Performance Evaluation of Deep Learning Compilers for Edge Inference

21st May, PAISE 2021

Gaurav Verma\textsuperscript{1}, Yashi Gupta\textsuperscript{1}, Abid M. Malik\textsuperscript{2}, Barbara Chapman\textsuperscript{1,2}

\textsuperscript{1}Stony Brook University, NY, \textsuperscript{2}Brookhaven National Laboratory, NY

Acknowledgement: This research work is supported in part by the U.S. Office of the Under Secretary of Defense for Research and Engineering (OUSD(R&E)) under agreement number FA8750-15-2-0119.
Highlights And Contributions

- This work presents a detailed performance analysis of TensorFlow Lite and TensorFlow-TensorRT (TF-TRT) inference compilers by comparing throughput, latency, and power consumption.

- It describes inference compilers’ behavior specific to DL model's architecture and computing hardware.

- It recognizes a need for a standardized benchmark suite to analyze the inference compilers’ optimization pipeline for edge computing.

- The results presented in this paper will provide scientific computing community solutions to optimize the inference performance at Edge.
Background And Motivation

- Challenges in the cloud-based inference
  - Limited bandwidth when compared to volume of data
  - Undesirable high latency
  - Security and privacy concerns

- Together with our colleagues at PVAMU, we perform research to build the capabilities addressing complex problems posed by IoT, ML and Big Data.

http://credit.pvamu.edu/

An Industrial IOT Example

Can we “efficiently” bring computation to the Edge??
Edge Computing And Deep Learning

- Edge Computing
  - “It is a distributed computing paradigm that brings computation and data storage closer to the location where it is needed to improve response times and save bandwidth”

- A surge in the Deep Learning-based applications on edge
  - Object detection
    - E.g., UAV tracking a moving target
  - Image Classification
    - E.g., Surveillance tech using CCTV

- Application in high energy physics
  - Real-time image analysis in HL-LHC and DUNE
Improving Efficiency Of Inference-on-Edge

- Computing hardware of the edge devices is usually CPUs, GPUs, or ASICs
- These xPUs are limited in compute and power resources compared to the cloud servers
- Research to enable efficient deep learning inference on the resource-constrained edge devices
  - Development of low-power SoCs specialized for deep learning. E.g., Google’s TPU, Intel’s VPU
  - Model compression techniques, like quantization, layer pruning
  - Design of lightweight models like MobileNet, YOLO
- The above approaches have limitations in addressing heterogeneous hardware and models
- Need for frameworks to implement fine-grained optimizations common across DL models
Need For Compiler Frameworks For Inference-on-Edge

- Diversified DL compute hardware/backend
- Challenges in deploying DL models in varying input formats
- Need for DL-oriented multi-IR to apply commonly adopted frontend and backend optimization techniques

**Compiler Framework**
- Performs target independent optimizations.
- Precision Calibration
- Layer Fusion
- Kernel Auto-tuning
- Dynamic Tensor Memory ...

**Optimized Graph Runtime**
- To be deployed on the Edge Devices
- Fast, Memory & Power-Efficient Execution

**DNN in the supported frameworks**
- TF, PyTorch, Caffe, ...

**Resource-Constrained Edge Devices**
- TensorFlow Lite
- Tensor RT
- TVM
- Glow
Compiler Frameworks For Inference-on-Edge

- The development of compilers for DNN is currently a hot research topic
  - GLOW, nGraph, TF XLA, TensorRT, TVM...

- Edge computing systems and tools
  - Cloudlet, SpanEdge, AirBox, Apache Edgent, Azure IoT Edge
  - Critical design issues like multi-user fairness, security, privacy, so on

- Inference on the edge is hindered by resource-constrained devices – introducing bandwidth, throughput, power, or efficiency-related challenges

- Development of TensorFlow-TensorRT (TF-TRT) integrated solution and TensorFlow Lite (TFLite) to optimize inference on the edge
Overview Of DL Compilers For Inference-on-Edge

- TensorFlow-TensorRT Integrated Solution (TF-TRT)

- TensorFlow Lite (TFLite)
TensorFlow-TensorRT Integrated Solution

- TensorRT – CUDA-based SDK for high-performance deep learning inference
- It is tightly integrated with TensorFlow
- Provides ONNX support
- Supports input in various frameworks like Caffe, MxNet, Chainer, PyTorch, etc
- Performs optimizations specific to Nvidia GPUs only
TensorRT Workflow

Import and optimize trained models to generate optimized plans

Deployment of generated inference engines

Source: NVIDIA TensorRT Documentation
TensorFlow Lite

Architecture of TensorFlow Lite
Pivotal Optimizations

- Quantization – FP16, INT8, MIXED Precision
- Horizontal and vertical layer and tensor fusion
- Dynamic tensor memory allocation
- TensorRT performs calibration to reduce the accuracy loss
- TensorRT performs kernel autotuning
- TFLite performs weights clustering, reducing the number of individual weights
Vertical And Horizontal Layer Fusion – An Example

