XDF XILINX DEVELOPER FORUM

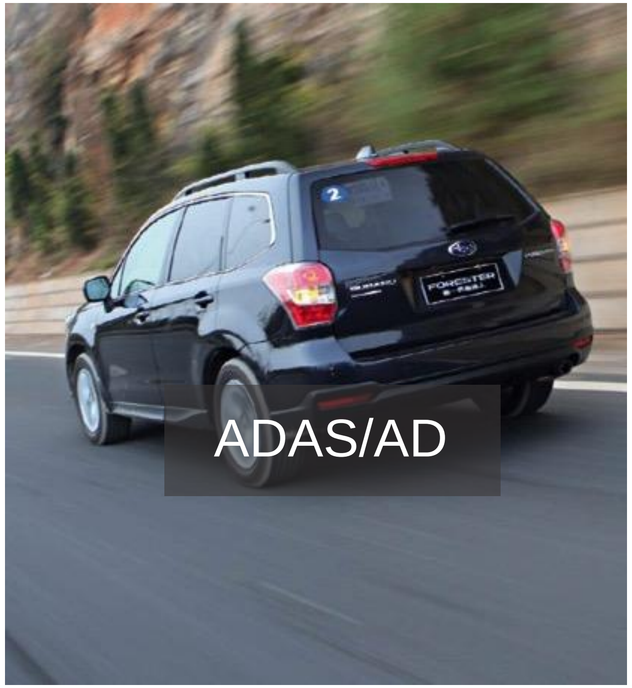
Machine learning for embedded deep dive

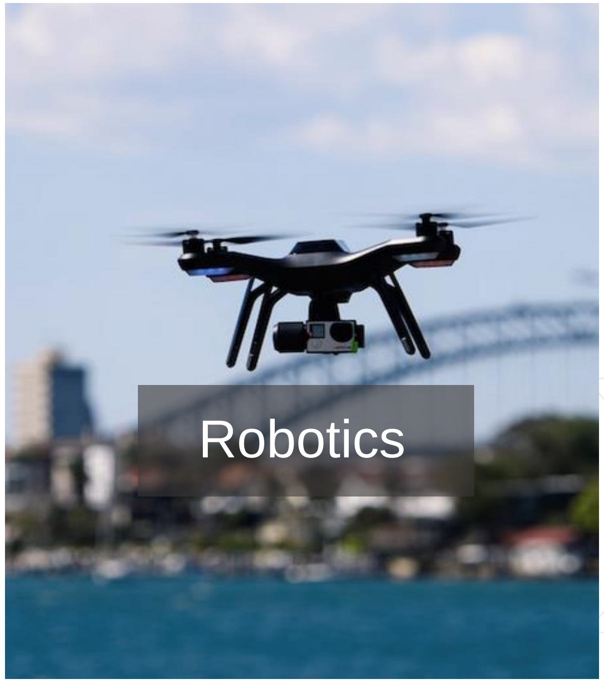
Presented By

Jingxui Lui
Sr. Product Marketing Manager

XILINX

Key Machine Learning Applications for Xilinx





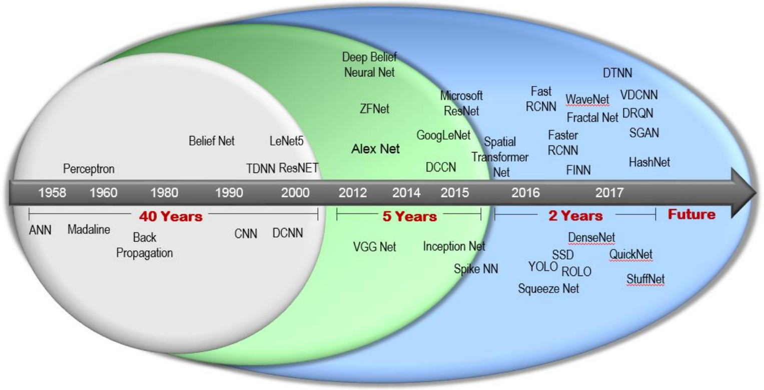
Cloud ML

And there are many more …

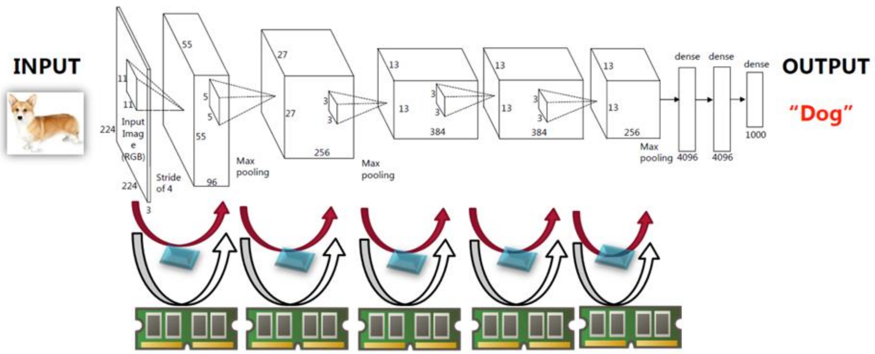
Edge ML

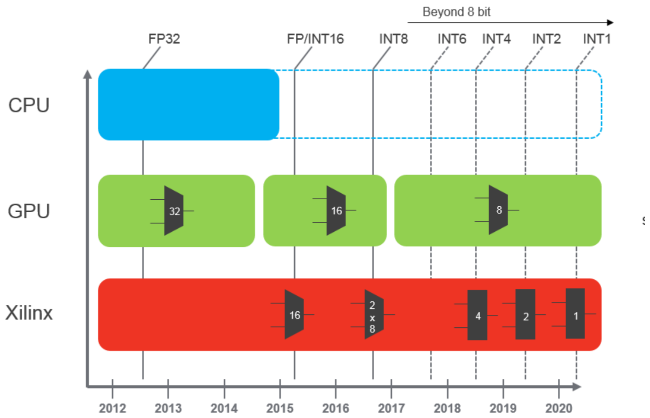
Xilinx Value Proposition in Edge/Embedded ML

1 **Only HW/SW configurable device for fast changing networks**

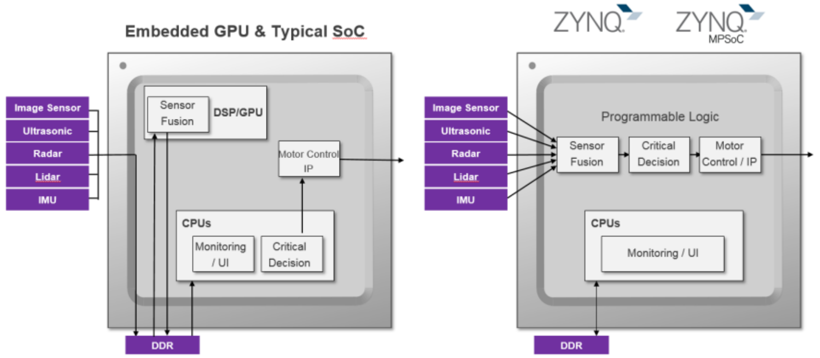


2 **High performance / low power with custom internal memory hierarchy**

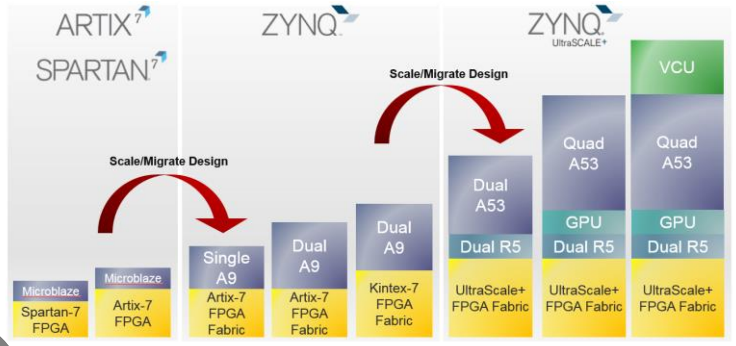




3 **Future proof to lower precisions**



4 **Low latency end-to-end**



5 **Scalable device family for different applications**

XILINX

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

OpenCV

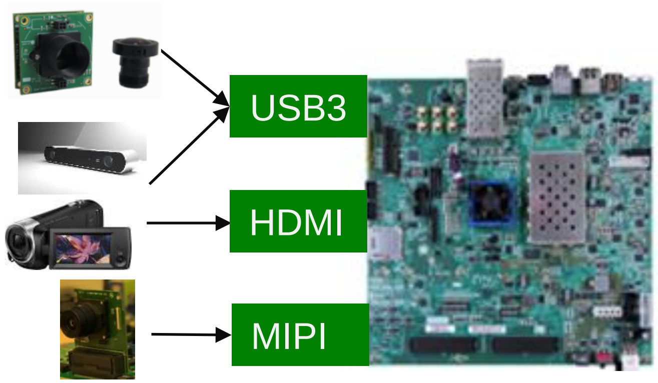
gstreamer

Machine Learning

Development tools

SDSoC™ Environment

Platforms



DEEPHi
深鉴科技

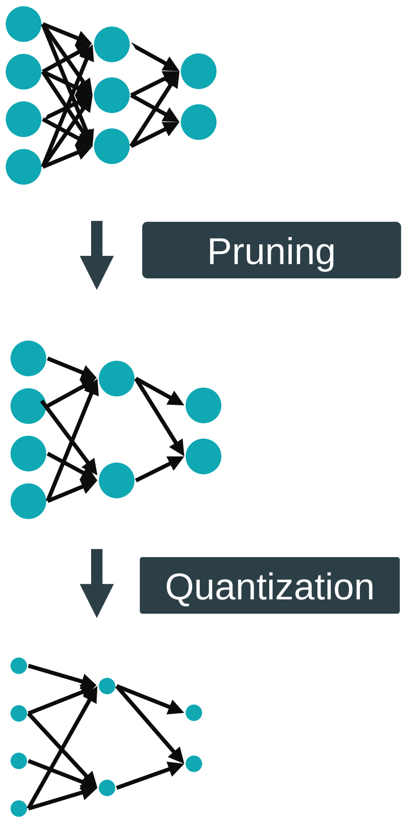
Xilinx Announces the Acquisition of DeePhi Tech

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:**

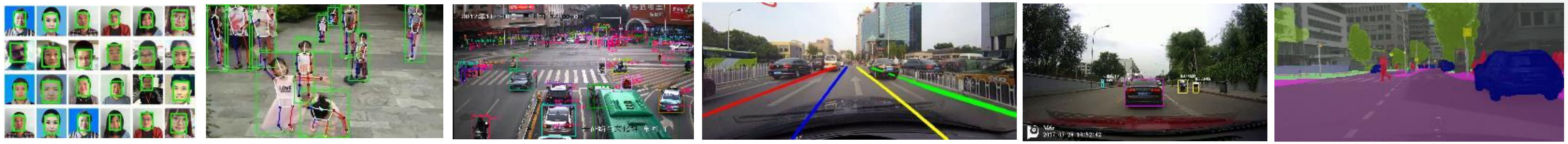
>> Reduce DL accelerator footprint into smaller devices

>> 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



Face detection | Pose estimation | Video analytics | Lane detection | Object detection | Segmentation

Framework

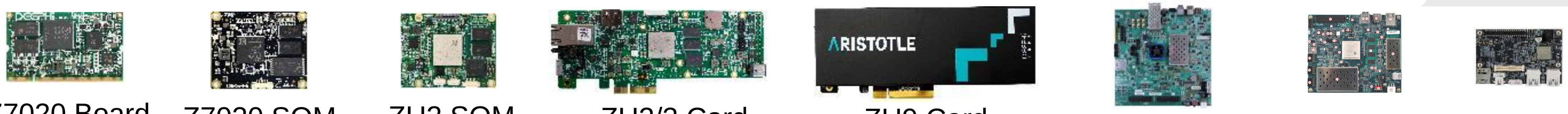
Caffe | Darknet | TensorFlow

Tools & IP

DEEPHi DNNDK | DEEPHi DPU

- **Compression**
  - Pruning
  - Quantization
- **Compilation**
  - Compiler
  - Assembler
- **Runtime**
  - Core API
  - Loader
  - Driver
  - Profiler

HW Platforms

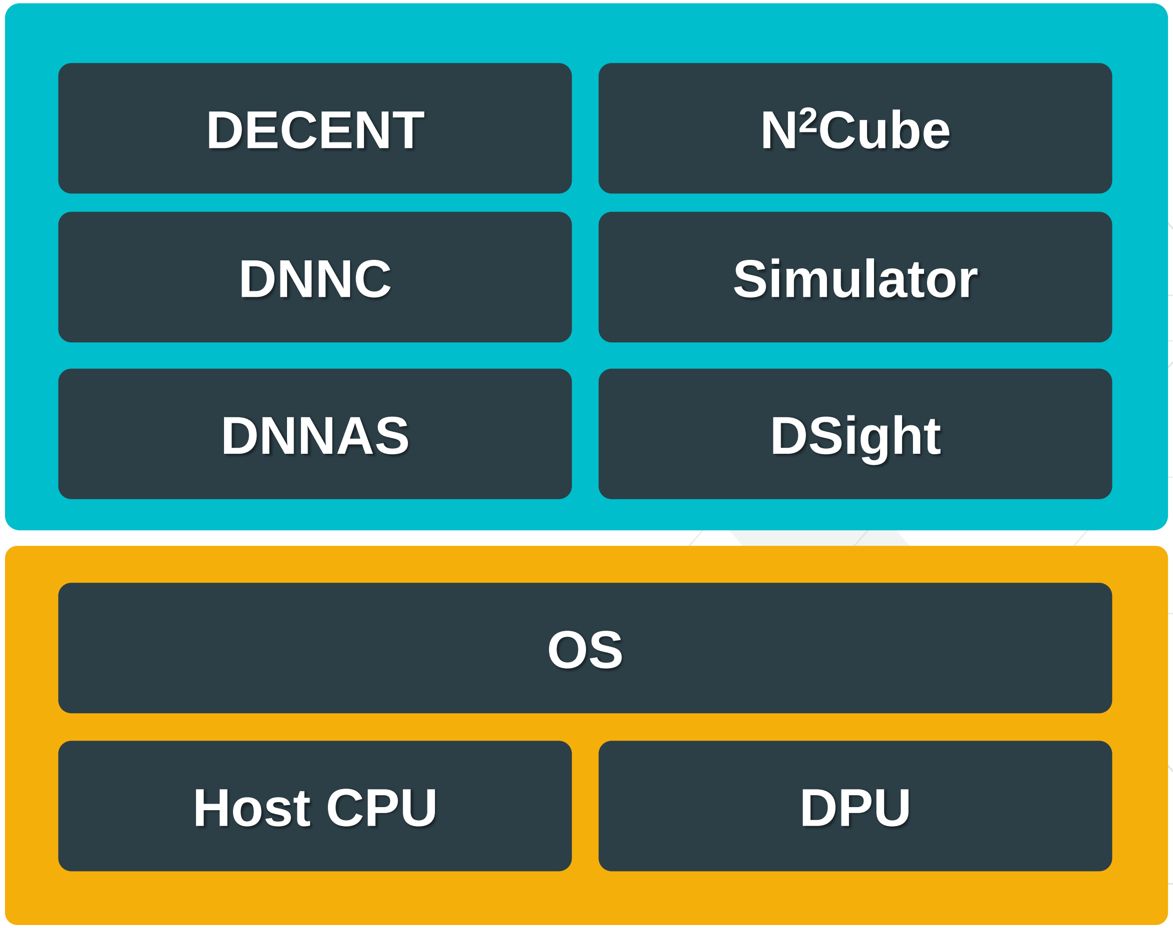


Z7020 Board | Z7020 SOM | ZU2 SOM | ZU2/3 Card | ZU9 Card | ZCU102 | ZCU104 | Ultra96

