End-To-End Deep Reinforcement Learning for First-Person Pedestrian Visual Navigation in Urban Environments

Honghu Xue[1], Rui Song[1,2], Julian Petzold[2], Benedikt Hein[3], Heiko Hamann[2] and Elmar Rueckert[4]

***Abstract*— We solve a pedestrian visual navigation problem with a first-person view in an urban setting via deep reinforcement learning in an end-to-end manner. The major challenges lie in severe partial observability and sparse positive experiences of reaching the goal. To address partial observability, we propose a novel 3D-temporal convolutional network to encode sequential historical visual observations, its effectiveness is verified by comparing to a commonly-used Frame-Stacking approach. For sparse positive samples, we propose an improved automatic curriculum learning algorithm NavACL$^+$, which proposes meaningful curricula starting from easy tasks and gradually generalizing to challenging ones. NavACL$^+$ is shown to facilitate the learning process with 21% earlier convergence, to improve the task success rate on difficult tasks by 40% compared to the original NavACL algorithm [1] and to offer enhanced generalization to different initial poses compared to training from a fixed initial pose.**

I. INTRODUCTION

Auxiliary advanced driver assistance systems have been increasingly developed for vulnerable road users besides intelligent autonomous vehicles for road safety issues, with previous efforts focusing on pedestrian behavior or vision prediction problems [2], [3]. On the other hand, Deep Reinforcement Learning (DRL) has been extensively investigated in navigation tasks [4], [5]. A major benefit of DRL is learning from scratch and even without map knowledge. In this work, we address a mapless visual navigation problem in an urban setting, where a pedestrian (agent) is merely provided with first-person greyscale visual images and learns to cross an intersection safely. The key challenges of this task lie in the scarcity of positive learning signals (class imbalance problems) and severe partial observability. We investigate if DRL approaches can solve the task and how the learned policy can generalize to different initial poses.

Partial observability (PO) refers to the settings where the agent cannot perceive the whole environment. In a typical first-person visual navigation task with no map knowledge, PO greatly raises the task difficulty in two aspects. The agent merely relies on a restricted frontal view where goals are often absent in visual information. Secondly, even with a 360° view, PO still exists as the goal can be obstructed by other objects or simply too small to be spotted when being distant. From DRL perspectives, PO also jeopardizes the learning procedure as classical algorithms are premised on the Markov Decision Process that the agent gains full knowledge on environments [6].

Similar to Deep Learning, DRL algorithms can only learn well given both positive and negative learning signals, i.e., without significant class imbalance problems. Balancing multiple learning signals can improve the sample efficiency [6]. In goal-reaching tasks, vital learning signals are those reaching the terminal states, which occur with low probability when starting with a random policy. To enhance the number of goal-reaching experiences, various approaches such as efficient exploration or Curriculum Learning (CL) [7] have been proposed. CL has been shown to facilitate the learning process, i.e., increase the sample efficiency by proposing simple tasks and gradually increasing the difficulties [8], [9]. Automatic Curriculum Learning (ACL) further automates curricula proposal steps instead of hardcrafting tasks and manually determining how to form curricula.

The contribution of this work is to propose an improved ACL algorithm called NavACL$^+$. We show that NavACL$^+$ can effectively improve the training performance and lead to a reasonable curriculum compared to an existing method NavACL [1]. It also significantly outperforms training from fixed start in generalizing to different initial poses. The validation task also pinpoints the assumption. Secondly, we propose a 3D Temporal Convolutional Network with additional historical rewards and actions to address the problem of PO in a first-person visual navigation task. The proposed network architecture is compared with a commonly-used Frame-Stacking encoder for visual inputs in [10], [11]. Finally, we show that our modified Distributional Soft Actor-Critic algorithm [12] can successfully be applied to a mapless visual navigation task featuring high-dimensional inputs in an end-to-end learning fashion, whereas the original work only shows its effectiveness in problems with low-dimensional inputs like *Ants* in Mujoco [13].

II. RELATED WORK

Extensive research efforts have been dedicated to solving visual navigation tasks using DRL. In [14], [15], they deal with a visual navigation task in a target-driven manner, where the agent is additionally provided with the image of the goal, seen as a generalization on the goal state. In [14], a pre-trained encoder is used to extract the features from visual data for domain generalization. The work [16] combines unsupervised learning of expert demonstration features and

[1]Institute for Robotics and Cognitive Systems, University of Luebeck, Luebeck; xue@rob.uni-luebeck.de, rui.song@student.uni-luebeck.de

[2]Institute of Computer Engineering, University of Luebeck, Luebeck; petzold@iti.uni-luebeck.de, hamann@iti.uni-luebeck.de

[3]Institute of Automation Technology, Helmut Schmidt University, Hamburg; benedikt.hein@hsu-hh.de

[4]Chair of Cyber-Physical-Systems, Montanuniversität Leoben, Leoben; rueckert@unileoben.ac.at

DRL to improve the policy on raw-pixel inputs. Others [17] apply end-to-end DRL in a map-based fashion, where actions are learned from the local occupancy map along with high-level goal information. However, the aforementioned approaches either break the assumption of end-to-end learning without prior knowledge, thus simplifying the task difficulties, or require costly efforts to access map knowledge, which is not always convenient in real scenarios. To realize end-to-end learning, sufficient meaningful learning signals must be provided to the agent. One way to achieve that is ACL, as tasks with properly-scaled difficulty facilitate the training compared to random tasks [1].

A. Automatic Curriculum Learning in Reinforcement Learning Domains

As pointed out in [8], one category of ACL algorithms shifts the initial or terminal state distribution. The work [18] proposes reverse curriculum expansion for goal reaching tasks, where the training starts from initial states close to the goal to raise the task success rate. A scheduler will determine when to increase the initial-goal distance based on the estimated return of the candidate initial states. A random walk is performed on the current initial state set to guarantee all initial states are valid. Another idea is to perform automatic goal generation from close to distant ones [19], [20], where the initial state is fixed. These works are based on the idea of goal-conditioned policies, where the state space also includes the goal information as first mentioned in the work [21]. In [20], a generative-adversarial network [22] is trained to generate goals of appropriate difficulty and the generated goals form a curriculum, whereas in [19] similar ideas are employed, they use a *settler* and *judge* for proposing goals under the criterion of *goal validity*, *feasibility* and *diversity*. In this work, we generalize and improve the novel reverse curriculum expansion algorithm NavACL [1]. NavACL is shown to greatly enhance the training performance on visual navigation tasks and outperform the approach in [20]. The details are elaborated in Section III-A.

