

Welcome to My GitHub Profile

I am Irfan. I am a passionate AI Engineer, always seeking to learn something new. I am passionate about **Artificial Intelligence, Computer Vision, Machine Learning and Robotics**. My works on these fields can be found here. I have used Python and C++ in these works. For detailed algorithm, code and results of each project, please see the respective sections. I have designed the flowcharts to precisely demonstrate, how each of the algorithms work. The page have many graphical contents, it may take 5-30 seconds to load properly based on internet connection. E-mail : irfanhasib.me@gmail.com (<mailto:irfanhasib.me@gmail.com>)

Table of Content

Machine Learning Algorithms from scratch	Deep Learning for Computer Vision	Reinforcement Learning Algo. from scratch
<ul style="list-style-type: none"> ◦ Neural Network ◦ Decision Tree Algorithms ◦ SVM, Logistic Regression ◦ Naive Bayes, KNN 	<ul style="list-style-type: none"> ◦ Yolo-V2.0 : Object Detection ◦ Unet : Semantic Segmentation ◦ FlowNet : Optical Flow Estimation ◦ Disparity Estimation 	<ul style="list-style-type: none"> ◦ DQN ◦ DDPG ◦ PPO ◦ A2C
Control Algorithms from scratch	Feature Engineering and Model Tuning	ROS & Robotics
<ul style="list-style-type: none"> ◦ ILQR ◦ MPC 	<ul style="list-style-type: none"> ◦ Kaggle House Price Prediction. ◦ Shakura Bloom Prediction. 	<ul style="list-style-type: none"> ◦ ROS : Simple two linked robot ◦ ROS : Husky and URbot(UR5) robot driver
Mars Rover Challenge (USA, Utah)	Professional AI Projects	Professional Embedded System Projects
<ul style="list-style-type: none"> ◦ University Rover Challenge - 2016 ◦ ◦ 	<ul style="list-style-type: none"> ◦ Process control automation (AI) ◦ Optimal shipping plan selection (AI) ◦ Early prediction Production of KPI (DL) ◦ Production Dynamics visualization (ML) 	<ul style="list-style-type: none"> ◦ Vault Security System ◦ Programmable Syringe Infusion Pump ◦ GPRS based Monitoring System ◦ Online digital weights machine
Academic Project and Thesis (Undergrad)	Robotics Projects Personal (Undergrad)	
<ul style="list-style-type: none"> ◦ Remote rescue robot ◦ GPS, IMU data Fusion for precision velocity 	<ul style="list-style-type: none"> ◦ Desktop CNC Machine(2014) ◦ GUI software for Robotic ARM(2014) 	

- [Visually Instructed Robotic ARM\(2013\)](#)

Neural Network Implementation from scratch

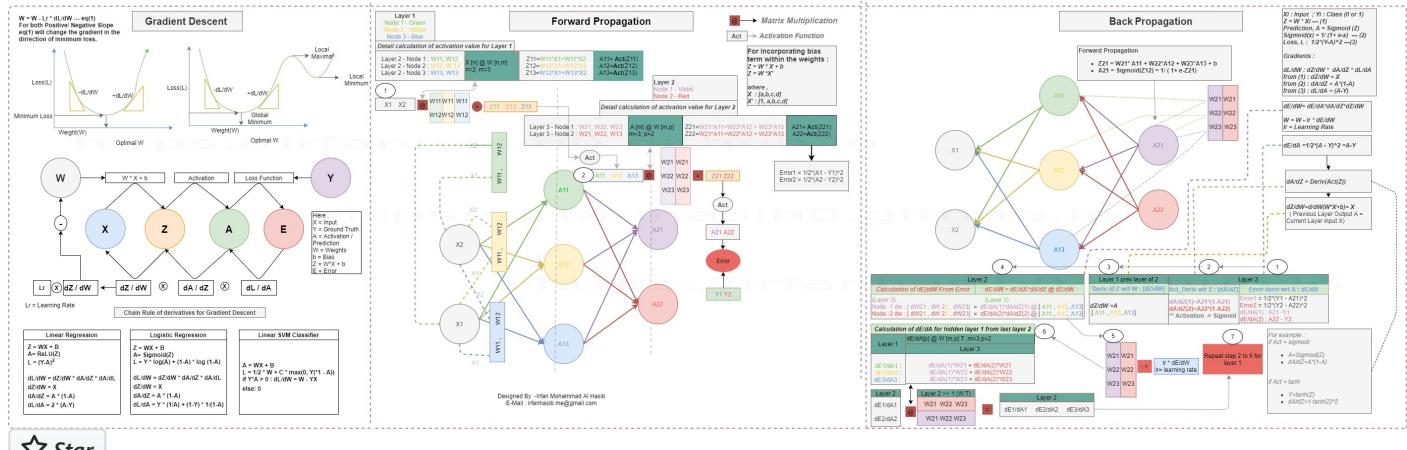
[Go Back to Table of Content](#)

- **Section 1: Overview**
- **Section 2. Flowchart**
- **Section 3. Code**
- **Section 4. Result**

1. Overview :

Code for Multi Layer Neural Network is implemented here from scratch, using raw Python programming language. Object oriented programming is used to better organize the code and making it user friendly. A **Layer** and a **Network Object** is the backbone of the implementation. A stack of Layer objects can be used to build a Network. Network object has the following functions(1) Layer :: "forward_propagation", (2) Layer :: "backward_propagation", (3) Layer :: "update_weights". I have tested it with Banknote dataset. It was fitting good. Result can be found in Results section

2. Flow Chart (ANN) : (Open Image in new tab for full resolution)



3. Implementation Code :

[GitHub : NN Implementation from scratch - Notebook \(\[https://github.com/irfanhasib0/Machine-Learning/blob/master/Machine_Learning_Algo_From_Scratch/ANN_From_Scratch_modular_class-.py\]\(https://github.com/irfanhasib0/Machine-Learning/blob/master/Machine_Learning_Algo_From_Scratch/ANN_From_Scratch_modular_class-.py\)\)](https://github.com/irfanhasib0/Machine-Learning/blob/master/Machine_Learning_Algo_From_Scratch/ANN_From_Scratch_modular_class-.py)

[V1.0.ipynb\)](#)

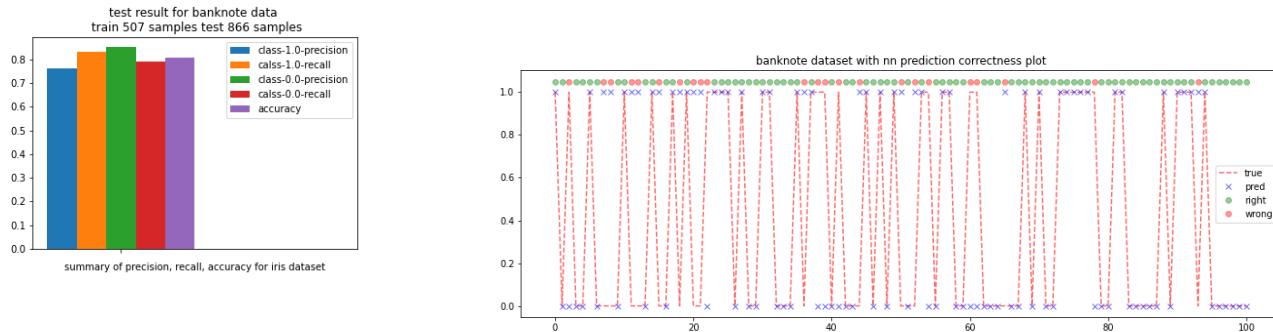
ANN Implementation From Scratch

```
In [1]: import numpy as np
from matplotlib import pyplot as plt
import pandas as pd
from sklearn.preprocessing import MinMaxScaler, RobustScaler
np.random.seed(0)
```

```
In [2]: dataset='banknote'
df=pd.read_csv('Dataset/banknote.csv').sample(frac=1.0)
N=len(df)

df=df.fillna(0)
```

4. Result of ANN implementation for Bank Note data- a.Precision, Recall and Accuracy and b.True Label vs Prediction



Decision Tree Implementation from scratch

[Go Back to Table of Content](#)

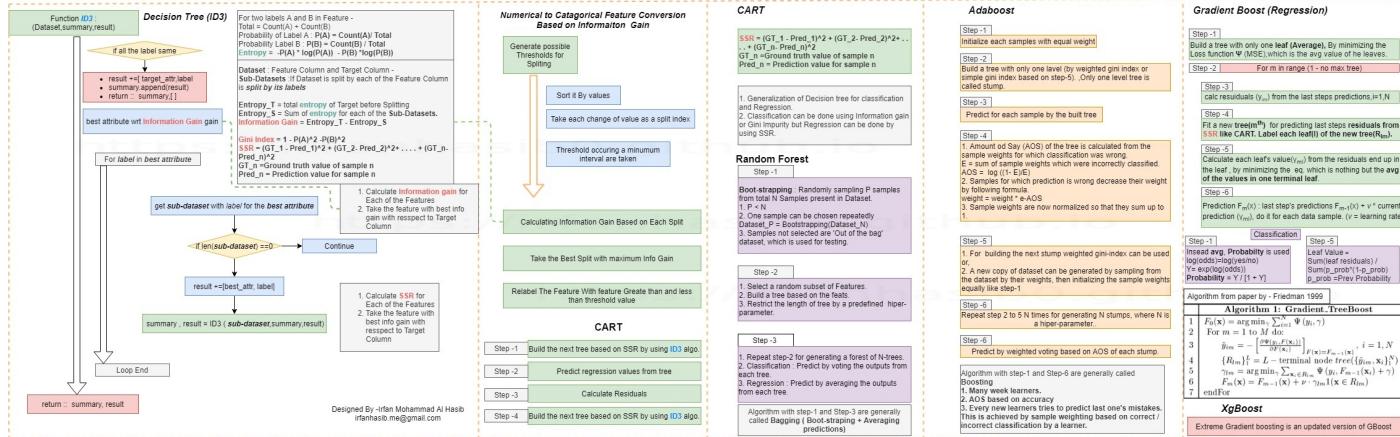
- **Section 1: Summary**
- **Section 2: Flowchart**
- **section 3: Code**
- **Section 4:Result (Results can be found after code demonstration for each algorithm)**

1. Overview

A fundamental decision tree based classifier is implemented here using ID3 algorithm with Python from scratch. Titanic and irish dataset was used for testing the algorithm. Function for (1) Continuous data splitting based on information gain, (2) Information Gain Calculation, (3) ID3 Algorithm (4) Prediction from the built tree, are main part of the implementation. For preventing overfitting reduced error pruning was applied. Accuracy,Precision,Recall is reported in Result section.

