CS 4803 / 7643: Deep Learning

Topics:

- Automatic Differentiation
  - (Finish) Forward mode vs Reverse mode AD
  - Patterns in backprop
  - Jacobians in FC+ReLU NNs

Dhruv Batra

Georgia Tech

Administrativia

- HW1 Reminder
  - Due: 09/26, 11:55pm

Project

- Goal
  - Chance to take on something open-ended
  - Encouraged to apply to your research (computer vision, NLP, robotics,…)

- Main categories
  - Application/Survey
    - Compare a collection of existing algorithms on a new application domain of your interest
  - Formulation/Development
    - Formulate a new model or algorithm for a new or old problem
  - Theory
    - Theoretically analyze an existing algorithm

Project

- Rules
  - **Combine with other classes / research / credits / anything**
    - You have our blanket permission
    - Get permission from other instructors; delineate different parts
  - Must be done this semester.
  - Groups of 3-4

- Expectations
  - 20% of final grade = individual effort equivalent to 1 HW
  - Expectation scales with team size
  - Most work will be done in Nov but please plan early.

Project Ideas

- NeurIPS Reproducibility Challenge
  - https://reproducibility-challenge.github.io/neurips2019/
  - https://reproducibility-challenge.github.io/neurips2019/task/

Reproducibility Challenge   NeurIPS 2019   Task Description   Resources   Registration   Important Dates   Organizers

Reproducibility Challenge @ NeurIPS 2019

The Annual Machine Learning Reproducibility Challenge

Computing

- Major bottleneck
  - GPUs

- Options
  - Your own / group / advisor's resources
  - Google Cloud Credits
    - $50 credits to every registered student courtesy Google
  - Google Colab
    - jupyter-notebook + free GPU instance

Administrativia

- Project Teams Google Doc
  - https://docs.google.com/spreadsheets/d/1ouD6ctaemV_3nb2MQHs7rUOAaW9DFLu8I5Zd3yOFs7E/edit?usp=sharing
  - Project Title
  - 1-3 sentence project summary TL;DR
  - Team member names

Recap from last time

How do we compute gradients?

- Analytic or “Manual” Differentiation
- Symbolic Differentiation
- Numerical Differentiation
- Automatic Differentiation
  - Forward mode AD
  - Reverse mode AD
    - aka “backprop”

Chain Rule: Composite Functions

$$\left[L(x) = f(g(x)) = (f \circ g)(x) \right]$$

$$f(x) = g_\ell(g_{\ell-1} \cdots g_1(x))$$

$$L(w) = (g_\ell \circ g_{\ell-1} \cdots \circ g_1)(x)$$

$$\frac{\partial L}{\partial \vec{w}}$$

Chain Rule: Scalar Case

$x \xrightarrow[g(\cdot)]{} z \xrightarrow[f(\cdot)]{} y \rightarrow a$

$x, y, z \in \mathbb{R}^1$

$a \in \mathbb{R}^1$

$= f(\underbrace{g(x)}_{z})$

$$\boxed{\frac{\partial y}{\partial x}} = \frac{\partial y}{\partial z} \cdot \frac{\partial z}{\partial x}$$

scalar prod.

$$\frac{\partial a}{\partial x} = \frac{\partial a}{\partial y} \cdot \boxed{\frac{\partial y}{\partial x}}$$

$$= \frac{\partial a}{\partial y} \cdot \frac{\partial y}{\partial z} \cdot \frac{\partial z}{\partial x}$$

Chain Rule: Vector Case

$$\boxed{\vec{x} \in \mathbb{R}^d} \xrightarrow[\underbrace{\quad}_{\mathbb{R}^d \to \mathbb{R}^m}]{g(\cdot)} \boxed{\vec{z} \in \mathbb{R}^m} \xrightarrow[\underbrace{\quad}_{\mathbb{R}^m \to \mathbb{R}^c}]{f(\cdot)} \vec{y} \in \mathbb{R}^c \to \vec{a}$$

$$\underbrace{\boxed{\frac{\partial \vec{y}}{\partial \vec{x}}}}_{J_{f \circ g}} = \underbrace{\boxed{\frac{\partial \vec{y}}{\partial \vec{z}}}}_{J_f} \cdot \underbrace{\boxed{\frac{\partial \vec{z}}{\partial \vec{x}}}}_{J_g}$$

Matrix Mult

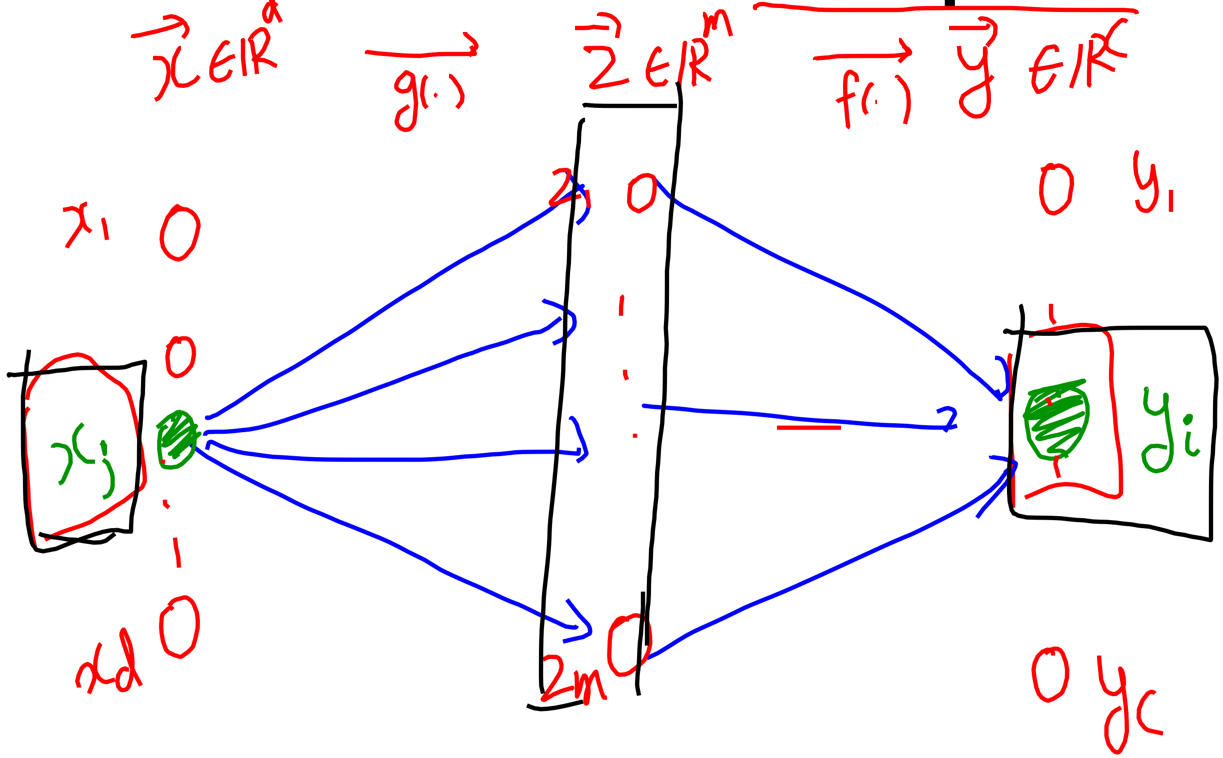
Chain Rule: Jacobian view

$$\underbrace{\frac{\partial \vec{y}}{\partial \vec{x}}}_{c \times d} = \underbrace{\frac{\partial \vec{y}}{\partial \vec{z}}}_{c \times m} \; \underbrace{\frac{\partial \vec{z}}{\partial \vec{x}}}_{m \times d}$$

$$\begin{array}{c} \\ i \end{array}\begin{array}{c} j \\ \left[\; \boxed{\frac{\partial y_i}{\partial x_j}} \; \right]_{c \times d} \end{array} = \begin{array}{c} \\ i \end{array}\left[\; \frac{\partial y_i}{\partial z_k} \; \right]_{c \times m} \begin{array}{c} j \\ \left[\; \frac{\partial z_k}{\partial x_j} \; \right]_{m \times d} \end{array}$$

$$\boxed{\frac{\partial y_i}{\partial x_j}} = \sum_{k} \frac{\partial y_i}{\partial z_k} \, \frac{\partial z_k}{\partial x_j}$$

Chain Rule: Graphical view



$$\frac{\partial y_i}{\partial x_j} = \sum_{\text{paths}} \underbrace{\left(\frac{\partial y_i}{\partial z_k}\right) \cdot \left(\frac{\partial z_k}{\partial x_j}\right)}_{k \text{ is on path}} \cdots$$

Chain Rule: Cascaded

$$\underset{\in \mathbb{R}^d}{\vec{h}^0} \xrightarrow{g_1()} \vec{h}^{(1)} \longrightarrow \vec{h}^2 \cdots\cdots \longrightarrow \vec{h}^{(\ell)} \in \mathbb{R}^d$$

$$\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \xrightarrow{g_{(1)}} \begin{bmatrix} \\ \\ \\ \end{bmatrix} \xrightarrow{g_{(2)}} \begin{bmatrix} \\ \\ \\ \end{bmatrix} \cdots\cdots \longrightarrow \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \xrightarrow{g_{\ell+1}} L \in \mathbb{R}^1$$

$$\frac{\partial L}{\partial \vec{h}^\ell} = [\quad]$$

$$\frac{\partial \vec{h}^{(\ell)}}{\partial \vec{h}^{(0)}} = \frac{\partial L}{\partial \vec{h}^\ell} \frac{\partial \vec{h}^\ell}{\partial \vec{h}^{\ell-1}} \cdot \frac{\partial \vec{h}^{\ell-1}}{\partial \vec{h}^{\ell-2}} \cdots\cdots \frac{\partial \vec{h}^1}{\partial \vec{h}^0}$$

d

$O(\ell d^2)$

$d \times d$

$O(\ell d^3)$

Deep Learning = Differentiable Programming

- Computation = Graph
  - Input = Data + Parameters
  - Output = Loss
  - Scheduling = Topological ordering

- Auto-Diff
  - A family of algorithms for implementing chain-rule on computation graphs

Directed Acyclic Graphs (DAGs)

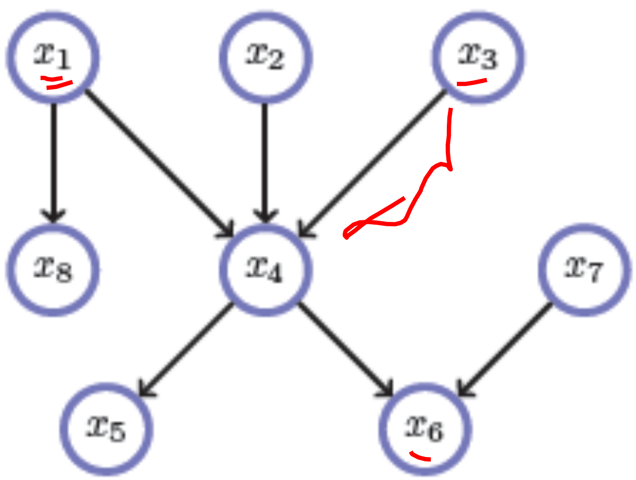
- Exactly what the name suggests
  - Directed edges
  - No (directed) cycles
  - Underlying undirected cycles okay

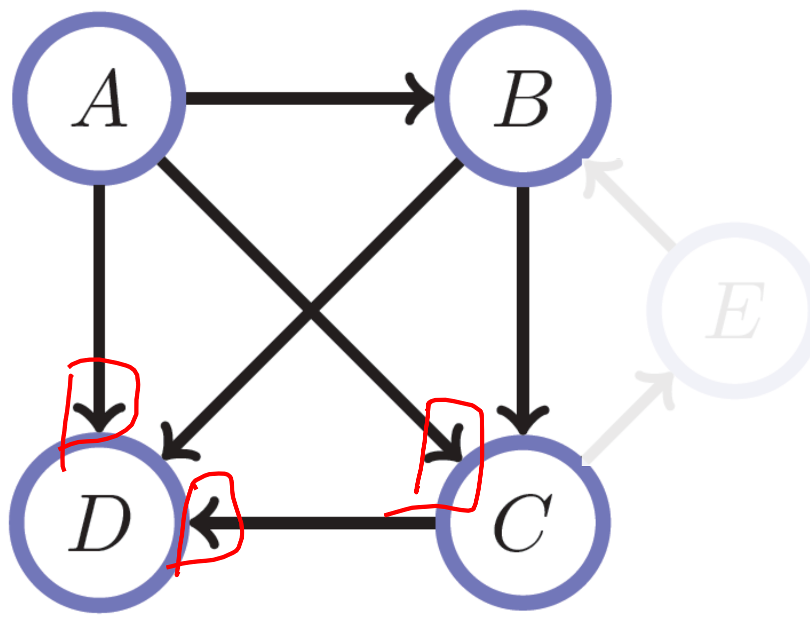
$G = (V, E)$

$E = \{ (v_i, v_j) \mid v_i, v_j \in V \}$

Vertex

Vertices





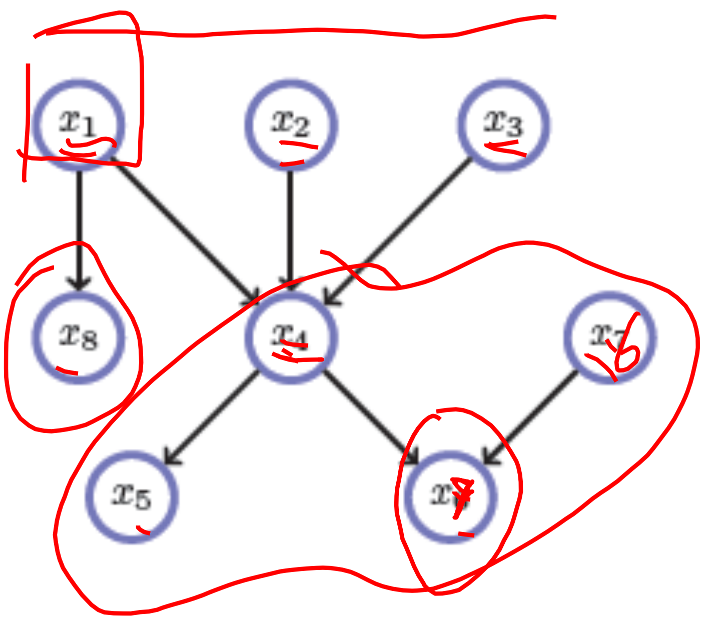
Directed Acyclic Graphs (DAGs)

