Real-time Object Detection with Convolutional Neural Networks

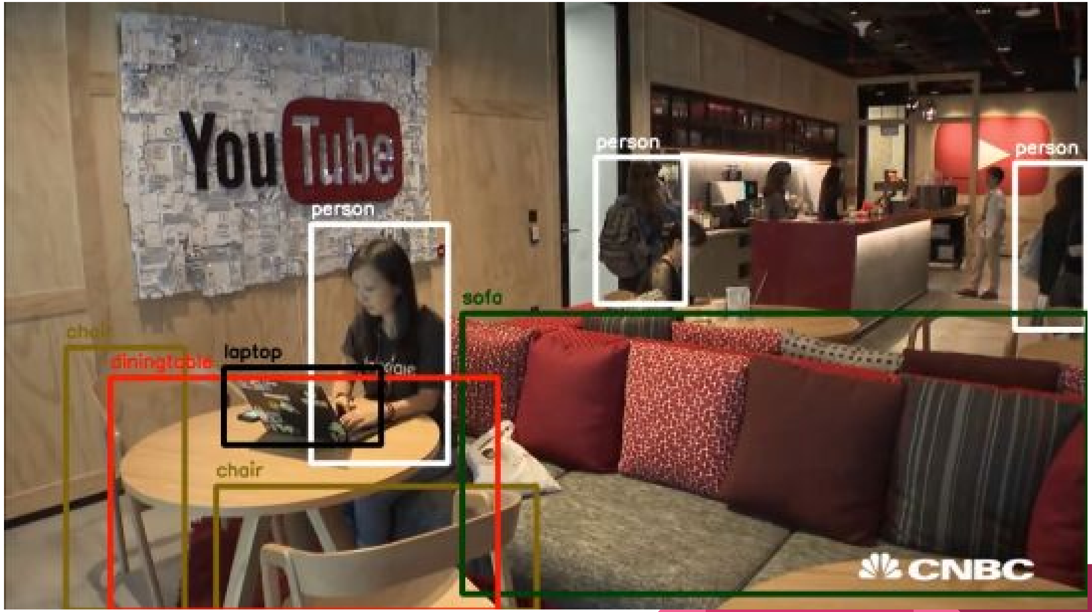
o.shykhkerimov@quantumobile.com
i.mishchenko@quantumobile.com

Plan

- Introduction to real-time object detection problem
- YOLO and Tiny-YOLO architecture
- SqueezeNet architecture
- SqueezeDet architecture
- Evaluation environment
- Results

Introduction

- R-CNN, Fast R-CNN, Faster R-CNN as object detectors
- YOLO, YOLOv2 models for better performance
- SqueezeNet, SqueezeDet, Tiny-YOLO models for less memory consuming

