Memory Footprint Reduction Techniques for DNN Training: An Overview

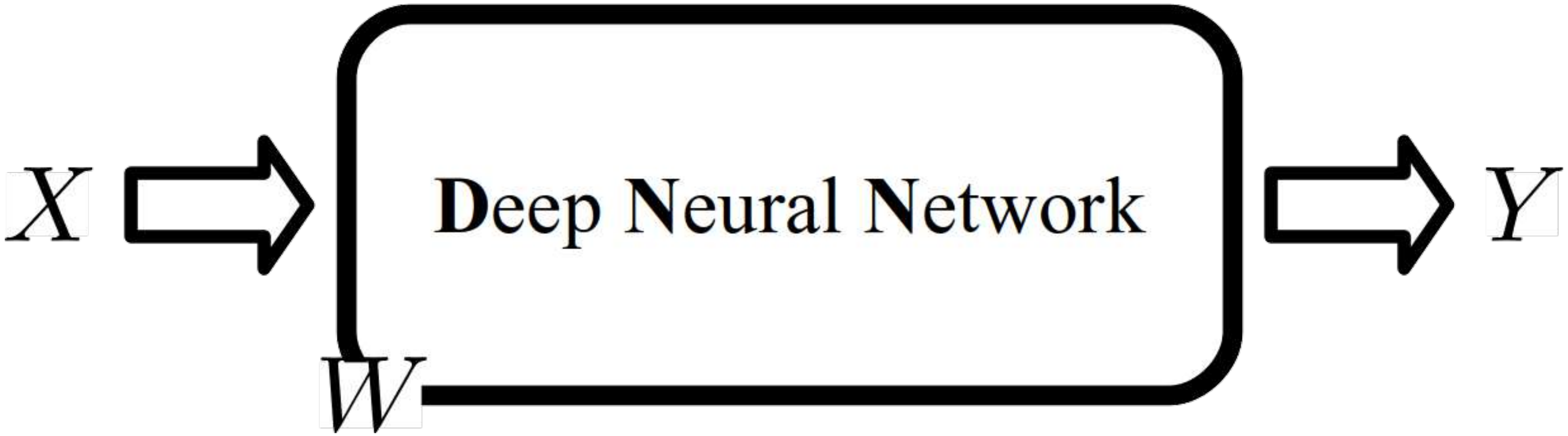
Gennady Pekhimenko, Assistant Professor

EcoSystem Group

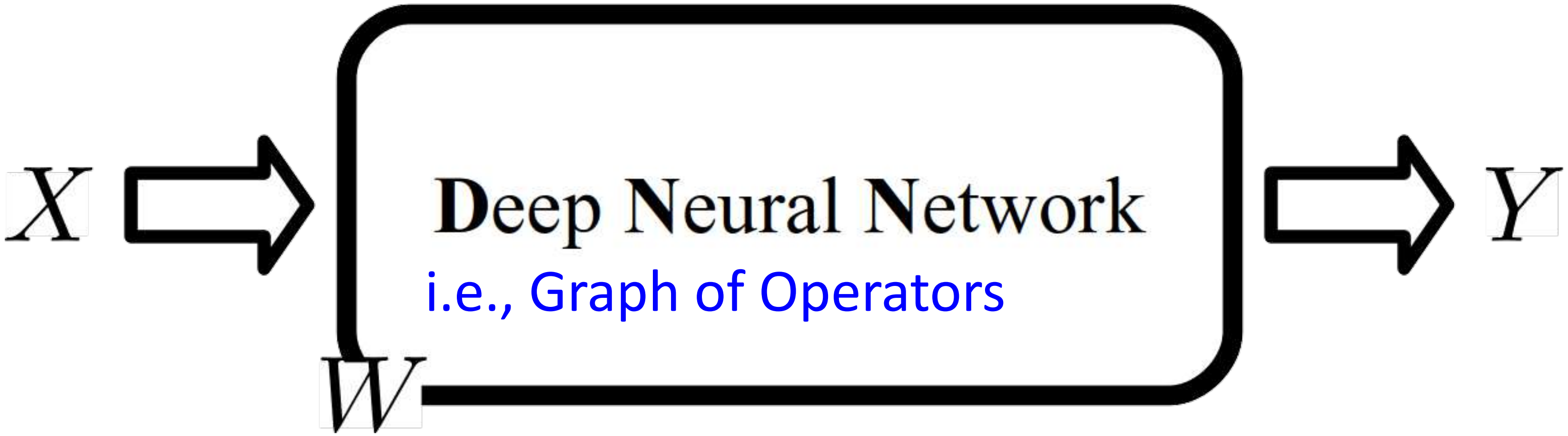
Outline

- Background on DNNs and Their Memory Allocations
- Why larger GPU memory?
- A History of Prior Works
- Prior Works on Feature Maps Reduction

DNN Training



DNN Training



DNN Training

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

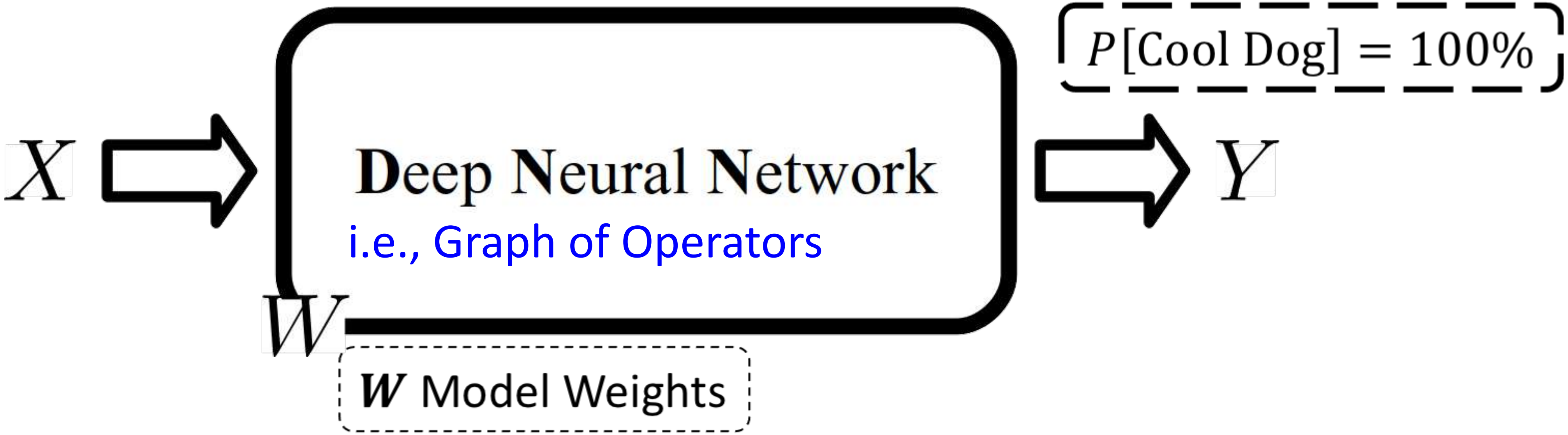
i.e., Graph of Operators

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

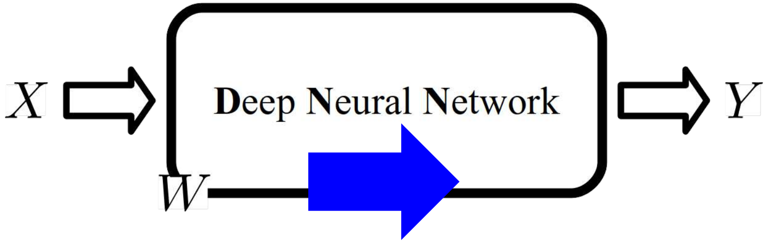
W

$\boldsymbol{W}$ Model Weights

DNN Training



❶ Forward Pass



DNN Training

$X \Rightarrow$ **Deep Neural Network** i.e., Graph of Operators $\Rightarrow Y$

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

W

W Model Weights

❶ Forward Pass

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

W

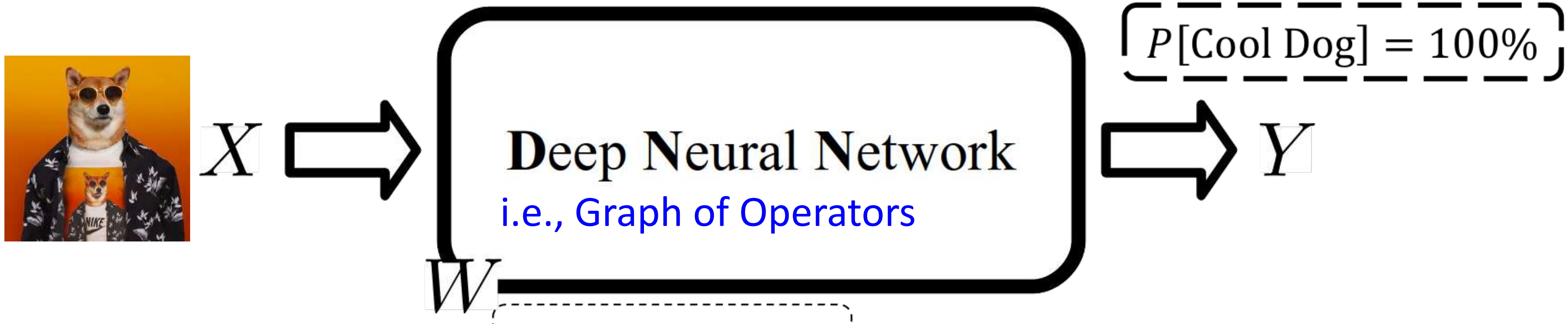
❷ Backward Pass

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

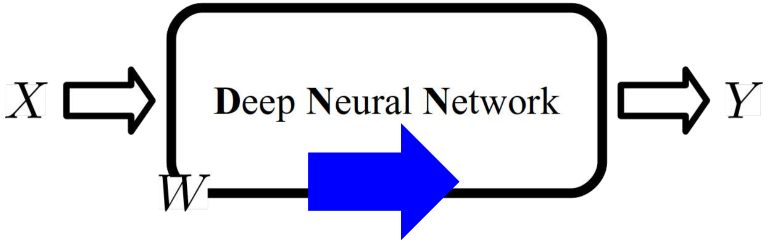
$\frac{dE}{dW}$

E Training Loss

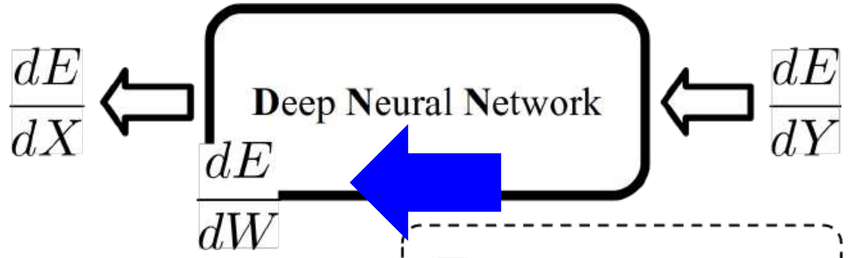
DNN Training



❶ Forward Pass



❷ Backward Pass



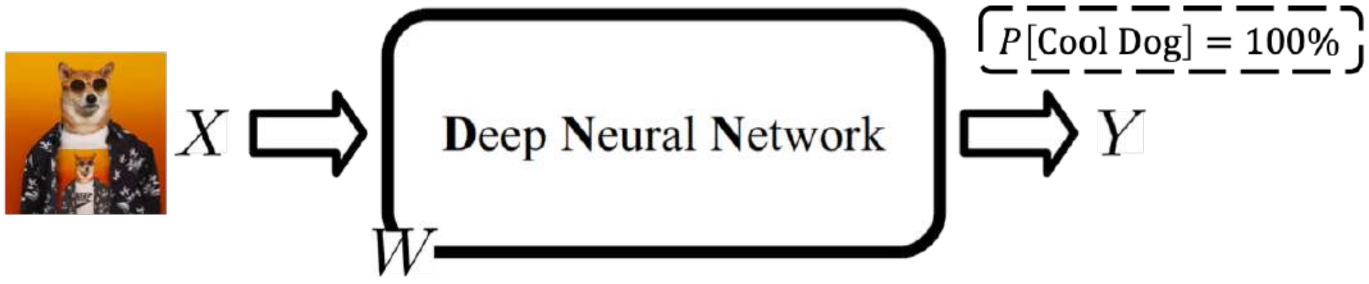
E Training Loss

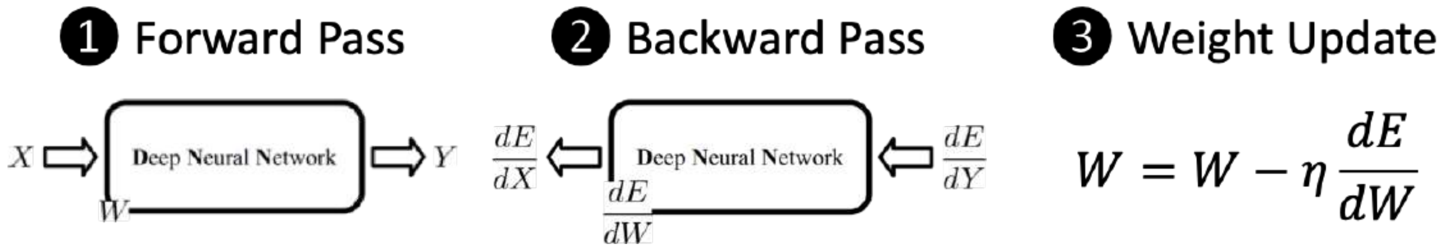
❸ Weight Update

$$W = W - \eta \frac{dE}{dW}$$

η Learning Rate

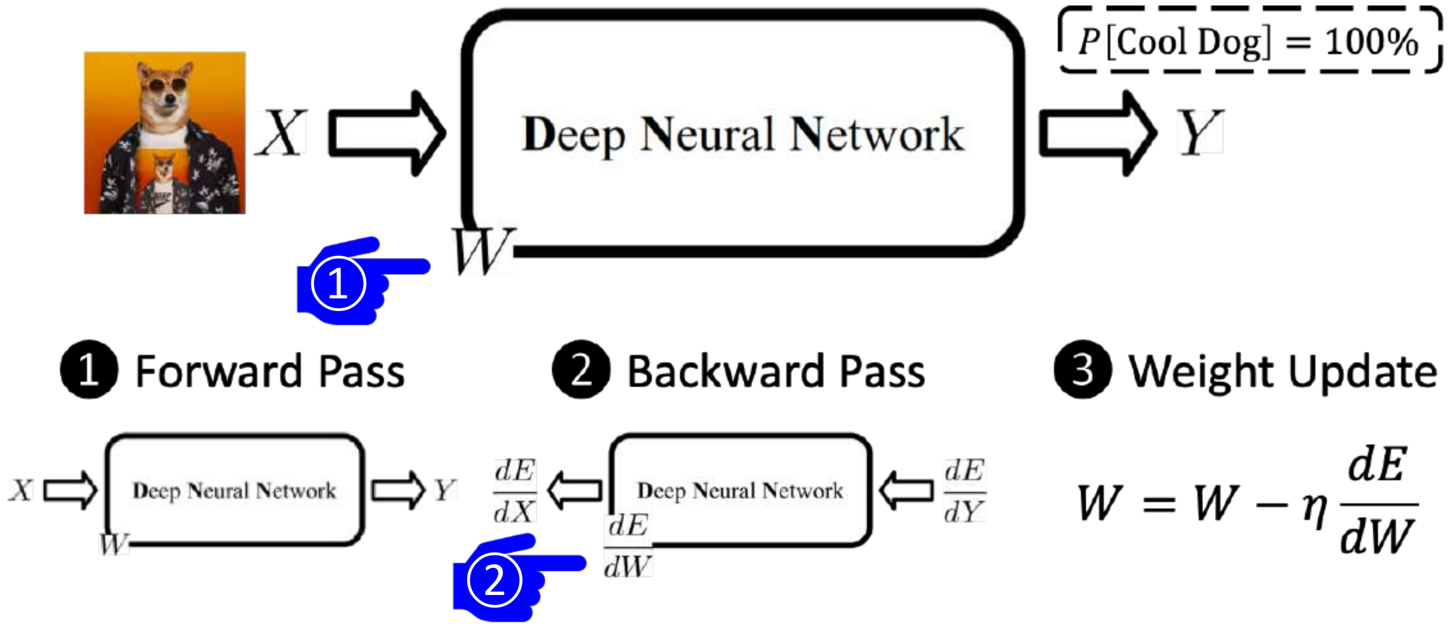
GPU Memory Allocations in DNN Training





- Major GPU memory consumers: **Weights** & **Feature Maps**

GPU Memory Allocations in DNN Training

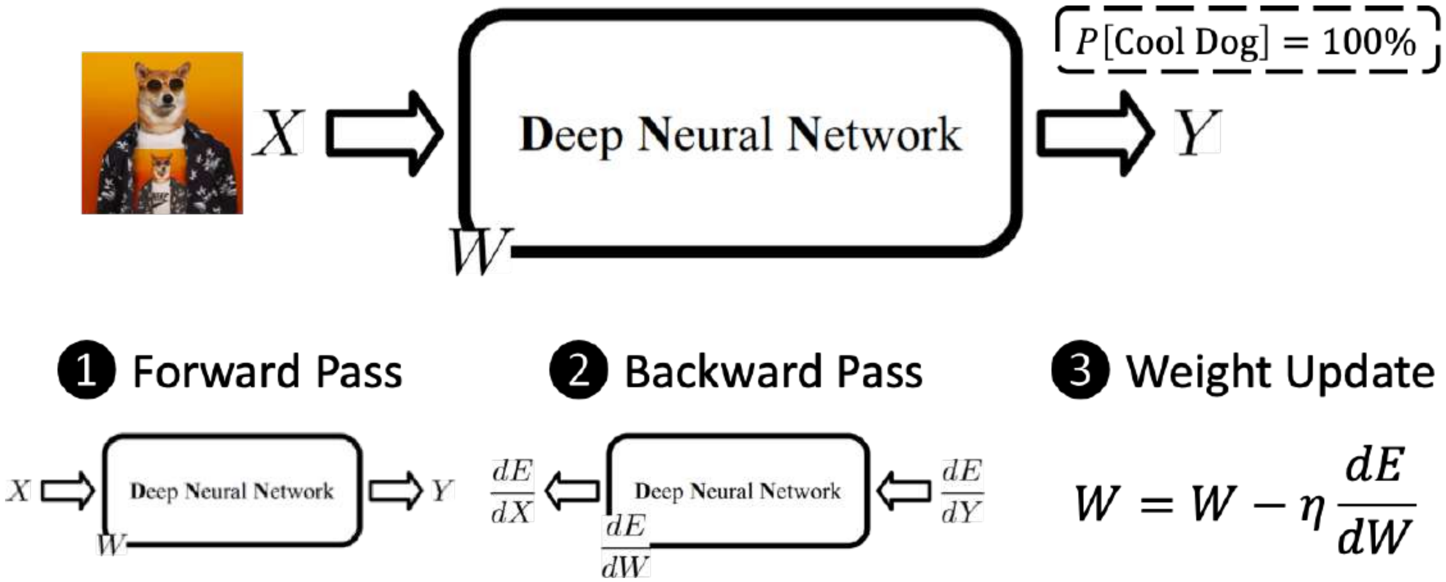


In state-of-the-art weight update optimizers (e.g., Adam[1]), need extra states ∀weight.

- Major GPU memory consumers: **Weights** & Feature Maps
  - **Weights** (General): weights (①), gradients (②), optimizer states (③)

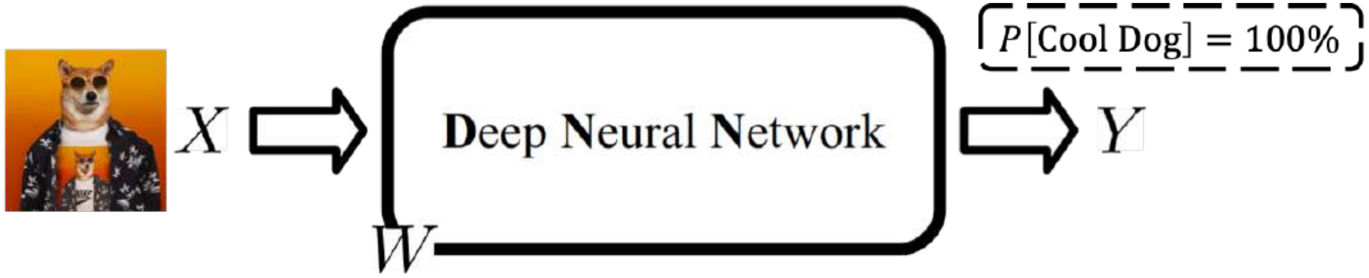
[1] D. Kingma, J. Ba. *Adam: A Method for Stochastic Optimization*. ICLR 2015

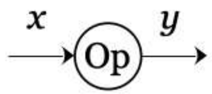
GPU Memory Allocations in DNN Training



- Major GPU memory consumers: Weights & **Feature Maps**
  - **Weights** (General): weights (①), gradients (②), optimizer states (③)
  - **Feature Maps**: Data entries stashed by the forward pass to compute the gradients in the backward pass.

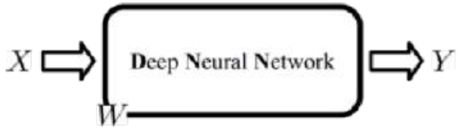
GPU Memory Allocations in DNN Training



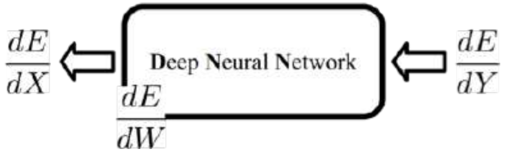


Operator $y = x + 1$, y could **reuse** the storage of x (i.e., in-place).

❶ Forward Pass



❷ Backward Pass

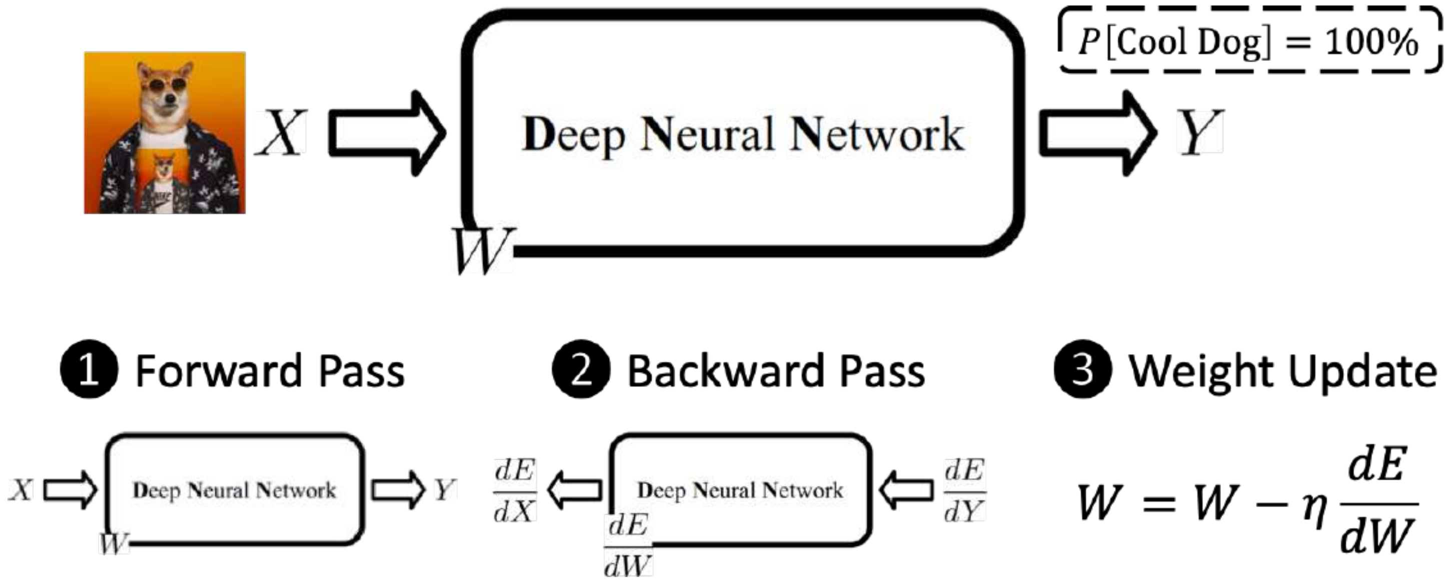


❸ Weight Update

$$W = W - \eta \frac{dE}{dW}$$

- Major GPU memory consumers: Weights & **Feature Maps**
  - **Weights** (General): weights (①), gradients (②), optimizer states (③)
  - **Feature Maps**: Data entries stashed by the forward pass to compute the gradients in the backward pass.

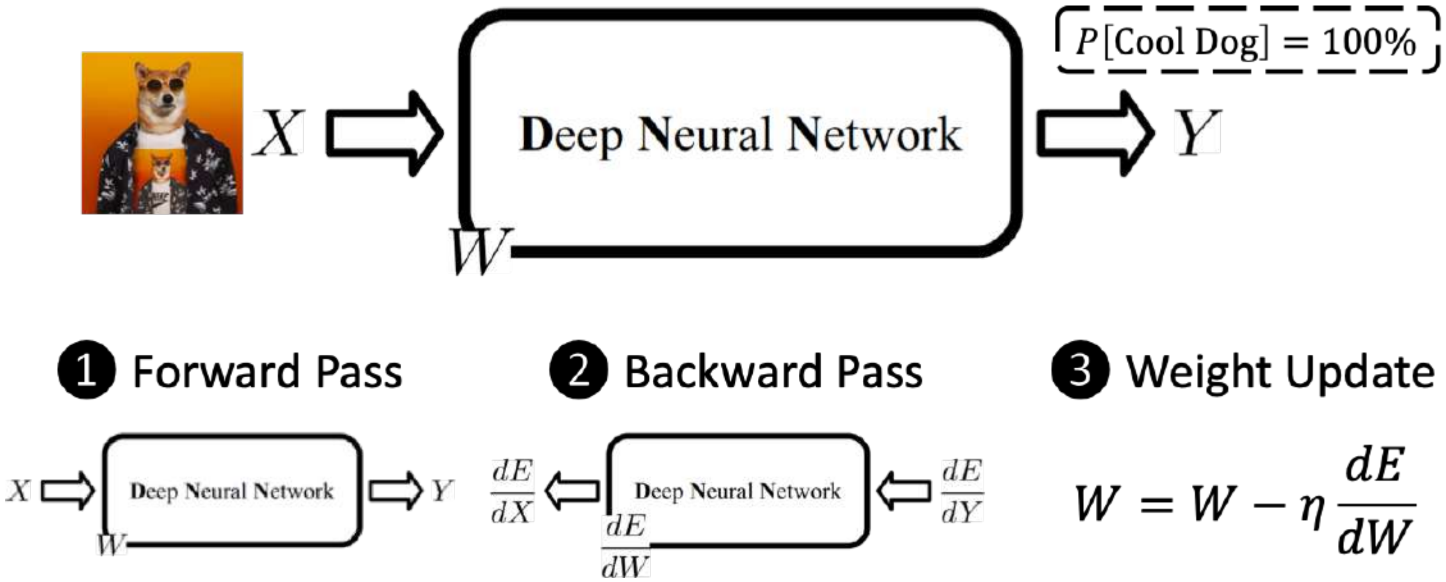
GPU Memory Allocations in DNN Training



Operator $y = \tanh x$, since $y' = 1 - y^2$, need to **stash** y for x'.

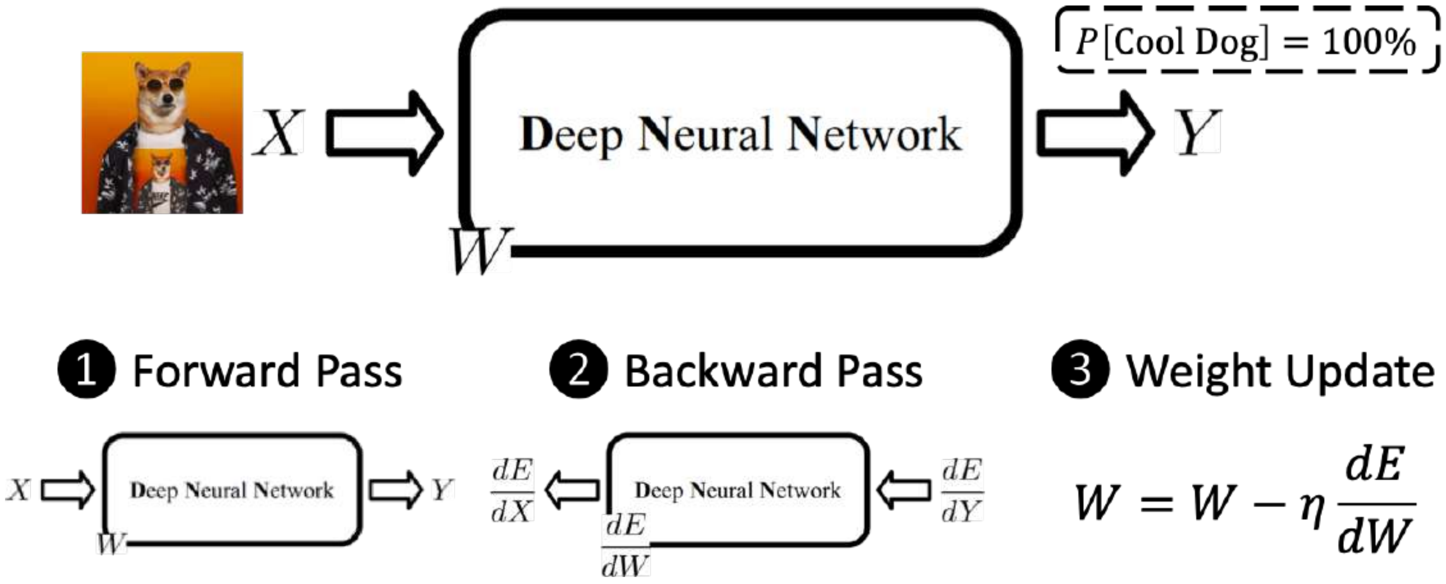
- Major GPU memory consumers: Weights & **Feature Maps**
  - **Weights** (General): weights (①), gradients (②), optimizer states (③)
  - **Feature Maps**: Data entries stashed by the forward pass to compute the gradients in the backward pass.

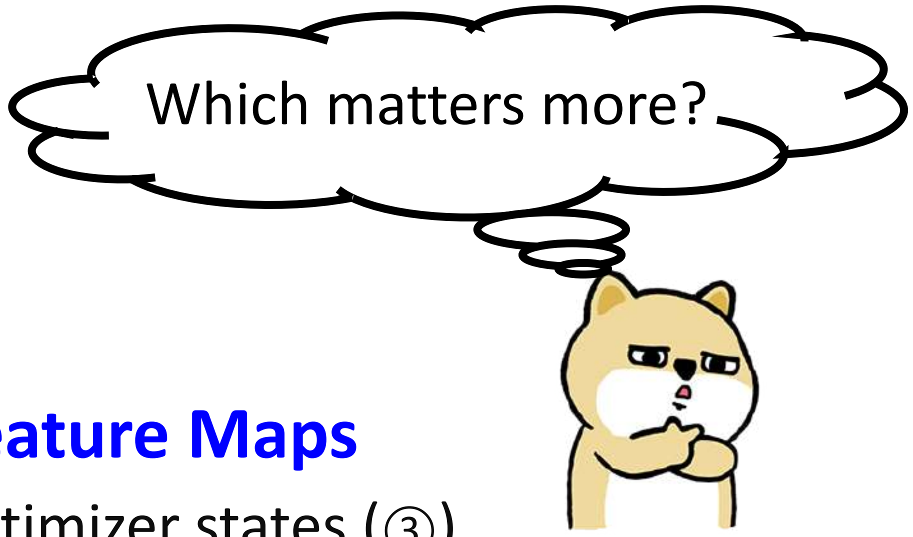
GPU Memory Allocations in DNN Training



- Major GPU memory consumers: **Weights** & **Feature Maps**
  - **Weights** (General): weights (①), gradients (②), optimizer states (③)
  - **Feature Maps**: Data entries stashed by the forward pass to compute the gradients in the backward pass.

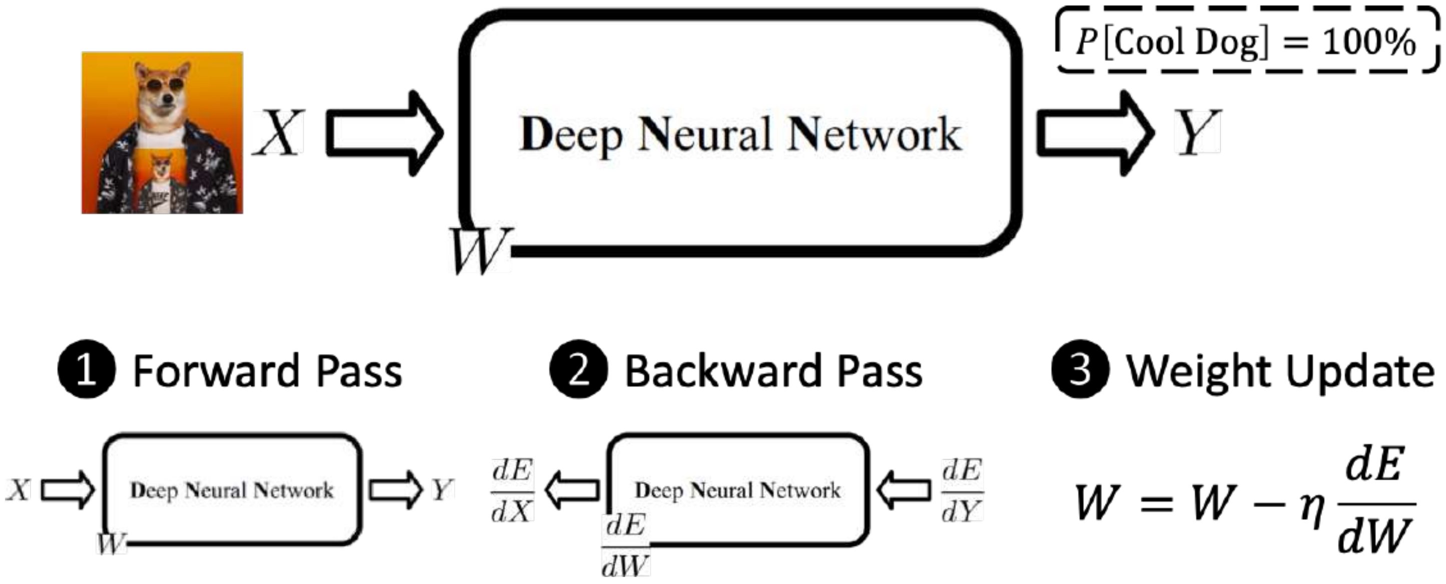
GPU Memory Allocations in DNN Training

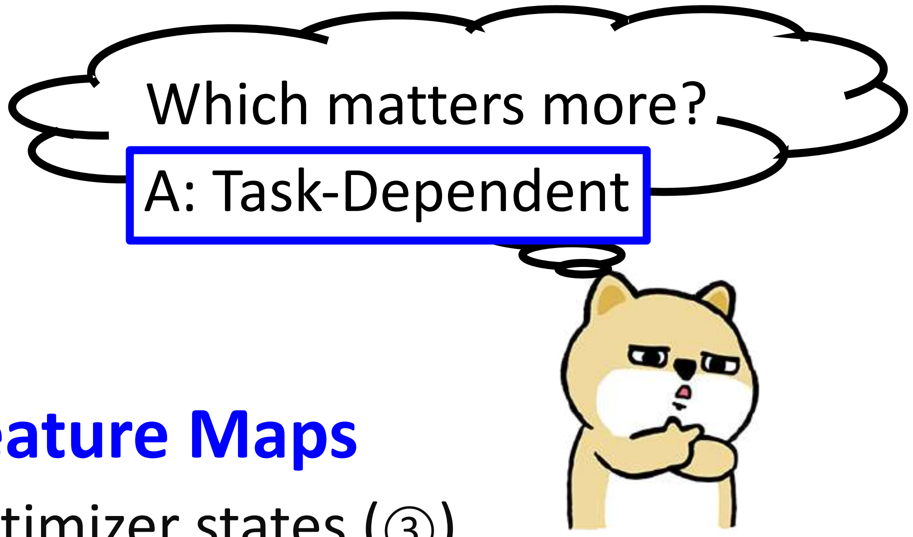




- Major GPU memory consumers: **Weights** & **Feature Maps**
  - **Weights** (General): weights (①), gradients (②), optimizer states (③)
  - **Feature Maps**: Data entries stashed by the forward pass to compute the gradients in the backward pass.

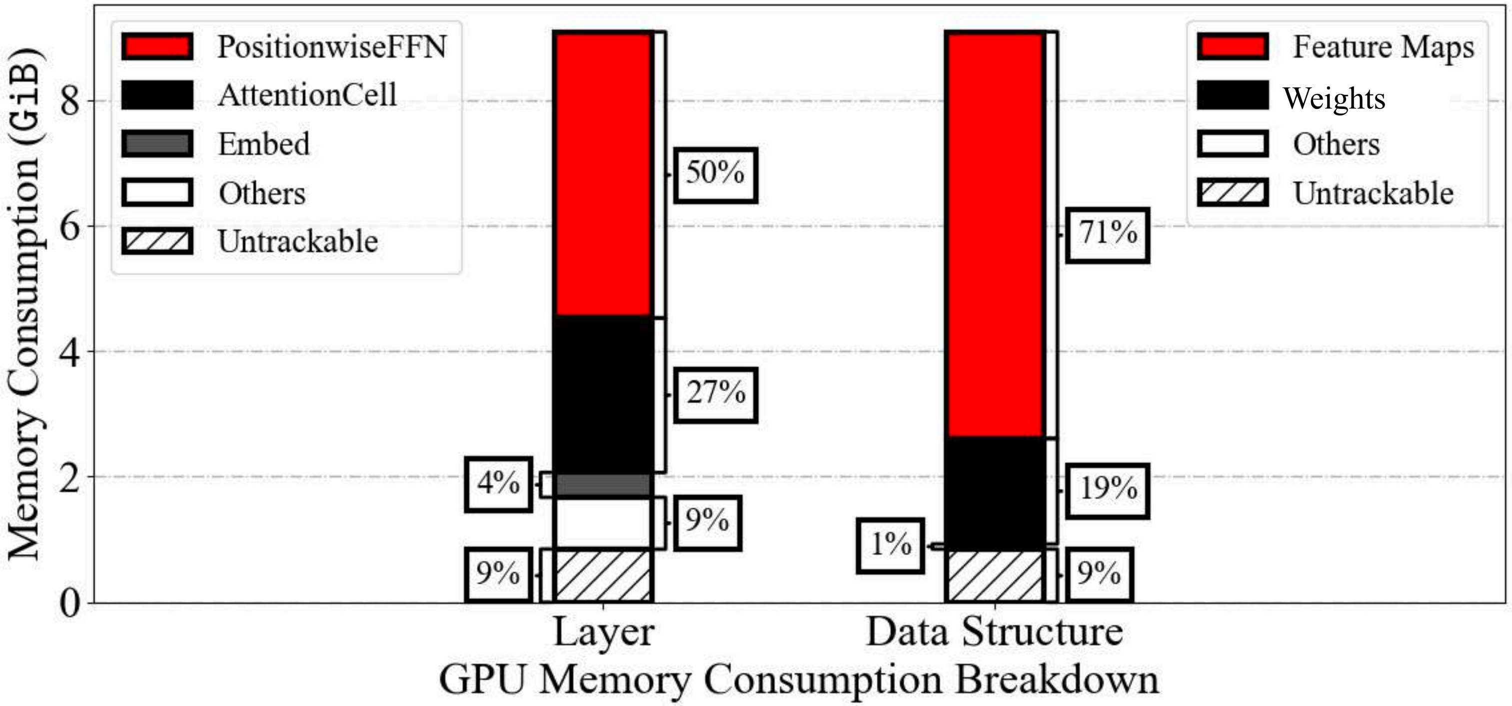
GPU Memory Allocations in DNN Training





- Major GPU memory consumers: **Weights** & **Feature Maps**
  - **Weights** (General): weights (①), gradients (②), optimizer states (③)
  - **Feature Maps**: Data entries stashed by the forward pass to compute the gradients in the backward pass.

