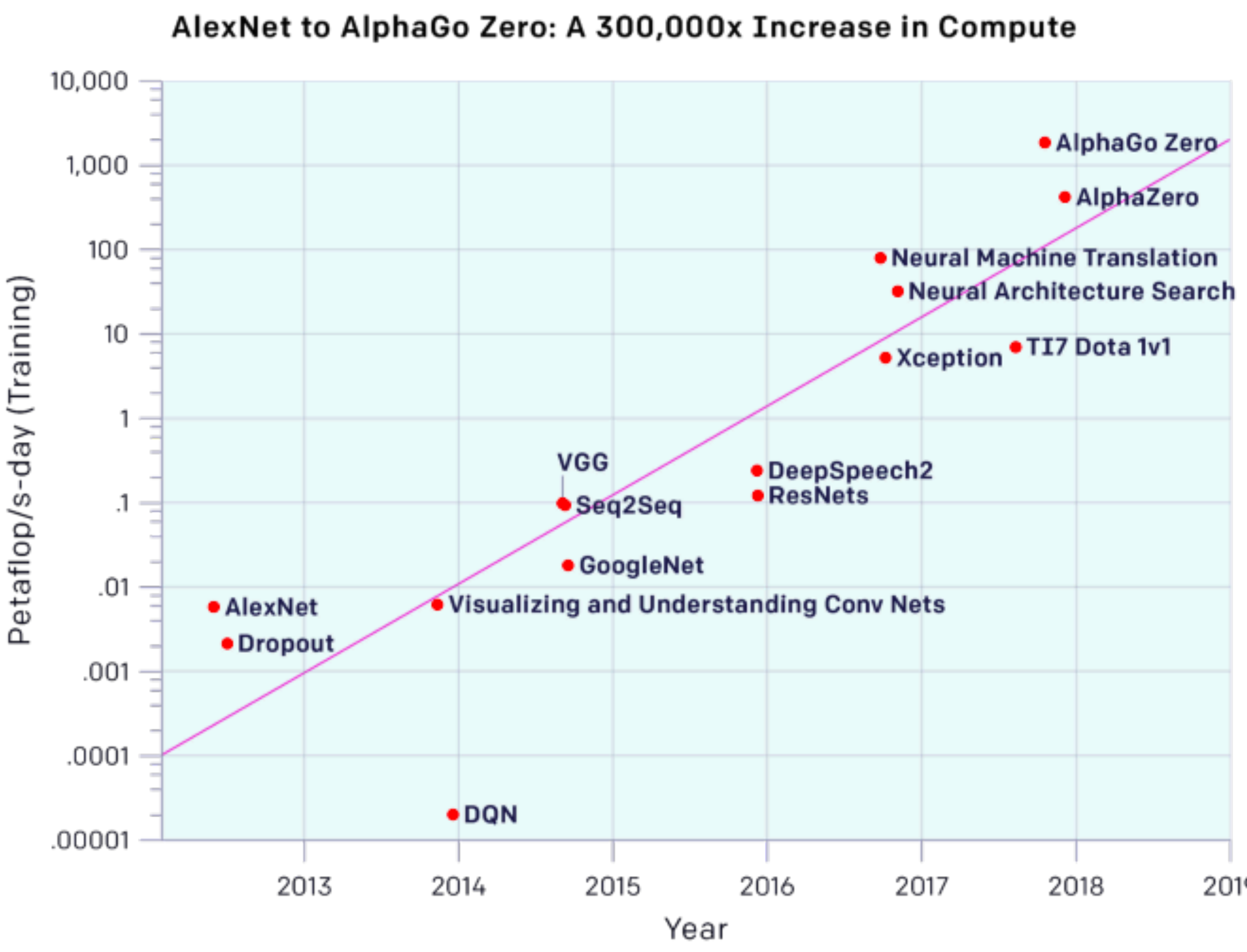
ML Systems Project

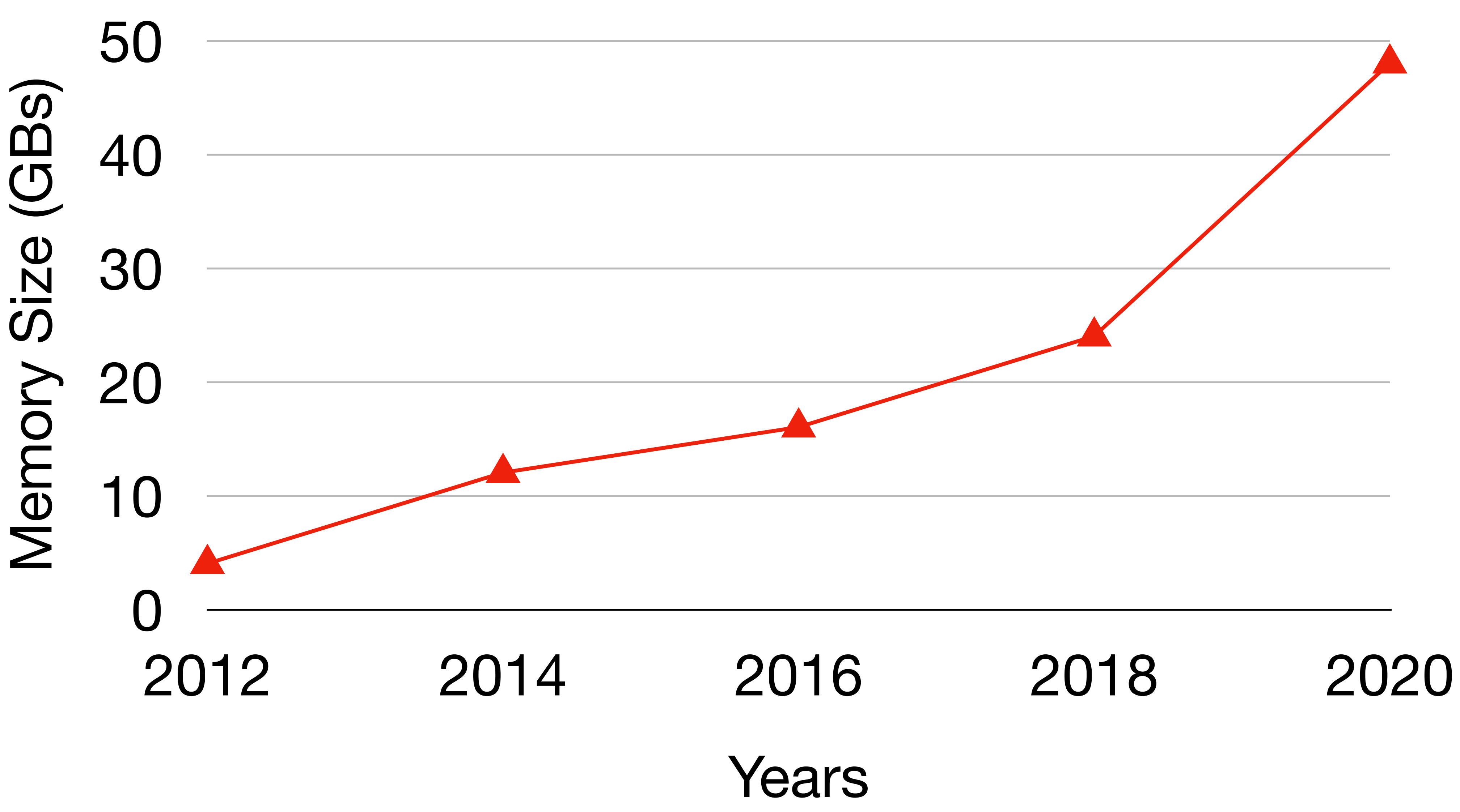
CS265 Spring 2025

DASlab
@ Harvard SEAS

Memory is the Bottleneck in NN Training



Open AI: https://openai.com/blog/ai-and-compute/



NVIDIA GPU Memory Sizes

DASlab @ Harvard SEAS

Neural Network Training

Forward Pass

Backward Pass

Black-box function: f
Input: X
Model Parameters: W

X , Y_{true} $\longrightarrow$ $\boxed{f(W,X)}$ $\longrightarrow$ Y_{pred} $\longrightarrow$ $L(Y_{true}, Y_{pred})$

$\frac{\nabla L}{\nabla W}$ Update

Loss Function: L
Prediction: Y_{pred}
True Label: Y_{true}

Loss (L): Distance from True Value

