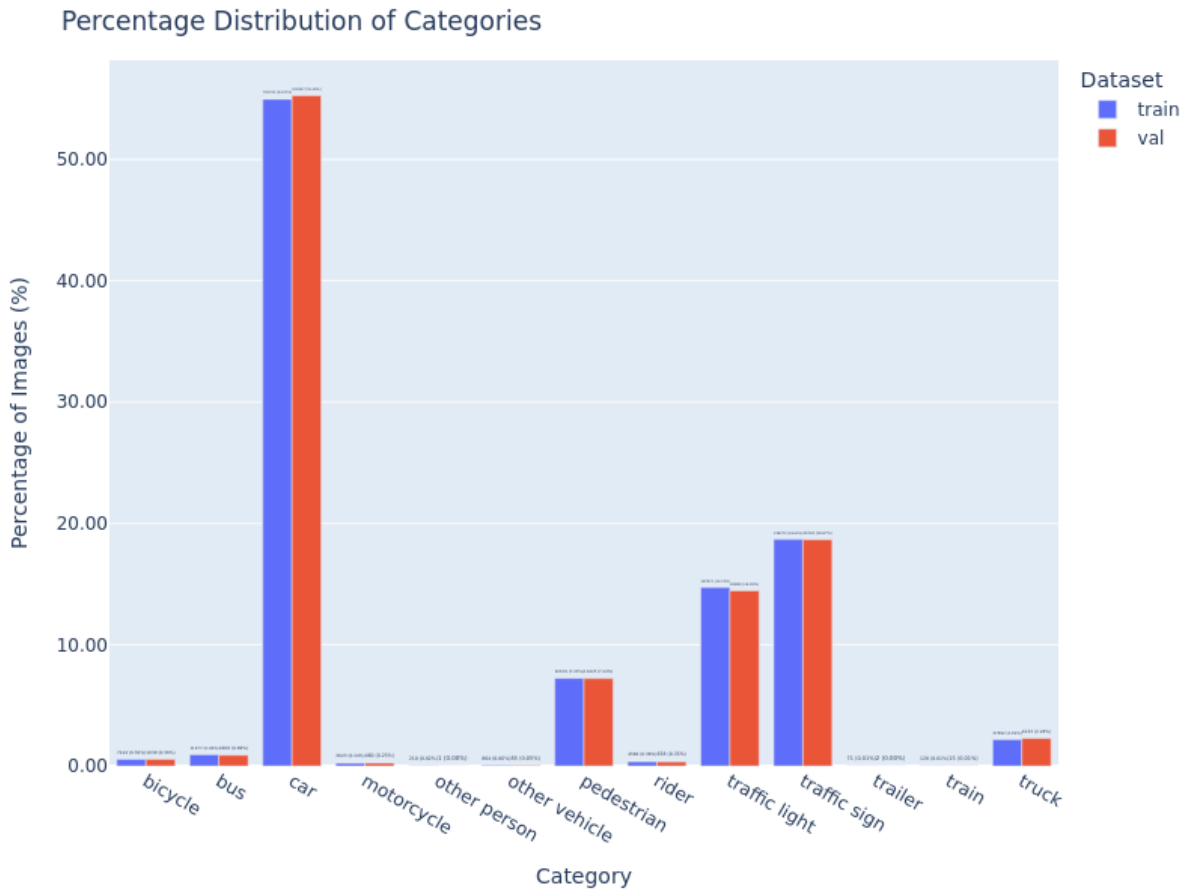
Table of contents

BDD100K Object Detection.

Exploratory Data Analysis

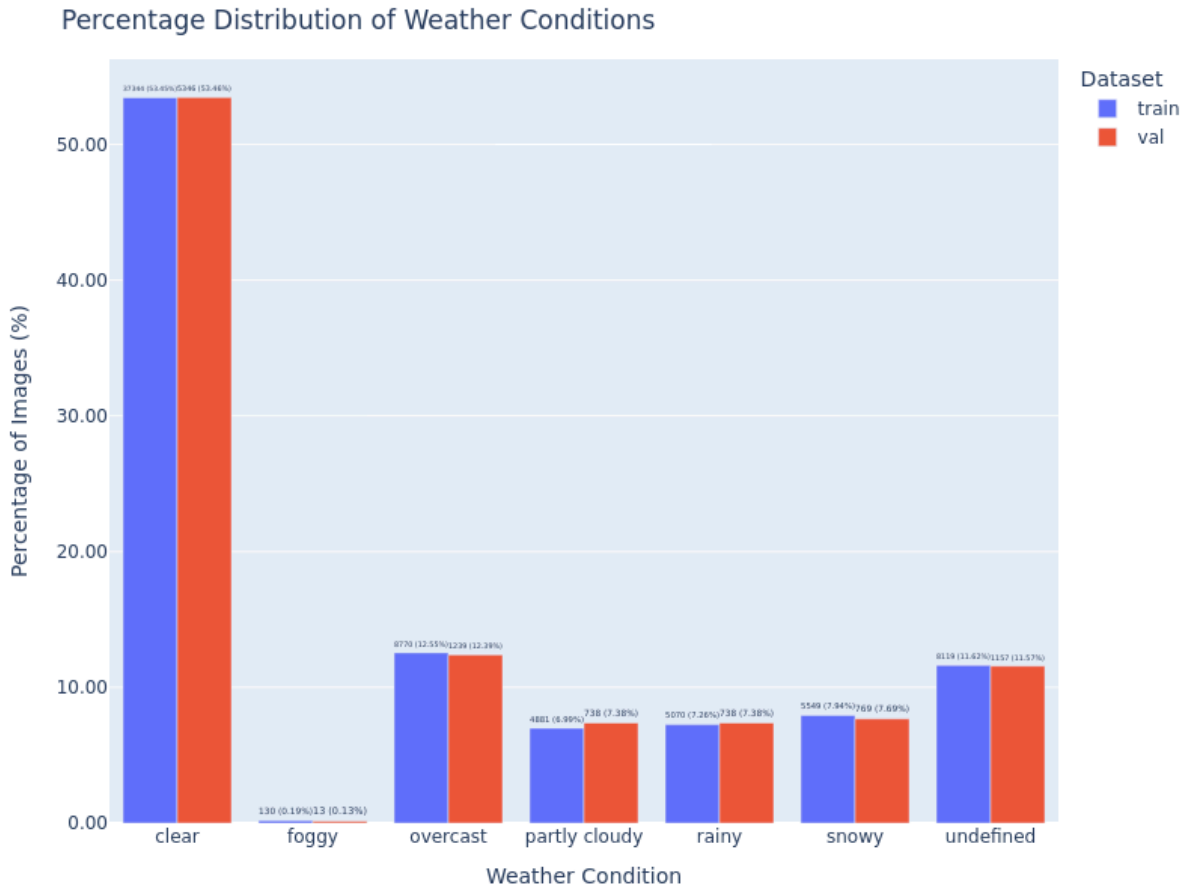
Analysis of class distribution in Train and Val dataset



Observations:

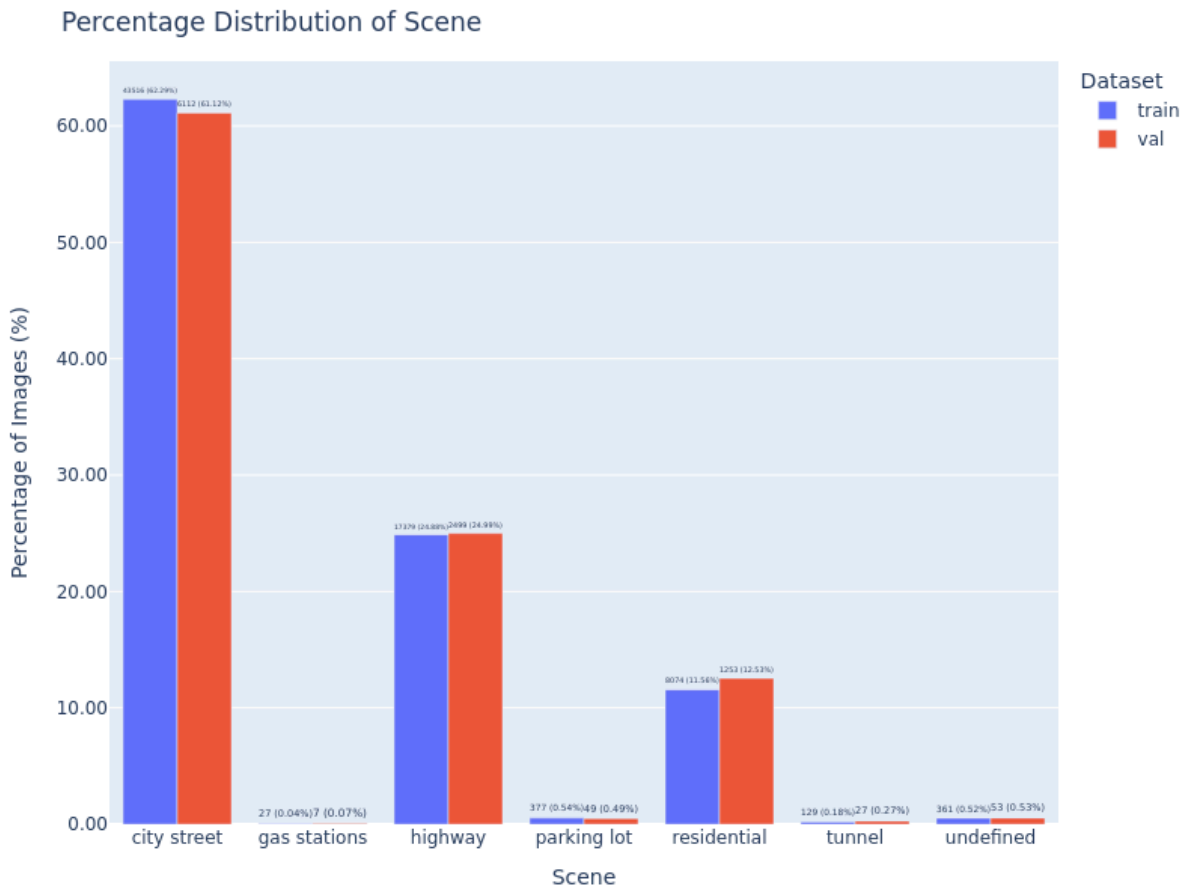
- We can see similar distribution of classes in both train and val
- ‘Car’ class accounts for **50%** of overall annotations
- Roughly **70%** of annotations are **cars, traffic signs and traffic lights**.
- Train, bicycle, bus, motorcycle combined makes less than **3%** of overall annotation in both train and val split. As the distribution of train,motorcycle, bus, bicycle, truck is significantly lower in the dataset, we can expect poor performance in these classes.
- We have around **185526** objects in the validation data.

Image attribute analysis



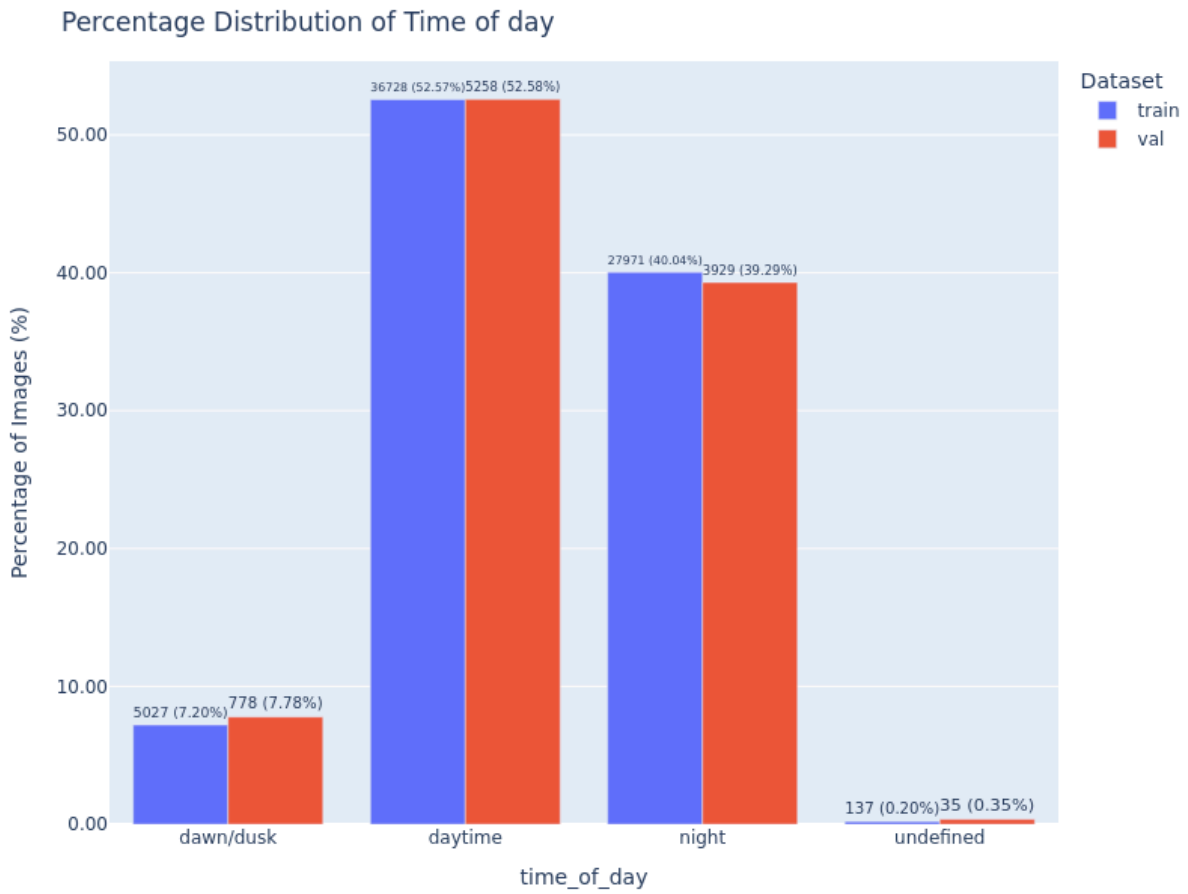
Observations:

- It is observed that weather distribution is similar in train and validation set.
- Most of the time the weather is clear and very few times the weather is foggy.
- Also we can find few images that don't have proper attributes for weather and its undefined.



