Chain Rule Review

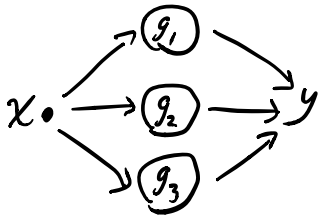
Scalar valued functions of scalars

$$y(x) = f(g(h(x)))$$
$$y'(x) = f'(g(h(x)))\, g'(h(x))\, h'(x)$$

Scalar output

$$y(x) = f(g_1(x), g_2(x), g_3(x))$$
$$y'(x) = \frac{\partial f}{\partial g_1}\frac{dg_1}{dx} + \frac{\partial f}{\partial g_2}\frac{dg_2}{dx} + \frac{\partial f}{\partial g_3}\frac{dg_3}{dx}$$

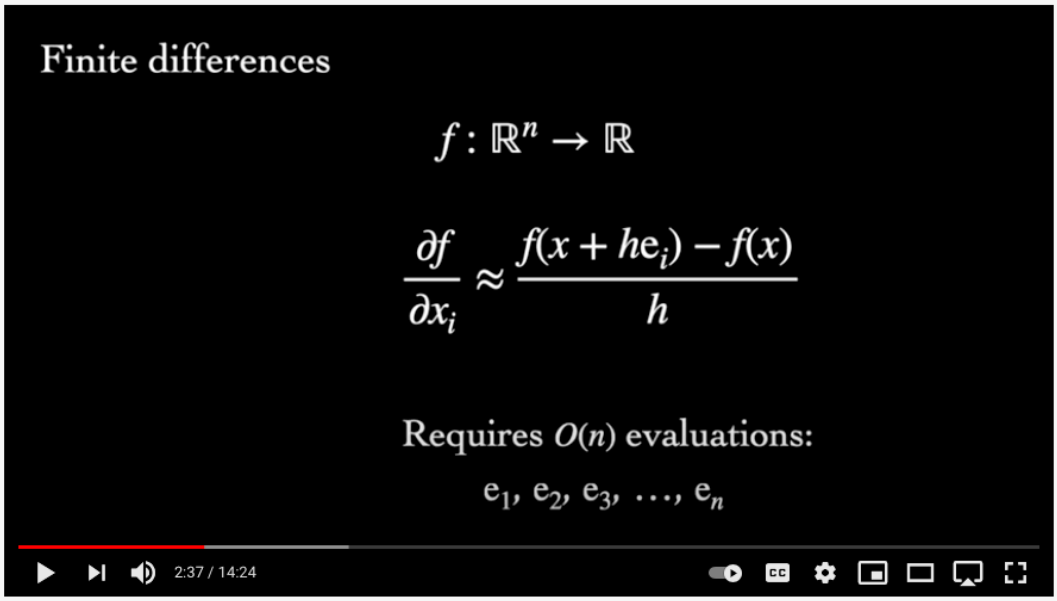
Influence Diagram



Scalar Output Multivariate input

$$f(x_1, x_2, x_3)$$
$$\nabla f = \left(\partial_{x_1} f, \partial_{x_2} f, \partial_{x_3} f\right)$$

https://www.youtube.com/watch?v=wG_nF1awSSY



What is Automatic Differentiation?

What are the benefits/challenges with computing gradients by finite differences?

Benefit

Only need black box access to f

Challenge

Choose value of h

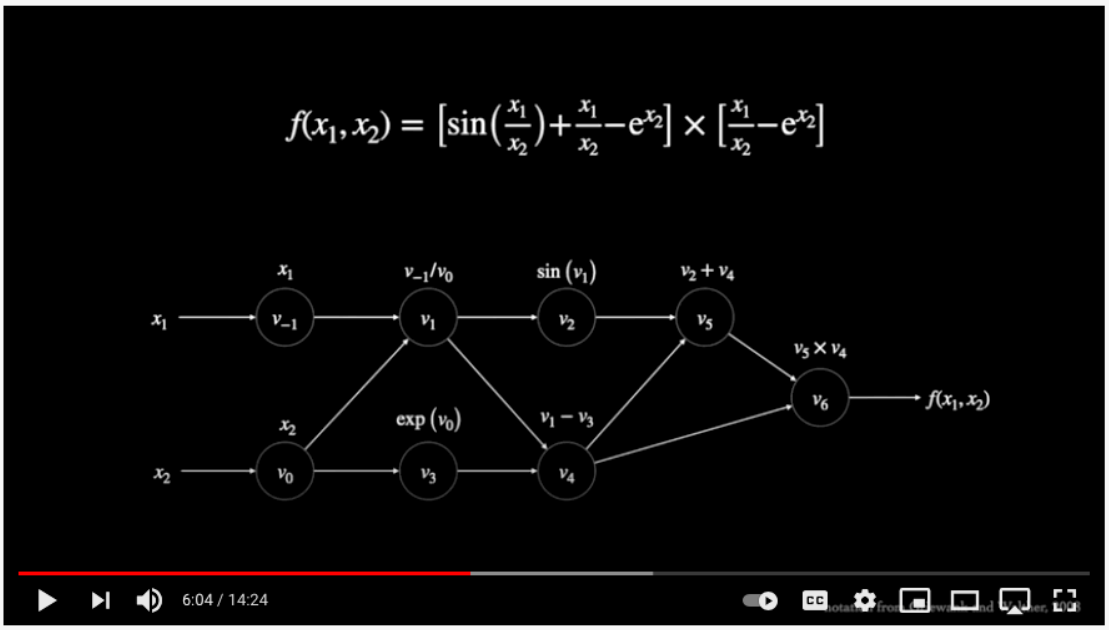
Numerical instability if h small

inaccurate if h large

To find $\nabla_x f$, compute $\frac{\partial f}{\partial x_i}$ for $i = 1 \cdots n$

