Full stack Data Science Systems

**POTHOLE DETECTION SYSTEM**

**(AI SECURE)**

**Group - 2**

**Team Members:**

Saamarth Rastogi

Dhrumil Jayeshbhai Patel

Vishwas Chandran

Soham Mane

Nandini Manyam

Kavish Shah

[**Introduction:**](#_3uatdapq6ped) **2**

[**Problem Statement:**](#_n6jt3ga105mp) **2**

[**Business Outcome:**](#_kkqhsw6aij85) **2**

**Project Flow 2**

[**Benchmarking the Models:**](#_t9nm0wyt0l08) **3**

[SSD:](#_mbaofuc2njiy) 3

[Faster RCNN:](#_8reitsdgg4v0) 3

[Yolo V5:](#_hsw38g9lzbu5) 3

[Conclusion:](#_kzu7cmohansa) 4

[**Developing Web Application:**](#_4w6zwbanjxfq) **4**

[**Docker:**](#_6c32tsojfo3t) **5**

[Azure Container Registry:](#_g5lnda2xua39) 6

[Azure CLI:](#_xoc0dhm93hi) 6

**Results 7**

**Challenges Faced 8**

# Introduction:

AI SECURE, as an organization, we present the project pothole detection. The main goal here is to detect potholes, as the name suggests, eventually preventing and reducing any accidental risks. Data is acquired by scraping the web and some kaggle databases and finally merging them. Some portions of the data set are manually annotated. After that, it is preprocessed with the help of the roboflow platform. Finally, we have used multiple models such as Faster RCNN, SSD, and YOLO v5. deploying the model and bridging it to the front end in the final stage.

# Problem Statement:

In Ontario, while 90% of car accidents are caused by driver error, the other 10% are caused by poor road conditions. On an average 150,000 potholes are filled yearly only in Toronto city. Safety being the utmost priority of every human being while driving, the value of detecting and avoiding potholes can be identified.

# Business Outcome:

To understand the whole scenario, we can present the pothole detection model. After multiple trials and errors of different models and optimizers, the final outcome has shown a satisfactory result of detecting potholes, because of which we can help the automobile vehicle to steer its way around the pothole, avoiding financial damage to the car and, in some cases, physical damage to the human.

**Project Flow**

Our main goal is detecting potholes. Here The problem type is object detection. There are pretrained models available for object detection. We are taking dataset from kaggle and website. Some images are annotated but some images are annotated .We are annotating those images using roboflow. Then we passed this dataset to ssd ,faster rcnn ,yolov5 pretrained models and benchmarked models based on performing on new unseen data.

Once the deep learning model is ready , we connect it to the backend. Backend helps to interact with the model,to execute logic and process behind the application, to send test samples and get results from the model. Then we create frontend pages for making more beautiful, interactive and displaying the predicted image, video on web screen.

Once the Web Application is ready, a docker image is generated. Docker is a container that has everything software needs to run like libraries, systems, tools, code and other. It allows us to build, test and deploy applications quickly. Then we deploy this into cloud like azure,aws, pythonanywhere so users, carsystem in our case can run that website/port number and use build applications to detect potholes.

# Benchmarking the Models:

CNN models are the best when it comes to image datasets and performing object detection. From the initial phase, we knew not to create a CNN model from scratch due to the given short timeline and also the known fact that we might need to use a massive amount of data to train the model, which would also consume lots of time, and even after that, the efficiency/accuracy of the model might not be up to the mark. So we decided to go with the pre-trained models, keeping in mind the criteria of time efficiency, less complexity, and technical efficiency.

As per our **Use-Case** we have selected three viable options to proceed with, after some thorough research and analysis:

* SSD(Single Shot Detection)
* Faster-RCNN
* YOLO V5

## SSD:

We observed that SSD is really quick to detect images due to its single pass object detection characteristics, and also accurately detects larger objects and can even outperform Faster RCNN in this case. But unfortunately, in the real world, all objects are not large enough.

## Faster RCNN:

Although Faster-RCNN is slow compared to SSD as it uses two phases for object detection, at the same time using RPN(Region Proportion Network) to detect ROI and then using a separate search method to find the region of proposal. It can detect both small and large objects accurately when compared to SSD.