Observations:

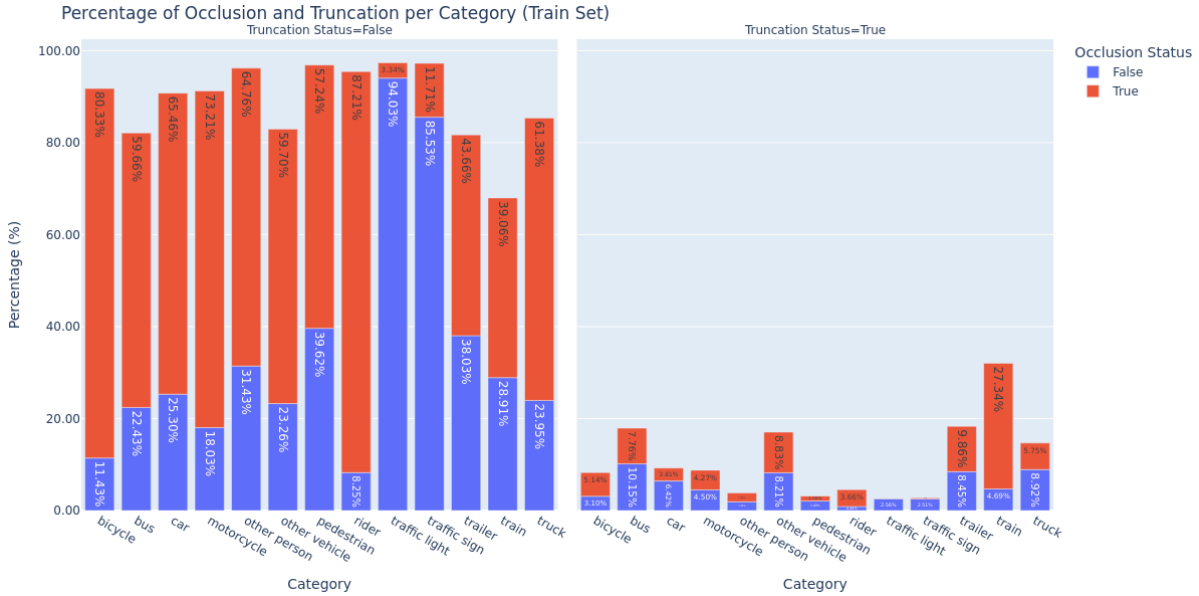
- More than **60%** of data is collected in **city streets and highways**.
- Very little data is collected near parking lots, gas stations and tunnels



Observations:

- We can find good representation of data from both daytime and night time. The data distribution is similar to the real world scenario where mostly people drive during daytime or night..

Class level attribute analysis

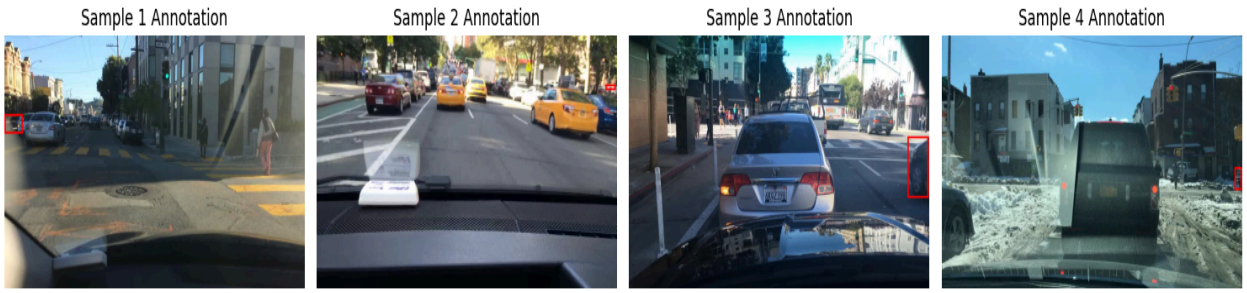


Observations:

- We can see traffic light and traffic sign objects do not suffer from occlusion as they generally appear at the top and will not be hidden by other objects
- All the other classes suffer from occlusion and notably class cars which have high frequency in the dataset appear **>60%** in occlusion which can hamper the detection performance.

Image Analysis

Sample images where class ‘car’ is truncated

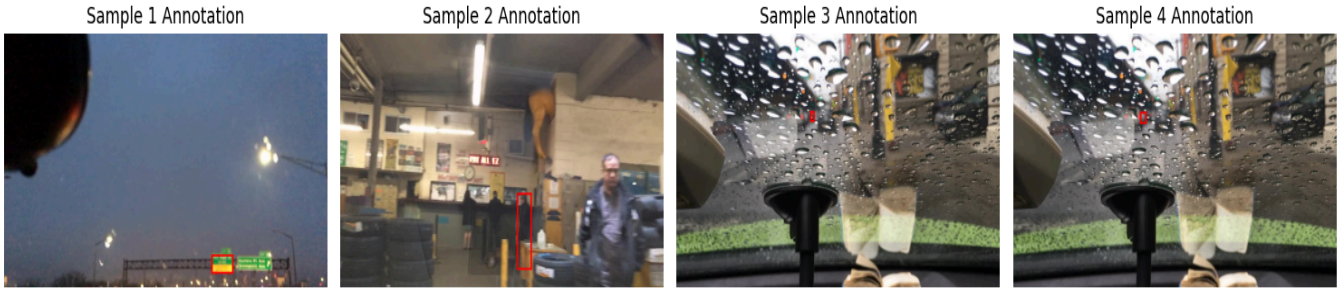


Samples from train class

Observation: We can see that ‘train’ class distribution is less in the dataset and also samples are hard even for humans to detect it.

Hard samples from traffic sign where its occluded

Outlier/ Noisy samples



In the above image, if we see sample 2,3,4 we can find the annotation looks like noise and which can affect the performance of the model. ***Interesting observation is that for these kinds of images we can find some of the image attributes as undefined.***

In the above image - Sample 1, we can clearly see there is no object visible and the model can struggle and may not converge on these kinds of examples.

Samples of Traffic light

Some samples of wrong annotations in image:

In the below image we can see there is a small annotation at car front part

In the below image, we can see entire image is annotated as one single class

In the below image some reflection on the car windshield is marked as a particular class

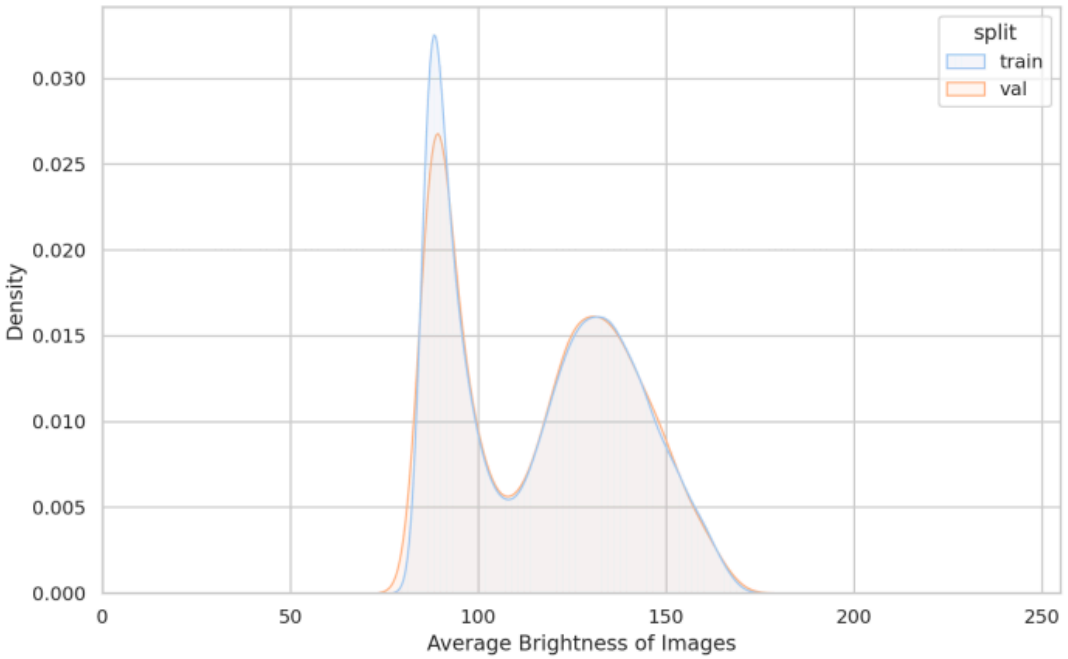
Image: 9d6673e0-664e11dc.jpg

Image: 320608e7-6da10836.jpg

1.1. General Statistics

	Train	Validation
Images	70000	10000
Classes	13	13
Classes in use	13	13
Annotations	1274792	186033
Annotations per images	18.21	18.60
Images with no annotations	147	0
Median image resolution	720x1280	720x1280
Smallest annotation	0	9
Largest annotation	918322	573682
Most annotations in an image	88	66

