$

The derivative of C w.r.t. a single output activation is

$$\frac{\partial C}{\partial a_i^3} = \frac{1}{2}2(y_i - a_i^3)(-1) = (a_i^3 - y_i) \Rightarrow \begin{bmatrix} a_1^3 - y_1 \\ a_2^3 - y_2 \end{bmatrix}$$

Derivatives for Output Layer

Suppose we're using sigmoid neurons and the squared-loss function.

For a single training example, this loss function is:

$$C = \frac{1}{2}\|\mathbf{y} - \mathbf{a}^L\|^2 = \sum_{i=1}^{2}(y_i - a_i^L)^2$$

The derivative of C w.r.t. a single output activation is

$$\frac{\partial C}{\partial a_i^3} = 2 \cdot \frac{1}{2}(y_i - a_i^3) \cdot (-1) = (a_i^3 - y_i)$$

$2 \cdot \frac{1}{2}(y_i - a_i^3) \cdot \frac{d}{da_i^3}(y_i - a_i^3$

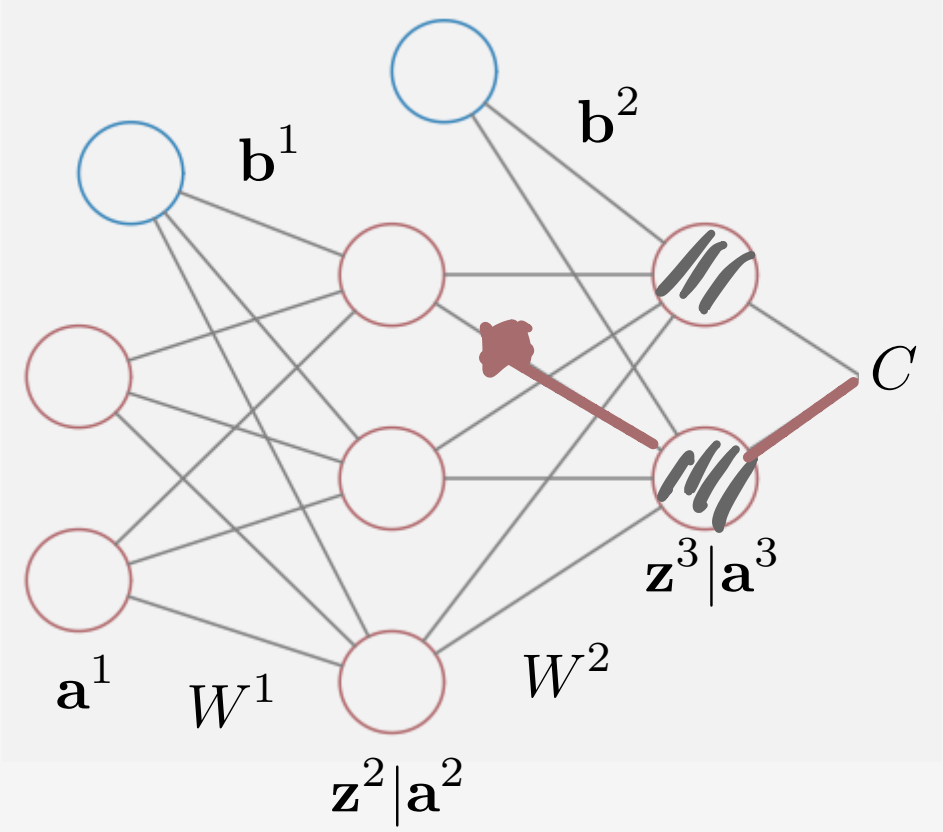
Since we're using sigmoid activations, we have $g'(z_i^3) = g(z_i^3)(1 - g(z_i^3))$

Finally, the vectorized version is $\delta^3 = (\mathbf{a}^3 - \mathbf{y}) \odot g(\mathbf{z}^3) \odot (1 - g(\mathbf{z}^3))$

