



BDD 100K DATASET

ANALYSIS AND MODELLING

ABOUT THE DATASET

The Dataset consisted of about **100k** images, split as **70k** for training **10k** for validation and **20k** for training. And altogether it contained **1273707** detection boxes in training and **185945** for validation, which somewhat occurs to be a 10% split of training detection boxes.

The dataset is split into 10 classes as specified in the documentation identifying pedestrian, rider, car, truck, bus, train, motorcycle, bicycle, traffic light, traffic sign. But the labels also contained other two classes in minor numbers such as Other Person and Other Vehicle which were omitted due to their lesser number and non existence in documentation

DATA ANALYSIS

The data analysis is done by differentiating the scenarios in which the data was captured, and predicting if the distribution will affect the results.

Also the training and validation split between scenarios is validated to have similar distribution

Any additional changes that could be made to improve quality of data, or efficiency of the model built using the data is specified.

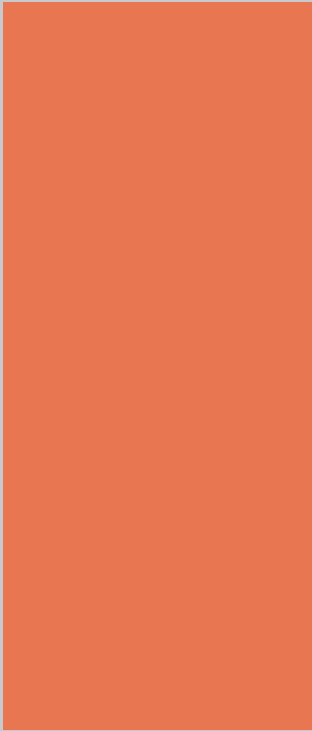




Data Analysis

PROCEDURE TO VISUALIZE THE
FRONTEND USING DOCKER

DOCKER IMAGE NAME

- 
- **kaviasubramanian:road_object_detection:v1.4**
 - **Dockerhub link :**
https://hub.docker.com/r/kaviasubramanian/road_object_detection/
 - Pull and run the docker and execute the python flask file app.py to visualize the frontend
 - **docker run -it --net=host -p 5000:5000 kaviasubramanian/road_object_detection:v1.4**
 - **cd /home/road_object_det**
 - **python3 app.py**
 - JSON parser and graph builder present in visualize.py
 - Data visualization with ground truth bounding boxes is added in the next part.




DETECTION MODEL

MODEL REASONING

- The model chosen for detecting the bounding boxes present in the dataset is YOLOV8.
- Since there was no specification given about deploying device and compute capability ,I preferred to choose the model that gives best accuracy.
- Multiple models were considered based on the SOTA performance, recent techniques , popularity and ease of use.
- Considering all these YOLOV8 was chosen.Since training XL was not possible with the machine I have,I went with yolov8-l for now.

