Deep learning inference with Intel FPGA tools

Vladimir Loncar

Workshop on ML on FPGAs for HEP INFN — Sezione di Bologna
November 4th, 2022

What is an FPGA?

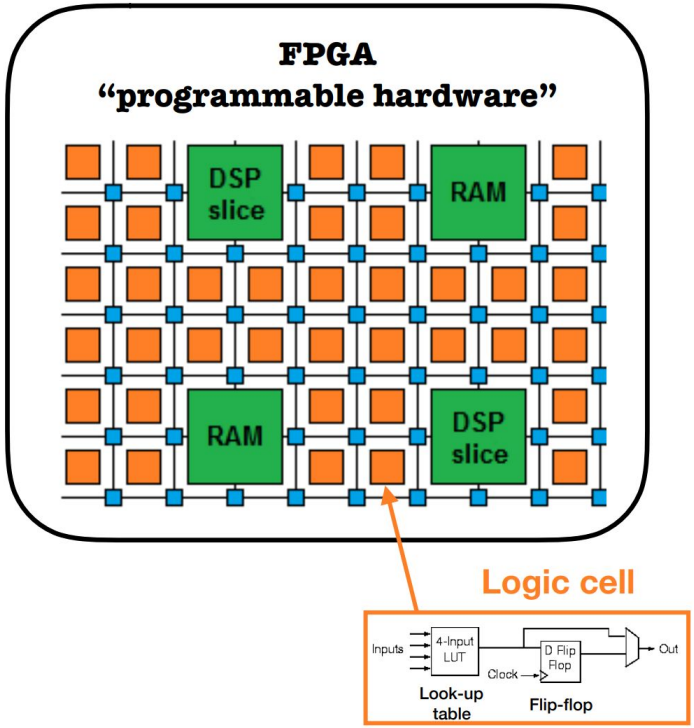
Reprogrammable integrated circuits

Configurable logic blocks and embedded components

- Flip-Flops (registers)
- LUTs (logic)
- DSPs (arithmetic)
- Block RAMs (memory)

Massively parallel

Low power



What is an FPGA?

