Object Detection Tensorflow API

Paper (Speed/accuracy trade-offs for modern convolutional object detectors)

Experiment Results

22 June 2017

Junho Song

Tensorflow Object Detection API

Creating accurate machine learning models capable of localizing and identifying multiple objects in a single image remains a core challenge in computer vision. The TensorFlow Object Detection API is an open source framework built on top of TensorFlow that makes it easy to construct, train and deploy object detection models. At Google we've certainly found this codebase to be useful for our computer vision needs, and we hope that you will as well.

Speed/accuracy trade-offs for modern convolutional object detectors

Jonathan Huang Vivek Rathod Chen Sun Menglong Zhu Anoop Korattikara
Alireza Fathi Ian Fischer Zbigniew Wojna Yang Song Sergio Guadarrama
Kevin Murphy
Google Research

However, it can be difficult for practitioners to decide what architecture is best suited to their application. Standard accuracy metrics, such as mean average precision (mAP), do not tell the entire story, since for real deployments of computer vision systems, running time and memory usage are also critical. For example, mobile devices often require a small memory footprint, and self driving

for practitioner
- mAP
- running time
- memory usage

Unfortunately, only a small subset of papers (e.g., R-FCN [6], SSD [26] YOLO [29]) discuss running time in any detail. Furthermore, these papers typically only state that they achieve some frame-rate, but do not give a full picture of the speed/accuracy trade-off, which depends on many other factors, such as which feature extractor is used, input image sizes, etc.

In this paper, we seek to explore the speed/accuracy trade-off of modern detection systems in an exhaustive and fair way. While this has been studied for full image classification((e.g., [3]), detection models tend to be significantly more complex. We primarily investigate single-model/single-pass detectors, by which we mean models that do not use ensembling, multi-crop methods, or other "tricks" such as horizontal flipping. In other words, we only pass a single image through a single network. For simplicity (and because it is more important for users of this technology), we focus only on test-time performance and not on how long these models take to train.

survey

detection systems, and describe how the leading ones follow very similar designs.

implementation

- We describe our flexible and unified implementation of three meta-architectures (Faster R-CNN, R-FCN and SSD) in Tensorflow which we use to do extensive experiments that trace the accuracy/speed trade-off curve for different detection systems, varying meta-architecture, feature extractor, image resolution, etc.

(three - meta - architecture)

- Our findings show that using fewer proposals for Faster R-CNN can speed it up significantly without a big loss in accuracy, making it competitive with its faster cousins, SSD and RFCN. We show that SSDs performance is less sensitive to the quality of the feature extractor than Faster R-CNN and R-FCN. And we identify sweet spots on the accuracy/speed trade-off curve where gains in accuracy are only possible by sacrificing speed (within the family of detectors presented here).

Faster RCNN

SSD > Faster RCNN, RFCN

- Several of the meta-architecture and feature-extractor combinations that we report have never appeared before in literature. We discuss how we used some of these novel combinations to train the winning entry of the 2016 COCO object detection challenge.

combination

Model	Top-1 accuracy	Num. Params.
VGG-16	71.0	14,714,688
MobileNet	71.1	3,191,072
Inception V2	73.9	10,173,112
ResNet-101	76.4	42,605,504
Inception V3	78.0	21,802,784
Inception Resnet V2	80.4	54,336,736

Table 2: Properties of the 6 feature extractors that we use. Top-1 accuracy is the classification accuracy on ImageNet.