Derivatives for Output Layer

OK **Great**! Now we can easily-ish compute δ^3 for the output layer

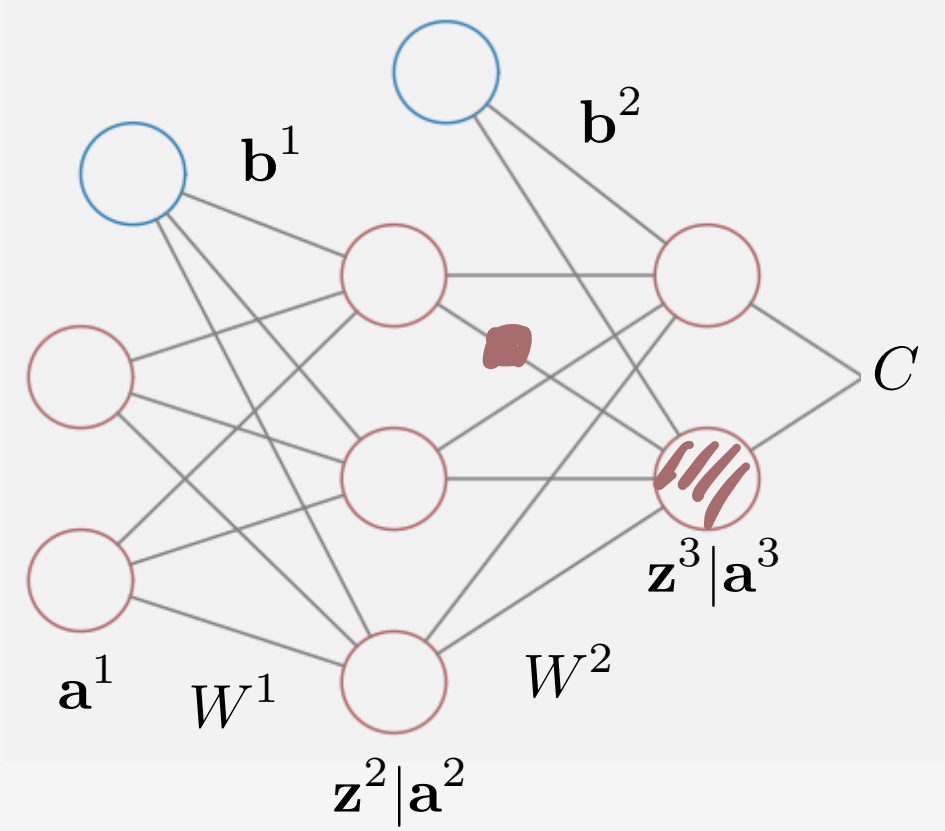
But really our goal is to take derivatives w.r.t. to the weights and biases



