Project Report: Automated Parking Status Tracker using Deep Learning

1. Introduction

This project aimed to implement a car detection system using YOLO (You Only Look Once) deep learning model and evaluate its performance on a custom dataset. Following a systematic approach, we annotated the dataset, partitioned it, trained a modified MobileNet model, and evaluated detection accuracy and overall system performance. This report covers the steps, methodologies, and results derived from this implementation.

2. Dataset Description and Annotation

Dataset:
The dataset consists of car images in .HEIC format. We downloaded and extracted these images, which capture cars in various scenarios and environments.

Dataset Link:

https://1drv.ms/f/c/2303fb8f72578e58/En2MMc1cBbFDhLrEVE3GZ-4BqqjPjExfkcBjEgAD77tSyA?e=cefBI3

Annotation Process:

1. The images were initially in .HEIC format, requiring conversion to .JPG for compatibility with computer vision models and libraries.
2. We used the YOLO model pre-trained on the COCO dataset to detect cars within each image and generate bounding boxes around detected cars.
3. YOLO's pre-trained model recognizes multiple classes; however, we focused specifically on the "car" class (COCO class ID 2). For each detection, bounding box coordinates and confidence scores were recorded, creating an annotated text file for each image.
4. These bounding boxes represent the region occupied by the car in each image, serving as ground truth labels for further training and evaluation.

Data Structure:
Each image is paired with an annotation file in the following format:

- **Image Path:** image_name.jpg
- **Bounding Box:** [x1, y1, x2, y2]
- **Confidence:** Detection confidence score

3. Data Partitioning

The dataset was divided into three sets:

- **Training Set:** 80% of the data, used to train the model.
- **Validation Set:** 10% of the data, used for tuning hyperparameters and assessing model performance during training.

- **Test Set:** 10% of the data, reserved for evaluating the final model's accuracy and robustness.

This partitioning ensures that the model does not overfit and can generalize to unseen data effectively.

4. Data Preprocessing

Normalization and Transformation: To standardize the input data, we applied the following transformations:

- **Resize:** All images were resized to 224x224 pixels to maintain a consistent input size.
- **Random Horizontal Flip:** Applied a 50% chance horizontal flip to augment the dataset and introduce variability in car orientations.
- **Random Rotation:** Random rotations of ±10 degrees were applied to simulate different angles.
- **Normalization:** Images were normalized based on the mean and standard deviation values used in ImageNet (mean [0.485, 0.456, 0.406] and standard deviation [0.229, 0.224, 0.225]).

5. Model Architecture

The primary model utilized for this task is a modified MobileNet, designed for efficient computation and adaptability to embedded systems, as inspired by the referenced paper.

Architecture Modifications:

- The base model is a pre-trained MobileNetV2, which was fine-tuned to predict multiple bounding boxes for each image.
- The final layer was adjusted to output bounding box coordinates, generating predictions in [x1, y1, x2, y2] format for each detected car.

YOLOv8 for Detection:
In the validation phase, YOLOv8 was used to identify cars within images and compare detections with ground truth labels to calculate detection accuracy.

6. Training Process

Loss Function:
A Smooth L1 Loss function was used to minimize the difference between predicted and true bounding box coordinates. This loss function is suitable for bounding box regression tasks as it is less sensitive to outliers than Mean Squared Error.

