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

Bojian Zheng[1,2], Nandita Vijaykumar[1,3], Gennady Pekhimenko[1,2]

1 Computer Science 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: $\mathbf{3}\times$ footprint reduction with $\mathbf{1}\%$ overhead
  $\rightarrow$ Batch Size↑ $\mathbf{1.35}\times$ 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, GPU Memory Profiler: https://issues.apache.org/jira/browse/MXNET-1404

Background: DNN Training

$X \Rightarrow$ **Deep Neural Network** $\Rightarrow Y$

W

$P[\text{Cool Dog}] = 100\%$

❶ Forward Pass

$X \Rightarrow$ **Deep Neural Network** $\Rightarrow Y$

W

❷ Backward Pass

$\frac{dE}{dX} \Leftarrow$ **Deep Neural Network** $\Leftarrow \frac{dE}{dY}$

$\frac{dE}{dW}$

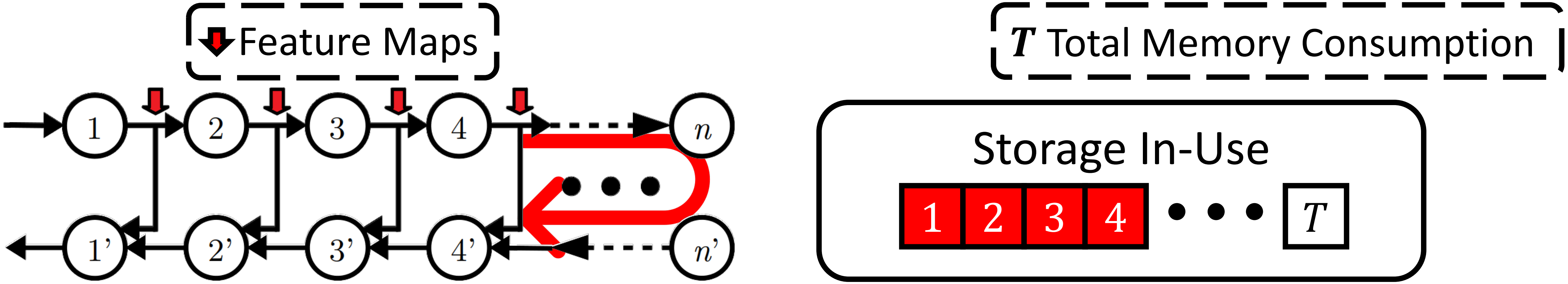
E Training Loss

❸ 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



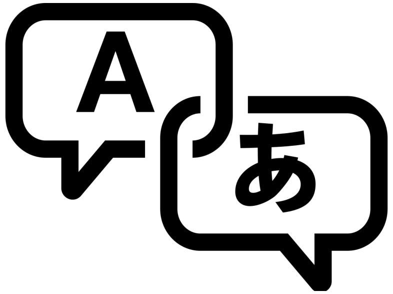
Large **Temporal Gap** between Usage

- The cause of high memory footprint in Convolutional Neural Networks (CNNs)[1, 2]

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

[2] A. Jain et al. *Gist: Efficient Data Encoding for Deep Neural Network Training*. ISCA 2018

Background: LSTM RNN

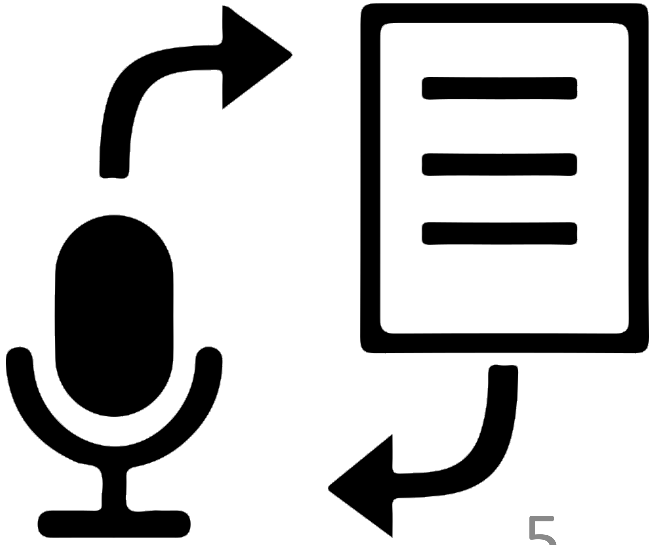


Neural Machine Translation (NMT)

- 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

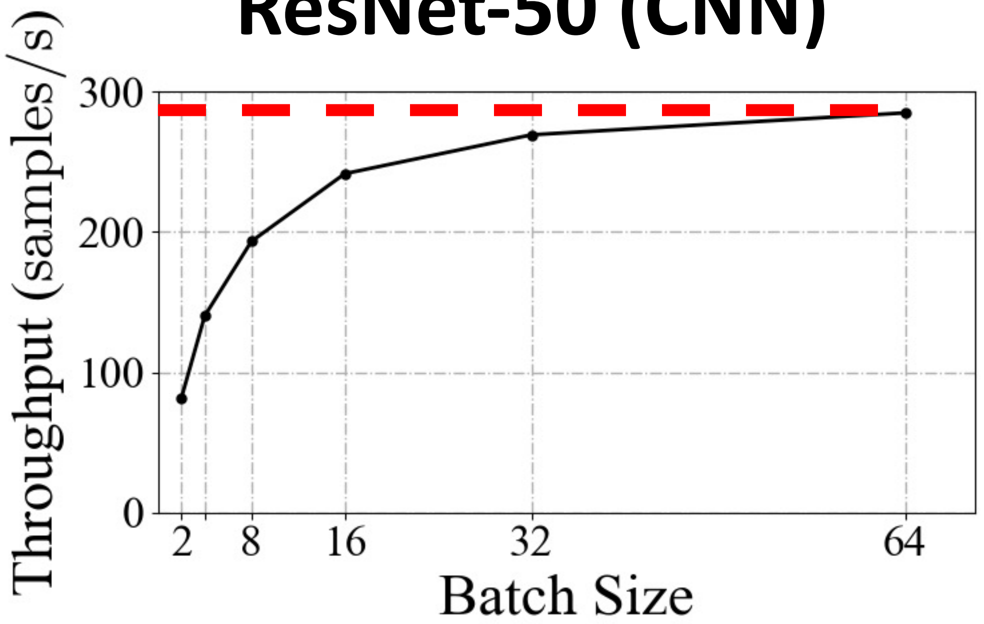
DeepSpeech2



Why LSTM RNN Training is Inefficient?

