Convolutional Neural Networks

A. M. Sadeghzadeh, Ph.D.

Sharif University of Technology
Computer Engineering Department (CE)
Data and Network Security Lab (DNSL)

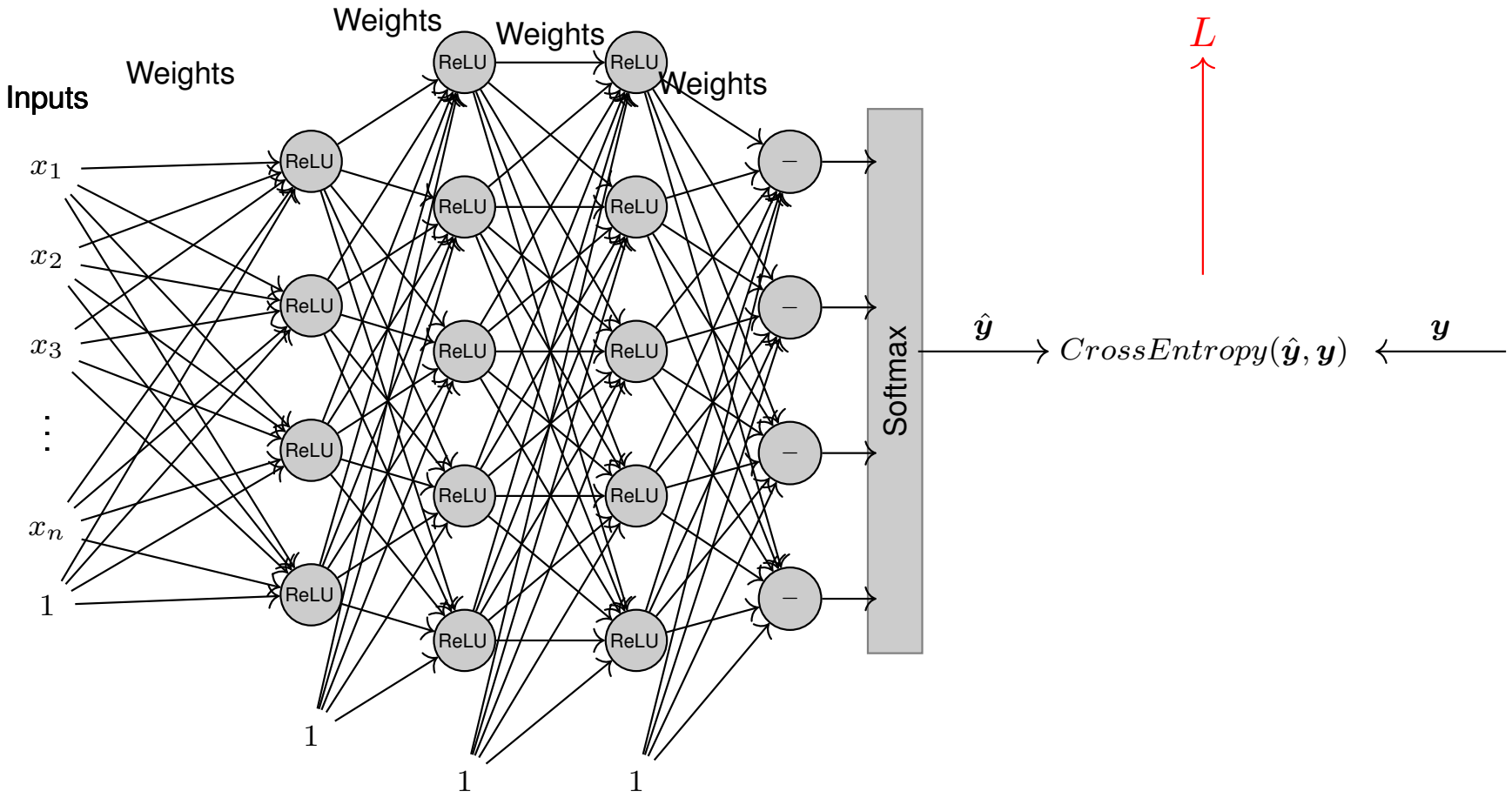
March 4, 2023

Most slides have been adapted from Bhiksha Raj, 11-785, CMU 2020, Justin Johnson, EECS 498-007, University of Michigan 2020, and Fei Fei Li, cs231n, Stanford 2017

Today's agenda

Recap

Loss function



$$\hat{y} = Softmax(W^4(max(0, W^3(max(0, W^2(max(0, W^1 x + b^1)) + b^2)) + b^3)) + b^4)$$

Gradient descent

- The gradient descent method to find the minimum of a function iteratively

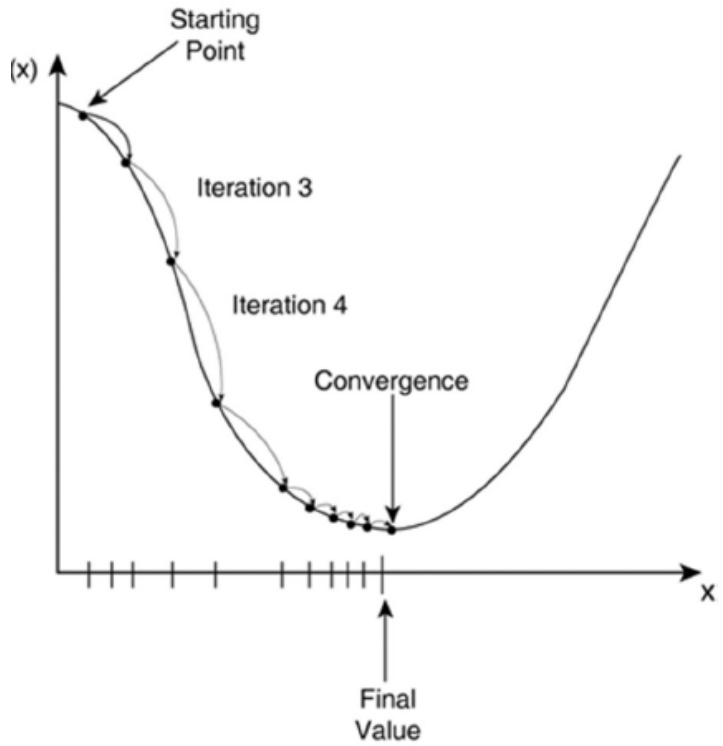
$$\boldsymbol{x}^{t+1} = \boldsymbol{x}^t - \eta \nabla_{\boldsymbol{x}} f(\boldsymbol{x}^t)$$

  - η is the "step size" (Also called "learning rate")

- The gradient descent algorithm converges when the following criterion is satisfied.

$$|f(\boldsymbol{x}^{t+n}) - f(\boldsymbol{x}^t)| < \epsilon$$

  - n is a hyperparameter.



Training neural nets through gradient descent

- Total training loss

$$L(f(X, W), \boldsymbol{y}) = \frac{1}{N} \sum_i L_i(f(\boldsymbol{x}^{(i)}, W), y^{(i)})$$

- Gradient descent algorithm:
  - Initialize all weights and biases $w_{ij}^{(\ell)}$
    - Using the extended notation: the bias is also a weight
  - Do:
    - For every layer ℓ for all i, j update:

$$w_{ij}^{(\ell)} = w_{ij}^{(\ell)} - \eta \frac{\partial L}{\partial w_{ij}^{(\ell)}}$$

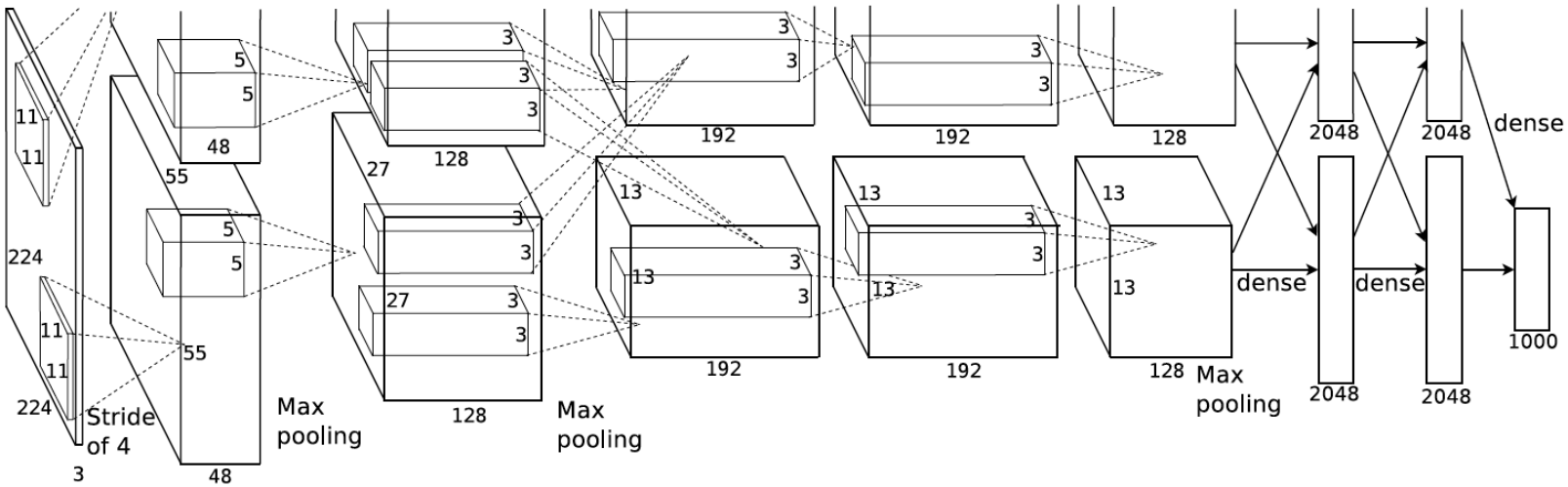
  - Until loss has converged

Idea: derive $\frac{\partial L}{\partial w_{ij}^{\ell}}$ on paper

- Problems
  - Very tedious: Lots of matrix calculus, need lots of paper
  - What if we want to change loss? Need to re-derive from scratch. Not modular!
  - Not feasible for very complex models!

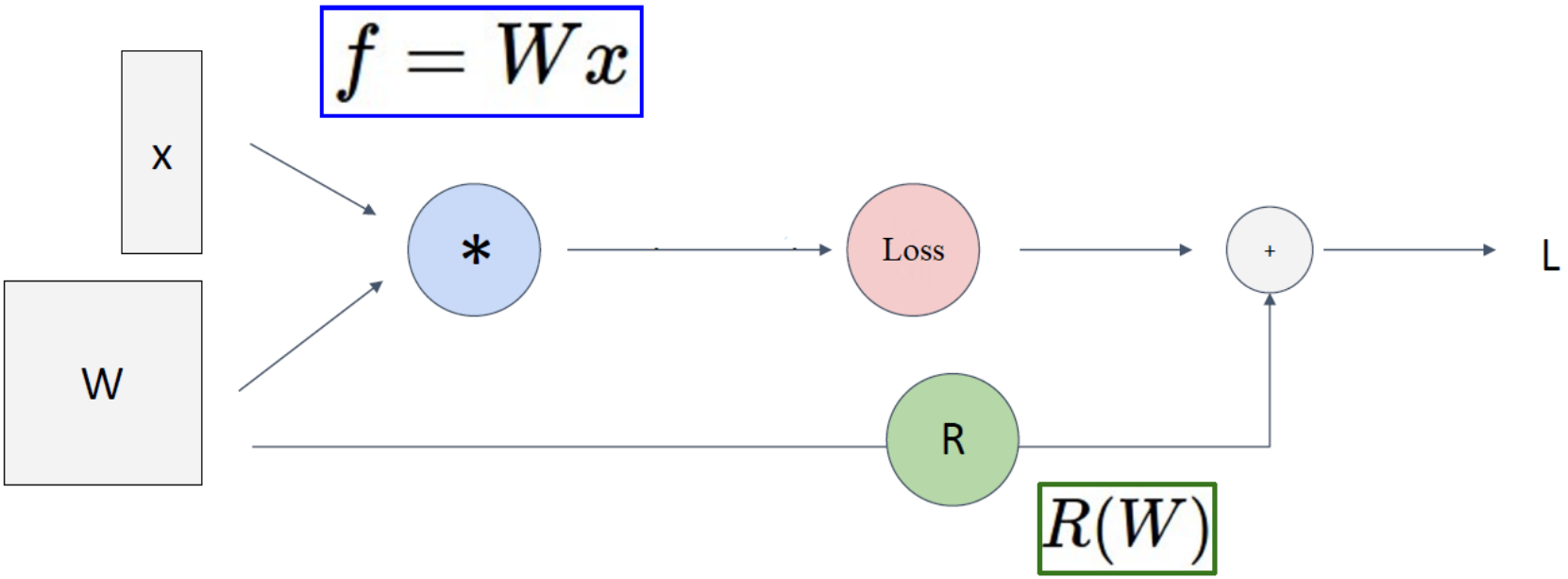
Idea: derive $\frac{\partial L}{\partial w_{ij}^{\ell}}$ on paper

- Problems
  - Very tedious: Lots of matrix calculus, need lots of paper
  - What if we want to change loss? Need to re-derive from scratch. Not modular!
  - Not feasible for very complex models!

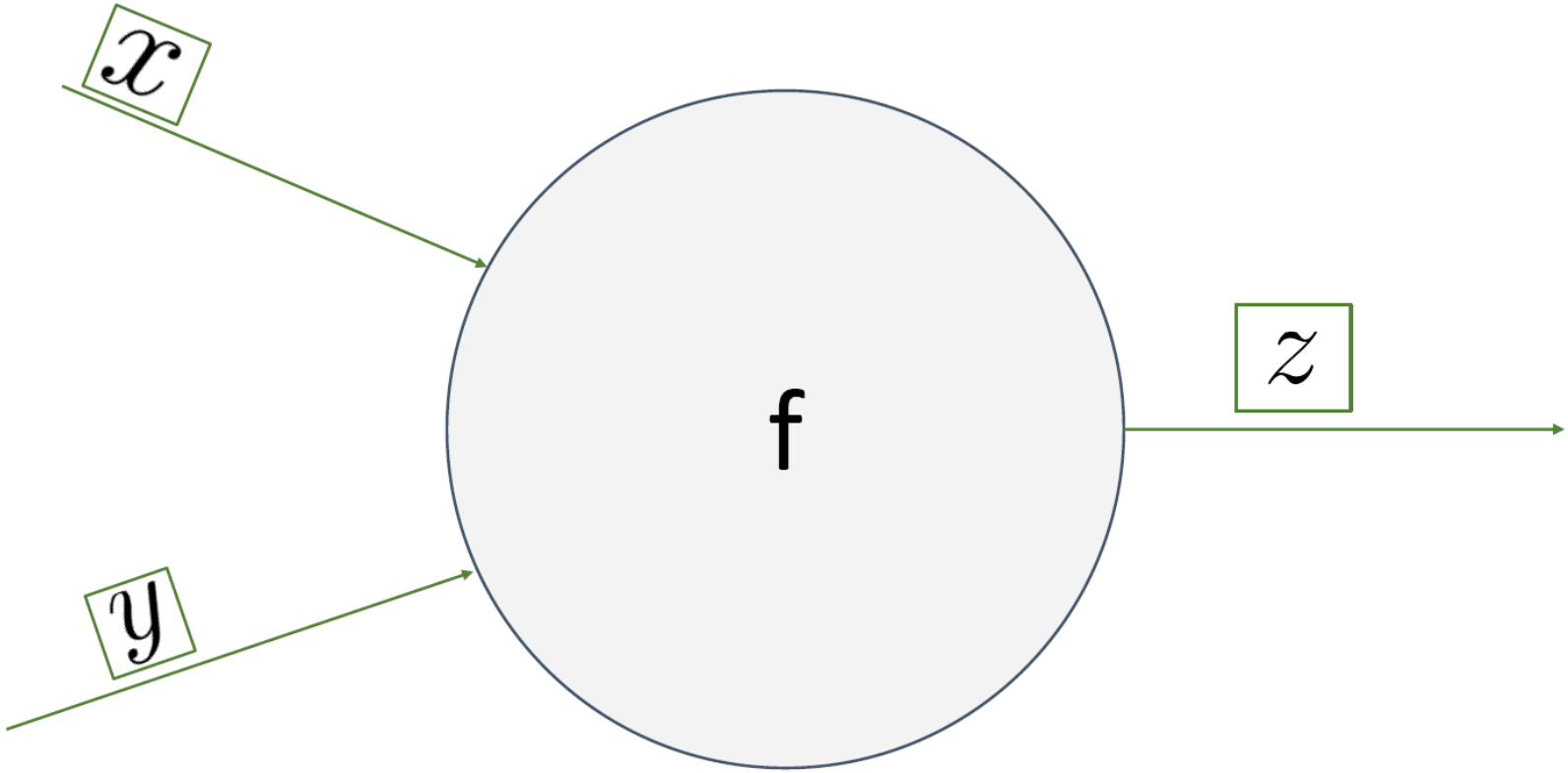


AlexNet has about 660K units, 61M parameters,

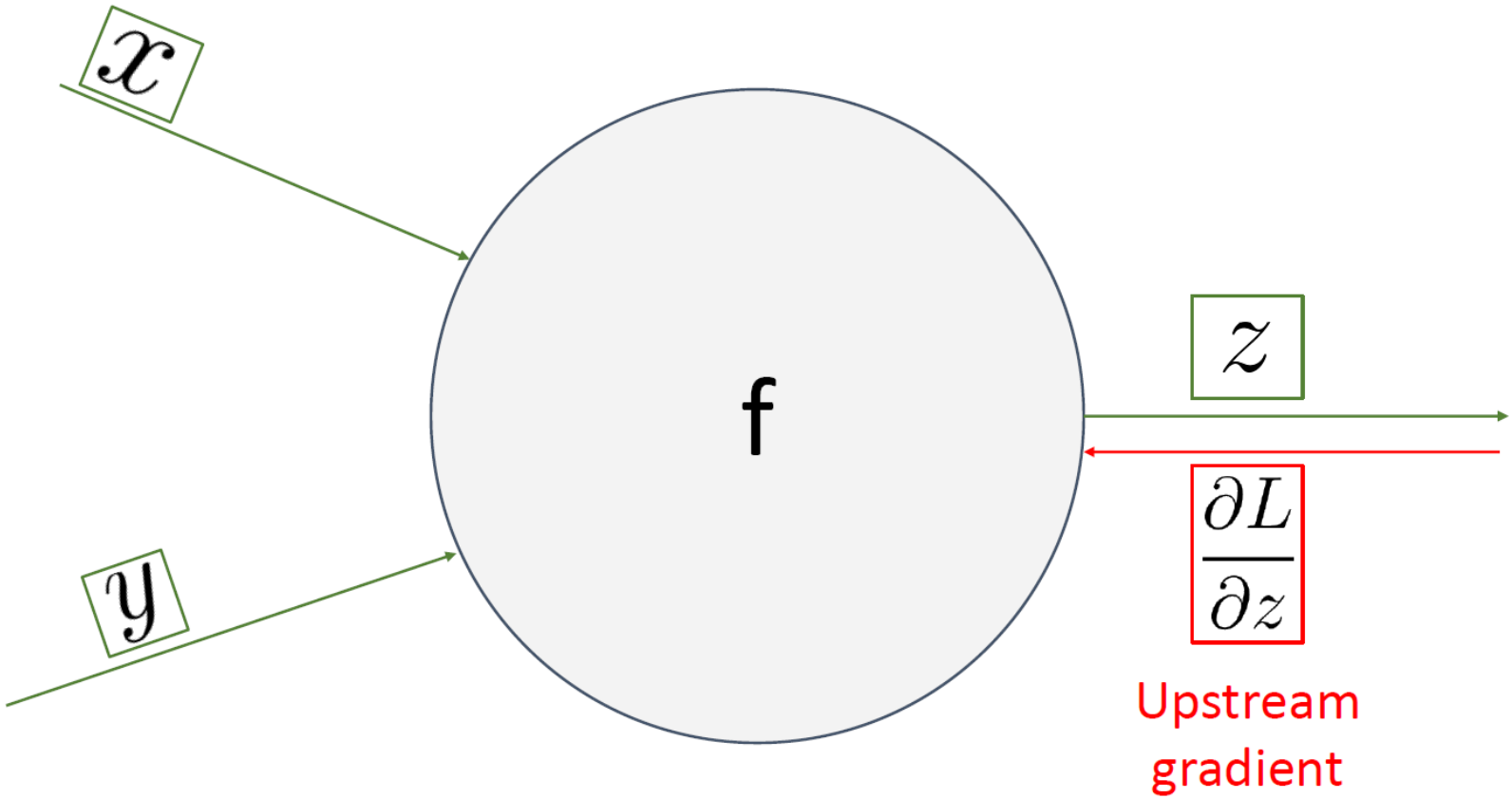
Better Idea: Computational Graphs



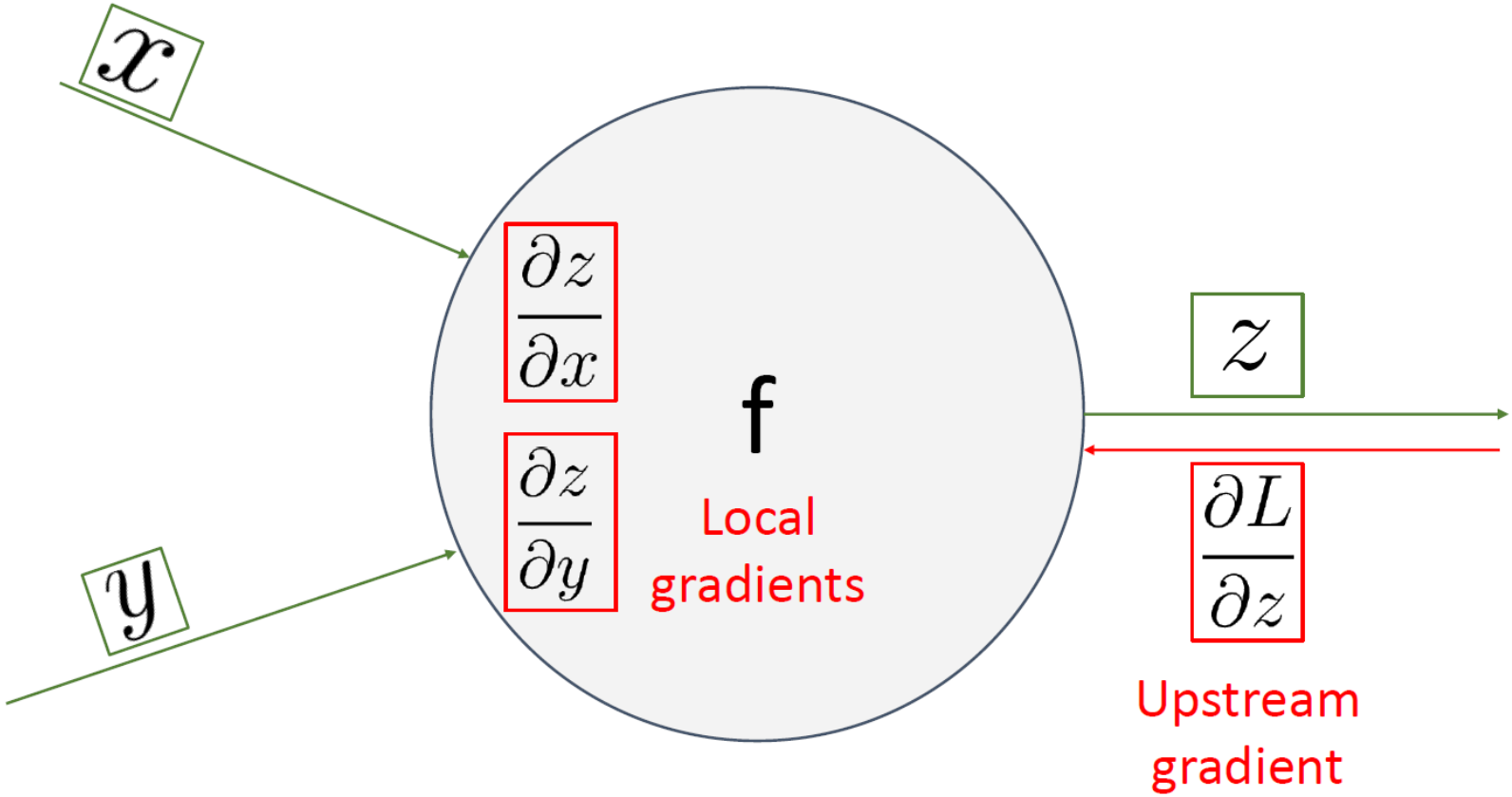
Up/Down stream gradients



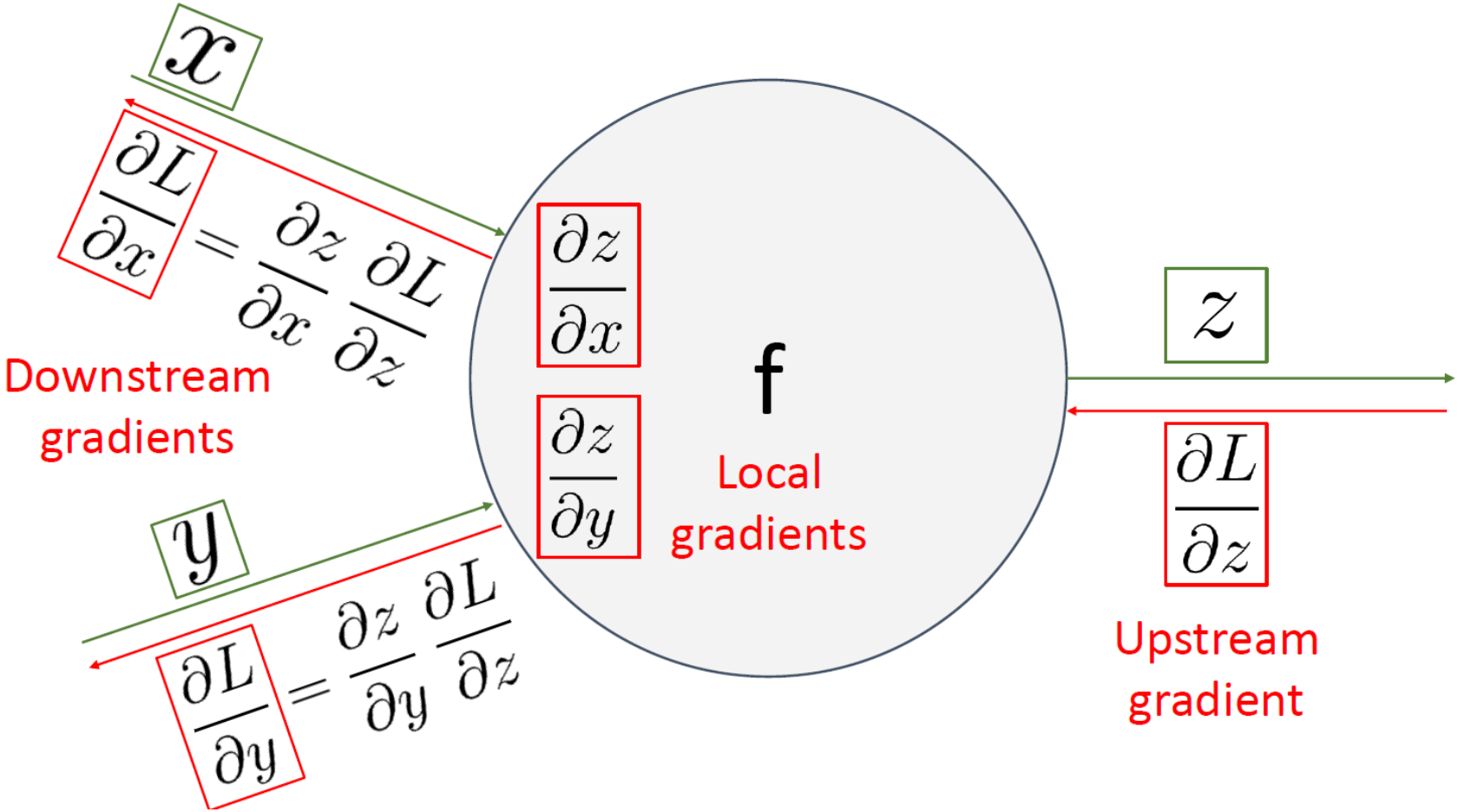
Up/Down stream gradients



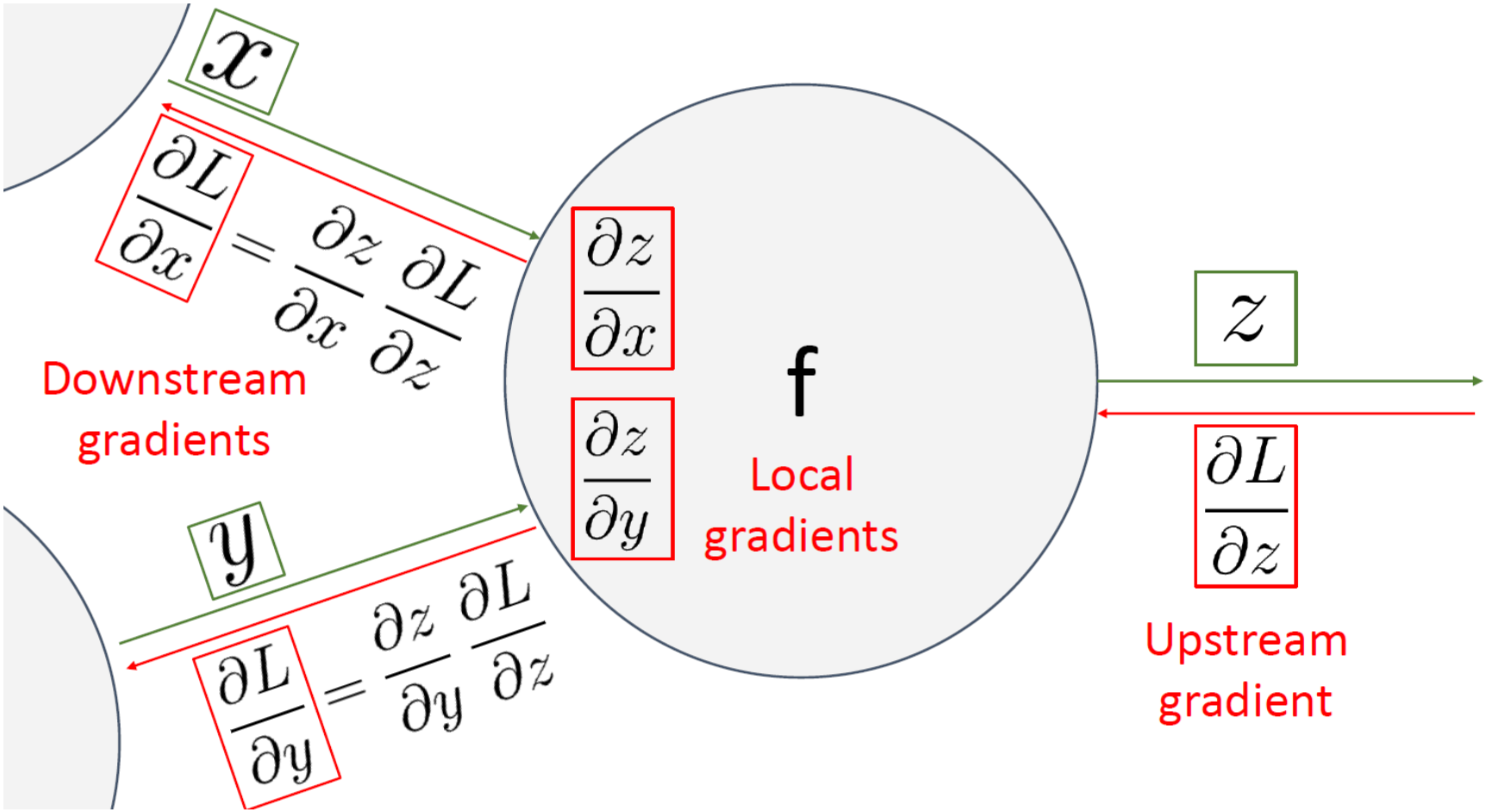
Up/Down stream gradients



Up/Down stream gradients

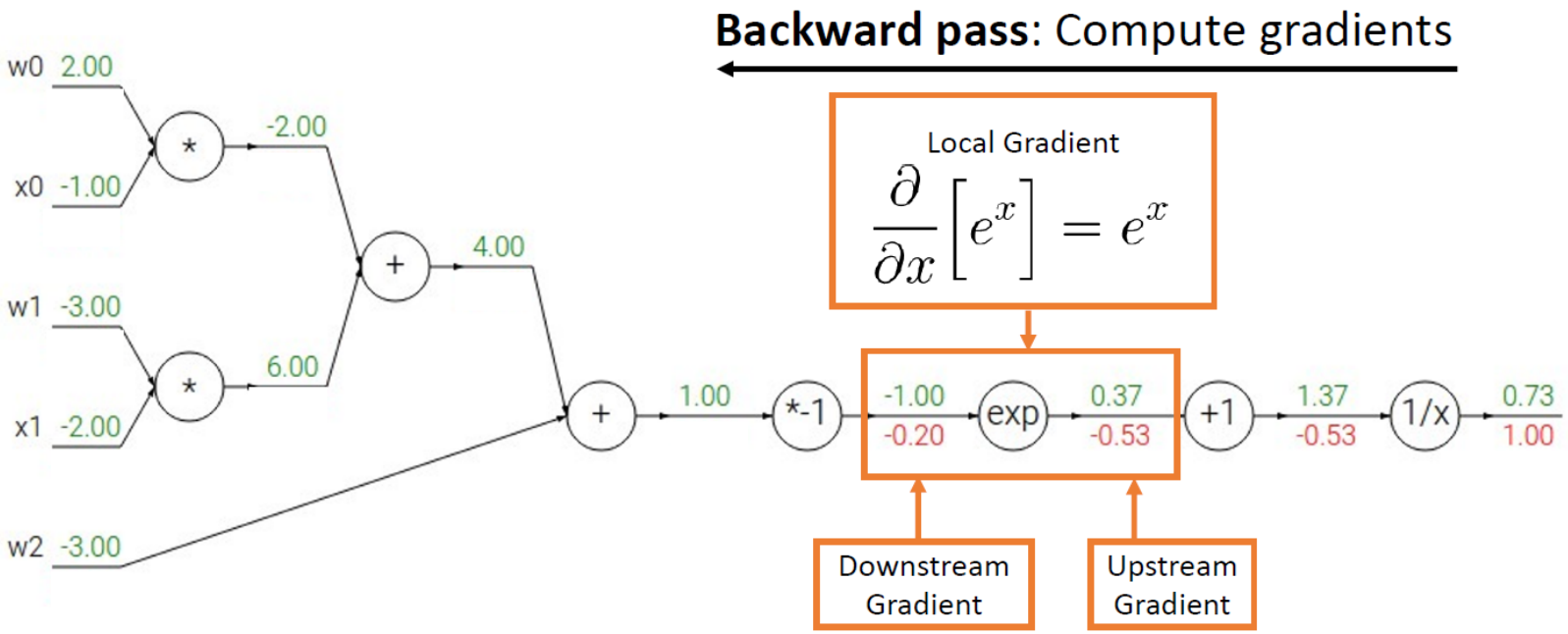


Up/Down stream gradients



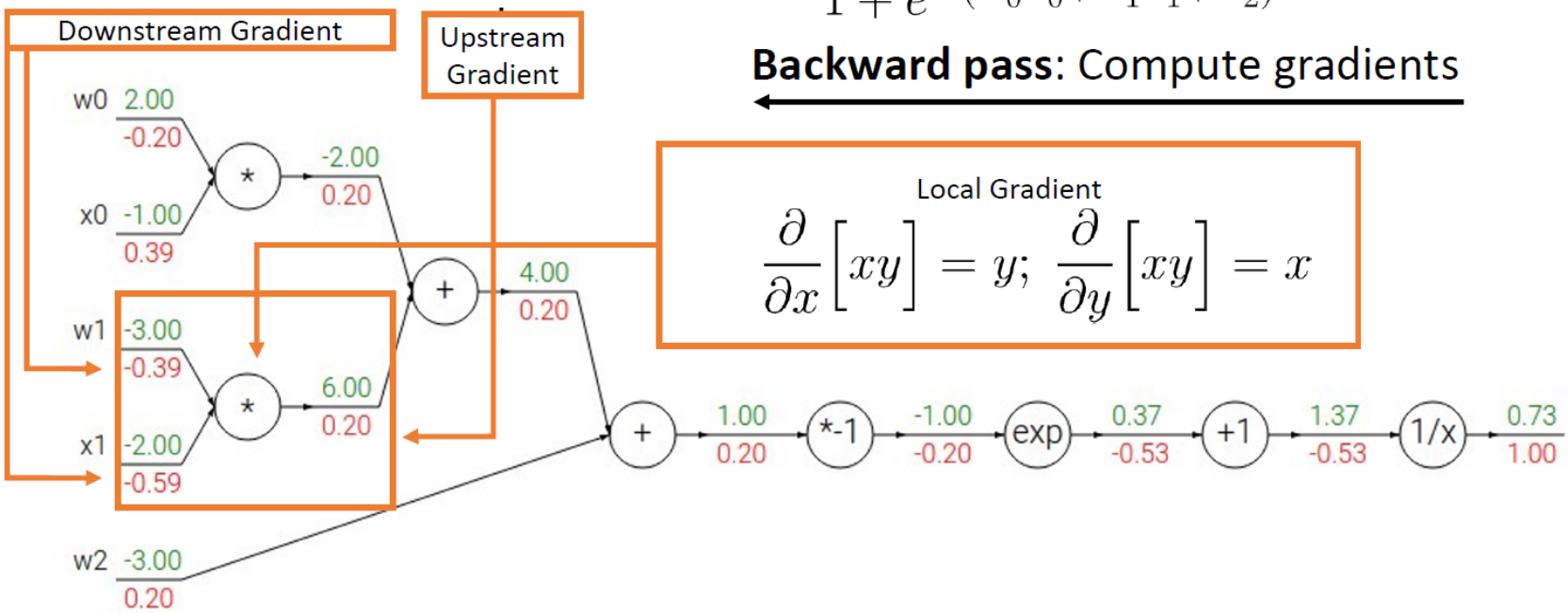
Example

$$f(x,w) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$



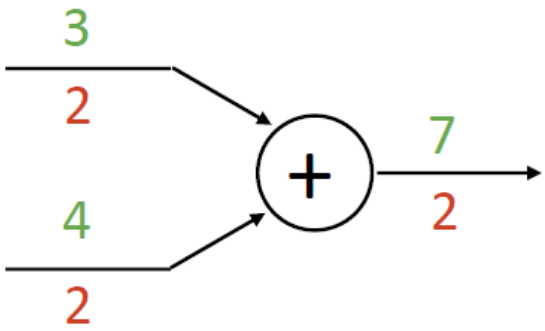
Example

$$f(x,w) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$

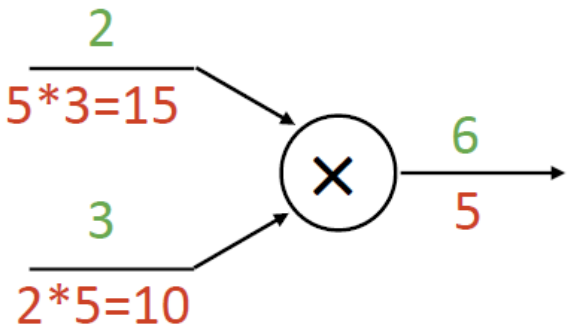


Patterns in Gradient Flow

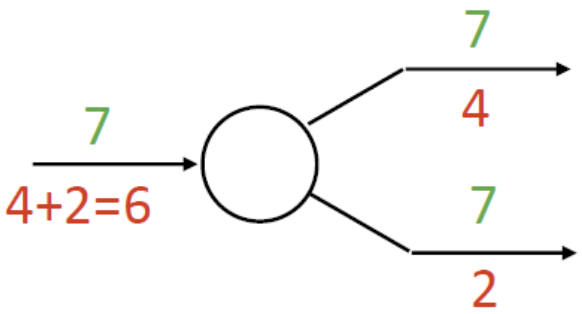
add gate: gradient distributor



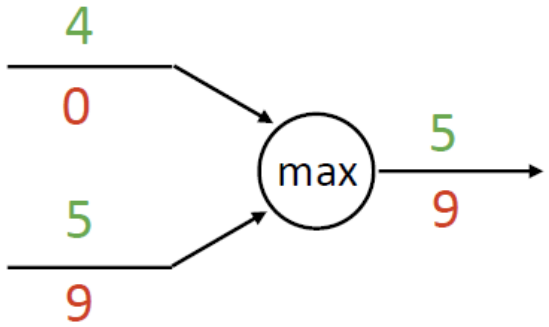
mul gate: "swap multiplier"



