Distributionally Robust Policy Learning via Adversarial Environment Generation

Allen Z. Ren and Anirudha Majumdar, *Member, IEEE*

***Abstract*—Our goal is to train control policies that generalize well to unseen environments. Inspired by the Distributionally Robust Optimization (DRO) framework, we propose DRAGEN — Distributionally Robust policy learning via Adversarial Generation of ENvironments — for iteratively improving robustness of policies to realistic distribution shifts by generating adversarial environments. The key idea is to learn a generative model for environments whose latent variables capture cost-predictive and realistic variations in environments. We perform DRO with respect to a Wasserstein ball around the empirical distribution of environments by generating realistic adversarial environments via gradient ascent on the latent space. We demonstrate strong Out-of-Distribution (OoD) generalization in simulation for (i) swinging up a pendulum with onboard vision and (ii) grasping realistic 3D objects. Grasping experiments on hardware demonstrate better sim2real performance compared to domain randomization.**

***Index Terms*—Reinforcement learning, data sets for robot learning, generalization, continual learning, grasping.**

I. Introduction

ONE of the fundamental challenges for learning-based control of robots is the severely limited ability of trained policies to generalize beyond the specific distribution of environments they were trained on. For example, imagine a home-robot with manipulation capabilities that has been trained on a dataset containing thousands of objects. How likely is this system to succeed when deployed in different homes containing objects that the system has never encountered before? Similarly, how likely is a vision-based navigation policy for a drone or autonomous vehicle to succeed when deployed in environments with varying weather conditions, lighting (Fig. 1), or obstacle densities? Unfortunately, current techniques for learning-based control of robots (e.g., those based on deep reinforcement learning) can fail dramatically when faced with even mild distribution shifts [1].

In this work, we pose the problem of learning policies with *Out-of-Distribution* (OoD) generalization capabilities in the framework of *Distributionally Robust Optimization (DRO)*; given a dataset of environments (e.g., objects in the case of manipulation), our goal is to learn a policy that minimizes the *worst-case expected cost* across a set $\mathcal{P}$ of distributions around the empirical distribution:

$$\inf_{\theta\in\Theta} \sup_{P\in\mathcal{P}} \mathop{\mathbb{E}}_{E\sim P}[C_E(\pi_\theta)], \tag{1}$$

where θ are the parameters of the policy (e.g., weights of a neural network), and $C_E(\pi_\theta)$ is the cost incurred by policy π_θ when deployed in environment E (see Section II for a formal problem formulation). One of the key ingredients of this formalism is the choice of the set $\mathcal{P}$ of distributions over which one performs worst-case optimization. This choice is crucial in robotics applications and must satisfy two important criteria. First, the set $\mathcal{P}$ should contain *realistic* distributions. Second, it should be *broad enough* to encompass distributions that vary in *task-relevant* features (e.g., geometric features of the objects in grasping task are task-relevant, while colors are not) and thus help improve generalization to real-world OoD environments. The main technical insight of our work is to combine ideas from the theory of *Wasserstein metrics* with advances in generative modeling and adversarial training to propose a DRO framework for learning policies that are robust to realistic distribution shifts. We highlight key features of our approach next.

Statement of Contributions: The primary contribution of this work is to propose DRAGEN, a framework based on DRO for iteratively improving the robustness of policies to realistic distribution shifts by generating adversarial environments (Fig. 1). To this end, we make four specific contributions.

- Develop an approach for learning a generative model for environments (using a given training dataset) whose latent variables capture task-relevant and realistic variations in environments (Section III-A, III-B). This is achieved by training the latent variables to be *cost-predictive* and regularizing the Lipschitz constant of the cost predictor; this ensures that distances in the latent space correspond to task-relevant differences in environments.
- Propose a method for specifying the set $\mathcal{P}$ of distributions over which we perform DRO as a *Wasserstein ball* defined with respect to distance in the latent space (Section III-A). This ensures that $\mathcal{P}$ contains distributions over task-relevant and realistic variations in environments. We also provide a distributionally robust bound on the worst-case expected predicted cost of distributions in $\mathcal{P}$.
- Develop an algorithm for performing DRO with respect to the Wasserstein ball by adversarial generation of environments (Section III-C). Our overall approach then iteratively

Manuscript received September 9, 2021; accepted December 20, 2021. Date of publication January 4, 2022; date of current version January 12, 2022. This letter was recommended for publication by Associate Editor J. Kober and Editor Y. Huang upon evaluation of the reviewers' comments. This work was supported in part by the Toyota Research Institute and in part by the Office of Naval Research under Award Number N00014-18-1-2873. *(Corresponding author: Allen Z. Ren.)*

The authors are with the Mechanical and Aerospace Engineering Department, Princeton University, Princeton, NJ 08540 USA (e-mail: allen.ren@princeton.edu; ani.majumdar@princeton.edu).