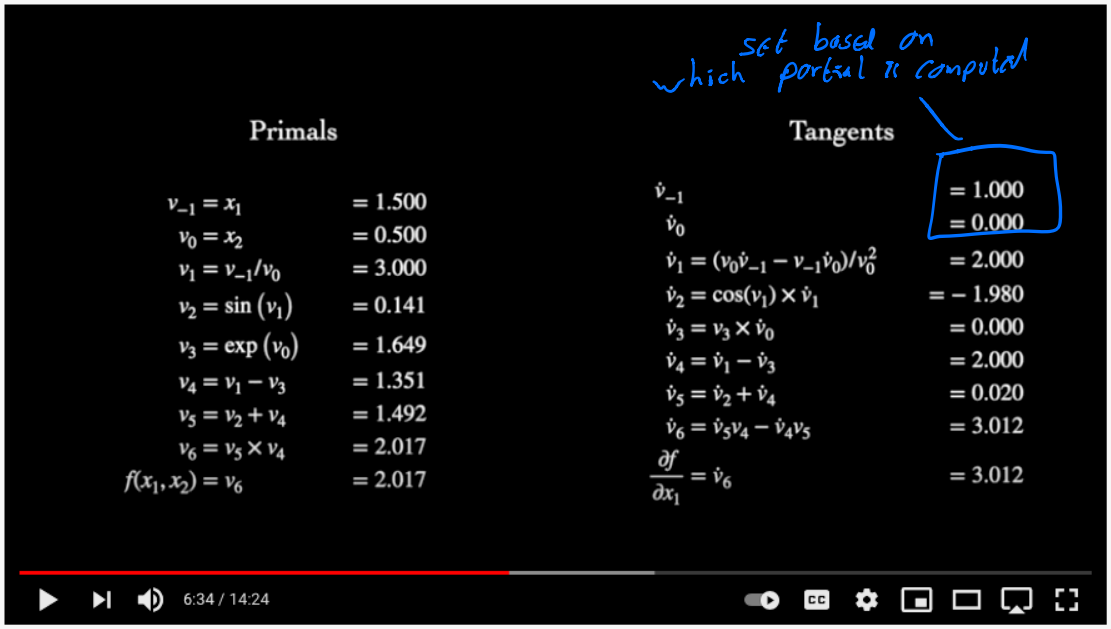
$n+1$ forward passes of f

Computational Graphs + Forward Mode Auto. Diff.

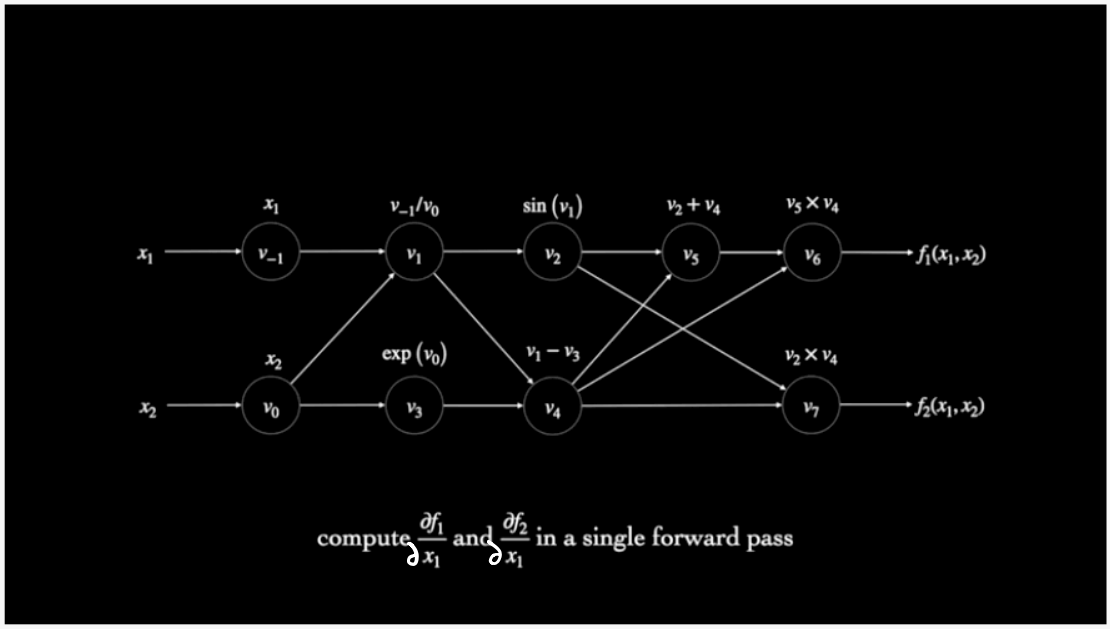


What is Automatic Differentiation?

$\frac{\partial f}{\partial x_1}$

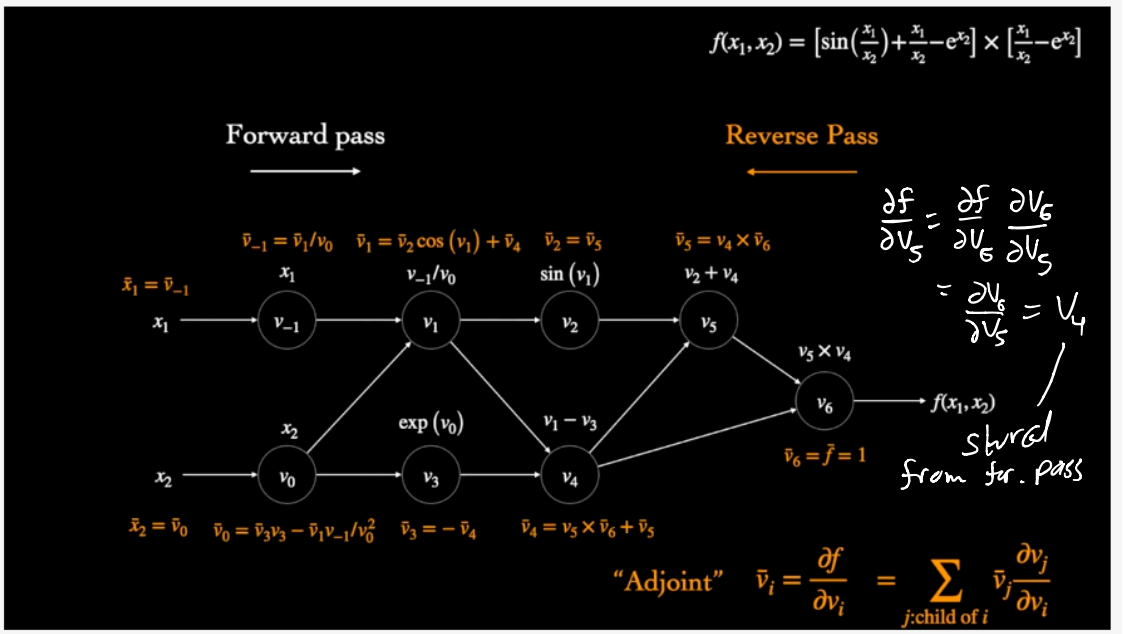


What is Automatic Differentiation?



