PROJECT-03: IMAGE TAGGING AND ROAD OBJECT DETECTION

By Group 4

GitHub Link: 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" .

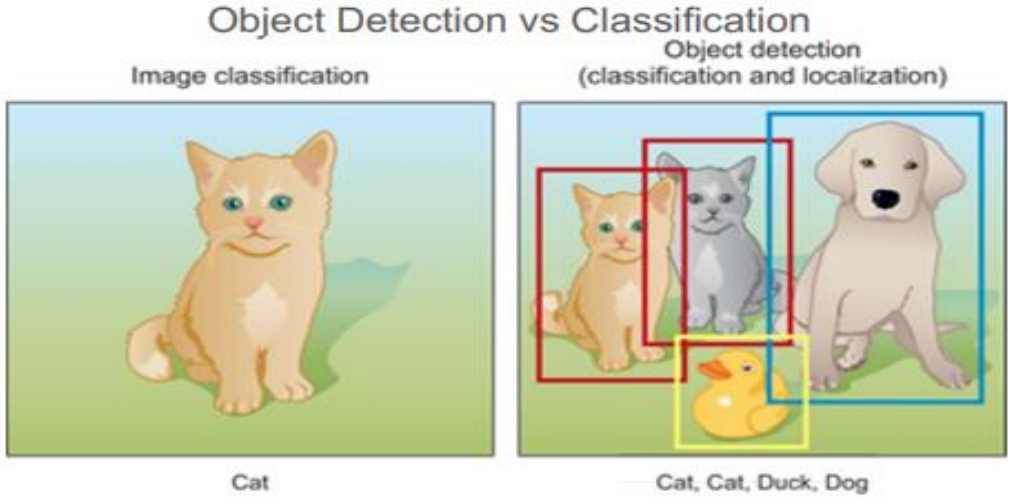
➢ Uses independent logistic classifiers for class.

➢ 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)

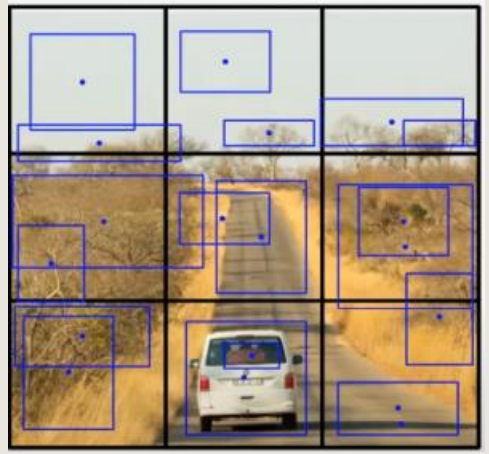
Summary of Research Papers:

YOLO Overview:

- First, image is split into a SxS grid
- For each grid square, generate B bounding boxes
- For each bounding box, there are 5 predictions: x, y, w, h, confidence

YOLO Training:

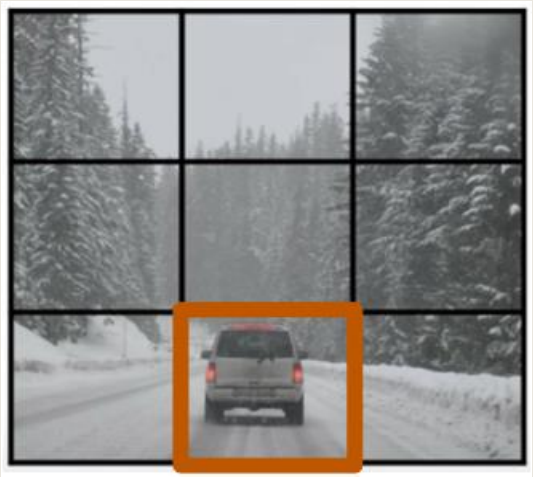
- YOLO is a regression algorithm. What is X? What is Y?
- X is simple, just an image width (in pixels) * height (in pixels) * RGB values
- Y is a tensor of size S * S * (B * 5 + C)
- B*5 + C term represents the predictions + class predicted distribution for a grid block



S = 3, B = 2

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

ρ_1
b_x_1
b_y_1
b_h_1
b_w_1
ρ_2
b_x_2
b_y_2
b_h_2
b_w_2
c_1
c_2

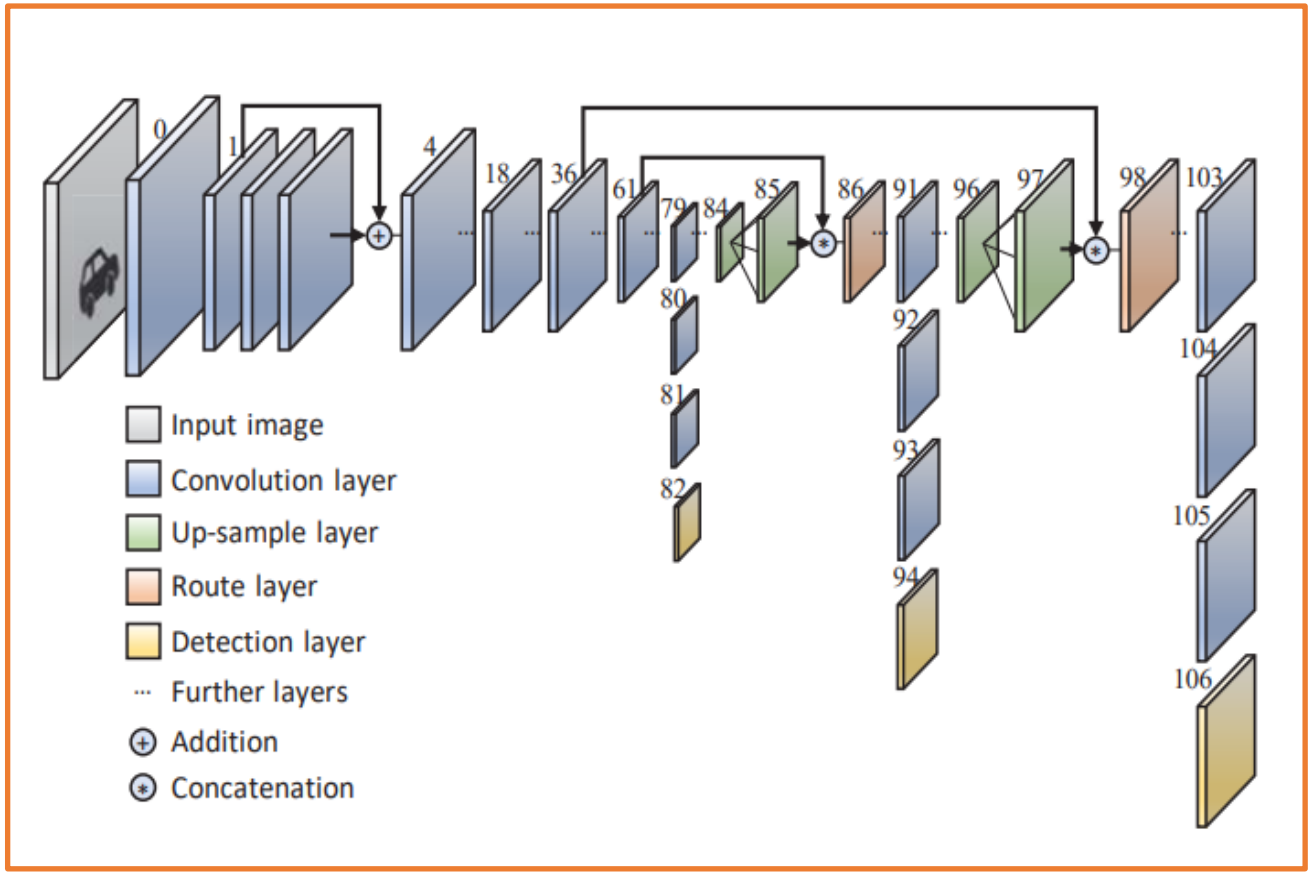


1
b_x_1
b_y_1
b_h_1
b_w_1
0
?
?
?
?
c_1 = 1
c_2 = 0

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 x 416, the resultant feature map would be of size 13 x 13.
- The second detection is made by the 94th layer, yielding a detection feature map of 26 x 26 x 255.
- The third or final detection is made by 106th layer, yielding feature map of size 52 x 52 x 255.



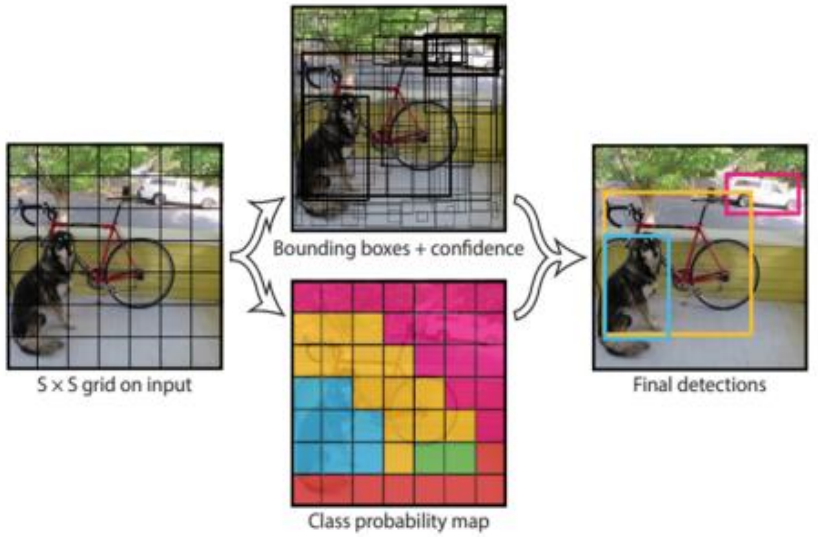
Summary of Research Papers:

YOLO Prediction:

❖ We then use the output to make final detections.

❖ Use a threshold to filter out bounding boxes with low P(Object).

❖ In order to know the class for the bounding box compute score take argmax over the distribution Pr(Class|Object) for the grid the bounding box's center is in.



