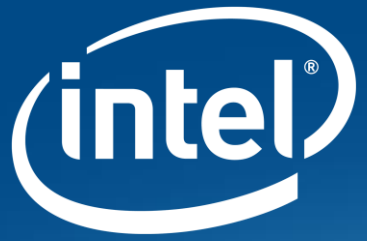


ACCELERATE DEEP LEARNING INFERENCE USING INTEL® TECHNOLOGIES

INTRODUCTION: SMART VIDEO

INTEL® DISTRIBUTION OF OPENVINO™ TOOLKIT 2020.R2 VERSION

May 2020.

AGENDA

- Intel® Smart Video/Computer vision Tools Overview
- Model Optimizer
- Inference Engine
- Multiple Models in One Application
- 15 Minute Break : approx. 1.5 hours from now....
- Accelerators based on Intel® Movidius™ Vision Processing Unit
- Accelerators based on Intel® Arria® FPGA
- DL Workbench & DL Streamer
- Register for access to Intel® DevCloud for the Edge
- Labs on Intel® DevCloud for the Edge: : 1 hour of online help

OPTIMIZATION NOTICE

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness or any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice. Notice Revision #20110804.

LEGAL NOTICES AND DISCLAIMERS (1 OF 2)

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software, or service activation. Performance varies depending on system configuration. No computer system can be absolutely secure. Check with your system manufacturer or retailer or learn more at www.intel.com.

Performance estimates were obtained prior to implementation of recent software patches and firmware updates intended to address exploits referred to as "Spectre" and "Meltdown." Implementation of these updates may make these results inapplicable to your device or system.

Cost reduction scenarios described are intended as examples of how a given Intel-based product, in the specified circumstances and configurations, may affect future costs and provide cost savings. Circumstances will vary. Intel does not guarantee any costs or cost reduction.

This document contains information on products, services, and processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.

Any forecasts of goods and services needed for Intel's operations are provided for discussion purposes only. Intel will have no liability to make any purchase in connection with forecasts published in this document.

Arduino* 101 and the Arduino infinity logo are trademarks or registered trademarks of Arduino, LLC.

Altera, Arria, the Arria logo, Intel, the Intel logo, Intel Atom, Intel Core, Intel Nervana, Intel Xeon Phi, Movidius, Saffron, and Xeon are trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

*Other names and brands may be claimed as the property of others.

LEGAL NOTICES AND DISCLAIMERS (2 OF 2)

This document contains information on products, services, and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications, and roadmaps. Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software, or service activation. Learn more at intel.com, or from the OEM or retailer.

No computer system can be absolutely secure.

Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. Consult other sources of information to evaluate performance as you consider your purchase. For more complete information about performance and benchmark results, visit www.intel.com/performance.

Cost-reduction scenarios described are intended as examples of how a given Intel-based product, in the specified circumstances and configurations, may affect future costs and provide cost savings. Circumstances will vary. Intel does not guarantee any costs or cost reduction.

Statements in this document that refer to Intel's plans and expectations for the quarter, the year, and the future are forward-looking statements that involve a number of risks and uncertainties.

A detailed discussion of the factors that could affect Intel's results and plans is included in Intel's SEC filings, including the annual report on Form 10-K.

The products described may contain design defects or errors, known as *errata*, which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Performance estimates were obtained prior to implementation of recent software patches and firmware updates intended to address exploits referred to as "Spectre" and "Meltdown." Implementation of these updates may make these results inapplicable to your device or system.

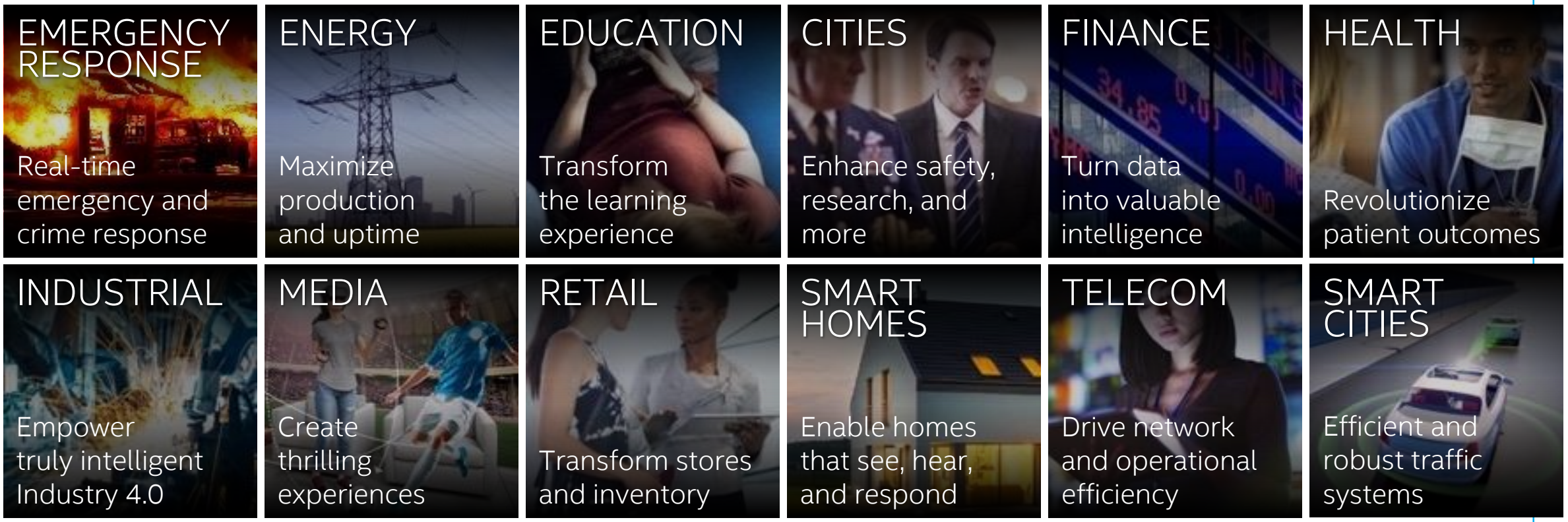
No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel does not control or audit third-party benchmark data or the web sites referenced in this document. You should visit the referenced web site and confirm whether referenced data are accurate.

Intel, the Intel logo, Pentium, Celeron, Atom, Core, Xeon, Movidius, Saffron, and others are trademarks of Intel Corporation in the United States and other countries.

*Other names and brands may be claimed as the property of others.

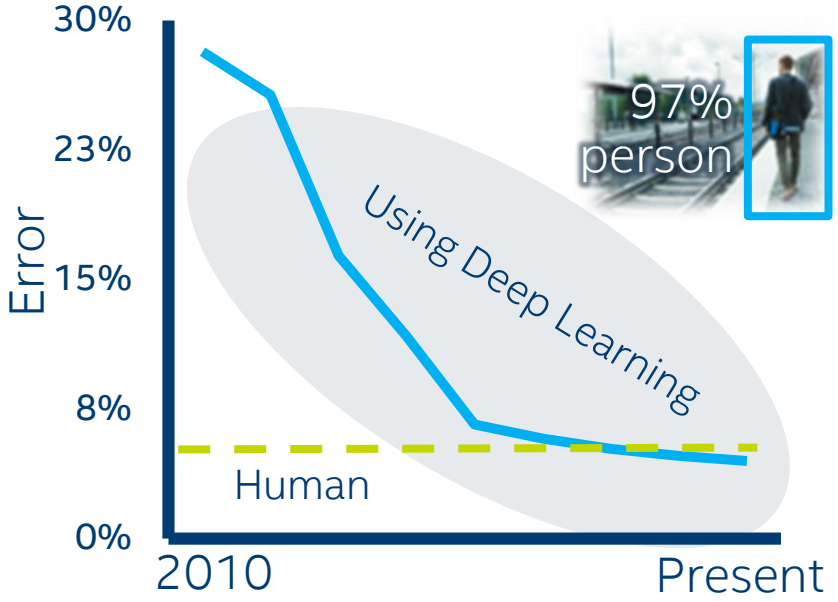
AI IS CHANGING EVERY MARKET



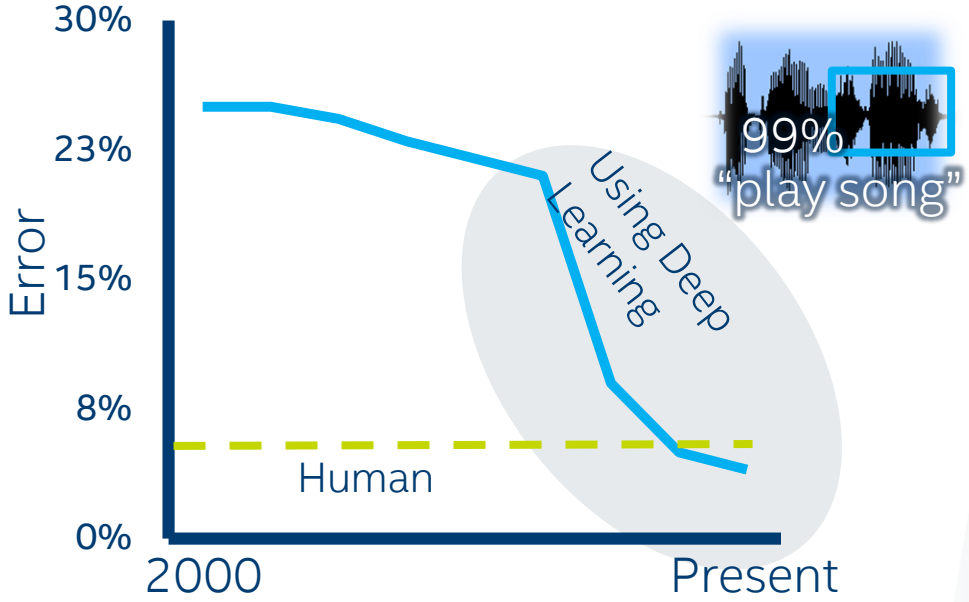
DEEP LEARNING BREAKTHROUGHS AND OPPORTUNITIES

Machines able to meet or exceed human image and speech recognition

Image Recognition



Speech Recognition



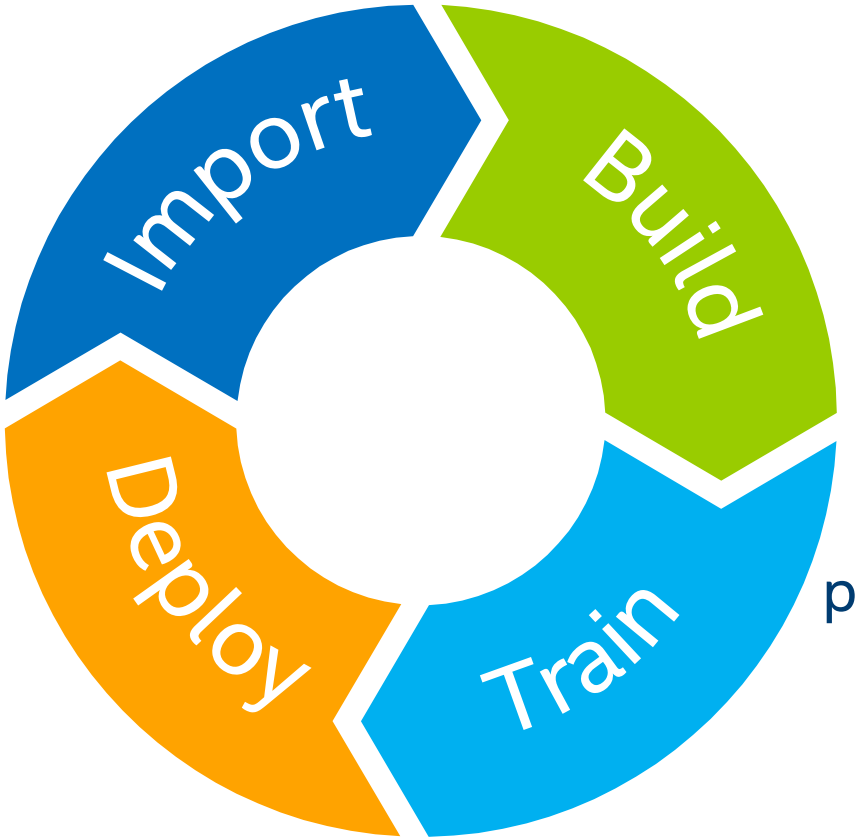
ADDITIONAL ECONOMIC IMPACT DRIVEN BY AI
$13 TRILLION IN 2030

Source: ILSVRC ImageNet winning entry classification error rate each year 2010-2016 (Left), https://www.microsoft.com/en-us/research/blog/microsoft-researchers-achieve-new-conversational-speech-recognition-milestone/ (Right)
Source: https://www.mckinsey.com/featured-insights/artificial-intelligence/notes-from-the-ai-frontier-applications-and-value-of-deep-learning

DEEP LEARNING DEVELOPMENT CYCLE

Data acquisition and organization

Create models



Integrate trained models with application code

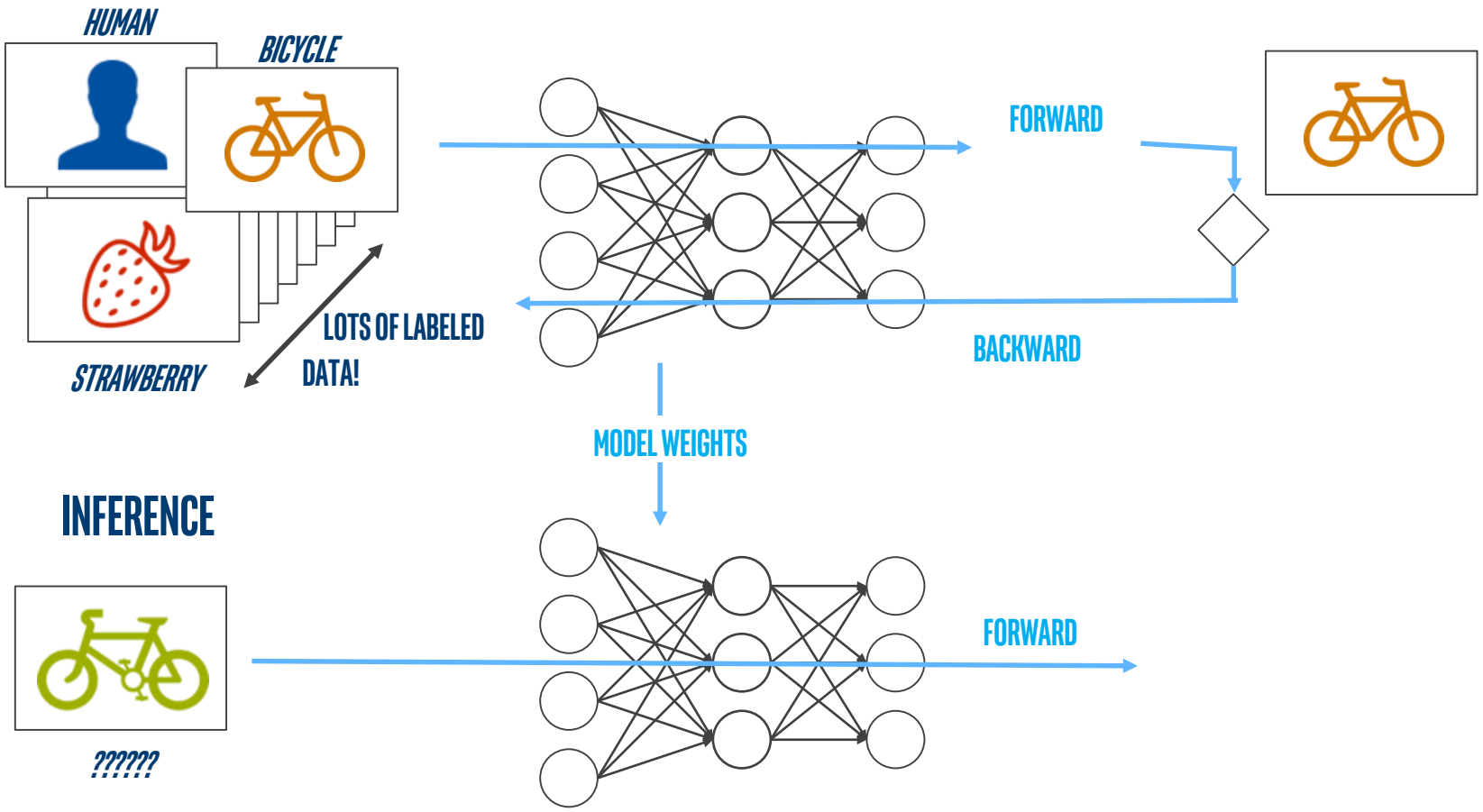
Adjust models to meet performance and accuracy objectives

Intel® Distribution OpenVINO™ Toolkit Provides Deployment from Intel® Edge to Cloud

intel

DEEP LEARNING: TRAINING VS. INFERENCE

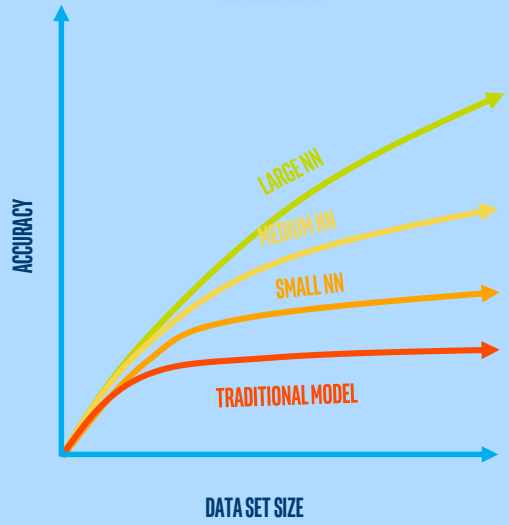
TRAINING



INFERENCE

DID YOU KNOW?

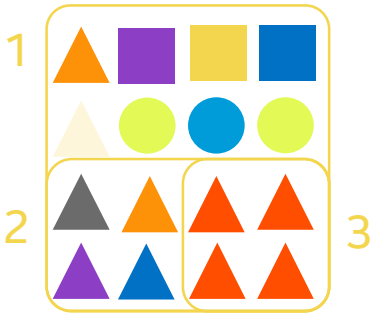
Training requires a very large data set and deep neural network (many layers) to achieve the highest accuracy in most cases



AI COMPUTE CONSIDERATIONS

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

WORKLOADS

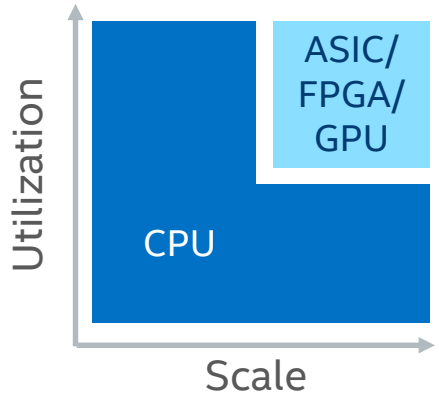


What is my workload profile?

REQUIREMENTS

What are my use case requirements?

DEMAND

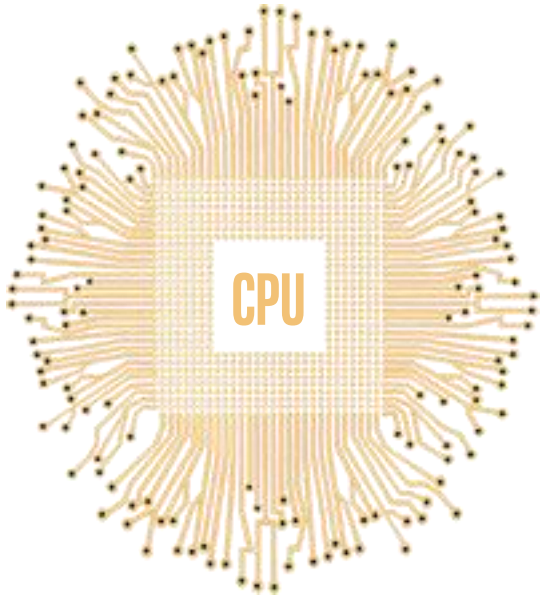


How prevalent is AI in my environment?

OpenVINO

WHY INTEL AI COMPUTE?