2. ID3 Flow Chart of Implementation (Open Image in new tab for full resolution)

Star



3. ID3 Implementation Code

[GitHub : ID3 Implementation from scratch - Notebook \(\[https://github.com/irfanhasib0/Machine-Learning/blob/master/Machine_Learning_Algo_From_Scratch/ID3_with_continuous_feature_support_V1.0.ipynb\]\(https://github.com/irfanhasib0/Machine-Learning/blob/master/Machine_Learning_Algo_From_Scratch/ID3_with_continuous_feature_support_V1.0.ipynb\)\)](https://github.com/irfanhasib0/Machine-Learning/blob/master/Machine_Learning_Algo_From_Scratch/ID3_with_continuous_feature_support_V1.0.ipynb)

ID3 Implemented from scratch with python

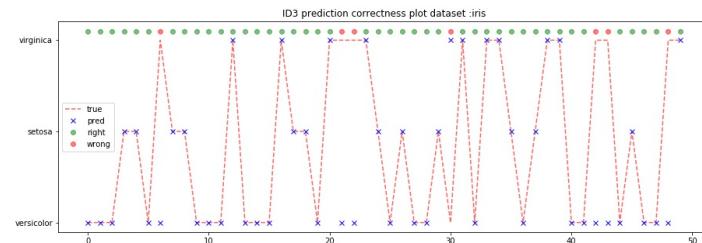
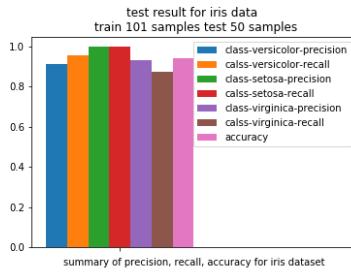
** some basic pandas and numpy utility functions are used

1. Continuout to Discrete Conversion

Utility functions

Here I tried to avoid pandas utility functions and implemented many of them from scratch.

4. Result of ID3 implementation for iris data - a.Precision, Recall and Accuracy and b.True Label vs Prediction -



SVM, Logistic Regression Implementation from scratch

[Go Back to Table of Content](#)

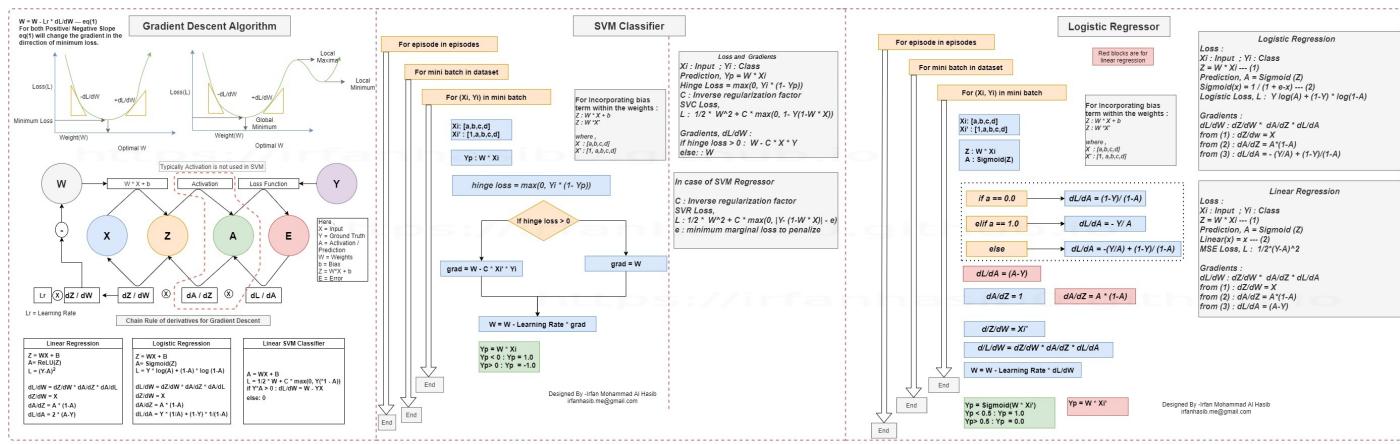
- **Section 1: Summary**
- **Section 2: Flowchart (SVM & Logistic Regression)**
- **section 3: Code (3.1 SVM, 3.2 Logistic Regression)**

◦ Section 4: Result(4.1 SVM ,4.2 Logistic Regression)

1. Overview

SVM and Logistic Regression code from scratch is presented below. SvmC object with the functions (a) SvnC.predict (b) SvnC.calc_gradients (c) SvnC.update_weights is implemented. LogisticRegression object also have the same functions. Banknote dataset is used for testing both of the algorithms. SVM and LogisticRegression Class is implemented accordingly for the respective algorithms. For each of the 2 class predict, calc loss, calc grads, update weights functions are implemented. Both of the algorithms showed good accuracy for banknote dataset after minimal parameter tuning. Results can be found in respective results section.

2.0 SVM(Left), Logistic Regression(Right) Flow Chart of Implementation (Open Image in new tab for full resolution)



3.1 Implementation Code SVM

(a) GitHub : SVM Implementation from scratch - Notebook (https://github.com/irfanhasib0/Machine-Learning/blob/master/Machine_Learning_Algo_From_Scratch/SVM.ipynb).

SVM Implementation From Scratch

```
In [1]: import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler, RobustScaler
dataset='banknote'
df=pd.read_csv('Dataset/banknote.csv').sample(frac=1.0).reset_index(drop=True)
N=len(df)

df=df.fillna(0)
feats=df.columns.values[:-1]
target=df.columns.values[-1]
```

3.1 Implementation Code Logistic Regression

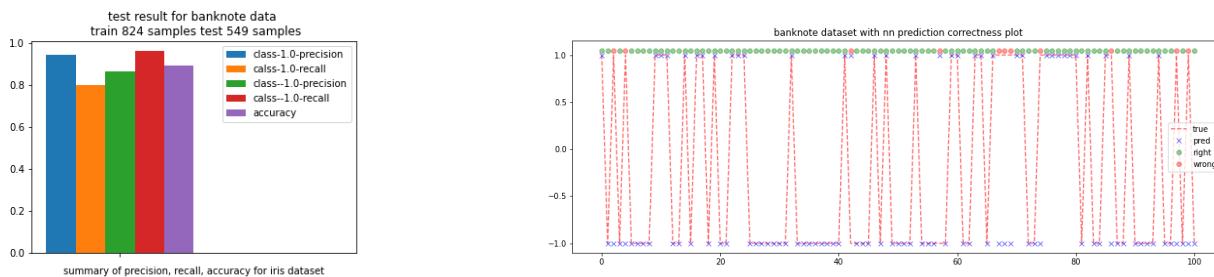
(b) GitHub : Logistic Regression Implementation from scratch - Notebook (https://github.com/irfanhasib0/Machine-Learning/blob/master/Machine_Learning_Algo_From_Scratch/Logistic_Regression.ipynb)

Logistic Regression Implementation From Scratch

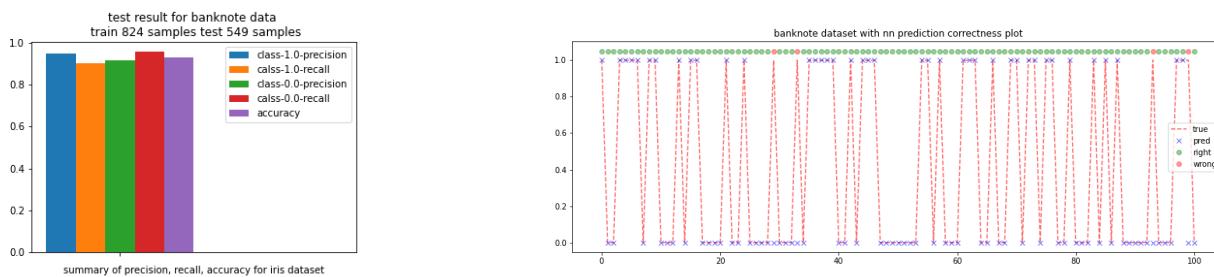
```
In [1]: import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler, RobustScaler
dataset='banknote'
df=pd.read_csv('Dataset/banknote.csv').sample(frac=1.0).reset_index(drop=True)
N=len(df)

df=df.fillna(0)
feats=df.columns.values[:-1]
target=df.columns.values[-1]
```

4.1 Result of SVM implementation - a.Precision, Recall and Accuracy and b.True Label vs Prediction



4.2 Result of Logistic Regression implementation - a.Precision, Recall and Accuracy and b.True Label vs Prediction



Naive Bayes and KNN Implementation for text classification

[Go Back to Table of Content](#)

- **Section 1: Overview**

- **Section 2: Flowchart (Naive Bias & KNN)**
- **Section 2: Code (3.1 Naive Bias , 3.2 KNN)**
- **SEction 3: Result (4.1 Naive Bias, 4.2 KNN)**

1. Overview

o Dataset :

Archived data from stack exchange is used for classification.

o Naive Bias Steps :

- Text Preprocessing was done with raw python.
- Calculating Attribute(word) Probabilities given each class
- Calculating Attribute(word) Probabilities given each Samples
- Applying Bayes Theorem for getting class probabilities
- Max class probability is the predicted label

o KNN Steps :

- Text Preprocessing was done with raw python.
- Calculating Attribute Probability vector for each sample
- Calculating Cosine/Eucleidian/Humming Distances from test sample to each of train sample
- Taking labels of K min distance samples from train data
- Taking mode of K labels as prediction (tie breaks by random choise)

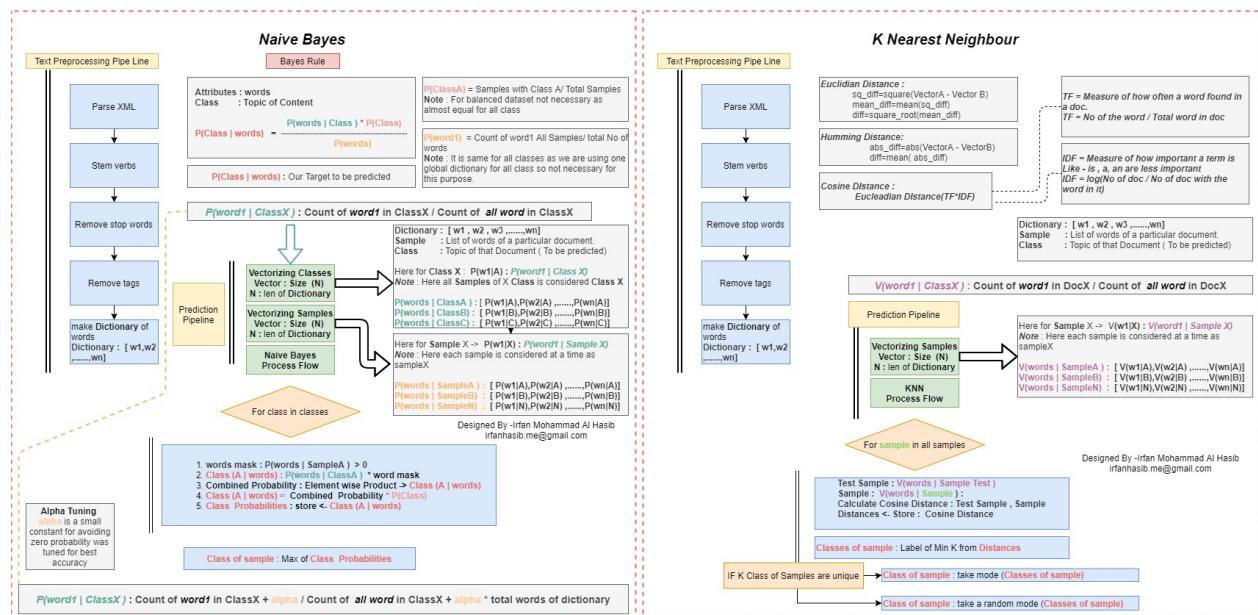
o Tuning :

The model was Tuned for best alpha values.

o Prediction :

Accuracy,Precision,Recall Reporting.

2. (a)Naive Bayes (b)K Nearest Neighbour algorithm Flow Chart



3.1 Implementation Code Naive Bias

(a) [GitHub : Naive Bayes Implementation from scratch - Notebook](https://github.com/irfanhasib0/Machine-Learning/blob/master/Machine_Learning_Algo_From_Scratch/Naive_Bayes_Stack_Exchange.ipynb)

(https://github.com/irfanhasib0/Machine-Learning/blob/master/Machine_Learning_Algo_From_Scratch/Naive_Bayes_Stack_Exchange.ipynb).

