NOVEL ENVIRONMENT GENERALIZATION OF ACTION-CHUNKING TRANSFORMER VISUOMOTOR POLICY

GEORGE GAO [GEGAO@SEAS.UPENN.EDU],
TIANYOU WANG [WTY518@SEAS.UPENN.EDU],
KAIHAN XU [KAIHANXU@SEAS.UPENN.EDU],

ABSTRACT. This paper aims to present a scalable learning paradigm for generalizing visuomotor policies to unseen environments. We introduce GDN-ACT, or General Distillation Network prepended Action-Chunking Transformer. GDN is a trained network that maps visual image inputs from diverse environments into some ground truth state-space information identified via human domain knowledge. GDN-ACT is a modified version of the original ACT implementation [1], where the input into the transformer encoder is no longer the jointly trained CNN network, but rather the pre-trained GDN network. We show that for an example task of bimanual block grasping, the GDN-ACT trained on a **single** environment can generalize in multiple complex unseen environments with similar success rates.

1. INTRODUCTION

Imitation policies learning, at its heart, is a class of supervised learning problem that attempts to map visual observations to a sequence of output actions. When training this mapping from observation to action, past work [1][2] has often focused on producing this correspondence in a single surrounding environment. This, though understandable, has created limitations in the policies' abilities to generalize when the scene of the task is altered (although not the focus of our work, we will demonstrate this limitation in our experiments). This is, of course, to be expected, since any supervised learning problems can only at best generalize to their training distribution.

In order to tackle this limitation, one can easily argue that by training visoumotor policies on enough visual variations in the observation, that we could obtain a general policy that works robustly in any given environment. However, considering the effort needed to collect training observations and actions along with the training time cost of just a single environment, it is simply not scalable to perform this generally. In this work, we subdivide the training of visuomotor policies on various environment into a smaller problem, by attempting to create a mapping between observation and some task-dependent state-space information, and using this estimated state-space information, to inform the policy, asking it to produce the next actions. By constructing an observation-to-state mapping that is expressive and accurate enough, the training of the policy should be completely independent of visual information, and thus the policy would only need to be trained on one singular environment, and the diverse mapping would take care of generalizing to other environments that are unseen by the policy training.

Specifically, we modified the Action-Chunking Transformer [1], a powerful visuomotor policy, by replacing the original image feature embedding convolutional neural network with a pre-trained observation-to-state mapping which we call the General Distillation Network, or GDN for short. We named this combined network GDN-ACT. We will show our construction and training of the GDN network, the construction of the prepended visuomotor policy which we named GDN-ACT, and how training the GDN-ACT on the observation-action pairs of a single environment will automatically see the visuomotor policy generalize to new environments.

1.1. Contributions.

Our primary contribution is to present an approach to achieve **zero-shot generalizations to various novel environments for visuomotor policies trained on a single environment**. We show that our approach is able to achieve a reasonable task success rate of **61.60%** across 5 environments that are **unseen** by the visuomotor policy. By only training an image-to-state mapping, we show that our approach is also **highly scalable**.

2. BACKGROUND

2.1. Original ACT Formulation.

Action Chunking with Transformers (ACT) [1] is a popular visuomotor policy for fine manipulation tasks. It is composed of a conditional VAE with an VAE encode and an VAE decoder. The VAE encoder takes in action sequence and joint observation and outputs z, the latent input for the VAE decoder [3]. The VAE decoder synthesizes images from various viewpoints, joint positions, and z, and predicts action sequence. The VAE encoder is composed of a single transformer encoder block, while the VAE decoder is composed of a transformer

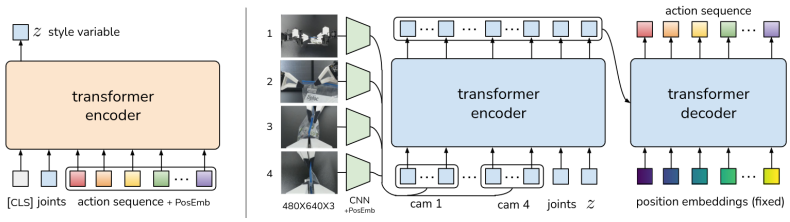

FIGURE 1. Architecture of Action Chunking with Transformers

encoder block and a transformer decoder block. The VAE encoder is discarded at inference time. The architecture of the ACT model is shown in Figure 1.