MAXIMIZE



Get the most out of the foundation for AI from the CPU leader

OPTIMIZE

Choose the right compute for you from the one with all the options

SIMPLIFY

OPTIMIZED SW
DATA PIPELINE
ANALYTICS & AI
SUPPORT
MOVE/STORE

Reduce “moving parts” by building on an optimized AI platform

LEAD

Lead your industry by aligning with the builder of next-gen AI solutions

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

INSIDE INTEL® DISTRIBUTION OF OPENVINO™ TOOLKIT

Deep Learning

Intel® Deep Learning Deployment Toolkit

Model Optimizer
Convert & Optimize

IR

Inference Engine
Optimized Inference
+ samples

IR = Intermediate Representation file

Open Model Zoo

- Intel & Public Pretrained Models
- Demos
- Model Downloader
- Accuracy Checker

- Deployment Manager
- Post Training Optimization Toolkit
- Benchmark App
- DL Workbench
- DL Streamer
- New

Traditional Computer Vision

- OpenCV*
- Samples

For Intel® CPU & GPU/Intel® Processor Graphics

Tools & Libraries

Increase Media/Video/Graphics Performance

- **Intel® Media SDK**
  Open Source version
- **OpenCL™ Drivers & Runtimes**

For GPU/Intel® Processor Graphics

Optimize Intel® FPGA (Linux* only)

- **FPGA RunTime Environment**
  (from Intel® FPGA SDK for OpenCL™)
- **Bitstreams**

OS Support: CentOS* 7.4 (64 bit), Ubuntu* 16.04.3 LTS (64 bit), Microsoft Windows® 10 (64 bit), Yocto Project* version Poky Jethro v2.0.3 (64 bit), macOS* 10.13 & 10.14 (64 bit)

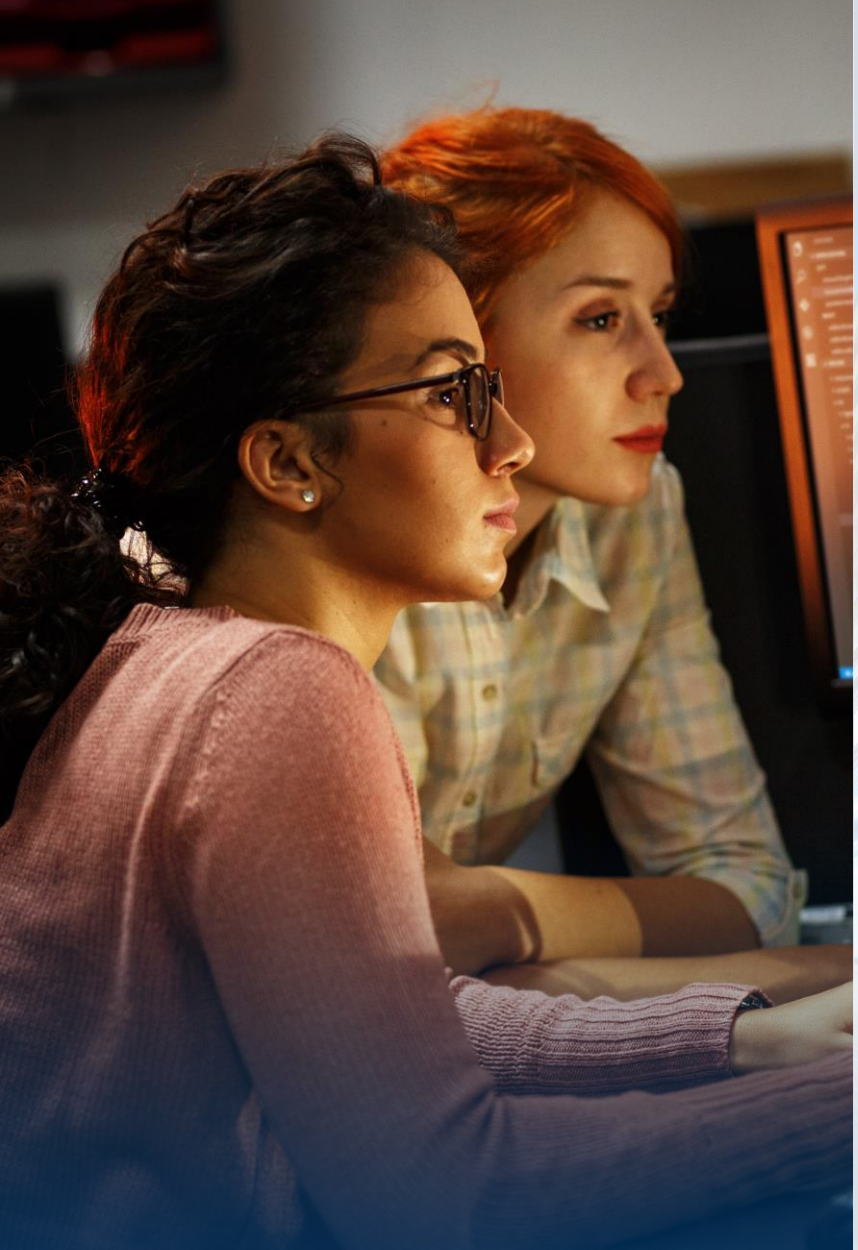
Intel® Architecture-Based Platforms Support

CELERON inside | ATOM inside | CORE inside | XEON inside | ARRIA 10 inside | MOVIDIUS inside | IRIS Pro GRAPHICS

Intel® Vision Accelerator Design Products & AI in Production/ Developer Kits

An open source version is available at 01.org/openvinotoolkit (some deep learning functions support Intel CPU/GPU only).


DEPLOY DEEP LEARNING SOLUTIONS WITH INTEL®
DISTRIBUTION OF OPENVINO™ TOOLKIT
1. BUILD
2. OPTIMIZE
3. DEPLOY



1. BUILD

2. OPTIMIZE

3. DEPLOY

BREADTH OF SUPPORTED FRAMEWORKS MAXIMIZES DEVELOPMENT

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

1. BUILD

2. OPTIMIZE

3. DEPLOY

FROM OPTIMIZATION TO DEPLOYMENT

OpenVINO

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

Development Guide
https://docs.openvinotoolkit.org/latest/_docs_MO_DG_Deep_Learning_Model_Optimizer_DevGuide.html

Inference Engine

- High-level, C/C++ and Python, inference 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

Development Guide
https://docs.openvinotoolkit.org/latest/_docs_IE_DG_Deep_Learning_Inference_Engine_DevGuide.html

TOOLS DESIGNED TO SPEED UP TEST CYCLES AND DEVELOPMENT

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

Model Analyzer

- Provides theoretical data on models: computational complexity (flops), number of neurons, memory consumption

Post-training Optimization

- Reduce model size into low precision data types, such as INT8
- Reduces model size while also improving latency

Benchmark App

- Measure performance (throughput, latency) of a model
- Get performance metrics per layer and overall basis

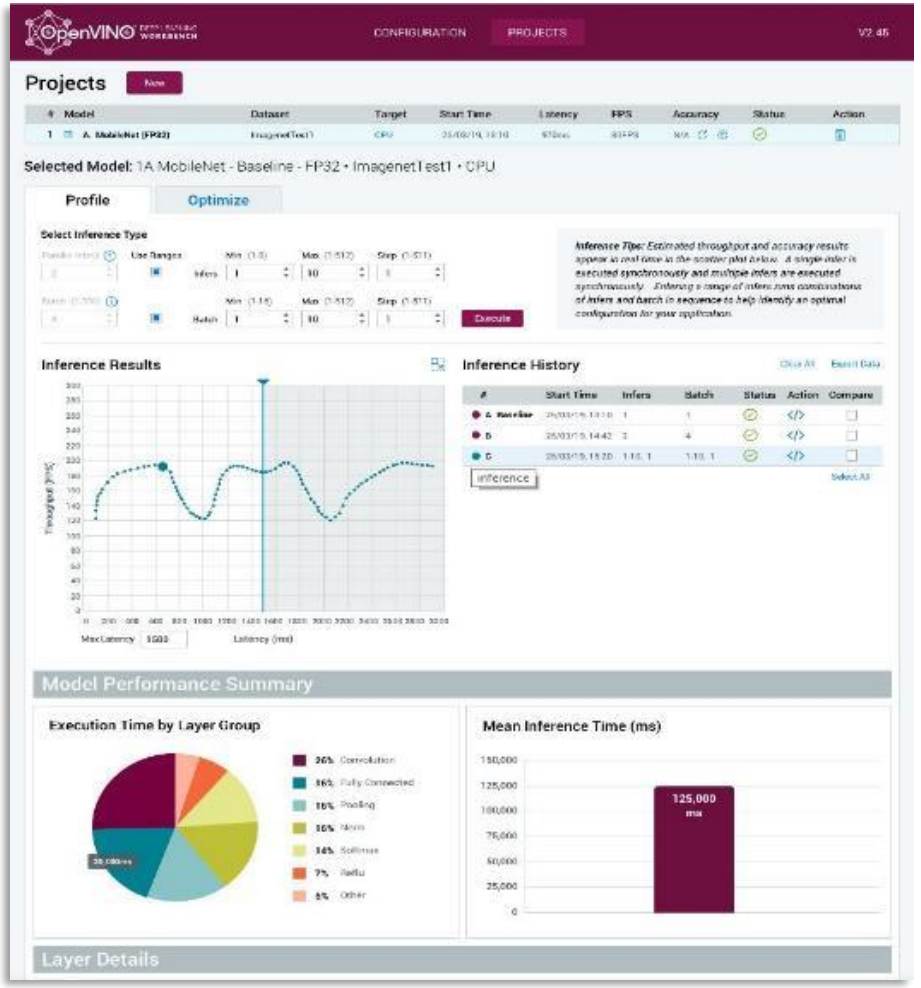
Deployment Manager

- Generate an optimal, minimized runtime package for deployment
- Deploy with smaller footprint compared to development package

Get Started ▸ https://docs.openvinotoolkit.org/latest/_docs_IE_DG_Tools_Overview.html –or- by using the Deep Learning Workbench

Deep Learning Workbench

- Web-based, UI extension tool of the Intel® Distribution of OpenVINO™ toolkit
- 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
- Provide accuracy information through accuracy checker
- Direct access to models from public set of Open Model Zoo



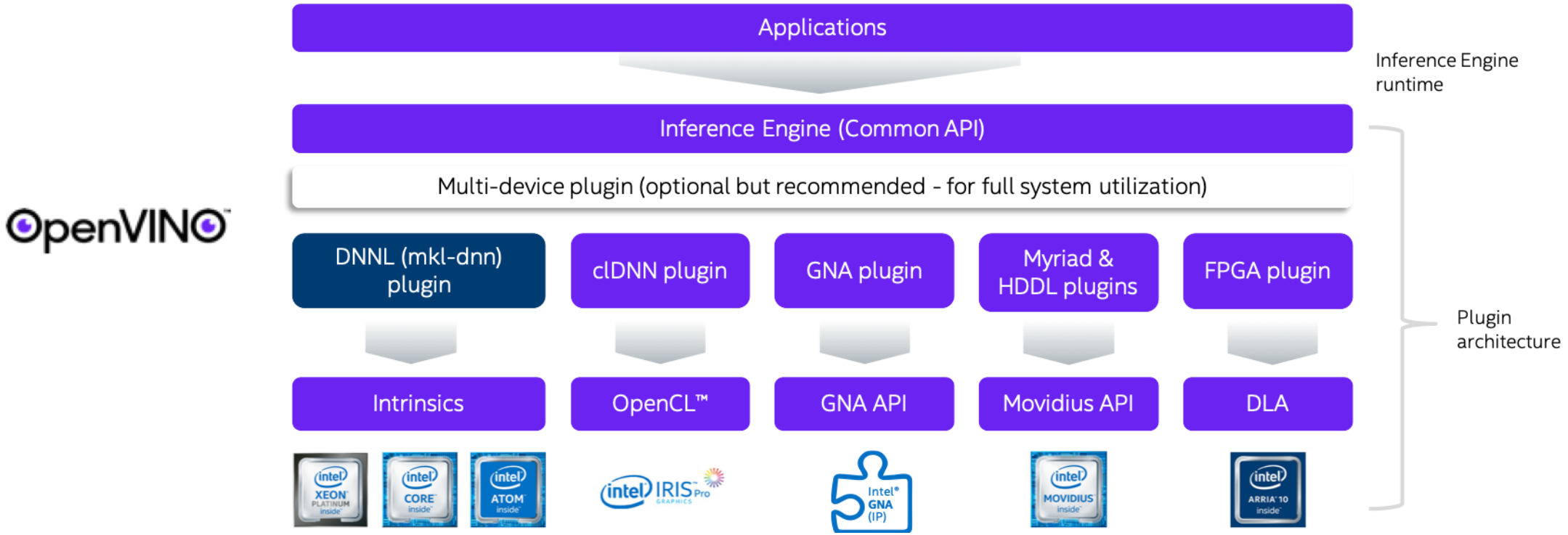
Development Guide
https://docs.openvinotoolkit.org/latest/_docs_Workbench_DG_Introduction.html

1. BUILD

2. OPTIMIZE

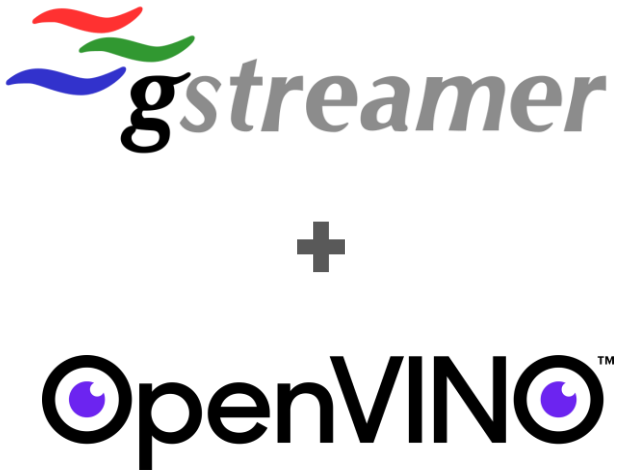
3. DEPLOY

INTEL® DISTRIBUTION OF OPENVINO™ TOOLKIT
WRITE ONCE, DEPLOY ANYWHERE



Deep Learning Streamer

- Intel® Distribution of OpenVINO™ toolkit **Deep Learning (DL) Streamer**, now part of the default installation package
- Enables developers to **create and deploy** optimized streaming media analytics pipelines across Intel® architecture from edge to cloud
- Optimal pipeline interoperability with a familiar developer experience built using the **GStreamer*** multimedia framework



Learn More ▸ https://docs.openvinotoolkit.org/latest/index.html#toolkit_components

ACCELERATE DEVELOPMENT USING THE OPEN MODEL ZOO

Open source resources with pre-trained models, samples and demos

Computer Vision

Object detection

Object recognition

Reidentification

Semantic segmentation

Instance segmentation

Human pose estimation

Image processing

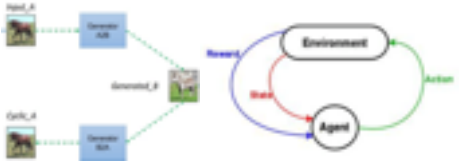
Audio, Speech, Language

Text detection

Text recognition

Recommender

Action recognition



Other

(Data Generation, Reinforcement Learning)

Compression models

Image retrieval

And more..

PRE-TRAINED MODELS

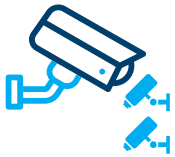
https://github.com/opencv/open_model_zoo

ACCELERATE DEVELOPMENT USING THE OPEN MODEL ZOO

Open source resources with pre-trained models, demos, and tools

The Open Model Zoo demo applications are console applications that demonstrate how you can use your applications to solve specific use-cases.

Smart Classroom
Recognition and action detection demo for classroom settings



Multi-Camera, Multi-Person
Tracking multiple people on multiple cameras for public safety use cases

Gaze Estimation
Face detection followed by gaze estimation, head pose estimation and facial landmarks regression.

Super Resolution
Enhances the resolution of the input image

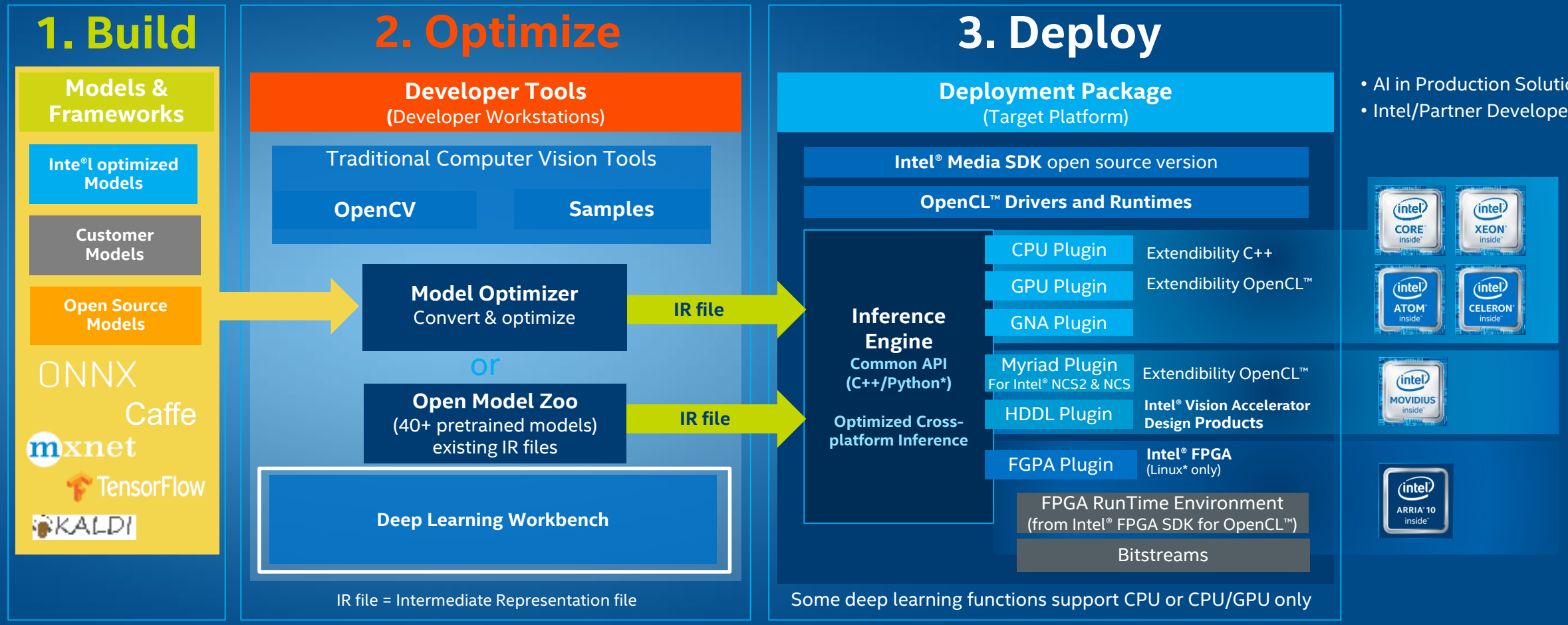
Action Recognition
Classifies actions that are being performed on input video

And more..

DEMO APPLICATIONS
https://github.com/opencv/open_model_zoo

USING THE INTEL® DISTRIBUTION OF OPENVINO™ TOOLKIT

ADVANCED CAPABILITIES TO STREAMLINE DEEP LEARNING DEPLOYMENT



Intel® NCS = Intel® Neural Compute Stick (VPU)

TEST HARDWARE WITH THE INTEL® DEVCLOUD FOR THE EDGE

Powered by Intel® Distribution of OpenVINO™ toolkit

Trained Model
Model trained using one of the supported frameworks

-or-

Using a pre-trained model available from the Open Model Zoo

Intel® Distribution of OpenVINO™ toolkit
Model Optimizer
Inference Engine

Intel® DevCloud for the Edge
A development sandbox to try AI and vision workloads remotely before purchasing Intel® platforms

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

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

Deploy and Scale

GETTING STARTED WITH INTEL® DISTRIBUTION OF OPENVINO™ TOOLKIT

Recommended Developer Flow

QUALIFY

- Use a trained model and check if framework is supported

  - *or* –

- Take advantage of a pre-trained model from the Open Model Zoo

INSTALLATION

- Download the Intel® Distribution of OpenVINO™ toolkit package from Intel® Developer Zone, or by YUM or APT repositories
- Utilize the Getting Started Guide

PREPARE

- Understand sample demos and tools included
- Understand performance
- Choose hardware option with Performance Benchmarks
- Build, test and remotely run workloads on the Intel® DevCloud for the Edge before buying hardware

HANDS ON

- Visualize metrics with the Deep Learning Workbench
- Utilize prebuilt, Reference Implementations to become familiar with capabilities
- Optimize workloads with these performance best practices
- Use the Deployment Manager to minimize deployment package

SUPPORT

- Ask questions and share information with others through the Community Forum
- Engage using #OpenVINO on Stack Overflow
- Visit documentation site for guides, how to's, and resources
- Attend training and get certified

AGENDA

- Intel® Smart Video/Computer vision Tools Overview
- Model Optimizer
- Inference Engine
- Multiple Models in One Application
- 15 Minute Break : approx. 1.5 hours from now....
- Accelerators based on Intel® Movidius™ Vision Processing Unit
- Accelerators based on Intel® Arria® FPGAs
- DL Workbench & DL Streamer
- Register for access to Intel® DevCloud for the Edge
- Labs on Intel® DevCloud for the Edge: : 1 hour of online help

MODEL OPTIMIZER

INTEL® DEEP LEARNING DEPLOYMENT TOOLKIT

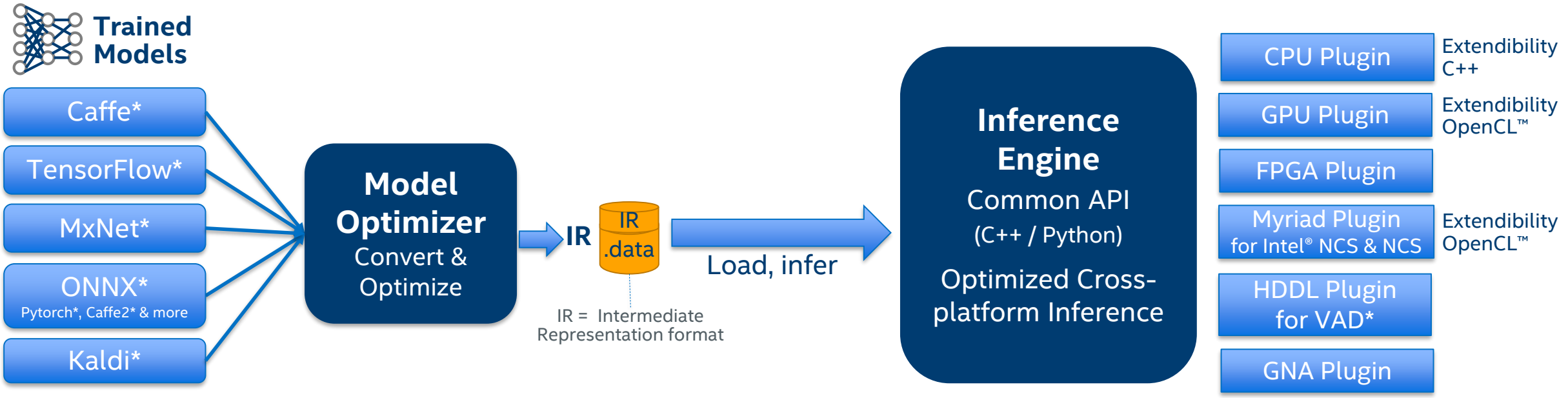
FOR DEEP LEARNING INFERENCE

Model Optimizer

- **What it is**: A Python* based tool to import trained models and convert them to Intermediate representation.
- **Why important**: Optimizes for performance/space with conservative topology transformations; biggest boost is from conversion to data types matching hardware.

