OpenVINO™

Intel® Distribution of OpenVINO™ toolkit

Dr. Séverine Habert, Deep Learning Software Engineer

April 9th, 2021

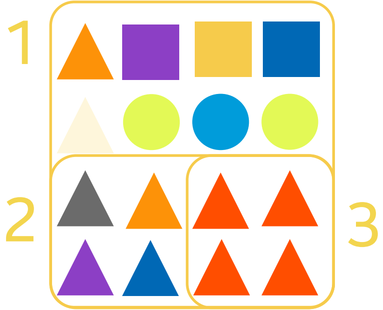
intel®

lrz

AI Compute Considerations

How do you determine the right computing for your AI needs?

WORKLOADS



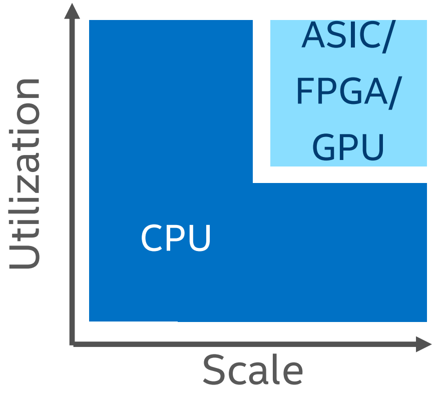
What is my workload profile?

REQUIREMENTS

RESPONSE TIME
SECURITY
DATA
VERSATILITY
SCALABILITY
COST
POWER
SIZE
RELIABILITY
THROUGHPUT
EFFICIENCY
MANAGEABILITY
FLEXIBILITY
ACCURACY

What are my use case requirements?

DEMAND



How prevalent is AI in my environment?

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

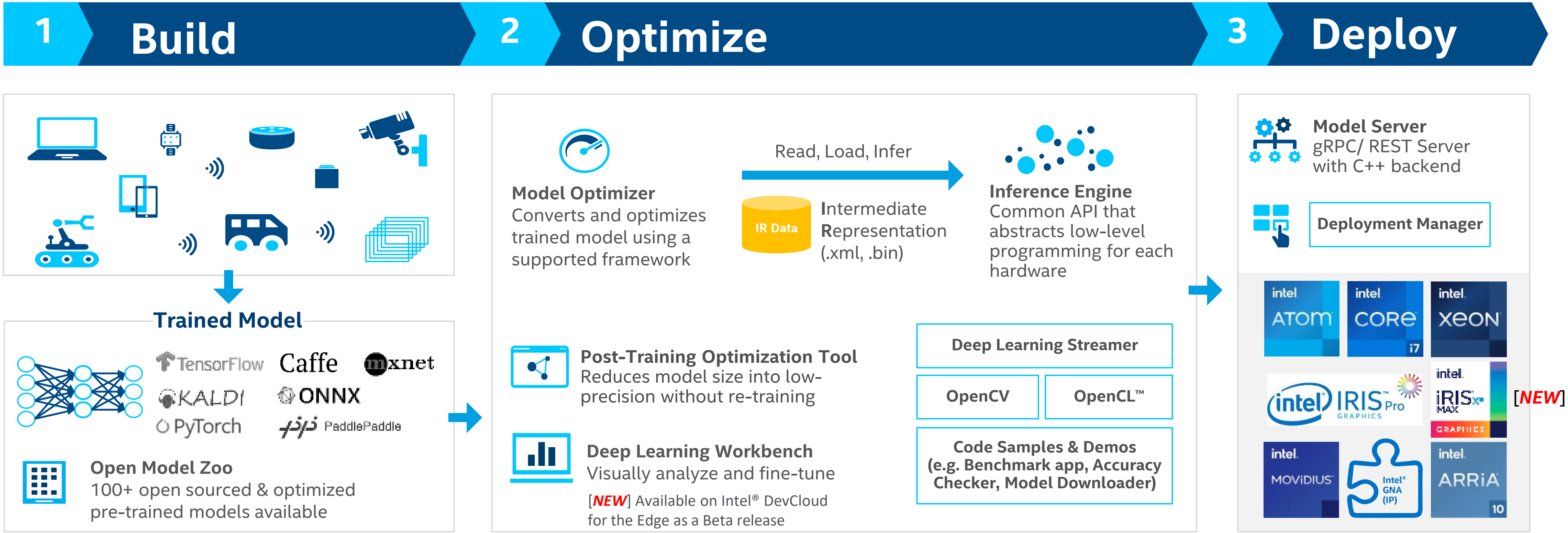
High-Performance,
Deep Learning Inference

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™.

Three steps for the Intel® Distribution of OpenVINO™ toolkit



1 **Build**

2 **Optimize**

3 **Deploy**

Trained Model

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

Model Optimizer
Converts and optimizes trained model using a supported framework

Read, Load, Infer

IR Data

Intermediate Representation (.xml, .bin)

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

[*NEW*] Available on Intel® DevCloud for the Edge as a Beta release

Deep Learning Streamer

OpenCV

OpenCL™

Code Samples & Demos (e.g. Benchmark app, Accuracy Checker, Model Downloader)

