Xilinx边缘深度学习加速器的设计与应用

Alex He
AI Solution Specialist FAE
7/8/2019

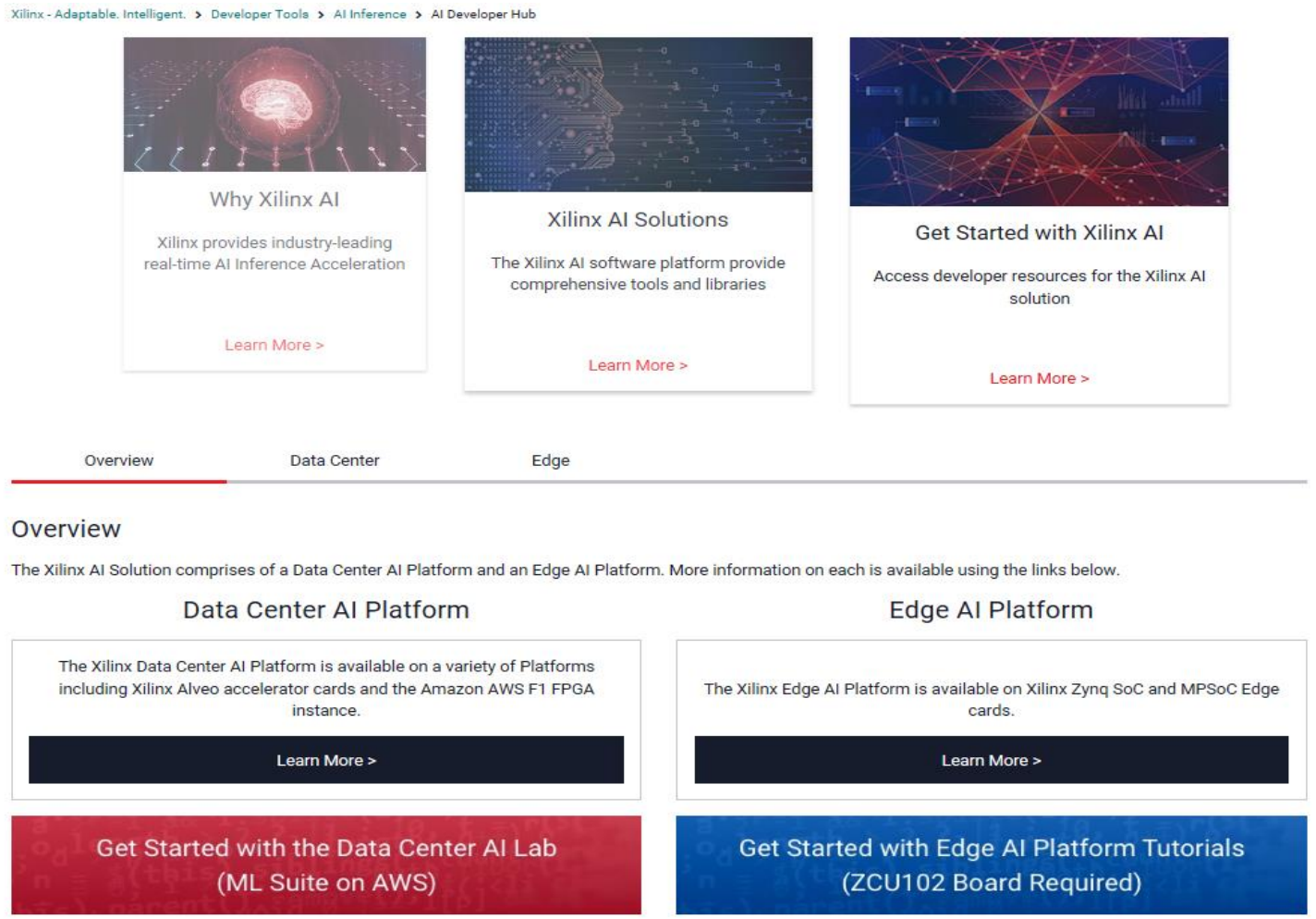
XILINX.

- Xilinx AI Overview
- Xilinx Edge AI Solution
- Xilinx Edge AI Development Flow

Xilinx AI Overview

AI Developer Hub

Xilinx AI focus on inference

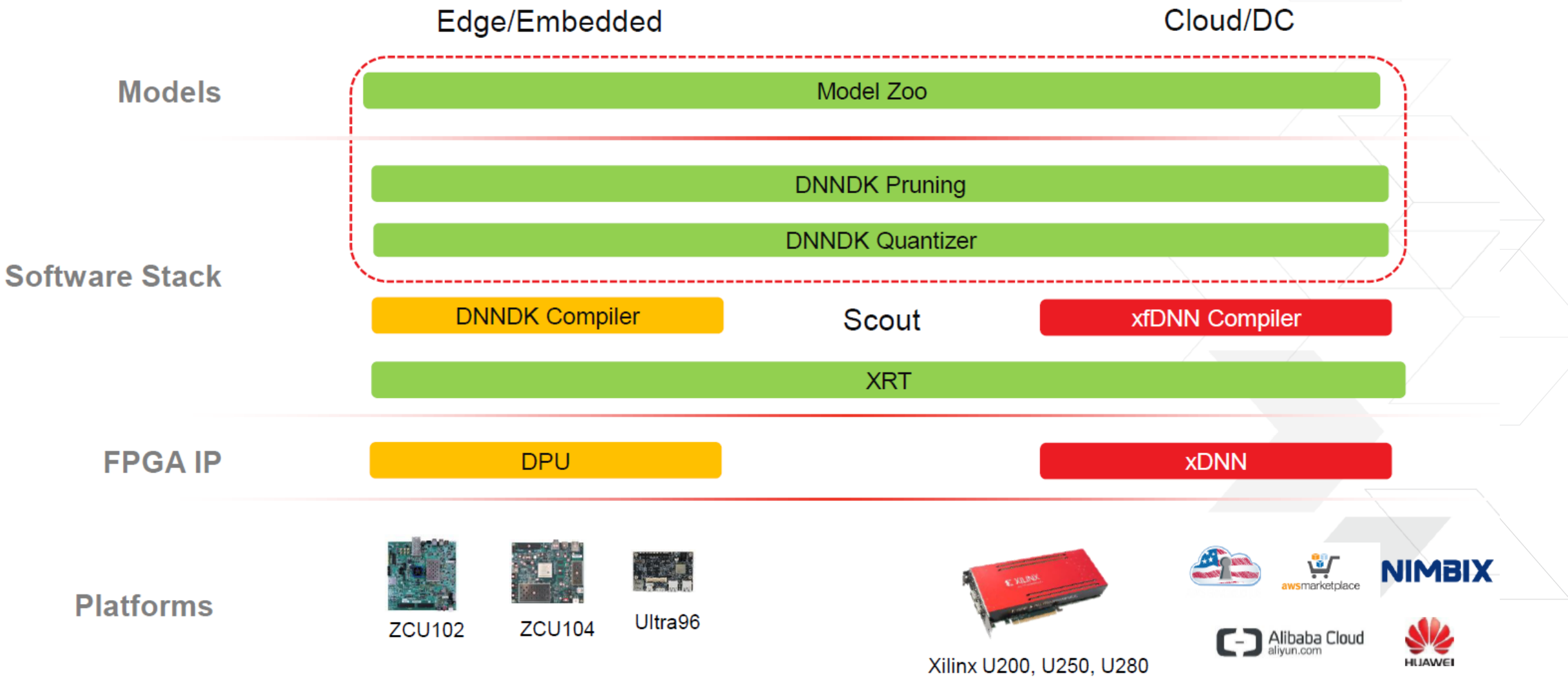


*https://www.xilinx.com/products/design-tools/ai-inference/**ai-developer-hub.html***

Separate Solutions for Edge/Embedded and Cloud/Alveo

	Edge/Embedded	Cloud/DC
Models	Deephi Models	xfDNN Models
Software Stack	Pruning	
	DNNDK Quantizer	xfDNN Quantizer
	DNNDK Compiler	xfDNN Compiler
	DNNDK Runtime	SDAccel
FPGA IP	DPU	xDNN
Platforms	 ZCU102  ZCU104  Ultra96	 Xilinx U200, U250, U280

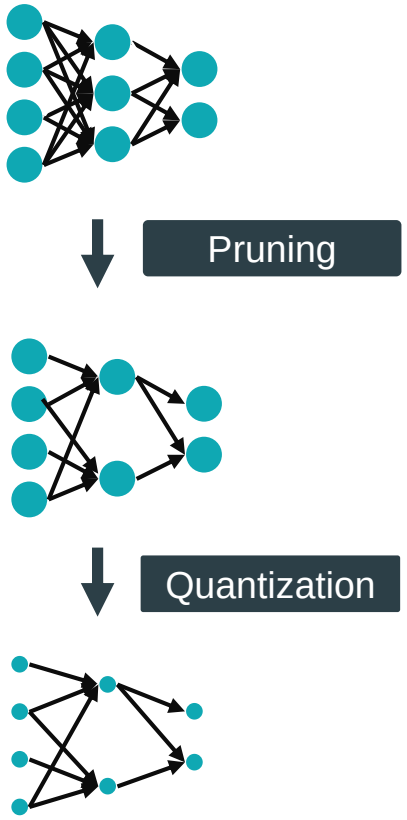
Unification Gets Started



Xilinx Edge AI Solution

XILINX

Unique, Patented Deep Learning Acceleration Techniques



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

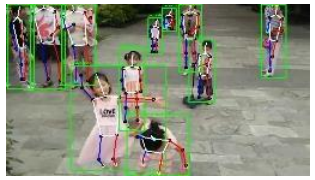
Unique Pruning Technology Provides a Significant Competitive Advantage

