



Giải pháp HW xử lý cho AI-Vision

- ◆ 1. Hardware overview
- ◆ 2. Deployment solution
- ◆ 3. Complexcity and Optimization
- ◆ 4. Benchmark result
- ◆ 5. Conclution

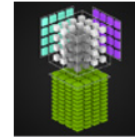


1. Hardware overview



◆ Architecture

- ◆ INTEL: SKYPE-COFFE-ICE... LAKE
- ◆ SHAVE: Hybrid RISC-DSP-GPU VLIW architecture
- ◆ TPU
- ◆ GPU
- ◆ FPGA



TENSOR CORE

Equipped with 640 Tensor Cores, Tesla V100 delivers 125 teraFLOPS of deep learning performance. That's 12X Tensor FLOPS for DL Training, and 6X Tensor FLOPS for DL Inference when compared to NVIDIA Pascal™ GPUs.

◆ Parameter of performance

- ◆ OPS: Operation per second
- ◆ TOPS – (INT8, INT4): Tera (10¹²) operation (INT8, INT4) per second
- ◆ TFLOPS – (FP32, FP16, Mix): Tera operation (FP32, FP16, Mix) per second
- ◆ $\text{FLOPS}(\text{FP32}) = \sim 2 \times \text{FLOPS}(\text{FP16}) = \sim 2^3 \times \text{FLOPS}(\text{Mix}) = \sim 2^4 \text{ TOPS}(\text{int8}) = \sim 2^5 \text{ TOPS}(\text{int4})$



Hardware overview



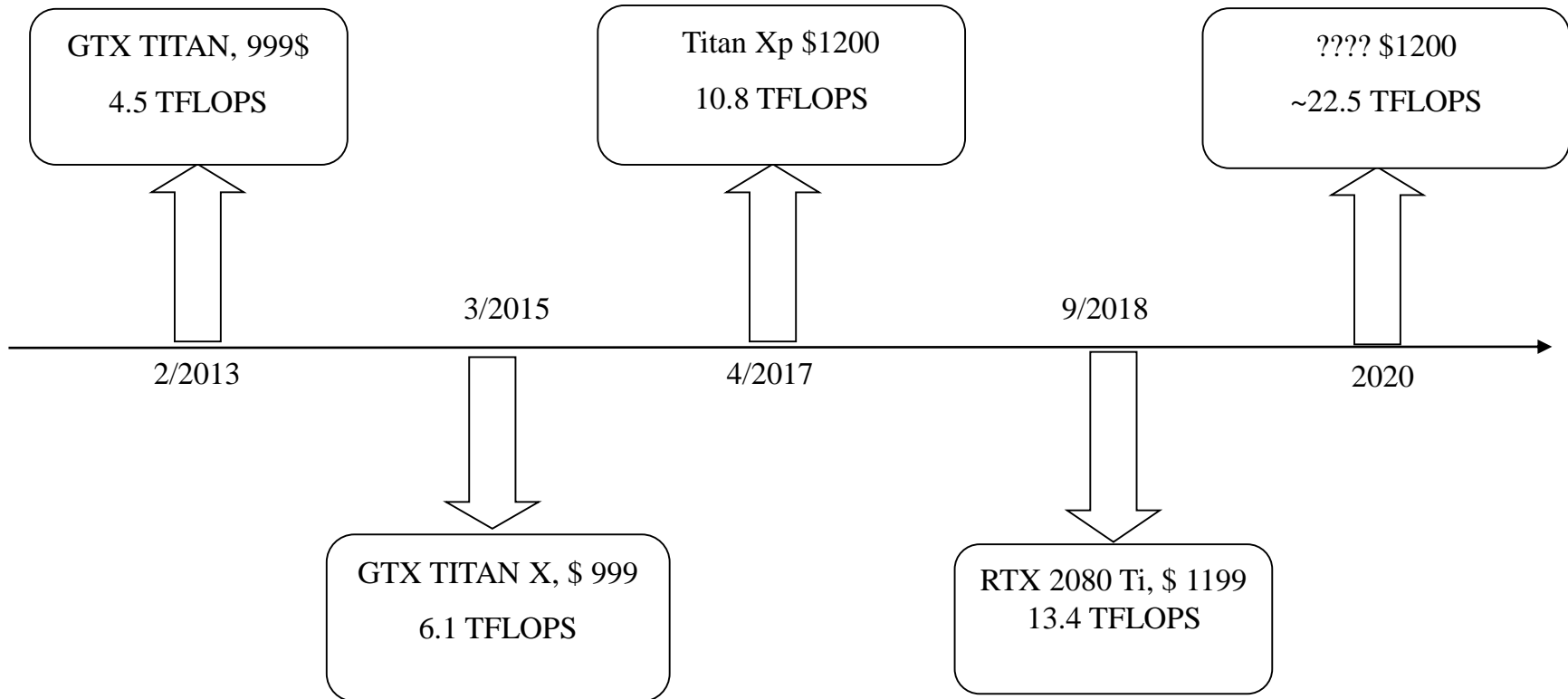
FLOPS(FP32) ~ 2 x FLOPS(FP16) ~ 2³ x FLOPS(Mix) ~ 2⁴ TOPS(int8) ~ 2⁵ TOPS (int4)

No	Board	Vendor	OPS		GOP/ W	Frameworks	Prices (eval..)
			FLOPS	TOPS			
1	RTX 2080 Ti	NVIDIA	13.45 TFLOPS (FP32)	N/A	N/A	TensorFlow, PyTorch, Caffe, MXNet, Theano	2000\$
2	Tesla T4	NVIDIA	8.1 TFLOPS (FP32) 65 TFLOPS (Mix)	130 TOPS (int8)	1850	TensorFlow, PyTorch, Caffe, MXNet, Theano	3200\$
3	Tesla V100 PCIe	NVIDIA	14 TFLOPS (FP32) 112 TFLOPS (Mix)	N/A	N/A		13000\$
4	U250	Xilinx	N/A	33.3 TOPS (int8)	151	Caffe, tensorflow	6995\$
5	Jetson Nano	NVIDIA	472 GFLOPS	N/A	94.4	TensorFlow, PyTorch, Caffe, MXNet, Theano	99\$
6	Coral Edge TPU	Google	N/A	4 TOPS (int8)	2000	TensorFlow Lite	115\$
7	AI Core X, Movidius	AAEON (ASUS)	N/A	1 TOPS (int8)	400	Caffe, tensorflow	79\$
8	ZCU104	Xilinx	N/A	2.4 TOPS (int8)	N/A	Caffe, tensorflow	895\$
9	Sipeed MAIX	Seeed Studio (China)	N/A	250 -500 GOPS (int8)	833.3	N/A	5\$ (chip)
10	Hailo	HAILO	N/A	26 TOPS (int8)	15500	N/A	N/A