3. RELATED WORK

Ju et al. enhanced the sim-to-real transfer of deep reinforcement learning robot control policies by leveraging techniques such as domain adaptation, domain randomization, inverse dynamics models, and meta-reinforcement learning [4]. Chebotar et al.'s method iteratively adapted the distribution of simulation parameters based on real-world rollouts, thereby aligning policy behavior across simulation and reality for improved transferability [5]. Tobin et al. applied domain randomization to generate millions of procedurally created objects for training a deep neural network, which achieves high success rates in real-world grasping tasks despite being trained exclusively on simulated objects [6]. However, our work aims to show that for a singular task, there exist highly scalable approaches that can generalize imitation learning policies to novel environments with promising consistency and robustness.

4. APPROACH

4.1. **Bimanual Grasping Task Setup.** For this work, we specifically focused on the bimanual grasping task, where one robot manipulator grabs a randomly placed cube on the table and passes it to the other robot manipulator. A sample task demonstration can be found here. For each task, our approach requires humans to identify critical state-space information that would conducive to the success of the task: this will be the output of the GDN network. For the bimanual grasping task, we have decided that the cube's Euclidean position would be that state-space information. Thus, for this task, the goal of the GDN would be to construct a mapping between each image frame and the Euclidean position of the cube in the frame.

4.2. **Environment Variations for Data Collection.** In order to construct an expressive GDN mapping, we modified the original bimanual grasping MuJoco environment to have increased visual variations as shown in Figure 2. The parameters which are modified and artifacts added will be listed in section *Novel Environment Setup* below.

4.2.1. *Novel Environments Setup.*

- **Setup.** Desktop, two robotic arms, single block, three light sources, ambient headlight, five desktop ornaments, two chairs, two walls
- **Color:** Desktop, ornaments, chairs, walls: RGB uniformly sampled (0 to 1). Block: RGB uniformly sampled (0 to 1) while norm difference with desk $>= 0.6$. Robot arms: same as original. Headlight: RGB Gaussian (mean = 0.4, std = 0.2)

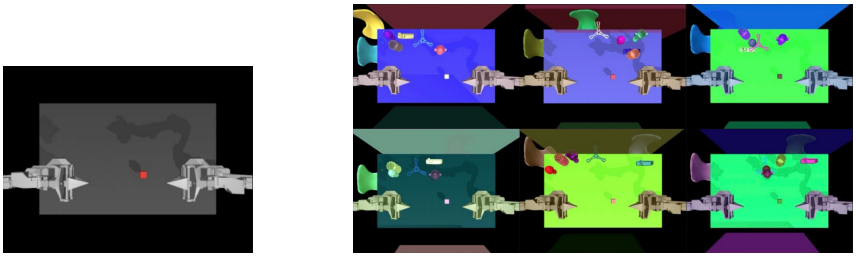

FIGURE 2. Initial Configuration of Original Environment & 6 Sampled Environment. Left: Env 0. Right: Top Row (left to right): Env 1, Env 2, Env 3; Bottom Row (left to right): Env 4, Env 5, Env 6.

- **Transparency(α).** Desktop, robot arms: fixed to 1. Block, ornaments, chairs: uniformly sampled (0.7 to 1). Walls: uniformly sampled (0 to 1).
- **Position.** Desktop, robot arms: same as original. Block: uniformly sampled, x (0 to 0.2 m), y (0.4 to 0.6 m). Lights: Gaussian (mean = [1, -1, 1]m, [1, -1, 1]m, [0, 1, 1]m, respectively; std = [0.2, 0.2, 0.2] m). Ornament 1-4: uniformly sampled, x [-0.4, 0.4], y [0.7, 0.9]; ornament 5: x [-0.4, -0.2], y [0.9, 1.1]. Chair 1: fixed x = -0.6 m, y Gaussian (mean = 0.8, std = 0.1); chair 2: fixed y = 1, x Gaussian (mean = 0, std = 0.4 m). Walls: Gaussian (mean = [0, 1, 0]m, [0, -0.2, 0]m, respectively; std = [0.1, 0.1, 0.1] m).
- **Diffusion & Specular.** Lights: RGB Gaussian (mean = 0.3, std = 0.1)
- **Direction.** Desktop, robot arms, block, ornaments, chairs, walls: fixed as original. Lights: Gaussian (means = [0, -1, -1], [-1, 1, -1], [0, -1, -1], respectively; std = [0.5, 0.5, 0.5])

4.3. **General Distillation Network Formulation.** The architecture of the GDN consists of a multi-layer convolution block, and a few fully connected layers outputting a desired state dimension. In the case of the bimanual task, the GDN network takes inputs of concatenated images of all views, and outputs a 3-vector of normalized cube positions. Figure 3 visualizes the structure of the GDN network.

