Fall 12-22-2020

END-TO-END LEARNING UTILIZING TEMPORAL INFORMATION FOR VISION- BASED AUTONOMOUS DRIVING

Dapeng Guo
San Jose State University

END-TO-END LEARNING UTILIZING TEMPORAL INFORMATION FOR VISION-BASED AUTONOMOUS DRIVING

A Project

Presented to

The Faculty of the Department of Computer Science

San Jose State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Dapeng Guo

December 2020

The Designated Project Committee Approves the Project Titled

END-TO-END LEARNING UTILIZING TEMPORAL INFORMATION FOR VISION-BASED AUTONOMOUS DRIVING

by

Dapeng Guo

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSE STATE UNIVERSITY

December 2020

Teng-Sheng Moh, Ph.D.	Department of Computer Science
Melody Moh, Ph.D.	Department of Computer Science
Ching-Seh Wu, Ph.D.	Department of Computer Science

ABSTRACT

END-TO-END LEARNING UTILIZING TEMPORAL INFORMATION FOR VISION-BASED AUTONOMOUS DRIVING

by Dapeng Guo

End-to-End learning models trained with conditional imitation learning (CIL) have demonstrated their capabilities in driving autonomously in dynamic environments. The performance of such models however is limited as most of them fail to utilize the temporal information, which resides in a sequence of observations. In this work, we explore the use of temporal information with a recurrent network to improve driving performance. We propose a model that combines a pre-trained, deeper convolutional neural network to better capture image features with a long short-term memory network to better explore temporal information. Experimental results indicate that the proposed model achieves performance gain in several tasks in the CARLA benchmark, compared to the state-of-the-art models. In particular, comparing with other CIL-based models in the most challenging task, navigation in dynamic environments, we achieve a 96% success rate while other CIL-based models had 82-92% in training conditions; we also achieved 88% while other CIL-based models did 42-90% in the new town and new weather conditions. The subsequent ablation study also shows that all the major features of the proposed model are essential for improving performance. We, therefore, believe that this work contributes significantly towards safe, efficient, clean autonomous driving for future smart cities.

Keywords: Autonomous Driving, Conditional Imitation Learning (CIL), Convolutional Neural Network (CNN), End-to-End Learning, Long Short-Term Memory (LSTM).

ACKNOWLEDGMENT

I would like to express my deepest appreciation to my advisor Dr. Teng-Sheng Moh for his continued guidance and encouragement throughout this project. Without his persistent help, this project would not have been possible.

My completion of this project also could not have been accomplished without the support of my committee members Dr. Melody Moh and Dr. Ching-Seh Wu. I would like to offer my special thanks to them.

Finally, I am extremely grateful to my family and friends for their love, understanding, patient, and continuing support to complete this research work.

TABLE OF CONTENTS

1. INTRODUCTION

Autonomous driving has received increasing attention in recent years. With the rapid development of Deep Learning and Internet of Things (IoT) technologies, autonomous driving evolves from a science-fiction concept to part of the Internet of Autonomous Things (IoAT) systems that are beneficial to smart cities and smart societies [1].

The benefits of autonomous driving may be described below. First, autonomous driving improves road traffic safety. According to the National Highway Traffic Safety Administration (NHTSA), 94% of serious crashes are caused by human factors. Autonomous driving removes human factors from driving, which reduces road traffic accidents to prevent injuries and save lives [2].

Second, autonomous driving reduces economic loss due to road traffic accidents. NHTSA suggests billions of dollars have been lost each year because of road traffic accidents, in the form of lost workplace productivity, loss of life, and decreased quality of life. Since autonomous driving improves road traffic safety, it can greatly reduce such economic loss.

Third, autonomous driving improves efficiency. In dense traffic, a small driver mistake can propagate behind the road and get amplified to cause traffic congestion. Autonomous driving provides smoother vehicle control and drives the vehicles at an optimized speed which can ease the traffic congestion in the current road infrastructure. And eventually, with the help of IoAT networks, autonomous driving vehicles will be able to share their driving information with the network to enable global traffic optimization for better routing and facilitate coordination between vehicles to further improve safety and efficiency.

Finally, autonomous driving enables new applications for vehicles, which brings convenience to daily life. Fully autonomous driving requires no human intervention. This makes self-parking

possible that passengers can be picked up and dropped off without worrying about finding a parking spot. Also, it allows the vehicle to be shared by multiple family members as the vehicle can be remotely summoned. Furthermore, it provides people who are not fit to drive, especially, the old people, and even visually impaired people a way for mobility.

The early attempt at autonomous driving dates to the 1930s, where an electric vehicle was designed to trace the electromagnetic fields generated by radio-controlled devices embedded under the road surface. This vehicle indicates a concrete effort towards autonomous driving. However, it did not generate the intelligence needed for the vehicle to interact with a dynamic physical environment. Its feasibility is also in doubt as it relies on a new type of road infrastructure.

In 1986, Ernst Dickmanns' VaMoRs Mercedes van pioneered in using computer vision for autonomous driving. Multiple generations of autonomous driving systems had been tested in this vehicle, which not only demonstrated the capability of computer vision algorithms but also provided valuable experience in building a small yet powerful autonomous driving system. In 1994, the experience acquired from the 5-ton VaMoRs van was transferred into a VaMoRs-P Mercedes S-class sedan. The VaMoRs-P is also a computer vision-based autonomous driving system that drives like a human which takes both vision and inertial information into account [3]. Furthermore, with two sets of cameras mounted at the front and rear window, the VaMoRs-P can use both the forward and backward vision for object detection and tracking for up to 5 objects within 100 meters in each direction. A minimum of 2 cameras is used in each direction which enables the system to run in both bifocal and multifocal mode. A 4-D approach is used in the VaMoRs-P, which utilizes both spatial and temporal information. The VaMoRs-P is not simply state-of-the-art at its time, its design ideas have found their ways into today's deep learning (DL)

based autonomous driving systems, such as using multiple cameras for visual perception, 3D object detection and tracking. Most importantly, it explores temporal information in the input sequence, which is underutilized in many modern systems.

In 2004, the Defense Advanced Research Projects Agency (DARPA) held the first DARPA Grand Challenge where 15 teams with their autonomous driving vehicles competed to finish an off-road route [5]. Although no team finished the whole route in 2004, this contest and the similar contests in the following years are the early push in accelerating the development of modern autonomous driving technology. With decades of advancement in sensor technologies, computing hardware, and deep learning methods, significant progress has been made towards autonomous driving on public roads, especially the ones utilizing deep learning [4, 5].

Although we are on the edge of entering the era of autonomous driving, numerous vehicle manufactures and autonomous driving solution providers have postponed the deployment of their fully autonomous driving systems. The roadblocks exist in both ethical and technological aspects. Ethically, there are moral dilemmas like the trolly problem, which is not only difficult for the machine to solve, but also hard for humans to reason with. If we cannot create a set of human acceptable moral standards for the machine, the fully autonomous driving vehicles might never be accepted by the public. Technologically, the current autonomous driving vehicles do not meet the criteria of being highly reliable in a dynamic environment. The safety of such vehicles is a major obstacle that needs to be overcome.

The end-to-end learning model proposed in this paper aims to improve autonomous driving, especially in dynamic environments. It is vision-based, accepts navigational commands for controllable routing, utilizes temporal information through recurrent networks to better control

speed and distance, and uses transfer learning on a pre-trained, deep, efficient image module that greatly improves its ability in generalizing to new environments.

The rest of the paper is organized as follows. Section 2 describes the background and related works. Section 3 and 4 cover the proposed method and the proposed model, respectively. Section 5 and 6 present our experiment design and the results, respectively. Finally, Section 7 concludes the paper including future works.

