DEEP LEARNING INFERENCE WITH INTEL® FPGA

Class 2

What We Discussed in Previous Lesson

Deep Learning (DL) is a type of machine learning for extracting patterns from data using neural networks

DL neural networks are built and trained using frameworks and combining various layers

FPGAs are made up of a variety of building blocks that using FPGA development tools will translate code into custom hardware

FPGAs provide a flexible, deterministic low-latency, high-throughput, and energy-efficient solution for accelerating the constantly changing networks and precisions for DL inference

Agenda

Computer Vision (CV) Primer

- Why CV is needed
- CV algorithm basics
- Components of Vision Systems Architecture

Where Inferencing fits into CV

Building a deep learning computer vision application

- Different programming languages, hardware and tools

Objectives

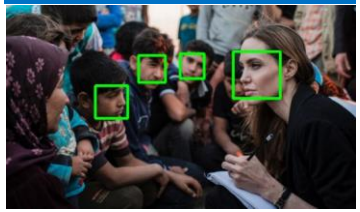
You will be able to:

- Define the stages of a computer vision application
- Explain where deep learning inference can be used and how it is relevant for computer vision applications
- Define the programming languages and hardware platforms available to create computer vision applications
- Relate to the advantages of FPGA acceleration of deep learning computer vision applications

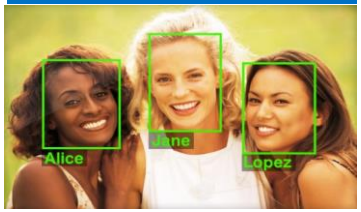
Computer Vision Enables Amazing New Applications

PEOPLE DETECTION

FACE DETECTION



FACE RECOGNITION



OBJECT DETECTION & CLASSIFICATION

RETAIL INTELLIGENCE

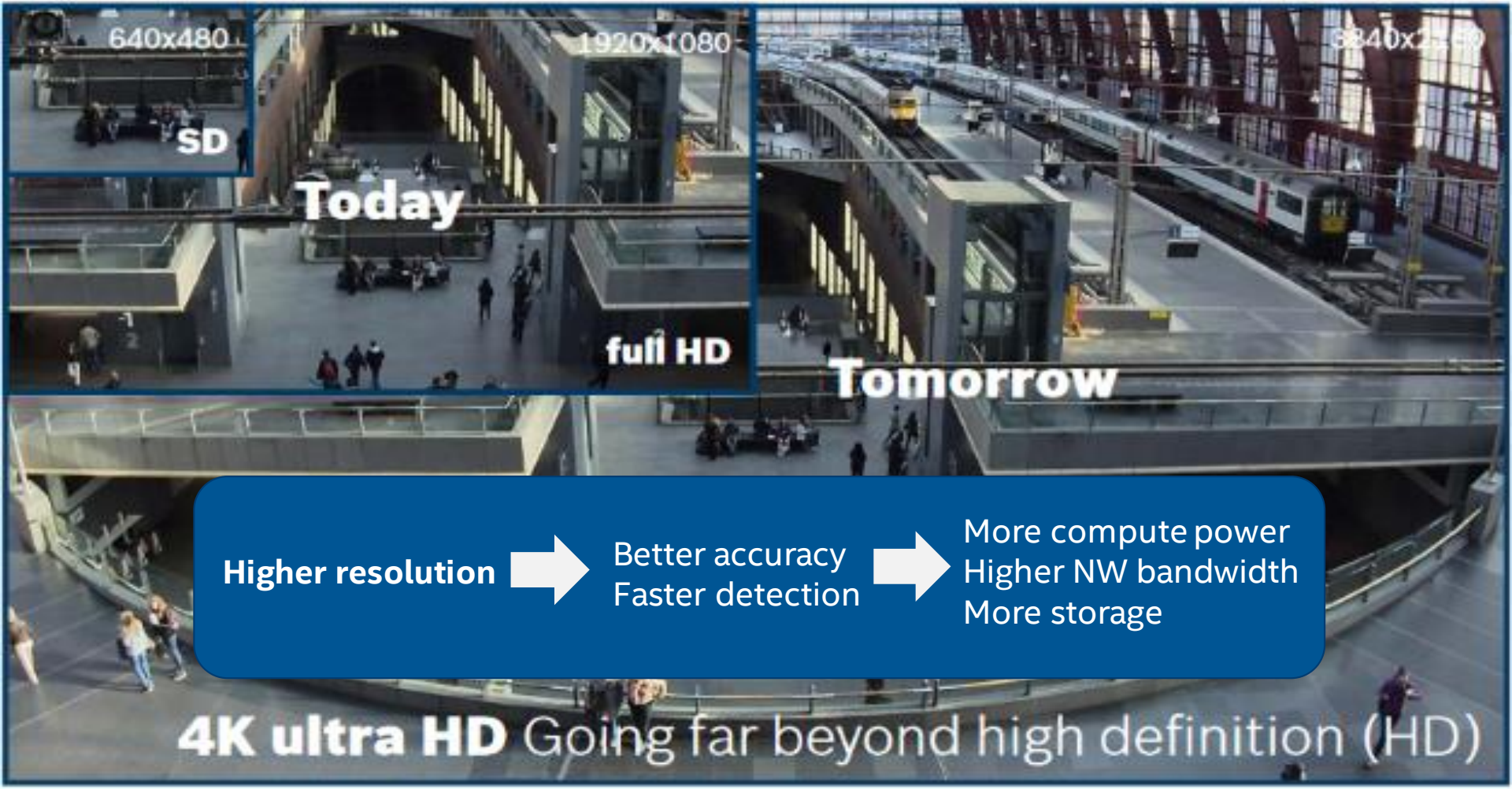
VEHICLE DETECTION

VEHICLE CLASSIFICATION

AUTOMOTIVE

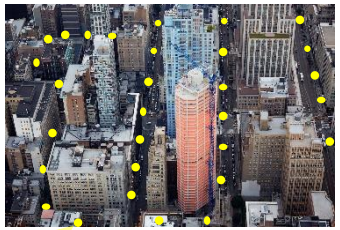
INDUSTRIAL INSPECTION

Use of video, computer vision and deep learning is growing rapidly across industries


640x480
SD
1920x1080
Today
full HD
Tomorrow
Higher resolution
Better accuracy
Faster detection
More compute power
Higher NW bandwidth
More storage
4K ultra HD Going far beyond high definition (HD)

Video Surveillance Problem

Proliferation of Surveillance Cameras And Sensors



Explosion of Data
(especially video)