Note that every weight connected to an output neuron depends on a single δ_i^3

Derivatives for Output Layer

Note that every weight connected to an output neuron depends on a single δ_i^3

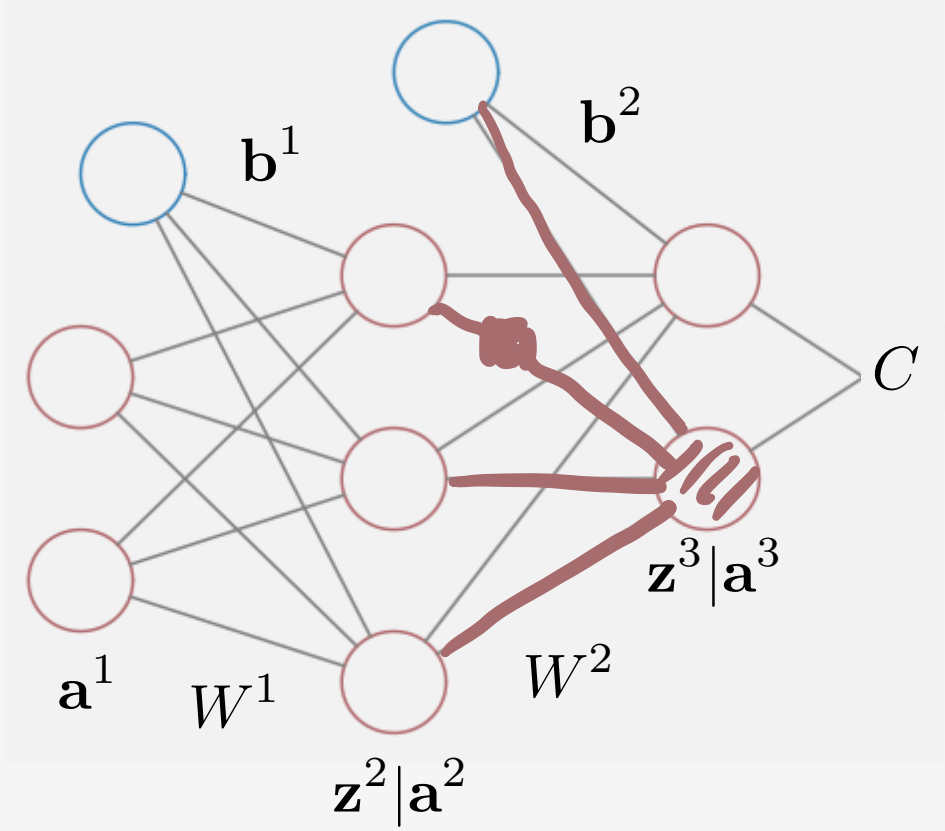


Example: Compute $\frac{\partial C}{\partial w_{21}^2} = \frac{\partial C}{\partial z_2^3} \frac{\partial z_2^3}{\partial w_{21}^2}$

$= \delta_2^3 \frac{\partial z_2^3}{\partial w_{21}^2}$

Derivatives for Output Layer

Note that every weight connected to an output neuron depends on a single δ_i^3



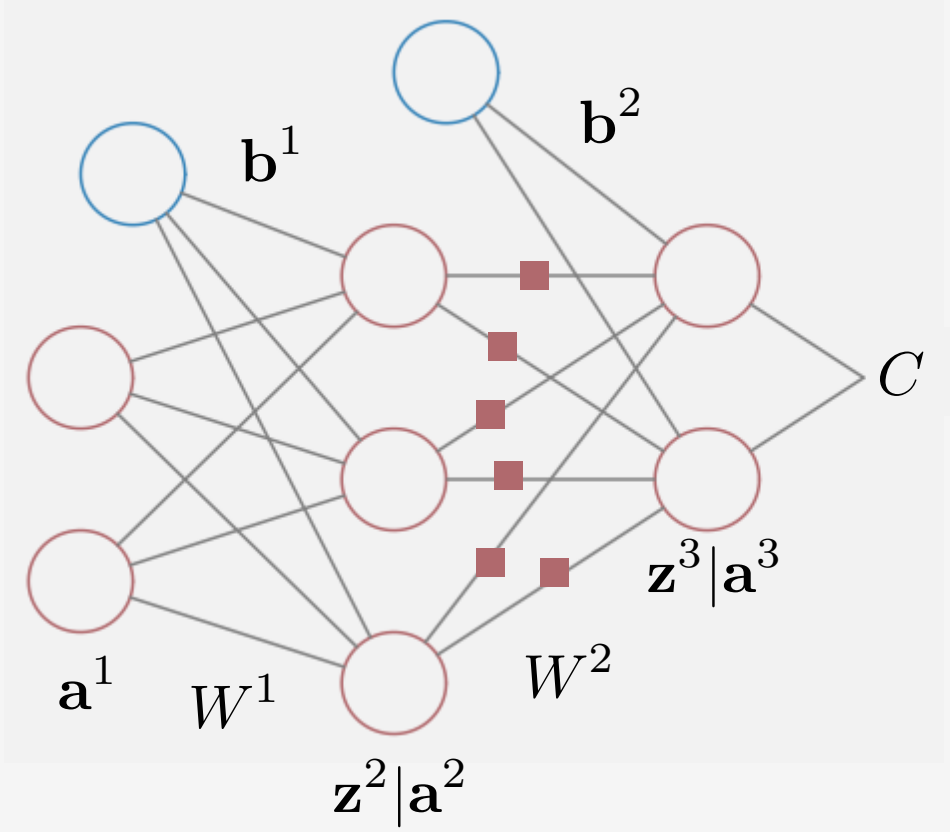
Example: Compute $\dfrac{\partial C}{\partial w_{21}^2} = \dfrac{\partial C}{\partial z_2^2}\dfrac{\partial z_2^2}{\partial w_{21}^2} = \delta_2^3 \dfrac{\partial z_2^2}{\partial w_{21}^2}$