Digital Object Identifier 10.1109/LRA.2021.3139949

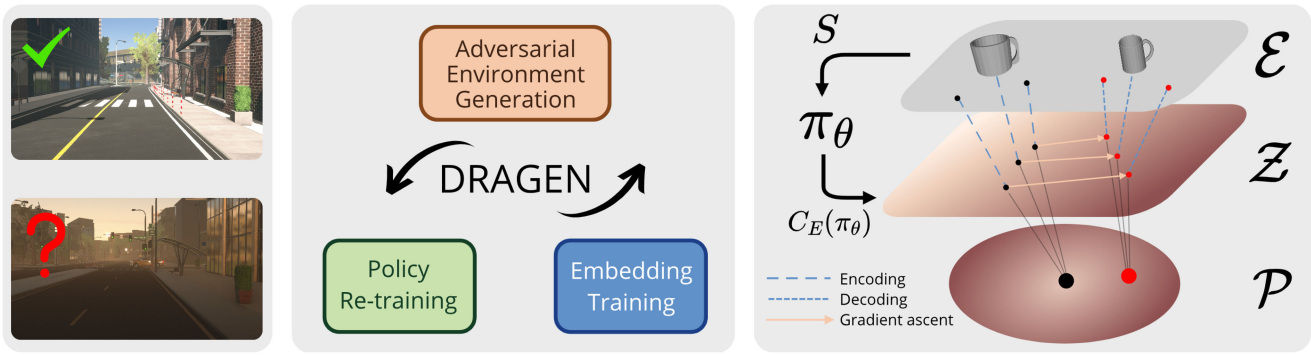

Fig. 1. (Left) Control policies often fail under distributional shift of environments such as changing lighting conditions. (Middle) Our proposed framework, DRAGEN, trains policies to generalize to such test environments (Section IV). It alternates between training a cost-predictive latent space of a generative model, generating adversarial environments via the latent space, and re-training the policy using the augmented dataset. (Right) The training dataset S consists of environments E (e.g., mugs to be grasped) in $\mathcal{E}$. They are embedded in the latent space $\mathcal{Z}$ of a generative model. We consider the resulting set of latent embeddings as a discrete distribution around which the uncertainty set $\mathcal{P}$ is defined. We apply recent progress in DRO [2] for performing worst-case optimization over $\mathcal{P}$ by perturbing the support of the discrete distribution. Costs $C_E(\pi_\theta)$ incurred by the policy trains $\mathcal{Z}$ to be cost-predictive, and allows for gradient ascent on $\mathcal{Z}$. Decoding adversarial latent variables generates realistic adversarial environments, such as mugs with smaller openings.

improves the policy by alternating between (i) re-training the generative model, (ii) generating adversarial environments for DRO, and (iii) re-training the policy using the augmented dataset.

- Demonstrate the ability of our approach to learn policies with strong OoD generalization in simulation for (i) swinging up a pendulum with onboard vision and (ii) grasping realistic 2D/3D objects (Section IV). We also validate our approach on grasping experiments in hardware and demonstrate that DRAGEN outperforms domain randomization in sim2real transfer.

A. Related Work

Distributionally robust optimization: Our work is inspired by the DRO framework in supervised learning [2]–[5], which minimizes the risk under worst-case distributional shift of data (similar to (1)). Recent progress provides a direct solution to the Lagrangian relaxation of the formulation [2], which suits our approach (Section III). In terms of the choice of uncertainty set $\mathcal{P}$, [2], [4] defines it using Wasserstein distance; this allows $\mathcal{P}$ to include distributions with different support and can thus provide robustness to unseen data. In contrast to [6] where the Wasserstein distance is defined on the semantic space (output of the last hidden layer) of the classifier, we define Wasserstein distance based on distance on the latent space of a generative model; this allows us to capture realistic distributional shifts of environments. In addition, we train a Lipschitz-regularized cost predictor from the low-dimensional latent space. This provides structure to the latent space by ensuring that nearby points correspond to environments with similar costs, and improves distributional robustness.

Environment augmentation in policy learning: Domain Randomization techniques generate new training environments by randomizing pre-specified parameters such as object textures or lighting intensities [7], [8], or randomly chaining shape primitives into new objects [9] for grasping. Similarly, data augmentation techniques such as random cutout and cropping [10], [11] have been applied to vision-based reinforcement learning (RL). Despite their simplicity, both types of techniques can be inefficient and do not necessarily generate realistic environments. For instance, training on randomly generated objects leads to worse performance in grasping realistic objects than training on the same number of realistic objects [9]. Another line of work generates increasingly difficult and complex environments [12], [13] using minimax formulations based on the agent's performance with the current policy, which are similar to our approach but do not focus on OoD generalization. Also these approaches are designed for simple environments such as gridworld and 2D bipedal terrains that are fully specified using a set of parameters, whereas our approach addresses more complex environments in the form of images and 3D objects that cannot be parameterized simply.

Adversarial training with generative modeling: Adversarial training [14] is popular in supervised learning (especially image classification) for improving the robustness of classifiers. One related direction shows that synthesizing adversarial data by searching over the latent space of a generative model can be more effective in attacking the classifier than searching over the raw image space [15]. More recent work learns possible perturbations from pairs of datasets [16] or pairs of original and perturbed data [17]. A closely related work is [18], where a set of image perturbations are pre-specified and the model learns to be robust to confusing images through a minimax objective. Among other applications, [19] attacks 3D point cloud classifiers by perturbing the latent variables of an autoencoder, similar to our setup. However, one key distinction is that while the loss/cost is differentiable through the classifier in supervised learning, the cost of an environment in our approach is determined by non-differentiable simulation. We resort to learning a differentiable cost predictor as a proxy. Adversarial training has also been applied to robotic grasping, either by randomly perturbing mesh vertices or training a Generative Adversarial

Network (GAN) [20], [21]. However, the difficulty of the objects is determined by a heuristic approach based on the antipodal metric of sampled grasps, unlike training a policy and evaluating the cost as in our approach.

II. Problem Formulation

We assume that the discrete-time dynamics of the robot are given by $s_{t+1} = f_E(s_t, a_t)$ where $s_t \in \mathcal{S} \subseteq \mathbb{R}^{n_s}$ is the state of the robot at time-step t, $a_t \in \mathcal{A} \subseteq \mathbb{R}^{n_a}$ is the action, and $E \in \mathcal{E}$ is the environment that the robot is operating in. "Environment" here broadly refers to external factors such as the object that a manipulator is trying to grasp, or the visual backdrop for a vision-based control task. Importantly, we do not assume knowledge of $\mathcal{E}$, which may be extremely high-dimensional and may not be parameterized simply. We assume access to a dataset $S := \{E_1, \ldots, E_M\}$ of M training environments. We let $P_0 := \sum_{i=1}^{M} \frac{1}{M} \delta_{E_i}$ denote the empirical distribution supported on S, where each δ_{E_i} is a Dirac delta distribution on E_i.