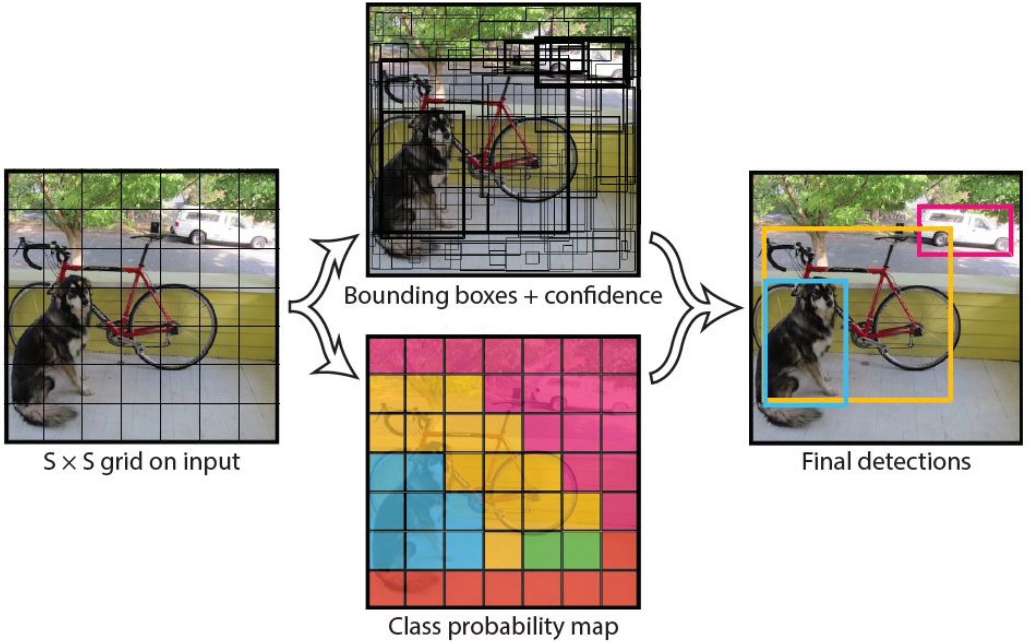


Models

You Only Look Once

- “Extremely Fast and Refreshingly Simple”
- “YOLO reasons globally about the image when making predictions. Unlike sliding window and region proposal-based techniques, YOLO sees the entire image during training and test time so it implicitly encodes contextual information about classes as well as their appearance.”