$\frac{\nabla L}{\nabla W}$: Gradient of Loss (L) wrt weights (W)

Forward Pass

$Input\ X$

$Z_1 = W_1 X$

$Z_2 = \sigma(Z_1)$

$Z_3 = W_3 Z_2$

$Z_4 = \sigma(Z_3)$

$L = \frac{1}{2}(Z_4 - Y)^2$

$True\ Label\ Y$

$$W_i = W_i - \alpha \nabla W_i$$

Update Rule

Function Composition

$y = f(x)$

$z = g(y)$

$z = g(f(x))$

$$\frac{\delta z}{\delta x} = \frac{\delta g(y)}{\delta y}\frac{\delta y}{\delta x}$$

$$\frac{\delta z}{\delta x} = \frac{\delta g(f(x))}{\delta f(x)}\frac{\delta f(x)}{\delta x}$$

Chain Rule

Backward Pass

$\nabla W_1 = \nabla Z_1\ X$

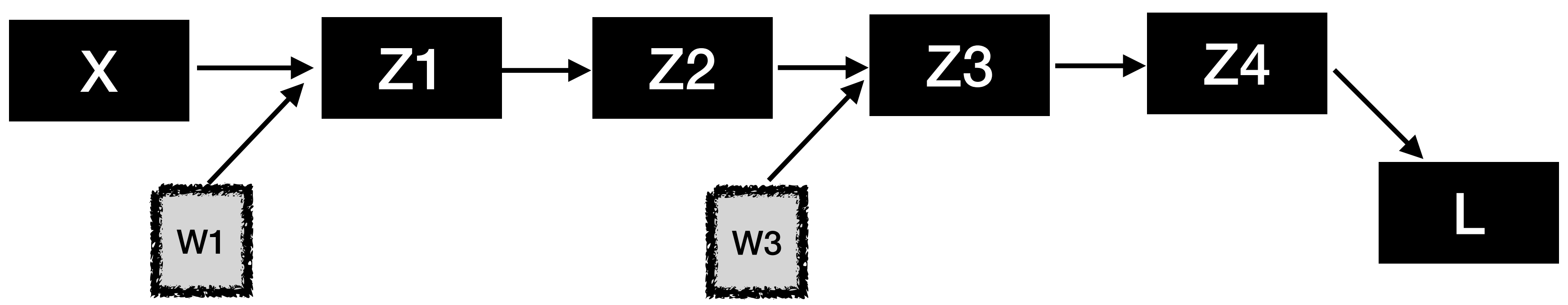
$\nabla Z_1 = \nabla Z_2\ Z_2\ (1 - Z_2)$