1.6 Exabytes/Day by 2020

Inefficiencies and Increased Cost

Human Monitoring

- Effective for only 20 min
- Slow to react
- Expensive

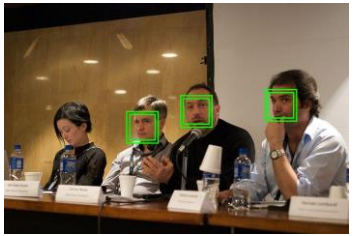
Distribution

- Network Overload

Storage

- Storage growing @30%

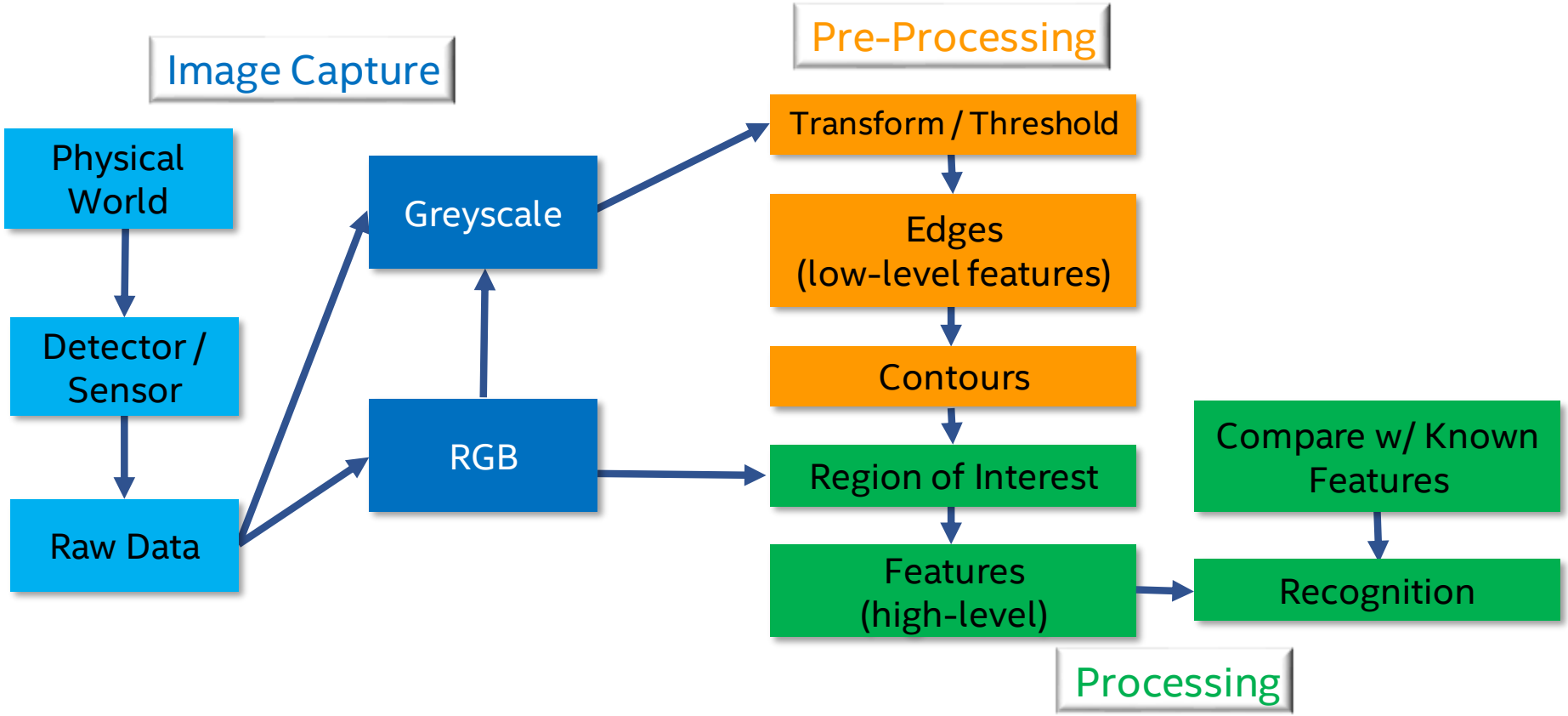
Intel Is Removing Obstacles



Video Analytics accelerated by Intel® FPGAs

- Eliminate need for human operators
- Delivers deterministic latency orders of magnitude lower
- Delivers accurate results 24/7/365
- Reduces storage and network requirements by orders of magnitude
- Provides a flexible architecture for next generation mathematical approaches

Visual System Architecture



Characteristics of the Physical World

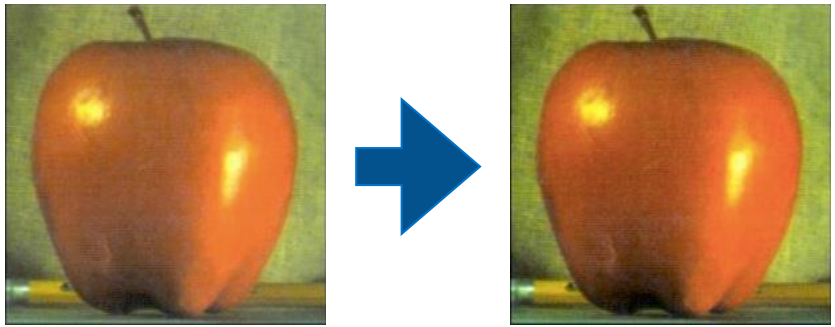
What light is getting to the image capture equipment?

- Color
- Brightness
- Intensity
- Light scatter

Pre-Processing

Modern cameras use software to perform many pre-processing steps on images captured:

- Sampling data received to create image of appropriate resolution
- Denoising to increase sharpness
- Correcting contrast to increase clarity
- Scale correction
- Extraction of low-level features



Detection and Segmentation of Image Components

Define region of interest

Scene segmentation - divide image into objects at the pixel level

High-Level Processing

Counting and identifying attributes of objects of a type

- Using object size, object position, or other feature and assigning these to specific categories
- Counting attributes of specific object types
  - object size/position, objs → categories

Example CV system could identify that:

- There are 20 birds in this image:
  - 5 Ducks
  - 15 Pigeons

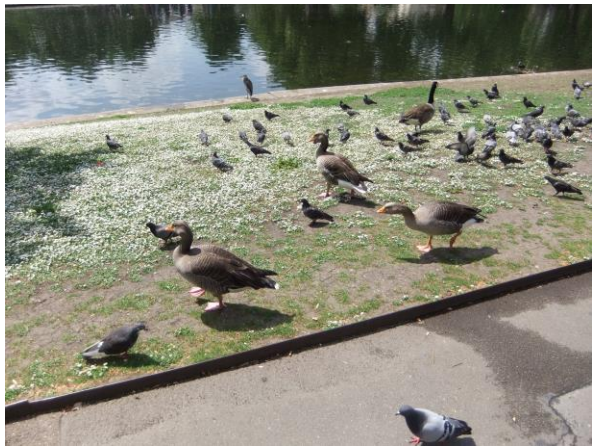
High-Level Processing

“Image understanding” (Wikipedia)

- Low level includes image primitives such as edges, texture elements, or regions;
- Intermediate level includes boundaries, surfaces and volumes
- High level includes objects, scenes, or events.
- LEARNING SYSTEM is implemented here

Examples:

- Intermediate level learning:
  - “This image has grass, water, and concrete surfaces.”
- High-level learning:
  - “Birds are sitting near a pond in Regent’s Park, London.”



