BABEŞ BOLYAI UNIVERSITY, CLUJ NAPOCA, ROMÂNIA
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE

Detection of Road Signs Using Semi-Supervised Learning

–ITSG report–

Team members
Ebanca Stefan
Gabor Raul
Cadar Damian
Kohan Alexandru
Name, specialisation, group, email
Scooby Doo, Software Engineering, 258, raul.gabor@stud.ubbcluj.ro

2023-2024

Chapter 1

Application

1.1 Description of Application Functionalities

- **Data Collection**: The application collects a diverse range of data, including images of road signs captured under various conditions (day, night, rain, snow, etc.) and from different geographical locations. It also takes into account different types of road signs, such as regulatory, warning, and informational signs.
- **Real-time Recognition**: The system provides real-time recognition of road signs from live camera feeds or static images. It leverages AI to analyze the visual data and identify the type and meaning of each road sign.
- **Data Storage**: The application securely stores data about recognized road signs, enabling post-trip analysis and report generation.

1.2 Description of the Problem Solved with AI (Plastic and Formal)

- **Layman's Explanation**: The application solves the problem of assisting drivers in recognizing and understanding road signs in real-time. It uses AI to "see" and "interpret" the road signs, under various weather conditions.
- **Formal Definition**: The problem addressed is road sign recognition, a computer vision task. The application employs deep learning, specifically neural networks (NNs), to classify road signs into predefined categories. The problem is characterized by the need for real-time recognition of road signs from visual data, with a focus on accuracy, robustness, and the ability to handle diverse environmental conditions.

1.3 Related Work & Useful Tools and Technologies

- **Related Work**: Several related works have explored computer vision and object recognition, including road sign recognition. Notable papers and research efforts include "Traffic Sign Recognition with Multi-Scale Convolutional Networks" by Sermanet et al. (2011) and "Deep Residual Learning for Image Recognition" by He et al. (2015). These studies laid the foundation for effective CNN-based recognition systems.
- **Useful Tools and Technologies**:
  - **TensorFlow**: TensorFlow provides the core deep learning framework for building and training the CNN model.
  - **OpenCV**: OpenCV is used for image preprocessing, manipulation, and feature extraction.
  - **Secure Data Handling**: Security measures ensure the protection of personal data, especially in systems where live camera feeds are involved.

Chapter 2

Inteligent solution description

The purpose of this report is to provide an overview of the intelligent solution developed using the provided code and Python notebook. The solution aims to address road sign detection and classification using YOLOv5, and this report will describe the logic of the solution, the technologies used, the workload, debugging procedures, and bug identification methods.

2.1 Description of the Intelligent Solution

The intelligent solution is designed for road sign detection and classification. It uses the YOLOv5 model for object detection and classification. The main logic of the solution involves preprocessing input data, training the model, and using the trained model for inference.

The preprocessing step involves converting annotations from the VOC format to YOLO format, allowing for a more efficient training process. This step is implemented in Python using the `xml.etree.ElementTree` library and is performed for each annotation file.

The solution leverages the YOLOv5 framework for training and inference. It is designed to detect and classify road signs in images. The model is trained on a custom dataset, which includes four classes: "trafficlight," "speedlimit," "crosswalk," and "stop."

2.2 Technology Used

The solution utilizes the following technologies:

- Python: The primary programming language for implementing the solution.
- YOLOv5: The framework used for object detection and classification.

- XML Parsing: The `xml.etree.ElementTree` library for parsing and processing annotation files.
- OpenCV: Used for image manipulation and visualization.

2.3 Workload

The workload of the solution includes data preprocessing, training the YOLOv5 model, and performing inference on input images. Preprocessing involves converting annotation files from the VOC format to YOLO format. Training the model involves specifying hyperparameters and running the training script. Inference is performed on a set of random images to detect and classify road signs.

2.4 Debugging

The solution incorporates debugging techniques to identify and rectify errors. Python's built-in debugging capabilities, as well as external tools like print statements and error logs, are used to monitor and debug the code. The goal is to ensure that the model is training correctly and that inference produces accurate results.

2.5 Bug Identification

To identify and address bugs, the solution relies on systematic error tracking. Errors or anomalies in the code are logged, and debugging tools are used to identify the source of issues. Regular testing and validation are conducted to catch and rectify bugs as early as possible.