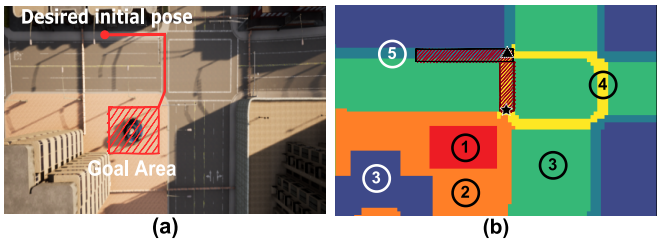

Fig. 1. The designed training environment in CARLA. (a) A bird's view on the urban scene (training environment). The red line represents an exemplary trajectory. (b) The corresponding reward map with each color being a different reward. The slashed area is the curriculum regions of NavACL$^+$ and NavACL, while the black star represents the easiest initial pose before entering ACL. For Non-ACL trials, we set the initial location as marked by the black triangle. Note that we have already simplified the task for Non-ACL variants, as they mostly fail to reach the goal when trained from the desired initial pose, which is more distant to the goal.

B. Alleviating Partial Observability in DRL

Another major challenge in visual navigation is PO. One common way of mitigating this issue is to maintain historical state information for decision making. The most common approach is to stack historical visual frames along the channel dimension and then use regular 2D-convolutions to extract features. The works [10], [11] perform this trick in a few Atari games. Some other work [23] processes historical visual data by using a Long-Short-Term Memory [24].

An alternative to tackle temporal data instead of recurrent network is to use temporal convolution networks (TCNs) [25]. TCNs avoid the gradient vanishing or exploding problem in recurrent networks by applying stacked dilated convolutions and also greatly reduce the number of learnable parameters compared to standard causal convolution. TCNs have been shown to handle long sequence data better than recurrent networks [26].

III. Methods

Here we present the details of our ACL method NavACL$^+$ as well as the network architecture to respectively deal with the challenge of sparse positive experiences and PO in a mapless first-person visual navigation task. We also show the interaction of ACL with one state-of-the-art DRL algorithm Distributional Soft Actor-Critic [12] to achieve end-to-end learning.

A. Automatic Curriculum Learning: NavACL

We start with the general paradigm of NavACL [1], from which our algorithm NavACL$^+$ is extended. In NavACL, a curriculum is generated by proposing initial states. Let the task $h = \phi(s_0,\ s_g)$, where s_0 and s_g represent the initial and goal state, and $\phi(\cdot)$ is a set of handcrafted features mapping the states to a feature space. A success prediction network is trained to predict the success rate $f^s_\pi(h)$ of reaching the goal following policy π on the task h. The superscript s denotes the success rate, which will be distinguished from our approach. A number of candidate tasks are generated randomly and classified into three categories: *easy*, *frontier*, and *random* via computing the mean μ_f and the standard deviation σ_f of $f^s_\pi(h)$ from all candidates and then thresholding to get their class types. Specifically, tasks fulfilling the condition $f^s_\pi(h) > \mu_f + \beta^h_e \sigma_f$ or condition $\mu_f - \beta^l_f \sigma_f < f^s_\pi(h) < \mu_f + \beta^h_f \sigma_f$ are categorized as *easy* or *frontier* respectively, where β^h_e, β^h_f and β^l_f are hyperparameters to determine the thresholds for task types. This procedure is called *Adaptive Filtering* in [1]. *Easy* tasks consolidate the knowledge acquired by the agent and prevent catastrophic forgetting [8], while the *frontier* tasks are the challenging ones given agent's current ability. Then, a scheduler determines which type of task is selected for the new episode by assigning a probability to each task type. The exact details are demonstrated the original work [1].

B. Automatic Curriculum Learning: NavACL$^+$

Our work improves the original NavACL algorithm in three aspects. The first enhancement is to train a network f^r_π

to predict the discounted return $G_{t:T}$ instead of the success rate.

$$G_{t:T} = \begin{cases} \sum_{k=0}^{T-t-1} \gamma^k r_{t+k+1} + \gamma^{T-t} f_\pi^r(h_T), \text{if } T > T_{max}, \\ \sum_{k=0}^{T-t} \gamma^k r_{t+k+1} \quad , \text{elsewise}, \end{cases}$$

where t, T and T_{max} denote the current time step, terminal time horizon and the maximal episodic length. This results in a more precise evaluation of the agent's current performance on the task, as the success rate can not reflect the spent time horizon to reach the goal. For these reasons, to further characterize the quality of the current policy π, we replace $f_\pi^s(h)$ with the discounted return $f_\pi^r(h)$.

The second improvement is to generate more data samples for training f_π^r. In NavACL, the network f_π^s is trained only using the initial states, causing a sparse training set, which easily causes overfitting and training instabilities. To improve the sample efficiency, we train f_π^r on an augmented dataset that considers, in addition to the initial task $h = (s_0, s_g)$, all augmented tasks $h = \{\phi(s_1, s_g), ...\phi(s_T, s_g)\}$ along the trajectory in the current episode, where a trajectory is defined as $\langle s_0, a_o, r_1 ... s_T, a_T, r_{T+1} \rangle$. This is valid as $s_{0:T}$ still follows the current policy distribution π and therefore describes agent's current performance under the policy π. In our case, we simply set $\phi(\cdot)$ as (dx, dy, yaw), where dx, dy are the distances between goal and initial state along the x/y-axis in a 2D plane, and yaw is the initial orientation of the agent. Such setting can uniquely define the initial state given a fixed goal position.

The final modification is that the training will not enter the ACL process until it surpasses a defined threshold on the discounted return in a pre-specified easy task. We include this change according to the findings in [27], as the original NavACL may fail to differentiate *easy* from *frontier* tasks given very few successful episodes reaching the target, i.e., suffering from class-imbalance problems with only negative samples, resulting in an irrational curriculum. With this modification, f_π^r can distinguish *easy* tasks from others so as to generate a proper curriculum.

