ECHO: Compiler-based GPU Memory Footprint Reduction for LSTM RNN Training

Bojian Zheng\textsuperscript{1,2}, Nandita Vijaykumar\textsuperscript{1,3}, Gennady Pekhimenko\textsuperscript{1,2}
Executive Summary

• The GPU memory capacity limits the LSTM RNN training performance
  • Strategies: CPU Offloading, Data Encoding/Compression, Selective Recomputation

• **Echo** addresses 2 key challenges of selective recomputation:
  Estimation of ❶ memory footprint & ❷ runtime overhead

• Key Results: 3× footprint reduction with 1% overhead
  → Batch Size↑ 1.35× faster convergence to the same validation quality

• **Echo** and the MXNet GPU memory profiler are both open-sourced

Echo: https://issues.apache.org/jira/browse/MXNET-1450, GPU Memory Profiler: https://issues.apache.org/jira/browse/MXNET-1404
Background: DNN Training

1. Forward Pass

2. Backward Pass

3. Weight Update

\[ W = W - \alpha \frac{dE}{dW} \]
Background: Feature Maps

• Data entries that are stashed by the forward pass to compute the backward gradients

• The cause of high memory footprint in Convolutional Neural Networks (CNNs)\textsuperscript{[1, 2]}

\[ \text{Feature Maps} \]

\[ T_{\text{Total Memory Consumption}} \]

\[ T_{\text{Storage In-Use}} \]

1 2 3 4 \[ T \]

Large Temporal Gap between Usage

\[ \begin{align*}
1 & \rightarrow 2' \\
2' & \rightarrow 3' \\
3' & \rightarrow 4' \\
4' & \rightarrow n'
\end{align*} \]

\[ \begin{align*}
1 & \rightarrow 2 \\
2 & \rightarrow 3 \\
3 & \rightarrow 4 \\
n & \rightarrow T
\end{align*} \]

\textsuperscript{[1]} M. Rhu et al. vDNN: Virtualized Deep Neural Networks for Scalable, Memory-Efficient Neural Network Design. MICRO 2016

\textsuperscript{[2]} A. Jain et al. Gist: Efficient Data Encoding for Deep Neural Network Training. ISCA 2018
Background: LSTM RNN

• Long-Short-Term-Memory Recurrent Neural Network (LSTM RNN)

• Applications in machine translation (NMT) & speech recognition (DeepSpeech2)

• Its training is inefficient on the GPUs, especially when compared with CNN[1, 2]

[1] J. Bradbury et al. Quasi-Recurrent Neural Networks. ICLR 2016
[2] T. Lei et al. Simple Recurrent Units for Highly Parallelizable Recurrence. EMNLP 2018
Why LSTM RNN Training is Inefficient?

Training throughput **saturates** as batch size increases

**ResNet-50 (CNN)**

**NMT (LSTM RNN)**

- Training throughput is limited by the memory capacity
- Memory capacity limits the NMT training throughput

11 GB Memory Capacity
GPU Memory Profiling Results

Feature maps dominate the GPU memory footprint

https://issues.apache.org/jira/browse/MXNET-1404
Memory Capacity Limit: 3 Main Strategies

1. CPU Offloading (e.g., vDNN\cite{rhu2016v})
   - General
   - Intensive Use of Interconnect

2. Data Encoding/Compression (e.g., Gist\cite{jain2018gist})
   - Low Performance Overhead
   - Model/Layer-Specific

3. Selective Recomputation
   - General & Low Performance Overhead

---

\cite{rhu2016v}  M. Rhu et al. vDNN: Virtualized Deep Neural Networks for Scalable, Memory-Efficient Neural Network Design. MICRO 2016

\cite{jain2018gist}  A. Jain et al. Gist: Efficient Data Encoding for Deep Neural Network Training. ISCA 2018
Selective Recomputation

• **Key Idea:** Trade **runtime** with **memory capacity**

• The recomputation path should only involve **lightweight** operators

```
Feature Maps

Recomputation Path
```

```
Storage In-Use

1 2 3 4 – 3
```

```
Total Memory Consumption

T
```

```
Total Memory Consumption

T - Recomputation
```
## Prior Work on Selective Recomputation

| NMT | NO Recomputation | T. Chen et al.\[1]\ |
|-----|------------------|-------------------|
| Memory (GB) | 10.0 | 7.4 \(\downarrow 1.35\times\) |
| Throughput (samples/sec) | 1192 | 983 \(\downarrow 17\%\) |

Prior work **fails** to deliver satisfactory memory footprint reduction with acceptable overhead.

\[1\] T. Chen et al. *Training Deep Nets with Sublinear Memory Cost*. ArXiv e-prints 2016 #1604.06174
Prior Work on Selective Recomputation

Failure to address 2 key challenges:
1. Estimation of memory footprint
2. Runtime overhead
1 Memory Footprint Estimation

For each recomputation to be efficient, need to estimate its effect on the memory footprint.

