FEEDFORWARD Volume 1 No. 1 January – March 2022

A Lean Approach to Embedded Deep Learning

A. Gokhberg
FRAGATA COMPUTER SYSTEMS AG

***Abstract*—This article describes Arhat, a software framework providing innovative implementation for deployment of deep learning workflows. Unlike the conventional deep learning frameworks Arhat translates neural network descriptions directly into lean standalone executable code. Using Arhat, it is possible to generate executable code for each specific combination of a model and a target platform. This approach yields a very compact code and can greatly simplify the deployment process. Arhat supports a wide range of neural network operations and multiple target platforms including Intel CPUs and GPUs (via oneDNN) and NVIDIA GPUs (via CUDA/cuDNN or TensorRT inference library). Arhat is integrated with Intel deep learning software ecosystem including OpenVINO Toolkit and oneAPI Deep Neural Network Library (oneDNN). We have evaluated Arhat on Intel Tiger Lake i7-1185G7E and NVIDIA Jetson Xavier systems in the embedded object detection problem domain.**

■ **THE INTRODUCTION** Efficient deployment of modern deep learning solutions represents substantial challenges caused by high fragmentation of corresponding hardware and software ecosystems. This fragmentation makes deployment of each deep learning model a substantial engineering project and often requires using cumbersome software stacks.

Currently, the deep learning ecosystems represent heterogeneous collections of various software and hardware components, which include: (1) training software frameworks that produce trained models, (2) software frameworks designed specifically for inference, (3) exchange formats for neural networks, (4) computing hardware platforms from different vendors, and (5) platform-specific low level programming tools and libraries. These components are evolving at a very rapid pace. Compatibility and interoperability of individual components is frequently limited. There is a clear need for a streamlined approach for navigating in this complex world.

To address these challenges we have developed Arhat, a cross-platform framework for efficient deployment of deep learning inference workflows in the cloud and on the edge. Unlike the conventional deep learning frameworks Arhat translates neural network descriptions directly into lean standalone executable code.

In developing Arhat, we pursued two principal objectives: (1) providing a unified platform-agnostic approach towards deep learning deployment and (2) facilitating performance evaluation of various platforms on a common basis.

In this article we discuss design and architecture of Arhat and demonstrate its use for deployment of object detection models for embedded applications.

Reference architecture

Conceptual Arhat architecture is shown on **Figure 1**.

The central component is Arhat engine that receives model descriptions from various sources and steers backends generating the platform-specific code.

There are two principal ways for obtaining model descriptions. (1) Arhat provides the front-end API for programmatic description of models. (2) Pre-trained models can be imported from the external frameworks via bridge components supporting various exchange formats.

The interchangeable backends generate code for various target platforms. Backends for Intel, NVIDIA, and AMD hardware are currently available.

This architecture is extensible and provides regular means for adding support for new model layer types, exchange formats, and target platforms.

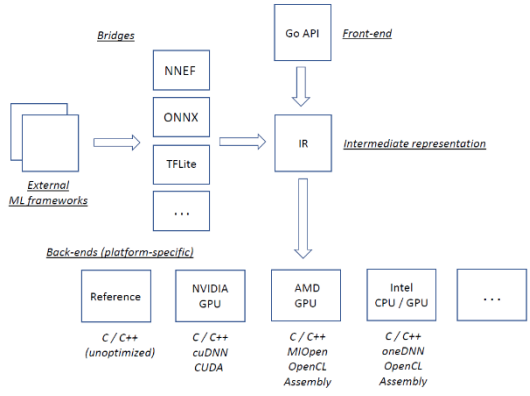

Figure 1. Arhat reference architecture.

Arhat core is implemented in pure Go programming language. Consequently, unlike most conventional platforms that use Python, Arhat uses Go also as a language for the frontend API. This API specifies models by combining tensor operators selected from an extensible and steadily growing operator catalog. The API defines two levels of model specification. The higher level is common for all platforms and workflows. The lower level is platform- and workflow-specific. The Arhat engine performs automated conversion from the higher to the lower level.

Support for Intel platforms via oneDNN

Intel oneAPI Deep Neural Network Library (oneDNN) [1] is a cross-platform library implementing basic deep learning operations optimized for various Intel CPU and GPU platforms.

