Using the Intel® Distribution of the OpenVINO™ Toolkit for Deploying Accelerated Deep Learning Applications – Part1 [2021.3]

April 2021

intel®

Agenda

Part 1: OpenVINO Workshop (90mins):

- Overview of OpenVINO Toolkit
- Model Optimizer
- Inference Engine
- VPU Accelerators
- Multiple models in one application
- Deployment Manager
- Conditional Compilation [NEW]
- DevCloud Overview

• Part2: Hands-On Training (30mins):

- DevCloud Registration
- Sample Tutorials

Notices and Disclaimers

- Performance varies by use, configuration and other factors. Learn more at www.Intel.com/PerformanceIndex.
- Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See backup for configuration details. No product or component can be absolutely secure.
- Your costs and results may vary.
- Intel technologies may require enabled hardware, software or service activation.
- All product plans and roadmaps are subject to change without notice.
- Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.
- © Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.

Introduction to Intel® Distribution of OpenVINO™ Toolkit

April 2021

intel®

Intel® Distribution of OpenVINO™ Toolkit

- Tool Suite for High-Performance, Deep Learning Inference
- Fast, accurate real-world results using high-performance, AI and computer vision inference deployed into production across Intel® architecture from edge to cloud

High-Performance,
Deep Learning Inference

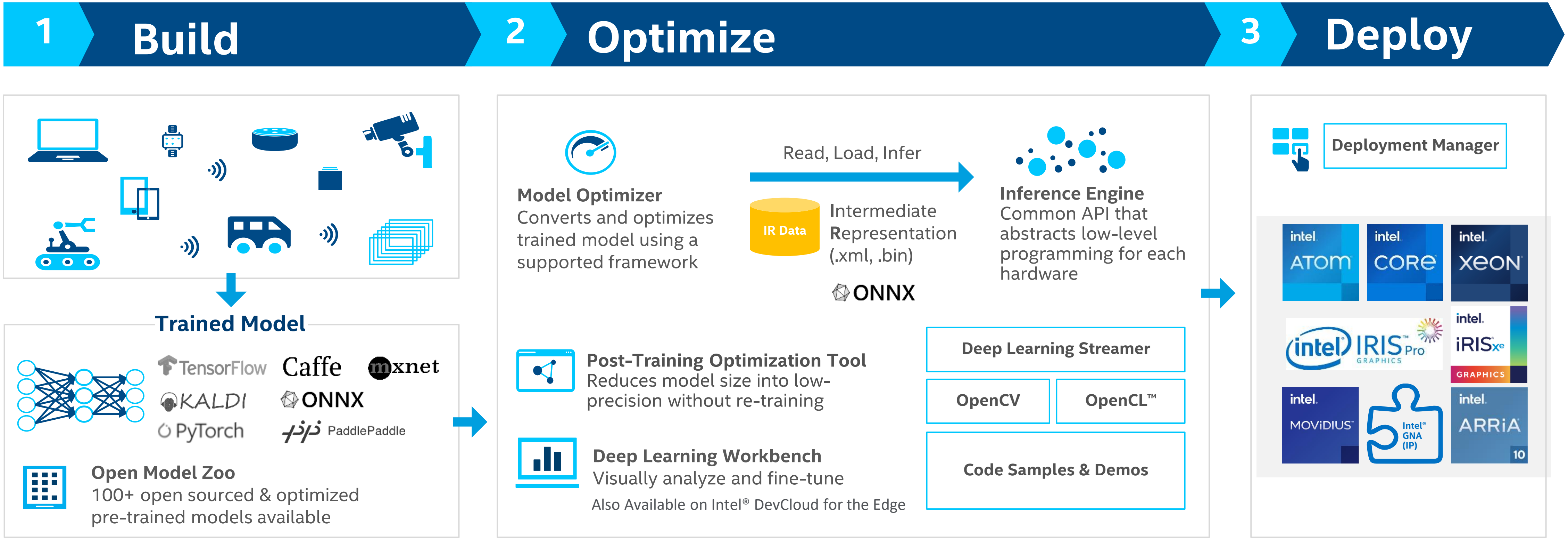
Streamlined Development,
Ease of Use

Write Once,
Deploy Anywhere

- Enables deep learning inference from the edge to cloud.
- Supports heterogeneous execution across Intel accelerators, using a common API for the Intel® CPU, Intel® Integrated Graphics, Intel® Gaussian & Neural Accelerator, Intel® Neural Compute Stick 2, Intel® Vision Accelerator Design with Intel® Movidius™ VPUs.
- Speeds time-to-market through an easy-to-use library of CV functions and pre-optimized kernels.
- Includes optimized calls for CV standards, including OpenCV* and OpenCL™.

For workloads and configurations visit www.Intel.com/PerformanceIndex. Results may vary.

Three steps for developing with the Intel® Distribution of OpenVINO™ toolkit



1 Build

Trained Model

TensorFlow, Caffe, mxnet, KALDI, ONNX, PyTorch, PaddlePaddle

Open Model Zoo
100+ open sourced & optimized pre-trained models available

2 Optimize

Model Optimizer
Converts and optimizes trained model using a supported framework

Read, Load, Infer

IR Data

Intermediate Representation (.xml, .bin)

ONNX

Inference Engine
Common API that abstracts low-level programming for each hardware

Post-Training Optimization Tool
Reduces model size into low-precision without re-training

Deep Learning Workbench
Visually analyze and fine-tune

Also Available on Intel® DevCloud for the Edge

Deep Learning Streamer

OpenCV

OpenCL™

Code Samples & Demos

3 Deploy

Deployment Manager

For workloads and configurations visit www.Intel.com/PerformanceIndex. Results may vary.

OpenVINO™ Add-ons

OpenVINO™ Model Server (OVMS)

OpenVINO™ Model Server (OVMS) is a scalable, high-performance solution for serving machine learning models optimized for Intel® architectures. The server provides an inference service via gRPC or REST API - making it easy to deploy new algorithms and AI experiments using the same architecture as TensorFlow* Serving for any models trained in a framework that is supported by OpenVINO.

OpenVINO™ Security Add-on (OVSA)

The OpenVINO™ Security Add-on works with the OpenVINO™ Model Server on Intel® architecture. Together, the OpenVINO™ Security Add-on and the OpenVINO™ Model Server provide a way for Model Developers and Independent Software Vendors to use secure packaging and secure model execution to enable access control to the OpenVINO™ models, and for model Users to run inference within assigned limits.

OpenVINO™ Add-ons

Neural Network Compression Framework (NNCF)

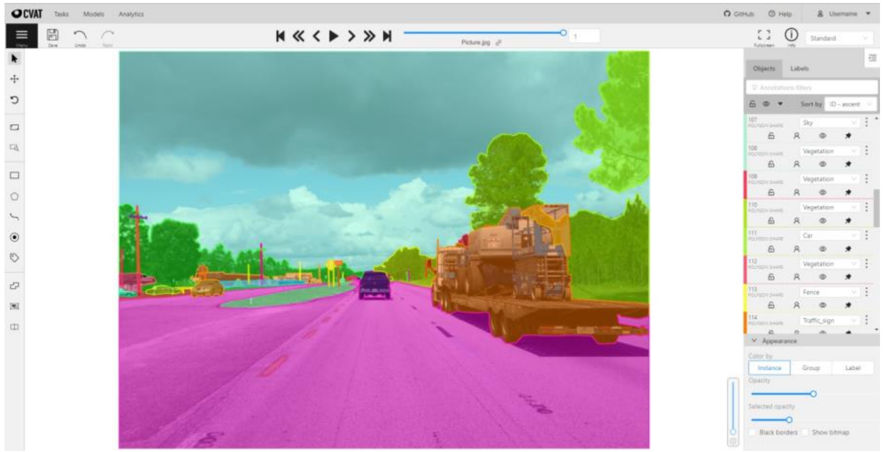
Contains a PyTorch*-based framework and samples for neural networks compression.

The framework is organized as a Python* package that can be built and used in a standalone mode. The framework architecture is unified to make it easy to add different compression methods.

The samples demonstrate the usage of compression algorithms for three different use cases on public models and datasets.

Computer Vision Annotation Tool (CVAT)

CVAT is free, online, interactive video and image annotation tool for computer vision. It is being used by our team to annotate million of objects with different properties. Many UI and UX decisions are based on feedbacks from professional data annotation team.



Choose between Release Types

Standard Releases vs Long-Term Support Releases

Standard Release (3-4 releases a year): Users looking to take advantage of new features, tools and support in order to keep current with the advancements in deep learning technologies

Long-Term Support Release: Users looking for a stable and reliable version that is maintained for a longer period, and are looking for little to no new feature changes

Supported OSes and installation options

April 2021

intel®

Supported OS - Development Platform

https://software.intel.com/content/www/us/en/develop/tools/openvino-toolkit/system-requirements.html

Processors

- 6th to 11th generation Intel® Core™ and Intel® Xeon® processors
- Pentium® processor N4200/5, N3350/5, N3450/5 with Intel® HD Graphics
- Intel Atom® processor with SSE4.1 support

Development Platform

- Ubuntu* 20.04 LTS (64 bit)
- Ubuntu 18.04 LTS (64 bit)
- Windows® 10 (64 bit)
- CentOS* 7 (64 bit)
- macOS* 10.15 (64 bit)

Supported OS – Target System Platform

https://software.intel.com/content/www/us/en/develop/tools/openvino-toolkit/system-requirements.html

CPU

- Processors
  - 6th to 11th generation Intel Core processors
  - Intel® Xeon® Scalable processors (formerly code-named Skylake)
  - 2nd generation Intel Xeon Scalable processors (formerly code-named Cascade Lake)
  - 3rd generation Intel Xeon Scalable processors (formerly code-named Cooper Lake and Ice Lake) [New]
  - Pentium® processor N4200/5, N3350/5, N3450/5 with Intel HD Graphics
  - Intel Atom processor with SSE4.1 support

- Compatible Operating Systems
  - Ubuntu 18.04 LTS (64 bit)
  - Ubuntu 20.04 LTS (64 bit)
  - Windows 10 (64 bit)
  - CentOS 7 (64 bit)
  - Red Hat* Enterprise Linux* 8 (64 bit)
  - macOS 10.15 (64 bit)
  - Yocto Project* Poky Zeus v3.0.x (64 bit)

Supported OS – Target System Platform

https://software.intel.com/content/www/us/en/develop/tools/openvino-toolkit/system-requirements.html

GPU

- Processors
  - 6th to 9th generation Intel® Iris® Plus graphics, Intel UHD Graphics, Intel HD Graphics†, and Xe architecture
  - Intel® Iris® X^e MAX graphics
- Compatible Operating Systems
  - Ubuntu 18.04 LTS (64 bit)
  - Ubuntu 20.04 LTS (64 bit)
  - Windows 10 (64 bit)
  - CentOS 7 (64 bit)