We finally come up with 2 NavACL$^+$ algorithm variants adopting the aforementioned modifications. They differ in the definition of task type and curriculum scheduling. For the first variant, named as NavACL$^+$ (Slope), we merely change the lower bound of the *frontier* task to be gradually decreasing to cover the most challenging task, where $\beta_f^l = 1 + S \cdot \min(t_c / t_{thre}, 1)$, where t_c is cumulative decision step counts in the ACL phase. The remaining hyperparameters S and t_{thre} are shown in Table IV. The second variant NavACL$^+$ (Hard) replaces the *random* task to be *hard* task with the threshold of $f_\pi^r(h) < \mu_f - \beta_h^h \sigma_f$. The intention is to enable f_π^r to better distinguish task difficulties with an additional *hard* category. The *hard* task also follows a linear increasing probability starting from 0% until maximal 20% proportional to t_c / t_{thre}. The remaining probability are equally assigned to *easy* and *frontier* tasks as in NavACL algorithm to ensure both positive and negative samples are provided to train $f_\pi^r(h)$ for better differentiation on task types. The general procedure of NavACL$^+$ with DRL algorithms in-a-loop is shown in Algorithm 1.

Algorithm 1 NavACL$^+$ combined with DRL algorithm Distributional Soft Actor-Critic

Require: maximum training episodes N_{ep}

Initialize policy π, return-prediction network f_π^r, $i = 0$, Tasks $H = \emptyset$,
while $i < N_{ep}$, i++ **do**
  if $PassFirstTask$ **then** ▷ The fixed easy task is well trained.
    $\mu_f, \sigma_f \leftarrow FitNormal(f_\pi)$
    $h \leftarrow GetDynamicTask(f_\pi^r, \mu_f, \sigma_f)$
  else
    $h \leftarrow GetFirstTask()$
  end if
  //record all augmented tasks, trajectory τ and discounted returns of an episode.
  $H_{aug},\ \tau,\ G_{0...T-1} \leftarrow RunEpisode(h, \pi)$
  Store τ into $\mathcal{B}$
  H.append $(H_{aug}, G_{0...T-1})$
  if $i \% \Delta_{ep} == 0$ **then**
    Update $\pi \leftarrow DSAC()$ ▷ Can be any RL algorithm.
    Update $f_\pi^r \leftarrow NavACL^+Train(H)$
    $H = \emptyset$
  end if
end while

C. 3D-Temporal Convolutional Network

Inspired from the work [28], we also use 3D-convolutions to process the image sequence, where the temporal dimension is the additional dimension. This is advantageous over stacking the frames along channel and performing 2D-convolutions [11] as the temporal correlation after the first 2D convolution is lost [28]. In the temporal dimension, we apply hierarchically-structured dilated convolutions similarly to TCN [26]. The details of the complete network are visualized in Figure 2. Besides, we incorporate $N_h - 1$ historical actions and rewards into the observation space as suggested in [29] to further mitigate the effect of PO.

D. Distributional Soft Actor-Critic with Modifications

We build our learning algorithm on top of one state-of-the-art off-policy continuous control DRL algorithm Distributional Soft Actor-Critic (DSAC) [12]. Distributional RL is reported to greatly enhance the cumulative rewards [30], [31]. DSAC inherits the advantage of maximal-entropy RL that encourages exploration, while outperforming the soft actor-critic algorithm (SAC) [32]. Here, we mentioned our modifications on top of that and the exact algorithmic explanation is in [12]. For the distributional RL part, we also use the state-of-the-art algorithm Fully Parameterized Quantile Function [31] and further combine Non-decreasing Quantile Function Network [33] to guarantee monotonically-increasing quantile estimates. We abandon the use of target

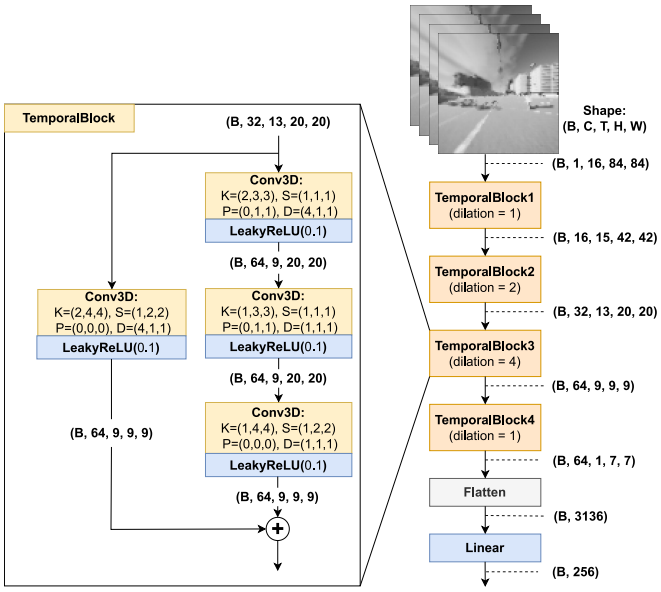

Fig. 2. 3D-Temporal Convolutional structure for encoding vision sequences. Slightly different from the TCN in [26], the temporal length is reduced after each dilated convolution to reduce memory usage. The dilation factor along the temporal dimension is set as 1 for the last temporal convolution to capture the information from all hidden temporal nodes. 'K', 'S', 'P', 'D' respectively denote kernel, stride, padding and dilation. The image sequence shape is (B, C, T, H, W), where B, C, T, H and W denote batch size, channel, temporal length, height and weight of the image respectively. For the Frame-Stacking approach, we use 3 blocks of 2D-convolutions to downscale the image to the shapes (B,16,42,42), (B,32,21,21) and (B,64,7,7) sequentially, similarly to [11].

actor as we find it causes performance drops in some toy tasks and does not conform to SAC either, where no target actor is used in SAC.

For the replay buffer, we adapt the idea from [34] and set up two replay buffers. The work [34] indicates a mixed sampling of expert demonstrations and collected experiences by RL agent greatly facilitates the whole learning procedure. In our case, two replay buffers respectively store non-terminal transition experiences and terminal ones. The terminal samples serve as the sparse positive learning signals which are vital for RL-agent. Moreover, DSAC features one-step temporal-difference learning, $\hat{q}(s_t, a_t)$ is updated based on the value of $\hat{q}(s_{t+1}, a_{t+1})$. Thus, only by learning a correct q-value estimate on terminal state-action pairs $\hat{q}(s_T, a_T)$ can other state-action pairs $\hat{q}(s_{T-1}, a_{T-1})...\hat{q}(s_0, a_0)$ along the trajectory be learned correctly. Hence, by ensuring at least one terminal sample is present in each batch update, it can enhance the learning speed. Rather than take a fixed proportion of samples from both buffers as in [34], the number of positive samples is dynamically scaled. Importance sampling ratio is also applied to terminal and non-terminal samples for updating Q-value, policy and SAC exploration coefficient α as [35] does. The details are shown in Algorithm 2.