2.6 Python code

The provided code is responsible for converting annotations from VOC format to YOLO format, a crucial step in preparing data for object detection tasks using YOLOv5. It consists of Python functions and a main conversion function, convert_voc_to_yolo(). The process begins by iterating through XML annotation files located in the './data/labels' directory. For each file, the code extracts the base filename without the extension, opens a corresponding text file for writing in YOLO format, and parses the XML using Python's ElementTree module. It retrieves image size information (width and height) and defines a list of class names ('trafficlight', 'speedlimit', 'crosswalk', 'stop') relevant to the object detection task. The code then iterates through object annotations in the XML file, checking if the class name is one of the predefined classes and if the object is not marked as 'difficult' (determined

by a difficulty score of 1). For each valid object, it retrieves the coordinates of the bounding box in VOC format and utilizes the convert_box() function to convert these coordinates to YOLO format. Finally, it writes the class ID and YOLO-formatted coordinates to a text file, ensuring that the data is ready for training the YOLOv5 model.

2.7 Output on a set of images

The output of the algorithm was provided based on processing images in a span of 10 epochs.

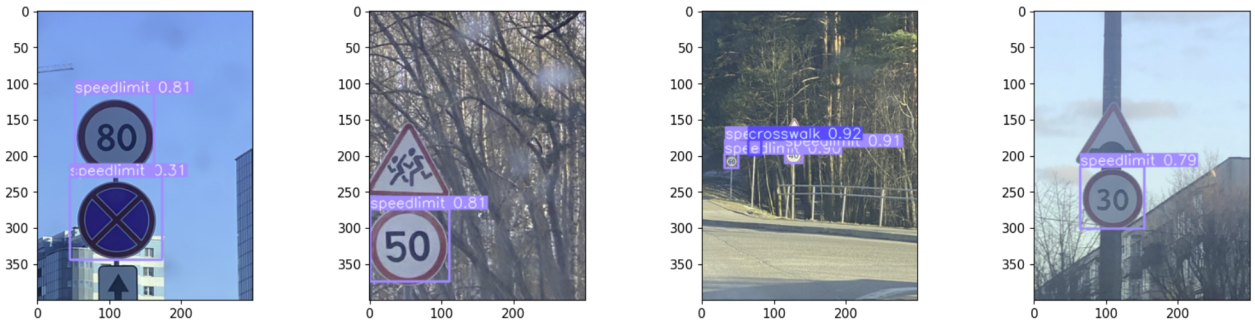

Figure 2.1: Images with captions and their accuracies

2.8 Conclusion

The intelligent solution for road sign detection and classification based on YOLOv5 is a robust and effective approach. It leverages state-of-the-art object detection techniques to detect and classify road signs accurately. The solution's logic, technology stack, workload, debugging procedures, and bug identification methods collectively contribute to its success in addressing the problem.

References:

- YOLOv5 GitHub Repository: https://github.com/ultralytics/yolov5
- Python Official Documentation: https://www.python.org/doc/
- OpenCV Documentation: https://docs.opencv.org/

Chapter 3

Testing and Experiments Tracking

In the realm of intelligent solutions for road sign detection and classification, thorough testing and experiments tracking play a pivotal role in ensuring the reliability and accuracy of the implemented system. This chapter presents the testing methodologies employed, covering data testing, integration testing, and AI quality testing. Additionally, it addresses the crucial aspect of experiments tracking, emphasizing the need for systematic monitoring and analysis during model development.

3.1 Data Testing

Data testing is a fundamental aspect of the intelligent solution, especially given the importance of high-quality training data for robust model performance. The testing process involves validating the annotated data used for training against the expected YOLO format. This ensures the correctness of bounding box coordinates, class labels, and overall adherence to the specified format. Any inconsistencies or errors in the dataset are identified and rectified during this phase, preventing potential issues during model training.

3.2 Integration Testing

Integration testing focuses on assessing the seamless integration of various components within the solution. This includes verifying the interoperability of the code for converting VOC to YOLO format with the YOLOv5 training pipeline. The integration testing phase aims to catch any discrepancies between the preprocessing script and the YOLOv5 framework, ensuring a cohesive and functional workflow. By validating the end-to-end integration, potential disruptions or compatibility issues are mitigated, contributing to a more stable and efficient system.