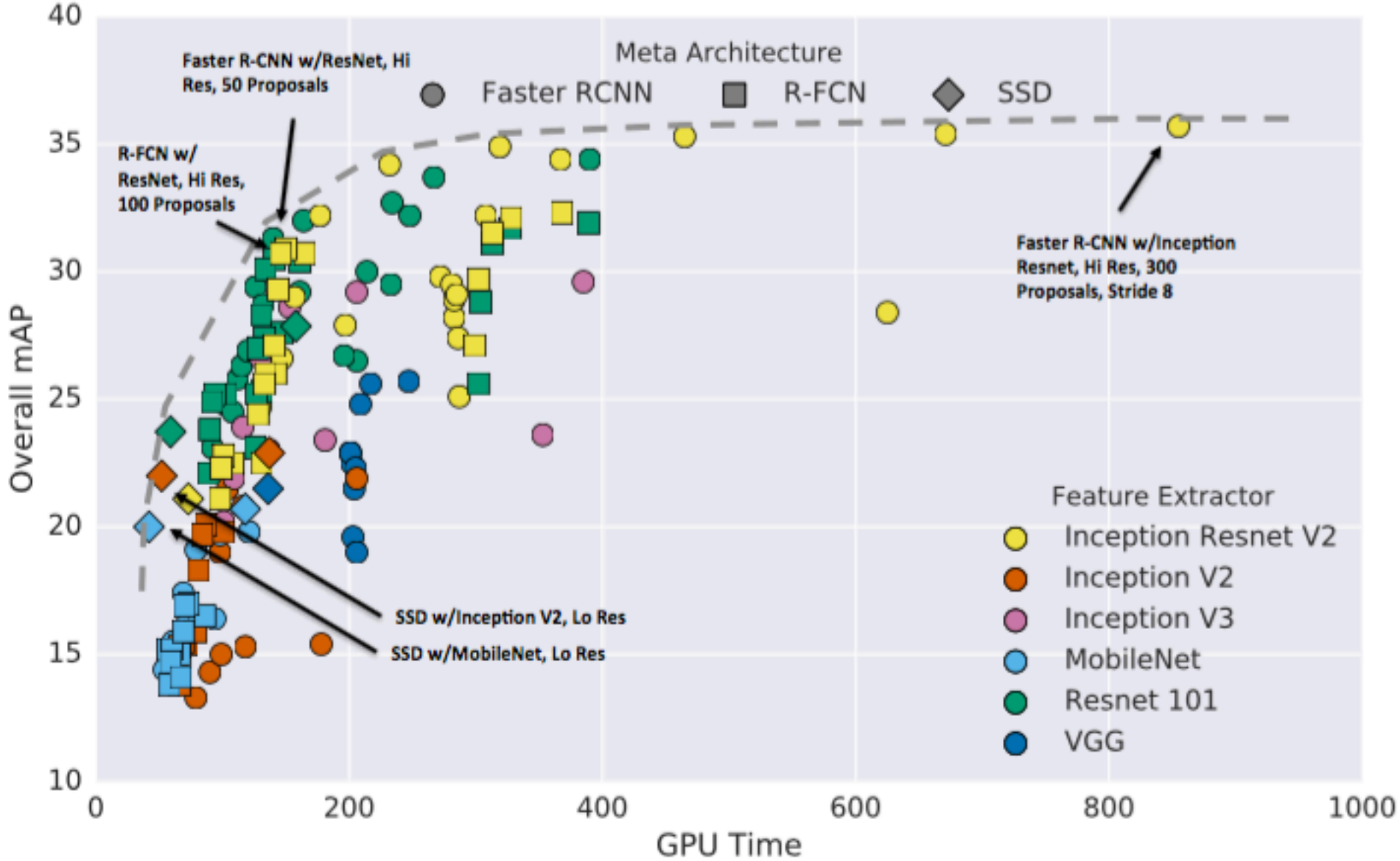


Figure 2: Accuracy vs time, with marker shapes indicating meta-architecture and colors indicating feature extractor. Each (meta-architecture, feature extractor) pair can correspond to multiple points on this plot due to changing input sizes, stride, etc.

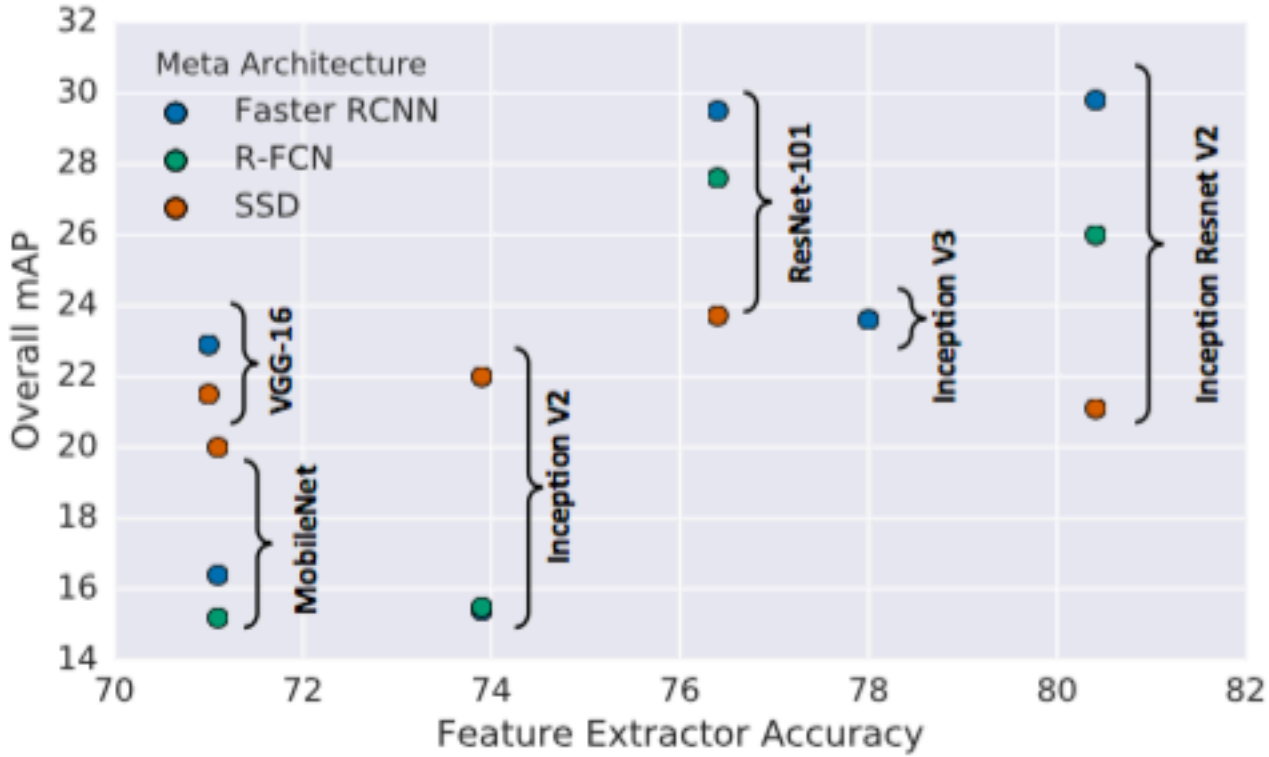


Figure 3: Accuracy of detector (mAP on COCO) vs accuracy of feature extractor (as measured by top-1 accuracy on ImageNet-CLS). To avoid crowding the plot, we show only the low resolution models.