IV. Experimental Design

We perform a pedestrian navigation task in the simulator CARLA [36]. We select a road corner in a default map, as shown in Figure 1(a). The pedestrian with a field of view of $135°$, which mimics a human setting, learns to cross the road along a zebra-crossing and reach the goal area. Here we present how to cast the navigation task into an RL problem.

TABLE I

A Summary of Observation Design and their Corresponding Dimensions.

Type of Networks	Observation Components	Dimensions
3D-TCN	Grayscale visual sequence	$\mathbb{R}^{B\times 1\times N_h\times 84\times 84}$
	Historical action sequence	$\mathbb{R}^{B\times (N_h-1)\times 3}$
	Historical reward sequence	$\mathbb{R}^{B\times (N_h-1)\times 1}$
Frame-Stacking	Grayscale visual sequence	$\mathbb{R}^{B\times N_h\times 84\times 84}$

The action $a_t \in \mathbb{R}^3$ consists of a translational velocity $v \in [0m/s, 2m/s]$, a rotational velocity $\omega_{yaw} \in [-80°/s, 80°/s]$, and a head turning motion m_f. The actor outputs the Gaussian mean and the lower triangular matrix of the covariance using Cholesky Decomposition so that the dependence between each action dimension is considered. The immediate reward is $r(t) = r_m(t) + r_v(t) + r_c(t)$, where the speed reward $r_v(t) = 0.125v$ encourages the agent to move fast, the collision reward is $r_c(t) = -3$ in case of a collision. The map reward $r_m(t)$ is shown in Figure 1(b) and takes different values depending on the location of the agent. The goal (area 1) gives a reward of 15. For undesired regions like streets (area 3), we assign a reward of –3. Areas 2, 4 and 5 represent safe regions like sidewalks or zebra-crossings and return the reward of –0.6, –0.8 and –1 respectively. An episode is terminated when the agent reaches the goal or the number of steps is larger than 8000 and each decision step endures $0.25s$.

For the observation space, the design differs in our 3D-TCN approach and Frame-Stacking approach used in [10]. In 3D-TCN, each observation o_t includes a sequence of greyscaled visual observations along with historical rewards and actions, whereas in the Frame-Stacking approach, previous frames are stacked directly along the channel dimension with no auxiliary input of historical rewards and actions. The details are in Table I and Figure 2.

V. Results

Here we would like to answer the following questions:

- Is 3D-TCN advantageous over the Frame-Stacking approach in this first-person visual navigation task featuring PO? (Analyzed in Section V-A)
- Do our proposed NavACL$^+$ variants bring a performance boost over NavACL? Does NavACL$^+$ propose meaningful curricula? (Analyzed in Section V-B)
- How is the performance of NavACL$^+$ variants compared to training from fixed start and NavACL, when generalized to interpolated and extrapolated initial positions? (Analyzed in Section V-C)

To investigate those effects, we run 6 different settings as shown in Table II, with each setting run 5 times and about 3.5M decision steps. A typical run takes 6 to 11 days on a NVIDIA DGX A100 Machine.

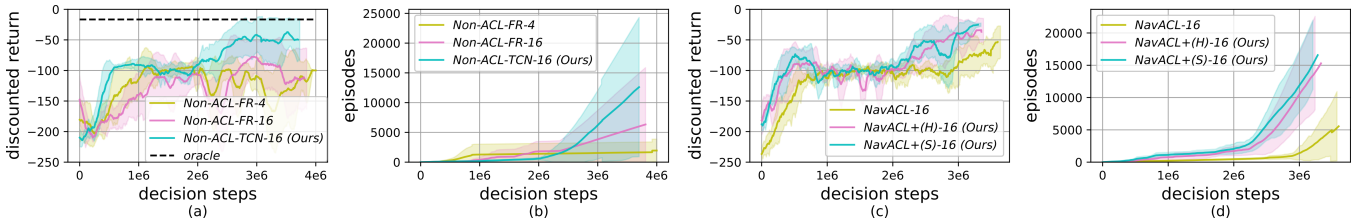

Fig. 3. The illustrations of training performance (smoothed) averaged over 5 runs in mean and standard deviation scheme, with each run having reached convergence. The 'oracle' denotes the optimal discounted return for the fixed initial task, while the worst possible value is −300. For the case of not reaching the target within $T_{max} = 8000$ decision steps, but still walking in low-penalty regions, this corresponds to a discounted return in the interval of $[-80, -60]$. A value difference of 20 in discounted return can mark a huge difference in the quality of policy, where the differences are better reflected in (b),(d). Note that all NavACL variants feature diverse initial poses, corresponding to a non-fixed optimal discounted return. For the easiest task (point 0) and the most difficult task (point 7), the optimal discounted returns given favorable initial orientations are roughly 4 and −23. Both of the NavACL$^+$ variants enter the ACL phase at around 0.9M steps.

A. 3D-TCN vs. Frame-Stacking approach

To verify the effectiveness of our proposed 3D-TCN approach on PO, we compare the performance of Non-ACL-FR-4, Non-ACL-FR-16 and Non-ACL-TCN-16. They feature exactly the same task shown in Figure 1(b) and the same DRL algorithm, only differing in the network design for encoding inputs.

The training performance of the three variants can be seen in Figure 3(a)(b). It can be observed that Non-ACL-TCN-16 shows superior performance in discounted return over two other variants. 3 out of 5 runs converge robustly to the near-optimal policy. As a contrast, none of the Non-ACL-FR-4 trials and merely one Non-ACL-FR-16 trial end up converging robustly to the near-optimal policy. The performance gain of Non-ACL-TCN-16 can also be verified in the validation task in Section V-C. In 2 unsuccessful runs of Non-ACL-TCN-16, the agent converges to a local optimum, i.e., converges to the discounted return of −100, where most of the episodes end up reaching T_{max}. We address this issue using ACL in Section V-B. Figure 3(b) shows the relation episode counts and cumulative number of decision steps. A steeper slope means the agent spends less decision steps to reach the goal and is desired. It can be clearly seen that Non-ACL-TCN-16 has the steepest slope and Non-ACL-FR-16 has the second steepest slope. The sign of improvement is present around 2.5M decision steps. For Non-ACL-FR-4, most episodes end up exceeding the maximal episodic length.

