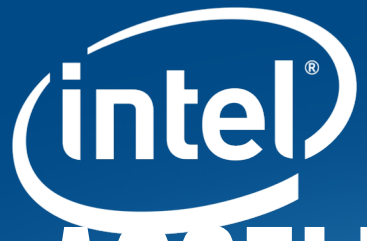


ACCELERATE DEEP LEARNING INFERENCE USING INTEL TECHNOLOGIES

OBJECT DETECTION USING INTEL® DISTRIBUTION OF OPENVINO™ TOOLKIT

November 2018

Core and Visual Computing Group

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

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/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.

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

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.

Smart Video Workshop Overview

Introduction

1. Introduction to Intel technologies for deep learning inference
2. Hardware acceleration techniques

Each module contains a hands-on lab exercise that introduces various Intel technologies to accelerate computer vision application with hardware heterogeneity.

Intel® Distribution of OpenVINO™ 101

2. Basic End-to-End Object Detection Example

Hardware Acceleration

3./4./5. Hardware Acceleration with CPU, Integrated GPU, Intel® Movidius™ NCS, FPGA

Optimization

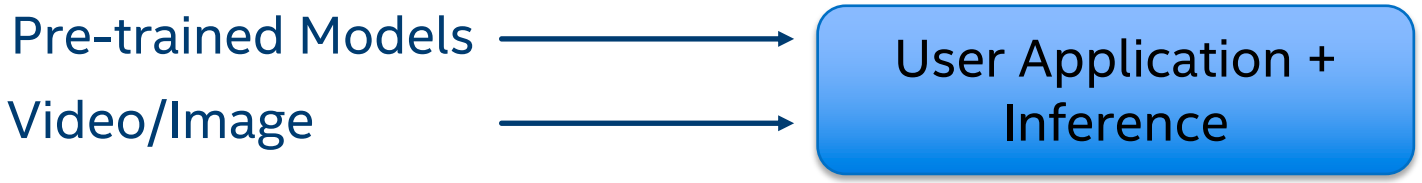
6. Optimization Tools and Techniques

Application

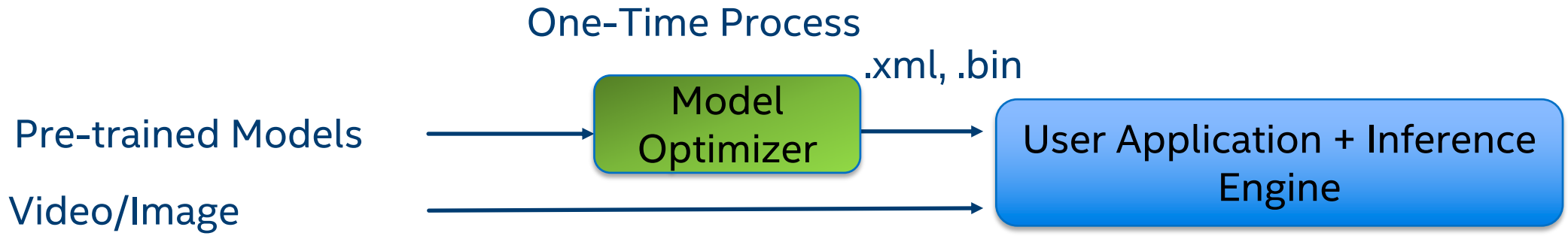
7. Advanced Video Analytics

Deep Learning Application Deployment