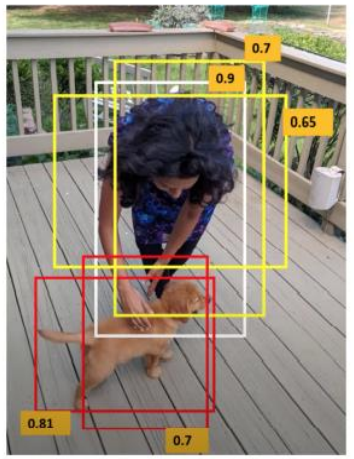
$$\Pr(\text{Class}_i|\text{Object}) * \Pr(\text{Object}) * \text{IOU}^{\text{truth}}_{\text{pred}} = \Pr(\text{Class}_i) * \text{IOU}^{\text{truth}}_{\text{pred}}$$

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:

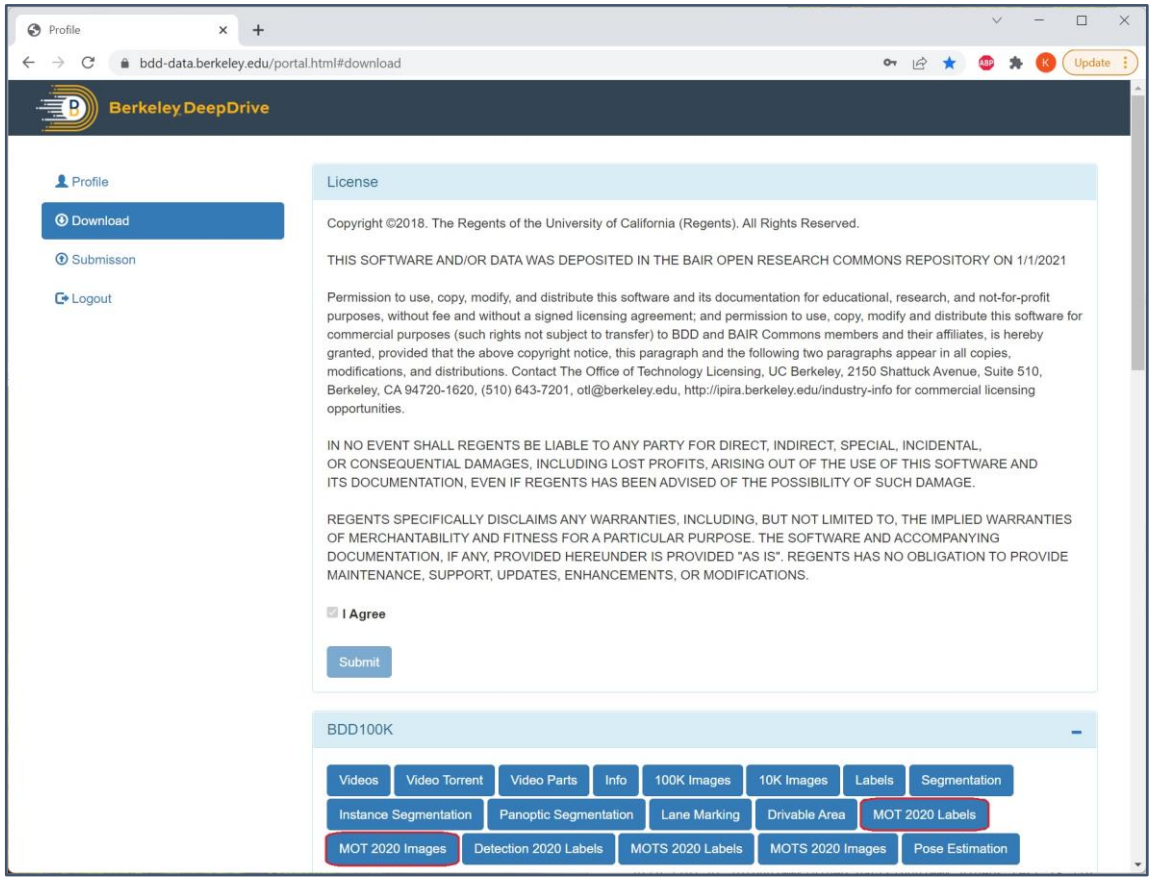
YOLO Objective Function For YOLO, using Sum of squared error, we need to minimize the following loss.	
$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$ $+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$	**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.
$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$	**Confidence Loss**: Loss based on confidence ONLY IF there is object.
$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$	**No Object Loss**: based on confidence if there is no object.
$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2$	**Class Loss**: minimize loss between true class of object in grid box.

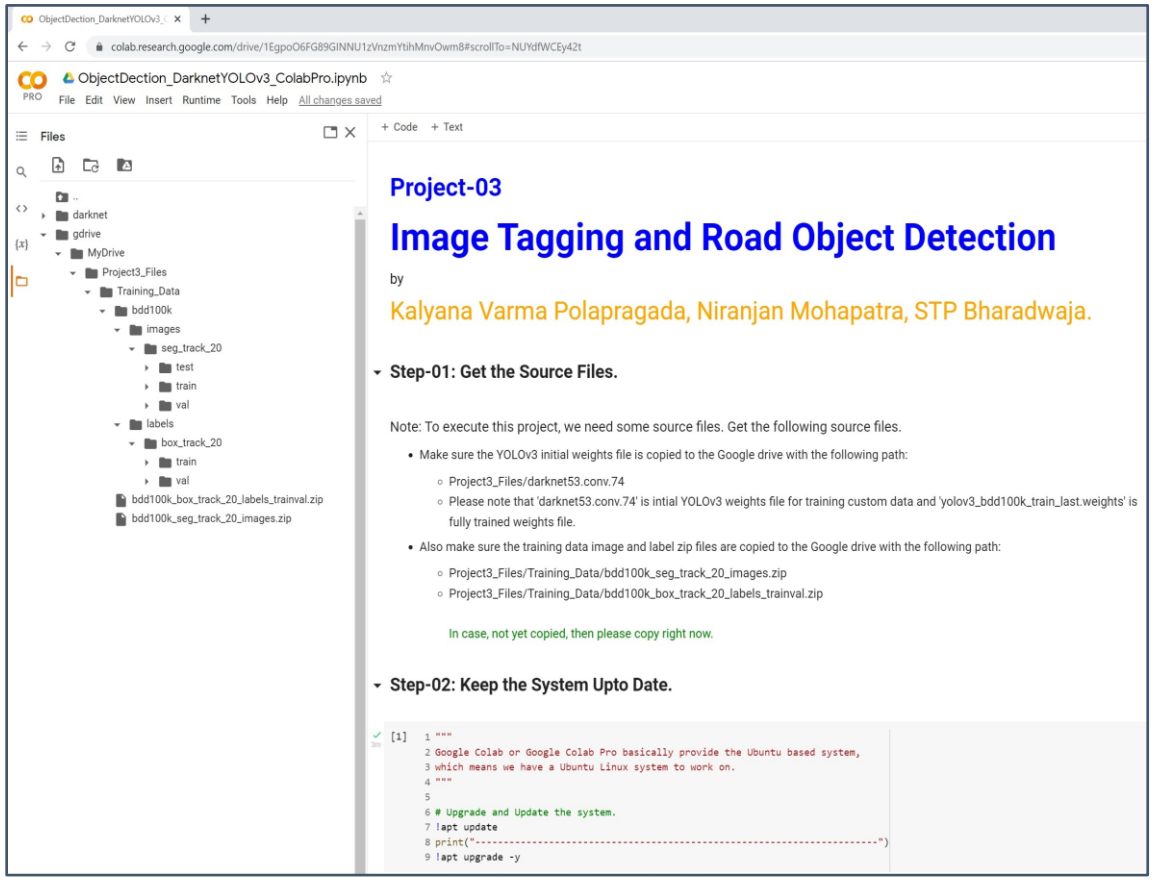
Architecture:

- ***Algorithm***: YOLOv3
  - ✓ YOLO is a fully convolutional network and its eventual output is generated by applying a 1 x 1 kernel on a feature map.
  - ✓ In YOLO v3, the detection is done by applying 1 x 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 ‘darkent’ 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)






```
[ ]  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 -xvzf "/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

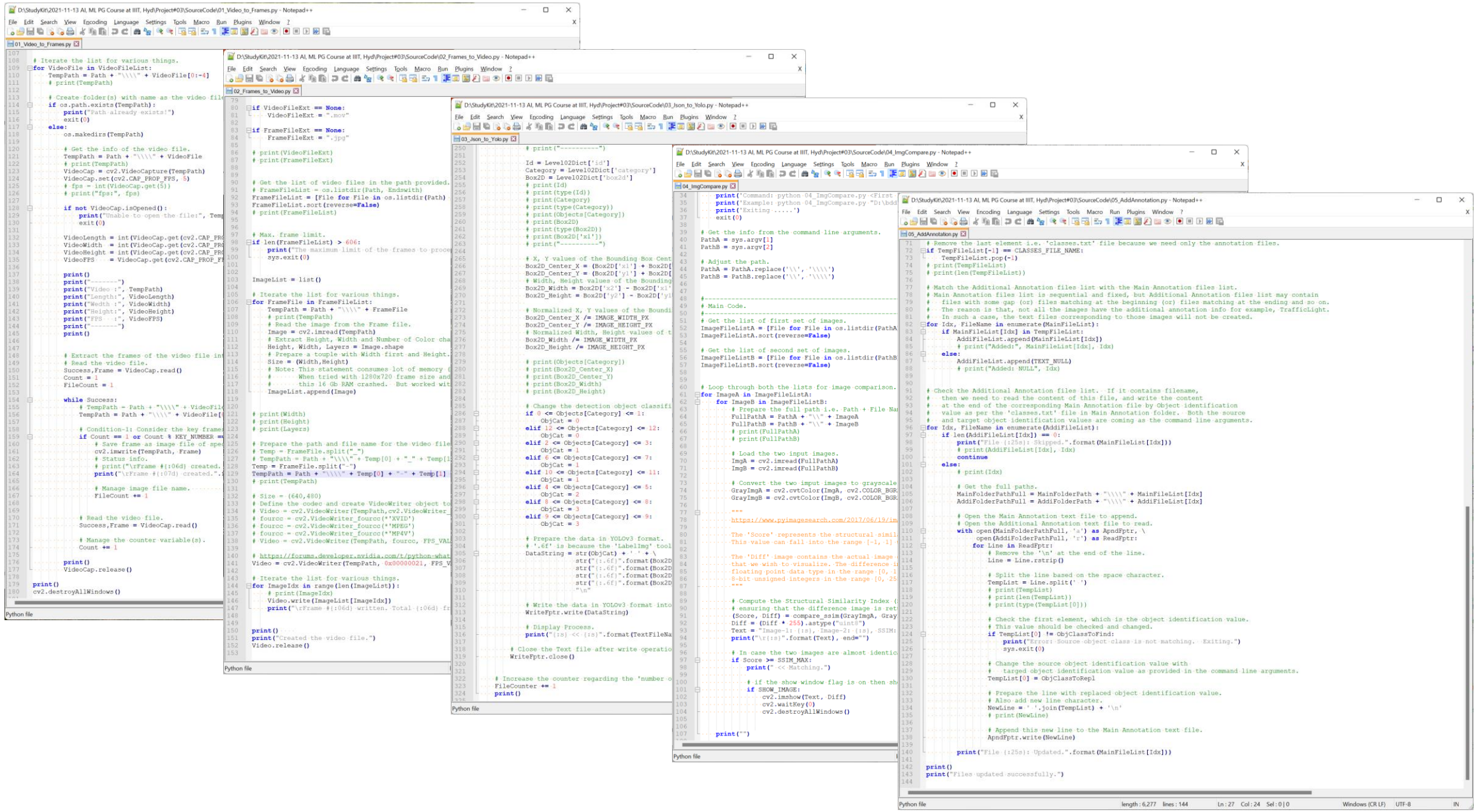
Tools:

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

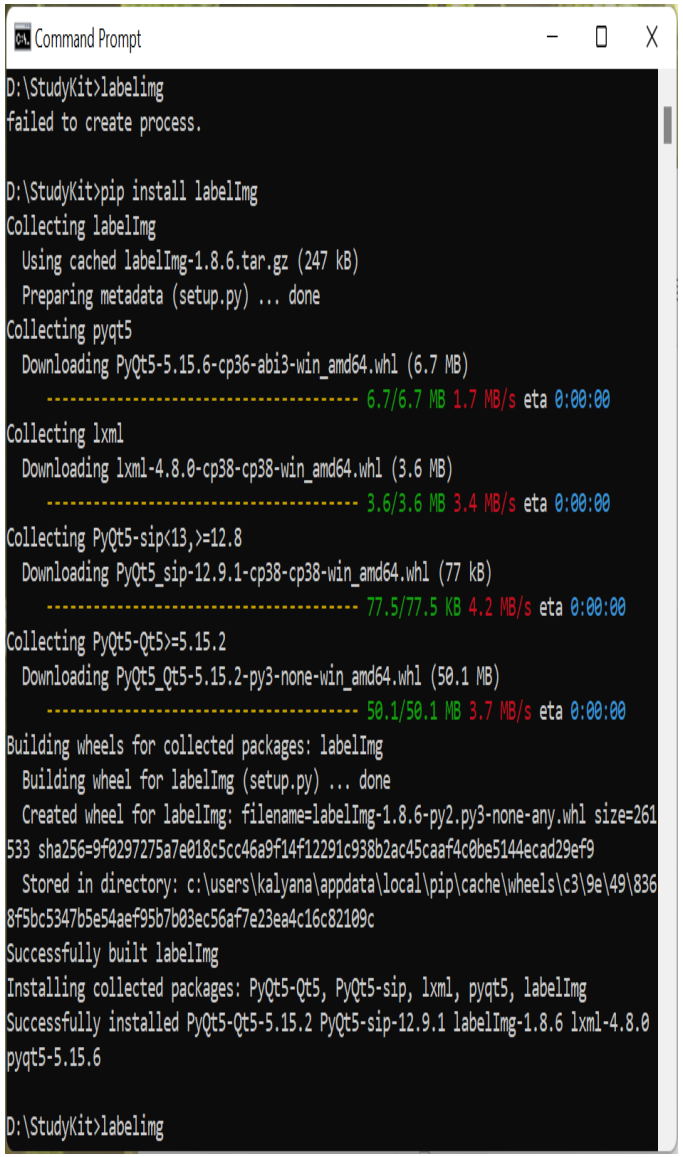
Training Images:

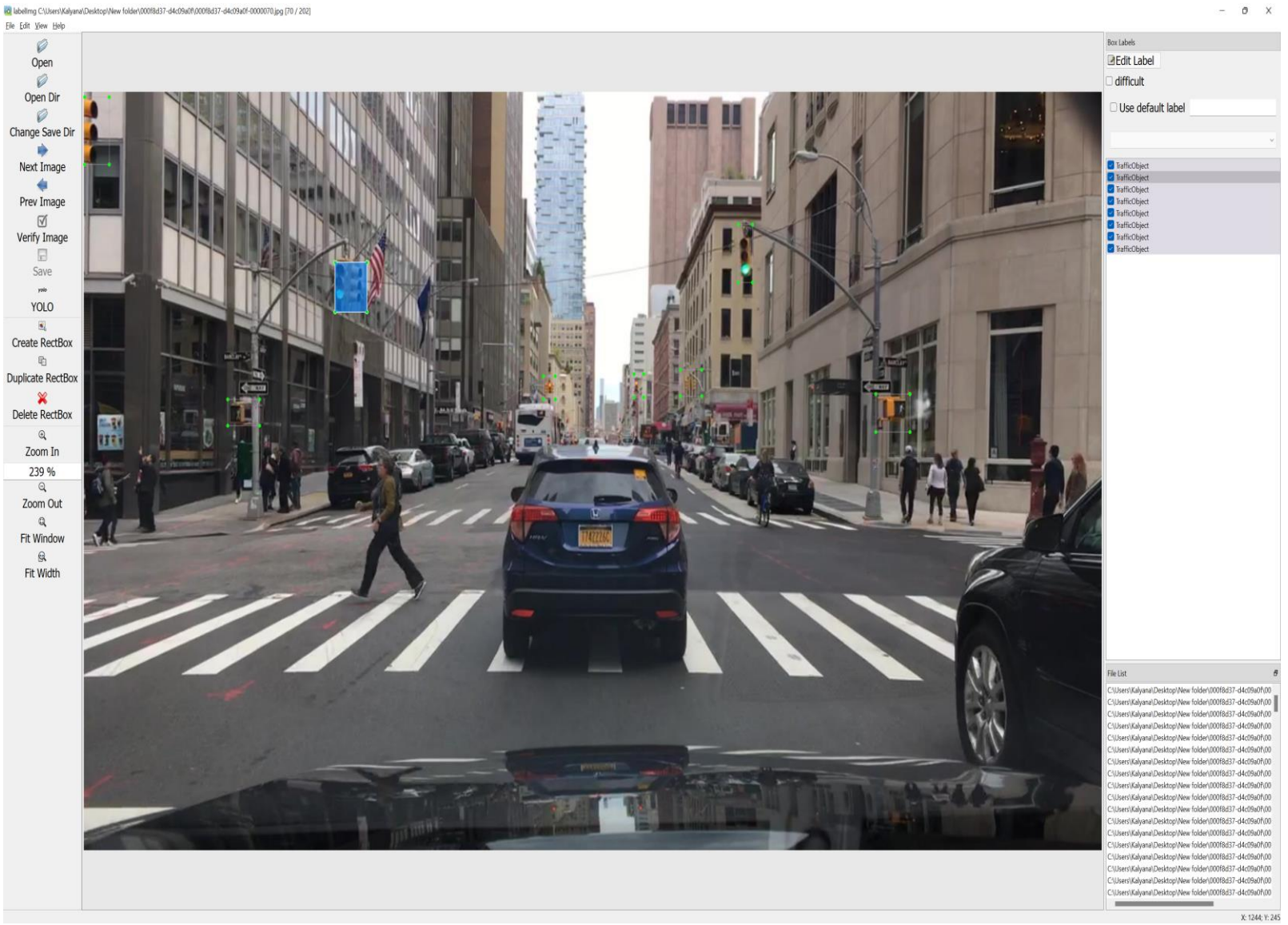
```
Total image files from 'train' directory         : 30,327