Example: $Z = \tanh(X + Y)$
Memory Footprint Estimation

For each recomputation to be efficient, need to estimate its effect on the memory footprint.

Example: $Z_i = \tanh(X + Y_i), i \in [1, T]$

Global Memory Footprint Analysis:
1. shapes and data types
2. reuse Challenging!

(+) feature maps: $T^2N \rightarrow 2TN$
Runtime Overhead Estimation

For each recomputation to be efficient, need to estimate its effect on the runtime overhead.

Layer-Specific Property:
\[
\frac{dE}{dX} = \frac{dE}{dY} W \quad \text{&} \quad \frac{dE}{dW} = \frac{dE^T}{dY} X
\]
(NO Dependency on \(Y\))

Example: \(Y = XW^T\)

- **Compute-Heavy**
  - 50% of the NMT training time
- Excluded in prior works
**ECHO**: A Selective Recomputation Graph Compiler Pass

- Integrated in the MXNet NNVM\(^1\) module
- Fully **Automatic & Transparent**
  - Requires NO changes in the training source code
- Addresses the 2 key challenges: Estimation of
  1. memory footprint: *Bidirectional Dataflow Analysis*
  2. runtime overhead: *Layer Specific Optimizations*

---

\(^1\) [https://github.com/apache/incubator-mxnet/tree/master/src/nvml](https://github.com/apache/incubator-mxnet/tree/master/src/nvml)
**ECHO**: Bidirectional Dataflow Analysis

**Example**: $Z = \tanh(X + Y)$

- **Backward Pass**
  - Breaks at compute-heavy layers to **partition** the graph
  - Constructs a recomputation path that consists of nodes visited
**ECHO**: Bidirectional Dataflow Analysis

**Example:** $Z = \tanh(X + Y)$

- **Backward Pass**
  - Breaks at compute-heavy layers to partition the graph
  - Constructs a recomputation path that consists of nodes visited

- **Forward Pass**
  - Remove operator nodes from the recomputation path if $\text{sizeof(FeatureMaps}_{\text{new}}) \leq \text{sizeof(FeatureMaps}_{\text{old}})$
**Echo:** Bidirectional Dataflow Analysis

- **Storage Reuse**
  
  Causes ALL correlated operators to forward propagate simultaneously

  \[
  \text{sizeof} \left( \sum \text{FeatureMaps}_{\text{new}} \right) \leq \text{sizeof} \left( \sum \text{FeatureMaps}_{\text{old}} \right)
  \]

  \[T^2N \leq 2TN\]

  **Example:**
  \[Z_i = \tanh(X + Y_i), \ i \in [1, T]\]
Overview

Motivation
• Memory capacity limits training performance

Challenges
• Estimation of
  ❶ memory footprint &
  ❷ runtime overhead

ECHO
• Bidirectional Dataflow Analysis
• Layer-Specific Optimizations

Evaluation
• How ECHO performs on real DNN models?
Evaluation: Benchmarks

Sockeye\[^{[1]}\]

\[^{[1]}\] F. Hieber et al. Sockeye: A Toolkit for Neural Machine Translation. ArXiv e-prints 2017 #1712.05690

- State-of-the-Art Neural Machine Translation Toolkit under MXNet
- **Datasets:**
  - IWSLT’15 English-Vietnamese (*Small*)
  - WMT’16 English-German (*Large*)
- **Key Metrics:**
  - Training Throughput
  - GPU Memory Consumption
  - Training Time to Validation Accuracy (BLEU Score)
Evaluation: Infrastructure

Hardware

4× NVIDIA RTX 2080 Ti GPU
(Turing; 11 GB GDDR6 Memory)

