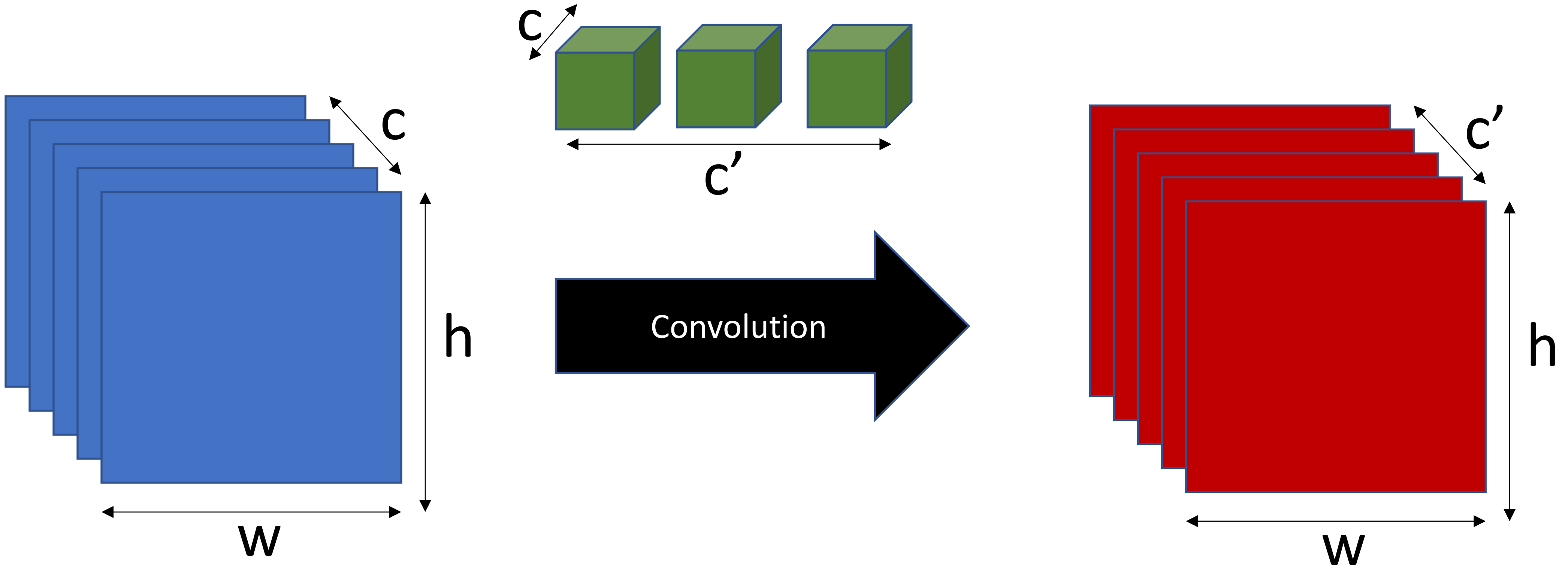
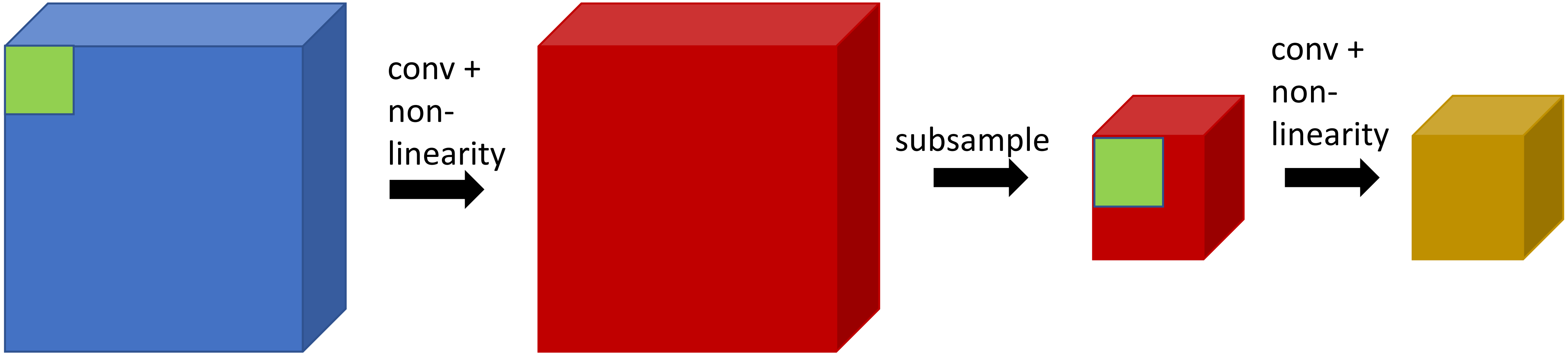
Convolutional networks

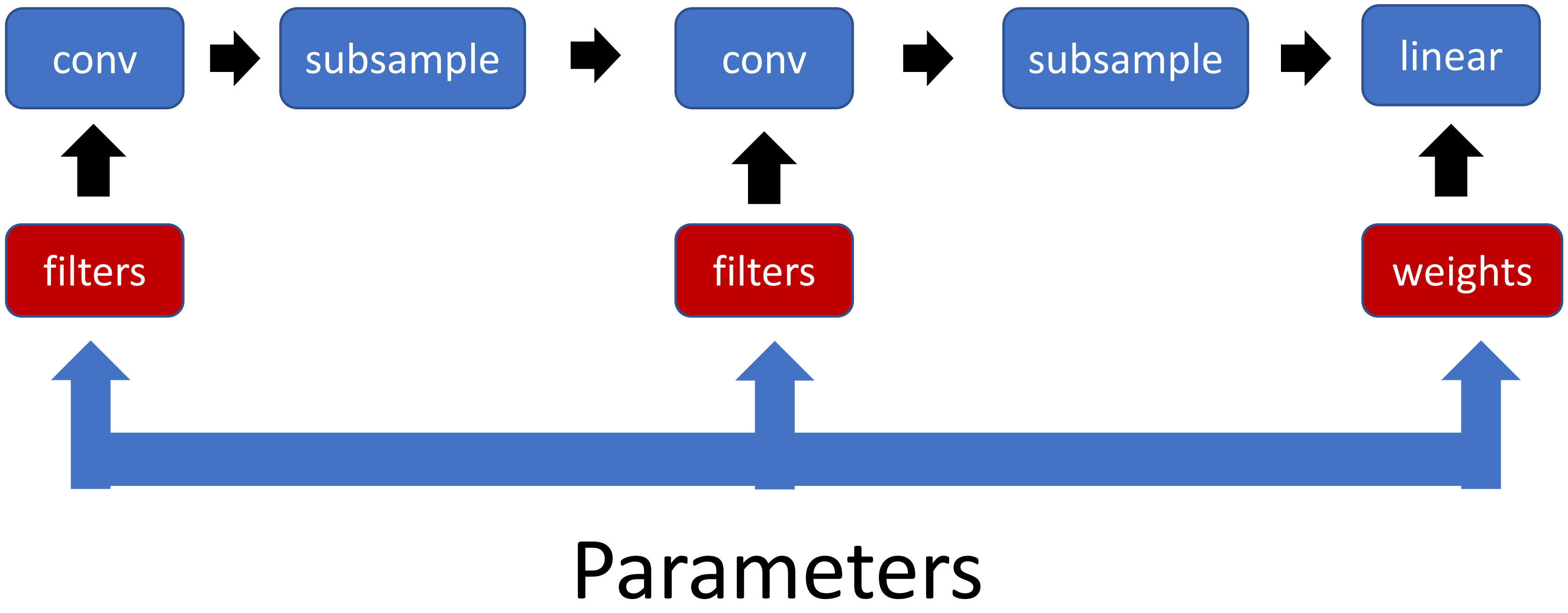
Convolution as a primitive



Convolution subsampling convolution



Convolutional networks



Empirical Risk Minimization

$$\min_{\boldsymbol{\theta}} \frac{1}{N} \sum_{i=1}^{N} L(h(x_i; \boldsymbol{\theta}), y_i)$$

