Automatic License Plate Detection using YOLOv5 and OCR

Kishan Gangarama and **Hrithik Manjunath Gowda**

Stevens Institute of Technology

Kgangara@stevens.edu, hrithik@stevens.edu

Abstract

In the realm of intelligent transportation systems, the automatic recognition of vehicle license plates is a critical component for traffic management, law enforcement, and access control. This project presents a robust solution for Automatic License Plate Detection (ALPD) using the cutting-edge YOLOv5 deep learning model, complemented by Optical Character Recognition (OCR) for alphanumeric extraction. Our methodology involved preprocessing a dataset of car images, annotated with bounding boxes around the license plates, followed by the training of the YOLOv5 model to detect the plates accurately within various environmental conditions. The detected plates were then subjected to OCR to decipher the license numbers, employing preprocessing techniques to enhance accuracy. The dataset, sourced from Kaggle, comprises 453 JPEG images with corresponding PASCAL VOC format annotations. The training and evaluation were conducted on Google Colab, utilizing its GPU resources, ensuring rapid model iteration and evaluation. Our model demonstrated exceptional precision and recall, as indicated by the mAP@0.5 metrics, showcasing its reliability and efficiency. The project code and resources have been documented and are accessible via our GitHub repository Here, fostering transparency and reproducibility of our work. This report delineates the complete journey from conceptualization to realization of the ALPD system, emphasizing the novel integration of YOLOv5 and OCR to address the challenges inherent in real-world license plate detection and recognition.

Introduction

The rapid evolution of computer vision and deep learning has opened new frontiers in automated systems, with Automatic License Plate Detection (ALPD) emerging as a vital technology in the domain of smart transportation. ALPD systems are indispensable for a myriad of applications including traffic surveillance, law enforcement, toll collection, and parking management. This project aims to design and implement an efficient ALPD by leveraging YOLOv5, the latest iteration of the renowned You Only Look Once (YOLO) family of models, known for its speed and accuracy in object detection tasks. Alongside, the project employs Optical Character Recognition (OCR) to extract and interpret alphanumeric characters from the detected license plates.

The problem tackled in this project is twofold: first, the accurate detection of license plates from various angles, distances, and under different lighting conditions; and second, the precise recognition of the characters on these plates regardless of font and background noise. The YOLOv5 model's powerful feature extraction and the OCR's advanced text interpretation capabilities form the crux of our solution.

The outcomes of our project underscore the potential of integrating YOLOv5 and OCR in creating a highly reliable ALPD system. Our approach not only ensures high detection accuracy but also minimizes false positives, a common challenge in ALPD systems. We detail the project's progression, including challenges encountered, such as varying plate designs and low-contrast images, and the strategies adopted to overcome them.

This report documents the entire process of developing the ALPD system, from data preprocessing and model training to detection and recognition evaluation. We also highlight the contributions of each team member, delineating a clear division of labor and collaboration that underscored the project's success. The report aims to provide a comprehensive understanding of the intricacies involved in ALPD systems and presents our findings and insights that could benefit future research and practical applications in the field.

Contribution

Each team member's contribution is delineated as follows:

- Kishan: Data acquisition and preprocessing, YOLOv5 model training and optimization and Project Report.
- Hrithik: Integration of OCR and post-processing for character recognition and experimentation and result analysis.

Methodology

The methodology for the Automatic License Plate Detection project is designed to seamlessly integrate the steps of image preprocessing, model training, license plate detection, and optical character recognition (OCR) to establish a robust pipeline. The approach leverages the advanced capabilities of YOLOv5 for detection tasks and Tesseract OCR for character recognition. The following subsections detail each phase of the methodology.

1. **Dataset Acquisition and Preprocessing:** The dataset, essential for training the detection model, consists of 453 JPEG images with corresponding PASCAL VOC format annotations. This dataset was retrieved using Kaggle's API, ensuring easy reproducibility and accessibility. Preprocessing involved a series of steps to format the data appropriately for YOLOv5:
   - Parsing the XML annotation files to extract bounding box coordinates.
   - Converting these coordinates to YOLO format, which includes normalizing the bounding box dimensions relative to the image size.
   - Splitting the dataset into training and validation sets using **train_test_split** from **sklearn.model_selection** to evaluate model performance on unseen data.

