Disturbance-Aware Neuro-Optimal System Control Using Generative Adversarial Control Networks

Kai-Fung Chu, *Student Member, IEEE,* Albert Y.S. Lam, *Senior Member, IEEE,* Chenchen Fan, and Victor O.K. Li, *Fellow, IEEE*

***Abstract*—Disturbance, which is generally unknown to the controller, is unavoidable in real-world systems and it may affect the expected system state and output. Existing control methods, like Robust Model Predictive Control, can produce robust solutions to maintain the system stability. However, these robust methods trade the solution optimality for stability. In this paper, a method called Generative Adversarial Control Networks (GACNs) is proposed to train a controller via demonstrations of the optimal controller. By formulating the optimal control problem in the presence of disturbance, the controller trained by GACNs obtains neuro-optimal solutions without knowing the future disturbance and determines the objective function explicitly. A joint loss, composed of the adversarial loss and the least square loss, is designed to be used in the training of the generator. Experimental results on simulated systems with disturbance show that GACNs outperform other compared control methods.**

***Index Terms*—Optimal control, generative adversarial networks, neural networks, adaptive dynamic programming, approximate dynamic programming, inverse reinforcement learning.**

I. Introduction

DISTURBANCE is unavoidable in real-world systems. As an input to a control system, it may deviate the expected system state and output. With known dynamics and disturbance, control and planning methods, such as model predictive control (MPC) [1], can be used to optimally control the system. However, disturbance is generally unknown to the controller. To address this problem, one usually estimates the disturbance and then performs the action that can compensate the influence of the disturbance [2]. Some researchers use machine learning techniques such as neural networks to estimate the disturbance [3], [4]. However, inaccurate estimation can adversely affect both the optimality and stability and we need to rely on robust control methods, such as Robust MPC [5] to reserve the stability. To do this, we minimize the corresponding objective function by considering the boundary of maximum disturbance. However, there is a tradeoff between stability and optimality when using robust methods. We can enlarge the disturbance boundary for stability but this will sacrifice the optimality.

Stability is usually a basic requirement in system control and it can be achieved by various methods based on optimal control theory [6]. Dynamic programming (DP) is commonly used to solve optimal control problems with optimality preserved. In the presence of disturbance and due to the "curse of dimensionality", DP often cannot produce the optimal solutions in practice. To address these issues, approximate dynamic programming (ADP) [7] was proposed to approximate the dynamic programming solutions using neural networks, which is also known as Adaptive Dynamic Programming [8] or Reinforcement Learning (RL) [9], [10]. Such neural networks are usually trained by their extensive interactions with the environments and they approximate the system dynamics and also the dynamics equations of optimal solution. It aims at determining an optimal action based on the given state by maximizing the long-term reward. For those systems with uncertainty, robust ADP [11], as an extension of ADP, was proposed to deal with the disturbance. In [12], a neuro-optimal control method using ADP was developed to manipulate non-linear systems in the presence of disturbance. It contains a recurrent neural network and a feedforward neural network as an function approximator to reconstruct the system dynamics and disturbance, respectively. In this way, the optimal control problem with disturbance is transformed into a two-player zero-sum control problem. However, the worst scenario due to disturbance is taken into account in the computation resulting in sub-optimal solution in the above methods. In [13], a neural network-based control method was presented to approximate the system uncertainties and disturbances of the discrete-time non-linear system. Another neural network-based decentralized control method was also proposed for a large-scale non-linear system where the system state variables were interconnected between the subsystems [14]. In general, ADP suffers from certain limitations:

- The approximation and control performance heavily depend on the neural network learning technique;
- The reward function, or reinforcement signal, is usually difficult to be defined manually. Consider autonomous driving [15] as an example. Maximizing a reward function that minimizes the travel time is generally not sufficient and there are other measures, such as safety distance to other vehicles and pedestrians, which cannot be well modeled by the manually defined reward function. In [16], an RL problem with safety constraints is transformed to a constrained Markov decision process and solved by a Lyapunov-based approach, but they did not consider disturbance.
- RL may require extensive interactions between the con-

K.F. Chu, A.Y.S. Lam and V.O.K. Li are with the Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong (e-mail: kfchu@eee.hku.hk; ayslam@eee.hku.hk; vli@eee.hku.hk).

C. Fan is with the Department of Mechanical Engineering, The University of Hong Kong, Hong Kong (e-mail: fancc@connect.hku.hk).

troller and the system to learn an approximation of the system dynamics and to maximize the reinforcement signal, but the system may not be available to the controller at the training stage.

To overcome the above limitations, the controller may learn from a given set of expert demonstrations without defining a reward function manually, or constantly interacting with the system. In this case, the objective of the controller becomes performing similarly to the expert demonstrations as much as possible and thus the interaction with the system is unnecessary at the training stage. This is called imitation learning [17] and one of the several possible approaches to solve the problem is by using Inverse Reinforcement Learning (IRL) [18]. IRL is an approach to determine the best reward function that can explain given expert demonstrations. A method based on Generative Adversarial Networks (GANs) [19], called generative adversarial imitation learning (GAIL) [20], was proposed to tackle the imitation learning problem and it can avoid the scaling problem induced by some existing IRL algorithms. GANs were proposed in recent years [19] and it is a framework originally designed for generating artificial images by modeling a two-player minimax game, in which a generator and a discriminator are trained simultaneously to generate and classify the artificial images, respectively. Other than image generation, GANs have other applications like three-dimensional object generation [21] and human pose estimation [22]. In GAIL, control actions are generated by the generator and the discriminator classifies whether its input is generated control actions or expert demonstrations. However, GAIL does not consider the presence of disturbance, leading to direct correspondence between the state and action. In fact, with disturbance, randomness is involved and such direct correspondence may not achieve the best performance.

In this paper, we propose a GAN-based approach, called Generative Adversarial Control Networks (GACNs), to train the controller. To the best of our knowledge, we are the first to model a state feedback control problem in the presence of unknown disturbance to function approximation problem solvable by the proposed GACNs. Historical system states and expert demonstrations are used as the training data for GACNs, in which a generator and a discriminator are trained simultaneously to determine an estimate of the objective function and the neuro-optimal control actions without knowing the future disturbance. The convergence analysis is provided to show the convergent property of the approach. We also find that the control performance is improved by using a joint loss function, instead of the single adversarial loss function proposed in the original GAN model, in our experiments. The controller trained by GACNs is shown to be robust to disturbance and outperform other compared control methods.

The main contributions of this paper can be summarized as follows:

1. We re-formulate the optimal control problem with disturbance as an IRL problem solvable by GACNs;
2. We propose a variant method based on GANs called GACNs that is specifically designed to solve the problem;
3. A convergence analysis is given to investigate the convergence of the proposed method;
4. To the best of our knowledge, we are the first to handle the optimal control with disturbance using the adversarial training approach; and
5. We perform extensive experiments in a discrete time system and cartpole system using the proposed method and other compared methods.

The rest of this paper is organized as follows. Section II presents the background information of Optimal control, IRL and GANs. Section III defines the system model and the system control problem with disturbance. We present our proposed control approach in Section IV and experiments are illustrated in Section V. Finally, Section VI concludes this paper.

II. Background

In this section, IRL and GANs are reviewed to provide background information for subsequent sections.

A. Inverse Reinforcement Learning

According to [23], IRL is characterized informally as a problem that determines the reward function being optimized by giving measurements of an agent's behavior over time, measurements of the sensory inputs to that agent, and the model of the environment. In other words, IRL aims to determine an estimated reward function $\hat{r}(\cdot)$, which can explain the observed behavior π^* produced from the true reward function $r(\cdot)$ [18]. The estimated reward function $\hat{r}(\cdot)$ is maximized to obtain the relevant actions by RL algorithms. The IRL problem can be modeled as follows:

$$\mathbb{E}\Big[\sum_{\tau=t}^{t+T} r^*(s(\tau), a(\tau))|\pi^*\Big] \geq \mathbb{E}\Big[\sum_{\tau=t}^{t+T} r^*(s(\tau), a(\tau))|\pi\Big], \forall\pi,$$

where $\pi = \{a(t), \ldots, a(t+T)\}$, $a(\tau)$ is the control action at time τ, $s(\tau)$ is the system state at time τ, and T is the latest allowed time in the time horizon [24].