Inference Engine

- **What it is**: High-level inference API
- **Why important**: Interface is implemented as dynamically loaded plugins for each hardware type. Delivers best 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

- Node merging
- Horizontal fusion
- Batch normalization to scale shift
- Fold scale shift with convolution
- Drop unused layers (dropout)

The simplest way to convert a model is to run mo.py with a path to the input model file

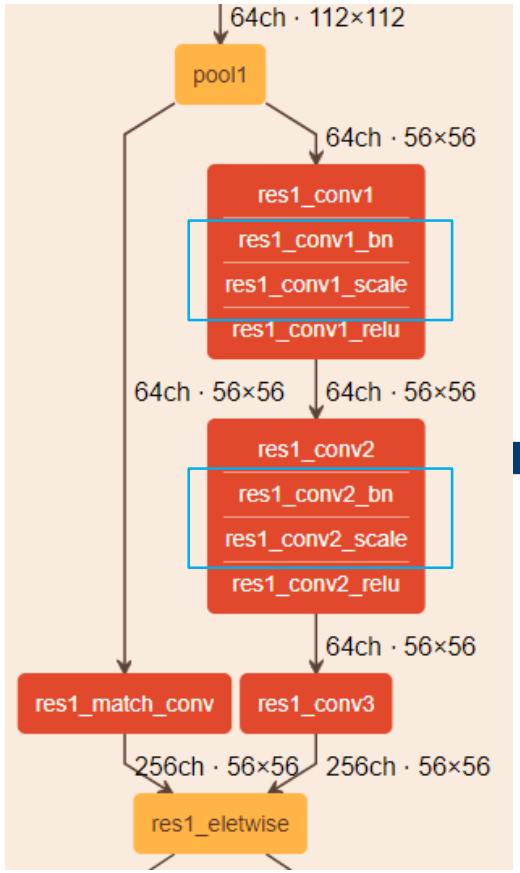
- By default, generic optimization will be automatically applied, unless manually set disable

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

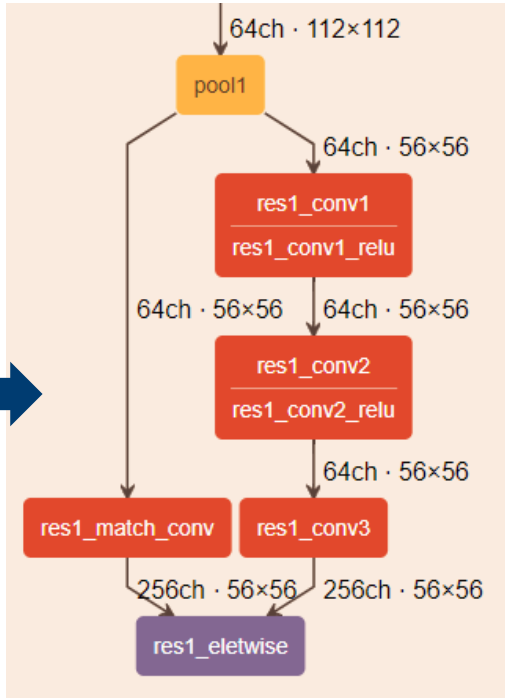
MODEL OPTIMIZATION TECHNIQUES

Linear Operation Fusing: 3 stages

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 FullybConnected layers.



Caffe* Resnet269 block (from Netscope)

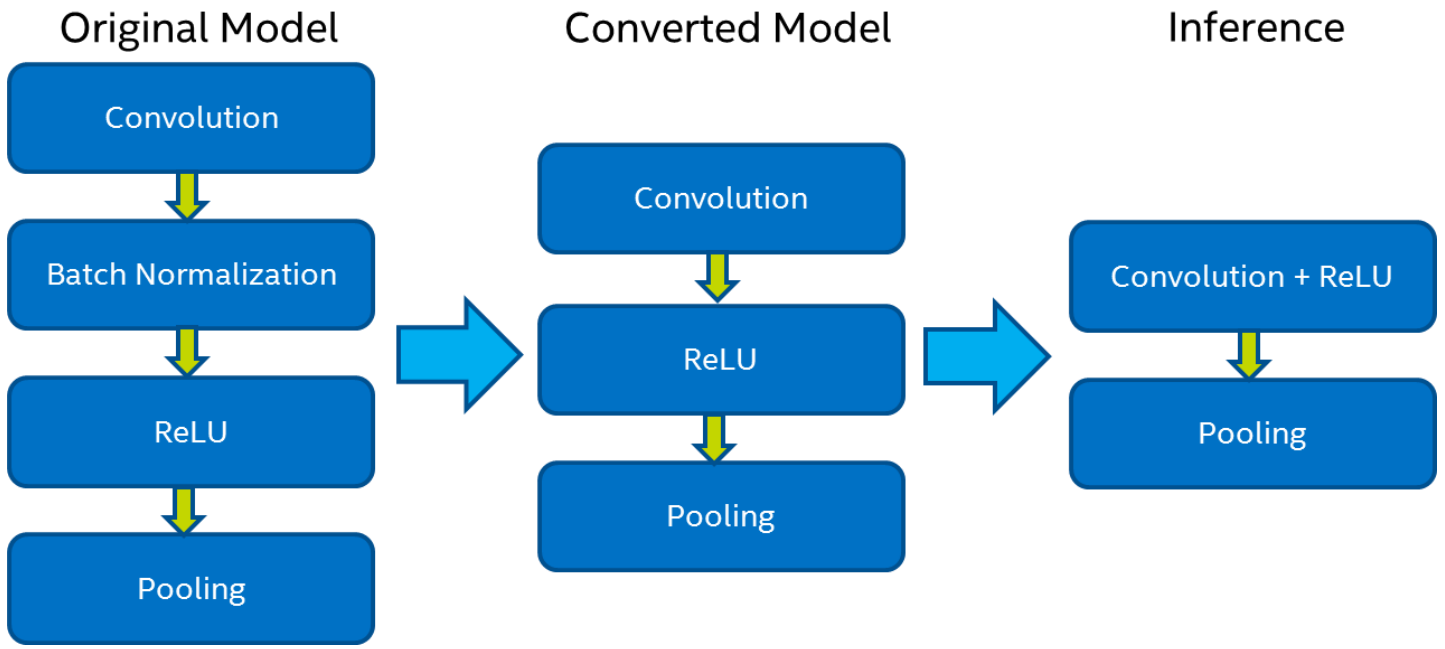


Merged Caffe* Resnet269 block (from Netscope*)

MODEL OPTIMIZER: LINEAR OPERATION FUSING

Example

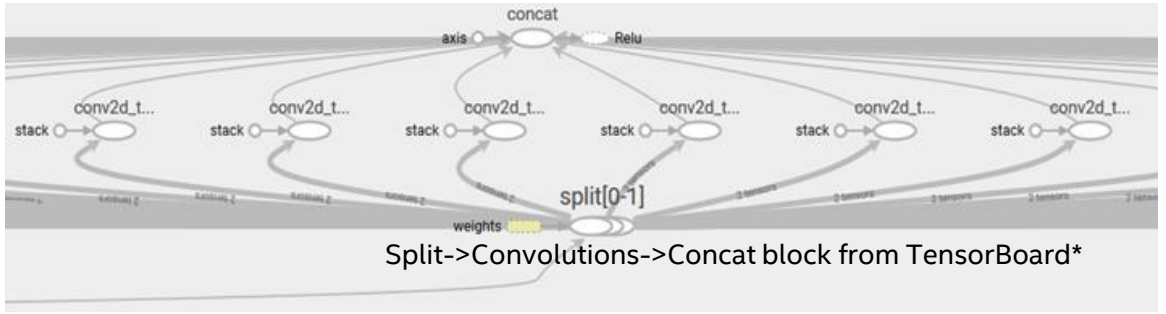
1. Remove Batch normalization stage.
2. Recalculate the weights to 'include' the operation.
3. Merge Convolution and ReLU into one optimized kernel.



MODEL OPTIMIZER: FRAMEWORK OR TOPOLOGY SPECIFIC OPTIMIZATION

Grouped Convolutions Fusing

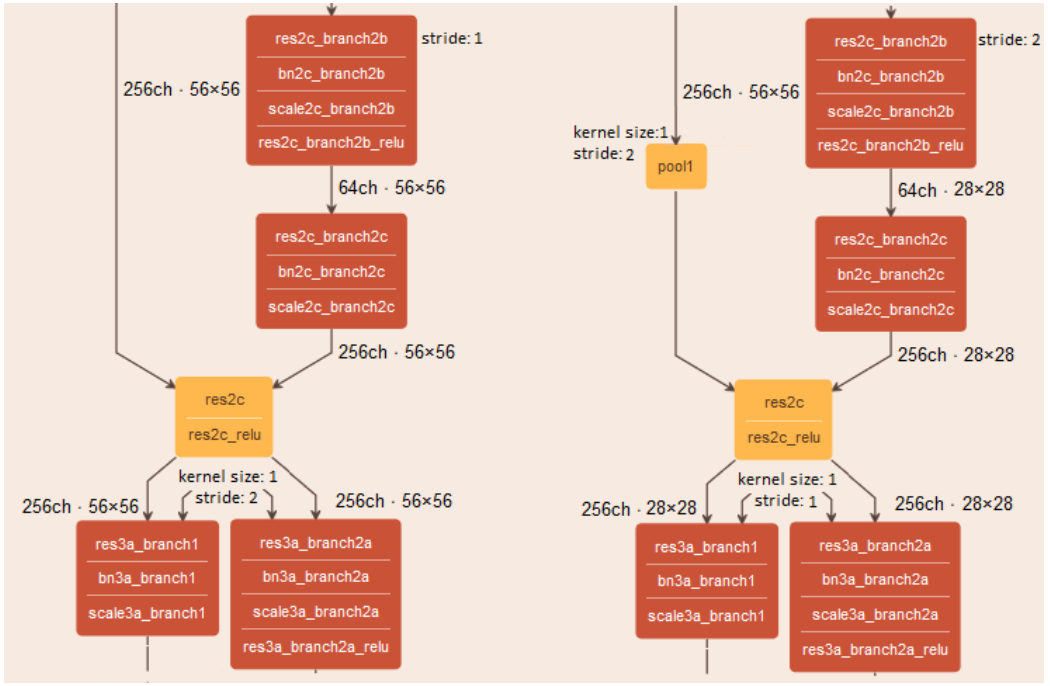
- Grouped convolution fusing is a specific optimization that applies for TensorFlow* topologies. The main idea of this optimization is to combine convolutions results for the Split outputs and then recombine them using **Concat** operation in the same order as they were out from **Split**.



Split->Convolutions->Concat block from TensorBoard*

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.



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.

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 Optimization Tool
2. FPGA also support FP11, convert happens on FPGA

```
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/
```

MODEL OPTIMIZER: OTHER CONVERSION PARAMETERS

--mean_values MEAN_VALUES, -ms MEAN_VALUES

- Mean values to be used for the input image per channel. Values to be provided in the (R,G,B) or [R,G,B] format. Can be defined for desired input of the model, for example: "--mean_value data[255,255,255],info[255,255,255]". The exact meaning and order of channels depend on how the original model was trained.

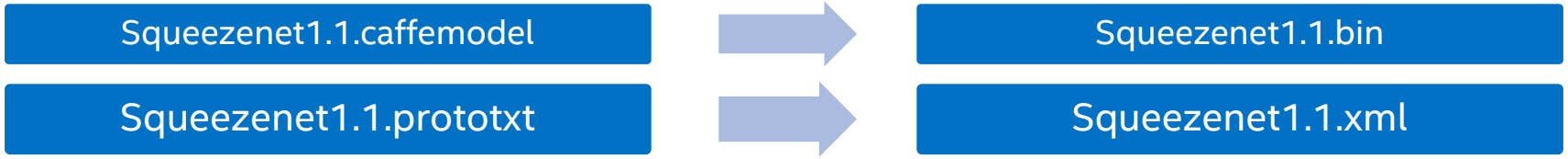
--scale_values SCALE_VALUES

- Scale values to be used for the input image per channel. Values are provided in the (R,G,B) or [R,G,B] format. Can be defined for desired input of the model, for example: "--scale_values data[255,255,255],info[255,255,255]". The exact meaning and order of channels depend on how the original model was trained.

--input_shape INPUT_SHAPE

- Model Optimizer performs necessary transformations to convert the shape to the layout required by Inference Engine (N,C,H,W). The shape should not contain undefined dimensions (? or -1) and should fit the dimensions defined in the input operation of the graph

INTERMEDIATE REPRESENTATION (IR)




```
layer {
  name: "data"
  type: "Input"
  top: "data"
  input_param { shape: { dim: 1 dim: 3 dim: 227 dim: 227 } }
}
layer {
  name: "conv1"
  type: "Convolution"
  bottom: "data"
  top: "conv1"
  convolution_param {
    num_output: 64
    kernel_size: 3
    stride: 2
  }
}
```

```
<net batch="1" name="model" version="2">
  <layers>
    <layer id="100" name="data" precision="FP32" type="Input">
      <output>
        <port id="0">
          <dim>1</dim>
          <dim>3</dim>
          <dim>227</dim>
          <dim>227</dim>
        </port>
      </output>
    </layer>
    <layer id="129" name="conv1" precision="FP32" type="Convolution">
      <data dilation-x="1" dilation-y="1" group="1" kernel-x="3" kernel-y="3" output="64" pa
      <input>
        <port id="0">
          <dim>1</dim>
          <dim>3</dim>
          <dim>227</dim>
          <dim>227</dim>
        </port>
      </input>
      <output>
        <port id="3">
          <dim>1</dim>
          <dim>64</dim>
          <dim>113</dim>
          <dim>113</dim>
        </port>
      </output>
      <blobs>
        <weights offset="2275104" size="6912"/>
        <biases offset="4805920" size="256"/>
      </blobs>
```

OPEN MODEL ZOO & MODEL DOWNLOADER DEMO

Open Model Zoo

- https://github.com/opencv/open_model_zoo
- **/opt/intel/openvino/deployment_tools/open_model_zoo**

```
open_model_zoo
├── demos
│   ├── common
│   ├── crossroad_camera_demo
│   ├── gaze_estimation_demo
│   ├── human_pose_estimation_demo
│   ├── interactive_face_detection_demo
│   ├── mask_rcnn_demo
│   ├── multi_channel
│   ├── object_detection_demo_faster_rcnn
│   ├── object_detection_demo_ssd_async
│   ├── object_detection_demo_yolov3_async
│   ├── pedestrian_tracker_demo
│   ├── python_demos
│   ├── security_barrier_camera_demo
│   ├── segmentation_demo
│   ├── smart_classroom_demo
│   ├── super_resolution_demo
│   ├── text_detection_demo
│   └── thirdparty
├── intel_models -> models/intel
├── models
│   ├── intel
│   └── public
└── tools
    ├── accuracy_checker
    └── downloader
```

Model Downloader Parameters:

```
shane@shane-VirtualBox:~$ python3 /opt/intel/openvino/deployment_tools/open_mo
del_zoo/tools/downloader/downloader.py --help
usage: downloader.py [-h] [--name PAT[,PAT...]] [--list FILE.LST] [--all]
                     [--print_all] [--precisions PREC[,PREC...]] [-o DIR]
                     [--cache_dir DIR] [--num_attempts N]
                     [--progress_format {text,json}]

optional arguments:
  -h, --help            show this help message and exit
  --name PAT[,PAT...]   download only models whose names match at least one of
                        the specified patterns
  --list FILE.LST       download only models whose names match at least one of
                        the patterns in the specified file
  --all                 download all available models
  --print_all           print all available models
  --precisions PREC[,PREC...]
                        download only models with the specified precisions
                        (actual for DLDT networks)
  -o DIR, --output_dir DIR
                        path where to save models
  --cache_dir DIR       directory to use as a cache for downloaded files
  --num_attempts N      attempt each download up to N times
  --progress_format {text,json}
                        which format to use for progress reporting
```

Download a public model: mobilenet-ssd

```
shane@shane-VirtualBox:~$ python3 /opt/intel/openvino/deployment_tools/open_mo
del_zoo/tools/downloader/downloader.py --name mobilenet-ssd
################|| Downloading models ||################

========== Downloading /home/shane/public/mobilenet-ssd/mobilenet-ssd.prototxt
... 100%, 28 KB, 74123 KB/s, 0 seconds passed

========== Downloading /home/shane/public/mobilenet-ssd/mobilenet-ssd.caffemod
el
... 100%, 22605 KB, 47724 KB/s, 0 seconds passed

################|| Post-processing ||################
```

MODEL OPTIMIZER DEMO

Model Optimizer:

- **/opt/intel/openvino/deployment_tools/model_optimizer/mo.py**

FP32 vs FP16:

```
shane@shane-VirtualBox:~/Desktop$ tree -h
.
└── [4.0K]  mobilenet
    ├── [4.0K]  FP16
    │   ├── [ 11M]  mobilenet-ssd.bin
    │   ├── [ 14K]  mobilenet-ssd.mapping
    │   └── [171K]  mobilenet-ssd.xml
    └── [4.0K]  FP32
        ├── [ 22M]  mobilenet-ssd.bin
        ├── [ 14K]  mobilenet-ssd.mapping
        └── [171K]  mobilenet-ssd.xml
```

Are we using the right parameters for converting a certain public model?