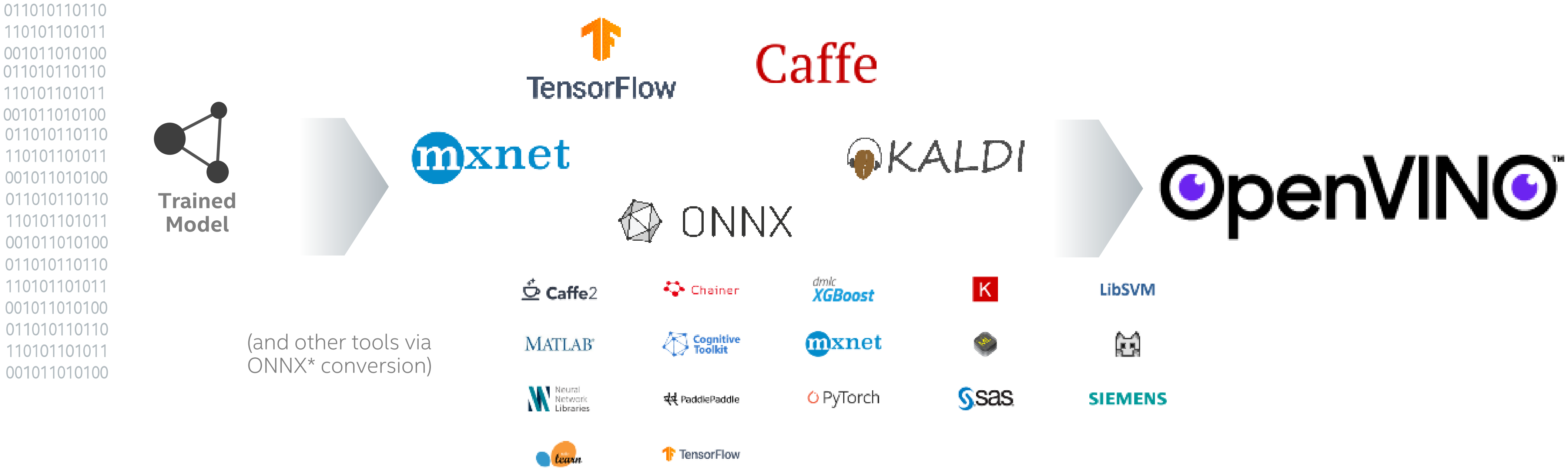
Model Server
gRPC/ REST Server with C++ backend

Deployment Manager

[*NEW*]

Supported Frameworks

Breadth of supported frameworks to enable developers with flexibility



(and other tools via ONNX* conversion)

Supported Frameworks and Formats ▸ https://docs.openvinotoolkit.org/latest/_docs_IE_DG_Introduction.html#SupportedFW
Configure the Model Optimizer for your Framework ▸ https://docs.openvinotoolkit.org/latest/_docs_MO_DG_prepare_model_Config_Model_Optimizer.html

Model Optimization

Breadth of supported frameworks to enable developers with flexibility

Model Optimizer loads a model into memory, reads it, builds the internal representation of the model, optimizes it, and produces the **Intermediate Representation**.

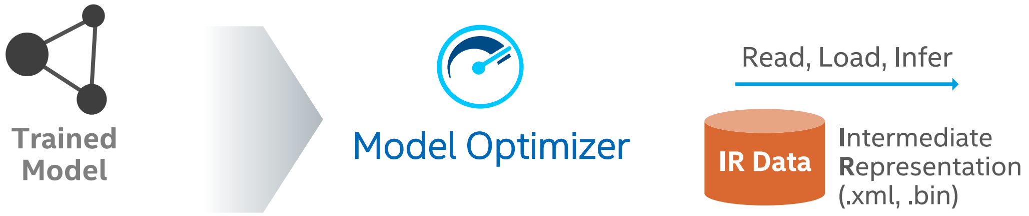
Optimization techniques available are:

— Linear operation fusing

— Stride optimizations

— Group convolutions fusing

Note: Except for ONNX (.onnx model formats), all models have to be converted to an IR format to use as input to the Inference Engine



.xml – describes the network topology
.bin – describes the weights and biases binary data

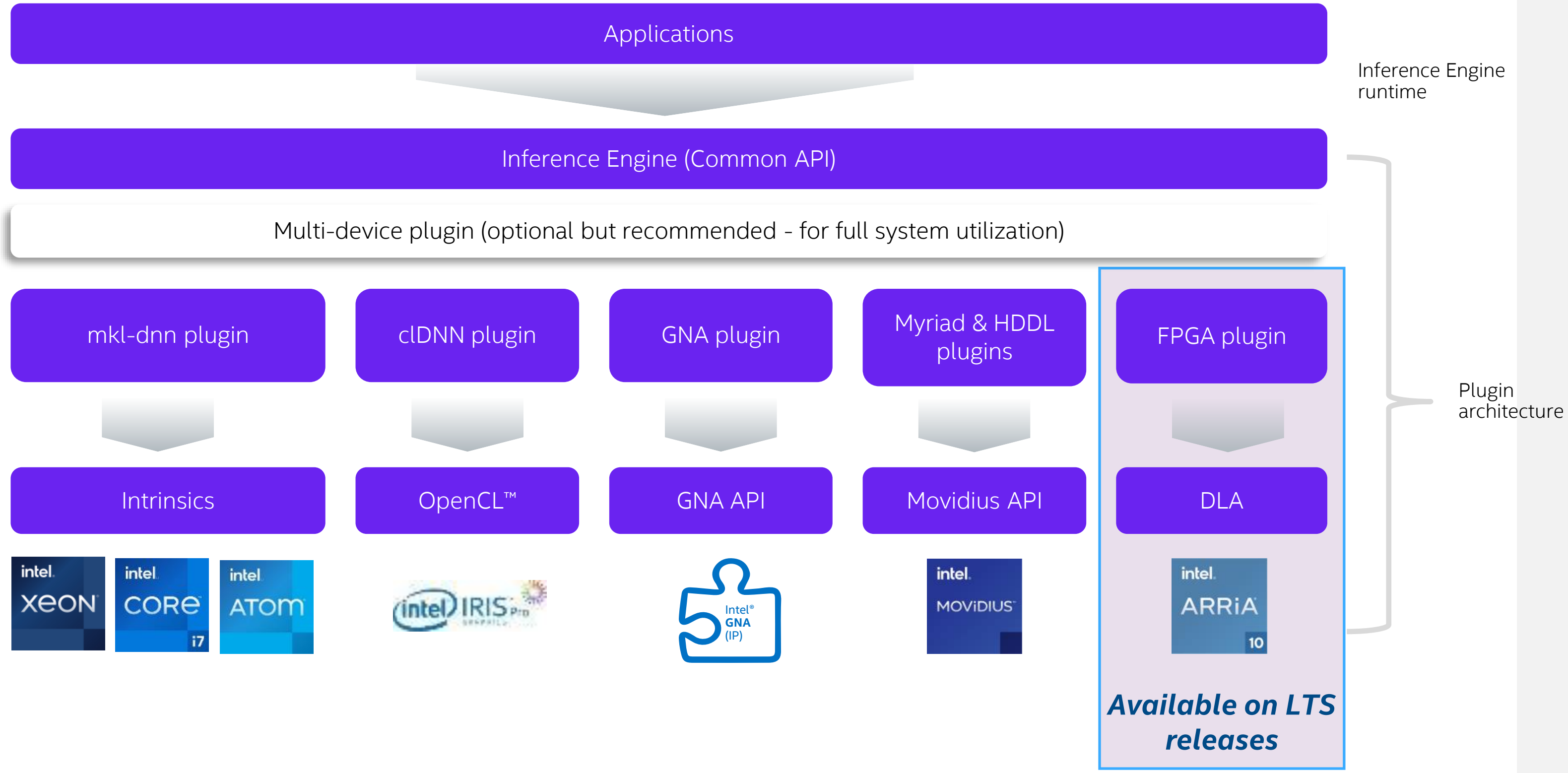
Optimal Model Performance Using the Inference Engine