1) Maximum Entropy IRL: One special form of IRL is maximum entropy IRL (MaxEnt-IRL) [25], in which Boltzmann distribution is used to model the expert demonstrations given as follows:

$$p_\theta(v) = \frac{1}{Z} e^{-F_\theta(v)}, \tag{1}$$

where $v = \{s(t), a(t), \ldots, s(t+T), a(t+T)\}$ is a trajectory, $F_\theta(v) = \sum_\tau f_\theta(s(\tau), a(\tau))$ is a learned cost (objective) function parameterized by the adjustable parameters θ with the cost function $f_\theta(\cdot)$ for each time slot, and Z is the integral of $e^{-F_\theta(v)}$ over all trajectories [25]. $p_\theta(v)$ is optimized by adjusting parameters θ to produce the maximum probability $p^*_\theta(v)$ with the given expert trajectory v. With the optimized parameters θ^*, $F^*_\theta(v)$ accounts for the expert trajectory and may generate optimal trajectories in the new environment. In [25], Z is estimated by dynamic programming.

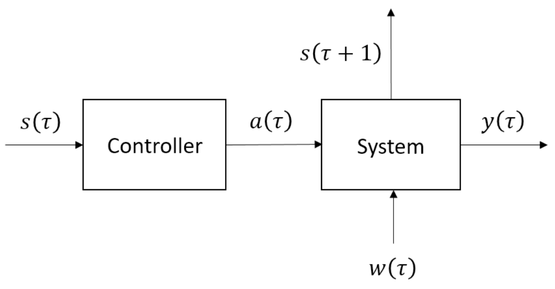


Fig. 1. System block diagram

B. Generative Adversarial Networks

In this framework, a generative model G and a discriminative model D are trained simultaneously. The goal of G is to generate expected outputs that minimize the divergence between the distributions of generated outputs $q(x)$ and the real data $p(x)$ such that the input of D is misclassified. On the other hand, D classifies whether its inputs are real data or generated outputs from G. GANs correspond to a two-player minimax game with the value function $V(G, D)$ as:

$$\min_G \max_D V(G,D) = \mathbb{E}_{x\sim p}[\log D(x)] + \mathbb{E}_{x\sim G}[\log(1-D(x))]. \tag{2}$$

G takes random noise z as input and outputs a sample $x \sim G(z)$. We train G such that the distribution of $G(z)$ is mapped to the data space $p(x)$. The loss function of the generator $\mathcal{L}_{\text{gen}}(G)$ is given as:[1]

$$\mathcal{L}_{\text{gen}}(G) = \mathbb{E}_{x\sim G}[-\log D(x)] + \mathbb{E}_{x\sim G}[\log(1-D(x))]. \tag{3}$$

D takes a sample x as input and outputs $D(x)$, where $D(x)$ represents the probability that x is sampled from $p(x)$. The loss function of the discriminator $\mathcal{L}_{\text{dis}}(D)$ is:

$$\mathcal{L}_{\text{dis}}(D) = \mathbb{E}_{x\sim p}[-\log D(x)] + \mathbb{E}_{x\sim G}[-\log(1-D(x))]. \tag{4}$$

III. System Model

Figure 1 shows a discrete time system with disturbance and the corresponding controller. At discrete time $\tau \in \mathbb{N}$, the controller takes an n-dimensional system state $s(\tau) \in \mathbb{R}^n$ as input and outputs an m-dimensional control action $a(\tau) \in \mathbb{R}^m$ to the system. The controller adjusts the control action based on $s(\tau)$ such that an objective function $f(\cdot) \geq 0$ is minimized. Let $w(\tau) \in \mathbb{R}^p$ be the disturbance in p dimensions. We model the system dynamics as

$$\begin{cases} s(\tau+1) &= g(s(\tau), a(\tau), w(\tau)), \\ y(\tau) &= h(s(\tau), a(\tau)), \end{cases} \tag{5}$$

where $g(\cdot)$ and $h(\cdot)$ are the corresponding system dynamic functions, and $y(\tau)$ is the system output.

We design the optimal controller by considering the following optimization problem, in which we plan the actions for the next T time-steps at each time t:

[1]The first term is a well-known alternative called "the $-logD$ trick" [26] to deal with the gradient vanishing problem in the original GAN loss function. This function provides a better training signal compared to the original one. Therefore, it is common to consider the sum of these two loss functions as the loss function of the generator as in [27].

Problem 1 (Optimal Control Problem):

$$\min_{a(\tau)} \quad \sum_{\tau=t}^{t+T} f\left(s(\tau), a(\tau)\right) \tag{6a}$$
$$\text{s.t.} \quad s(\tau+1) = g(s(\tau), a(\tau), w(\tau)), \tau = t, \ldots, t+T-1 \tag{6b}$$
$$y(\tau) = h(s(\tau), a(\tau)), \tau = t, \ldots, t+T \tag{6c}$$
$$s(\tau) \in \mathcal{S}, a(\tau) \in \mathcal{A}, w(\tau) \in \mathcal{W}, \tau = t, \ldots, t+T, \tag{6d}$$

where $\mathcal{S}$, $\mathcal{A}$, and $\mathcal{W}$ are the sets of all possible states, actions, and disturbances, respectively. Problem 1 can only be solved when the disturbance is given. However, the disturbance is generally unknown in real-world scenarios. To overcome this, one can adopt robust optimization, which is a minimax approach that minimizes the objective by considering the maximum disturbance boundary. The robust optimal control problem can be defined as:

Problem 2 (Robust Control Problem):

$$\min_{a(\tau)\in\mathcal{A}} \max_{w(\tau)\in\mathcal{W}} \sum_{\tau=t}^{t+T} f\left(s(\tau), a(\tau)\right) \tag{7a}$$
$$\text{s.t.} \quad (6b) - (6d). \tag{7b}$$

By solving the problem within the boundary of maximum disturbance, the solution is robust in the sense that the system is stable for any disturbance $w(\tau) \in \mathcal{W}$. However, such robustness sacrifices optimality. We can see that there are certain limitations when solving problems 1 or 2. Problem 1 cannot be solved without known disturbance while solving problem 2 suffer from sub-optimal solution. To preserve the optimality with unknown disturbance, we first reformulate the optimal control problem to the function approximation problem as follows:

Problem 3 (Function Approximation Problem):

$$\min_{\hat{a}(\tau)} \quad \sum_{\tau=t}^{t+T} \hat{f}(\hat{s}(\tau), \hat{a}(\tau)) \tag{8a}$$
$$\text{s.t.} \quad \hat{s}(\tau+1) = \hat{g}(\hat{s}(\tau), \hat{a}(\tau)), \tau = t, \ldots, t+T-1 \tag{8b}$$
$$\hat{y}(\tau) = h(\hat{s}(\tau), \hat{a}(\tau)), \tau = t, \ldots, t+T \tag{8c}$$
$$\hat{s}(\tau) \in \mathcal{S}, \hat{a}(\tau) \in \mathcal{A}, \tau = t, \ldots, t+T, \tag{8d}$$

where $\hat{g}(\hat{s}(\tau), \hat{a}(\tau))$ is an approximation of $g(s(\tau), a(\tau), w(\tau))$ and $\hat{y}(\tau)$ is an estimate of $y(\tau)$. In this formulation, disturbance is unnecessary with the approximated function given. We aim to find the equivalent functions $\hat{f}(\cdot)$ and $\hat{g}(\cdot)$ such that $\hat{f}(\hat{s}(\tau), \hat{a}(\tau)) = f(s(\tau), a(\tau))$, which allows us to address Problem 3.

Instead of solving Problem 3 directly, we define a trainable function $G(s(\tau))$ to generate the control action $a(\tau)$. The optimal G^* is given as:

$$G^*(s(\tau)) = \hat{a}(\tau)^* = \arg\min_{\hat{a}(\tau)\in\mathcal{A}} \sum_{\tau=t}^{t+T} \hat{f}(\hat{s}(\tau), \hat{a}(\tau)), \tag{9}$$

where $\hat{s}(\tau+1) = \hat{g}(\hat{s}(\tau), \hat{a}(\tau))$. The training approach to determine G^* is presented in the next section.

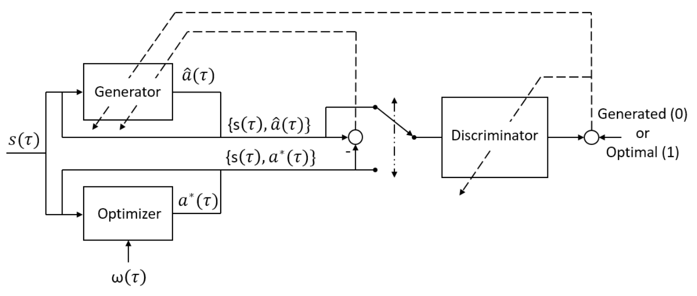


Fig. 2. Model of GACNs

IV. Generative Adversarial Control Networks

In this section, we design a GAN-based control method with IRL to address Problem 3 and explain why the designed control method is solving the given problem. We also provide a convergence analysis for the method.