Recall: $\mathbf{z}^3 = W^2\mathbf{a}^2 + \mathbf{b}^2$ and z_2^3 is given by

$$z_2^3 = \mathbf{w_{21}^2} a_1^2 + w_{22}^2 a_2^2 + w_{23}^2 a_3^2 + b_2^2$$

So... $\dfrac{\partial C}{\partial w_{21}^2} = \delta_2^3 \dfrac{\partial z_2^3}{\partial w_{21}^2} =$ $\delta_2^3 a_1^2$

Derivatives for Output Layer



For a general derivative of L w.r.t. w_{ij}^2 we have

$$\frac{\partial C}{\partial w_{ij}^2} = \delta_i^3 a_j^2 \quad \text{for } i = 1, 2 \text{ and } j = 1, 2, 3$$

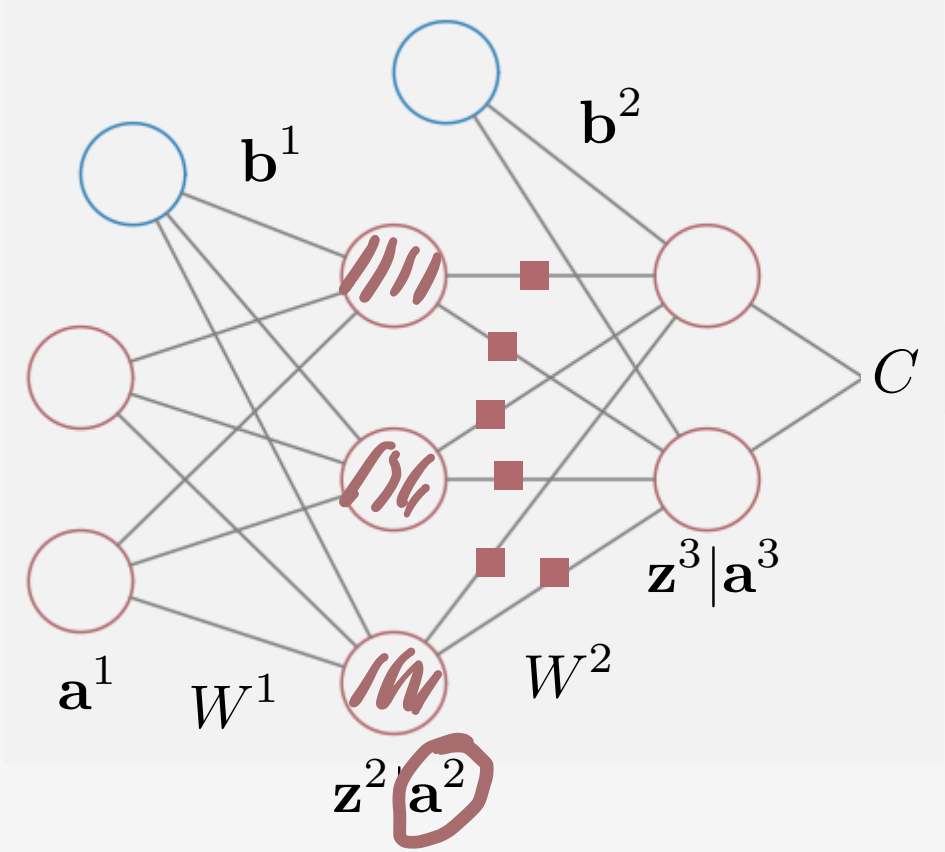
Let's make the notation a bit more practical

We store weights in the matrix

$$W^2 = \begin{bmatrix} w_{11}^2 & w_{12}^2 & w_{13}^2 \\ w_{21}^2 & w_{22}^2 & w_{23}^2 \end{bmatrix}$$

Store derivatives in a matrix too!

Derivatives for Output Layer



For a general derivative of L w.r.t. w_{ij}^2 we have

$$\frac{\partial C}{\partial w_{ij}^2} = \delta_i^3 a_j^2 \quad \text{for } i = 1, 2 \text{ and } j = 1, 2, 3$$

Store the weight derivatives in a matrix

$$\frac{\partial C}{\partial W^2} = \begin{bmatrix} \frac{\partial C}{\partial w_{11}^2} & \frac{\partial C}{\partial w_{12}^2} & \frac{\partial C}{\partial w_{13}^2} \\ \frac{\partial C}{\partial w_{21}^2} & \frac{\partial C}{\partial w_{22}^2} & \frac{\partial C}{\partial w_{23}^2} \end{bmatrix} = \begin{bmatrix} \delta_1^3 a_1^2 & \delta_1^3 a_2^2 & \delta_1^3 a_3^2 \\ \delta_2^3 a_1^2 & \delta_2^3 a_2^2 & \delta_2^3 a_3^2 \end{bmatrix}$$