More importantly, Non-ACL-FR-16 turns out to be unstable in learning. 4 out of 5 runs end up in divergence on fitting quantile and policy losses in DSAC, only to learn policies with discounted returns below −100 (Figure 3). For Non-ACL-FR-4, all 5 runs end up in divergence, resulting in returns below −100. In contrast, Non-ACL-TCN-16 does not diverge, even in 2 failure cases. The divergence arises from the incapacity to address PO. In the 2 failure runs of Non-ACL-TCN-16, despite not reaching the target, the pedestrian still makes rational decisions which lead to a high return, i.e, the pedestrian performs circling movements on zebra-crossing or pavements. This still indicates that the agent encodes the visual observation correctly with the 3D-TCN structure to avoid high-penalty areas like roads. In all the divergent cases of Non-ACL-FR-4/16, the agent behaves like random walks. Interestingly, directly adding more historical observations does not prevent divergence in the Frame-Stacking network when one compares Non-ACL-FR-16 to Non-ACL-FR-4. However, in Atari games, the Frame-Stacking network still works, as most of the Atari games have a third-person view which suffers less from PO when compared to our first-person visual navigation task.

Based on these findings, it can be concluded that 3D-TCN with historical rewards and actions can effectively address PO in the first-person visual navigation task and we use 3D-TCN structure for further experiments.

TABLE II

A SUMMARY OF THE SETTINGS OF DIFFERENT DRL VARIANTS.

Settings	Using ACL	Visual Process	N_h
Non-ACL-FR-4	Fixed start	Frame-Stack	4
Non-ACL-FR-16	Fixed start	Frame-Stack	16
Non-ACL-TCN-16	Fixed start	3D-TCN	16
NavACL-16	NavACL	3D-TCN	16
NavACL$^+$(S)-16	NavACL$^+$ (Slope)	3D-TCN	16
NavACL$^+$(H)-16	NavACL$^+$ (Hard)	3D-TCN	16

B. NavACL$^+$ vs. NavACL

The last section shows that Non-ACL-TCN-16 can effectively mitigate PO. However, it still has 2 unsuccessful runs, where most of the episodes terminate with not reaching the goal. The reason is sparse positive experiences, especially in the cases of large initial-goal distances. ACL approaches are hence applied to increase the positive experiences and here we show to which extent our NavACL$^+$ variants outperform the original NavACL. The performance enhancement of NavACL over training from fixed starts will be discussed in Section V-C.

The training performance of NavACL-16, NavACL$^+$(S)-16 and NavACL$^+$(H)-16 is illustrated in Figure 3(c)(d). NavACL$^+$(S)-16 and NavACL$^+$(H)-16 exhibit similar performance and are notably better than NavACL-16, where NavACL$^+$ variants show signs of convergence in around 2.3M decision steps and NavACL-16 starts to converge after 3M steps from Figure 3(d). All 5 runs of NavACL$^+$(H)-16 and NavACL$^+$(S)-16 converge to near-optimal policies for most of the proposed initial poses, whereas 3 runs of

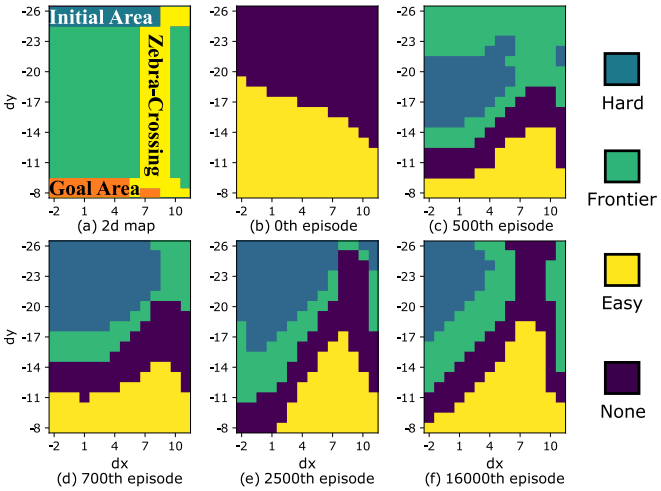

Fig. 4. Evolution of the classified task types of NavACL$^+$(H)-16 in different training stages. (a) is a clipped part from Figure 1(b).

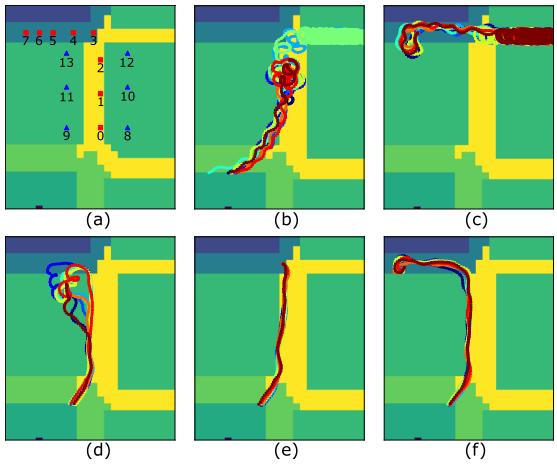

Fig. 5. A qualitative representation of our validation results. For each pose, 10 runs are performed. (a) 14 validation points are chosen. 8 red points denote interpolated tasks while 6 blues points represent extrapolated ones not encountered in the NavACL process. Point 3 is the initial pose for training Non-ACL variants. The task difficulty rises monotonically from point 0 to 7. (b)-(c) Trajectory distribution of NavACL-16 on interpolated tasks, Failure cases can be seen. (d) NavACL$^+$(S)-16 on one extrapolated task. (e)-(f) NavACL$^+$(S)-16 on interpolated tasks.

NavACL-16 show similar patterns. The remaining 2 runs of NavACL-16 show successes in initial poses close to the goal, but fails in distant initial poses, e.g., in cases of the initial pose used in Non-ACL-TCN-16. NavACL-16 shows similar behaviors as Non-ACL-TCN-16, performing circling movements in low-penalty areas but not reaching the goal. This is in accordance with our expectation as NavACL-16 fails to distinguish *easy* from *frontier* tasks, where it could propose difficult tasks as *easy* tasks in the worst case. When frequently trained on difficult tasks and positive experiences are absent, DSAC agent is likely to stagnate at a local optimum.