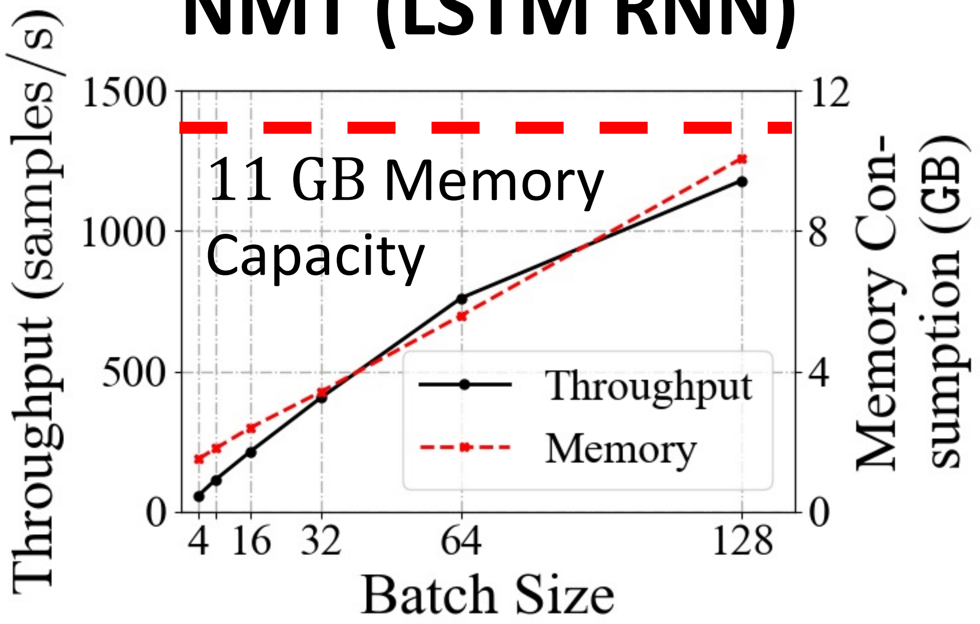
Training throughput **saturates** as batch size increases

ResNet-50 (CNN)



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

NMT (LSTM RNN)

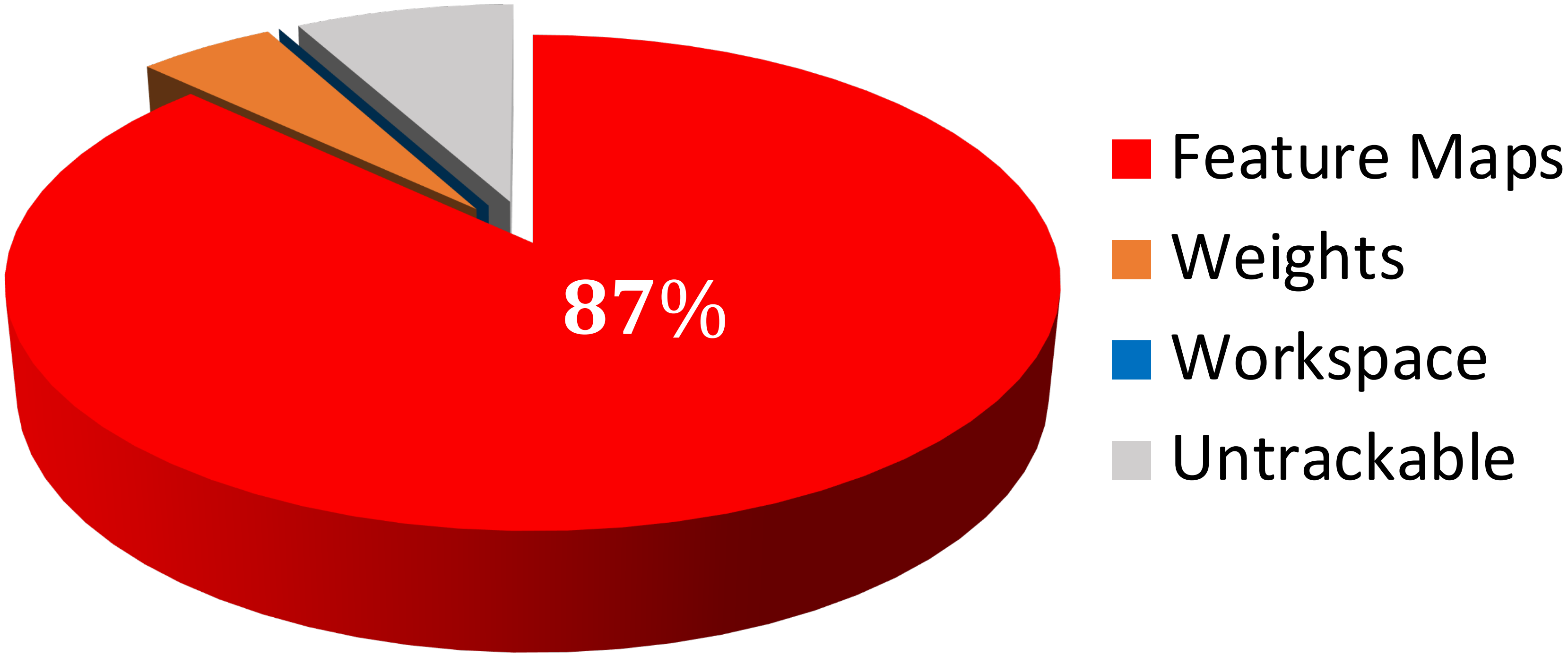


Memory capacity limits the NMT training throughput

GPU Memory Profiling Results

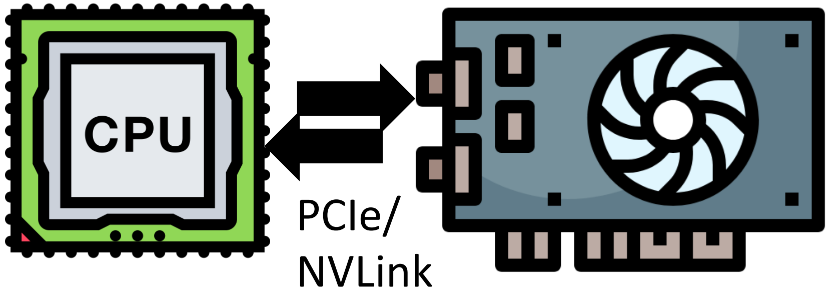
MXNet GPU Memory Profiler

https://issues.apache.org/jira/browse/MXNET-1404

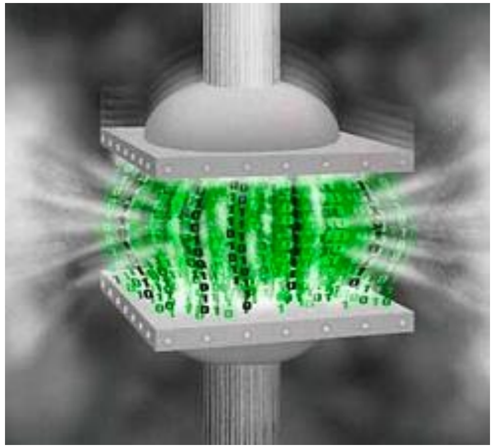


Feature maps dominate the GPU memory footprint

Memory Capacity Limit: 3 Main Strategies



1. CPU Offloading (e.g., vDNN[1])
   + General
   – Intensive Use of Interconnect
2. Data Encoding/Compression (e.g., Gist[2])
   + Low Performance Overhead
   – Model/Layer-Specific