copy gate: gradient adder

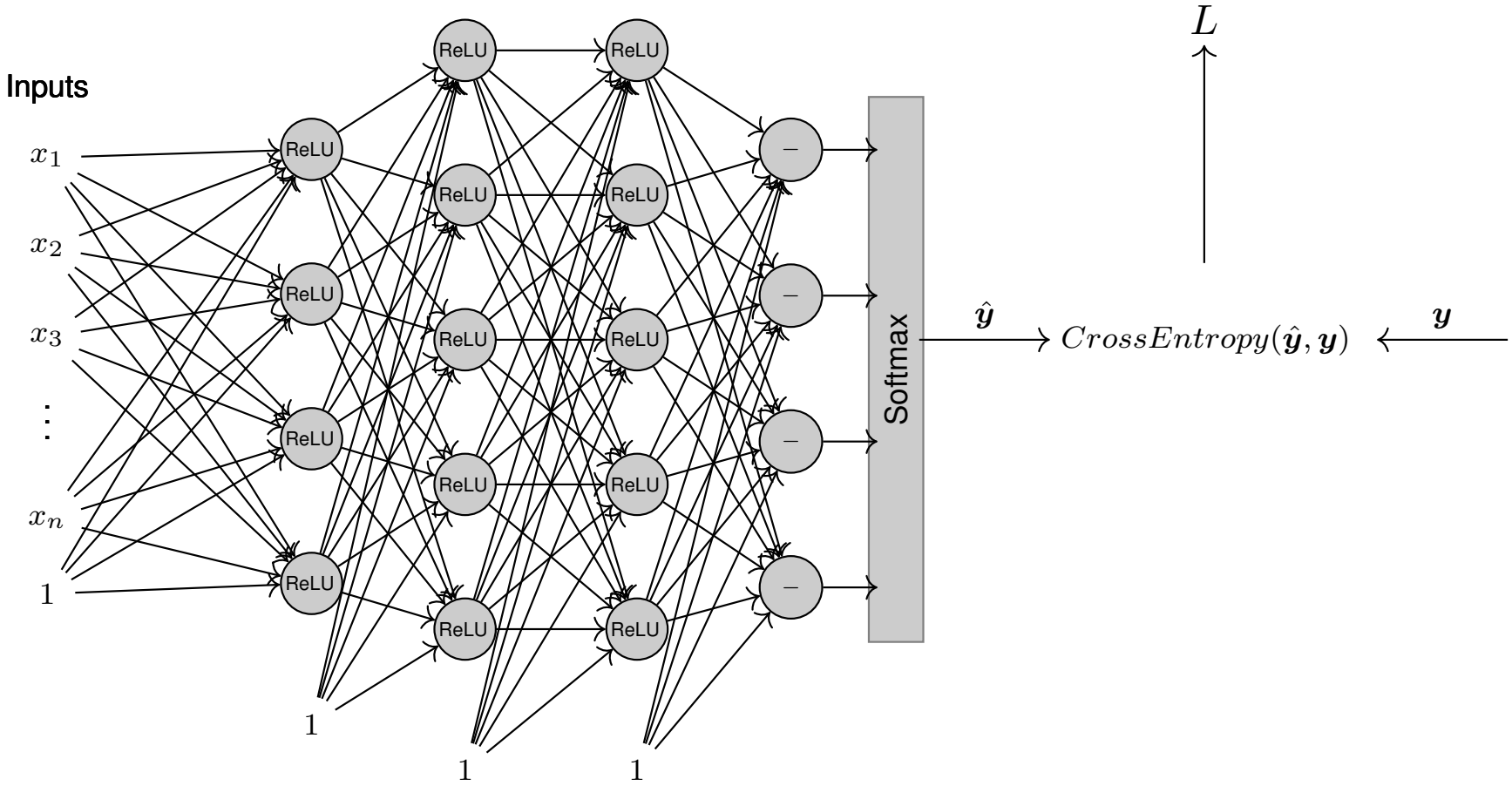


max gate: gradient router



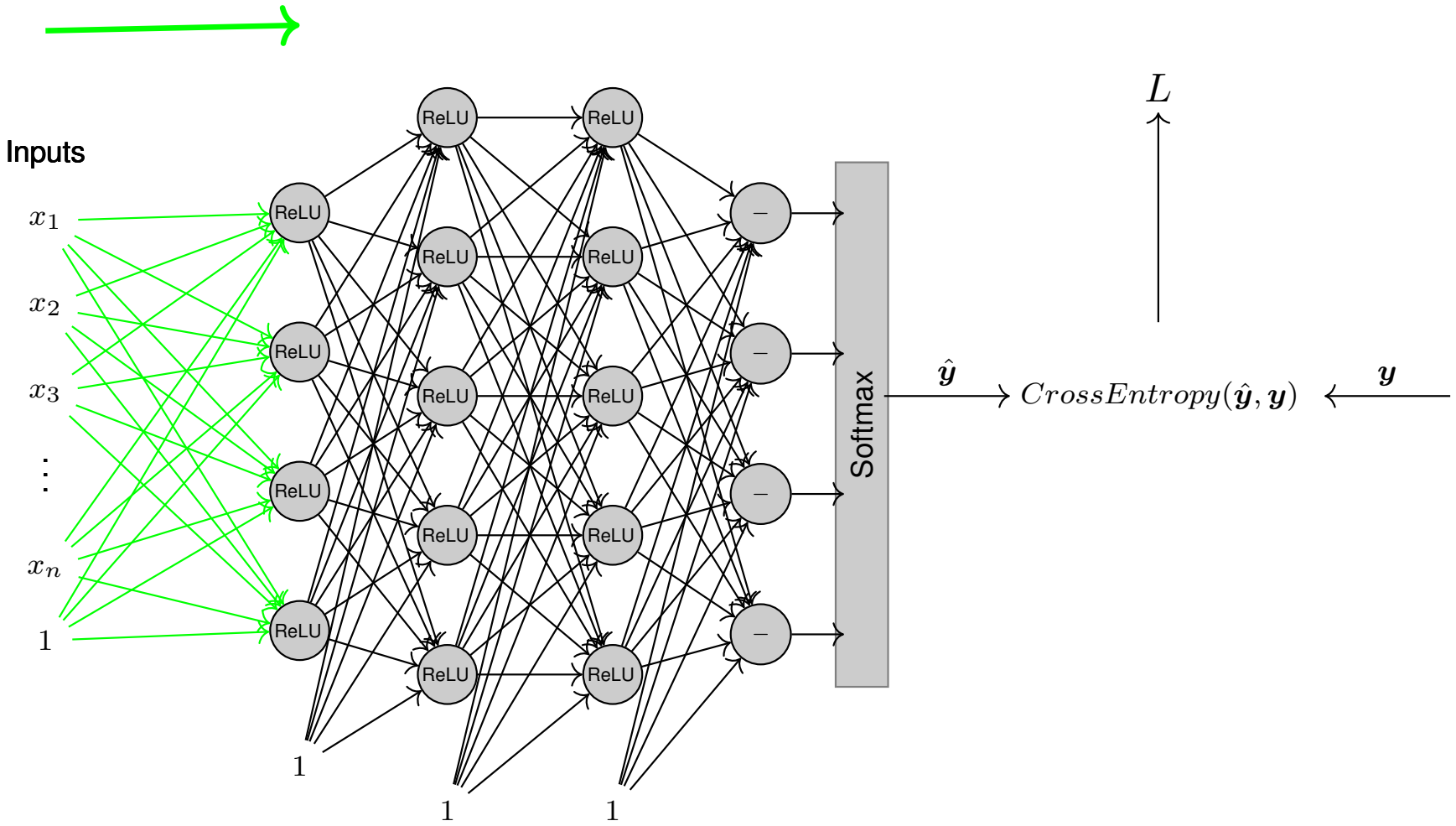
Backpropagation

Fully connected neural network - Forward Pass



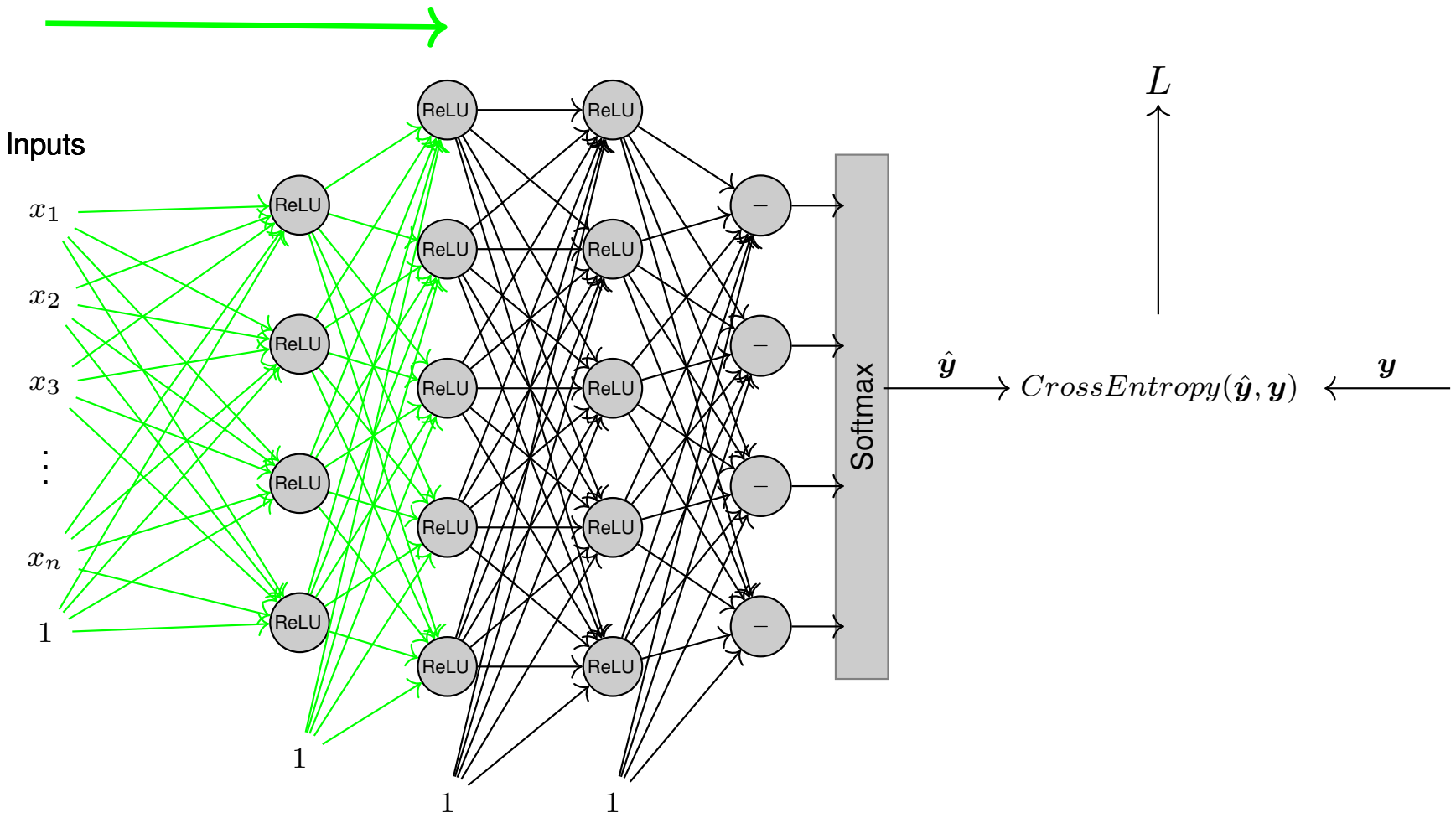
$$\hat{y} = Softmax(W^4(max(0, W^3(max(0, W^2(max(0, W^1 x + b^1)) + b^2)) + b^3)) + b^4)$$

Fully connected neural network - Forward Pass



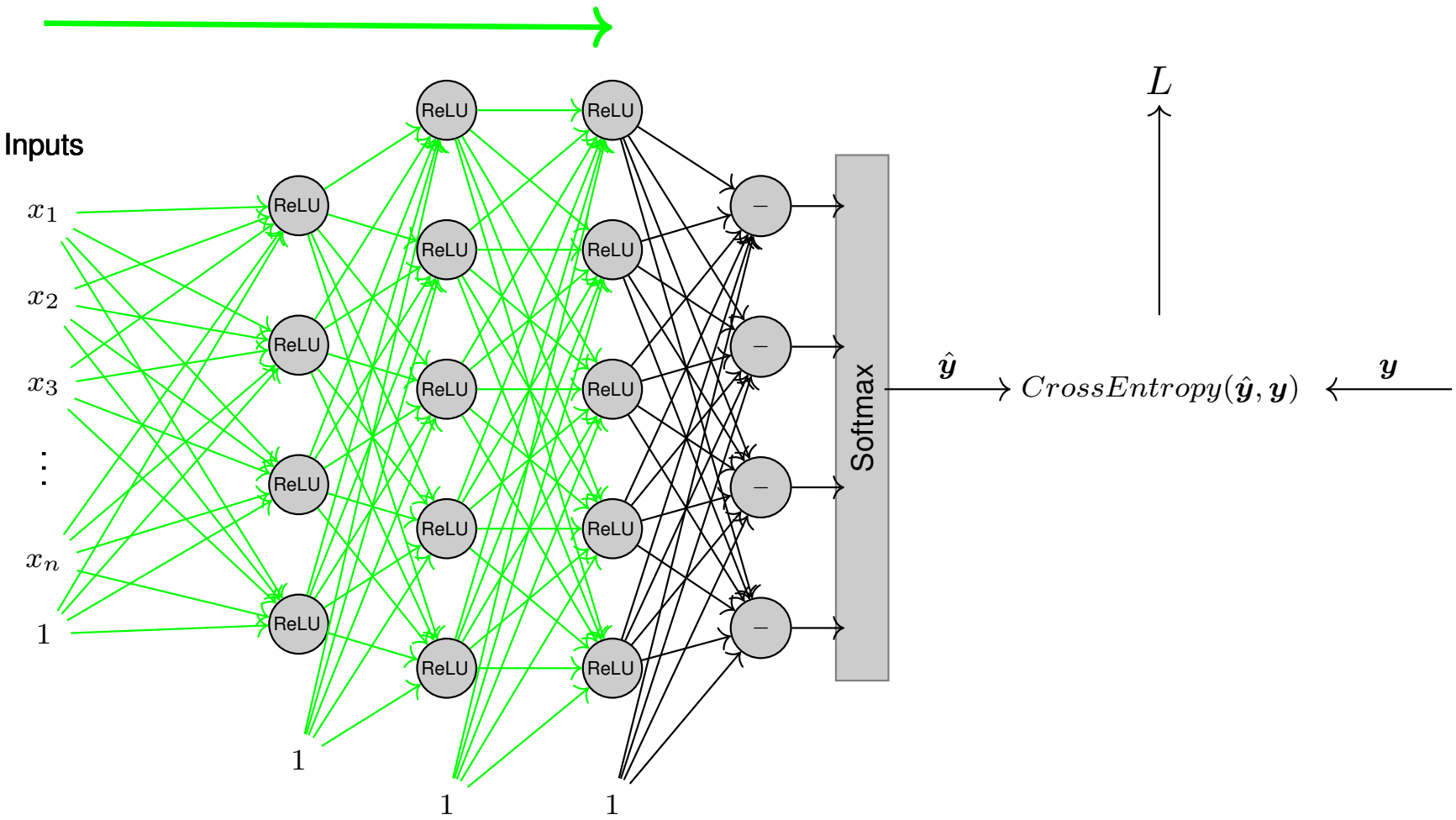
$$z^1 = W^1 x + b^1, \quad o^1 = ReLU(z^1)$$

Fully connected neural network - Forward Pass



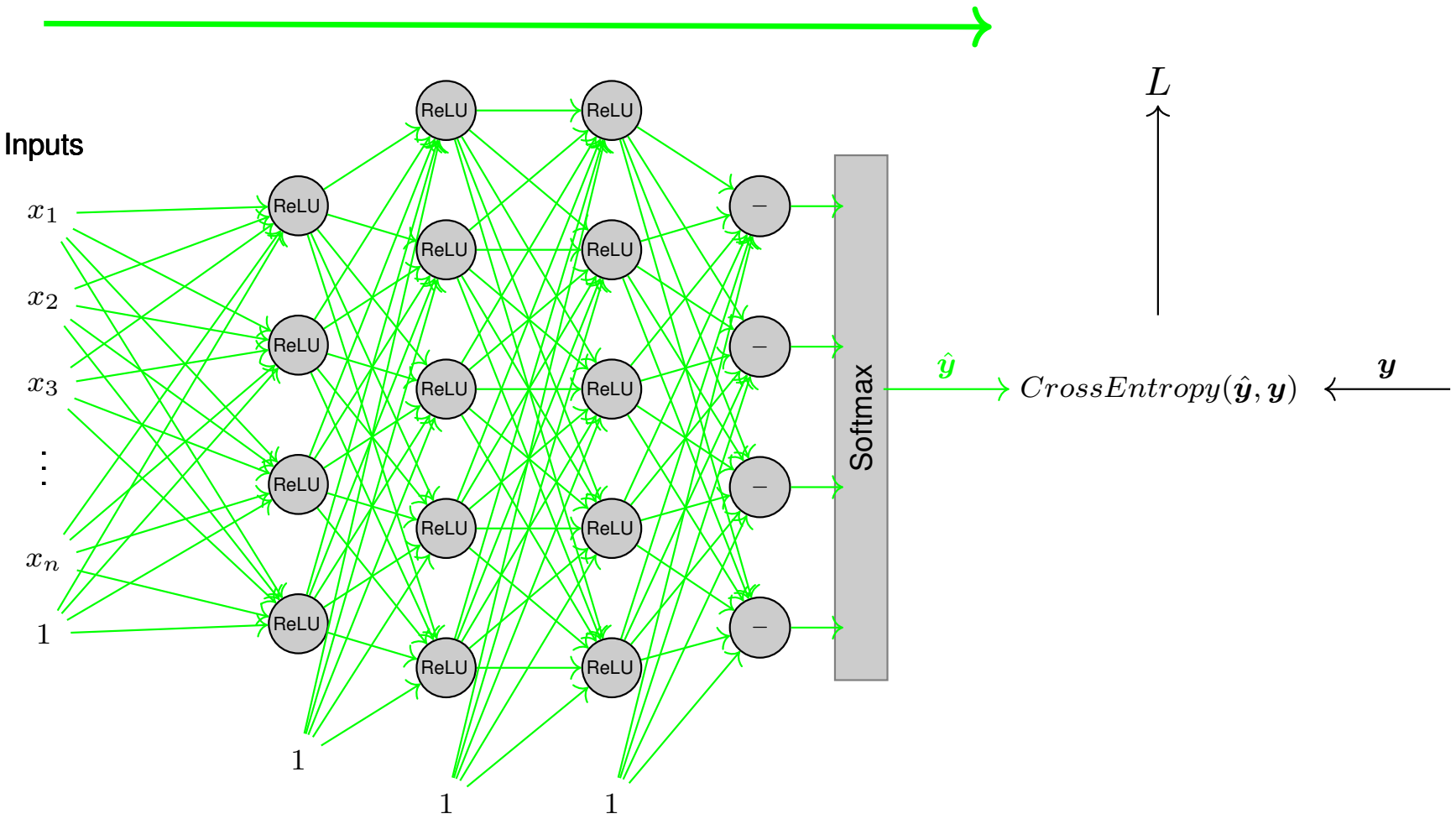
$$z^2 = W^2 o^1 + b^2, \quad o^2 = ReLU(z^2)$$

Fully connected neural network - Forward Pass



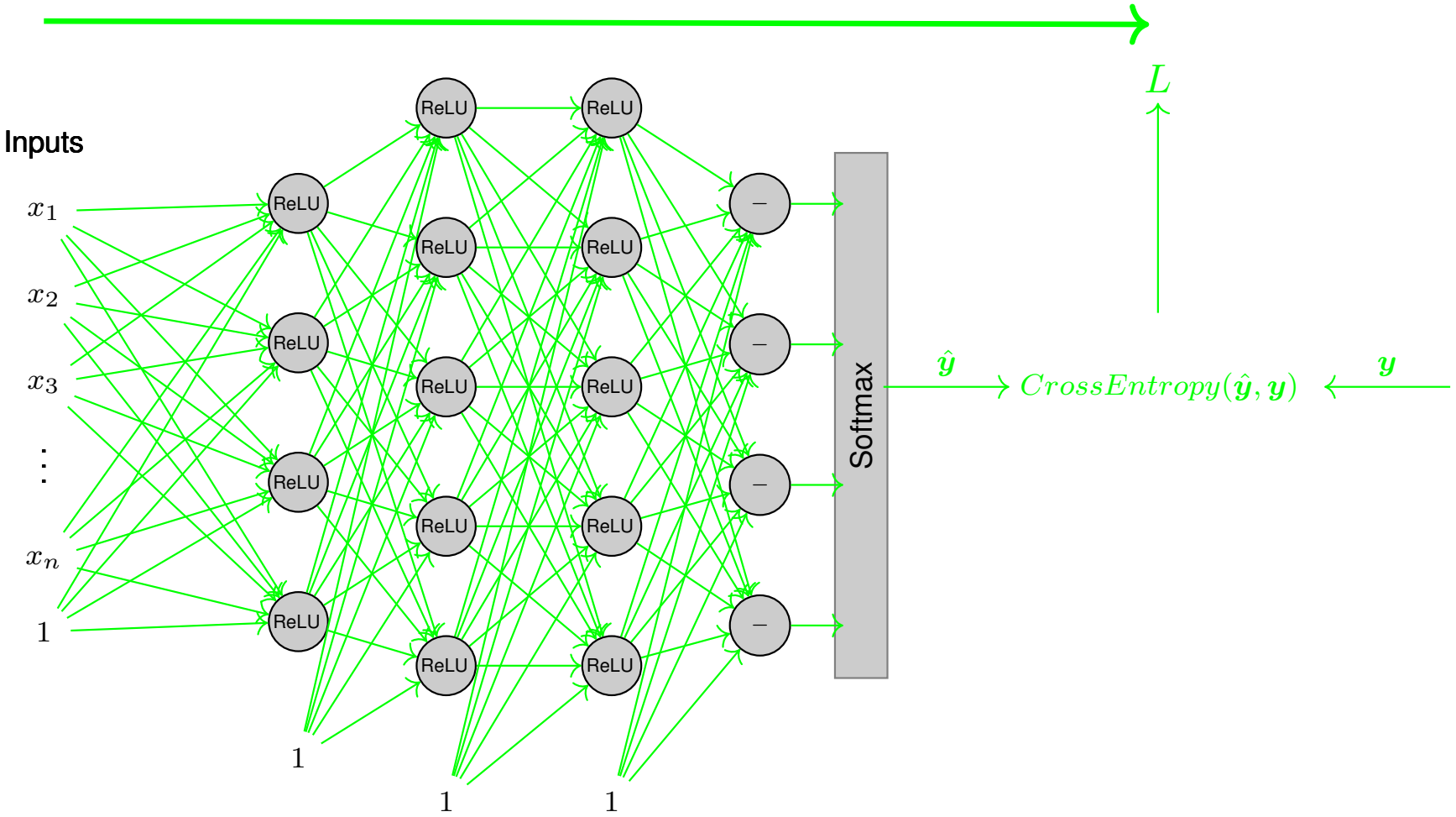
$$z^3 = W^3 o^2 + b^3, \quad o^3 = ReLU(z^3)$$

Fully connected neural network - Forward Pass



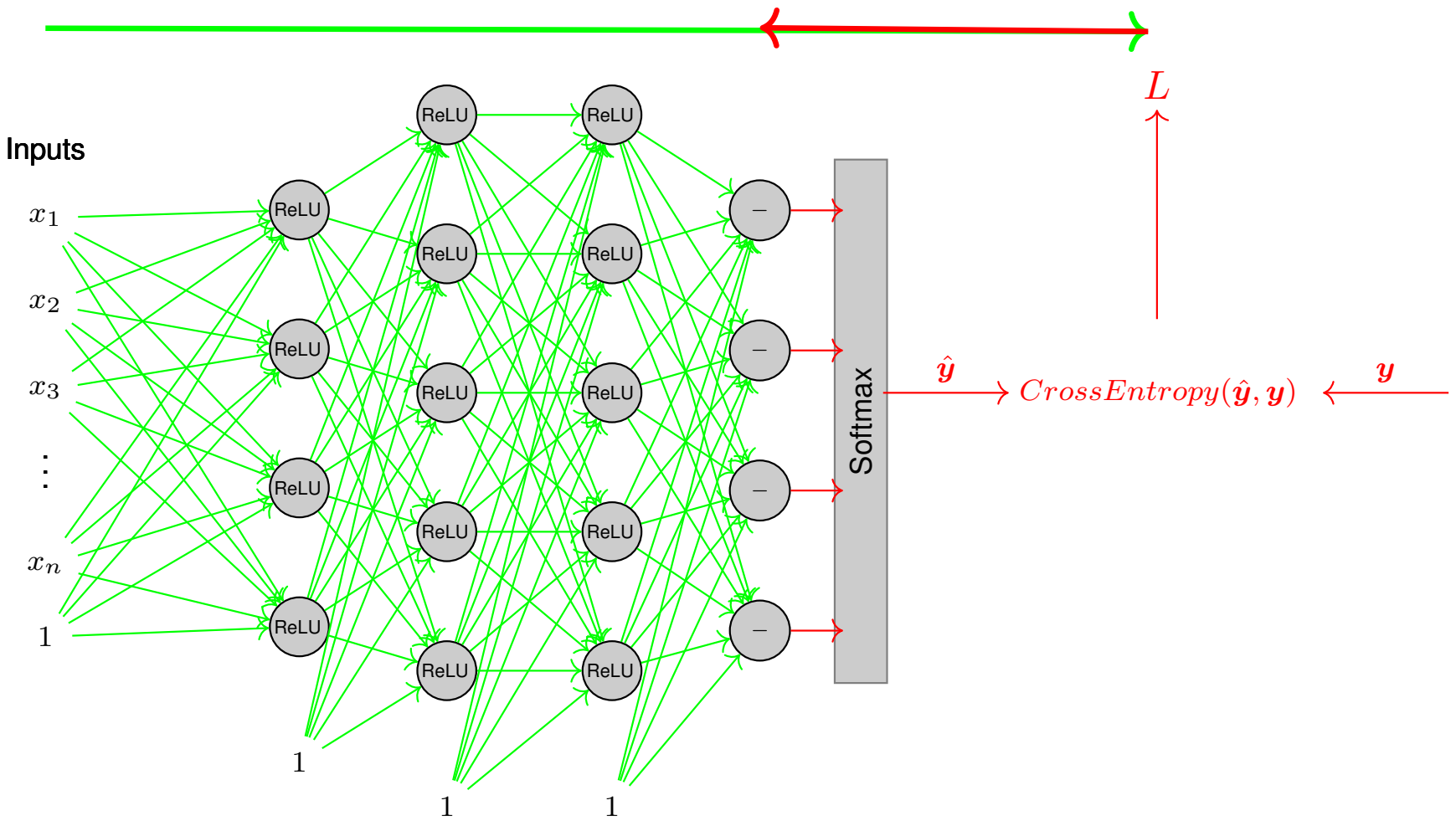
$$z^4 = W^4 o^3 + b^4, \quad \hat{y} = softmax(z^4)$$

Fully connected neural network - Forward Pass



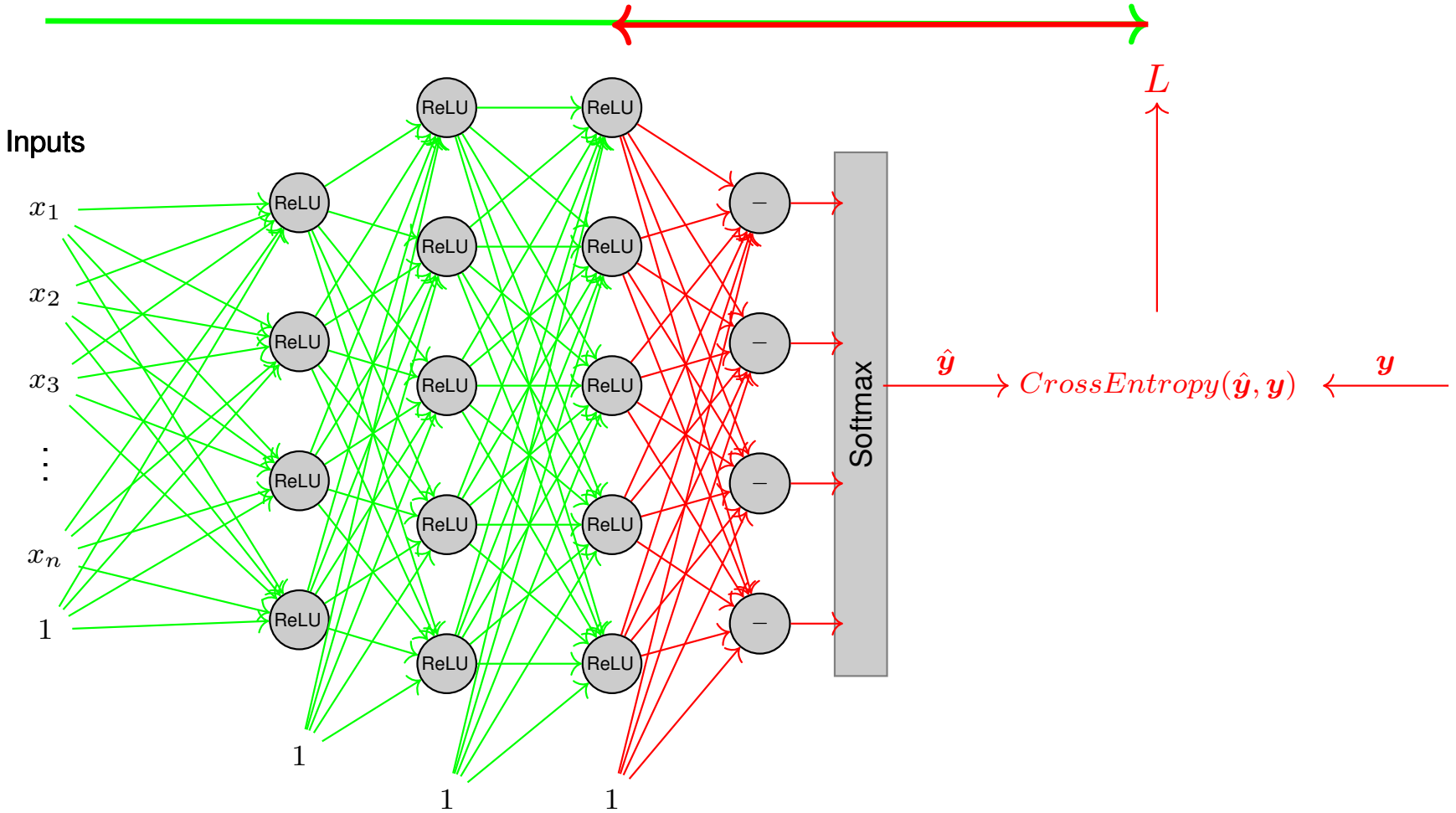
$$L = CrossEntropy(\hat{\boldsymbol{y}}, \boldsymbol{y})$$

Fully connected neural network - Backward Pass



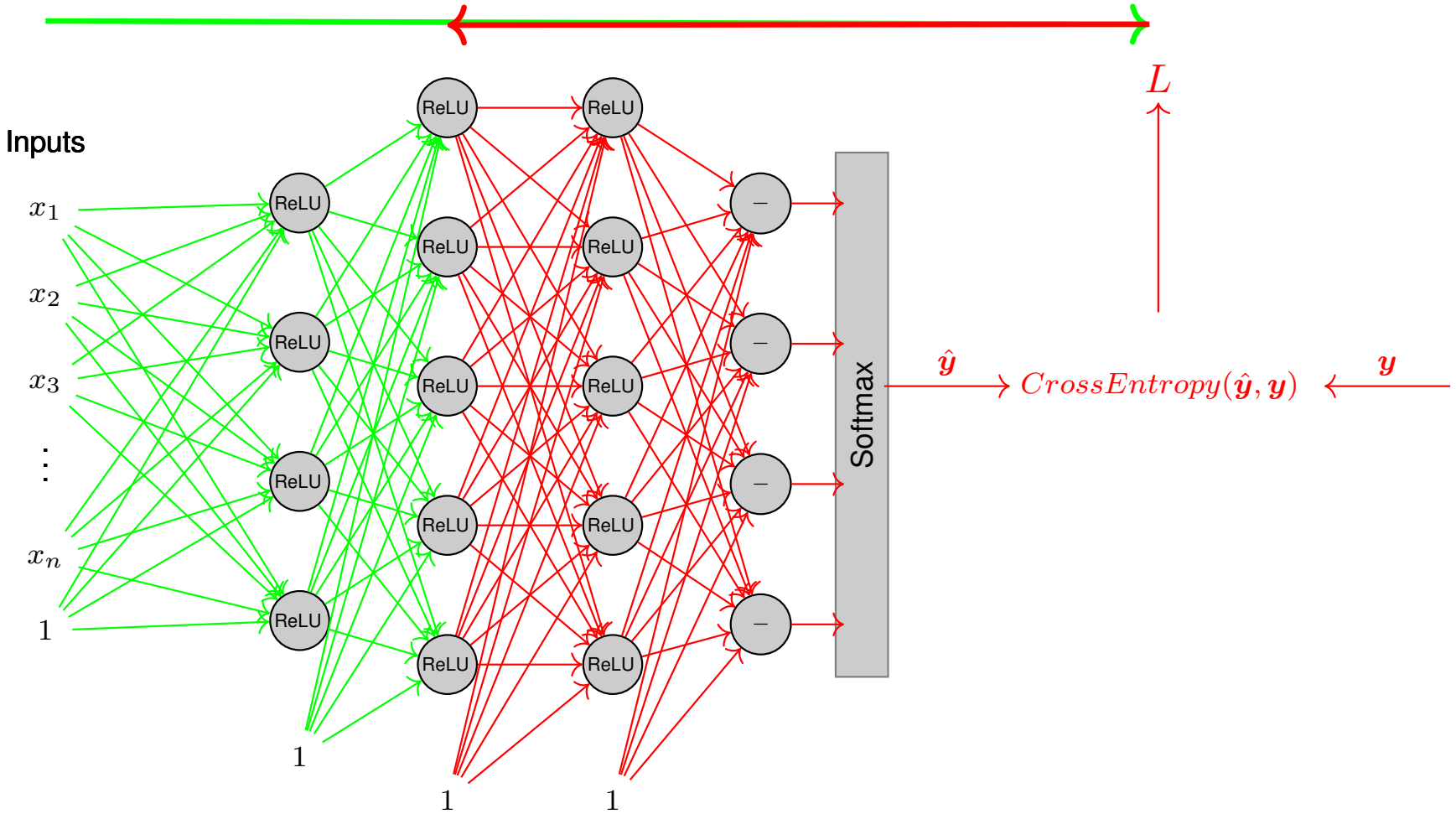
$$\frac{\partial L}{\partial z^4} = \hat{y} - y$$

