Appendices

A Proof of Theorem 1

$I_\phi(m_t; a_t|a_{t-1})$ is the parametrized conditional mutual information between m_t and a_t on the condition of a_{t-1}. The first equality holds since $a_t = r_t + a_{t-1}$. Then, the second equality can be obtained by using definitions of mutual information to expand $I_\phi(m_t; r_t|a_{t-1})$. Note that the conditional entropy $\mathbf{H}(r_t|a_{t-1})$ is not related to our optimizing variables ϕ since it doesn't contain m_t. Furthermore, according to the total probability formula, we can expand $\mathbf{H}_\phi(m_t, r_t|a_{t-1})$ to eliminate $\mathbf{H}_\phi(m_t|a_{t-1})$ and derive the third equality. The final inequality holds since the conditions $\{m_t, a_{t-1}\}$ is a superset of the conditions $\{m_t\}$.

$$
\begin{aligned}
&I_\phi(m_t; a_t|a_{t-1}) \\
&= I_\phi(m_t; r_t|a_{t-1}) \\
&= \mathbf{H}_\phi(m_t|a_{t-1}) + \mathbf{H}(r_t|a_{t-1}) - \mathbf{H}_\phi(m_t, r_t|a_{t-1}) \\
&= \mathbf{H}(r_t|a_{t-1}) - \mathbf{H}_\phi(r_t|m_t, a_{t-1}) \\
&\geq \mathbf{H}(r_t|a_{t-1}) - \mathbf{H}_\phi(r_t|m_t)
\end{aligned}
$$

B Implementation Details of Experiments in CARLA

B.1 Architectural details & Loss functions

We use the backbone of conditional imitation learning framework CILRS [9] and set all the input speed v_{in} to zero to create a POMDP [42].

The input o_t and $\hat{o}_t$ of all models is a three-dimensional tensor with the size of $30 \times 288 \times 80$. We stack the observed images ($3 \times 288 \times 80$ RGB images) along the first dimension in chronological order and set the total number of channels of all input tensors to 30 for fairness. o_t contains only the current frame and $\hat{o}_t$ has a relatively long observation history. However, both o_t and $\hat{o}_t$ have less than 10 images, so we set the remaining channels to all zeros.

We use ImageNet-pretrained ResNet34 [15] as the perception backbone for all methods to obtain latent representation. To accommodate 30-channel input, we repeat the first-layer convolution kernel 10 times in the first dimension and normalize the pretrained weight to 1/10 of the original.

The details of BCOH are shown in Fig. 5. Resnet34 casts the input $\hat{o}_t$ into a 512-dimensional compact representation. This representation is fed into a 3-layer MLP to obtain the estimated ego-velocity v_t (a scalar). Besides, the representation is concatenated with the output of 2-layer MLP with all-zero input. Then the concatenated feature is fed into a 1-layer MLP which reduces its dimension to 512. This fusion 512-dimensional vector is then fed into the corresponding 3-layer MLP conditioned on the current time-step command c_t, which finally outputs the current action a_t (a 2-dimensional vector). BCOH uses the speed

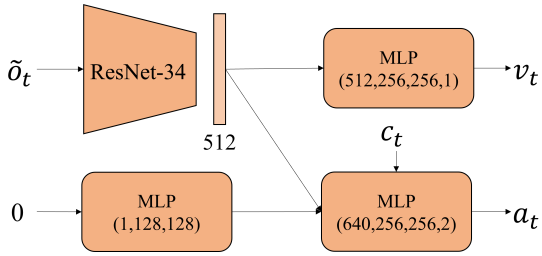

Fig. 5. CILRS architecture: The model is used as the BCOH.

regularization [9] to address the causal confusions to some extent. Thus, the loss function for BCOH is defined as follows,

$$L_{\text{BCOH}} = \alpha L(a_t, a_t^{gt}) + (1-\alpha)L(v_t, v_t^{gt}), \tag{4}$$

where a_t^{gt} and v_t^{gt} are the ground truths of the current action a_t and the speed v_t respectively, α denotes the weighting to the loss of a_t, and L is an L1 loss function.