As discussed in Section II-A, IRL aims to determine an estimated reward function $\hat{r}(\cdot)$ with given $\{s(\tau),a(\tau)\},\tau=t,\dots,t+T$. Consider that the optimal state-action pair $\{s(\tau),a^*(\tau)\}$ are given in Problem 3 and we have $\hat{r}(\tau)=\hat{f}(\hat{s}(\tau),a(\tau))$. We can determine the estimated system dynamics $\hat{g}(\cdot)$ and objective functions $\hat{f}(\cdot)$ by using IRL. Then we can solve Problem 3 with the estimated functions $\hat{g}(\cdot)$ and $\hat{f}(\cdot)$ using MPC. However, IRL algorithms are often computational intensive [28] which lower its applicability in this case. In recent years, GANs have shown promising results in solving various problems [19]. As estimating the reward function in MaxEnt-IRL is shown to be equivalent to optimizing both the generator and discriminator in GANs [27], we propose GACNs to jointly estimate the objective function and to obtain the neuro-optimal control action simultaneously.

A. Connection between MaxEnt-IRL and GANs

For a given G with $q(x)$ as the distribution of the generated data, the optimal discriminator $D^*_G(x)$ is given as [19]:

$$D^*_G(x)=\frac{p(x)}{p(x)+q(x)}, \tag{10}$$

where $p(x)$ is real data distribution. Instead of estimating Eq. (10) directly, the discriminator D_θ with parameters θ can be considered as an estimate of $p(x)$ with a known $q(x)$, i.e.,

$$D_\theta(x)=\frac{\tilde{p}_\theta(x)}{\tilde{p}_\theta(x)+q(x)}. \tag{11}$$

From the model of MaxEnt-IRL as shown in Eq.(1), the estimate of real data distribution $\tilde{p}_\theta(v)$ is substituted by the Boltzmann distribution in order to relate MaxEnt-IRL with GANs. The optimal trajectory has the largest likelihood in MaxEnt-IRL. For the loss function of the discriminator, it should be the largest when the trajectory is optimal. Therefore, the substitution is the necessary condition of the loss function of the discriminator and thus we can always substitute $\tilde{p}_\theta(x)$ by the Boltzmann distribution. The discriminator becomes:

$$D_\theta(v)=\frac{\frac{1}{Z}e^{-F_\theta(v)}}{\frac{1}{Z}e^{-F_\theta(v)}+q(v)}=\frac{\frac{1}{Z}e^{-F_\theta(v)}}{\tilde{\zeta}(v)}, \tag{12}$$

where $\tilde{\zeta}(v)$ is estimate of $\frac{1}{Z}e^{-F_\theta(v)}+q(v)$.

By substituting Eq. (12) to Eq. (4), the loss function of the discriminator can be derived as follows:

$$\begin{aligned}\mathcal{L}_{\text{dis}}(D_\theta)&=\mathbb{E}_{v\sim p}[-\log D_\theta(v)]+\mathbb{E}_{v\sim q}[-\log(1-D_\theta(v))]\\&=\mathbb{E}_{v\sim p}\left[-\log\left(\frac{\frac{1}{Z}e^{-F_\theta(v)}}{\tilde{\zeta}(v)}\right)\right]+\\&+\mathbb{E}_{v\sim q}\left[-\log\left(\frac{q(v)}{\tilde{\zeta}(v)}\right)\right]\\&=\log Z+\mathbb{E}_{v\sim p}[F_\theta(v)]-\mathbb{E}_{v\sim q}[\log q(v)]+\\&+2\mathbb{E}_{v\sim\zeta}\left[\log\tilde{\zeta}(v)\right].\end{aligned} \tag{13}$$

The optimal point of the discriminator is its derivative with respect to θ:

$$\partial_\theta\mathcal{L}_{\text{dis}}(D_\theta)=\mathbb{E}_{v\sim p}[\partial_\theta F_\theta(v)]-\mathbb{E}_{v\sim\zeta}\left[\frac{\frac{1}{Z}e^{-F_\theta(v)}\partial_\theta F_\theta(v)}{\tilde{\zeta}(v)}\right]. \tag{14}$$

On the other hand, the log-likelihood objective of MaxEnt-IRL is:

$$\begin{aligned}\mathcal{L}_{\text{cost}}(\theta)&=\mathbb{E}_{v\sim p}[-\log p_\theta(v)]\\&=\mathbb{E}_{v\sim p}[F_\theta(v)]+\log Z\\&=\mathbb{E}_{v\sim p}[F_\theta(v)]+\log\mathbb{E}_{v\sim\zeta}\left[\frac{e^{-F_\theta(v)}}{\tilde{\zeta}(v)}\right],\end{aligned} \tag{15}$$

where Z is estimated from the derivative of the discriminator's loss with respect to Z, which is:

$$Z=\left\{\mathbb{E}_{v\sim\zeta}\left[\frac{e^{-F_\theta(v)}}{\tilde{\zeta}(v)}\right]\Bigg|_{\partial_Z\mathcal{L}_{\text{dis}}(D_\theta)=0}\right\}. \tag{16}$$

The derivative with respect to θ of the log-likelihood objective of MaxEnt-IRL is given as:

$$\partial_\theta\mathcal{L}_{\text{cost}}(\theta)=\mathbb{E}_{v\sim p}[\partial_\theta F_\theta(v)]-\mathbb{E}_{v\sim\zeta}\left[\frac{\frac{1}{Z}e^{-F_\theta(v)}\partial_\theta F_\theta(v)}{\tilde{\zeta}(v)}\right]. \tag{17}$$

We can see that the optimal points of both the discriminator and MaxEnt-IRL log-likelihood are the same by comparing Eqs. (14) and (17). Hence, GANs can be considered as a sample-based algorithm for MaxEnt-IRL and the details can be found in [27].

B. Model architecture

Here a GAN-based architecture is presented to solve Problem 3. In this model, there are three components, namely, optimizer O, generator G and discriminator D. Figure 2 illustrates the detailed architecture of GACNs, where the dotted lines indicate the parameters update by the corresponding loss functions (Eq. (20) for G and Eq. (21) for D). Suppose that we have the historical training dataset of initial state $s(\tau)$ and disturbance $w(\tau)$. To generate a set of corresponding optimal control action $a^*(\tau)$, O can solve Problem 1. The states $s(\tau)$ and actions $a^*(\tau)$ form a labeled dataset $\{s(\tau),a^*(\tau)\}$ for training. The generator aims to generate control actions that stably and optimally control the system. It takes $s(\tau)$ as input and outputs a generated control action $\hat{a}(\tau)$. The inputs of D

are randomly sampled between the optimal state-action pairs $\{s(\tau), a^*(\tau)\}$ and generated state-action pairs $\{s(\tau), \hat{a}(\tau)\}$. D classifies whether its input is an optimal state-action pair from O or a generated state-action pair from G. If generated state-action pair $\{s(\tau), \hat{a}(\tau)\}$ is given, D is expected to output a zero. When optimal state-action pair $\{s(\tau), a^*(\tau)\}$ is given, D is expected to output a one. It is an iterative process where the parameters of G and D are iteratively updated using the loss function given in Eq. (20) and (21). As discussed in [19], training of the GAN-based architecture facilitates the approximation of G to O. In other words, with sufficient training, the output of G is close to that of O and D cannot distinguish its input from G or O.

Traditional GANs aim to generate artificial images and thus the generator and discriminator are usually constructed as convolutional neural networks [29]. However, GACNs generate control actions from the given system states. For simplicity and generalization purposes, we adopt multilayer perceptrons to form the architecture for both the generator and the discriminator in our design. Other kinds of neural networks can also be considered, depending on the actual structure of the state and control action.

The generator is trained with two loss functions, i.e., an adversarial loss and a least square loss. The least square loss is responsible for constructing the average control action from the distribution while the adversarial loss refines the control action by selecting a particular trajectory from the distribution. The objective is to map the input state to output control action using the generator function. The optimal control action from O can be used as the supervisory signal for the function. We minimize the distance between the generated and the optimal using a least square approach, in which the least square loss is given by:

$$\mathcal{L}_{\text{LS}} = \mathbb{E}[(\hat{a}(\tau) - a^*(\tau))^2]. \tag{18}$$

However, the least square loss tends to produce an average value from the distribution of possible control actions, which may degrade the control performance. Therefore, the adversarial loss, as

$$\mathcal{L}_{\text{adv}} = \mathbb{E}_{x\sim G}[-\log D(x) + \log(1 - D(x))], \tag{19}$$

is also introduced to refine the output by choosing a particular trajectory from the distribution that is more similar to the real output from the optimal controller. As discussed in Section IV-A, optimizing the adversarial loss is equivalent to estimating the objective function, which allows the generator to determine the control actions. This leads to generalization of the objective of optimal controller rather than simple behavioral cloning of the optimal controller. G tries to generate control actions as close as possible to the control actions from O such that D is misclassified. Therefore, the logistic likelihood between the discriminator output and the correct labels should be minimized.