★ Deephi also has LSTM IP for KU115/VU9P as a part of Cloud ML

DNNDK Overview

- DECENT (DEepomprEssioN Tool)
- DNNC (Deep Neural Network Compiler)
- DNNAS (Deep Neural Network ASsembler)
- Runtime N^2Cube (Cube of Nerual Network)
- DPU Simulator – Internal tool
- Profiler DSight



Framework Support

Caffe

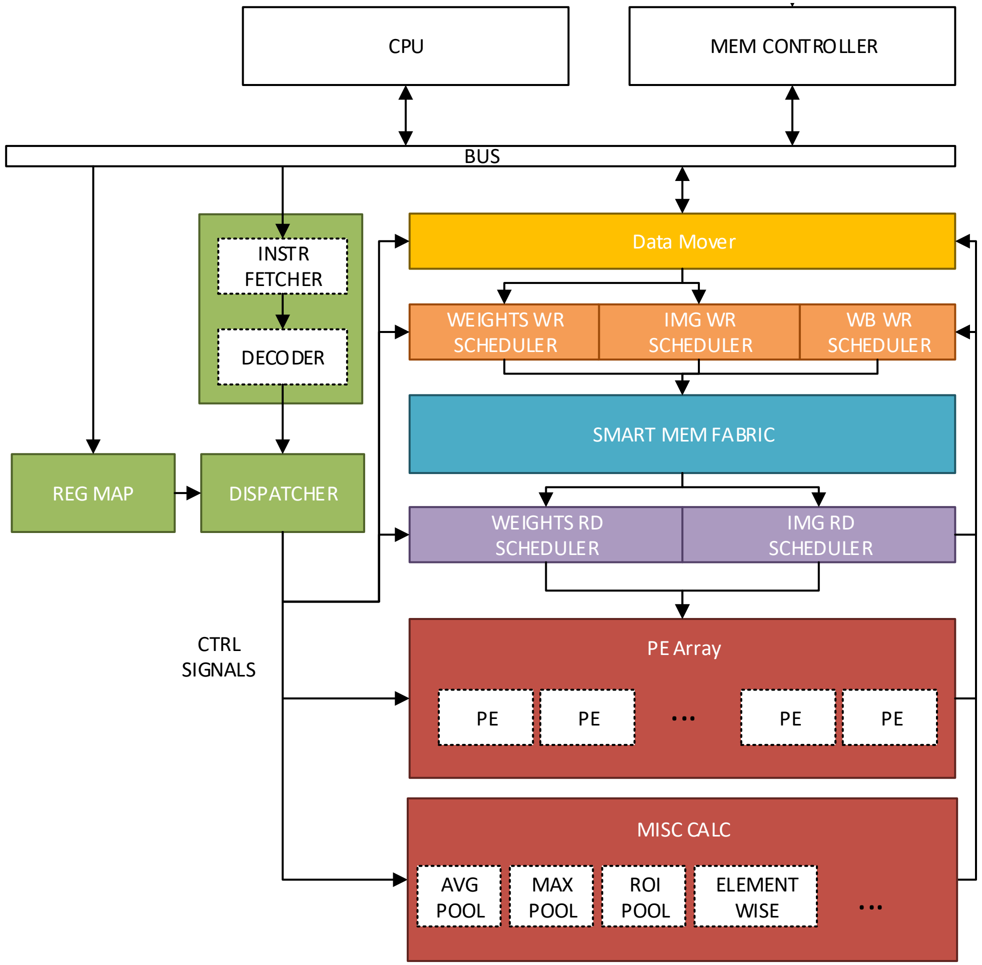
- Pruning
- Quantization
- Compilation

Darknet

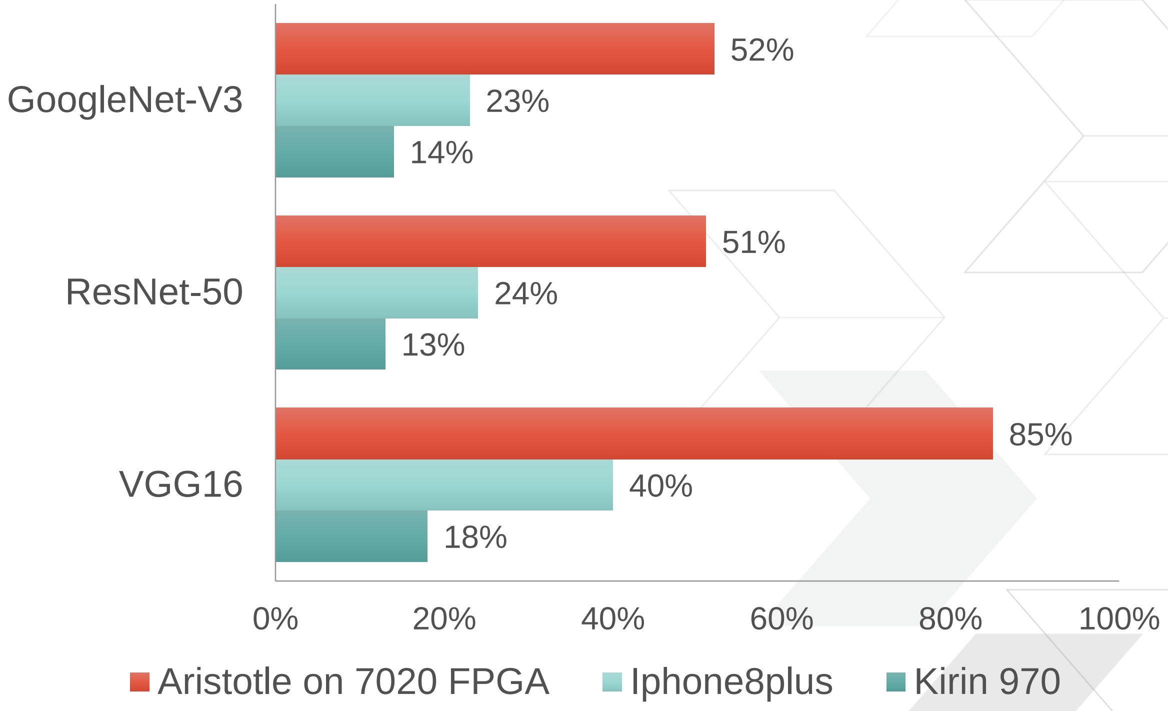
- Pruning
- Quantization
- Convertor for caffe

- 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)

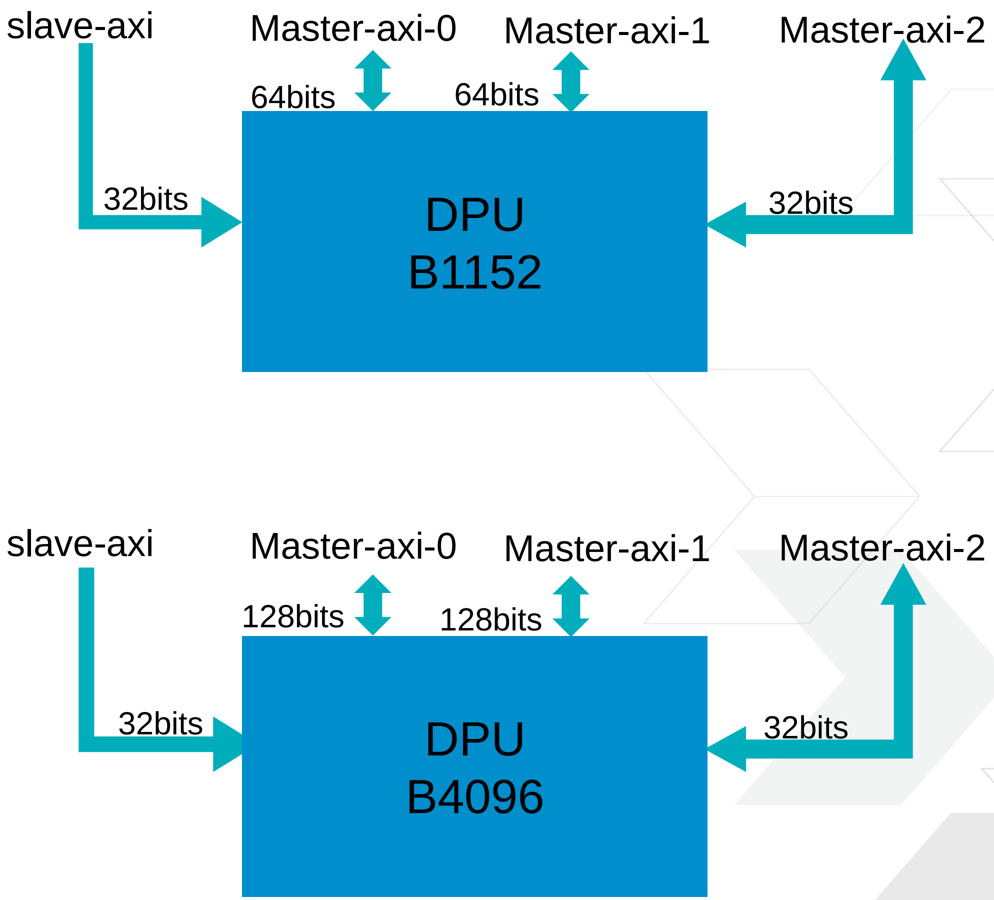
Constraints Between Layers

Layer Type \ Next Layer	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**

XDF XILINX DEVELOPER FORUM

XILINX

DPU Utilization

Single B1152 on Z7020

	LUT	Slice_reg	Block Ram	DSPs
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
Utilization ratio	86.18%	58.31%	74.67%	100.00%

Single B1152 on ZU3

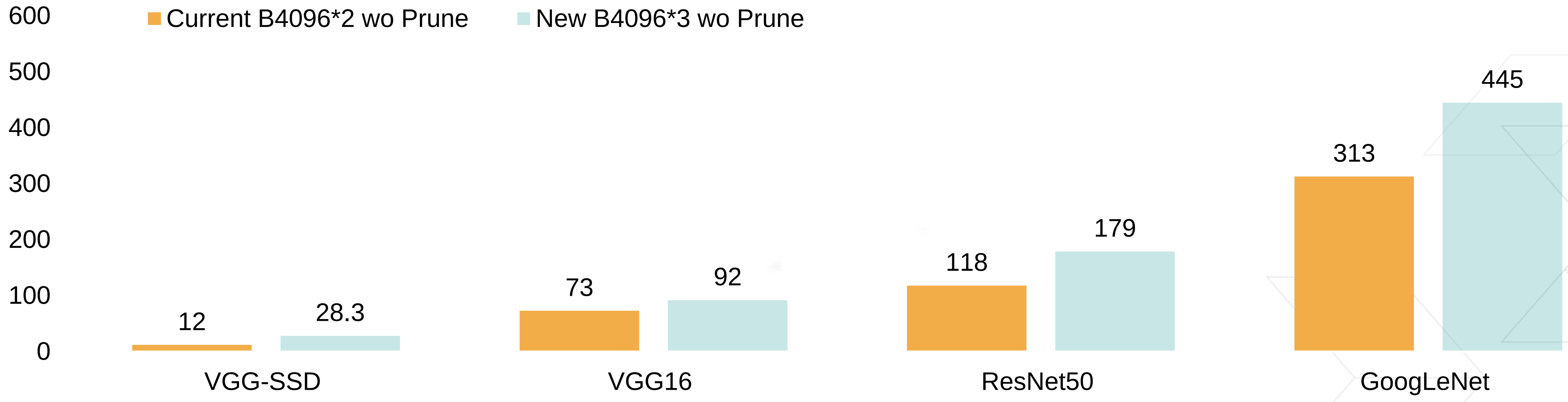
	LUT	Slice_reg	Block Ram	DSPs
All logic	70560	141120	216	360
DPU_B1152	36560	68729	115.5	288
Utilization ratio	51.81%	48.70%	53.47%	66.67%

Dual B4096 on ZU9

	LUT	Slice_reg	Block Ram	DSPs
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)

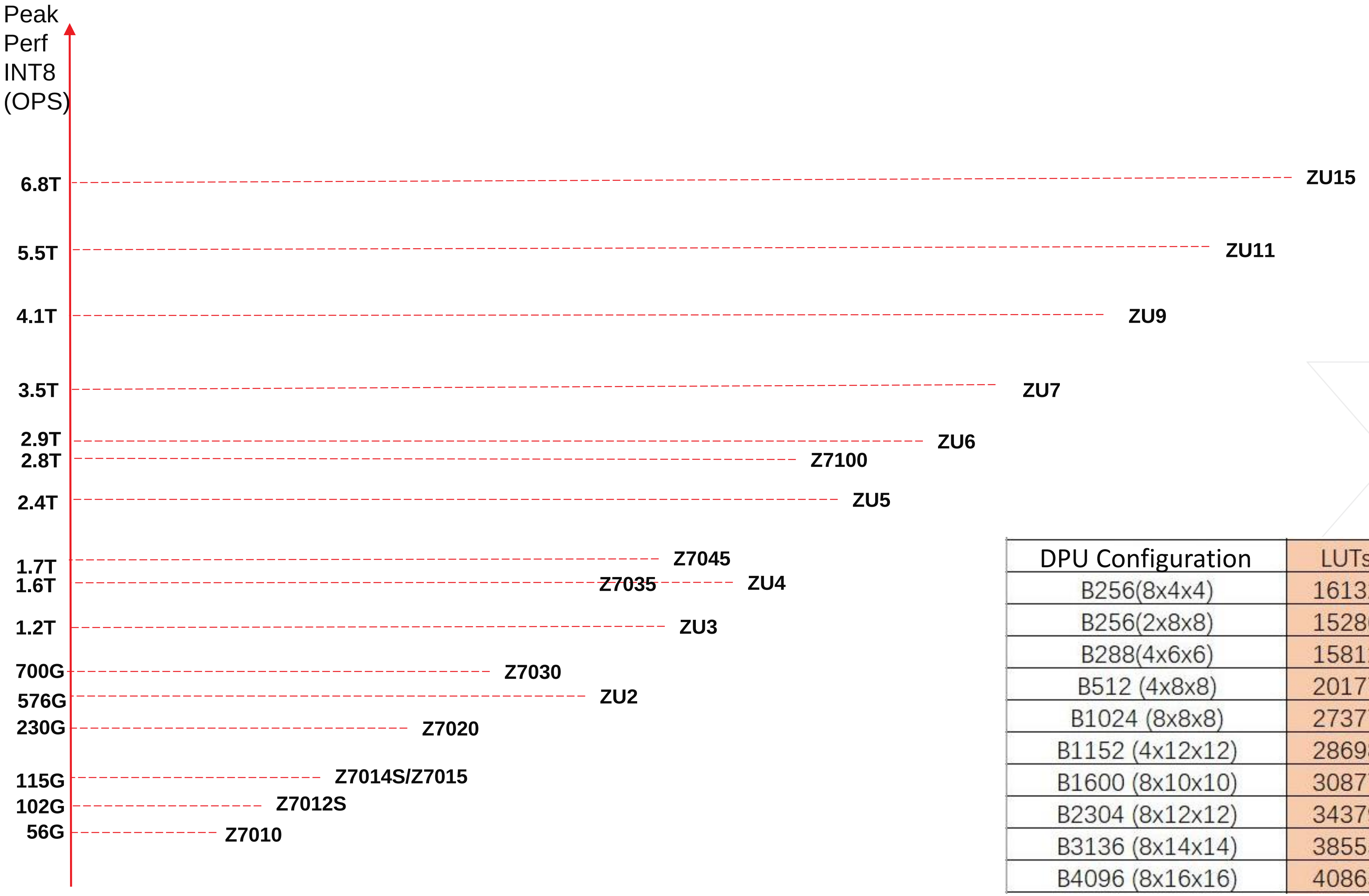


*The FPS of VGG-SSD of end to end performance
*The FPS of VGG16/ResNet50/GoogLeNet is of CONV part (w/o FC layer)

