# Applied CV Coding Assignment

## Introduction

This document explains the analysis made on the dataset, building a model pipeline and deploying in a docker container, evaluating the pretrained model on BDD dataset and visualizing the performance of the same quantitatively and qualitatively.

## Usage Guide

### Commands to Run Docker Images

|  |  |
| --- | --- |
| Data Analysis | docker pull shravanganji/analysis:v1.0  docker run -p 8501:8501 analysis:v1.0 |
| Inference | docker pull shravanganji/inference:v1.0  docker run –gpus all -p 8502:8502 inference:v1.0 |

### 2.2 GitHub Details

Below is the link to access the given assignment.

[**https://github.com/shravanganji/bdd100k-assignment**](https://github.com/shravanganji/bdd100k-assignment)

## Data Analysis

Github folder data\_analysis

Annotations have been transformed into a CSV file for data analysis, and the script is annotation\_to\_csv.py available in the data\_analysis folder in github

***Note****: The Docker image does not include this due to the large dataset and storage constraints.*

The analysis requires CSV files as input, which can be downloaded from [csv\_data](https://drive.google.com/file/d/1WS-Pqje2d42mjxj6WJuVrrUtsrj6R9qE/view?usp=sharing)

1. **Training Set Statistics:**
   1. Dataset size: Rows: 1234, Columns: 1234
   2. Number of Images in Training Set: N
2. **Validation Set Statistics:**
   1. Dataset Size: Rows: 1234, Columns: 1234
   2. Number of Images in validation dataset: N

**Insights into the data reveal the below pattern:**

1. The "Car" category contains the most annotations, while the "Train" category contains the fewest annotations.

2. There are fewer observations during night-time.

3. Each category exhibits some outliers, visualizable via the link generated from the analysis Docker image.

The dataset includes various categories available in segmentation with bounding boxes (x1, y1, x2, y2). However, it is important to note that the training dataset does not contain all the categories outlined in the documentation.

Moreover, it exhibits category/class imbalance. For retraining, it is necessary to balance it in order to prevent bias towards predictions which might be the case of overfitting for some classes

For some categories data is missing

## Inference

Github folder : Inference

Dataset can be downloaded from [Dataset](https://drive.google.com/file/d/1kGQxFdTsXMTvx8lI6P4HGdYsKilZRRlh/view?usp=sharing)

* Validation Annotation File Conversion:
  1. The validation annotation file is converted to the COCO dataset format.
  2. The conversion process is done by using bddtococo.py available in the inference folder in [github](https://github.com/shravanganji/bdd100k-assignment)
* Output Format and Evaluation:
  1. The resulting outputs are formatted in JSON.
  2. These outputs serve as the foundation for evaluation procedures.
* Inference Script Execution:
  1. The script executes within a timeframe of 20-25 minutes.
  2. This ensures comprehensive evaluation while maintaining efficiency.
* Evaluation Results Access:
  1. Upon completion of the evaluation process, users receive a link for effortless access to the results.

**FRCNN Model:**

The Faster R-CNN (FRCNN) model has been selected for inference due to the following advantages:

1. **Higher Accuracy:** FRCNN model offers superior accuracy in object detection tasks.
2. **End-to-End Training**: It supports end-to-end training with custom datasets, ensuring adaptability to diverse applications.
3. **Elimination of Sliding Window**: FRCNN eliminates the need for sliding window techniques as it incorporates Region Proposal Network (RPN).
4. **Shared Convolutional Layers**: Convolutional layers are shared between RPN and object detection, enhancing computational efficiency.
5. **Pre-Trained Models**: Availability of pre-trained models facilitates efficient transfer learning and reduces training time.
6. **Backbone Architecture Flexibility**: Users have the flexibility to choose different backbone architectures, enhancing model customization.

## Evaluation and Visualization

This summary encapsulates the key evaluation metrics and improvements made to the model, providing clear understanding of the model's performance and enhancements.

***Note****: Evaluation is done on the whole validation set but for calculating metrices only 1000 images are used and these metrices are copied from the local system to the docker image to provide visualization along with predicted images*

### Evaluation Summary:

Script for evaluation: evaluator.py

* Mean Average Precision (mAP): [Insert mAP Value]
* Recall, Precision, IoU and AP: Metrics capturing the performance of the model comprehensively assessed.
* Class-wise Precision and Recall: Detailed breakdown of precision, recall and IoU metrics per class.
* Threshold for IoU is 0.5, and can determine for some classes there are no predictions
* Overall the precision and recall is low and hence can be concluded that model fails to detect accurately
* IoU scores are also indicating less overlap of ground truth bbox and predicted bbox
* For categories car, truck, bus and traffic light have slightly higher precision and recall but not consistent enough

### 5.2 Model Improvements:

* **Categories with Balanced Annotations**: Enhancements made to ensure balanced annotations across categories, improving model performance and fairness.
* **Inclusion of Night Scene Data**: Dataset augmented with night scene data to enhance model robustness and performance in low-light conditions.
* **Classification Model for Traffic Light Detection**: Introduction of a specialized classification model aimed at detecting traffic lights, enhancing the model's ability to identify critical objects for traffic management

## Retraining

This summary encapsulates the retraining process using detectron2 FRCNN model.

### Retraining Summary:

Github folder : Retraining

Platform: Detectron 2

Task: Object detection

Model: FRCNN(pretrained)

Dataset: Coco Format

* Detectron2 architecture is used for building the retraining process
* Only 1 class is used for retraining i.e traffic sign
* It has got instances of 239686
* For the retraining process, coco format json is used which is converted from bddtococo.py available in retraining directory in github
* Due to resource constraints, not able to run for 1 epoch, so no evaluation on the validation dataset
* Retraining\_script.py does the retraining on the dataset provided