

Domain Generalization for Vision-based Driving Trajectory Generation

Yunkai Wang¹, Dongkun Zhang^{1,2}, Yuxiang Cui¹, Zexi Chen¹,
Wei Jing², Junbo Chen², Rong Xiong¹, Yue Wang^{1†}

Abstract—One of the challenges in vision-based driving trajectory generation is dealing with out-of-distribution scenarios. In this paper, we propose a domain generalization method for vision-based driving trajectory generation for autonomous vehicles in urban environments, which can be seen as a solution to extend the Invariant Risk Minimization (IRM) method in complex problems. We leverage an adversarial learning approach to train a trajectory generator as the decoder. Based on the pre-trained decoder, we infer the latent variables corresponding to the trajectories, and pre-train the encoder by regressing the inferred latent variable. Finally, we fix the decoder but fine-tune the encoder with the final trajectory loss. We compare our proposed method with the state-of-the-art trajectory generation method and some recent domain generalization methods on both datasets and simulation, demonstrating that our method has better generalization ability. Our project is available at <https://sites.google.com/view/dg-traj-gen>.

I. INTRODUCTION

Vision-based navigation is an appealing research topic in recent years. One of the approaches in vision-based navigation is learning-based trajectory generation from RGB images. Most of the works are only validated on data from the same domain for training and testing, as shown in Fig. 1(a). However, as a common challenge in learning-based algorithms, out-of-distribution (OOD) scenarios can lead these trajectory generation approaches to have poor generalization results and make dangerous decisions.

To allow the model to be trained on a specific dataset and transferred to a new scenario, a straightforward solution to deal with the OOD problem is *domain adaptation (DA)* approach, which collects some data from the target domain to fine-tune the model, shown in Fig. 1(b). However, in the autonomous driving application scenario, a more realistic setting is that the target domain is unknown, so it is impossible to obtain the target domain data in advance. The problem dealing with such OOD scenario is introduced as *domain generalization (DG)* [1], which aims to learn a model from source domains and generalize to any OOD target domain, shown in Fig. 1(c).

To realize domain generalization, one approach is to use labels of multi-domain training data [2] [3] to build an

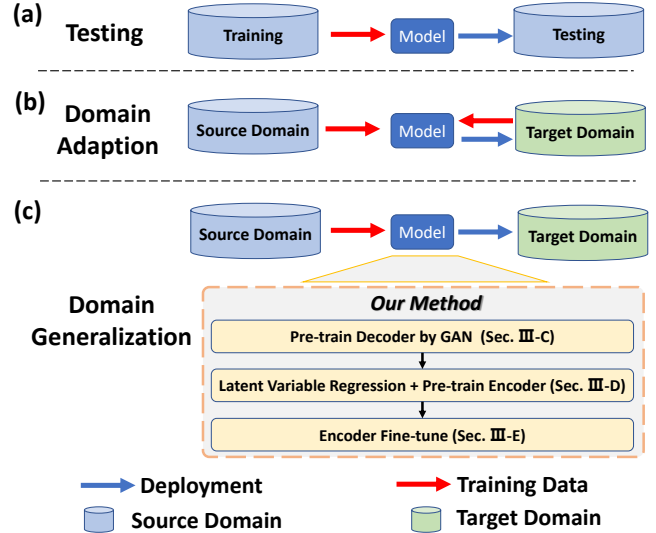


Fig. 1: Different frameworks for testing, domain adaption, and domain generalization method are shown in (a), (b), and (c) respectively. We propose a three-stage training approach to realize real domain generalized trajectory generation method for autonomous vehicles.

auxiliary task, trying to learn domain invariant data representations. However, domain labels are difficult to clearly define on most current driving datasets. One approach to relax this problem is using ensemble learning [4] to train a certain number of models with different training data and obtain a more robust planning result according to outputs of all models, while this approach needs a longer training time and inference time, making it resource-consuming. One appealing approach proposed recently is *Invariant Risk Minimization (IRM)* [5], which assumes the invariance of the feature-conditioned label distribution and aims to remove spurious correlations (i.e. dataset-specific biases) in a representation. However, most of the works using IRM are restricted to classification problems with simple datasets and linear classifiers.

In this paper, we proposed a trajectory generation method to real domain generalized visual navigation for autonomous vehicles by extending the IRM approach. We construct an encoder-decoder network structure, where the decoder uses the method in [6] to represent continuous trajectories. In order to satisfy the constraint in the IRM problem, we use a Lagrangian form, which is the squared norm of the gradient of the trajectory generator. It is difficult to solve this

This work was supported by Alibaba Group through Alibaba Innovative Research (AIR) Program.

