CS 265: Intro to Deep Learning Training and Projects

DODRIO

Distributed Data-parallel with Recomputing & Orchestration

Sanket Purandare
Lecture Outline
Lecture Outline

Neural Network Training Fundamentals
Lecture Outline

Neural Network Training Fundamentals

Distributed Data Parallel Training
Lecture Outline

- Neural Network Training Fundamentals
- Motivation and Problem Definition
- Distributed Data Parallel Training
Lecture Outline

- Neural Network Training Fundamentals
- Motivation and Problem Definition
- Distributed Data Parallel Training
- Project Description
Lecture Outline

- Neural Network Training Fundamentals
- Distributed Data Parallel Training
- Motivation and Problem Definition
- Project Description
- Ongoing work and Applications
Neural Network Training

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

Loss Function: $L$
Prediction: $Y_{\text{pred}}$
True Label: $Y_{\text{true}}$
Neural Network Training

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

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

$X, Y_{\text{true}} \rightarrow f(W, X) \rightarrow Y_{\text{pred}}$
Neural Network Training

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

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

Loss (L): Distance from True Value
Neural Network Training

Forward Pass

Black-box function: \( f \)
Input: \( X \)
Model Parameters: \( W \)

Loss Function: \( L \)
Prediction: \( Y_{\text{pred}} \)
True Label: \( Y_{\text{true}} \)

\[
X, Y_{\text{true}} \rightarrow f(W,X) \rightarrow Y_{\text{pred}} \rightarrow L(Y_{\text{true}}, Y_{\text{pred}})
\]

Loss (L): Distance from True Value
Neural Network Training

- Black-box function: \( f \)
- Input: \( X \)
- Model Parameters: \( W \)
- Loss Function: \( L \)
- Prediction: \( Y_{\text{pred}} \)
- True Label: \( Y_{\text{true}} \)

Loss (\( L \)): Distance from True Value
Neural Network Training

Black-box function: \( f \)
Input: \( X \)
Model Parameters: \( W \)

Loss Function: \( L \)
Prediction: \( Y_{\text{pred}} \)
True Label: \( Y_{\text{true}} \)

Loss (\( L \)): Distance from True Value

\[
X, Y_{\text{true}} \rightarrow f(W,X) \rightarrow Y_{\text{pred}} \rightarrow L(Y_{\text{true}}, Y_{\text{pred}})
\]
Neural Network Training

Backward Pass

Black-box function: \( f \)
Input: \( X \)
Model Parameters: \( W \)

\[
X, Y_{\text{true}} \rightarrow f(W,X) \rightarrow Y_{\text{pred}} \rightarrow L(Y_{\text{true}}, Y_{\text{pred}})
\]

Loss Function: \( L \)
Prediction: \( Y_{\text{pred}} \)
True Label: \( Y_{\text{true}} \)

Loss (L): Distance from True Value
Neural Network Training

Backward Pass

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

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

Loss ($L$): Distance from True Value

\[ \nabla L \]
\[ \nabla W \] : Gradient of Loss ($L$) wrt weights ($W$)
Forward Pass

Input $X$

$Z_1 = W_1X$

$Z_2 = \sigma(Z_1)$

$Z_3 = W_3Z_2$

$Z_4 = \sigma(Z_3)$

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

True Label $Y$
Forward Pass

Input $X$

$Z_1 = W_1X$

$Z_2 = \sigma(Z_1)$

$Z_3 = W_3Z_2$

$Z_4 = \sigma(Z_3)$

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

True Label $Y$

$W_i = W_i - \alpha \nabla W_i$

Update Rule
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

Function Composition

Forward Computational Graph

\[ W_i = W_i - \alpha \nabla W_i \]

Update Rule
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 $Y$

Forward Computational Graph

Function Composition

$W_i = W_i - \alpha \nabla W_i$

Update Rule

PyTorch @ Harvard SEAS
**Forward Pass**

- **Input $X$**
- $Z_1 = W_1X$
- $Z_2 = \sigma(Z_1)$
- $Z_3 = W_3Z_2$
- $Z_4 = \sigma(Z_3)$
- $L = \frac{1}{2}(Z_4 - Y)^2$
- *True Label $Y$*

**Update Rule**

\[ W_i = W_i - \alpha \nabla W_i \]

**Function Composition**

\[ y = f(x) \]

**Forward Computational Graph**
Input $X$

1. $Z_1 = W_1 X$
2. $Z_2 = \sigma(Z_1)$
3. $Z_3 = W_3 Z_2$
4. $Z_4 = \sigma(Z_3)$
5. $L = \frac{1}{2} (Z_4 - Y)^2$