High-Level Processing Gives the Data Context

If This is a Chair…

Blurry too …

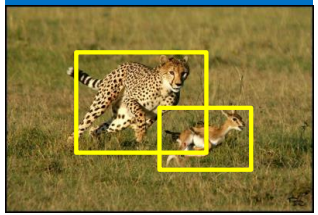
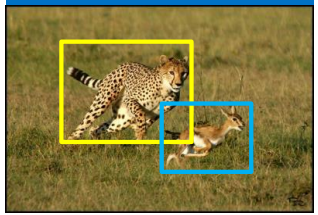
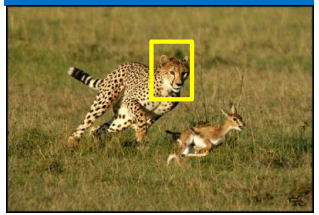
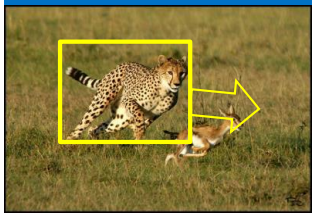
And this is a chair…

And this mirrored image…

High-Level Processing Gives the Data Context

But, should a self driving car see these the same?

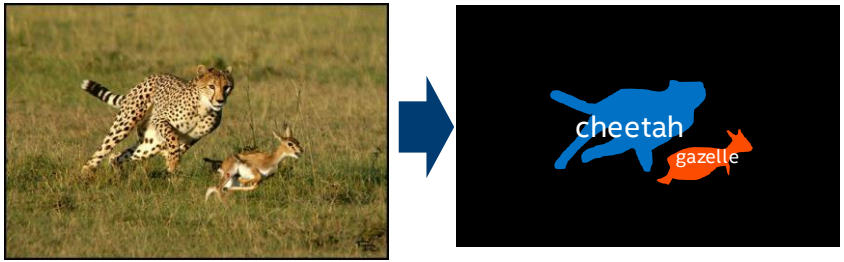
CV Algorithm Common Tasks

Segment	Detect	Classify	Recognize	Track
				
Background/Object	Animals	Cheetah, Gazelle	Sandy T. Cheetah	East @ 40 mph

Segmentation

Semantic Segmentation

- Label each pixel of an image by the object class

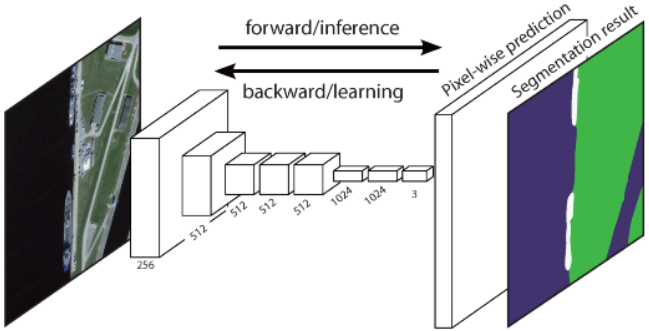


Instance Segmentation

- Label each pixel of an image by the object class and object instance

Common CNN segmentation networks

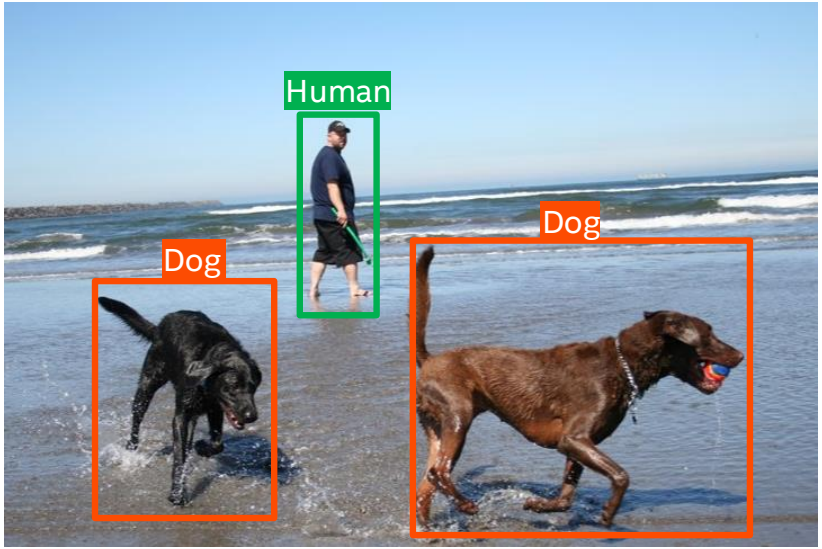
- Fully-Convolutional Net (FCN)
- SegNet



Object Detection

Localize and classify all objects in the image

- Common CNN Networks:
  - YOLO, Single Shot MultiBox Detector (SSD), Faster-RCNN, etc
  - Outputs
    - Bounding Box
    - Object Class
    - Possibly Confidence Score



Object Localization

Predict the region that contains the dominant object

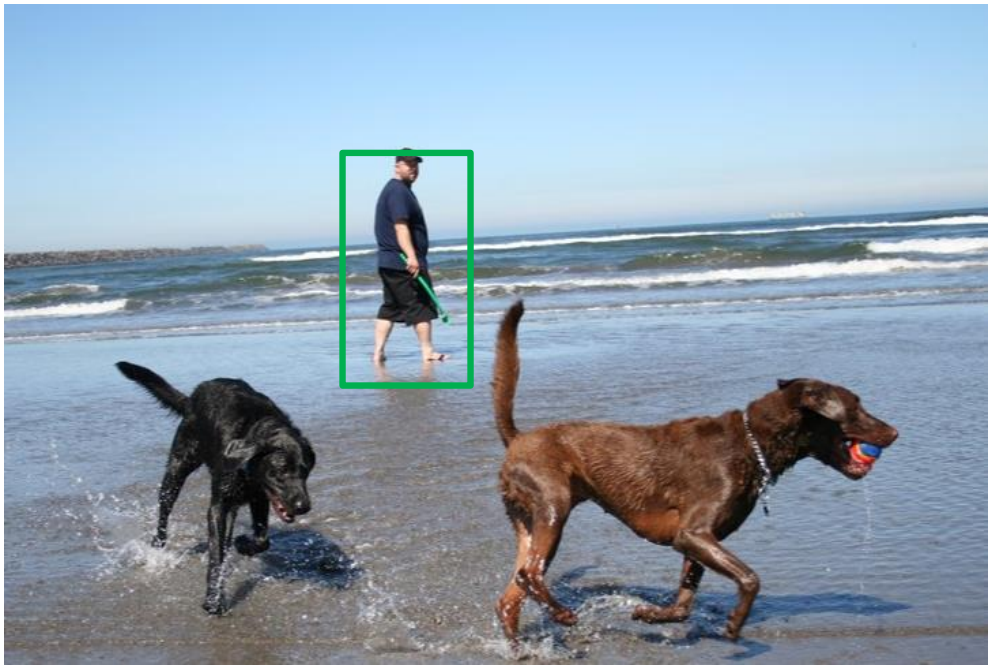


Image Classification

Classify an imaged based on the dominant object inside it

99.572% bald eagle
0.247% kite
0.033% limpkin
0.026% vulture
0.026% albatross

58.702% pickup
21.595% convertible
4.819% sports car
3.753% car wheel
3.753% grille