Arhat supports Intel hardware via oneDNN backend. The architecture of this backend is shown on **Figure 2**.

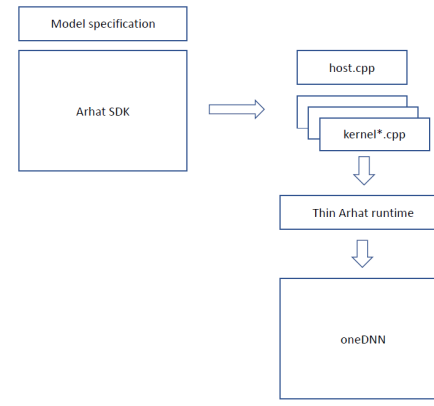

Figure 2. Arhat oneDNN backend.

The backend translates model specification into C++ code consisting of one host module and multiple kernel modules. This code runs on the top of a thin Arhat runtime that directly interacts with oneDNN. This approach results in a very slim deployable software stack that can run on any Intel hardware supporting oneDNN.

Interoperability with Intel OpenVINO

Intel OpenVINO toolkit [2] allows developers to deploy pre-trained models on various Intel hardware platforms. It includes the Model Optimizer tool that converts pre-trained models from various popular formats to the uniform Intermediate Representation.

We leverage OpenVINO Model Optimizer for supporting various model representation formats in Arhat. For this purpose, we have designed the OpenVINO IR bridge that can import models produced by the OpenVINO Model Optimizer. This immediately enables Arhat to handle all model formats supported by OpenVINO. The respective workflow is shown on **Figure 3**.

This development, on the one hand, integrates Arhat into Intel deep learning ecosystem and, on the other hand, makes OpenVINO capable of the native inference on platforms of different vendors. With Arhat, the end-users can view OpenVINO as a vendor-neutral solution. This might be beneficial for adoption of OpenVINO as an inference platform of choice.

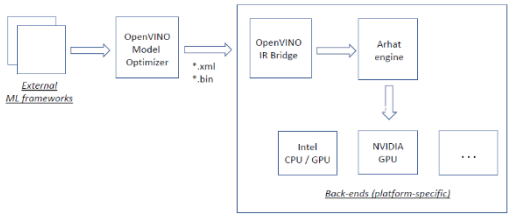

Figure 3. Arhat interoperability with OpenVINO.

Case study: object detection

We have validated the approach using as a case study object detection on embedded platforms. For this study we have selected 8 representative pre-trained models of 3 different families (SSD, Faster R-CNN, and YOLO) from Intel Open Model Zoo [3]. We used Arhat to generate code for Intel and NVIDIA and evaluated it on the respective embedded platforms.

We started with performance comparison of Arhat code powered by oneDNN against the open-source version of OpenVINO.

As a platform we used Tiger Lake i7-1185G7E, a system combining four CPU cores with integrated GPU. To obtain stable results and emulate embedded environment we have disabled turbo boost. We have measured average processing time for one image. Separate test runs have been made for the CPU cores and the GPU. As a baseline, we also included metrics for the ResNet50 image classification network.

Results for the CPU and the GPU are summarized in **Tables 1 and 2**. The numbers represent time, in milliseconds, required to process one input image.

Table 1. Performance evaluation on Tiger Lake, multi-core CPU

Model	Arhat	OpenVINO
resnet50-v1-7	23.5	23.0
ssd_mobilenet_v1_fpn_coco	397	401
ssd_mobilenet_v2	14.7	14.9
ssd_resnet50_v1_fpn_coco	563	568
faster_rcnn_inception_v2_coco	244	236
faster_rcnn_resnet50_coco	722	719
yolo_v2_tf	188	190
yolo_v3_tf	205	201
yolo_v4_tf	409	427

Table 2. Performance evaluation on Tiger Lake, integrated GPU

Model	Arhat	OpenVINO
resnet50-v1-7	10.1	11.3
ssd_mobilenet_v1_fpn_coco	144	125
ssd_mobilenet_v2	9.0	10.7
ssd_resnet50_v1_fpn_coco	195	183
faster_rcnn_inception_v2_coco	117	82.8
faster_rcnn_resnet50_coco	252	259
yolo_v2_tf	64.4	60.8
yolo_v3_tf	72.3	68.2
yolo_v4_tf	140	135

These tables demonstrate that Arhat with oneDNN backend provides inference performance compared to OpenVINO on both multi-code CPU and integrated GPU of Tiger Lake i7-1185G7E.

