Traffic Signs Detection and Recognition System (TSDR)

Team members

Marco Magdy William

Bolis Karam Soliman

Pavly Salah Zaki

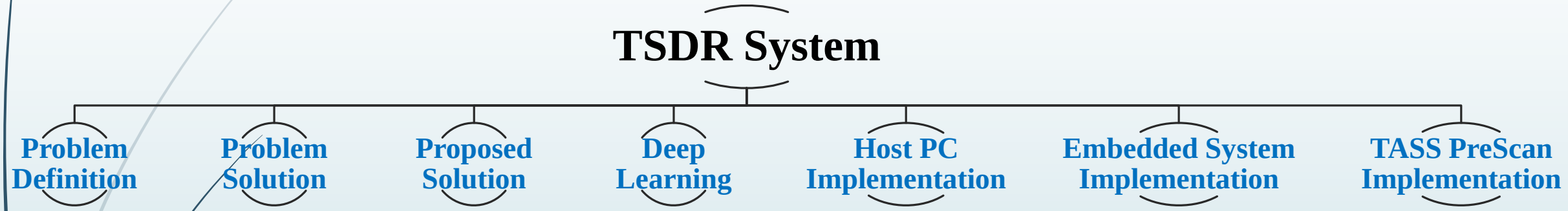
Kerolos Gamal Alexsan

Under the appreciated supervision of

Dr. Maher Mansour

Eng. Kerolos Karam **and** ***Dr. Magdy El-Morsy*** **from** ***Mentor Graphics***

Contents

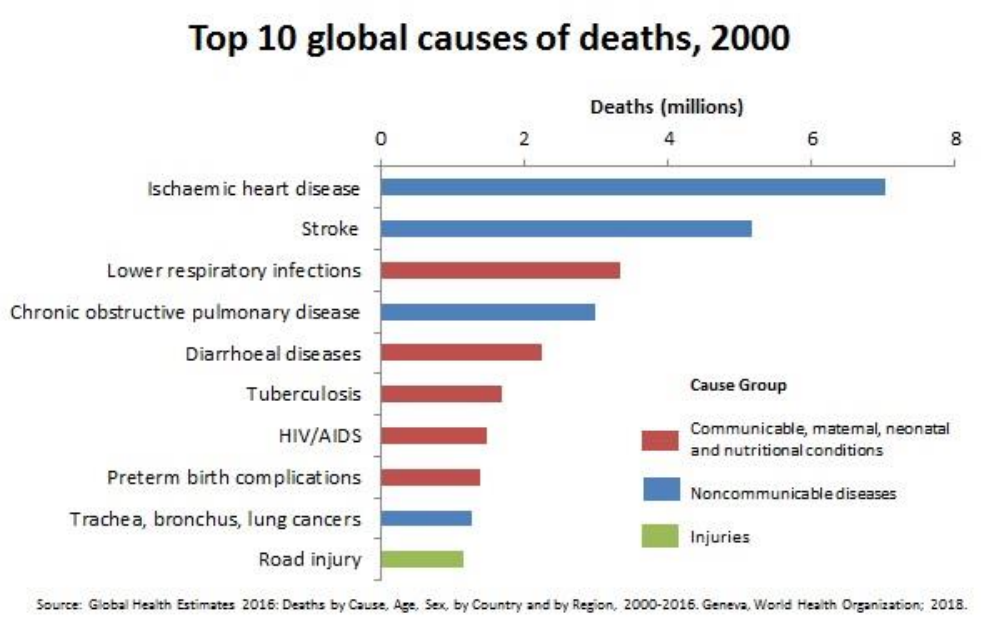

TSDR System
Problem Definition
Problem Solution
Proposed Solution
Deep Learning
Host PC Implementation
Embedded System Implementation
TASS PreScan Implementation

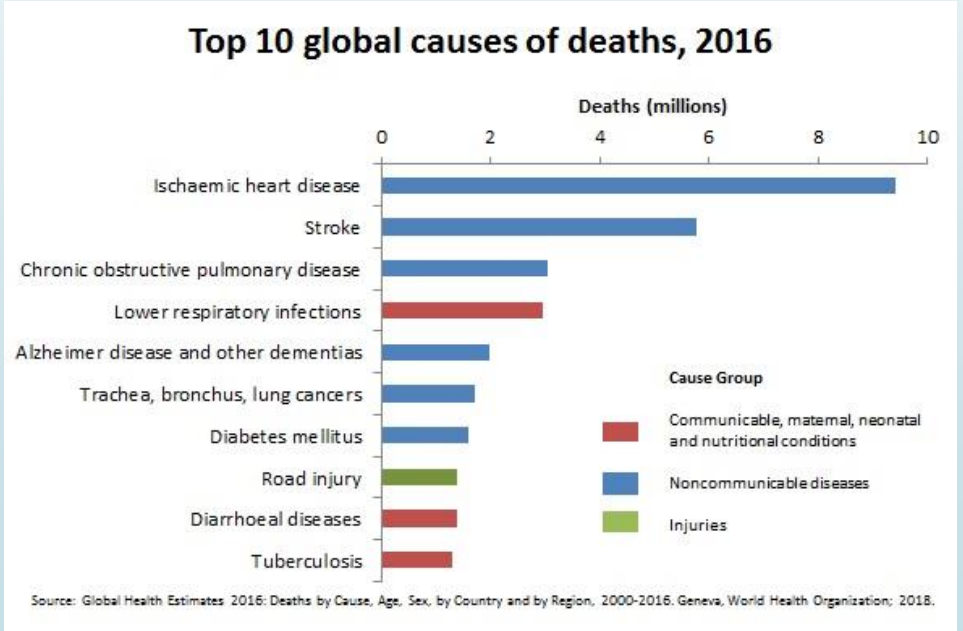
Problem Definition

Problem Definition – Road Accidents

- Around the world, more than 1.3 million people die every year in car crashes (including about 1,500 children under the age of 15) while another 50 million people are injured. **[1]**

- In fact, car crashes are the 8th leading cause of death in the world in 2016 **[2]**.





Statistic by the World Health Organization shows that road accidents are 8th leading cause of death around the world in 2016 after being the 10th in 2000!

Problem Causes

Complex real-life situations

- **Lighting conditions** are changeable
- **The presence of other objects** in the scene
- **Visibility angle** and position in the image.
- **The long exposure to the sunlight**, color of traffic signs often gradually fades.
- If the image is acquired from a moving car, then it often suffers from **motion blur** and car vibration.

Lighting conditions

Presence of objects

Viewing angles

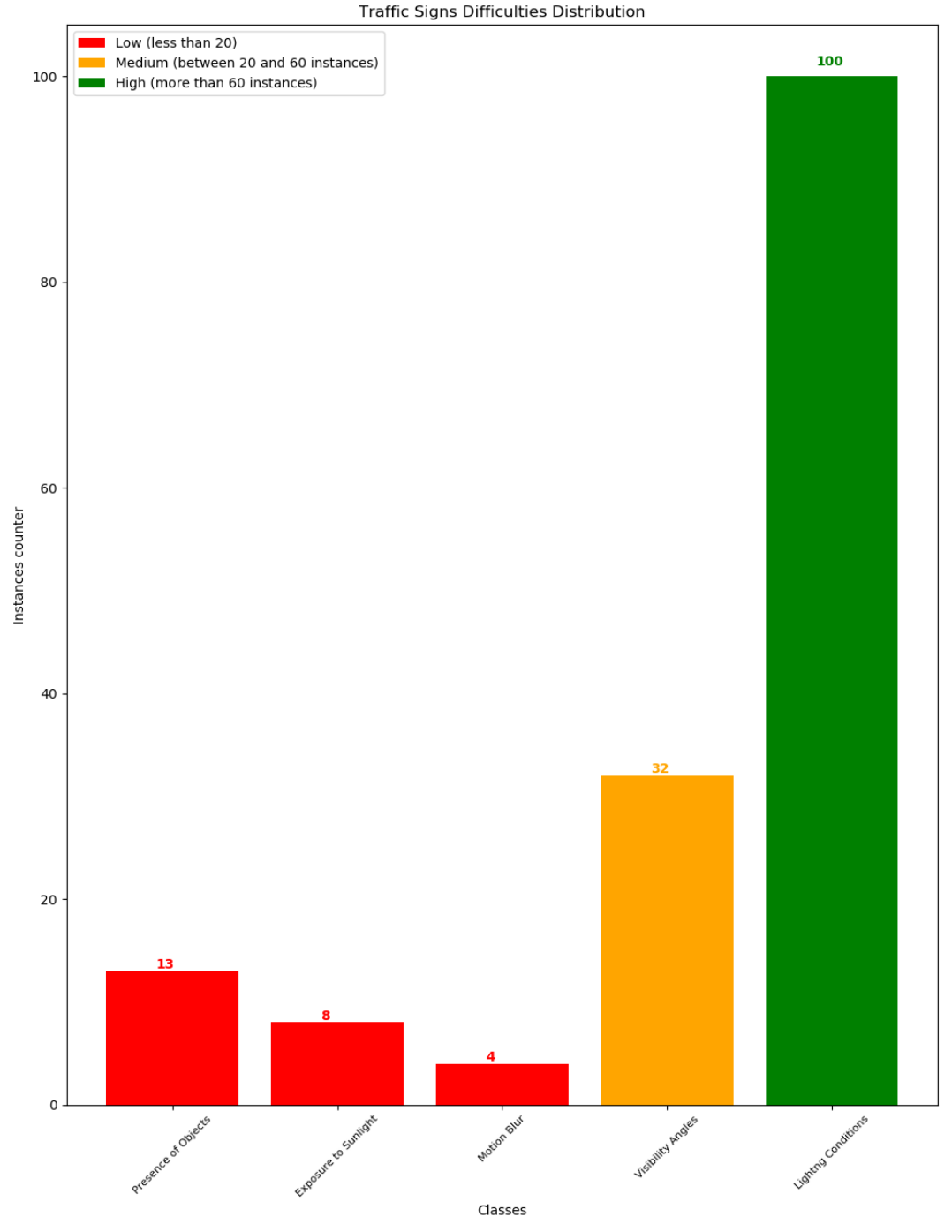
Exposure to sunlight

Motion blur

Problem Causes in GTSDB

Presence of the aforementioned traffic signs difficulties in the *German Traffic Signs Detection Benchmark* (GTSDB)

Problem	Number of instances
Lighting Conditions	100
Visibility Angles	32
Presence of Objects	13
Exposure to Sunlight	8
Motion Blur	4