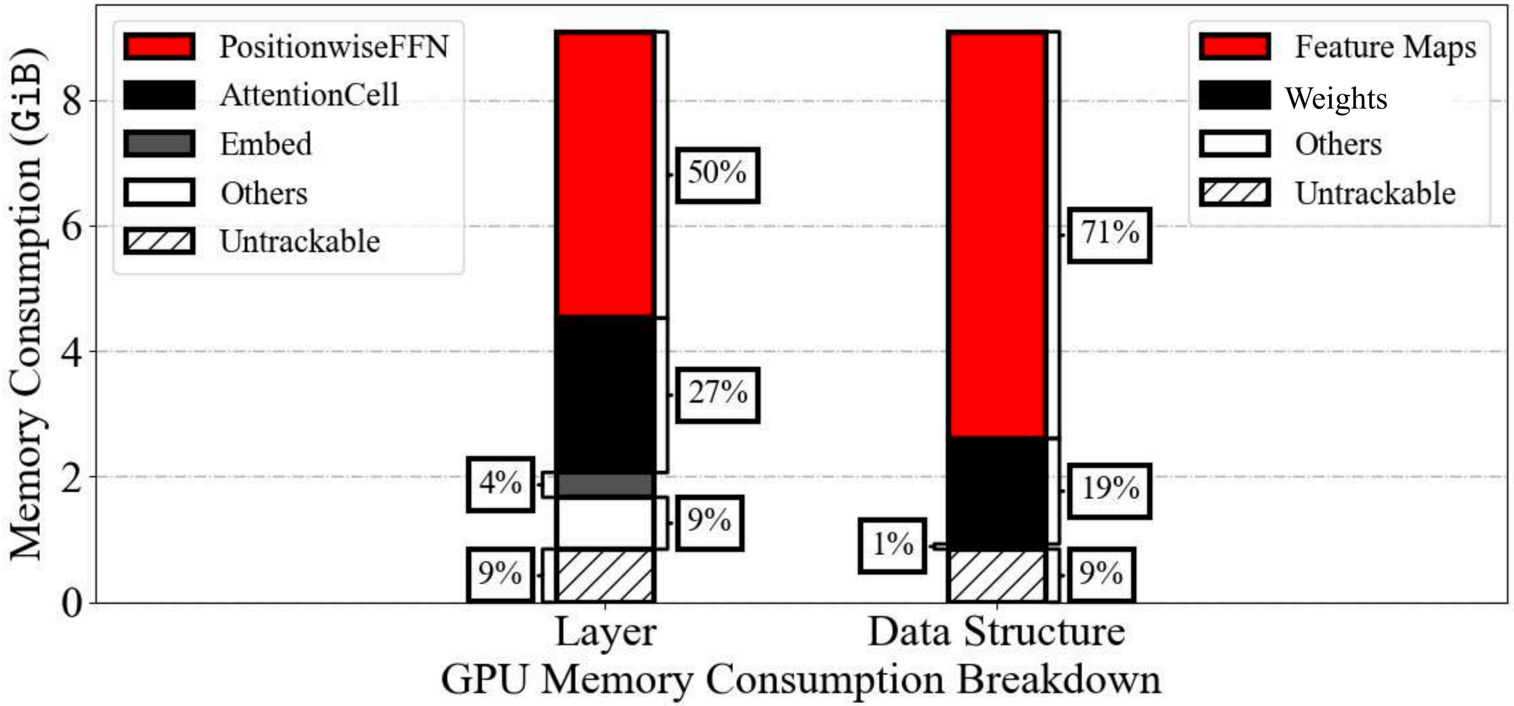
GPU Memory Profiler[1] (BERT)



[1] B. Zheng et al. *Echo: Compiler-based GPU Memory Footprint Reduction for LSTM RNN Training*. ISCA 2020

[2] J. Devlin. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. NAACL 2019*

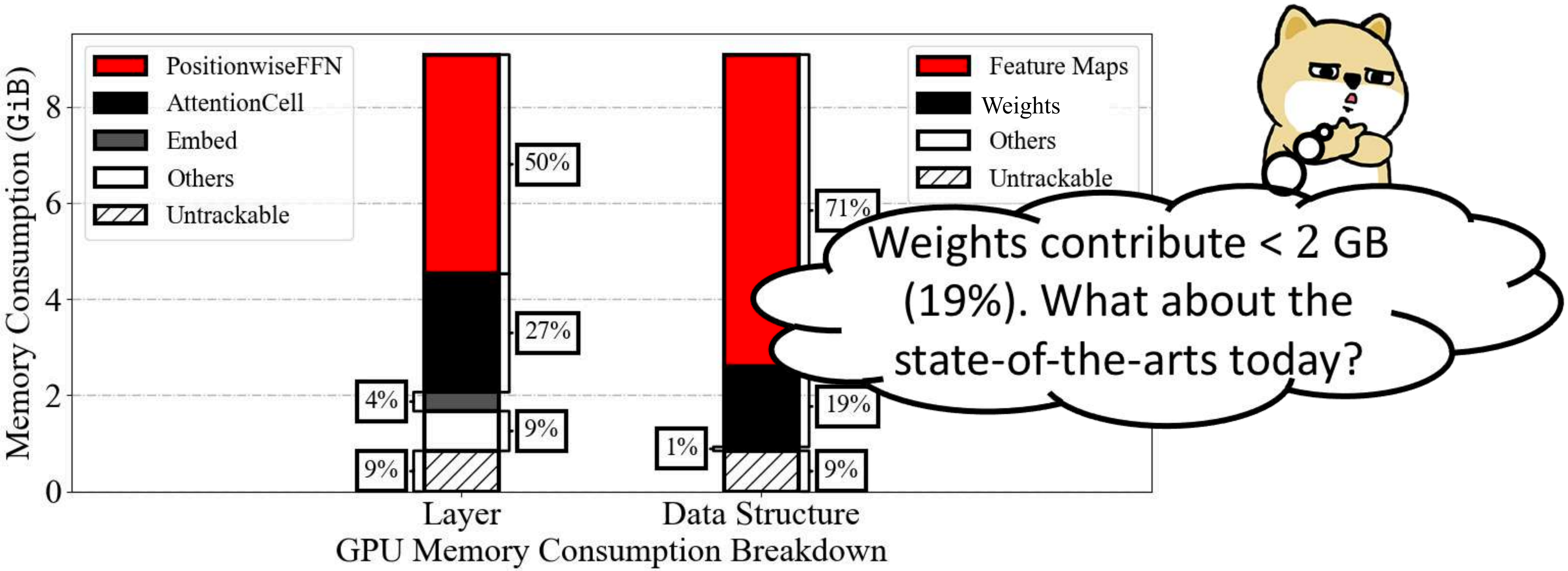
GPU Memory Profiler[1] (BERT)



Feature maps are more important than weights in BERT[2] training

[1] B. Zheng et al. *Echo: Compiler-based GPU Memory Footprint Reduction for LSTM RNN Training*. ISCA 2020
[2] J. Devlin. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. NAACL 2019*

GPU Memory Profiler[1] (BERT)



Feature maps are more important than weights in BERT[2] training

[1] B. Zheng et al. *Echo: Compiler-based GPU Memory Footprint Reduction for LSTM RNN Training*. ISCA 2020
[2] J. Devlin. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. NAACL 2019*

DNNs Today

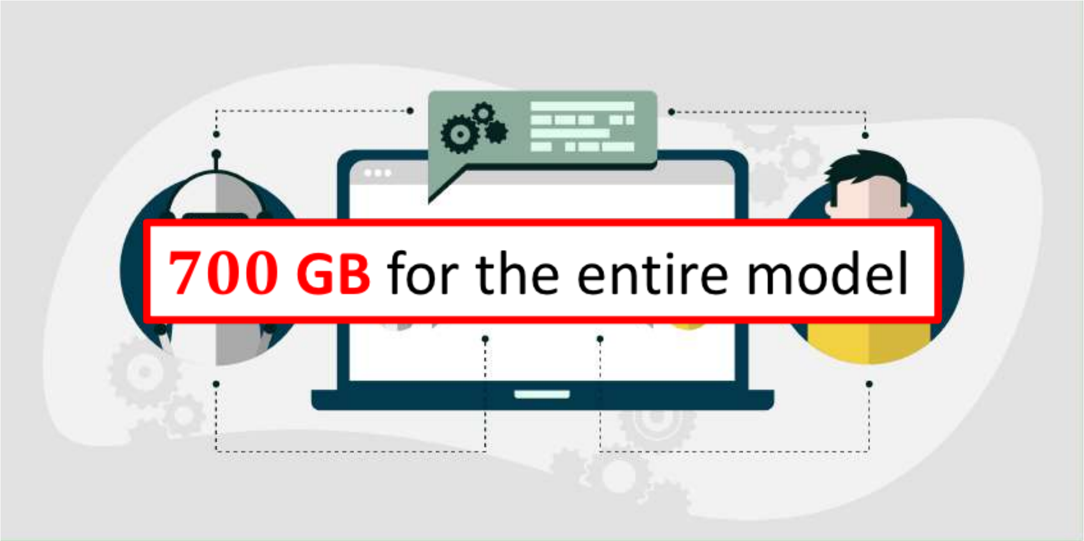
Natural Language Processing[1]

Recommendation[2]

[1] T. B. Brown et al. *Language Models are Few-Shot Learners*. arXiv 2020
[2] M. Wang et al. *Characterizing Deep Learning Training Workloads on Alibaba-PAI*. IISWC 2019

DNNs Today

Natural Language Processing[1]

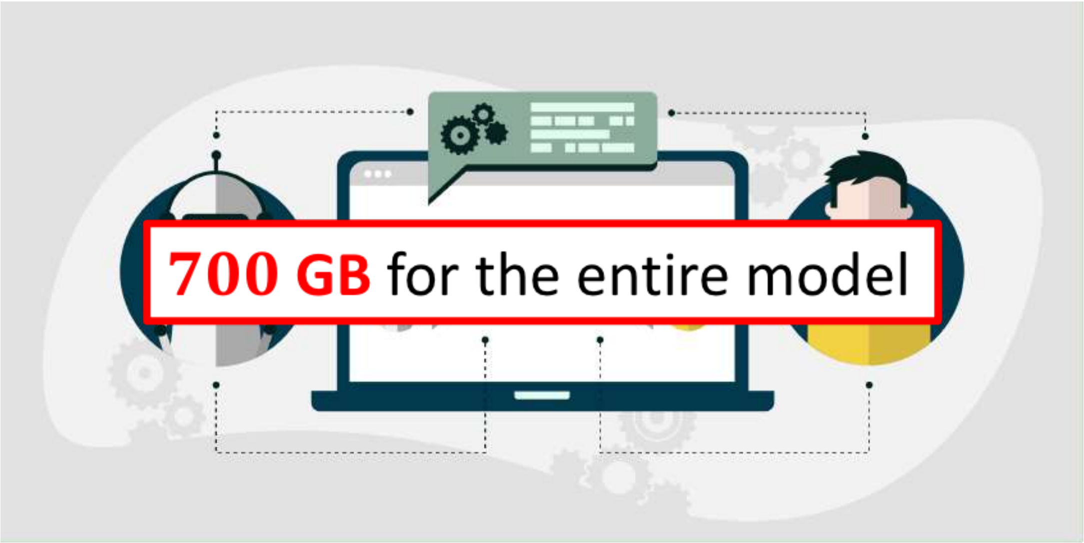


Recommendation[2]

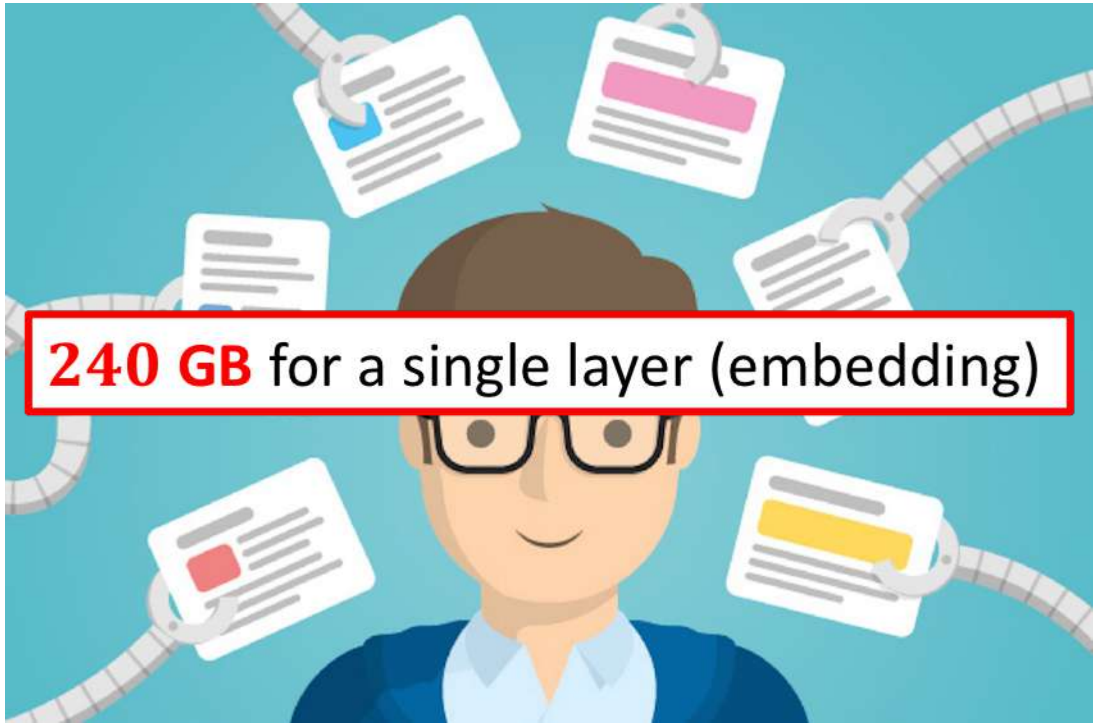
[1] T. B. Brown et al. *Language Models are Few-Shot Learners*. arXiv 2020
[2] M. Wang et al. *Characterizing Deep Learning Training Workloads on Alibaba-PAI*. IISWC 2019

DNNs Today

Natural Language Processing[1]



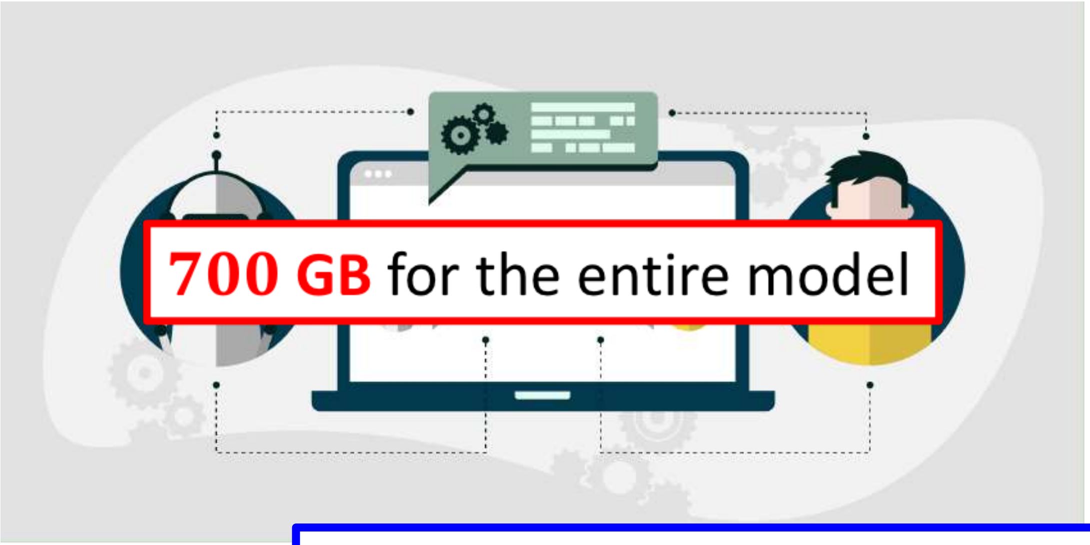
Recommendation[2]



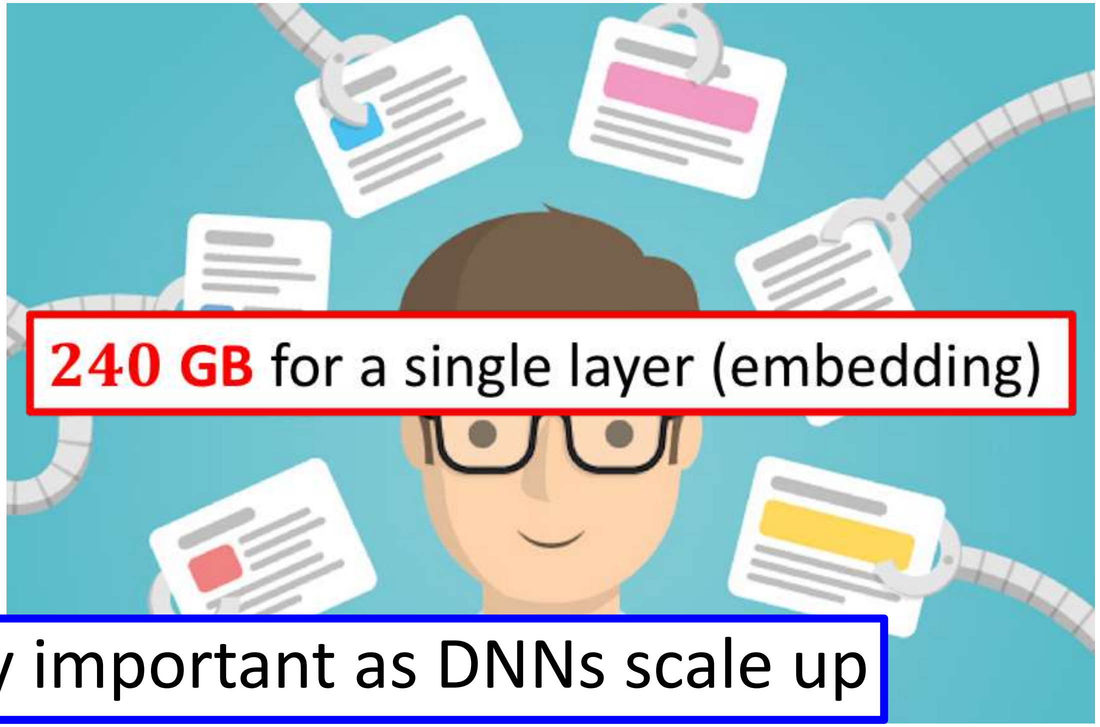
[1] T. B. Brown et al. *Language Models are Few-Shot Learners*. arXiv 2020
[2] M. Wang et al. *Characterizing Deep Learning Training Workloads on Alibaba-PAI*. IISWC 2019

DNNs Today

Natural Language Processing[1]



Recommendation[2]

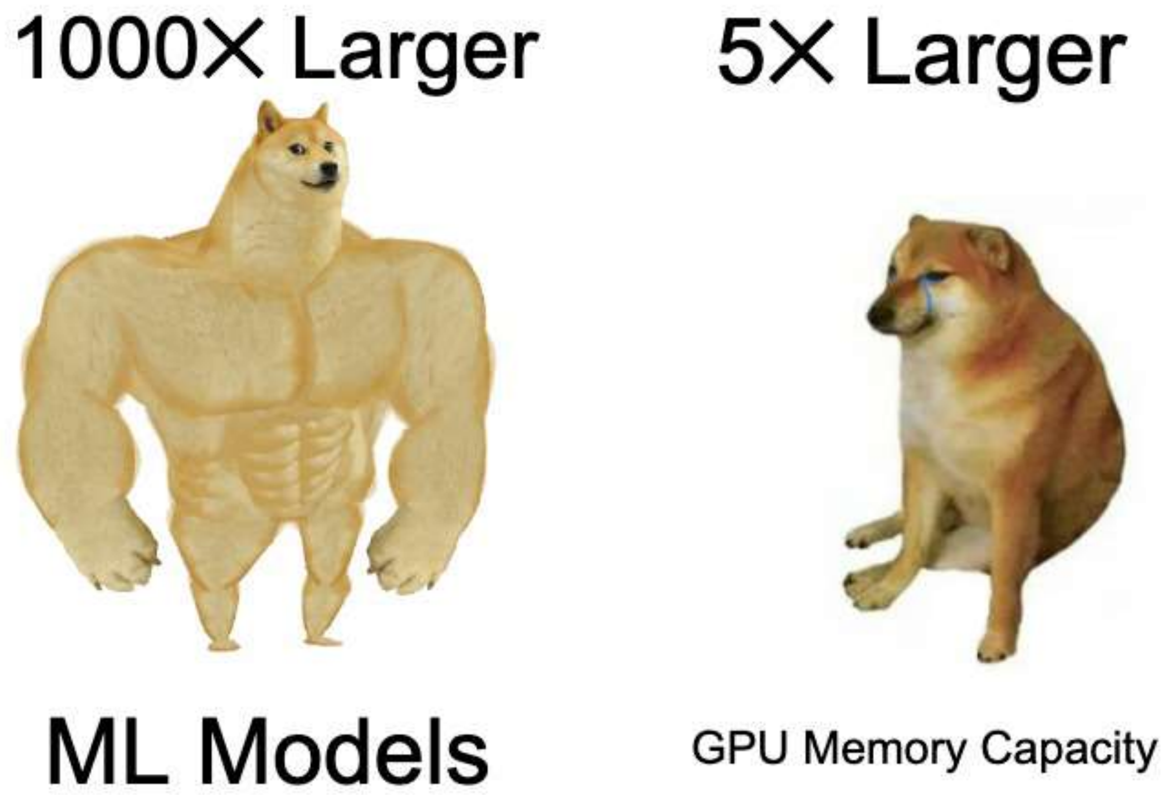


Weights become growingly important as DNNs scale up

[1] T. B. Brown et al. *Language Models are Few-Shot Learners*. arXiv 2020
[2] M. Wang et al. *Characterizing Deep Learning Training Workloads on Alibaba-PAI*. IISWC 2019

Why Larger GPU Memory?

In 3 years[1], …

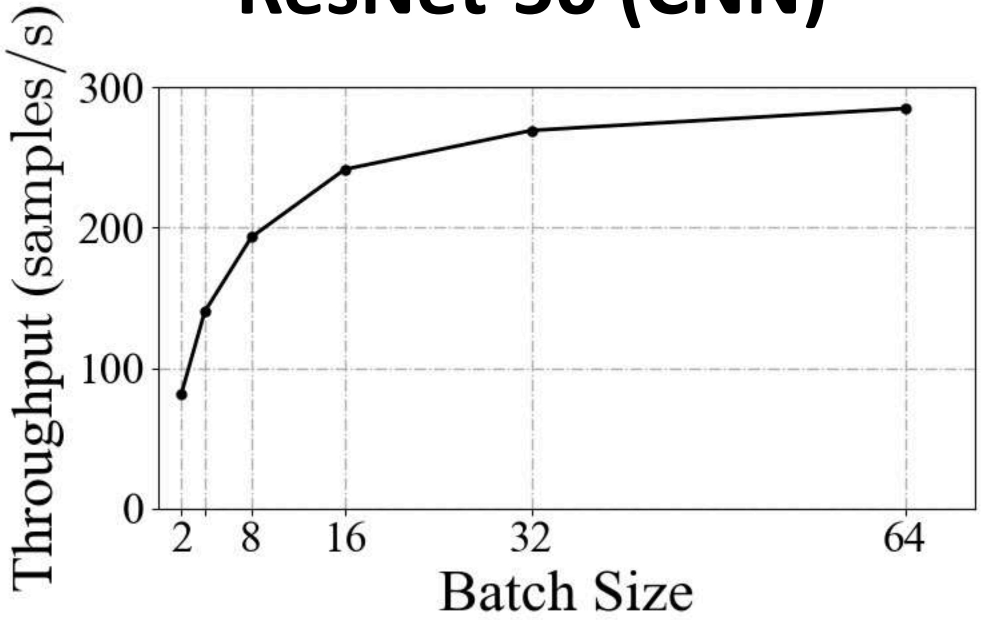


Democratize state-of-the-art machine learning models

[1] S. Rajbhandari et al. *ZeRO-Infinity: Breaking the GPU Memory Wall for Extreme Scale Deep Learning*. arXiv 2021

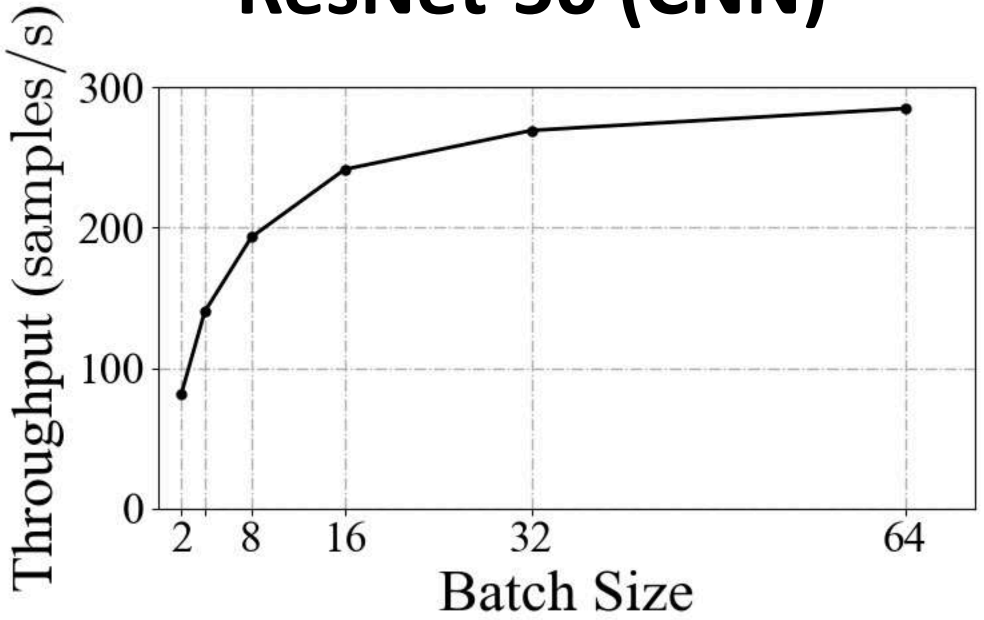
Why Larger GPU Memory?

ResNet-50 (CNN)

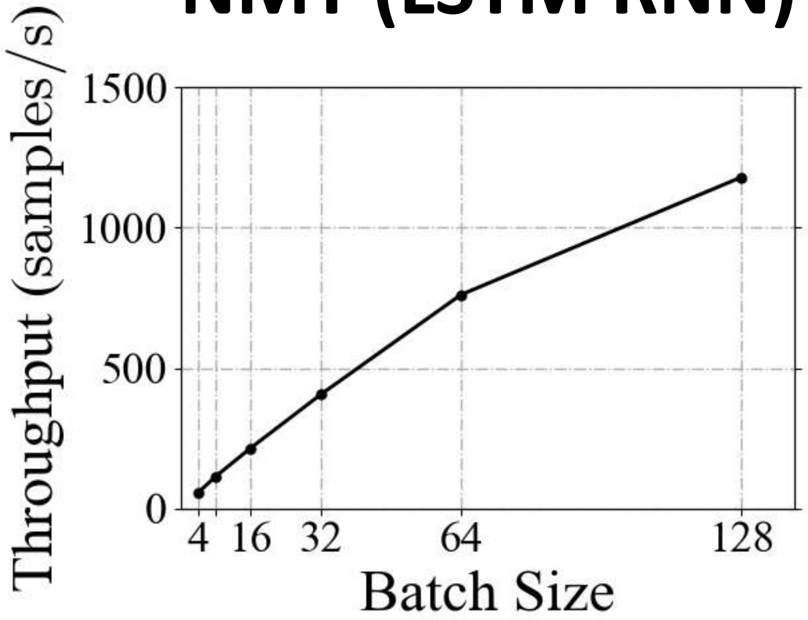


Why Larger GPU Memory?

ResNet-50 (CNN)



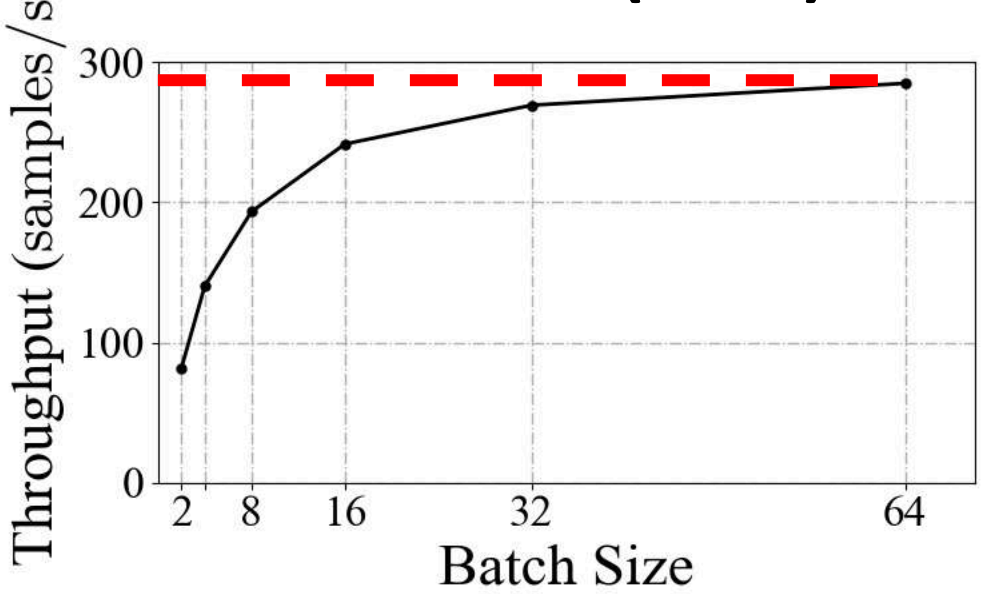
NMT (LSTM RNN)



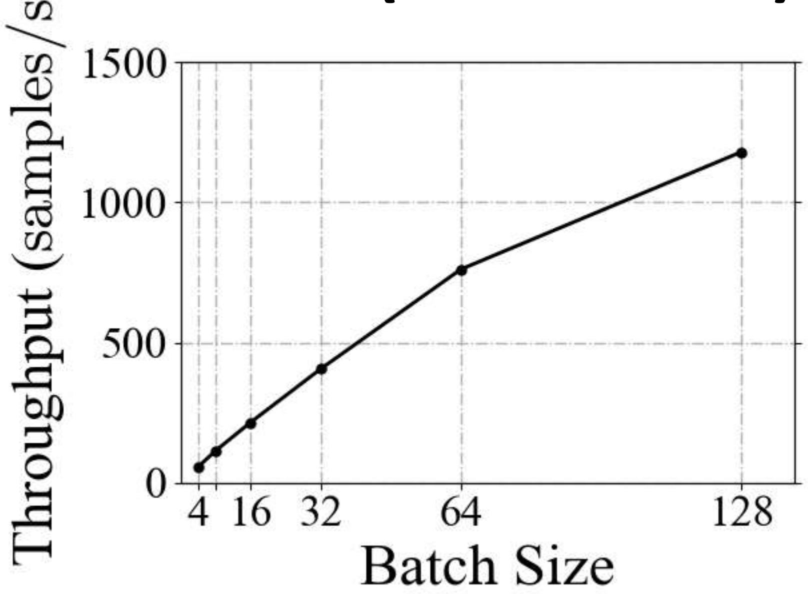
Why Larger GPU Memory?

- Training throughput **saturates** as batch size increases

ResNet-50 (CNN)



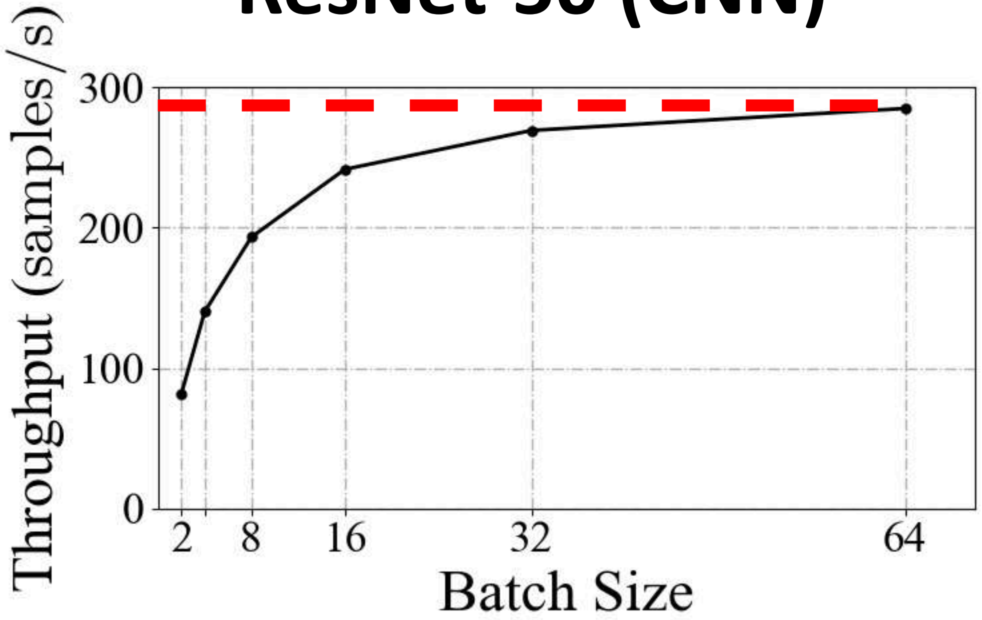
NMT (LSTM RNN)



Why Larger GPU Memory?

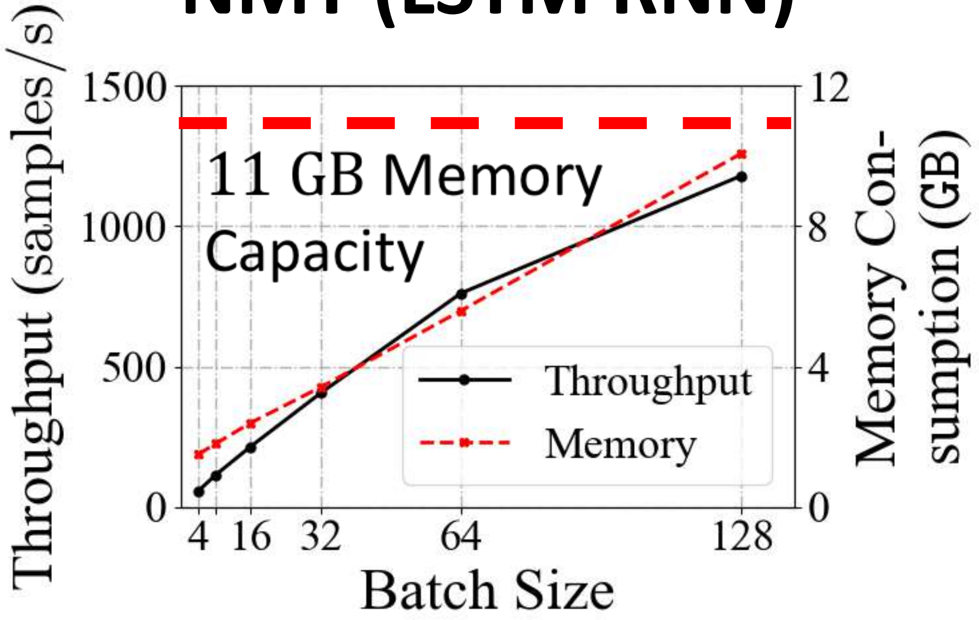
- Training throughput **saturates** as batch size increases

ResNet-50 (CNN)



- Training throughput is **limited** by the memory capacity[1, 2]

NMT (LSTM RNN)



[1] H. Zhu et al. *Benchmarking and Analyzing Deep Neural Network Training*. IISWC 2018
[2] B. Zheng et al. *Echo: Compiler-based GPU Memory Footprint Reduction for LSTM RNN Training*. ISCA 2020

A History of Prior Works

2015-2016

Weight Pruning for Efficient **Inference**

[1] T. Chen et al. *DianNao: A Small-Footprint High-Throughput Accelerator for Ubiquitous Machine-Learning*. ASPLOS 2014
[2] S. Han et al. *EIE: Efficient Inference Engine on Compressed Deep Neural Network*. ISCA 2015
[3] S. Han et al. *Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding*. ICLR 2015
[4] Y. Chen et al. *Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks*. ISCA 2016
…

A History of Prior Works

2015-2016 **2016-**

Weight Pruning for Efficient **Inference**

Feature Maps Reduction for Efficient **Training**

[1] T. Chen et al. *DianNao: A Small-Footprint High-Throughput Accelerator for Ubiquitous Machine-Learning*. ASPLOS 2014
[2] S. Han et al. *EIE: Efficient Inference Engine on Compressed Deep Neural Network*. ISCA 2015
[3] S. Han et al. *Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding*. ICLR 2015
[4] Y. Chen et al. *Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks*. ISCA 2016
…

[5] T. Chen et al. *Training Deep Nets with Sublinear Memory Cost*. arXiv 2016
[6] M. Rhu et al. *vDNN: Virtualized Deep Neural Networks for Scalable, Memory-Efficient Neural Network Design*. MICRO 2016
[7] A. Jain et al. *Gist: Efficient Data Encoding for Deep Neural Network Training*. ISCA 2018
[8] B. Zheng et al. *Echo: Compiler-based GPU Memory Footprint Reduction for LSTM RNN Training*. ISCA 2020
…

A History of Prior Works

2015-2016 | **2016-** | **2019-**

Weight Pruning for Efficient **Inference**

Feature Maps Reduction for Efficient **Training**

Weight Placement for **EX-Large Models**

[1] T. Chen et al. *DianNao: A Small-Footprint High-Throughput Accelerator for Ubiquitous Machine-Learning*. ASPLOS 2014
[2] S. Han et al. *EIE: Efficient Inference Engine on Compressed Deep Neural Network*. ISCA 2015
[3] S. Han et al. *Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding*. ICLR 2015
[4] Y. Chen et al. *Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks*. ISCA 2016
…

[5] T. Chen et al. *Training Deep Nets with Sublinear Memory Cost*. arXiv 2016
[6] M. Rhu et al. *vDNN: Virtualized Deep Neural Networks for Scalable, Memory-Efficient Neural Network Design*. MICRO 2016
[7] A. Jain et al. *Gist: Efficient Data Encoding for Deep Neural Network Training*. ISCA 2018
[8] B. Zheng et al. *Echo: Compiler-based GPU Memory Footprint Reduction for LSTM RNN Training*. ISCA 2020
…

[9] M. Naumov et al. *Deep Learning Training in Facebook Data Centers: Design of Scale-up and Scale-out Systems*. arXiv 2020
[10] S. Rajbhandari et al. *ZeRO-Infinity: Breaking the GPU Memory Wall for Extreme Scale Deep Learning*. arXiv 2021
…

A History of Prior Works

2015-2016

Weight Pruning for Efficient **Inference**

[1] T. Chen et al. *DianNao: A Small-Footprint High-Throughput Accelerator for Ubiquitous Machine-Learning*. ASPLOS 2014
[2] S. Han et al. *EIE: Efficient Inference Engine on Compressed Deep Neural Network*. ISCA 2015
[3] S. Han et al. *Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding*. ICLR 2015
[4] Y. Chen et al. *Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks*. ISCA 2016
…