We assume that the robot has a sensor which provides observations $o_t \in \mathcal{O}$ of the environment. Let $\pi_\theta : \mathcal{O} \to \mathcal{A}$ denote a policy parameterized by $\theta \in \Theta$ (e.g., weights of a neural network). The robot's task is specified via a cost function; we let $C_E(\pi_\theta)$ denote the cumulative cost (over a specified time horizon) incurred by the policy π_θ when deployed in an environment E. Our goal is to learn a policy π_θ that minimizes the *worst-case expected cost* across a set $\mathcal{P}$ of distributions that contains P_0:

$$\theta^\star = \arginf_{\Theta} \sup_{P \in \mathcal{P}} \mathop{\mathbb{E}}_{E \sim P} [C_E(\pi_\theta)]. \tag{2}$$

In the subsequent sections, we will demonstrate how to tackle the two main challenges highlighted in Section I: (1) choosing a meaningful set $\mathcal{P}$ of distributions over which the worst-case optimization is performed, and (2) performing the inner maximization (generating meaningful, adversarial distributions over environments) for the outer minimization (training the policy).

III. Approach

The overall approach is visualized in Fig. 1. The key idea is to learn a generative model whose latent variables are cost-predictive and capture realistic variations in environments. This allows us to define a set $\mathcal{P}$ of distributions on this space for capturing realistic and task-relevant distribution shifts. We perform distributionally robust optimization using this set $\mathcal{P}$.

A. Learning Realistic Variations in Environments

Defining the uncertainty set $\mathcal{P}$ requires first defining the space over which distributions are supported. One option is to use the space of raw observations of environments (e.g., the raw pixel space of images). However, this space can be extremely high-dimensional and perturbations in this space typically do not correspond to realistic variations in environments, which is evident in "imperceptible attacks" in the image classification domain [14]. Instead, we opt for the latent space $\mathcal{Z} \subseteq \mathbb{R}^{n_z}$ of a generative model that captures realistic variations of environments. Previous work has demonstrated that perturbation or

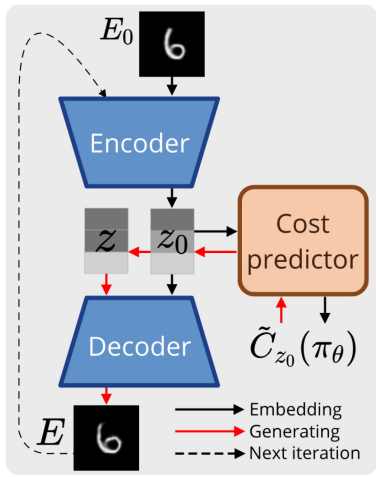

Fig. 2. Training an autoencoder and a cost predictor allows iteratively generating adversarial environments (digit images used as visual backdrops) via gradient ascent on the latent space.

interpolation in the latent space can generate realistic variations in data [22]–[24]. In the two examples detailed in Section IV, the raw environment is either a high-dimensional RGB image or a 3D object mesh. We use an autoencoder [25] as the generative model (Fig. 2):

$$z = g(E), E' = f(g(E)), \tag{3}$$

where the encoder $g : \mathcal{E} \to \mathcal{Z}$ maps the environment to a latent representation, and the decoder $f : \mathcal{Z} \to \mathcal{E}$ reconstructs the environment using the latent variable. Strictly speaking, an autoencoder is not a generative model but rather a representation model; however in practice, perturbations in the latent space generate meaningful variations in environments as shown in Section IV and in [23], [24].

We embed the empirical distribution P_0 corresponding to the training dataset S of environments (ref. Section II) into the latent space $\mathcal{Z}$. This induces a distribution $P_{\mathcal{Z}_0}$ on the latent space:

$$P_{\mathcal{Z}_0} := \sum_{i=1}^{M} \tfrac{1}{M} \delta_{z_{0_i}}, z_{0_i} = g(E_i), E_i \in S. \tag{4}$$

We then define the set $\mathcal{P}$ using the *Wasserstein distance* from optimal transport [26]. For probability measures X and Y supported on $\mathcal{Z}$, and their couplings $\Pi(X, Y)$, the Wasserstein distance over the metric space $\mathcal{Z}$ is defined as:

$$W_d(X, Y) := \inf_{H \in \Pi(X,Y)} \mathbb{E}_H[d(x, y)], \tag{5}$$

where $d(\cdot, \cdot)$ is the metric on the space $\mathcal{Z}$ (we use $d(x, y) = \|x - y\|_2$). We define the uncertainty set $\mathcal{P}$ as a Wasserstein "ball" around $P_{\mathcal{Z}_0}$ with radius ρ:

$$\mathcal{P} := \{P_{\mathcal{Z}} : W_d(P_{\mathcal{Z}}, P_{\mathcal{Z}_0}) \leq \rho\}. \tag{6}$$

Intuitively, the Wasserstein distance (also known as the "earth mover's distance") measures the minimum cost of morphing one distribution into the other. There are two key advantages of the Wasserstein distance over other divergences (e.g., the KL-divergence or other f-divergences) that make it appealing in

our setting. First, the Wasserstein distance is easier to interpret and more physically meaningful since it takes into account the underlying geometry of the space on which the probability distributions are defined (see, e.g., [27]). For example, consider distributions over objects only differing in their lengths and embedded in a latent space perfectly based on length (m). Consider three uniform distributions $P_{\mathcal{Z}_1}, P_{\mathcal{Z}_2}, P_{\mathcal{Z}_3}$ with supports [0,1], [1,2], and [2,3] respectively. The Wasserstein distance captures our intuition that the distance between $P_{\mathcal{Z}_1}$ and $P_{\mathcal{Z}_2}$ is the same as that between $P_{\mathcal{Z}_2}$ and $P_{\mathcal{Z}_3}$ while the distance between $P_{\mathcal{Z}_1}$ and $P_{\mathcal{Z}_2}$ is smaller than that between $P_{\mathcal{Z}_1}$ and $P_{\mathcal{Z}_3}$. The Wasserstein distance thus ensures distances in the latent embedding space correspond to differences in physical properties of the environments. Second, the Wasserstein distance between two distributions that do not share the same support can still be finite. This allows us to define a ball $\mathcal{P}$ around the distribution $P_{\mathcal{Z}_0}$ which contains distributions with differing supports; performing worst-case optimization over $P_{\mathcal{Z}_0}$ can thus provide robustness to unseen data.