What is Automatic Differentiation?

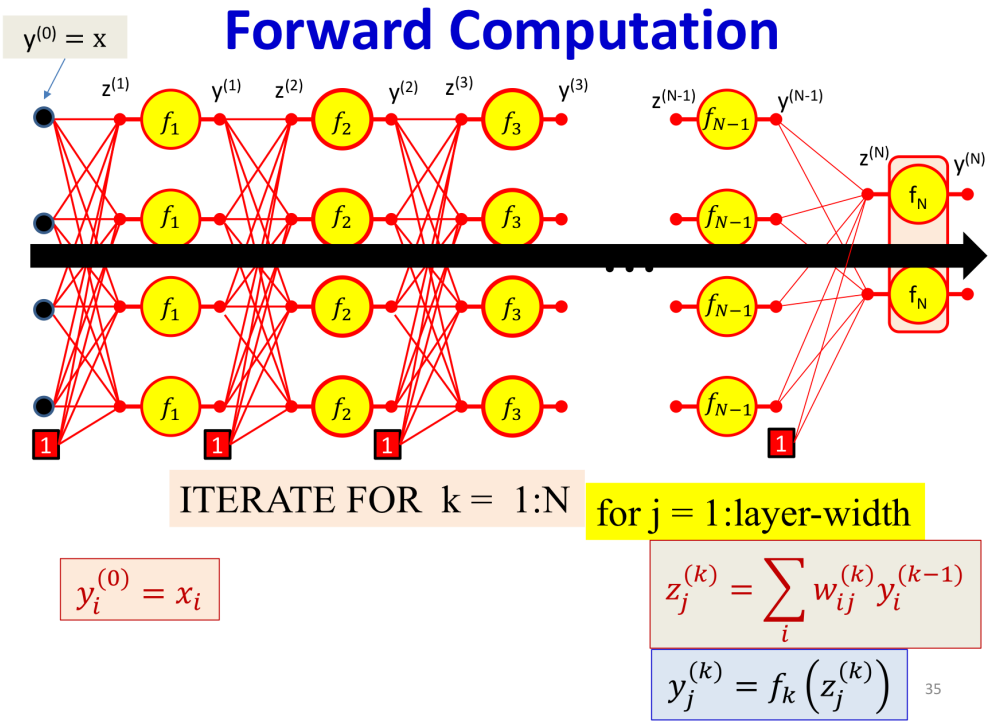
Benefits of Forward Mode Auto. Diff.



What is Automatic Differentiation?

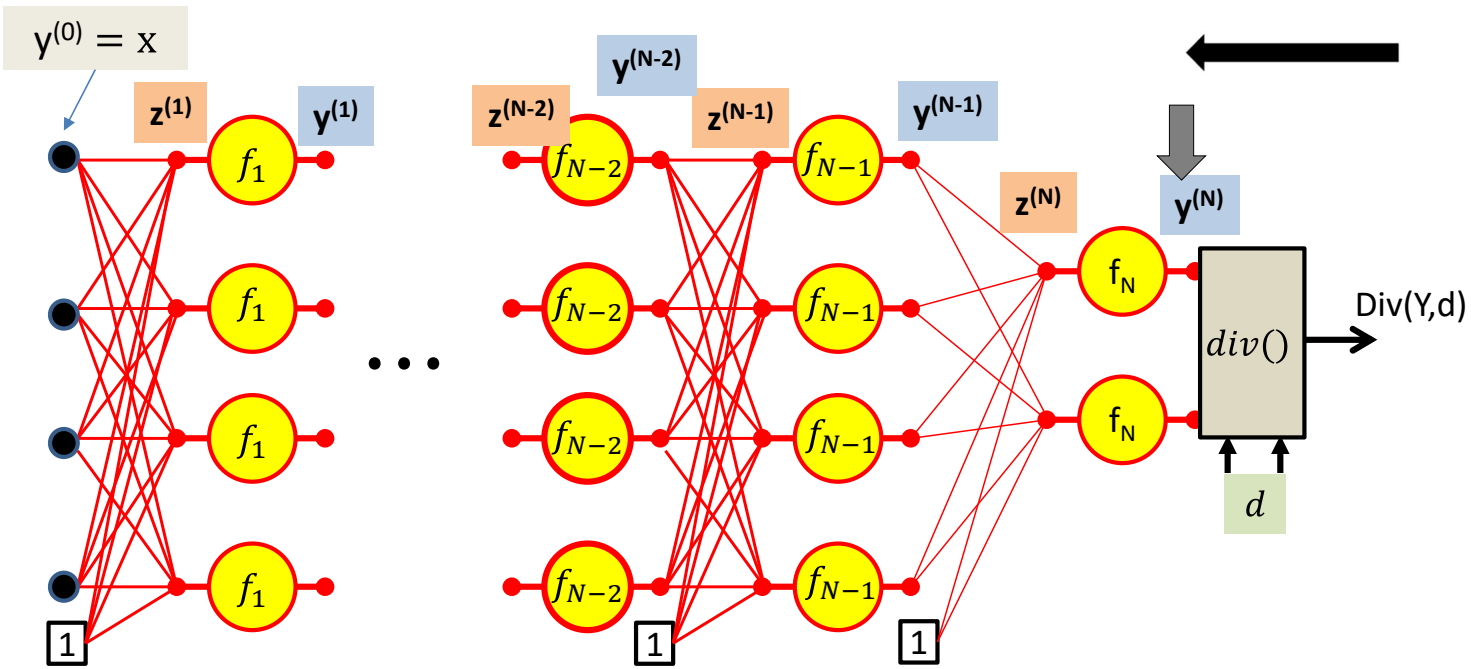
In forward pass, calculate activation of all neurons. We store those activations. In the backward pass, we will use these values to compute partial derivatives.