2016-

Feature Maps Reduction for Efficient **Training**

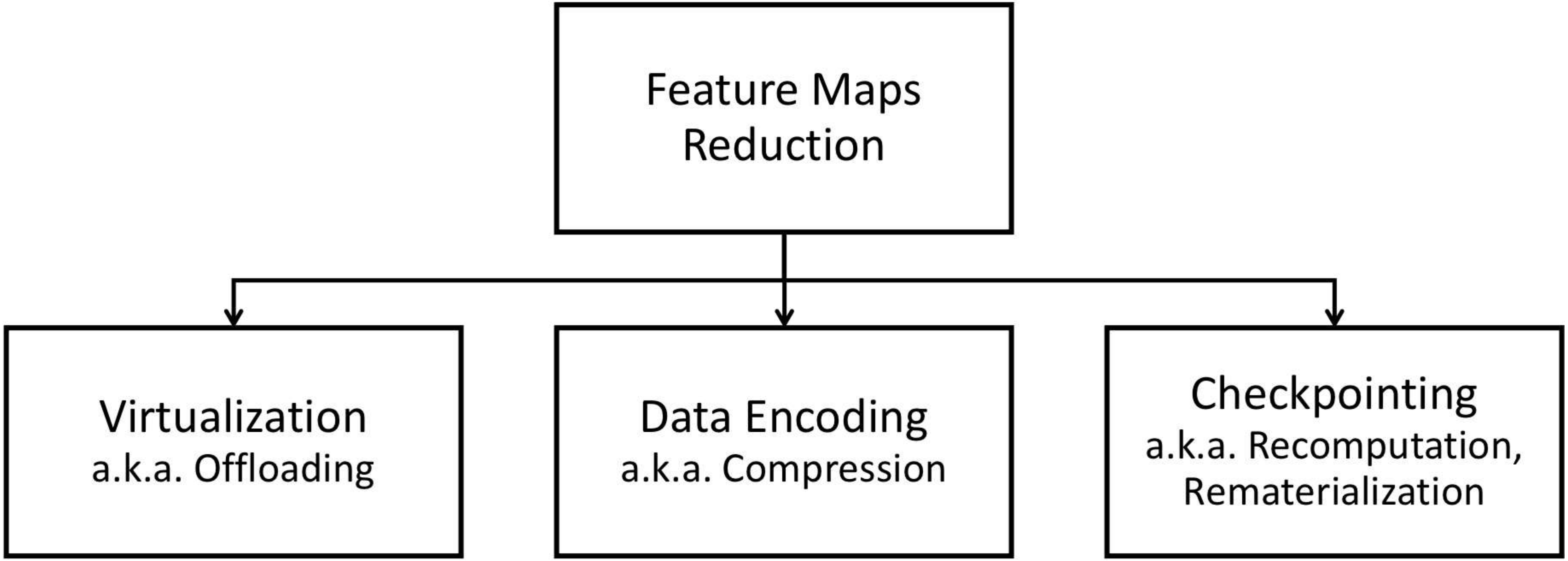
[5] T. Chen et al. *Training Deep Nets with Sublinear Memory Cost*. arXiv 2016
[6] M. Rhu et al. *vDNN: Virtualized Deep Neural Networks for Scalable, Memory-Efficient Neural Network Design*. MICRO 2016
[7] A. Jain et al. *Gist: Efficient Data Encoding for Deep Neural Network Training*. ISCA 2018
[8] B. Zheng et al. *Echo: Compiler-based GPU Memory Footprint Reduction for LSTM RNN Training*. ISCA 2020
…

2019-

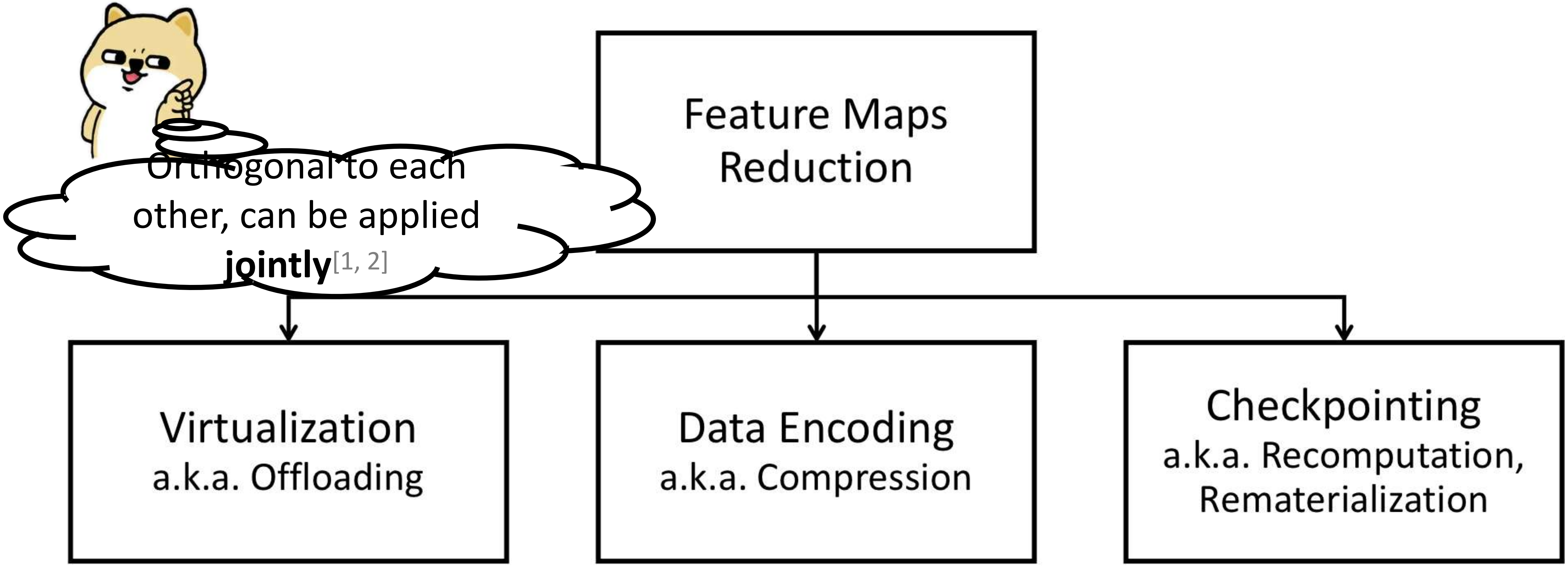
Weight Placement for **EX-Large Models**

[9] M. Naumov et al. *Deep Learning Training in Facebook Data Centers: Design of Scale-up and Scale-out Systems*. arXiv 2020
[10] S. Rajbhandari et al. *ZeRO-Infinity: Breaking the GPU Memory Wall for Extreme Scale Deep Learning*. arXiv 2021
…

Feature Maps Reduction



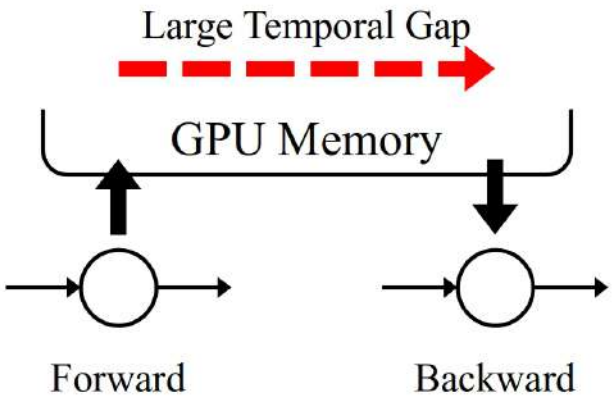
Feature Maps Reduction



[1] X. Peng et al. Capuchin*: Tensor-based GPU Memory Management for Deep Learning*. ASPLOS 2020
[2] L. Wang et al. *SuperNeurons: Dynamic GPU Memory Management for Training Deep Neural Networks*. PPoPP 2018

Baseline

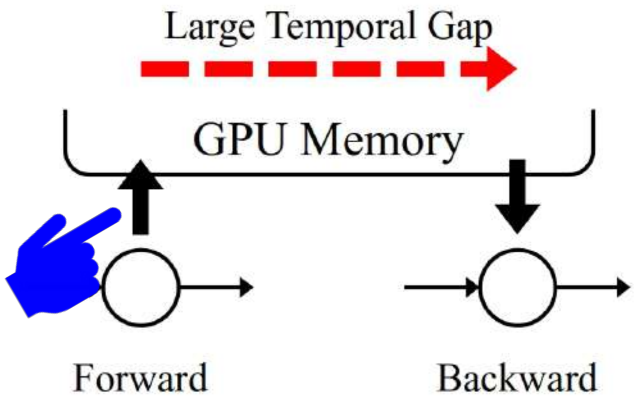
Baseline



Baseline

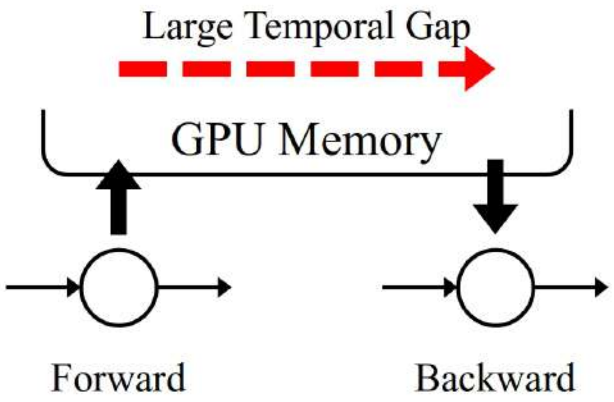
Baseline

Feature Maps

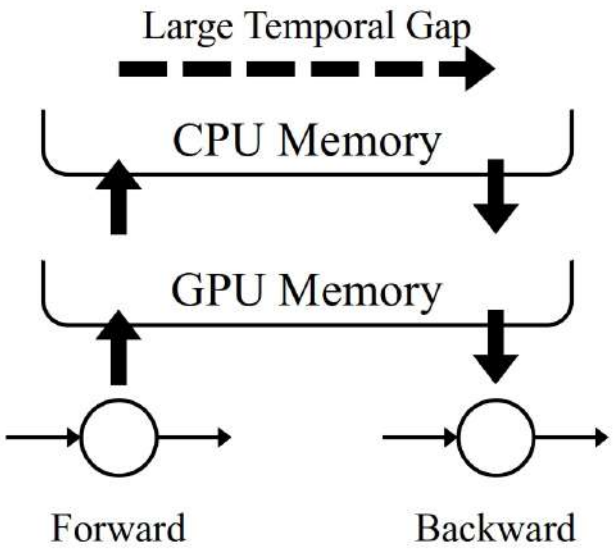


Virtualization

Baseline



Virtualization[1, 2]

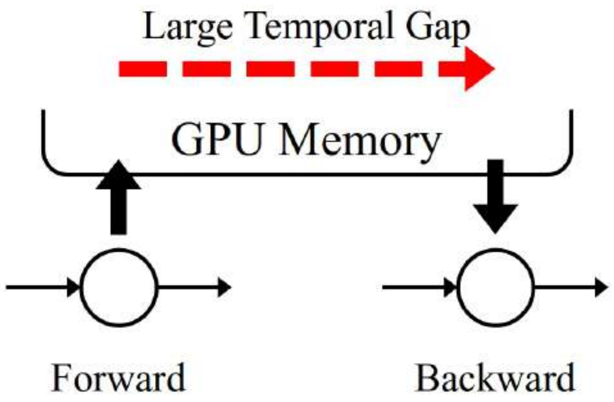


[1] M. Rhu et al. *vDNN: Virtualized Deep Neural Networks for Scalable, Memory-Efficient Neural Network Design*. MICRO 2016
[2] M. Rhu et al. *Compressing DMA Engine: Leveraging Activation Sparsity for Training Deep Neural Networks*. HPCA 2018

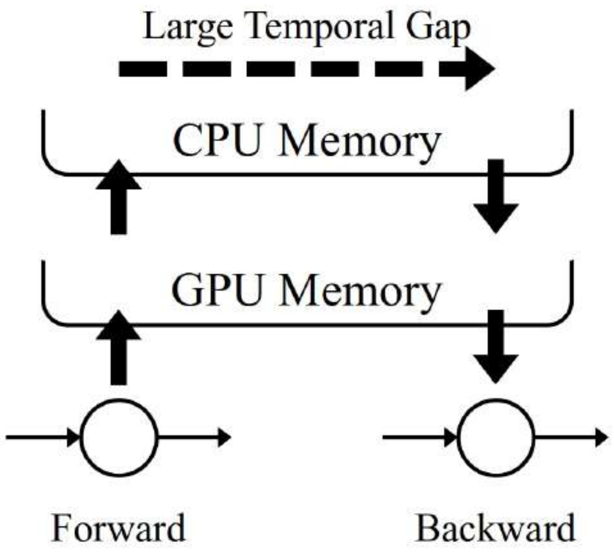
Virtualization

✅ Generic ✅ Lossless
✅ Large Reduction Ratio (up to 20×)[1]
✗ High Runtime Overhead (18%)[1]

Baseline



Virtualization[1, 2]

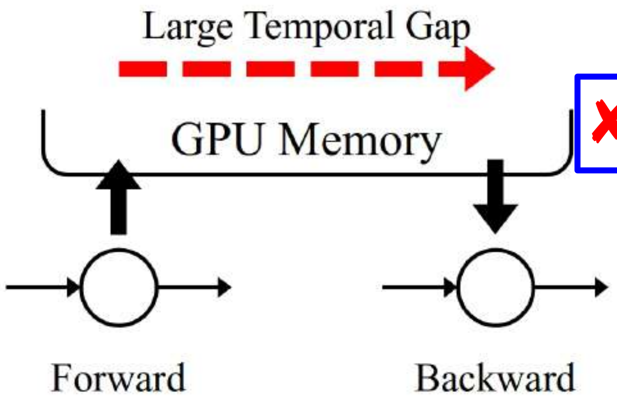


[1] M. Rhu et al. *vDNN: Virtualized Deep Neural Networks for Scalable, Memory-Efficient Neural Network Design*. MICRO 2016
[2] M. Rhu et al. *Compressing DMA Engine: Leveraging Activation Sparsity for Training Deep Neural Networks*. HPCA 2018

Virtualization

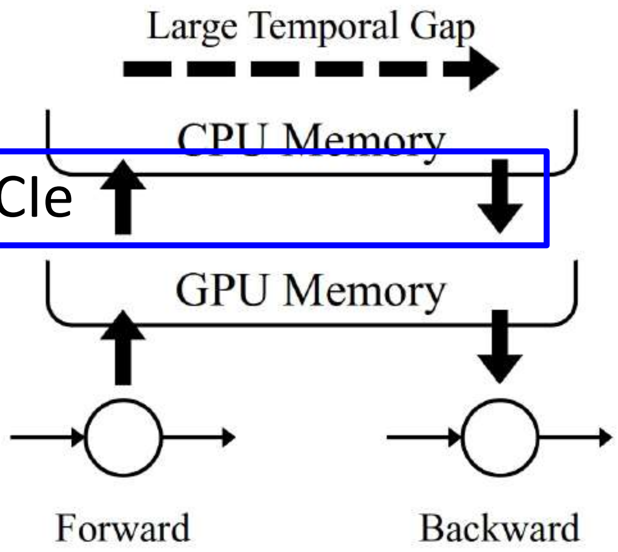
✅ Generic ✅ Lossless
✅ Large Reduction Ratio (up to 20×)[1]
✗ High Runtime Overhead (18%)[1]

Baseline



Virtualization[1, 2]

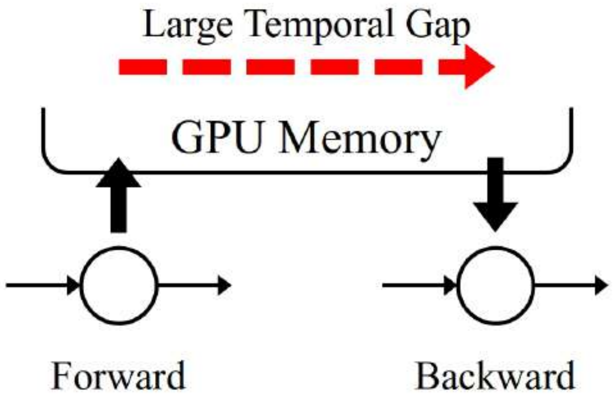
✗ Intensive Use of PCIe



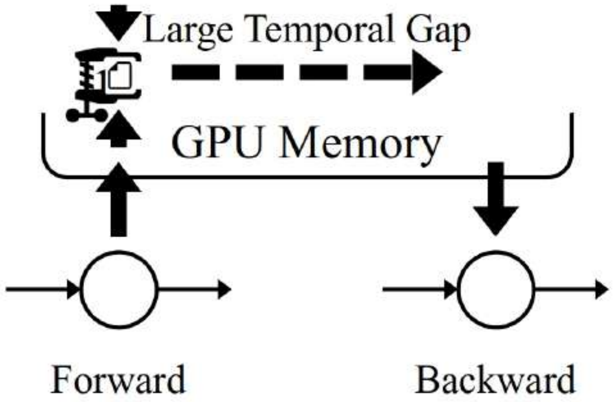
[1] M. Rhu et al. *vDNN: Virtualized Deep Neural Networks for Scalable, Memory-Efficient Neural Network Design*. MICRO 2016
[2] M. Rhu et al. *Compressing DMA Engine: Leveraging Activation Sparsity for Training Deep Neural Networks*. HPCA 2018

Data Encoding

Baseline



Data Encoding[1, 2]



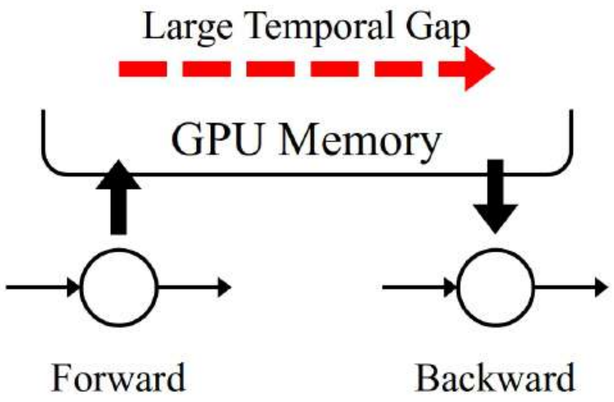
[1] A. Jain et al. *Gist: Efficient Data Encoding for Deep Neural Network Training*. ISCA 2018

[2] J. Chen, L. Zheng et al. *ActNN: Reducing Training Memory Footprint via 2-Bit Activation Compressed Training*. ICML 2021

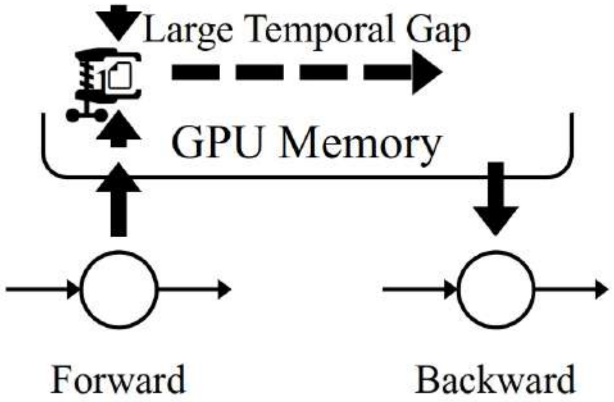
Data Encoding

- ✗ Lossless but Layer-Specific[1]
- ✗ Lossy but Generic[2]
- ✅ Large Reduction Ratio (up to 1.8×)
- ✅ Low Runtime Overhead (4%)[1]

Baseline



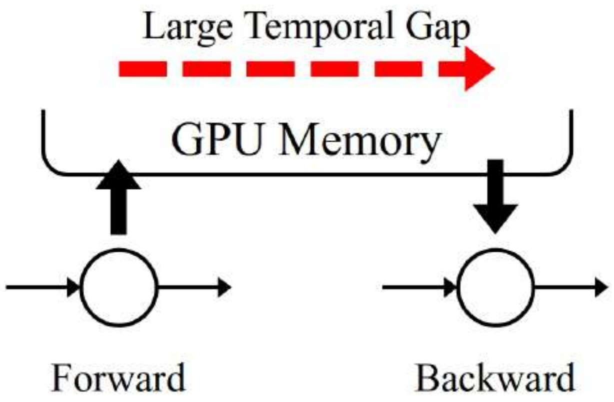
Data Encoding[1, 2]



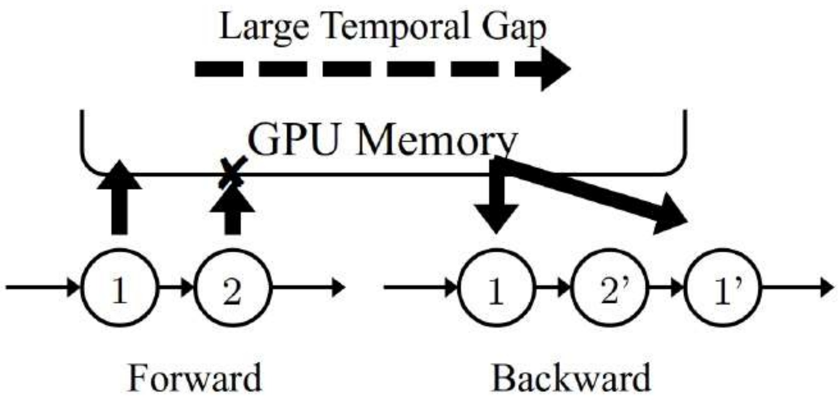
[1] A. Jain et al. *Gist: Efficient Data Encoding for Deep Neural Network Training*. ISCA 2018
[2] J. Chen, L. Zheng et al. *ActNN: Reducing Training Memory Footprint via 2-Bit Activation Compressed Training*. ICML 2021

Checkpointing

Baseline



Checkpointing[1, 2, 3, 4, 5]

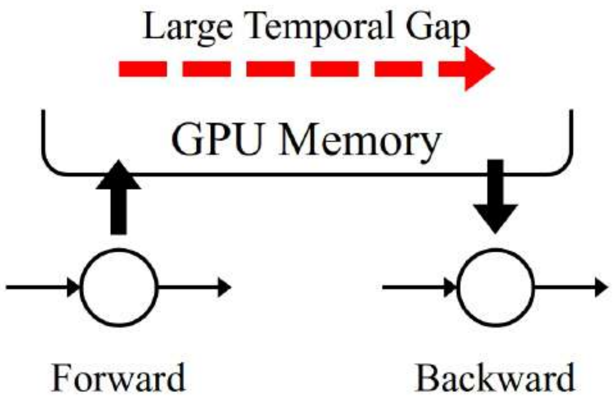


[1] T. Chen et al. *Training Deep Nets with Sublinear Memory Cost*. arXiv 2016
[2] R. Kumar et al. *Efficient Rematerialization for Deep Networks*. NeurIPS 2019
[3] P. Jain et al. *Checkmate: Breaking the Memory Wall with Optimal Tensor Rematerialization*. MLSys 2020
[4] B. Zheng et al. *Echo: Compiler-based GPU Memory Footprint Reduction for LSTM RNN Training*. ISCA 2020
[5] M. Kirisame et al. *Dynamic Tensor Rematerialization*. ICLR 2021

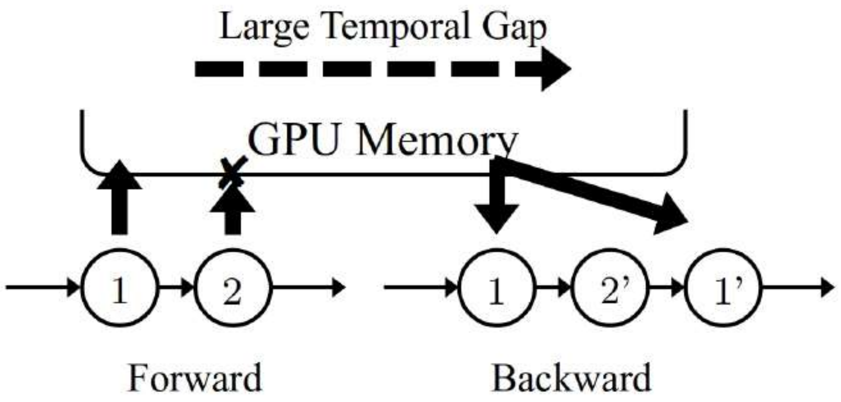
Checkpointing

✅ Generic ✅ Lossless
✅ Large Reduction Ratio (up to 3.1×)[4]
(-) Modest Runtime Overhead

Baseline



Checkpointing[1, 2, 3, 4, 5]



[1] T. Chen et al. *Training Deep Nets with Sublinear Memory Cost*. arXiv 2016
[2] R. Kumar et al. *Efficient Rematerialization for Deep Networks*. NeurIPS 2019
[3] P. Jain et al. *Checkmate: Breaking the Memory Wall with Optimal Tensor Rematerialization*. MLSys 2020
[4] B. Zheng et al. *Echo: Compiler-based GPU Memory Footprint Reduction for LSTM RNN Training*. ISCA 2020
[5] M. Kirisame et al. *Dynamic Tensor Rematerialization*. ICLR 2021

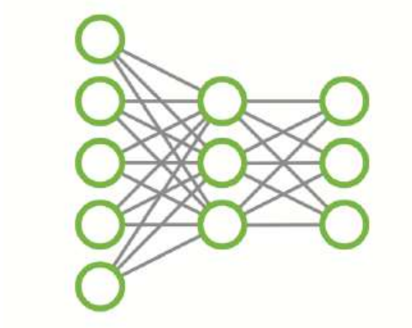
Summary

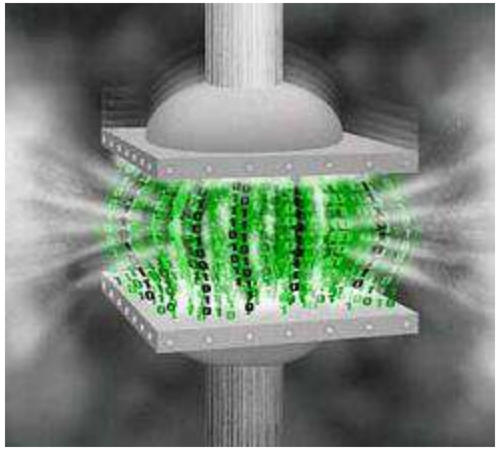
- Background on DNNs and Their Memory Allocations
  - Major memory consumers: **Weights** & **Feature Maps**
- Why larger GPU memory?
  - Larger models; Higher training throughputs
- A History of Prior Works
  - Weight → Feature Maps → Weights
- Prior Works on Feature Maps Reduction
  - 3 major techniques: **Virtualization**; **Data Encoding**; **Checkpointing**

Example Works

- Gist, ISCA'18

- Echo, ISCA'20





1. Gist: Efficient Data Encoding for Deep Neural Network Training

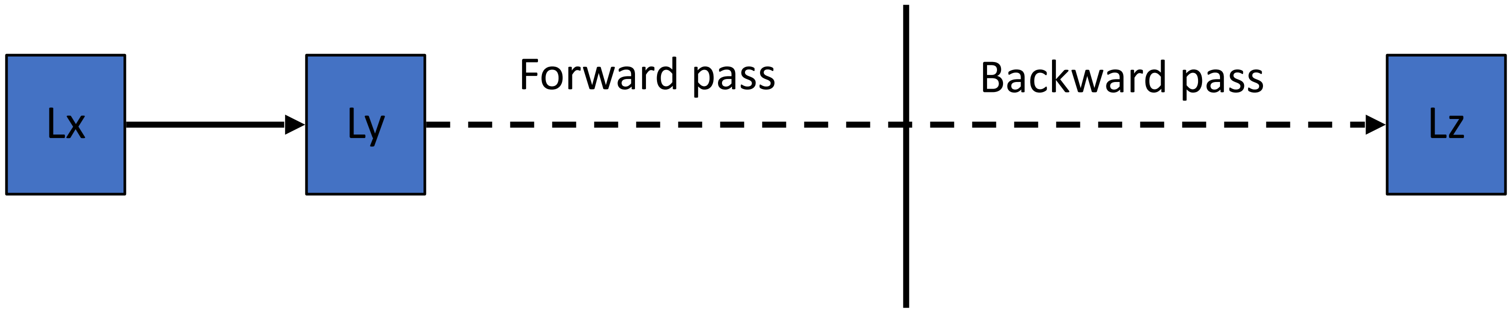
ISCA 2018

Limitations of Prior Work

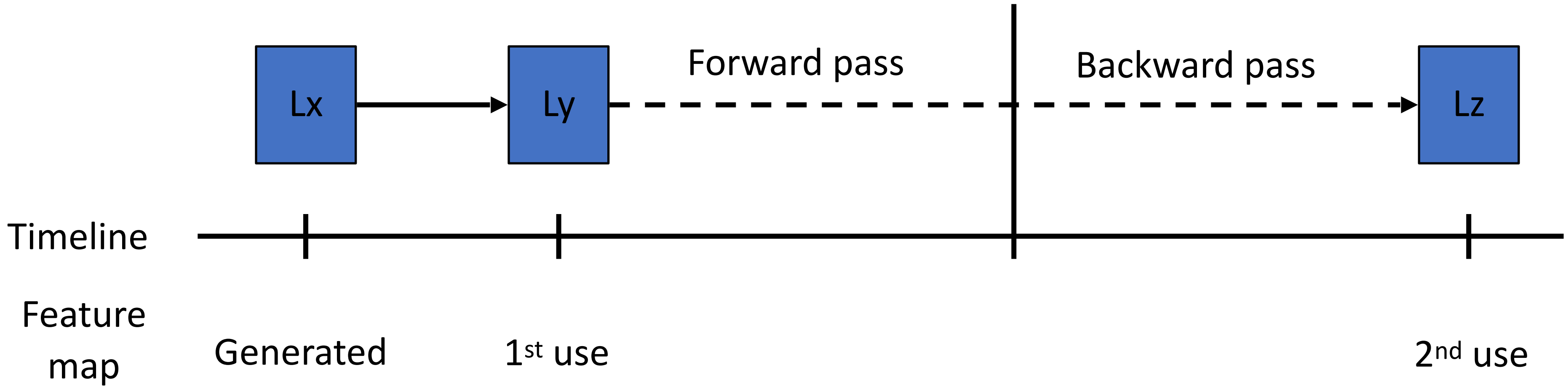
- Focus on DNN inference, *i.e.*, weights
  - Apply pruning, quantization and Huffman encoding
  - However, weights are a small fraction of memory footprint

- Additionally, techniques are not well suited for training
  - Training requires frequent weight updates
  - Map poorly on the GPU HW

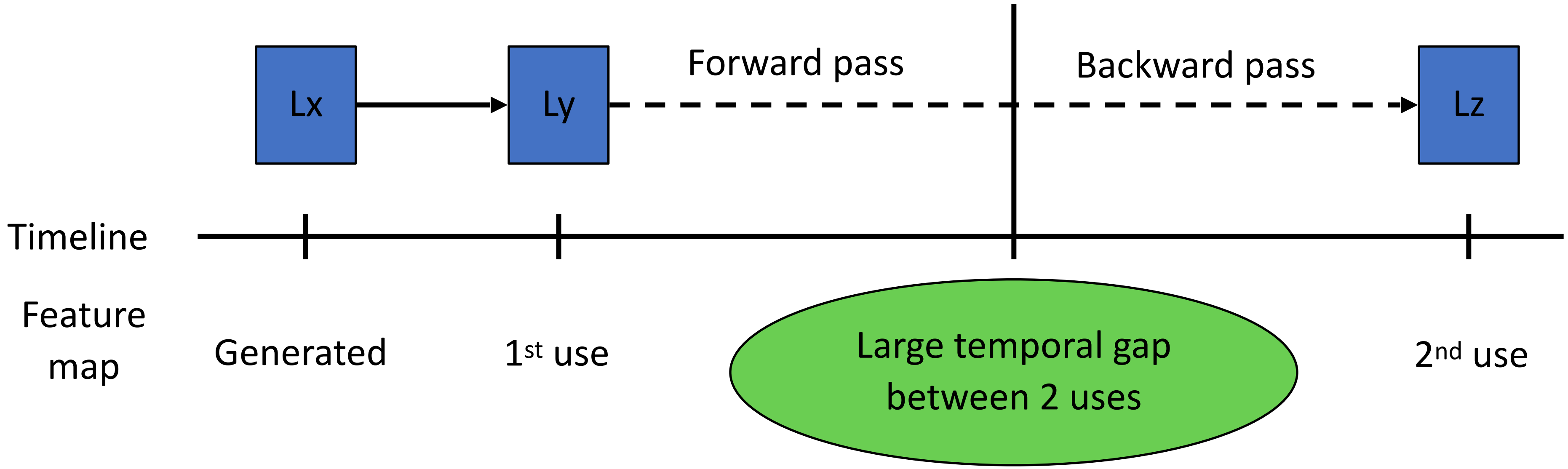
Our Insight



Our Insight

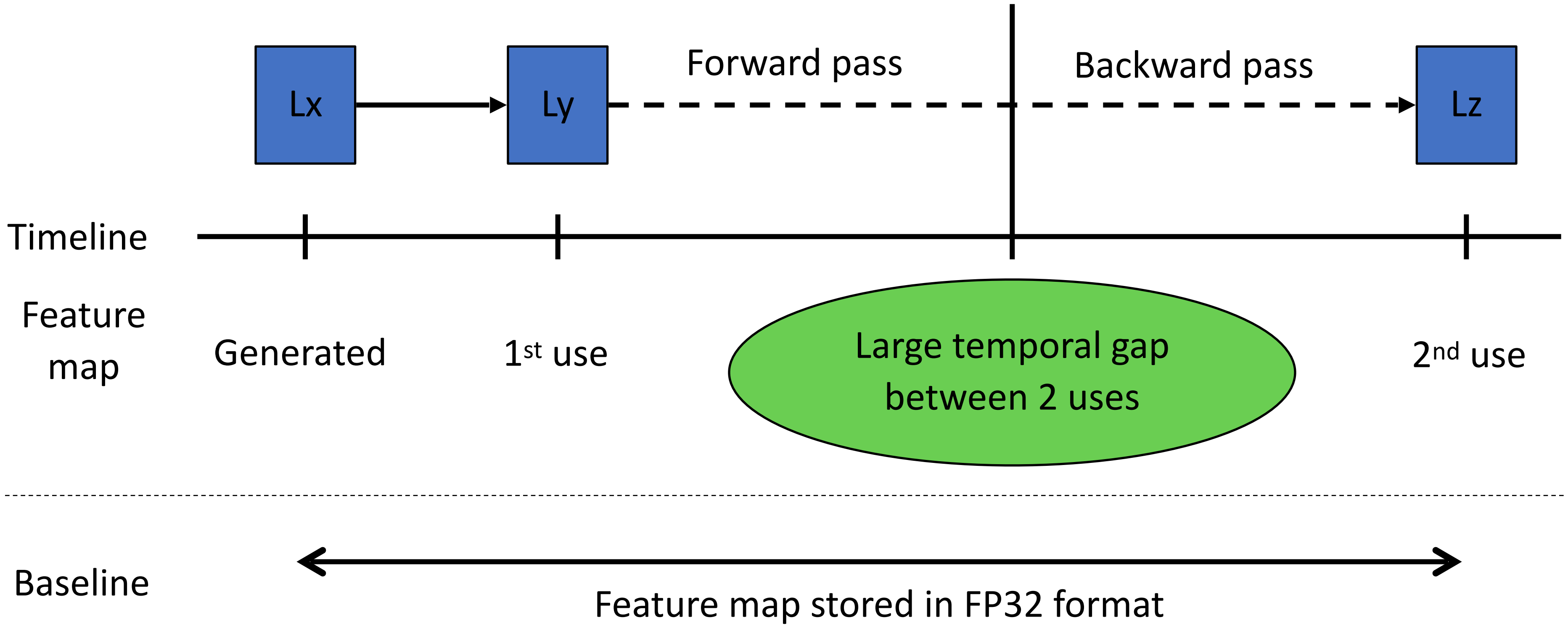

Lx
Ly
Forward pass
Backward pass
Lz
Timeline
Feature map
Generated
1st use
2nd use

Our Insight



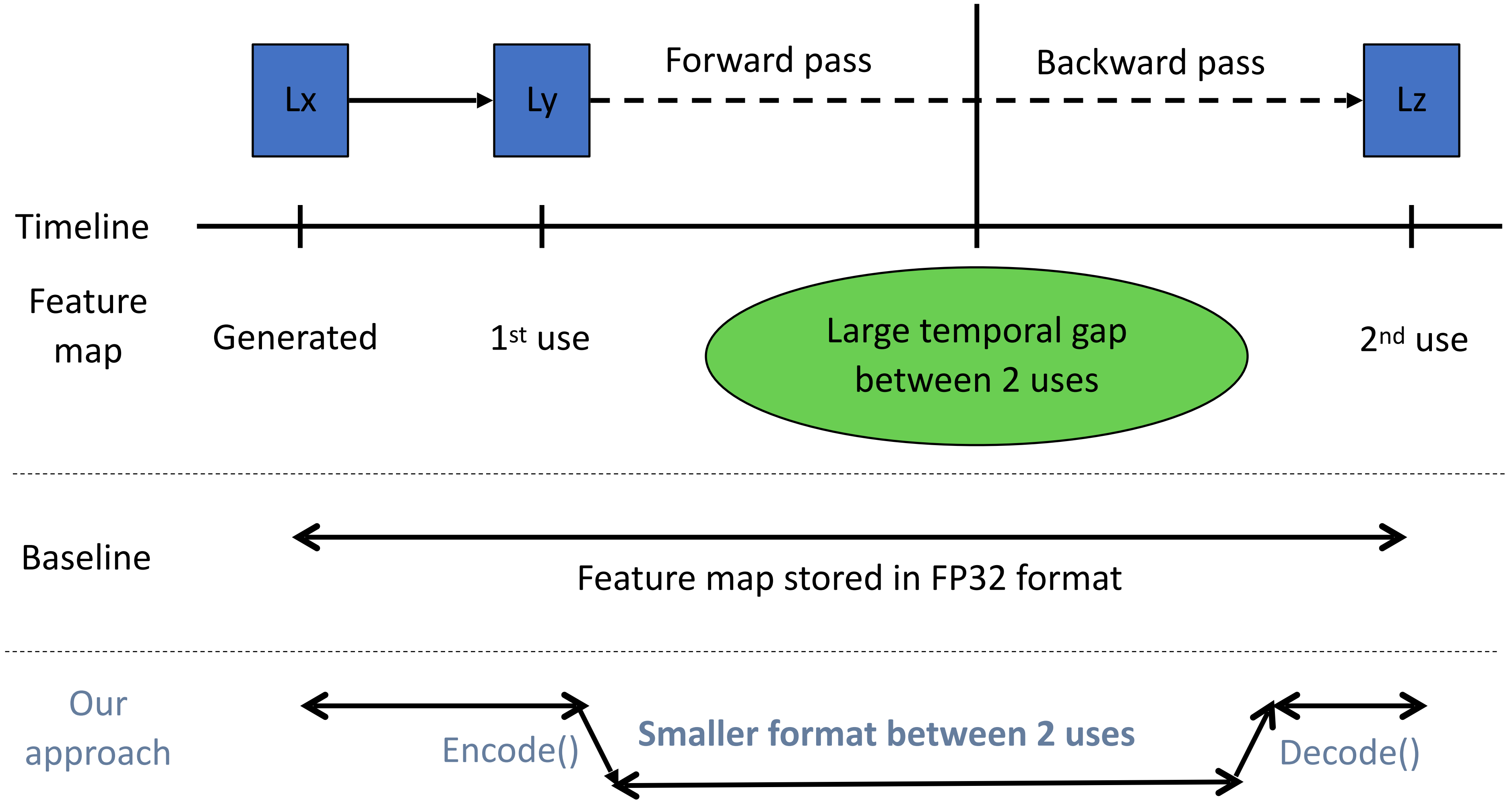
Lx
Ly
Forward pass
Backward pass
Lz
Timeline
Feature map
Generated
1st use
Large temporal gap between 2 uses
2nd use

Our Insight



Lx
Ly
Forward pass
Backward pass
Lz
Timeline
Feature map
Generated
1st use
Large temporal gap between 2 uses
2nd use
Baseline
Feature map stored in FP32 format