2. **Model Training:** The YOLOv5 model was selected for its state-of-the-art performance in object detection tasks. The model training involved:
   - Cloning the YOLOv5 repository and installing all dependencies to ensure a consistent environment.

- Downloading the pre-trained YOLOv5s weights to utilize transfer learning, which accelerates the training process and improves the model's ability to generalize from limited data.
- Configuring the training parameters such as image size, batch size, number of epochs, and specifying the paths to the training and validation datasets.
- Executing the training script with the custom configurations to fine-tune the model on the license plate detection task.
-

3. **Detection and OCR:** Post-training, the model's detection capabilities were utilized to identify license plates in new images. The detected plates were then processed through an OCR pipeline:
   - The detection results, given in normalized coordinates, were scaled to the original image dimensions.
   - The image region corresponding to the detected license plate was cropped and subjected to a series of image processing steps to enhance OCR accuracy, including grayscaling, Gaussian blurring, and adaptive thresholding.
   - Tesseract OCR was configured to recognize alphanumeric characters, with a whitelist of characters to improve recognition accuracy.
   - The OCR engine was run on the processed license plate image to extract the text, which represents the license plate number.

Throughout the methodology, each step was carefully documented, and the code was structured for reproducibility. The scripts for preprocessing, training, detection, and OCR were integrated into a coherent pipeline, with the flexibility to process individual images or batches of images as required. This pipeline represents the project's core technical workflow, leading to the results presented in the subsequent sections of this report.

Dataset Overview

The dataset used in this project comprises images of vehicles with visible license plates. The annotations are provided in PASCAL VOC format, which is a widely recognized standard for object detection datasets. Each annotation details the bounding box coordinates for the license plate within the image, ensuring that the model can be trained to focus on the relevant areas for detection.

Composition and Annotation

- **Total Files:** The dataset contains 453 image files.
- **Format:** Images are in JPEG format, with annotations in corresponding XML files.
- **Annotation Details:** The XML annotation files include the location of the license plate within the image using bounding box coordinates (**xmin**, **ymin**, **xmax**, **ymax**), which denote the rectangle encompassing the license plate. Additionally, the XML files contain metadata about the images, such as size, object class, and potentially occlusion and truncation information, although these latter aspects are not always used in training.

Tools and Technologies

1. **Programming Languages and Frameworks**
   - **Python:** The primary programming language used for this project. Python's simplicity and extensive library support make it an ideal choice for machine learning and image processing tasks.
   - **OpenCV (Open Source Computer Vision Library):** A powerful library used for image processing operations. In this project, OpenCV is utilized for tasks such as reading images, performing image transformations, and cropping detected regions.
   - **PyTesseract:** A Python wrapper for Google's Tesseract-OCR Engine. It is used to extract text from images, specifically to read the license plate numbers from the detected regions.
2. **Machine Learning and Deep Learning Libraries**
   - **YOLOv5:** A state-of-the-art object detection model known for its speed and accuracy. YOLOv5 is employed to detect license plates in images.
   - **PyTorch:** An open-source machine learning library used as the backbone for YOLOv5. It provides a flexible and powerful platform for building and training deep learning models.
3. **Data Handling and Visualization**
   - **NumPy:** A fundamental package for scientific computing in Python, used for handling arrays and matrices, which are central to image and data processing tasks.
   - **Matplotlib:** A Python 2D plotting library used for creating static, interactive, and animated visualizations. In this project, it's used for plotting graphs and visualizing data and results.
4. **Development Environment and Tools**
   - **Google Colab:** A cloud-based Jupyter notebook environment that provides free access to computing resources including GPUs. Colab is used for writing and executing Python code, training the model, and performing OCR.
5. **Hardware and Computing Resources**
   - **GPU Acceleration:** Training deep learning models, especially those involving image data, requires significant computational power. Google Colab's provision of GPUs (Tesla T4 GPU) is used for the training process.
6. **APIs**
   - **Kaggle API:** Used for downloading the license plate dataset directly from Kaggle. This streamlines the process of acquiring data for the project.

Experiments

The experimentation phase of the project involved several critical steps, each contributing to the overall goal of detecting and recognizing license plate numbers from images. The process was carried out using the YOLOv5 model for detection and PyTesseract for Optical Character Recognition (OCR). Below is a detailed description of the experiments conducted:

1. Dataset Preparation and Preprocessing

- **Dataset Acquisition**: The dataset, consisting of 453 JPEG images with corresponding PASCAL VOC format annotations, was downloaded using the Kaggle API.