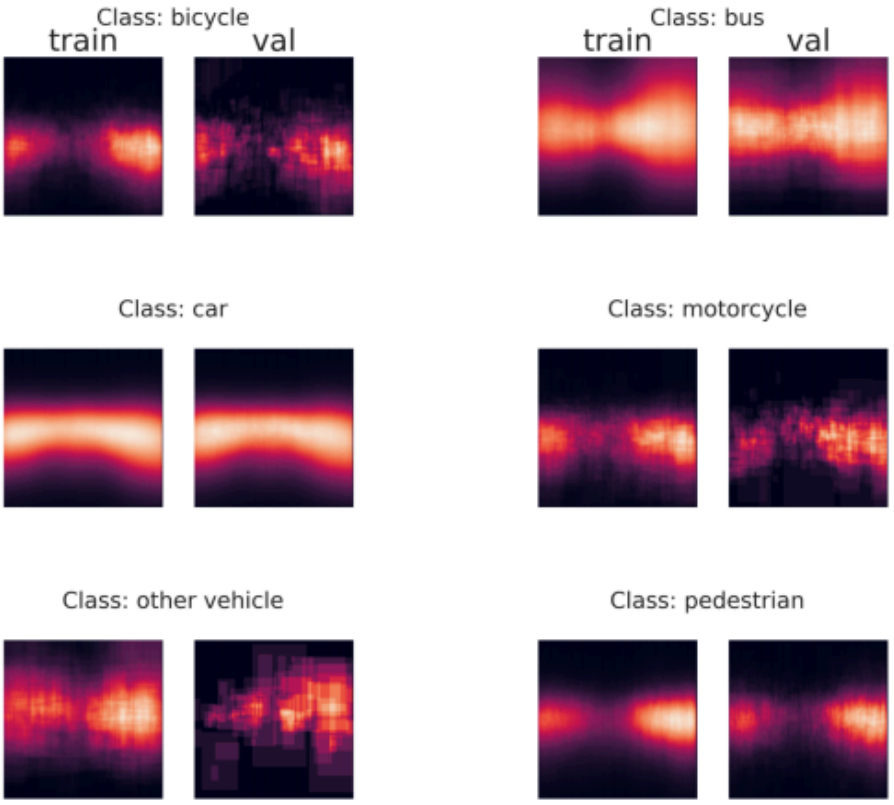
1.4. Image Brightness Distribution



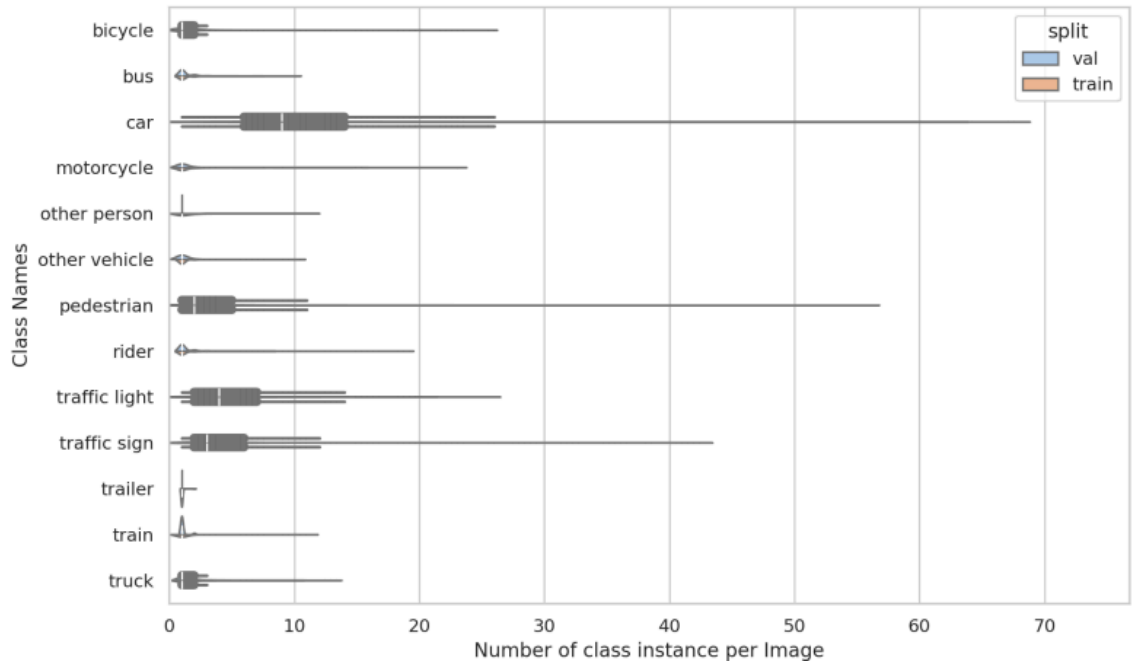
We can observe that both train and val set follow the similar brightness pattern pixel intensity between 75 - 175.

2. Object Detection Features

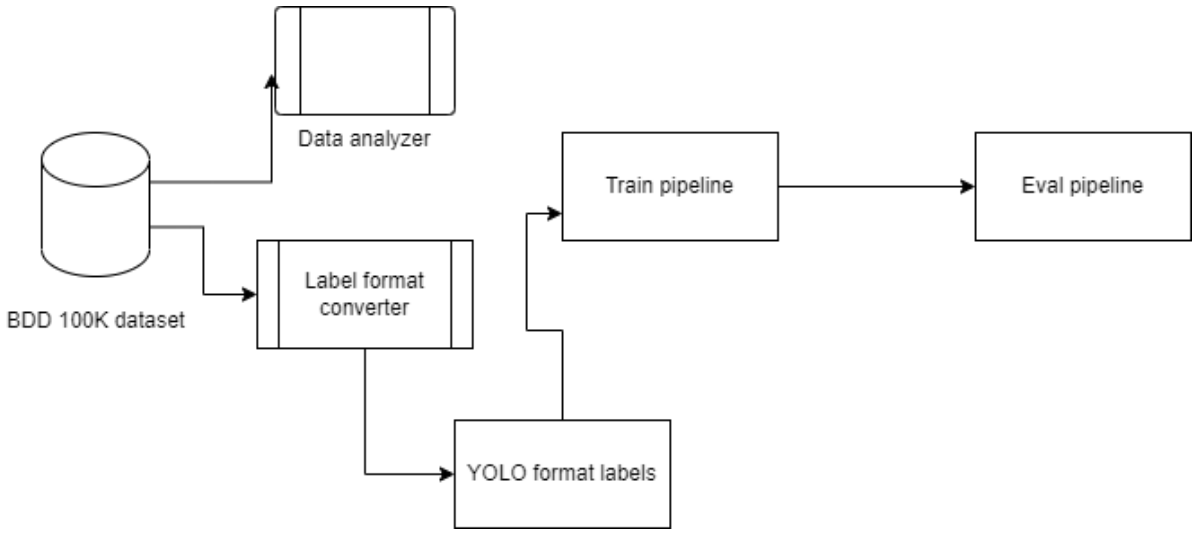
2.1. Bounding Box Density



2.6. Distribution of Class Frequency per Image



System setup:



Steps:

- BDD100K dataset is downloaded
- The labels in BDD100K will be of format list of dict where each item in the list corresponds to image - which will have image level attribute, list of annotation where each annotation will have annotation level attribute and bounding box in xyxy format.
- From the train and val json file, we flatten it into a pandas dataframe for effective data analysis to understand the data.
- Data analysis module will perform analysis on image stats, stats of overall data in train and val set and then image analysis to find patterns of image in the dataset - example : Find number of annotations which are small, occluded /truncated.
- Then label converter module which will convert the raw bdd100k data into required format for model training.
- The data loader module will handle the loading and batching of data for the training loop.
- The trainer module will take care of training and hyper param tuning.
- The evaluate module will perform inference on the best checkpoint and calculate necessary metrics.
- Then the eval analysis module will look for patterns and issues in the prediction.