$\nabla Z_2 = \nabla Z_3\ W_3$

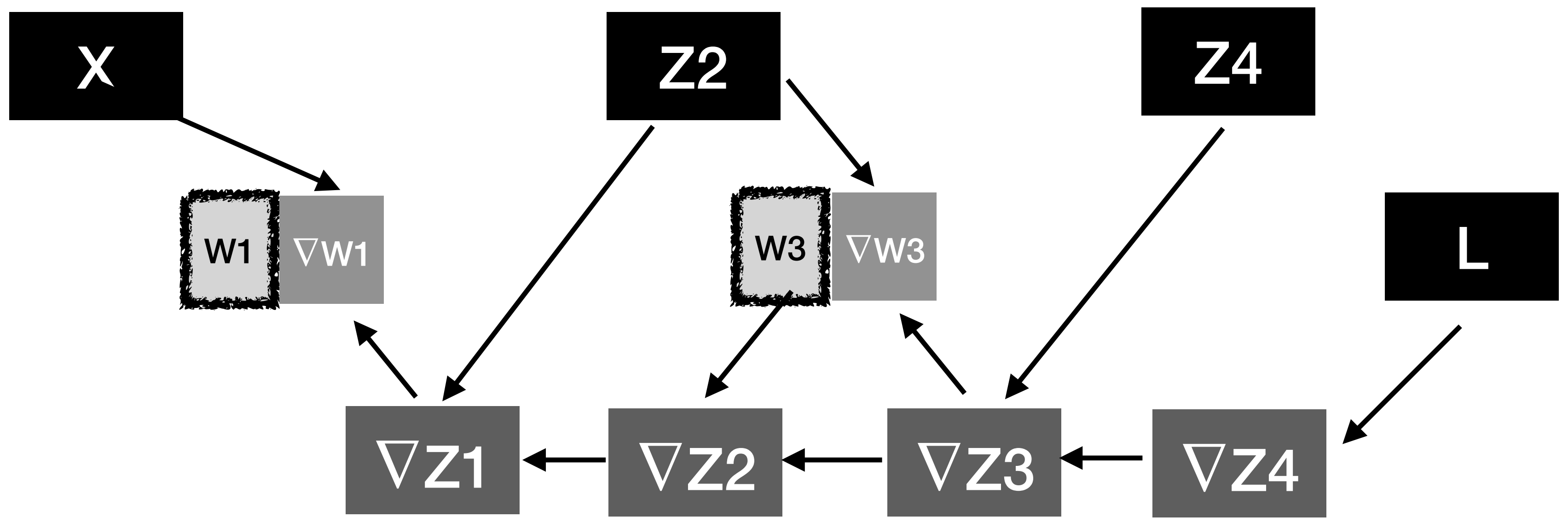
$\nabla W_3 = \nabla Z_3\ Z_2$

$\nabla Z_3 = \nabla Z_4\ Z_4\ (1 - Z_4)$

$\nabla Z_4 = (Z_4 - Y)$



Forward Computational Graph



Backward Computational Graph

Forward Pass

$Input\ X$

$Z_1 = W_1 X$

$Z_2 = \sigma(Z_1)$

$Z_3 = W_3 Z_2$

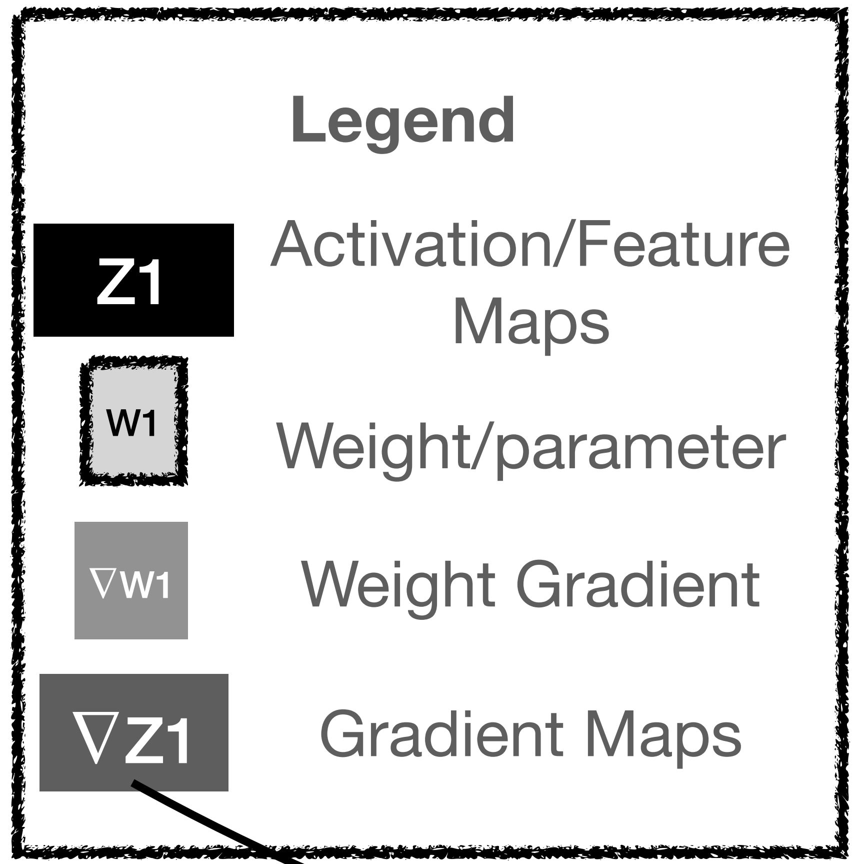
$Z_4 = \sigma(Z_3)$

$L = \frac{1}{2}(Z_4 - Y)^2$

$True\ Label\ Y$

$$W_i = W_i - \alpha \nabla W_i$$

Update Rule



Backward Pass

$\nabla W_1 = \nabla Z_1\ X$

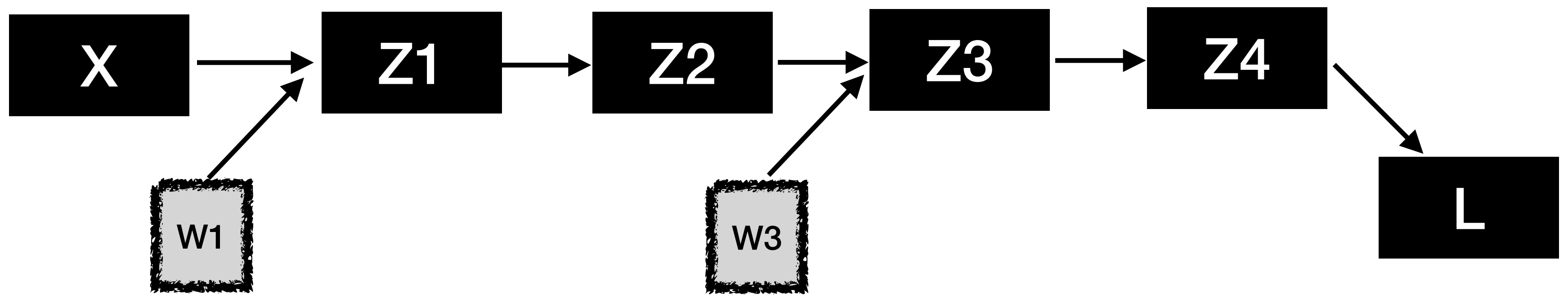
$\nabla Z_1 = \nabla Z_2\ Z_2\ (1 - Z_2)$