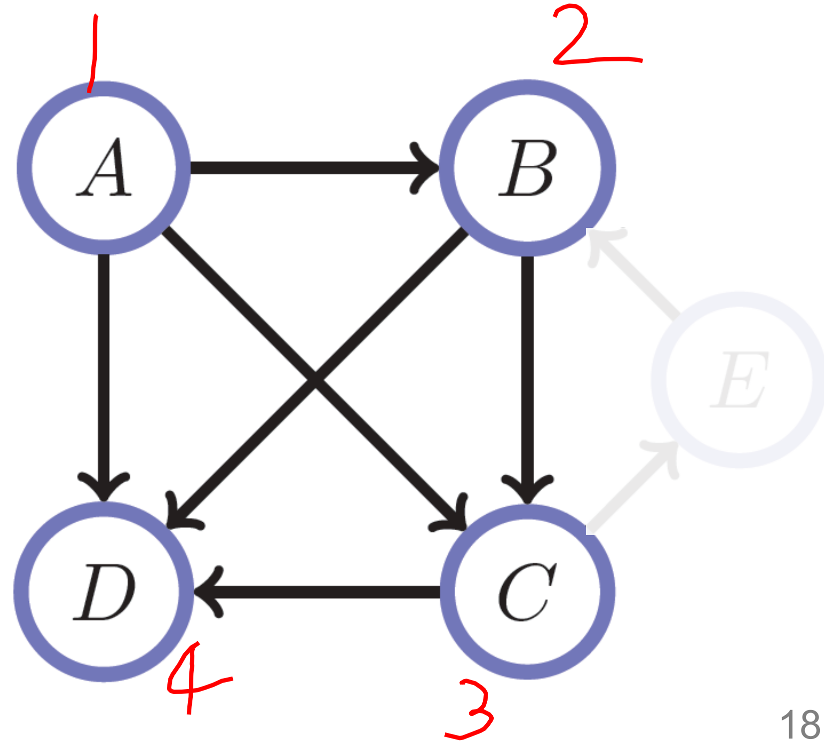
- Concept
  - Topological Ordering

$\exists$ bijection $\sigma : V \to \{1, \ldots, n\}$

s.t $\forall (v_i, v_j) \in E$

$\sigma(v_i) < \sigma(v_j)$



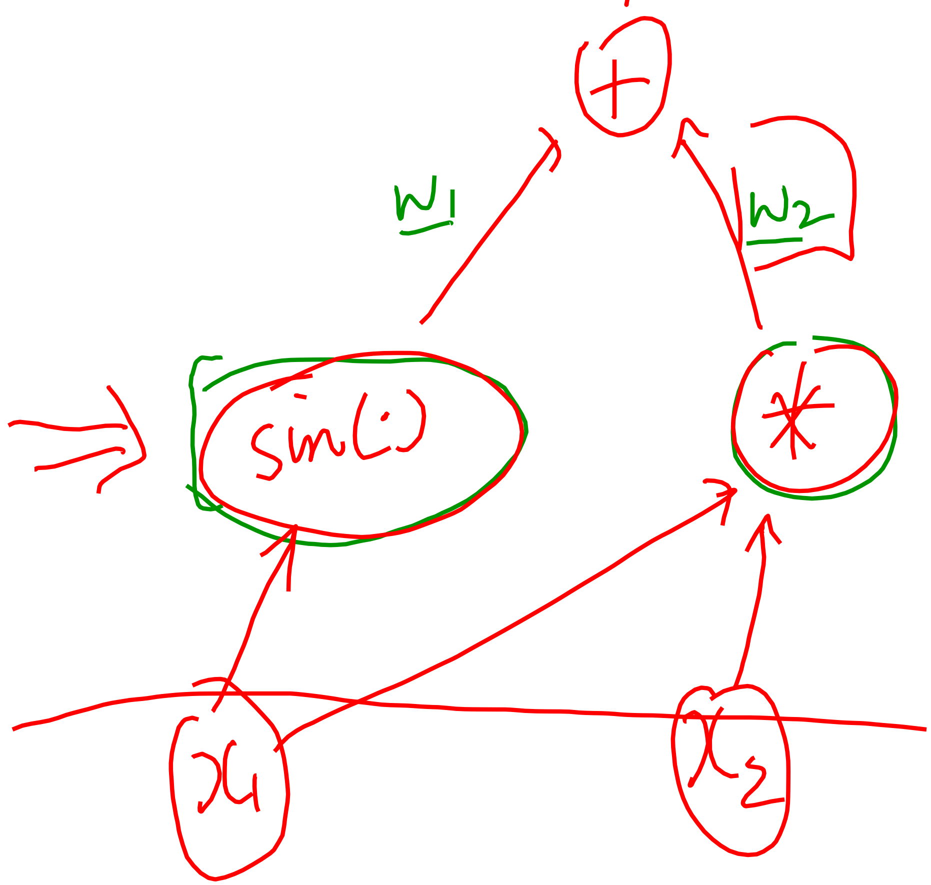


Computational Graphs

- Notation

$$f(x_1, x_2) = \overbrace{x_1 x_2}^{w_2} + \underbrace{\sin(x_1)}_{w_1}$$

w_3



Deep Learning = Differentiable Programming

- Computation = Graph
  - Input = Data + Parameters
  - Output = Loss
  - Scheduling = Topological ordering

- Auto-Diff
  - A family of algorithms for
    implementing chain-rule on computation graphs

Forward mode AD

Goal: $\boxed{\dfrac{\partial L}{\partial \vec{x}}}$

layer ℓ

$\vec{W}$

$\vec{x}$

$\boxed{L}$

$\dfrac{\partial L}{\partial \vec{x}}$

$\boxed{\vec{h}^{(\ell-1)}}$

$g(\ )$

$\vec{h}^{(\ell)} = g(\vec{h}^{\ell-1})$

Input: $\boxed{\dfrac{\partial \vec{h}^{\ell-1}}{\partial \vec{x}}}$

$$\boxed{\frac{\partial \vec{h}^{\ell}}{\partial \vec{x}}} = \boxed{\frac{\partial \vec{h}^{\ell}}{\partial \vec{h}^{\ell-1}}} \cdot \boxed{\frac{\partial \vec{h}^{\ell-1}}{\partial \vec{x}}}$$

Jacobian of g

Input

Reverse mode AD

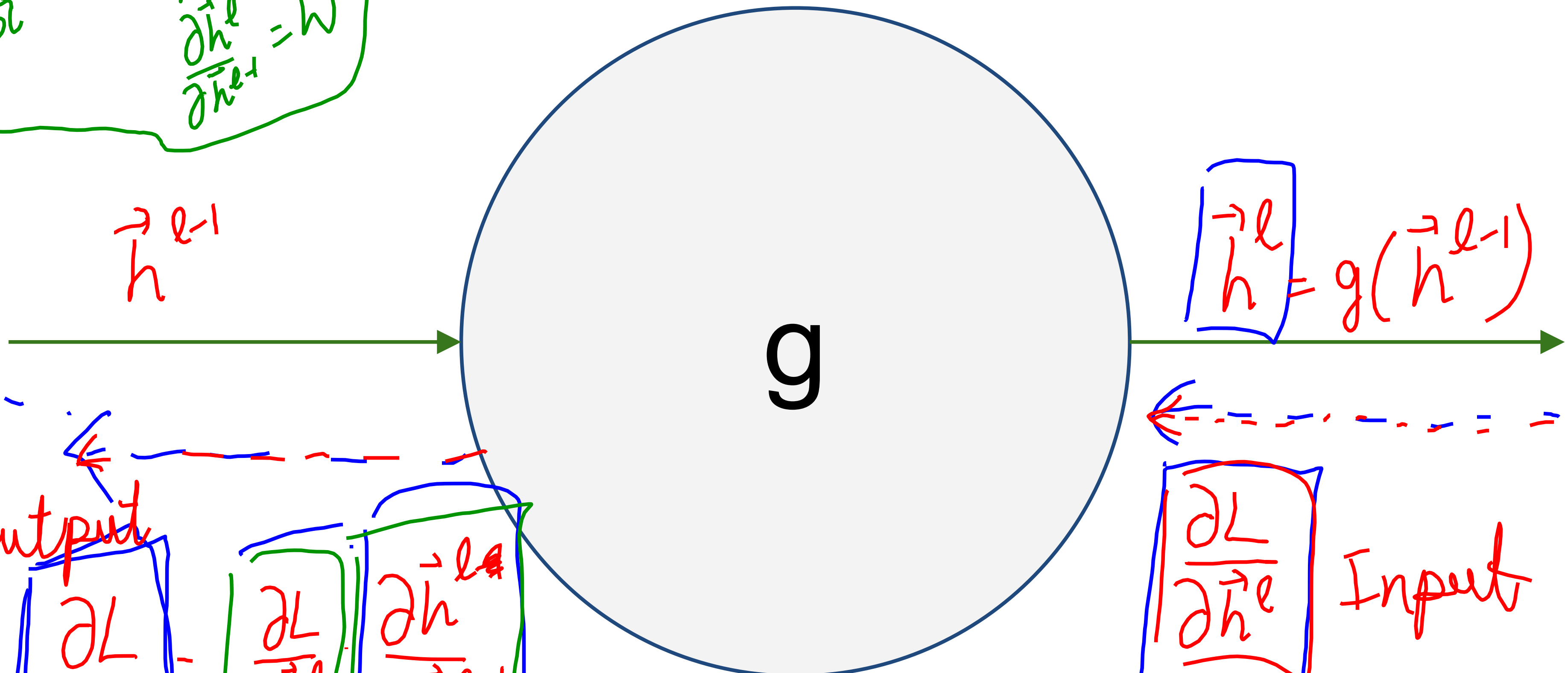
Goal: $\boxed{\dfrac{\partial L}{\partial \vec{x}}}$

$\vec{y} = A\vec{x}$

$\dfrac{\partial \vec{y}}{\partial \vec{x}} = A$

$\vec{h}^{\ell} = W\vec{h}^{\ell-1}$

$\dfrac{\partial \vec{h}^{\ell}}{\partial \vec{h}^{\ell-1}} = W$



$\vec{x}$

$\vec{h}^{\ell-1}$

$\boxed{\vec{h}^{\ell}} = g(\vec{h}^{\ell-1})$

$\boxed{\dfrac{\partial L}{\partial \vec{h}^{\ell}}}$ Input

Output

$\boxed{\dfrac{\partial L}{\partial \vec{h}^{\ell-1}}} = \boxed{\dfrac{\partial L}{\partial \vec{h}^{\ell}}} \boxed{\dfrac{\partial \vec{h}^{\ell}}{\partial \vec{h}^{\ell-1}}}$

$\dfrac{\partial L}{\partial \vec{x}}$

Input

Jacobian of g

Plan for Today

- Automatic Differentiation
  - (Finish) Forward mode vs Reverse mode AD
  - Backprop
  - Patterns in backprop
  - Jacobians in FC+ReLU NNs

Example: Forward mode AD

$$f(x_1, x_2) = \overbrace{x_1 x_2}^{w_2} + \overbrace{\sin(x_1)}^{w_1}$$

$$\frac{\partial f}{\partial \vec{x}} = \left[\frac{\partial f}{\partial x_1} \quad \frac{\partial f}{\partial x_2}\right]$$

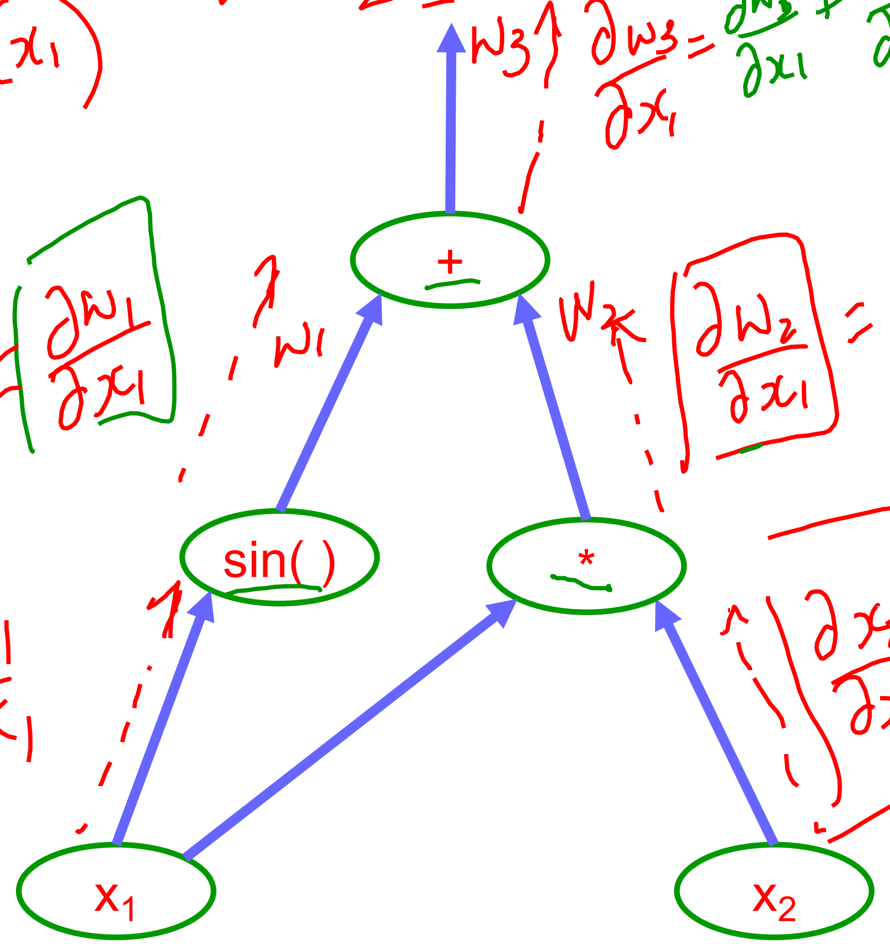
$w_1 + w_2 = w_3$ $\qquad \frac{\partial w_3}{\partial x_1} = \frac{\partial w_1}{\partial x_1} + \frac{\partial w_2}{\partial x_1}$

$w_1 = \sin(x_1)$ $\qquad w_2 = x_1 x_2$

$\boxed{\frac{\partial f}{\partial x_1} = \cos x_1 + x_2}$

$\cos(x_1) = \frac{\partial w_1}{\partial x_1}$ $\qquad \frac{\partial w_2}{\partial x_1} = x_2$