Software

NVIDIA CUDA v10.0
cuDNN v7.6.3
mxnet v0.12.1
## Evaluation: Systems

| Baseline          | Baseline System without Selective Recomputation |
|-------------------|-----------------------------------------------|
| Mirror            | T. Chen et al. [1]                           |
| **Echo**          | Compiler-based Automatic and Transparent Optimizations |

[1] T. Chen et al. *Training Deep Nets with Sublinear Memory Cost*. ArXiv e-prints 2016 #1604.06174
**ECHO’s Effect on Memory and Performance**

Small Dataset, Single-GPU Experiment

- **Baseline** $B = 128$
- **Mirror** $B = 128$
- **Echo** $B = 128$
- **Echo** $B = 256$

2× Training Batch Size

- **Memory Consumption (GB)**
  - 11 GB Memory Capacity
  - $0.52 +$
  - $0.32 +$
  - $1.00 +$
- **Throughput (samples/s)**
  - $1.27 +$
  - $0.99 +$
  - $1.00 +$

**Better**

- **Reduction Ratio**
- **Overhead**

|   | Mirror | High | High |
|---|--------|------|------|

**DOUBLE WIN!**

23
Echo’s Effect on Training Convergence

Large Dataset, Multi-GPU Experiment, Same Number of Training Steps

Echo achieves:
+ Same Validation BLEU Score
+ Faster Convergence
+ Fewer Compute Devices
Other Results in the Paper

• More State-of-the-Art Models:
  • DeepSpeech2 ($1.56\times$), Transformer ($1.59\times$), ResNet-152 ($2.13\times$)

• More Benefits from Memory Footprint Reduction:
  • GPU energy consumption saving ($1.35\times$)
  • maximum number of layers with the same GPU memory budget ($2\times$)
Conclusion

• The GPU memory capacity limits the LSTM RNN training performance.
  • Major Strategy: Selective Recomputation

• **ECHO** addresses 2 key challenges of selective recomputation:
  Estimation of ❶ memory footprint & ❷ runtime overhead

• Key Results: $3 \times$ footprint reduction with $1\%$ overhead
  $\rightarrow$ Batch Size $\uparrow$ $1.35 \times$ faster convergence to the same validation quality

• **ECHO** and the MXNet GPU memory profiler are both open-sourced

  **ECHO**: [https://issues.apache.org/jira/browse/MXNET-1450](https://issues.apache.org/jira/browse/MXNET-1450),
  **GPU Memory Profiler**: [https://issues.apache.org/jira/browse/MXNET-1404](https://issues.apache.org/jira/browse/MXNET-1404)
ECHO: Compiler-based GPU Memory Footprint Reduction for LSTM RNN Training

Bojian Zheng¹,², Nandita Vijaykumar¹,³, Gennady Pekhimenko¹,²

ECHO: https://issues.apache.org/jira/browse/MXNET-1450, GPU Memory Profiler: https://issues.apache.org/jira/browse/MXNET-1404
ReLU vs. tanh/sigmoid Activation

• The tanh/sigmoid activation does **NOT** produce much zero sparsity.
**Echo**’s Effect on DeepSpeech2

- **Memory Consumption (GB)**
  - Batch Size: 8, 16, 24, 32
  - Baseline: 1.00x, 1.12x, 1.36x, 1.64x
  - Mirror: 1.04x, 1.24x, 1.72x, 2.14x
  - Echo: 1.41x

- **Throughput (samples/s)**
  - Batch Size: 8, 16, 24, 32
  - Baseline: 0.98x, 1.93x, 1.92x, 2.80x
  - Mirror: 2.66x, 2.77x
  - Echo: 3.63x

**Echo**’s benefits are across different models.
Echo vs. Hand-tuned

Large Dataset, Multi-GPU Experiment, Same Number of Training Steps

Hand-tuned Recomputation:
- Better Performance
- Model/Layer-Specific
**Echo**’s Effect on EC2 p3.8xlarge Instance

**Large Dataset, Multi-GPU Experiment, Same Number of Training Steps**

- **Baseline\(_{B=64}^{\text{Dev}=4}\)**
- **Mirror\(_{B=64}^{\text{Dev}=2}\)**
- **Echo\(_{B=128}^{\text{Dev}=1}\)**

**Memory Consumption (GB)**
- Baseline: 36
- Mirror: 12
- Echo: 12

**Throughput (samples/s)**
- Baseline: 450
- Mirror: 300
- Echo: 150

**Echo**’s benefits are across hardware platforms