Training Project Lab Report

1. Introduction

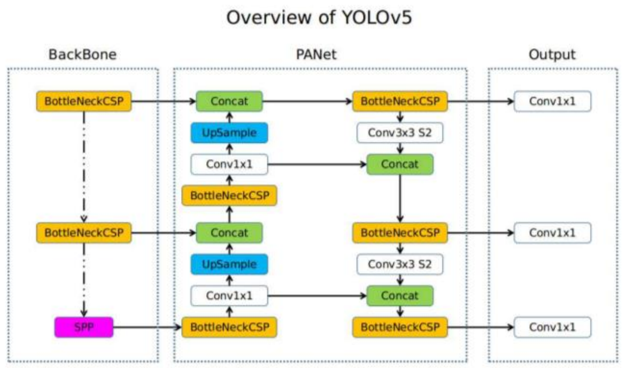
Object detection is a crucial aspect in the development of autonomous vehicles, enabling them to perceive and navigate their surroundings effectively. This project focuses on implementing object detection system using YOLO neural network-based model with KITTI dataset. The primary goal is to train a model capable of accurately identifying and localizing objects in real-world driving scenarios.

2. YOLO

You Only Look Once (YOLO) is a family of real-time object detection algorithms that gained popularity for their speed and accuracy. The YOLO algorithm works by dividing an image into a grid and making predictions for bounding boxes and class probabilities within each grid cell, enabling efficient and rapid detection of multiple objects in a single pass. And which also claims to be one of the most promising deep learning detection systems out in the wild.

For this project, we will be using **YOLOv5**, which developed by Ultralytics, builds upon the success of its predecessors.

- It introduced a more modular architecture, enabling easy customization and adaptation.
- Emphasizes simplicity, flexibility, and ease of use.
- Offers various model sizes (small, medium, large) to accommodate different resource constraints.



- Architecture

  YOLOv5 architecture is modular and contains of three main components: a backbone, a neck, and a head.

Backbone:

Utilizes a CSPNet (Cross-Stage Partial Network) as the backbone for feature extraction. Enhances feature representation and information flow through the network.

Neck:

Employs a PANet (Path Aggregation Network) as the neck to refine features at different scales. Facilitates better localization and recognition of objects.

Head:

The head of YOLOv5 predicts bounding box coordinates and class probabilities for each grid cell. This prediction is performed in a single pass through the network.

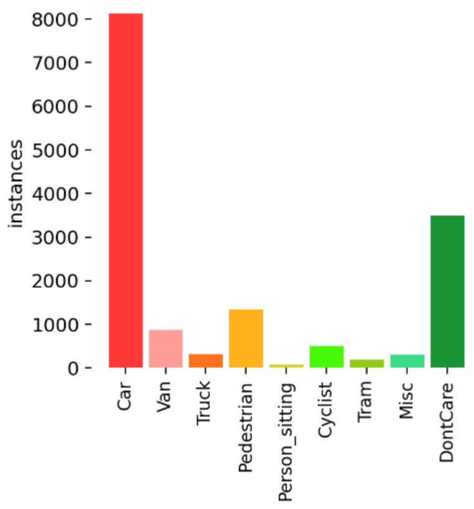
3. Dataset- KITTI

The KITTI dataset, renowned for its relevance to autonomous driving research, serves as the primary dataset for this project. They are captured from driving sessions around the city of Karlsruhe (Germany), in rural areas and on highways.
The dataset consists of 7481 training images with a resolution of 1240 x 375 pixels.

Classes notated in the datasets include:

- Car
- Van
- Truck
- Pedestrian
- Person sitting
- Cyclist
- Tram
- Misc.
- Don’t care

Classes like tram or cyclist are important for an autonomous driving system since car behavior will dramatically vary depending on the type of vehicle ahead.