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

XDF XILINX DEVELOPER FORUM

XILINX

DNNDK Dev Flow

Five Steps with DNNDK

01 Model Compression

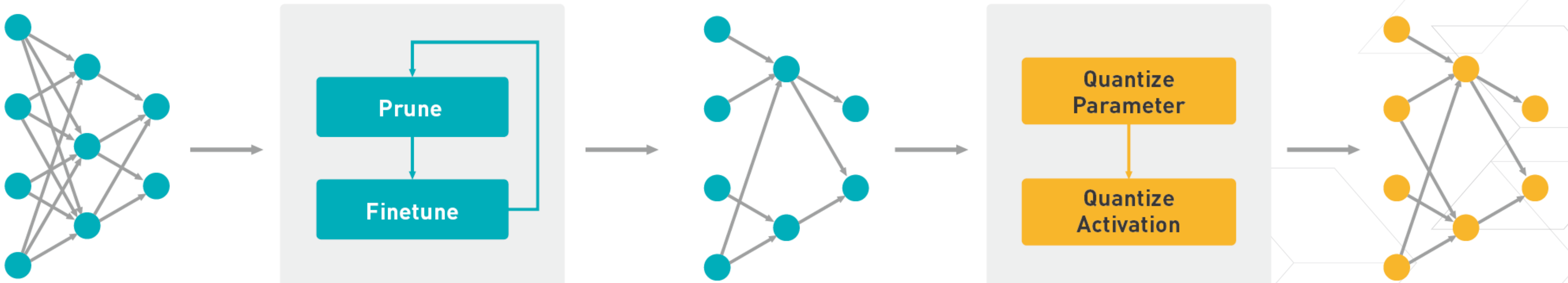
02 Model Compilation

03 Programming

04 Hybrid Compilation

05 Execution

DECENT – Deephi Deep Compression Tool



Dense Neural Network (FP32)

Pruning (Less number of param)

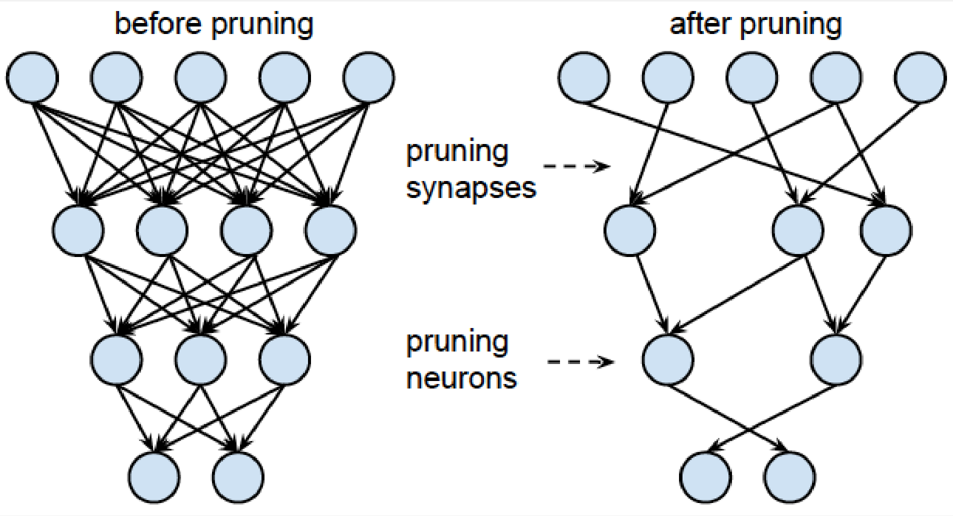
Pruned Neural Network (FP32)

Quantization (Less bits per param)

Compressed sparse Neural Network (INT8)

Deep Compression Overview

Deep compression
Makes algorithm smaller and lighter



Highlight

1/3	1/10	1/10	3 X
Weight number	Bandwidth load	Model size	Perfor-mance

Compression efficiency

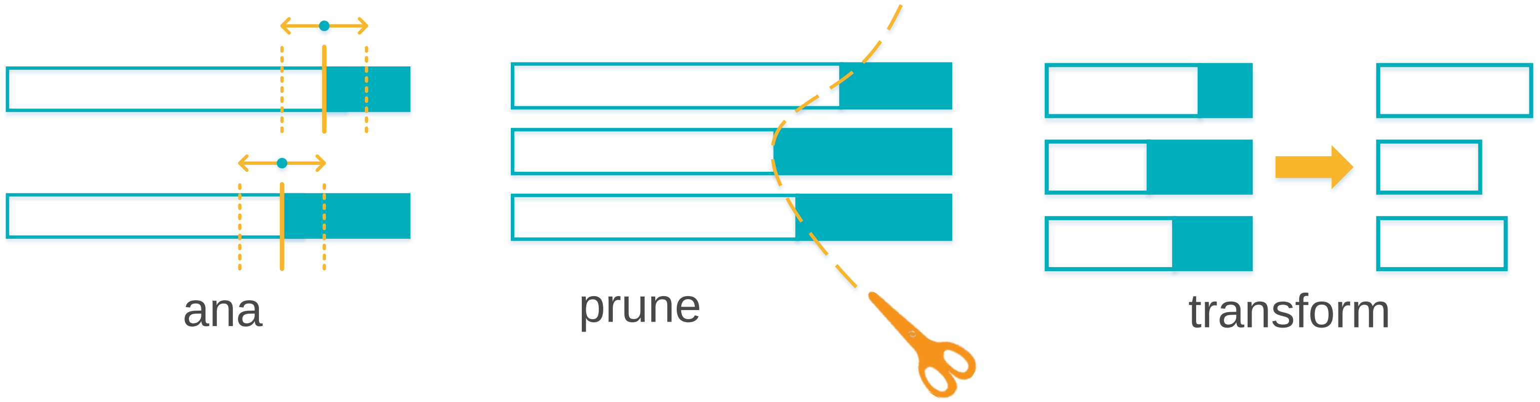
Deep Compression Tool can achieve significant compression on **CNN** and **RNN**

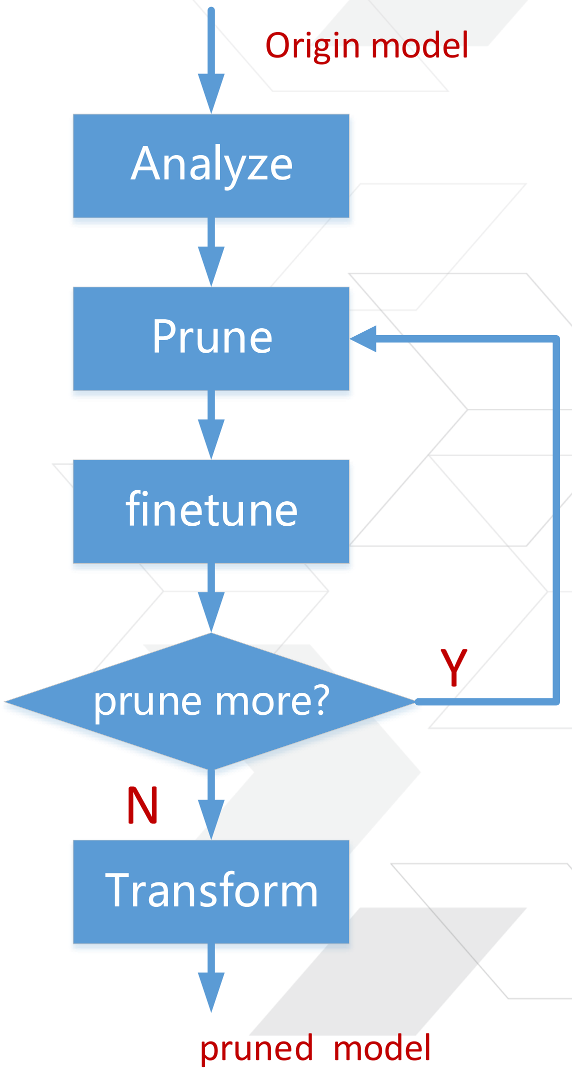
Accuracy

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



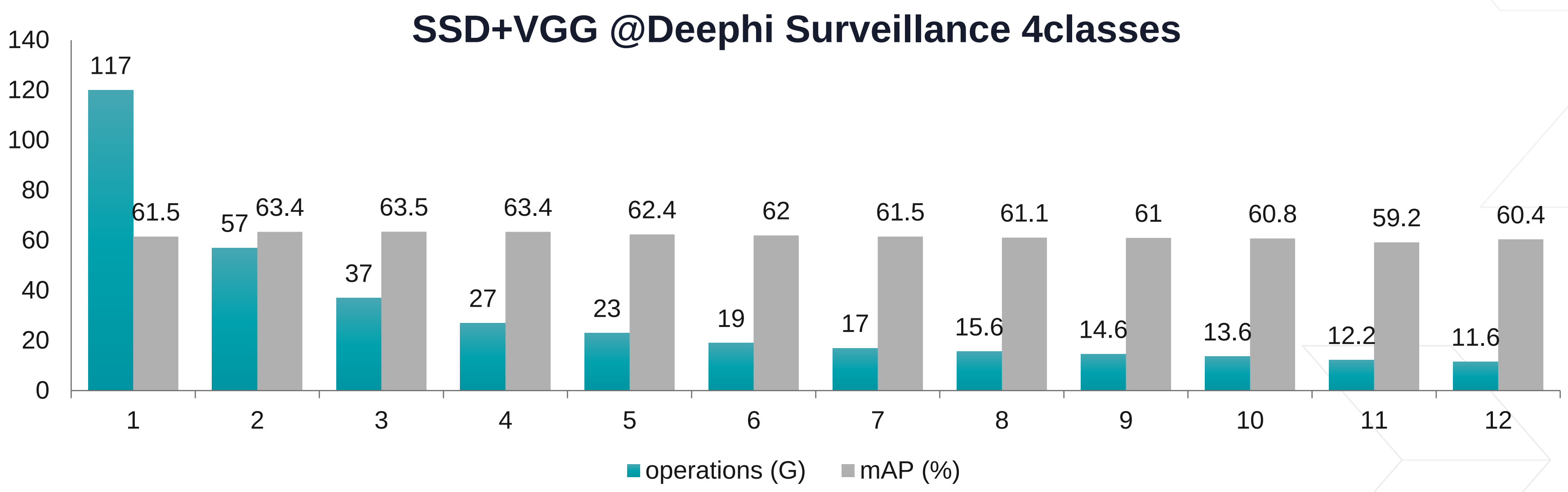


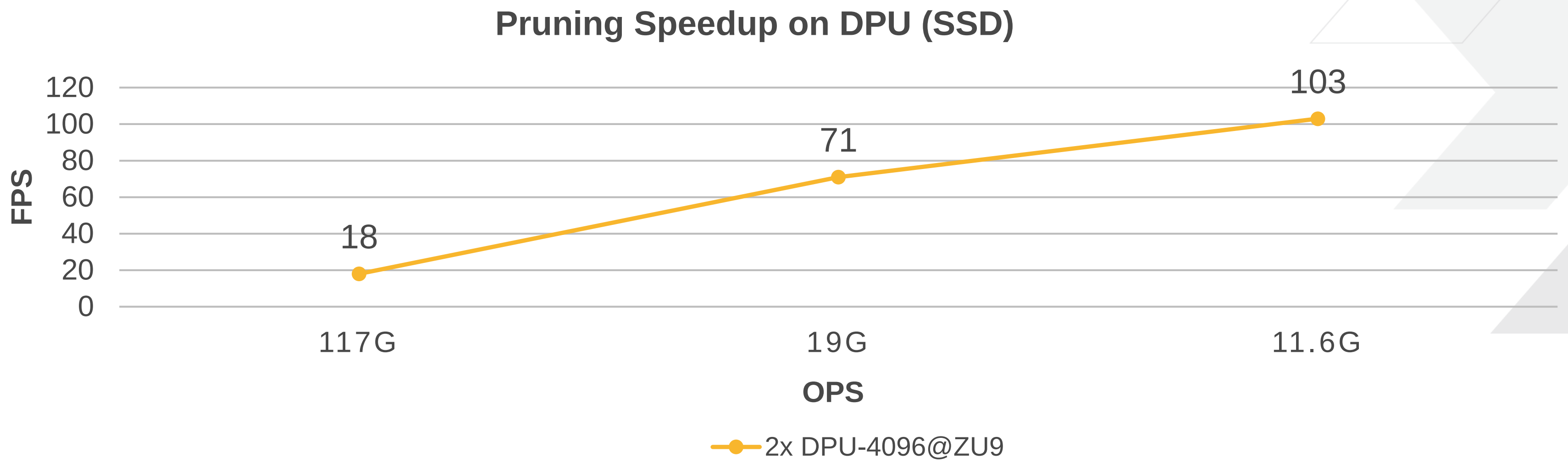
Pruning Results

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%

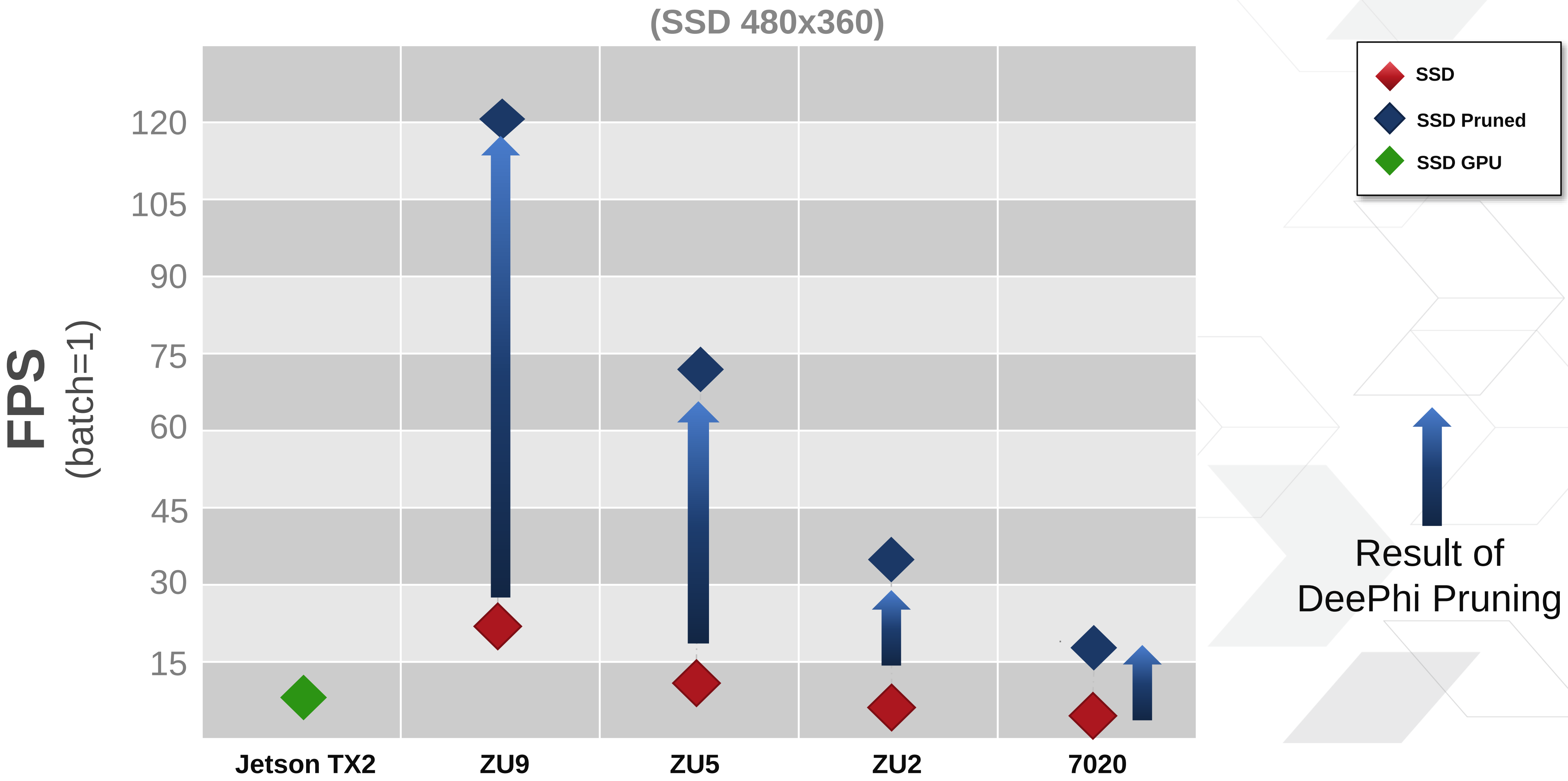
Pruning Example - SSD






Before Compression
18FPS@117GOPs
After 10X Compression
103FPS@11.6GOPs
DEEPHi
深 鉴 科 技