Modelling:

For the detection purpose the chosen model is YOLO11s.

Brief about the model:

Key Idea: It processes an image in a single pass, directly predicting bounding boxes and class probabilities for objects within the image.

Components of YOLO:

Component	Function	Details
Backbone	Feature Extraction	Transforms image to multi scale feature maps
Neck	Feature aggregation	Enhances features by fusion
Head	Prediction	Final output

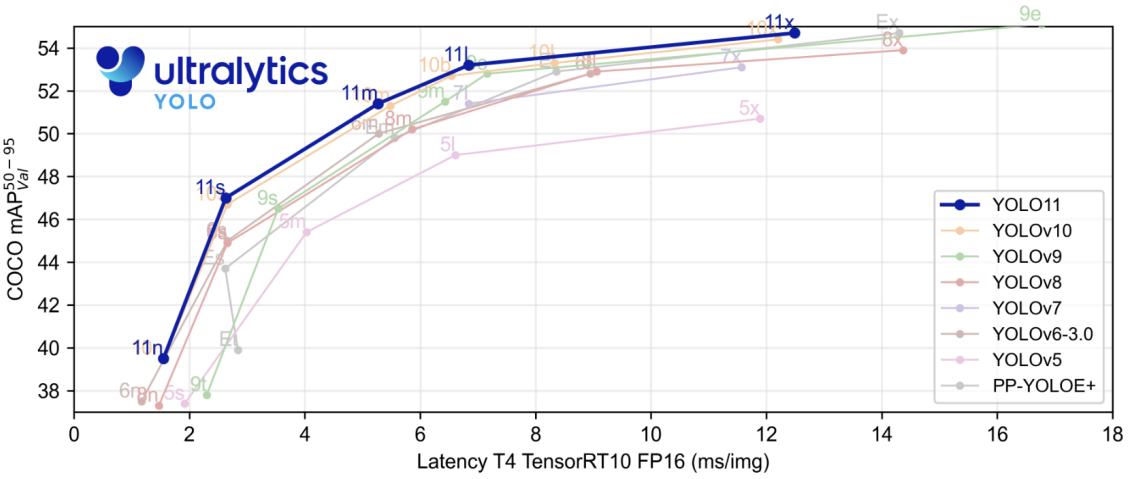
Loss function: YOLO V11 uses combination of 3 loss function for training

Loss	**Description**
Class Loss	Cross entropy loss - Loss fn for penalizing the misclassification of class
BBox Loss	IOU loss
DF Loss	DFL(distribution focal loss) is used in the bounding box regression. The general idea is to predict distributions of the box offsets instead of predicting the values directly.

Why YOLO?

Model	Size (pixels)	mAPval 50-95	Speed CPU ONNX (ms)	Speed T4 TensorRT10 (ms)	Params (M)	FLOPs (B)
YOLOv5n	640	34.3	73.6	1.06	2.6	7.7
YOLOv5s	640	43.0	120.7	1.27	9.1	24
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.2	11.2	28.6
YOLOv9n	640	38.3	NA	NA	2.0	7.7
YOLOv9s	640	46.8	NA	NA	7.2	26.7
YOLOv11n	640	39.5	56.1	1.5	2.6	6.5
YOLOv11s	640	47.0	90.0	2.5	9.4	21.5

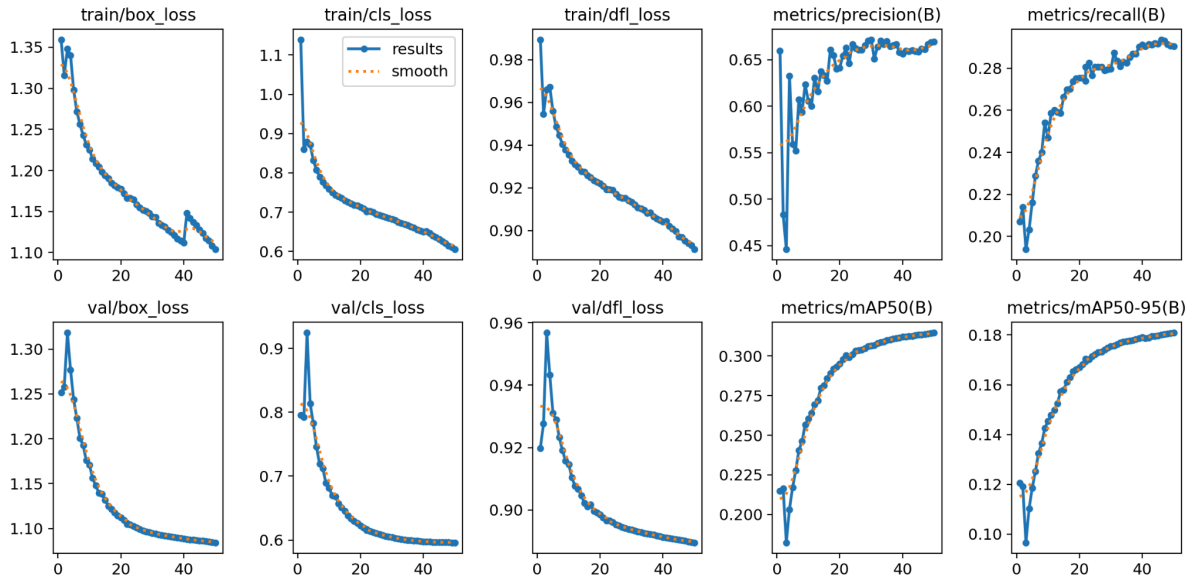
Reference : YOLOv11 Architecture Explained: Next-Level Object Detection with Enhanced Speed and Accuracy | by S Nikhileswara Rao | Medium



Based on the above image from the ultralytics which is the official maintainer of the YOLO V11 we can see there is clear improvement in MAP score and inference time.

Training:

The model is trained for upto 50 epochs at fixed image size of 416 using NVIDIA RTX A5000 24 GB VRAM on batch size of 128. Other hyper parms are kept as default values.



Sample train samples in batch

Eval Metrics:

Precision, Recall and MAP scores are used as metrics for the detection.

Overall metrics average across all classes

Metric	**Value**
Precision	0.66
Recall	0.29
MAP	0.31

```
pt: 100%|██████████| 10000/10000 [00:00<?, ?it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 1
<00:00, 12.32it/s]
                   all      10000     186033      0.662      0.294      0.314       0.18
               bicycle        592       1039      0.415      0.348      0.309       0.15
                   bus       1299       1660      0.618      0.477      0.523      0.405
                   car       9882     102837      0.747      0.638      0.689      0.446
            motorcycle        346        460      0.583      0.298      0.334      0.161
          other person          1          1          1          0          0          0
         other vehicle         70         85      0.158     0.0471     0.0438     0.0296
            pedestrian       3261      13425      0.611      0.424      0.459      0.217
                 rider        527        658      0.511      0.331      0.326      0.154
         traffic light       5651      26884       0.65      0.345       0.39      0.144
          traffic sign       8211      34724      0.674      0.403      0.457      0.237
               trailer          2          2          1          0          0          0
                 train         14         15          1          0    0.00124   0.000619
                 truck       2733       4243      0.638      0.505      0.547      0.401
Speed: 0.0ms preprocess, 0.7ms inference, 0.0ms loss, 0.5ms postprocess per image
```

Why is an overall metric not a better one?

In our experiment our overall recall is **0.294** which may look bad but the overall recall is calculated as the average of recall of individual classes. If we check our data analysis, there are classes which have less than 1% in the train set which are expected to perform badly in the val set and we may not predict anything for particular classes which are very few in number. When we take average across classes it's expected to reduce the metrics. Either looking at class wise metrics or taking weighted average should be a better option.