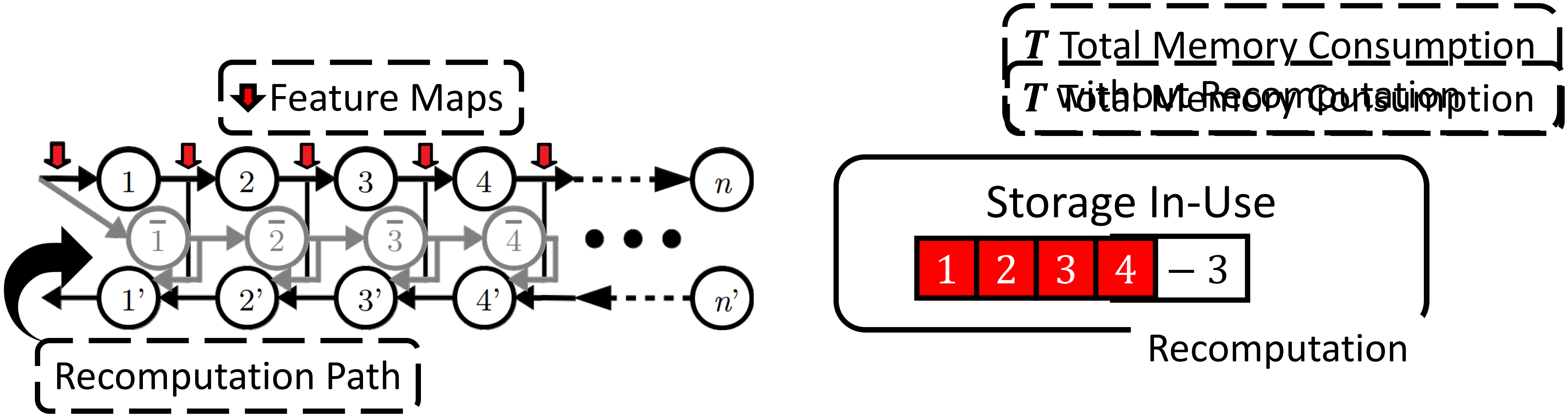


3. Selective Recomputation ✓
   + General & Low Performance Overhead

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

Prior Work on Selective Recomputation

NMT	**NO Recomputation**	**T. Chen et al.[1]**
Memory (GB)	10.0	7.4 ↓ 1.35×
Throughput (samples/sec)	1192	983 ↓ 17%

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

[1] T. Chen et al. *Training Deep Nets with Sublinear Memory Cost*. ArXiv e-prints 2016 #1604.06174

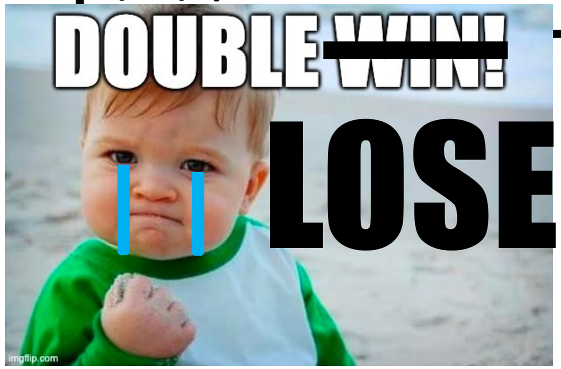
Prior Work on Selective Recomputation

Failure to address 2 key challenges:
Estimation of ❶ memory footprint &
❷ runtime overhead

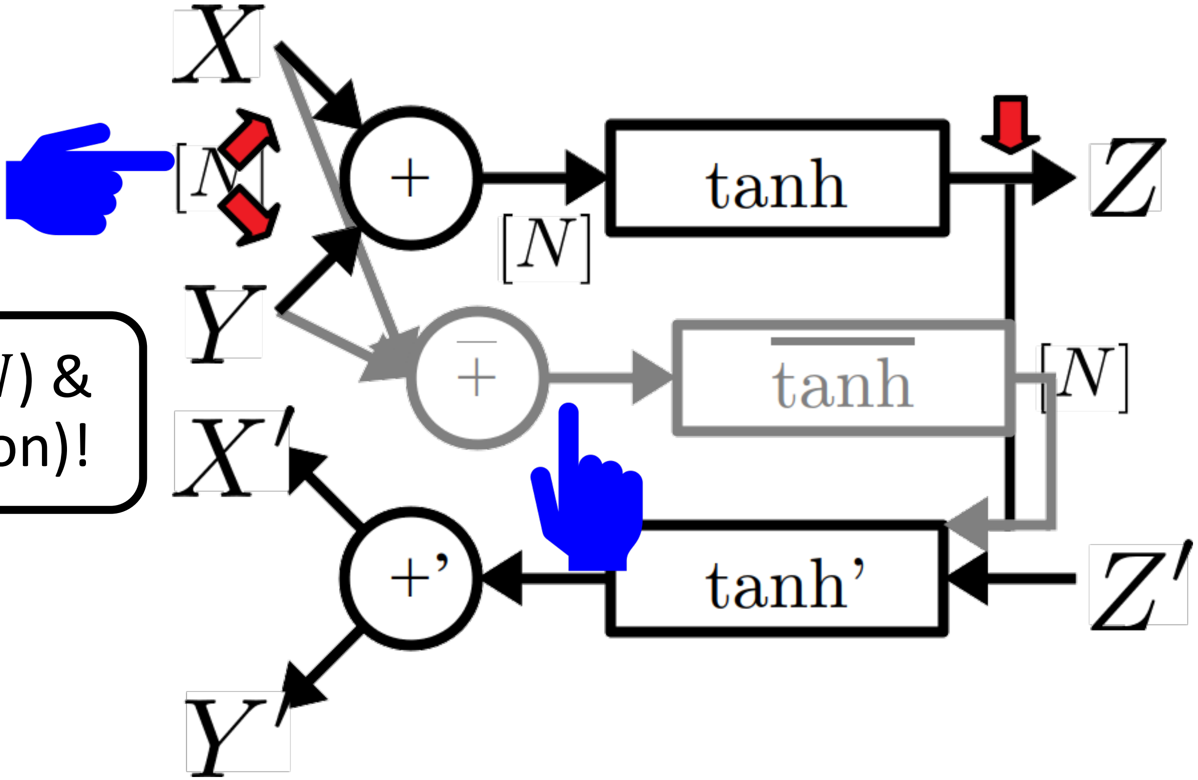
❶ Memory Footprint Estimation

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

(−) memory footprint ↑ ($N \rightarrow 2N$) &
(−) performance ↓ (recomputation)!