Problem Solution

Traffic Signs Detection and Recognition System

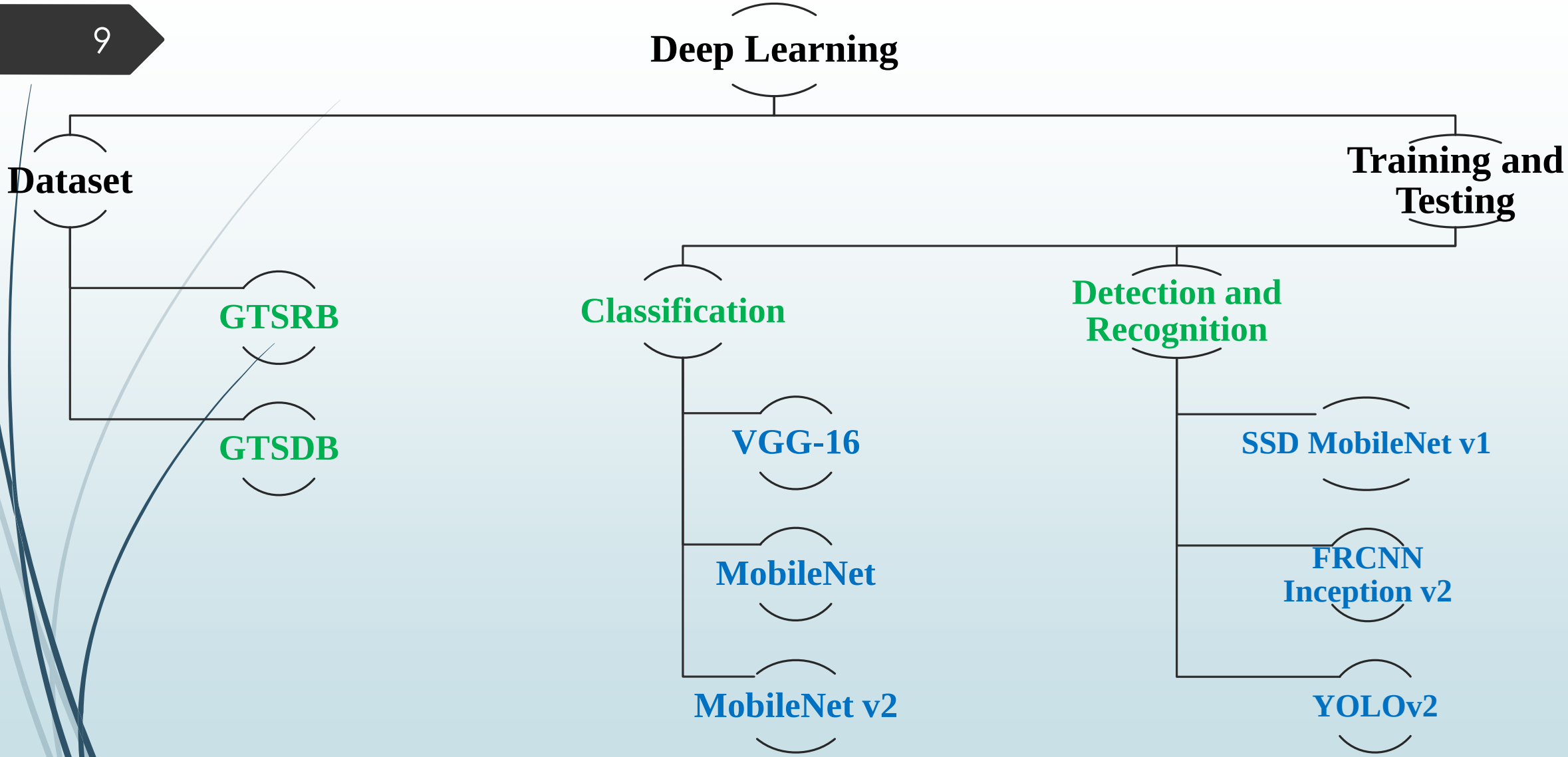
Problem Solution – TSDR System

Traffic Sign Recognition System

Advanced Driver Assistance System (ADAS) is a high-tech in-vehicle systems that are designed to increase road safety by alerting the driver of hazardous road conditions. Examples of the crucial ADAS sub-systems are Lane Departure, Collision Avoidance, and ***Traffic Signs Recognition (TSR)***.

TSR system demos are being worked on now [3]

Our TSR system (v1) in action


Deep Learning
Dataset
Training and Testing
GTSRB
GTSDB
Classification
Detection and Recognition
VGG-16
MobileNet
MobileNet v2
SSD MobileNet v1
FRCNN Inception v2
YOLOv2

Deep Learning

Datasets

Deep Learning – Datasets

Datasets

GTSDB [4]

German Traffic Signs Detection Benchmark

GTSDB dataset (900 images)

GTSRB [5]

German Traffic Signs Recognition Benchmark

GTSRB dataset (52,000 images)

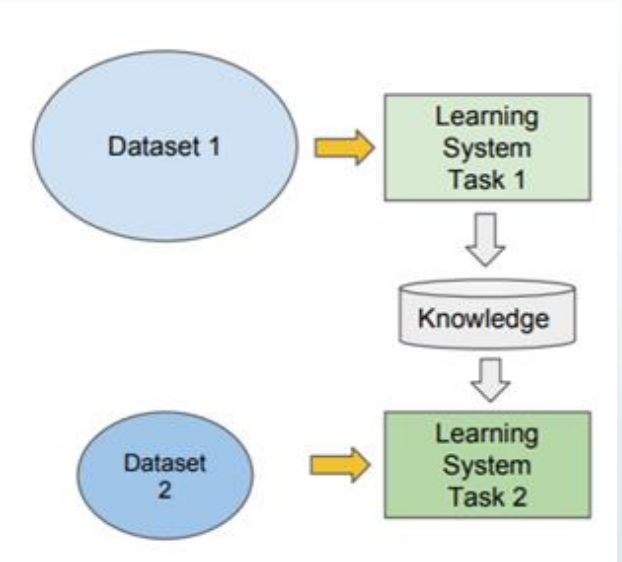
Note the huge difference in the numbers because it will play a big role later on

Deep Learning

Deep learning – Transfer Learning

Transfer Learning

It is the improvement of learning in a new task through the transfer of knowledge from a related task that has already been learned.



A. Classification

- ***VGG-16***
- ***MobileNet v1***
- ***MobileNet v2***

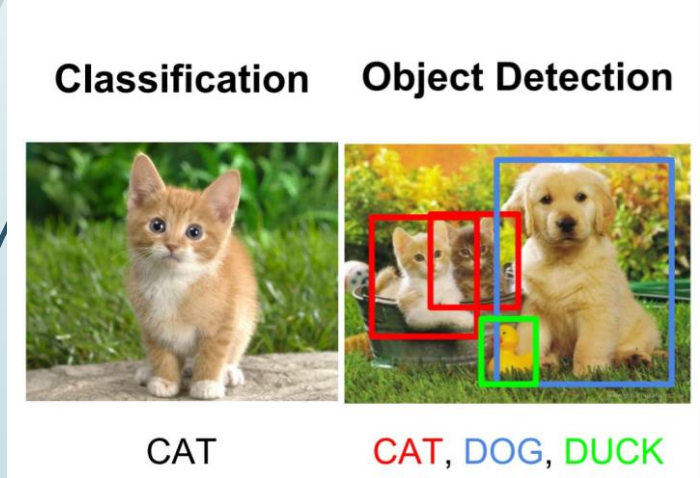
B. Detection and Recognition

- ***SSD MobileNet v1***
- ***Faster RCNN Inception v2***
- ***Tiny-YOLOv2***

We had talked about the classification portion in details during the first term exam, so we will go directly to the Detection and Recognition portion.

Real-time Detection

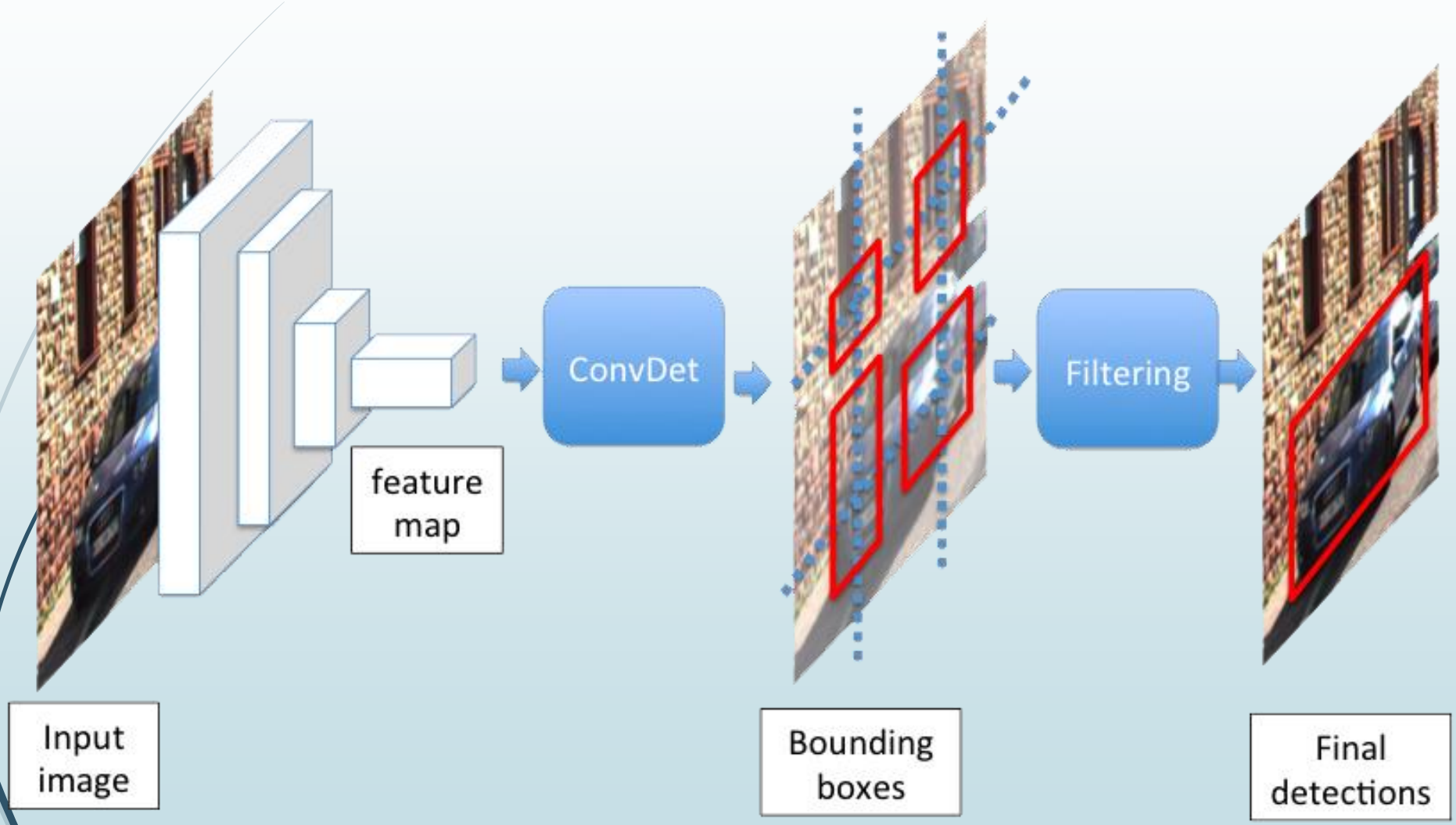
*Real-time object detection is a much more difficult problem than image classification; the model has to first find the object (**detection**), trace it (**tracking**) and finally classify it (**recognition**).*



Classification vs Detection

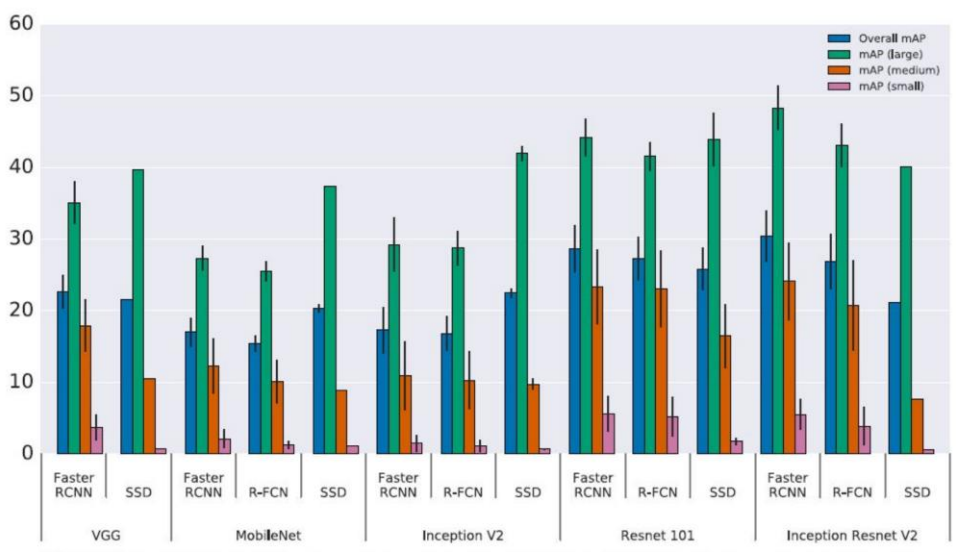
Real-time Traffic Signs Detection and Recognition