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

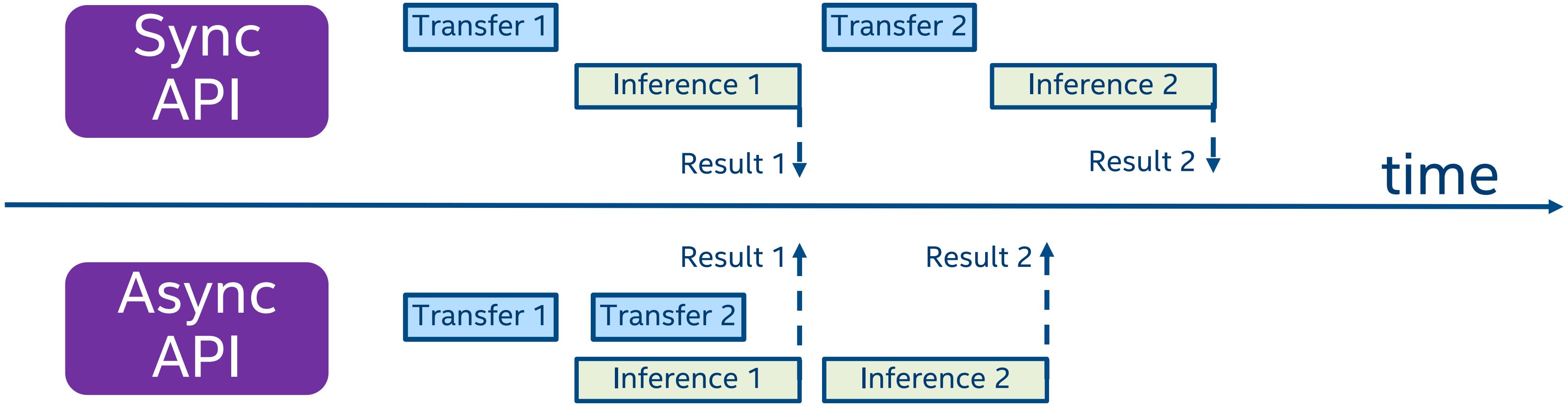
GNA = Gaussian mixture model and Neural Network Accelerator

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

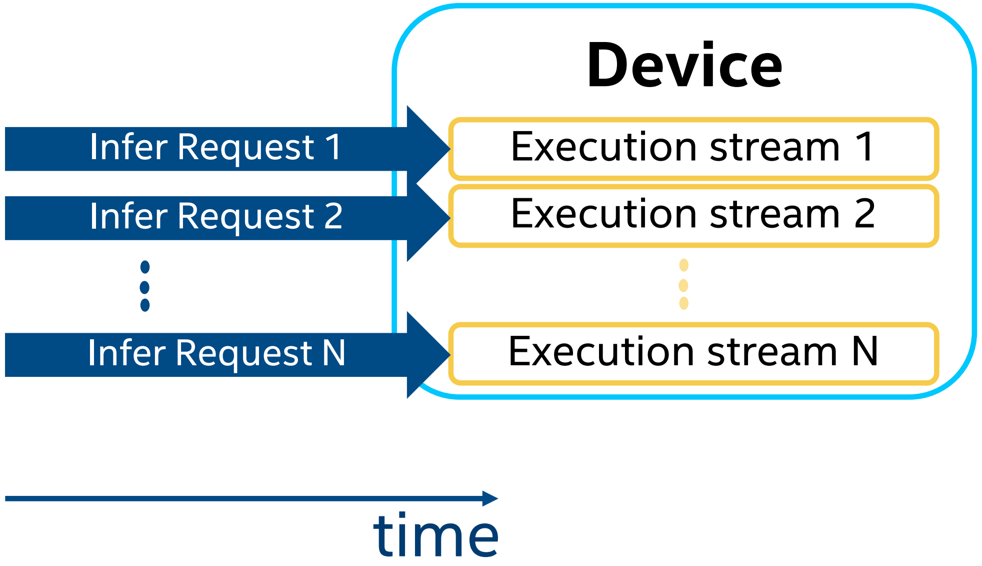
    do something



Inference Engine

Throughput Mode for CPU, iGPU and VPU

- Latency – inference time of 1 frame (ms).
- Throughput – overall amount 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:

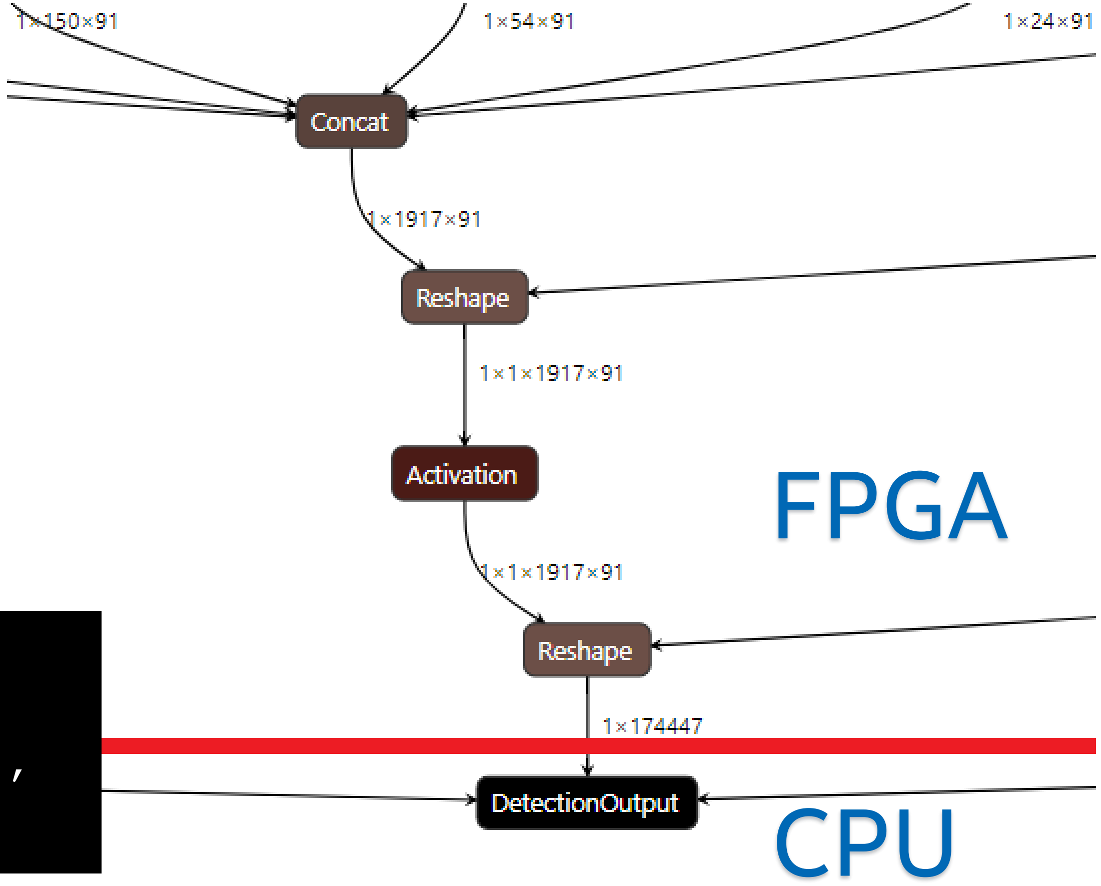
```
ie = IECore()
ie.GetConfig(CPU, KEY_CPU_THROUGHPUT_STREAMS)
```

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 (`CNNLayer::affinity`)
- All device combinations are supported (CPU, GPU, FPGA, MYRIAD, HDDL)
- Samples/demos usage “`-d HETERO:FPGA,CPU`”

