Sequential Masking Imitation Learning for Handling Causal Confusion in Autonomous Driving

Huanghui Zhang[1] and Zhi Zheng[1,2](✉)

[1] College of Computer and Cyber Security, Fujian Normal University, Fuzhou 350117, China
zhengz@fjnu.edu.cn

[2] College of Control Science and Engineering, Zhejiang University, Hangzhou 310027, China

Abstract. Training agents for autonomous driving using imitation learning seems like a promising way since its only requirement is the demonstration from expert drivers. However, causal confusion is a problem existing in imitation learning, which is that with more features offered, an agent may perform even worse. Here, we aim to augment agents' imitation ability in driving scenarios under sequential setting, by a novel method we proposed: Sequential Masking Imitation Learning(SEMI). First, we train a Vector Quantised-Variational AutoEncoder(VQ-VAE) to encode a sequence of images into a latent representation with discrete codes. After that we deploy several masks on the encoded images, the masks here will randomly hide some semantic objects in the encoded images. Finally, we design the behavior clone network as a predictor of expert action, using an encoded and masked image sequence as input, encouraging the network to make expert-like predictions when some partition of information about the environment is missing. The masking procedure in SEMI helps the imitator identify the contribution of each encoded feature to the expert's prediction. We demonstrate that this method could alleviate causal confusion in driving simulation by deploying it to the CARLA simulator, and compared it with other methods. Experimental results show that SEMI can effectively reduce confusion in autonomous driving. The agent trained with SEMI method reduces the collision rate by 45% compared to methods without masking procedure, and obtain the highest average survival timesteps among competing methods.

Keywords: Causal confusion · Invariant feature learning · Imitation learning

1 Introduction

Benefiting from the development of perception and computing capabilities, autonomous driving nowadays is able to build a model that can deal with complex situations. Recent research that can transform raw sensor data into a form that helps a model better understand its surrounding, like [1] can translate images captured by a car's front RGB camera, into an overhead map or bird's-eye view images. Combining those progress together, models relying on pre-processed data can perform better than traditional end-to-end models that use raw sensor images as input.

B. Xin et al. (Eds.): IWACIII 2023, CCIS 1931, pp. 200–214, 2024.
https://doi.org/10.1007/978-981-99-7590-7_17

Previous approaches in autonomous driving often took reinforcement learning(RL) methods to train a policy model to get higher rewards. RL methods highly rely on the reward function proposed by researchers, while designing a reward function to guide our model to do what we really want could be difficult [2], especially when the intent of the action is hard to express with mathematical expressions. RL methods also require interacting with the environment while training, which is difficult and unsafe. In addition, RL methods often require a time-consuming reward maximization procedure to have their agents performance guaranteed.

Traditional imitation learning(IL) methods can learn a strategy directly without actual interaction with the environment, using only expert samples organized as state-action sets. Using networks like deep convolutional neural network (CNN) model combine with IL methods like Behavioral Cloning(BC) could produce seemingly well results. However, IL often suffers from a problem for a long time: "causal confusion" [3]. Due to the distribution differences between training and testing states, a imitator may misidentify the real cause of an expert's action and rely on suspicious correlates to make decisions. Effects of the problem showed more obvious and severe when the state information given by the environment is plentiful and the scenario is complex. This confusion led to models learned by IL methods like behavioral cloning, performed poorly when it meet new states different from training samples.

Considering these problems above, we aim to organize the vehicle's surrounding states into semantic bird-eye view images in sequence, design an imitation learning method that encourages imitators to avoid causal confusion and make predictions based on important features.

In this paper, we provide the following contributions.

- We combine semantic bird-eye view images [4], with Object-aware REgularizatiOn (OREO) method from [5] in sequential setting, propose our Sequential Masking Imitation Learning(SEMI) method which show as a robust method for addressing causal confusion in a high-fidelity driving simulation.
- We implement SEMI and train an end-to-end model that can output direct continuous control commands to a vehicle, lead it to follow a pre-set route, unlike many models that rely on much hand-engineered involvement.
- Finally, we deploy our SEMI method on the CARLA simulator [6], test it with several environments that it is unfamiliar with, compare its performance with other methods, analyze its advantages, and demonstrate its consistent ability to perform an expert-like strategy. Experimental results show that the agent trained with SEMI method reduces the collision rate by 45% compare to methods without masking procedure, and obtain the highest average survival timesteps among competing methods.

The remainder of the paper is organized as follows. We review related work in Sect. 2. Our proposed SEMI method and its theory description are presented in Sect. 3. In Sect. 4, we describe the network structure for SEMI in the experiment and give a brief introduction about the simulation setting and the process of collecting data. In Sect. 5, we present the evaluation of our proposed method and compare it with several competing methods. Finally, we conclude the paper in Sect. 6.