¹Yunkai Wang, Dongkun Zhang, Yuxiang Cui, Zexi Chen, Rong Xiong, and Yue Wang are with the State Key Laboratory of Industrial Control Technology and Institute of Cyber-Systems and Control, Zhejiang University, Hangzhou, China.

²Dongkun Zhang, Wei Jing and Junbo Chen are with the Department of Autonomous Driving Lab, Alibaba DAMO Academy, Hangzhou, China

[†] Corresponding author, wangyue@iipc.zju.edu.cn

problem directly because the gradient norm term can hinder the optimization. To solve this problem, we propose a three-stage training approach, which is shown in the lower part of Fig. 1(c): 1) We leverage an adversarial learning approach to train a trajectory generator as the decoder. 2) Based on the pre-trained decoder, we infer the latent variables corresponding to the trajectories, and pre-train the encoder by regressing the inferred latent variable. 3) We fix the decoder but fine-tune the encoder with the final trajectory loss.

We compare our proposed method with the state-of-the-art trajectory generation method and some recent domain generalization methods on both datasets and simulation, demonstrating that our method has better generalization ability. To the best of our knowledge, this is the first work to train driving trajectory generation models in one domain and directly transfer them to other domains. To summarize, the main contributions of this paper include the following:

- We formulate the domain generalization for driving trajectory generation problem as a non-linear IRM problem. And we propose a set of network training strategies for optimizing the non-linear IRM problem.
- We implement a trajectory generator with good domain generalization ability. We test our method on both datasets and simulation, showing that our method has a stronger generalization ability than others in both open-loop and closed-loop experiments.

II. RELATED WORKS

A. Domain Generalization

The goal of the domain generalization (DG) problem [1] is to learn a model using data from the source domain and generalize to any out-of-distribution (OOD) target domain. Existing domain generalization approaches generally fall into the following groups:

Domain-Adversarial Learning. The goal of domain-adversarial learning [2], [7], [8] is to align the distributions among different domains, and it formulates the distribution minimization problem through a minimax two-player game, without explicitly measuring the divergence between two probability distributions.

Learning Disentangled Representations. These approaches learn to represent the data as multiple features instead of a single domain-invariant feature and separate out the domain-specific parts [3], [9].

Ensemble Learning. It uses different splits of training data to learn multiple models with the same structure but different weights, which can boost the performance of a single model [10] [11]. Ensemble learning is effective to cope with OOD data with fewer constraints. However, these approaches require more computational resources, the training time and inference time of these approaches grow linearly with the number of models.

Invariant Risk Minimization. It was first proposed by Arjovsky et al. [5], which aims to remove spurious correlations in a representation and ensure that the learned representation can lead to a minimal classification error over

all source domains. Recent new works inspired from IRM to address the OOD generalization problem include [12]–[15]. However, most of these approaches are validated only by doing classification tasks on toy datasets. We extend the IRM method by using non-linear models to do the trajectory generation task. And compared to other domain generalization methods, our proposed method does not rely on domain knowledge, pixel-reconstruction, or multiple models ensembling, making it easy to train and apply.

B. OOD Generalization in Driving Policy Learning

Recently, several works have investigated the problem of OOD generalization problem in driving policy learning. Zhang et al. [16] proposed to use bisimulation metrics to learn robust latent representations which encode only the task-relevant information from observations in reinforcement learning. Unfortunately, there are still challenges in applying reinforcement learning methods to real-world driving tasks. Filos et al. [4] proposed an epistemic uncertainty-aware trajectory planning method by training an ensemble of density estimators and online optimizing the trajectory concerning the most pessimistic model. This approach achieves good results for OOD scenarios. However, model ensembling and online optimization will consume more computational resources and have a longer inference time, while in this paper, we only use one single model to implement OOD generalization.

III. METHOD

A. Background

Most works in driving tasks use *Empirical Risk Minimization (ERM)* principle, which assumes there is a joint probability distribution $P(x, y)$ over input data X and label data Y , and the training data is drawn i.i.d. from $P(x, y)$. Given a loss function $L(\hat{y}, y)$ which measures how different the prediction \hat{y} is from the ground truth y , ERM aims to find a hypothesis h^* to minimize the empirical risk, which can be formulated as the following optimization problem:

$$h^* = \arg \min_h \frac{1}{N} \sum_{i=1}^N L(h(x_i), y_i) \quad (1)$$

However, when the joint probability distribution $P(x, y)$ varies in the testing data, ERM methods may yield poor generalization results. To overcome the distribution shift problem, as well as the absence of target domain data, domain generalization is introduced.