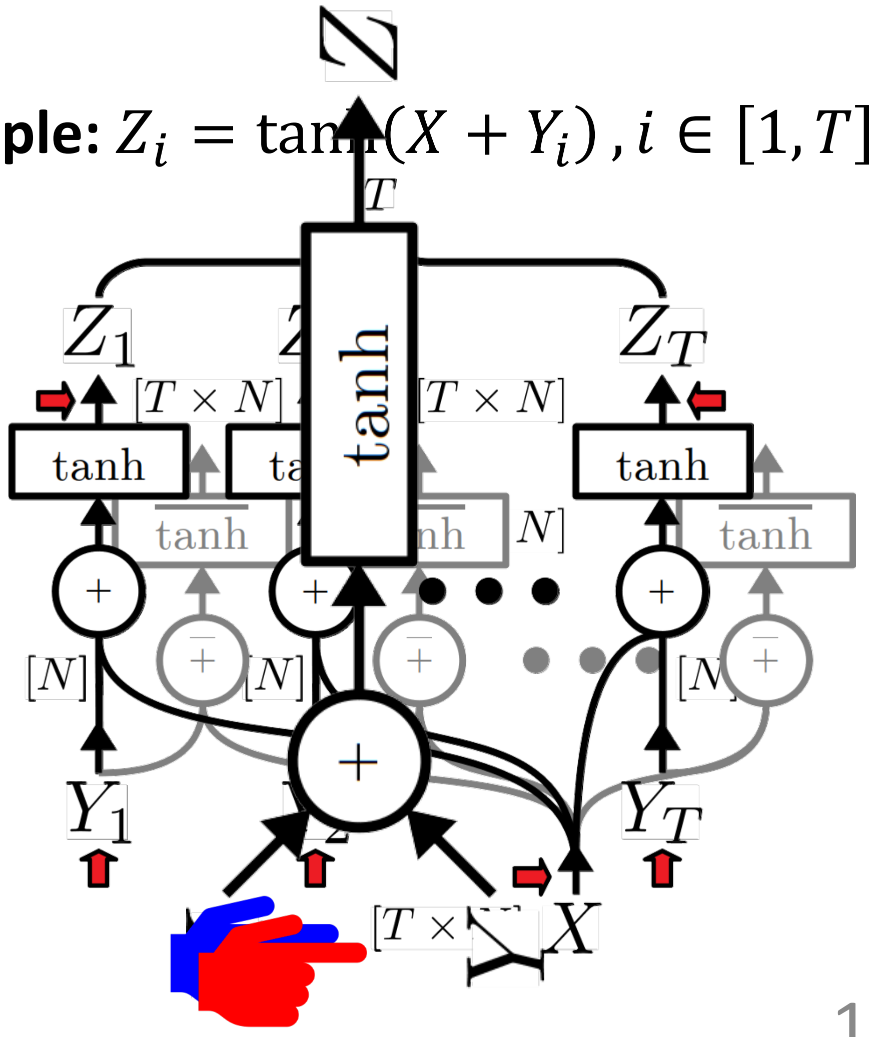


Example: $Z = \tanh(X + Y)$



❶ Memory Footprint Estimation

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



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

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

Global Memory Footprint Analysis:

1. shapes and data types
2. reuse **Challenging!**

❷ Runtime Overhead Estimation

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

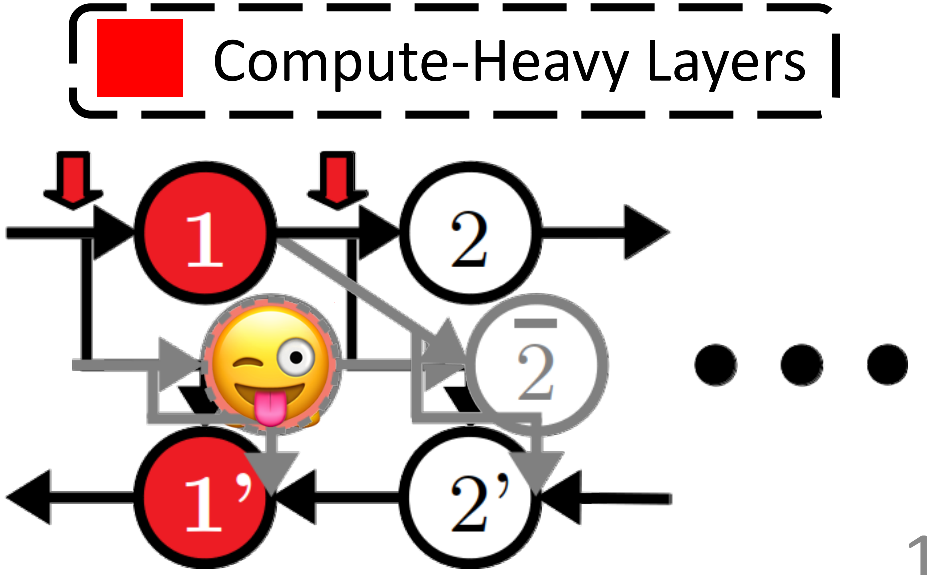
Layer-Specific Property:

$$\frac{dE}{dX} = \frac{dE}{dY} W \;\&\; \frac{dE}{dW} = \frac{dE^T}{dY} X$$

(NO Dependency on Y)