Naive Bias Implementation From Scratch

```
In [1]: from xml.dom import minidom
import re
import pandas as pd
import sys
import pickle
import numpy as np
#from nltk.stem import PorterStemmer
#from nltk.tokenize import sent_tokenize, word_tokenize
#ps = PorterStemmer()
```

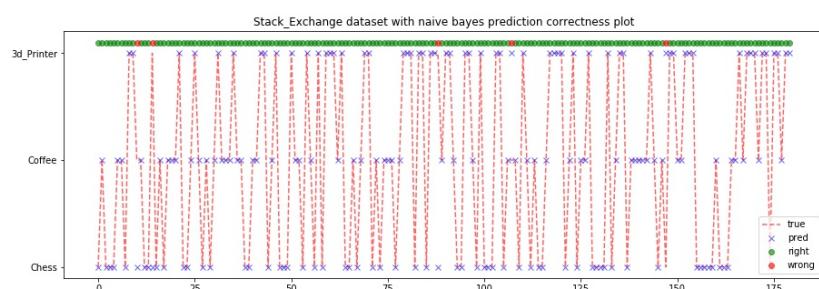
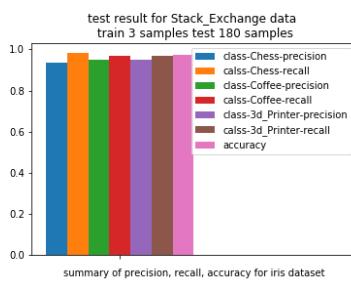
3.2 Implementation Code K Nearest Neighbour

(b) GitHub : KNN Implementation from scratch - Notebook (https://github.com/irfanhasib0/Machine-Learning/blob/master/Machine_Learning_Algo_From_Scratch/KNN_Stack_Exchange.ipynb).

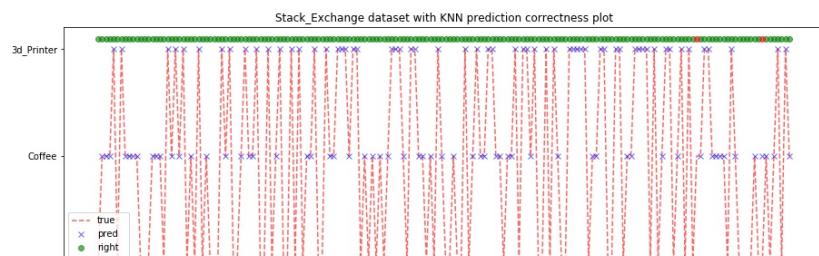
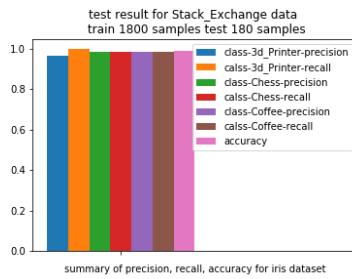
KNN Implementation from Scratch

```
In [1]: from xml.dom import minidom
import re
import pandas as pd
import sys
import pickle
import numpy as np
import time
from nltk.stem import PorterStemmer
from nltk.tokenize import sent_tokenize, word_tokenize
ps = PorterStemmer()
sys.__stdout__=sys.stdout
```

4.1 Result of Naive Bayes implementation - a.Precision Recall and Accuracy b.True Label vs Prediction



4.2 Result of KNN implementation - a.Precision Recall and Accuracy b.True Label vs Prediction



Yolo-V2.0 with KERAS and Tensorflow for Object detection

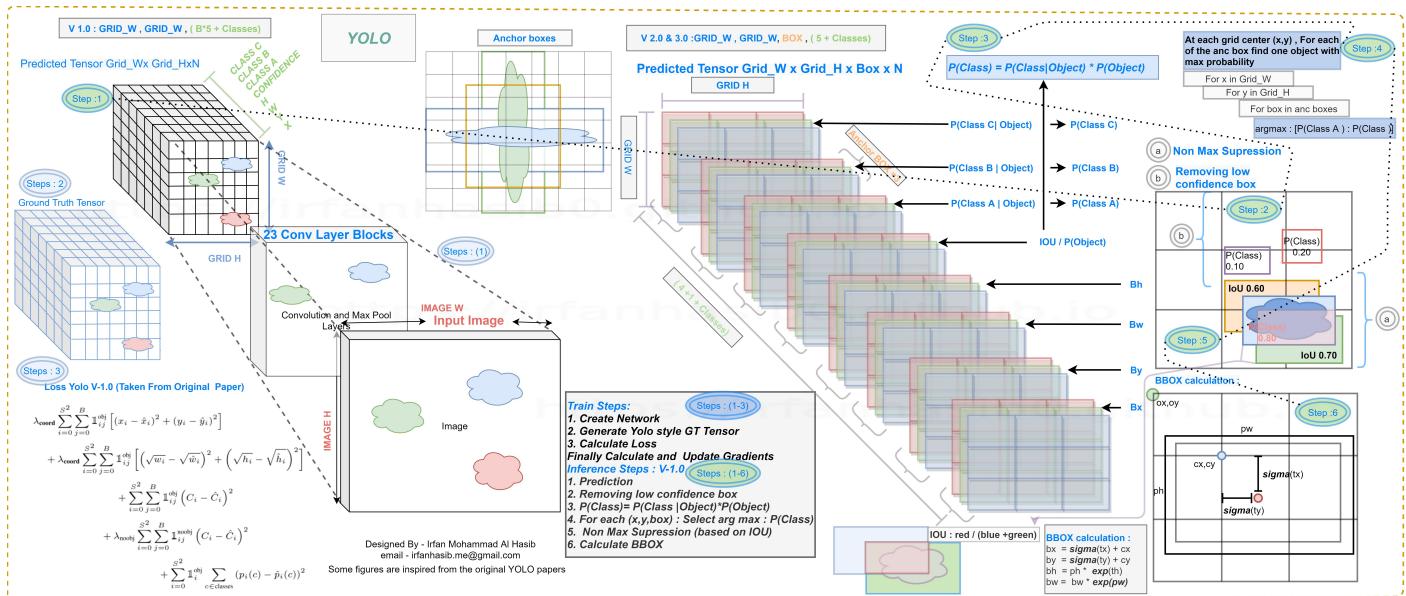
[Go Back to Table of Content](#)

- **Section 1: Overview**
- **Section 2. Flowchart**
- **Section 3. Code**
- **Section 4. Result**

1. Overview

Yolo V-2.0 is implemented here and tested with COCO dataset. It is one of the SOTA algorithm for object detection. The algorithm is showed in the figure in algorithm section. Python code can be found in code section.

2. Flowchart



2. Code YOLO-V2.0 for COCO dataset

[GitHub Link : Yolo-V2 \(\[https://github.com/irfanhasib0/CNN-Projects/blob/master/CNN_Basic/Minimal_yolo_coco-v-2.0-exp-COCO.ipynb\]\(https://github.com/irfanhasib0/CNN-Projects/blob/master/CNN_Basic/Minimal_yolo_coco-v-2.0-exp-COCO.ipynb\)\)](https://github.com/irfanhasib0/CNN-Projects/blob/master/CNN_Basic/Minimal_yolo_coco-v-2.0-exp-COCO.ipynb)

Star

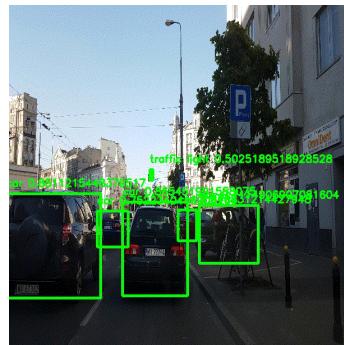
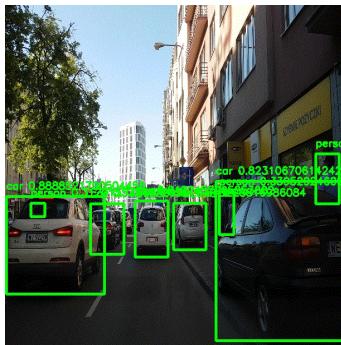
YOLO -V2.0 Implementation

1. Implemented from scratch using tensorflow
2. Tested for COCO dataset

```
In [1]: #!pip install tensorflow==1.12.0
```

```
In [2]: # Initialization
colab_run=False
anc_box= True
_grid_offset=True
train=False
```

3. Results:



U-Net with KERAS for City Space Dataset

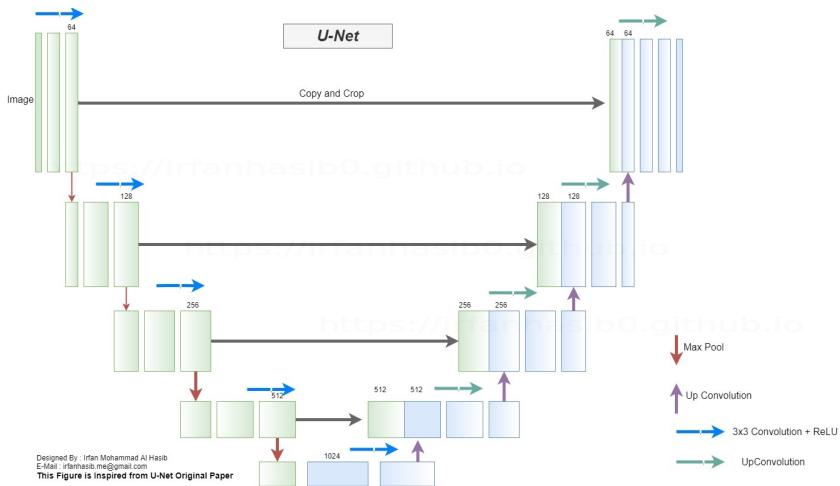
[Go Back to Table of Content](#)

- **Section 1: Overview**
- **Section 2. Network Architecture**
- **Section 3. Code**
- **Section 4. Result**

1. Overview

U-Net is implemented here and tested with CityScapes dataset. It is one of the SOTA algorithm for semantic segmentation. The network architecture is showed in the figure of Network Architecture section. Python code can be found in code section.

2. Network Architecture



3. Code of U-Net for CityScapes dataset

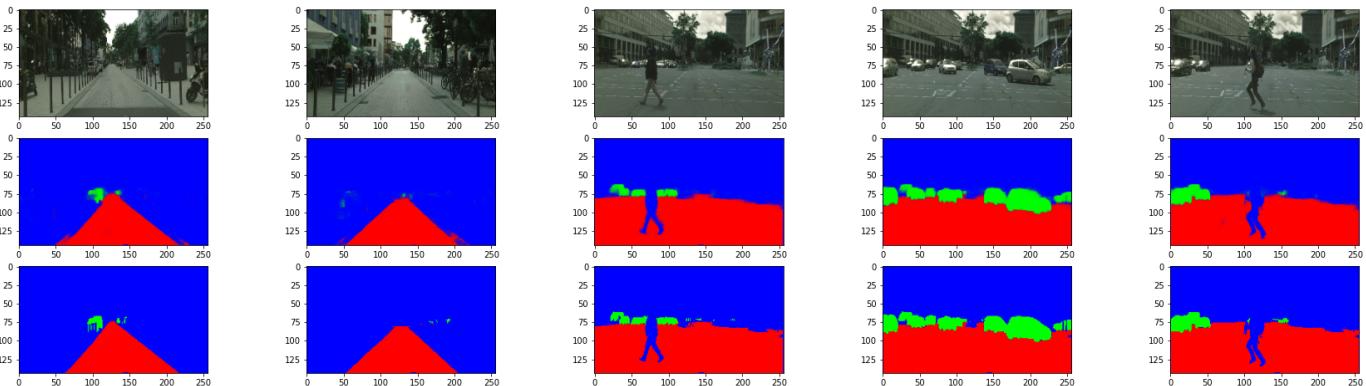
[GitHub :Unet for segmenting City Space Dataset](https://github.com/irfanhasib0/CNN-Projects/blob/master/CNN_Basic/U-Net_cityscapes.ipynb)
(https://github.com/irfanhasib0/CNN-Projects/blob/master/CNN_Basic/U-Net_cityscapes.ipynb).

