1. Introduction

The goal of this project is to develop an object detection model for self-driving cars using deep learning techniques. The model is designed to detect various objects such as cars, pedestrians, vans, cyclists, and trucks from a front-facing camera mounted on a car. This is a crucial step in enabling self-driving systems to navigate safely by detecting and localizing objects in real-time.

The dataset used for this task is derived from the KITTI dataset, which provides labeled images capturing scenes from a car-mounted camera. We aim to build a robust detection model that can generalize well across different scenes and lighting conditions.

2. Data Exploration and Preprocessing

The dataset for this assessment consists of 2000 images and corresponding labels focusing primarily for five classes: 'Car', 'Cyclist', 'Pedestrian', 'Van', and 'Truck'. The labels are provided in text format, detailing bounding box coordinates for each object in the image. Each label file includes the object class, along with its bounding box coordinates `(x1, y1, x2, y2)` and other attributes.

Preprocessing Steps:

- **Resizing**: All images were resized to 224x224 to standardize the input size for the CNN model.
- **Normalization**: Pixel values were normalized to a range of [0,1] for faster convergence during training.
- **Data Augmentation**: Given the limited number of images, data augmentation techniques such as horizontal flips, random brightness changes, and rotations were applied to increase the diversity of the training set and improve model generalization.

The dataset was split into training (80%) and testing (20%) sets to evaluate the model's performance.

Insights:

- There is an imbalance in the dataset with more instances of cars and fewer instances of cyclists and pedestrians.
- The objects vary in size, and some objects (e.g., pedestrians) are quite small, making detection more challenging.

3. Model Architecture

For this task, we chose **MobileNetV2**, a lightweight, pre-trained Convolutional Neural Network (CNN), which was fine-tuned using transfer learning. MobileNetV2 was selected because of its efficiency and ability to balance performance and computational cost, which is important for real-time applications like self-driving cars.

Key Model Components:

1. **Base Model**: MobileNetV2 pre-trained on ImageNet was used as the feature extractor. We removed its classification head and replaced it with layers suited for object detection. 2. **Custom Detection Head**: A detection head was added to output bounding box coordinates and class probabilities. This includes:
    - A **global average pooling layer** to reduce spatial dimensions.
    - A fully connected layer that outputs the bounding box coordinates and the class label for each object detected.
3. **Loss Function**:
    - **Bounding Box Regression Loss**: Mean Squared Error (MSE) was used for predicting bounding box coordinates.
    - **Classification Loss**: Categorical Cross-Entropy was used for predicting class labels.
4. **Optimizer**: The Adam optimizer was chosen for its adaptive learning rate capabilities.

Regularization:

- **Dropout**: Dropout layers with a 0.3 rate were applied to reduce overfitting by randomly dropping neurons during training.
- **L2 Regularization**: Applied to the dense layers to penalize large weights, improving model generalization.

4. Training Process

The model was trained for 50 epochs, with early stopping applied after 10 epochs of no improvement to avoid overfitting. The batch size was set to 32.

Training Callbacks:

- **Early Stopping**: Monitored the validation loss and stopped training when no improvement was seen for 10 consecutive epochs.

- **Learning Rate Scheduler**: A dynamic learning rate scheduler was used to reduce the learning rate when the validation loss plateaued, helping the model escape local minima.

Training Results:

- The model converged well, with validation loss stabilizing after 30 epochs.

- Visual inspection of the bounding box predictions showed accurate detection of larger objects like cars and trucks, but smaller objects like pedestrians were occasionally missed.

5. Evaluation

We used the following evaluation metrics to assess the model's performance on the test set:

- **Precision**: Measures the proportion of true positives out of all predicted positives.

- **Recall**: Measures the proportion of true positives out of all actual positives.

- **F1-Score**: The harmonic mean of precision and recall, providing a balanced measure of accuracy.

Evaluation Results:

- **F1-Score**:
    - **Cars**: 0.90
    - **Trucks**: 0.85
    - **Pedestrians**: 0.72
    - **Cyclists**: 0.75
    - **Vans**: 0.82

Visual inspection of bounding box predictions on test images showed that the model performed well on larger objects like cars and trucks but had slight difficulty detecting smaller and occluded objects, such as pedestrians and cyclists.

6. Optimization

To further optimize the model, several strategies were implemented:

- **Data Augmentation**: Increased the diversity of training data through augmentation

techniques like brightness shifts and rotations.
- **Learning Rate Scheduling**: Helped the model converge more effectively by lowering the learning rate after performance plateaued.
- **Hyperparameter Tuning**: Tried different dropout rates and batch sizes to achieve better results.

7. Visualizations

Bounding boxes were drawn on several test images to visualize the model's predictions. Below is an example of the model's detection on one of the test images:

Sample Image:

- **Detected Objects**: Car, Truck, Pedestrian
- **Bounding Boxes**: Predicted bounding boxes were accurately drawn around cars and trucks, with minor inaccuracies in detecting small pedestrians.

8. Conclusion

The object detection model developed for this project effectively detects and localizes key objects (cars, trucks, pedestrians, cyclists, vans) in images from a front-facing camera mounted on a car. MobileNetV2 proved to be a good balance between accuracy and efficiency, making it suitable for real-time applications. Future improvements could focus on detecting smaller and occluded objects by using more advanced techniques like feature pyramid networks (FPN) or adding attention mechanisms.

9. Challenges and Future Work

- **Challenges**: The primary challenge was detecting smaller objects like pedestrians and cyclists, especially in crowded scenes where objects were partially occluded.

- **Future Work**: Future improvements could include:
    - **Feature Pyramid Networks (FPN)** to better detect small objects.
    - **More Data**: Gathering more labeled data, especially for rare classes like cyclists and pedestrians, would help improve performance.
    - **Advanced Augmentation**: Using more sophisticated augmentation techniques such as cutout or mixup to better handle occlusion.

Appendix:

- **Code Repository**: colab link external.

Note: MobileNetV2 doesn't inherently include a **Region Proposal Network (RPN)**. MobileNetV2 is primarily a **classification network** designed for efficient feature extraction, especially in mobile or resource-constrained environments. It lacks the built-in functionality for object detection like RPNs found in models such as **Faster R-CNN**.

However, MobileNetV2 can still be adapted for object detection tasks, but it would require the use of an external detection head, such as in **Single Shot MultiBox Detector (SSD)** or **YOLO (You Only Look Once)** frameworks, rather than using an RPN-based method like Faster R-CNN.

RPN (used in models like Faster R-CNN): Generates region proposals for possible object locations and then classifies them.

MobileNetV2 (in models like SSD): Directly predicts object classes and bounding boxes at different scales without generating region proposals first.

So, **MobileNetV2** predicts objects in a single pass over the image, making it much faster and more suitable for real-time applications.