2 Related Work

2.1 Pipelines of Autonomous Driving

Many autonomous-driving companies utilize a traditional engineer stack, where the driving problem is divided into perception, prediction, model planning, and control [7]. Solving these sub-tasks requires much hand-engineered involvement, which could be hard and lead to sub-optimal overall performance. Using discrete variables as the operation command is also a prevailing practice in the field of autonomous driving since this can simplify the driving task into a classification problem. However, the prediction of discrete variables that represent the high-level command still needs its downstream module to execute the concrete action. Also, using a finite number of discrete values as the direct control command may be harmful to a driving system's flexibility since operations like setting steering angle and acceleration often require precise control.

End-to-end driving, on the other side, known as mapping raw images to certain control commend, seems like a more promising way to autonomous driving. However, models rely on raw data require tons of samples that could cover a variety of situations like different weathers and different light intensities. Considering the weakness of common end-to-end driving, researchers put much effort to process raw data to improve models' robustness.

Our approach belongs to the end-to-end side of the autonomous driving spectrum, using pre-processed data to simulate the real driving task via imitation learning.

Follow what Chen et al. proposed in [4], we organize our input representation in the form of semantic bird-eye view image by using LiDAR and RGB camera sensors and built-in road maps, as shown in Fig. 1(a).

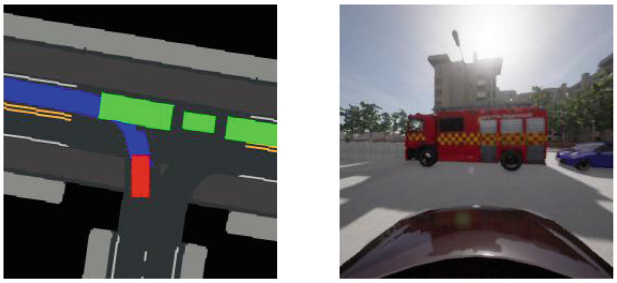

(a) Bird eye view image (b) RGB camera image

Fig. 1. (a): A sample of bird-eye view images. The red rectangle represents the vehicle controlled by the model, and the green rectangles represents other vehicles. The road painted in blue represents the planned route. (b): The corresponding front camera image of (a). (Color figure online)

2.2 Confusion in Imitation

One branch of IL can be called "Inverse Reinforcement Learning(IRL)". Ever since Ng et al. [8] created the notion of IRL, there have been efforts to guide models in learning the intention behind an expert's action. IRL methods [2,9,10] intend to infer a reward function that can explain an expert's trajectory. However, inferring a reward function could be time-consuming when solving iteratively Markov decision processes(MDP) [2,11], and training a policy network often require interaction with environment, which could be dangerous when applying it to the real world.

Behavioral cloning is the other branch of IL, known as learning only through expert's demonstration [12]. We choose BC because it shows no requirement to interaction with the environment. While learning through only datasets seems stunning fashion, the causality problem in IL can not be ignored due to the distributional shift and the complex state information. When de Haan et al. [3] summarized causal confusion in imitation learning, they proposed to fix it with "Expert query" or "Policy execution", but those seem unpractical when it comes to driving in the real world. Katz et al. [13] use causal reasoning to construct an explanation for an expert's action, and generate a executing plan based on this explanation to carry out the expert's goal. Although this approach can explain an action from an expert properly, it requires the domain authors to enumerate direct causal associations to infer indirect causal relation, which is hard to implement when carrying out a complex job like driving. Hence, we consider methods that could address causal confusion indirectly, like randomly dropping/erasing units from input features to regularize policy [14–16].

Inspired by previous works [3,5,17], we encode raw bird-eye view images into discrete codes through a VQ-VAE [17], organize them in sequence as model input, train the model by BC, and regularize them by randomly masking out semantically similar objects.

Although Park et al. [5] did mention that masking semantic objects from a sequence of observation can be effective in practice, their efforts mainly focused on simple environments with clear goals like Atari games. Here we combine these methods and modify them to reduce the complexity of driving tasks, and address the causal confusion in end-to-end driving simulation.

3 SEMI Methodology

3.1 Semantic Encoder

The semantic encoder is the first part of our SEMI network. By training a VQ-VAE network, we make use of its encoder and vector quantizer to manage a bird-eye view image x which represents a state st [17], into discrete latent representations that can be seen as semantic representations.