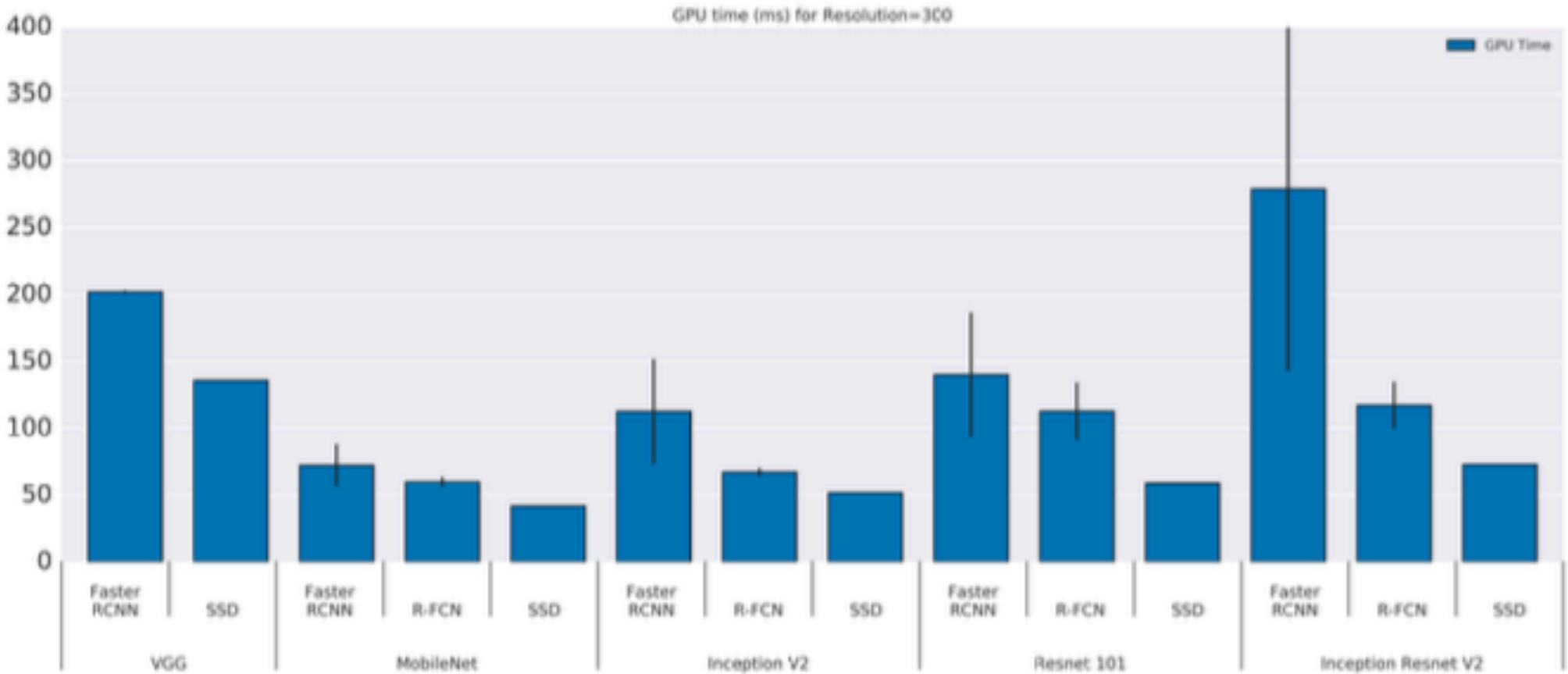
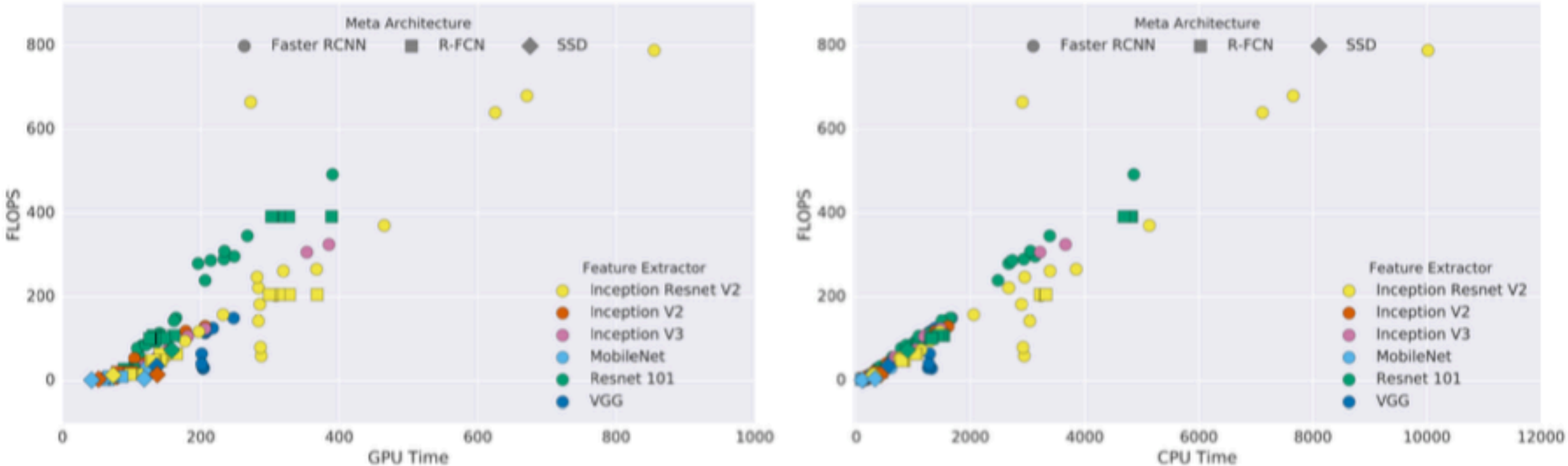