U-Net Implementation

1. Implemented with Tensorflow and Keras
2. Trained and tested on CityScapes Dataset

```
In [2]: seed=1
import os
os.environ['PYTHONHASHSEED']=str(0)
import random
random.seed(seed)
import numpy as np
np.random.seed(seed)
```

4. Result



Flow-Net with KERAS and Tensorflow

[Go Back to Table of Content](#)

- **Section 1: Overview**
- **Section 3. Code**
- **Section 4. Result**

1. Overview

FlowNet is one of the most popular architecture for optical flow prediction. Here the implementation code and results on KITTI Flow Dataset is presented below

2. Code

[GitHub Link :FlowNet for Optical Flow Kitti Dataset
\(https://github.com/irfanhasib0/CNN-
Projects/blob/master/CNN_Basic/FlowNetS_KITTI_Flow_Data.ipynb\).](https://github.com/irfanhasib0/CNN-Projects/blob/master/CNN_Basic/FlowNetS_KITTI_Flow_Data.ipynb)

Flow-NetS Implementation

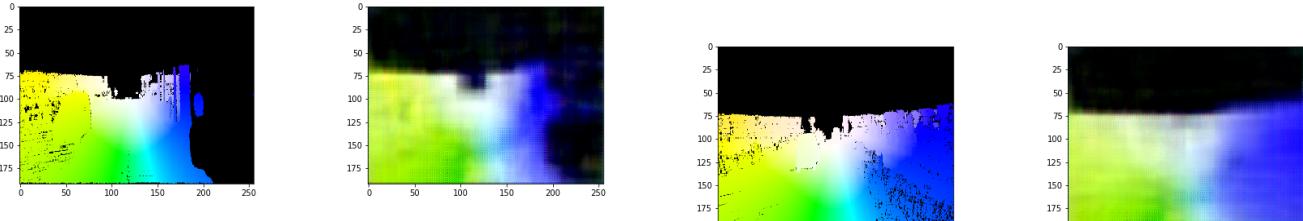
1. Using Tensorflow
2. tested on KITTI Flow Dataset

```
In [1]: train= False
_gen_flow=True
_compress=False
epochs=2000
```

```
In [2]: exp_name='exp-F3\\'
data_root='D:\\KITTI\\'
```

3. Result

Train GT, Train prediction, Test GT, Test Prediction



Monocular Disparity using KERAS and Tensorflow

[Go Back to Table of Content](#)

- **Section 1: Overview**
- **Section 3. Code**
- **Section 4. Result**

1. Overview

Slightly modified Architecture of FlowNet is used for monocular depth estimation. It still performed satisfactorily. The implementation code and results on KITTI Flow Dataset(on disp ground truth) is presented below

Code

[GitHub Link : Single View Depth Estimation \(\[https://github.com/irfanhasib0/CNN-Projects/blob/master/CNN_Basic/FlowNetS_KITTI_Disparity.ipynb\]\(https://github.com/irfanhasib0/CNN-Projects/blob/master/CNN_Basic/FlowNetS_KITTI_Disparity.ipynb\)\).](https://github.com/irfanhasib0/CNN-Projects/blob/master/CNN_Basic/FlowNetS_KITTI_Disparity.ipynb)

Monocular Depth Estimation Implementation

1. FlowNetS architecture was used for depth estimation.
2. Implemented using Tensorflow
3. Tested on KITTI Flow Dataset (KITTI Flow dataset have disparity ground truth also).

```
In [1]: train= False  
_gen_disp=True  
epochs=2500
```

Result

Top row : Train ; Bottom Row : Validation ; Left : Ground Truth ; Right : Prediction



DQN , DDPG, PPO A2C Implementation from scratch

[Go Back to Table of Content](#)

- **Section 1: GitHub Code Links**
- **Section 2: Overview**
- **Section 3. Flowchart**
- **Section 4. PPO Code**
- **Section 5. Result**

1. GitHub Code Links :

- [Multi Arm Bandit - Notebook \(\[https://github.com/irfanhasib0/RL-Algorithms/blob/master/1.0_Multi_Arm_Bandit.ipynb\]\(https://github.com/irfanhasib0/RL-Algorithms/blob/master/1.0_Multi_Arm_Bandit.ipynb\)\)](https://github.com/irfanhasib0/RL-Algorithms/blob/master/1.0_Multi_Arm_Bandit.ipynb)
- [Value Iteration - Notebook \(\[https://github.com/irfanhasib0/RL-Algorithms/blob/master/2.1_Value_Iteration.ipynb\]\(https://github.com/irfanhasib0/RL-Algorithms/blob/master/2.1_Value_Iteration.ipynb\)\)](https://github.com/irfanhasib0/RL-Algorithms/blob/master/2.1_Value_Iteration.ipynb)
- [DQN for Mountain Car - Notebook \(\[https://github.com/irfanhasib0/RL-Algorithms/blob/master/3.0_DQN_Mountain_Car.ipynb\]\(https://github.com/irfanhasib0/RL-Algorithms/blob/master/3.0_DQN_Mountain_Car.ipynb\)\)](https://github.com/irfanhasib0/RL-Algorithms/blob/master/3.0_DQN_Mountain_Car.ipynb)
- [DDPG for Pendulum - Notebook \(\[https://github.com/irfanhasib0/RL-Algorithms/blob/master/4.0_DDPG_Pendulum-mod.ipynb\]\(https://github.com/irfanhasib0/RL-Algorithms/blob/master/4.0_DDPG_Pendulum-mod.ipynb\)\)](https://github.com/irfanhasib0/RL-Algorithms/blob/master/4.0_DDPG_Pendulum-mod.ipynb)
- [PPO for Bipedal Walker - Notebook \(\[https://github.com/irfanhasib0/RL-Algorithms/blob/master/5.1_PPO_Bipedal_Walker-mod.ipynb\]\(https://github.com/irfanhasib0/RL-Algorithms/blob/master/5.1_PPO_Bipedal_Walker-mod.ipynb\)\)](https://github.com/irfanhasib0/RL-Algorithms/blob/master/5.1_PPO_Bipedal_Walker-mod.ipynb)
- [PPO for Lunar Lander - Notebook \(\[https://github.com/irfanhasib0/RL-Algorithms/blob/master/5.2_PPO_Lunar_Lander.ipynb\]\(https://github.com/irfanhasib0/RL-Algorithms/blob/master/5.2_PPO_Lunar_Lander.ipynb\)\)](https://github.com/irfanhasib0/RL-Algorithms/blob/master/5.2_PPO_Lunar_Lander.ipynb)

2. Overview

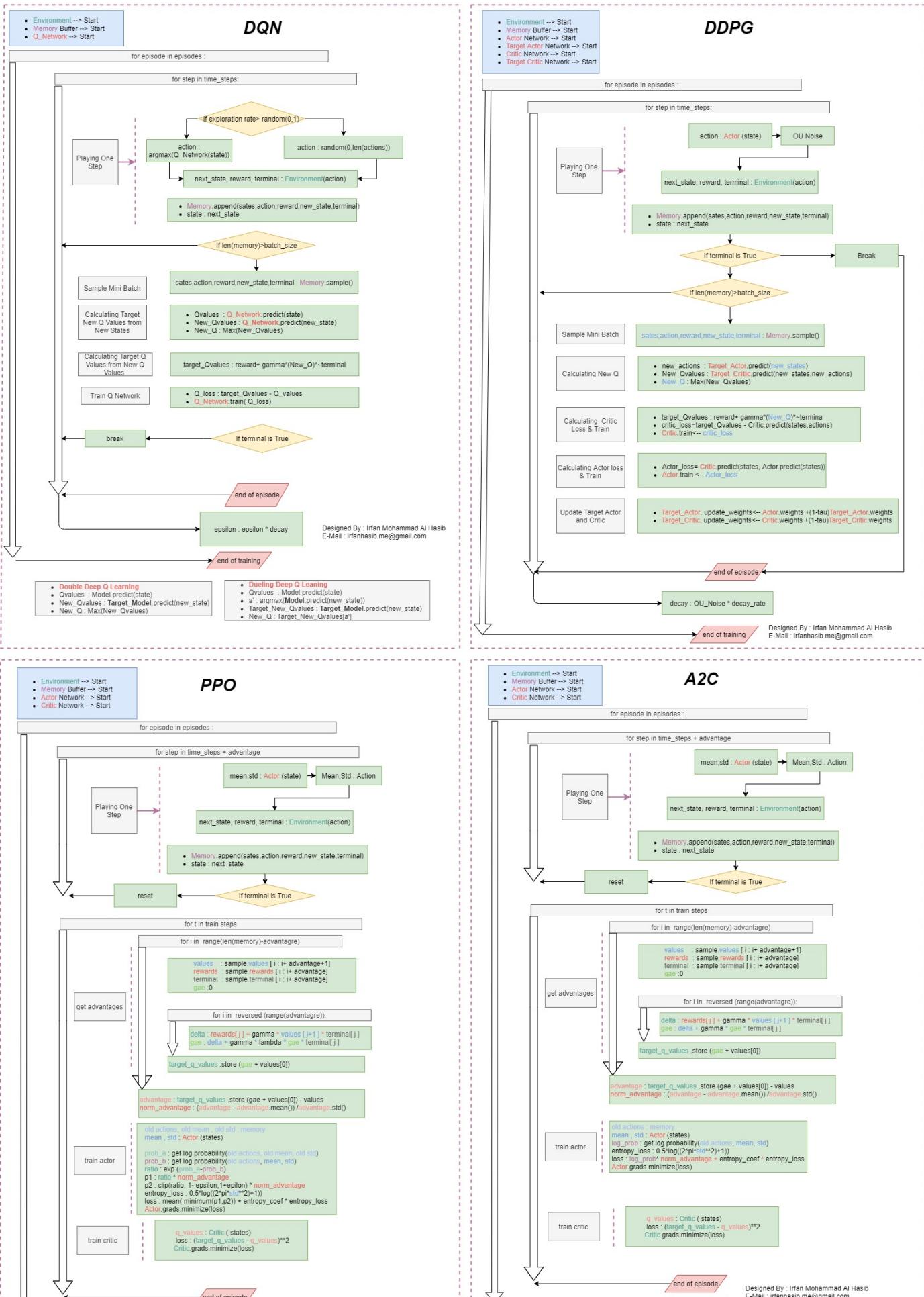
DQN implemented and tested on OpenAI gym Mountain Car ENvironment, DDPG implemented and tested on OpenAI gym Pendulum Environment, PPO implemented and tested on OpenAI gym Bipedal Walker and Lunar Lander Environment. All these algorithms are implemented from scratch using TensorFlow, Numpy and Python.