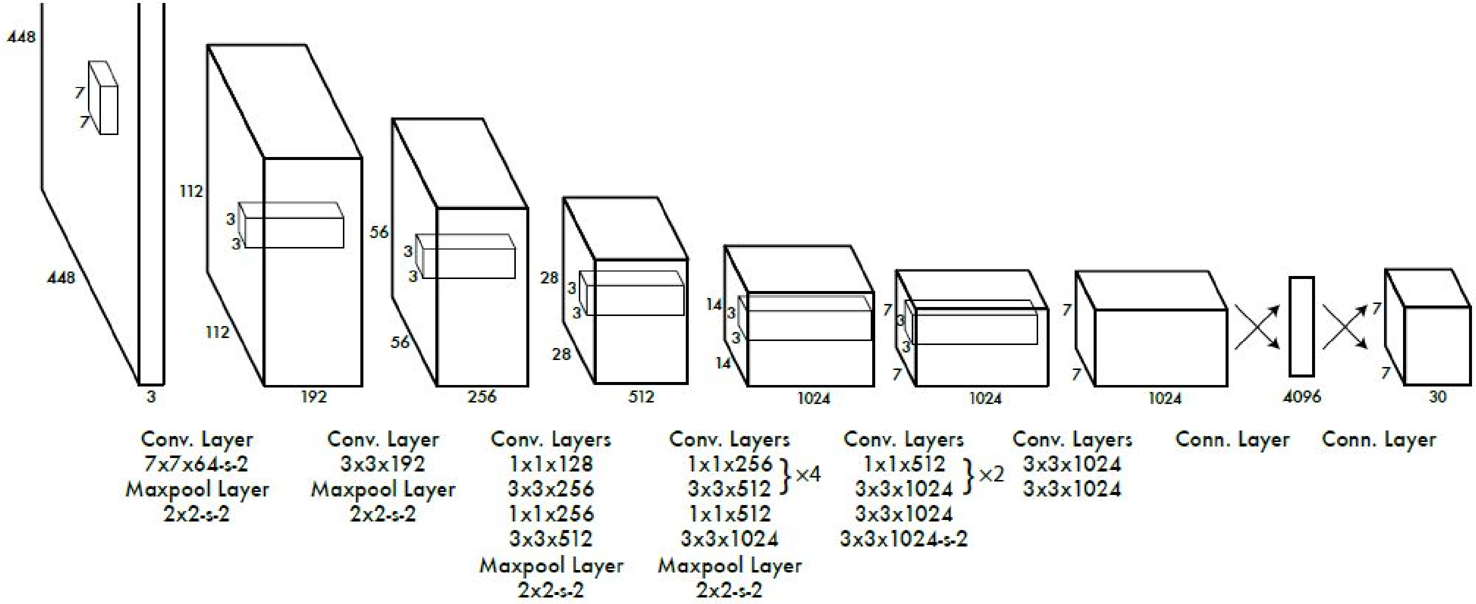
YOLO Model



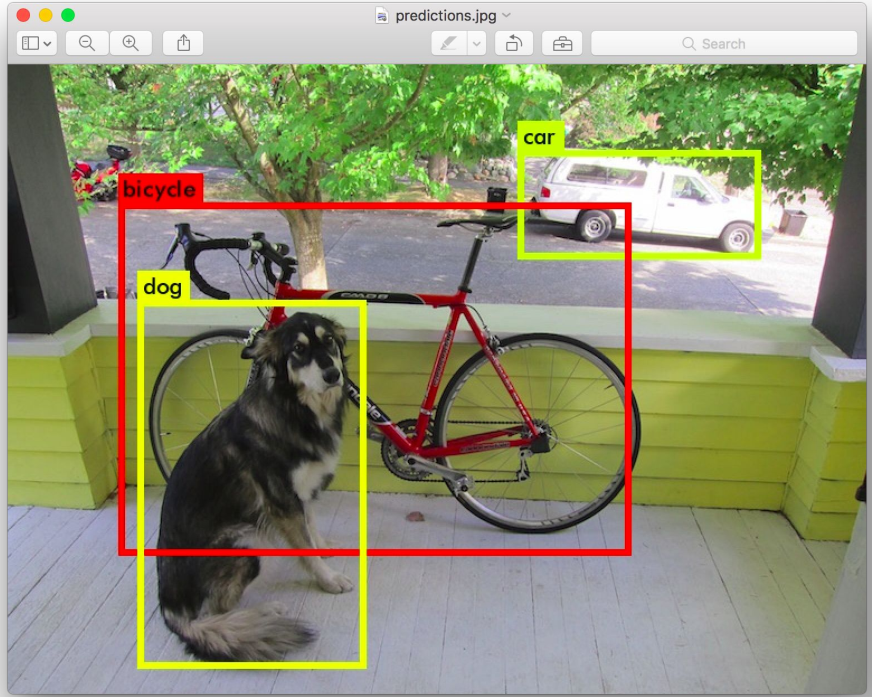
$$\Pr(\text{Class}_i|\text{Object}) * \Pr(\text{Object}) * \text{IOU}^{\text{truth}}_{\text{pred}} = \Pr(\text{Class}_i) * \text{IOU}^{\text{truth}}_{\text{pred}}$$

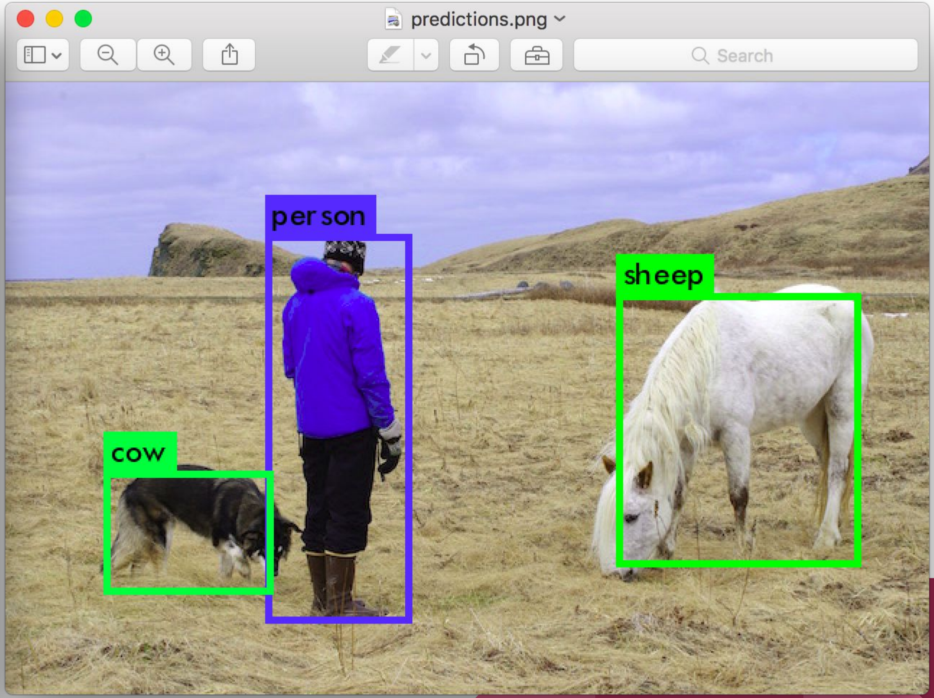
- Even grid;
- simultaneously predicts bounding boxes, confidence in those boxes, and class probabilities;
- predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor;
- S = 7, B = 2, C = 20 -> ~ 1500

