Accelerating Large-Scale Video Surveillance for Smart Cities with TensorRT

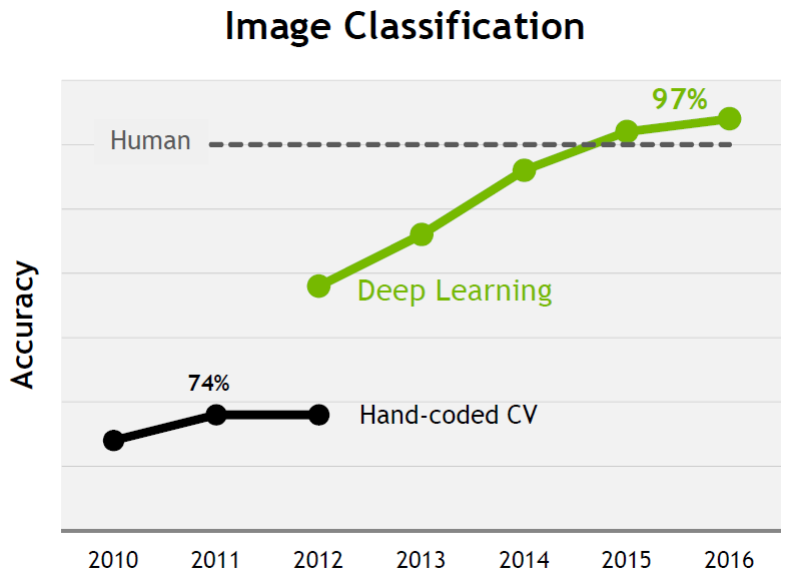
Shounan An, Seungji Yang, Hyungjoon Cho

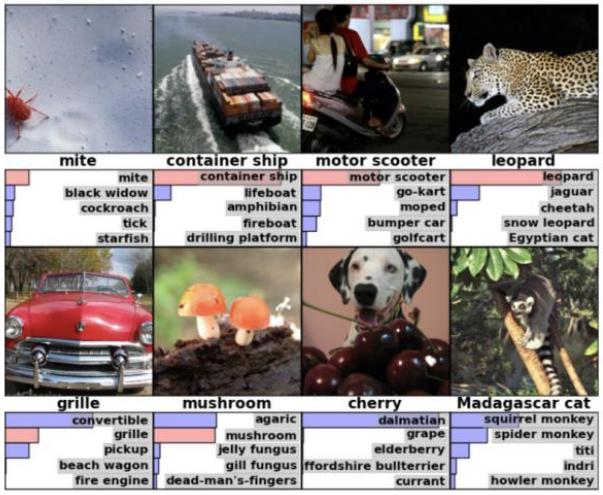
March 2018

Table of contents

Intro

Current deep learning for video surveillance is perfect?





Ready for video surveillance?

DL is truly a technological enabler, but need to be more developed for surveillance.

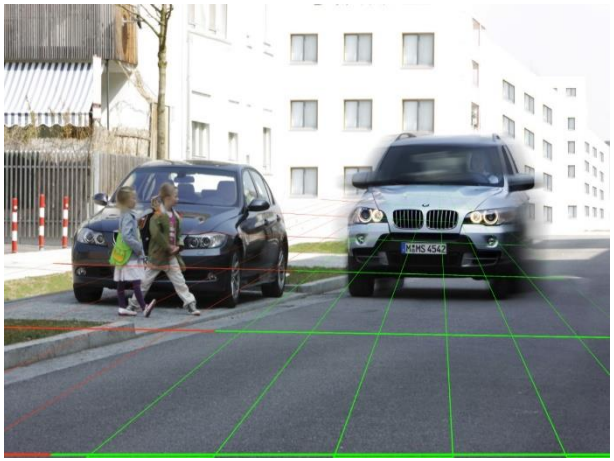
Intro

AI is still in its infancy in computer vision industry applications

Persons of interest?

Different visual aspects according to applications

CCTV



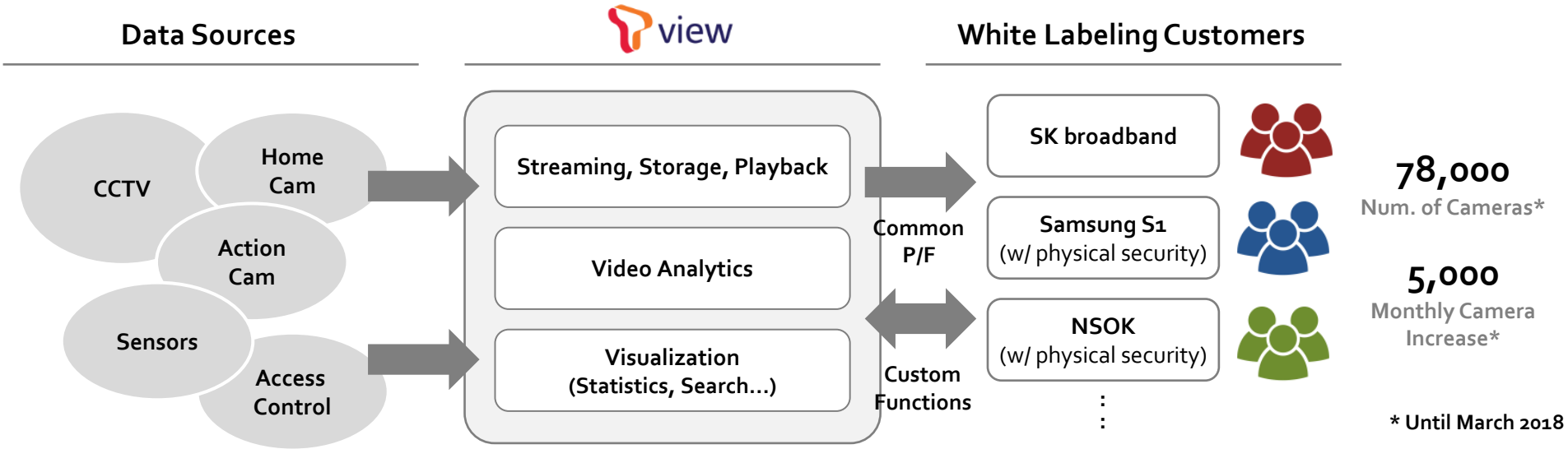
Automotive

Photo album

Home

Towards large-scale inference engine for T view

T view is a Video Surveillance as a Service (VSaaS)