Example: $Y = XW^T$

- **Compute-Heavy**
  - 50% of the NMT training time
- Excluded in prior works



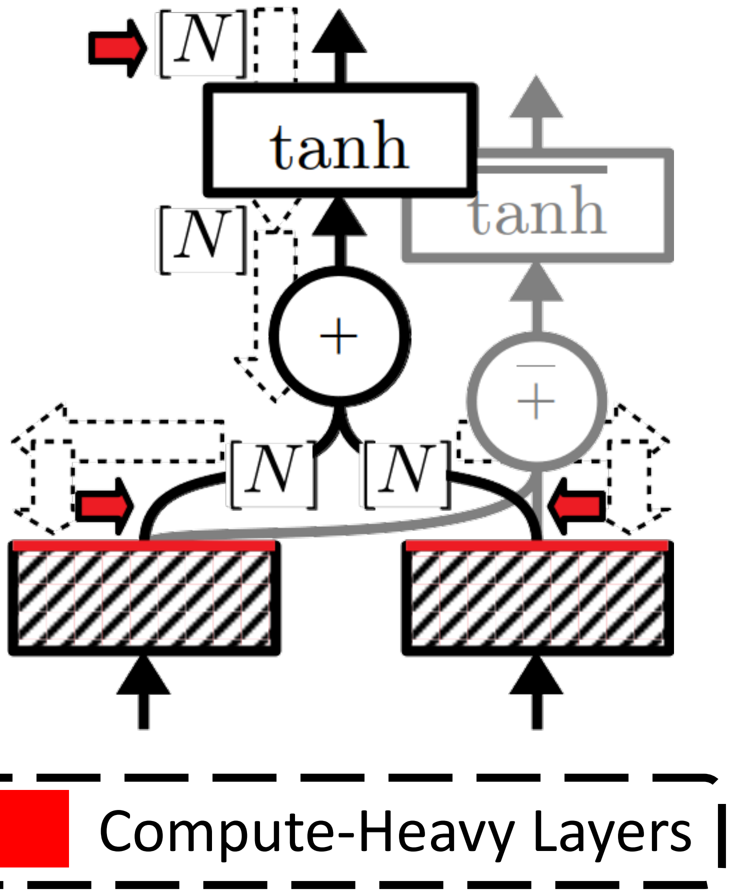
ECHO: A Selective Recomputation Graph Compiler Pass

- 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
  - ❶ memory footprint: *Bidirectional Dataflow Analysis*
  - ❷ runtime overhead: *Layer Specific Optimizations*

[1] https://github.com/apache/incubator-mxnet/tree/master/src/nnvm

ECHO: Bidirectional Dataflow Analysis

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



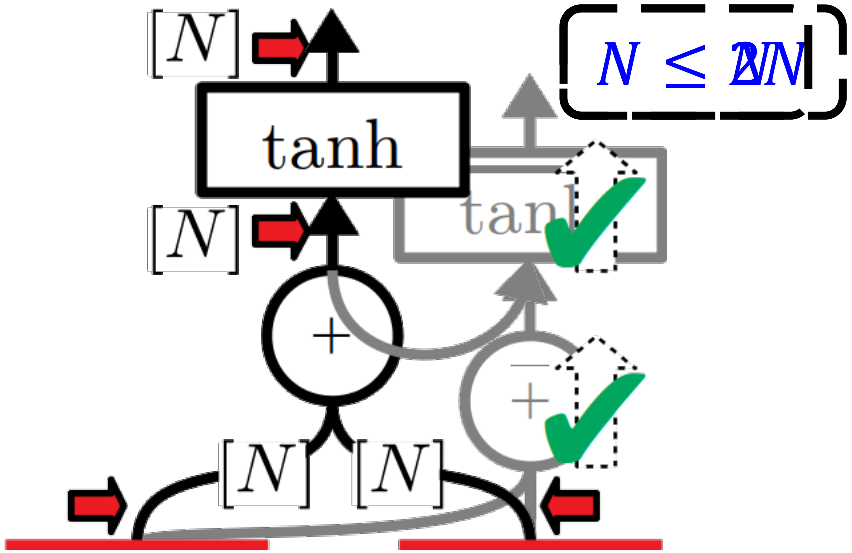
▼ **Backward Pass**

Breaks at compute-heavy layers to partition the graph

Constructs a recomputation path that consists of nodes visited

ECHO: Bidirectional Dataflow Analysis

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



▼ **Backward Pass**

Breaks at compute-heavy layers to partition the graph

Construct a recomputation path that consists of nodes visited

\+ **Practical & Accurate**

▲ **Forward Pass**

Remove operator nodes from the recomputation path if $\mathrm{sizeof}(\mathrm{FeatureMaps}_{\mathrm{new}}) \leq \mathrm{sizeof}(\mathrm{FeatureMaps}_{\mathrm{old}})$

ECHO: Bidirectional Dataflow Analysis

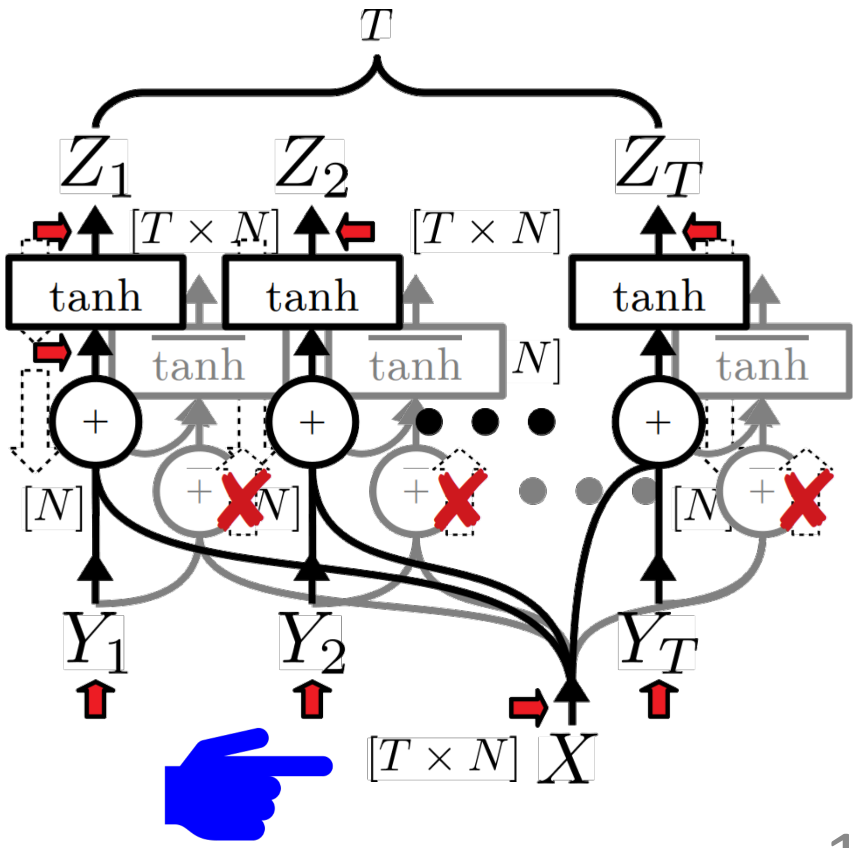
Example: $Z_i = \tanh(X + Y_i), i \in [1, T]$