VPU

- Processor
  - Intel® Movidius™ Myriad™ X VPU
- Supported Hardware
  - Intel® Neural Compute Stick 2
  - Intel Vision Accelerator Design with Intel® Movidius™ Vision Processing Unit (VPU)
- Compatible Operating Systems
  - Ubuntu 18.04 LTS (64 bit)
  - Windows 10 (64 bit)

FPGA (LTS only)

- Supported Hardware
  - Intel® Vision Accelerator Design with Intel® Arria 10
- Compatible Operating System
  - Ubuntu 18.04 LTS (64 bit)

Common Install options across Linux, Windows and macOS

- Download the online or local installation package
  - https://software.intel.com/content/www/us/en/develop/tools/openvino-toolkit.html
- Build OpenVINO toolkit from source on GitHub/Gitee
  - https://github.com/openvinotoolkit/openvino.git
  - https://gitee.com/openvinotoolkit-prc/openvino.git
- Python Package Installer
  - https://pypi.org/project/openvino/
  - https://pypi.org/project/openvino-dev/ [NEW]
- Intel Edge Software Hub
  - Edge Insights for Vision
- Customize a Dockerfile
  - https://github.com/openvinotoolkit/docker_ci
- Docker Hub
  - docker pull openvino/ubuntu20_runtime
  - docker pull openvino/ubuntu20_runtime
  - docker pull openvino/ubuntu18_dev
- Intel® DevCloud for the Edge
  - https://devcloud.intel.com/edge

Special install options for different Linux OSes

- APT Repository Package Manager
  - Runtime Packages
    - sudo apt-cache search intel-openvino-runtime-ubuntu18
    - sudo apt-cache search intel-openvino-runtime-ubuntu20
  - Developer Packages
    - sudo apt-cache search intel-openvino-dev-ubuntu18
    - sudo apt-cache search intel-openvino-dev-ubuntu20

- YUM Repository Package Manager
  - To install the latest version
    - sudo yum install intel-openvino-runtime-centos7
  - To install a specific version
    - sudo yum install intel-openvino-runtime-centos7-<VERSION>.<UPDATE>.<BUILD_NUM
- Red Hat* Quay [NEW]
  - docker run -it --rm quay.io/openvino/rhel8_runtime
- Raspbian OS
  - https://storage.openvinotoolkit.org/

OpenVINO™ Extra Modules

https://github.com/openvinotoolkit/openvino_contrib

- No stable API
- Not well-tested
- Not part of official OpenVINO distribution
- Library maintains backward compatibility for better performance
- Developed separately and published in the **openvino_contrib** repository at first
- Will be moved to the central OpenVINO repository when mature and popular

- arm_plugin: ARM CPU Plugin -- allows to perform deep neural networks inference on ARM CPUs, using OpenVINO API.
- java_api: Inference Engine Java API -- provides Java wrappers for Inference Engine public API.
- mo_pytorch: PyTorch extensions for Model Optimizer -- native PyTorch to OpenVINO IR converter

Open Model Zoo

April 2021

Open Model Zoo

https://github.com/openvinotoolkit/open_model_zoo

- Pre-trained models
  - Intel pre-trained models
  - Public pre-trained models
- Demo Applications
  - Console applications written in C, C++, Python:
- Tools
  - Model Downloader
  - Accuracy Checker

Pre-trained Models

Open-sourced repository of pre-trained models and support for public models

Intel Pre-trained Models

- Action Recognition Models
- Classification Models
- Head Pose Estimation Models
- Human Pose Estimation Models
- Image Processing Models
- Instance Segmentation Models
- Machine Translation Models
- Object Attribute Estimation Models
- Object Detection Models
- Optical Character Recognition Models
- Question Answering Models
- Semantic Segmentation Models
- Text-to-speech Models
- Token Recognition Models

Public Pre-trained Models

- Action Recognition Models
- Classification Models
- Colorization Models
- Face Recognition Models
- Human Pose Estimation Models
- Image Inpainting Models
- Image Processing Models
- Image Translation Models
- Instance Segmentation Models
- Monocular Depth Estimation Models
- Object Attribute Estimation Models
- Object Detection Models
- Optical Character Recognition Models
- Place Recognition Models
- Semantic Segmentation Models
- Sound Classification Models
- Speech Recognition Models
- Style Transfer Models
- Text-to-speech Models

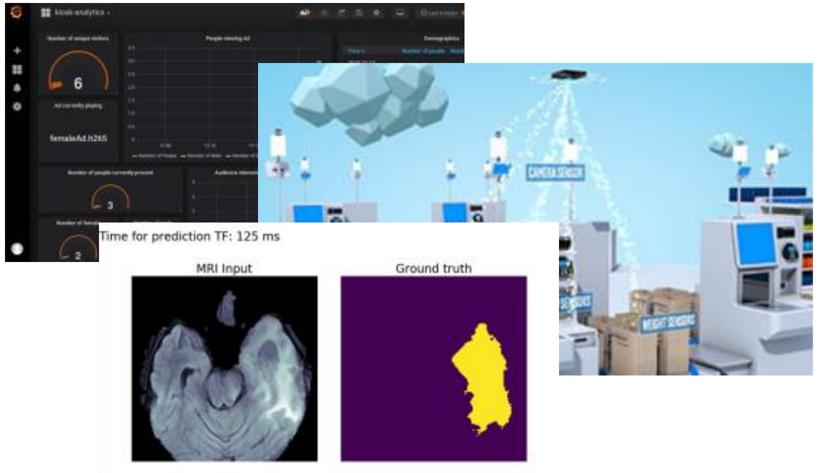
PRE-TRAINED MODELS

https://github.com/openvinotoolkit/open_model_zoo/tree/master/models

Demos Applications

Quickly get started with example demo applications

Take advantage of **pre-built, open-sourced** example implementations with step-by-step guidance and required components list



- 3D Human Pose Estimation Python* Demo
- 3D Segmentation Python* Demo
- Action Recognition Python* Demo
- BERT Question Answering Embedding Python* Demo
- BERT Question Answering Python* Demo
- Classification C++ Demo
- Colorization Demo
- Crossroad Camera C++ Demo
- Face Detection MTCNN Python* Demo
- Formula Recognition Python* Demo
- G-API Interactive Face Detection Demo
- Gaze Estimation Demo
- Gesture Recognition Python* Demo
- Handwritten Text Recognition Demo
- Human Pose Estimation C++ Demo
- Human Pose Estimation Python* Demo
- Image Deblurring Python* Demo
- Image Inpainting Python Demo
- Image Retrieval Python* Demo
- Image Segmentation C++ Demo
- Image Segmentation Python* Demo
- Image Translation Demo
- Instance Segmentation Python* Demo
- Interactive Face Detection C++ Demo
- Machine Translation Python* Demo
- MonoDepth Python Demo
- Multi Camera Multi Target Python* Demo
- Multi-Channel Face Detection C++ Demo
- Multi-Channel Human Pose Estimation C++ Demo
- Multi-Channel Object Detection Yolov3 C++ Demo
- Object Detection C++ Demo
- Object Detection Python* Demo
- Pedestrian Tracker C++ Demo
- Place Recognition Python* Demo
- Security Barrier Camera C++ Demo
- Single Human Pose Estimation Demo (top-down pipeline)
- Smart Classroom C++ Demo
- Sound Classification Python* Demo
- Speech Recognition Demo
- Super Resolution C++ Demo
- TensorFlow* Object Detection Mask R-CNNs Segmentation C++ Demo
- Text Detection C++ Demo
- Text Spotting Python* Demo
- Text-to-speech Python* Demo
- Whiteboard Inpainting Demo

DEMO APPLICATIONS

https://github.com/openvinotoolkit/open_model_zoo/tree/master/demos

Tools

Model Downloader

- Provides an easy way of accessing a number of public models as well as a set of pre-trained Intel models

Accuracy Checker

- Check for accuracy of the model (original and after conversion) to IR file using a known data set

- **downloader.py** (model downloader) downloads model files from online sources and, if necessary, patches them to make them more usable with Model Optimizer;
- **converter.py** (model converter) converts the models that are not in the Inference Engine IR format into that format using Model Optimizer.
- **quantizer.py** (model quantizer) quantizes full-precision models in the IR format into low-precision versions using Post-Training Optimization Toolkit.
- **info_dumper.py** (model information dumper) prints information about the models in a stable machine-readable format.

TOOLS

https://github.com/openvinotoolkit/open_model_zoo/tree/master/tools

Model Optimizer

April 2021

Intel® Deep Learning Deployment Toolkit

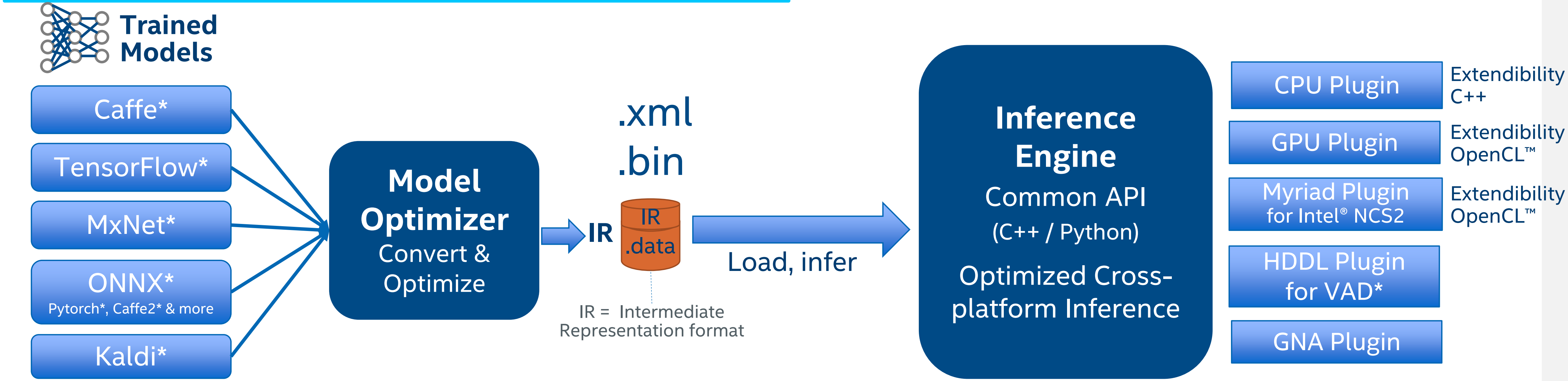
For Deep Learning Inference

Model Optimizer

- A Python* based tool to **import** trained models and **convert** them to Intermediate Representation
- **Optimizes for performance** or space with conservative topology transformations
- **Hardware-agnostic** optimizations

Inference Engine

- High-level, C/C++ and Python, inference **runtime API**
- Interface is implemented as **dynamically loaded plugins** for each hardware type
- Delivers advanced performance for each type **without requiring users to implement and maintain multiple code pathways**