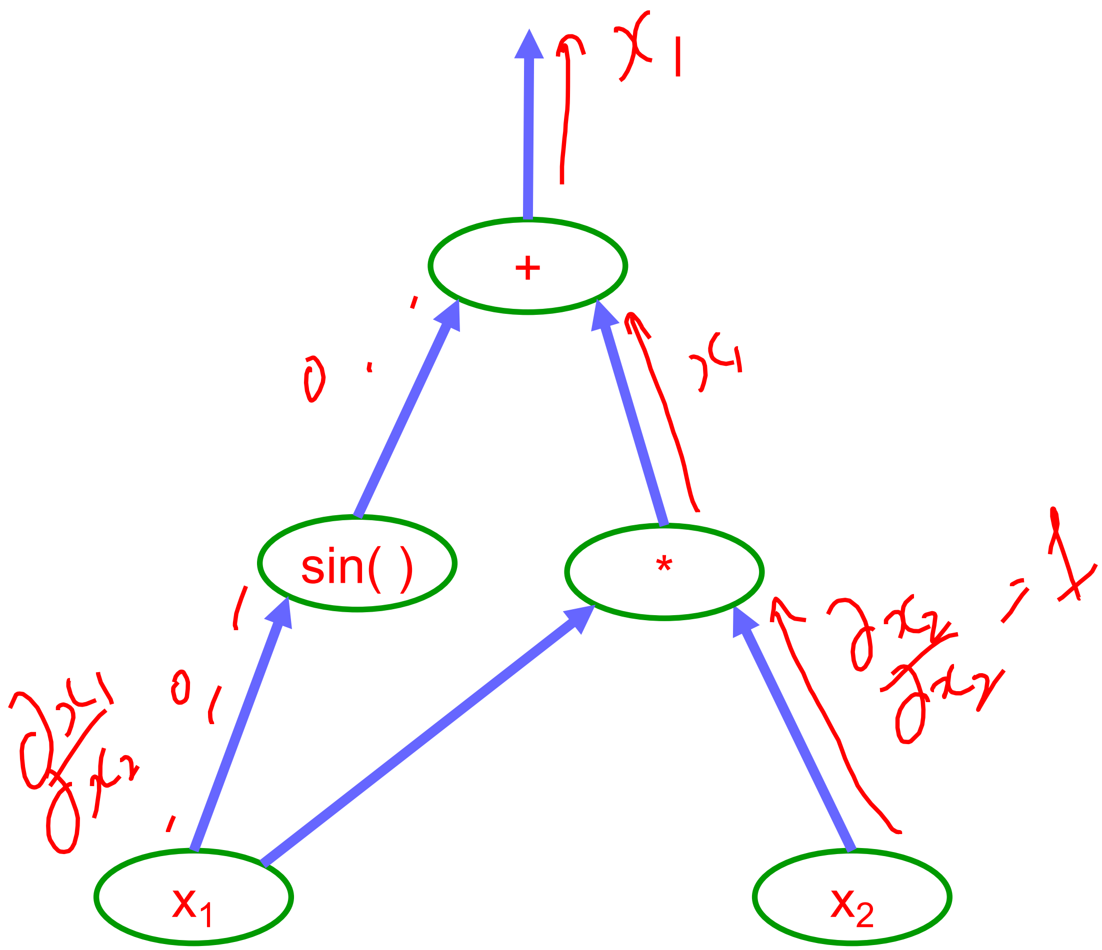
$1 = \frac{\partial x_1}{\partial x_1}$ $\qquad \frac{\partial x_2}{\partial x_1} = 0$



Example: Forward mode AD

$$f(x_1, x_2) = \boxed{x_1 x_2 + \cancel{\sin(x_1)}}$$

$$\boxed{\frac{\partial f}{\partial x_2}} = x_1$$



Example: Forward mode AD

$$f(x_1, x_2) = \overbrace{x_1 x_2}^{w_2} + \overbrace{\sin(x_1)}^{w_1}$$

Goal: $\frac{\partial f}{\partial \vec{x}} = \left[\frac{\partial f}{\partial x_1} \quad \frac{\partial f}{\partial x_2}\right]$

$\frac{\partial x_1}{\partial a} = \dot{x}_1$

$\frac{\partial x_2}{\partial a}$

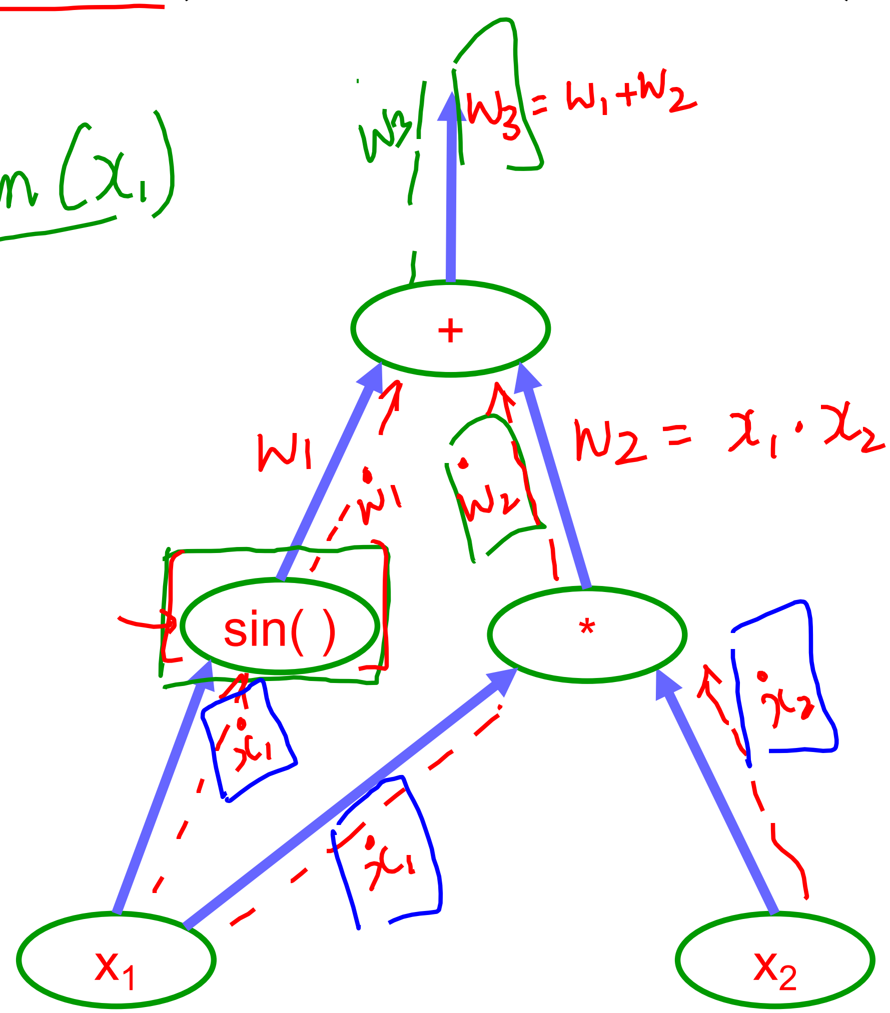
$w_1 = \sin(x_1)$

$\boxed{\frac{\partial w_1}{\partial a}} = \underbrace{\boxed{\frac{\partial w_1}{\partial x_1}}}_{} \boxed{\frac{\partial x_1}{\partial a}}$ — J, Input

$\cos(x_1) \cdot \dot{x}_1$

$\frac{\partial w_2}{\partial a}$

$\boxed{\frac{\partial w_3}{\partial a}}$



$$\frac{\partial w_2}{\partial a} = x_1 \frac{\partial x_2}{\partial a} + x_2 \frac{\partial x_1}{\partial a}$$

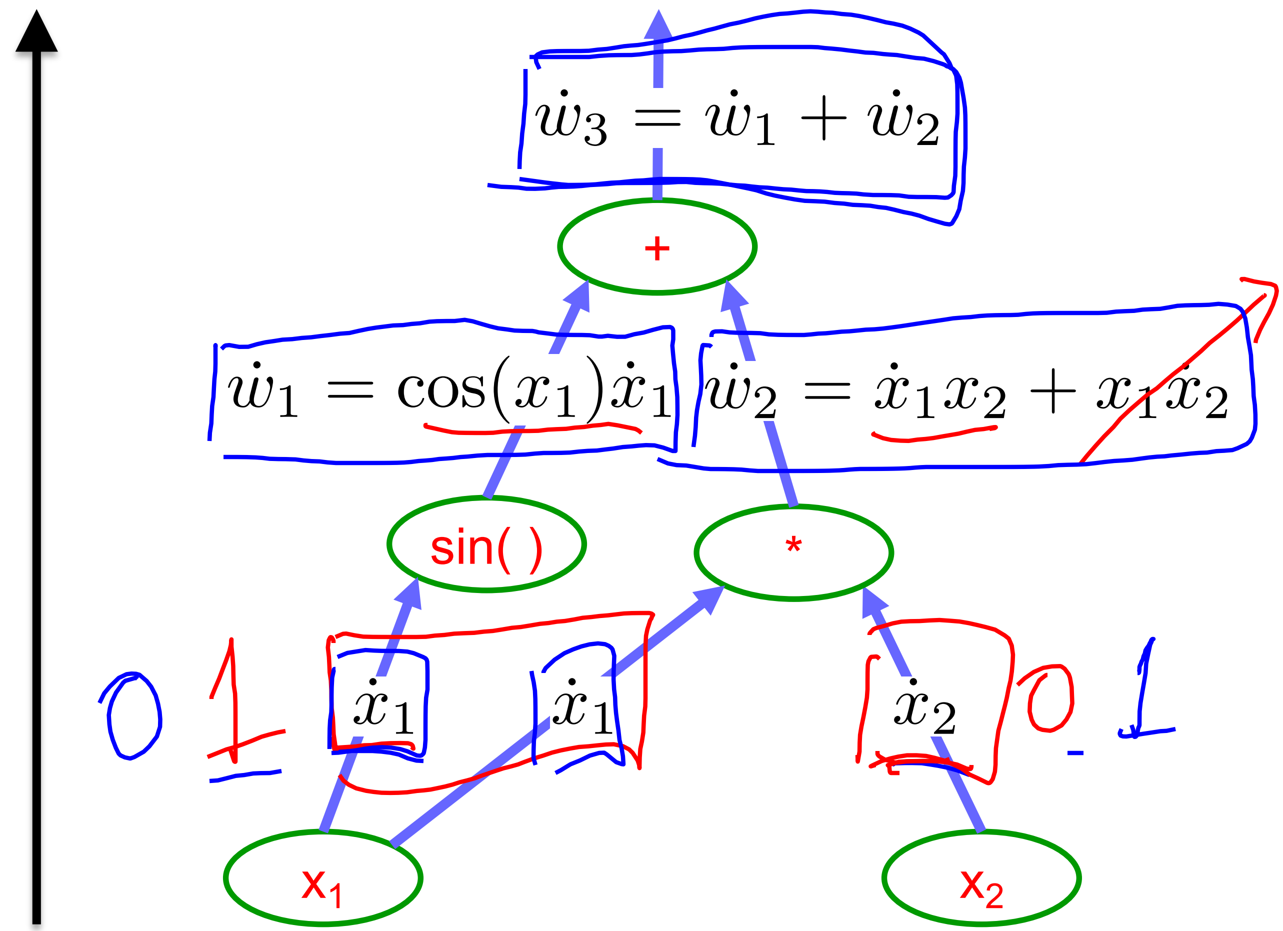
$$= x_1 \dot{x}_2 + x_2 \dot{x}_1$$

$\frac{\partial f}{\partial a}$

$$\boxed{a \in \{x_1, x_2\}}$$

Example: Forward mode AD

$$f(x_1, x_2) = x_1 x_2 + \sin(x_1) \qquad \frac{\partial f}{\partial \vec{x}} = \left[\frac{\partial f}{\partial x_1} \;\; \frac{\partial f}{\partial x_2} \right]$$



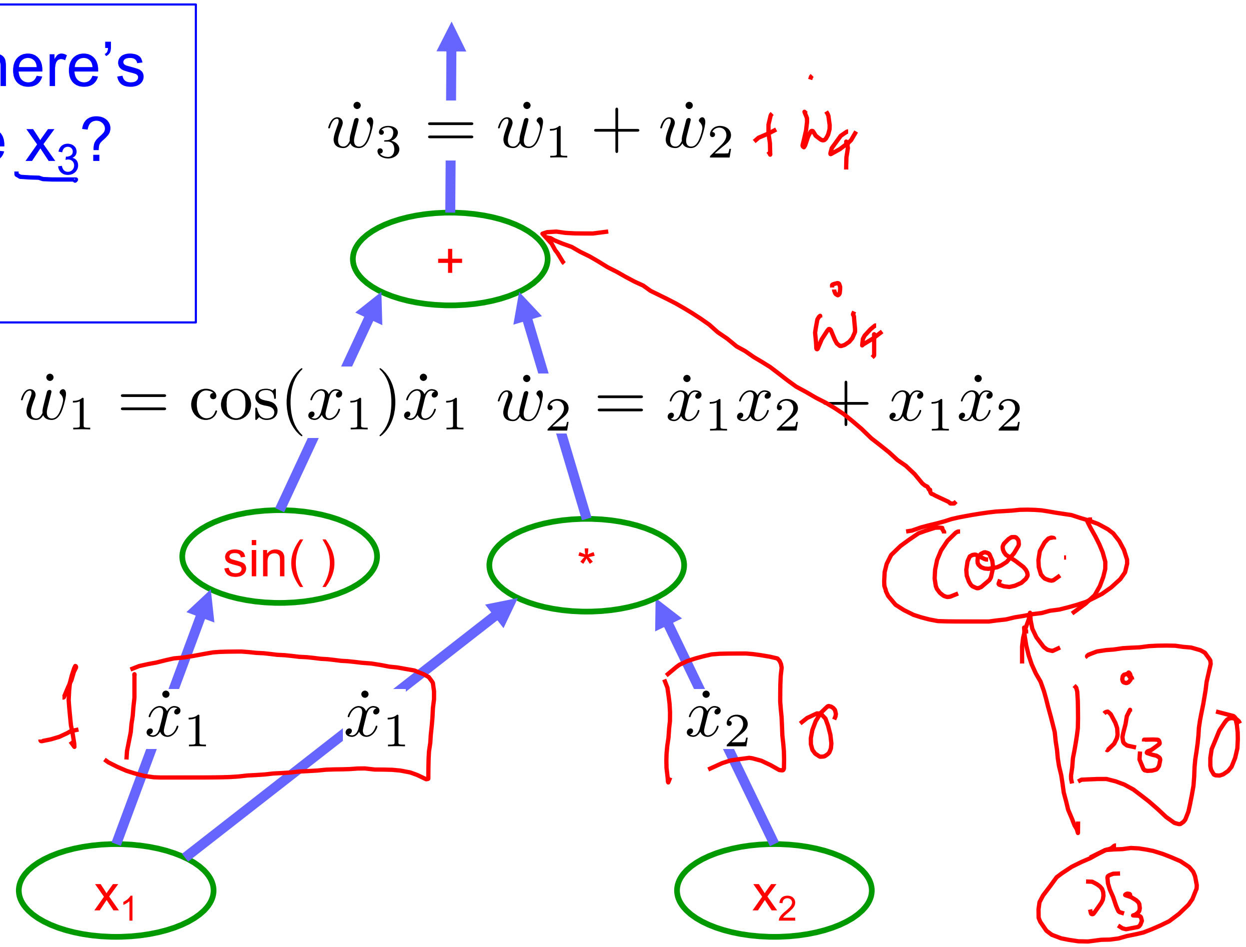
$\dot{x}_1 = \frac{\partial x_1}{\partial a}$

