

# PROJECT-03: IMAGE TAGGING AND ROAD OBJECT DETECTION

By Group 4

GitHub Link: [https://github.com/KalyanaVarma/AIML\\_Project03](https://github.com/KalyanaVarma/AIML_Project03)

## STEPS



# Image Tagging and Road Object Detection

## STEP 1

Understand the problem statement. Search the best methods and technologies to implement the solution for the problem.

## STEP 2

Analyze and understand the data. Try to match the data and the methods, technologies to implement the solution for the problem. Understand the required eco system and the resources to implement the solution.

## STEP 3

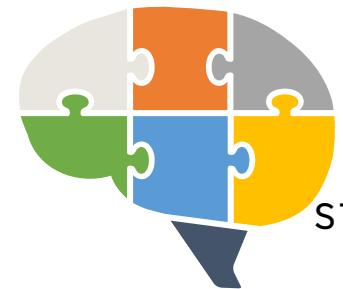
Freeze the method to implement the solution based on the findings from Step-01 to Step-02.

## STEP 4

Go through and understand the research papers, algorithms, architectures to implement the solution for the problem.



## STEPS



# Image Tagging and Road Object Detection

## STEP 5

Get the necessary tools downloaded and installed, for example, LabelImg. Implement code that makes the things easy, for example, 01\_Video\_to\_Frames.py, 02\_Frames\_to\_Video.py etc.

## STEP 6

Have a proof of concept with the available resources and make sure the solution can be implemented.

## STEP 7

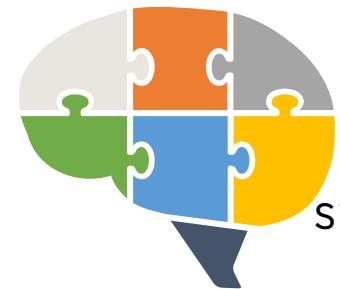
Get the required eco system to implement the solution for the problem like Google Colab Pro subscription, Google One i.e. Google Drive extra storage space etc.

## STEP 8

Create User accounts and Logins for the repository and deployment platforms like GitHub and Heroku.



## STEPS



# Image Tagging and Road Object Detection

STEP 9

Train and test the model. Get the performance metrics. If required, finetune and re-train the model to improve the performance.

STEP 10

Test the model with unseen data and make sure the model is working fine.

STEP 11

Deploy the model locally and test from different web browsers locally.



STEP 12

Deploy the model on a Cloud platform and test from different web browsers as well as different devices like Mobile phone, Tab, Laptop etc.

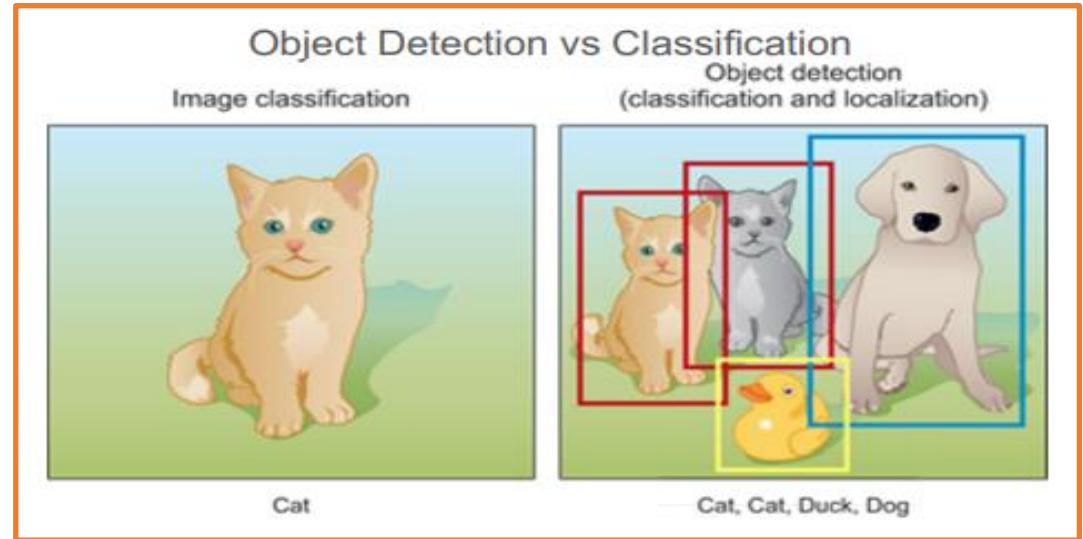
# Summary of Research Papers:

## **Reference Papers:**

- ❖ YOLO v2 ([\[1506.02640\] You Only Look Once: Unified, Real-Time Object Detection \(arxiv.org\)](#)) (extends on the work greatly) (Redmond et al 2016)
  - Class probability distribution per bounding box, not per grid .
  - High resolution classifier (finetune on high resolution) .
- ❖ YOLO v3 ([\[1804.02767\] YOLOv3: An Incremental Improvement \(arxiv.org\)](#))
  - “Incremental Improvement” .
  - Uses independent logistic classifiers for class.
  - Allows for more specificity in classes.
- ❖ Gaussian YOLOv3: An Accurate and Fast Object Detector Using Localization Uncertainty for Autonomous Driving ([\[1904.04620\] Gaussian YOLOv3: An Accurate and Fast Object Detector Using Localization Uncertainty for Autonomous Driving \(arxiv.org\)](#))

# Summary of Research Papers:

- ❖ Object detection is the problem of both locating AND classifying objects
- ❖ Goal of YOLO algorithm is to do object detection both fast AND with high accuracy



## Key Insights

### Previous Approaches

- ❖ A separate model for generating bounding boxes and for classification (more complicated model pipeline)
- ❖ Need to run classification many times (expensive computation)
- ❖ Looks at limited part of the image (lacks contextual information for detection)

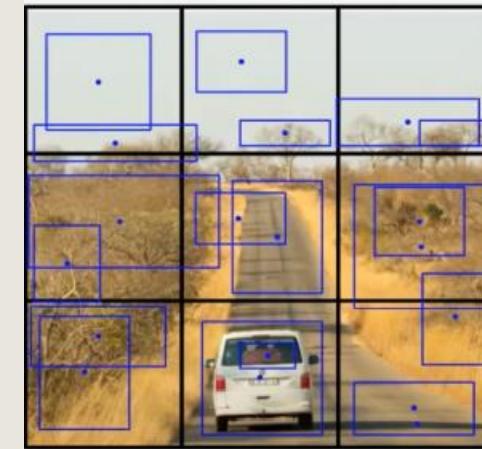
### YOLO Algorithm

- ❖ A single neural network for localization and for classification (less complicated pipeline)
- ❖ Need to inference only once (efficient computation)
- ❖ Looks at the entire image each time leading to less false positives (has contextual information for detection)

# Summary of Research Papers:

## YOLO Overview:

- ❖ First, image is split into a  $S \times S$  grid
- ❖ For each grid square, generate  $B$  bounding boxes
- ❖ For each bounding box, there are 5 predictions:  $x, y, w, h, \text{confidence}$



$$S = 3, B = 2$$

For each grid block, we have a vector like this.  
For this example  $B$  is 2 and  $C$  is 2

## YOLO Training:

- ❖ YOLO is a regression algorithm. What is X? What is Y?
- ❖ X is simple, just an image width (in pixels) \* height (in pixels) \* RGB values
- ❖ Y is a tensor of size  $S \times S \times (B \times 5 + C)$
- ❖  $B \times 5 + C$  term represents the predictions + class predicted distribution for a grid block