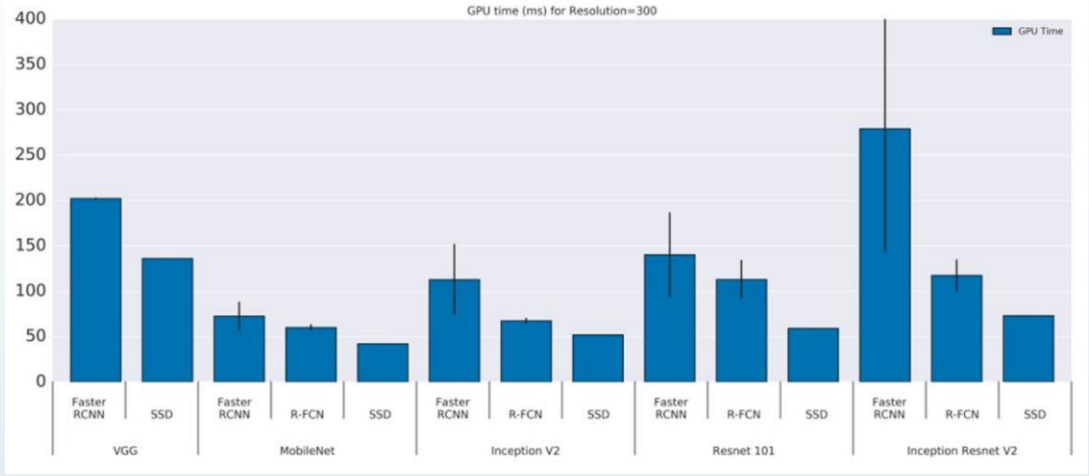
System Design



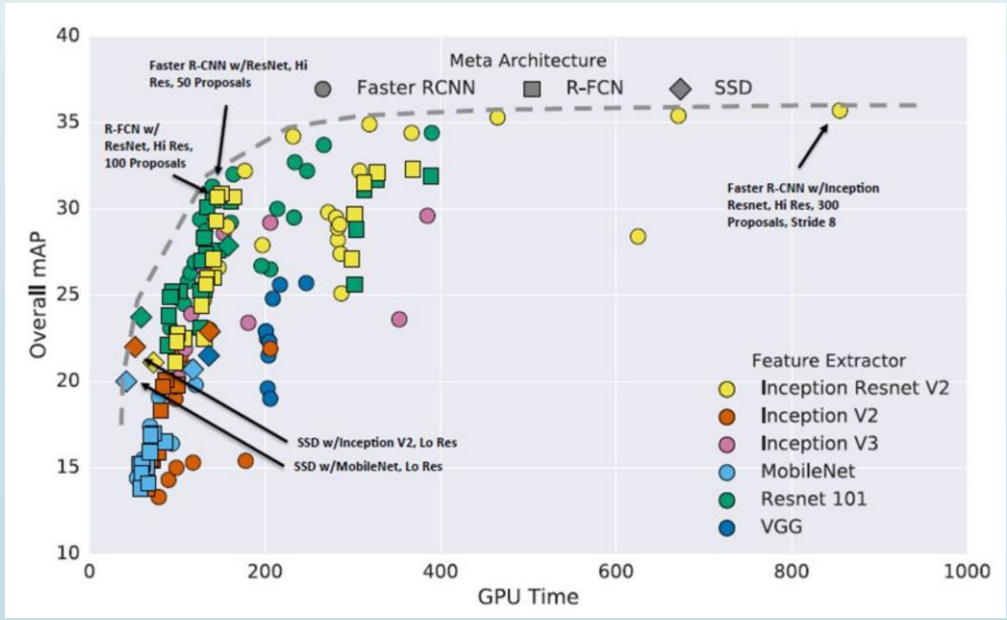
How an Object Detection model predicts the Region of Interests and classifies objects

Deep Learning – Models





Parameters/model	SSD	F-RCNN
Accuracy	Less accurate	More accurate
Speed	Faster	Slower



Deep Learning

Detection and Recognition Training on SSD MobileNet

Parameters

SSD MobileNet v1 Training

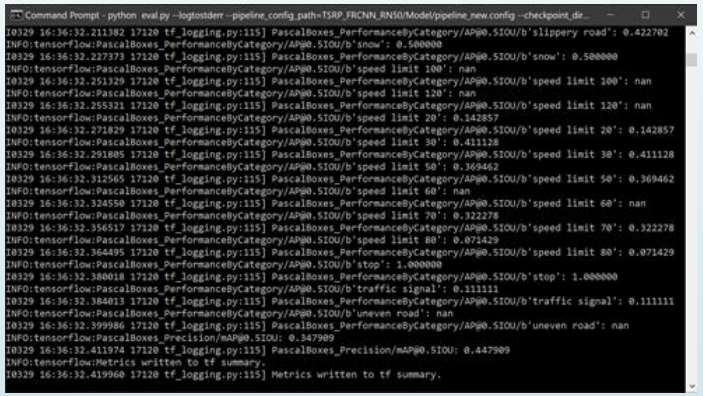
Model	SSD MobileNet v1
Number of epochs	20k
Initial learning rate	0.0002
Learning rate decay	-
Data augmentation	Random grey scale Random 25% crop Random Vertical flip*
Number of classes	**43**
Number of training samples	**800**
Architecture Modification	**Resizer = 300x300** **Softmax classification (43)**
Total loss	1.5

Deep Learning

Detection and Recognition Testing on SSD MobileNet

Results

SSD MobileNet v1 Testing



Accuracy: 45%

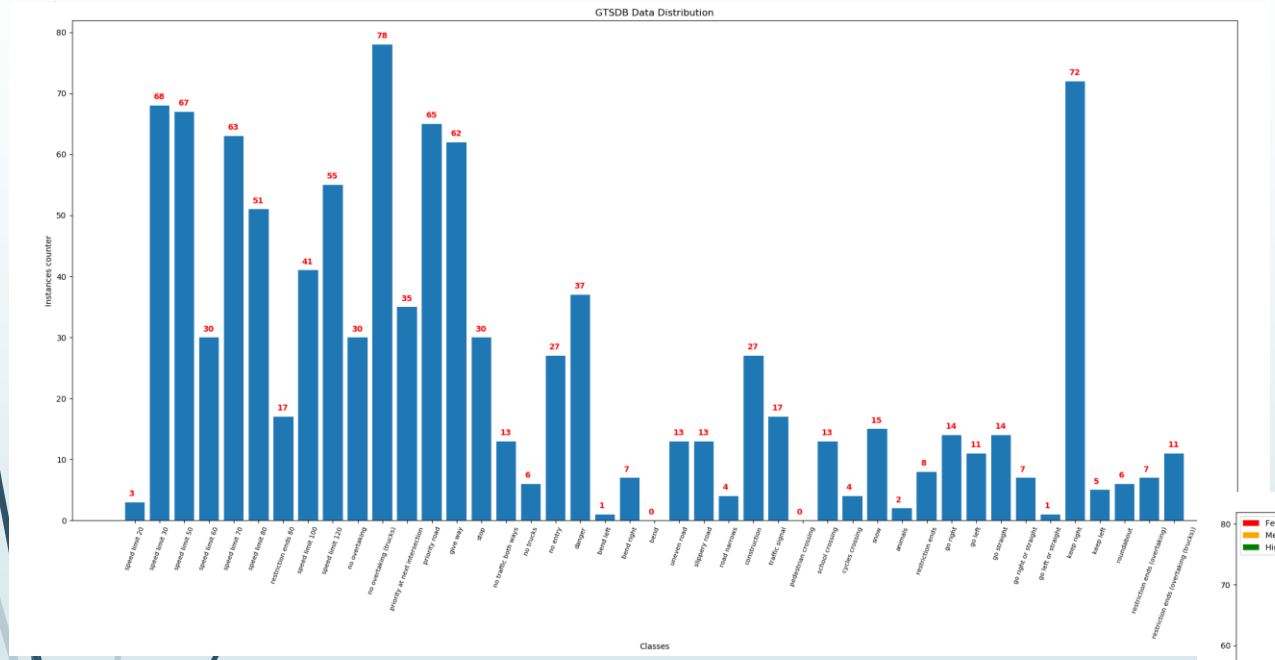
Exploratory Data Analysis

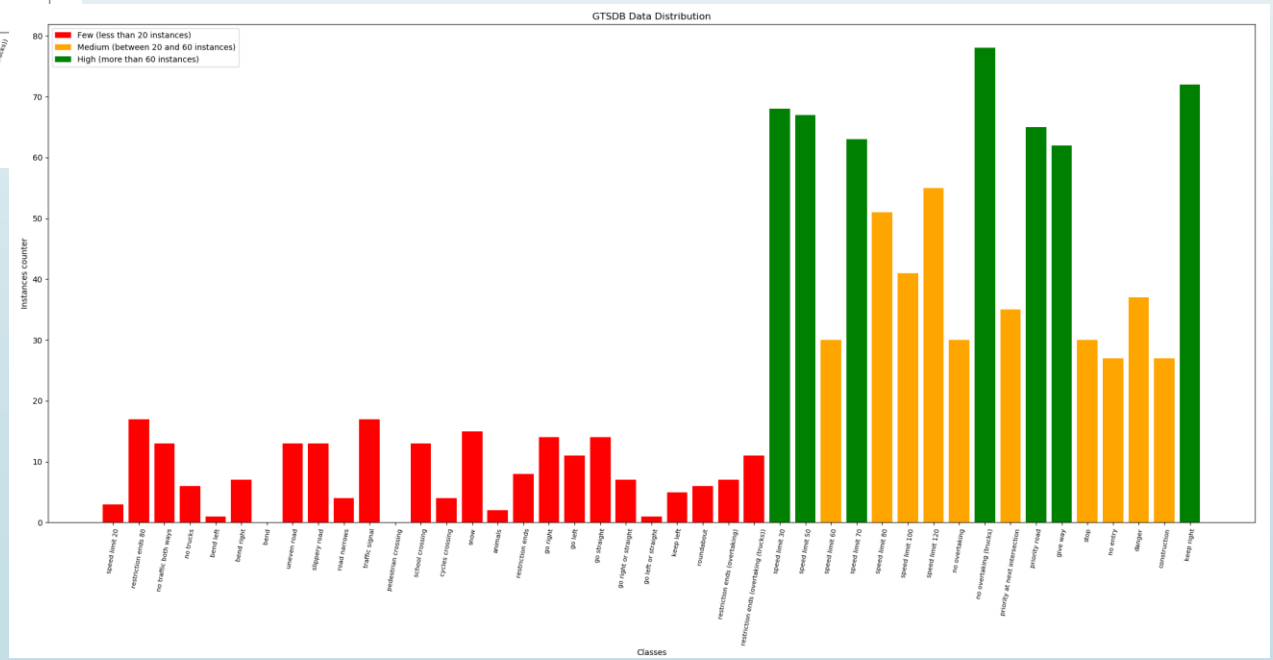
SSD MobileNet v1 Testing

What caused such low accuracy?