Reprogrammable integrated circuits

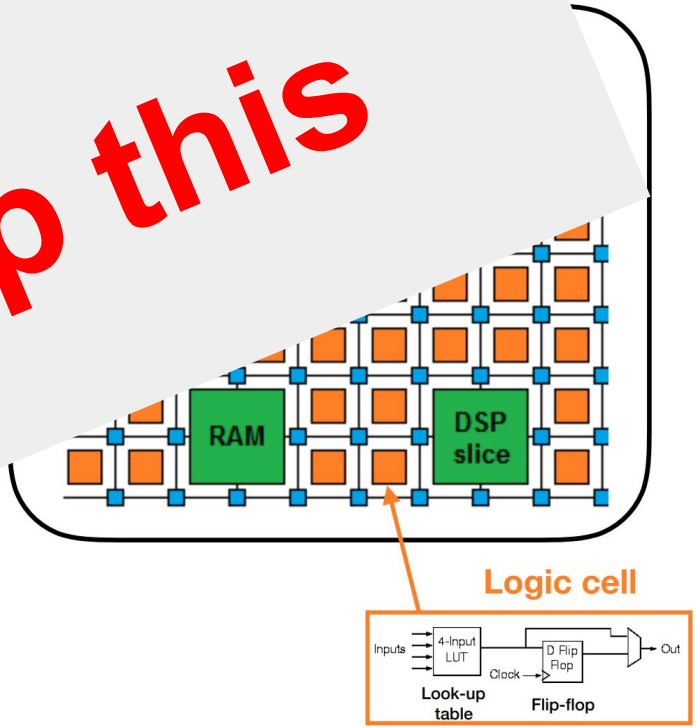
Configurable logic blocks and emb

- Flip-Flops (register
- LUTs (logic)
- DSP

M

Lov

We will skip this



Agenda

Why FPGAs are good for machine learning

- Focus on computer vision / convolutional neural networks

Intel solutions for ML on FPGAs

- Building ML processing pipelines

OpenVINO

- Model optimizer and Inference engine

Intel FPGA Deep Learning Acceleration Suite

Solving Challenges with FPGA

Real-Time

- Deterministic, low-latency

Flexibility

- Customizable hardware

Ease of use

- Software abstraction, libraries

FPGAs for machine learning

Highly parallel architecture

- Efficient low-batch stream processing

Configurable bit-width of DSP Blocks

- Accelerates computation by tuning compute performance

High-bandwidth memory

Programmable Data Path

- Improved efficiency through removal of unnecessary data movements

Configurability

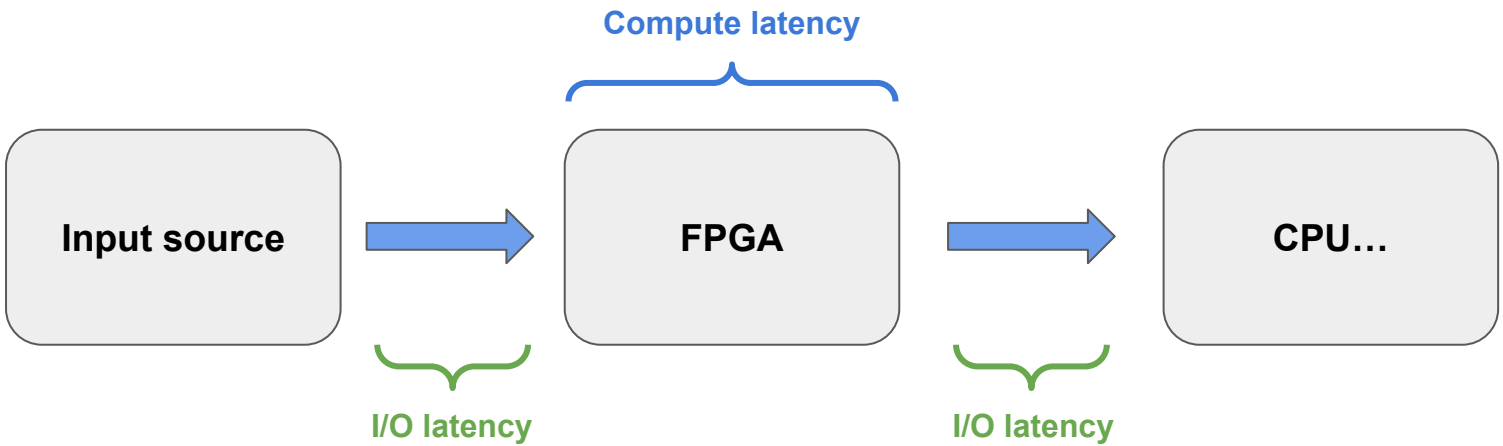
- Support for variable precision

Deterministic System Latency

Parallelism used to reduce latency

Customizable I/O with low & deterministic I/O latency

System Latency = I/O Latency + Compute Latency



FPGA flexibility

FPGA's flexibility allows them to supports arbitrary architectures

Improving the efficiency of NNs on GPUs is much more limited

- Batching
- Reduced bit width
- Sparse weights and/or activations
- Weight sharing
- …

Convolutional NNs on FPGAs

Convolutional Neural Networks are at the heart of computer vision

Efficiently mapped on on FPGAs

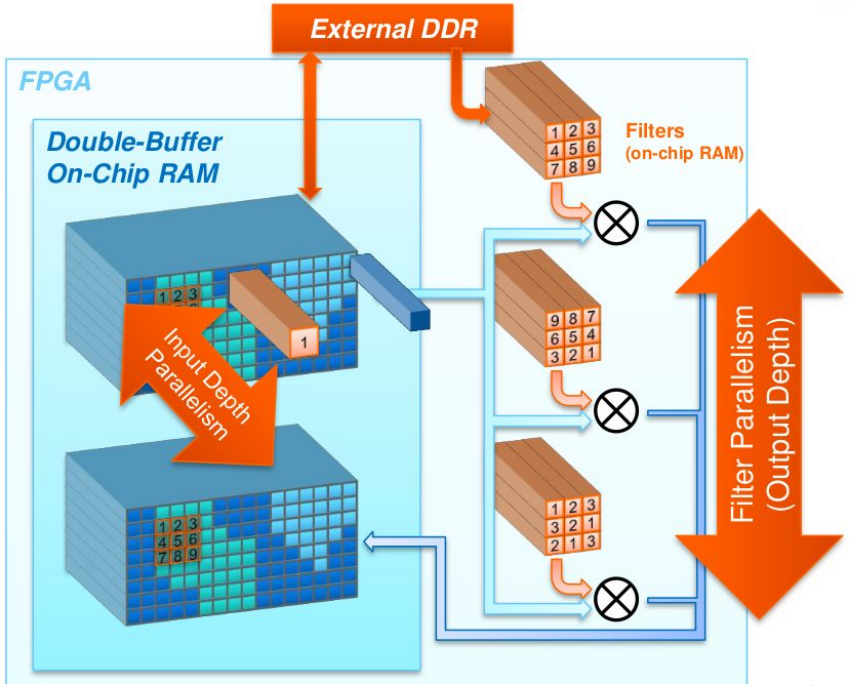
Parallel Convolutions

- Different filters of the same convolution layer processed in parallel in different processing elements (PEs)

Vectored Operations

- Across the depth of feature map

PE Array geometry can be customized



Common CV tasks

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)