- Check the .yml file under:
- **/opt/intel/openvino/deployment_tools/open_model_zoo/model/puclic/#MODEL NAME#**

```
model_optimizer_args:
  - --input_shape=[1,3,300,300]
  - --input=data
  - --mean_values=data[127.5,127.5,127.5]
  - --scale_values=data[127.5]
```

```
shane@shane-VirtualBox:~/Desktop$ python3 /opt/intel/openvino/deployment_tools/model_optimizer/mo.py
--input_model /home/shane/public/mobilenet-ssd/mobilenet-ssd.caffemodel -o /home/shane/Desktop/mobile
net/FP32 --data_type FP32
Model Optimizer arguments:
Common parameters:
        - Path to the Input Model:      /home/shane/public/mobilenet-ssd/mobilenet-ssd.caffemodel
        - Path for generated IR:        /home/shane/Desktop/mobilenet/FP32
        - IR output name:       mobilenet-ssd
        - Log level:    ERROR
        - Batch:        Not specified, inherited from the model
        - Input layers:         Not specified, inherited from the model
        - Output layers:        Not specified, inherited from the model
        - Input shapes:         Not specified, inherited from the model
        - Mean values:  Not specified
        - Scale values:         Not specified
        - Scale factor:         Not specified
        - Precision of IR:      FP32
        - Enable fusing:        True
        - Enable grouped convolutions fusing:   True
        - Move mean values to preprocess section:       False
        - Reverse input channels:       False
Caffe specific parameters:
        - Path to Python Caffe* parser generated from caffe.proto:      /opt/intel/openvino/deploymen
t_tools/model_optimizer/mo/front/caffe/proto
        - Enable resnet optimization:   True
        - Path to the Input prototxt:   /home/shane/public/mobilenet-ssd/mobilenet-ssd.prototxt
        - Path to CustomLayersMapping.xml:      Default
        - Path to a mean file:  Not specified
        - Offsets for a mean file:      Not specified
Model Optimizer version:        2020.2.0-60-g0bc66e26ff
[ WARNING ]
Detected not satisfied dependencies:
        protobuf: installed: 3.0.0, required: == 3.6.1

Please install required versions of components or use install_prerequisites script
/opt/intel/openvino_2020.2.120/deployment_tools/model_optimizer/install_prerequisites/install_prerequ
isites_caffe.sh
Note that install_prerequisites scripts may install additional components.

[ SUCCESS ] Generated IR version 10 model.
[ SUCCESS ] XML file: /home/shane/Desktop/mobilenet/FP32/mobilenet-ssd.xml
[ SUCCESS ] BIN file: /home/shane/Desktop/mobilenet/FP32/mobilenet-ssd.bin
[ SUCCESS ] Total execution time: 21.13 seconds.
[ SUCCESS ] Memory consumed: 175 MB.
```

AGENDA

- Intel® Smart Video/Computer vision Tools Overview
- Model Optimizer
- Inference Engine
- Multiple Models in One Application
- 15 Minute Break : approx. 1.5 hours from now....
- Accelerators based on Intel® Movidius™ Vision Processing Unit
- Accelerators based on Intel® Arria® FPGAs
- DL Workbench & DL Streamer
- Register for access to Intel® DevCloud for Edge
- Labs on Intel® DevCloud for Edge: : 1 hour of online help

INFERENCE ENGINE

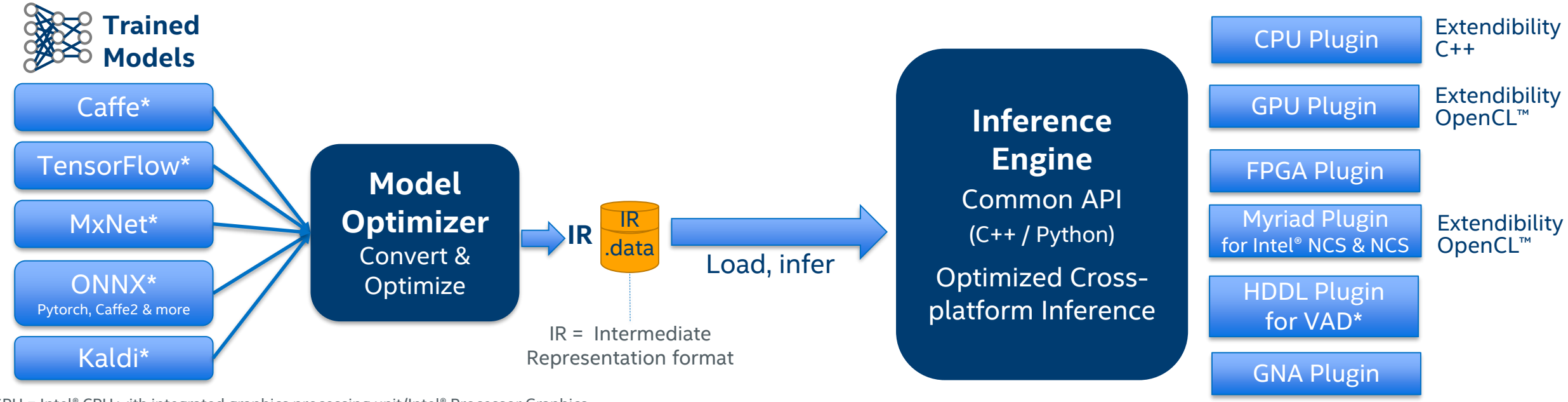
INTEL® DEEP LEARNING DEPLOYMENT TOOLKIT
FOR DEEP LEARNING INFERENCE

Model Optimizer

- **What it is**: A Python* based tool to import trained models and convert them to Intermediate representation.
- **Why important**: Optimizes for performance/space with conservative topology transformations; biggest boost is from conversion to data types matching hardware.

Inference Engine

- **What it is**: High-level inference API
- **Why important**: Interface is implemented as dynamically loaded plugins for each hardware type. Delivers best performance for each type without requiring users to implement and maintain multiple code pathways.



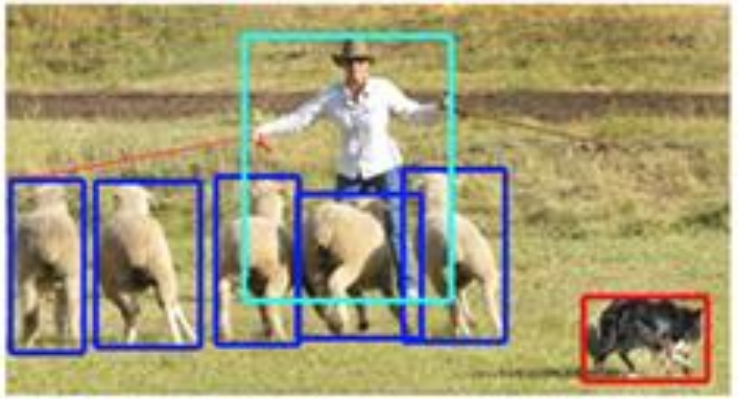
GPU = Intel® CPU with integrated graphics processing unit/Intel® Processor Graphics

*VAD = Vision Accelerator Design Products (HDDL-R)

INFERENCE ON AN INTEL® EDGE SYSTEM

Many deep learning networks are available—choose the one you need.

(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.

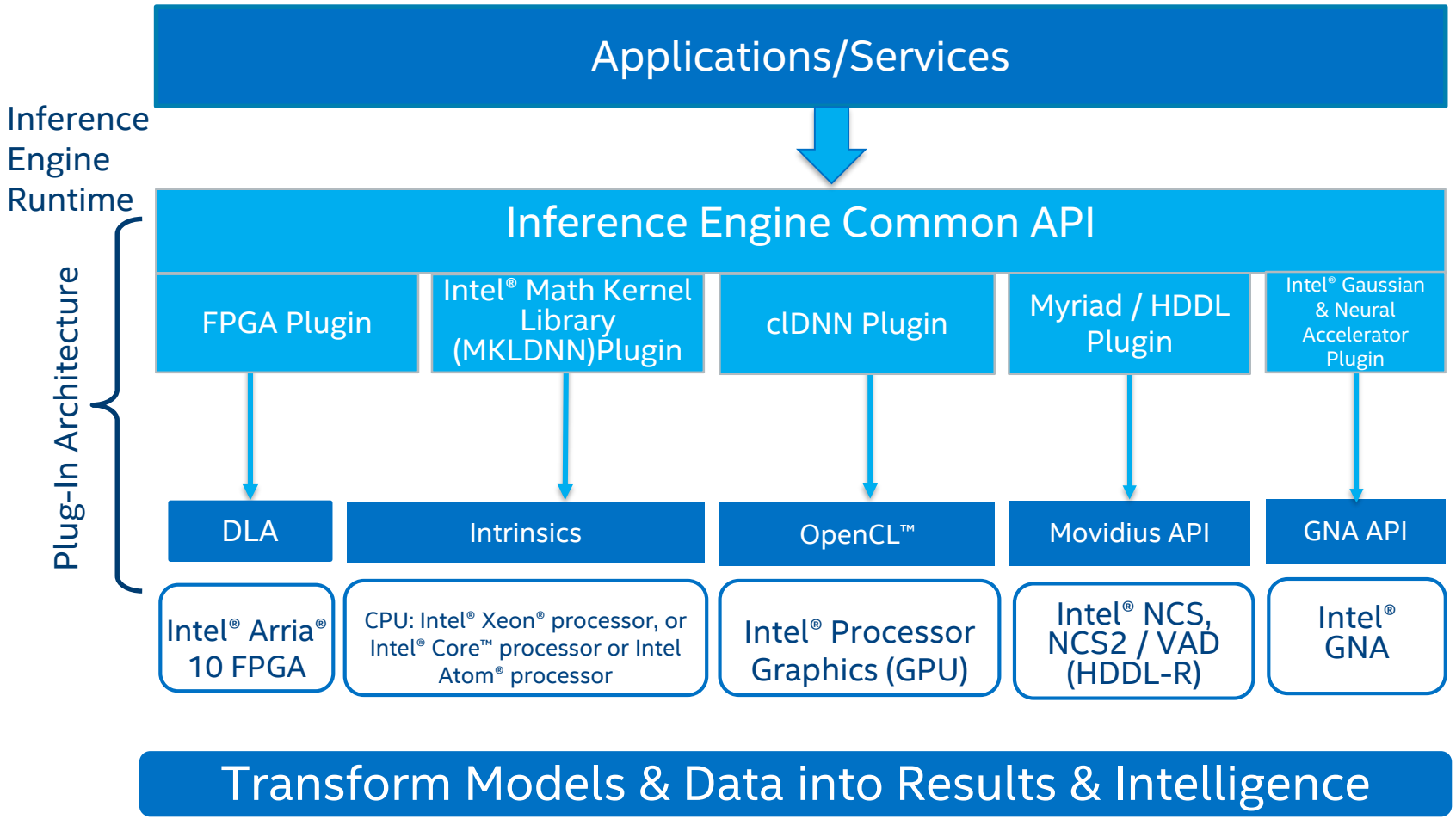
OPTIMAL MODEL PERFORMANCE USING THE INFERENCE ENGINE

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

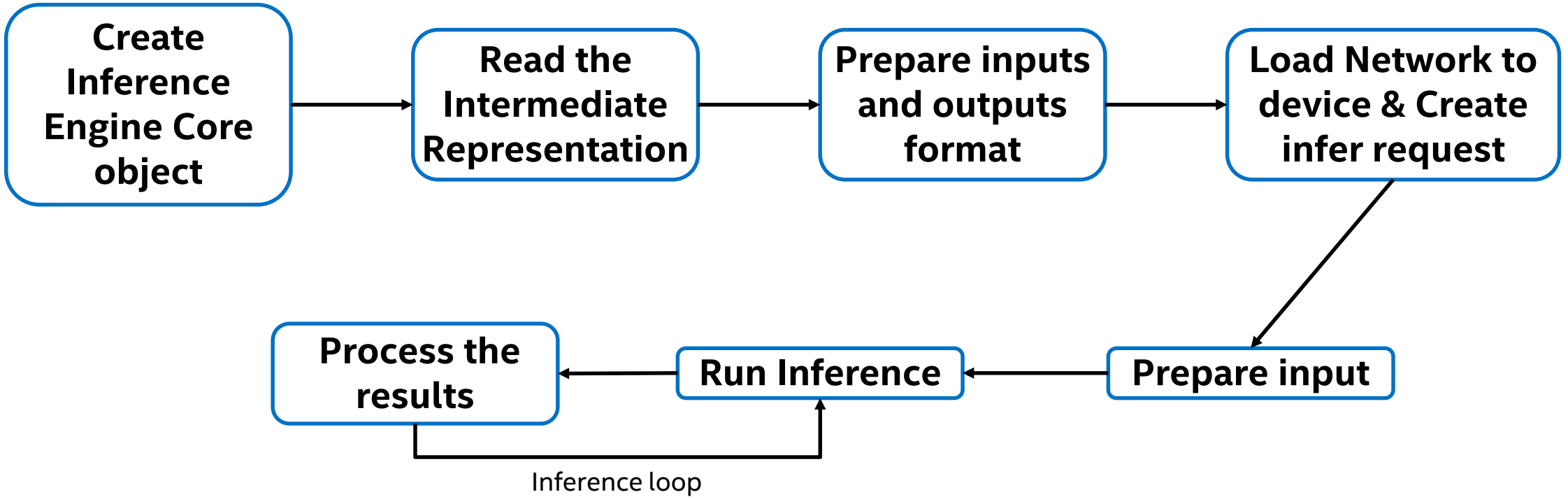
Device-specific Plugin Libraries

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



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

USE THE INFERENCE ENGINE API IN YOUR CODE PART.1

OBJECT DETECTION SSD EXAMPLE

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.inputs))**
- **output_blob = next(iter(net.outputs))**

USE THE INFERENCE ENGINE API IN YOUR CODE PART.2

OBJECT DETECTION SSD EXAMPLE

Load Network to device & Create infer request

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

Prepare input

1. Read the input's dimensions: n=batch size, c=number of channels, h=height, w=width:
   n, c, h, w = net.inputs[input_blob].shape
2. Resize image dimensions form image to model's input w x h:
   in_frame = cv2.resize(image, (w, h))
3. Change data layout from HWC to CHW:
   in_frame = in_frame.transpose((2, 0, 1))
4. Reshape to input dimensions:
   in_frame = in_frame.reshape((n, c, h, w))

Run Inference

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

USE THE INFERENCE ENGINE API IN YOUR CODE PART.3

OBJECT DETECTION SSD EXAMPLE

Process the results (Post-processing)

The array of detection summary info, name - `detection_out`, shape - `1, 1, N, 7`, where N is the number of detected bounding boxes. For each detection, the description has the format: [`image_id`, `label`, `conf`, `x_min`, `y_min`, `x_max`, `y_max`], where:

- `image_id` - ID of the image in the batch
- `label` - predicted class ID
- `conf` - confidence for the predicted class
- (`x_min`, `y_min`) - coordinates of the top left bounding box corner (coordinates are in normalized format, in range [0, 1])
- (`x_max`, `y_max`) - coordinates of the bottom right bounding box corner (coordinates are in normalized format, in range [0, 1])

```
res = res[out_blob]
boxes, classes = {}, {}
data = res[0][0]
for number, proposal in enumerate(data):
    if proposal[2] > 0:
        imid = np.int(proposal[0])
        ih, iw = images_hw[imid]
        label = np.int(proposal[1])
        confidence = proposal[2]
        xmin = np.int(iw * proposal[3])
        ymin = np.int(ih * proposal[4])
        xmax = np.int(iw * proposal[5])
        ymax = np.int(ih * proposal[6])
        print("[{},{}] element, prob = {:.6}    ({},{})-({},{}) batch
        id : {}".format(number, label, confidence, xmin, ymin, xmax,
        ymax, imid), end="")
        if proposal[2] > 0.5:
            print(" WILL BE PRINTED!")
            if not imid in boxes.keys():
                boxes[imid] = []
            boxes[imid].append([xmin, ymin, xmax, ymax])
            if not imid in classes.keys():
                classes[imid] = []
            classes[imid].append(label)
    else:
        print()

for imid in classes:
    tmp_image = cv2.imread(args.input[imid])
    for box in boxes[imid]:
        cv2.rectangle(tmp_image, (box[0], box[1]), (box[2], box[3]), (
        232, 35, 244), 2)
    cv2.imwrite("out.bmp", tmp_image)
    log.info("Image out.bmp created!")
```

INFERENCE ENGINE ASYNC API

- Improves overall frame-rate of the application
- Executes a request asynchronously (in the background) and waits until ready, when the result is actually needed.
- Continues doing things on the host, while the accelerator is busy.
- While the current is processed, the input frame for the next is captured. This essentially hides the latency of capturing, so the overall framerate is determined by the MAXIMUM(detection time, input capturing time) and not the SUM(detection time, input capturing time).

```
# Creating the ExecutableNetwork with two request "slots"
exec_net = ie.load_network(network=net, device_name="CPU", num_requests=2)
output_layer = next(iter(net.outputs))

# Submitting two asynchronous Inference.
infer_request_0 = exec_net.start_async(request_id=0, inputs={input_layer: input_image_0})
infer_request_1 = exec_net.start_async(request_id=1, inputs={input_layer: input_image_1})

# Wait until inference 0 is done, then process
infer_request_0.wait()
postProcess(infer_request_0.outputs[output_layer])

# Wait until inference 1 is done, then process
exec_net.requests[1].wait()
postProcess(exec_net.requests[1].outputs[output_layer])
```

INFERENCE ENGINE: HETERO PLUGIN

ENABLES COMPUTING FOR INFERENCE ON ONE NETWORK ON SEVERAL DEVICES

Use cases for heterogeneous mode

- To utilize accelerators power and calculate heaviest parts of network on accelerator and execute not supported layers on fallback devices like CPU
- To utilize all available hardware more efficiently during one inference

Two independent decoupled steps.

1. Setting of affinity to layers

2. Loading a network to the Heterogeneous plugin, splitting the network to parts, and executing them through the plugin

APPLY DEVICE AFFINITIES TO LAYERS AUTOMATICALLY USING FALLBACK POLICY

```
1  InferenceEngine::CNNNetReader reader;
2  reader.ReadNetwork("Model.xml");
3  reader.ReadWeights("Model.bin");
4
5  InferenceEngine::Core core;
6  auto executable_network = core.LoadNetwork(reader.getNetwork(), "HETERO:FPGA,CPU");
```

- The fallback automatic policy means greedy behavior and assigns all layers which can be executed on certain device on that device follow priorities.
- This example shows FPGA,CPU points to fallback policy with first priority on FPGA and further fallback to CPU.
- You can point more than two devices: HETERO:FPGA,GPU,CPU.

APPLY DEVICE AFFINITIES TO LAYERS MANUALLY

Another way to annotate a network is setting affinity manually using CNNLayer::affinity field. This field accepts string values of devices like "CPU" or "FPGA".

```
ie = IECore()
net = ie.read_network(model=path_to_xml_file, weights=path_to_bin_file)
layers_map = ie.query_network(network=net, device_name="HETERO:GPU,CPU")
layers = net.layers
for layer, device in layers_map.items():
    layers[layer].affinity = device
```

INFERENCE ENGINE: MULTI PLUGIN
INFERENCE ON AVAILABLE DEVICES IN PARALLEL

- Improved throughput that multiple devices can deliver (compared to single-device execution)
- More consistent performance, since the devices can now share the inference burden (so that if one device is becoming too busy, another device can take more of the load)
- Don't need to explicitly load the network to every device, create and balance the inference requests
- C++ example (Python* is similar)

```
Core ie;
ExecutableNetwork exec = ie.LoadNetwork(network, "MULTI:HDDL,GPU", {});
```

DEMO – OBJECT DETECTION EXAMPLE WITH INFERENCE ENGINE(IE)

- Run the Object Detection example with previously converted IR format mobilenet-ssd model

AGENDA

- Intel® Smart Video/Computer vision Tools Overview
- Model Optimizer
- Inference Engine
- Multiple Models in One Application
- 15 Minute Break : approx. 1.5 hours from now....
- Accelerators based on Intel® Movidius™ Vision Processing Unit
- Accelerators based on Intel® Arria® FPGA
- DL Workbench & DL Streamer
- Register for access to Intel® DevCloud for Edge
- Labs on Intel® DevCloud for Edge: : 1 hour of online help

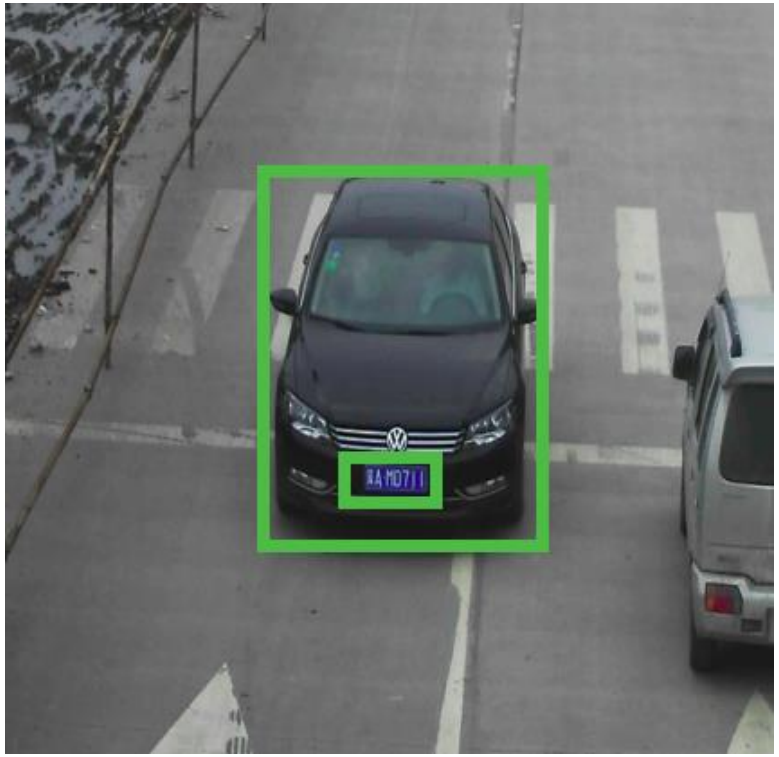
MULTIPLE MODELS IN ONE APPLICATION SECURITY BARRIER DEMO

VIDEO ANALYTICS IN INTEL® DISTRIBUTION OF OPENVINO™ TOOLKIT

Topology	Type	Description
vehicle-license-plate-detection-barrier-0007	detection	Multiclass (vehicle, license plates) detector based on RESNET* 10 plus SSD.
vehicle-attributes-recognition-barrier-0010	object_attributes	Vehicle attributes recognition with modified RESNET 10 backbone.
license-plate-recognition-barrier-0001	ocr	Chinese license plate recognition.

VEHICLE-LICENSE-PLATE-DETECTION-BARRIER-007
USE CASE/HIGH-LEVEL DESCRIPTION

RESNET* 10 plus SSD-based vehicle and (Chinese) license plate detector for "Barrier" use case.