 Star

3. Detailed Flow Chart for DQN and DDPG : (Please Zoom or Open in New tab for proper resolution)

 Star

4. Code PPO for bipedal environment



Proximal Policy Optimization

1. Implementation from scratch with tensorflow
2. Tested for Bipedal walker environment of OpenAI

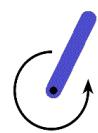
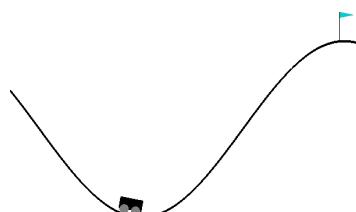
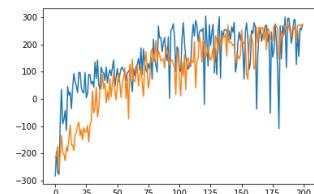
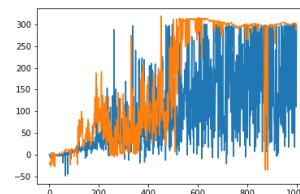
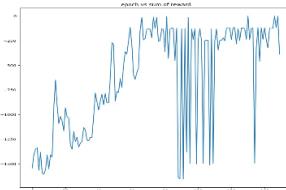
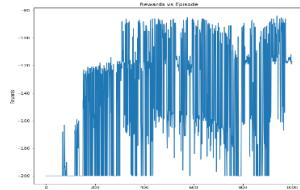
In [1]:

```
seed=1
import os
os.environ["CUDA_VISIBLE_DEVICES"] = "-1"
os.environ['PYTHONHASHSEED']=str(seed)
import numpy as np
np.random.seed(seed)
import tensorflow as tf
tf.random.set_seed(seed)
```

5. Results

* a. Results DQN on Mountain Car * b. Results DDPG on Pendulum * c. Results PPO on Bipedal Walker * d.

Results PPO on Lunar Lander [Go Back to Table of](#)



[Content](#)

House Price Prediction :: Data Pre-Processing and Hiper-Parameter Tuning

[Go Back to Table of Content](#)

- **Section 1: Overview**
- **Section 2: Demonstration of Project Code**
- **Section 3: Flow Chart**

1. Overview :

- *House Price Dataset from kaggle.com*
- *Data Preprocessing*
- *ANN Class with tensorflow low level API*
- *Hiperparameter Tuning*
- *All the graphs of Data preprocessing and Hiperparameter Tuning can be found in [Notebook](#) (https://github.com/irfanhasib0/Machine-Learning/blob/master/Kaggle/ANN_Tensorflow_Kaggle_Houseprice_prediction.ipynb).*

[Click to expand](#)

[Notebook : House Price Prediction - Notebook \(Project Presentation and Code Link\)](#) (https://github.com/irfanhasib0/Machine-Learning/blob/master/Kaggle/ANN_Tensorflow_Kaggle_Houseprice_prediction.ipynb)



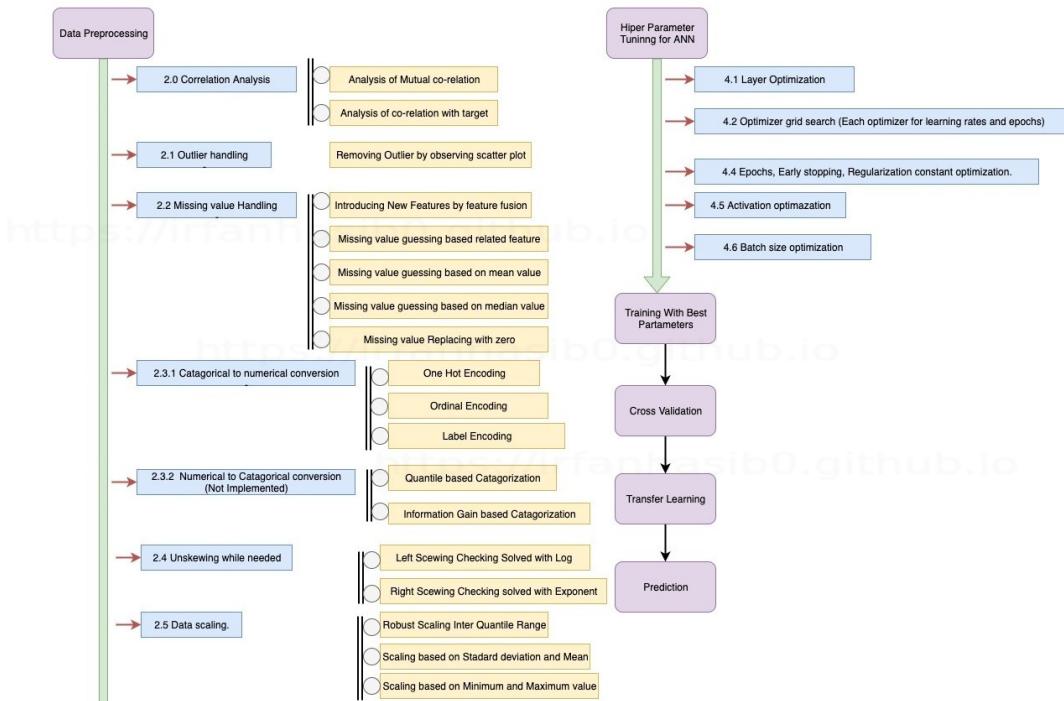
House Prices: Advanced Regression Techniques

Summary :

House price is a kaggle competition which is based on predicting house price from different features of house using machine learning techniques. The following processes are done for solving the house price prediction problem.

1. Data is preprocessed by following steps-

2. Project Flow Chart :



Japanese Job Entrance Problem :: Shakura Bloom Prediction

[Go Back to Table of Content](#)

- **Section 1: Overview**
- **Section 2: Demonstration of Project Code**
- **Section 3: Flow Chart**

1. Overview

- Weather data from japanese meteorological agencym
- Feature Extraction and Data Preprocessing
- ANN Class with tensorflow low level API
- Hiperparameter Tuning
- All the graphs of Data preprocessing and Hiperparameter Tuning can be found in [Notebook](https://github.com/irfanhasib0/Machine-Learning/blob/master/Kaggle/Sakura_TF_NN_Report.ipynb) (https://github.com/irfanhasib0/Machine-Learning/blob/master/Kaggle/Sakura_TF_NN_Report.ipynb).

[Click to expand](#)

[Notebook : Shakura Bloom Prediction - Notebook \(Project Presentation and Code Link\)](#) (https://github.com/irfanhasib0/Machine-Learning/blob/master/Kaggle/Sakura_TF_NN_Report.ipynb)

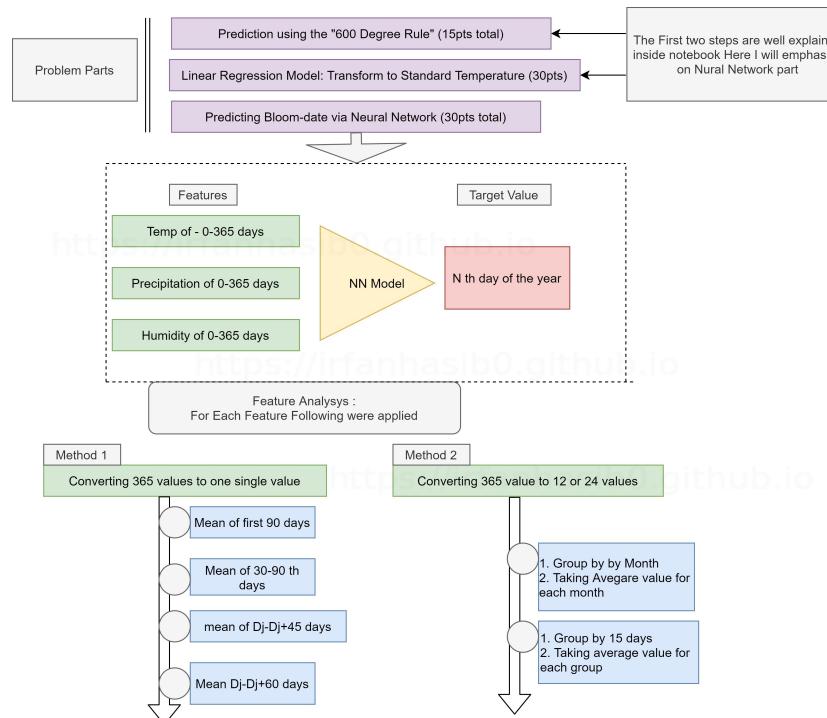
Entrance Challenge: When Will the Sakura Bloom?

Basics of the Sakura Bloom-cycle (5pts total)

In a year, sakura trees basically go through 4 phases: energy production, hibernation, growth, and of course flowering. These phases roughly follow the seasons, but not exactly.

Production phase: Initial development of the buds (Summer-Fall)

2. Project Flow Chart :



ILQR and MPC :: Implementation from scratch for self driving car simulator

[Go Back to Table of Content](#)

GitHub links

- [GitHub : ILQR Implementation - Notebook](https://github.com/irfanhasib0/Control-algorithms/blob/master/MPC_GYM_CAR_RACING_V0/ilqr_on_airsim_env.ipynb)
- [GitHub : MPC Implementation - Notebook](https://github.com/irfanhasib0/Control-algorithms/blob/master/MPC_GYM_CAR_RACING_V0/mpc_on_car_env.ipynb)

1. Overview :

- **Simulation Platform :**
 - AIRSIM by Microsoft Inc.
 - OpenAI GYM - Car Environment was tested
- **IO :**

Input --> Map Points , Output --> Steering Angle, Acceleration, Brake
- **MAP to Trajectory : Steps :**
 - Environment Module
 - Map Tracker Module
 - Data Preprocessor Module
- **Optimization Algorithm : Trajectory to Optimal Steering, Acceleration, Brake :**
 - iLQR using Raw Python and Numpy
 - MPC using Python , Numpy and CVXPY

[Click to expand](#)

MPC modeling

State Space : $z = [x, y, v, \phi]$ where, x : x – position, y : y – position, v : velocity, ϕ : yaw angle
Action Space : $u = [a, \delta]$ where, a : acceleration, δ : steering angle

Cost and Constraints :

Cost :

$$\min Q_f(z_{T,ref} - z_T)^2 + Q\Sigma(z_{t,ref} - z_t)^2 + R\Sigma u_t^2 + R_d\Sigma(u_{t+1} - u_t)^2$$

z_{ref} : target states

Constraints :

$$z_{t+1} = Az_t + Bu + C$$

$$\text{Maximum steering speed} = \text{abs}[u_{t+1} - u_t] < du_{max}$$

$$\text{Maximum steering angle} = u_t < u_{max}$$

$$\text{Initial state} = z_0 = z_{0,ob}$$

$$\text{Maximum and minimum speed} = v_{min} < v_t < v_{max}$$

$$\text{Maximum and minimum input} = u_{min} < u_t < u_{max}$$

State Space model for Car system

$$z_{t+1} = Az_t + Bu + C \text{ where, } A = \begin{bmatrix} 1 & 0 & \cos(\bar{\phi})dt & -\bar{v}\sin(\bar{\phi})dt \\ 0 & 1 & \sin(\bar{\phi})dt & \bar{v}\cos(\bar{\phi})dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{\tan(\bar{\delta})}{L}dt & 1 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ dt & 0 \\ 0 & \frac{\bar{v}}{L\cos^2(\bar{\delta})}dt \end{bmatrix} \quad C = \begin{bmatrix} \bar{v}\sin(\bar{\phi})\bar{\phi}dt \\ -\bar{v}\cos(\bar{\phi})\bar{\phi}dt \\ 0 \\ -\frac{\bar{v}\bar{\delta}}{L\cos^2(\bar{\delta})}dt \end{bmatrix}$$