One of the important approaches in domain generalization is *Invariant Risk Minimization* [5], which assumes the invariance of the feature-conditioned label distribution $\mathbb{E}[y|\Phi(x)]$. To find an approximate solution, a Lagrangian form is introduced [17]:

$$\min_{\Phi, \mathcal{G}} \sum_{e \in \mathcal{E}} \left[\mathcal{R}^e(\Phi, \mathcal{G}) + \lambda \|\nabla_{\mathcal{G}} \mathcal{R}^e(\Phi, \mathcal{G})\|_2^2 \right] \quad (2)$$

where Φ is a data representation, \mathcal{G} is a classifier, e is a kind of environment from the set \mathcal{E} , and λ is a regularization parameter. The latter IRM penalty term constrains the

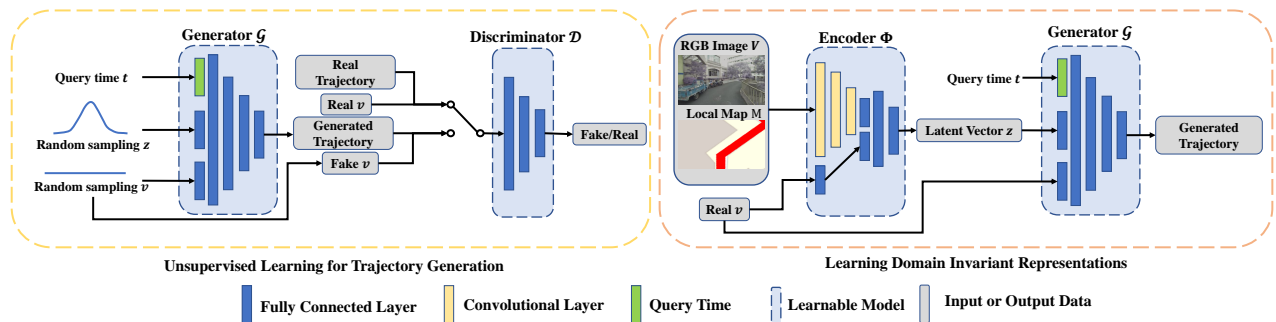


Fig. 2: Network structure diagram of our proposed method. The generator \mathcal{G} is trained with the discriminator \mathcal{D} by an unsupervised adversarial learning approach as a decoder. Based on the pre-trained decoder, we infer the latent variables corresponding to the trajectories, and pre-train the encoder by regressing the inferred latent variable. Finally, we fix the decoder but fine-tune the encoder with the final trajectory loss. The inputs of the encoder Φ include the RGB image V which is acquired from the front-view camera, the local routing planning map M which is cropped from an offline map according to the current low-cost GPS and IMU data, and the current speed v .

classifier to be the optimal classifier and is not worse for each sample, avoiding the situation in ERM methods where some data errors may be very large. In the linear regression case, the analytical solution of the problem can be directly calculated, and the IRM problem can be simplified as *IRMv1* [5]:

$$\min_{\Phi} \sum_{e \in \mathcal{E}} \left[\mathcal{R}^e(\Phi) + \lambda \|\nabla_{\mathcal{G}|\mathcal{G}=1.0} \mathcal{R}^e(\Phi, \mathcal{G})\|_2^2 \right] \quad (3)$$

where Φ becomes the entire invariant predictor, $\mathcal{G} = 1.0$ is a scalar and fixed “dummy” classifier.

B. Problem Setup

For the vision-based driving trajectory generation task, we leverage three sensor data as input of our method: the front-view RGB image V which contains environmental information, the local route planning map M which contains driving intention information as inputs, and the current speed v of the vehicle. The local route planning map M is cropped from an offline map based on the current pose from the low-cost GPS and inertial measurement unit (IMU), which is similar to [18] and [19]. Both images are resized to 400×200 and concatenated together as part of the encoder’s input.

We assume that in the driving trajectory generation task, there is an encoder Φ which maps all the sensor data to the latent variable z and a trajectory generator (decoder) \mathcal{G} which uses the latent variable z , current speed v , and the query time t to generate trajectory points. The expression for the encoder Φ can be written as:

$$z = \Phi(V, M, v) \quad (4)$$

and the expression for the trajectory generator \mathcal{G} [6] can be written as:

$$\mathbf{y}(t) = \mathcal{G}(t, z, v) \quad (5)$$

The network structure is shown in the right part of Fig. 2.

The risk function used in this paper is:

$$\mathcal{R} = \sum \left[\|\mathbf{y}_{\parallel} - \hat{\mathbf{y}}_{\parallel}\|_2^2 + \alpha \|\mathbf{y}_{\perp} - \hat{\mathbf{y}}_{\perp}\|_2^2 \right] \quad (6)$$

where \mathbf{y}_{\parallel} and \mathbf{y}_{\perp} are the generated longitudinal and lateral displacement respectively, $\hat{\mathbf{y}}_{\parallel}$ and $\hat{\mathbf{y}}_{\perp}$ are the ground truth longitudinal and lateral displacement respectively, α is a regularizer balancing between longitudinal loss and lateral loss, and we use $\alpha = 5$ in our experiments.