Object Detection

- Localize and classify all objects in the image

Object Localization

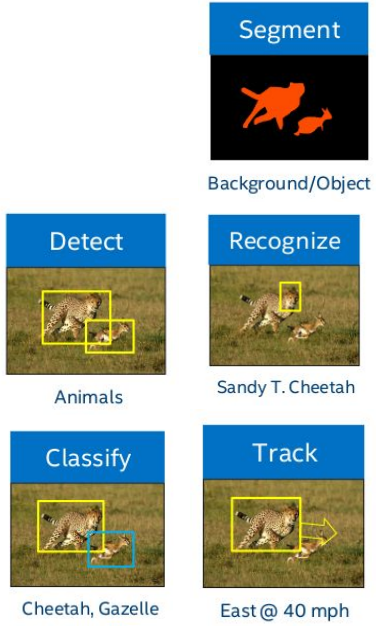
- Predict the region that contains the dominant object

Image Classification

- Classify an image based on the dominant object inside it

Face Recognition

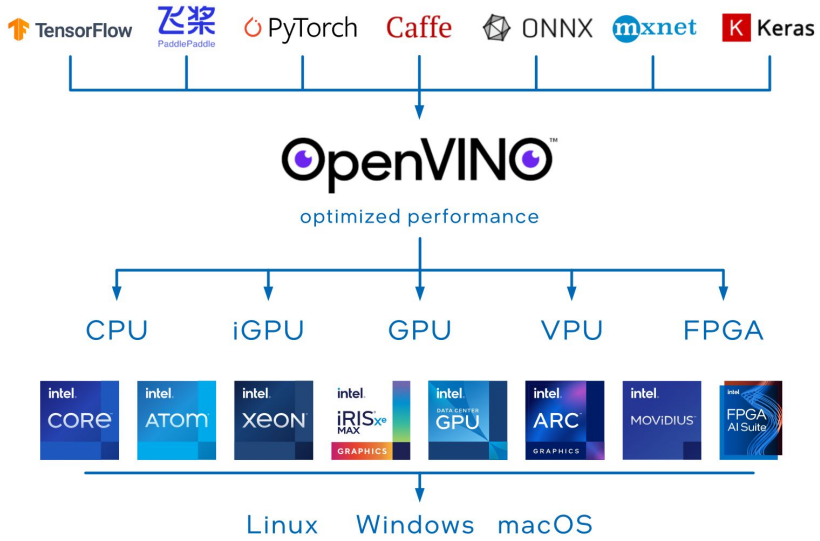
- Maps detected face with a stored reference



Intel OpenVINO Toolkit

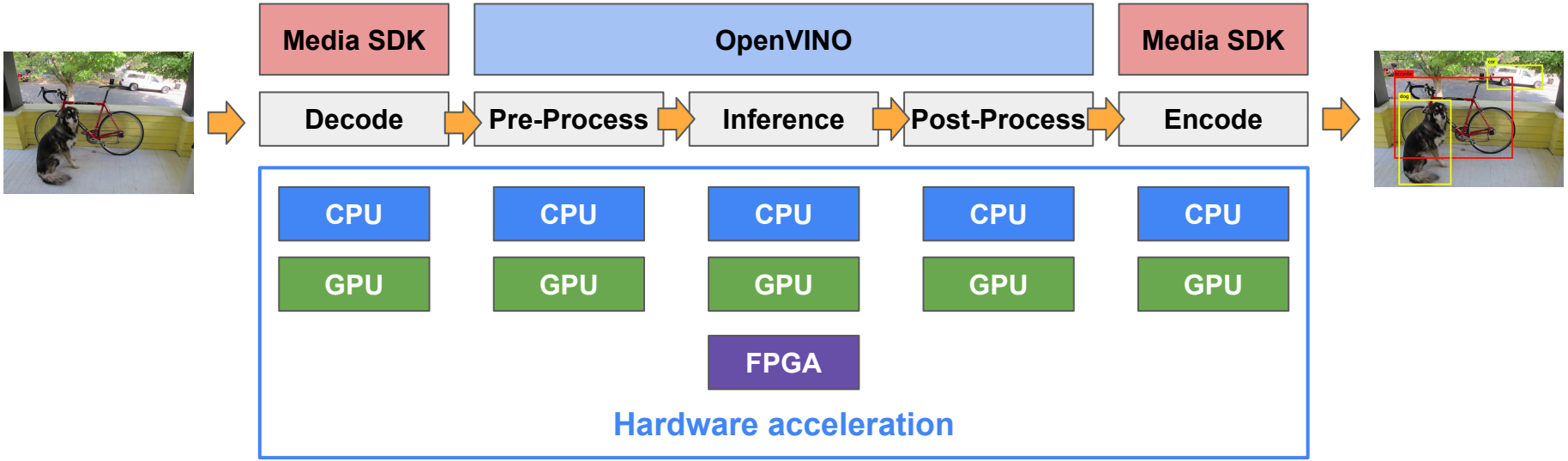
Intel’s solution for boosting deep learning performance in computer vision

- Applicable to other domains as well, such as speech recognition, natural language processing and other common tasks
- Works with models trained with popular frameworks like TensorFlow, PyTorch and others