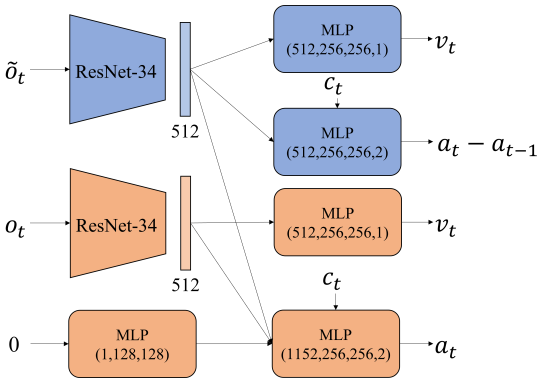

Fig. 6. Our architecture: blue blocks are the memory extraction module; orange blocks are the policy module. Each module is a variant of the CIRLS architecture.

The details of our model are shown in Fig.6. The memory module and the policy module in our model share a similar architecture with BCOH's described above. However, the memory module removes the MLP for all-zero input v_{in}, and the input for the policy module is o_t. The basic training objectives of policy module π_θ and memory extraction module M_ϕ are a_t and $a_t - a_{t-1}$ respectively. Similar to BCOH, each module of our model uses speed regularization. Therefore,

the loss functions we designed for each module are:

$$\begin{aligned} L_{M_\phi} &= \alpha L(a_t - a_{t-1}, a_t^{gt} - a_{t-1}^{gt}) + (1-\alpha)L(v_t, v_t^{gt}), \\ L_{\pi_\theta} &= \alpha L(a_t, a_t^{gt}) + (1-\alpha)L(v_t, v_t^{gt}), \\ L_{\text{overall}} &= L_{M_\phi} + L_{\pi_\theta} \end{aligned} \tag{5}$$

where a_{t-1}^{gt} is the ground truth of the previous action a_{t-1} and other symbols are the same with those in Eq.(4).

B.2 Architectural details of baselines in Ablation Studies

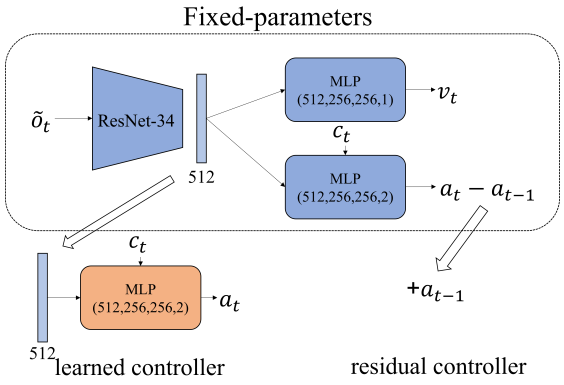

Fig. 7. Memory only details

Fig. 7 shows the details of **Memory only**. We fixed the parameters of a well-trained memory extracted module and try to use this module's output or intermediate feature to predict the current action a_t. The **Memory only: residual controller** adds the predicted residue output directly into last-step action a_{t-1} to obtain the prediction of a_t. The **Memory only: learned controller** uses the extracted feature (the output of ResNet-34) as the input to regress a_{t-1} via a 3-layer MLP.

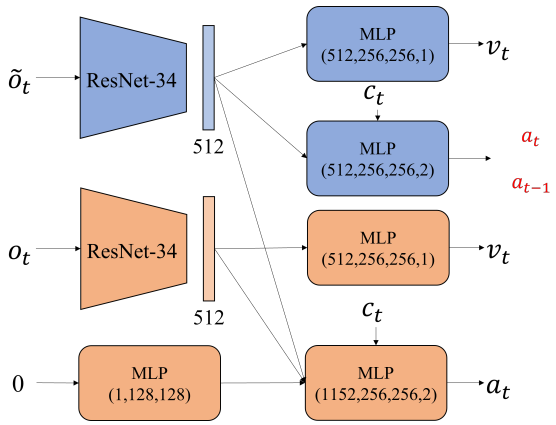

Fig. 8. Memory module objective details

Fig. 8 shows the details of **Memory module objective**. We train the model with different objectives (a_t or a_{t-1}) for the memory extraction module, and the remaining setup is the same with the our proposed model.