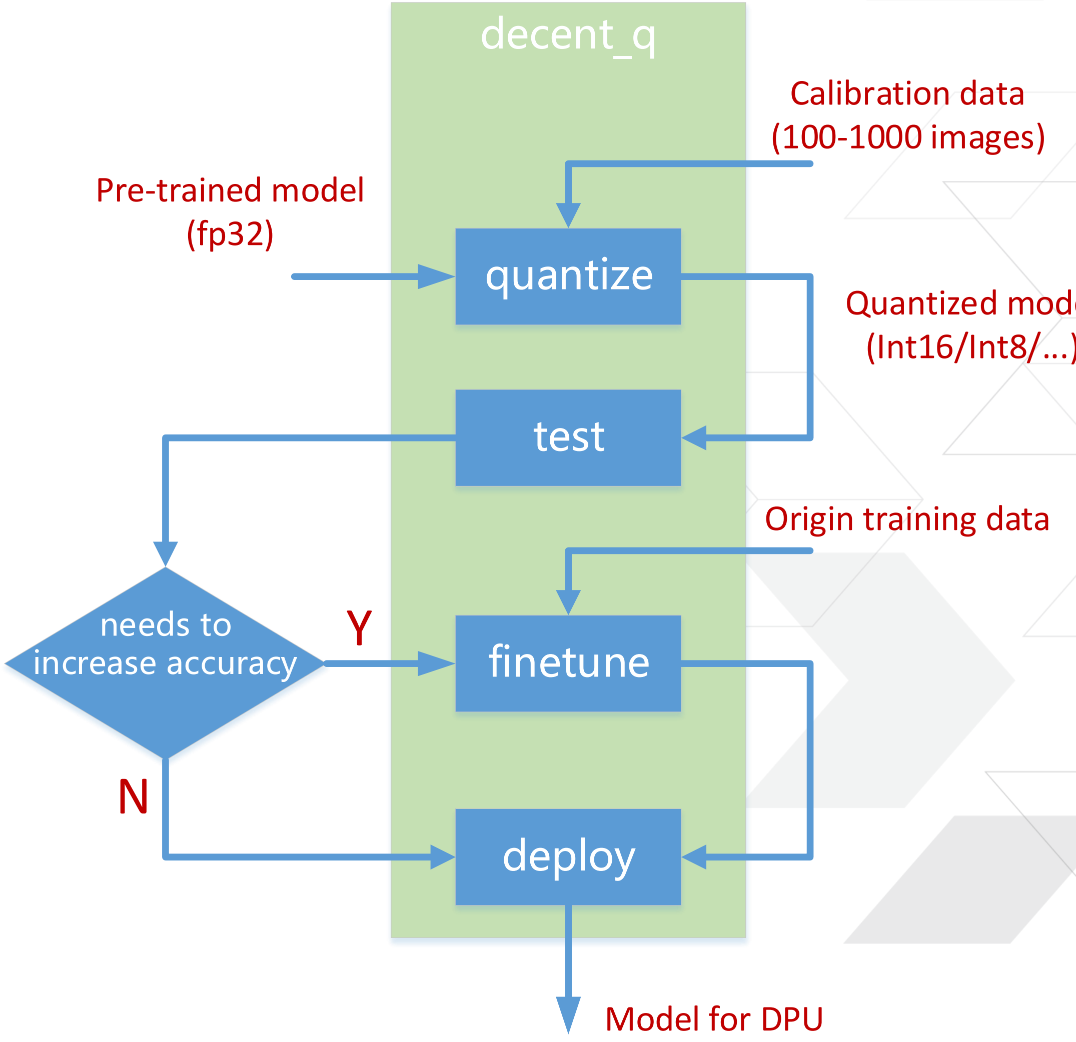
Makes Big Difference with Pruning



Power	10W	10W	5W	3W	2W

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()
dpuSetIntputTensorInCHWInt8()
dpuSetIntputTensorInCHWFP32()
dpuSetIntputTensorInHWCInt8()
dpuSetIntputTensorInHWCFP32()
dpuGetOutputTensorInCHWInt8()
dpuGetOutputTensorInCHWFP32()
dpuGetOutputTensorInHWCInt8()
dpuGetOutputTensorInHWCFP32()

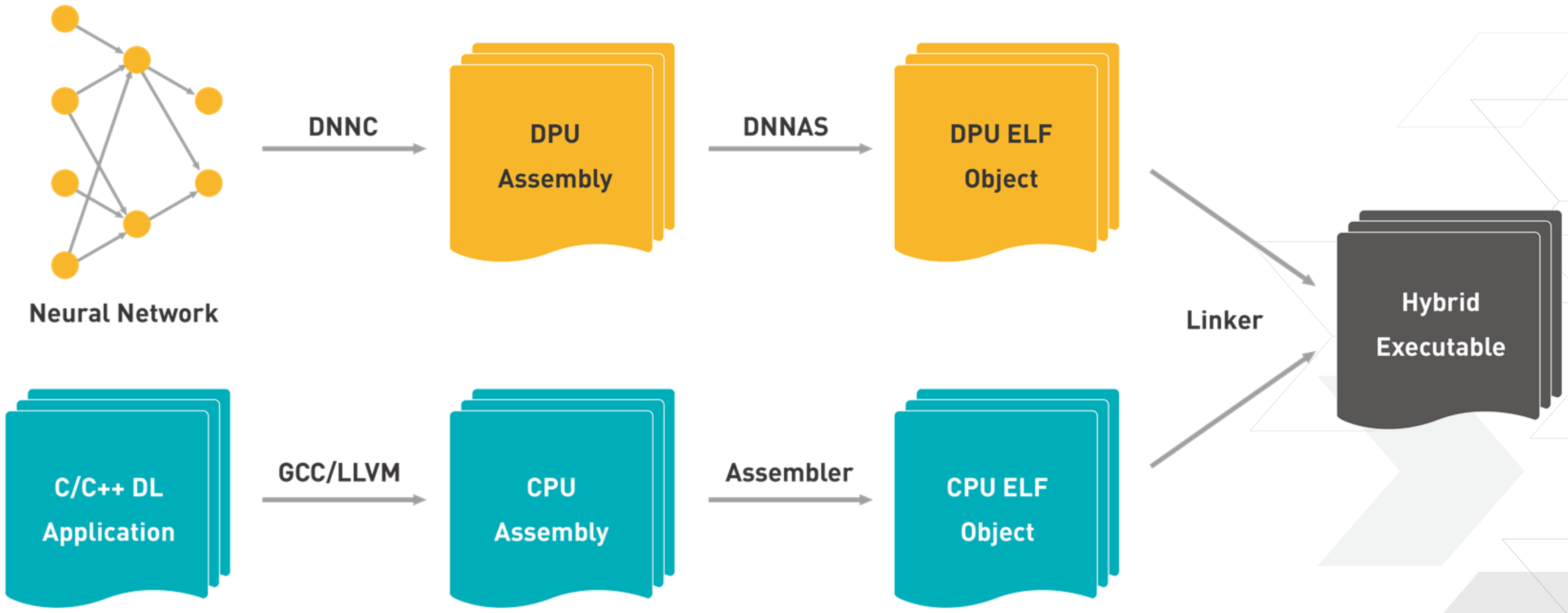
> **For more details, refer to DNNDK User Guide**

http://www.deephi.com/technology/dnndk

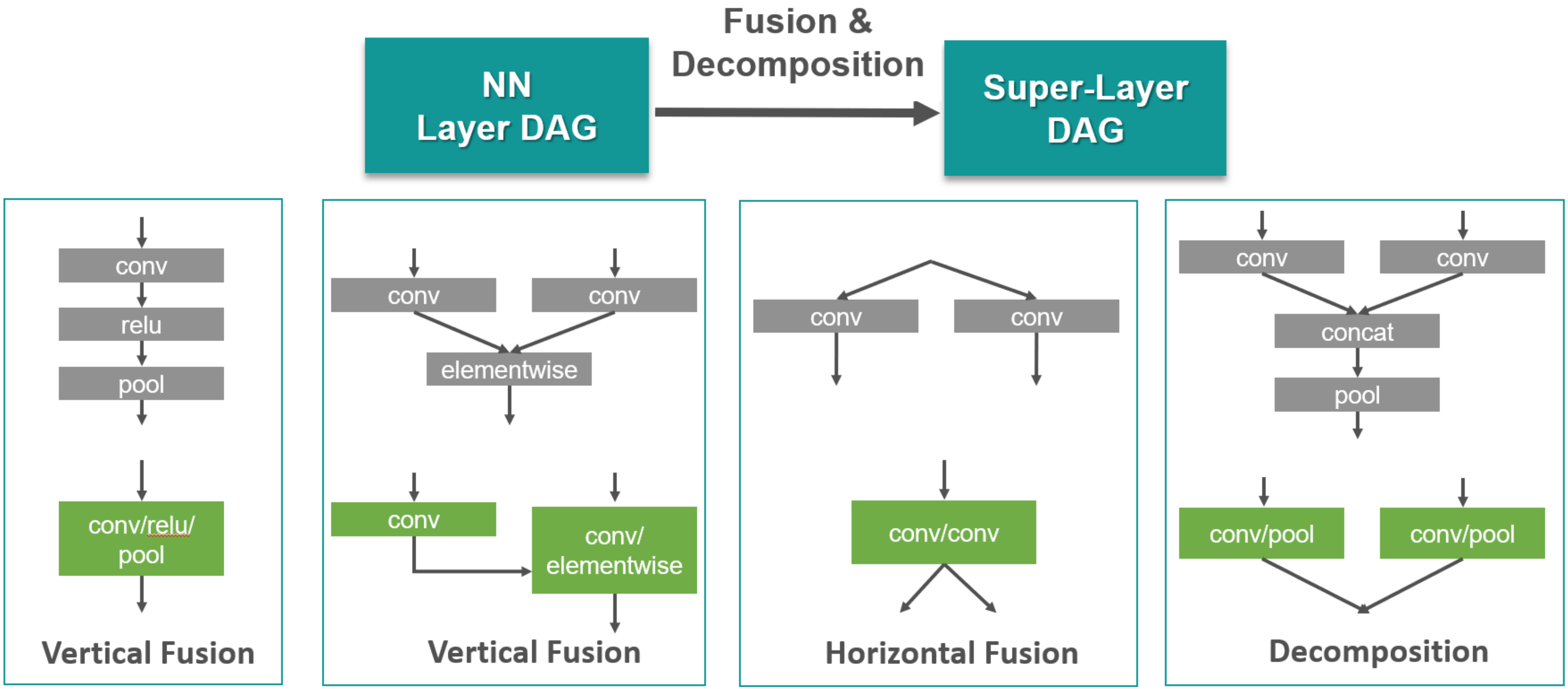
Programming with DNNDK API

```
int main(int argc, char *argv[])
{
    DPUKernel *kernel_conv;
    DPUKernel *kernel_fc;
    DPUTask   *task_conv;
    DPUTask   *task_fc;
    char      *input_addr;
    char      *output_addr;

    /* DNNDK API to attach to DPU driver */
    dpuInit();

    /* DNNDK API to create DPU kernels for CONV & FC networks */
    kernel_conv = dpuLoadKernel("resnet50_conv", 224, 224);
    kernel_fc = dpuLoadKernel("resnet50_fc", 1, 1);

    /* Create tasks from CONV & FC kernels */
    task_conv = dpuCreateTask(kernel_conv);
    task_fc = dpuCreateTask(kernel_fc);

    /* Set input tensor for CONV task and run */
    input_addr = dpuGetTensorAddress(dpuGetTaskInputTensor(task_conv));
    setInputImage(Mat &image, input_addr);
    dpuRunTask(task_conv);
    output_addr = dpuGetTensorAddress(dpuGetTaskOutputTensor(task_conv));

    /* Run average pooling layer on CPU */
    run_average_pooling(output_addr);

    /* Set input tensor for FC task and run */
    input_addr = dpuGetTensorAddress(dpuGetTaskInputTensor(task_fc));
    setFCInputData(task_fc, input_addr);
    dpuRunTask(task_fc);
    output_addr = dpuGetTensorAddress(dpuGetTaskOutputTensor(task_fc));

    /* Diaplay the Classification result from FC task */
    displayClassificationResult(output_addr);

    /* DNNDK API to destroy DPU tasks/kernels */
    dpuDestroyTask(task_conv);
    dpuDestroyTask(task_fc);

    dpuDestroyKernel(kernel_conv);
    dpuDestroyKernel(kernel_fc);

    /* DNNDK API to dettach from DPU driver and free DPU resources */
    dpuFini();

    return 0;
}
```