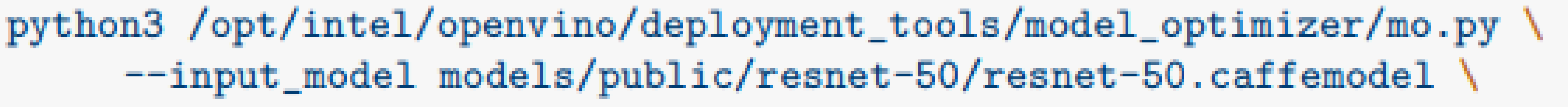


GPU = Intel® CPU with integrated GPU/Intel® Processor Graphics, Intel® NCS = Intel® Neural Compute Stick (VPU)
*VAD = Intel® Vision Accelerator Design Products (HDDL-R)

Model Optimizer: Generic Optimization

- **Model optimizer performs generic optimization**
  - Drop unused layers (dropout)
  - Node merging

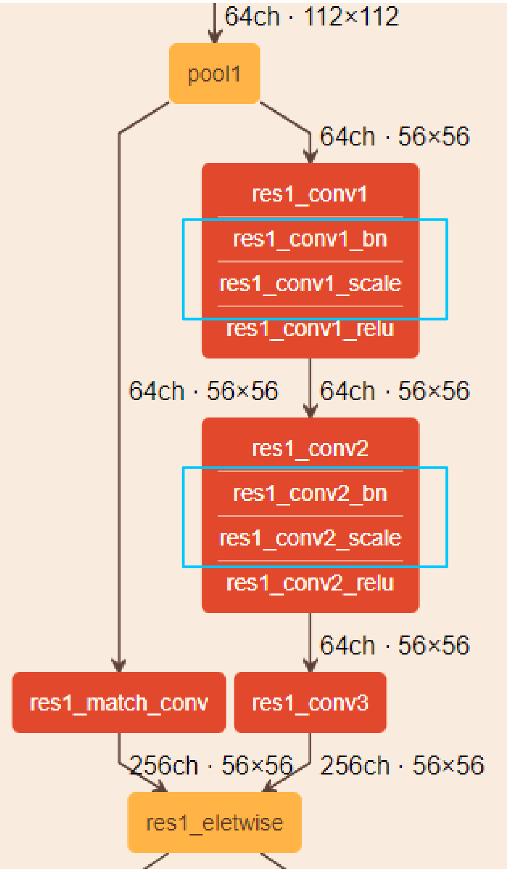
- **The simplest way to convert a model is to run mo.py with a path to the input model file**
  - By default, generic optimization will be automatically applied, unless manually set disable



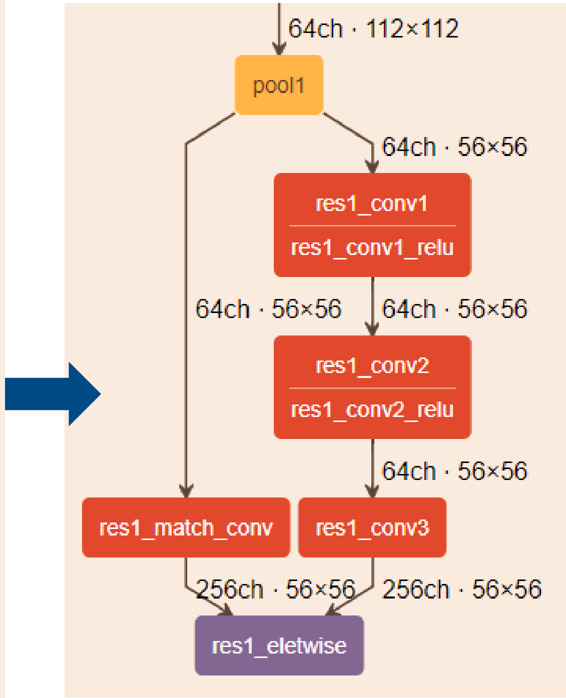
```
python3 /opt/intel/openvino/deployment_tools/model_optimizer/mo.py \
     --input_model models/public/resnet-50/resnet-50.caffemodel \
```

Model Optimizer: Linear Operation Fusing

1. **BatchNorm and ScaleShift decomposition:** *BN* layers decomposes to *Mul->Add->Mul->Add* sequence; ScaleShift layers decomposes to *Mul->Add* sequence.
2. **Linear operations merge:** Merges sequences of Mul and Add operations to the **single** Mul->Add instance.
3. **Linear operations fusion:** Fuses Mul and Add operations to Convolution or FullyConnected layers.



Caffe* Resnet269 block (from Netscope)



Merged Caffe* Resnet269 block (from Netscope*)

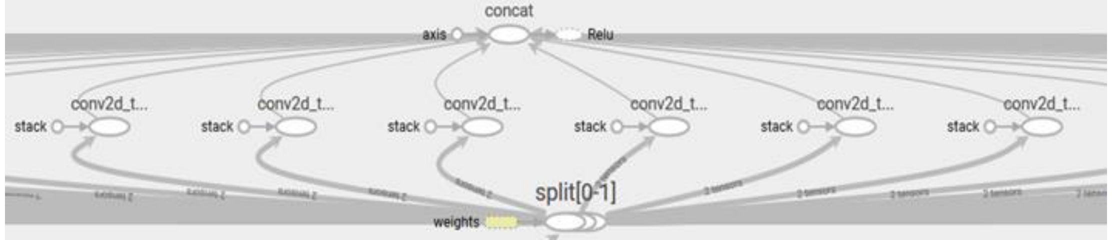
Model Optimizer: Framework or topology specific optimization

Grouped Convolutions Fusing

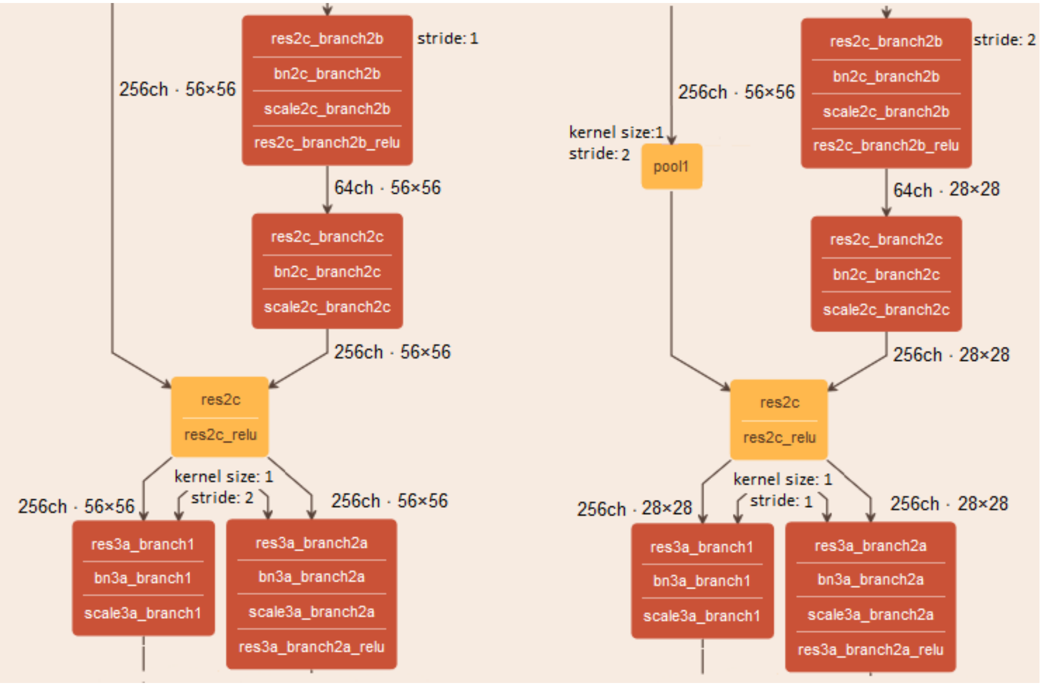
- Grouped convolution fusing is a specific optimization that applies for TensorFlow* topologies. The main idea of this optimization is to combine convolutions results for the Split outputs and then recombine them using **Concat** operation in the same order as they were out from **Split**.

ResNet* optimization (stride optimization)

- This optimization is to move the stride that is greater than 1 from Convolution layers with the kernel size = 1 to upper Convolution layers. In addition, the Model Optimizer adds a Pooling layer to align the input shape for a Eltwise layer, if it was changed during the optimization.



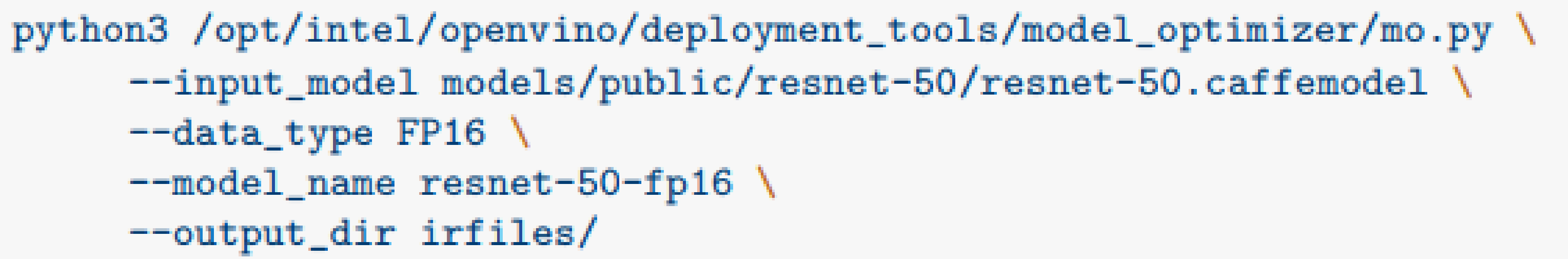
Split->Convolutions->Concat block from TensorBoard*



Model Optimizer: Quantization

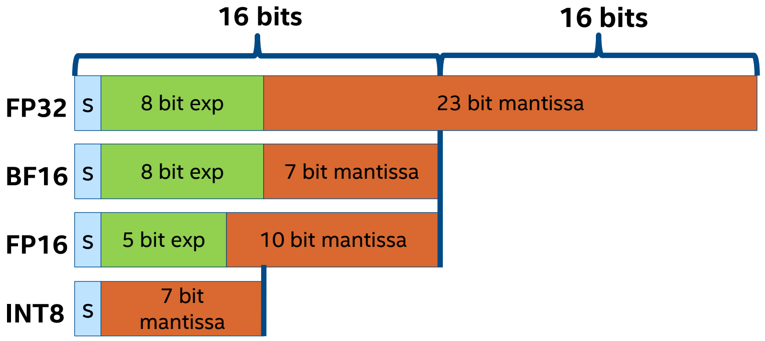
--data_type {FP16,FP32,half,float}

- Data type for all intermediate tensors and weights.
- If original model is in FP32 and --data_type=FP16 is specified, all model weights and biases are quantized to FP16.



```
python3 /opt/intel/openvino/deployment_tools/model_optimizer/mo.py \
     --input_model models/public/resnet-50/resnet-50.caffemodel \
     --data_type FP16 \
     --model_name resnet-50-fp16 \
     --output_dir irfiles/
```