VQ-VAE defines a latent embedding space $e \in \mathbb{R}^{K \times D}$, where K is the size of the discrete latent space, and the D is the dimension of each latent embedding vector e_i. The encoder map x into its latent representation $z_e(x)$, where $z_e(x)$ is a set of latent variables. Vector quantizer quantizes $z_e(x)$ to discrete representations $z_q(x)$. Decoder is another part of VQ-VAE, aiming to reconstruct x from $z_q(x)$. Decoder and encoder share the same cookbook $C = \{e_k\}_{k=1}^{K}$ of prototype vectors learned through training. Therefore, the objective of training a VQ-VAE can be formed as minimizing the following term:

$$L_{VQ-VAE} = \log p(x|z_q(x)) + ||sg[z_e(x)] - e||_2^2 + \beta||z_e(x) - sg[e]||_2^2, \quad (1)$$

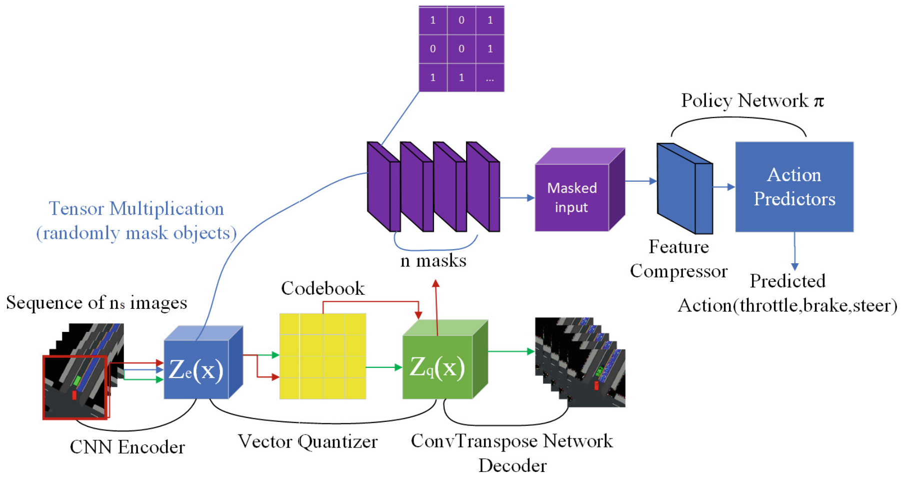


Fig. 2. Overview of our SEMI model. Green arrow represents the process of training VQ-VAE network. Red arrow represents the process of constructing n masks with the first image from sequence. Blue arrow indicates the process of training policy network using masked variable map as input. Notice that once the VQ-VAE trained, we do not need its decoder for imitation learning. (Color figure online)

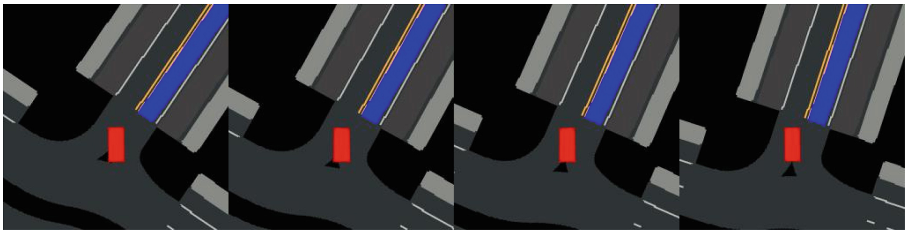
where sg is the abbreviation of stop gradient operator, and β here is a weight on a commitment loss. We can divide Eq. (1) into three components: reconstruction loss, quantizer loss, and commitment loss. The reconstruction loss optimizes the encoder and decoder. The quantizer loss optimizes the vector quantizer so that it can bring codebook representations closer to the encoder outputs $z_e(x)$. The commitment loss is weight by β to make sure the encoder commits to an embedding and its output does not grow, as described in [17], the resulting algorithm is quite robust to β, so we use $\beta = 0.25$ in our experiments.

3.2 Masking Semantic Objects in Sequential Setting

After training a VQ-VAE network, we exact its encoder and vector quantizer for further use. Here, we first reorganize all bird-eye view images from training dataset with a total length n_t, into n_t sequences that each sequence contains n_s images(c^i represents images from $(i - n_s + 1)$th to ith, and x^i_j represents the jth image of c_i), and denote those sequences as $\{c_i\}_{i=1}^{n_t}$.

From then on, each time when we sample from dataset, we take batch B of sequences as our SEMI model inputs. From our intuition, expert demonstrations organized in sequence can help better reveal an expert's intention in a certain environment and show relations between state and action. However, training a model using samples in the form of sequence, often meet the over-fitting problem. We aim to ease it by randomly masking semantic objects.