VEHICLE-ATTRIBUTES-RECOGNITION-BARRIER-0010
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

DEMO –MULTIPLE MODELS EXAMPLE

- Run Security Barrier Demo to show three models cascaded in one application

15 MINS BREAK

Survey: https://bit.ly/VINOsurvey

Download the Intel® Distribution of OpenVINO(TM) toolkit
https://software.intel.com/content/www/us/en/develop/tools/openvino-toolkit/choose-download.html

Intel® Edge Software Hub – Edge Computing Software and Packages
https://www.intel.com/content/www/us/en/edge-computing/edge-software-hub.html

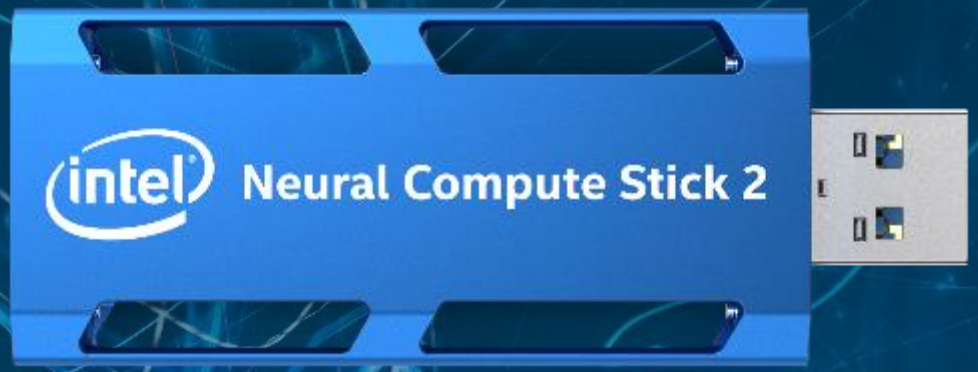
Schedule for the Intel® Distribution of OpenVINO™ Toolkit Virtual Workshops
https://software.seek.intel.com/OpenVINOworkshops

Go to Market with the Intel® Distribution of OpenVINO™ Toolkit
https://software.intel.com/content/www/us/en/develop/topics/iot/training/go-to-market-with-openvino.html

AGENDA

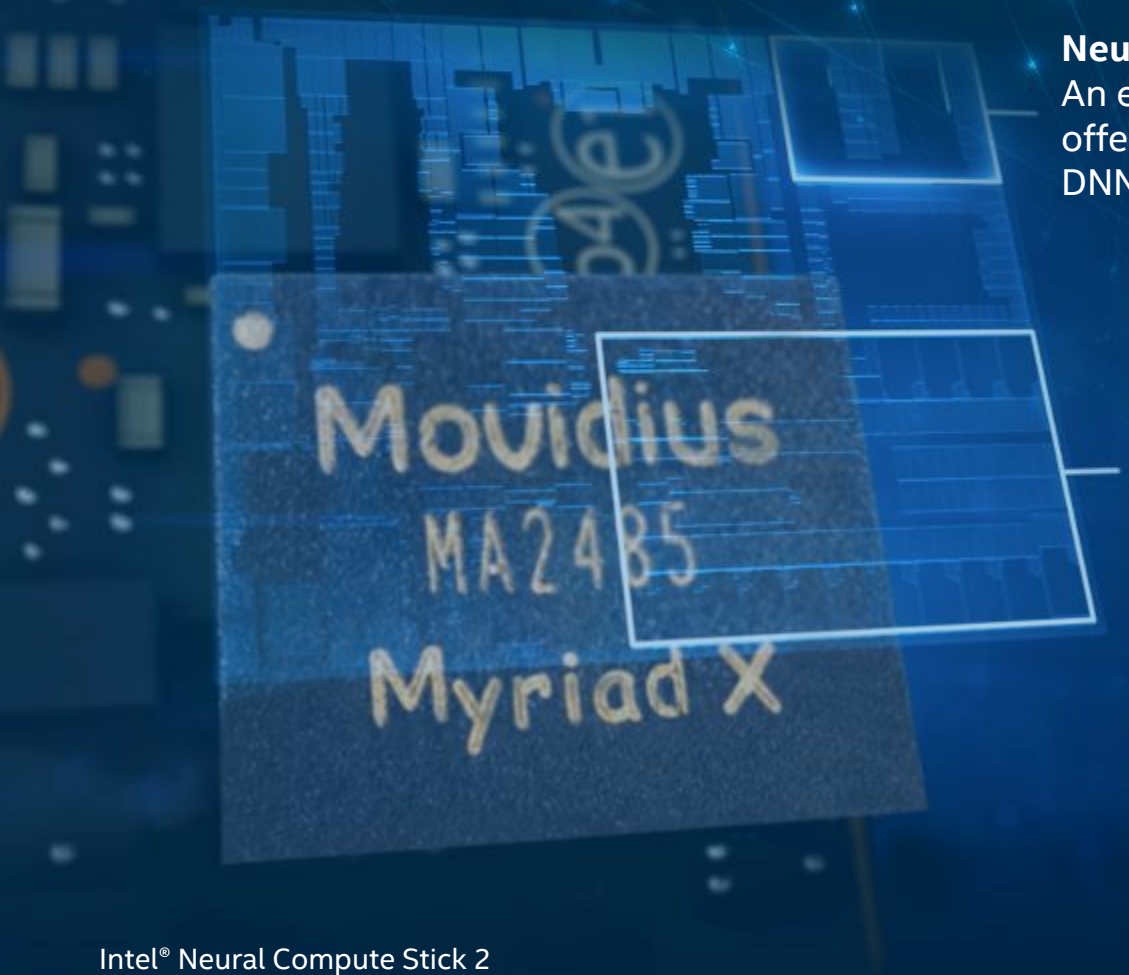
- Intel® Smart Video/Computer vision Tools Overview
- Model Optimizer
- Inference Engine
- Multiple Models in One Application
- 15 Minute Break : approx. 1.5 hours from now....
- Accelerators based on Intel® Movidius™ Vision Processing Unit
- Accelerators based on Intel® Arria® FPGA
- DL Workbench & DL Streamer
- Register for access to Intel® DevCloud for Edge
- Labs on Intel® DevCloud for Edge: : 1 hour of online help

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

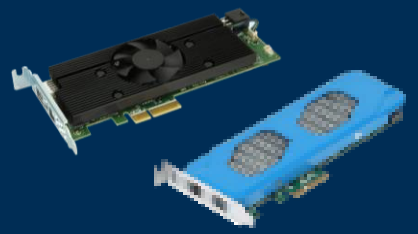


Neural Compute Engine
An entirely new deep neural network (DNN) inferencing engine that offers flexible interconnect and ease of configuration for on-device DNNs and computer vision applications

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

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, ADLINK, ADVANTECH, iEi, JW IPC, NEXCOM, tinyGO, uzei		
SOFTWARE TOOLS	INTEL® DISTRIBUTION OF OPENVINO™ TOOLKIT Develop NN Model; Deploy across Intel® CPU, GPU, VPU, FPGA; Leverage common algorithms		

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

Click here for Latest Publicly Posted Benchmarks
Click here for Programing Guide for Use with Intel® Distribution of OpenVINO toolkit

DEMO - HW ACCELERATION WITH INTEL® MOVIDIUS™ NEURAL COMPUTE STICK 2

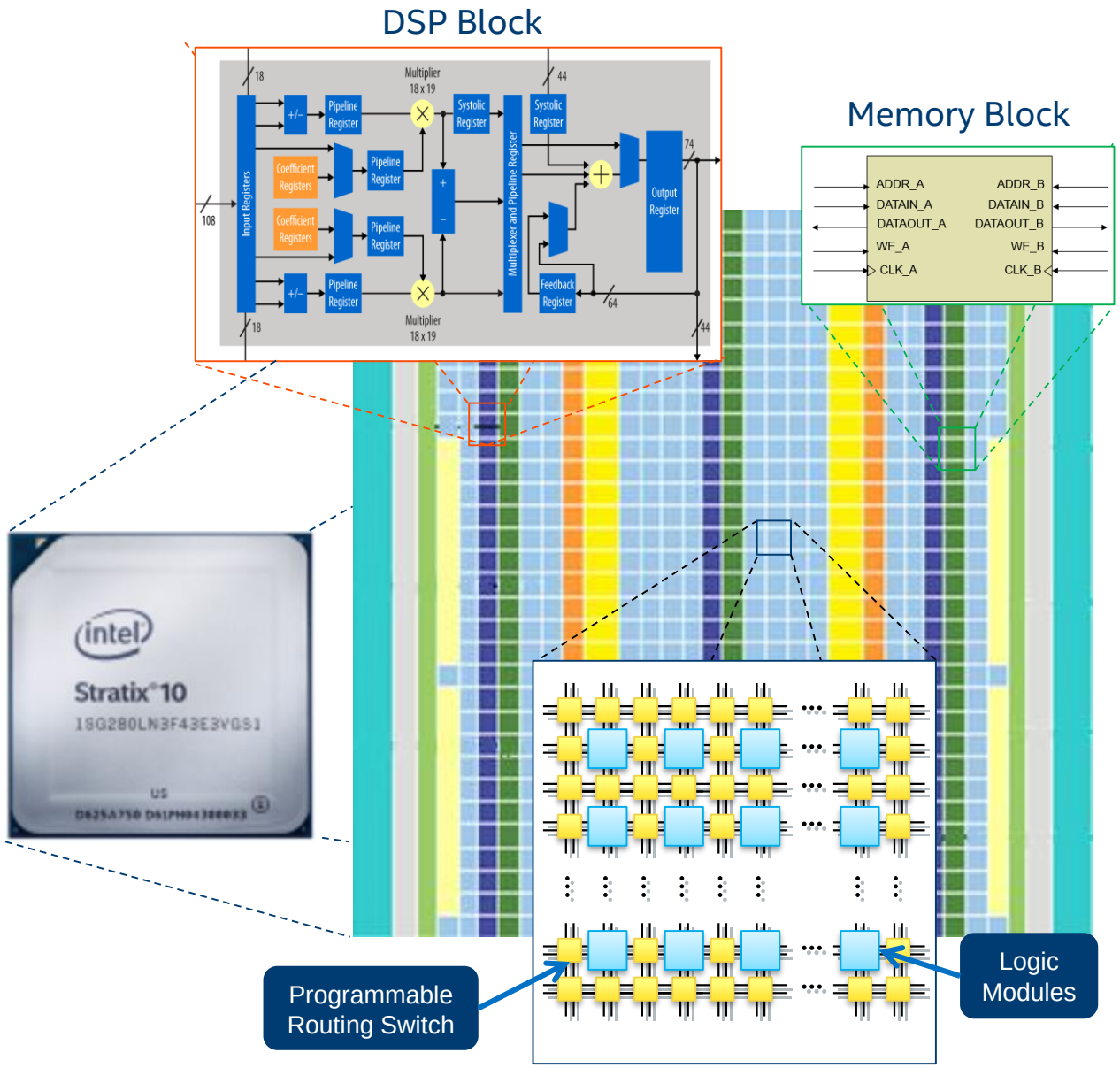
- Run the Object Detection example with IR format FP16 mobilenet-ssd model on Intel® NCS2
- Demo multiple Intel® NCS2 with benchmark_app

AGENDA

- Intel® Smart Video/Computer vision Tools Overview
- Model Optimizer
- Inference Engine
- Multiple Models in One Application
- 15 Minute Break : approx. 1.5 hours from now....
- Accelerators based on Intel® Movidius™ Vision Processing Unit
- Accelerators based on Intel® Arria® FPGA
- DL Workbench & DL Streamer
- Register for access to Intel® DevCloud for Edge
- Labs on Intel® DevCloud for Edge: : 1 hour of online help

FPGA OVERVIEW

- Field Programmable Gate Array (FPGA)
  - Millions of logic elements
  - Thousands of embedded memory blocks
  - Thousands of DSP blocks
  - Programmable routing
  - High speed transceivers
  - Various built-in hardened IP
- Used to create **Custom Hardware!**



WHAT'S INSIDE INTEL® DISTRIBUTION OF OPENVINO™ TOOLKIT

OpenVINO

Deep Learning

Intel® Deep Learning Deployment Toolkit

Model Optimizer
Convert & Optimize

IR

Inference Engine
Optimized Inference

IR = Intermediate Representation file

Open Model Zoo

- 40+ Pretrained Models
- Samples
- Model Downloader

Deep Learning Workbench

- Calibration Tool
- Model Analyzer
- Benchmark App
- Accuracy Checker
- Aux. Capabilities

Traditional Computer Vision

Optimized Libraries & Code Samples

- OpenCV*
- Samples

For Intel® CPU & GPU/Intel® Processor Graphics

Tools & Libraries

Increase Media/Video/Graphics Performance

- **Intel® Media SDK** Open Source version
- **OpenCL™ Drivers & Runtimes**

For GPU/Intel® Processor Graphics

Intel® FPGA Deep Learning Acceleration Suite

- **FPGA Runtime Environment (RTE)** (from Intel® FPGA SDK for OpenCL™)
- **Bitstreams**

OS Support: CentOS* 7.4 (64 bit), Ubuntu* 16/18 LTS (64 bit), Microsoft Windows® 10 (64 bit), Yocto Project* version Poky Jethro v2.0.3 (64 bit), macOS* 10.13 & 10.14 (64 bit)

Intel® Architecture-Based Platforms Support

CELERON inside | ATOM inside | CORE inside | XEON inside | ARRIA 10 inside | MOVIDIUS inside | IRIS Pro GRAPHICS

Intel® Vision Accelerator Design Products & AI in Production/ Developer Kits

An open source version is available at 01.org/openvinotoolkit (some deep learning functions support Intel CPU/GPU only).

INTEL® DISTRIBUTION OF OPENVINO™ TOOLKIT WITH DLA USER FLOWS

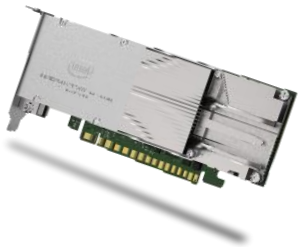
Application Developer

Design

Intel® Distribution of OpenVINO™ toolkit

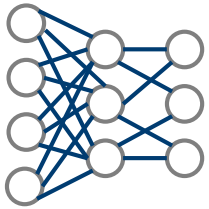
Run

Program



Turnkey Software Deployment Flow

Neural Net



Bitstream Library

Choose from many **precompiled** FPGA Images

Or

Custom create FPGA bitstream

FPGA Architecture Development Flow

IP Architect

Design

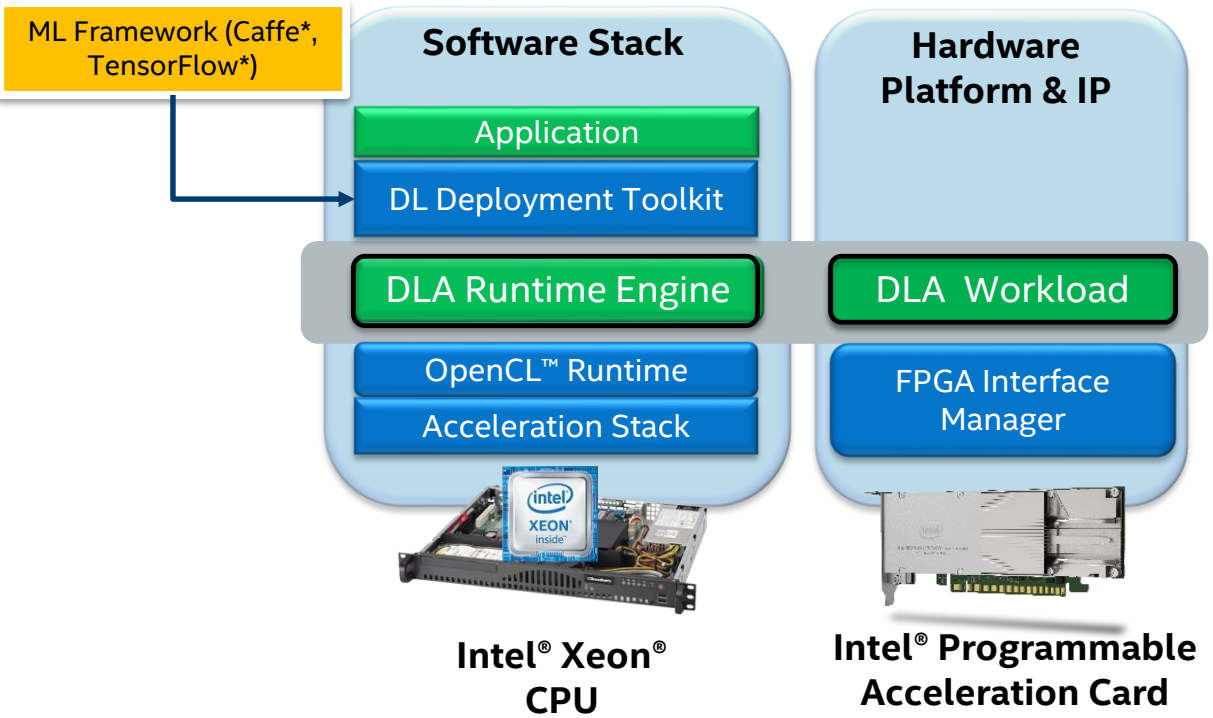
User Customization of DLA Suite Source Code

Compile

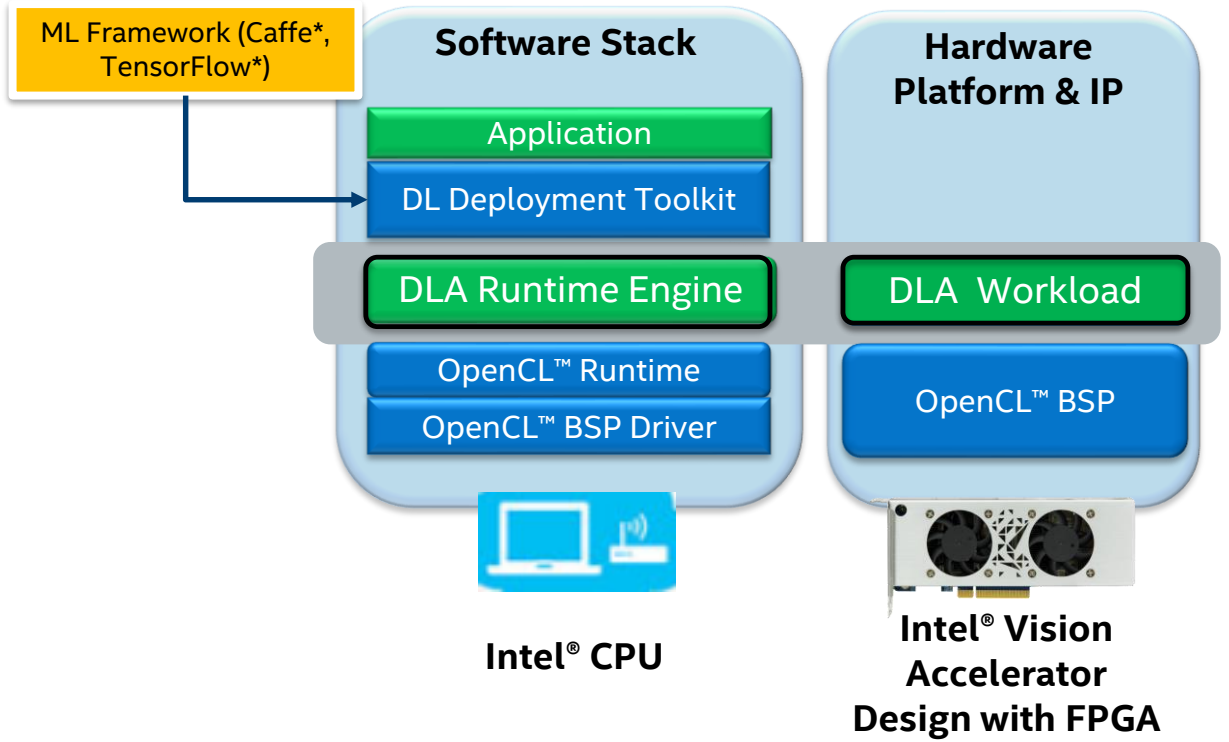
Intel® FPGA SDK for OpenCL™

MACHINE LEARNING ON INTEL® FPGA PLATFORM

Acceleration Stack Platform Solution



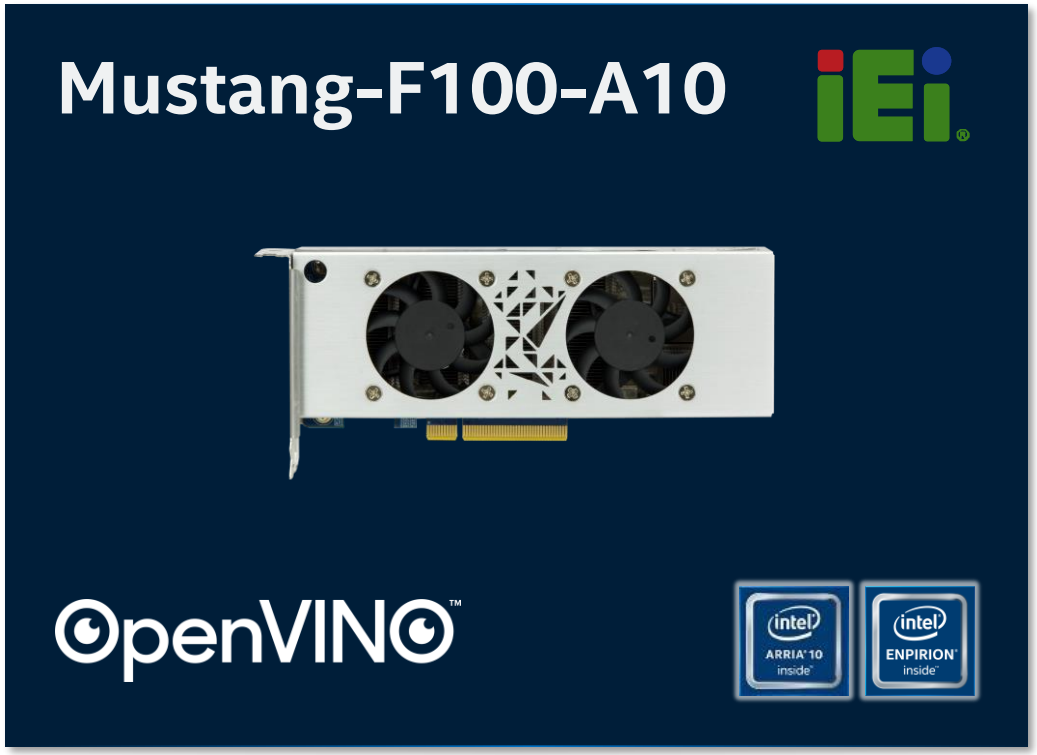
Edge Computing Solution



[Intel® FPGA Acceleration Hub]