$\dot{w}_1 = \frac{\partial w_1}{\partial a}$

Example: Forward mode AD

$$f(x_1, x_2) = \underline{x_1} x_2 + \sin(\underline{x_1}) + \cos(x_3)$$

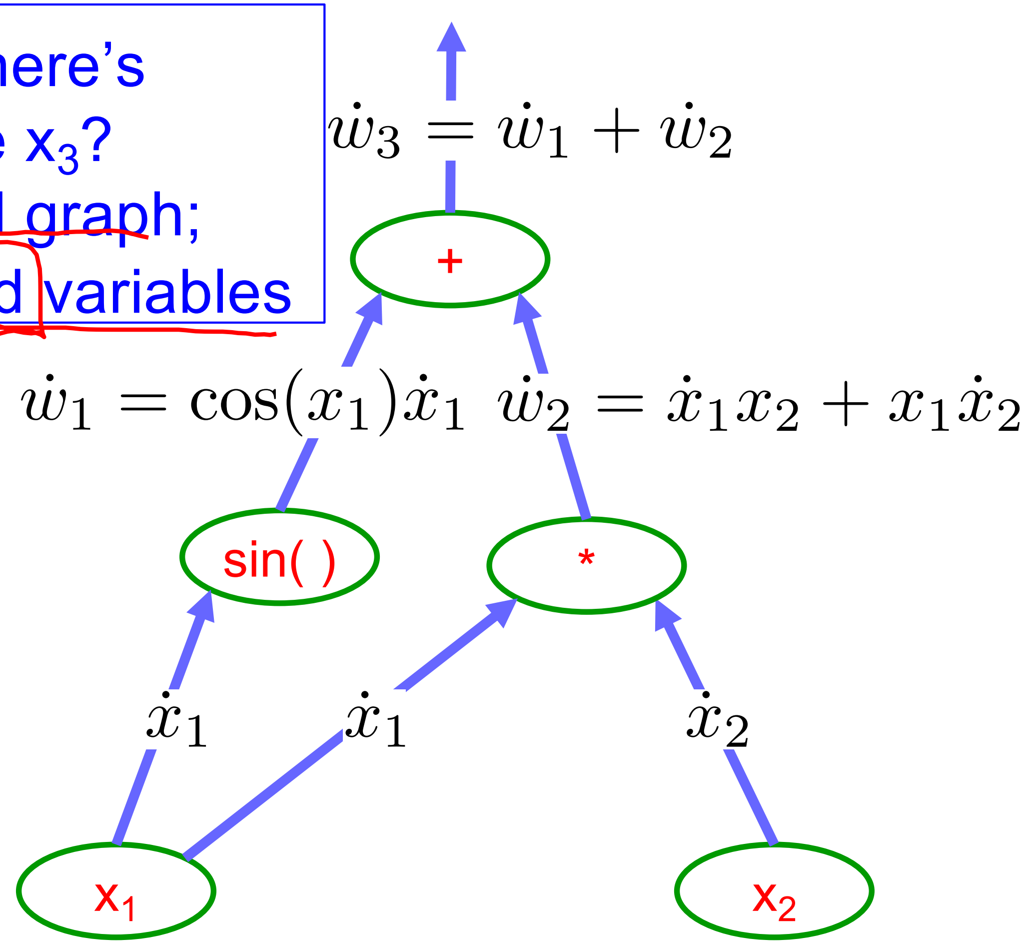
Q: What happens if there's another input variable x_3?



Example: Forward mode AD

$$f(x_1, x_2) = x_1 x_2 + \sin(x_1)$$

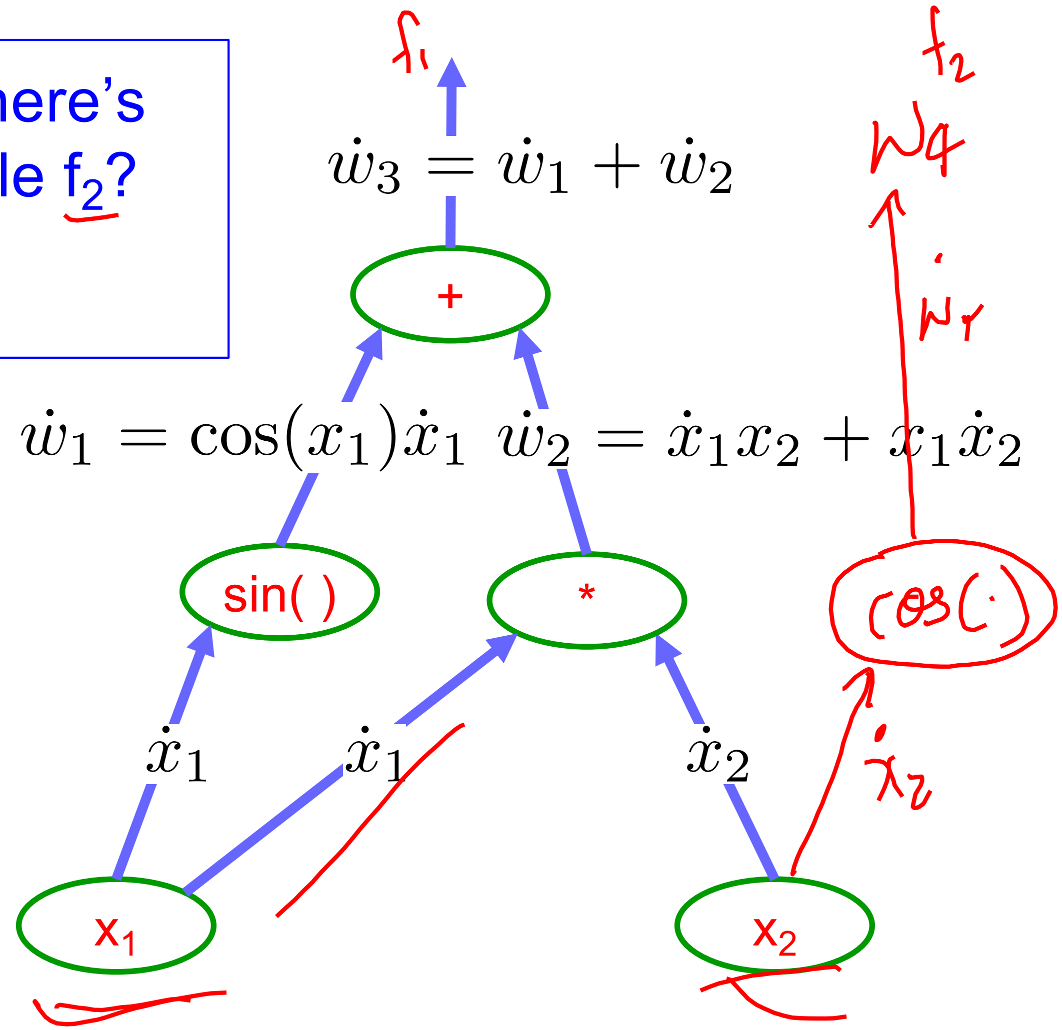
Q: What happens if there's another input variable x_3?
A: more sophisticated graph;
d "forward props" for d variables



Example: Forward mode AD

$$f_1(x_1, x_2) = x_1 x_2 + \sin(x_1) \quad f_2 = \cos(x_2)$$

Q: What happens if there's another output variable f_2?

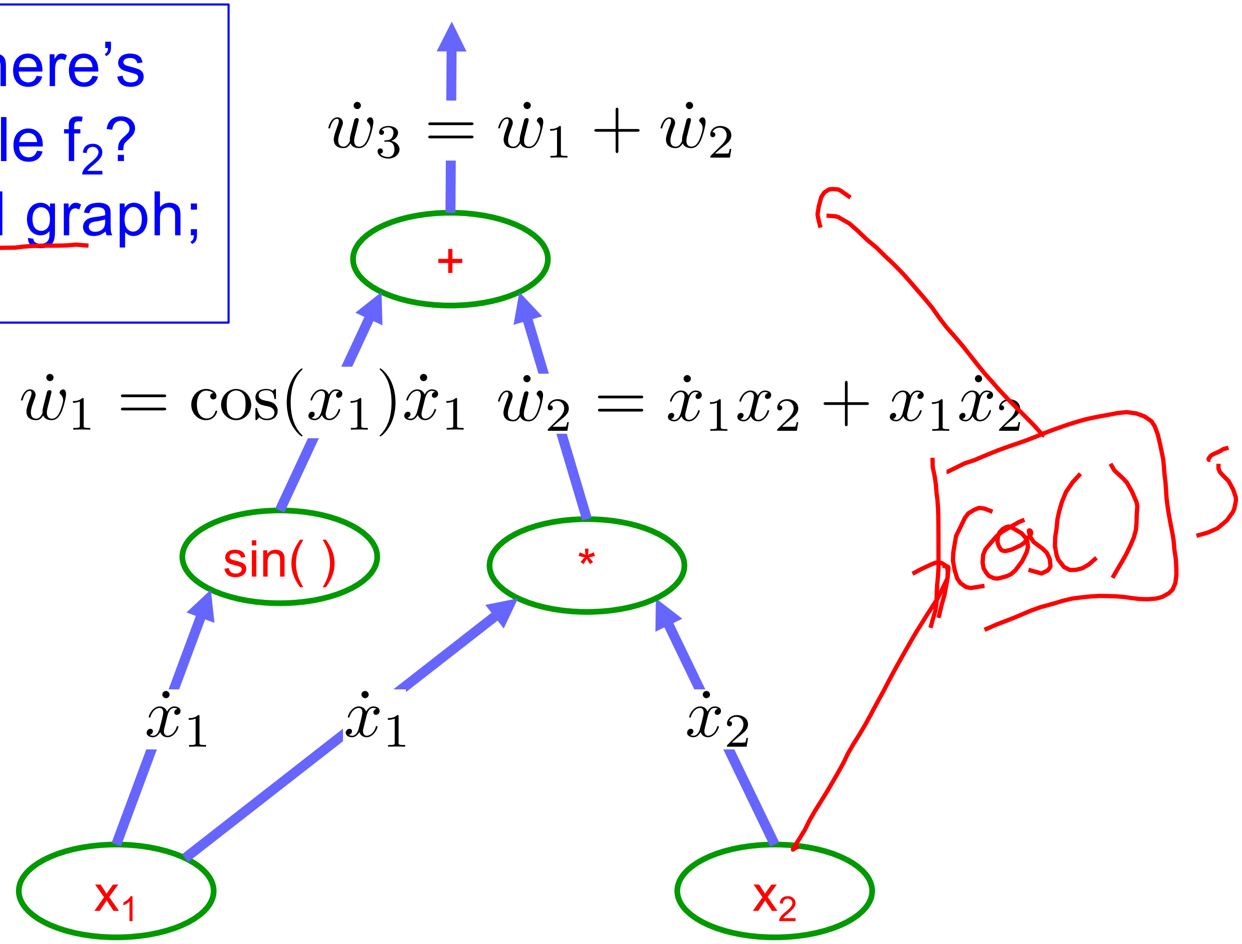


Example: Forward mode AD

$$f(x_1, x_2) = x_1 x_2 + \sin(x_1)$$

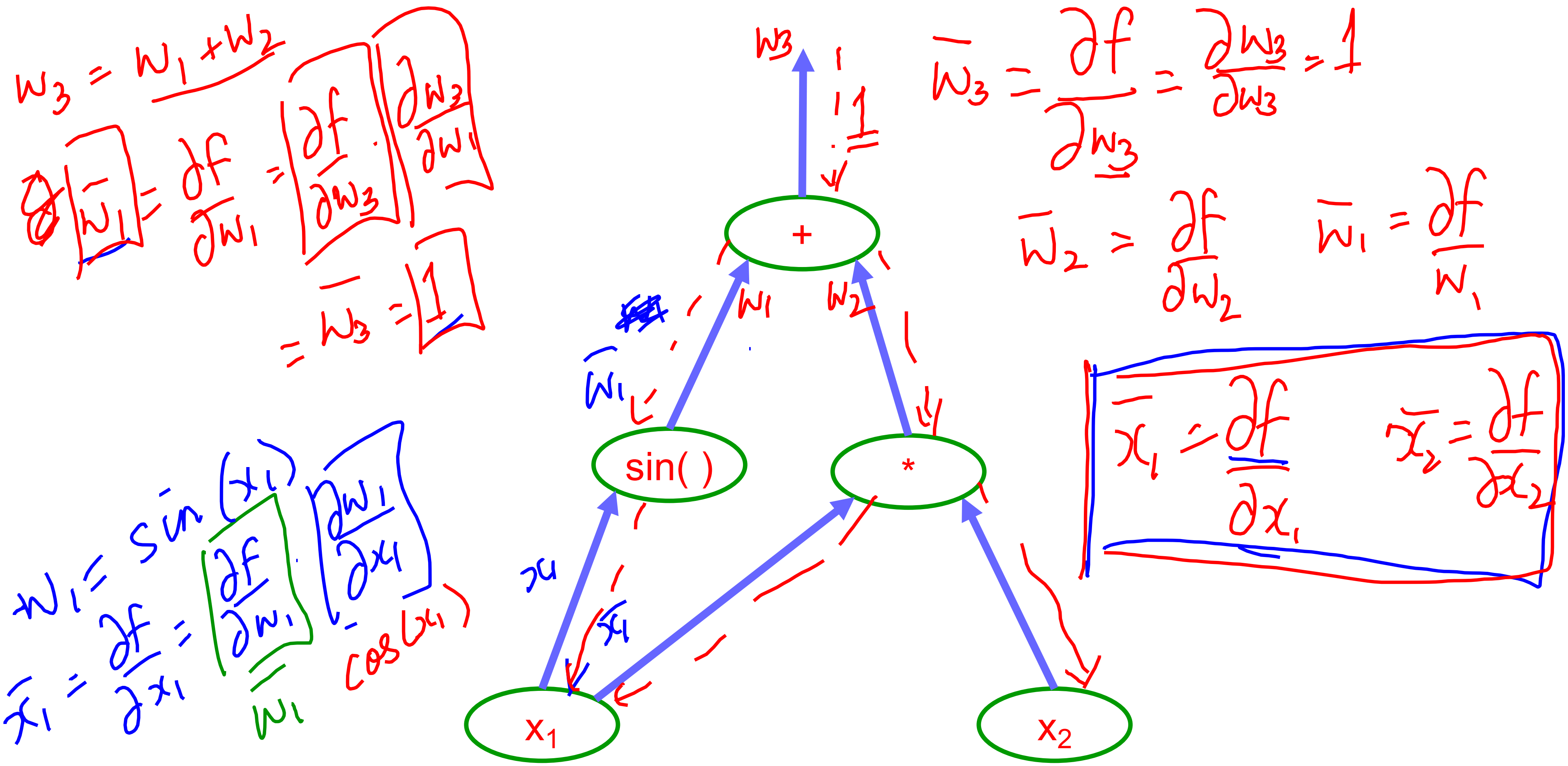
Q: What happens if there's another output variable f_2?
A: more sophisticated graph; ~~single~~ 2 "forward prop"