Furthermore, we investigate whether a reasonable curriculum is formed during training. A qualitative illustration can be seen in Figure 4. For instance, an *easy* task gradually spreads its coverage from the neighborhood of goal area in early stages to more distant areas like zebra-crossing as the policy improves (Figure 4(b)-(f)). The same goes for the evolutions for *frontier* and *hard* tasks. In contrast, the success prediction network f_π^s of NavACL-16 shows no regular or meaningful *easy-frontier* task segmentation over different episodes after our investigation. The reason is manifest that the success prediction network is prone to overfitting few initial pose samples, only to generalize incorrectly to task types of other states.

C. Validation Results

To examine the generalizability of the trained policy with respect to different initial poses, i.e., different task difficulties, we perform a systematic validation as shown in Figure 5(a). Altogether 14 validation points are chosen, where 8 of them are interpolated tasks and the rest are extrapolated ones not encountered in NavACL process. For each initial point (x, y) in the 2D-map, we also set 8 different orientations in yaw-angle $\{0°, \pm45°, \pm90°, \pm135°, 180°\}$ to check the agent's behavior under the effect of PO. For every initial pose (x, y, yaw), we sample 10 trajectories, resulting in a total number of 1120 validation runs for each algorithm variant. The best run of each variant is used for validation and the results are shown both quantitatively in Table III and qualitatively in Figure 5(b)-(f). For validation we clip $T_{max} = 2000$.

In accordance with our expectation, Table III shows that our two NavACL$^+$ variants outperform the original NavACL and Non-ACL variants significantly, especially on some hard tasks, e.g. point $6, 7$. Besides, all ACL approaches demonstrate consistently better performance than Non-ACL trials on all extrapolated tasks. It is explainable, as training from different initial poses benefits generalization more than training from a fixed initial state. Furthermore, NavACL$^+$(S)-16 and NavACL$^+$(H)-16 are comparable considering the performance on all tasks.

Interestingly, we also have discovered the following phenomena: The agent performs less stochastically in the regions frequently reached in training phase, e.g., Figure 5(e)(f). In contrast, stochastic behavior can be seen in extrapolated initial poses that the agent has rarely visited in training, corresponding to Figure 5(d). This is reasonable as DSAC eventually converges to a less stochastic policy on frequently-visited states to achieve a persistent high return. This also has some connection with interpreting the result of PO. Given some initial orientations where the agent can not directly observe the goal, the agent will first rotate until it is oriented towards the optimal path. In most of the interpolated cases, we observe that the agent rotates in consistent directions even if this is sometimes non-optimal, e.g., rotating 270° clockwise instead of 90° counter-clockwise. In extrapolated cases, the agent rotates more stochastically. A higher standard deviation in extrapolated tasks than in interpolated ones in Table III also verifies this point. As a summary, the agent behaves nearly optimally regardless of initial orientations as long as that state is frequently visited in training.

TABLE III

COMPLETE VALIDATION RESULTS AVERAGED OVER 8 ORIENTATIONS FOR EACH VALIDATION POINT.

Initial Pose	Average Discounted Return						
	Non-ACL-FR-4	Non-ACL-FR-16	Non-ACL-TCN-16	NavACL-16	NavACL$^+$(H)-16	NavACL$^+$(S)-16	Random Walk
Point 0	-242.7 $\pm$ 11.1	-9.6 $\pm$ 3.0	-7.2 $\pm$ 3.8	-9.4 $\pm$ 9.0	$\mathbf{-5.4 \pm 6.0}$	-7.3 $\pm$ 7.7	-180.1 $\pm$ 40.1
Point 1	-201.3 $\pm$ 10.2	-17.5 $\pm$ 3.9	-14.0 $\pm$ 3.0	-18.0 $\pm$ 14.9	$\mathbf{-12.1 \pm 5.2}$	-13.0 $\pm$ 7.7	-201.6 $\pm$ 31.6
Point 2	-199.7 $\pm$ 20.3	-21.8 $\pm$ 3.2	-21.6 $\pm$ 3.9	-25.6 $\pm$ 13.6	$\mathbf{-18.1 \pm 6.0}$	-23.6 $\pm$ 13.0	-211.0 $\pm$ 24.1
Point 3	-131.6 $\pm$ 21.6	-30.5 $\pm$ 4.0	-25.7 $\pm$ 3.5	-26.6 $\pm$ 5.7	$\mathbf{-18.6 \pm 2.6}$	-20.4 $\pm$ 4.1	-206.0 $\pm$ 13.3
Point 4	-115.2 $\pm$ 2.7	-45.8 $\pm$ 4.4	-36.9 $\pm$ 2.6	-26.0 $\pm$ 6.8	-24.6 $\pm$ 5.5	$\mathbf{-24.2 \pm 7.2}$	-209.4 $\pm$ 18.2
Point 5	-99.7 $\pm$ 0.0	-75.7 $\pm$ 7.5	-58.5 $\pm$ 2.8	-40.1 $\pm$ 9.0	$\mathbf{-28.8 \pm 5.0}$	-29.0 $\pm$ 6.1	-214.5 $\pm$ 12.4
Point 6	-99.9 $\pm$ 0.0	-91.3 $\pm$ 5.2	-70.7 $\pm$ 5.1	-43.6 $\pm$ 4.5	-30.9 $\pm$ 4.0	$\mathbf{-30.4 \pm 5.0}$	-220.5 $\pm$ 19.9
Point 7	-100.2$\pm$0.4	-101.2 $\pm$ 3.6	-84.4 $\pm$ 9.2	-53.0 $\pm$ 3.8	$\mathbf{-32.6 \pm 2.7}$	-35.3 $\pm$ 6.9	-217.2 $\pm$ 13.4
Avg.Int	-182.6	-54.3	-52.4	-36.0	$\mathbf{-30.6}$	-31.5	-217.9
Point 8	-288.3 $\pm$ 1.2	-56.9 $\pm$ 11.2	-80.3 $\pm$ 47.9	$\mathbf{-34.7 \pm 8.1}$	-41.5 $\pm$ 12.4	-35.0 $\pm$ 9.4	-228.3 $\pm$ 29.8
Point 9	-288.1 $\pm$ 1.8	-65.0 $\pm$ 9.8	-75.1 $\pm$ 14.3	-47.8 $\pm$ 10.1	$\mathbf{-25.0 \pm 10.8}$	-26.4 $\pm$ 10.9	-250.1 $\pm$ 21.4
Point 10	-209.2 $\pm$ 43.4	-72.3 $\pm$ 8.7	-102.0 $\pm$ 13.7	-48.5 $\pm$ 6.2	-57.6 $\pm$ 14.0	$\mathbf{-44.9 \pm 9.7}$	-252.6 $\pm$ 23.0
Point 11	-216.9 $\pm$ 7.5	-46.0 $\pm$ 7.2	-45.2 $\pm$ 15.4	$\mathbf{-28.0 \pm 13.4}$	-40.6 $\pm$ 9.3	-51.7 $\pm$ 20.7	-200.0 $\pm$ 32.6
Point 12	-217.2 $\pm$ 5.6	-78.3 $\pm$ 7.1	-75.8 $\pm$ 10.4	-58.0 $\pm$ 23.4	-65.0 $\pm$ 21.7	$\mathbf{-49.6 \pm 8.8}$	-232.2 $\pm$ 19.3
Point 13	-192.9 $\pm$ 1.2	-83.9 $\pm$ 10.9	-75.1 $\pm$ 11.0	$\mathbf{-52.5 \pm 13.2}$	-63.1 $\pm$ 12.6	-62.7 $\pm$ 20.5	-233.9 $\pm$ 13.9
Avg.Ext	-190.3	-60.2	-58.8	-37.3	-36.5	$\mathbf{-33.6}$	-219.0