Reverse Mode Auto. Diff.



http://deeplearning.cs.cmu.edu/S21/index.html

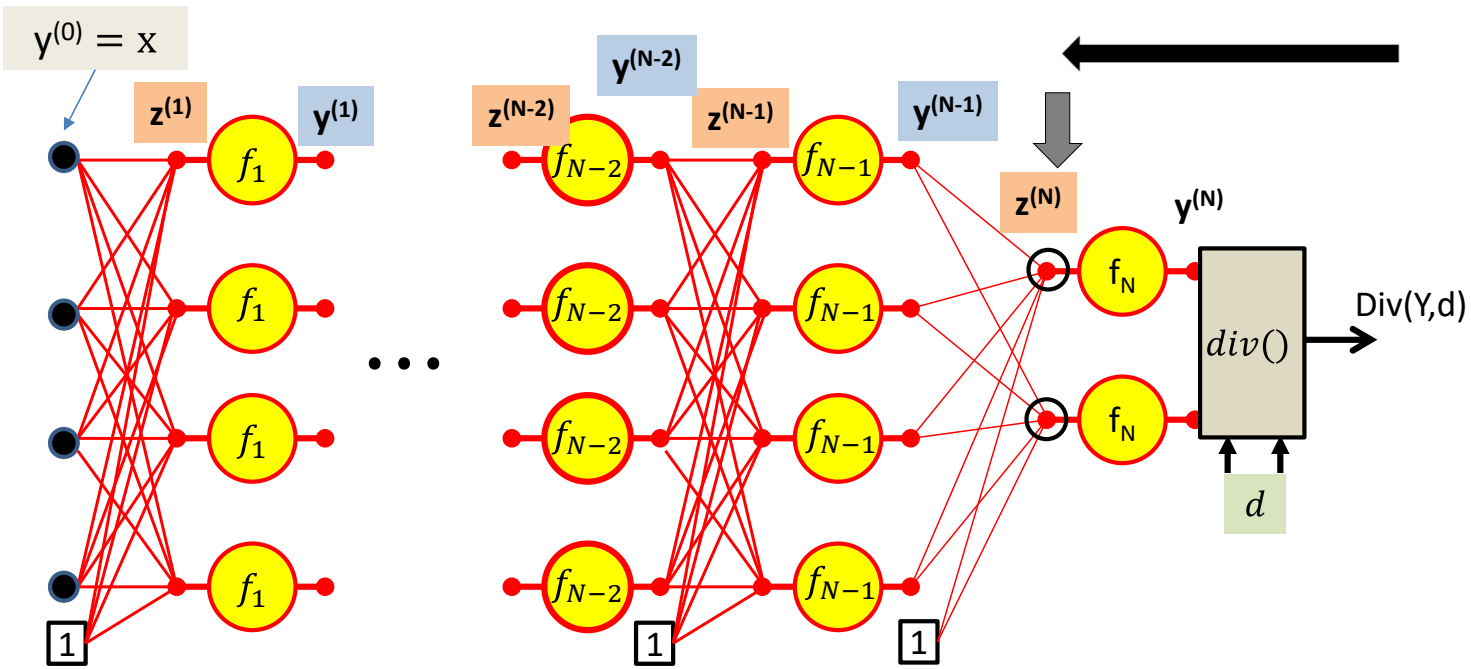
Computing derivatives



The derivative w.r.t the actual output of the final layer of the network is simply the derivative w.r.t to the output of the network

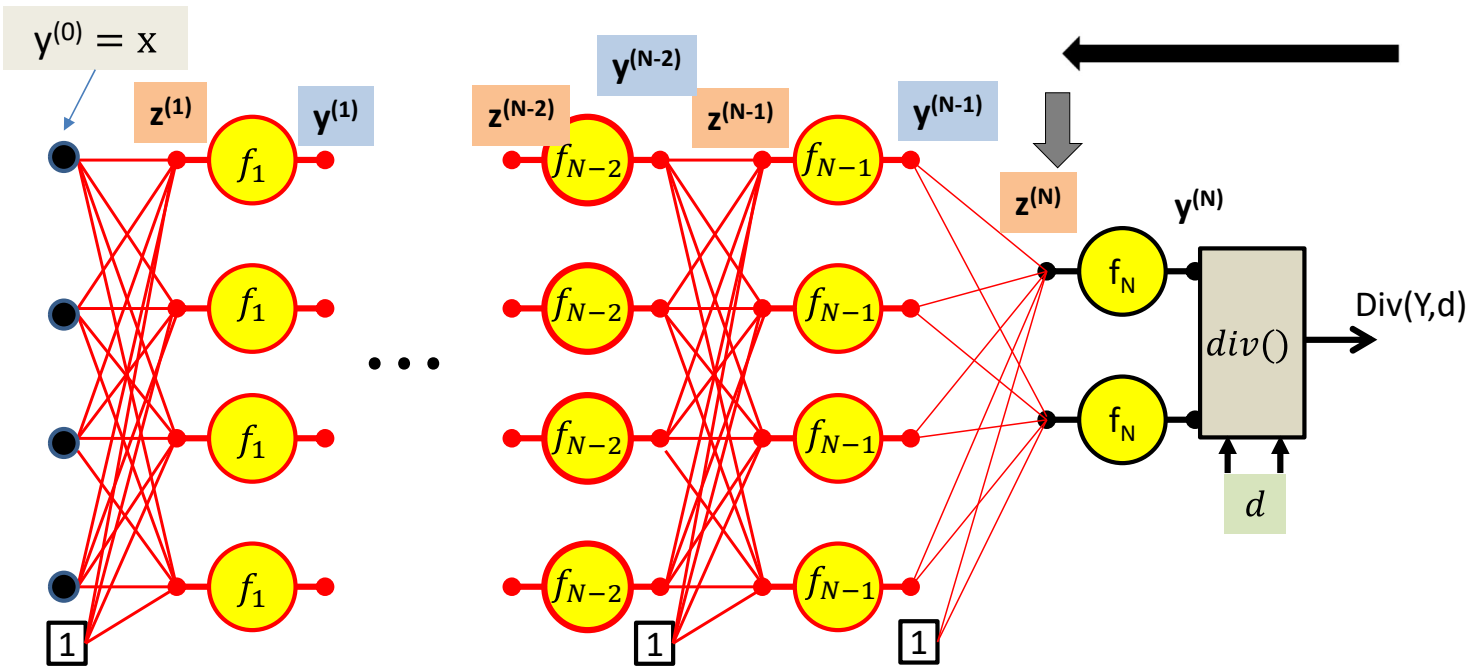
$$\frac{\partial Div(Y,d)}{\partial y_i^{(N)}} = \frac{\partial Div(Y,d)}{\partial y_i}$$