for $a \in \{x_1, x_2\}$



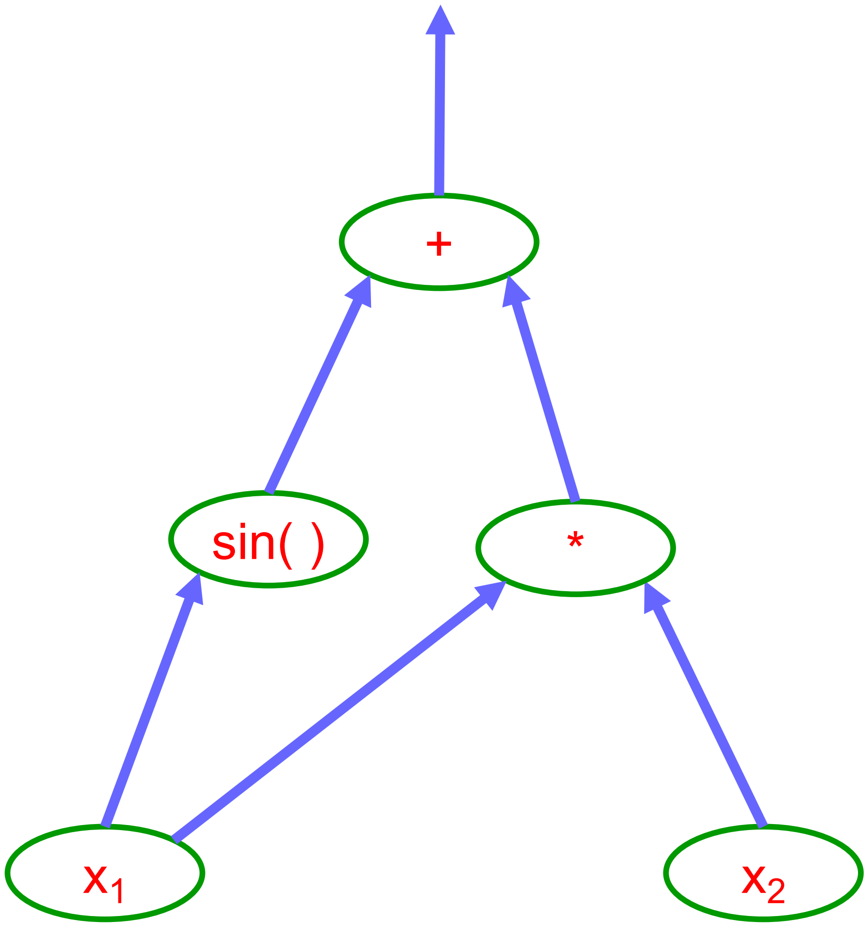
Example: Reverse mode AD

$$f(x_1, x_2) = x_1 x_2 + \sin(x_1)$$



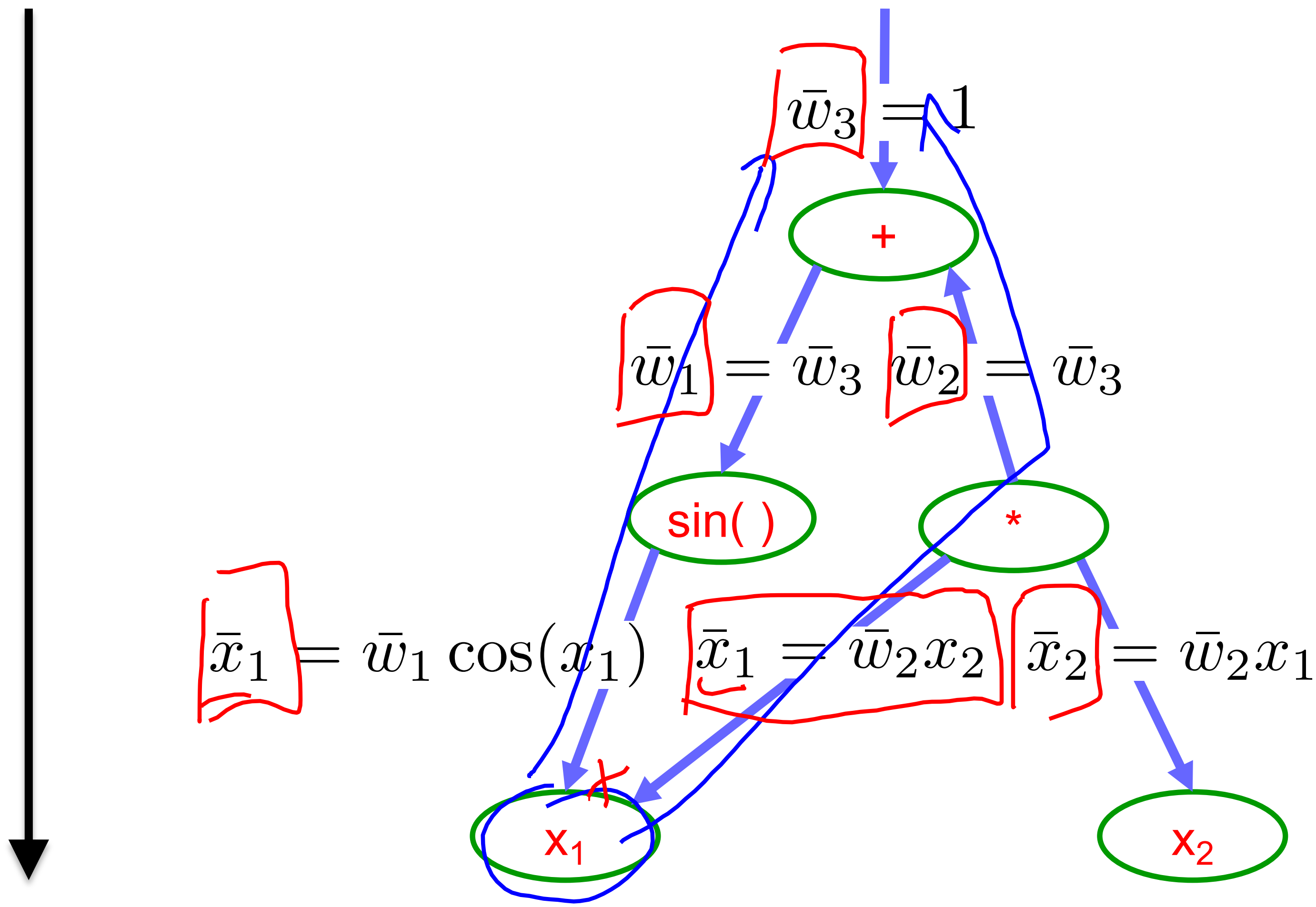
Example: Reverse mode AD

$$f(x_1, x_2) = x_1 x_2 + \sin(x_1)$$

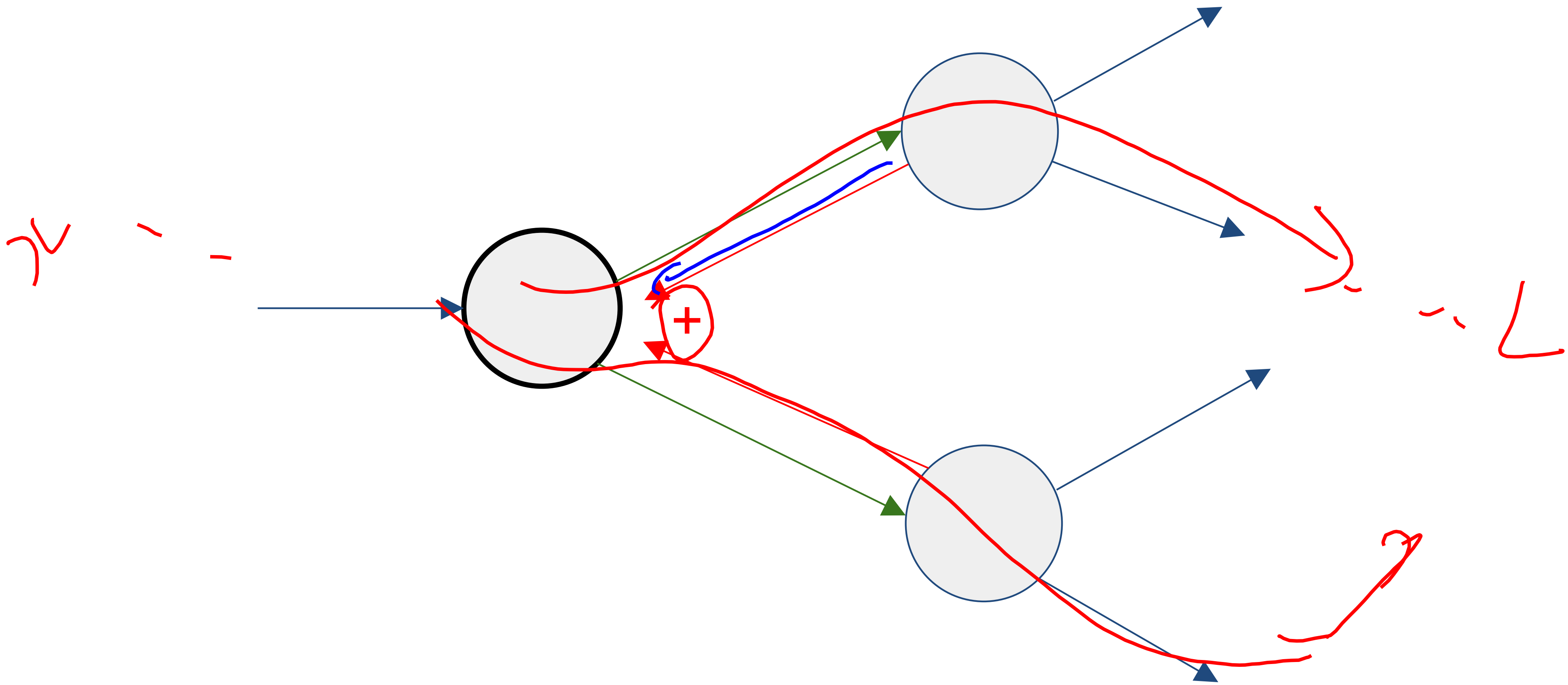


Example: Reverse mode AD

$$f(x_1, x_2) = x_1 x_2 + \sin(x_1)$$



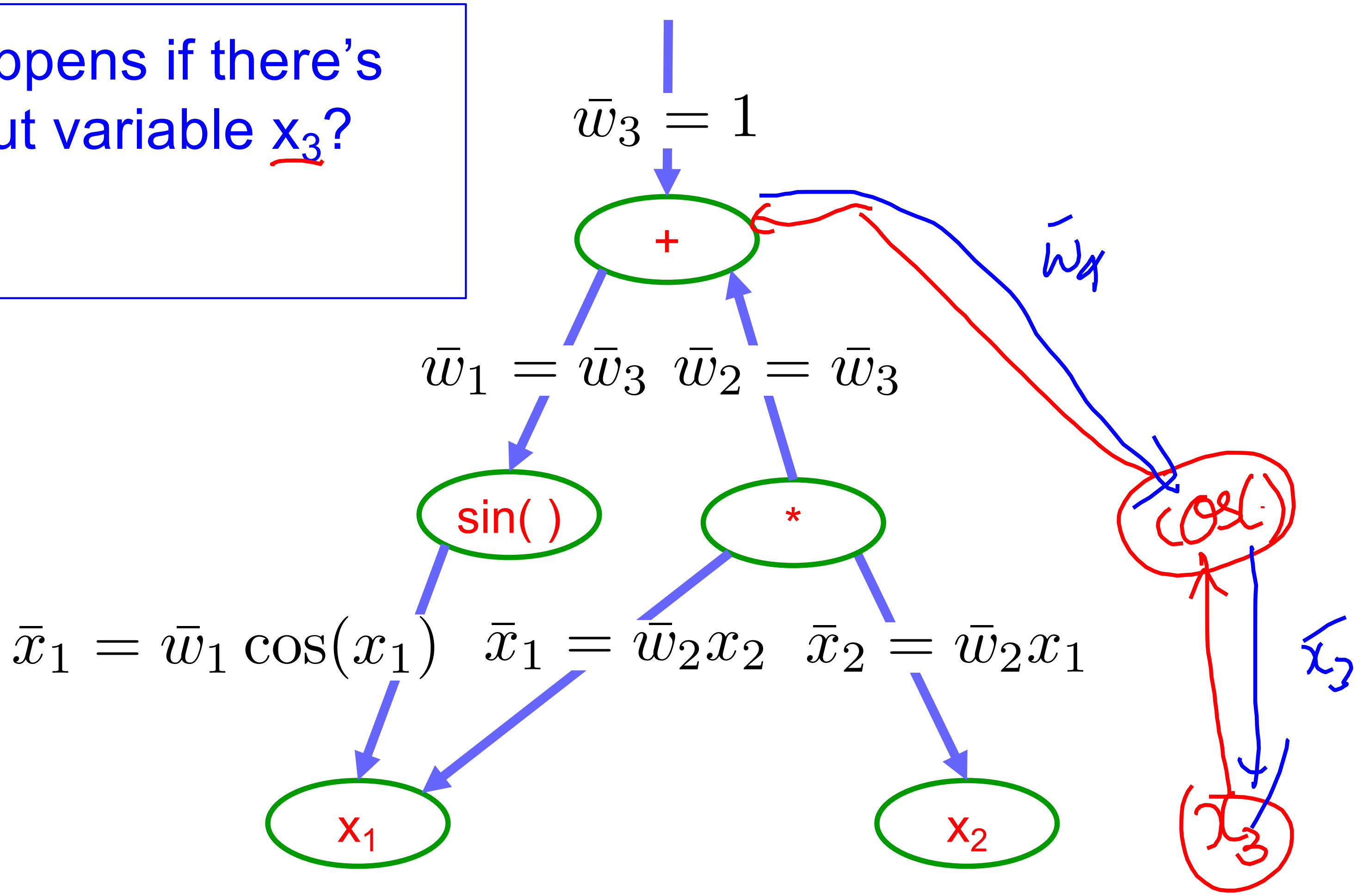
Gradients add at branches



Example: Reverse mode AD

$$f(x_1, x_2) = x_1 x_2 + \sin(x_1) + \cos(x_3)$$

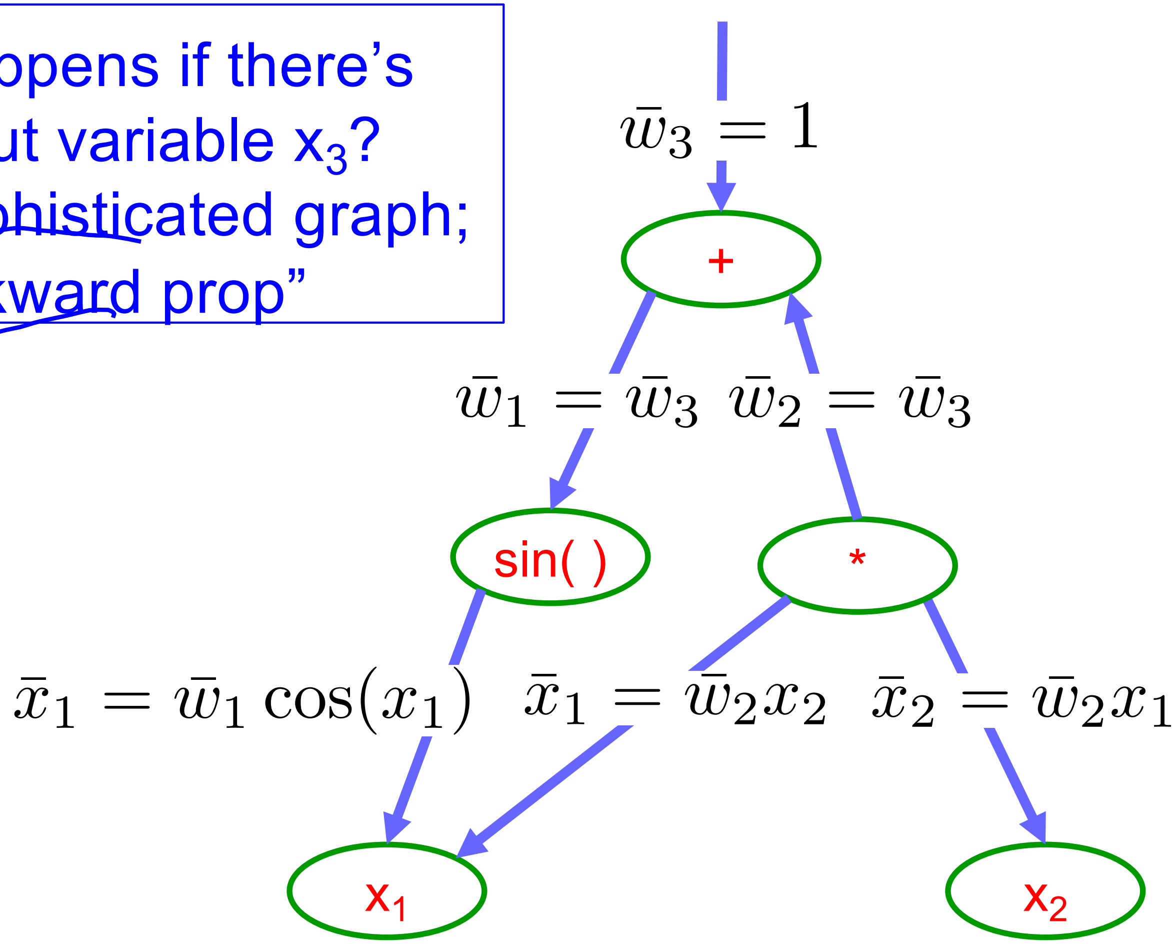
Q: What happens if there's another input variable x_3?