[Intel® Vision Accelerator Design Products]

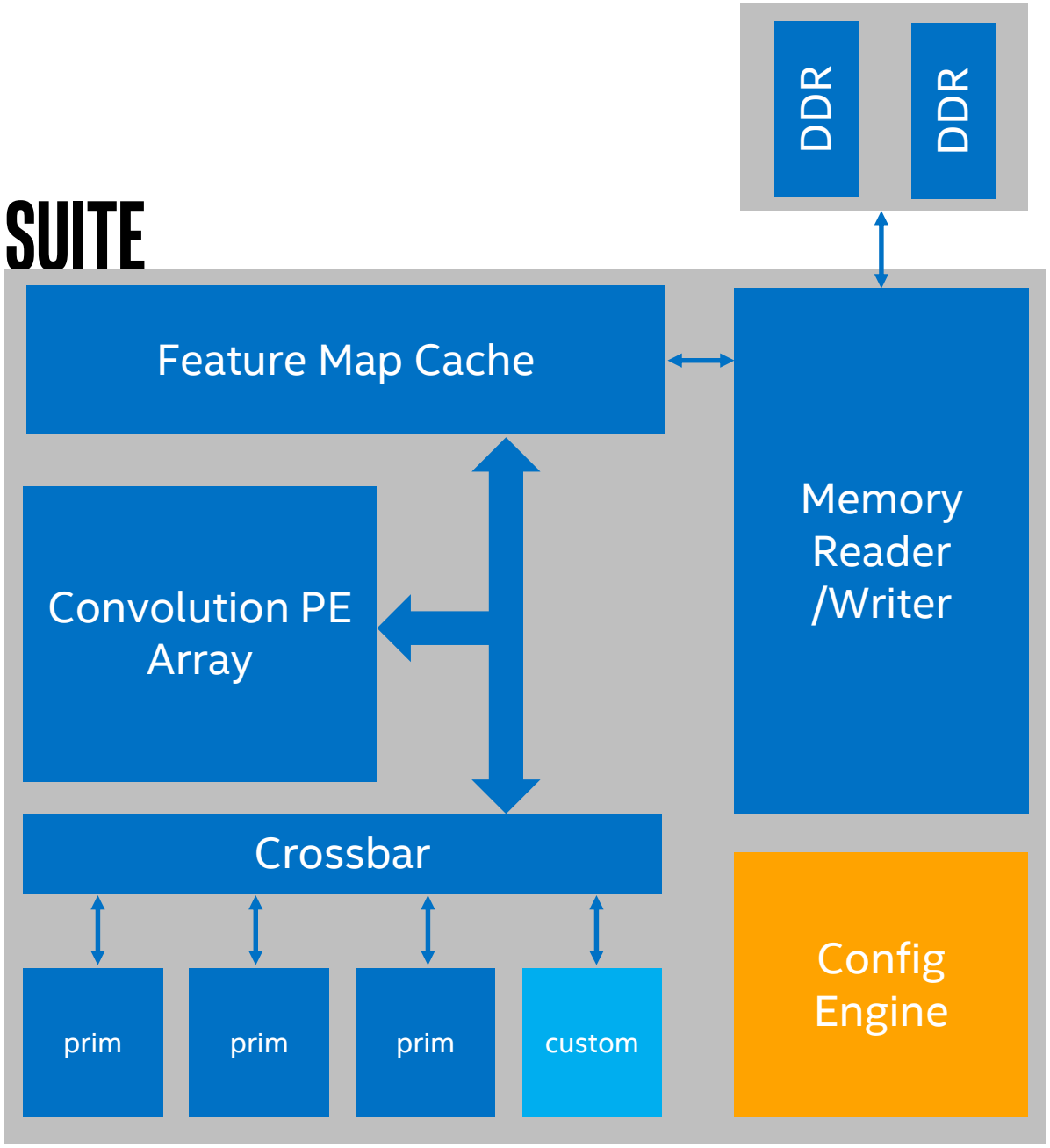
INTEL® VISION ACCELERATION DESIGN WITH INTEL® ARRIA® 10 FPGA
KEY DIFFERENTIATORS



- High performance, low latency
- Flexibility to adapt to new, evolving, and custom networks
- Supports large image sizes (e.g., 4K)
- Large networks (up to 4 billion parameters)
- Wide ambient temperature range (0° C to 65° C)
- 24/7/365 operation
- Long lifespan (8–10 years)

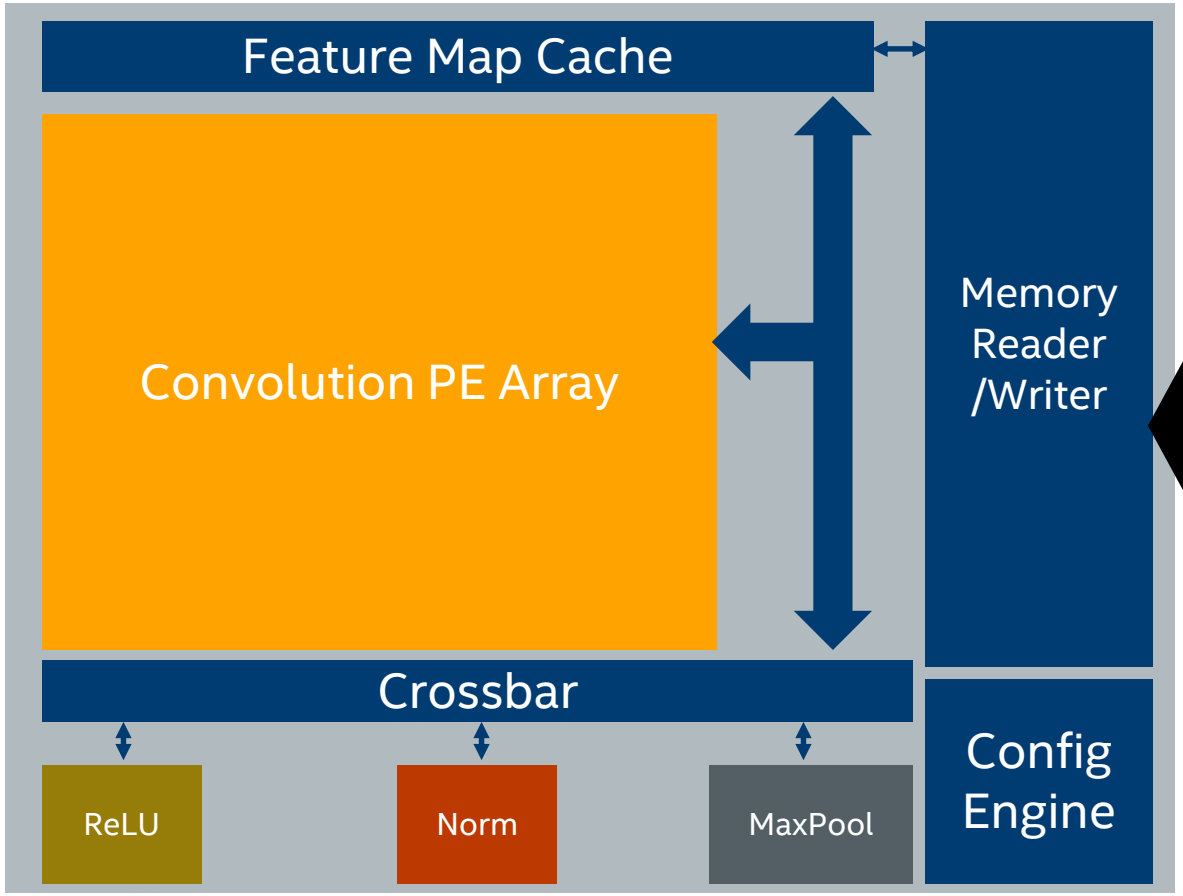
INTEL® FPGA DEEP LEARNING ACCELERATION SUITE

- CNN inference acceleration engine for topologies executed in a graph loop architecture
  - AlexNet, GoogleNet, SqueezeNet, VGG, ResNet*, MobileNet*, Yolo, SSD, …
- Software Deployment
  - No FPGA compile required
  - Run-time reconfigurable
- Customized Hardware Development
  - **Custom architecture creation w/ parameters**
  - Custom primitives using OpenCL™ flow

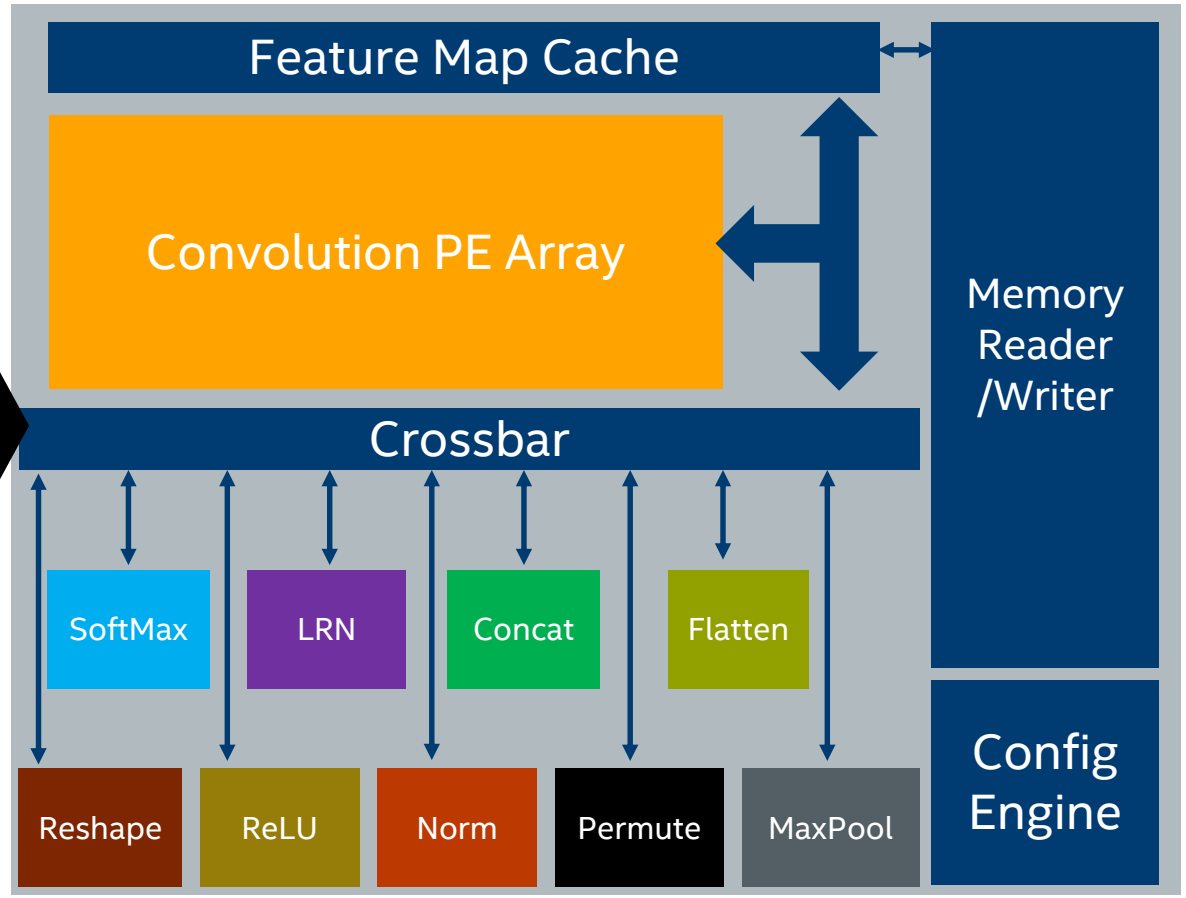


SUPPORT FOR DIFFERENT TOPOLOGIES

Adapts to support new or evolving networks

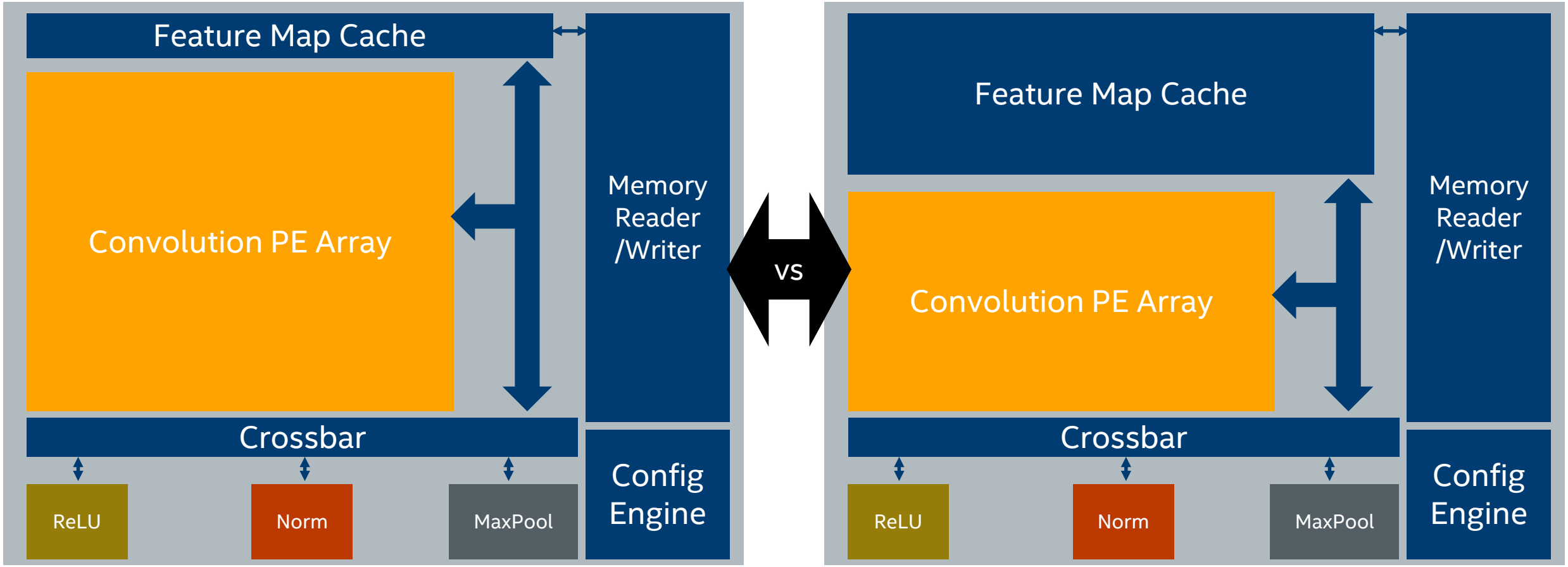


vs



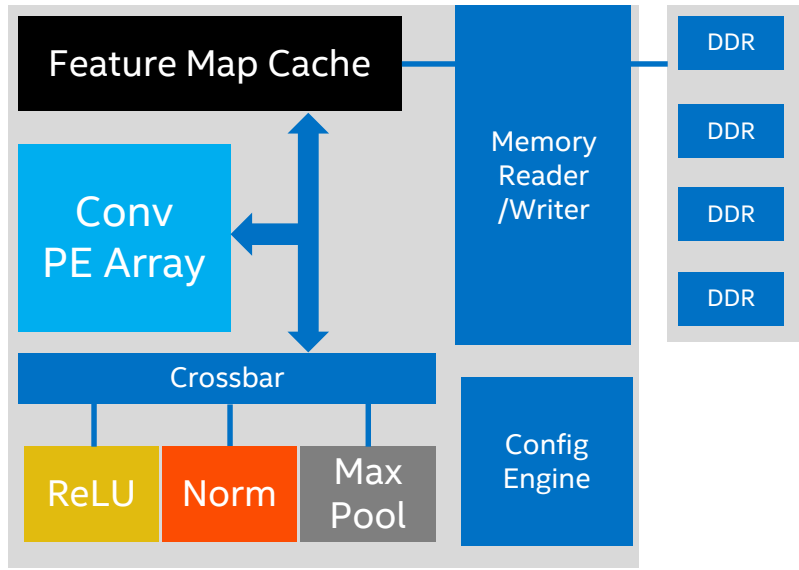
OPTIMIZE FOR BEST PERFORMANCE

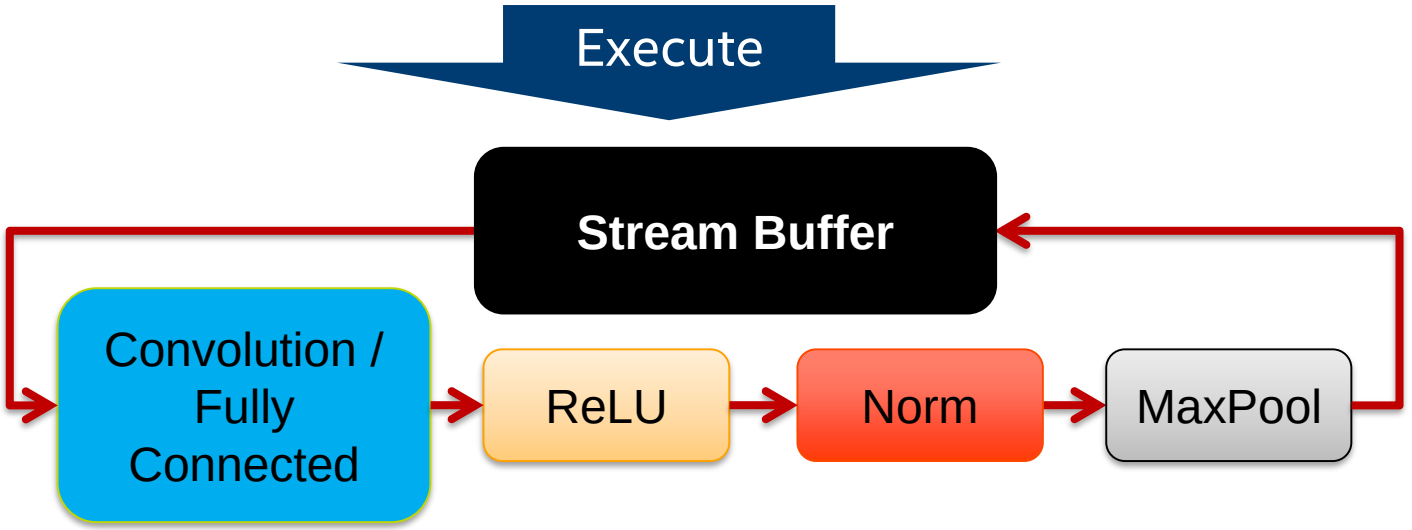
Tradeoff between size of Feature Map cache and convolutional PE array

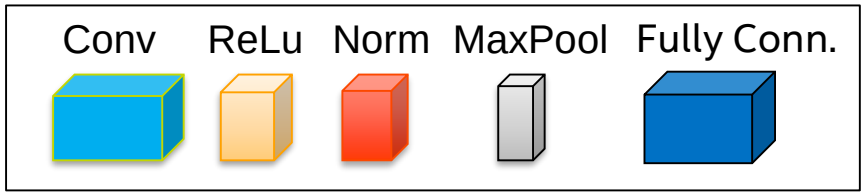


DLA ARCHITECTURE: BUILT FOR PERFORMANCE

- Maximize Parallelism on the FPGA
  - Filter Parallelism (Processing Elements)
  - Input-Depth Parallelism
  - Winograd Transformation
  - Batching
  - Feature Stream Buffer
  - Filter Cache
- Choosing FPGA Bitstream
  - Data Type / Design Exploration
  - Primitive Support