By combining $\mathcal{L}_{\text{LS}}$ and $\mathcal{L}_{\text{adv}}$, the joint loss function of the generator is:

$$\mathcal{L}_{\text{gen}}(G) = \lambda_{\text{adv}}\mathcal{L}_{\text{adv}} + \lambda_{\text{LS}}\mathcal{L}_{\text{LS}}, \tag{20}$$

where λ_{adv} and λ_{LS} are the weights of the corresponding losses. A similar joint loss function was used for image inpainting [30], which showed that the joint loss function enhances performance over a single loss function. To the best of our knowledge, we are the first to embrace the joint adversarial-least square loss for system control.

We train D to better determine whether the input state-action pair is from O or G. Hence, we minimize the negative logistic likelihood between the discriminator outputs and the correct labels. The loss function of the discriminator is:

$$\mathcal{L}_{\text{dis}}(D) = \mathbb{E}_{x\sim p}[-\log D(x)] + \mathbb{E}_{x\sim G}[-\log(1 - D(x))]. \tag{21}$$

C. Stability and Convergence Analysis

We give the stability and convergence analysis of the proposed GACNs approach below:

1) Stability analysis: The asymptotic stability of the system under the designed control law is investigated first. The optimal control law of Problem 1 is given as

$$a^* = \arg\min_{a(\tau)\in\mathcal{A}} \sum_{\tau=t}^{t+T} f(s(\tau), a(\tau)) \tag{22}$$

By denoting $L(s(\tau)) = \min_{a(\tau)\in\mathcal{A}} \sum_{\tau=t}^{t+T} f(s(\tau), a(\tau))$ and according to the Bellman's principle of optimality, we have

$$\begin{aligned} L(s(\tau+1)) &= L(g(s(\tau), a(\tau), w(\tau)) \\ &= \min_{\{a(\tau+1),\cdots,a(t+T)\}\in\mathcal{A}} \sum_{\tau=t+1}^{t+T} f(s(\tau), a(\tau)). \end{aligned} \tag{23}$$

Moreover, $L(s(\tau))$ can be rewritten as:

$$\begin{aligned} L(s(\tau)) \;=\; & \min_{a(t)\in\mathcal{A}} f(s(t), a(t)) + \min_{\{a(\tau+1),\cdots,a(t+T)\}\in\mathcal{A}} \\ & \sum_{\tau=t+1}^{t+T} f(s(\tau), a(\tau)). \end{aligned} \tag{24}$$

Taking $L(s(\tau))$ as a Lyapunov function candidate, we have

$$\begin{aligned} \Delta L(\tau) &= L(s(\tau+1)) - L(s(\tau)) \\ &= -\min_{a(t)\in\mathcal{A}} f(s(t), a(t)) < 0, \end{aligned}$$

as the cost function $f(s(t), a(t)) > 0$, $\forall s \neq 0$.

The above analysis implies that for Problem 1, under the optimal control law a^*, the system is asymptotically stable. Next, we just need to show that $\hat{a}(\tau)$ will converge to $a^*(\tau)$. Therefore, in the following, we will investigate the convergence analysis of the detailed implementation process of GACNs.

2) Convergence analysis of GACNs:

a) Generator: The estimated control action $\hat{a}(\tau)$ is obtained from the output of the generator. Then the approximate control action can be given as

$$\hat{a}(\tau) = G(s(\tau)). \tag{25}$$

To update the generator $G(s(\tau))$, we need to minimize the joint loss function given as (20) to get the update law of the approximate control $\hat{a}(\tau)$. It is obvious that (18) is a convex function since $\frac{\partial^2 \mathcal{L}_{LS}}{\partial G^2} > 0$. For training errors using least square regression, the convergence was proven in [31]. For the adversarial loss function (19), we need to investigate whether it is convex for fixed D, the following equations hold:

$$\frac{\partial \mathcal{L}_{adv}}{\partial G} = \mathbb{E}_{x\sim G}\left[-\frac{2}{D(x)} - \frac{2}{(1-D(x))}\right],$$
$$\frac{\partial^2 \mathcal{L}_{adv}}{\partial G^2} = \mathbb{E}_{x\sim G}\left[\frac{2}{D^2(x)} + \frac{2}{(1-D(x))^2}\right] > 0.$$

As $\frac{\partial^2 \mathcal{L}_{adv}}{\partial G^2} > 0$, the adversarial loss function (19) is convex.

The joint loss function of the generator is also convex due to the convexity-preserving characteristic of the non-negative weighted sum of two convex functions. Therefore, the estimate control law $\hat{a}(\tau)$ will converge to $a^*(\tau)$ according the gradient-descent-based direction of the joint loss function.

b) Discriminator: The loss function of the discriminator is chosen as (21). For fixed G, the optimal discriminator D is

$$D^*(x) = \frac{p(x)}{p(x)+G(x)}. \tag{26}$$

From Proposition 2 in [19], if G and D have enough capacity, the discriminator is allowed to reach its optimum given the generator. Generated data $G(x)$ are updated based on the convex loss function of the generator according to Section IV-C2a to minimize the criterion

$$\mathbb{E}_{x\sim p}[-\log D^*(x)] + \mathbb{E}_{x\sim G}[-\log(1-D^*(x))]. \tag{27}$$

Then the generated data $G(x)$ converge to p.

During the training process, we need to minimize the loss function of the generator (20) and the discriminator (21) simultaneously. Based on the above analysis, the objective function is convex in G. In other words, when D is optimal, a gradient descent update for G at the optimal D is computed. The minimum of the objective function with optimal D is convex in G with a unique global optimum. therefore when G is updated with sufficiently small step size, G will converge to p.

In summary, $\hat{a}(\tau)$ will converge to $a^*(\tau)$ by training GACNs. By regarding the networks of the generator and the discriminator as action network and critic network, respectively [32], the estimation error $|\hat{a}(\tau) - a^*(\tau)|$ is uniformly ultimately bounded. Combining the asymptotic stability with control action $a^*(\tau)$ and the uniformly ultimately bounded convergence, the uniformly ultimately bounded stability of the approximate system (Eq. (5)) can be obtained.

V. Experiments

We evaluate our proposed GACNs on a discrete time linear system and on the OpenAI Gym "CartPole" environment [33]. As in [34], 25 future steps are evaluated in both test cases. For both test scenarios, the disturbance is represented by the sum of uniform random noise and a sine function, as follows:

$$w(\tau) = \alpha\mathcal{W}(\tau) + \beta\sin(2\pi\frac{(\tau-t)}{T}), \tag{28}$$

where $\mathcal{W}(\tau) \in [-1,1]$ and $\alpha+\beta=1$. α and β are the weights for simulating different levels of randomness of the disturbance. For example, the disturbance is completely random if $\alpha=1$ and $\beta=0$. If $\alpha=0$ and $\beta=1$, it is a pure sine function without randomness.

The following control approaches are compared with GACNs:

1. Perfect MPC: The controller has perfect information of the future disturbance, which allows us to compute the global optimal control action using MPC.
2. Robust MPC (Open-loop minimax MPC) [35]: The objective function is minimized by considering the maximum disturbance boundary as presented in Problem 2.
3. Robust MPC (Closed-loop minimax MPC): This method was proposed in [34] to solve Problem 2. The difference between this and the open-loop is that the future control inputs are parameterized as affine functions of past disturbances but the open-loop does not.
4. Stochastic control (certainty-equivalent) [36]: This method assumes that the distribution of the disturbance is known and thus the unknown disturbance variable is substituted by the expected value.
5. Future prediction MPC: Disturbance is predicted with different prediction accuracy. The predicted disturbance is incorporated in the computation of Problem 1 using MPC.
6. Supervised learning: A multilayer perceptron is trained to minimize the least square loss (Eq. (18)) between the outputs and expert demonstrations.
7. GANs: A GAN-based approach is trained to minimize the adversarial losses of the generator (Eq. (19)) and the discriminator (Eq. (21)).
8. LSGANs [37]: A variant of GANs with least square loss of the generator and the discriminator where $\mathcal{L}_{\text{gen}}(G) = \mathbb{E}_{x\sim G}[(D(x)-1)^2]$ and $\mathcal{L}_{\text{dis}}(D) = \mathbb{E}_{x\sim p}[(D(x)-1)^2] + \mathbb{E}_{x\sim G}[(D(x)^2)]$.
9. WGANs [26]: A variant of GANs with four modifications to GANs: (i) Remove the sigmoid function at the last layer of the discrimintor network; (ii) Use $\mathcal{L}_{\text{gen}}(G) = \mathbb{E}_{x\sim G}[-D(x)]$ and $\mathcal{L}_{\text{dis}}(D) = \mathbb{E}_{x\sim p}[-D(x)] + \mathbb{E}_{x\sim G}[D(x)]$; (iii) Clip the weight to a fixed box (e.g., [-0.01, 0.01]) after each gradient update; (iv) Use RMSProp [38] as the training algorithm.
10. WGAN-GP [39]: In addition to WGANs, a gradient penalty is added to the loss function instead of the weight clipping method in WGANs.