PLUGIN	FP32	FP16	INT8
CPU plugin	Supported and preferred	Supported	Supported
GPU plugin	Supported	Supported and preferred	Supported*
VPU plugins	Not supported	Supported	Not supported
GNA plugin	Supported	Supported	Not supported
FPGA plugin	Supported	Supported	Not supported



Note:
1. To create INT8 models, you will need DL Workbench or Post Training Optimization Tool
2. FPGA also support FP11, convert happens on FPGA

Model Optimizer: Other Common Parameters

- --scale, --scale_values, --mean_values, --mean_file
  - Usually, neural network models are trained with the normalized input data. This means that the input data values are converted to be in a specific range, for example, [0, 1] or [-1, 1]. Sometimes the mean values (mean images) are subtracted from the input data values as part of the pre-processing
- --input_shape
  - when the input data shape for the model is not fixed, like for the fully-convolutional neural networks. In this case, for example, TensorFlow* models contain -1 values in the shape attribute of the Placeholder operation. Inference Engine does not support input layers with undefined size, so if the input shapes are not defined in the model, the Model Optimizer fails to convert the model.
- --reverse_input_channels
  - Inference Engine samples load input images in the BGR channels order. However, the model may be trained on images loaded with the opposite order

Inference Engine

April 2021

Intel® Deep Learning Deployment Toolkit

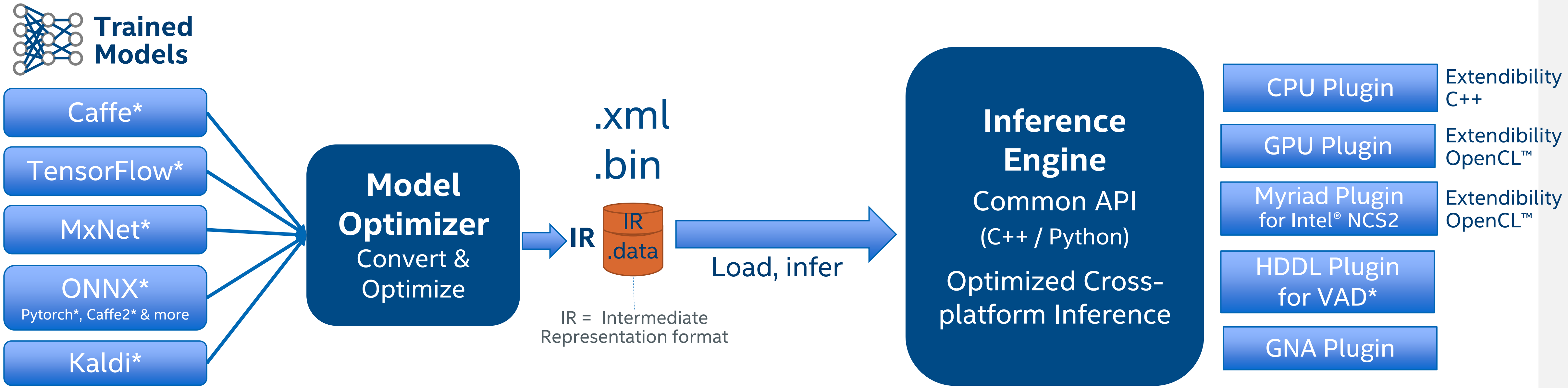
For Deep Learning Inference

Model Optimizer

- A Python* based tool to **import** trained models and **convert** them to Intermediate Representation
- **Optimizes for performance** or space with conservative topology transformations
- **Hardware-agnostic** optimizations

Inference Engine

- High-level, C/C++ and Python, inference **runtime API**
- Interface is implemented as **dynamically loaded plugins** for each hardware type
- Delivers advanced performance for each type **without requiring users to implement and maintain multiple code pathways**



GPU = Intel® CPU with integrated GPU/Intel® Processor Graphics, Intel® NCS = Intel® Neural Compute Stick (VPU)
*VAD = Intel® Vision Accelerator Design Products (HDDL-R)

OpenCL and the OpenCL logo are trademarks of Apple Inc. used by permission by Khronos

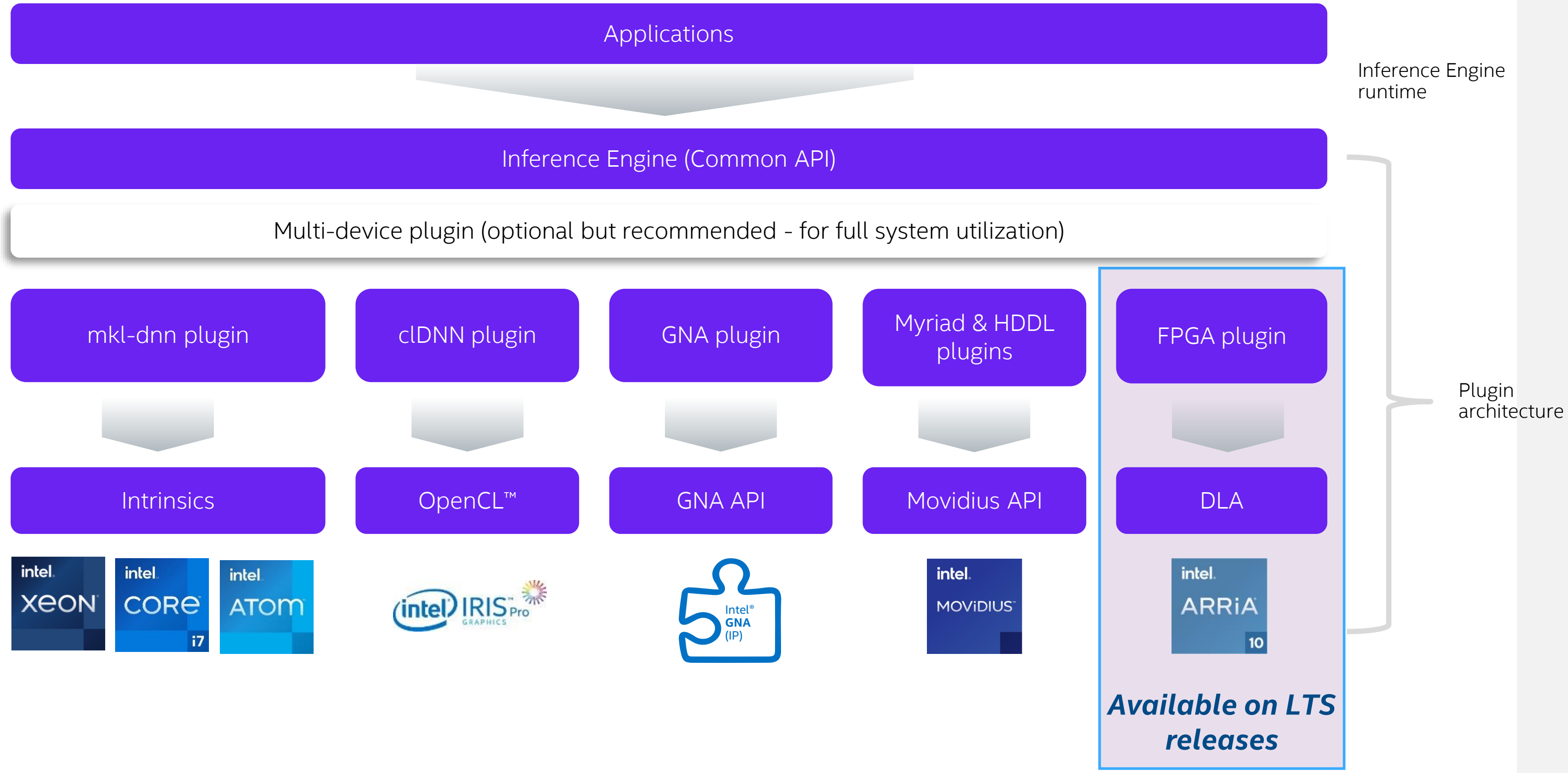
Optimal Model Performance Using the Inference Engine

Core Inference Engine Libraries

- Create Inference Engine Core object to work with devices
- Read the network
- Manipulate network information
- Execute and pass inputs and outputs

Device-specific Plugin Libraries

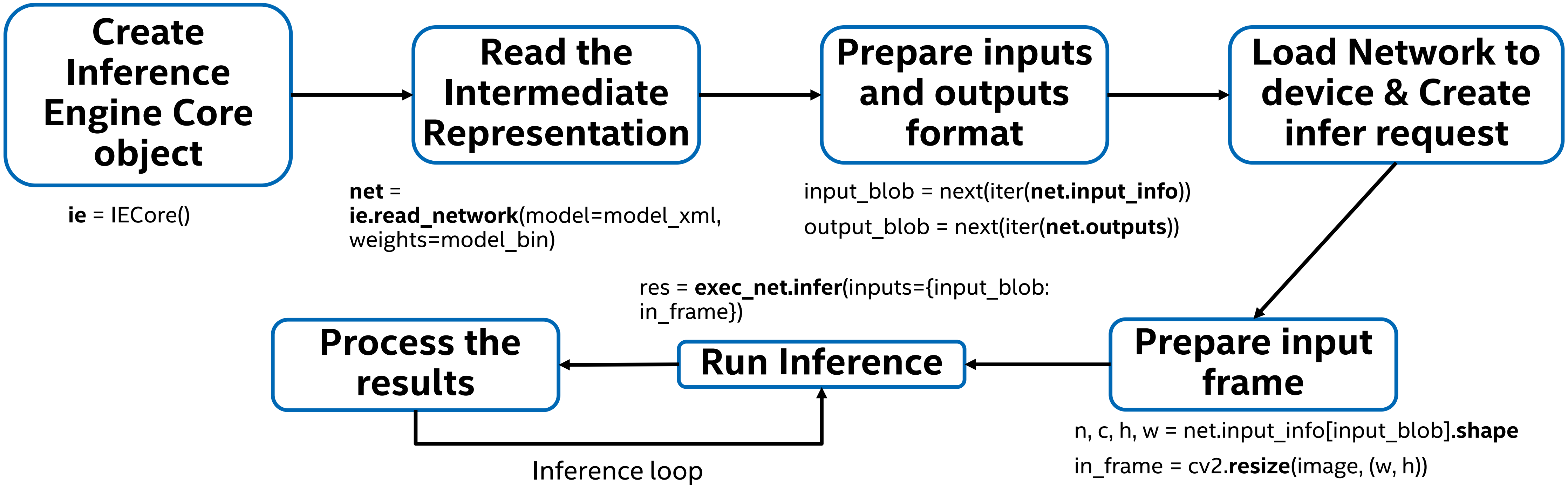
- For each supported target device, Inference Engine provides a plugin — a DLL/shared library that contains complete implementation for inference on this device.