True Label $Y$

$W_i = W_i - \alpha \nabla W_i$

Update Rule

Function Composition

$y = f(x)$
$z = g(y)$
\[ Z_1 = W_1 X \]
\[ Z_2 = \sigma(Z_1) \]
\[ Z_3 = W_3 Z_2 \]
\[ Z_4 = \sigma(Z_3) \]
\[ L = \frac{1}{2}(Z_4 - Y)^2 \]

**Input X**

**Forward Pass**

**Function Composition**

\[ y = f(x) \]
\[ z = g(y) \]
\[ z = g(f(x)) \]

**Forward Computational Graph**
Forward Pass

Input $X$

$Z_1 = W_1X$

$Z_2 = \sigma(Z_1)$

$Z_3 = W_3Z_2$

$Z_4 = \sigma(Z_3)$

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

True Label $Y$

Update Rule

$W_i = W_i - \alpha \nabla W_i$

Function Composition

$y = f(x)$

$z = g(y)$

$z = g(f(x))$

Chain Rule

Forward 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** $Y$

**Update Rule**

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

**Function Composition**

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

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

**Forward Computational Graph**
Forward Pass

- Input $X$
- $Z_1 = W_1X$
- $Z_2 = \sigma(Z_1)$
- $Z_3 = W_3Z_2$
- $Z_4 = \sigma(Z_3)$
- $L = \frac{1}{2}(Z_4 - Y)^2$

True Label $Y$

Update Rule

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

Function Composition

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

Chain Rule

$$\delta L \delta x = \frac{\partial g(y)}{\partial y} \delta y \frac{\partial f(x)}{\partial y} \delta y$$

Forward 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

Function Composition

$y = f(x)$

$z = g(y)$

$z = g(f(x))$

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

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*

\[
\begin{align*}
W_i &= W_i - \alpha \nabla W_i \\
\text{Update Rule}
\end{align*}
\]

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

\[ X \rightarrow Z_1 \rightarrow Z_2 \rightarrow Z_3 \rightarrow Z_4 \rightarrow L \]

\[ W_1 \rightarrow W_3 \rightarrow \nabla W_1 \rightarrow \nabla W_3 \rightarrow \nabla Z_1 \rightarrow \nabla Z_2 \rightarrow \nabla Z_3 \rightarrow \nabla Z_4 \rightarrow L \]

Legend

- **Z1**: Activation/Feature Maps
- **W1**: Weight/parameter
- **\nabla W1**: Weight Gradient
- **\nabla Z1**: Gradient Maps

PyTorch

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 $Y$

Backward Pass

Update Rule

$W_i = W_i - \alpha \nabla W_i$

$
\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)
$
\[ W_i = W_i - \alpha \nabla W_i \]

**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** \( Y \)

**Legend**
- **Activation/Feature Maps**
- **Weight/parameter**
- **Weight Gradient**
- **Gradient Maps**

**Backward Pass**

- \( \nabla W_1 = \nabla Z_1 \cdot X \)
- \( \nabla Z_1 = \nabla Z_2 \cdot Z_2 \cdot (1 - Z_2) \)
- \( \nabla Z_2 = \nabla Z_3 \cdot W_3 \)
- \( \nabla W_3 = \nabla Z_3 \cdot Z_2 \)
- \( \nabla Z_3 = \nabla Z_4 \cdot Z_4 \cdot (1 - Z_4) \)
- \( \nabla Z_4 = (Z_4 - Y) \)

**Forward Computational Graph**

**Legend**
- **Activation/Feature Maps**
- **Weight/parameter**
- **Weight Gradient**
- **Gradient Maps**
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 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 $Y$

Backward Pass

$W_i = W_i - \alpha \nabla W_i$

Update Rule

$\nabla W_1 = \nabla Z_1 X$
$\nabla Z_1 = \nabla Z_2 Z_2 (1 - Z_2)$
$\nabla Z_2 = \nabla Z_3 W_3$
$\nabla W_3 = \nabla Z_3 Z_2$
$\nabla Z_3 = \nabla Z_4 Z_4 (1 - Z_4)$
$\nabla Z_4 = (Z_4 - Y)$
Forward 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*

**Update Rule**

\[ W_i = W_i - \alpha \nabla W_i \]

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 Pass

Input $X$

$Z_1 = W_1X$

$Z_2 = \sigma(Z_1)$

$Z_3 = W_3Z_2$

$Z_4 = \sigma(Z_3)$

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

True Label $Y$

$W_i = W_i - \alpha \nabla W_i$

Update Rule

Backward Pass

$\nabla W_1 = \nabla Z_1 X$

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

$\nabla Z_2 = \nabla Z_3 W_3$

$\nabla W_3 = \nabla Z_4 Z_2$

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

$\nabla Z_4 = (Z_4 - Y)$

Legend

Activation/Feature Maps

Weight/parameter

Weight Gradient

Gradient Maps
Update Rule

\[ W_i = W_i - \alpha \nabla W_i \]

\[ \nabla Z_4 = (Z_4 - Y) \]

\[ \nabla Z_3 = \nabla Z_4 Z_2 (1 - Z_2) \]

\[ \nabla W_3 = \nabla Z_3 Z_2 \]

\[ \nabla Z_2 = \nabla Z_3 W_3 \]

\[ \nabla W_1 = \nabla Z_1 X \]

\[ \nabla Z_1 = \nabla Z_2 Z_2 (1 - Z_2) \]

\[ \nabla Z_4 = (Z_4 - Y) \]