GACNs are different from the above GAN-based approach in the sense that GACNs are trained to minimize the joint loss functions of the generator (Eq. 20) and the loss function of the discriminator (Eq. 21). The generator and discriminator of GAN are two-layer perceptrons. The hidden layer contains 64 hidden units. We purposely avoid using deep neural networks and other complicated techniques. The reason is that different specific control problems may have different structures of state and action, and the neural network architecture needs to be fine-tuned accordingly. But such fine-tuning process is out of the scope of this paper. We try to test the proposed framework instead of using deep neural networks to boost its performance. For the training parameters, we use Adam [40] with the learning rate of 0.001, batch size of 32 and trained for 100,000 iterations and select the best trained model based on the validation set for testing.

In particular, for future prediction MPC, we define ϕ as the accuracy of the predicted value to simulate certain percentage of the prediction error in the prediction. The percentage error is defined using Symmetric Mean Absolute Percent Error (SMAPE) and we have

$$\phi = 1 - \text{SMAPE}(w, \hat{w}) = 1 - \frac{|w - \hat{w}|}{|w| + |\hat{w}|}, \tag{29}$$

where $\hat{w}$ is the predicted value of w for a pre-defined accuracy ϕ. Given w and ϕ, we can calculate $\hat{w}$ based on Eq. (29), which allows us to simulate the prediction of w with different prediction accuracy ϕ. The MPC problems are modeled and solved using YALMIP [41] and GUROBI [42]. Supervised learning, GAN, and other GAN-based approaches are implemented using Tensorflow [43] and Python.

A. Test 1: Discrete time system

We follow [34] to model the discrete time linear system as follows:[2]

$$\begin{cases} f(s(\tau), a(\tau)) & = |y(\tau) - 1|, \\ g(s(\tau), a(\tau), w(\tau)) & = As(\tau) + Ba(\tau) + Ew(\tau), \\ h(s(\tau), a(\tau)) & = Cs(\tau) + Da(\tau), \end{cases}$$

where

$$A = \begin{pmatrix} 2.938 & -0.7345 & 0.25 \\ 4 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}, B = \begin{pmatrix} 0.25 \\ 0 \\ 0 \end{pmatrix},$$
$$C = \begin{pmatrix} -0.2072 & 0.04141 & 0.07256 \end{pmatrix}, D = 0,$$
$$E = \begin{pmatrix} 0.0625 \\ 0 \\ 0 \end{pmatrix}.$$

From the objective function $f(s(\tau), a(\tau))$, we can see that it is an optimal tracking problem controlling the output to track a desired reference value of 1. Table I shows the average objective function value of different control approaches and different levels of disturbance. The lower the objective function value, the better the control approach since Problem 3 is a minimization program. The control action of perfect MPC is optimal since it is given all future information which is impractical in real situations. It mainly serves as a benchmark to show the global optimal value. In general, the larger the value of α, the larger the objective function value. This is because a larger α increases the randomness level of the disturbance and thus it makes the system harder to control. Among the five tested GAN-based models, only GAN and GACN smoothly decrease the values when α decrease while other three models have abnormal values. It may due to the reason that the networks stuck at sub-optimal during the training. Overall, on all the tested randomness levels of disturbance, GACNs perform the best among other practical control approaches in general.

We purposely avoid using deep neural networks and other complicated architecture, and use a simple two-layer perceptron in the joint adversarial-least square loss GAN for the experiment to test the performance of this framework. MPC is an important baseline as it is widely used in the industry over the past decades. The simulation results show that this framework outperforms MPC in the presence of disturbance even with a simple shallow neural network.

[2]Discretized version of the system is $\frac{0.25^2(-2s+1)}{s(s^2+0.25^2)}$.

Fig. 3 shows the validation loss during training for the five GAN-based models of different values of α and β. From the figure, we can see that GANs, WGANs, WGAN-GP and GACNs can converge after about 30,000 to 50,000 iterations while LSGANs oscillates during the training. In particular, WGANs and WGAN-GP converge to a relatively high level for $\alpha = 0.1$ and $\beta = 0.9$. Only GANs and our proposed joint loss GACNs can converge to a low loss value in all tested cases, which shows that these two methods are more suitable for this test case.

Fig. 4 shows the output and control action of a discrete time system with different disturbance levels. Since $f(s(\tau), a(\tau)) = |y(\tau) - 1|$, minimizing this objective function means regulating the output y to 1. In general, the trained model can regulate the output to 1 at about 5 to 10 time steps and maintain a similar value in later time steps. For smaller α (smaller randomness level of disturbance), the magnitude of control action is large at the beginning and becomes smaller at later time steps. This is because the control action has to regulate the output to 1 at the beginning. After reaching the steady state at later time steps, the control action is only required to compensate its effect on the system. For larger α (smaller randomness level of disturbance), the control action oscillates to compensate the effect of disturbance such that the output is steady. That means the model already has the ability to predict the disturbance and thus maintain the steady state outputs.

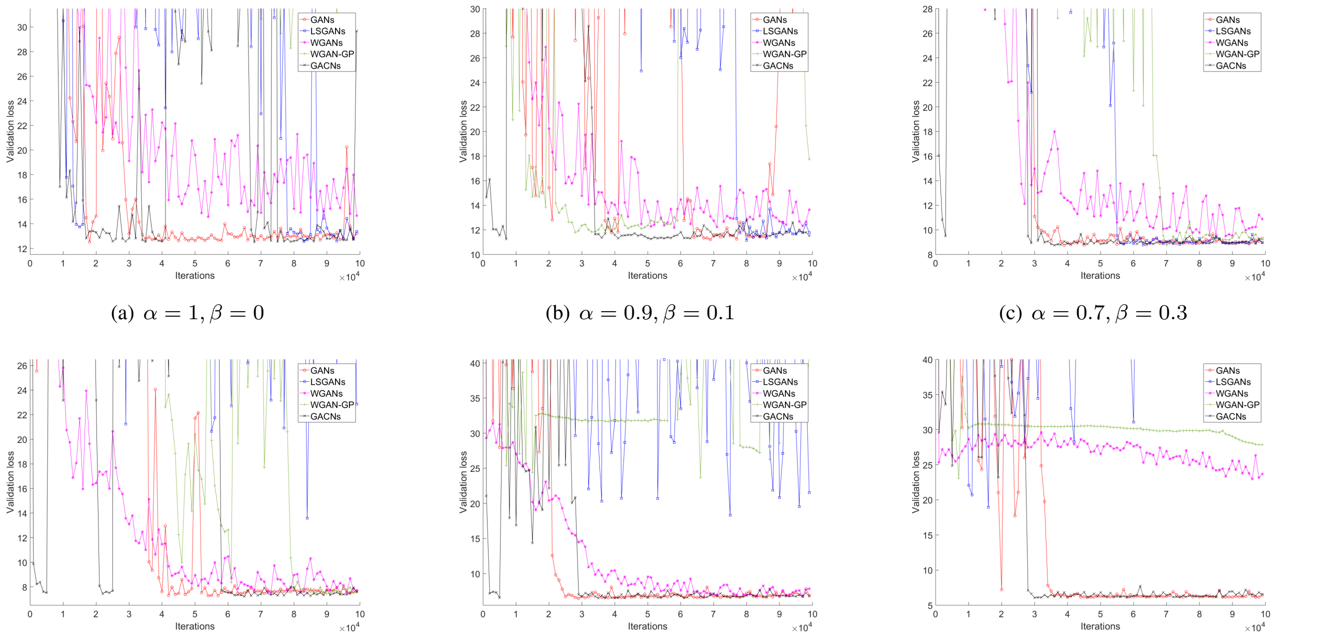
B. Test 2: OpenAI Gym "CartPole" environment

The environment "CartPole-v1" is used in the experiments. In this environment, a pole stands upright on a cart with a random small state value at the beginning. The state is composed of 4 elements: cart position, cart velocity, pole angle, and pole velocity at the tip. The controller tries to prevent the pole from falling by applying a force to the cart. The objective function of the controller is to maintain the pole standing upright by minimizing its vertical angle, i.e., the third element of the state. The parameter settings, such as for the mass of cart and pole, follow the default values given in OpenAI Gym. Two modifications are made to fit our problem: (1) An additional random horizontal force $w \in [-0.5, +0.5]$ is applied on the pole at each time-step to simulate the disturbance; (2) Instead of $a \in \{-10, +10\}$, the system is controlled by applying a force $a \in [-10, +10]$ to the cart, which is a more general control action to verify the controller performance when compared to the original binary control input.

