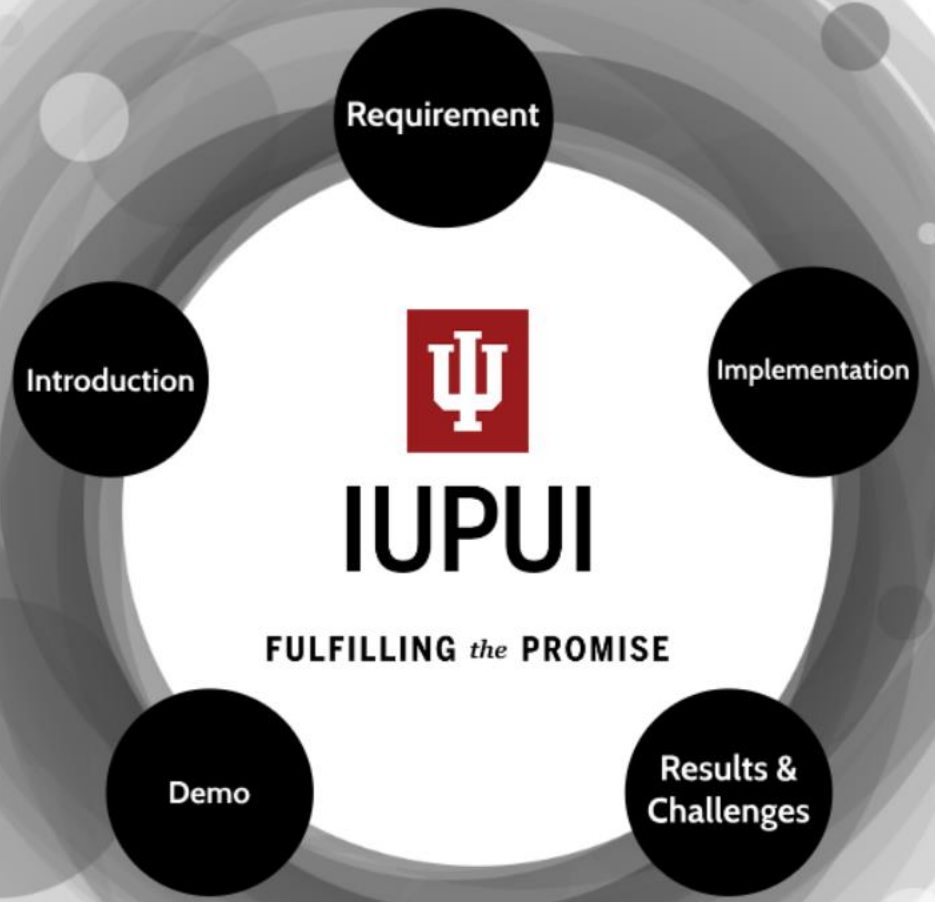
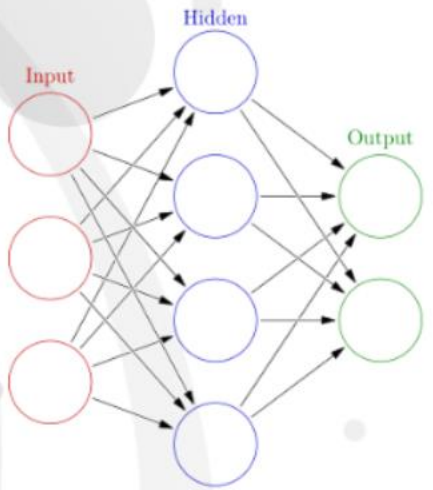
ECE-59500 - EMBEDDED AUTONOMOUS SYSTEMS





By,
Akash Sunil Gaikwad
Surya Kollazhi Manghat
Sarada Gajjala

Traffic Sign Recognition using Convolution Neural Network with RTMaps-Embedded & BlueBox

INTRODUCTION

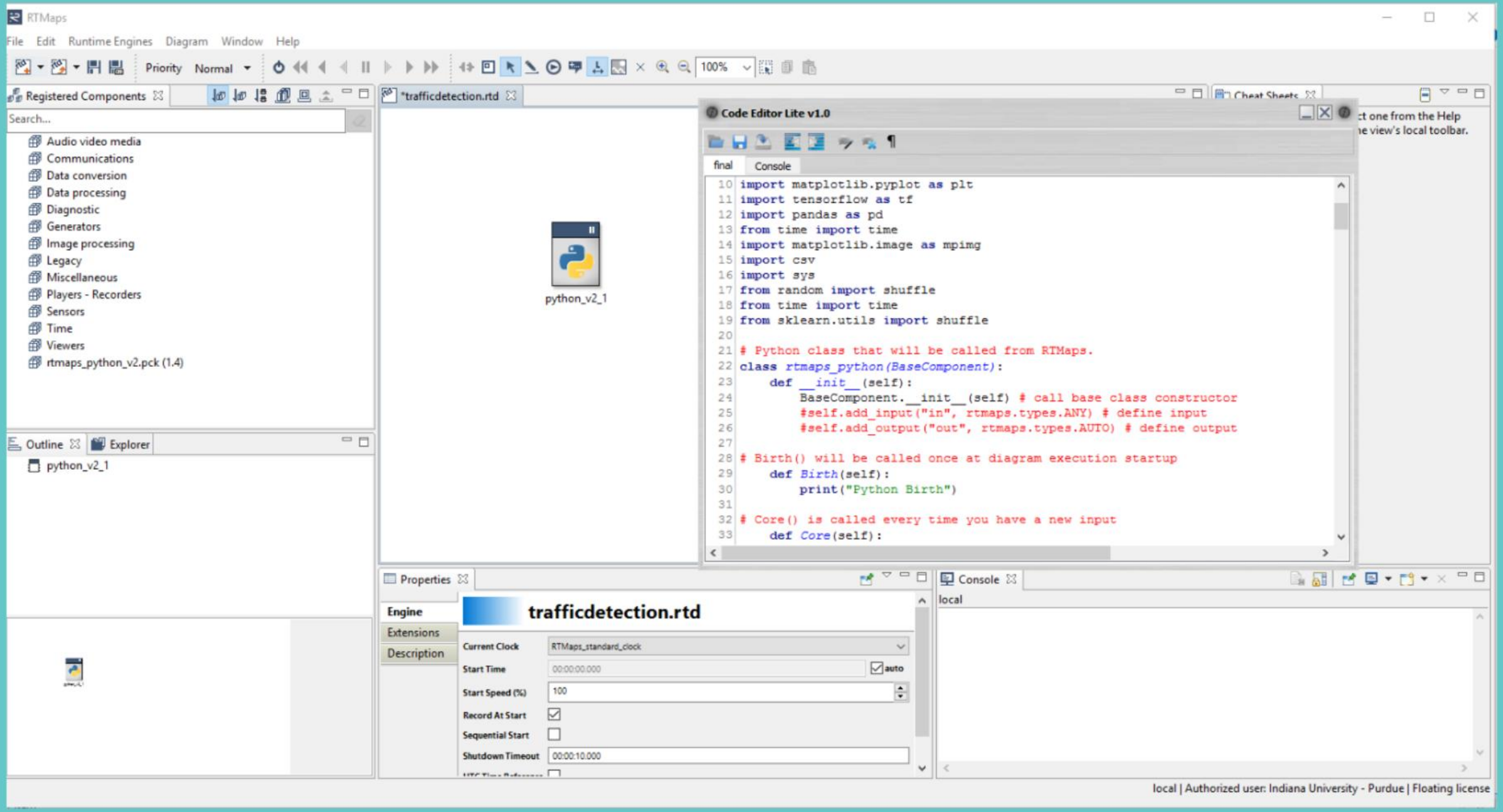
Index

System Overview

- Introduction
- Overview
- Requirements of the project
  - Software
  - Hardware
- Implementation
- Challenges faced
- Result & Conclusion
- Project Demonstration

Overview

- The goal of this project is to establish a neural network in RTMaps-embedded python block to identify the traffic sign board from the input images given.
- We used German traffic sign dataset of ~39000 images for training.
- The trained neural network is added to the python block of RTMaps-embedded.
- The model created as RTMaps component is classifying the input traffic sign to one of the 43 classes.