We notice that the NavACL$^+$ agent can also fail when the initial poses are too distant from the initial poses experienced during training. Specifically, the agent performs circling motions, some with a gradual trend of approaching the goal, until it reaches the state that has been well trained. After that, the agent follows a near-optimal policy towards the goal. Our finding is analogous to the work [27]. In fact, DRL has a limited degree of generalization and one practical way to achieve generalizability is to include extrapolated domains in the training phase. We verify this by two additional runs of NavACL$^+$(S)-16 with enlarged ACL-proposed regions and some hyperparameter changes. The agent then behaves almost optimally in extrapolated poses, as shown in our video.

The pedestrian also shows the averaged walking velocity of around $1.9\,m/s$, when walking straight, which approaches the maximal speed. This is desired, as our reward setting encourages the agent to reach the goal as soon as possible.

VI. CONCLUSIONS

In this paper, we use Deep Reinforcement Learning with Automatic Curriculum Learning (ACL) to solve an end-to-end first-person visual navigation problem in urban areas. In our tasks, a pedestrian learns to cross the road and to reach a goal area via zebra-crossings. To address the challenges of severe partial observability and sparse positive experiences, we propose a 3D Temporal Convolutional Network (3D-TCN) and an improved ACL algorithm NavACL$^+$. The 3D-TCN, together with auxiliary information of historical rewards and actions, show its effectiveness in addressing partial observability compared to a commonly-used Frame-Stacking approach in RL. The Frame-Stacking approaches often diverge during training. Our ACL algorithm NavACL$^+$ generates meaningful curricula and therefore results in a 40% higher success rate in solving the hard tasks. It also converges 0.7M decision step earlier than the original NavACL, given a total training of 3.3M steps. Moreover, NavACL$^+$ generalizes to different initial states in the curriculum-proposed regions. We also verify that by including the extrapolated tasks in NavACL$^+$ proposal regions, the agent also achieves near-optimal policies in extrapolated tasks. Finally, our experiments validate that our modified Distributional Soft Actor-Critic algorithm can be extended to high-dimensional visual input tasks besides the low-dimensional ones in the original work. In future, we will extend our work with additional traffic flow.

APPENDIX

Here we present the hyperparameters used in the modified DSAC algorithm, NavACL and NavACL$^+$ in Table IV. As a supplement, *PassFirstTask* is defined as the average of 30 recent discounted returns ≥ -25. We also present the details of the double replay buffer in Algorithm 2.

Algorithm 2 Sampling Scheme from Double Replay Buffer

Require: Buffer storing terminating samples $\mathcal{B}_\mathcal{T}$, Buffer storing non-terminating samples $\mathcal{B}_\mathcal{N}$, Batch size N_B

$p_T = |\mathcal{B}_\mathcal{T}| / (|\mathcal{B}_\mathcal{T}| + |\mathcal{B}_\mathcal{N}|)$ ▷ $|\mathcal{B}_\mathcal{T}|$ denotes number of samples stored in $\mathcal{B}_\mathcal{T}$

$N_T = \max(\text{round}(p_T \cdot N_B), 1)$ ▷ At least one sample is taken from $\mathcal{B}_\mathcal{T}$.

Sample N_T experiences $< s_\mathcal{T}, a_\mathcal{T}, r_\mathcal{T}, s'_\mathcal{T}, d_\mathcal{T} >$ uniformly from $\mathcal{B}_\mathcal{T}$

Sample $N_B - N_T$ experiences $< s_\mathcal{N}, a_\mathcal{N}, r_\mathcal{N}, s'_\mathcal{N}, d_\mathcal{N} >$ uniformly from $\mathcal{B}_\mathcal{N}$ ▷ d: episode termination flag

// Compute importance sampling ratio

$P = N_B / (|\mathcal{B}_\mathcal{T}| + |\mathcal{B}_\mathcal{N}|)$

$\tilde{\eta}_\mathcal{T} = P \cdot |\mathcal{B}_\mathcal{T}| / N_T$

$\tilde{\eta}_\mathcal{N} = P \cdot |\mathcal{B}_\mathcal{N}| / (N_B - N_T)$

$\eta_\mathcal{T} = \tilde{\eta}_\mathcal{T} / \max(\tilde{\eta}_\mathcal{T}, \tilde{\eta}_\mathcal{N})$, $\eta_\mathcal{N} = \tilde{\eta}_\mathcal{N} / \max(\tilde{\eta}_\mathcal{T}, \tilde{\eta}_\mathcal{N})$

Return $< s, a, r, s', d, \eta >_{\{\mathcal{T},\mathcal{N}\}}$

REFERENCES

[1] S. D. Morad, R. Mecca, R. P. Poudel, S. Liwicki, and R. Cipolla, "Embodied visual navigation with automatic curriculum learning in real environments," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 683–690, 2021.

TABLE IV

HYPERPARAMETER SETTINGS FOR ALGORITHMS DSAC, NAVACL, NAVACL$^+$.