Traditional



With Intel® Distribution of OpenVINO™ Toolkit



Intel® Deep Learning Deployment Toolkit

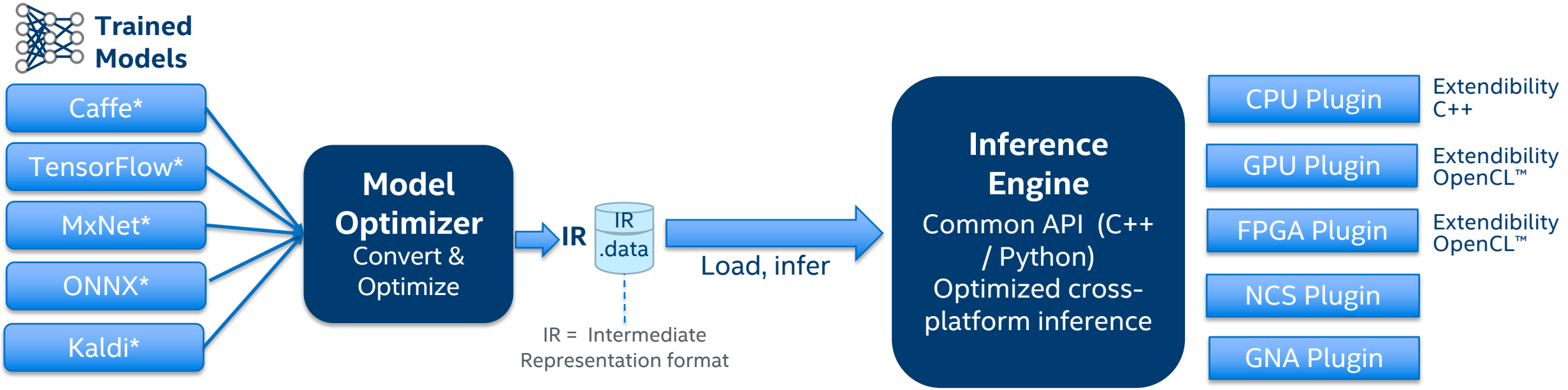
For Deep Learning Inference

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

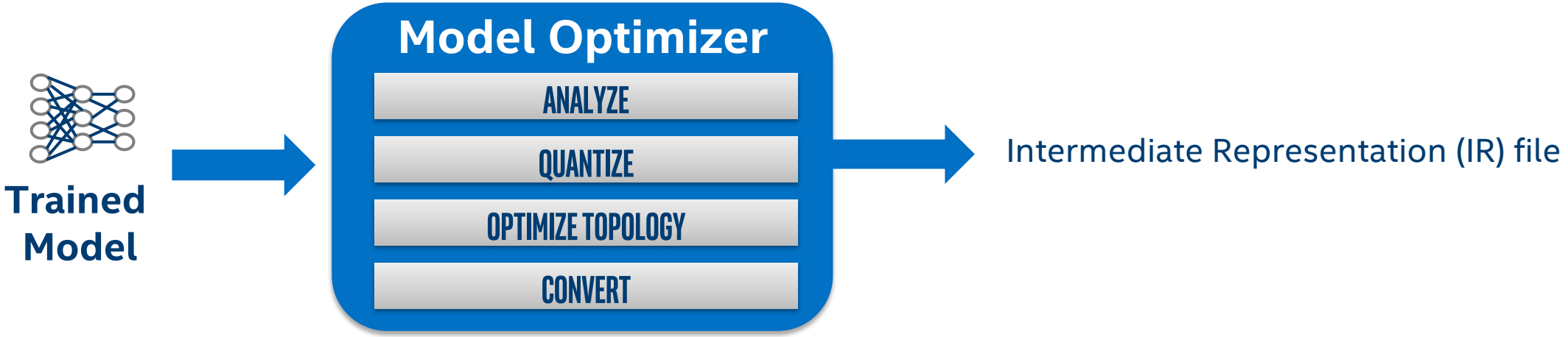
- **What it is**: High-level inference API
- **Why important**: Interface is implemented as dynamically loaded plugins for each hardware type. Delivers best performance for each type without requiring users to implement and maintain multiple code pathways.



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

Model Optimizer

Improve Performance with Model Optimizer



- Easy to use, Python*-based workflow does not require rebuilding frameworks.
- Import Models from various supported frameworks - Caffe*, TensorFlow*, MXNet*, ONNX*, Kaldi*.
- 100+ models for Caffe, MXNet and TensorFlow validated. All public models on ONNX* model zoo supported.
- With Kaldi support, the model optimizer extends inferencing for non-vision networks.
- IR files for models using standard layers or user-provided custom layers do not require Caffe.
- Fallback to original framework is possible in cases of unsupported layers, but requires original framework.

