Using Arhat framework with Intel® oneDNN library and OpenVINO™ toolkit for object detection applications

Alexey Gokhberg

oneAPI Developer Summit at SC
14.11.2021

Introduction

Arhat:

- specialized deep learning framework
- translates neural network descriptions into lean standalone executable C++ code
- designed for deployment of deep learning inference workflows in the cloud and at the edge

Objectives:

- unified platform-agnostic approach towards deep learning deployment
- performance evaluation of various platforms on a common basis

Challenges:

- highly fragmented deep learning hardware and software landscape
- cumbersome software stacks

Fragmented landscape

Training frameworks:

- PyTorch
- TensorFlow
- CNTK
- Caffe

Inference frameworks:

- OpenVINO (Intel)
- TensorRT (NVIDIA)
- MIGraphX (AMD)
- ONNX runtime

Model exchange formats:

- ONNX
- NNEF
- TensorFlow Lite
- OpenVINO IR (Intel)
- UFF (NVIDIA)

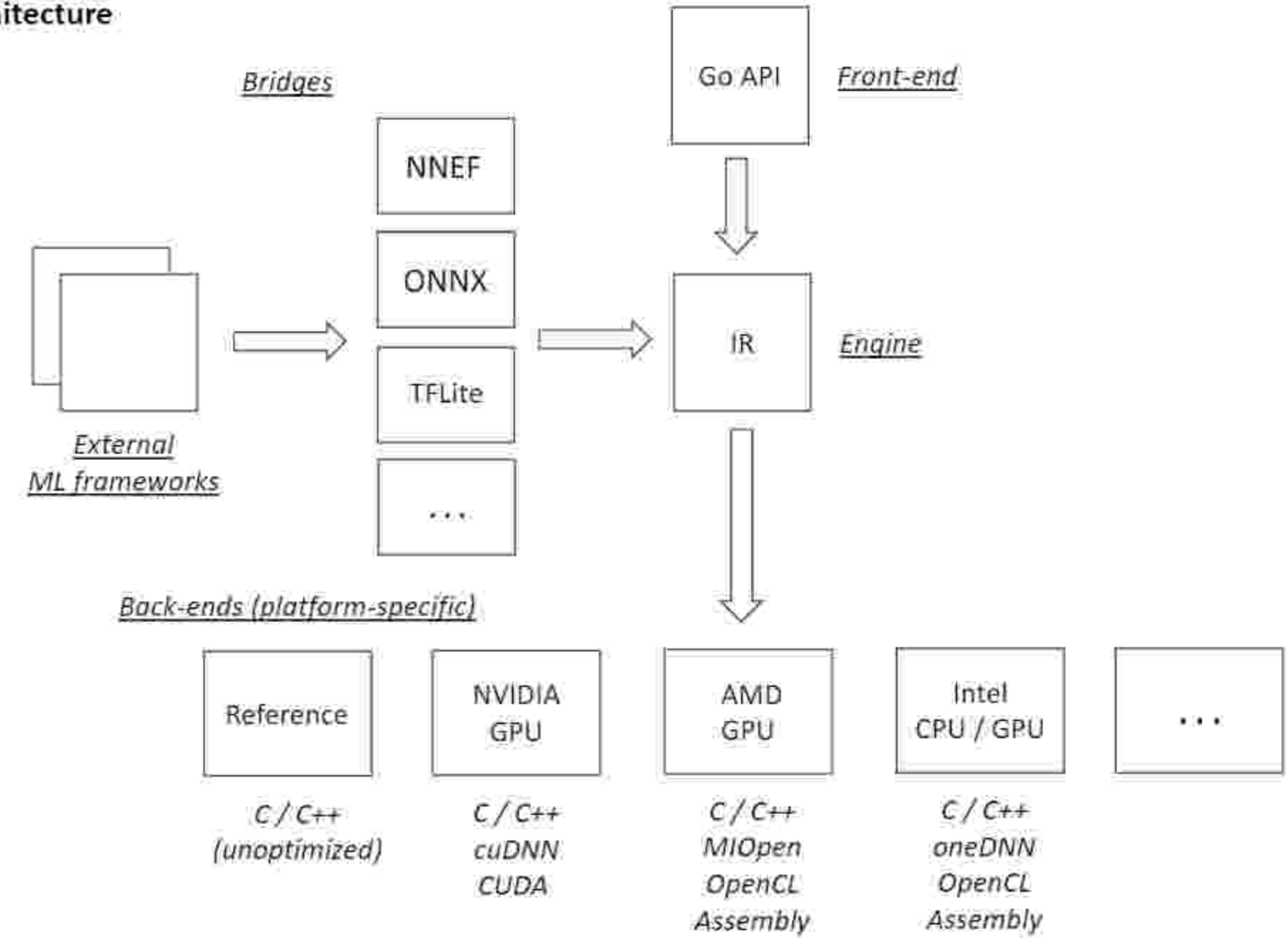
Computing hardware:

- CPU
- GPU
- AI accelerators

Vendor tools and libraries:

- oneAPI / oneDNN (Intel)
- CUDA / cuDNN (NVIDIA)
- HIP / MIOpen (AMD)

Reference architecture



Model specification with Go API

Arhat operator catalog: 120+ types of DNN operators

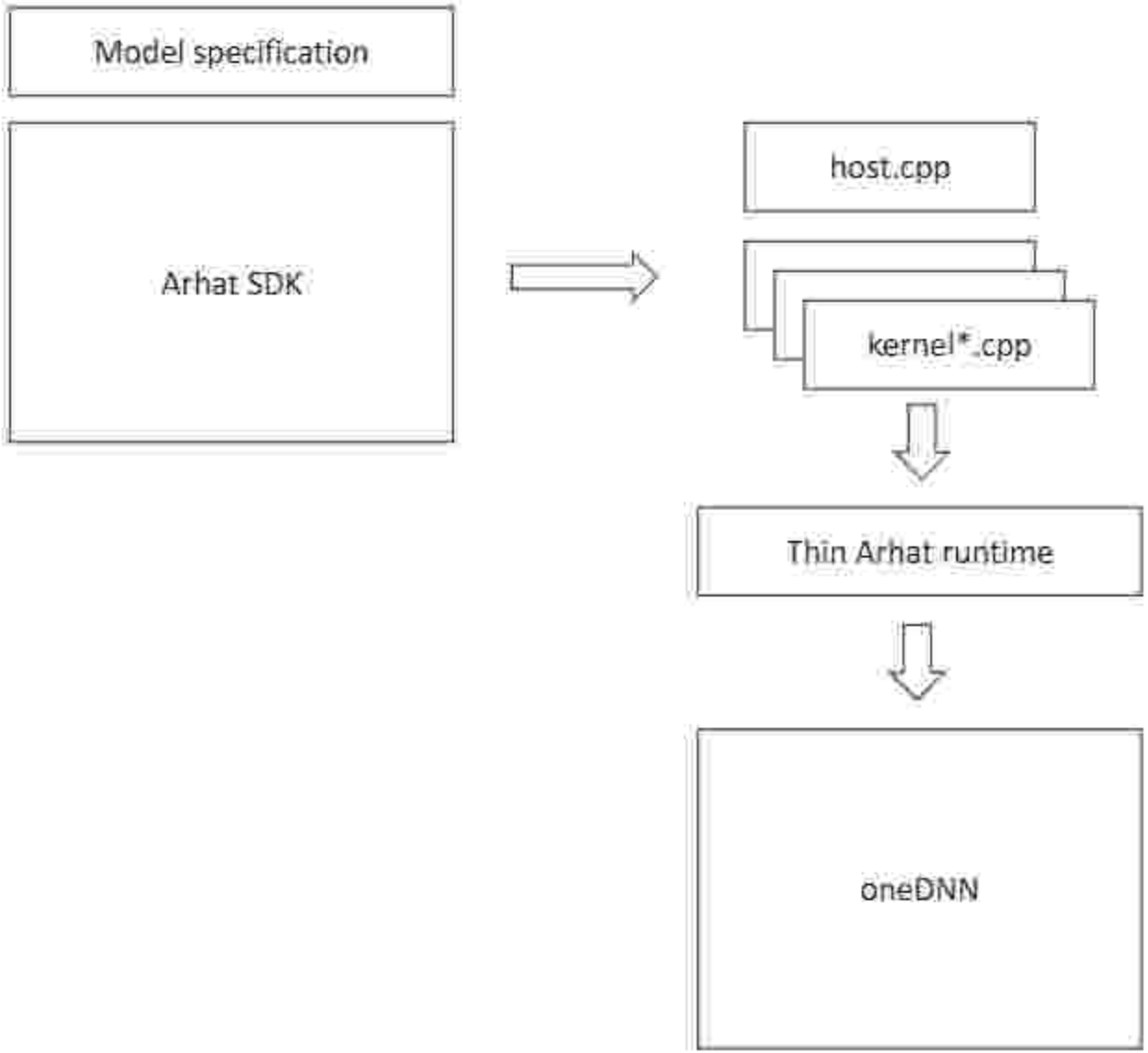
[1] Model level representation (task- and platform-agnostic)

```
import "fragata/arhat/front/models"

func Conv(
        m *models.Model,
        t string,
        xShape []int,
        wShape []int,
        bShape []int,
        kernel []int,
        dilation []int,
        stride []int,
        pads []int,
        group int) {
    x := m.External("dtype", t, "shape", xShape)
    w := m.Variable("dtype", t, "shape", wShape, "label", "weight")
    b := m.Variable("dtype", t, "shape", bShape, "label", "bias")
    y :=
        m.Conv(
            x,
            w,
            b,
            "kernel", kernel,
            "dilation", dilation,
            "stride", stride,
            "pads", pads,
            "group", group)
    m.Return(y, 0)
}
```

