2020 IEEE International Conference on Robotics and Automation (ICRA)
31 May - 31 August, 2020. Paris, France

Navigation Command Matching for Vision-based Autonomous Driving

Yuxin Pan[1], Jianru Xue[1], Pengfei Zhang[1], Wanli Ouyang[2], Jianwu Fang[1] and Xingyu Chen[1]

Abstract— **Learning an optimal policy for autonomous driving task to confront with complex environment is a long-studied challenge. Imitative reinforcement learning is accepted as a promising approach to learn a robust driving policy through expert demonstrations and interactions with environments. However, this model utilizes non-smooth rewards, which have a negative impact on matching between navigation commands and trajectory (state-action pairs), and degrade the generalizability of an agent. Smooth rewards are crucial to discriminate actions generated from sub-optimal policy. In this paper, we propose a navigation command matching (NCM) model to address this issue. There are two key components in NCM, 1) a matching measurer produces smooth navigation rewards that measure matching between navigation commands and trajectory; 2) attention-guided agent performs actions given states where salient regions in RGB images (i.e. roadsides, lane markings and dynamic obstacles) are highlighted to amplify their influence on the final model. We obtain navigation rewards and store transitions to replay buffer after an episode, so NCM is able to discriminate actions generated from sub-optimal policy. Experiments on CARLA driving benchmark show our proposed NCM outperforms previous state-of-the-art models on various tasks in terms of the percentage of successfully completed episodes. Moreover, our model improves generalizability of the agent and obtains good performance even in unseen scenarios.**

I. INTRODUCTION

Learning an optimal driving policy to confront with complex and dynamic environment is an ongoing challenge in the field of intelligent vehicle. The traditional modular pipeline is not able to handle complex environments [1], [2] while reinforcement learning is prohibited to use in real-world tasks by its huge amount of exploration, and imitation learning-based methods [3], [4] are constrained by human driving data, which results in poor performance when behaviors are not included in the dataset. Thus, researchers attempt to make an agent learn robust policy via imitative reinforcement learning, by which the agent can explore in smaller action space after imitating expert demonstrations.

Imitative reinforcement learning based model trains an agent through both expert demonstrations and interactions with environments [5]. Due to the fact that imitative reinforcement learning is capable of reducing the amount of exploration and confining the actions within reasonable action space, it facilitates deep reinforcement learning and improves the performance significantly. However, we find that this model utilizes non-smooth rewards, which have a negative impact on matching between navigation commands and trajectory, and cause degradation of generalizability of autonomous driving agents. Smooth rewards are crucial to discriminate actions that are generated following sub-optimal policy from actions that are generated following optimal policy when interacting with various environments. Therefore, it is vital to discriminate the actions that are generated from sub-optimal policy and performed before navigation command mismatching to reduce the risk of navigation command mismatching. It is natural that the rewards generated from sub-optimal policy are less than rewards generated from optimal policy, but more than the rewards resulting from navigation command mismatching. Thus, a carefully designed reward function that implements the intuition above is important.

In addition, the self-driving agent should pay more attention on salient regions in RGB images for driving task, i.e. roadsides, lane markings, and dynamic obstacles (i.e. pedestrians and vehicles), to follow the navigation commands and avoid collisions with other things (e.g. vehicles, pedestrians, trees, and poles). Thus, it is desirable that the salient regions in RGB images should be highlighted for driving tasks to amplify their influence on the final model.

In this paper, we propose a navigation command matching (NCM) model to produce smooth navigation reward, which is based on controllable imitative reinforcement learning (CIRL) [5], where CIRL is an off-policy, replay-memory-based, actor-critic, and Deep Deterministic Policy Gradient [6] algorithm. Due to the fact that the rewards of CIRL are indiscriminate between actions generated from sub-optimal policy and actions generated from optimal policy, our NCM improves the discriminative capability by smoothing the rewards, which measures the matching between navigation commands (i.e. follow, straight, turn left and turn right) and trajectory. Because we obtain the navigation rewards and store transitions to replay buffer after an episode, NCM is able to discriminate actions generated from sub-optimal policy by smoothing rewards. And there is no negative impact on performance because of high-capacity of replay buffer. Moreover, previous works ignore to consider the influence of salient regions in RGB images on the behaviors of agent, we adopt the attention-guided agent into NCM to amplify the influence of these regions on the final model.

The main contributions of our work are summarized as:

- We propose a novel NCM model which produces smooth rewards to encourage matching between navigation commands and trajectory, that is, our model is able to discriminate actions that are generated following sub-

*This work was supported by National Natural Science Foundation of China (No. 61751308 and 61773311) and China Postdoctoral Science Foundation under Grant 2017M613152.