Model Optimizer

Model optimizer performs generic optimization:

- Node merging
- Horizontal fusion
- Batch normalization to scale shift
- Fold scale shift with convolution
- Drop unused layers (dropout)
- FP16/FP32 quantization
- Model optimizer can add normalization and mean operations, so some preprocessing is ‘added’ to the deep learning model

  --mean_values (104.006, 116.66, 122.67)

  --scale_values (0.07, 0.075, 0.084)

	FP32	FP16
CPU	yes	no
GPU	yes	recommended
MYRIAD	no	yes
FPGA/DLA	no	yes

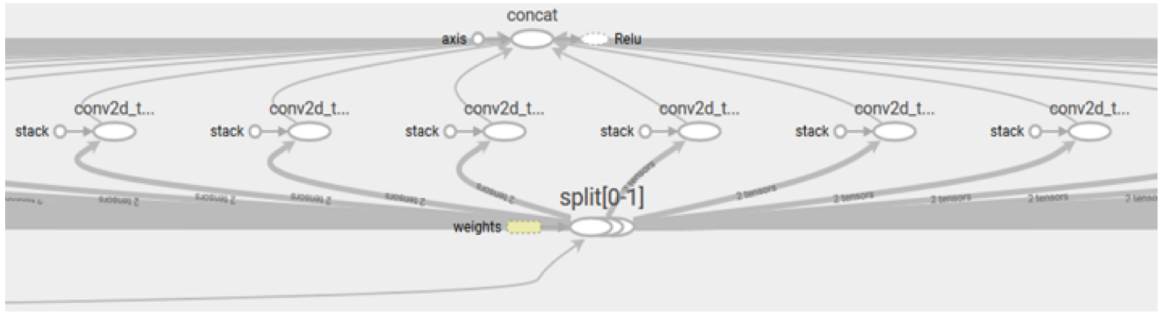
Intel® DL Deployment Toolkit Functionality
Model Optimizer/Optimization Tech

Linear Operation Fusing

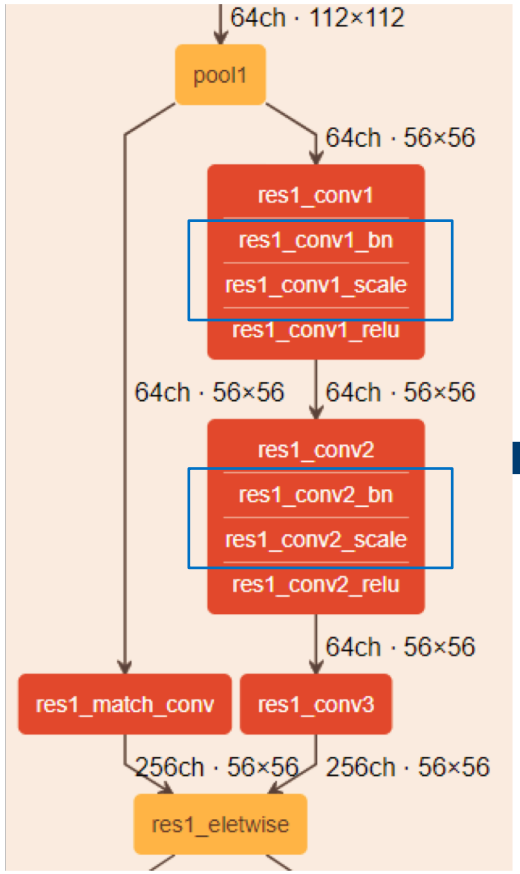
- **BatchNorm and ScaleShift decomposition:** *BN* layers decomposes to *Mul->Add->Mul->Add* sequence; ScaleShift layers decomposes to *Mul->Add* sequence.
- **Linear operations merge:** Merges sequences of Mul and Add operations to the **single** Mul->Add instance.
- **Linear operations fusion:** Fuses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

Grouped Convolutions Fusing

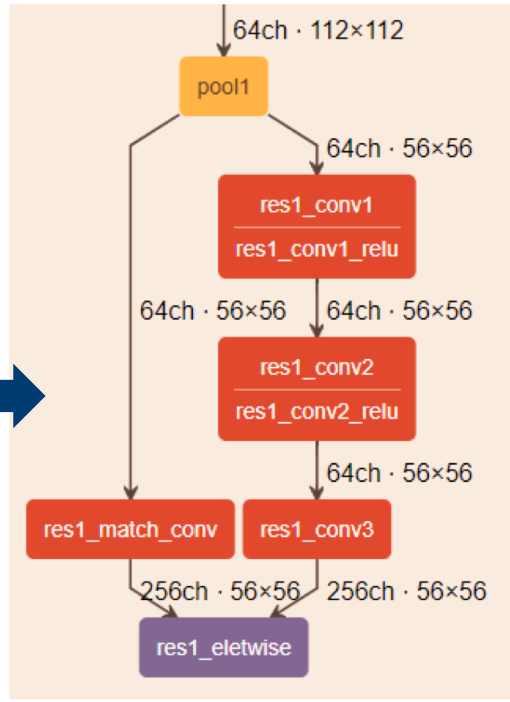
Specific optimization that applies for TensorFlow* topologies. (Xception*)