B. Learning Task-Relevant Variations in Environments

Ideally, we would like the set $\mathcal{P}$ to contain distributions over both realistic and *task-relevant* features of environments. For example, consider a robotic manipulator learning to grasp objects and suppose that $\mathcal{P}$ contains different distributions over colors of objects. Such a set $\mathcal{P}$ can be very "large" (in terms of the radius of the Wasserstein ball); however, performing distributionally robust optimization over $\mathcal{P}$ will result in a policy that is only robust to the task-irrelevant feature of color and not necessarily robust to task-relevant geometric features. Our key intuition is that the latent space $\mathcal{Z}$ captures task-relevant variations in the environments if (i) $\mathcal{Z}$ is *cost-predictive* and (ii) closeness in the latent space corresponds to closeness in terms of costs. Then $\mathcal{P}$ captures cost-relevant (i.e., task-relevant) variations in environments.

By "cost-predictive," we mean that for the latent space $\mathcal{Z}$, there exists a mapping from the latent variable of the environment to the true cost of the environment. To satisfy (ii), such mapping should be Lipschitz continuous.

Definition 1 (γ-cost-predictive): The latent space $\mathcal{Z}$ is γ-cost-predictive if there exists a γ-Lipschitz-continuous function h that maps the latent variable of an environment to the true cost of the environment.

In practice, we train a cost predictor $h_\psi : \mathcal{Z} \to \mathbb{R}$ that maps the latent variable z of an environment E to a predicted cost $\tilde{C}_z(\pi_\theta)$ (the tilde denotes *predicted* cost instead of true cost), where π_θ is our current policy (Fig. 2). The training labels are the true cost of the environments evaluated in simulation with the current policy, $C_E(\pi_\theta)$. Furthermore, we constrain the Lipschitz constant γ_{h_ψ} of the cost predictor. Now, environments with similar predicted costs are close in the latent space, and a Wasserstein ball with large radius contains distributions over environments with large variations in task-relevant features.

As described in Section III-C, we iteratively update the policy by performing distributionally robust optimization via adversarial environment generation. Initially the policy may be sensitive to irrelevant features (e.g., color) and the set $\mathcal{P}$ contains distributions over irrelevant features. However, once we perform distributionally robust optimization with respect to $\mathcal{P}$, the policy becomes less sensitive to these irrelevant features, and $\mathcal{P}$ starts to capture task-relevant features (Fig. 3 c). Without regularizing Lipschitzness, the latent space may be cost-predictive but far away points may have similar costs; in this case, the Wasserstein distance may not capture how much distributions differ in terms of task-relevant features (see ablation in Section IV).

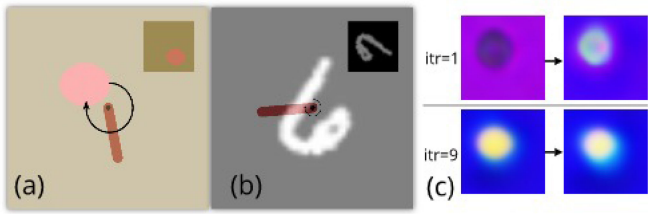

Fig. 3. (a) Landmark and (b) Digit images used in pendulum task. Top right shows the robot view from the onboard camera. (c) Original and perturbed landmark images (cropped) at different iterations. Initially generated images contain both task-relevant (landmark dimension/location) and irrelevant (landmark/background color) variations, but after iterations only task-relevant variations remain.

Embedding training: In our experiments, we use a cost predictor h_ψ with two linear layers and sigmoid activation; then γ_{h_ψ} can be upper bounded [2]:

$$\gamma_{h_\psi} \leqslant \overline{\gamma}_{h_\psi} := \|\psi_0\|_2 \|\psi_1\|_2 / 16, \tag{7}$$

where ψ_0 and ψ_1 are the weight matrices at the two layers. In practice, we constrain $\overline{\gamma}_{h_\psi}$ to some fixed value $\overline{\gamma}$. Overall, we train the encoder g, decoder f, and cost predictor h_ψ concurrently. The total loss function $\mathcal{L}$ is a weighted sum of four components:

$$\mathcal{L} = \mathcal{L}_{\text{rec}} + \alpha_1 \mathcal{L}_{\text{pred}} + \alpha_2 \mathcal{L}_{\text{Lip}} + \alpha_3 \mathcal{L}_{\text{norm}}. \tag{8}$$

where $\mathcal{L}_{\text{rec}}$ is the reconstruction loss of the autoencoder, $\mathcal{L}_{\text{pred}}$ is the l_2 loss between the predicted cost and true cost evaluated with the current policy, $\mathcal{L}_{\text{Lip}}$ is the l_2 loss between $\overline{\gamma}_{h_\psi}$ and $\overline{\gamma}$, and $\mathcal{L}_{\text{norm}}$ minimizes the norm of the embedded latent variables, which prevents the Lipschitz constant from being trivially constrained (by scaling the magnitude of latent variables to some fixed range).

C. Distributionally Robust Policy Learning via Adversarial Environment Generation

Next we explain our procedures for solving the minimax optimization (2). From Section III-A, we have chosen the uncertainty set as $\mathcal{P} = \{P_{\mathcal{Z}} : W_d(P_{\mathcal{Z}}, P_{\mathcal{Z}_0}) \leq \rho\}$. The optimization problem (subject to the uncertainty set constraint) can be reformulated as:

$$\underset{\pi_\theta}{\text{minimize}} \sup_{P_{\mathcal{Z}}} \mathop{\mathbb{E}}_{z \sim P_{\mathcal{Z}}} [C_{f(z)}(\pi_\theta)], \tag{9}$$

where $f(z)$ is the reconstructed environment (passing z through the decoder f). Searching over $\mathcal{P}$ exactly is intractable; we thus follow [2] by applying a Lagrangian relaxation with a penalty

Algorithm 1: Distributionally Robust Policy Learning via Adversarial Environment Generation.