Table II shows the average objective function values (pole angle in radian) of different control approaches and disturbance. Since the controller aims to minimize the pole angle, the lower the value, the better the control approach. The values in Table II may look similar and small since the value

TABLE I
DISCRETE TIME SYSTEM RESULTS.

Methods	Average objective function value with different disturbance levels					
	$\alpha = 1, \beta = 0$	$\alpha = 0.9, \beta = 0.1$	$\alpha = 0.7, \beta = 0.3$	$\alpha = 0.5, \beta = 0.5$	$\alpha = 0.3, \beta = 0.7$	$\alpha = 0.1, \beta = 0.9$
Perfect MPC	7.1855	6.8705	6.3136	5.9818	5.8305	5.7269
Robust MPC (Open-loop)	30.1059	26.0293	18.8557	12.3883	9.7572	9.4783
Robust MPC (Closed-loop)	28.7224	24.4317	16.9499	10.3468	7.8011	7.5684
Stochastic	19.4087	15.6614	9.7708	7.9950	7.8632	7.9755
Future prediction MPC	$\phi = 1\%$	$\phi = 10\%$	$\phi = 30\%$	$\phi = 50\%$	$\phi = 70\%$	$\phi = 90\%$
	45.3642	39.6001	15.9469	8.3091	7.2852	6.6823
Supervised learning	14.4433	13.0550	9.9841	7.7452	7.0111	6.8159
GANs	**12.3042**	11.0794	**8.6491**	7.2400	6.5087	6.0672
LSGANs	12.3410	11.0613	8.6649	13.6291	14.0102	15.6013
WGANs	13.8431	11.6170	9.2477	7.4890	6.8097	22.9780
WGAN-GP	27.5566	11.3695	8.6814	7.3004	17.6359	22.4305
GACNs	**12.3042**	**11.0572**	8.6640	**7.1919**	**6.5069**	**6.0457**



(a) $\alpha = 1, \beta = 0$

(b) $\alpha = 0.9, \beta = 0.1$

(c) $\alpha = 0.7, \beta = 0.3$

(d) $\alpha = 0.5, \beta = 0.5$

(e) $\alpha = 0.3, \beta = 0.7$

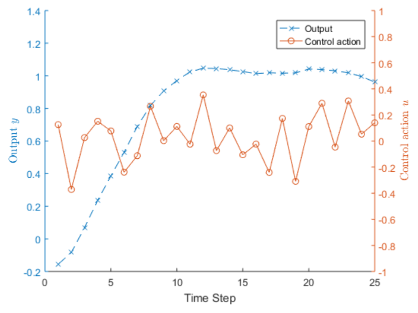
(f) $\alpha = 0.1, \beta = 0.9$

Fig. 3. Discrete time system validation loss during training with different disturbance levels.

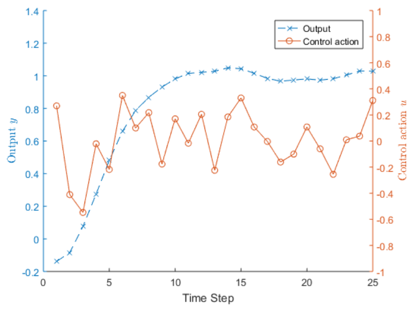
is in radians (1 degree = 0.01745 rad). A few degrees of tilt may cause the pole to fall. From the results, although future prediction MPC outperforms other approaches with a 90% prediction accuracy, its solution is infeasible when accuracy gets lower than 30%. The performance of future prediction MPC depends greatly on the prediction accuracy. For $\alpha = 0.3$ to 1, GACNs perform the best among all the compared methods.

Fig. 5 shows the validation loss during training for the five GAN-based models of different values of α and β. In general, LSGANs, and GACNs converge very fast to a low validation loss while GANs need additional iterations for convergence. WGANs and WGAN-GP oscillate a lot during the training. WGANs only converge after about 80,000 to 100,000 iterations for $\alpha = 0.1$ and $\beta = 0.9$. The fast convergence and stable training property of LSGANs and GACNs show that these two methods are good for this test case.

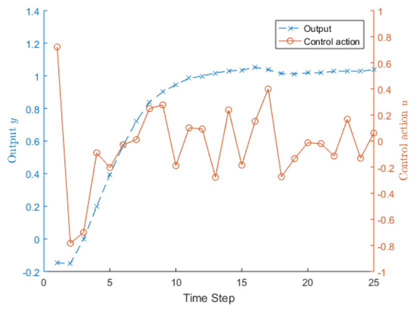
Fig. 6 shows the pole angle and control force applied to the cart of the cartpole system with different disturbance levels. The objective is to maintain the pole upright, i.e., minimize the pole angle. From the figure, we can see that the angle reduces to zero at about 5 time steps and remains at similar values afterwards. For smaller α (smaller randomness level of disturbance), the pole angle and control action are relatively steady. For larger α (randomness level of disturbance), although the pole angle oscillates during the process, it is still controlled within an acceptable range. This shows that the model can control the system such that the effect of disturbance is lessened.



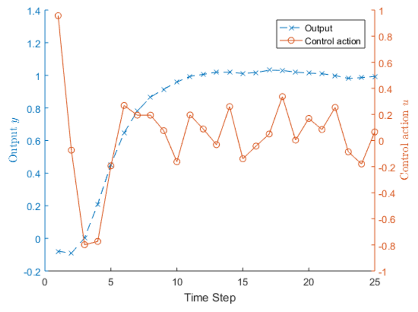
(a) $\alpha = 1, \beta = 0$



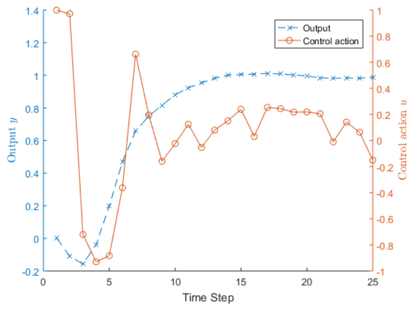
(b) $\alpha = 0.9, \beta = 0.1$



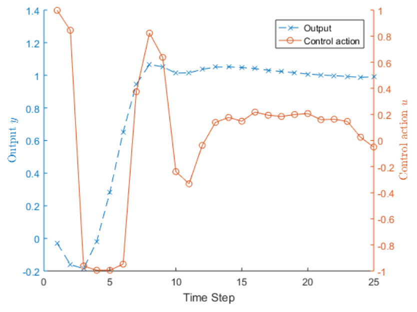
(c) $\alpha = 0.7, \beta = 0.3$



(d) $\alpha = 0.5, \beta = 0.5$



(e) $\alpha = 0.3, \beta = 0.7$



(f) $\alpha = 0.1, \beta = 0.9$

Fig. 4. Output and control action of discrete time system with different disturbance levels.

TABLE II
CARTPOLE RESULTS.

Methods	Average objective function value with different disturbance levels					
	$\alpha = 1, \beta = 0$	$\alpha = 0.9, \beta = 0.1$	$\alpha = 0.7, \beta = 0.3$	$\alpha = 0.5, \beta = 0.5$	$\alpha = 0.3, \beta = 0.7$	$\alpha = 0.1, \beta = 0.9$
Perfect MPC	0.0784	0.0783	0.078	0.0777	0.0776	0.0776
Robust MPC (Open-loop)	0.1827	0.1722	0.1579	0.1583	0.1775	0.2031
Robust MPC (Closed-loop)	0.1823	0.1715	0.1577	0.1573	0.1757	0.1992
Stochastic	Pole falls	Pole falls	Pole falls	Pole falls	Pole falls	Pole falls
Future prediction MPC	$\phi = 1\%$	$\phi = 10\%$	$\phi = 30\%$	$\phi = 50\%$	$\phi = 70\%$	$\phi = 90\%$
	Pole falls	Pole falls	Pole falls	0.2676	0.1516	**0.0996**
Supervised learning	0.1897	0.1794	0.1617	0.15	0.1427	0.1375
GANs	0.1846	0.1723	0.1567	0.1428	0.1285	0.1214
LSGANs	**0.1803**	0.1701	0.1540	0.1396	**0.1271**	0.1212
WGANs	0.1898	0.1746	0.4294	0.4454	0.4975	0.5894
WGAN-GP	0.1874	0.2023	0.1584	0.1409	0.1288	0.1257
GACNs	**0.1803**	**0.1698**	**0.1539**	**0.1393**	**0.1271**	0.1210

VI. CONCLUSION

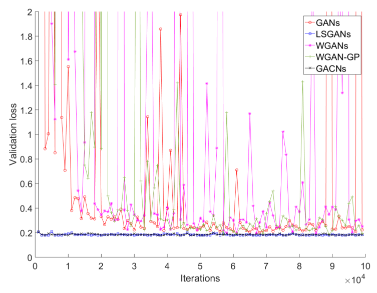
An efficient controller that can learn to adapt to the disturbance is desired for systems with unknown disturbance. In this paper, GACNs are proposed to control systems with disturbance. We also presented an optimal control problem which can be transformed to a function approximation problem, solvable by the proposed GACNs. The controller trained by GACNs can learn from the demonstrations of optimal controller in which the controller obtains a neuro-optimal solution without knowing the future disturbance and the explicit approximated objective function. A joint adversarial-least square loss is used to train the generator and the convergence analysis is given to prove the convergent property of the proposed approach. We test GACNs on a discrete time system and the OpenAI Gym CartPole environment. Experiment results show that GACNs are robust to disturbance and outperform other compared control methods.