Common CNN Networks for Classification

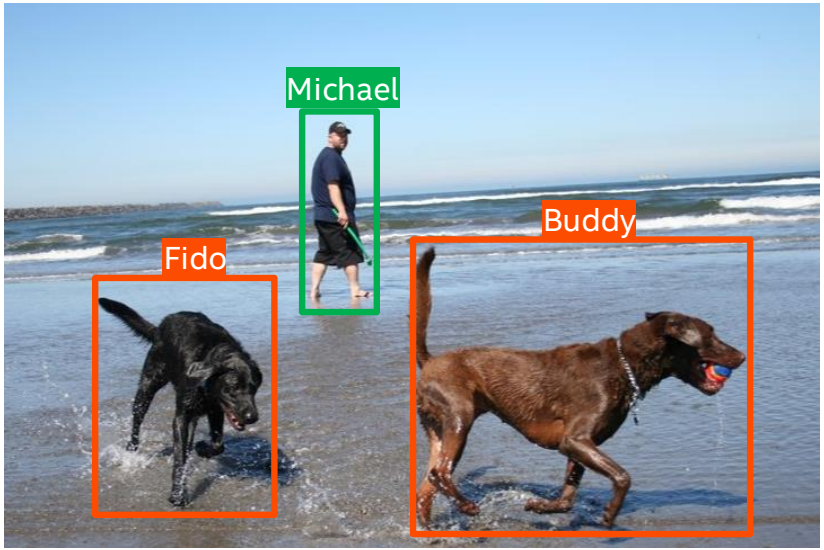
- AlexNet, GoogLeNet, SqueezeNet, VGG, MobileNet, ResNet, etc…
  - For each image, outputs probability for every class

Face Recognition

Maps detected face with a stored reference

Common CNN Networks:

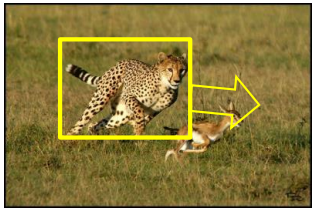
- CNNs with ability to differentiate face features e.g., SphereFace



Other Tasks

Object Tracking

- Locating object in successive video frames



East @ 40 mph

Optical Character Recognition (OCR)

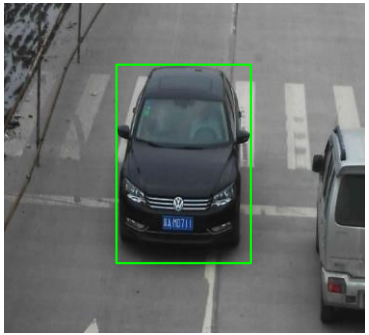
- Object Detection for numbers and characters

Keypoint Detection

- Detect locations of a set of predefined keypoints of an object
  - i.e., human body, face

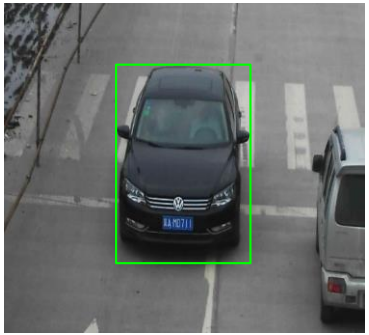
Different Applications Need Different Things

A Car

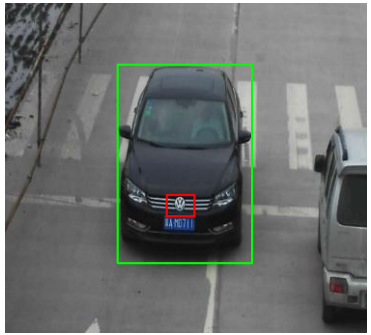


Car detection for autonomous driving

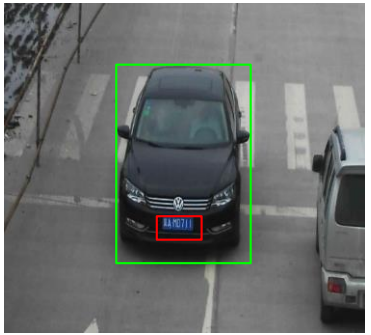
A black car



Volkswagen Passat

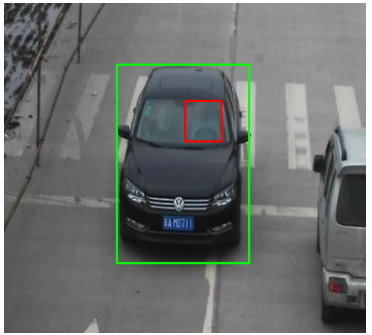


License plate number



Car detection for Smart City analytics, police tracking for a particular car, or authorization through a toll

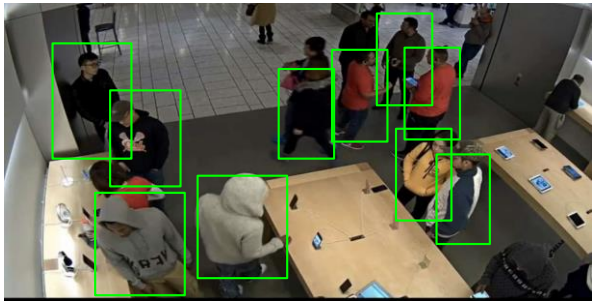
Not the owner !!