Our Insight

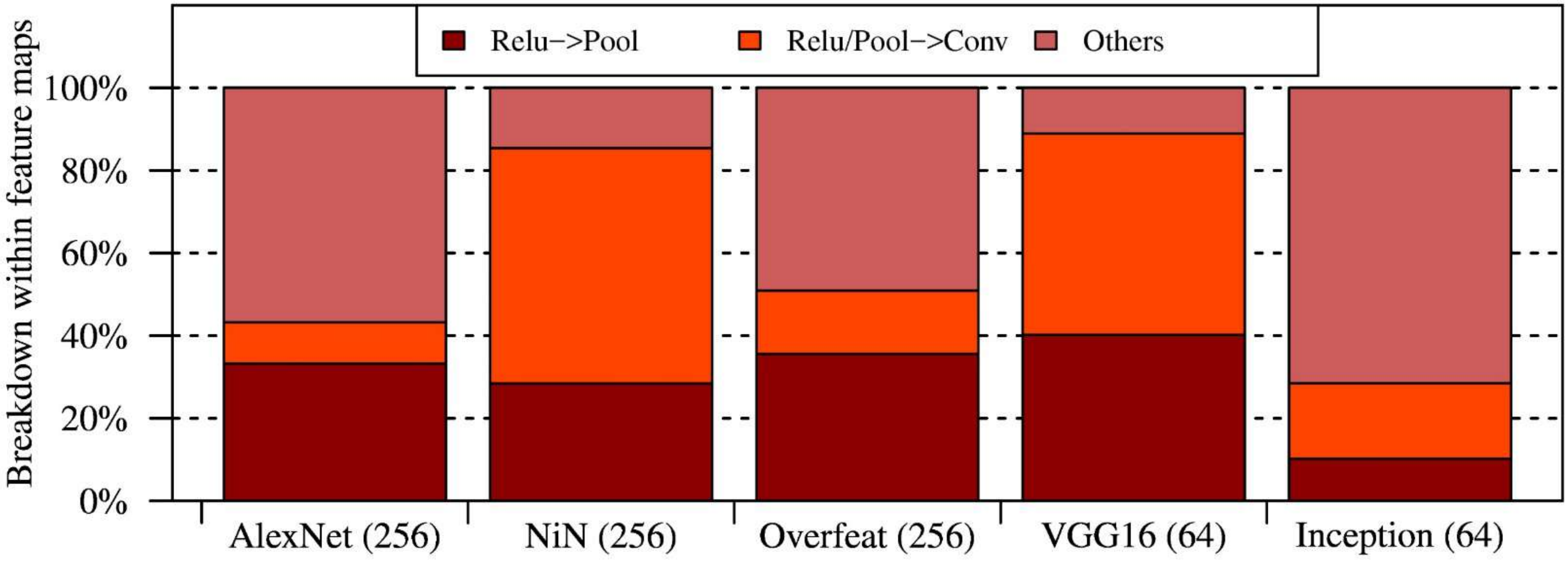


Lx
Ly
Lz
Forward pass
Backward pass
Timeline
Feature map
Generated
1st use
Large temporal gap between 2 uses
2nd use
Baseline
Feature map stored in FP32 format
Our approach
Encode()
Smaller format between 2 uses
Decode()

Layer-Specific Encodings

- Key Idea:
  - Use layer-specific compression
- Can be both fast and efficient
- Can be even lossless
  - Usually difficult for FP32

Relu Importance

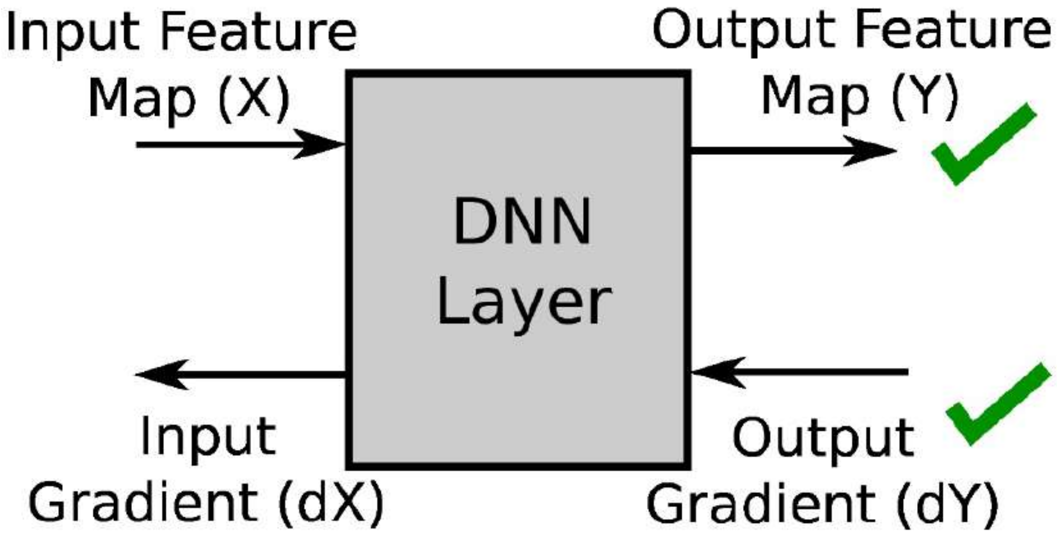


Significant footprint is due to Relu layer

CNTK Profiling

Relu -> Pool

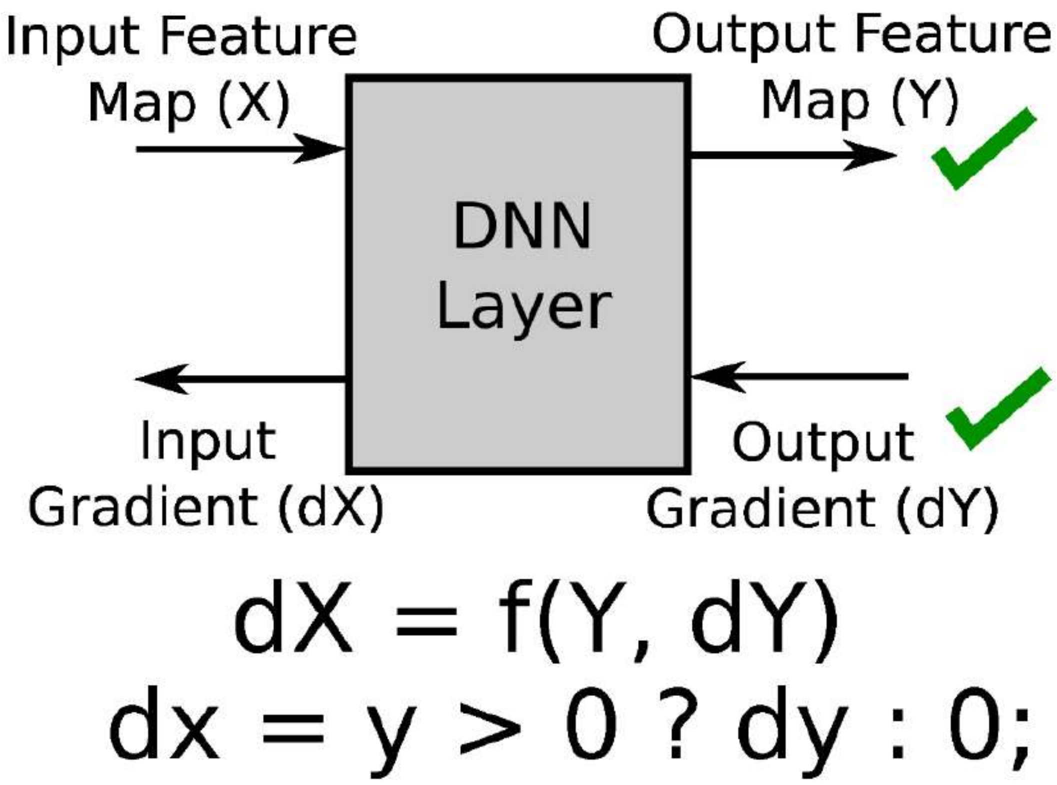
Relu Backward Propagation



$$dX = f(Y, dY)$$
$$dx = y > 0\ ?\ dy : 0;$$

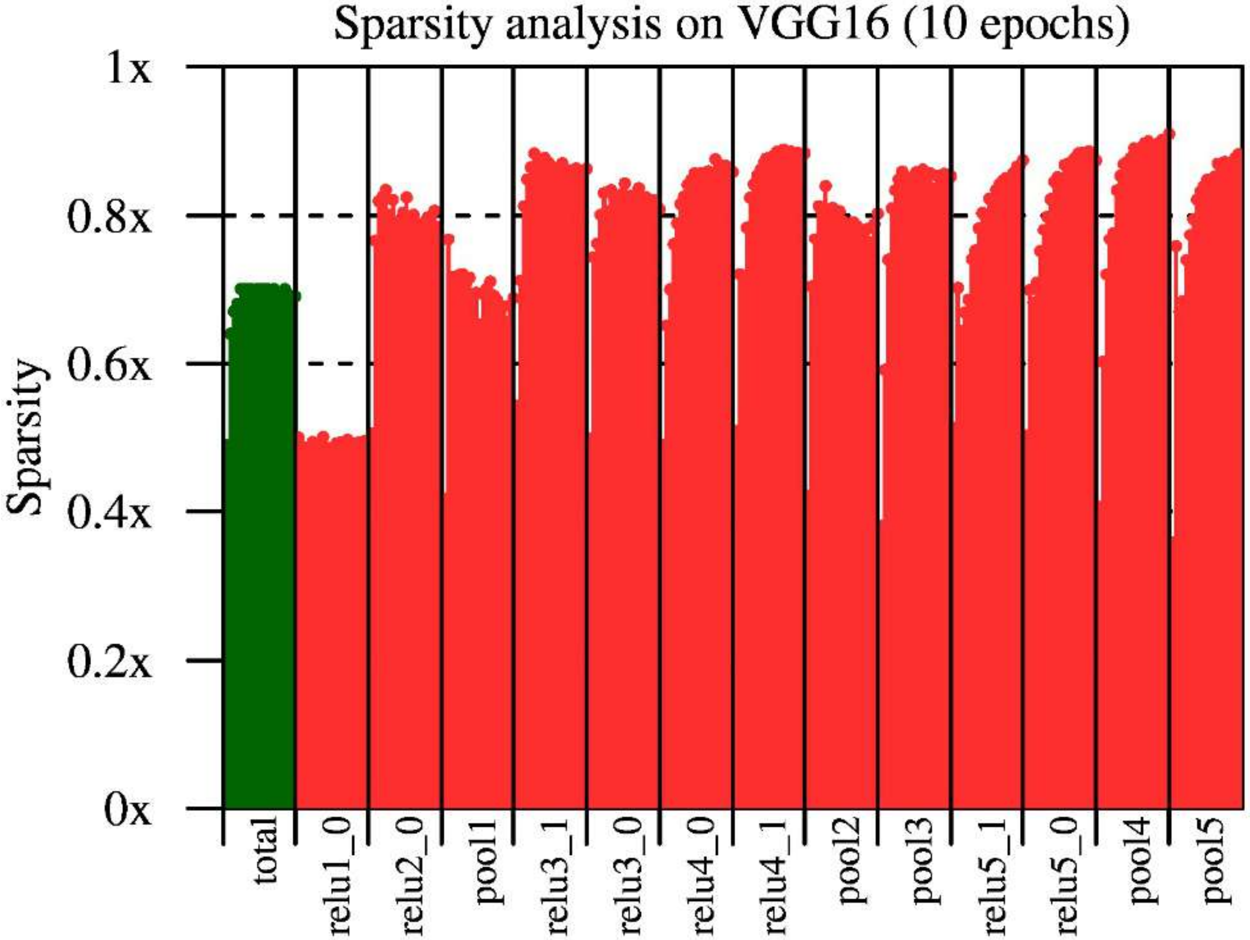
Relu -> Pool

Relu Backward Propagation

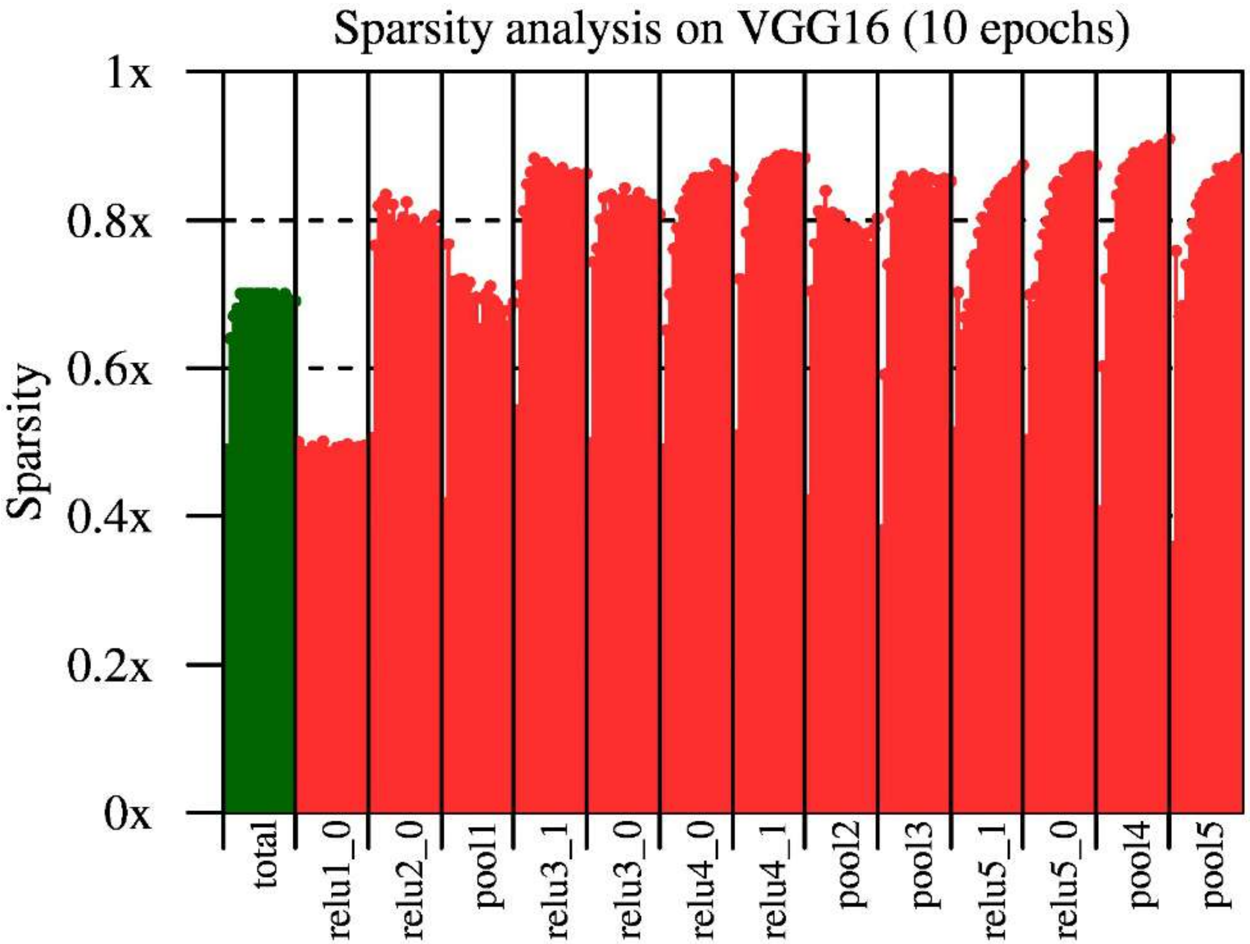


Binarize – 1 bit representation
(Lossless)

Relu/Pool -> Conv



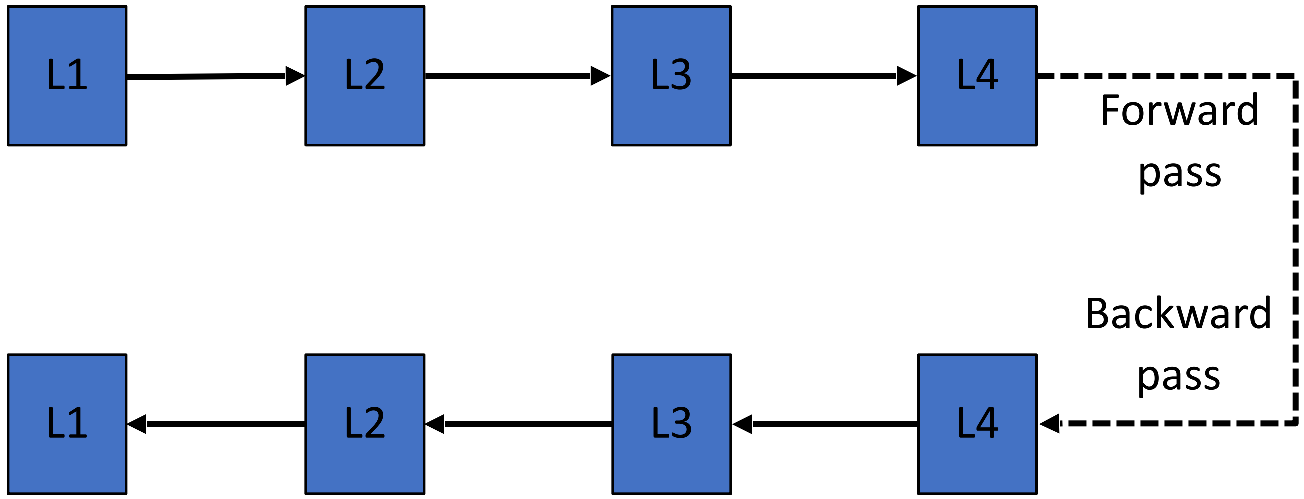
Relu/Pool -> Conv



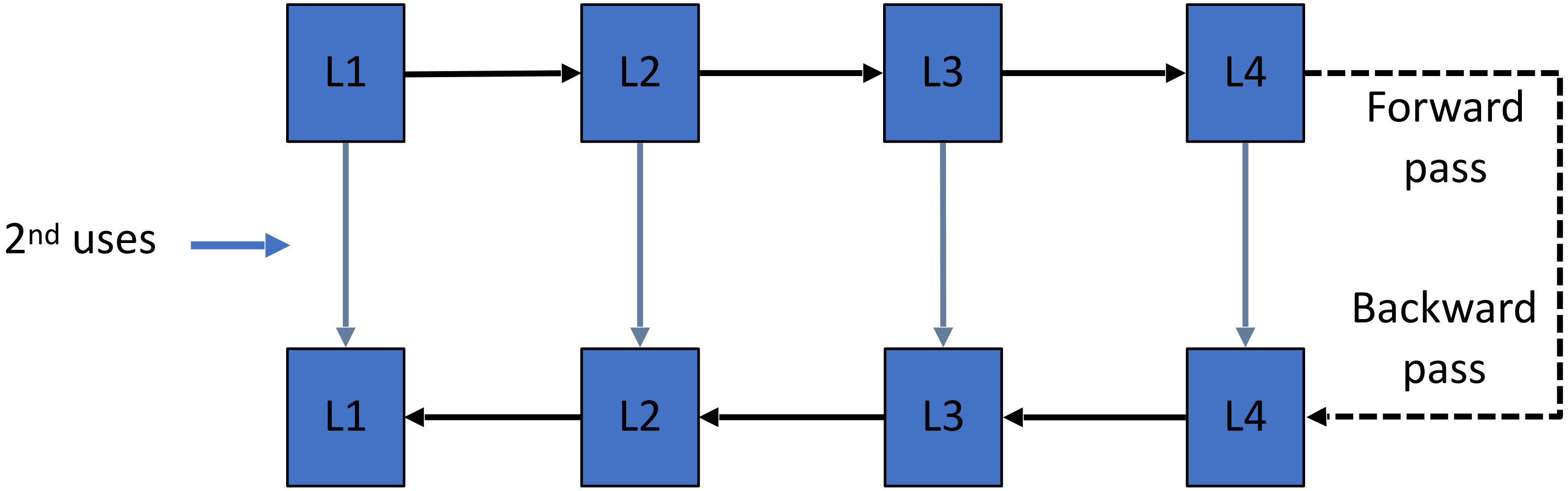
Sparse Storage Dense Compute

(Lossless)