2. BACKGROUND AND RELATED WORKS

There are two major paradigms in deep learning-based autonomous driving: namely, the end-to-end learning paradigm and the modular pipeline paradigm. Although the performance of the models in both paradigms has been greatly improved with the help of deep learning, each paradigm still has its strength and weakness. We first describe the main related works in the end-to-end learning paradigm. Next, for completeness, we also include some major works in the modular pipeline paradigm. Finally, we discuss how the proposed work is built upon the related works.

2.1. End-to-End Learning Paradigm

The end-to-end paradigm uses deep learning to provide a single model that handles the complete self-driving task without any intermediate step. Such models directly map the inputs from sensors to vehicle control signals, without the need to understand any human-defined steps, such as objection detection, path planning, etc. The assumption is that, with the appropriate deep learning model and training method, end-to-end learning will learn the internal features that are most suitable, and optimize all the processing steps simultaneously, which will eventually lead to better performance. With the advancement of deep learning, end-to-end learning models can be

much simpler in structure compared to modular pipeline models while maintaining comparable performance. Moreover, end-to-end learning models are also very flexible, the structures can be easily modified for improvement, and adaption for new features.

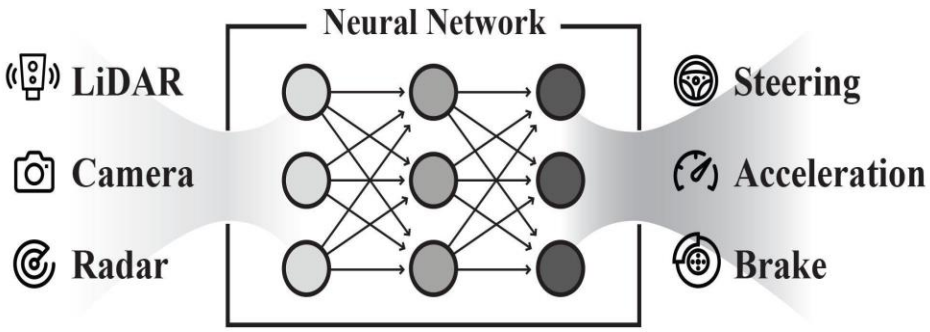

Figure 1. An End-to-End Model

A generalized end-to-end learning model is shown in Figure 1. The model uses LiDARs, cameras, and ultrasonic radars as the main perceptual sensors. The inputs are directly mapped to vehicle control signals using a deep neural network. Ever since Nvidia demonstrated their vision-based end-to-end deep learning autonomous vehicle can successfully drive itself on public roads in long-distance, the end-to-end learning paradigm has received an unprecedented amount of attention. We see great potential in end-to-end learning models, and some selected models with significance are covered here [12-23].

In an end-to-end learning model, we cannot precisely divide the model into different functional modules. However, by feature analysis, we know certain parts of the network are mainly learning image related features and we refer to this part of the model as the image perceptual module [12, 13]. And we also refer to the parts of the network that consumes the output of the perceptual module and mainly be used to generate control signals or other types of predictions as the prediction module.

The end-to-end learning models we surveyed share much in common. For instance, the models surveyed in this section all utilize imitation learning as part of the training process [12-23]. Imitation learning is a type of supervised learning where the model focuses on imitating expert demonstrations. This training method is very intuitive and effective that, since humans can drive, if the model can mimic human actions, it will also be able to drive. In terms of autonomous driving, expert demonstrations are driving footages. We can collect the training data by recording the observations from the cameras, the corresponding control signals from the vehicle, and optionally, the navigational intentions from the driver. By syncing these data, the observations are implicitly labeled. In contrast, another way to train an end-to-end model is using reinforcement learning. However, the action space for autonomous driving is continuous and large, it is inefficient to train the models using reinforcement learning alone as it takes too long to converge [23].

Furthermore, the core structures of the models surveyed are very similar as they all use variations of Convolutional Neural Network (CNN) as the main perceptual network [12-23]. With the recent advancement in computing hardware and labeled dataset, CNN has been widely adopted in pattern recognition tasks. In 2016, Nvidia published the research [12] to demonstrate that by using imitation learning, CNN can also be extended to autonomously steering a vehicle based on RGB images efficiently and effectively. In 2017, Nvidia published follow-up research [13] providing more details and internal feature analysis of their model. The visualization of the activations in the intermediate layer shows, the model not only explicitly learns to recognize driving-related objects that are included in training data, such as traffic signs, road lanes, vehicles, and unmarked road boundaries, it also implicitly learns to recognize driving-related objects that are not included in the training data, such as a construction vehicle exiting a

construction site. The [12, 13] help shape the recent research trend in autonomous driving as we see an increasing number of researchers are utilizing end-to-end approaches trained with imitation learning.

Each end-to-end learning model we surveyed also has a unique contribution. In [12, 13], the authors proposed to collect training images by placing three front-facing cameras: left, middle, right on top of the vehicle. The images captured by the left and right cameras are used to augment the training set. By offsetting the steering angles, such data teaches the car how it should recover itself when driving off-center. This method is also adopted in other research works [15-20, 22] and is proven critical in improving online performance [17]. However, the model in [12, 13] mostly focuses on autonomously lane keeping, it does not accept navigational commands, therefore, the vehicles can only roam aimlessly.

Autonomous driving powered by deep learning also requires a large amount of training data. How to collect large amounts of high-quality training data with high variety and how to test the system in a realistic yet safe environment becomes a problem every autonomous driving project needs to deal with. CARLA is an open-source driving simulation platform specifically designed for the development, training, and validation of autonomous driving systems [15]. It supports customizable environment and road layouts where users can test both urban and highway driving. It provides a flexible set of sensors including RGB cameras, LiDARs, semantic segmentation cameras, and depth sensors to enable more types of autonomous driving systems. It also allows the users to dynamically change the weather and lighting conditions which brings variety in both training and evaluation. CARLA simulator has been successfully utilized in work [15-19, 21-23] to train and validate autonomous driving models with both imitation learning and reinforcement learning.

In [15], the work not only introduces the CARLA simulator and its functions, but also presented a baseline performance comparison of three vision-based models, namely, the modular pipeline (MP), imitation learning (IL), and reinforcement learning (RL) models. This research briefly introduces each of the three models where the MP models use a semantic segmentation network based on ResNet pre-trained on ImageNet and a waypoint planner. The IL model uses a CNN model like [12, 13] with the addition of a speed measurement module. There are four control signal prediction branches selectively activated by navigational high-level commands (HLC). The RL uses A3C style training methods. The result suggests when comparing the IL with MP, the performance of the two approaches are similar in most testing conditions. However, the IL performs better in lane-keeping, especially in a new testing environment, while the MP performs better in avoiding collision with obstacles. When comparing IL with RL, IL largely outperforms RL in most of the cases even if the RL is trained for 12 days. The work suggests this is due to urban driving with continuous action space is much more difficult than problems previously solved by RL, thus the model training does not converge well.

The research [16] proposed conditional imitation learning (CIL) models. This paper focuses on incorporating the HLC into the model so the vehicle can follow a specific route. Two network structures are being compared in this work. The command input model takes HLC as an input and uses one action prediction branch. The command conditional model uses HLC as a switch to activate one of the four action branches.

To improve generalization, image data augmentation is used for both networks where a subset of image transformations including changes in contrast, brightness, tone, as well as the addition of Gaussian blur, Gaussian noise, salt-and-pepper noise, and region dropout are performed with randomly sampled magnitude. The two models are trained with imitation

learning on human driving data collected in the CARLA simulator. Since human driving is mostly driving in the centerline, the authors introduce motion drifts during the data collection process, and only records the recovery process of the human driver to make the dataset more balanced and helps the model to learn how to recover from mistakes.