GPU = Intel CPU with integrated graphics/Intel® Processor Graphics/GEN

GNA = Gaussian mixture model and Neural Network Accelerator

Common Workflow for Using the Inference Engine API



Create Inference Engine Core object

```
ie = IECore()
```

Read the Intermediate Representation

```
net = ie.read_network(model=model_xml, weights=model_bin)
```

Prepare inputs and outputs format

```
input_blob = next(iter(net.input_info))
output_blob = next(iter(net.outputs))
```

Load Network to device & Create infer request

```
exec_net = ie.load_network(network=net, device_name=device, num_requests=request_number)
```

Prepare input frame

```
n, c, h, w = net.input_info[input_blob].shape
in_frame = cv2.resize(image, (w, h))
in_frame = in_frame.transpose((2, 0, 1))
in_frame = in_frame.reshape((n, c, h, w))
```

Run Inference

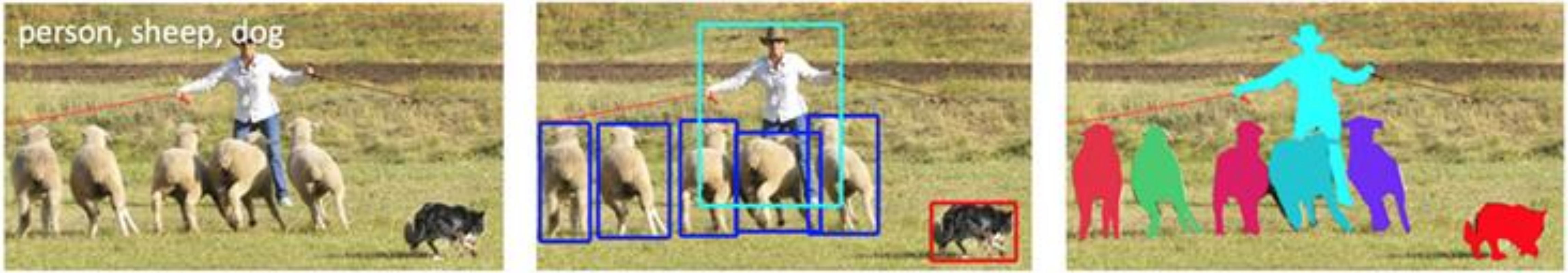
```
res = exec_net.infer(inputs={input_blob: in_frame})
```

Process the results

Inference loop

http://docs.openvinotoolkit.org/latest/_docs_IE_DG_Integrate_with_customer_application_new_API.html

Three Typical Types of Models for Computer Vision Use Cases



(a) classification (b) detection (c) segmentation

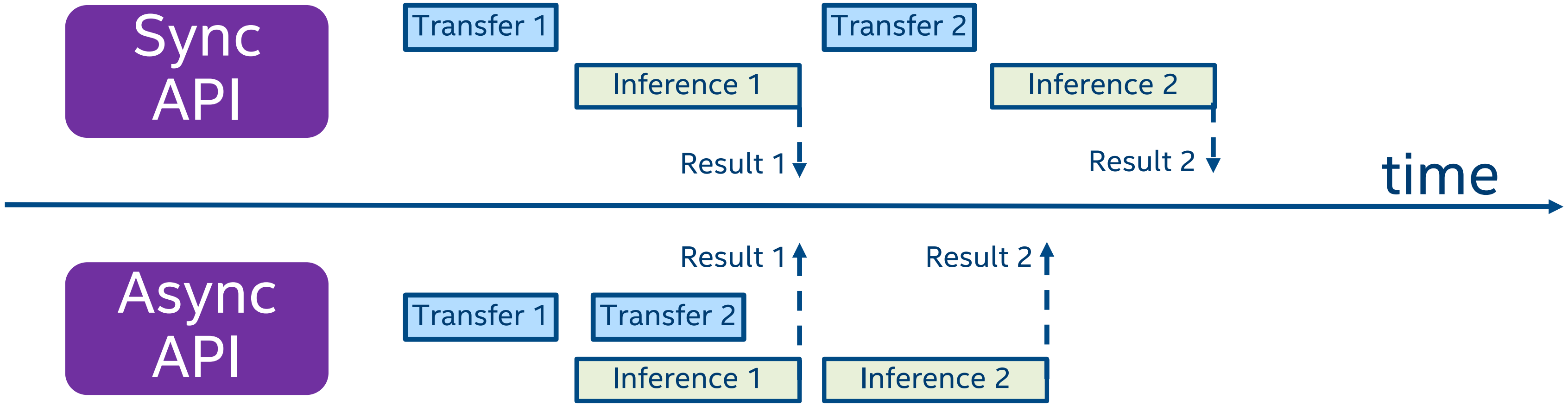
- The complexity of the problem (data set) dictates the network structure. The more complex the problem, the more ‘features’ required, the deeper the network.

Inference Engine

Synchronous vs Asynchronous Execution

- In IE API model can be executed by **Infer Request** which can be:
- **Synchronous** - blocks until inference is completed.
  - exec_net.infer(inputs = {input_blob: in_frame})
- **Asynchronous** – checks the execution status with the wait or specify a completion callback *(recommended way)*.
  - exec_net.start_async(request_id = id, inputs={input_blob: in_frame})
  - If exec_net.requests[id].wait() != 0

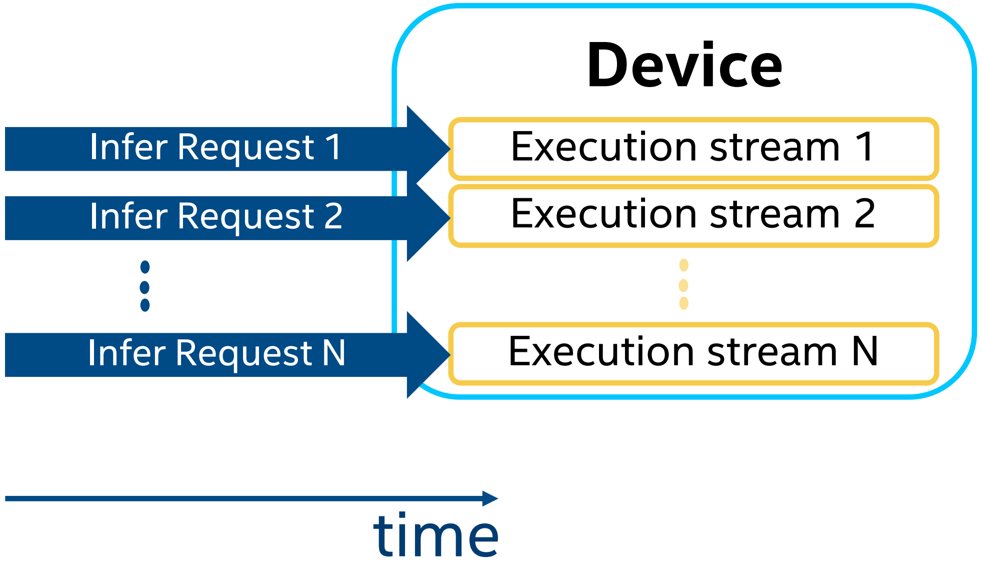
    do something



Inference Engine

Throughput Mode for CPU

- **Latency** – inference time of 1 frame (ms).
- **Throughput** – overall number of frames inferred per 1 second (FPS)
- **“Throughput"** mode allows the Inference Engine to efficiently run multiple infer requests simultaneously, greatly improving the overall throughput.
- Device resources are divided into execution “**streams**” – parts which runs infer requests in parallel



CPU Example:

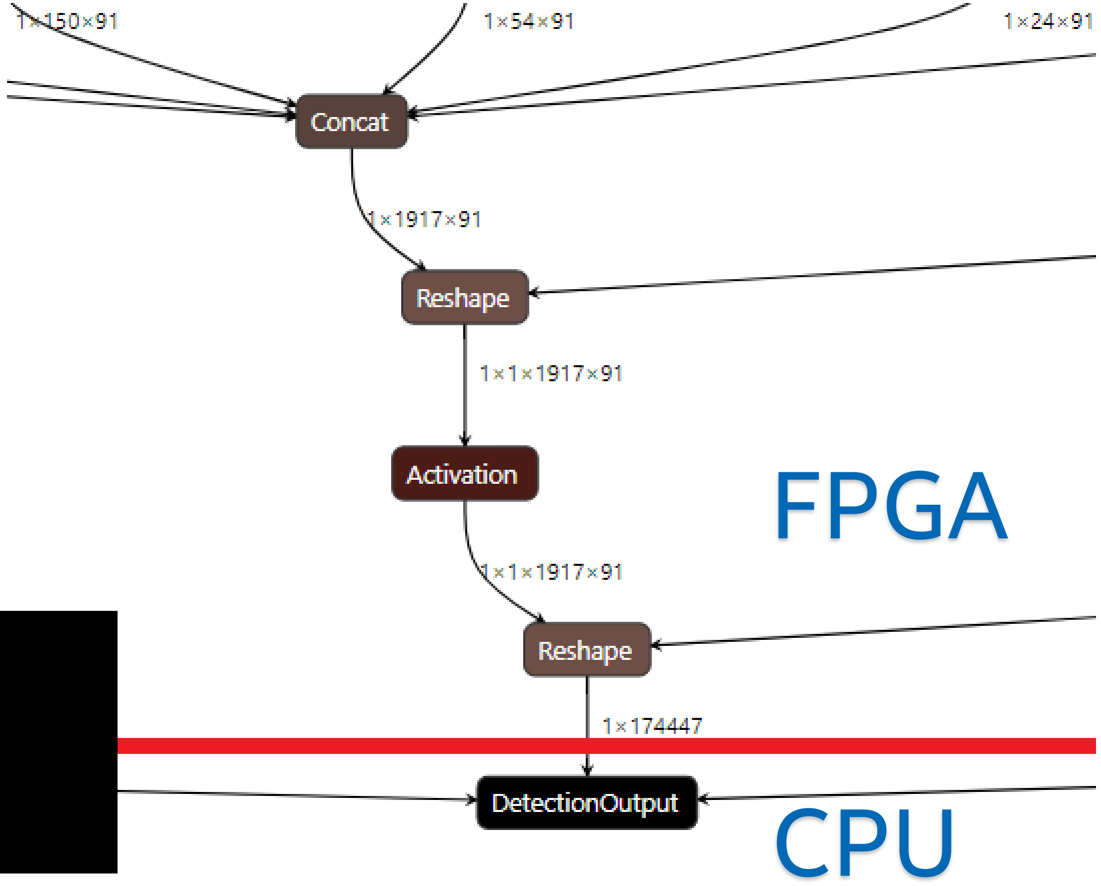
```
ie = IECore()
ie.GetConfig(CPU, KEY_CPU_THROUGHPUT_STREAMS)
```

Inference Engine

Heterogeneous Support

- You can execute different layers on different HW units
- Offload unsupported layers on fallback devices:
  - Default affinity policy
  - Setting affinity manually
- All device combinations are supported (CPU, GPU, FPGA, MYRIAD, HDDL)

```
InferenceEngine::Core core;
auto executable_network =
core.LoadNetwork(reader.getNetwork(),
"HETERO:FPGA,CPU");
```