[1]The authors are with Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University, China. jrxue@mail.xjtu.edu.cn

[2]The author is with the University of Sydney, SenseTime Computer Vision Research Group, Australia. wanli.ouyang@sydney.edu.cn

optimal policy from actions that are generated following optimal policy.

- We adopt attention-guided agent into NCM whose attention guiding (AG) module highlights salient regions in RGB images, i.e. roadsides, lane markings and dynamic obstacles, for driving task to amplify their influence on the final model.
- Significant improvement of performance on CARLA benchmark demonstrates the effectiveness of our proposed NCM. Meanwhile, our proposed NCM can obtain good performance even in unseen scenarios.

II. Related Work

There are three major approaches for autonomous driving, i.e. modular pipeline, end-to-end approach, and direct perception approach. Modular pipeline splits the autonomous driving task into three smaller subtasks: perception, path planning and control [1], [2]. The end-to-end approach directly maps vision sensory inputs onto control outputs via supervised training [7], [8]. The direct perception falls in between modular pipeline and end-to-end approach. It maps raw vision inputs onto several intermediate affordances which are used to predict control outputs [9], [10]. Our method belongs to the second paradigm.

There exits some previous works [11], [12], [13], [4] which focused on vision-based self-driving using neural network. Pomerleau *et al.* first employed neural networks to imitate human behaviors from raw vision inputs [11]. Muller *et al.* utilized imitation learning to navigate an off-road truck to avoid obstacles in various scenarios [12]. In addition, there exists some other points of focus in this field. For instance, [14], [15] introduced different attention mechanisms; [16], [17] leveraged auxiliary tasks to assist the self-driving agent. [18], [19] attempted to interpret the behaviors of self-driving agent via visual explanations. However, they need huge number of training data which is expensive to collect and annotate. It is desirable to enable the agent to learn policy through interaction with environments.

Reinforcement learning is an effective method to enable the agent to interact with environments in a trial-and-error manner with no supervision. In recent years, deep reinforcement learning has achieved success in various tasks, including object recognition [20], computer games [21], robot control [22], autonomous driving [23], [24] and so on. CIRL [5] first applied reinforcement learning to learn robust driving policy after imitating expert demonstrations. However, the model is not able to produce smooth rewards. Smooth rewards are crucial to discriminate actions which are generated from sub-optimal policy and performed before navigation command mismatching. Our proposed NCM can effectively address this issue by smoothing the rewards. In addition, inspired by visual explanations of [18], [19], we adopt attention-guided agent into NCM whose attention guiding module highlights salient regions in RGB images for driving tasks to assist navigation.

Different novel reward functions of reinforcement learning are designed to deal with different tasks. As suggested by [25], the agent needs intermediate rewards for progress towards the goal by reward shaping. Recently, the model in [26] obtained dense rewards from reward network which is jointly trained from expert examples on GridLU. Wang *et al.* utilized the encoder-decoder framework to acquire dense rewards given natural language instructions and robot's trajectory [27]. However, these reward functions can not be applied to autonomous driving tasks. They fail to discriminate actions which are performed before navigation command mismatching and generated from sub-optimal policy. Our NCM smooths the rewards, and we obtain rewards and store transitions to replay buffer after an episode, which allows NCM to discriminate those actions.

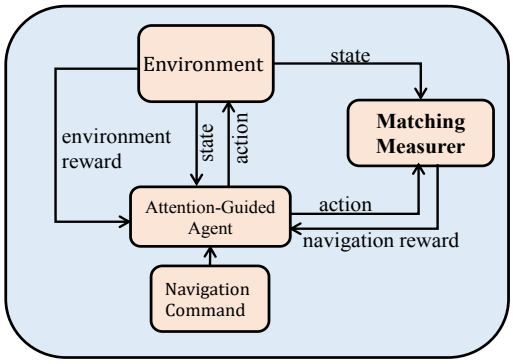

Fig. 1. The overview of NCM framework. The attention-guided agent interacts with environments given navigation command and matching measurer produces navigation rewards given state-action pairs.

III. Method

A. Overview

We consider a self-driving agent that learns to make driving decisions (i.e. steering angle, acceleration, braking) through the visual perception from camera and navigation commands (i.e. follow, straight, turn left, turn right). The navigation command matching (NCM) model mainly consists of two components: attention-guided agent and matching measurer, which is shown in Fig. 1. At time step t, the attention-guided agent receives the current state s_t and navigation command c_t, and then learns to perform an action a_t in order to reach the target location. After an episode, we can generate a trajectory τ (state-action pairs). In NCM, our rewards consist of environment rewards and navigation rewards. Environment rewards are provided by environment, which can indicate traffic infractions. Navigation rewards are produced by matching measurer, which can measure matching between navigation commands and trajectory.