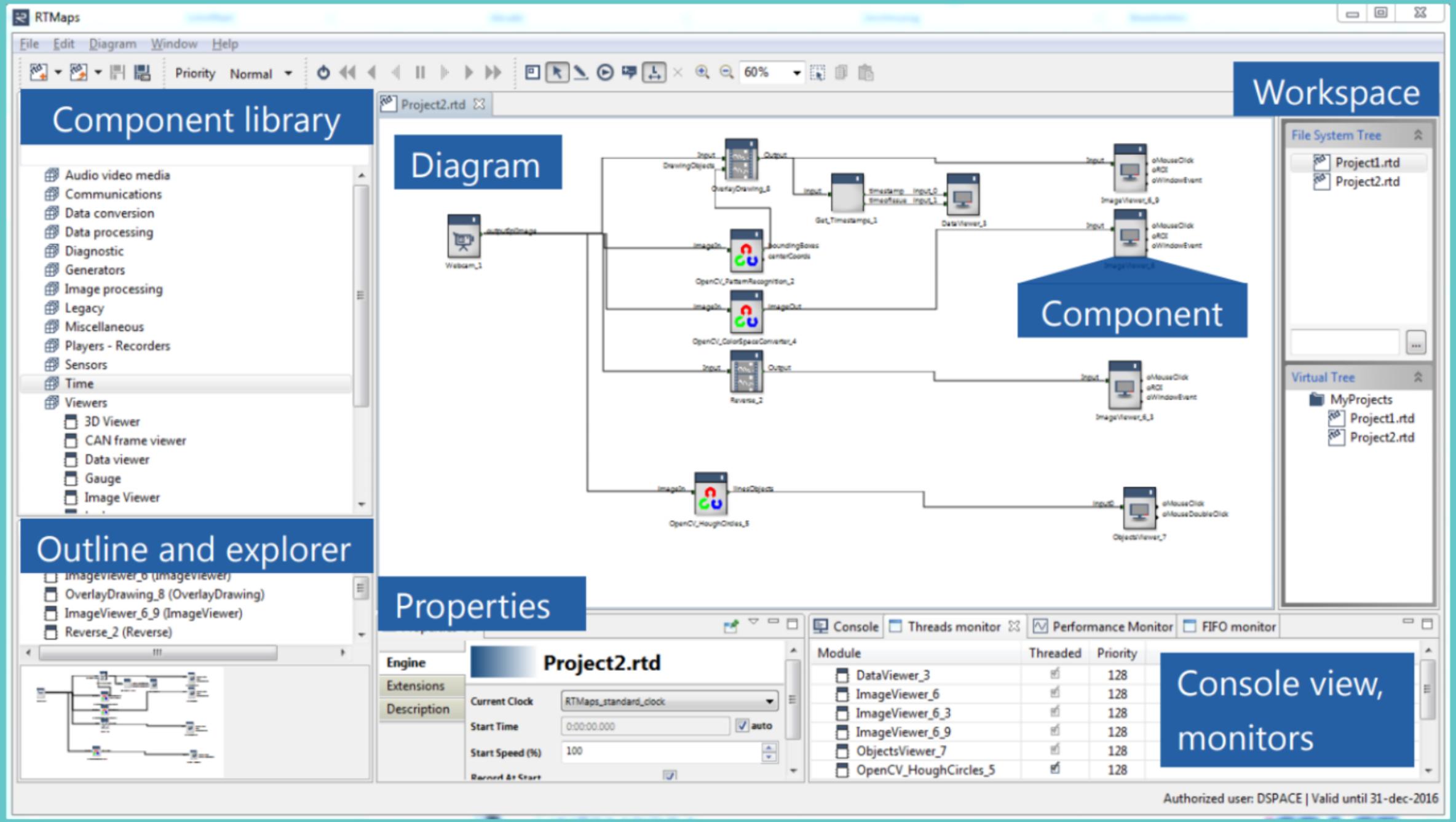

RTMaps
File Edit Runtime Engines Diagram Window Help
Priority Normal
100%
Registered Components
Search...
Audio video media
Communications
Data conversion
Data processing
Diagnostic
Generators
Image processing
Legacy
Miscellaneous
Players - Recorders
Sensors
Time
Viewers
rtmaps_python_v2.pck (1.4)
Outline
Explorer
python_v2_1
*trafficdetection.rtd
python_v2_1
Code Editor Lite v1.0
final
Console
10 import matplotlib.pyplot as plt
11 import tensorflow as tf
12 import pandas as pd
13 from time import time
14 import matplotlib.image as mpimg
15 import csv
16 import sys
17 from random import shuffle
18 from time import time
19 from sklearn.utils import shuffle
20
21 # Python class that will be called from RTMaps.
22 class rtmaps_python(BaseComponent):
23 def __init__(self):
24 BaseComponent.__init__(self) # call base class constructor
25 #self.add_input("in", rtmaps.types.ANY) # define input
26 #self.add_output("out", rtmaps.types.AUTO) # define output
27
28 # Birth() will be called once at diagram execution startup
29 def Birth(self):
30 print("Python Birth")
31
32 # Core() is called every time you have a new input
33 def Core(self):
Properties
Engine
Extensions
Description
trafficdetection.rtd
Current Clock
RTMaps_standard_clock
Start Time
00:00:00.000
auto
Start Speed (%)
100
Record At Start
Sequential Start
Shutdown Timeout
00:00:10.000
Console
local
local | Authorized user: Indiana University - Purdue | Floating license

Software Used

Hardware Used

RTMaps -Embedded

- Asynchronous Realtime Multisensor Processing Framework is more intuitive to operate and easier to program in C++ , Python, Simulink
- Off-the-shelf component libraries for sensors and data processing, e.g., for OpenCV
- A continuous transition from prototyping on PCs to production close implementations on embedded ARM platforms


RTMaps
File Edit Diagram Window Help
Priority Normal
60%
Component library
Audio video media
Communications
Data conversion
Data processing
Diagnostic
Generators
Image processing
Legacy
Miscellaneous
Players - Recorders
Sensors
Time
Viewers
3D Viewer
CAN frame viewer
Data viewer
Gauge
Image Viewer
Project2.rtd
Diagram
Workspace
Component
File System Tree
Project1.rtd
Project2.rtd
Virtual Tree
MyProjects
Project1.rtd
Project2.rtd
Outline and explorer
OverlayDrawing_8 (OverlayDrawing)
ImageViewer_6_9 (ImageViewer)
Reverse_2 (Reverse)
Properties
Engine
Extensions
Description
Project2.rtd
Current Clock
RTMaps_standard_clock
Start Time
0:00:00.000
auto
Start Speed (%)
100
Console
Threads monitor
Performance Monitor
FIFO monitor
Module
Threaded
Priority
DataViewer_3
ImageViewer_6
ImageViewer_6_3
ImageViewer_6_9
ObjectsViewer_7
OpenCV_HoughCircles_5
128
Console view, monitors
Authorized user: DSPACE | Valid until 31-dec-2016

RTMaps

RTMAPS 4™
REAL TIME MULTISENSOR ADVANCED PROTOTYPING SOFTWARE

RTMaps Studio (Developer Version)

- Connection to RTMaps on an embedded platform
- Developing and editing algorithms on a PC
- Use of components compiled for an embedded platform