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

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 \( Y \)

Backward Pass
- \( \nabla W_1 = \nabla Z_1 X \)
- \( \nabla Z_1 = \nabla Z_2 Z_2 (1 - Z_2) \)
- \( \nabla Z_2 = \nabla Z_3 W_3 \)
- \( \nabla W_3 = \nabla Z_3 Z_2 \)
- \( \nabla Z_3 = \nabla Z_4 Z_4 (1 - Z_4) \)
- \( \nabla Z_4 = (Z_4 - Y) \)

Legend
- Activation/Feature Maps
- Weight/parameter
- Weight Gradient
- Gradient Maps
Forward Pass

Input $X$

$Z_1 = W_1X$

$Z_2 = \sigma(Z_1)$

$Z_3 = W_3Z_2$

$Z_4 = \sigma(Z_3)$

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

True Label $Y$

$W_i = W_i - \alpha \nabla W_i$

Update Rule

Backward Pass

$\nabla W_1 = \nabla Z_1 \times X$

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

$\nabla Z_2 = \nabla Z_3 \times W_3$

$\nabla W_3 = \nabla Z_3 \times Z_2$

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

$\nabla Z_4 = (Z_4 - Y)$

Forward Computational Graph

Backward Computational Graph
Forward Pass

- Input $X$
- $Z_1 = W_1X$
- $Z_2 = \sigma(Z_1)$
- $Z_3 = W_2Z_2$
- $Z_4 = \sigma(Z_3)$
- $L = \frac{1}{2}(Z_4 - Y)^2$
- True Label $Y$

Backward Pass

- Update Rule: $W_i = W_i - \alpha \nabla W_i$
- $\nabla W_1 = \nabla Z_1 X$
- $\nabla Z_1 = \nabla Z_2 Z_2 (1 - Z_2)$
- $\nabla Z_2 = \nabla Z_3 W_3$
- $\nabla W_3 = \nabla Z_3 Z_2$
- $\nabla Z_3 = \nabla Z_4 Z_4 (1 - Z_4)$
- $\nabla Z_4 = (Z_4 - Y)$

Legend:
- Activation/Feature Maps
- Weight/parameter
- Weight Gradient
- Gradient Maps

Forward Computational Graph

Backward Computational Graph
Forward Pass

Input $X$

$Z_1 = W_1 X$

$Z_2 = \sigma(Z_1)$

$Z_3 = W_2 Z_2$

$Z_4 = \sigma(Z_3)$

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

True Label $Y$

Backward Pass

$\nabla W_1 = \nabla Z_1 X$

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

$\nabla Z_2 = \nabla Z_3 W_3$

$\nabla W_3 = \nabla Z_3 Z_2$

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

$\nabla Z_4 = (Z_4 - Y)$

Gradient Maps

Weight/parameter

Activation/Feature Maps

Update Rule

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

Backward Pass

- $\nabla W_1 = \nabla Z_4 X$
- $\nabla Z_1 = \nabla Z_2 Z_2 (1 - Z_2)$
- $\nabla Z_2 = \nabla Z_3 W_3$
- $\nabla W_3 = \nabla Z_3 Z_2$
- $\nabla Z_3 = \nabla Z_4 Z_4 (1 - Z_4)$
- $\nabla Z_4 = (Z_4 - Y)$

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

\[ \nabla W_1 = \nabla Z_1 \cdot X \]
\[ \nabla Z_1 = \nabla Z_2 \cdot Z_2 \cdot (1 - Z_2) \]
\[ \nabla Z_2 = \nabla Z_3 \cdot \nabla W_3 \]
\[ \nabla W_3 = \nabla Z_3 \cdot Z_2 \]
\[ \nabla Z_3 = \nabla Z_4 \cdot Z_4 \cdot (1 - Z_4) \]
\[ \nabla Z_4 = (Z_4 - Y) \]
Motivation

Large Memory Footprint

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

Linear increase in GPU memory sizes
Motivation

Linear increase in GPU memory sizes

![Graph showing linear increase in GPU memory sizes from 2012 to 2020. The x-axis represents years (2012 to 2020) and the y-axis represents memory size in GBs. The memory size increases from approximately 5 GBs in 2012 to around 50 GBs in 2020.]