p_1
b_x_1
b_y_1
b_h_1
b_w_1
p_2
b_x_2
b_y_2
b_h_2
b_w_2
c_1
c_2

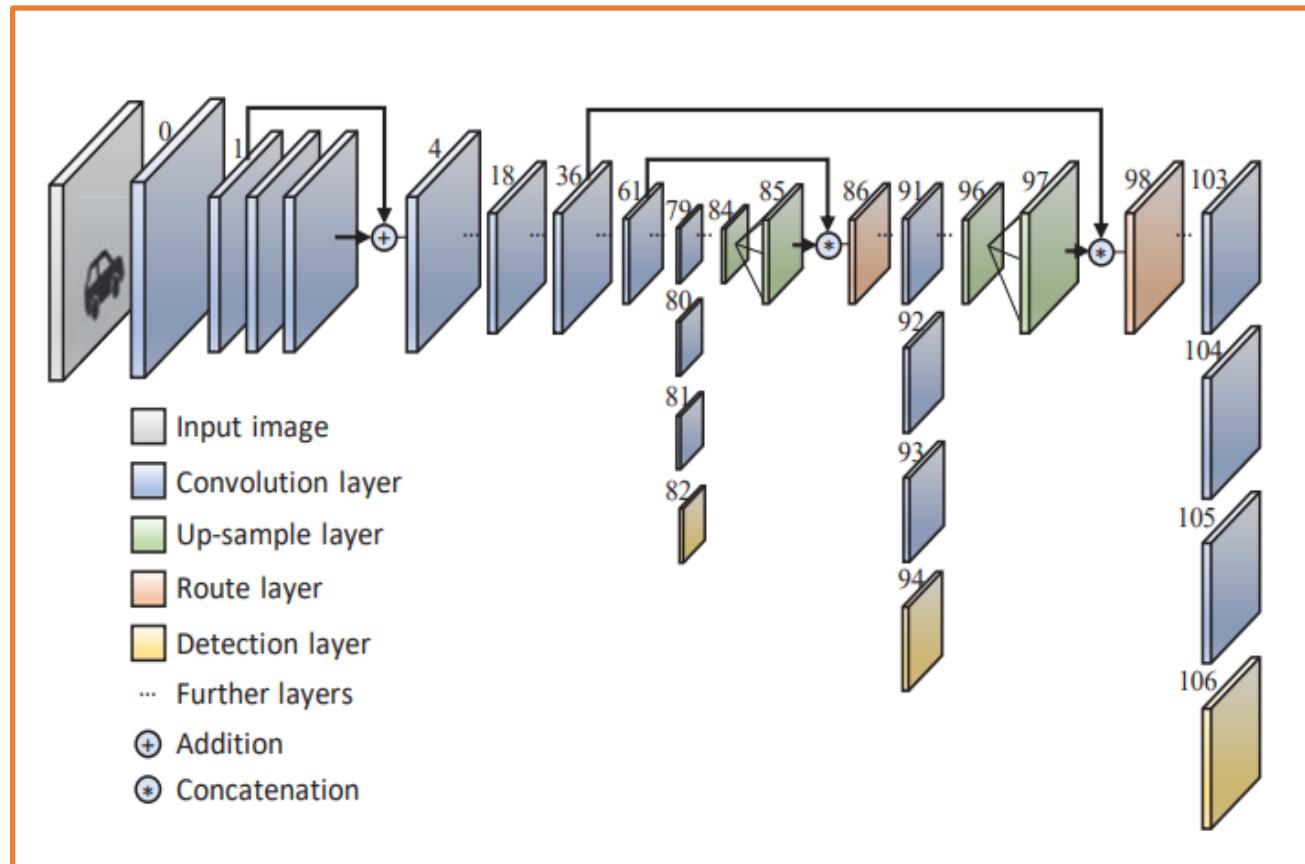


1
b_x_1
b_y_1
b_h_1
b_w_1
0
?
?
?
c_1 = 1
c_2 = 0

# Summary of Research Papers:

## YOLO Architecture:

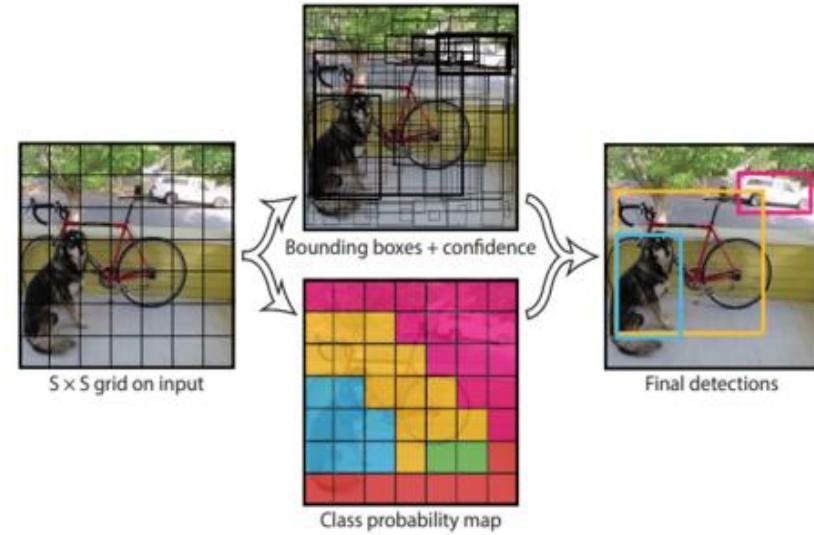
- The first detection is made by the 82nd layer. For the first 81 layers, the image is down sampled by the network such that the 81st layer has a stride of 32. If we have an image of  $416 \times 416$ , the resultant feature map would be of size  $13 \times 13$ .
- The second detection is made by the 94th layer, yielding a detection feature map of  $26 \times 26 \times 255$ .
- The third or final detection is made by 106th layer, yielding feature map of size  $52 \times 52 \times 255$ .



# Summary of Research Papers:

## YOLO Prediction:

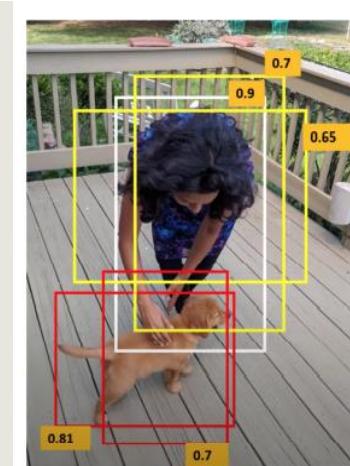
- ❖ We then use the output to make final detections.
- ❖ Use a threshold to filter out bounding boxes with low  $P(\text{Object})$ .
- ❖ In order to know the class for the bounding box compute score take argmax over the distribution  $\Pr(\text{Class}|\text{Object})$  for the grid the bounding box's center is in.



$$\Pr(\text{Class}_i|\text{Object}) * \Pr(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}} = \Pr(\text{Class}_i) * \text{IOU}_{\text{pred}}^{\text{truth}}$$

## Non-Maximal Suppression:

- ❖ Most of the time objects fall in one grid, however it is still possible to get redundant boxes (rare case as object must be close to multiple grid cells for this to happen).
- ❖ Discard bounding box with high overlap (keeping the bounding box with highest confidence).
- ❖ Adds 2-3% on final mAP score.



# Summary of Research Papers:

## YOLO Objective Function

For YOLO, using Sum of squared error, we need to minimize the following loss.

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left[ (x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left[ (\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left( C_i - \hat{C}_i \right)^2$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{noobj}} \left( C_i - \hat{C}_i \right)^2$$

$$+ \sum_{i=0}^{S^2} \mathbb{1}_i^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2$$

**Coordinate Loss:** Minimize the difference between x, y, w, h pred and x, y, w, h ground truth. ONLY IF object exists in grid box and if bounding box is resp for pred.

**Confidence Loss:** Loss based on confidence ONLY IF there is object.

**No Object Loss:** based on confidence if there is no object.

**Class Loss:** minimize loss between true class of object in grid box.

# Architecture:

- **Algorithm:** YOLOv3
  - ✓ YOLO is a fully convolutional network and its eventual output is generated by applying a  $1 \times 1$  kernel on a feature map.
  - ✓ In YOLO v3, the detection is done by applying  $1 \times 1$  detection kernels on feature maps of three different sizes at three different places in the network.
  - ✓ YOLOv3 (You Only Look Once, Version 3) is a real-time object detection algorithm that identifies specific objects in videos, live feeds, or images. YOLO uses features learned by a deep convolutional neural network to detect an object. Versions 1-3 of YOLO were created by Joseph Redmon and Ali Farhadi.
  - ✓ YOLO v3 makes prediction at three scales, which are precisely given by downsampling the dimensions of the input image by 32, 16 and 8 respectively.
- **Framework:** Darknet
  - ✓ Darknet is an open source custom neural network framework written in C and CUDA. It is fast, easy to install, and supports both CPU and GPU computations. The ‘darknet’ framework features ‘You Only Look Once (YOLO)’, a state-of-the-art, real-time object detection system.
  - ✓ Darkflow: It is a nickname of an implementation of YOLO on TensorFlow.
  - ✓ Alexey Bochkovskiy, aka AlexeyAB, created a fork on GitHub and wrote an extensive guide to customizing YOLO’s network architecture, added new features, and has answered zillions of questions.