RTMaps Remote Studio

PC

SSL[1] connection, TCP/IP

[1] Secure Socket Layer

RTMaps Run-Time Version

- Easy deployment on embedded platforms
- Execution of algorithms (RTMaps diagram)
- Advantage of hardware acceleration

RTMaps Remote Studio Connector

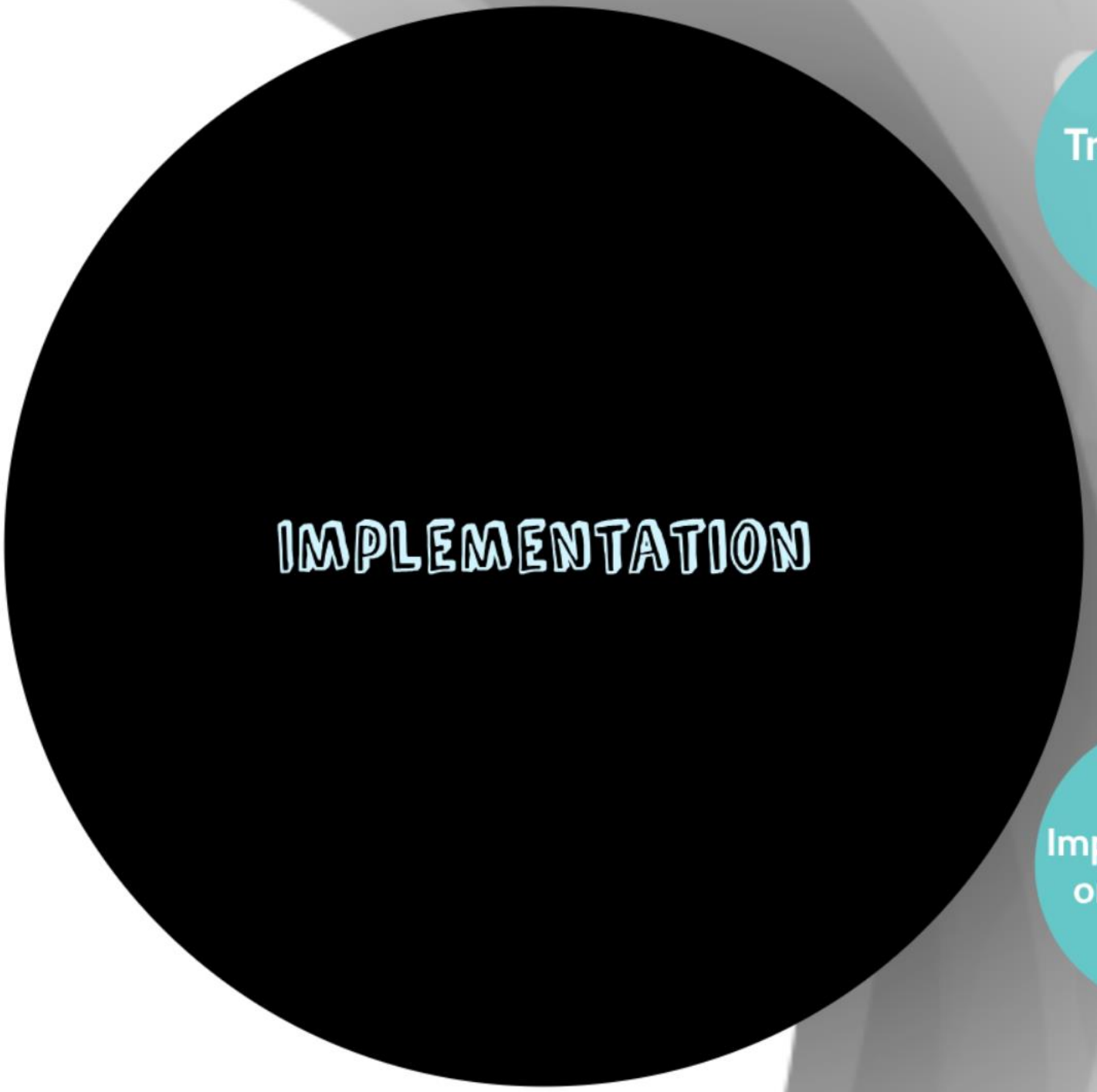
RTMaps diagram

Embedded platform

NXP BLUEBOX 2.0

- Bluebox consists of 3 major components integrated together to provide a powerful, yet flexible, and safe development platform for automotive and industrial applications
- The LS2 provides high single and multithreaded performance with eight ARM A72 cores
- The S32V features a massively parallel Vision accelerator known as the APEX2, which is well suited for image processing algorithms and light machine learning inference work
- The S32R is offered as an ASIL-D supervisor that complements the ASIL-B ready S32V


RTMaps
Real-Time Multisensor applications
NXP
POWER
S32V RST
S32R RST
LS2 RST
10GE P1
10GE P3
USB
LS2 SD
S32V SD
UART
HDMI



Training The Model

Evaluation of The Model

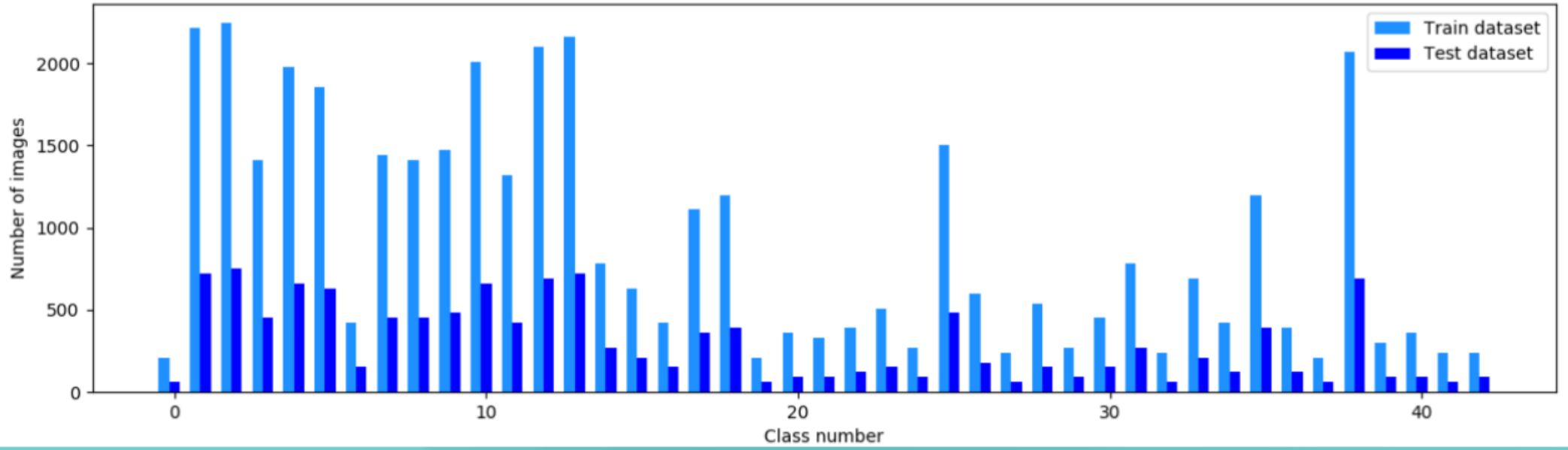
Implementation on Hardware

Training the Convolution Neural Network

- Data Preparation
- Data Augmentation
- Data Processing
- Convolution Neural Network Architecture
- Save Tensorflow Model
- Evaluation of The Model