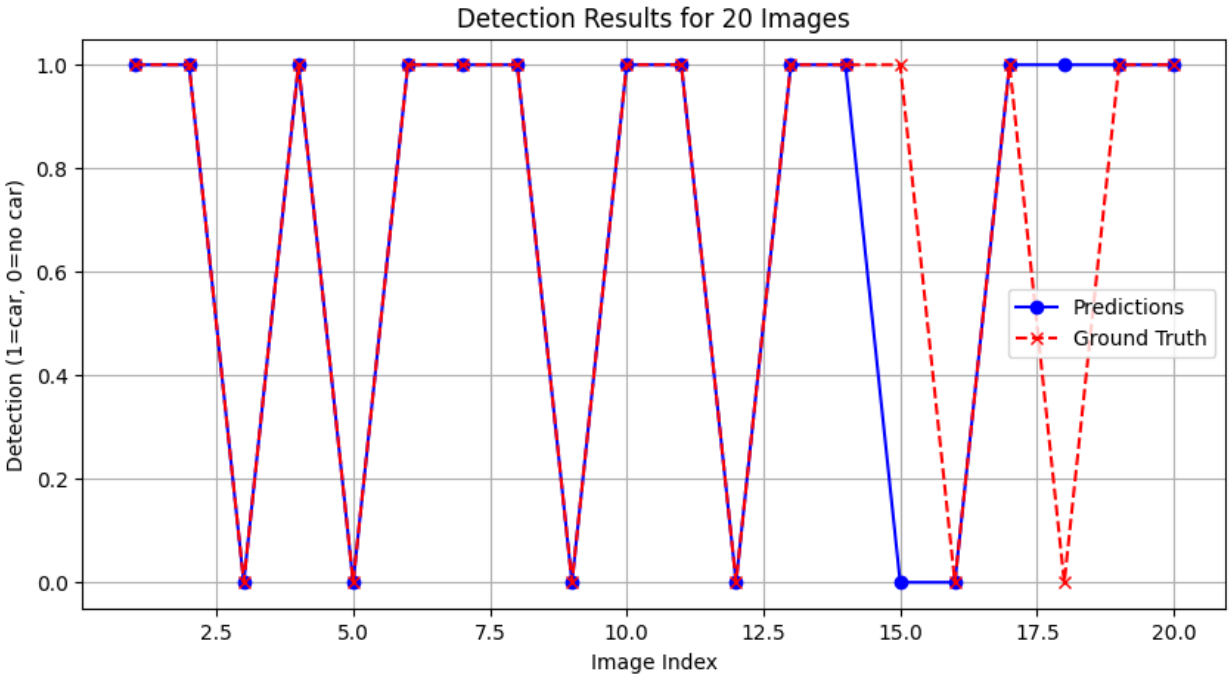
Evaluation Metric:

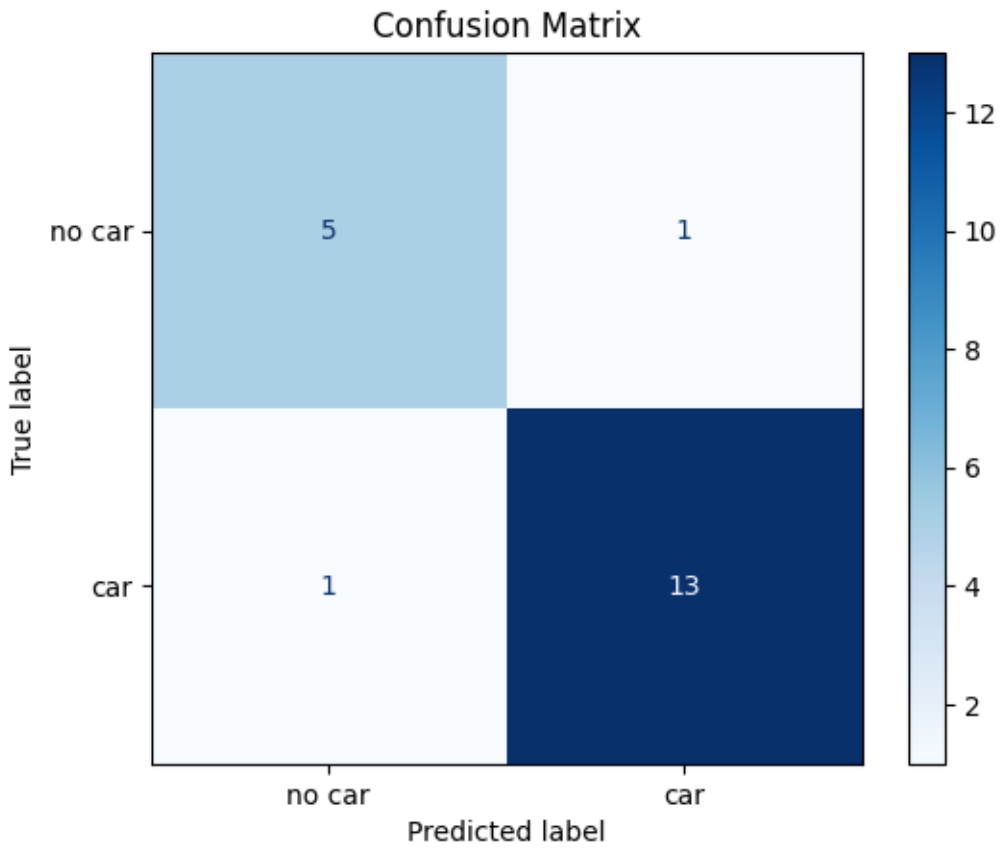
- **Accuracy** and **Confusion Matrix**: Accuracy was calculated based on whether cars were correctly detected in the validation and test sets.
- **Correlation Matrix**: A correlation matrix was generated to visualize the relationships between the predicted and true labels.