In the future, we shall investigate using GACNs in more complex systems, such as hybrid systems [44], [45], distributed systems [46], and networked systems [47].

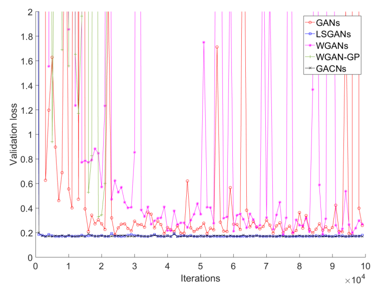
REFERENCES

[1] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, 2000.

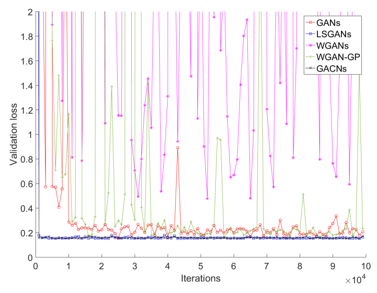
[2] W. Chen, J. Yang, L. Guo, and S. Li, "Disturbance-observer-based control and related methods—an overview," *IEEE Trans. Ind. Electron.*, vol. 63, no. 2, pp. 1083–1095, Feb 2016.



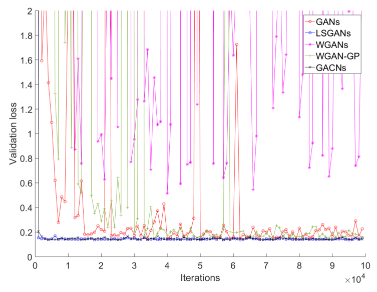
(a) $\alpha = 1, \beta = 0$



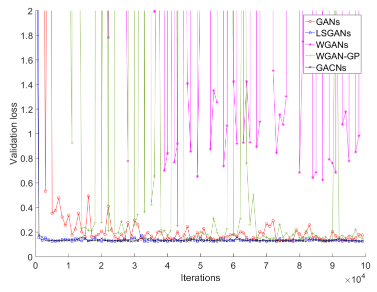
(b) $\alpha = 0.9, \beta = 0.1$



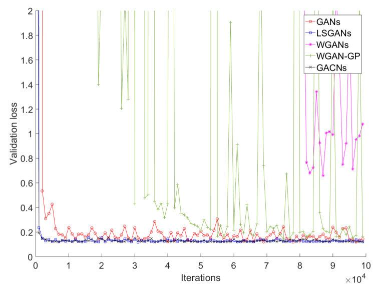
(c) $\alpha = 0.7, \beta = 0.3$



(d) $\alpha = 0.5, \beta = 0.5$

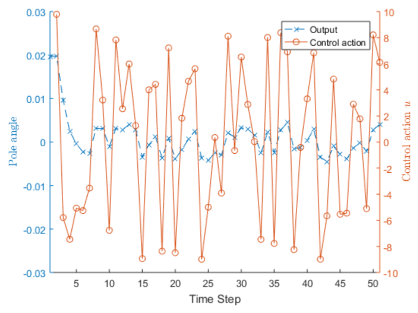


(e) $\alpha = 0.3, \beta = 0.7$

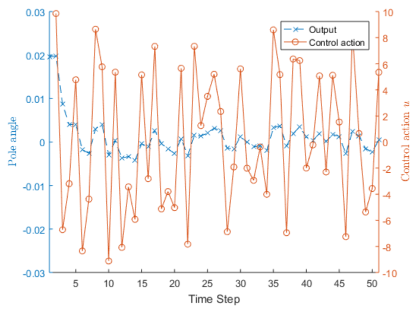


(f) $\alpha = 0.1, \beta = 0.9$

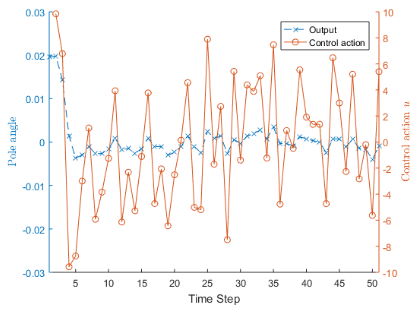
Fig. 5. Cartpole validation loss during training with different disturbance levels.



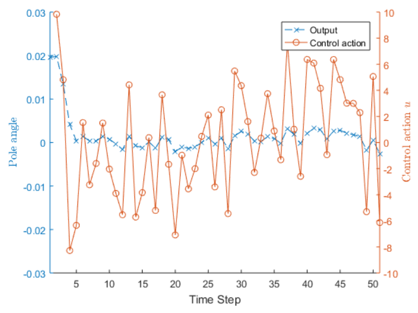
(a) $\alpha = 1, \beta = 0$



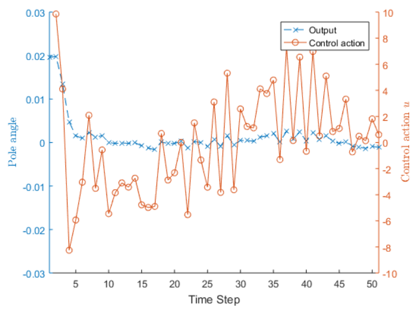
(b) $\alpha = 0.9, \beta = 0.1$



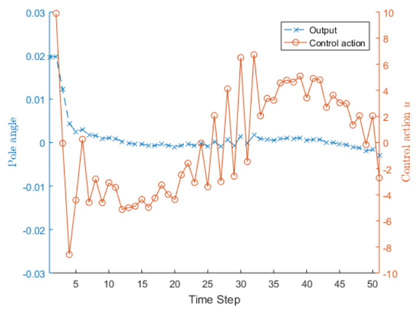
(c) $\alpha = 0.7, \beta = 0.3$



(d) $\alpha = 0.5, \beta = 0.5$



(e) $\alpha = 0.3, \beta = 0.7$



(f) $\alpha = 0.1, \beta = 0.9$

Fig. 6. Pole angle and control force applied to the cart of the cartpole system with different disturbance levels.

[3] H. Sun and L. Guo, "Neural network-based DOBC for a class of nonlinear systems with unmatched disturbances," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 2, pp. 482–489, Feb 2017.

[4] B. Xu, Y. Shou, J. Luo, H. Pu, and Z. Shi, "Neural learning control of strict-feedback systems using disturbance observer," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 5, pp. 1296–1307, May 2019.

[5] A. Bemporad and M. Morari, "Robust model predictive control: A survey," in *Robustness in identification and control*. Springer, 1999, pp. 207–226.

[6] R. E. Bellman, "Dynamic programming," *Princeton University Press*, 1957.

[7] W. B. Powell, *Approximate Dynamic Programming: Solving the curses of dimensionality*. John Wiley & Sons, 2007, vol. 703.

[8] F.-Y. Wang, H. Zhang, and D. Liu, "Adaptive dynamic programming: An introduction," *IEEE Comput. Intell. Mag.*, vol. 4, no. 2, 2009.

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

[10] B. Kiumarsi, K. G. Vamvoudakis, H. Modares, and F. L. Lewis, "Optimal

and autonomous control using reinforcement learning: A survey," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 6, pp. 2042–2062, June 2018.

[11] Y. Jiang and Z. Jiang, "Robust adaptive dynamic programming and feedback stabilization of nonlinear systems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 5, pp. 882–893, May 2014.

[12] Q. Wei, R. Song, and P. Yan, "Data-driven zero-sum neuro-optimal control for a class of continuous-time unknown nonlinear systems with disturbance using ADP," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 2, pp. 444–458, Feb 2016.