Recognition of potential stolen car

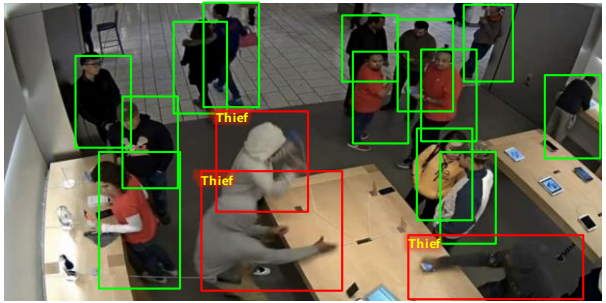
Different Applications Need Different Things

People Detection



People Tracking

Analyze behavior/ intentions



Smart City application in retail for marketing purposes

Detect Age and gender of people shopping for particular products or authorization of a particular person to use an ATM

Track shopping patterns of particular demographics

Asset Protection against potential thieves

Decode → Pre-Processing → Inference → Post-Processing → Encode

Encoding/Decoding

Decoding

- Decode video from source format to a format that can be easily processed

Encoding

- Convert/compress video to a format for display or other type of consumption
- H.265 (HEVC), H.264 (AVC), MPEG-2, etc…

Pre-Processing/Post-Processing

Pre-Processing

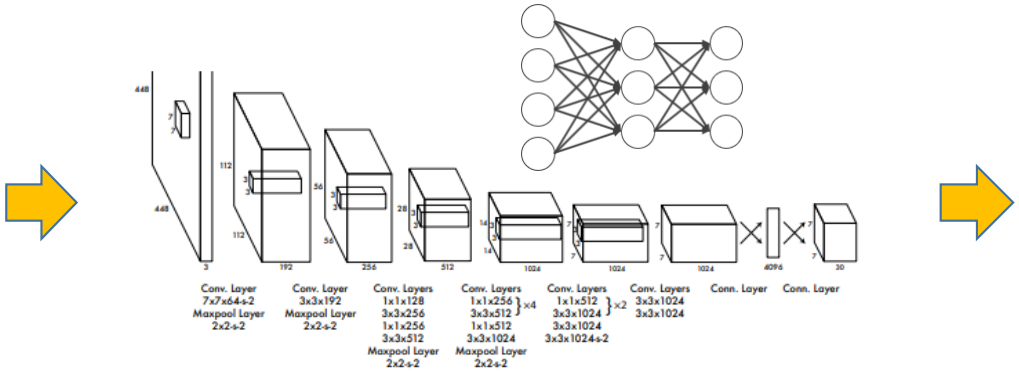
- Convert video format to one expected by the inference network
- Common tasks: resize, scale, deinterlace, color space conversions, de-noise, sharpen, etc…

Post-Processing

- Prepare inference output for application consumption
- Common tasks: draw rectangle around classified object to highlight it

Inference

Computationally-intensive process of executing a trained network on the input video/image data to find a pattern



Example of End-to-End Inference Application

License Plate Recognition at Toll Booth (3 inferences with 3 different networks)

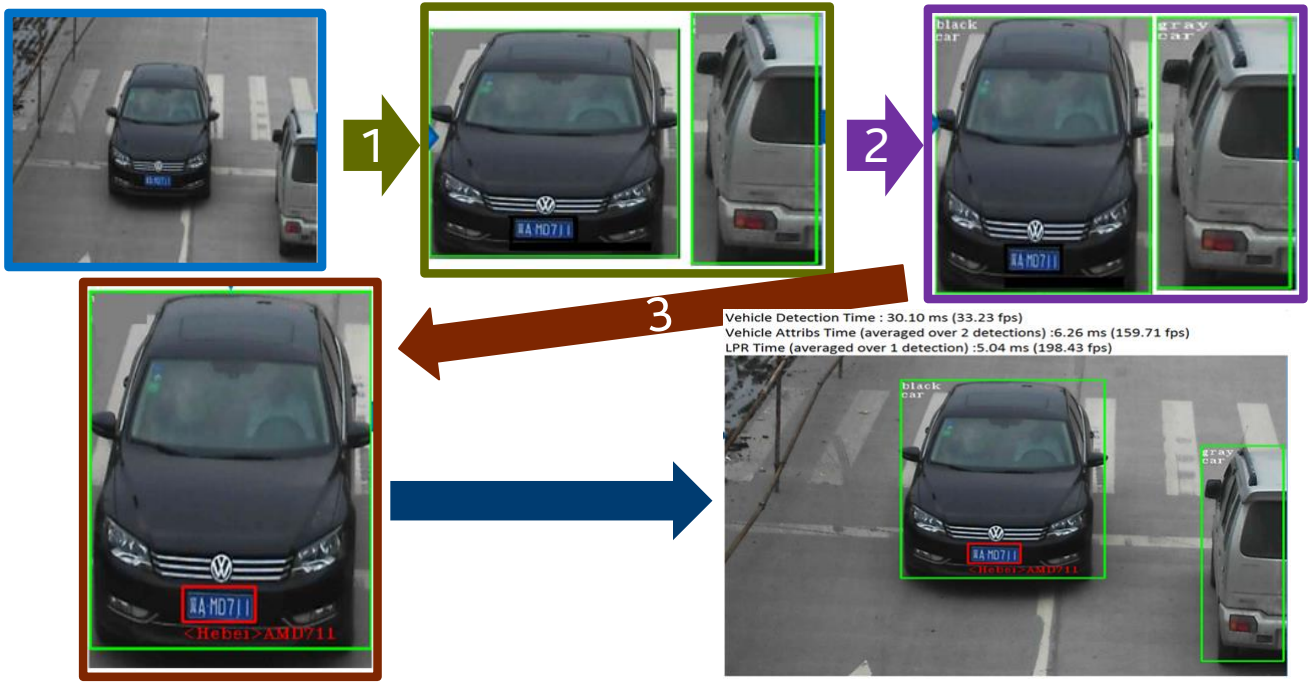
Input Image

Inference 1: Vehicle Detection

Inference 2: Vehicle Classification

Inference 3: Detects License Plate

Displays Result



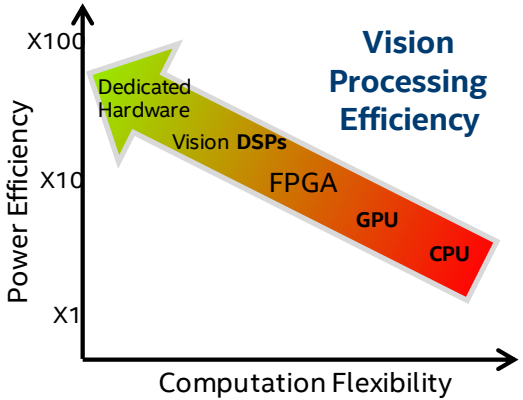
Different Languages and Hardware Platforms Available

Programming languages

- DL Frameworks for CNN topologies
- OpenCL*
- OpenCV
- OpenVX*

Hardware Platforms

- CPU
- FPGA
- GPU
- VPU
- Dedicated ASICs



Deep Learning Frameworks

Software framework for designing, training, validating, and deploying deep neural networks

Various frameworks designed for different purposes

Easy to use high-level programming interface

Drawbacks

- May not be fully optimized for each hardware accelerator
- Not lightweight since they provide end-to-end functionality
- Different frameworks are not interoperable

Open Computing Language (OpenCL™)

General purpose programming model for writing programs that execute across heterogeneous parallel platforms