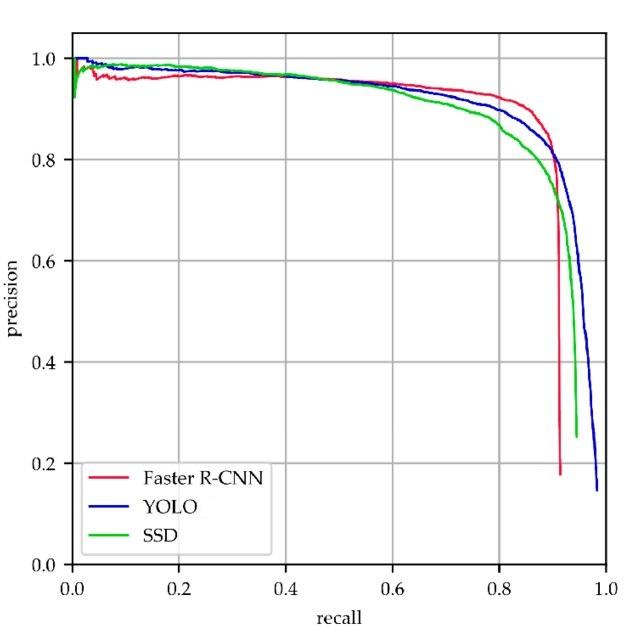
## Yolo V5:

YOLO(You Only Look Once) V5, is a CNN model, similar to SSD in terms of single phase of detection, but much quicker than both faster-rcnn and SSD, and not only that, but it also has remarkable object detection and classification accuracy. It can detect small or far off objects and large or close up objects very accurately.

The YOLO V5 model runs about 2.5 times faster than **Faster-RCNN** while managing better. performance in detecting smaller objects. The results are also cleaner, with little to no overlapping boxes.

## Conclusion:

We decided to go with Yolo V5, as it turns out to be the best when it comes to less complexity, time efficiency, and technical efficiency. The recall of Yolo v5 is better than both the other models. We are considering the recall factor to dominate here, as we need to reduce False Negatives(FN).

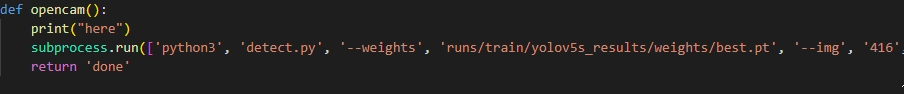


**Fig: Comparative study of the Recall vs. Precision graph for all three models.**

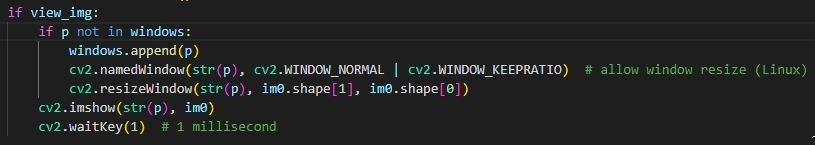
# Developing Web Application:

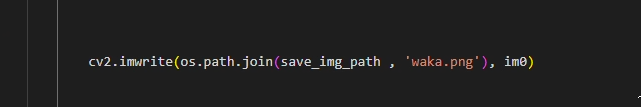
Django and Flask can be used for this machine learning model. Since the initial slug size of the entire project is 1.6 GB, Flask can be used as it is lighter than Django.

Using the **opencam** function in the flask app, a web camera can be launched to start the prediction. This function executes a detection python file that uses best weights.

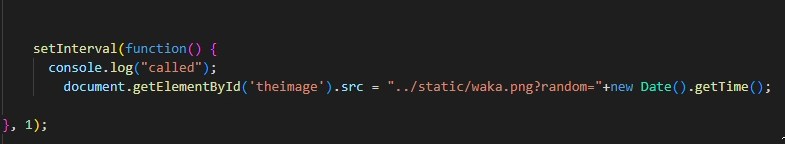


After executing, the image frames can be saved in a folder for later use, as well as shown in the webapp.





Using a Java Script function, each frame of the prediction output can be shown on the web page.