Corresponding text files from 'train' directory : 30,327
Total image files from 'val' directory           :  6,329
Corresponding text files from 'val' directory    :  6,329
Total image files in the 'bdd100k' directory     : 36,656
```

Dataset (Tools)

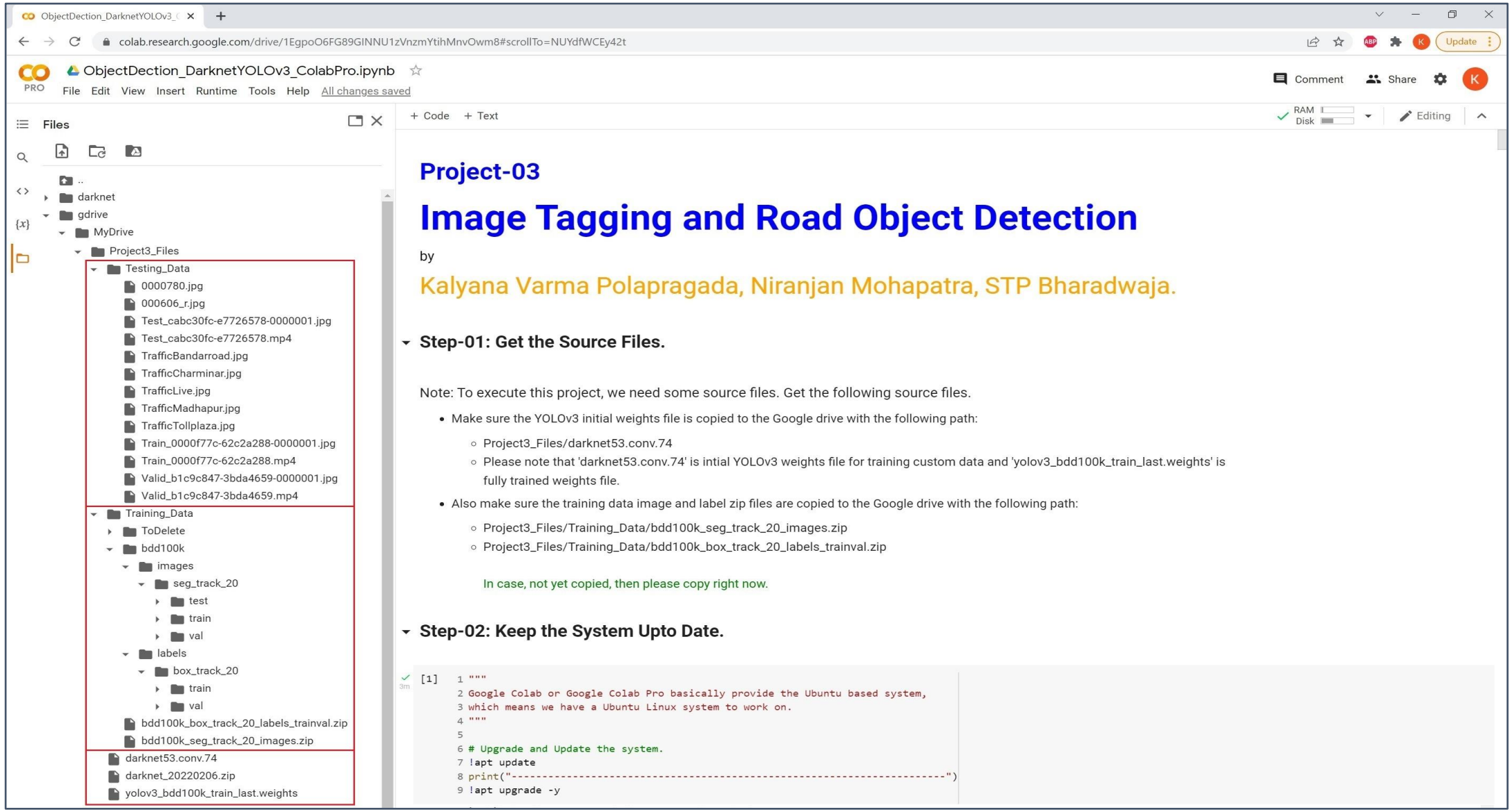


Dataset (Annotation)

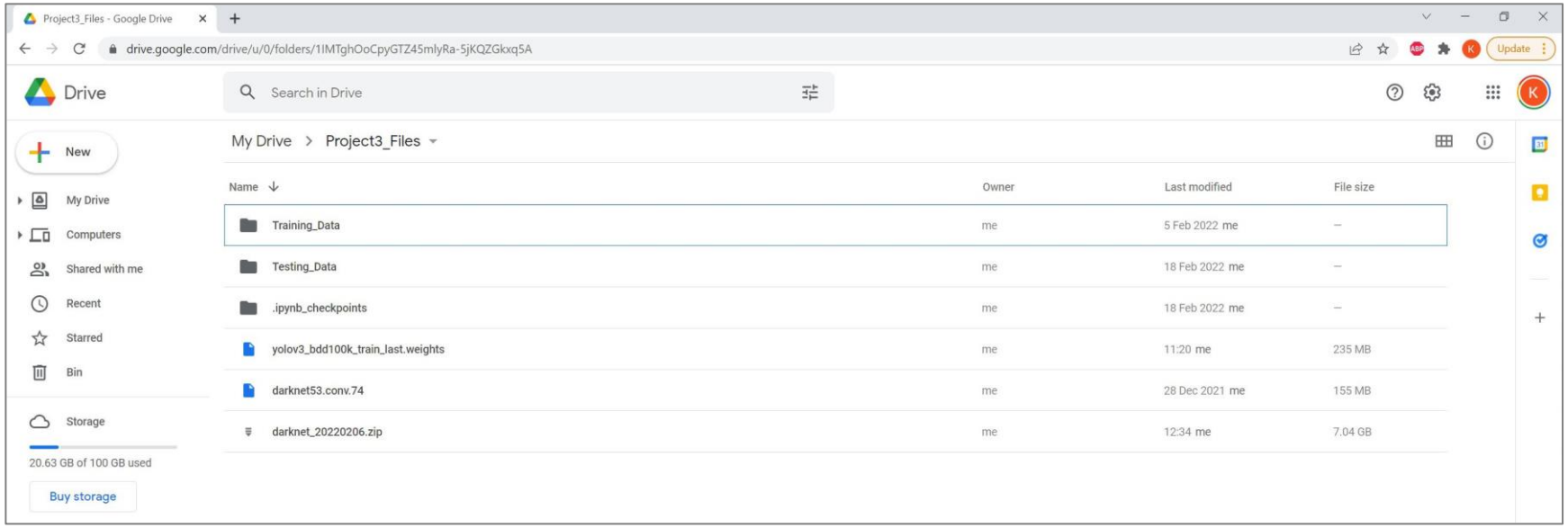




Model Training:



Model Training:



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

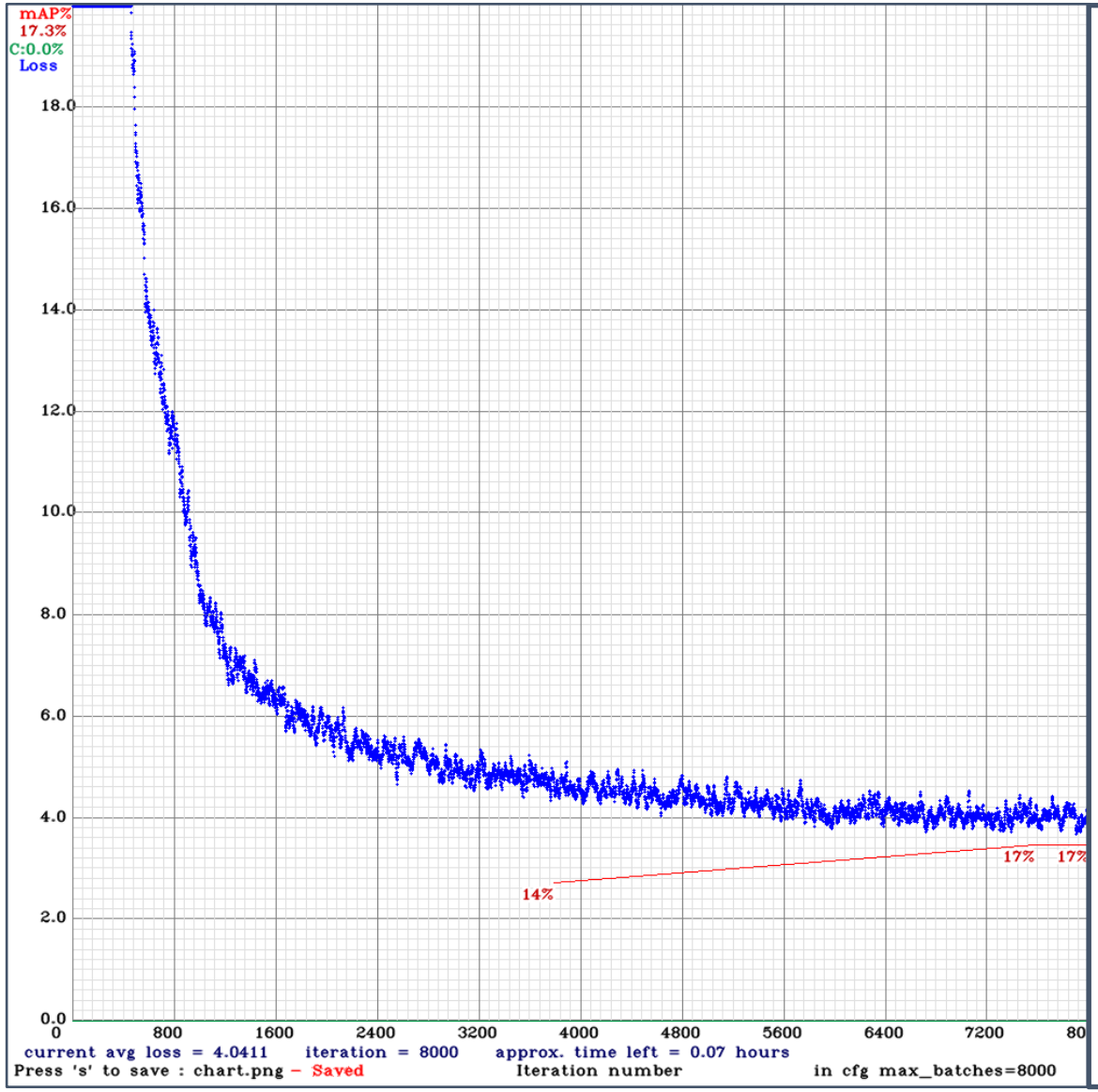
2. **Accuracy:**
   - Accuracy is the ratio of the number of correct predictions to the total number of input samples.
   - `Accuracy = (TP+TN)/(TP+TN+FP+FN)`