XILINX

Xilinx Solution Stack for Edge/Embedded ML

Models


Face detection


Pose estimation


Video analytics


Lane detection


Object detection


Segmentation

Framework

Caffe

Darknet

TensorFlow

Tools & IP

DNNDK

DPU

HW Platforms


Z7020 Board


Z7020 SOM


ZU2 SOM


ZU2/3 Card

ZU9 Card


ZCU102

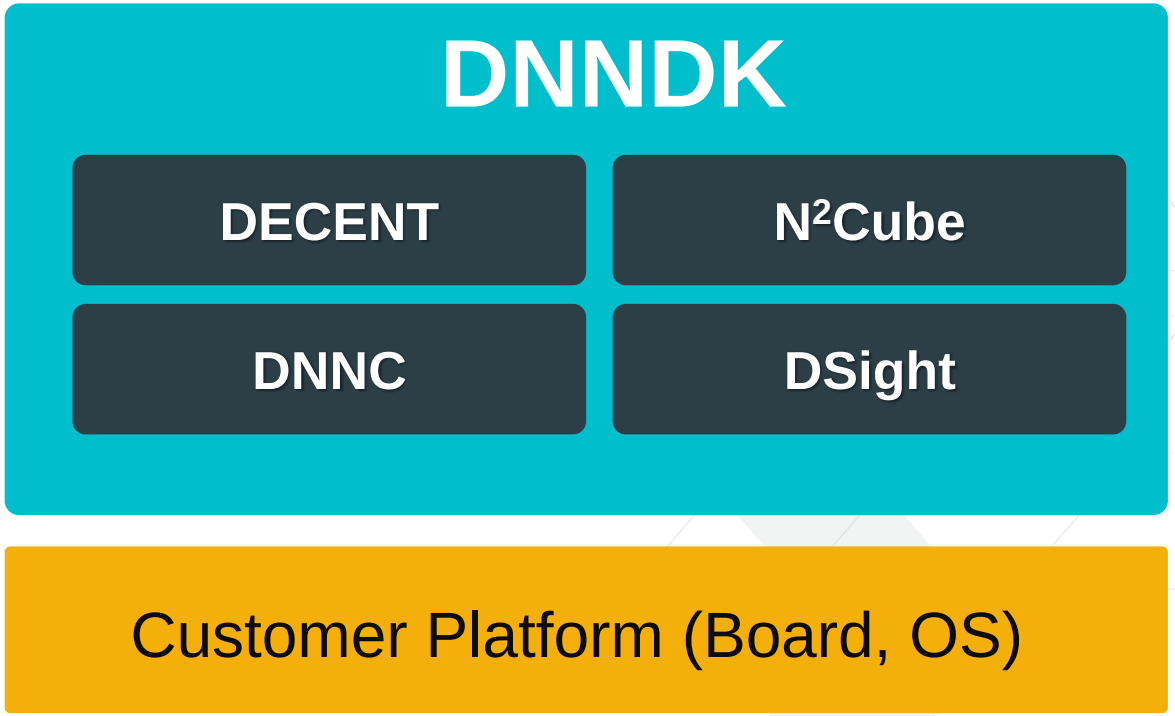

ZCU104


Ultra96

XILINX.

DNNDK – Deep Neural Network Development Kit

- DECENT (DEep ComprEssioN Tool)
- DNNC (Deep Neural Network Compiler)
- Runtime N^{2}Cube (Cube of Neural Network)
- Profiler DSight



Framework Support

Caffe

- Pruning
- Quantization
- Compilation

- Pruning
- Quantization
- Convertor to Caffe

- Quantization & Compilation
  - Beta version
- Pruning
  - Beta version

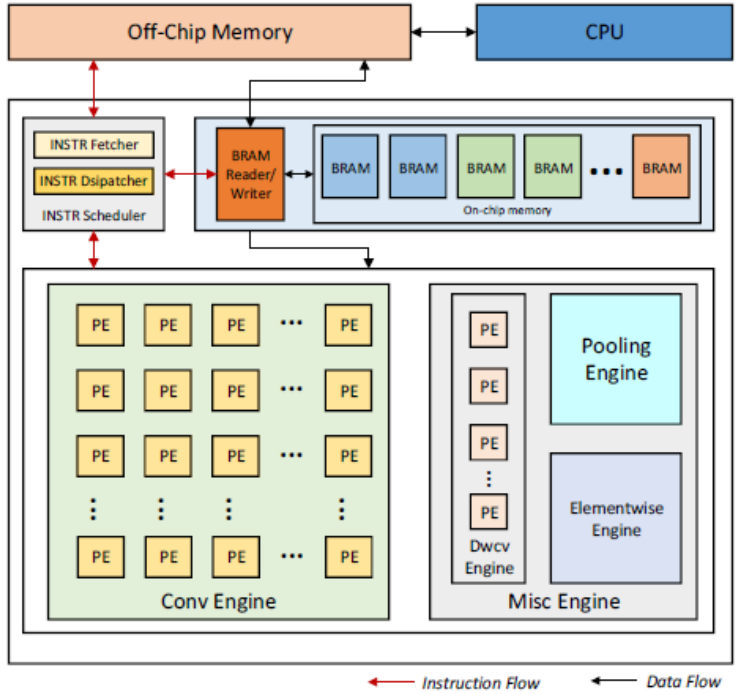
DPUv2s Architecture, Resources & Peak Performance

Table 10: Resources of Different DSP Usage

High DSP Usage					Low DSP Usage				
Arch	LUT	Register	BRAM	DSP	Arch	LUT	Register	BRAM	DSP
B512	20177	31782	69.5	98	B512	20759	33572	69.5	66
B800	20617	35065	87	142	B800	21050	33752	87	102
B1024	27377	46241	101.5	194	B1024	29155	49823	101.5	130
B1152	28698	46906	117.5	194	B1152	30043	49588	117.5	146
B1600	30877	56267	123	282	B1600	33130	60739	123	202
B2304	34379	67481	161.5	386	B2304	37055	72850	161.5	290
B3136	38555	79867	203.5	506	B3136	41714	86132	203.5	394
B4096	40865	92630	249.5	642	B4096	44583	99791	249.5	514

Table 11: DPU_EU Performance (GOPs) on Different Device

Device	DPU Configuration	Frequency (MHz)	Peak Performance
Z7020	B1152x1	200	230 Gops
ZU2	B1152x1	370	426 Gops
ZU3	B2304x1	370	852 Gops
ZU5	B4096x1	350	1.4 Tops
ZU7EV	B4096x2	330	2.7 Tops
ZU9	B4096x3	333	4.1 Tops

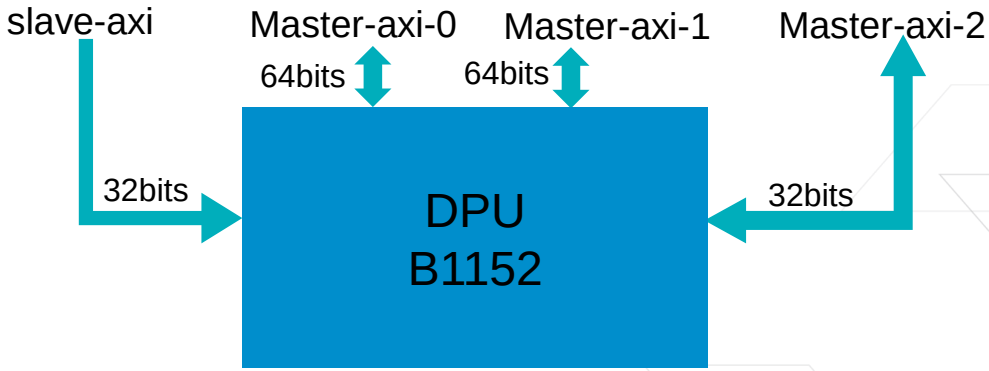


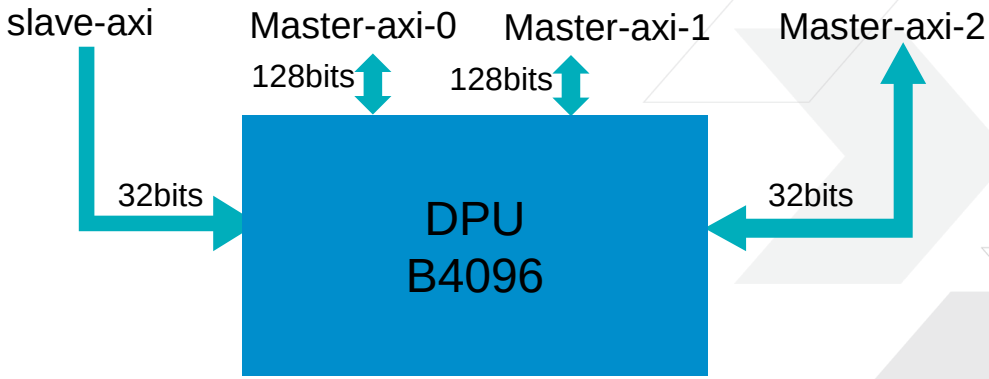
Peak Performance =
Number of DPU cores * Clock Freq * DPU Arch

XILINX.

DPU Typical Options & Interfaces

- **3-level parallelism is exploited**
  - Pixel * input channel * output channel
- **Small core - B1152**
  - Parallelism: 4*12*12
  - target Z7020/ZU2/ZU3
- **Big core - B4096**
  - Parallelism: 8*16*16
  - Target ZU5 and above





XILINX.