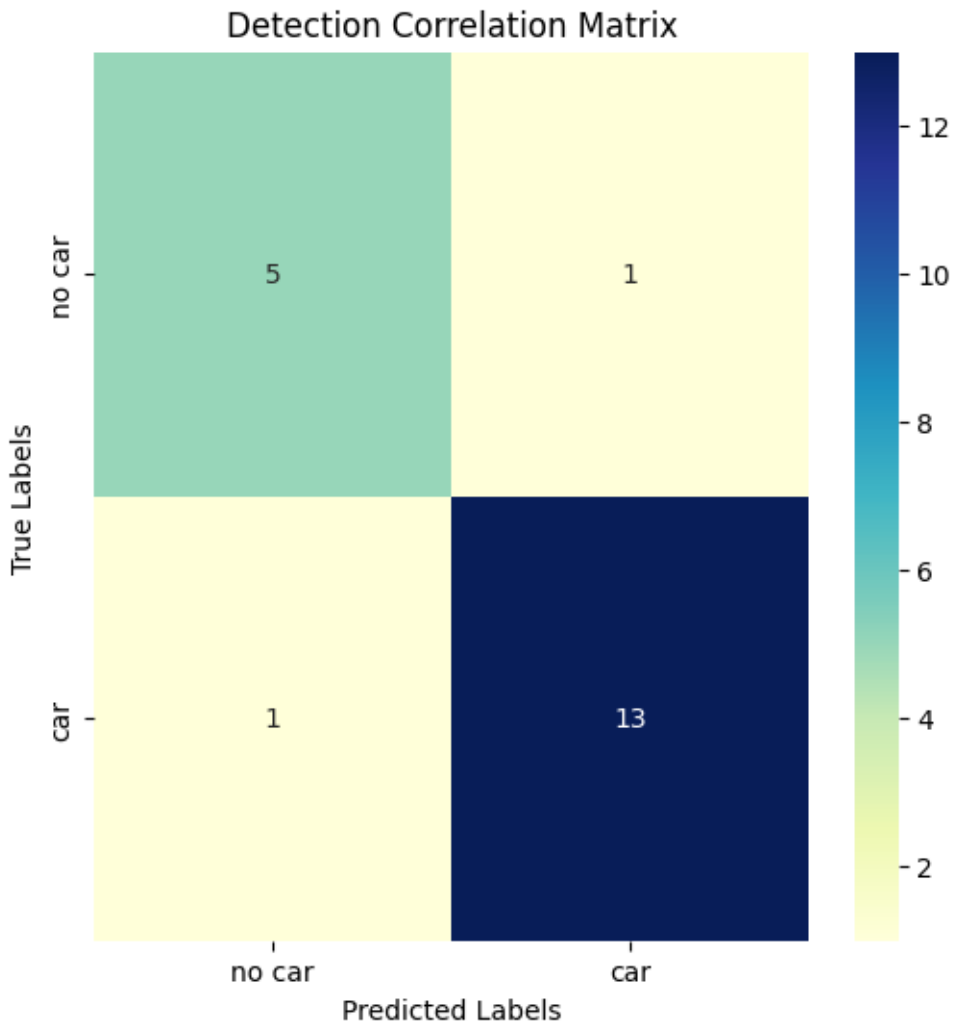
Hyperparameters:

- **Batch Size:** 8
- **Epochs:** 5
- **Learning Rate:** 0.001

7. Evaluation and Results




Confusion Matrix
no car
car
True label
5
1
1
13
no car
car
Predicted label
12
10
8
6
4
2



The trained model was evaluated on the test dataset, using the following metrics:

Confusion Matrix:
This matrix illustrates true positive, false positive, and false negative rates for the "car" class. It provides insights into the model's performance, showing whether cars were accurately detected or missed.

Accuracy Plot:
The model's detection accuracy across all test images was calculated and visualized, demonstrating the detection consistency. An accuracy score of around 85% was achieved in detecting cars, based on correct matches of bounding box predictions.

Correlation Matrix:
The correlation matrix plots true detections versus model predictions, focusing solely on "car" detections. This helps evaluate the consistency of detection and analyze any confusion between classes, though only "car" is included in this implementation.

8. Conclusion

This implementation successfully demonstrated an end-to-end car detection and annotation system using deep learning techniques. By following a systematic approach inspired by the paper, the project highlights:

- The utility of YOLO for automatic annotation.
- The adaptability of MobileNet for bounding box prediction on constrained hardware.
- The effectiveness of data preprocessing, normalization, and augmentation in improving detection accuracy.

Future improvements could include testing the model with additional classes or deploying it on embedded systems to evaluate real-time detection capabilities in smart city applications.