# Dataset (Source)

The screenshot shows the Berkeley DeepDrive portal. On the left, there's a sidebar with 'Profile', 'Download' (selected), 'Submission', and 'Logout'. The main area has a 'License' section with copyright information and terms of use. Below it is a detailed 'BDD100K' dataset navigation menu with tabs like 'Videos', '100K Images', 'Labels', 'Segmentation', 'MOT 2020 Labels', etc. The 'MOT 2020 Labels' tab is highlighted.

The screenshot shows a Google Colab notebook titled 'ObjectDetection\_DarknetYOLOv3\_ColabPro.ipynb'. It features a 'Files' sidebar with a tree view of files in 'MyDrive/Project3\_Files/bdd100k'. A large text block describes the project: 'Project-03 Image Tagging and Road Object Detection by Kalyana Varma Polapragada, Niranjan Mohapatra, STP Bharadwaja'. Below this, a section titled 'Step-01: Get the Source Files.' provides instructions for getting source files from Google Drive. Another section, 'Step-02: Keep the System Upto Date.', contains a code snippet for upgrading the system:

```
[1] 1 """
2 Google Colab or Google Colab Pro basically provide the Ubuntu based system,
3 which means we have a Ubuntu Linux system to work on.
4 """
5
6 # Upgrade and Update the system.
7 !apt update
8 print("-----")
9 !apt upgrade -y
```

```
[ ] 1 # Extract the image and annotation 'JSON' files from the respective '.zip' files.
2 # Necessary sub-directories are created and files are extracted into respective sub-directories.
3 # Note: Make sure there is enough free space available in Google Drive.
4
5 # !unzip -xvf "/content/gdrive/MyDrive/Project3_Files/Training_Data/bdd100k_box_track_20_labels_trainval.zip" -d "/content/gdrive/MyDrive/Project3_Files/Training_Data"
6 !unzip "/content/gdrive/MyDrive/Project3_Files/Training_Data/bdd100k_box_track_20_labels_trainval.zip" -d "/content/gdrive/MyDrive/Project3_Files/Training_Data"
7 !unzip "/content/gdrive/MyDrive/Project3_Files/Training_Data/bdd100k_seg_track_20_images.zip" -d "/content/gdrive/MyDrive/Project3_Files/Training_Data"
```

# Dataset (Info)

## Labels:

- Person
- N-Wheeler
- Transport
- 2-Wheeler

## Tools:

- LabelImg
- 01\_Video\_to\_Frames.py
- 02\_Frames\_to\_Video.py
- 03\_Json\_to\_Yolo.py
- 04\_ImgCompare.py
- 05\_AddAnnotation.py

## Training Images:

Total image files from 'train' directory	:	30,327
Corresponding text files from 'train' directory	:	30,327
Total image files from 'val' directory	:	6,329
Corresponding text files from 'val' directory	:	6,329
Total image files in the 'bdd100k' directory	:	36,656

# Dataset (Tools)

The image shows five windows side-by-side, each displaying a different Python script. From left to right:

- 01\_Video\_to\_Frames.py**: A script that iterates through a list of video files, creates a folder for each, and extracts frames into them. It handles frame file extensions and frame limits.
- 02\_Frames\_to\_Video.py**: A script that takes a path and converts multiple frame files into a single video file.
- 03\_Json\_to\_Yolo.py**: Converts JSON annotations from a specific format into YOLO3 compatible XML files.
- 04\_ImgCompare.py**: Compares two sets of images (FrameFileListA and FrameFileListB) using OpenCV's imread function.
- 05\_AddAnnotation.py**: Adds additional annotation files to a main annotation list. It handles file paths, file extensions, and object identification values.

```
01_Video_to_Frames.py
02_Frames_to_Video.py
03_Json_to_Yolo.py
04_ImgCompare.py
05_AddAnnotation.py
```

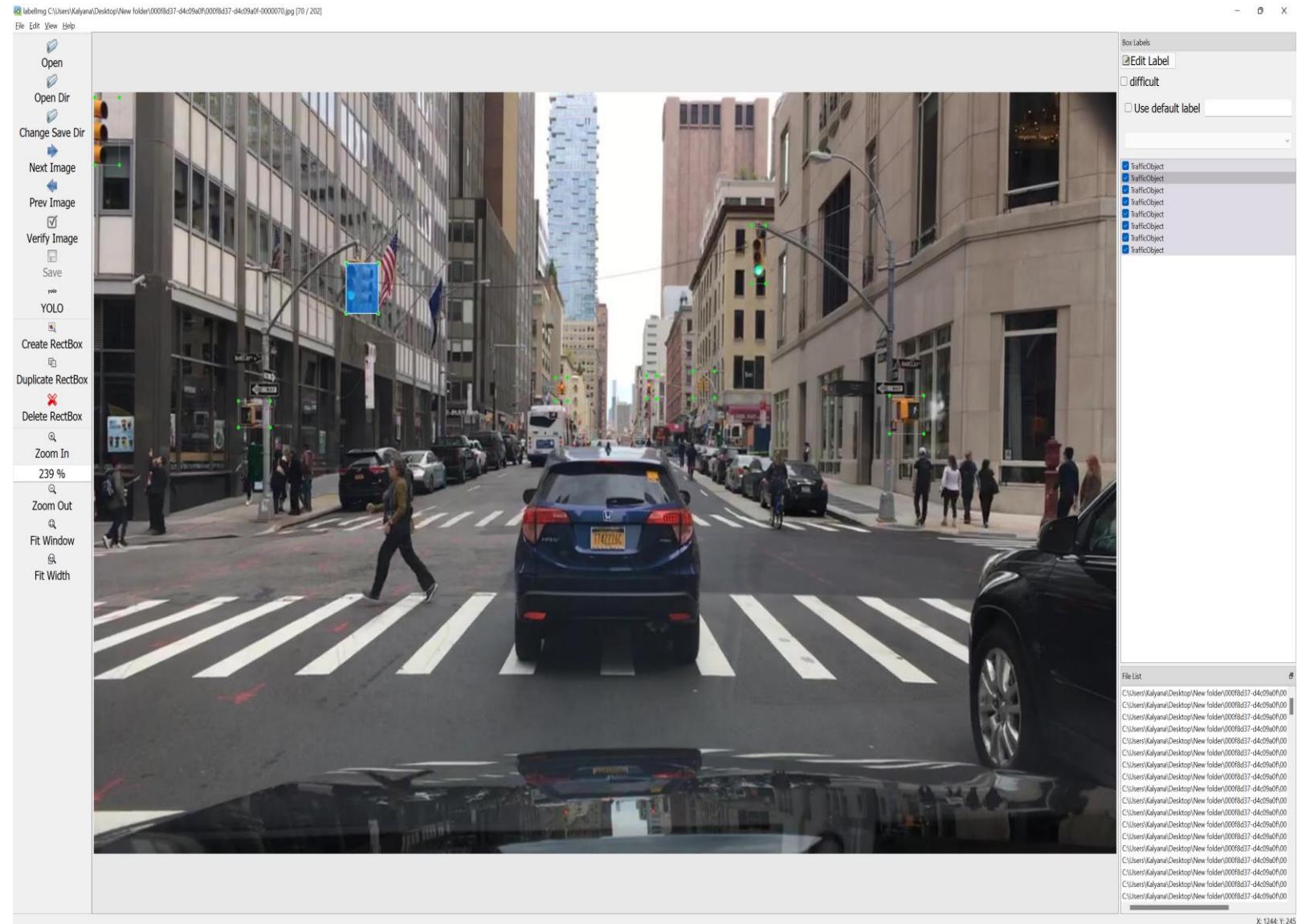
Each window displays the code for its respective script, with syntax highlighting and line numbers. The code is written in Python and performs various tasks related to video processing, file conversion, and annotation management.