Example: Reverse mode AD

$$f(x_1, x_2) = x_1 x_2 + \sin(x_1)$$

Q: What happens if there's another input variable x_3?
A: more sophisticated graph; single "backward prop"

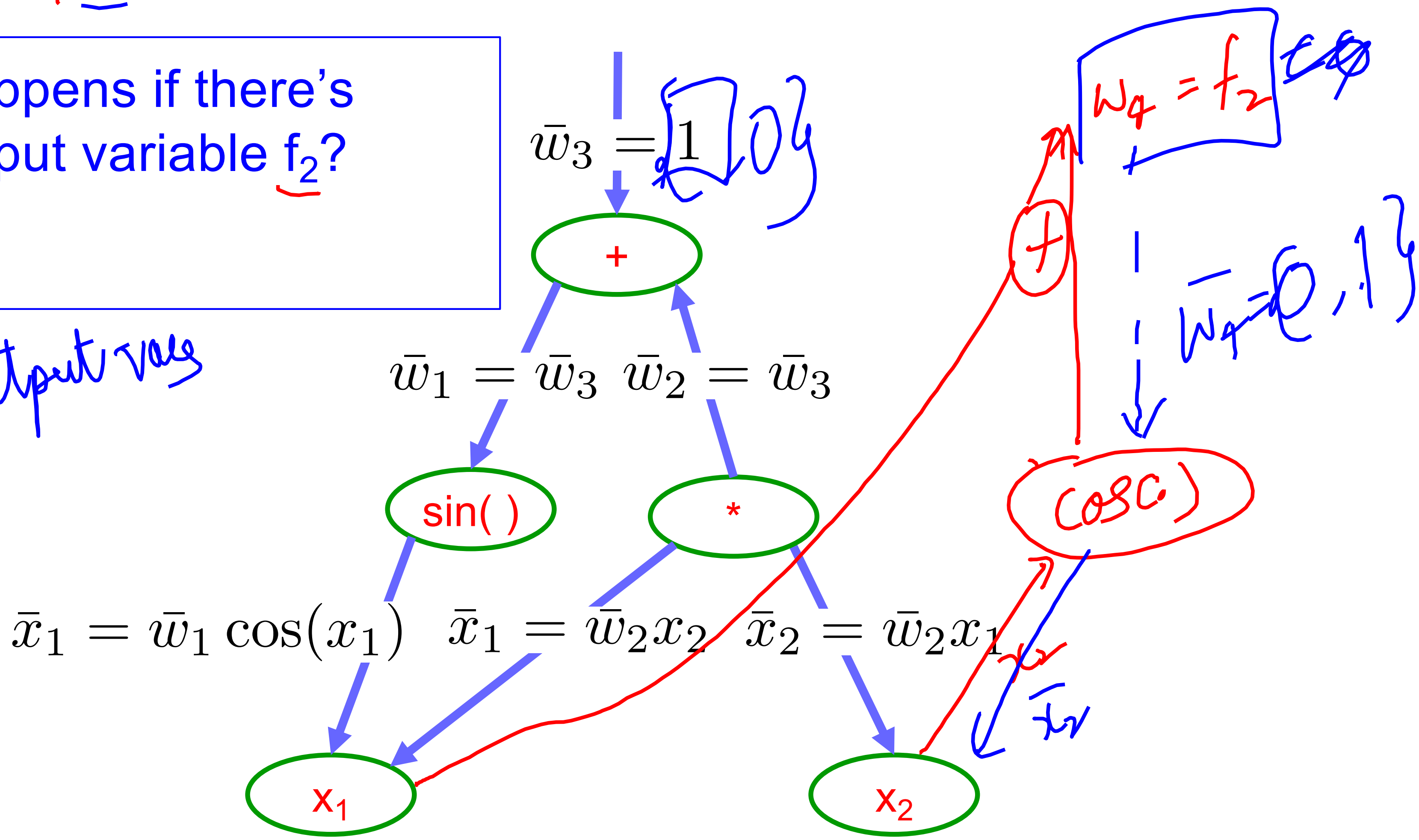


Example: Reverse mode AD

$$\boxed{f_1}(x_1, x_2) = x_1 x_2 + \sin(x_1) \quad \boxed{f_2} = \cos(x_2) + x_1$$

Q: What happens if there's another output variable f_2?

For an output var

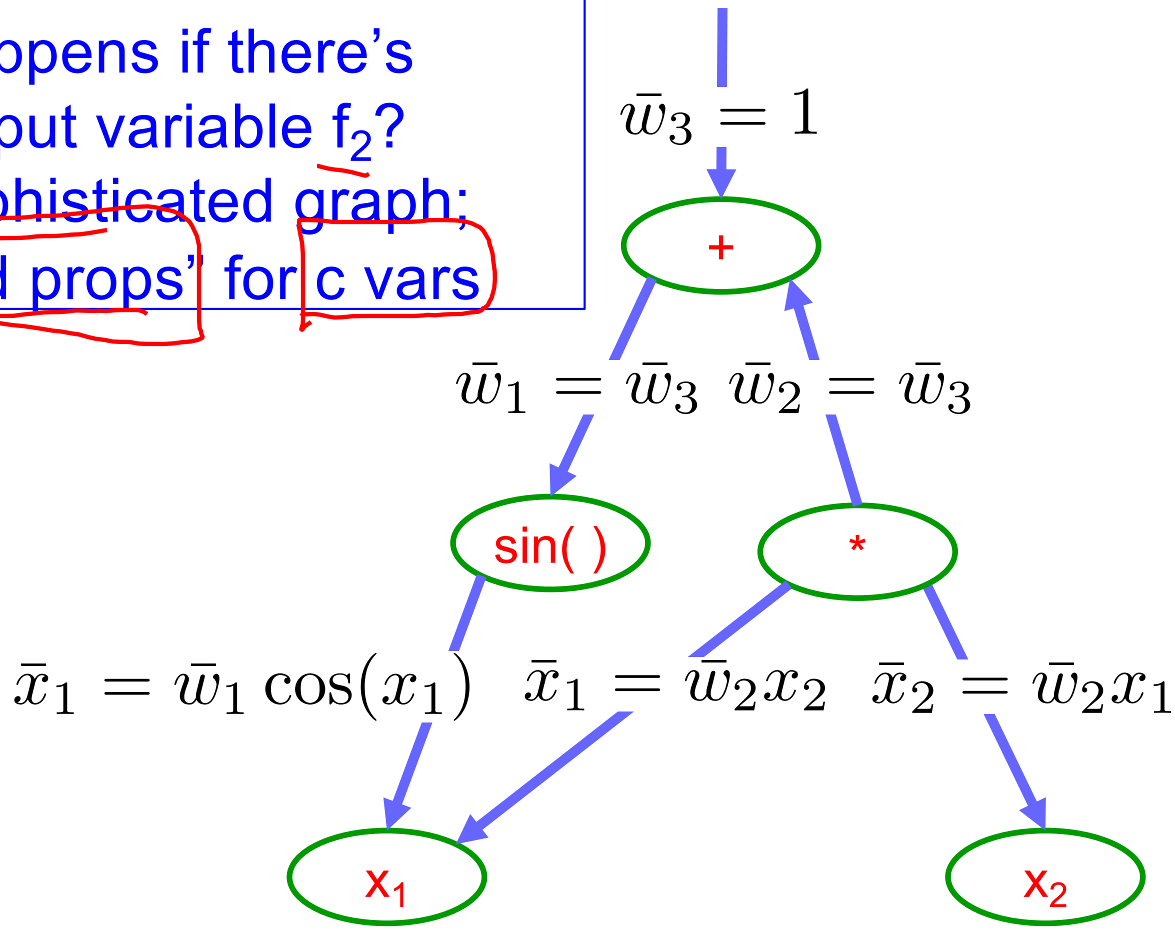


Example: Reverse mode AD

$$f(x_1, x_2) = x_1 x_2 + \sin(x_1)$$

f_1 . . . f_c

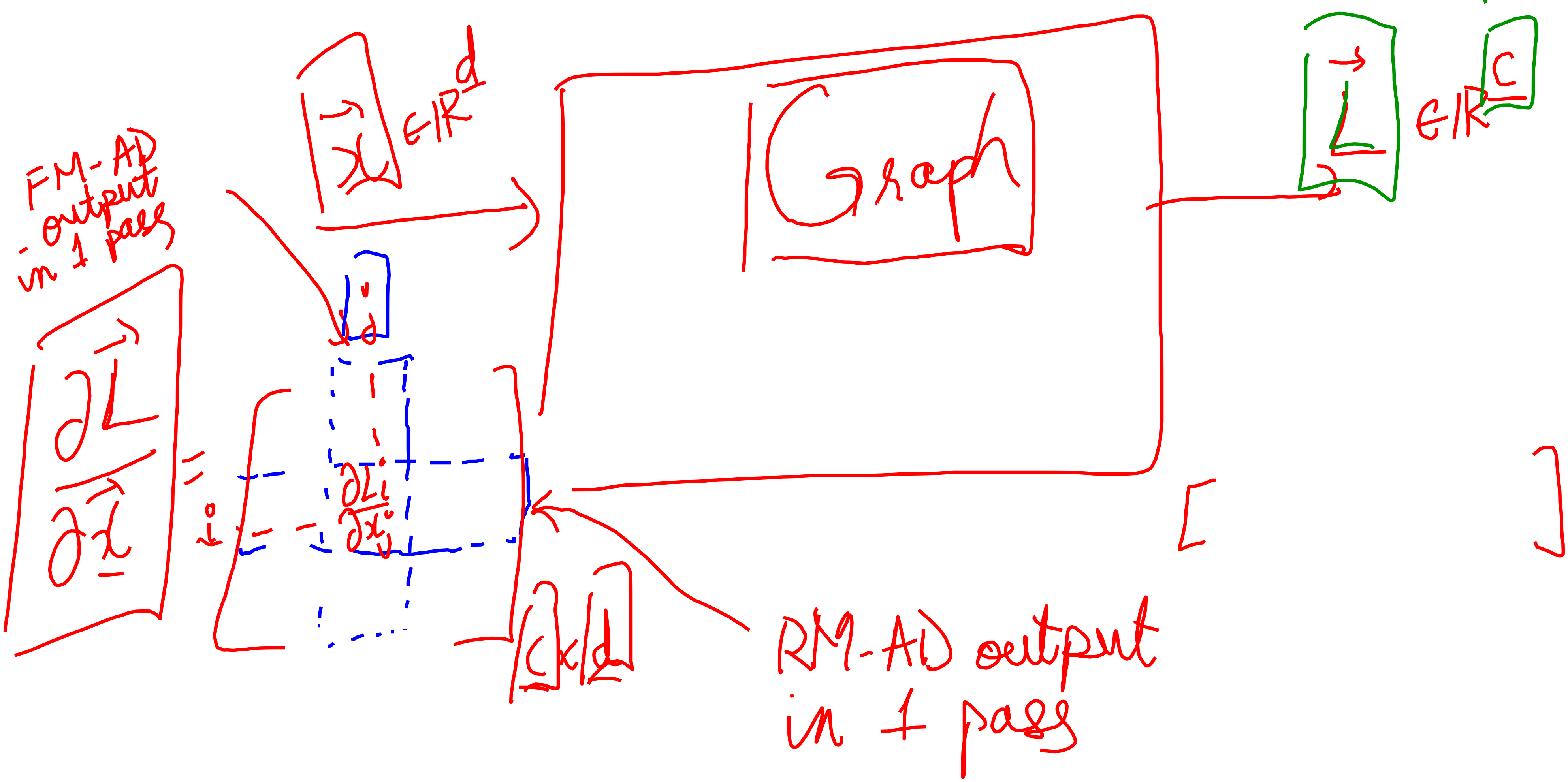
Q: What happens if there's another output variable f_2?
A: more sophisticated graph; c "backward props" for c vars



Forward mode vs Reverse Mode

- x → Graph → L
- Intuition of Jacobian

losses



Forward mode vs Reverse Mode

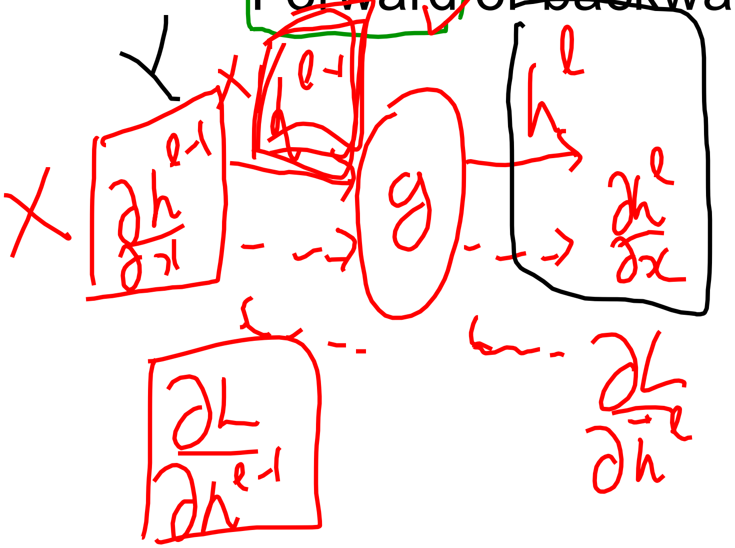
- What are the differences?

- Which one is faster to compute?
  - Forward or backward?