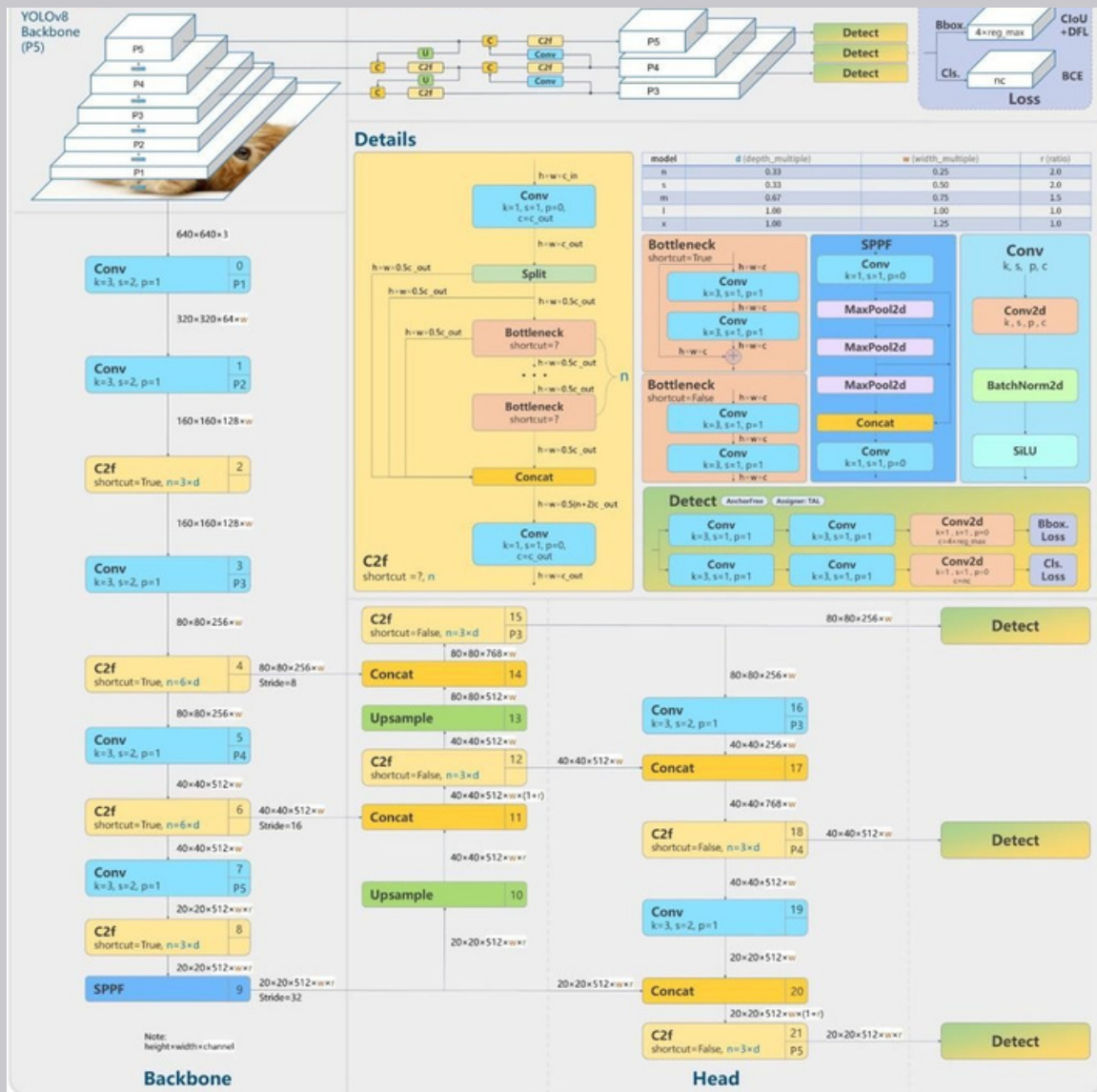
MODEL CHOICE AND EXECUTION



PROCEDURE TO TRAIN MODEL AND VISUALIZE OTPUT

- **Clone the shared git link**
- Specify the mounted data and json path in config.json file .
- Create a virtual environment with Python 3.8.10 and install the requirements file
- Download the model file from onedrive link and paste in the same folder
- Onedrive link:
https://drive.google.com/file/d/1AMXPVSnpwCy3FPaFOHD36U_5ARwKB5ur/view?usp=sharing
- **python train.py** -trains the data with the specified data from config.json file and stores the model as weights in model name specified in config.json.
- **python visualize.py** - gives option to visualize training ground truth data or predictions on validation data.

YOLO v8 MODEL ARCHITECTURE



The important aspects to be noted in the model architecture are:

1. C2f layer that concatenates all the bottleneck layer
2. Anchorless detection that helps fast learning for multiple sizes.
3. Loss function used in the model is CIOU and distributional focal loss for bounding box regression loss, Binary Cross Entropy for classification loss

In this process I have trained the model for **4 epochs** as sample and used that model to evaluate the results.