- Provides increased performance with hardware acceleration on CPUs, GPUs, and FPGAs
- Host API and kernel language
  - Low-level programming language based on C/C++
- Open, royalty-free standard managed by Khronos® Group (Intel® leads group)
  - www.khronos.org

Open Source Computer Vision Library (OpenCV)

Free cross-platform portable library aimed at real-time computer vision

- 2500+ algorithms and functions
- C++, Python*, Java* and Matlab* interfaces
- Written in optimized C/C++ and enabled with OpenCL™

Low-level library providing building blocks for your application

- Functionality: image retrieval, pre-processing, feature extraction, segmentation, object detection, recognition, pose estimation, reconstruction, motion analysis, machine learning, etc…

Example OpenCV Program

Canny Edge Detector

```
#include "opencv2/opencv.hpp"

using namespace cv;

int main(int argc, char** argv)
{
      Mat img, gray;
      img = imread(argv[1], 1);
      imshow("original", img);

      cvtColor(img, gray, COLOR_BRGR2GRAY);
      GaussianBlur(gray, gray, Size(7, 7), 1.5);
      Canny(gray, gray, 0, 50);

      imshow("edges", gray);
      waitKey();
      return 0;
}
```

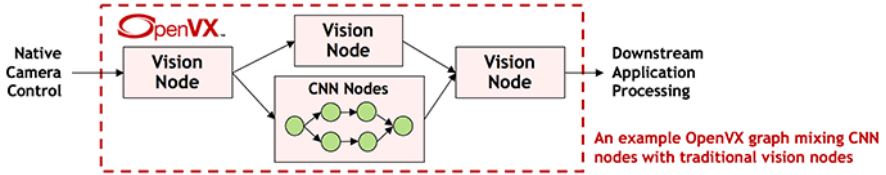
OpenVX™

Open, royalty free standard API managed by the Khronos® Group*

Contains library of predefined and customizable vision functions

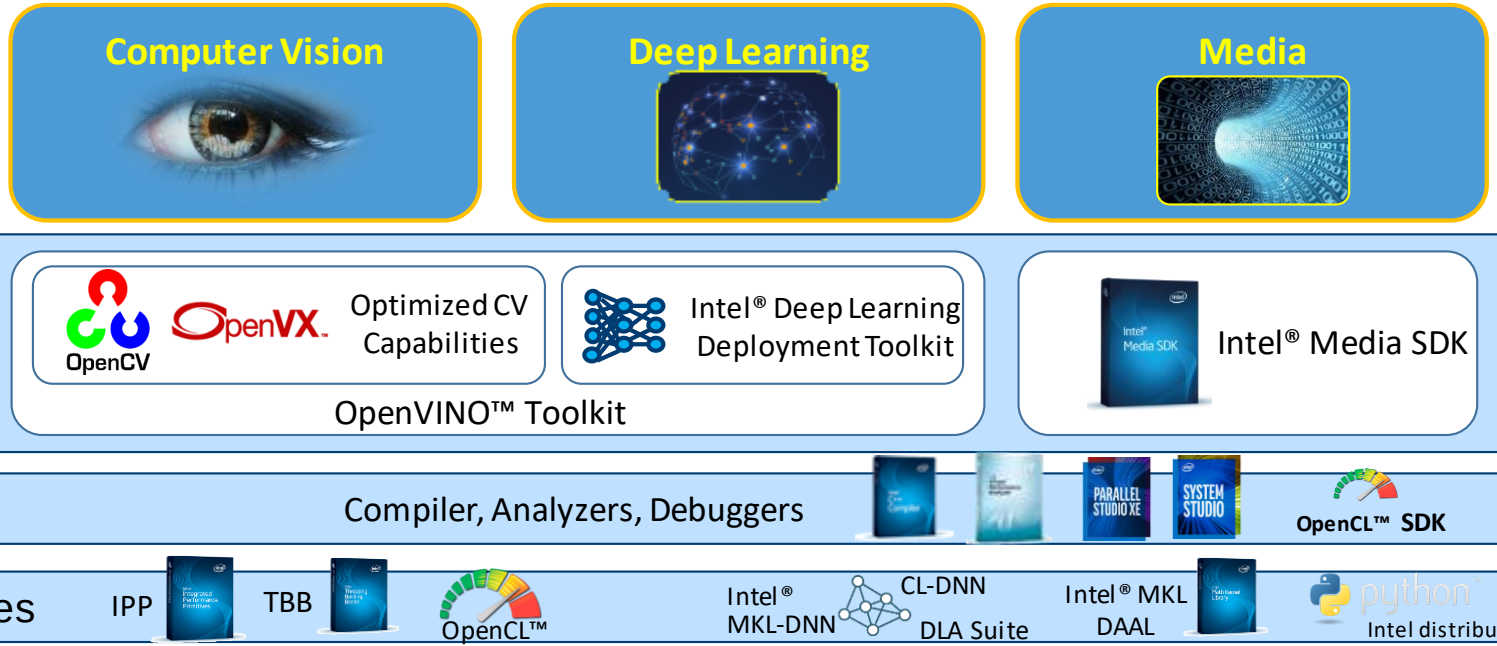
- E.g. CNN, Various Filters, Edge Detector, Color Convert, HOG, HoughLines, Magnitude, Mean and Standard Deviation, Scale etc…

C API for building, verifying, and coordinating graph execution (Nodes)