YOLO Architecture



YOLO vs. Tiny YOLO





YOLOv2

	YOLO								YOLOv2
batch norm?		✓	✓	✓	✓	✓	✓	✓	✓
hi-res classifier?			✓	✓	✓	✓	✓	✓	✓
convolutional?				✓	✓	✓	✓	✓	✓
anchor boxes?				✓	✓				
new network?					✓	✓	✓	✓	✓
dimension priors?						✓	✓	✓	✓
location prediction?						✓	✓	✓	✓
passthrough?							✓	✓	✓
multi-scale?								✓	✓
hi-res detector?									✓
VOC2007 mAP	63.4	65.8	69.5	69.2	69.6	74.4	75.4	76.8	**78.6**

YOLOv2. Performance

Detection Frameworks	Train	mAP	FPS
Fast R-CNN [5]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[15]	2007+2012	73.2	7
Faster R-CNN ResNet[6]	2007+2012	76.4	5
YOLO [14]	2007+2012	63.4	45
SSD300 [11]	2007+2012	74.3	46
SSD500 [11]	2007+2012	76.8	19
YOLOv2 288 × 288	2007+2012	69.0	91
YOLOv2 352 × 352	2007+2012	73.7	81
YOLOv2 416 × 416	2007+2012	76.8	67
YOLOv2 480 × 480	2007+2012	77.8	59
YOLOv2 544 × 544	2007+2012	**78.6**	40

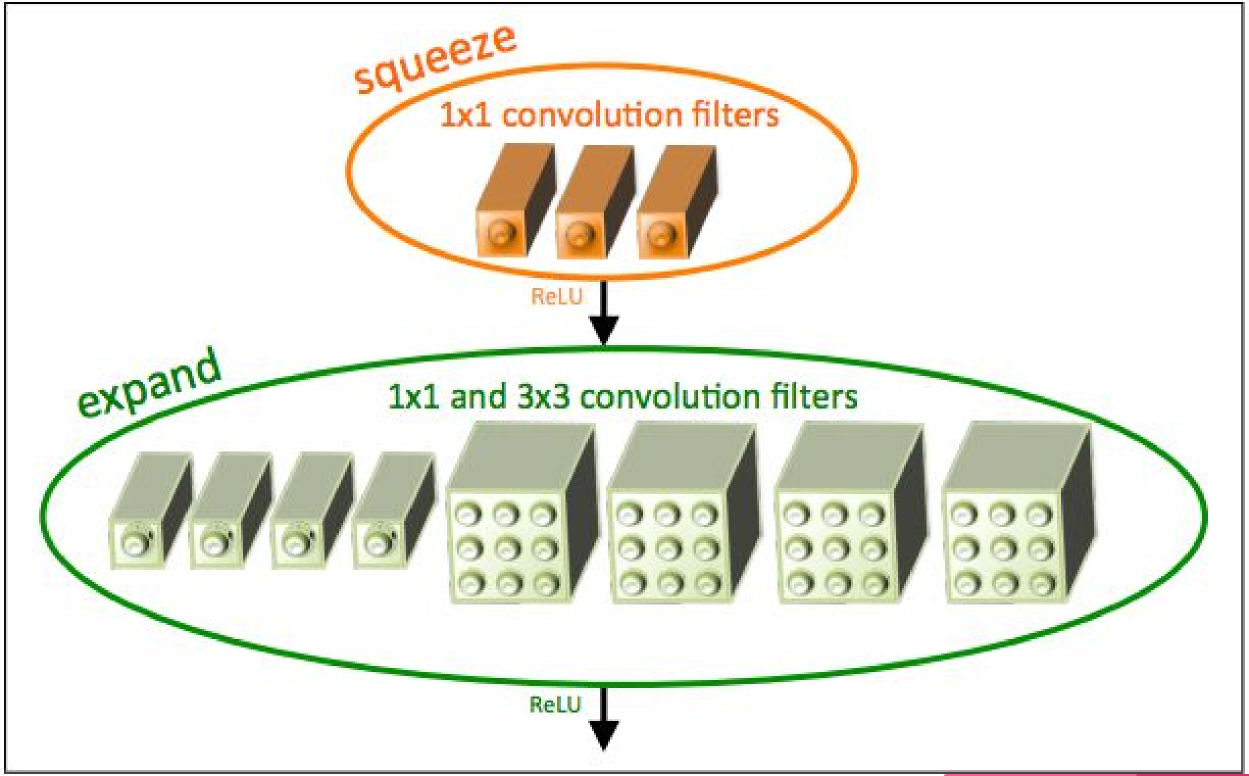
- PASCAL VOC 2007 as a dataset;
- different sizes of input for YOLOv2;
- all timing information is on a Geforce GTX Titan X.