The most straightforward approach is to use the *IRMv1* method to regress discrete trajectory points using a linear predictor, as mentioned in [5]. We call this method as *Traj IRM*. In our subsequent experiments, we find that *Traj IRM* method does not perform well enough in the trajectory generation task. Therefore, we introduce a non-linear decoder to improve the model representation ability. Refer to Eq. 2, we also consider using a Lagrangian form, and the non-linear IRM optimization problem becomes:

$$\min_{\theta, w} \sum_{e \in \mathcal{E}} \left[\mathcal{R}^e(\Phi_{\theta}, \mathcal{G}_w) + \lambda \|\nabla_w \mathcal{R}^e(\Phi_{\theta}, \mathcal{G}_w)\|_2^2 \right] \quad (7)$$

where θ is the parameters of the encoder Φ , and w is the parameters of the decoder \mathcal{G} . The former term is the ERM term and the latter term is the IRM penalty term. It is difficult to solve this problem directly because the IRM penalty term can hinder the optimization.

Hence, we propose to use a three-stage approach to learn the IRM regularized trajectory generation: 1) We leverage an adversarial learning approach to train a trajectory generator as the decoder. 2) Based on the pre-trained decoder, we infer the latent variables corresponding to the trajectories and pre-train the encoder by regressing the inferred latent variable. 3) We fix the decoder but fine-tune the encoder with our proposed trajectory loss in an end-to-end manner.

C. Unsupervised Learning for Trajectory Generation

Trajectory Representation. In this paper, we use a non-linear decoder to generate continuous trajectories, which was proposed in [6]. In that work, the trajectory generator \mathcal{G} takes three inputs: current speed v as a condition, the latent variable z as a trajectory prior, and the query time t , and it outputs the trajectory point corresponding to the time t , which has the same expression as Eq. 5. High-order physical

quantities such as velocity and acceleration can be obtained analytically by calculating the high-order partial derivatives of the outputs with respect to time t :

$$v(t) = \frac{\partial \mathbf{y}(t)}{\partial t}, a(t) = \frac{\partial^2 \mathbf{y}(t)}{\partial t^2} \quad (8)$$

In this paper, we follow this method to represent trajectories.

Latent Action Space Learning. Since the linear classifier of IRM can be solved analytically, the convergence of the classifier is not a concern, while the non-linear classifier has convergence problems. Therefore, it is important to train a good decoder without relying on high-dimensional inputs. We propose to use the unsupervised GAN to train the decoder. After training, the generator can be seen as a trained and converged decoder, which to some extent can be analogous to the classifier in linear IRM case, and we then can fix it in the process to optimize Eq. 7.

Specifically, inspired by the multi-modal experiment in *Conditional GAN* [20], we sample the trajectory prior \mathbf{z} from a standard Gaussian distribution $p_z(\mathbf{z})$, and a one-dimensional noise speed \tilde{v} from a uniform distribution $p_{\tilde{v}}(\tilde{v})$ between 0 and the maximum speed as a condition:

$$\mathbf{z} \sim p_z(\mathbf{z}), \tilde{v} \sim p_{\tilde{v}}(\tilde{v}) \quad (9)$$

And the trajectory generator \mathcal{G} takes these variables to generate a fake trajectory:

$$\tilde{\mathbf{y}}(t) = \mathcal{G}(t, \mathbf{z}, \tilde{v}) \quad (10)$$

Then we use a time series with equal interval sampling and input to the trajectory function to obtain the corresponding discrete trajectory points. The discriminator network \mathcal{D} takes these generated trajectory points $\tilde{\mathbf{y}}$ or ground truth trajectory points \mathbf{y} as input, and determines whether it is sampled from the generator network or from the ground truth data. The network structure is shown in the left part of Fig. 2. To improve the stability of the training process and prevent severe model collapse, we use WGAN-GP [21] as our adversarial learning approach.