Fully connected neural network - Backward Pass



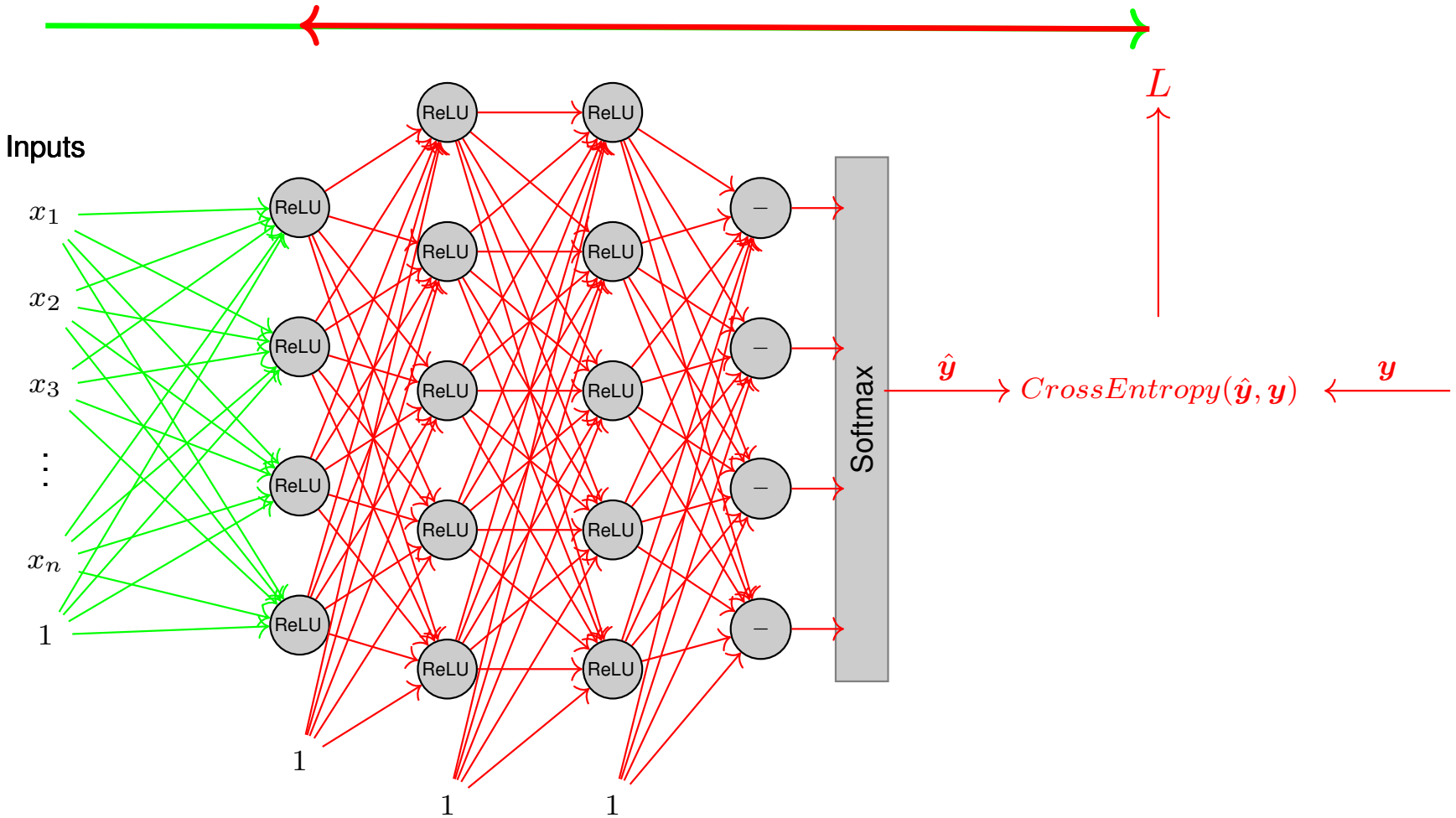
$$\frac{\partial L}{\partial W^4} = \frac{\partial L}{\partial z^4}\frac{\partial z^4}{\partial W^4} = \frac{\partial L}{\partial z^4}o^3, \quad \frac{\partial L}{\partial b^4} = \frac{\partial L}{\partial z^4}\frac{\partial z^4}{\partial b^4} = \frac{\partial L}{\partial z^4}, \quad \frac{\partial L}{\partial o^3} = \frac{\partial L}{\partial z^4}\frac{\partial z^4}{\partial o^3} = \frac{\partial L}{\partial z^4}W^4$$

Fully connected neural network - Backward Pass



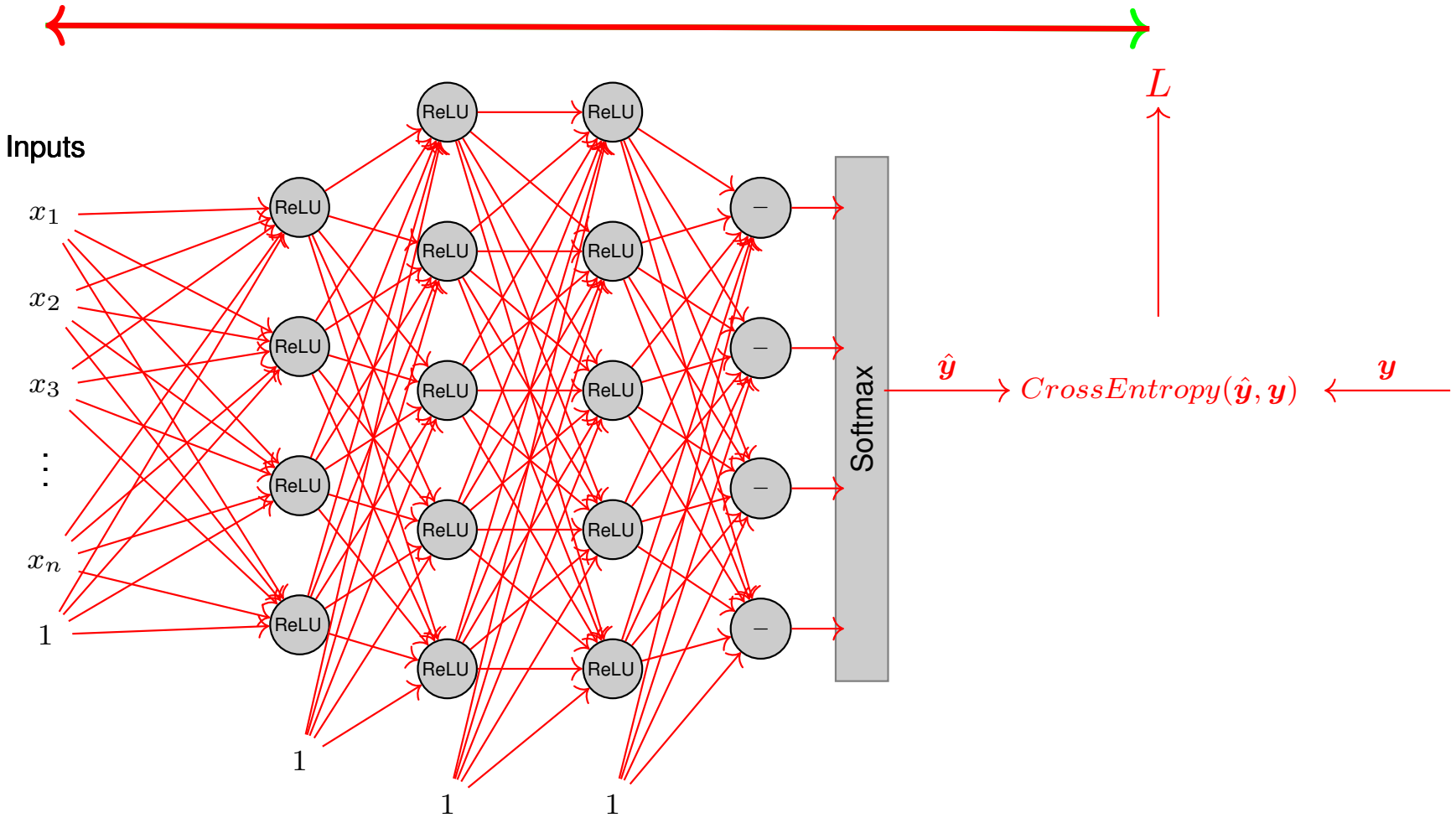
$$\frac{\partial L}{\partial z^3} = \frac{\partial L}{\partial o^3}\frac{\partial o^3}{\partial z^3} = \frac{\partial L}{\partial o^3} ReLU(sign(z^3)), \quad \frac{\partial L}{\partial W^3} = \frac{\partial L}{\partial z^3} o^2, \quad \frac{\partial L}{\partial b^3} = \frac{\partial L}{\partial z^3}, \quad \frac{\partial L}{\partial o^2} = \frac{\partial L}{\partial z^3} W^3$$

Fully connected neural network - Backward Pass



$$\frac{\partial L}{\partial z^2} = \frac{\partial L}{\partial o^2} ReLU(sign(z^2)), \quad \frac{\partial L}{\partial W^2} = \frac{\partial L}{\partial z^2} o^1, \quad \frac{\partial L}{\partial b^2} = \frac{\partial L}{\partial z^2}, \quad \frac{\partial L}{\partial o^1} = \frac{\partial L}{\partial z^2} W^2$$

Fully connected neural network - Backward Pass



$$\frac{\partial L}{\partial z^1} = \frac{\partial L}{\partial o^1} ReLU(sign(z^1)), \quad \frac{\partial L}{\partial W^1} = \frac{\partial L}{\partial z^1} x, \quad \frac{\partial L}{\partial b^1} = \frac{\partial L}{\partial z^1}, \quad \frac{\partial L}{\partial x} = \frac{\partial L}{\partial z^1} W^1$$

Unfold Gradient $\frac{\partial L}{\partial W^1}$

$$\frac{\partial L}{\partial W^1} = \frac{\partial L}{\partial \boldsymbol{o}^1} ReLU(sign(\boldsymbol{z}^1))\boldsymbol{x}$$

Unfold Gradient $\frac{\partial L}{\partial W^1}$

$$\frac{\partial L}{\partial W^1} = \frac{\partial L}{\partial \boldsymbol{o}^1} ReLU(sign(\boldsymbol{z}^1))\boldsymbol{x}$$

$$\frac{\partial L}{\partial W^1} = \frac{\partial L}{\partial \boldsymbol{o}^2} ReLU(sign(\boldsymbol{z}^2))W^2 ReLU(sign(\boldsymbol{z}^1))\boldsymbol{x}$$

Unfold Gradient $\frac{\partial L}{\partial W^1}$

$$\frac{\partial L}{\partial W^1} = \frac{\partial L}{\partial \boldsymbol{o}^1} ReLU(sign(\boldsymbol{z}^1))\boldsymbol{x}$$

$$\frac{\partial L}{\partial W^1} = \frac{\partial L}{\partial \boldsymbol{o}^2} ReLU(sign(\boldsymbol{z}^2))W^2 ReLU(sign(\boldsymbol{z}^1))\boldsymbol{x}$$

$$\frac{\partial L}{\partial W^1} = \frac{\partial L}{\partial \boldsymbol{o}^3} ReLU(sign(\boldsymbol{z}^3))W^3 ReLU(sign(\boldsymbol{z}^2))W^2 ReLU(sign(\boldsymbol{z}^1))\boldsymbol{x}$$

Unfold Gradient $\frac{\partial L}{\partial W^1}$

$$\frac{\partial L}{\partial W^1} = \frac{\partial L}{\partial \boldsymbol{o}^1} ReLU(sign(\boldsymbol{z}^1))\boldsymbol{x}$$

$$\frac{\partial L}{\partial W^1} = \frac{\partial L}{\partial \boldsymbol{o}^2} ReLU(sign(\boldsymbol{z}^2))W^2 ReLU(sign(\boldsymbol{z}^1))\boldsymbol{x}$$

$$\frac{\partial L}{\partial W^1} = \frac{\partial L}{\partial \boldsymbol{o}^3} ReLU(sign(\boldsymbol{z}^3))W^3 ReLU(sign(\boldsymbol{z}^2))W^2 ReLU(sign(\boldsymbol{z}^1))\boldsymbol{x}$$

$$\frac{\partial L}{\partial W^1} = (\hat{\boldsymbol{y}} - \boldsymbol{y})W^4 ReLU(sign(\boldsymbol{z}^3))W^3 ReLU(sign(\boldsymbol{z}^2))W^2 ReLU(sign(\boldsymbol{z}^1))\boldsymbol{x}$$

What about vector-valued functions

- **Scalar to Scalar**
  - Derivative

$$x \in \mathbb{R}, y \in \mathbb{R} \rightarrow \frac{dy}{dx} \in \mathbb{R}$$

  - If x changes by a small amount, how much will y change?

What about vector-valued functions

- **Scalar to Scalar**
  - Derivative

$$x \in \mathbb{R}, y \in \mathbb{R} \rightarrow \frac{dy}{dx} \in \mathbb{R}$$

  - If x changes by a small amount, how much will y change?
- **Vector to Scalar**
  - Derivative is Gradient

$$\boldsymbol{x} \in \mathbb{R}^N, y \in \mathbb{R} \rightarrow \frac{\partial y}{\partial \boldsymbol{x}} \in \mathbb{R}^N, \left(\frac{\partial y}{\partial \boldsymbol{x}}\right)_n = \frac{\partial y}{\partial x_n}$$

  - For each element of $\boldsymbol{x}$, if it changes by a small amount then how much will y change?
  - A vector of the same size of $\boldsymbol{x}$

What about vector-valued functions

- **Scalar to Scalar**
  - Derivative

$$x \in \mathbb{R}, y \in \mathbb{R} \rightarrow \frac{dy}{dx} \in \mathbb{R}$$

  - If x changes by a small amount, how much will y change?
- **Vector to Scalar**
  - Derivative is Gradient

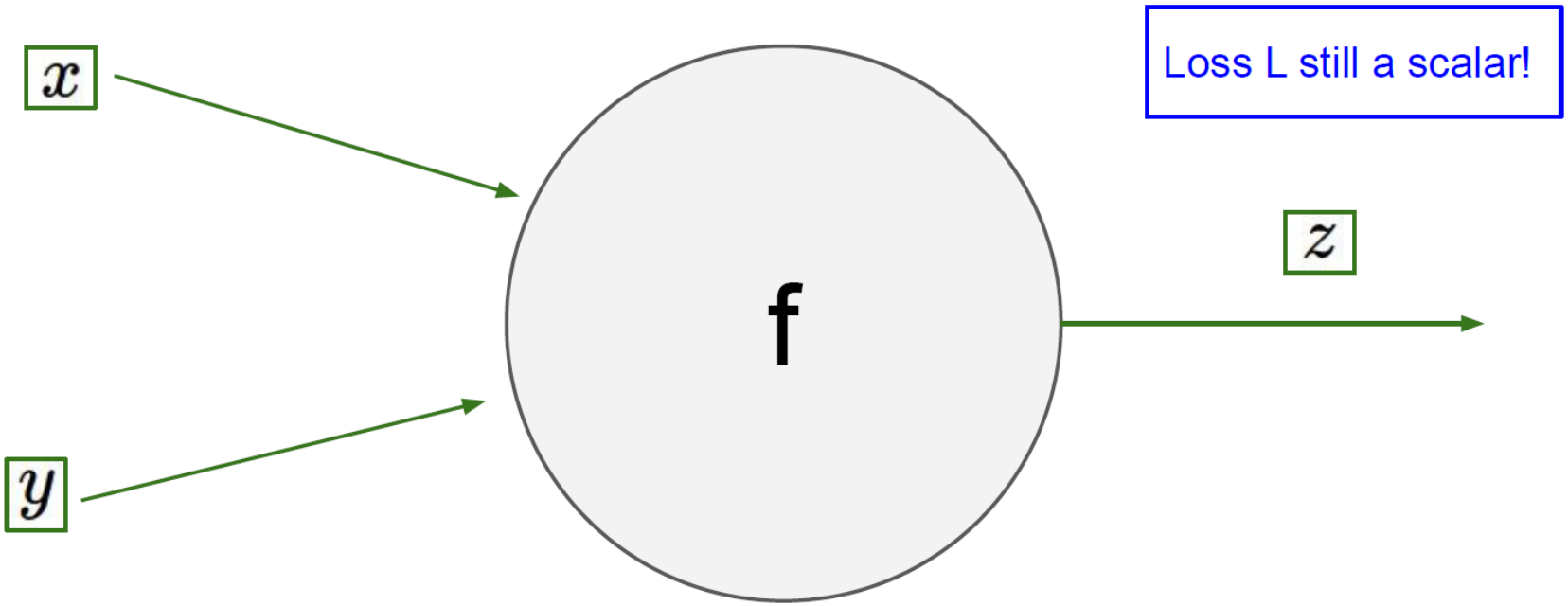
$$\boldsymbol{x} \in \mathbb{R}^N, y \in \mathbb{R} \rightarrow \frac{\partial y}{\partial \boldsymbol{x}} \in \mathbb{R}^N, \left(\frac{\partial y}{\partial \boldsymbol{x}}\right)_n = \frac{\partial y}{\partial x_n}$$

  - For each element of $\boldsymbol{x}$, if it changes by a small amount then how much will y change?
  - A vector of the same size of $\boldsymbol{x}$
- **Vector to Vector**
  - Derivative is Jacobian

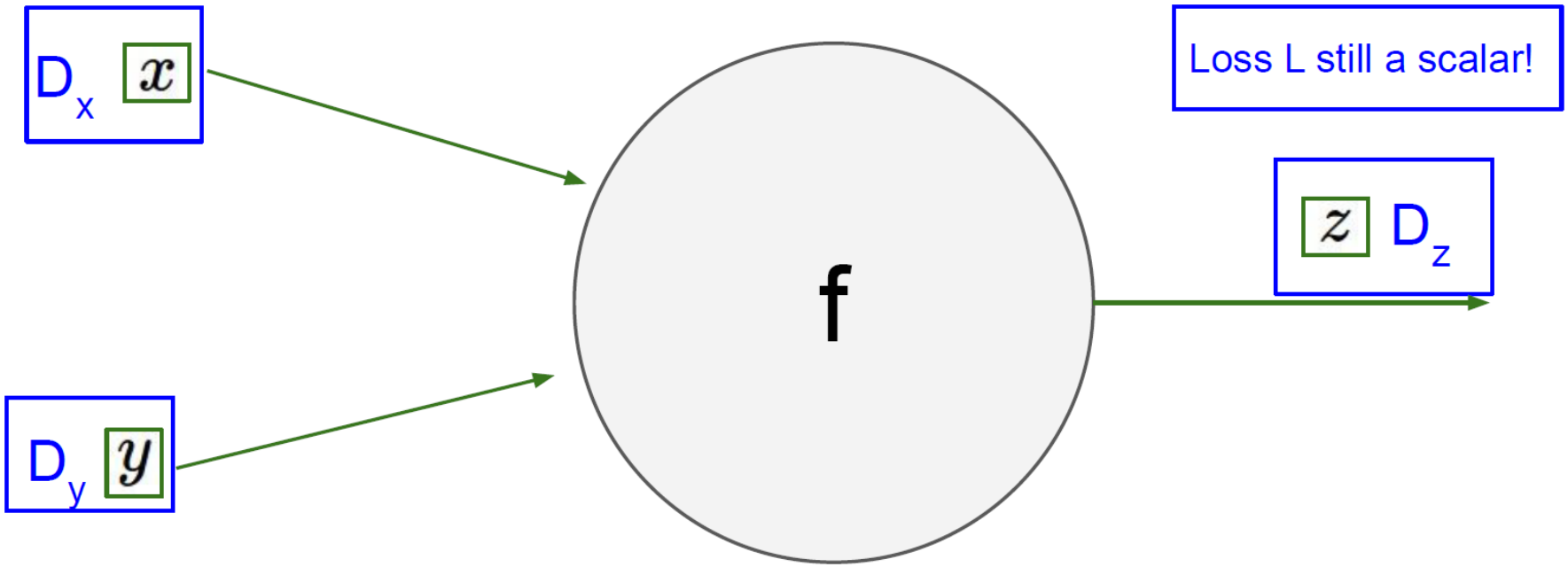
$$\boldsymbol{x} \in \mathbb{R}^N, \boldsymbol{y} \in \mathbb{R}^M \rightarrow \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}} \in \mathbb{R}^{N \times M}, \left(\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}}\right)_{n,m} = \frac{\partial y_m}{\partial x_n}$$

  - For each element of $\boldsymbol{x}$, if it changes by a small amount then how much will each element of $\boldsymbol{y}$ change?