Using the mask method from OREO [5], we can produce n masks for every sequence c, where n is a hype parameter pre-defined. By setting the drop probabil-


(a) Original images

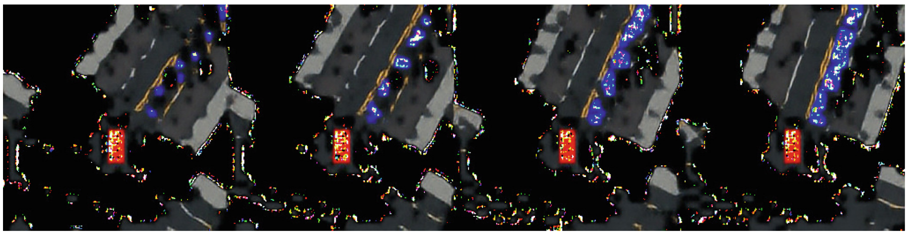

(b) Masked images

Fig. 3. (a): A sequence of bird-eye view images. (b): A sequence of images reconstructed with the masked latent representation of images from (a). The colored specks in reconstructed images are produced by the decoder during the decode phase, since the masking operation may perturb the latent representation and lead to unstable reconstruction of images. Recall that the red rectangle represents the ego vehicle, and the blue line represents the guidance line. Although the images in (b) are hard to interpret after the randomly masking operation and reconstruction, we can still see that the operation randomly masks the road behind the ego vehicle in the latent representation, while the ego vehicle and the guidance line remain. (Color figure online)

ity p, we sample K binary random variables $b_k \in 0, 1, k = 1, 2, ..., K$ from a Bernoulli distribution with probability $1 - p$. We encode the image sequence c_i into the form of latent variables $z_e(c_i)$ using encoder, then construct n masks $m = (b \times z_{q_k}(x_0^i))_{k=1}^K$ for the sequence c^i, based on the embedding vectors produced by vector quantizer using the sequence's first image x_0^i. After constructing masks, we make tensor multiplication between every $z_e(c_i)$ and masks corresponding to c_i, this will output n new sequences corresponding to c_i, containing $n \times n_s$ masked sets of discrete latent variables.

Conclude the masking process described above: we randomly put several masks to drop some semantic objects from sequential images' discrete latent variables, and produce n new sequences with those variable sets.

Figure 3 is a visualization of masking semantic objects in a sequence of images, we can use it as an illustration of the masking idea. The colored specks in reconstructed images are produced by the decoder during the decode phase, since the masking operation may perturb the latent representation and lead to unstable reconstruction of objects' edges after decoding. As we can see in Fig. 3(b), a random mask is computed and deployed for this particular sequence. Latent variables that correspond to the road texture behind the ego vehicle are masked while the ego vehicle and the guidance line remain. So in the view of human drivers, the masked images can offer nearly the same information as the original images, since the texture of the passed road has little effect on the current action.

By applying several random masks for each sequence, we can examine whether the downstream policy network can still perform well when certain parts of semantic objects as input are masked. With this masking process, the downstream policy network is encouraged to make predictions without relying on features that may lead to confusion, and focus on important features that affect drivers' decisions.

We use several random masks to alleviate the causal confusion in the following theory. If a policy network (the imitator), denote it as $\overline{\pi}$, relies on a set of wrong features(e.g., the road texture behind the ego vehicle, positions of vehicles behind ego vehicle, or the positions of vehicles in the opposite lane) to make its prediction, then when some of these features are masked, the optimizer of $\overline{\pi}$ will update the weight of $\overline{\pi}$ to guide it to make a prediction based on the rest features due to the vast loss computed from the failed prediction.

It is worth noticing that, generally, the number of features that can cause causal confusion and still help $\overline{\pi}$ to achieve well performance in training is less than the number of features that an expert takes into consideration for prediction. In other words, many features in the environment affect the expert's decision, while only a small amount of other features are proxies of the expert's action (features that induce causal confusion). So if these proxy features have been randomly masked in some epoch during training, their effects are diminished and the $\overline{\pi}$ is forced to use other features to predict, therefore these proxy features are no longer consistent shortcuts for $\overline{\pi}$ to make predictions in the training phase. Therefore, the masking process can guide the downstream policy network to have a better understanding of the role of each feature in the prediction tasks.

3.3 Behavior Cloning with Imbalanced Dataset

After masking semantic objects, we connect the n new variable sequences into n maps, each map contains all variables sets of a sequence, and we denote those maps as $\{t_o^i\}_{o=1}^{n_s}$. We take $\{t_o^i\}_{o=1}^{n_s}$ and their corresponding action a_i as our SEMI policy network's input, notice that action a_i is a single control command captured at the time of c_i ended.

With our experience from real-world driving and observation from the simulator, one phenomenon can be learned is that the brake and steer are less triggered compared to the throttle. This phenomenon leads to a situation that is similar to the class imbalance in classification tasks, where the difference between classes in the dataset is imbalanced. Models learned from imbalanced datasets directly, usually show limited performance in generalization.