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.
   - `Precision = TP/(TP+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!
   - `Recall = TP(TP+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.
   - `F1-score = (2*Precision*Recall)/(Precision+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:

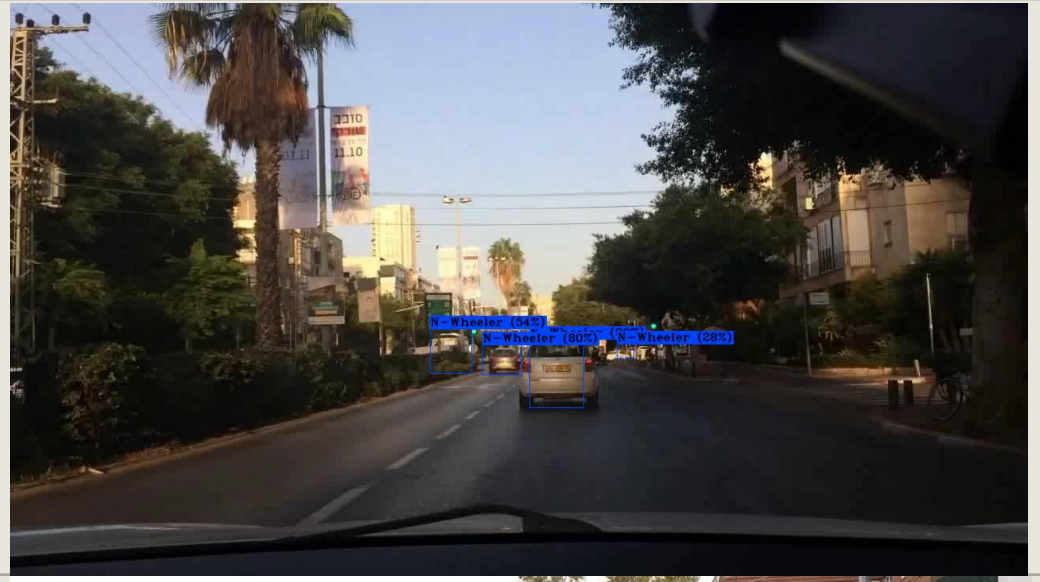
`IoU = Area of Intersection / Area of Union`

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.

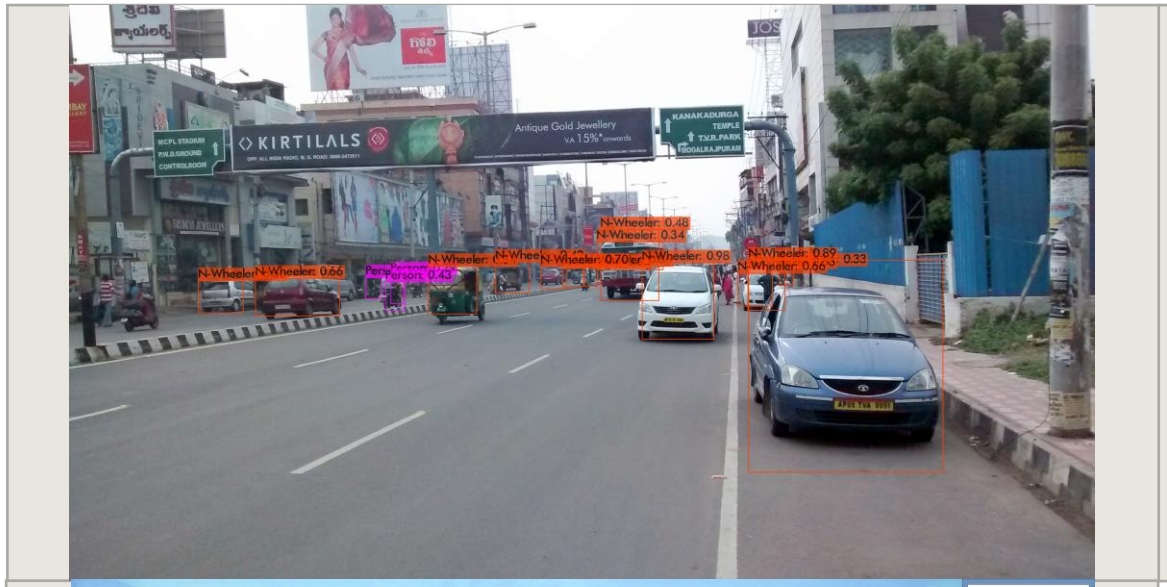
Outcomes: Testing with ‘bdd100k’ Data

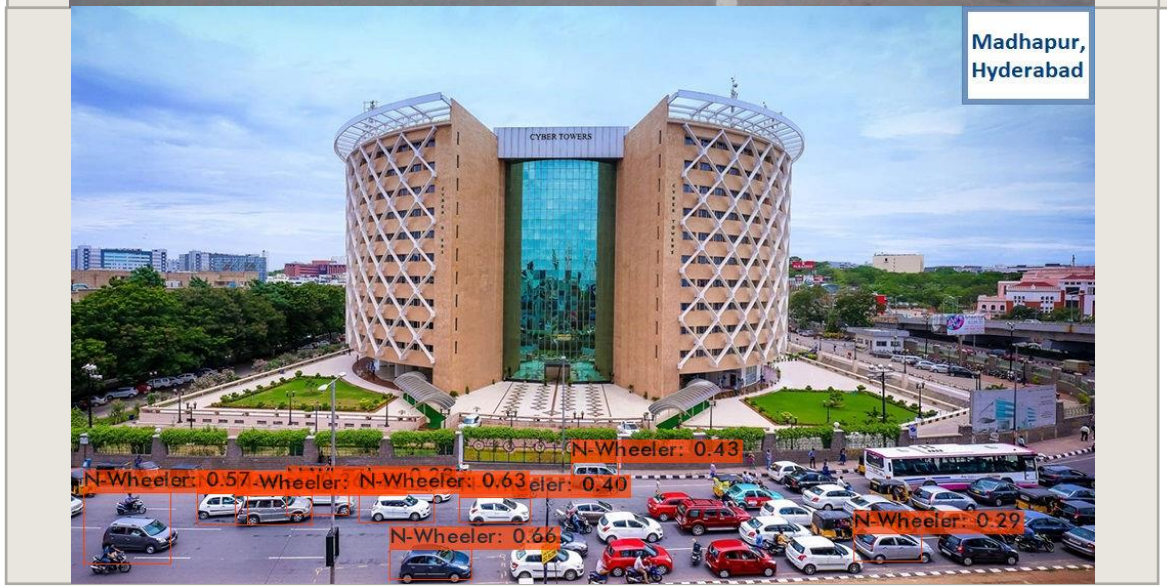
- `classes.names`
- `yolov3_bdd100k_test.cfg`
- `yolov3_bdd100k_train_last.weights`



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