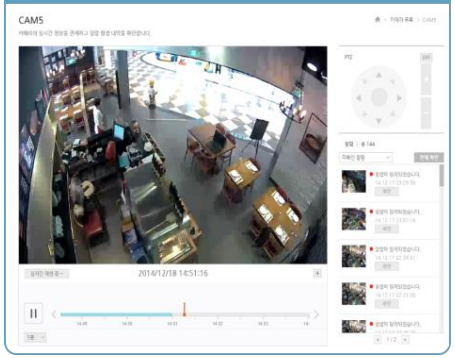


Secure & Safe Cloud Platform

Intelligent Video Surveillance

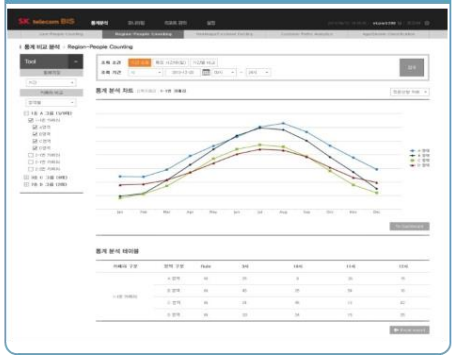
Analytics for Business Intelligence

Real-time monitoring



Intrusion detection & People Counting

Statistics with POS data



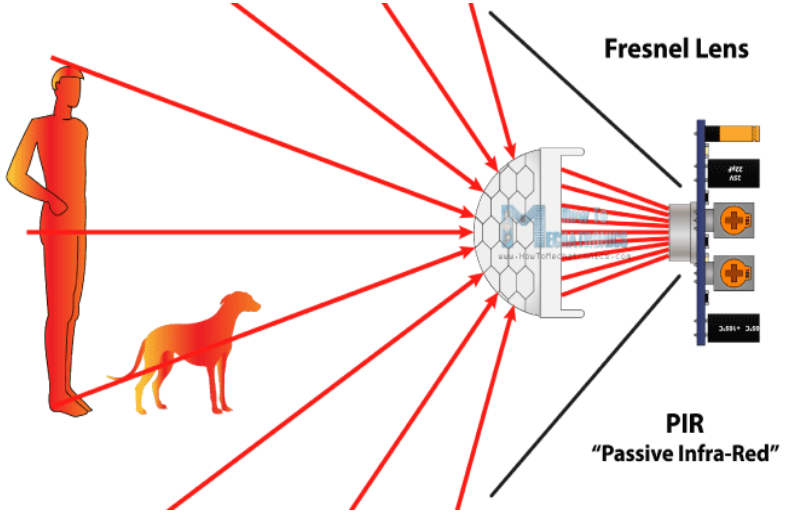
Dashboard

Problem

Conventional intrusion detection for physical security service

Real intrusion event is very rare!

Sensor Alarm (PIR sensor) → ***Physical Security Service*** (Dispatch security officers) → ***High False-alarm (over 99%)***



NSOK

900 alarms / day

1 true positive / day

99% false-alarms

< 5 main cause for false-alarm >

1. *Indoor and outdoor temperature difference*
2. *Animal (dog, cat)*
3. *Air conditioner operation*
4. *Inappropriate direction*
5. *Direct sunlight or vehicle headlight*

Motivation

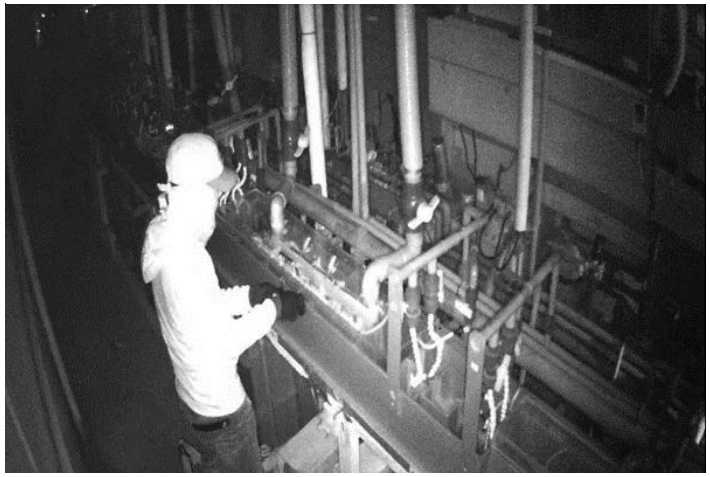
Deep video analytics for physical security service

Examples of real problem

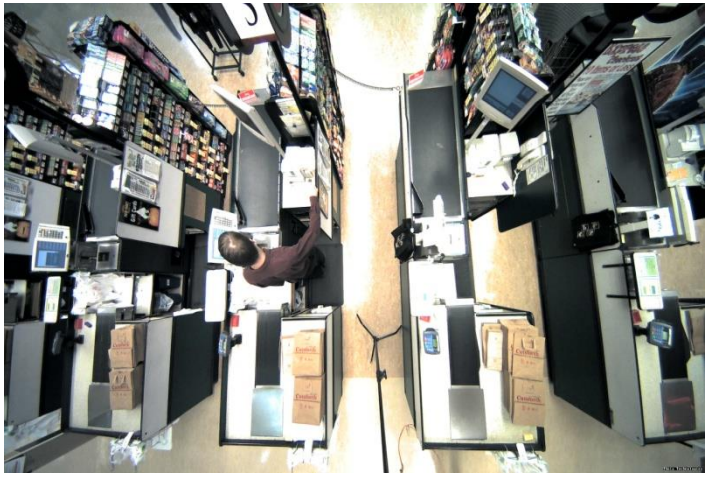
Challenge: accuracy

Challenges in accuracy… surely!

Night (illumination)



View Variation & Distortion

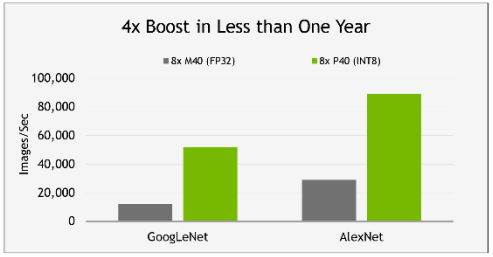


Weather

Long Distance & Low Quality

Challenge: performance

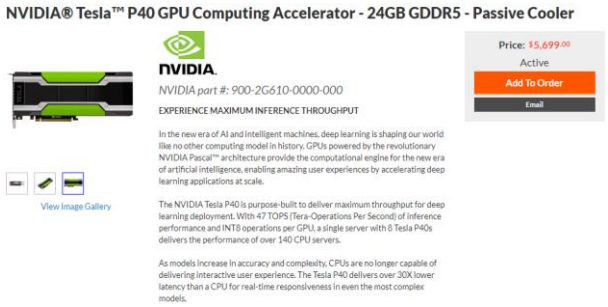
Challenges in cost $$$... rare interest of DL infra cost for service (inference)!



P40	
# of CUDA Cores	3840
Peak Single Precision	12 TeraFLOPS
Peak INT8	47 TOPS
Low Precision	4x 8-bit vector dot product with 32-bit accumulate
Video Engines	1x decode engine, 2x encode engines
GDDR5 Memory	24 GB @ 346 GB/s
Power	250W

GoogLeNet, AlexNet, batch size = 128, CPU: Dual Socket Intel E5-2697v4

6,000 USD

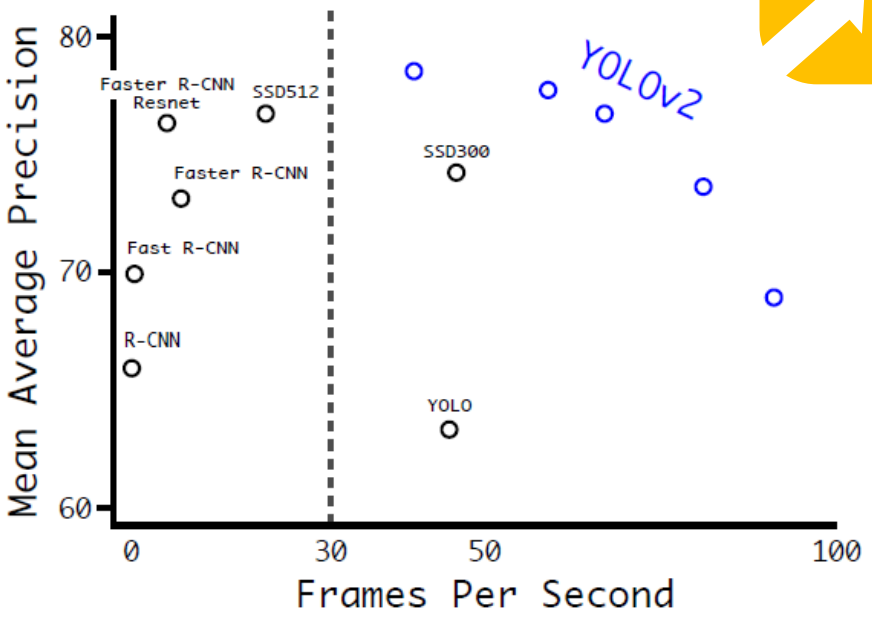


Inference* cost (time)