After the two models are trained with static images, they are tested in an unseen map during the simulation. The average distance per infraction and the success rate of finishing the route is recorded. The results suggest, the two proposed models utilizing HLC outperform the models that do not use the HLC or only use the direction of destination as input. The results also suggest the command conditional model follows the HLC better and achieves a higher route success rate while the command input model makes less driving mistakes and performs better in distance per infraction metric. Although the proposed models take navigational command into account, they still use a simple CNN as the perceptual module, which does not generalize well to unseen environments.

In [17], the result suggests using Mean Absolute Error (MAE) of the trained model during offline evaluation has a higher correlation to the online performance comparing to using Mean Squared Error (MSE), therefore, MAE should be used as the loss function. In [18], depth information is used to supplement the RGB image, and the early fusion scheme which improves upon CIL produces a better result compared to using RGB image alone. There are also works incorporating temporal information which is important for improving the performance [20, 21]. In [22], the author proposed to use pre-trained ResNet as the image perceptual module and utilize speed regularization to further improve the performance. In [23], the author proposed to perform reinforcement learning using the trained weight from imitation learning as the initial weight, which will greatly reduce the time required to converge.

2.2. Modular Pipeline Paradigm

For completeness in this subsection, we describe the main works in the modular pipeline paradigm. The modular pipeline paradigm follows the divide-and-conquer principle which has been widely used in robotic fields. Such a model divides the complex autonomous driving problem into clearly defined stages and subproblems that are easier to solve. Then, each subproblem is solved and optimized separately. Finally, the related subproblems are connected through pipelines with the assumption that the optimal output from the previous stage will be the optimal input for the next stage [5]. It is the most widely adopted approach since the early attempts on autonomous driving, including the VaMoRs-P mentioned above.

Since each module is only responsible for a well-defined subproblem, the modules can be distributed to developers with different expertise and tackled independently. Because the subproblems are well defined, it is easier to understand and explain the inner working and behaviors of the system. This approach also allows each module to be easily modified to meet specific requirements without affecting the performance of unrelated modules. However, human-defined subproblems and features may not be optimal for the overall system. Developing such a system often requires expert knowledge in the field of each subproblem, and the system may get very complex.

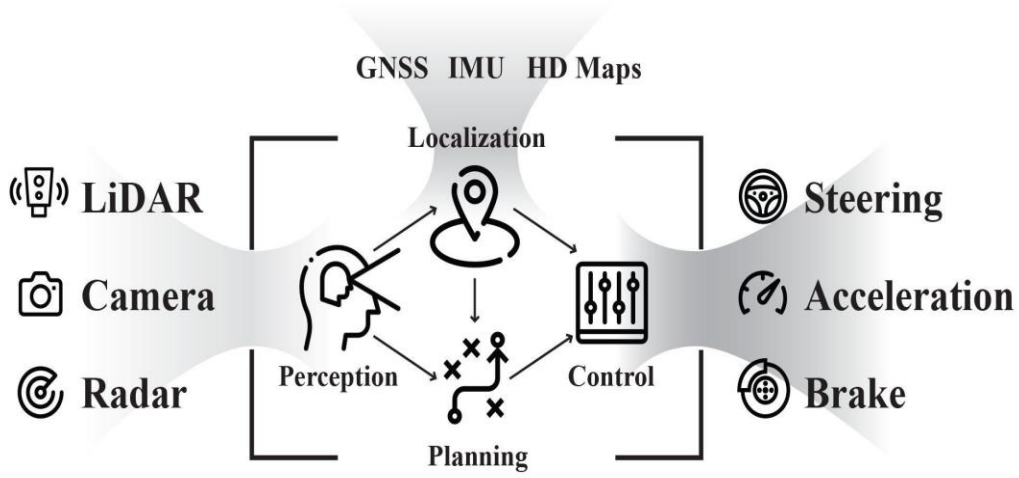

Figure 2. A Modular Pipeline Model

A generalized modular pipeline model is shown in Figure 2. Four major modules reside in four stages in this model. In the first stage, the perception module performs tasks such as objection detection, objection tracking, semantic segmentation on inputs from a fusion of sensors including LiDAR, camera, and ultrasonic radar. In the next stage, the localization module performs its tasks with input from GNSS, IMU, HD Maps, and the output of the perception module. The planning module is in the next stage, which consumes the outputs from both the perception module and localization module for tasks such as trajectory planning. Finally, in the last stage, the control module will turn the output from localization and planning modules into vehicle control signals to steer, accelerate, and brake the vehicle.

Multiple works fall into the modular pipeline paradigm [6-11]. However, due to the high complexity of modular pipeline models, most of the works reviewed only focus on improving upon a specific subproblem, rather than providing a complete pipeline to drive a vehicle.

In [6], the author proposed a computer vision-based direct perception model, which falls in the modular pipelined paradigm but with part of it being like the end-to-end learning paradigm. Instead of recognizing and tracking all driving-related objects to create a representation of its

surrounding environment, and make decisions based on all the information, [6] argues that a subset of this information is sufficient for the driving task, computing all the information increases the complexity. Therefore, a direct perception approach is proposed where meaningful indicators such as the angle of the car, distance to lane mark, and distance to adjacent cars are generated from the input image using CNN following end-to-end learning fashion. Then, based on the indicators, a closed-form controller is used to drive the vehicle. The model is mainly tested on recognizing and predicting the indicators, as well as generating steer control accordingly. The results indicate the model is relatively capable in both offline evaluations with the KITTI dataset and the simulator. However, the major flaw is the indicators this model predicts are handcrafted, in situations where the handcrafted indicators do not exist in the input, the proposed approach cannot work properly.

Both camera and LiDAR have been used as the main perceptual source. Compared to cameras, LiDAR is considered expensive, which makes the autonomous vehicle less affordable. However, LiDAR-based 3D object detection has higher accuracy compared to image-based 3D object detection. The performance gap has been a roadblock that hinders the development of imaged-based autonomous driving. Generally, the performance gap between the two approaches is considered caused by the error of depth estimation grows quadratically with distance in the image-based approach while it only grows linearly with the LiDAR-based approach.

However, [7] suggests differently that the representation of data is a major cause of the performance gap. By converting the image-based depth estimation which is usually represented as an additional channel of the image into a 3D point cloud pseudo-LiDAR representation, the result pseudo-LiDAR data is considered very similar to the ground truth LiDAR data. Furthermore, with the pseudo-LiDAR method, any 3D object detection method designed for

point cloud can be applied to image data, which provides more flexibility. The test results confirm the pseudo-LiDAR approach combined with point cloud-based 3D detection methods outperforms the baseline state-of-the-art image-based 3D detection methods, especially in the bird-eye-view format. However, LiDAR-based methods still outperform pseudo-LiDAR based methods, especially in the far distance cases.

The work [8] proposed a flexible pipeline that applies any existing 2D object detection method to 3D object detection tasks. The state-of-the-art 2D object detection methods have achieved very high accuracy; this proposed pipeline enables us to transfer the success in 2D object detection into 3D object detection. The pipeline is a LiDAR and camera fusion approach where one branch of the pipeline performs 2D object detection on an image to produce a 2D bounding box. Then, the 2D bounding box is projected into the 3D point cloud to select a subset of the points. Since a 2D bounding box in 3D space can contain points from different distances and different objects, the model proposes multiple 3D bounding boxes to be fitted on three generalized vehicle models. The proposal with the highest fitting score will be further fine-tuned by another CNN to form the final bounding box. This pipeline ranks second among the 3D object detection algorithms in the KITTI benchmark at its time. Although the performance is good on the KITTI dataset, there are many more types of vehicles and objects related to driving in the real world. The generalized vehicle model is handcrafted using a 3D CAD dataset, this implies, to make the model able to detect other objects in the real world, we will need to manually create 3D CAD models for many more objects, which is tedious. This will make it difficult to apply this pipeline in a more general term.