B. Model

We first describe the inference process of attention-guided agent and matching measurer in NCM.

Attention-guided agent. The agent π_θ performs actions given the RGB images and navigation commands. Some salient regions in RGB images (i.e. roadsides, lane markings and dynamic obstacles) are closely related to driving task

and heavily influence decisions of the agent. Thus, we adopt attention-guiding agent into our framework which consists of attention guiding module and backbone network. The attention guiding module produces the attention-guided image. Then, we feed the attention-guided image, speed and navigation command to the backbone network and obtain the action outputs (i.e. steering angle, acceleration, braking). The overview of attention-guided agent is shown in Fig. 2.

Attention Guiding Module. The attention guiding module utilizes attention mask M to highlight salient regions in image. We employ attention function A to obtain the attention mask, which determines the category of each pixel in an image using semantic segmentation network [28] and sets the element of an attention mask as the probability of falling into those categories as mentioned above. The model of [28] can accurately classify multiple object categories at pixel level in real-time. Then, we employ Hadamard product between original image I^o and attention mask M to gain masked image I^m, and we add the original image I^o and the masked image I^m together to obtain attention-guided image I^a:

$$I^a = I^o \oplus \underbrace{(I^o \otimes \overbrace{A(I^o)}^{M})}_{I^m} \tag{1}$$

Backbone Network. We utilize the backbone network as [4] which is a shallow network. The attention-guided image and speed are fed to the backbone network, and the navigation command selectively activates different branches of the network (i.e. follow the lane (Follow), drive straight at the next intersection (Straight), turn left at the next intersection (TurnLeft), and turn right at the next intersection (TurnRight)).

Matching Measurer. In the course of interaction with environment, at time step t, the agent receives current observation s_t and navigation command c_t, and then learns to perform an action a_t. We utilize the matching measurer F^M which measures matching between navigation commands $c = \{c_t\}_{t=0}^{T}$ and trajectory $\tau = \{(s_t, a_t)\}_{t=0}^{T}$ to produce navigation rewards $R^{navi} = \{R_t^{navi}\}_{t=0}^{T}$:

$$R^{navi} = F^M(\tau, c) = F^M(\{s_t, a_t\}_{t=0}^{T}, c) \tag{2}$$

We always obtain the navigation rewards R^{navi} and store transitions $\{(s_t, a_t, r_t, s_{t+1})\}_{t=0}^{T}$, which represent that agent performs an action a_t from current state s_t to next state s_{t+1} and obtains the reward r_t, to replay buffer after an episode, so NCM is allowed to discriminate actions generated from sub-optimal policy by smoothing the rewards.

To measure matching, we measure the probabilities p of navigation commands c given the trajectory τ. The higher the probabilities, the better the conducted trajectory aligned with the navigation commands. Thus, we can define the navigation rewards as:

$$R^{navi} = p(c|\tau) = p(c|\pi_\theta(\{s_t\}_{t=0}^{T}, c)) \tag{3}$$

To measure the probabilities of navigation commands, we propose two solutions. 1) We adopt Gaussian-distributed

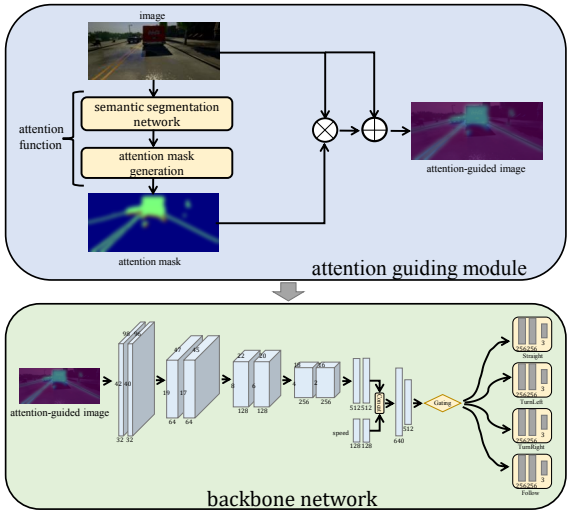

Fig. 2. The overview of attention-guided agent where raw RGB image is fed to attention guiding module to produce attention-guided image which is fed to backbone network to predict control outputs.

weights to smooth the initial probabilities of navigation commands given sequential actions. The initial probabilities of navigation commands are from prior statistical knowledge of the dataset [4]. 2) We employ bi-directional LSTM (BL) [29] to produce the distribution of each navigation command, which measures the probabilities of navigation commands given trajectory.