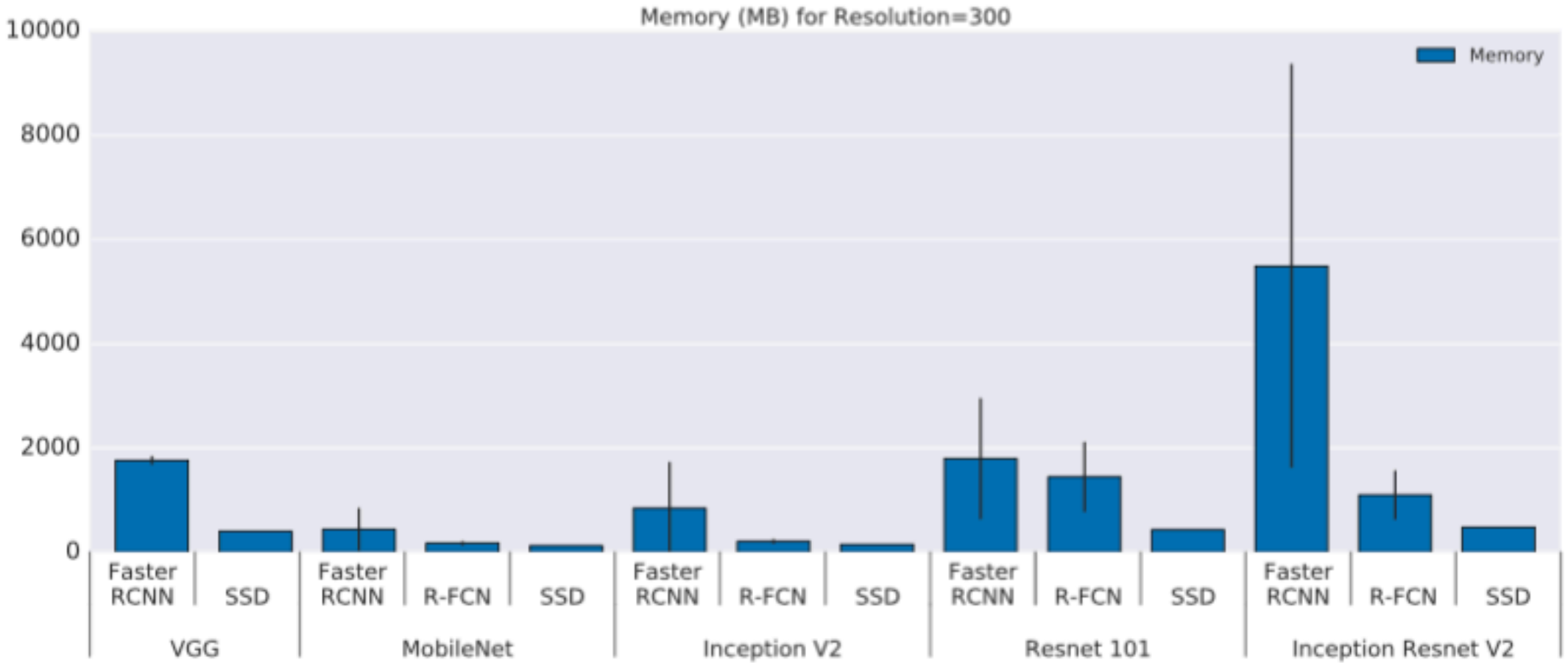
Figure 7: GPU time (milliseconds) for each model, for image resolution of 300.

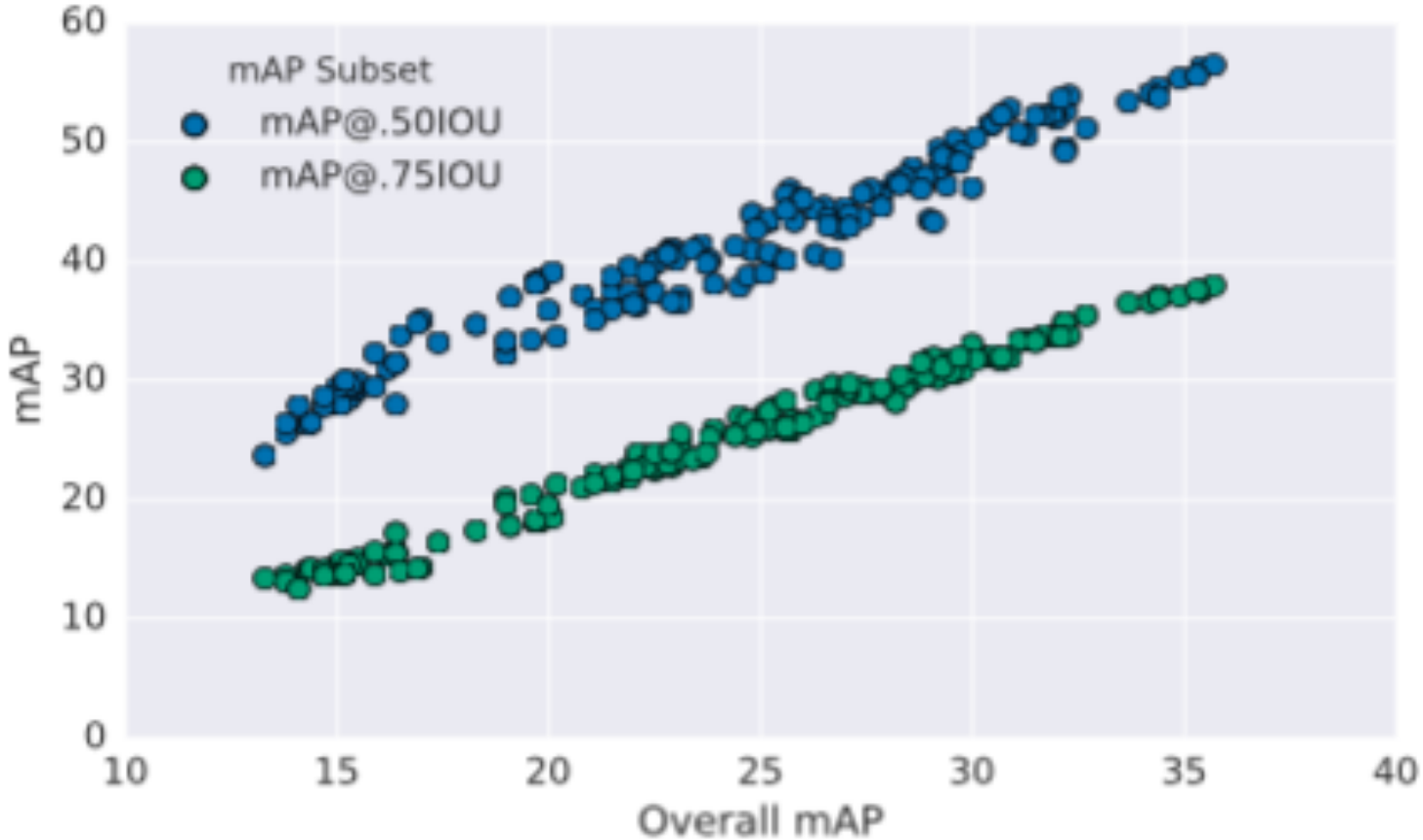


(a) GPU.

(b) CPU.

Figure 8: FLOPS vs time.


Memory (MB) for Resolution=300
Memory
10000
8000
6000
4000
2000
0
Faster RCNN
SSD
Faster RCNN
R-FCN
SSD
Faster RCNN
R-FCN
SSD
Faster RCNN
R-FCN
SSD
Faster RCNN
R-FCN
SSD
VGG
MobileNet
Inception V2
Resnet 101
Inception Resnet V2


mAP Subset
mAP@.50IOU
mAP@.75IOU
mAP
Overall mAP
0
10
20
30
40
50
60
10
15
20
25
30
35
40

Tensorflow detection model zoo

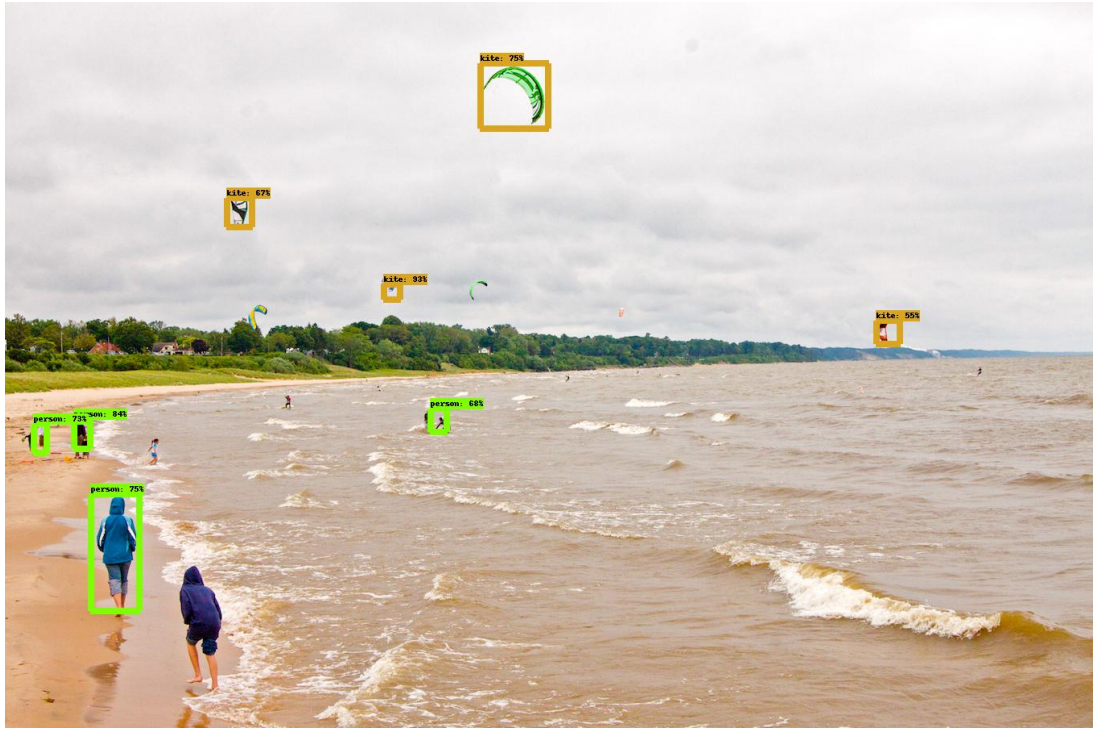
We provide a collection of detection models pre-trained on the COCO dataset. These models can be useful for out-of-the-box inference if you are interested in categories already in COCO (e.g., humans, cars, etc). They are also useful for initializing your models when training on novel datasets.

In the table below, we list each such pre-trained model including:

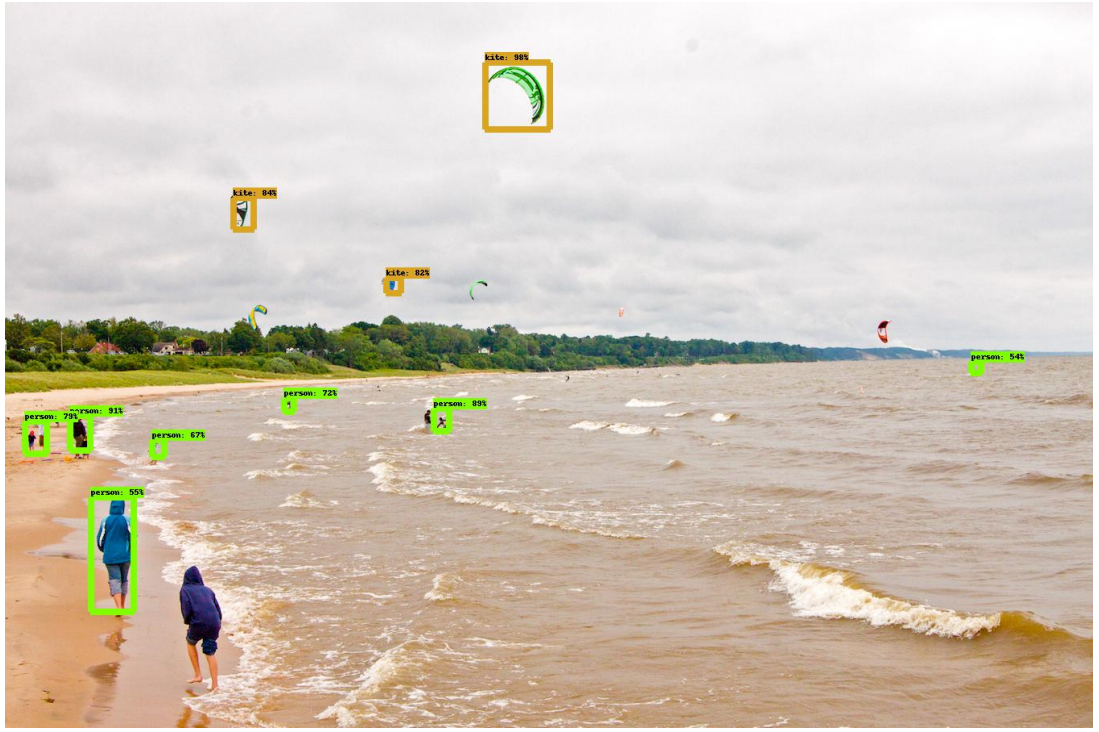
- a model name that corresponds to a config file that was used to train this model in the `samples/configs` directory,
- a download link to a tar.gz file containing the pre-trained model,
- model speed (one of {slow, medium, fast}),
- detector performance on COCO data as measured by the COCO mAP measure. Here, higher is better, and we only report bounding box mAP rounded to the nearest integer.
- Output types (currently only `Boxes`)

Model name	Speed	COCO mAP	Outputs
ssd_mobilenet_v1_coco	fast	21	Boxes
ssd_inception_v2_coco	fast	24	Boxes
rfcn_resnet101_coco	medium	30	Boxes
faster_rcnn_resnet101_coco	medium	32	Boxes
faster_rcnn_inception_resnet_v2_atrous_coco	slow	37	Boxes

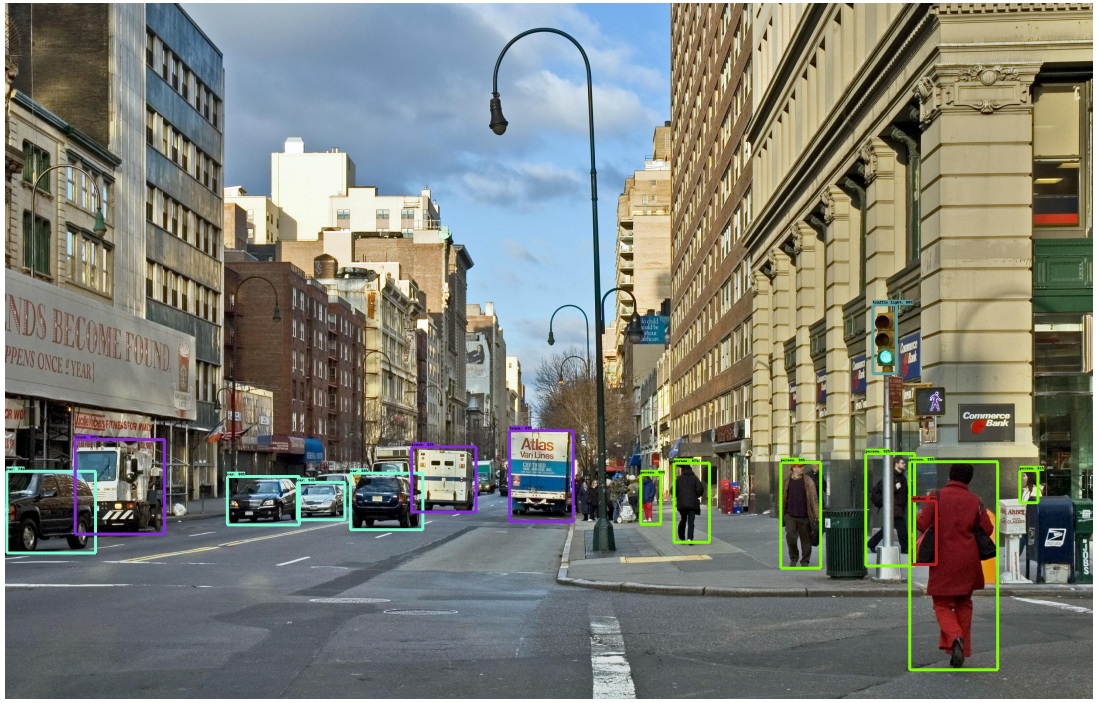
ssd_mobilenet_v1_coco



ssd_inception_v2_coco



rfcn_resnet101_coco



faster_rcnn_resnet101_coco

faster_rcnn_inception_resnet_v2_atrous_coco

Memory Usage

Model name	Memory Usage
squeezeDet	302 MiB
YOLO_small	812 MiB

Model name	Memory Usage
SSD_mobilenet_v1_coco	396 MiB
SSD_inception_v2_coco	513 MiB
RFCN_resnet101_coco	639 MiB
FasterRCNN_resnet101_coco	1,126 MiB
FasterRCNN_inception_resnet_v2_ atrous_coco	1,429 MiB