D. Encoder Pre-training

We view the encoder pre-training task as a regression problem on the latent action space. However, the problem is that there is no target latent variables $\hat{\mathbf{z}}$ to supervise the encoder training. Therefore, we infer the latent variables for each trajectory. Referring to the Eq. 13, we use the risk function along with the IRM loss. Since the latent variables are sampled from the standard Gaussian distribution in the training process of the GAN model, we also add a constraint on the norm of the latent variable \mathbf{z} so that its distribution is as close as possible to the standard Gaussian distribution. The final loss function to obtain the target latent variable $\hat{\mathbf{z}}$ is shown in Eq. 11.

$$\hat{\mathbf{z}} = \arg \min_{\mathbf{z}} \sum_{e \in \mathcal{E}} \left[\mathcal{R}^e(\mathbf{z}, \mathcal{G}_{w_0}) + \lambda \|\nabla_{w|w=w_0} \mathcal{R}^e(\mathbf{z}, \mathcal{G}_w)\|^2 + \lambda_2 \|\mathbf{z}\|_2^2 \right] \quad (11)$$

where w_0 is the parameters pre-trained by GAN, λ_2 is a regularizer. In practice, we use Adam optimizer with 0.1 initial learning rate to optimize the latent variable $\hat{\mathbf{z}}$ as the target label data for supervised learning. Then we pre-train the encoder by regressing the latent variable \mathbf{z} , and the loss function is:

$$\theta_0 = \arg \min_{\theta} \sum \|\Phi_{\theta}(V, M, v) - \hat{\mathbf{z}}\|_2^2 \quad (12)$$

After obtaining the latent variables $\hat{\mathbf{z}}$ from the multi-step optimization, a simple way is to follow the IRM approach to do linear regression on the latent space by using *IRMv1* method [5]. However, we believe that it will introduce both the error of the latent variables used for supervision and the error of learning these latent variables, and our subsequent experiments show that this method works, but not very well. We call this method as *Latent IRMv1*.

E. End-to-End Training

After pre-training the encoder, we train our model in an end-to-end manner. In order to guarantee the feature-conditioned label distribution $\mathbb{E}[y|\Phi(x)]$ remains invariant, we propose to fix the parameters of the decoder \mathcal{G} in the training process, and just fine-tune the encoder. We use the GAN pre-trained parameters as the fixed parameters for the decoder, instead of random parameters, which can reduce the difficulty of optimization in the latent space and speed up training.

Therefore, the optimization problem in Eq. 7 is reduced to only search the parameters of the encoder:

$$\min_{\theta} \sum_{e \in \mathcal{E}} \left[\mathcal{R}^e(\Phi_{\theta}, \mathcal{G}_{w_0}) + \lambda \|\nabla_{w|w=w_0} \mathcal{R}^e(\Phi_{\theta}, \mathcal{G}_w)\|^2 \right] \quad (13)$$

We call the loss function in Eq. 13 as *Non-linear IRM (NIRM)* loss.

In our experiments, we also discuss the use of a decoder with random parameters, i.e., the decoder has not yet converged well, but we use it to train the model in the same way. We call this method *Random NT+NIRM*. And our subsequent experiments demonstrate the poor performance of the model trained by this method.

IV. EXPERIMENTS

In this paper, we use different ablated models to validate the effectiveness of our proposed method on the open-source driving datasets and compare our method with the state-of-the-art trajectory generation method and recent domain generalization methods on the open-source driving datasets and the CARLA simulation.

A. Dataset and Metrics

We use three driving datasets to validate our method.

Oxford Radar RobotCar (RobotCar) [22] is a radar extension to the Oxford RobotCar Dataset [23], providing 280km driving data around Oxford, UK. Since this dataset has simpler and practical data collection conditions, with limited geographic space and different collection times, we use this dataset as a training dataset.

TABLE I: Ablation study results of generalization from RobotCar Dataset to KITTI and CARLA Dataset. The three columns on the right are the average displacement error (m) on three different testing datasets. Lower metrics have better results.

Algorithm	DFX	DPT	IRM	RobotCar*	KITTI	CARLA
E2E NT [6]				0.60	2.13	1.36
E2E NT+NIRM			NIRM	0.68	2.06	1.25
Random NT+NIRM	✓		NIRM	1.50	2.32	1.45
Traj IRM	✓		IRMv1	0.67	2.16	1.31
Latent IRMv1	✓	✓	IRMv1	0.77	1.77	1.02
Ours	✓	✓	NIRM	0.85	1.70	0.92

*“DFX” means “Decoder Fixed”, “DPT” means “Decoder Pre-trained”, and “IRM” means “With IRM”.

KITTI Raw Data (KITTI) [24] contains 6 hours of traffic scenarios using a variety of sensor modalities. Since this dataset has more diverse driving scenarios, we use this dataset as a testing dataset.

CARLA Dataset is collected by a human driver for about 2 hours in the CARLA [25] simulation, with a speed limit of 30 *km/h* under different weather conditions. Since the cost of changing weather conditions in the simulation is very low and the driving trajectories are relatively simple, it is used as a testing dataset in the experiments.

Metrics. In this paper, we use average displacement error [26], which is the average Euclidean distance between the ground truth trajectory points and the generated trajectory points at the corresponding moment, to evaluate the performance of different methods.