Interoperability with TensorRT

To run OpenVINO models on NVIDIA GPUs Arhat provides two alternative backends. These backends use CUDA Deep Neural Network library (cuDNN) [4] and TensorRT inference library [5] respectively.

The architecture of TensorRT backend is shown on **Figure 4**.

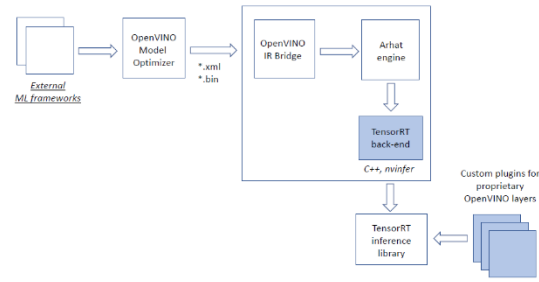

Figure 4. Arhat interoperability with TensorRT.

There are several OpenVINO proprietary layer types that are not directly supported by TensorRT. We have implemented them in CUDA as custom TensorRT plugins. These layer types include *DetectionOutput*, *PriorBoxClustered*, *Proposal*, *RegionYolo* and *RoiPooling*.

Evaluation on Jetson Xavier NX

As the next step, we included the NVIDIA platform in evaluation. We have chosen Jetson Xavier NX. This system has six ARM CPU cores and a GPU. It has various configurable power modes with 10W and 15W TDP envelopes and 2, 4, or 6 cores enabled.

The best performance is demonstrated with 15W and 2 cores, therefore we used this mode for comparison.

Results are summarized in **Table 3**.

Table 3. Performance evaluation on Jetson Xavier NX with Arhat cuDNN and TensorRT backends

Model	cuDNN	TensorRT
resnet50-v1-7	26.7	14.2
ssd_mobilenet_v1_fpn_coco	212	187
ssd_mobilenet_v2	23.4	12.3
ssd_resnet50_v1_fpn_coco	305	248
faster_rcnn_inception_v2_coco	195	159
faster_rcnn_resnet50_coco	413	366
yolo_v2_tf	114	83.3
yolo_v3_tf	139	94.8
yolo_v4_tf	250	191

This tables demonstrates that TensorRT inference library provides superior inference performance compared to cuDNN on Jetson Xavier NX.

Arhat backend for TensorRT enables native deployment of OpenVINO models on NVIDIA GPUs and opens a way for achieving the best OpenVINO performance on the hardware of the other vendors.

Conclusion

This study demonstrates that Arhat can efficiently interoperate with various key components of Intel and NVIDIA deep learning ecosystems. Furthermore, using Arhat allows to overcome limitations of each individual component and achieve results that are difficult or not possible to achieve otherwise. Arhat extends capabilities of Intel deep learning software by providing a way for the native deployment of OpenVINO models on the wider range of platforms.

Arhat can be also used for the streamlined on-demand benchmarking of models on various platforms. Using Arhat for performance evaluation eliminates overhead that might be caused by external deep learning frameworks because code generated by Arhat directly interacts with the optimized platform-specific deep learning libraries.

REFERENCES

1. oneAPI Deep Neural Network Library (oneDNN). [Online] Available: https://github.com/oneapi-src/oneDNN (URL.)
2. OpenVINO Toolkit. [Online] Available: https://github.com/openvinotoolkit/openvino (URL.)
3. Open Model Zoo repository. [Online] Available: https://github.com/openvinotoolkit/open_model_zoo (URL.)
4. NVIDIA cuDNN. [Online] Available: https://developer.nvidia.com/cudnn (URL.)
5. NVIDIA TensorRT. [Online] Available: https://developer.nvidia.com/tensorrt (URL.)

Alexey Gokhberg is a seasoned software engineer with more than 25 years of experience in various industrial and academic branches. His professional interests include deep learning, high-performance computing, programming language construction, and computational geophysics.