- **Storage Reuse**

  Causes ALL correlated operators to forward propagate simultaneously

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

$$T^2N \nleq 2TN$$



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

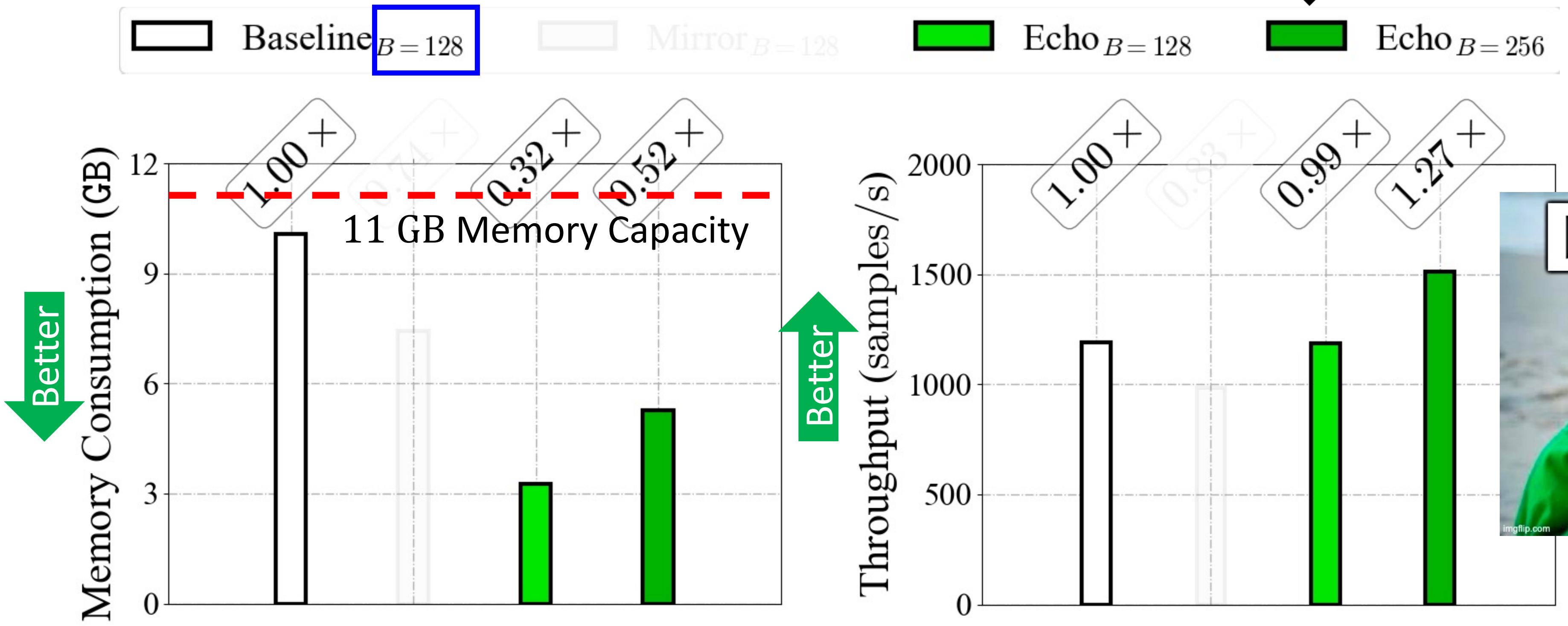
Baseline	Baseline System without Selective Recomputation
Mirror	T. Chen et al.[1]
ECHO	Compiler-based Automatic and Transparent Optimizations

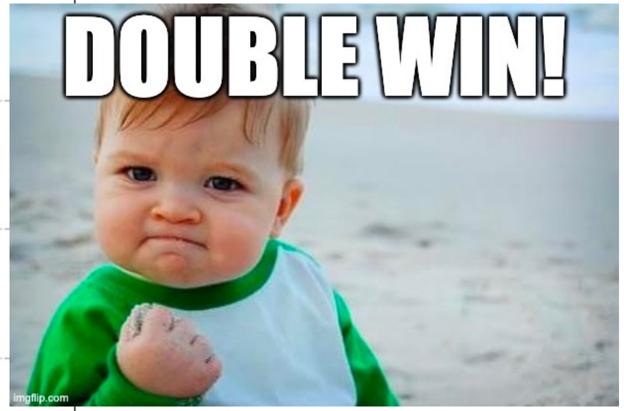
[1] T. Chen et al. *Training Deep Nets with Sublinear Memory Cost*. ArXiv e-prints 2016 #1604.06174