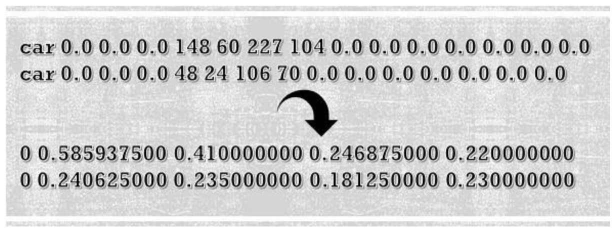
- The main classes in the dataset are: cars, pedestrians, with 20% to cars, of and cyclists with 5% to cars.

Main advantage of KITTI dataset is that it very well documented and has annotations for more classes. Also, the images are more exposure balances and with better contrast.

While the most inconvenience with this dataset is the size and aspect ratio of the images which lead to incompatibility and memory issues.

Therefore, due to memory issues, we used half of the training images, split it using traditional 80-20-20 method, Train-Validation-Test Split.

To align the dataset with YOLOv5 requirements, a pre-processing step was executed to convert the KITTI dataset's images and labels into the YOLO format.



```
car 0.0 0.0 0.0 148 60 227 104 0.0 0.0 0.0 0.0 0.0 0.0 0.0
car 0.0 0.0 0.0 48 24 106 70 0.0 0.0 0.0 0.0 0.0 0.0 0.0

0 0.585937500 0.410000000 0.246875000 0.220000000
0 0.240625000 0.235000000 0.181250000 0.230000000
```

4. Evaluation

4.1 Metrics

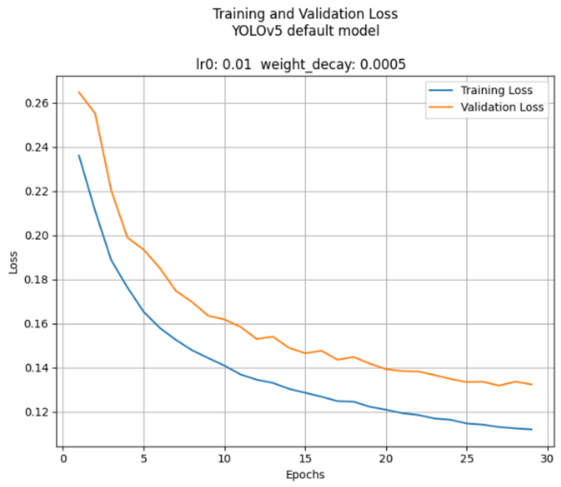
Object detection, a fundamental task in computer vision, involves not only identifying objects within an image but also precisely localizing them. The complexity of this task necessitates the use of quantitative measures to gauge the performance of detection models.

The metrics used to quantitative evaluate the results can be categorized into classification and detection metrics. And we had to consider both metrics on this model.

1. Classification – includes Precision, Recall, and F-Score (reference value which could give us how well the model performs)
2. Detection metrics: mainly kept track of Mean Average Precision (mAP).

4.2 Overfit Mitigation

When working with a custom dataset like KITTI, the challenge often lies in overcoming overfitting rather than underfitting, especially when leveraging pre-trained models. Overfitting occurs when a machine learning model becomes too tailored to the training dataset, failing to generalize effectively to unseen data.



First training with default YOLOv5s model.
Results on the test dataset.

Precision	0.57
Recall	0.33
F-Score	0.3
mAP_50	0.32
Epochs	30

4.2.1 Techniques used

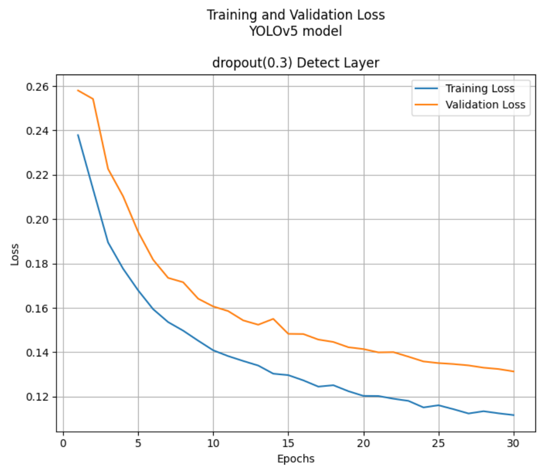
To handle the overfitting issue in our model, we adopted a three-prolonged approach:

1. **Dataset splitting:**
   Since KITTI dataset is the list of video sequences, randomly splitting the training and validation set make sures that they do not come from same video sequences.
2. **Dropout:**
   Implementing dropout, a regularization technique, during the training process helps in preventing the model from relying too heavily on specific neurons. And it commonly be used to prevent overfitting.
   Application to Detect Layer:
   The detection layer of a model is often responsible for making critical decisions based on complex features extracted from the data. Applying dropout at this stage helps to diversify the information processed by the detection layer. By randomly

excluding certain elements during training, we encourage the model to be more robust and less sensitive to the exact configuration of features present in the training data.

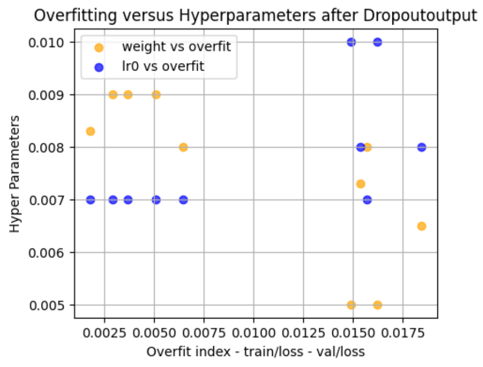
We can see that precisions are increased with 1% and there is no significant effect on the overfit.

Precision	0.4
Recall	0.397
F-Score	0.33
mAP_50	0.346
Epochs	30



3. Hyperparameter Tuning

Fine-tuning hyperparameters played a crucial role in mitigating our overfitting issue. By systematically adjusting parameters such as learning rates, weight decay, and regularization strengths, we optimized the model's performance on the validation set, striking a balance between fitting the training data well and generalizing to new, unseen data.



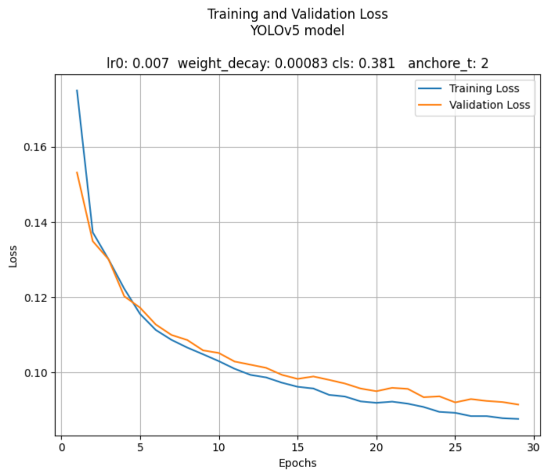
From the hyper parameters vs overfit index graph, We could observe that smaller the lr0(learning rate), and larger the weight_decay value, overfit is decreasing.

The value of learning rate is critical in determining how quickly the network converges to the optimal solution and how well it generalizes to new data.

And weight decay prevents overfitting by adding a penalty term to the loss function.

We can observe big jump in the metrics while keeping overfitting under control. Most important two metrics mAP and F-score was increased with almost with 50 percent.