Convolutional network

$$\underbrace{\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \lambda \frac{1}{N} \sum_{i=1}^{N} \nabla L(h(x_i; \boldsymbol{\theta}), y_i)}_{\text{Gradient descent update}}$$

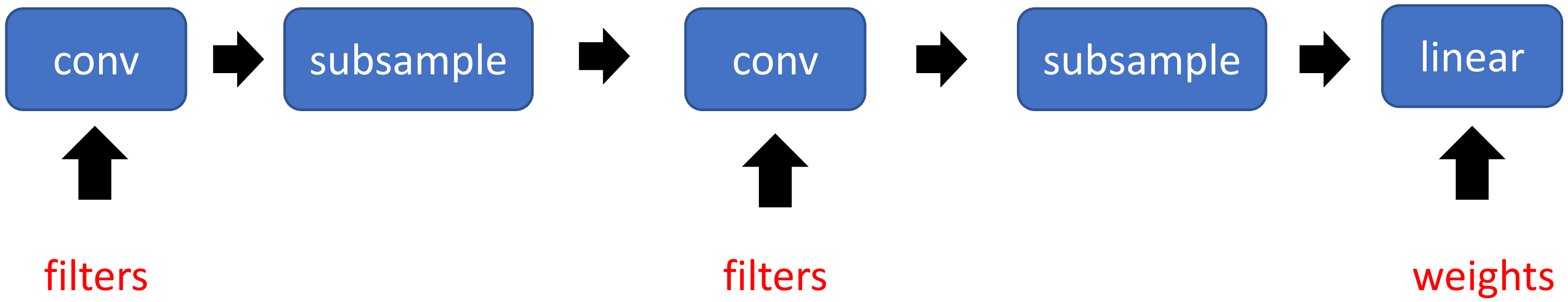
Computing the gradient of the loss

$$\nabla L(h(x;\boldsymbol{\theta}), y)$$

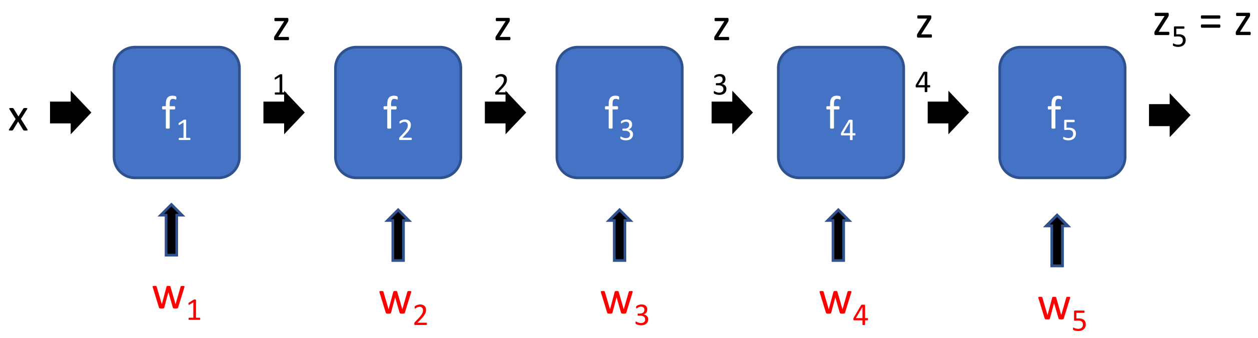
$$z = h(x;\boldsymbol{\theta})$$

$$\nabla_{\boldsymbol{\theta}} L(z, y) = \frac{\partial L(z, y)}{\partial z} \frac{\partial z}{\partial \boldsymbol{\theta}}$$