# Dataset (Annotation)

```
Command Prompt
D:\StudyKit>labelImg
failed to create process.

D:\StudyKit>pip install labelImg
Collecting labelImg
  Using cached labelImg-1.8.6.tar.gz (247 kB)
    Preparing metadata (setup.py) ... done
Collecting pyqt5
  Downloading PyQt5-5.15.6-cp36-abi3-win_amd64.whl (6.7 MB)
    6.7/6.7 MB 1.7 MB/s eta 0:00:00
Collecting lxml
  Downloading lxml-4.8.0-cp38-cp38-win_amd64.whl (3.6 MB)
    3.6/3.6 MB 3.4 MB/s eta 0:00:00
Collecting PyQt5-sip<13,>=12.8
  Downloading PyQt5_sip-12.9.1-cp38-cp38-win_amd64.whl (77 kB)
    77.5/77.5 KB 4.2 MB/s eta 0:00:00
Collecting PyQt5-Qt5>=5.15.2
  Downloading PyQt5_Qt5-5.15.2-py3-none-win_amd64.whl (50.1 MB)
    50.1/50.1 MB 3.7 MB/s eta 0:00:00
Building wheels for collected packages: labelImg
  Building wheel for labelImg (setup.py) ... done
  Created wheel for labelImg: filename=labelImg-1.8.6-py2.py3-none-any.whl size=261
  533 sha256=9f0297275a7e018c5cc46a9f14f12291c938b2ac45caaf4c0be5144ecad29ef9
  Stored in directory: c:\users\kalyana\appdata\local\pip\cache\wheels\c3\9e\49\836
  8f5bc5347b5e54aef95b7b03ec56af7e23ea4c16c82109c
Successfully built labelImg
Installing collected packages: PyQt5-Qt5, PyQt5-sip, lxml, pyqt5, labelImg
Successfully installed PyQt5-Qt5-5.15.2 PyQt5-sip-12.9.1 labelImg-1.8.6 lxml-4.8.0
pyqt5-5.15.6

D:\StudyKit>labelImg
```



# Model Training:

The screenshot shows a Google Colab Pro notebook titled "ObjectDetection\_DarknetYOLOv3\_ColabPro.ipynb". The left sidebar displays a file tree for "Project3\_Files" containing "Testing\_Data" and "Training\_Data" folders. The "Testing\_Data" folder is highlighted with a red border and contains numerous image files like "0000780.jpg", "000606\_r.jpg", and "Train\_0000f77c-62c2a288-0000001.jpg". The "Training\_Data" folder contains "ToDelete" and "bdd100k" subfolders, with "bdd100k" further divided into "images" and "labels" subfolders, each containing "seg\_track\_20" and "box\_track\_20" subfolders with "train" and "val" subfolders, and "bdd100k\_box\_track\_20\_labels\_trainval.zip" and "bdd100k\_seg\_track\_20\_images.zip" files. A "darknet53.conv.74" file is also present in the main "Project3\_Files" directory. The right panel features a title section with "Project-03" and "Image Tagging and Road Object Detection" in large blue font, followed by the authors' names: "Kalyana Varma Polapragada, Niranjan Mohapatra, STP Bharadwaja." Below this is a section titled "Step-01: Get the Source Files." with instructions and a note about copying source files. A code cell at the bottom shows system upgrade commands.

## Project-03

# Image Tagging and Road Object Detection

by

**Kalyana Varma Polapragada, Niranjan Mohapatra, STP Bharadwaja.**

### Step-01: Get the Source Files.

Note: To execute this project, we need some source files. Get the following source files.

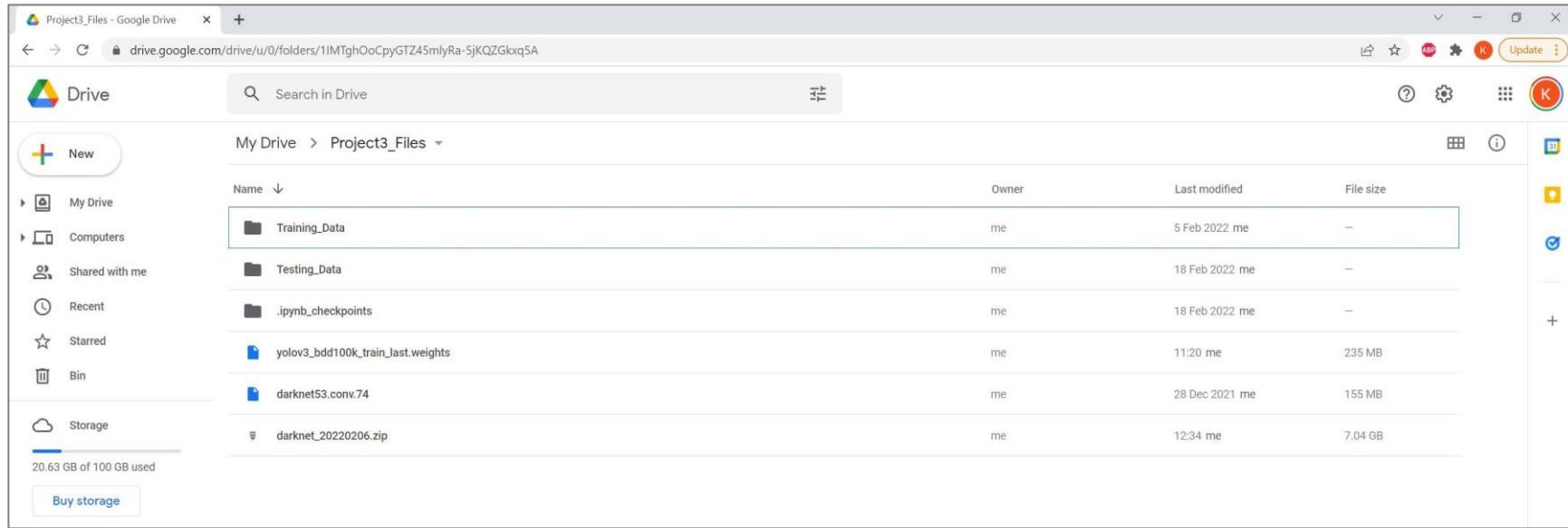
- Make sure the YOLOv3 initial weights file is copied to the Google drive with the following path:
  - Project3\_Files/darknet53.conv.74
  - Please note that 'darknet53.conv.74' is initial YOLOv3 weights file for training custom data and 'yolov3\_bdd100k\_train\_last.weights' is fully trained weights file.
- Also make sure the training data image and label zip files are copied to the Google drive with the following path:
  - Project3\_Files/Training\_Data/bdd100k\_seg\_track\_20\_images.zip
  - Project3\_Files/Training\_Data/bdd100k\_box\_track\_20\_labels\_trainval.zip

In case, not yet copied, then please copy right now.

### Step-02: Keep the System Upto Date.

```
[1] 1 """
2 Google Colab or Google Colab Pro basically provide the Ubuntu based system,
3 which means we have a Ubuntu Linux system to work on.
4 """
5
6 # Upgrade and Update the system.
7 !apt update
8 print("-----")
9 !apt upgrade -y
```

# Model Training:



The screenshot shows a Google Drive interface with the following details:

- Title Bar:** Project3\_Files - Google Drive
- Address Bar:** drive.google.com/drive/u/0/folders/1IMTghOoCpyGTZ45mlyRa-5jKQZGkxq5A
- User Profile:** A red circular icon with a white letter 'K' is visible in the top right corner.
- Left Sidebar:** Includes 'New' button, 'My Drive' folder, 'Computers', 'Shared with me', 'Recent', 'Starred', 'Bin', and 'Storage' section showing 20.63 GB of 100 GB used.
- Search Bar:** Search in Drive
- Table View:** A list of files and folders under 'Project3\_Files'. The columns are 'Name', 'Owner', 'Last modified', and 'File size'.
- Items Listed:**
  - Training\_Data (Folder)
  - Testing\_Data (Folder)
  - .ipynb\_checkpoints (Folder)
  - yolov3\_bdd100k\_train\_last.weights (File, 235 MB)
  - darknet53.conv.74 (File, 155 MB)
  - darknet\_20220206.zip (File, 7.04 GB)

# Model Performance

```
Last accuracy mAP@0.50 = 17.29 %, best = 17.29 %
8000: 3.566150, 4.041121 avg loss, 0.000000 rate, 3.057349 seconds, 256000 images, 0.067682 hours left
Resizing to initial size: 416 x 416  try to allocate additional workspace_size = 52.43 MB
CUDA allocate done!

calculation mAP (mean average precision)...
Detection layer: 82 - type = 28
Detection layer: 94 - type = 28
Detection layer: 106 - type = 28
6332
detections_count = 526758, unique_truth_count = 78226
class_id = 0, name = Person, ap = 13.07%           (TP = 1649, FP = 1949)
class_id = 1, name = N-Wheeler, ap = 51.34%         (TP = 25949, FP = 8465)
class_id = 2, name = Transport, ap = 4.57%          (TP = 4, FP = 19)
class_id = 3, name = 2-Wheeler, ap = 0.33%          (TP = 0, FP = 0)

for conf_thresh = 0.25, precision = 0.73, recall = 0.35, F1-score = 0.47
for conf_thresh = 0.25, TP = 27602, FP = 10433, FN = 50624, average IoU = 50.77 %

IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
mean average precision (mAP@0.50) = 0.173275, or 17.33 %
Total Detection Time: 133 Seconds
```

# Model Performance

## 1. Confusion matrix:

### ***True Positive (TP):***

- The actual label of the given instance is Positive.
- The classifier also predicts it as Positive.