B.3 Other Details

For all implemented methods, we apply the same hyper-parameters shown in Table 6, including total training iterations, batch size, α, loss function, optimizer setup, and other configurations about the learning rate (LR) scheduling.

Table 6. Hyper-parameters of experiments

Configuration	Value
Total training iterations	100k
Batch size	160
α	0.95
Loss function	L_1
Optimizer	Adam
Betas	(0.9, 0.999)
Eps	1e-08
Weight decay	0
Initial LR	2e-4
LR decay threshold	5000
LR decay rate	0.1
LR lower bound	1e-7

LR scheduling: LR starts with an initial learning rate (initial LR) and decays when the best loss is unable to go down further for a preset number of iterations

(LR decay threshold). Then, each decay learning rate is multiplied by a decay rate (LR decay rate) until it is lower than the set minimum learning rate (LR lower bound). As a result, the LR adjusts adaptively and will not vanish in the whole training process.

Data Augmentation: We apply noise injection [19] and multi-camera data augmentation [4, 13] on our training dataset to alleviate the distribution shift. Both of them are commonly used in the autonomous driving.

Random seeds: We retrained the proposed framework 3 times with different random initialization and test our agent on 25 routes for 4 kinds of weather with 3 different seeds. It makes sure we obtain a statistically significant better result.

B.4 Failure mode in CARLA *NoCrash*

Table 7. Failure mode on training conditions.

Traffic	*Regular*		*Dense*	
Method	*#COLLISION*	*#TIMEOUT*	*#COLLISION*	*#TIMEOUT*
BCSO	53.0 ± 7.9	$\mathbf{10.2 \pm 3.1}$	76.4 ± 3.5	$\mathbf{11.1 \pm 2.9}$
BCOH	11.1 ± 3.1	21.9 ± 12.7	30.2 ± 7.9	36.1 ± 14.5
OURS	$\mathbf{6.8 \pm 1.3}$	15.2 ± 0.2	$\mathbf{25.0 \pm 5.4}$	23.3 ± 7.6
DAGGER	14.8 ± 2.9	15.9 ± 8.5	35.0 ± 3.6	23.0 ± 7.1
HD	18.3 ± 5.2	12.2 ± 4.4	45.3 ± 3.5	20.3 ± 5.6
FCA	14.7 ± 3.3	27.3 ± 8.8	34.4 ± 8.1	35.3 ± 9.6
Keyframe	13.8 ± 2.7	11.9 ± 5.8	33.9 ± 6.6	24.8 ± 7.9

Table 8. Failure mode on new weather

Traffic	*Regular*		*Dense*	
Method	*#COLLISION*	*#TIMEOUT*	*#COLLISION*	*#TIMEOUT*
BCSO	31.7 ± 5.8	8.7 ± 3.1	42.3 ± 0.9	$\mathbf{6.3 \pm 1.2}$
BCOH	7.0 ± 1.4	16.0 ± 6.4	18.3 ± 4.6	16.3 ± 6.8
OURS	$\mathbf{5.7 \pm 1.5}$	$\mathbf{3.7 \pm 4.7}$	$\mathbf{18.0 \pm 2.6}$	6.3 ± 3.2
DAGGER	12.0 ± 1.4	10.7 ± 1.7	22.7 ± 2.6	13.3 ± 7.1
HD	11.0 ± 2.8	11.3 ± 7.6	21.0 ± 3.6	12.3 ± 6.2
FCA	9.0 ± 2.2	22.3 ± 13.9	18.7 ± 9.6	23.0 ± 12.3
Keyframe	7.3 ± 1.2	9.3 ± 6.2	22.7 ± 2.9	11.7 ± 6.6

Table 9. Failure mode on on new town

Traffic	*Regular*		*Dense*	
Method	*#COLLISION*	*#TIMEOUT*	*#COLLISION*	*#TIMEOUT*
BCSO	52.0 ± 2.2	30.3 ± 0.9	73.0 ± 1.6	$\mathbf{22.3 \pm 1.7}$
BCOH	33.0 ± 7.5	42.0 ± 13.6	$\mathbf{43.3 \pm 11.1}$	52.0 ± 13.4
OURS	$\mathbf{32.7 \pm 6.7}$	$\mathbf{28.0 \pm 4.6}$	50.3 ± 4.7	30.7 ± 3.1
DAGGER	31.3 ± 4.2	36.0 ± 7.5	52.3 ± 3.7	36.7 ± 6.6
HD	30.7 ± 4.2	37.7 ± 1.7	55.3 ± 6.3	34.0 ± 6.5
FCA	31.3 ± 9.1	48.3 ± 10.3	49.0 ± 8.6	43.3 ± 10.5
Keyframe	34.3 ± 1.2	31.7 ± 4.1	48.3 ± 3.9	38.0 ± 6.7