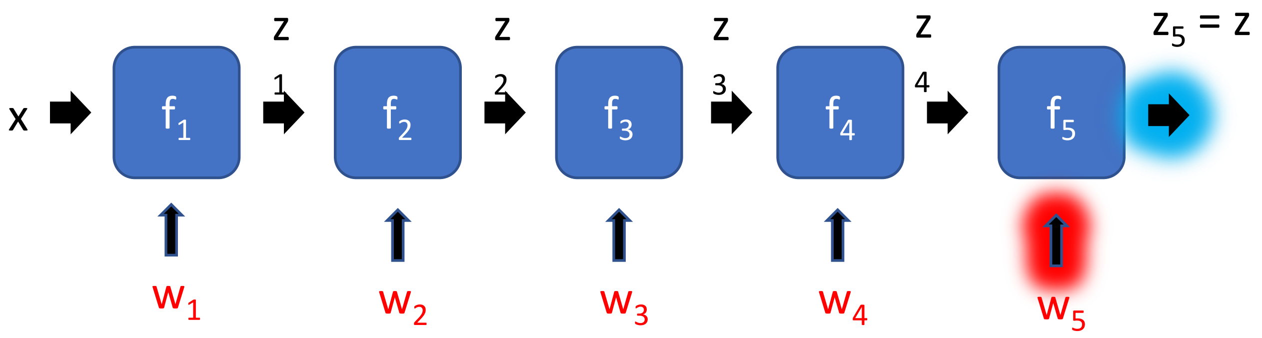
Convolutional networks



The gradient of convnets

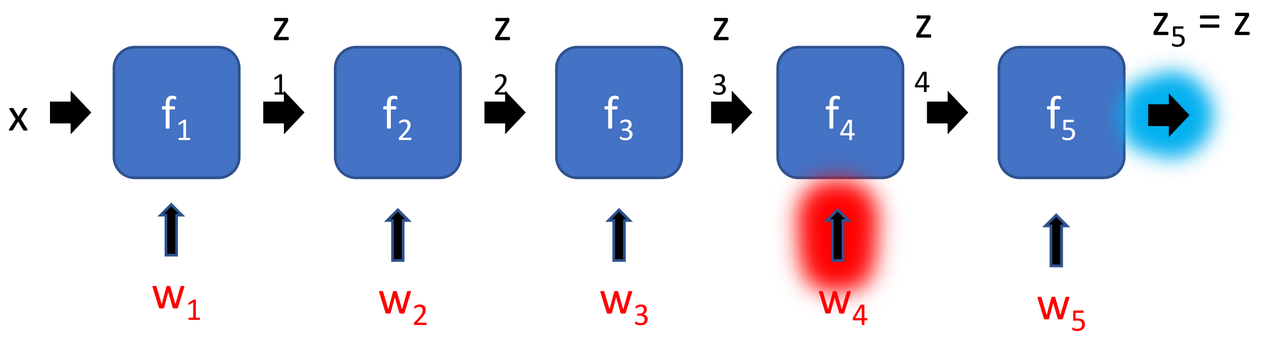


The gradient of convnets



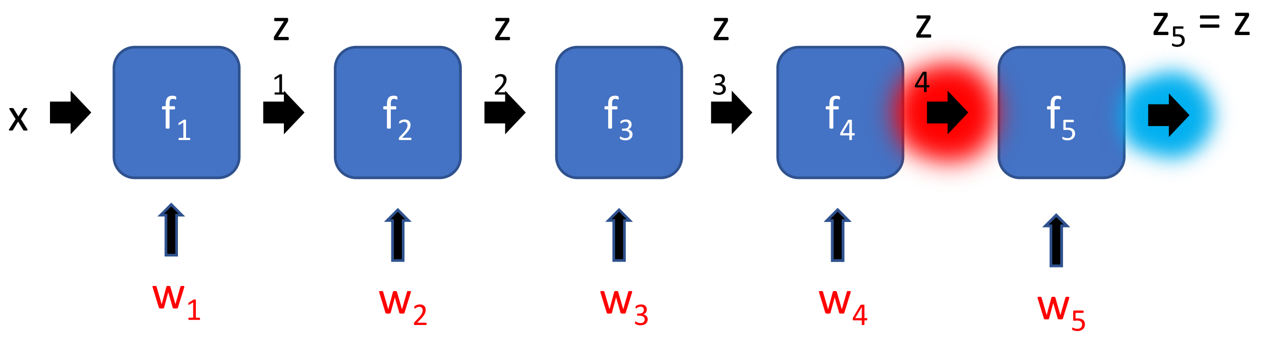
$$\frac{\partial z}{\partial w_5}$$

The gradient of convnets



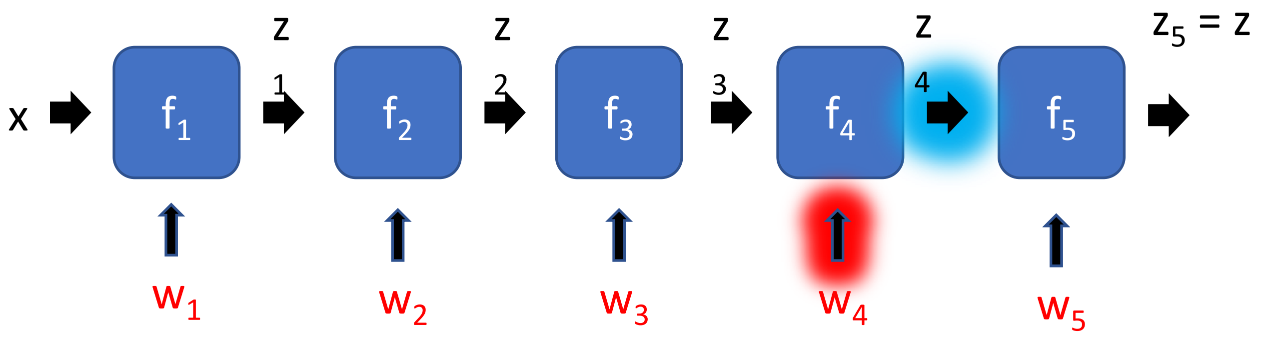
$$\frac{\partial z}{\partial w_4}$$

The gradient of convnets



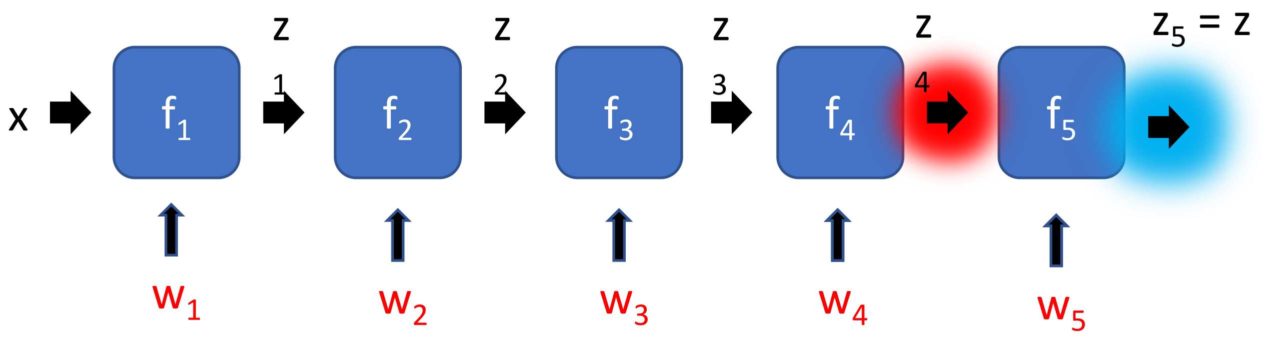
$$\frac{\partial z}{\partial w_4} = \frac{\partial z}{\partial z_4} \frac{\partial z_4}{\partial w_4}$$

The gradient of convnets



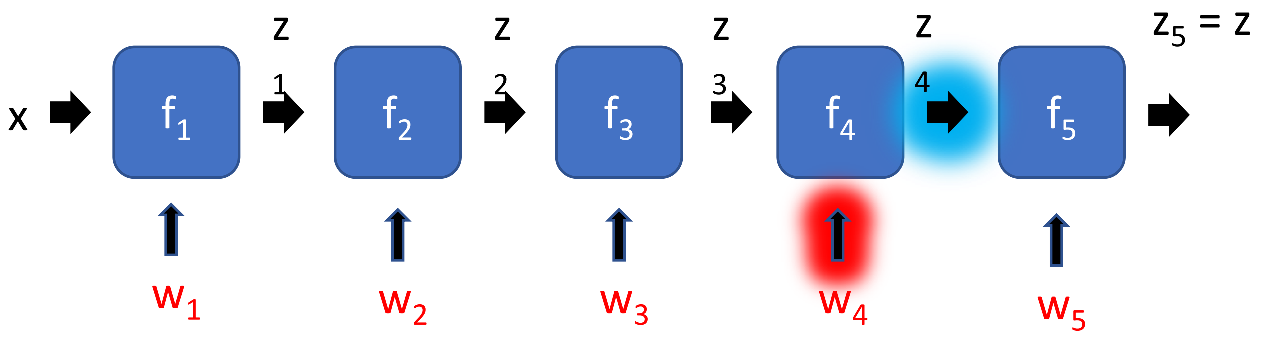
$$\frac{\partial z}{\partial w_4} = \frac{\partial z}{\partial z_4}\frac{\partial z_4}{\partial w_4}$$