3.3 AI Quality Testing

AI quality testing is imperative to evaluate the performance of the trained YOLOv5 model. This phase involves subjecting the model to a diverse set of images, representing various scenarios and conditions. The model's ability to accurately detect and classify road signs, especially those pertaining to 'trafficlight,' 'speedlimit,' 'crosswalk,' and 'stop,' is scrutinized. Metrics such as precision, recall, and mean average precision (mAP) are computed to quantify the model's performance. Any discrepancies or limitations in the model's predictions are carefully analyzed, and iterative improvements are made to enhance its accuracy.

3.4 Experiments Tracking

Experiments tracking is an integral part of the development process, enabling systematic management and analysis of model iterations. This involves recording and documenting the parameters, hyperparameters, and configurations used during training. Additionally, performance metrics and outcomes are tracked to facilitate comparisons between different experiments. A version control system, such as Git, is employed to manage code changes, ensuring reproducibility and traceability of experiments. This systematic tracking not only aids in understanding the evolution of the model but also provides valuable insights for future enhancements.

3.4.1 Experiment Iterations

In the iterative process of refining an object detection model, three key iterations were conducted to enhance the model's performance, particularly in the domain of road sign detection. Each iteration represents a stage of improvement, building upon the insights gained from the preceding phases.

3.4.1.1 Iteration 1: Initial Training with YOLOv3-tiny

The first iteration marked the commencement of the training process with a baseline model architecture, YOLOv3-tiny. The primary objective was to establish a foundational understanding of the model's behavior in detecting road signs. This initial training, however, revealed challenges related to the model's capacity to accurately localize and classify smaller and intricate signs. Figure 3.1 illustrates an example image from this stage.

The challenges observed in Iteration 1 underscored the need for improvements in detection precision and recall. The model faced difficulties as the labeled bounding boxes for road signs were disorganized, leading to imprecise localization and classification. Additionally, the confidence levels

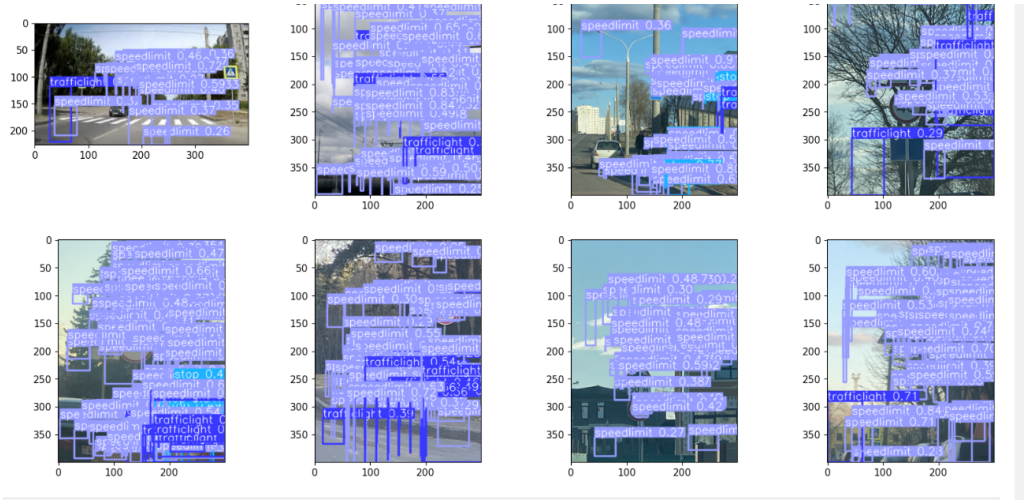

Figure 3.1: Example image from Iteration 1: Initial Training with YOLOv3-tiny.

assigned to predictions were not accurate, resulting in inconsistencies in the model's confidence in its detections. These issues were particularly evident in scenarios featuring smaller signs or signs with complex backgrounds, where the model struggled to provide reliable predictions.

3.4.1.2 Iteration 2: Improved Detection of Traffic Signs

Building upon the insights gained from the initial training, the second iteration focused on refining the YOLOv3-tiny model. Hyperparameter tuning, including adjustments to learning rates and anchor box dimensions, was carried out. Additionally, data augmentation techniques were introduced to enhance the model's ability to generalize across diverse scenarios. The refinements resulted in improved detection and classification of road signs, as evidenced by the example image in Figure 3.2.