SqueezeNet

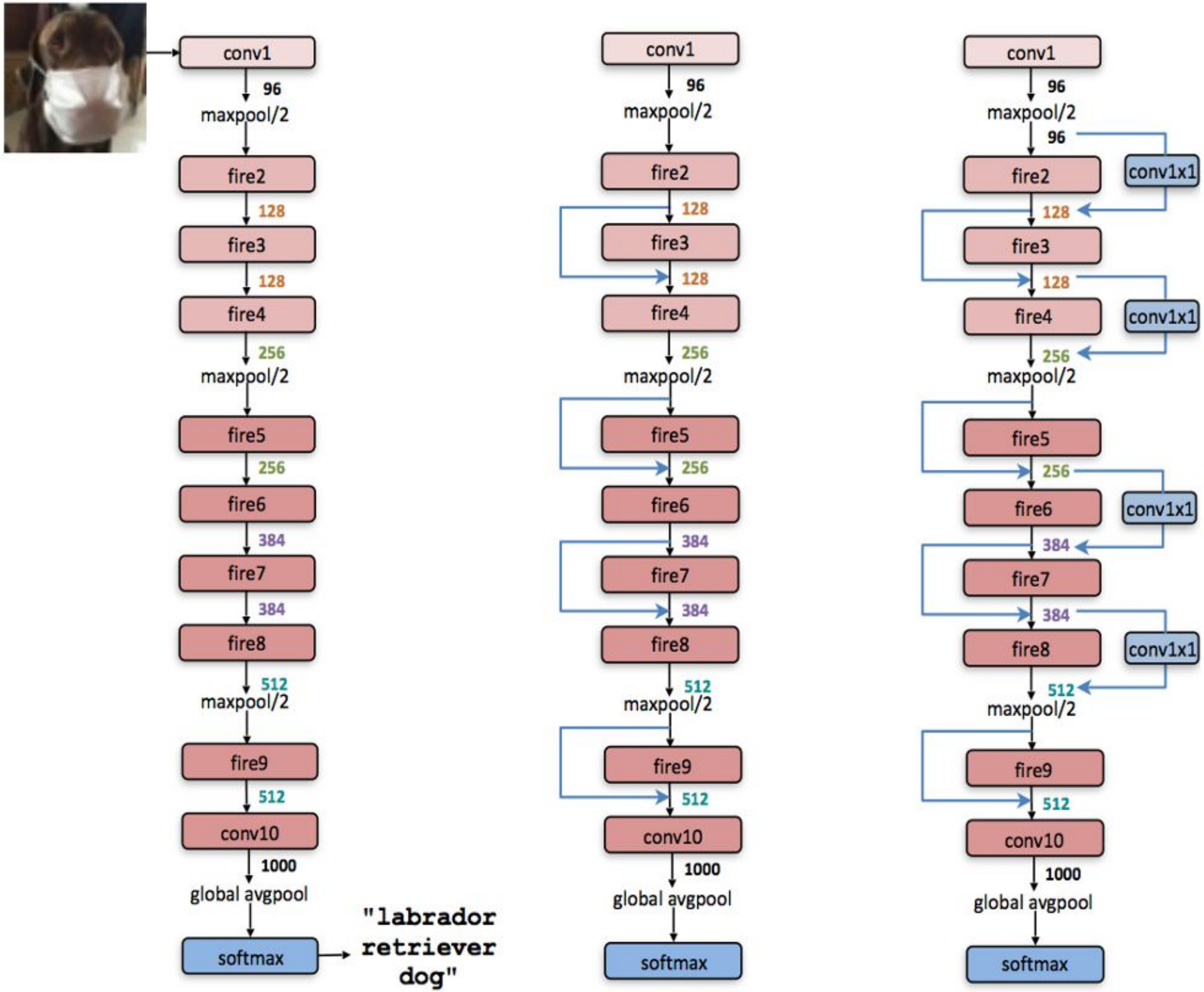
“SqueezeNet achieves AlexNet-level accuracy on ImageNet with 50x fewer parameters”

- Strategy 1. Replace 3x3 filters with 1x1 filters.
- Strategy 2. Decrease the number of input channels to 3x3 filters.
- Strategy 3. Downsample late in the network so that convolution layers have large activation maps.