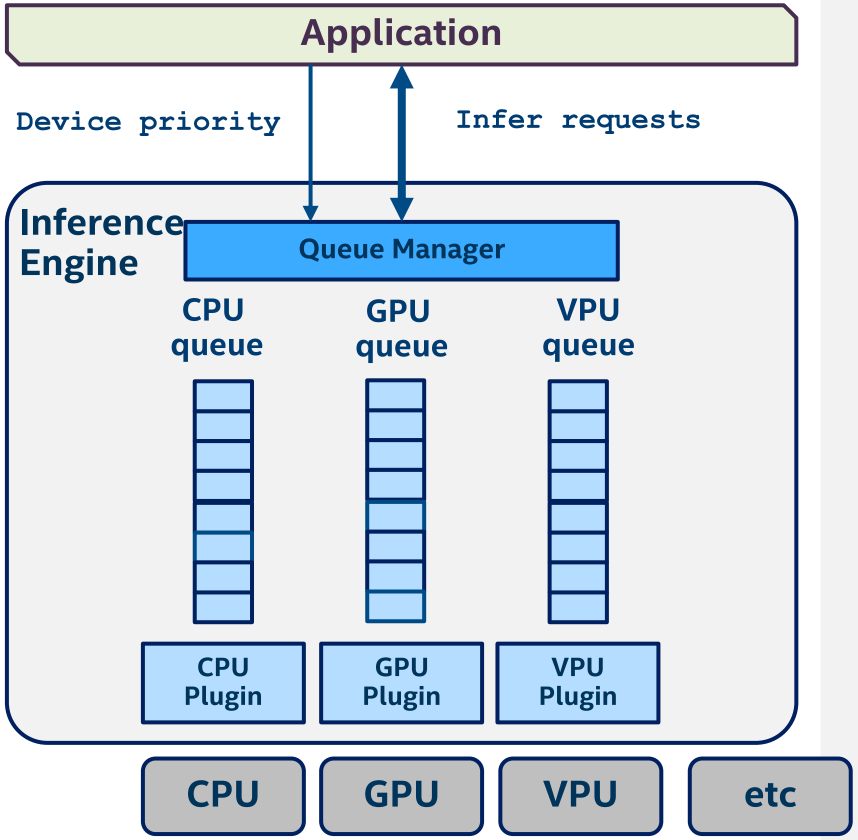
Inference Engine

Multi-device Support

Automatic load-balancing between devices (inference requests level) for full system utilization

- Any combinations of the following devices are supported (CPU, GPU, VPU, HDDL)
- As easy as “-d MULTI:CPU,GPU” for cmd-line option of your favorite sample/demo

```
Core ie;
ExecutableNetwork exec =
ie.LoadNetwork(network,{{“DEVICE_PRIORITIES”,
“CPU,GPU”}}, “MULTI”)
```



Accelerators based on Intel® Movidius™ Vision Processing Unit

April 2021

intel®

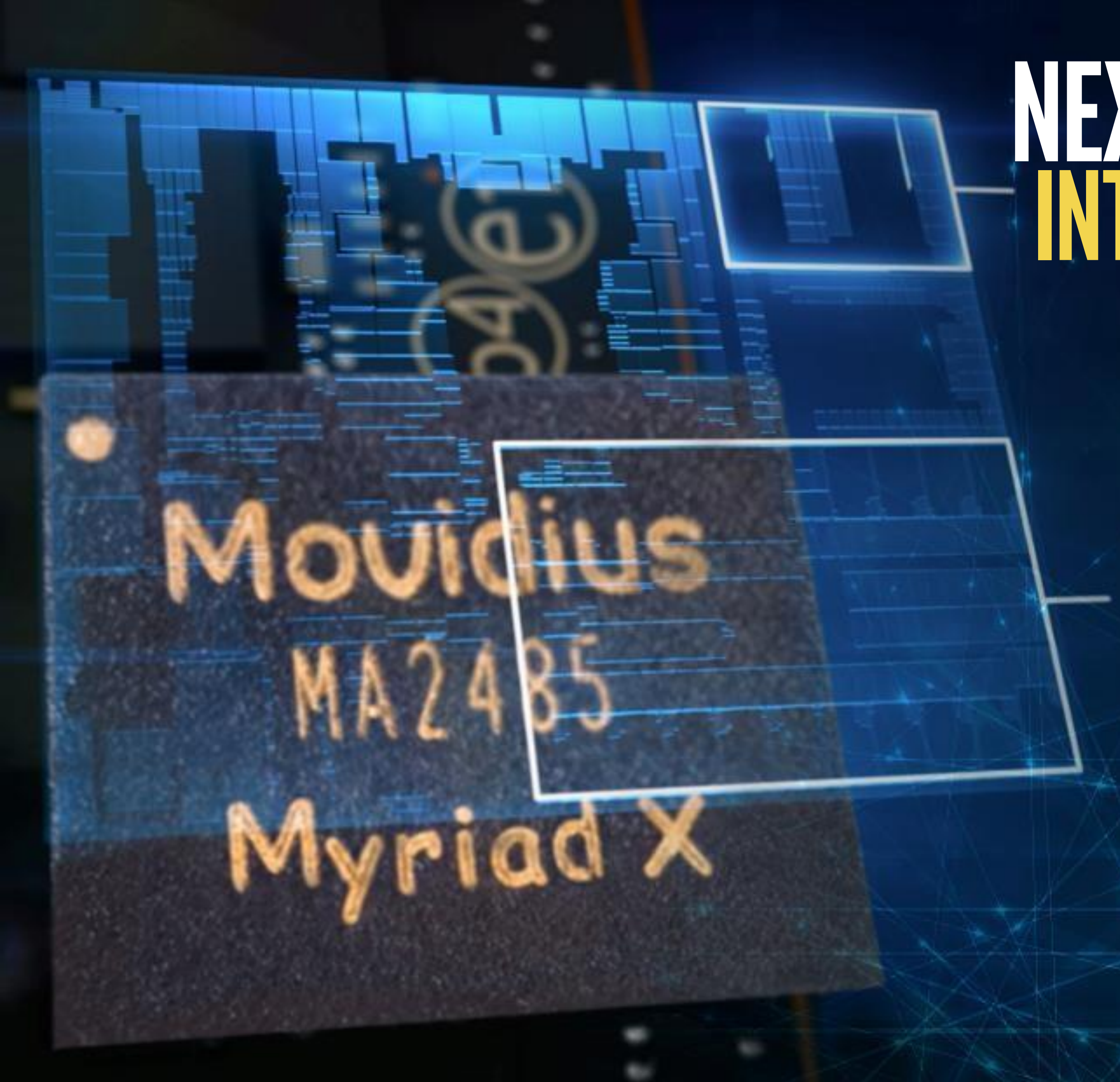
REDEFINING THE AI DEVELOPMENT KIT

INTEL® NEURAL COMPUTE STICK 2

Vision Processing Unit (VPU)	Intel® Movidius™ Myriad™ X VPU
Software Development Kit	Intel® Distribution of OpenVINO™ toolkit
Operating Software Support	Ubuntu* 16.04 or 18.04 LTS (64 bit), Windows® 10 (64 bit), CentOS* 7.4 (64 bit), macOS* 10.4.4, Raspbian*, and other via the open-source distribution of OpenVINO™ toolkit
Supported Framework	TensorFlow*, Caffe*, MXNet*, ONNX*, and PyTorch* / PaddlePaddle* via ONNX* conversion
Connectivity	USB 3.1 Type-A
Dimensions	72.5mm X 27mm X 14mm
Operating Temperature	0° - 40° C
Material Master Number	964486
MSRP	$69 as of July 14th 2019

NEXT GENERATION AI INFERENCE

INTEL® MOVIDIUS™ MYRIAD™ X VPU

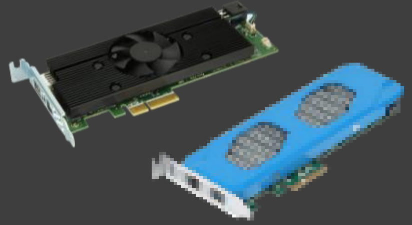


Neural Compute Engine
An entirely new deep neural network (DNN) inferencing engine that offers flexible interconnect and ease of configuration for on-device DNNs and computer vision applications

16 SHAVE Cores
VLIW (DSP) programmable processors are optimized for complex vision & imaging workloads

Hardware-based encoder
for up to 4K video resolution and includes a new stereo depth block that is capable of processing dual 720p feeds at up to 180Hz.

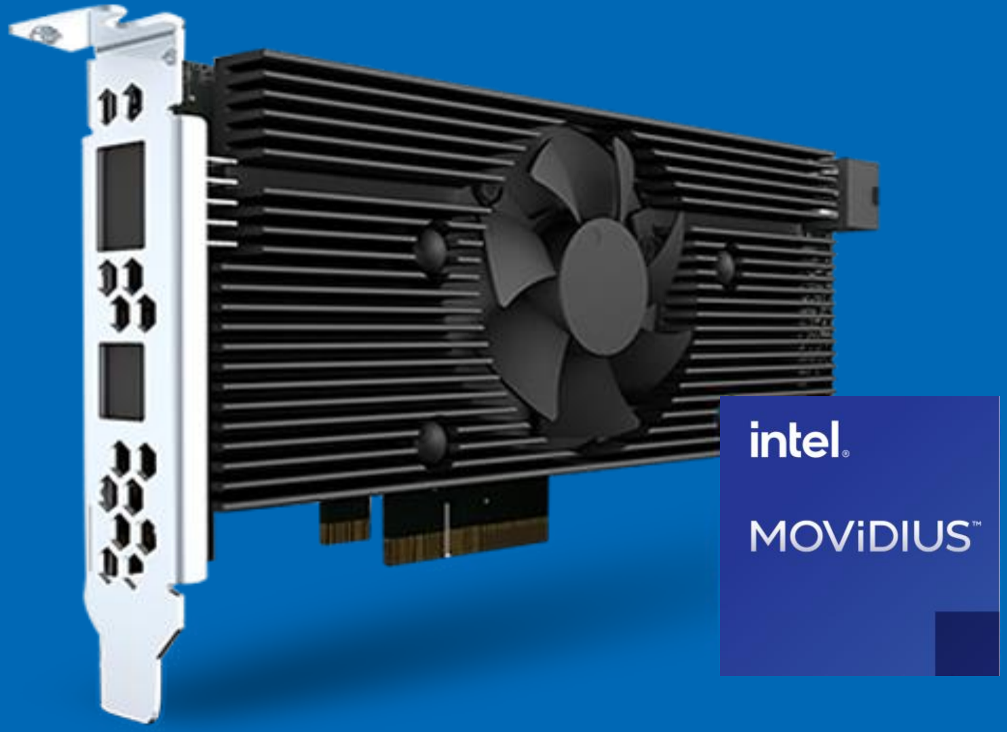
Examples of Intel® Vision Accelerator Design Products
Accelerators based on Intel® Movidius™ VPU

Example card based on Vision Accelerator Designs	1 Intel® Movidius™ VPU	2 Intel® Movidius™ VPUs	 8 Intel® Movidius™ VPUs
Interface	M.2, Key E	miniPCIe	PCIe x4
Currently manufactured by	AAEON an ASUS assoc. co. ADLINK Leading EDGE COMPUTING ADVANTECH Enabling an Intelligent Planet iEi JWIPC NEXCOM tinyGO uzel		
Software tools	INTEL® DISTRIBUTION OF OPENVINO™ TOOLKIT		

*Please contact Intel representative for complete list of ODM manufacturers. Other names and brands may be claimed as the property of others.