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/ Relu6
- 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

Edge AI Resources

https://www.xilinx.com/products/design-tools/ai-inference/ai-developer-hub.html#edge

Edge AI Tools

Product	Documentation	Tool Download	File Size	MD5 Checksum
DNNDK	DNNDK User Guide (UG1327)	xlnx_dnndk_v3.0_190531.tar.gz	3.2 GB	d1d116aa346adcb41ae0a409292d6c19
	Xilinx AI SDK User Guide (UG1354)			
	Xilinx AI SDK Programming Guide (UG1355)			
DNNDK for SDSoC	DNNDK User Guide for SDSoC (UG1331)	xilinx_dnndk_v2.08_for_sdsoc_190214.tar.gz	667 MB	7f165aff5062497e4bb69b70773c49b1

Edge AI Evaluation Boards

Product	Documentation	Image Download	DNNDK Version	File Size	MD5 Checksum
ZCU102 Kit	ZCU102 User Guide (UG1182)	xilinx-zcu102-prod-dpu1.4-2018.3-desktop-buster-2019-04-24.img.zip	v3.0	657 MB	d49eab4d293d8d1af40fcc369e1c4f53
		2018-12-04-zcu102-desktop-stretch.img.zip	v2.08	571 MB	d0d5faf8ece80b96f5591d09756d5a5d
ZCU104 Kit	ZCU104 User Guide (UG1267)	xilinx-zcu104-prod-dpu1.4-desktop-buster-2019-04-23.img.zip	v3.0	655 MB	503661dd1ee4549a562775034b95d0c8
		2018-12-04-zcu104-desktop-stretch.img.zip	v2.08	571 MB	ada2420c4afbd89efdeea741e0917e26
Avnet Ultra 96	Ultra 96 User Guide	xilinx-ultra96-prod-dpu1.4-desktop-buster-2019-05-31.img.zip	v3.0	576 MB	c9c6a5f5a772077abc8fffde6ea8f3db
		xilinx-ultra96-desktop-stretch-2018-12-10.img.zip	v2.08	566 MB	c5d2422063213b4bc4c18a3223c6adc8

Edge AI Resources (con.)

https://www.xilinx.com/products/design-tools/ai-inference/ai-developer-hub.html#edge

Edge AI Targeted Reference Designs (TRD)

Product	Documentation	Image Download	DNNDK Version	File Size	MD5 Checksum
DPU TRD v2.0	DPU IP Product Guide (PG338 v2.0)	zcu102-dpu-trd-2018-2-190531.zip	v3.0	410 MB	9ce946849d309c86e84a5e5c5fd90661
DPR TRD v1.0	DPU IP Product Guide (PG338 v1.0)	zcu102-dpu-trd-2018-2-190322.zip	v2.08	469 MB	3101cc91a5d121a8969613367b890b77

Platform Downloads

Product	Download	File Size	MD5 Checksum
ZCU102 SDSoC 2018.3 Platform for DNNDK	zcu102-rv-ss-2018-3-dnndk.tar.gz	1.3 GB	7102c6942eb65d8b9d258914f69c6eaa
ZCU104 SDSoC 2018.3 Platform for DNNDK	zcu104-rv-ss-2018-3-dnndk.tar.gz	1.3 GB	d5bc80aa8135a719e273e2ff6ca85762

Other Resources

- Go to Developer Forum for questions & discussion
- Learn how to quantize, compile and deploy pre-trained network models with Xilinx edge AI platforms
- Learn how to build a complete embedded system incorporating AI inference, OpenCV, sensor input, and display with SDSoC
- Get started with Edge AI Platform Tutorials (ZCU102 board required)

Edge AI Platform Tutorials

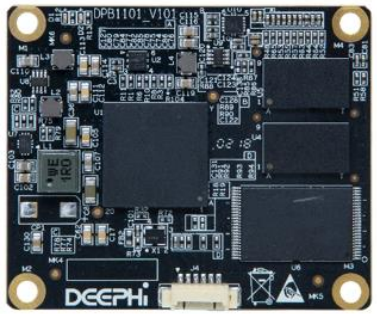
https://github.com/Xilinx/**Edge-AI-Platform-Tutorials**

Tutorial	Description
CIFAR10 Caffe Tutorial (UG1335)	Train, quantize, and prune custom CNNs with the CIFAR10 dataset using Caffe and the Xilinx® DNNDK tools.
Cats vs Dogs Tutorial (UG1336)	Train, quantize, and prune a modified AlexNet CNN with the Kaggle Cats vs Dogs dataset using Caffe and the Xilinx DNNDK tools.
ML SSD PASCAL Caffe Tutorial (UG1340)	Train, quantize, and compile SSD using PASCAL VOC 2007/2012 datasets with the Caffe framework and DNNDK tools, then deploy on a Xilinx ZCU102 target board.
DPU Integration Lab (UG1350)	Build a custom system that utilizes the Xilinx Deep Learning Processor (DPU) IP to accelerate machine learning algorithms.
Yolov3 Tutorial with Darknet to Caffe Converter and Xilinx DNNDK (UG1334)	Use the Yolov3 example, which converts the Darknet model to Caffe model and uses the DNNDK tool chain for quantization, compilation, and deployment on the FPGA.
MNIST Classification with TensorFlow (UG1337)	Learn the DNNDK v3.0 TensorFlow design process for creating a compiled `.elf` file that is ready for deployment on the Xilinx® DPU accelerator from a simple network model built using Python. This tutorial uses the MNIST test dataset.
CIFAR10 Classification with TensorFlow (UG1338)	Learn the DNNDK v3.0 TensorFlow design process for creating a compiled `.elf` file that is ready for deployment on the Xilinx® DPU accelerator from a simple network model built using Python. This tutorial uses the CIFAR-10 test dataset.

TRD for Zynq

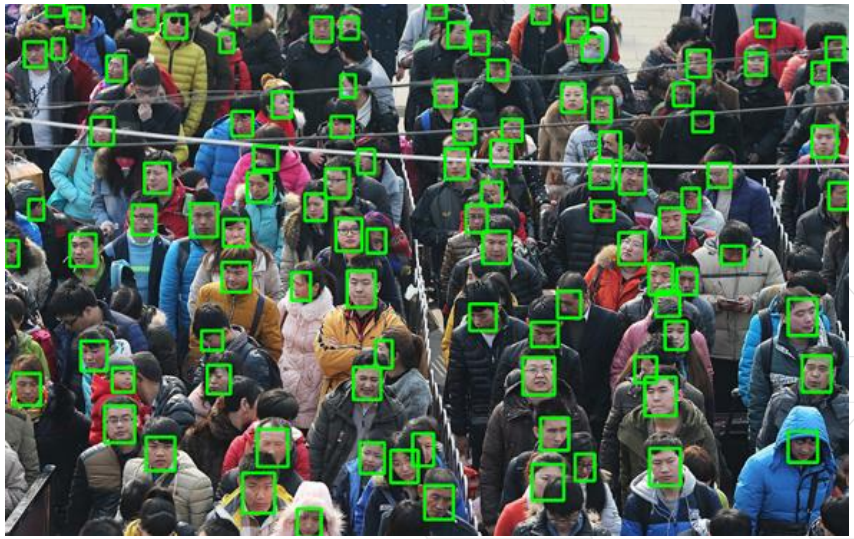
- **DPU TRD @ Zynq Board**
  - ZC702
    (Released @ gitenterprise now by Alex He, alex.he@Xilinx.com)
  - Pynq-Z2
    (Released by XUP China Team, contact to Joshua.Lu@Xilinx.com)
  - Zedboard
    (Released by Avnet)