B. Ablation Study

We validate the advantages of our proposed method by testing different ablated models from the original model as below:

- *E2E NT*: We directly train the model end-to-end in an ERM manner.
- *E2E NT+NIRM*: We train the model end-to-end with *NIRM* loss.
- *Random NT+NIRM*: This method is proposed in Sec. III-E. We use a fixed neural trajectory generation model with random parameters instead of the parameters of the pre-trained GAN model.
- *Traj IRM*: This method was proposed in [5], and is mentioned in Sec. III-B. We use this method to regress the position coordinates of 16 discrete trajectory points.
- *Latent IRMv1*: This method is proposed in Sec. III-D, which do linear regression on the latent space by using *IRMv1* method.
- *Ours*: The method proposed in this paper, which is trained by our proposed three-stage training approach.

The ablation study results of transfer from RobotCar Dataset to KITTI Dataset and CARLA Dataset are shown in Tab. I. Comparing with the results of the *E2E NT* method, using IRM can improve model generalization ability, which illustrates the effectiveness of the IRM approach. Comparing with the results of the *E2E NT+NIRM* method and *Random NT+NIRM* method, our method has a stronger

generalization ability. Since the pre-trained decoder we use is a converged decoder with fixed parameters, we believe it is a correct analogy to the linear IRM case, so our method has a better performance. Comparing the results of the *Traj IRM* method, it shows that the decoder using a simple “dummy” predictor does not have enough model capacity for more complex problems such as trajectory generation for autonomous vehicles. The second-best performing *Latent IRMv1* is also analogous to the linear IRM case, but it only uses intermediate results as supervision, so there are fitting errors introduced, resulting in slightly worse performance.

C. Comparative Study

We compare our proposed method with the recent trajectory generation method which aims to overcome the OOD challenge and other domain generalization methods. To be fair, all methods use the same inputs as our proposed method. For the discrete trajectory generation methods, we use linear interpolation to get the trajectory points at the corresponding moment. For methods that require domain labels, we divide three datasets into 5 domains respectively, according to the time of data collection. Note that our method does not require domain labels.

End-to-End Neural Trajectory (E2E NT) is a variant of [6], which has the same network structure and inputs as our proposed method. All its parameters are trained end-to-end with only L_2 loss. This method is treated as a baseline method in this paper.

Robust Imitative Planning (RIP) [4] is one of the state-of-the-art methods for trajectory generation to overcome distribution shifts. We use the *Worst Case Model (WCM)* over 5 models and Adam [27] optimizer for online optimization with 50 steps and 0.1 initial learning rate. According to the official code provided, the model output is 4 discrete trajectory points, and the points at other moments are obtained using linear interpolation.

MixStyle [28] is a method to make CNNs more domain-generalizable by mixing instance-level feature statistics of training samples across domains without using domain labels. We use this plugin in our end-to-end model without changing other settings for a fair comparison.

Domain Invariant Variational Autoencoders (DIVA) [3] is a generative model that tackles the domain generalization problem by learning three independent latent subspaces, one for the domain, one for the class, and one for any residual variations. We extend this method to turn the classification task into a trajectory generation task, using the continuous trajectory generation model proposed in [6]. This method needs domain labels in the training process.

Domain-Adversarial Learning (DAL) [29] methods leverage adversarial learning to allow the generator to extract domain invariant features. We use this method in our end-to-end model, and design a discriminator to determine whether two features come from the same domain. This method also needs domain labels in the training process.

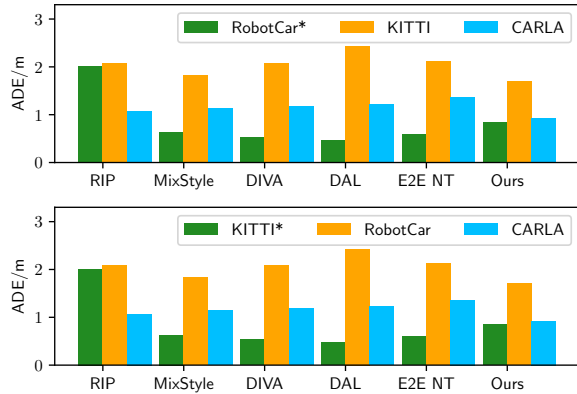


Fig. 3: Generalization performance (average displacement error in meters) on three different datasets. The models are trained on the dataset labeled by “*”, and directly generalize to the testing dataset and other two target datasets.

TABLE II: Closed-loop testing of success rate (% , the first one of each item) and average speed (m/s , the second one of each item) in CARLA. Higher metrics have better results. Highest success rates are highlighted in bold font.