Is $c > d$ or $c \ll d$

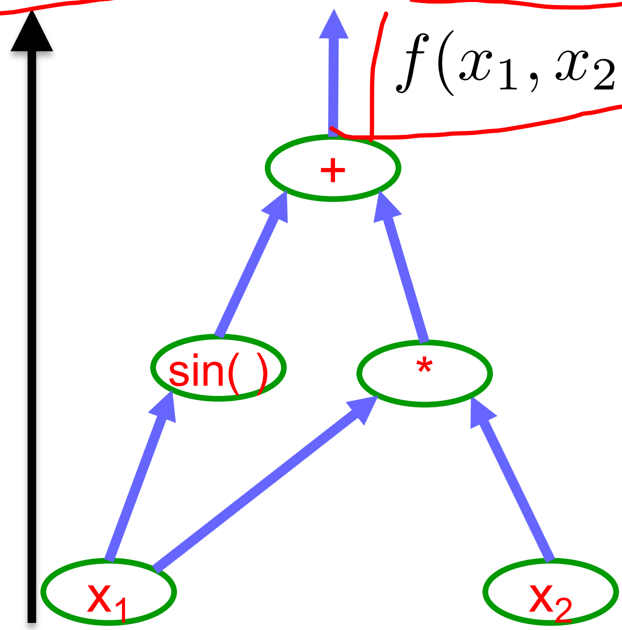
Forward mode vs Reverse Mode

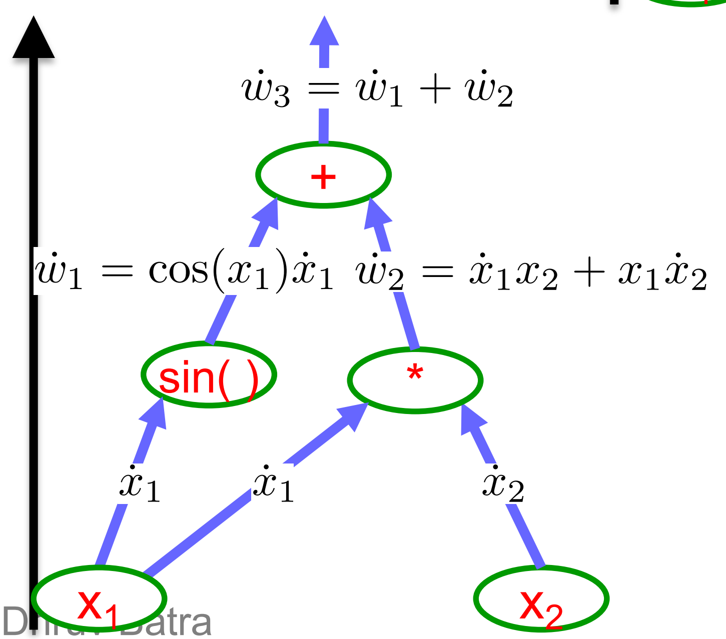
- What are the differences?
- Which one is faster to compute?
  - Forward or backward?
- Which one is more memory efficient (less storage)?
  - Forward or backward?

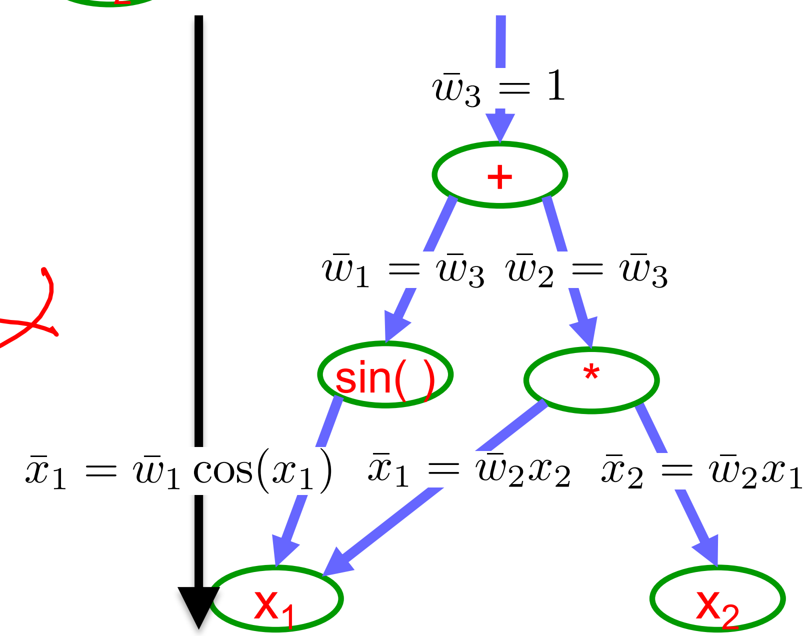


Forward Pass vs Forward mode AD vs Reverse Mode AD

$$f(x_1, x_2) = x_1 x_2 + \sin(x_1)$$



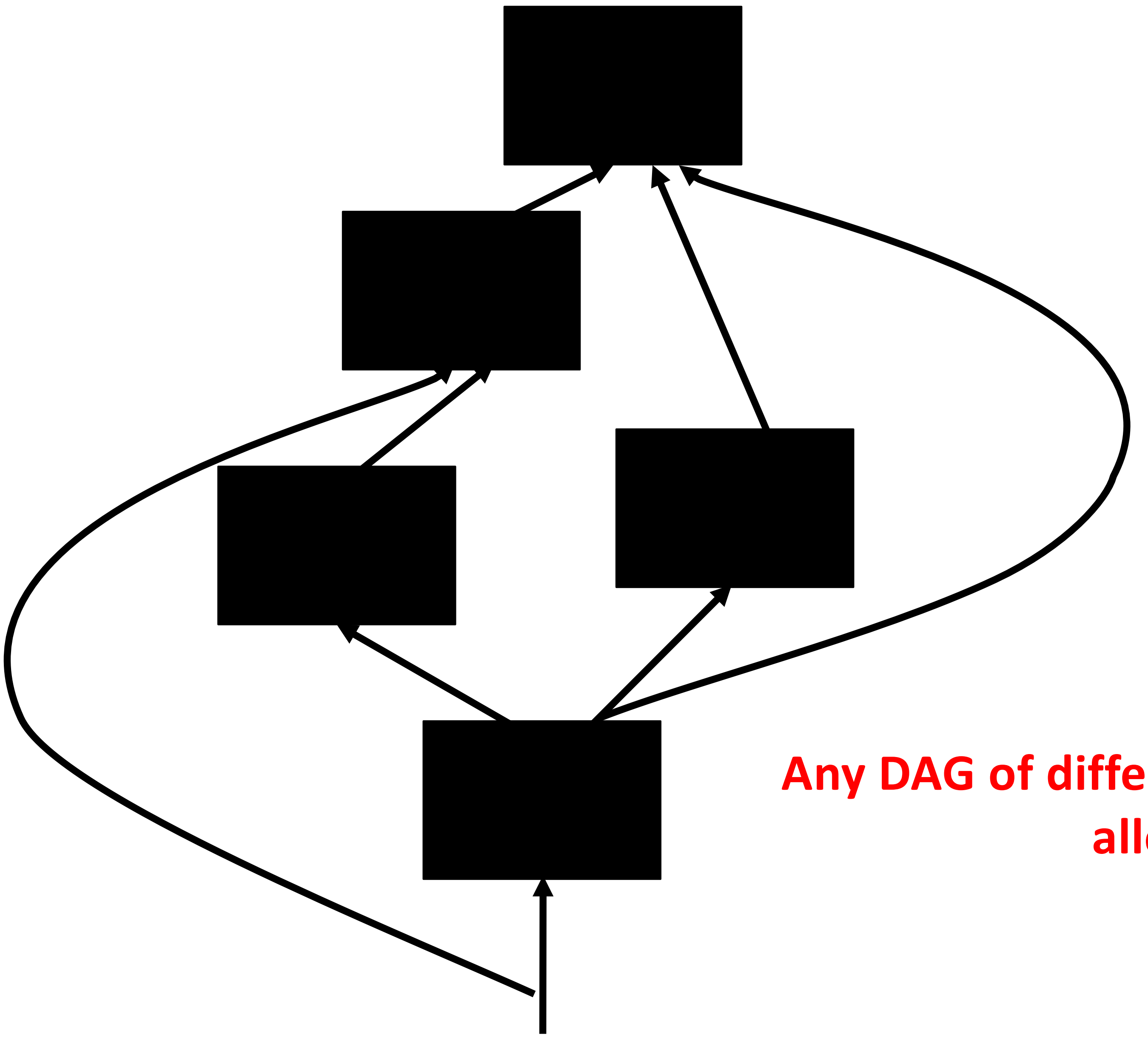




Plan for Today

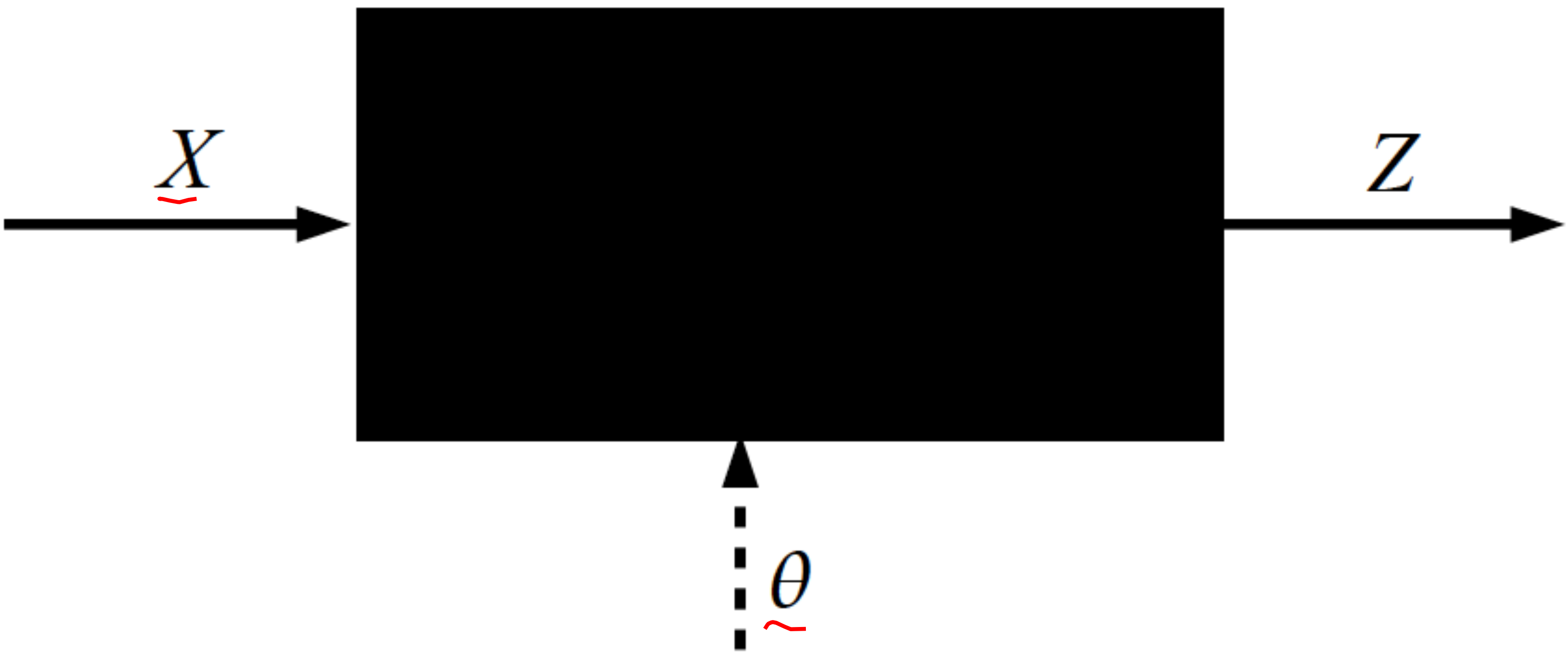
- Automatic Differentiation
  - (Finish) Forward mode vs Reverse mode AD
  - Backprop
  - Patterns in backprop
  - Jacobians in FC+ReLU NNs

Computational Graph

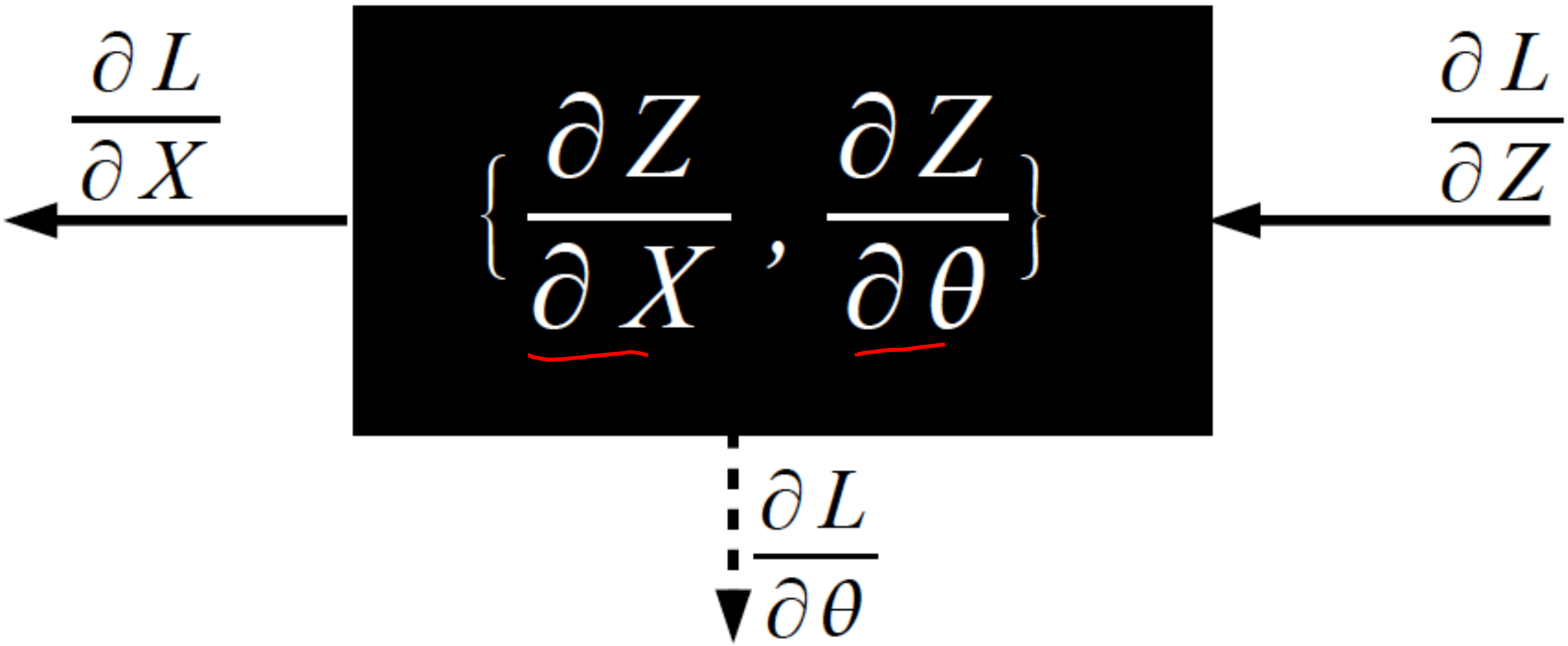


Any DAG of differentiable modules is allowed!