So, vectorizing, we have $\frac{\partial C}{\partial W^2} = \delta^3 (a^2)^T$

Derivatives for Output Layer

So after forward propping a training example we have all the $\mathbf{z}^\ell$s and $\mathbf{a}^\ell$s

The derivatives of the loss function w.r.t. weights connected to the output layer are

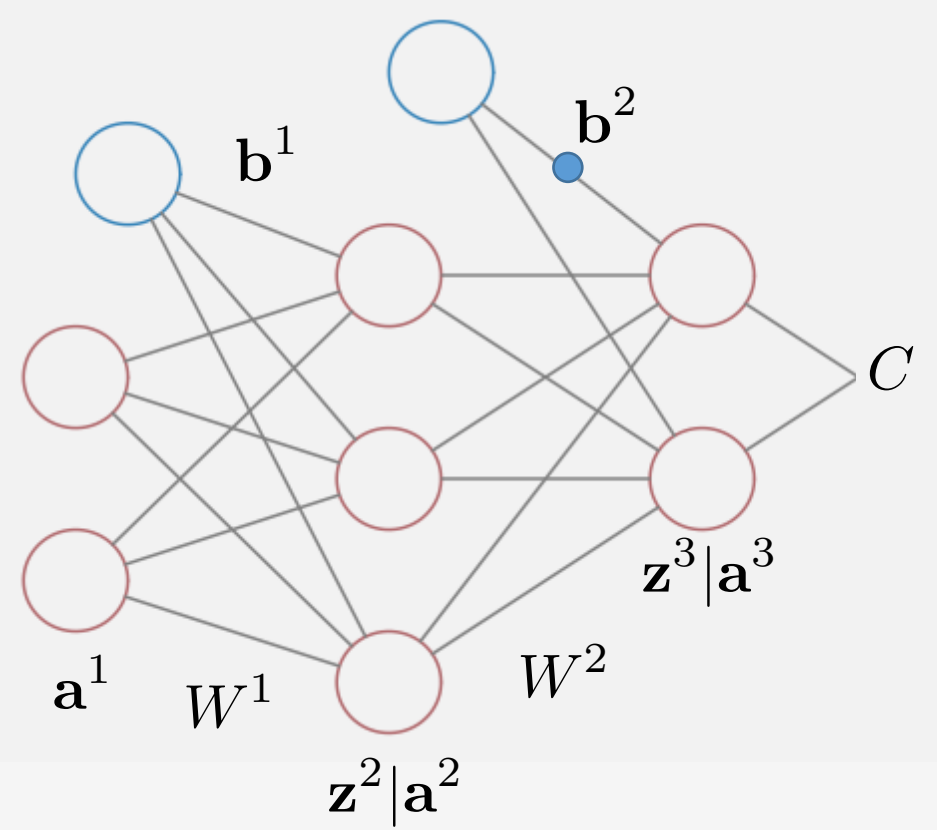
$$\frac{\partial C}{\partial W^2} = \delta^3 (\mathbf{a}^2)^T$$

Notice that if we hardcode the derivative of L w.r.t. the activities and the derivative of the activation function, then these weight derivatives can be computed with entirely vectorized operations!

OK, cool. Them's the weights, but what about the biases?

