ACCELERATE DEEP LEARNING INFERENCE USING INTEL TECHNOLOGIES

OPTIMIZATION: TOOLS AND TECHNIQUES

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 (1 of 2)

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.

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

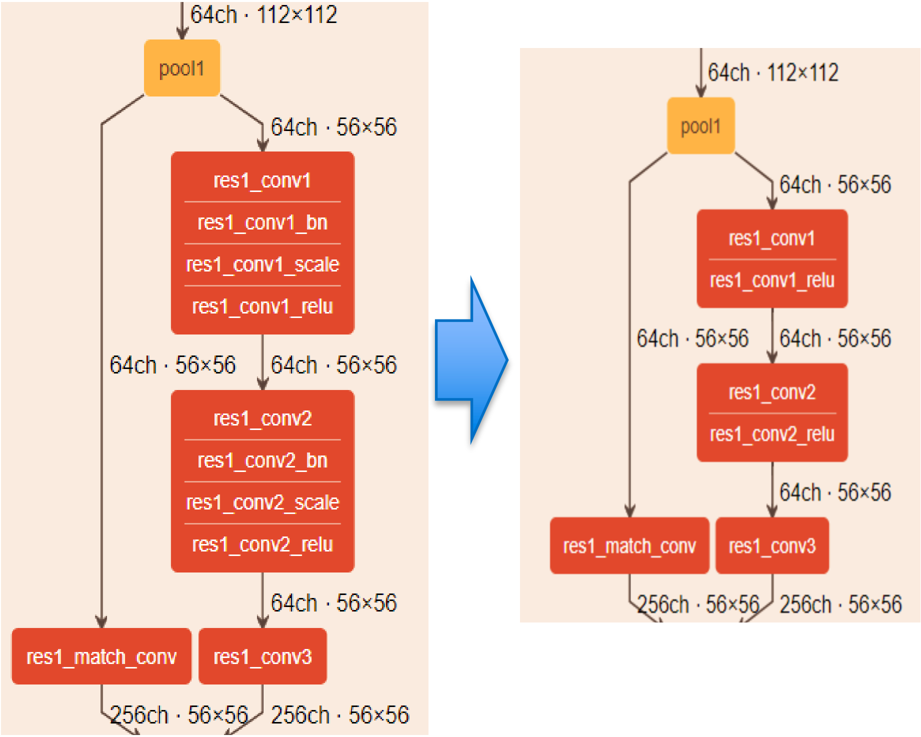
Optimizing Inference Applications

- Model optimizer: tune parameters.
- Use/train a model with the right performance/accuracy tradeoffs. Performance differences between models can be bigger than any optimization you can do at the inference app level.
- Use an optimized inference implementation. Performance difference between using the inference engine (IE) and a non-optimized framework can be bigger than differences between accelerators.
- Use the right data type for your target hardware and accuracy needs.
- Use async.
- Don’t infer every frame if not needed.
- Use Intel® VTune™ Analyzer.
- Optimize the whole pipeline.

Model Conversion with Model Optimizer

General topology-level opts are automatic. User knobs:

- Precision
- Batch (SSD!)
  - --scale and --mean_values
  - --reverse_input_channels



1. Pick the Right Model

1. Use/Train a Model with the Right Performance plus Accuracy Tradeoffs.

Performance is based on many factors:

- Topography complexity/layer implementation plus scheduling
- Number of color channels (that is, BGR vs. grayscale)
- Model resolution

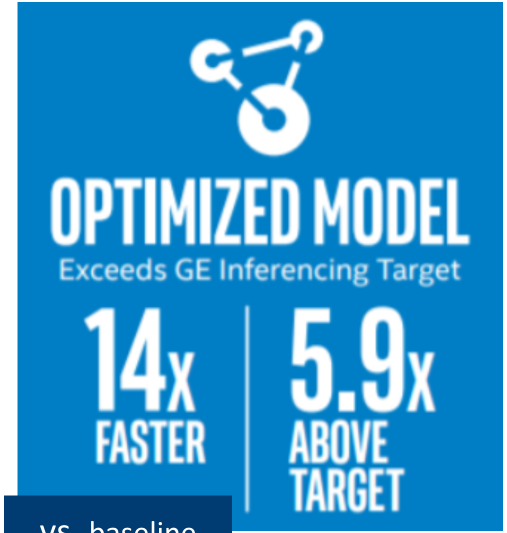
Exercise: Range of Model Performance

Complete this table in the hands-on exercise:

	CPU ms/frame	GPU ms/frame	Intel® Movidius™ Myriad™ ms/frame
ssd512			
ssd300			
Mobilnet-ssd*			

2. Use Inference Engine

2. Use an Optimized Inference Implementation



vs. baseline

https://ai.intel.com/wp-content/uploads/sites/53/2018/03/IntelSWDevTools_OptimizeDLforHealthcare.pdf

The performance difference between running inference engine vs. unoptimized can be bigger than the difference between accelerators.

Inference Engine Samples: Measure Correctly

- Beware of the one-time costs (OpenCL™ kernels compilation)
- Measure infer and other items (pre-processing, decoding) separately
- Multiple iterations (“-ni”)!
- Batching (“-i”) !
- Performance counters (“-pc”)
- Device (“-d”, and “-d_...”)
- Latency vs. throughput (asynchronous case)
- Classification(_async)/ObjectDetectionSSD samples

Inference Engine, CPU-specifics

Example “-pc” output from the sample (notice the helper layers)

conv1	*EXECUTED*	*layerType:* ***Convolution***	*realTime: 706*	*cpu: 706*	*execType:* ***jit_avx2***
conv2_1_x1	*EXECUTED*	*layerType: Convolution*	*realTime: 137*	*cpu: 137*	*execType: jit_avx2_1x1*
fc6	*EXECUTED*	*layerType: Convolution*	*realTime: 233*	*cpu: 233*	*execType: jit_avx2_1x1*
fc6_nChw8c_nchw	*EXECUTED*	*layerType:* ***Reorder***	*realTime: 20*	*cpu: 20*	*execType: reorder*
out_fc6	*EXECUTED*	*layerType: Output*	*realTime: 3*	*cpu: 3*	*execType: unknown*
relu5_9_x2	***OPTIMIZED_OUT***	*layerType: ReLU*	*realTime: 0*	*cpu: 0*	*execType: undef*

Custom Layers

- Examples in the “extension” dir in the samples folder
- Few flavors (SSE/AVX2/AVX512)
- Watch for outstanding Reorders, and use blocking format (nchw_8c)
- Use OpenMP* as parallel engine

Inference Engine, GPU-Specifics

Prefer fp16 over fp32

- MO can generate both variants and the fp32 is default

Use batches (SSD!)

OpenCL™ kernels compilation overhead

Custom Kernels

- Before implementing a full-blown code, estimate the final performance by stub ("infinitely" fast) kernel that does nothing
- Merge chain of custom kernels into a super-kernel
- Only heavy layers are worth offloading (plus “glue” layers)
- Don’t impl lightweight layers for GPU, keep them on the CPU

3. Use the Correct Data Type

3. Use the Correct Data Type

	FP32	FP16
CPU	yes	no
GPU	yes	recommended
Intel® Movidius™ Myriad™	no	yes
FPGA/DLA	no	yes

FPGA/DLA also supports FP11.

4. Use async

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.
- The demo keeps two parallel infer requests, and, 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)`.