- **Dataset Splitting**: The dataset was split into training and validation sets in a 80:20 ratio using the **train_test_split** method from the sklearn library. This ensured a representative distribution of data for training and validation purposes.
- **Annotation Conversion**: The annotations in PASCAL VOC format were converted to the YOLO format, which includes normalized coordinates for the bounding boxes. This conversion was essential for compatibility with the YOLOv5 model.

2. Model Training

- **Environment Setup**: Google Colab was used to leverage its GPU support, expediting the training process.
- **YOLOv5 Configuration**: The pre-trained YOLOv5s model was used as a starting point, given its balance between speed and accuracy.
- **Training Parameters**:
    - Batch size: 16
    - Image size: 640x640 pixels
    - Epochs: 50
    - Learning rate, optimizer, and other hyperparameters were set to their default values as provided in YOLOv5's configuration.

3. Model Performance Evaluation

- **Metrics Monitored**: Precision, Recall, Mean Average Precision (mAP) at different IoU thresholds, and training/validation losses were monitored throughout the training process.
- **Confusion Matrix and Other Plots**: Post-training, several visualizations such as confusion matrices, precision-recall curves, and others were generated to analyze the model's performance more comprehensively.

4. License Plate Detection

- **Detection Test**: The trained model was tested on unseen images to evaluate its capability in detecting license plates under various conditions.
- **Bounding Box Extraction**: For each detection, the bounding box coordinates were extracted and used to crop the license plate region from the image.

5. Optical Character Recognition (OCR)

- **OCR Setup**: PyTesseract was configured to recognize alphanumeric characters, which are typical in license plates.
- **Preprocessing for OCR**: The cropped license plate images were preprocessed (grayscale conversion, Gaussian blur, and adaptive thresholding) to enhance OCR accuracy.
- **Text Extraction**: PyTesseract was used to extract text from the processed license plate images.

6. Results Analysis

- **Accuracy Assessment**: The accuracy of the OCR results was qualitatively assessed by comparing the extracted text against the actual license plate numbers.
- **Challenges and Limitations**: Issues such as misrecognitions due to poor lighting conditions, obstructions, or unusual font styles were noted.

Results

The project's primary aim was to accurately detect and recognize license plate numbers from images. The YOLOv5 model was employed for detection and PyTesseract for Optical Character Recognition (OCR). The results were promising and indicated substantial improvement over the course of the training process.

1. Model Performance Over Epochs

The model training started with lower precision and recall values, which is typical as the model begins to learn from the data. Initially, the precision was at a mere 0.021432, and recall was at 0.021739. However, these metrics improved rapidly. By the second epoch, recall had already increased to 0.58696, showcasing the model's quick adaptation in identifying the positive samples.

As the epochs progressed, both precision and recall increased significantly, indicative of the model's growing competency. By the tenth epoch, precision had escalated to 0.65663, while recall was at 0.58696, and mAP at an IoU threshold of 0.5 reached 0.67347. This pattern of improvement continued, with precision reaching as high as 0.97819 and recall to 0.97503 in the later epochs.