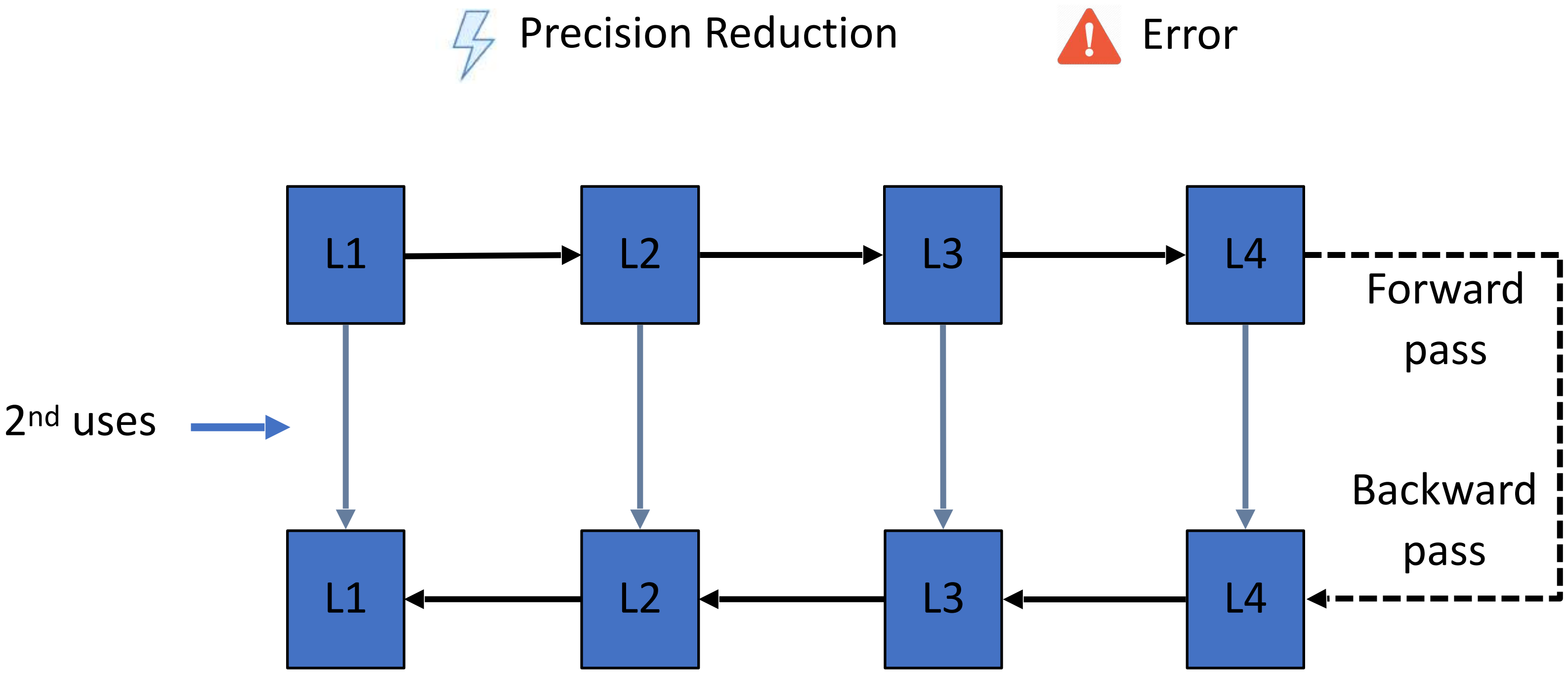
Opportunity for Lossy Encoding



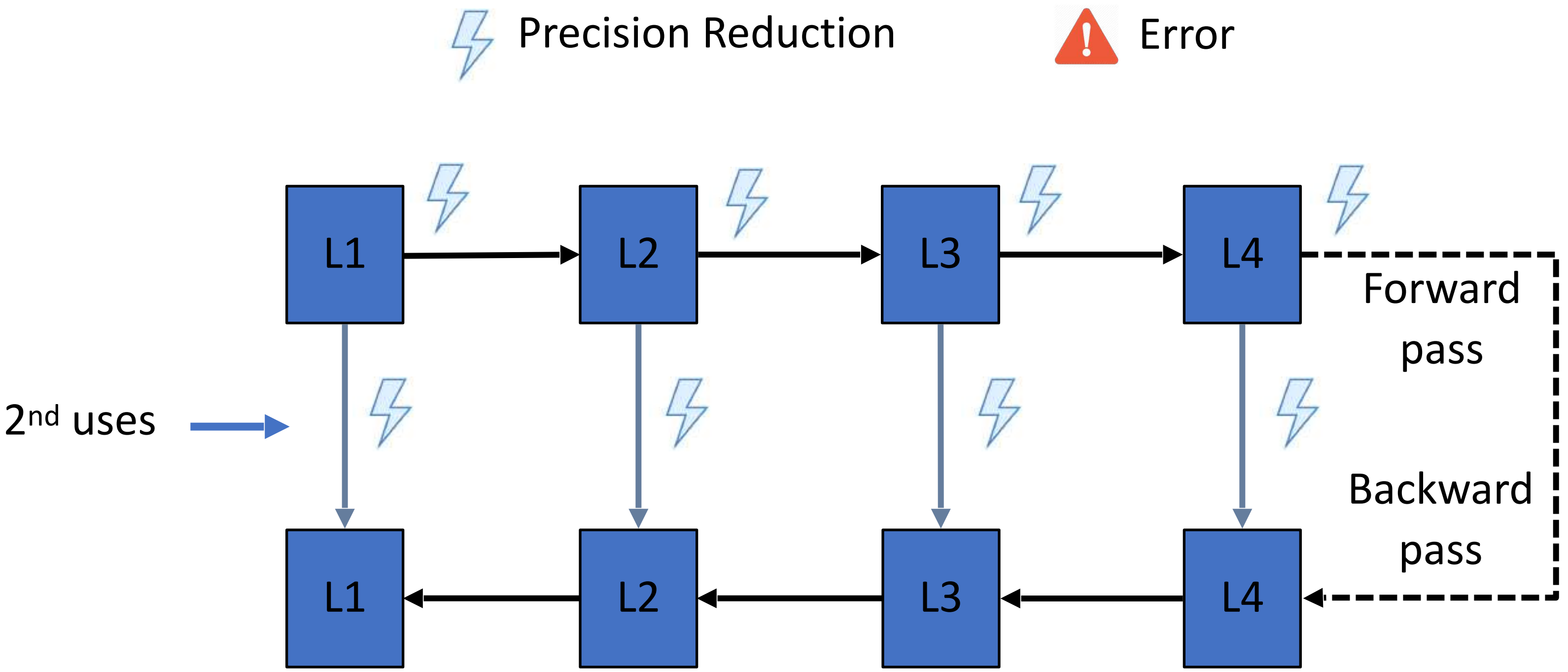
Opportunity for Lossy Encoding



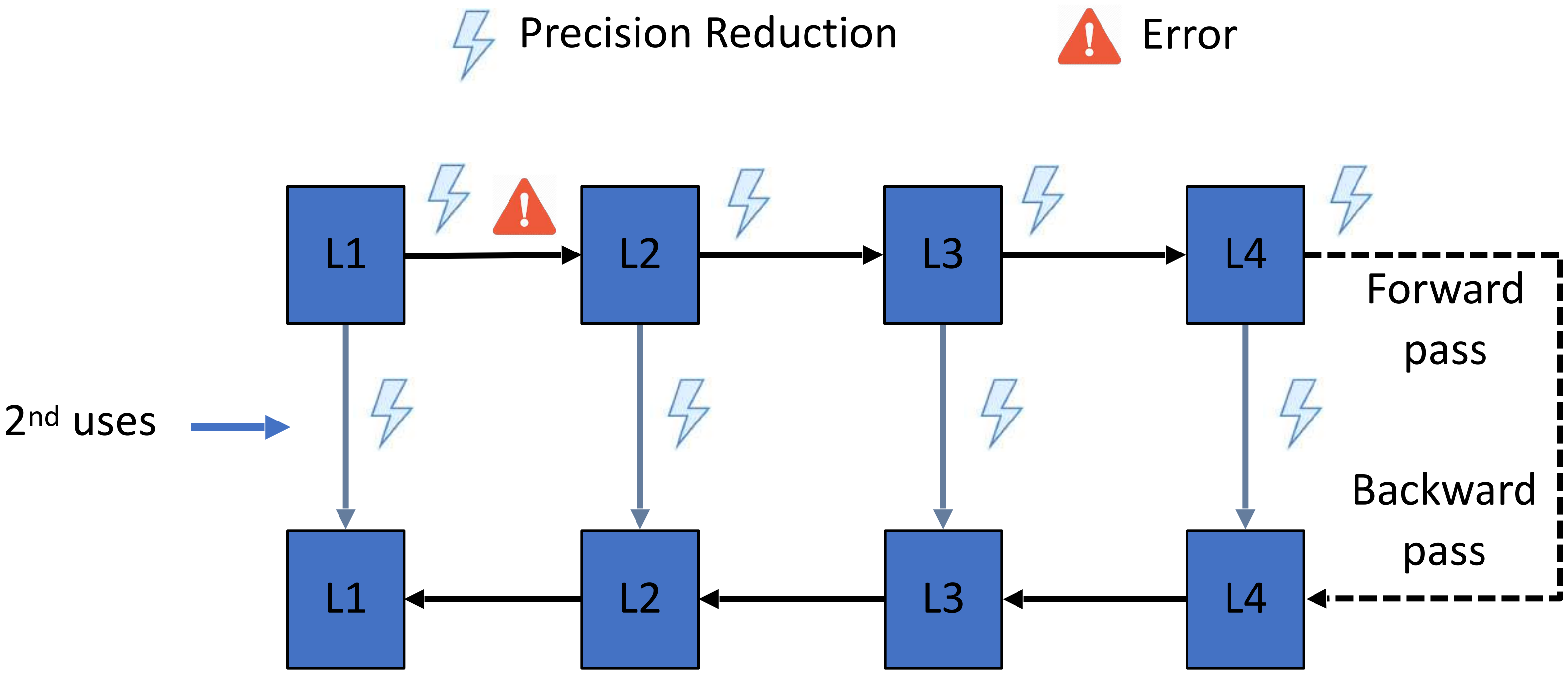
Opportunity for Lossy Encoding



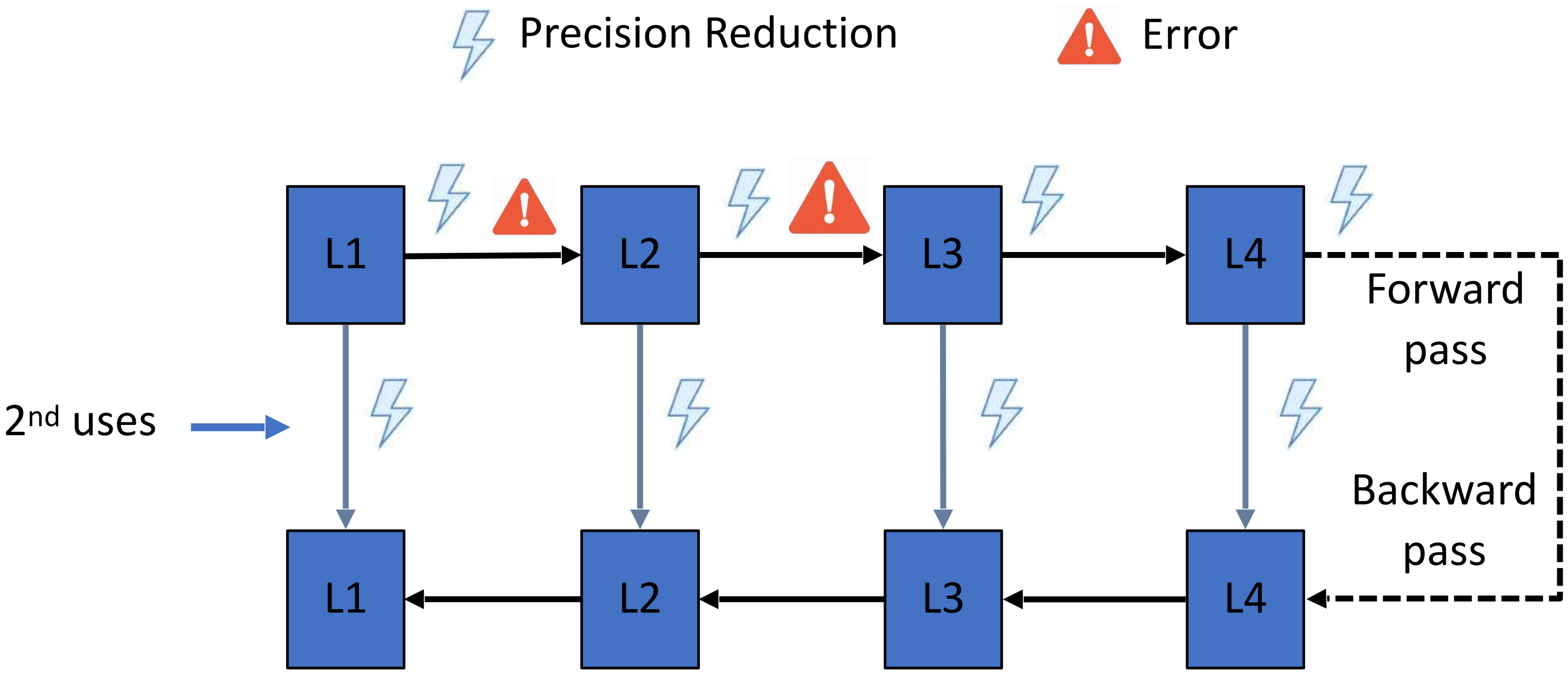
Opportunity for Lossy Encoding



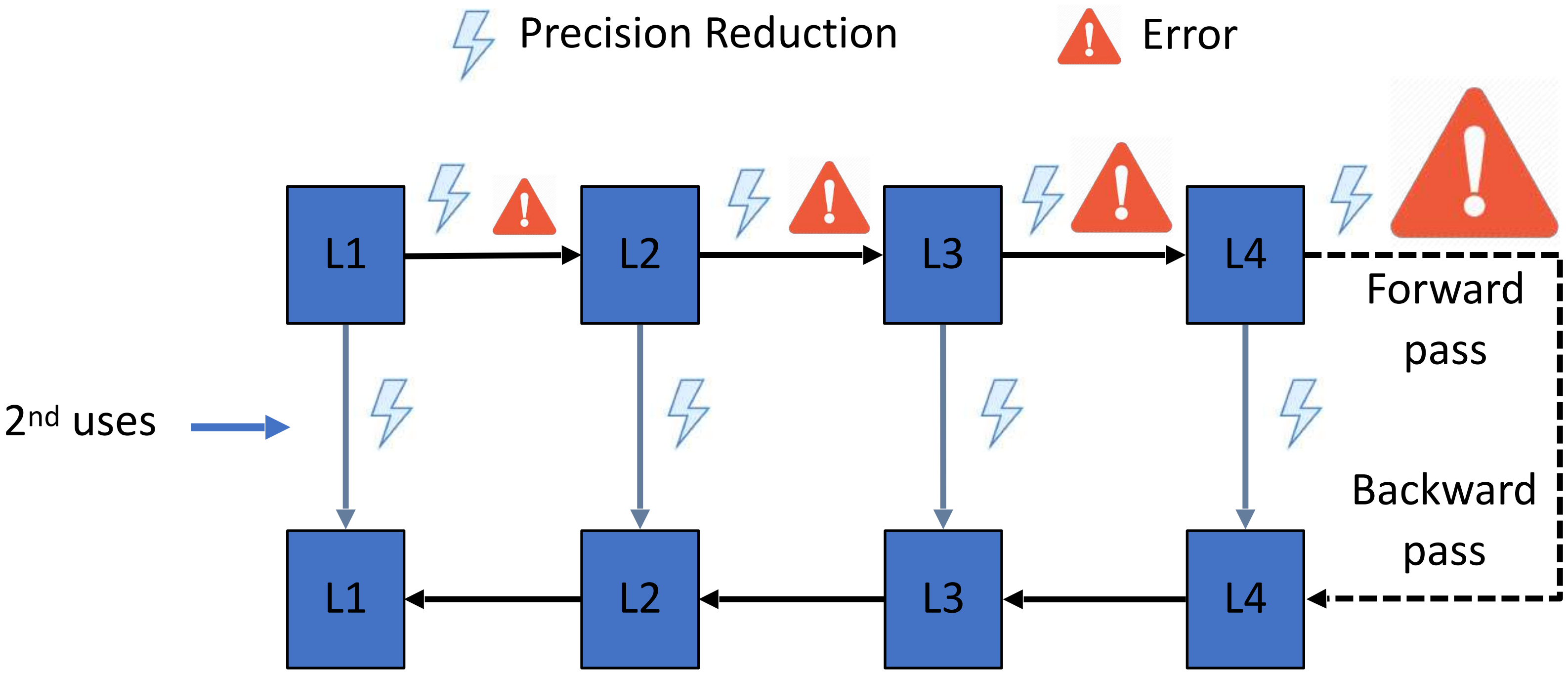
Opportunity for Lossy Encoding



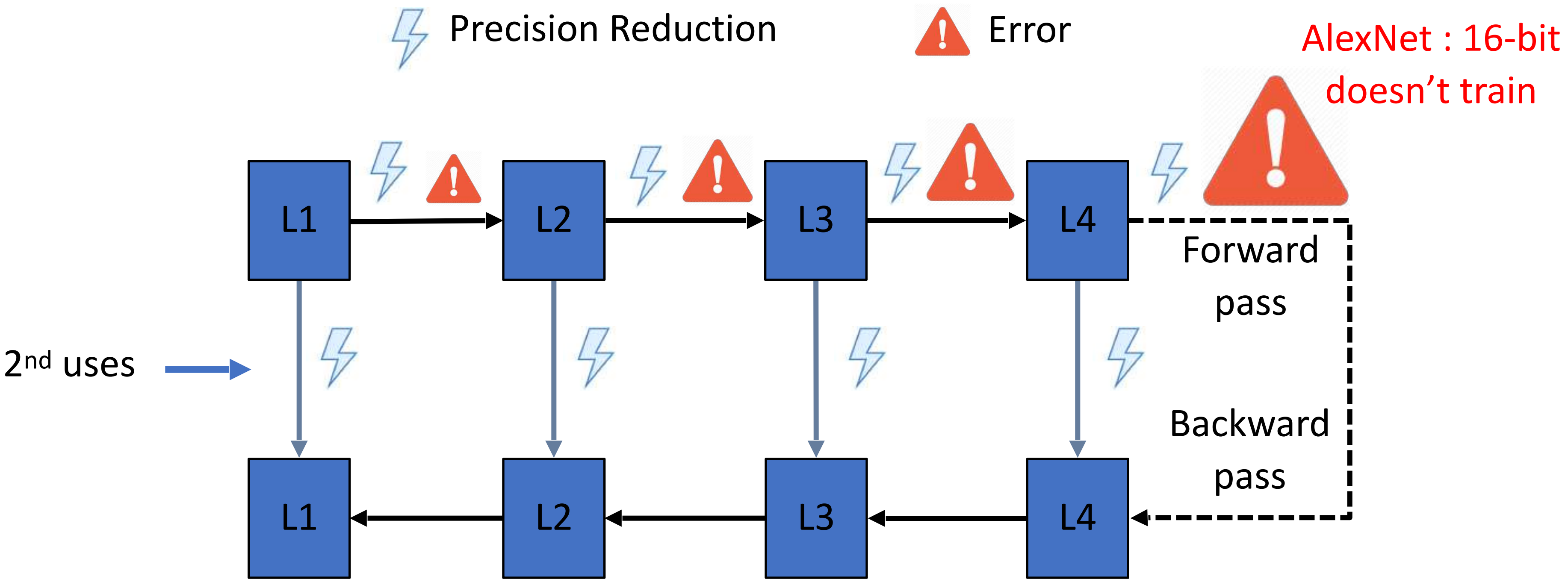
Opportunity for Lossy Encoding



Opportunity for Lossy Encoding



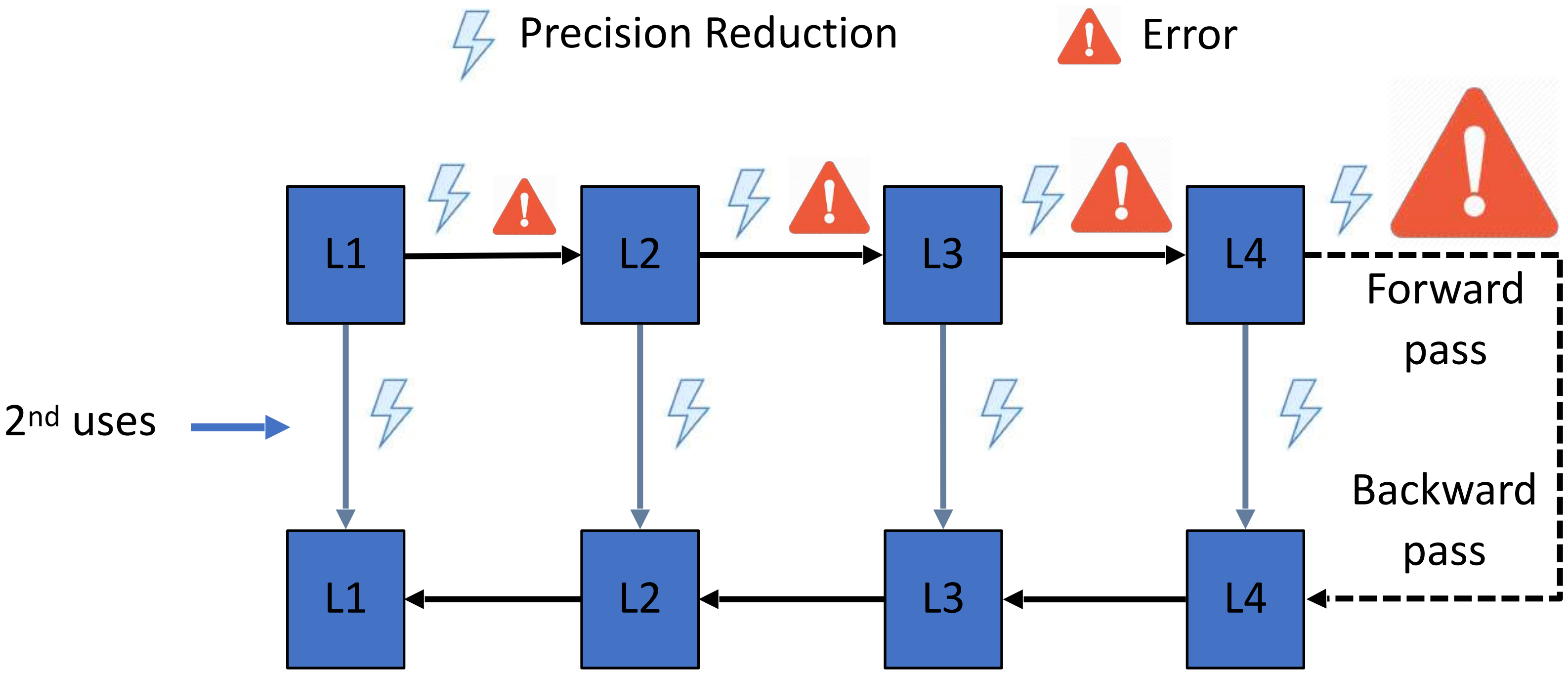
Opportunity for Lossy Encoding



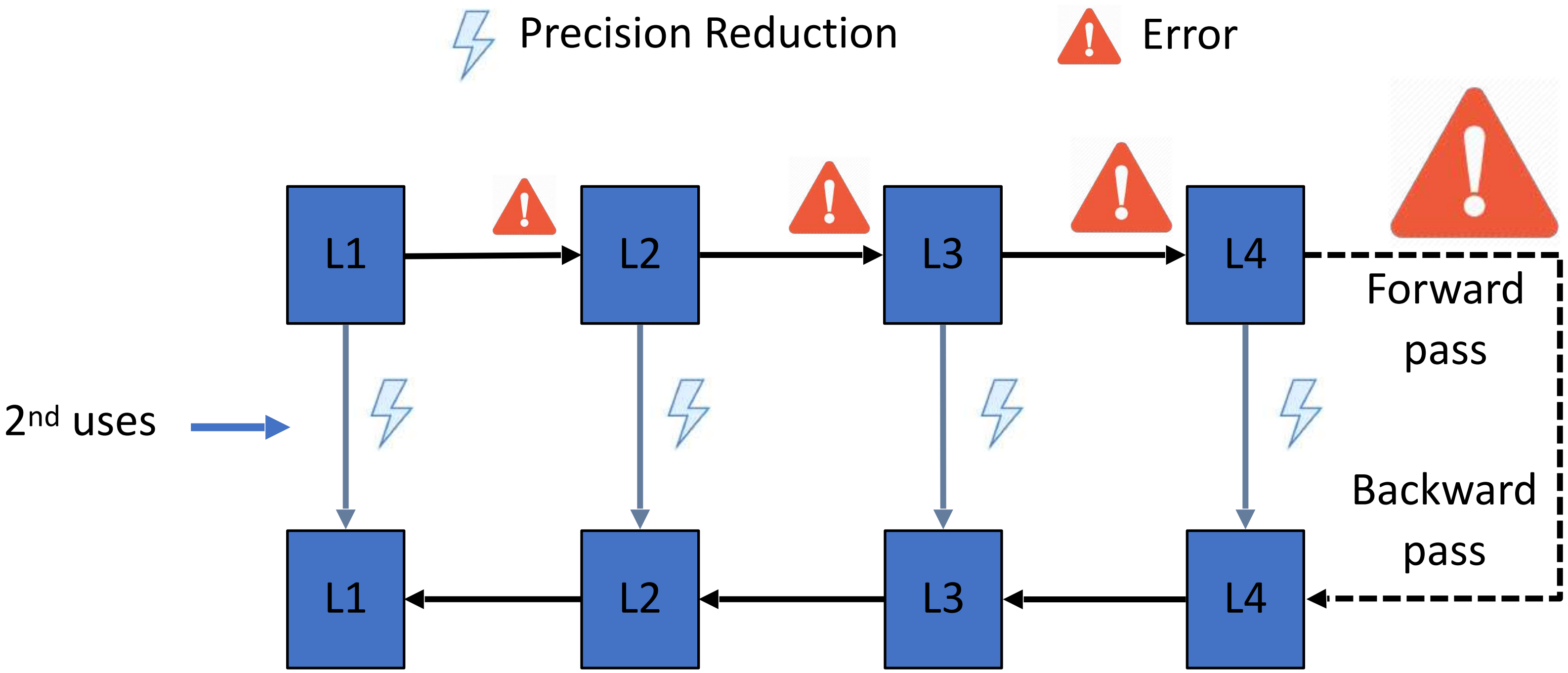
Opportunity for Lossy Encoding

Precision reduction in forward pass quickly degrades accuracy

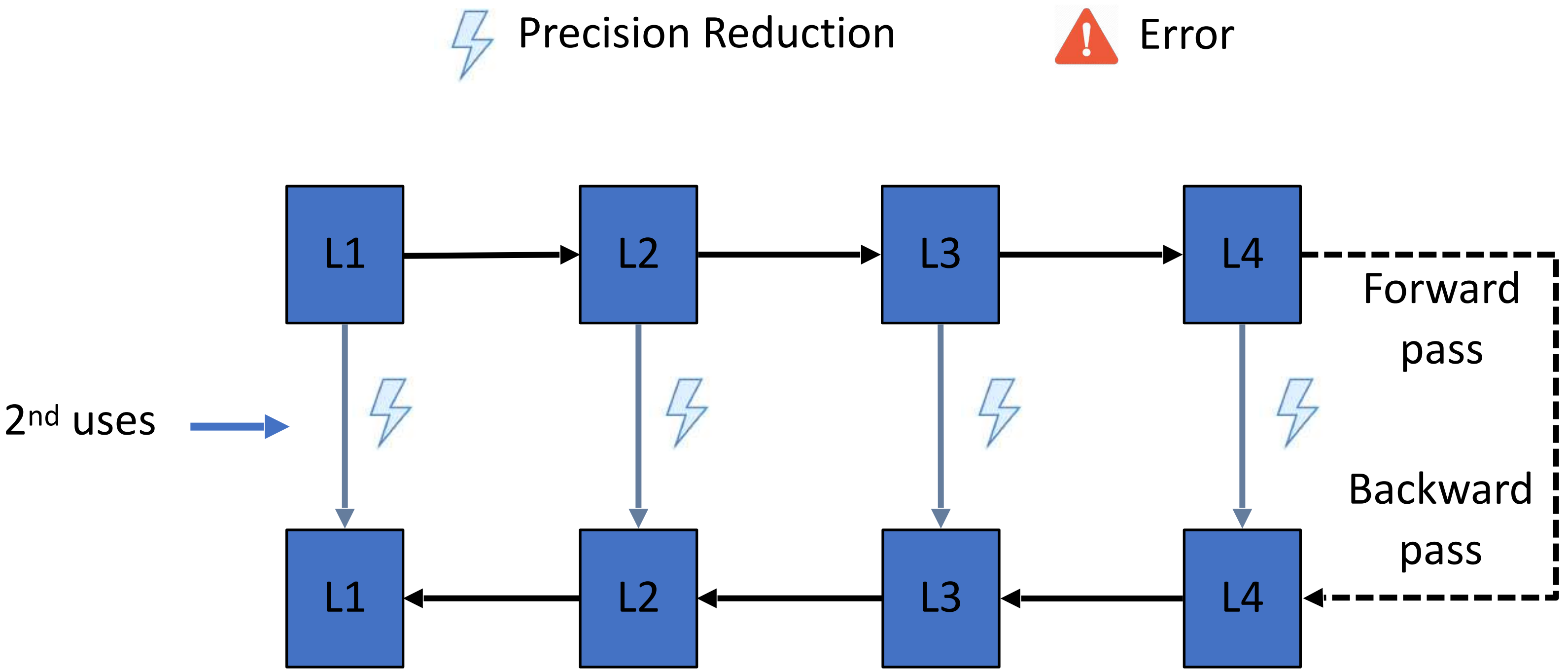
Opportunity for Lossy Encoding



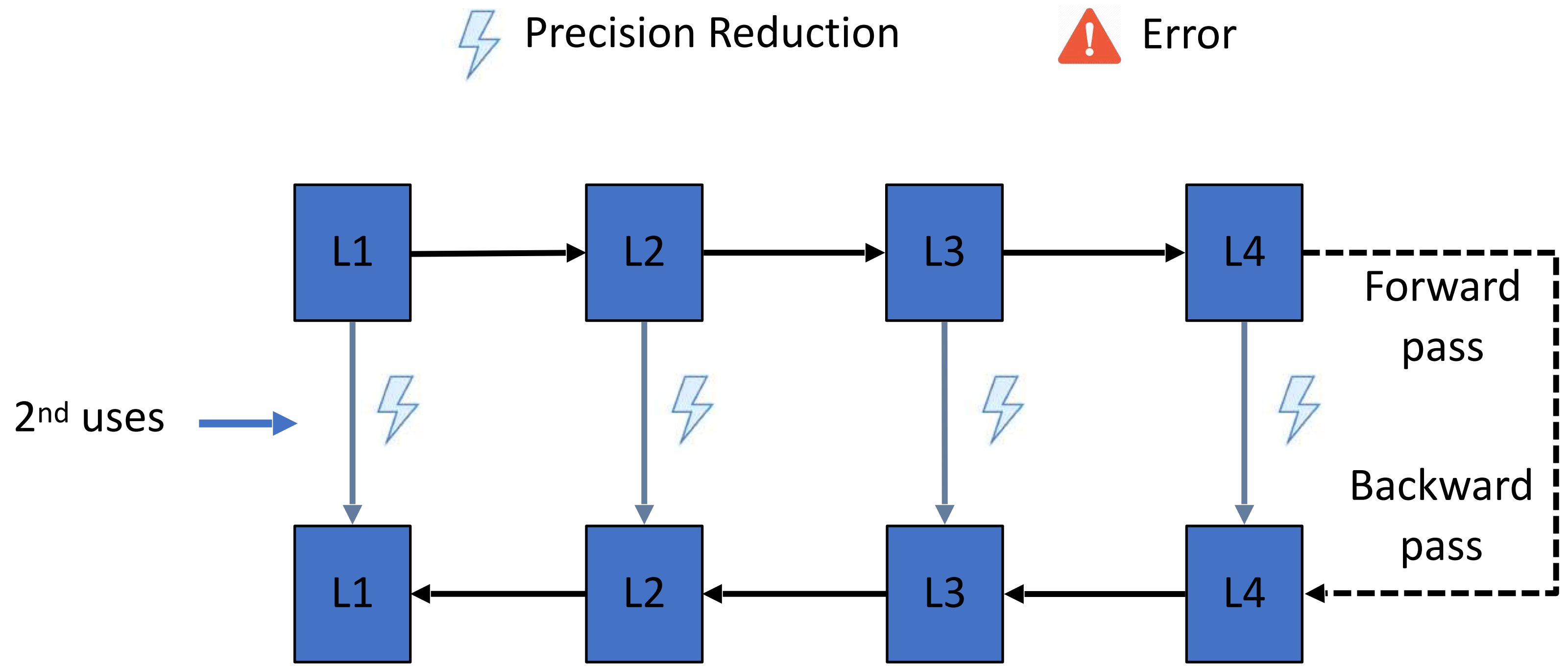
Opportunity for Lossy Encoding



Opportunity for Lossy Encoding



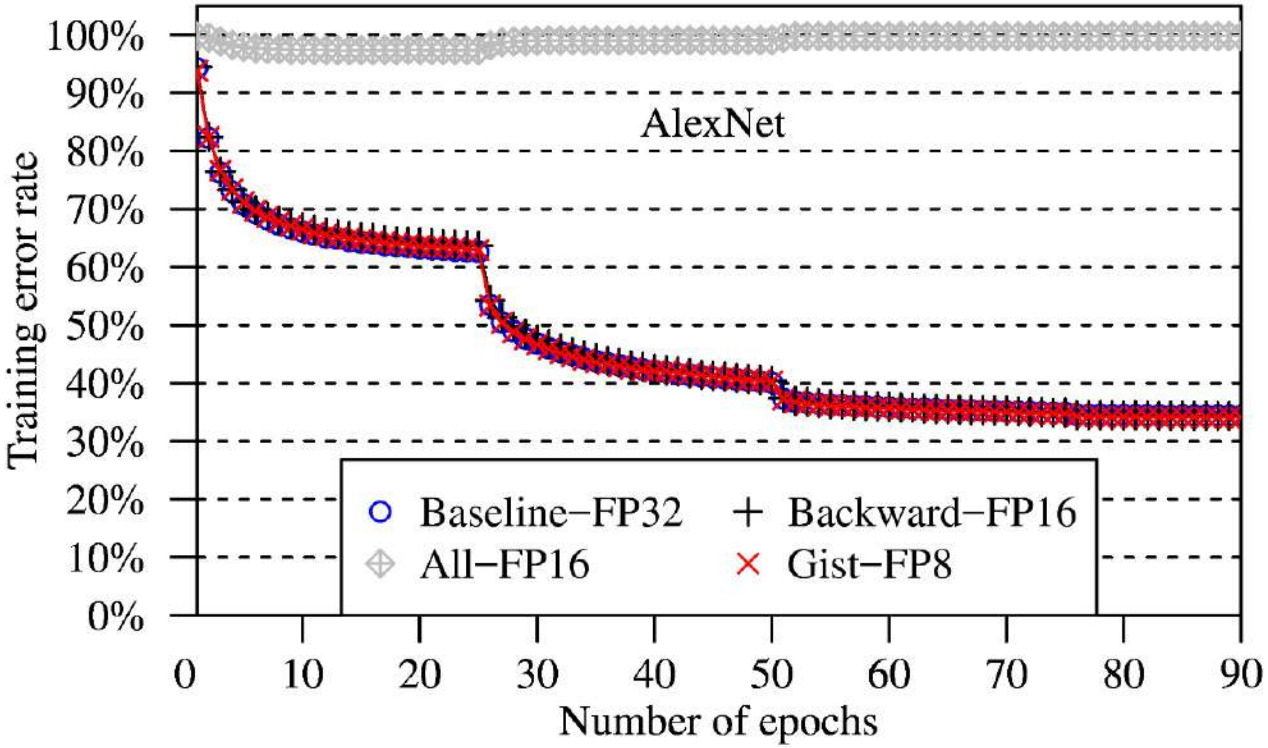
Opportunity for Lossy Encoding



Restricting precision reduction to the 2nd use results in aggressive bit savings with no effect on accuracy

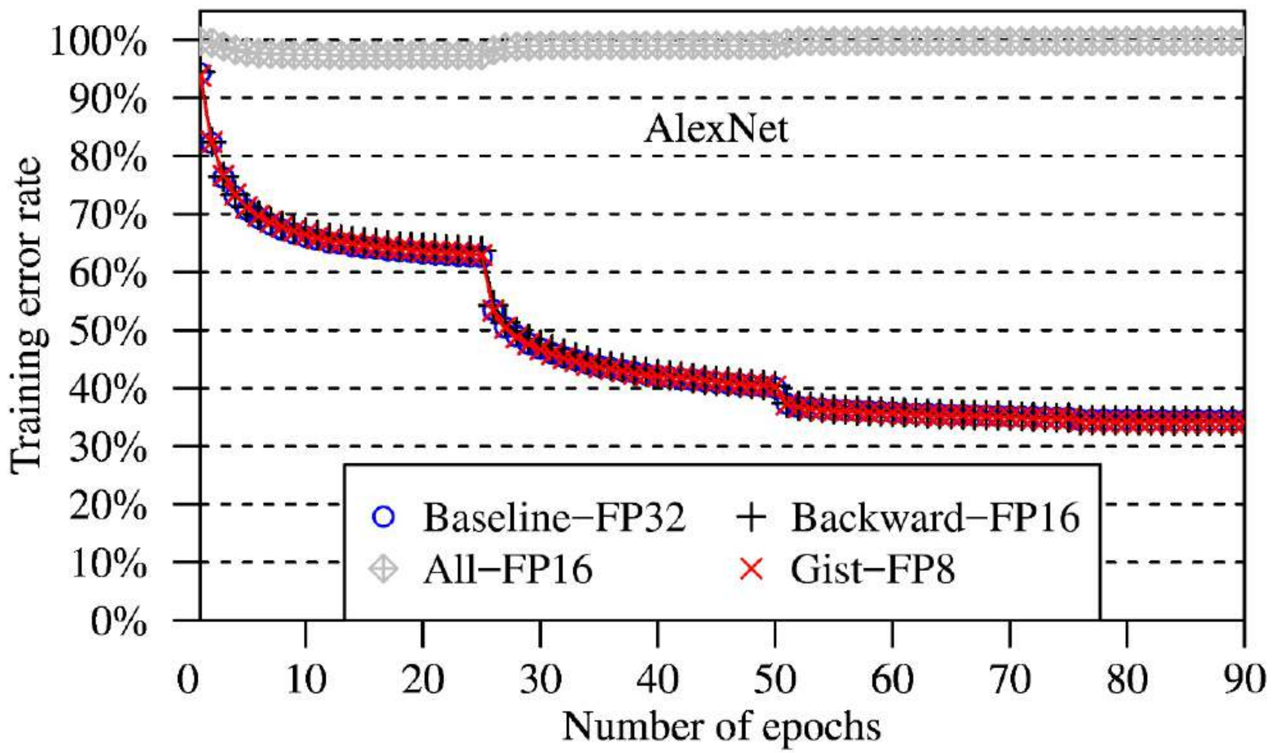
Delayed Precision Reduction

Training with Reduced Precision



Delayed Precision Reduction

Training with Reduced Precision



<u>Delayed Precision Reduction</u>

(Lossy)

Proposed System Architecture - Gist

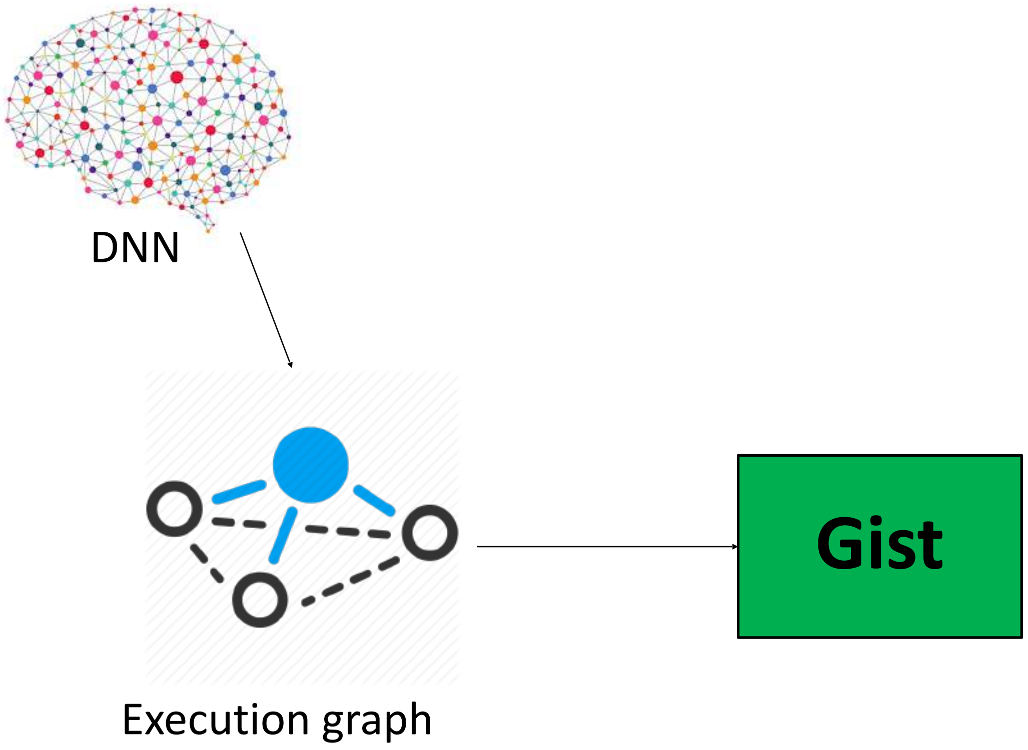
DNN

Proposed System Architecture - Gist

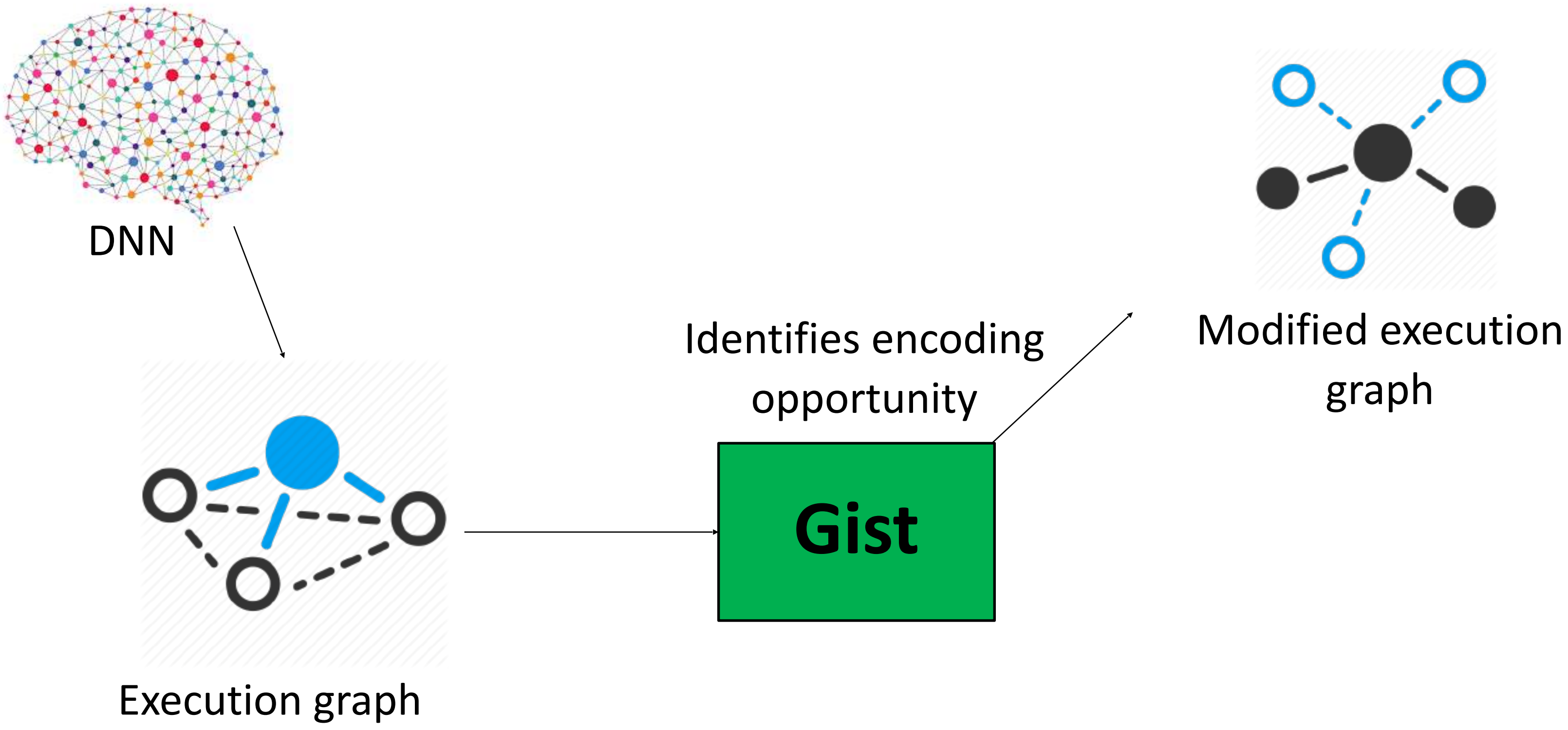
DNN

Execution graph

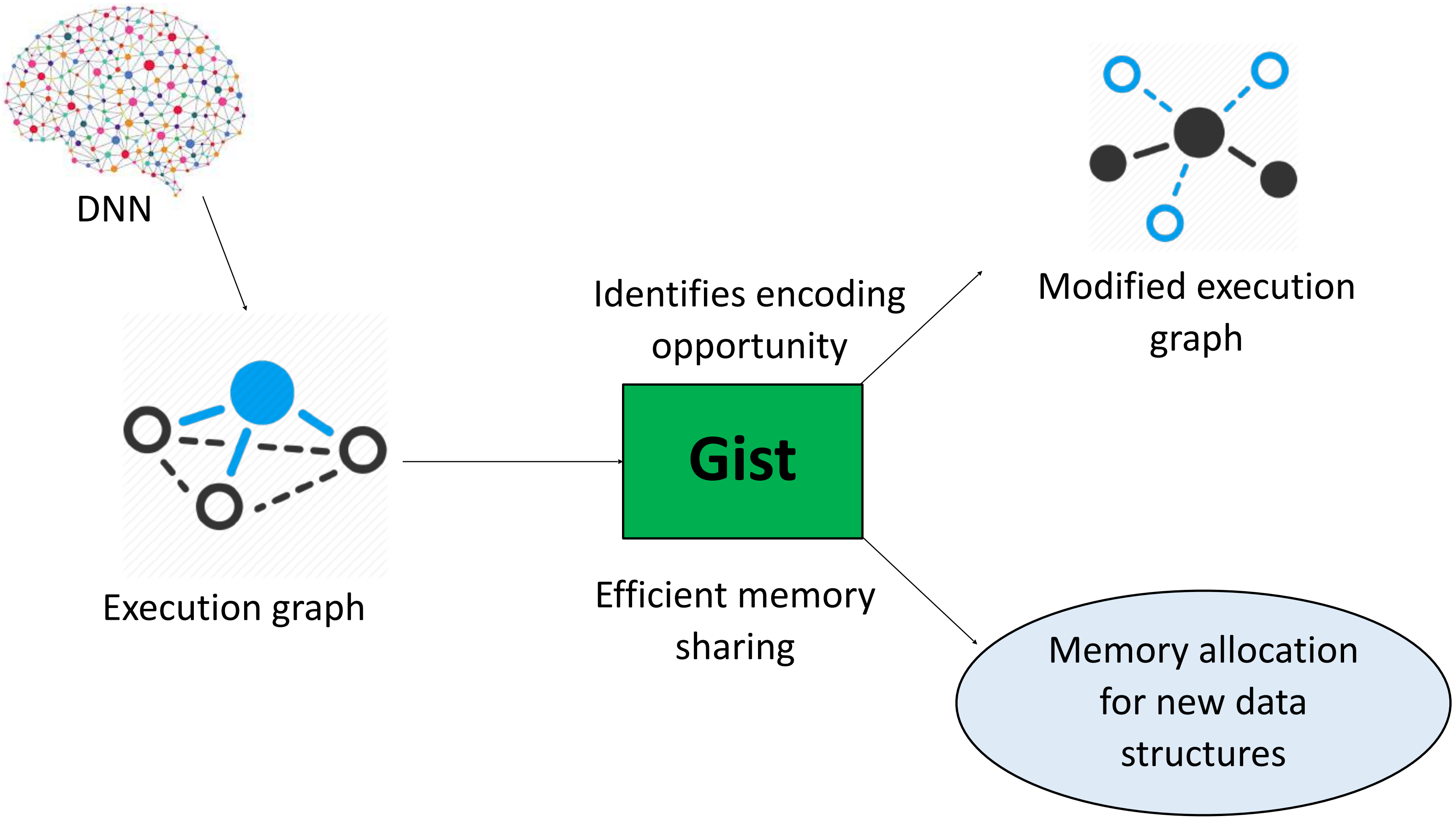
Proposed System Architecture - Gist



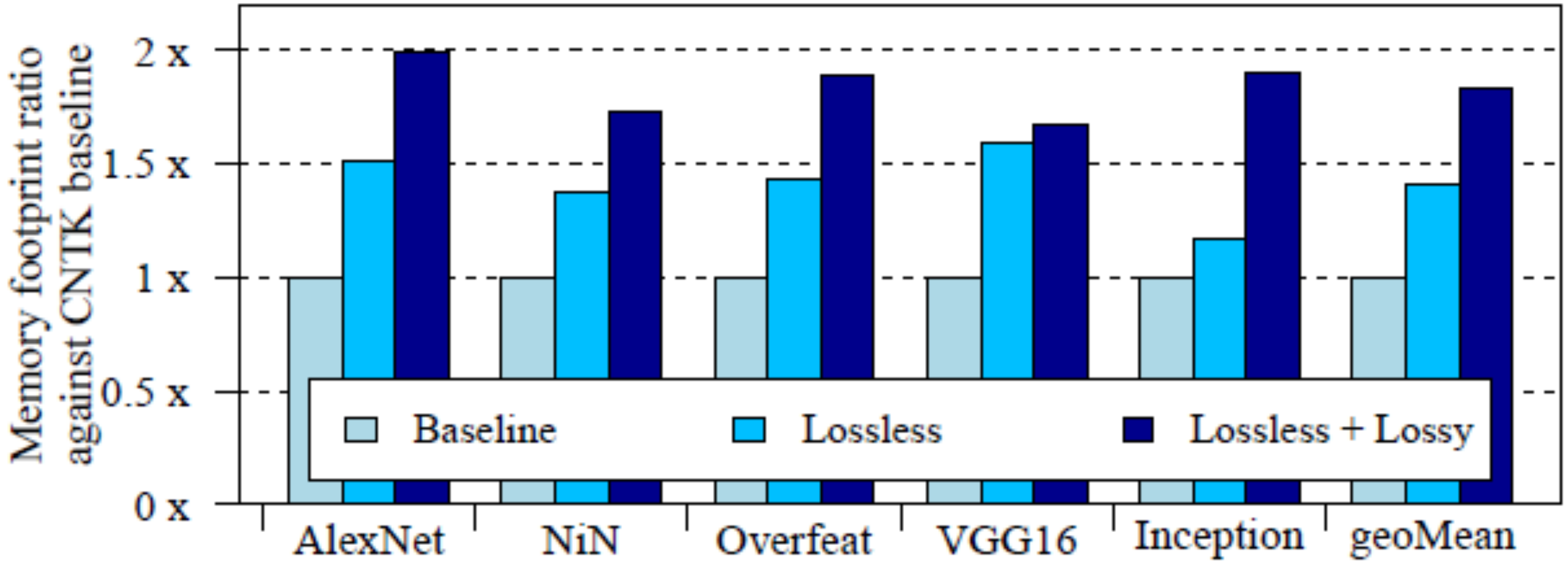
Proposed System Architecture - Gist



Proposed System Architecture - Gist



Compression Ratio



Up to 2X compression ratio

With minimal performance overhead

Gist Summary

- Systematic memory breakdown analysis for image classification
- Layer-specific **lossless** encodings
  - Binarization and sparse storage/dense compute
- Aggressive lossy encodings
  - With delayed precision reduction
- Footprint reduction measured on real systems:
  - Up to **2X** reduction with only 4% performance overhead
  - Further optimizations – more than **4X** reduction

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

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

1 Computer Science UNIVERSITY OF TORONTO

2 VECTOR INSTITUTE

ISCA 2020

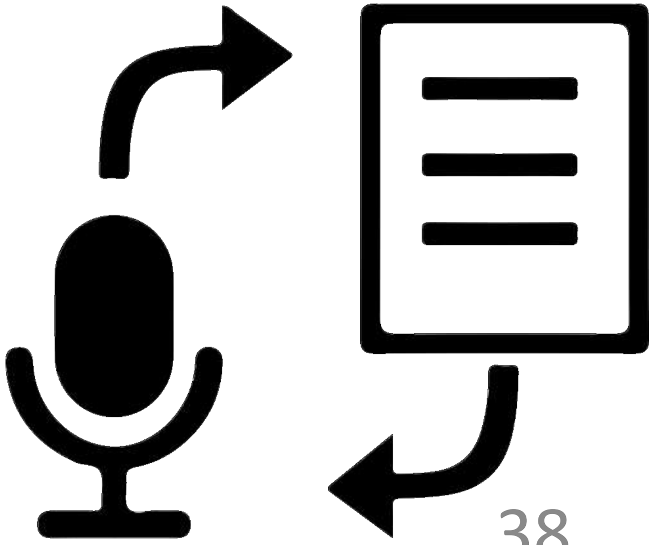
Background: LSTM RNN

Neural Machine Translation (NMT)

- Long-Short-Term-Memory Recurrent Neural Network (LSTM RNN)
- Heavily adopted in sequence analysis (e.g., machine translation (NMT) & speech recognition (DeepSpeech2).
- Its **training** is **inefficient** on the **GPUs**, especially when compared with CNN.[1, 2]

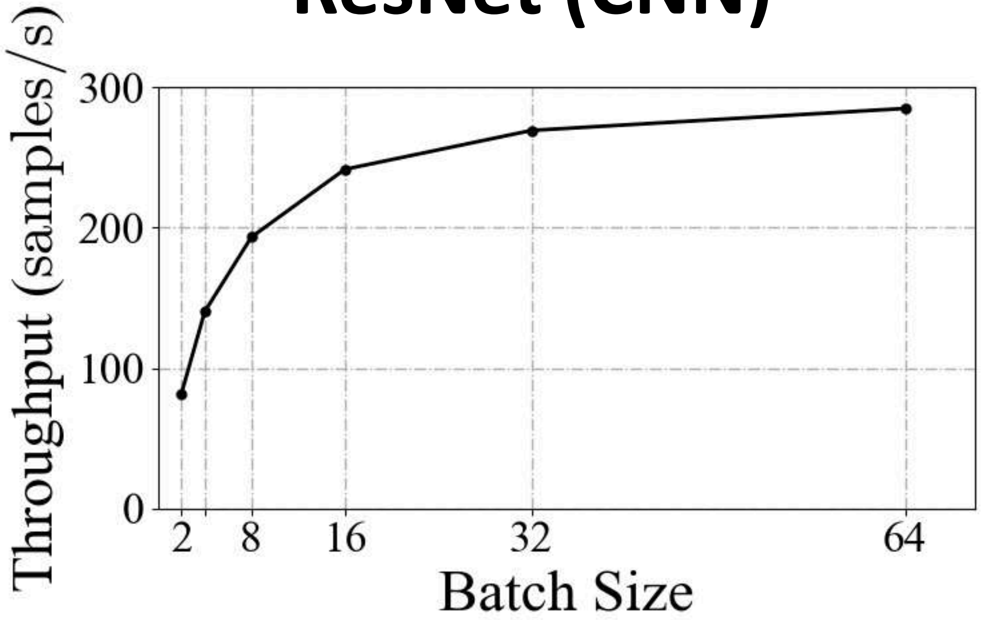
[1] J. Bradbury et al. *Quasi-Recurrent Neural Networks*. ICLR 2016
[2] T. Lei et al. *Simple Recurrent Units for Highly Parallelizable Recurrence*. EMNLP 2018

DeepSpeech2



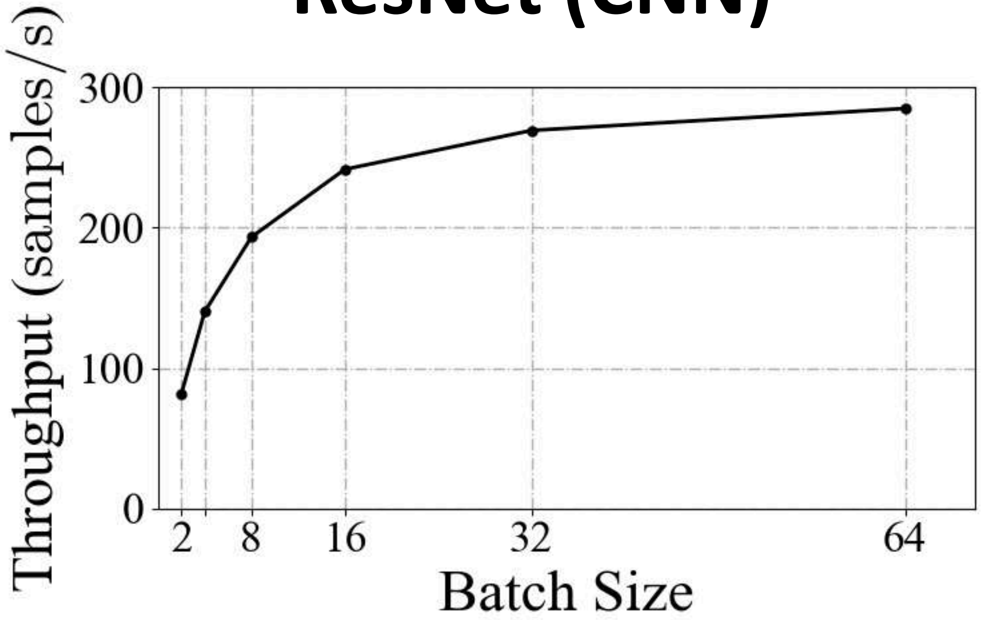
Why LSTM RNN Training is Inefficient?

ResNet (CNN)

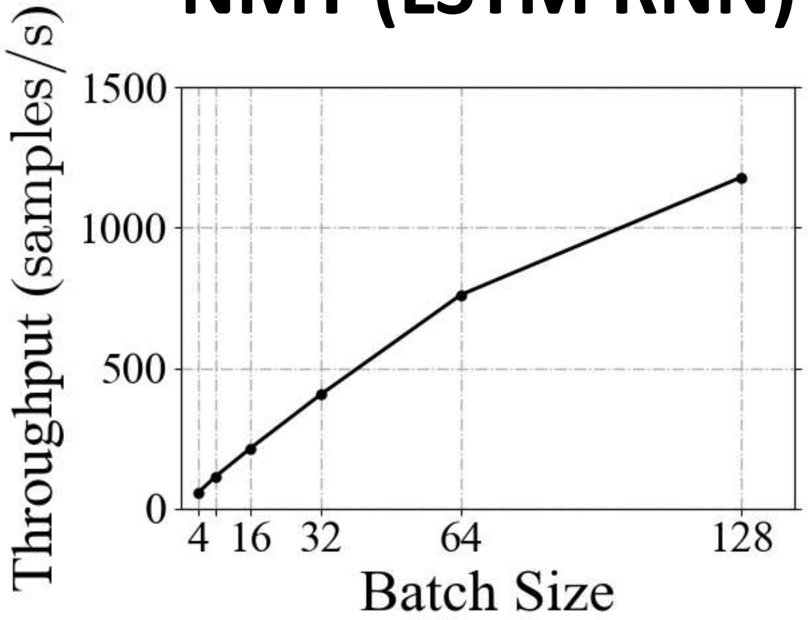


Why LSTM RNN Training is Inefficient?