$\nabla Z_2 = \nabla Z_3\ W_3$

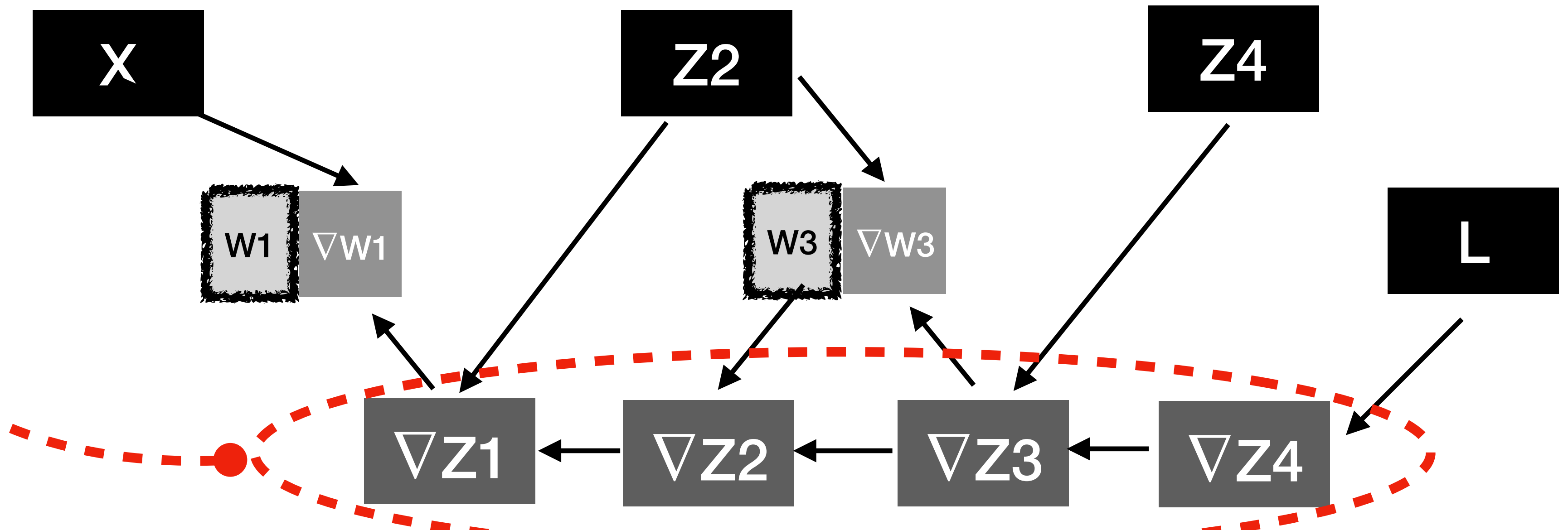
$\nabla W_3 = \nabla Z_3\ Z_2$

$\nabla Z_3 = \nabla Z_4\ Z_4\ (1 - Z_4)$

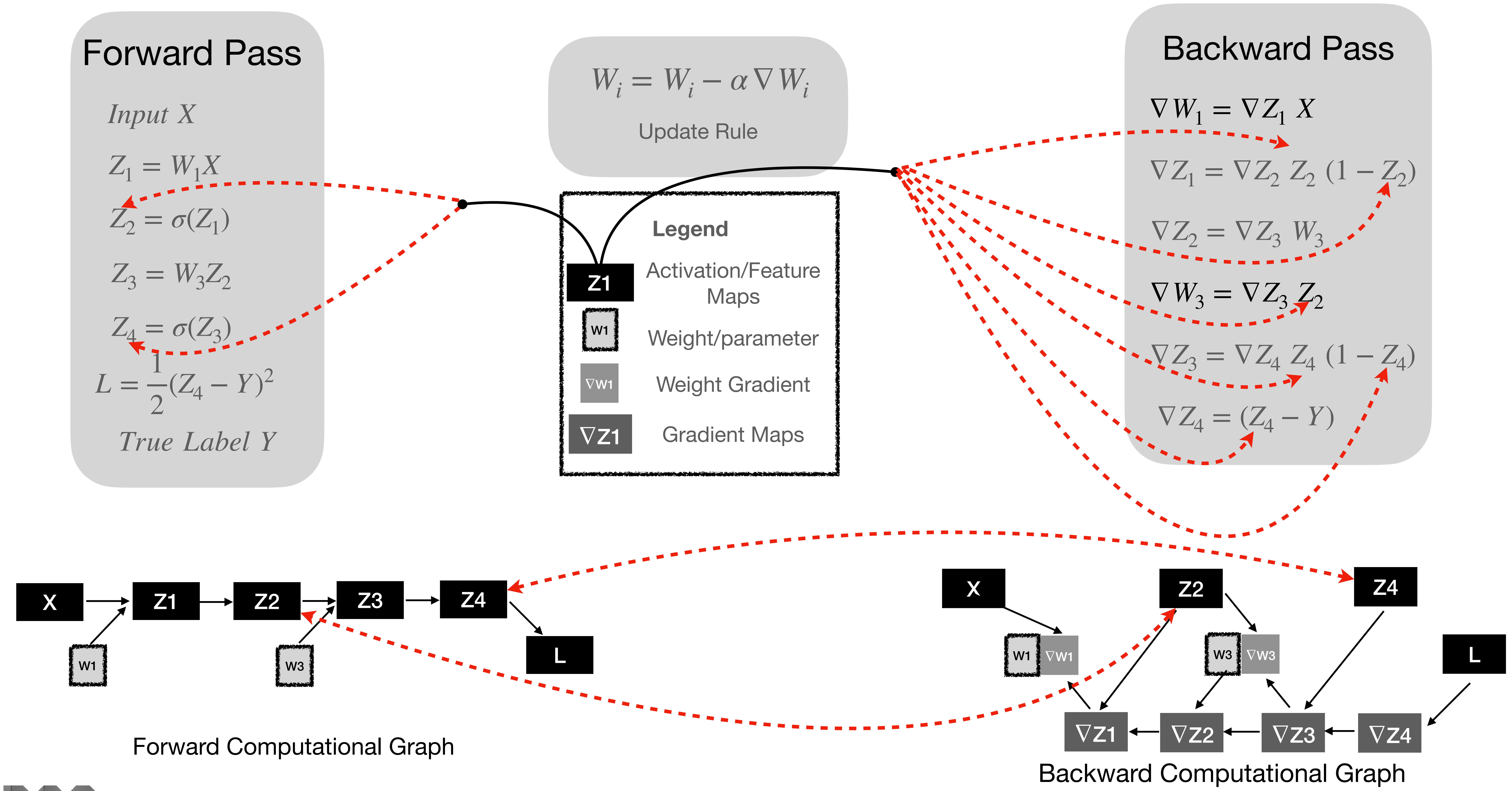
$\nabla Z_4 = (Z_4 - Y)$



Forward Computational Graph



Backward Computational Graph



DASlab
@ Harvard SEAS

Forward Pass

Input X

$Z_1 = W_1 X$

$Z_2 = \sigma(Z_1)$

$Z_3 = W_3 Z_2$

$Z_4 = \sigma(Z_3)$

$L = \frac{1}{2}(Z_4 - Y)^2$

True Label Y

$$W_i = W_i - \alpha \nabla W_i$$

Update Rule

Legend

- Z1 — Activation/Feature Maps
- W1 — Weight/parameter
- ∇W1 — Weight Gradient
- ∇Z1 — Gradient Maps

Backward Pass

$\nabla W_1 = \nabla Z_1 \; X$

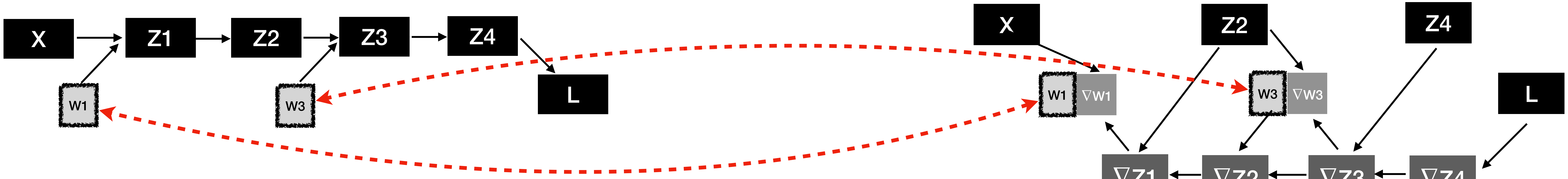