Computing derivatives



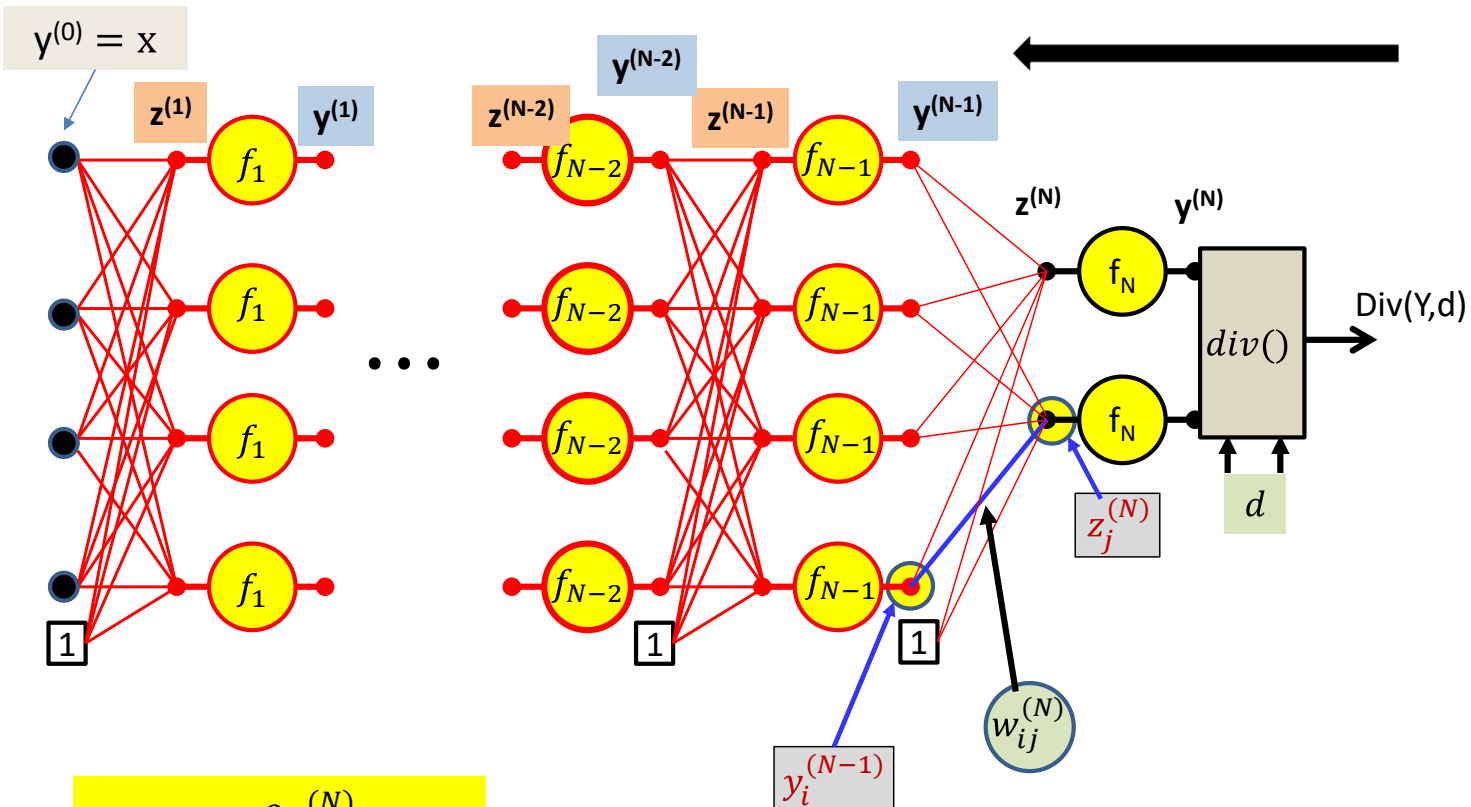
$$\frac{\partial Div}{\partial z_i^{(N)}} = \frac{\partial y_i^{(N)}}{\partial z_i^{(N)}} \frac{\partial Div}{\partial y_i^{(N)}}$$