Objective

< Goal >
Higher accuracy in human detection &
Lower cost

for surveillance



R-CNN→ MultiBox → SPP-Net → DeepID-Net →NoC → **Fast R-CNN** → DeepBox → **MR-CNN** →
2013.11 ECCV '14 PAMI '16 ICCV '15 ICCV '15

Faster R-CNN →**YOLO** → AttentionNet → DenseBox → **SSD** →Inside-OutsideNet(ION) →G-CNN →
NIPS '15 CVPR '16 ICCV '15 ECCV '16 CVPR '16

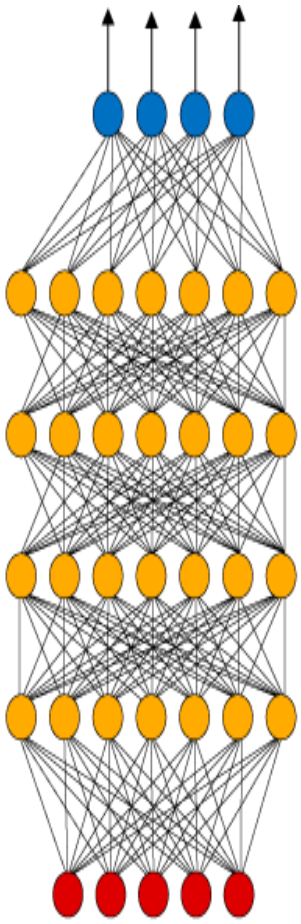
HyperNet → MultiPathNet → CRAFT → OHEM → **R-FCN** → **MS-CNN** → PVANET → GBDNet →
BMVC '16 CVPR '16 CVPR '16 NIPS '16 ECCV '16

StuffNet → Feature Pyramid Net **(FPN)** → **YOLO v2** → DSSD → CC-Net → Mask R-CNN …
CVPR '17

Challenge: accuracy

threshold 0.2 is applied for both networks

SIDNet: SKT Intrusion Detection Net



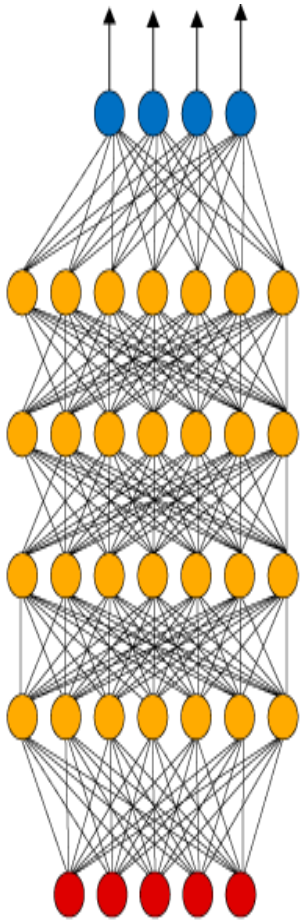
YOLO v2

Modify the # of layers and activation function

New anchor coordinates calculation

300 videos / 137,000 images / 0.7million labels

CCTV DB construction for training



SIDNet

Challenge: performance

Challenges in cost $$$... 30x more cost than the conventional motion analysis

【Conventional VA】

Motion-based Human Detection

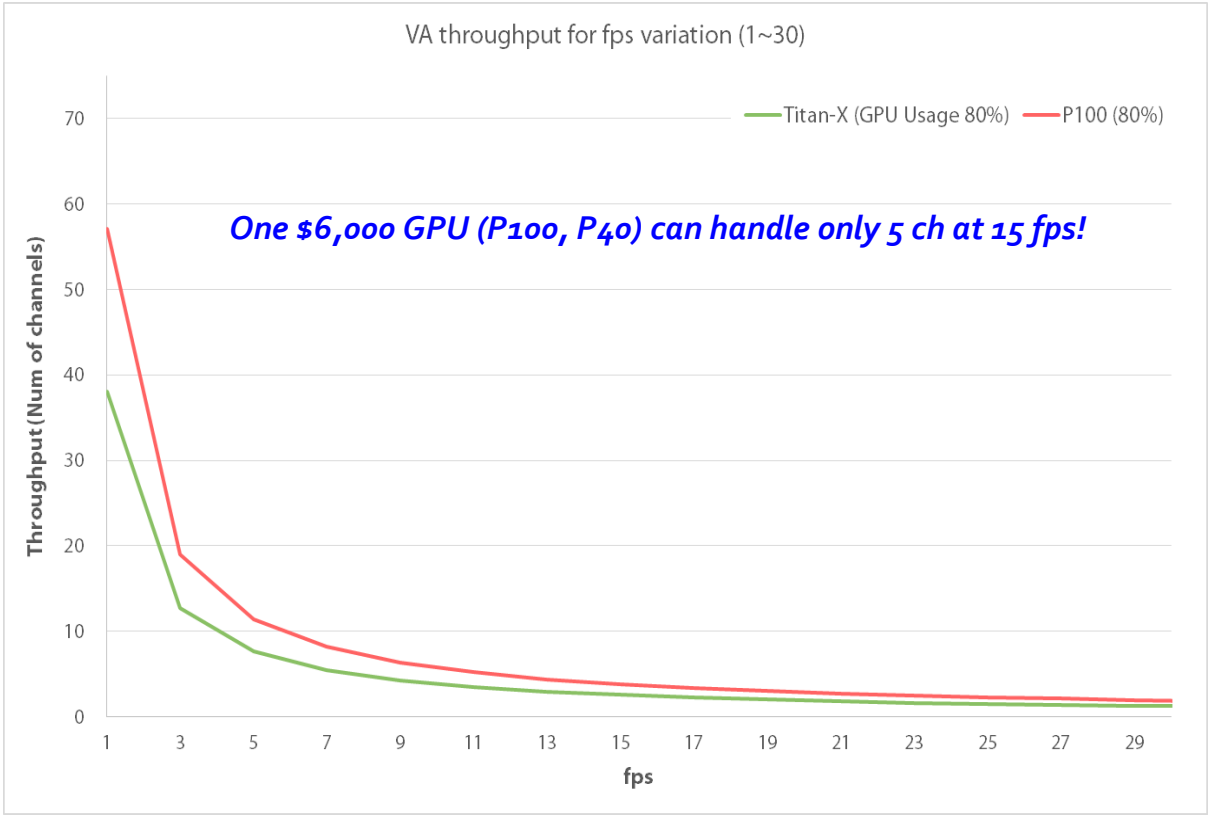
***2,129** fps (150 ch * 15 fps) @ 20 core CPU*
***3,750** fps (250 ch * 15 fps) @ K5000 GPU*

30x more cost!