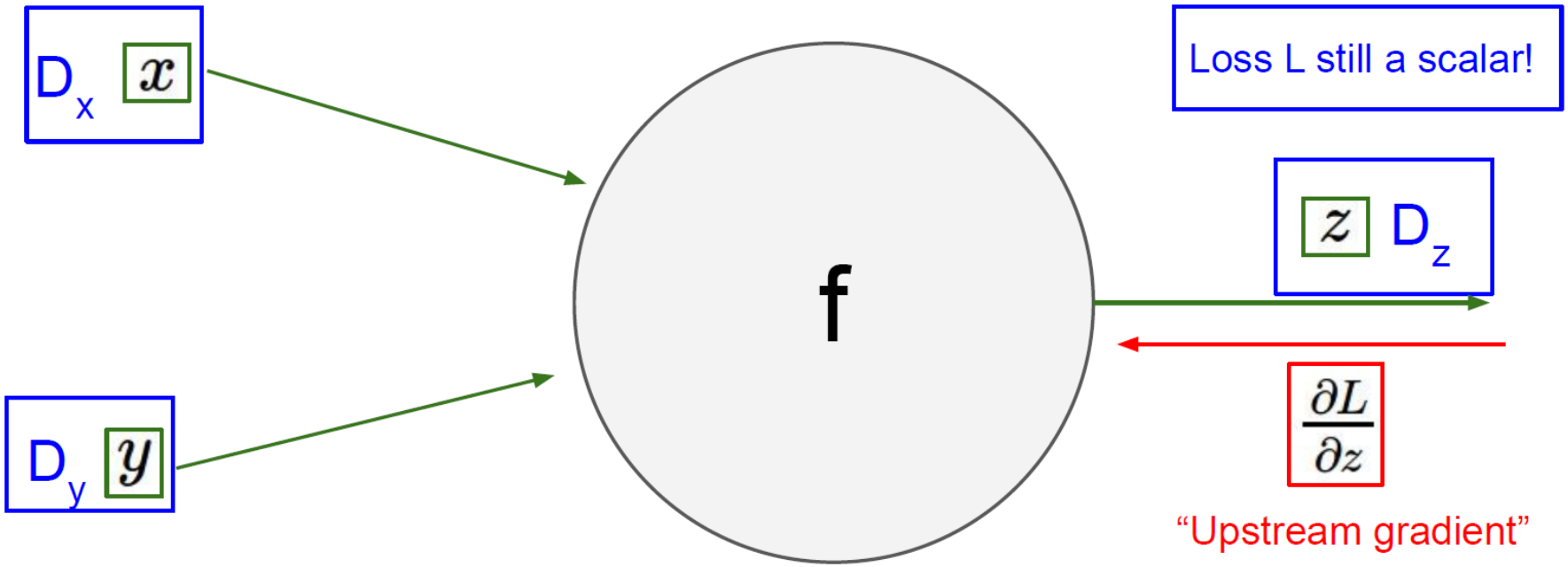
Backprop with Vectors



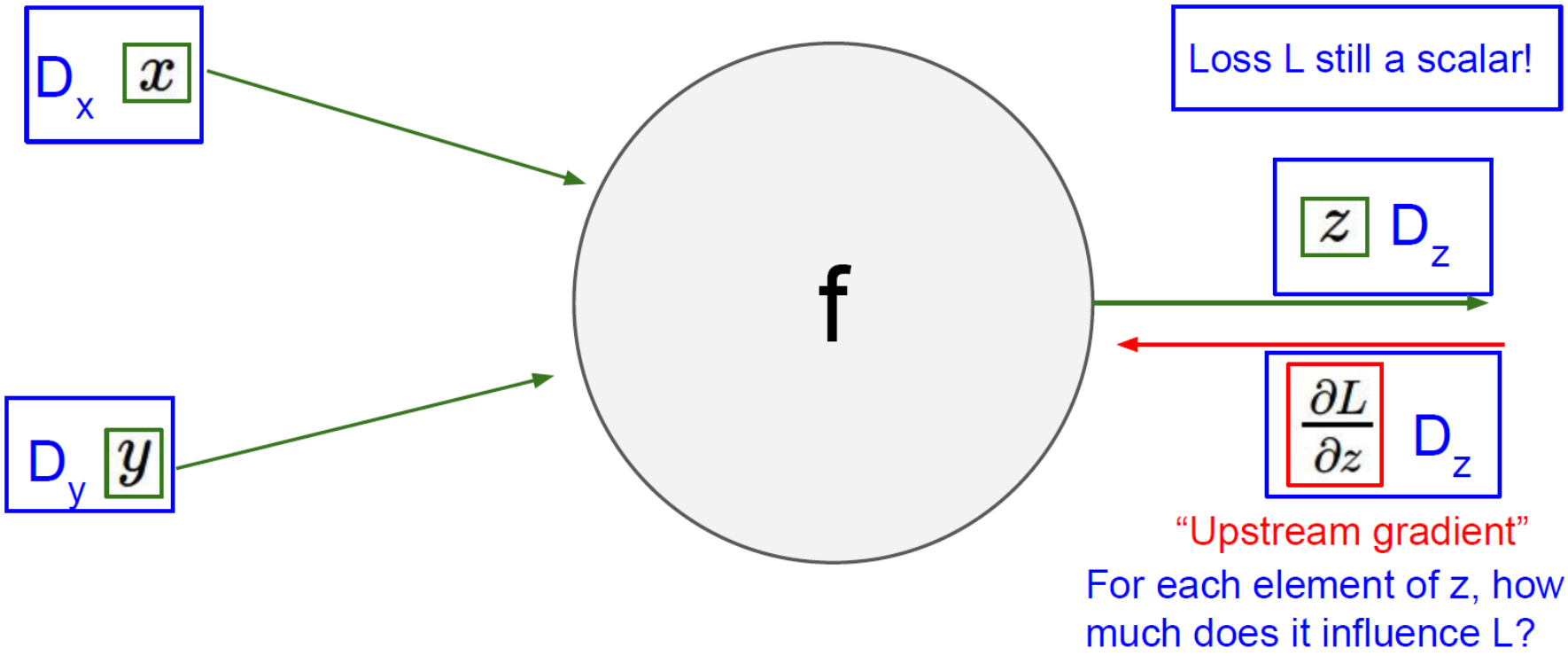
Backprop with Vectors



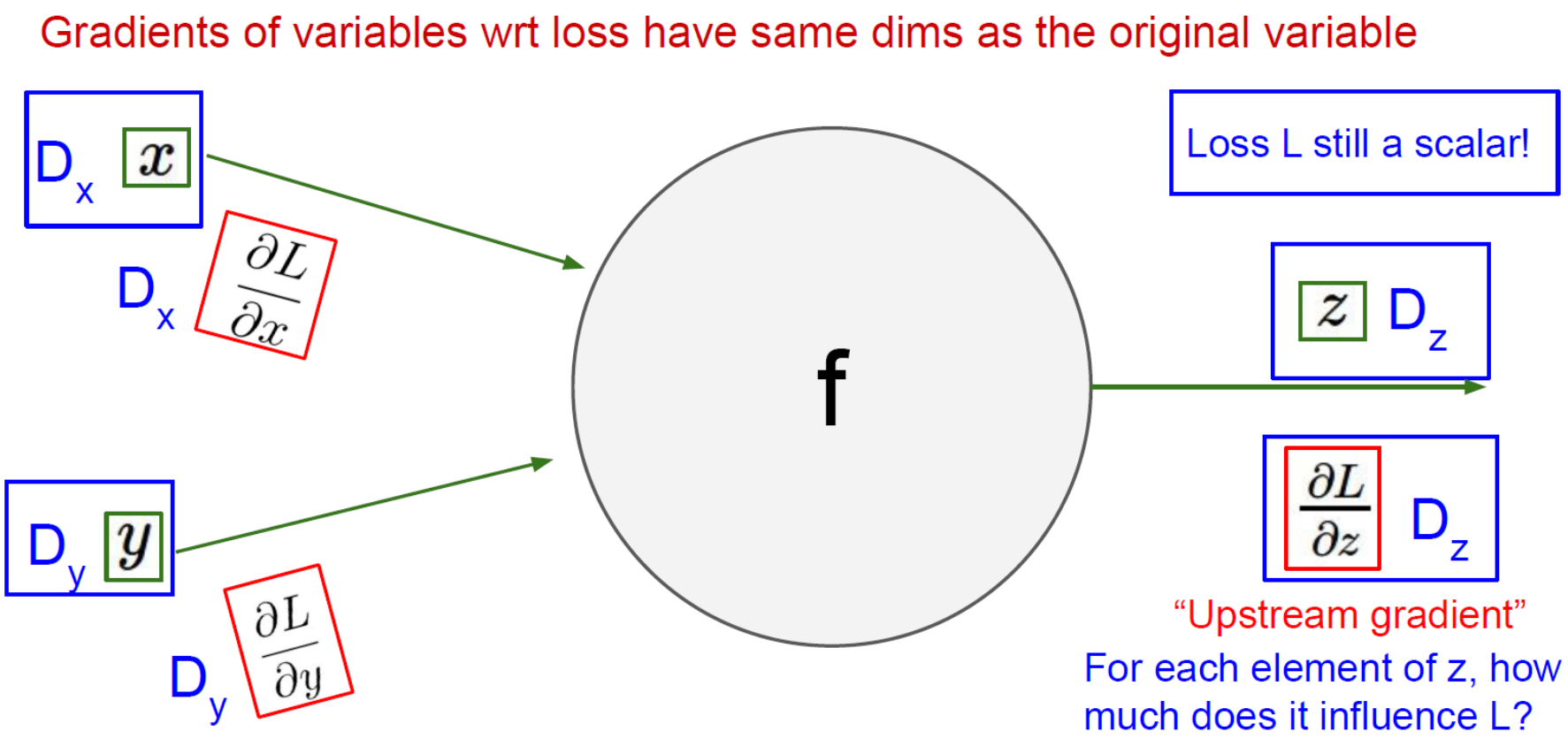
Backprop with Vectors



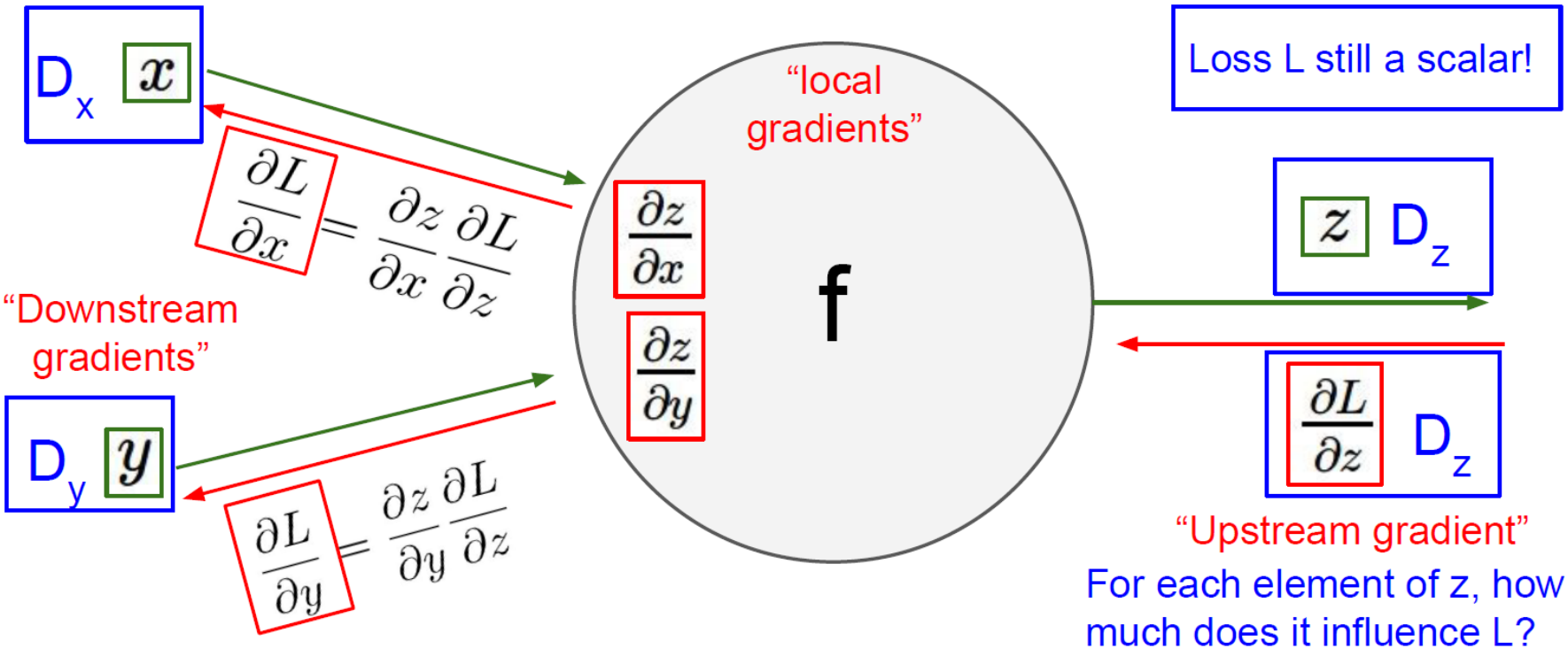
Backprop with Vectors



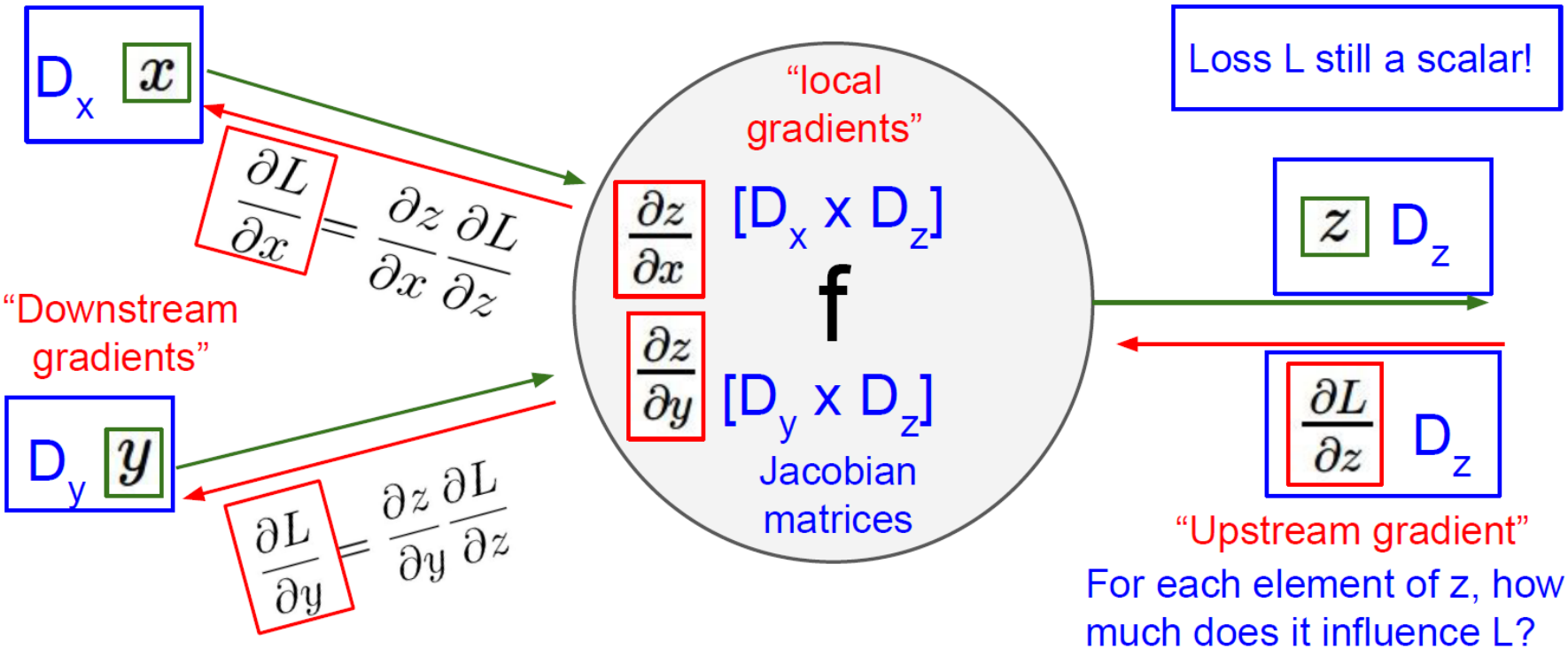
Backprop with Vectors



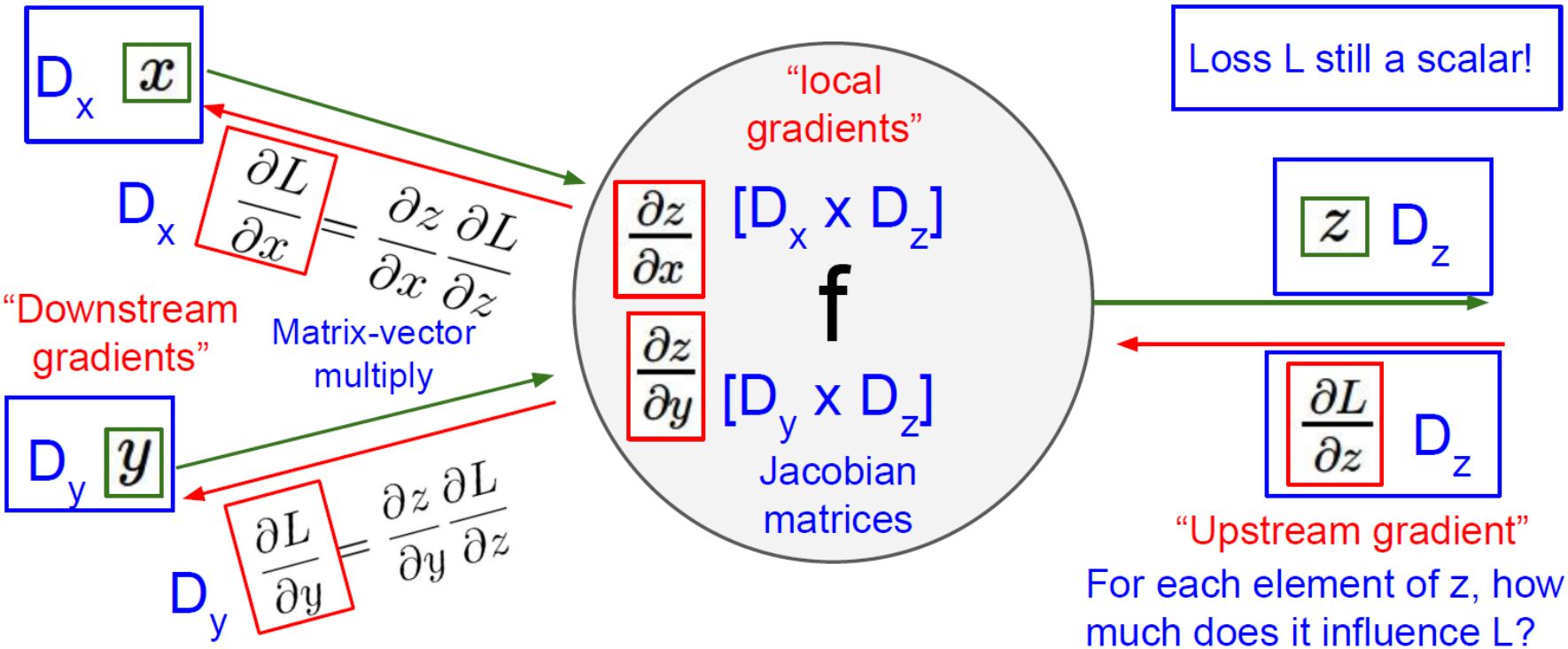
Backprop with Vectors



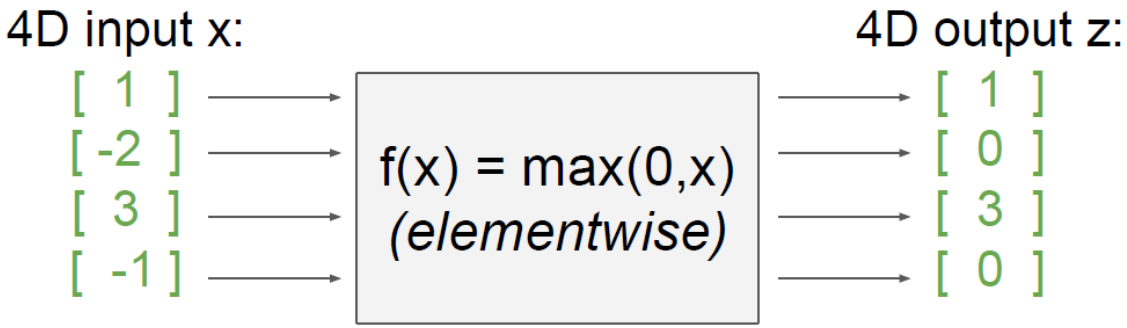
Backprop with Vectors



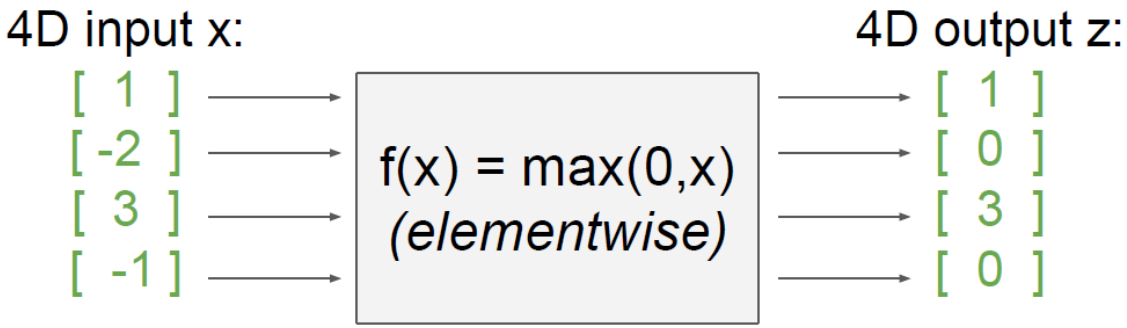
Backprop with Vectors



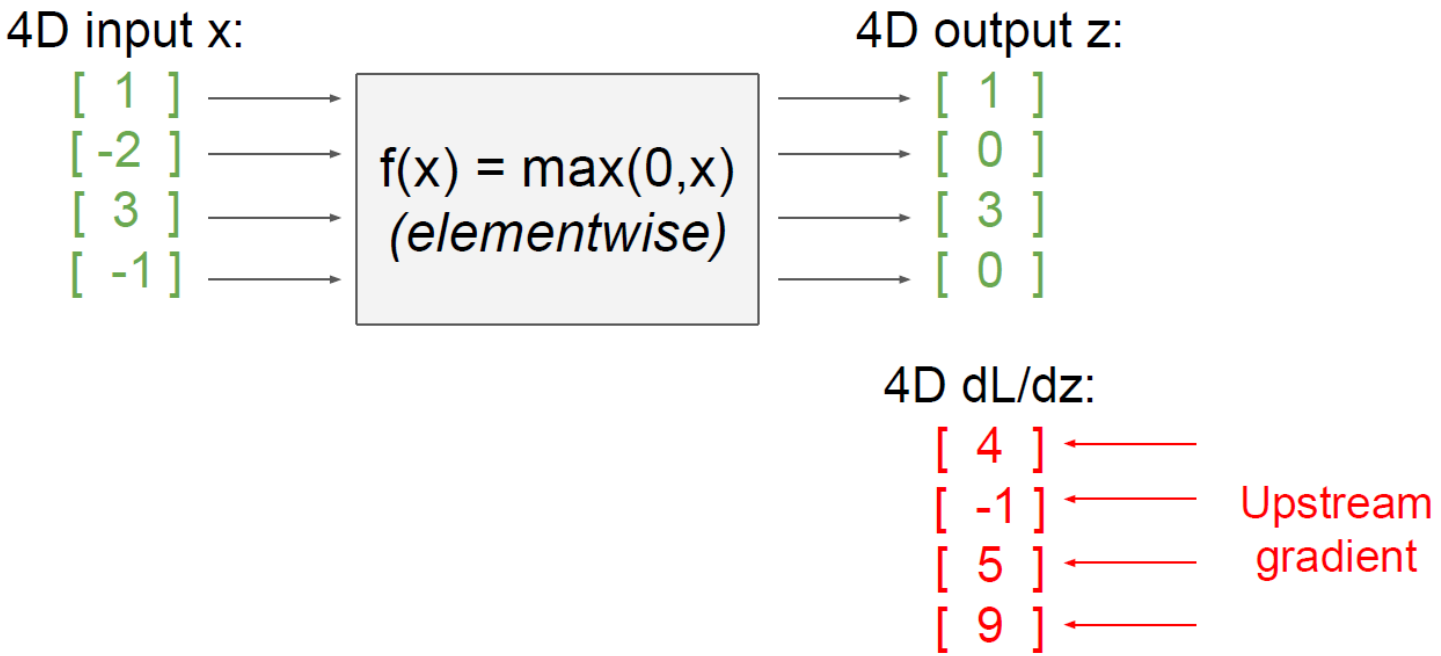
Backprop with Vectors



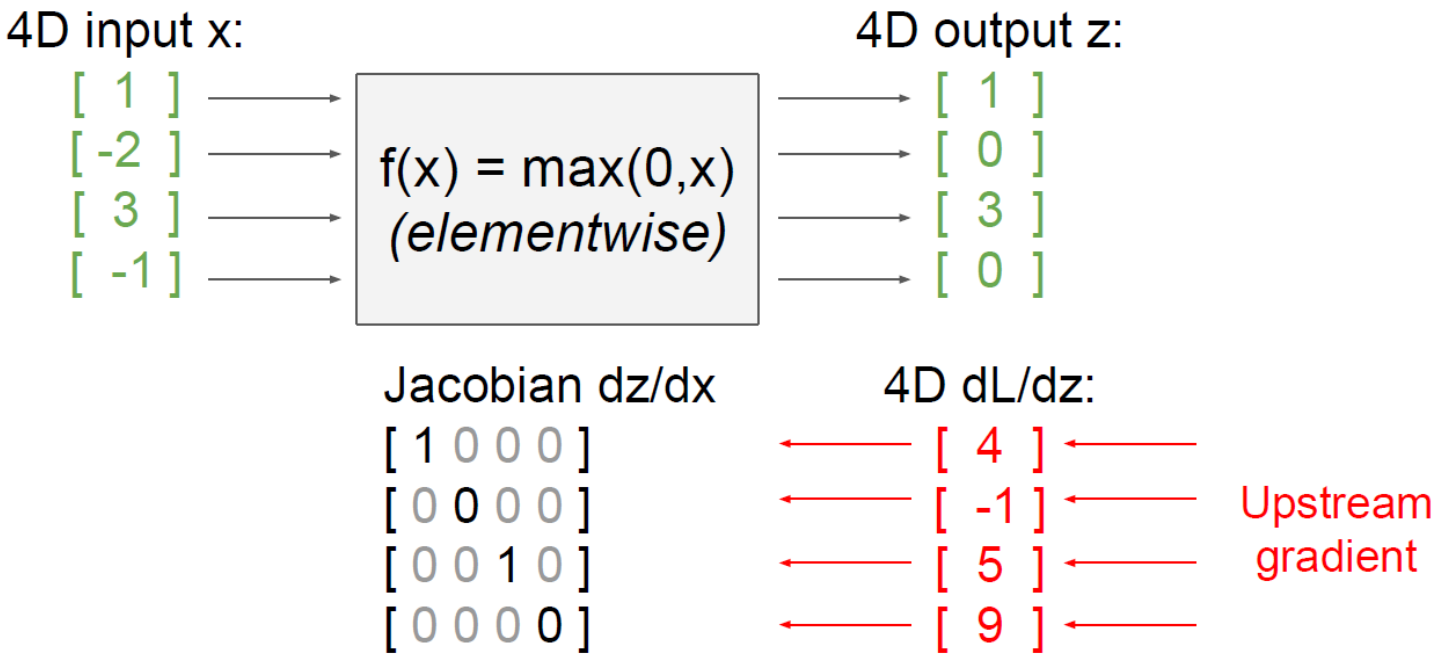
Backprop with Vectors



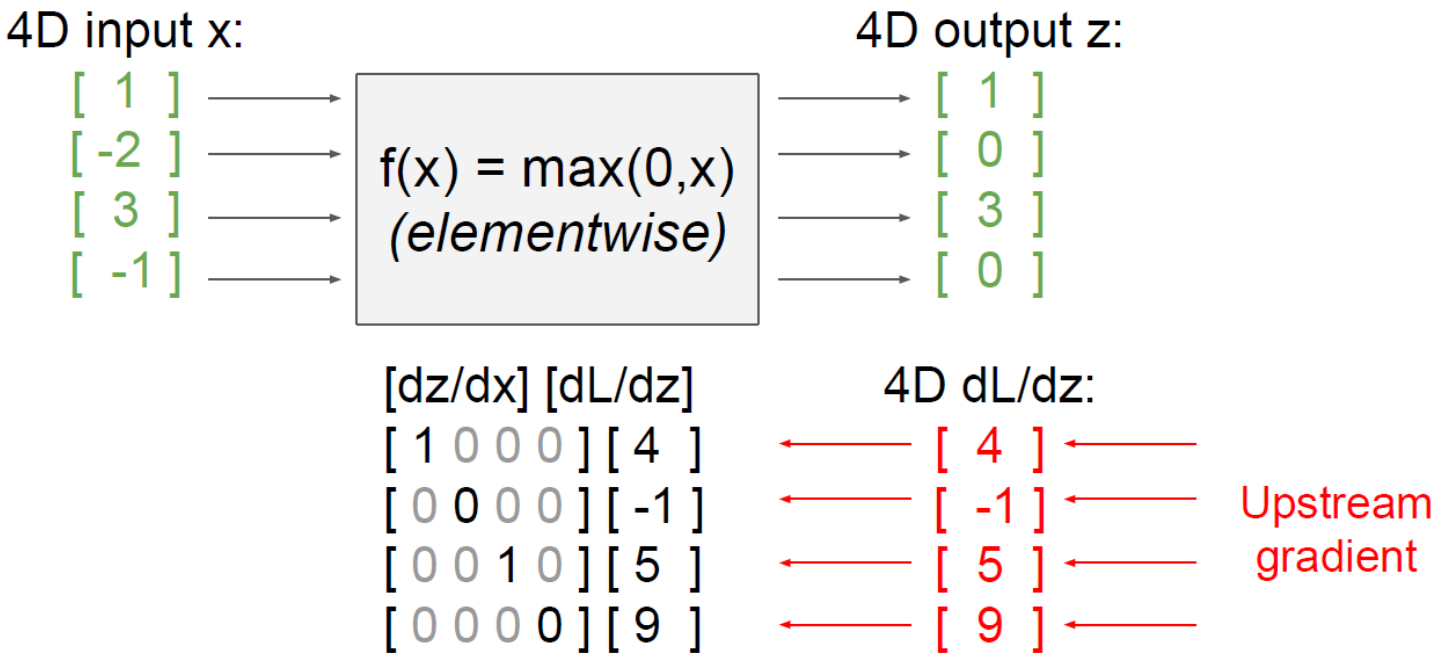
Backprop with Vectors



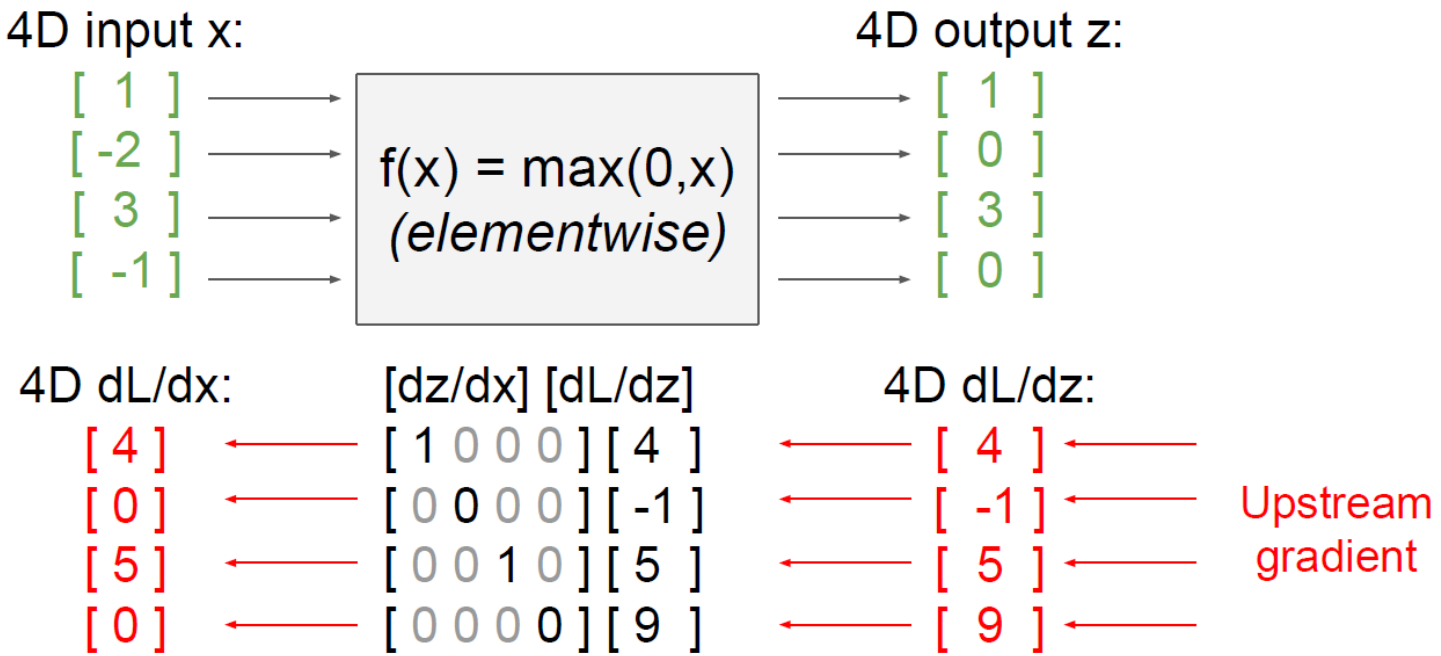
Backprop with Vectors



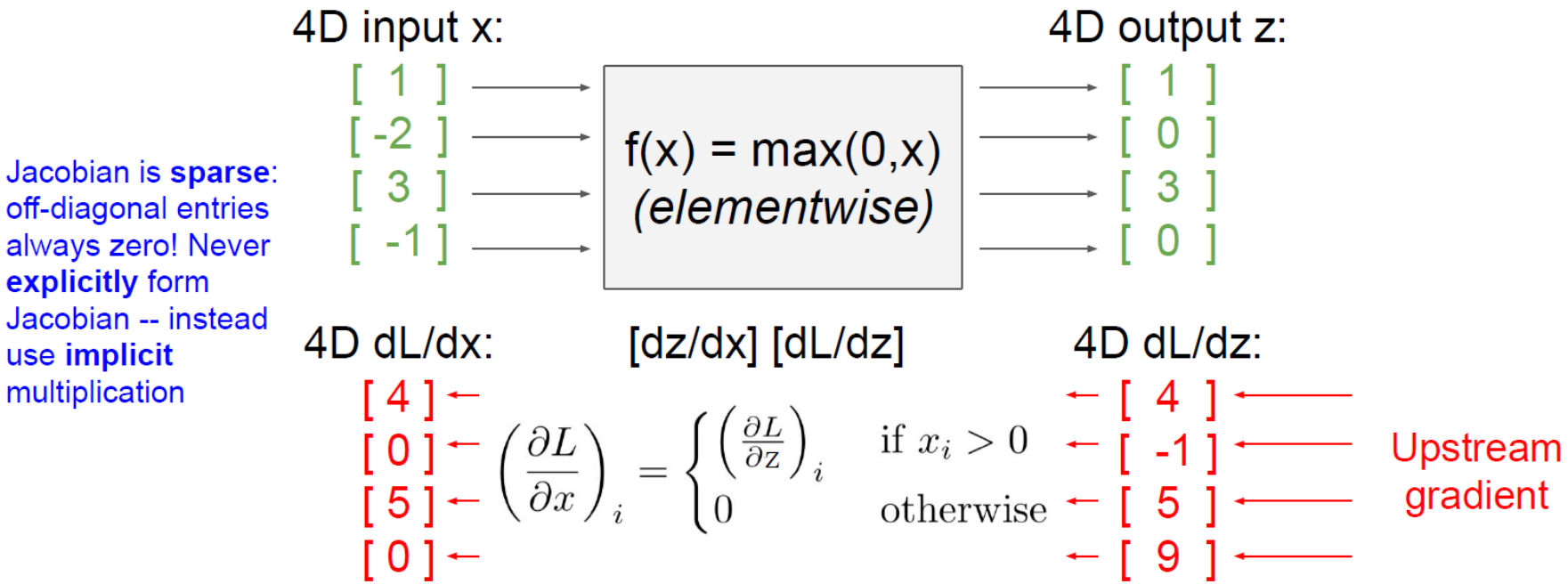
Backprop with Vectors



Backprop with Vectors



Backprop with Vectors



A vectorized example

A vectorized example: $f(x, W) = ||W \cdot x||^2 = \sum_{i=1}^{n} (W \cdot x)_i^2$

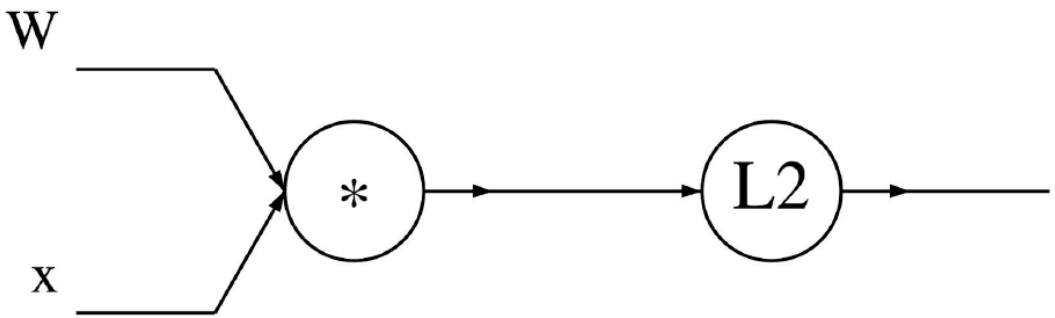
A vectorized example

A vectorized example: $f(x, W) = ||W \cdot x||^2 = \sum_{i=1}^{n} (W \cdot x)_i^2$

$\in \mathbb{R}^n$ $\in \mathbb{R}^{n \times n}$

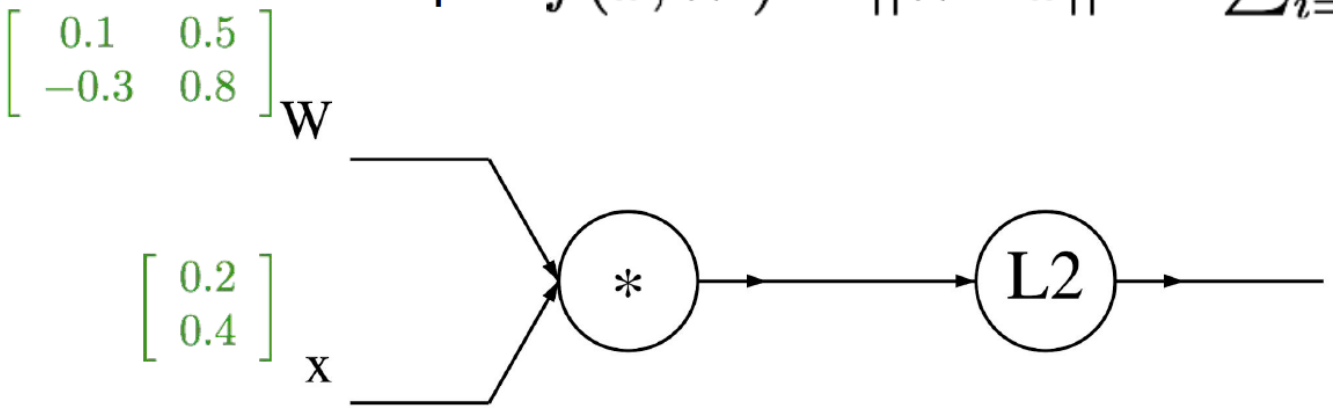
A vectorized example

A vectorized example: $f(x, W) = ||W \cdot x||^2 = \sum_{i=1}^{n} (W \cdot x)_i^2$



A vectorized example

A vectorized example: $f(x, W) = ||W \cdot x||^2 = \sum_{i=1}^{n} (W \cdot x)_i^2$



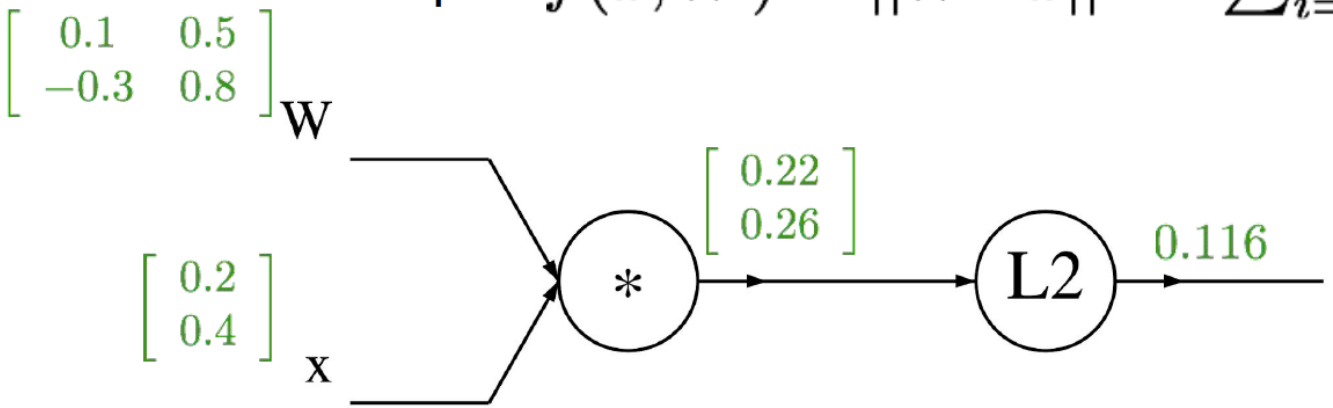
$$q = W \cdot x = \begin{pmatrix} W_{1,1}x_1 + \cdots + W_{1,n}x_n \\ \vdots \\ W_{n,1}x_1 + \cdots + W_{n,n}x_n \end{pmatrix}$$

$$f(q) = ||q||^2 = q_1^2 + \cdots + q_n^2$$

$$\begin{bmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} w_{1,1}x_1 + w_{1,2}x_2 \\ w_{2,1}x_1 + w_{2,2}x_2 \end{bmatrix} = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}, \quad f(\boldsymbol{x}, W) = q_1^2 + q_2^2$$

A vectorized example

A vectorized example: $f(x, W) = ||W \cdot x||^2 = \sum_{i=1}^{n} (W \cdot x)_i^2$



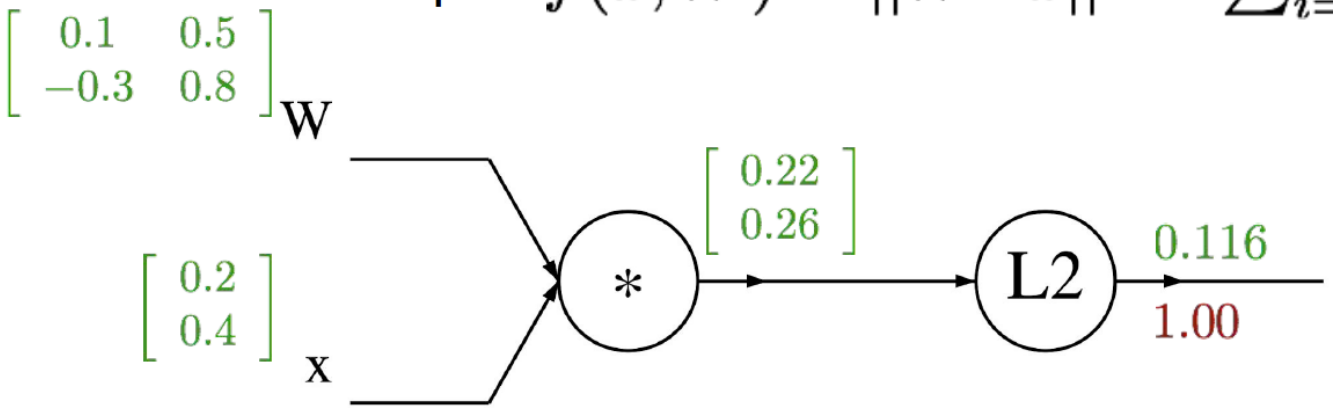
$$q = W \cdot x = \begin{pmatrix} W_{1,1}x_1 + \cdots + W_{1,n}x_n \\ \vdots \\ W_{n,1}x_1 + \cdots + W_{n,n}x_n \end{pmatrix}$$

$$f(q) = ||q||^2 = q_1^2 + \cdots + q_n^2$$

$$\begin{bmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} w_{1,1}x_1 + w_{1,2}x_2 \\ w_{2,1}x_1 + w_{2,2}x_2 \end{bmatrix} = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}, \quad f(\boldsymbol{x}, W) = q_1^2 + q_2^2$$

A vectorized example

A vectorized example: $f(x, W) = ||W \cdot x||^2 = \sum_{i=1}^{n} (W \cdot x)_i^2$



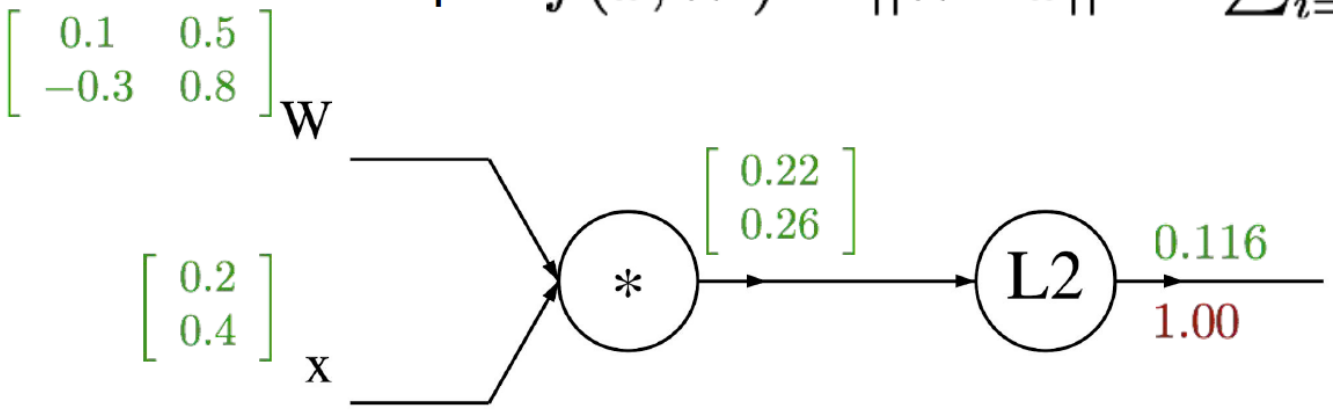
$$q = W \cdot x = \begin{pmatrix} W_{1,1}x_1 + \cdots + W_{1,n}x_n \\ \vdots \\ W_{n,1}x_1 + \cdots + W_{n,n}x_n \end{pmatrix}$$

$$f(q) = ||q||^2 = q_1^2 + \cdots + q_n^2$$

$$\begin{bmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} w_{1,1}x_1 + w_{1,2}x_2 \\ w_{2,1}x_1 + w_{2,2}x_2 \end{bmatrix} = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}, \quad f(\boldsymbol{x}, W) = q_1^2 + q_2^2$$

A vectorized example

A vectorized example: $f(x, W) = ||W \cdot x||^2 = \sum_{i=1}^{n} (W \cdot x)_i^2$



$$q = W \cdot x = \begin{pmatrix} W_{1,1}x_1 + \cdots + W_{1,n}x_n \\ \vdots \\ W_{n,1}x_1 + \cdots + W_{n,n}x_n \end{pmatrix}$$

$$f(q) = ||q||^2 = q_1^2 + \cdots + q_n^2$$

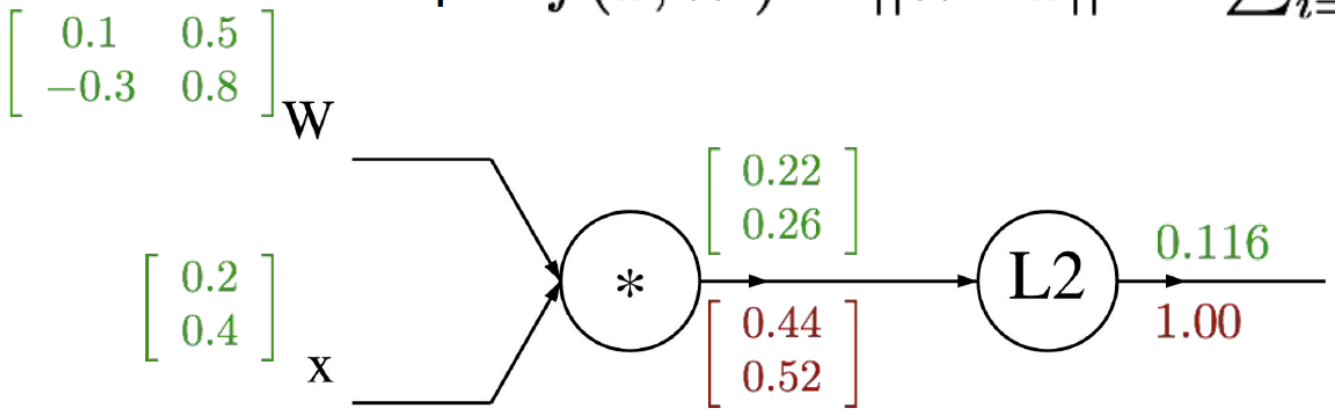
$$\frac{\partial f}{\partial q_i} = 2q_i$$

$$\boxed{\nabla_q f = 2q}$$

$$\begin{bmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} w_{1,1}x_1 + w_{1,2}x_2 \\ w_{2,1}x_1 + w_{2,2}x_2 \end{bmatrix} = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}, \quad f(\boldsymbol{x}, W) = q_1^2 + q_2^2$$

A vectorized example

A vectorized example: $f(x, W) = ||W \cdot x||^2 = \sum_{i=1}^{n} (W \cdot x)_i^2$



$$q = W \cdot x = \begin{pmatrix} W_{1,1}x_1 + \cdots + W_{1,n}x_n \\ \vdots \\ W_{n,1}x_1 + \cdots + W_{n,n}x_n \end{pmatrix}$$

$$f(q) = ||q||^2 = q_1^2 + \cdots + q_n^2$$

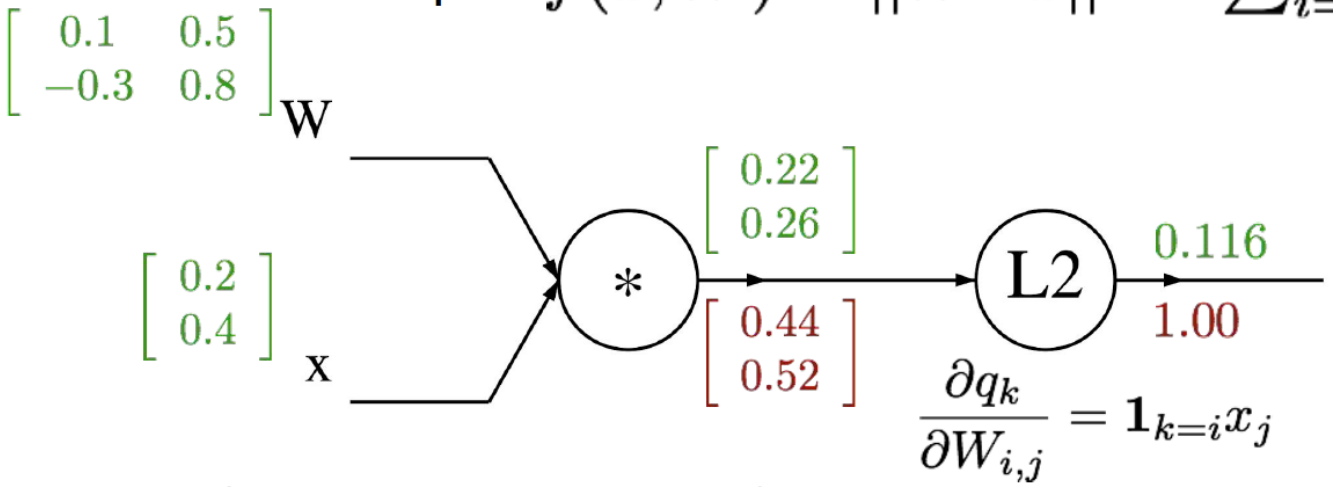
$$\frac{\partial f}{\partial q_i} = 2q_i$$

$$\boxed{\nabla_q f = 2q}$$

$$\begin{bmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} w_{1,1}x_1 + w_{1,2}x_2 \\ w_{2,1}x_1 + w_{2,2}x_2 \end{bmatrix} = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}, \quad f(\boldsymbol{x}, W) = q_1^2 + q_2^2$$

A vectorized example

A vectorized example: $f(x, W) = ||W \cdot x||^2 = \sum_{i=1}^{n} (W \cdot x)_i^2$



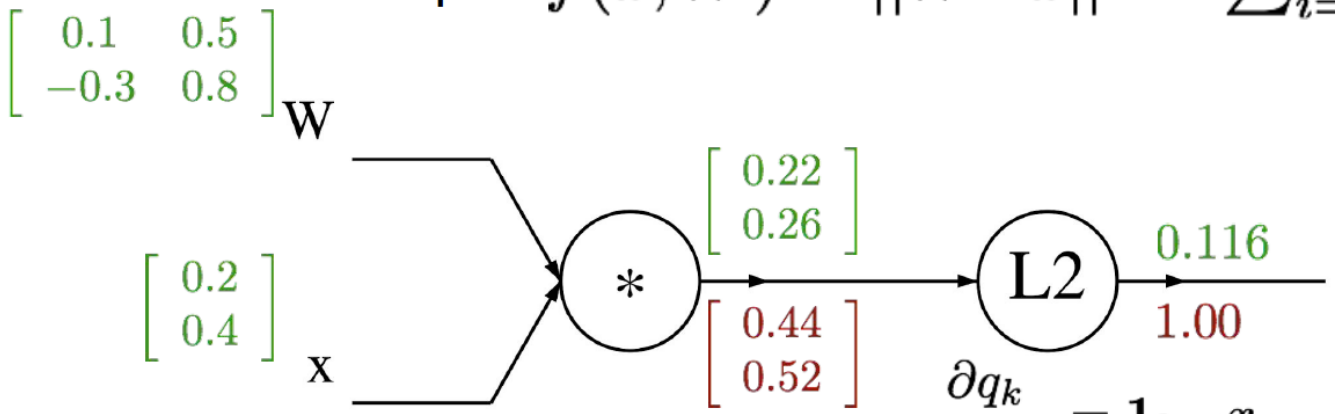
$$q = W \cdot x = \begin{pmatrix} W_{1,1}x_1 + \cdots + W_{1,n}x_n \\ \vdots \\ W_{n,1}x_1 + \cdots + W_{n,n}x_n \end{pmatrix}$$

$$f(q) = ||q||^2 = q_1^2 + \cdots + q_n^2$$

$$\begin{bmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} w_{1,1}x_1 + w_{1,2}x_2 \\ w_{2,1}x_1 + w_{2,2}x_2 \end{bmatrix} = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}, \quad f(\boldsymbol{x}, W) = q_1^2 + q_2^2$$