Object Detection Sample SSD async

$ ./object_detection_demo_ssd_async -h

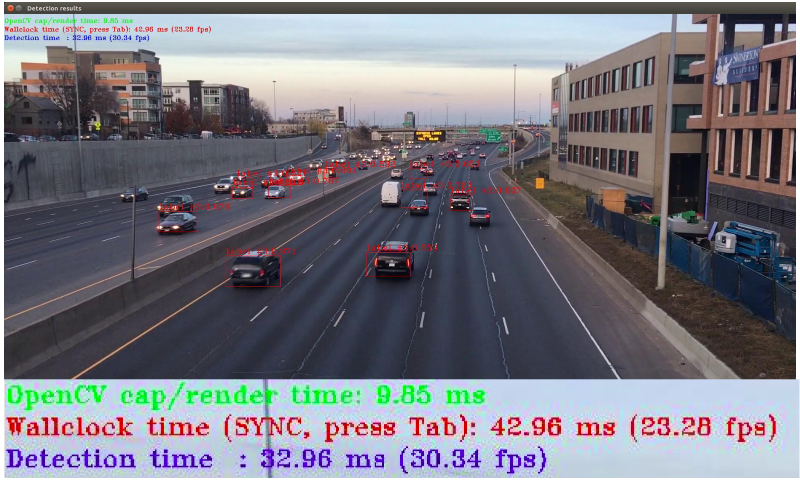
```
async_object_detection_ssd [OPTION]
Options:

    -h                          Print a usage message.
    -i "<path>"                 Required. Path to an video file (specify "cam" to work with camera).
    -m "<path>"                 Required. Path to an .xml file with a trained model.
      -l "<absolute_path>"        Required for MKLDNN (CPU)-targeted custom layers.Absolute path to a shared library with the kernels i
mpl.
          Or
      -c "<absolute_path>"        Required for clDNN (GPU)-targeted custom kernels.Absolute path to the xml file with the kernels desc.
    -d "<device>"               Specify the target device to infer on (CPU, GPU, FPGA, or MYRYAD. Sample will look for a suitable plugi
n for device specified
    -pc                         Enables per-layer performance report.
    -r                          Inference results as raw values.
    -t                          Probability threshold for detections.
```

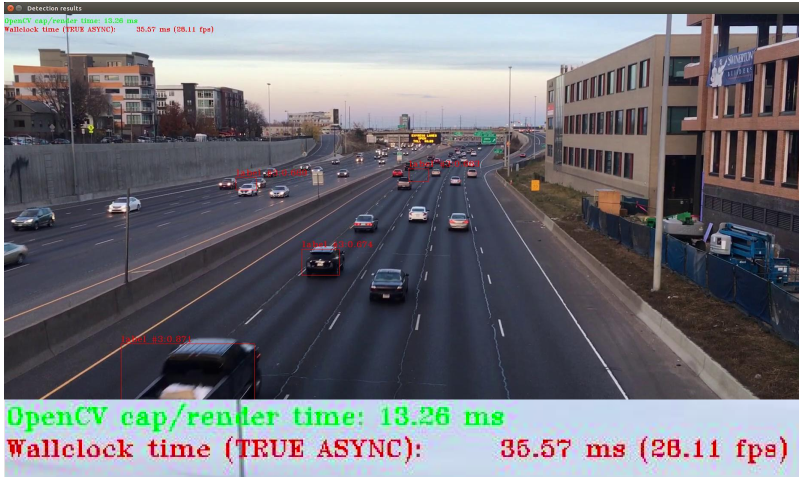
Running Object Detection Sample SSD async

```
$ ./object_detection_demo_ssd_async -i /home/intel/workshopApr25/workshop-
tutorials/test_content/video/cars_1920x1080.h264 -m /home/intel/workshopApr25/workshop-
tutorials/test_content/IR/SSD/SSD_GoogleNet_v2_fp32.xml
```