SqueezeNet. Fire Module



SqueezeNet Architecture



SqueezeDet

- “Inspired by YOLO, we also adopt a single-stage detection pipeline in which region proposal and classification is performed by one single network simultaneously”
- “When choosing the “backbone” CNN structure, our focus is mainly on model size and energy efficiency, and SqueezeNet is our top candidate”

SqueezeDet Architectural Dimensions

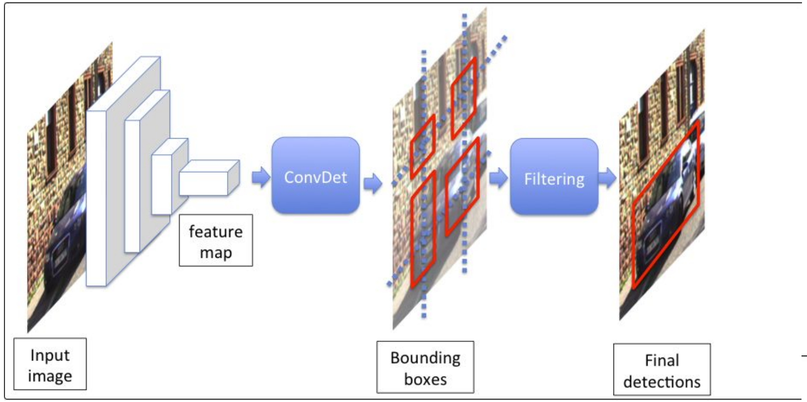
Table 1. SqueezeNet architectural dimensions. (The formatting of this table was inspired by the Inception2 paper [14].)

layer name/type	output size	filter size / stride (if not a fire layer)	depth	s_{1x1} (#1x1 squeeze)	e_{1x1} (#1x1 expand)	e_{3x3} (#3x3 expand)
input image	224x224x3					
conv1	111x111x96	7x7/2 (x96)	1			
maxpool1	55x55x96	3x3/2	0			
fire2	55x55x128		2	16	64	64
fire3	55x55x128		2	16	64	64
fire4	55x55x256		2	32	128	128
maxpool4	27x27x256	3x3/2	0			
fire5	27x27x256		2	32	128	128
fire6	27x27x384		2	48	192	192
fire7	27x27x384		2	48	192	192
fire8	27x27x512		2	64	256	256
maxpool8	13x12x512	3x3/2	0			
fire9	13x13x512		2	64	256	256
conv10	13x13x1000	1x1/1 (x1000)	1			
avgpool10	1x1x1000	13x13/1	0			

activations
(input/output data between layers)