Require: S, initial set of environments; π_θ, initial policy
1: Pre-train π_θ with S
2: **for** N iterations **do** ▷Run minimax N times
3: Evaluate $C_E(\pi_\theta), \forall E \in S$
4: Train embedding with (8) and update $P_{\mathcal{Z}_0}$
5: Sample $\{z_{0_i}\}_{i=1}^K \sim P_{\mathcal{Z}_0}$; generate $\{E_i\}_{i=1}^K$ with (13) and add to S
6: Improve π_θ with S
7: **end for**

parameter $\lambda \geq 0$:

$$\underset{\pi_\theta}{\text{minimize}} \sup_{P_\mathcal{Z}} \{\mathbb{E}_{P_\mathcal{Z}}[C_{f(z)}(\pi_\theta)] - \lambda W_d(P_\mathcal{Z}, P_{\mathcal{Z}_0})\}, \quad (10)$$

Maximizing over the distribution $P_\mathcal{Z}$ is still difficult, and thus we further apply an equivalent dual re-formulation [2] to allow maximizing over the latent variable z instead:

$$\underset{\pi_\theta}{\text{minimize}} \underset{z_0 \sim P_{\mathcal{Z}_0}}{\mathbb{E}} \{\sup_z [C_{f(z)}(\pi_\theta) - \lambda d(z, z_0)]\}. \quad (11)$$

Equivalence between (10) and (11) requires $C_{f(z)}(\pi_\theta)$ to be continuous in $f(z)$ [2], which is not reasonable as the space $\mathcal{E}$ where $f(z)$ belongs can be extremely high-dimensional (e.g. pixel space for images), and $C_{f(z)}(\pi_\theta)$ is evaluated using non-differentiable simulation. To eschew the issue, we substitute $C_{f(z)}(\pi_\theta)$ with the predicted cost $\tilde{C}_z(\pi_\theta)$ from the $\overline{\gamma}$-Lipschitz cost predictor,

$$\underset{\pi_\theta}{\text{minimize}} \underset{z_0 \sim P_{\mathcal{Z}_0}}{\mathbb{E}} \{\sup_z [\tilde{C}_z(\pi_\theta) - \lambda d(z, z_0)]\}, \quad (12)$$

Now the inner supremum can be performed with gradient ascent on the latent space (Fig. 2:

$$z \leftarrow z_0 + \eta \nabla_z [\tilde{C}_z(\pi_\theta) - \lambda d(z, z_0)], \quad (13)$$

where η is the step size. In practice we perform the minimax procedure iteratively: during inner maximization, a set of K latent variables $\{z_{0_i}\}_{i=1}^K$ are sampled from the current latent distribution $P_{\mathcal{Z}_0}$ and perturbed into $\{z_i\}_{i=1}^K$. The reconstructed environments $\{E_i\}_{i=1}^K := \{f(z_i)\}_{i=1}^K$ are added to the dataset S; during the minimization phase, the policy is re-trained using the augmented S. Between the two phases, we train the embedding using all environments in S and their cost evaluated at the current policy π_θ; this is used to update $P_{\mathcal{Z}_0}$, whose support grows over iterations.

Target ascent in the latent space: In practice we find it difficult to tune the number of gradient ascent steps when perturbing a latent variable. Instead, we set a target on how much the predicted cost $\tilde{C}_z(\pi_\theta)$ of the perturbed latent variable should increase from that of the original latent variable $\tilde{C}_{z_0}(\pi_\theta)$, and run (13) until the target is reached. However, since the cost predictor is Lipschitz-regularized, the range of its output is likely to be smaller than the true range of the cost evaluated in simulation. Thus at each iteration before generating new environments, we calculate the empirical range $\mathcal{R}(\tilde{C})$ (difference of the maximum and minimum predicted cost over all environments in S), and set the target ascent $\Delta\tilde{C}$ using a percentage $\Delta\tilde{C}_p \in [0, 1]$ of $\mathcal{R}(\tilde{C})$. We find that $\Delta\tilde{C}_p = 0.2$ works well for all experiments.

IV. Experiments

We implement our approach on two robotics tasks in simulation: (1) swinging up a pendulum with onboard vision, and (2) grasping realistic 3D objects. We also test grasping policies on a real robot arm. Through these experiments we aim to investigate the following questions: (1) Does our method offer superior OoD performance compared to data augmentation or domain randomization techniques? (2) Does our method generate seemingly meaningful environments for training? (3) Does regularizing the Lipschitz constant of the cost predictor lead to more meaningful environment variations and better OoD performance? (4) Does our method improve sim2real performance for the grasping task? For all experiments in simulation we run the minimax procedures for 30 iterations, and all results are evaluated at the iteration with the best training performance and averaged over 10 seeds. See App. A4/A5 of the extended version [28] for more ablation studies, experimental details, and hyperparameters.

A. Swinging up a Pendulum With Onboard Vision

Task and environment specification: Imagine a camera mounted to a pendulum and facing a visual backdrop (Fig. 3); the pendulum needs to swing up and balance itself using visual feedback. This is different from typical image-based pendulum tasks where the virtual camera is located away from the pendulum and is pointed at the rotating pendulum and a static backdrop (distraction). Our onboard camera setup is more representative of robotics tasks (e.g., vision-based navigation) and requires the policy to extract features from the backdrop. We consider an environment E as a backdrop image, and use two types of images (Fig. 3): (1) Landmark: randomly colored backdrop with a randomly colored, elliptical "landmark" at a fixed radial location; (2) Digit: black backdrop with white digits from the MNIST [29] and USPS [30] datasets. At each time-step, the robot's policy maps image observations from the past three time-steps to the torque applied at the joint.

Control policy training: We perform off-policy training using Soft Actor Critic (SAC) [31]. Episodes are sampled with the pendulum initialized at any angle. The reward function penalizes angle deviation from upright, angular velocity, and torque applied.

DRAGEN training: The training dataset of images is embedded in the low-dimensional latent vector space of an autoencoder. Both the encoder and decoder consist of convolutional layers and linear layers, and the decoder upsamples bilinearly. For training the cost predictor, we evaluate the cost of each image using average cost of episodes with the pendulum initialized around the lowest point. The cost is normalized between [0,1]; the lower bound corresponds to the pendulum not moving at all with itself hanging downwards, and the upper bound corresponds to the cost when the policy is trained using the true states of the pendulum instead of the camera image.