To improve the performance of our policy network in rare events where brake/steer is required, we first use the undersampling technique in our dataset. By using all of the rare events and reducing the number of abundant events, we keep the ratio of rare events to abundant events at about 1:3 to amplify the importance of braking and steering.

The structure of our policy network can be divided into two parts: feature compressor and action predictors. The compressor receives the map of variables sets and integrates features for predictors. We deploy three action predictors for three parts of action prediction: throttle, brake, and steer, and these predictors will output three scalars as their estimation of expert action, noted as $\hat{throttle}$, $\hat{brake}$, and $\hat{steer}$. The feature compressor and each action predictor are all Multi Layer Perceptrons(MLP).

The input features are first feed into the feature compressor and run through several linear layers and activation layers, then we obtain the compressed features as the output of the feature compressor. After that, the compressed features are passed into each action predictor that outputs a scalar as its prediction. In the deployment stage, the action will be the combination of the three scalars output from these predictors.

For each set of N samples, the policy network π can be optimized by computing and minimizing the mean-squared error(MSE) between action $[\hat{throttle}, \hat{brake}, \hat{steer}]$ predicted by π, and the real expert action$[throttle, brake, steer]$, following the term below:

$$\begin{aligned} L_{BC} =& \frac{1}{N}\sum\left(\hat{throttle} - throttle\right)^2 + \frac{1}{N}\sum\left(\hat{brake} - brake\right)^2 \\ &+ \frac{1}{N}\sum\left(\hat{steer} - steer\right)^2 . \end{aligned} \tag{2}$$

Algorithm 1: Process of SEMI

```
Initialize encoder, vector quantizer, decoder, policy $\pi$ randomly;
Define batch-size $B$, drop probability $p$, num-mask $n$, sequence length $n_s$;
while not converged do
    Sample batch of state images $X \sim$ demonstration;
    foreach $x$ in $X$ do
        Encode $x$ into $z_e(x)$ using encoder;
        Quantize $z_e(x)$ into $z_q(x)$ using quantizer;
        Decode $x$ from $z_q(x)$ with decoder;
    Optimize parameters by minimizing $L_{VQ-VAE}$ from Eq.1;
end while
Delete decoder;
Organize demonstration images into sequences of images;
while not converged do
    Sample batch of sequences of state images and corresponding actions $(C, A) \sim$
    demonstration;
    foreach $(c_i, a_i)$ in $(C, A)$ do
        Encode image sequence $c_i$ into $z_e(c_i)$ using encoder;
        Quantize the encoder's output of first image $z_e(x_0^i)$ into $z_q(x_0^i)$;
        Construct masks $\{m_i\}_{i=1}^{(n_s \times n)}$ based on $z_q(x_0^i)$;
        Mask $c_i$ to new maps $\{t_o^i\}_{o=1}^{n_s}$;
        Take $\{t_o^i\}_{o=1}^{n_s}$ as input of $\pi$ to predict action $\hat{a}_i$;
    Optimize parameters by minimizing $L_{BC}$ from Eq.2;
end while
```

Table 1. Hyper parameters of our model.

Hyper parameter	Value
Learning rate:	1e-7
Embedding dimension:	16
Num_embeddings:	128
Num_masks:	4
Sequence_length:	4
Mask_prob:	0.6
Batch_size:	32

4 Experiment

4.1 Network Structure

As shown in Fig. 2, the whole SEMI model can be divided into two parts, the VQ-VAE network and the policy network.

The encoder of VQ-VAE consists of several convolutional layers to amplify the input's channel from 3 to the dimension of embedding and compress information about input. The vector quantizer calculates the discrete latent variable with respect to the output of decoder by finding the nearest embedding vector, then outputs it's index. The decoder contains several transposed convolutional layers corresponding to the encoder. The decoder takes the embedding vector produced by vector quantizer corresponding to the index as input to produce the reconstructed image.

The feature compressor and the action predictors of our SEMI policy network are both made up of MLPs. After flattening data that came from the encoder, we deploy it to pass through linear layers and predict each part of action separately. The hyper parameters we used in the experiment are shown in Table 1.

4.2 Simulation Environment and Data Collection

Based on the code from [4], we collect data and evaluate our proposed SEMI method on the CARLA simulator, along with other methods that we compared.

In order to act as an expert agent driver, we first write code to add manual driving support to the CARLA environment. We then implement an expert agent by having a human driver control the vehicle using an Xbox One controller. The driver will drive following a pre-set trajectory which will guide it to a random destination waypoint. Driver here need to control the vehicle within the correct lane given by trajectory, and avoid collision with other vehicles. We record bird-eye view images and the ego vehicle state (current speed) and control commands for each frame. We run the simulation for about 4 h and generated about 100k images and input commands. To be noticed, we only deploy the agent on the third map of the CARLA simulator for collecting training data, we do this to test the generalization ability of our SEMI method.