Data Preparation

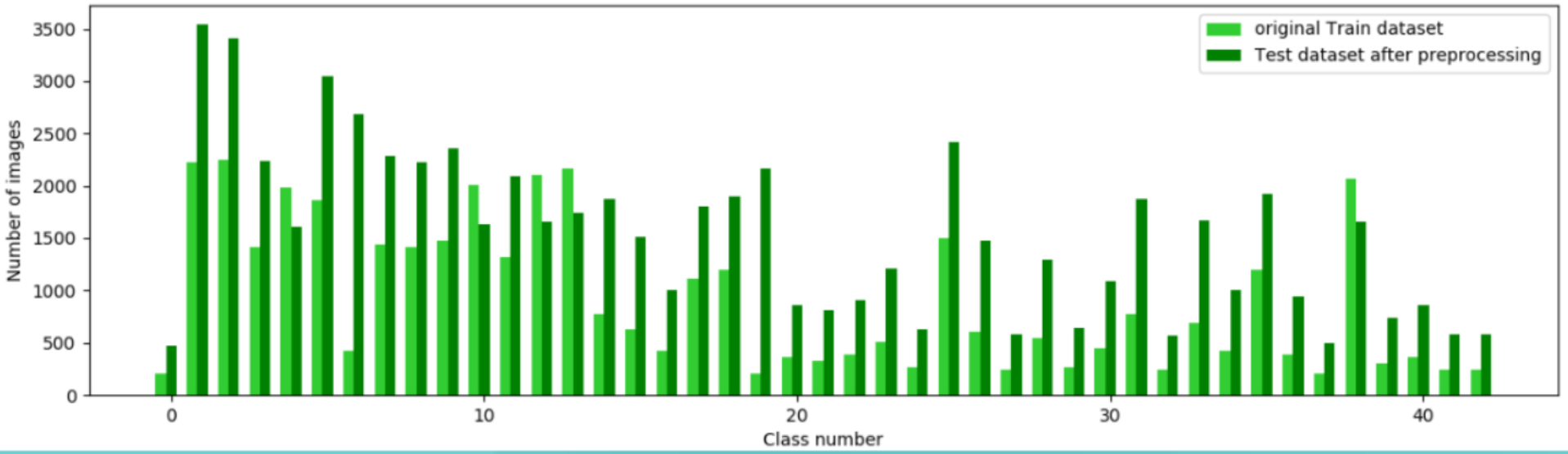
- Divide the Dataset into training and testing dataset. (German Traffic Signs)
- Create a Pickle Object using Training images and labels


Train dataset
Test dataset
Number of images
2000
1500
1000
500
0
0
10
20
30
40
Class number

Data Augmentation

- Image augmentation artificially creates training images through different ways of processing or combination of multiple processing, such as random rotation, shifts, shear and flips, etc.

- Increase the number of samples of the training dataset

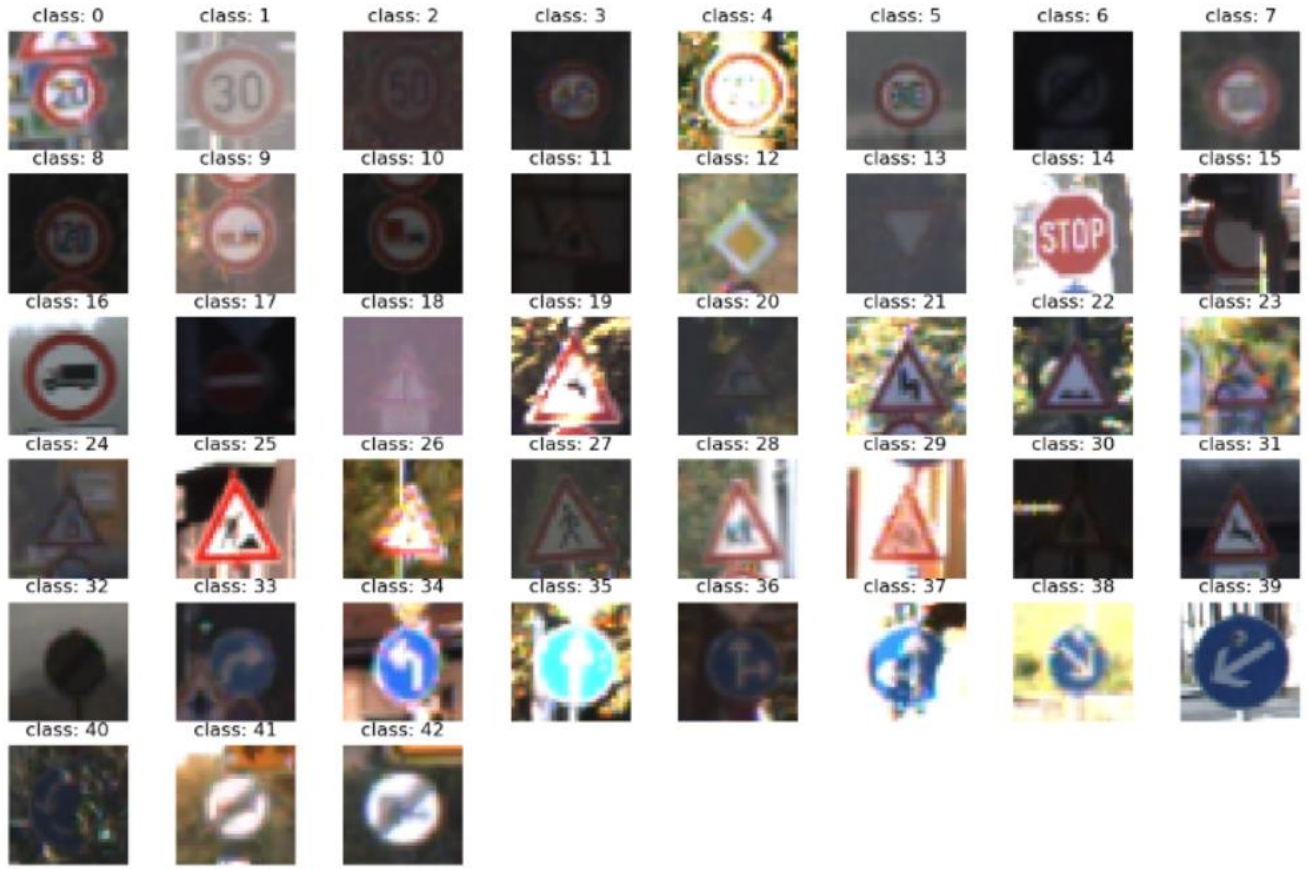


Data Processing

- Transform the Images to Grayscale format.
- Transform the images to proper format (ex. 64x64)
- Normalise the Images

```
(keras_tf) akash@akash-GL553VE:~/Desktop/595_Final_Proj

Image Shape: (32, 32, 3)

Training Set:   39209 samples
Test Set:       12630 samples

Number of training examples = 39209
Number of testing examples = 12630

Image data shape = (32, 32, 3)
Number of classes = 43
Images for Train dataset (before enlargement process):
Minimum for class = 210
Mean for class = 912
Maximum for class = 2250
Images for Test dataset:
Minimum for class = 60
Mean for class = 294
```

Images after Data Processing



Convolution Neural Network Architecture

#_____Layer 1_____
Convolutional. Input = 32x32x1. Filter = 5x5x1. Output = 28x28x6.
ReLu activation function

#_____Layer 2_____
Convolutional. Input = 28x28x6. Filter = 3x3x6. Output = 14x14x6.
Pooling. Input = 28x28x6. Output = 14x14x6.
#_____Layer 3_____#
Convolutional. Input = 14x14x6. Filter = 5x5x12. Output = 10x10x16.
ReLu activation function
Pooling. Input = 10x10x16. Output = 5x5x16.

#_____Layer 4_____#
Flatten. Input = 5x5x16. Output = 400.
Fully Connected. Input = 400. Output = 120.
ReLu activation function
#_____Layer 5_____#
Fully Connected. Input = 120. Output = 84.
ReLu activation function
#_____Layer 6_____#
Fully Connected. Input = 84. Output = 43.