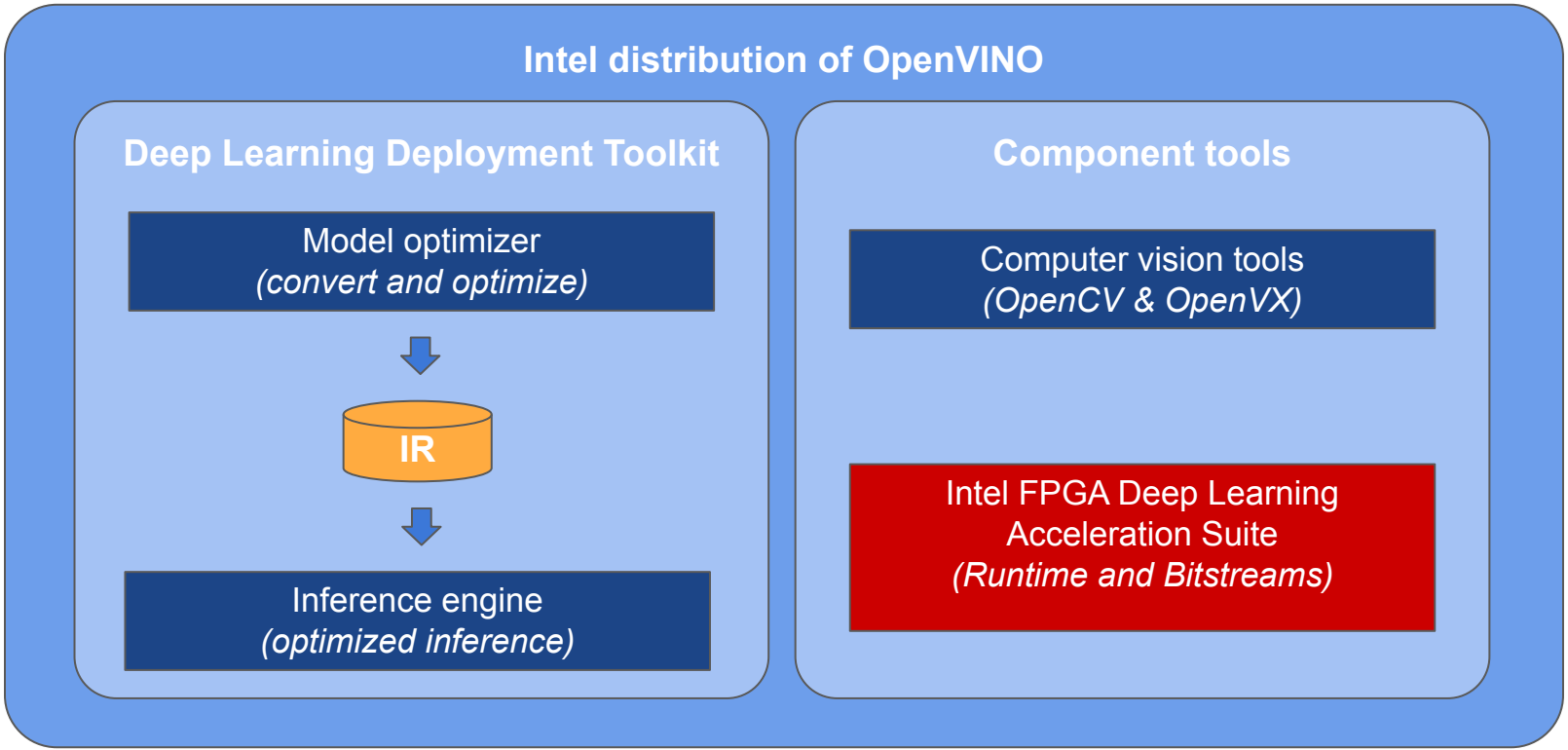


Accelerated CV processing pipeline

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



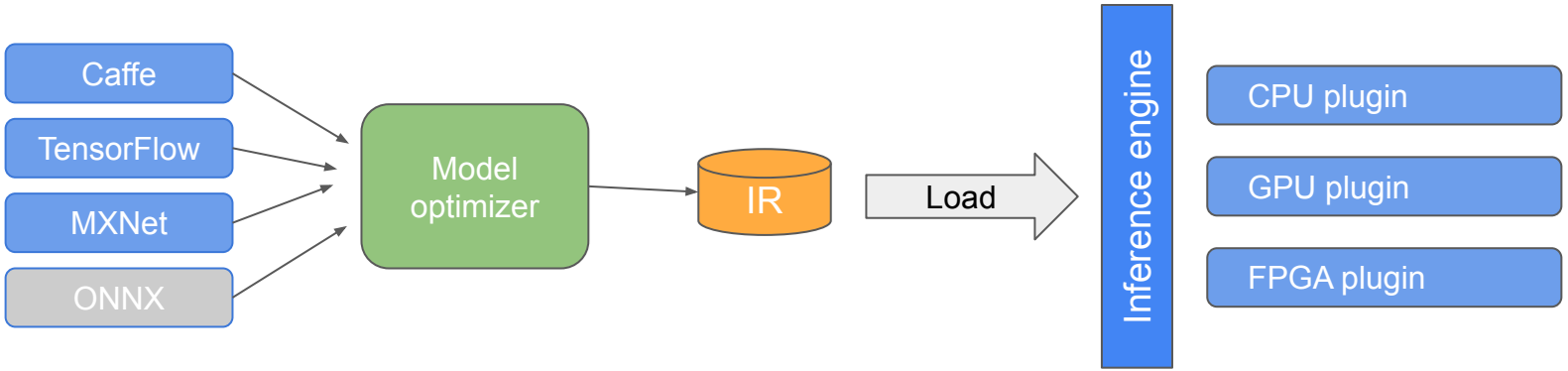
Components of OpenVINO toolkit



Deep Learning Deployment Toolkit

Enable deployment of trained model on all Intel architectures

- CPU, GPU, VPU, FPGA…
- Optimized for best execution on each architecture



Deep Learning Deployment Toolkit Details

Model Optimizer

- Imports trained models from popular deep learning frameworks regardless of training hardware
- Conservative topology transformations
- Converts to a range of data types (Matched to HW)

Inference Engine

- Optimizes Inference execution for target hardware (computational graph analysis, scheduling, model compression, quantization)
- Enables seamless integration with application logic

Model optimizer

Convert models from frameworks (Caffe, TensorFlow, MXNet, ONNX)

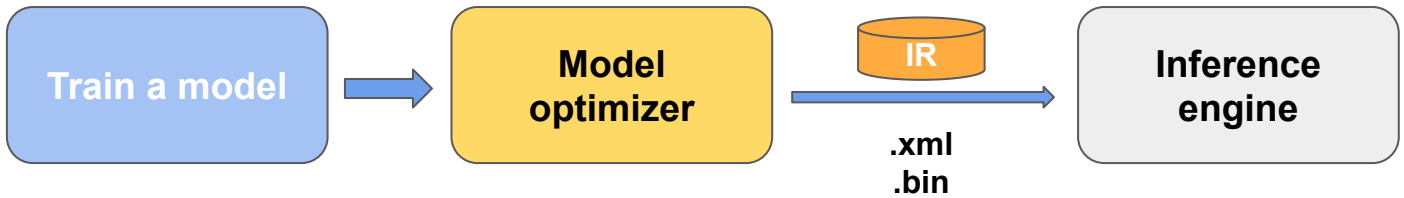
- PyTorch via ONNX

Converts to a unified internal representation (IR)