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



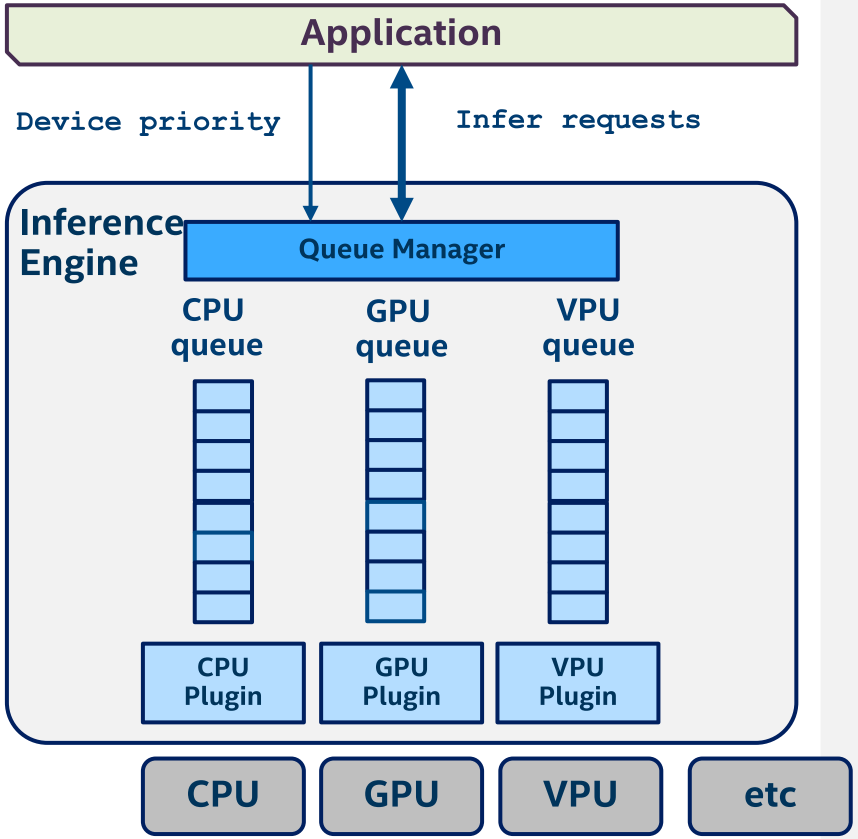
Inference Engine

Multi-device Support

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

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

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

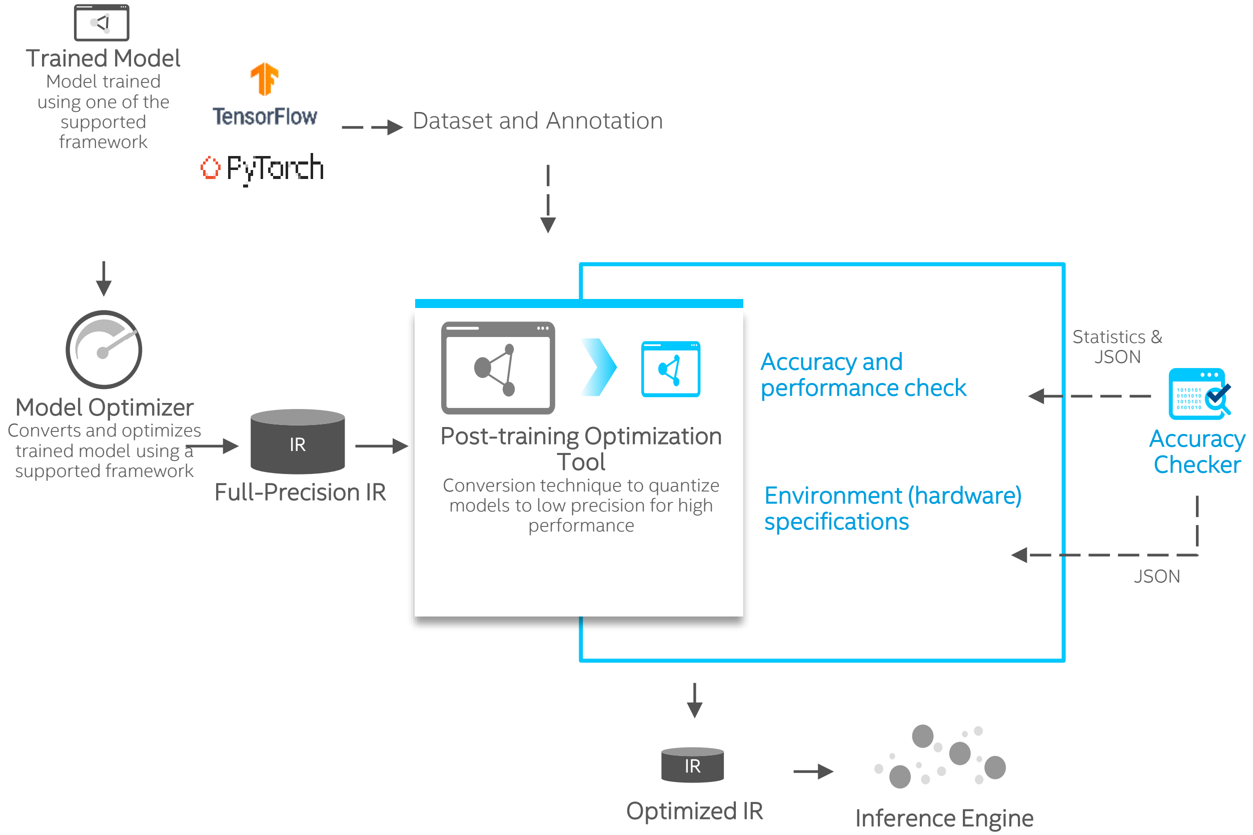


Post-Training Optimization Tool

Conversion technique that reduces model size into low-precision without re-training

Reduces model size **while also improving latency, with little degradation** in model accuracy and without model re-training.

Different optimization approaches are supported: quantization algorithms, etc.

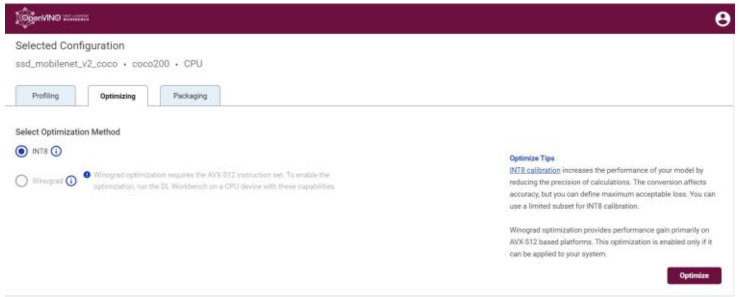


Trained Model
Model trained using one of the supported framework
TensorFlow
PyTorch
Dataset and Annotation
Model Optimizer
Converts and optimizes trained model using a supported framework
IR
Full-Precision IR
Post-training Optimization Tool
Conversion technique to quantize models to low precision for high performance
Accuracy and performance check
Environment (hardware) specifications
Statistics & JSON
Accuracy Checker
JSON
IR
Optimized IR
Inference Engine

Deep Learning Workbench

Web-based UI extension tool for model analyses and graphical measurements

- **Visualizes performance data for** topologies and layers to aid in model analysis
- **Automates analysis** for optimal performance configuration (streams, batches, latency)
- **Experiment with INT8 or Winograd calibration** for optimal tuning using the Post Training Optimization Tool
- Provide **accuracy information** through accuracy checker
- **Direct access to models** from public set of Open Model Zoo
- Enables **remote profiling**, allowing the collection of performance data from multiple different machines without any additional set-up.



Selected Configuration

ssd_mobilenet_v2_coco • coco200 • CPU

Select Optimization Method

● INT8 ⓘ

○ Winograd ⓘ

Winograd optimization requires the AVX-512 instruction set. To enable the optimization, run the DL Workbench on a CPU device with these capabilities.

Optimize Tips

INT8 calibration increases the performance of your model by reducing the precision of calculations. The conversion affects accuracy, but you can define maximum acceptable loss. You can use a limited subset for INT8 calibration.

Winograd optimization provides performance gain primarily on AVX-512 based platforms. This optimization is enabled only if it can be applied to your system.