- ◆ **Desktop History: Increase TFLOPS ~30% each year**
- ◆ **Prices 2018 (Full GPU+CPU...): ~200\$/TFLOPS; N/A\$/TOPS**

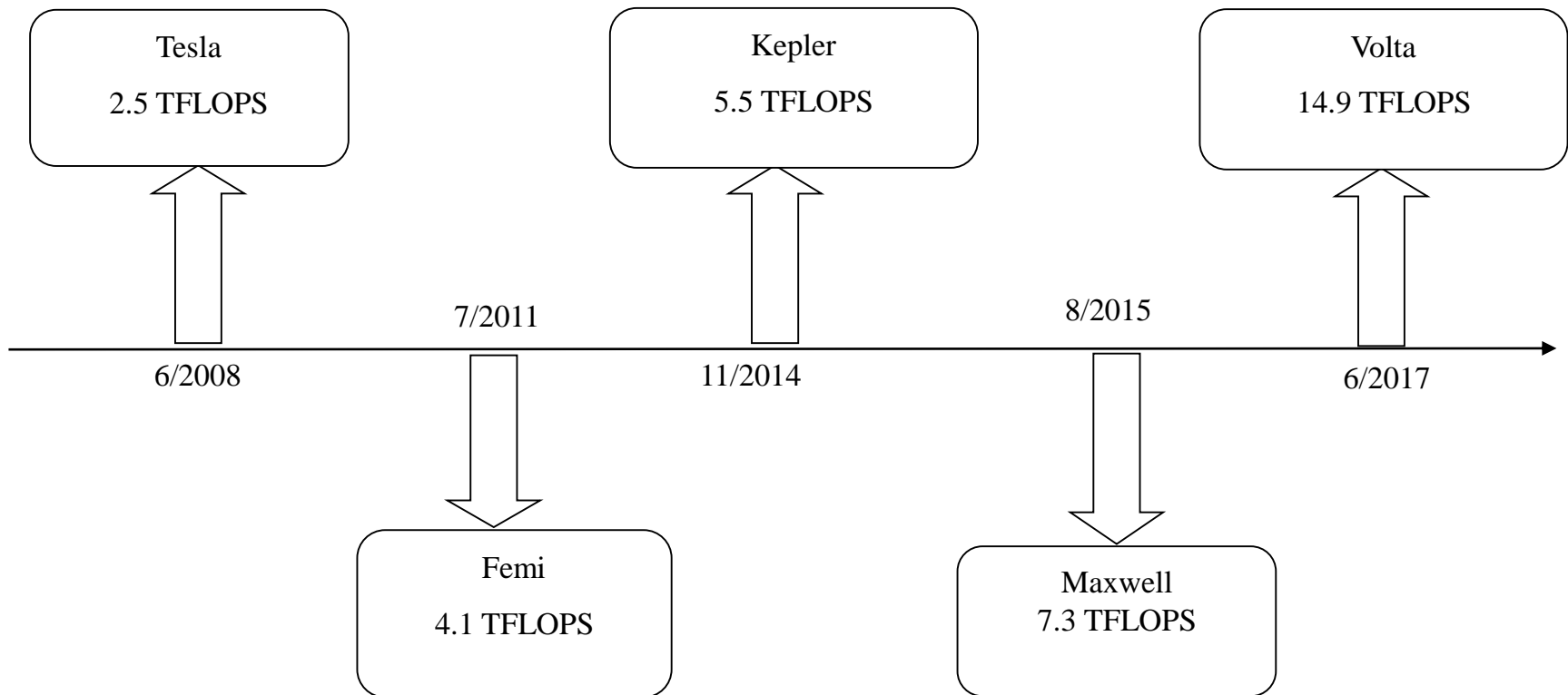


Ref: https://en.wikipedia.org/wiki/List_of_Nvidia_graphics_processing_units





- ◆ **Workstation History: Increase ~50% each year**
- ◆ Prices 2017 (Full DGX): ~1000\$/TFLOPS; N/A\$/TOPS
- ◆ Prices 2018 (T4 support INT4, Dell R740+4Cards = 13200\$): ~400\$/TFLOPS; ~**12\$/TOPS**



[Ref: \[https://en.wikipedia.org/wiki/List_of_Nvidia_graphics_processing_units\]\(https://en.wikipedia.org/wiki/List_of_Nvidia_graphics_processing_units\)](https://en.wikipedia.org/wiki/List_of_Nvidia_graphics_processing_units)

DGX-1: 8 cards => 129k\$;