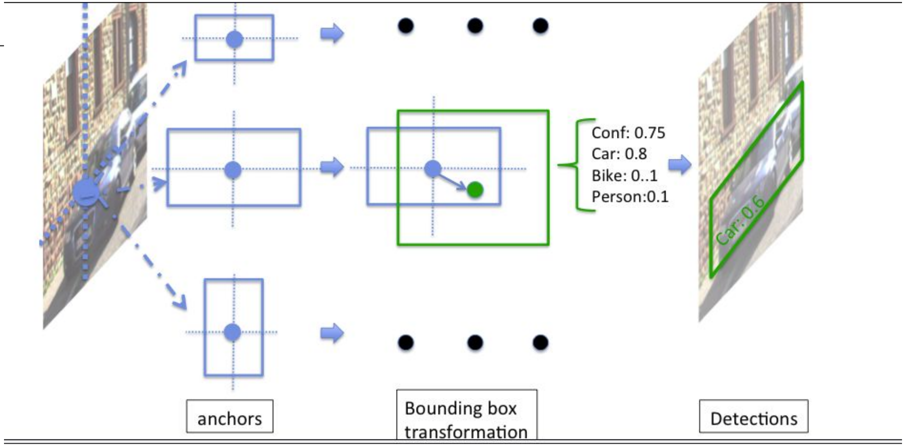
parameters

SqueezeDet



Detection Pipeline

Bounding Box Transformation



Evaluation

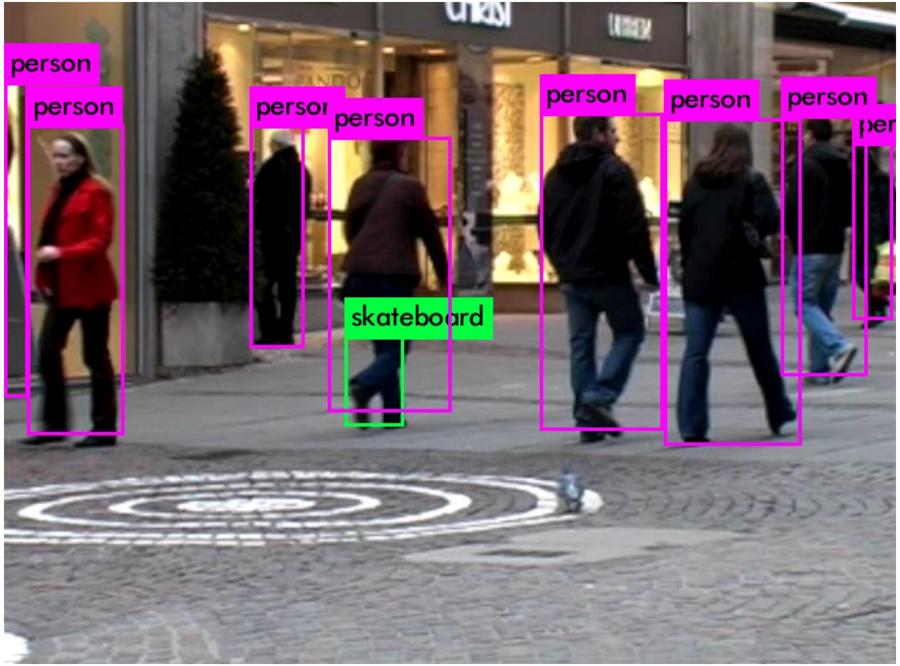
Evaluation environment

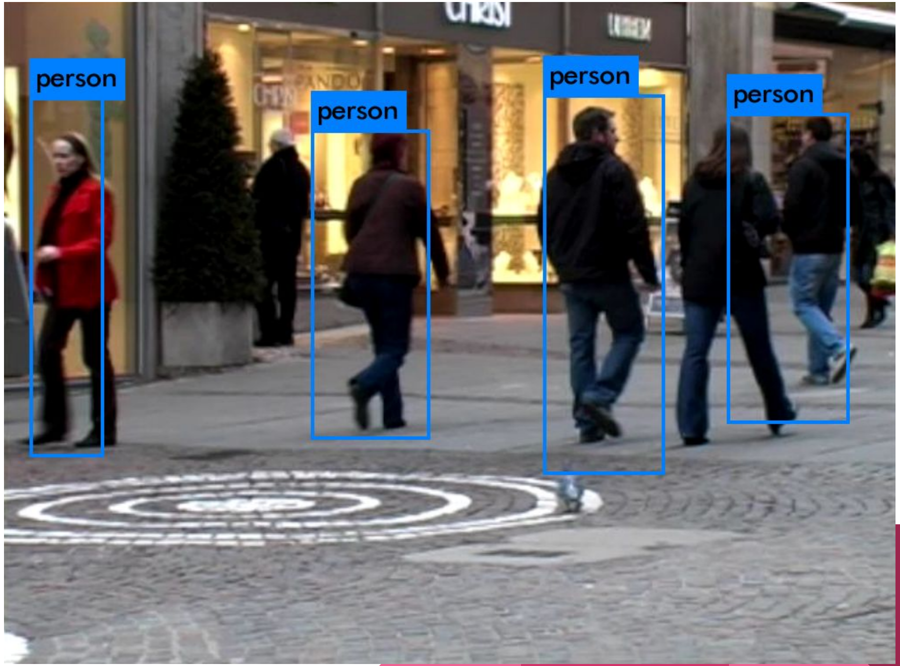
	Nvidia Jetson TX2	Custom GPU instance
GPU	NVIDIA Pascal, 256 CUDA cores	GeForce GTX 1080 Ti 11GB, 3584 CUDA cores
CPU	HMP Dual Denver 2/2 MB L2 + Quad ARM® A57/2 MB L2	Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz
Memory	8 GB 128 bit LPDDR4	DDR4 16Gb 3600MHz