[2] Graph level representation (task- and platform-specific)

```
import (
    "fragata/arhat/core"
    "fragata/arhat/graph/api"
    "fragata/arhat/op/base"
)

func Conv(
        g api.Graph,
        t core.Type,
        xShape []int,
        wShape []int,
        bShape []int,
        kernel []int,
        dilation []int,
        stride []int,
        pads []int,
        group int) {
    x := g.NewTensor(t)
    w := g.NewTensor(t)
    b := g.NewTensor(t)
    y := g.NewTensor(t)
    g.External(x, xShape)
    g.Variable(w, wShape, "weight")
    g.Variable(b, bShape, "bias")
    g.ConvOp(x, w, b, y, kernel, dilation, stride, pads,
        base.LegacyPaddingNotSet, group)
    g.Return(y)
}
```

oneDNN back-end



host.cpp

```
Conv1(0, 1, 2, 3);
Relu2(3, 4);
MaxPool3(4, 5);
Conv4(5, 6, 7, 8);
Relu5(8, 9);
MaxPool6(9, 10);
Conv7(10, 11, 12, 13);
Relu8(13, 14);
Conv9(14, 15, 16, 17);
Relu10(17, 18);
Conv11(18, 19, 20, 21);
Relu10(21, 22);
MaxPool13(22, 23);
FullyConnected14(23, 24, 25, 26);
Relu15(26, 27);
FullyConnected16(27, 28, 29, 30);
Relu15(30, 31);
FullyConnected18(31, 32, 33, 34);
Softmax19(34, 35);
```

kernel01.cpp

```
void Conv1(int x, int w, int b, int y) {
    arhat::onednn::InitConv(
        x,
        w,
        b,
        y,
        dnnl::memory::data_type::f32,
        {10, 3, 224, 224},
        {64, 3, 11, 11},
        {64},
        {10, 64, 55, 55},
        {4, 4},
        {0, 0},
        {2, 2},
        {2, 2});
}
```

Proof of concept: image classification

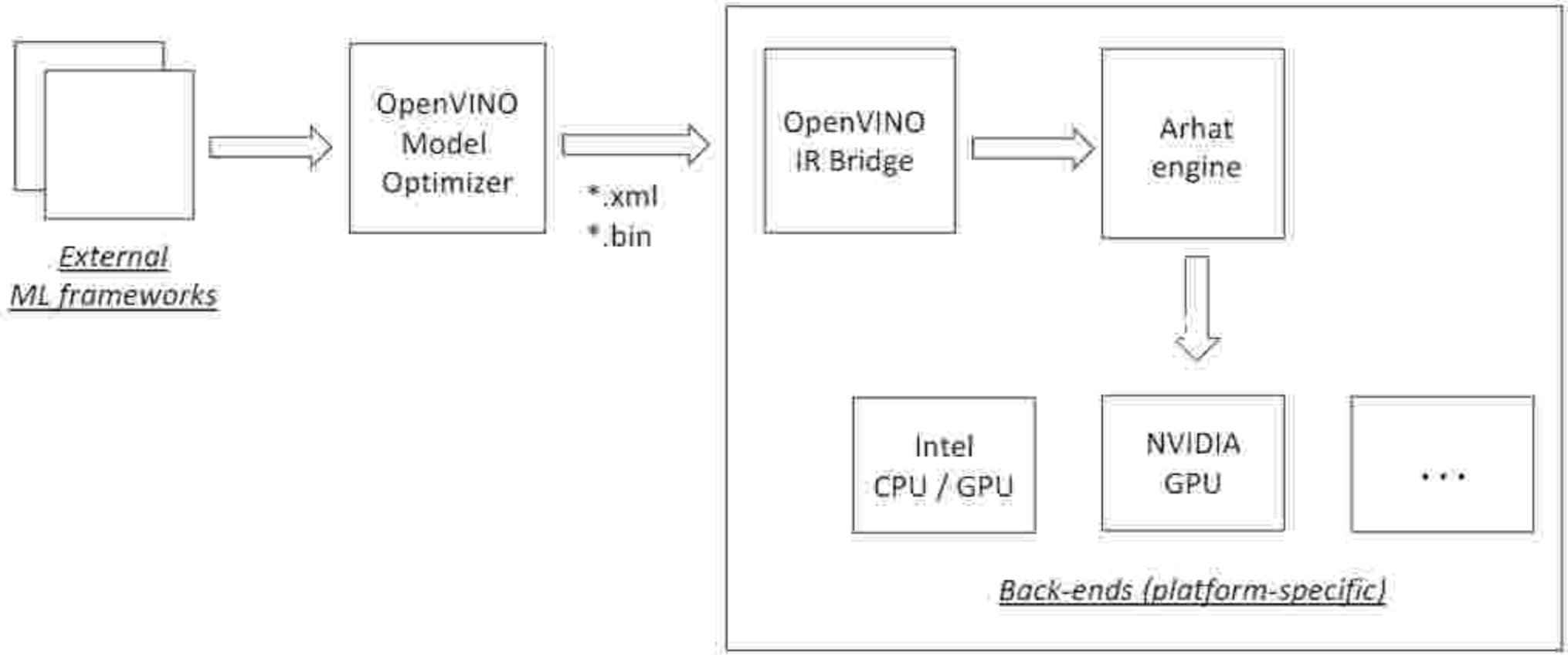
A collection of pre-trained image classification models ported from PyTorch torchvision package to Arhat SDK