TABLE I
NORMALIZED REWARD (MEAN AND STANDARD DEVIATION OVER 10 SEEDS) FOR THE PENDULUM TASK. TOP: LANDMARK; BOTTOM: DIGIT

Method	Train	Test			
	Normal	Closer	Farther	Smaller	Larger
DRAGEN	0.858 ± 0.026	$\mathbf{0.432 \pm 0.018}$	$\mathbf{0.577 \pm 0.024}$	$\mathbf{0.761 \pm 0.025}$	$\mathbf{0.740 \pm 0.016}$
Perlin noise	0.798 ± 0.031	0.350 ± 0.027	0.495 ± 0.033	0.638 ± 0.032	0.678 ± 0.024
Gaussian noise	0.892 ± 0.030	0.357 ± 0.023	0.499 ± 0.025	0.699 ± 0.028	0.681 ± 0.026
None	$\mathbf{0.892 \pm 0.022}$	0.359 ± 0.023	0.517 ± 0.028	0.714 ± 0.025	0.669 ± 0.026

Method	Test						
	2	3	4	5	6	7	9
DRAGEN	$\mathbf{0.608 \pm 0.016}$	0.787 ± 0.021	$\mathbf{0.626 \pm 0.023}$	$\mathbf{0.589 \pm 0.018}$	$\mathbf{0.842 \pm 0.012}$	$\mathbf{0.841 \pm 0.014}$	$\mathbf{0.793 \pm 0.018}$
DRAGEN-NoLip	0.519 ± 0.022	0.771 ± 0.023	0.579 ± 0.025	0.541 ± 0.020	0.789 ± 0.018	0.782 ± 0.014	0.654 ± 0.040
DRAGEN-NoCost	0.553 ± 0.020	0.785 ± 0.023	0.611 ± 0.025	0.522 ± 0.019	0.755 ± 0.020	0.802 ± 0.018	0.741 ± 0.022
Perlin noise	0.540 ± 0.021	$\mathbf{0.797 \pm 0.018}$	0.618 ± 0.028	0.529 ± 0.021	0.750 ± 0.023	0.814 ± 0.019	0.732 ± 0.025
Gaussian noise	0.551 ± 0.018	0.754 ± 0.026	0.613 ± 0.027	0.511 ± 0.021	0.679 ± 0.027	0.772 ± 0.022	0.707 ± 0.031
Random cutout	0.549 ± 0.023	0.779 ± 0.016	0.609 ± 0.025	0.543 ± 0.019	0.756 ± 0.020	0.772 ± 0.021	0.648 ± 0.040
None	0.551 ± 0.027	0.761 ± 0.020	0.552 ± 0.037	0.540 ± 0.023	0.733 ± 0.016	0.803 ± 0.017	0.724 ± 0.034

Baselines: We benchmark DRAGEN against commonly-used data augmentation techniques in RL including pixel-wise Gaussian noise, Perlin noise [32], and random cutout [11]. Note that since the policy needs to extract spatial information from the image, some other techniques such as flipping and rotating cannot be applied.

Results: Landmark: We generate a set of training environments ("Normal" in Table I, 200 images) where the landmarks are centered along the radial direction and normally sized, and four sets of test environments where the landmarks are closer to or farther away from the center, or smaller or larger in dimensions ("Closer," "Farther," "Smaller," "Larger" in Table I). DRAGEN outperforms all baselines among all test datasets. Fig. 3 c demonstrates that DRAGEN learns to focus on generating task-relevant variations such as landmark locations and dimensions over iterations.

Results: Digit: We run experiments using digits $2, 3, 4, 5, 6, 7, 9$; digits $0, 1, 8$ are not suitable due to symmetry about both axes. Policies are trained separately for each digit with 200 training images from the MNIST dataset and then tested on the USPS dataset. The USPS dataset contains human drawings with more cursive digits. Due to space limits, we only show test performance in Table I. DRAGEN outperforms all baselines for digits $2, 5, 6, 7, 9$, and performs comparably to the strongest baseline for $3, 4$. Note that Perlin noise is a strong baseline for this setting as its structure may "augment" the white digit (Fig. 4 e). In App. A4 of [28], we show that DRAGEN outperforms Perlin noise in larger margins when colored distractions are added to the black background.

Ablation: Regularizing Lipschitz constant of the cost predictor: We also investigate whether it is useful to constrain the Lipschitz constant of the cost predictor, which we hypothesize induces task-relevant variations in generated environments. We run the additional baseline without Lipschitz regularization ("DRAGEN-NoLip" in Table I) using digit datasets, and it performs worse than DRAGEN across all test datasets. Fig. 4 c shows that DRAGEN-NoLip generates less task-relevant variations in images.

Ablation: Learning a cost-predictive embedding of environments: We remove the cost prediction loss L_{pred} from (8) - the cost predictor is not learned, and thus it is not possible to perform gradient ascent in the latent space to find adversarial environments. Instead, we randomly perturb the latent variables of existing environments in the latent space to generate new ones. Perturbations are sampled from zero-mean Gaussian distributions with diagonal covariances to roughly match the amount of perturbations generated by DRAGEN. The results are shown in Table. I as DRAGEN-NoCost. Without searching for adversarial environments varying in task-relevant features, the baseline performs worse than DRAGEN and on par with other data augmentation techniques. Fig. 4 d shows that DRAGEN-NoCost generates images with worse qualities than DRAGEN.

Fig. 4. Samples of images of digit 6 at one iteration: (a) original, (b) DRAGEN, (c) DRAGEN-NoLip, (d) DRAGEN-NoCost, (e) Perlin noise [32], (f) pixel-wise Gaussian noise, and (g) random cutout. Images at the same column are based on the same original image on the top row. DRAGEN generates new images that tend to rotate or straighten the long stroke of digit 6; such features are cost/task-relevant. Variations generated by DRAGEN-NoLip and DRAGEN-NoCost tend to be irregular and disorganized. Also Perlin noise tends to "augment" the white digit with its structure, and thus provides a strong baseline for the task.

Runtime comparison: Each experiment is run using one Nvidia RTX 2080Ti GPU and 16 server CPU threads. It takes 3 hours to run 30 iterations of DRAGEN or DRAGEN-NoLip training, and 2.5 hours for DRAGEN-NoCost. All other baselines take about 2 hours.

TABLE II
RESULTS (MEAN AND STANDARD DEVIATION (OVER 10 SEEDS) FOR THE GRASPING TASK

Method	Train Reward	Test Success - 3DNet			Hardware
		0.3	0.4	0.5	
DRAGEN	**0.911 ± 0.011**	**0.733 ± 0.025**	**0.877 ± 0.034**	**0.939 ± 0.024**	**0.975, 0.975, 0.95**
DR	0.866 ± 0.030	0.723 ± 0.038	0.814 ± 0.035	0.861 ± 0.044	0.90, 0.90, 0.90
EGAD	0.875 ± 0.016	0.721 ± 0.029	0.831 ± 0.040	0.853 ± 0.029	0.925, 0.90, 0.90
None	0.890 ± 0.034	0.703 ± 0.051	0.748 ± 0.045	0.813 ± 0.040	0.85, 0.85, 0.825

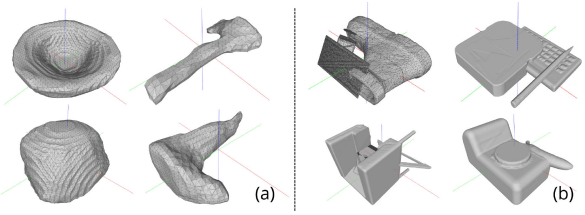

Fig. 5. Sample objects generated by (a) DRAGEN, (b) Domain Randomization. Objects in (b) are unrealistic and more irregular in shapes. Although they vary in shapes significantly, they are not realistic and can hinder the training progress. Objects generated by DRAGEN generally contain less perturbations from original objects and tend to be more realistic.

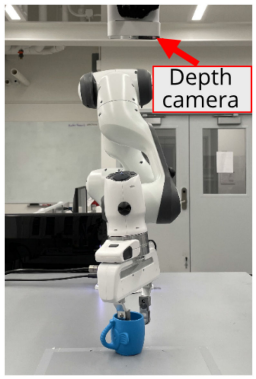

Fig. 6. Hardware setup for grasping.

B. Grasping Realistic 3D Objects

Task and environment specification: A robot arm needs to pick up an object placed at a fixed location in the PyBullet simulator [33]. Before executing the grasp, the robot receives a heightmap image from an overhead camera and decides the 3D positions and yaw orientation of the grasp. We consider an environment E as an object, and diverse synthetic 3D objects from the 3DNet [34] dataset are used for training. Policies trained in simulation with synthetic data are also transferred to a real setup (Fig. 6).

Control policy training: We follow the off-policy Q learning from [35]. The Q function is modeled as a fully convolutional network (FCN) that maps a heightmap image to a pixel-wise prediction of success of executing the grasp at the corresponding 2D location. The heightmap image is rotated into six different orientations (30 ° interval) and stacked as input to the network. The pixel with the highest value across the six output maps is used as the 2D position and yaw orientation of the grasp. The grasp height is chosen as 3 centimeters lower than the height value at the picked pixel. The friction coefficient is fixed as 0.3. The reward function is either 0 or 1 based on whether the object is successfully lifted. Due to the use of fully convolutional network and multiple grasp orientations, the cost of an object is invariant to its position and orientation.

DRAGEN training: The training dataset of object meshes is embedded in the low-dimensional latent vector space of an autoencoder. Generative modeling for 3D object meshes is more involved than that for images. The encoder is a PointNet network [36] that encodes objects from sampled 3D points on their surfaces. The decoder follows recent work in learning continuous signed distance functions (SDF, distance of a spatial point to the closest surface) for shape representation [23], [37]: it maps the pair of a query 3D location and latent variable to the SDF value at that location. After querying the SDF at many points, the mesh can be rasterized via the Marching cubes algorithm.

Training the cost predictor requires a continuous cost for the objects to allow for gradient ascent in the latent space. We assign the cost of an object between [0,1] based on the minimum friction coefficient among [0.10,0.55] needed for a successful grasp (lower value corresponds to lower cost).

Baselines: Besides no data augmentation (None), we also implement Domain Randomization (DR) technique from [9] that randomly chains shape primitives into new objects. We hypothesize that DR does not generate realistic objects and can be less efficient and effective when training policies to be tested on realistic objects. Another baseline (EGAD) is to substitute adversarial objects generated at each iteration with objects from the EGAD dataset [21]. The EGAD dataset consists of grasping objects of diverse complexities and difficulties, but most of them are not realistic, especially the ones with high complexity and difficulty.

Results: The 60 categories of objects from the 3DNet dataset are split into training and test datasets, each with 255 and 205 objects. Table II shows that DRAGEN outperforms all baselines again in both test datasets. Surprisingly, DRAGEN also performs best in the training dataset. Our generated objects (Fig. 5) may form a better training curriculum than those generated with Domain Randomization or objects from the EGAD dataset. Both DR and EGAD baselines performed worse than no augmentation in training reward.

Results: Hardware: Policies trained in simulation are tested with 40 common objects (see App. A5 of [28] for images and discussion) on a Franka Panda arm and a Microsoft Azure Kinect depth camera. For each method, we run 3 out of the 10 policies

trained in simulation (corresponding to the 10 different seeds). DRAGEN performs the best in both settings (Table II). Videos of representative trials are provided in the supplementary materials.

Runtime comparison: Each experiment is run using 1 RTX 2080Ti GPU and 32 server CPU threads. It takes about 6 hours to run 30 iterations of DRAGEN training. The domain randomization baseline takes longer time (about 8 hours) since generating new objects involves chaining shape primitives and checking if all primitives overlap. Without any data augmentation, the baseline takes about 4 hours to run. EGAD training takes the same time since new objects added to the dataset are pre-available.