The gradient of convnets



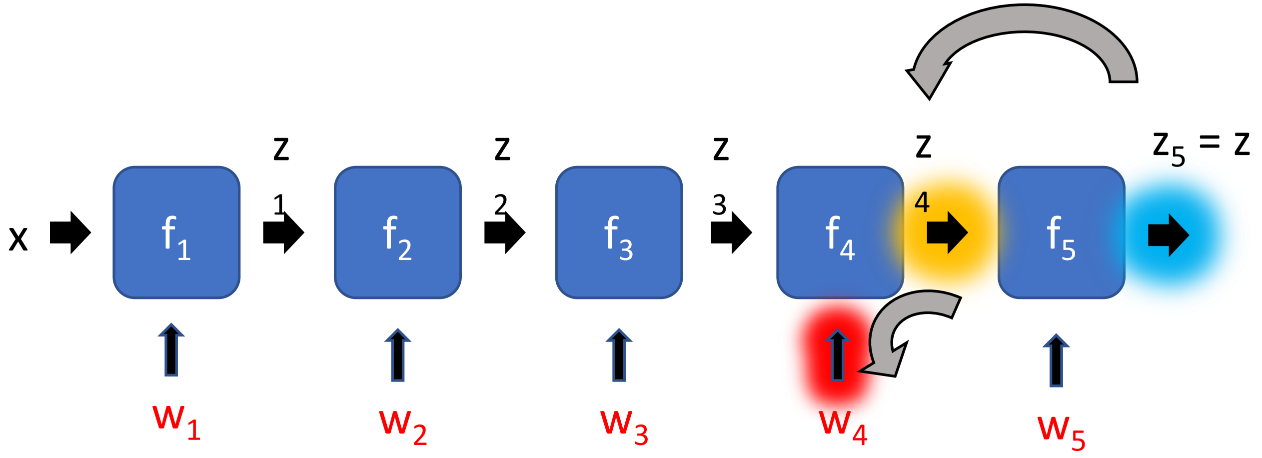
$$\frac{\partial z}{\partial w_4} = \frac{\partial z}{\partial z_4}\frac{\partial z_4}{\partial w_4} = \frac{\partial f_5(z_4, w_5)}{\partial z_4}\frac{\partial f_4(z_3, w_4)}{\partial w_4}$$

The gradient of convnets



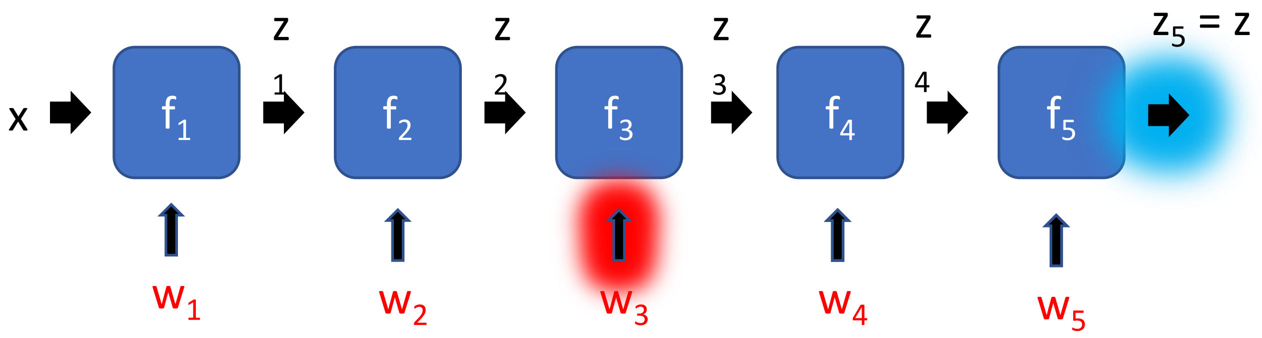
$$\frac{\partial z}{\partial w_4} = \frac{\partial z}{\partial z_4}\frac{\partial z_4}{\partial w_4} = \frac{\partial f_5(z_4, w_5)}{\partial z_4}\frac{\partial f_4(z_3, w_4)}{\partial w_4}$$

The gradient of convnets



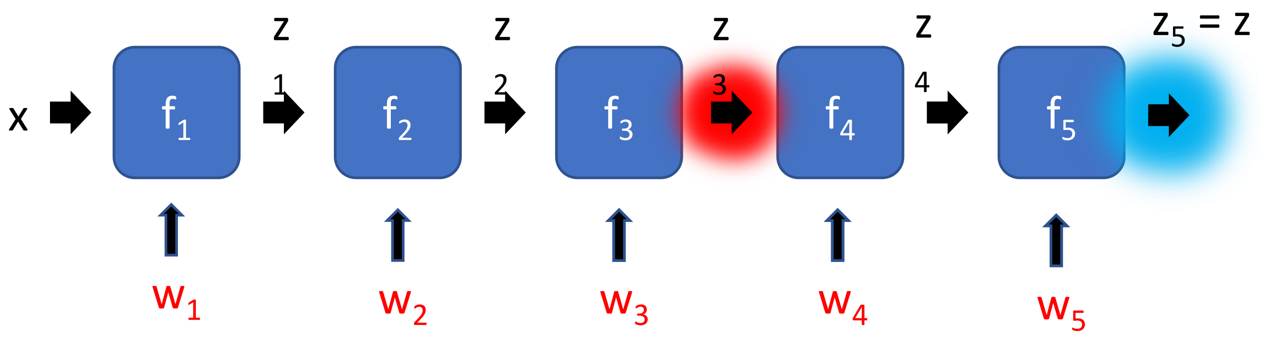
$$\frac{\partial z}{\partial w_4} = \frac{\partial z}{\partial z_4}\frac{\partial z_4}{\partial w_4} = \frac{\partial f_5(z_4, w_5)}{\partial z_4}\frac{\partial f_4(z_3, w_4)}{\partial w_4}$$