【Current Deep VA】

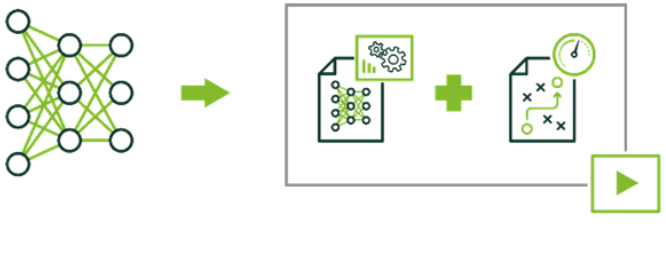
Deep Learning-based Human Detection

***67** fps @ TitanX-Maxwell GPU*
***85** fps @ P40 GPU*

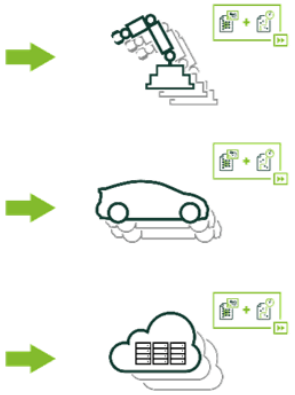


We need much faster inference engine for service!

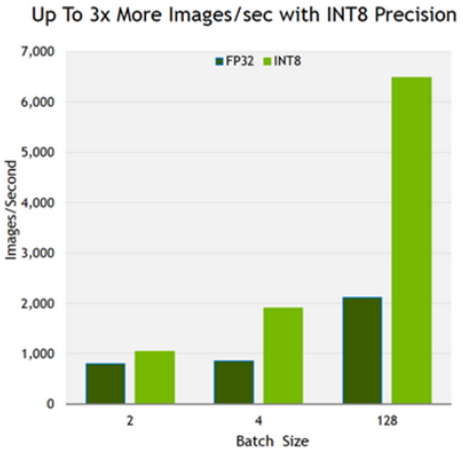
New opportunity for 3rd party… NVIDIA's TensorRT



Step 1: Optimize a trained neural network



Step 2: Perform real-time inference with GIE Runtime



GoogLenet, FP32 vs INT8 precision + TensorRT on Tesla P40 GPU, 2 Socket Haswell E5-2698 v3@2.3GHz with HT off

	FP32		INT8					
			Calibration using 5 batches		Calibration using 10 batches		Calibration using 50 batches	
NETWORK	Top1	Top5	Top1	Top5	Top1	Top5	Top1	Top5
Resnet-50	73.23%	91.18%	73.03%	91.15%	73.02%	91.06%	73.10%	91.06%
Resnet-101	74.39%	91.78%	74.52%	91.64%	74.38%	91.70%	74.40%	91.73%
Resnet-152	74.78%	91.82%	74.62%	91.82%	74.66%	91.82%	74.70%	91.78%
VGG-19	68.41%	88.78%	68.42%	88.69%	68.42%	88.67%	68.38%	88.70%
Googlenet	68.57%	88.83%	68.21%	88.67%	68.10%	88.58%	68.12%	88.64%
Alexnet	57.08%	80.06%	57.00%	79.98%	57.00%	79.98%	57.05%	80.06%
NETWORK	Top1	Top5	Diff Top1	Diff Top5	Diff Top1	Diff Top5	Diff Top1	Diff Top5
Resnet-50	73.23%	91.18%	0.20%	0.03%	0.22%	0.13%	0.13%	0.12%
Resnet-101	74.39%	91.78%	-0.13%	0.14%	0.01%	0.09%	-0.01%	0.06%
Resnet-152	74.78%	91.82%	0.15%	0.01%	0.11%	0.01%	0.08%	0.05%
VGG-19	68.41%	88.78%	-0.02%	0.09%	-0.01%	0.10%	0.03%	0.07%
Googlenet	68.57%	88.83%	0.36%	0.16%	0.46%	0.25%	0.45%	0.19%
Alexnet	57.08%	80.06%	0.08%	0.08%	0.08%	0.07%	0.03%	-0.01%

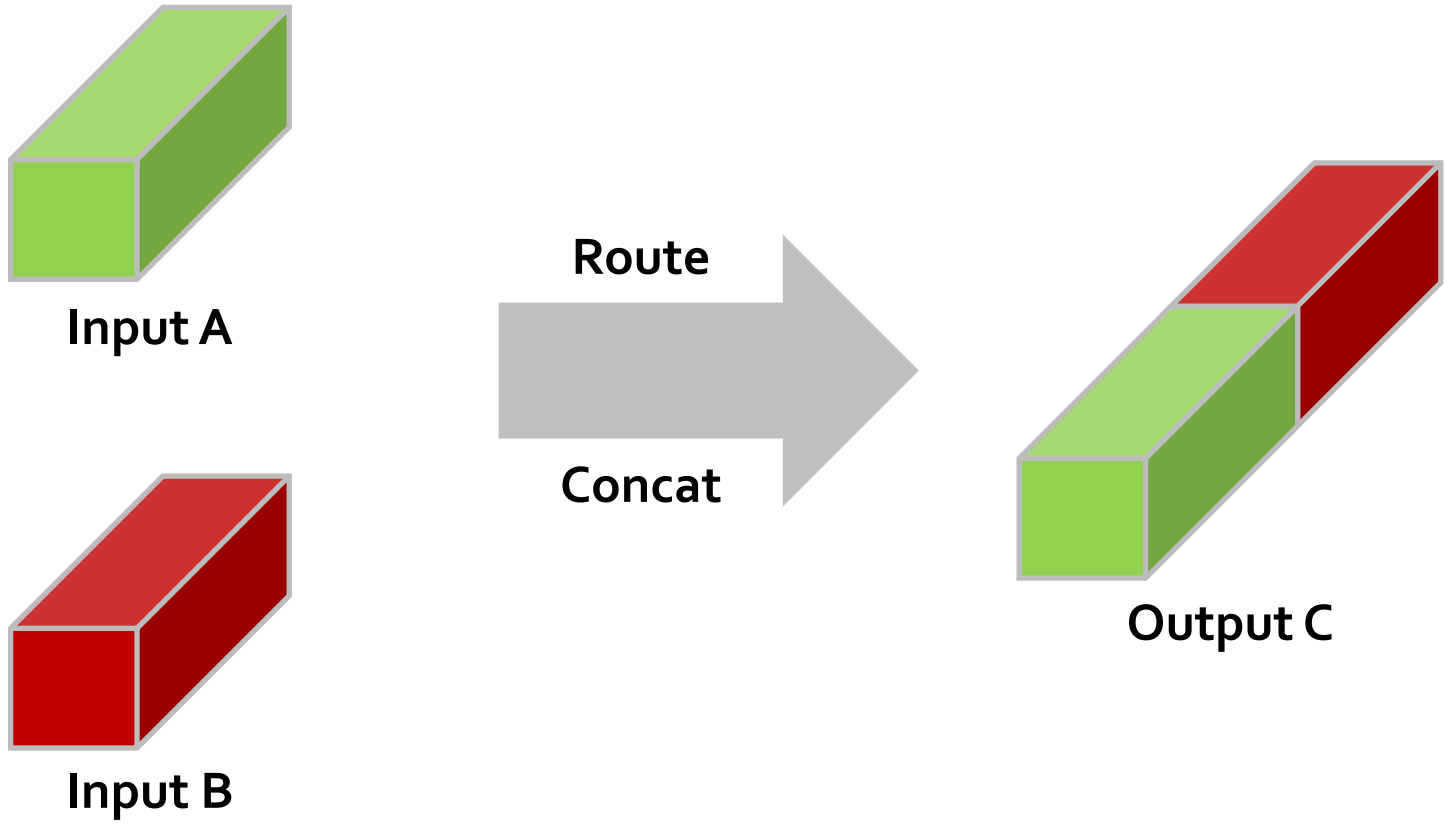
Proposed approach

Apply TensorRT to SIDNet

Layers	SIDNet	TensorRT	Remarks
Input	O	O	
Convolution	O	O	
Batch norm	O	O	
Leaky-RELU	RELU	X	▪ *Replace Leaky-RELU with RELU*
Max-pooling	O	O	
Route	O	X	▪ *Implement rout layer via concat layer* ▪ *No computation, no issue with INT8*
Reorg	O	X	▪ *CUDA implementation as custom plug-in layer* ▪ *No computation, no issue with INT8*
Region	O	X	▪ *CUDA implementation as custom plug-in layer*