A vectorized example

A vectorized example: $f(x, W) = ||W \cdot x||^2 = \sum_{i=1}^{n} (W \cdot x)_i^2$



$$q = W \cdot x = \begin{pmatrix} W_{1,1}x_1 + \cdots + W_{1,n}x_n \\ \vdots \\ W_{n,1}x_1 + \cdots + W_{n,n}x_n \end{pmatrix}$$

$$f(q) = ||q||^2 = q_1^2 + \cdots + q_n^2$$

$$\frac{\partial q_k}{\partial W_{i,j}} = \mathbf{1}_{k=i} x_j$$

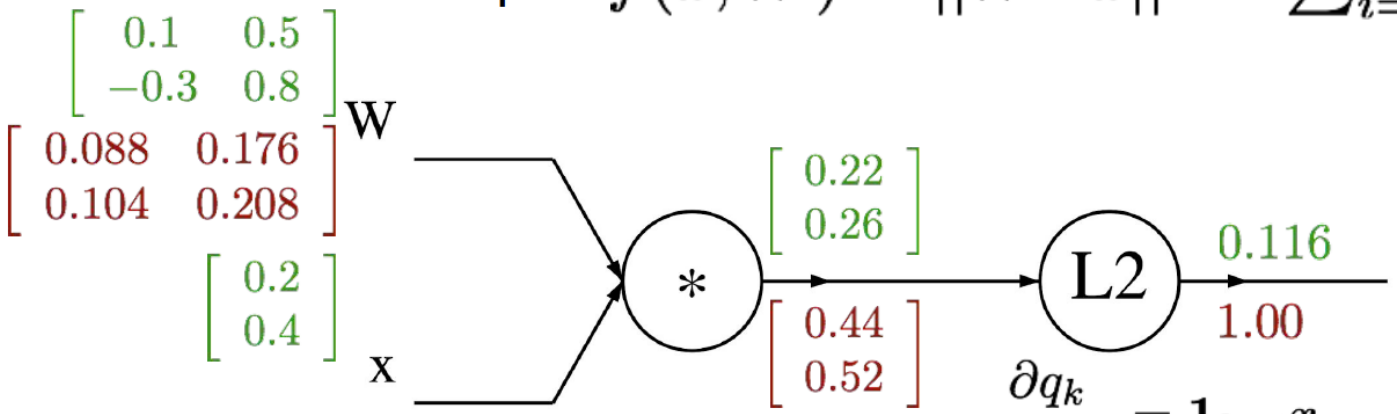
$$\begin{aligned} \frac{\partial f}{\partial W_{i,j}} &= \sum_k \frac{\partial f}{\partial q_k} \frac{\partial q_k}{\partial W_{i,j}} \\ &= \sum_k (2q_k)(\mathbf{1}_{k=i} x_j) \\ &= 2q_i x_j \end{aligned}$$

$$\begin{bmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} w_{1,1}x_1 + w_{1,2}x_2 \\ w_{2,1}x_1 + w_{2,2}x_2 \end{bmatrix} = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}, \quad f(\boldsymbol{x}, W) = q_1^2 + q_2^2$$

$$\frac{\partial f}{\partial w_{2,1}} = \frac{\partial f}{\partial q_1}\frac{\partial q_1}{\partial w_{2,1}} + \frac{\partial f}{\partial q_2}\frac{\partial q_2}{\partial w_{2,1}} = 2q_1 \times 0 + 2q_2 \times x_1 = 2 \times 0.26 \times 0.2 = 0.104$$

A vectorized example

A vectorized example: $f(x, W) = ||W \cdot x||^2 = \sum_{i=1}^{n} (W \cdot x)_i^2$



$$q = W \cdot x = \begin{pmatrix} W_{1,1}x_1 + \cdots + W_{1,n}x_n \\ \vdots \\ W_{n,1}x_1 + \cdots + W_{n,n}x_n \end{pmatrix}$$

$$f(q) = ||q||^2 = q_1^2 + \cdots + q_n^2$$

$$\frac{\partial q_k}{\partial W_{i,j}} = \mathbf{1}_{k=i} x_j$$

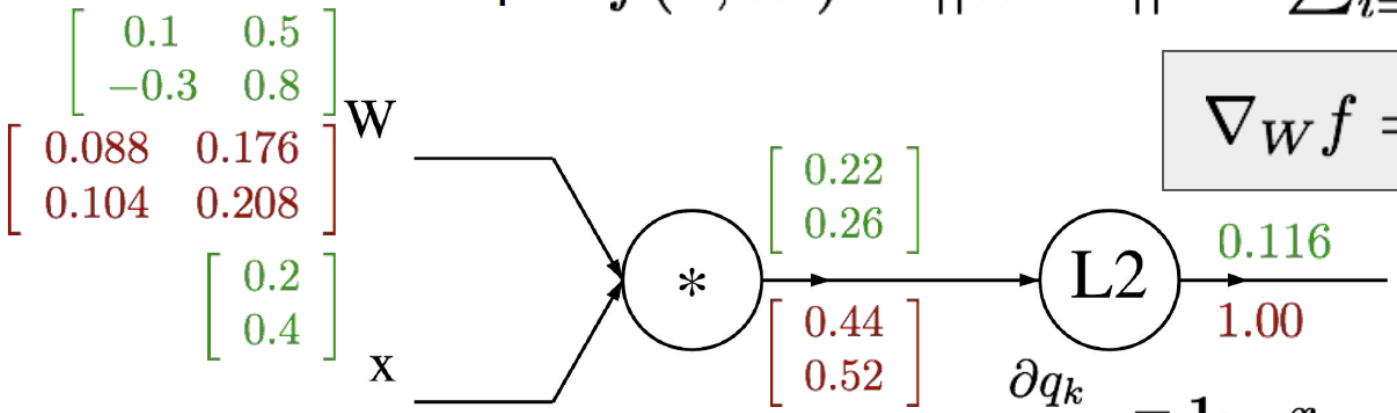
$$\begin{aligned} \frac{\partial f}{\partial W_{i,j}} &= \sum_k \frac{\partial f}{\partial q_k} \frac{\partial q_k}{\partial W_{i,j}} \\ &= \sum_k (2q_k)(\mathbf{1}_{k=i} x_j) \\ &= 2q_i x_j \end{aligned}$$

$$\begin{bmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} w_{1,1}x_1 + w_{1,2}x_2 \\ w_{2,1}x_1 + w_{2,2}x_2 \end{bmatrix} = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}, \quad f(\boldsymbol{x}, W) = q_1^2 + q_2^2$$

$$\frac{\partial f}{\partial w_{2,1}} = \frac{\partial f}{\partial q_1}\frac{\partial q_1}{\partial w_{2,1}} + \frac{\partial f}{\partial q_2}\frac{\partial q_2}{\partial w_{2,1}} = 2q_1 \times 0 + 2q_2 \times x_1 = 2 \times 0.26 \times 0.2 = 0.104$$

A vectorized example

A vectorized example: $f(x, W) = ||W \cdot x||^2 = \sum_{i=1}^{n} (W \cdot x)_i^2$



$$\nabla_W f = 2q \cdot x^T$$

$$q = W \cdot x = \begin{pmatrix} W_{1,1}x_1 + \cdots + W_{1,n}x_n \\ \vdots \\ W_{n,1}x_1 + \cdots + W_{n,n}x_n \end{pmatrix}$$

$$f(q) = ||q||^2 = q_1^2 + \cdots + q_n^2$$

$$\frac{\partial q_k}{\partial W_{i,j}} = \mathbf{1}_{k=i} x_j$$

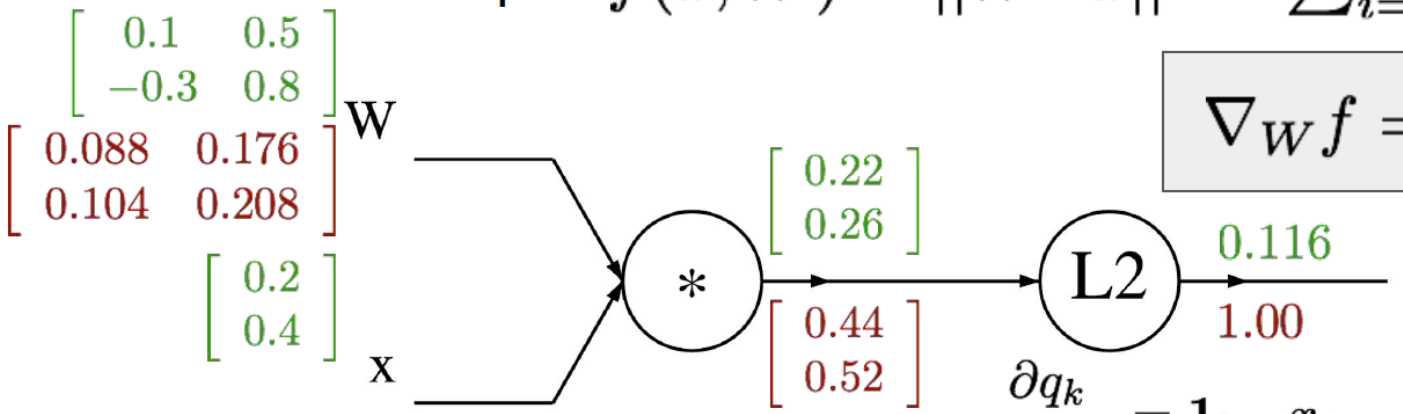
$$\begin{aligned} \frac{\partial f}{\partial W_{i,j}} &= \sum_k \frac{\partial f}{\partial q_k} \frac{\partial q_k}{\partial W_{i,j}} \\ &= \sum_k (2q_k)(\mathbf{1}_{k=i} x_j) \\ &= 2q_i x_j \end{aligned}$$

$$\begin{bmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} w_{1,1}x_1 + w_{1,2}x_2 \\ w_{2,1}x_1 + w_{2,2}x_2 \end{bmatrix} = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}, \quad f(\boldsymbol{x}, W) = q_1^2 + q_2^2$$

$$\frac{\partial f}{\partial w_{2,1}} = \frac{\partial f}{\partial q_1}\frac{\partial q_1}{\partial w_{2,1}} + \frac{\partial f}{\partial q_2}\frac{\partial q_2}{\partial w_{2,1}} = 2q_1 \times 0 + 2q_2 \times x_1 = 2 \times 0.26 \times 0.2 = 0.104$$

A vectorized example

A vectorized example: $f(x, W) = ||W \cdot x||^2 = \sum_{i=1}^{n} (W \cdot x)_i^2$



$$\nabla_W f = 2q \cdot x^T$$

Always check: The gradient with respect to a variable should have the same shape as the variable

$$q = W \cdot x = \begin{pmatrix} W_{1,1}x_1 + \cdots + W_{1,n}x_n \\ \vdots \\ W_{n,1}x_1 + \cdots + W_{n,n}x_n \end{pmatrix}$$

$$f(q) = ||q||^2 = q_1^2 + \cdots + q_n^2$$

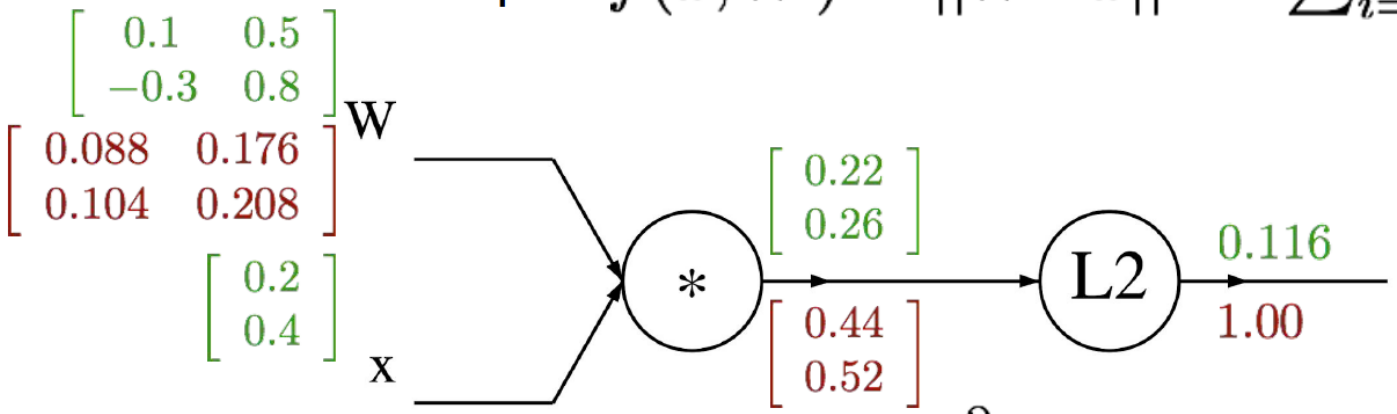
$$\frac{\partial q_k}{\partial W_{i,j}} = \mathbf{1}_{k=i} x_j$$

$$\begin{aligned} \frac{\partial f}{\partial W_{i,j}} &= \sum_k \frac{\partial f}{\partial q_k} \frac{\partial q_k}{\partial W_{i,j}} \\ &= \sum_k^k (2q_k)(\mathbf{1}_{k=i} x_j) \\ &= 2q_i x_j \end{aligned}$$

$$\begin{bmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} w_{1,1}x_1 + w_{1,2}x_2 \\ w_{2,1}x_1 + w_{2,2}x_2 \end{bmatrix} = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}, \quad f(\boldsymbol{x}, W) = q_1^2 + q_2^2$$

A vectorized example

A vectorized example: $f(x, W) = ||W \cdot x||^2 = \sum_{i=1}^{n} (W \cdot x)_i^2$



$$\frac{\partial q_k}{\partial x_i} = W_{k,i}$$

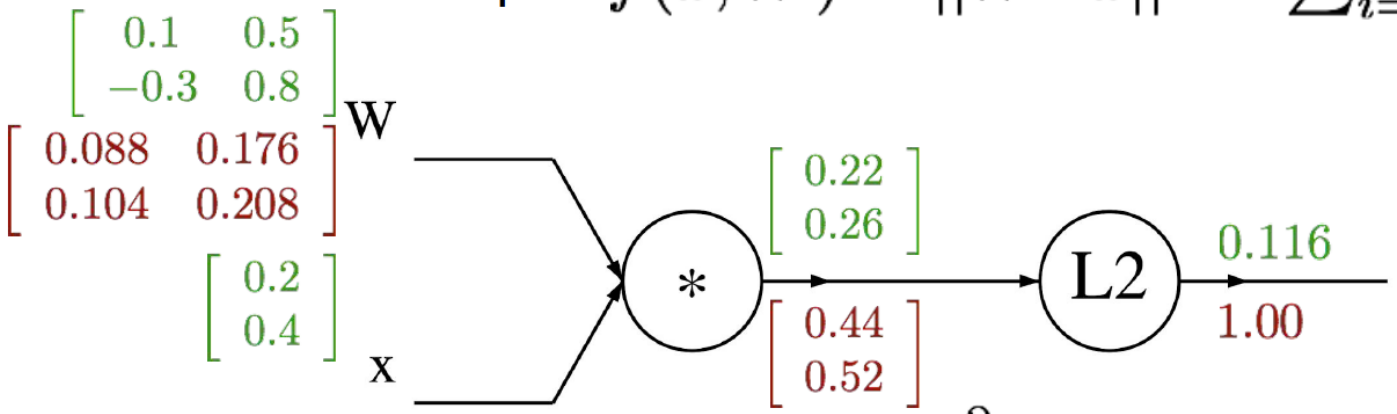
$$q = W \cdot x = \begin{pmatrix} W_{1,1}x_1 + \cdots + W_{1,n}x_n \\ \vdots \\ W_{n,1}x_1 + \cdots + W_{n,n}x_n \end{pmatrix}$$

$$f(q) = ||q||^2 = q_1^2 + \cdots + q_n^2$$

$$\begin{bmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} w_{1,1}x_1 + w_{1,2}x_2 \\ w_{2,1}x_1 + w_{2,2}x_2 \end{bmatrix} = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}, \quad f(\boldsymbol{x}, W) = q_1^2 + q_2^2$$

A vectorized example

A vectorized example: $f(x, W) = ||W \cdot x||^2 = \sum_{i=1}^{n} (W \cdot x)_i^2$



$$q = W \cdot x = \begin{pmatrix} W_{1,1}x_1 + \cdots + W_{1,n}x_n \\ \vdots \\ W_{n,1}x_1 + \cdots + W_{n,n}x_n \end{pmatrix}$$

$$f(q) = ||q||^2 = q_1^2 + \cdots + q_n^2$$

$$\frac{\partial q_k}{\partial x_i} = W_{k,i}$$

$$\frac{\partial f}{\partial x_i} = \sum_k \frac{\partial f}{\partial q_k}\frac{\partial q_k}{\partial x_i} = \sum_k 2q_k W_{k,i}$$

$$\begin{bmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} w_{1,1}x_1 + w_{1,2}x_2 \\ w_{2,1}x_1 + w_{2,2}x_2 \end{bmatrix} = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}, \quad f(\boldsymbol{x}, W) = q_1^2 + q_2^2$$

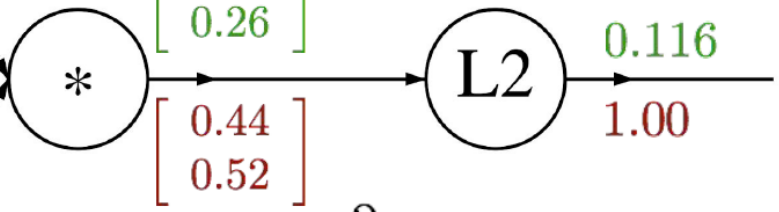
$$\frac{\partial f}{\partial x_1} = \frac{\partial f}{\partial q_1}\frac{\partial q_1}{\partial x_1} + \frac{\partial f}{\partial q_2}\frac{\partial q_2}{\partial x_1} = 2q_1 \times w_{1,1} + 2q_2 \times w_{2,1} = 0.44 \times 0.1 + 0.52 \times (-03) = -0.112$$

A vectorized example

A vectorized example: $f(x, W) = ||W \cdot x||^2 = \sum_{i=1}^{n} (W \cdot x)_i^2$

$$\begin{bmatrix} 0.1 & 0.5 \\ -0.3 & 0.8 \end{bmatrix}$$
$$\begin{bmatrix} 0.088 & 0.176 \\ 0.104 & 0.208 \end{bmatrix}$$
W

$$\begin{bmatrix} 0.2 \\ 0.4 \end{bmatrix}$$
$$\begin{bmatrix} -0.112 \\ 0.636 \end{bmatrix}$$
x



$$\begin{bmatrix} 0.22 \\ 0.26 \end{bmatrix}$$
$$\begin{bmatrix} 0.44 \\ 0.52 \end{bmatrix}$$
0.116
1.00

$$\nabla_x f = 2W^T \cdot q$$

$$q = W \cdot x = \begin{pmatrix} W_{1,1}x_1 + \cdots + W_{1,n}x_n \\ \vdots \\ W_{n,1}x_1 + \cdots + W_{n,n}x_n \end{pmatrix}$$

$$f(q) = ||q||^2 = q_1^2 + \cdots + q_n^2$$

$$\frac{\partial q_k}{\partial x_i} = W_{k,i}$$

$$\frac{\partial f}{\partial x_i} = \sum_k \frac{\partial f}{\partial q_k} \frac{\partial q_k}{\partial x_i} = \sum_k 2q_k W_{k,i}$$

$$\begin{bmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} w_{1,1}x_1 + w_{1,2}x_2 \\ w_{2,1}x_1 + w_{2,2}x_2 \end{bmatrix} = \begin{bmatrix} q_1 \\ q_2 \end{bmatrix}, \quad f(\boldsymbol{x}, W) = q_1^2 + q_2^2$$

$$\frac{\partial f}{\partial x_1} = \frac{\partial f}{\partial q_1}\frac{\partial q_1}{\partial x_1} + \frac{\partial f}{\partial q_2}\frac{\partial q_2}{\partial x_1} = 2q_1 \times w_{1,1} + 2q_2 \times w_{2,1} = 0.44 \times 0.1 + 0.52 \times (-03) = -0.112$$

:)


dont worry about it if you don't
understand

Epoch

- Epoch
  - An epoch means training the neural network with all the training data for one cycle.
  - In an epoch, we use all of the data exactly once.
  - A forward pass and a backward pass together are counted as one pass: An epoch is made up of one or more batches, where we use a part of the dataset to train the neural network.

Gradient Descent

- Stochastic Gradient Descent
  - Stochastic gradient descent is a variation of the gradient descent algorithm that calculates the error and updates the model for *each example* in the training dataset.
    - The frequent updates immediately give an insight into the performance of the model and the rate of improvement.
    - Avoid local minima
    - The frequent updates can result in a noisy gradient signal, which may cause the model parameters and in turn the model error to jump around.
    - Updating the model so frequently is more computationally expensive than other configurations of gradient descent, taking significantly longer to train models on large datasets.

Gradient Descent

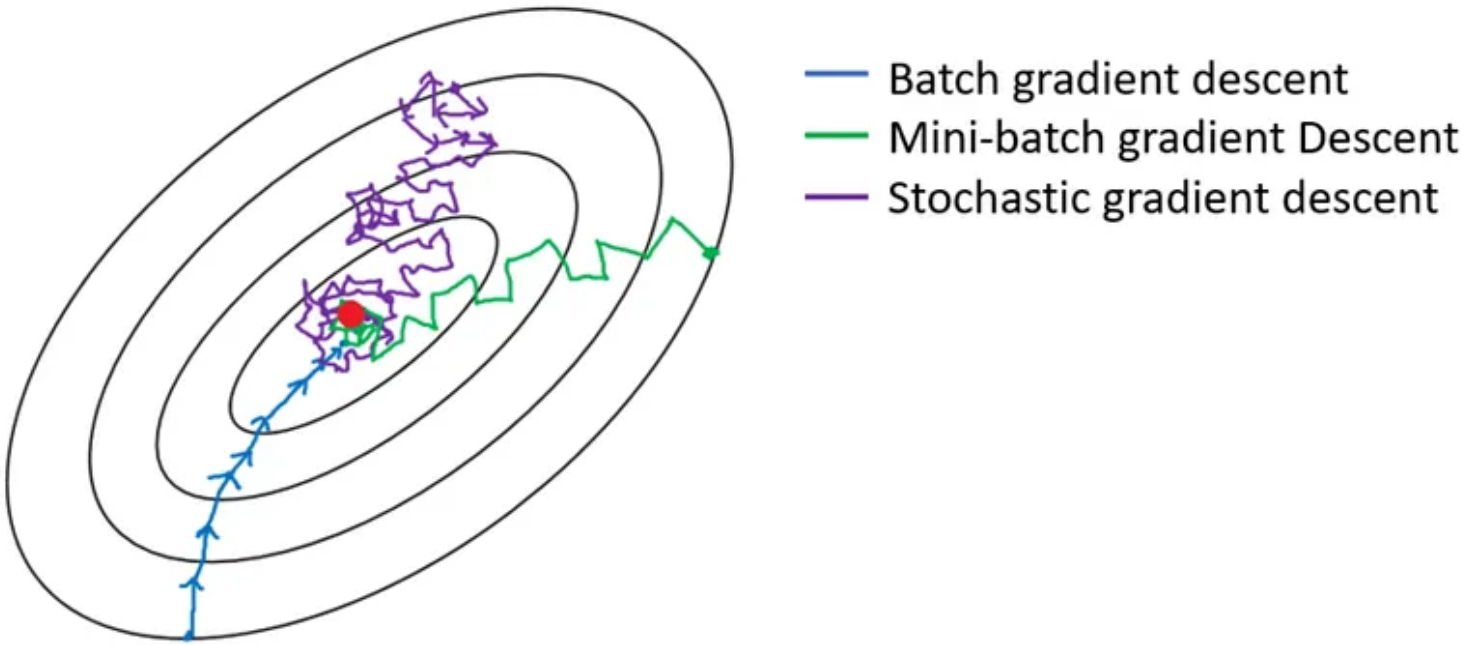
- Stochastic Gradient Descent
  - Stochastic gradient descent is a variation of the gradient descent algorithm that calculates the error and updates the model for *each example* in the training dataset.
    - The frequent updates immediately give an insight into the performance of the model and the rate of improvement.
    - Avoid local minima
    - The frequent updates can result in a noisy gradient signal, which may cause the model parameters and in turn the model error to jump around.
    - Updating the model so frequently is more computationally expensive than other configurations of gradient descent, taking significantly longer to train models on large datasets.
- Batch (Full-batch) Gradient Descent
  - Full-batch gradient descent is a variation of the gradient descent algorithm that calculates the error for each example in the training dataset, but only updates the model after all training examples have been evaluated.
    - The decreased update frequency results in a more stable error gradient and may result in a more stable convergence on some problems.
    - Parallel processing based implementations.
    - It requires the entire training dataset in memory and available to the algorithm.

Gradient Descent

- Stochastic Gradient Descent
  - Stochastic gradient descent is a variation of the gradient descent algorithm that calculates the error and updates the model for *each example* in the training dataset.
    - The frequent updates immediately give an insight into the performance of the model and the rate of improvement.
    - Avoid local minima
    - The frequent updates can result in a noisy gradient signal, which may cause the model parameters and in turn the model error to jump around.
    - Updating the model so frequently is more computationally expensive than other configurations of gradient descent, taking significantly longer to train models on large datasets.
- Batch (Full-batch) Gradient Descent
  - Full-batch gradient descent is a variation of the gradient descent algorithm that calculates the error for each example in the training dataset, but only updates the model after all training examples have been evaluated.
    - The decreased update frequency results in a more stable error gradient and may result in a more stable convergence on some problems.
    - Parallel processing based implementations.
    - It requires the entire training dataset in memory and available to the algorithm.
- Mini-Batch Gradient Descent
  - Mini-batch gradient descent is a variation of the gradient descent algorithm that splits the training dataset into small batches that are used to calculate model error and update model weights.
    - Mini-batch gradient descent seeks to find a balance between the robustness of stochastic gradient descent and the efficiency of batch gradient descent. It is the most common implementation of gradient descent used in the field of deep learning.
    - 32 / 64 / 128 are common sizes for a mini-batch
    - It is a new hyper-parameter!

Source

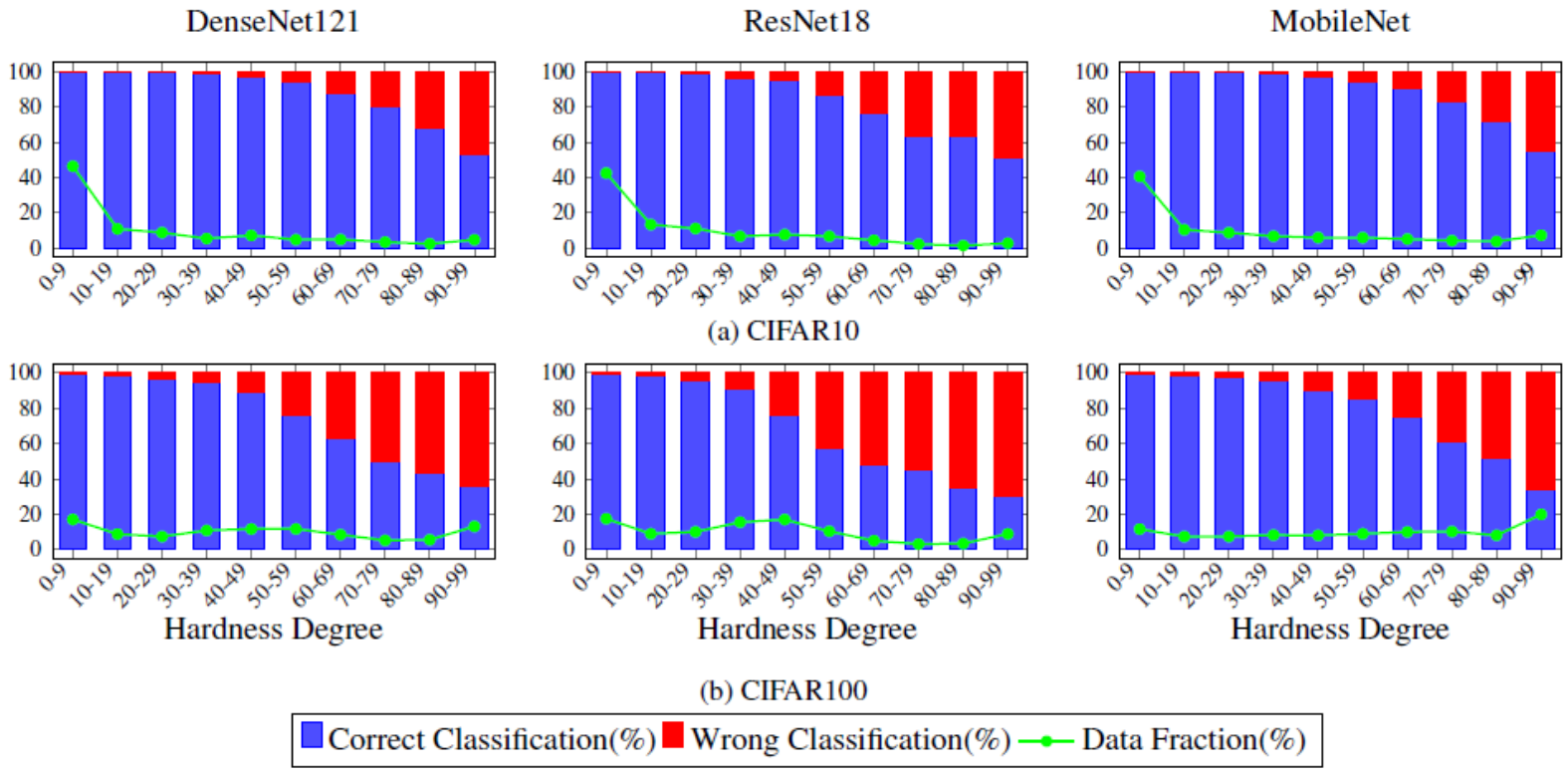
Gradient Descent



Source

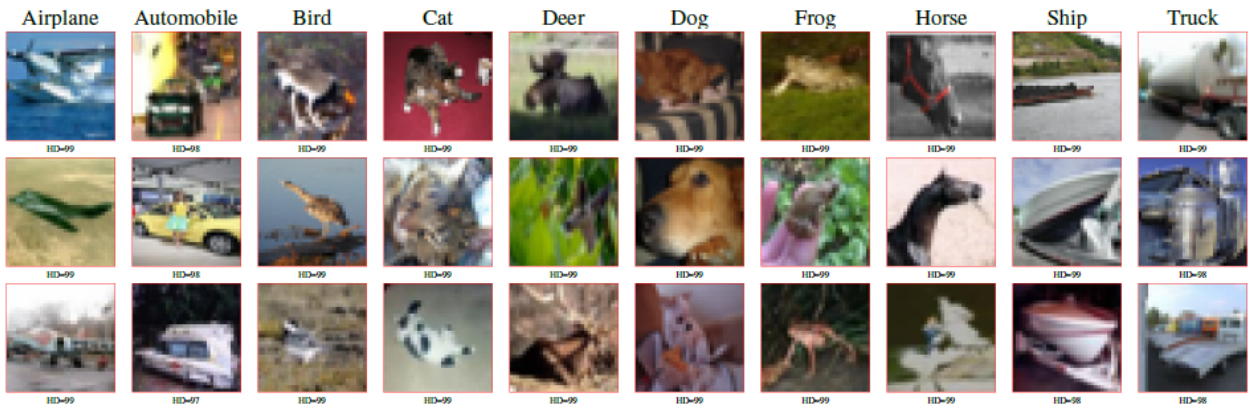
Hardness of Samples

We expect the predicted label of easy samples to converge sooner than the predicted label of hard samples during the course of training. Hence, the hardness degree of a sample depends on the predicted label convergence speed for that sample during training.

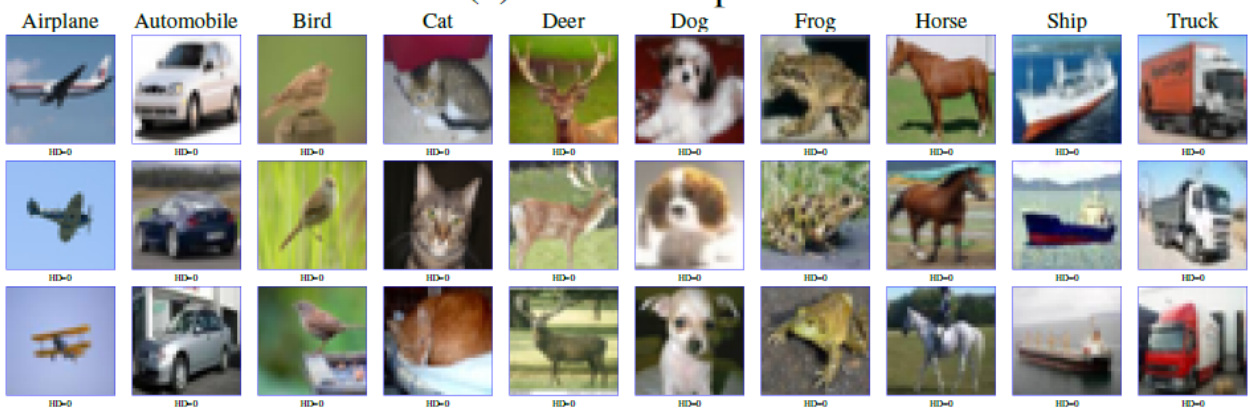


(Sadeghzadeh, 2021)

Visualization of CIFAR10 test samples



(a) Hard Samples

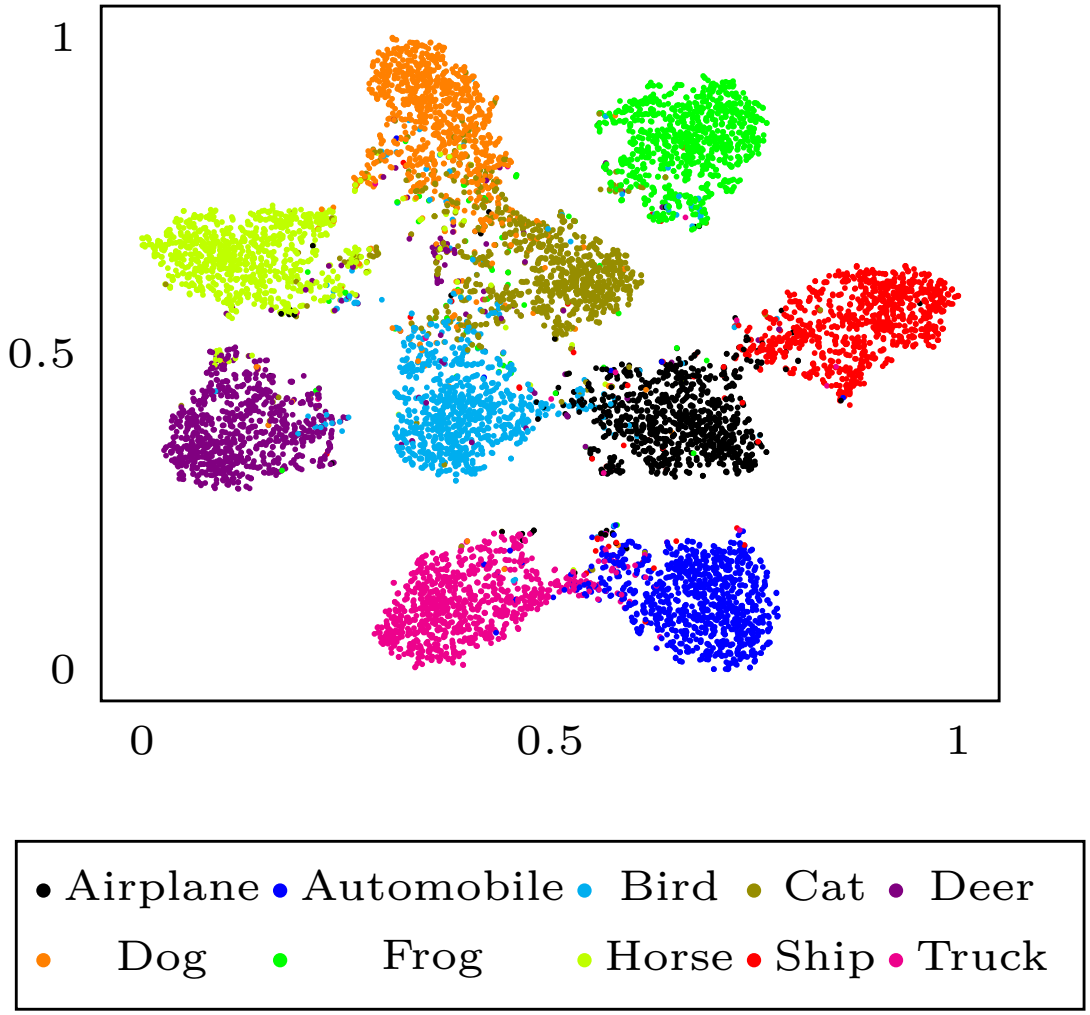


(b) Easy Samples

Fig. 2: Some of the easiest and hardest CIFAR10 test samples based on the hardness degree for each class.

