End-to-End Deep Learning Compiler Stack

AWS AI

Presenter: Animesh Jain
Amazon SageMaker Neo
Deep Learning is Pervasive

Amazon Lex
Build chatbots to engage customers

Amazon Transcribe
Automatic speech recognition

Amazon Polly
Natural sounding text to speech

Amazon Amazon Rekognition
Deep learning-based image and video analysis

Amazon Translate
Fluent translation of text

Amazon Comprehend
Discover insights and relationships in text
ARM – Unique Role in AWS Ecosystem

A1 EC2 Instance
*Optimized cost and performance for scale-out workloads*

Alexa Devices
*Build natural voice experiences*

3rd party Edge Devices
*Portable Deep Learning*
How to Accelerate Deep Learning?

- Build a ML model with the framework of your choice
- Train and tune the model using Amazon SageMaker
- Choose target hardware platform
- Amazon SageMaker Neo
  - SageMaker Neo will optimize the trained model for the target hardware platform
- You can then deploy your models on the cloud or at the edge
Agenda

• **Overview of Neo**
• Relay Graph Optimizations
• TVM Tensor IR Optimizations
• Evaluation
Deep Learning Inference

**Before**

- Model Marketplace
  - TensorFlow
  - mxnet
  - PyTorch
  - CNTK
- Inference Target
  - Intel Skylake
  - Mobile
  - Amazon Echo

**After**

- Model Marketplace
  - TensorFlow
  - mxnet
  - PyTorch
  - CNTK
- Deep Learning Compiler
- Inference Target
  - Intel Skylake
  - Mobile
  - Amazon Echo

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.
Models and hardware targets are far away!

TVM: end-to-end optimization stack
TVM Overview

- **Framework Graph**
  - TF
  - Mxnet
  - ... parsers

- **Relay Graph**
  - Target-independent Relay passes
  - Target-dependent Relay passes
  - Target-optimized graph

- **Schedule templates written in TVM Tensor IR**
  - Intel x86
  - ARM CPU
  - Nvidia GPU
  - ARM GPU
  - ... More targets

- **AutoTVM** – Tuning the kernels

- **Codegen** – LLVM, Cuda, C, ...

- **Optimized Binary**

**Framework Parsers**

**Graph level optimizations**

**Tensor-level optimizations**

**Machine code generation**
Agenda

• Overview of TVM
• **Relay Graph Optimizations**
• TVM Tensor IR Optimizations
• Evaluation
Computation Graph Optimization

Represent high-level deep learning computations

Target Independent
- Constant propagation
- Dead code elimination
- Operation fusion

Target Dependent
- Data layout transform
- Graph partitioning
- Legalization
Operation Fusion Example
Target Dependent Layout Transformation

Observation - Data layout NCHWc leads to better memory accesses

ATC’19 Optimizing CNN Model Inference on CPUs
Agenda

• Overview of TVM
• Relay Graph Optimizations
• TVM Tensor IR Optimizations
• Evaluation
TVM Tensor IR

- Compute definition

\[
C = \text{tvm.compute}((m, n), \\
    \text{lambda } i, j: \text{tvm.sum}(A[i, k] \times B[k, j], \text{axis=k}))
\]

- TVM Schedule – *Developer-friendly loop transformations*

  - *Do not need hardware ISA knowledge to perform loop optimizations*

\[
s = \text{tvm.create_schedule}(\text{C.op}) \\
\text{xo, yo, xi, yi} = s[\text{C}].\text{tile}(\text{C.op.axis}[0], \text{C.op.axis}[1], \text{bn}, \text{bn}) \\
\text{ko, ki} = s[\text{C}].\text{split}(k, \text{factor=4}) \\
s[\text{C}].\text{reorder}(\text{ko, xi, ki, yi}) \\
s[\text{C}].\text{unroll}(\text{ki}) \\
s[\text{C}].\text{vectorize}(\text{yi}) \\
s[\text{C}].\text{parallel}(\text{xo}) \\
\ldots
\]
Large Search Space for Schedule

Large search space for optimization choices

Compute Description

```python
C = tvm.compute((m, n),
                lambda y, x: tvm.sum(A[k, y] * B[k, x], axis=k))
```

- Loop Transformations
- Thread Bindings
- Cache Locality
- Thread Cooperation
- Tensorization
- Latency Hiding

Hardware
AutoTVM - Learning-based Program Optimizer

- Relatively low experiment cost
- Domain-specific problem structure
- Large quantity of similar tasks

NeurIPS’19 Learning to Optimize Tensor Programs
Agenda

- Overview of TVM
- Relay Graph Optimizations
- TVM Tensor IR Optimizations
- **Evaluation**
Experiment Setup

- **Server - EC2 A1 Instance**
  - 16-core ARMv8

- **Edge device - Acer aiSage**
  - Rockchip RK3399 SoC + ARM Mali GPU T-860
TVM – Evaluation on ARM A1 Server

Speedup of TVM execution normalized to Tensorflow-Eigen
TVM – Evaluation on Edge Device Acer aiSage

Baseline: ACL
Batch size = 1

ICPP’19 A Unified Optimization Approach for CNN Model Inference on Integrated GPUs
Effects of Tuning Convolution operators in TVM

Batch size = 1

Acer aiSage

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>w/o</th>
<th>w</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet50</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MobileNet</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SqueezeNet</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Speedup

ICPP’19 A Unified Optimization Approach for CNN Model Inference on Integrated GPUs
Takeaways

- Deep learning compilation is essential for portability and performance across a variety of targets.

- Optimizations are important at all levels – graph- and tensor-level.

- Abstracting compute and HW-dependent schedule enables developers to write kernels without extensive knowledge of HW ISA.

- Open-source collaborations are the key to achieve the dream of running deep learning everywhere.
Linaro Community

- TVM enjoys LLVM ARM codegen support
  - Better support for Int8 instructions
  - Better support for different ARM variants

- Better schedules
  - Data layout optimizations are hardware dependent
  - TVM performance of ARM GPUs can be improved

- ACL support for TVM
Thank you!

https://github.com/neo-ai/
https://github.com/dmlc/tvm