Accelerating Model Optimization on the Edge Through Automated Performance Benchmarking and End-to-End Profiling

Nayara Aguiar, PhD
Performance Engineer
MathWorks
When does performance become a concern in your AI deployment pipeline?
Evaluating performance throughout the development process enables early detection of bottlenecks.

- Tracking performance makes it easier to pinpoint the source of issues.
- Mitigation of performance issues is less costly with early detection.
- Quality targets for final product can be met while enhancing development process.
Optimizing performance of the AI deployment pipeline presents multiple challenges

- Dependencies
  - Multi-step process
  - Cross-team collaboration

- Definition
  - Speed
  - Energy
  - Accuracy
  - Memory

- Standardization
  - Tools
  - HW/SW Stack
  - Processes
We will focus on our ongoing efforts to enhance the process for performance benchmarking and profiling
Integrating MLPerf™ to our internal performance benchmarking strategy helped us standardize tooling.

Source: https://mlcommons.org/
We used automation to enhance the benchmarking process.

Network
- resnet50

BatchSize
- 128

TargetConfiguration
- TargetLanguage: C++
- Hardware: RaspberryPi

BenchmarkMode
- PerformanceOnly
  - SingleStream

Prepare High-Level Code

Generate C++ Code

Deploy Code to Device

Invoke MLPerf™ Benchmark

Results
- MLPerf™ Logs
This benchmark can also be wrapped in our testing infrastructure for further automation

1. Select pre-trained models:
   - mobilenetv2
   - resnet50

2. Create configurations for Raspberry Pi runs:
   - Using original network (FP32)
   - Using quantized network (INT8)
   - Using network equalized before quantization (INT8)

3. Batch size:
   - 1024

4. Select benchmark mode:
   - AccuracyOnly, SingleStream

<table>
<thead>
<tr>
<th>Network</th>
<th>BatchSize</th>
<th>TargetConfiguration</th>
<th>BenchmarkMode</th>
<th>Original (FP32)</th>
<th>Quantized (INT8)</th>
<th>Equalized (INT8)</th>
</tr>
</thead>
<tbody>
<tr>
<td>mobilenetv2</td>
<td></td>
<td>configA, configB</td>
<td>mode1, mode2</td>
<td>70.3%</td>
<td>0.2%</td>
<td>60.3%</td>
</tr>
<tr>
<td>resnet50</td>
<td></td>
<td>configA, configB</td>
<td>mode1, mode2</td>
<td>72.2%</td>
<td>69.3%</td>
<td>68.9%</td>
</tr>
</tbody>
</table>
Automated benchmarks allow for performance monitoring … and profiling helps investigating performance bottlenecks

<table>
<thead>
<tr>
<th>Product version</th>
<th>Image Classification Inference</th>
<th>90th percentile latency (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>v1</td>
<td>505.4</td>
<td></td>
</tr>
<tr>
<td>v2</td>
<td>505.3</td>
<td></td>
</tr>
<tr>
<td>v3</td>
<td>502.8</td>
<td></td>
</tr>
<tr>
<td>v_{future}</td>
<td></td>
<td>502.8</td>
</tr>
</tbody>
</table>
We will focus on our ongoing efforts to enhance the process for performance benchmarking and profiling.

Automating Performance Benchmarking

Enabling End-to-End Performance Profiling
We develop tools that cover the entire AI deployment pipeline

To facilitate end-to-end performance investigations, we developed the Unified Timeline
The *Unified Timeline* facilitates performance analysis across the AI deployment pipeline.

- Generic
- Easy to use
- Continuously enhanced
We can use our in-house profiling tool to investigate performance bottlenecks.

Network

Prepare High-Level Code

Inference for image classification

tic
output = net.predict(img);
toc

BatchSize

profile on
output = net.predict(img);
profile off

Total time ~ 700 ms

Time breakdown
We can visualize the timeline for this code in a browser and zoom in stacks of interest

Inference for image classification

Can this be optimized?
Make performance a core aspect for AI applications on the edge

- Automate performance benchmarking
- Explore performance profiling tools

... accelerate model optimization on the edge
Copyright Notice

This presentation in this publication was presented at the tinyML® Summit 2024. The content reflects the opinion of the author(s) and their respective companies. The inclusion of presentations in this publication does not constitute an endorsement by tinyML Foundation or the sponsors.

There is no copyright protection claimed by this publication. However, each presentation is the work of the authors and their respective companies and may contain copyrighted material. As such, it is strongly encouraged that any use reflect proper acknowledgement to the appropriate source. Any questions regarding the use of any materials presented should be directed to the author(s) or their companies.

tinyML is a registered trademark of the tinyML Foundation.

www.tinyml.org