V. Conclusion

We have presented DRAGEN, a framework that iteratively improves the robustness of control policies to realistic distributional shifts. By training a generative model with a cost-predictive latent space, DRAGEN can generate task-relevant and realistic variations in environments, which are then added to the training dataset to improve the policy. Results on two different robotic tasks in simulation and in sim2real transfer demonstrate the strong OoD performance of our approach.

Challenges and Future Work: Our current choice of autoencoders for generative modeling limits the ability to generate more fine-grained variations in environments. Using more sophisticated models based on GANs may improve the quality of generated environments. In addition, our current approach learns possible perturbations from the training dataset — one potential direction is to augment this approach by prescribing a set of possible perturbations and training the generative model to select/combine the provided perturbations [16]–[18].

Acknowledgment

This article solely reflects the opinions and conclusions of its authors and not TRI or any other Toyota entity.

References

[1] N. Sünderhauf *et al.*, "The limits and potentials of deep learning for robotics," *Int. J. Robot. Res.*, vol. 37, no. 4/5, pp. 405–420, 2018.
[2] A. Sinha, H. Namkoong, and J. Duchi, "Certifiable distributional robustness with principled adversarial training," in *Proc. Int. Conf. Learn. Representations*, 2018.
[3] J. Blanchet and K. Murthy, "Quantifying distributional model risk via optimal transport," *Math. Oper. Res.*, vol. 44, no. 2, pp. 565–600, 2019.
[4] P. M. Esfahani and D. Kuhn, "Data-driven distributionally robust optimization using the Wasserstein metric: Performance guarantees and tractable reformulations," *Math. Program.*, vol. 171, no. 1, pp. 115–166, 2018.
[5] H. Namkoong and J. C. Duchi, "Stochastic gradient methods for distributionally robust optimization with f-divergences," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, vol. 29, pp. 2208–2216.
[6] R. Volpi, H. Namkoong, O. Sener, J. C. Duchi, V. Murino, and S. Savarese, "Generalizing to unseen domains via adversarial data augmentation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 5334–5344.
[7] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 23–30.
[8] B. Mehta, M. Diaz, F. Golemo, C. J. Pal, and L. Paull, "Active domain randomization," in *Proc. Conf. Robot Learn.*, 2020, pp. 1162–1176.
[9] J. Tobin *et al.*, "Domain randomization and generative models for robotic grasping," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 3482–3489.
[10] D. Yarats, I. Kostrikov, and R. Fergus, "Image augmentation is all you need: Regularizing deep reinforcement learning from pixels," in *Proc. Int. Conf. Learn. Representations*, 2021.
[11] M. Laskin, K. Lee, A. Stooke, L. Pinto, P. Abbeel, and A. Srinivas, "Reinforcement learning with augmented data," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, vol. 33, pp. 19884–19895.
[12] M. Dennis *et al.*, "Emergent complexity and zero-shot transfer via unsupervised environment design," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, vol. 33, pp. 13049–13061.
[13] R. Wang, J. Lehman, J. Clune, and K. O. Stanley, "Paired open-ended trailblazer (POET): Endlessly generating increasingly complex and diverse learning environments and their solutions," 2019, *arXiv:1901.01753*.
[14] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proc. Int. Conf. Learn. Representations*, 2015.
[15] A. Jalal, A. Ilyas, C. Daskalakis, and A. G. Dimakis, "The robust manifold defense: Adversarial training using generative models," 2017, *arXiv:1712.09196*.
[16] A. Robey, H. Hassani, and G. J. Pappas, "Model-based robust deep learning," 2020, *arXiv:2005.10247*.
[17] E. Wong and J. Z. Kolter, "Learning perturbation sets for robust machine learning," in *Proc. Int. Conf. Learn. Representations*, 2021.
[18] S. Zakharov, W. Kehl, and S. Ilic, "DeceptionNet: Network-driven domain randomization," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 532–541.
[19] K. Lee, Z. Chen, X. Yan, R. Urtasun, and E. Yumer, "ShapeAdv: Generating shape-aware adversarial 3D point clouds," 2020, *arXiv:2005.11626*.
[20] D. Wang *et al.*, "Adversarial grasp objects," in *Proc. IEEE Int. Conf. Automat. Sci. Eng.*, 2019, pp. 241–248.
[21] D. Morrison, P. Corke, and J. Leitner, "EGAD! an evolved grasping analysis dataset for diversity and reproducibility in robotic manipulation," *IEEE Robot. Automat. Lett.*, vol. 5, no. 3, pp. 4368–4375, Jul. 2020.
[22] A. Joshi, A. Mukherjee, S. Sarkar, and C. Hegde, "Semantic adversarial attacks: Parametric transformations that fool deep classifiers," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 4772–4782.
[23] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "DeepSDF: Learning continuous signed distance functions for shape representation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 165–174.
[24] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas, "Learning representations and generative models for 3D point clouds," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 40–49.
[25] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.
[26] C. Villani, *Optimal Transport: Old and New*. Berlin: Springer, 2009, vol. 338.
[27] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 214–223.
[28] A. Z. Ren and A. Majumdar, "Distributionally robust policy learning via adversarial environment generation," 2021, *arXiv:2107.06353*.
[29] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[30] J. J. Hull, "A database for handwritten text recognition research," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, no. 5, pp. 550–554, May 1994.
[31] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1861–1870.
[32] K. Perlin, "An image synthesizer," *ACM Siggraph Comput. Graph.*, vol. 19, no. 3, pp. 287–296, 1985.
[33] E. Coumans and Y. Bai, "PyBullet, a Python module for physics simulation for games, robotics and machine learning," 2016–2021. [Online]. Available: http://pybullet.org
[34] W. Wohlkinger, A. Aldoma, R. B. Rusu, and M. Vincze, " 3DNet: Large-scale object class recognition from CAD models," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2012, pp. 5384–5391.
[35] A. Zeng, S. Song, S. Welker, J. Lee, A. Rodriguez, and T. Funkhouser, "Learning synergies between pushing and grasping with self-supervised deep reinforcement learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 4238–4245.
[36] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 652–660.
[37] M. Kleineberg, M. Fey, and F. Weichert, "Adversarial generation of continuous implicit shape representations," 2020, *arXiv:2002.00349*.