Computing derivatives



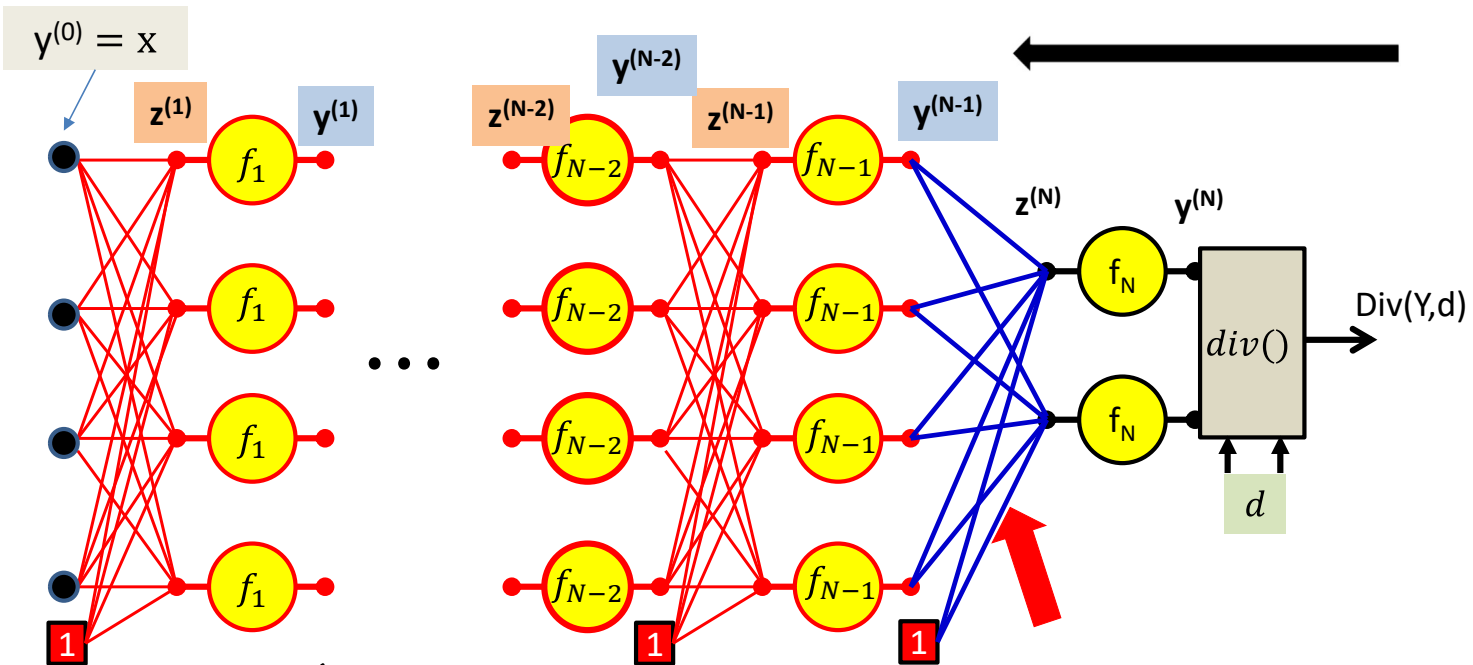
$$\frac{\partial Div}{\partial z_i^{(N)}} = f_N'\left(z_i^{(N)}\right)\frac{\partial Div}{\partial y_i^{(N)}}$$

Computing derivatives



$$\frac{\partial Div}{\partial w_{ij}^{(N)}} = \frac{\partial z_j^{(N)}}{\partial w_{ij}^{(N)}} \frac{\partial Div}{\partial z_j^{(N)}}$$

Computing derivatives



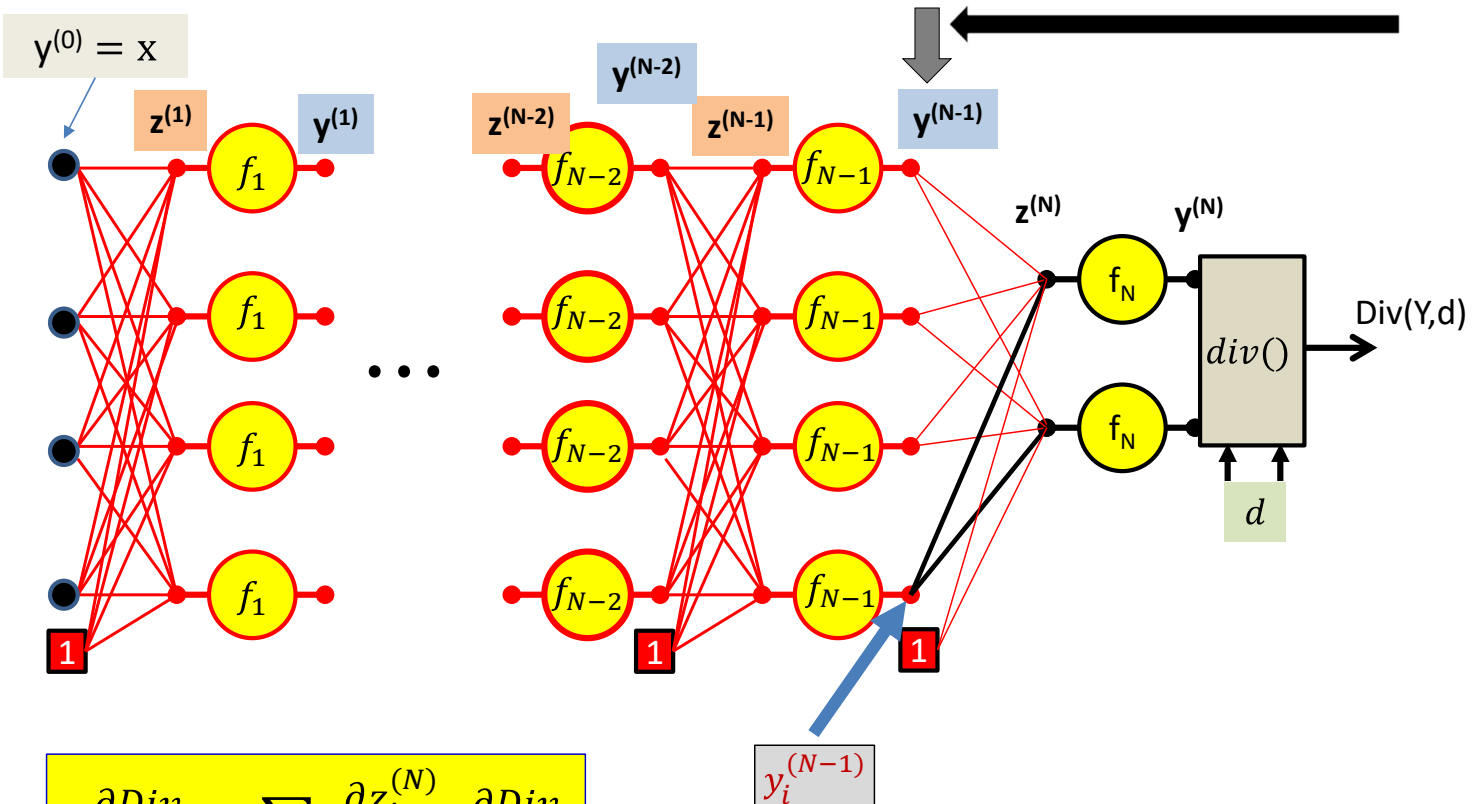
activ. value

$$\frac{\partial Div}{\partial w_{ij}^{(N)}} = y_i^{(N-1)} \frac{\partial Div}{\partial z_j^{(N)}}$$