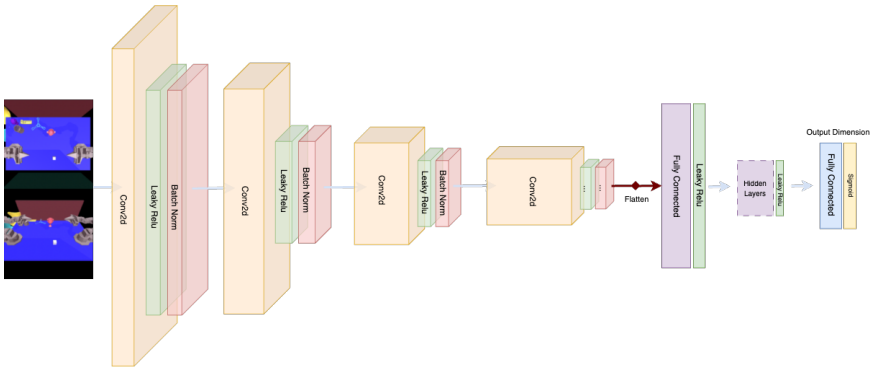

FIGURE 3. Architecture of GDN

4.4. **Implementing GDN-ACT.** To incorporate GDN into ACT, we discard the CNN feature embedding input into the VAE decoder and adopt the output from GDN instead. We illustrate this change in Figure 4. We keep the rest of the ACT unchanged. The pre-trained GDN is kept keep frozen always. During training and inference, the GDN synthesized images from multiple viewpoints and predicts the pertinent state-space information. The input to the VAE decoder are now these state-space information, robot joint positions, and latent value z along with their position embedding. By decoupling the ACT with jointly trained image input networks, we have removed the ACT's direct dependence on image pixel values. As a result, we ask the ACT to rely on the information provided by the GDN.

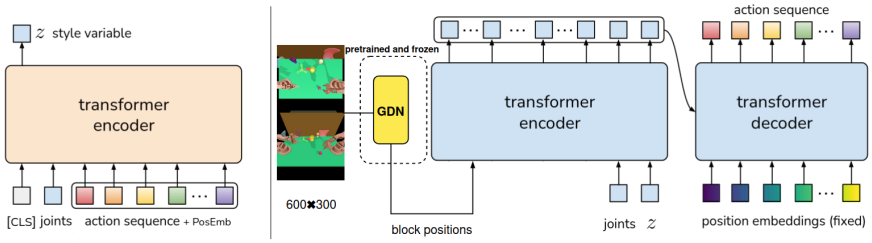

FIGURE 4. Architecture of GDN-ACT

5. EXPERIMENTAL RESULTS

5.1. **Experiment Results for GDN.** We aim to demonstrate GDN's capability to map images to cube positions with high accuracy.

5.1.1. *GDN Data Collection.* To collect training data for GDN, we ran a scripted bimanual grasping policy with different cube position initialization and collected 300x300 images from two viewing angles and cube position data directly from the MuJoco simulation. Each time the policy runs, we collect 200 image-position pairs spaced out equally: we call this one demonstration. We collected two datasets: the small dataset consists of 3 environments, each with 25 demonstrations; the big dataset consists of 6 environments, each with 40 demonstrations.

	Ending Training Loss	Ending Testing Loss
GDN on Small Dataset	2.43e-05	5.79e-05
GDN on Big Dataset	6.16e-06	1.16e-05

TABLE 1. Ending Training and testing losses of GDN for smaller and bigger dataset

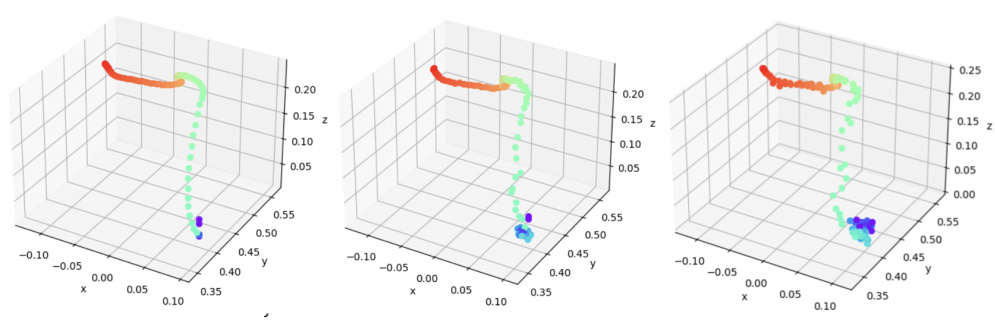

FIGURE 5. Ground truth cube positions (left). Example output of GDN on an entire demonstration of images: GDN on small dataset (right), GDN on big dataset (middle). Purple is the start of the demonstration, and red is the end.