Video Surveillance ML Solutions

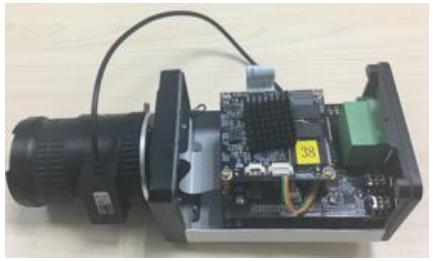


Intelligent
IP Camera Solution

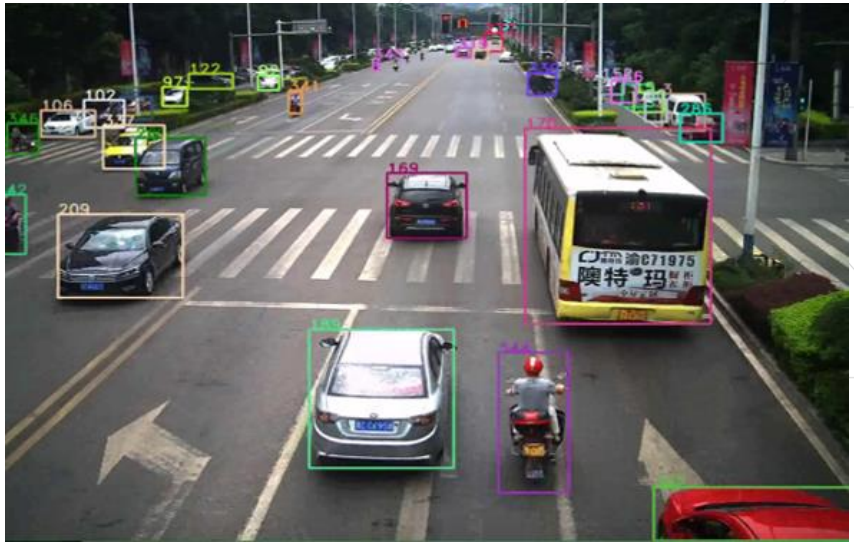
Video Analytics
Acceleration Solution



Face recognition camera
with Zynq7020

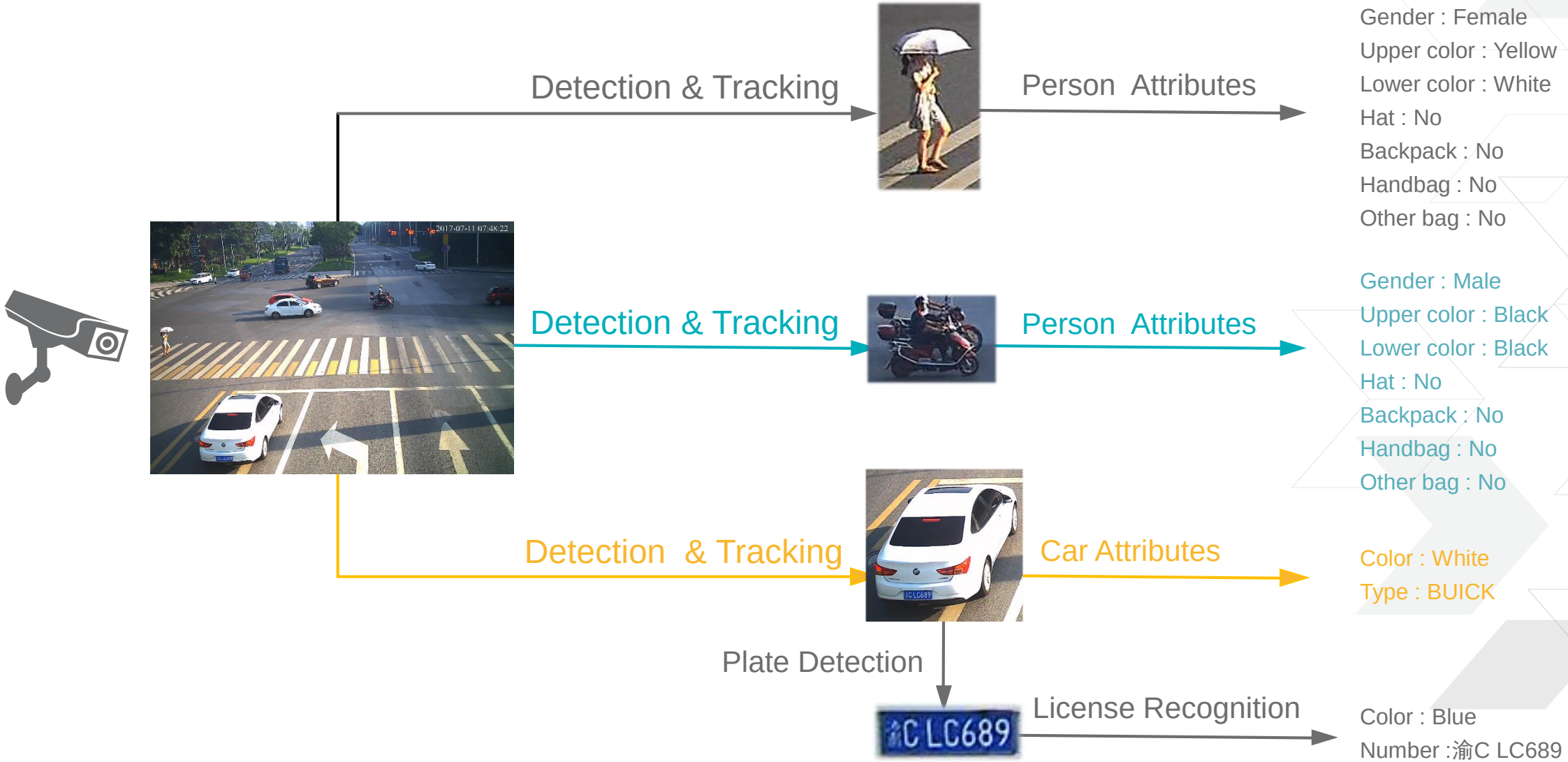


12-channel 1080P Video Analytics
with ZU9EG



XILINX.

Video Surveillance ML Ref Design



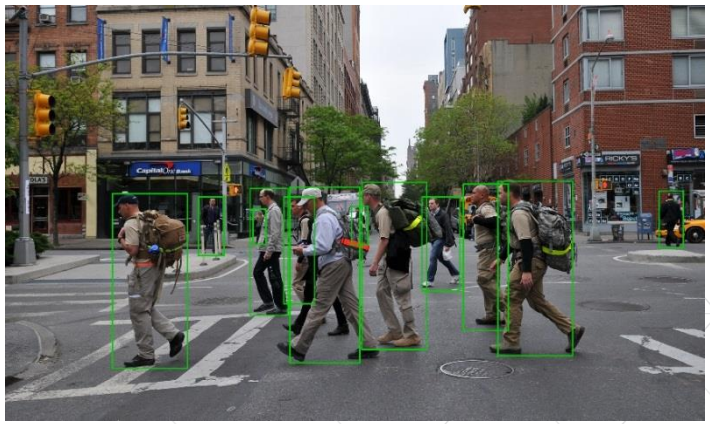
XILINX.

ADAS/AD ML Reference Design

2D/3D Object Detection

Lane Detection

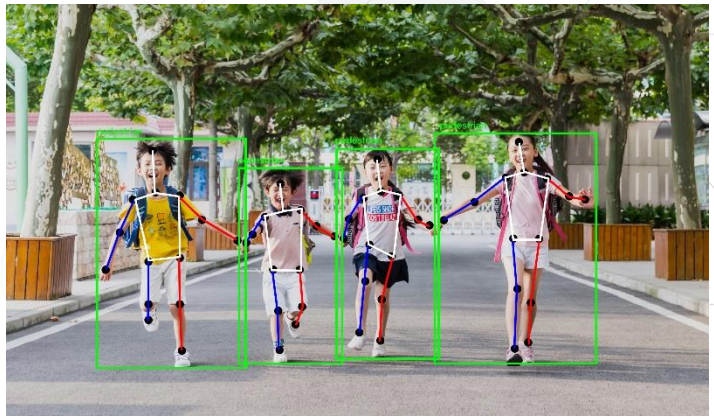
Pedestrian Detection



Segmentation + Detection

Segmentation

Pose Estimation



XILINX.