Algorithm	Clear Noon	Wet Cloudy Sunset	Hard Rain Sunset	Heavy Fog Morning
RIP [4]	86/3.6	78/3.8	82/3.7	76/3.7
MixStyle [28]	94/4.1	77/4.9	72/7.4	34/6.4
DIVA [3]	79/9.6	71/9.3	54/8.7	39/7.5
DAL [29]	79/4.0	31/4.9	27/4.7	38/3.8
E2E NT [6]	100/8.6	50/4.5	35/3.9	31/9.9
Ours	94/5.6	82/6.8	100/7.6	79/5.7

D. Comparison Results on Datasets

We implement our network by using PyTorch 1.6 with CUDA 10.2 and cuDNN 7.6.5 libraries. We use a batch size of 32 and Adam [27] optimizer with an initial learning rate of 0.0003. All networks are trained on a PC with AMD 3700X CPU and NVIDIA RTX 2060 Super GPU. We train all methods on training dataset until the models converge, and evaluate them on the testing datasets from both the same domain and other domains.

The generalization results of the different models are shown in Fig 3. In terms of generalization performance, our method outperforms other comparison methods on the testing datasets under different domains, which validates that our method has a stronger generalization ability. The results show that *MixStyle* and *DIVA* these two domain generalization methods also have a great performance improvement compared to the *E2E NT* baseline method. While the *RIP* method and the *DAL* method do not always show a stable generalization performance advantage. Under certain conditions, the performance of these two methods may be worse than that of the baseline method.

E. Closed-loop Experiments in Simulation

Since we want to train our driving model on the dataset collected in one environment and transfer this model to a new environment to implement driving tasks, evaluation on

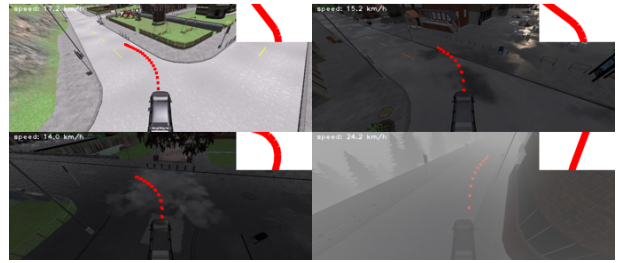


Fig. 4: Model generalization results of our method under four different weather conditions in CARLA. The model is only trained on RobotCar dataset and directly generalize to CARLA. The discrete red points are the generated trajectory points.

open-loop datasets is not enough to prove the effectiveness of our method. Therefore, we test our method with closed-loop visual navigation tasks in the CARLA [25] 0.9.9.4 simulation and compare it with other methods.

Experiments Setup. We train models on RobotCar Dataset and transfer them in the CARLA simulator, testing the driving success rates under different driving tasks of different models. We use the same vehicle and set random starting and target points in *Town01* with four different weather conditions: *Clear Noon*, *Cloudy Sunset*, *Hard Rain Sunset*, and *Heavy Fog Morning* and two traffic condition: *Empty* and *with Dynamic Obstacles*, where the setting of obstacles is the same as its in the CARLA benchmark [25].

Closed-loop Experiment Result. The results are shown in Tab. II and Fig. 4. Our method gets second place in success rate under *Clear Noon* weather condition and has the highest success rate under all other weather conditions. Compared to the *E2E NT* method, our method has high success rates in all weather conditions, while the success rates of *E2E NT* method vary greatly under different weather conditions, which means that our method has a stronger model transfer ability to handle different weather conditions. Compared to the *RIP* method, our method has higher success rates and higher average speeds, while the *RIP* method using *Worst Case Model (WCM)* generates more conservative trajectories with low speed. Compared to *MixStyle* and *DIVA*, which have high success rates under the former three weather conditions, our method can also get a high success rate under *Heavy Fog Morning* weather condition, where there is a close field of view and severe visual disturbance, which also validate our method has a stronger transfer ability.

V. CONCLUSION

In this paper, we propose a domain generalization method for vision-based driving trajectory generation for autonomous vehicles in urban environments, which can be seen as a solution to extend the IRM method in non-linear cases. We compare our proposed method with the state-of-the-art trajectory generation method and some recent domain generalization methods on both datasets and simulation, demonstrating that our method has better generalization ability.

