CRIL: Continual Robot Imitation Learning via Generative and Prediction Model

Chongkai Gao[1], Haichuan Gao[1], Shangqi Guo[1], Tianren Zhang[1], and Feng Chen[123]

***Abstract*— Imitation learning (IL) algorithms have shown promising results for robots to learn skills from expert demonstrations. However, for versatile robots nowadays that need to learn diverse tasks, providing and learning the multi-task demonstrations all at once are both difficult. To solve this problem, in this work we study how to realize continual imitation learning ability that empowers robots to continually learn new tasks one by one, thus reducing the burden of multi-task IL and accelerating the process of new task learning at the same time. We propose a novel trajectory generation model that employs both a generative adversarial network and a dynamics prediction model to generate pseudo trajectories from all learned tasks in the new task learning process to achieve continual imitation learning ability. Our experiments on both simulation and real world manipulation tasks demonstrate the effectiveness of our method.**

I. INTRODUCTION

Intelligent robots nowadays are expected to develop diverse skills to operate well in the real world. Although imitation learning (IL) has shown the power for robots to acquire skills from expert demonstrations [1], [2], it suffers when the robot is required to learn numerous tasks simultaneously, since it is unrealistic to provide robots with all necessary demonstrations in advance [3], and it is also quite a challenge for IL algorithms today to handle with various expert guidance [4]. One way to tackle this problem is to perform *continual learning* in IL algorithms to let the robot continuously update its policy using demonstrations from potentially unlimited new tasks over its lifetime, while avoiding the abrupt degradation of previously learned skills [5]. Continual IL can relieve the burden of collecting demonstrations and enables IL algorithms learn only one task at one time, thus serving as a practical way to develop versatile robots in the real world.

A number of previous works have been studying this problem. Most of them aim to use training data from learned tasks in the training process of a new task to maintain previous knowledge [6]–[9]. However, to acquire real data of learned tasks, they either need a large buffer to store all data of learned tasks, or assume the robot can go back to previous environments to be trained in them again, while keeping environments or raw training data of past tasks unchanged is basically infeasible in real world scenarios [3].

This problem comes from that these approaches all need to obtain *real data* of learned tasks for the training of the new task. However, this is is not essentially required in continual IL. Recently, in machine learning domain, a prevailing idea called deep generative replay (DGR) [10] aims to produce *pseudo data* of learned tasks in the training of the new task. DGR employs a generative model to memorize the data distribution [11] of learned tasks and generates samples from the learned distribution when learning on new tasks and interleaves them with real data of new task to update its models, therefore avoiding the need to obtain real data.

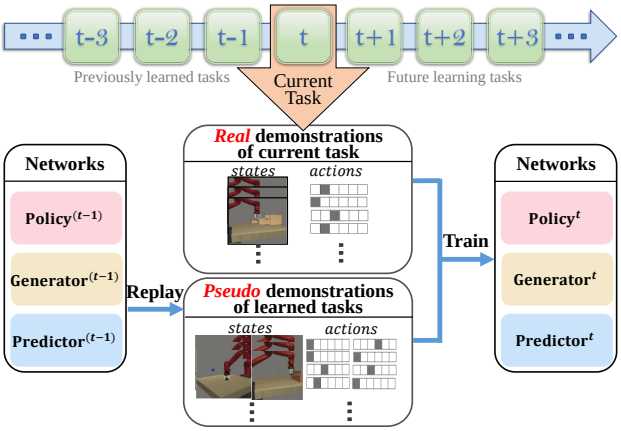


Fig. 1: The paradigm of CRIL. It replays pseudo data of learned tasks and interleaves them with real data of new task to update its networks.

The key challenge to apply DGR to continual robot IL lies in how to generate satisfactory pseudo data of learned tasks. The problem is that robot demonstrations are trajectories, i.e., time-series. The long time-series data property hugely increases the dimension of the target distribution space, and most video generation models today [12], [13] struggle to capture the temporal dependence contained in the trajectory data. These problems make it difficult for a single generative model to capture the data space of the whole trajectory [32]. However, unlike ordinary videos that only consist of state information, robot trajectories here contain rich action information provided by experts. This unique information can be leveraged to help the generative model capture the transition dynamics in the trajectories by predicting the next frame according to the previous frame and action, thus casting the video generation process to generating each video frame progressively.

Following this idea, in this work we present CRIL, a DGR-based method for continual robot IL tasks that designed for trajectory generation. Concretely, we use WGAN-GP [14] to generate only the first frame of each trajectory from learned

[1]Department of Automation, Tsinghua University, Beijing, 100084, China. {gck20,ghc18,gsq15,zhang-tr19}@mails.tsinghua.edu.cn, chenfeng@mail.tsinghua.edu.cn
[2]Beijing Innovation Center for Future Chip, Beijing 100086
[3]LSBDPA Beijing Key Laboratory, Beijing, 100084. China.

tasks, and use an action-conditioned video prediction network to predict subsequent frames of the same trajectory based on generated states and actions from DGR policy. As illustrated in Fig. 2, our method jointly generates states and actions in the replaying stage and explicitly captures the stepwise transition dynamics in the demonstration trajectories. CRIL alleviates the training complexity of the generator by reducing the search space of GAN model from the original large video space to a much smaller single image space, and transfers most of the training complexity to the prediction model. The prediction model is supervised learned by abundant frames in the provided demonstrations, thus can be well trained and the quality of generated trajectories can be guaranteed.

The main contribution of this work is that we are the first to successfully apply the deep generative replay idea to continual robot IL tasks and give out a specialized implementation for robot trajectory generation. In our experiments, we show that our approach can achieve continual learning ability in both simulation and real world tasks. We evaluate the proposed model on diverse manipulation tasks both in meta-world [11] tasks, and in several manipulation tasks using a Jaco2 robot arm in real world. Both experiment results show the superiority of our method.

II. RELATED WORK

A. Continual Imitation Learning for Robotics

Achieving continual learning ability is an open question in robot IL domain [4], [15]. It requires robots to learn from demonstrations in a sequence of tasks while avoiding catastrophic forgetting [16], in which the model's performance on previous tasks abruptly degrades when trained on a new task. Dynamically expanding the policy network [17] and elastic weight consolidation [18] are feasible ways to go, but the first way consumes lots of memory space, and the second way does not perform as good as replay-based methods that replay and retrain robots with previous data [3], [15].