Key Computation: Forward-Prop

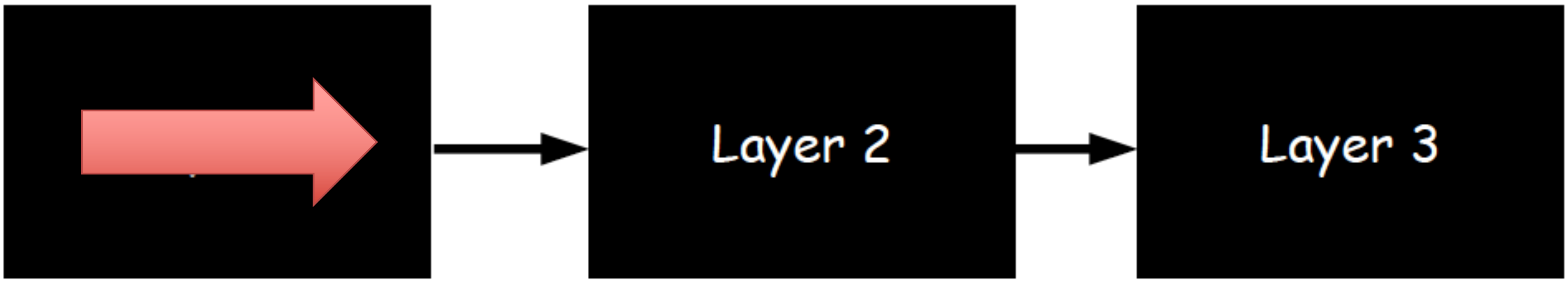


Key Computation: Back-Prop



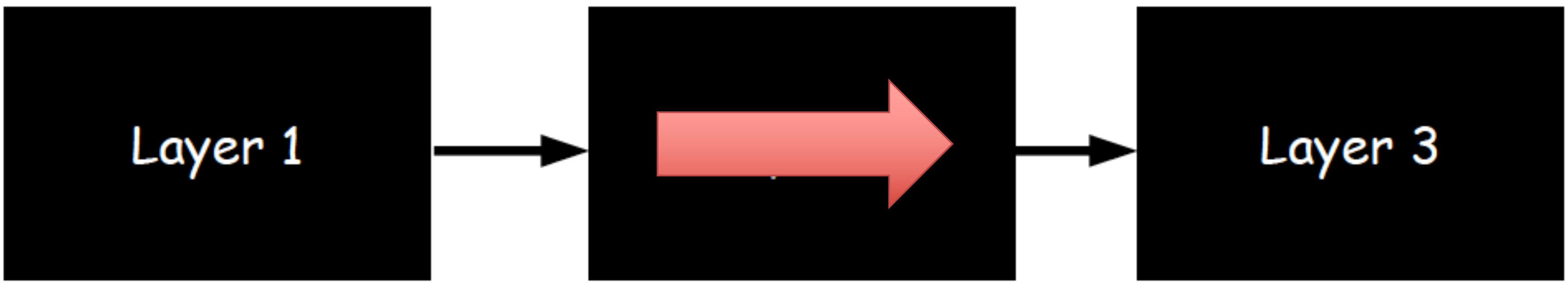
Neural Network Training

- Step 1: Compute Loss on mini-batch [F-Pass]



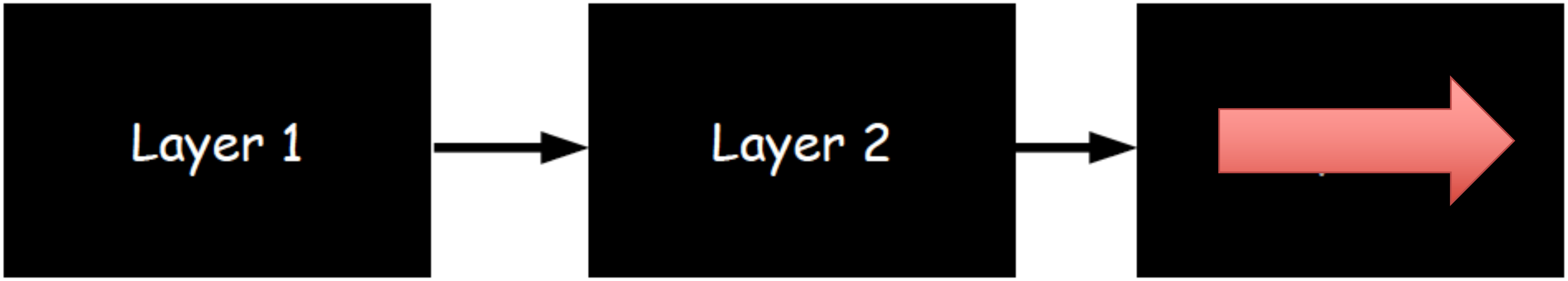
Neural Network Training

- Step 1: Compute Loss on mini-batch [F-Pass]



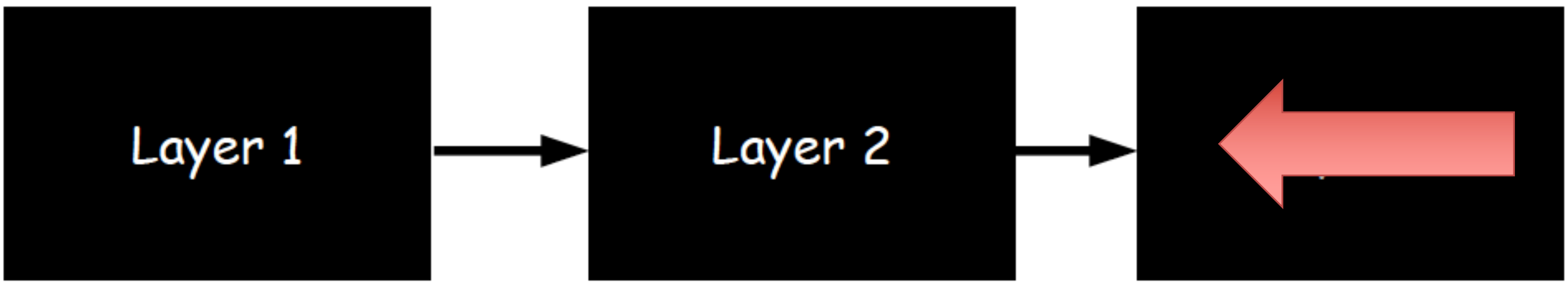
Neural Network Training

- Step 1: Compute Loss on mini-batch [F-Pass]



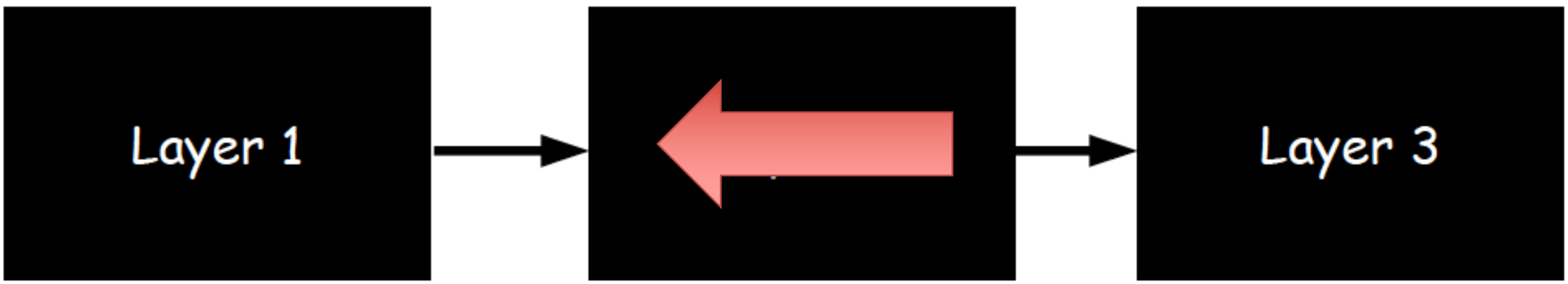
Neural Network Training

- Step 1: Compute Loss on mini-batch [F-Pass]
- Step 2: Compute gradients wrt parameters [B-Pass]



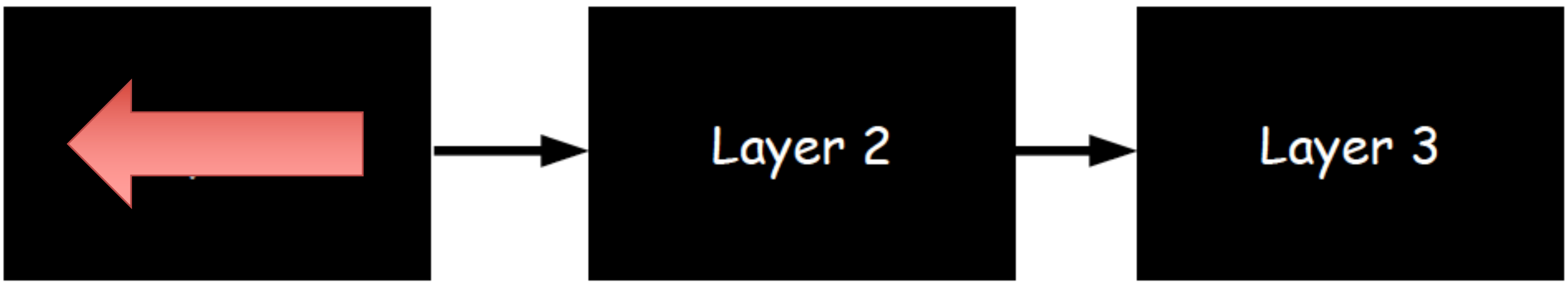
Neural Network Training

- Step 1: Compute Loss on mini-batch [F-Pass]
- Step 2: Compute gradients wrt parameters [B-Pass]



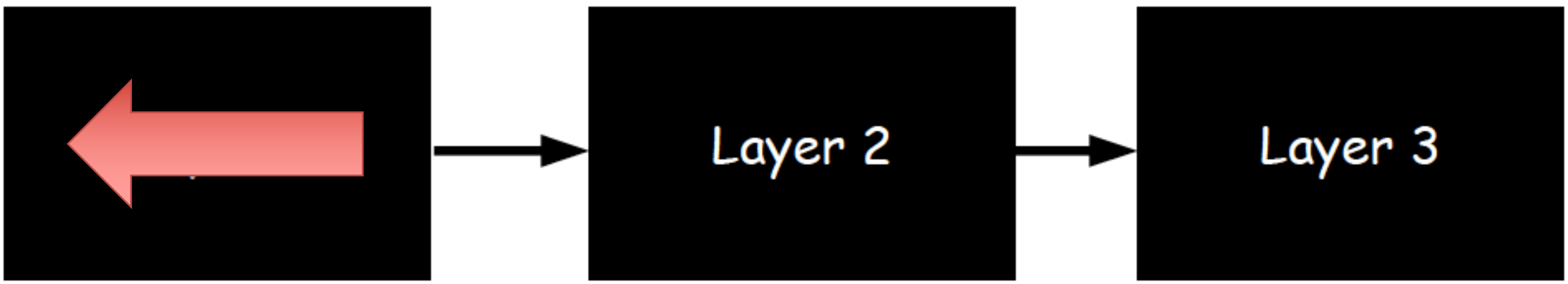
Neural Network Training

- Step 1: Compute Loss on mini-batch [F-Pass]
- Step 2: Compute gradients wrt parameters [B-Pass]

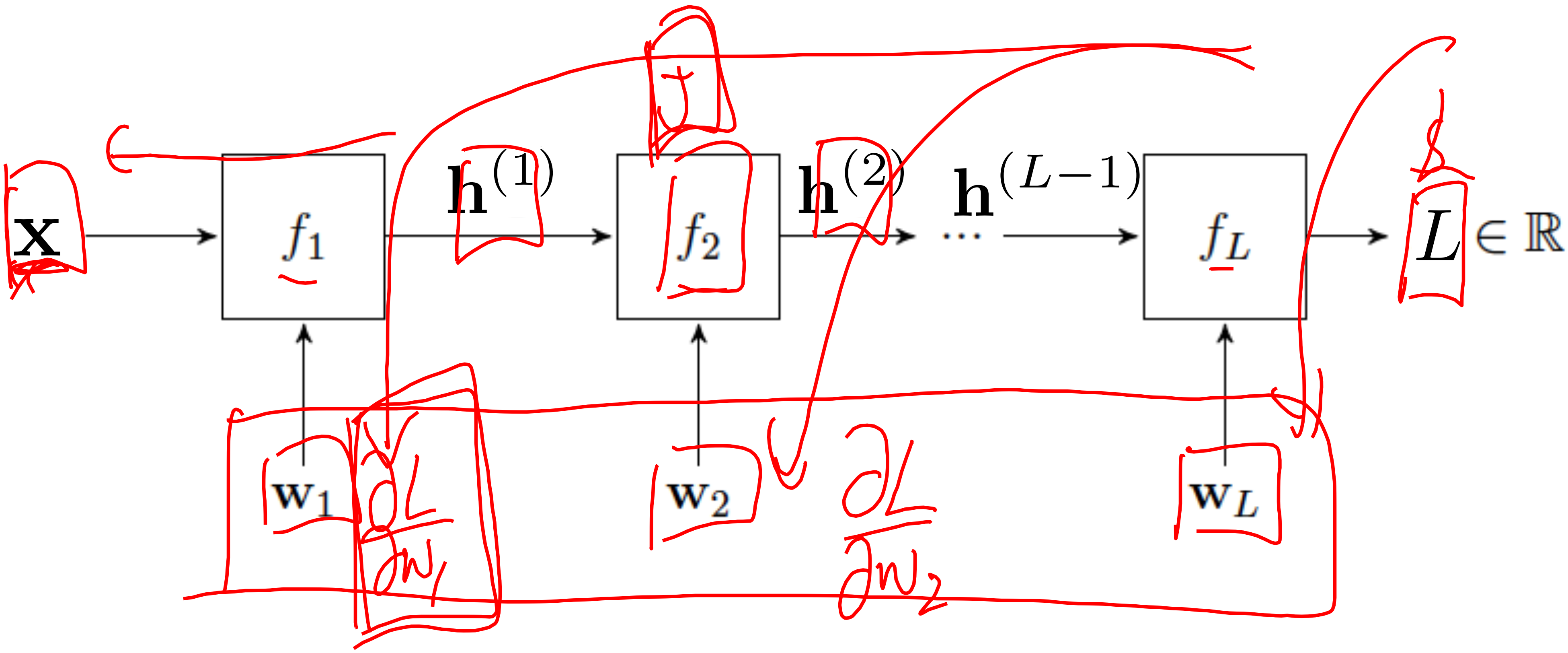


Neural Network Training

- Step 1: Compute Loss on mini-batch [F-Pass]
- Step 2: Compute gradients wrt parameters [B-Pass]
- Step 3: Use gradient to update parameters



$$\theta \leftarrow \theta - \eta \frac{dL}{d\theta}$$



$\mathbf{x}$
f_1
$\mathbf{h}^{(1)}$
f_2
$\mathbf{h}^{(2)}$
$\mathbf{h}^{(L-1)}$
f_L
$L \in \mathbb{R}$
$\mathbf{w}_1$
$\mathbf{w}_2$
$\mathbf{w}_L$
$\frac{\partial L}{\partial w_1}$
$\frac{\partial L}{\partial w_2}$