Let's analyze and visualize the dataset and see what went wrong…

Data Visualization for GTSDB

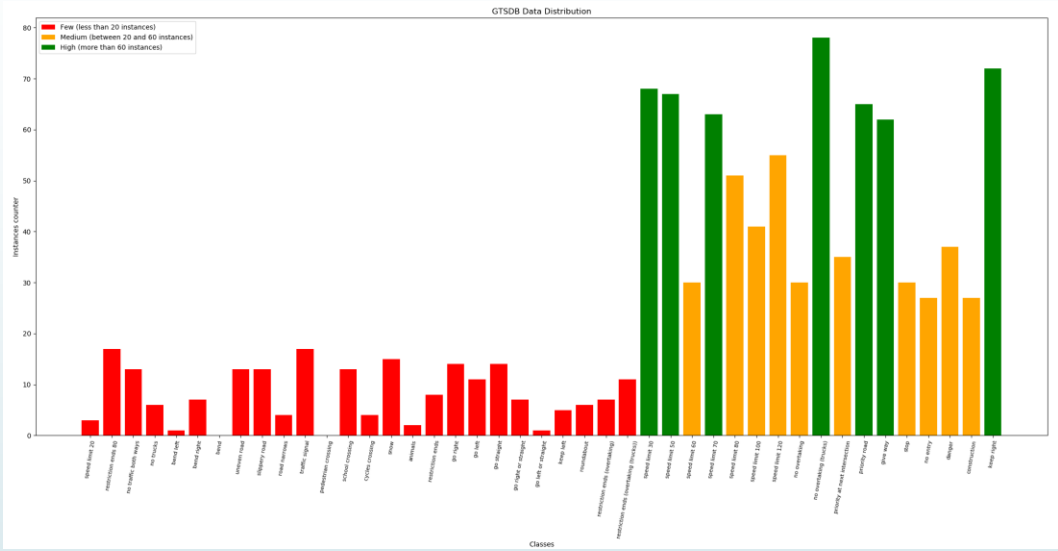


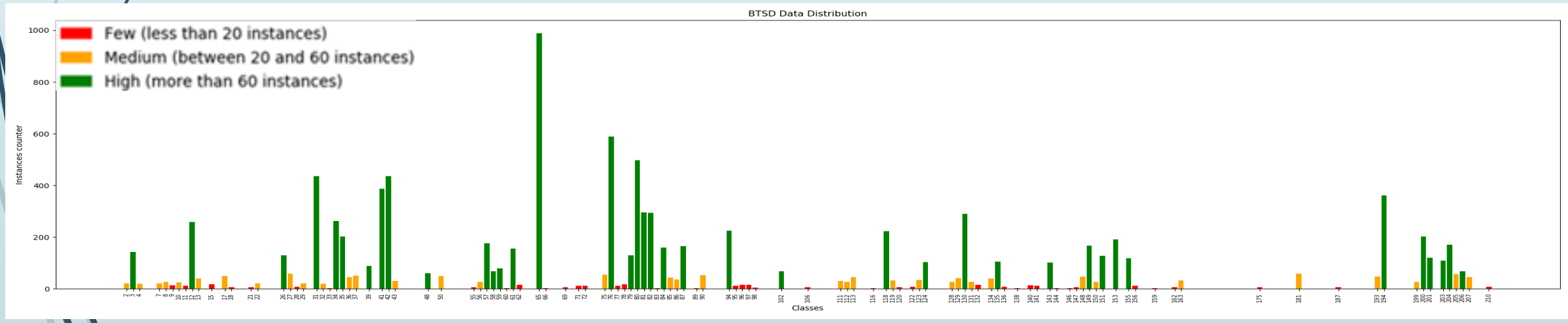


Data Analysis – GTSDB vs BTSD

Notice the enormous difference between the green bars (number of classes that have more 60 instances) in the GTSDB against that of the Belgium Traffic Signs Dataset (BTSD) [6].

Although the BTSD is found to be much better for our task, we unfortunately could not use it because it's *50GB*!





Solutions

Two possible solutions:

- **Use F-RCNN instead of SSD since it is more accurate when it comes to small objects***

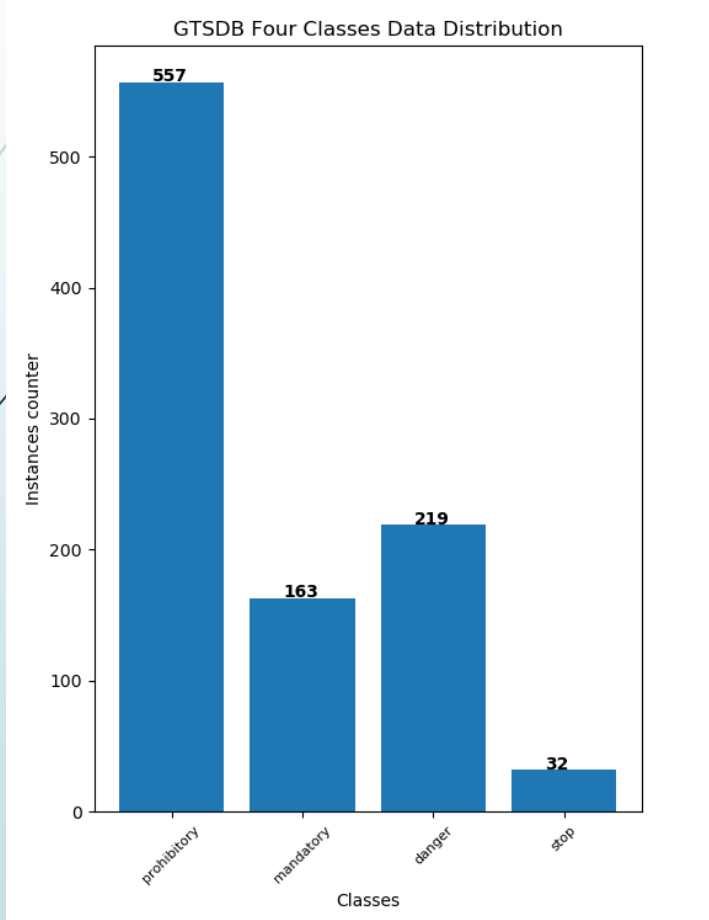
Parameters/model	SSD	F-RCNN
Accuracy	Less accurate	**More accurate**
Speed	Faster	**Slower**

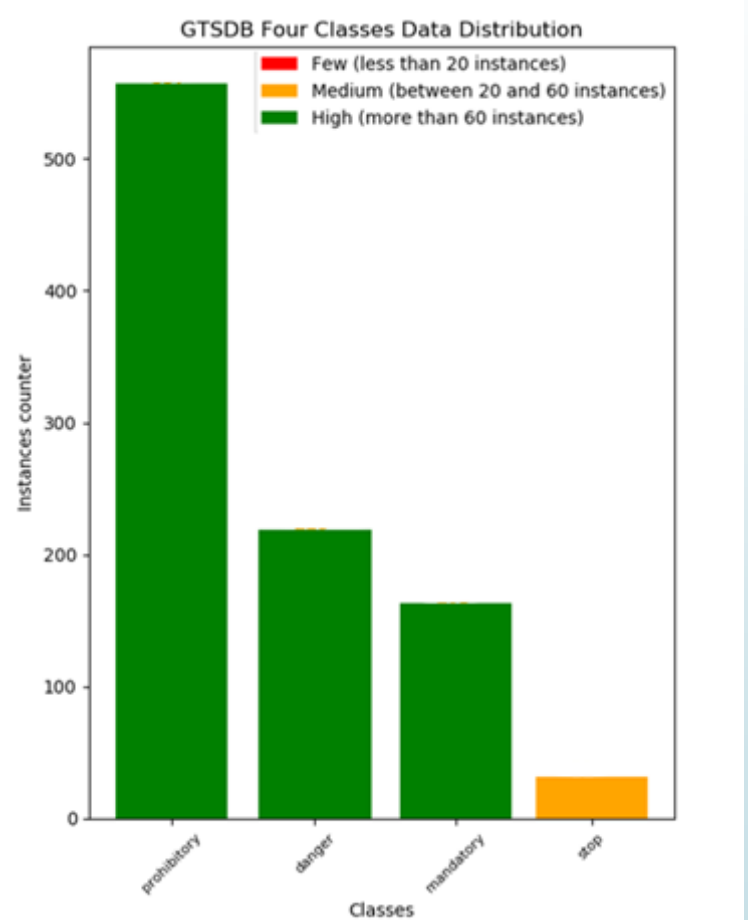
- **Instead of training 800 images on 43 classes, we'd train 900 images on the three super classes and the Stop class**
  - ***Prohibitory***
  - ***Mandatory***
  - ***Danger***
  - ***Stop***

Category	Shape	Color	Example
Prohibitory	Circular	Red, Blue, White & Black	
Mandatory	Rectangular & Circular	Blue, White & Black	
Danger	Triangular	Red, White & Black	

*https://towardsdatascience.com/understanding-ssd-multibox-real-time-object-detection-in-deep-learning-495ef744fab

Data Analysis for 4 Super-classes of GTSDB





Deep Learning

Detection and Recognition Training on F-RCNN Incv2

Parameters

FRCNN Inception v2 Training Parameters

Model	SSD MobileNet v1	F-RCNN Inception v2
Number of epochs	20k	**8.5k**
Initial learning rate	0.0002	**0.0002**
Learning rate decay	-	**1e-3 every 3000 steps**
Data augmentation	Random grey scale Random 25% crop Random Vertical flip*	**Random grey scale** **Random 25% crop** **Random vertical flip** **Random horizontal flip**
Number of classes	43	**4**
Number of training samples	800	**900**
Architecture Modification	**Resizer = 300x300** **Softmax classification (43)**	**Resizer: fixed aspect ratio** **IoU threshold: 0.7** **NMS: 0.7** **L2 Regularization** **Batch Normalization** **Drop out (30%)** **Softmax classification (4)**
Total loss	1.5	**0.4**

Comparison between the 1st aforementioned model (SSD MobileNet v1) and the 2nd used model (FRCNN Inception v2)

Deep Learning

Detection and Recognition Testing on F-RCNN Incv2

Results

Detection Testing (GTSDB) F-RCNN Inception v2

Accuracy: 95%

Average speed on 720p video: 25FPS

Average time on a 1080p image = 0.32sec

Testing the FRCNN Inceptionv2 model

GTSDB Competition [7]

Our model scored an accuracy higher than most of the algorithms in the GTSDB competition* and would have placed in the top 25 spot.