5.1.2. *Training the GDN Network.* We trained a GDN model on the two datasets mentioned in the previous subsection separately. Training for 120 epochs with a batch number of 64 for both datasets, using a mean MSE loss across batch size, we were able to obtain a loss shown in Table 1. We also included an example of the estimated cube position output of both of the GDNs on an entire demonstration of images, compared to the ground truth cube positions in Figure 5. From these results, we see that with the bigger dataset, the GDN training was easily able to achieve much more accurate and precise cube position outputs, which is corroborated by a five times lower testing loss in Table 1 and a much less noisy cube trajectory in Figure 5. It is also important to note that the testing data for both datasets include images from all environments. Therefore, this result generalizes across drastically different environments. For the notebook of the training and testing pipeline of the big dataset for example, see here.

5.2. **Experiments Results for GDN-ACT.** We aim to show GDN-ACT's potency of generalization, the consistency of generalization, and its potential of improvement. We present the performances of GDN-ACT with GDN of different capacities for block transferring task in the MuJoco simulation. The metric we use is success rate, which reflects the rate of successful instances that the block could be grasped, and transferred to another manipulator.

5.2.1. *Training the GDN-ACT.* To train the GDN-ACT on the bimanual grasping task for a single environment, we follow an identical procedure to training the original ACT [1], running a scripted bimanual grasping policy for 50 different cube position initializations, which we call episodes. For each episode, we record 400 image observations, action sequences of length k, and joint positions. The training, which is inherently a CVAE reconstruction problem, uses L1 loss to at its essence, attempt to reconstruct the action sequence given action sequence, joint positions, learning CLS token, and observations. The model has about 80M parameters. Training in one environment takes around 60 minutes on a single 8G RTX 3070 Ti GPU, and evaluation on one environment takes around 30 minutes on the same laptop.

5.2.2. *Potency of Generalization.* To evaluate GDN-ACT's potency of generalization, we first used the GDN model trained on the small dataset, which was consisted of 25 demonstrations from three environments - environment 1, 2, and 3, and trained GDN-ACT using training procedure outlined above on environment 1. Similarly, we trained the original ACT policy on environment also. Then, both policies are test in environment 1 to 2. The results are shown in Table 2. As we can see from results shows, the original ACT has a high success rate, but it cannot handle unseen environments. In contrast, GDN-ACT was able to handle the bimanual task reasonably in the unseen environment. It show that pre-trained GDN brings the model potency of generalization, which shows the potential of policy generalization training with smaller datasets. We notice GDN-ACT's performance in the training environment is not as good as that of ACT, and thus expected with more training instances we could improve GDN-ACT's success rate further.

5.2.3. *Consistency of Generalization.* We aim to demonstrate GDN-ACT's consistency of generalization by feeding GDN with the big dataset. As discussed before, the larger dataset encompasses finer data from environment 1 through 6, and thus was used to train a newer, larger version of the GDN. Again, we trained GDN-ACT with the larger GDN model on a single environment (environment 3), and evaluated this larger GDN-ACT, the smaller GDN-ACT, and the original ACT policy in environment 1 through 6. The result of this evaluation is shown in Figure 3. Overall, the larger GDN-ACT was able to achieve an impressive average task success rate of **61.60%**. Therefore, we are able to conclude that as the training data for GDN increases, we can see that GDN-ACT's ability to

Environment	1	2
Success Rate of GDN-ACT	62%	44%
Success Rate of ACT	96%	2%

TABLE 2. Performance for the smaller GDN-ACT and ACT. For GDN-ACT and ACT, we train with 50 successful demonstrations in environment 1 and test in environments 1-2. Overall, GDN-ACT outperforms ACT in the unseen case.

Environment	1	2	3	4	5	6
Success Rate of larger GDN-ACT	64%	76%	60%	54%	76%	38%
Success Rate of smaller GDN-ACT	60%	48%	56%	16%	26%	6%
Success Rate of ACT	6%	4%	96%	2%	4%	0%

TABLE 3. Performance for All Three Models. We train the larger GDN-ACT, the smaller GDN-ACT, and ACT with 50 successful demonstrations in environment 3 and test in environment 1-6. Overall, GDN-ACT performs better in unseen environments with larger GDN.