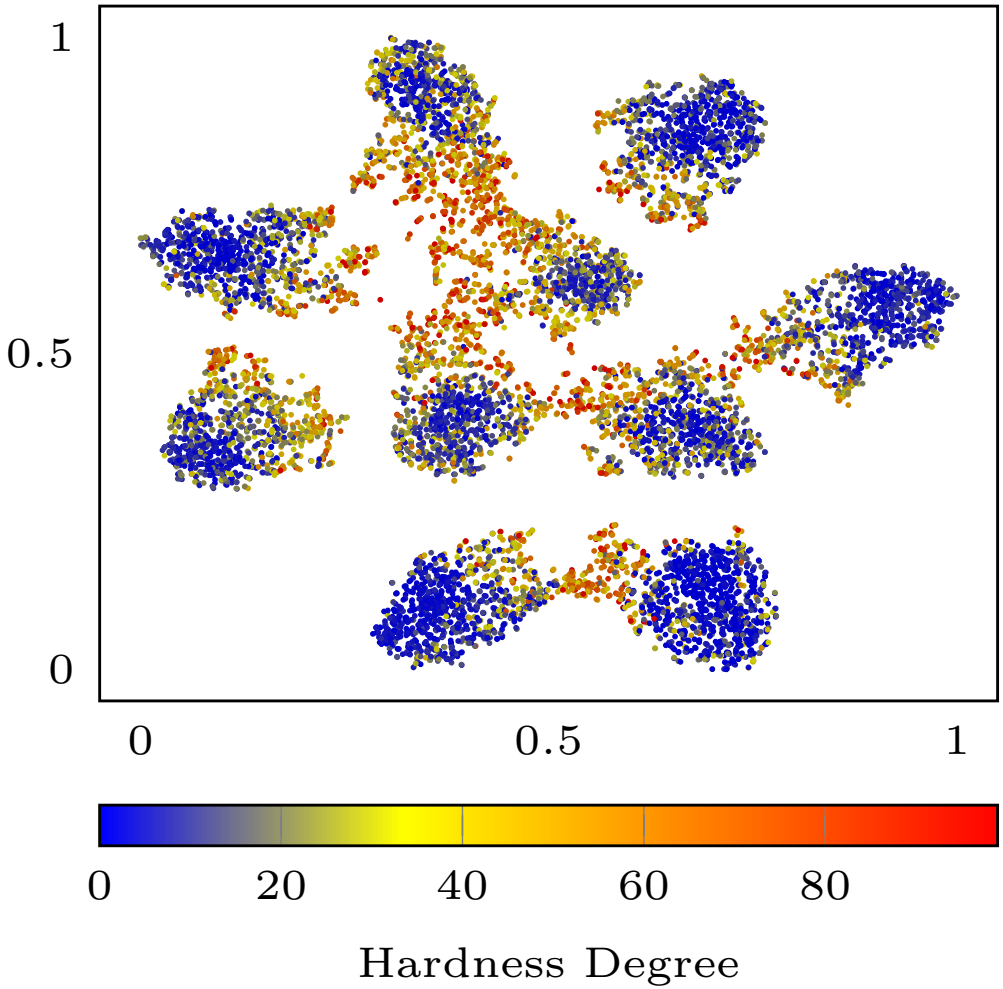
(Sadeghzadeh, 2021)

Visualization of CIFAR10 test samples



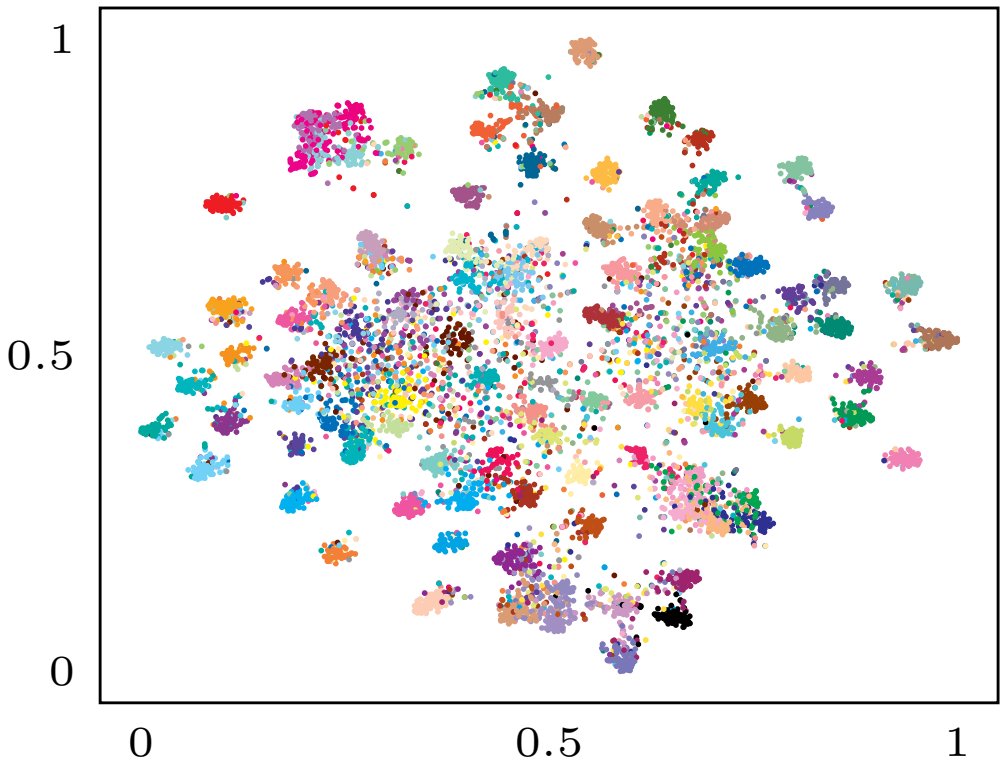
(Sadeghzadeh, 2021)

Hardness of CIFAR10 test samples



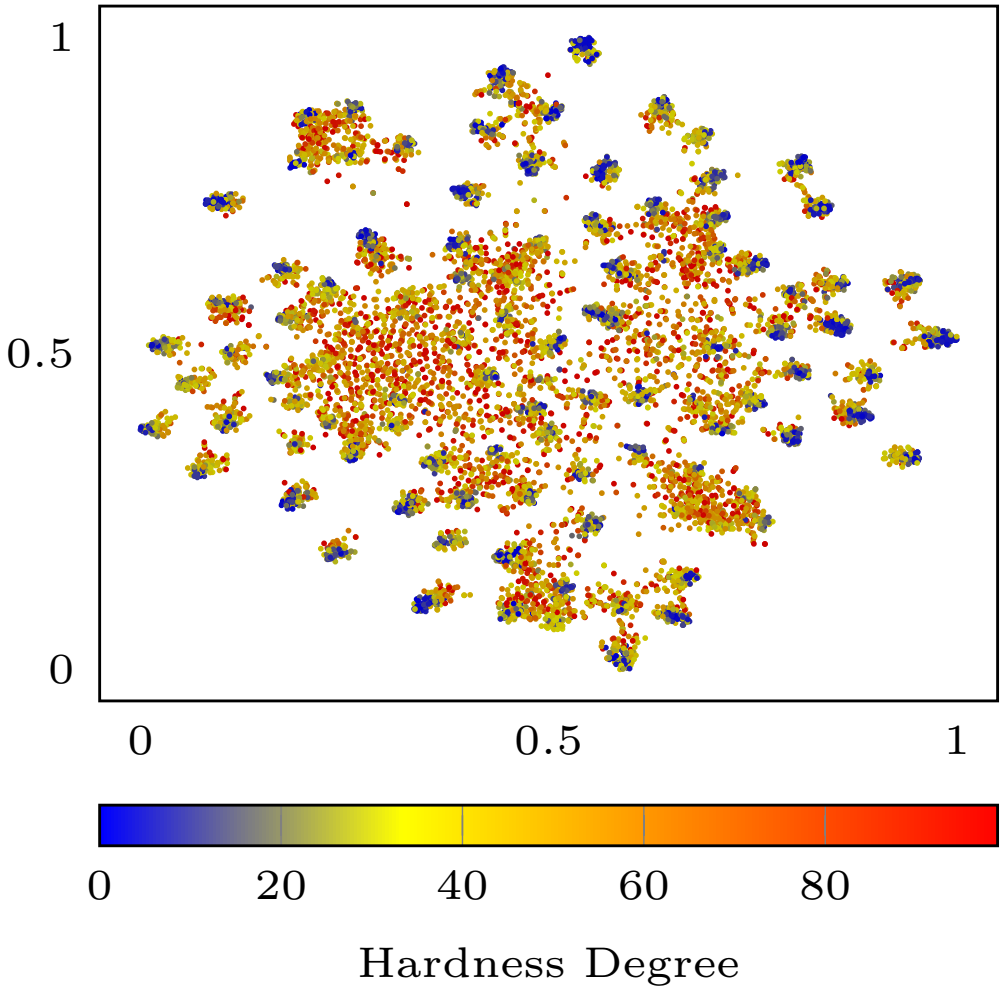
(Sadeghzadeh, 2021)

Visualization of CIFAR100 test samples



- Each color is for samples of one class.

Hardness of CIFAR100 test samples



(Sadeghzadeh, 2021)

Validation

Hyperparameters

- Examples
  - Weight initialization
  - Number of steps
  - Learning rate
  - Batch size
  - The architecture of neural network

Hyperparameters

- Examples
  - Weight initialization
  - Number of steps
  - Learning rate
  - Batch size
  - The architecture of neural network

- **hyperparameters**: choices about our learning algorithm that we don't learn from the training data; instead we set them at the start of the learning process
  - **Very problem-dependent. In general need to try them all and see what works best for our data / task.**

Hyperparameters

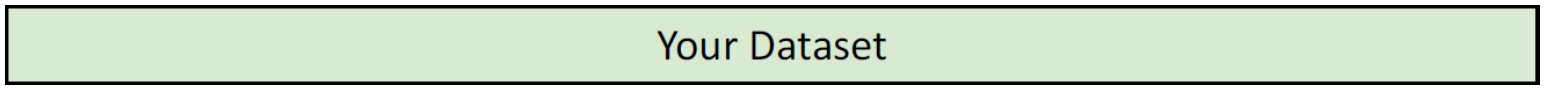
- Examples
  - Weight initialization
  - Number of steps
  - Learning rate
  - Batch size
  - The architecture of neural network

- **hyperparameters**: choices about our learning algorithm that we don't learn from the training data; instead we set them at the start of the learning process
  - **Very problem-dependent. In general need to try them all and see what works best for our data / task.**

- Test error estimation using **validation error** to avoid overfitting

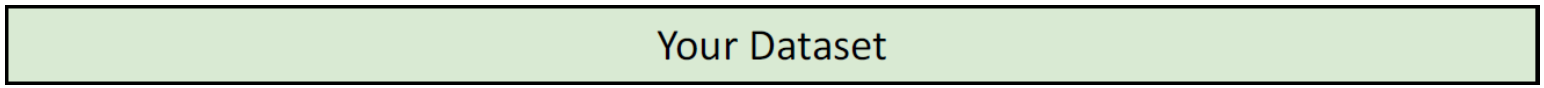
Setting hyperparameters

- Idea 1: Choose hyperparameters that work best on the data
  - Overfitting



Setting hyperparameters

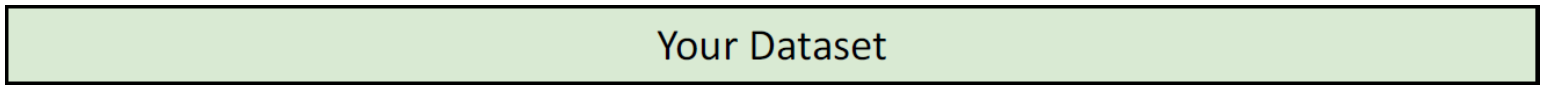
- Idea 1: Choose hyperparameters that work best on the data
  - Overfitting



- Idea 2: Split data into train and test, choose hyperparameters that work best on test data
  - No idea how algorithm will perform on new data

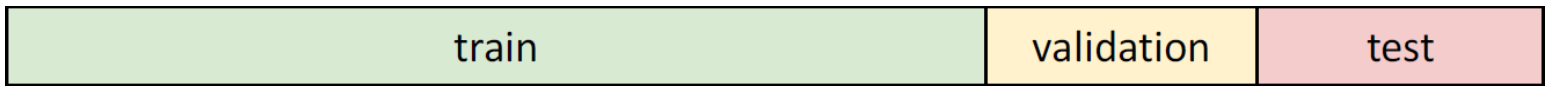
Setting hyperparameters

- Idea 1: Choose hyperparameters that work best on the data
  - Overfitting



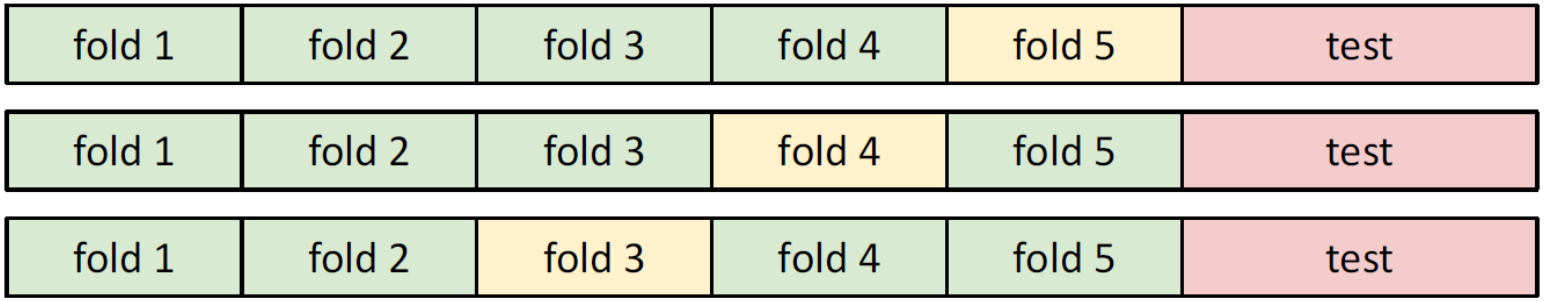
- Idea 2: Split data into train and test, choose hyperparameters that work best on test data
  - No idea how algorithm will perform on new data

- Idea 3: Split data into train, val, and test; choose hyperparameters on val and evaluate on test
  - Better!



Cross validation

- Idea 4: Cross-Validation: Split data into folds, try each fold as validation and average the results
  - Useful for small datasets, but (unfortunately) not used too frequently in deep learning

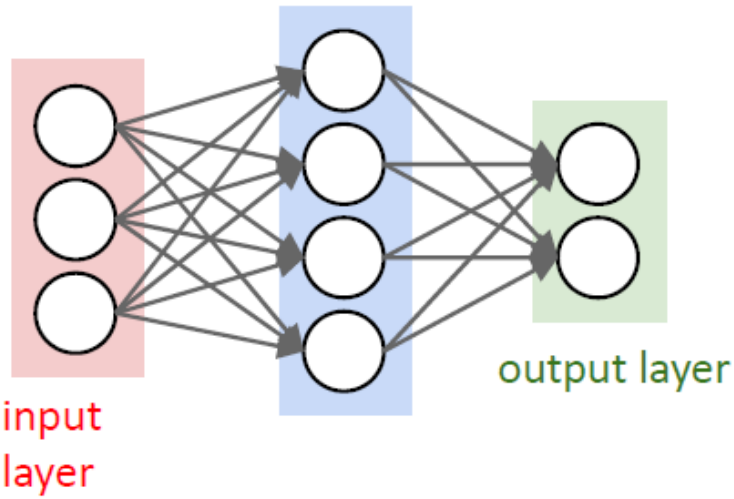


Convolutional Neural Networks

The model so far

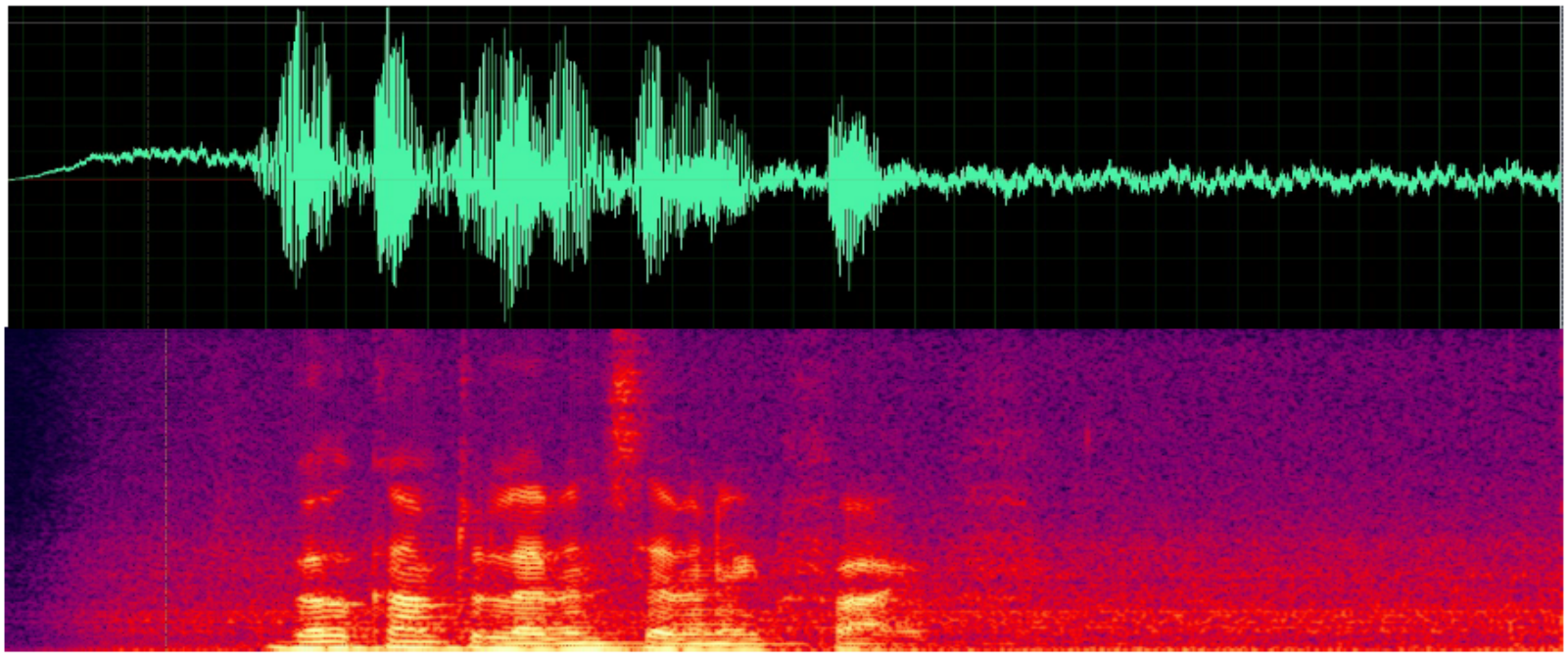
- Can recognize patterns in data
  - E.g. digits
  - Or any other vector data

Or, more generally
a vector input



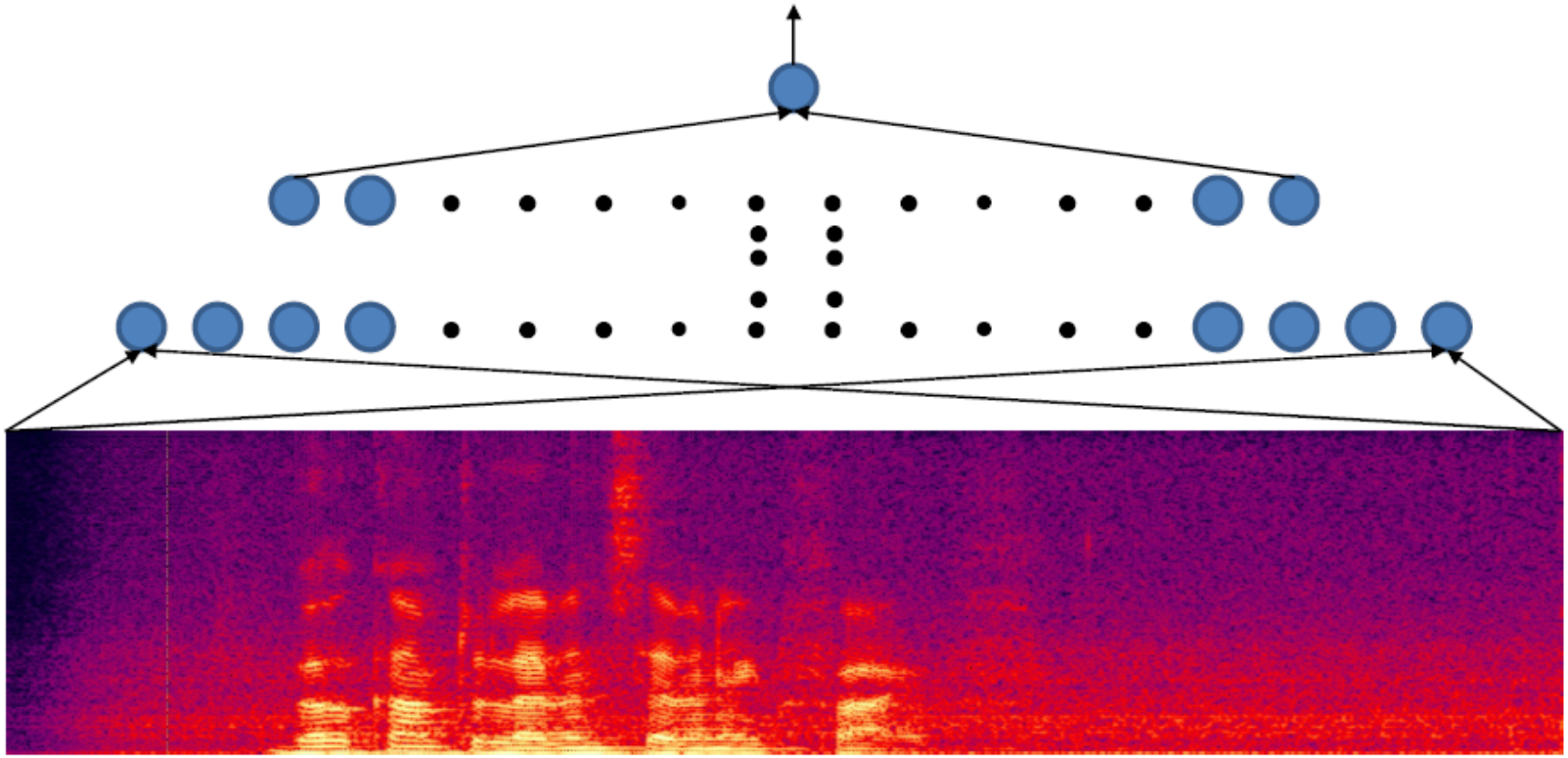
A new problem

- Does this signal contain the word “Welcome”?
- Compose an MLP for this problem.
  - Assuming all recordings are exactly the same length.



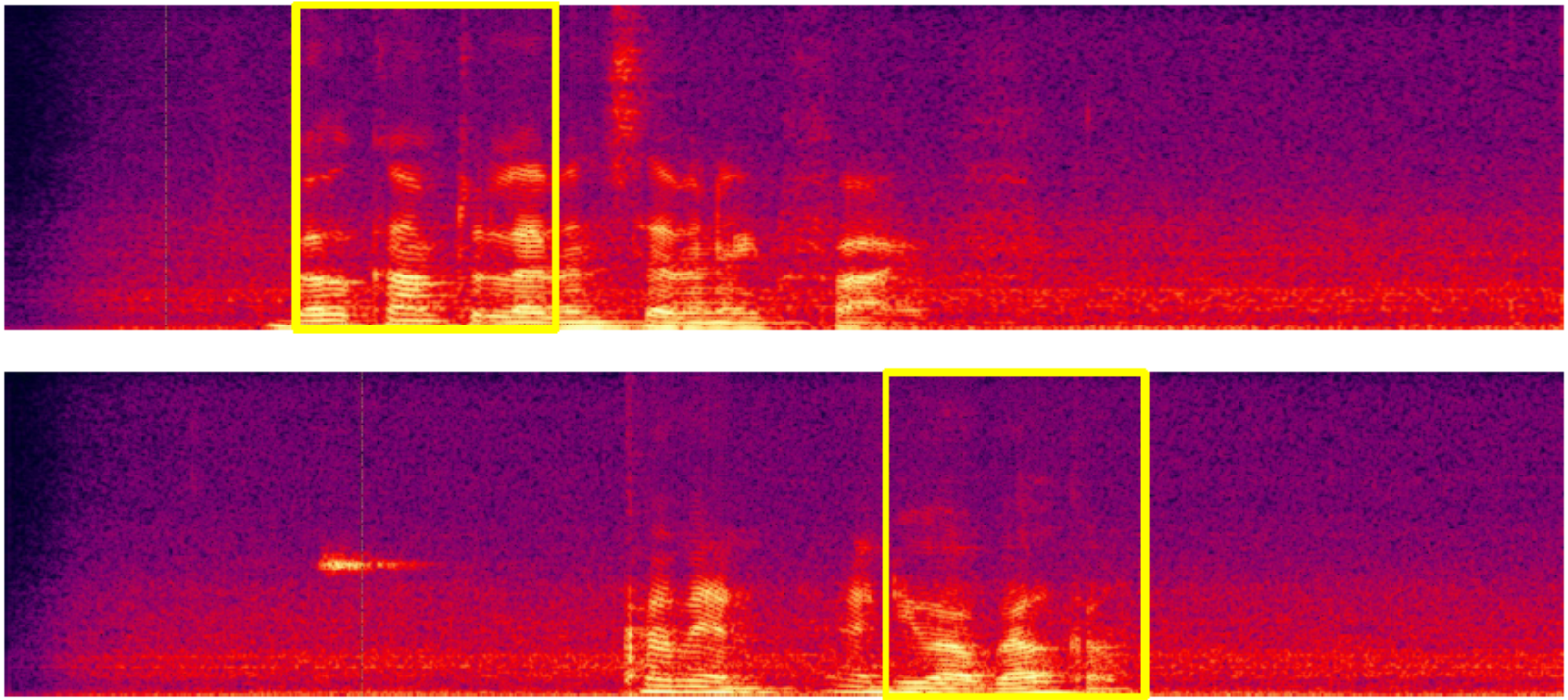
Finding a Welcome

- Trivial solution: Train an MLP for the entire recording



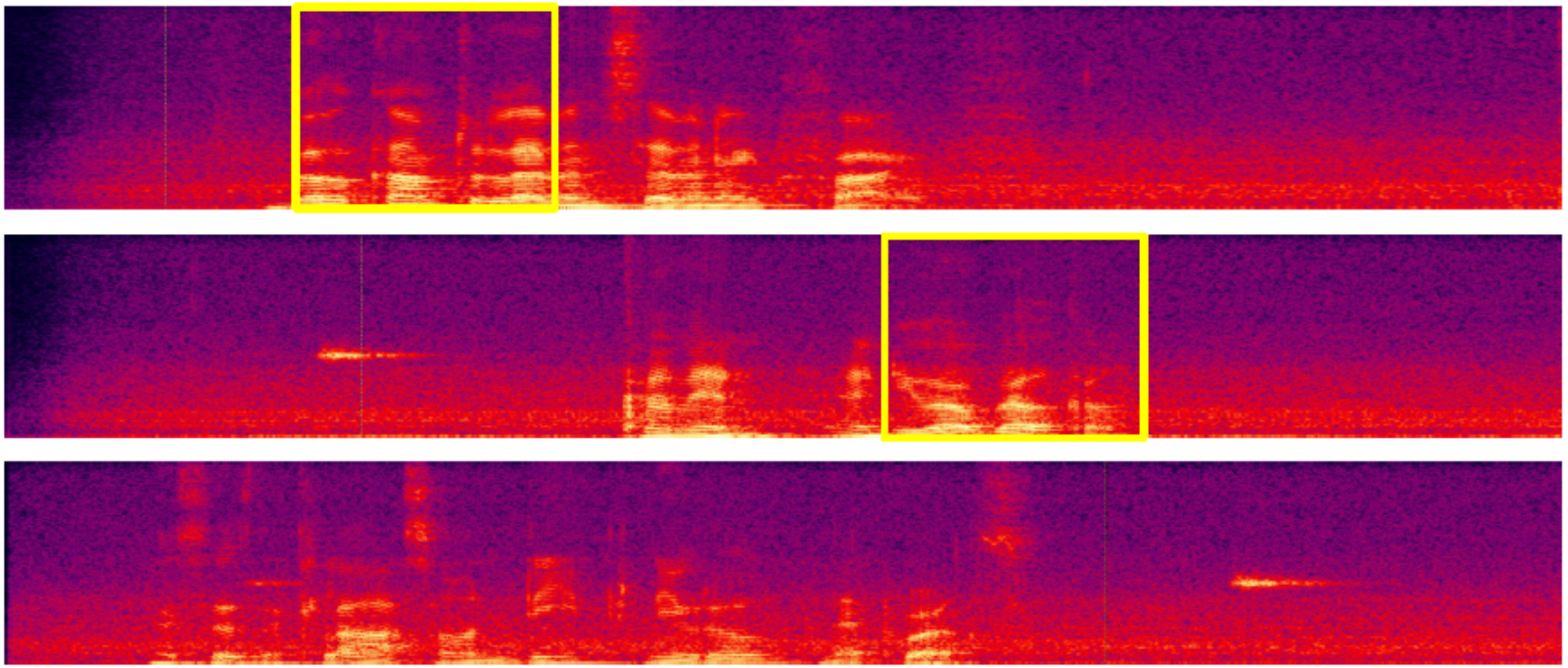
Finding a Welcome

- Problem with trivial solution: Network that finds a “welcome” in the top recording will not find it in the lower one
  - Unless trained with both
  - Will require a very large network and a large amount of training data to cover every case



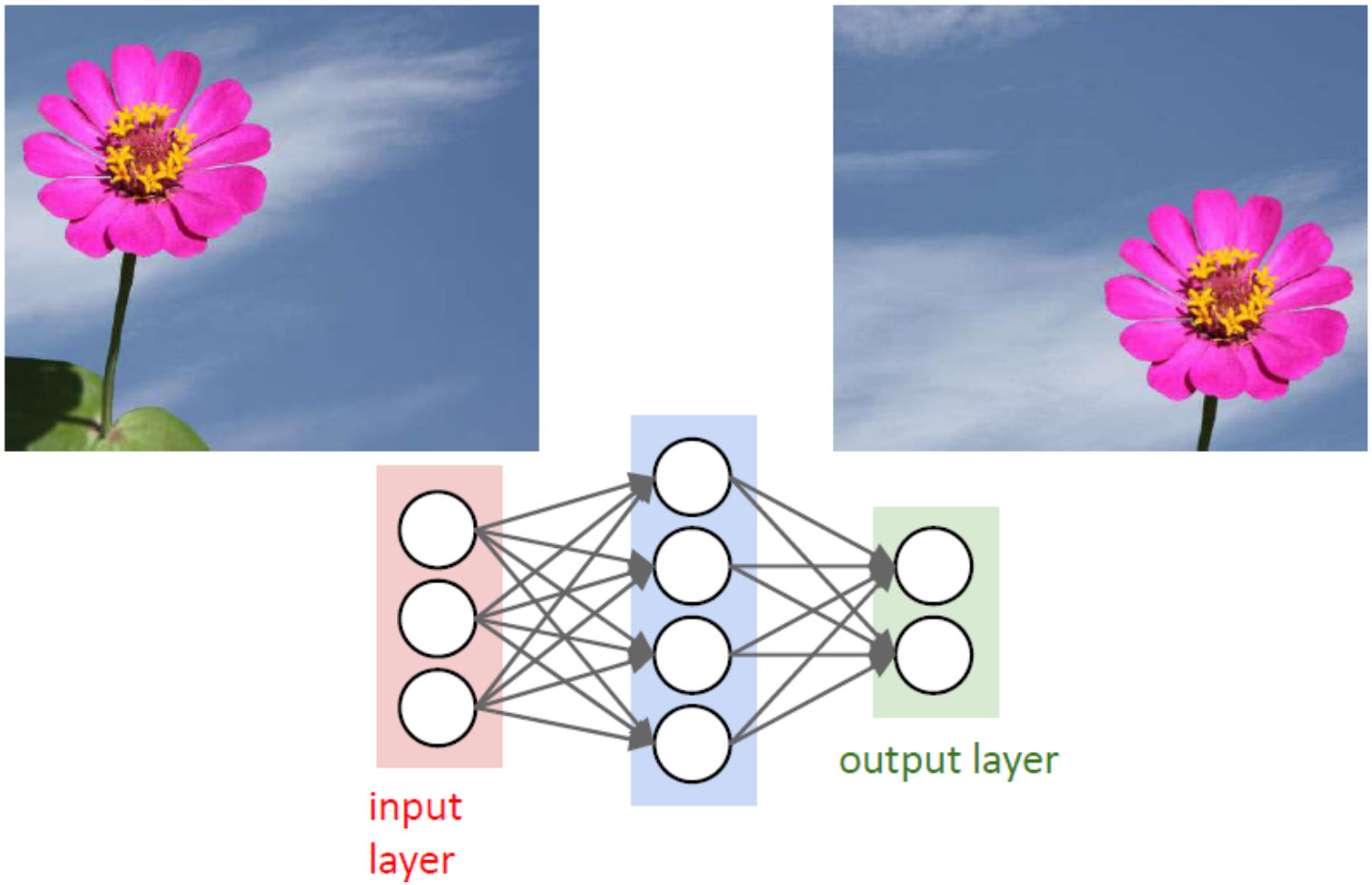
Finding a Welcome

- Need a simple network that will fire regardless of the location of “Welcome”
  - and not fire when there is none



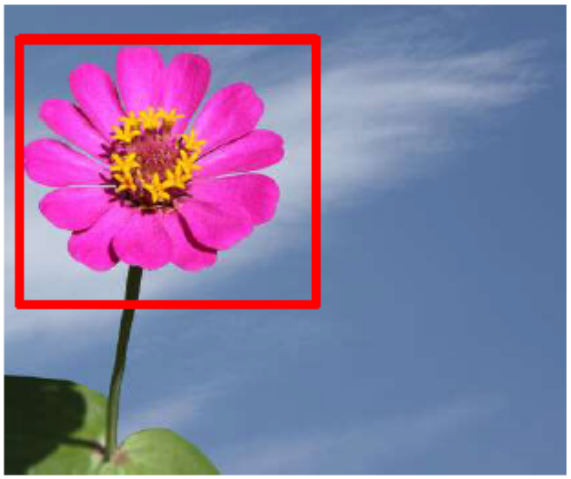
A problem

- Will an MLP that recognizes the left image as a flower also recognize the one on the right as a flower?



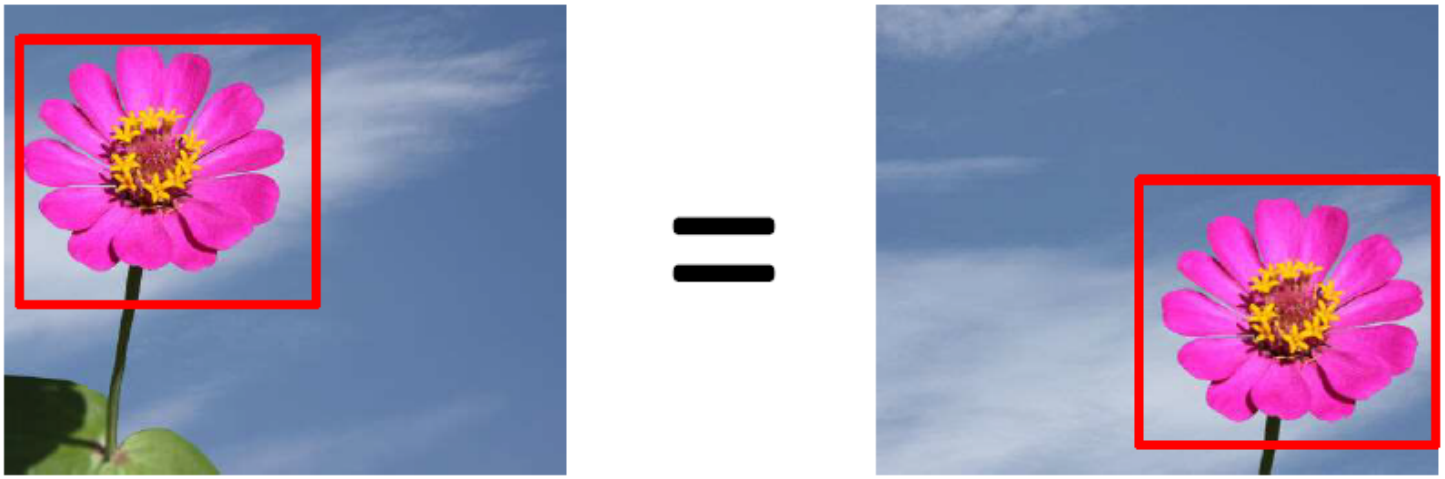
A problem

- Need a network that will “fire” regardless of the precise location of the target object



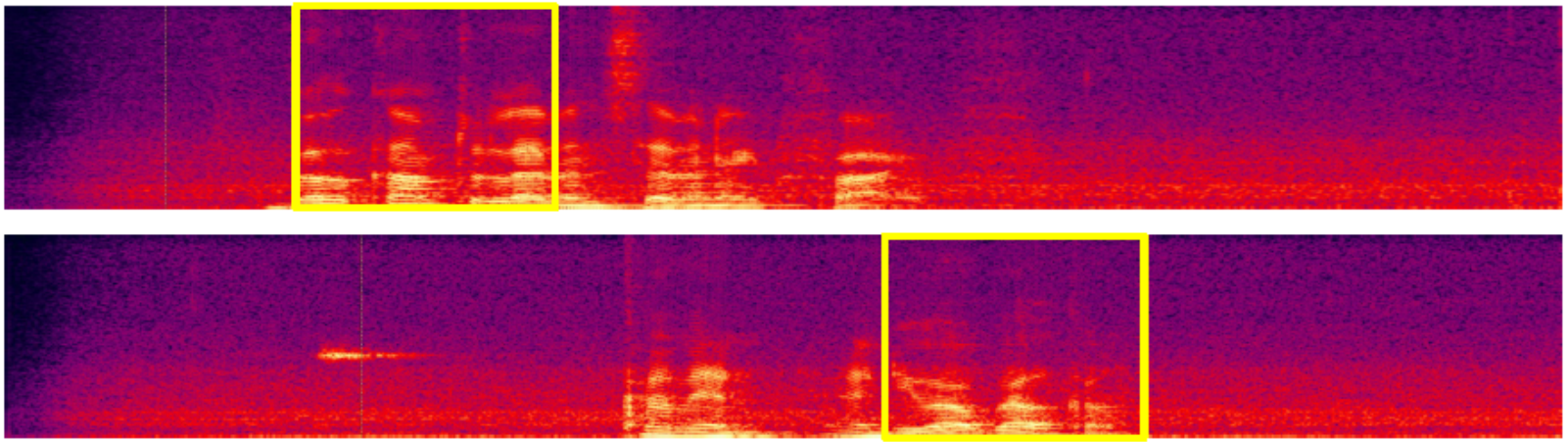
A problem

- In many problems the location of a pattern is not important
  - Only the presence of the pattern
- Conventional MLPs are sensitive to the location of the pattern
  - Moving it by one component results in an entirely different input that the MLP wont recognize
- Requirement: Network must be **shift invariant**