ECHO 's Effect on Memory and Performance

Small Dataset, Single-GPU Experiment

2× Training Batch Size

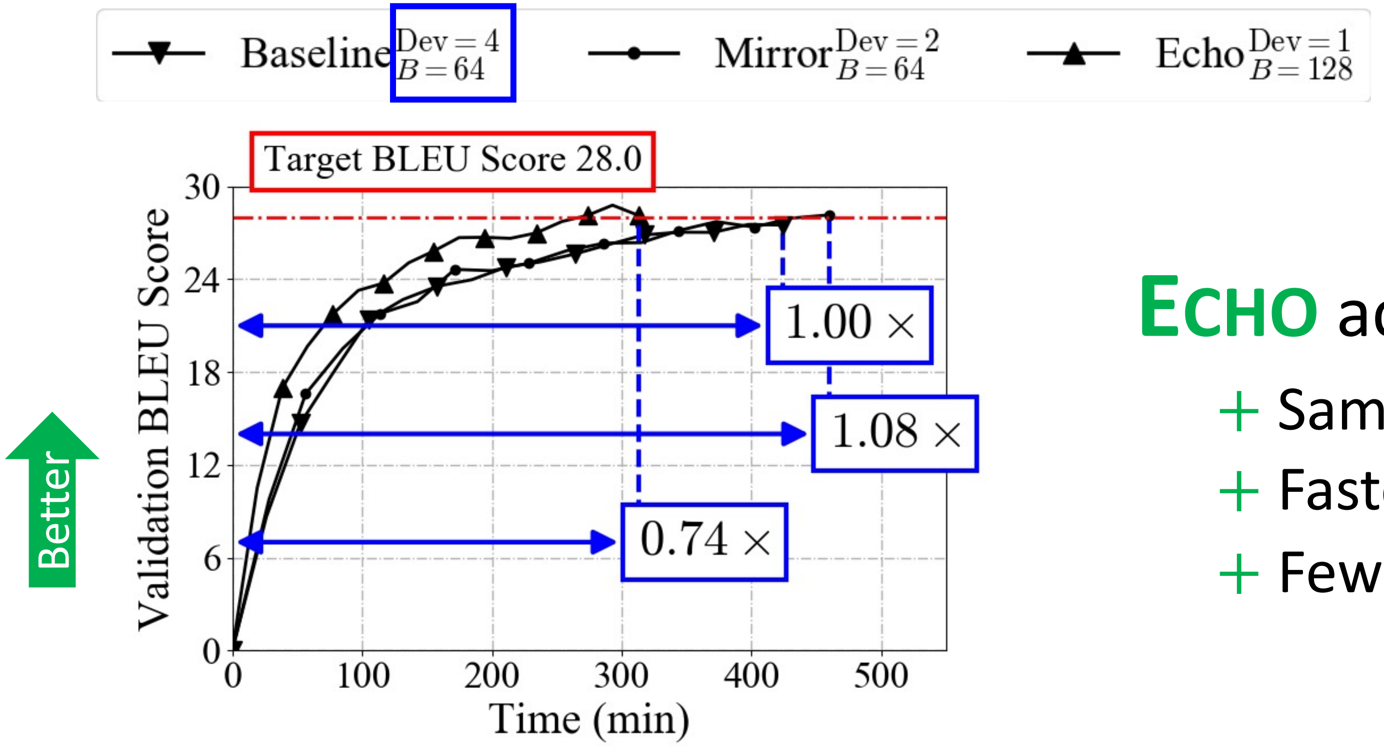




	Reduction Ratio	Overhead
Mirror Echo	High Low	High Low

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: $\mathbf{3}\times$ footprint reduction with $\mathbf{1}\%$ overhead $\rightarrow$ Batch Size↑ $\mathbf{1.35}\times$ 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, GPU Memory Profiler: https://issues.apache.org/jira/browse/MXNET-1404

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

Bojian Zheng[1,2], Nandita Vijaykumar[1,3], Gennady Pekhimenko[1,2]

1 Computer Science UNIVERSITY OF TORONTO

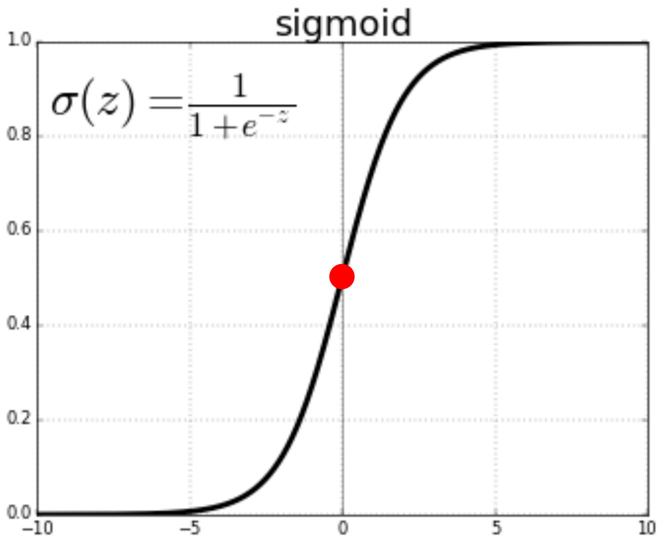
2 VECTOR INSTITUTE

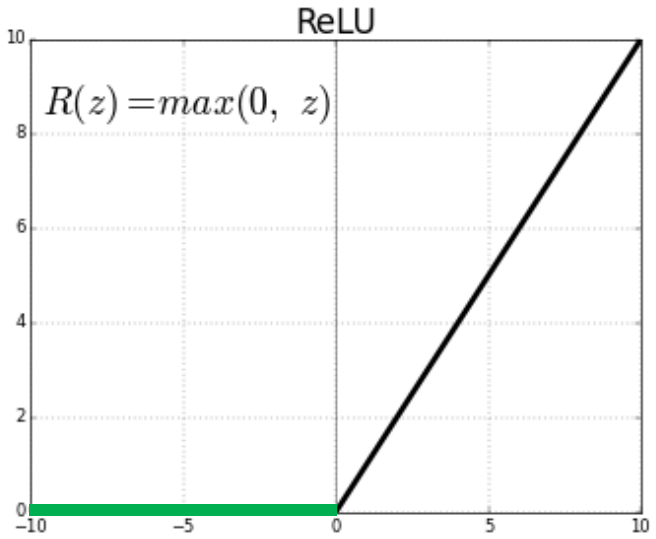
3 intel

ECHO: https://issues.apache.org/jira/browse/MXNET-1450, GPU Memory Profiler: https://issues.apache.org/jira/browse/MXNET-1404

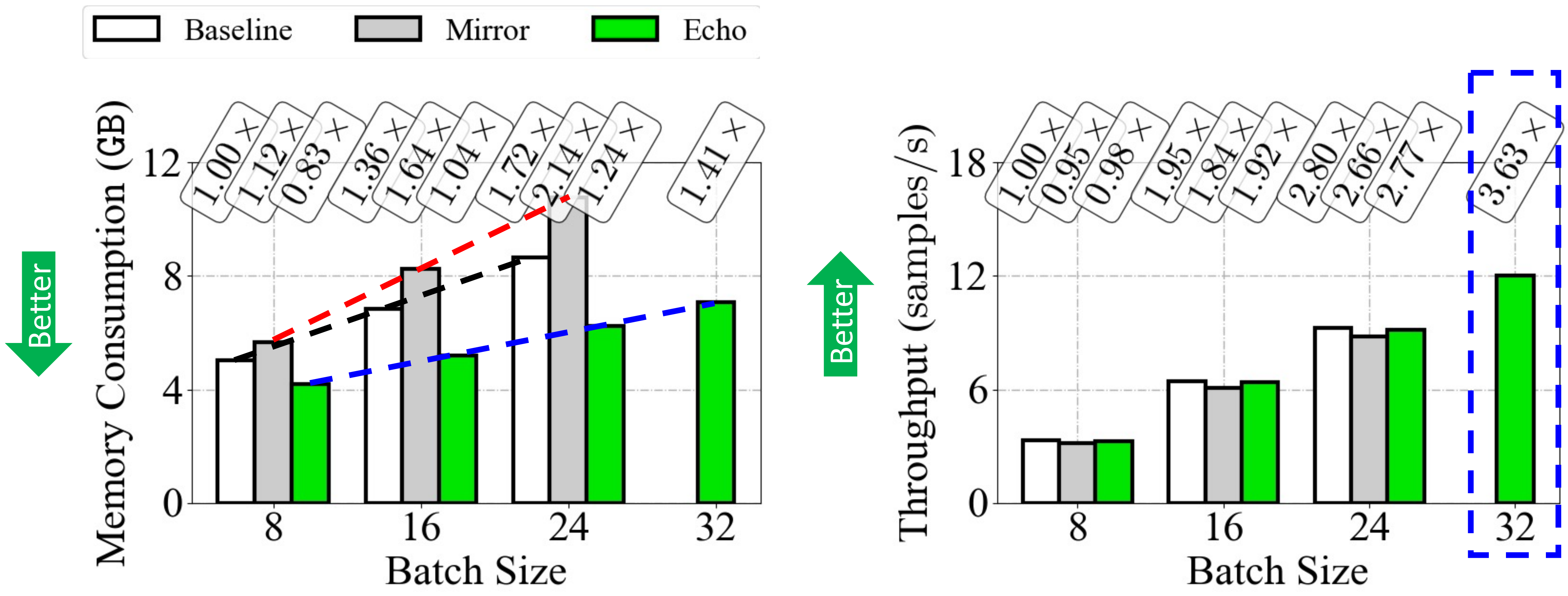
ReLU vs. tanh/sigmoid Activation

- The tanh/sigmoid activation does **NOT** produce much zero sparsity.