The gradient of convnets



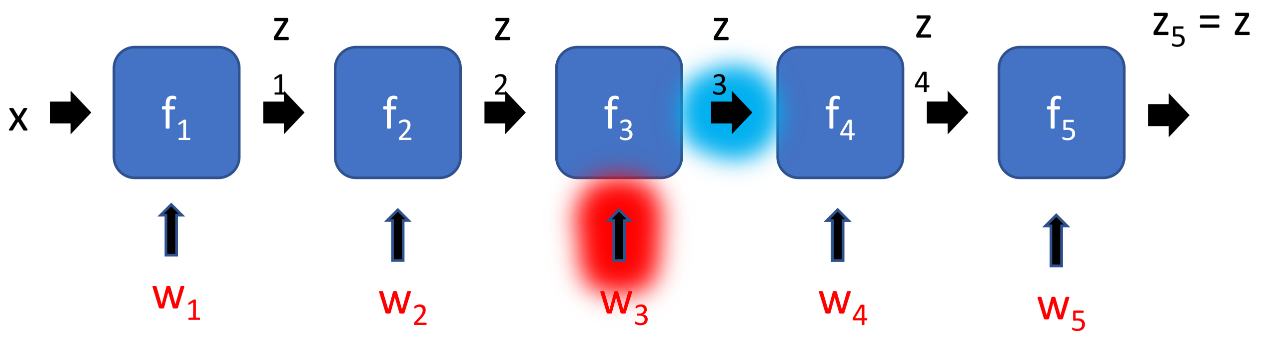
$$\frac{\partial z}{\partial w_3}$$

The gradient of convnets



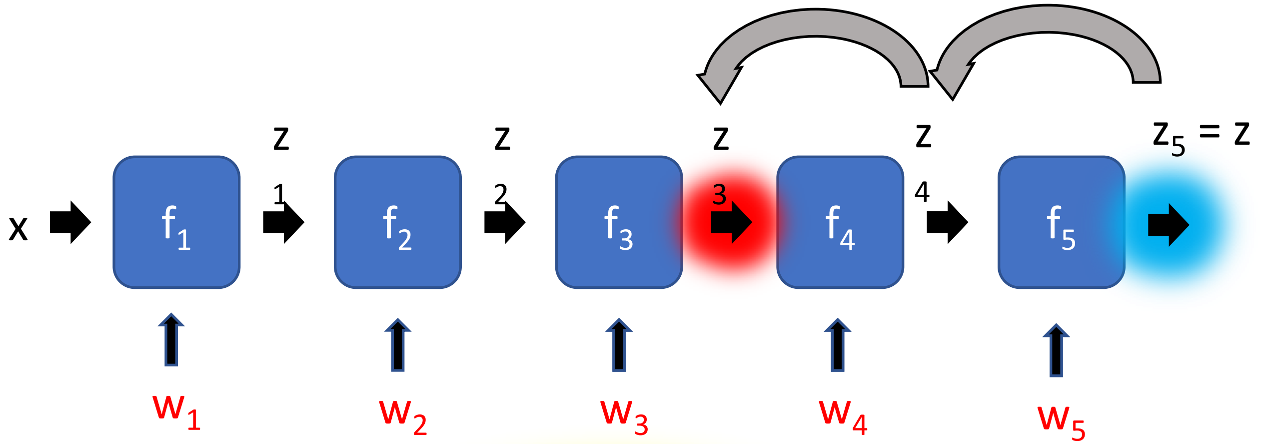
$$\frac{\partial z}{\partial w_3} = \frac{\partial z}{\partial z_3}\frac{\partial z_3}{\partial w_3}$$

The gradient of convnets



$$\frac{\partial z}{\partial w_3} = \frac{\partial z}{\partial z_3} \frac{\partial z_3}{\partial w_3}$$

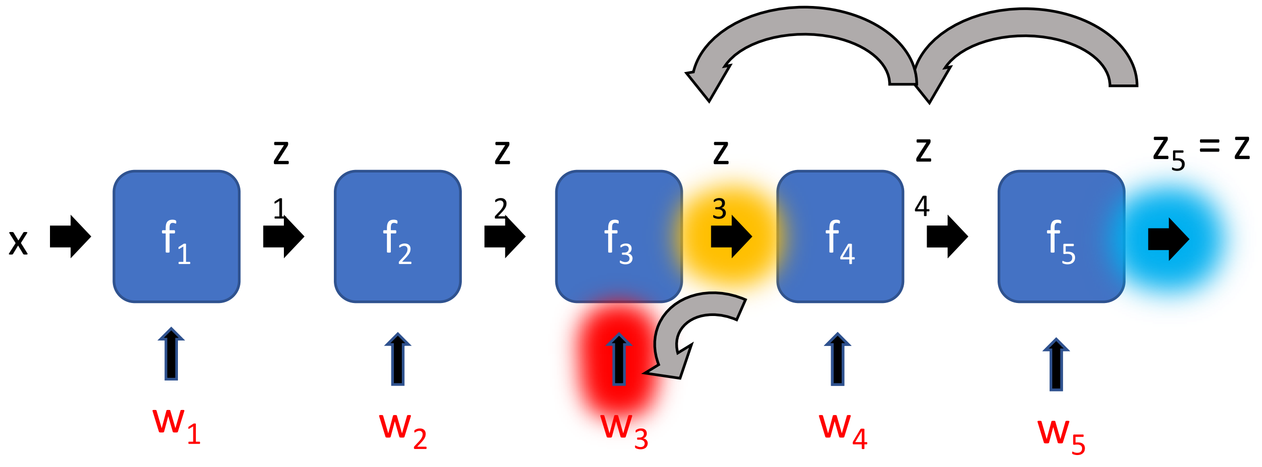
The gradient of convnets



$$\frac{\partial z}{\partial z_3} = \frac{\partial z}{\partial z_4}\frac{\partial z_4}{\partial z_3}$$

$$\frac{\partial z}{\partial w_3} = \frac{\partial z}{\partial z_3}\frac{\partial z_3}{\partial w_3}$$

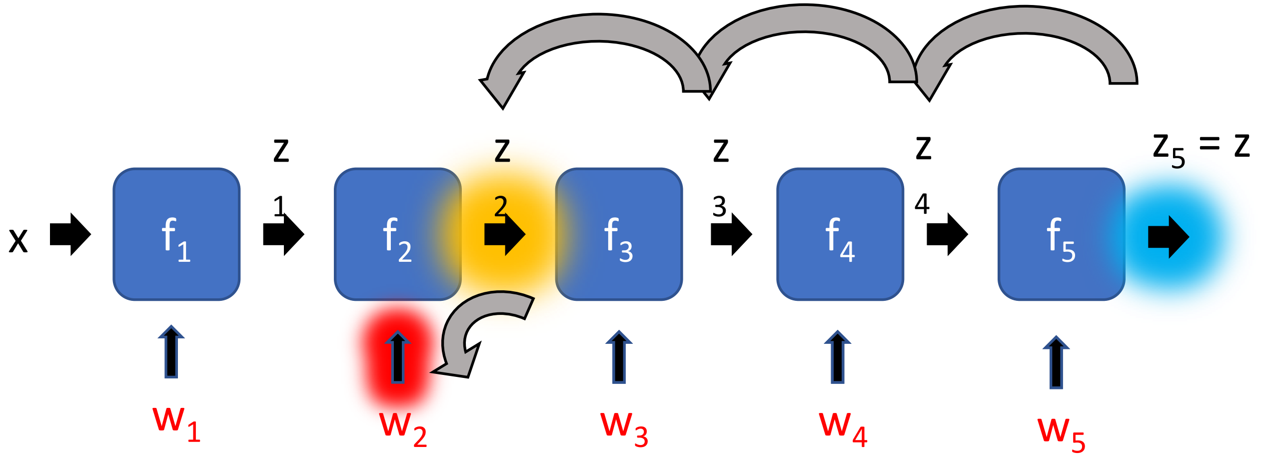
The gradient of convnets



$$\frac{\partial z}{\partial z_3} = \frac{\partial z}{\partial z_4} \frac{\partial z_4}{\partial z_3}$$

$$\frac{\partial z}{\partial w_3} = \frac{\partial z}{\partial z_3} \frac{\partial z_3}{\partial w_3}$$

The gradient of convnets

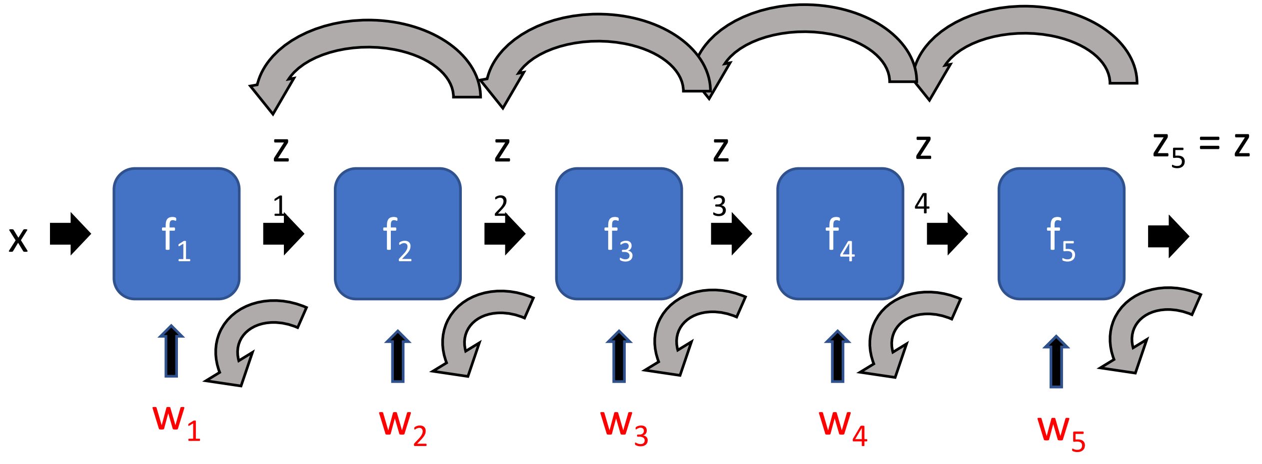


$$\frac{\partial z}{\partial z_2} = \frac{\partial z}{\partial z_3}\frac{\partial z_3}{\partial z_2}$$