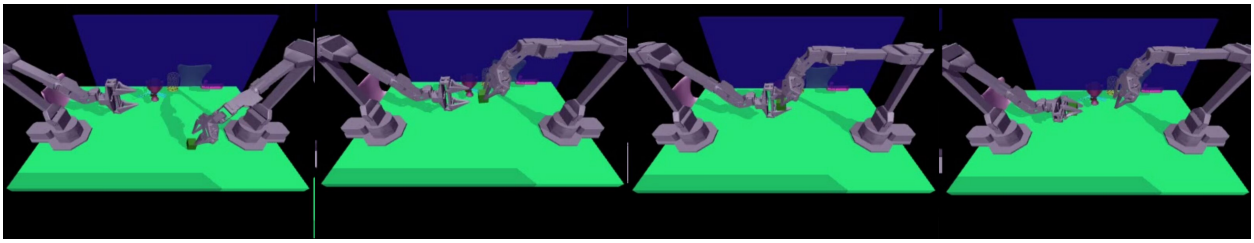

FIGURE 6. The Process of Transferring Block in Environment 6

generalize, which is training in one environment and handling other environments, increases. Thus, we expect the powerful, large GDN can bring the model capacity to handle general cases.

We show the process of transferring the block in an example of the challenging (the block and the table have similar colors) and unseen environment 6 in Figure 6. **We give more test examples for environments 1-6 at the following link.**

5.2.4. *Potential Performance of GDN-ACT.* We analyze the failure cases of GDN-ACT here to seek the logic behind failure and explore the potential improvement. We notice most of the failure cases are caused by failure to grab the block at the beginning due to small offsets. With more powerful GDNs, the end effector gets closer to the block's location, which shows that the model indeed is learning from the information from GDN. In contrast, the bare ACT policy makes the same movement in every test case, which shows GDN outperforms the precious CNN backbones in conveying essential information like 'where' and 'how' in imitation learning problems [7]. Thus, we can expect an even larger GDN with more cases can help eliminate the offsets.

What is more, we notice training GDN-ACT in a more challenging environment helps improve the ability to generalize. In the previous sections, we show that in training in environment 3, where there are dazzling light and hard-to-recognize colors, the performance of the model is better than that of training in environment 1. Thus, we guess we can improve the performance of GDN-ACT by training more powerful GDN and training the whole model in a more challenging environment. In this way, we expect the model could handle general cases after trained in a single environment.

6. DISCUSSION

With the shift from image to positional inputs for the VAE decoder, the accuracy of the GDN becomes critical. Despite limited RAM space and training time limiting our data volume, we aim to enhance box positioning by increasing episodes per environment to improve GDN-ACT's success rate for future work. Moreover, we plan to extend GDN-ACT's applicability to novel, unseen environments through extensive environment training on GDN. Furthermore, we attempt to incorporate a one-shot training on the novel environment to test both ACT and GDN ability to adapt quickly to new settings, which also reflects the model's capability of generalization, mirroring human learning dynamics, and could significantly refine its adaptability and performance.

REFERENCES

[1] Tony Z Zhao et al. "Learning fine-grained bimanual manipulation with low-cost hardware". In: *arXiv preprint arXiv:2304.13705* (2023).

[2] Cheng Chi et al. "Diffusion Policy: Visuomotor Policy Learning via Action Diffusion". In: *ArXiv* abs/2303.04137 (2023). URL: https://api.semanticscholar.org/CorpusID:257378658.

[3] Jacob Devlin et al. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

[4] H. Ju, R. Juan, R. Gomez, et al. "Transferring policy of deep reinforcement learning from simulation to reality for robotics". In: *Nature Machine Intelligence* 4 (2022), pp. 1077–1087.

[5] Y. Chebotar et al. "Closing the sim-to-real loop: adapting simulation randomization with real world experience". In: *Proceedings of the International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 8973–8979.

[6] Joshua Tobin, Wojciech Zaremba, and P. Abbeel. "Domain Randomization and Generative Models for Robotic Grasping". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2017), pp. 3482–3489. URL: https://api.semanticscholar.org/CorpusID:4530385.

[7] Shikhar Bahl et al. "Affordances from human videos as a versatile representation for robotics". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 13778–13790.