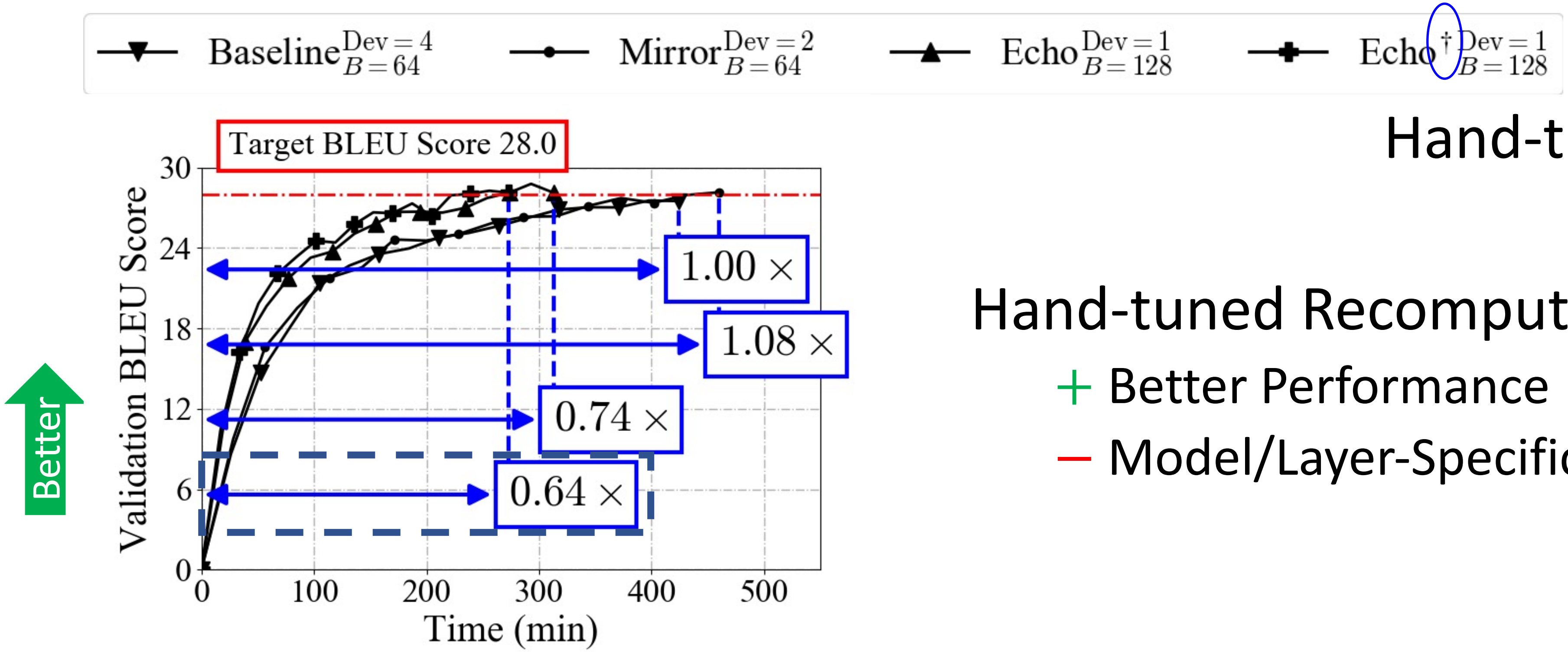
ECHO's Effect on DeepSpeech2



ECHO's benefits are across different models

ECHO vs. Hand-tuned

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



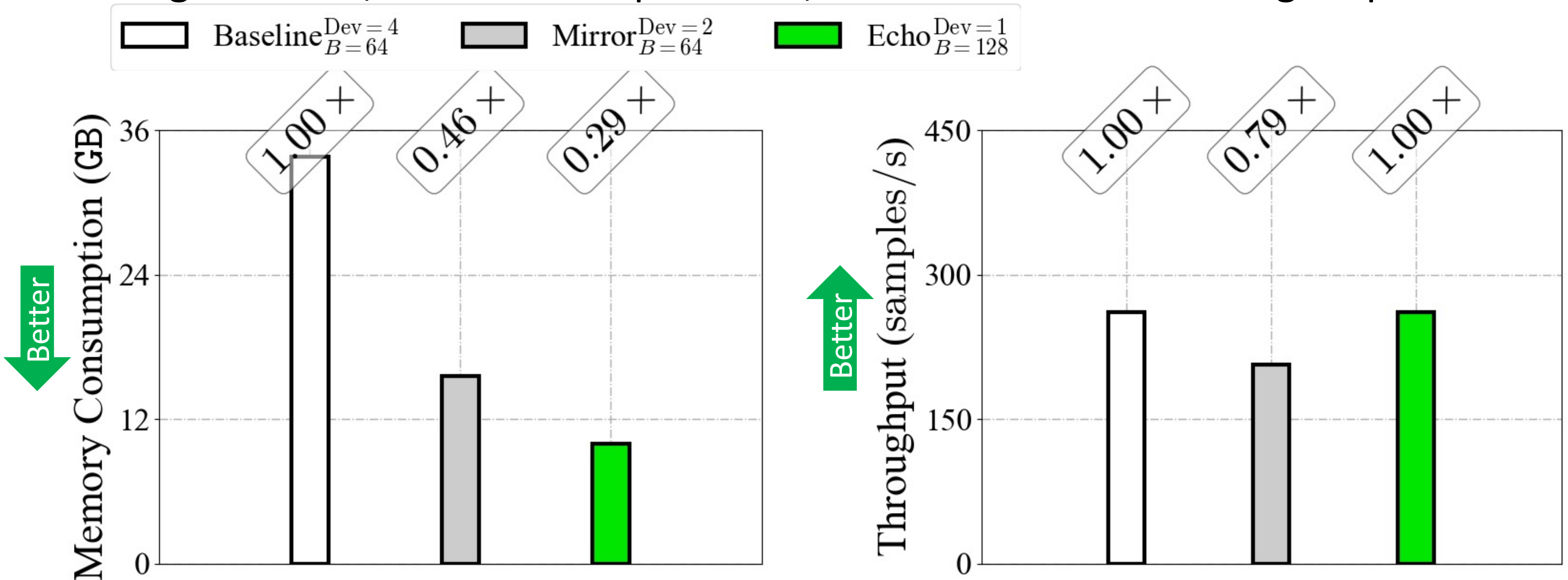
Hand-tuned

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