$$\frac{\partial z}{\partial w_2} = \frac{\partial z}{\partial z_2}\frac{\partial z_2}{\partial w_2}$$

Recurrence going backward!!

The gradient of convnets



Backpropagation

Backpropagation for a sequence of functions

$$z_i = f_i(z_{i-1}, w_i)$$

$$z_0 = x$$

$$z = z_n$$

Previous term

$$\frac{\partial z}{\partial z_i} = \frac{\partial z}{\partial z_{i+1}} \frac{\partial z_{i+1}}{\partial z_i}$$

Function derivative

$$\frac{\partial z}{\partial w_i} = \frac{\partial z}{\partial z_i} \frac{\partial z_i}{\partial w_i}$$

Backpropagation for a sequence of functions

$$z_i = f_i(z_{i-1}, w_i) \qquad z_0 = x \qquad z = z_n$$

- Assume we can compute partial derivatives of each function

$$\frac{\partial z_i}{\partial z_{i-1}} = \frac{\partial f_i(z_{i-1}, w_i)}{\partial z_{i-1}} \qquad \frac{\partial z_i}{\partial w_i} = \frac{\partial f_i(z_{i-1}, w_i)}{\partial w_i}$$

- Use $g(z_i)$ to store gradient of z w.r.t z_i, $g(w_i)$ for w_i
- Calculate $g(z_i)$ by iterating backwards

$$g(z_n) = \frac{\partial z}{\partial z_n} = 1 \qquad g(z_{i-1}) = \frac{\partial z}{\partial z_i}\frac{\partial z_i}{\partial z_{i-1}} = g(z_i)\frac{\partial z_i}{\partial z_{i-1}}$$

- Use $g(z_i)$ to compute gradient of parameters $g(w_i)$

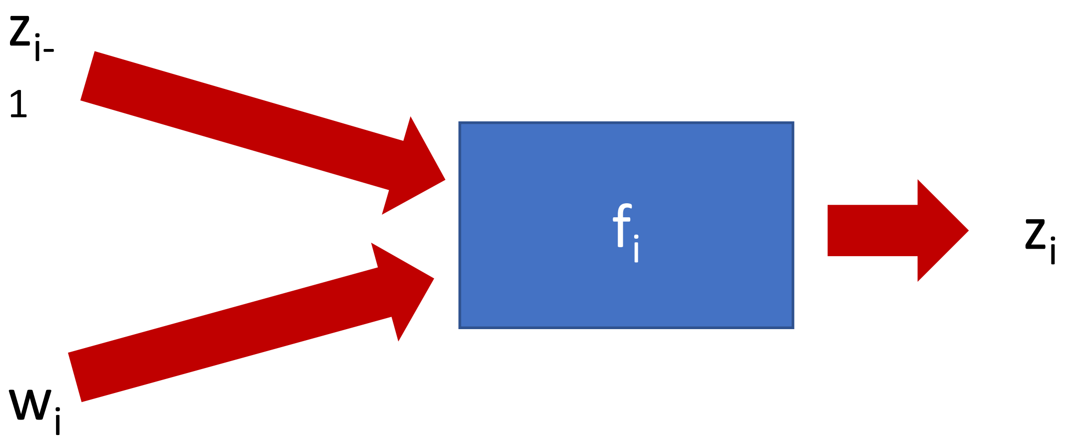
$$g(w_i) = \frac{\partial z}{\partial z_i}\frac{\partial z_i}{\partial w_i} = g(z_i)\frac{\partial z_i}{\partial w_i}$$

Backpropagation for a sequence of functions

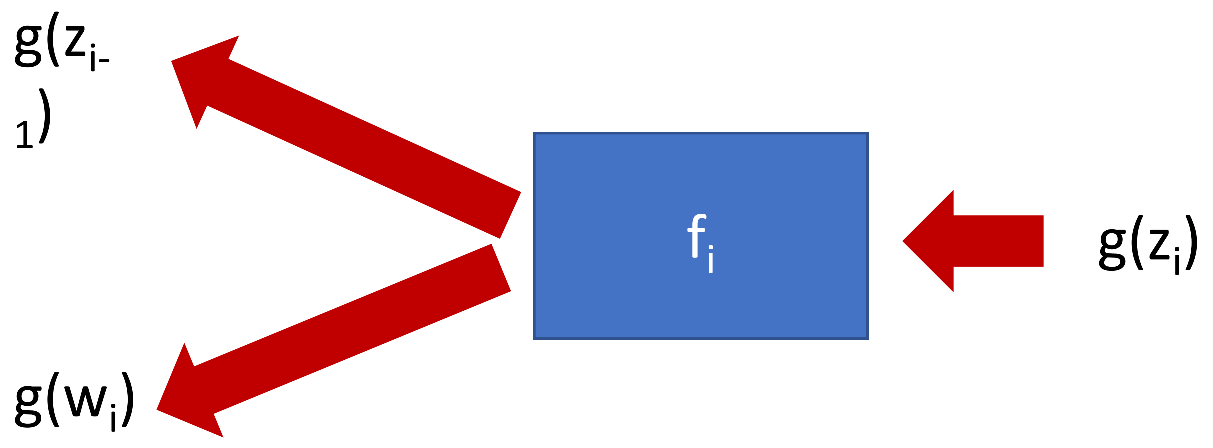
- Each “function” has a “forward” and “backward” module
- Forward module for f_i
  - takes z_{i-1} and weight w_i as input
  - produces z_i as output
- Backward module for f_i
  - takes $g(z_i)$ as input
  - produces $g(z_{i-1})$ and $g(w_i)$ as output

$$g(z_{i-1}) = g(z_i)\frac{\partial z_i}{\partial z_{i-1}} \qquad g(w_i) = g(z_i)\frac{\partial z_i}{\partial w_i}$$