For organizing images into sequences, we group n_s successive frames' images and concatenate them together as a wide image to represent a sequence for our SEMI model's input.

4.3 Contrast Experiment

When it comes to comparison, we will not use models that output actions that require lower modules to execute, since those modules are difficult to program for different simulators or real world and our goal is to build an end-to-end model.

Here we train a CNN model to represent traditional BC methods [18]. We train a model using OREO [5] method as well(i.e. compute mask and predict based on a single image). We also train a sequential model without masking semantic objects in the training phase.

5 Results

5.1 Evaluation Procedure

From the maps provided by the CARLA simulator, we select 5 representative maps with different road styles for evaluation of model performance: the first, third, fourth, fifth, and seventh maps were selected. We denote these maps as Town1, Town2, Town3, Town4, and Town5. Each town has 10 different pre-computed routes. We compute each model's average result from these route as their result in a town. The simulation conditions, such as vehicle to driving and weather, remain consistent with the conditions in the recorded samples.

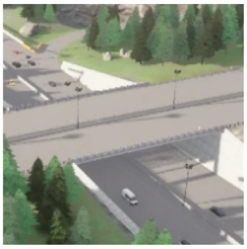
Towns: We will give a brief description of the towns we used for evaluation. Town1 and Town3 are both small simple towns with differences in road design. Town2 is the most complex town in CARLA since it has roundabouts and large junctions. Town4 is a squared-grid town with cross junctions and it has multiple lanes per direction. Town5 is a rural environment with narrow roads, corn, barns, and hardly any traffic lights. However, since the routes in each town are pre-computed randomly, the experimental results are affected by both the ability of the agent and the trait of the given route. Some unique parts in each town are shown in Fig. 4.

Metrics: Similar to Anzalone et al. [19], we evaluate models using four metrics. Collision rate is the crash rate of a single frame. A collision is recorded when the collision detector built into the ego vehicle reports a collision that happens with other vehicles or other still objects in the environment like buildings or roadblocks. Similarity measures the alignment between vehicle and planned road. Average speed is measured by compute the average of every frame's speed. Timesteps are the number of frames without collision.

Process: For every route in a town, each model will run test once. A test will stop when the vehicle runs safely for 2500 frames or collides with the environment or other vehicles. A model's metrics in a town will be computed by the results of 5 routes with the weight of routes: model's surviving frames in one route divided by surviving frames in 5 routes. The total metrics are the sum of weighted metrics from all towns.

(a) The river and the bridge in Town1.

(b) The roundabout of Town2.


(c) The underpass and overpass in Town3.

(d) Multi-lane roads and junctions in Town4.

(e) Simple junctions and unmarked roads in Town5.

Fig. 4. We pick and show some unique parts in each town we used, such as the bridge in Town1, the roundabout in Town2, the underpass in Town3, the multi-lane roads in Town4, and the unmarked roads in Town5. Notice that these situations except the roundabout are all unfamiliar to the agents since we only collect samples in Town2.

5.2 Discussion

As we can see from Table 2, the SEMI method performs better than its competitors in the metric of collision rate and average survival timesteps, while remains the ability to follow pre-set route. The average speed of our SEMI model is not as fast as its competitors, however, it could be proof that our model tends to brake aggressively to avoid crashing. Our attempt to address causal confusion in imitation learning does improve our model's driving ability. These traits demonstrated that models trained with SEMI method have the ability to perform expert-like strategy.

Models trained by OREO method and the method using sequential samples both show well capabilities in driving following pre-set route. Their results indicate that each part of our SEMI method helps the performance more or less. However, models trained by these methods do not have our model's ability to cope with other vehicles, led to their slightly higher collision rate.

The CNN method without any masking operation does suffer from causal confusion: though it gets low loss in the training phase, it runs terribly in testing scenes. In observation, it's the only model that cannot handle curves and braking well, leading to the shortest survival timestep among the four models.

Of all four metrics we considered in this paper, the collision rate is the most important metric we take into consideration about model performance. Because when it

Table 2. Performance of methods: Object-aware REgularizatiOn(OREO), CNN, Sequential samples Without masking(SW), and our Sequential Masking Imitation Learning(SEMI). Best results are highlighted in bold.