### ***True Negative (TN):***

- The actual label of the given instance is Negative.
- The classifier also predicts it as Negative.

### ***False Positive (FP):***

- The actual label of the given instance is Negative.
- The classifier incorrectly predicts it as Positive.
- Also called as 'Type 1 Error'.

### ***False Negative (FN):***

- The actual label of the given instance is Positive.
- The classifier incorrectly predicts it as Negative.
- Also called as 'Type 2 Error'.

# Model Performance

## 2. Accuracy:

- Accuracy is the ratio of the number of correct predictions to the total number of input samples.
- $\text{Accuracy} = (\text{TP} + \text{TN}) / (\text{TP} + \text{TN} + \text{FP} + \text{FN})$

## 3. Precision:

- Precision tells us how many of the correctly predicted cases actually turned out to be positive.
- Precision is a useful metric in cases where False Positive is a higher concern than False Negatives.
- Precision is important in music or video recommendation systems, e-commerce websites, etc. Wrong results could lead to customer churn and be harmful to the business.
- $\text{Precision} = \text{TP} / (\text{TP} + \text{FP})$

# Model Performance

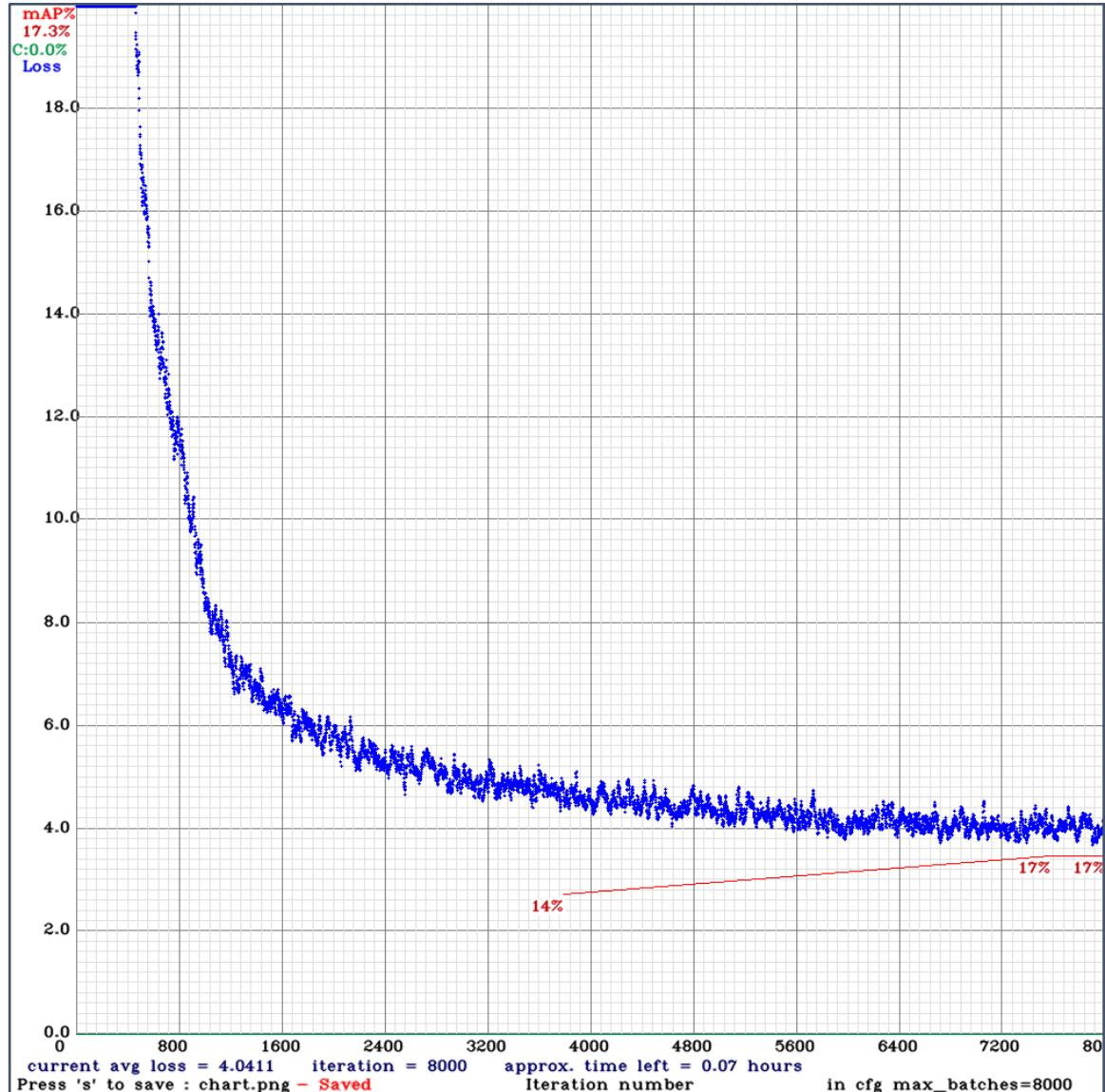
## 4. Recall:

- Recall tells us how many of the actual positive cases we were able to predict correctly with our model.
- Recall is a useful metric in cases where False Negative trumps False Positive.
- Recall is important in medical cases where it doesn't matter whether we raise a false alarm but the actual positive cases should not go undetected!
- $\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$

## 5. F1-Score:

- In practice, when we try to increase the Precision of our model, the Recall goes down and vice-versa. The F1-score captures both the trends in a single value.
- F1-score is a harmonic mean of Precision and Recall, and so it gives a combined idea about these two metrics. It is maximum when Precision is equal to Recall.
- There is a catch, the interpretability of the F1-score is poor. This means that we don't know what our classifier is maximizing - Precision or Recall? So, we use it in combination with other metrics which gives us a complete picture of the result.
- $\text{F1-score} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$

# Model Performance



## Mean Average Precision (mAP):

Mean average precision is an object detection metric that multiple data sets use to compare different object detectors. 'mAP' calculated by the area under a precision vs recall curve. The maximum Mean Average Precision indicates the prediction accuracy is more.

## Intersection over Union (IoU):

In 'IoU', it takes the actual and predicted bounding box value and calculates the IoU of two boxes by using the formula:

$$\text{IoU} = \frac{\text{Area of Intersection}}{\text{Area of Union}}$$

If the value of IoU is more than or equal to the threshold value then it is a good prediction. The threshold value is just an assuming value. We can also take greater threshold value to increase the accuracy or for better prediction of the object.

## Loss:

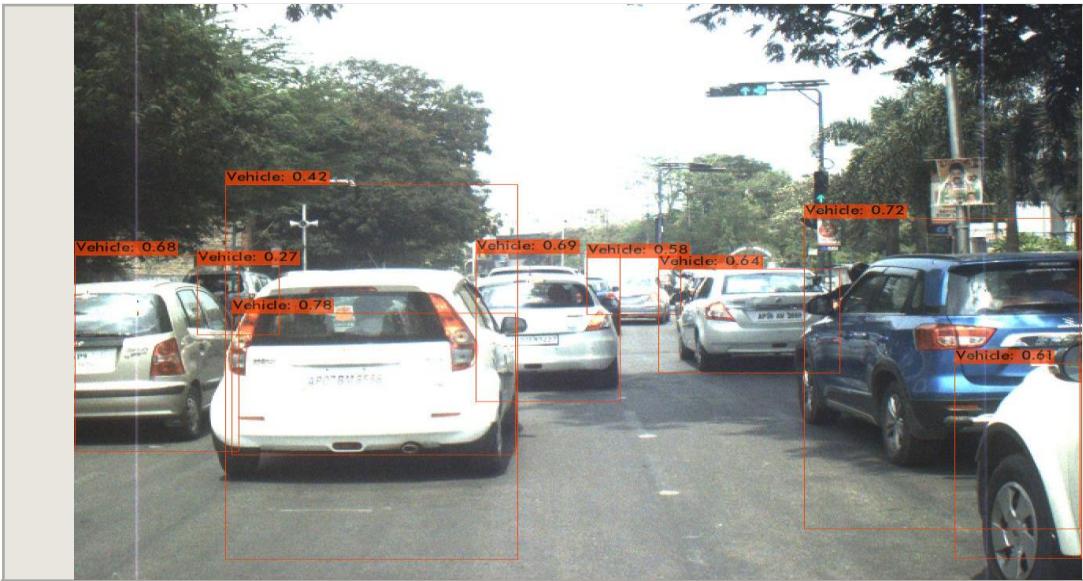
We increase the loss from bounding box coordinate predictions and decrease the loss from confidence predictions for boxes that don't contain objects.