In [9-11], the research focuses on improving the speed of point cloud-based object detection. Both [10, 11] use bird-eye-view (BEV) representation of the point cloud data to better explore

the depth information. In [9], it proposes to run a fully convolutional network on a 2D projection of the frontal-view point cloud with dilated convolution to increase the receptive field. In [10], the author increases the number of convolutional layers and combines the residual of different convolution layers to reduce detail loss. The work [11] suggests, directly performing 3D convolution on dense data is slow. Therefore, the work proposed to divide the input into voxels and use a computation mask to make the model focus on objects on the road, reduce the amount of computation needed.

2.3. Building upon and Contrasting with Related Works

Although multiple enhancements have been made upon the basic model of end-to-end learning [12, 13], including those described in Section 2.1, we found none of the models satisfies all our design goal, including an image perceptual module that generalizes well to new environments, accepting HLC for navigation, and utilizing temporal information to make a better decision in a dynamic environment. Therefore, our proposed model is built upon the command input model in CIL [16] which accepts HLC as an input and uses one prediction network to generate control signals. We find using one prediction network makes the model better generalize to the new environment. We incorporate the idea of using pre-trained and deeper CNN [22] to better capture image features and generalize to new environments. However, instead of using ResNet in [22], we use MobileNet which is lightweight and is suitable to be used with recurrent networks. We are also inspired by [20, 21] to explore temporal information using a recurrent network and our choice of recurrent network is LSTM, which has been validated to be beneficial to the driving performance in [21]. Furthermore, we adopt the idea proposed by [22] to perform speed regularization by using a separate speed prediction branch to force the

perceptual better learn speed-related features. Finally, as it is suggested in [17] we use the MAE as the loss function for a stronger correlation between online and offline evaluation results.

3. PROPOSED METHOD: TEMPORAL CONDITIONAL IMITATION LEARNING (TCIL)

In this work, we propose the Temporal Conditional Imitation Learning (TCIL) model. The proposed TCIL model is an end-to-end vision-based autonomous driving model that utilizes Long Short-Term Memory (LSTM) network to explore temporal information, accepts HLC as input to achieve meaningful navigation, incorporates speed prediction regularization to help the image module better learn speed-related features, and use transfer learning to improve generalization. These features are described below.

Vision-based model. The proposed model is vision-based where the primary perception source is a center-mounted RGB camera. Since we are imitating human driving behaviors, we would like the model to perceive the world as close as to the way humans do. By using RGB cameras, we are also able to keep sensor cost and complexity low, which makes the model more feasible to deploy in the real world. Furthermore, processing 2D RGB images require less computing power compared to 3D point cloud data, which compensates for the additional computation introduced by the LSTM network.

End-to-End Paradigm. The model follows the end-to-end learning paradigm. In this work, we improved upon the command input model in [16], and a CNN-LSTM network is used to map image, speed, and HLC inputs directly into vehicle control signals. The overall performance of our model can be directly evaluated in the CARLA benchmark and compared against previous models.

Imitation Learning. The end-to-end learning model is trained with imitation learning, where the model tries to imitate expert demonstrations. As we mentioned, imitation learning is considered very intuitive in the context of autonomous driving, we choose to use imitation learning over other training methods such as reinforcement learning is for its simplicity and efficiency. First, the training data is implicitly labeled by syncing the recorded human control signals. Compared to reinforcement learning, we only need to collect driving footage, then, simply train the deep learning model in a supervised way, without the need to manually craft rewards. Second, driving requires multiple dimension control including steering, brake, and throttle. Since the action space for autonomous driving is continuous and large, reinforcement learning may take a very long time in training and it is not guaranteed to converge well while imitation learning can achieve acceptable performance in hours.

Navigational Command as an Input. Accepting navigational command is a key part to achieve autonomous driving. Intuitively, end-to-end autonomous driving is a mapping from the observation of camera images and measurements to control signals that drive the vehicle, this mapping does not hold if we want to build a model that will meaningfully navigate the road. It not only requires a model to generate one set of control signals to operate the vehicle safely, more importantly, for the same observation, but it also requires the model to generate different sets of control signals corresponding to different navigational intentions. From our preliminary study, we find using HLC in the form of one-hot encoding as input is sufficient to communicate the navigational intention to the model, therefore, we design our model based on the command input model proposed in [16].

Utilizing Temporal Information. The model utilizes temporal information to improve performance. In [6], the author claims that end-to-end models that map input from a single time

step to the output cannot understand the driving task well as the action in the previous time step has an impact on the action for the next time step. Although the perceptual inputs for the current time step may be similar in different scenarios, the action for the next time step can be very different thus it is hard to make a correspondent decision without knowing the previous states. Our proposed model utilizes the LSTM network to explore the temporal information which exists in a sequence of observations. Temporal information is critical to improving the reliability of the models as it helps the model to learn the sense of speed and relative movement. The feature helps the model to make a better judgment in a dynamic environment, especially avoiding crashing.

Deep, Efficient, and Pre-trained Perception Network. In the previous works, the image perception networks are mainly variations of CNN trained from scratch. The depth of such image perception networks is usually shallow, and the size is small. Although it is easy and fast to train such networks, it is also easy to reach their limitations. To improve the performance of a vision-based, end-to-end autonomous driving model, a more capable image perception network is needed. The proposed model uses MobileNet as the image perceptual network. It is a lightweight, efficient variation of CNN that has about 4 million parameters compared to 22 million parameters in ResNet34 used in the previous work [22]. It is designed to run on low-powered devices such as mobile devices with little impact on accuracy. Since we are combining a deeper and more capable image perception network with LSTM, we strive to keep the computation requirement of the image perception low. Furthermore, the MobileNet network is pre-trained on the ImageNet dataset, by using transfer learning, we can further improve the generalization capability of our model. The high efficacy and accuracy of MobileNet make it a suitable choice for our autonomous driving model.

Speed Prediction Regularization and Loss Function. The proposed model incorporates speed regularization, which uses a separate prediction branch for vehicle speed prediction based on the features learned from the image perception module [22]. This setup forces the perceptual module to learn speed-related features, therefore, the overall model better understands the vehicle dynamic and controls the speed without solely relying on the speed input. Our proposed model also adopts using MAE as the loss function during the training instead of MSE. According to [17], using MAE as the loss function yields a stronger correlation between the offline evaluation result and online simulation evaluation result. Since the online evaluation takes a long time to run, we are not able to test out all the trained models to find the one with the best online performance. However, with a stronger correlation between online results and offline results, it gives us more confidence in selecting the best-trained model for further evaluation according to offline evaluation results only.