Metric/Method		Town					
		Town1	Town2	Town3	Town4	Town5	**Total**
Collision Rate(‰)	OREO [5]	0.32	0.74	1.19	0.60	0.59	0.64
	CNN [18]	1.28	8.47	1.21	0.86	3.14	1.68
	SW	0.92	0.77	0.54	0.23	0.87	0.59
	SEMI(Ours)	**0.19**	**0.52**	**0.46**	**0.19**	**0.44**	**0.33**
Similarity(%)	OREO [5]	**92.3**	84.6	91.0	94.2	74.4	88.07
	CNN [18]	**92.3**	61.1	91.1	83.1	63.3	83.33
	SW	91.6	**88.7**	89.9	**95.8**	75.6	**89.21**
	SEMI(Ours)	84.8	87.6	**94.5**	93.3	**81.4**	88.41
Average Speed(km/h)	OREO [5]	15.5	**15.2**	**42.9**	22.5	**21.9**	23.34
	CNN [18]	15.5	7.68	26.7	**26.2**	21.7	**23.58**
	SW	**18.0**	13.4	18.8	16.4	15.5	16.12
	SEMI(Ours)	16.5	14.6	19.2	19.9	14	17.11
Average Timesteps	OREO [5]	1216	534	840	1318	1010	984
	CNN [18]	312	118	825	928	318	500
	SW	431	1036	**1324**	1681	918	1078
	SEMI(Ours)	**2047**	**1148**	1303	**2009**	**1352**	**1572**

comes to the deployment and use of autonomous driving, the safety issue is the most cared part. So we can say that the SEMI method we proposed can maintain a safer driving style have more advantages compared to their competitors.

We can also analyze the experiment results based on the traits of each town we evaluated on. As we can see that SEMI model demonstrated its ability to cope with other vehicles and different environments, even when situations of these environments are complicated like Town2 and Town5. While complex traffic does affect the overall performance of models trained from each method, we can still notice that the SEMI model generally maintained a steady driving style, while other competing methods may fail to cope with it, leading to unstable results of these methods in several towns with different difficulties.

Overall, the result demonstrates that SEMI method does handle causal confusion well. An autonomous driving model trained from the SEMI method can manage each part of a direct control command and simulate an expert's behavior well.

5.3 Analysis

Figure 5 is a visualization of the first layer's weight of the trained policy network π from SEMI, use Fig. 1(a) as an example. The elements that do not appear with gray are the features π mainly used for prediction.

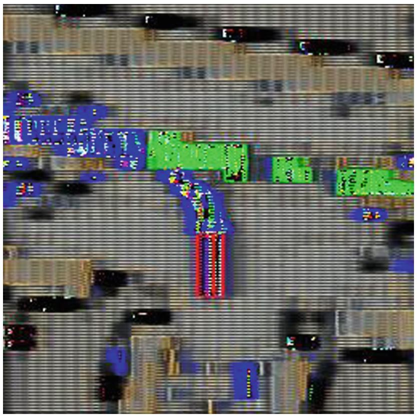


Fig. 5. The image is reconstructed from the combination of the encoded latent representation of Fig. 1(a) and the weight of the first layer in trained policy network π from SEMI, we make several processing to make the decision basis of π more interpretable for readers. Recall that in Fig. 1(a), red rectangle represents the ego vehicle, blue line represents the guidance line, and green rectangles represent other vehicles. The conspicuous elements there are features that π mainly takes into consideration for prediction, while other parts that are colored in gray are features that have little effect on π. (Color figure online)

We can use Fig. 5 for analyzing the decision basis of policy network π that trained with encoding and masking processes as proposed in SEMI. In a situation like Fig. 1(a), π generally takes the important features including the guidance line, the ego vehicle, the vehicles in the guided lane, and the boundary of the road into consideration. These picked features are also often used by human drivers to decide their actions.

The phenomenon that the downstream imitator trained with the SEMI method tends to perform imitation based on real important features can be an explanation for its good scores in several crucial metrics, since it has been proposed in the causal literature that imitators whose decisions rely on the same features that experts use, can eventually obtain expert-like strategies.

6 Conclusion

In this paper, we proposed a method, Sequential Masking Imitation Learning(SEMI) that can train an end-to-end driving model and handle causal confusion by randomly masking semantic objects in a sequence of observation samples. We evaluated our SEMI method on the CARLA simulator and found it outperforms several other behavior clone methods in several metrics we think an expert-like agent should maintain. To be specific, we found that the driving agent trained with our SEMI method showed a preference for taking other vehicles' information into consideration and its willing to brake to avoid collision. These traits, along with the ability to drive following visual guidance, lead our agent equipped with expert-like strategy.

The major limitation of this work is the observation format: the bird-eye view image is hard to obtain in the real world and it contains ground truth information which helps model making decisions. Reducing reliance on such information will be the direction of our future efforts. Another limitation is our model did collide with other vehicles and the environment sometimes, due to the situation surrounding may be unfamiliar to our model, especially when the expert's demonstration had not covered those situations.