Expand to see derivation

Iterative Linear Quadratic Regulator :

1. U as a function of Previous U , X and Previous X :

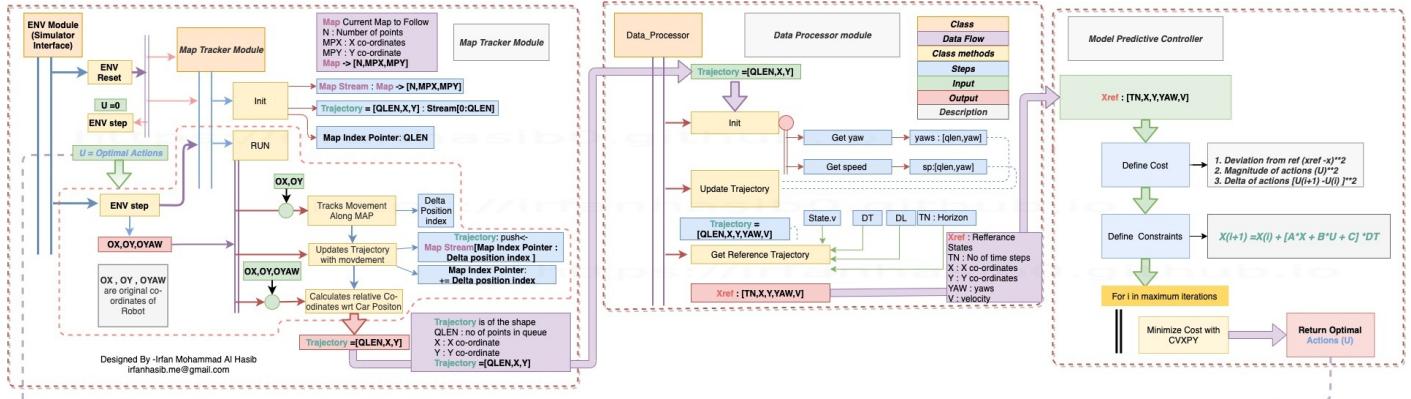
$$u'(i) = u(i) + k(i)(x'(i) - x(i))$$

2. Calculating K and k : (Expand to see)

4. For detail derivation please see the paper :

Synthesis and Stabilization of Complex Behaviors through Online Trajectory Optimization by - Yuval Tassa
citations(369) according to February 27 2020, Published on - 2012

2. Project Flow Chart :

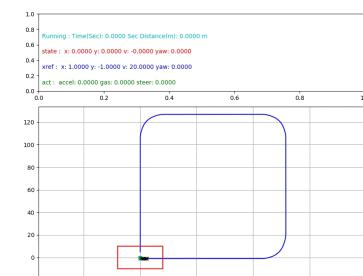
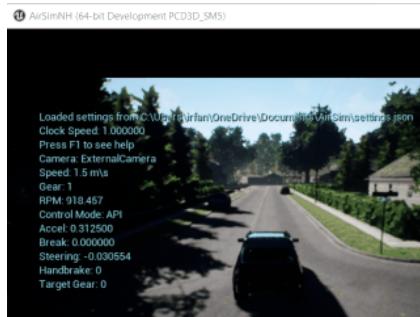
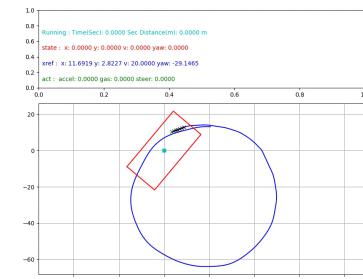
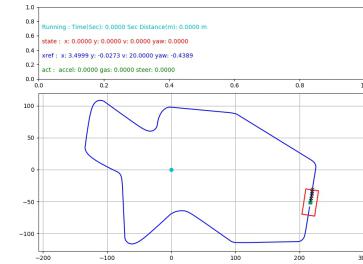
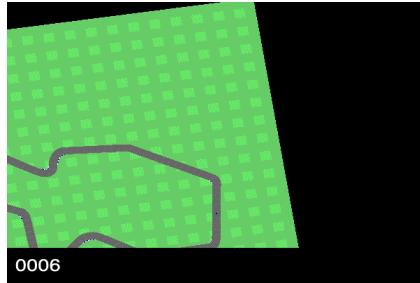


[Click to expand](#)



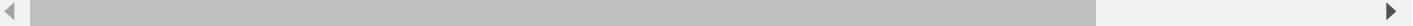
3. Results (ILQR) :

* OpenAI Gym Car Environment * Airsim City Space Environment * Airsim Neighbourhood Environment



Inspired from -

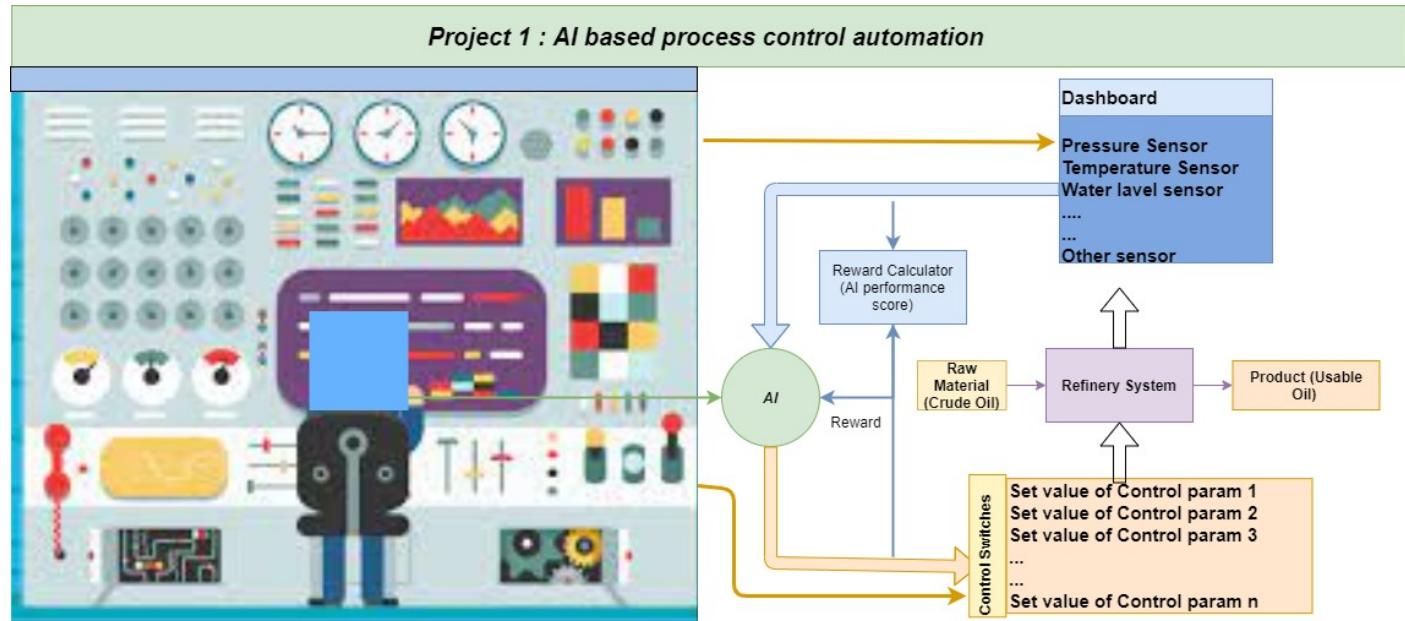
AtsushiSakai/PythonRobotics

Reference

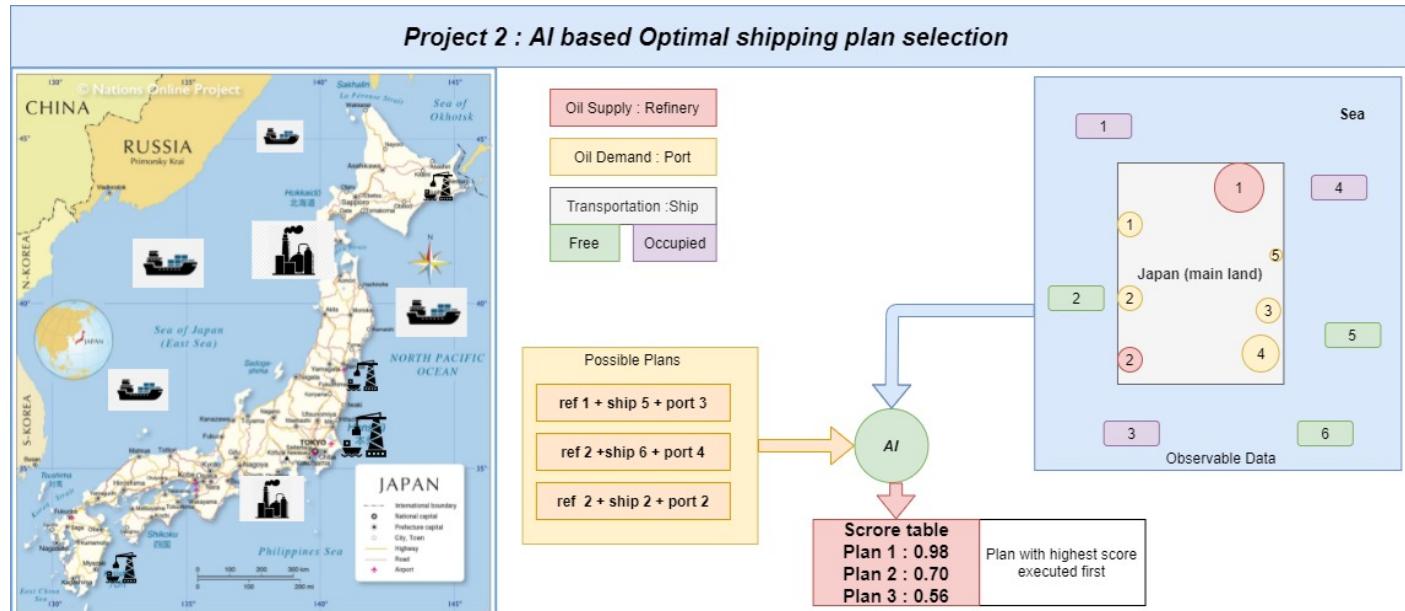
Professional AI Solutions

[Go Back to Table of Content](#)

These are some of my professional AI solutions. Just the outline of the projects (without any technical information) is shared for confidentiality reasons.