There are two kinds of failure modes in CARLA *NoCrash*: collision and timeout. The collision means the driving agent falls the episode due to collision with other objects such as vehicles, pedestrians, and guardrails; The timeout means it exceeded the time limit of the episode. Failure mode results in CARLA *NoCrash* are shown in Tab. 7, Tab. 8, and Tab. 9. We note that our method is not always the lowest for the timeout failure rate, and that is because other methods might have a much higher collision rate. For example, BCSO is consistently the best in *#TIMEOUT* metric because most of its episodes end with collisions. Severe copycat problems with BCOH also lead to a high timeout failure rate.

B.5 Other Experiments

Reactions to traffic lights Traffic lights are essential facilities for driving, and it decides whether the vehicle can pass the intersections safely. However, a traffic light occupies only a few pixels of the entire picture, and if it changes, it's hard for the imitation learner to concentrate on this slight but important change. Moreover, suppose the imitation learner suffers from copycat problems and has shortcuts. In that case, it will ignore the semantic information of the observation and miss the instructions of traffic lights, which may cause more vehicle collisions or traffic jams. To evaluate how much attention our framework pays to traffic lights, we count the percentage of each imitator passing the intersection while the traffic light is green in CARLA *Nocrash Dense*.

Table 10. Percentage of obeying traffic lights

Method	**BCOH**	**Keyframe**	**OURS**
Green light(%)(↑)	30.6	42.1	**66.3**

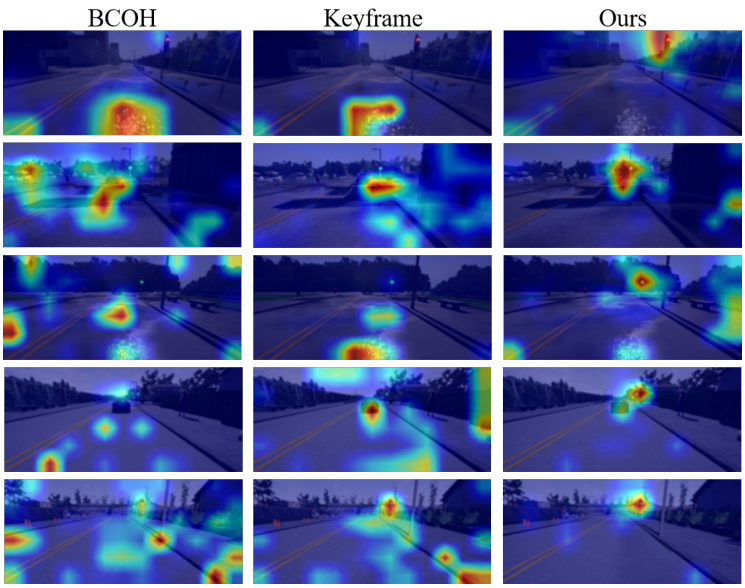

Fig. 9. Attention maps generated by Grad-CAM [35]

Table 10 shows the percentage of obeying traffic lights for all methods, and Fig. 9 displays some visualization results about the observation with the traffic light in the validation set. Our method is the most compliant with traffic lights which helps our method achieve high *#SUCCESS*. The visualization results also show our method focuses more on the correct causal clue of the traffic light while BCOH and Keyframe concentrate on spurious road features.

Minimize any previous action's impact The model we propose only removes the information about last-step action a_{t-1}. However, the whole sequence can somehow have an impact on the shortcut learning of the predicted action a_t. In order to minimize any previous action's impact, we have done an interesting ablation by adding more objectives for the memory module. Intuitively, we define m residual prediction branches for memory module, and the ith branch's objective is $a_t - a_{t-i}$. We tested it on CARLA *NoCrash* Dense Benchmark. The success rate of one branch is 52.0%. After increasing to 2 branches it slightly increases to 52.6%; while further going to 4 branches degrade to 50.7%. This suggests that having more branches can be beneficial, but having too many branches will not help.