Optimizes topologies

- Node merging, batch normalization fusion, horizontal layer fusion etc

In ONNX and Tensorflow, folds constants



Optimizations in Model optimizer

Node merging

- Optimized kernels for convolution+pooling+relu

Horizontal fusion

- 1x1 convolution for e.g. Inception model

Batch normalization to scale-shift and optionally fold scale shift with convolution

Drop unused neurons (dropout)

FP16/INT8 quantization

Model-specific optimizations

- ResNet, Inception

Using Model optimizer

Python script, found at: `$MO_DIR/mo.py`

Example:

```
mo --input_model MobileNet.pb # tools available for freezing & saving TF graphs
   --input_shape [2,300,300,3]
   --data_type FP16
   --output_dir /some/path # optional
   # optional pre-processing
   --layout nhwc
   --scale 1
```

Other flags available for disabling optimizations

Inference engine

Common API shared across all Intel architectures

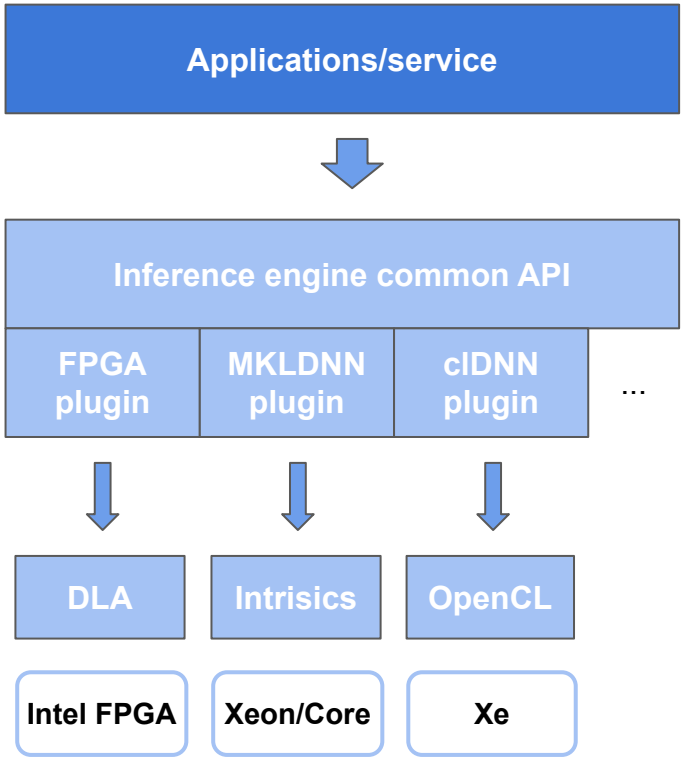
- C++ API
- Futureproof w.r.t. new Intel processors