Optimization Notice

Intel® Vision Accelerator Design With Intel® Movidius™ Vision Processing Unit (VPU)



- Specialized processors designed to deliver high-performance machine vision at ultra-low power.
- Supports up to 16 video streams per device
- Ideal for camera and network video recorder (NVR) use cases with power, size, and cost constraints
- Supports small memory footprint networks

Multiple Models in One Application Security Barrier Demo

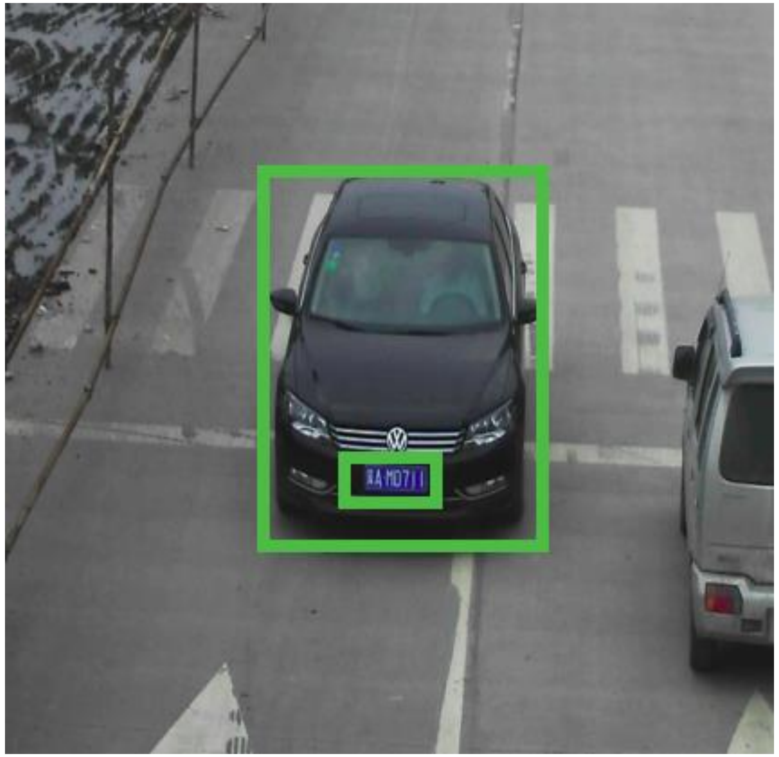
April 2021

Video Analytics in Intel® Distribution of OpenVINO™ Toolkit

Topology	Type	Description
vehicle-license-plate-detection-barrier-0106	Object Detection	MobileNetV2 + SSD-based vehicle and (Chinese) license plate detector
vehicle-attributes-recognition-barrier-0039	Object Recognition	vehicle attributes classification algorithm for a traffic analysis scenario
license-plate-recognition-barrier-0001	Object Recognition	small-footprint network trained end-to-end to recognize Chinese license plates in traffic

vehicle-license-plate-detection-barrier-0106 Use Case/High-Level Description

- MobileNetV2 + SSD-based vehicle and (Chinese) license plate detector for the "Barrier" use case



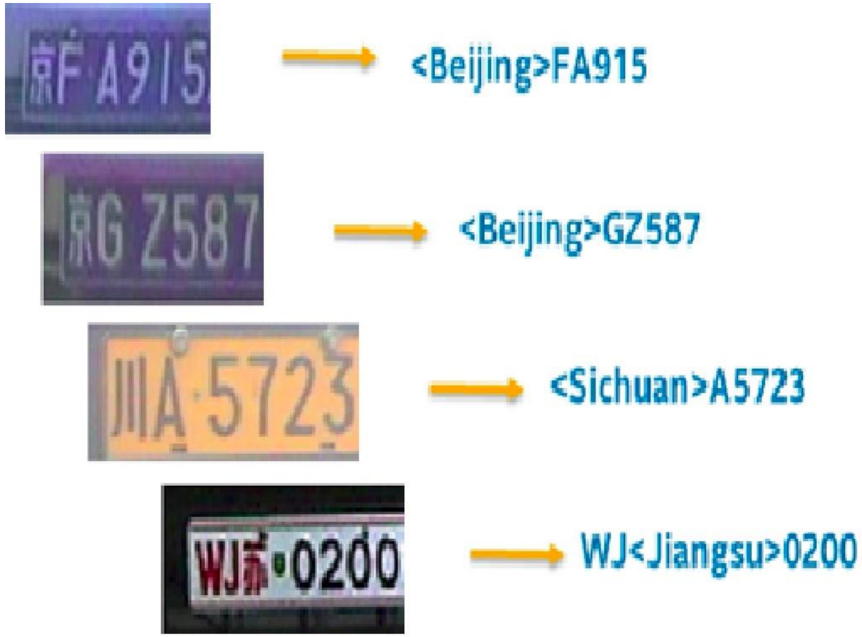
vehicle-attributes-recognition-barrier-0039 Use Case/High-Level Description

- Vehicle attributes classification algorithm for a traffic analysis scenario

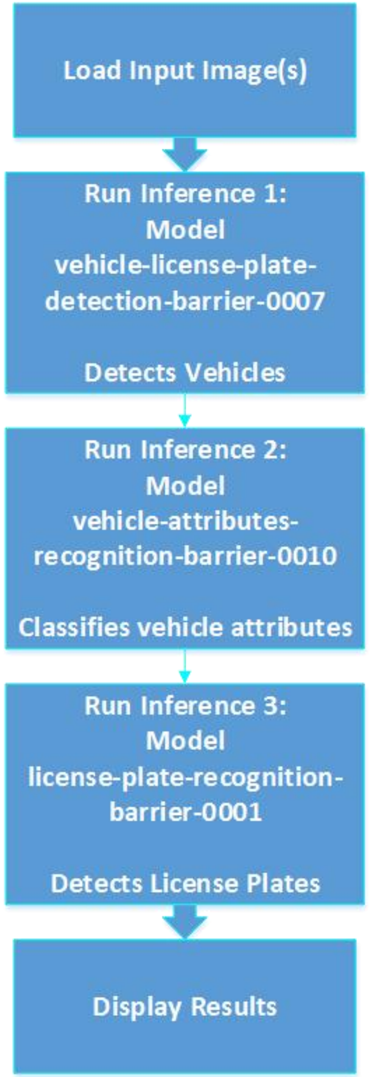
Type: regular
Color: black

license-plate-recognition-barrier-0001 Use Case/High-Level Description

- Small-footprint network trained E2E to recognize Chinese license plates in traffic scenarios.
- Note: The license plates in the image are modified from the originals.



Security Barrier Demo



Load Input Image(s)

Run Inference 1: Model vehicle-license-plate-detection-barrier-0007

Detects Vehicles

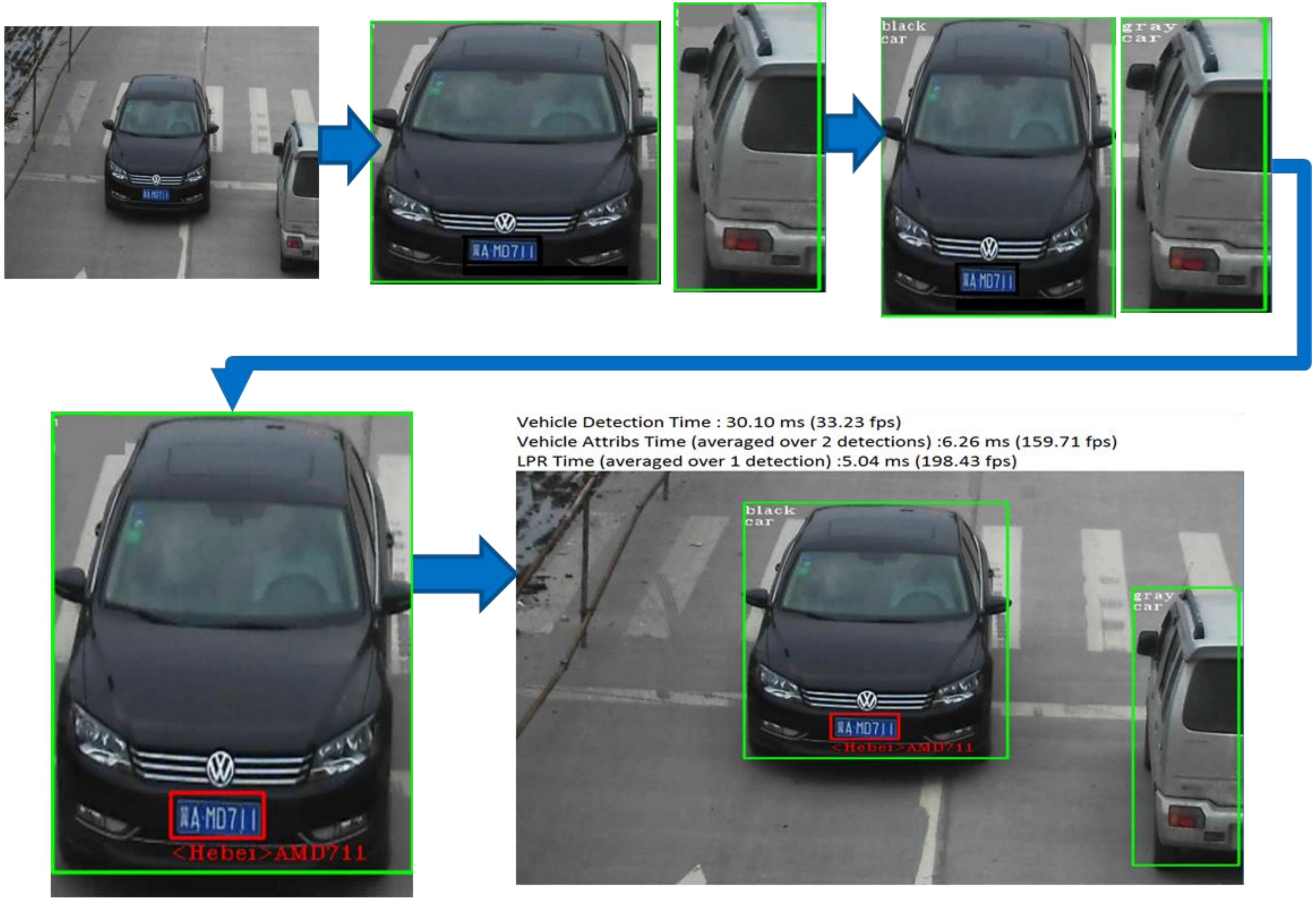
Run Inference 2: Model vehicle-attributes-recognition-barrier-0010