Derivatives for Output Layer

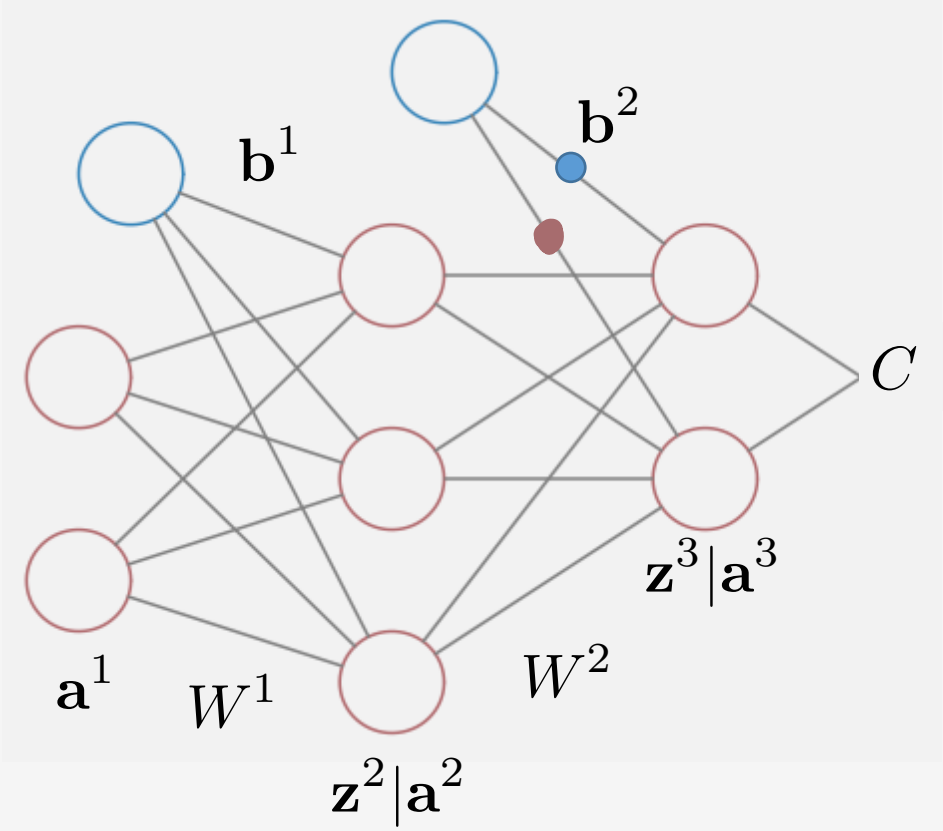
Note that every **bias** connected to an output neuron depends on a single δ_i^3



Example: Compute $\dfrac{\partial C}{\partial b_1^2}$

Derivatives for Output Layer

Note that every **bias** connected to an output neuron depends on a single δ_i^3



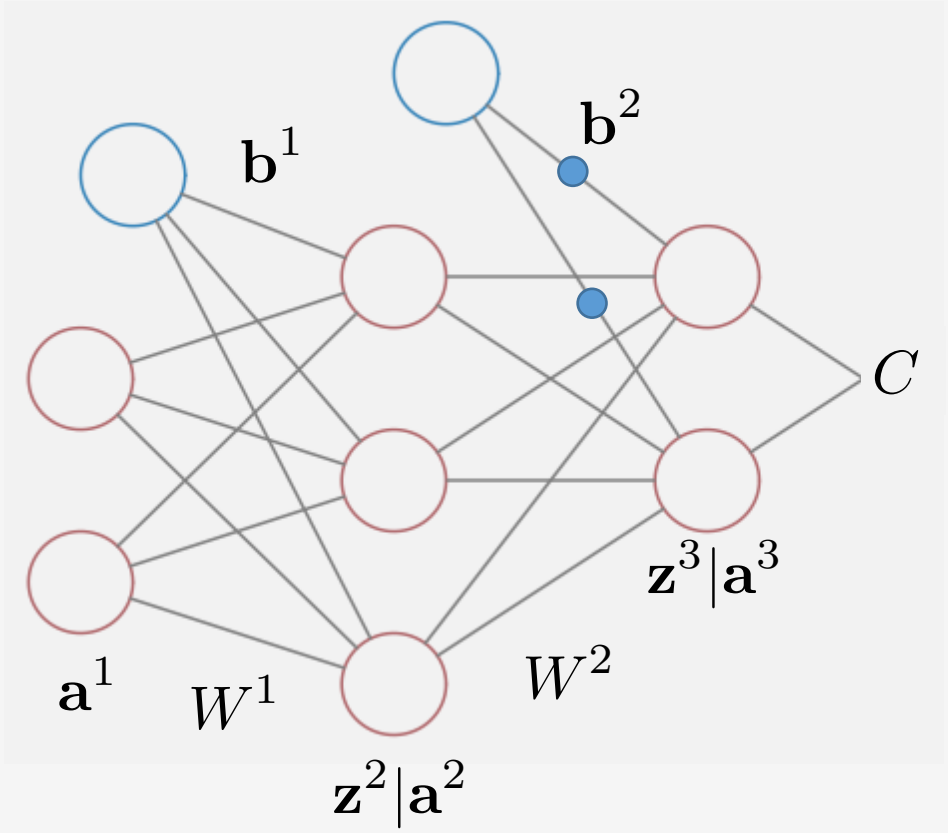
Example: Compute $\dfrac{\partial C}{\partial b_1^2} = \dfrac{\partial C}{\partial z_1^3}\dfrac{\partial z_1^3}{\partial b_1^2}$