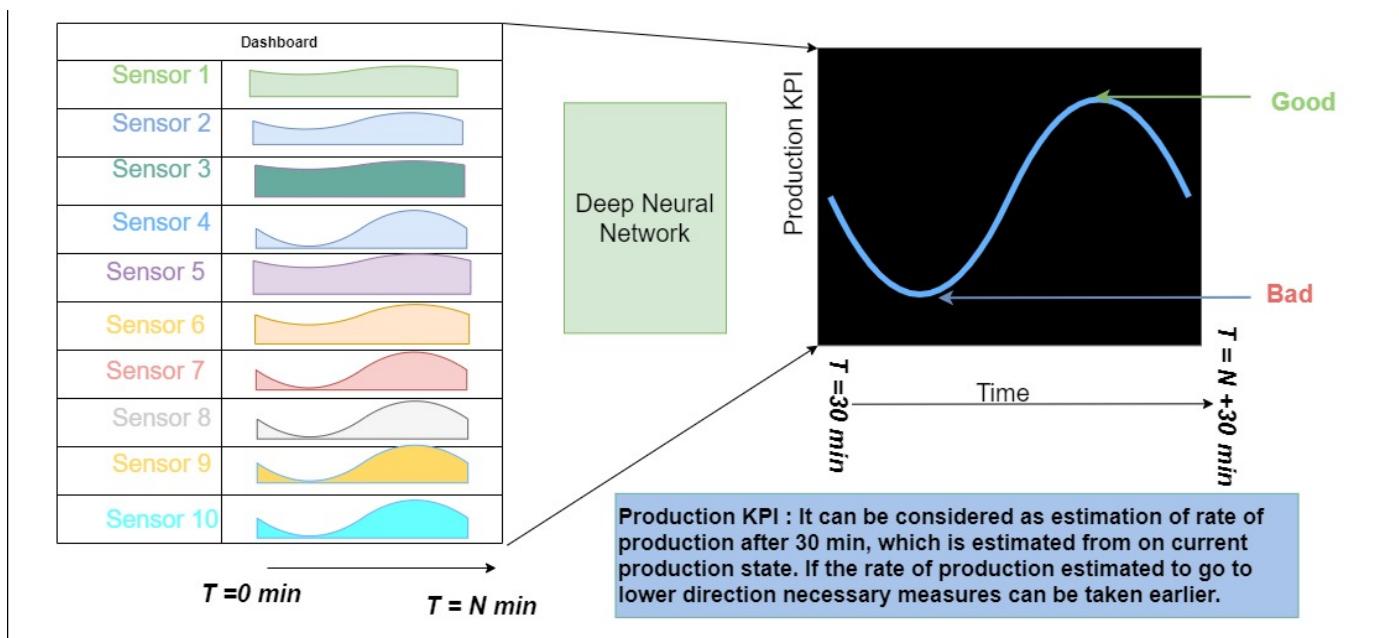


[Go Back to Table of Content](#)

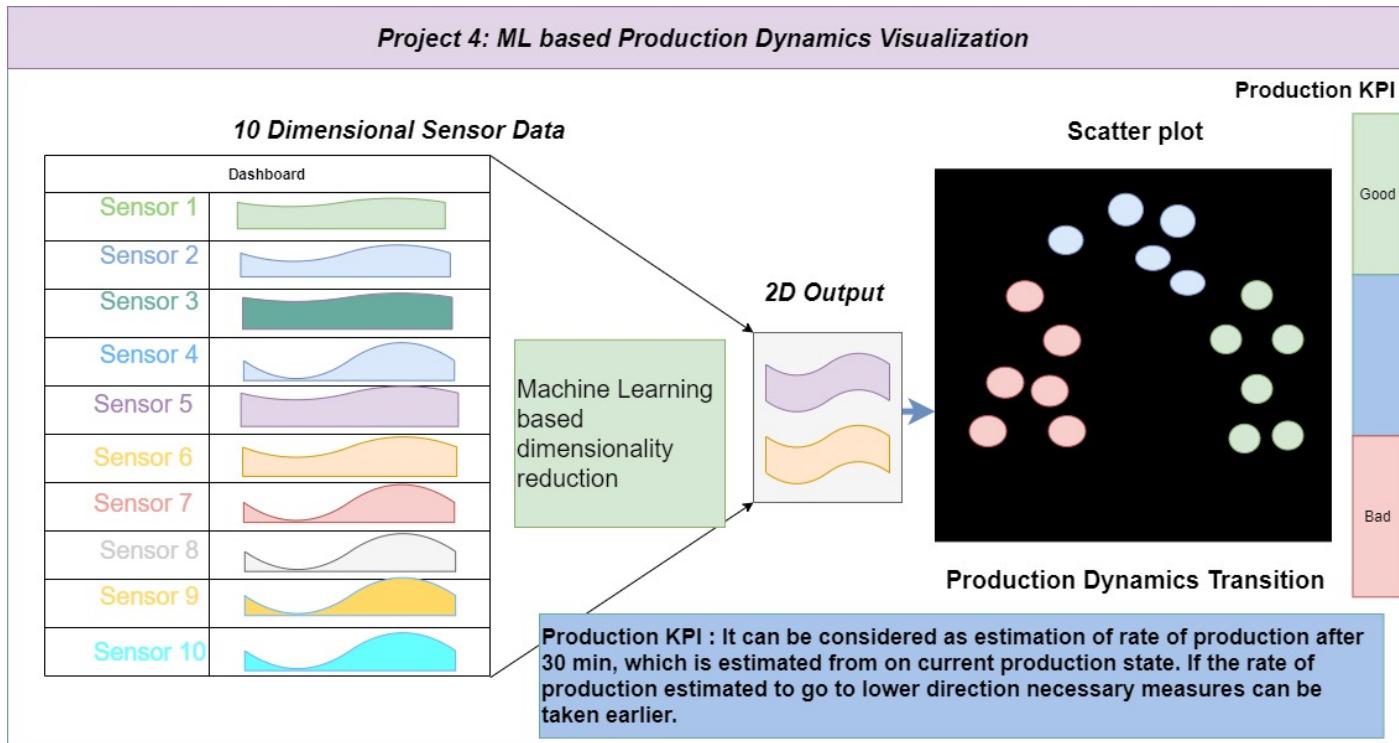


[Go Back to Table of Content](#)

Project 3: DL based Production KPI Estimation



[Go Back to Table of Content](#)



[Go Back to Table of Content](#)

Professional Embedded System Projects

[Go Back to Table of Content](#)

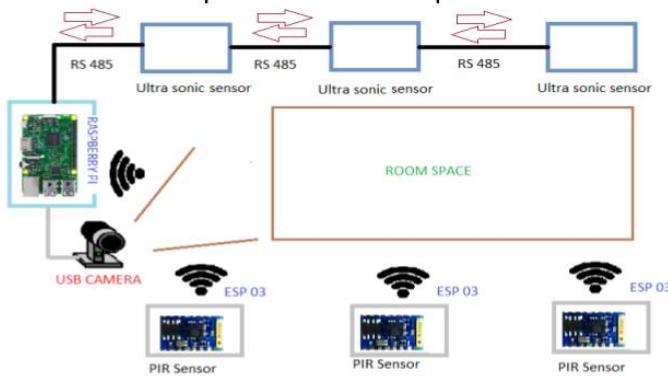
All these projects I did as an employee of Pi Labs BD Ltd. www.pilabsbd.com

VAULT SECURITY SYSTEM - PI Labs BD Ltd.

I was co-developer of vault security system v-1.0

I have developed -

- ❖ The wireless PIR (passive infrared sensor) sensor nodes with ESP 03 (ESP 8266 based Wi-Fi transceiver module) for sending data to raspberry pi server.
- ❖ Motion detection with USB camera on raspberry pi
- ❖ Sending Ultra sonic sensor data to raspberry pi via RS485 protocol.
- ❖ Worked on low power modes of operation of ESP 03.

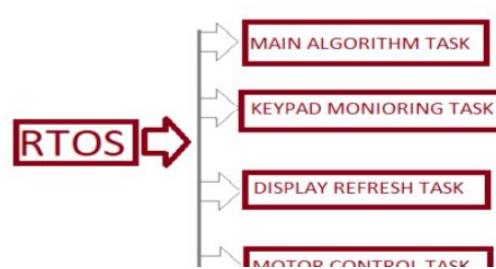


[Go Back to Table of Content](#)

SYRINGE INFUSION PUMP Pi Labs Bangladesh Ltd.

I was main developer of syringe infusion pump.

- ❖ The pump was programmable for pumping liquid at desired rate for desired time period.
- ❖ The system was designed on AVR platform using RTOS (Real time operating system).
- ❖ Keypad and Display for programming the pump.
- ❖ Stepper motor controlling with TB6560 driver with 1/32 micro step controlling.
- ❖ 3 modes of operation. They are – Automatic Mode, Manual Mode and Refill Mode.
- ❖ Pumping volume precision up to 0.01 cc/s



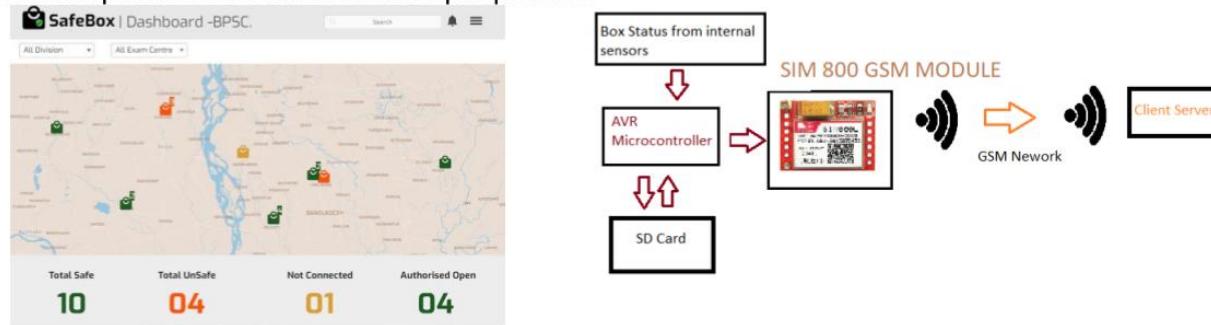


[Go Back to Table of Content](#)

SafeBox – Box tracking Device for Pi Labs BD Ltd.

I was co-developer of vault Safe Box v-1.0 a box tracking and box safety status reporting device.

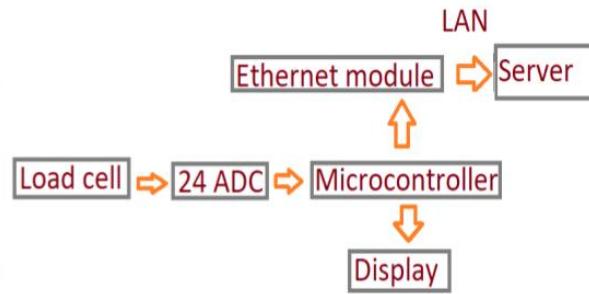
- ❖ I have worked on sending data packet to the server at regular interval via GSM network with SIM800 with least data loss.
- ❖ I have developed an algorithm for keeping back up of box status data on SD card (specially when GSM network strength is low) with FATFS on AVR platform by SPI interfacing with SD Card and reporting backup to server when network is back again.
- ❖ I have also worked on the battery backup time enhancement and researched on power consumption of SIM800 and other peripherals.



[Go Back to Table of Content](#)

Online weight machine for Pi Labs Bangladesh Ltd.

- The weight machine was developed on AVR microcontroller platform.
- The weight machine capable of showing up to +/- 3 g precision using 24 bit ADC(Analog to digital conversion).
- The weight machine by e capable of sending measured weight directly to server by using LAN connection.
- I have also developed prototype of version two with native raspberry pi server with python-flask and Wi-Fi support.



ROS : Custom 2 Link robot robot inspired from rrbott

[Go Back to Table of Content](#)

I have developed a simple 2 link robot inspired from [rrbot \(\[http://gazebosim.org/tutorials?tut=ros_urdf\]\(http://gazebosim.org/tutorials?tut=ros_urdf\)\)](http://gazebosim.org/tutorials?tut=ros_urdf). It is capable of obtaining Laser Scan values from laser sensor and Camera feed from both overhead camera and camera attached to arm front. I have provided the URDF and Gazebo config files and python driver code for publishing driving command and listening to sensor values.