Classifies vehicle attributes

Run Inference 3: Model license-plate-recognition-barrier-0001

Detects License Plates

Display Results



```
Vehicle Detection Time : 30.10 ms (33.23 fps)
Vehicle Attribs Time (averaged over 2 detections) :6.26 ms (159.71 fps)
LPR Time (averaged over 1 detection) :5.04 ms (198.43 fps)
```

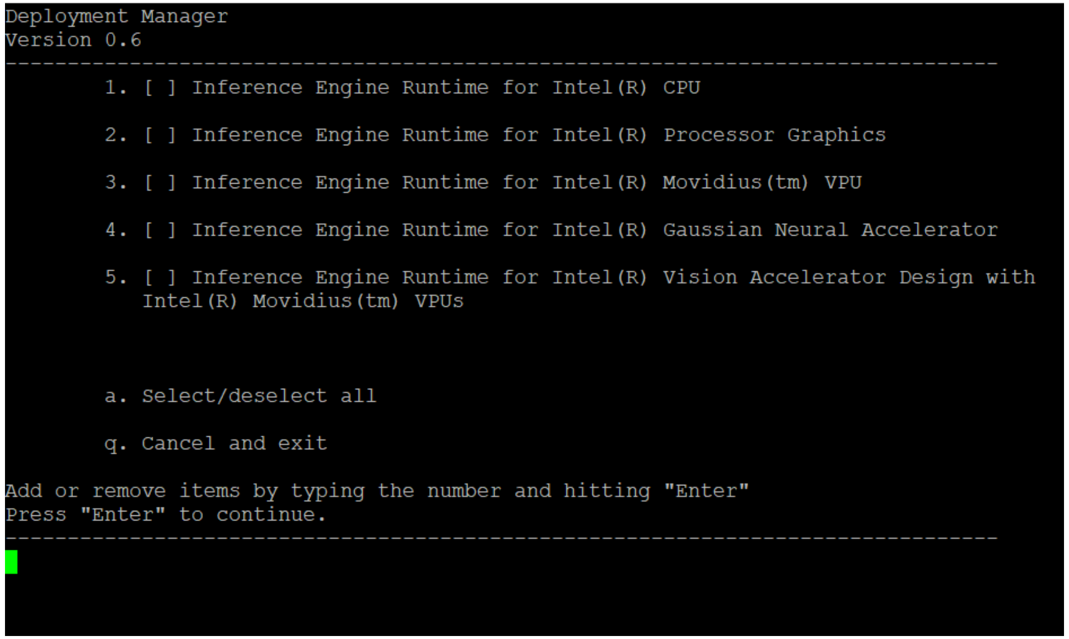
Deployment Manager

April 2021

Deployment Manager

The Deployment Manager of Intel® Distribution of OpenVINO™ creates a deployment package by assembling the **model**, **IR files**, **your application**, and associated **dependencies** into a runtime package for your target device.

- Create Deployment Package
  - Interactive Mode
  - Standard CLI Mode
    - ./deployment_manager.py <--targets> [--output_dir] [--archive_name] [--user_data]
- Deploy Package on Target
  1. Unpack the archive
     - tar xf openvino_deployment_package.tar.gz -C <destination_dir>
  2. Install additional dependencies
     - sudo -E ./install_openvino_dependencies.sh
  3. Set up the environment variables
     - source ./bin/setupvars.sh



```
Deployment Manager
Version 0.6
--------------------------------------------------------------------------------
        1. [ ] Inference Engine Runtime for Intel(R) CPU

        2. [ ] Inference Engine Runtime for Intel(R) Processor Graphics

        3. [ ] Inference Engine Runtime for Intel(R) Movidius(tm) VPU

        4. [ ] Inference Engine Runtime for Intel(R) Gaussian Neural Accelerator

        5. [ ] Inference Engine Runtime for Intel(R) Vision Accelerator Design with
           Intel(R) Movidius(tm) VPUs


        a. Select/deselect all

        q. Cancel and exit

Add or remove items by typing the number and hitting "Enter"
Press "Enter" to continue.
--------------------------------------------------------------------------------
```

Conditional Compilation for Particular Models

April 2021

intel®

Conditional Compilation for Particular Models

https://github.com/openvinotoolkit/openvino/wiki/ConditionalCompilation

Conditional compilation can significantly reduce OpenVINO™ binaries size by excluding unnecessary components for particular models inference:

- layers and graph transformations in nGraph and plugins
- nGraph operations
- jit kernels in a CPU plugin
- arbitrary code that is not used for particular model inference

However, conditional compilation has a significant drawback - **the resulting OpenVINO runtime will work only with a limited set of models and devices.**

Conditional compilation has two stages:

- Collecting information about code usage
  - Run CMake with **Selective Build** and **Instrumentation and Tracing** Enabled
  - Select a models to be used
  - Run target application with **ITT collector** and generate a .csv file contains the analysis statistics
- Building the result binaries without unused components or parts
  - Re-run CMake with the .csv file loaded
  - Watch for the CPU plugin library size

Building for devices with different ISA

- The analysis step should be performed on target devices and all CSV files with statistics should be copied to the build machine.

Intel® DevCloud for the Edge

April 2021

Accelerate Test Cycles with the Intel® DevCloud for the Edge

A Development Sandbox for Developers, Researchers, and Startups to Test AI and Vision Workloads Remotely before Deployment.

With the Intel® DevCloud for the Edge users can:

- **Prototype** on the latest hardware and software to future proof the solution
- **Benchmark** the customized AI application
- Run AI applications from **anywhere in the world**
- **Reduce** development time and cost

DL Workbench + Intel® DevCloud for the Edge

Developers can now graphically analyze models using the DL Workbench on Intel® DevCloud for the Edge (instead of local machine only) to compare, visualize and fine-tune a solution against multiple remote hardware configurations

Intel® DevCloud for the Edge

Deploy and scale

intel ATOM | intel CORE i7 | intel XEON | intel IRIS Pro GRAPHICS | Intel® GNA (IP) | intel MOVIDIUS | intel ARRIA 10

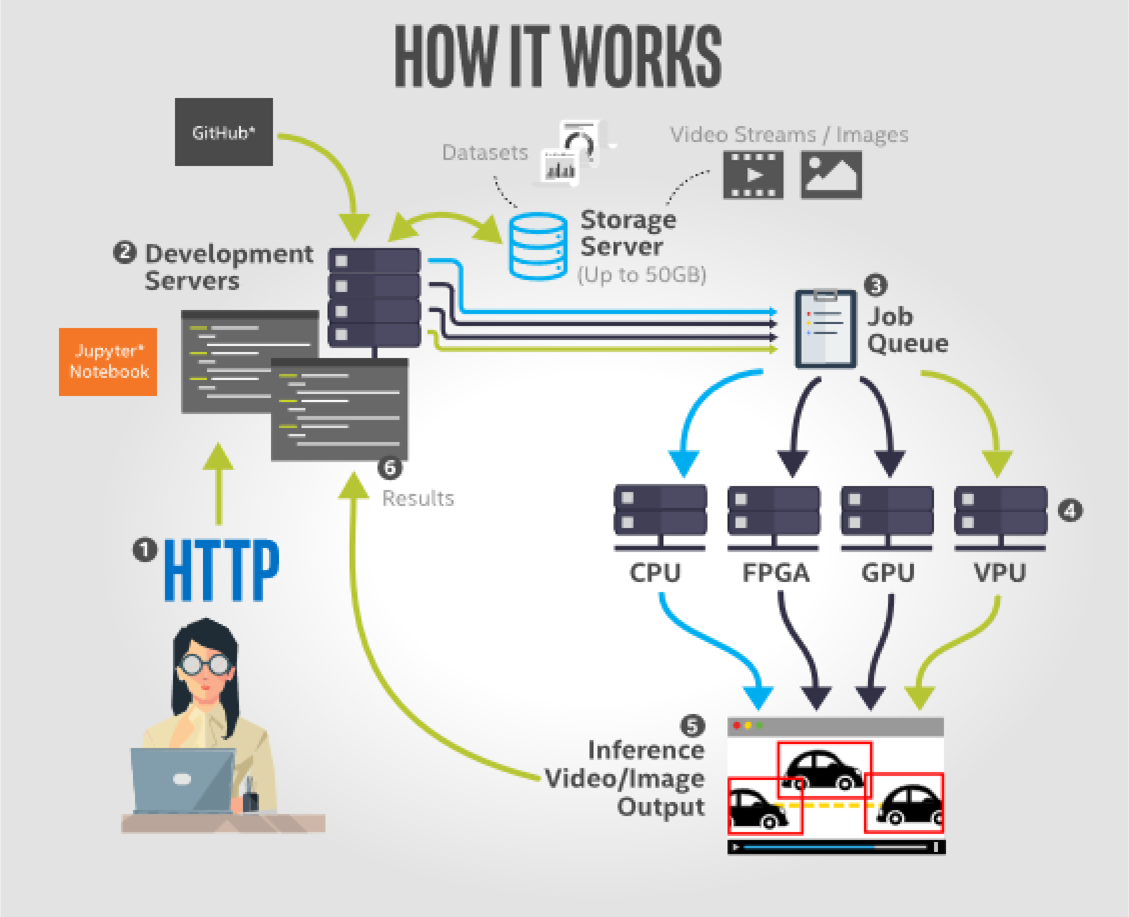
For more information visit ▸ https://devcloud.intel.com/edge/

How Intel® DevCloud For the Edge Works

1 Access the Intel® DevCloud for the Edge through your web browser

2 Develop and test applications online using GitHub and datasets stored in the Intel® DevCloud's cloud storage

3 Test sample code to showcase benchmarking capabilities to customers. Customers can also test their own applications for benchmark performance results



4 Runs tests to benchmark the application's performance on selected Intel processors and accelerators

5 Produces the inference video/image as output

6 Provides performance results to find the optimal hardware for the tested AI vision application

Resources to Get Started

OpenVINO™

Intel® Distribution of OpenVINO™ Toolkit:
https://software.intel.com/content/www/us/en/develop/tools/openvino-toolkit.html

Intel® Edge Software Hub

Download prevalidated software to learn, develop, and test your solutions for the edge.

Intel® Edge Software Hub:
https://software.intel.com/content/www/us/en/develop/topics/iot/edge-solutions.html

Intel® DevCloud FOR THE EDGE

Intel® DevCloud for the Edge:
https://devcloud.intel.com/edge/home

To get access to the full video series, please complete the short form: http://intel.ly/38B9ix6

intel®