Gaussian-weighted Probabilities (GW) of Navigation Commands. After an episode, we can obtain sequential steer-angle actions which denote the probabilities of navigation commands. We use Gaussian-distributed weights W^G to smooth the initial probabilities $P^{c^i} = \{p_t^{c^i}\}_{t=t^*-T_1}^{t=t^*+T_2}$ of navigation commands $c^i, i \in (1,2,3,4)$ (i represents four types of navigation commands) for getting the probability $p(c_{t^*}^i|a_{t^*}^s)$ of navigation command $c_{t^*}^i$ at time step t^*. Due to the fact that we take the probabilities $\{p_t^{c^i}\}_{t=t^*}^{t=t^*+T_2}$ of navigation commands into consideration when smoothing the rewards, NCM is able to discriminate actions which are generated from sub-optimal policy and performed before navigation command mismatching. The smoothing equation is shown as:

$$p(c_{t^*}^i|a_{t^*}^s) = \sum_{t=t^*-T_1}^{t^*+T_2} p_t^{c^i} \times W_{t-t^*}^G \tag{4}$$

where $W^G \sim N(0,\sigma)$, and T_1+T_2 is the truncated length of W^G. Then we use softmax function to get the final probabilities of navigation commands and navigation rewards at time step t^* as:

$$R^{navi} = p(c_{t^*}^i|a_{t^*}^s) = \frac{e^{p(c_{t^*}^i|a_{t^*}^s)}}{\sum_{i=1}^{i=4} e^{p(c_{t^*}^i|a_{t^*}^s)}} \tag{5}$$

To obtain the initial probabilities of navigation commands, we divide the steer-angle set in dataset of [4] into 20 bins with 0.1 as the step size. We count up the number of each

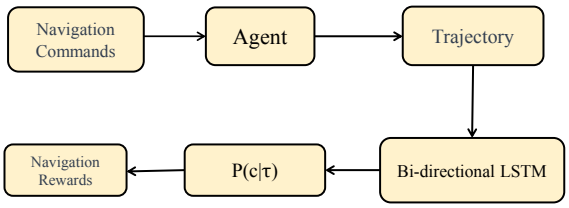

Fig. 3. The pipeline of matching measurer using bi-directional LSTM. The agent produces trajectory (state-action pairs) given navigation commands. Then, bi-directional LSTM measures probabilities between trajectory and navigation commands. We take the probabilities as navigation rewards.

navigation command in each bin. And then we normalize them with summation of the number in each bin.

Bi-directional LSTM. There exits consistency between states s received by agent π_θ and navigation commands c. So we utilize the trajectory (state-action pairs) $\{(s_t, a_t)\}_{t=0}^{T}$ as the inputs of matching measurer (T is the length of an episode). Because we need discriminate actions which are generated from sub-optimal policy and performed before the navigation mismatching, we utilize bi-directional LSTM F^b which not only include the contributions of previous trajectory to current reward, but also the contributions of future trajectory. The pipeline of matching measurer using bi-directional LSTM is shown in Fig. 3.

The agent receives the current state s_t and navigation command c_t, and then performs an action a_t. After an episode, we generate a trajectory τ (state-action pairs) and feed the trajectory to bi-directional LSTM. Then, the bi-directional LSTM produces the probabilities of navigation commands. We take the probability $p(c_t^i|\tau)$ of current navigation command c_t^i as the current navigation reward R_t^{navi}:

$$R^{navi} = p(c_t^i|\tau)|_{t=0}^{T} = F^b(\tau) = F^b(\pi_\theta(s_t, c_t)|_{t=0}^{T}) \quad (6)$$

The state s_t is a 512-dimentional vector encoded by backbone network from an image and the action that consists of steering angle, acceleration and braking is also encoded to a 512-dimentional vector. Then, we concatenate them to obtain feature vector of trajectory τ. Bi-directional LSTM is fed the feature vector of trajectory τ to produce probabilities of navigation commands. The architecture of matching measurer using bi-directional LSTM is shown in Fig. 4.

Environment Rewards. We require the self-driving agent to complete an episode without traffic infractions. Thus, we need introduce environment rewards R^{envi} to our framework for indicating traffic infractions. The traffic infractions consist of collisions with other vehicles, pedestrians and other things (e.g. trees and poles), overlapping with sidewalk and opposite lane. Once there is a traffic infraction, the environment reward is set as -1 otherwise 1.

C. Training

We utilize the same training strategy as [5], which uses imitation learning to initialize the backbone network and then trains the agent through reinforcement learning. Here we discuss how to train the modules of NCM in details.