REFERENCES

- [1] K. Zhou, Z. Liu, Y. Qiao, T. Xiang, and C. C. Loy, "Domain generalization: A survey," *arXiv preprint arXiv:2103.02503*, 2021.
- [2] Y. Li, X. Tian, M. Gong, Y. Liu, T. Liu, K. Zhang, and D. Tao, "Deep domain generalization via conditional invariant adversarial networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 624–639, 2018.
- [3] M. Ilse, J. M. Tomczak, C. Louizos, and M. Welling, "Diva: Domain invariant variational autoencoders," in *Medical Imaging with Deep Learning*, pp. 322–348, PMLR, 2020.
- [4] A. Filos, P. Tigkas, R. McAllister, N. Rhinehart, S. Levine, and Y. Gal, "Can autonomous vehicles identify, recover from, and adapt to distribution shifts?," in *International Conference on Machine Learning*, pp. 3145–3153, PMLR, 2020.
- [5] M. Arjovsky, L. Bottou, I. Gulrajani, and D. Lopez-Paz, "Invariant risk minimization," *arXiv preprint arXiv:1907.02893*, 2019.
- [6] Y. Wang, D. Zhang, J. Wang, Z. Chen, Y. Li, Y. Wang, and R. Xiong, "Imitation learning of hierarchical driving model: from continuous intention to continuous trajectory," *IEEE Robotics and Automation Letters*, 2021.
- [7] H. Li, S. J. Pan, S. Wang, and A. C. Kot, "Domain generalization with adversarial feature learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5400–5409, 2018.
- [8] F. M. Carlucci, P. Russo, T. Tommasi, and B. Caputo, "Hallucinating agnostic images to generalize across domains.," in *ICCV Workshops*, pp. 3227–3234, 2019.
- [9] J. Xing, T. Nagata, K. Chen, X. Zou, E. Neftci, and J. L. Krichmar, "Domain adaptation in reinforcement learning via latent unified state representation," *arXiv preprint arXiv:2102.05714*, 2021.
- [10] G. Kahn, A. Villaflor, V. Pong, P. Abbeel, and S. Levine, "Uncertainty-aware reinforcement learning for collision avoidance," *arXiv preprint arXiv:1702.01182*, 2017.
- [11] L. Tai, P. Yun, Y. Chen, C. Liu, H. Ye, and M. Liu, "Visual-based autonomous driving deployment from a stochastic and uncertainty-aware perspective," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2622–2628, IEEE, 2019.
- [12] D. Krueger, E. Caballero, J.-H. Jacobsen, A. Zhang, J. Binas, D. Zhang, R. Le Priol, and A. Courville, "Out-of-distribution generalization via risk extrapolation (rex)," in *International Conference on Machine Learning*, pp. 5815–5826, PMLR, 2021.
- [13] W. Jin, R. Barzilay, and T. Jaakkola, "Domain extrapolation via regret minimization," *arXiv preprint arXiv:2006.03908*, 2020.
- [14] E. Rosenfeld, P. Ravikumar, and A. Risteski, "The risks of invariant risk minimization," *arXiv preprint arXiv:2010.05761*, 2020.
- [15] K. Ahuja, K. Shanmugam, K. Varshney, and A. Dhurandhar, "Invariant risk minimization games," in *International Conference on Machine Learning*, pp. 145–155, PMLR, 2020.
- [16] A. Zhang, R. T. McAllister, R. Calandra, Y. Gal, and S. Levine, "Learning invariant representations for reinforcement learning without reconstruction," in *International Conference on Learning Representations*, 2021.
- [17] E. Rosenfeld, P. K. Ravikumar, and A. Risteski, "The risks of invariant risk minimization," in *International Conference on Learning Representations*, 2021.
- [18] A. Amini, G. Rosman, S. Karaman, and D. Rus, "Variational end-to-end navigation and localization," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 8958–8964, IEEE, 2019.
- [19] H. Ma, Y. Wang, L. Tang, S. Kodagoda, and R. Xiong, "Towards navigation without precise localization: Weakly supervised learning of goal-directed navigation cost map," *arXiv preprint arXiv:1906.02468*, 2019.
- [20] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.
- [21] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved training of wasserstein gans," *arXiv preprint arXiv:1704.00028*, 2017.
- [22] D. Barnes, M. Gadd, P. Murcutt, P. Newman, and I. Posner, "The oxford radar robotcar dataset: A radar extension to the oxford robotcar dataset," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6433–6438, IEEE, 2020.
- [23] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, "1 year, 1000 km: The oxford robotcar dataset," *The International Journal of Robotics Research*, vol. 36, no. 1, pp. 3–15, 2017.
- [24] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.
- [25] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Conference on robot learning*, pp. 1–16, PMLR, 2017.
- [26] P. Cai, Y. Sun, H. Wang, and M. Liu, "Vtgnnet: A vision-based trajectory generation network for autonomous vehicles in urban environments," *IEEE Transactions on Intelligent Vehicles*, 2020.
- [27] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
- [28] K. Zhou, Y. Yang, Y. Qiao, and T. Xiang, "Domain generalization with mixstyle," in *International Conference on Learning Representations*, 2021.
- [29] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *International conference on machine learning*, pp. 1180–1189, PMLR, 2015.