Synchronous Mode



Press **Tab** to Enable Asynchronous Mode



5. Don’t Infer If Not Needed

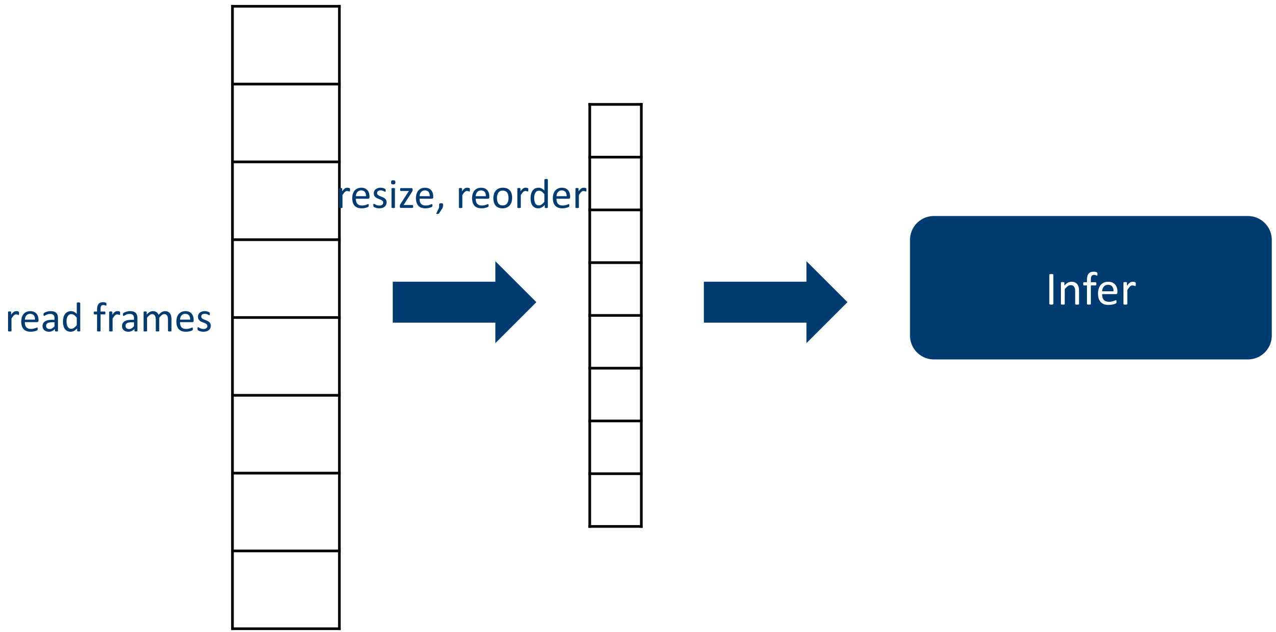
Tracking

Many ways to track:

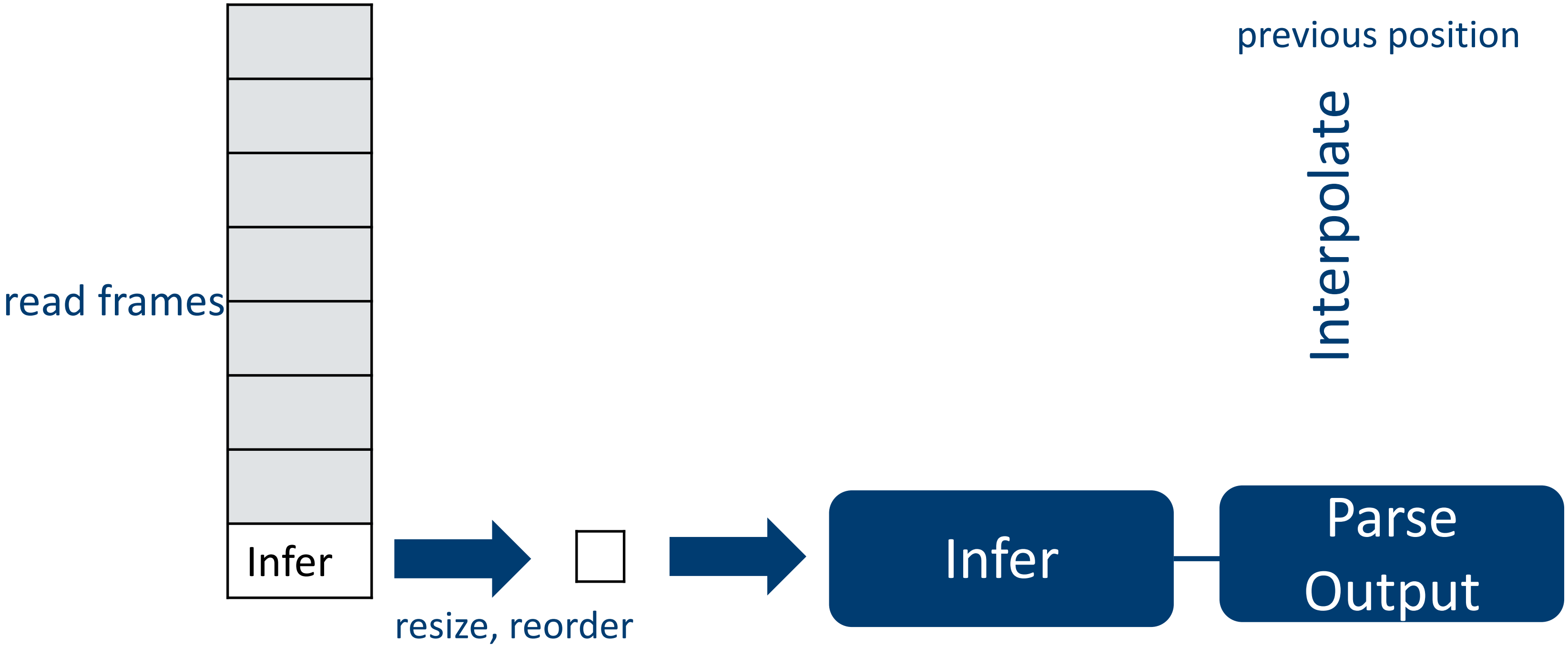
- Optical flow
- Kalman filters
- Single/simplistic tracking (what we show here)
- Multiple object tracking: like above but find ways to identify each region so rectangles have a persistent identity over time