Save Tensorflow Model

Save the TensorFlow model as following files

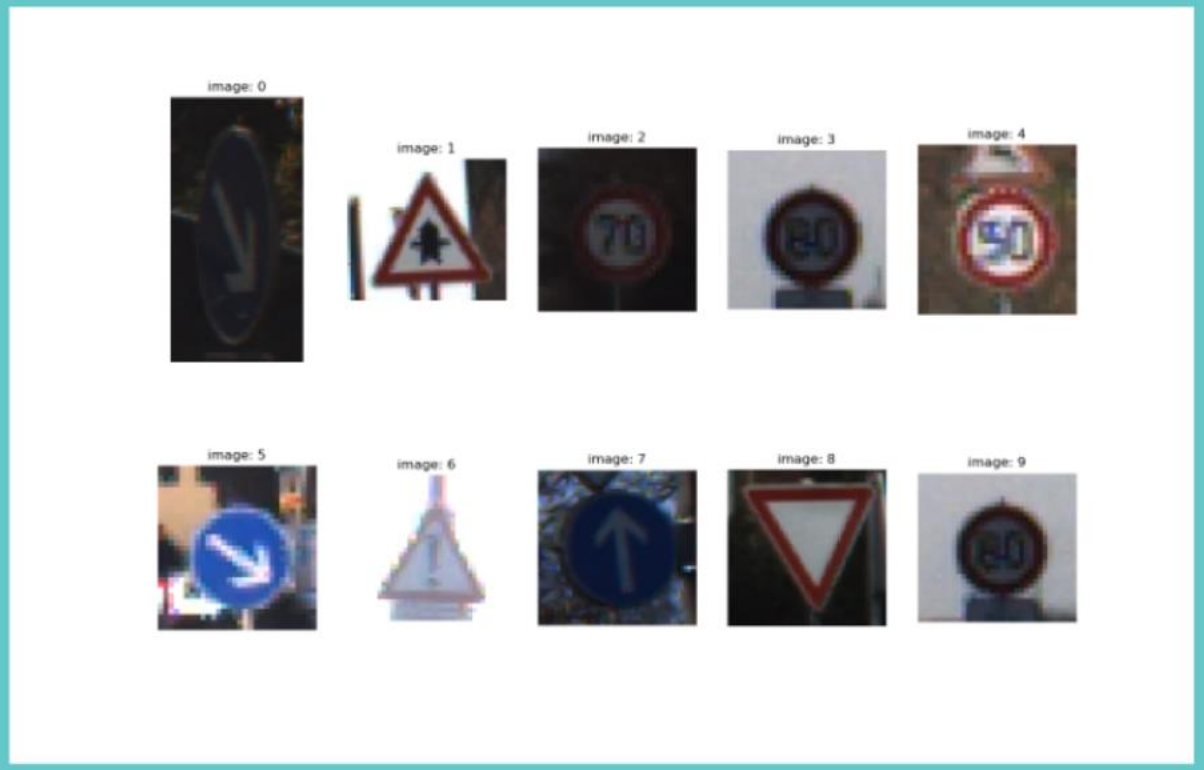
- my_test_model-1000.index
- my_test_model-1000.meta
- my_test_model-1000.data-00000-of-00001
- checkpoint

```
Model saved
Final Validation Accuracy = 0.947
Total time for model training:  6.0 min, 14 s
New images after reshape:  (10, 32, 32, 1)
2018-04-29 14:35:37.294693: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1120] Creating TensorFlow device (/device:GPU:0) ->
capability: 6.1)
```

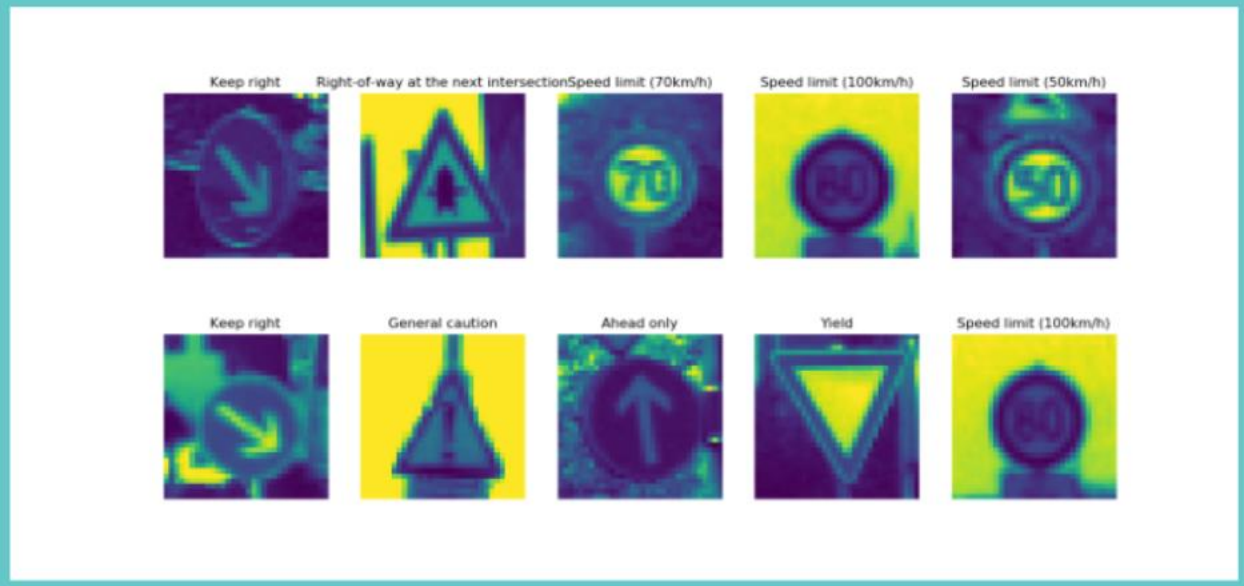
Evaluation of The Model

- Evaluate the CNN by passing New Images to Model.
- Pass New image to feed Forward Network and Get the probability of class

portable pixmap format (PPM)