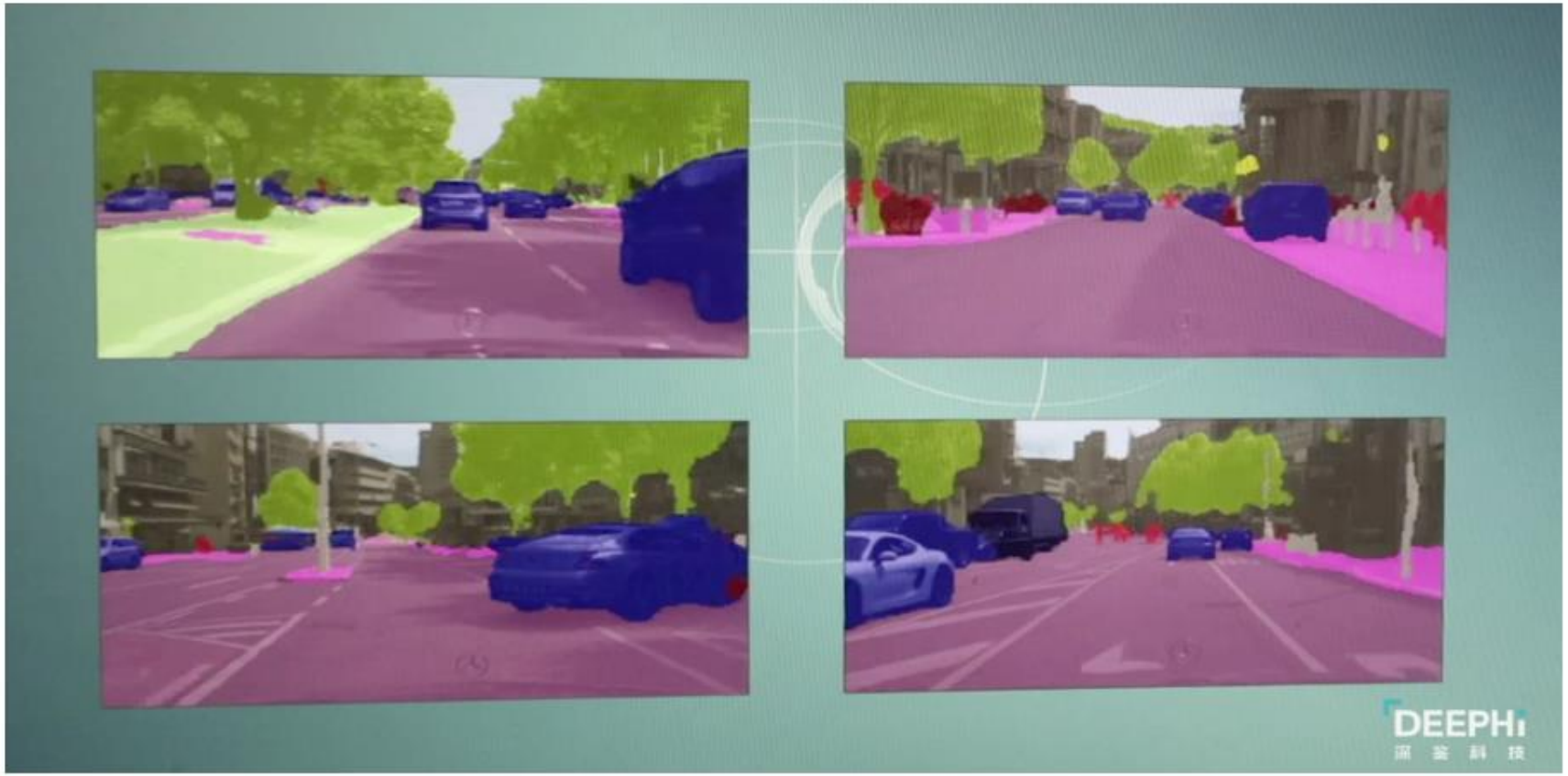
8CH Detection Demo

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

*Removed Video

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

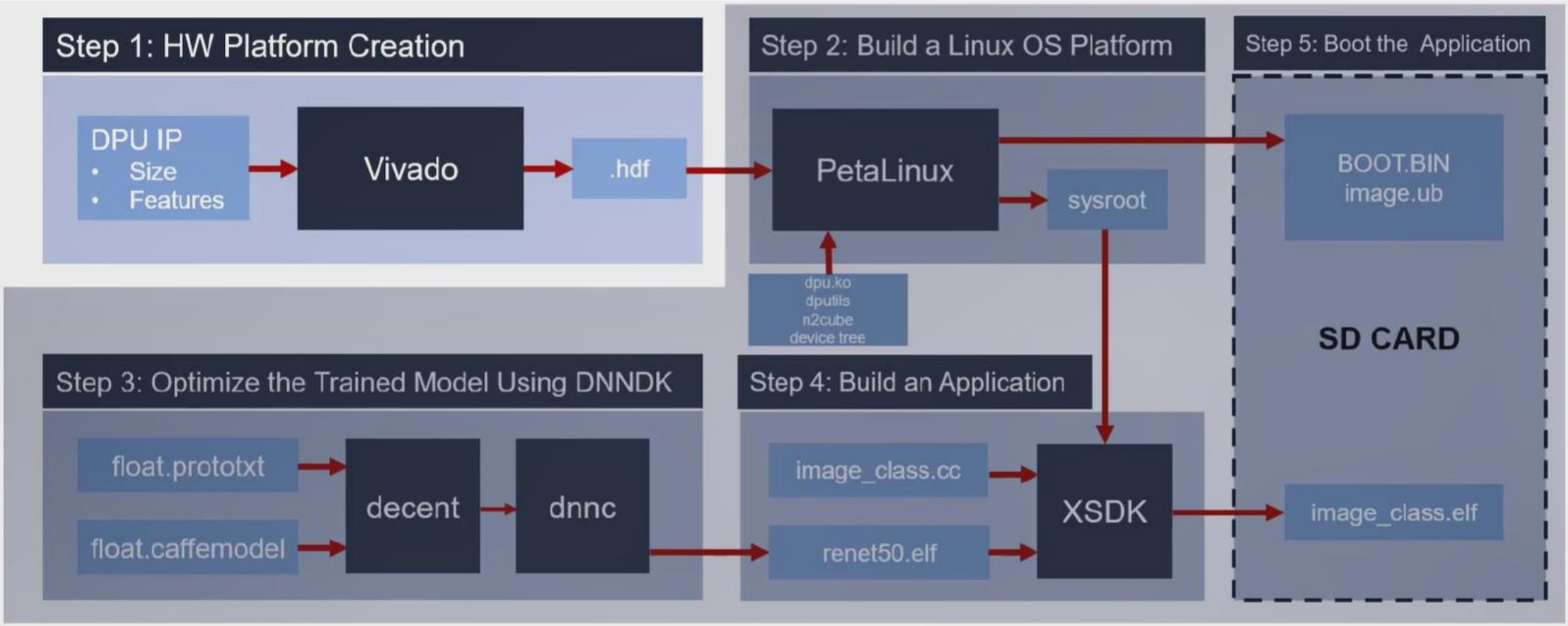


*Removed Video

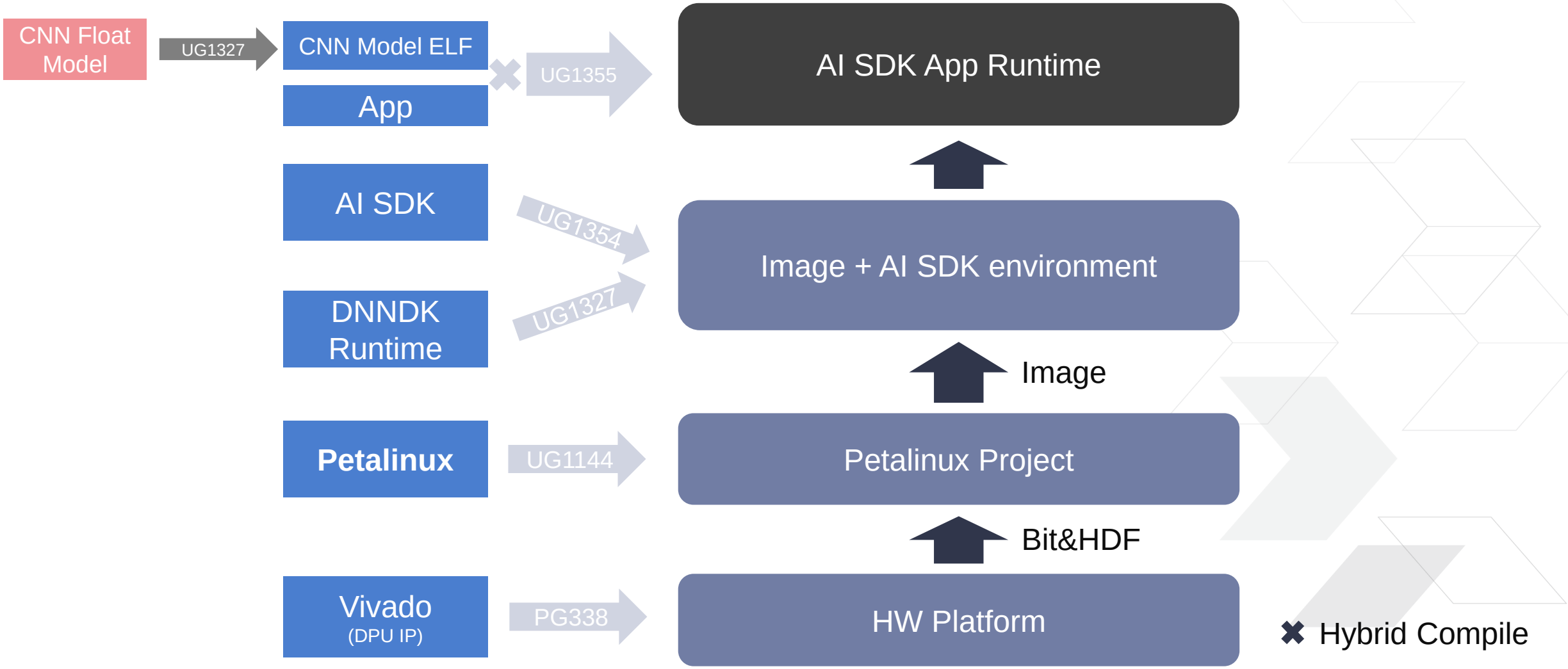
Xilinx Edge AI Development Flow

XILINX

Vivado & SDK Dev Flow with DPU & DNNDK



Documents in Development Flow



XILINX.