Pre-Trained Models and Public Models

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

Use free **Pre-trained Models** to speed up development and deployment

Take advantage of the **Model Downloader** and other automation tools to quickly get started

Iterate with the **Accuracy Checker** to validate the accuracy of your models

100+ Pre-trained Models
Common AI tasks

Object Detection
Object Recognition
Reidentification
Semantic Segmentation
Instance Segmentation
Human Pose Estimation
Image Processing
Text Detection
Text Recognition
Text Spotting
Action Recognition
Image Retrieval
Compressed Models
Question Answering

100+ Public Models
Pre-optimized external models

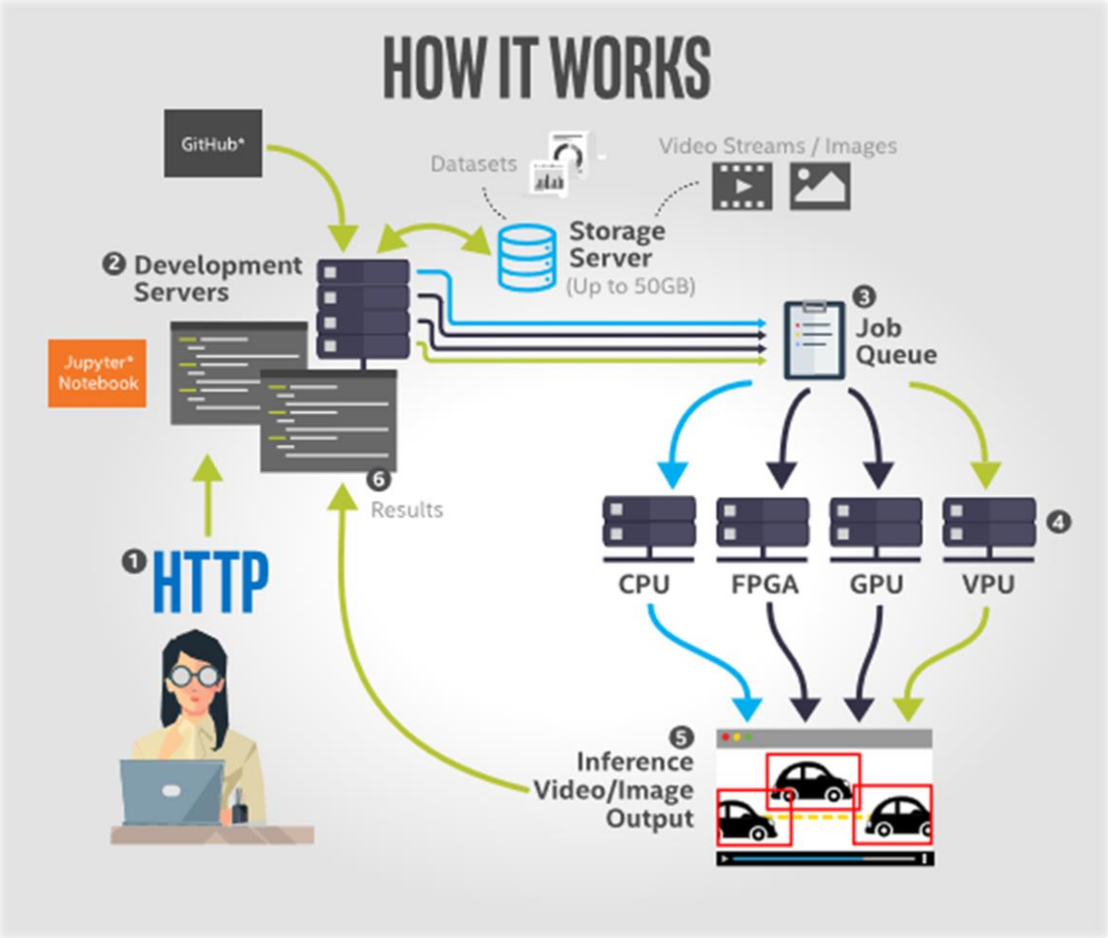
Classification
Segmentation
Object Detection
Human Pose Estimation
Monocular Depth Estimation
Image Inpainting
Style Transfer
Action Recognition
Colorization

Questions?

Intel® DevCloud for the Edge

Accelerate Time to Production with Intel® DevCloud for the Edge

See immediate AI Model performance across Intel's vast array of Edge Solutions



- **Instant, Global Access**
  Run AI applications from anywhere in the world
- **Prototype on the Latest Hardware and Software**
  Develop knowing you're using the latest Intel technology
- **Benchmark your Customized AI Application**
  Immediate feedback - frames per second, performance
- **Reduce Development Time and Cost**
  Quickly find the right compute for your edge solution

Learn more

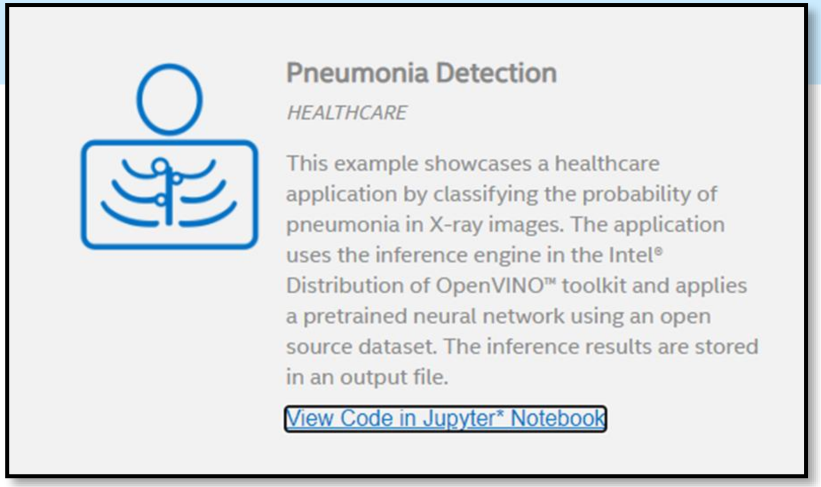
Sign up now for access

Demo

✓ **Pneumonia Classification with Class Activation Maps**

https://devcloud.intel.com/edge/advanced/sample_applications/

-->**Development Environment:** OpenVINO 2020.3 Jupyter Notebook



Pneumonia Detection

HEALTHCARE

This example showcases a healthcare application by classifying the probability of pneumonia in X-ray images. The application uses the inference engine in the Intel® Distribution of OpenVINO™ toolkit and applies a pretrained neural network using an open source dataset. The inference results are stored in an output file.