New Images

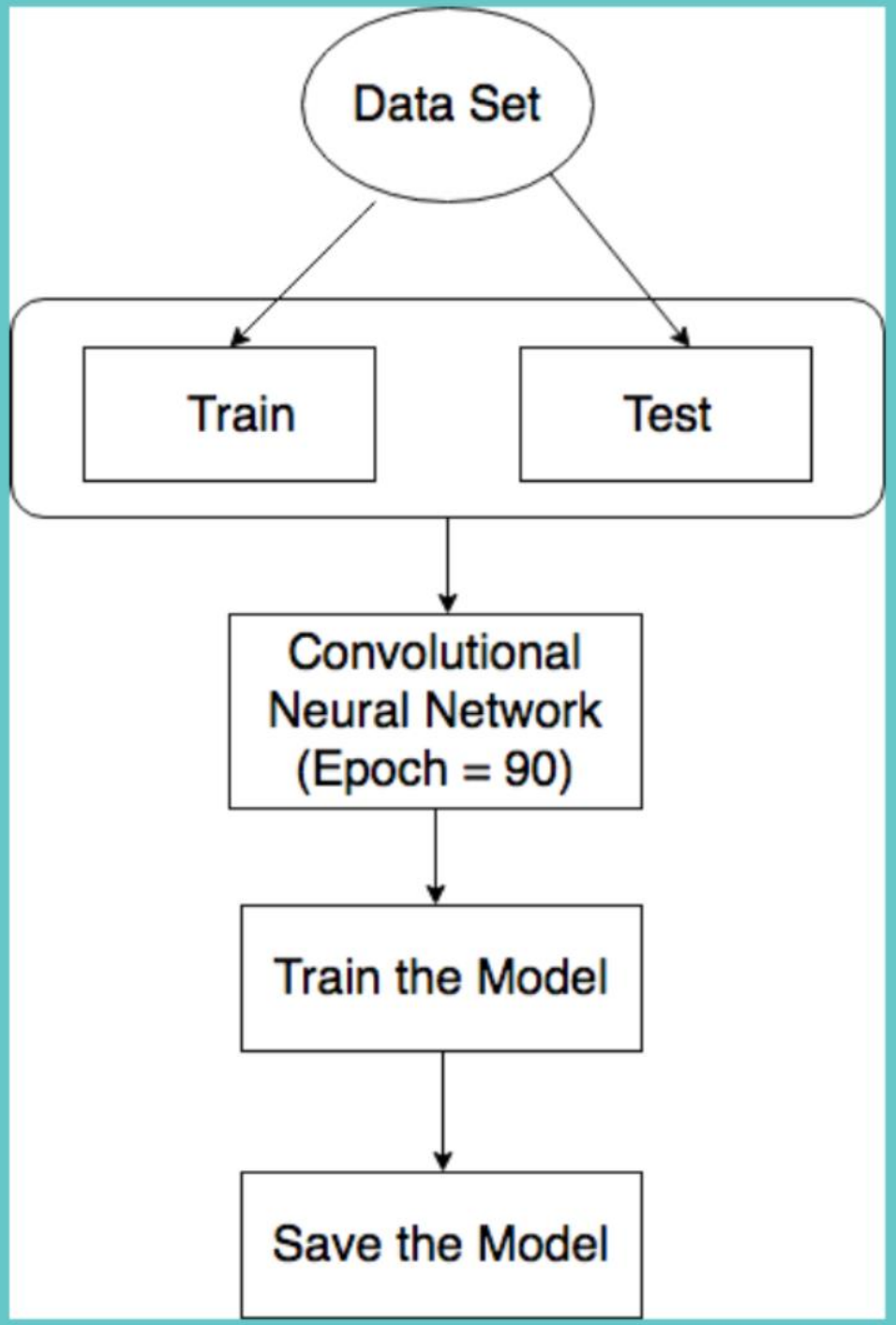


Predicted Output

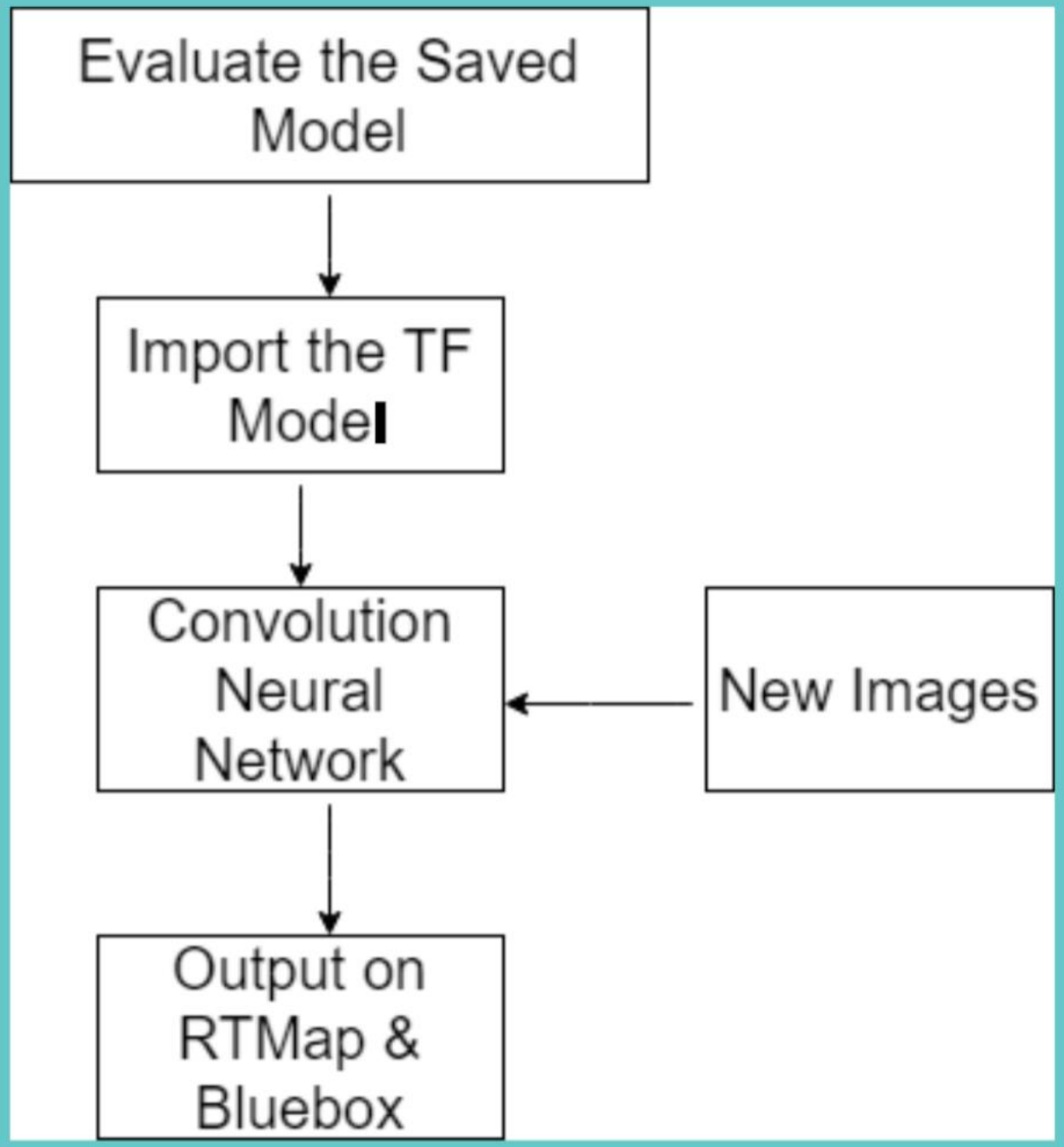
portable pixmap format (PPM)

Implementation on Hardware

- Implementation of CNN on PC (Ubuntu 16.04 LTS)
- Implementation of CNN on RTMaps studio (Simulation)
- Implementation of CNN on RTMaps runtime engine and BlueBox. (Hardware)


Data Set
Train
Test
Convolutional Neural Network (Epoch = 90)
Train the Model
Save the Model

Flowchart


Evaluate the Saved Model
Import the TF Model
Convolution Neural Network
New Images
Output on RTMap & Bluebox