Optimized inference on Intel hardware targets

- CPU/GPU/FPGA

Allows execution of layers across hardware types

- E.g., Run only portion of the model on FPGA



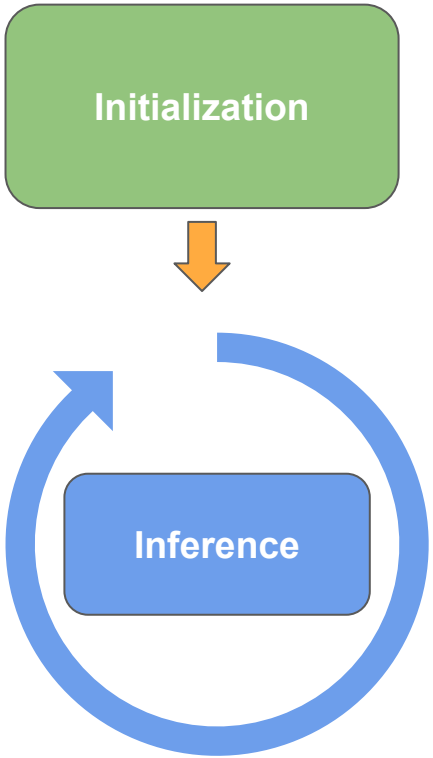
Inference Engine Workflow

Initialization

1. Load model and weights
2. Set batch size (if needed)
3. Load Inference Plugin (CPU, GPU, FPGA)
4. Load network to plugin
5. Allocate input, output buffers

Main loop

1. Fill input buffer with data
2. Run inference
3. Interpret output results



Inference Engine Plugins

CPU MKLDNN Plugin (Intel Math Kernel Library for Deep Neural Networks)

- Supports Intel Xeon/Core/Atom platform
- Widest set of network classes supported

GPU clDNN Plugin (Compute Library for Deep Neural Networks)

- Supports Intel Iris and Xe graphics processors
- Extensibility mechanism to develop custom layers through OpenCL

FPGA DLA Plugin

- Supports Intel Arria 10 GX and above devices
- Basic set of layers are supported on FPGA
    - Non-supported layers inferred through other plugins

Automatic Fallback with Hetero Plugin

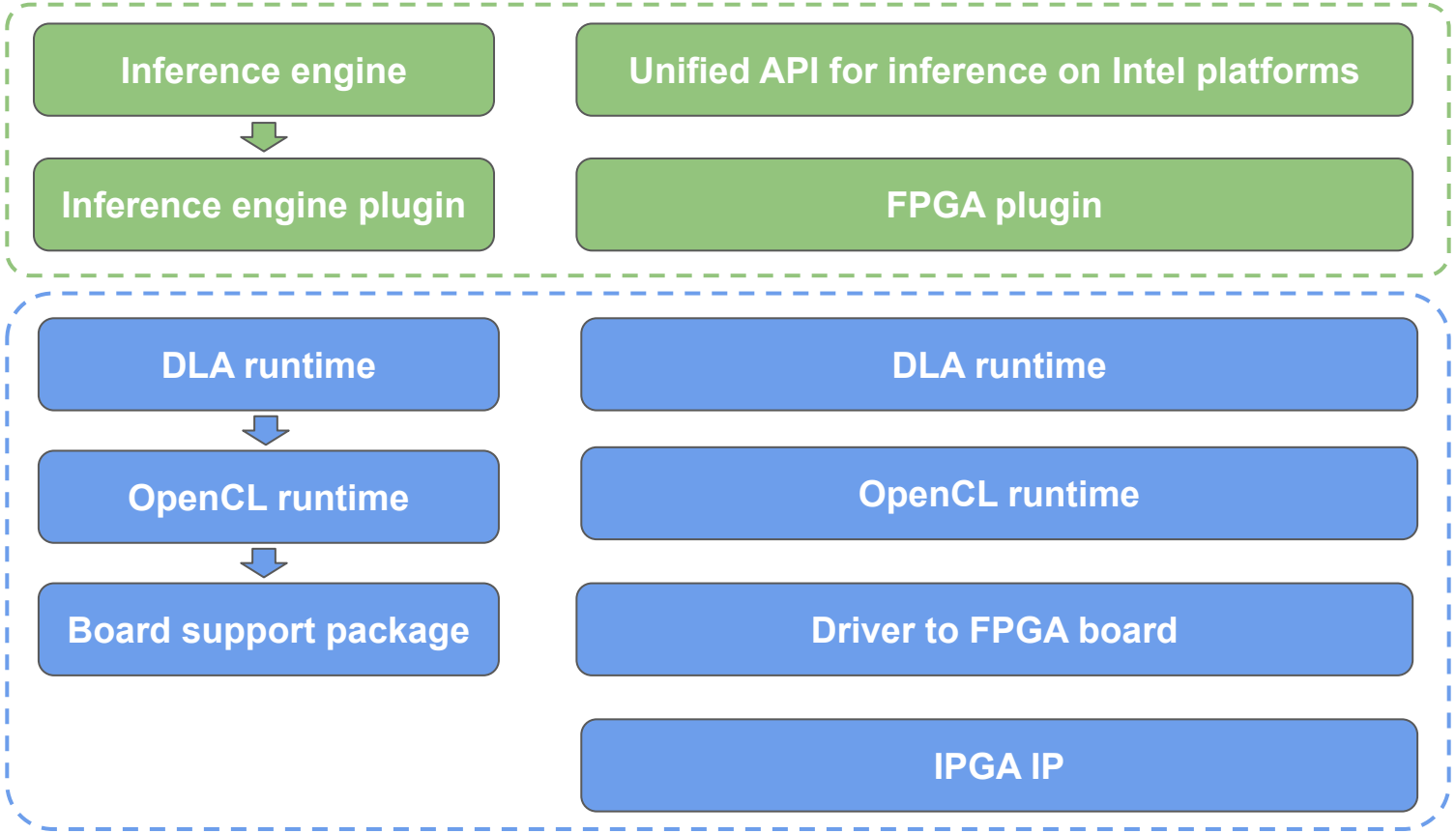
HETERO plugin

- Functions the FPGA does not support falls back to the CPU
- Fallback happens automatically, according to search priority (e.g., try GPU first, then CPU)

Manual splitting of original network may be required

- One network fully accelerated on the FPGA device
- Second network (consumes output of the first) executed on CPU or other devices
- Easier to split in original framework model

Inference Engine Architecture with FPGAs



Intel FPGA Deep Learning Acceleration Suite

CNN acceleration engine for common topologies executed in a graph loop architecture

Feature Map Cache

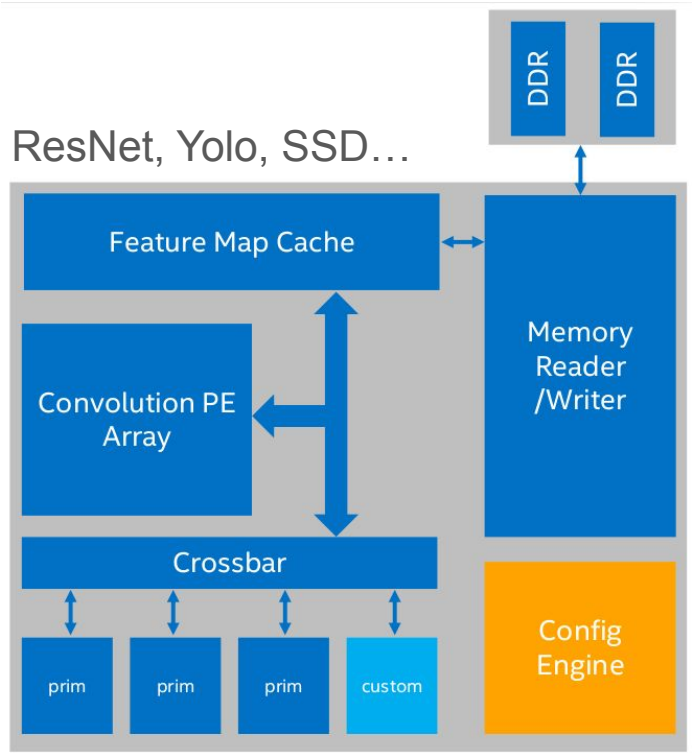
- AlexNet, GoogleNet, LeNet, SqueezeNet, VGG16, ResNet, Yolo, SSD…

Software Deployment

- No FPGA compile required
- Run-time reconfigurable

Customized Hardware Development

- Custom architecture creation w/ parameters
- Custom primitives using OpenCL flow