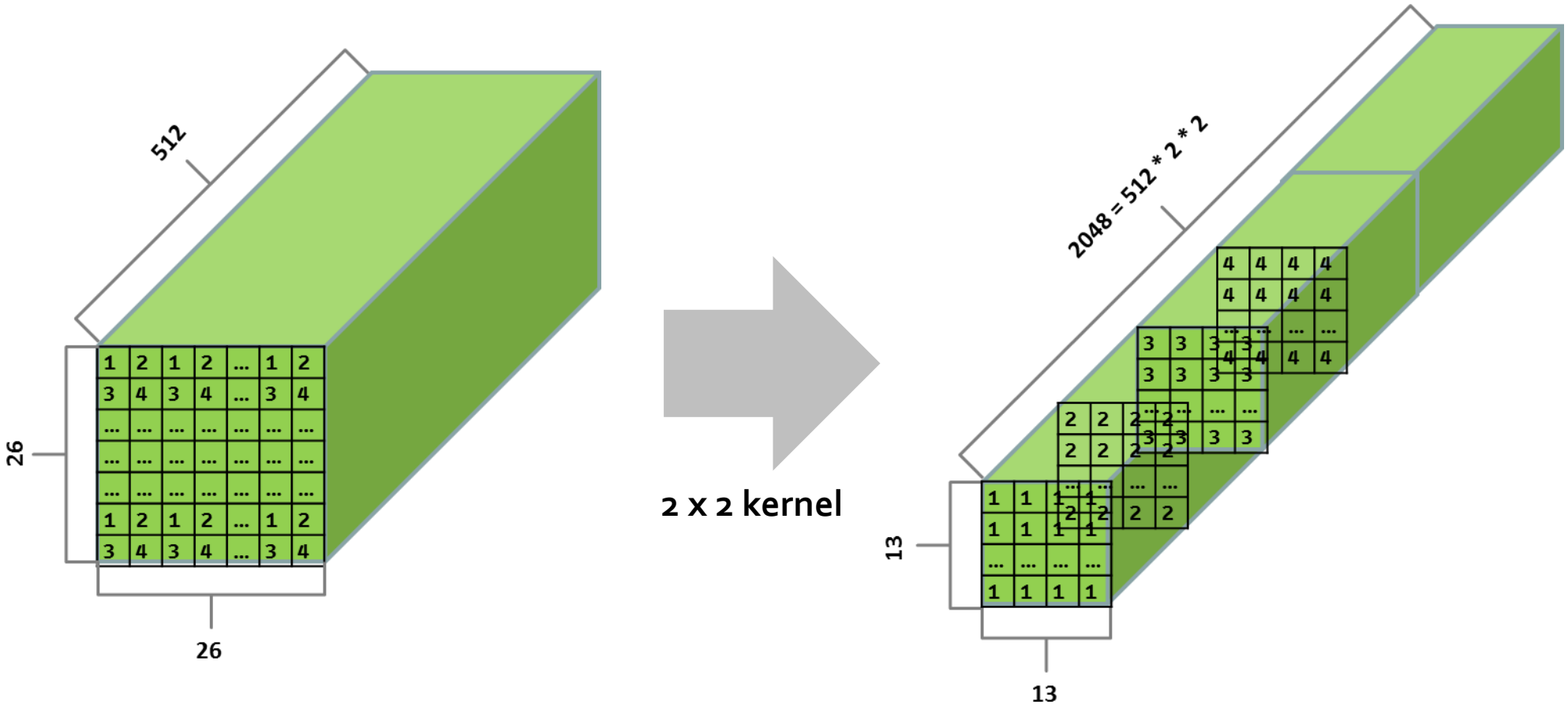
Proposed approach

Route layer *– equivalent with concat layer*



Proposed approach

Reorg layer *– CUDA implementation of Reorg layer as TensorRT's custom plug-in layer*

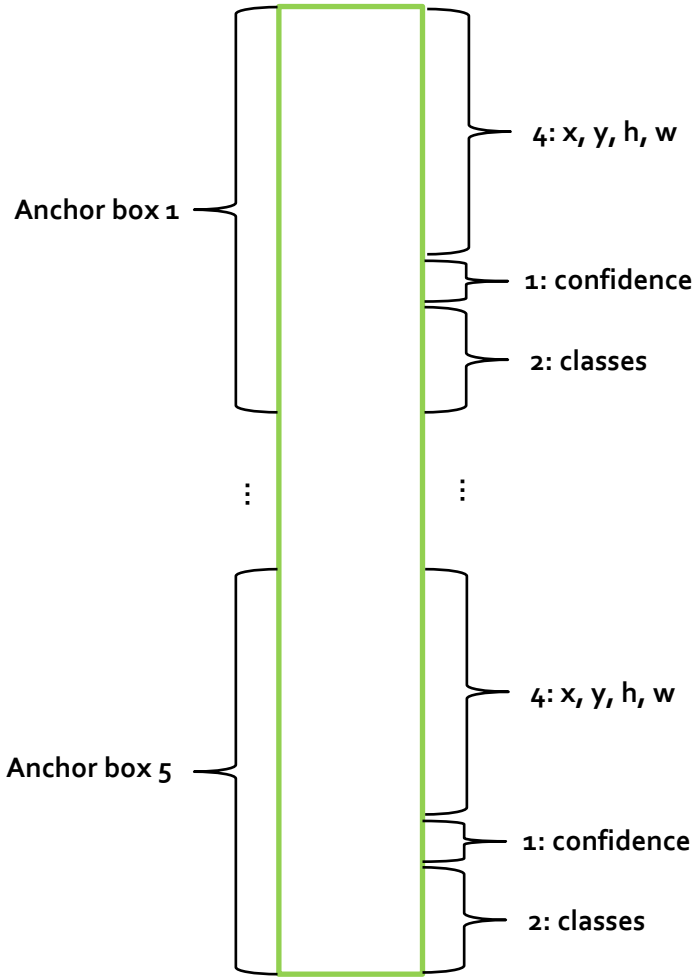


[Input]
N x 512 x 26 x 26

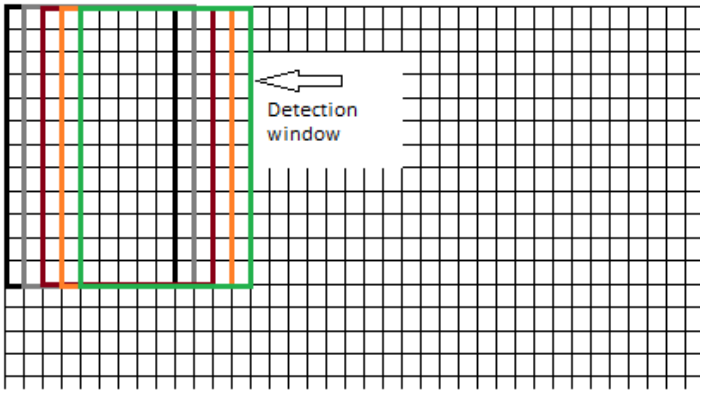
[Output]
N x 2048 x 13 x 13

Proposed approach

Region layer *– CUDA implementation of Region layer as TensorRT's custom plug-in layer*