Recall: $\mathbf{z}^3 = W^2\mathbf{a}^2 + \mathbf{b}^2$ and z_1^3 is given by

$$z_1^3 = w_{11}^2 a_1^2 + w_{12}^2 a_2^2 + w_{13}^2 a_3^2 + \mathbf{b_1^2}$$

So... $\dfrac{\partial C}{\partial b_1^2} = \dfrac{\partial C}{\partial z_1^3}\dfrac{\partial z_1^3}{\partial b_1^2} = \delta_1^3 \cdot 1 = \delta_1^3$

Derivatives for Output Layer



For a general derivative of L w.r.t. b_i^2 we have

$$\frac{\partial C}{\partial b_i^2} = \delta_i^3 \quad \text{for } i = 1, 2$$

Store the bias derivatives in a vector

$$\frac{\partial C}{\partial \mathbf{b}^2} = \begin{bmatrix} \frac{\partial C}{\partial b_1^2} \\ \\ \frac{\partial C}{\partial b_2^2} \end{bmatrix} = \begin{bmatrix} \delta_1^3 \\ \delta_2^3 \end{bmatrix} = \delta^3$$

Derivatives for Output Layer

Intermediate Summary:

For a given training example $\mathbf{x}_k$, perform forward substitution to get activities $\mathbf{z}^\ell$ and activations $\mathbf{a}^\ell$ on each layer.

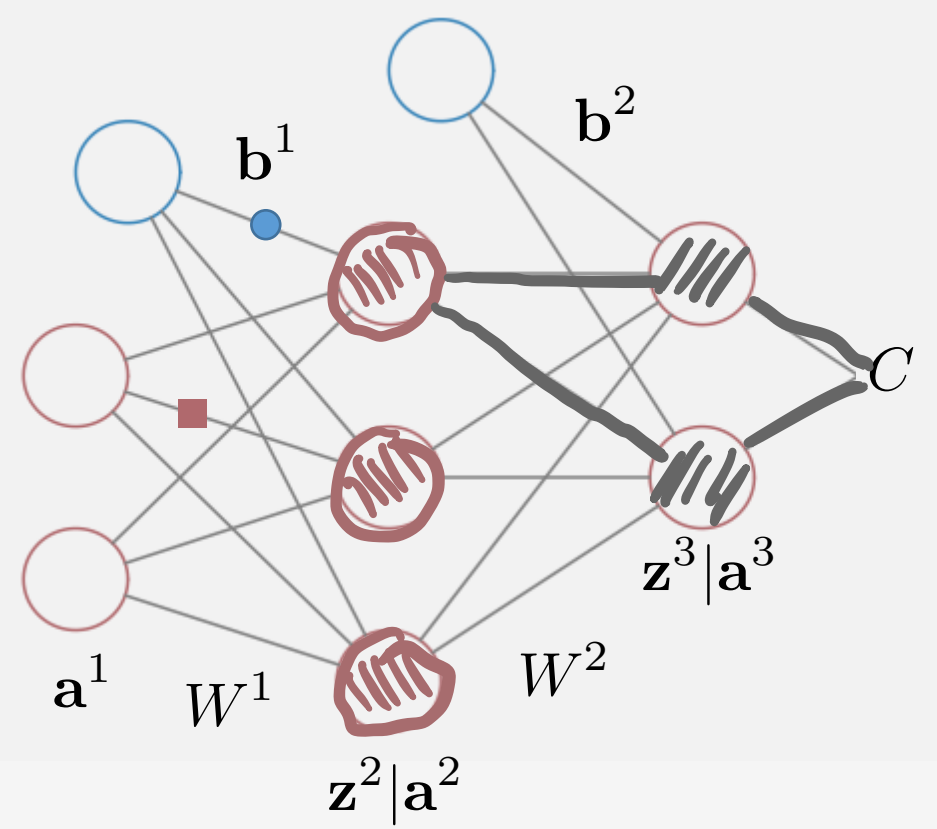
Then, to get partial derivatives w.r.t. weights and biases connected to output layer