- [Simple 2 Link robot : Driver \(Python3\)](#)
[\(\[https://github.com/irfanhasib0/ros_ws/tree/master/src/run_gui_hk.py\]\(https://github.com/irfanhasib0/ros_ws/tree/master/src/run_gui_hk.py\)\)](https://github.com/irfanhasib0/ros_ws/tree/master/src/run_gui_hk.py).
- [Simple 2 Link robot : description file \(URDF\) Link](#)
[\(\[https://github.com/irfanhasib0/ros_ws/tree/master/src/rrbot/rrbot_description\]\(https://github.com/irfanhasib0/ros_ws/tree/master/src/rrbot/rrbot_description\)\)](https://github.com/irfanhasib0/ros_ws/tree/master/src/rrbot/rrbot_description).
- [Simple 2 Link robot : Controller Code\(Launch file\) Link](#)
[\(\[https://github.com/irfanhasib0/ros_ws/tree/master/src/rrbot/rrbot_control\]\(https://github.com/irfanhasib0/ros_ws/tree/master/src/rrbot/rrbot_control\)\)](https://github.com/irfanhasib0/ros_ws/tree/master/src/rrbot/rrbot_control).
- [Simple 2 Link robot : Gazebo config Link](#)
[\(\[https://github.com/irfanhasib0/ros_ws/tree/master/src/rrbot/rrbot_gazebo\]\(https://github.com/irfanhasib0/ros_ws/tree/master/src/rrbot/rrbot_gazebo\)\)](https://github.com/irfanhasib0/ros_ws/tree/master/src/rrbot/rrbot_gazebo).
- [Simple 2 Link robot : YouTube Link](#) ([<https://youtu.be/lJbyy89X7gM>](https://youtu.be/lJbyy89X7gM)).

Ros Gazebo Simple Arm Laser S...

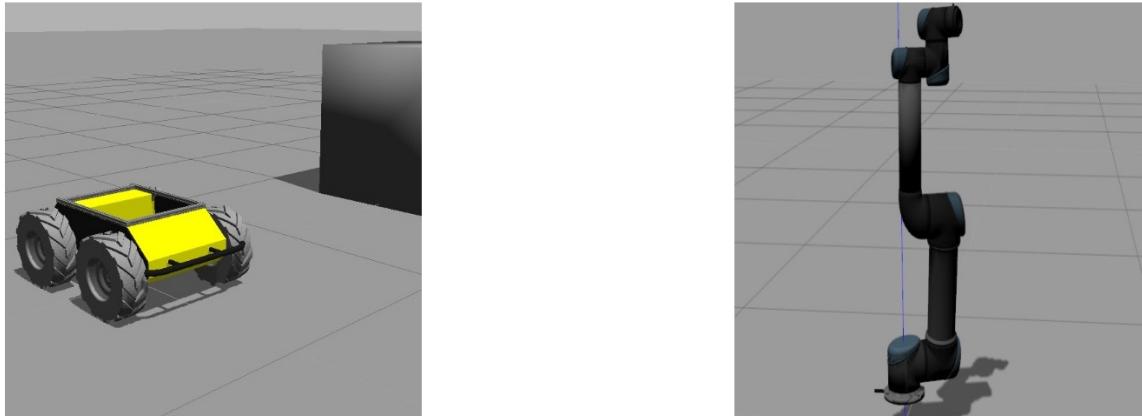


ROS : Husky, URbot driver in Python

[Go Back to Table of Content](#)

Husky is a tarain travelling robot which is developed by [Clearpath Robotics](https://clearpathrobotics.com/husky-unmanned-ground-vehicle-robot/) (<https://clearpathrobotics.com/husky-unmanned-ground-vehicle-robot/>). URbot(UR-5) is one of the robotic arms developed by [Universal-Robots](https://www.universal-robots.com/products/ur5-robot/). (<https://www.universal-robots.com/products/ur5-robot/>) Below are the GitHub link of the basic driver code capable of publishing driving commands and listening to sensor valued using ROS which I have written for them.

- [Husky Controller\(Lunch File\)](#)
[\(\[https://github.com/irfanhasib0/ros_ws/tree/master/src/husky_mod/husky_cor\]\(https://github.com/irfanhasib0/ros_ws/tree/master/src/husky_mod/husky_cor\)\)](https://github.com/irfanhasib0/ros_ws/tree/master/src/husky_mod/husky_cor).
- [Husky Driver \(Python3\)](#)
[\(\[https://github.com/irfanhasib0/ros_ws/tree/master/src/run_gui_hk.py\]\(https://github.com/irfanhasib0/ros_ws/tree/master/src/run_gui_hk.py\)\)](https://github.com/irfanhasib0/ros_ws/tree/master/src/run_gui_hk.py).
- [Universal Robots \(UR5\) Driver \(Python3\)](#)
[\(\[https://github.com/irfanhasib0/ros_ws/tree/master/src/run_gui_ur.py\]\(https://github.com/irfanhasib0/ros_ws/tree/master/src/run_gui_ur.py\)\)](https://github.com/irfanhasib0/ros_ws/tree/master/src/run_gui_ur.py).



University Rover Challenge - 2016

[Go Back to Table of Content](#)

Along with my team, Interplaneter I have participated in [University Rover challenge \(<http://urc.marsociety.org/>\)](http://urc.marsociety.org/), 2016 at Utah, USA. Our team attained 5th position in Phobos final. I was in charge of Robotic Arm Design and deployment. The Competition is organized by [Mars Society \(<https://www.marsociety.org/>\)](https://www.marsociety.org/), USA annually for college students world wide. [URC 2016 Result \(<http://urc.marsociety.org/home/about-urc/urc2016-scores>\)](http://urc.marsociety.org/home/about-urc/urc2016-scores), video link [YouTube \(<https://www.youtube.com/watch?v=MIN-VFj14LE>\)](https://www.youtube.com/watch?v=MIN-VFj14LE).



ERC 2016



Academic Project and Thesis:

[Go Back to Table of Content](#)

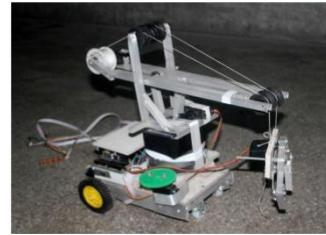
[Thesis Book \(\[https://github.com/irfanhasib0/irfanhasib0.github.io/blob/master/academic_thesis/thesis-book.pdf\]\(https://github.com/irfanhasib0/irfanhasib0.github.io/blob/master/academic_thesis/thesis-book.pdf\)\)](https://github.com/irfanhasib0/irfanhasib0.github.io/blob/master/academic_thesis/thesis-book.pdf). (loading can be slow, In that case download will be faster)

- My undergrad project of instrumentation and measurement course.
- My undergrad thesis.

Academic Project and Thesis

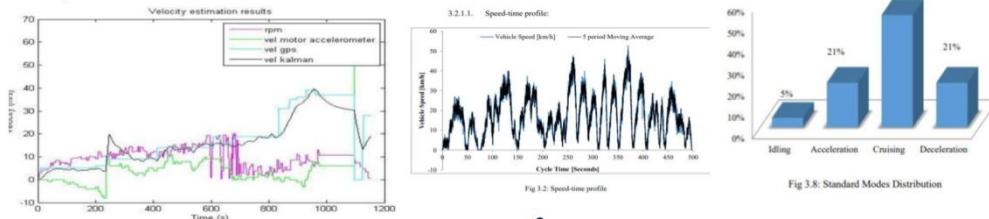
Level-3 Term-1 project-Remote Controlled Surveillance Robot:

- ❖ Development of a Remote control Surveillance bot Hardware and Software Platform for soil sample collection and data acquisition and transmission.



Level 4 -Thesis

- ❖ Development of a precision vehicle speed measurement and data logging system for making a Drive Cycle of Bangladesh. (A study of precision velocity measurement system including Inertial Measurement system (IMU), GPS and sensor fusion with kalman filter.)



[Go Back to Table of Content](#)

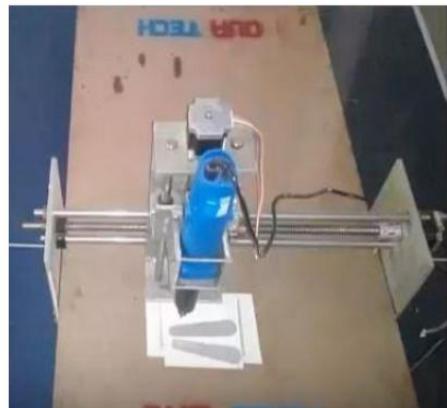
Embedded System & Robotics Projects Personal (Undergrad) :

[Go Back to Table of Content](#)

- Hobby CNC machine 'Ourtech v 2.0' and 'Ourtech v1.0', Desktop CNC Machine.[Video Link](https://www.youtube.com/watch?v=xU7YMPpZMYS) (<https://www.youtube.com/watch?v=xU7YMPpZMYS>)
- Interfacing ov7670 camera sensor with atmega 32 and using object tracking algorithm on AVR platform (Feb-May - 2013)[Video Link](https://youtu.be/tKWbYbAJSEs) (<https://youtu.be/tKWbYbAJSEs>)
- Software platform for controlling Robotic arm with openCV, python and Raspberry pi (Apr-Sep,2014)[Video Link](https://www.youtube.com/watch?v=hj1Wc6-8-7w) (<https://www.youtube.com/watch?v=hj1Wc6-8-7w>)

Desktop CNC Machines

- I have developed 2 Desktop CNC machines as my hobby project.
- Both were capable of cutting 2D designs from G-Code from Computer via Serial.
- I have used GRBL Firmware(Open source) with AVR microcontroller.
- I have also designed H-brige drivers for stepper motors with mosfet.



11

[Go Back to Table of Content](#)

OBJECT TRACKING WITH ROBOTOC ARM ON AVR PLATFORM

- ❖ Object Tracking with minimal hardware. i.e. 8 bit microcontroller and camera sensor.
- ❖ Interfacing OV-7670 Camera sensor with Atmega-32 microcontroller using SCCB (Serial Camera Control Bus).
- ❖ Implementing Sobel filter on AVR platform.
- ❖ Note : Object tracking can perform far better on computer platform or raspberry pi but I wanted to build the system with 8 bit microcontroller and camera sensor.

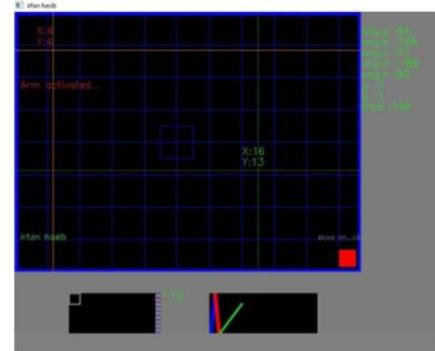


9

[Go Back to Table of Content](#)

Software platform for controlling robotic ARM

- ❖ Developed a User Interface for Controlling a robotic arm with – Joystick / Mouse/ Keyboard.
- ❖ UART to USB support was added for connecting PC with microcontroller that is controlling the servo motors.
- ❖ Developed Inverse Kinematics calculations for robotic arm for converting xy co-ordinates to servo angles.
- ❖ Added robot pose visualization support.
- ❖ The GUI was developed with Tkinter and OpenCV.



10

[Go Back to Table of Content](#)

In []: