Report of Practical course Simulation-Based Autonomous Driving in Crowded city

Zequan Li

***Abstract*—TThe GNN framework represents a significant advancement in the field of au- tonomous driving. By harnessing the power of imitation learning and real-world demonstrations, this approach enables autonomous vehicles to acquire complex driving skills and navigate urban environments with robustness and adaptability. The findings of this study demonstrate the effectiveness of the self-developed GNN model in learning from diverse real- world scenarios, surpassing the limitations of traditional handcrafted rules and heuristics.the self-developed GNN framework, with its ability to learn from real-world demonstrations, offers great potential for the development of autonomous driving systems. The introduction of autonomous vehicles (AVs) will mark a significant milestone in the advancement of transportation and personal mobility. AVs are anticipated to bring about considerable reductions in accidents and traffic congestion, while also offering economic and environmental advantages. However, numerous challenges need to be addressed before realizing this ideal scenario. One critical element of the self-driving system is the planning module, which currently acts as a bottleneck impeding the progress of autonomous driving.With further research and technological advancements, it has the capacity to revolutionize urban mobility, improve traffic flow, and enhance the overall quality of transportation, ultimately leading to a safer and more sustainable future.**

I. Introduction

With the application of the Internet in various fields, the key to current autonomous driving technology is to stably and accurately predict some rare driving scenarios, that is, in addition to some ordinary driving scenarios, unexpected situations encountered on the road. For the key components of autonomous driving, such as the planning module, it refers to manually calculating the loss function, calculating the minimum error on the basis of traditional path planning, and selecting the trajectory to determine the action plan. However, the disadvantage of this method is that it causes a huge workload for the magician, as engineers not only need to adjust the proportion of terms in the loss function based on different driving scenarios, but also need to recalculate the terms in the design loss function in some rare and sudden scenarios or new environmental positions, which is inefficient. The work of [1] demonstrates how to apply machine learning to achieve autonomous driving, and these strategies are directly obtained from human demonstrations. Compared to traditional manual calculations, applying machine learning has the advantages of being more accurate and effective. However, it cannot guarantee safety during driving, so it cannot be applied in practical scenarios. Therefore, in this practical course, we extend and add data to new fields to obtain direct learning of driving strategies. In practical scenarios, in order to read and collect urban environmental data, a large amount of real-world data collected is simulated and trained. Driven by large-scale datasets, the latest progress in rich middle layer representation [2], high-definition maps, and high-performance perception systems are utilized to achieve the process of urban driving. This progress has led to the development of new methods and impressive performance in motion prediction [3]. Based on the inspiration from the above achievements, we built a simulator from the collected real dataset and high-definition maps, and used previous experience to predict and achieve future driving scenarios. In addition, vectorization is used to reduce computational complexity.

This paragraph of the first footnote will contain the date on which you submitted your paper for review. It will also contain support information, including sponsor and financial support acknowledgment. For example, "This work was supported in part by the U.S. Department of Commerce under Grant BS123456."

The next few paragraphs should contain the authors' current affiliations, including current address and e-mail. For example, F. A. Author is with the National Institute of Standards and Technology, Boulder, CO 80305 USA (e-mail: author@boulder.nist.gov).

S. B. Author, Jr., was with Rice University, Houston, TX 77005 USA. He is now with the Department of Physics, Colorado State University, Fort Collins, CO 80523 USA (e-mail: author@lamar.colostate.edu).

T. C. Author is with the Electrical Engineering Department, University of Colorado, Boulder, CO 80309 USA, on leave from the National Research Institute for Metals, Tsukuba, Japan (e-mail: author@nrim.go.jp).

This paragraph will include the Associate Editor who handled your paper.

II. Preparation of theoretical knowledge

In practical design, it is a challenge to establish safe and accurate routes for most scenarios, so developing a system that can identify and correct trajectories will greatly improve efficiency. Machine-learned planning: Using ML enables designs that do not require manual design rules and can scale with data.

Among them, imitation learning is a type of supervised learning, which can transmit and map sensor data to steering to realize autonomous driving functions such as lane tracking. Reinforcement learning, on the other hand, interacts with the environment and can judge the reward function according to the environment model in road driving.

The graph neural network method can deal with the graph data of the relationship between elements. It is a neural model that captures the dependency between graphs by passing messages between graph nodes. Based on deep learning, this method models a set of nodes and their surroundings, and is used for node classification, prediction, and clustering. The design process for a particular GNN model is: (1) Find the graph structure, (2) specify the graph type and scale, (3) design the loss function, and (4) build the model using the calculation

module. GNN models typically combine various computing modules to build their architecture. The middle section of Figure 2.5 depicts a common GNN model architecture. It involves propagating information within each layer using convolution operators, loop operators, sampling modules, and jump joins. In addition, pooled modules are used to extract advanced information. These layers are usually stacked to improve the quality of the presentation.

In this practical lesson, we employ the above mentioned mimicry learning and self-perturbation for trajectory planning, and we present a baseline ML planning inspired by [1], based on which we successfully train and evaluate its feasibility on our own data set. The results show that the performance of machine learning programming improves significantly with the size of the training data, which is generally consistent with our predictions.

III. Implementation

A. Practical implementation of path planning algorithm

First we introduce a microsimulator, which is based on the collected real-world experience as an approximation of the new driving experience. The simulator plays a crucial role in the strategy learning process by evaluating the performance of the current strategy and facilitating the calculation of the strategy gradient. Using microsimulators, we can effectively assess the impact of current policies on simulated driving scenarios and iteratively improve and optimize policies through a gradient-based approach.

(1)

B. Vector representation

Inspired by Vector Net [3], we can capture the dynamics of multiple agents and the structuring of scenes directly from the vectorized form of multiple agents. In this vector representation, road features have a geographic range and can be represented in geographic coordinates as points, polygons, or curves. For example, lane boundaries are composed of multiple control points that form splines, crosswalks are defined by several points that form polygons, and stop signs are represented by a single point. These geographic entities can be effectively approximated as polylines defined by multiple control points and their associated attributes.

Similarly, we can also capture the spatial and temporal aspects of the agent's movement and relate them to the structuring of the scene. This vectorized representation allows for a unified and comprehensive understanding of multi-agent dynamics and scene context. To integrate these vector sets, we use graph neural networks (gnn). In our method, each vector is treated as a node in the graph, and the node characteristics are defined by the starting position, ending position, and other properties of each vector. These properties might include polyline group ids and semantic tags. By building graphical representations, we can effectively capture contextual information from the trajectories of high-definition maps and other mobile agents.

C. Represents tracks and maps

Annotations from high-definition maps are mainly composed of splines (such as lanes), closed shapes (such as crossing areas), and points (such as traffic lights). These annotations are accompanied by additional attribute information, including semantic labels and the current state (e.g., the color of a traffic light, the speed limit of a road). On the other hand, the trajectory of the agent is represented as a directed spline over time. To approximate these elements in vector form, we follow a sequence-based approach. For map features, we select the starting point and direction, sample key points uniformly from the spline at equal spatial intervals, and connect adjacent key points successively to form vectors.

By choosing a sufficiently small space or time interval, the resulting polyline can be used as an approximation of the original map features and trajectories. This vectorized representation enables efficient processing and analysis of data while retaining basic spatial and temporal information. Inspired by Vector Net [4], our vectorization process establishes a one-to-one mapping between continuous trajectories, map annotations, and corresponding vector sets, even though the vector sets themselves are unordered. This mapping allows us to build a graph representation based on these vector sets, which can be efficiently encoded using graph neural networks.

We normalize the coordinates of all vectors by centering the position of the target agent at its last observed time step. This normalization process ensures that the input node characteristics are invariant for the specific location of the target agent.

IV. Nuplan Datasets

Nuplan's main purpose is to design autonomous driving that ADAPTS to all conditions, excluding heavy rain and night data. And all of Nuplan's data is collected by human drivers, making it easier to obtain data suitable for imitation learning. In the process of data collection, the driver should drive safely, confidently and decisively, so as to obtain more accurate and convincing data.

Data annotation When the data is collected according to the above conditions, Nuplan will enhance the data with metadata such as semantic maps, objects in the scene, traffic light status, and the type of scene observed.

For maps, nuplan offers detailed 2d HD maps that are manually annotated with semantic categories such as roads, sidewalks, crosswalks, lanes, traffic lights, and more. For automatic labeling, it is a lot of work to label every object in every frame of the data set due to the size of the data set. Instead, nuplan uses a machine learning-based system to label data automatically. To this end, the nuplan team has developed a world-class offline awareness system. Previous datasets, such as Argoverse and Lyft, used in-vehicle sensing systems that resulted in low-quality target tracks due to limitations in the vehicle. Nuplan runs online awareness systems on powerful machines in the cloud.

nuplan has developed a new system that can infer the state of traffic lights from the movements of self-vehicles and other participants in a scene. In addition, it must operate offline,

as nuplan looks at the entire trajectory of the vehicle at the intersection, both past and future. This trajectory is then matched to a known "lane connector" in a human-annotated high-definition map. As a result, Nuplan can infer that a particular traffic light is set to green or yellow at a particular time window. This method has the advantage of covering the lanes of all observable vehicles. This allows nuplan to realistically simulate traffic flow in all directions, not just the self-driving direction.

Due to computer performance limitations, we only use Nuplan's minimal data set to train the model. The evaluation criterion for training is MAE(mean Absolute Error), as a measure of the average loss of how similar the model's predictions are to the expert's trajectories. We used the Adam optimizer with a learning rate of 0.0001. The default value of the learning rate is quite high, resulting in the improvement of validation loss stopped early.

V. Docker

For this practical lesson, the above work content needs to be transferred to the website through Docker.

Docker is a project initiated by dotCloud and open source in 2013. The open source is very popular, and its main project has 54k stars on github so far. It is developed using the Go language, based on the Linux kernel cgroup, namespace and AUFS class technology to isolate the process, belongs to an operating system level virtualization technology. Since then, further development has begun to use runC and containerd, further packaging, from the file system to the network interconnection, and then to the isolation of everything, greatly simplifying the creation and maintenance of containers, making Docker more lightweight and faster than virtual machines.

A. Why use Docker?

Like traditional virtual machines, Docker is a virtualization technology, but it has many advantages that virtual machines cannot match: • Continuous delivery and deployment • Faster migration • More efficient use of system resources • Faster startup time • Consistent operating environment • Easier maintenance and expansion The first two are what most developers feel most about: continuous delivery and deployment, and faster migration.

This is a big headache for a lot of developers, and during the development process will encounter this kind of complaint: "Can it run on my computer?" Why wouldn't a different computer work?" • While the presence of requirements such as maven, nodejs package.json, and Python make migration easier, they tend to make it easier to migrate third-party toolkits, but have little effect on systems and development environments. docker ensures a consistent inline environment that can run on multiple platforms, making application migration easier. In addition, docker uses hierarchical storage and mirroring technology, which makes the reuse of repeated parts of applications easier, and can be expanded more based on the basic image, making the maintenance of the system easier.

B. Basic concept

The three concepts most exposed to docker are as follows: • image: image • container: container • repository: repository By understanding these three concepts, you have an understanding of the entire lifecycle of a container. Describe the above three concepts in simple language. Mirroring: This is equivalent to a streamlined file system, such as the official Ubuntu image, which contains only a minimal root file system. Container: A container is a process that has its own root file system, its own network configuration, and its own namespace. Images and containers are like classes and instances in programming; the image is a static definition, while the entity of the image runtime is the container. Repositories: The docker image repository is like the github code repository, when a person builds a project and wants to run it on other computers, they clone the project from the code repository. docker image warehouse is the same, when you build an image, want to use this image on other servers, you need a centralized storage, distribution service, warehouse is such a service. The official image repository is DockerHub, which stores a wealth of images.

C. Dockerfile

Understand some basic concepts of docker and complete docker installation The next step is to learn how to use docker. For most developers, the most central part of the docker process is the Dockerfile. Dockerfile is a text file that contains some instructions. The construction of a docker image is completed through the instructions in the Dockerfile. In other words, to build an image, you need a Dockerfile, and then configure some instruction sets according to your needs. Here is a look at some of the instructions used in the Dockerfile. FROM: specifies the base mirror To customize our image, we need to be based on an image, that is, the basic image, such as Ubuntu, nginx, postgres, mysql, etc., for example, FROM Ubuntu: 16.04, if there is a local Ubuntu base image, the local base image is used. If there is no local Ubuntu base image, the official image repository will be pulled. 16.04 is the image version number. RUN: run the command RUN specifies the commands we need to execute when building the image, such as apt-get install to install some software, pip install to install Python dependency packages, configure the software source, configure the time zone, etc., for example, RUN apt-get install python3.

VI. Conclusion

In summary, We use a transformer based network combined with GNN subgraph embedding method. By combining the strengths of Transformers and GNNs, this approach can effectively capture both local and global dependencies in graph- structured data. It leverages the sequential modeling capabilities of Transformers and the graph-awareness of GNNs to achieve improved performance on various graph-related tasks.

REFERENCES

[1] Bansal, M., Krizhevsky, A., and Ogale, A. "Chauffeur-net: Learning to drive by imitating the best and synthesizing the worst". In: arXiv preprint arXiv:1812.03079 (2018).

[2]Houston, J., Zuidhof, G., Bergamini, L., Ye, Y., Chen, L., Jain, A., Omari, S., Iglovikov, V., and Ondruska, P. "One thousand and one hours: Self-driving mo- tion prediction dataset". In: Conference on Robot Learning. PMLR. 2021, pp. 409– 418.

[3]Liang, M., Yang, B., Hu, R., Chen, Y., Liao, R., Feng, S., and Urtasun, R. "Learning lane graph representations for motion forecasting". In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16. Springer. 2020, pp. 541–556.

[4]Gao, J., Sun, C., Zhao, H., Shen, Y., Anguelov, D., Li, C., and Schmid, C. "Vector- net: Encoding hd maps and agent dynamics from vectorized representation". In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recogni- tion. 2020, pp. 11525–11533.