Although there are limitations exist, our SEMI method does show a promising way for autonomous driving that requires no interaction with the actual environment and handle the causal confusion in imitation learning.

In future work, we expect to further investigate the causation in autonomous driving, and improve the ability of agent to interpret its surrounding environment. Tools such as causal discovery and counterfactual representation can be used for this purpose.

Acknowledgements. The work is supported by the National Key Research and Development Program of China (No. 2022YFB4201603), the National Natural Science Foundation of China (No. 61873033) and the Natural Science Foundation of Fujian Province (No. 2020H0012).

References

1. Saha, A., Mendez, O., Russell, C., Bowden, R.: Translating images into maps. In: 2022 IEEE International Conference on Robotics and Automation (ICRA). IEEE (2022)
2. Hadfield-Menell, D., Milli, S., Abbeel, P., Russell, S.J., Dragan, A.: Inverse reward design. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems, vol. 30. Curran Associates, Inc. (2017)
3. de Haan, P., Jayaraman, D., Levine, S.: Causal confusion in imitation learning. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing System, vol. 32. Curran Associates, Inc. (2019)
4. Chen, J., Xu, Z., Tomizuka, M.: End-to-end Autonomous Driving Perception with Sequential Latent Representation Learning. arXiv e-prints arXiv:2003.12464 (Mar 2020)
5. Park, J., et al.: Object-aware regularization for addressing causal confusion in imitation learning. In: Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., Vaughan, J.W. (eds.) Advances in Neural Information Processing Systems, vol. 34, pp. 3029–3042. Curran Associates, Inc. (2021)
6. Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: CARLA: an open urban driving simulator. In: Proceedings of the 1st Annual Conference on Robot Learning, pp. 1–16 (2017)
7. Zeng, W., Luo, W., Suo, S., Sadat, A., Yang, B., Casas, S., Urtasun, R.: End-to-end interpretable neural motion planner. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 8652–8661 (June 2019). https://doi.org/10.1109/CVPR.2019.00886
8. NG, A.: Algorithms for inverse reinforcement learning. In: Proceedings of of 17th International Conference on Machine Learning, vol. 2000, pp. 663–670 (2000)
9. Abbeel, P., Ng, A.: Apprenticeship learning via inverse reinforcement learning, p. 1. ACM Press (2004). https://doi.org/10.1145/1015330.1015430
10. Ratliff, N., Bagnell, J., Zinkevich, M.: Maximum margin planning. In: Proceedings of the 23rd international conference on Machine learning - ICML 2006, pp. 729–736. ACM Press (2006). https://doi.org/10.1145/1143844.1143936

11. Ziebart, B., Maas, A., Bagnell, J., Dey, A.: Maximum entropy inverse reinforcement learning, pp. 1433–1438. AAAI (2008)
12. Codevilla, F., Santana, E., Lopez, A., Gaidon, A.: Exploring the limitations of behavior cloning for autonomous driving. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV), pp. 9328–9337 (2019). https://doi.org/10.1109/ICCV.2019.00942
13. Katz, G., Huang, D.W., Hauge, T., Gentili, R., Reggia, J.: A novel parsimonious cause-effect reasoning algorithm for robot imitation and plan recognition. IEEE Trans. Cognitive Develop. Syst. **10**(2), 177–193 (2018). https://doi.org/10.1109/tcds.2017.2651643
14. Srivastava, N.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**(1), 1929–1958 (2014)
15. Yun, S., Han, D., Chun, S., Oh, S.J., Yoo, Y., Choe, J.: Cutmix: regularization strategy to train strong classifiers with localizable features. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV), pp. 6022–6031 (Oct 2019). https://doi.org/10.1109/ICCV.2019.00612
16. Zhong, Z., Zheng, L., Kang, G., Li, S., Yang, Y.: Random erasing data augmentation. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, pp. 13001–13008 (2020)
17. van den Oord, A., Vinyals, O., kavukcuoglu, k.: Neural discrete representation learning. In: Guyon, I. (eds.) Advances in Neural Information Processing Systems, vol. 30. Curran Associates, Inc. (2017)
18. Gleave, A., et al.: imitation: Clean imitation learning implementations. arXiv:2211.11972v1 [cs.LG] (2022). https://arxiv.org/abs/2211.11972
19. Anzalone, L., Barra, S., Nappi, M.: Reinforced curriculum learning for autonomous driving in carla. In: 2021 IEEE International Conference on Image Processing (ICIP), pp. 3318–3322 (2021). https://doi.org/10.1109/ICIP42928.2021.9506673