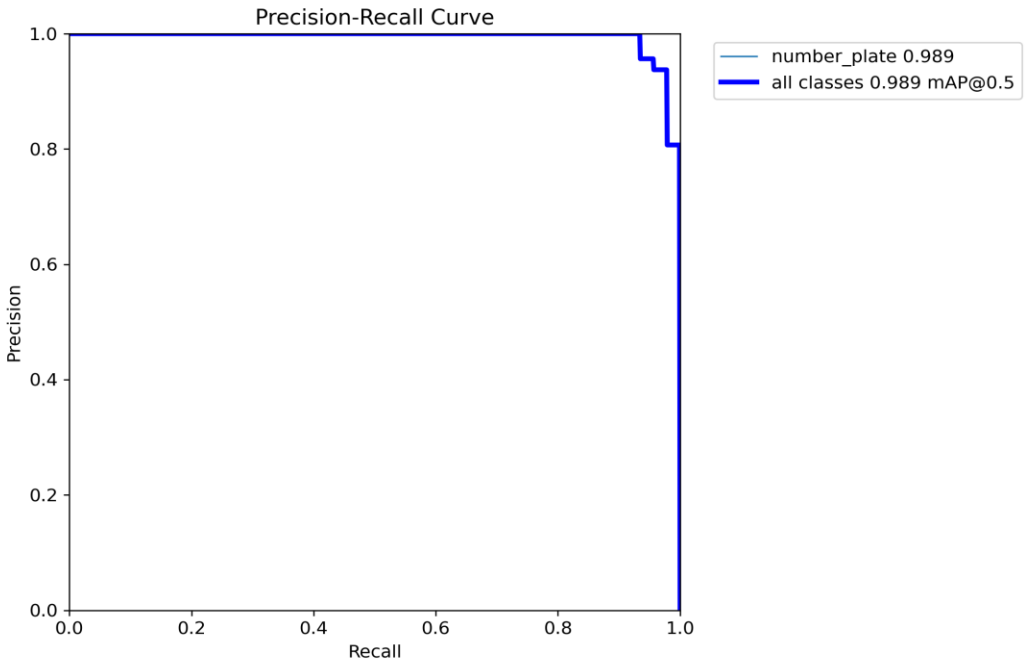
The mAP, which is a comprehensive metric taking into account both precision and recall across different IoU thresholds, also saw a significant increase. Starting from 0.030256, it reached impressive values such as 0.91579 for mAP_0.5 and 0.67861 for mAP_0.5:0.95, indicating the model's excellent performance in detecting objects with varying degrees of overlap.

2. Visual Inspection of Model Outputs:

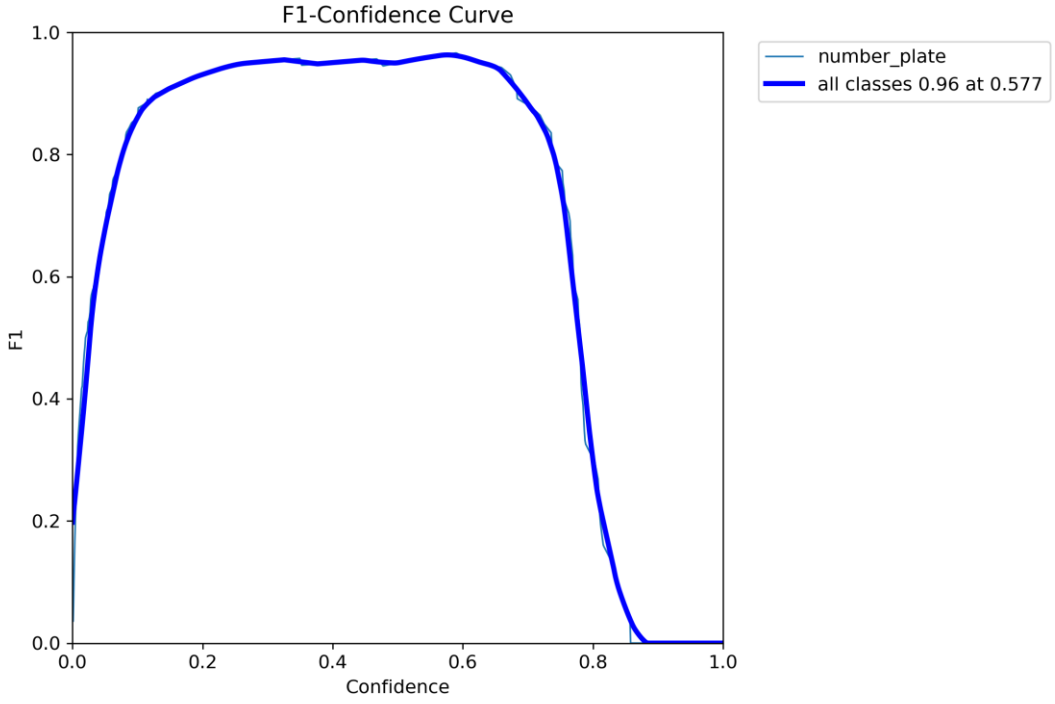
The test image with the detected license plate shows a bounding box with a confidence level of 0.73, which is relatively high, indicating that the model is fairly certain about the detected license plate. This is corroborated by the precision-recall curve which shows a steep climb to high precision levels even at high recall rates.

