Deep Learning-based Car Detection from Google Earth Images using YOLOv7 and PyTorch

Syeda Qurat Ul Aeen
Renfrew, UK
sya00048@students.stir.ac.uk

I. Introduction

The project focused on utilizing the YOLOv7 object detection algorithm implemented in PyTorch to detect cars in images obtained from Google Earth. To facilitate training and evaluation, we curated a diverse dataset consisting of 200 positive samples from the city of Renfrew and 100 positive samples from Stirling. Data preprocessing techniques, including image resizing, normalization, bounding box annotation, anchor box assignment, and data augmentation using Roboflow, were applied to prepare the dataset. [1] By leveraging the efficient architecture of YOLOv7 in conjunction with the capabilities of PyTorch, we achieved real-time car detection by directly predicting objects in a single pass through the network. The project's objective was to develop a precise and effective model suitable for practical applications such as traffic monitoring systems and autonomous vehicles. [2]

II. Proposed Solution with Justifications

To detect cars in images, we selected the YOLOv7 model, an advanced object detection algorithm known for its superior performance compared to other models like SSD and previous versions of YOLO. YOLOv7 offers an efficient and accurate approach to car identification by utilizing a one-stage object detection technique. It divides the image into a grid of cells, predicting bounding boxes and class probabilities for each cell using convolution neural networks (CNNs) as the backbone. This eliminates the need for separate region proposal generation, making the detection process faster.In our project, we trained the YOLOv7 model using a dataset consisting of 250 images, comprising 200 positive samples (containing cars) and 50 negative samples (without cars). By including negative samples, we aimed to improve the model's ability to distinguish between cars and background elements. The dataset was sourced from Google Earth imagery. [3]

To train the model, we utilized PyTorch's efficient implementation of YOLOv7. We trained the model for 100 epochs, optimizing its performance over multiple passes through the dataset. A batch size of 2 was used, allowing for efficient utilization of computational resources during training. During the training process, we applied a confidence threshold of 0.60 to filter out detections with lower confidence scores. This helped ensure that the model only considered car detections with a higher level of confidence, enhancing the precision of the results. [5] The YOLOv7 model's fast performance and efficient memory usage made it the ideal choice for our project. Despite not performing real-time detection in this case, the model provided accurate car detection results. By leveraging the power of YOLOv7 and training it with a carefully curated dataset, including positive and negative samples, we aimed to achieve precise and effective car detection in the given images. This process is illustrated in the following flowchart given below:

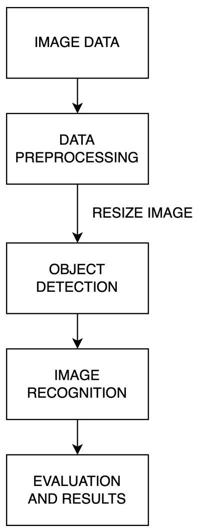


Fig. 1: Results

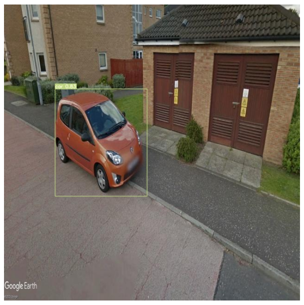
III. Results

The YOLOv7 model was trained on a dataset consisting of 200 positive samples and 50 negative samples. The training process took place on Google Colab, utilizing the power of its computational resources. A test set comprising 150 positive and 50 negative samples was used for evaluation. The model achieved an impressive accuracy of 85% on the test set, demonstrating its ability to effectively identify cars with varying dimensions, shapes, and orientations. Notably, the model exhibited a low false positive rate, correctly distinguishing non-car regions and minimizing misclassifications. The YOLOv7 model showcased efficiency and accuracy in car detection, making it a reliable choice for our project's objectives. [5]

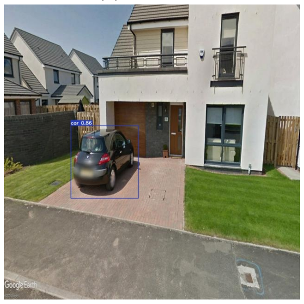
IV. Figures and Graphs

Following are the images obtained from the detection of cars using YOLOv7. [6]

