Team07 Simulation-Based Autonomous Driving In Crowded City Final Report

Shihong Zhang, Technical University of Munich

***Abstract*—This study introduces an innovative end-to-end autonomous driving system developed within a Unity simulator environment, leveraging PyTorch. By integrating a ResNet50 model for steering angle prediction and employing a pre-trained YOLOv8 for real-time detection of vehicles and traffic lights, the system navigates with remarkable accuracy. The paper discusses overcoming challenges such as data imbalance and integrating diverse models for seamless vehicle control. Experimental results demonstrate the system's effectiveness, offering significant insights into the potential of end-to-end learning in autonomous driving applications. Through this work, we provide a novel approach to autonomous vehicle control, highlighting the benefits and feasibility of deep learning models in simulative environments [1].**

***Index Terms*—end to end, autonomous driving, simulator**

I. Introduction

THE quest for autonomous driving technology has become a pivotal focus in the realm of artificial intelligence, promising to revolutionize transportation systems worldwide. Traditional approaches to developing autonomous vehicles often involve complex, modular systems that separately handle tasks such as perception, decision-making, and control. However, the recent surge in deep learning offers an enticing alternative: end-to-end models that can directly map sensory inputs to driving actions. This paper introduces a novel autonomous driving system implemented within a Unity-based simulator, leveraging the power of PyTorch for real-time vehicle control. Our system uniquely integrates a ResNet50 model for steering angle prediction and a pre-trained YOLOv8 model for detecting vehicles and traffic lights, thereby enabling nuanced control over the vehicle's movement. Through this integration, we address significant challenges, including data imbalance and model compatibility, presenting a comprehensive solution that underscores the viability of end-to-end models in autonomous driving research. The following sections detail our methodology, system architecture, experimental setup, and the insights gained through this pioneering approach[2].

II. System Arcitecture

The architecture of our autonomous driving system is designed to facilitate seamless integration between the simulation environment, deep learning models, and real-time vehicle control mechanisms. At its core, the system leverages a Unity-based simulator, renowned for its robustness and flexibility in creating complex environments, which serves as the testbed for our autonomous driving algorithms.

L.Zhou is with the Informatik 6 - Lehrstuhl für Robotik, Künstliche Intelligenz und Echtzeitsysteme (Prof. Knoll), Technische Universität München (TUM), Munich, Germany.

A. Simulation Environment

Our choice of Unity as the simulation platform was driven by its extensive support for 3D graphics, physics, and scripting capabilities, enabling the creation of a realistic driving environment. This environment includes varied terrains, weather conditions, and traffic scenarios, providing a comprehensive suite of challenges for testing our autonomous driving models.

B. Data Collection and Preprocessing

Data collection is conducted through manual control of the simulated vehicle, where screen recording and telemetry tools within Unity capture both visual inputs from the vehicle's camera and dynamic parameters such as speed, steering angle, and brake status. The collected data undergoes preprocessing to balance the distribution of steering angles, addressing the challenge of data imbalance typically seen with a predominance of straight-driving instances. Specifically, we implemented custom scripts to filter and augment the dataset, ensuring a more uniform representation of driving scenarios.

C. Model Integration and Control Logic

1) Steering Angle Prediction : The backbone of our steering control is the ResNet50 model, adapted for the task of predicting steering angles from visual inputs. Given its pre-trained weights on large-scale image datasets, ResNet50 provides a solid foundation for feature extraction. The model inputs are standardized images from the vehicle's front-facing camera, resized to 224x224 pixels. The output, a continuous value representing the steering angle, is derived through a regression head added to the pre-trained model[3]. To train this model, we use Mean Squared Error (MSE) loss and the Adam optimizer, focusing on the fine-tuning of the regression layer while freezing the earlier layers to leverage learned features.