Parameter Name	Parameter Value
Distributional Soft Actor-Critic	
Discount Factor γ	0.99
Minimum Expected Entropy	1/3
Soft Update Coefficient	2.5e-3
Initial Temperature α_0	0.25
Learning Rate of Temperature lr_α	1e-5
Learning Rate of Quantile Network lr_q	5e-5
Learning Rate of Fraction Proposal Network lr_f	2.5e-9
Learning Rate of Policy Network lr_p	1e-4
Non-Terminating Replay Size $\lvert\mathcal{B}_\mathcal{N}\rvert$	1.5e6
Terminating Replay Size $\lvert\mathcal{B}_\mathcal{T}\rvert$	1.5e6
Batch Size	64
Number of Quantile Estimates	32
Number of Experiences Before Training	5e4
Train Interval	8 steps/training
NavACL/NavACL$^+$	
Learning Rate of ReturnPredictionNet	1e-3
Batch Size for Training ReturnPredictionNet	64
Easy Task Coefficient β_e	1
Frontier Task Coefficient higher bound β_f^h	0.1
Frontier Task Coefficient lower bound β_f^l	1
Slope S in NavACL$^+$	5
Step threshold t_{thre}	1.5e6
Number of tasks to estimate the task distribution	1024
Train DSAC and NavACL$^+$ every Δ_{ep} episodes	5

[2] J. Petzold, M. Wahby, F. Stark, U. Behrje, and H. Hamann, ""if you could see me through my eyes": Predicting pedestrian perception," in *2022 8th International Conference on Control, Automation and Robotics (ICCAR)*. IEEE, 2022, pp. 184–190.

[3] A. Jain, S. Casas, R. Liao, Y. Xiong, S. Feng, S. Segal, and R. Urtasun, "Discrete residual flow for probabilistic pedestrian behavior prediction," in *Conference on Robot Learning*. PMLR, 2020, pp. 407–419.

[4] G. Kahn, A. Villaflor, B. Ding, P. Abbeel, and S. Levine, "Self-supervised deep reinforcement learning with generalized computation graphs for robot navigation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 5129–5136.

[5] J. Choi, C. Dance, J.-e. Kim, S. Hwang, and K.-s. Park, "Risk-conditioned distributional soft actor-critic for risk-sensitive navigation," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 8337–8344.

[6] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[7] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th annual international conference on machine learning*, 2009, pp. 41–48.

[8] S. Narvekar, B. Peng, M. Leonetti, J. Sinapov, M. E. Taylor, and P. Stone, "Curriculum learning for reinforcement learning domains: A framework and survey," *arXiv preprint arXiv:2003.04960*, 2020.

[9] D. Tanneberg, E. Rueckert, and J. Peters, "Evolutionary training and abstraction yields algorithmic generalization of neural computers," *Nature Machine Intelligence*, vol. 2, no. 12, pp. 753–763, 2020.

[10] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[11] 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, pp. 529–533, 2015.

[12] X. Ma, L. Xia, Z. Zhou, J. Yang, and Q. Zhao, "Dsac: Distributional soft actor critic for risk-sensitive reinforcement learning," *arXiv preprint arXiv:2004.14547*, 2020.

[13] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2012, pp. 5026–5033.

[14] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 3357–3364.

[15] J. Kulhánek, E. Derner, T. De Bruin, and R. Babuška, "Vision-based navigation using deep reinforcement learning," in *2019 European Conference on Mobile Robots (ECMR)*. IEEE, 2019, pp. 1–8.

[16] Y. Li, J. Song, and S. Ermon, "Infogail: Interpretable imitation learning from visual demonstrations," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[17] G. Chen, L. Pan, P. Xu, Z. Wang, P. Wu, J. Ji, X. Chen, *et al.*, "Robot navigation with map-based deep reinforcement learning," in *2020 IEEE International Conference on Networking, Sensing and Control (ICNSC)*. IEEE, 2020, pp. 1–6.

[18] C. Florensa, D. Held, M. Wulfmeier, M. Zhang, and P. Abbeel, "Reverse curriculum generation for reinforcement learning," in *Conference on robot learning*. PMLR, 2017, pp. 482–495.

[19] S. Racaniere, A. Lampinen, A. Santoro, D. Reichert, V. Firoiu, and T. Lillicrap, "Automated curriculum generation through setter-solver interactions," in *International conference on learning representations*, 2019.

[20] C. Florensa, D. Held, X. Geng, and P. Abbeel, "Automatic goal generation for reinforcement learning agents," in *International conference on machine learning*. PMLR, 2018, pp. 1515–1528.

[21] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. Pieter Abbeel, and W. Zaremba, "Hindsight experience replay," *Advances in neural information processing systems*, vol. 30, 2017.

[22] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[23] S. Kapturowski, G. Ostrovski, J. Quan, R. Munos, and W. Dabney, "Recurrent experience replay in distributed reinforcement learning," in *International conference on learning representations*, 2018.

[24] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[25] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, "Graph wavenet for deep spatial-temporal graph modeling," *arXiv preprint arXiv:1906.00121*, 2019.

[26] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv preprint arXiv:1803.01271*, 2018.

[27] H. Xue, B. Hein, M. Bakr, G. Schildbach, B. Abel, and E. Rueckert, "Using deep reinforcement learning with automatic curriculum learning for mapless navigation in intralogistics," *Applied Sciences*, vol. 12, no. 6, p. 3153, 2022.

[28] Y. Zhao, B. Deng, C. Shen, Y. Liu, H. Lu, and X.-S. Hua, "Spatio-temporal autoencoder for video anomaly detection," in *Proceedings of the 25th ACM International Conference on Multimedia*, 2017, pp. 1933–1941.

[29] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel, "A simple neural attentive meta-learner," *arXiv preprint arXiv:1707.03141*, 2017.

[30] W. Dabney, G. Ostrovski, D. Silver, and R. Munos, "Implicit quantile networks for distributional reinforcement learning," in *International conference on machine learning*. PMLR, 2018, pp. 1096–1105.

[31] D. Yang, L. Zhao, Z. Lin, T. Qin, J. Bian, and T.-Y. Liu, "Fully parameterized quantile function for distributional reinforcement learning," *Advances in neural information processing systems*, vol. 32, 2019.

[32] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, *et al.*, "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905*, 2018.

[33] F. Zhou, Z. Zhu, Q. Kuang, and L. Zhang, "Non-decreasing quantile function network with efficient exploration for distributional reinforcement learning," *arXiv preprint arXiv:2105.06696*, 2021.

[34] T. Pohlen, B. Piot, T. Hester, M. G. Azar, D. Horgan, D. Budden, G. Barth-Maron, H. Van Hasselt, J. Quan, M. Večerík, *et al.*, "Observe and look further: Achieving consistent performance on atari," *arXiv preprint arXiv:1805.11593*, 2018.

[35] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

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