Eval analysis:

Once the prediction is done, few samples of predicted images are analyzed.

Few of the underperforming images where the prediction is fewer than the ground truth.

image_id	ground_truth_count	predicted_count	difference
c3a18207-504667c7.jpg	63	11	52
b5a50c2d-ccafe43a.jpg	50	8	42
c71a5c66-93c9eca0.jpg	57	15	42
b41d35f8-6cf85033.jpg	59	17	42
bc574681-412ec1a0.jpg	46	9	37

From the above, we can see that for the first image, yolo predicted 11 out of 63 ground truths which translates to 18% was predicted and > 80% was missed for the particular image.

Class Wise confidence score stats:

	category	mean_conf	min_conf	max_conf
0	bicycle	0.423982	0.250179	0.909651
1	bus	0.619618	0.250655	0.949279
2	car	0.581212	0.250000	0.949239
3	motorcycle	0.456430	0.250504	0.853361
4	other vehicle	0.395774	0.264577	0.580318
5	pedestrian	0.444944	0.250009	0.887471
6	rider	0.487212	0.252513	0.878536
7	traffic light	0.447411	0.250030	0.835715
8	traffic sign	0.509058	0.250009	0.936944
9	truck	0.567682	0.250030	0.953601

Observations:

- Though the class ‘car’ had more samples in the train set, the mean confidence of the model predicting a car is 0.58 and it's lower than the class bus.
- Traffic light, traffic sign which are next marjory class in the dataset, we can see the average confidence in predicting those two class is (0.45-0.50)

<u>Samples where model didn’t predict anything</u>

Image: c5c6fd07-9a05cafd.jpg

Image: bdb26d74-327526b3.jpg

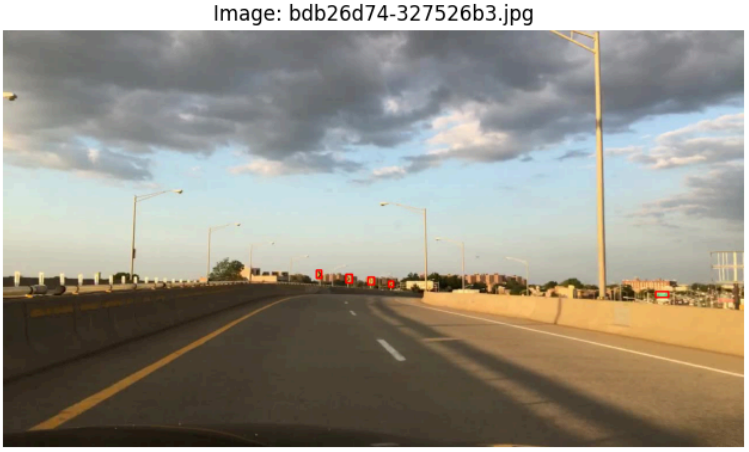
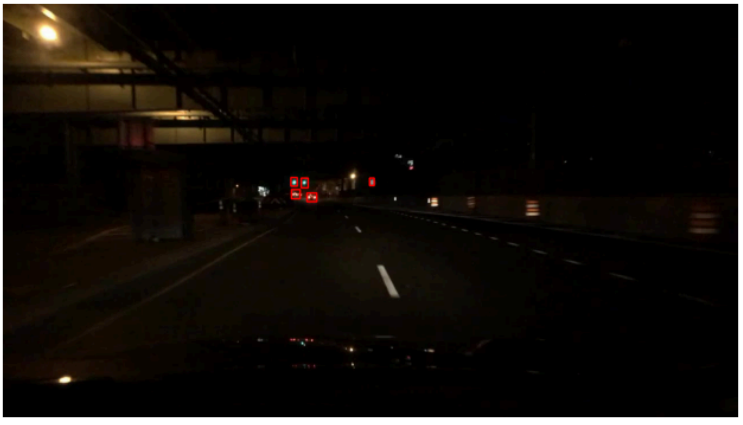
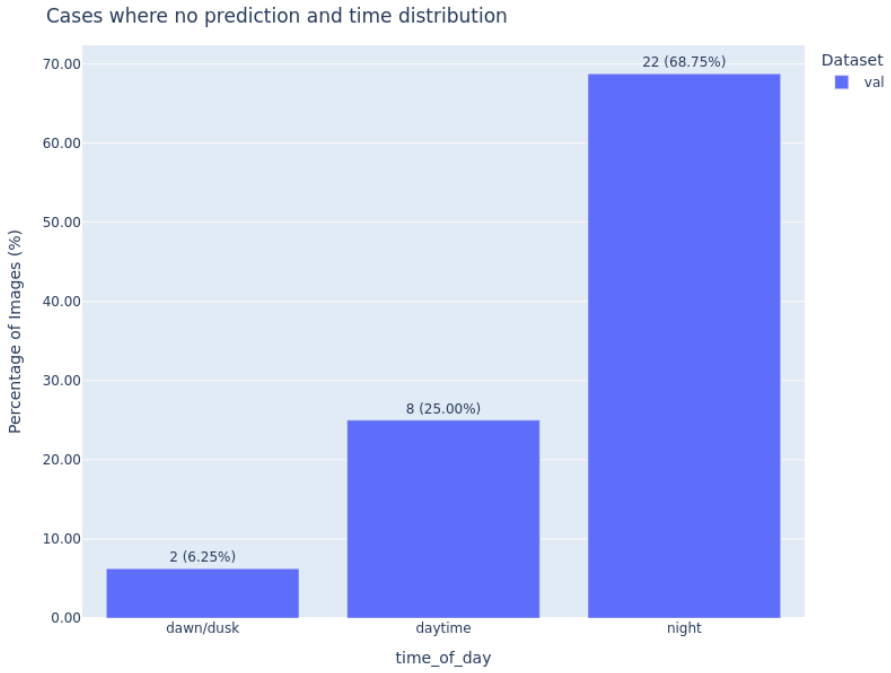


Image: c32dbcaf-4947ca34.jpg

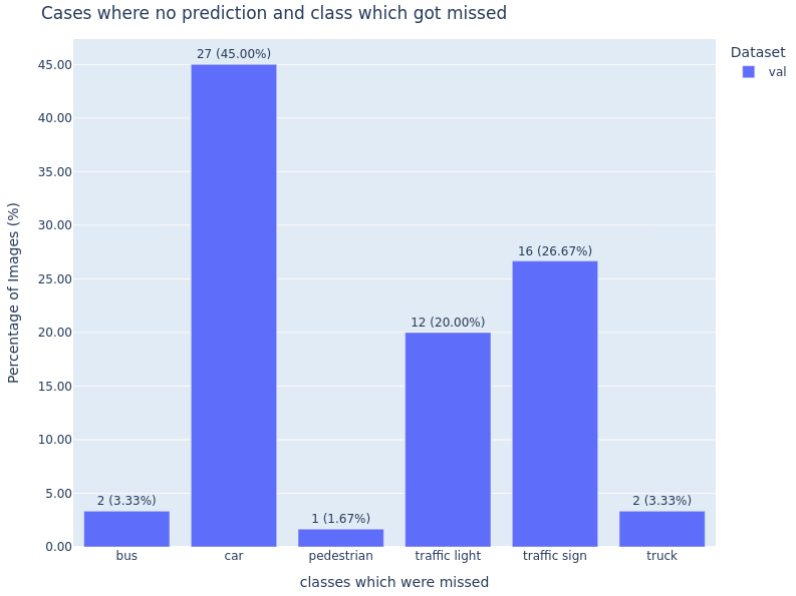


Observations:

- On eyeballing the sample images where there are no predictions the common pattern observed is mostly they are night scene images and object size is small.
- One way to improve on these kinds of cases is to add more samples which are small and generate synthetic night scene data for models to learn.



From the above graph we can observe that images which ahad zero predictions where from night time and size of objects were small.