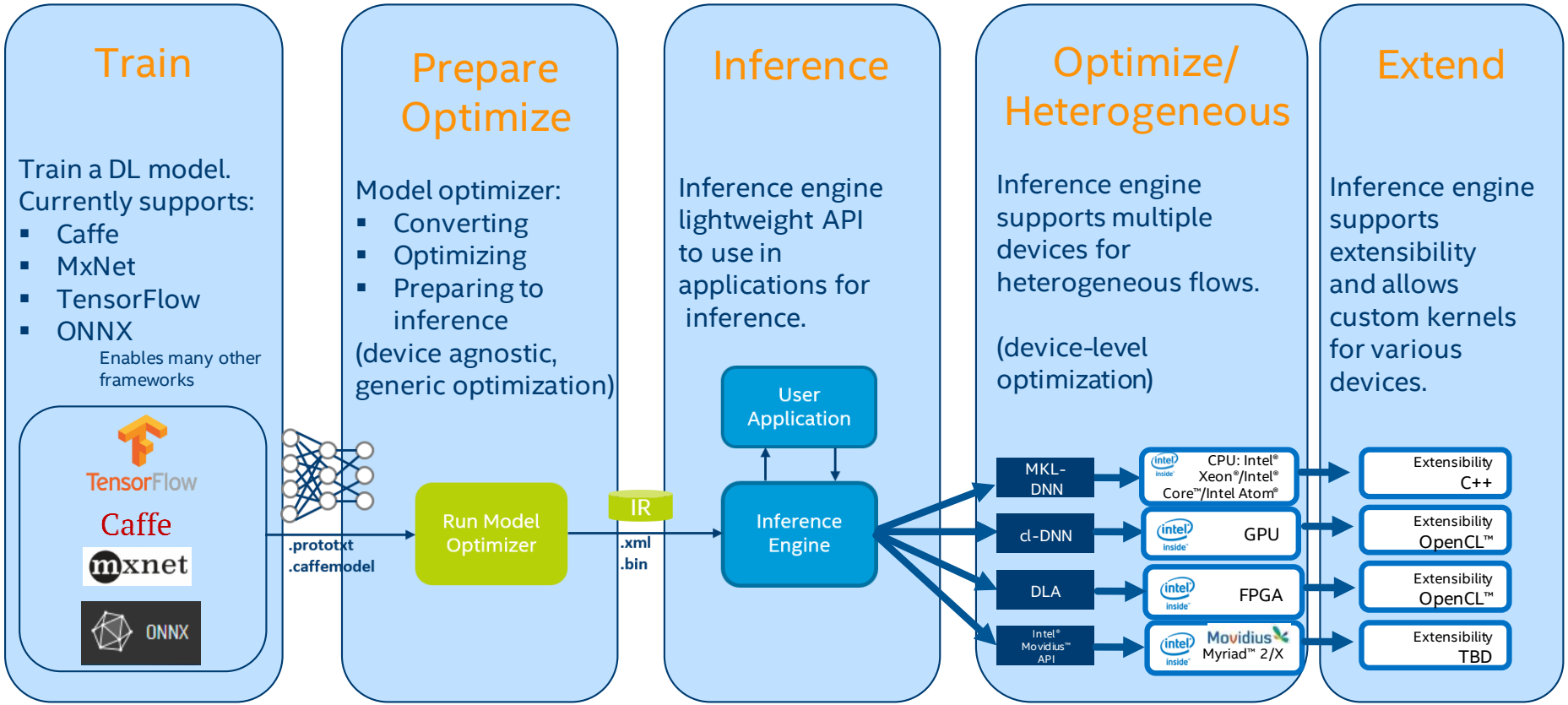


Putting It All Together

Intel® tools and libraries work together in the best possible way to minimize development time and optimize system power/performance while abstracting the hardware platform away



Application Development with OpenVINO™ Toolkit



Video Analytics Models Included with OpenVINO™

Topology	Type	Description
age-gender-recognition-retail-0013	object_attributes	Age & gender classification. Used in 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+RFCN). Used in Audience Analytics.
license-plate-recognition-barrier-0001	ocr	Chinese license plate recognition
face-detection-adas-0001	detection	Face Detection (MobileNet with reduced channels + SSD with weights sharing)
person-detection-retail-0012	detection	MobileNet + single SSD - cluster. Used in Audience Analytics.
head-pose-estimation-adas-0001	headpose	Vanilla CNN trained from scratch yaw + pitch + roll + 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+SSD

Many more coming soon

Intel® Media SDK

API to Access Intel® Quick Sync Video: Hardware Accelerated Encoding, Decoding, and Processing

- H.265 (HEVC)
- H.264 (AVC)
- MPEG-2 and more
- Resize, scale, deinterlace
- Color conversion, composition
- Denoise, sharpen, and more

Benefits

- Outstanding performance
- Rich API to tune encoding pipeline
- Future proofed: support new processor without code changes

Targeting Digital Security and Surveillance, Connected Car Applications, and More

Smart Camera

Car Infotainment and Cluster Display

using

Intel Atom®, Pentium®, and Celeron® [1]

and

Embedded Linux*

yocto PROJECT

[1] Intel® Celeron® Processor N3350, Intel® Pentium® Processor N4200, Intel Atom® E3930, E3940, E3950 processors

Video Pipeline Acceleration Hardware

Decode → Pre-Processing → Inference → Post-Processing → Encode

Decode	Pre-Processing	Inference	Post-Processing	Encode
CPU	CPU	CPU	CPU	CPU
GPU	GPU	GPU	GPU	GPU
		FPGA		
		VPU		

Hardware Acceleration

Enable Video Pipeline Acceleration Hardware

Using Intel® Media SDK and the OpenVINO™ toolkit together enables customers to build high performance, intelligent vision solutions.

Intel® Media SDK

OpenVINO™ toolkit

Intel® Media SDK

Decode → Pre-Processing → Inference → Post-Processing → Encode

Decode	Pre-Processing	Inference	Post-Processing	Encode
CPU	CPU	CPU	CPU	CPU
GPU	GPU	GPU	GPU	GPU
		FPGA		
		VPU		

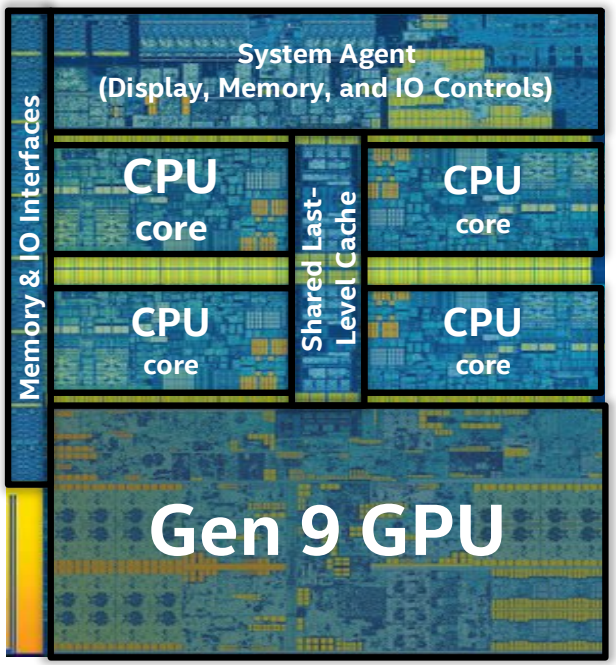
Hardware Acceleration

Intel Integrated Graphics

Intel® Core™ Processors include Gen GPU hardware

Gen GPUs can be used for graphics and also as general compute resources

Libraries contained in the OpenVINO™ toolkit (and many others) support Gen offload using OpenCL™