Split->Convolutions->Concat block from TensorBoard*



Caffe Resnet269 block (from Netscope)

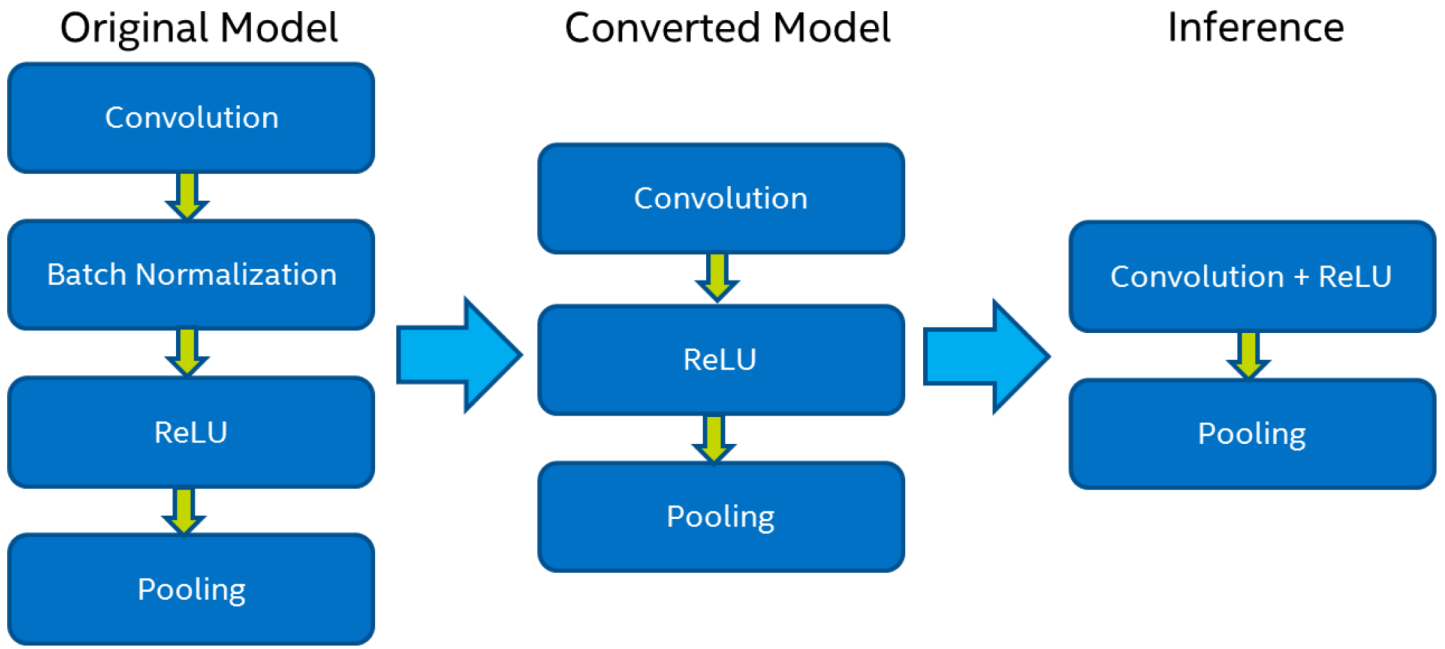


Fused Caffe* Resnet269 block (from Netscope*)

Model Optimizer (1 of 2)

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 (2 of 2)

Model optimizer can change the topology:

- Model optimizer can add normalization and mean operations, so some pre-processing is ‘added’ to the deep learning model.

  --mean_values (104.006, 116.66, 122.67)

  --scale_values (0.07, 0.075, 0.084)

Model optimizer can cut out a portion of the network:

- Model has pre/post-processing parts that cannot be mapped to existing layers.
- Model has a training part that is not used during inference.
- Model is too complex and cannot be converted in one shot.
- Use the --input and --output options.