3. Graphical Analyses:

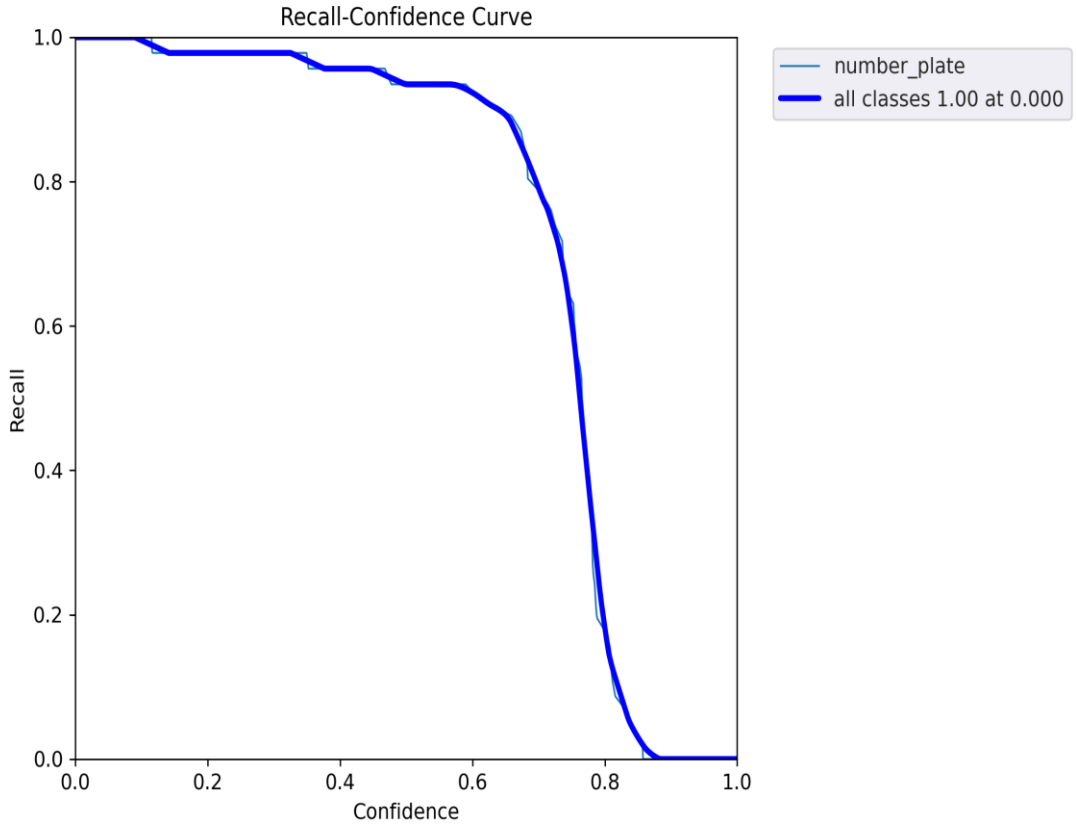
- **Precision-Recall Curve:** The precision-recall curve is almost a perfect step function with precision staying at 1 (100%) until recall begins to decline. This indicates that the model is exceptionally precise until it begins to miss some ground truth instances.