An example of CNN

Vertical Layer Fusion

Horizontal Layer Fusion

Source: https://developer.nvidia.com/blog/deploying-deep-learning-nvidia-tensorrt/
Experimental Setup

- Evaluated Models
- Dataset
- Hardware Specifications
- Software Specifications
Evaluated Models

- **Image Classification**
  - ResNet50_v2
  - MobileNet_v2

- **Object Detection**
  - SSD_MobileNet_v2

Dataset

- **ImageNet**
  - Collection of human-annotated images organized according to the WordNet Hierarchy
  - Suited for computer vision applications such as image classification and object detection
  - Over 14 million images organized into 21,000 subcategories

- **Common Object in Context (COCO)**
  - Microsoft’s COCO dataset is large-scale object detection, segmentation, and captioning dataset
  - Consists of everyday scenes comprising common objects in their natural context
  - 165K+ train, 81K+ test and 81K+ validation images
Hardware And Software Specifications

- **Compute Backend**
  - NVIDIA GPUs
    - GeForce RTX 2080
    - Tesla T4
  - Android Studio Emulator
    - Android Studio v4.0.1
    - Pixel 3a XL w/ android 10 and API 29

- **TF-TRT Integrated Solution**
  - TensorFlow v2.4
  - CUDA v10.1
  - CuDNN v7.5

- **TFLite**
  - TFLite v2.3
  - CUDA v10.1
  - CuDNN v7.5
Results and Discussion
Evaluation Metrics

- Throughput: volume of inferences within a given period (images/sec)
- Latency: execution time to perform inference on one image (milliseconds, ms)
- Power: refers to the power drawn by the GPU to perform one inference (watt, W)
- Model Size: saved model’s (.pb or .tflite) size on the disk (MB)
Comparison Between TFLite And TF-TRT On GeForce RTX 2080

ResNet50_v2 model trained on ImageNet dataset
Comparison Between TFLite And TF-TRT On GeForce RTX 2080 – Avg Power And Model Size

SSD_MobileNet_v2 model pre-trained on COCO Dataset provided by MLPerf.
Comparison Between GeForce 2080 And Tesla T4

Average Throughput (imgs/sec) on GeForce 2080 (non-tensor core) vs. Tesla T4 (tensor core) GPU
## Execution Of TFLite Models On An Android Device

<table>
<thead>
<tr>
<th>Backend</th>
<th>Model</th>
<th>Precision</th>
<th>Avg_Throughput (imgs/sec)</th>
<th>Avg_latency (ms)</th>
<th>Model_Size (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPU</td>
<td>MobileNet</td>
<td>Floating</td>
<td>745</td>
<td>11</td>
<td>17</td>
</tr>
<tr>
<td>CPU</td>
<td>MobileNet</td>
<td>Floating</td>
<td>311</td>
<td>26</td>
<td>17</td>
</tr>
<tr>
<td>GPU</td>
<td>MobileNet</td>
<td>Quantized</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
</tr>
<tr>
<td>CPU</td>
<td>MobileNet</td>
<td>Quantized</td>
<td>571</td>
<td>15</td>
<td>4</td>
</tr>
<tr>
<td>GPU</td>
<td>SSD_MobileNet</td>
<td>Floating</td>
<td>42</td>
<td>24</td>
<td>27</td>
</tr>
<tr>
<td>CPU</td>
<td>SSD_MobileNet</td>
<td>Floating</td>
<td>20</td>
<td>53</td>
<td>27</td>
</tr>
</tbody>
</table>

* 4 Threads; NS: Not Supported
Conclusions And Future Research Directions

- Compiler frameworks have been proved vital in applying fine-grained and low-level optimizations to the DL models at the edge.

- TF-TRT-integrated solution consistently displays better performance with different computing backend, especially with GPUs using tensor cores.

- TFLite performs better with lightweight DL models compared to the deep CNN models.

- Inference compilers such as TF-TensorRT and TensorFlow Lite under a scientific ML workload can prove crucial for the scientific community.

- Extend this experimentation for modern AI accelerators. For example, Application-Specific Integrated Circuit (ASIC) such as Vision Processing Unit (VPU), and Tensor Processing Unit (TPU), and FPGAs.
Thank You!

Acknowledgement: This research work is supported in part by the U.S. Office of the Under Secretary of Defense for Research and Engineering (OUSD(R&E)) under agreement number FA8750-15-2-0119.

Gaurav Verma: gverma@cs.stonybrook.edu