View Code in Jupyter* Notebook

Ready to get started?

Download directly from Intel for free

Intel® Distribution of OpenVINO ™ toolkit
(Recommended)

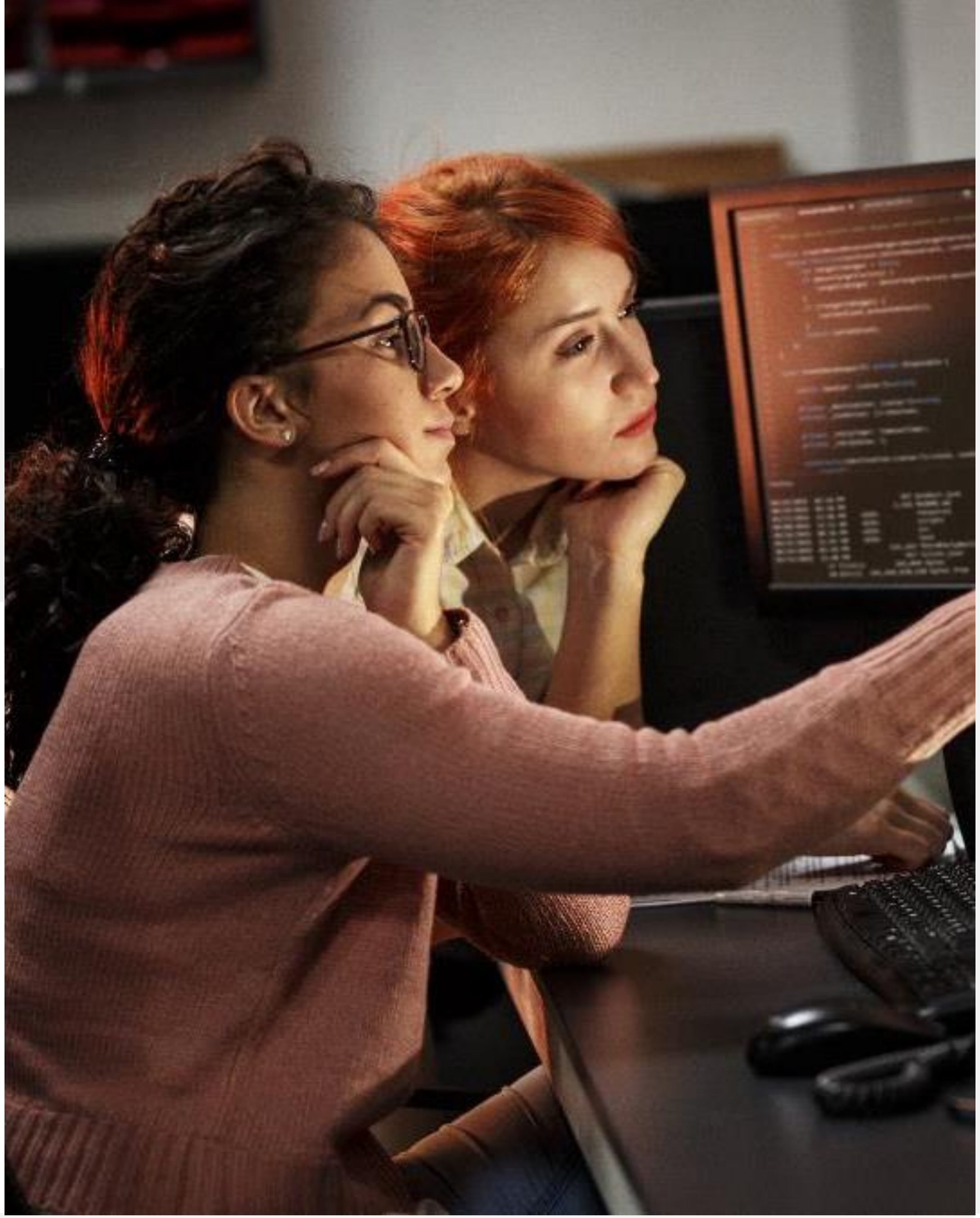
Also available from

Intel's Edge Software Hub | Intel® DevCloud for the Edge | PIP |
DockerHub | Dockerfile | Anaconda Cloud | YUM | APT

Build from source

GitHub | Gitee (for China)