# Outcomes: Testing with ‘bdd100k’ Data

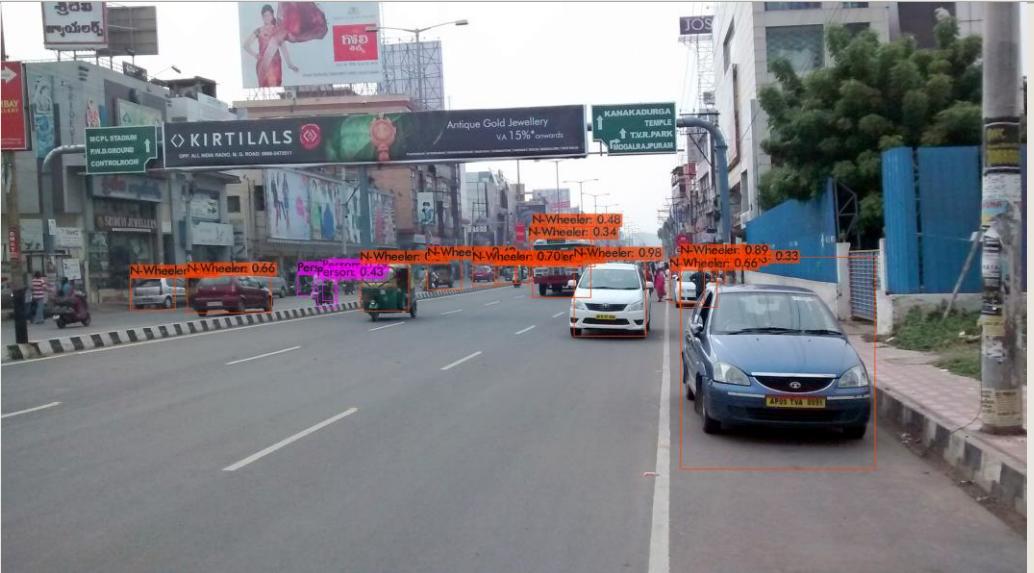
- `classes.names`
- `yolov3_bdd100k_test.cfg`
- `yolov3_bdd100k_train_last.weights`



# Outcomes: Testing with 'IIIT' Data



# Outcomes: Testing with ‘Live’ Data



# Observations and Learnings:



**Following table summarizes our dataset:**

Label	#Instance of the Class Objects	Percentage
Person	60,923	16.27
N-Wheeler	2,99,914	80.08
Transport	8,429	2.25
2-Wheeler	5,255	1.40

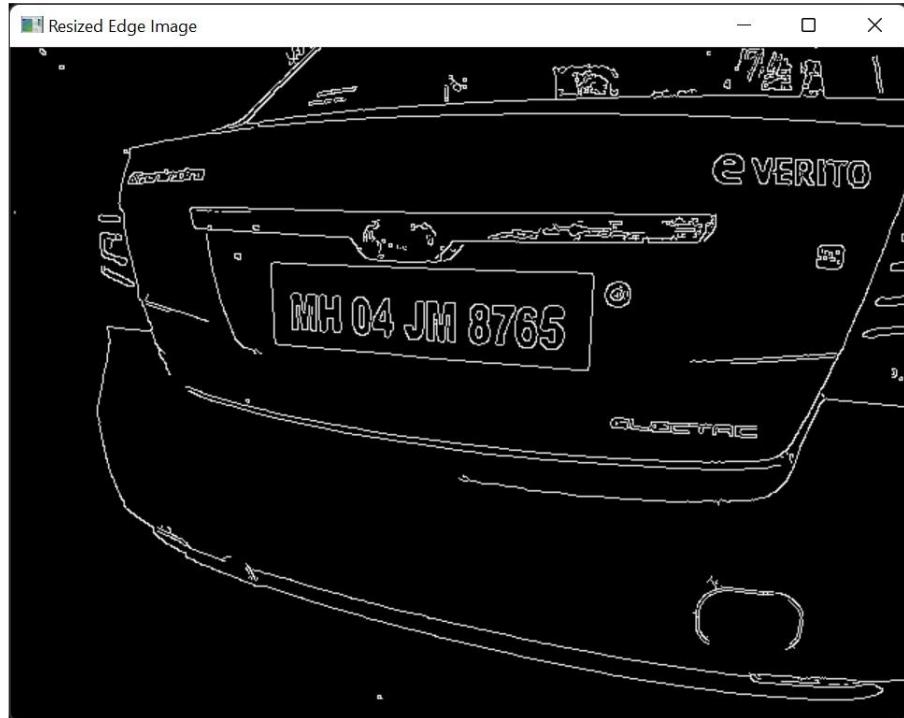
**Imbalanced dataset:** From the statistics, the dataset is significantly imbalanced. The top class object (N-Wheeler) account for 80% of all the objects and the top 2 class objects (N-Wheeler, Person) make up 96% of the total objects. The most frequent object class is 'N-Wheeler' that comprises 80% of the dataset.

**Multiple instances in the same image:** In each image, there is more than one instance of the same class objects. For example, the 'N-Wheeler' class has 3–4 instances per image. In some images, this number is even bigger. This is understandable given that the images are street images where cars are usually nearby. Finding all the instances of different sizes in an image is an added challenge for the model.

**Fixing the class imbalance problem:**

- The first method is called 'Under-sampling' which means to sample less from the majority classes.
- The second method is called 'Oversampling' which means to sample more from the minority classes.
- The result of the 'Under-sampling' or 'Oversampling' is resampled dataset.
- A completely balanced dataset is not possible due to the nature of the data. For example, when sampling for 'Transport' or '2-Wheeler', we are forced to take 'N-Wheeler' instances too.

# Unique Feature of the Project (Tried):



# Unique Feature of the Project (Tried):



A photograph of the rear of a white Mahindra e-Verito electric vehicle. The license plate area is highlighted with a green bounding box, containing the text "MH 04 JM 8765".

Command Prompt

```
D:\StudyKit\2021-11-13 AI, ML PG Course at IIIT, Hyd\Project#03\SourceCode\Img2Text>python 06_NumPlateInfo.py "D:\StudyKit\2021-11-13 AI, ML PG Course at IIIT, Hyd\Project#03\SourceCode\Img2Text\Pics" "[Cat01] HR_Pri_IC_001.jpg" "C:\Program Files\Tesseract-OCR\tesseract.exe" "1"

Extracted Vehicle Number: MH 04 JM 8765 |

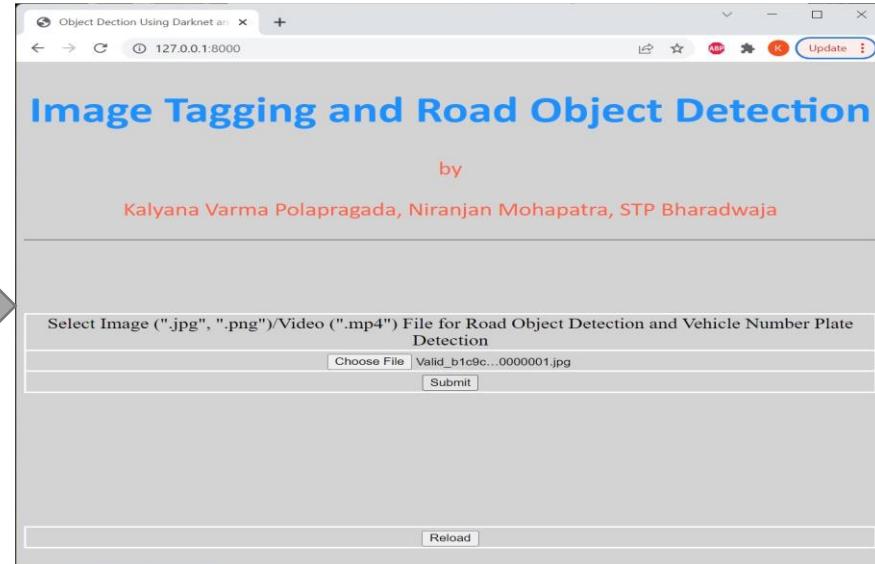
Vehicle Info:
-----
Registration Number : MH04JM8765
State/UT of Registration: Maharashtra
Number Plate BG Color : Green
Number Plate FG Color : Yellow
Vehicle Category : Commercial EV

D:\StudyKit\2021-11-13 AI, ML PG Course at IIIT, Hyd\Project#03\SourceCode\Img2Text>
```