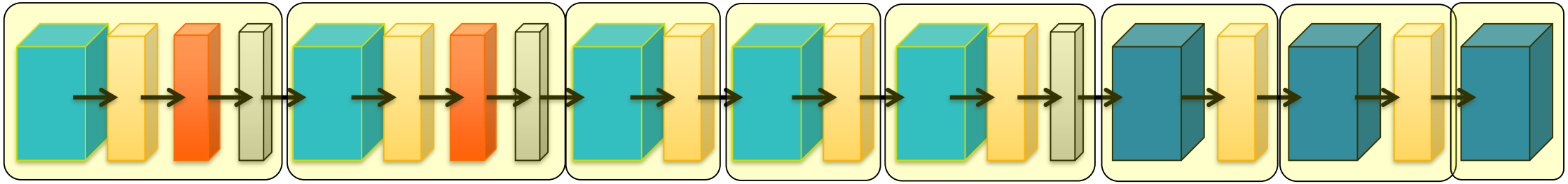


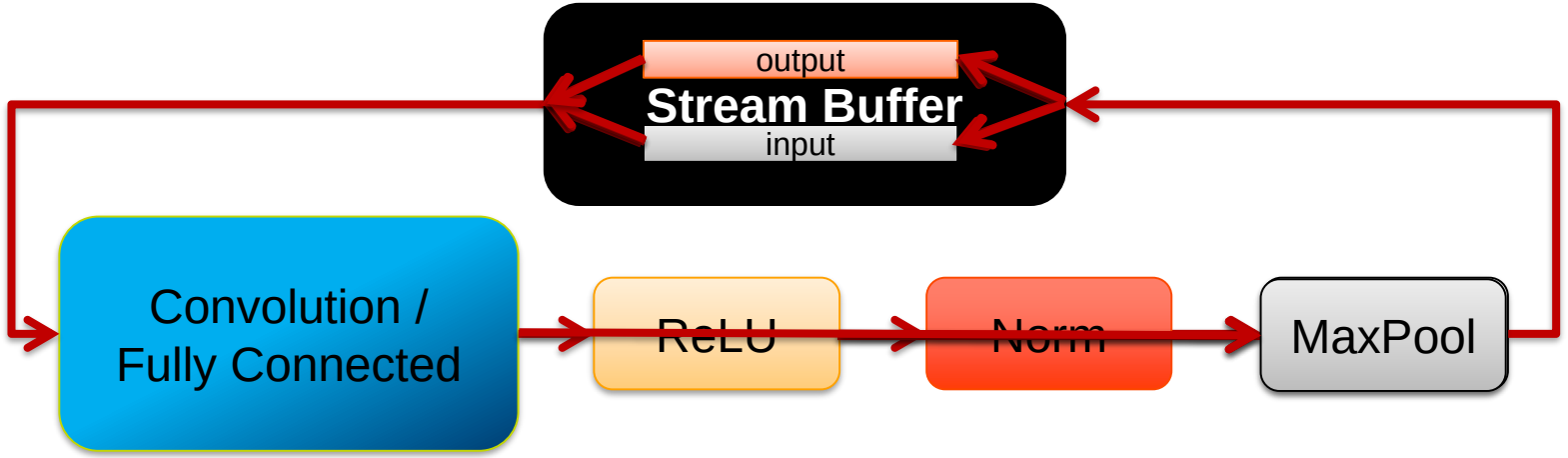




MAPPING GRAPHS IN DLA

AlexNet Graph

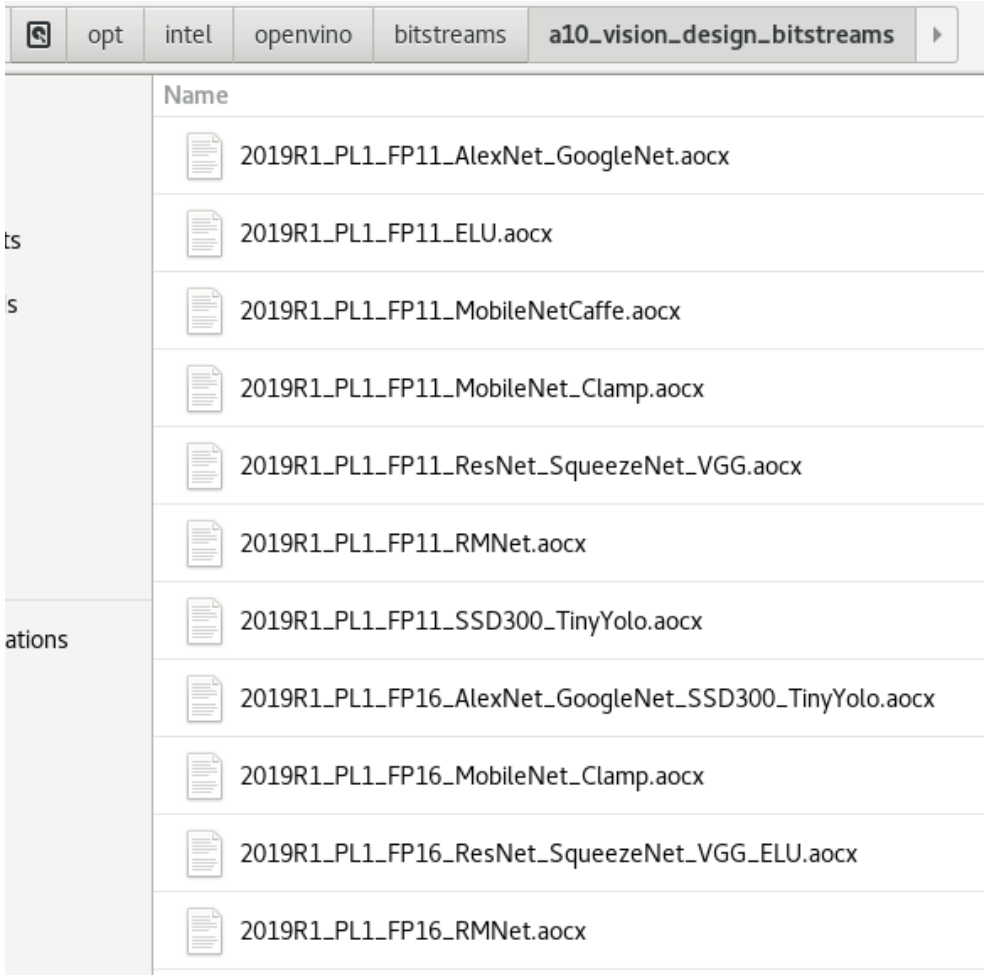




Blocks are run-time reconfigurable and bypassable

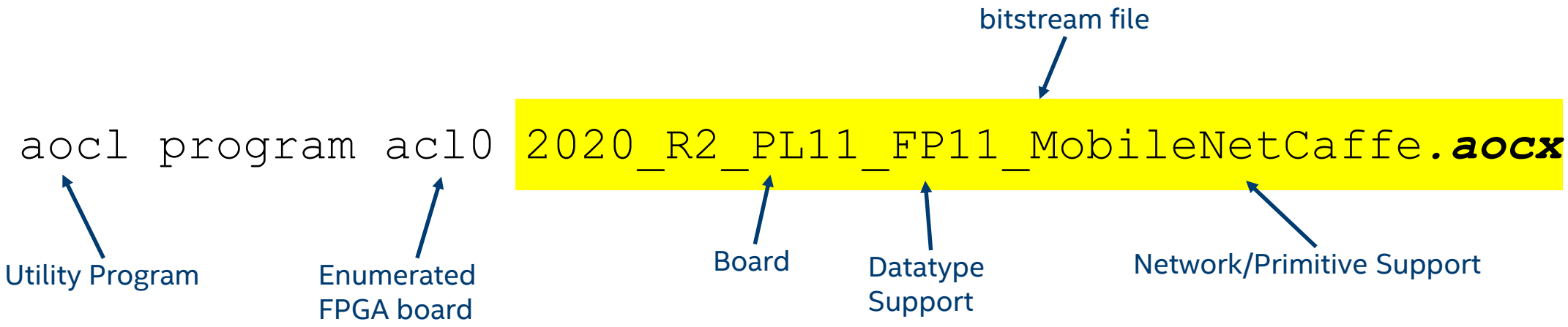
DLA ARCHITECTURE SELECTION

- Intel® Distribution of OpenVINO™ toolkit ships with many FPGA images for various boards/data types/topologies
  - <version>_<board>_<data type>_<Topologies/Feature>.aocx
- Find ideal FPGA image that meets your needs
- Check documentation for list of FPGA images and supported topologies
  - https://docs.openvinotoolkit.org/latest/_docs_IE_DG_supported_plugins_FPGA.html
- Example: ResNet* focused image does not have Norm (better performance)



LOAD SELECTED BITSTEAM PRIOR TO EXECUTION

- Program the FPGA with the selected FPGA bitstream



AGENDA

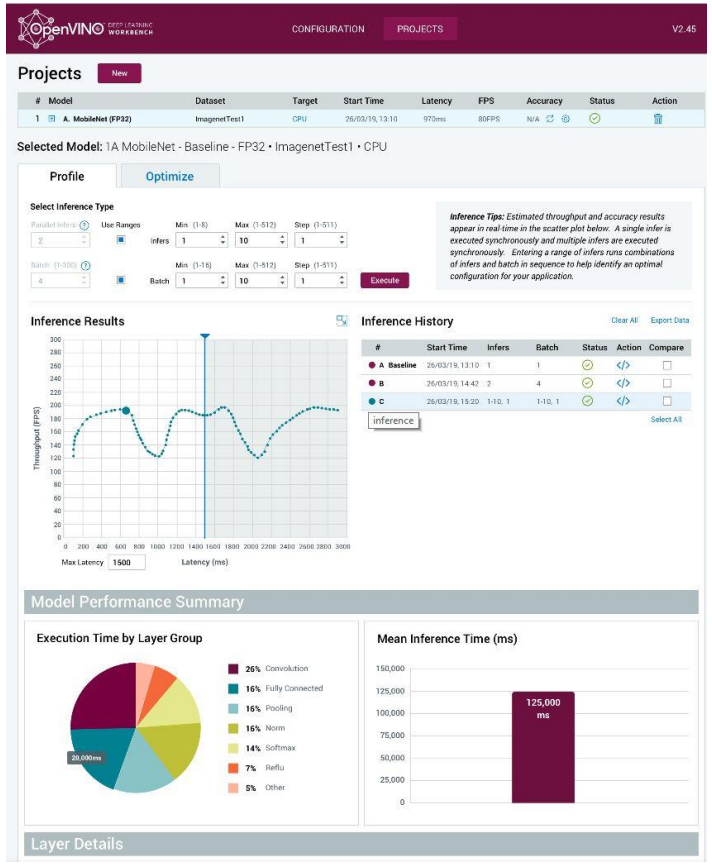
- Intel® Smart Video/Computer vision Tools Overview
- Model Optimizer
- Inference Engine
- Multiple Models in One Application
- 15 Minute Break : approx. 1.5 hours from now....
- Accelerators based on Intel® Movidius™ Vision Processing Unit
- Accelerators based on Intel® Arria® FPGA
- DL Workbench & DL Streamer
- Register for access to Intel® DevCloud for Edge
- Labs on Intel® DevCloud for Edge: : 1 hour of online help

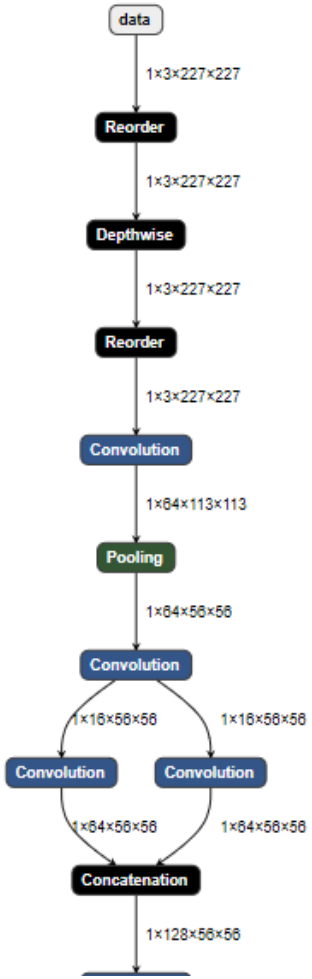
USE DL WORKBENCH

DEEP LEARNING WORKBENCH

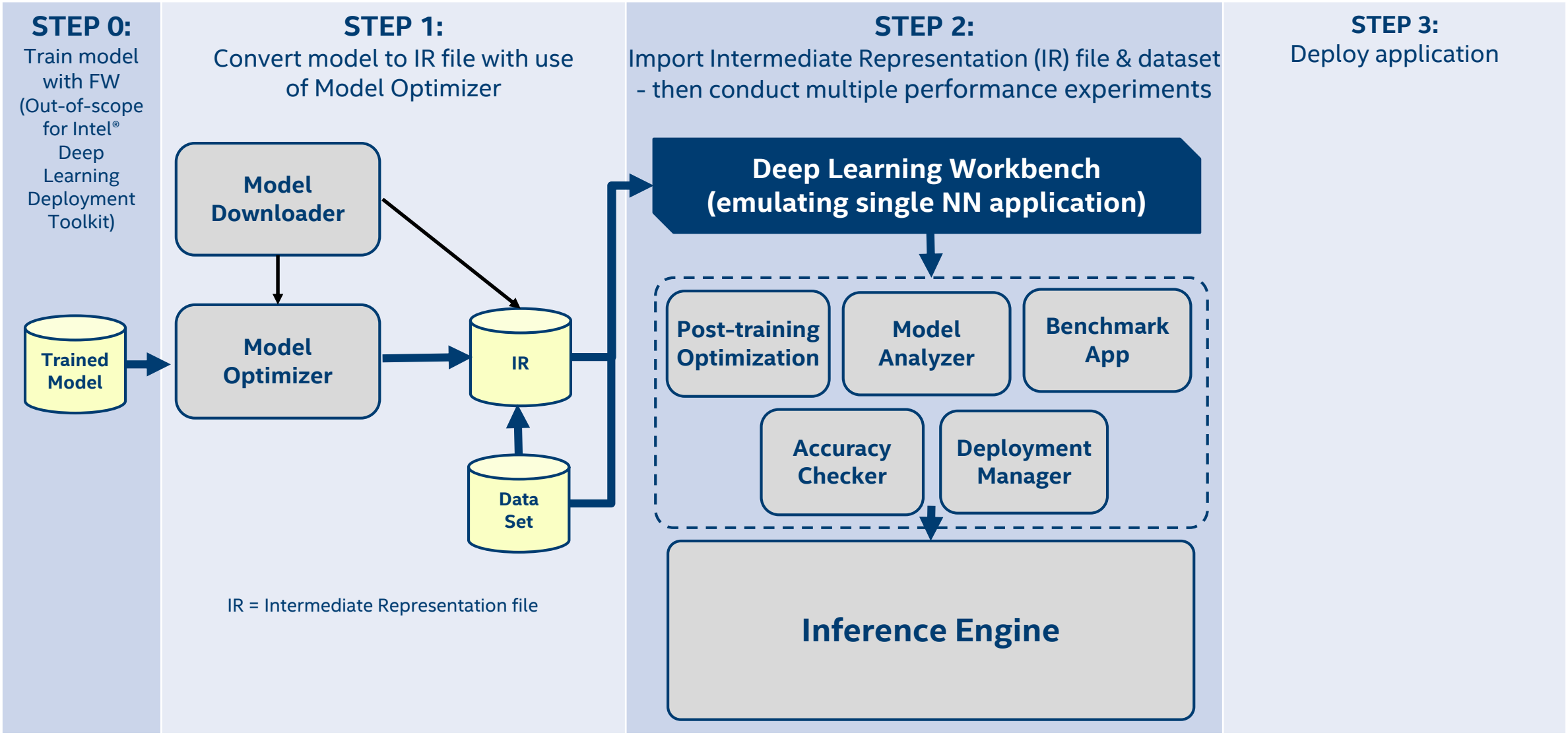
Deep Learning Workbench capabilities

- Web-based tool - UI extension of Intel® Distribution of OpenVINO™ toolkit functionality
- Visualizes performance data for topologies/ layers to aid in model analysis
- Automate analysis for optimal performance configuration (streams, batches, latency)
- Experiment with int8 calibration for optimal tuning
- Provide accuracy info through accuracy checker
- Direct access to Models from public set of Open Model Zoo



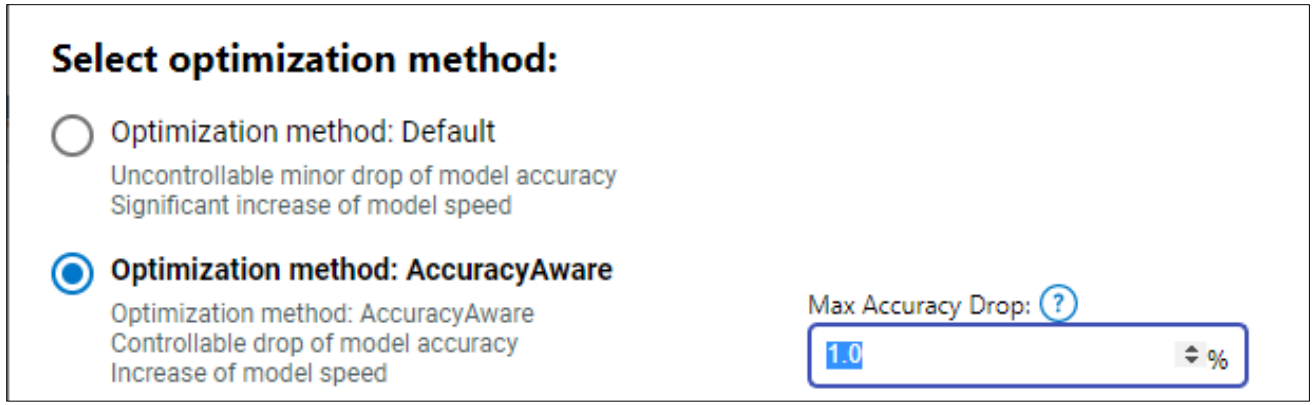


DEEP LEARNING WORKBENCH DATA FLOW

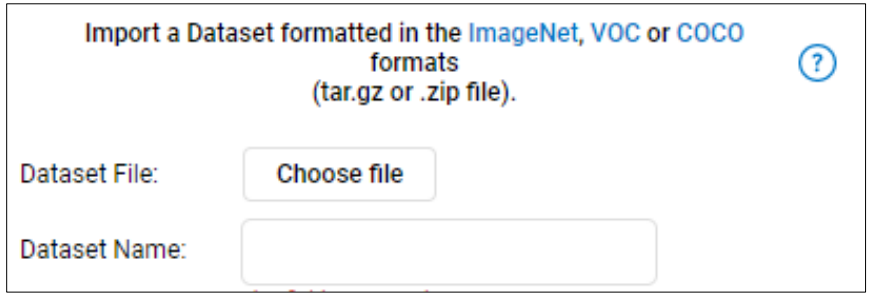


DEEP LEARNING WORKBENCH : NEW FEATURES

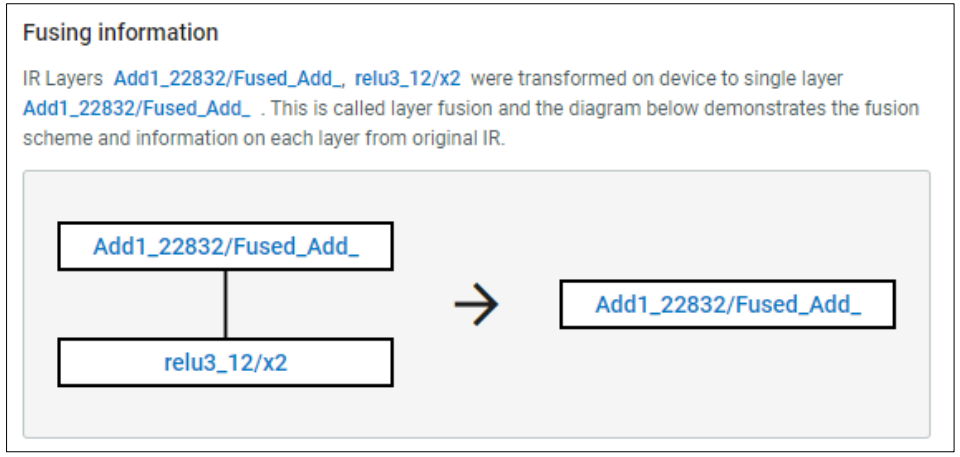
CONVERT MODEL TO INT8 USING 2 NEW CALIBRATION ALGORITHMS



IMPORT DATASET IN COCO FORMAT TO USE WITH MODEL



IMPROVED PER-LAYER DATA VISUALIZATION AND COMPARISON MODE.



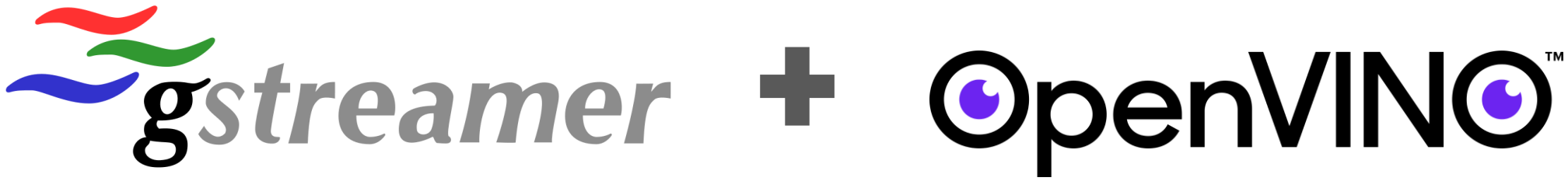
DEMO

DL WORKBENCH WALKTHROUGH

USE DL STREAMER

INTRODUCING.. DL STREAMER

- Intel® Distribution of OpenVINO™ toolkit Deep Learning (DL) Streamer, now part of the **default installation** package
- Enables developers to **create and deploy optimized streaming media analytics pipelines** across Intel® architecture from edge to cloud
- Optimal pipeline interoperability with a **familiar developer experience** built using the GStreamer multimedia framework

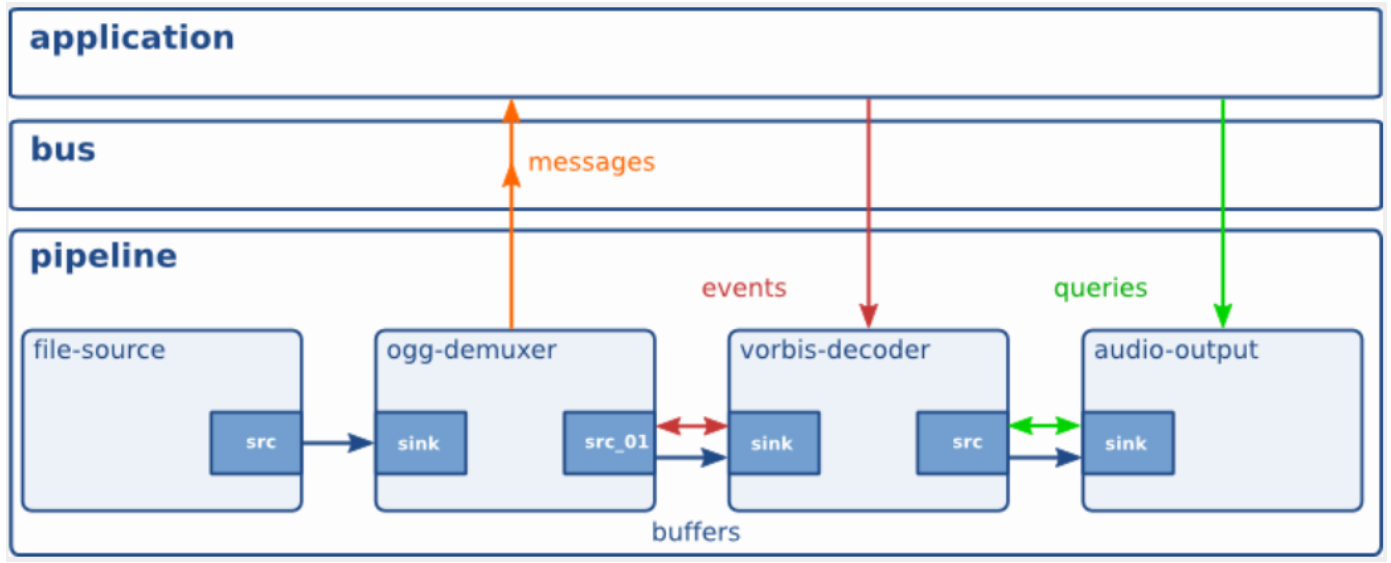


HARDWARE SUPPORTED
CPU
iGPU
VPU
FPGA
GNA

OS SUPPORTED
WIN 10
WIN SERVER
LINUX
MAC

WHAT IS GSTREAMER?