$\nabla Z_1 = \nabla Z_2 \; Z_2 \; (1 - Z_2)$

$\nabla Z_2 = \nabla Z_3 \; W_3$

$\nabla W_3 = \nabla Z_3 \; Z_2$

$\nabla Z_3 = \nabla Z_4 \; Z_4 \; (1 - Z_4)$

$\nabla Z_4 = (Z_4 - Y)$



Forward Computational Graph

Backward Computational Graph

DASlab @ Harvard SEAS

Forward Pass

$Input\ X$

$Z_1 = W_1 X$

$Z_2 = \sigma(Z_1)$

$Z_3 = W_3 Z_2$

$Z_4 = \sigma(Z_3)$

$L = \frac{1}{2}(Z_4 - Y)^2$

$True\ Label\ Y$

$$W_i = W_i - \alpha \nabla W_i$$

Update Rule

Legend

Z1	Activation/Feature Maps
W1	Weight/parameter
∇W1	Weight Gradient
∇Z1	Gradient Maps

Backward Pass

$\nabla W_1 = \nabla Z_1\ X$

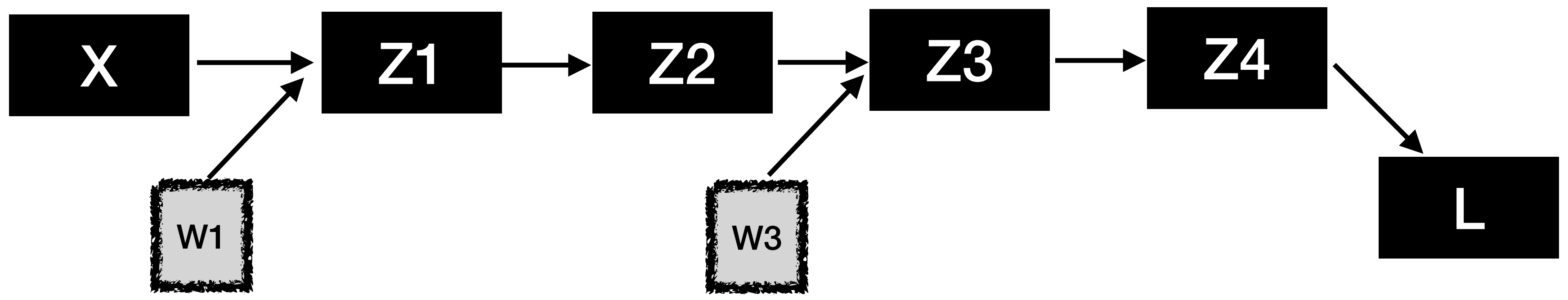
$\nabla Z_1 = \nabla Z_2\ Z_2\ (1 - Z_2)$

