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

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

¹ University of Toronto
² Vector Institute
³ Intel
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

Background: DNN Training

Forward Pass

\[ X \xrightarrow{\text{Deep Neural Network}} Y \]

\[ W \]

Backward Pass

\[ \frac{dE}{dX} \rightarrow \frac{dE}{dY} \rightarrow \frac{dE}{dW} \]

Weight Update

\[ W = W - \alpha \frac{dE}{dW} \]

\[ P[\text{Cool Dog}] = 100\% \]
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)[1, 2]

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]

Why LSTM RNN Training is Inefficient?

Training throughput **saturates** as batch size increases

Training throughput is limited by the **memory capacity**

**ResNet-50 (CNN)**

**NMT (LSTM RNN)**

11 GB Memory Capacity

**Memory capacity** limits the NMT training throughput
Feature maps dominate the GPU memory footprint.
Memory Capacity Limit: 3 Main Strategies

1. CPU Offloading (e.g., vDNN\textsuperscript{[1]})
   + General
   – Intensive Use of Interconnect

2. Data Encoding/Compression (e.g., Gist\textsuperscript{[2]})
   + Low Performance Overhead
   – Model/Layer-Specific

3. Selective Recomputation
   + General & Low Performance Overhead

---

\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
Selective Recomputation

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

  - The recomputation path should only involve *lightweight* operators

[Diagram showing feature maps and recomputation path]

- Total Memory Consumption: $T$
- Total Memory Consumption without Recomputation: $T - 3$

Storage In-Use:

```
1 2 3 4
```

Recomputation: $- 3$
## Prior Work on Selective Recomputation

<table>
<thead>
<tr>
<th></th>
<th>NO Recomputation</th>
<th>T. Chen et al.([1])</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory (GB)</td>
<td>10.0</td>
<td>7.4 (\downarrow) 1.35(\times)</td>
</tr>
<tr>
<td>Throughput (samples/sec)</td>
<td>1192</td>
<td>983 (\downarrow) 17%</td>
</tr>
</tbody>
</table>

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

---

Prior Work on Selective Recomputation

Failure to address 2 key challenges:
1. Estimation of memory footprint
2. Runtime overhead
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 ↑ $(N \rightarrow 2N)$ & (−) performance ↓ (recomputation)!
Memory Footprint Estimation

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

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

(+) feature maps: \( T^2N \rightarrow 2TN \)

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

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

Layer-Specific Property:
\[
\frac{dE}{dX} = \frac{dE}{dY} W \quad \text{and} \quad \frac{dE}{dW} = \frac{dE^T}{dY} X
\]
(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/nnvm
**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
**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 $\text{sizeof}(\text{FeatureMaps}_{\text{new}}) \leq \text{sizeof}(\text{FeatureMaps}_{\text{old}})$

**ECHO:** Bidirectional Dataflow Analysis

Practical & Accurate
**Echo**: Bidirectional Dataflow Analysis

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

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

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

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

- Memory capacity limits training performance
- Estimation of ❶ memory footprint & ❷ runtime overhead
- Bidirectional Dataflow Analysis
- Layer-Specific Optimizations
- How ECHO performs on real DNN models?
Evaluation: Benchmarks

Sockeye\textsuperscript{[1]}


• 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

<table>
<thead>
<tr>
<th></th>
<th>Baseline System without Selective Recomputation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Baseline</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Mirror</strong></td>
<td>T. Chen et al.(^1)</td>
</tr>
<tr>
<td><strong>ECHO</strong></td>
<td>Compiler-based Automatic and Transparent Optimizations</td>
</tr>
</tbody>
</table>

**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

- **11 GB Memory Capacity**
- **Better**

<table>
<thead>
<tr>
<th>Reduction Ratio</th>
<th>Overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mirror</td>
<td>High</td>
</tr>
<tr>
<td>High</td>
<td>High</td>
</tr>
</tbody>
</table>

**DOUBLE WIN!**
ECH0’s Effect on Training Convergence

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

ECH0 achieves:
+ Same Validation BLEU Score
+ Faster Convergence
+ Fewer Compute Devices
Other Results in the Paper

• **More State-of-the-Art Models:**
  • DeepSpeech2 (1.56×), Transformer (1.59×), ResNet-152 (2.13×)

• **More Benefits from Memory Footprint Reduction:**
  • GPU energy consumption saving (1.35×)
  • maximum number of layers with the same GPU memory budget (2×)
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× 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**: Compiler-based GPU Memory Footprint Reduction for LSTM RNN Training

Bojian Zheng$^{1,2}$, Nandita Vijaykumar$^{1,3}$, Gennady Pekhimenko$^{1,2}$

ReLU vs. tanh/sigmoid Activation

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

**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

ECHO’s benefits are across hardware platforms