Chapter 2

Related Work

2.1 Taxonomy of Learning from Demonstration

Learning from Demonstration (LfD) embraces all the methods that learn the ability to perform new tasks from expert demonstrations [1, 4], such as imitation learning [5, 6], programming by demonstration [7], behavioral cloning [8, 9] and inverse reinforcement learning [10, 11]. Ravichandar *et al.* [1] categorized LfD based on two criteria. One is how demonstrations are secured, and the other is the outcome of learning as shown in Fig. 2.1.

From the perspective of how the agents (robots) acquire demonstration, LfD

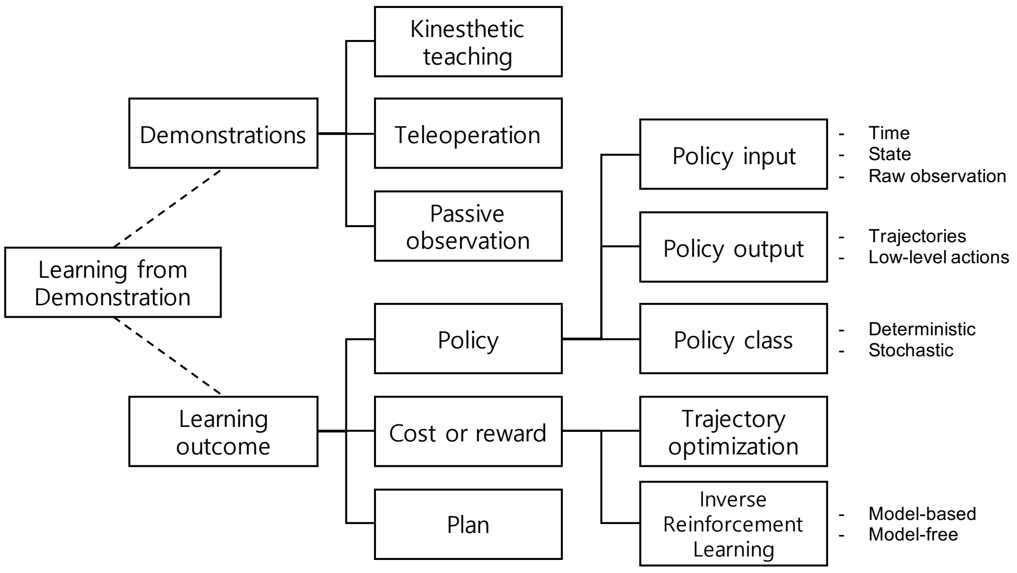

Figure 2.1: Taxonomy of LfD [1].

is divided into three types: kinesthetic teaching, teleoperation, and passive observation [1]. Kinesthetic teaching literally means the expert moves the robot physically to execute the desired task. In the demonstration process, the robot can collect the data such as joint angle and velocity trajectories from their own sensors. The collected demonstration data can be used directly to reproduce motion or learn the task through further learning process [12]. However, when the robot platform has high degree of freedom to be controlled (e.g., anthropomorphic robotic hands) or requires dynamic balancing (e.g, quadruped robots), it is hard to provide appropriate demonstrations through physical contact.

Teleoperation approach delivers the expert demonstration through interfaces such as remote controller and exo-skeleton [1]. LfD via teleoperation is usually adopted to mobile robot such as helicopter [13] due to its remote control ability. In addition, interfaces such as exo-skeleton makes it possible to deliver the demonstration to anthropomorphic robotic hands possible. Although teleoperation approach using interfaces overcomes some limitations of kinesthetic teaching approach, it is hard to make suitable interface system for each platform [1]. Additionally, the demonstrator has to become experienced the interface system to solve the task skillfully.

Aforementioned two approaches are very eidetic approaches from the robot's standpoint, because there is a mapping function that converts demonstration from the expert's domain to the robot's domain, such as the robot's sensor and teleoperation interfaces. Therefore, it is easy to understand the expert demonstration because it is already converted into to the robot's own domain and transmitted as shown in Fig. 2.2. However, ironically, experts are limited in performing demonstrations in these approaches.

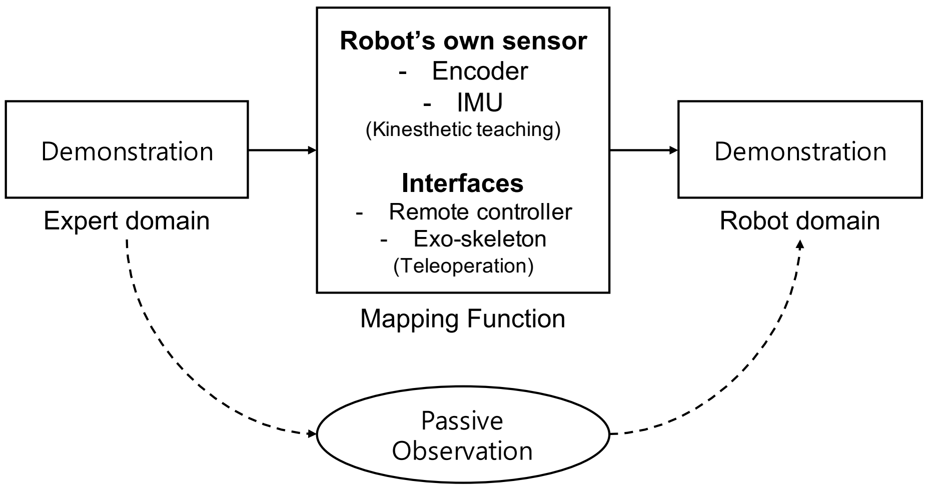

Figure 2.2: Demonstration mapping in learning from demonstration. There is missing link between expert domain and robot domain in passive observation approach.

In the real world, however, the most of LfD is not carried out in these ways. Humans usually only observe the demonstration from the perspective of third person as a passive observer, we do not get egocentric demonstrations [1, 3, 14]. Ravichandar *et al.* [1] categorized this kind of LfD approach as passive observation. In this approach, the robot does not involved in the task, but just observe the expert demonstration. Therefore, the expert is able to perform the demonstration without further hindrance. However, there is missing link between the expert domain and the robot domain in passive observation approach. In order for the robot to perceive the expert demonstration, a mapping function between them is needed [14]. In the recent, machine learning techniques based on deep neural networks have been to the fore the missing link between them [3, 15]. The proposed framework in this thesis is also part of passive observation approach. Sometimes the passive observation approach is referred to as imitation learning

[1, 16], while sometimes the term imitation learning means a more comprehensive sense of LfD. For a clearer discussion, we will discuss about the imitation learning in more detail in the section 2.3.

LfD can also be categorized based on it's learning outcome: policy, cost or reward, and plan [1], as shown in Fig. 2.1. Generally, policy is defined as a mapping function from perceived information of the environment (states) to appropriate actions in recognized situation [1, 17]. Policy-learning methods aim to learn a policy that is able to accomplish the desired task based on the expert demonstration. Policy-learning method can be further categorized based on it's input type of policy, output type of policy and class of policy (e.g., stochastic or deterministic) [1]. For more detail, see [1].

Learning cost or reward functions from demonstration is known as one of the successful branches in LfD literature. Inverse reinforcement learning [10, 11] is the represented approaches in these methods especially. IRL assumes that the expert trying to optimize a hidden unknown reward function and it can be recovered by utilizing provided demonstrations. After that, a reinforcement learning procedure is conducted by leveraging the recovered reward function. We will discuss further about IRL in the subsection 2.3.2.

2.2 Reinforcement Learning

In terms of learning, robot requires supervision such as reward function in reinforcement learning or expert demonstration in imitation learning to learn a new task [3]. In this section, a brief theoretical overview of reinforcement learning framework is presented to describe imitation learning frameworks in the

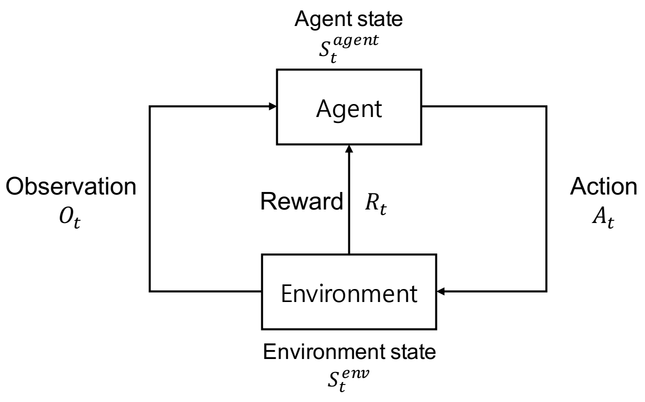

Figure 2.3: The agent and environment interface in RL.

next section.

2.2.1 Markov Decision Processes

RL is a learning process that the agent learning how to achieve a goal from interacting with the environment. The agent who executes sequential decision making is the learner in RL. As shown in Fig. 2.3, at each time step t, the agent receives observation O_t and reward R_t from the environment, and processes observation O_t to recognize present situation. The processed information, which is used to choose the next action A_t, is called agent state S_t^{agent}. The environment receives action A_t executed by the agent and update its own environment state S_{t+1}^{env}, then outputs next observation O_{t+1} and reward R_{t+1}. And this process is repeated recursively as in Eq. (2.1).

$$H_t = O_0, R_0, A_0, O_1, R_1, A_1, \ldots, A_{t-1}, O_t, R_t \tag{2.1}$$

The sequence or trajectory of observations, actions, and rewards is called history H_t. The concept of state is summary of the history and formally it is a function of the history [17].

$$S_t = f(H_t) \tag{2.2}$$

The environment state S_t^{env} is usually invisible to the agent. It is the private representation of the environment. The environment choose next observation and reward based on the environment state and received action from the agent. This relation of input and output is called dynamics. The agent state S_t^{agent} that the information the agent uses to choose next action can be any function of history.

Definition 2.2.1 (Markov state)**.** A state S_t is Markov state if and only if $P[S_{t+1}|S_t] = P[S_{t+1}|S_1, S_2, \ldots, S_t]$

In words, a Markov state is a sufficient statistic of the future. In other words, a Markov state summarizes all useful information from the history to predict future. The environment state S_t^{env} is Markov state and the history H_t is also Markov state, but a observation itself is not usually Markov state [17]. When the agent is able to observe environment state directly (i.e., $O_t = S_t^{env} = S_t^{agent}$), it is called fully observable environment, and formally, this is a Markov decision process (MDP). When the agent is not able to directly observe environment state, it is called partially observable environment, and formally, this is a partially observable Markov decision process (POMDP). In POMDP, the agent has to its own state representation mapping from the history.

MDPs formally describe the RL problem. MDP is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \eta, \gamma \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \mathcal{S} = P(s_{t+1} = s'|s_t = s, a_t = a, s \text{ and } s' \in \mathcal{S} \text{ and } a \in \mathcal{A})$ is a state transition probability,

$\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the reward function, η is the initial state distribution, and $\gamma \in [0, 1]$ is a discount factor.

RL is accomplished through the supervision signal which is called reward. The objective of the agent is to maximize cumulative reward. RL is based on reward hypothesis [17].

Definition 2.2.2 (Reward hypothesis [17])**.** All of goals that we desire to achieve can be described as the maximization of expected cumulative reward.

The cumulative reward is referred to the return G_t.

Definition 2.2.3 (Return [17])**.** The return G_t is the total sum of discounted reward from timestep t,

$$G_t = R_{t+1} + \gamma R_{t+2} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{2.3}$$

The agent interacts with the environment by following policy π. A policy is a mapping from state to action.

Definition 2.2.4 (Policy [17])**.** A policy π is a probability distribution over actions given states,

$$\pi(a|s) = P(A_t = a | S_t = s) \tag{2.4}$$

The state-value function and action-value function evaluate how good is it to be in state s or how good is it to take action a in state s if the agent following policy π.

Definition 2.2.5 (State-value function [17])**.** The state-value function $v_\pi(s)$ of an MDP is the expected return starting from s and then following policy π,

$$v_\pi(s) = E_\pi[G_t | S_t = s] \tag{2.5}$$

Definition 2.2.6 (Action-value function [17])**.** The action-value function $q_\pi(s, a)$ is the expected return taking action a in state s, and then following policy π,

$$q_\pi(s, a) = E_\pi[G_t | S_t = s, A_t = a] \tag{2.6}$$

The objective of RL problem is finding a policy that maximize expected return which is called optimal policy, and formally, it is represented as in Eq. (2.7).

$$\begin{aligned} \pi^* &= \underset{\pi}{\text{argmax}} E_\pi \Big[\sum_t R(s_t, a_t) \Big] \\ &= \underset{\pi}{\text{argmax}} E_{s_1 \sim \eta} [v_\pi(s_1)] \end{aligned} \tag{2.7}$$

2.3 Imitation Learning Frameworks

Since the terminology that imitation learning has been widely used in other sciences [4], several surveys [1, 4] have proposed to use the terminology learning from demonstration, instead of imitation learning, to encompass all the paradigm that robot learns skills from the expert demonstration. However, as research in the related fields has been actively conducted in the recent years, the term imitation learning (or imitation) is used interchangeably with the term learning from demonstration in many researches considered as milestone [3, 6, 14, 15, 18]. Therefore, in this thesis, we will use the term imitation learning as equivalent to learning from demonstration.

In this section, the represented imitation learning frameworks will be discussed, furthermore, relations and comparisons between previous works and the proposed framework in this thesis as well. Traditional imitation learning frameworks can be divided into supervised learning approaches represented by behavioral cloning [8] and inverse reinforcement learning [10] approaches [3, 14, 19]. We briefly discuss about these two approaches in the subsection 2.3.1 and 2.3.2, respectively. Since the proposed framework in this thesis extensively adopts techniques from the generative adversarial networks framework [20], we review it and related imitation learning algorithms in the subsection 2.3.3. Recently, studies on imitation learning from passive (or raw) observation of demonstration, which is obtained from the perspective of a third-person, are being conducted. It will be described in the subsection 2.3.4, because the framework presented in this thesis also belongs to this literature.

2.3.1 Behavioral Cloning

2.3.2 Inverse Reinforcement Learning

Inverse Reinforcement Learning (IRL) approaches assume that expert's behavior can be described by a unknown reward function and it can be inferred from expert's demonstrations [10, 11]. In that IRL tries to recover the reward function (cost function), it is in line with inverse optimal control (IOC) in the control field [1]. While behavioral cloning simply copies the expert demonstration as supervised learning fashion, IRL tries to infer what the expert is trying to achieve.

IRL/IOC is the problem of recovering a reward function whose optimal solution is the given expert demonstrations and obtaining a policy, which is similar to

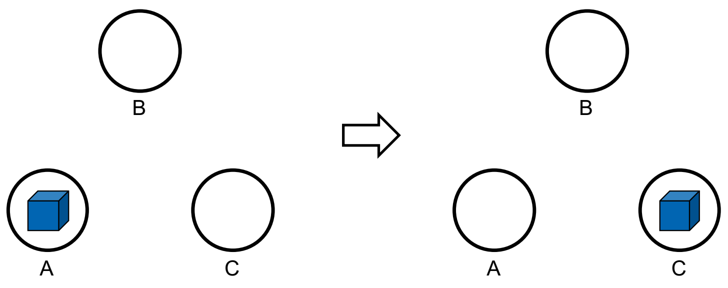

Figure 2.4: Demonstration can be explained by many different reward functions. This demonstration can be explained as “Expert moves box from site A to site C” or as “Expert moves box far away from site A and B.”

expert’s policy, via forward reinforcement learning procedure with the recovered reward function. However, IRL problem is a ill-posed problem, because given expert demonstrations can be represented by many different reward functions [1]. For example, let us assume a expert moves the blue box from site A to site C as shown in Fig. 2.4, this demonstration can be explained as “Expert moves box from site A to site C”. On the other hand, it can also be explained as “expert moves box far away from site A and B.” Therefore, in order to recover the intended reward function, it may be necessary to give hand-crafted features [10, 11]. In addition, it is hard to evaluate that which recovered reward function is better and expert’s demonstration may not be consistent and optimal solution for the task.

To resolve the ambiguity, Ziebart *et al.* [21] propose a probabilistic model approach based on the principle of maximum entropy. The probabilistic model of behavior is represented by Boltzmann distribution as in Eq. (2.8) [22].

$$p_{r_\psi}(\tau) = \frac{1}{Z}\exp(R_\psi(\tau)) \tag{2.8}$$

where $\tau = \{s_1, a_1, \dots, s_t, a_t, \dots, s_T\}$ is trajectory, $R_\psi(\tau) = \sum_t r_\psi(s_t, a_t)$ is the sum of the learned reward function $r_\psi(s_t, a_t)$ parameterized by ψ along the trajectory τ, and $Z = \int \exp(R_\psi(\tau))d\tau$ is the partition function.

With this probabilistic model, trajectories with same reward have same probability. In addition, higher rewarded trajectories are exponentially more likely and less rewarded trajectories are exponentially less likely [21]. By maximizing the likelihood as in Eq. (2.9), the agent performs equivalently to the expert's demonstrations.

$$\begin{aligned}\max_\psi L(\psi) &= \max_\psi E_{\tau\sim\mathcal{D}}[\log p_{r_\psi}(\tau)] \\ &= \max_\psi \frac{1}{N}\sum_{\tau\in\mathcal{D}} \log p_{r_\psi}(\tau) \\ &= \max_\psi \frac{1}{N}\sum_{\tau\in\mathcal{D}} R_\psi(\tau) - \log Z\end{aligned} \tag{2.9}$$

where $\mathcal{D}$ is the set of expert demonstrations and N is the number of given expert demonstrations.

To solve the maximum likelihood problem, we can get the gradient of $L(\psi)$ as in Eq. (2.10).

$$\begin{aligned}\nabla_\psi L(\psi) &= \frac{1}{N}\sum_{\tau\in\mathcal{D}} \nabla_\psi R_\psi(\tau) - \frac{1}{Z}\int \exp(R_\psi(\tau))\nabla_\psi R_\psi(\tau)d\tau \\ &= E_{\tau\sim\pi^*}[\nabla_\psi R_\psi(\tau)] - E_{\tau\sim\pi_{r_\psi}}[\nabla_\psi R_\psi(\tau)]\end{aligned} \tag{2.10}$$

where π^* is optimal policy given by the expert, and π_{r_ψ} is current optimal policy with respect to the current reward function r_ψ.

Eq. (2.10) can be interpreted as maximizing the likelihood is equivalent to increase the expectation of gradient of the reward with respect to the expert demonstrations and decrease the expectation of gradient of the reward with respect to the current policy. When the gradient is zero (i.e., $\nabla_\psi L(\psi) = 0$, at the maximum), the two expectation become equal. In other words, trajectory distributions of the expert policy and the current policy with respect to the current reward function are matched.

To obtain the gradient of the likelihood, the expectation over the expert optimal policy can be estimated from the given demonstrations. However, to estimate the expectation over the current policy, we need to solve the MDP to get the current optimal policy with respect to the current reward function. The expectation over the current policy can be rewritten as in Eq. (2.11).

$$\begin{aligned} E_{\tau\sim\pi_{r_\psi}}[\nabla_\psi R_\psi(\tau)] &= E_{\tau\sim\pi_{r_\psi}}[\nabla_\psi \sum_{t=1}^{T} r_\psi(s_t, a_t)] \\ &= \sum_{t=1}^{T} E_{(s_t,a_t)\sim\pi_{r_\psi}}[\nabla_\psi r_\psi(s_t, a_t)] \\ &= \sum_{t=1}^{T} \int\int \mu_t(s_t, a_t) \nabla_\psi r_\psi(s_t, a_t) ds_t da_t \end{aligned} \tag{2.11}$$

where $\mu_t(s_t, a_t)$ is a probability of state-action visitation.

In summary, given ψ and expert demonstrations, solve the MDP to acquire current optimal policy with respect to the current reward function r_ψ, then obtain the state-action visitation probability using the current optimal policy. After that compute the gradient of the likelihood and update the reward function r_ψ.

Levine *et al.* [23] have proposed nonlinear reward function approximation for IRL via Gaussian Processes (GP) to overcome limitations of reward function, which is a simple linear combination of features, in prior works. However, GP is not suitable for high-dimensional complex problems, because of its computational complexity. To remedy this problem, Wulfmeier *et al.* [24] have proposed to employ deep neural network as a reward function approximator. They showed that deep neural network not only can handle large and complex reward structure, but also can capture the reward function from raw input without any hand-crafted features. Although they showed a promising direction of deep neural network as a reward function approximator in IRL problem, it has not been applied to the real-world high-dimensional problem such as robot manipulation or autonomous driving.

As mentioned above, previous works suffer with solving MDP for optimal policy with respect to current reward function in the inner loop. This forward procedure is not only expensive computationally, but also difficult to apply to unknown dynamic systems. In addition, since many of previous works require hand-crafted features to recover reward function, their approaches struggle with high-dimensional, complex problems. Finn *et al.* [25] have proposed Guided Cost Learning (GCL) based on efficient sample-based approximation and importance sampling technique to address the presented issues. Since learning an optimal policy with respect to current reward function is computationally expensive procedure, they just take one step policy optimization to improve the policy by maximizing the learned cost and entropy as in Eq. (2.12).

$$\max_{\theta} L(\theta) = \max_{\theta} E_{\tau\sim\pi_\theta}[R_\psi(\tau)] - E_{\tau\sim\pi_\theta}[\log \pi_\theta(\tau)] \tag{2.12}$$

However, in that case, we can not get the exact expectation of the gradient of the current reward function in Eq. (2.10). To estimate the gradient of likelihood,

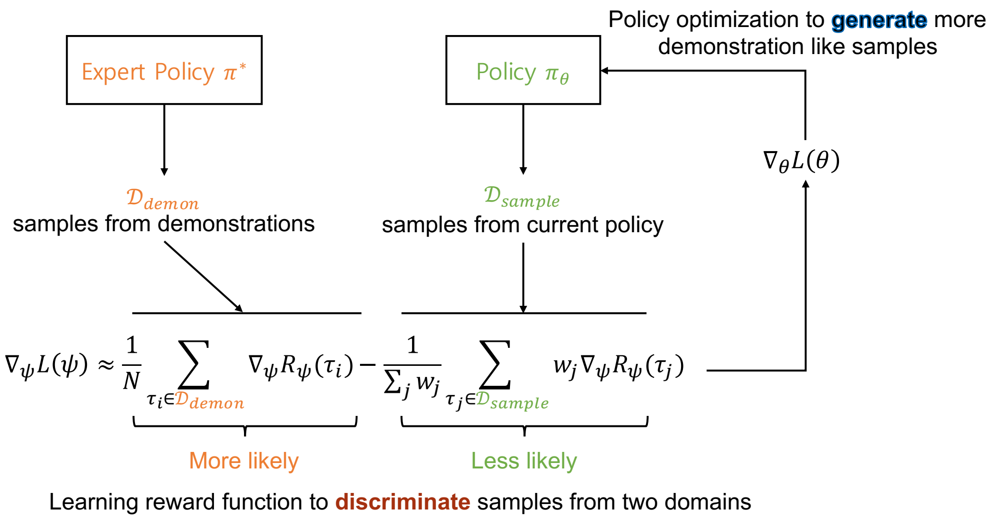
Policy optimization to generate more demonstration like samples
Expert Policy π^*
Policy π_θ
$\nabla_\theta L(\theta)$
$\mathcal{D}_{demon}$ samples from demonstrations
$\mathcal{D}_{sample}$ samples from current policy
$\nabla_\psi L(\psi) \approx \frac{1}{N} \sum_{\tau_i \in \mathcal{D}_{demon}} \nabla_\psi R_\psi(\tau_i) - \frac{1}{\sum_j w_j} \sum_{\tau_j \in \mathcal{D}_{sample}} w_j \nabla_\psi R_\psi(\tau_j)$
More likely
Less likely
Learning reward function to discriminate samples from two domains

Figure 2.5: In GCL, Learning reward function procedure can be interpreted as trying to discriminate demonstration samples from samples generated by current policy, and policy optimization procedure can be interpreted as trying to generate samples more looks like demonstration samples based on current reward function.

they impose importance sampling as in Eq. (2.13).

$$\nabla_\psi L(\psi) \approx \frac{1}{N} \sum_{\tau_i \in \mathcal{D}_{demon}} \nabla_\psi R_\psi(\tau_i) - \frac{1}{\sum_j w_j} \sum_{\tau_j \in \mathcal{D}_{sample}} w_j \nabla_\psi R_\psi(\tau_j) \tag{2.13}$$

where $w_j = \frac{\exp(R_\psi(\tau_j))}{\pi(\tau_j)}$.

The GCL procedure can be summarized as in Fig. 2.5. As shown in Fig. 2.5, GCL looks like a competition between policy π_θ and reward function r_ψ such as generator and discriminator in Generative Adversarial Network [20]. We will describe more details in the next subsection 2.3.3.

2.3.3 GAN-based Frameworks

Generative Adversarial Networks

GANs [20] is a generative model estimating framework based on minimax problem between a generator and a discriminator. The objective of the discriminator is to estimate the probability that a sample came from the real data distribution rather than the generative model distribution, while the objective of the generator (i.e., generative model) is to deceive the discriminator by capturing the real data distribution. Since GAN framework does not need approximate inference or Markov chains to catch the real data distribution and also deep neural networks can be used for the generator and the discriminator, it receives the attention of a wide research area.

Let $\mathbf{x} \sim p_{data}(\mathbf{x})$ be a sample from the real data distribution $p_{data}(\mathbf{x})$, $\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})$ be a input random vector, G be the generator which takes $\mathbf{z}$ as input and generates a sample $G(\mathbf{z}) = \tilde{\mathbf{x}} \sim p_g$, and D be the discriminator which takes a sample $\mathbf{x}$ as input and outputs the probability that the sample came from the real data distribution p_{data}. G and D is trained via solving the minimax two-player game as in Eq. (2.14) and Eq. (2.15).

$$\min_G \max_D V(D, G). \tag{2.14}$$

where the value function V, given by

$$V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{data}(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))]. \tag{2.15}$$

G is trained to minimize $\log(1 - D(G(\mathbf{z})))$, which means the generator trying to fool the discriminator into believing that the generated sample $\tilde{\mathbf{x}} = G(\mathbf{z})$ came from the real data distribution $p_{data}(\mathbf{x})$ (i.e., $D(G(\mathbf{z})) = 1$). D is trained concurrently to distinguish the real data and the generated data (i.e., $D(\mathbf{x}) = 1$ and $D(G(\mathbf{z})) = 0$).

Goodfellow *et al.* [20] show that the minimax problem converges to a global optimum $p_g = p_{data}$. Therefore, the generator G can generate a sample that seems like it's from the real data distribution p_{data}. In the perspective of the generator G, Eq. (2.14) can be interpreted as minimizing problem that minimizes the Jensen-Shannon divergence between the real data distribution p_{data} and the generator distribution p_g (i.e., $JSD(p_{data} \| p_g)$) (For more detail, see the *proof* of the **Theorem 1.** in [20].). In other words, GAN framework can be described as distribution distance minimizing problem between p_g and p_{data}, as a result, GAN frameworks that introduce other effective distance metrics such as Wasserstein distance (WGAN [26]) or least square distance (LSGAN [27]) were proposed for more stable training procedure.

Generative Adversarial Imitation Learning

As described in the previous subsection 2.3.2, the learning procedure of GCL closely resemble that of GAN. Finn *et al.* [22] have shown that GAN with generator, whose density can be evaluated, mathematically equivalent to GCL and also closely related with Energy-based models. During GCL, we alternatively optimize Eq. (2.9) and Eq. (2.12) to distinguish expert demonstration samples from samples generated by current optimal policy and to match the sample distribution with expert demonstration sample distribution.

In GAN framework, the optimal discriminator for a fixed generator $\pi_\theta(\tau)$ is given by Eq. (2.16).

$$D^*_{\pi_\theta}(\tau) = \frac{p(\tau)}{p(\tau) + \pi_\theta(\tau)} \tag{2.16}$$

where $p(\tau)$ is the real data distribution.

GAN framework optimize the discriminator to directly output the value $D^*_{\pi_\theta}$ without explicit estimation of $p(\tau)$. In the perspective of GCL, we can rewrite Eq. (2.16) as in Eq. (2.17).

$$D_\psi(\tau) = \frac{\frac{1}{Z}\exp(R_\psi(\tau))}{\frac{1}{Z}\exp(R_\psi(\tau)) + \pi_\theta(\tau)} \tag{2.17}$$

In other words, if $\frac{1}{Z}\exp(R_\psi(\tau))$ is equal to the real data distribution $p(\tau)$, then the discriminator becomes optimal discriminator regardless of the generator π_θ. In addition, it significantly resolves the inherent unstable characteristic of GAN training procedure [22]. Furthermore, since when maximizing discriminator loss as in Eq. (2.18), Z becomes the importance sampling estimate of the partition function, there is no need to struggle to estimate the partition function.

$$L_{disc}(D_\psi) = E_{\tau\sim p}[\log D_\psi(\tau)] + E_{\tau\sim\pi_\theta}[\log(1 - D_\psi(\tau))] \tag{2.18}$$

Based on the modified discriminator in Eq. (2.17), they have shown the mathematical equivalence between GAN and GCL. For more details, see [22]. Note that GCL approach obtains both the learned reward function to describe expert behaviors and correspond optimal policy.

A little earlier than the work of Finn *et al.* [22], Ho and Ermon [18] have also been proposed a GAN-based framework for model-free imitation learning referred to Generative Adversarial Imitation Learning (GAIL). The proposed algorithm

GAIL omits RL procedure in an inner loop, which is indispensable in traditional IRL frameworks, by directly learning a policy from expert demonstrations. They employ GAN to fit distributions of state-action visitation of expert policy [18]. They defined occupancy measure that can be interpreted distribution of state-action visitation of given policy π as in Eq. (2.19) and showed that IRL problem can be transformed to occupancy measure matching problem. In other words, IRL can be treated as a problem trying to acquire a policy that matches the occupancy measure of expert demonstrations.

$$\rho_\pi(s,a) = \pi(a|s)\sum_{t=1}^{\infty}\gamma^t P(s_t = s|\pi) \tag{2.19}$$

In addtion, Ho and Ermon showed that an instantiation of proposed framework with regularizer, whose convex conjugate function is as in Eq. (2.20), has a connection with GAN framework.

$$\psi^*_{GA}(\rho_\pi - \rho_{\pi_E}) = \max_{D\in(0,1)^{\mathcal{S}\times\mathcal{A}}} E_{\pi_E}[\log(D(s,a))] + E_\pi[\log(1 - D(s,a))] \tag{2.20}$$

The convex conjugate function of regularizer ψ has same structure of the discriminators in GAN and GCL as in Eq. (2.15) and Eq. (2.18), respectively. The $D(s,a)$ in Eq. (2.20) acts like a discriminator in GAN to distinguish between state-action samples from expert policy and samples from agent policy.

To summarize, in GAIL, parameterized discriminator D_ψ is trained to distinguish expert demonstration from samples of parameterized agent policy π_θ, and the parameterized policy is trained to generate state-action samples that is similar to the expert demonstration by using policy optimization algorithm (such as Trust Region Policy Optimization (TRPO) or Proximal Policy Optimization

(PPO)), utilizing $log(D_\psi)$ as a reward function, as in Eq. (2.21).

$$\nabla_\theta L(\theta) \approx \frac{1}{M} \sum_{j=1}^{M} \nabla_\theta \log \pi_\theta(\tau_j) \log D_\psi(\tau_j) + \lambda \nabla_\theta H(\pi_\theta) \tag{2.21}$$

As described above, GCL and GAIL is trained in adversarial training manner to obtain a policy from expert demonstration. The main difference between them is that while GCL uses modified discriminator utilizing the generator's density, GAIL uses standard binary classification discriminator. Therefore, while GCL is able to learn both the explicit reward function and the policy as a result, GAIL only can obtain the policy as a result because the reward function is represented implicitly in the discriminator [22]. Note that based on the taxonomy of learning from demonstration in the previous section 2.1, GAIL is not affiliated to IRL framework because it does not acquire explicit reward function as a learning result [1].

Third-Person Imitation Learning

The previous imitation learning frameworks, which are described in the above, are based on first-person perspective demonstrations (state-action pairs). In other words, they do not require any mapping function to transform demonstrations in the expert domain to the robot (learner) domain, as shown in Fig. 2.2. However, as described in the section 2.1, collecting the ego-centric demonstration is quite difficult and unnatural process. In the nature, humans or animals observe others execute task in the third-person perspective, infer the goal of the task and imitate others' behavior, by transforming it to their own domain, to accomplish the goal.

Stadie *et al.* [14] have proposed a method for unsupervised third-person imitation learning by employing domain confusion technique [28] and GAN framework [20]. The proposed framework is trying to recover a domain-agnostic representation of the agent's observations to represent expert demonstrations without the mapping function and reformulate GAIL framework based on this representation. Contrary to the work of Finn *et al.* [25] and Ho & Ermon [18] which consider imitation learning from fisrt-person perspective raw sensory data, Stadie *et al.* [14] have considered imitation learning from third-person perspective raw sensory data.

To recover a domain-agnostic representation, Stadie *et al.* [14] divide discriminator into a feature extractor which is extracting domain-agnostic representation, and a classifier which is distinguishing the expert behaviors from the agent behaviors. The feature extractor is trained by another classifier that determines whether features produced by the feature extractor came from the expert domain or the agent domain. In addition, they also impose technique that flips the sign of the gradient when backpropagating to address competing issue between the behavior classifier and the feature classifier [14]. For more details, see [14].

Although, Stadie *et al.* [14] have shown a promising results of imitation learning with third-person perspective observations, their experiments are limited by simple tasks in simulation environment and viewpoint change, and also they only consider the two domains. However, in practice, humans or animals observe demonstrations come from several variant domain for same task [3] and learning from it.

Adversarial Inverse Reinforcement Learning

In contrast to the frameworks directly extracting policy from demonstration such as GAIL, IRL frameworks recover the reward function that can be used for re-optimization in general environment or reasoning the expert's intentions [1, 29]. Although GCL [22, 25] has shown the promising results, GCL is less capable for complex problems than GAIL, because GCL utilizes samples based on entire trajectories to recover a reward function [29].

Ziebart *et al.* [21] proposed maximum entropy IRL framework to resolve an ambiguity of optimal policy, in other words, there can be many optimal policies that can describe given expert demonstrations. However, there is another ambiguity problem that there can be many reward functions that can describe an optimal policy, that is, the reward function recovered by IRL frameworks can be overfitted to the trained environment, and it is hard to generate optimal policy in general environments [29, 30]. To mitigate this ambiguity problem, Fu *et al.* [29] have proposed adversarial inverse reinforcement learning framework (AIRL) that is able to learn robust reward function referred to as disentangled rewards. AIRL is not based on entire trajectories of roll-outs but samples of state-action pairs to prevent high variance, to end this, Fu *et al.* convert Eq. (2.17) to as in Eq. (2.22) and show that f_ψ in Eq. (2.22) recovers the advantage function of the optimal policy at optimality as in Eq. (2.23).

$$D_\psi = \frac{\exp(f_\psi(s,a))}{\exp(f_\psi(s,a)) + \pi(a|s)} \tag{2.22}$$

$$f_\psi^*(s,a) = \log(\pi^*(a|s)) = A^*(s,a) \tag{2.23}$$

Variational Discriminator Bottleneck

Generative Adversarial MDP Alignment

2.3.4 Imitation Learning from Observation

Most of previous imitation learning frameworks have required demonstrations from the first-person point of view or with egocentric action information. For example, the expert holds and moves the robot physically following the desired trajectory to acquire the expert demonstration data. However, it is hard to collect these kind of demonstration data and also is expensive and time consumptive process. Although some of previous works trying to use passive observation in the perspective of third-person, they need additional devices such as motion tracking sensor to measure the expert motion and they also require post processing for collected sensor data.

Imitation learning in this way is bound to be very limited. Humans or intelligent animals have capability to imitate demonstrations that is observed in third-person point of view without egocentric action information and sometimes is demonstrated by the expert who has difference in embodiment, by transforming observed demonstration into their action space. This type of imitation learning is referred as imitation from observation (IfO) or imitation learning from observation (ILfO) [3, 31]. In this thesis, for clarity, we term Liu *et al*'s work as IfO and this problem as ILfO. ILfO frameworks will be able to further broaden the field of application of IL, such as task learning using YouTube videos.

TPIL [14] has shown the possibility of ILfO by learning a domain agnostic representation. However, their results are limited in simulation environments with very small difference in viewpoint. In addition, TPIL is only able to handle viewpoint differences between only two context, while demonstrations usually come from various contexts practically [3]. In this subsection, we will discuss recent progresses in researches about ILfO problem.

Time-Contrastive Networks

Sermanet *et al.* [15] have proposed a self-supervised approach, which is called time-contrastive network (TCN) to learn viewpoint invariant representation of demonstrations by leveraging videos recorded from multiple viewpoints and imitation learning framework using the learned representation. In addition, the proposed framework is able to handle embodiment difference between demonstrator (the expert) and the agent (the robot), since it learns representation from various demonstration data that have embodiment differences.

For multi-view video aligned in time domain, TCN is implemented using triplet loss [32] to embed frames at the same time closer together and frames at different time farther apart. It enables to learn viewpoint-invariant representation in unsupervised manner. The learned representation allows to map third-person viewpoint demonstration and robot motion information from it's own viewpoint into latent space, and to imitate demonstration by tracking trajectory of embedded demonstration in the latent space. To do so, Sermanet *et al.* [15] have proposed reward function as in Eq. (2.24) using learned representation and execute RL using the reward function to realize imitation learning.

$$R(\mathbf{v}_t, \mathbf{w}_t) = -\lambda_1 \|\mathbf{w}_t - \mathbf{v}_t\|^2 - \lambda_2 \sqrt{\lambda_3 + \|\mathbf{w}_t - \mathbf{v}_t\|^2} \tag{2.24}$$

where $\mathbf{v}_t$ and $\mathbf{w}_t$ are the TCN embeddings of frames in demonstration and robot execution video, respectively, and λ_1, λ_2, and λ_3 are hyperparameter.

Sermanet *et al.* [15] implemented the proposed framework in simulation dish arrangement task and real world pouring task. Although Sermanet *et al.* [15] have proposed self-supervised learning method for viewpoint, embodiment invariant representation and showed promising results in imitation learning using the learned representation, TCN requires multi view demonstration data aligned in time domain to learn representation. This kind of demonstration data is hard to collect or access, furthermore TCN needs task-specific demonstration data [15].

Imitation from Observation

Liu *et al.* [3] have proposed imitation learning framework for imitation learning from observation based on video prediction with context translation that converts context (e.g., viewpoint or embodiment) of a demonstration to robot's context. We will call this framework to IfO as mentioned above. Similar to TCN, IfO tries to learn representation that enables to track demonstration in robot's domain.

Liu *et al.* [3] argued that ILfO problem divided into two sub-problems. One is determining which information from given demonstration should be tracked in agent's context, and the other is which action should be executed to track the information. To address the former problem, Liu *et al.* [3] learning internal representation that represents information independent of context change. For the latter problem, Liu *et al.* [3] defined reward function of the learned representation, and learned a policy to tracking internal representation through a post RL process.

IfO assumed that given demonstrations $\{D_i = [o_0^i, o_1^i, \ldots, o_T^i] | i = 1, 2 \ldots, n\}$

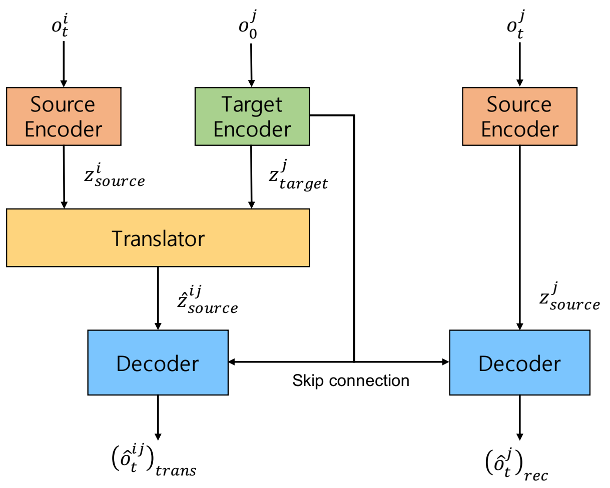

Figure 2.6: The network architecture of IfO consists of four sub-networks: source encoder, target encoder, translator, and decoder.

in different contexts $w_{1,2,...,n}$ are aligned in time domain and the only first frame of demonstration o_0^i can be used to extract context information. The context translation model is trained to convert demonstration D_i in one context w_i to demonstration D_j in the other context w_j using first frame o_0^j of demonstration D_j. The proposed model consists of four sub-networks as shown in Fig. 2.6.

The source encoder and target encoder encode source observation o_t^i in context w_i and target context observation o_0^j in context w_j into embedding z_{source}^i and z_{target}^j, respectively. The encoded embedding z_{source}^i is translated into target context $\hat{z}_{source}^{ij}$ by the translator given target context embedding z_{target}^j. The decoder reconstructs observation image $(\hat{o}_t^j)_{trans}$ in context w_j. There are skip connections between each layer of the encoder and decoder. Meanwhile, the source encoder and decoder recover source observation o_t^j in context w_j directly without translator. The encoded source embedding and recovered observation are referred as z_{source}^j and $(\hat{o}_t^j)_{rec}$, respectively. The context translation model is trained to minimize translation error loss $L_{trans} = \|(\hat{o}_t^j)_{trans} - o_t^j\|^2$. Liu *et al.* [3] trained the model with additional loss $L_{rec} = \|(\hat{o}_t^j)_{rec} - o_t^j\|^2$ and $L_{align} = \|\hat{z}_{source}^{ij} - z_{source}^j\|^2$ to regularize the representation to provide useful information for tracking demonstration.

To train the agent tracking the representation of demonstration, Liu *et al.* [3] have defined reward functions using the learned representation and translation model as in Eq. (2.25).

$$R(o_t^l) = R_{feat}(o_t^l) + \lambda R_{img}(o_t^l) \tag{2.25}$$

$$R_{feat}(o_t^l) = -\|z_{source}^l - \frac{1}{n}\sum_i^n \hat{z}_{source}^{il}\|^2 \tag{2.26}$$

$$R_{img}(o_t^l) = -\|o_t^l - \frac{1}{n}\sum_i^n (\hat{o}_t^{il})_{trans}\|^2 \tag{2.27}$$

where o_t^l is the observation of the agent's execution at time t and λ is a hyperparameter. For post RL process, Liu *et al.* [3] used trust region policy optimization (TRPO) [33] and guided policy search (GPS) [34] for simulation environment and real world environment, respectively, as a RL algorithm.

IfO has shown much more robust and superior results than TPIL, GAIL and the perceptual rewards method [35] in several simulation environment. In addition, IfO has shown promising results in several real world environment experiments. However, IfO requires demonstrations aligned in time domain which is hard to collect and considerable number of demonstrations to train the proposed translation model. Additionally, while IfO is able to handle viewpoint differences, it is not able to handle embodiment differences between the expert and the agent.

Behavioral Cloning from Observation

Torabi *et al.* [36] have proposed behavioral cloning from observation (BCO) to solve imitation learning from observation problem by leveraging pre-trained inverse dynamics model. BCO learns a task-independent inverse dynamics model from interaction with the environment using random policy to adapt behavioral cloning in situation where the demonstration action data is not available. The learned inverse dynamics model is used to infer action that caused given state transition. In other words, state-only demonstration trajectory is able to be converted to state-action demonstration trajectory by using the learned inverse dynamic model, and therefore BC becomes available. While Torabi *et al.* [36] have shown comparable results to GAIL which requires action information of demonstration, their results is limited low-dimensional state scenarios, not high-dimensional raw visual demonstration scenarios. Furthermore, learning inverse dynamics model requires a large amount of interaction, although it is required in pre-training step.

Generative Adversarial imitation from observation

While IfO and TCN have shown promising results in ILfO problems, they relatively more concentrated on perception (i.e., learning latent representation) of the demonstration than learning control policy that is performed by normal RL algorithm with pre-defined surrogate reward function. Torabi *et al.* [37] have proposed an imitation learning algorithm called generative adversarial imitation learning from observation (GAIfO) that is able to imitate from the observation of demonstration by trying to recover state-transition-only cost function of the expert without pre-defined reward function.

GAIfO is instantiation of GAIL, which uses GANs to match the distribution of state and action pairs of the agent and the expert, in ILfO problem. Torabi *et al.* [37] defined a cost function as a function of state transition $c : \mathcal{S} \times \mathcal{S} \rightarrow \mathbb{R}$ rather than as a function of state and action pairs $c : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ as in common IRL and RL frameworks. By doing so, Torabi *et al.* [37] have formulated problem of IRL from observation as in Eq. (2.28)

$$IRLfO_{\psi}(\pi_E) = \tilde{c} = \underset{c \in \mathbb{R}^{\mathbb{S} \times \mathbb{S}}}{\operatorname{argmax}} -\psi(c) + (\min_{\pi} E_{\pi}[c(s, s')] - E_{\pi_E}[c(s, s')]) \quad (2.28)$$

where ψ is a convex cost function regularizer. Then, any RL algorithm find out optimal imitation policy using $\tilde{c}$. Based on the definition in Eq. (2.28), similar to GAIL, Torabi *et al.* [37] have derived GAIfO as in Eq. (2.29). For more detail, see [18, 37].

$$\min_{\pi} \psi^{*}_{GA}(\rho^{s}_{\pi} - \rho^{s}_{\pi_E}) = \min_{\pi} \max_{D} E_{\pi}[\log(D(s, s'))] + E_{\pi_E}[\log(1 - D(s, s'))] \quad (2.29)$$

Torabi *et al.* [37] have shown that GAIfO is able to be implemented in not only low-dimensional state environment, but also high-dimensional raw visual input environment. However, their implementations are limited to fixed viewpoint and embodiment demonstration in simulation environment. Additionally, GAIfO requires many interaction with the environment, because it is based on on-policy policy optimization technique. Therefore, GAIfO is hard to be adopted to real world tasks such as manipulation.

Imitating Latent Policies from Observation

GAIL, GAIfO and TPIL, which are based on GAIL framework, require considerable interaction data with the environment to imitate the expert policy. While IfO and TCN have shown promising results in imitation learning from observation without actions, they need post-RL process with rewards that are consist of learned features trajectory. Edwards *et al.* [38] have proposed an IL algorithm that is able to imitate the expert policy directly without additional RL process. The proposed algorithm, which is referred as imitating latent policies from observation (ILPO), learns policies from demonstration in latent state and action space as offline manner and then learn the mapping between latent action space and real action space using a few interaction with the environment [38]. Edwards *et al.* [38] define discrete latent action space to predict a multi-modal forward dynamics model using the demonstration observations.

Latent policy learning consists of two components: learning latent forward dynamics model and learning latent policy using the latent forward dynamics model. Latent forward dynamics model $G_\theta(E_p(s_t), z)$ predicts the next state s_{t+1}, given an expert state s_t and latent action z, where E_p is an embedding. G is trained to minimize state prediction error as in Eq. (2.30).

$$L_{min} = \min_{z} \|s_{t+1} - G_\theta(E_p(s_t), z)\|^2 \tag{2.30}$$

Latent policy $\pi_w(z|E_p(s_t))$ predicts latent action z, given expert state s_t, that generates similar state transition with expert state transition. π_w is trained to minimize state transition error as in Eq. (2.31)

$$L_{exp} = \|s_{t+1} - \hat{s}_{t+1}\|^2 \tag{2.31}$$

where $\hat{s}_{t+1} = E_{\pi_w}[s_{t+1}|s_t] = \sum_z \pi_w(z|s_t)G_\theta(E_p(s_t), z)$.

After learning latent policy, action remapping learning, that learning a mapping between latent action space and real action space, is conducted to imitate expert demonstration. Although, interaction with the environment is inevitable, learning only a mapping from true action to latent action, rather than a full inverse dynamics model as in BCO, is enough for ILPO [38].

First, collecting agent state-action transition (s_t, a, s_{t+1}) data using any policy such as random policy. Second, predict latent action given state transition s_t, s_{t+1} using the learned latent forward dynamics model G_θ. Then, remapped policy $\pi_\zeta(a_t|z_t, E_a(s_t))$ is trained in a supervised manner with the predicted latent action and real action as in Eq. (2.32).

$$L_{map} = \log \frac{\pi_\zeta(a_t|z_t, E_a(s_t))}{\sum_a \pi_\zeta(a|z_t, E_a(s_t)} \tag{2.32}$$

where $z_t = \underset{z}{\operatorname{argmin}}\|E_p(s_{t+1} - E_p(G_\theta(E_p(s_t), z))\|^2$.

Edwards *et al.* [38] have shown promising results for imitation learning from observation problem not only in the environments that agent internal state is available but also in the environments that only high-dimensional visual observation is available. While Edwards *et al.* [38] have shown that ILPO outperforms BCO in several environments and ILPO is more sample efficient than BCO, their application is limited to discrete action environment and stochastic demonstrations.

Automated Visual Instruction following with Demonstrations

2.4 Meta-Learning

2.5 Video Prediction Frameworks

It is a very key ability in learning that, given a new situation that has not been seen before, to infer or imagine how to solve a new situation accordingly based on one's experience or the experience of observing other's behavior. However, for high-dimensional information such as images, predicting its future stream is a very challenging problem because of complexity in the real-world such as interaction between objects and occlusions. Nevertheless, in the field of computer vision, much progress has been made recently regarding the video prediction problem. A complete review of video prediction in the perspective computer vision literature is outside the scope of this thesis. Therefore, this section provides a brief overview about video prediction especially related with robot learning and the proposed framework in this thesis.

2.5.1 Visual Foresight

Contrary to previous works in video prediction literature which needs labels such as pose information or segmentation mask, Finn *et al.* [39] proposed action-conditioned video prediction framework to model pixel motion explicitly without any labeled data for supervision. Since the proposed model acquires appearance information from previous frame and predicts explicitly its pixel motion distribution conditioned on robot's action, the model is able to adapt to unseen objects at training time. In addition, the model can be adopted for planning or decision making, because it has capability to imagine future frames based on various action sequences. The model is built based on convolutional LSTM [40] and is trained on about 1.5 million video frames.

Finn *et al.* [41] proposed a planning method that combining action-conditioned video prediction framework [39] with model-predictive control for nonprehensile pushing task. Thanks to the action-conditioned video prediction model, the proposed approach can be trained self-supervised fashion without any labeled data or reward function, and can handle unseen object in training time. The goal in planning process is specified by choosing source pixels and its corresponding goal pixel positions in image. The proposed method samples sequence of actions based on cross-entropy method and the samples are evaluated by video prediction model. Finn *et al.* [41] show a promising results that video prediction model can be used to robot planning and decision making.

Although Finn *et al.* [41] have shown that a promising direction of robot planning via video prediction framework, their video prediction framework has limitation that it is vulnerable to a occlusion by robot arm or other objects. To overcome the limitation, Ebert *et al.* [42] propose video prediction model with temporal skip connection. In contrast to the previous method [39, 41], which simply predicts transformations only from the previous image, Ebert *et al.* [42] introduce skip connection between predicted transformation and all previous images to predict the transformations conditioned on of all previous images. Through the temporal skip connection, information on occluded objects can be preserved, and objects disappeared from view due to temporary occlusion can be tracked continuously. In addition, Ebert *et al.* [42] also propose to use Euclidean distance in pixel domain as objective function of visual MPC for better performance.

Xie *et al.* [43] have presented a approach based on visual foresight framework for improvisational tool use problem that figuring out how to achieve the goal using the objects in image observation as tools. To resolve this problem, they propose build generalizable models using self-supervised data and train action proposal model using diverse demonstration data. The proposed approach, which is combining imitation learning and visual MPC, outperforms each individual approach [43]. Contrary to most of previous works in imitation learning literature, their approach leverage not demonstrations of a single task, but demonstrations of many different tasks. The demonstrations from many different tasks can be used not only to explore state space, which is rarely visited via random policy, but also to guide sampling-based planner when it has difficulty to find appropriate solutions. To leverage demonstrations from many different tasks, Xie *et al.* presented an action proposal model that outputs directly a distribution over a sequence of actions conditioned on the initial image and state [43]. Although Xie *et al.* [43] have shown that a promising direction of combining imitation learning and visual MPC for improvisational tool use, they take advantage of kinesthetic and fixed viewpoint demonstrations (i.e., egocentric observations), not a passive observations in the perspective of third-person.

2.5.2 Motion-Content Network

While visual foresight predict pixel transformation of next frame, Villegas *et al.* [44] have proposed Motion-Content Network (MCnet) for future frame prediction of natural video that directly generate image pixel value of next frame. Villegas *et al.* [44] leverage spatio-temporal correlations of video for unsupervised learning of video prediction. Villegas *et al.* [44] simplify video generation problem by decomposing complicated transition of pixels into motion and content representing temporal dynamics and spatial layout of videos respectively. MCnet consists of motion and content encoder pathway that extract features of local dynamics of spatial regions and spatial layout of video frames respectively, then MCnet predicts future frame based on content and motion information up to the last frame.

Motion encoder is made up of convolutional LSTM to extract temporal dynamics feature from given image difference stream, while content encoder consists of CNN to extract spatial layout feature from given the last frame. Villegas *et al.* [44] argue that this asymmetric architecture enables to learning motion and content information separately without any supervision. Decoder that has residual connection with encoders takes extracted features as inputs and outputs pixel-level prediction of next frame.

For training the proposed network, Villegas *et al.* [44] implement GAN loss as in Eq. (2.33) to generate sharp and realistic prediction image, since average pixel-level error loss L_{img} tends to generate blurry image.

$$L = \alpha L_{img} + \beta L_{GAN}, \tag{2.33}$$

$$L_{img} = L_p(\mathbf{x}_{t+k}, \hat{\mathbf{x}}_{t+k}) + L_{gdl}(\mathbf{x}_{t+k}, \hat{\mathbf{x}}_{t+k}), \tag{2.34}$$

$$L_p(\mathbf{y}, \mathbf{z}) = \sum_{k=1}^{T} \|\mathbf{y} - \mathbf{z}\|^p, \tag{2.35}$$

$$L_{gdl}(\mathbf{y}, \mathbf{z}) = \sum_{i,j}^{h,w} |(|\mathbf{y}_{i,j} - \mathbf{y}_{i-1,j}| - |\mathbf{z}_{i,j} - \mathbf{z}_{i-1,j}|)|^{\lambda} \tag{2.36}$$

$$+ |(|\mathbf{y}_{i,j-1} - \mathbf{y}_{i,j}| - |\mathbf{z}_{i,j-1} - \mathbf{z}_{i,j}|)|^{\lambda} \tag{2.37}$$

where, $\mathbf{x}_{t+k}$ is the target frame and $\hat{\mathbf{x}}_{t+k}$ is the predicted frame.

Although MCnet have shown promising results for video prediction in several benchmark datasets, it needs to take image difference stream for future frame prediction that is not appropriate motion planning application.

2.5.3 MoCoGAN

Tulyakov *et al.* [2] have proposed Motion and Content decomposed Generative Adversarial Network (MoCoGAN), which is similar to MCnet [44], that decompose image space into motion and content subspace to generate video clip. While MCnet predicts future frames using image difference stream, MoCoGAN generates sequence of video frames from sequence of random vectors.

Tulyakov *et al.* [2] indicate that assuming a video clip as a point in the latent space only complicates the problem, because every single video clips such as same video clips with different play speed are located as different points in the latent space. Tulyakov *et al.* [2] have proposed to consider a video clip as a traversing in the image latent space. Furthermore, they recognize video as objects performing actions and assume that the image latent space can be decomposed into motion and content subspaces [2]. This decomposing makes it possible to generate more controllable and flexible videos such as videos with different content of the same motion or videos with different motions of the same content [2].

A latent space of images $Z_I \in \mathbb{R}^d$ is decomposed into motion subspace $Z_M \in \mathbb{R}^d_M$ and content subspace $Z_C \in \mathbb{R}^d_C$, where $d = d_M + d_C$. In addition, a point $\mathbf{z} \in Z_I$ represents an image and a path in Z_I, $[\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_t] = [\{\mathbf{z}_C, \mathbf{z}_{M,1}\}, \{\mathbf{z}_C, \mathbf{z}_{M,2}\}, \ldots, \{\mathbf{z}_C, \mathbf{z}_{M,t}\}]$, represents a video, where $\mathbf{z}_C \in Z_C$ and $\mathbf{z}_{M,t} \in Z_M$. Tulyakov *et al.* [2] assume content vector to be constant in short

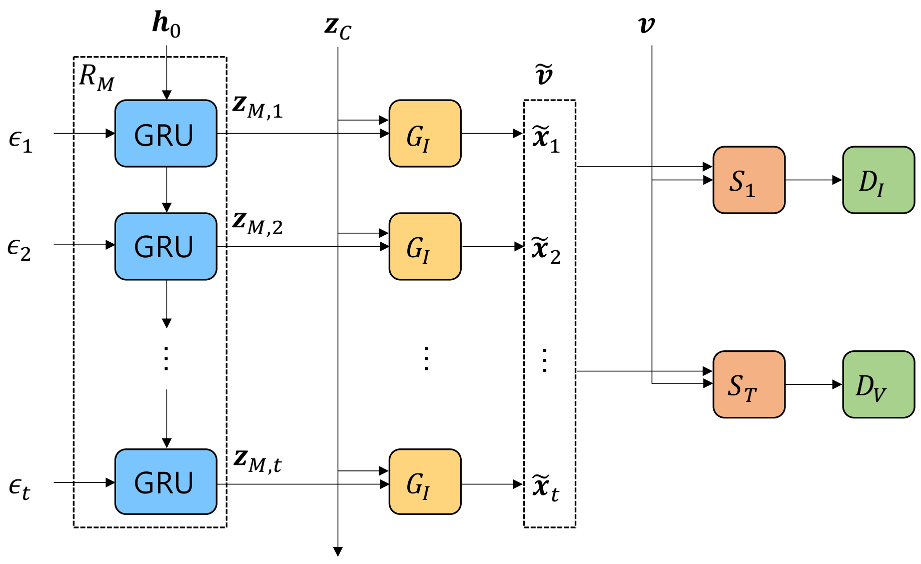

Figure 2.7: The network architecture of MoCoGAN [2]. R_M is motion generator which consist of GRU network, G_I is image generator from given motion and content latent vector, S_1 and S_T are frame sampler which samples one frame and T frames from given video clip, respectively, D_I is image discriminator and D_V is video discriminator.

video clip and model content subspace as Gaussian distribution. Although the path in motion subspace Z_M, $[\mathbf{z}_{M,1}, \mathbf{z}_{M,2}, \dots, \mathbf{z}_{M,t}]$, represents motion of the content in video, not all paths in Z_M produce meaningful motion [2]. To learn to generate meaningful paths, Tulyakov *et al.* [2] design a motion generation process using a GRU network [45], a type of recurrent neural network (RNN).

MoCoGAN consists of motion generator, image generator, image discriminator and video discriminator as shown in Fig. 2.7. The motion generator R_M takes random noise vector sequence, $[\epsilon_1, \epsilon_2, \dots, \epsilon_t]$, sampled from Gaussian distribution as input and generates motion latent vector sequence $[\mathbf{z}_{M,1}, \mathbf{z}_{M,2}, \dots, \mathbf{z}_{M,t}]$. The motion generator R_M is trained to generate meaningful motion, the image generator G_I is trained to generate images looks like real video frame, the image and video discriminator are trained to distinguish image and video, which are sampled from real video clip, from image and video generated by motion and image generator. To train MoCoGAN, Tulyakov *et al.* [2] propose the GAN style objective function as in Eq. (2.38).

$$\begin{aligned}\max_{G_I, R_M} \min_{D_I, D_V} & E_{\mathbf{v}}[-\log D_I(S_1(\mathbf{v}))] + E_{\hat{\mathbf{v}}}[-\log(1 - D_I(S_1(\hat{\mathbf{v}})))] \\ & + E_{\mathbf{v}}[-\log D_V(S_T(\mathbf{v}))] + E_{\hat{\mathbf{v}}}[-\log(1 - D_V(S_T(\hat{\mathbf{v}})))]\end{aligned} \tag{2.38}$$

Tulyakov *et al.* [2] have shown that MoCoGAN is able to learn motion generation in latent space and generate controllable video clip from given noise vector. In addition, Tulyakov *et al.* [2] have shown that MoCoGan framework with encoder-decoder architecture is able to predict and generate future video frame sequence from not image difference stream as in MCnet [44], but given only a first frame.

In the perspective of decision-making or imitation learning, the latent space of motion in MoCoGAN framework can be interpreted as latent action space in ILPO [38]. In addition, motion generator in MoCoGAN framework, which consists of RNN, can be comprehended as either latent policy as in ILPO [38] or action planner as in visual foresight frameworks [39, 41, 42, 43]. With this in mind, we will propose an imitation learning framework from unpaired, unaligned observation based on the video prediction framework. We will discuss further in the next chapter 3.

Bibliography

[1] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard, "Recent advances in robot learning from demonstration," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, 2020.

[2] S. Tulyakov, M.-Y. Liu, X. Yang, and J. Kautz, "Mocogan: Decomposing motion and content for video generation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1526–1535, 2018.

[3] Y. Liu, A. Gupta, P. Abbeel, and S. Levine, "Imitation from observation: Learning to imitate behaviors from raw video via context translation," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1118–1125, IEEE, 2018.

[4] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[5] Y. Duan, M. Andrychowicz, B. Stadie, O. J. Ho, J. Schneider, I. Sutskever, P. Abbeel, and W. Zaremba, "One-shot imitation learning," in *Advances in neural information processing systems*, pp. 1087–1098, 2017.

[6] C. Finn, T. Yu, T. Zhang, P. Abbeel, and S. Levine, "One-shot visual imitation learning via meta-learning," *arXiv preprint arXiv:1709.04905*, 2017.

[7] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, "Survey: Robot programming by demonstration," *Handbook of robotics*, vol. 59, no. BOOK_CHAP, 2008.

[8] D. A. Pomerleau, "Efficient training of artificial neural networks for autonomous navigation," *Neural computation*, vol. 3, no. 1, pp. 88–97, 1991.

[9] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635, 2011.

[10] A. Y. Ng, S. J. Russell, *et al.*, "Algorithms for inverse reinforcement learning.," in *Icml*, vol. 1, p. 2, 2000.

[11] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the twenty-first international conference on Machine learning*, p. 1, 2004.

[12] J. Kober and J. Peters, "Learning motor primitives for robotics," in *2009 IEEE International Conference on Robotics and Automation*, pp. 2112–2118, IEEE, 2009.

[13] P. Abbeel, A. Coates, and A. Y. Ng, "Autonomous helicopter aerobatics through apprenticeship learning," *The International Journal of Robotics Research*, vol. 29, no. 13, pp. 1608–1639, 2010.

[14] B. C. Stadie, P. Abbeel, and I. Sutskever, "Third-person imitation learning," *arXiv preprint arXiv:1703.01703*, 2017.

[15] P. Sermanet, C. Lynch, Y. Chebotar, J. Hsu, E. Jang, S. Schaal, S. Levine, and G. Brain, "Time-contrastive networks: Self-supervised learning from video," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1134–1141, IEEE, 2018.

[16] C. L. Nehaniv, K. Dautenhahn, *et al.*, "The correspondence problem," *Imitation in animals and artifacts*, vol. 41, 2002.

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

[18] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, 2016.

[19] F. Torabi, G. Warnell, and P. Stone, "Recent advances in imitation learning from observation," in *IJCAI*, 2019.

[20] 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, pp. 2672–2680, 2014.

[21] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning.," in *Aaai*, vol. 8, pp. 1433–1438, Chicago, IL, USA, 2008.

[22] C. Finn, P. Christiano, P. Abbeel, and S. Levine, "A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models," *arXiv preprint arXiv:1611.03852*, 2016.

[23] S. Levine, Z. Popovic, and V. Koltun, "Nonlinear inverse reinforcement learning with gaussian processes," *Advances in neural information processing systems*, vol. 24, pp. 19–27, 2011.

[24] M. Wulfmeier, P. Ondruska, and I. Posner, “Maximum entropy deep inverse reinforcement learning,” *arXiv preprint arXiv:1507.04888*, 2015.

[25] C. Finn, S. Levine, and P. Abbeel, “Guided cost learning: Deep inverse optimal control via policy optimization,” in *International conference on machine learning*, pp. 49–58, PMLR, 2016.

[26] M. Arjovsky, S. Chintala, and L. Bottou, “Wasserstein gan,” *arXiv preprint arXiv:1701.07875*, 2017.

[27] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. Paul Smolley, “Least squares generative adversarial networks,” in *Proceedings of the IEEE international conference on computer vision*, pp. 2794–2802, 2017.

[28] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell, “Deep domain confusion: Maximizing for domain invariance,” *arXiv preprint arXiv:1412.3474*, 2014.

[29] J. Fu, K. Luo, and S. Levine, “Learning robust rewards with adverserial inverse reinforcement learning,” in *International Conference on Learning Representations*, 2018.

[30] A. Y. Ng, D. Harada, and S. Russell, “Policy invariance under reward transformations: Theory and application to reward shaping,” in *Icml*, vol. 99, pp. 278–287, 1999.

[31] W. Sun, A. Vemula, B. Boots, and D. Bagnell, “Provably efficient imitation learning from observation alone,” in *International Conference on Machine Learning*, pp. 6036–6045, PMLR, 2019.

[32] F. Schroff, D. Kalenichenko, and J. Philbin, “Facenet: A unified embedding for face recognition and clustering,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 815–823, 2015.

[33] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, “Trust region policy optimization,” in *International conference on machine learning*, pp. 1889–1897, PMLR, 2015.

[34] S. Levine, C. Finn, T. Darrell, and P. Abbeel, “End-to-end training of deep visuomotor policies,” *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[35] P. Sermanet, K. Xu, and S. Levine, “Unsupervised perceptual rewards for imitation learning,” *arXiv preprint arXiv:1612.06699*, 2016.

[36] F. Torabi, G. Warnell, and P. Stone, “Behavioral cloning from observation,” in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pp. 4950–4957, 2018.

[37] F. Torabi, G. Warnell, and P. Stone, “Generative adversarial imitation from observation,” *arXiv preprint arXiv:1807.06158*, 2018.

[38] A. Edwards, H. Sahni, Y. Schroecker, and C. Isbell, “Imitating latent policies from observation,” in *International Conference on Machine Learning*, pp. 1755–1763, PMLR, 2019.

[39] C. Finn, I. Goodfellow, and S. Levine, “Unsupervised learning for physical interaction through video prediction,” *arXiv preprint arXiv:1605.07157*, 2016.

[40] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-k. Wong, and W.-c. Woo, "Convolutional lstm network: a machine learning approach for precipitation nowcasting," in *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 1*, pp. 802–810, 2015.

[41] C. Finn and S. Levine, "Deep visual foresight for planning robot motion," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2786–2793, IEEE, 2017.

[42] F. Ebert, C. Finn, A. X. Lee, and S. Levine, "Self-supervised visual planning with temporal skip connections.," in *CoRL*, pp. 344–356, 2017.

[43] A. Xie, F. Ebert, S. Levine, and C. Finn, "Improvisation through physical understanding: Using novel objects as tools with visual foresight," *arXiv preprint arXiv:1904.05538*, 2019.

[44] R. Villegas, J. Yang, S. Hong, X. Lin, and H. Lee, "Decomposing motion and content for natural video sequence prediction," in *5th International Conference on Learning Representations, ICLR 2017*, International Conference on Learning Representations, ICLR, 2017.

[45] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[46] W. Jeon, W. Jeong, K. Son, and H. Yang, "Speckle noise reduction for digital holographic images using multi-scale convolutional neural networks," *Optics letters*, vol. 43, no. 17, pp. 4240–4243, 2018.

[47] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[48] 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*, pp. 5026–5033, IEEE, 2012.