Arm AI Virtual Tech Talk

Improve PyTorch App Performance with Android NNAPI Support

Koki Mitsunami
December 14th, 2021
Welcome!

Tweet us: @ArmSoftwareDev -> #AIVTT

Check out our Arm Software Developers YouTube channel

Signup now for our next AI Virtual Tech Talk: www.arm.com/techtalks
## Our upcoming Arm AI Tech Talks

<table>
<thead>
<tr>
<th>Date</th>
<th>Title</th>
<th>Host</th>
</tr>
</thead>
<tbody>
<tr>
<td>December 14th</td>
<td>Improve PyTorch App Performance with Android NNAPI Support</td>
<td>Arm</td>
</tr>
<tr>
<td>January 11, 2022</td>
<td>Qeexo AutoML on Arm Cortex-M0+: Bringing TinyML to the tiniest Arm MCUs</td>
<td>Qeexo</td>
</tr>
<tr>
<td>January 25, 2022</td>
<td>Peaks, Valleys and Thresholds: The art of segmenting real-time sensor data for tinyML classification, regression and anomaly detection</td>
<td>Reality AI</td>
</tr>
<tr>
<td>February 8, 2022</td>
<td>Faster time-to-production for computer vision AI on Arm powered edge devices, but just how fast is fast?</td>
<td>Deeplite</td>
</tr>
<tr>
<td>February 22, 2022</td>
<td>* New Arm ML Quarterly Research Special *&lt;br&gt;Federated Learning Based on Dynamic Regularization to Debias Model Updates</td>
<td>Arm ML Research</td>
</tr>
</tbody>
</table>

Visit: [www.arm.com/techtalks](http://www.arm.com/techtalks)
• Contribute to enabling a successful developer experience with Arm-based technology

• Demonstrated multiple applications using deep learning focusing computer vision

• Enjoy working on hardware, software, and algorithms

Koki Mitsunami
Staff Engineer @ Arm
Cambridge, UK
Agenda

• Introduction
  • Purpose of this talk
  • PyTorch
  • Android NNAPI

• PyTorch Mobile with NNAPI
  • Workflow
  • Benchmark Conditions
  • Execution Time Comparison
  • Profiling by Streamline

• Summary
Purpose of this talk

• Introduce PyTorch Mobile with Android NNAPI
  • One way of deploying ML models into mobile devices

• Provide examples on how ML models are executed on mobile devices through PyTorch with NNAPI
PyTorch

- PyTorch has become popular Deep Learning framework
  - Wide range of supported operators
  - Ease of writing
  - A lot of activity for production

Source: https://paperswithcode.com/trends
Android Neural Networks API (NNAPI)

• C API designed for running ML models
  • Provide a base layer of functionality
  • Perform hardware-accelerated inference operations on supported devices

• Typically used by ML frameworks
  • Apps would not use NNAPI directly
  • Apps would use ML frameworks
Android NNAPI support is now available in Beta
Announced in PyTorch 1.10 Release

Prototype (Nov 2020)
- NNAPI with the PyTorch framework
- Android 10+
- Linear Convolution models
- Multi-Layer Perceptron models

Beta (Oct 2021)
- Additional operator types
- Support for load-time flexible shapes
- Run models on host for testing

PyTorch Feature Classification
Prototype → Beta → Stable

https://pytorch.org/blog/prototype-features-now-available-apis-for-hardware-accelerated-mobile-and-arm64-builds/
https://pytorch.org/blog/pytorch-1.10-released/#beta-android-nnapi-support-in-beta
PyTorch Mobile

• Mechanism to run ML models on mobile
  • Provides end-to-end workflow within PyTorch ecosystem
  • Support for Arm CPU/GPU and hardware accelerators via XNNPACK/QNNPACK, Vulkan, and NNAPI

• TorchScript format
  • Intermediate representation (IR) of a PyTorch model
    –includes code, parameters, attributes, and debug information
  • Can be run in a high-performance environment such as C++