DNNDK Dev Flow

Five Steps with DNNDK

01 Model Compression

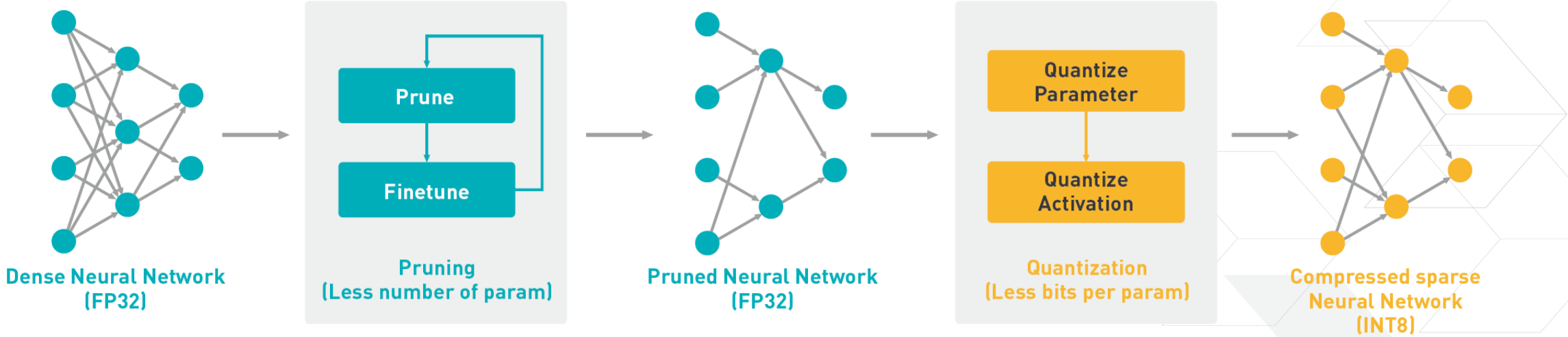
02 Model Compilation

03 Programming

04 Hybrid Compilation

05 Execution

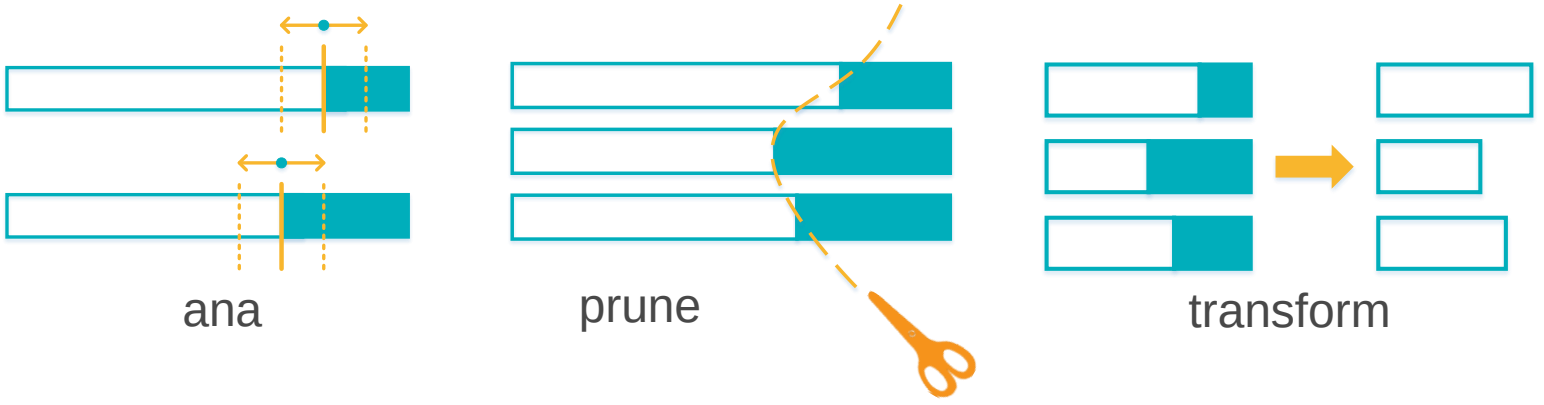
DECENT – Xilinx Deep Compression Tool

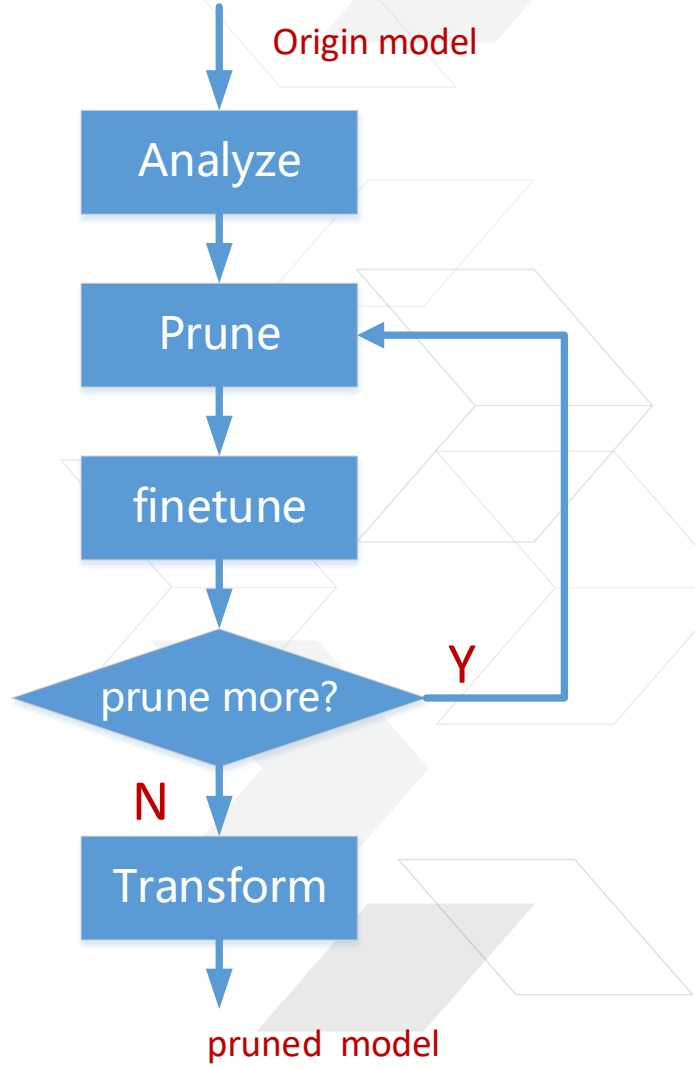


XILINX.

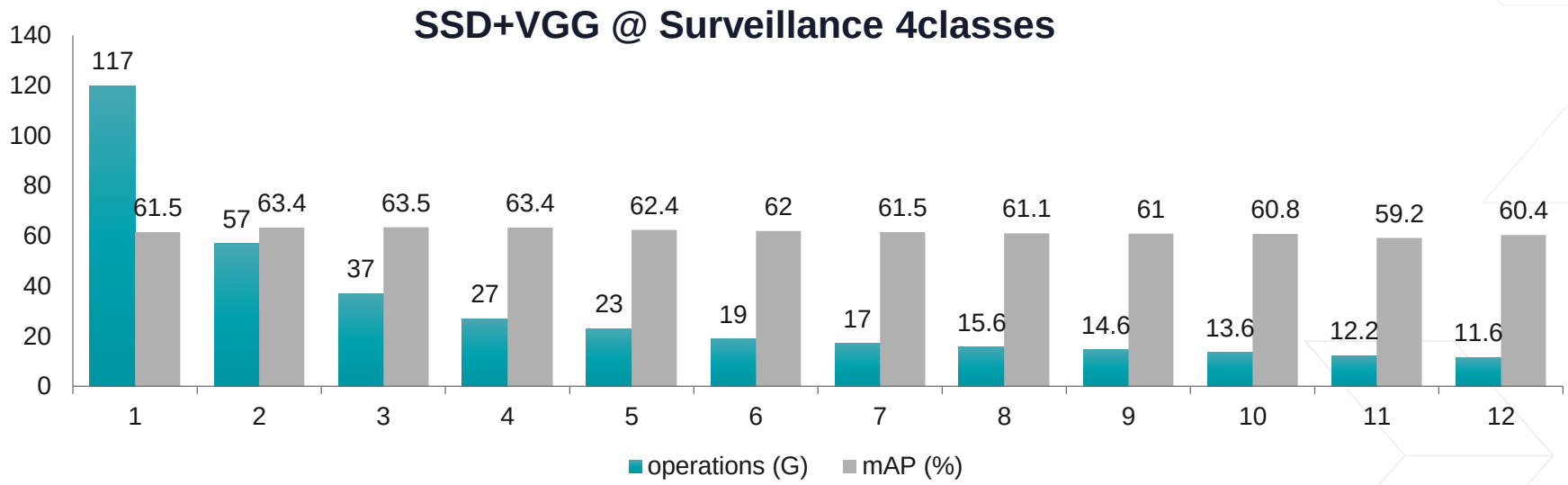
Pruning Tool – decent_p

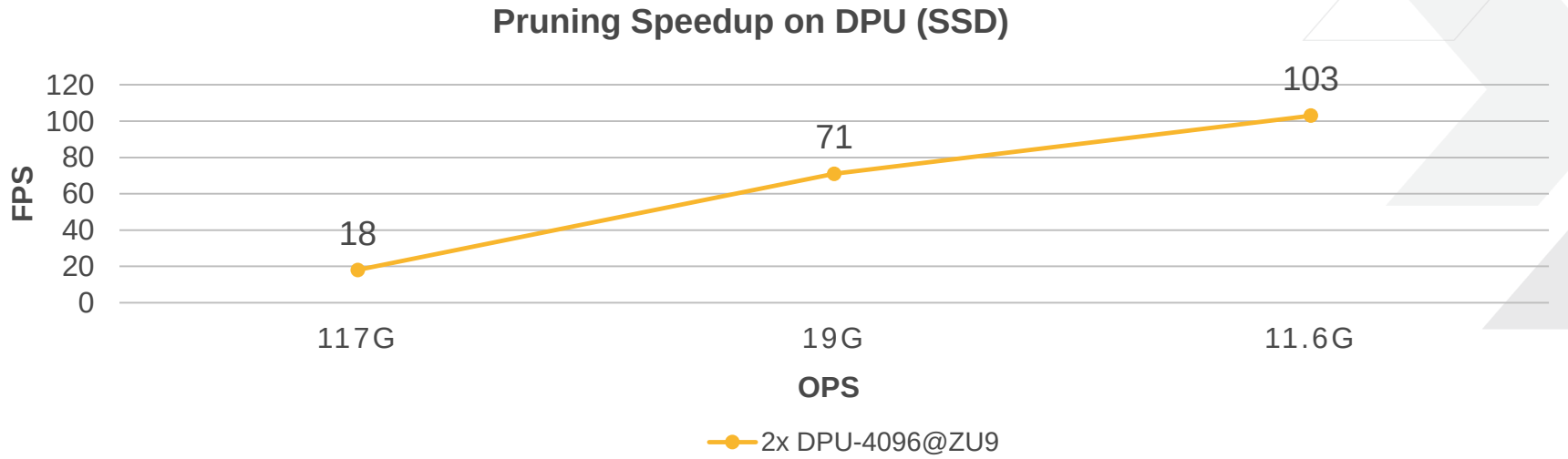
- **4 commands in decent_p**
  - Ana
    - analyze the network
  - Prune
    - prune the network according to config
  - Finetune
    - finetune the network to recover accuracy
  - Transform
    - transform the pruned model to regular model