Automatically generated lean C++ libraries are available for Intel (oneDNN) , NVIDIA (cuDNN/CUDA), and AMD (MIOpen)

- AlexNet
- VGG
  - vgg11
  - vgg11_bn
  - vgg13
  - vgg13_bn
  - vgg16
  - vgg16_bn
  - vgg19
  - vgg19_bn
- ResNet
  - resnet18
  - resnet34
  - resnet50
  - resnet101
  - resnet152
- SqueezeNet
  - squeezenet_0
  - squeezenet_1
- DenseNet
  - densenet121
  - densenet161
  - densenet169
  - densenet201
- Inception v3
- GoogLeNet
- ShuffleNet v2
  - shufflenet_v2_x0_5
  - shufflenet_v2_x1_0
  - shufflenet_v2_x1_5
  - shufflenet_v2_x2_0
- MobileNet v2
- ResNeXt
  - resnext50_32x4d
  - resnext101_32x8d
- Wide ResNet
  - wide_resnet50_2
  - wide_resnet101_2
- MNASNet
  - mnasnet0_5
  - mnasnet0_75
  - mnasnet1_0
  - mnasnet1_3

https://github.com/fragata-ai/arhat-sdk/tree/master/go/src/fragata/arhat/examples/torch/vision

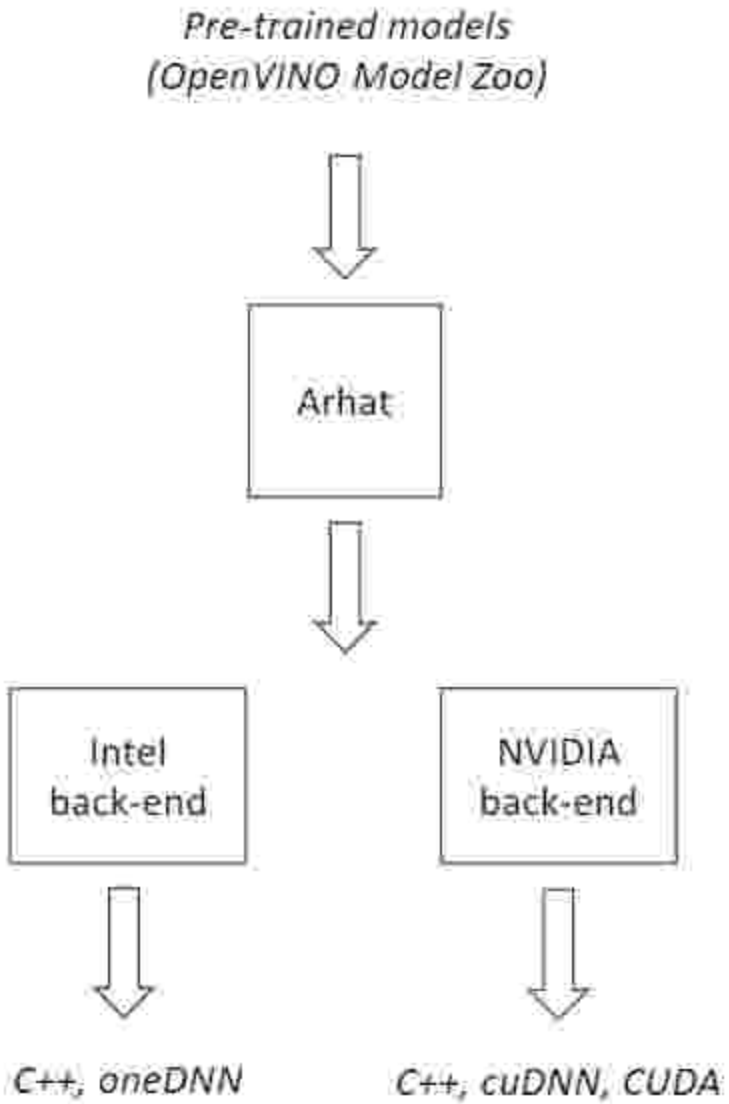
Interoperability with OpenVINO



Highlights:

- Integrates Arhat with Intel deep learning ecosystem
- Provides a light-weight OpenVINO extension for applications requiring leaner code
- Adds native support for various vendor-specific platforms
- Enables using Intel Open Model Zoo with Arhat
- Works with the mainstream (unpatched) version of oneDNN

Case study: object detection



Model selection

SSD

- ssd_inception_v1_fpn_coco (640 x 640)
- ssd_inception_v2_coco (300 x 300)
- ssd_resnet50_v1_fpn_codo (640 x 640)

Faster R-CNN

- faster_rcnn_mobilenet_v2_coco (600 x 1024)
- faster_rcnn_resnet50_coco (600 x 1024)

YOLO

- yolo-v2-tf (608 x 608)
- yolo-v3-tf (416 x 416)
- yolo-v4-tf (608 x 608)

Model repository: https://github.com/openvinotoolkit/open_model_zoo

Performance evaluation: Arhat vs. OpenVINO

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

GPU

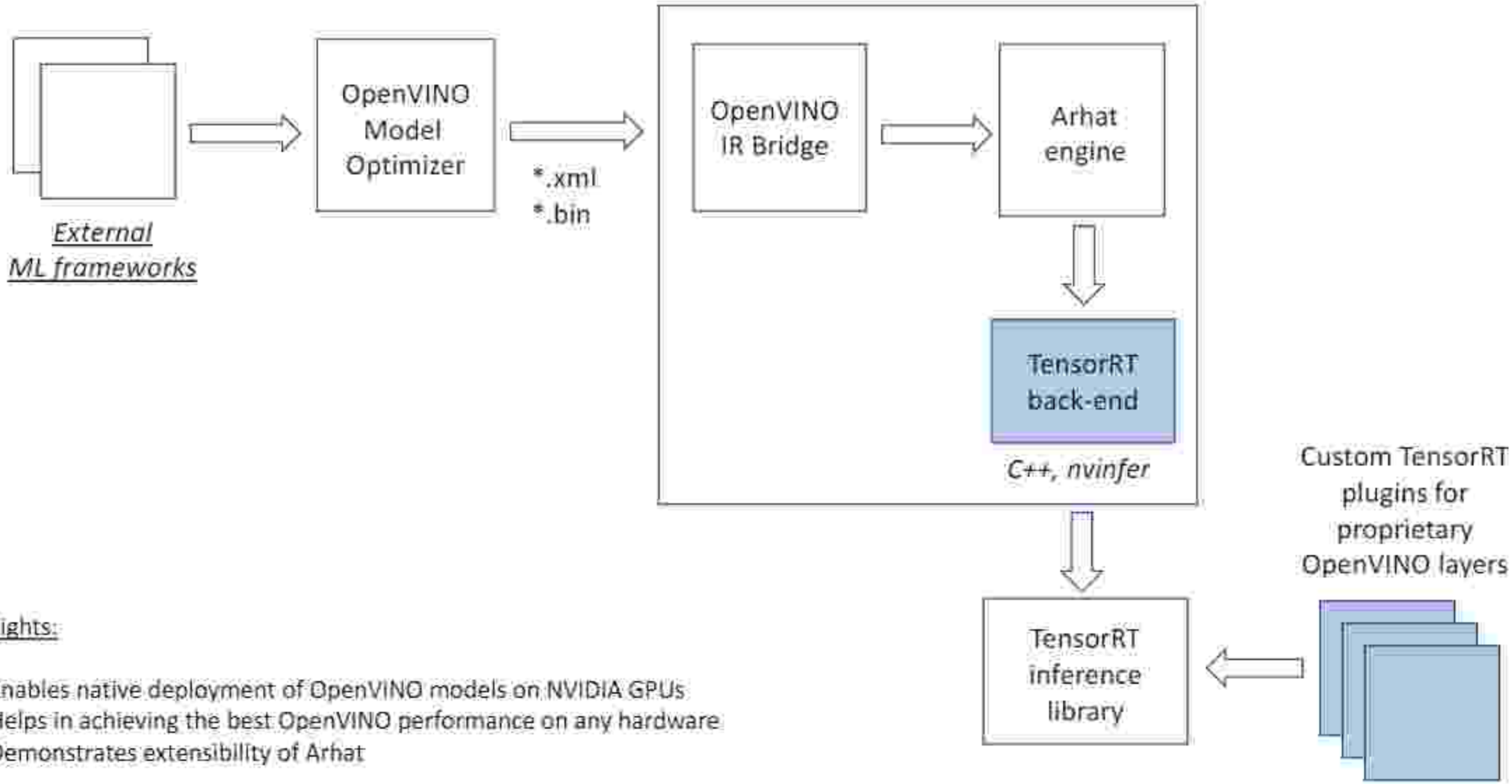
Model	Arhat	OpenVINO
resnet50-v1-7	10.1	11.3
ssd_mobilenet_v1_fpn_coco	144	125
ssd_mobilenet_v2	9.0	10.7
ssd_resnet50_v1_fpn_coco	195	183
faster_rcnn_inception_v2_coco	117	82.8
faster_rcnn_resnet50_coco	252	259
yolo_v2_tf	64.4	60.8
yolo_v3_tf	72.3	68.2
yolo_v4_tf	140	135