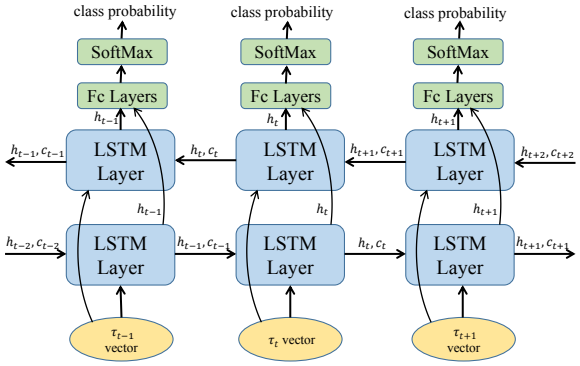

Fig. 4. The architecture of matching measurer using bi-directional LSTM where feature vectors of trajectory τ are fed to bi-directional LSTM to produce probabilities of navigation commands.

Semantic Segmentation Network. The semantic segmentation network [28] is trained with semantic segmentation dataset in which each pixel in image is labeled with the class of its enclosing object or region. We utilize cross entropy as the loss function defined as:

$$L_a = -\sum_{i=1}^{H}\sum_{j=1}^{W} \log p_{i,j}(l_{i,j}^s) \quad (7)$$

where H and W denote the height and width of an image, $p_{i,j}$ indicates the predicted probability of the pixel in the position (i, j), and $l_{i,j}^s$ is the ground-truth label in the position (i, j).

Backbone Network. The backbone network F is trained via imitation learning based on the given N expert driving demonstrations v_i, $i \in \{(1, ..., N)\}$, which consist of attention-guided images $I_{i,t}^a$, control commands $c_{i,t}$, speeds $s_{i,t}$, and actions $a_{i,t}$. The action $a_{i,t}$ contains three continuous actions: steering angle $a_{i,t}^s$, acceleration $a_{i,t}^a$, and braking $a_{i,t}^b$. The command $c_{i,t}$ controls the selective branch activation via the gating function $G(c_{i,t})$.

The objective loss function of backbone is defined as follows:

$$L_b = \sum_{i}^{N}\sum_{t}^{T_i} L(F(I_{i,t}^o, G(c_{i,t}), s_{i,t}), a_{i,t}) \quad (8)$$

We use the weighted summation of L2 losses for three predicted actions $\hat{a}$ (i.e. steering angle, acceleration, braking):

$$L(\hat{a}, a) = \sum_{a \in (a^s, a^a, a^t)} w_a \|a - \hat{a}\|^2 \quad (9)$$

Bi-directional LSTM. Bi-directional LSTM is trained with ground-truth trajectory-command pairs $\{(\hat{\tau}_k, \hat{c}_k)\}_{k=1}^{N}$, via supervised learning from dataset of [4]. We use cross entropy as loss function defined as:

$$L_c = -\sum_{k=1}^{N}\sum_{t=0}^{T_i} log\ p(\hat{c}_{k,t}|\hat{\tau}_{k,t}) \quad (10)$$

TABLE I

QUANTITATIVE COMPARISONS WITH SOTA MODELS ON FOUR NAVIGATION TASKS, I.E. STRAIGHT (T_1), ONE TURN (T_2), NAVIGATION (T_3) AND NAVIGATION WITH DYNAMIC OBSTACLES (T_4), IN TERMS OF THE PERCENTAGE (%) OF SUCCESSFULLY COMPLETED EPISODES.

model	Training Conditions				New Town				New Weather				New Town&New Weather			
	T_1	T_2	T_3	T_4	T_1	T_2	T_3	T_4	T_1	T_2	T_3	T_4	T_1	T_2	T_3	T_4
MP [3]	98	82	80	77	92	61	24	24	100	95	94	89	50	50	47	44
RL [3]	89	34	14	7	74	12	3	2	86	16	2	2	68	20	6	4
CIRL [5]	98	97	93	82	**100**	71	53	41	100	94	86	80	98	82	68	62
CAL [10]	**100**	97	92	83	93	82	70	**64**	100	**96**	90	82	94	72	68	64
MT [17]	98	87	81	81	**100**	81	72	53	100	88	88	80	96	82	**78**	62
baseline (IL [3])	95	89	86	83	97	59	40	38	98	90	84	82	80	48	44	42
baseline + AG	93	93	93	81	96	73	53	41	100	90	96	78	98	72	70	54
baseline + NCM (GW)	96	92	96	89	96	79	58	41	100	94	96	90	96	78	70	58
baseline + NCM (BL)	98	**98**	**99**	**98**	96	**84**	**74**	56	**100**	94	**96**	**92**	**100**	**84**	**78**	**76**

where $\{p(\hat{c}_{k,t}|\hat{\tau}_{k,t})\}_{t=0}^{T_i} = F^b(\hat{\tau}_i)$, and the $\hat{\tau}_{k,t}$ is the state-action pair in $\hat{\tau}_k$ at time step t, the $\hat{c}_{k,t}$ is the navigation command in $\hat{c}_k$ at time step t.