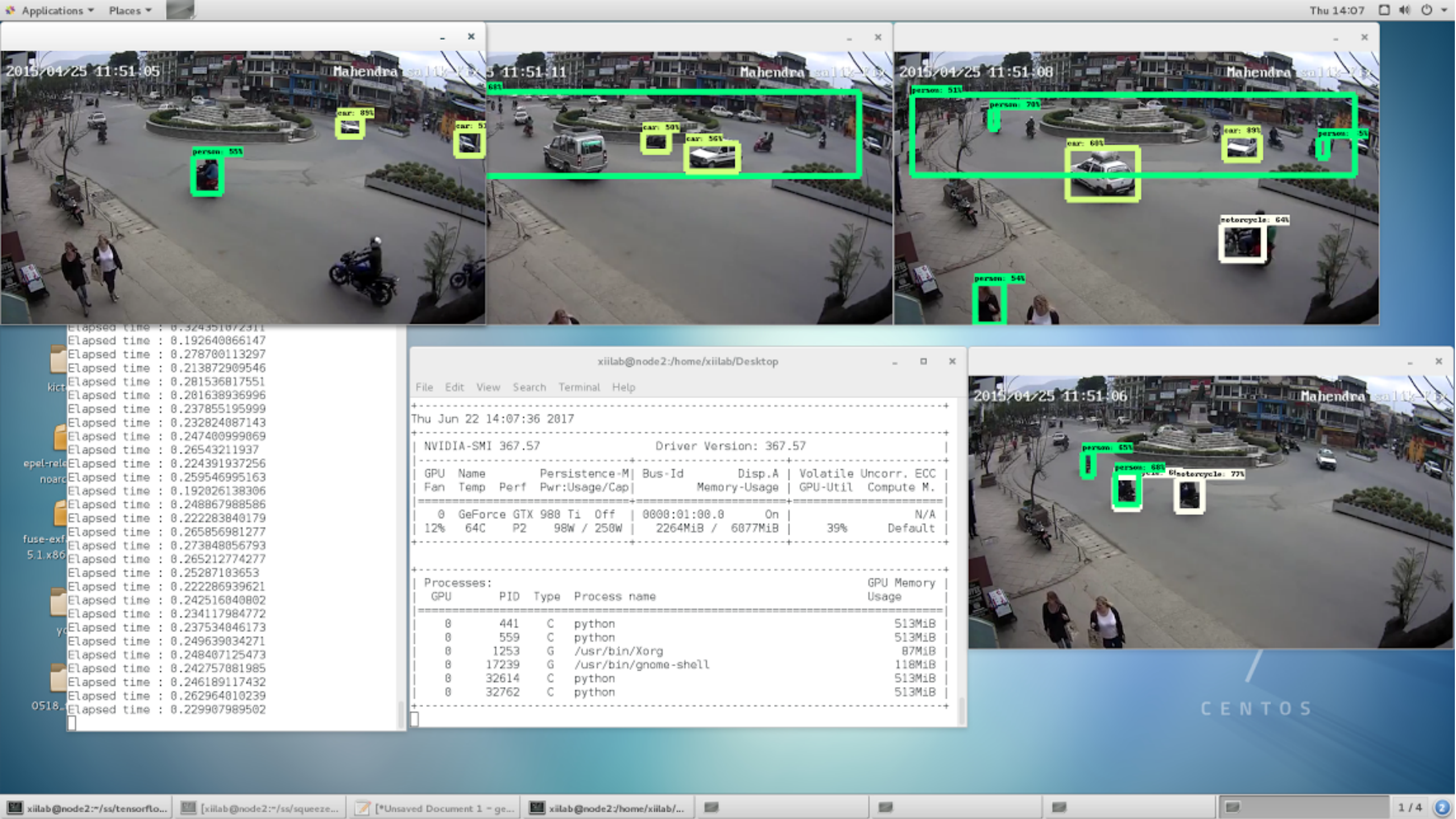
Inference Time

- Model
  - ✓ MobileNet v1

- GPU
  - ✓ 980ti GPU UNIT 1 (6077MB)

- Results
  - ✓ 1 process 당 최소 GPU Memory : 396MB

  - ✓ 1 process / 1 frame : 0.8~1s -> 1초당 10~12 frame
  - ✓ 4 process / 1 frame : 0.2~0.3s -> 1초당 3~5 frame


Applications
Places
Thu 14:07
2015/04/25 11:51:05
Mahendra
person: 55%
car: 89%
2015/04/25 11:51:11
Mahendra
car: 50%
car: 56%
2015/04/25 11:51:08
Mahendra
person: 51%
person: 70%
car: 68%
car: 89%
motorcycle: 64%
person: 54%
Elapsed time : 0.192640866147
Elapsed time : 0.278700113297
Elapsed time : 0.213872909546
Elapsed time : 0.281536817551
Elapsed time : 0.201638936996
Elapsed time : 0.237855195999
Elapsed time : 0.232824087143
Elapsed time : 0.247400999069
Elapsed time : 0.26543211937
Elapsed time : 0.224391937256
Elapsed time : 0.259546995163
Elapsed time : 0.192026138306
Elapsed time : 0.248867988586
Elapsed time : 0.222283840179
Elapsed time : 0.265856981277
Elapsed time : 0.273848056793
Elapsed time : 0.265212774277
Elapsed time : 0.25287103653
Elapsed time : 0.222286939621
Elapsed time : 0.242516040802
Elapsed time : 0.234117984772
Elapsed time : 0.237534046173
Elapsed time : 0.249639034271
Elapsed time : 0.248407125473
Elapsed time : 0.242757081985
Elapsed time : 0.246189117432
Elapsed time : 0.262964010239
Elapsed time : 0.229907989502
xiilab@node2:/home/xiilab/Desktop
File Edit View Search Terminal Help
Thu Jun 22 14:07:36 2017
NVIDIA-SMI 367.57 Driver Version: 367.57
GPU Name Persistence-M Bus-Id Disp.A Volatile Uncorr. ECC
Fan Temp Perf Pwr:Usage/Cap Memory-Usage GPU-Util Compute M.
0 GeForce GTX 980 Ti Off 0000:01:00.0 On N/A
12% 64C P2 98W / 250W 2264MiB / 6077MiB 39% Default
Processes: GPU Memory
GPU PID Type Process name Usage
0 441 C python 513MiB
0 559 C python 513MiB
0 1253 G /usr/bin/Xorg 87MiB
0 17239 G /usr/bin/gnome-shell 118MiB
0 32614 C python 513MiB
0 32762 C python 513MiB
2015/04/25 11:51:06
Mahendra
person: 85%
person: 68%
motorcycle: 77%
CENTOS
xiilab@node2:~/ss/tensorflo...
[xiilab@node2:~/ss/squeeze...
[*Unsaved Document 1 - ge...
xiilab@node2:/home/xiilab/...
1 / 4