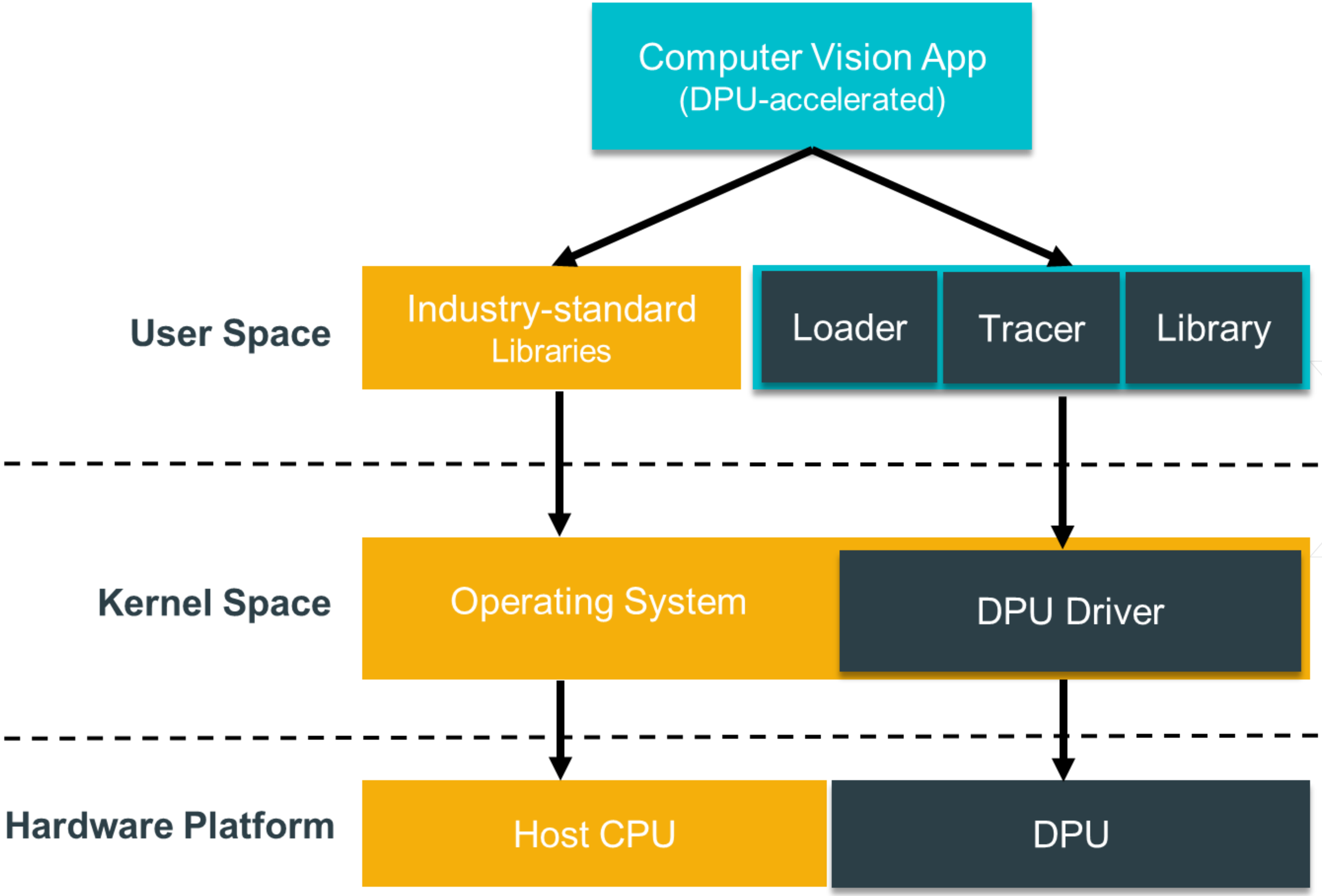
DNNDK Hybrid Compilation Model



Optimization in DNNC



DNNDK Runtime Engine



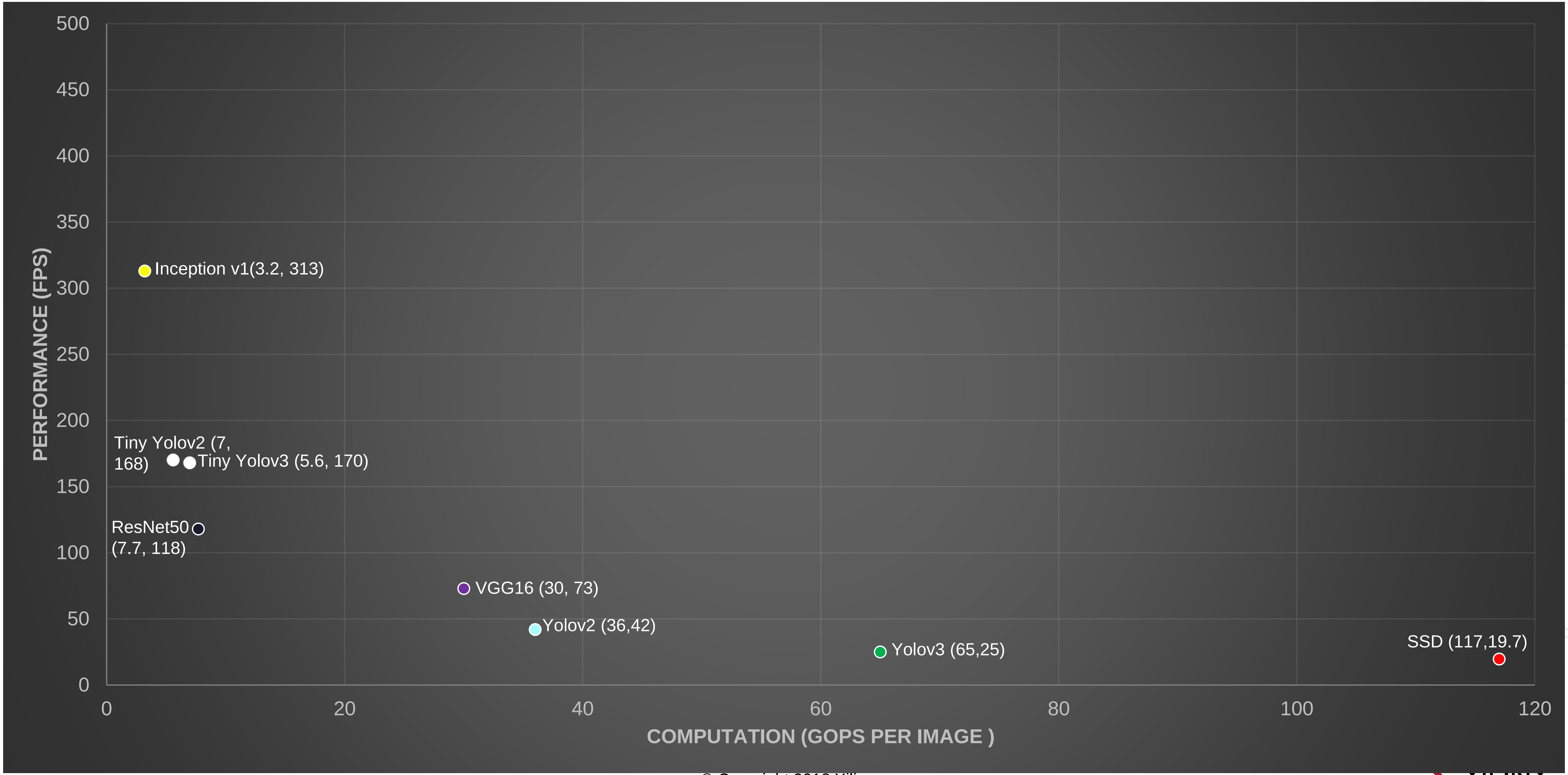
Runtime N^2Cube

- Library
- Loader
- Tracer
- Driver

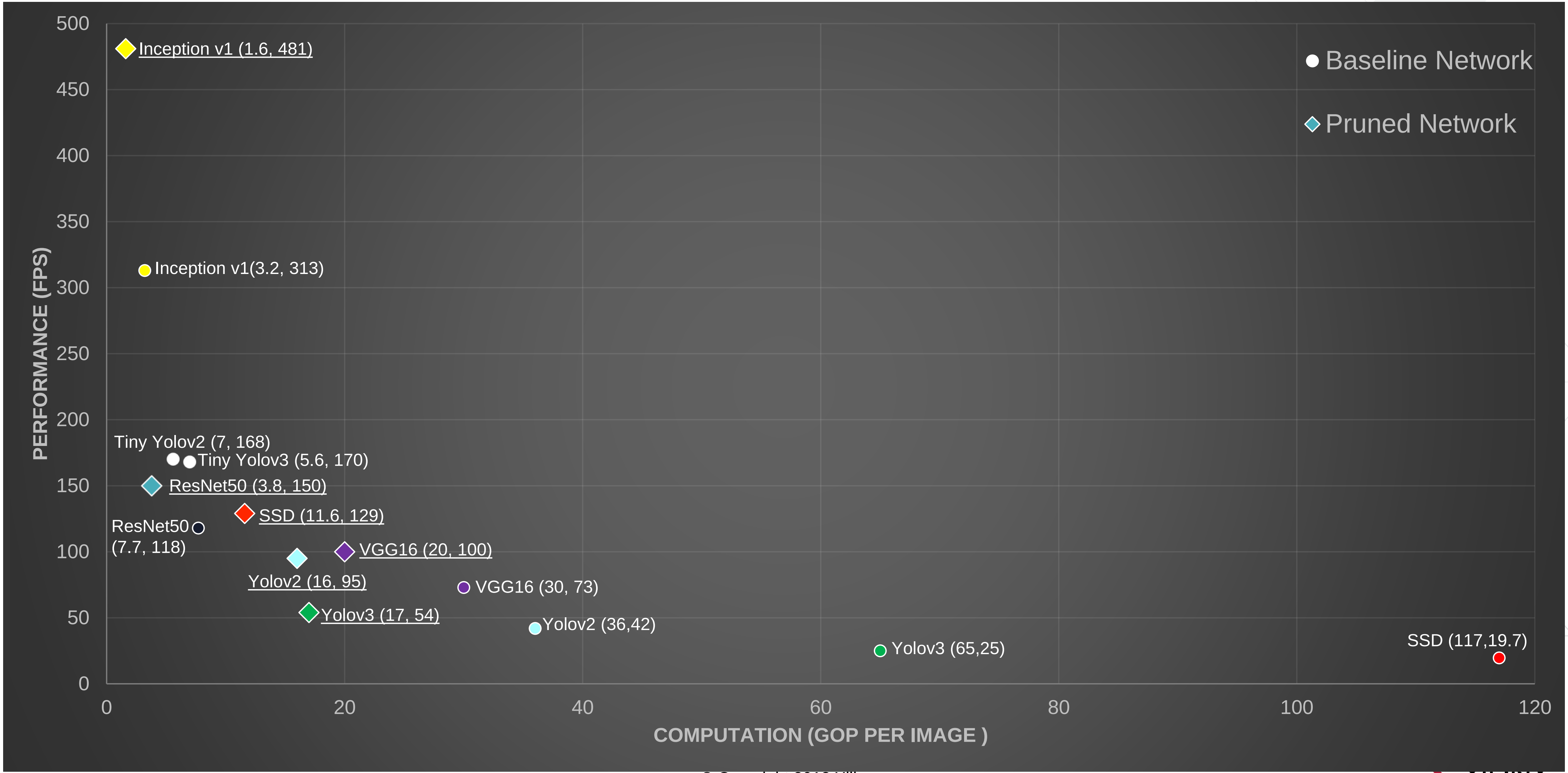
Supported Networks

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	✔	✔	✔
	Drivable Space Detection	MobilenetV2-FPN	✔		
	Multi-task (Detection+Segmentation)	Deephi	✔		

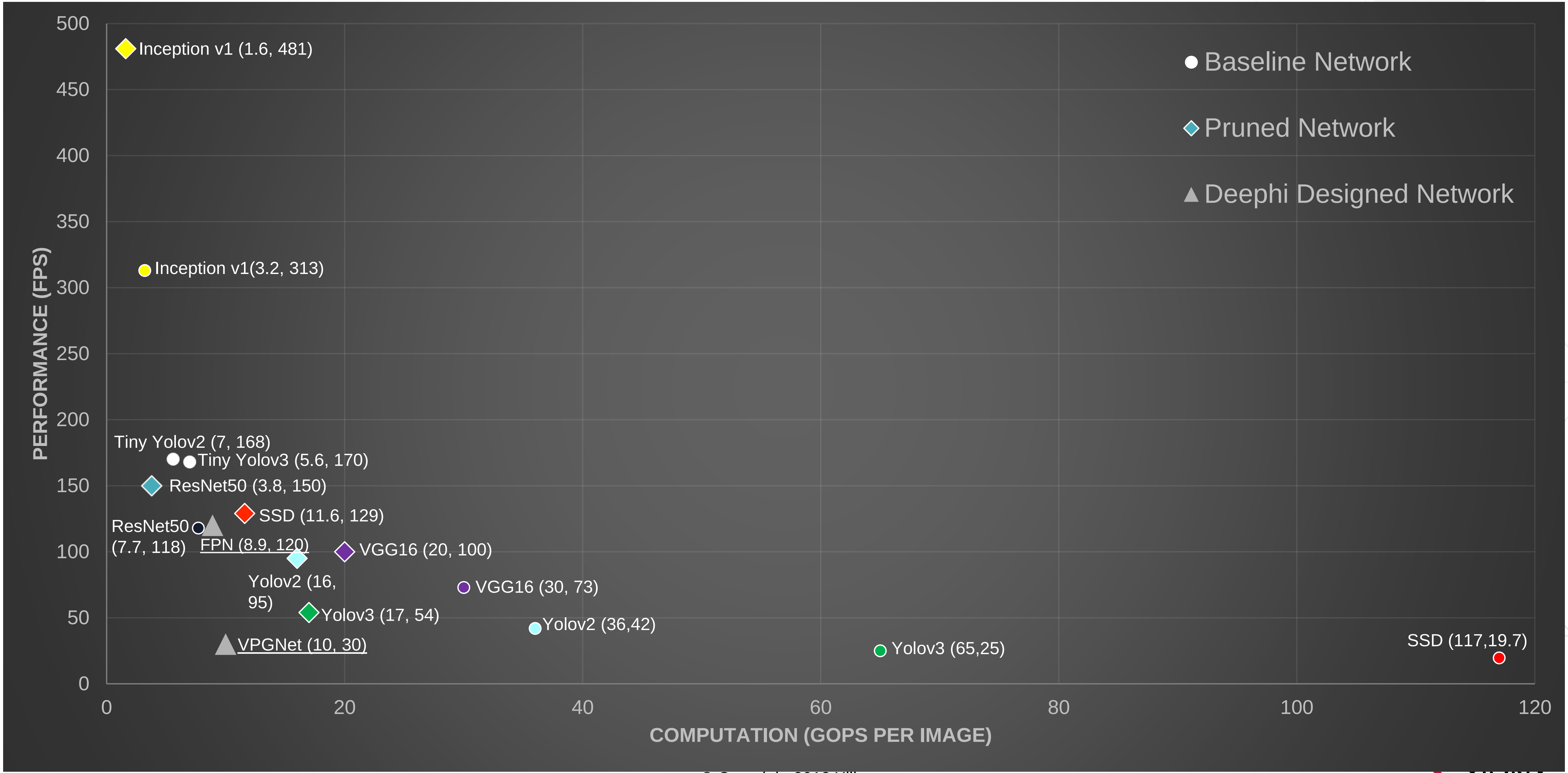
Measured Performance



Measured Performance (Cont.)

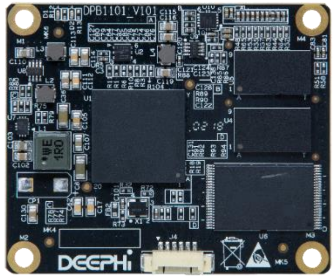


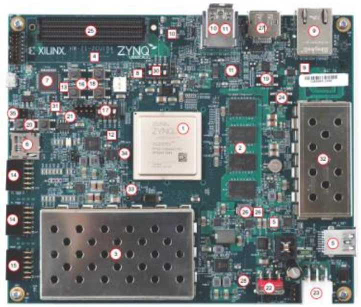
Measured Performance (Cont.)