$\frac{\partial}{\partial w}(w \cdot y_i)$

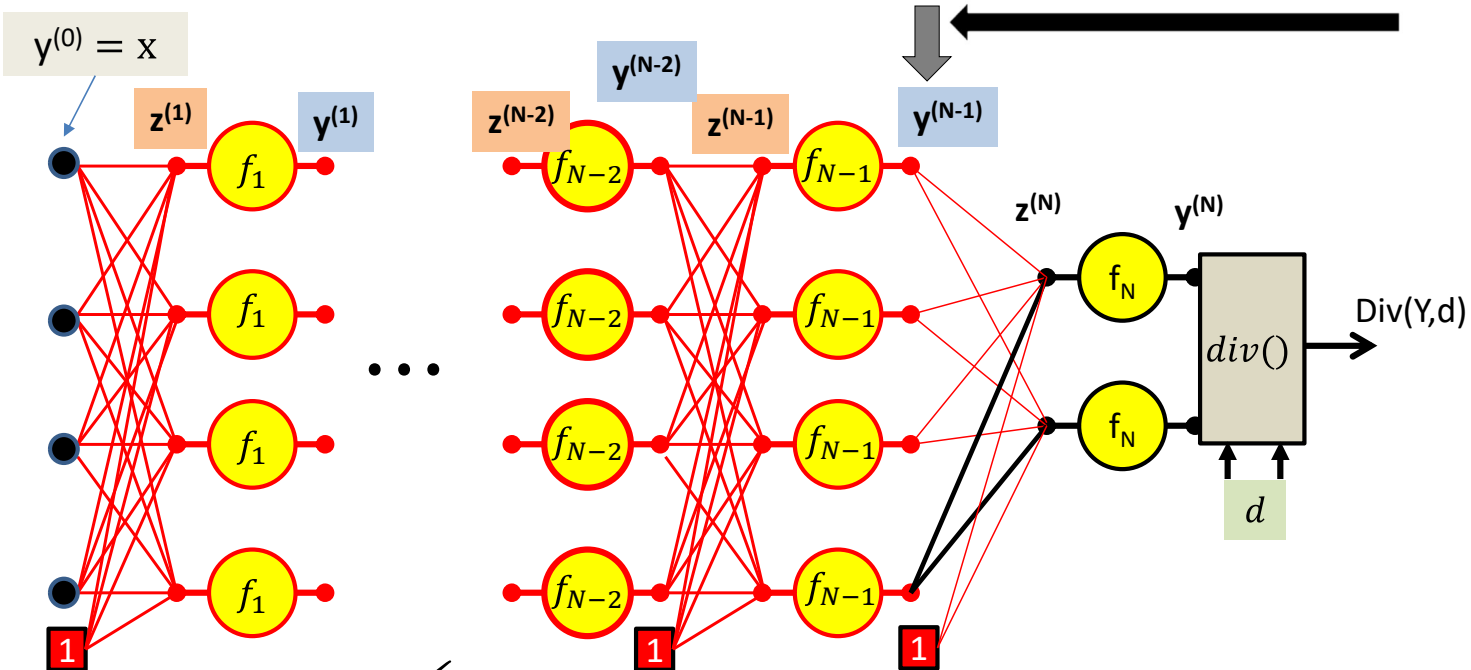
For the bias term $y_0^{(N-1)} = 1$

Computing derivatives



$$\frac{\partial Div}{\partial y_i^{(N-1)}} = \sum_j \frac{\partial z_j^{(N)}}{\partial y_i^{(N-1)}} \frac{\partial Div}{\partial z_j^{(N)}}$$

Computing derivatives



$$\frac{\partial Div}{\partial y_i^{(N-1)}} = \sum_j w_{ij}^{(N)} \frac{\partial Div}{\partial z_j^{(N)}}$$

$\partial_y (w \cdot y)$

Gradients: Backward Computation

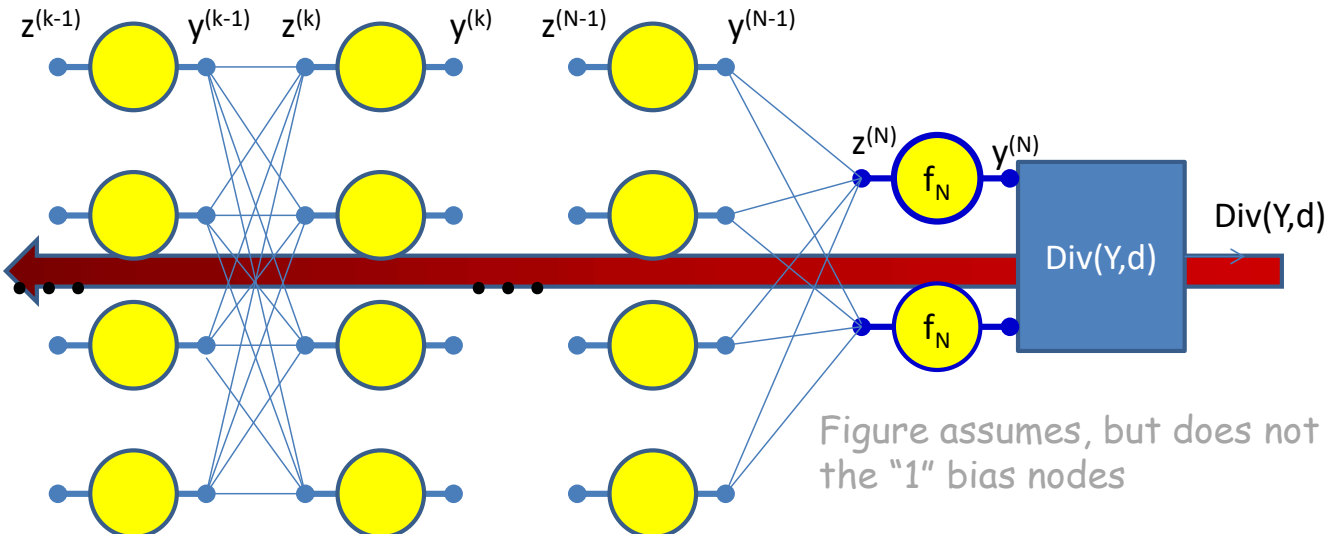


Figure assumes, but does not show the "1" bias nodes

Initialize: Gradient w.r.t network output

$$\frac{\partial Div}{\partial y_i^{(N)}} = \frac{\partial Div(Y,d)}{\partial y_i}$$

$$\frac{\partial Div}{\partial z_i^{(N)}} = f_k'\left(z_i^{(N)}\right)\frac{\partial Div}{\partial y_i^{(N)}}$$