For replay-based methods, recently [7] and [8] borrow the idea of ELLA [19] to store and update a set of reward function basis learned from demonstrations by inverse reinforcement learning with the help of stored data. [6] and [20] use policy distillation to distill knowledge of previous learned policies to a new policy. [9] and [21] use rehearsal to store all the training data of previous tasks in a buffer and then use them in the training of new task. [22] and [23] use state representation learning to build a general perception model for objects and environments, and then use the representation to facilitate the new task learning process, such as curiosity-driven exploration [24]. However, these methods have different assumptions that make them impractical, inefficient and lack of scalability in real world robot tasks. These assumptions include: having a large storage space [8], [9], [21], having access to previous environments [6], [7], [20], or similar skills can be adopted in different tasks [22], [23].

B. Deep Generative Replay

Deep generative replay [10] (DGR) is a continual learning approach that alleviates catastrophic forgetting by leveraging a generative model to memorize the data distribution of learned tasks and generate samples from it in the training process of a new task [11], [15]. In each task training procedure, the generator is first used to generate data of learned tasks, then the policy network would label these replayed data, finally the hybrid data of learned and new tasks are used to update of both policy and generator networks. The generative model can be original GAN network [25], WGAN-GP [10], conditional GAN [5] or variational autoencoder [26]. DGR has been widely used to achieve continual learning ability in supervised learning [27], unsupervised learning [26] and reinforcement learning [22].

C. Video Generation and Prediction

Video generation aims to generate realistic videos based on various inputs, such as random noise [28] or text [29]. It can be used for DGR to replay full trajectories. Recent works usually employ RNN networks as both generator and discriminator of the GAN network to realize time-series generation [13], [30]. On the other hand, video prediction [31], [32] aims to predict subsequent frames in a video given the previous frames. The difference of these two problem is that the input of video generation problems is usually only a vector drawn from some latent space [13], while the input of video prediction are previous frames. The latter one can usually produce images with better quality, and according to the results of [13], directly generating whole trajectories using common video generation approaches can not get enough image quality to support continual robot IL tasks, which is also verified in our experiments.

III. METHOD

In this section, we analyze how to apply DGR to continual robot IL tasks. We first give the formulation of continual robot IL problem, and then analyze it and propose CRIL to tackle this problem.

A. Background: Imitation Learning

We consider the continual learning ability of robot IL algorithms [4]. IL aims to learn a behavior policy π_θ parameterized by θ through mimicking a set of expert demonstrations [1], [4]. Demonstrations are usually provided as a dataset of state-action pairs $\mathcal{D} = \{(s_i, a_i)\}_{i=1\ldots N}$ that belong to different trajectories τ_i, where N is the total number of state-action pairs. These data come from one or different tasks, where each task $\mathcal{T}_i$ is a MDP without the reward function and the discounting factor and can be represented as a tuple: $\langle \mathcal{S}, \mathcal{A}, T, \rho_0 \rangle$ with state-space $\mathcal{S}$, action-space $\mathcal{A}$, transition dynamics $T : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ and the initial state distribution ρ_0. Additionally, we denote the state-action distribution and state distribution of a policy by $\rho^\pi(s, a)$ and $\rho^\pi(s)$. There are mainly two kinds of methods for IL: behavior cloning (BC) and inverse reinforcement learning (IRL). Note in this work we focus on the continual learning ability of IL, and our method is essentially not limited to any specific IL algorithms. We choose BC as our IL algorithm, since performing IRL is quite time-consuming on real world robots.

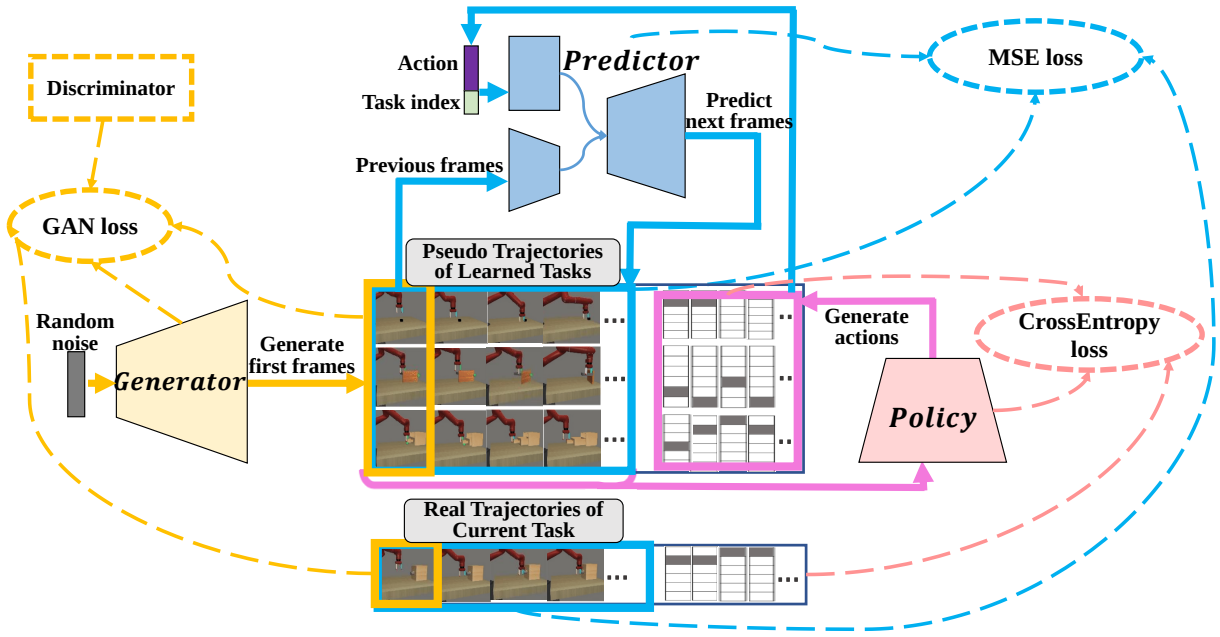


Fig. 2: Network Architecture. Solid lines represent the process of replaying pseudo data and dashed lines represent the loss for training. There are three main modules in CRIL: The generator generates first frames of different trajectories, and the policy and the predictor will iteratively generate actions and subsequent frames. These modules will be trained according to their own loss functions respectively, which are introduced in Section 3 and Section 4.