Reinforcement Learning Policy Gradient. With both navigation rewards and environment rewards, we obtain the total rewards $R^{total}(R^{navi} + R^{envi})$ and define the objective function of reinforcement learning as:

$$\max_{\theta} J(\theta) = E_{a\sim\pi_\theta}[R^{total}] \tag{11}$$

Since the self-driving agent predicts continuous actions (i.e. steering angles, acceleration and braking) and we utilize the actor-critic approach as [5], the gradient of reward-based loss function and gradient-updating function can be respectively derived as:

$$\nabla_{\theta^\pi} J(\theta^\pi) = E[\nabla_a Q(s, a; \theta^Q)_{|a=\pi(s,c)} \nabla_{\theta_\pi} \pi(s, c; \theta^\pi)] \tag{12}$$

$$\theta^\pi + \alpha \nabla_{\theta^\pi} J(\theta^\pi) \rightarrow \theta^\pi \tag{13}$$

where $Q(s, a; \theta^Q)$ is the evaluation of state-action pair (s, a) produced by critic network θ^Q, and α is the learning rate of the agent network.

IV. EXPERIMENTS

A. Experimental Setup

Evaluation Benchmark. We evaluate our proposed NCM on the CARLA simulator benchmark [3]. CARLA provides a large variety of assets, including cars and pedestrians. There are two Towns in CARLA simulator: Town 1 and Town 2. For fair comparison with other approaches [3], [5], [10], [17], we use Town 1 for training and Town 2 for testing. The weather set consists of two groups, including Training Weather set and New Weather set. Training weather set consists of clear noon, clear sunset, hard rain noon and noon after rain. New weather set consists of cloudy noon after rain and soft rain at sunset. Our test settings are same as IL [4] and CIRL [5].

State-of-the-art Pipelines. We compare our model with six SOTA methods on CARLA benchmark: modular pipeline (MP) [3], imitation learning (IL) [3], reinforcement learning (RL) [3], controllable imitative reinforcement learning (CIRL) [5], conditional affordance learning (CAL) [10], and multi-task learning (MT) [17]. We evaluate our model on four increasingly difficult driving tasks, i.e. Straight (T_1), One turn (T_2), Navigation (T_3) and Navigation with dynamic obstacles (T_4). In addition, we adopt the same backbone network as IL [3] for fair comparison. But the backbone networks of CAL and MT are VGG16 [30] and ResNet [31] respectively, which are better than ours. For all methods, the agent cannot be fine-tuned respectively for each task. For each combination of a task, a town, and a weather set, the paths are carried out 25 episodes. In each episode, an agent is initialized somewhere in town and need reach a goal location. An episode is considered successful if the agent reaches the goal within a time budget, which is set to reach the goal along the optimal path at a speed of 10 km/h. Thus, we select the percentage of successfully completed episodes as the metric.

Implementation Details. During the attention-guided imitation learning, we use ErfNet [28] to produce the attention mask. And we collected the semantic segmentation dataset from CARLA simulator, which contains 2400 images and fairly covers four weathers as training condition. For training backbone network, we use the same experiment settings and dataset as [4] for fair comparison.

During the reinforcement learning, the initial learning rate of actor network is set as 0.00001 and the learning rate of critic network is set as 0.001. Learning rate is linearly decreased to zero over the course of training. The actor-critic networks are trained with 700 episodes (about 0.3 million simulation steps). Further details are referred in [5].

For the experiment settings of matching measurer with Gaussian-weighted probabilities of navigation commands, the standard deviation of Gaussian weights is 0.5 and we set truncated length T_1 and T_2 as 37 and 38 respectively.

For the experiment settings of matching measurer with bi-directional LSTM, we obtain the ground-truth trajectory-command pairs from the dataset [4]. The initial learning rate is set as 0.00001 and exponentially decreased. We set the decay rate and decay step as 0.5 and 10000, respectively. We use Adam optimizer to train the model 200000 steps,

TABLE II

ABLATION STUDY ON SEEN AND UNSEEN SETTINGS IN TERMS OF THE PERCENTAGE (%) OF SUCCESSFULLY COMPLETED EPISODES.

model	Training Conditions				New Town				New Weather				New Town&New Weather			
	T_1	T_2	T_3	T_4	T_1	T_2	T_3	T_4	T_1	T_2	T_3	T_4	T_1	T_2	T_3	T_4
CIRL [5]	98	**97**	93	82	**100**	71	53	41	**100**	**94**	86	80	98	82	68	62
CIRL [5] + MM (GW)	98	95	90	80	95	79	56	46	96	90	86	84	96	76	70	68
CIRL [5] + MM (BL)	98	96	**99**	**91**	96	**84**	**64**	**49**	96	**94**	**92**	**90**	**100**	**84**	**78**	**74**
CIRL [5] + AG	95	92	92	88	94	69	54	48	96	92	86	78	**100**	82	62	56
CIRL [5] + AG + MM (GW)	96	92	96	89	**96**	79	58	41	**100**	94	**96**	90	96	78	70	58
CIRL [5] + AG + MM (BL)	**98**	**98**	**99**	**98**	**96**	**84**	**74**	**56**	**100**	**94**	**96**	**92**	**100**	**84**	**78**	**76**

and we set batch size and time step length of LSTM as 16 and 75 respectively. The dimension of hidden state in LSTM is set as 512.