[13] S. Shao, M. Chen, and Y. Zhang, "Adaptive discrete-time flight control using disturbance observer and neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 12, pp. 3708–3721, 2019.

[14] H. Wang, P. X. Liu, J. Bao, X. Xie, and S. Li, "Adaptive neural output-feedback decentralized control for large-scale nonlinear systems with stochastic disturbances," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 3, pp. 972–983, 2020.

[15] C. Urmson *et al.*, "Autonomous driving in urban environments: Boss and the urban challenge," *J. of Field Robot.*, vol. 25, no. 8, pp. 425–466, 2008.

[16] Y. Chow, O. Nachum, E. Duenez-Guzman, and M. Ghavamzadeh, "A Lyapunov-based approach to safe reinforcement learning," in *Proc. NIPS*, 2018, pp. 8103–8112.

[17] S. Schaal, "Is imitation learning the route to humanoid robots?" *Trends in cognitive sciences*, vol. 3, no. 6, pp. 233–242, 1999.

[18] A. Y. Ng and S. J. Russell, "Algorithms for inverse reinforcement learning." in *Proc. ICML*, 2000, pp. 663–670.

[19] I. Goodfellow *et al.*, "Generative adversarial nets," in *Proc. NIPS*, 2014, pp. 2672–2680.

[20] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Proc. NIPS*, 2016, pp. 4565–4573.

[21] J. Wu, C. Zhang, T. Xue, B. Freeman, and J. Tenenbaum, "Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling," in *Proc. NIPS*, 2016, pp. 82–90.

[22] L. Ma *et al.*, "Pose guided person image generation," in *Proc. NIPS*, 2017, pp. 406–416.

[23] S. Russell, "Learning agents for uncertain environments," in *Proc. 11th Ann. Conf. on Comput. Learn. Theory*. ACM, 1998, pp. 101–103.

[24] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proc. ICML*, 2004.

[25] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning," in *Proc. AAAI*, 2008.

[26] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proc. ICML*, 2017, pp. 214–223.

[27] C. Finn, P. Christiano, P. Abbeel, and S. Levine, "A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models," in *Proc. NIPS Workshop on Adversarial Training*, 2016.

[28] C. Finn, S. Levine, and P. Abbeel, "Guided cost learning: Deep inverse optimal control via policy optimization," in *Proc. ICML*, 2016.

[29] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," in *Proc. ICLR*, May 2016.

[30] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *Proc. IEEE CVPR*, 2016, pp. 2536–2544.

[31] F. Bach and E. Moulines, "Non-strongly-convex smooth stochastic approximation with convergence rate $O(1/n)$," in *Proc. NIPS*, 2013.

[32] Y. Sokolov, R. Kozma, L. D. Werbos, and P. J. Werbos, "Complete stability analysis of a heuristic approximate dynamic programming control design," *Automatica*, vol. 59, pp. 9–18, 2015.

[33] G. Brockman *et al.*, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[34] J. Löfberg, "Approximations of closed-loop MPC," in *Proc. 42nd IEEE Conf. on Decision and Control*, 2003, pp. 1438–1442.

[35] ——, "Minimax approaches to robust model predictive control," Ph.D. dissertation, Linkoping University, Linkoping, Sweden, 2003.

[36] H. Van de Water and J. Willems, "The certainty equivalence property in stochastic control theory," *IEEE Trans. Autom. Control*, vol. 26, no. 5, pp. 1080–1087, Oct. 1981.

[37] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. Paul Smolley, "Least squares generative adversarial networks," in *Proc. ICCV*, 2017.

[38] G. Hinton, N. Srivastava, and K. Swersky, "Neural networks for machine learning - lecture 6a - overview of mini-batch gradient descent," *CSC321 Lecture slides*, 2012.

[39] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of wasserstein gans," in *Proc. NIPS*, 2017.

[40] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[41] J. Löfberg, "YALMIP : a toolbox for modeling and optimization in MATLAB," in *Proc. IEEE ICRA*, Sept 2004, pp. 284–289.

[42] Gurobi Optimization, LLC, "Gurobi optimizer reference manual," 2018. [Online]. Available: http://www.gurobi.com

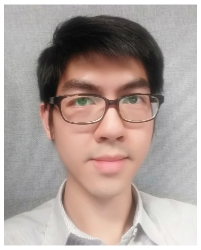
[43] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[44] M. S. Branicky, V. S. Borkar, and S. K. Mitter, "A unified framework for hybrid control: Model and optimal control theory," *IEEE Trans. Autom. Control*, vol. 43, no. 1, pp. 31–45, 1998.

[45] C. Fan, J. Lam, and X. Xie, "Peak-to-peak filtering for periodic piecewise linear polytopic systems," *Int. J. of Syst. Sci.*, vol. 49, no. 9, pp. 1997–2011, 2018.

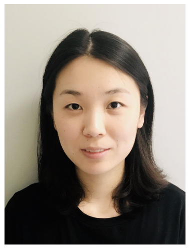
[46] A. V. Fursikov, *Optimal control of distributed systems. Theory and applications*. Amer. Math. Soc., 1999.

[47] Y. Tipsuwan and M.-Y. Chow, "Control methodologies in networked control systems," *Control Eng. Practice*, vol. 11, no. 10, pp. 1099–1111, 2003.



Kai-Fung Chu (S'17) received the B.Eng. (First Class Honors) and M.Sc. degrees both in electronic and information engineering from The Hong Kong Polytechnic University, Hong Kong, in 2013 and 2016, respectively. He is currently pursuing the Ph.D. degree in the Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong. He was a Project Engineer from 2013 to 2016. His research interests include deep learning and reinforcement learning, artificial intelligence, distributed control, optimization, and intelligent transportation systems.

Albert Y.S. Lam (S'03–M'10–SM'16) received the BEng degree (First Class Honors) in Information Engineering and the PhD degree in Electrical and Electronic Engineering from the University of Hong Kong (HKU), Hong Kong, in 2005 and 2010, respectively. He was a postdoctoral scholar at the Department of Electrical Engineering and Computer Sciences of University of California, Berkeley, CA, USA, in 2010-–12. Now he is the Chief Scientist and the Chief Technology Officer at Fano Labs, and an adjunct assistant professor at the Department of Electrical and Electronic Engineering of HKU. He is a Croucher research fellow. His research interests include optimization theory and algorithms, artificial intelligence, smart grid, and smart city. He is an Associate Editor of IEEE Transactions on Intelligent Transportation Systems and IEEE Transactions on Evolutionary Computation.



Chenchen Fan received the B.E. degree in Automation and the M.E. degree in Control Science and Engineering from Harbin Institute of Technology, Harbin, China, in 2014 and 2016, respectively. She is currently working toward her Ph.D. degree in Mechanical Engineering at the University of Hong Kong, Pokfulam, Hong Kong, as an awardee of the Hong Kong PhD Fellowship Scheme (HKPFS). Her research interests include robust filtering, periodic systems, switched systems and robust control.

Victor O.K. Li (S'80—M'81—F'92) received SB, SM, EE and ScD degrees in Electrical Engineering and Computer Science from MIT. Prof. Li is Chair of Information Engineering and Cheng Yu-Tung Professor in Sustainable Development at the Department of Electrical & Electronic Engineering (EEE) at the University of Hong Kong. He is the Director of the HKU-Cambridge Clean Energy and Environment Research Platform, and of the HKU-Cambridge AI to Advance Well-being and Society Research Platform, which are interdisciplinary collaborations with Cambridge University. He was the Head of EEE, Assoc. Dean (Research) of Engineering and Managing Director of Versitech Ltd. He serves on the board of Sunevision Holdings Ltd., listed on the Hong Kong Stock Exchange and co-founded Fano Labs Ltd., an artificial intelligence (AI) company with his PhD student. Previously, he was Professor of Electrical Engineering at the University of Southern California (USC), Los Angeles, California, USA, and Director of the USC Communication Sciences Institute. He served as Visiting Professor at the Department of Computer Science and Technology at the University of Cambridge from April to August 2019. His research interests include big data, AI, optimization techniques, and interdisciplinary clean energy and environment studies. In Jan 2018, he was awarded a USD 6.3M RGC Theme-based Research Project to develop deep learning techniques for personalized and smart air pollution monitoring and health management. Sought by government, industry, and academic organizations, he has lectured and consulted extensively internationally. He has received numerous awards, including the PRC Ministry of Education Changjiang Chair Professorship at Tsinghua University, the UK Royal Academy of Engineering Senior Visiting Fellowship in Communications, the Croucher Foundation Senior Research Fellowship, and the Order of the Bronze Bauhinia Star, Government of the HKSAR. He is a Fellow of the Hong Kong Academy of Engineering Sciences, the IEEE, the IAE, and the HKIE.