# Docker:

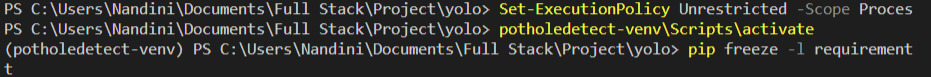
Docker is an open source platform that empowers developers to build, deploy, run, update, and manage containers.Containers are standardised, executable components that combine application source code with the operating system (OS) libraries and dependencies necessary to run that code in any environment.

Steps Included:

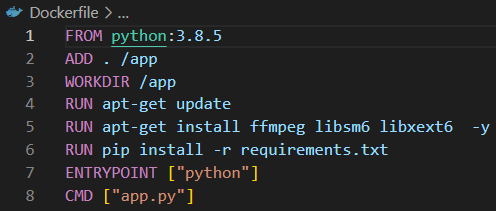
* Creating a virtual environment



* Producing requirements.txt file



* Adding Dockerfile



* Build Docker image using Docker file



* Run it locally



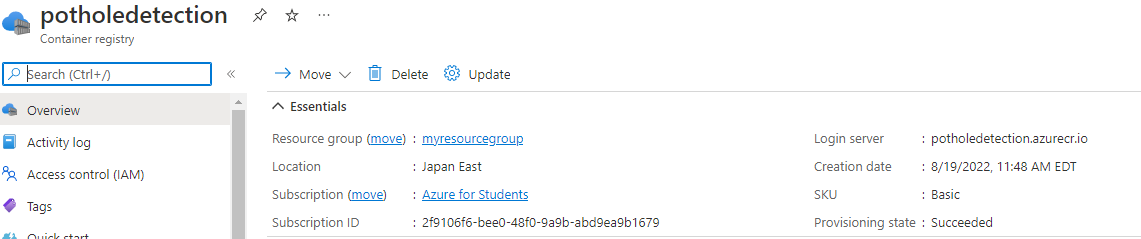
## Azure Container Registry:

Microsoft's own hosting service for Docker images is called the Azure container registry. You would manage private Docker container images and other relevant artefacts in this private registry. These images may then be downloaded and used locally, or they can be deployed via containers to a hosting platform.

## Azure CLI:

A cross-platform command-line tool called the Azure Command-Line Interface (CLI) allows users to login to Azure and run administrative commands on its resources. It enables the use of interactive command-line prompts or scripts to execute commands through a terminal.