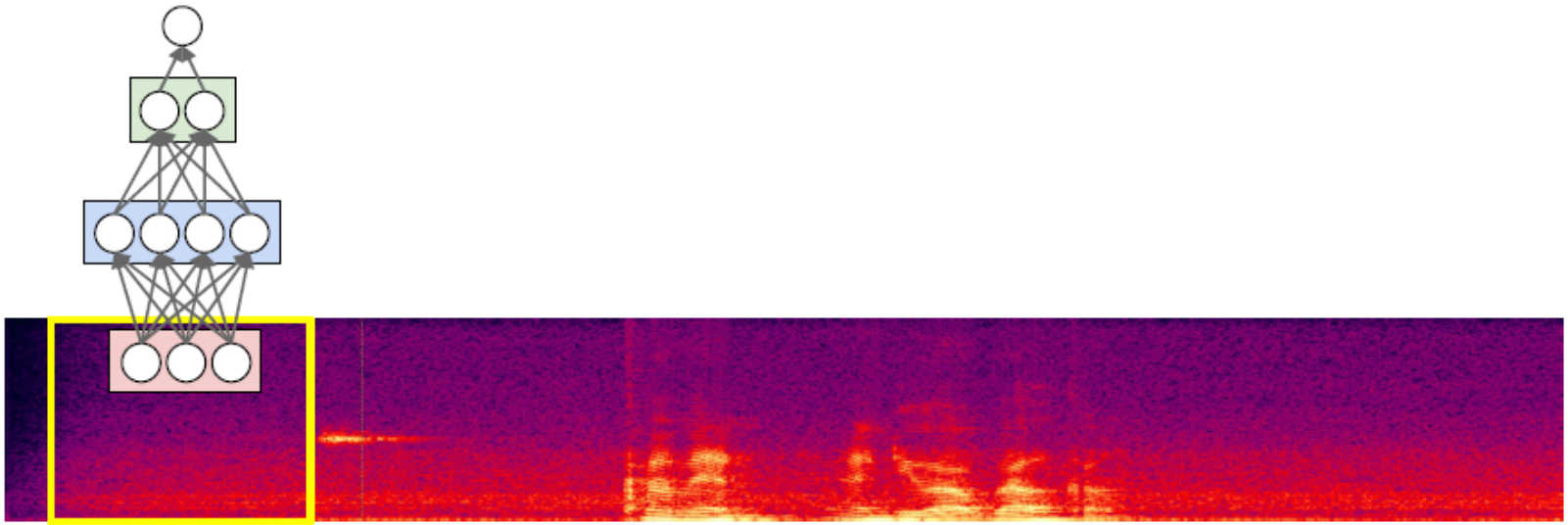
A problem

- In many problems the location of a pattern is not important
  - Only the presence of the pattern
- Conventional MLPs are sensitive to the location of the pattern
  - Moving it by one component results in an entirely different input that the MLP wont recognize
- Requirement: Network must be **shift invariant**



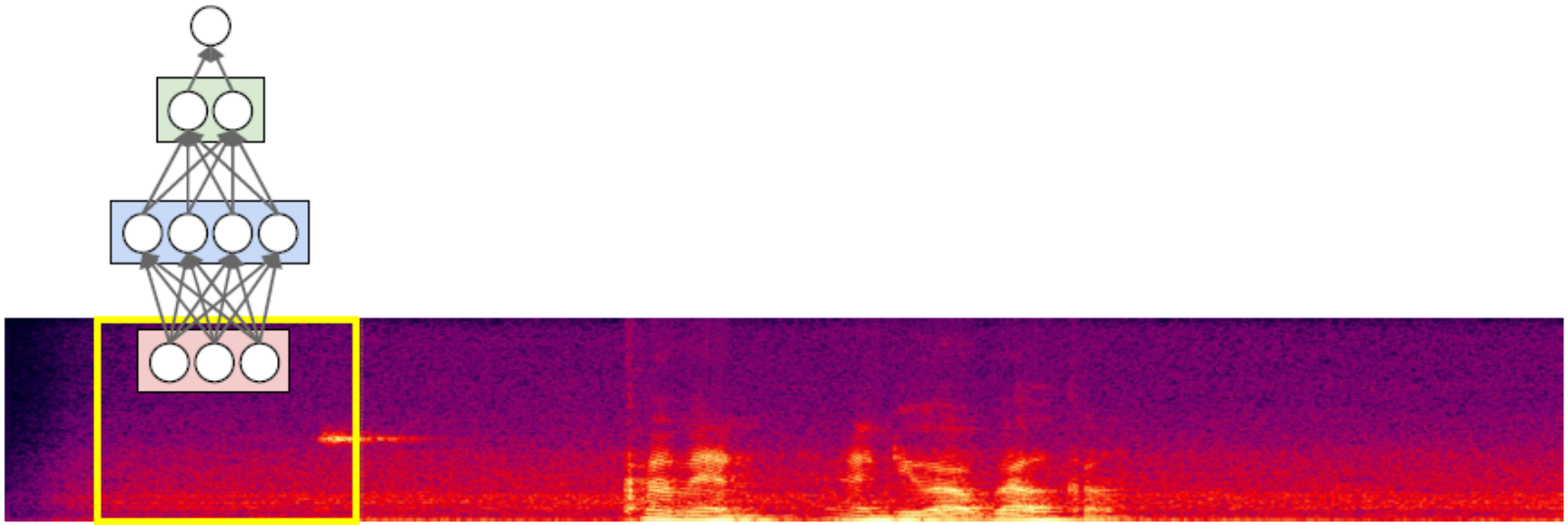
Solution: Scan

- Scan for the target word
  - The spectral time-frequency components in a “window” are input to a “welcome-detector” MLP



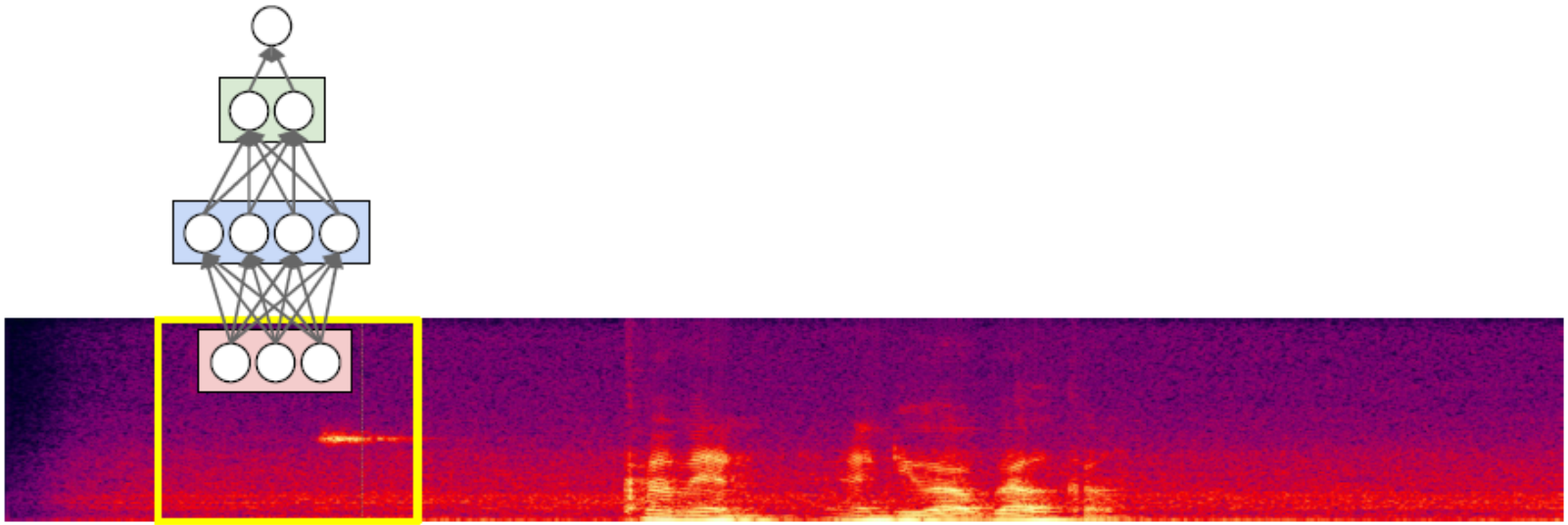
Solution: Scan

- Scan for the target word
  - The spectral time-frequency components in a “window” are input to a “welcome-detector” MLP



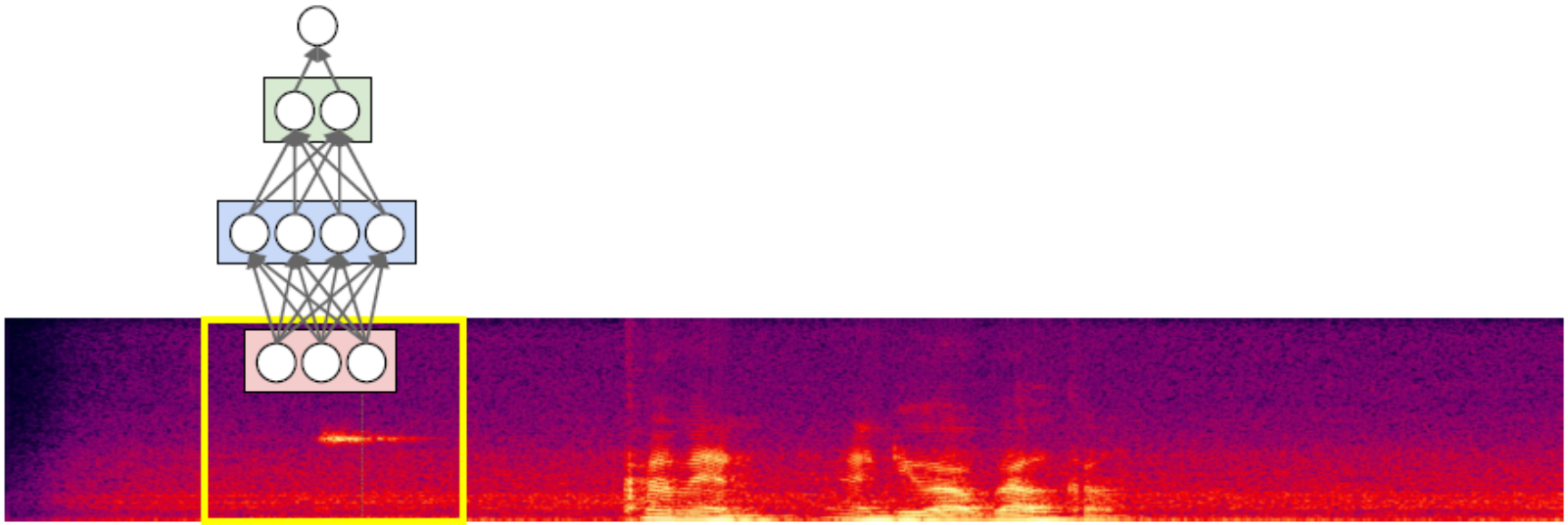
Solution: Scan

- Scan for the target word
  - The spectral time-frequency components in a “window” are input to a “welcome-detector” MLP



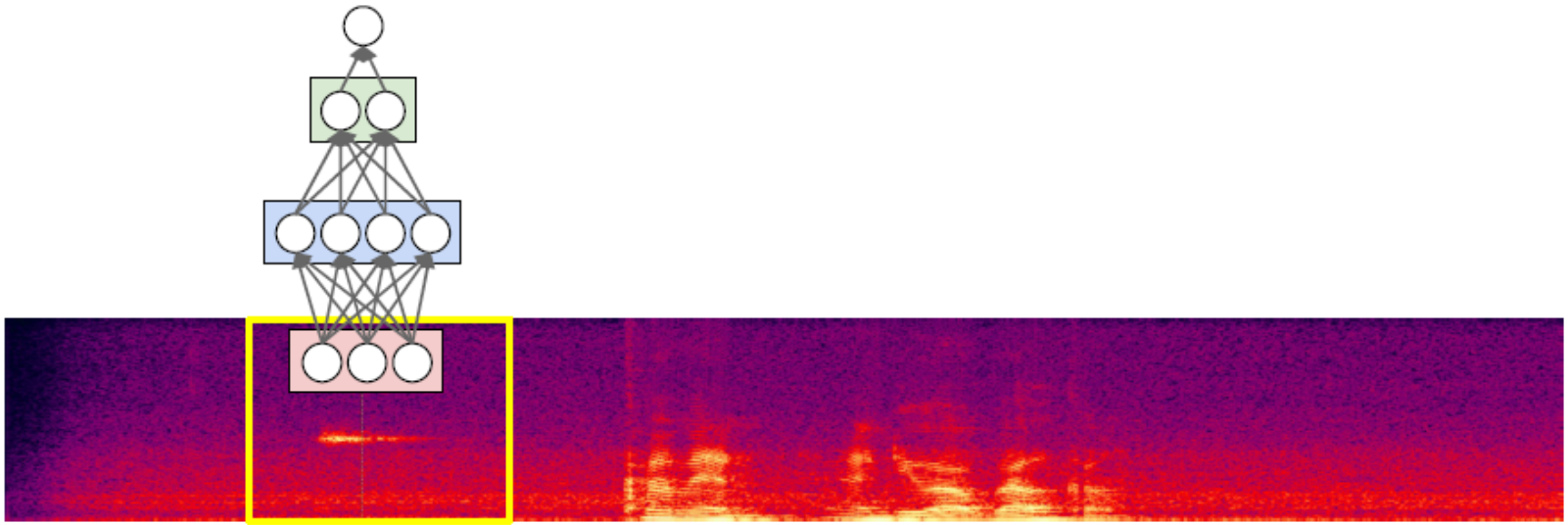
Solution: Scan

- Scan for the target word
  - The spectral time-frequency components in a “window” are input to a “welcome-detector” MLP



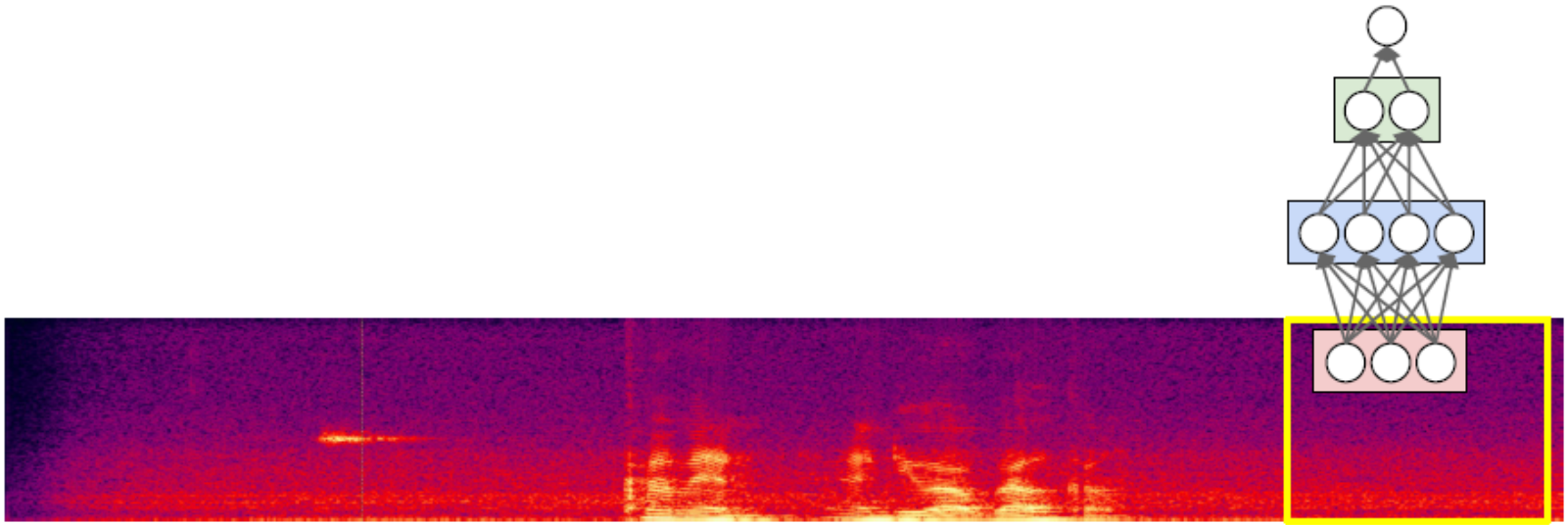
Solution: Scan

- Scan for the target word
  - The spectral time-frequency components in a “window” are input to a “welcome-detector” MLP



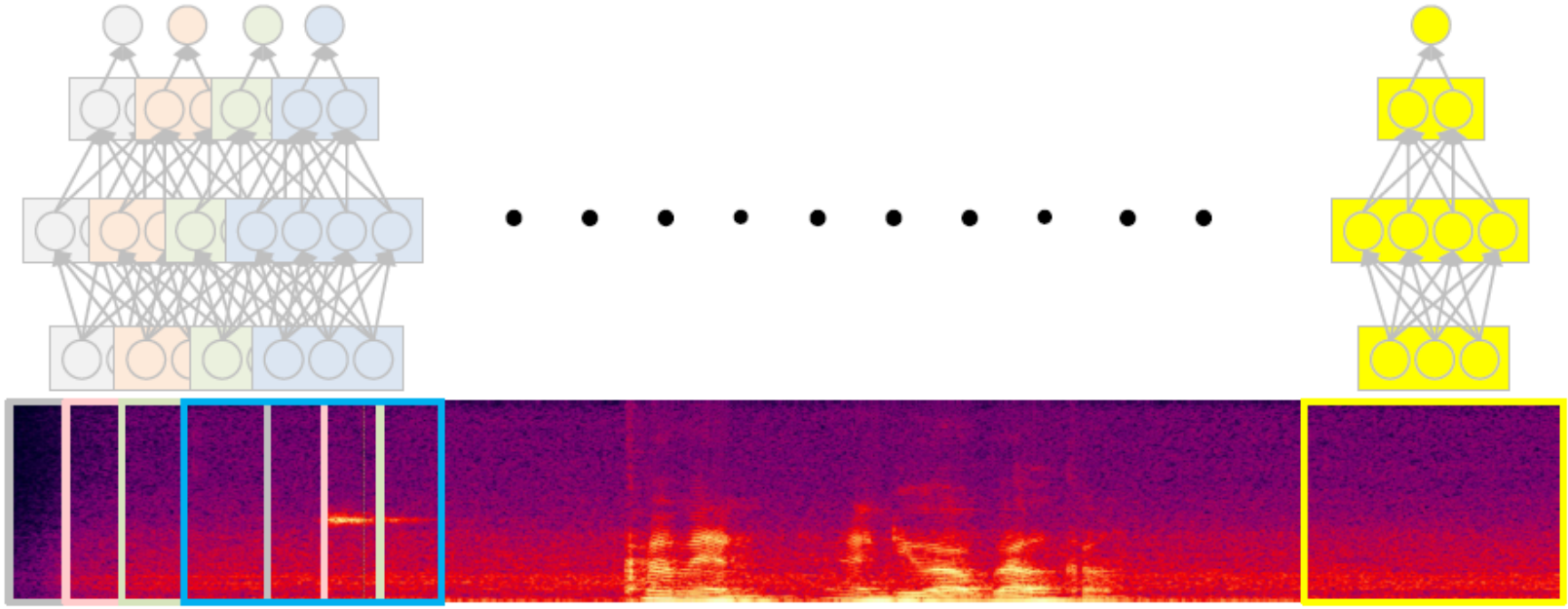
Solution: Scan

- Scan for the target word
  - The spectral time-frequency components in a “window” are input to a “welcome-detector” MLP



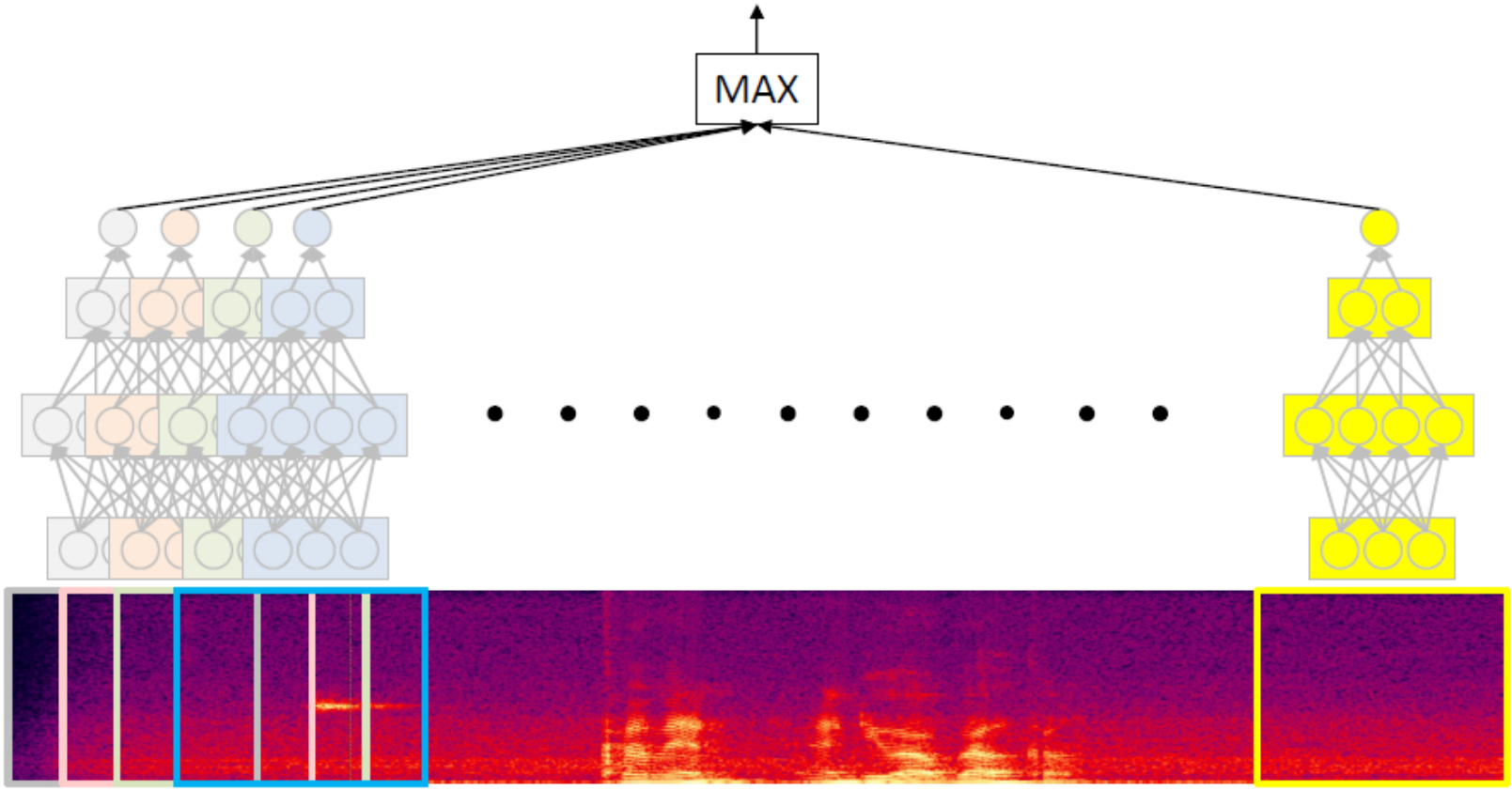
Solution: Scan

- "Does welcome occur in this recording?"
  - We have classified many "windows" individually
  - "Welcome" may have occurred in any of them



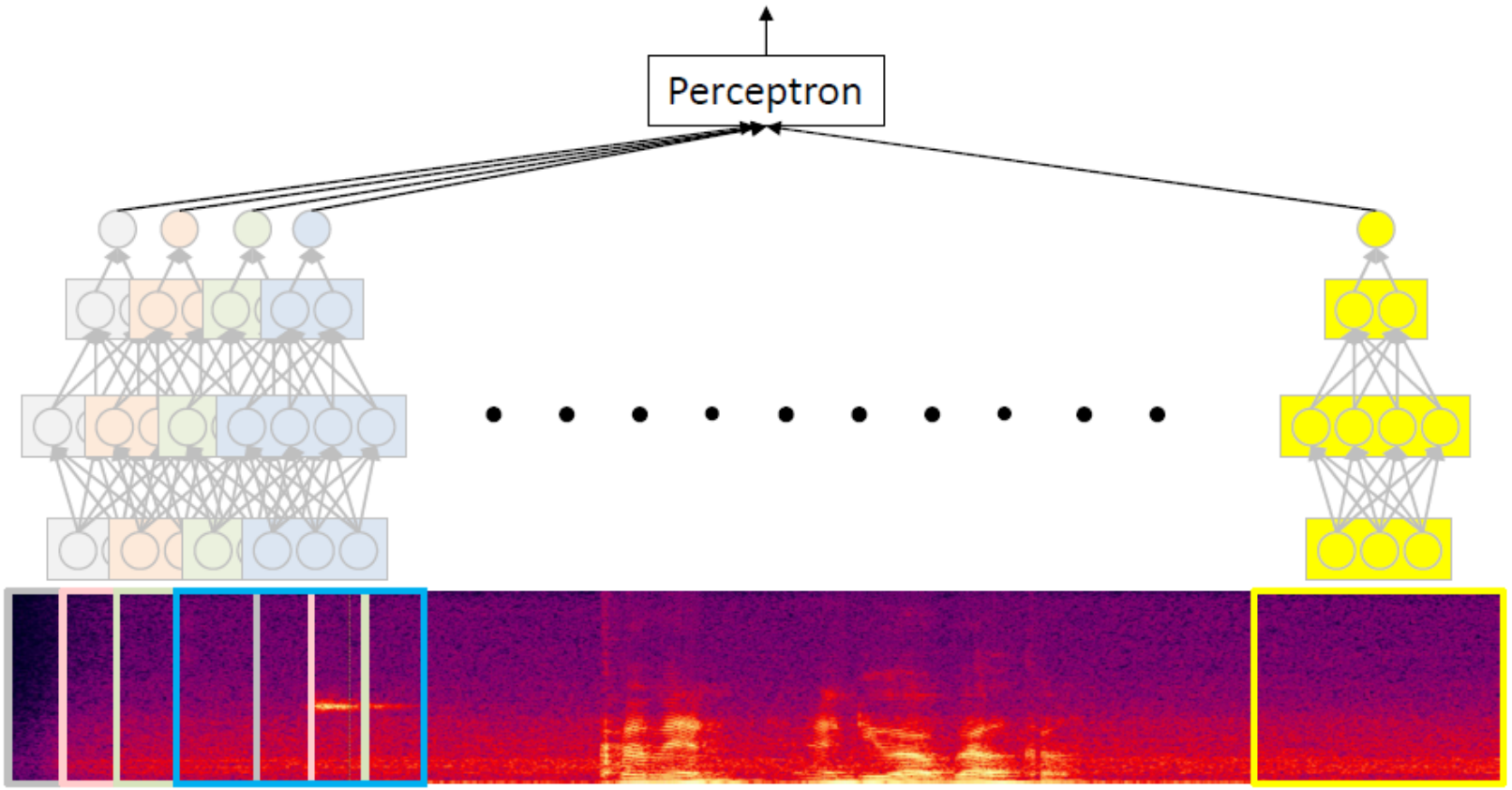
Solution: Scan

- "Does welcome occur in this recording?"
  - We have classified many "windows" individually
  - "Welcome" may have occurred in any of them
  - Maximum of all the outputs (Equivalent of Boolean OR)



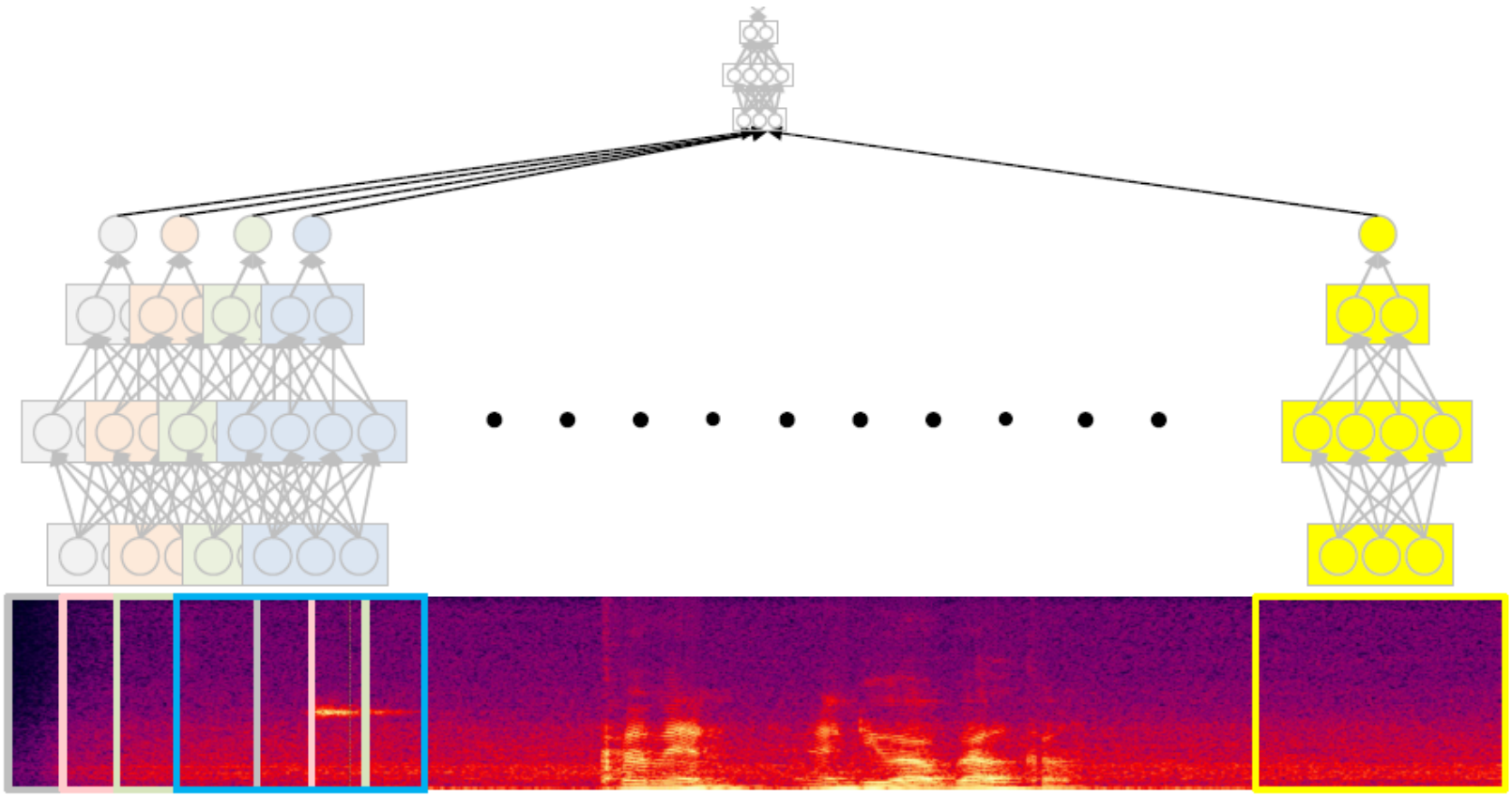
Solution: Scan

- "Does welcome occur in this recording?"
  - Maximum of all the outputs (Equivalent of Boolean OR)
  - Or a proper softmax/logistic
    - Finding a welcome in adjacent windows makes it more likely that we didn't find noise
    - Adjacent windows can combine their evidence



Solution: Scan

- "Does welcome occur in this recording?"
  - Maximum of all the outputs (Equivalent of Boolean OR)
  - Or a proper softmax/logistic
  - Or even an MLP



Scanning with an MLP

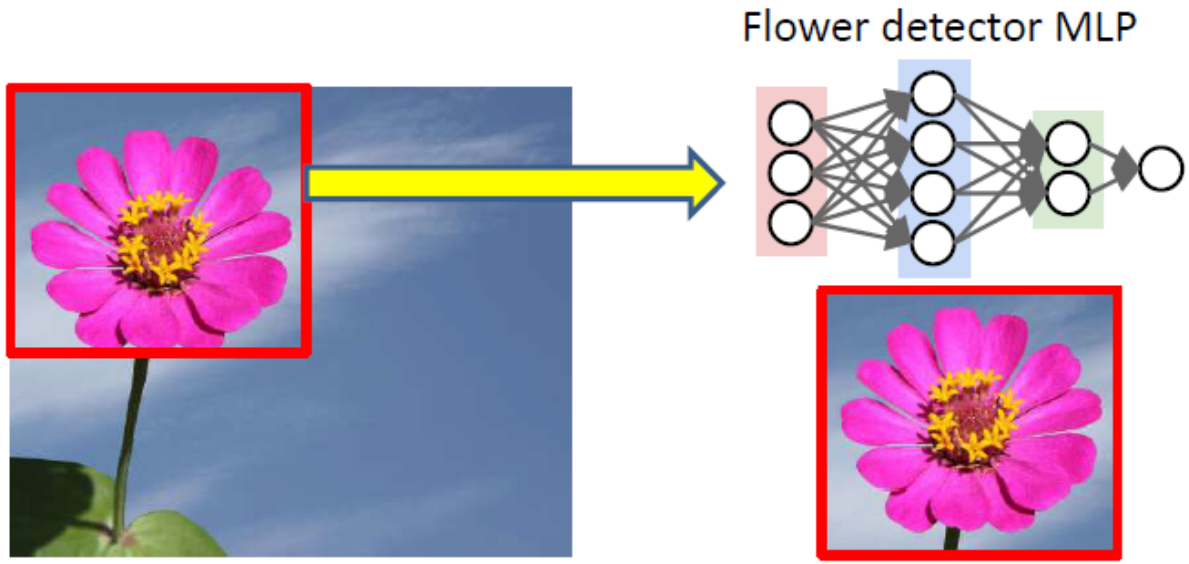
- K = width of "patch" evaluated by MLP

```
For t = 1:T-K+1
  XSegment = x(:, t:t+K-1)
  y(t) = MLP(XSegment)

Y = softmax(y(1)..y(T-K+1))
```

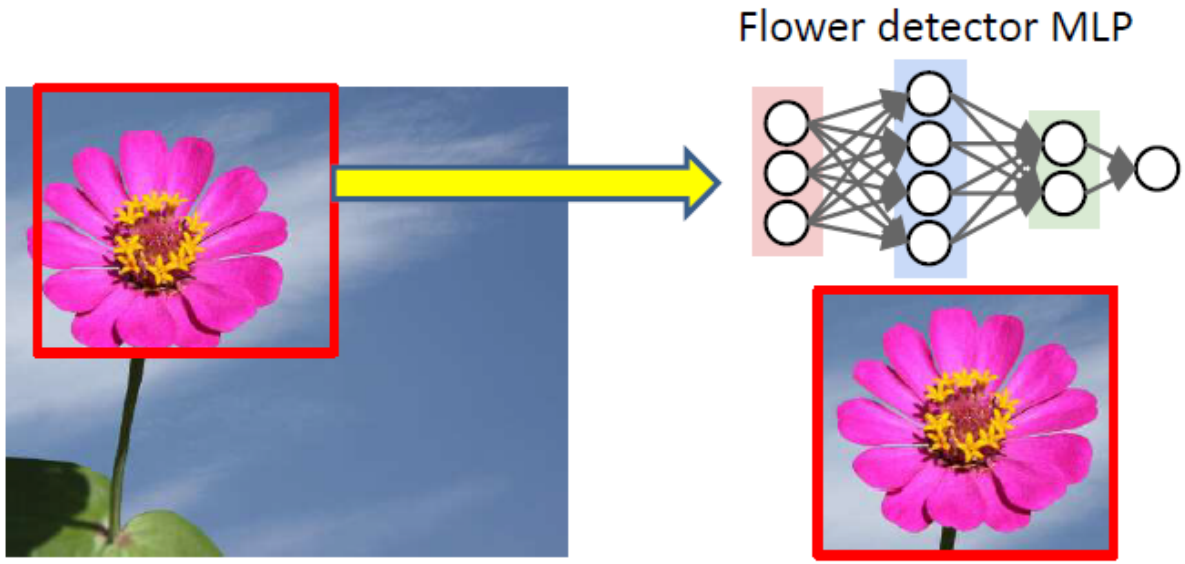
Scanning the picture to find a flower

- Scan for the desired object
- At each location, the entire region is sent through the MLP



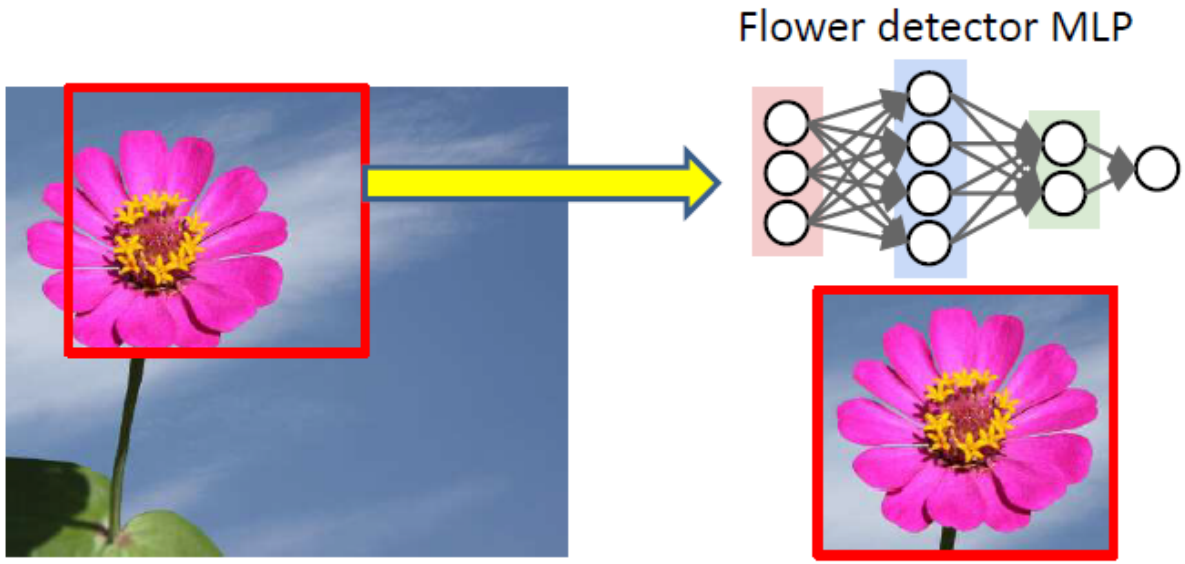
Scanning the picture to find a flower

- Scan for the desired object
- At each location, the entire region is sent through the MLP



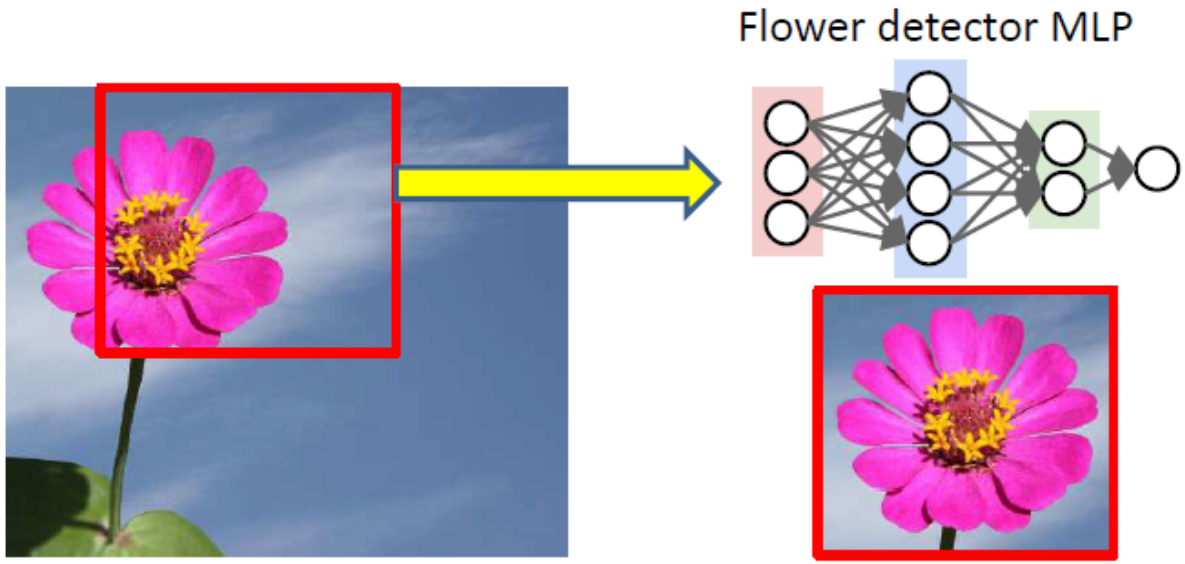
Scanning the picture to find a flower

- Scan for the desired object
- At each location, the entire region is sent through the MLP



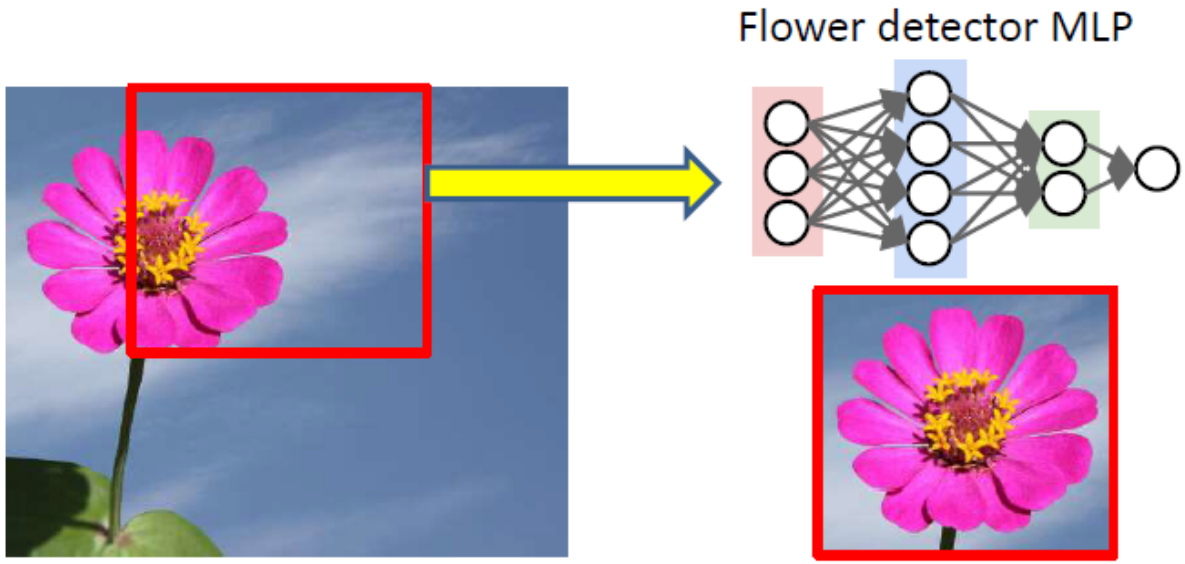
Scanning the picture to find a flower

- Scan for the desired object
- At each location, the entire region is sent through the MLP



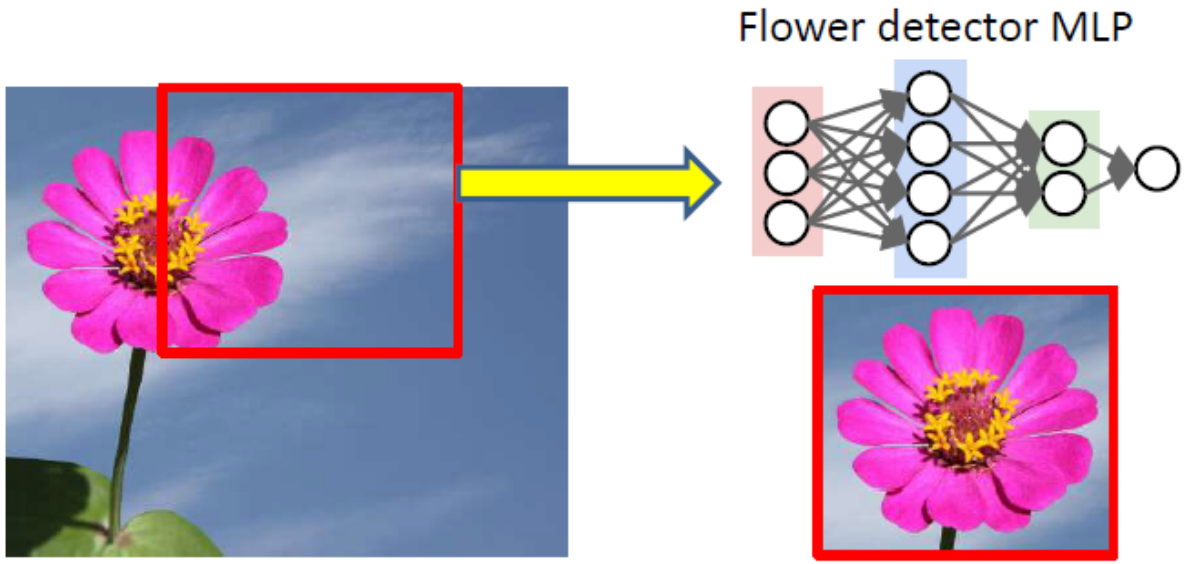
Scanning the picture to find a flower

- Scan for the desired object
- At each location, the entire region is sent through the MLP



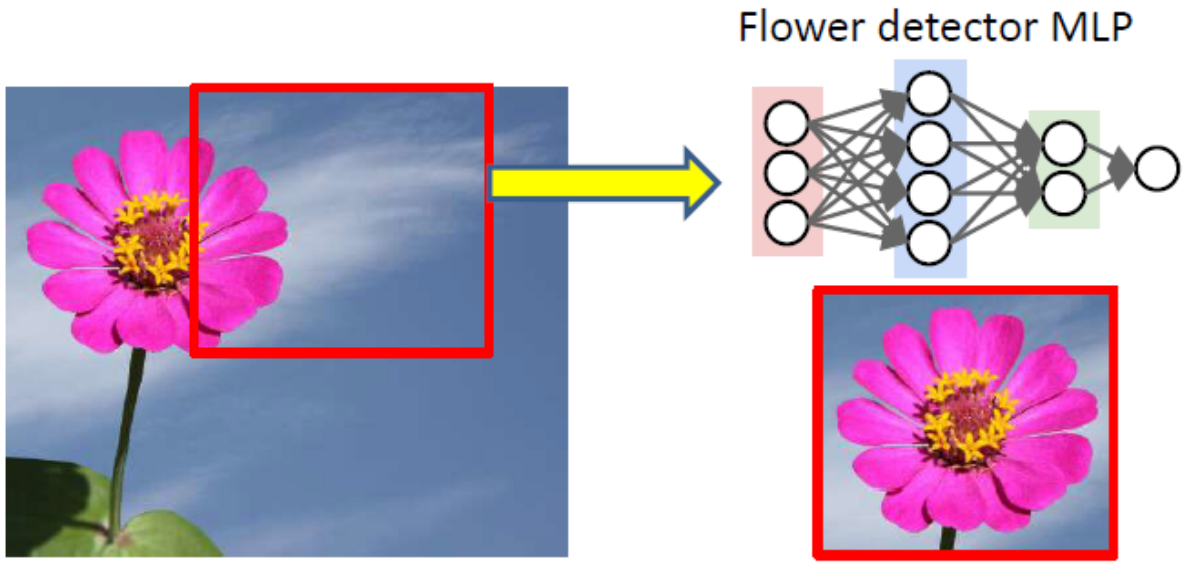
Scanning the picture to find a flower

- Scan for the desired object
- At each location, the entire region is sent through the MLP



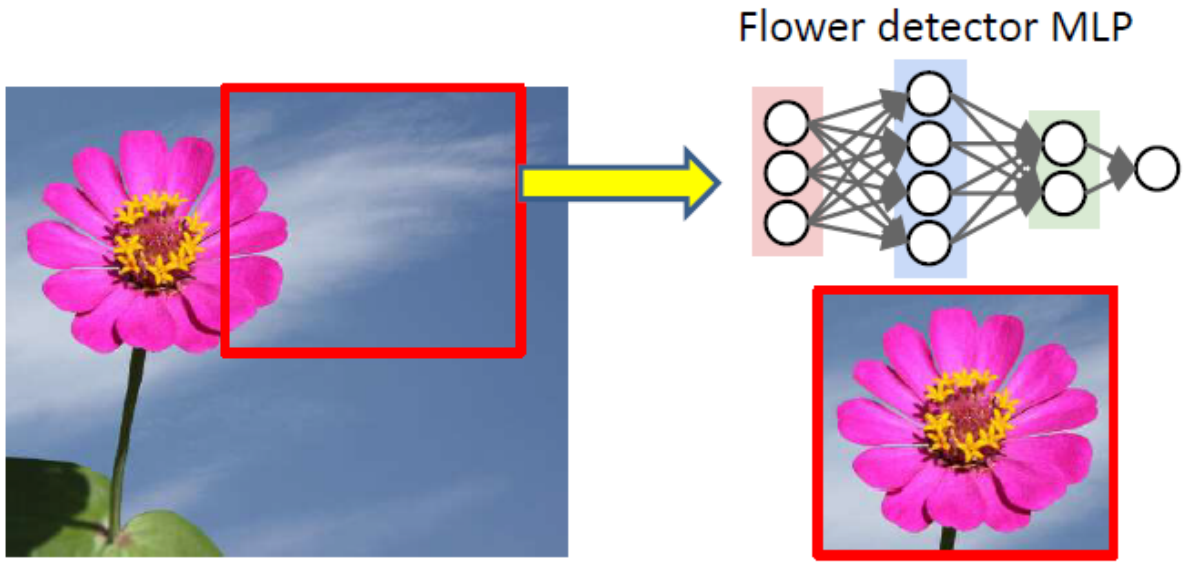
Scanning the picture to find a flower

- Scan for the desired object
- At each location, the entire region is sent through the MLP



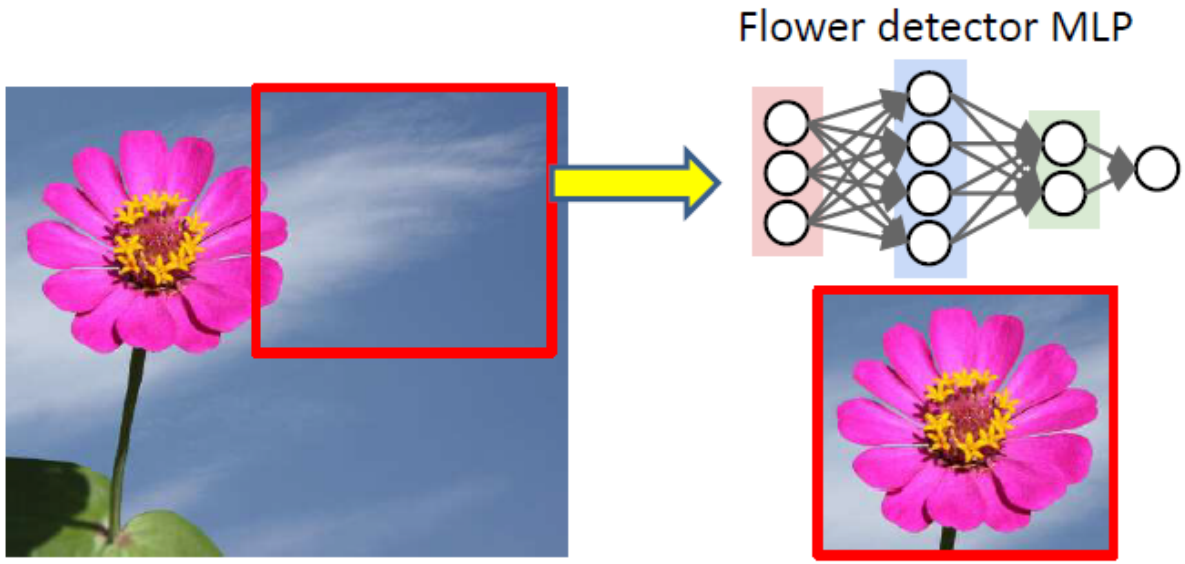
Scanning the picture to find a flower

- Scan for the desired object
- At each location, the entire region is sent through the MLP



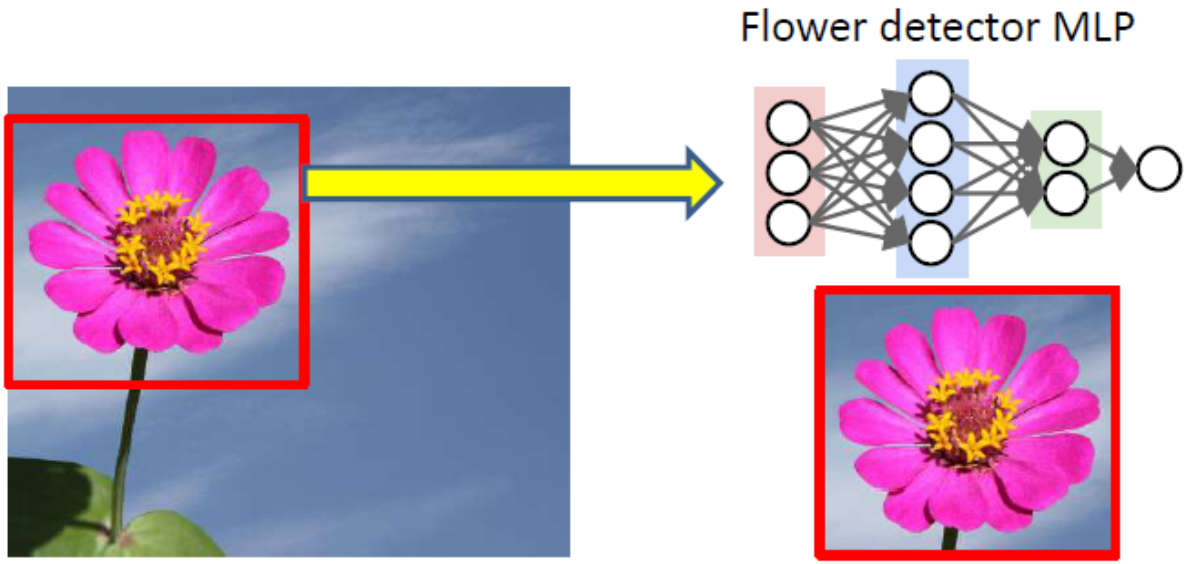
Scanning the picture to find a flower

- Scan for the desired object
- At each location, the entire region is sent through the MLP



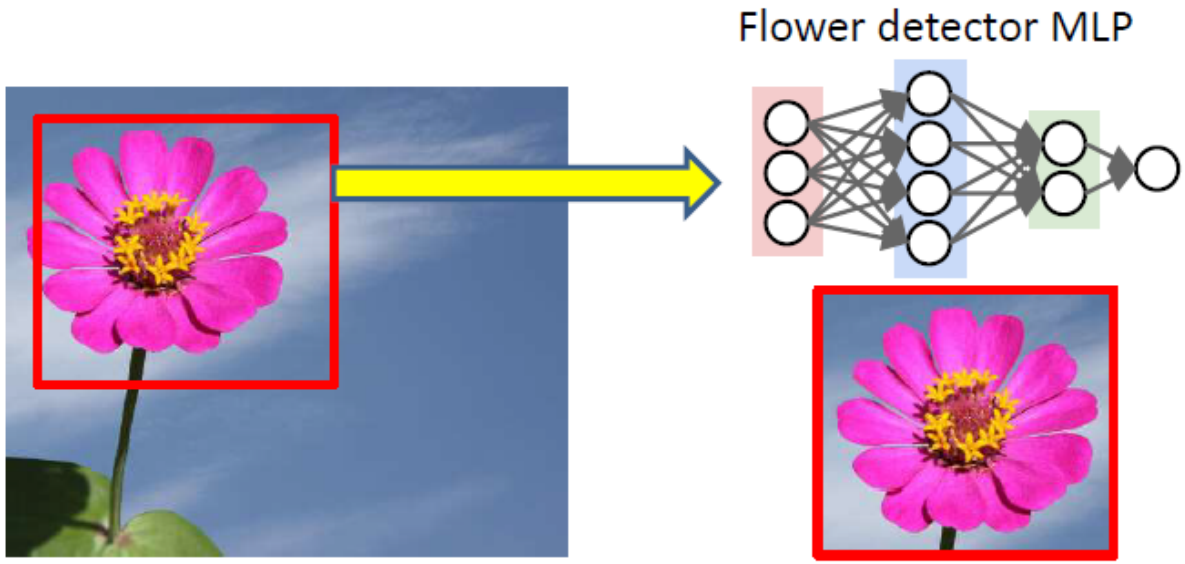
Scanning the picture to find a flower

- Scan for the desired object
- At each location, the entire region is sent through the MLP



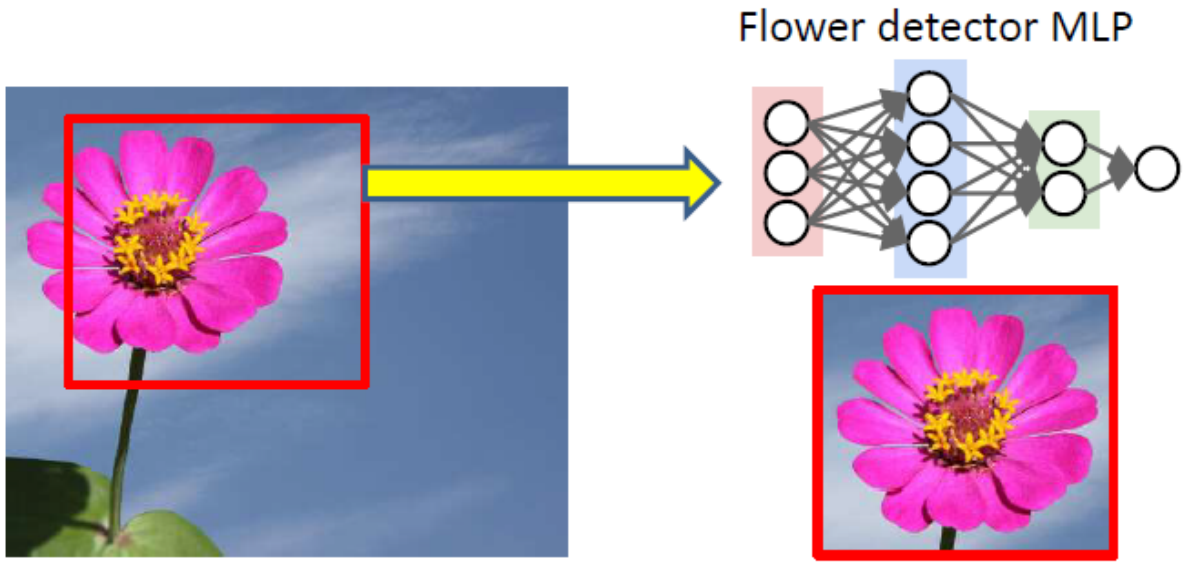
Scanning the picture to find a flower

- Scan for the desired object
- At each location, the entire region is sent through the MLP



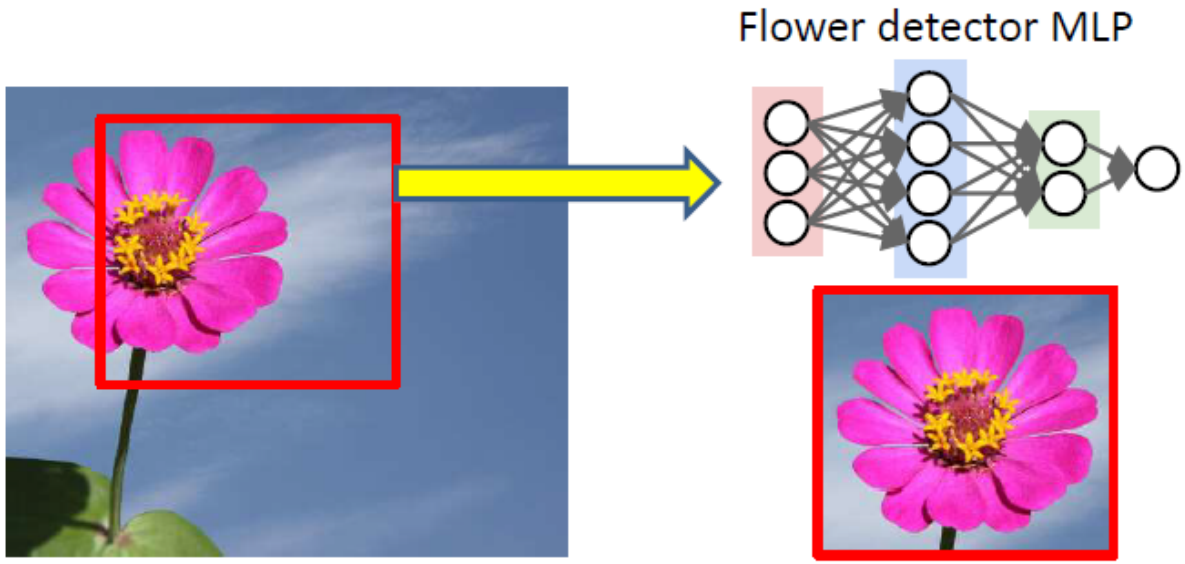
Scanning the picture to find a flower

- Scan for the desired object
- At each location, the entire region is sent through the MLP



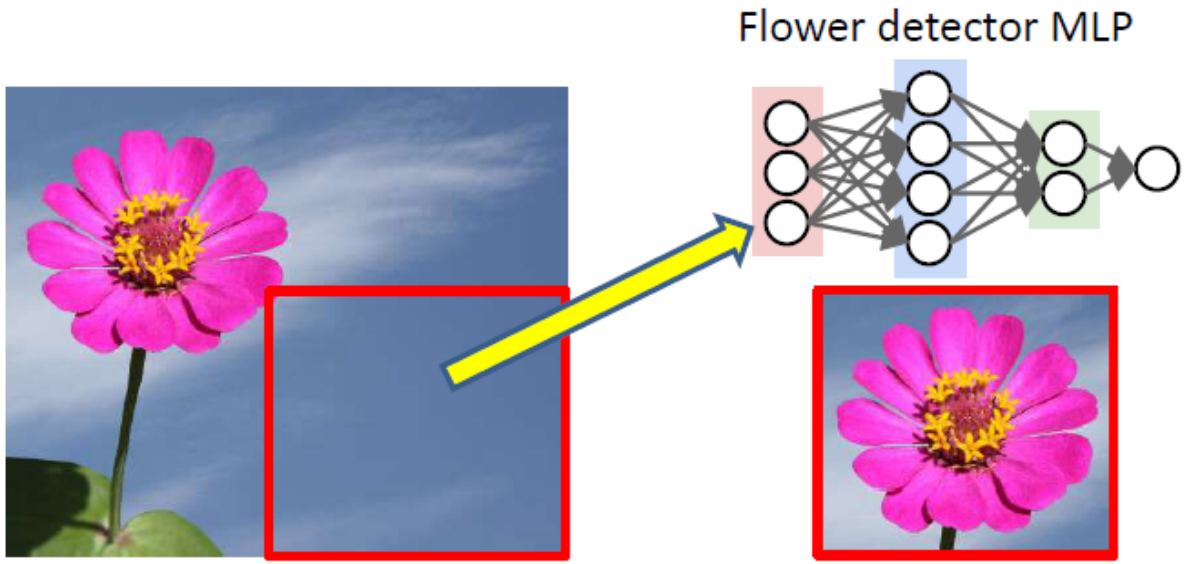
Scanning the picture to find a flower

- Scan for the desired object
- At each location, the entire region is sent through the MLP



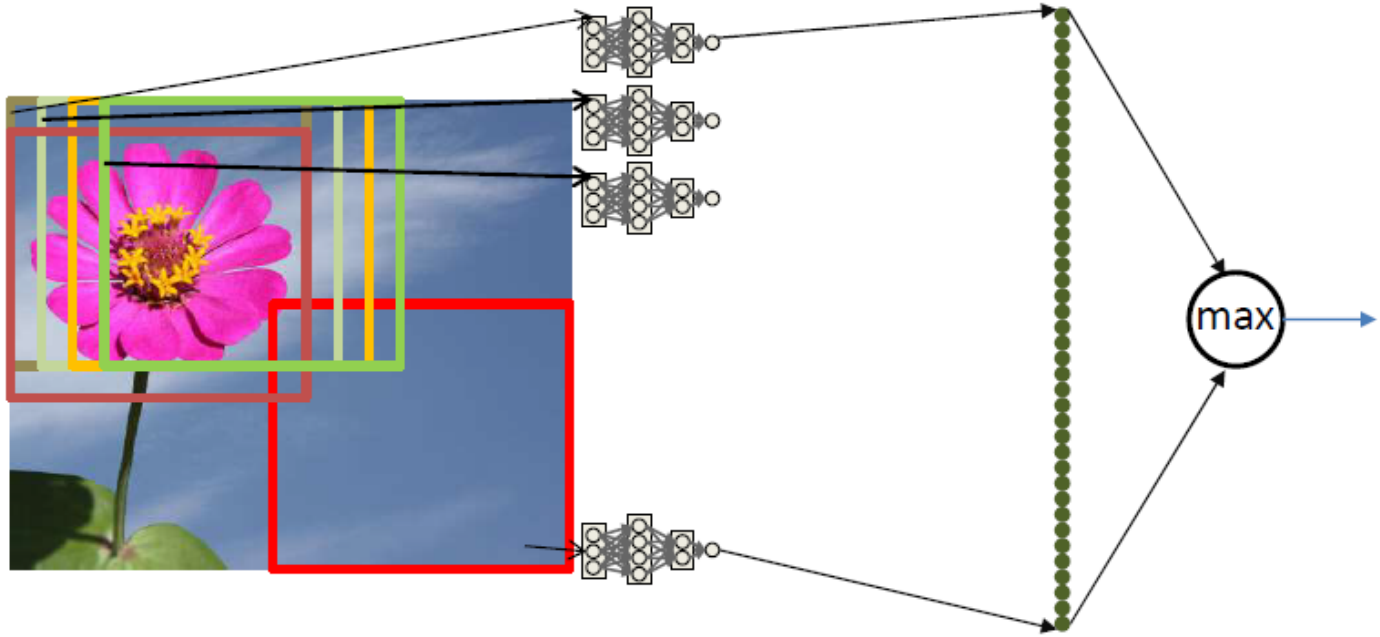
Scanning the picture to find a flower

- Scan for the desired object
- At each location, the entire region is sent through the MLP



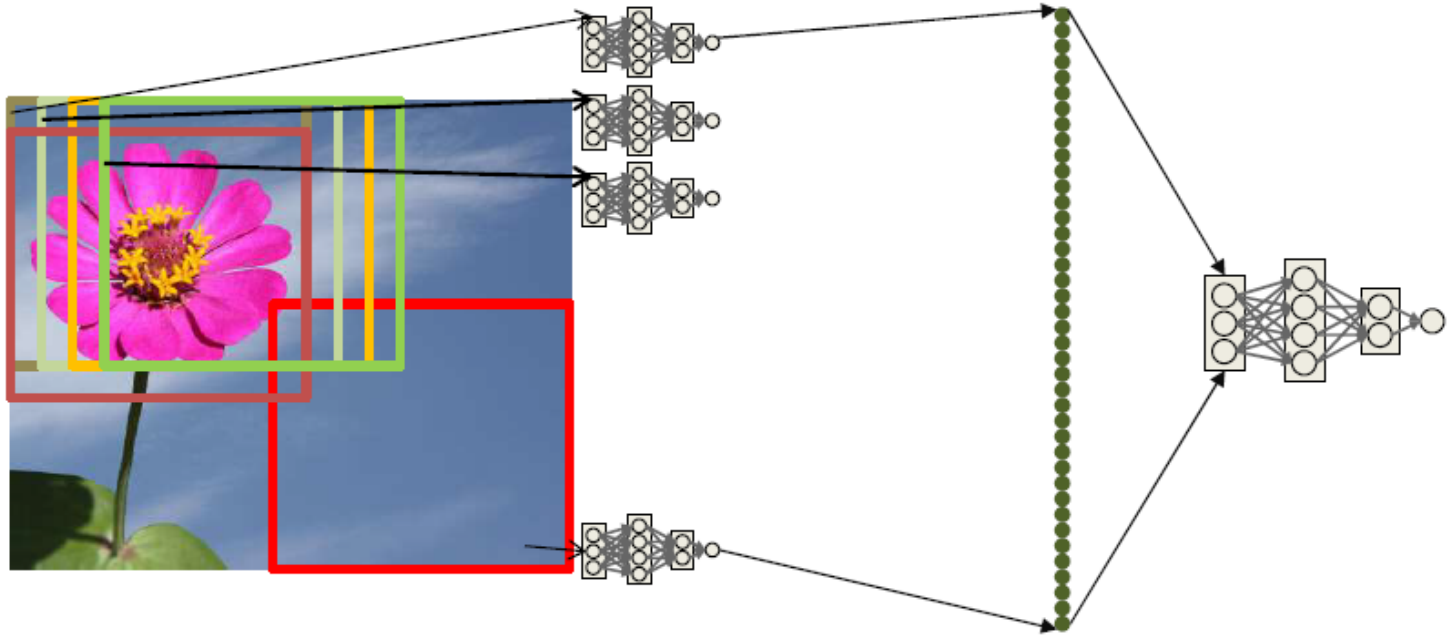
Scanning the picture to find a flower

- Determine if any of the locations had a flower
  - We get one classification output per scanned location
    - Each dot in the right represents the output of the MLP when it classifies one location in the input figure
    - The score output by the MLP
  - Look at the maximum value
    - If the picture has a flower, the location with the flower will result in high output value



Scanning the picture to find a flower

- Determine if any of the locations had a flower
  - We get one classification output per scanned location
    - Each dot in the right represents the output of the MLP when it classifies one location in the input figure
    - The score output by the MLP
  - Look at the maximum value
    - If the picture has a flower, the location with the flower will result in high output value
    - Or pass it through a softmax or even an MLP



Scanning with an MLP

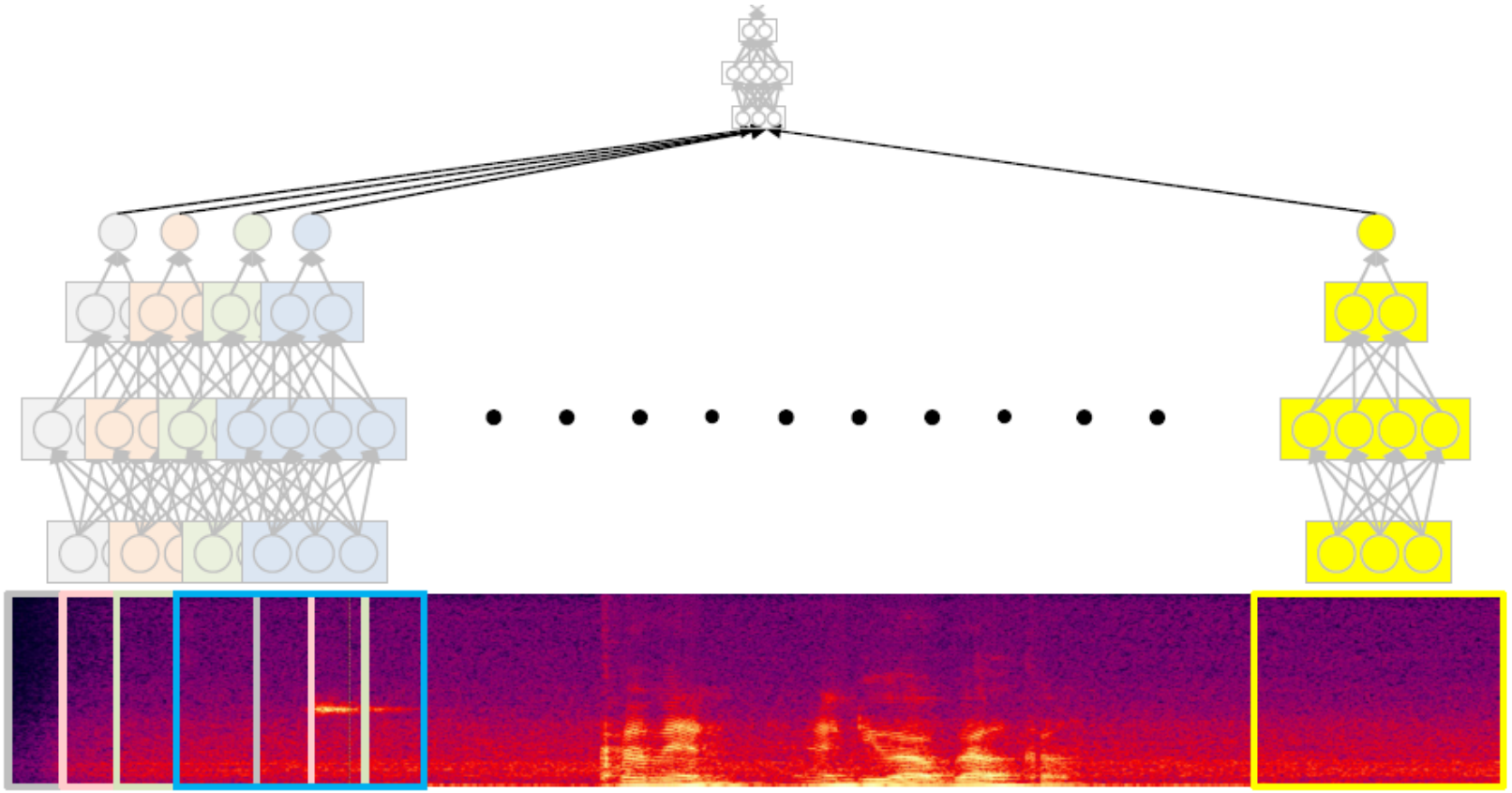
- $K \times K$ = size of "patch" evaluated by MLP
- W is width of image
- H is height of image

```
For i = 1:W-K+1
    For j = 1:H-K+1
        ImgSegment = Img(i:i+K-1, j:j+K-1)
        y(i,j) = MLP(ImgSegment)

Y = softmax( y(1,1)..y(W-K+1,H-K+1) )
```

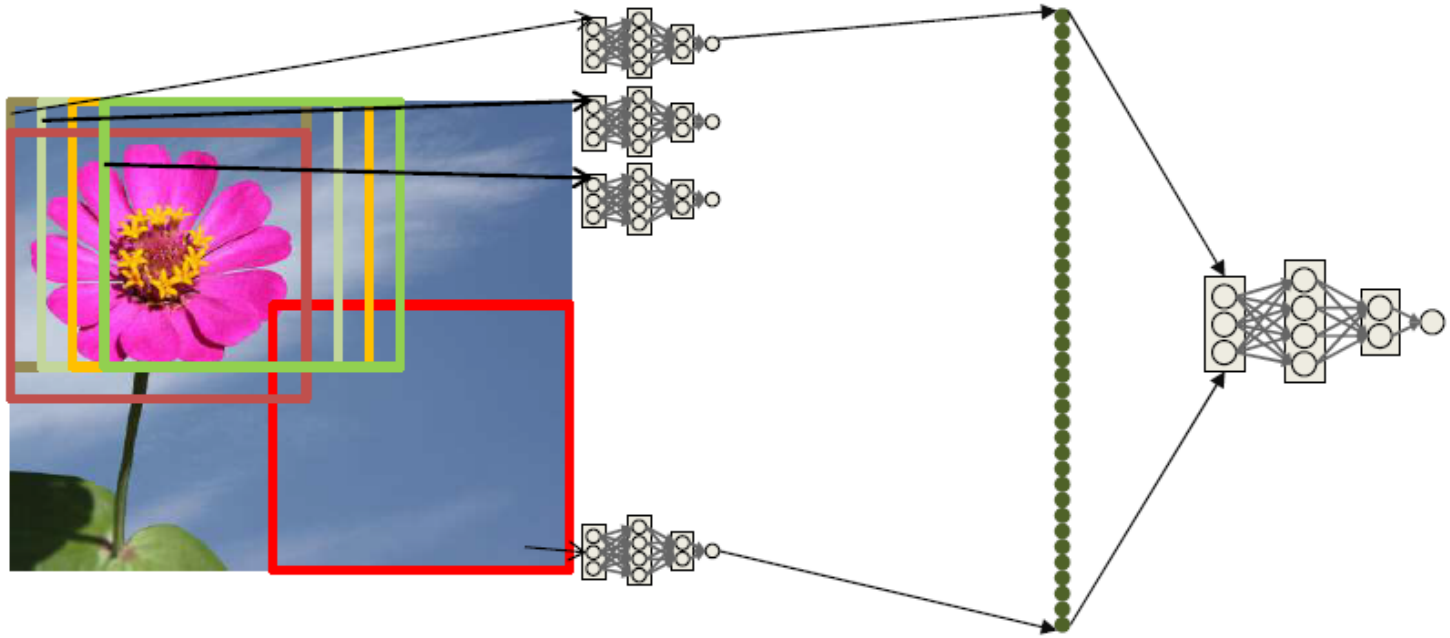
Its just a giant network with common subnets

- The entire operation can be viewed as one **giant network**
  - With many subnetworks, one per window
  - Restriction: **All subnets are identical and have an identical set of parameters**
    - This is what we call a shared parameter network
- The network is shift-invariant!



Its just a giant network with common subnets

- The entire operation can be viewed as one giant network
  - With many subnetworks, one per window
  - With one key feature: **all subnets are identical**
- The network is shift-invariant!



References

- Deep Learning, Ian Goodfellow, MIT Press, Ch. 6 and ch. 9