$\nabla Z_2 = \nabla Z_3\ W_3$

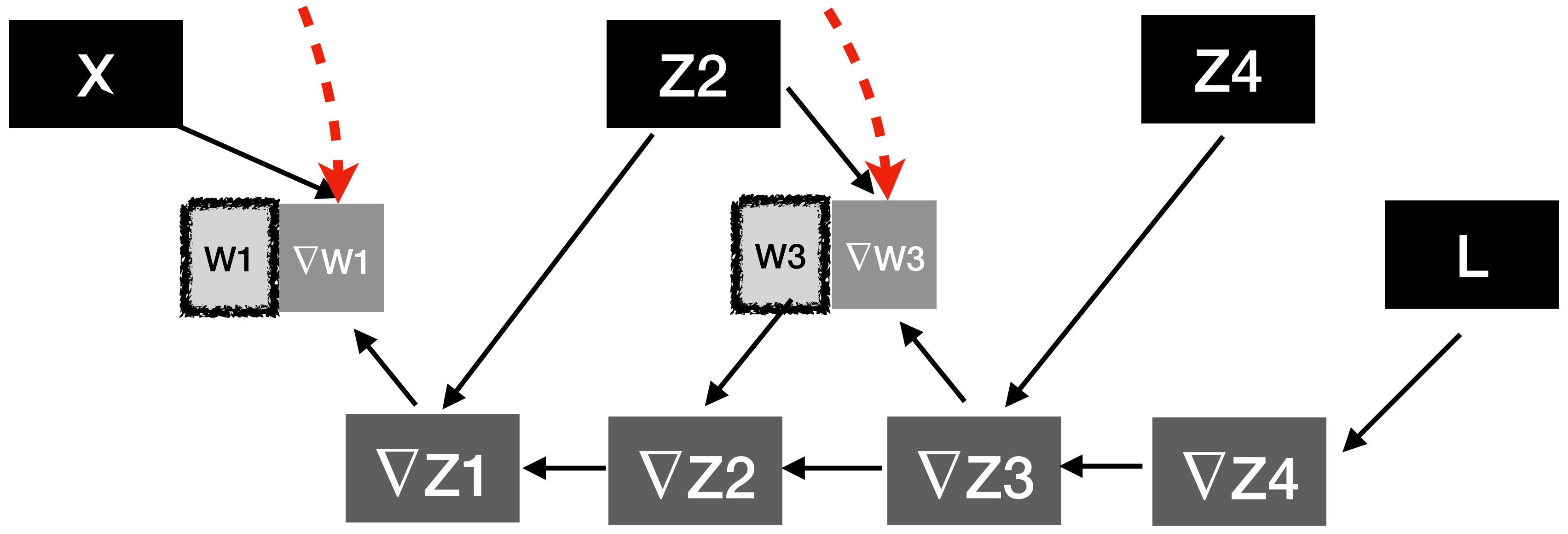
$\nabla W_3 = \nabla Z_3\ Z_2$

$\nabla Z_3 = \nabla Z_4\ Z_4\ (1 - Z_4)$

$\nabla Z_4 = (Z_4 - Y)$



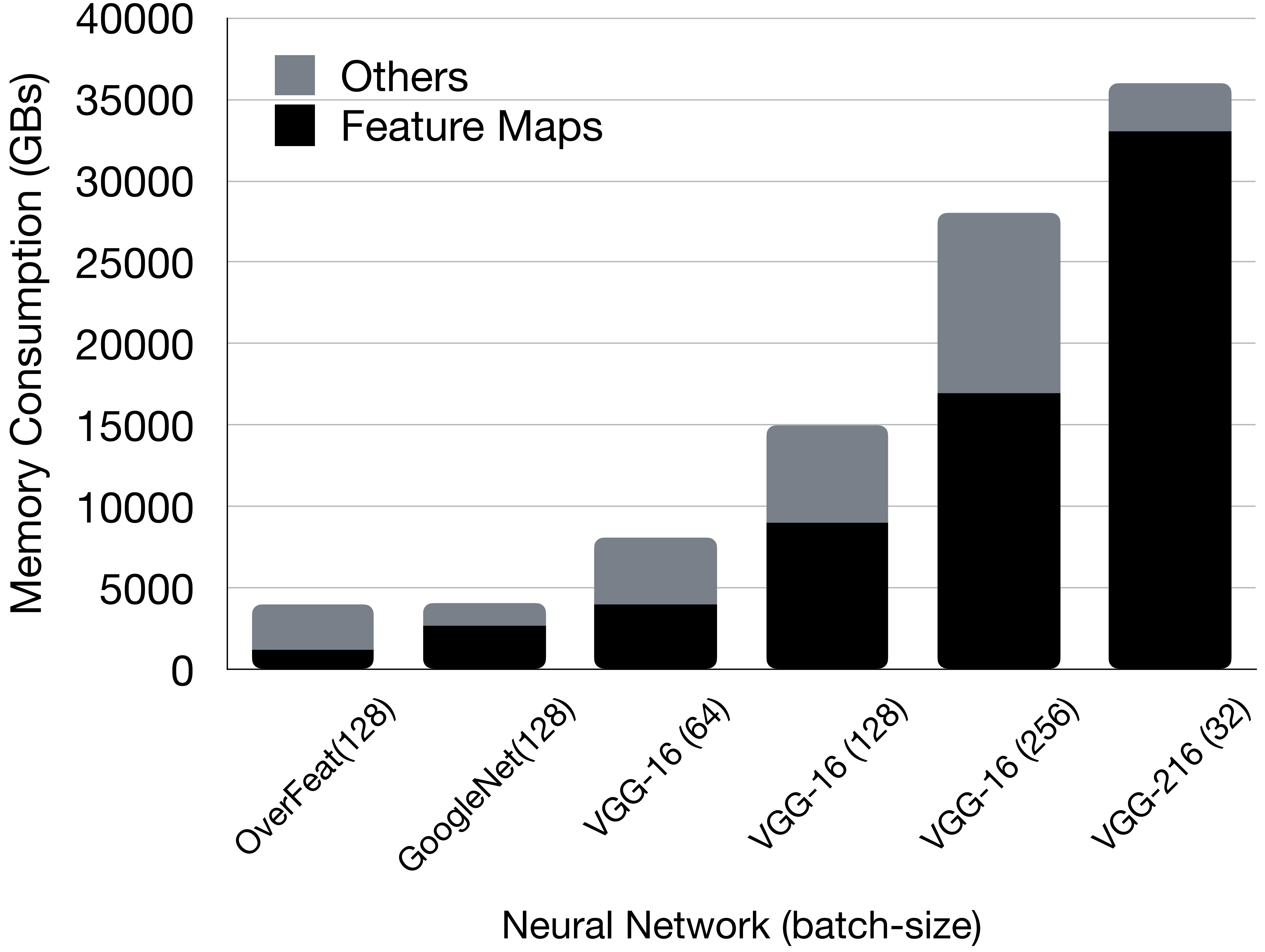
Forward Computational Graph



Backward Computational Graph

DASlab
@ Harvard SEAS

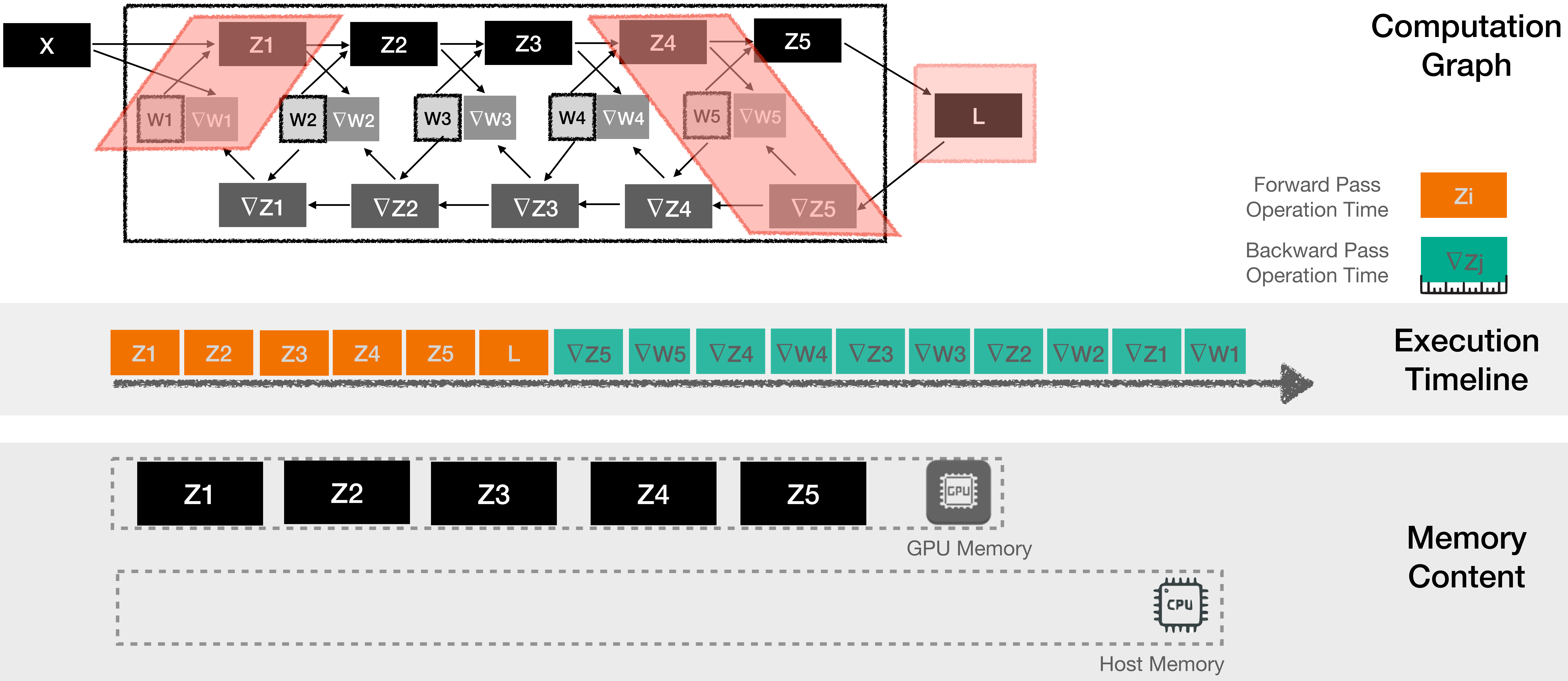
Reduce Memory Footprint



Adapted From: Minsoo Rhu et al. vDNN, MICRO 2016

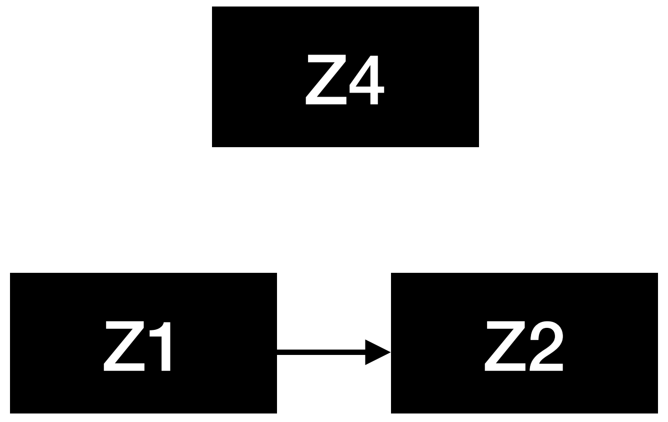
@ Harvard SEAS

Execution Framework

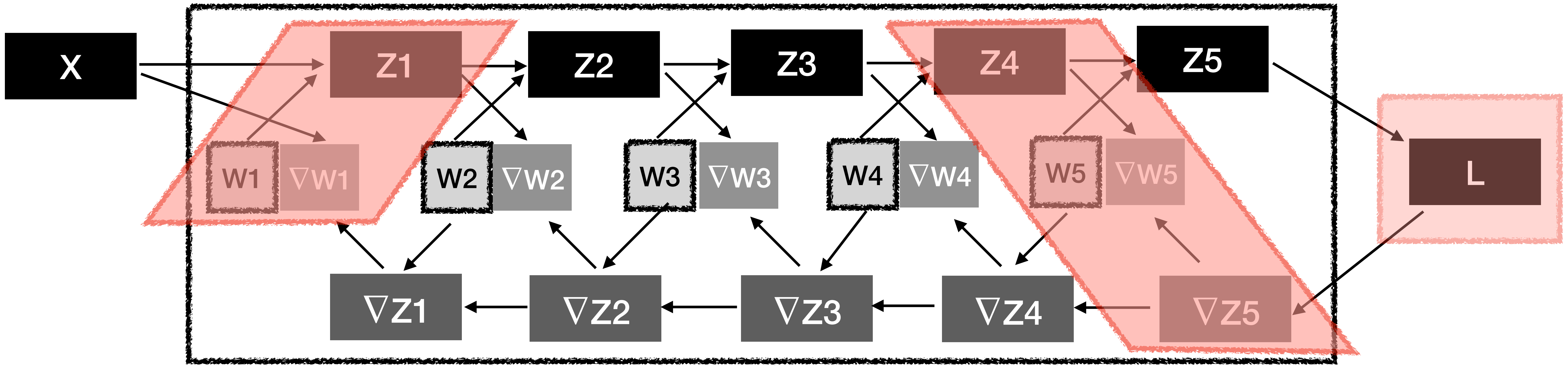


Activation Checkpointing

Recompute



Computation Graph

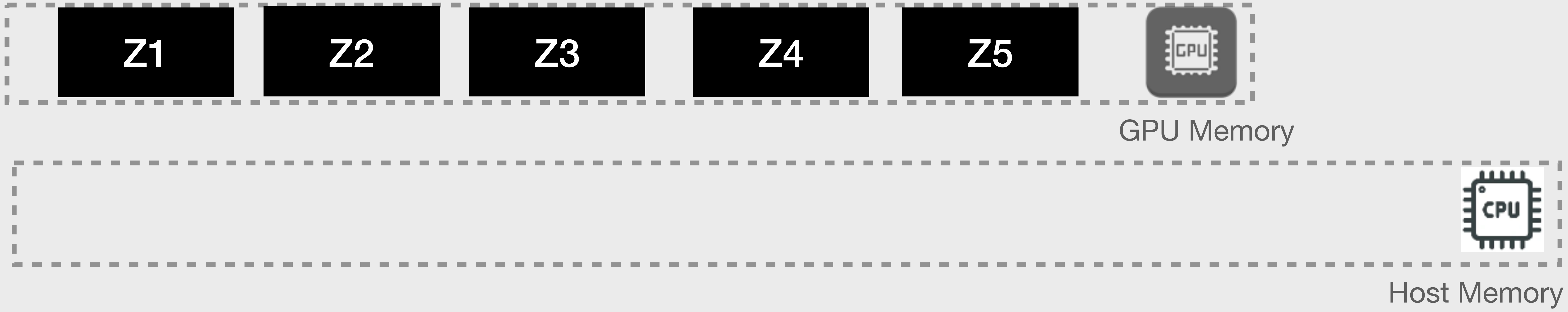


Forward Pass Operation Time — Zi

Backward Pass Operation Time — ∇Zj

Execution Timeline

Discard