TEAM	METHOD	AREA UNDER CURVE	AVERAGE OVERLAP
wgy@HIT501	ELL_SVM2 (Prohibitive)	100 %	90.08 %
visics	boosted_intChn_ratio_scales (Prohibitive)	100 %	88.22 %
LITS1	intColorShapeAppearance (Prohibitive)	100 %	86.91 %
glc12125	robok (Prohibitive)	100 %	85.57 %
DSL_clusters	ToneMap+AcfChunks (Prohibitive)	100 %	83.56 %
qichang.hu@adelaide.edu.au	sp_Cov_P (Prohibitive)	100 %	82.03 %
huqc@msn.com	spCov+chns_P (Prohibitive)	99.99 %	82.62 %
shawn_pan	intChn_50scales (Prohibitive)	99.98 %	88.99 %
BolognaCVLab	MSER-1+HOG-2+SVM+Heights+TL (Prohibitive)	99.98 %	84.95 %
BolognaCVLab	MSER-1+HOG+SVM+Heights+TL (Prohibitive)	99.97 %	84.37 %
qcom	VSSD3 (Prohibitive)	99.89 %	91.93 %
wgy@HIT501	HOG_LDA_SVM (Prohibitive)	99.78 %	80.81 %
visics	boosted_intChn_7ops (Prohibitive)	99.12 %	88.88 %
huqc@msn.com	First try (Prohibitive)	99 %	82.54 %
s_trafficsign	DL2 (prohibitory) (Prohibitive)	99 %	86.34 %
wgy@HIT501	ELL_SVM (Prohibitive)	98.99 %	90.17 %
hb@hit501	hb_pro_0.8_0.6_13 (Prohibitive)	98.98 %	79.6 %
LITS1	hog+svm1 (Prohibitive)	98.97 %	88.49 %
VJ+Filter	VJ+LBP+HSV+colorSIFT (Prohibitive)	98.96 %	86.33 %
exp	Multi-features filter (Prohibitive)	98.96 %	86.33 %
exp	exp-multi-pro (Prohibitive)	98.91 %	86.59 %
VJ+Filter	VJ+LBP+HSV (Prohibitive)	98.87 %	86.34 %
NII-UIT	light-colorSIFT_V2 (Prohibitive)	98.11 %	81.71 %
genius07	vgg_detection (Prohibitive)	98 %	90.55 %
VJ+Filter	VJ+LBP+HSV+SIFT (Prohibitive)	97.96 %	86.33 %
VJ+Filter	test (Prohibitive)	97.94 %	86.58 %
LITS1	hog+svm (Prohibitive)	97.73 %	87.8 %
huqc@msn.com	Second try (Prohibitive)	97.02 %	83.02 %
temp	frir (Prohibitive)	97 %	91.79 %
s_trafficsign	DL (prohibitory) (Prohibitive)	97 %	86.33 %
BolognaCVLab	MSER+HOG+SVM+Heights+TL (Prohibitive)	96.01 %	80.45 %
bkmw	haar_upscale (Prohibitive)	95.82 %	84.48 %
exp	Bag of visual word (Prohibitive)	95.23 %	87.64 %
aboltabol	aboltabol3 (Prohibitive)	95 %	92.49 %
NII-UIT	denseSIFT (Prohibitive)	94.65 %	87.48 %
ducluu	lbp_filter_16_20_wsize (Prohibitive)	93.12 %	85.37 %
exp	Bag of visual words (Prohibitive)	92.34 %	87.55 %
ShapeSaliency	ShapeSaliency+Color (Prohibitive)	92.16 %	85.49 %
Test	12_STAGES_UPSIZE (Prohibitive)	91.82 %	82.91 %
Test	test_all (Prohibitive)	91.49 %	82.94 %
code monkey	11D3 prohibitory (Prohibitive)	91.4 %	88.1 %
Test	12 STAGES_UPSIZE (Prohibitive)	91.15 %	82.87 %
hoqmawla	hoqmawla3 (Prohibitive)	91.14 %	83.73 %
INI-RTCV	Viola-Jones (Prohibitive)	90.81 %	87.85 %
code monkey	Modified SSD (Prohibitive)	88.8 %	90.27 %
mraouf.cairo.4	Prohibitory_13_21__ (Prohibitive)	88.6 %	93.28 %
shawn_pan	intChn_MultiScale (Prohibitive)	88.35 %	87.94 %
huqc@msn.com	Combine1 (Prohibitive)	88.11 %	83.66 %
mraouf.cairo.3	Prohibitory_Testing_13_20 (Prohibitive)	87.64 %	93.16 %

http://benchmark.ini.rub.de/?section=gtsdb&subsection=results

Our model achieved an accuracy of 95% placing it in the top-25 spots

Deep Learning

Detection and Recognition Training on Tiny-YOLOv2

Parameters

Tiny-YOLOv2 Training Parameters

Model	SSD MobileNet v1	F-RCNN Inception v2	Tiny YOLOv2
Number of epochs	20k	8.5k	**1.5k**
Initial learning rate	0.0002	0.0002	**0.0002**
Learning rate decay	-	1e-3 every 3000 steps	**-**
Data augmentation	Random grey scale Random 25% crop Random Vertical flip*	Random grey scale Random 25% crop Random Vertical flip Random horizontal flip	**Random grey scale** **Random Vertical flip** **Random horizontal flip**
Number of classes	43	4	**4**
Number of training samples	800	900	**900**
Total loss	1.5	0.4	**2.8**

Comparison between the first two models (SSD MobileNetv1 and FRCNN Inception v2) and Tiny TOLOv2

Deep Learning

Detection and Recognition Testing on Tiny-YOLOv2

Results

Detection Testing (GTSDB) – Tiny-YOLOv2

Accuracy: 73%

Average speed on 720p video: 70FPS

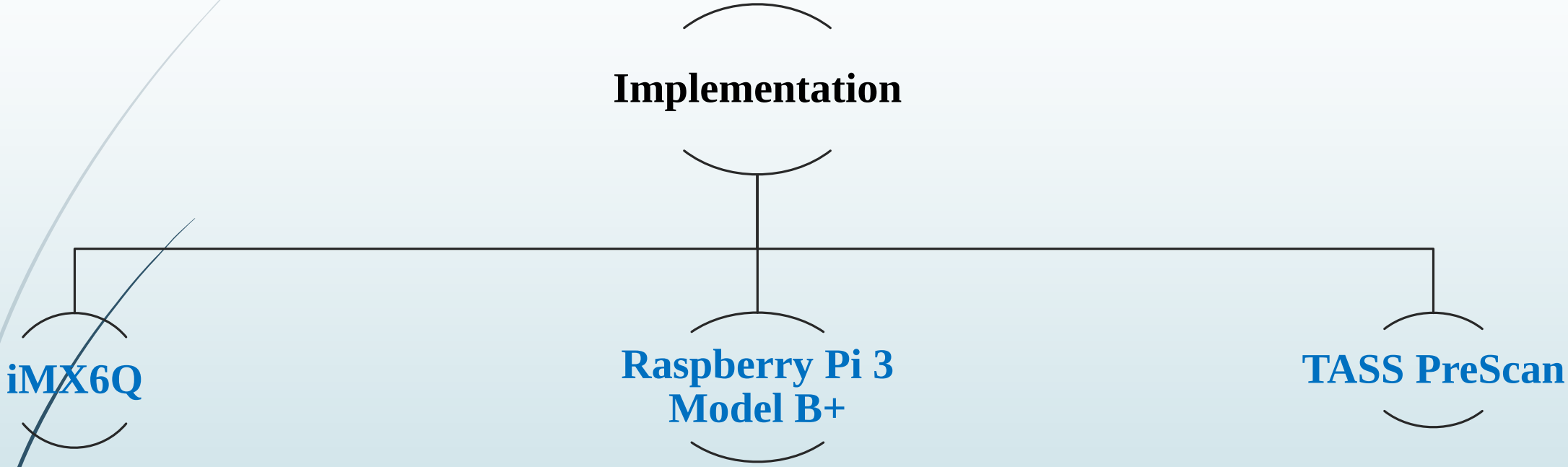
Average time on a 1080p image = 0.02sec

Testing the YOLOv2 model

Conclusion

Model	SSD MobileNet v1	F-RCNN Inception v2	Tiny YOLOv2
Number of epochs	20k	8.5k	1.5k
Initial learning rate	0.0002	0.0002	0.0002
Learning rate decay	-	1e-3 every 3000 steps	-
Data augmentation	Random grey scale Random 25% crop Random Vertical flip*	Random grey scale Random 25% crop Random Vertical flip Random horizontal flip	Random grey scale Random Vertical flip Random horizontal flip
Number of classes	**43**	**4**	**4**
Number of training samples	**800**	**900**	**900**
Total loss	1.5	**0.4**	2.8
Accuracy	45%	**95%**	73%
Speed per image (sec)	0.15	0.32	**0.02**
Speed (FPS)	32FPS	25FPS	**70FPS**

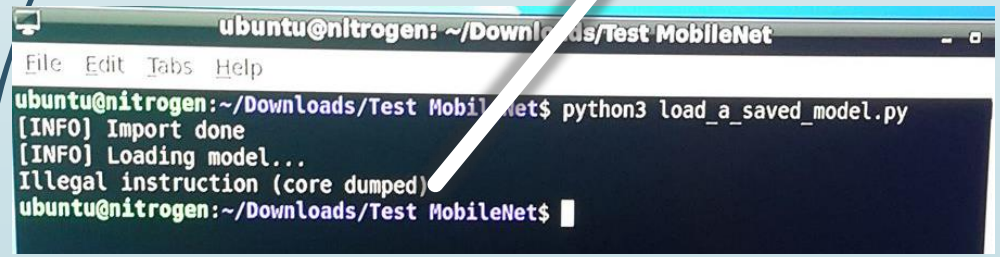
Comparison between all the models' results obtained while testing on the host PC


Implementation
iMX6Q
Raspberry Pi 3 Model B+
TASS PreScan

Embedded System
Implementation
iMX6Q

Testing on the iMX6Q

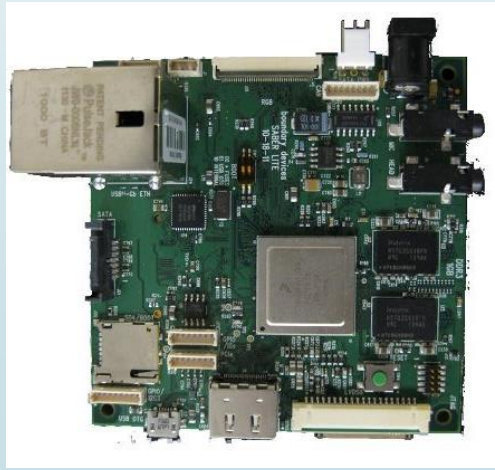
Tensorflow was too heavy for the board's CPU that it crashed every time we tried to create a variable and initiate a session.



Error message whenever we tried loading a Tensorflow model despite how light it is

	iMX6Q
CPU	ARM Cortex-A9 (Quad-Core) 1 GHZ
RAM	1GB DDR3
GPU	Vivante GC2000

iMX6Q specs



iMX6Q

Embedded System
Implementation
Raspberry Pi 3 Model B+

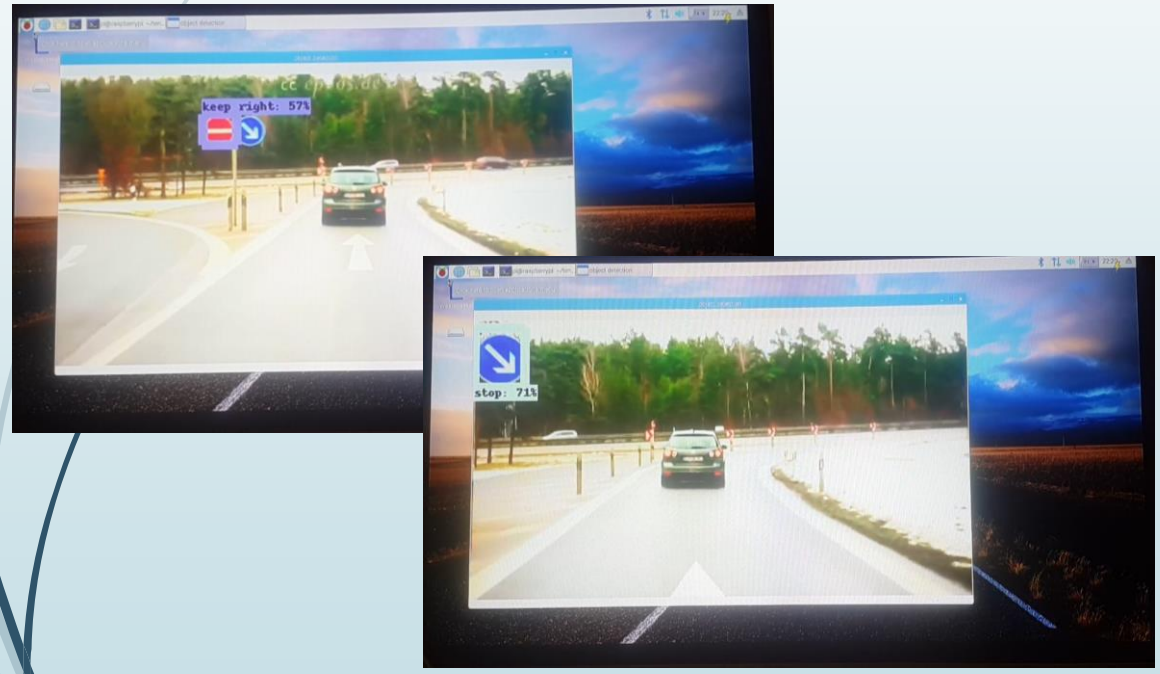
Raspberry Pi 3 Model B+ vs Host PC

	iMX6Q	RPi 3B+	Host PC
CPU	ARM Cortex-A9 (Quad-Core) 1 GHZ	**ARM Cortex-A53 (Quad-Core) 1.2GHz**	**I7 6700k Quad-core 3.2GHz**
RAM	1GB DDR3	**1GB LP DDR2**	**16GB DDR3**
GPU	Vivante GC2000	**Dual Core Video Core IV**	**Nvidia GTX 1070 6GB**

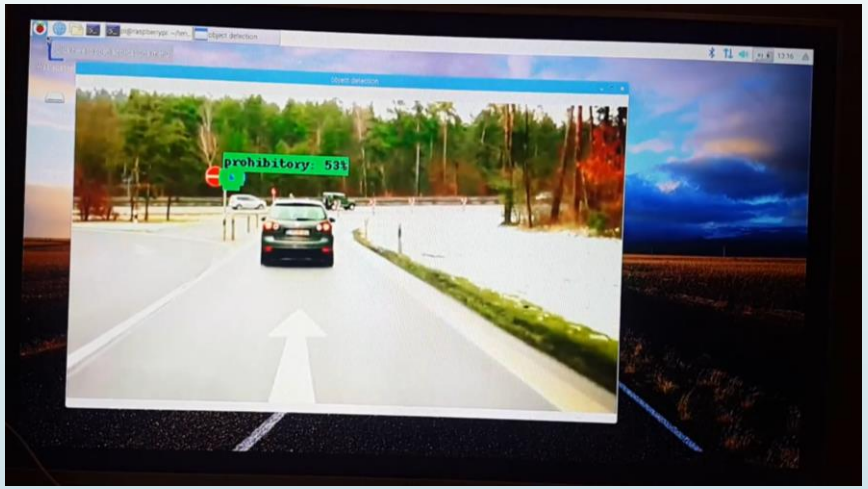
iMX6Q vs. RPi 3 Model B+ vs. Host PC specs

Testing the TSDR System on RPi 3 Model B+

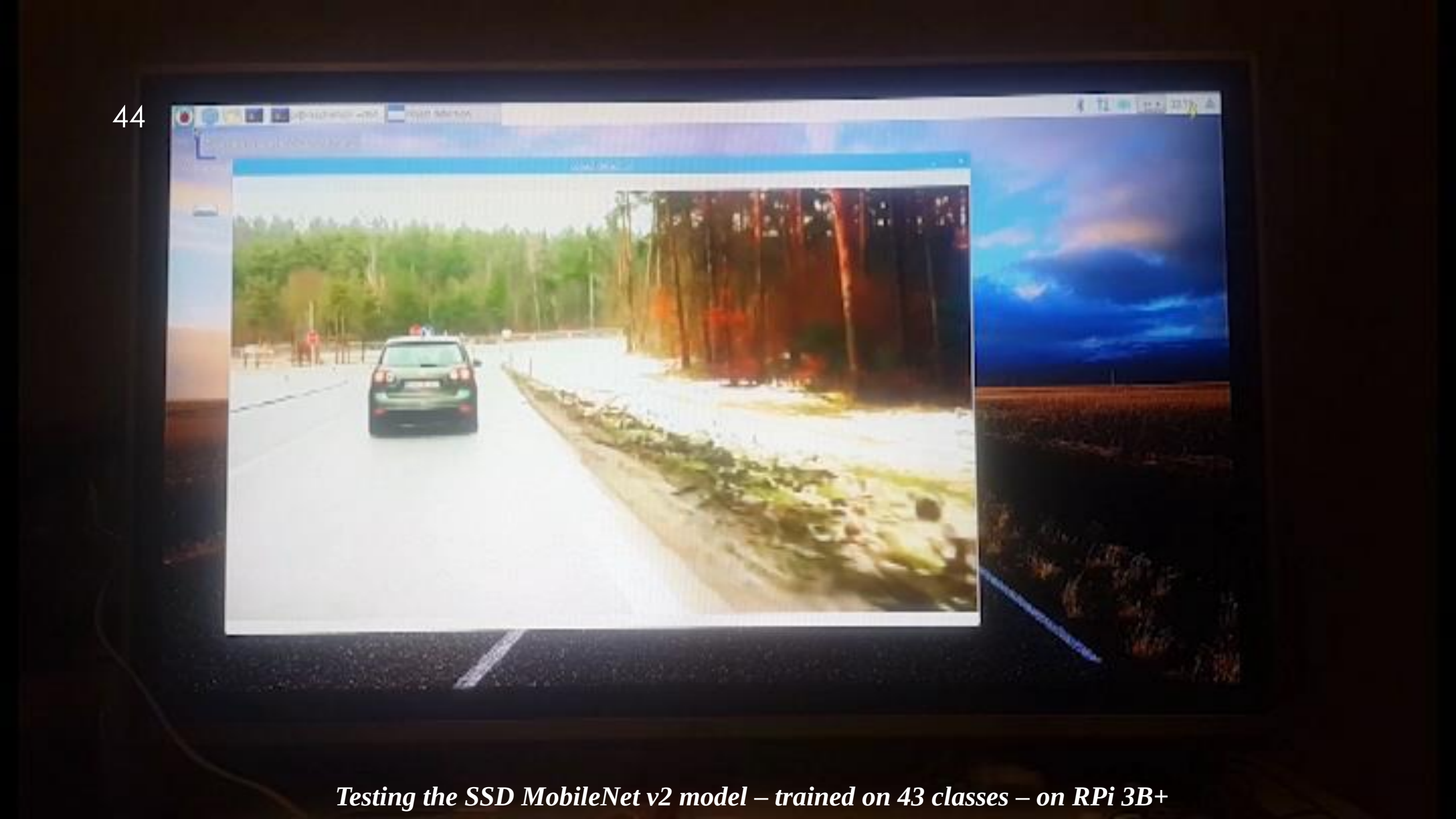
Models used: SSD MobileNetv2 (43 classes and 4 classes)
Video resolution: 640x480 (480p)
Average speed (FPS): 1FPS



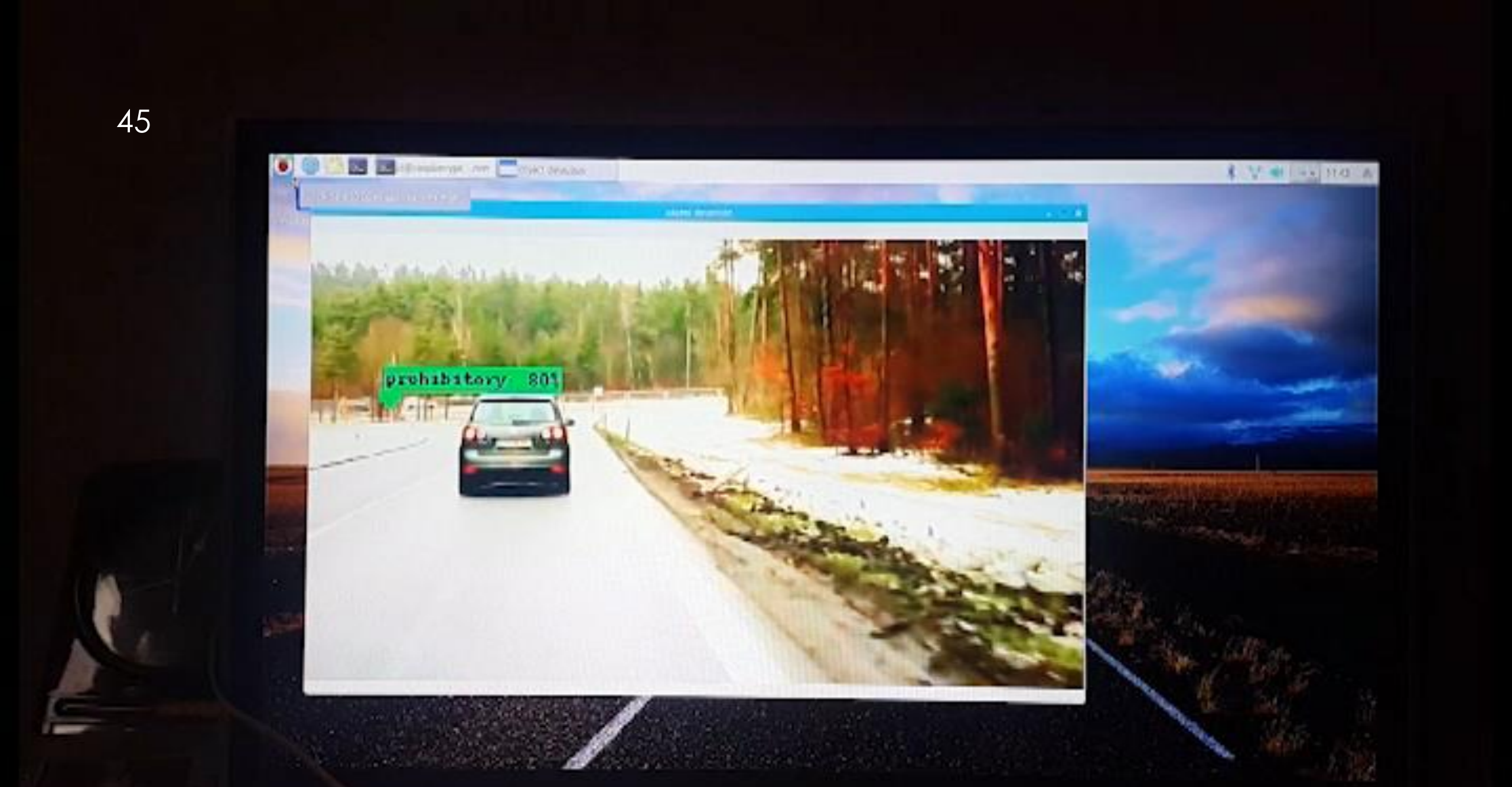
Testing the SSD MobileNet v2 model – trained on 43 classes – on RPi 3B+



Testing the SSD MobileNet v2 model – trained on 4 classes – on RPi 3B+



Testing the SSD MobileNet v2 model – trained on 43 classes – on RPi 3B+



Testing the SSD MobileNet v2 model – trained on 4 classes – on RPi 3B+

Virtual Board Implementation TASS PreScan

TASS PreScan Implementation

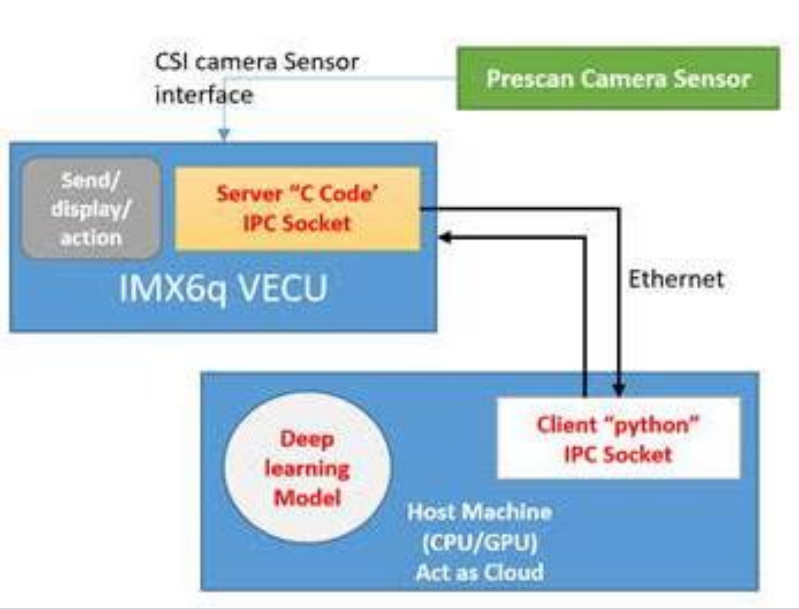
Introduction

TASS PreScan [8] is a virtual simulation for a self-driving car recently acquired by ***Siemens*** in ***2018*** with the purpose to simulate real-life scenarios that might face a self-driving car.

We've had the pleasure to implement our TSDR System in PreScan.

Process

- A ***real-time video is generated*** by ***PreScan***.
- Each ***frame*** is ***flattened*** to 1-D.
- The ***1-D frame is sent*** to our Host PC.
- ***Our algorithm*** converts the received bytes ***1-D frame into a 2D RGB frame***.
- The ***2D frame*** is then passed to the ***F-RCNN Inception v2 model***.
- The resulting ***detected frame is shown*** and ***saved*** in the Host PC.

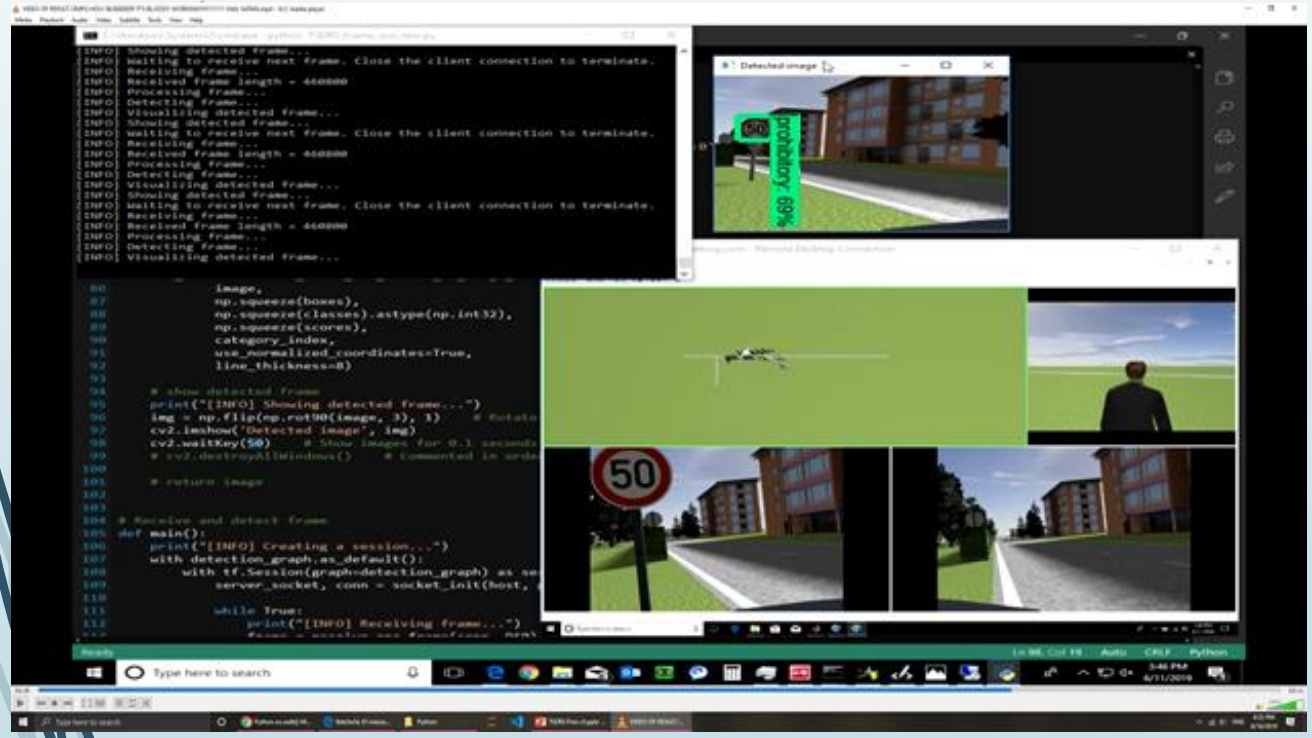


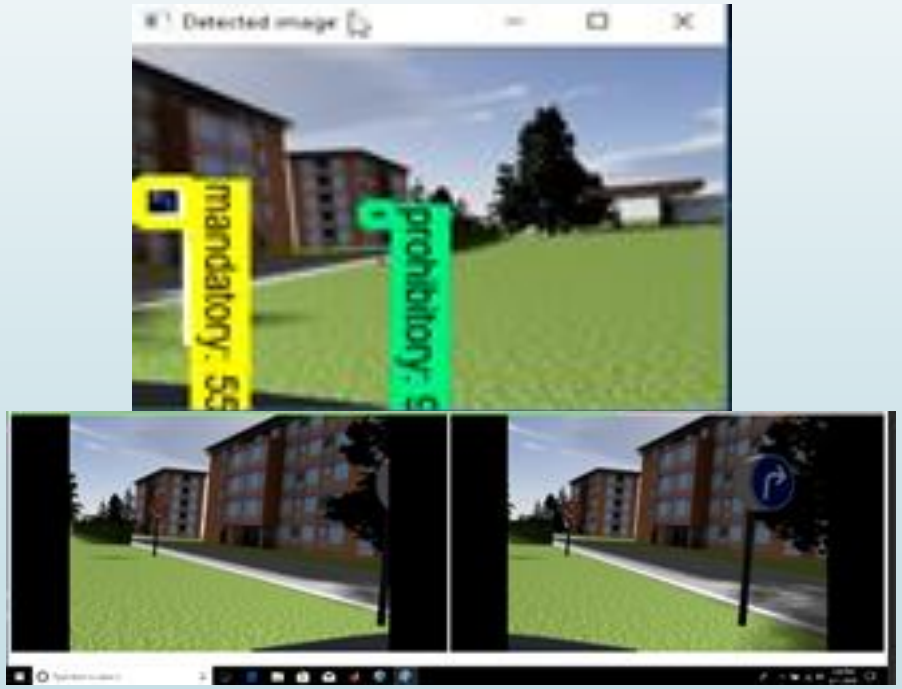
Process of detecting frames from PreScan

PreScan Implementation - Results

Model: FRCNN Inception v2

FPS: PreScan speed – 4





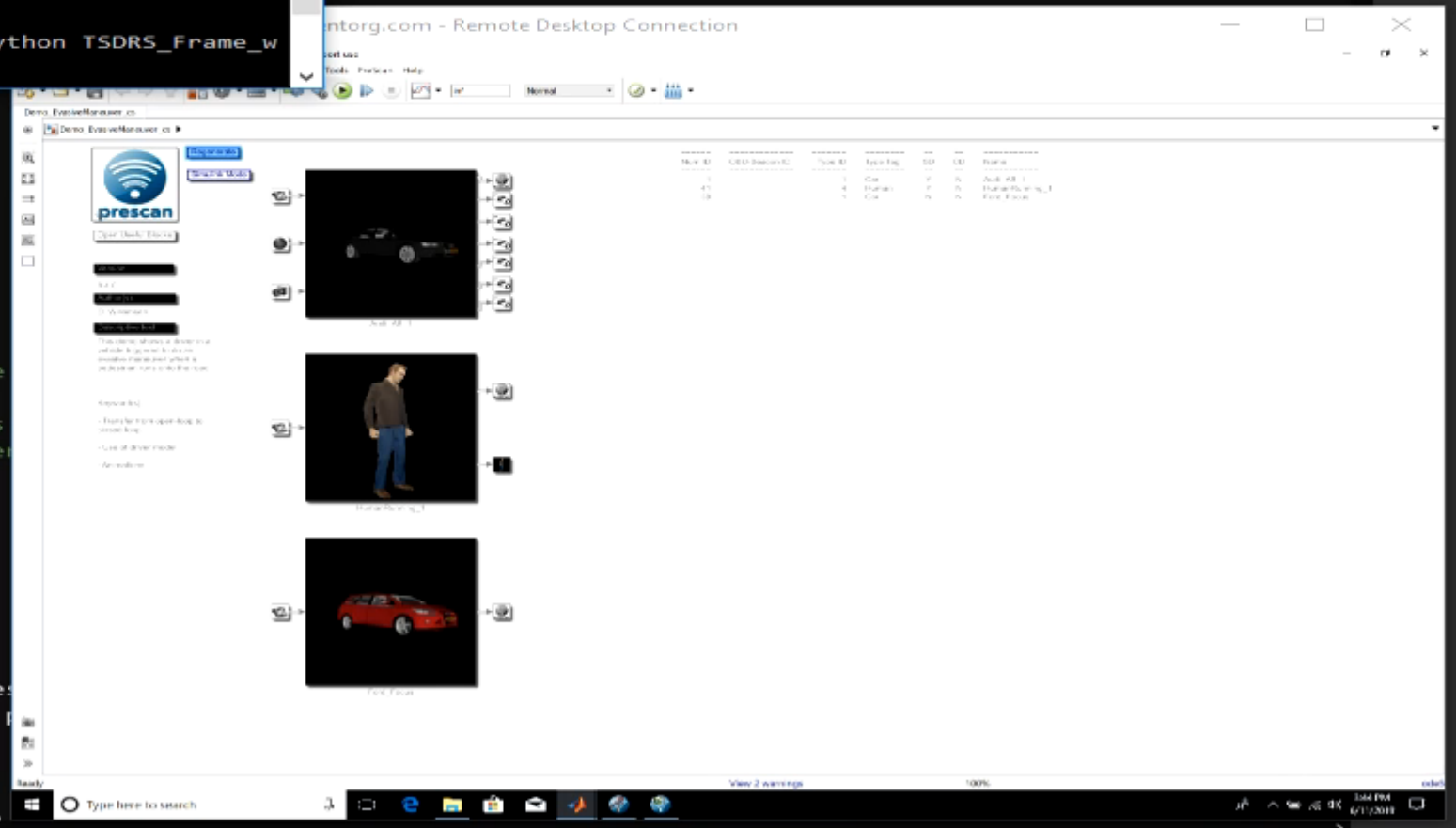
FRCNN Detecting PreScan signs

C:\Windows\System32\cmd.exe

```
[INFO] Showing detected frame...
[INFO] Waiting to receive next frame. Close the client connection to terminate.
[INFO] Receiving frame...
[INFO] Received frame length = 460800
[INFO] Processing frame...
[INFO] Detecting frame...
[INFO] Visualizing detected frame...
[INFO] Showing detected frame...
[INFO] Waiting to receive next frame. Close the client connection to terminate.
[INFO] Receiving frame...
[INFO] Received frame length = 460800
[INFO] Processing frame...
[INFO] Detecting frame...
[INFO] Visualizing detected frame...
[INFO] Showing detected frame...
[INFO] Waiting to receive next frame. Close the client connection to terminate.
[INFO] Receiving frame...
[INFO] Client disonnected!
[INFO] Connection closed!

C:\Users\markma\Desktop\New folder\research\object_detection>python TSDRS_Frame_w
in_new.py
```

```
            image,
            np.squeeze(boxes),
            np.squeeze(classes).astype(np.int32),
            np.squeeze(scores),
            category_index,
            use_normalized_coordinates=True,
            line_thickness=8)

    # show detected frame
    print("[INFO] Showing detected frame...")
    img = np.flip(np.rot90(image, 3), 1)    # Rotate
    cv2.imshow('Detected image', img)
    cv2.waitKey(50)    # Show images for 0.1 seconds
    # cv2.destroyAllWindows()    # Commented in orde

    # return image


# Receive and detect frame
def main():
    print("[INFO] Creating a session...")
    with detection_graph.as_default():
        with tf.Session(graph=detection_graph) as se
            server_socket, conn = socket_init(host,

            while True:
                print("[INFO] Receiving frame...")
```

ntorg.com - Remote Desktop Connection



Ready Ln 98, Col 19 Auto CRLF Python

Type here to search 3:44 PM 6/11/2019

Models Implementation Conclusion

Parameters	Results		
Model	**SSD MobileNet v2 (4)**	**F-RCNN Inception v2**	**Tiny YOLOv2**
Accuracy	75%	**95%**	73%
Speed per image (Host PC)	0.21	0.32	**0.02**
FPS (Host PC)	30	25FPS	**70FPS**
Speed TASS PreScan	Not tested	**TASS' speed – 4 frames**	Not tested
Speed RPi 3B+	**1FPS**	Could not be loaded	Not tested

Implementation results: SSD MobileNet v2 vs. F-RCNN Inception v2 vs YOLO v2

Our Results vs Previous TSDR Systems

Parameters	Our Results		Previous Systems	
Algorithm	**SSD MobileNet v2 (4)**	**F-RCNN Inception v2**	**HoG + SVM [9]**	**MSER + CNN [10]**
Dataset	GTSDB	GTSDB	GTSDB	GTSDB
Accuracy	75%	**95%**	88.9%	95%
Speed per image (Host PC)	200ms	324ms	-	450ms (1280x1024)
FPS (Host PC)	30	25FPS	8-12FPS	-
Speed TASS PreScan	Not tested	**TASS' speed – 4 frames**	Not tested	Not tested
Speed RPi 3B+	**1FPS**	Could not be loaded	Not tested	Not tested

Implementation results comparison – Our results vs previous results

References

[1] "ASIRT Road Accident," [Online]
Available: https://www.asirt.org/safe-travel/road-safety-facts/

[2] "World Health Organization," [Online]
Available: https://www.who.int/news-room/fact-sheets/detail/the-top-10-causes-of-death

[3] "Ford Traffic Signs Recognition," [Online]
Available: https://www.youtube.com/watch?v=BQgulv2hbuU

[4] "GTSDB," [Online]
Available: http://benchmark.ini.rub.de/?section=gtsdb&subsection=news

[5] "GTSRB," [Online]
Available: http://benchmark.ini.rub.de/?section=gtsrb&subsection=news

[6] "Belgium Traffic Signs Dataset," [Online]
Available: http://www.vision.ee.ethz.ch/~timofter/traffic_signs/

[7] "GTSDB Competition," [Online]
Available: http://benchmark.ini.rub.de/?section=gtsdb&subsection=results

[8] "TASS PreScan," [Online]
Available: https://tass.plm.automation.siemens.com/prescan

References

[9] **Chunsheng Liu, Faliang Chang, Zhenxue Chen, and Dongmei Liu, "Fast Traffic Sign Recognition via High-Contrast Region Extraction and Extended Sparse Representation," IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, VOL. 17, NO. 1, JANUARY 2016**

[10] **LUO *et al.* "Traffic sign recognition using multi-task convolutional neural network," IEEE transactions on intelligent transportation systems, vol. 19, no. 4, april 2018**

Thank You!

Backup Slides

Classification

A. Classification

Image classification is the problem in which the model is given a small image of an object and it recognizes to which class (category) this object belongs.

Traffic sign classification

Deep Learning

Classification Training

Parameters

Training Parameters

Model	MobileNet v1	MobileNet v2
Number of epochs	10	10
Initial learning rate	0.0002	0.0002
Data augmentation	• Random grey scale • Random 25% crop • Random vertical flip • Random 20-60% brightness shift • Random contrast shift	• Random grey scale • Random 25% crop • Random vertical flip • Random 20-60% brightness shift • Random contrast shift
Number of classes	43	43
Number of training samples	39, 200	39, 200
Number of validation samples	12, 800	12, 800
Total loss achieved	0.54	***0.04***
Training accuracy	82%	***99.8%***

Deep Learning

Classification Testing

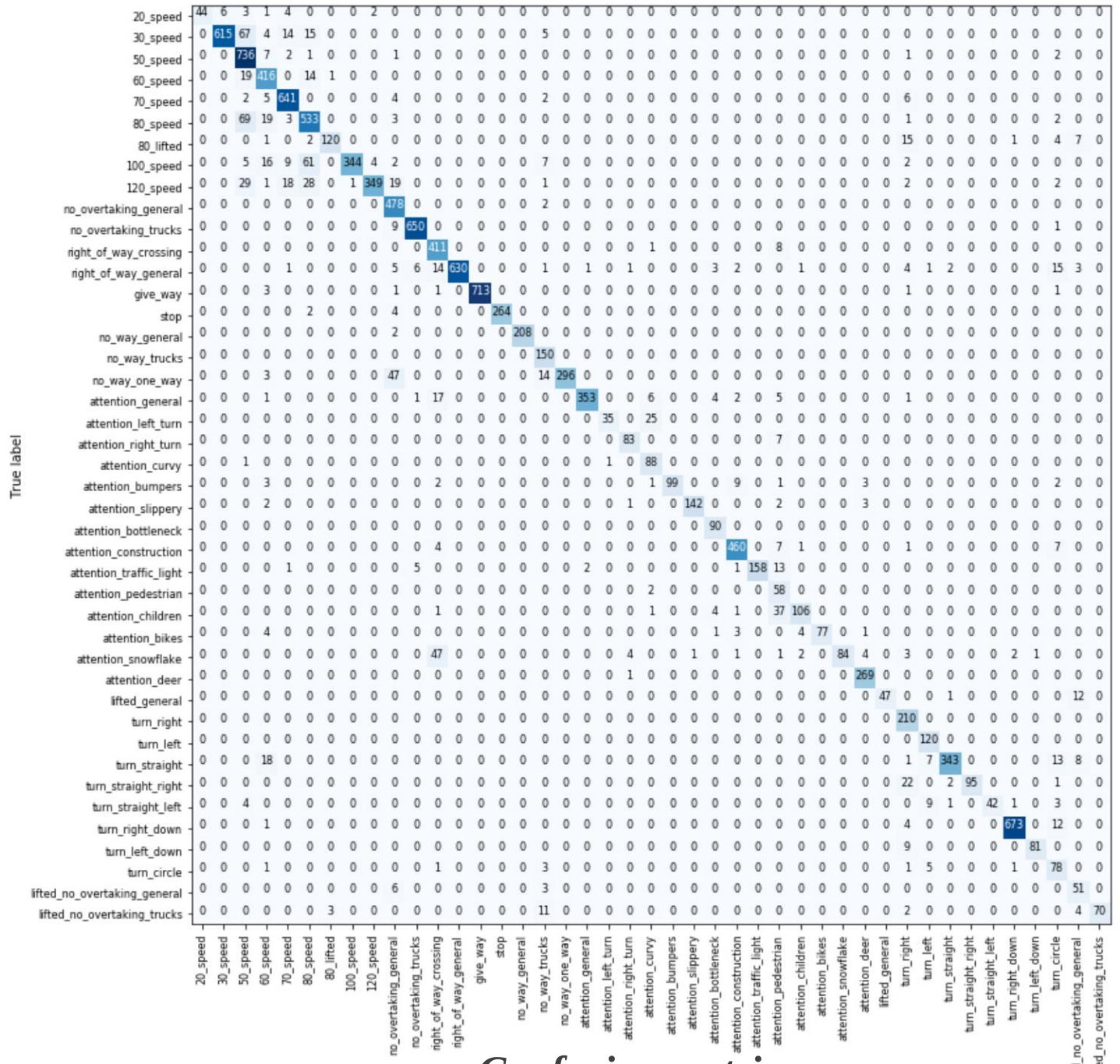
Results

Testing

MobileNet v2

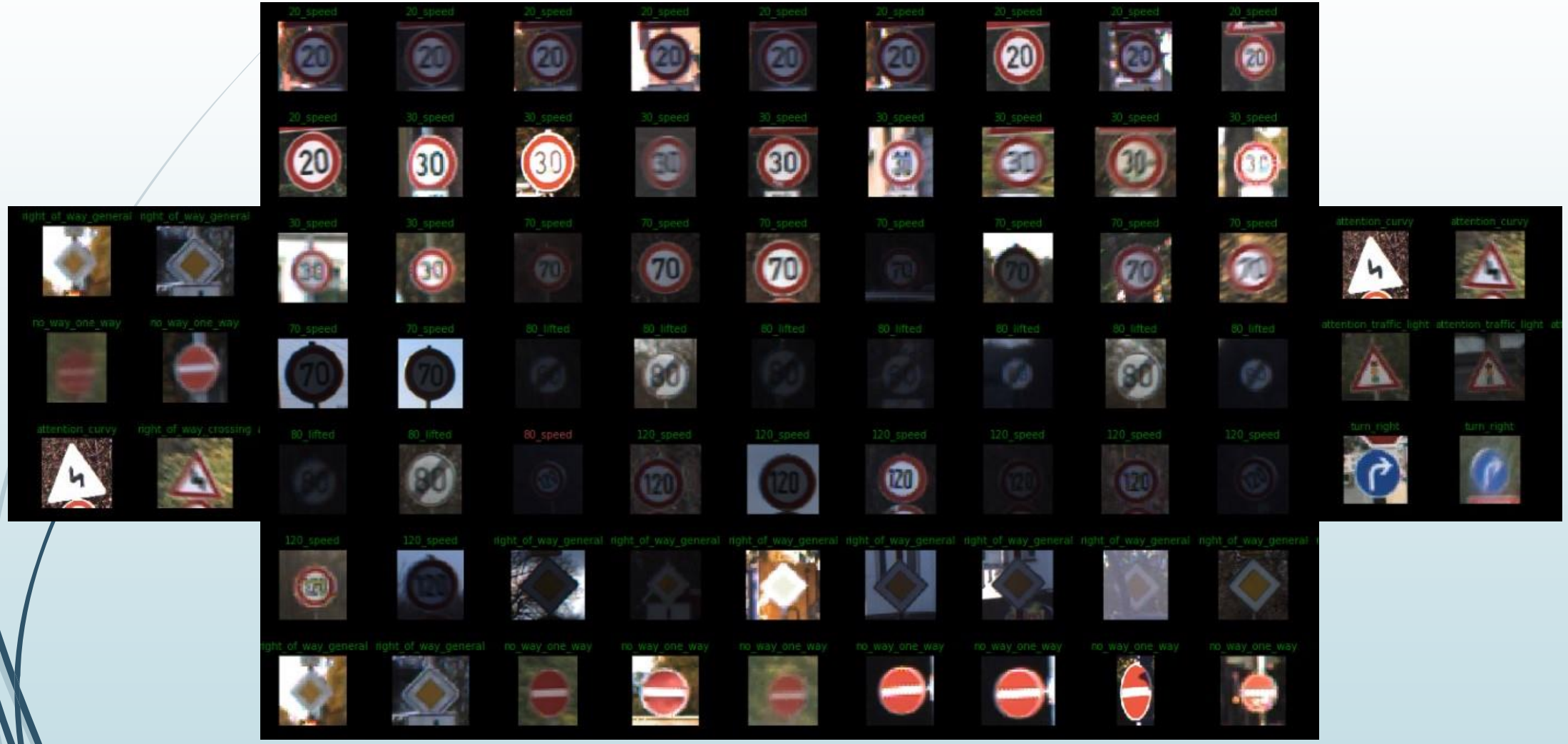
Loss = 0.0264

Accuracy = 99.832%



Confusion matrix

Testing MobileNetv2



Green = Correct prediction

Some testing results

Red = Incorrect prediction

GTSRB Competition [5]

Our model scored an accuracy higher than that of the winning algorithm in the GTSRB competition*

TEAM	METHOD	TOTAL	SUBSET (All signs)
[156] DeepKnowledge Seville	CNN with 3 Spatial Transformers	99.71%	99.71%
[3] IDSIA ★	Committee of CNNs	99.46%	99.46%
[155] COSFIRE	Color-blob-based COSFIRE filters for object recogn	98.97%	98.97%
[1] INI-RTCV ★	Human Performance	98.84%	98.84%
[4] sermanet ★	Multi-Scale CNNs	98.31%	98.31%
[2] CAOR ★	Random Forests	96.14%	96.14%
[6] INI-RTCV	LDA on HOG 2	95.68%	95.68%
[5] INI-RTCV	LDA on HOG 1	93.18%	93.18%
[7] INI-RTCV	LDA on HOG 3	92.34%	92.34%

Accuracy = 99.832%

The winning algorithm scored 99.7% accuracy whilst our model achieved 99.8%

* http://benchmark.ini.rub.de/?section=gtsrb&subsection=results