ResNet (CNN)

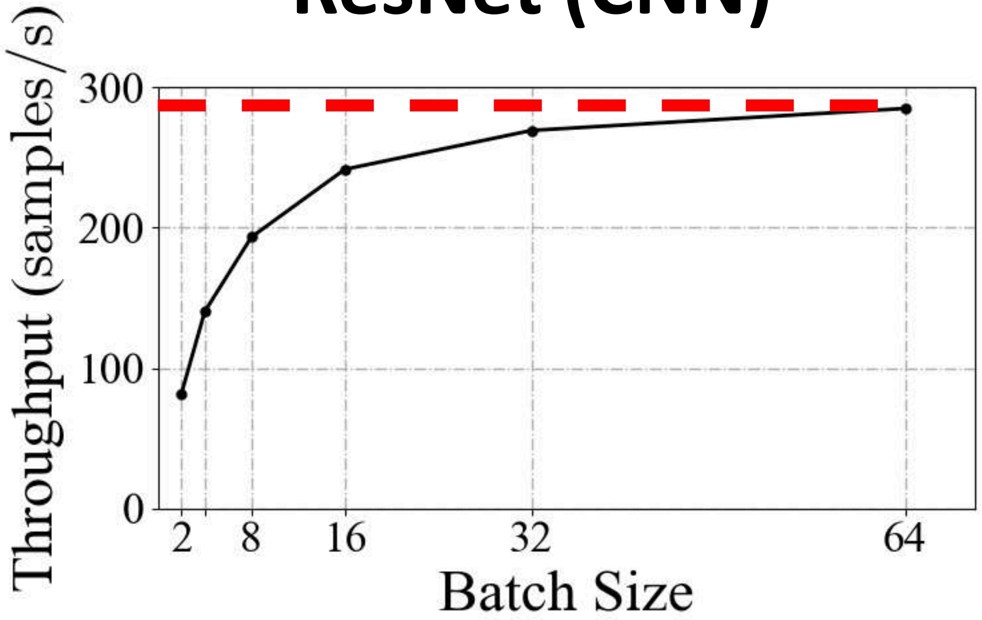


NMT (LSTM RNN)

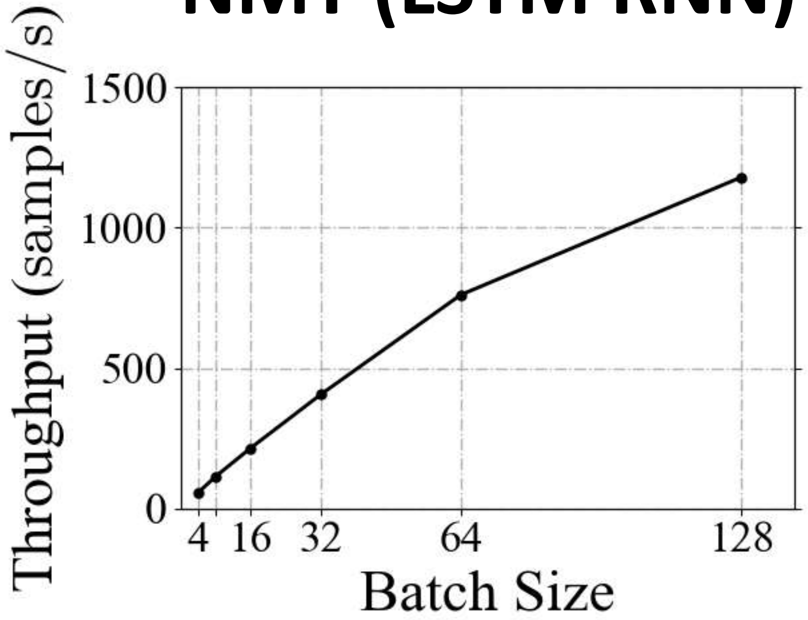


Why LSTM RNN Training is Inefficient?

ResNet (CNN)



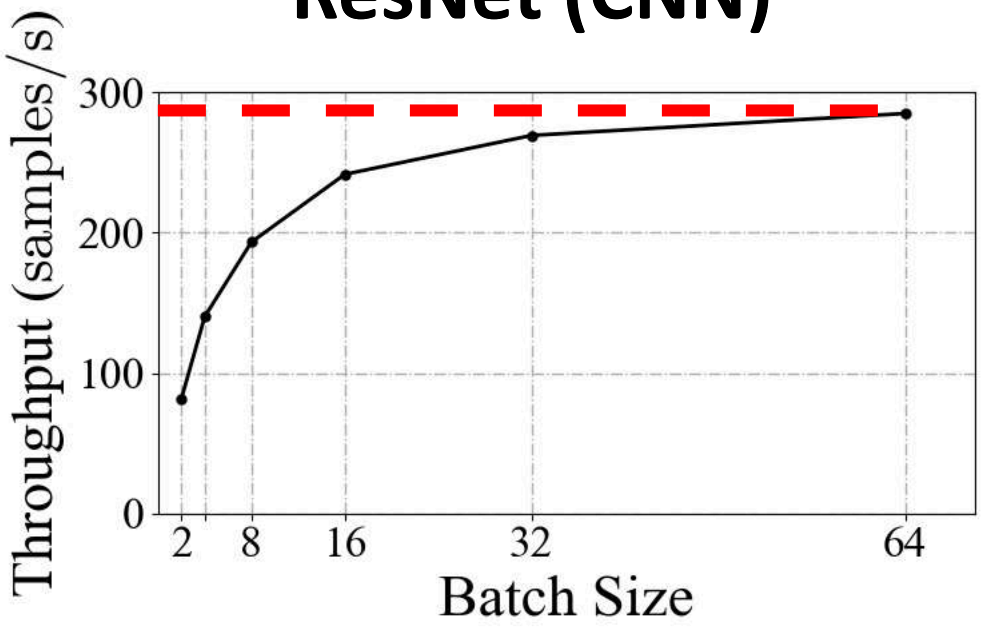
NMT (LSTM RNN)



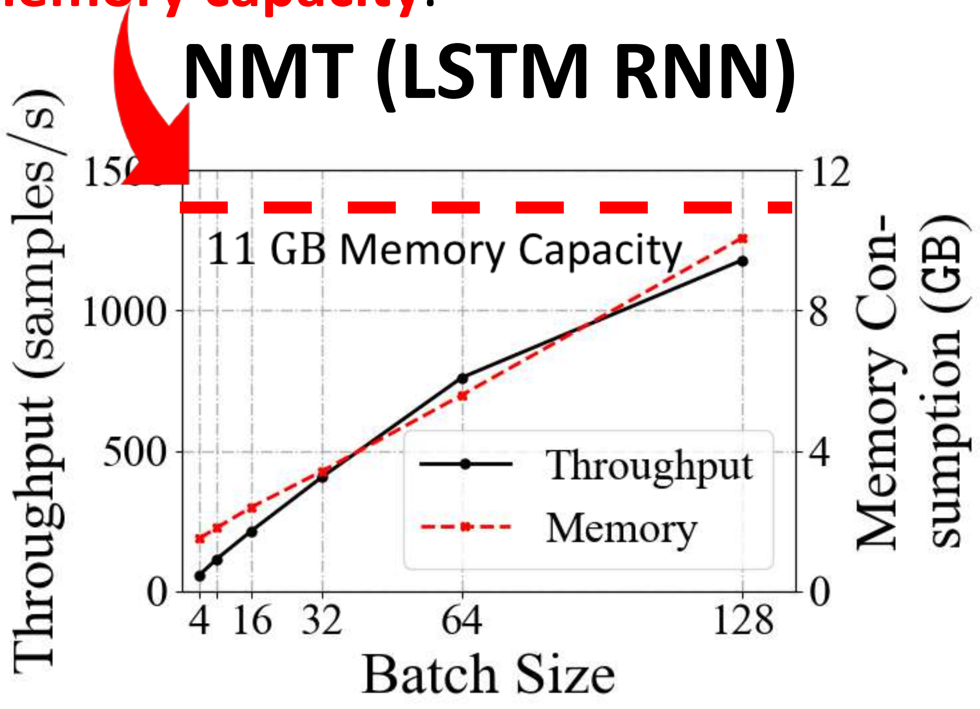
Why LSTM RNN Training is Inefficient?

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

ResNet (CNN)



NMT (LSTM RNN)

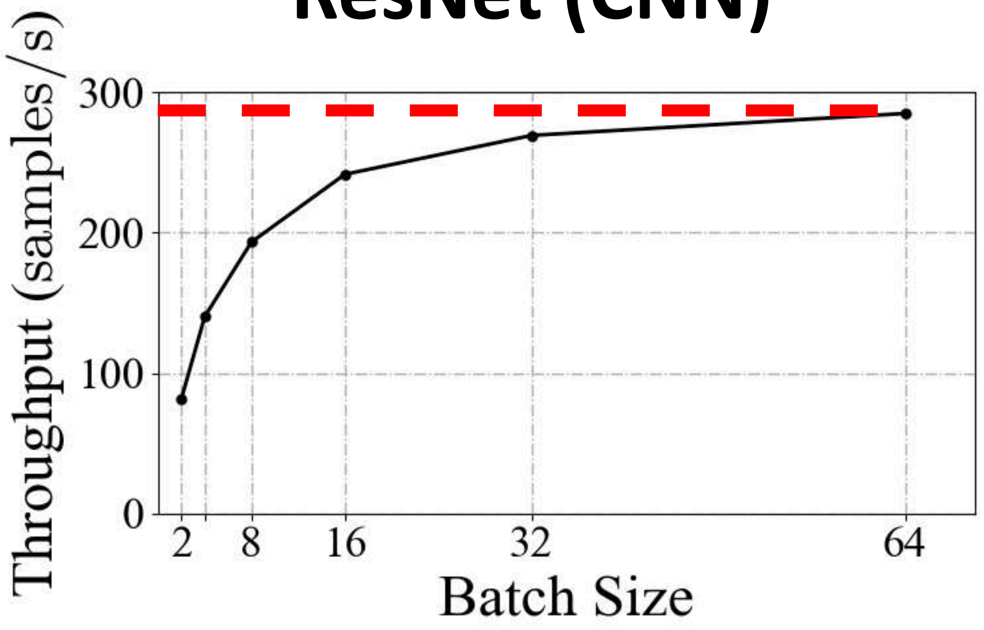


Why LSTM RNN Training is Inefficient?

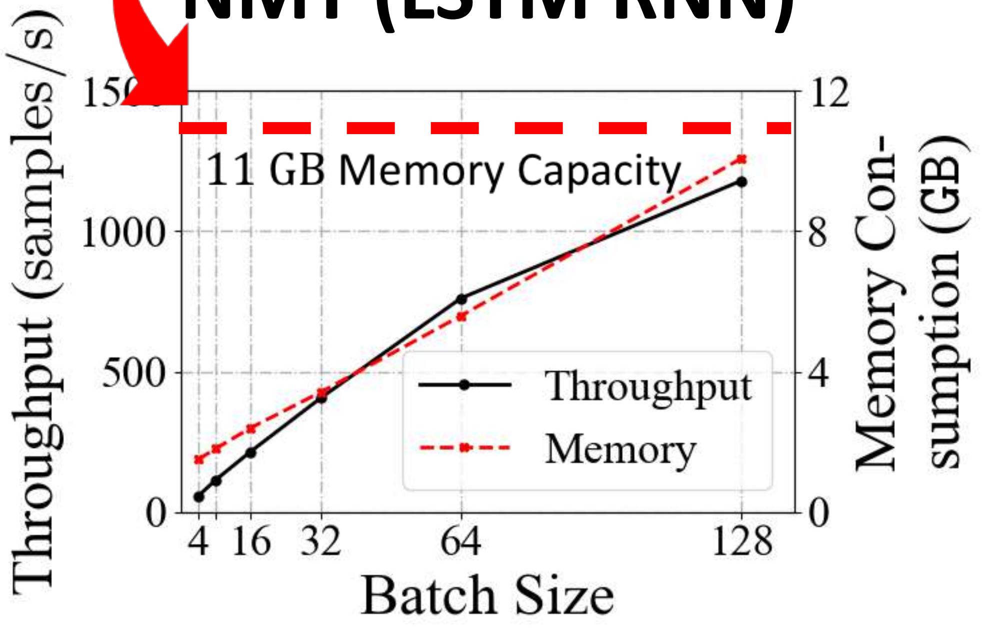
Training throughput **saturates** as batch size increases.

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

ResNet (CNN)



NMT (LSTM RNN)



Memory capacity limits the NMT training throughput

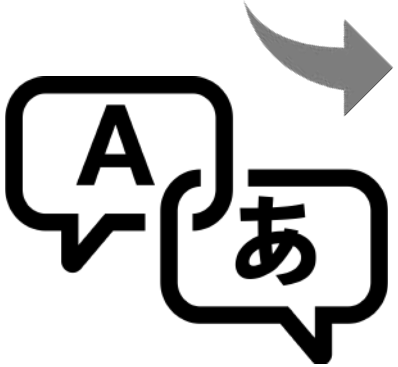
GPU Memory Profiling Results

MXNet GPU Memory Profiler

87%

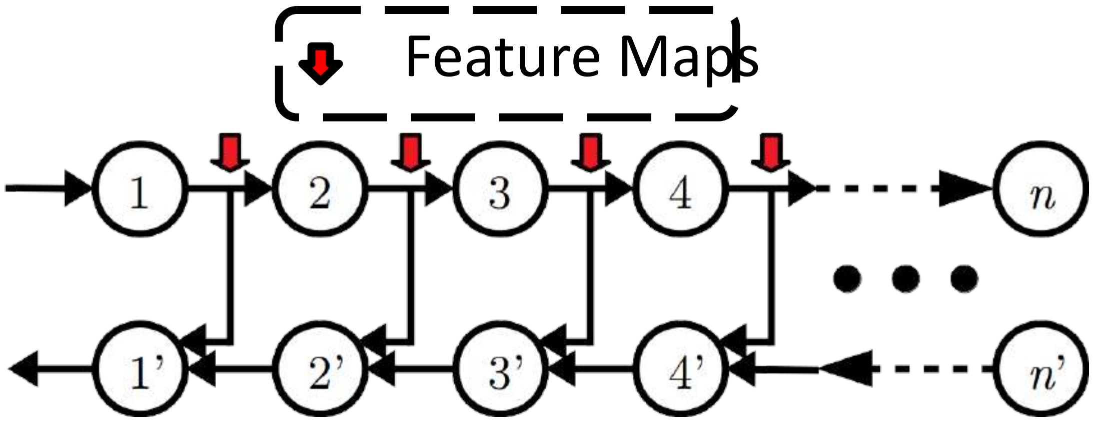
GPU Memory Profiling Results

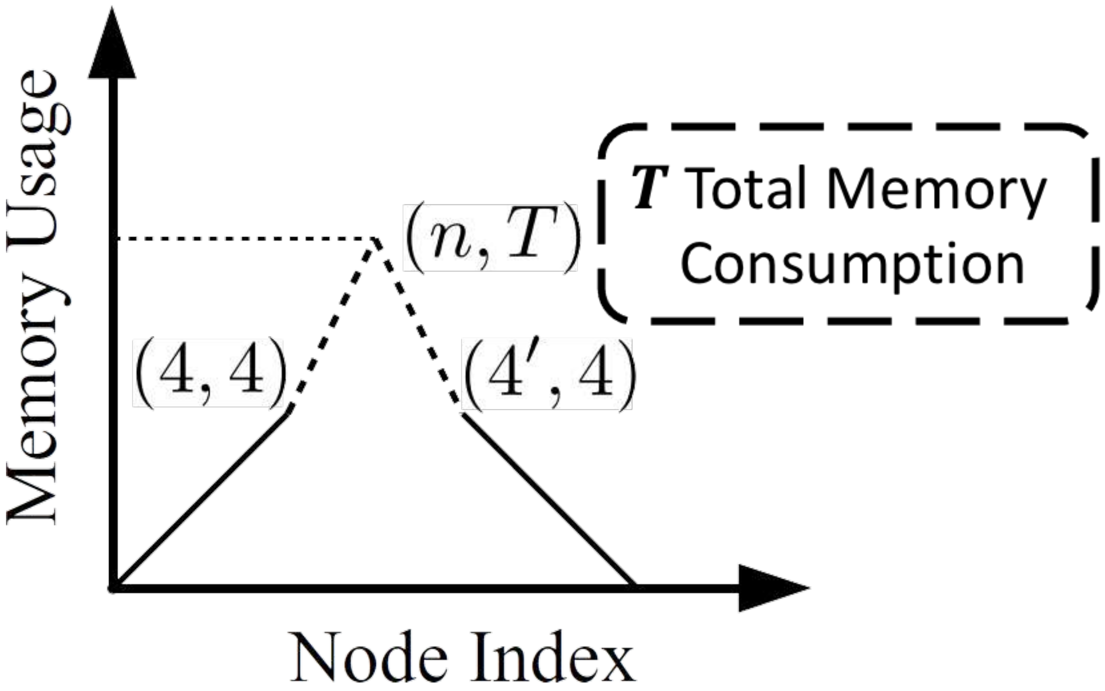
MXNet GPU Memory Profiler



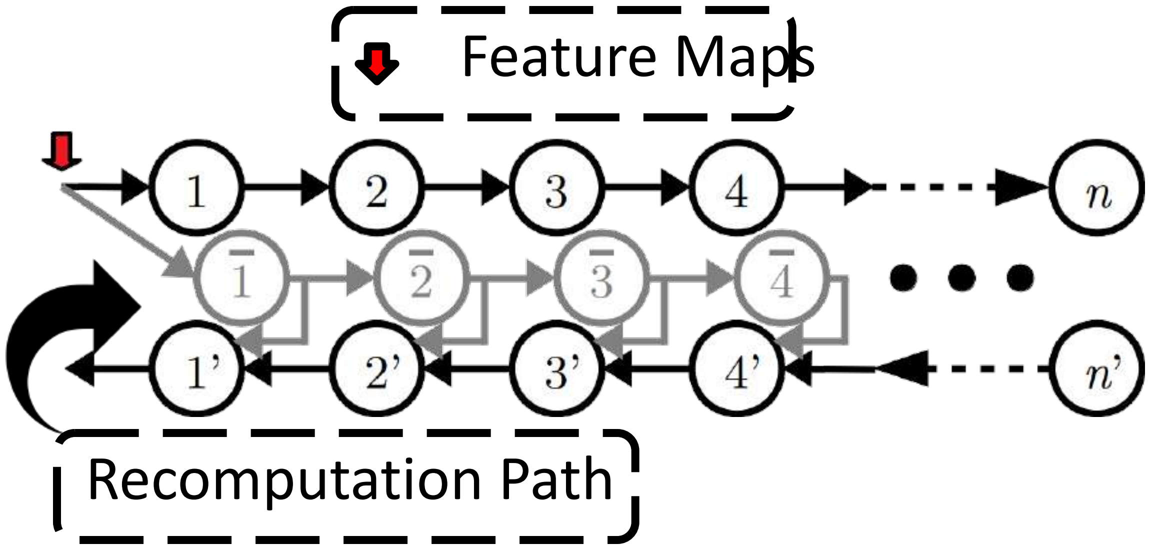
Feature maps dominate the GPU memory footprint.

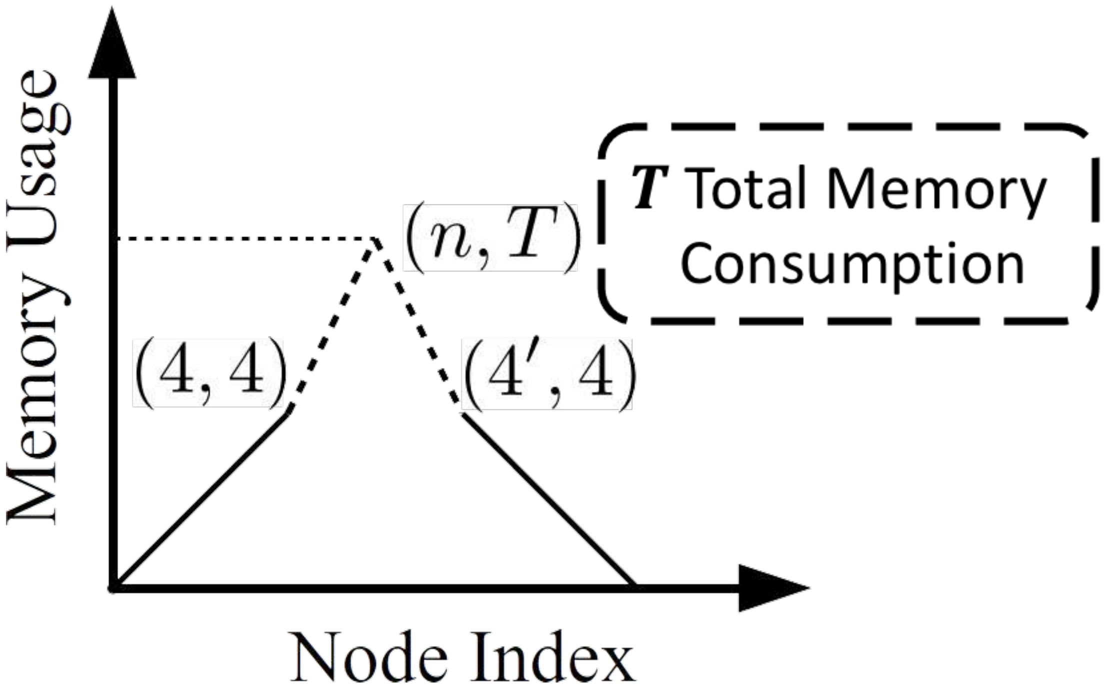
Selective Recomputation



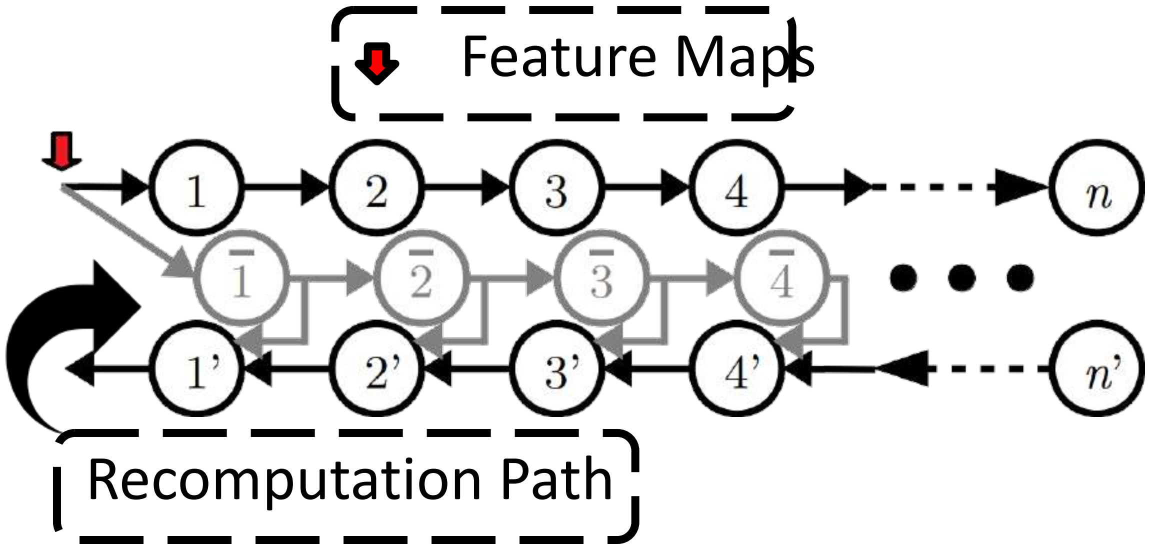


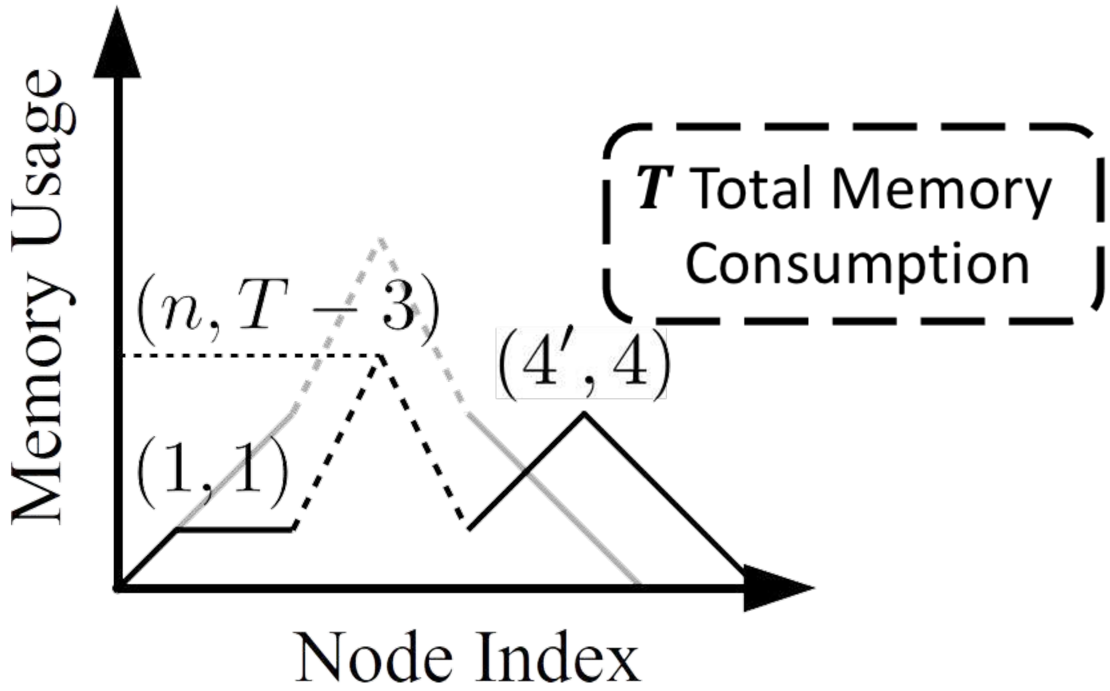
Selective Recomputation



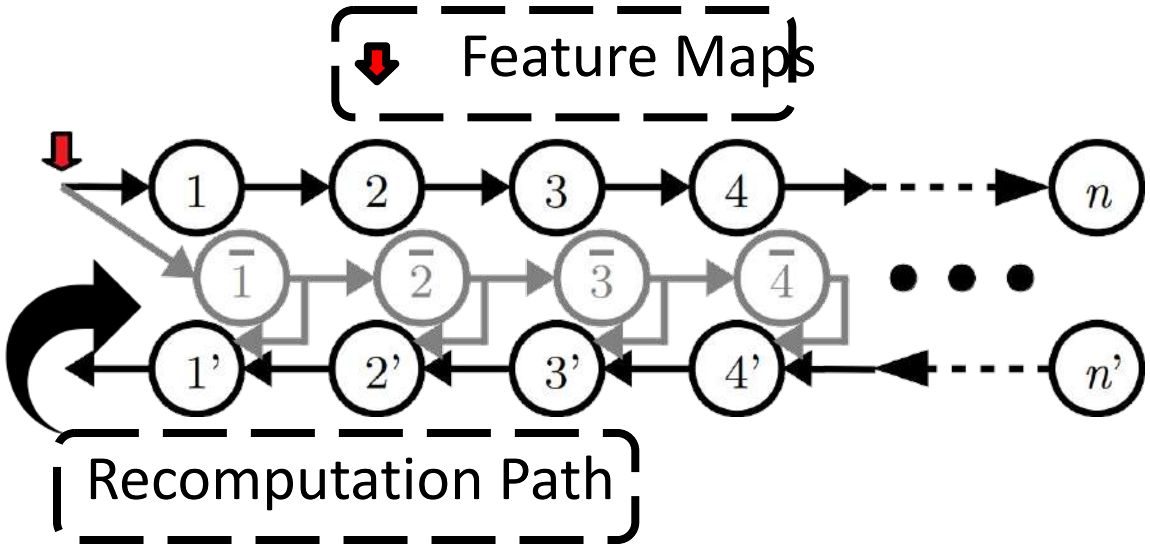


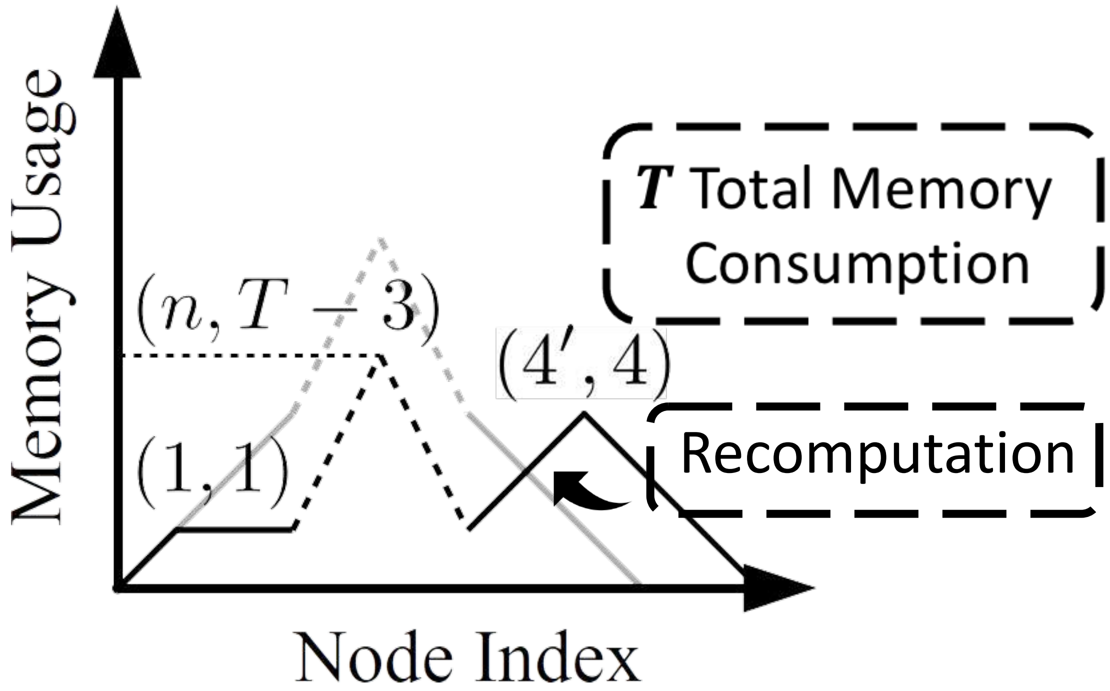
Selective Recomputation





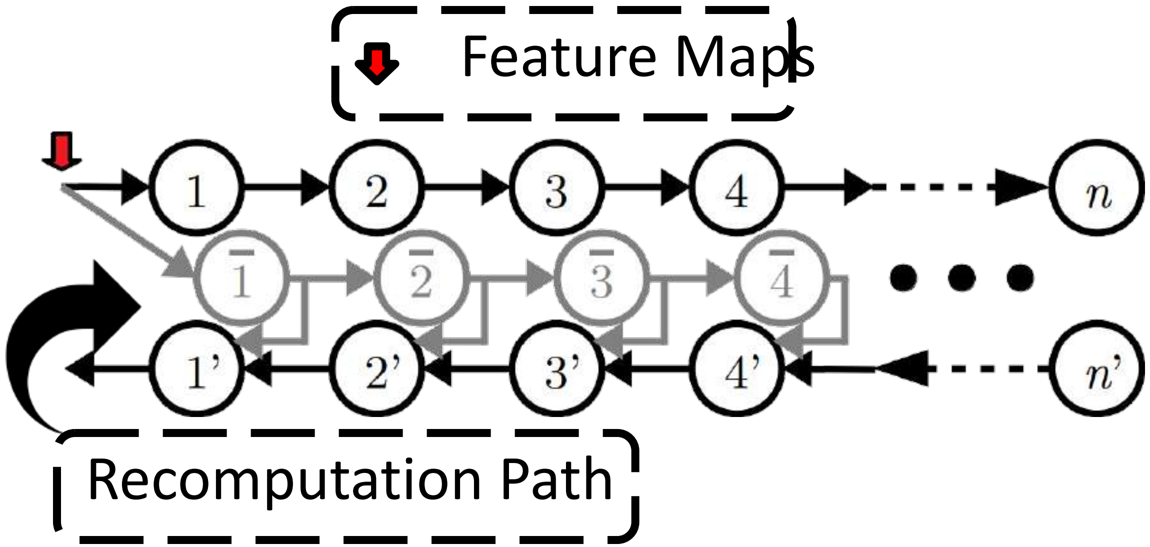
Selective Recomputation

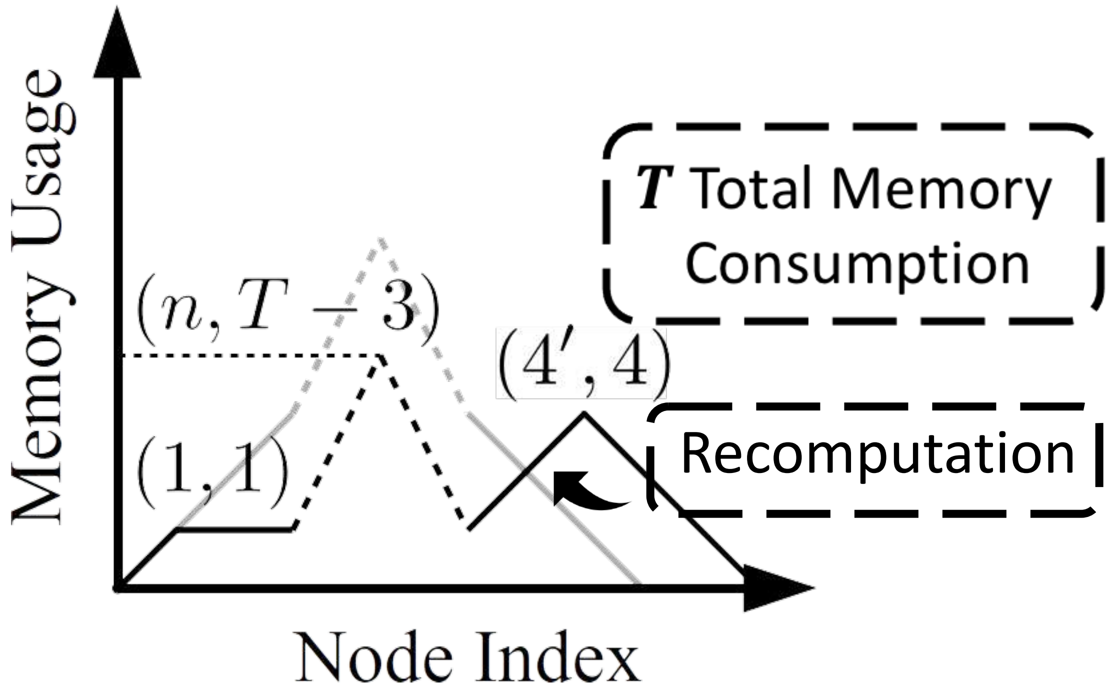




Selective Recomputation

- **Key Idea**: Trade **runtime** with **memory**.





- The recomputation path should only involve **lightweight** operators.

❶ Accurate Footprint Estimation

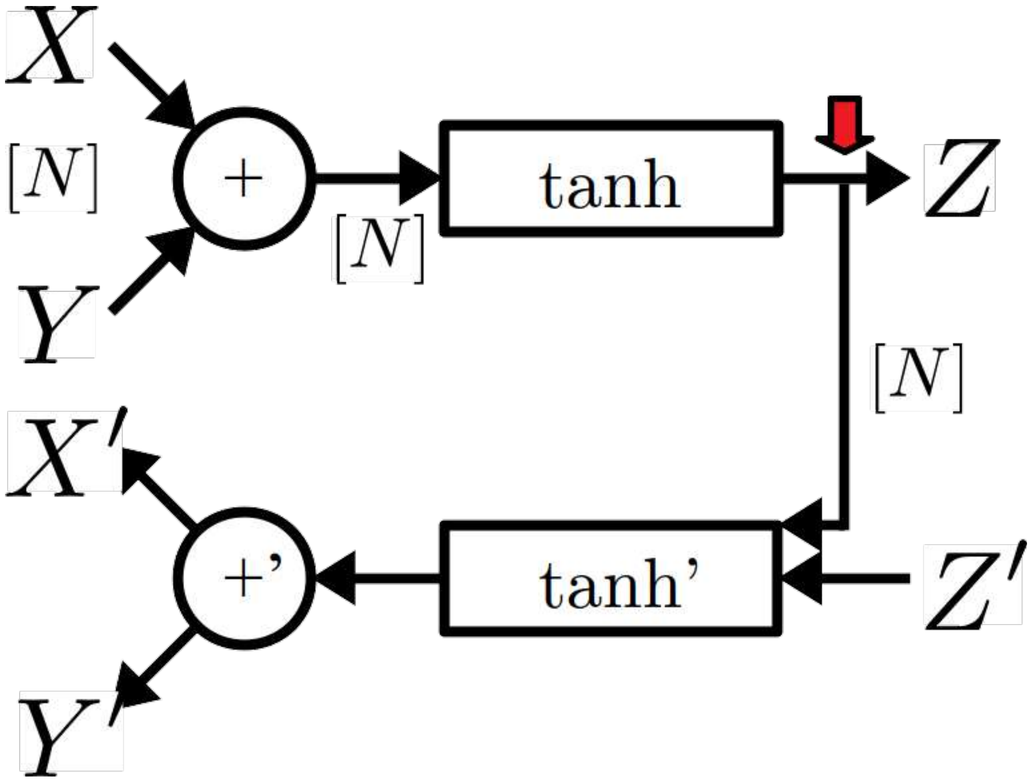
Example: $Z = \tanh(X + Y)$

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

❶ Accurate Footprint Estimation

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

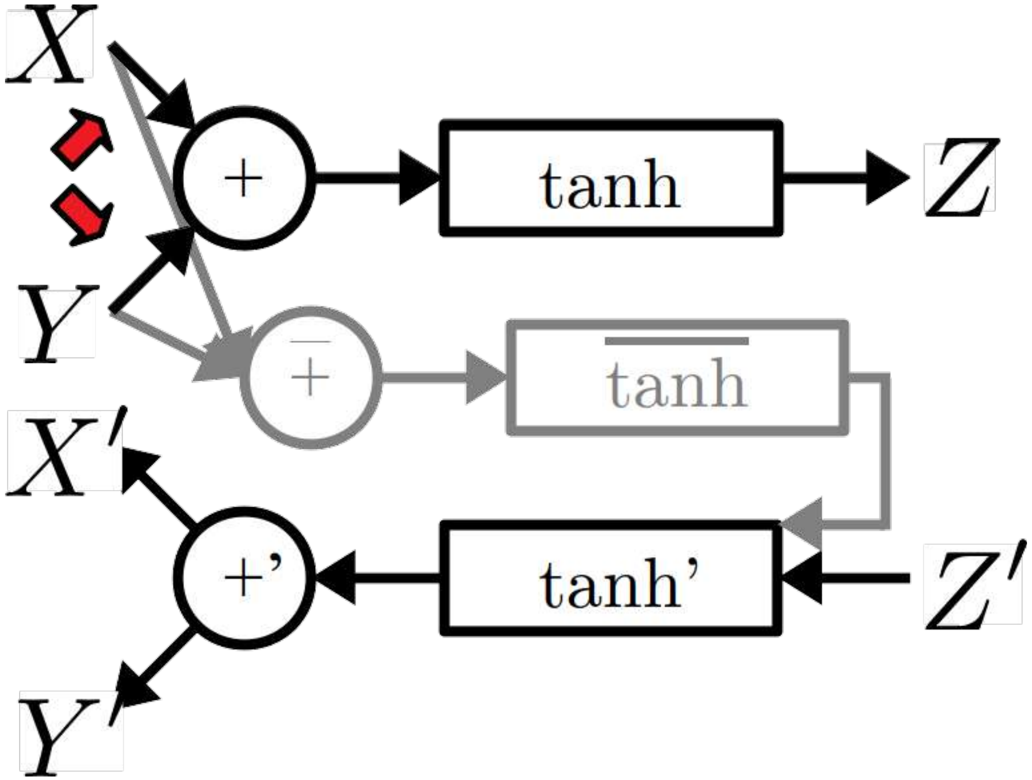
Example: $Z = \tanh(X + Y)$



❶ Accurate Footprint Estimation

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

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

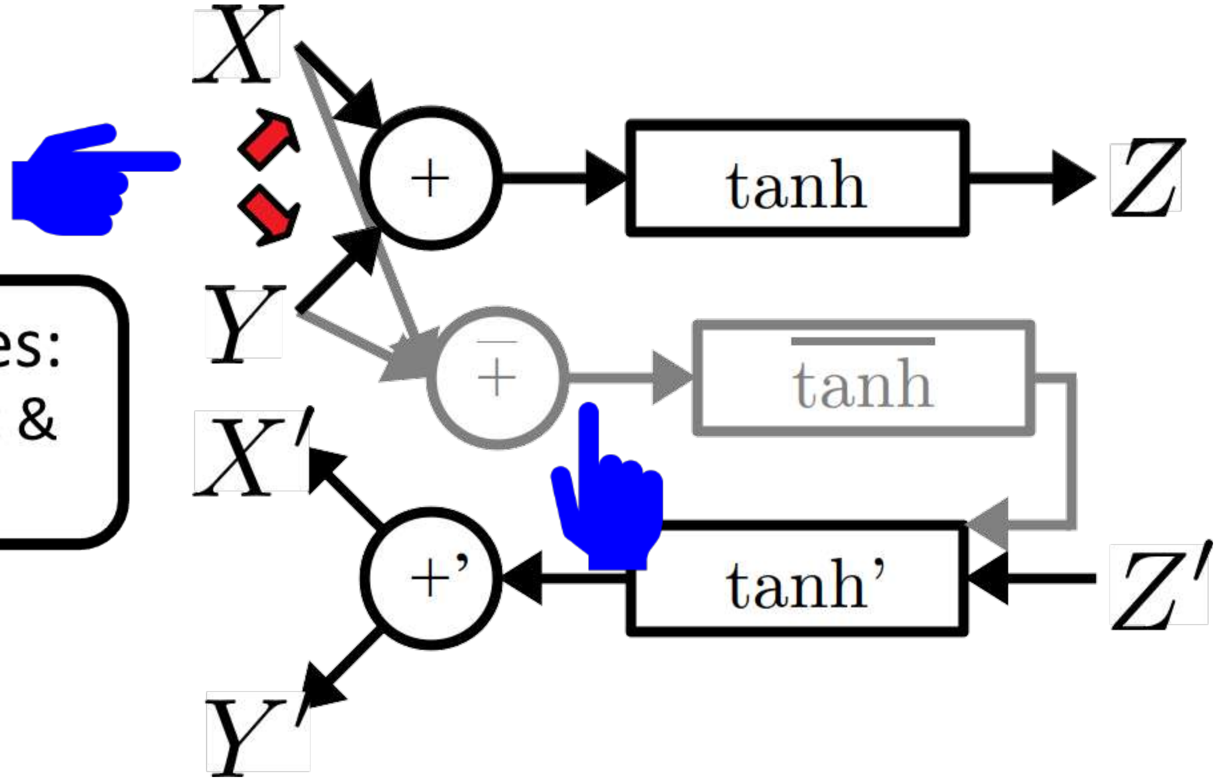


❶ Accurate Footprint Estimation

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

Selective Recomputation causes:
(–) increased memory footprint &
(–) performance degradation!