Backpropagation for a sequence of functions



Backpropagation for a sequence of functions



Chain rule for vectors

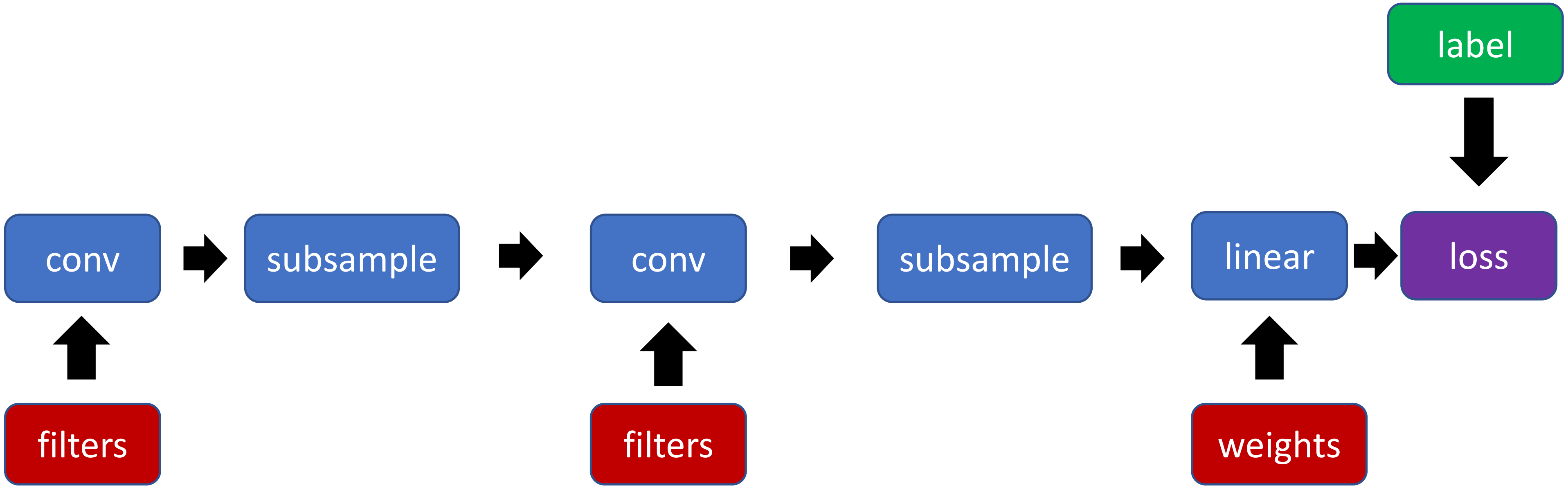
$$\frac{\partial a}{\partial b} = \frac{\partial a}{\partial c}\frac{\partial c}{\partial b} \qquad \frac{\partial a_i}{\partial b_j} = \sum_k \frac{\partial a_i}{\partial c_k}\frac{\partial c_k}{\partial b_j}$$

$$\frac{\partial \mathbf{a}}{\partial \mathbf{b}}(i,j) = \frac{\partial a_i}{\partial b_j}$$

Jacobian

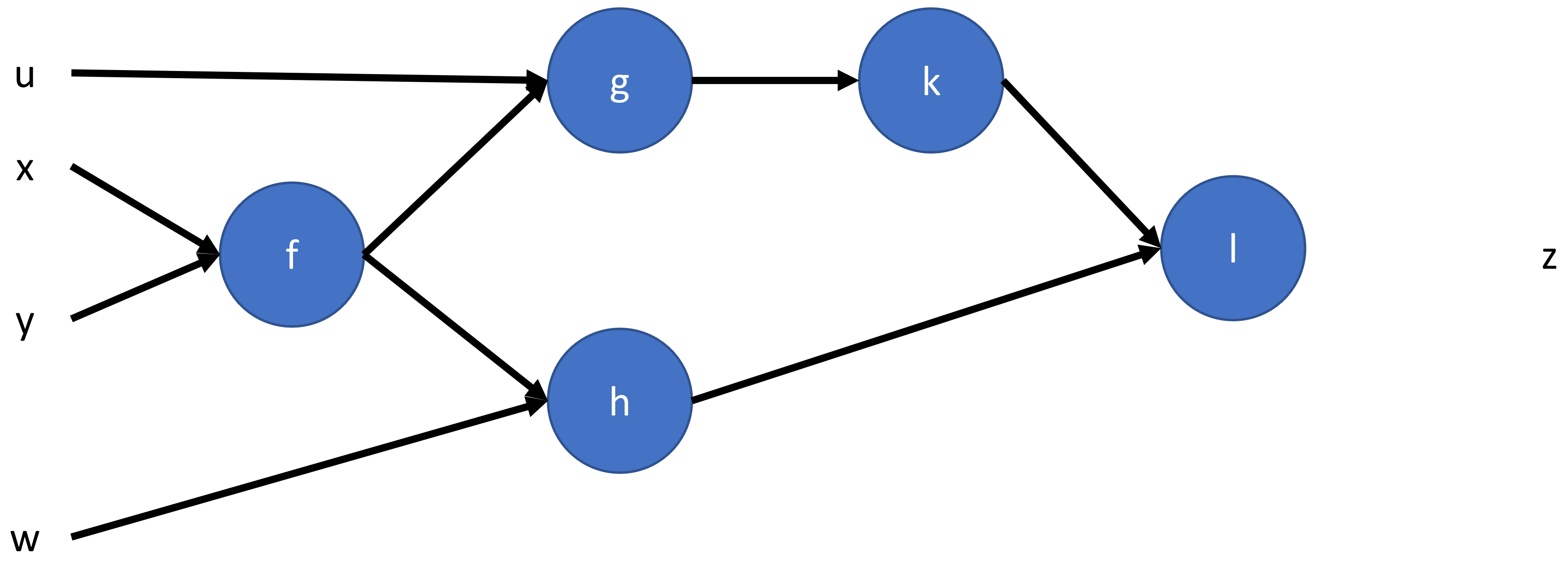
$$\frac{\partial \mathbf{a}}{\partial \mathbf{b}} = \frac{\partial \mathbf{a}}{\partial \mathbf{c}}\frac{\partial \mathbf{c}}{\partial \mathbf{b}}$$

Loss as a function



Beyond sequences: computation graphs

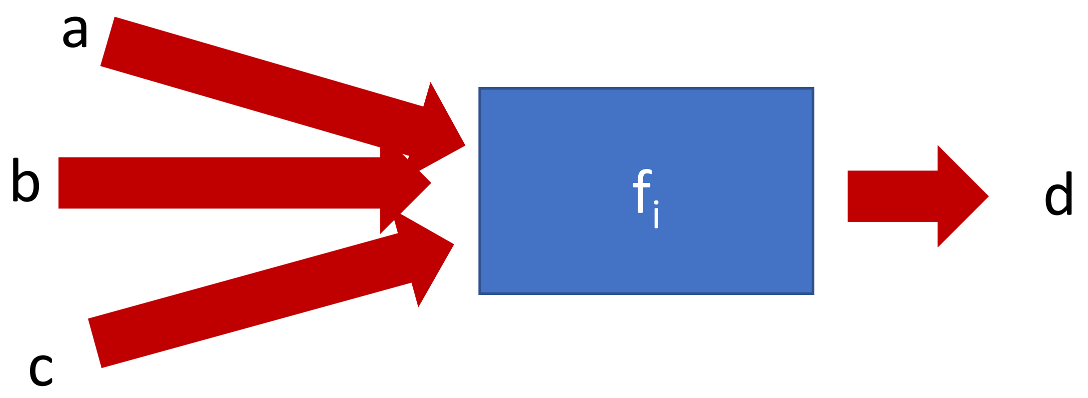
- Arbitrary *graphs* of functions
- No distinction between intermediate outputs and parameters