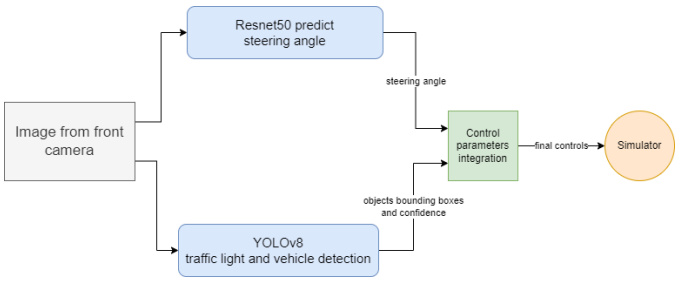

Fig. 1. System Architecture.

2) Obstacle and Traffic Light Detection : For detecting obstacles and traffic lights, our system employs a pre-trained YOLOv8 model. This choice is motivated by YOLOv8's efficiency and accuracy in real-time object detection, crucial for the dynamic aspects of autonomous driving. During inference, the model processes images from the vehicle's camera to identify and locate vehicles and traffic lights within the scene. Based on the detection results and predefined safety thresholds (e.g., distance to a detected vehicle), the system dynamically adjusts the brake status, promoting safety in various traffic conditions.

D. Integration for Vehicle Control

The integration of steering prediction and obstacle detection models underpins our vehicle control logic. The system operates on a simple yet effective principle: maintain a constant throttle unless an obstacle necessitates braking. This logic is realized through a function that evaluates the outputs of both models, prioritizing safety by activating brakes when a potential collision is detected or when a red traffic light is recognized. Conversely, in the absence of immediate hazards, the system focuses on steering control, guiding the vehicle through the simulated environment based on the predicted steering angles.

This architecture not only highlights the synergy between deep learning models and simulation technologies but also showcases the practical implementation of autonomous driving systems in controlled environments. Through rigorous testing and iterative refinement, our system demonstrates promising capabilities, paving the way for further research and development in the field of autonomous vehicles.

III. Exploratory Models and Methodology

In the pursuit of developing an effective end-to-end autonomous driving system, our research journey encompassed a broad exploration of models and methodologies. This section reflects on those initial efforts, providing insights into the iterative process that ultimately shaped our final implementation strategy.

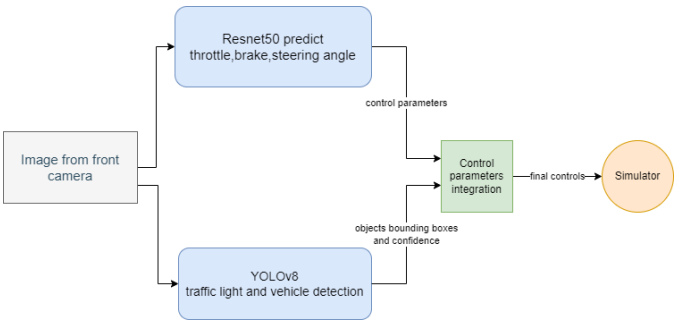

Fig. 2. System Architecture-original.

A. Initial Approach and Rationale

Our project initially experimented with a dual-headed model architecture, aiming to predict both the steering angle (a regression task) and brake/throttle actions (a classification task) within a single network[4]. This approach was motivated by the intention to streamline the decision-making process by consolidating control mechanisms.

B. Challenges Encountered

Despite its conceptual appeal, this dual-headed model presented significant challenges. The primary issue was the divergent nature of the tasks; the regression task required precise, continuous output for the steering angle, while the classification task demanded discrete decisions for braking and acceleration. Balancing the training for these disparate outputs proved difficult, leading to suboptimal performance in preliminary tests.

C. Lessons Learned

The exploration of this integrated model highlighted critical lessons about model complexity and task specificity. We learned that the coupling of fundamentally different tasks could introduce unnecessary complexity, detracting from the model's performance on either task. This realization prompted a reevaluation of our approach, steering us towards a more modular system design.

D. Evolution Towards the Final Model

Informed by these challenges, we shifted our focus to developing separate, specialized models for steering control and object detection. This decision was underscored by the discovery of YOLOv8's capabilities in object detection, which aligned perfectly with our requirements for recognizing obstacles and traffic signals. Simultaneously, ResNet50 emerged as a robust solution for steering prediction, offering a strong feature extraction capability that was adaptable to our needs.