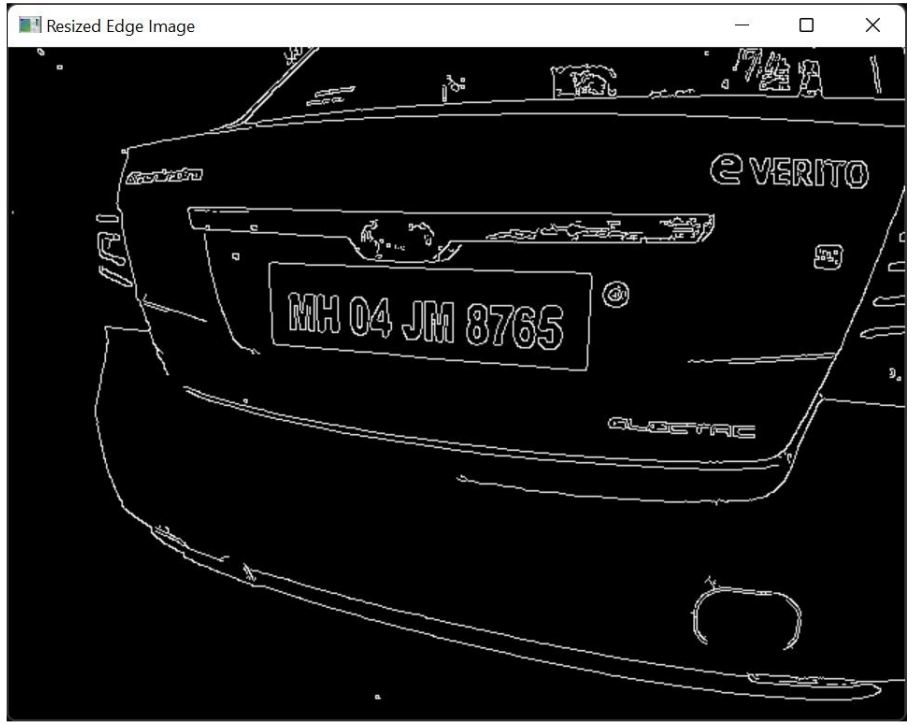
Imbalanced dataset: From the statistics, the dataset is significantly imbalanced. The top class object (N-Wheeler) account for 80% of all the objects and the top 2 class objects (N-Wheeler, Person) make up 96% of the total objects. The most frequent object class is 'N-Wheeler' that comprises 80% of the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

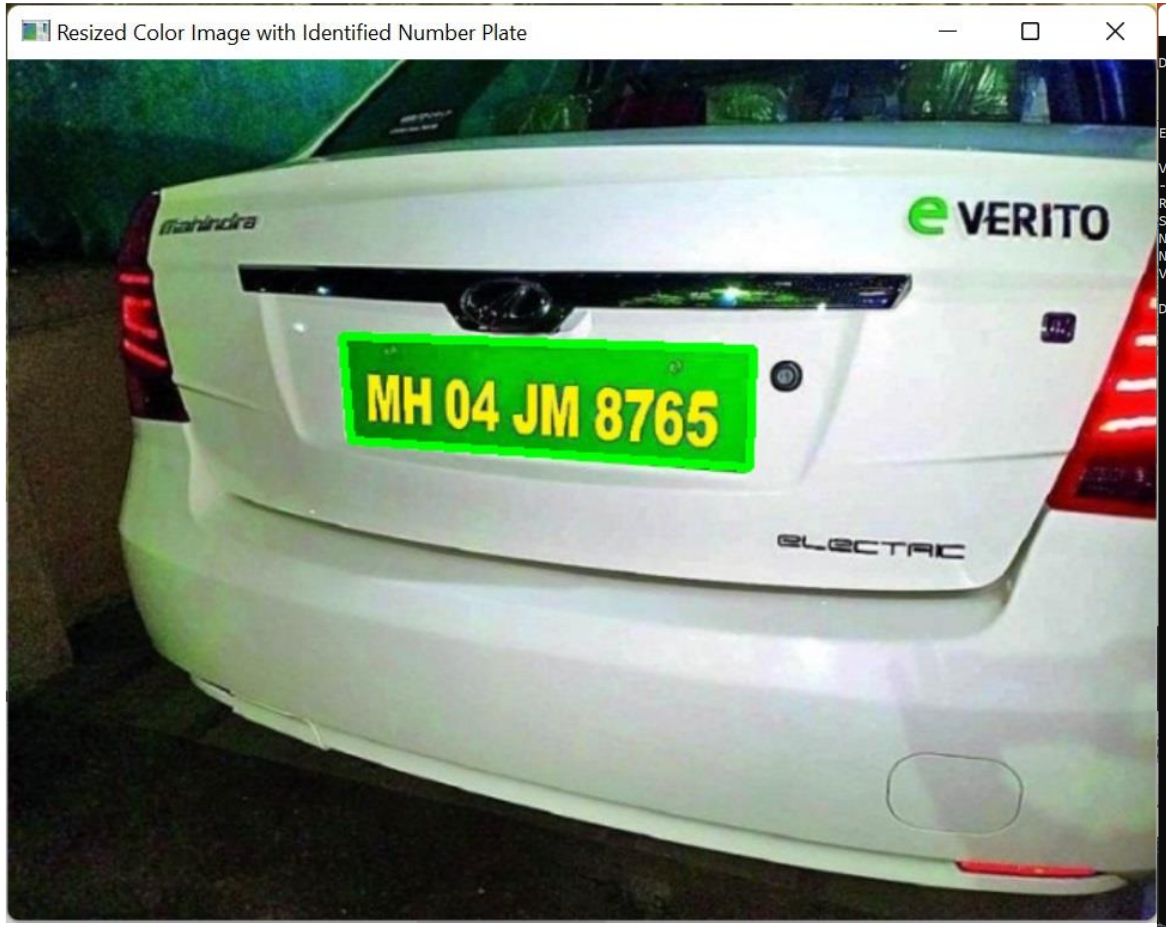
Fixing the class imbalance problem:

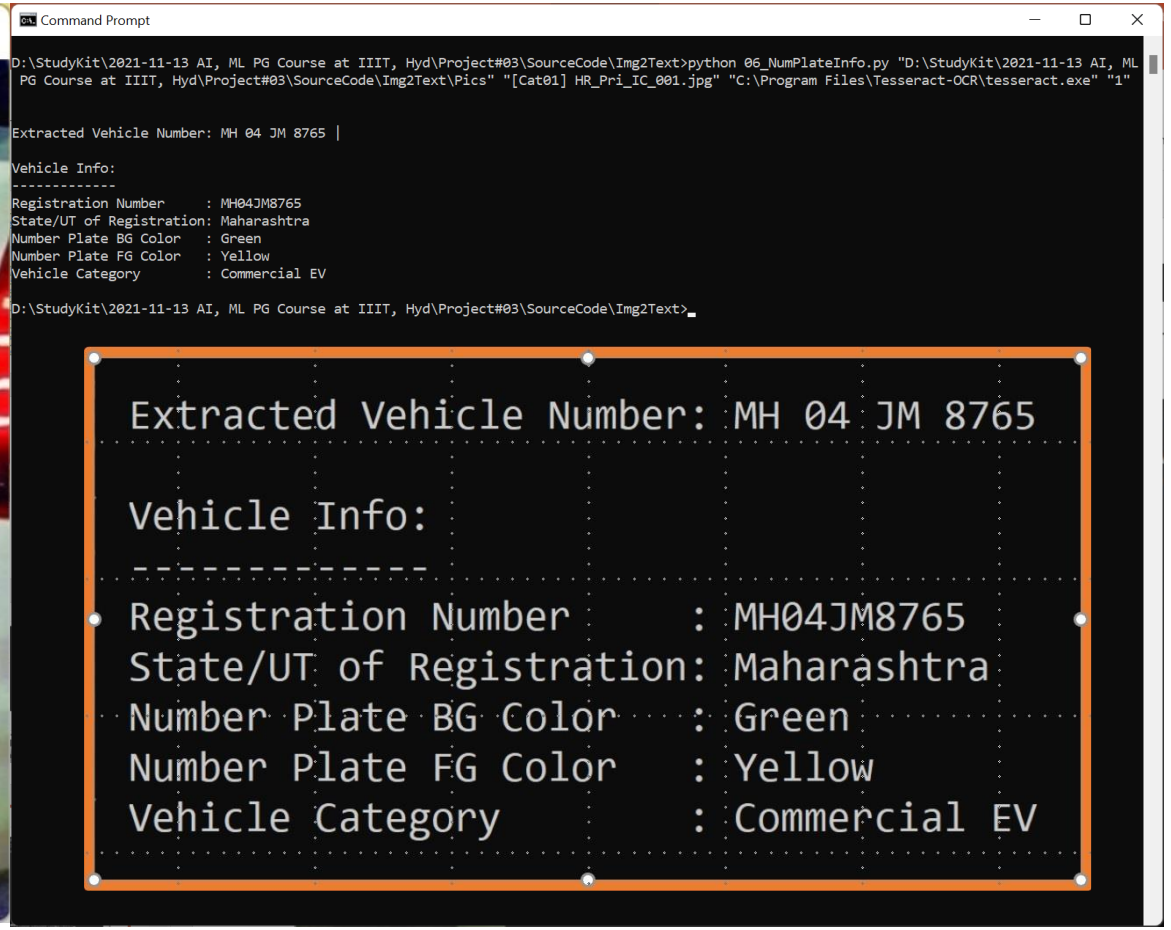
- The first method is called 'Under-sampling' which means to sample less from the majority classes.
- The second method is called 'Oversampling' which means to sample more from the minority classes.