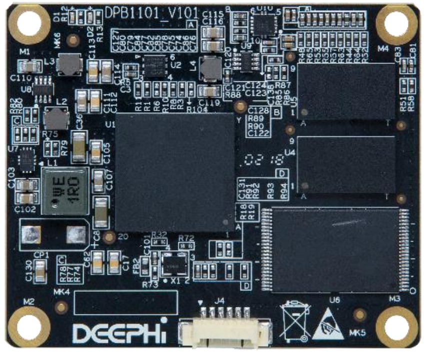
Out-of-box Supported Boards

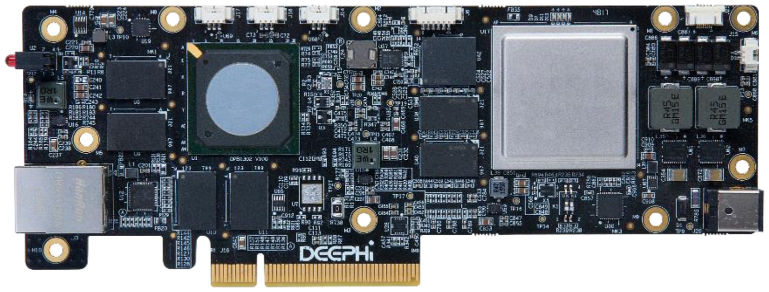
- DP8000
  - Z7020 SOM
- DP2400
  - ZU9 PCIe card
- Deephi ZU2/3 board
- Xilinx ZCU102
- Xilinx ZCU104
- Avnet Ultra96





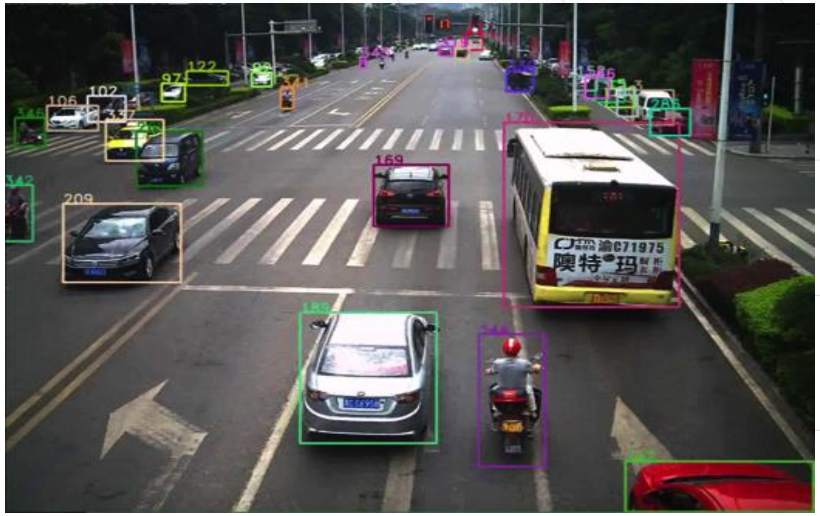
Video Surveillance ML Solutions





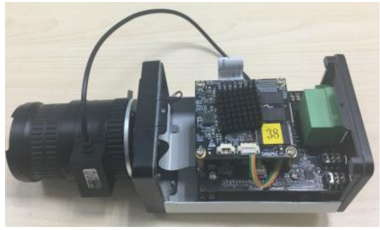
Intelligent
IP Camera Solution

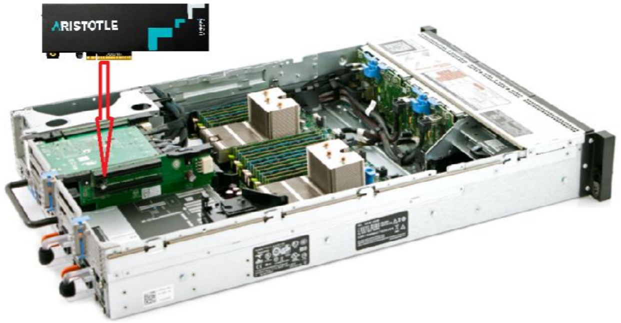
Video Analytics
Acceleration Solution



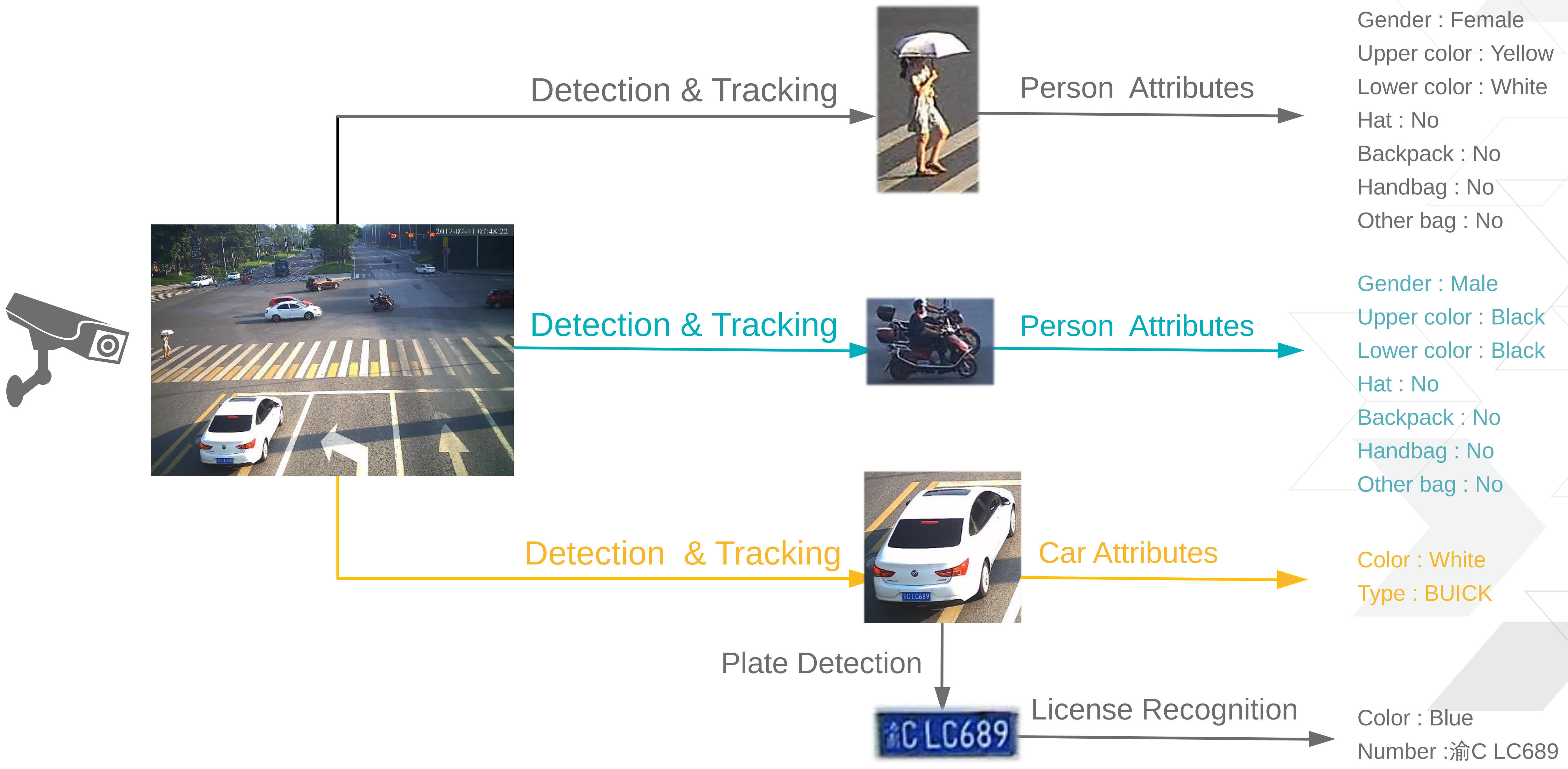
Face recognition camera
with Zynq7020

8-channel 1080P Video Analytics
with ZU9EG





Video Surveillance ML Ref Design



ADAS/AD ML Reference Design

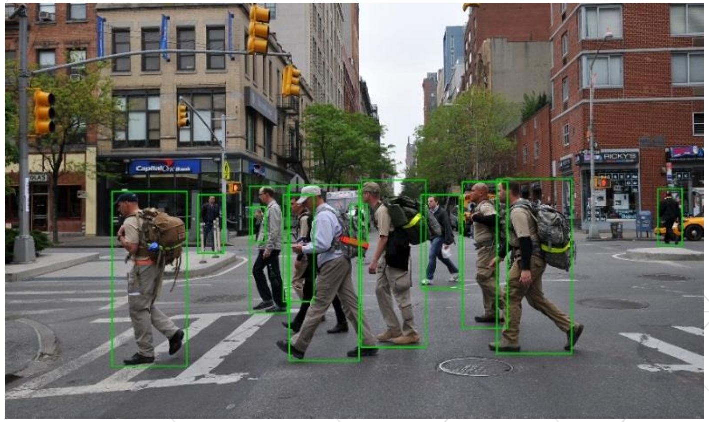
2D/3D Object Detection

Segmentation + Detection

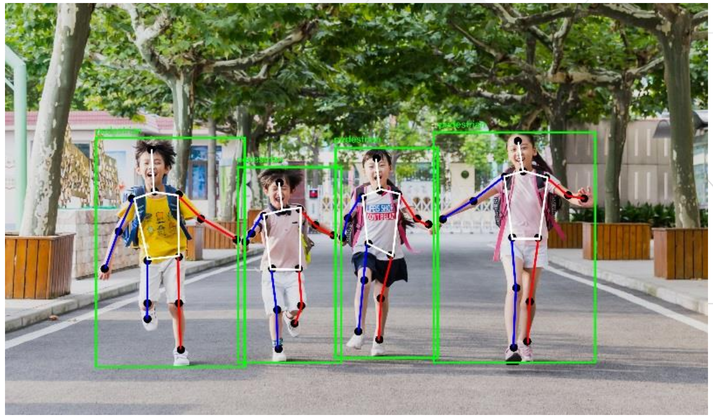
Lane Detection

Segmentation

Pedestrian Detection



Pose Estimation

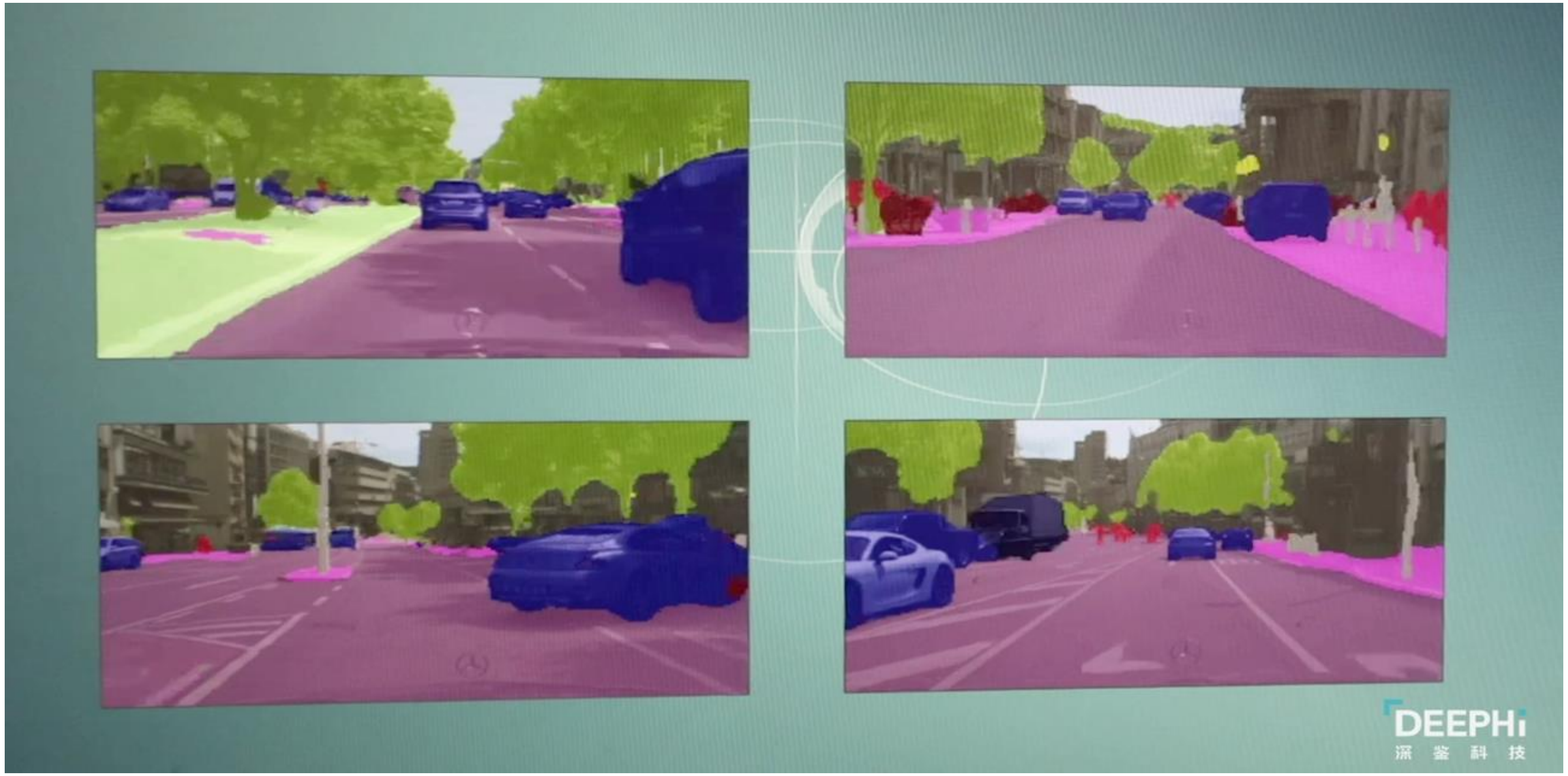


8-ch Detection Demo

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

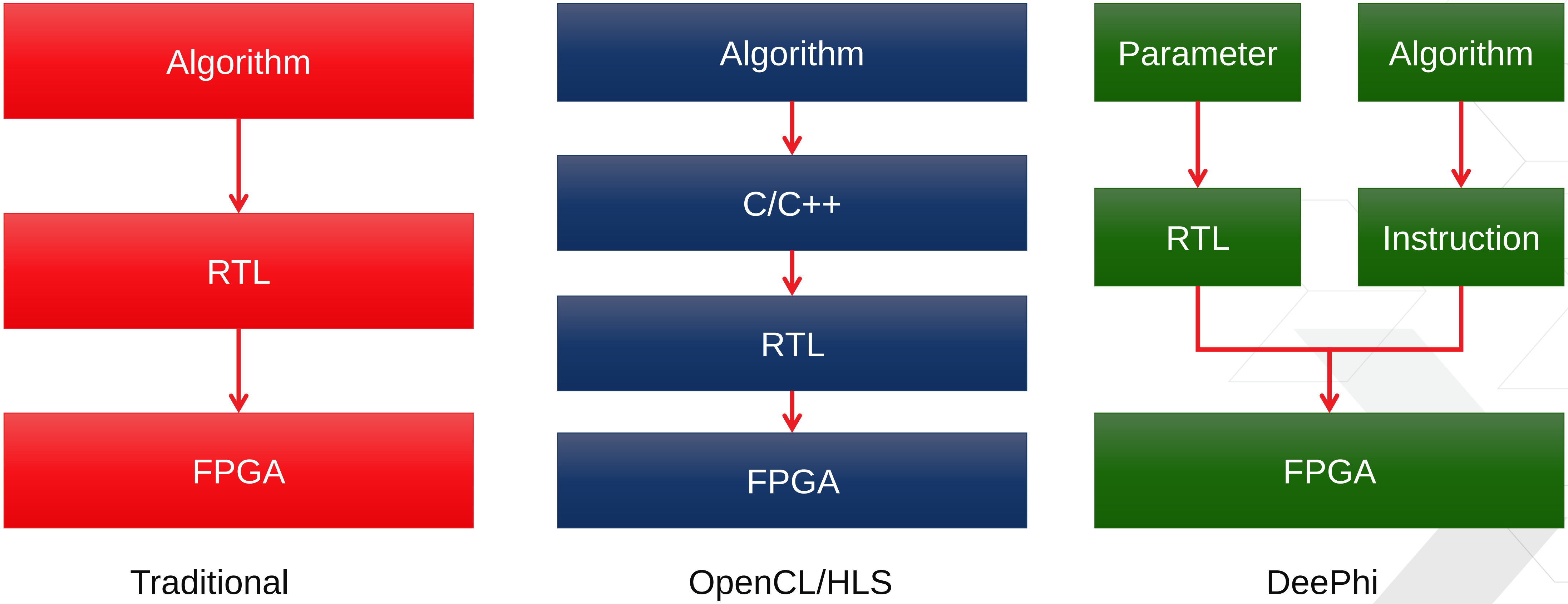
4-ch Segmentation + Detection Demo

- 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



Two Development Flows of Using Deephi DPU IP

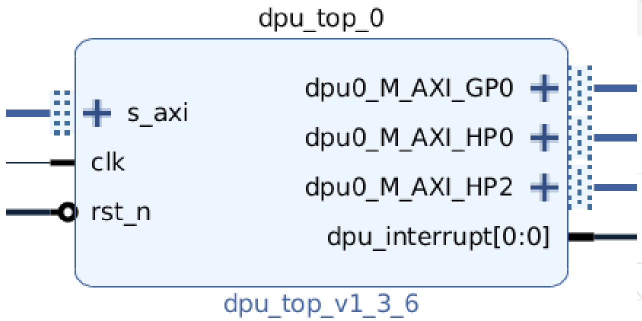
- **Vivado & SDK**
  - Traditional flow
  - Bottom up approach
  - Suitable for FPGA designer
  - Fine-grained customization