B. Continual Robot Imitation Learning Problems

In contrast to learning all demonstrations at one time, continual IL focuses on how to learn from demonstrations that come from online sequential tasks. Formally, based on the loss function of BC, we give out the following definition of continual IL:

Definition 1 *Continual imitation learning (CIL) aims to update policy π_θ with a sequential tasks $\mathcal{T}^{(1)}, \mathcal{T}^{(2)} \ldots, \mathcal{T}^{(N_{max})}$ where N_{max} is the total number of tasks. In the training of task $\mathcal{T}^{(t)}$, it minimizes the following objective provided with state-action pairs that come from expert distribution $\rho^{(t)}_{exp}$:*

$$\mathcal{L}^{(t)}_{CIL}(\theta) = -\sum_{i=1}^{t} \lambda^{(i)} \mathbb{E}_{s^{(t)},a^{(t)}\sim\rho^{(t)}_{exp}}[\log \pi_\theta(a^{(t)} \mid s^{(t)})], \tag{1}$$

where $\lambda^{(i)} \in (0, 1)$ is the task weight coefficient that determines the importance of task $\mathcal{T}^{(i)}$ among all tasks.

In this work we choose $\lambda^{(i)} = 1/t, i = 1, 2, \ldots, t$ to set each task equally important.

This definition reveals the biggest difficult of optimizing the continual IL problem: in the training process of task $\mathcal{T}^{(t)}$, we only have access to demonstrations $\mathcal{D}^{(t)}$ from $\rho^{(t)}_{exp}$, while we need to optimize an objective that is evaluated by data from all tasks $\mathcal{D}^{(1,2,\ldots,t)}$ that sampled from $\rho^{(1,2,\ldots,t)}_{exp}$. Thus the key problem is how to acquire $\mathcal{D}^{(1,2,\ldots,t)}$ during the training of $\mathcal{T}^{(t)}$.

C. DGR for Continual Robot Imitation Learning

In order to get over the above difficulty, DGR generates pseudo data of learned tasks by a generative model to fill these data vacancies. Thus there are two networks in DGR for continual IL process: the generator $G^{(t)}_\psi$ parameterized by ψ and the policy network $\pi^{(t)}_\theta$. The full training procedure is illustrated in Fig. 1 and Fig. 2. At the start of training a new task, the generator would generate *states* of learned tasks from its own distribution. The current policy would then label actions for these pseudo states, and then the pseudo states and actions would be mixed together with real data of the new task to train the policy through a certain IL algorithm. Finally the generator will be trained to fit the mixed state distribution of generated pseudo states and the real states from new task demonstrations. This mixed distribution can approximately represent real state distribution of all learned tasks so far.

Formally, this procedure involves two independent processes. For training $G^{(t)}$, a GAN loss is employed to capture the mixed distribution of pseudo states and real states:

$$\begin{aligned}\mathcal{L}^{(t)}_G(\psi) = \mathbb{E}_{s\sim p_{\text{mixed}}(s)}[\log D^{(t)}(s)] \\ + \mathbb{E}_z[\log(1 - D^{(t)}(G^{(t)}_\psi(z)))],\end{aligned} \tag{2}$$

where $p_{mixed}(s)$ is the mixed distribution of real and pseudo states, and $z \sim \mathcal{N}(0, 1)$ is randomly sampled from a unit Gaussian distribution. For training $\pi^{(t)}_\theta$, a supervised learning loss is used to train $\pi^{(t)}_\theta$ to capture the action distribution conditioned on states:

$$\mathcal{L}^{(t)}_\pi(\theta) = \mathbb{E}_{(s,a)\sim p_{\text{mixed}(s,a)}}[\log \pi_\theta(a^{(t)} \mid s^{(t)})]. \tag{3}$$

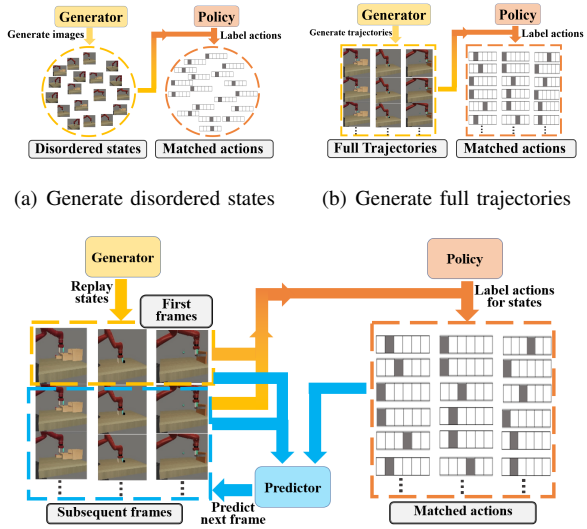
where $p_{mixed}(s, a)$ is the mixed distribution of real and

Algorithm 1 Continual Robot Imitation Learning

Initialize: Task index $t = 0$, policy network $\pi_\theta^{(0)}$, image generation network $G_\psi^{(0)}$, prediction network $P_\phi^{(0)}$, learning rates λ_θ, λ_ψ and λ_ϕ.
While haveNewTask() **do**
  $t = t + 1$
  Get demonstration data $\mathcal{D}^{(t)}$ that consist of m trajectories from new task $\mathcal{T}^{(t)}$
  Initialize a trajectory buffer $B^{(t)}$ and put $\mathcal{D}^{(t)}$ into $B^{(t)}$
  for $i = 1, t$ **do**
    Use $G_\psi^{(t-1)}$ to generate m first images for m different trajectories $\{\tau^{(i)}\}$ of task $\mathcal{T}^{(i)}$;
    Use $P_\phi^{(t-1)}$ and $\pi_\theta^{(t-1)}$ in each trajectory $\tau^{(i)}$ to generate the full trajectory based on first frames;
    Collect generated $\{\tau^{(i)}\}$ and put them into $B^{(t)}$;
  end for
  Update networks using $B^{(t)}$:
    Update $\theta^{(t)} \leftarrow \theta^{(t-1)} - \lambda_\theta \nabla_\theta \mathcal{L}_\pi^{(t-1)}(\theta)$
    Update $\psi^{(t)} \leftarrow \psi^{(t-1)} - \lambda_\psi \nabla_\psi \mathcal{L}_G^{(t-1)}(\psi)$
    Update $\phi^{(t)} \leftarrow \phi^{(t-1)} - \lambda_\phi \nabla_\phi \mathcal{L}_P^{(t-1)}(\phi)$
  Delete buffer $B^{(t)}$
end while

pseudo state-action pairs. A key problem here is that what kind of generation model we should choose for $G_\psi^{(t)}$. For robot demonstration data, the generator needs to deal with time-series trajectories. There exist typically two ways to generate trajectory samples, we call them Original DGR and Trajectory DGR, as shown in Fig. 3(a) and 3(b). For Original DGR, $G_\psi^{(t)}$ is an image generator that generates independent states randomly, without considering the integrity of generated trajectories. This method treat all states of learned tasks as a disordered set, and could directly borrow any commonly used GAN models that perform image-level generation. For Trajectory DGR, it generates a full trajectory at one time, using a time-series GAN model like in [13].

However, these two methods cannot perform well in our problem settings. The reason is that they all employ a single GAN network to capture the whole state space, which is much larger than common continual learning tasks, and therefore much harder for GAN networks to capture. Essentially, the problem comes from that they split the generation of states and actions, thus put the entire complexity of video generation to the generator. This comes from the traditional application scenarios of DGR, which are usually supervised learning tasks like image classification. In those tasks, there is no correlation between different states (pictures to be classified), and actions (image labels) have no effect on adjacent states. However, in most robot imitation learning tasks, two adjacent frames do have strong dynamics correlation conditioned on the action information, which can be explicitly leveraged to substantially reduce the generation difficulty.



(a) Generate disordered states (b) Generate full trajectories

(c) Generate first frames and predict subsequent frames

Fig. 3: Different generating strategies.

D. CRIL: Continual Robot Imitation Learning

In order to integrate action generation into the state generation process, we need to consider the state-action joint distribution, rather than only the state distribution for the generator. We first note that the probabilistic distribution of a complete trajectory τ consisting both states and actions can be decomposed as follows:

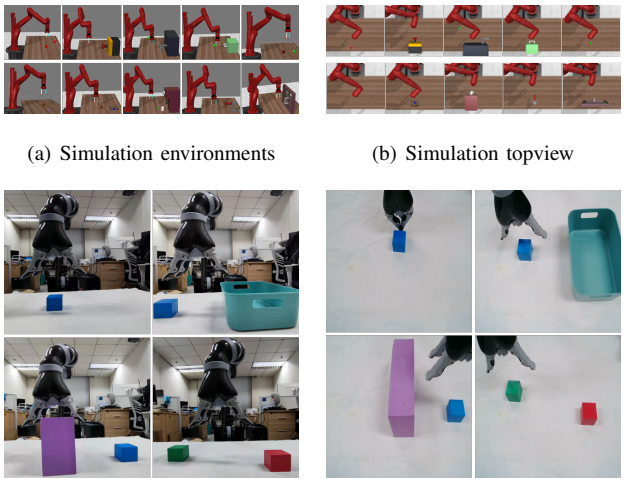
$$\rho_{\pi_\theta}(\tau) = \underbrace{\rho_0(s)}_{\text{first frame generator}} \prod_{i=1}^{T} \underbrace{\pi_\theta(a_i|s_i)}_{\text{policy}} \underbrace{p(s_{i+1}|s_i, a_i)}_{\text{next frame predictor}}, \quad (4)$$

where T is the number of time steps in this trajectory. This decomposition indicates us that we need three models for generating a complete trajectory: a first frame generator G_ψ, a policy network π_θ, and a next frame predictor P_ϕ parameterized by ϕ. As shown in Fig. 3(c), for generating a new trajectory, G_ψ first generates a start frame, and then the policy and predictor iteratively spread out the subsequent frames. By adding a predictor into DGR process, we successfully capture the transition dynamics information to facilitate the trajectory generation. Intuitively, CRIL can be viewed as first initializing an environment, and then reproducing the expert behaviors in this environment. The whole process of CRIL is shown in Algorithm 1. The loss function of $P_\phi^{(t)}$ is a mean square error loss:

$$\mathcal{L}_P^{(t)}(\phi) = \sum_{i=1}^{N} \sum_{j=1}^{T} (\hat{s_{ij}} - P_\phi^{(t)}(s_{ij}, a_{ij}))^2, \quad (5)$$

where N is the number of trajectories and T is the number of steps in one trajectory, and $\hat{s_{ij}}$ are the real states.

Although the high-level idea of CRIL is straightforward, its optimality needs to be proved to ensure that it can achieve the same result of standard DGR method theoretically, e.g., Trajectory DGR. We analyze the optimality in the following section.



(a) Simulation environments (b) Simulation topview

(c) Real world tasks (d) Real world topview

Fig. 4: Tasks in our experiment. Simulation tasks are: reach, button-press, door-open, drawer-open, push, sweep, sweep-into, coffee-button, faucet-open and window-open. Real world tasks are: push, place, pick-over and stack. The cameras in all environments are set at the topview position.

IV. ANALYSIS OF OPTIMALITY

In this section, we theoretically analyze the optimality of CRIL and point out it preserves same optimal solution as Trajectory DGR. We firstly analyze the optimizing process of Trajectory DGR and then prove the optimality of CRIL.

A. The Optimizing Process of Trajectory DGR

We denote a *state trajectory* in task $\mathcal{T}^{(t)}$ that only contains *states* as $\tau_s^{(t)}$, the distribution of generated pseudo state trajectories as $p_g({\tau_s}^{(1,2,\dots,t)})$, and the distribution of real state trajectories from all learned tasks as $p_r({\tau_s}^{(1,2,\dots,t)})$. The objective of Trajectory DGR is to minimize the distance of $p_g({\tau_s}^{(1,2,\dots,t)})$ and $p_r({\tau_s}^{(1,2,\dots,t)})$. Since we use WGAN-GP in this paper, we employ *Earth-Mover* distance $W(p_g, p_r)$ as the distance metric. We give the following lemma that reveals the mathematical form of the optimizing process in Trajectory DGR:

Lemma 1 *In Trajectory DGR, during the training of task $\mathcal{T}^{(t)}$, the distance between $p_g({\tau_s}^{(1,2,\dots,t)})$ and $p_r({\tau_s}^{(1,2,\dots,t)})$ is:*

$$W(p_g, p_r) = \mathbb{E}_{\tau_s \sim p_m^{(t)}}[D^{(t)}(\tau_s)] - \mathbb{E}_z[D^{(t)}(G_\psi^{(t)}(z))], \quad (6)$$

where

$$p_m^{(t)}(\tau_s) = \frac{1}{t}[p_r({\tau_s}^{(t)}) + (t-1)p_g({\tau_s}^{(1,2,\dots,t-1)})], \quad (7)$$

and the gradient direction for optimizing the trajectory generator $G_\psi^{(t)}$ is:

$$\nabla_\psi W(p_g, p_r) = -\mathbb{E}_z[\nabla_\psi D^{(t)}(G_\psi^{(t)}(z))], \quad (8)$$

where $z \sim \mathcal{N}(0,1)$ is a Gaussian random variable and $D^{(t)}$ is the trajectory discriminator in the training of task $\mathcal{T}^{(t)}$.

Proof: In the training of task $\mathcal{T}^{(t)}$, the target distribution that $G_\psi^{(t)}$ aims to fit is the mixed distribution $p_m^{(t)}(\tau_s)$ of real data from task $\mathcal{T}^{(t)}$ and the replayed pseudo data from $p_g({\tau_s}^{(1,2,\dots,t-1)})$. If we assume every task is equally important, we could get equation (7), and the desired $W(p_g, p_r)$ becomes $W(p_g, p_m)$. According to the Kantorovich-Rubenstein duality [33], the Earth-Mover distance $W(p_g, p_r)$ can covert to the following form:

$$\begin{aligned} W(p_g, p_m) &= \inf_{\gamma \sim \Pi(p_g, p_m)} \mathbb{E}_{(x,y)\sim\gamma}[\|x-y\|] \\ &= \frac{1}{K} \sup_{\|f\|_L \le K} \mathbb{E}_{\tau_s \sim p_m}[f(\tau_s)] - \mathbb{E}_{\tau_s \sim p_g}[f(\tau_s)], \end{aligned} \quad (9)$$

where $\Pi(p_g, p_m)$ denotes all possible joint contribution of (p_g, p_m), and f lies in $\mathcal{F} = \{f : \tau_s \to \mathbb{R}, f \in C_b(\tau_s), \|f\|_L \le K\}$. Since τ_s is compact, we can construct a neural network $D^{(t)}$ whose parameters are restricted to a certain range to make it lies in $\mathcal{F}$ as in [14], then we can use $D^{(t)}$ as f in (9) get (6). For (8), as $\mathbb{E}_z \|\nabla_\psi D^{(t)}(G_\psi^{(t)}(z)\| < +\infty$ and $\nabla_\psi D^{(t)}(G_\psi^{(t)}(z)$ is well defined almost everywhere [11], by the dominated convergence, we could exchange the integral and derivation symbol to get (8):

$$\begin{aligned} \nabla_\psi W(p_g, p_r) &= -\nabla_\psi \mathbb{E}_z[D^{(t)}(G_\psi^{(t)}(z))] \\ &= -\mathbb{E}_z[\nabla_\psi D^{(t)}(G_\psi^{(t)}(z))]. \end{aligned} \quad (10)$$

B. Optimality of CRIL

In order to prove the optimality of CRIL, we need to prove that the loss functions (2) and (5) can form another eligible distance measurement that lies in $\mathcal{F}$, and the gradient of CRIL must be equal to (8). We give out the following theorem:

Theorem 1 *The training procedure of CRIL is equal to minimize an Earth-Mover distance between $p_g({\tau_s}^{(1,2,\dots,t)})$ and $p_r({\tau_s}^{(1,2,\dots,t)})$:*

$$W(p_g, p_m) = \mathbb{E}_{\tau_s \sim p_m}[\tilde{f}(\tau_s)] - \mathbb{E}_{\tau_s \sim p_g}[\tilde{f}(\tau_s)], \quad (11)$$

where

$$\tilde{f}(\tau_s) = D(s_0) + \frac{1}{T-1}\sum_{i=1}^{T}(s_i - \hat{s}_i)^2, \quad (12)$$

where $s_{0:T}$ are the states in τ_s, and $\hat{s}_i$ is the states from $p_m^{(t)}$, and $\nabla W(p_g, p_m)$ is equal to (8).

Proof: Since $D^{(t)}$ here is also a generator that trained with WGAN-GP method, $D(s_0)$ can be Lipschitz continuous just like it in Trajectory DGR . On the other hand, for $(s_i - \hat{s}_i)^2$, through image normalization, s_i and $\hat{s}_i$ can both lie in $[-1, 1]$, thus (12) is Lipschitz continuous. Meanwhile, we can train the image discriminator D to increase $D(s_0)$ and train P_ϕ to make $\frac{1}{T-1}\sum_{i=1}^{T}(s_i - \hat{s}_i)^2$ to zero to get the supremacy of all eligible functions in $\mathcal{F}$. This is exactly the max-min procedure of optimizing standard GAN networks. Thus CRIL becomes the same problem of Trajectory DGR.

The only thing left here is to prove optimizing (11) can lead to the same optimal solution of Trajectory DGR to prove CRIL

TABLE I: Quantitative results of different methods, including the mean and standard variance from 5 random seeds.

Method	Simulation Experiments			Real World Experiments		
	Ω_{base}	Ω_{new}	Ω_{all}	Ω_{base}	Ω_{new}	Ω_{all}
Finetune	0.304±0.014	0.997±0.001	0.432±0.017	0.192±0.046	1.000±0.000	0.440±0.009
Original DGR	0.816±0.024	0.899±0.002	0.762±0.038	0.759±0.039	0.885±0.004	0.797±0.015
Trajectory DGR	0.865±0.036	0.991±0.000	0.933±0.009	0.920±0.020	0.997±0.002	0.960±0.017
CRIL	0.990±0.004	0.994±0.002	0.980±0.001	0.998±0.003	0.999±0.001	0.998±0.002
Rehearsal	0.995±0.002	0.997±0.001	0.996±0.002	0.998±0.001	0.999±0.000	0.998±0.001

can preserve the optimality. This could be proved through by explaining the original gradient direction (10) is equal to the gradient of (12). We rewrite the trajectory τ_s as a vector $[s_0, s_1, \ldots, s_T] = [G_\psi(z), P_\phi(s_0, a_0), \ldots, P_\phi(s_{T-1}, a_{T-1})]$ and get the following derivation of (10):

$$
\begin{aligned}
&\mathbb{E}_z[\nabla D^{(t)}(G^{(t)}(z))] \\
=&\mathbb{E}_{\tau_s}[\nabla D^{(t)}(G_\psi^{(t)}(z), \nabla P_\phi^{(t)}(s_0, a_0), \ldots, P_\phi^{(t)}(s_{T-1}, a_{T-1}))] \\
=&\mathbb{E}_z[\nabla_\psi D^{(t)}(G_\psi^{(t)}(z)), 0, \ldots, 0] \\
&+ \mathbb{E}_{s_{1:T-1}}[\nabla_\phi D^{(t)}(0, P_\phi^{(t)}(s_0, a_0), \ldots, P_\phi^{(t)}(s_{T-1}, a_{T-1}))] \\
=&\mathbb{E}_z[\nabla_\psi D^{(t)}(G_\psi^{(t)}(z))] \\
&+ \mathbb{E}_{s_{1:T-1}}[\nabla_\phi \sum_{t=1}^{T-1}(\hat{s_{t+1}} - P_\phi^{(t)}(s_t, a_t))],
\end{aligned}
\tag{13}
$$

which equals to the gradient of (11). Thus CRIL can preserve the optimality of solution.

To summarize, CRIL is also minimizing an Earth-Mover distance between the same two distributions in Trajectory DGR, thus preserves the optimality. As we all know, training a time-series GAN to capture the whole video data space is quite difficult. Under the premise of preserving the optimality, CRIL reduces most of the generation complexity of GAN module to make it only generate the first frames. Thus our method becomes much easier and faster to train.

V. EXPERIMENT

Our experiments aim to answer the following questions: (1) can CRIL be successfully applied to robot continual IL problems in both simulation environments and real world environments? (2) how is the performance of our method compared to Original DGR and Trajectory DGR? and (3) how dose our proposed predictor affect the generating result? We answer (1) and (2) in the quantitative section and answer (3) using qualitative results. Codes are available at https://github.com/HeegerGao/CRIL.

A. Experiment Setup

We set up two separated groups of tasks:

Simulation environments: We choose ten manipulation tasks from meta-world benchmark [34] as shown in Fig. 4(a) and 4(b). The initial configuration of each task is randomized within certain limits. For each simulation task, we collect 100 trajectories as expert demonstrations. Each trajectory contains 23 frames to 124 frames, where each frame is a 3x64x64 RGB image from a top view camera. The action space of robot is a discrete space that consists of seven different actions, including six actions for moving along six axes directions and a stop action that indicates the end of the task.

Real world environments: We use a 6-DoF Kinova Jaco2 robot arm and design four manipulation tasks for experiments as shown in Fig. 4(c) and 4(d). We set a RGB camera at the top view position of each task to collect image observations which are all in a 3x64x64 size. We use the keyboard to control the Cartesian position of the end effector of the robot arm to give out demonstrations. For each task, we collect 15 trajectories as expert demonstrations, which takes us about 1 minute for collecting each trajectory. The action space here is the same as in simulation environments.

B. Methods and Metrics

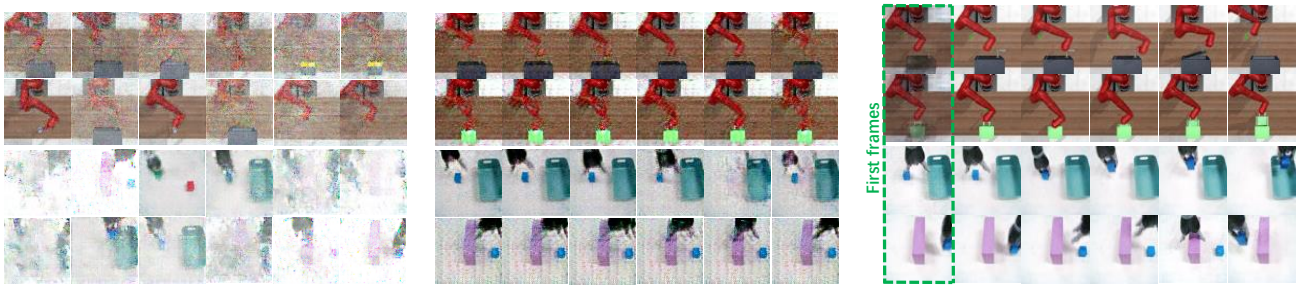
We compare five different methods in above continual IL setting. Note for Original DGR and Trajectory DGR, there are no existing implementations, thus we employ two prevailing generation networks and implement them by ourselves:

- **Finetune:** No continual learning algorithm performed, which means the policy would be updated using only new demonstrations.
- **Rehearsal:** The real training data of learned tasks is directly replayed for policy training via a large replay buffer. This result is the upper bound of the performance of all other methods.
- **Original DGR:** The method in Fig. 3(a). We use WGAN-GP [14] as our generator to generate separated images.
- **Trajectory DGR:** The method in Fig. 3(b). We borrow MoCoGAN [13] as our generator model. MoCoGAN decomposes motion and content in videos and achieves SOTA performance on video generation domain.
- **CRIL:** The proposed model in this work.

We adopt the commonly used metrics in continual learning proposed in [25], which includes three criteria:

$$
\Omega_{base} = \frac{1}{N-1}\sum_{i=2}^{N}\frac{\alpha_{base,i}}{\alpha_{ideal}}, \tag{14}
$$

$$
\Omega_{new} = \frac{1}{N-1}\sum_{i=2}^{N}\alpha_{new,i}, \tag{15}
$$



(a) Images generated by Original DGR (b) Images generated by Trajectory DGR (c) Images generated by CRIL

Fig. 5: Images generated by CRIL, Original DGR and Trajectory DGR after all tasks are leaned. For CRIL and Trajectory DGR, we select four whole trajectories to show. For Original DGR, since the integrity of trajectory can not be guaranteed, we can only select separated images that come from different trajectories randomly.

$$\Omega_{all} = \frac{1}{N-1}\sum_{i=2}^{N}\frac{\alpha_{all,i}}{\alpha_{ideal}}, \tag{16}$$

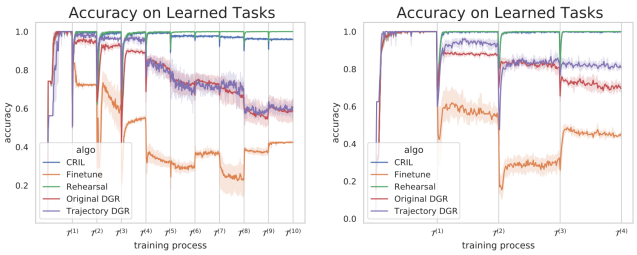
where N is the total number of tasks, and $\alpha_{new,i}$, $\alpha_{base,i}$ and $\alpha_{all,i}$ denote the test accuracy of the new task, the first task and all learned tasks respectively, and α_{ideal} is used to normalize Ω_{base} and Ω_{all}. $\Omega_{base}, \Omega_{new}$ and Ω_{all} measure the model's ability of remembering previous knowledge, transferring knowledge to new tasks and the performance on overall tasks. For all three metrics, higher value indicates better performance.

C. Quantitative Comparison

Table I and Fig. 6 show the quantitative results and the forgetting curves of different methods. Since Finetune does not carry out any kind of continual learning approach, its performance abruptly degrades across the continual learning process. Original DGR and Trajectory DGR can maintain the accuracy on the first few tasks, but their performances also drop when the number of task increases. Our proposed method achieves promising performances that are close to the upper bound performance of rehearsal in both simulation and real world environments.

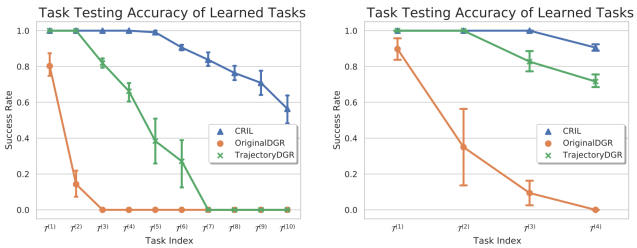
As robot IL is not an image classification problem but a sequential decision-making problem, in which the robot must be placed in the environments to accomplish tasks by reproducing expert demonstrations, the task success rate should be considered as another important performance metric. Fig. 7 shows the success rates of three methods when they are employed to operate in tasks, where CRIL outperforms conventional DGR methods by a large margin. In this situation, the statistical accuracy of state-action mappings becomes less important, since even if the robot make mistakes in only one step, it will probably fail to complete the whole task. Note these results are also correlated to the IL algorithm that employed for continual learning. For instance, using IRL or GAIL may lead to better results.

It is worth mentioning that the training time of Original DGR and Trajectory DGR are both much longer than CRIL, since the generators in the first two methods are hard to train. For Original DGR, it becomes more serious since the generated images are separated and cannot construct complete



(a) Simulation Results (b) Real world Results

Fig. 6: Accuracy of all tasks learned so far in the training process of different methods in both simulation and real world environments. Each method is tested using 5 random seeds.



(a) Simulation Results (b) Real world Results

Fig. 7: Testing success rates of all learned tasks of different methods. Each point is the mean and standard deviation based on 5 test trials.

trajectories. Thus in order to cover states in a trajectory as much as possible, in our experiments, we generate 10 times more samples in Original DGR method that the others, and this leads to a slower training process.

D. Qualitative Analysis of Generated Images

Fig. 5 shows the generated trajectories of three different methods. CRIL produces out the highest quality images. For Original DGR, the generated images are independent of each other, thus cannot form a complete trajectory. It is interesting that, in the generated images of CRIL, first frames of each trajectory are also in poor qualities (see the results in simulation environments in Fig. 5). However, subsequent frames in these trajectories can go back to high qualities, which reveals the power of our prediction model. Since the

prediction model is learned by supervised learning, its output can easily and always be a high quality image compared to the generative model. This makes the whole model have strong robustness, thus most part of generated trajectories are high quality images. This situation also shows that, even if we have limited the generator to producing only the first frames, generating high quality images is still very hard for a GAN network as the number of tasks increases.

VI. CONCLUSION AND FUTURE WORK

In this work, we present a novel method to apply DGR in robot IL tasks to achieve continual learning ability. Our novel trajectory generation model decomposes the full trajectory generation problem to a first-frame generation problem and an action-conditioned next-frame prediction problem, which enables our model to generate high-quality trajectories to support continual robot learning. For future work, researchers might consider how to apply DGR with multi-modal demonstrations, which may contain conflicting actions on the same states, which brings challenges for the predictor. Another attractive direction would be studying on how to stably generate pseudo data in the DGR process, such as learning an embedding space rather than directly generate high-dimensional raw images.

ACKNOWLEDGMENT

We would like to thank Xin Su, Zhile Yang and Yizhou Jiang for various discussions on DGR theory and experiments of GANs. This work was supported in part by the National Natural Science Foundation of China under Grant 61671266 and Grant 61836004, in part by the Tsinghua-Guoqiang research program under Grant 2019GQG0006, and in part by Qualcomm Technologies, Inc.

REFERENCES

[1] Y. Zhu, Z. Wang, J. Merel, A. A. Rusu, T. Erez, S. Cabi, S. Tunyasuvunakool, J. Kramár, R. Hadsell, N. de Freitas, and N. Heess, "Reinforcement and imitation learning for diverse visuomotor skills," in *Robotics: Science and Systems*, 2018.

[2] T. Yu, P. Abbeel, S. Levine, and C. Finn, "One-shot hierarchical imitation learning of compound visuomotor tasks," *arXiv preprint arXiv:1810.11043*, 2018.

[3] G. I. Parisi and C. Kanan, "Rethinking continual learning for autonomous agents and robots," *arXiv preprint arXiv:1907.01929*, 2019.

[4] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, and J. Peters, "An algorithmic perspective on imitation learning," *arXiv preprint arXiv:1811.06711*, 2018.

[5] T. Lesort, A. Gepperth, A. Stoian, and D. Filliat, "Marginal replay vs conditional replay for continual learning," in *Artificial Neural Networks and Machine Learning*, 2019.

[6] A. A. Rusu, S. G. Colmenarejo, C. Gulcehre, G. Desjardins, J. Kirkpatrick, R. Pascanu, V. Mnih, K. Kavukcuoglu, and R. Hadsell, "Policy distillation," *arXiv preprint arXiv:1511.06295*, 2015.

[7] S. Piao, Y. Huang, and H. Liu, "Online multi-modal imitation learning via lifelong intention encoding," in *International Conference on Advanced Robotics and Mechatronics*, 2019.

[8] J. A. Mendez, S. Shivkumar, and E. Eaton, "Lifelong inverse reinforcement learning," in *Annual Conference on Neural Information Processing Systems*, 2018.

[9] T. L. Hayes, N. D. Cahill, and C. Kanan, "Memory efficient experience replay for streaming learning," in *International Conference on Robotics and Automation*, 2019.

[10] H. Shin, J. K. Lee, J. Kim, and J. Kim, "Continual learning with deep generative replay," in *Annual Conference on Neural Information Processing Systems*, 2017.

[11] X. Su, S. Guo, T. Tan, and F. Chen, "Generative memory for lifelong learning," *IEEE transactions on neural networks and learning systems*, 2019.

[12] J. Yoon, D. Jarrett, and M. van der Schaar, "Time-series generative adversarial networks," in *Annual Conference on Neural Information Processing Systems*, 2019.

[13] S. Tulyakov, M. Liu, X. Yang, and J. Kautz, "Mocogan: Decomposing motion and content for video generation," in *Conference on Computer Vision and Pattern Recognition*, 2018.

[14] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved training of wasserstein gans," *arXiv preprint arXiv:1704.00028*, 2017.

[15] T. Lesort, V. Lomonaco, A. Stoian, D. Maltoni, D. Filliat, and N. D. Rodríguez, "Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges," *Information fusion*, 2020.

[16] R. M. French, "Catastrophic interference in connectionist networks: Can it be predicted, can it be prevented?" in *Advances in Neural Information Processing Systems*, 1993.

[17] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks," *arXiv preprint arXiv:1606.04671*, 2016.

[18] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the national academy of sciences*, 2017.

[19] P. Ruvolo and E. Eaton, "Ella: An efficient lifelong learning algorithm," in *International Conference on Machine Learning*, 2013.

[20] R. Traoré, H. Caselles-Dupré, T. Lesort, T. Sun, N. Díaz-Rodríguez, and D. Filliat, "Continual reinforcement learning deployed in real-life using policy distillation and sim2real transfer," *arXiv preprint arXiv:1906.04452*, 2019.

[21] D. J. Mankowitz, A. Žídek, A. Barreto, D. Horgan, M. Hessel, J. Quan, J. Oh, H. van Hasselt, D. Silver, and T. Schaul, "Unicorn: Continual learning with a universal, off-policy agent," *arXiv preprint arXiv:1802.08294*, 2018.

[22] H. Caselles-Dupré, M. Garcia-Ortiz, and D. Filliat, "Continual state representation learning for reinforcement learning using generative replay," *arXiv preprint arXiv:1810.03880*, 2018.

[23] R. Camoriano, G. Pasquale, C. Ciliberto, L. Natale, L. Rosasco, and G. Metta, "Incremental robot learning of new objects with fixed update time," in *International Conference on Robotics and Automation*, 2017.

[24] D. Pathak, P. A. A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," in *International Conference on Machine Learning*, 2017.

[25] R. Kemker, M. McClure, A. Abitino, T. L. Hayes, and C. Kanan, "Measuring catastrophic forgetting in neural networks," in *Conference on Artificial Intelligence*, 2018.

[26] T. Lesort, H. Caselles-Dupré, M. G. Ortiz, A. Stoian, and D. Filliat, "Generative models from the perspective of continual learning," in *International Joint Conference on Neural Networks*, 2019.

[27] N. Kamra, U. Gupta, and Y. Liu, "Deep generative dual memory network for continual learning," *arXiv:1710.10368*, 2017.

[28] M. Saito and S. Saito, "Tganv2: Efficient training of large models for video generation with multiple subsampling layers," *arXiv preprint arXiv:1811.09245, 2018*, 2018.

[29] Y. Pan, Z. Qiu, T. Yao, H. Li, and T. Mei, "To create what you tell: Generating videos from captions," in *ACM Multimedia Conference*, 2017.

[30] C. Vondrick, H. Pirsiavash, and A. Torralba, "Generating videos with scene dynamics," in *Annual Conference on Neural Information Processing Systems*, 2016.

[31] M. Mathieu, C. Couprie, and Y. LeCun, "Deep multi-scale video prediction beyond mean square error," in *International Conference on Learning Representations*, 2016.

[32] E. Denton and R. Fergus, "Stochastic video generation with a learned prior," in *International Conference on Machine Learning*, 2018.

[33] C. Villani, *Optimal transport: Old and new*. Springer Science & Business Media, 2008, vol. 338.

[34] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine, "Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning," in *Annual Conference on Robot Learning*, 2019.