B. Comparisons with State-of-the-arts

Table I reports the comparisons with SOTA pipelines on CARLA benchmark in terms of the percentage of successfully completed episodes under four conditions. For training conditions, the models are tested on the combination of town 1 and training weather which has the same environments and conditions as the training stage. The settings including new town (town 2) or new weather are conducted for evaluating generalization.

We can observe that our NCM with bi-directional LSTM (BL) outperforms all previous methods and significantly diminishes the performance gap between seen and unseen environment. On the seen settings, our NCM (BL) fully outperforms previous models, e.g. 98% of NCM (BL) vs. 83% and 82% of CAL and CIRL. Furthermore, our NCM (BL) shows superior generalization capabilities on the unseen settings, which obtains better performance over other models, e.g. 76% of NCM (BL) vs. 64% and 62% of CAL and CIRL, respectively. For the Navigation task (T_3) between training conditions and new weather, the performance gap of our NCM (BL) is 2%, but that of CIRL is 7%. However, generalization to a new town also presents a challenge because of new textures and 3D models. This problem can be mitigated by utilizing a better feature extractor and training in various environments.

Note that NCM with bi-directional LSTM (BL) outperforms NCM with Gaussian weight (GW). Because the NCM (GW) is a linear weighting method which may ignores the far but important actions that contribute to discrimination on actions generated from sub-optimal policy. NCM (BL) is a no-linear method which can measure matching between trajectory and navigation command better.

It is also interesting that IL with attention guiding (AG) has better performance than IL, especially the generalization capabilities on the unseen settings, e.g. 70% of IL with AG vs. 44% of IL, which demonstrates this simple module effectively promotes the generalization capabilities.

More qualitative results are shown in our supplementary materials, which provides qualitative comparisons among baseline (IL [3]), baseline with AG, baseline with NCM (GW) and baseline with NCM (BL).

C. Ablation Study

We conduct an ablation study to compare the effect of matching measurer (MM) with Gaussian weights (GW) and bi-directional LSTM (BL) shown in Table II. Comparing Row 1 with Row 3, we observe the CIRL with MM (BL) outperforms the CIRL on both seen and unseen setting. And the generalization of the CIRL with MM (BL) is promoted, e.g. 74% of CIRL with MM (BL) vs. 62% of CIRL. Comparing Row 2 with Row 3, we observe that the performance of CIRL with MM (BL) is better than that of CIRL with MM (GW) in most tasks, which also demonstrates the bi-directional LSTM can measure matching between trajectory and navigation commands better.

Furthermore, we can observe the influence of MM on CIRL with AG. We obtain the same results as above. The MM can significantly promote the performance of CIRL with AG and can significantly diminish the performance gap between seen and unseen environments. For the One Turn task (T_2) between training conditions and new town, the performance gap of CIRL with AG and MM is 14%, but that of CIRL with AG is 23%.

Comparing Row 1 with Row 4, we observe that the performance of CIRL is better than CIRL with AG in most tasks. But the performance of IL with AG is better than IL shown in Table I. We reckon that it's really an abnormal result that reinforcement learning degrades the performance of original model. The only rational reason is that the reward function of [5] is not suitable for our attention-guided model. However, our MM can produce reliable rewards which can promote the performance of all models.

V. CONCLUSION

In this paper, we proposed a novel approach, navigation command matching (NCM), which leverages bi-directional LSTM to discriminate the actions generated from sub-optimal policy from actions generated from optimal policy by smoothing rewards for vision-based self-driving. Experiment shows the effectiveness and efficiency of our model in both seen and unseen environments. And our NCM diminishes the performance gap between seen and unseen environments, which demonstrates NCM promotes capability of generalization. Matching measurer can be migrated into other tasks that can be solved by reinforcement learning. We think that the immediate and smooth rewards produced by neural network are necessary for promoting performance of agent.

REFERENCES

[1] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, *et al.*, "Stanley: The robot that won the darpa grand challenge," *Journal of Field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.

[2] J. Ziegler, P. Bender, M. Schreiber, H. Lategahn, T. Strauss, C. Stiller, T. Dang, U. Franke, N. Appenrodt, C. G. Keller, *et al.*, "Making bertha drivean autonomous journey on a historic route," *IEEE Intelligent Transportation Systems Magazine*, vol. 6, no. 2, pp. 8–20, 2014.