Only T4 supporting with INT4

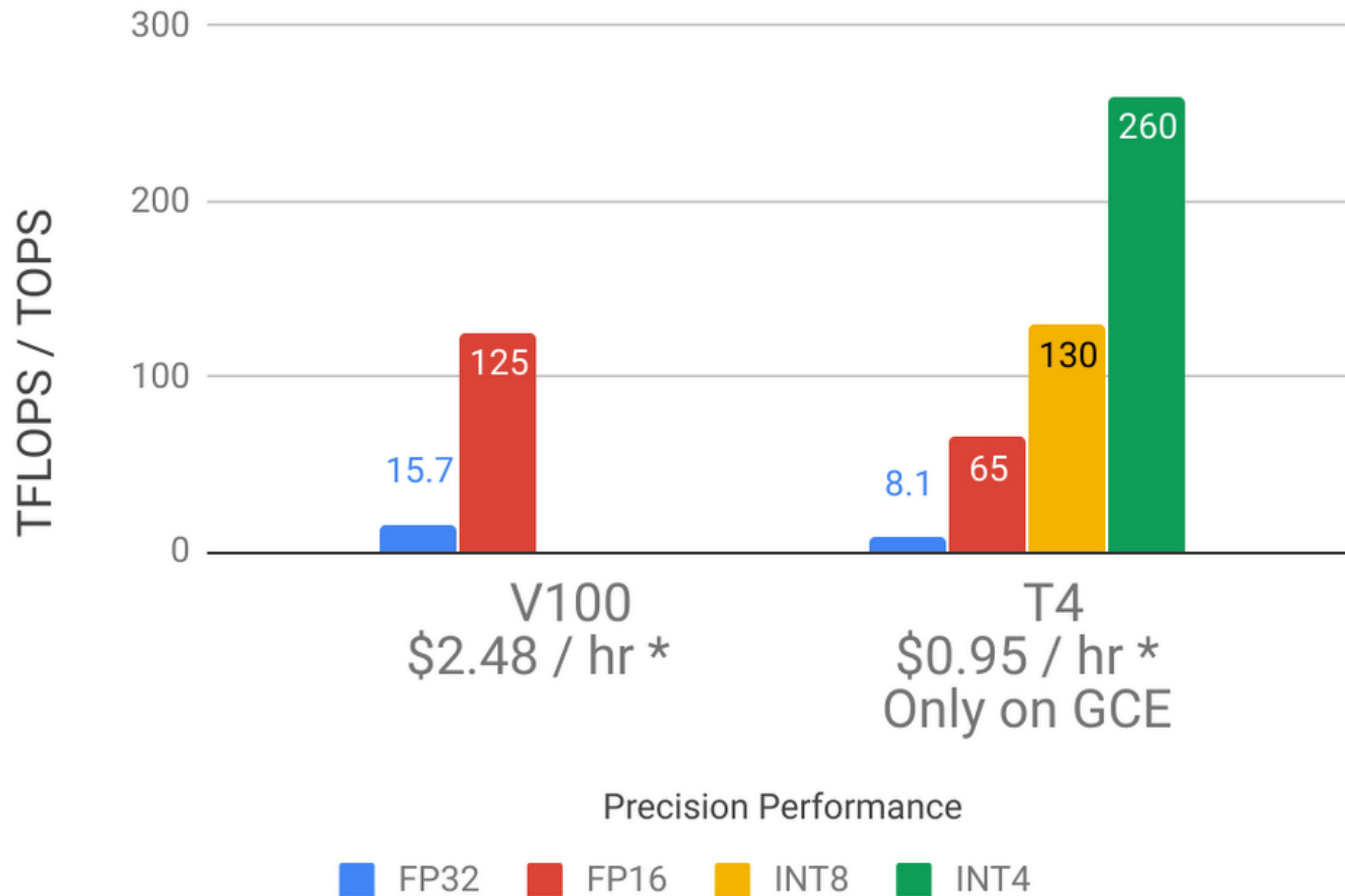




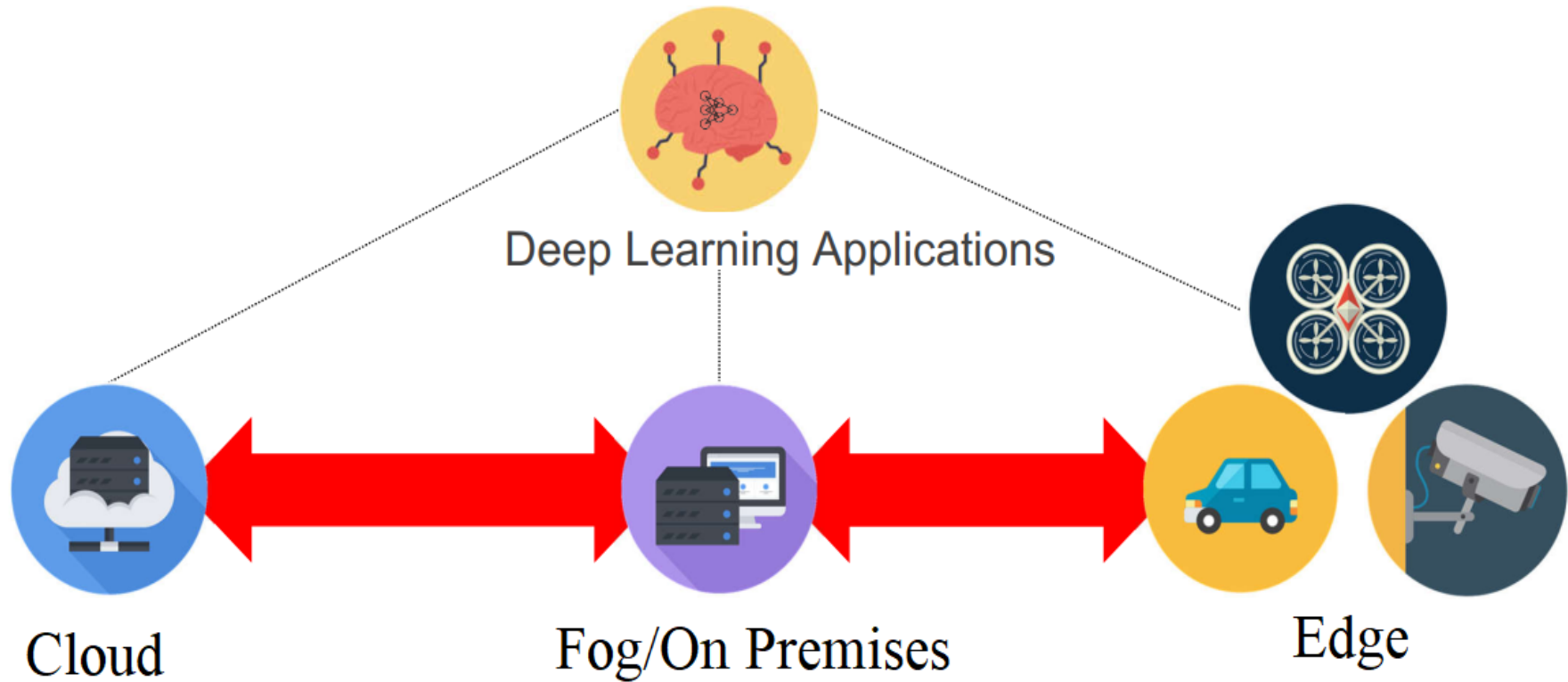
◆ Processing with type precision depend on **DEVICE**:

◆ V100 and other Nvidia not support for INT8/INT4

NVIDIA T4 Multi Precision Compute



2. Deployment solution (1)



Deployment solution (2)

◆ FOG/EDGE solving: Delay/Bandwidth/Responding

CLOUD LAYER

Big Data Processing
Business Logic
Data Warehousing

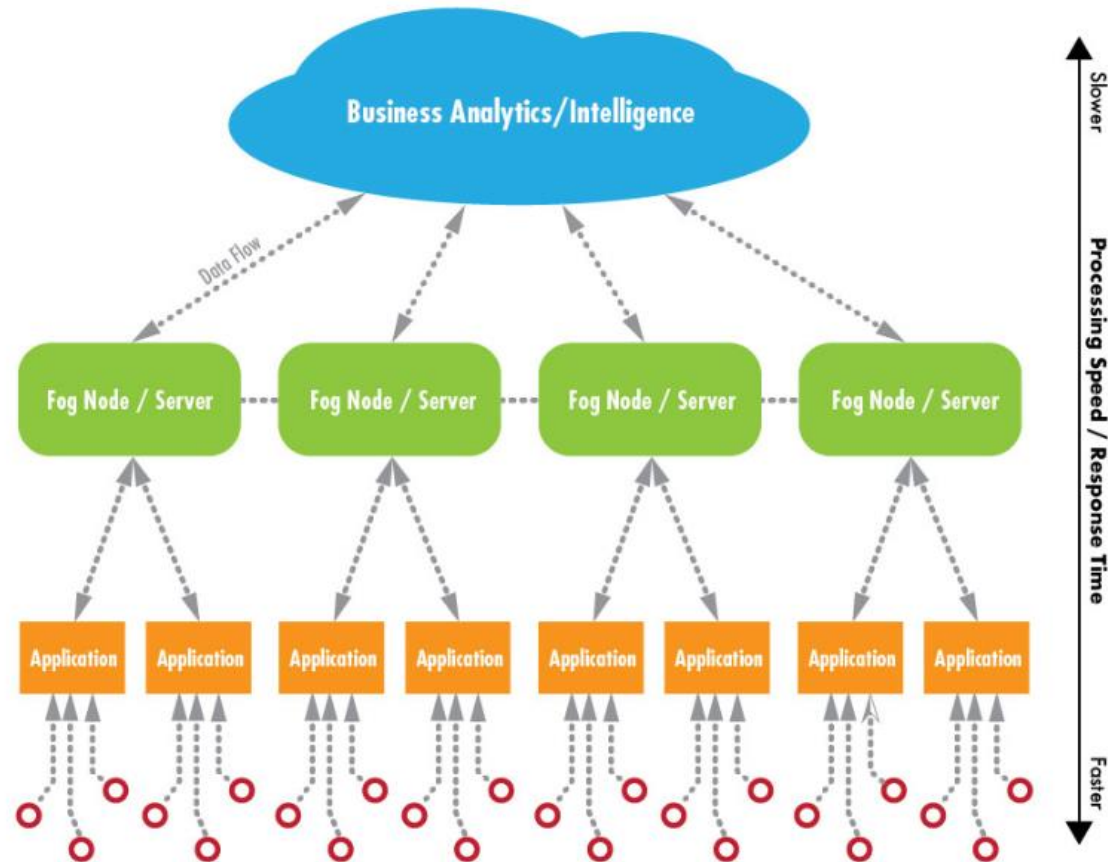
FOG LAYER

Local Network
Data Analysis & Reduction
Control Response
Virtualization/Standardization

EDGE LAYER

Large Volume Real-time Data Processing
At Source/On Premises Data Visualization
Industrial PCs
Embedded Systems
Gateways
Micro Data Storage

Sensors & Controllers (data origination)



Hardware & SDKs

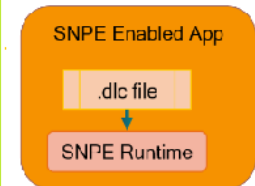
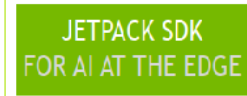
Models



Framework



Tools



HW Platforms



Xilinx



GeForce® RTX 2080 Ti 11GB

DGX/Tesla

Nvidia

Intel....



ZCU102

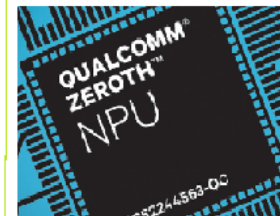


ZCU104

Xilinx



Nvidia



Qualcomm

Intel..

Cloud

Edge

Tên slide.....

Deploy Algorithm with Any Devices?



◆ Problem in Cloud/Fog:

- ◆ Framework: Tensorflow, Keras, Pytorch, Mxnet, Caffe

◆ Problem in Edge:

- ◆ SDK in edge: JetsonSDK, DNNDK, OpenVINO, SNPE....
- ◆ **Not support** User-defined Neural Network layers