NVIDIA GPU Memory Sizes
Scaling out
Distributed parallelization paradigms

Model Parallelism
Machine 1
Machine 2
Machine 3
Machine 4

Data Parallelism
Machine 1
Machine 2
Machine 3
Machine 4

PyTorch @ Harvard SEAS
Distributed Data Parallel (DDP)
Distributed Data Parallel (DDP)

Replicate the entire model (all weights) across all the GPUs
Distributed Data Parallel

Split the incoming batch into micro batches
Run all the models simultaneously

Mini Batch

Micro Batch 1

Micro Batch 2

Micro Batch 3

Micro Batch 4
Distributed Data Parallel

Each GPU generates its own gradients based on same weights but different data
Distributed Data Parallel

Communicate and accumulate gradients at each iteration.
Distributed Data Parallel

Each GPU gets all the reduced gradients
All the weights across all the GPUs receive identical weight update
All Reduce

Process 1
1 2 3 4

Process 2
1 2 3 4

Process 3
1 2 3 4

Process 4
1 2 3 4

AllReduce

Process 1
1 2 3 4

Process 2
1 2 3 4

Process 3
1 2 3 4

Process 4
1 2 3 4
Execution Timeline per Iteration
Execution Timeline per Iteration
Execution Timeline per Iteration

FW-1  BW-1

Execution Timeline
Execution Timeline per Iteration

- FW-1
- BW-1
- C1
Execution Timeline per Iteration

Execution Timeline
Execution Timeline per Iteration

FW-1  BW-1  C1  FW-2  BW-2

Execution Timeline
Execution Timeline per Iteration

FW-1  BW-1  C1  FW-2  BW-2  C2

Execution Timeline
Execution Timeline per Iteration
Execution Timeline per Iteration

Forward Pass Operation Time
Backward Pass Operation Time
Gradient Communication Time

Z1 Z2 Z3 Z4 Z5 L
∇Z1 ∇Z2 ∇Z3 ∇Z4 ∇Z5
∇W1 ∇W2 ∇W3 ∇W4 ∇W5

Compute
Communicate

PyTorch
DASlab @ Harvard SEAS
Execution Timeline per Iteration

Execution Timeline (Lazy)
## Communication Strategy Tradeoffs

### Parameters vs Strategy

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Eager</th>
<th>Lazy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bandwidth Utilization</td>
<td><img src="image" alt="Meter" /></td>
<td>Lowest</td>
</tr>
<tr>
<td>Communication Overhead</td>
<td><img src="image" alt="Globe" /></td>
<td>Highest</td>
</tr>
<tr>
<td>Compute Overlap</td>
<td>Partial</td>
<td>Nil</td>
</tr>
</tbody>
</table>

### Diagram

- **Eager**: \( \nabla W_5 \) \( \nabla W_4 \) \( \nabla W_3 \) \( \nabla W_2 \) \( \nabla W_1 \)
- **Lazy**: \( \nabla W_5 \nabla W_4 \nabla W_3 \nabla W_2 \nabla W_1 \)

- Gradient Communication Time: \( \nabla W_k \)
Bucketing Strategy

Execution Timeline (Bucketing)
# Communication Strategy Tradeoffs

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Eager</th>
<th>Bucketing</th>
<th>Lazy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bandwidth Utilization</td>
<td>Lowest</td>
<td>High</td>
<td>Highest</td>
</tr>
<tr>
<td>Communication Overhead</td>
<td>Highest</td>
<td>Low</td>
<td>Lowest</td>
</tr>
<tr>
<td>Compute Overlap</td>
<td>Partial</td>
<td>Maximal</td>
<td>Nil</td>
</tr>
</tbody>
</table>

## Tradeoffs

- **Eager**
  - Bandwidth Utilization: Lowest
  - Communication Overhead: Highest
  - Compute Overlap: Partial

- **Bucketing**
  - Bandwidth Utilization: High
  - Communication Overhead: Low
  - Compute Overlap: Maximal

- **Lazy**
  - Bandwidth Utilization: Highest
  - Communication Overhead: Lowest
  - Compute Overlap: Nil
Real World Scenario

Execution Timeline (Bucketing Strategy-1)
Real World Scenario

∇Z4 and ∇W2 are large matrix multiplies

Execution Timeline (Bucketing Strategy-1)
Real World Scenario

∇Z4 and ∇W2 are large matrix multiplies

∇W3 is large

Execution Timeline (Bucketing Strategy-1)
Real World Scenario

\[ \nabla Z_4 \text{ and } \nabla W_2 \text{ are large matrix multiplies} \]

\[ \nabla Z_4 \text{ and } \nabla W_2 \text{ are large matrix multiplies} \]

Variable Time of Operations
Real World Scenario

Variable Time of Operations
Variable Sizes of Weight Gradients

\( \nabla Z_4 \) and \( \nabla W_2 \) are large matrix multiplies

\( \nabla W_3 \) is large
Real World Scenario

Variable Time of Operations
Variable Sizes of Weight Gradients
Communication time with respect to data size
Real World Scenario

Execution Timeline (Bucketing Strategy-1)
Real World Scenario

Execution Timeline (Bucketing Strategy-1)

Execution Timeline (Bucketing Strategy-2)
Real World Scenario

Execution Timeline (Bucketing Strategy-2)

Want a Strategy that is..