IV. Methodology

This section outlines the methodology adopted in the development of our end-to-end autonomous driving system. The process encompasses data collection, preprocessing, model development, and the integration of machine learning algorithms for vehicle control.

A. Data Collection

The initial step in our approach was the collection of data necessary for training our machine learning models. Utilizing the Unity simulator, we manually controlled the vehicle in various simulated environments to gather a diverse dataset. This dataset not only comprised raw video footage from the vehicle's onboard camera but also included telemetry data, such as steering angles, throttle positions, and brake status, crucial for understanding and predicting realistic vehicle behavior under different driving conditions.

A noteworthy aspect of our data collection was the inclusion of images from three distinct camera perspectives mounted at the front of the vehicle: left, center, and right. This tri-camera setup was designed to capture a wide range of visual

Fig. 3. Image before processing.

inputs, simulating the variability encountered during actual driving. For the purpose of our model training, we employed a custom script to selectively extract only the center images. This decision was motivated by the center camera's direct view of the road ahead, offering the most relevant visual information for steering angle prediction.

Fig. 4. Image after processing.

Following the extraction, these center images underwent further preprocessing to align with the input requirements of our chosen model architecture, ResNet50. This involved resizing the images to 224x224 pixels, the standard input dimension for ResNet50. This resizing step was crucial not only for compatibility with the model but also for ensuring uniformity in the dataset, allowing the neural network to efficiently learn from the visual data without the additional computational burden of processing varying image sizes.

This meticulous approach to data collection and preprocessing forms the foundation of our training dataset, enabling the development of a machine learning model that is both robust and capable of generalizing across a range of driving scenarios within the simulated environment.

B. Data Preprocessing and Augmentation

Given the significant imbalance in the dataset, particularly with a majority of data points representing straight-line driving (steering angle of zero), we employed custom scripts for data preprocessing[5]. The goal was to achieve a more balanced distribution of steering angles, which is crucial for training a model capable of handling a wide range of driving scenarios. This involved selectively filtering out excessive straight-driving instances and augmenting the dataset with transformed images to represent various driving conditions, thereby enriching the dataset with more diverse examples of turning and maneuvering.

C. Steering Angle Prediction Model

The cornerstone of our approach is the use of ResNet50, a deep convolutional neural network, as the backbone for our steering angle prediction model. ResNet50, known for its deep architecture and residual learning framework, facilitates effective feature extraction from input images, a critical aspect of understanding complex driving scenes[6]. We adapted ResNet50 for our specific task by adding a regression head designed to predict the steering angle as a continuous variable directly from the input image. This model was trained using a mean squared error (MSE) loss function, focusing on minimizing the difference between the predicted and actual steering angles. Training was performed with an Adam optimizer, leveraging its adaptive learning rate capabilities to improve convergence[7].

Fig. 5. Training and Validation Loss.

D. Obstacle and Traffic Light Detection Using YOLOv8

In addition to steering angle prediction, our system incorporates a pre-trained YOLOv8 model for the detection of obstacles and traffic lights within the vehicle's environment. YOLOv8, renowned for its speed and accuracy in detecting objects in real-time, allows our system to identify potential hazards and control the vehicle's brakes accordingly. We defined specific logic to interpret the detection results, particularly focusing on the presence of vehicles and the status of traffic lights in the path of the autonomous vehicle. The model's output is integrated into the vehicle control system, ensuring that the vehicle can react to obstacles and traffic signals by adjusting its brake status dynamically.

E. Integration and Vehicle Control Logic

The integration of the steering angle prediction and obstacle detection models forms the basis of our vehicle control logic. A key aspect of our methodology is ensuring that the throttle

and brake controls operate in a mutually exclusive manner, maintaining a constant throttle except when a braking action is initiated based on the detection of obstacles or red traffic lights. This control logic is encapsulated within a function that processes the outputs of both models, determining the appropriate vehicle response to navigate the simulated environment safely and efficiently.