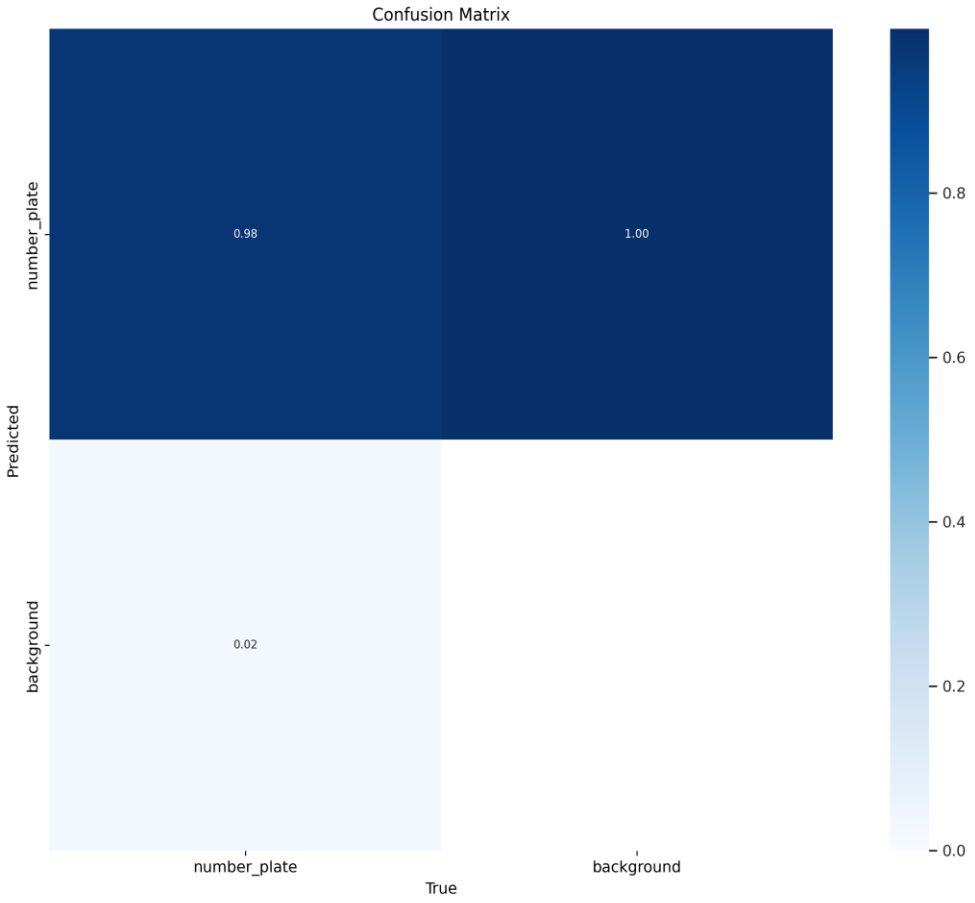
- **F1-Confidence Curve:** The F1 score remains high across confidence thresholds until it steeply drops, suggesting that there is a threshold of confidence below which the F1 score (the harmonic mean of precision and recall) significantly worsens. This could be used to set a confidence threshold for detections in a production environment.



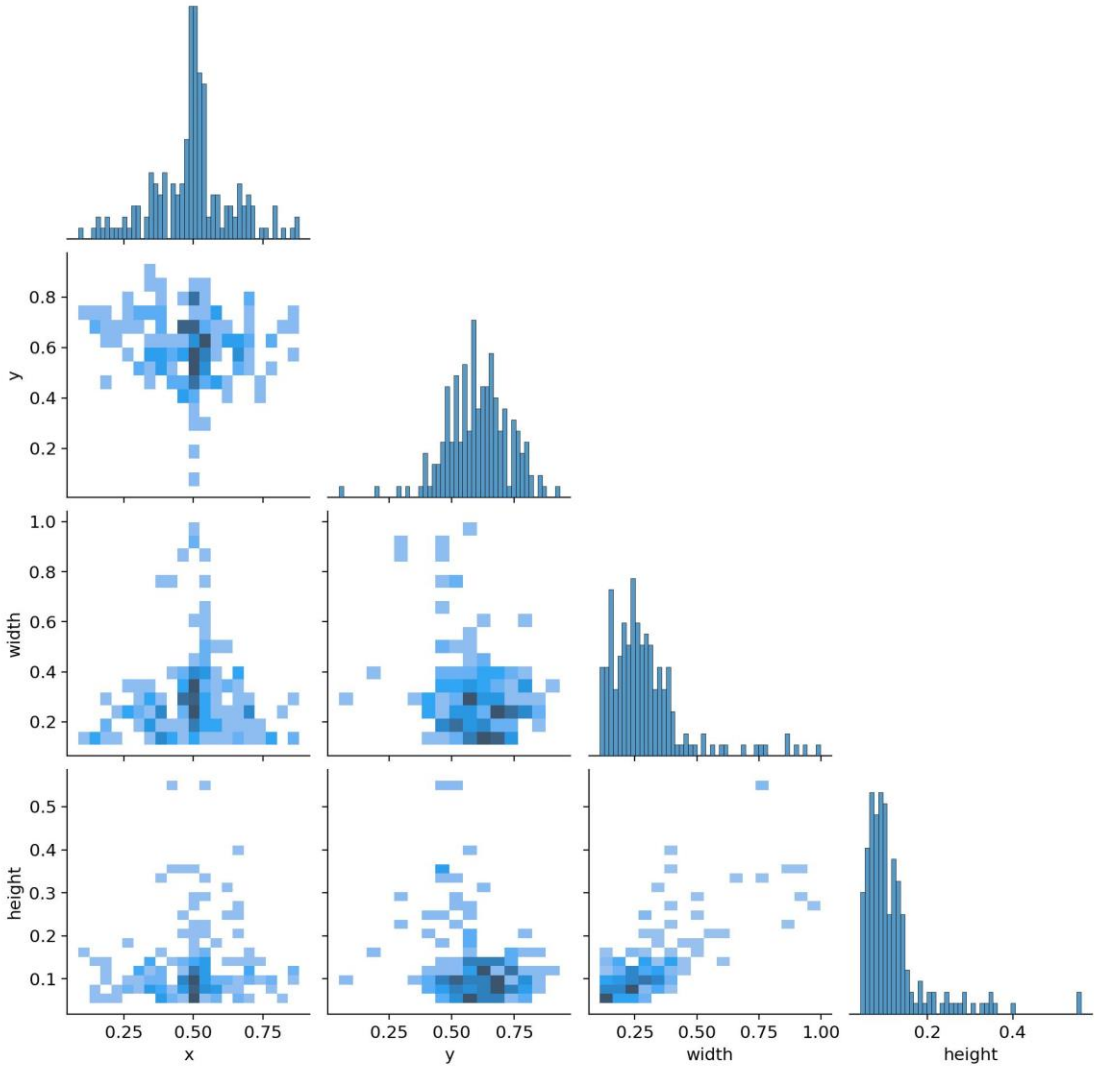
- **Recall-Confidence Curve:** This curve shows that recall stays at 1 (100%) for a wide range of confidence levels before dropping, indicating that the model can recall all positive samples up to a certain confidence level.