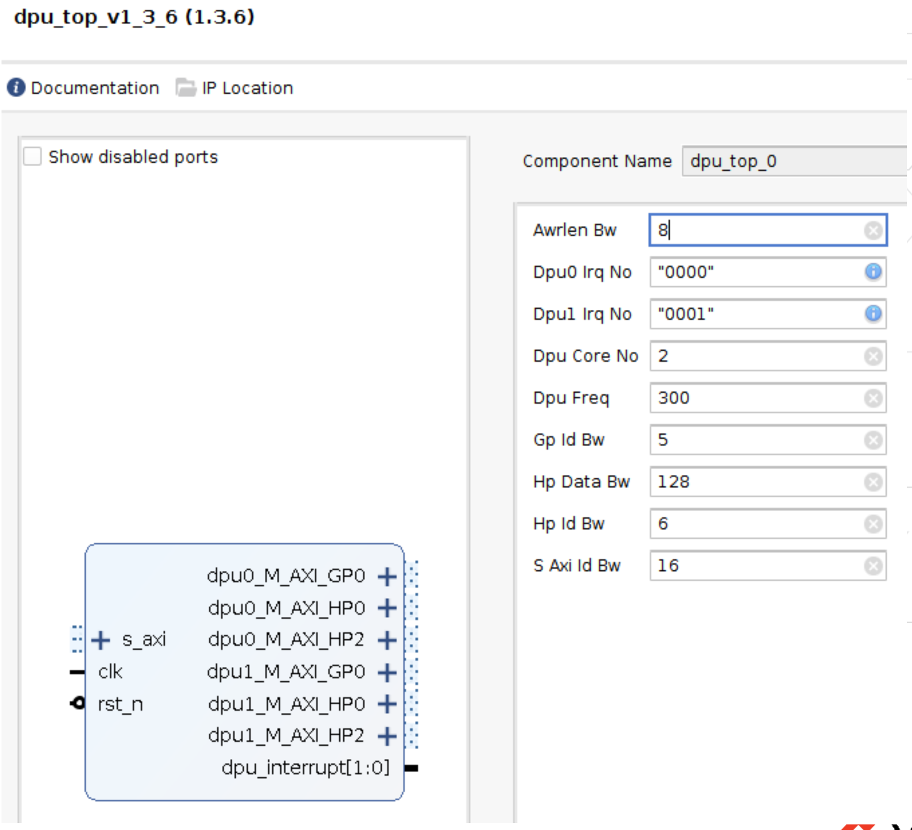
- **SDSoC**
  - New high-level abstraction flow
  - Top down approach
  - Suitable for algorithm & software developer
  - Higher Productivity

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





HW Integration with Vivado IPI (Cont.)

> Steps(Cont.)

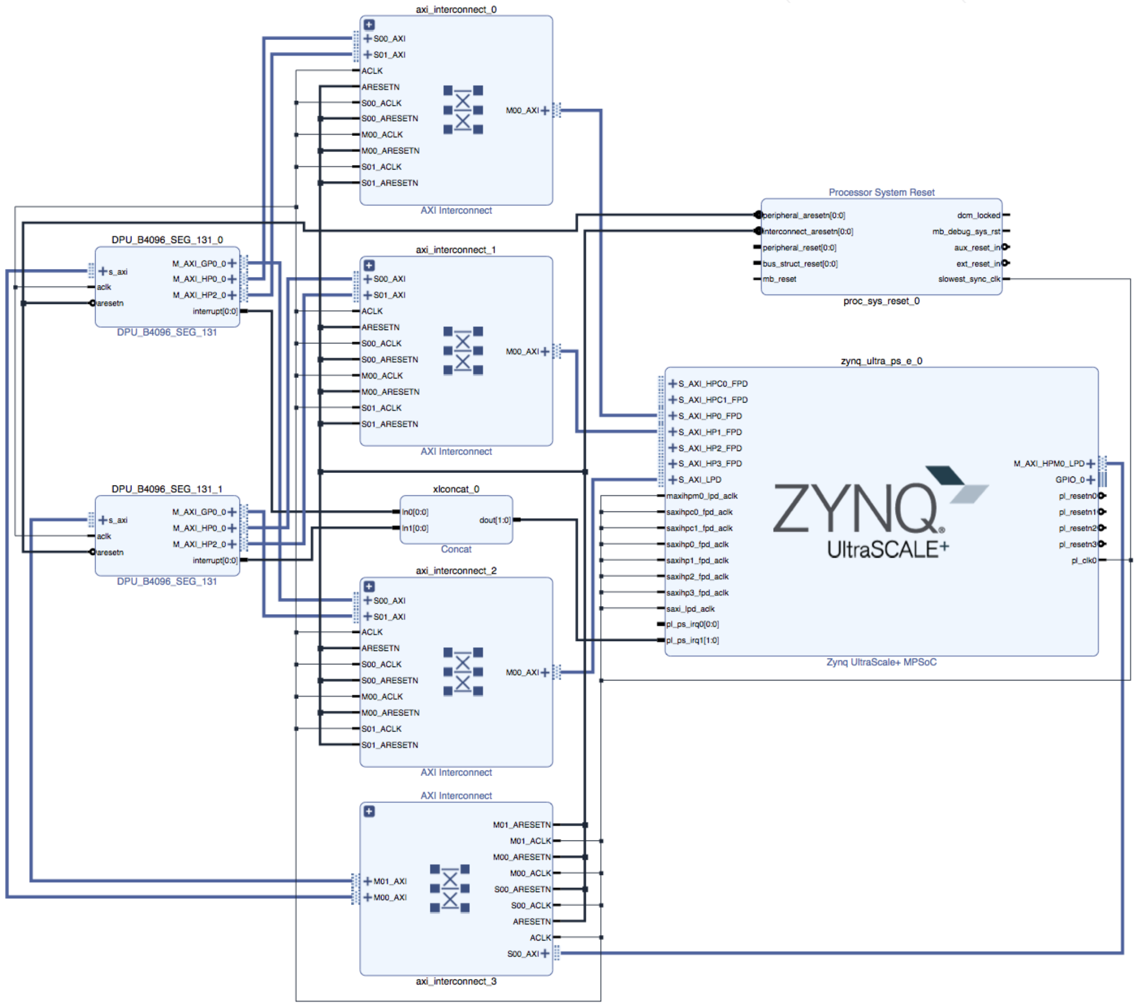
- >> Create top wrapper
- >> Generate bitstream
- >> Generate BOOT.BIN using Petalinux etc.

> 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

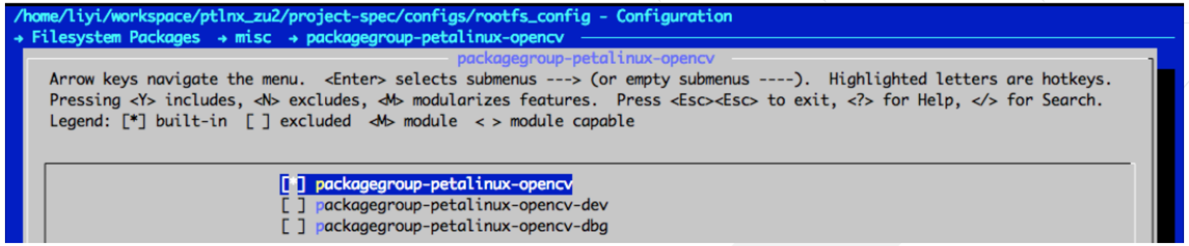
> **Driver and DNNDK lib**

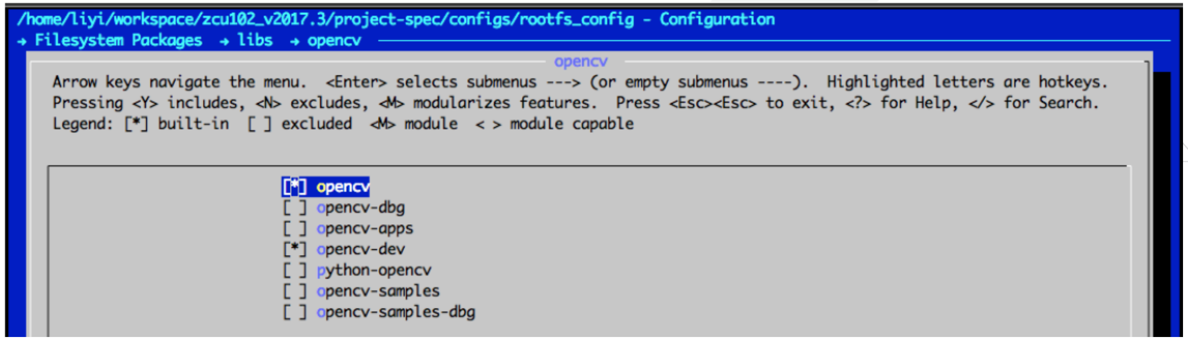
>> Provide kernel information & OpenCV version to Deephi

>> Deephi will provide driver and DNNDK package with install script

>> Install driver and DNNDK lib

```
amba {
    ...
    dpu@80000000 {

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

}
```