Intermediate Representation (IR)

Squeezenet1.1.caffemodel → Squeezenet1.1.bin

Squeezenet1.1.prototxt → Squeezenet1.1.xml

```
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
  }
}
```

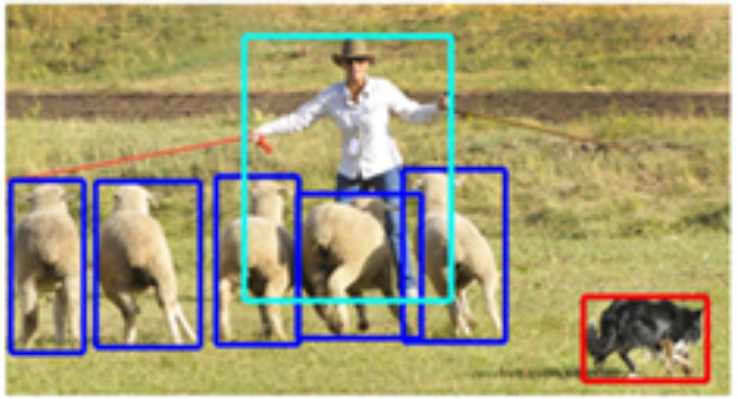
```
<net batch="1" name="model" version="2">
  <layers>
    <layer id="100" name="data" precision="FP32" type="Input">
      <output>
        <port id="0">
          <dim>1</dim>
          <dim>3</dim>
          <dim>227</dim>
          <dim>227</dim>
        </port>
      </output>
    </layer>
    <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>
```