The influence of two-streams architecture To address the concern about the potential unfairness brought by the larger capacity of the two-stream network, we provide two extra ablations by running BCOH and KeyFrame with the two-stream architecture. We choose BCOH and KeyFrame since their performance is strong as shown in Table 2. More specifically, we keep the two-stream architecture the same but replace the inputs to both streams as the observa-

tions with histories. We supervise the policy stream with the corresponding loss function of BCOH and KeyFrame. The results are shown in Table 11. Much lower *#SUCCESS* and higher *#TIMEOUT* of two-streams baselines indicate that two-streams architecture alone, without our method, suffers from severe copycat problems. We hypothesized that two stream architecture has even lower performance than their one stream counterparts because more parameters make it more vulnerable to the copycat problem.

Table 11. Results of two-stream architecture on CARLA *Nocrash Dense* benchmark

Metrics	*#SUCCESS*	*#TIMEOUT*
Two-streams BCOH	23.7 ± 3.1	48.7 ± 2.1
Two-streams Keyframe	38.7 ± 2.5	33.0 ± 7.5
BCOH	34.1 ± 7.5	36.1 ± 14.5
Keyframe	41.9 ± 6.2	24.8 ± 7.9
OURS	52.0 ± 2.3	23.3 ± 7.6

C Implementation Details of Experiments in MuJoCo-Image

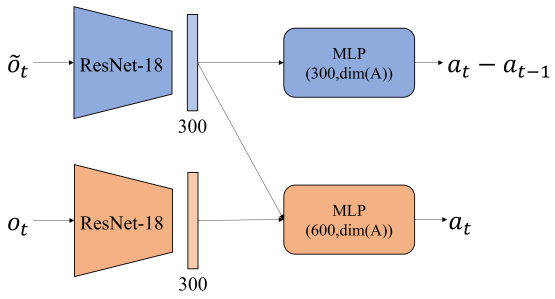

Fig. 10. Our MuJoCo model: blue blocks are the memory extraction module; orange blocks are the policy module. dim(A) denotes the dimension of any action $a \in A$.

Fig. 10 shows our model we used in MuJoCo. Both memory extraction module and policy module apply ResNet18 as their perception backbone to obtain a 300-dimensional feature and utilize this extracted feature to predict the defined objective via a one-layer MLP. The overall loss function is defined as follows

$$L_{\text{overall}} = L(a_t - a_{t-1}, a_t^{gt} - a_{t-1}^{gt}) + L(a_t, a_t^{gt}), \tag{6}$$

where all the symbols are the same with those in Eq.(5).

We apply the hyper-parameters shown in Table 12, including total training iterations, batch size, α, loss function, optimizer setup, and other configurations about the learning rate (LR) scheduling, which has been explained in Sec.B.3.

Table 12. Hyper-parameters of experiments in MuJoCo-Image

Configuration	Value
Total training iterations	120k
Batch size	128
Loss function	L_2
Optimizer	Adam
Betas	(0.9, 0.999)
Eps	1e-08
Weight decay	0.03
Initial LR	0.1
LR decay threshold	40k
LR decay rate	0.1
Early Stop	True

Other MuJoCo Environments Following the original setting, we further conduct experiments in three more MuJoCo environments, including Ant, Reacher, and Humanoid. The demonstration trajectories are collected by TRPO experts. There are 1k samples for Ant, 5k samples for Reacher, and 200k samples for Humanoid, according to the task complexities. As shown in Table 13, our method outperforms the baselines in all these new MuJoCo Environments. We compare to BCOH and KeyFrame since they are the two stronger baselines as shown in Table 3.

Table 13. The average reward

Environment	Ant	Reacher	Humanoid
BCOH	746 ± 96	-81 ± 8	258 ± 3
Keyframe	790 ± 85	-71 ± 5	294 ± 53
OURS	860 ± 68	-62 ± 7	372 ± 20