1. Compute $\delta^L = \nabla_{\mathbf{a}^L} C \odot g'(\mathbf{z}^L)$
2. Compute $\dfrac{\partial C}{\partial W^{L-1}} = \delta^L (\mathbf{a}^{L-1})^T$ and $\dfrac{\partial C}{\partial \mathbf{b}^{L-1}} = \delta^L$

OK, that wasn't so bad! We found simple, vectorized formulas for derivatives connected to the output layer.

Problem: How do we get derivatives w.r.t. weights and biases earlier in network?

Back Propagation



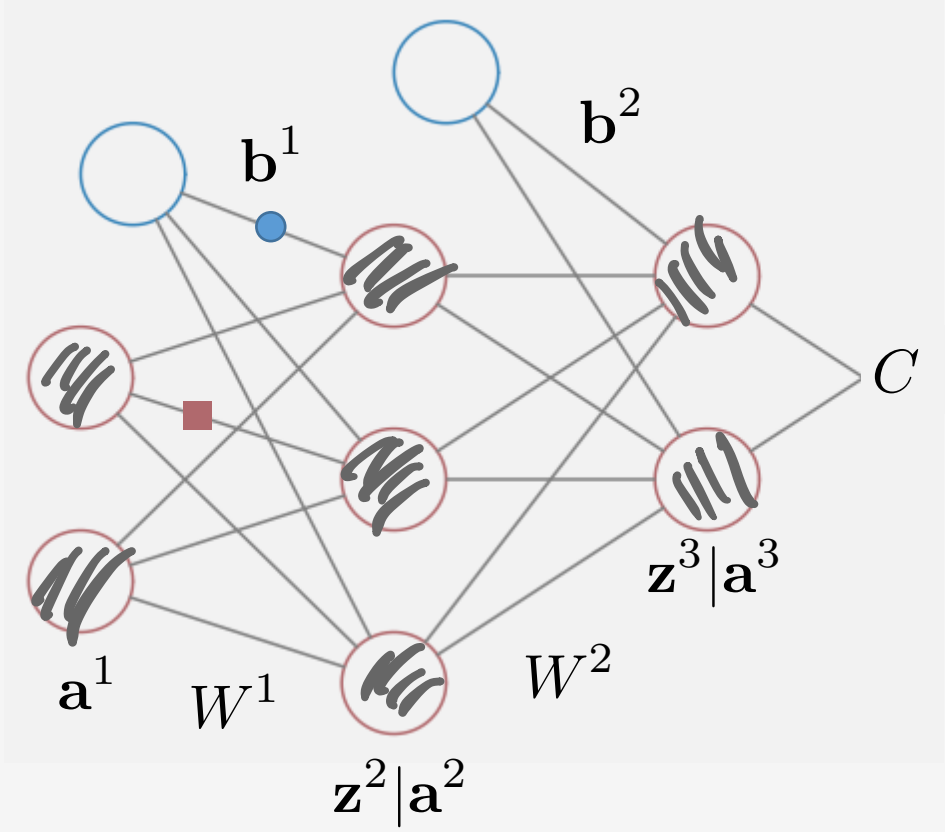
Formulas for W^2 and $\mathbf{b}^2$ were nice b/c we knew δ^3

Seems like formulas for W^1 and $\mathbf{b}^1$ would be nice if we knew δ^2

But relationship b/w C and $\mathbf{z}^2$ is super complicated

Multiple passes through activation functions :(

Back Propagation



Formulas for W^2 and $\mathbf{b}^2$ were nice b/c we knew δ^3

Seems like formulas for W^1 and $\mathbf{b}^1$ would be nice if we knew δ^2

But relationship b/w C and $\mathbf{z}^2$ is super complicated

Multiple passes through activation functions :(

But it's OK! There's a nice-ish process for **propagating** $\delta's$ **backwards** through layers!