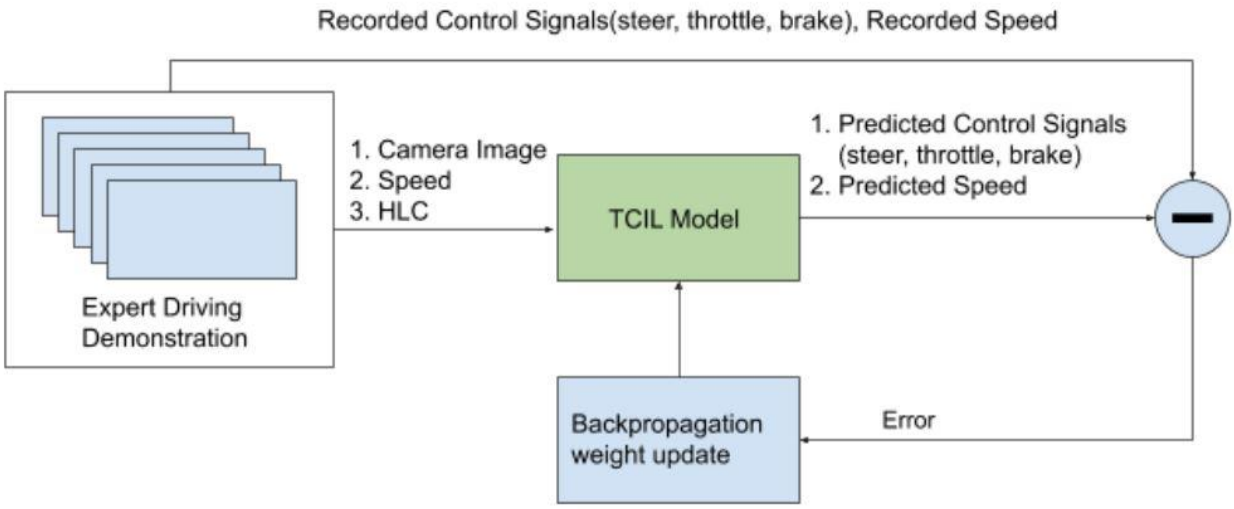

Figure 3. System Structure during Offline Training

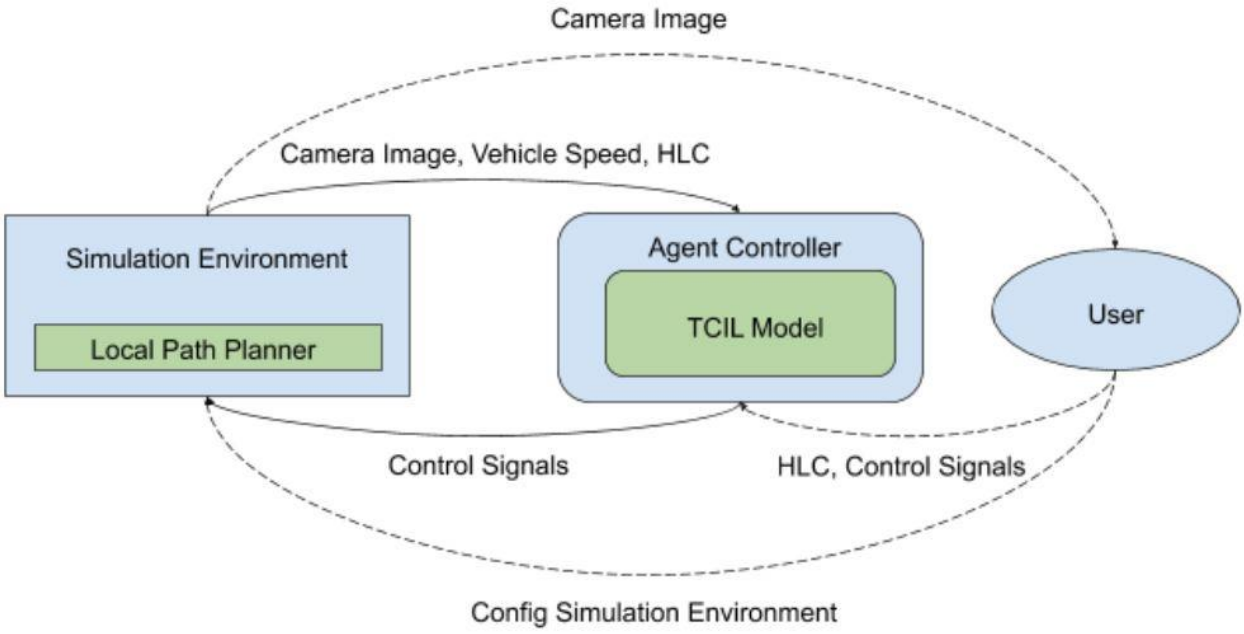

Figure 4. System Structure during Online Evaluation

Overall System Structure. The overall system structure during offline training is shown in Figure 3. Expert driving demonstration contains sequences of observations of camera images and the synced control signals, vehicle speeds, and HLCs. During training, we sample from the original sequences to form short and fixed-length observation sequences to be consumed by the temporal imitation learning model. The model then predicts the control signals of the vehicle with steering, throttle, and brake, as well as the vehicle speed. By using MAE as the loss function, we perform back-propagation on the model to update the weights.

System Structure During Online Evaluation. As shown in Figure 4, we utilize the CARLA simulator to evaluate our model. The solid lines indicate how our model directly interacts with the simulation environment, while the dashed lines indicate how a user can observe and alter the vehicle control and simulation environment with keyboard input.

CARLA Benchmark. CARLA benchmark is used to compare our model with previous works. It has a set of predefined tasks for the agent controller to run, and the simulation

environment will self-configure the weather and lighting conditions according to the benchmark suite. For each simulation timestep, the simulation environment provides an observation including camera image, vehicle speed, and HLC generated from the local path planner to the agent controller. In the agent controller, we use a buffer to record the observations to form an observation sequence. From the sequence, we sample a short and fixed-length sequence the same way as in the training process and use the short sequence to make predictions for speed and control signals. In our experiment, there is a control signal post-processing step, where we filter out the noisy brake signal and alter throttle signals to regulate vehicle speed according to a maximum vehicle speed and the predicted vehicle speed.

4. PROPOSED MODEL: CNN-LSTM NETWORK

The structure of the proposed temporal conditional imitation learning model is shown in Figure 5. The overall model is based on CNN-LSTM network structure, which is often used in sequence prediction tasks such as video sentiment classification. The model is inspired by the command input model in [16] where the HLC is used as a part of the input, and a single action prediction module is used for all HLC types. Furthermore, the model incorporates the speed prediction regularization feature proposed in [22], and the LSTM network to explore temporal information [21].

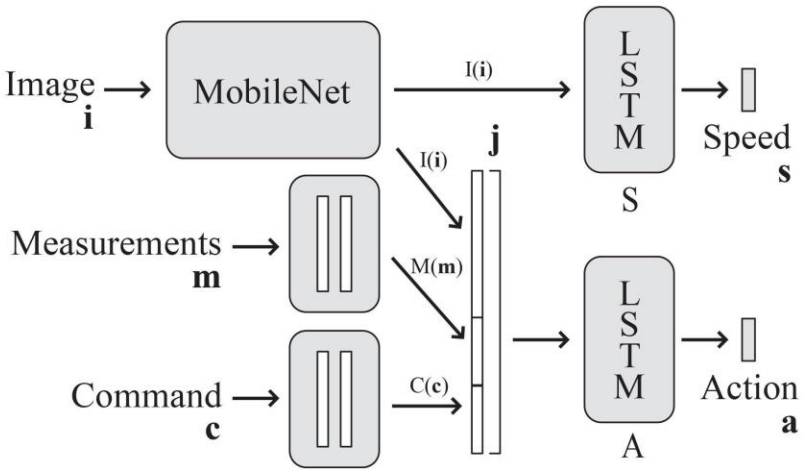

Figure 5. Temporal Conditional Imitation Learning Model

The proposed model includes the following, and described in detail below:

- Three input modules, namely, the image module **I**, the measurement module **M**, and the command module **C**.
- Two prediction branches, including speed prediction branch **S**, and action prediction branch **A**.

Image Module I: Image module **I** is mainly responsible for learning image features. This is one of the most important components of the model. Initially, we tried to use a self-built CNN similar to the one in [16] as the image module, however, we find the self-built CNN does not generalize well with the new conditions, and not even in the training conditions. We find MobileNet works well for our model as it balances efficiency and performance. We choose to use the full-sized MobileNet pre-trained on the ImageNet dataset. All trainable layers are unfrozen during the training; therefore, we can take advantage of transfer learning to let the pre-trained MobileNet learn features related to autonomous driving. In this way, we run a more capable image module on the vehicle itself to improve the reliability, without the need for extremely high-performance computing hardware, or offload the work to the cloud, which will

add a significant amount of latency. The image input **i** is represented as 200 x 88 pixels RGB image format in our experiment.

Measurement Module M and Command Module C: The two modules share the same structure. Two fully connected layers of size 128 are used in our experiment. Currently, the measurement **m** only consists of the speed of the vehicle, thus, it is represented as a scalar value. The command **c** is represented by a one-hot encoding vector, corresponding to four HLC types: following the lane, turn left at the intersection, turn right at the intersection, and go straight. ReLU nonlinearity is used as the activation function.

Action Prediction Module A: The output of the three input modules are concatenated, then fed into the action prediction module **A**. Instead of the fully connected network used in [16], we use an LSTM network with 64 nodes, which allows us to explore the temporal information that exists in input sequences [21]. We organize the input **i**, **m**, **c** into sequences equally sampled from previous timesteps, and apply modules **I**, **M**, **C** in a timely distributed manner. By exploring temporal information, the model can understand the dynamic environment better and make better decisions to avoid crashes and control speed.

Speed Prediction Module S: To further improve vehicle performance in the dynamic world, we incorporate speed prediction regularization [22]. We jointly trained an LSTM speed prediction module **S** connected to the image perceptual module **I**, which forces the image perceptual module to learn speed-related features, thus the overall model will not overly rely on the speed measurement from the input, and it will learn a better sense of speed, which is extremely useful in avoiding accidents. The speed prediction is also used to enforce the speed of the agent vehicle during simulation tests, such as avoiding the vehicle being stopped due to causal confusion.

Temporal Conditional Imitation Learning Model as Controller F: For the TCIL model, at each timestep **t**, we use an input sequence consisting of **n** observations equally sampled from current and previous timesteps, denoted as images $\mathbf{i^n_t}$, speed measurement $\mathbf{m^n_t}$, and HLC $\mathbf{c^n_t}$, to make each action prediction $\mathbf{A(i^n_t, m^n_t, c^n_t)}$ and speed prediction $\mathbf{S(i^n_t)}$. The training dataset consists of such observations and ground truth action pairs recorded from the experts, and is denoted as $\mathbf{D = \{((i^n_j, m^n_j, c^n_j), (a_j, s_j))\}^N_{j=1}}$. The TCIL model acts as a controller **F** and the goal is to find trained weight θ that will minimize the loss:

$$\underset{\theta}{\operatorname{argmin}} \sum_j L(F(i_j^n, m_j^n, c_j^n), (a_j, s_j)) \tag{1}$$

Loss Function: In our work, we use MAE as the loss function, and it is defined as:

$$L = MAE = \frac{1}{n}\sum\left(\lambda|a - a_{gt}| + (1-\lambda)|s - s_{gt}|\right) \tag{2}$$

5. EXPERIMENT

For results reproducibility, in this section, we introduce how the experiment is carried out in terms of the dataset, the hardware, and software setup we used, and explaining the training and evaluation protocols. During our experiment, we have switched between different versions of the learning framework and simulators, which sometimes requires different dependencies to be installed. To reduce the complexity of maintaining different versions of dependencies, our experiments are all carried out in Docker containers. We also share the xhost from the host machine with the docker containers so we can run a GUI application in docker, which is mainly used to drive and observe the agent vehicle during the simulation.

5.1. Dataset

In our work, we use the dataset provided by [16]. The dataset consists of 40GB of expert demonstration including the ones we are interested in including RGB camera images, steer, brake, throttle signals, speed measurements, and HLCs collected from the CARLA simulator. CARLA simulator is implemented in Unreal Engine 4, which provides a realistic recreation of urban driving environments with buildings, roads, weather, lighting, vehicles, and pedestrian traffics. It provides a wide variety of driving environments which is ideal for our experiments. There are also multiple maps provided in CARLA where different street layouts and visual styles are used. In this dataset, the training data is recorded from map Town 1 with three different weather conditions. The map town 2 will be exclusively used for testing. Furthermore, this dataset is consistent with the conditions in the CoRL2017 CARLA benchmark, thus, we can directly compare the performance of our model with other previous works. There are both human and machine driving footage included, and the vehicles are driven at a speed below 60 km/h while avoiding collision with obstacles. Traffic rules such as traffic lights, and stop signs are ignored in the dataset, therefore, our trained model will not be able to follow traffic lights or stop at stop signs.

It is important to use the same sensor setup during the model training and running the benchmark. The camera used in this dataset is mounted at (x=2, y=0, z=1.4) relative to the target vehicle with 15-degree pitch down in terms of the CARLA coordination system. The original images are captured at 15 FPS, have a resolution of 800x600 pixels with a field of view of 100 degrees stored in RGB format, however, the images in the dataset we get are already cropped into 200x88 pixels by removing the top 115 pixels of mostly sky, and bottom 90 pixels of mostly ground and being downsampled to 200x88 pixels. The steering signal is in the range [-1, 1]

which follows the standard CARLA convention without any conversion. The same applies to both throttle and brake signals, however, the range is [0, 1]. The speed measurement is provided in m/s with the range of [0, 25]. The HLC is represented as integers in the dataset, with 2 for the following lane, 3 for turning left, 4 for turning right, 5 for going straight. Further processing is needed during the training for optimal results, which we will talk about in the next section.

5.2. Training

5.2.1. Training Setup

The experiments are carried out on a PC with a Ryzen 3600 CPU, 16 GB memory, and an Nvidia RTX 2070 graphic card with 8 GB of graphic memory. We implement and train our model using TensorFlow 2 machine learning framework. The original dataset was processed by the data generator which is responsible for splitting the data into a training set (80%) and a validation set (20%), generating an input sequence of length five with a sample interval of three for the CNN-LSTM network, scaling the input values including RGB images and speed measurements, and augment the RGB images for better generalization.

For the training parameters, we use Adam optimizer with an initial learning rate of 0.0002 and MAE as the loss function. Since our model is CNN-LSTM based, such a model generally requires a lot more graphic memory compared to a CNN based model, and it also takes longer to train. To fit our model into the 8 GB graphic memory, we use a batch size of 64 data samples instead of 120 data samples which has been seen in [16, 22]. We train the model in an episodic way where in each epoch, all training data is used. We train the model for 60 epochs, where each epoch takes about one hour. The network weights are saved after each epoch, together with the corresponding training and validating errors, which will be used to evaluate the model.

5.2.2. Data Generator

The data generator is a key part of training our model. It has multiple important responsibilities in our experiment; therefore, we will briefly explain how we design and implement our data generator. Our original dataset contains multiple driving video clips with their synced measurements are stored in multiple H5PY data files where each data file contains 200 observations. By examining the observations, we find, the observations are captured at 15 FPS, and the observations from the same video clip can spread across multiple data files. Since we want the input sequence to be meaningful for the model, all observations in the same input sequence should be equally sampled with a fixed time interval from the same video clip, we design the generator to first generate a set of indices indicating all observations that satisfy the condition where from this observation, there are 15 consecutive observations in the same driving video clip. This feature guarantees that when we use the generated indices to access the observations in the files, we will be able to form a sequence of length 5, with a sample interval of 3 that all comes from the same clip. After generating a list of indices, we then shuffle and split the indices list into a training set indices list (80%) and a validation set indices list (20%). The indices can be converted back to the file name and observation number to access the raw data.

Our data generator will also scale or convert the input sequence value where the RGB images are converted from taking the value of [0, 255] to [0, 1] and speed measurement from taking the value of [0, 25] to [0, 1]. The high-level commands provided in the dataset are represented by integers, we convert the four types of HLC into the corresponding one-hot encoding vectors to be better utilized by the model. Since ground truth control signals are already in the range of either [0, 1] or [-1, 1], we leave them in their original forms.

5.3. Evaluation

We evaluate the model with the smallest validation loss on the CoRL2017 CARLA benchmark to get a set of benchmark scores. The CARLA benchmark tool is a module that is built upon the CARLA simulator, and it is designed to evaluate and measure the performance of different driving agents. The CoRL2017 CARLA benchmark uses the same sensor setup as our training dataset and consists of 4 tasks to be performed in a combination of seen and unseen weather and maps. The CoRL2017 CARLA benchmark has been used in multiple previous works; therefore, we can have a direct comparison.

6. RESULTS

We analyze the experiment results in two parts; first with the state-of-the-art models, then an ablation study to demonstrate the effects of different features of our model.

6.1. Comparison to the State-of-the-Art

We start analyzing our results in terms of the success rate on the CoRL2017 CARLA benchmark. In each task of the benchmark, there are multiple tracks for the agent to finish. The success rate measures the percentage of tracks the agent successfully finished on average among different weathers. Here, we report the best result in 5 runs. Our proposed model is referred to as TCIL, and it is compared with the state-of-the-art models including CIL [16], CIRL [23], CILRS [22], AEF [18]. All the reference models follow the branched approach while our model is the only one using the HLC as an input to the model. To better utilize the speed prediction, we use the predicted speed to regulate our agent vehicle speed to prevent the vehicle stops due to causal

confusion, and to prevent the vehicle over speed. We also limit the top speed of our agent to 45 km/h. The results are shown in Table 1.

Table 1. Success Rate % Comparison with State-of-the-Art on CARLA Benchmark.

	Training Conditions					New Town and New Weather				
Tasks	CIL	CIRL	CILRS	AEF	TCIL	CIL	CIRL	CILRS	AEF	TCIL
Straight	98	98	96	**99**	98	80	**98**	96	96	96
One Turn	89	97	92	**99**	94	48	80	92	84	**98**
Navigation	86	93	**95**	92	93	44	68	92	90	**94**
Navigation Dynamic	83	82	92	89	**96**	42	62	90	**94**	88

Since not all related papers provide detailed training methods or trained models, here, we are only comparing the reported success rates with our tested success rate. There are four tasks in the CoRL2017, namely, straight, one turn, navigation, navigation in dynamic traffic. And there are two sets of evaluation conditions we are interested in. The training condition only contains the map and weather that are included in the training data, while the new town and new weather condition only contains the map and weather that never show up in the training data. We interpret the result under training conditions mostly as the model's capability in driving autonomously and the result under the new condition mostly as the model's capability in generalization to the new environments.

Training Conditions

To our surprise, in the training condition, none of the models perform perfectly even in the straight task, which is the easiest one where the AEF model performs the best at 99% while our

model is trailing 1% behind at 98%. In our case, the failure mostly comes from the vehicle being confused by the observation and it stops in place until the episode times out. For the one-turn task, the agent is required to navigate a track including making one single turn at the intersection. The AEF model still takes the lead with a 99% success rate while our model is not far behind at 94%. In navigation tasks, the agent vehicle is required to navigate tracks with multiple turns, including driving straight in the multiple way intersections. The CILRS model takes the lead at a 95% success rate while our model has the second-high success rate at 93%. Finally, in navigation in a dynamic environment, our model performs the best with a 96% success rate while the second highest is at 92%.

Although our model does not top at every task, our model still has advantages over different models. TCIL outperforms the base CIL model in every task, and the improved CILRS model in three out of the four tasks. The reason we directly compare our model with CIL and CILRS is that all three models are trained similarly with imitation learning and take the same kind of input. Our model is based on the CIL model and share much in common with the CILRS model including using a pre-trained image module, a speed branch for speed regularization, and trained with imitation learning only. The AEF model uses additional depth input, therefore, it has a slight advantage over our model in terms of input, while our model still outperforms it in the most difficult navigation in dynamic environment task. Although the CIRL is designed to improve both the driving performance and generalization, and we are only able to beat this model in navigation in the dynamic environment task, we will see the CIRL does not generalize well in new environments. The result shows, our model has an edge in handling complex and dynamic driving environments.

New Town and New Weather Conditions

In the new town and new weather conditions, the results show the generalization capability of the models. In the straight task, most models can keep up with their performance in training conditions except for CIL. CIRL tops at this task with a 98% success rate while our model is at 96%. In one turn task, we see a trend that the models without pre-trained and deeper image modules, including CIL, CIRL, AEF lose performance dramatically. Our model TCIL performs the best at a 98% success rate. In the navigation task, our model tops at 94% with CILRS and AEF keeping up at 90% and 92%. We observe an even larger performance drop for CIL and CIRL, which means both models do not generalize well to new environments, and this trend also applies to the navigation in dynamic environment tasks. In navigation in dynamic environment tasks, the AEF model impressively outperforms the rest at a 94% success rate, while our model is third highest at 88%. We conclude, the pre-trained deeper image network helps the CILRS and TCIL better generalize to new environments in most cases. And with a deeper pre-trained image network and LSTM for decision network, our TCIL tops at two out of four tasks. It shows our model generalizes relatively well to new environments, although we still need to work on the performance and generalization issue in the navigation in dynamic environment tasks.

6.2. Ablation Study

In addition to the comparison with state-of-the-art models, we also want to show the effect of different components of our model. Some of the components we are interested in are using the LSTM network as the prediction module, incorporating speed regularization, and using MobileNet as the image module. To examine the effectiveness of each component, we remove the target component from the complete TCIL, then, train and evaluate the reduced model. In the ablation study, the success rate of the CoRL2017 CARLA benchmark is shown in Table 2.

Table 2. Success Rate % among Ablated Versions.

	Training Conditions				New Town and New Weather			
Tasks	TCIL	TCIL w/o LSTM	TCIL w/o Speed	TCIL w/o MobileNet	TCIL	TCIL w/o LSTM	TCIL w/o Speed	TCIL w/o MobileNet
Straight	98	96	88	**99**	**96**	96	78	94
One Turn	**94**	87	89	90	**98**	86	89	74
Navigation	**93**	84	79	88	**94**	76	78	52
Navigation Dynamic	**96**	88	76	87	**88**	84	70	44

Without LSTM: One key feature of our proposed model is using the LSTM network to better explore temporal information. In the ablation study, we replace the LSTM action prediction branch and LSTM speed prediction branch with fully connected networks in the same structure as in the CIL paper [16]. The benchmark result shows, without the LSTM network, both the driving performance and generalization capability are lowered, especially in the more complex tasks. We also observe that the agent vehicle does not regulate its speed well compared to the TCIL model where the vehicle is more likely to reach the speed limit, and crash in turns due to the vehicle speed being too high. Therefore, we consider the LSTM network has a positive impact on improving the model in both driving performance and generalization capability.

Without Speed Prediction and Regulation: In our proposed model, we incorporate speed regularization by using a dedicated prediction branch to cope with causal confusion where the vehicle will be confused and stop even if there is no obstacle present. In the ablation study, we remove such a speed prediction branch and train the model as usual. The result suggests the model experienced a large performance drop. We also observe that the vehicle will stop when there is no obstacle more frequently, which is accountable for a large portion of the failed cases. We conclude that although speed regularization cannot fully resolve causal confusion, it can greatly reduce the chance the agent stops when it should not. Incorporating speed regularization helps in the overall performance of our model.

Without MobileNet: Lastly, we replace the MobileNet with the CNN used in [16]. We observe a big performance drop in complex tasks, especially in the new town and new weather conditions. We observe many failed cases in bad weather, and the vehicle gets in the wrong lane and sidewalk frequently. The results suggest that the pre-trained MobileNet better captures driving-related features and contributes to both driving performance and generalization capability.