Adaptive to operation times
Adaptive to communication time
Considers variable size buckets
Real World Scenario

Execution Timeline (Bucketing Strategy-2)

Want a Strategy that is..

Adaptive to operation times
Adaptive to communication time
Considers variable size buckets
Ingredients for the strategy

- Accurate Operator Times
- Accurate Communication Time
Project 1: GPU Kernel Time Prediction using Machine Learning

**Forward Pass**

- Input $X$
- $Z_1 = W_1X$
- $Z_2 = \sigma(Z_1)$
- $Z_3 = W_3Z_2$
- $Z_4 = \sigma(Z_3)$
- $L = \frac{1}{2}(Z_4 - Y)^2$
- True Label $Y$

**Backward Pass**

- $\nabla W_1 = \nabla Z_1 \ X$
- $\nabla Z_1 = \nabla Z_2 \ Z_2 (1 - Z_2)$
- $\nabla Z_2 = \nabla Z_3 \ W_3$
- $\nabla W_3 = \nabla Z_3 \ Z_2$
- $\nabla Z_3 = \nabla Z_4 \ Z_4 (1 - Z_4)$
- $\nabla Z_4 = (Z_4 - Y)$
Project 1: GPU Kernel Time Prediction using Machine Learning

What is a GPU Kernel?

Forward Pass
Input $X$
$Z_1 = W_i X$
$Z_2 = \sigma(Z_1)$
$Z_3 = W_2 Z_2$
$Z_4 = \sigma(Z_3)$
$L = \frac{1}{2} (Z_4 - Y)^2$
True Label $Y$

Backward Pass
$\nabla W_1 = \nabla Z_4 X$
$\nabla Z_1 = \nabla Z_2 Z_2 (1 - Z_2)$
$\nabla Z_2 = \nabla Z_3 W_3$
$\nabla W_3 = \nabla Z_3 Z_2$
$\nabla Z_3 = \nabla Z_4 Z_4 (1 - Z_4)$
$\nabla Z_4 = (Z_4 - Y)$

Kernel 1- GEMM
$Z_{i+1} = W_i \ast Z_i$
Project 1: GPU Kernel Time Prediction using Machine Learning

**What is a GPU Kernel?**

- **Forward Pass**
  - Input $X$
  - $Z_1 = W_1 X$
  - $Z_2 = \sigma(Z_1)$
  - $Z_3 = W_2 Z_2$
  - $Z_4 = \sigma(Z_3)$
  - $L = \frac{1}{2}(Z_4 - Y)^2$
  - True Label $Y$

- **Backward Pass**
  - $\nabla W_1 = \nabla Z_1 X$
  - $\nabla Z_1 = \nabla Z_2 Z_2 (1 - Z_2)$
  - $\nabla Z_2 = \nabla Z_3 W_3$
  - $\nabla W_3 = \nabla Z_3 Z_2$
  - $\nabla Z_3 = \nabla Z_4 Z_4 (1 - Z_4)$
  - $\nabla Z_4 = (Z_4 - Y)$

**CUDA Kernel**

Kernel 1 - GEMM
- $Z_{i+1} = W_i \ast Z_i$

**Inputs:** $W_i, Z_i$
**Outputs:** $Z_{i+1}$
**Function:** GEMM
**Type:** float_32
Project 1: GPU Kernel Time Prediction using Machine Learning

What is a GPU Kernel?

**Forward Pass**

- *Input X*
- $Z_1 = W_1 X$
- $Z_2 = \sigma(Z_1)$
- $Z_3 = W_2 Z_2$
- $Z_4 = \sigma(Z_3)$
- $L = \frac{1}{2}(Z_4 - Y)^2$
- *True Label Y*

**Backward Pass**

- $\nabla W_1 = \nabla Z_4 X$
- $\nabla Z_1 = \nabla Z_2 Z_2 (1 - Z_2)$
- $\nabla Z_2 = \nabla Z_3 W_1$
- $\nabla W_3 = \nabla Z_3 Z_2$
- $\nabla Z_3 = \nabla Z_4 Z_4 (1 - Z_4)$
- $\nabla Z_4 = (Z_4 - Y)$

**CUDA Kernel**

- **Kernel 1 - GEMM**
  - $Z_{i+1} = W_i * Z_i$
  - **Inputs:** $W_i, Z_i$
  - **Outputs:** $Z_{i+1}$
  - **Function:** GEMM
  - **Type:** float_32
- **Launch**
Project 1: GPU Kernel Time Prediction using Machine Learning

What could be a potential feature set?
Project 1: GPU Kernel Time Prediction using Machine Learning

What could be a potential feature set?