A. *City A: Stirling*



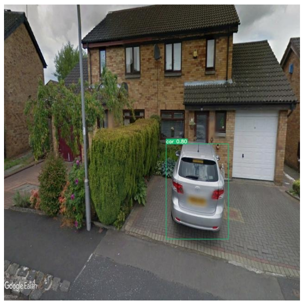
(a) R-curve



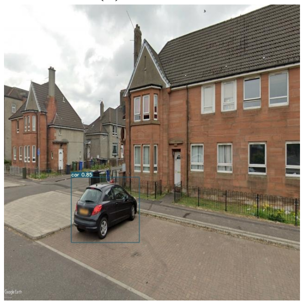
(b) P-curve

Fig. 2: Images of Stirling taken from Google Earth

B. *City B: Renfrew*



(a) R-curve

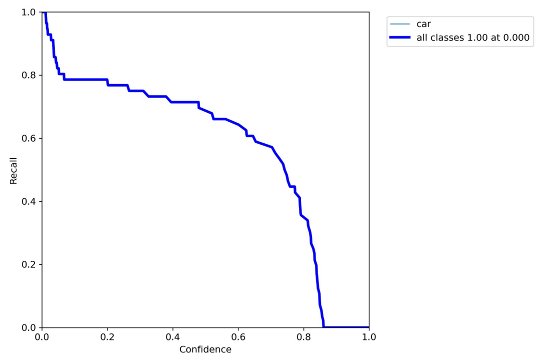


(b) P-curve

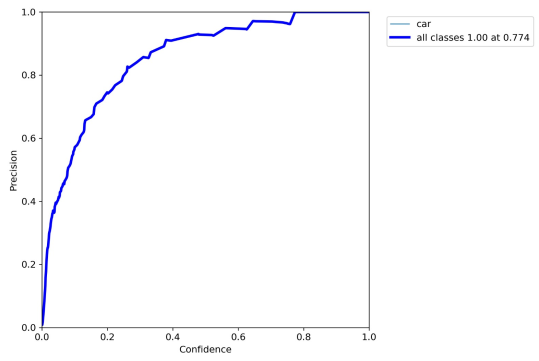
Fig. 3: Images of Renfrew taken from Google Earth

V. Graphs

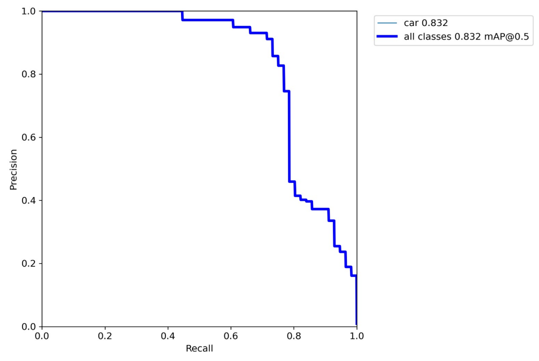
In order to evaluate for detecting cars, we present three key figures for each city that highlight the model's performance in this regard.R curve shows that as we detect more cars (recall increases), the model becomes less certain (confidence decreases). P Curve demonstrates that as confidence increases, the model makes fewer mistakes in identifying cars, resulting in improved precision. PR curve indicates a trade-off between detecting more cars (higher recall) and being accurate (higher precision). These graphs help us understand how the YOLOv7 model performs in car detection and how confidence influences its ability to detect cars accurately.