TorchScript example

```
graph(%0 : Float(3, 10), %1 : Float(3, 20), %2 : Float(3, 2)
%3 : Float(30, 10), %4 : Float(30, 20),
%5 : Float(30, 20), %6 : Float(30)) {
  %7 : Float(30, 20) = aten::t(%3)
  %8 : int[] = prim::ListConstruct(3, 80)
  %9 : Float(30, 20) = aten::expand(%5, %10, 1)
  %11 : Float(30, 20) = aten::addmm(%2, %9, %7, 1, 1)
  %12 : Float(30, 20) = aten::add(%11, %11, %7, 1, 1)
  %13 : Float(30, 20) = aten::add(%12, %7, 1, 1)
  %14 : Dynamic() = aten::chunk(%13, 4, 1)
  %15 : Float(30, 20), %31 : Float(30, 20),
  %32 : Float(30, 20), %33 : Float(30, 20) = prim::ListUng
ek
  %34 : Float(30, 20) = aten::sigmoid(%32)
  %35 : Float(30, 20) = aten::sigmoid(%31)
  %36 : Float(30, 20) = aten::tanh(%32)
```
Workflow for PyTorch Mobile with NNAPI

1. Create/prepare a model in PyTorch
2. Quantization (optional)
3. Convert to TorchScript
4. Optimize for Mobile (optional)
5. Convert to NNAPI-compatible model
6. Save the model
   - Treat as a TorchScript model
   - For applications already using PyTorch Mobile, no code changes are required

Example)

```python
model = torchvision.models.quantization.
        mobilenet.mobilenet_v2 \
        (pretrained=True, quantize=True)
input_tensor = input_tensor.contiguous\ 
              (memory_format=torch.channels_last)
input_tensor.nnapi_nhwc = True
traced = torch.jit.trace(model, input_tensor)
nnapi_model = torch.backends._nnapi.prepare. \ 
             convert_model_to_nnapi(traced, input_tensor)
nnapi_model._save_for_lite_interpreter\ 
             ("mobilenetv2-nnapi.pt")
```

Prepare a pre-trained quantized model

Convert to TorchScript

Convert to NNAPI model

Save the model

Deploy the model to mobile
Experiments - PyTorch Mobile with NNAPI

- ML Model
  - MobileNet v2
    - CPU models with Float32 and Int8
    - NNAPI models with Float32 and Int8

- Mobile devices
  - 8 mobile devices
    - 2 for Android 11
    - 6 for Android 10

- Execution time
  - Avg time over 200 iter

Model Generation Flow

- MobileNet v2 PyTorch Model
  - Quantization
  - Convert to TorchScript
    - Optimize for CPU
      - CPU Model with Float32
      - CPU Model with Int8
    - Convert to NNAPI Model
      - NNAPI Model with Float32
      - NNAPI Model with Int8

(PyTorch 1.8.0.dev + torchvision 0.9.0.dev)
NNAPI Speed-up on Various Devices

- Speed increase varies from device to device
  - Faster processing with the use of NNAPI on many devices

(As of April 2021)
Profiling with Streamline Performance Analyzer

- Arm Streamline allows you to see CPU and GPU activity inside device
Streamline Profiling with NNAPI Models (Device 1)

- NNAPI selects most performant hardware
  - GPU is used for Float32, while multi-core CPU is used for Int8
Streamline Profiling with NNAPI Models (Device 2)

- The same model can work differently on different devices by using NNAPI
  - NPU/DSP is used with Int8, which is not visible with Streamline
Hardware selected by NNAPI

- Quantized models are more likely to benefit from specialized processors
  - Int8 tends to have higher speed increase

(As of April 2021)
Arm NN and Arm Compute Library (ACL)

- Many Android devices have ML models accelerated by underlying Arm NN and ACL
- Superior Performance & Arm Specific Optimization
  - Uses advanced network optimization techniques
  - Quick adoption of new Arm technologies e.g. Armv9-A features
Faster Machine Learning in Android 12

• More than halved inference call overhead by introducing improvements
  • Such as padding, sync fences and reusable execution objects

• Made ML accelerator drivers updatable outside of platform releases
  • Make it easier for developers to take advantage of the latest drivers
  • ML performance improvements and bug fixes reach users faster than ever before

Summary

- Trends around PyTorch
  - Increasing popularity, especially in research field
  - Working to bridge the gap between research and production

- Advantage of NNAPI
  - Provides a single set of APIs
  - Select most performant hardware for running ML models for each device
    - Including GPUs, DSPs, and NPUs
  - Expected to support more and more in the future

- PyTorch with Android NNAPI
  - Moved forward from Prototype phase to Beta phase
  - Provides workflow that simplifies research to production within PyTorch ecosystem
Welcome!

Tweet us: @ArmSoftwareDev -> #AIVTT

Check out our Arm Software Developers YouTube channel

Signup now for our next AI Virtual Tech Talk: www.arm.com/techtalks