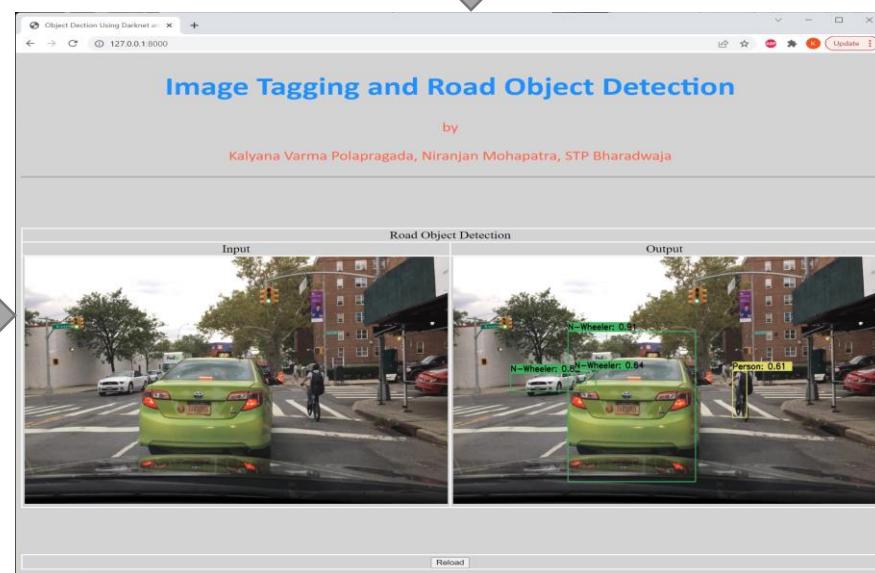
The Command Prompt window shows the output of the Python script. It first displays the extracted vehicle number "MH 04 JM 8765". Below that, it provides detailed vehicle information, including the registration number "MH04JM8765", state "Maharashtra", number plate colors (BG: Green, FG: Yellow), and category "Commercial EV".

# Local Deployment:

```
Command Prompt - unicorn BackEnd:App --reload
D:\StudyKit\2021-11-13 AI, ML PG Course at IIIT, Hyd\Project#03\06 Deployment>unicorn BackEnd:App --reload
+[32mINFO+[0m: Will watch for changes in these directories: ['D:\\StudyKit\\2021-11-13 AI, ML PG Course
at IIIT, Hyd\\Project#03\\06 Deployment']
+[32mINFO+[0m: Unicorn running on +[1mhttp://127.0.0.1:8000-[0m (Press CTRL+C to quit)
+[32mINFO+[0m: Started reloader process [+36m-[1m22156-[0m] using +[36m-[1mstatreload-[0m
+[33mWARNING+[0m: The --reload flag should not be used in production on Windows.
+[32mINFO+[0m: Started server process [+36m2372-[0m]
+[32mINFO+[0m: Waiting for application startup.
+[32mINFO+[0m: Application startup complete.
```



```
Command Prompt - unicorn BackEnd:App --reload
D:\StudyKit\2021-11-13 AI, ML PG Course at IIIT, Hyd\Project#03\06 Deployment>unicorn BackEnd:App --reload
+[32mINFO+[0m: Will watch for changes in these directories: ['D:\\StudyKit\\2021-11-13 AI, ML PG Course
at IIIT, Hyd\\Project#03\\06 Deployment']
+[32mINFO+[0m: Unicorn running on +[1mhttp://127.0.0.1:8000-[0m (Press CTRL+C to quit)
+[32mINFO+[0m: Started reloader process [+36m-[1m22156-[0m] using +[36m-[1mstatreload-[0m
+[33mWARNING+[0m: The --reload flag should not be used in production on Windows.
+[32mINFO+[0m: Started server process [+36m2372-[0m]
+[32mINFO+[0m: Waiting for application startup.
+[32mINFO+[0m: Application startup complete.
+[32mINFO+[0m: 127.0.0.1:57689 - "+[1mGET / HTTP/1.1-[0m" +[32m200 OK-[0m
+[32mINFO+[0m: 127.0.0.1:57704 - "+[1mPOST / HTTP/1.1-[0m" +[32m200 OK-[0m
+[32mINFO+[0m: 127.0.0.1:57704 - "+[1mGET /Dynamic/InputFile.jpg HTTP/1.1-[0m" +[32m200 OK-[0m
+[32mINFO+[0m: 127.0.0.1:57705 - "+[1mGET /Dynamic/OutputFile.jpg HTTP/1.1-[0m" +[32m200 OK-[0m
```



# Cloud Deployment:

The image shows two side-by-side browser windows illustrating the process of deploying a GitHub repository to Heroku.

**Left Window (GitHub Repository):**

- URL:** [github.com/KalyanaVarma/AIML\\_Project03](https://github.com/KalyanaVarma/AIML_Project03)
- Repository Name:** KalyanaVarma / AIML\_Project03 (Public)
- Code Tab:** Selected. Shows the main branch with 1 branch and 0 tags.
- Commits:** A recent commit by KalyanaVarma titled "Update requirements.txt" is shown, made 17 days ago.
- Languages:** Python (66.8%) and HTML (33.2%).

**Right Window (Heroku Deployment Interface):**

- URL:** [dashboard.heroku.com/apps/aiml-project03/deploy/github](https://dashboard.heroku.com/apps/aiml-project03/deploy/github)
- Automatic Deploys:** Enabled for the main branch. A message indicates that the main branch can now be changed to "main".
- Manual Deploy:** A section for deploying a GitHub branch.
- Pipeline:** A build pipeline for the main branch is shown, starting with "Installing requirements with pip" and ending with "Released v53" at <https://aiml-project03.herokuapp.com/>.
- Deploy Phase:** Set to "Deploy to Heroku".

# Cloud Deployment:

The image shows two side-by-side browser windows displaying application logs for a Heroku application named 'aiml-project03'.

**Left Browser Tab:** Shows the Heroku dashboard for 'aiml-project03'. The 'Application Logs' section displays a log entry from February 2022 at 02:24:16. The log message indicates the application has started successfully, with the command 'unicorn BackEnd:App --host=0.0.0.0 --port=\${PORT:-5000}' and the application running on port 5000.

```
2022-02-24T16:34:02.047868+00:00 heroku[web.1]: state changed from down to starting
2022-02-24T16:34:11.000649+00:00 heroku[web.1]: Starting process with command `unicorn BackEnd:App --host=0.0.0.0 --port=${PORT:-5000}`
2022-02-24T16:34:12.193571+00:00 app[web.1]: INFO:    Unicorn running on http://0.0.0.0:12402 (Press CTRL+C to quit)
2022-02-24T16:34:12.193645+00:00 app[web.1]: INFO:    Started parent process [4]
2022-02-24T16:34:13.245232+00:00 app[web.1]: INFO:    Started server process [11]
2022-02-24T16:34:13.245297+00:00 app[web.1]: INFO:    Waiting for application startup.
2022-02-24T16:34:13.245479+00:00 app[web.1]: INFO:    Application startup complete.
2022-02-24T16:34:13.256606+00:00 app[web.1]: INFO:    Started server process [10]
2022-02-24T16:34:13.256724+00:00 app[web.1]: INFO:    Waiting for application startup.
2022-02-24T16:34:13.257059+00:00 app[web.1]: INFO:    Application startup complete.
2022-02-24T16:33:22.000000+00:00 app[api]: Build started by user nirajan.cs2005@gmail.com
2022-02-24T16:34:13.607783+00:00 heroku[web.1]: State changed from starting to up
2022-02-24T16:34:01.636699+00:00 app[api]: Deploy b1e72b76 by user nirajan.cs2005@gmail.com
2022-02-24T16:34:01.636699+00:00 app[api]: Release v53 created by user nirajan.cs2005@gmail.com
2022-02-24T16:34:22.000000+00:00 app[api]: Build succeeded
```

