EcoRNN: Efficient Computing of LSTM RNN on GPUs

Bojian Zheng (Graduate Student), Gennady Pekhimenko (Advisor)
bojian, pekhimenko@cs.toronto.edu

EcoSystem Research Group, Department of Computer Science
University of Toronto
www.cs.toronto.edu/ecosystem

The 51st Annual IEEE/ACM International Symposium on Microarchitecture, 2018, Fukuoka, Japan
Background: Sequence Learning

Machine Translation

Hello 好
Background: Sequence Learning

Machine Translation

Speech Recognition
Background: Sequence Learning

Machine Translation

Speech Recognition

B. Zheng, G. Pekhimenko (EcoSystem) EcoRNN MICRO 51 SRC 2 / 12
Background. Long-Short-Term-Memory (LSTM) Recurrent-Neural-Network (RNN)

Timeline

$c_t$  RNN Cell at time $t$  $i_t$  Cell Input  $h_t$  Cell Hidden State
Background. Long-Short-Term-Memory (LSTM) Recurrent-Neural-Network (RNN)
Problem Statement: (1) Performance

- **Default** has **cudaLaunch overhead**.
- **CuDNN** is **closed-source, limits innovation**.

Reference: *cuDNN LSTM RNN*. Appleyard et al.
Problem Statement: (1) Performance

- **Default** has *cudaLaunch* overhead.
- **CuDNN** is closed-source, limits innovation.

Reference: *cuDNN LSTM RNN*. Appleyard et al.
Problem Statement: (2) Memory Capacity


Training throughput in ResNet-50 saturates at large batch size.
Problem Statement: (2) Memory Capacity


Training throughput in machine translation model increases almost linearly.
Problem Statement: (2) Memory Capacity


- RNN training is **Memory Capacity**-bounded.
EcoRNN is a new open-source implementation that has performance comparable with or even better than CuDNN. It has smaller memory footprint and supports auto-tuning.
EcoRNN is a new open-source implementation that has performance comparable with or even better than CuDNN. It has smaller memory footprint and supports auto-tuning.
EcoRNN is a new open-source implementation that has performance comparable with or even better than CuDNN. It has smaller memory footprint and supports auto-tuning.

All changes are transparent to the programmers.
Preliminary Results: (1) Performance

The runtime bottleneck is FC layers.
Preliminary Results: (1) Performance

The runtime bottleneck is **FC layers**.

⇒

**Data layout optimization** improves cache hit rate.
Preliminary Results: (1) Performance

Training Throughput Comparison on the MXNet Language Modeling Benchmark

- Up to 2× faster than Default, and
- Up to 1.3× faster than CuDNN.
Preliminary Results: (2) Memory Capacity

Memory Consumption Profile of the Machine Translation Model

The memory bottleneck is Features Maps of Attention and RNN Layers.
Future Work

- Weight Parameter Reuse
  - Same observation made by *Baidu Persistent RNN*.
  - **Inflexibility**: Difficult to port to new cell types and architectures
Future Work

- **Weight Parameter Reuse**
  - Same observation made by *Baidu Persistent RNN*.
  - **Inflexibility**: Difficult to port to new cell types and architectures
    ⇐ *Machine Learning Compilers* (e.g., *TVM, XLA*).
Future Work

- Weight Parameter Reuse
  - Same observation made by *Baidu Persistent RNN*.
  - **Inflexibility**: Difficult to port to new cell types and architectures
    ⇐ *Machine Learning Compilers* (e.g., *TVM*, *XLA*).

- Memory Compression
Future Work

- **Weight Parameter Reuse**
  - Same observation made by *Baidu Persistent RNN*.
  - **Inflexibility**: Difficult to port to new cell types and architectures
    ⇐ Machine Learning Compilers (e.g., TVM, XLA).

- Memory Compression ⇐ *Gist* (Jain et al., ISCA’18)
Summary

Problem Statement

- Performance, Memory Capacity

Key Observations

- Default suffers from cudaLaunch overhead ⇐ Kernel Fusion
- CuDNN has low cache-utilization ⇐ Data Layout Optimization

Future Work

- Weight Parameter Reuse ⇐ Machine Learning Compilers

The memory bottleneck in machine translation model is Feature Maps of Attention and RNN Layers ⇐ Gist.
Summary

Problem Statement

- Performance, Memory Capacity

Key Observations

- Default suffers from cudaLaunch overhead $\iff$ Kernel Fusion.
- CuDNN has low cache-utilization $\iff$ Data Layout Optimization.
Problem Statement

- Performance, Memory Capacity

Key Observations

- Default suffers from cudaLaunch overhead ⇐ Kernel Fusion.
- CuDNN has low cache-utilization ⇐ Data Layout Optimization.

Future Work

- Weight Parameter Reuse ⇐ Machine Learning Compilers
- The memory bottleneck in machine translation model is Feature Maps of Attention and RNN Layers ⇐ Gist.
Experimental Settings

- **CUDA Toolkit 8, cuDNN 6, MXNet Ver. 0.11.0.**

DeepSpeech2 Training Throughput

![Graph showing DeepSpeech2 training throughput with Mini-batch size on the x-axis and Throughput on the y-axis.](image-url)