Inference Engine

Inference on an Intel® Edge Systems

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

(a) classification



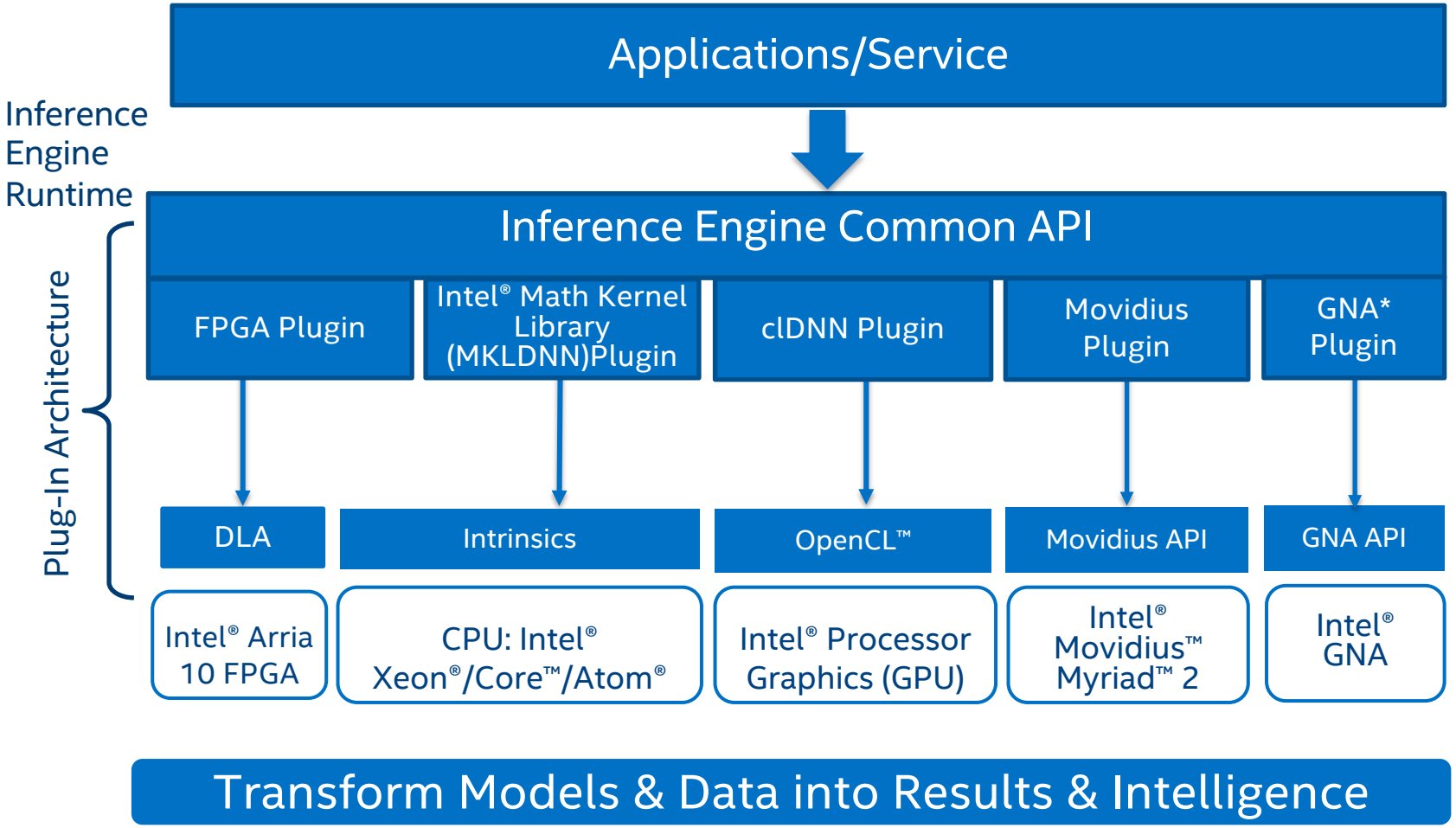
(b) detection

(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.

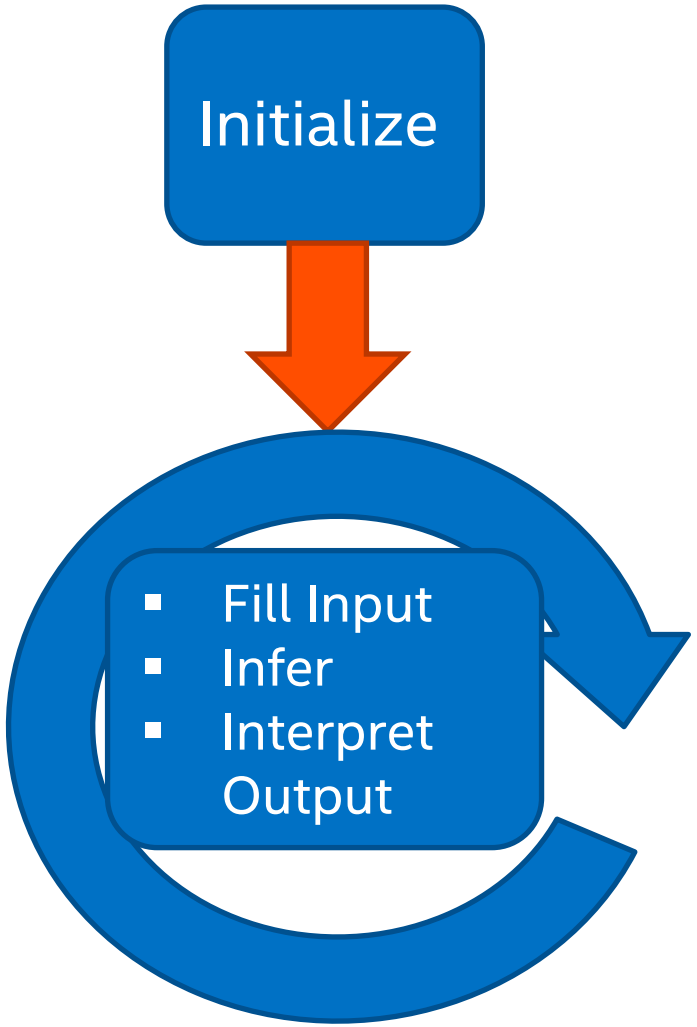
Optimal Model Performance Using the Inference Engine

- Simple & Unified API for Inference across all Intel® architecture
- Optimized inference on large IA hardware targets (CPU/GEN/FPGA)
- Heterogeneity support allows execution of layers across hardware types
- Asynchronous execution improves performance
- Futureproof/scale your development for future Intel® processors



GPU = Intel CPU with integrated graphics/Intel® Processor Graphics/GEN
GNA = Gaussian mixture model and Neural Network Accelerator

Workflow



Initialization

- Load model and weights
- Set batch size (if needed)
- Load inference plugin (CPU, GPU, FPGA)
- Load network to plugin
- Allocate input, output buffers