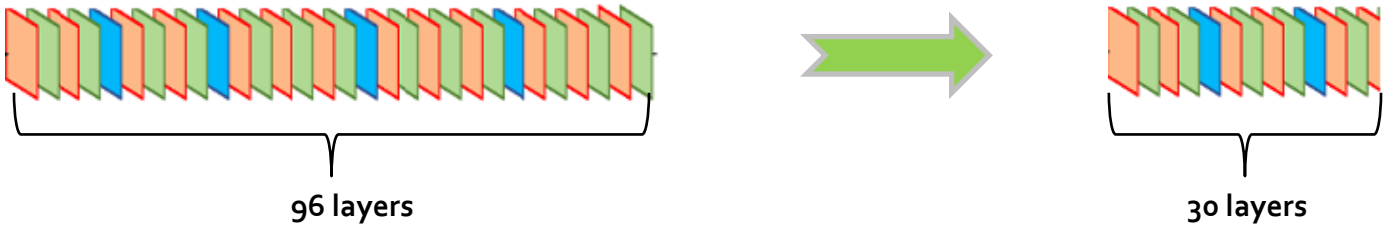
- ***Input: 13x13x35 feature map***
- ***13x13x5x(4+1+2) : [h, w, anchor, (x, y, h, w, confidence, class)]***
- ***Output: list of bbox, each bbox have (x, y, h, w, confidence)***
- ***Apply NMS to get final object detection result***



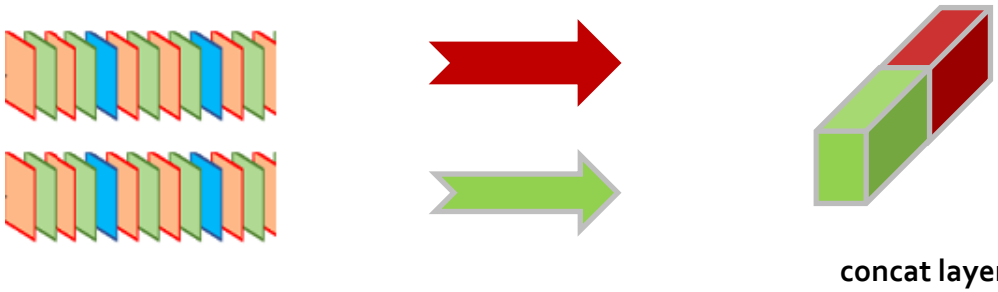
Proposed approach

*SIDNet@FP32 – **2x** faster with TensorRT*

- ***SIDNet has 96 layers, but after applying tensorRT only 30 layers remains***
- ***TensorRT merge conv+BN+scale+RELU 4 layers into just one layer***



- ***Efficiently use GPU memory to reduce unnecessary memcpy***



Proposed approach

SIDNet INT8 calibration process (Ref.: "8-bit inference with TensorRT")

- ***Calibration dataset: COCO 2014 validation 8,000 images(about 20% of validation set)***
- ***Batch file generation: 1000 batches with 8 images/batch***
- ***Apply exactly the same pre-processing as training steps***

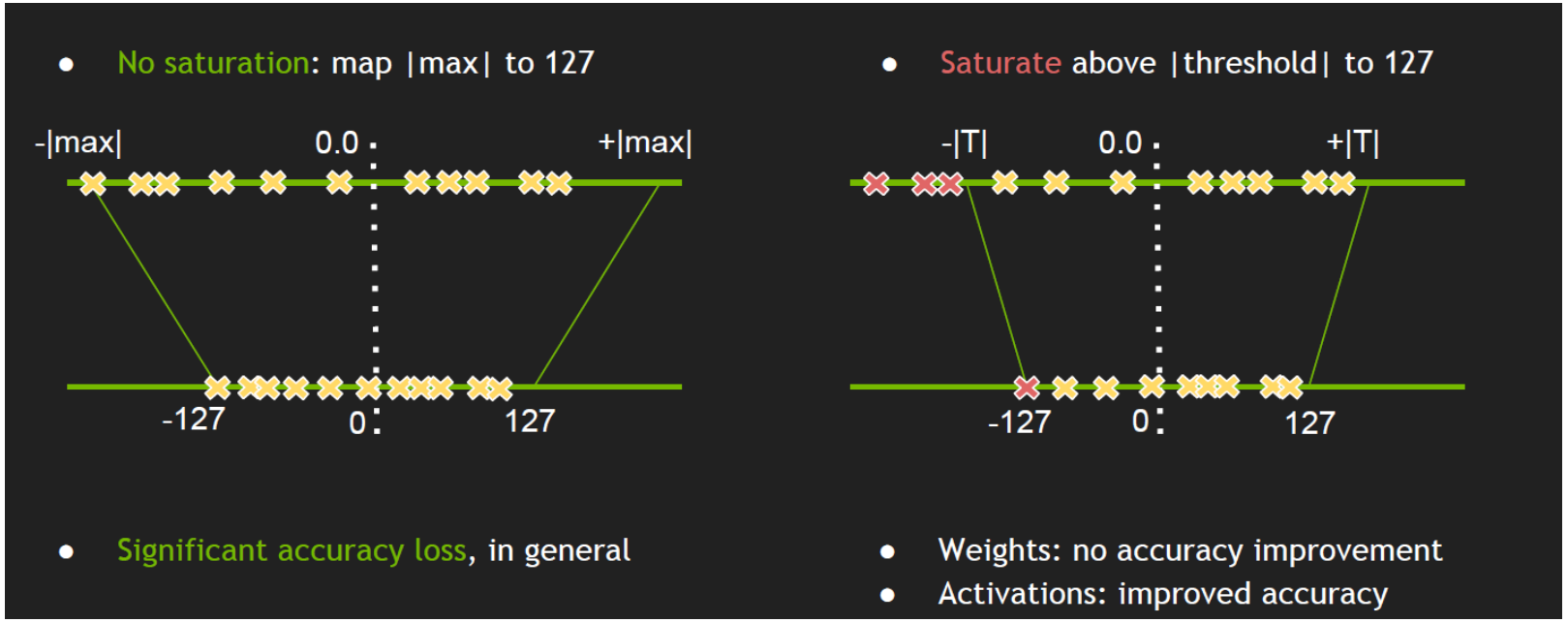


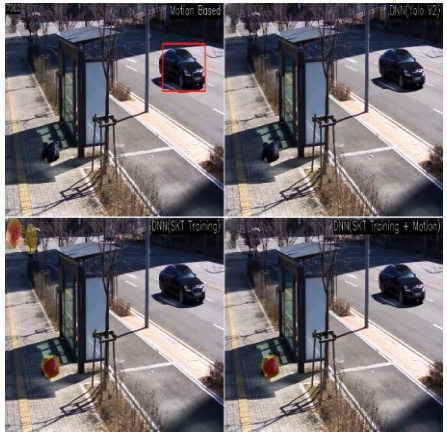
Figure from "8-bit inference with TensorRT", GTC 17'