[3] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Conference on Robot Learning*, 2017, pp. 1–16.

[4] F. Codevilla, M. Miiller, A. López, V. Koltun, and A. Dosovitskiy, "End-to-end driving via conditional imitation learning," in *IEEE International Conference on Robotics and Automation*. IEEE, 2018, pp. 1–9.

[5] X. Liang, T. Wang, L. Yang, and E. Xing, "Cirl: Controllable imitative reinforcement learning for vision-based self-driving," in *Proceedings of the European Conference on Computer Vision*, 2018, pp. 584–599.

[6] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *International Conference on Learning Representations*, 2016.

[7] D. Silver, J. A. Bagnell, and A. Stentz, "Learning from demonstration for autonomous navigation in complex unstructured terrain," *The International Journal of Robotics Research*, vol. 29, no. 12, pp. 1565–1592, 2010.

[8] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 33–55, 2016.

[9] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "Deepdriving: Learning affordance for direct perception in autonomous driving," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2722–2730.

[10] A. Sauer, N. Savinov, and A. Geiger, "Conditional affordance learning for driving in urban environments," in *Conference on Robot Learning*, 2018, pp. 237–252.

[11] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," in *Advances in Neural Information Processing Systems*, 1989, pp. 305–313.

[12] U. Muller, J. Ben, E. Cosatto, B. Flepp, and Y. L. Cun, "Off-road obstacle avoidance through end-to-end learning," in *Advances in Neural Information Processing Systems*, 2006, pp. 739–746.

[13] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[14] Y. Xia, J. Kim, J. Canny, K. Zipser, T. Canas-Bajo, and D. Whitney, "Periphery-fovea multi-resolution driving model guided by human attention," in *IEEE Winter Conference on Applications of Computer Vision*, 2020, pp. 1767–1775.

[15] D. Wang, C. Devin, Q.-Z. Cai, F. Yu, and T. Darrell, "Deep object-centric policies for autonomous driving," in *International Conference on Robotics and Automation*, 2019, pp. 8853–8859.

[16] H. Xu, Y. Gao, F. Yu, and T. Darrell, "End-to-end learning of driving models from large-scale video datasets," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 2174–2182.

[17] Z. Li, T. Motoyoshi, K. Sasaki, T. Ogata, and S. Sugano, "Rethinking self-driving: Multi-task knowledge for better generalization and accident explanation ability," *arXiv preprint arXiv:1809.11100*, 2018.

[18] J. Kim and J. Canny, "Interpretable learning for self-driving cars by visualizing causal attention," in *Proceedings of International Conference on Computer Vision*, 2017, pp. 2942–2950.

[19] M. Bojarski, P. Yeres, A. Choromanska, K. Choromanski, B. Firner, L. Jackel, and U. Muller, "Explaining how a deep neural network trained with end-to-end learning steers a car," *arXiv preprint arXiv:1704.07911*, 2017.

[20] X. Liang, L. Lee, and E. P. Xing, "Deep variation-structured reinforcement learning for visual relationship and attribute detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 848–857.

[21] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[22] G. Endo, J. Morimoto, T. Matsubara, J. Nakanishi, and G. Cheng, "Learning cpg-based biped locomotion with a policy gradient method: Application to a humanoid robot," *The International Journal of Robotics Research*, vol. 27, no. 2, pp. 213–228, 2008.

[23] X. Pan, Y. You, Z. Wang, and C. Lu, "Virtual to real reinforcement learning for autonomous driving," *arXiv preprint arXiv:1704.03952*, 2017.

[24] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "Safe, multi-agent, reinforcement learning for autonomous driving," in *NIPS Workshop on Learning, Inference and Control of Multi-Agent Systems*, 2016.

[25] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Sixteenth International Conference on Machine Learning*, vol. 99, 1999, pp. 278–287.

[26] D. Bahdanau, F. Hill, J. Leike, E. Hughes, A. Hosseini, P. Kohli, and E. Grefenstette, "Learning to understand goal specifications by modelling reward," in *International Conference on Learning Representations*, 2019.

[27] X. Wang, Q. Huang, A. Celikyilmaz, J. Gao, D. Shen, Y.-F. Wang, W. Y. Wang, and L. Zhang, "Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 6629–6638.

[28] E. Romera, J. M. Alvarez, L. M. Bergasa, and R. Arroyo, "Erfnet: Efficient residual factorized convnet for real-time semantic segmentation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 1, pp. 263–272, 2017.

[29] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm and other neural network architectures," *Neural Networks*, vol. 18, no. 5-6, pp. 602–610, 2005.

[30] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015.

[31] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.