CUDA Kernel
- Inputs: $W_i, Z_i$
- Outputs: $Z_{i+1}$
- Function: GEMM
- Type: float_32

Kernel 1 - GEMM
$Z_{i+1} = W_i \ast Z_i$

Launch

Input Parameters
Project 1: GPU Kernel Time Prediction using Machine Learning

What could be a potential feature set?

CUDA Kernel

Inputs: \( W_i, Z_i \)
Outputs: \( Z_{i+1} \)
Function: GEMM
Type: float_32

Kernel 1 - GEMM
\[ Z_{i+1} = W_i \times Z_i \]

Input Parameters
- Dimensions
- Data Type
- Stride
- Memory Layout
What could be a potential feature set?

CUDA Kernel

Kernel 1 - GEMM

$Z_{i+1} = W_i \times Z_i$

Inputs: $W_i, Z_i$
Outputs: $Z_{i+1}$
Function: GEMM
Type: float_32

Input Parameters
- Dimensions
- Data Type
- Stride
- Memory Layout

Kernel Characteristics
Project 1: GPU Kernel Time Prediction using Machine Learning

What could be a potential feature set?

CUDA Kernel

Inputs: $W_i, Z_i$
Outputs: $Z_{i+1}$
Function: GEMM
Type: float_32

Kernel 1 - GEMM
$Z_{i+1} = W_i \ast Z_i$

Launch

Input Parameters
Dimensions
Data Type
Stride
Memory Layout

Kernel Characteristics
Operation (ex. Conv)
Implementation (gemm vs fft)
Project 1: GPU Kernel Time Prediction using Machine Learning

What could be a potential feature set?

CUDA Kernel

Kernel 1- GEMM

$Z_{i+1} = W_i \cdot Z_i$

Launch

Input Parameters
- Dimensions
- Data Type
- Stride
- Memory Layout

Kernel Characteristics
- Operation (ex. Conv)
- Implementation (gemm vs fft)

Static Code Features
Project 1: GPU Kernel Time Prediction using Machine Learning

What could be a potential feature set?

CUDA Kernel

Input Parameters
- Dimensions
- Data Type
- Stride
- Memory Layout

Kernel Characteristics
- Operation (ex. Conv)
- Implementation (gemm vs fft)

Static Code Features
- Instruction Counts
- Loads
- Stores
- Instruction Types
- Arithmetic Intensity
- No. of Thread Blocks
- No. of Warps

Kernel 1 - GEMM

$Z_{i+1} = W_i \ast Z_i$

Launch

Input: $W_i, Z_i$
Output: $Z_{i+1}$
Function: GEMM
Type: float_32
Project 1: GPU Kernel Time Prediction using Machine Learning

What could be a potential feature set?

CUDA Kernel

Kernel 1 - GEMM

\[ Z_{i+1} = W_i \times Z_i \]

Inputs: \( W_i, Z_i \)
Outputs: \( Z_{i+1} \)
Function: GEMM
Type: float_32

Input Parameters
- Dimensions
- Data Type
- Stride
- Memory Layout

Kernel Characteristics
- Operation (ex. Conv)
- Implementation (gemm vs fft)

Static Code Features
- Instruction Counts
- Loads
- Stores
- Instruction Types
- Arithmetic Intensity
- No. of Thread Blocks
- No. of Warps

Compile Time Hardware Features

PyTorch @ Harvard SEAS
Project 1: GPU Kernel Time Prediction using Machine Learning

What could be a potential feature set?

CUDA Kernel

Inputs: \( W_i, Z_i \)

Function: GEMM

Type: float_32

Launch

Kernel 1 - GEMM

\[ Z_{i+1} = W_i \times Z_i \]

Input Parameters

- Dimensions
- Data Type
- Stride
- Memory Layout

Kernel Characteristics

- Operation (ex. Conv)
- Implementation (gemm vs fft)

Static Code Features

- Instruction Counts
- Loads
- Stores
- Instruction Types
- Arithmetic Intensity
- No. of Thread Blocks
- No. of Warps

Compile Time Hardware Features

- No. of Streaming Multiprocessors
- Cache Size
- Shared memory size
- No. of Tensor Cores
- max FLOPS

---

PyTorch

\( \text{DASlab} \) @ Harvard SEAS
Project 1: GPU Kernel Time Prediction using Machine Learning

What is the goal of the project?

- Input Parameters
- Kernel Characteristics
- Static Code Features
- Compile Time
- Hardware Features
Project 1: GPU Kernel Time Prediction using Machine Learning

What is the goal of the project?