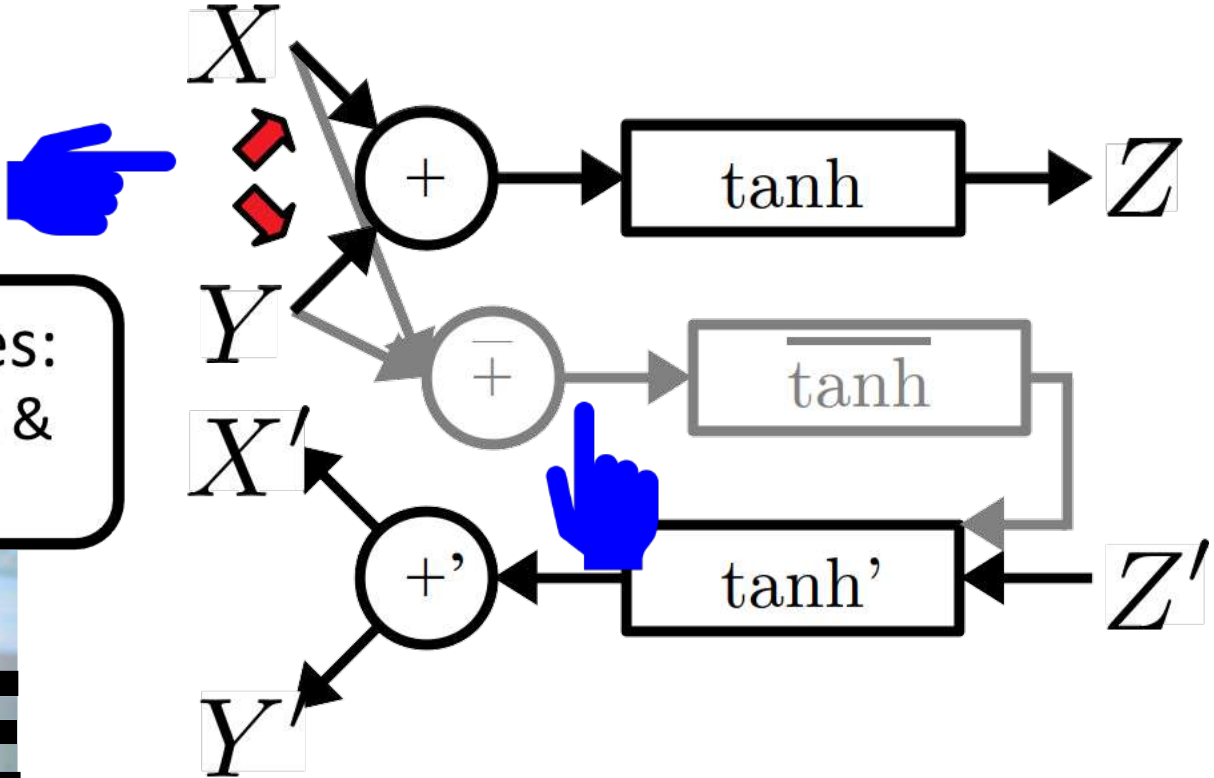
Example: $Z = \tanh(X + Y)$



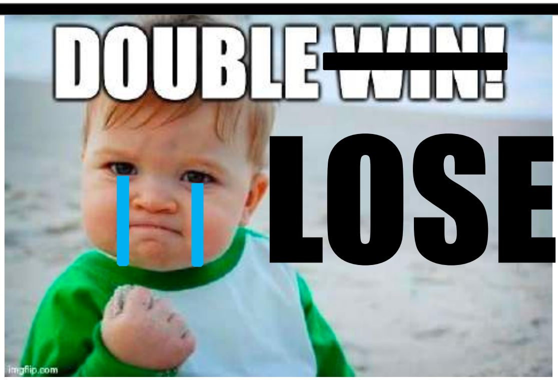
❶ Accurate Footprint Estimation

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

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



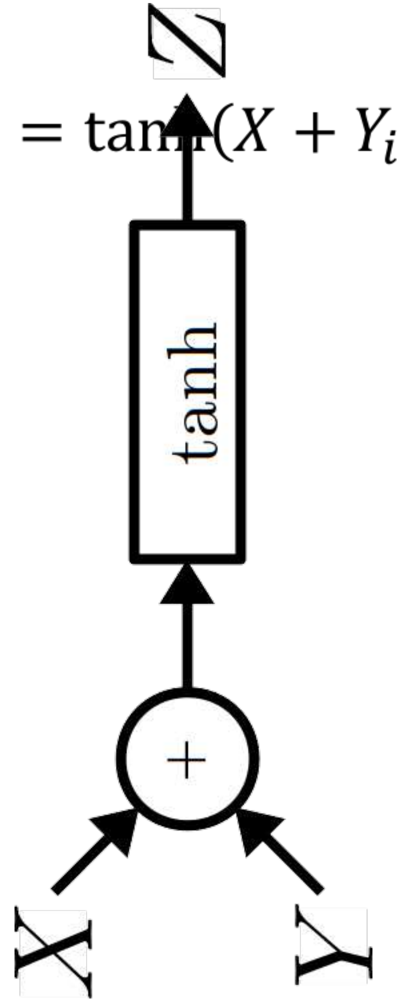
Selective Recomputation causes:
- (–) increased memory footprint &
- (–) performance degradation!



❶ Accurate Footprint Estimation

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

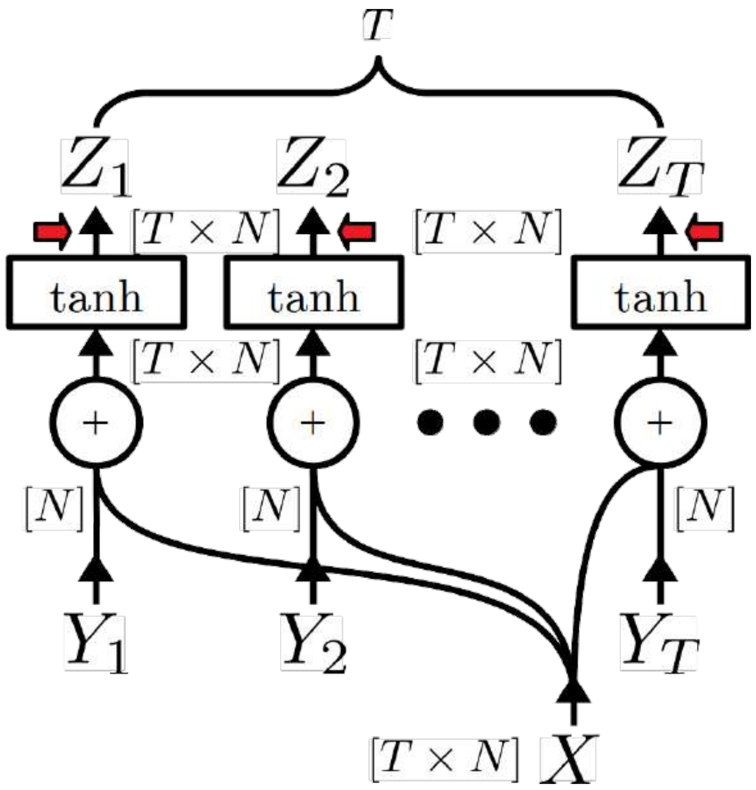
For each recomputation to be efficient, need to estimate its effect on the **global footprint**.



❶ Accurate Footprint Estimation

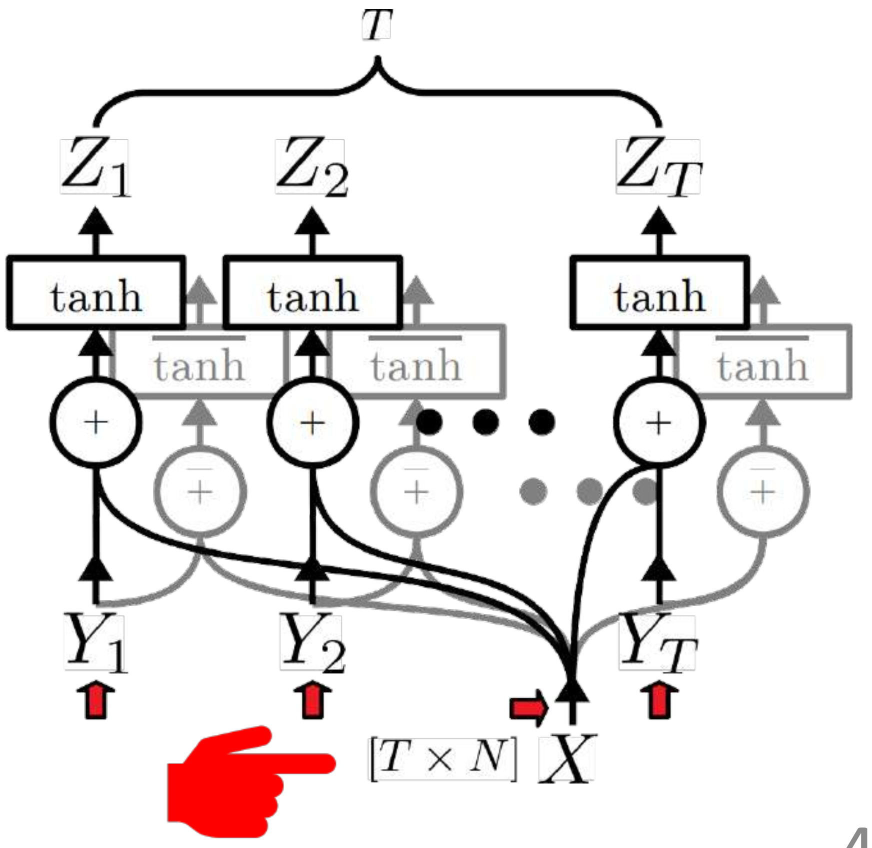
For each recomputation to be efficient, need to estimate its effect on the **global footprint**.

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



❶ Accurate 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 **global footprint**.

Selective Recomputation causes:
(+) feature maps: $T^{\mathbf{2}}N \rightarrow \mathbf{2}TN$

Global Footprint Analysis:

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

❷ Non-Conservative Overhead Estimation

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

Example: $Y = XW^T$

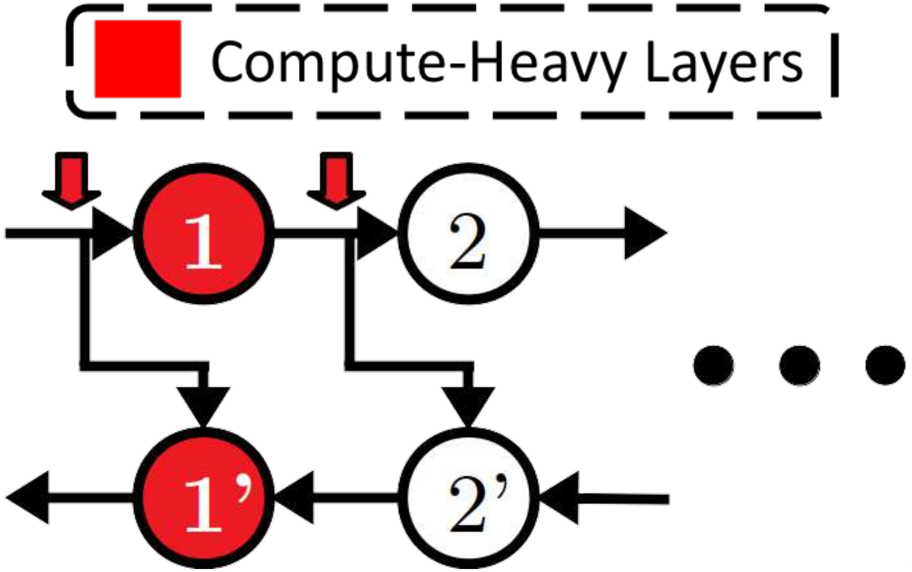
- **Compute-Heavy**
  - 50% of the NMT training time
- Excluded in prior works

❷ Non-Conservative Overhead Estimation

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

Example: $Y = XW^T$

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

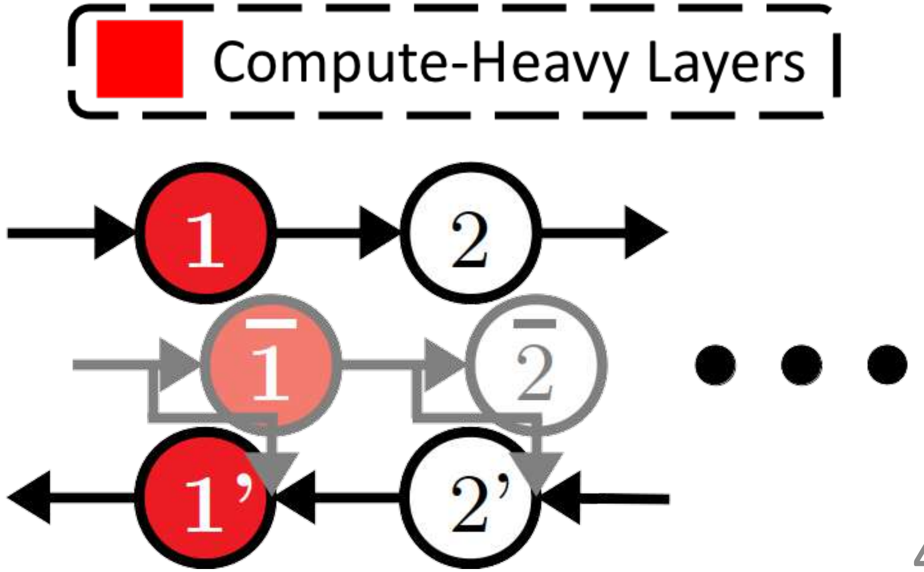


❷ Non-Conservative Overhead Estimation

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

Example: $Y = XW^T$

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



❷ Non-Conservative Overhead Estimation

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

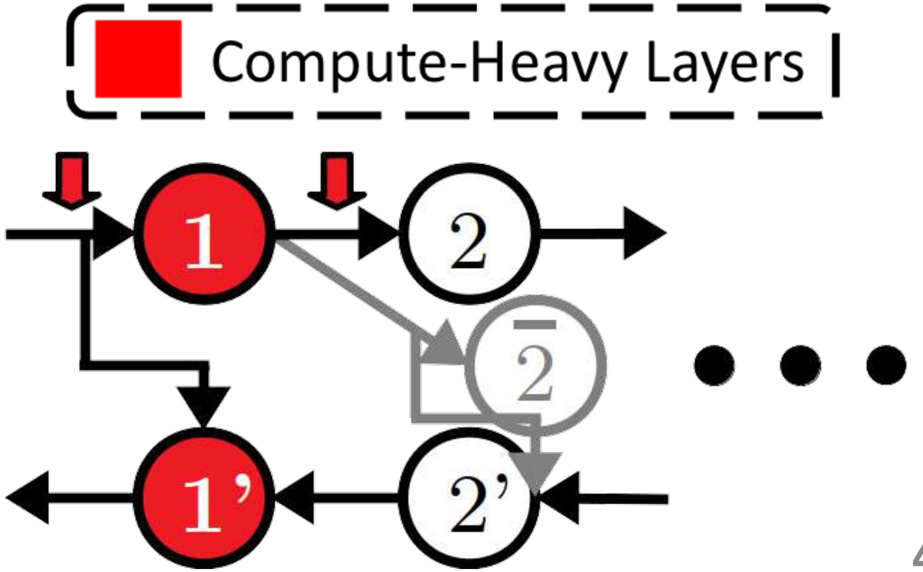
Layer-Specific Property:

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

(NO Dependency on Y)

Example: $Y = XW^T$

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



❷ Non-Conservative Overhead Estimation

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

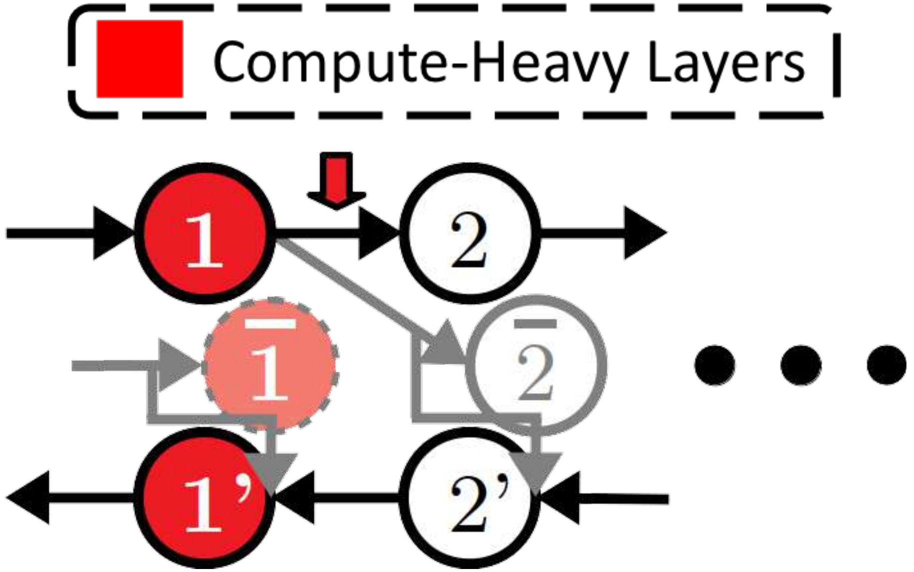
Layer-Specific Property:

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

(NO Dependency on Y)

Example: $Y = XW^T$

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



ECHO: A Graph Compiler Pass

- Integrated in the MXNet NNVM[1] module

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

ECHO: A Graph Compiler Pass

- Integrated in the MXNet NNVM[1] module
- Fully **Automatic & Transparent**
  - Requires NO changes in the training source code.

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

ECHO: A 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 of Selective Recomputation:

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

ECHO: A 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 of Selective Recomputation:

❶ Accurate Footprint Estimation

☞ *Bidirectional Dataflow Analysis*

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

ECHO: A 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 of Selective Recomputation:

❶ Accurate Footprint Estimation

☞ *Bidirectional Dataflow Analysis*

❷ Non-Conservative Overhead Estimation

☞ *Layer Specific Optimizations*

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

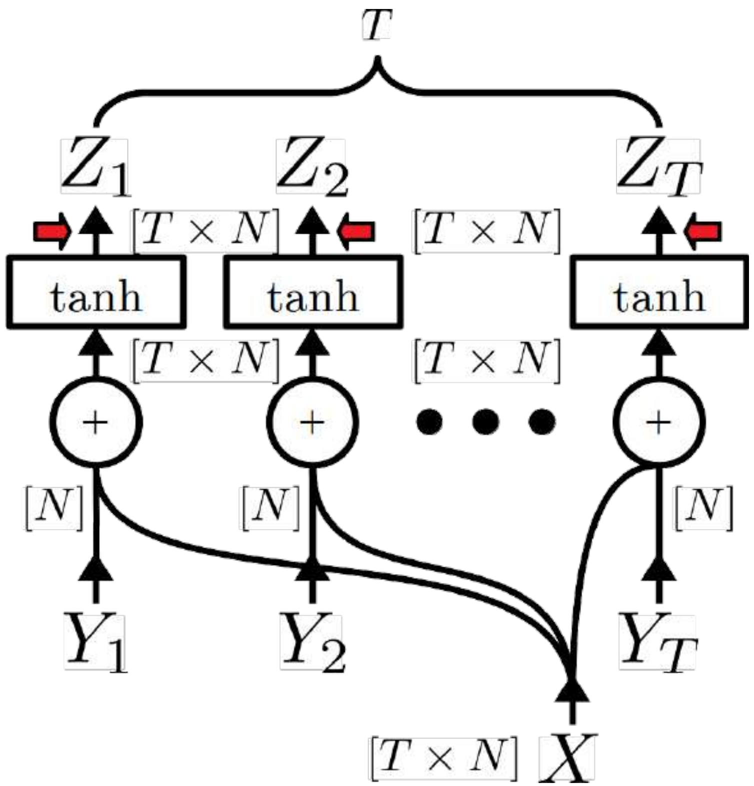
ECHO: Bidirectional Dataflow Analysis

- **Storage Reuse**

  Causes ALL correlated operators to forward propagate simultaneously.

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

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



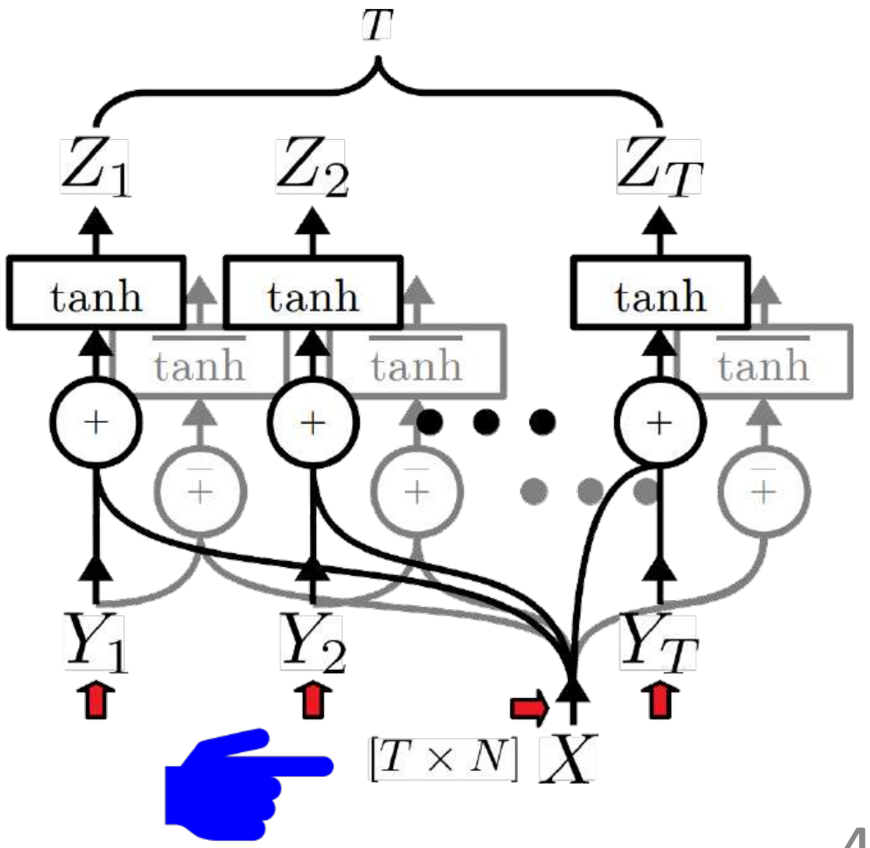
ECHO: Bidirectional Dataflow Analysis

- **Storage Reuse**

  Causes ALL correlated operators to forward propagate simultaneously.

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

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



ECHO: Bidirectional Dataflow Analysis

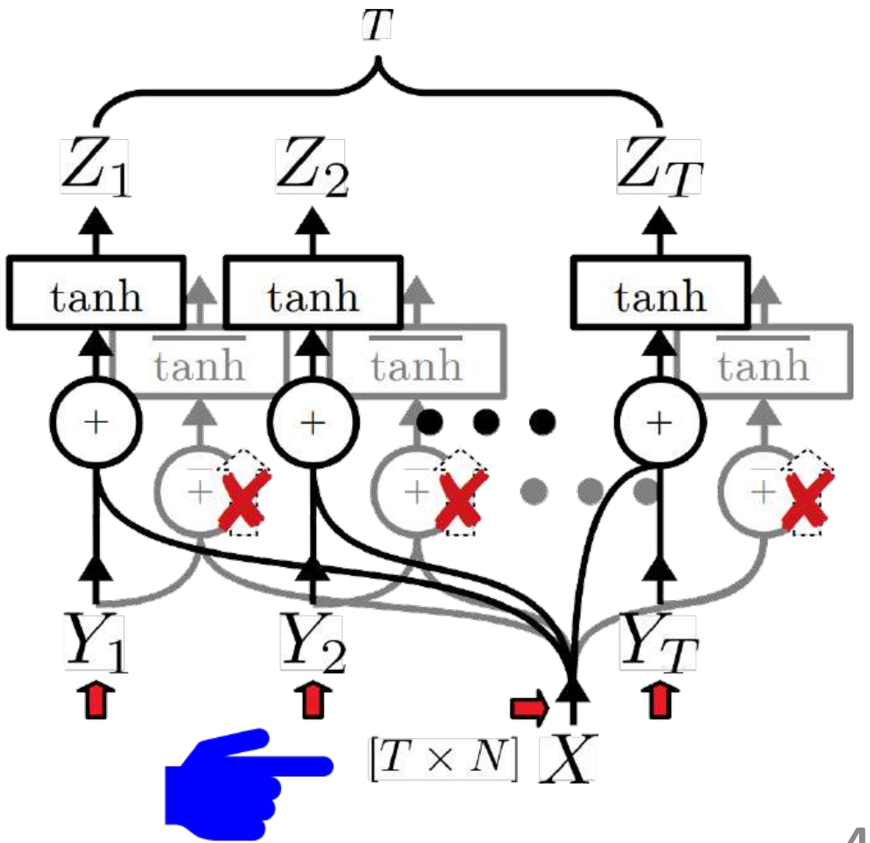
- **Storage Reuse**

  Causes ALL correlated operators to forward propagate simultaneously.

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

$\boxed{T^2N \nleq 2TN}$

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



Evaluation: Benchmarks

Sockeye[1]

[1] F. Hieber et al. *Sockeye: A Toolkit for Neural Machine Translation*. Arxiv Preprint 2017

- State-of-the-Art Neural Machine Translation Toolkit under MXNet

Evaluation: Benchmarks

Sockeye[1]

[1] F. Hieber et al. *Sockeye: A Toolkit for Neural Machine Translation*. Arxiv Preprint 2017

- State-of-the-Art Neural Machine Translation Toolkit under MXNet
- Datasets:
  - IWSLT’15 English-Vietnamese (Small)
  - WMT’16 English-German (Large)

Evaluation: Benchmarks

Sockeye[1]

[1] F. Hieber et al. *Sockeye: A Toolkit for Neural Machine Translation*. Arxiv Preprint 2017

- 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 BLEU Score

Evaluation: Systems

Baseline	Baseline System without Selective Recomputation

Evaluation: Systems

Baseline	Baseline System without Selective Recomputation
Mirror	T. Chen et al.[1]

[1] T. Chen et al. *Training Deep Nets with Sublinear Memory Cost*. Arxiv Preprint 2016

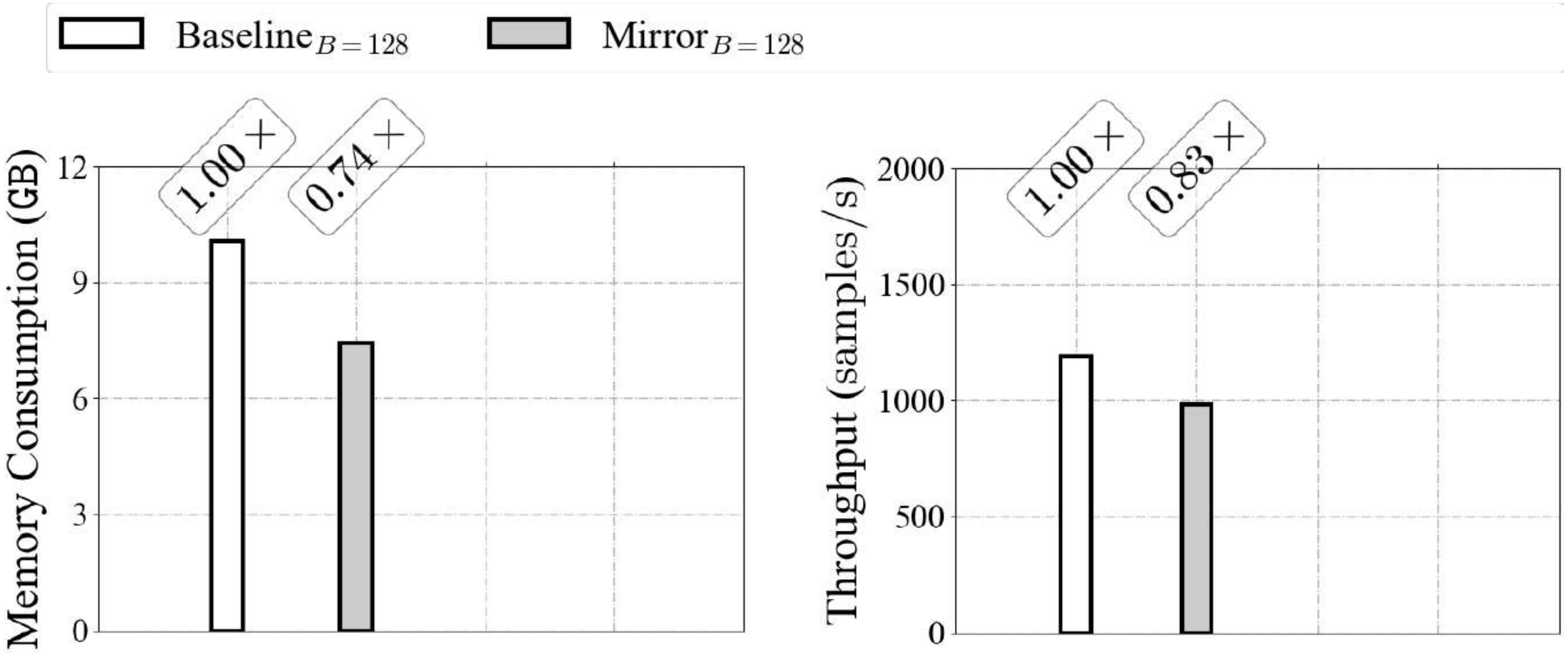
Evaluation: Systems

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 Preprint 2016

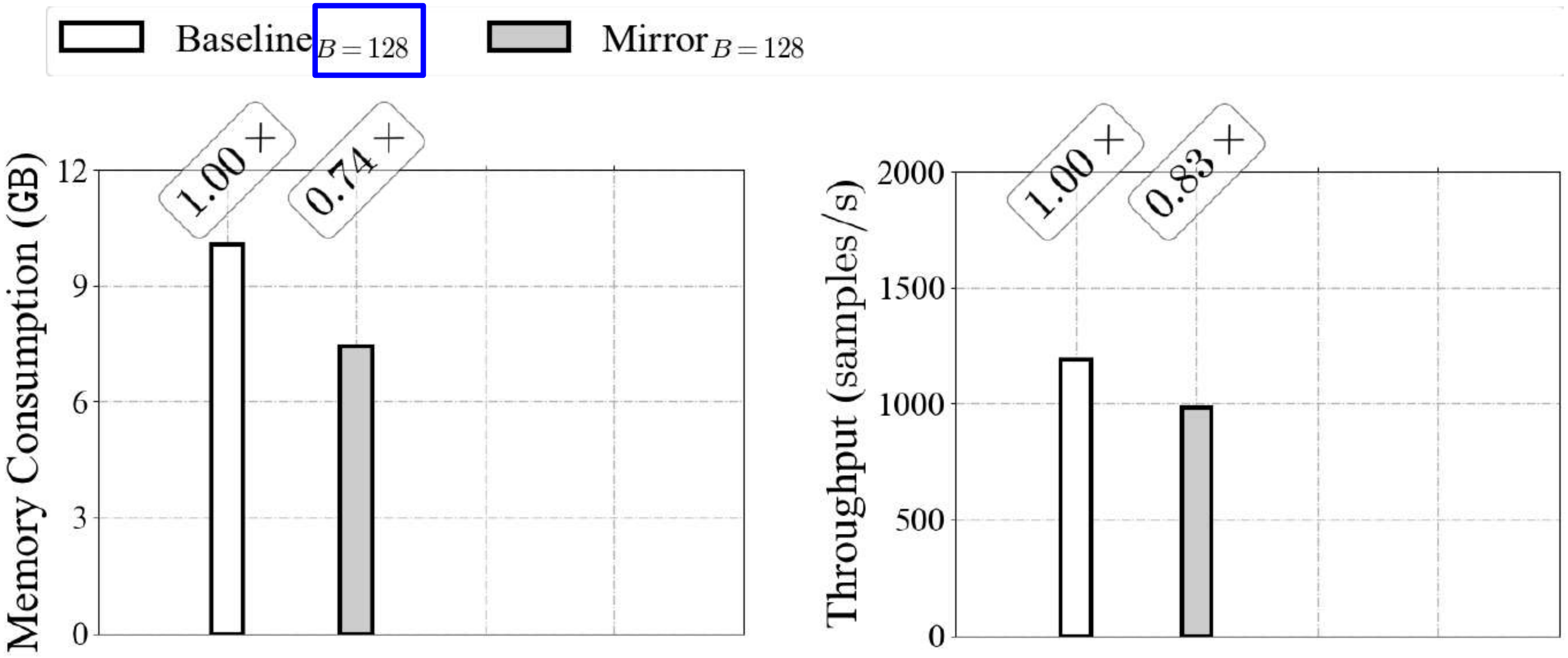
ECHO 's Effect on Memory and Performance

Small Dataset, Single-GPU Experiment



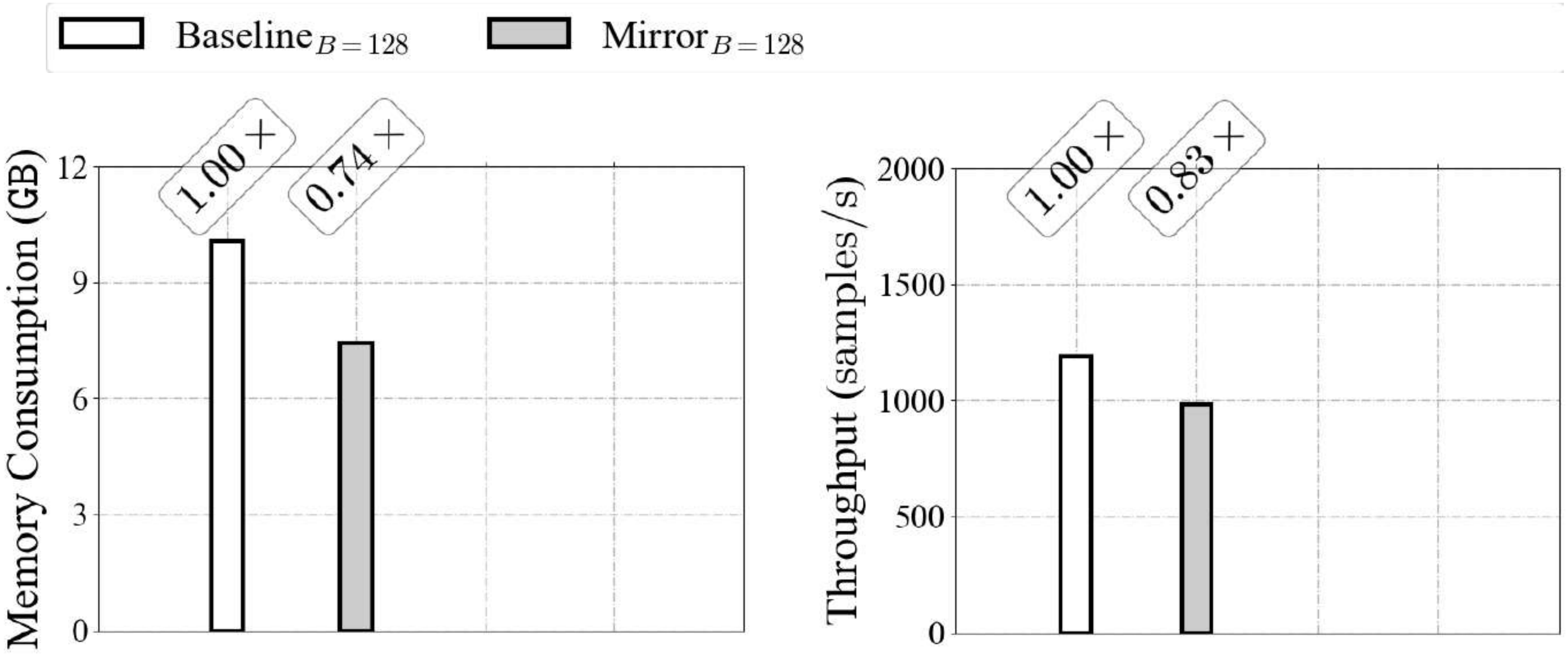
ECHO 's Effect on Memory and Performance

Small Dataset, Single-GPU Experiment



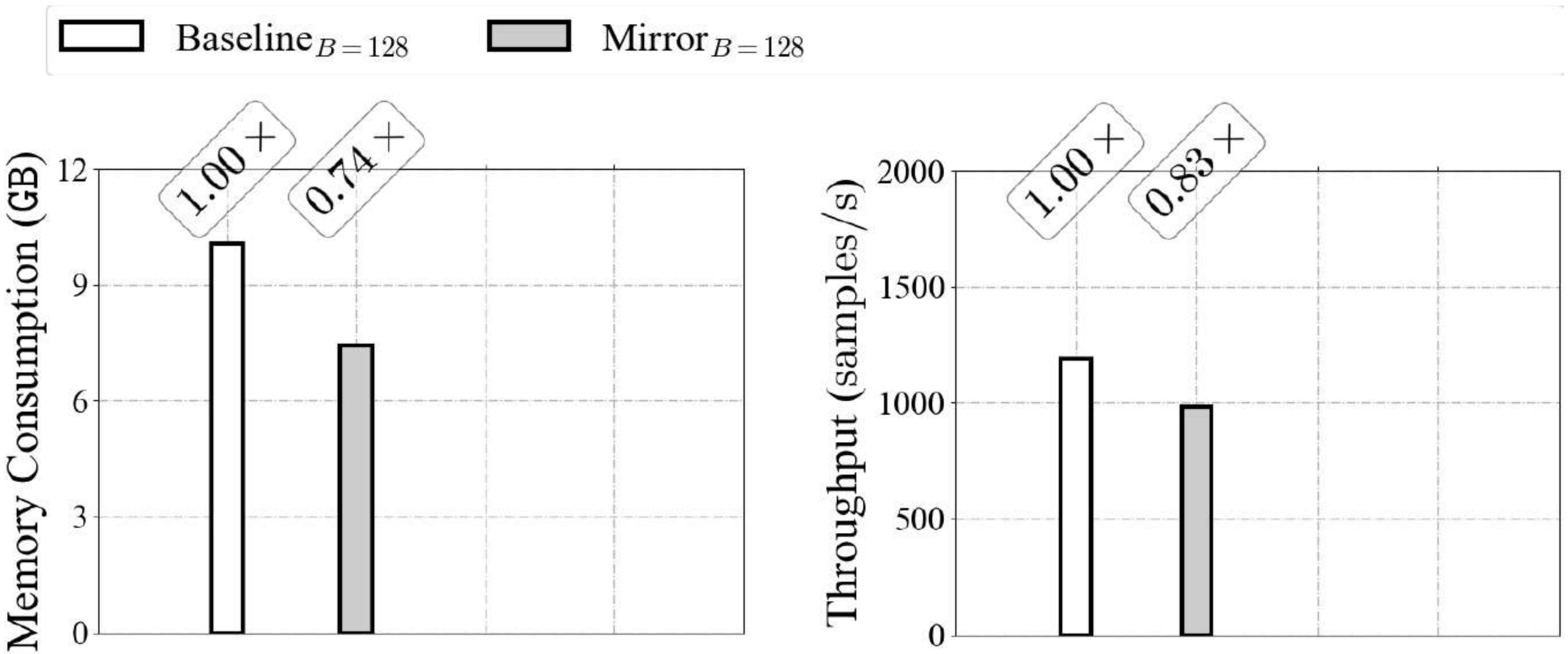
ECHO 's Effect on Memory and Performance