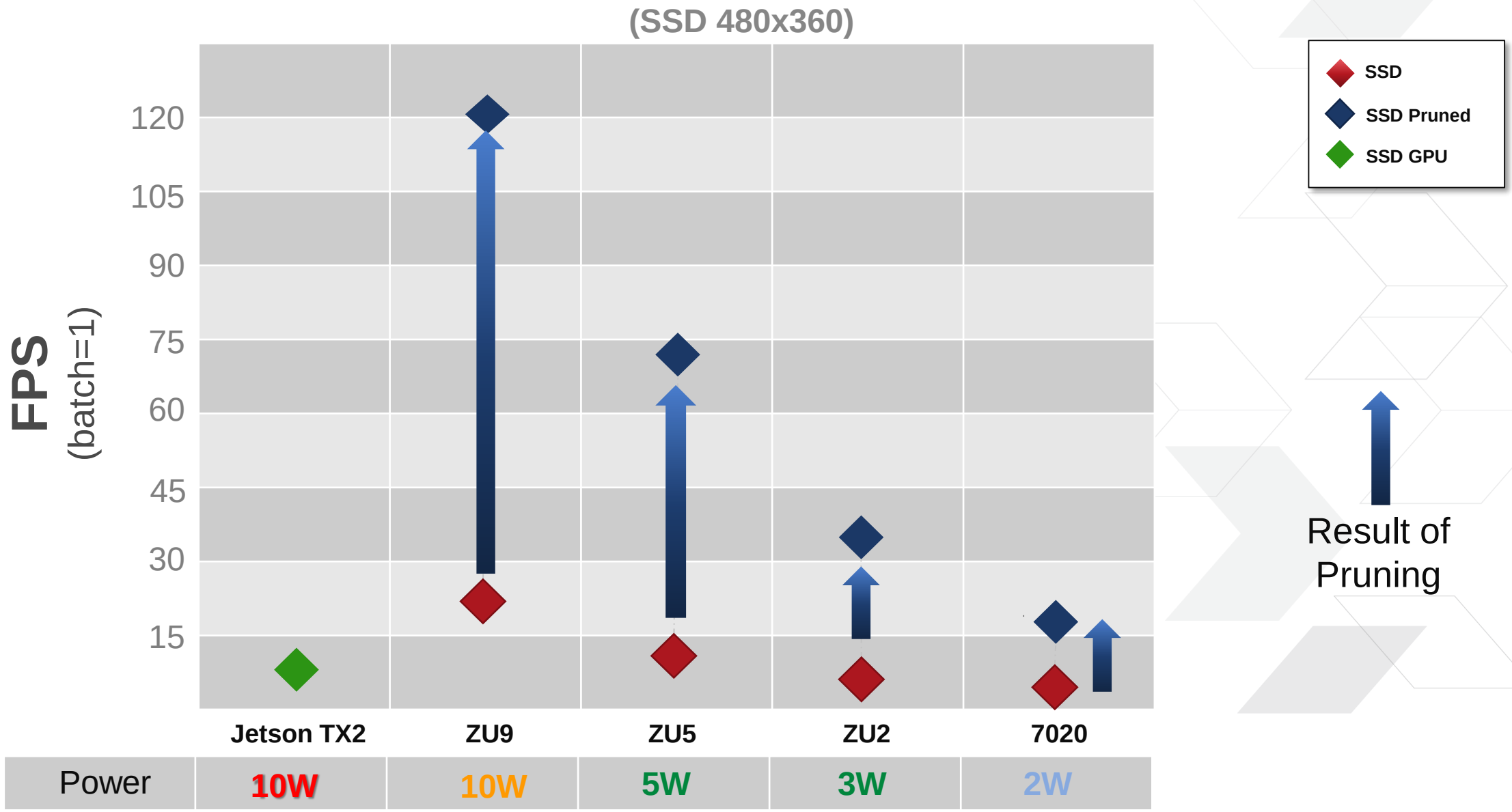
Pruning Example - SSD





XILINX.

Pruning Makes Big Difference



XILINX.

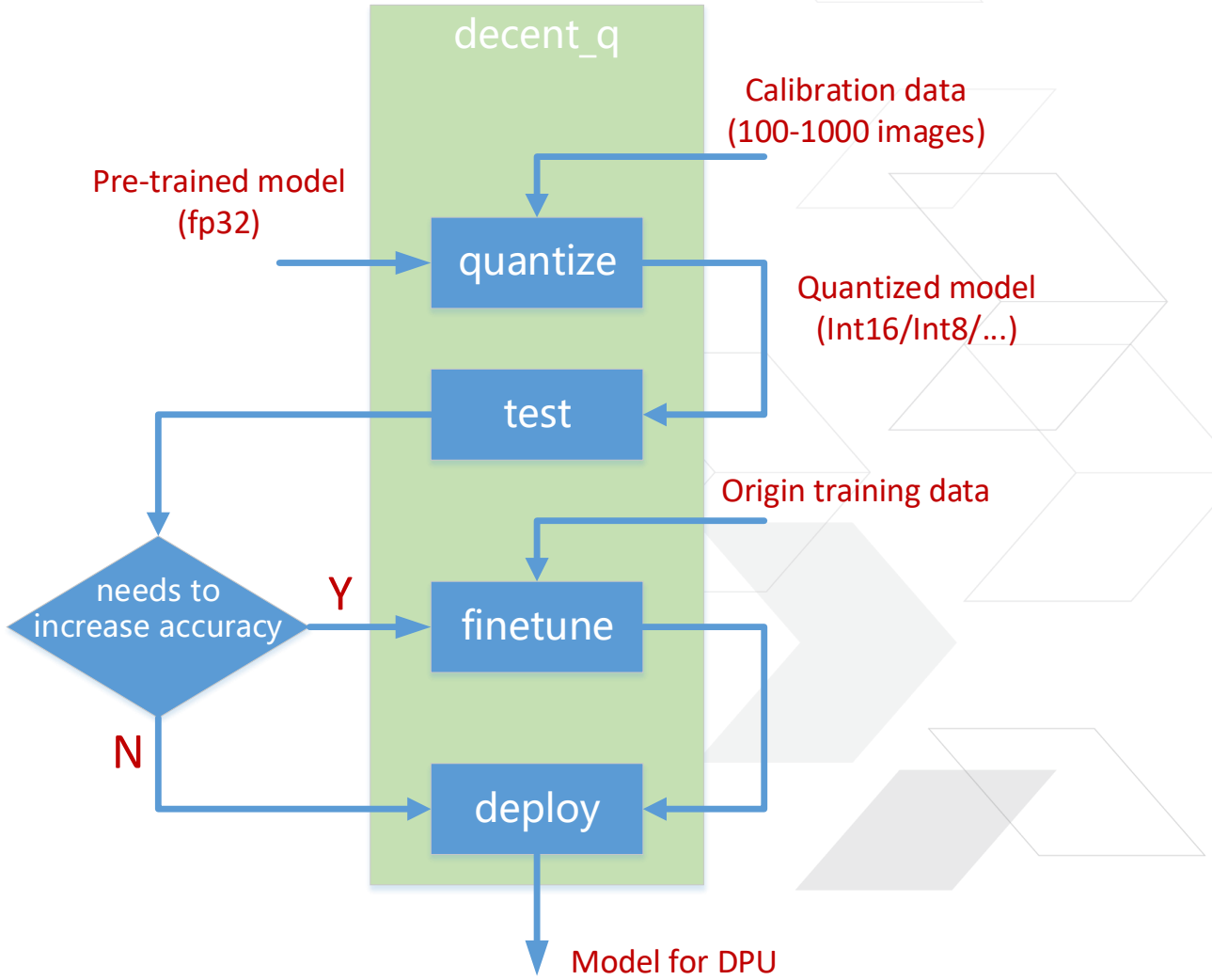
Pruning Results

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%

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



XILINX

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()

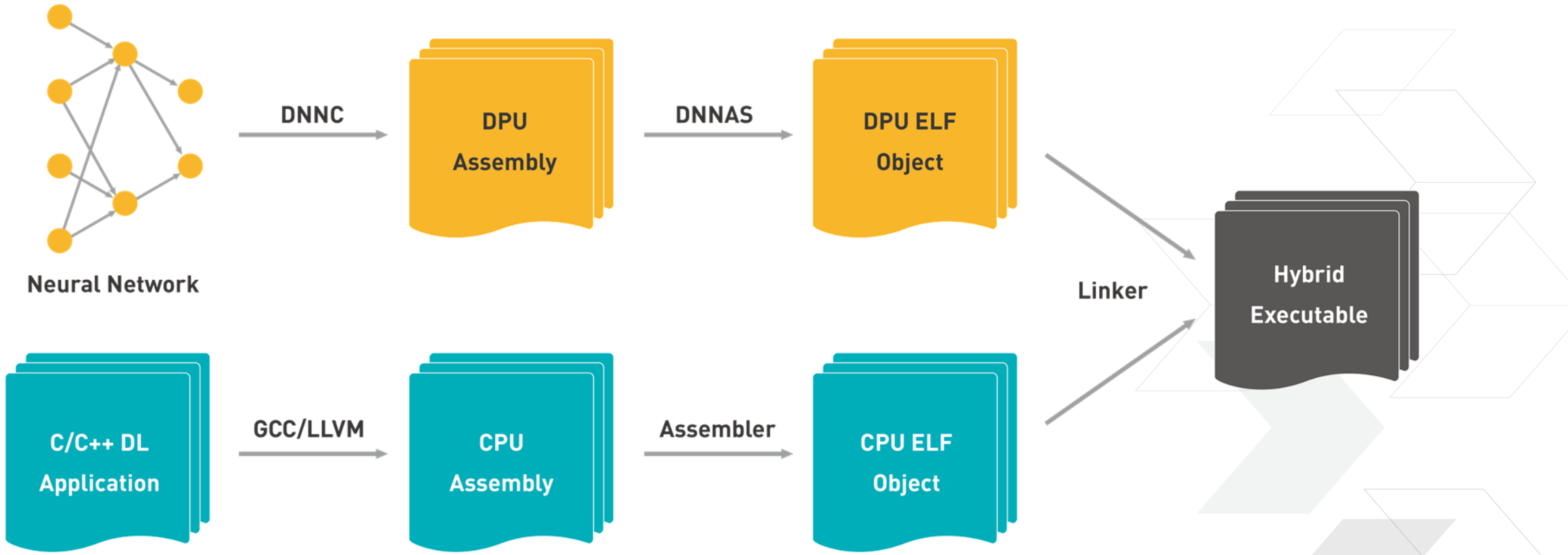
dpuGetOutputTensorSize()
dpuGetOutputTensorScale()
dpuGetOutputTensorHeight()
dpuGetOutputTensorWidth()
dpuGetOutputTensorChannel()
dpuGetTensorSize()
dpuGetTensorAddress()
dpuGetTensorScale()
dpuGetTensorHeight()
dpuGetTensorWidth()
dpuGetTensorChannel()
dpuSetIntputTensorInCHWInt8()
dpuSetIntputTensorInCHWFP32()
dpuSetIntputTensorInHWCInt8()
dpuSetIntputTensorInHWCFP32()
dpuGetOutputTensorInCHWInt8()
dpuGetOutputTensorInCHWFP32()
dpuGetOutputTensorInHWCInt8()
dpuGetOutputTensorInHWCFP32()