Main Loop

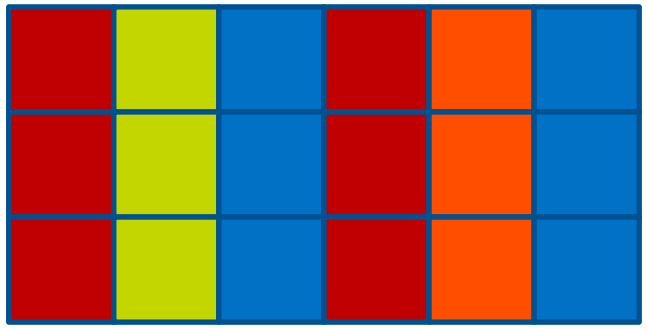
- Fill input buffer with data
- Run inference
- Interpret output results

Pre-processing

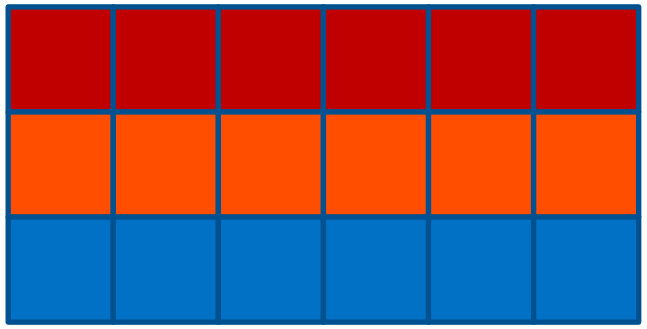
Most of the image formats are interleaved formats (RGB, BGR, BGRA, and so forth). Inference engine expects input to be in RGB planar format, such as:

R-plain, G-plain, B-plain

Interleaved



Planar



Post-processing

Developers are responsible for parsing inference output.

Many output formats are available. Some examples include:

- **Simple Classification (alexnet*):** an array of float confidence scores, # of elements=# of classes in the model
- **SSD:** Many “boxes” with a confidence score, label #, xmin,ymin, xmax,ymax

Unless a model is well documented, the output pattern may not be immediately obvious.

CPU/GPU/FPGA/Intel® Movidius™ Neural Compute Stick

Hardware Heterogenity

Heterogeneous Dichotomies

Data vs. Task Parallelism

- Data parallelism between devices is usually explicit and extremely error-prone (like in OpenCL™).
- Inference engine heterogeneity support is inherently **task-oriented** (node-level).

Dynamic vs. Static

- **User-defined work split** vs. adaptive schemes (like load-balancing).
- For both, there is a question on the granularity (communications costs, next).

***Heterogeneity* is basically *Fallback*.**

Apply Device Affinities to Layers Automatically Using Fallback Policy (1 of 2)

```
$ object_detection_sample_ssd -d HETERO:CPU,GPU -l
lib/libicv_extension.so -m ssd.xml -i snake.bmp
```

All IE samples support loading CPU and GPU extensions as usual ("-l" and "-c"). The "priorities" defines a greedy behavior:

- Keeps all layers that can be executed on the device (FPGA)
- Carefully respects topological and other limitations
- Follows priorities when searching (for example, CPU)

Apply Device Affinities to Layers Automatically Using Fallback Policy (2 of 2)

```
HeteroPluginPtr plugin(make_plugin_name("HeteroPlugin"));

CNNNetReader reader;

reader.ReadNetwork(“Model.xml”);

reader.ReadWeights(“Model.bin”);

CNNNetwork network = reader.getNetwork();

plugin->SetConfig({ { "TARGET_FALLBACK", “FPGA,CPU"} }, &response);

plugin->LoadNetwork(exeNetwork, network,{}, &response);
```