KISA & T view datasets

Day

Night

	KISA	T view
#videos	100	100



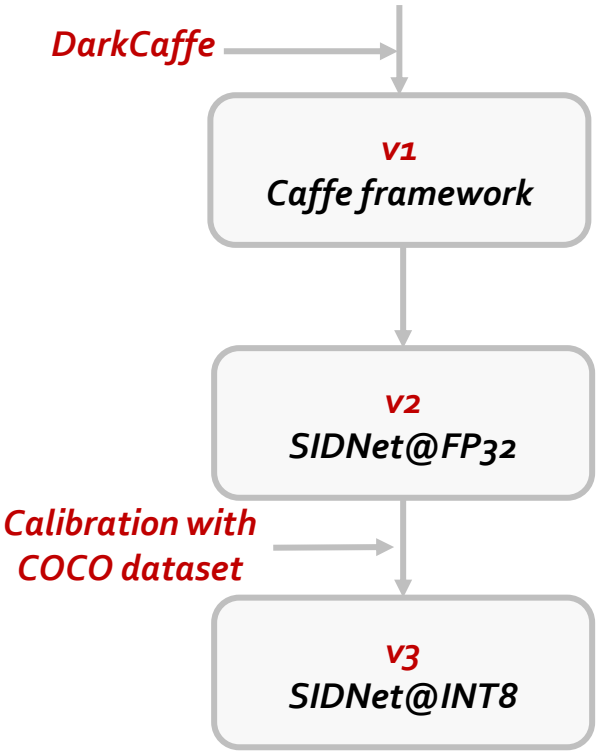
Outdoor

Indoor

Extremely small object

Pose variance

Results: accuracy



Model	Precision(KISA)	Recall (KISA)	Precision (T view)	Recall (T view)
Reference: YOLO v2	72	80.2	79.5	80.2
v1 Caffe framework	72	80.2	79.5	80.2
v2 SIDNet@FP32	74.2	89.4	84.2	75.1
v3 SIDNet@INT8	73.2	91.4	83.8	74.5

Results: performance

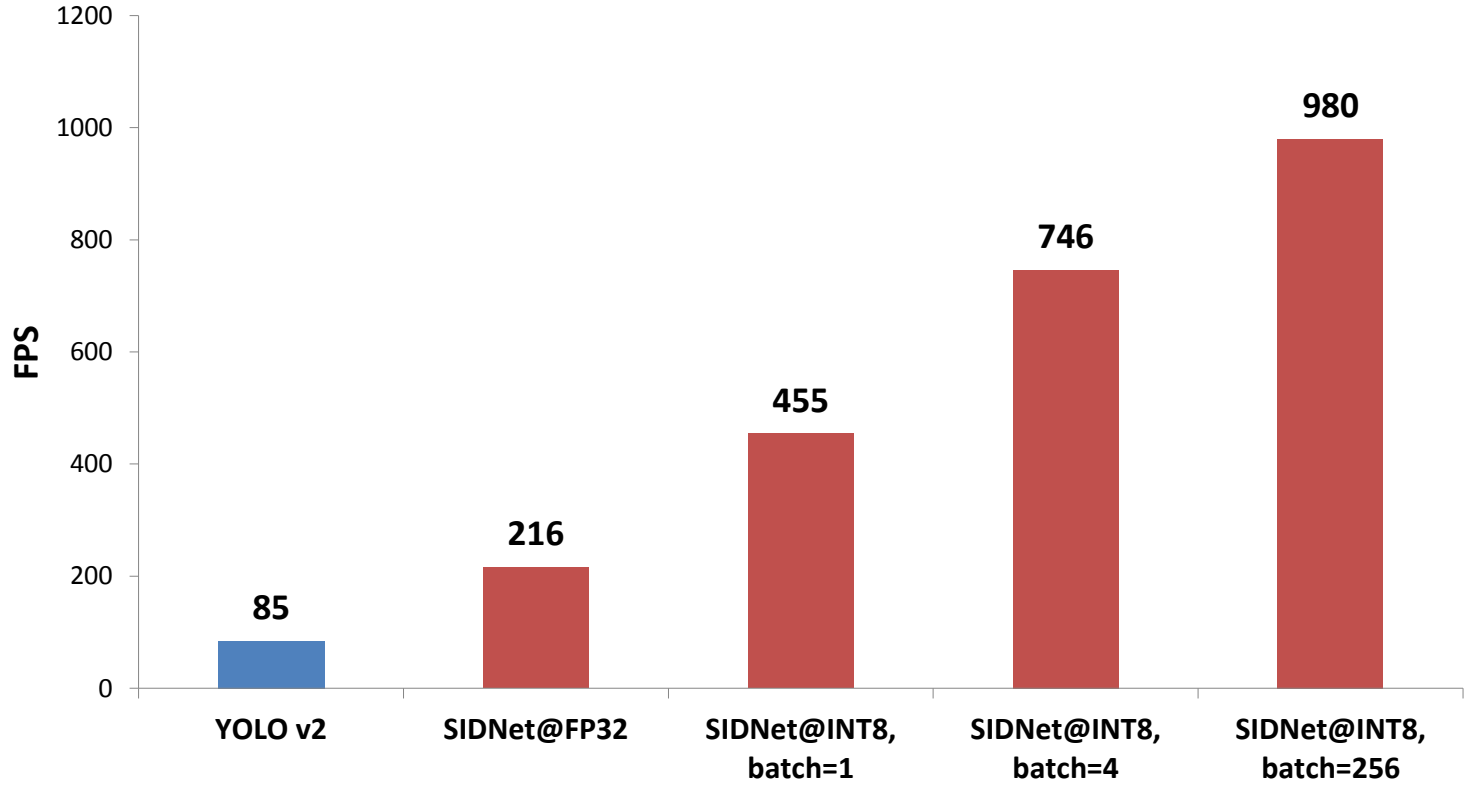
YOLO v2 *SIDNet*

Results: performance

Time measurement: network inference until bbox result

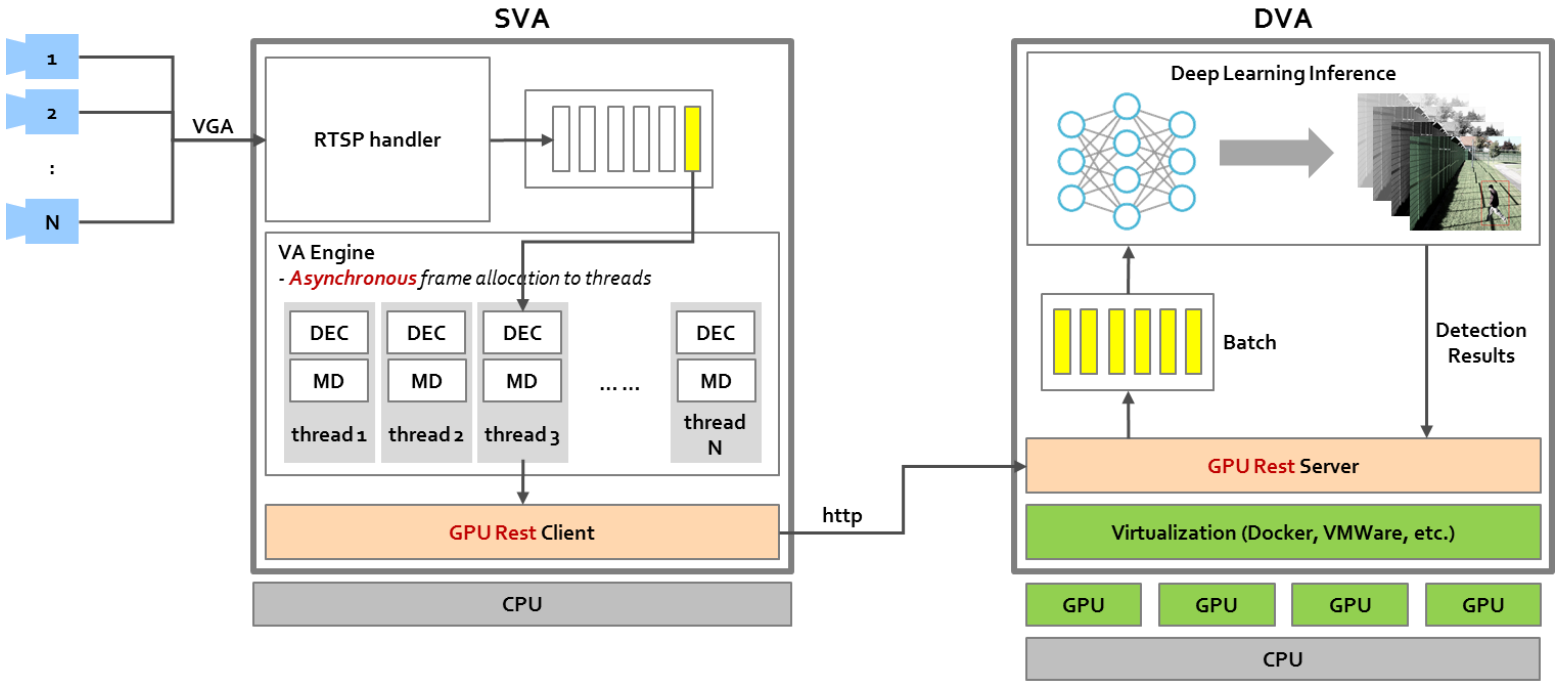
We exclude time of two modules for fair comparison for all experiments:

- *Image buffer transfer time from CPU to GPU for it depends on system hardware*
- *NMS, which is not necessary in our intrusion detection system*



Run on P40

Batch inference



SIDNet @INT8	Batch size							
	1	4	8	16	32	64	128	256
Total time (ms)	**2.20**	5.36	9.84	17.44	33.92	66.56	131.84	**261.12**
Time / frame (ms)	**2.20**	1.34	1.23	1.09	1.06	1.04	1.03	**1.02**

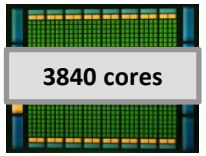
2.1x faster

1,000,000 runs on P40

Conclusion

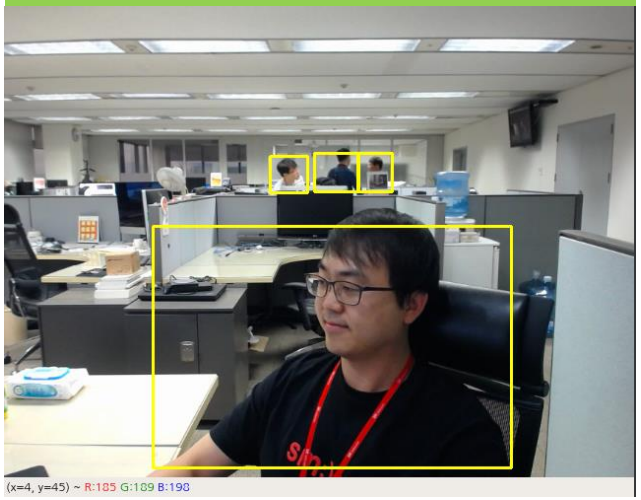
YOLO v2 : 85 fps

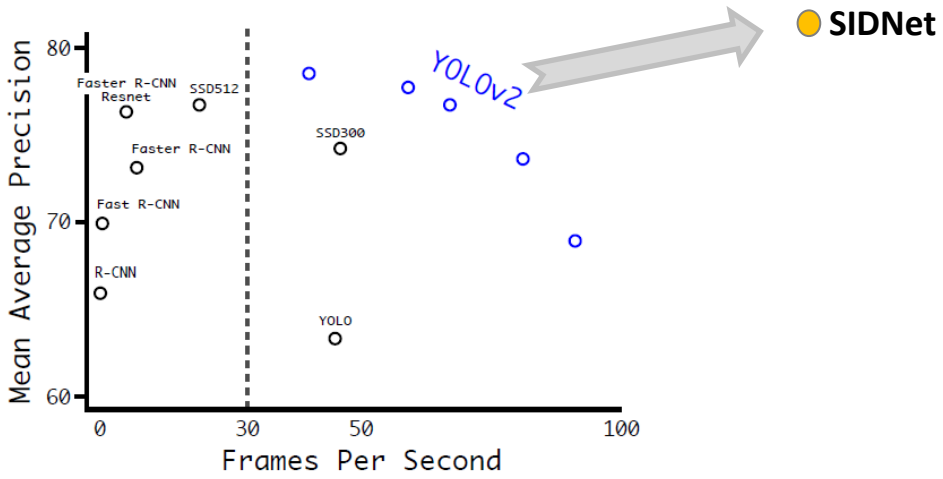
11x faster



NVIDIA Tesla P40

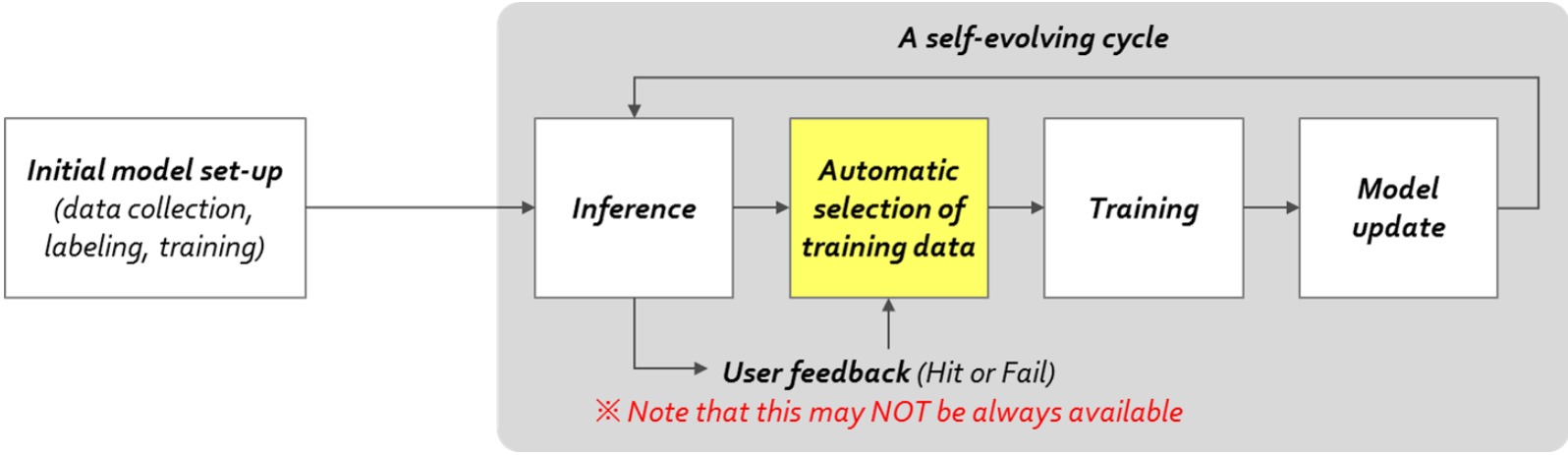
SIDNet : 980 fps





Future works

- *Online Incremental Learning for Individual Cameras*



- *Jetson TX based Deep Learning Inference for Front-end Devices (Camera, etc.)*