Input Parameters

Kernel Characteristics

Static Code Features

Compile Time
Hardware Features

AI Model
Project 1: GPU Kernel Time Prediction using Machine Learning

What is the goal of the project?

- Input Parameters
- Kernel Characteristics
- Static Code Features
- Compile Time
  - Hardware Features

→ AI Model
Project 1: GPU Kernel Time Prediction using Machine Learning

What is the goal of the project?

Input Parameters
Kernel Characteristics
Static Code Features
Compile Time
Hardware Features
AI Model
Latency (ms)
Project 1: GPU Kernel Time Prediction using Machine Learning

Roadmap for the project
Project 1: GPU Kernel Time Prediction using Machine Learning

Roadmap for the project

- GPU Fundamentals, FASRC
- Cluster Setup, Literature Review

[2 weeks]
Project 1: GPU Kernel Time Prediction using Machine Learning

Roadmap for the project

- GPU Fundamentals, FASRC Cluster Setup, Literature Review
  - [2 weeks]
- Benchmark Analysis
  - [1 week]
Project 1: GPU Kernel Time Prediction using Machine Learning

Roadmap for the project

- GPU Fundamentals, FASRC Cluster Setup, Literature Review
  - [2 weeks]

- Benchmark Analysis
  - [1 week]

- Experimental Design and Feature Engineering
  - [2 weeks]
Project 1: GPU Kernel Time Prediction using Machine Learning

Roadmap for the project

- GPU Fundamentals, FASRC Cluster Setup, Literature Review [2 weeks]
- Benchmark Analysis [1 week]
- Experimental Design and Feature Engineering [2 weeks]
- Benchmarking Experiments and data collection [2 weeks]
Project 1: GPU Kernel Time Prediction using Machine Learning

Roadmap for the project

- **GPU Fundamentals, FASRC Cluster Setup, Literature Review**
  - [2 weeks]

- **Benchmark Analysis**
  - [1 week]

- **Experimental Design and Feature Engineering**
  - [2 weeks]

- **Benchmarking Experiments and data collection**
  - [2 weeks]

- **Data preprocessing and ML model fitting**
  - [2 weeks]
Project 1: GPU Kernel Time Prediction using Machine Learning

Roadmap for the project

- GPU Fundamentals, FASRC Cluster Setup, Literature Review [2 weeks]
- Benchmark Analysis [1 week]
- Experimental Design and Feature Engineering [2 weeks]
- Benchmarking Experiments and data collection [2 weeks]
- Data preprocessing and ML model fitting [2 weeks]
- Performance Evaluation [1 week]
Project 1: GPU Kernel Time Prediction using Machine Learning

Skills to succeed

Rigorous Systems Course
CS 165/161

Machine Learning
CS 181/182/183/184/109A/109B

Computer Architecture
CS 141/146/246/247/249
Project 2: GPU Communication Collective Latency modeling

Accurate Communication Time

Nvidia Communication Collectives Library (NCCL)

Pronounced ‘Nickel’
Project 2: GPU Communication Collective Latency modeling

Nvidia Communication Collectives Library (NCCL)

Collective Operations
Project 2: GPU Communication Collective Latency modeling

Nvidia Communication Collectives Library (NCCL)

Collective Operations

All Reduce

\[ \text{out}[i] = \text{sum}(\text{in}[i]) \]
Project 2: GPU Communication Collective Latency modeling

Nvidia Communication Collectives Library (NCCL)

Collective Operations

All Reduce

All Gather
Project 2: GPU Communication Collective Latency modeling

Nvidia Communication Collectives Library (NCCL)

Collective Operations

All Reduce

All Gather

Reduce
Project 2: GPU Communication Collective Latency modeling

Nvidia Communication Collectives Library (NCCL)

Collective Operations

All Reduce

All Gather

Reduce

Broadcast
Project 2: GPU Communication Collective Latency modeling

Nvidia Communication Collectives Library (NCCL)

Collective Operations

All Reduce

All Gather

Reduce

Broadcast

Reduce Scatter
Project 2: GPU Communication Collective Latency modeling

Implementation Algorithm

All Reduce

\[ \text{out}(j) = \sum \text{in}(i)(j) \]
Project 2: GPU Communication Collective Latency modeling

Implementation Algorithm

All Reduce

\[ \text{out}[j] = \text{sum}(\text{in}[i]) \]

Ring All Reduce
Project 2: GPU Communication Collective Latency modeling

Implementation Algorithm

**All Reduce**

**Ring All Reduce**

**Tree All Reduce**
Project 2: GPU Communication Collective Latency modeling

Network Properties
Project 2: GPU Communication Collective Latency modeling

Network Properties

Topology: Physical device connection map
Project 2: GPU Communication Collective Latency modeling