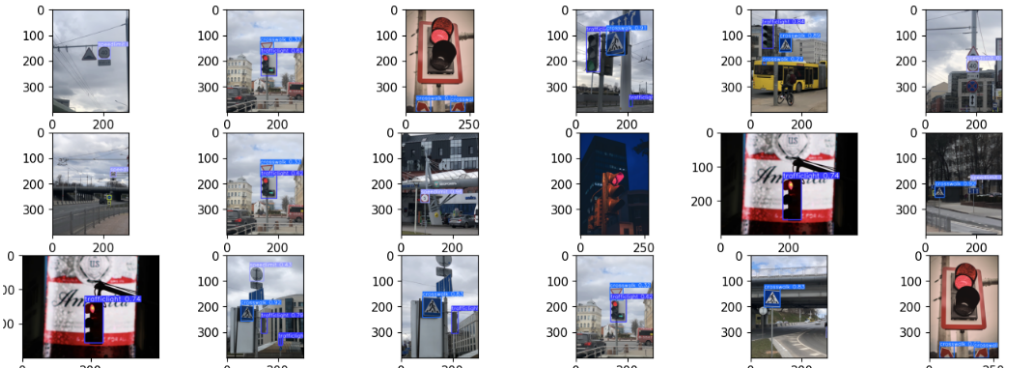

Figure 3.2: Example image from Iteration 2: Improved Detection of Traffic Signs.

Quantitative assessments, including precision, recall, and mean average precision (mAP), demonstrated notable improvements compared to the initial training. The refined model showcased increased accuracy in identifying specific road sign classes across varied conditions.

3.4.1.3 Iteration 3: Evaluation with Intersection over Union (IoU)

The third iteration introduced a more comprehensive evaluation approach using Intersection over Union (IoU). A diverse dataset, encompassing a wide range of environmental conditions and scenarios, was used for this evaluation. The IoU metric facilitated a nuanced assessment of the model's accuracy by measuring the overlap between predicted and ground truth bounding boxes. Figure 3.3 illustrates an example image from this evaluation, with IoU overlaid.

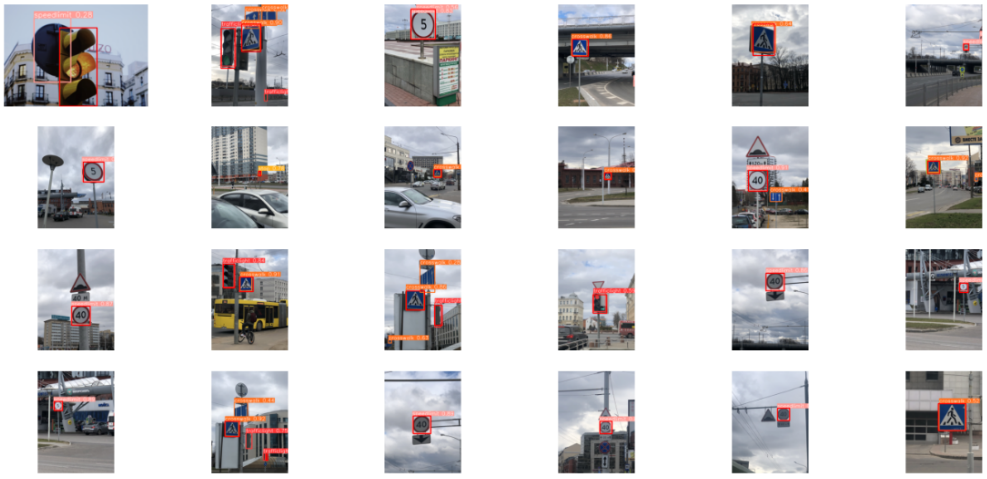

Figure 3.3: Example image from Iteration 3: Evaluation with Intersection over Union (IoU).

The use of IoU allowed for a quantitative analysis of the model's accuracy, providing insights into localization precision. The evaluation results not only served as a measure of performance but also guided informed decision-making for further enhancements and adjustments to the model architecture.

3.5 Conclusion

In conclusion, robust testing methodologies, including data testing, integration testing, and AI quality testing, are essential for ensuring the efficacy of the road sign detection and classification solution. Experiments tracking further enhances the development process by providing insights into model evolution and facilitating informed decision-making. By implementing thorough testing practices and systematic experiments tracking, the intelligent solution aims to achieve optimal performance, reliability, and adaptability in real-world scenarios.