Precision	0.65
Recall	0.56
F-Score	0.58
mAP_50	0.60
Epochs	30



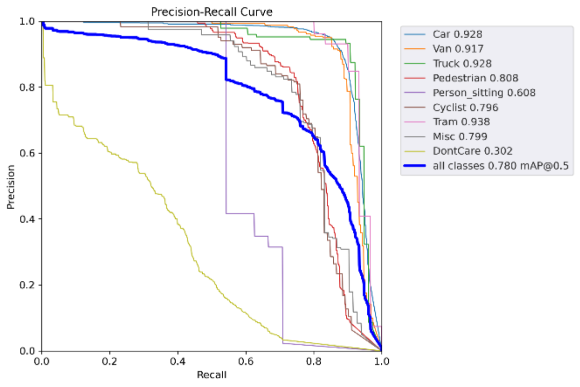
5. Model Size and Quantization

5.1 Model Size

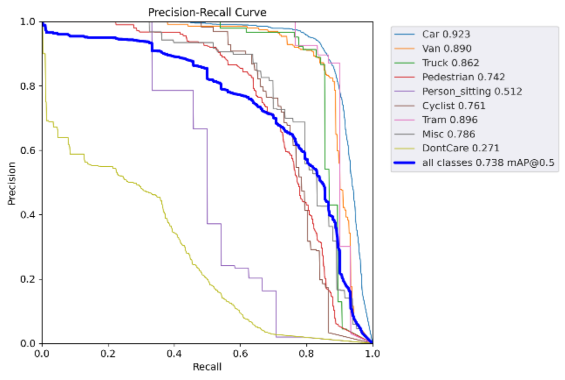
For real-world deployment, especially on resource-constrained devices, a compact model size is very crucial. Smaller models are faster to load, require less memory, and are more suitable for edge devices with limited computational capabilities.

Therefore, we made comparisons between smaller and medium models with the same implemented configurations. And tested them on same test dataset.

YOLOv5s

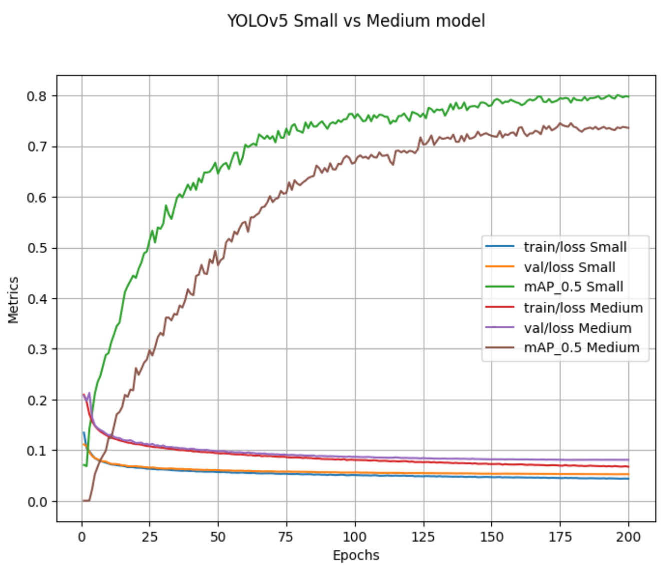


YOLOv5m



Although we anticipated a significant change in precision with the medium model, the small model performed very well on the test dataset. This suggests that our small model is capable of handling tasks comparable to those of the medium model, despite the difference in model size.

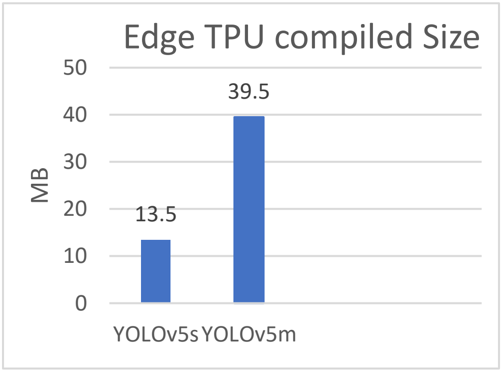
Model	YOLOv5s	YOLOv5m
Parameters	7,225,885	21,172,1273
Precision	0.853	0.818
Recall	0.757	0.698
F-Score	0.73	0.73
mAP	0.804	0.754



5.2 Quantization

Quantization is the process of reducing the precision of the model's weights and activations, typically from 32-bit floating-point numbers to lower bit-width integers (8-bit integers). It helps in compressing the model size, making it more efficient for deployment on edge devices.

We used Post-Training Quantization method which quantizes the pre-trained model after training is complete.



6. Conclusion

In summary, our study project shows that the smaller model performs as well as the medium model in terms of precision. This means we can opt for a more compact model without sacrificing accuracy in object detection. This finding has practical implications for resource-constrained environments, making smaller models a viable choice for real-world applications.