**Right Browser Tab:** Shows the Heroku application logs page for 'aiml-project03'. The log output is identical to the one on the left, showing the application starting and deploying successfully.

```
fuel="103.125.162.231" dyno=web.1 connect=0ms service=2ms status=404 bytes=173 protocol=http
2022-02-07T04:46:11.151537+00:00 heroku/router]: at->info method=POST path="/" host=aiml-project03.herokuapp.com request_id=5d4e3cd7-dicb-4cca-8832-8591dd9b8039
fuel="103.125.162.231" dyno=web.1 connect=0ms service=5190ms status=500 bytes=193 protocol=http
2022-02-07T04:46:11.150424+00:00 app[web.1]: ERROR:   Exception in ASGI application
2022-02-07T04:46:11.150425+00:00 app[web.1]: Traceback (most recent call last):
2022-02-07T04:46:11.150426+00:00 app[web.1]:   File "/app/heroku/python/lib/python3.8/site-packages/uvicorn/protocols/http/h11_impl.py", line 373, in run_asgi
2022-02-07T04:46:11.150426+00:00 app[web.1]:     result = await app(self.scope, self.receive, self.send)
2022-02-07T04:46:11.150427+00:00 app[web.1]:   File "/app/heroku/python/lib/python3.8/site-packages/uvicorn/middleware/proxy_headers.py", line 75, in __call__
2022-02-07T04:46:11.150427+00:00 app[web.1]:     return await self.app(scope, receive, send)
2022-02-07T04:46:11.150428+00:00 app[web.1]:   File "/app/heroku/python/lib/python3.8/site-packages/fastapi/applications.py", line 212, in __call__
2022-02-07T04:46:11.150428+00:00 app[web.1]:     await super().__call__(scope, receive, send)
2022-02-07T04:46:11.150429+00:00 app[web.1]:   File "/app/heroku/python/lib/python3.8/site-packages/starlette/applications.py", line 112, in __call__
2022-02-07T04:46:11.150429+00:00 app[web.1]:     await self.middleware_stack(scope, receive, send)
2022-02-07T04:46:11.150429+00:00 app[web.1]:   File "/app/heroku/python/lib/python3.8/site-packages/starlette/middleware/errors.py", line 181, in __call__
2022-02-07T04:46:11.150430+00:00 app[web.1]:     raise exc
2022-02-07T04:46:11.150430+00:00 app[web.1]:   File "/app/heroku/python/lib/python3.8/site-packages/starlette/middleware/errors.py", line 159, in __call__
2022-02-07T04:46:11.150430+00:00 app[web.1]:     await self.scope, receive, _send)
2022-02-07T04:46:11.150431+00:00 app[web.1]:   File "/app/heroku/python/lib/python3.8/site-packages/starlette/exceptions.py", line 82, in __call__
2022-02-07T04:46:11.150431+00:00 app[web.1]:     raise exc
2022-02-07T04:46:11.150431+00:00 app[web.1]:   File "/app/heroku/python/lib/python3.8/site-packages/starlette/exceptions.py", line 71, in __call__
2022-02-07T04:46:11.150431+00:00 app[web.1]:     await self.app(scope, receive, sender)
2022-02-07T04:46:11.150431+00:00 app[web.1]:   File "/app/heroku/python/lib/python3.8/site-packages/starlette/routing.py", line 656, in __call__
2022-02-07T04:46:11.150431+00:00 app[web.1]:     await route.handle(scope, receive, send)
2022-02-07T04:46:11.150432+00:00 app[web.1]:   File "/app/heroku/python/lib/python3.8/site-packages/starlette/routing.py", line 259, in handle
2022-02-07T04:46:11.150432+00:00 app[web.1]:     await self.app(scope, receive, send)
2022-02-07T04:46:11.150432+00:00 app[web.1]:   File "/app/heroku/python/lib/python3.8/site-packages/starlette/routing.py", line 61, in app
2022-02-07T04:46:11.150432+00:00 app[web.1]:     response = await func(request)
2022-02-07T04:46:11.150432+00:00 app[web.1]:   File "/app/heroku/python/lib/python3.8/site-packages/fastapi/routing.py", line 226, in app
2022-02-07T04:46:11.150433+00:00 app[web.1]:     raw_response = await run_endpoint_function
2022-02-07T04:46:11.150433+00:00 app[web.1]:   File "/app/heroku/python/lib/python3.8/site-packages/fastapi/routing.py", line 159, in run_endpoint_function
2022-02-07T04:46:11.150433+00:00 app[web.1]:     return await dependant.call(**values)
2022-02-07T04:46:11.150433+00:00 app[web.1]:   File ".\Backend.py", line 338, in root
2022-02-07T04:46:11.150433+00:00 app[web.1]:     OutputFile = DetectRoadObjects(InputFile)
2022-02-07T04:46:11.150433+00:00 app[web.1]:   File ".\Backend.py", line 123, in DetectRoadObjects
2022-02-07T04:46:11.150434+00:00 app[web.1]:     Network = cv2.dnn.readNetFromDarknet(PATH_CONFIG, PATH_WEIGHT)
2022-02-07T04:46:11.150434+00:00 app[web.1]: cv2.error: OpenCV(4.5.5) /io/opencv/modules/dnn/src/darknet/darknet.hpp:93: error: (-213): The function/feature is not implemented
2022-02-07T04:46:11.150435+00:00 app[web.1]: 
```

# Cloud Deployment:

## Image Tagging and Road Object Detection

by  
Kalyana Varma Polapragada, Niranjan Mohapatra, STP Bharadwaja

---

Select Image/Video File for Object Detection

Valid\_b1c9c...0000001.jpg

---

Input File	Output File
	
<input type="button" value="Reload"/>	

## Image Tagging and Road Object Detection

by  
Kalyana Varma Polapragada, Niranjan Mohapatra, STP Bharadwaja

---

Input File



Output File



---

<input type="button" value="Reload"/>
---------------------------------------

# Issues, Troubles and Challenges :



## ***Dataset:***

- Selecting data (images) from the ‘bdd100k’ dataset.
- Manual annotation for specific object classes using ‘LabelImg’.

## ***Training:***

- What is the best environment for training? Google Colab or Amazon SageMaker?
- How to avoid Google Colab disconnecting (Clearing RAM) during the training and during file copy?
- How to copy files between Google Colab folders and local Windows/Linux folders?
- How to manage Google drive's slow response to delete files permanently?
- How to configure and use GPU, CUDA for the training process?
- How many images do you need for object detection?
- How to train the model iteratively?
- How to understand and analyze the training output or log messages?

## ***Unique Feature:***

- Creating edged image and finding contours in the edged image. So much of parameter tuning is required.
- Issues with the ‘pytesseract’ OCR (Optical Character Recognition).

## ***Deployment:***

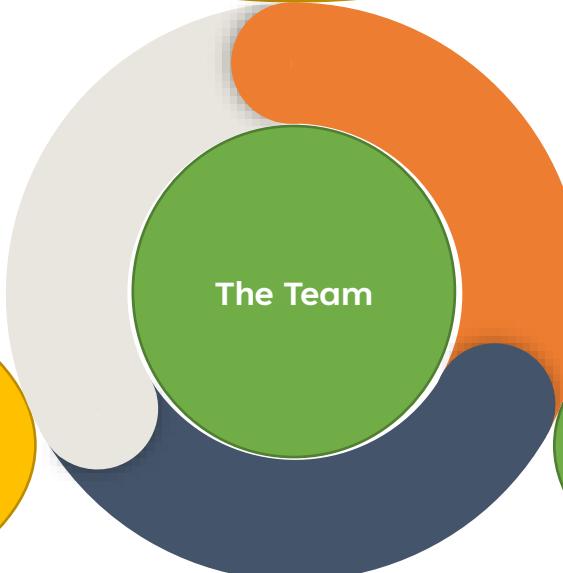
- Identifying the correct video CODEC because, videos generated with any CODEC are unable to play on the frontend HTML.
- GitHub’s strict limitation of Individual file size of 100 MB w.r.t. YOLO weights files.
- Issue of using specific API calls i.e. ‘cv2.dnn.readNetFromDarknet ()’ when using server specific python modules i.e. ‘opencv-python-headless’ on Heroku.



Niranjan  
Mohapatra



Kalyana Varma  
Polapragada



Bharadwaja  
STP

Thanks Thanks

Questions Questions

knowledge

Artificial

method

algorithms

actionable

analysis

processing

analytical

based

learning

language

machine

identify

structured

human

expert

processes

model

systems

interdisciplinary

data

unstructured

automates

AI

speech

insights

intelligence

make

field

apply

extract

especially

computer

recognition

building

science

application

specific

minimal

learn

intervention

domains

broad

natural

Thanks Questions

range

decisions

patterns

idea