Flowchart

RESULTS & CHALLENGES

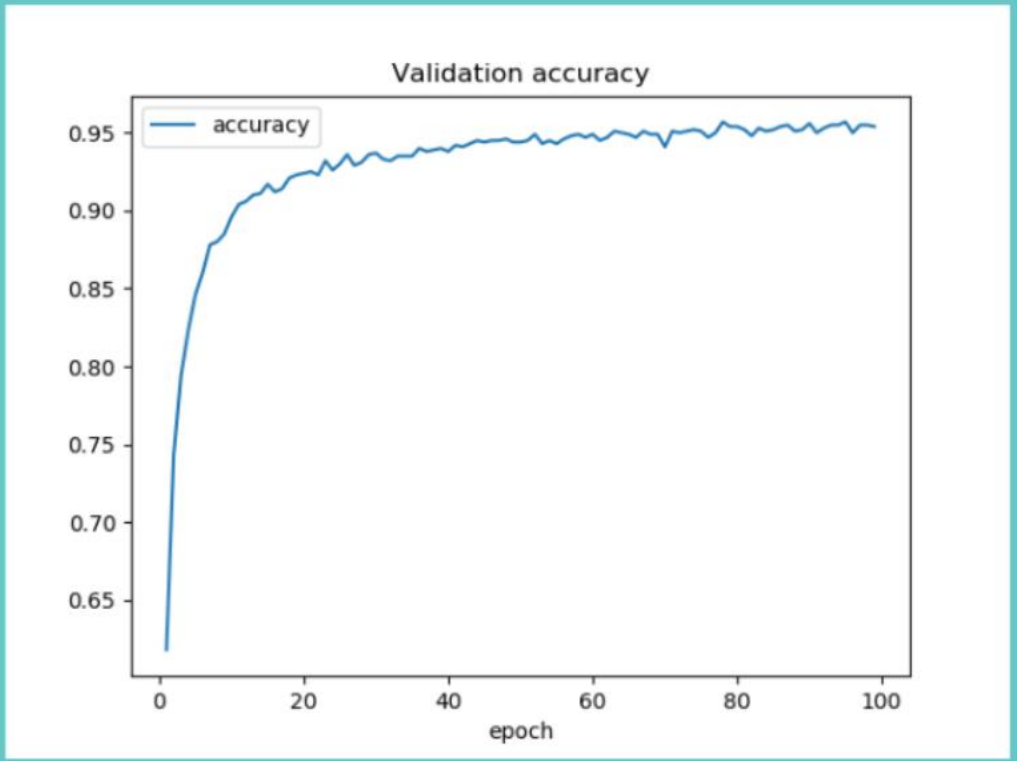
Results

Challenges

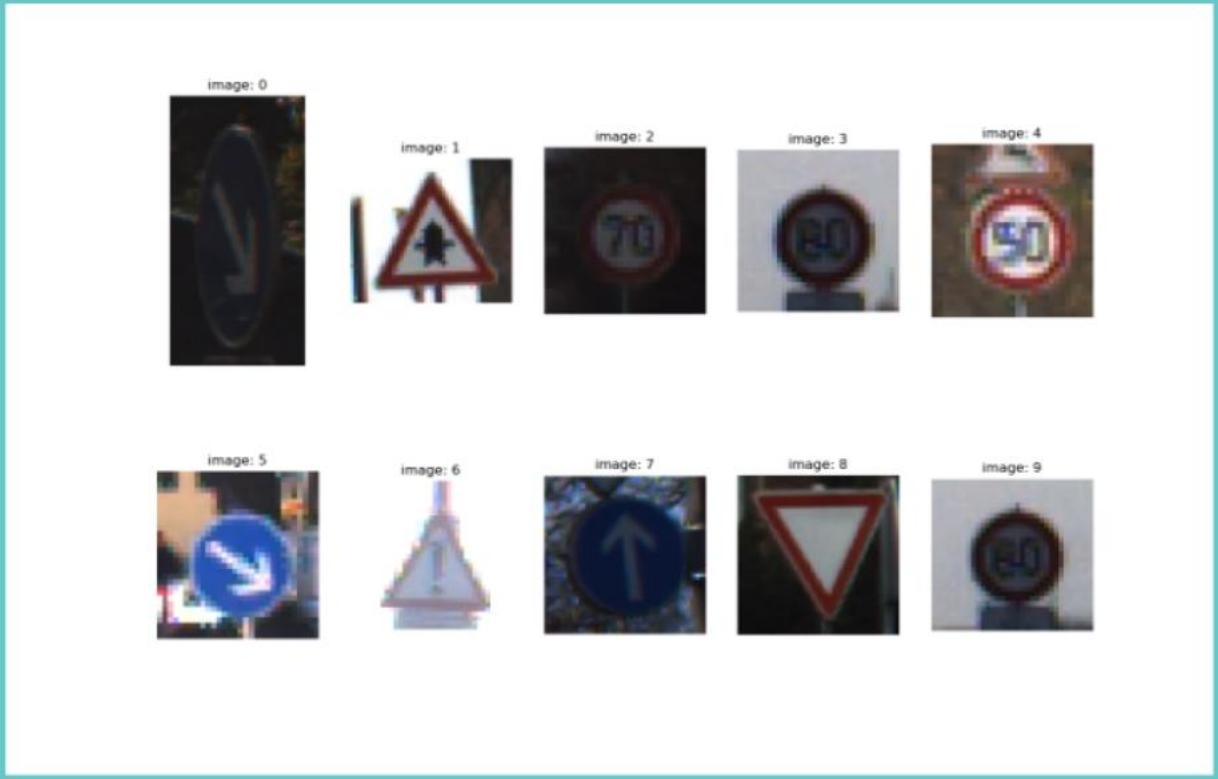
Results

Accuracy of the Model

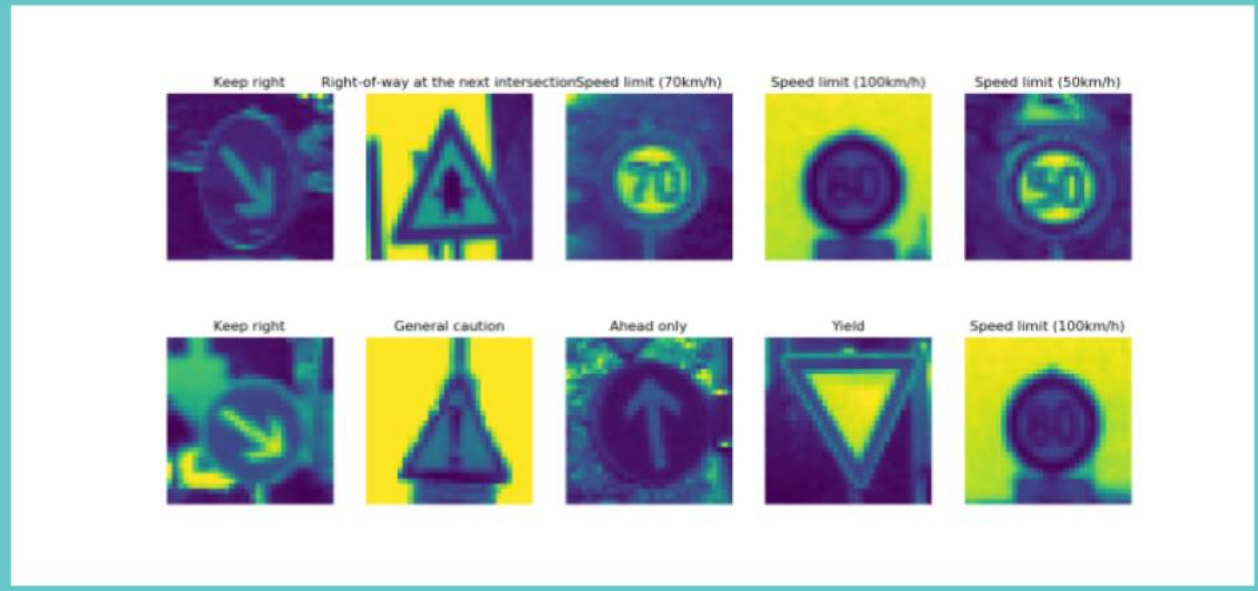
```
Model saved
Final Validation Accuracy = 0.947
```



Prediction Results (With GUI)