Results

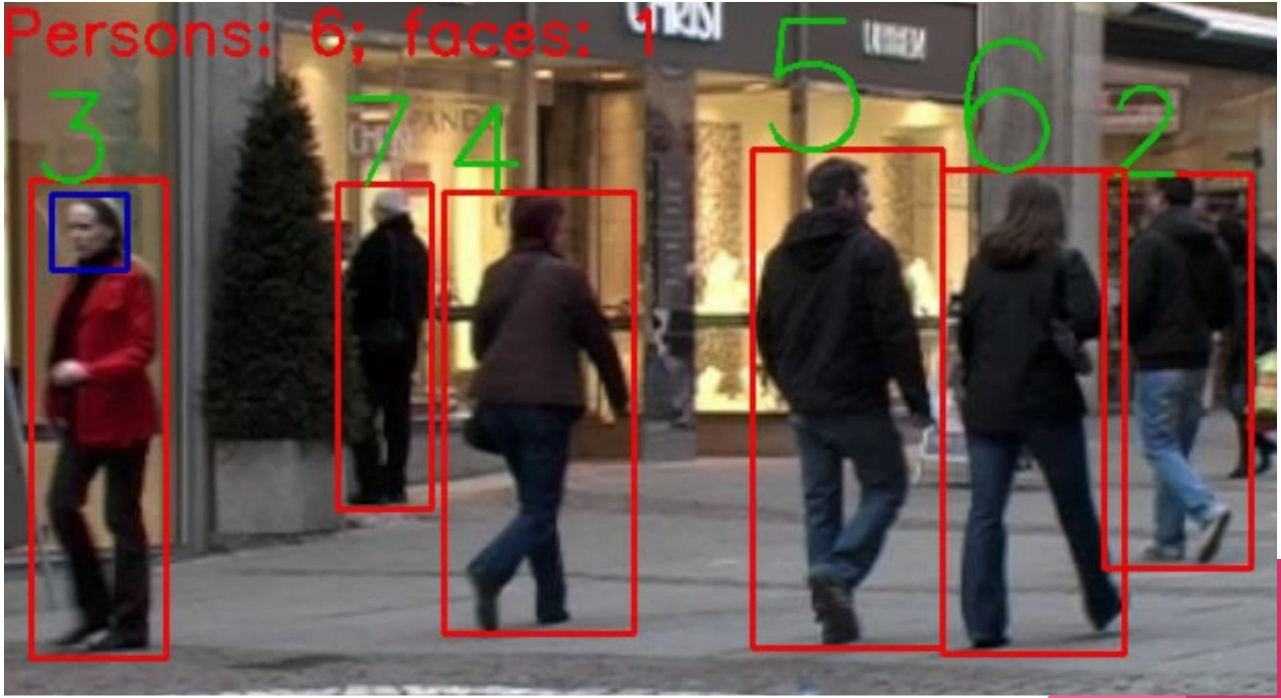
Model	Input Size	Environment	Framework	FPS
Tiny YOLO	416x416	Jetson TX2	DarkNet	30
Tiny YOLO	416x416	Jetson TX2	DarkFlow	8.4
Tiny YOLO	416x416	Custom GPU	DarkNet	48.7
Tiny YOLO	416x416	Custom GPU	DarkFlow	77.1
YOLO	608x608	Jetson TX2	DarkNet	5.1
YOLO	608x608	Jetson TX2	DarkFlow	2.6
YOLO	608x608	Custom GPU	DarkNet	20.2
YOLO	608x608	Custom GPU	DarkFlow	31.4
SqueezeDet	1242x375	Jetson TX2	TensorFlow	9.4
SqueezeDet	1242x375	Custom GPU	TensorFlow	114.2

Detection Results





Tracking Results



See you :)