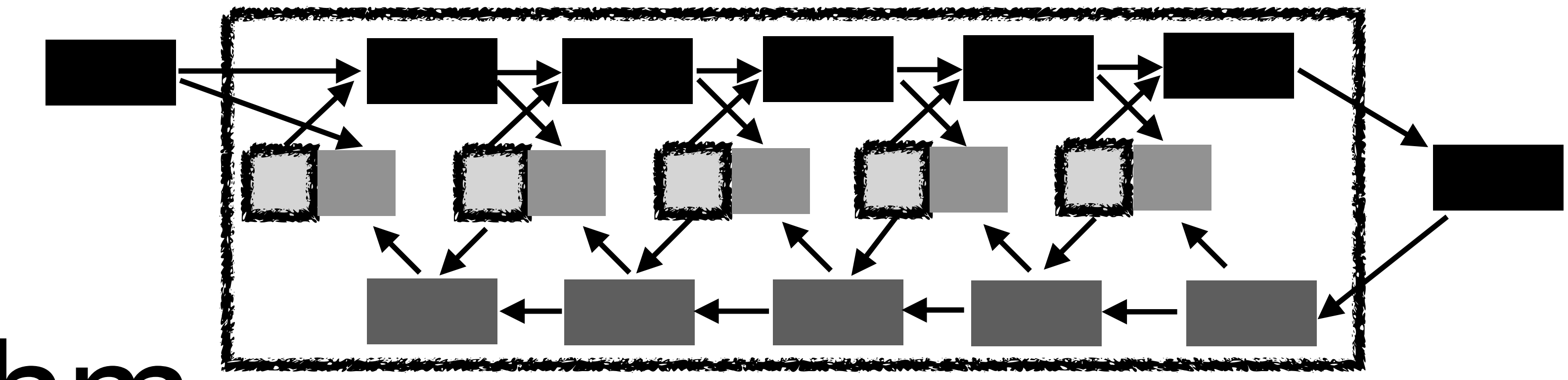
Memory Content

DASlab @ Harvard SEAS

Jain Paras et al.,Checkmate: Breaking the memory wall with optimal tensor rematerialization, *MLSys 2020*

The ML Systems Project

1. construct and profile the computational graph on PyTorch



2. design the activation checkpointing (AC) algorithm

3. rewrite the PyTorch computational graph to deploy activation checkpointing

http://daslab.seas.harvard.edu/classes/cs265/files/CS_265_Systems_Project_Description.pdf

How to Start

1. play with the starter code (instructions in READEME.md)

   https://github.com/qtwang/CS265-mlsys-project

2. check the reference paper

Sanket Purandare, Abdul Wasay, Animesh Jain, Stratos Idreos: μ-TWO: 3× Faster Multi-Model Training with Orchestration and Memory Optimization. MLSys 2023.

3. come to the labs with questions!

Midway Check-in

1. complete Phase 1: construct and profile the computational graph

2. a document with experimental analysis consisting of the following two deliverables:
    - computation and memory profiling statistics and static analysis
    - peak memory consumption vs. mini-batch size bar graph without AC

Final Deliverables

1. code deliverable + code review + demo = 50%

2. a document with experimental analysis = 50%
    - computation and memory profiling statistics and static analysis
    - peak memory consumption vs mini-batch size bar graph (w and w/o AC)
    - iteration latency vs mini-batch size performance graph (w and w/o AC)

Enjoy!