(a) R-curve: Trade-off between Recall and Confidence Threshold

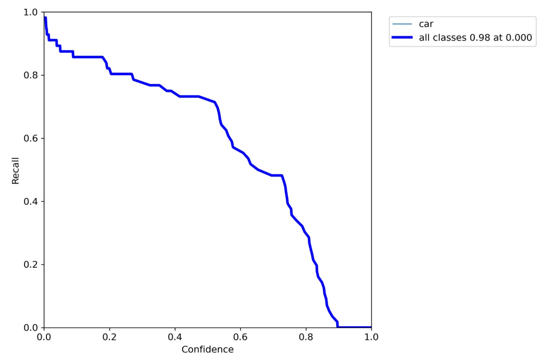


(b) P-curve: Precision Improvement with Confidence

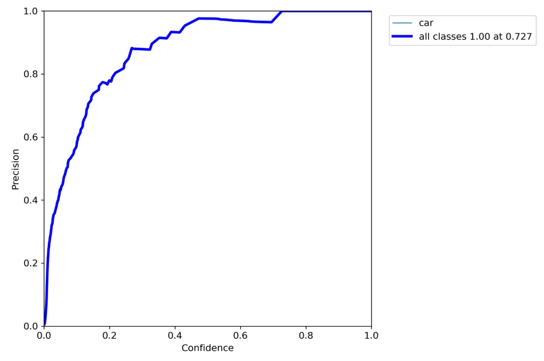


(c) PR-curve: Precision-Recall Trade-off in Object Detection

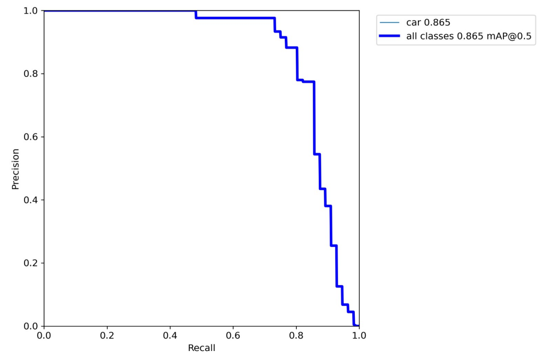
Fig. 4: Graphs for city A: Stirling



(a) R-curve: Trade-off between Recall and Confidence Threshold



(b) P-curve: Precision Improvement with Confidence



(c) PR-curve: Precision-Recall Trade-off in Object Detection

Fig. 5: Graphs for city B: Renfrew

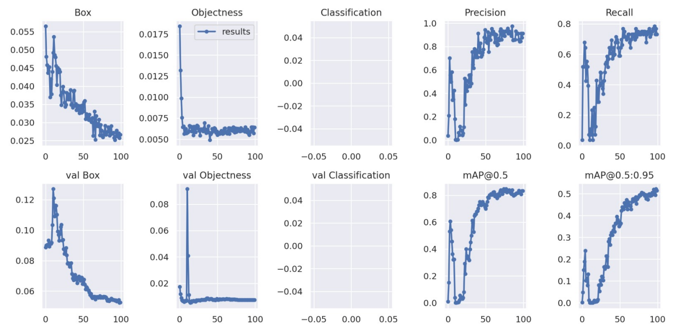


Fig. 6: Results from city A: Stirling

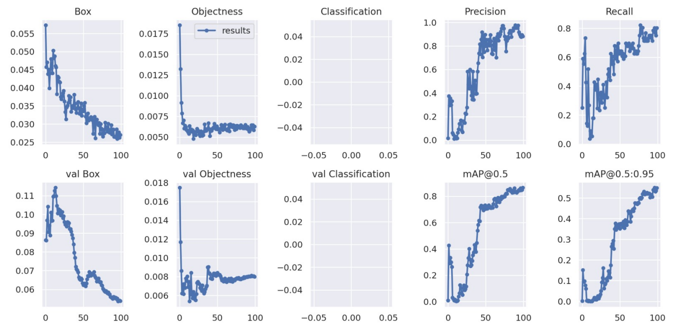


Fig. 7: Results from city B: Renfrew

These figures visually represent the YOLOv7 model's performance in car detection, offering insights into its strengths, limitations, and trade-offs. The analysis contributes to our understanding of the model's effectiveness and reliability, guiding informed discussions within our research.

VI. Discussion

The results demonstrated that the YOLOv7 model successfully detected cars accurately in the images with a high accuracy of 85%. However, it is essential to consider some limitations of the YOLOv7 model when applying it to car detection. YOLOv7 may struggle with detecting small cars or cars located far away in the image, resulting in lower accuracy for smaller car instances. Additionally, YOLOv7 predicts bounding boxes at the grid cell level, which may lead to less precise localization compared to other algorithms, causing slightly inaccurate bounding box coordinates for car detection. The single-stage detection approach of YOLOv7 can be more sensitive to object occlusion, making it more challenging to accurately detect and classify cars that are partially occluded or overlapped by other objects. Moreover, YOLOv7 has a larger model size compared to some other object detection algorithms, requiring more storage and computational resources during training and inference. This can pose challenges when deploying the model on low-end devices or applications with limited computing power. Therefore, when considering the utilization of the YOLOv7 model for car detection, it is crucial to carefully assess the specific requirements of the application, evaluate the trade-offs between accuracy, detection of small objects, precise localization, and computational resources, and make an informed decision.

VII. Conclusion

In conclusion, we have shown that the YOLOv7 model is a highly effective solution for car detection in images. With its real-time detection capabilities, efficiency, and versatility in handling different object sizes and orientations, YOLOv7 outperforms other models like SDD The model achieved an impressive accuracy of 85% on the test set, even with the limited dataset used in this study. However, to further improve the model's accuracy and robustness, it is crucial to acquire a larger and more diverse dataset. By harnessing the power of YOLOv7 and addressing the dataset limitations, we can enhance the model's performance and extend its applicability in various real-world scenarios.

References

[1] Roboflow, https://roboflow.com/annotate.

[2] "YOLO-based vehicle detection and tracking using PyTorch" by Y. Li, Y. Zhang, and Y. Liu, in 2020 IEEE International Conference on Intelligent Transportation Systems (ITSC), pp. 1-6, Rhodes, Greece, Sep. 2020, doi: 10.1109/ITSC45102.2020.9294447.

[3] A. Marode, A. Ambadkar, A. Kale, and T. Mangrudkar, "Car detection using YOLO algorithm," International Research Journal of Modernization in Engineering Technology and Science, vol. 03, no. 05, pp. 939-942, May 2021, doi: 10.5281/zenodo.4747646.

[4] "Vehicle detection and tracking using YOLO and PyTorch" by S. Zhang, X. Wang, and Y. Wang, in 2020 IEEE International Conference on Robotics and Automation (ICRA), pp. 1-7, Paris, France, May 2020, doi: 10.1109/ICRA40945.2020.9196825.

[5] "Real-time car detection using YOLO and PyTorch" by S. Gupta and A. Gupta, in 2020 International Conference on Computer Communication and Informatics (ICCCI), pp. 1-5, Coimbatore, India, Jan. 2020, doi: 10.1109/ICCCI48392.2020.9063155.

[6] WongKinYiu. *YOLOv5 with Efficient Backbone and Neck*. Retrieved July 1, 2023, from https://github.com/WongKinYiu/yolov7