Intel® Movidius™ Neural Compute Stick

Core and Visual Computing Group

Intel® Movidius™ Neural Compute Stick

Redefining the AI Developer Kit

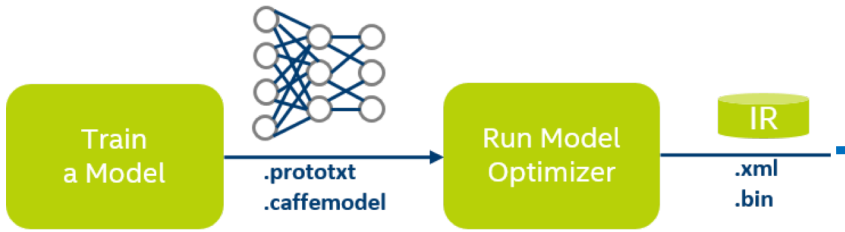
- Neural network accelerator in USB stick form factor
- Tensorflow* and Caffe* and many popular frames are supported
- Prototype, tune, validate, and deploy deep neural networks at the edge
- Features the same Intel® Movidius™ Vision Processing Unit (VPU) used in drones, surveillance cameras, virtual reality headsets, and other low-power intelligent and autonomous products

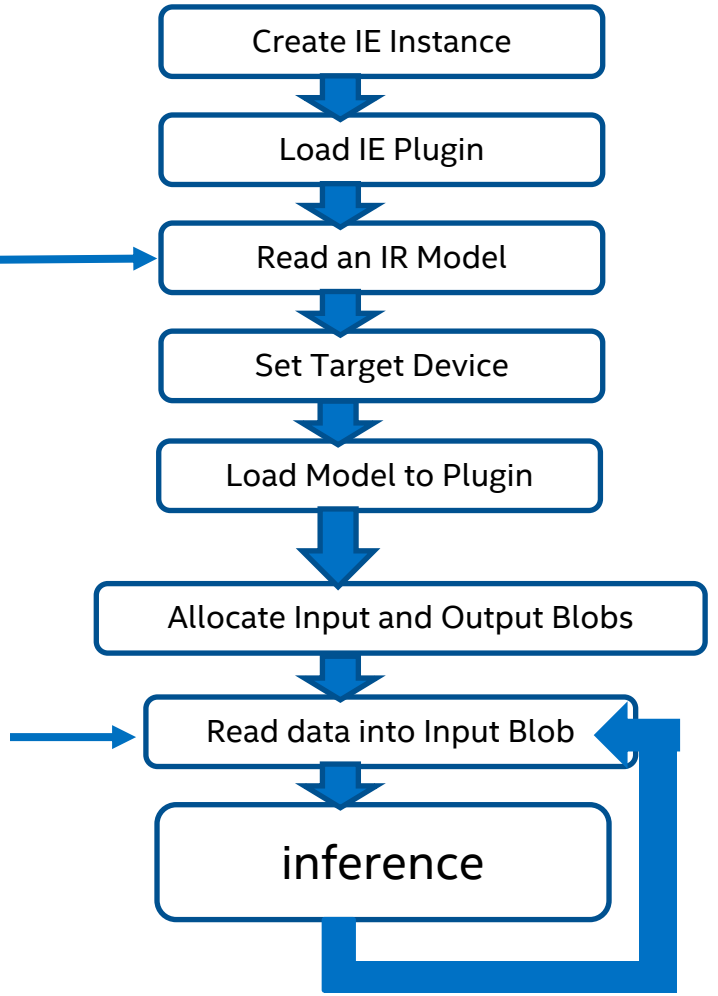
Car Detection

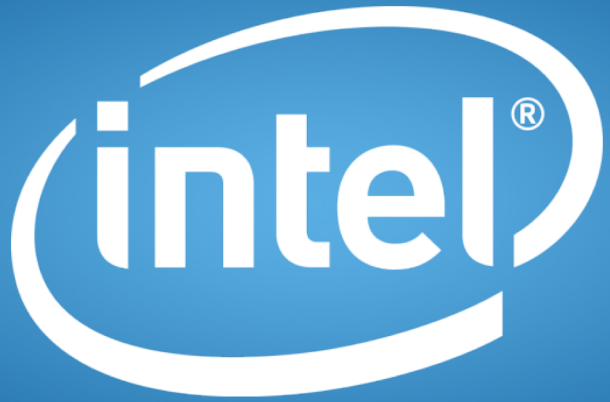
Hands-On Example

Offline, One-Time Run of the Model Optimizer

- Integrate inference engine (IE) API into the application.
- Intel® MKL-DNN plugin for CPU, Cl-DNN for GPU.
- Target device could be CPU/GPU/Intel® Movidius™ NCS/FPGA.






intel

Workshop hands-on exercise

https://github.com/intel-iot-devkit/smart-video-workshop

OR

Copy from the flash drive handed over in the workshop room