Test platform: Intel i7-1185G7E (turbo boost disabled)

Precision: FP32

Processing time for one image in milliseconds
(lower is better)

Cross-platform deployment: OpenVINO meets TensorRT

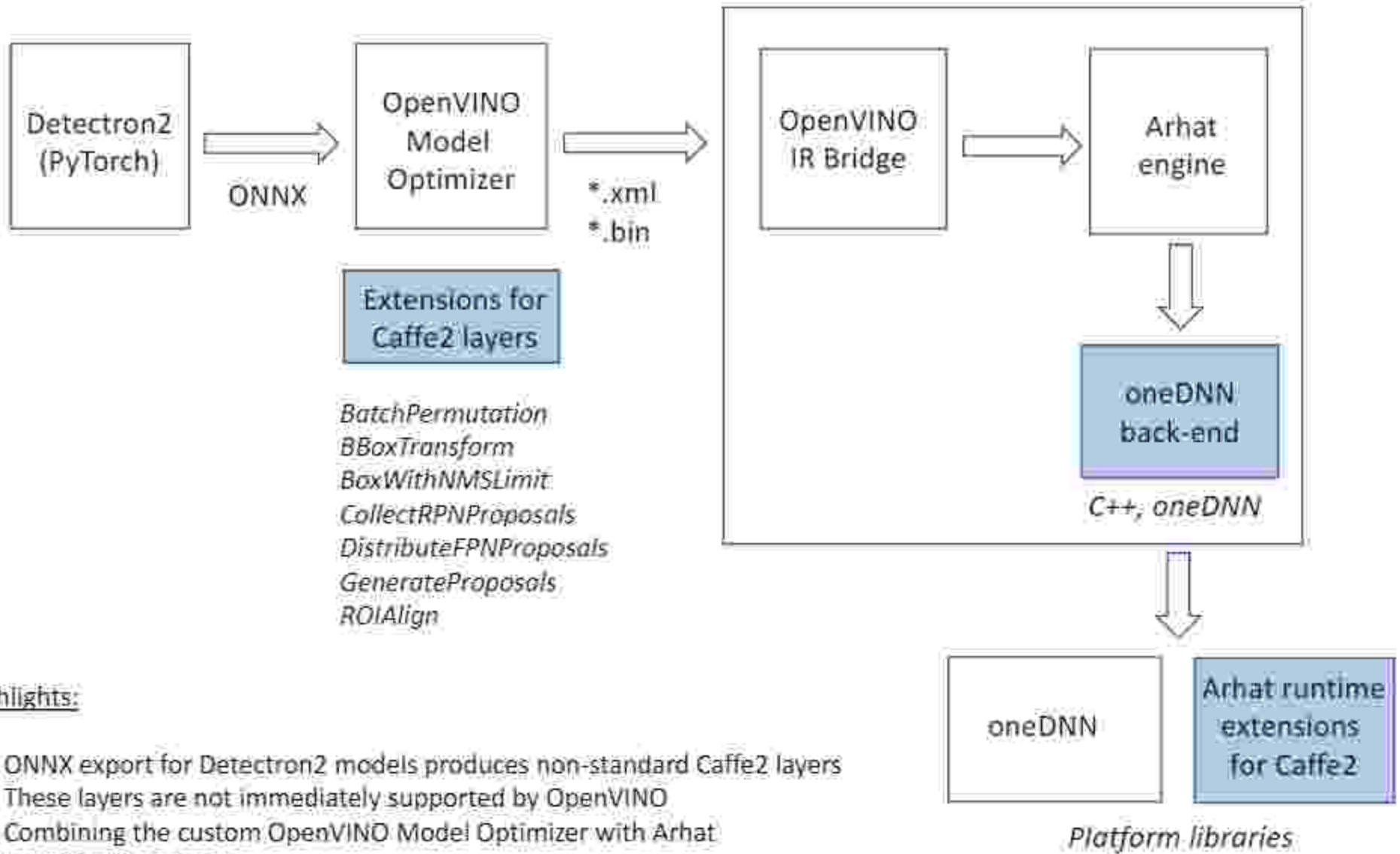


Highlights:

- Enables native deployment of OpenVINO models on NVIDIA GPUs
- Helps in achieving the best OpenVINO performance on any hardware
- Demonstrates extensibility of Arhat

"You can get much farther with vendor tools and Arhat than with vendor tools alone"

Advanced use case: deploying Detectron2 models with Arhat



Highlights:

- ONNX export for Detectron2 models produces non-standard Caffe2 layers
- These layers are not immediately supported by OpenVINO
- Combining the custom OpenVINO Model Optimizer with Arhat provides the solution

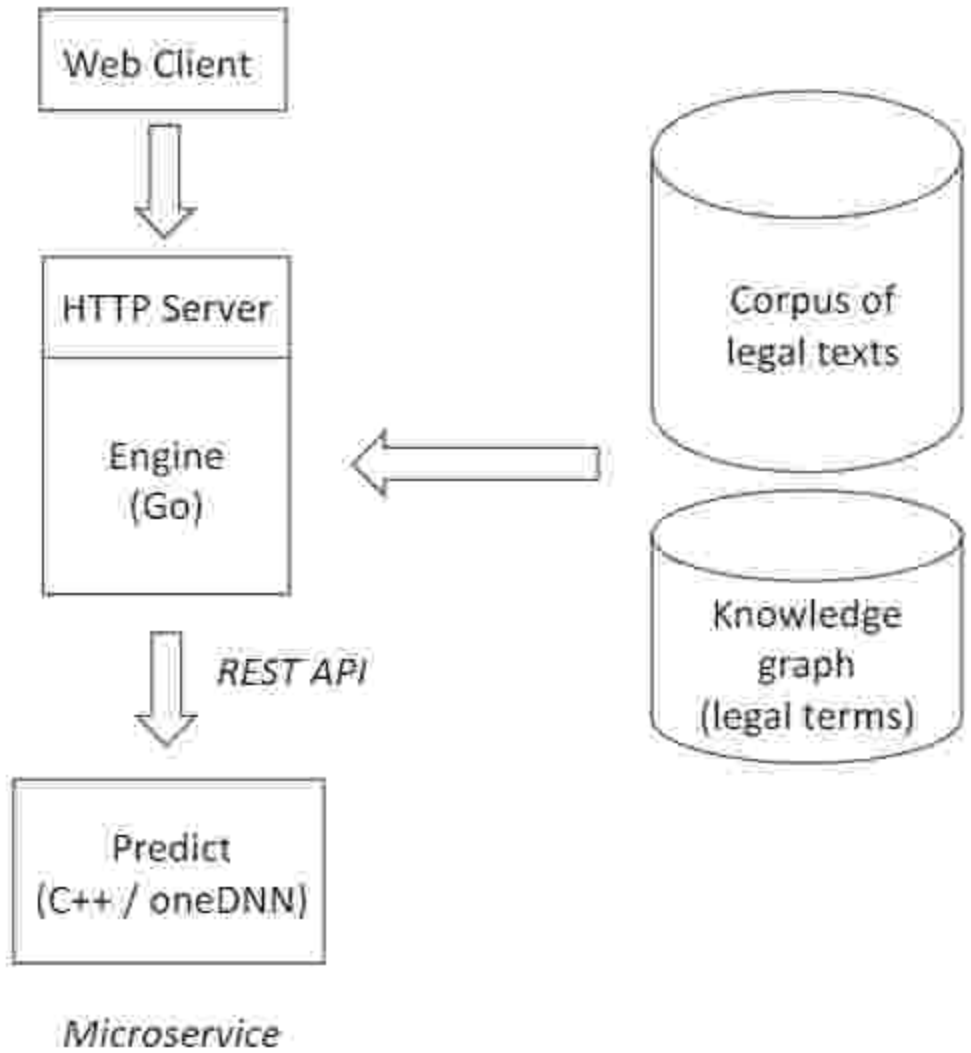
New projects: Analysis of large legal texts with Arhat and HuggingFace transformers

Input:

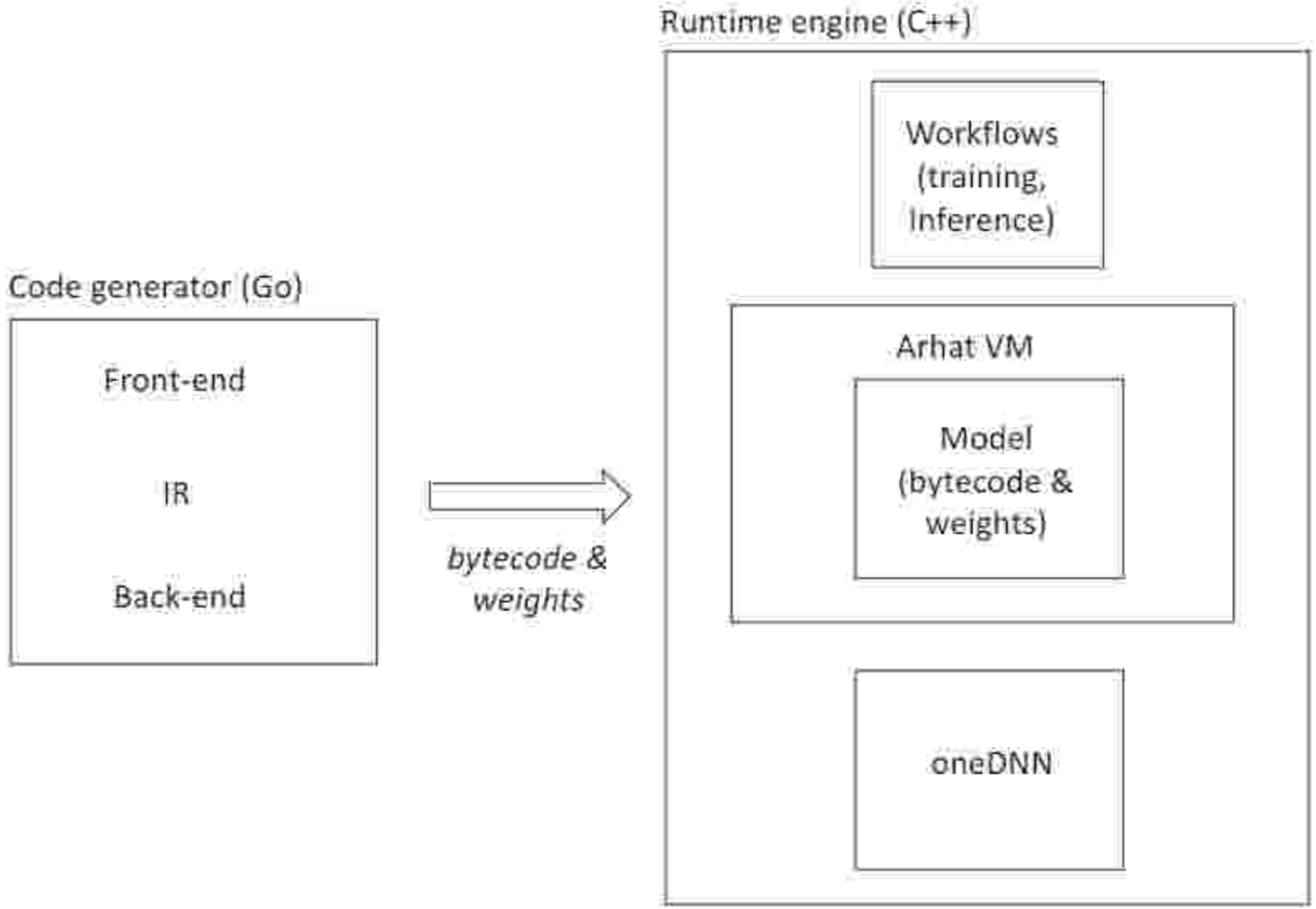
- Question in a natural language

Output:

- Set of text fragments (articles / paragraphs) matching the question



New projects: Reference deep learning framework for oneDNN



Conclusion

- Arhat performs automated generation of software code specific for the selected deep learning model and target platform. This results in a very lean application code with minimum external dependencies. Deployment and maintenance of such applications requires much lesser effort compared to the mainstream approach.

- When used for deployment of inference workflows, Arhat can combine heterogeneous tools of different vendors (like Intel OpenVINO and NVIDIA TensorRT) that are most suitable for handling the given task. This opens a way for achieving the best computational performance on any hardware.

- Arhat can be 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.

Acknowledgements and further references

Many thanks to Richard Gaechter, Abdulmecit Gungor (Intel), Hans Pabst (Intel), and Wei Wang (Intel)
for their valuable insights and support.

Technical materials

Arhat at Intel DevMesh:

https://devmesh.intel.com/projects/arhat

Articles

How to pack the power of AI into a product: Arhat redefines the concept of "lean", by Richard Gaechter:

https://www.linkedin.com/pulse/how-pack-power-ai-product-arhat-redefines-concept-lean-gaechter/

Arhat deep learning framework: a novel approach to meeting challenges of modern AI world:

https://blog.snglr.group/arhat-deep-learning-framework-a-novel-approach-to-meeting-challenges-of-modern-ai-world/

Integrating Arhat deep learning framework with Intel oneDNN library and OpenVINO toolkit:

https://devmesh.intel.com/post/968366/integrating-arhat-deep-learning-framework-with-intel-onednn-library-and-openvino-toolkit