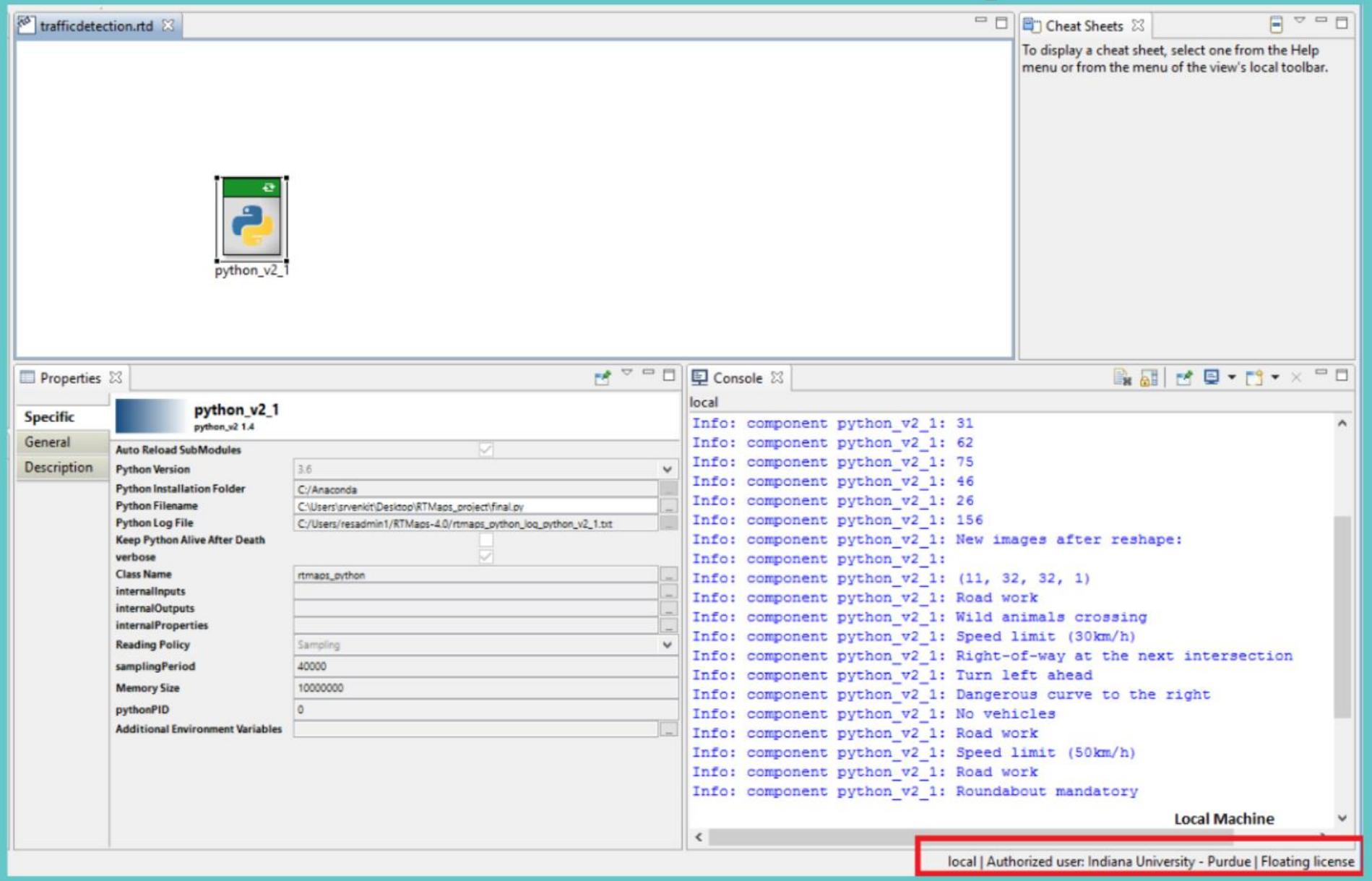
New Images



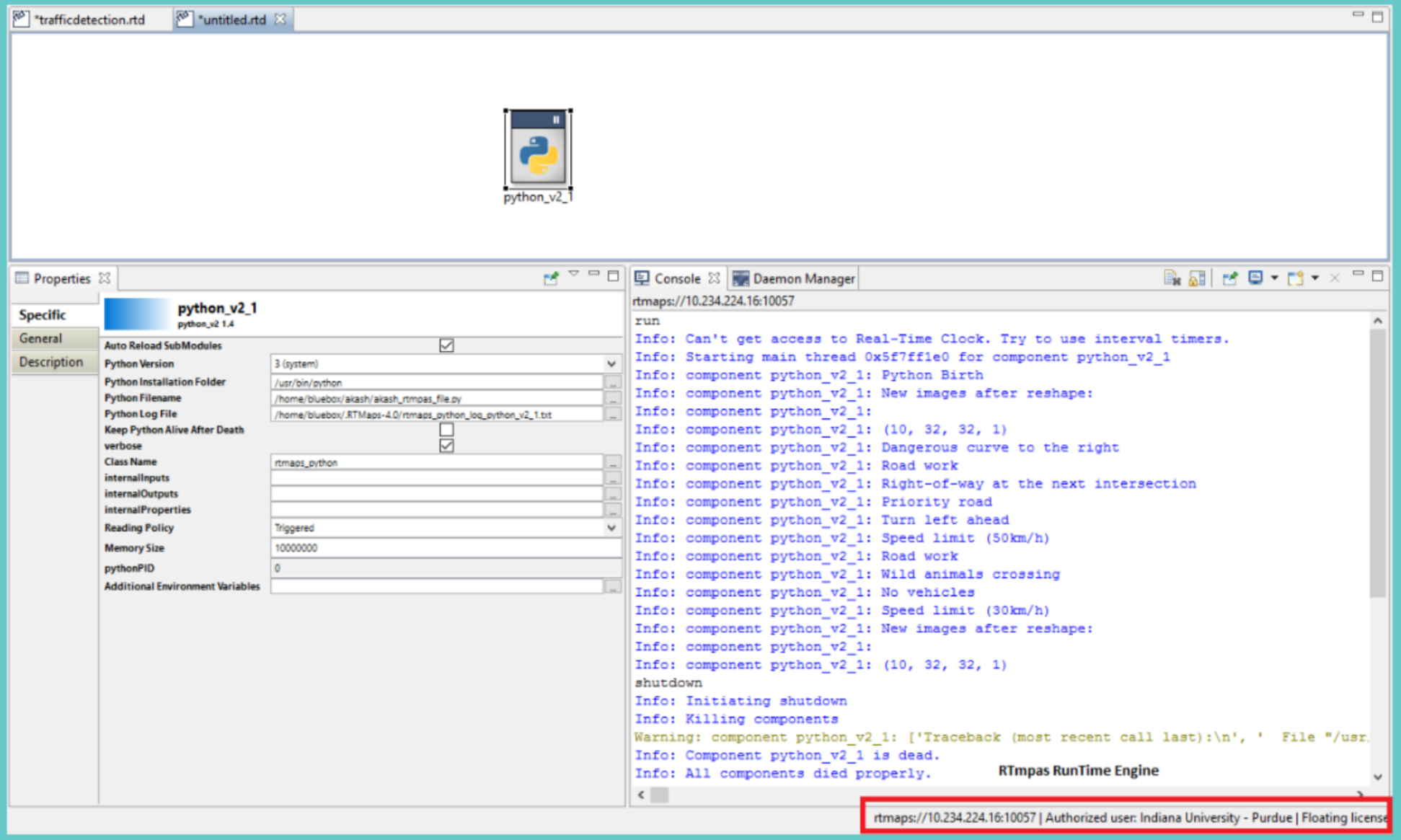
Prediction with Labels

PC Output

Prediction Results with RTmap Simulation



Prediction Results with RTmap & BlueBox



Challenges faced

- Compiling the OpenCV 3.0 for arm64 Architecture
- Installing TensorFlow for arm64 Architecture on Blue-Box
- Licensing Issues of RTMaps
- No Graphical user interface for RTMaps Runtime Engine (No X11 Connection)

FUTURE SCOPE & DEMO

Future Scope

Demo

Thank you!

Future Scope

- Maker use of S32V234 APEX vision accelerator to decrease the computational power.
- Interfacing camera to capture real time images.
- Along with other sensors forming a multi sensor data fusion for accurate decision.
- Atlast,building an intelligent autonomous vehicle which can read and react to traffic boards with the ISO26262 safety standard.

Thank you!

For your patience

Question Time !!