A Tracking Thought Experiment

Processing in batches already introduces latency



What If We Did Less Inference?



Determine If There Is Nothing to See

Inference is expensive to run each frame. It can save time to not run when there is nothing to identify.

- Check motion vectors
- Frame sizes
- bgsubmog
- SAD

These methods can be several orders of magnitude less expensive than inference. Use techniques to increase the total # of streams a system can watch.

ADVANCED VIDEO ANALYTICS

Core and Visual Computing Group

Video Analytics in Intel® Distribution of OpenVINO™ Toolkit

Topology	Type	Description
age-gender-recognition-retail-0013	object_attributes	Age and gender classification. Used in Intel® Audience Analytics.
face-detection-retail-0004	detection	Face detection (SqNet1.0modif+single scale) without BatchNormalization trained with negatives.
person-detection-retail-0001	detection	Person detection (HyperNet plus RFCN). Used in Intel® Audience Analytics.
license-plate-recognition-barrier-0001	ocr	Chinese license plate recognition.
face-detection-adas-0001	detection	Face detection (MobileNet* with reduced channels plus SSD with weights sharing).
person-detection-retail-0012	detection	MobileNet* plus single SSD minus cluster. Used in Intel® Audience Analytics.
head-pose-estimation-adas-0001	headpose	Vanilla CNN trained from scratch yaw plus pitch plus roll plus landmarks.
vehicle-attributes-recognition-barrier-0010	object_attributes	Vehicle attributes recognition with modified RESNET10* backbone.
person-vehicle-bike-detection-crossroad-0066	detection	Multiclass (person, vehicle, non-vehicle) detector based on SSD detection architecture, RMNet* backbone and learnable image downscale block.
vehicle-license-plate-detection-barrier-0007	detection	Multiclass (vehicle, license plates) detector based on RESNET10 plus SSD.

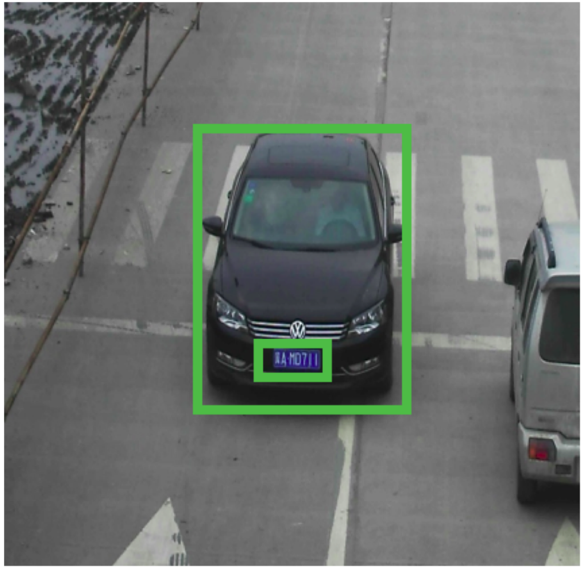
vehicle-attributes-recognition-barrier-0010 Use Case/High-Level Description

Vehicle attributes classification algorithm for a traffic analysis scenario.

Type: regular
Color: black

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.



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.

person-detection-retail-0001
Use Case/High-Level Description

Pedestrian detector based on backbone with hyper-feature plus R-FCN for retail scenario.

ADVANCED VIDEO ANALYTICS

SECURITY BARRIER DEMO

Security Barrier Demo

Load Input Image(s)

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

Detects Vehicles

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

Classifies vehicle attributes

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

Detects License Plates

Display Results