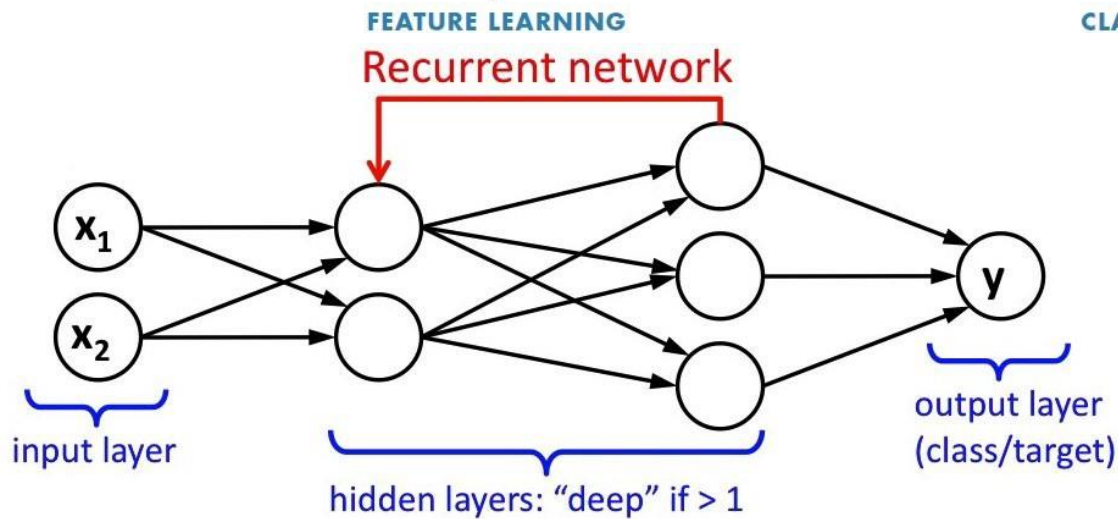
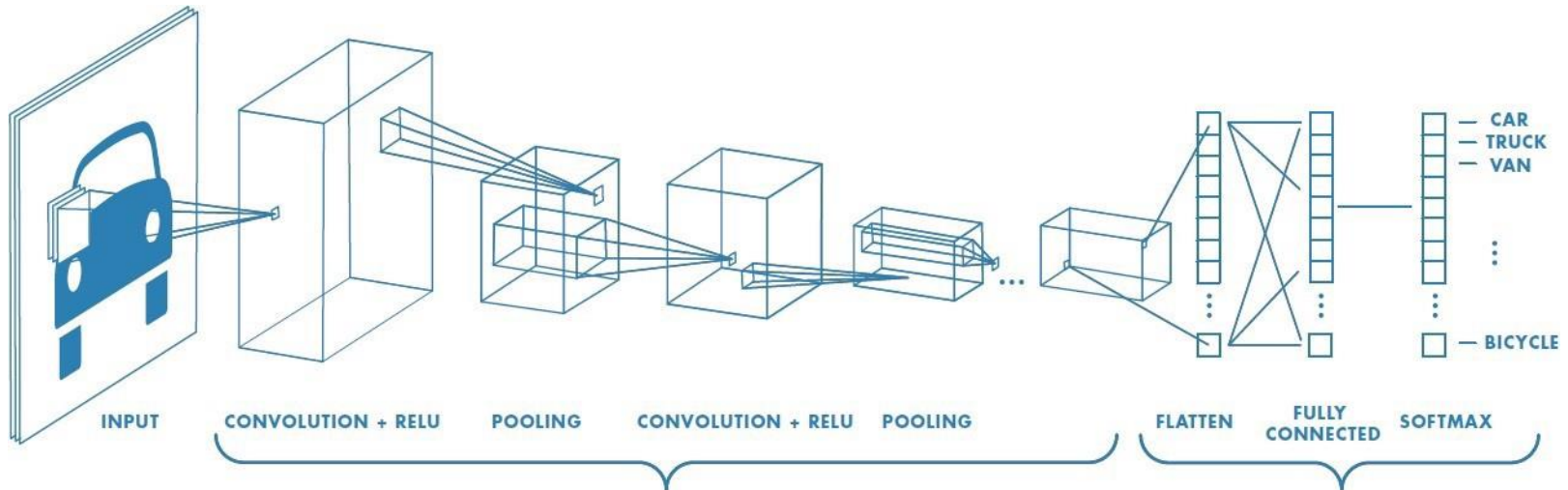


3. Complexity of Application AI



◆ Computing in AI: CNN, RNN...

=> 90% time-processing of Application involve NN



Tên slide.....

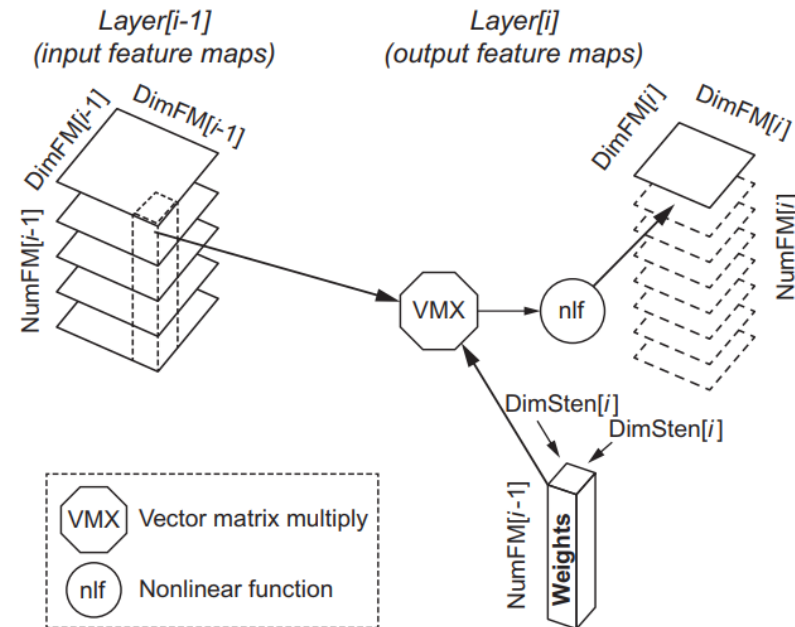


Complexcity of Algorithm AI



$$\text{OPS/Layer} = 2 * \text{MAC} = 2 \times \text{DimFM}[i]^2 \times \text{NumFM}[i] \times \text{NumFM}[i-1] \times \text{DimSten}[i]^2$$

- Number of weights per output Feature Map: $\text{NumFM}[i-1] \times \text{DimSten}[i]^2$
- Total number of weights per layer: $\text{NumFM}[i] \times \text{Number of weights per output Feature Map}$
- Number of operations per output Feature Map: $2 \times \text{DimFM}[i]^2 \times \text{Number of weights per output Feature Map}$
- Total number of operations per layer: $\text{NumFM}[i] \times \text{Number of operations per output Feature Map} = 2 \times \text{DimFM}[i]^2 \times \text{NumFM}[i] \times \text{Number of weights per output Feature Map} = 2 \times \text{DimFM}[i]^2 \times \text{Total number of weights per layer}$
- Operations/Weight: $2 \times \text{DimFM}[i]^2$



Complexity of Algorithm

- Layer 0:

$$2 * (416 * 416) * [32 * (3 * 3 * 3)]$$

- Layer 1:

$$2 * (208 * 208) * [64 * (3 * 3 * 32)]$$

- Layer 105:

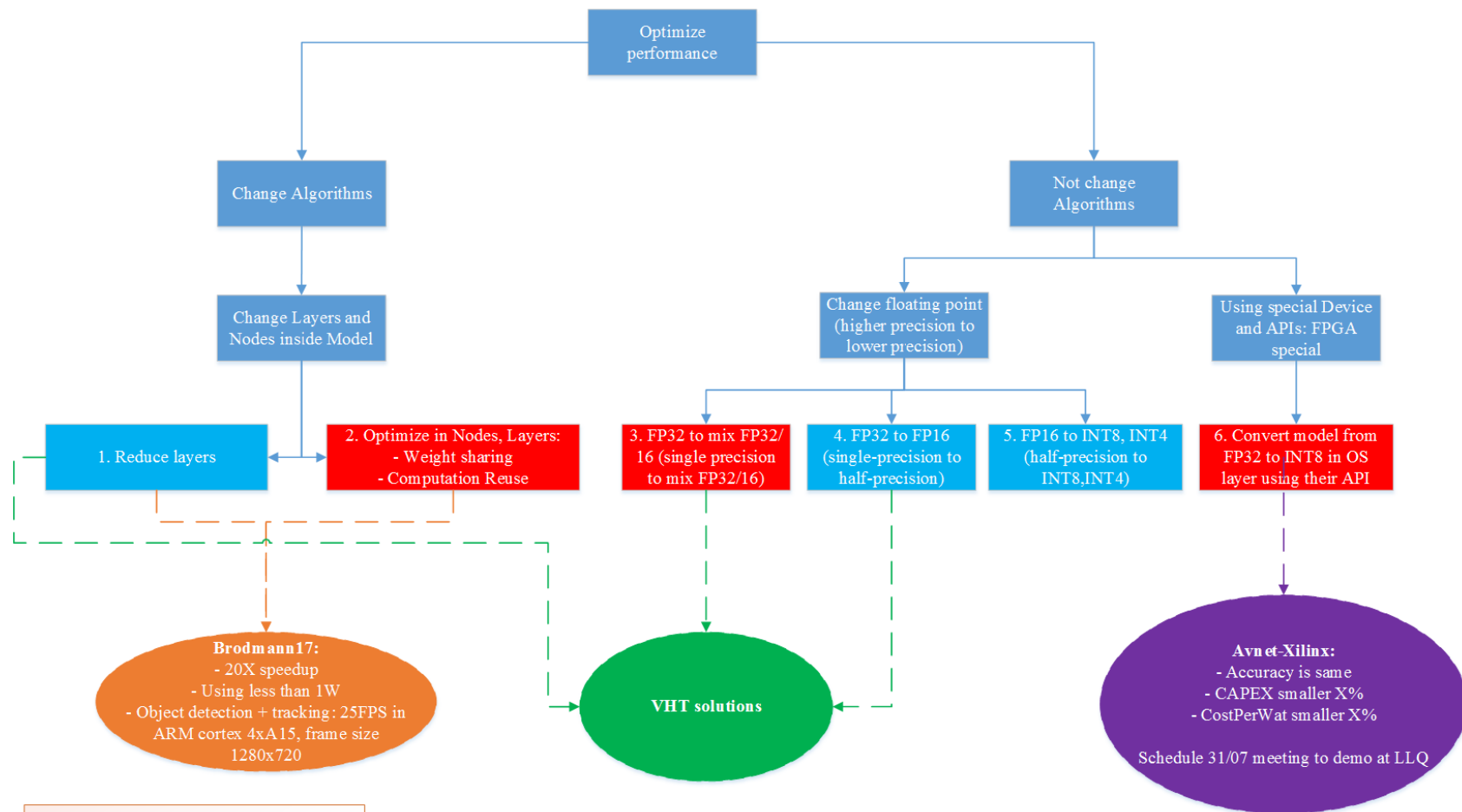
$$2 * (52 * 52) * [255 * (1 * 1 * 256)]$$

layer	filters	size	input	output	
0 conv	32	3 x 3	416 x 416 x 3	416 x 416 x 32	0.299 BFLOPs
1 conv	64	3 x 3 / 2	416 x 416 x 32	208 x 208 x 64	1.595 BFLOPs

105 conv	255	1 x 1 / 1	52 x 52 x 256	52 x 52 x 255	0.353 BFLOPs
106	detection				
truth_thresh: Using default '1.000000'					
Loading weights from yolov3.weights...Done!					
data/dog.jpg: Predicted in 0.029329 seconds.					
dog: 99%					
truck: 93%					
bicycle: 99%					

How to reduce cost

- ◆ 1. Market reduce prices of TOPS
- ◆ 2. Optimize Algorithm



Ghi chú:

Maintaining the accuracy

Affect the accuracy

Challenges: **MAC depend on Size-NN, so**

◆ 1. FP Precision to utilize HW new architecture

◆ Framework support change precision:

◆ Support to change to FP16, INT8, INT4?

◆ If not:

◆ Build from zero, not using pretrain: Data?

◆ 2. Algorithm to reduce MAC

◆ Reduce size filter, reduce layer => Has DONE, but **always Accuracy DOWN**

Model	Layers	FLOPS (B)	FPS	mAP	Dataset
YOLOv1	26	not reported	45	63.4	VOC
YOLOv1-Tiny	9	not reported	155	52.7	VOC
YOLOv2	32	62.94	40	48.1	COCO
YOLOv2-Tiny	16	5.41	244	23.7	COCO
YOLOv3	106	140.69	20	57.9	COCO
YOLOv3-Tiny	24	5.56	220	33.1	COCO