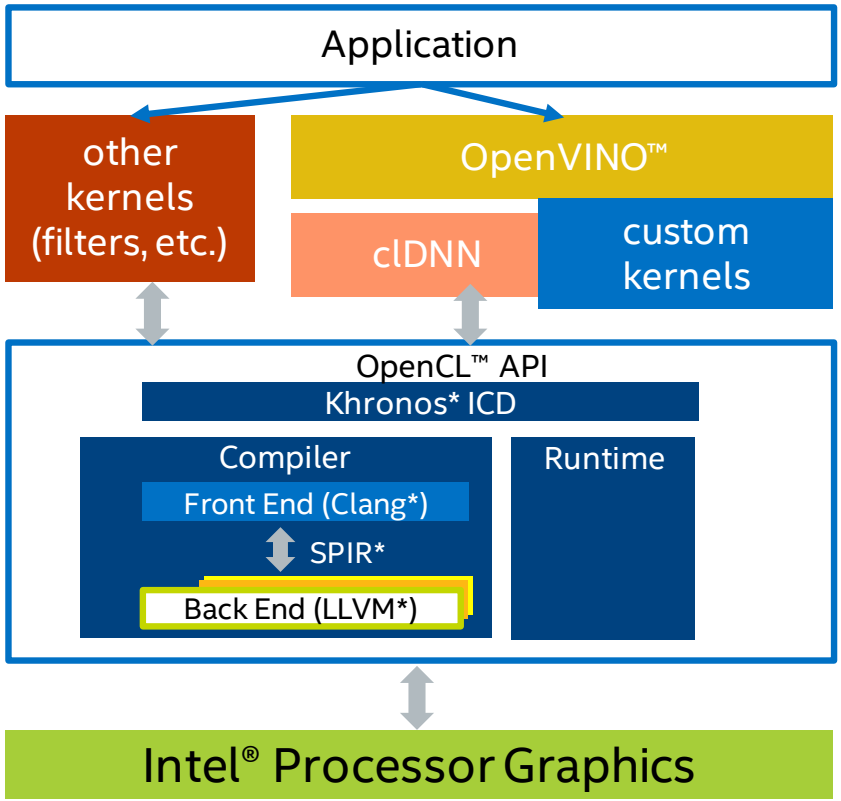
6th Generation Intel® Core™ i7 (Skylake) Processor

OpenCL™ Used to Enable Processor Graphics

Required to run with a GPU target (clDNN) using Intel® Processor Graphics

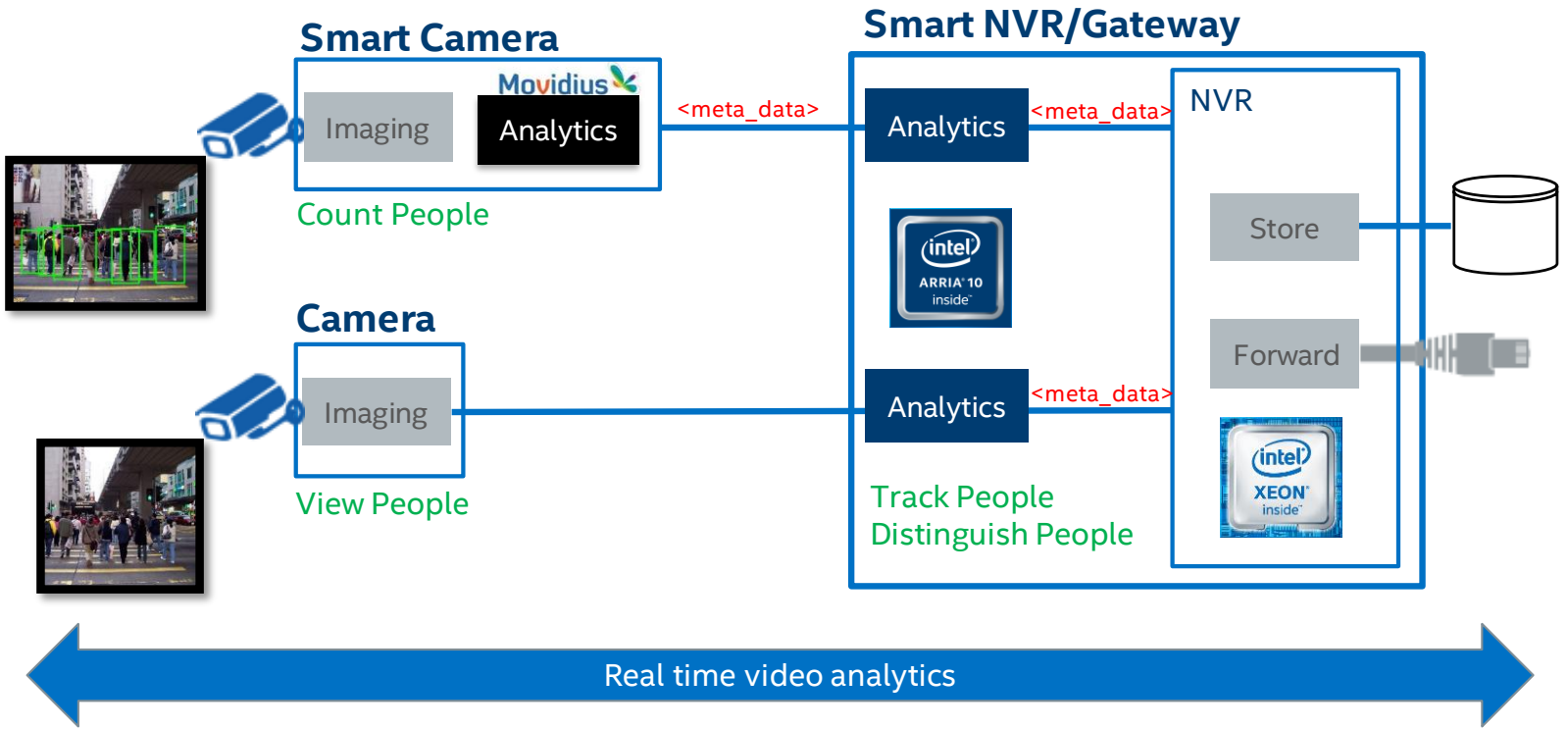
Custom kernels

Other kernels can be used for other non-inference pipeline stages, such as color conversions



Intel® Solution

Video Analytics at the Edge



Video Analytics with Intel

Building a real video analytics workload

- Heterogeneous system (CPU, GPU, FPGA, VPU)
- Computer Vision, Deep Learning, Media

Using Intel tools abstracts accelerator platform away

- Code the full workload for CPU (OpenCV), meet accuracy targets
- Until you meet the Power and performance targets
  - Identify critical path and most compute/memory demanding task
  - Offload critical task to an accelerator
  - Measure performance and device utilization

Get it here:

Intel® OpenVINO™ Toolkit

https://software.intel.com/en-us/openvino-toolkit

Intel® Media SDK

https://software.intel.com/media-sdk

Intel® System Studio

https://software.intel.com/en-us/intel-system-studio

Intel® Smart Video Portal

https://www.intel.com/content/www/us/en/embedded/solutions/smart-video/video-overview.html

Summary

Computer vision applications have multiple stages from image capture, pre processing, inference and post processing

Post processing provides context to the data and how it is used in the larger application

OpenCL™, OpenVX*, OpenCV are common programming languages used to develop CV applications

Intel provides a variety of hardware platforms and a common set of tools to implement CV applications

Legal Disclaimers/Acknowledgements

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. Check with your system manufacturer or retailer or learn more at www.intel.com.


intel®
experience
what's inside™