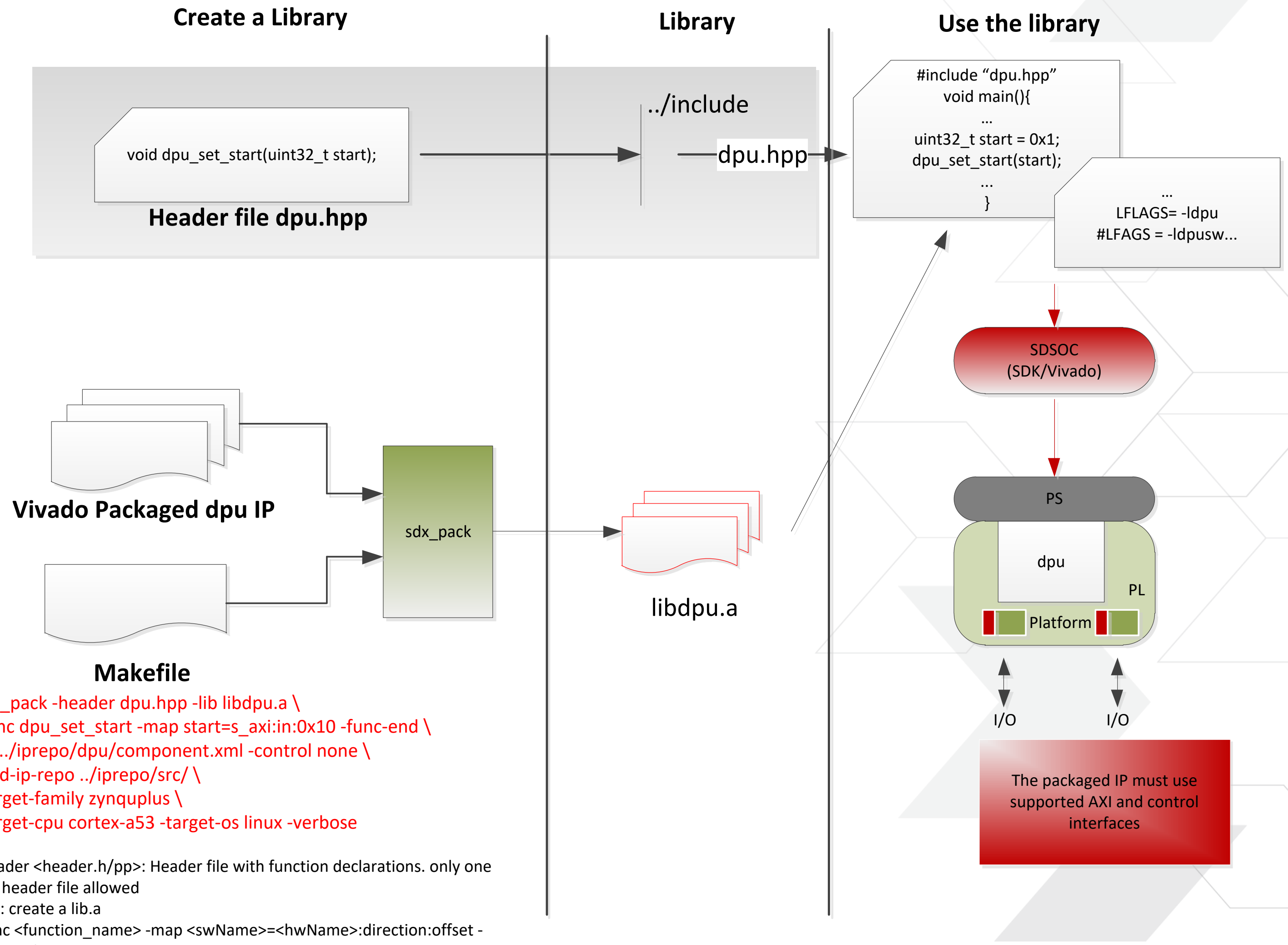




HW Integration with C-callable IP

> **Steps**

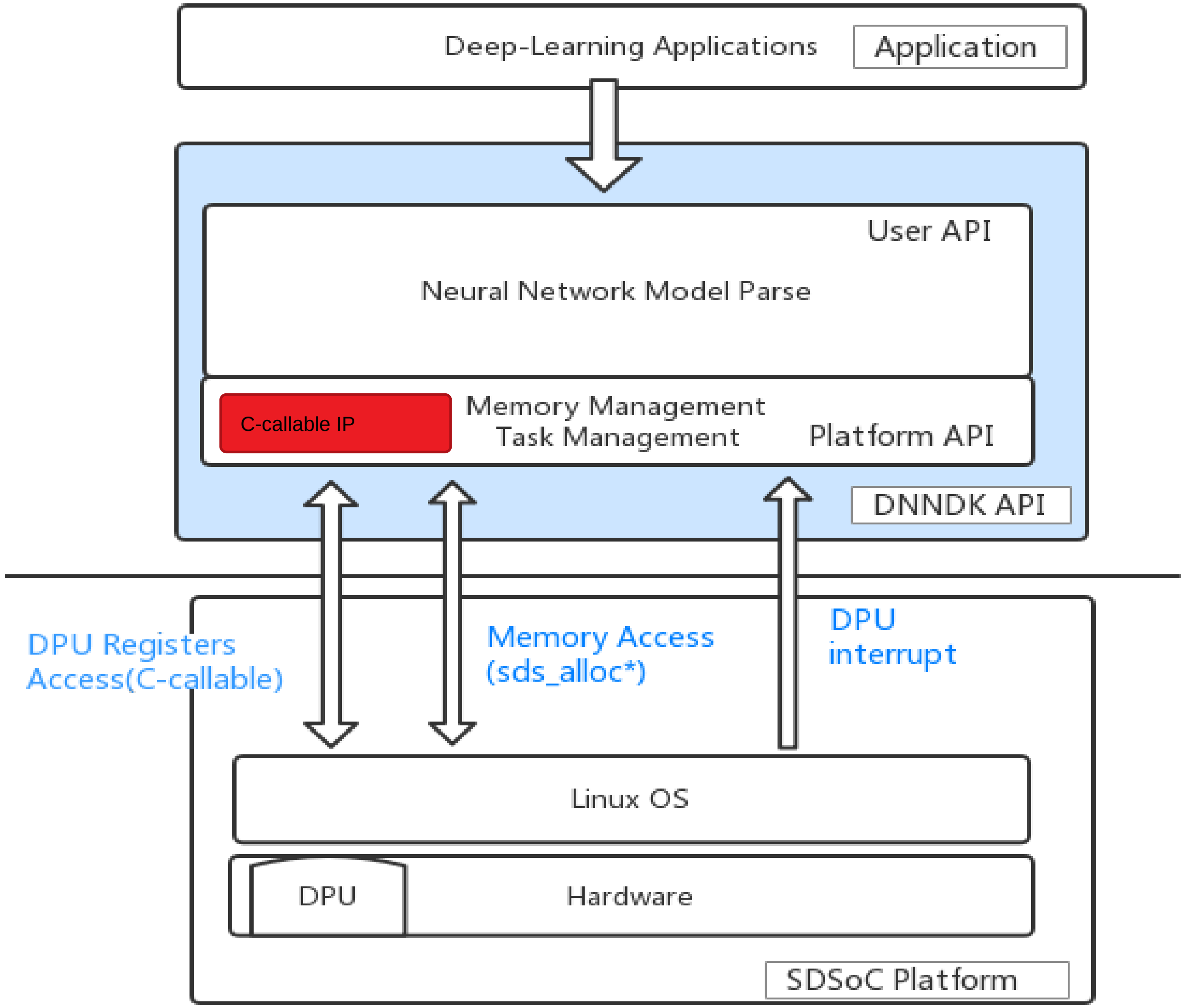
- >> Create header file
- >> Package IP in Vivado
- >> Create Makefile to generate *.a
- >> Configure DPU parameters
- >> Build application software




```
sdx_pack -header dpu.hpp -lib libdpu.a \
-func dpu_set_start -map start=s_axi:in:0x10 -func-end \
-ip ../iprepo/dpu/component.xml -control none \
-add-ip-repo ../iprepo/src/ \
-target-family zynquplus \
-target-cpu cortex-a53 -target-os linux -verbose
```

-header <header.h/pp>: Header file with function declarations. only one top header file allowed
-lib : create a lib.a
-func <function_name> -map <swName>=<hwName>:direction:offset -func-end
-ip: <component.xml>: IP packed by the Vivado IP integrator, only one top IP allowed.

Deephi DPU IP Integration with SDSoC



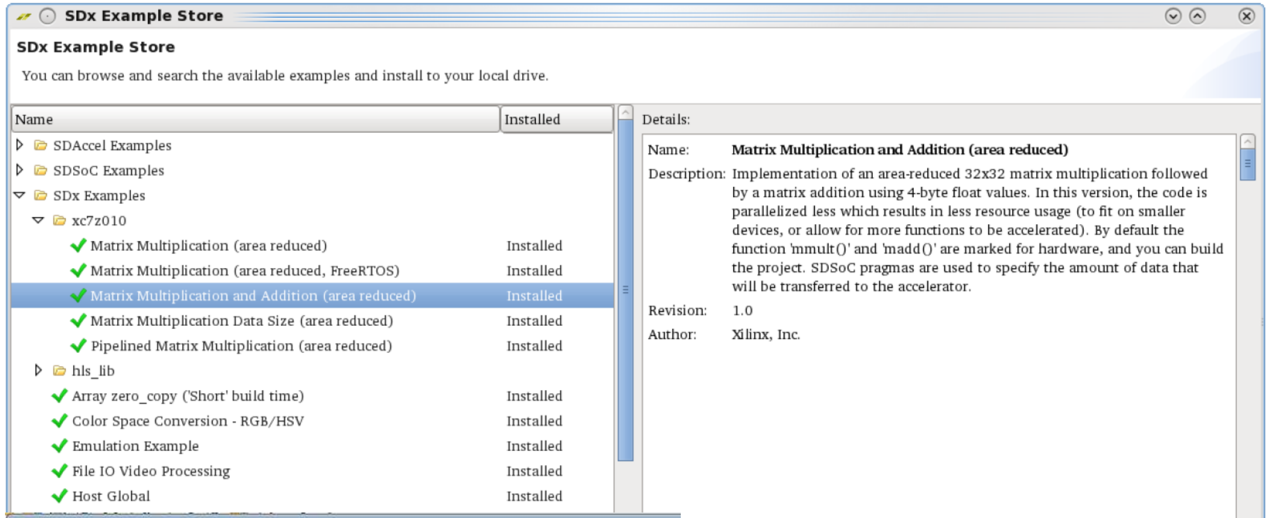
How to Use DNNK in SDSoC

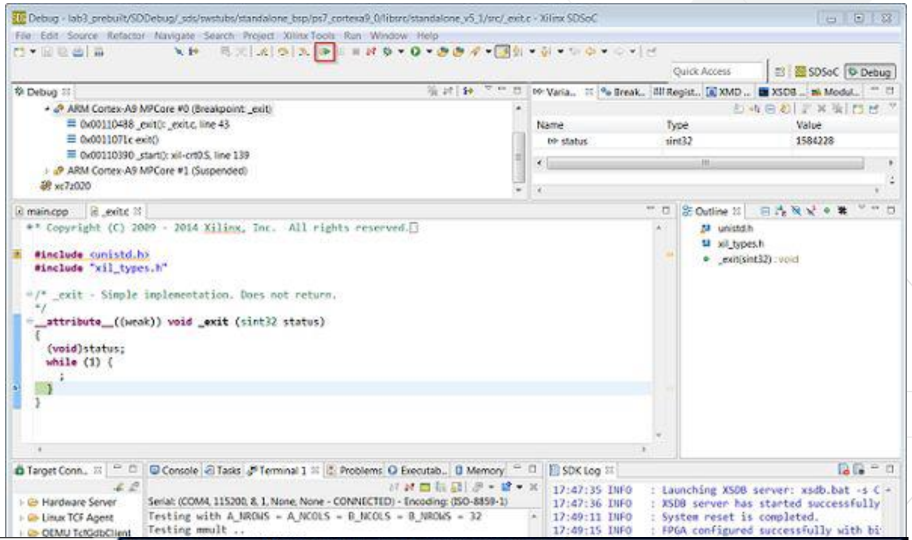
Only 3 steps!

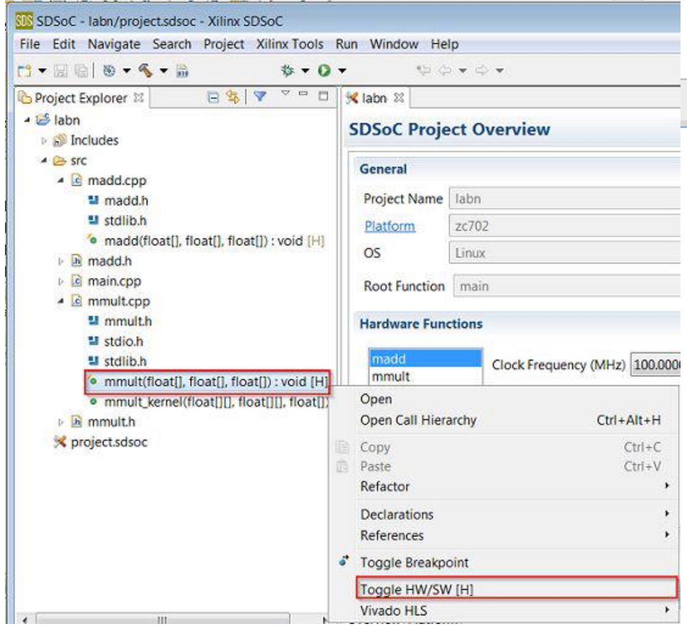
Write it → Compile it → Run it

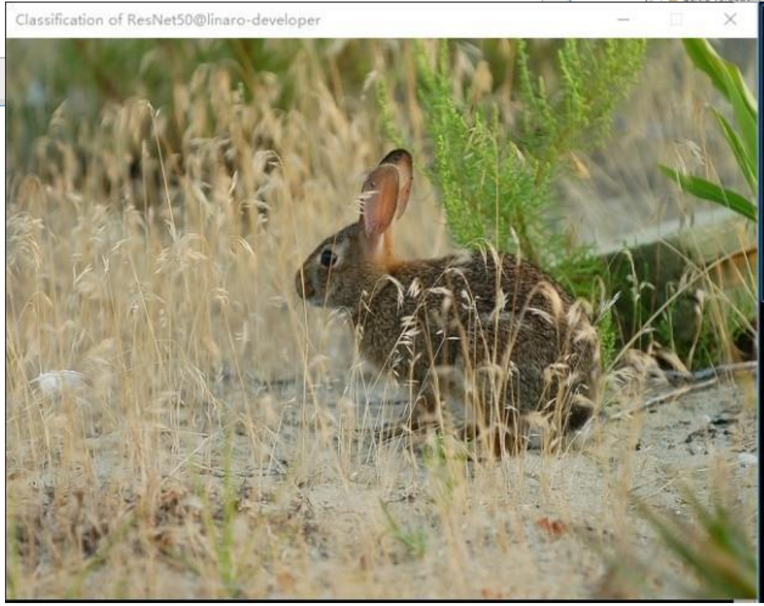
Software define development

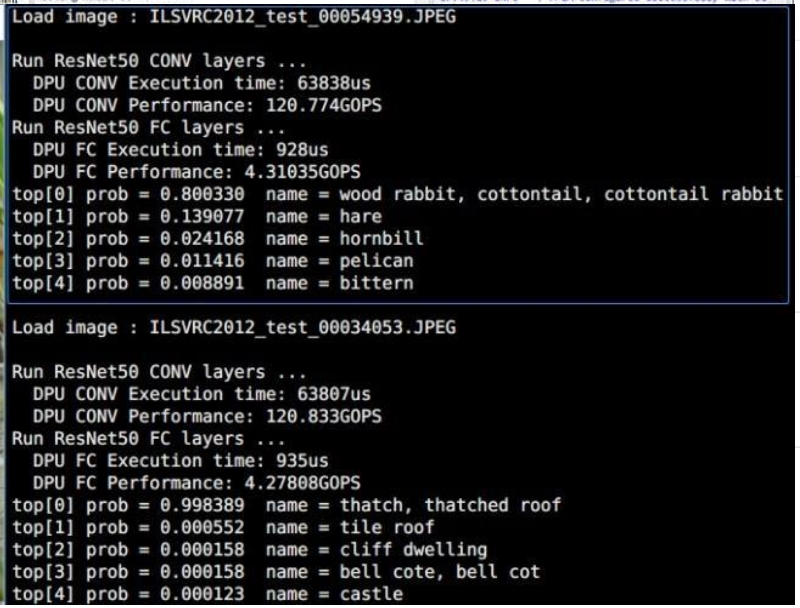
Resnet50 Example with C-callable DPU IP in SDSoC









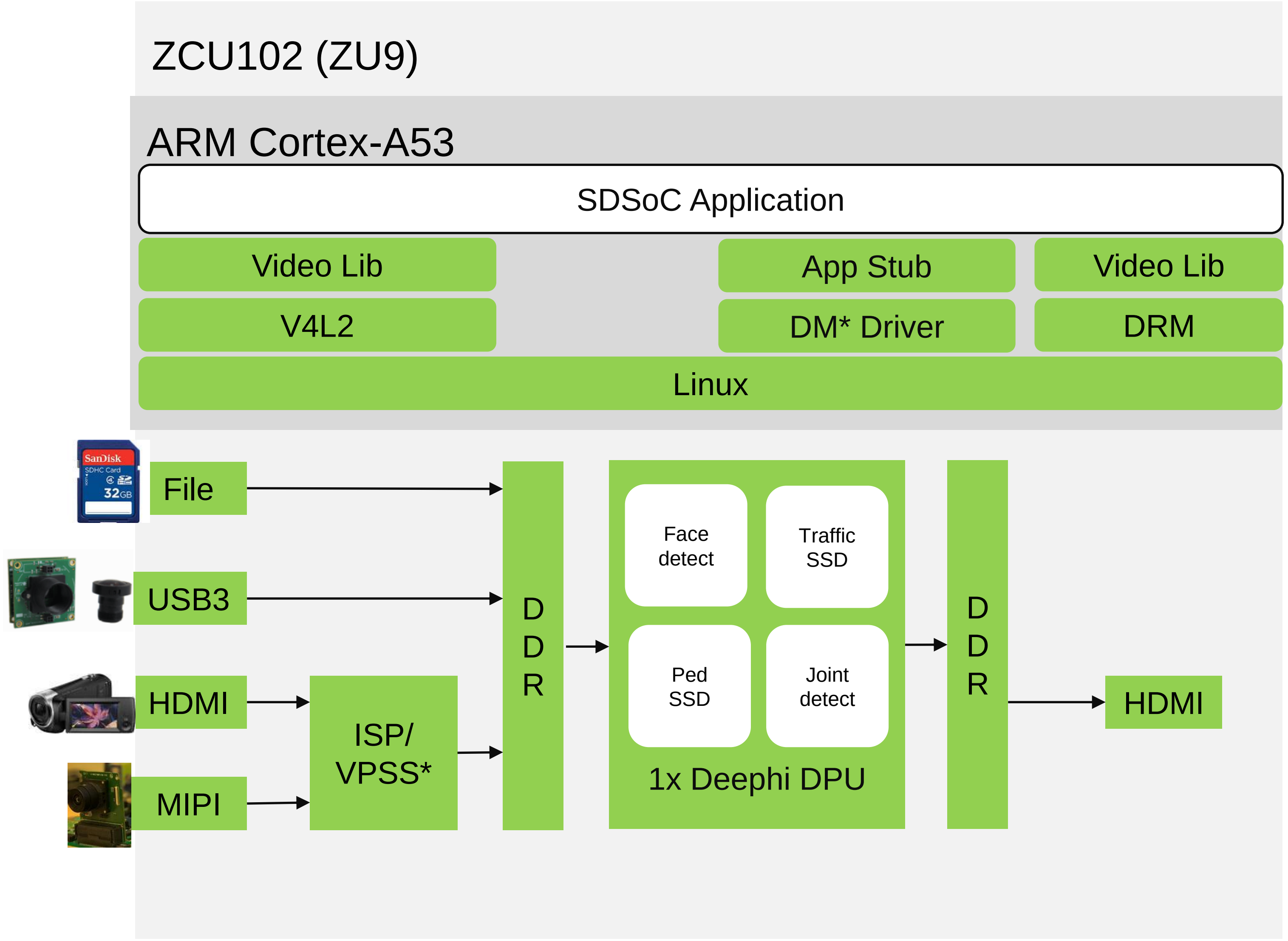


A Long Time for Every Build?

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

Free

Everything you need to do it yourself

DeePhi Basic
- Compiler
- Quantizer
- Pruned Models
- Unlimited Deployment

DeePhi Professional
- 3-day On-site Training
- Pruning Tools
- Compiler
- Quantizer
- Pruned Models
- Unlimited Deployment

Access Pruning Technology
&
3-day on-site training by a top-notch ML expert
&
30-day evaluation with encrypted pruning output

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/Embeded 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

XDF XILINX DEVELOPER FORUM