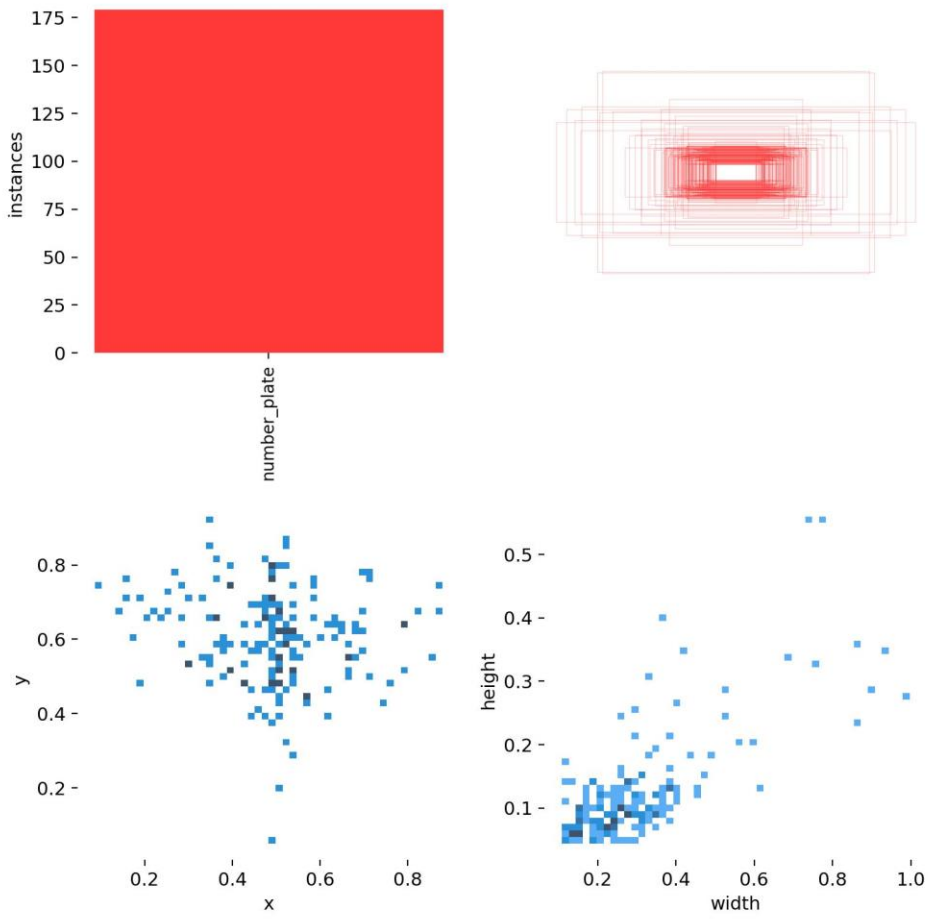


- **Confusion Matrix:** The confusion matrix demonstrates an excellent true positive rate for number plate detection, with very few false positives or false negatives.