◆ KEY: Innovation about **Architect NN&DATA** to get SOTA in optimizing

4. Benchmark result



STT	Algorithm	Complexity	Layers	Performance (FPS)								Note
				2080 Ti	V100	U250	ZCU 104	Jetson Nano	Coral Edge TPU	Pi 4	Hailo	
Detection	Yolov3 416x416	140.69 GFLOPS	106	60	80	N/A	29	2	-	-	-	Darknet-53
	Yolov3-tiny	5.56 GFLOPS	24	240	320	N/A	N/A	25	-	-	-	x4 times more with yolov3
	Yolov2-voc 608x608	62.94 GFLOPS	30	120	160	125	58	4	-	-	-	Darknet-19 https://arxiv.org/pdf/1811.05588.pdf
	SSD Mobilenet v2	13.2 GFLOPS					116	39	N/A	10	-	
Classification	VGG16 – FP32	N/A	16	169	233	476	73	N/A	-	-	-	
	VGG19 - 224x224	N/A	19	N/A		N/A	N/A	10	3.2	-	-	
	ResNet-50 – FP32	15.5 GFLOPS	50	286	368	1200	124	36	-	-	672 (INT8)	
	Arcface	24G FLOPs	100	50				N/A	-	-	-	iccv19-challenge: ResNet100FC-IR): 24G FLOPs
Pose Estimation	OpenPose	100 GFLOPS		22			37.2	10 (Mobile Net)	-	-	-	1280x960 MobileNet Thin I7-8700K: 25 2080Ti: 52.7 FPS
	AlphaPose	N/A		23				N/A	-	-	-	
Action recognition	I3D-TSN	16.3 BFLOPS		20				N/A	-	-	-	Backbone: BN-Inception 11.3M
	LSTM	8.42 MFLOPS						N/A	-	-	-	0.53M paprams

U250~2080Ti; U250 = 2 x ZCU 104



CAPEX For Production



	VHT		Partner working with IDC		Avnet-Xilinx ZCU	Brodman17	Smarty	Note
	Current	Target 1 year (10x)	Futurix	XRVision				
vSmartGate	40.000.000	4.000.000	N/A	N/A	N/A	N/A	160.000.000	2 camera/1 cards
vSmartSafe	80.000.000	8.000.000	N/A	N/A	N/A	N/A	N/A	1 camera/1 cards
vSmartAnalytic	20.000.000	2.000.000	330\$	250\$	450\$	N/A	N/A	4 camera/1 cards

Tên slide.....



5. Conclusion



- ◆ 1. Thử nghiệm deploy hệ thống trên 02 boards **Jetson nano (Nvidia)** and **i.MX8M MINI SOM** (Gyr Falcon) để đánh giá hiệu năng và tính khả thi triển khai.
- ◆ 2. Xây dựng thiết kế hệ thống dựa trên 02 loại chip AI là **Lightspeeur** (Gyr Falcon) và **Myriad** (Intel) để so sánh hiệu năng và giá.





Say it your way

Thank you



OPENVINO™ TOOLKIT

WRITE ONCE + SCALE TO DIVERSE, OPTIMIZED ACCELERATORS + LEVERAGE COMMON ALGORITHMS

nGraph™ mxnet
Caffe TensorFlow

API SOLUTION

Model Optimizer &
Inference Engine

CPU GPU FPGA VPU*

Neural Network Acceleration (DL)

Broad set of frameworks and topologies
Deep Learning Deployment Toolkit
Optimized Intel Algorithms

<https://software.intel.com/en-us/computer-vision-sdk>

VPU = Vision Processing Unit (Movidius)

1 = coming 2H 2018. Roadmap Notice: All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest Intel product specs and roadmaps.

*Other names and brands may be claimed as the property of others.



Open CV
API SOLUTION

CV Library
(Kernel and Graph APIs)

CPU GPU VPU¹

Classical Computer Vision (CV)

OpenCV* - Broadly used library, Intel owned / maintained
OpenCL™ - direct coding acceleration