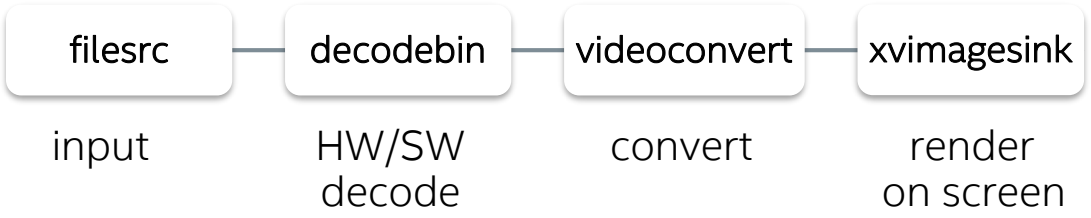
- A pipeline consists of **connected processing elements**
- Each element is provided by **a plug-in** and can be **grouped into bins**
- Elements communicate by means of **pads** – source pad and sink pad
- Data buffers flow **from Source** element **to Sink** element & from source pad to sink pad



Ref:
https://gstreamer.freedesktop.org/data/doc/gstreamer/head/manual/manual.pdf

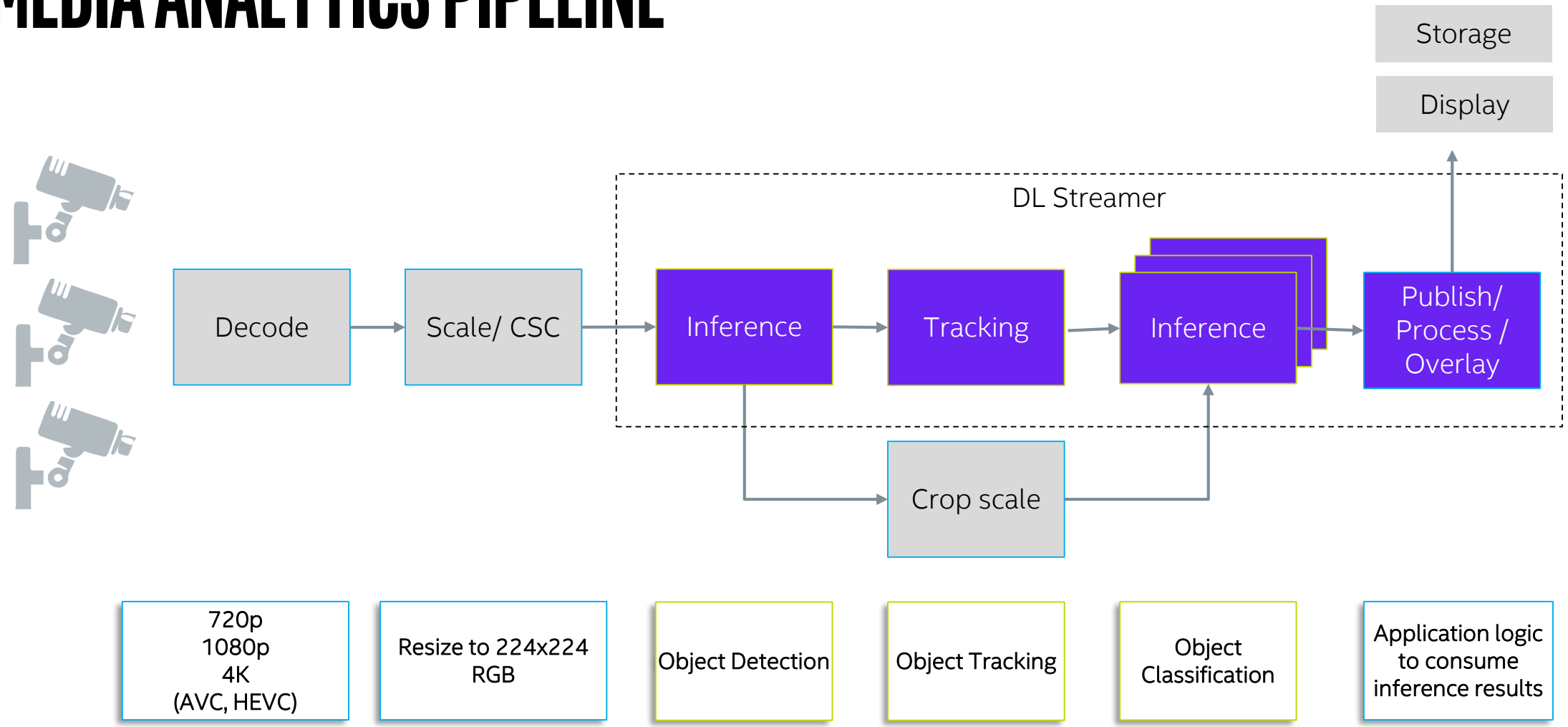
MEDIA PROCESSING PIPELINE

Video Pipeline – decode, convert, render

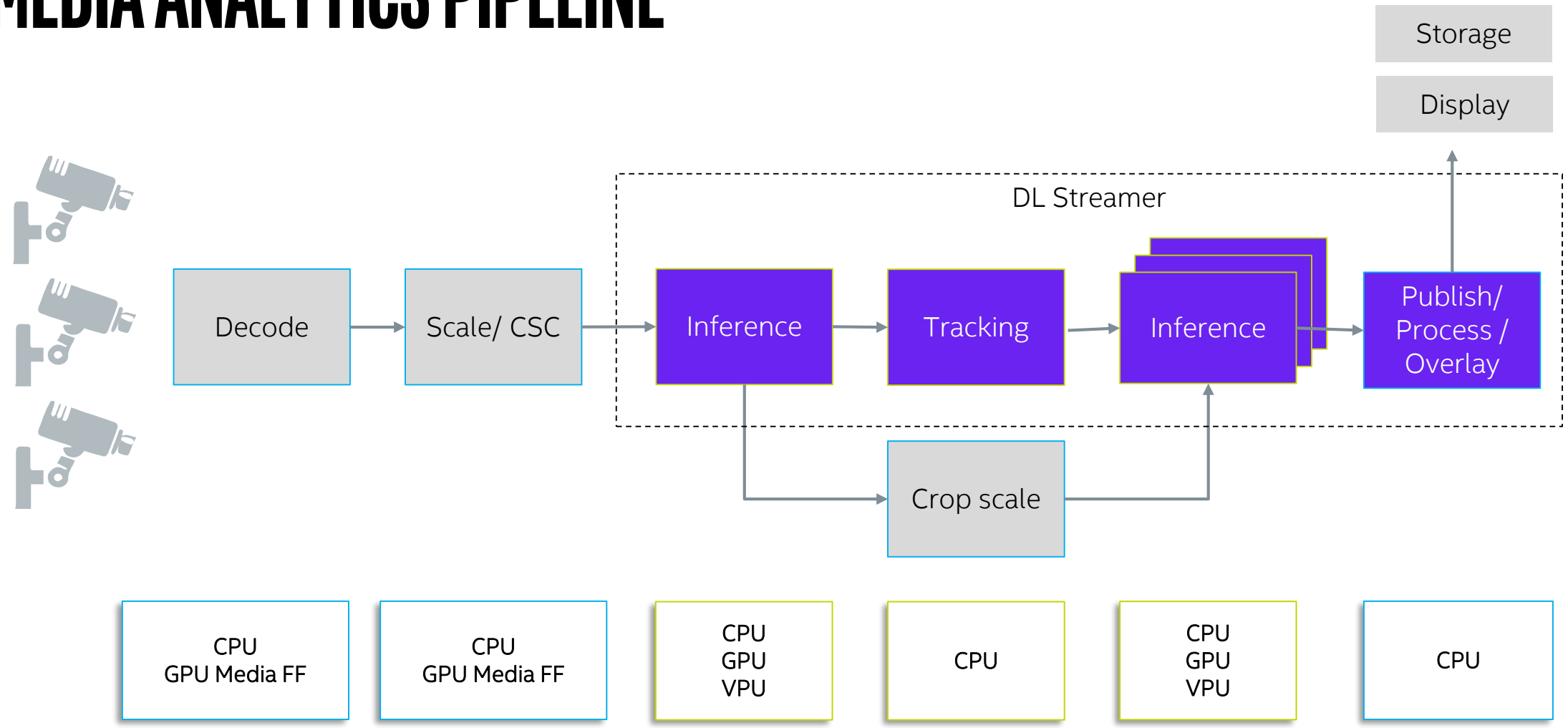


```
gst-launch-1.0 filesrc location=/path/to/video.mp4 ! decodebin ! videoconvert ! xvimagesink
```

MEDIA ANALYTICS PIPELINE

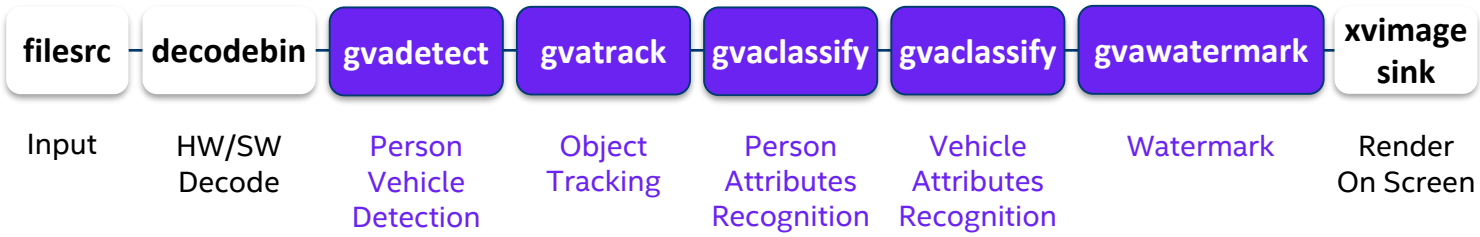


MEDIA ANALYTICS PIPELINE



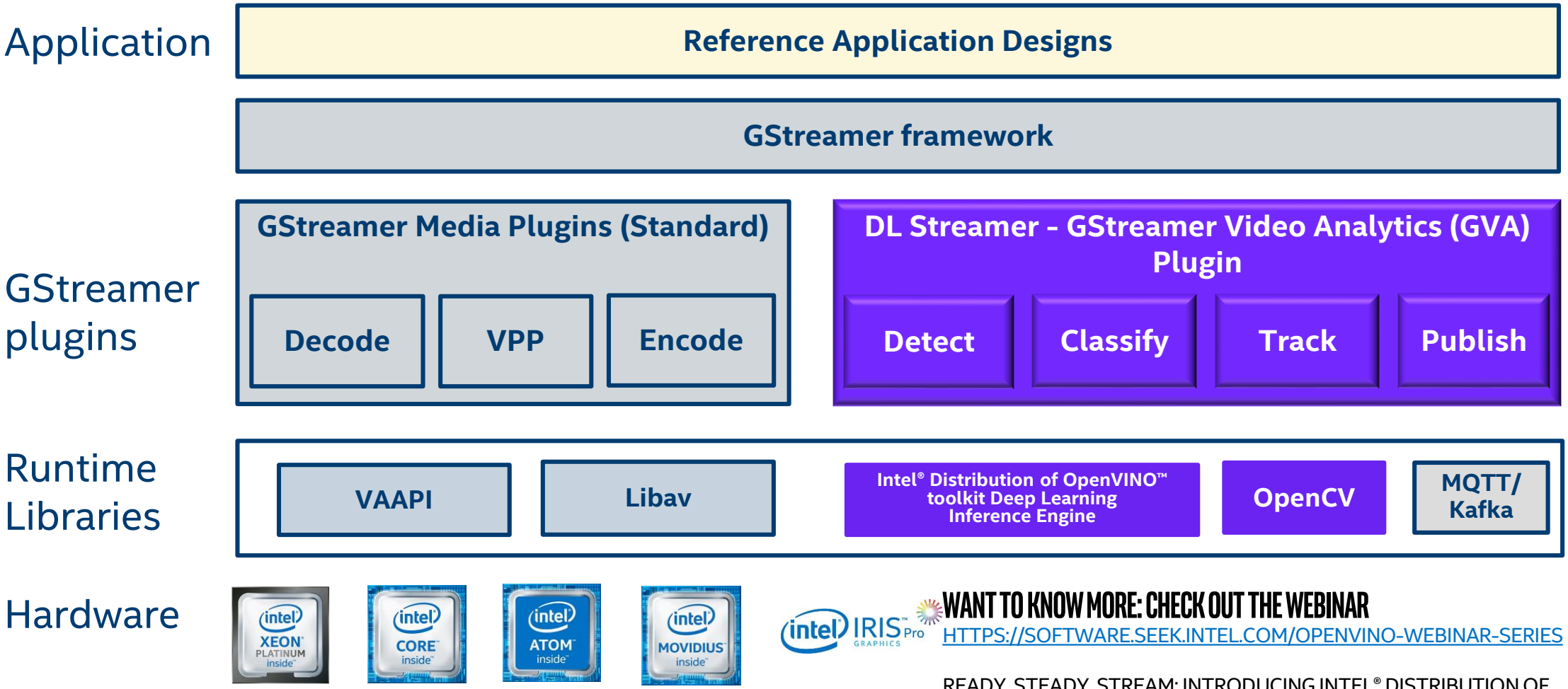
USING THE DL STREAMER

Video Analytics pipeline – person and vehicle detection, person, vehicle attributes classification




```
 gst-launch-1.0 filesrc location=/path/to/video.mp4 !
 decodebin ! videoconvert ! video/x-raw,format=BGRx ! \
 gvadetect model=person-vehicle-bike-detection-crossroad-0078.xml model-proc=person-vehicle-bike-detection-
crossroad-0078.json inference-interval=10 threshold=0.6 device=CPU ! queue ! \
 gvatrack tracking-type="short-term" ! queue ! \
 gvaclassify model= person-attributes-recognition-crossroad-0230.xml model-proc= person-attributes-recognition-
crossroad-0230.json reclassify-interval=10 device=CPU object-class=person ! queue ! \
 gvaclassify model= vehicle-attributes-recognition-barrier-0039.xml model-proc= vehicle-attributes-recognition-
barrier-0039.json reclassify-interval=10 device=CPU object-class=vehicle ! queue ! \
 gvawatermark ! videoconvert ! fpsdisplaysink video-sink=xvimagesink sync=true
```

UNDER THE HOOD: DL STREAMER



WANT TO KNOW MORE: CHECK OUT THE WEBINAR

HTTPS://SOFTWARE.SEEK.INTEL.COM/OPENVINO-WEBINAR-SERIES

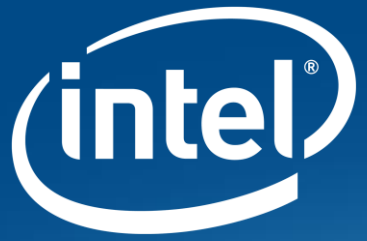
READY, STEADY, STREAM: INTRODUCING INTEL® DISTRIBUTION OF OPENVINO™ TOOLKIT DEEP LEARNING STREAMER

DEMO

VIDEO ANALYTICS PIPELINE – PERSON AND VEHICLE DETECTION WITH ATTRIBUTES CLASSIFICATION

AGENDA

- Intel® Smart Video/Computer vision Tools Overview
- Model Optimizer
- Inference Engine
- Multiple Models in One Application
- 15 Minute Break : approx. 1.5 hours from now....
- Accelerators based on Intel® Movidius™ Vision Processing Unit
- Accelerators based on Intel® Aria® FPGA
- DL Workbench & DL Streamer
- Register for access to Intel® DevCloud for Edge
- Labs on Intel® DevCloud for Edge: : 1 hour of online help

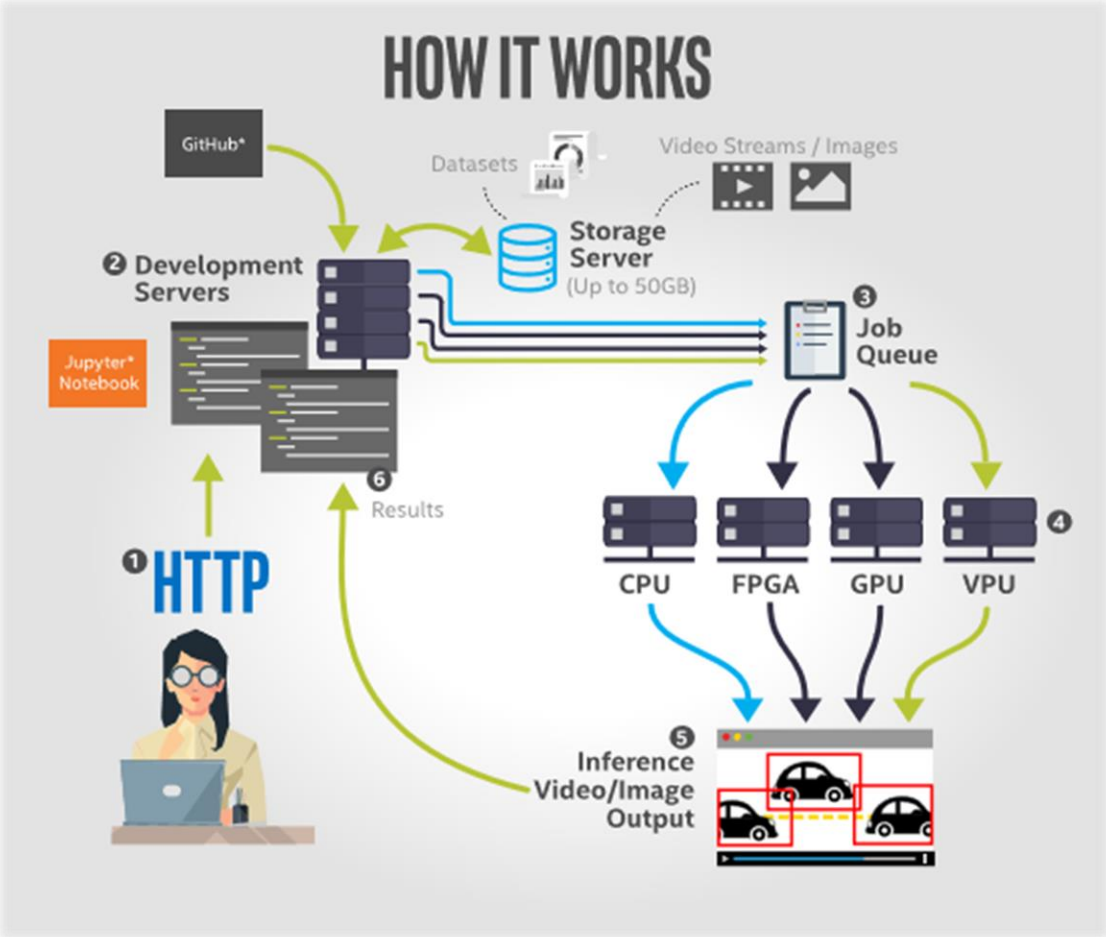


INTEL® DEVCLOUD FOR THE EDGE

Sign Up Here: https://devcloud.intel.com/edge

ACCELERATE TIME TO PRODUCTION WITH INTEL® DEVCLOUD FOR THE EDGE

SEE IMMEDIATE AI APPLICATION 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

Sign up now for access

DEMO

DEVCLOUD SAMPLE APPLICATION WALKTHROUGH

Signup for Access to the Intel® DevCloud for Edge

Sign Up Here: https://devcloud.intel.com/edge/

Intel's Registration Passcode:	
Code Valid From:	
Code Valid To:	
Account Activation:	
Account Deactivation:	Valid for 30 days

https://bit.ly/VINOsurvey