- **High-level Tensor-based APIs**
- **Please refer to DNNDK User Guide**

Programming with DNNDK API

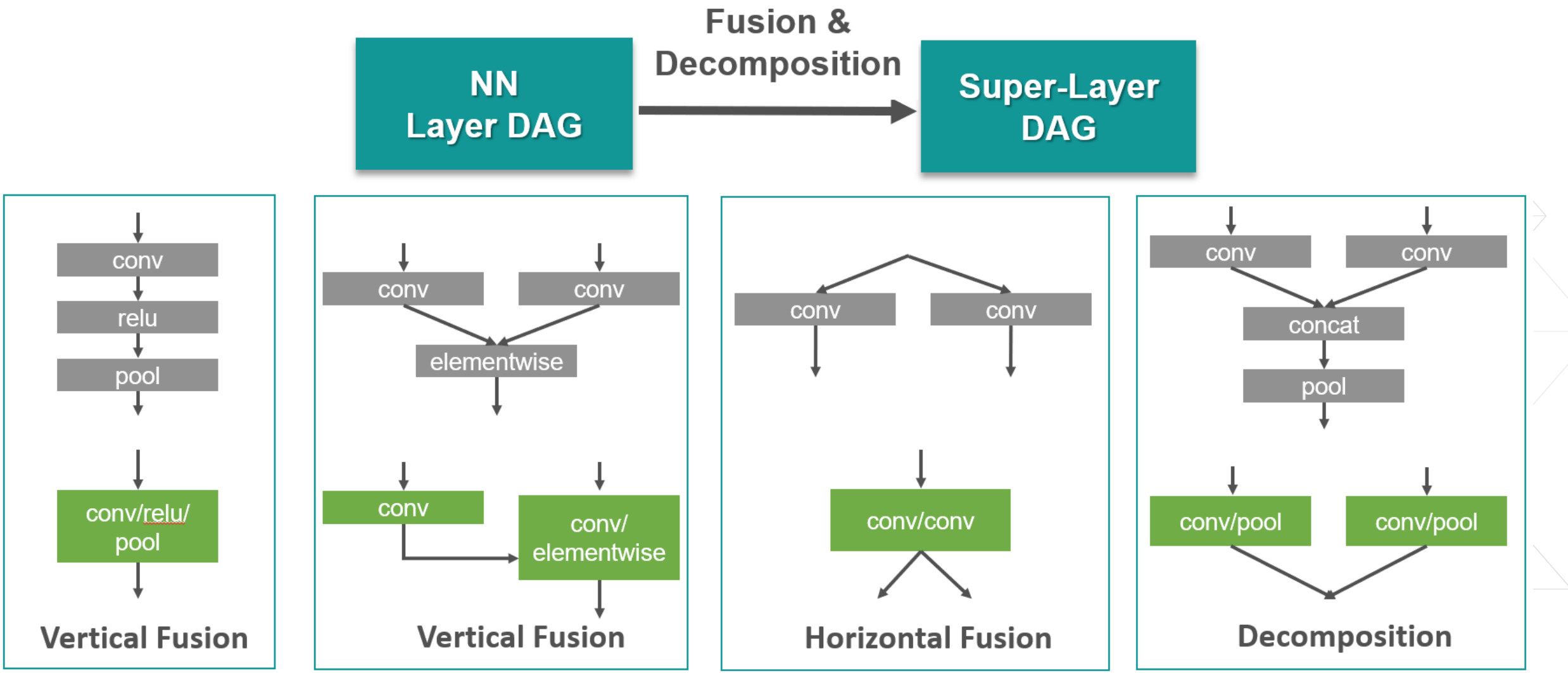
```c
int main(int argc, char *argv[])
{
    DPUKernel *kernel_conv;
    DPUKernel *kernel_fc;
    DPUTask   *task_conv;
    DPUTask   *task_fc;
    char      *input_addr;
    char      *output_addr;

    /* 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



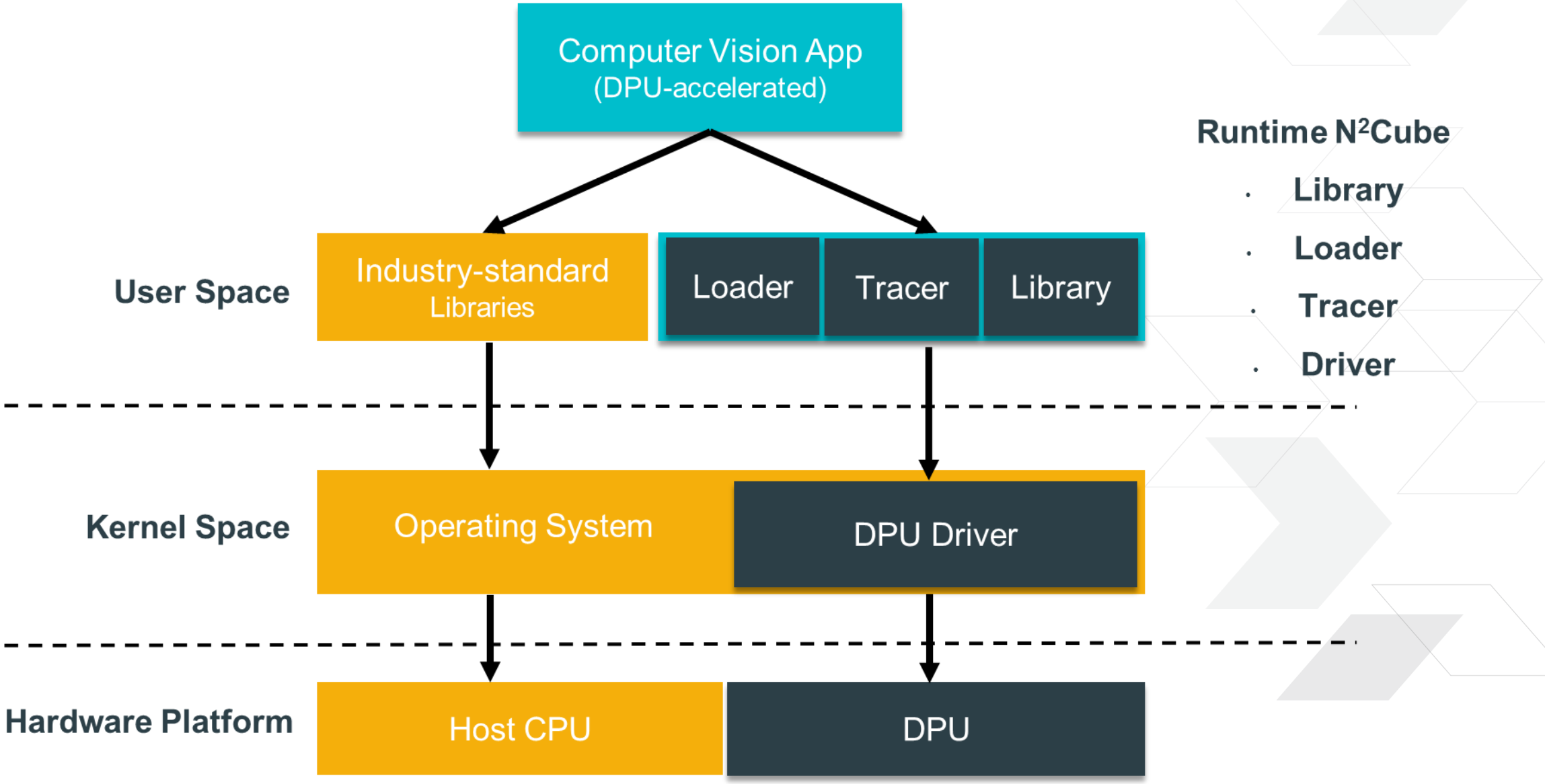
XILINX.

Optimization in DNNC



XILINX.

DNNDK Runtime Engine



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)	Xilinx	✔		

Adaptable.
Intelligent.

XILINX