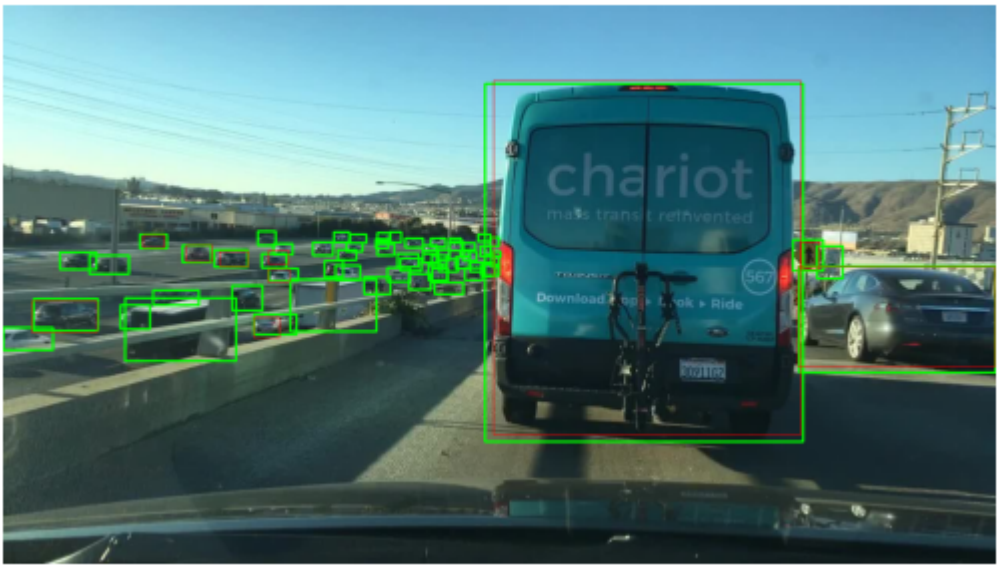


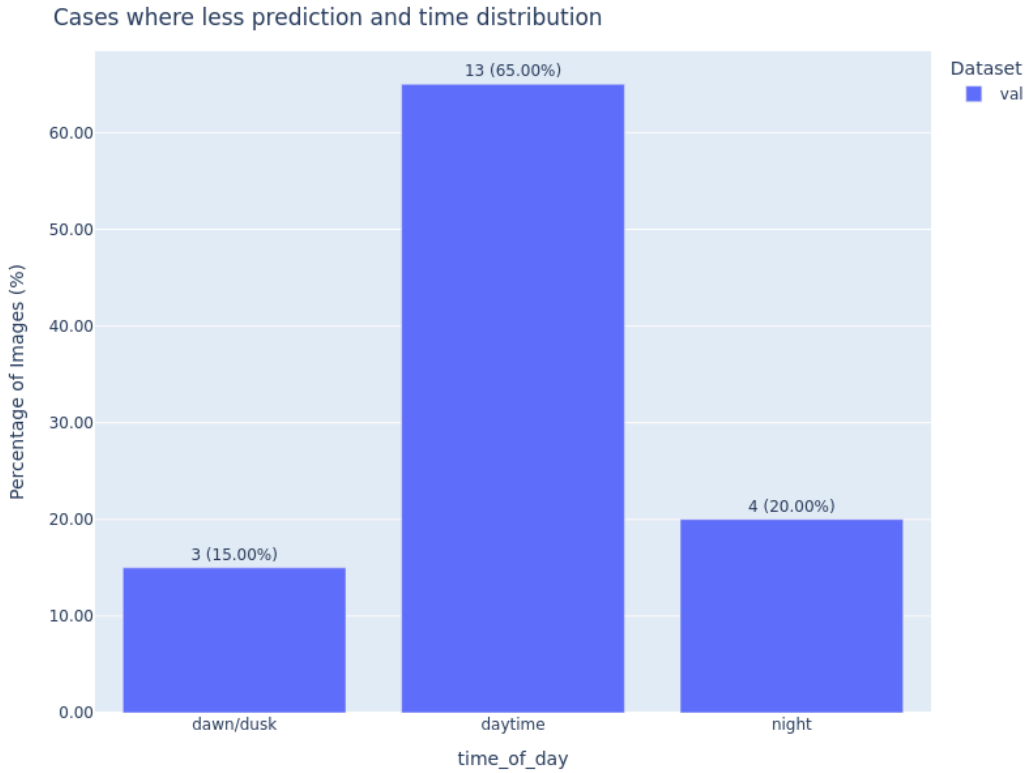
As cars, traffic lights and signs are the majority classes in our dataset, in images with no prediction, we can observe similar patterns.

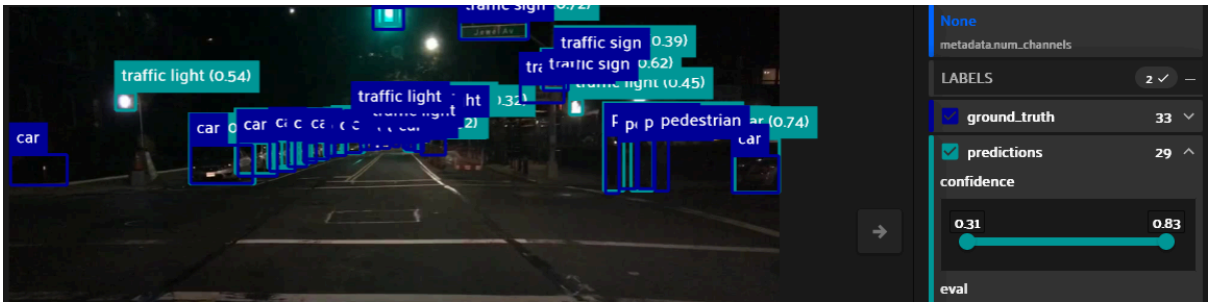
Analysis on underperforming images where the model missed to predict most of objects in image

Image: b5a50c2d-ccafe43a.jpg

Image: c3a18207-504667c7.jpg

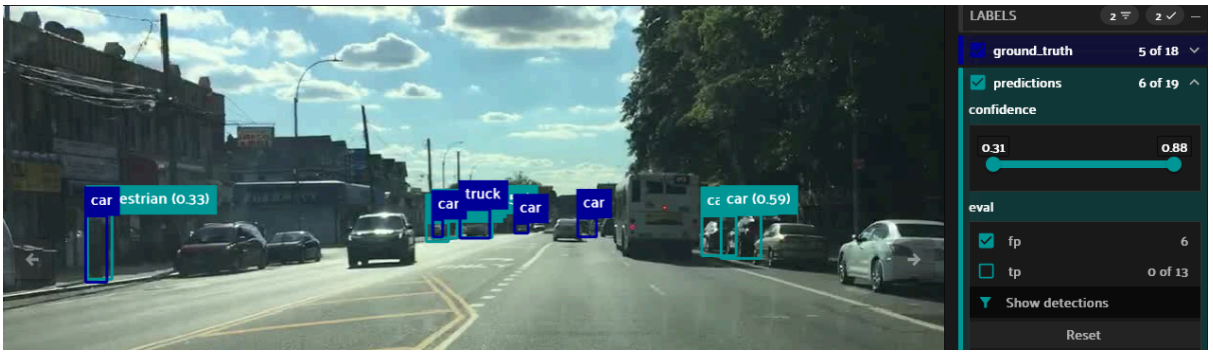






In the above image, the model has predicted a street light on the left as a traffic light which is a false positive. Also to the left of the street light, we can see a false negative where we missed to predict a car which is dark and not visible to human eyes but marked as a ground truth car.