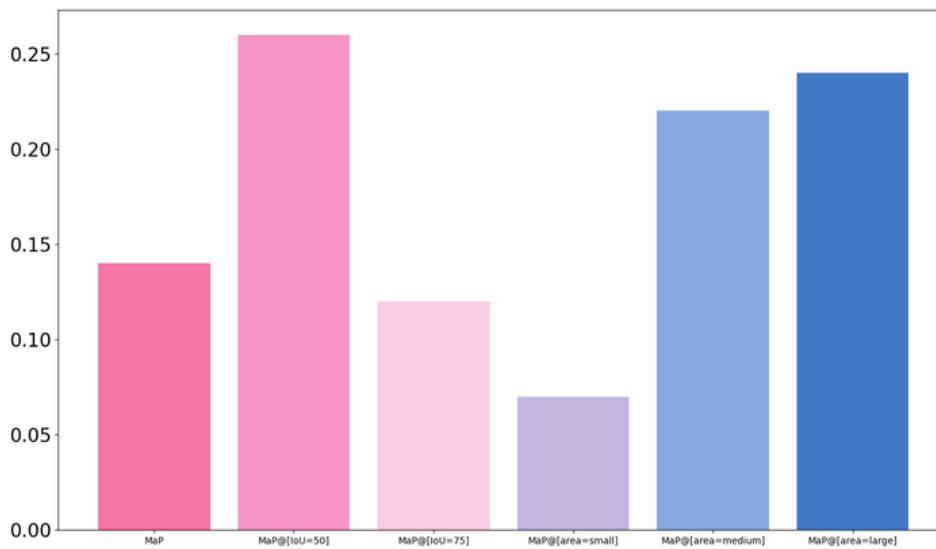
Prediction and Evaluation

VISUALIZING PREDICTIONS AND EVALUTATION METRICS

VISUALIZATION AND METRICS DISPLAY

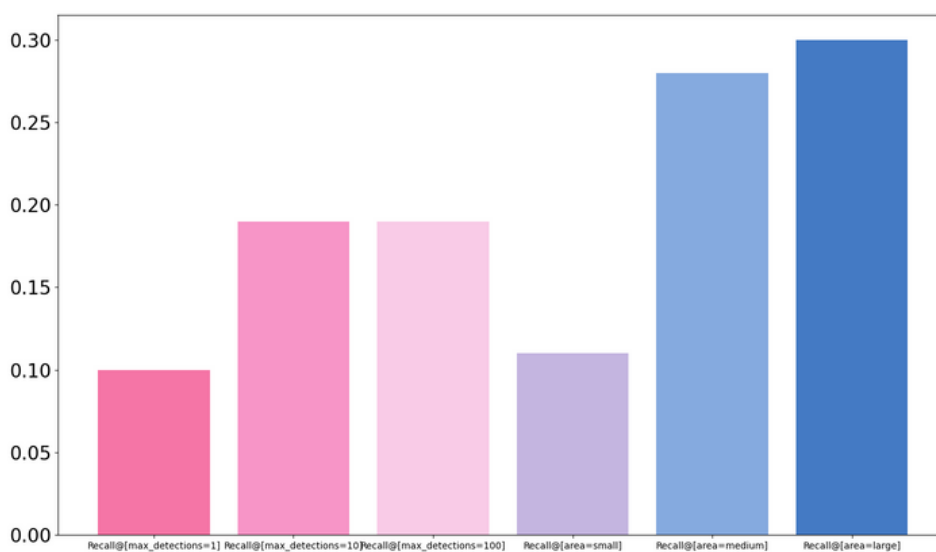
- visualize.py lets us view the validation images with its prediction bounding boxes and ground truth bounding boxes
- run **python visualize.py**
- metrics_display.py helps calculating BoxCOCO metrics and visualizing them in the form of graph
- run **python metrics_display.py**
- The graph images will be written in val_mAP.png and val_recall.png

MAP



When the IOU threshold increases from 50 to 75, the mAP drops. Also it is observed that mAP of large objects reduces compared to smaller objects, this happens because of the difficulty in learning minor features from small area.

RECALL



It is observed that recall increases when max_detections allowed increases till 100. Also as observed in mAP, recall also reduces when size of the object reduces.

Observations

- The Precision and Recall stated here are not so good, because it is trained for 4 epochs alone, accuracy could be increased by training for more epochs.
- Also currently no augmentation is used other than Flip and Resize, additional augmentation could be used.
- Also hyper parameter tuning should be done.
- Basically it is observed that the model performs poorly on small and medium sized objects.
- Multiple reasons are predicted here, either good features learned from large sized objects of same class dominate the small objects features or presence of less number small sized samples compared to large ones .
- In our efforts to void this we could try increasing small objects data ,or simulate small objects by resizing the large objected images .
- Since already the model has multi level feature map inclusion, that part in the model is covered.

Proposed Improvements

ANALYSING THE RESULTS

OTHER ANALYSIS THAT COULD BE DONE

- We have already separated the data based on different scenarios, the same can be used to find which scenarios fail mostly and try to include more data from that scenarios or try to augment to replicate that scenario, like weather, time of day, scene.
- Also a check with whether occluded or truncated images fail in detection would also pose helpful in altering the data