* Create Azure Container Registry



* Tag Docker image to the loginl server of Azure Container Registry



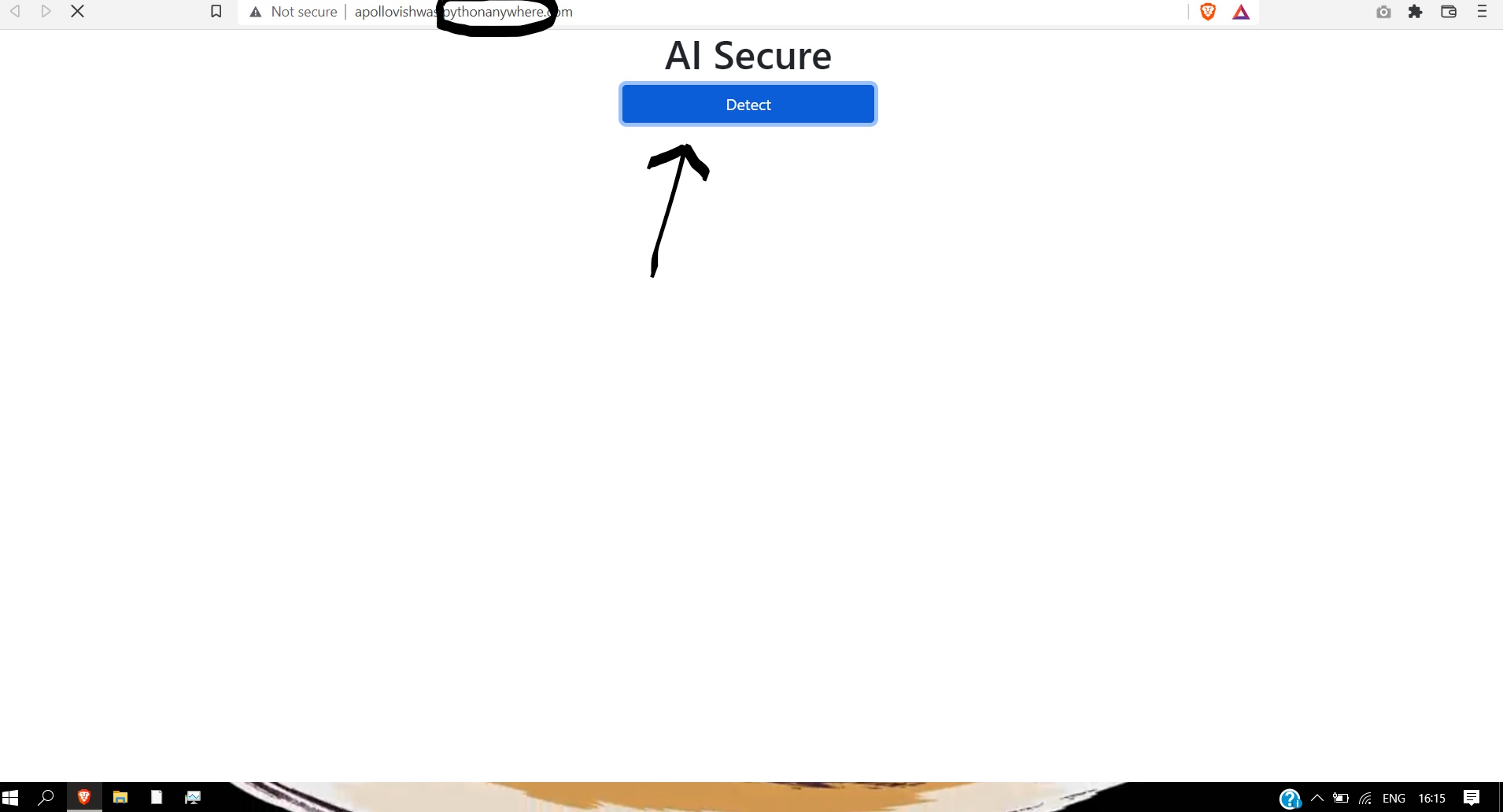
* Push Docker image to Registry using Azure CLI



* We deployed our initial application in azure servers and then tested the application. Due to the exhaustion of our credits on azure we found a different platform to deploy our application so we deployed our final application in python anywhere. Pythonanywhere is an online development environment and web hosting service on python language. Pythonanywhere has the benefit that it is easy to deploy, charges are less (cheaper) compared to other cloud platforms. Pythonanywhere’s performance is also good .So we deployed web application in pythonanywhere.

# Results:

## CLOUD



The above screen is the front end prototype GUI of our website where a user can click on the “DETECT” button and use the pothole detection

It can also be seen that the model is fully connected with the pythonanywhere’s cloud server where our model is deployed

## WEBAPP TESTING

During the initial stages of the project we instead of using a real time camera application we uploaded images manually to check if the pothole is correctly detected or not and the results we got were pretty promising



## REAL TIME LIVE DETECTION

After completely optimising the model for its best performance and deploying it to the cloud we tested our model on the web app which was a successful attempt as the potholes were correctly detected with a accuracy over 83%



# Challenges Faced

The major challenges that we faced during our whole timeline for the projects were

## Data annotation

The data that we chose were images from kaggle and from web which led to half of the images not being annotated.So each of the image had to be annotated manually.We used roboflow platform to annotate the unannoated images

## Model Accuracy:

The main focus which building a model was to make a model which was less complex,time efficient and technically efficient

To obtain these result we ran a lot of experiments which was time consuming considering the time window that we were provided with

## Embedding Camera to the front end

While using the front end we faced a problem where whenever the button to detect the pothole was clicked a different window was opened accessing the camera which was not optimal for a webpage

To solve this error we instead of directly streaming the video to the front end GUI sent each frames from the video which inturn made it look like a seamless video

## Accessing Camera Direct from Cloud

One of the major challenge that is still unexplored is accessing the local camera to be accessed by the cloud platform where our model is running

We are still finding a way to solve this problem which can further improve the scope of our project