



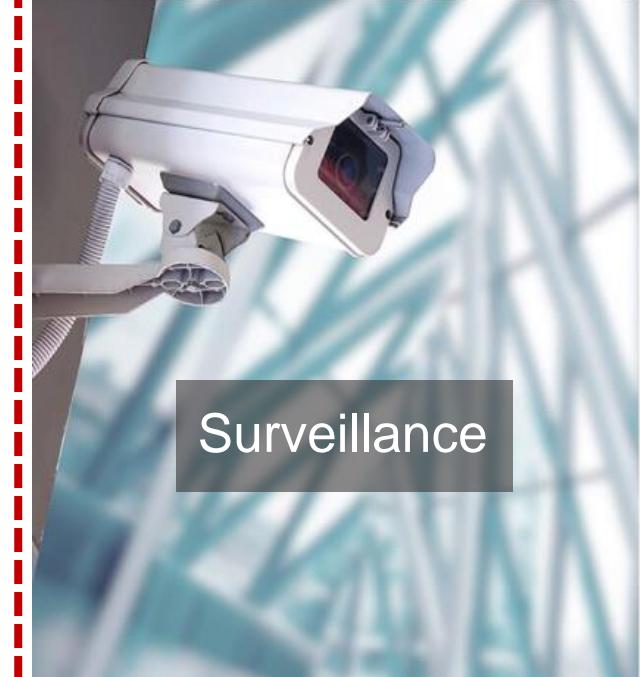
Machine learning for embedded deep dive

Presented By

Jingxui Lui
Sr. Product Marketing Manager



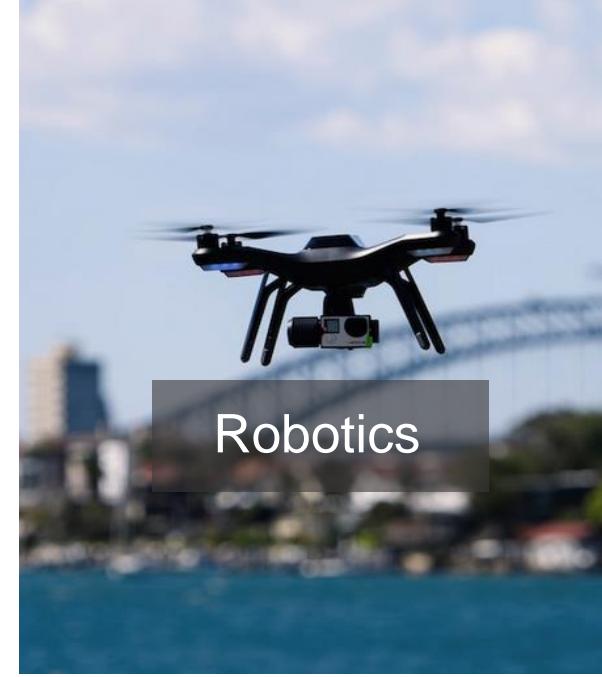
Key Machine Learning Applications for Xilinx



Surveillance



ADAS/AD



Robotics



Data Center

Cloud ML

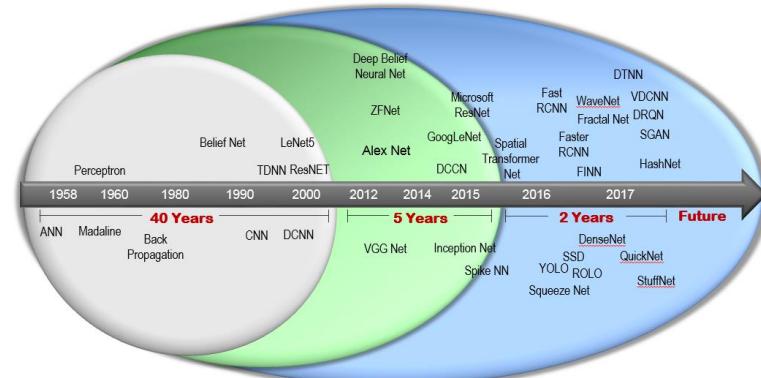
And there are many more ...

Edge ML

Xilinx Value Proposition in Edge/Embedded ML

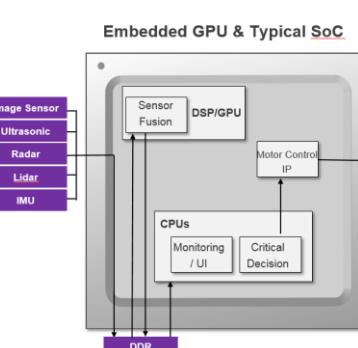
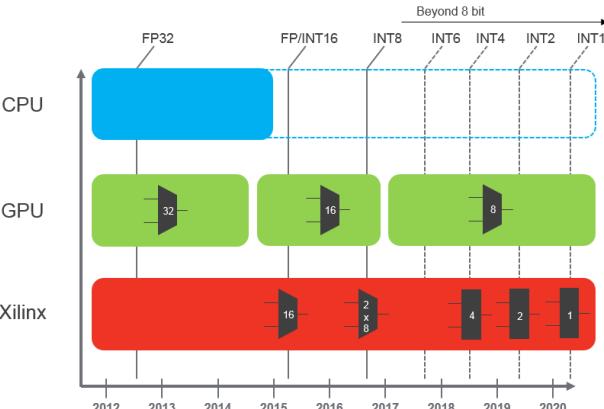
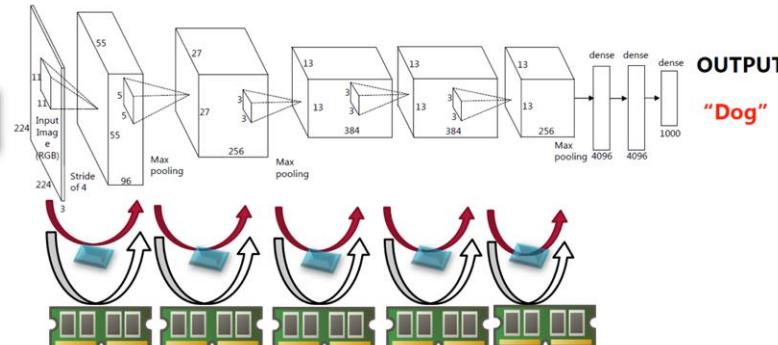
1

Only HW/SW configurable device
for fast changing networks



2

High performance / low power with
custom internal memory hierarchy



3

Future proof to lower
precisions

4

Low latency end-to-end

5

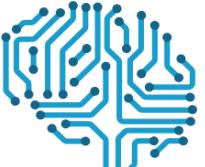
Scalable device family for
different applications

Key Challenges for Xilinx in Edge/Embedded ML

- 1 Deploy ML to Xilinx FPGA easily and quickly
- 2 Expand ML into non-FPGA customers
- 3 Delivers excellent performance with power & cost constraints for diverse embedded applications

reVISION Stack

Frameworks & Libraries

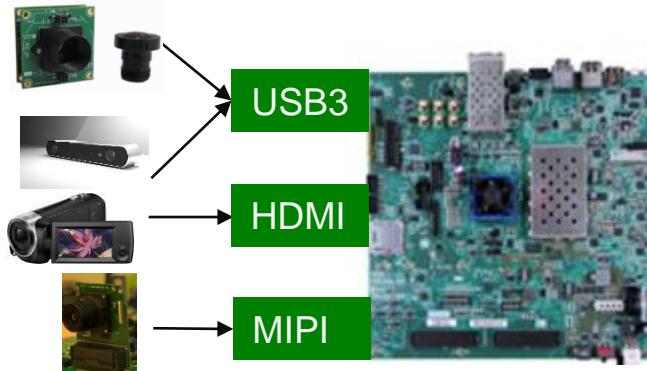


Machine Learning

Development tools

SDSoC™
Environment

Platforms



DEEPhi
深 鉴 科 技

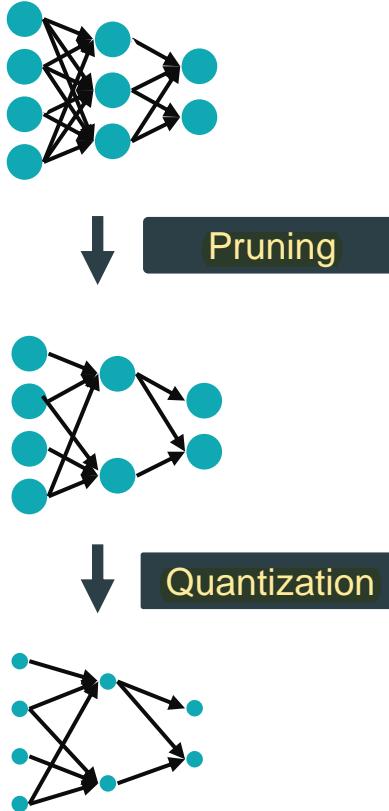
Xilinx Announces the Acquisition of DeePhi Tech

Deal to Accelerate Data Center and Intelligent Edge Applications

BEIJING and SAN JOSE, Calif., July 17, 2018 – Xilinx, Inc. (NASDAQ: XLNX), the leader in adaptive and intelligent computing, announced today that it has acquired DeePhi Tech, a Beijing-based privately held start-up with industry-leading capabilities in machine learning, specializing in deep compression, pruning, and system-level optimization for neural networks.

Deephi Edge ML Solution

Unique, Patented Deep Learning Acceleration Techniques



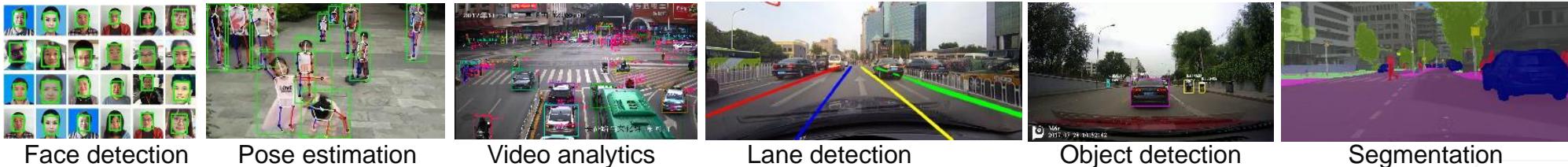
- > Best paper awards for breakthrough DL acceleration
- > Deephi's compression technology can:
 - >> Reduce DL accelerator footprint into smaller devices
 - >> Increase performance per watt (higher performance and/or lower energy)



Unique Pruning Technology Provides a Significant Competitive Advantage

DeePhi Solution Stack for Edge/Embedded ML

Models

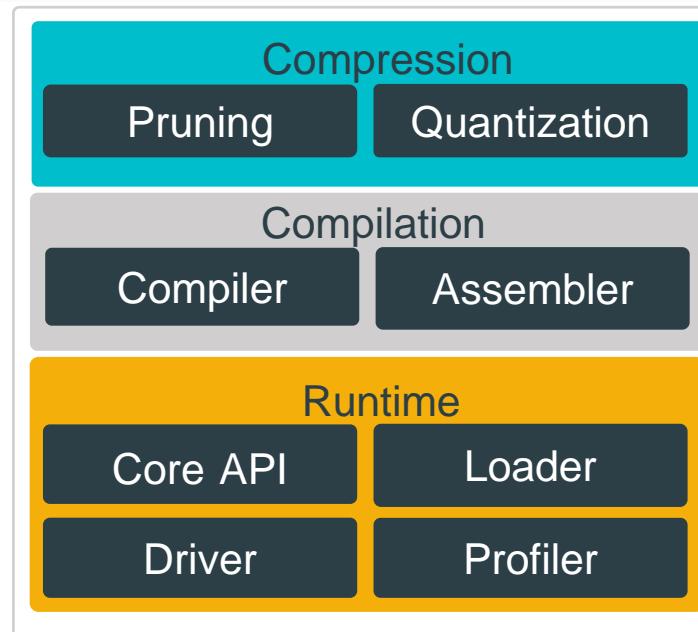


Framework

Caffe



Tools & IP



HW Platforms



Z7020 Board



Z7020 SOM



ZU2 SOM



ZU2/3 Card



ZU9 Card



ZCU102



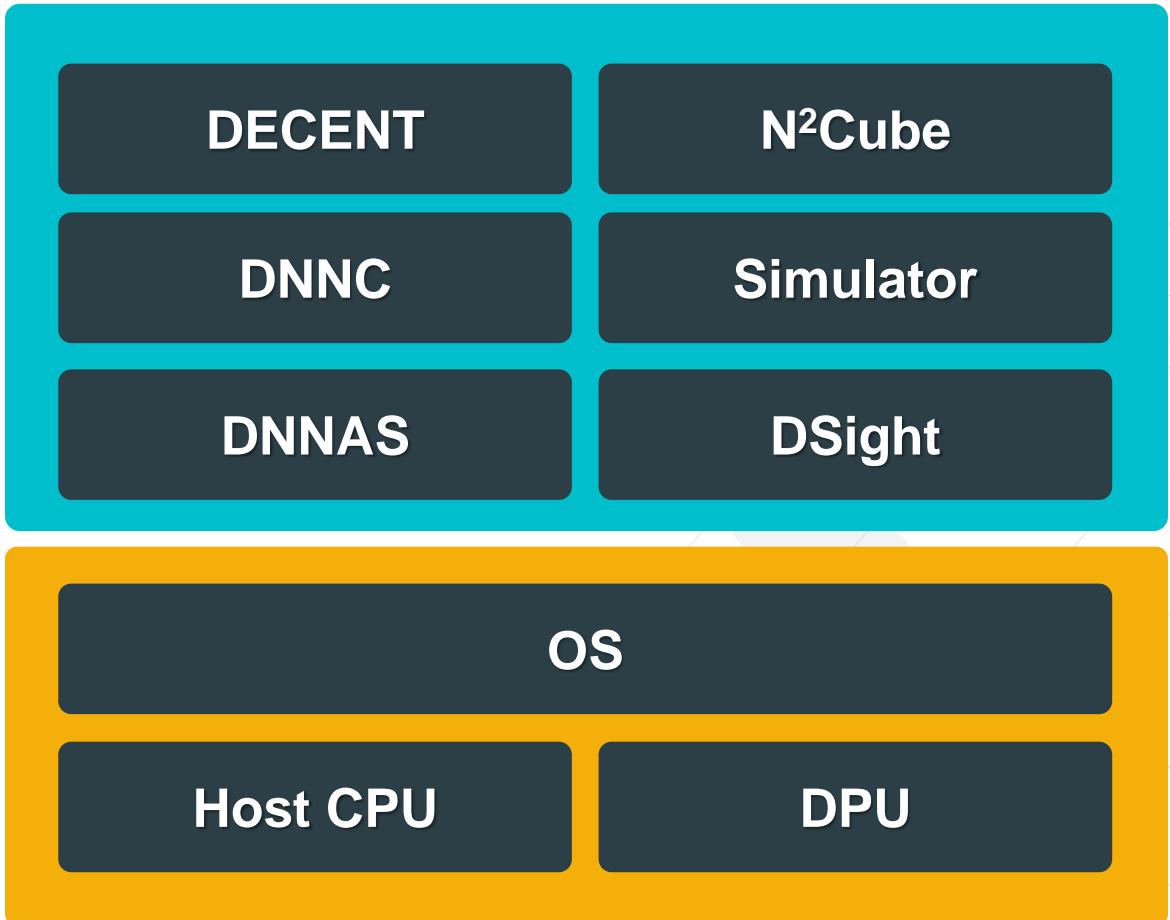
ZCU104



Ultra96

DNNDK Overview

- > DECENT (DEep ComprEssioN Tool)
- > DNNC (Deep Neural Network Compiler)
- > DNNAS (Deep Neural Network ASsembler)
- > Runtime N²Cube (Cube of Nerual Network)
- > DPU Simulator – Internal tool
- > Profiler DSight



Framework Support

Caffe

- Pruning
- Quantization
- Compilation

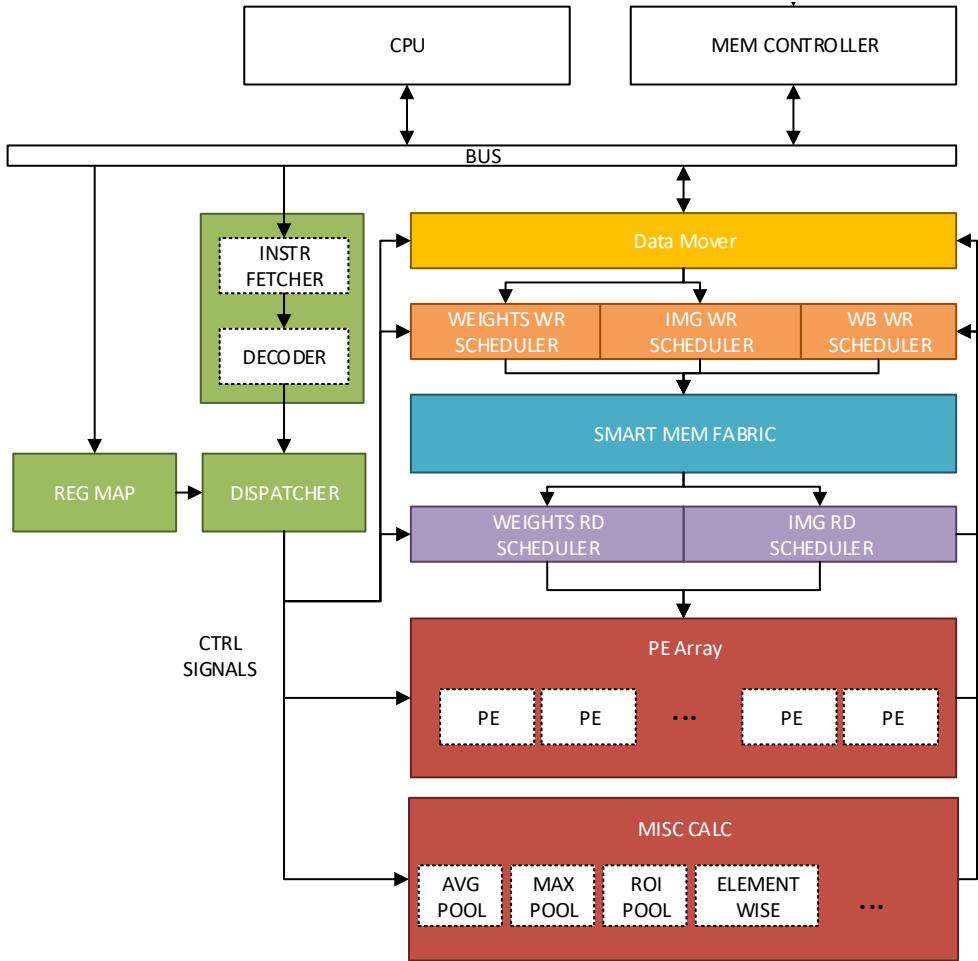


- Pruning
- Quantization
- Convertor for caffe

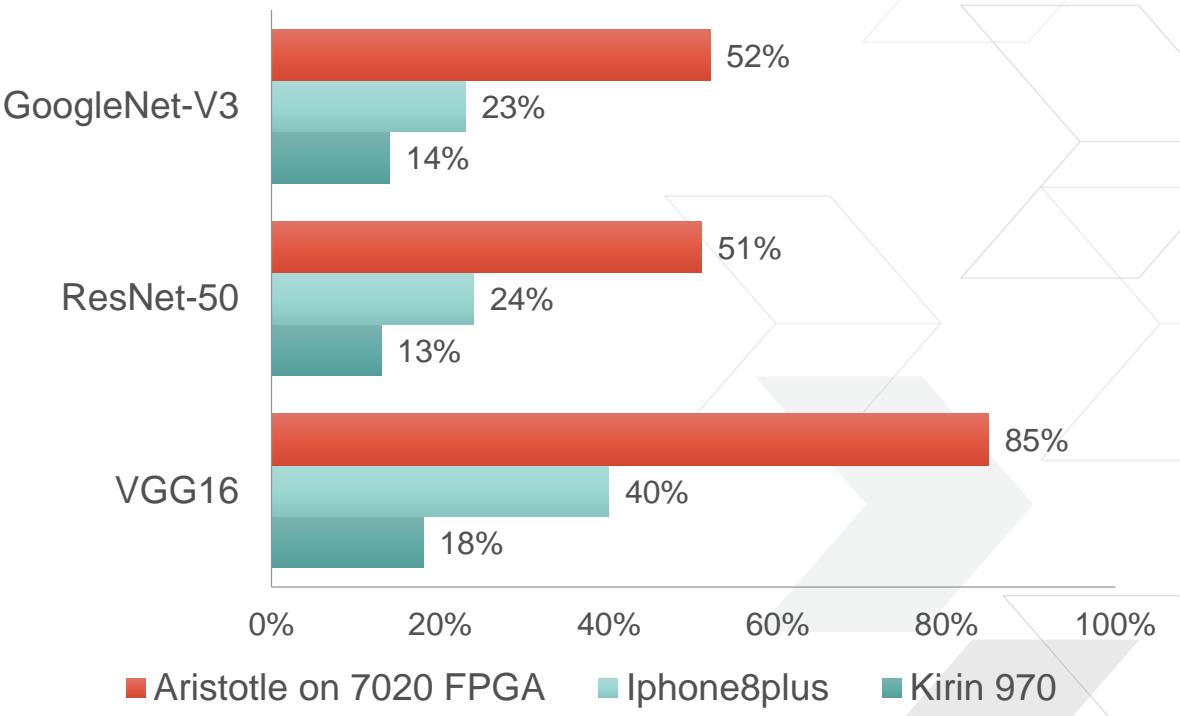


- Quantization & Compilation
 - Eval version
- Pruning
 - Internal version

DPU IP with High Efficiency



Utilization > 50% for mainstream neural networks



Source: Published results from Huawei

Supported Operators

- Arbitrary Input Image Size
- Conv
 - Arbitrary Conv Kernel Size
 - Arbitrary Conv Stride/Padding
 - Dilation
- Pooling
 - Max/Avg Pooling
 - Arbitrary Max Pooling Size
 - Avg Pooling kernel size: 2x2~7x7
 - Arbitrary Pooling Stride/Padding
- ReLU / Leaky Relu
- Concat
- Deconv
- Depthwise conv
- Elementwise
- FC(Int8/FP32)
- Mean scale
- Upsampling
- Batch Normalization
- Split
- Reorg
- Resize (Optional)
- Softmax (Optional)
- Sigmoid (Optional)

Constraints Between Layers

Next Layer Type \ Current Layer Type	Conv	Deconv	Depth-wise Conv	Inner Product	Max Pooling	Ave Pooling	BN	ReLU	LeakyReLU	Element-wise	Concat	As Input	As Output
Conv	●	●	○	●	●	○	●	●	○	●	●	●	●
Deconv	●	●	○	●	●	○	●	●	○	●	●	●	●
Depth-wise Conv	●	●	○	●	●	○	●	●	○	●	●	●	●
Inner Product	●	●	○	●	●	○	●	●	○	●	●	●	●
Max Pooling	●	●	○	●	●	○	○	×	×	●	●	●	●
Ave Pooling	○	○	○	○	○	○	○	×	×	○	○	○	○
BN	●	●	○	●	●	○	○	●	×	●	●	○	○
ReLU	●	●	○	●	●	○	○	×	×	●	●	--	●
LeakyReLU	○	○	○	○	○	○	○	×	×	○	○	--	○
Element-wise	●	●	○	●	●	○	○	●	○	●	●	--	●
Concat	●	●	○	●	●	○	○	×	×	●	●	--	●

●: Support

✗: Not support

○: Support when selecting additional features

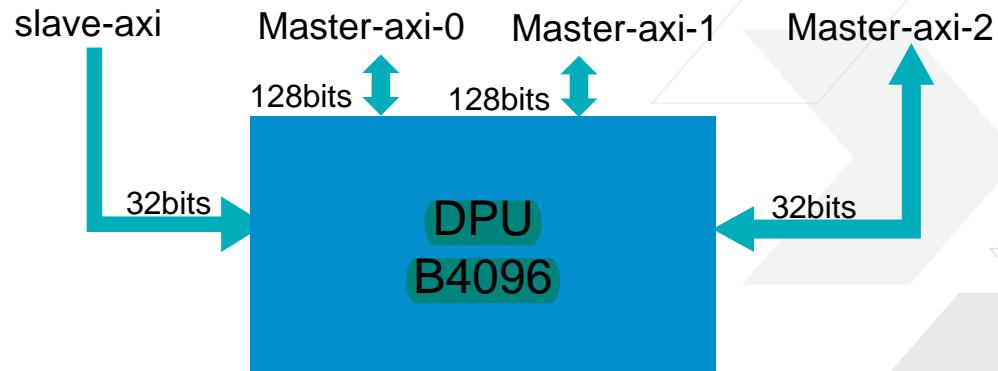
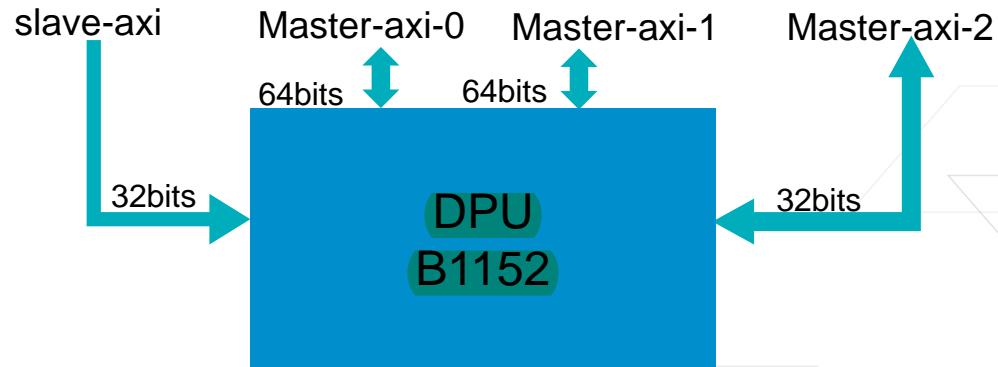
DPU Typical Options & Interfaces

> B1152

- » Parallelism: $4 * 12 * 12$
- » target Z7020/ZU2/ZU3

> B4096

- » Parallelism: $8 * 16 * 16$
- » Target ZU5 and above



DPU Peak Perf & Power

	LUT	Flip-Flops	Block RAM	DSP ¹⁾	DPU Config	MACs ²⁾	Peak ³⁾ performance	Frequency	Device Power
Z7020	53200	106400	4.9Mb	220	1xB1152	576	230GOPS	200MHz	2W
ZU2	47000	94000	5.3Mb	240	1xB1152	576	576GOPS	500MHz	3.5W
ZU3	71000	141000	7.6Mb	360	1xB1152	576	576GOPS	500MHz	N/A
ZU5 ⁴⁾	117000	234000	5.1Mb+18Mb	1248	1xB4096	2048	1350GOPS	330MHz	N/A
ZU7EV	230000	461000	11Mb+27Mb	1728	1xB4096 +2xB1152	2048 +2*576	2240GOPS	350MHz	N/A
ZU9	274000	548000	32.1Mb	2520	2xB4096	4096	2700GOPS	330MHz	10W

1) One DSP48E is used for two int8 multiplication

2) MACs is constructed by DSP and LUT (if DSP is not enough)

3) Peak performance is calculated by MACs: GOPS = 2*MACs*Frequency

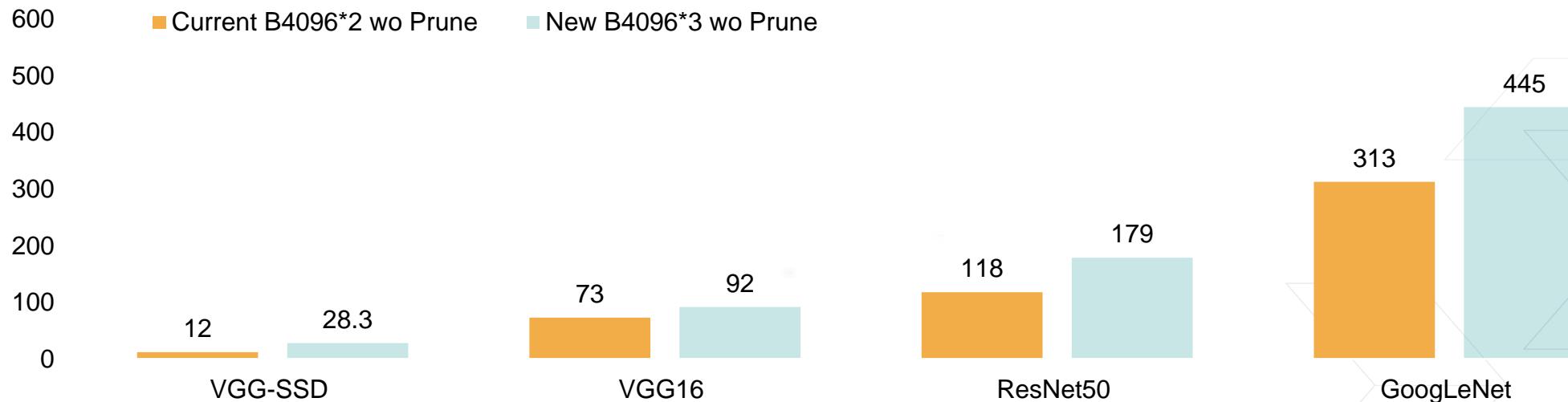
4) Just list our conservative projection in performance

DPU Utilization

		LUT	Slice_reg	Block Ram	DSPs
Single B1152 on Z7020	All logic	53200	106400	140	220
	DPU	45535	56961	110.5	220
	Utilization ratio	85.59%	53.53%	78.93%	100.00%
Single B1152 on ZU2		LUT	Slice_reg	Block Ram	DSPs
	All logic	47232	94464	150	240
	DPU	40703	55083	112	240
Single B1152 on ZU3	Utilization ratio	86.18%	58.31%	74.67%	100.00%
		LUT	Slice_reg	Block Ram	DSPs
	All logic	70560	141120	216	360
Dual B4096 on ZU9	DPU_B1152	36560	68729	115.5	288
	Utilization ratio	51.81%	48.70%	53.47%	66.67%
		LUT	Slice_reg	Block Ram	DSPs
Dual B4096 on ZU9	All logic	274080	548160	912	2520
	DPU	156744	224650	501	2048
	Utilization ratio	57.19%	40.98%	54.93%	81.27%

Perf Improvement with the Next Version DPU

Performance Comparison (FPS)



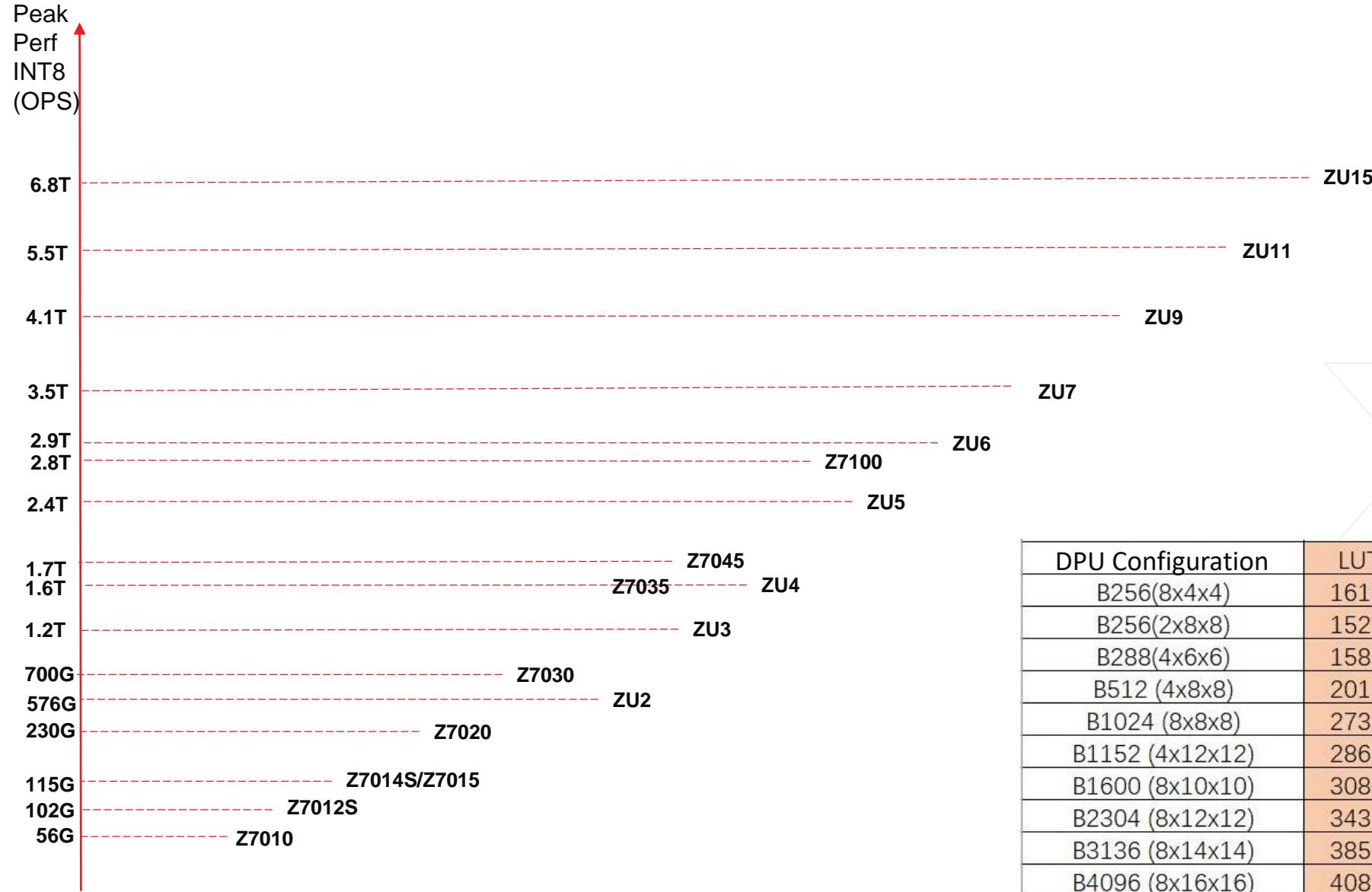
*The FPS of VGG-SSD of end to end performance

*The FPS of VGG16/ResNet50/GoogLeNet is of CONV part (w/o FC layer)

Resource Utilization Comparison

	DSP	LUT	FF	BRAM
Current B4096*2	2048	156744	224650	501
Next Version B4096*3	1926	110311	255020	748.5

DPU Scalability



DPU Configuration	LUTs	Registers	BRAM	DSP
B256(8x4x4)	16132	25064	43	66
B256(2x8x8)	15286	22624	53.5	50
B288(4x6x6)	15812	23689	46	62
B512 (4x8x8)	20177	31782	69.5	98
B1024 (8x8x8)	27377	46241	101.5	194
B1152 (4x12x12)	28698	46906	117.5	194
B1600 (8x10x10)	30877	56267	123	282
B2304 (8x12x12)	34379	67481	161.5	386
B3136 (8x14x14)	38555	79867	203.5	506
B4096 (8x16x16)	40865	92630	249.5	642

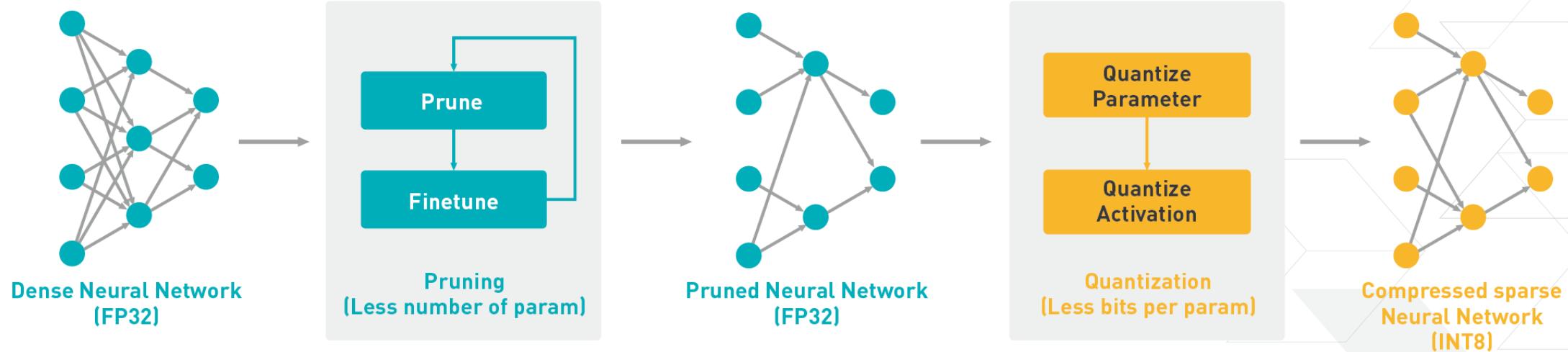
* B256/288/512/3136 work in progress

DNNDK Dev Flow

Five Steps with DNNDK

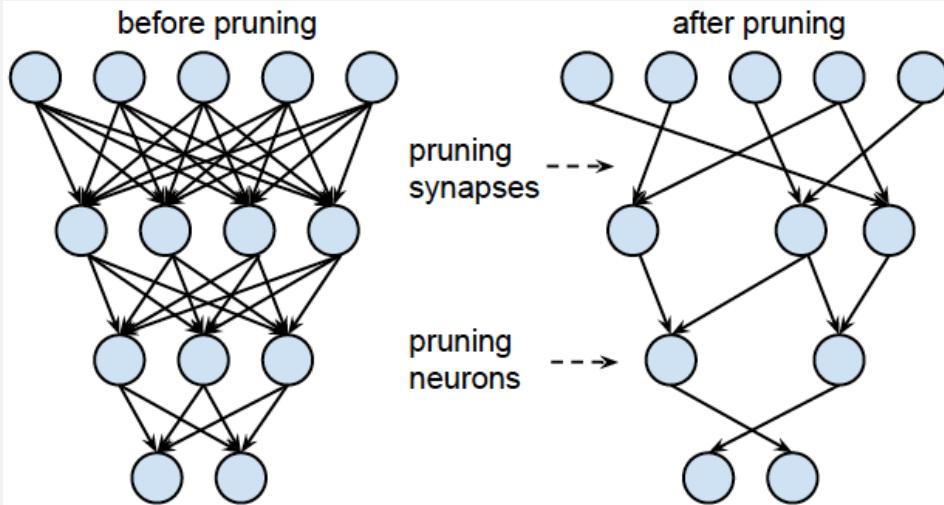
- 01 Model Compression
- 02 Model Compilation
- 03 Programming
- 04 Hybrid Compilation
- 05 Execution

DECENT – Deephi Deep Compression Tool



Deep Compression Overview

Deep compression
Makes algorithm smaller and lighter



Highlight



Compression efficiency

Deep Compression Tool can achieve significant compression on **CNN** and **RNN**

Accuracy

Algorithm can be **compressed 7 times without losing accuracy** under SSD object detection framework

Pruning Tool – decent_p

> 4 commands in decent_p

>> Ana

- analyze the network

>> Prune

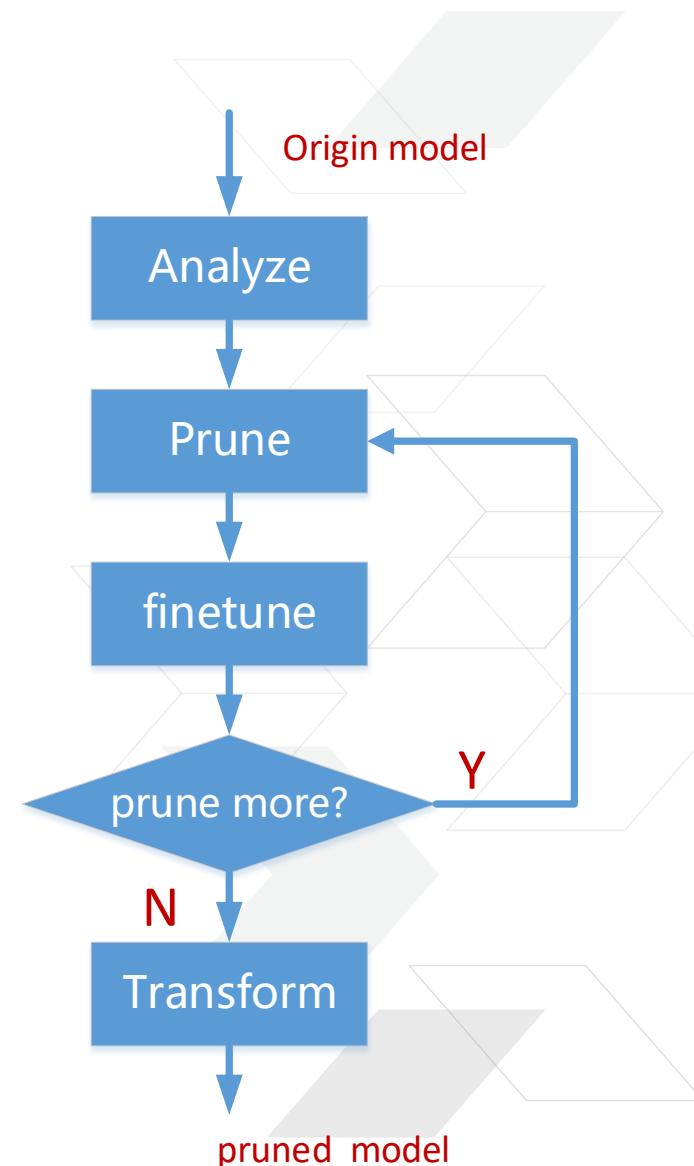
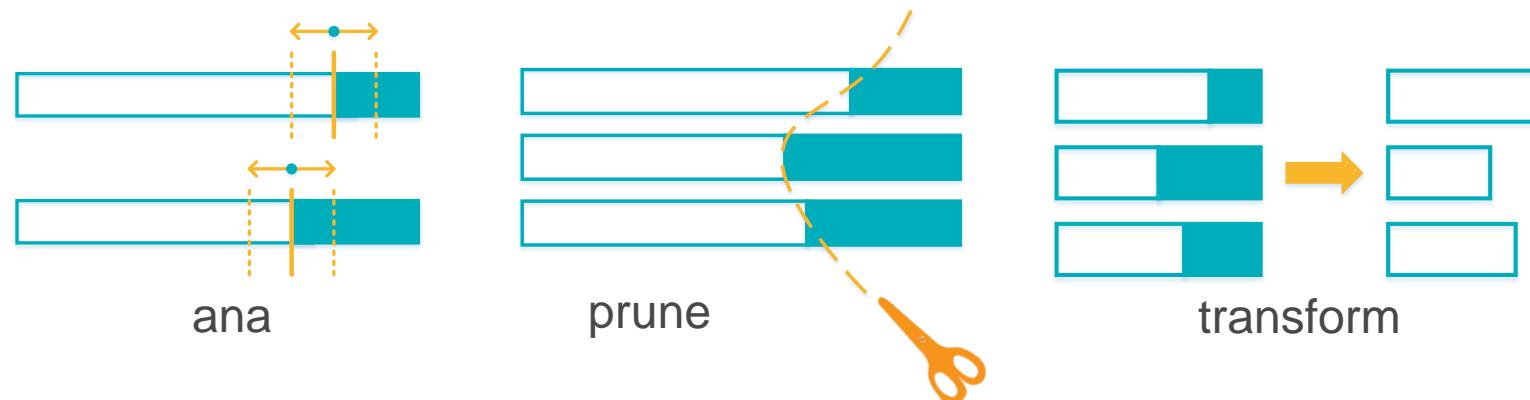
- prune the network according to config

>> Finetune

- finetune the network to recover accuracy

>> Transform

- transform the pruned model to regular model

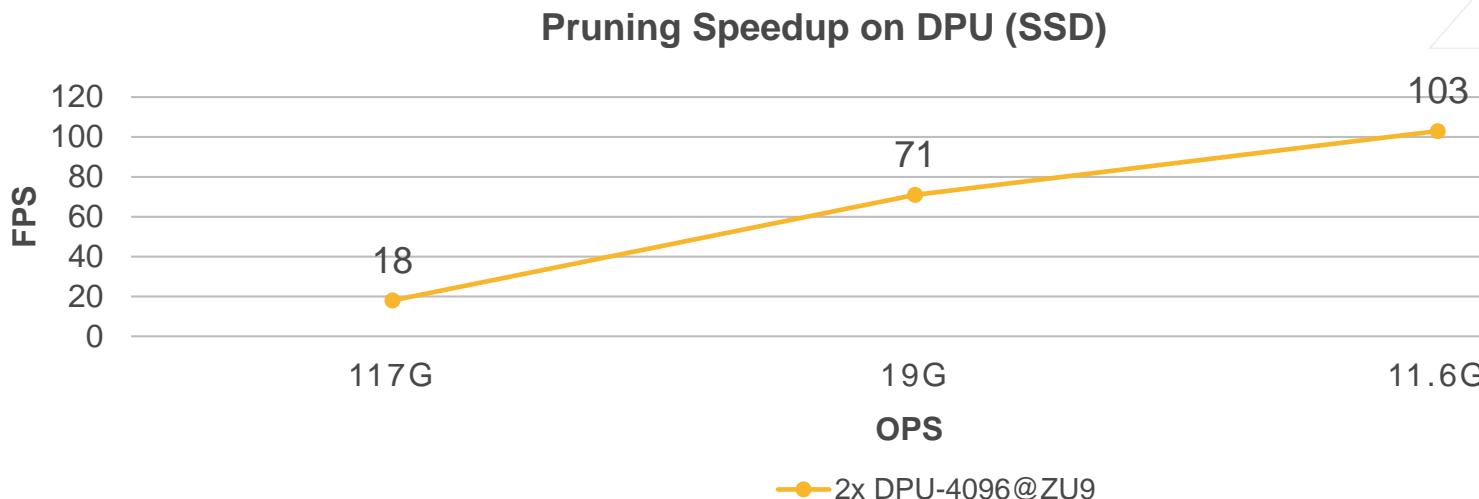
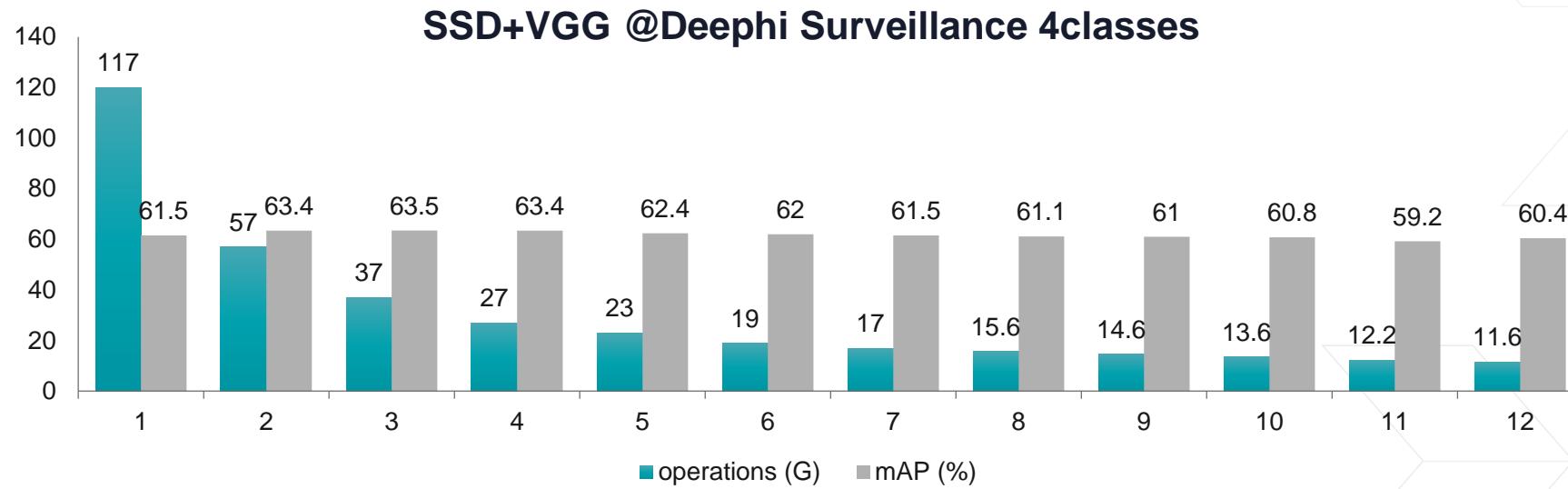


Pruning Results

Classification Networks	Baseline	Pruning Result 1			Pruning Result 2		
	Top-5	Top-5	ΔTop5	ratio	Top-5	ΔTop5	ratio
Resnet50 [7.7G]	91.65%	91.23%	-0.42%	40%	90.79%	-0.86%	32%
Inception_v2 [4.0G]	91.07%	90.37%	-0.70%	60%	90.07%	-1.00%	55%
SqueezeNet [778M]	83.19%	82.46%	-0.73%	89%	81.57%	-1.62%	75%

Detection Networks	Baseline mAP	Pruning Result 1			Pruning Result 2		
		mAP	ΔmAP	ratio	mAP	ΔmAP	ratio
DetectNet [17.5G]	44.46	45.7	+1.24	63%	45.12	+0.66	50%
SSD+VGG [117G]	61.5	62.0	+0.5	16%	60.4	-1.1	10%
[A] SSD+VGG [173G]	57.1	58.7	+1.6	40%	56.6	-0.5	12%
[B] Yolov2 [198G]	80.4	81.9	+1.5	28%	79.2	-1.2	7%

Pruning Example - SSD





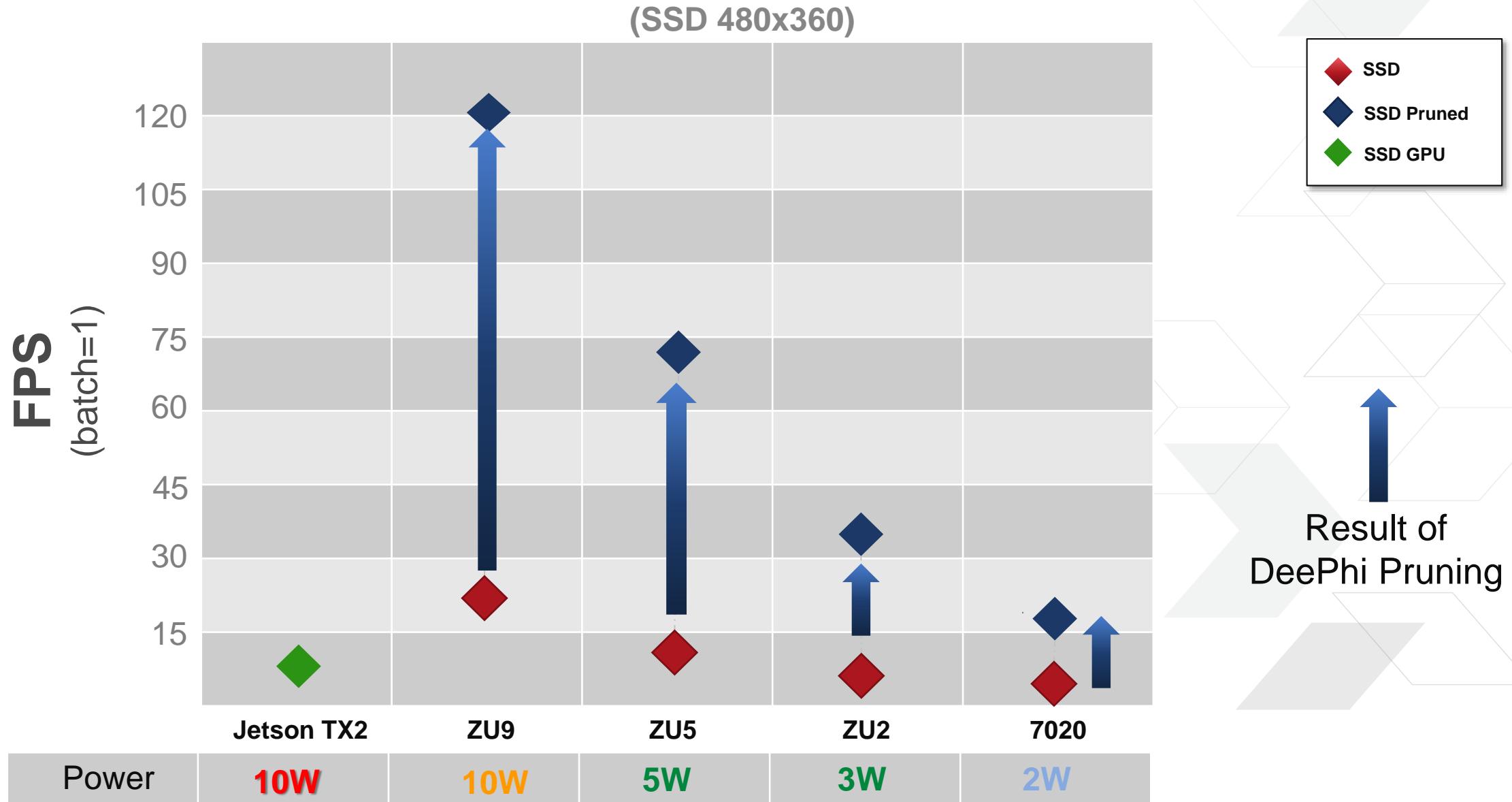
Before Compression
18FPS@117GOPs



After 10X Compression
103FPS@11.6GOPs

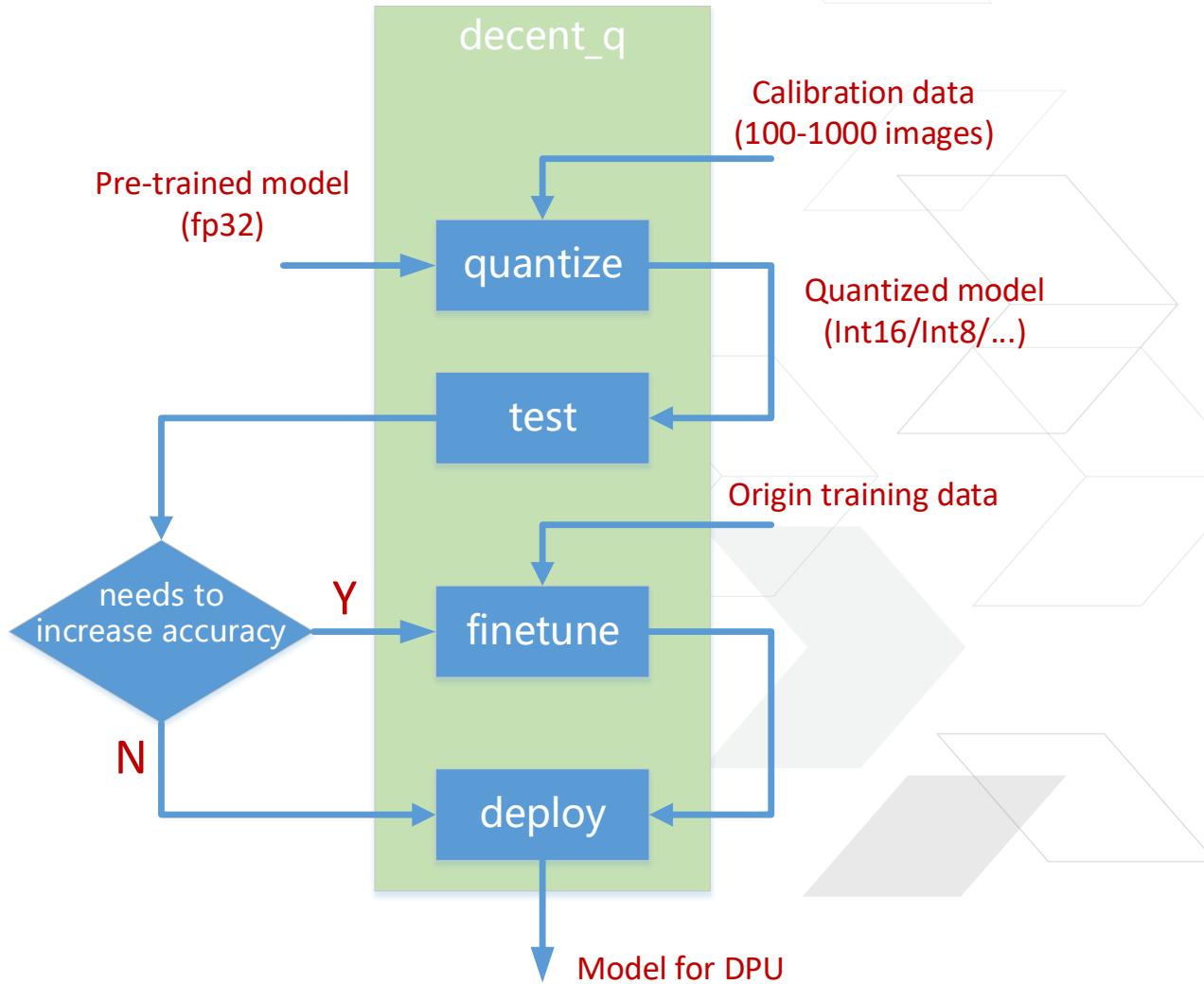
DEEPhi
深鉴科技

Makes Big Difference with Pruning



Quantization Tool – decent_q

- > 4 commands in decent_q
 - >> quantize
 - Quantize network
 - >> test
 - Test network accuracy
 - >> finetune
 - Finetune quantized network
 - >> deploy
 - Generate model for DPU
- > Data
 - >> Calibration data
 - Quantize activation
 - >> Training data
 - Further increase accuracy



Quantization Results

> Uniform Quantization

- >> 8-bit for both weights and activation
- >> A small set of images for calibration

Networks	Float32 baseline		8-bit Quantization			
	Top1	Top5	Top1	ΔTop1	Top5	ΔTop5
Inception_v1	66.90%	87.68%	66.62%	-0.28%	87.58%	-0.10%
Inception_v2	72.78%	91.04%	72.40%	-0.38%	90.82%	-0.23%
Inception_v3	77.01%	93.29%	76.56%	-0.45%	93.00%	-0.29%
Inception_v4	79.74%	94.80%	79.42%	-0.32%	94.64%	-0.16%
ResNet-50	74.76%	92.09%	74.59%	-0.17%	91.95%	-0.14%
VGG16	70.97%	89.85%	70.77%	-0.20%	89.76%	-0.09%
Inception-ResNet-v2	79.95%	95.13%	79.45%	-0.51%	94.97%	-0.16%

DNNDK API

dpuOpen()
dpuClose()
dpuLoadKernel()
dpuDestroyKernel()
dpuCreateTask()
dpuRunTask()
dpuDestroyTask()
dpuEnableTaskProfile()
dpuGetTaskProfile()
dpuGetNodeProfile()
dpuGetInputTensor()
dpuGetInputTensorAddress()
dpuGetInputTensorSize()
dpuGetInputTensorScale()
dpuGetInputTensorHeight()
dpuGetInputTensorWidth()
dpuGetInputTensorChannel()
dpuGetOutputTensor()
dpuGetOutputTensorAddress()

dpuGetOutputTensorSize()
dpuGetOutputTensorScale()
dpuGetOutputTensorHeight()
dpuGetOutputTensorWidth()
dpuGetOutputTensorChannel()
dpuGetTensorSize()
dpuGetTensorAddress()
dpuGetTensorScale()
dpuGetTensorHeight()
dpuGetTensorWidth()
dpuGetTensorChannel()
dpuSetInputTensorInCHWInt8()
dpuSetInputTensorInCHWFP32()
dpuSetInputTensorInHWCInt8()
dpuSetInputTensorInHWCFP32()
dpuGetOutputTensorInCHWInt8()
dpuGetOutputTensorInCHWFP32()
dpuGetOutputTensorInHWCInt8()
dpuGetOutputTensorInHWCFP32()

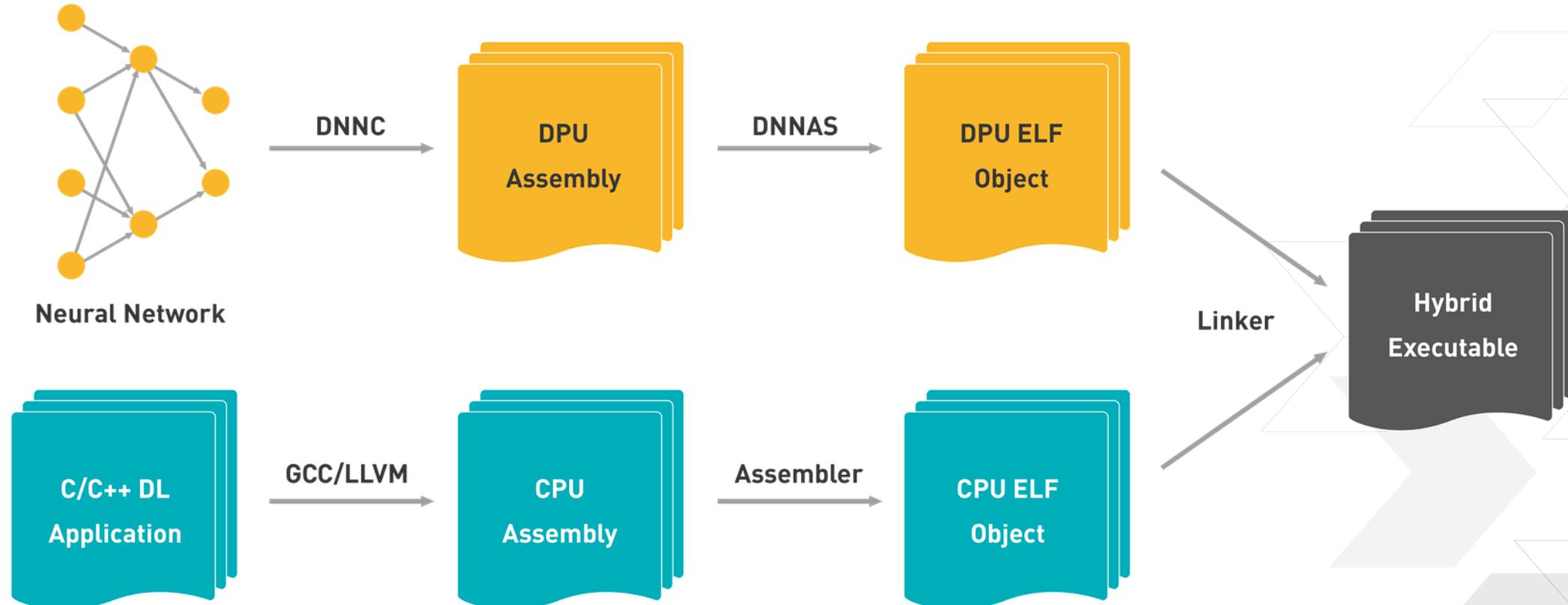
> **For more details, refer to
DNNDK User Guide**

[http://www.dephi.com/technology/
dnndk](http://www.dephi.com/technology/dnndk)

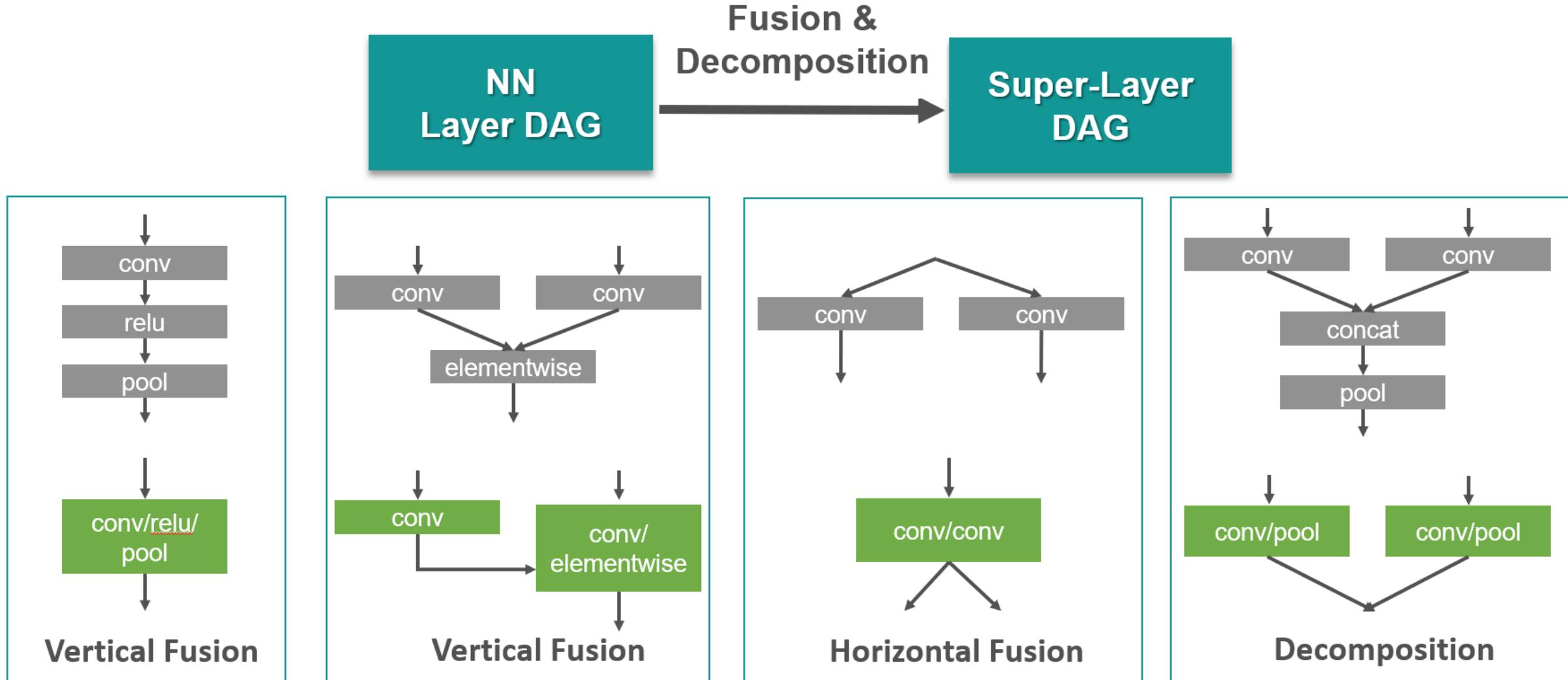
Programming with DNNDK API

```
1 int main(int argc, char *argv[])
2 {
3     DPUKernel *kernel_conv;
4     DPUKernel *kernel_fc;
5     DPUTask *task_conv;
6     DPUTask *task_fc;
7     char *input_addr;
8     char *output_addr;
9
10    /* DNNDK API to attach to DPU driver */
11    dpuInit();
12
13    /* DNNDK API to create DPU kernels for CONV & FC networks */
14    kernel_conv = dpuLoadKernel("resnet50_conv", 224, 224);
15    kernel_fc = dpuLoadKernel("resnet50_fc", 1, 1);
16
17    /* Create tasks from CONV & FC kernels */
18    task_conv = dpuCreateTask(kernel_conv);
19    task_fc = dpuCreateTask(kernel_fc);
20
21    /* Set input tensor for CONV task and run */
22    input_addr = dpuGetTensorAddress(dpuGetTaskInputTensor(task_conv));
23    setInputImage(Mat &image, input_addr);
24    dpuRunTask(task_conv);
25    output_addr = dpuGetTensorAddress(dpuGetTaskOutputTensor(task_conv));
26
27    /* Run average pooling layer on CPU */
28    run_average_pooling(output_addr);
29
30    /* Set input tensor for FC task and run */
31    input_addr = dpuGetTensorAddress(dpuGetTaskInputTensor(task_fc));
32    setFCInputData(task_fc, input_addr);
33    dpuRunTask(task_fc);
34    output_addr = dpuGetTensorAddress(dpuGetTaskOutputTensor(task_fc));
35
36    /* Diaplay the Classification result from FC task */
37    displayClassificationResult(output_addr);
38
39    /* DNNDK API to destroy DPU tasks/kernels */
40    dpuDestroyTask(task_conv);
41    dpuDestroyTask(task_fc);
42
43    dpuDestroyKernel(kernel_conv);
44    dpuDestroyKernel(kernel_fc);
45
46    /* DNNDK API to dettach from DPU driver and free DPU resources */
47    dpuFini();
48
49    return 0;
50 }
```

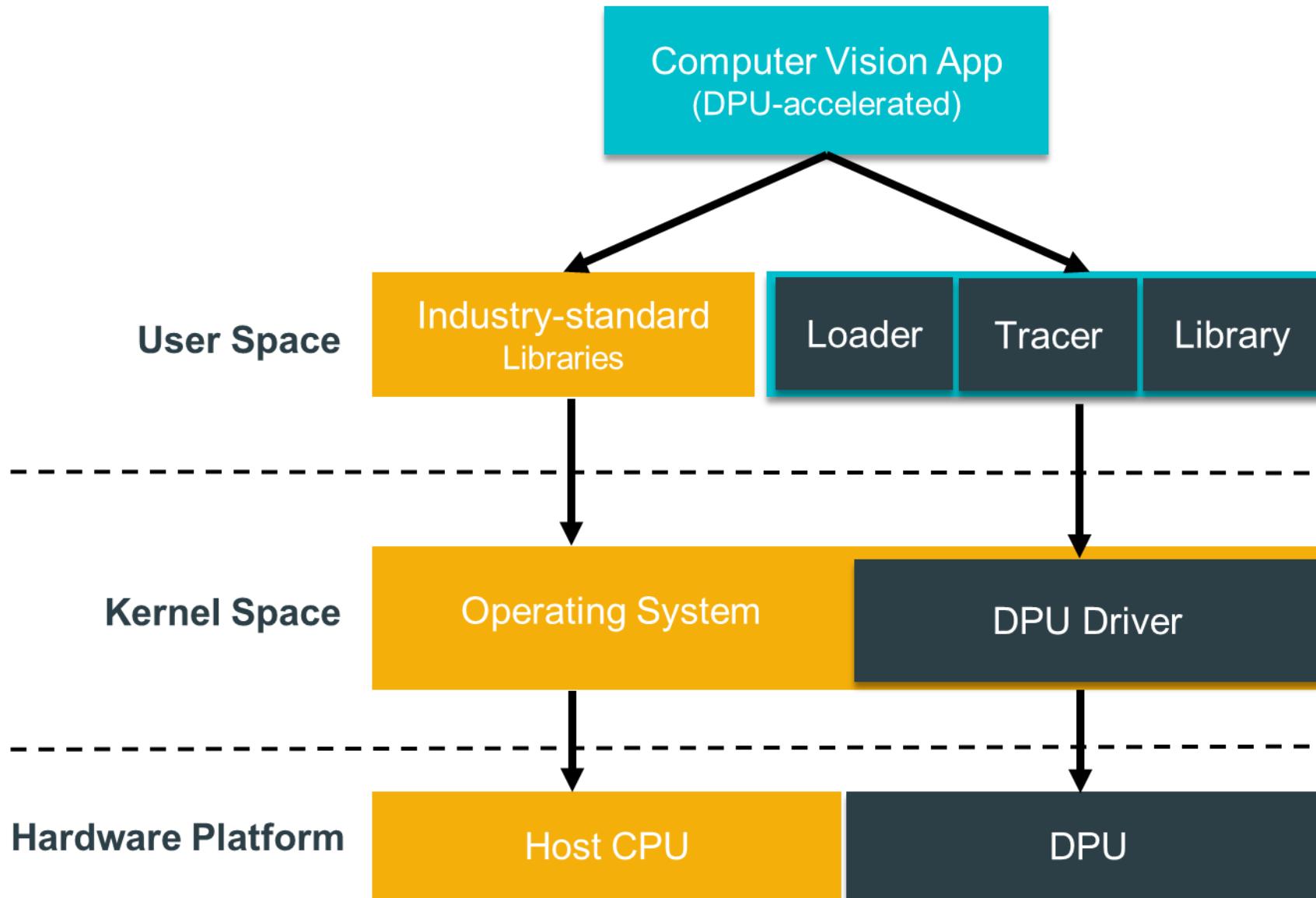
DNNDK Hybrid Compilation Model



Optimization in DNNC



DNNDK Runtime Engine



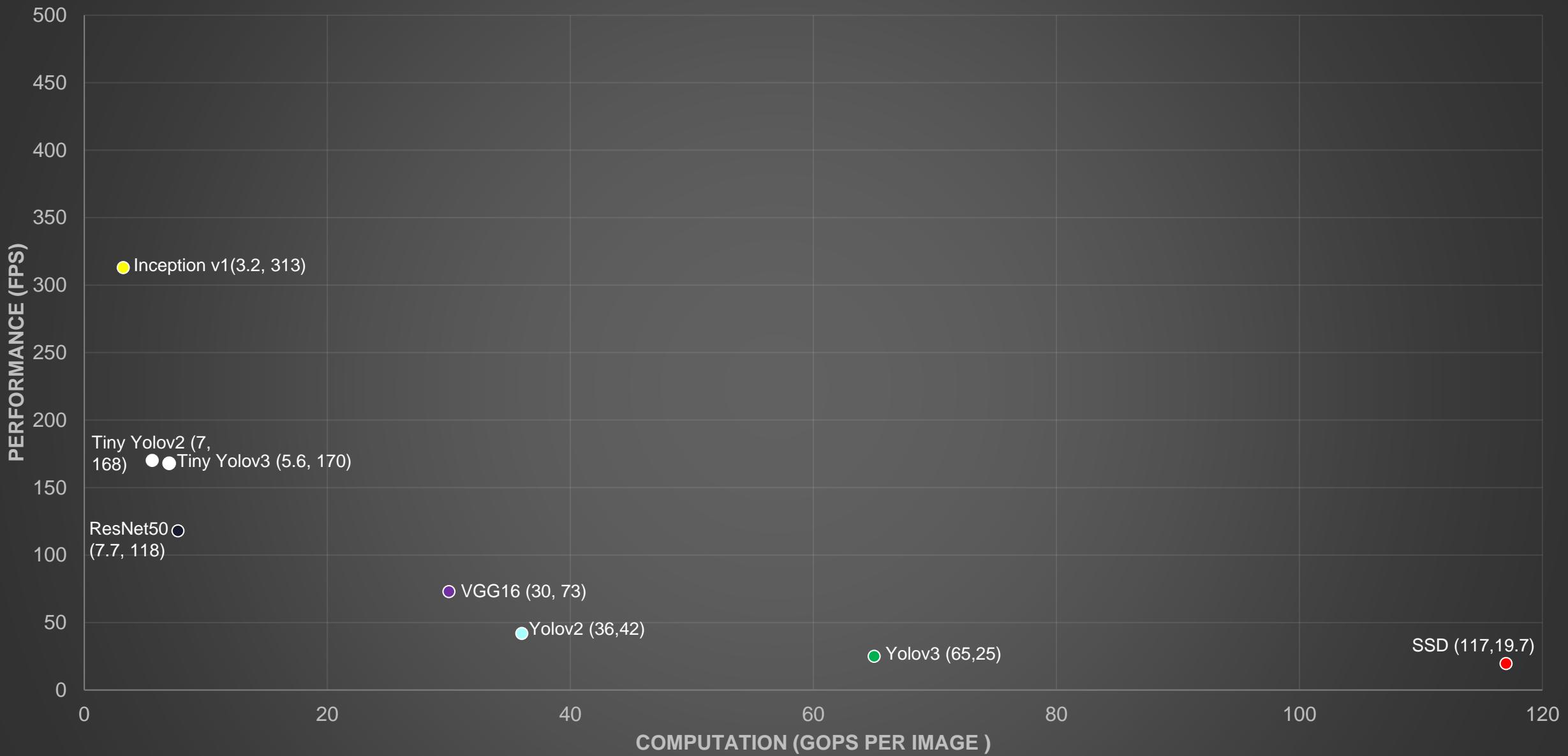
Runtime N²Cube

- Library
- Loader
- Tracer
- Driver

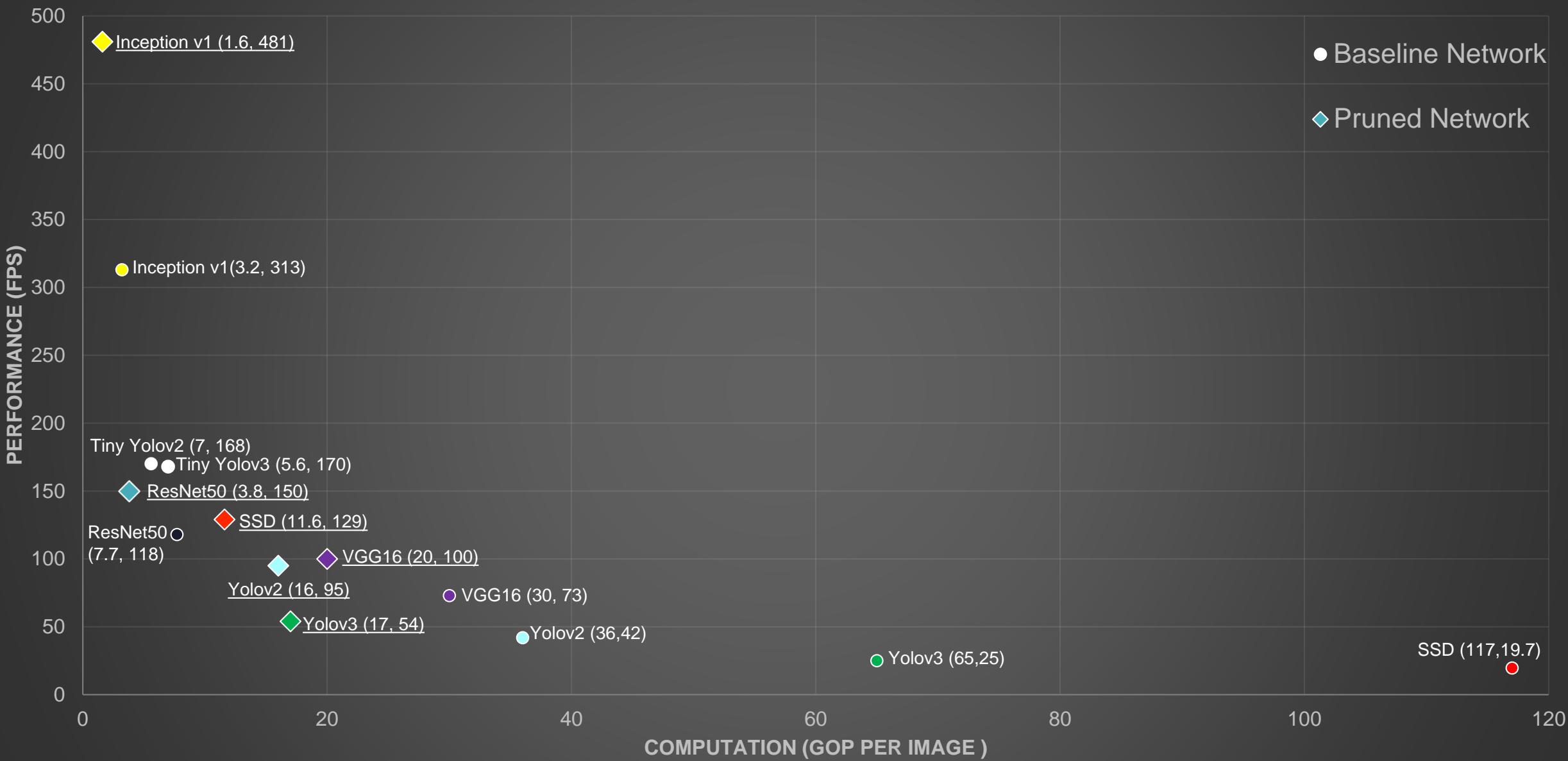
Supported Networks

Application	Module	Algorithm	Model Development	Compression	Deployment
Face	Face detection	SSD, Densebox	✓	✓	✓
	Landmark Localization	Coordinates Regression	✓	N / A	✓
	Face recognition	ResNet + Triplet / A-softmax Loss	✓	✓	✓
	Face attributes recognition	Classification and regression	✓	N / A	✓
Pedestrian	Pedestrian Detection	SSD	✓	✓	✓
	Pose Estimation	Coordinates Regression	✓	✓	✓
	Person Re-identification	ResNet + Loss Fusion	✓		
Video Analytics	Object detection	SSD, RefineDet	✓	✓	✓
	Pedestrian Attributes Recognition	GoogleNet	✓	✓	✓
	Car Attributes Recognition	GoogleNet	✓	✓	✓
	Car Logo Detection	DenseBox	✓	✓	
	Car Logo Recognition	GoogleNet + Loss Fusion	✓	✓	
	License Plate Detection	Modified DenseBox	✓	✓	✓
	License Plate Recognition	GoogleNet + Multi-task Learning	✓	✓	✓
ADAS/AD	Object Detection	SSD, YOLOv2, YOLOv3	✓	✓	✓
	3D Car Detection	F-PointNet, AVOD-FPN	✓		
	Lane Detection	VPGNet	✓	✓	✓
	Traffic Sign Detection	Modified SSD	✓		
	Semantic Segmentation	FPN	✓	✓	✓
	Driveable Space Detection	MobilenetV2-FPN	✓		
	Multi-task (Detection+Segmentation)	Deephi	✓		

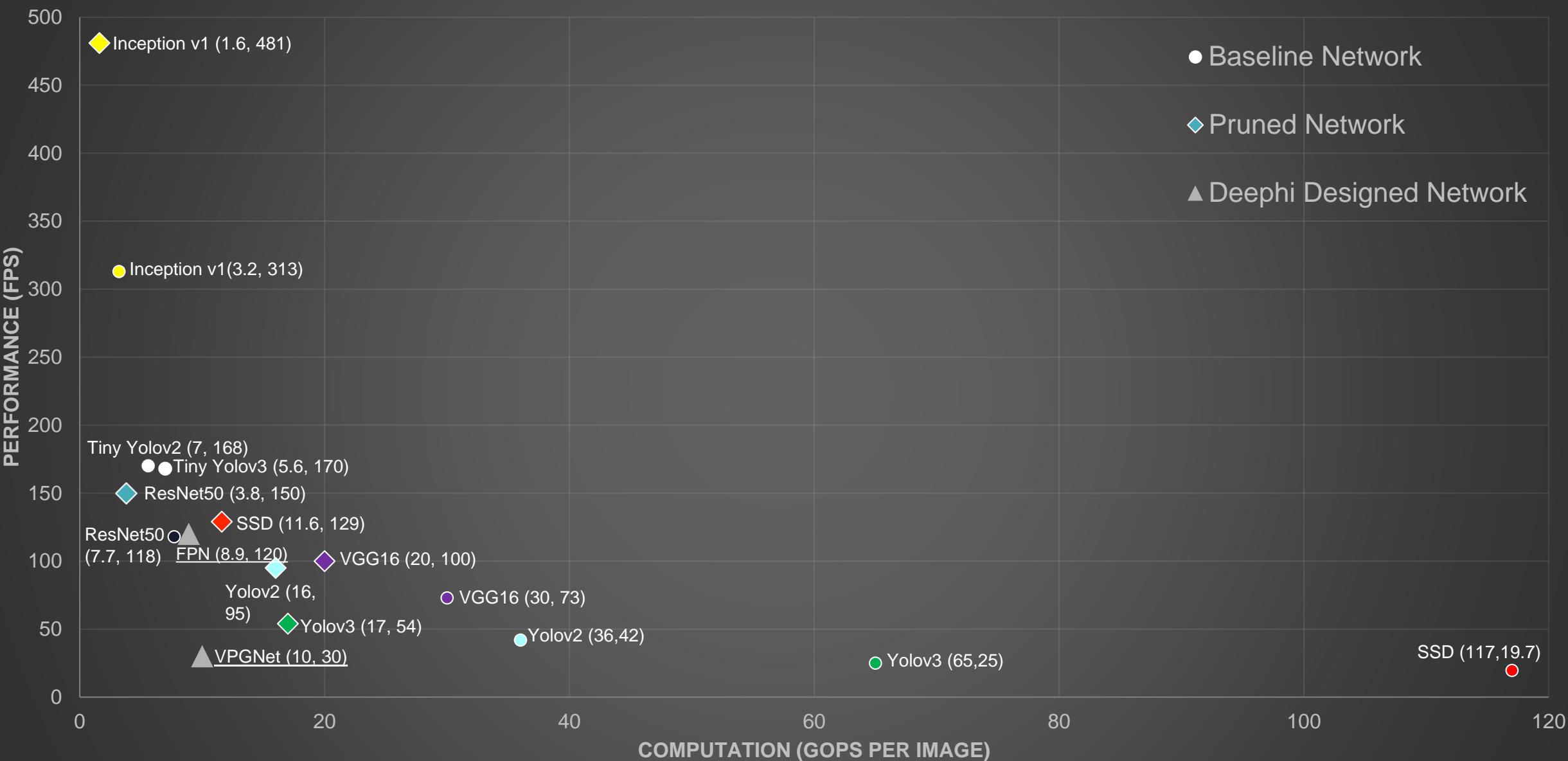
Measured Performance



Measured Performance (Cont.)

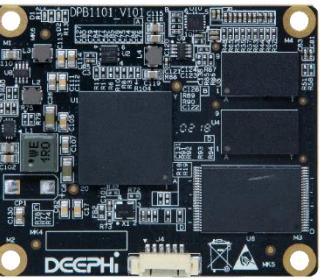


Measured Performance (Cont.)

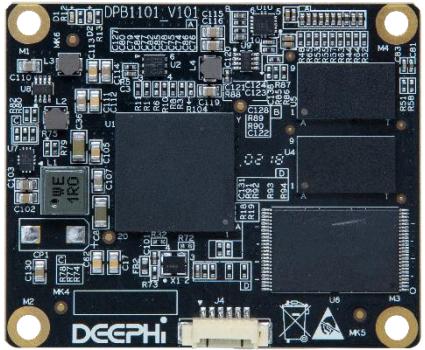


Out-of-box Supported Boards

- > **DP8000**
 - >> Z7020 SOM
- > **DP2400**
 - >> ZU9 PCIe card
- > **Deephi ZU2/3 board**
- > **Xilinx ZCU102**
- > **Xilinx ZCU104**
- > **Avnet Ultra96**



Video Surveillance ML Solutions



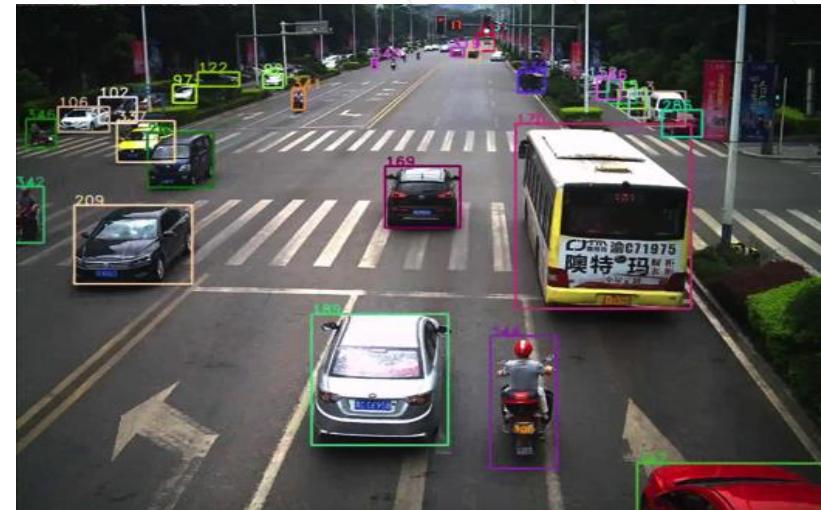
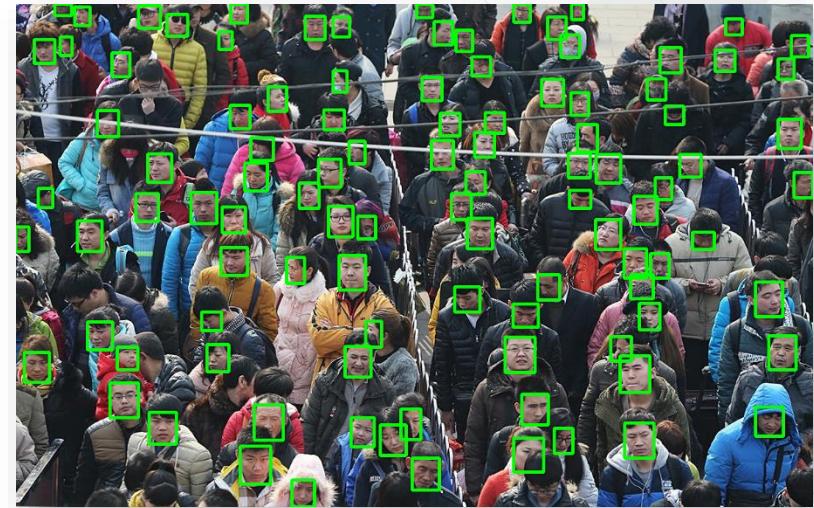
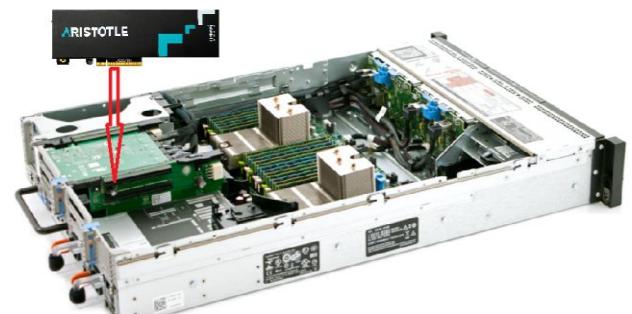
Intelligent
IP Camera Solution

Face recognition camera
with Zynq7020

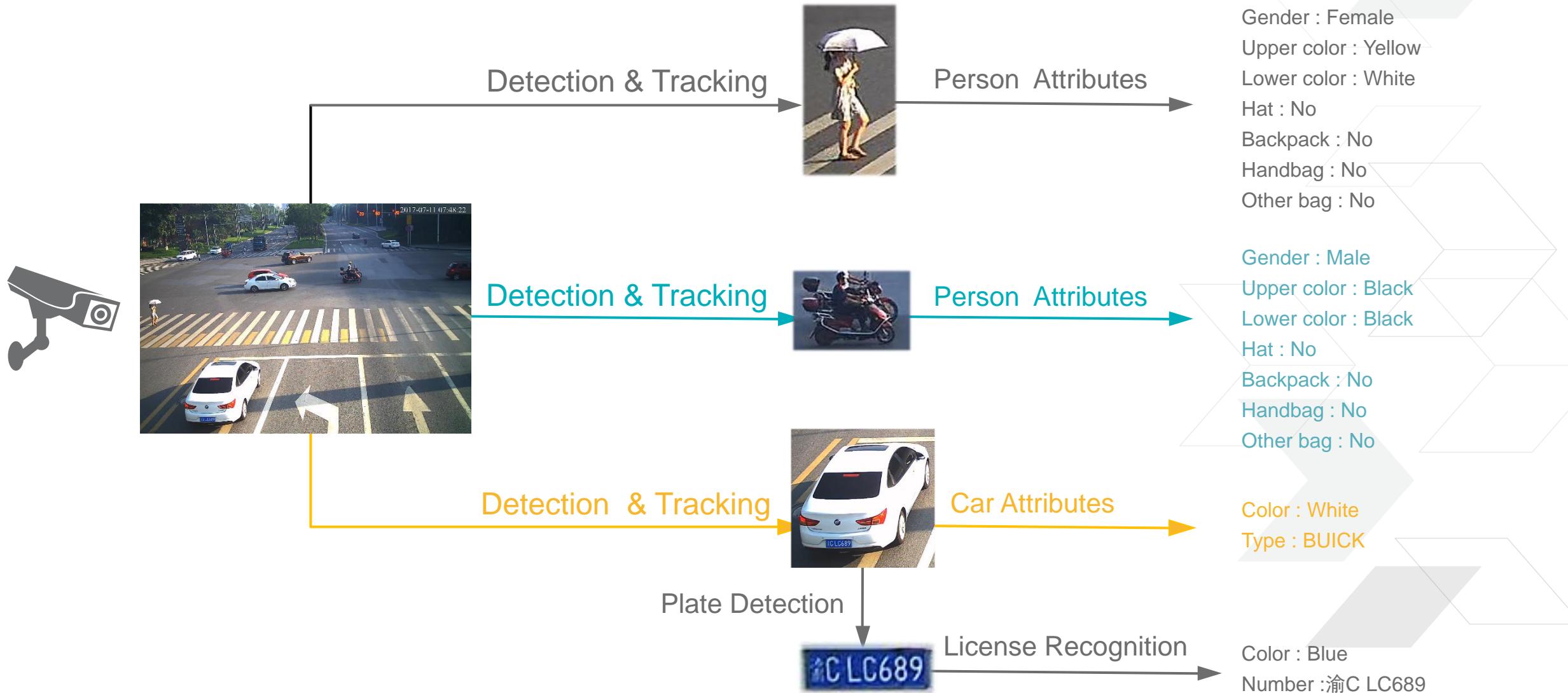


Video Analytics
Acceleration Solution

8-channel 1080P Video Analytics
with ZU9EG



Video Surveillance ML Ref Design

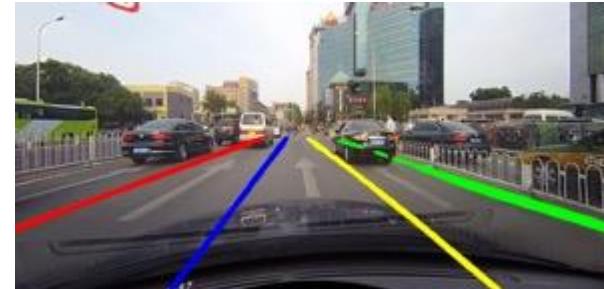


ADAS/AD ML Reference Design

2D/3D Object Detection



Lane Detection



Pedestrian Detection



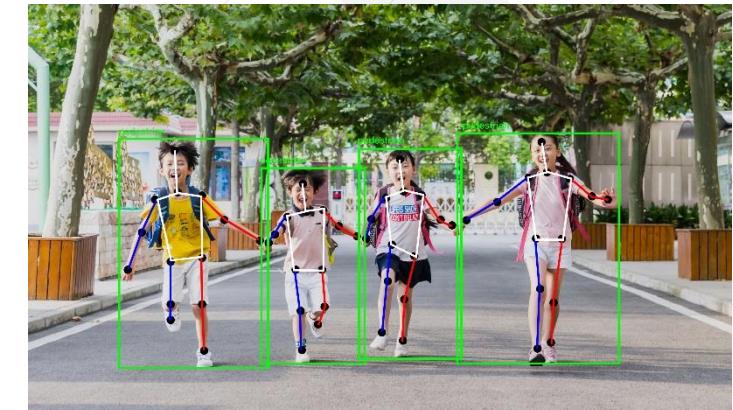
Segmentation + Detection



Segmentation



Pose Estimation



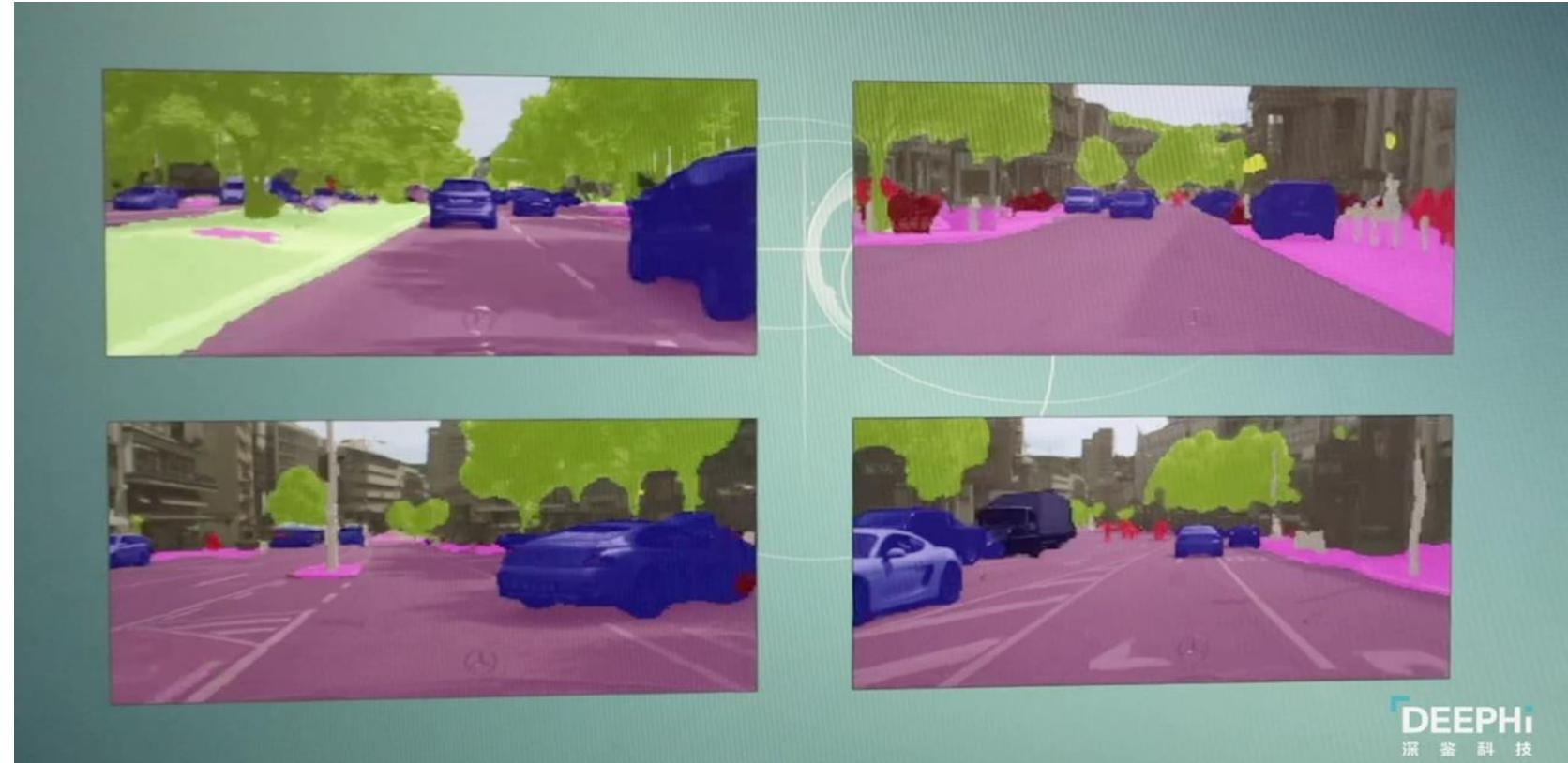
8-ch Detection Demo

- > Xilinx device
 - >> ZU9EG
- > Network
 - >> SSD compact version
- > Input image size to DPU
 - >> 480 * 360
- > Operations per frame
 - >> 4.9G
- > Performance
 - >> 30fps per channel



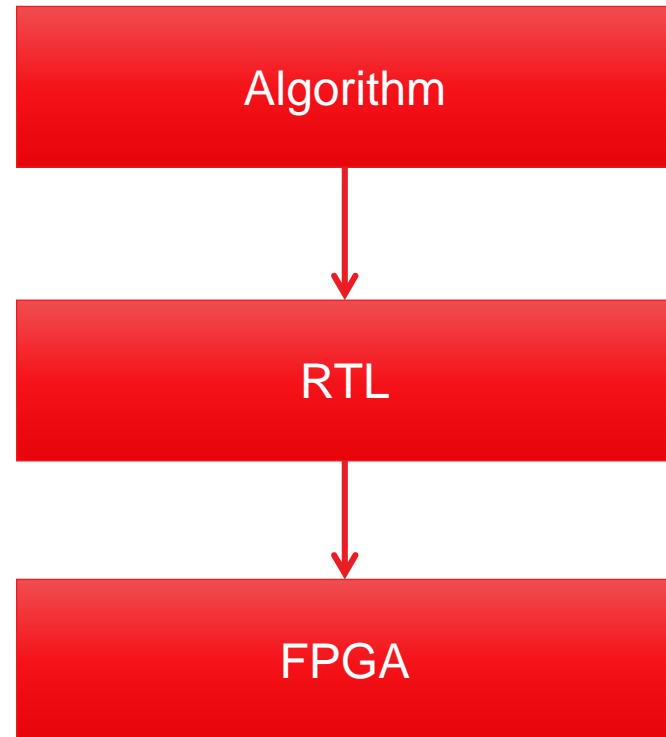
4-ch Segmentation + Detection Demo

- > Xilinx device
 - >> ZU9EG
- > Network
 - >> FPN compact version
 - >> SSD compact version
- > Input image size to DPU
 - >> FPN – 512 * 256
 - >> SSD – 480 * 360
- > Operations per frame
 - >> FPN – 9G
 - >> SSD – 4.9G
- > Performance
 - >> 15fps per channel

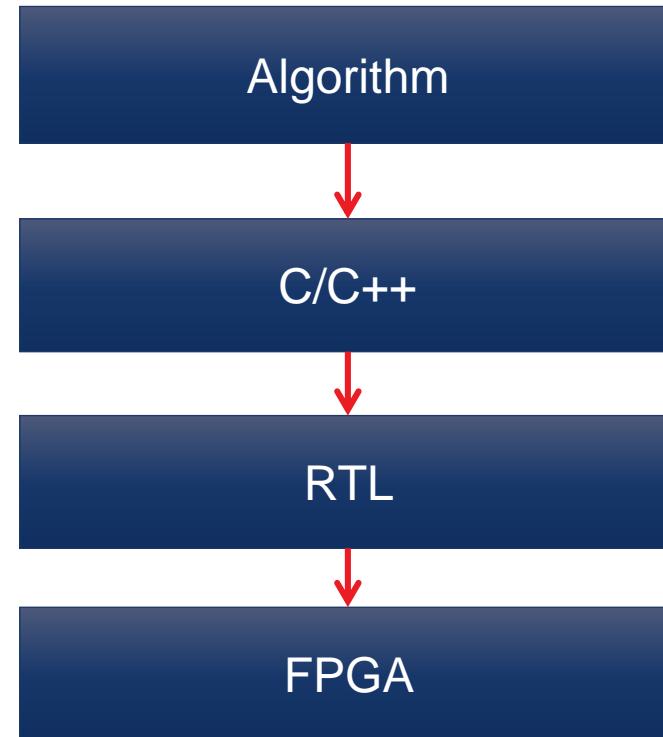


ML Development with Deephi Solution

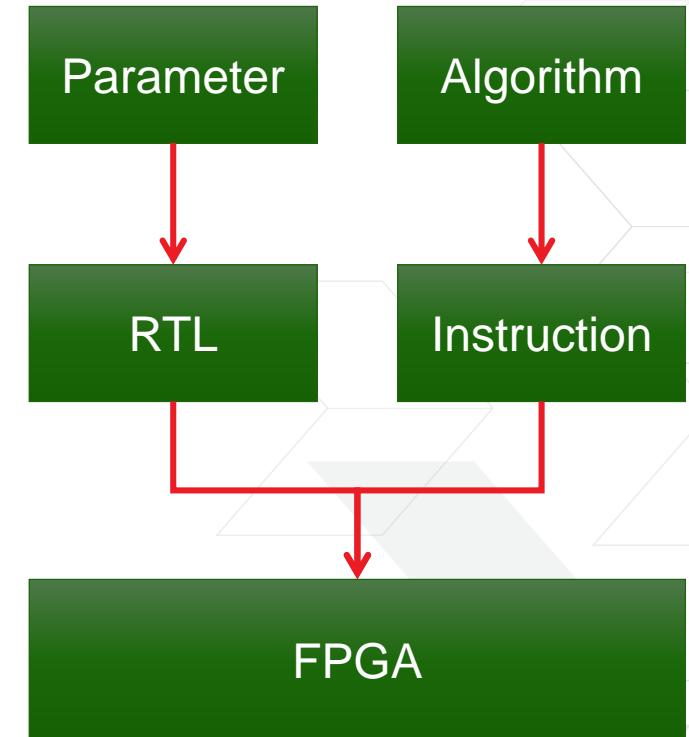
Development Method



Traditional



OpenCL/HLS



DeePhi

Two Development Flows of Using Deephi DPU IP

> Vivado & SDK

- »> Traditional flow
- »> Bottom up approach
- »> Suitable for FPGA designer
- »> Fine-grained customization

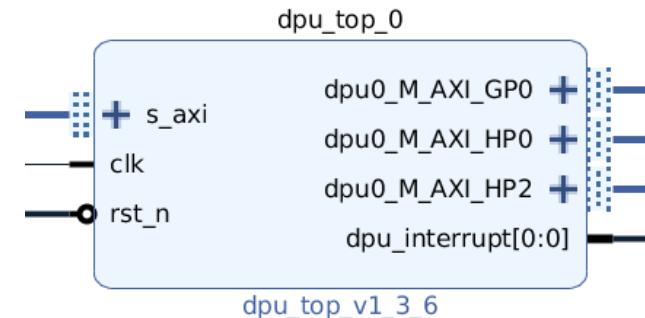
> SDSoC

- »> New high-level abstraction flow
- »> Top down approach
- »> Suitable for algorithm & software developer
- »> Higher Productivity

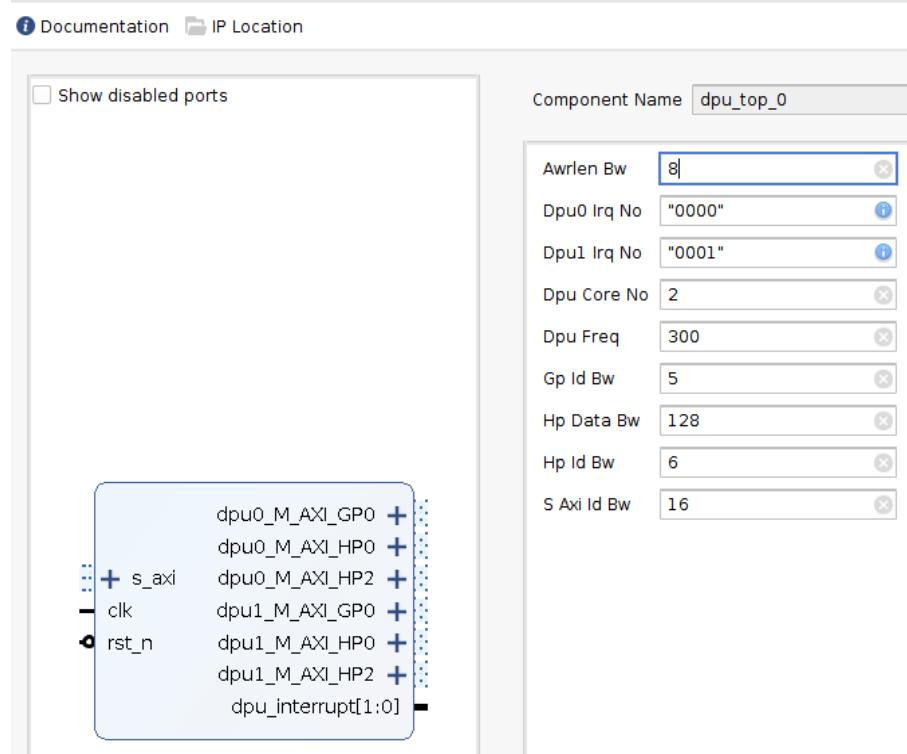
HW Integration with Vivado IPI

> Steps

- >> Add DPU IP into repository
- >> Add DPU into block design
- >> Configure DPU parameters
- >> Connect DPU with MPSoC(for reference)
 - M_AXI_HP0 <-> S_AXI_HP0_FPD (ZYNQ)
 - M_AXI_HP2 <-> S_AXI_HP1_FPD (ZYNQ)
 - M_Axi_GP0 <-> S_AXI_LPD(ZYNQ)
 - s_axi <-> M_AXI_HPM0_LPD (ZYNQ)
- >> Assign Reg address for DPU in address editor
 - e.g. 0x80000000, 4K space for one DPU



dpu_top_v1_3_6 (1.3.6)



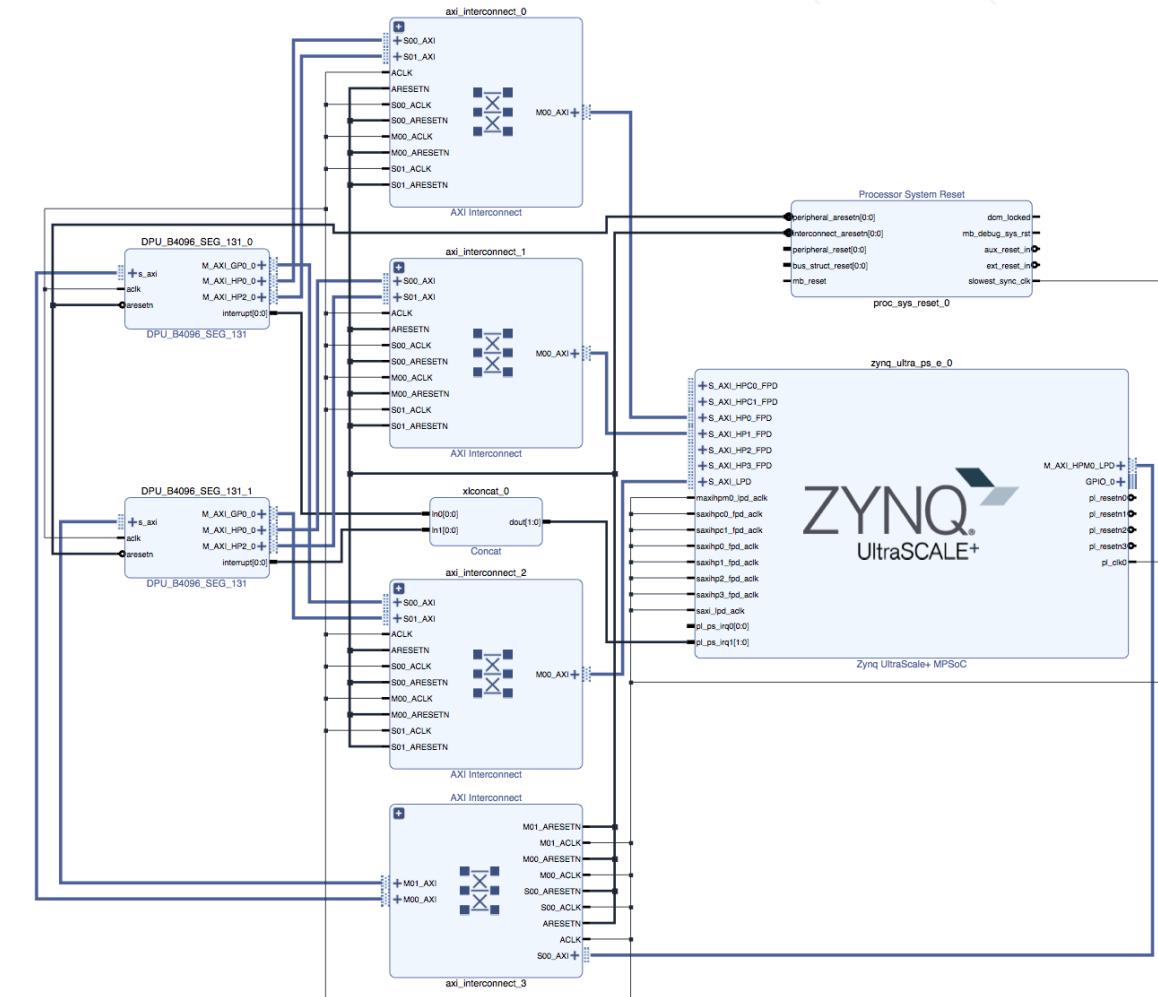
HW Integration with Vivado IPI (Cont.)

> Steps(Cont.)

- >> Create top wrapper
- >> Generate bitstream
- >> Generate BOOT.BIN using Petalinux etc.

> Note

- >> The port data width is consistent with DPU data width
- >> For frequency > 333MHz, clock wizard is needed between MPSoC and DPU
- >> Interrupt configuration was shown in binary.
[3]: 0- pl_ps_irq0 ; 1- pl_ps_irq1
[2:0]: interrupt number 0~7



SW Integration with SDK

> Device tree configuration

- >> set interrupt number according to block design
- >> set core-num

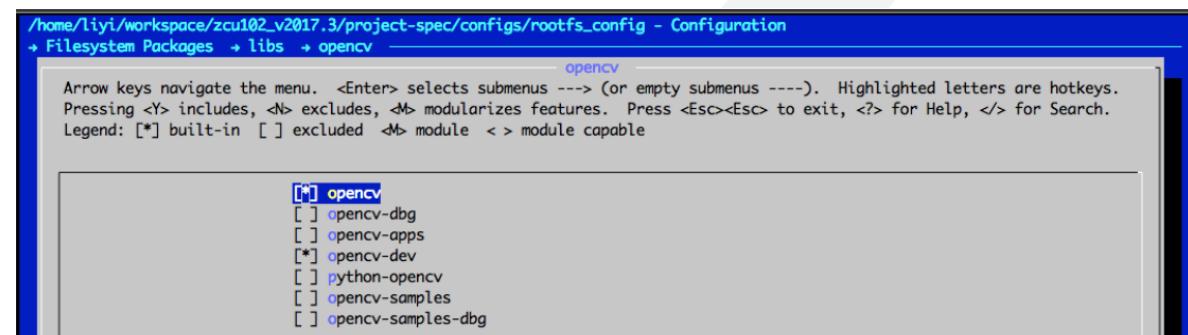
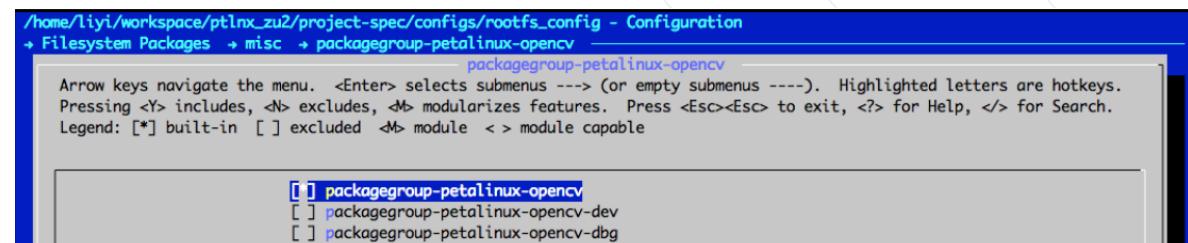
> OpenCV configuration

- >> Enable in Filesystem Packages -> misc or libs

> Driver and DNNDK lib

- >> Provide kernel information & OpenCV version to Deephi
- >> Deephi will provide driver and DNNDK package with install script
- >> Install driver and DNNDK lib

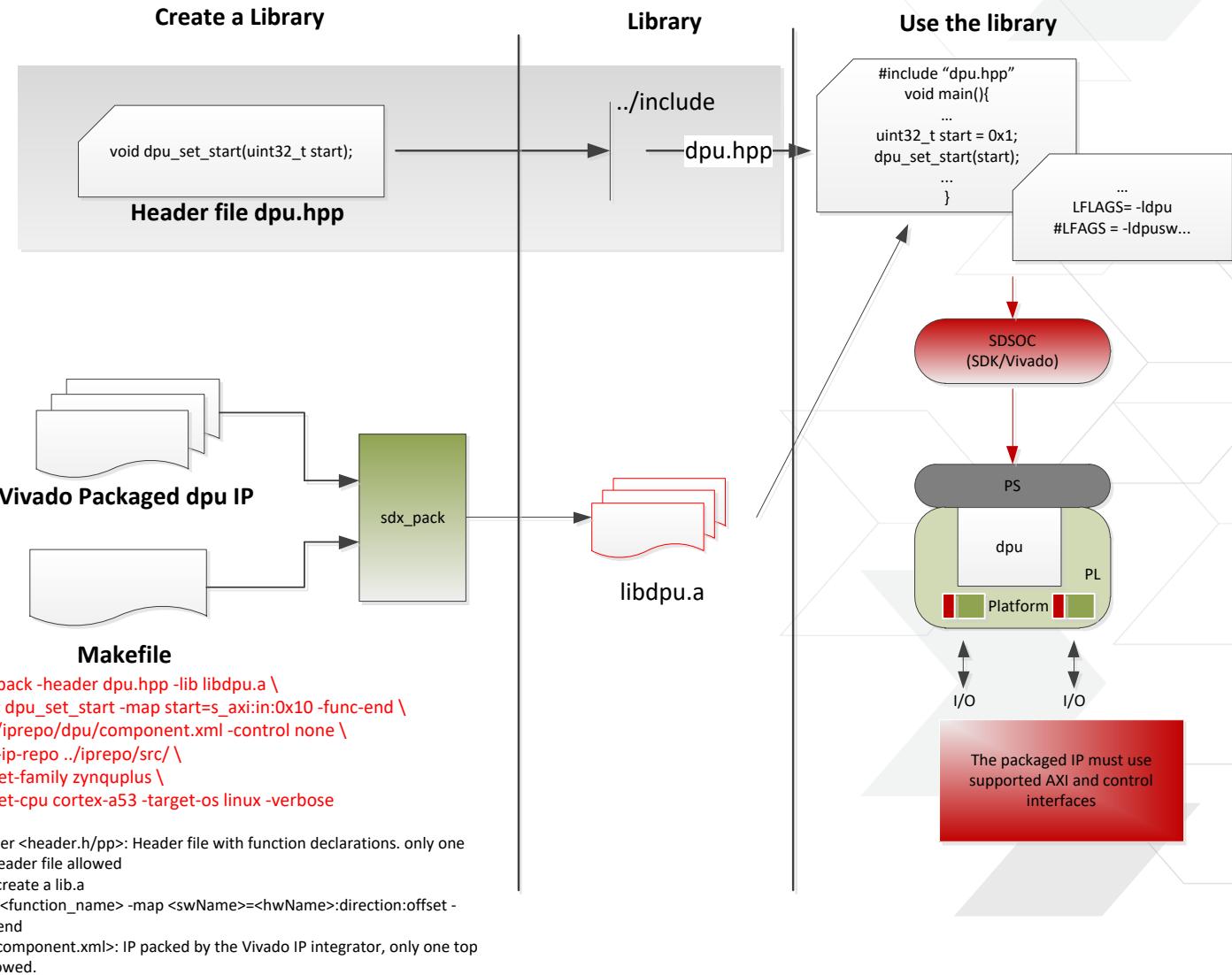
```
amba {  
...  
dpu@80000000 {  
  
    compatible = "deephisi, dpu";  
    interrupt-parent = <&intc>;  
    interrupts = <0x0 106 0x1 0x0 107 0x1>;  
    reg = <0x0 0x80000000 0x0 0x700>;  
    memory = <0x60000000 0x20000000>;  
    core-num = <0x2>;  
};  
....  
}
```



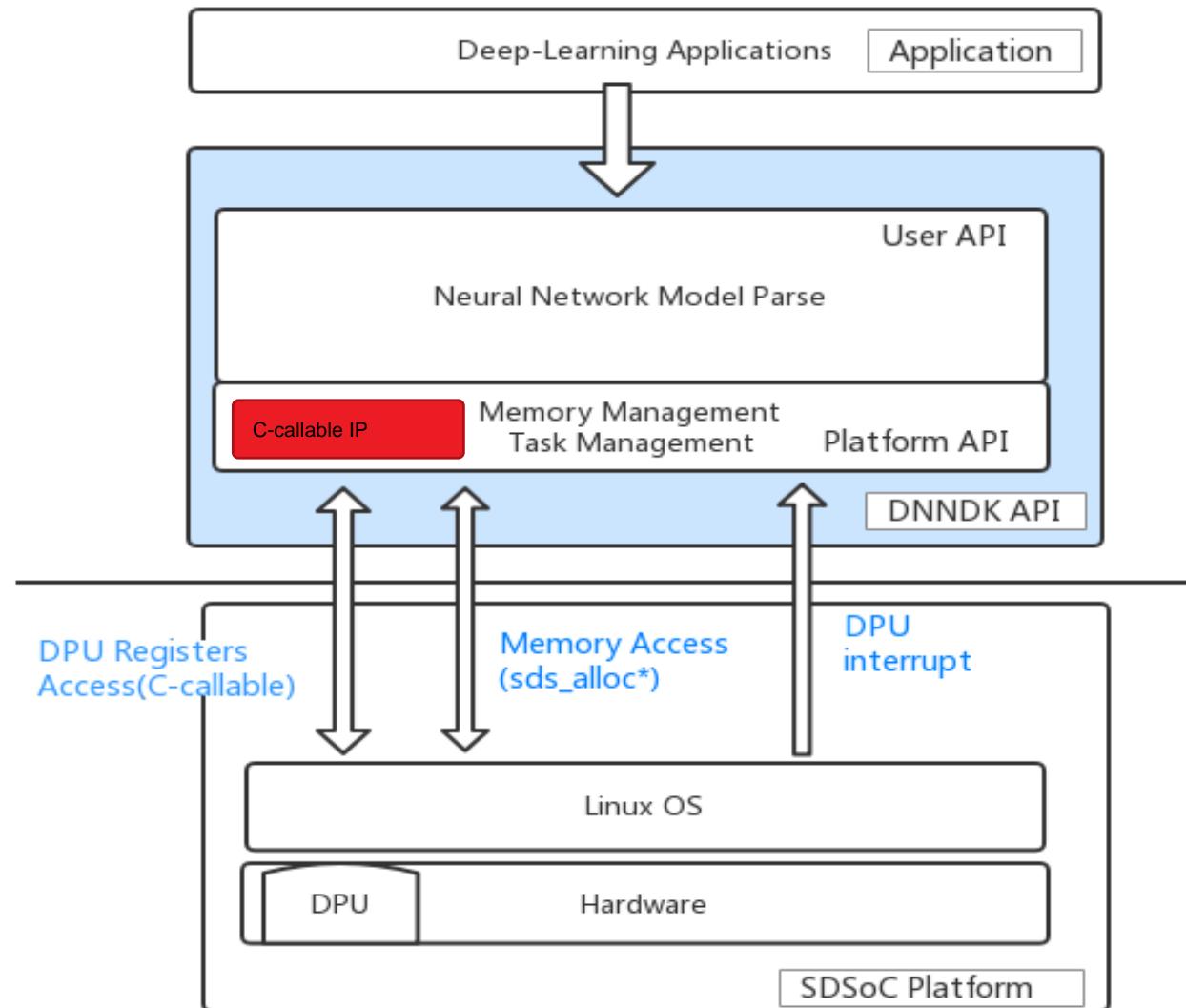
HW Integration with C-callable IP

> Steps

- >> Create header file
- >> Package IP in Vivado
- >> Create Makefile to generate *.a
- >> Configure DPU parameters
- >> Build application software

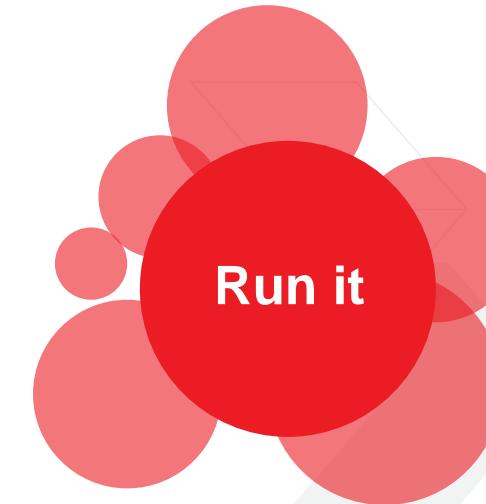
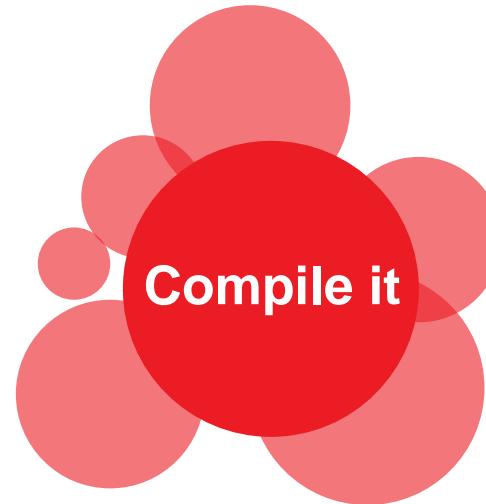


Deephi DPU IP Integration with SDSoc



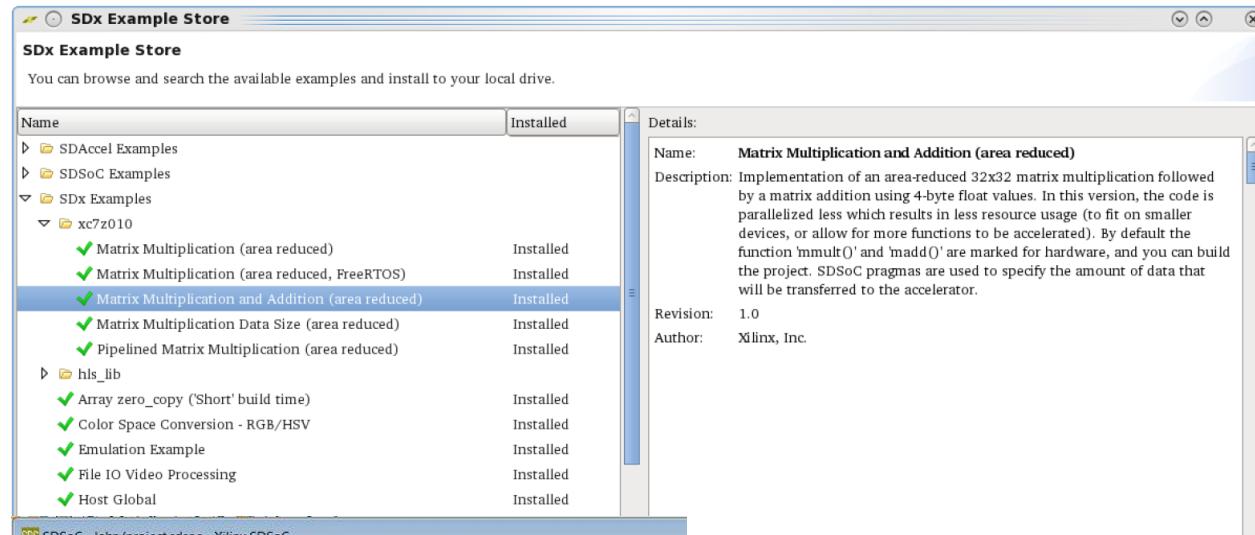
How to Use DNNK in SDSoc

Only 3 steps!



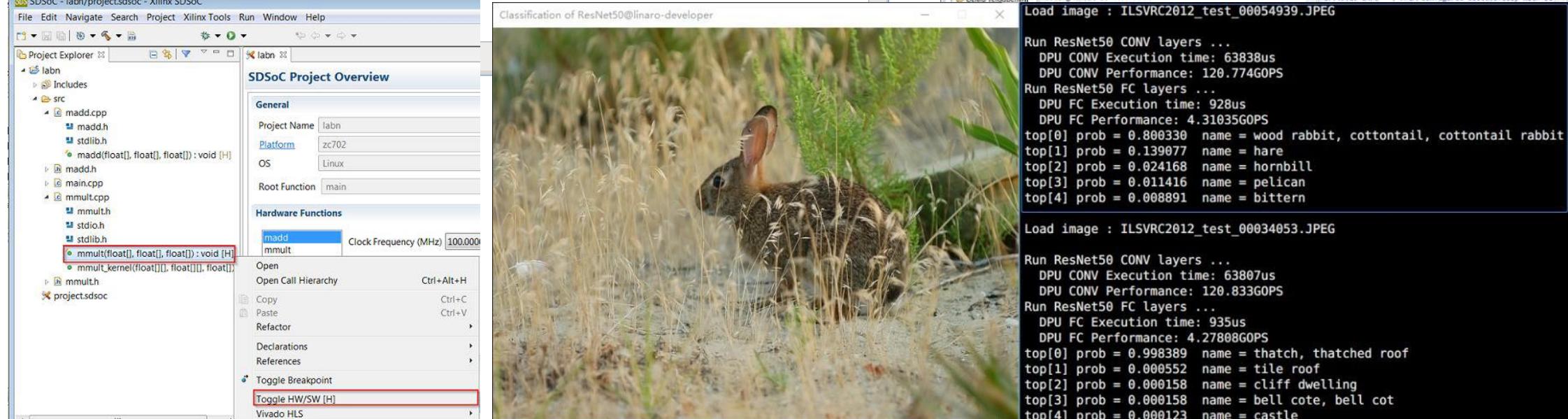
Software define development

Resnet50 Example with C-callable DPU IP in SDSoc



The screenshot shows the SDx Example Store interface. Under the 'SDx Examples' section, the 'Matrix Multiplication and Addition (area reduced)' example is selected. The details panel shows the following information:

- Name: Matrix Multiplication and Addition (area reduced)
- Description: Implementation of an area-reduced 32x32 matrix multiplication followed by a matrix addition using 4-byte float values. In this version, the code is parallelized less which results in less resource usage (to fit on smaller devices, or allow for more functions to be accelerated). By default the function 'mmult()' and 'madd()' are marked for hardware, and you can build the project. SDSoc pragmas are used to specify the amount of data that will be transferred to the accelerator.
- Revision: 1.0
- Author: Xilinx, Inc.



The SDSoc IDE interface includes a Project Explorer showing files like madd.cpp, main.cpp, mmult.cpp, and stdlib.h. A terminal window displays the output of a ResNet50 classification process. The first image shows a rabbit in a field, and the terminal output lists the top 5 classification results:

```
Classification of ResNet50@linaro-developer
Load image : ILSVRC2012_test_00054939.JPG
Run ResNet50 CONV layers ...
DPU CONV Execution time: 63838us
DPU CONV Performance: 120.774GOPS
Run ResNet50 FC layers ...
DPU FC Execution time: 928us
DPU FC Performance: 4.31035GOPS
top[0] prob = 0.800330 name = wood rabbit, cottontail, cottontail rabbit
top[1] prob = 0.139077 name = hare
top[2] prob = 0.024168 name = hornbill
top[3] prob = 0.011416 name = pelican
top[4] prob = 0.008891 name = bittern

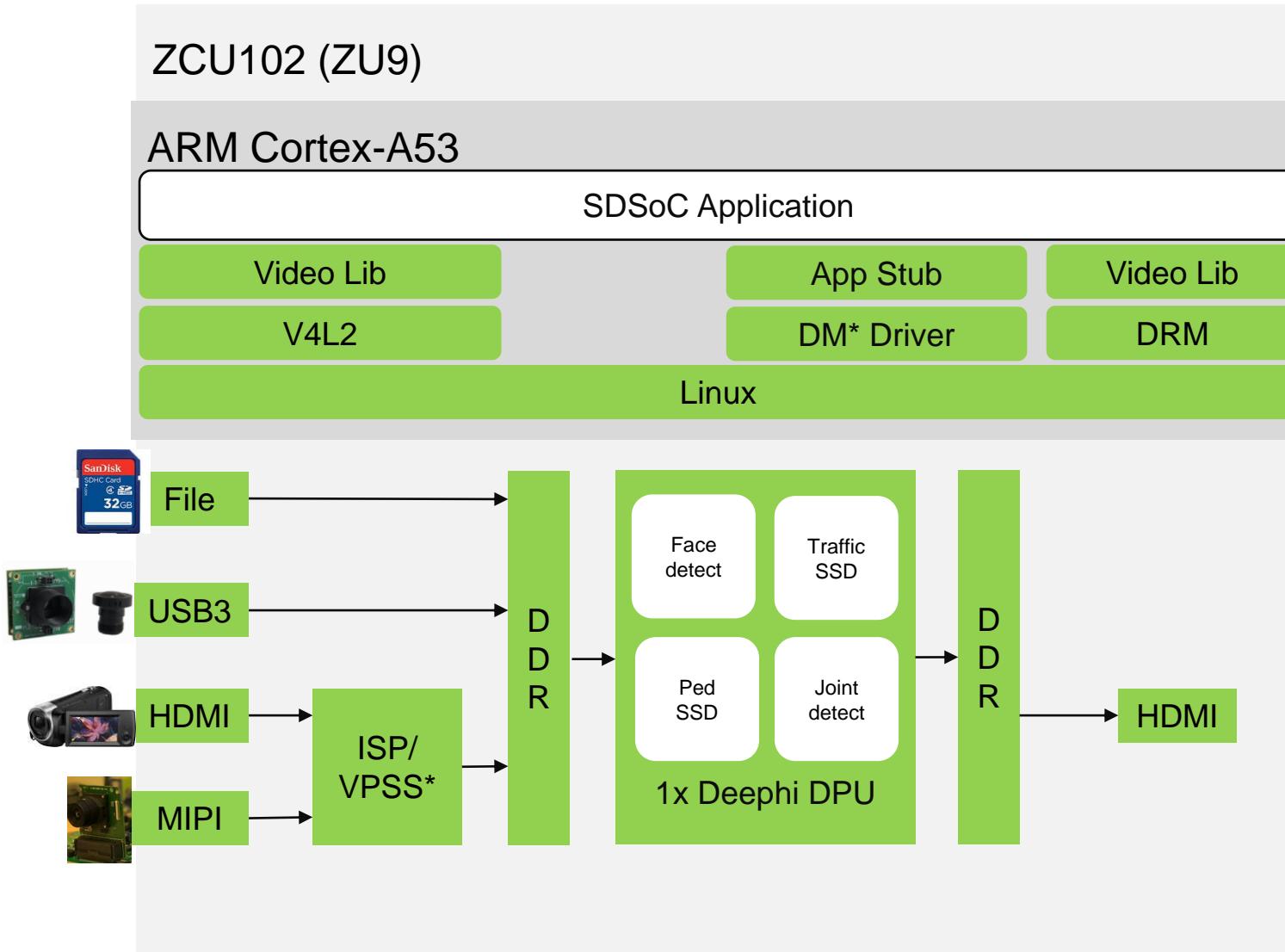
Load image : ILSVRC2012_test_00034053.JPG
Run ResNet50 CONV layers ...
DPU CONV Execution time: 63807us
DPU CONV Performance: 120.833GOPS
Run ResNet50 FC layers ...
DPU FC Execution time: 935us
DPU FC Performance: 4.27808GOPS
top[0] prob = 0.998389 name = thatch, thatched roof
top[1] prob = 0.000552 name = tile roof
top[2] prob = 0.000158 name = cliff dwelling
top[3] prob = 0.000158 name = bell cote, bell cot
top[4] prob = 0.000123 name = castle
```

A Long Time for Every Build?

- > SDSoc compiler compares the new data-motion network with the last one
- > If the same, vpl will not be called to rerun syn & impl
- > It only takes a few minutes if –
 - >> Use the same C-callable IP library
 - >> Use the same platform
 - >> Use the same project setting

```
Generating data motion network
INFO: [DMAAnalysis 83-4494] Analyzing hardware accelerators...
INFO: [DMAAnalysis 83-4497] Analyzing callers to hardware accelerators...
INFO: [DMAAnalysis 83-4444] Scheduling data transfer graph for partition 0
INFO: [DMAAnalysis 83-4446] Creating data motion network hardware for partition 0
INFO: [DMAAnalysis 83-4448] Creating software stub functions for partition 0
INFO: [DMAAnalysis 83-4450] Generating data motion network report for partition 0
INFO: [DMAAnalysis 83-4454] Rewriting caller code
Skipping block diagram (BD), address map, port information and device registration for partition 0
Rewrite caller functions
```

Multiple Sensors & Networks with C-callable DPU IP



- SDSoc 2018.2 Linux
- 4 CNN models
 - Face detect, Joint detect, Traffic SSD, Ped SSD
 - 30, 12, 15, 13 FPS respectively
- 3 Live inputs + file / HDMI output
- Under 10 Watts

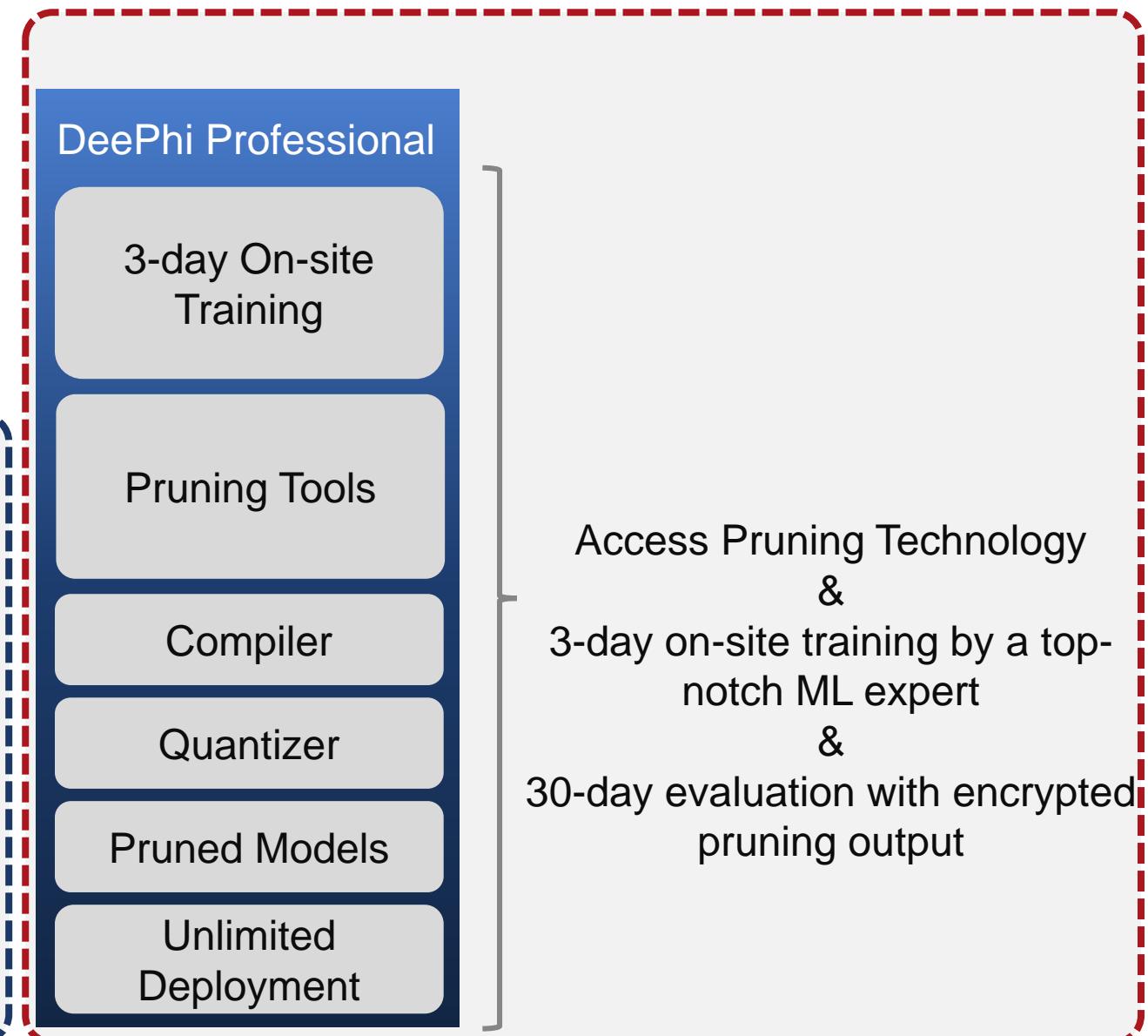
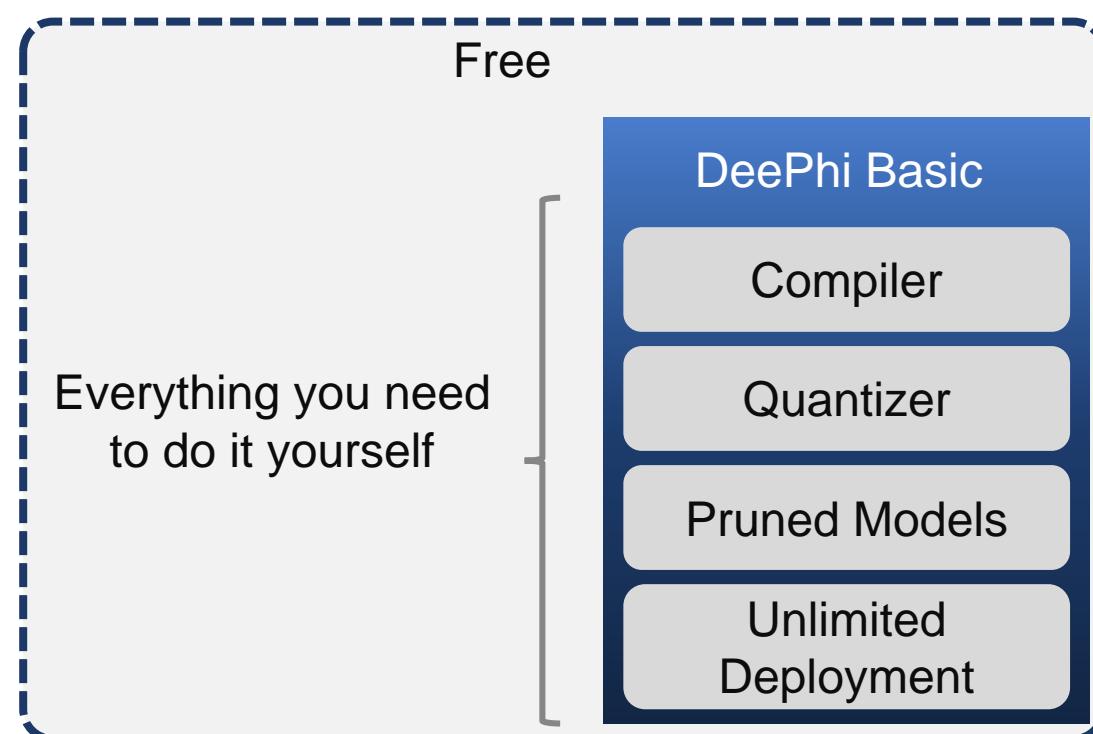


Availability

Basic and Professional Editions

Pricing TBD

- > **Timeframe**
 - » Early Access: Now
 - » Public Access: Jan 2019
- > **To be available on AWS in Cloud Editions**
- > **Add-on design service**



Availability

> DNNDK

- »> For DP8000(Z7020)/DP8020(ZU2) board, download from Deephi website
- »> For other boards, separate package upon request
- »> For pruning tool, separate upon request

> Demos & Ref Designs

- »> General: Resnet50, Googlenet, VGG16, SSD, Yolo v2-v3, Tiny Yolo v2-v3, Mobilenet v2 etc..
- »> Video surveillance: face detection & traffic structure
- »> ADAS/AD: multi-channel detection & segmentation
- »> C-callable DPU IP with SDSoc: Resnet50, Quad networks(Pedstrian, Pose, Face, Traffic)

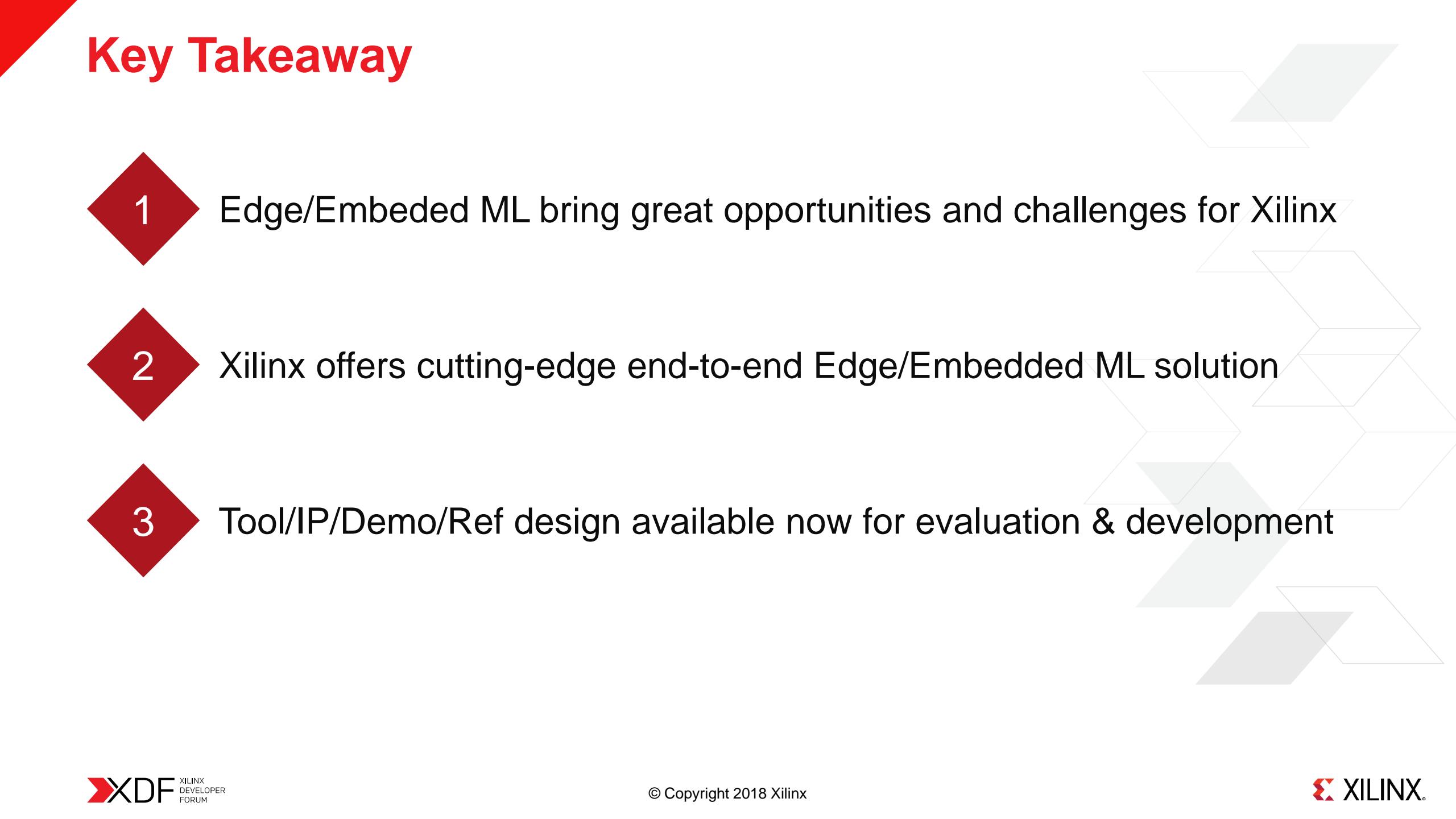
> Documentation

- »> DNNDK user guide
- »> C-callable DPU IP w SDSoc user guide
- »> DPU IP system integration user guide (Work in progress)
- »> Pruning user guide (Work in progress)

> Request or Inquiry

- »> Please contact Andy Luo, andy.luo@xilinx.com

Key Takeaway

- 
- 1 Edge/Embedded ML bring great opportunities and challenges for Xilinx
 - 2 Xilinx offers cutting-edge end-to-end Edge/Embedded ML solution
 - 3 Tool/IP/Demo/Ref design available now for evaluation & development