Ground truth noise sample

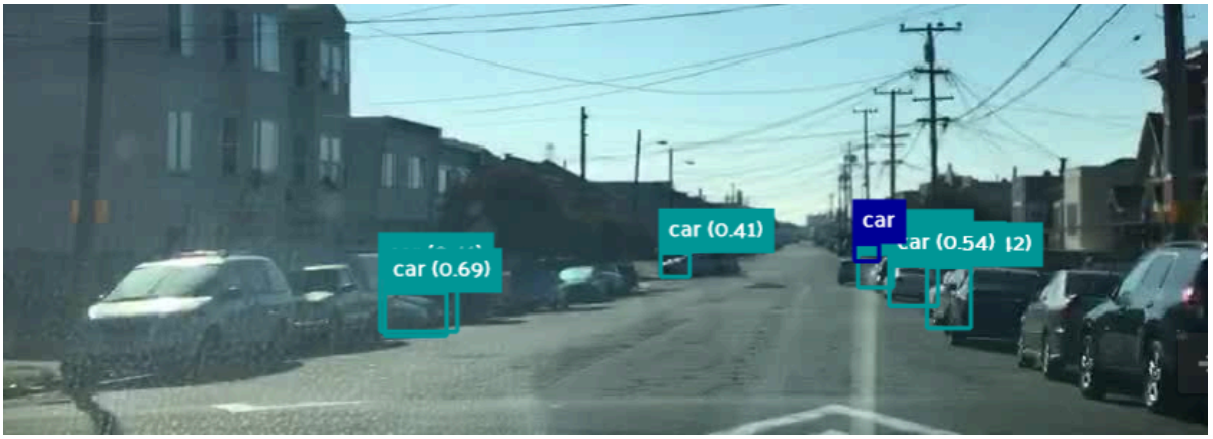


Here we can see the pedestrian is marked as a car in ground truth. In order to have better model performance, we have to review the ground truth and correct it.

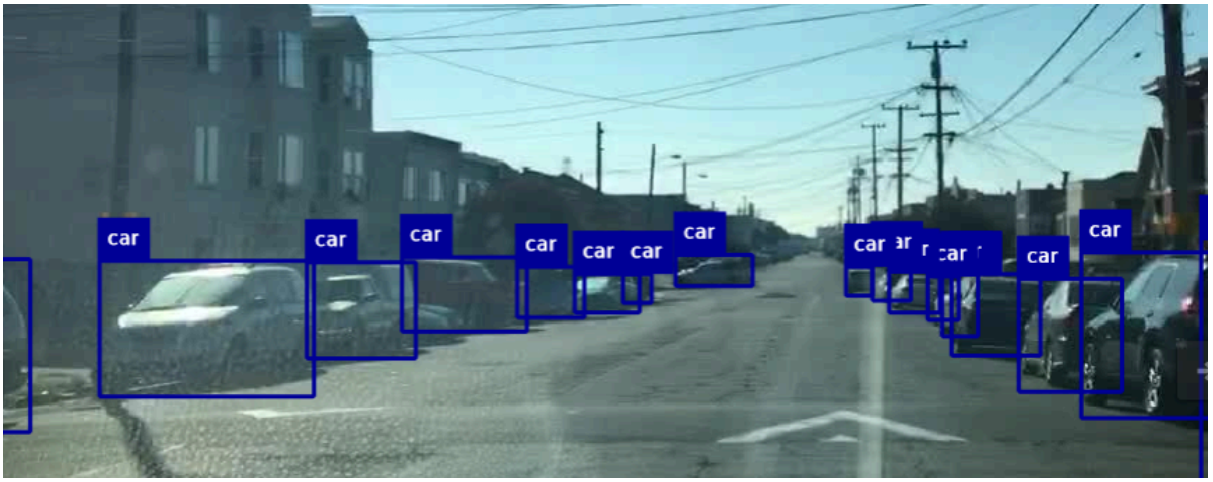
A sample of hard scenario

The model missed to detect the car which is far away and which is occluded at the right side.

Samples which were marked as FP due to improper bounding box



In the above image, all the boxes were not proper due to which it became a false positive. The corresponding ground truth for the above image is

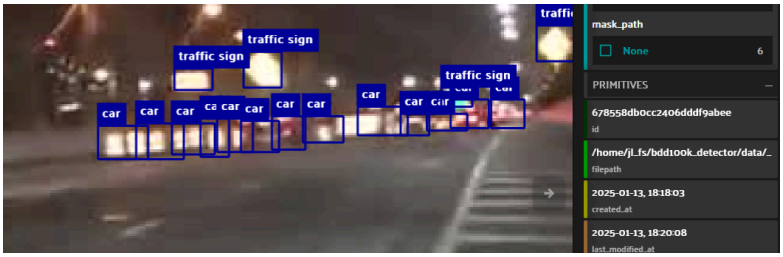


A challenging scenario where predictions went wrong

Below is the image example for challenging situation

The prediction on the above image is given as

The ground truth for that image zoomed in.



Below is a sample, where we missed on predicting a traffic sign which is in the middle of the image and not very clear.

Summary of Prediction Analysis on BDD100K Validation Set

1. **Missed Predictions on Images**:
    - Approximately 100 images had no predictions at all.
    - **Key Observations**: Most of these images were from nighttime scenarios where lighting conditions were poor, and objects were small due to their distance from the vehicle.
2. **Challenges with Crowded Scenes**:
    - In images with higher percentages of missed objects, the common issue was the presence of crowded objects.
    - Objects in these images often had small bounding box areas, making detection difficult.
3. **Noise in Ground Truth Annotations**:
    - Some ground truth annotations contained errors.
    - Example: A pedestrian was annotated as a car in one instance, leading to confusion during model evaluation.
4. **Hard Scenarios**:
    - Certain images presented challenging scenarios:
        - Objects were occluded and far away, appearing very small in the image.
        - These factors made accurate detection significantly harder for the model.
5. **Bounding Box Issues**:
    - In some cases, ground truth bounding boxes were inaccurately placed.
    - Example: Bounding boxes were not tightly aligned with objects, leading to false positives during prediction.
6. **Traffic Lights and Traffic Signs**:
    - **False Positives**:
        - Advertisement boards were sometimes detected as traffic signs.
        - A streetlight was detected as a traffic light in one instance.
    - **Missed Detections**:
        - Traffic signs appearing small in size were frequently missed.
        - Inside tunnels, tunnel lights were occasionally misclassified as traffic lights.

<u>Next Steps: (Metrics Improvements)</u>

1. Data-Centric Improvements

1. **Augment Nighttime Data**:
    - Enhance nighttime training data using augmentation techniques like brightness adjustment, contrast variation, and gamma correction to simulate diverse lighting conditions.
    - Collect additional nighttime images or use synthetic data generation for better representation of low-light scenarios.
2. **Focus on Small Object Detection**:
    - Augment training data by cropping and zooming into regions with small objects.

    - Use Mosaic or CutMix augmentation techniques to artificially increase the presence of small objects in the training set.
3. **Address Ground Truth Noise**:
    - Conduct a thorough review and cleanup of annotations, ensuring proper labeling and alignment of bounding boxes.
    - Correct misclassified examples (e.g., pedestrians labeled as a car) to improve model training quality.
4. **Balanced Class Distribution**:
    - Oversample underrepresented classes during training.
    - Use class-specific sampling to balance the dataset.

2. Model Architecture and Training Enhancements

1. **Hyperparameter tuning and Model architecture**:
    - Trying out yolo medium and large models
    - Trying different settings for hyper params like batch size, augmentation percent, learning rate, weight for each loss.
2. **Anchor Box Optimization**:
    - Add smaller anchor boxes for improved detection of small objects.
3. **Multi-Scale Training**:
    - Train the model using images of varying scales to make it robust against objects of different sizes and distances.
4. **Loss Function Customization**:
    - Use custom loss functions like **Focal Loss** to prioritize harder-to-detect small objects and address class imbalance.
    - More weightage if the model didn't predict small objects.

Deployment & serving

The model will be served as a Docker container and also can be run locally on notebooks.