DIRECT CODING SOLUTION

Custom Code

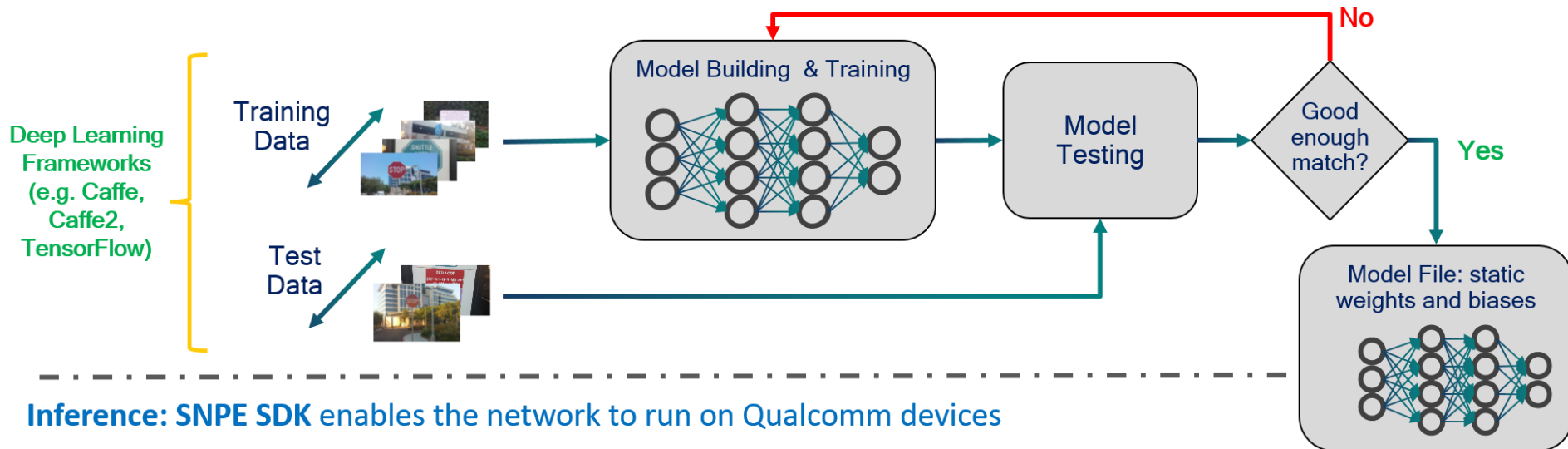
C/C++

Open CL

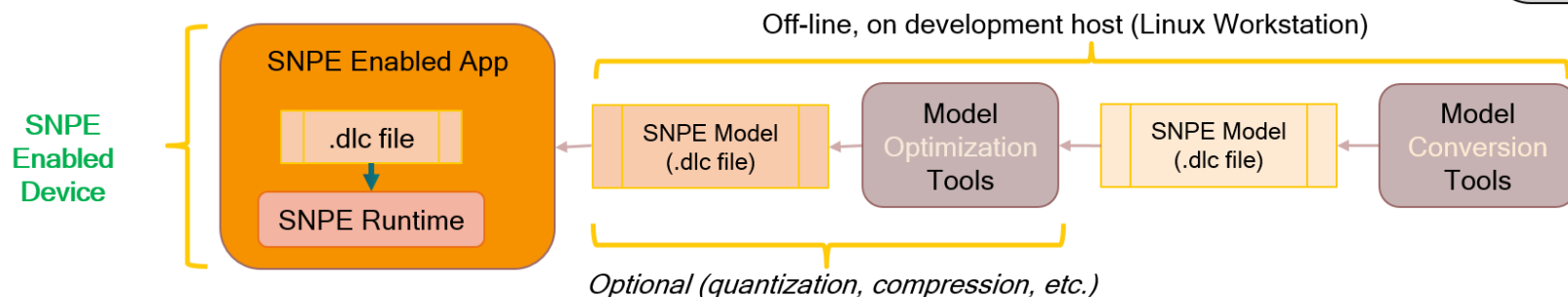
CPU GPU FPGA¹ VPU¹



Training: Machine Learning experts build and train their network to solve their particular problem



Inference: SNPE SDK enables the network to run on Qualcomm devices



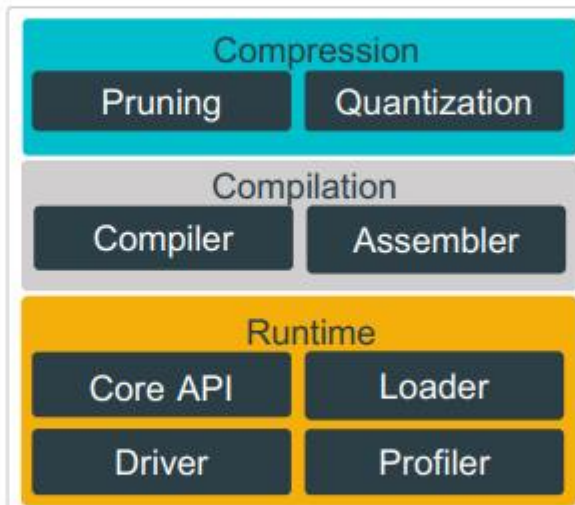
Models



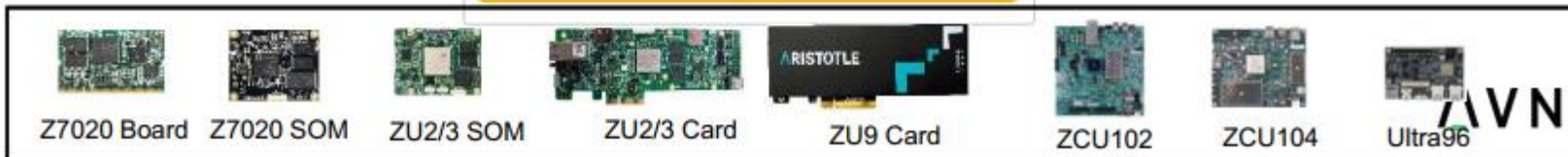
Framework



Tools



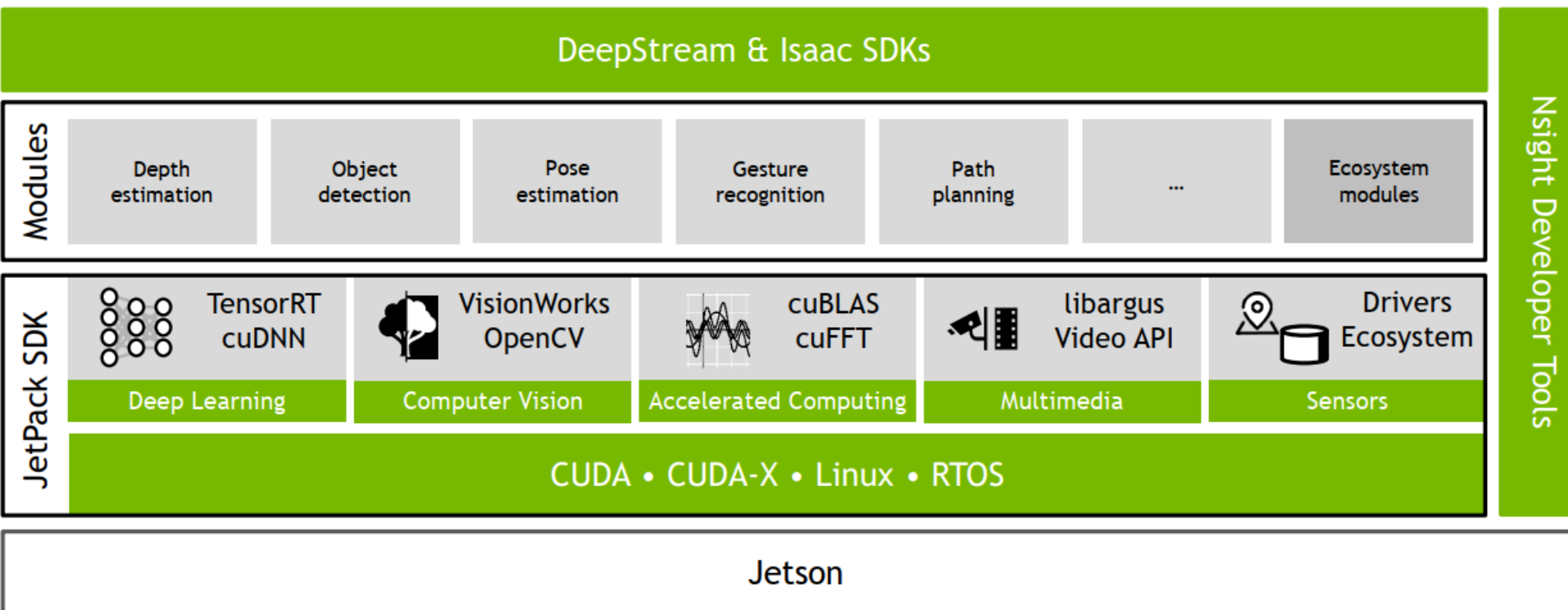
ML HW Platforms



>> 10



JETSON SOFTWARE



Night Developer Tools