7. CONCLUSION

In this paper, we proposed the Temporal Conditional Imitation Learning model. By exploring temporal information, the proposed model aims at improving both the driving performance and generalization capability, since it can have a better sense of speed and relative movement, resulting in better control of the vehicle. We achieve this by using the CNN-LSTM structure. To cope with the phenomenon where the agent vehicle stops when no obstacle presents, known as causal confusion, we incorporate speed regularization which utilizes a separate speed prediction branch. By having the speed prediction, we are also able to regulate the vehicle speed, which

further reduces the chance the vehicle stops due to causal confusion, or over speed. Finally, to better capture the image features, we use MobileNet pre-trained on ImageNet data as the image perceptual module; MobileNet not only captures image features better, but it is also more efficient compared to image models such as ResNet. Since we are using the CNN-LSTM network, the efficiency of the image network is extremely important as the image module will need to process multiple times the image data compared to regular CNN models. All these features have been shown to have a positive impact on autonomous driving performance through our ablation study. And our models are competitive compared to the state-of-the-art models according to the CARLA benchmark and outperform the other models in several tasks, especially in navigating dynamic environments, and in the new town and new weather condition.

There are also places we can improve upon. Future work first includes improving dataset. The one used mostly consists of footage driven by a controller using waypoints in the Unreal Engine. It would be beneficial to add more human driving footage, including handling more corner cases such as recovering from mistakes and avoiding perpendicular traffic at the intersection. We observed that the camera's field of view is not wide enough, as many times, the vehicle cannot see the entrance and exit of the turn, which should be enhanced in the future. Furthermore, utilizing depth information can greatly improve the overall system performance [18], this consists of collecting training data including depth image and properly exploring such information. Finally, imitation learning has its limitations, one cannot collect data for all possible cases for the model to learn. Using reinforcement learning to improve the generalization to a more variety of driving scenarios would be another promising future direction.

References

1. A. C. Marosi, R. Lovas, Á. Kisari and E. Simonyi, "A novel IoT platform for the era of connected cars," 2018 IEEE International Conference on Future IoT Technologies (Future IoT), Eger, 2018, pp. 1-11, doi: 10.1109/FIOT.2018.8325597.

2. National Highway Traffic Safety Administration (NHTSA), "Automated Vehicles for Safety.", 15 June 2020, www.nhtsa.gov/technology-innovation/automated-vehicles-safety.

3. E. D. Dickmanns et al., "The seeing passenger car 'VaMoRs-P'," Proceedings of the Intelligent Vehicles '94 Symposium, Paris, France, 1994, pp. 68-73, doi: 10.1109/IVS.1994.639472.

4. Q. Rao and J. Frtunikj, "Deep Learning for Self-Driving Cars: Chances and Challenges," 2018 IEEE/ACM 1st International Workshop on Software Engineering for AI in Autonomous Systems (SEFAIAS), Gothenburg, 2018, pp. 35-38.

5. J. Janai, "Computer Vision for Autonomous Vehicles: Problems, Datasets and State-of-the-Art", submitted to ISPRS Journal of Photogrammetry and Remote Sensing, April 2017, 50 pages.

6. C. Chen, A. Seff, A. Kornhauser and J. Xiao, "DeepDriving: Learning Affordance for Direct Perception in Autonomous Driving," 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, 2015, pp. 2722-2730, doi: 10.1109/ICCV.2015.312.

7. Y. Wang, W. Chao, D. Garg, B. Hariharan, M. Campbell and K. Q. Weinberger, "Pseudo-LiDAR From Visual Depth Estimation: Bridging the Gap in 3D Object Detection for Autonomous Driving," 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 2019, pp. 8437-8445, doi: 10.1109/CVPR.2019.00864.

8. X. Du, M. H. Ang, S. Karaman, and D. Rus, "A General Pipeline for 3D Detection of Vehicles," 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, QLD, 2018, pp. 3194-3200, doi: 10.1109/ICRA.2018.8461232.

9. K. Minemura, H. Liau, A. Monrroy and S. Kato, "LMNet: Real-time Multiclass Object Detection on CPU Using 3D LiDAR," 2018 3rd Asia-Pacific Conference on Intelligent Robot Systems (ACIRS), Singapore, 2018, pp. 28-34, doi: 10.1109/ACIRS.2018.8467245.

10. B. Yang, W. Luo and R. Urtasun, "PIXOR: Real-time 3D Object Detection from Point Clouds," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, 2018, pp. 7652-7660, doi: 10.1109/CVPR.2018.00798.

11. M. Ren, A. Pokrovsky, B. Yang and R. Urtasun, "SBNet: Sparse Blocks Network for Fast Inference," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, 2018, pp. 8711-8720, doi: 10.1109/CVPR.2018.00908.

12. M. Bojarski et al., "End to End Learning for Self-Driving Cars", preprint submitted to arXiv, Nvidia, 25 Apr 2016.

13. M. Bojarski, P. Yeres, A. Choromanska, K. Choromanski, B. Firner, L. Jackel, U. Muller, "Explaining How a Deep Neural Network Trained with End-to-End Learning Steers a Car", Nvidia, 25 Apr 2017.

14. Z. Chen and X. Huang, "End-to-end learning for lane keeping of self-driving cars," 2017 IEEE Intelligent Vehicles Symposium (IV), Los Angeles, CA, 2017, pp. 1856-1860, doi: 10.1109/IVS.2017.7995975.

15. A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez ,and V. Koltun, "CARLA: An Open Urban Driving Simulator." 2017 Proceedings of the 1st Annual Conference on Robot Learning. 1-16.

16. F. Codevilla, M. Müller, A. López, V. Koltun and A. Dosovitskiy, "End-to-End Driving Via Conditional Imitation Learning," 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, QLD, 2018, pp. 4693-4700, doi: 10.1109/ICRA.2018.8460487.

17. F. Codevilla, A. López, V. Koltun, A. Dosovitskiy, "On Offline Evaluation of Vision-Based Driving Models. " In: Ferrari V., Hebert M., Sminchisescu C., Weiss Y. (eds) Computer Vision – ECCV 2018. ECCV 2018. Lecture Notes in Computer Science, vol 11219. Springer, Cham. https://doi.org/10.1007/978-3-030-01267-0_15

18. Y. Xiao, F. Codevilla, A. Gurram, O. Urfalioglu and A. M. López, "Multimodal End-to-End Autonomous Driving," in IEEE Transactions on Intelligent Transportation Systems, doi: 10.1109/TITS.2020.3013234.

19. Q. Wang, L. Chen, B. Tian, W. Tian, L. Li, and D. Cao, "End-to-End Autonomous Driving: An Angle Branched Network Approach," in IEEE Transactions on Vehicular Technology, vol. 68, no. 12, pp. 11599-11610, Dec. 2019, doi: 10.1109/TVT.2019.2921918.

20. L. Chi and Y. Mu, "Learning End-to-End Autonomous Steering Model from Spatial and Temporal Visual Cues," in Proceedings of the Workshop on Visual Analysis in Smart and Connected Communities (VSCC '17). Association for Computing Machinery, New York, NY, USA, 9–16. DOI:https://doi.org/10.1145/3132734.3132737

21. H. Haavaldsen et al., “Autonomous Vehicle Control: End-to-End Learning in Simulated Urban Environments.” In Communications in Computer and Information Science Nordic Artificial Intelligence Research and Development, 2019, pp. 40–51.

22. F. Codevilla, E. Santana, A. Lopez and A. Gaidon, "Exploring the Limitations of Behavior Cloning for Autonomous Driving," 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea (South), 2019, pp. 9328-9337, doi: 10.1109/ICCV.2019.00942.

23. X. Liang, T. Wang, L. Yang, E. Xing, "CIRL: Controllable Imitative Reinforcement Learning for Vision-Based Self-driving, " In: Ferrari V., Hebert M., Sminchisescu C., Weiss Y. (eds) Computer Vision – ECCV 2018. ECCV 2018. Lecture Notes in Computer Science, vol 11211. Springer, Cham. https://doi.org/10.1007/978-3-030-01234-2_36