- The result of the 'Under-sampling' or 'Oversampling' is resampled dataset.
- A completely balanced dataset 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):

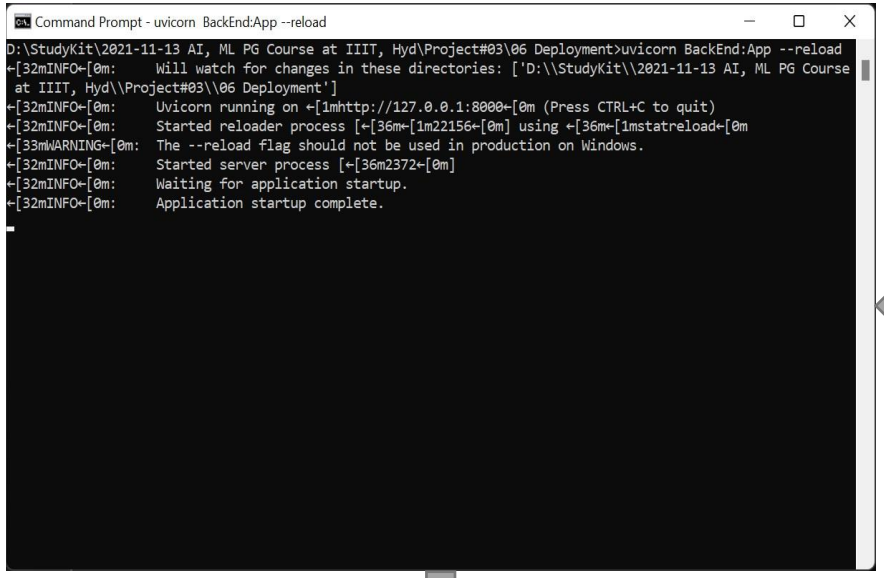


Unique Feature of the Project (Tried):

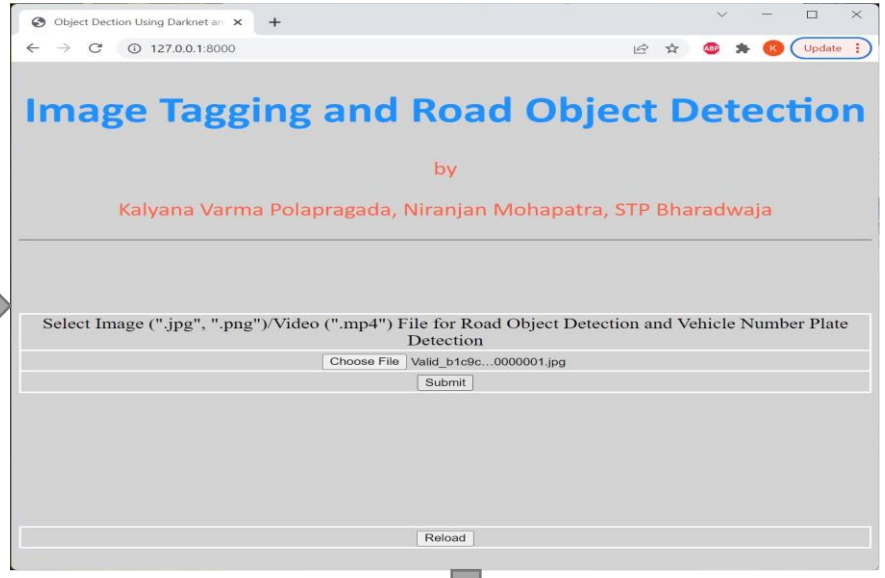


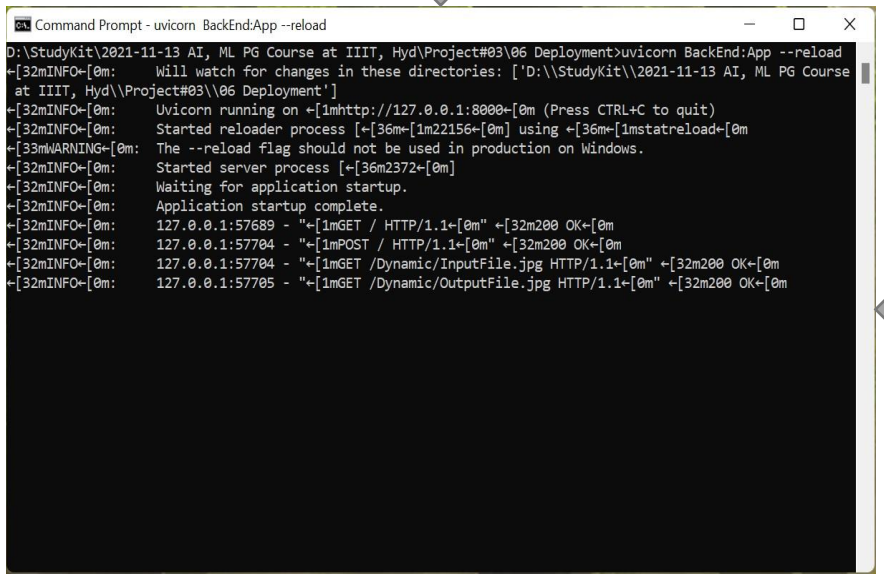


Local Deployment:

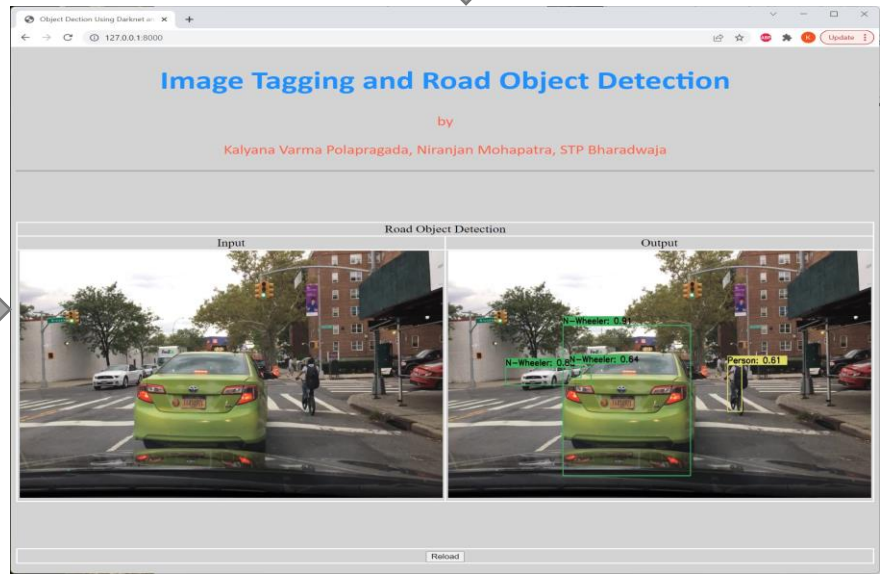


Server UI

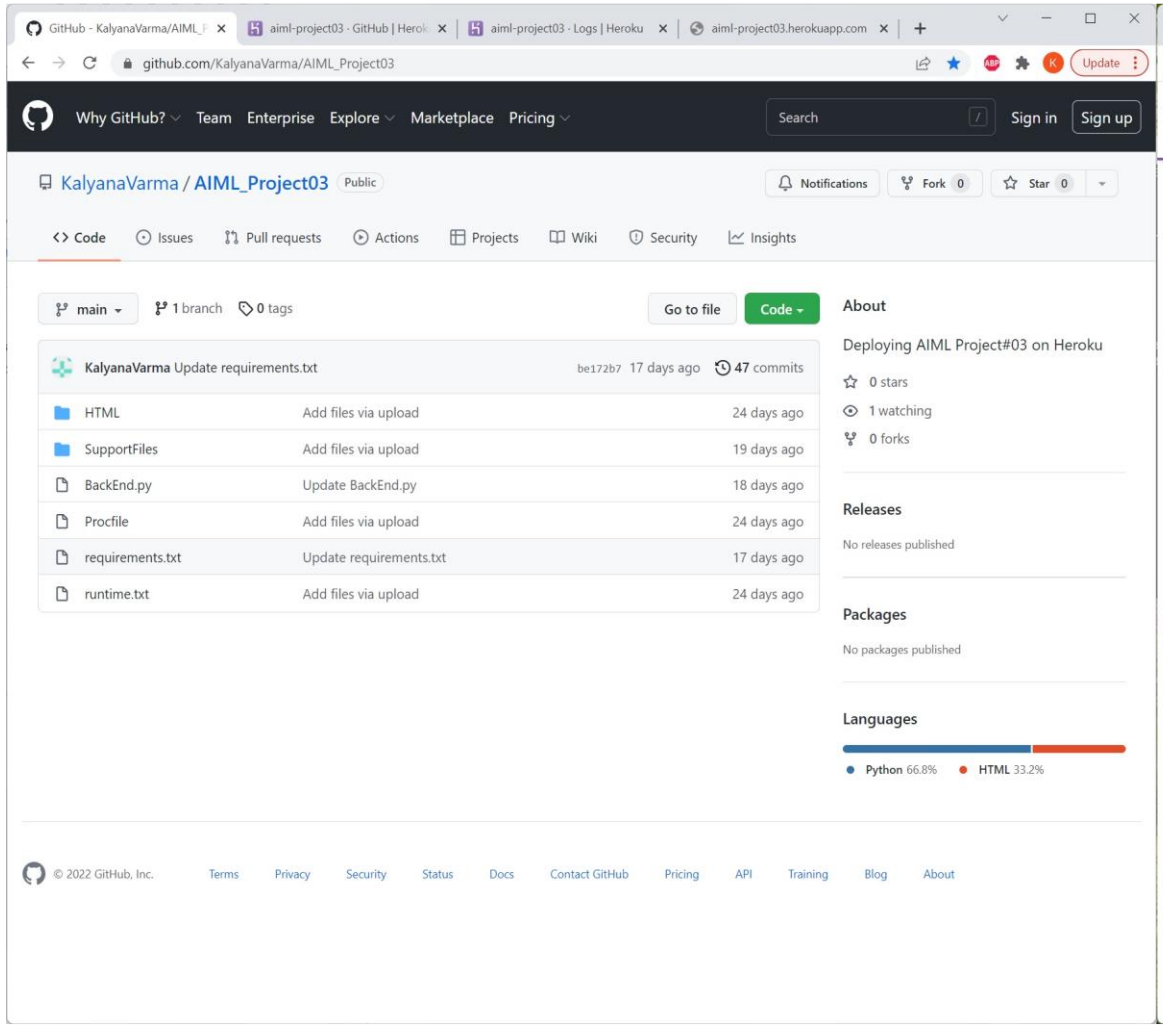


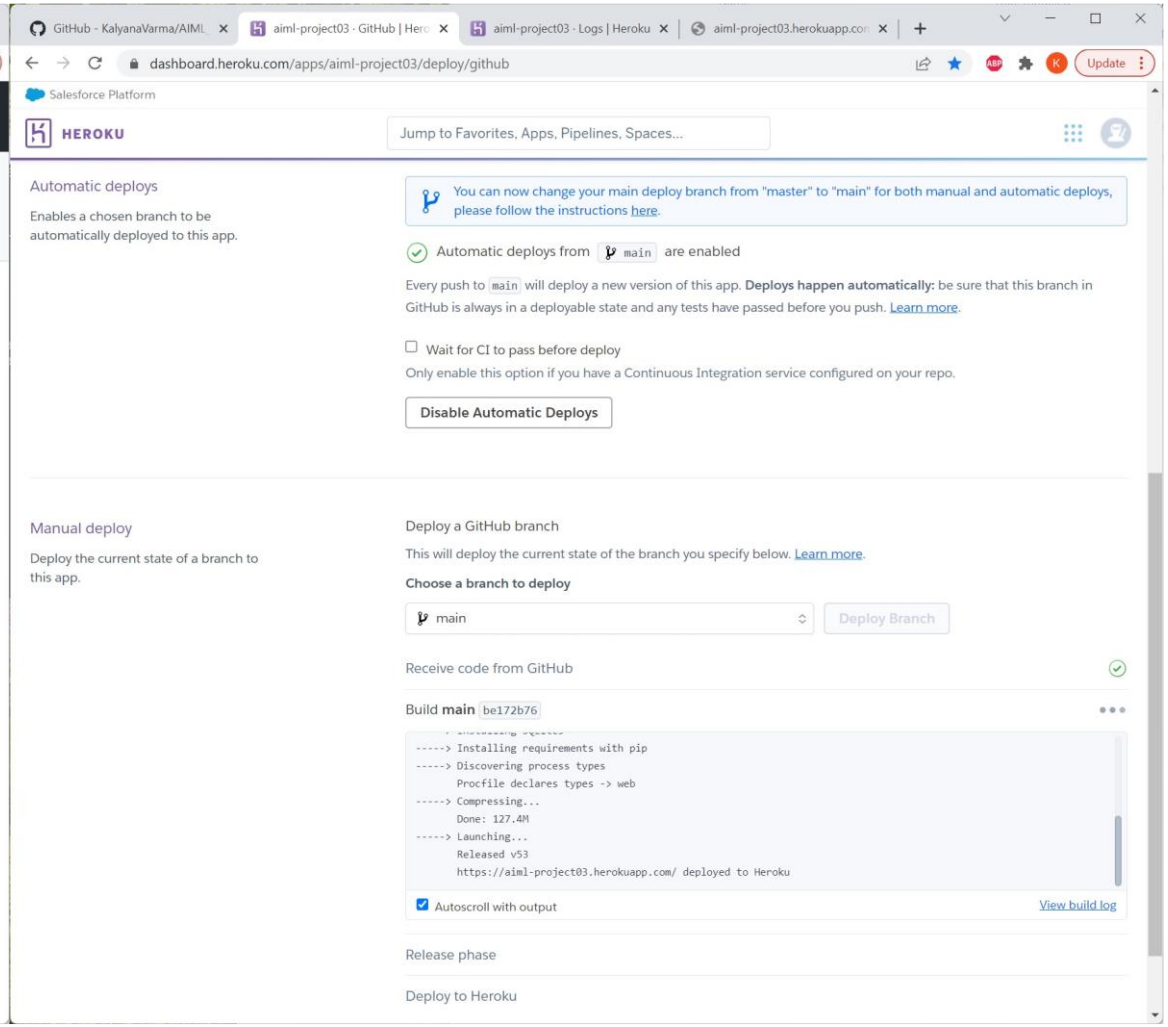


Server UI

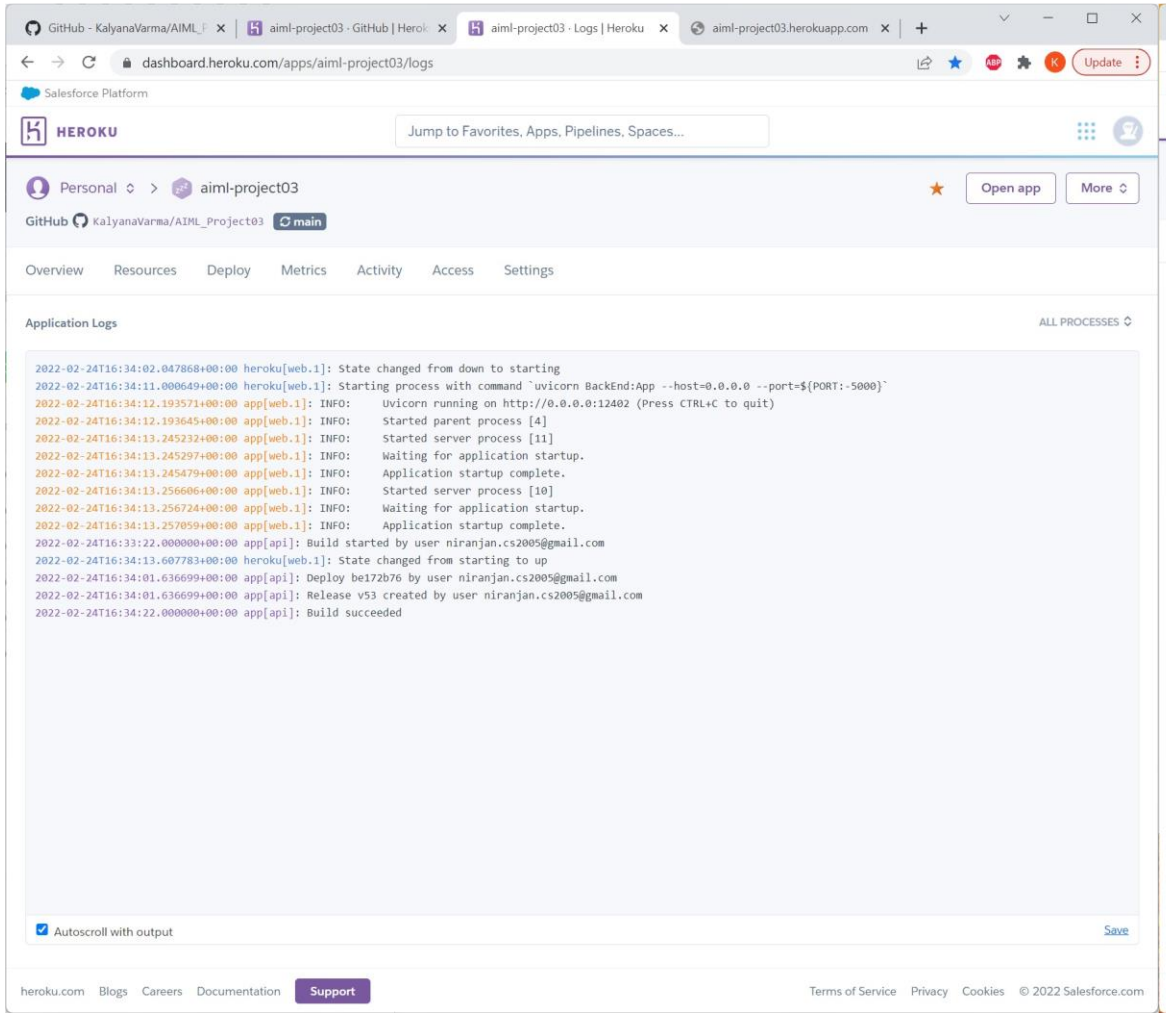


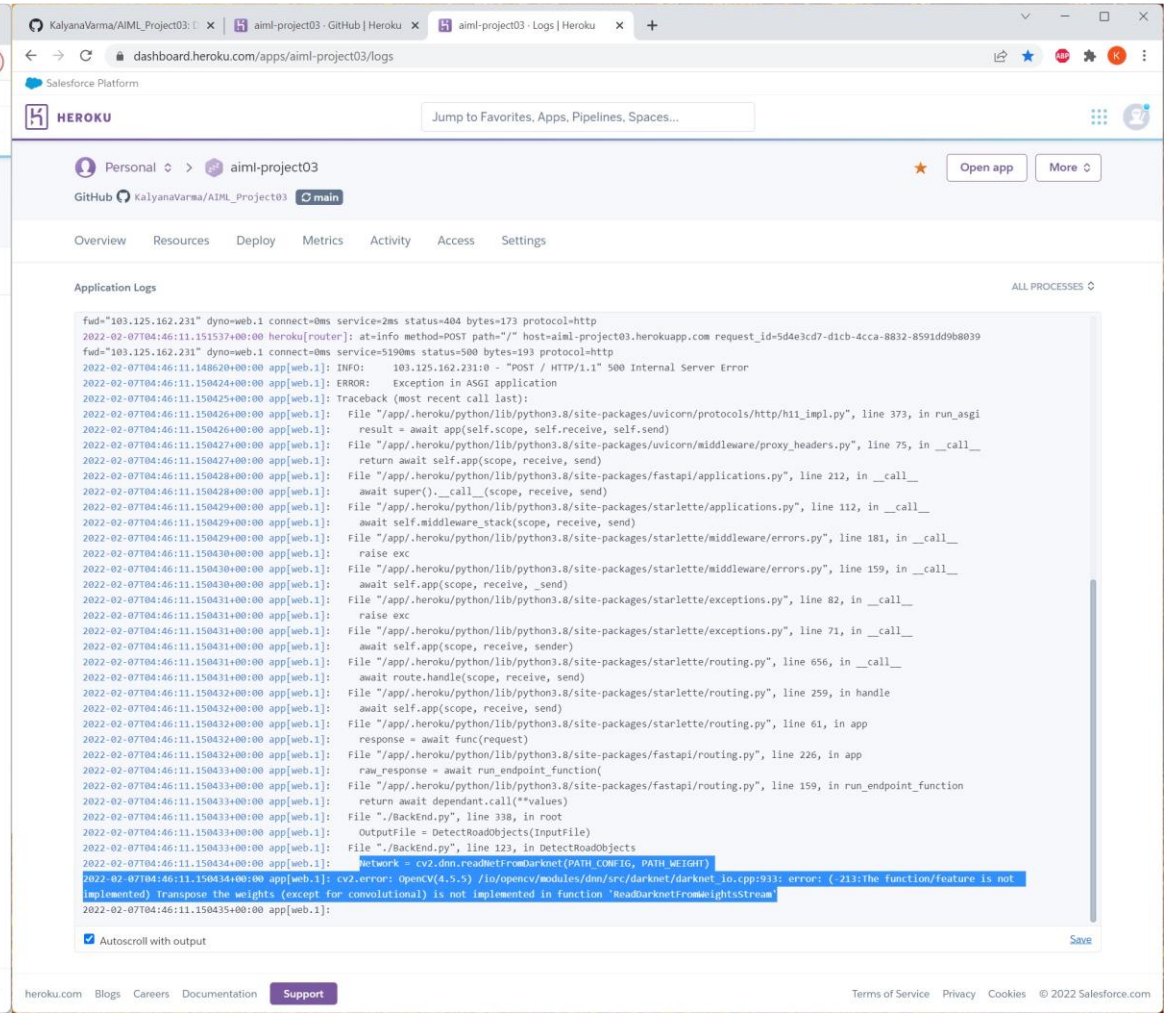
Cloud Deployment:



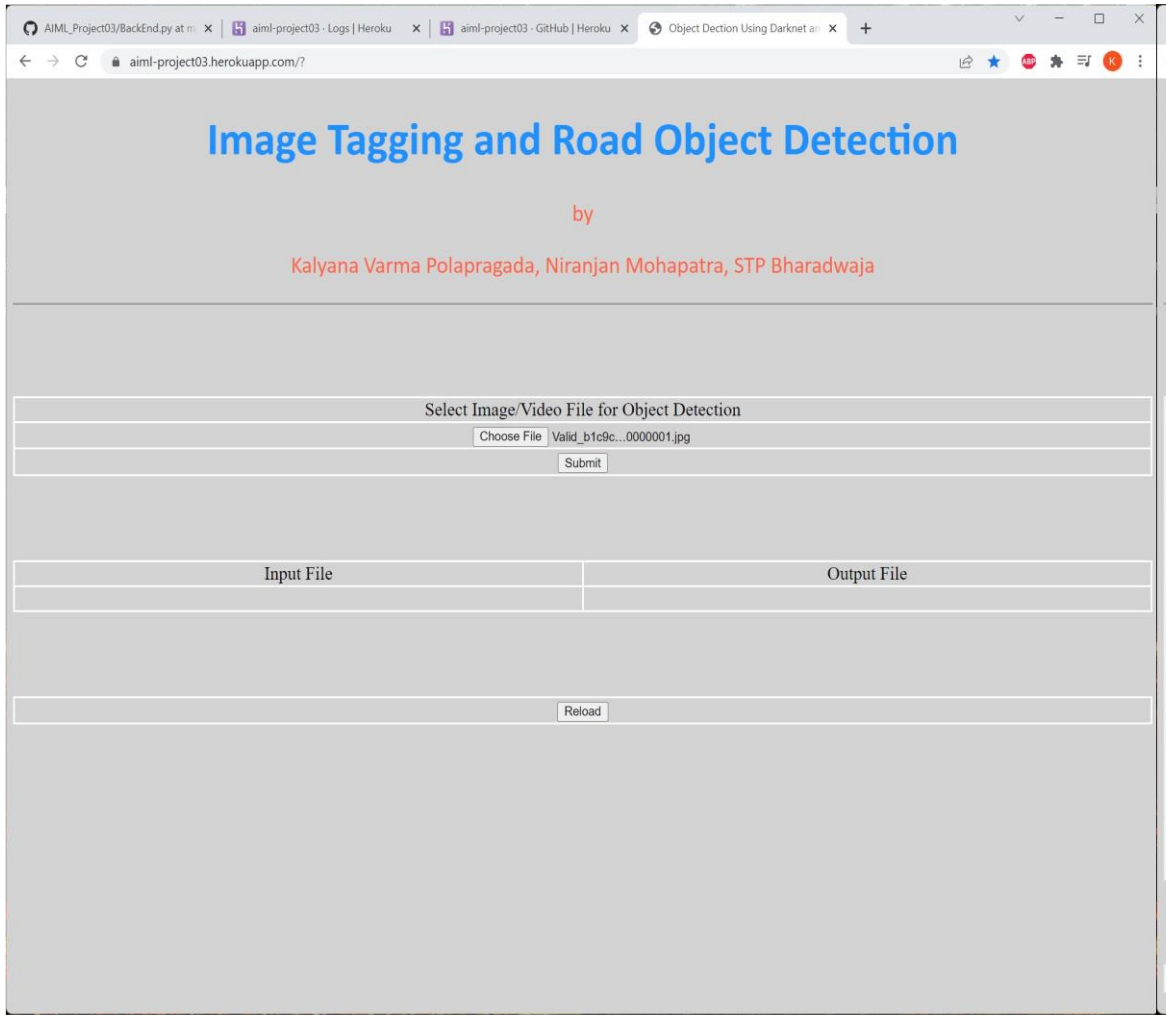


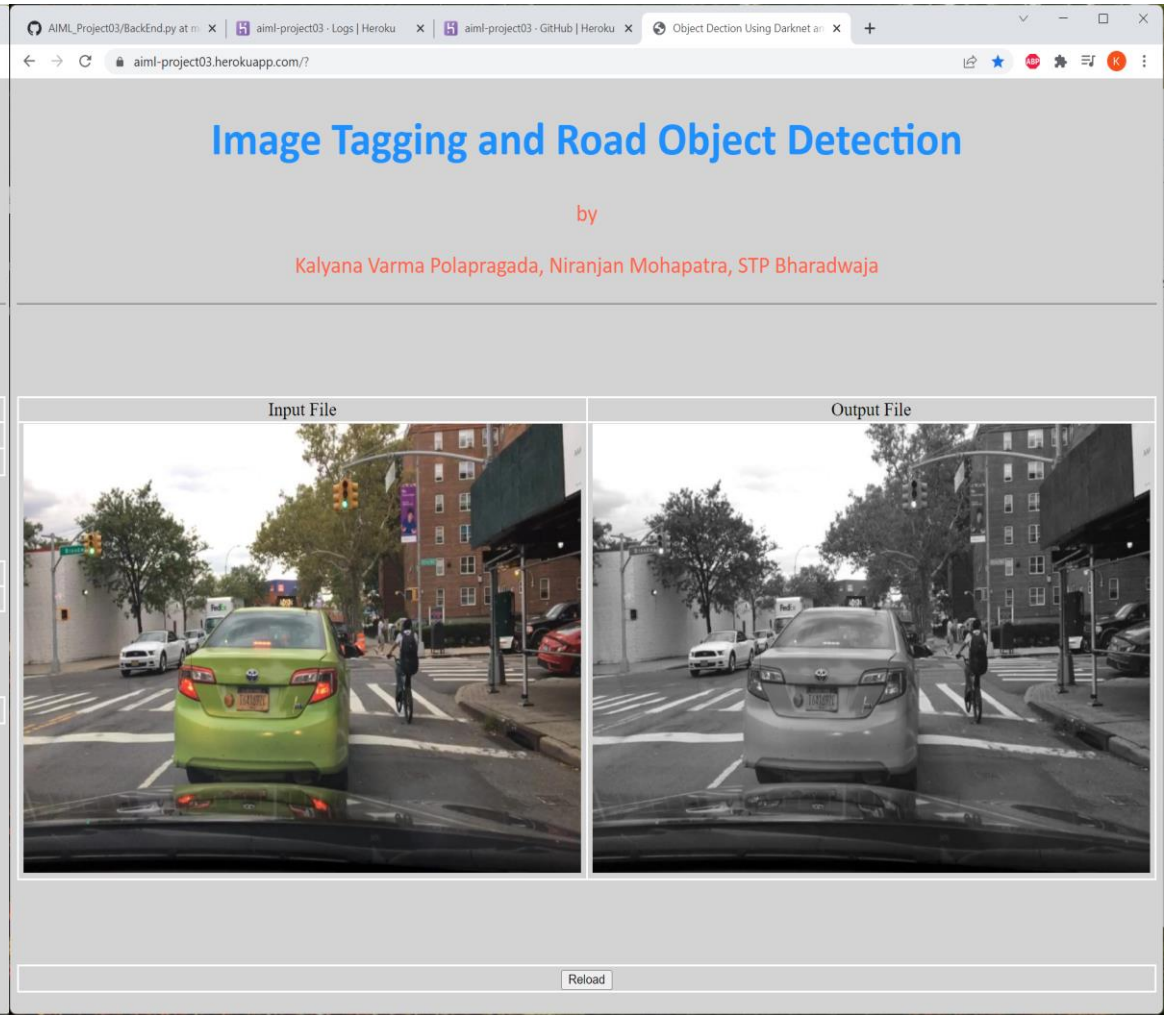
Cloud Deployment:





Cloud Deployment:





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.

Kalyana Varma Polapragada

Niranjan Mohapatra

The Team

Bharadwaja STP


automates
unstructured
data
AI
speech
interdisciplinary
scientific
systems
model
processes
expert
structured
human
identify
machine
insights
intelligence
field
make
apply
extract
especially
computer
knowledge
building
recognition
branch
include
simulation
Artificial
method
algorithms
actionable
application
science
learning
processing
analytical
based
analysis
language
Thanks Questions
Specific
Thanks Thanks
natural
broad
domains
learn
minimal
intervention
Questions Thanks vision
Questions Questions
range
decisions
patterns
idea