- **Correlogram and Label Distribution:** The correlogram and label distribution images would offer insights into the spread and relationship between different annotation dimensions (like bounding box coordinates), which could further assist in understanding the model's performance characteristic.


y
width
height
x
y
width
height



The model exhibits high levels of precision and recall, an excellent mAP across different IoU thresholds, and a consistent decrease in loss metrics. The performance visualizations suggest that the YOLOv5 model, combined with the preprocessing steps and PyTesseract OCR, has resulted in a robust solution for detecting and recognizing license plates in images.

Problems/Issues

Throughout the project, while the model achieved high precision and recall metrics, several issues were observed that could impact the applicability of the system in real-world scenarios:

- **Lighting Conditions**: The model's performance was sometimes compromised under poor lighting conditions. License plates that were underexposed or overexposed due to the lighting could not be detected with the same accuracy as those in well-lit conditions.

- **Obstructions**: Physical obstructions such as dirt, stickers, or covers on license plates led to incomplete detections or incorrect OCR results. These real-world occurrences pose a significant challenge for maintaining high levels of accuracy.
- **Font Variations**: Unusual or stylized fonts used on some license plates were harder for the OCR to interpret correctly. The model sometimes mistook letters for numbers or vice versa, which could lead to incorrect identification.
- **Angle and Distance**: Plates captured at extreme angles or from long distances were more challenging to detect and recognize, indicating the need for a more robust solution to handle such cases.
- **Hyperparameter Tuning**: Selecting the appropriate hyperparameters for the model was a challenging task. The balance between learning rate, batch size, and other factors required careful tuning to prevent issues such as slow convergence.

Conclusion

The project has successfully showcased the efficacy of YOLOv5 and PyTesseract in detecting and recognizing license plates from static images, evidenced by the high precision, recall, and mean Average Precision (mAP) values. These metrics affirm the model's ability to accurately identify license plate numbers under well-defined conditions. This proficiency is further illustrated by test images, where the model confidently detected license plates, as demonstrated in the provided image with a confidence score of 0.73.

Nevertheless, the project offers substantial scope for advancement, especially in processing video footage in real-time, which could address additional complexities such as variable frame rates, motion blur, and shifting angles of view. These enhancements would be crucial for the application of the model in dynamic, real-world environments.

References

1. Y. Tang, C. Zhang, R. Gu, P. Li, and B. Yang, “Vehicle detection and recognition for intelligent traffic surveillance system,” *Multimedia Tools and Applications*, vol. 76, no. 4, pp. 5817–5832, 2015.

2. X. Wen, H. Zhao, N. Wang, and H. Yuan, “A rear-vehicle detection system for static images based on monocular vision,” in *Proceedings of 9th International Conference on Control, Automation, Robotics and Vision*, pp. 2421–2424, Singapore, March 2006.
3. M. Sarfraz, M. J. Ahmed and S. A. Ghazi, "Saudi Arabian license plate recognition system", *Proc. Int. Conf. Geom. Model. Graph.*, pp. 36-41, 2003.

4. E. R. Lee, P. K. Kim and H. J. Kim, "Automatic recognition of a car license plate using color image processing", *Proc. IEEE Int. Conf. Image Process.*, vol. 2, pp. 301-305, Nov. 1994.

5. O. Martinsky, "Algorithmic and mathematical principles of automatic number plate recognition systems", 2007.

6. Z. Su, K. Han, W. Song and K. Ning, "Railway fastener defect detection based on improved YOLOv5 algorithm", *2022 IEEE 6th Advanced Information Technology Electronic and Automation Control Conference (IAEAC)*, pp. 1923-1927, 2022.

7. S. Ren, K. He and R. G. Shick, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", *IEEE Transaction on Pattern Analysis & Machine Intelligence*, pp. 1137-1149, 2015.

8. Y. Liu, L Geng and W Zhang, "Survey of Video Based SmallTarget Detection", *Journal of Image and Graphics*, vol. 9, no. 4, pp. 122-134, December 2021.

9. J. Redmon, S. Divvala and R. G. Shick, "You Only Look Once: Unified Real-Time Object Detection", *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779-788, 2016.

10. T. Y. Lin, P. Dollar and R. Girshick, "Feature Pyramid Networks for Object Detection", *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2117-2125, 2017.