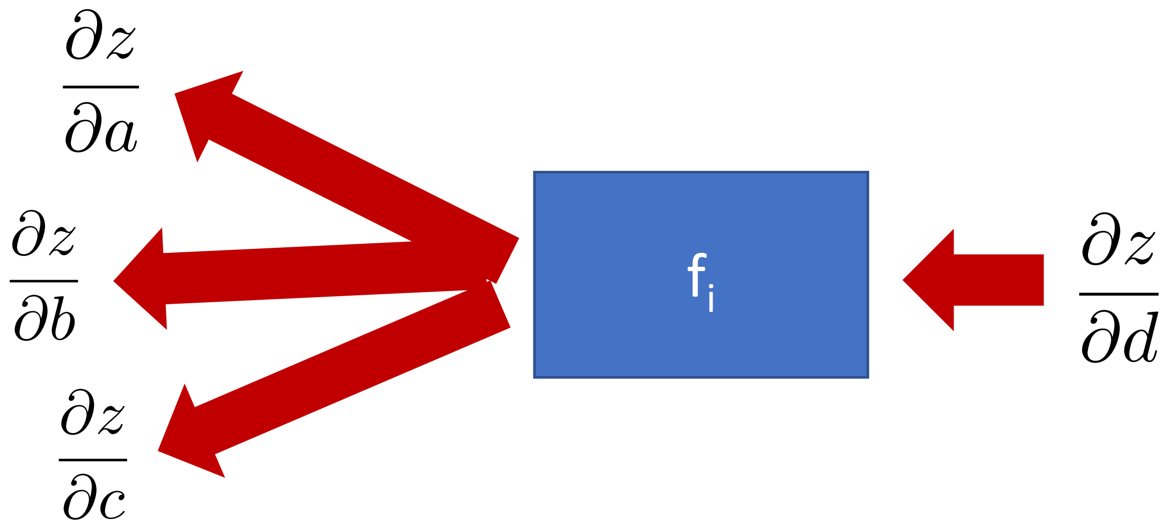
Computation graph - Functions

- Each node implements two functions
  - A “forward”
    - Computes output given input
  - A “backward”
    - Computes derivative of z w.r.t input, given derivative of z w.r.t output

Computation graphs



Computation graphs



Stochastic gradient descent

- Gradient on single example = unbiased sample of true gradient
- Idea: at each iteration sample single example $x^{(t)}$

$$\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} - \lambda \nabla_{\mathbf{w}} L(h_{\mathbf{w}}(x^{(t)}), y^{(t)})$$

step size

- Con: variance in estimate of gradient → slow convergence, jumping near optimum

Minibatch stochastic gradient descent

- Compute gradient on a small batch of examples
- Same mean (=true gradient), but variance inversely proportional to minibatch size

$$\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} - \lambda \frac{1}{|B^{(t)}|} \sum_{(x,y)\in B^{(t)}} \nabla_{\mathbf{w}} L(h_{\mathbf{w}}(x), y)$$

Momentum

- *Average* multiple gradient steps
- Use *exponential averaging*

$$\mathbf{p}^{(t+1)} \leftarrow (1-\mu)\mathbf{p}^{(t)} - \mu \frac{1}{|B^{(t)}|} \sum_{(x,y)\in B^{(t)}} \nabla_{\mathbf{w}} L(h_{\mathbf{w}}(x), y)$$

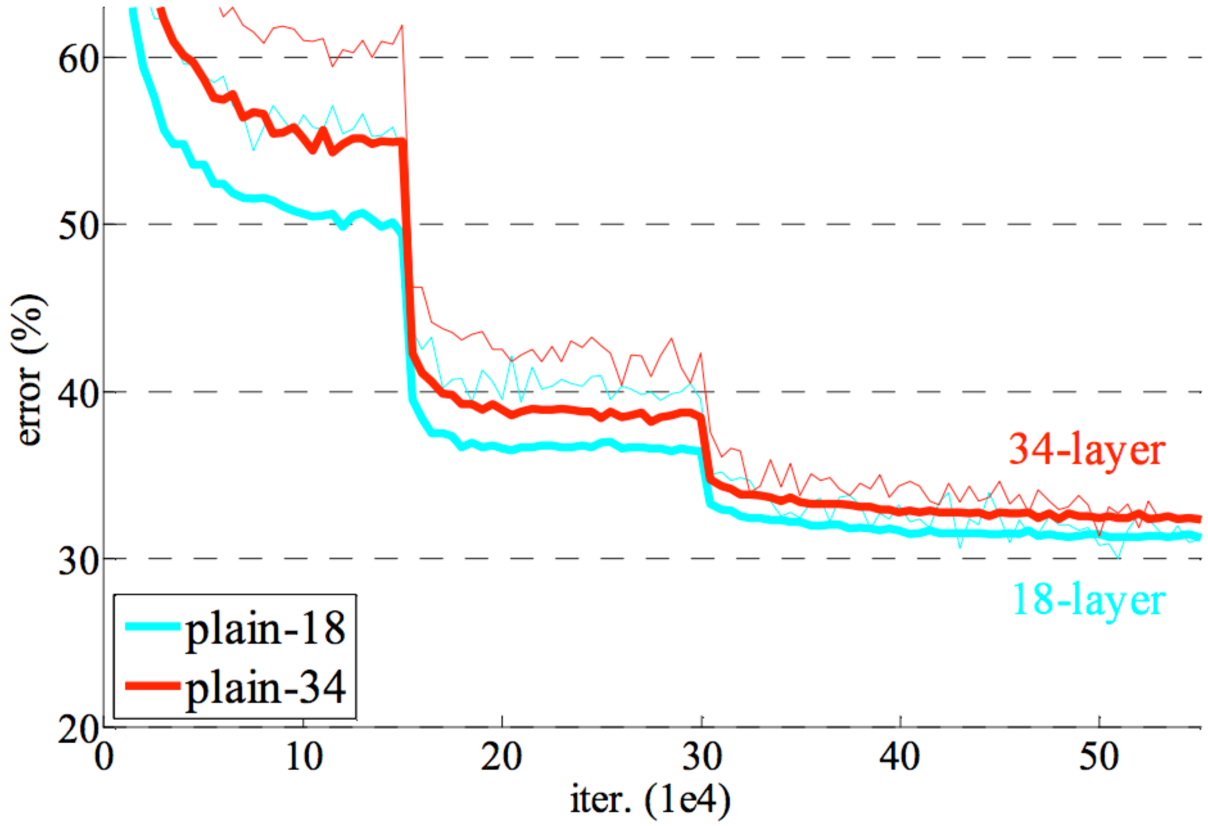
$$\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} - \lambda \mathbf{p}^{(t+1)}$$

Weight decay

- Add $-a\mathbf{w}^{(t)}$ to the gradient
- Prevents $\mathbf{w}^{(t)}$ from growing to infinity
- Equivalent to L2 regularization of weights

Learning rate decay

- Large step size / learning rate
  - Faster convergence initially
  - Bouncing around at the end because of noisy gradients
- Learning rate must be decreased over time
- Usually done in steps



Convolutional network training

- Initialize network
- Sample *minibatch* of images
- Forward pass to compute loss
- Backpropagate loss to compute gradient
- Combine gradient with momentum and weight decay
- Take step according to current learning rate