Choose & Download



Choose between Distributions

Tool/Component	Intel® Distribution of OpenVINO™ toolkit	OpenVINO™ toolkit (open source)	Open Source Directory
Installer (including necessary drivers)	✓		
Model Optimizer	✓	✓	https://github.com/openvinotoolkit/openvino/tree/master/model-optimizer
Inference Engine - Core	✓	✓	https://github.com/openvinotoolkit/openvino/tree/master/inference-engine
Intel CPU plug-in	✓ Intel® Math Kernel Library (Intel® MKL) only[1]	✓ BLAS, Intel® MKL[1], jit (Intel MKL)	https://github.com/openvinotoolkit/openvino/tree/master/inference-engine
Intel GPU (Intel® Processor Graphics) plug-in	✓	✓	https://github.com/openvinotoolkit/openvino/tree/master/inference-engine
Heterogeneous plug-in	✓	✓	https://github.com/openvinotoolkit/openvino/tree/master/inference-engine
Intel GNA plug-in	✓	✓	https://github.com/openvinotoolkit/openvino/tree/master/inference-engine
Intel® FPGA plug-in	✓		
Intel® Neural Compute Stick (1 & 2) VPU plug-in	✓	✓	https://github.com/openvinotoolkit/openvino/tree/master/inference-engine
Intel® Vision Accelerator based on Movidius plug-in	✓		
Multi-device & hetero plug-ins	✓	✓	
Public and Pretrained Models - incl. Open Model Zoo (IR models that run in IE + open sources models)	✓	✓	https://github.com/openvinotoolkit/open_model_zoo
Samples (APIs)	✓	✓	https://github.com/openvinotoolkit/openvino/tree/master/inference-engine
Demos	✓	✓	https://github.com/openvinotoolkit/open_model_zoo
Traditional Computer Vision			
OpenCV*	✓	✓	https://github.com/opencv/opencv
Intel® Media SDK	✓	✓[2]	https://github.com/Intel-Media-SDK/MediaSDK
OpenCL™ Drivers & Runtimes	✓	✓[2]	https://github.com/intel/compute-runtime
FPGA Runtime Environment, Deep Learning Acceleration & Bitstreams (Linux* only)	✓		

System Requirements

	Intel® Platforms	Compatible Operating Systems
Target Solution Platforms	CPU ▪ 6th-10th generation Intel® Core™ and Xeon® processors ▪ 1st and 2nd generation Intel® Xeon® Scalable processors	▪ Ubuntu* 18.04.3 LTS (64 bit) ▪ Microsoft Windows* 10 (64 bit) ▪ CentOS* 7.4 (64 bit) ▪ macOS* 10.13 & 10.14 (64 bit)
	▪ Intel® Pentium® processor N4200/5, N3350/5, N3450/5 with Intel® HD Graphics	▪ Yocto Project* Poky Jethro v2.0.3 (64 bit)
	Iris® Pro & Intel® HD Graphics ▪ 6th-10th generation Intel® Core™ processor with Intel® Iris™ Pro graphics & Intel® HD Graphics ▪ Intel® Xeon® processor with Intel® Iris™ Pro Graphics & Intel® HD Graphics (excluding E5 product family, which does not have graphics[1])	▪ Ubuntu 18.04.3 LTS (64 bit) ▪ Windows 10 (64 bit) ▪ CentOS 7.4 (64 bit)
	FPGA ▪ Intel® Arria® FPGA 10 GX development kit ▪ Intel® Programmable Acceleration Card with Intel® Arria® 10 GX FPGA operating systems ▪ OpenCV* & OpenVX* functions must be run against the CPU or Intel® Processor Graphics (GPU)	▪ Ubuntu 18.04.2 LTS (64 bit) ▪ CentOS 7.4 (64 bit)
	VPU: Intel Movidius™ Neural Compute Stick:, Intel® Neural Compute Stick2	▪ Ubuntu 18.04.3 LTS (64 bit) CentOS 7.4 (64 bit) ▪ Windows 10 (64 bit) macOS* (64 bit) Raspbian (target only)
	Intel® Vision Accelerator Design Products ▪ Intel® Vision Accelerator Design with Intel® Arria10 FPGA	▪ Ubuntu 18.04.2 LTS (64 bit)
	▪ Intel® Vision Accelerator Design with Intel® Movidius™ VPUs	▪ Ubuntu 8.04.3 LTS (64 bit) ▪ Windows 10 (64 bit)
Development Platforms	▪ 6th-10th generation Intel® Core™ and Intel® Xeon® processors ▪ 1st and 2nd generation Intel® Xeon® Scalable processors	▪ Ubuntu* 18.04.3 LTS (64 bit) ▪ Windows® 10 (64 bit) ▪ CentOS* 7.4 (64 bit) ▪ macOS* 10.13 & 10.14 (64 bit)
Additional Software Requirements	Linux* build environment required components ▪ OpenCV 3.4 or higher ▪ GNU Compiler Collection (GCC) 3.4 or higher ▪ CMake* 2.8 or higher ▪ Python* 3.4 or higher Microsoft Windows* build environment required components ▪ Intel® HD Graphics Driver (latest version)† ▪ OpenCV 3.4 or higher ▪ Intel® C++ Compiler 2017 Update 4 ▪ CMake 2.8 or higher ▪ Python 3.4 or higher ▪ Microsoft Visual Studio* 2015	
External Dependencies/Additional Software		View Product Site, detailed System Requirements

Commonly Asked Questions

Can I use the Intel® Distribution of OpenVINO™ toolkit for commercial usage? Yes, the Intel® Distribution of OpenVINO™ toolkit is licensed under Intel's End User License Agreements and the open-sourced OpenVINO™ toolkit is licensed under Apache License 2.0. For information, review the licensing directory inside the package.

Is the Intel® Distribution of OpenVINO™ toolkit subject to export control? Yes, the ECCN is EAR99.

How often does the software get updated? Standard releases are updated 3-4 times a year, while LTS releases are updated once a year.

What is the difference between Standard and LTS releases? Standard Releases are recommended for new users and users currently prototyping. It offers new features, tools and support to stay current with deep learning advancements. LTS Releases are recommended for experienced users that are ready to take their application into production and who do not require new features and capabilities for their application.

For technical questions, visit the Model Optimizer FAQ and Performance Benchmarks FAQ. If you don't find an answer, please visit the following community and support links.

Get Help

- Ask on the Community Forum
- Contact Intel Support
- File an Issue on GitHub*
- Get Answers on StackOverflow*

Get Involved

- Contribute to the Code Base
- Contribute to Documentation

Stay Informed

- Join the Mailing List
- Read the Documentation
- Read the Knowledge Base
- Read the Blog