V. YOLOv8 Integration

The integration of YOLOv8 into our autonomous driving system represents a pivotal component of our methodology, enhancing the vehicle's situational awareness and safety. This section delves into the incorporation of YOLOv8 for real-time obstacle and traffic light detection, as well as the subsequent control logic for braking based on these detections.

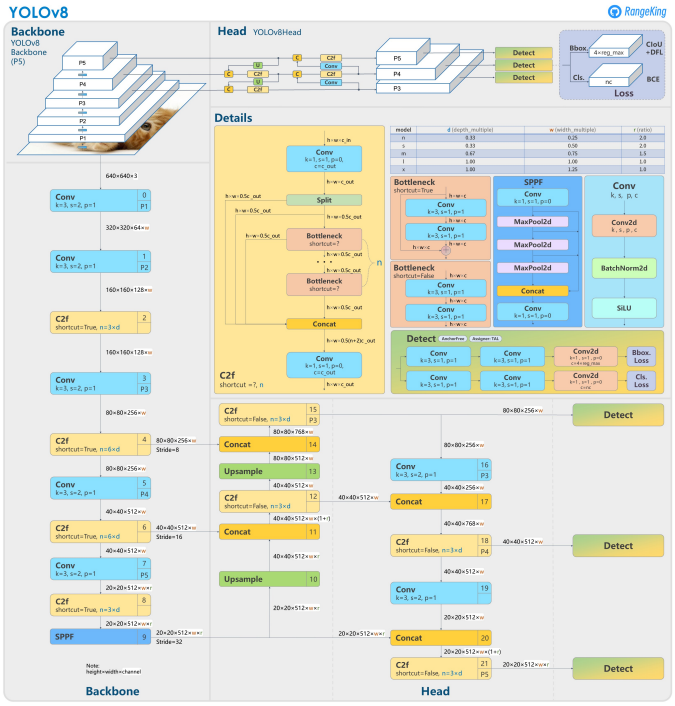

Fig. 6. YOLOv8 Architecture cited from RangeKing.

A. Object Detection Framework

YOLOv8, the latest iteration in the series of You Only Look Once (YOLO) models, is renowned for its exceptional speed and accuracy in detecting objects across a wide range of categories[8]. Our system employs YOLOv8 to identify key objects within the vehicle's field of view, namely other vehicles and traffic lights, which are critical for ensuring safe navigation through the simulated environment. The model processes images captured from the vehicle's front-facing camera, outputting bounding boxes and classification labels for detected objects[9].

B. Model Implementation and Optimization

Given the computational demands of real-time object detection, our implementation of YOLOv8 is optimized for efficiency without sacrificing accuracy. The model has been fine-tuned on a dataset representative of our simulated driving conditions, ensuring high precision in identifying vehicles and traffic lights. Furthermore, we have integrated the model within our system architecture to run concurrently with the steering angle prediction, allowing for simultaneous processing and decision-making.

Fig. 7. Detection from YOLOv8.

C. Detection Logic and Safety Thresholds

The integration of YOLOv8 extends beyond mere object detection; it encompasses sophisticated logic for interpreting the detected objects in the context of autonomous driving. For traffic lights, the system categorizes the detection into green, yellow, and red lights, with specific actions tied to each. Detection of a red light triggers an immediate brake application, overriding other control inputs to ensure compliance with traffic laws.

Similarly, the detection of vehicles ahead involves calculating the distance and relative position of the detected vehicle to determine if braking is necessary. We have established safety thresholds based on the time-to-collision metric, where the system calculates the required braking force to avoid potential collisions, factoring in the vehicle's current speed and the distance to the detected object.

D. Integration with Vehicle Control System

The output from YOLOv8 serves as a critical input to our vehicle's control system. A bespoke function evaluates the detection results, prioritizing safety by implementing braking actions when required, while allowing the vehicle to maintain a constant throttle under safe conditions. This function ensures that the vehicle's acceleration and braking are mutually exclusive, enhancing the predictability and safety of the autonomous driving system.