DLA User Flows

Application developer

Design

OpenVINO

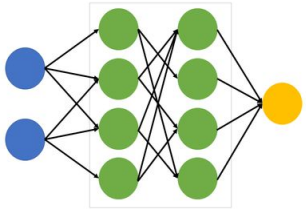
Run

CPU

Program

FPGA

Neural network shared between developer and architect



Bitstream library

Choose from many precompiled FPGA Images or create custom FPGA bitstream

IP Architect

Design

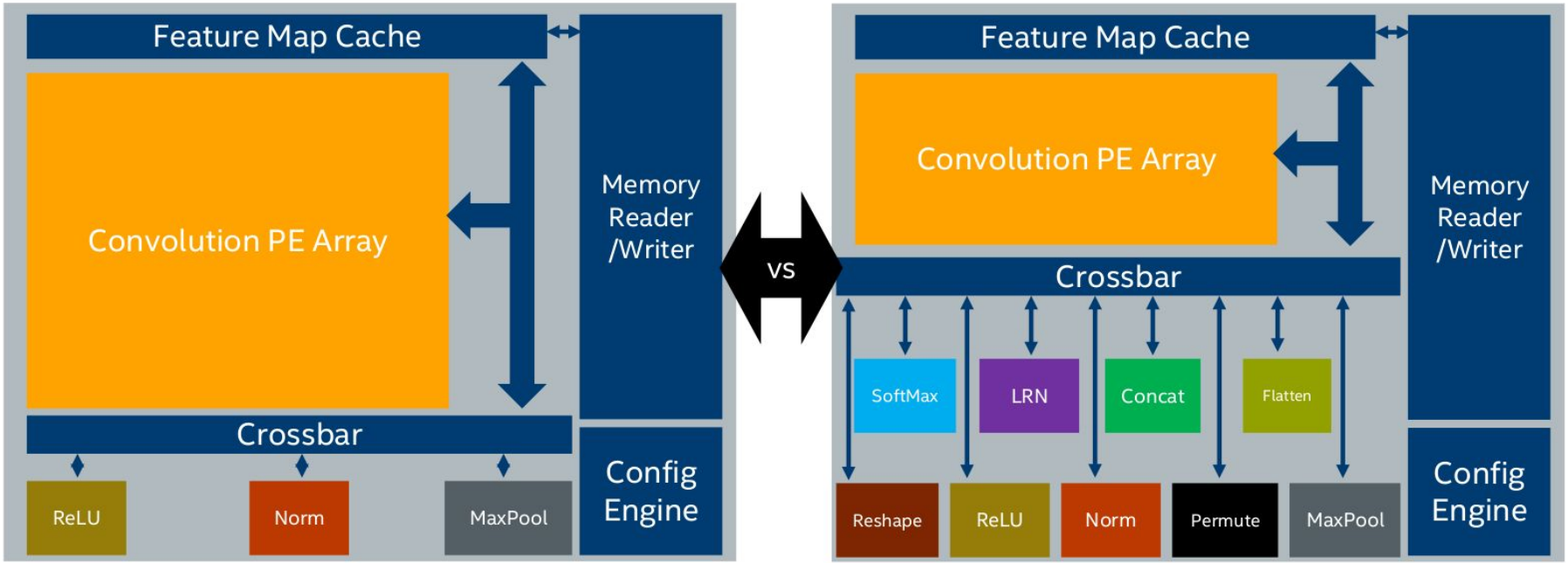
DLA customizations

Compile

Intel FPGA SDK

Support for Different Topologies

Tradeoff between features and performance



DLA Architecture Selection

DLA suite ships with many FPGA images for various boards/data types/topologies

- `<version>_<board>_<data type>_<Topologies/Feature>.aocx`
- For example:
  - `4-0_PL1_FP11_GoogleNet.aox`
  - `4-0_RC_FP16_MobileNet_ResNet_VGG_Clamp.aocx`

Find ideal FPGA image that meets your needs

Example:

- ResNet focused image does not have Norm (better performance)

Create custom FPGA image based on need

Supported Primitives and Topologies

Primitives

Conv	Eltwise
Pooling	Power
Fully Connected	ScaleShift
ReLu, Leaky ReLU	Batch Norm
Concat	LRM Normalization

Topologies

AlexNet

GoogLeNet

ResNet-18/50/101/152

SqueezeNet

VGG-16/19

Tiny Yolo

LeNet

MobileNet v1/v2

Design Exploration with Reduced Precision

Tradeoff between performance and accuracy

- Reduced precision allows more processing to be done in parallel
- Using smaller Floating Point format does not require retraining of network
- FP11 benefit over using INT8/9
  - No need to retrain, better performance, less accuracy loss

Summary

FPGAs provide a flexible, deterministic low-latency, high-throughput, and energy-efficient solution for accelerating the constantly changing networks and precisions for DL inference. FPGAs can implement lower precision without the need to retrain.

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

Intel OpenVINO toolkit provides a simple to use tool flow with a common front end to take models from common frameworks and target different hardware platforms easily. It can be used to map a trained network onto the FPGA architecture without needing an FPGA recompile or reload.

Deep Learning Accelerator Suite contains a library of architectures that support a variety of different primitive and topologies. The DLA architectures can be customized to add new primitives.

Resources

OpenVINO documentation

Intel's course on deep learning inference with FPGAs

- This talk is based on materials of that course

Intel Media SDK - an API for creating video (or video-like) processing pipeline

- Ties well with the OpenVINO