Network Properties

**Topology:** Physical device connection map

**Adapters:** How the device is connected to the network?
Project 2: GPU Communication Collective Latency modeling

Network Properties

**Topology:** Physical device connection map

**Adapters:** How the device is connected to the network?

**Interconnects:** NVLinks, Infiniband, Ethernet, PCI-E
Project 2: GPU Communication Collective Latency modeling

Network Properties

**Topology:** Physical device connection map

**Adapters:** How the device is connected to the network?

**Interconnects:** NVLinks, Infiniband, Ethernet, PCI-E

**Parameters:** Latency and Bandwidth
Project 2: GPU Communication Collective Latency modeling

What is the goal of the project?
Project 2: GPU Communication Collective Latency modeling

What is the goal of the project?
Project 2: GPU Communication Collective Latency modeling

What is the goal of the project?

- Tensor Size
- Implementation Algorithm
- Network Properties
- Collective Operations

AI Model

Analytical Model
Project 2: GPU Communication Collective Latency modeling

What is the goal of the project?
Project 2: GPU Communication Collective Latency modeling

Roadmap for the project
Project 2: GPU Communication Collective Latency modeling

Roadmap for the project

NCCL Library and FASRC Cluster Setup, Literature Review

[2 weeks]
Project 2: GPU Communication Collective Latency modeling

Roadmap for the project

- NCCL Library and FASRC Cluster Setup, Literature Review [2 weeks]
- NCCL Cost Model Analysis for each collective [1-2 weeks]
Project 2: GPU Communication Collective Latency modeling

Roadmap for the project

NCCL Library and FASRC Cluster Setup, Literature Review
[2 weeks]

NCCL Cost Model Analysis for each collective
[1-2 weeks]

Analytically Modeling Communication collective cost
[2 weeks]
Project 2: GPU Communication Collective Latency modeling

Roadmap for the project

- NCCL Library and FASRC Cluster Setup, Literature Review [2 weeks]
- NCCL Cost Model Analysis for each collective [1-2 weeks]
- Analytically Modeling Communication collective cost [2 weeks]
- Benchmarking Experiments and data collection [2 weeks]
Project 2: GPU Communication Collective Latency modeling

Roadmap for the project

- NCCL Library and FASRC Cluster Setup, Literature Review [2 weeks]
- Analytically Modeling Communication collective cost [2 weeks]
- Data preprocessing and ML model fitting [1-2 weeks]
- NCCL Cost Model Analysis for each collective [1-2 weeks]
- Benchmarking Experiments and data collection [2 weeks]
Project 2: GPU Communication Collective Latency modeling

Roadmap for the project

- NCCL Library and FASRC Cluster Setup, Literature Review [2 weeks]
- NCCL Cost Model Analysis for each collective [1-2 weeks]
- Analytically Modeling Communication collective cost [2 weeks]
- Benchmarking Experiments and data collection [2 weeks]
- Data preprocessing and ML model fitting [1-2 weeks]
- Performance Evaluation [1 week]
Project 2: GPU Communication Collective Latency modeling

Skills to succeed

Rigorous Systems Course
CS 165/161

Computer Networks
CS 143/145/243

Machine Learning
CS 181/182/183/184/109A/109B

Algorithms/Modeling
CS 124/222/223/224
Alternatives and Existential Questions?

Why not profiling?

Why learn all the communication collectives?

Are there any more applications for these projects?
Exponential Increase in Model Size
Exponential Increase in Model Size

Turing NLG, Microsoft Research blog, 2020
Fully Sharded Data Parallel (FSDP)

Even the model weights do not fit on a single GPU
Fully Sharded Data Parallel (FSDP)

Model weights are sharded across GPUs
Fully Sharded Data Parallel (FSDP)

Broadcast first set of weights to all the GPUs
Fully Sharded Data Parallel (FSDP)

Send different micro-batches of data to different GPUs
Execute the first forward operation on different micro-batches.
Fully Sharded Data Parallel (FSDP)

All GPUs except the owner discards the local weight copy
Fully Sharded Data Parallel (FSDP)

Similarly all other activations are produced
Fully Sharded Data Parallel (FSDP)

First weight gradient is calculated in the reverse order.
Fully Sharded Data Parallel (FSDP)
Fully Sharded Data Parallel (FSDP)

Reduce the weight gradient on the owner
Alternatives and Existential Questions?

Why not profiling?
Why learn all the communication collectives?

VARYING

Type of network (EFA vs ENA)
Type of distributed strategy (DDP vs FSDP vs Hybrid)
Type of the GPUs (A100 vs V100 vs P100)
Number of GPUs (1 -> N)
Micro-batch Size (1 -> B)
Weight sharding scheme (Size of parameter shard per GPU)
More Applications?

sanketpurandare@g.harvard.edu

sanketpurandare@fas.harvard.edu
(Harvard University Slack)

SEC 4.435 (Monday & Friday)