In cases where both steering adjustments and braking are warranted, the system prioritizes braking to mitigate immediate safety risks, subsequently adjusting the steering angle as needed once the vehicle is deemed to be in a safe state.

E. Impact on Autonomous Driving Performance

The integration of YOLOv8 significantly enhances the autonomous vehicle's ability to navigate complex urban environments. By accurately detecting and responding to vehicles and

traffic lights, our system demonstrates improved situational awareness and decision-making capabilities. This integration not only underscores the feasibility of real-time object detection in autonomous driving but also highlights the potential for advanced machine learning models to contribute to the development of safer, more reliable autonomous vehicles.

VI. Conclusion

This paper has presented a comprehensive study on the development and integration of an end-to-end autonomous driving system within a Unity-based simulator, leveraging the strengths of PyTorch, ResNet50 for steering angle prediction, and YOLOv8 for real-time object detection. Our research journey, from the initial exploration of integrated models to the final implementation of specialized models for steering control and obstacle detection, underscores the iterative nature of innovation in autonomous driving technologies.

The experimental results demonstrate the system's capability to navigate complex simulated environments with high accuracy and reliability. The steering angle prediction model, built upon ResNet50, showed promising performance in guiding the vehicle through various driving scenarios. Concurrently, YOLOv8's precision in detecting obstacles and traffic lights played a pivotal role in ensuring the vehicle's safety under dynamic conditions. Together, these components form a robust autonomous driving system that responds adeptly to the challenges of simulated urban navigation.

Reflecting on our exploratory models and methodologies has enriched our understanding of model selection and optimization in the context of autonomous driving systems. Despite encountering challenges, such as data imbalance and the integration of disparate tasks, our findings offer valuable insights into overcoming these hurdles and highlight the importance of a modular approach to system design[10]. Looking forward, the expansion of our system to encompass more diverse environmental conditions and unpredictable scenarios remains a key area of interest. The integration of additional sensory inputs and learning modalities, such as LiDAR or radar, could further enhance the system's perception and decision-making capabilities. Moreover, real-world testing and validation will be crucial in transitioning from simulated environments to practical applications, marking an exciting avenue for future research.

In conclusion, our work contributes a novel approach to autonomous driving research, emphasizing the synergy between deep learning models and simulation technologies. As the field continues to evolve, we anticipate that our findings will inform future developments, paving the way for safer, more intelligent autonomous vehicles.

References

[1] P. S. Chib and P. Singh, "Recent advancements in end-to-end autonomous driving using deep learning: A survey," *IEEE Transactions on Intelligent Vehicles*, vol. 9, no. 1, pp. 103–118, 2024.

[2] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE Access*, vol. 8, pp. 58 443–58 469, 2020.

[3] S. Miao, Z. J. Wang, and R. Liao, "A cnn regression approach for real-time 2d/3d registration," *IEEE transactions on medical imaging*, vol. 35, no. 5, pp. 1352–1363, 2016.

[4] Y. Xie, F. Xing, X. Kong, H. Su, and L. Yang, "Beyond classification: structured regression for robust cell detection using convolutional neural network," in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*. Springer, 2015, pp. 358–365.

[5] G. E. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," *ACM SIGKDD explorations newsletter*, vol. 6, no. 1, pp. 20–29, 2004.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[7] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[8] M. Sohan, T. Sai Ram, R. Reddy, and C. Venkata, "A review on yolov8 and its advancements," in *International Conference on Data Intelligence and Cognitive Informatics*. Springer, 2024, pp. 529–545.

[9] G. Jocher, A. Stoken, J. Borovec, L. Changyu, A. Hogan, L. Diaconu, J. Poznanski, L. Yu, P. Rai, R. Ferriday *et al.*, "ultralytics/yolov5: v3. 0," *Zenodo*, 2020.

[10] L. Chen, P. Wu, K. Chitta, B. Jaeger, A. Geiger, and H. Li, "End-to-end autonomous driving: Challenges and frontiers," *arXiv preprint arXiv:2306.16927*, 2023.