Small Dataset, Single-GPU Experiment



ECHO 's Effect on Memory and Performance

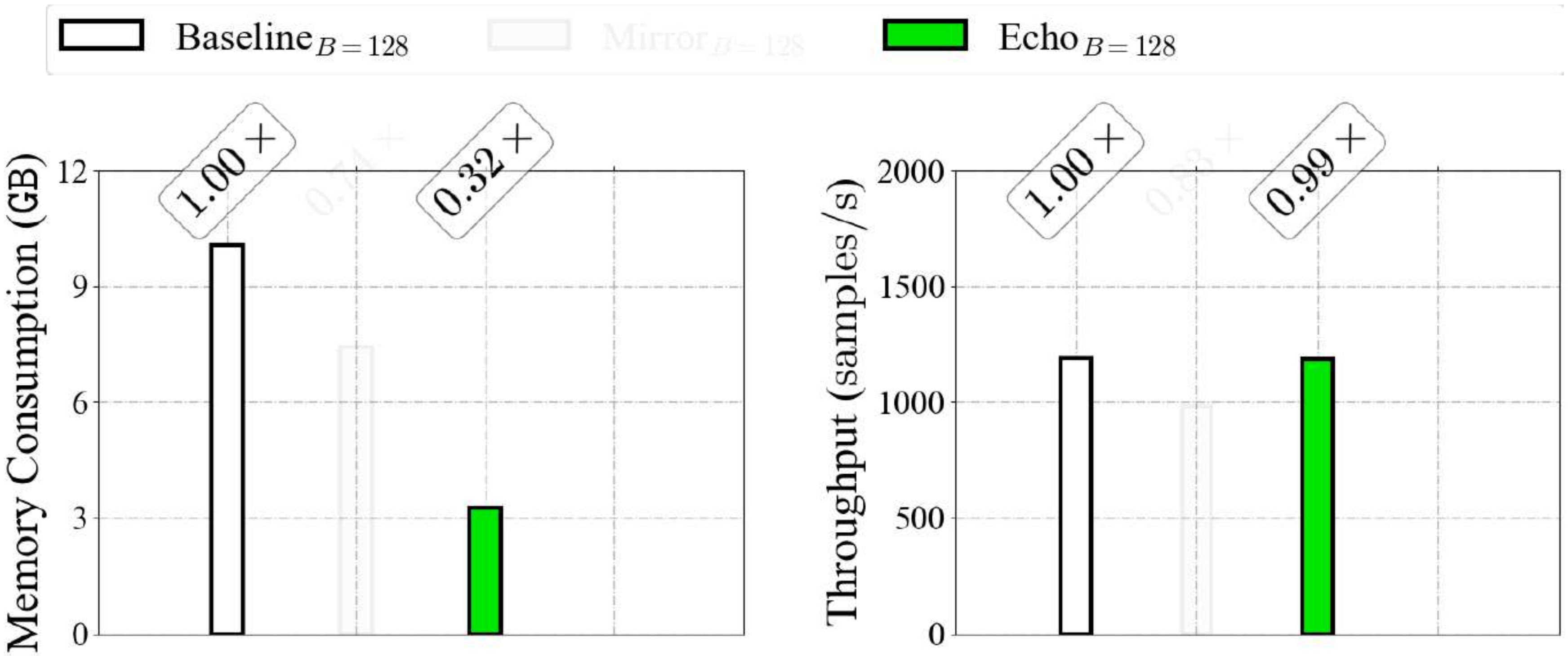
Small Dataset, Single-GPU Experiment



	Reduction Ratio	Overhead
Mirror	Low	High

ECHO ’s Effect on Memory and Performance

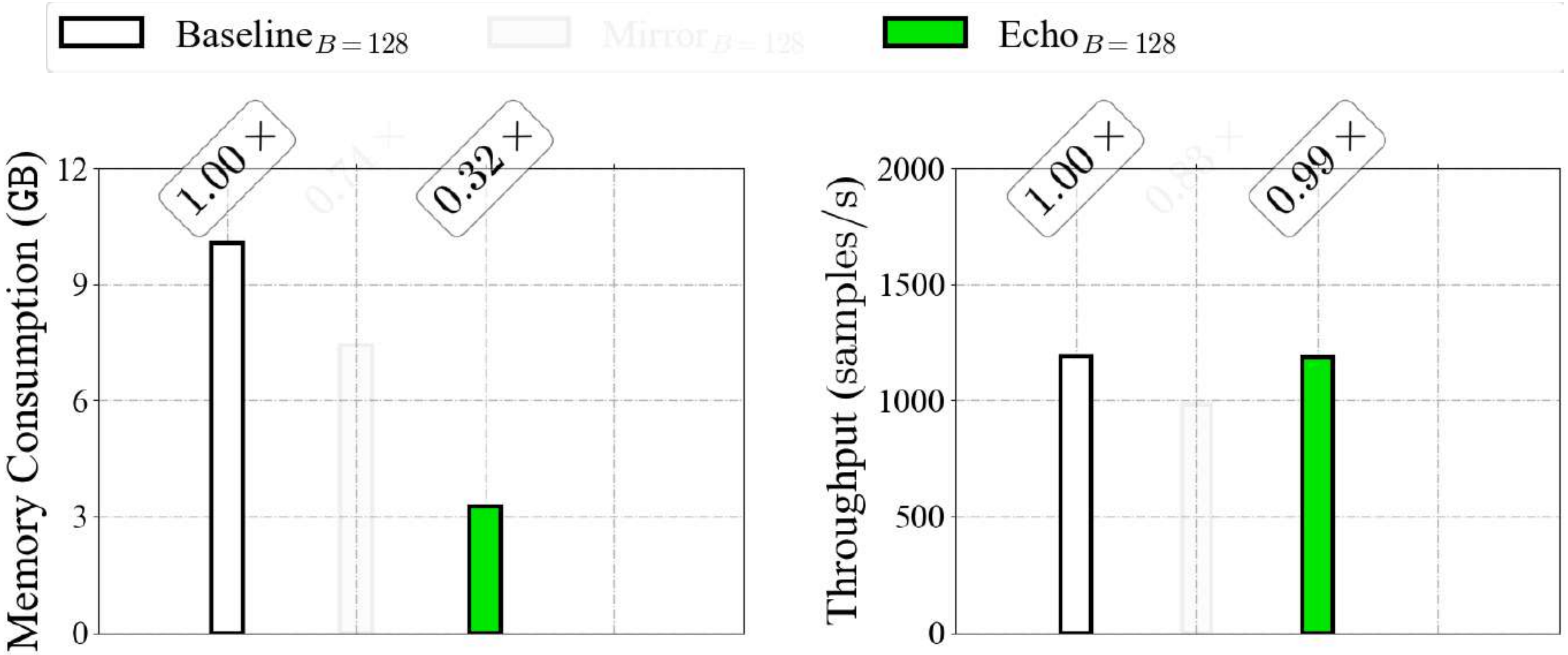
Small Dataset, Single-GPU Experiment



	Reduction Ratio	Overhead

ECHO 's Effect on Memory and Performance

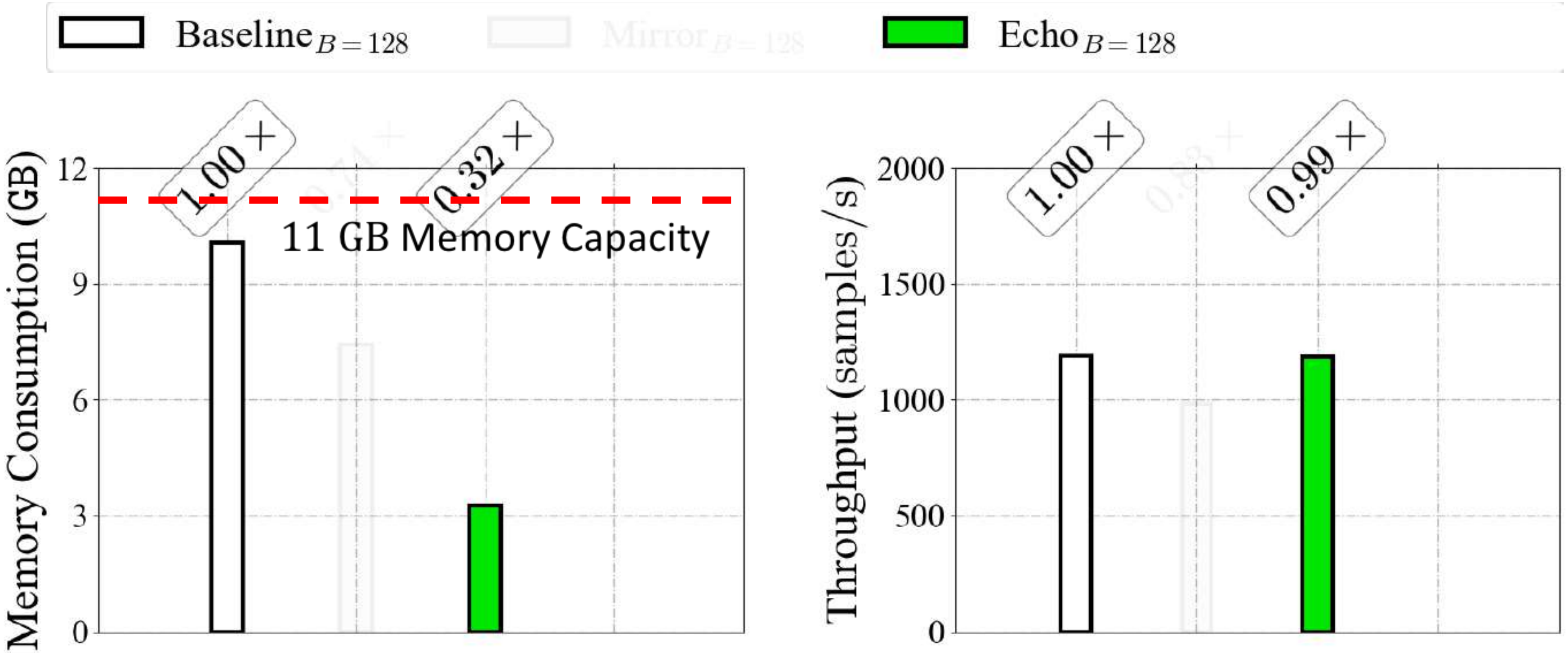
Small Dataset, Single-GPU Experiment



	Reduction Ratio	Overhead
ECHO	High	Low

ECHO 's Effect on Memory and Performance

Small Dataset, Single-GPU Experiment

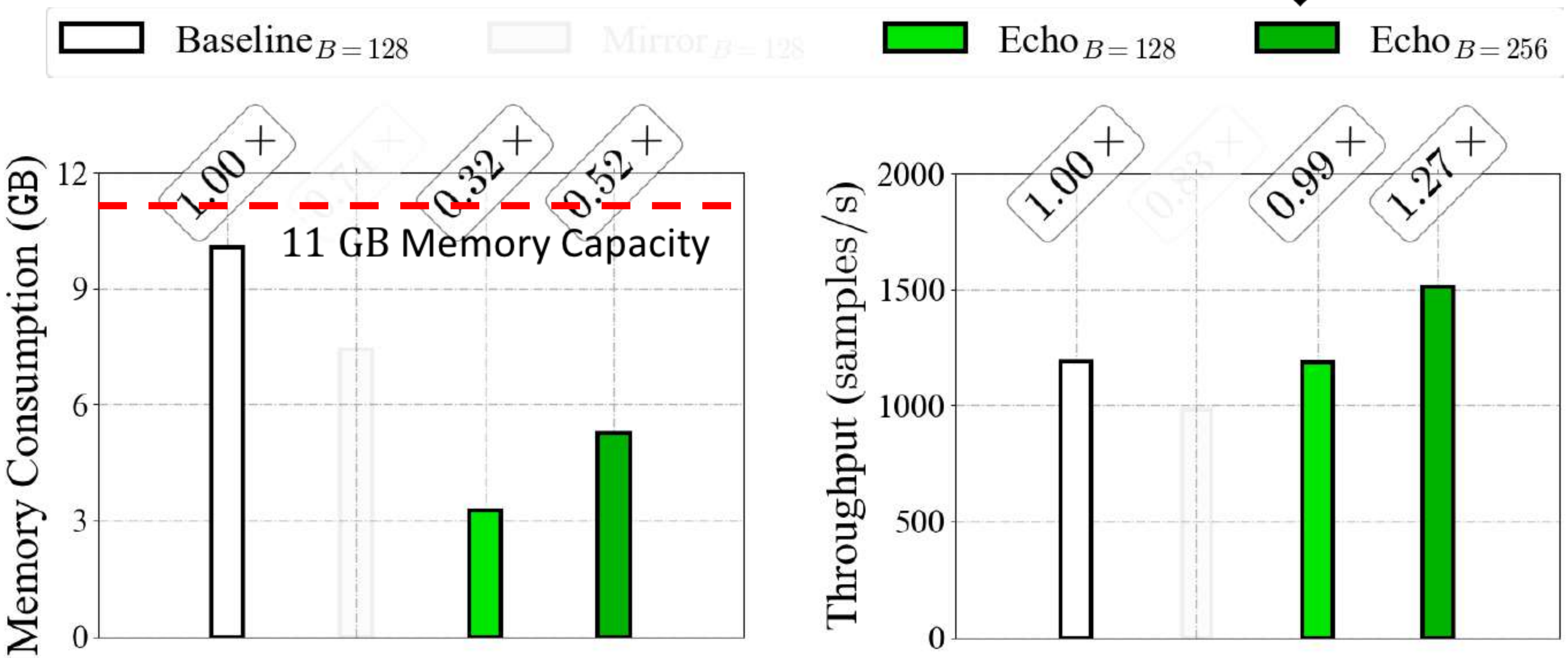


	Reduction Ratio	Overhead
ECHO	High	Low

ECHO 's Effect on Memory and Performance

Small Dataset, Single-GPU Experiment

2× Training Batch Size

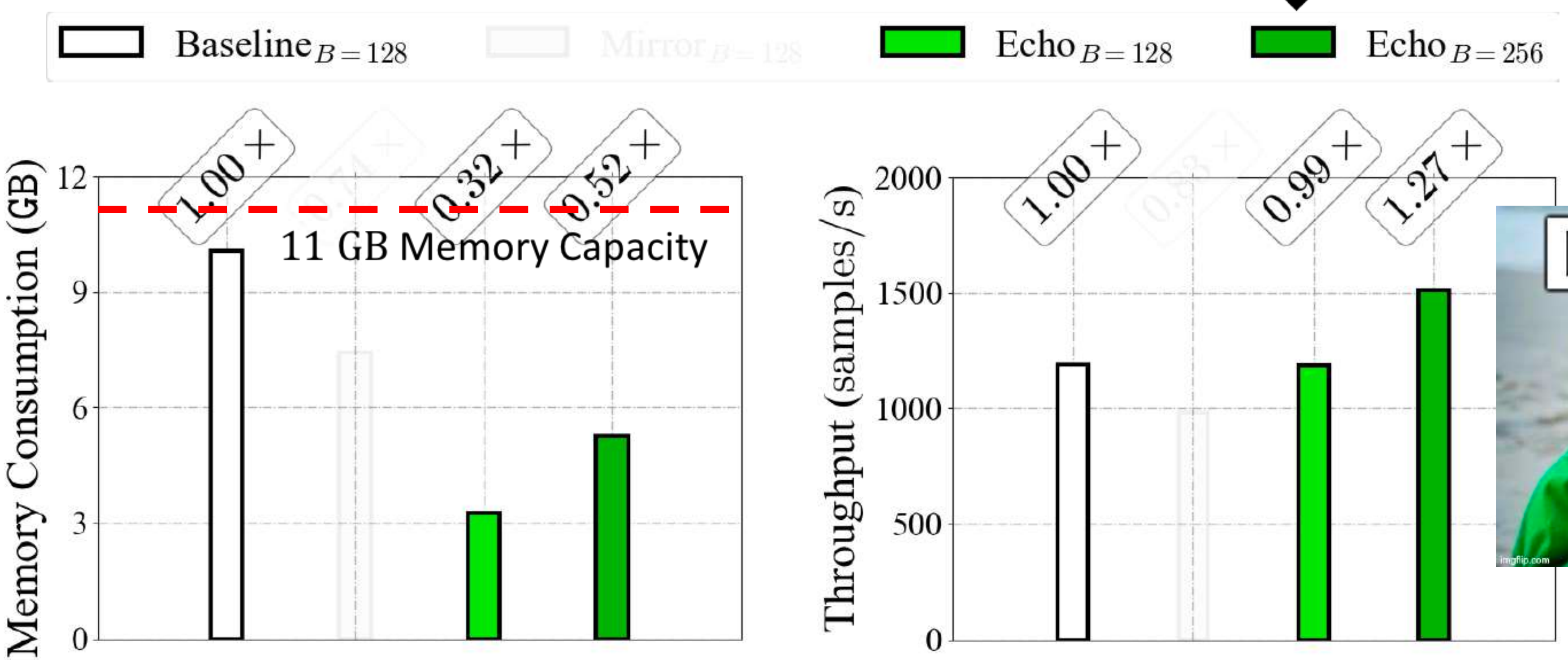


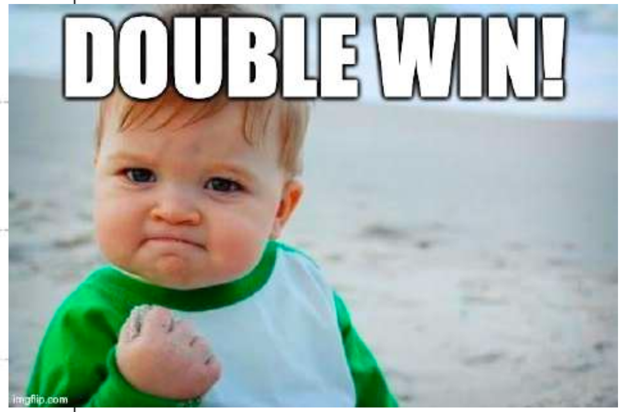
	Reduction Ratio	Overhead
ECHO	High	Low

ECHO 's Effect on Memory and Performance

Small Dataset, Single-GPU Experiment

2× Training Batch Size





	Reduction Ratio	Overhead
ECHO	High	Low

ECHO 's Effect on Training Convergence

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

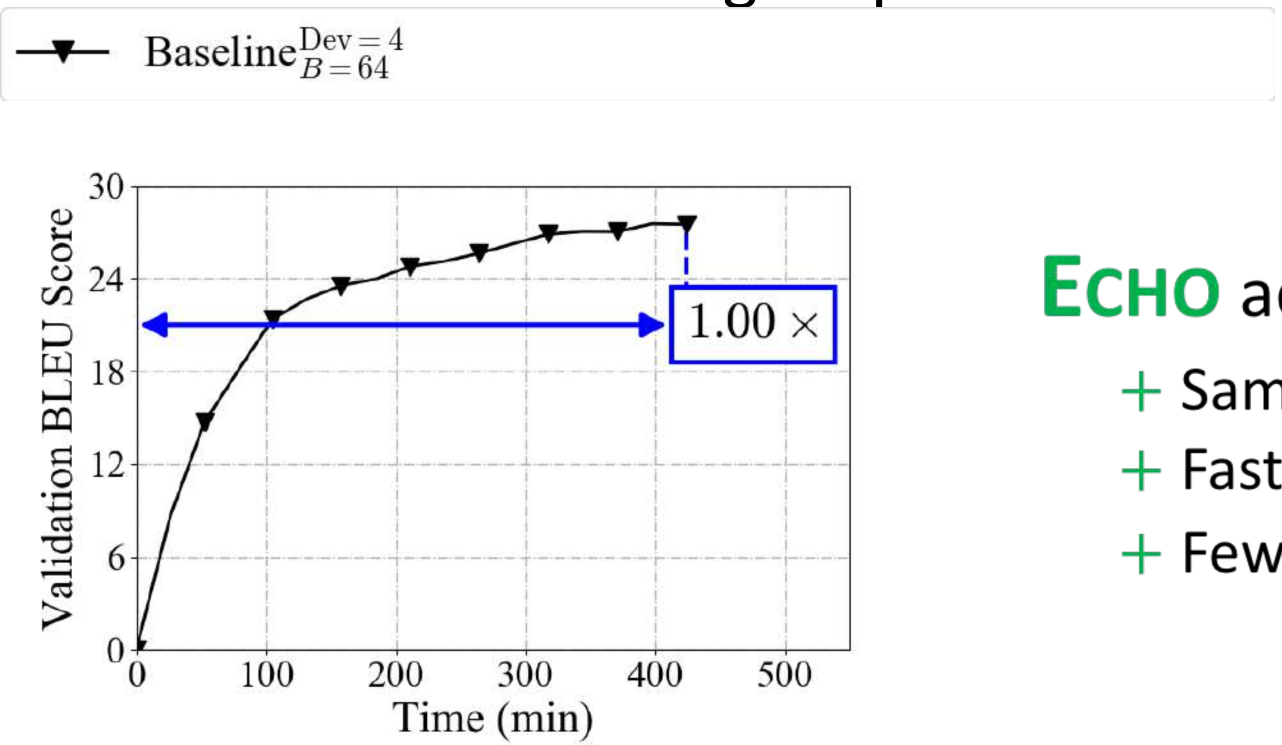
ECHO achieves:

+ Same Validation BLEU Score
+ Faster Convergence
+ Fewer Compute Devices

ECHO 's Effect on Training Convergence

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

Better



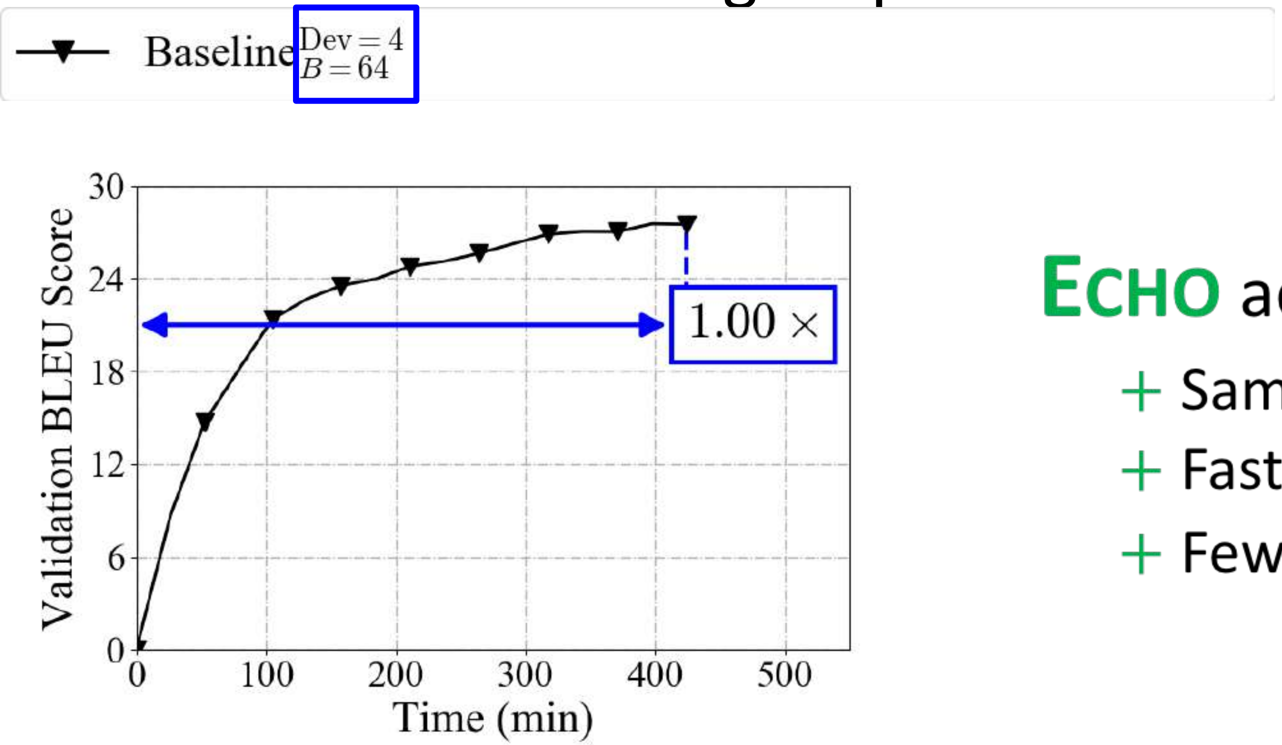
ECHO achieves:

+ Same Validation BLEU Score
+ Faster Convergence
+ Fewer Compute Devices

ECHO 's Effect on Training Convergence

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

Better



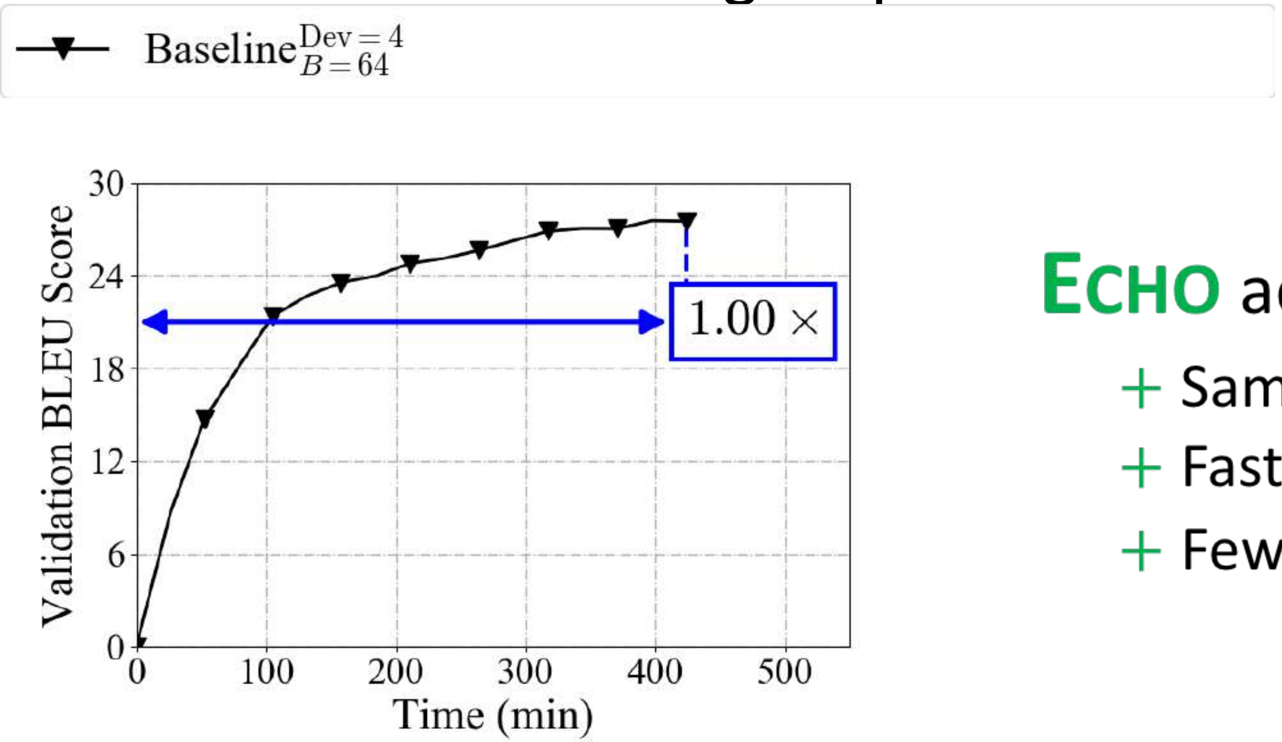
ECHO achieves:

- + Same Validation BLEU Score
- + Faster Convergence
- + Fewer Compute Devices

ECHO 's Effect on Training Convergence

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

Better

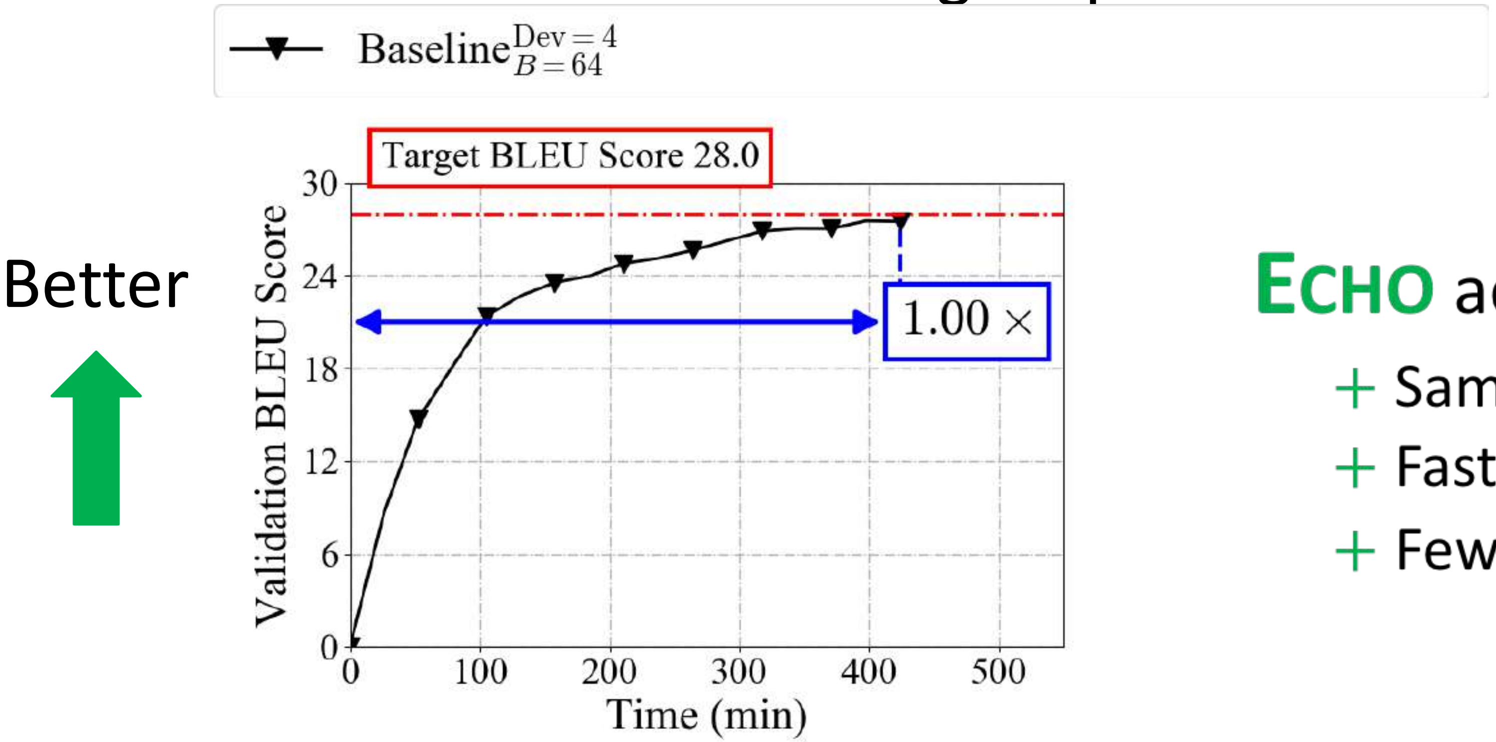


ECHO achieves:

+ Same Validation BLEU Score
+ Faster Convergence
+ Fewer Compute Devices

ECHO 's Effect on Training Convergence

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

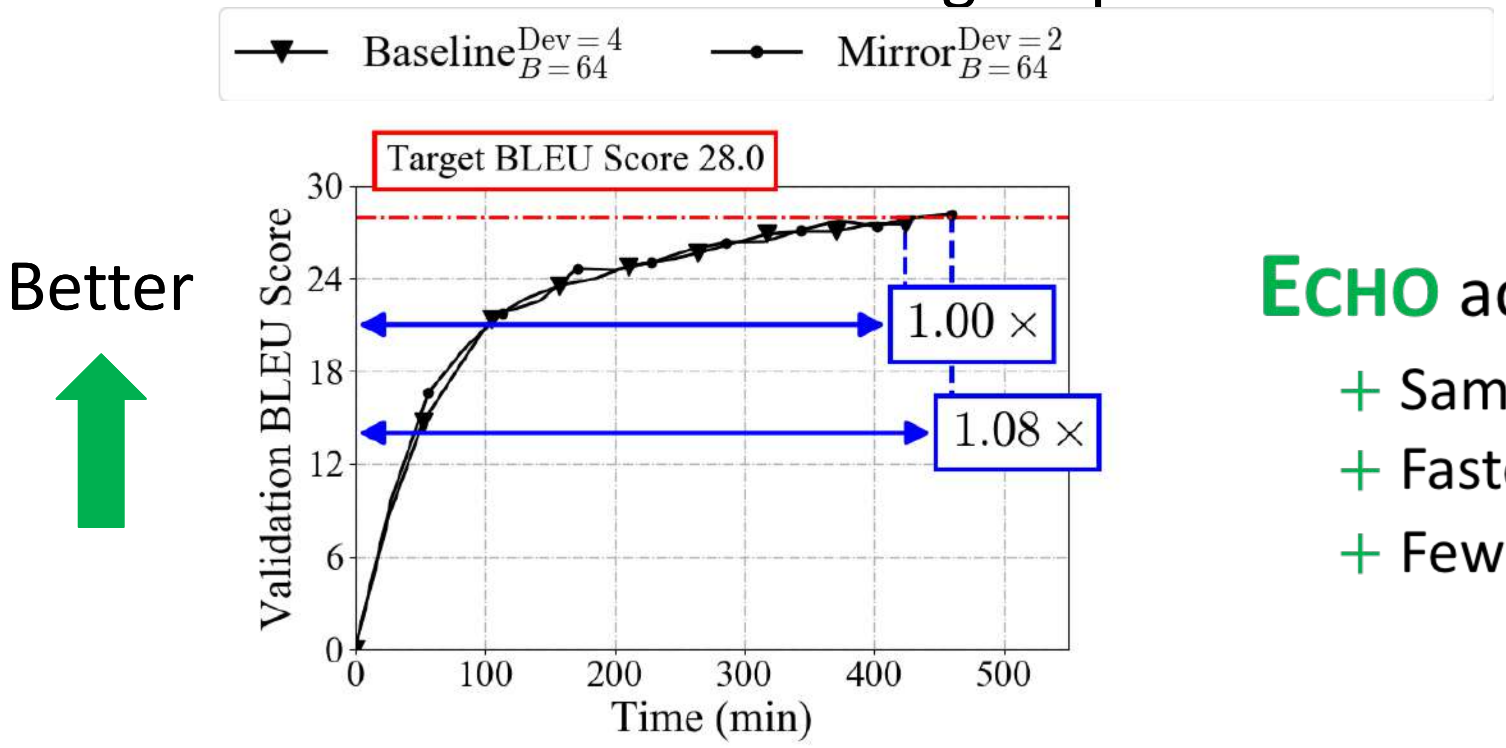


ECHO achieves:

+ Same Validation BLEU Score
+ Faster Convergence
+ Fewer Compute Devices

ECHO 's Effect on Training Convergence

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



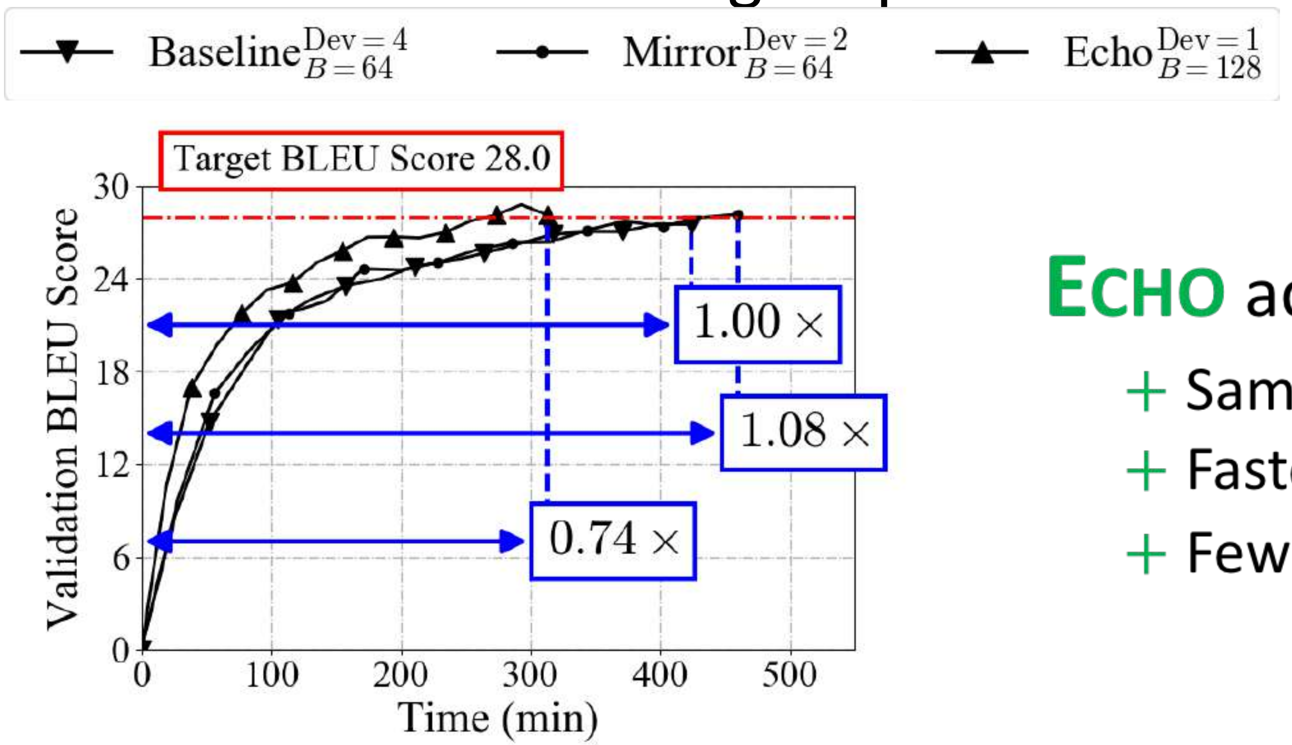
ECHO achieves:

+ Same Validation BLEU Score
+ Faster Convergence
+ Fewer Compute Devices

ECHO 's Effect on Training Convergence

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

Better



ECHO achieves:

+ Same Validation BLEU Score
+ Faster Convergence
+ Fewer Compute Devices

My Students: EcoSystem Research Group

- Bojian Zheng (PhD)
- Alexandra Tsvetkova (PhD)
- James Gleeson (PhD)
- Anand Jayarajan (PhD)
- Shang (Sam) Wang (PhD)
- Jiacheng Yang (PhD)
- Pavel Golikov (MSc)
- Yaoyao Ding (MASc)
- Daniel Snider (MSc)
- Kevin Song (MASc)
- Xin Li (MASc)
- Jasper Zhu (MSc)
- Peiming Yang (MASc)

- Yu Bo Gao (BSc)
- Qingyuan Qie (BSc)
- Chenhao Jiang (BSc)
- Murali Andoorveedu (BASc)

Thank you! Questions?