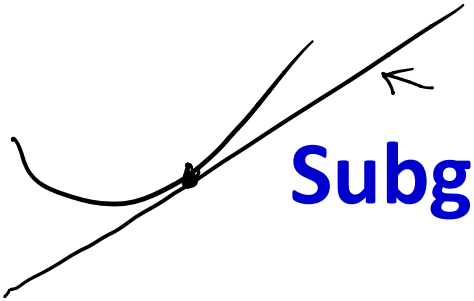
$For\ k = N-1..0$

$For\ i = 1: layer\ width$

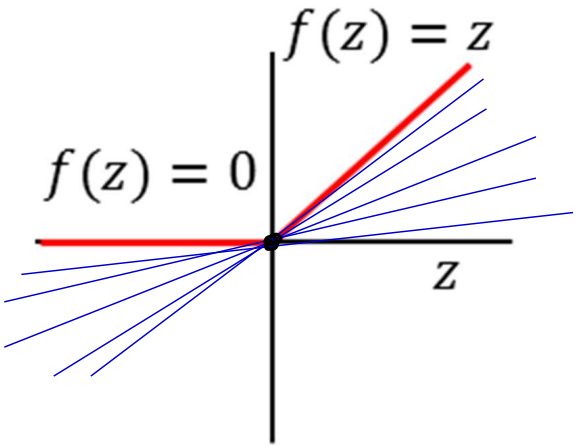
$$\frac{\partial Div}{\partial y_i^{(k)}} = \sum_j w_{ij}^{(k+1)}\frac{\partial Div}{\partial z_j^{(k+1)}}$$

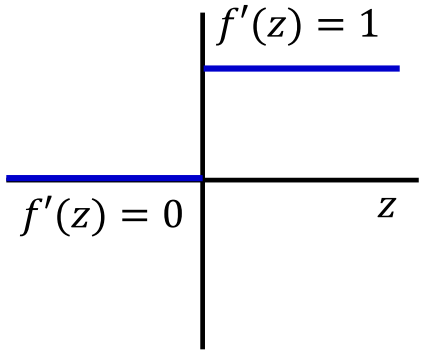
$$\frac{\partial Div}{\partial z_i^{(k)}} = f_k'\left(z_i^{(k)}\right)\frac{\partial Div}{\partial y_i^{(k)}}$$

$$\forall j\ \frac{\partial Div}{\partial w_{ij}^{(k+1)}} = y_i^{(k)}\frac{\partial Div}{\partial z_j^{(k+1)}}$$



Subgradients and the RELU





$$f'(z) = \begin{cases} 0, & z < 0 \\ 1, & z \geq 0 \end{cases}$$

- The *subderivative* of a RELU is the slope of any line that lies entirely under it
  - The subgradient is a generalization of the subderivative
  - At the differentiable points on the curve, this is the same as the gradient
- Can use any subgradient at 0
  - Typically, will use the equation given

Consider a network from $\mathbb{R}^2 \to \mathbb{R}^{1000} \to \mathbb{R}^{1000} \to \mathbb{R}$ (input)

Would computing the gradient with respect to the network weights be orders of magnitude slower than computing the gradient with respect to the input?

Must you computer the derivative wrt weights in order to compute derivative wrt the input?
No. The boxes in red below don't need the values in the yellow box

How many flops? →

How many flops? →

For $k = N-1..0$

For $i = 1: layer\ width$

$$\frac{\partial Div}{\partial y_i^{(k)}} = \sum_j w_{ij}^{(k+1)} \frac{\partial Div}{\partial z_j^{(k+1)}} \qquad \frac{\partial Div}{\partial z_i^{(k)}} = f_k'\left(z_i^{(k)}\right) \frac{\partial Div}{\partial y_i^{(k)}}$$

$$\forall j \quad \frac{\partial Div}{\partial w_{ij}^{(k+1)}} = y_i^{(k)} \frac{\partial Div}{\partial z_j^{(k+1)}}$$

78

$\sim 10^6$ flops

$\sim 10^6$ flops

Multivariate input & output

$$f : \mathbb{R}^n \to \mathbb{R}^m$$

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

$$J_{f,ij} =$$

Chain rule - multivariate input & output

$$f : \mathbb{R}^n \to \mathbb{R}^m$$
$$g : \mathbb{R}^p \to \mathbb{R}^n$$
$$f \circ g : \mathbb{R}^p \to \mathbb{R}^m$$

$$y(x) = f(g(x))$$

$$J_y(x) = \underbrace{\underbrace{J_f(g(x))}_{\mathbb{R}^{m \times n}} \underbrace{J_g(x)}_{\mathbb{R}^{n \times p}}}_{\mathbb{R}^{m \times p}}$$

Example

$$z(y) = Wy + b$$

$W \in \mathbb{R}^{m \times n}$, $y \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, $z \in \mathbb{R}^m$

$$J_z(y) = W$$

Example:

Let $f: \mathbb{R}^m \to \mathbb{R}$ $\quad W \in \mathbb{R}^{m \times n}$ $\quad y \in \mathbb{R}^n$

$$D = f(\underbrace{Wy + b}_{z})$$

$$J_D(y) = J_f(z) W$$

$$J_D(b) = J_f(z)$$

$$J_D(W) = J_f^t(z)\, y^t$$