Empirical Analysis of Sim-and-Real Cotraining of Diffusion Policies for Planar Pushing from Pixels

Adam Wei, Abhinav Agarwal, Boyuan Chen, Rohan Bosworth, Nicholas Pfaff, Russ Tedrake*

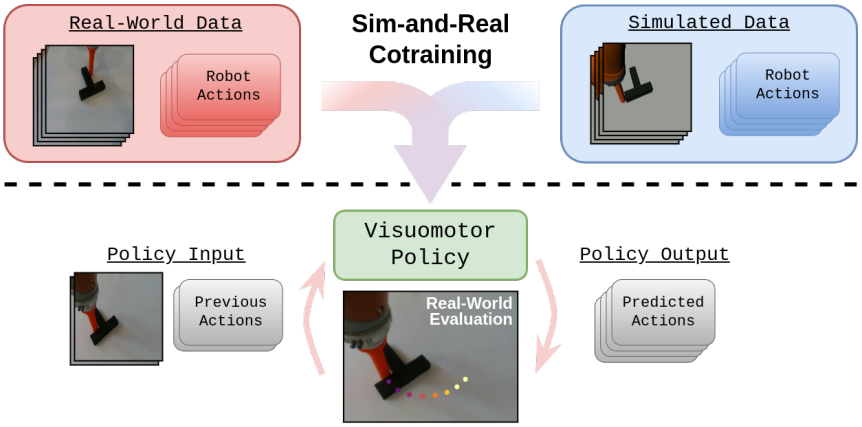


Figure 1: *Sim-and-real cotraining* aims to train visuomotor policies using both simulated and real-world robot data to maximize performance on a real-world objective.

Abstract: *Cotraining* with demonstrations generated both in simulation and on real hardware has emerged as a promising recipe for scaling imitation learning in robotics. This work seeks to elucidate basic principles of this *sim-and-real cotraining* to inform simulation design, sim-and-real dataset creation, and policy training. Our experiments confirm that cotraining with simulated data can dramatically improve performance, especially when real data is limited. We show that performance gains scale with additional simulated data until a plateau; adding more real data can increase this performance ceiling. The results also suggest that physical domain gaps may be more impactful than visual fidelity for contact-rich tasks. Perhaps surprisingly, some visual gap appears to help cotraining. We investigate this nuance and other mechanisms that facilitate positive transfer between sim-and-real. Lastly, we provide ablations for 2 alternative cotraining formulations. Videos can be found here: https://sim-and-real-cotraining.github.io/.

Keywords: cotraining, imitation learning, simulated data

1 Introduction

Foundation models trained on large datasets have transformed natural language processing [1][2] and computer vision [3]. However, this data-driven recipe has been challenging to replicate in robotics since real-world data for imitation learning can be expensive and time-consuming to collect [4]. Fortunately, alternative data sources, such as simulation and video, contain useful information for robot learning. In particular, simulation is promising since it can automate robot-specific data collection. This paper investigates the problem of *cotraining* policies via imitation learning with both simulated and real-world data. Our results confirm that simulation is a powerful tool for scaling imitation learning and improving performance. We also provide insights into the factors and underlying principles that affect *sim-and-real cotraining*.

*All authors are with the Massachusetts Institute of Technology, Cambridge, MA, 02139. Corresponding author: weiadam@mit.edu

9th Conference on Robot Learning (CoRL 2025), Seoul, Korea.

Many researchers are investing in simulation for data generation in robotics [5][6][7]. Concurrently, large consolidated datasets have made real-world data more accessible [8][9]. Both data sources are valuable, but neither has proven sufficient on its own [4]. For instance, simulated data is scalable, but requires sim2real transfer; real-world data does not have this issue, but it is more expensive and time-consuming to collect. Thus, understanding how to use both data sources symbiotically could unlock massive improvements in robot imitation learning.

Given data from simulation and the real-world, *sim-and-real cotraining* aims to train policies that maximize a real-world performance objective. This is a common and important problem in robotics [10][11][12]. Our goal is to understand the mechanisms underlying sim-and-real cotraining for imitation learning. We study how various factors affect performance, such as data scale, data mixtures, and distribution shifts. These findings can inform best practices for both sim-and-real cotraining and simulator design for synthetic data generation.

Specifically, we investigate sim-and-real cotraining for Diffusion Policy [13] and evaluate performance on the canonical task of planar-pushing from pixels. We provide an extensive suite of experiments and ablations that spans over 50 real-world policies (evaluated on 1000+ trials) and 250 simulated policies (evaluated on 50,000+ trials). More concretely, we show:

1. **Cotraining with sim data improves policy performance by up to 2-7x**, but the performance gains from scaling sim eventually plateau without more real data.
2. All sim2real gaps impact the value of synthetic data. **Improving the physical accuracy of simulators** could massively increase the downstream performance of cotrained policies.
3. **High-performing policies learn to distinguish sim from real** since the physics of each environment require different actions. Surprisingly, cotraining with perfectly rendered sim data reduces performance since the policies can no longer visually discern the two domains. In contrast, providing a one-hot encoding of the environment improves policy performance.
4. **Cotraining from sim provides positive transfer to real.** Simulated data can fill gaps in the real-world data and improve the real-world test loss according to a power law.
5. **Ablations for 2 alternative cotraining formulations**: one that uses classifier-free guidance [14] and another that modifies the loss to encourage better representation learning.

Focusing on the single canonical task of planar pushing from pixels allowed us to be exhaustive in our investigation, but it also limits our study. Thus, this paper's main contributions are the trends and analysis rather than the absolute values. We hope our findings can inform larger-scale cotraining, where thorough sweeps and analysis could be prohibitively expensive.

2 Related Work

Data Generation in Simulation: Prior works have shown that simulation can scale up data generation [5][6] and augmentation [7][15] for imitation learning. Additionally, infrastructure for synthetic data generation is improving, with advancements in environment generation [16][17], real2sim [18], and motion planning [19]. This paper investigates how the growing body of work in simulated data generation can be used in conjunction with real-world datasets [8][9] for robot imitation learning.

Sim2real Transfer vs Sim-and-Real Cotraining: Sim-and-real cotraining adopts a slightly different philosophy to policy learning than the traditional *sim2real* pipeline [11][20]. Instead of learning policies in sim and applying sim2real techniques [10][21][22], robots learn from sim and real-world data *simultaneously* [23]. Prior works have used sim-and-real cotraining on assembly [12] and kitchen tasks [24]. The general problem of learning from multiple domains is ubiquitous in machine learning [1][25][26][27]. In robotics, most works have focused on cotraining from *real* domains [8][28][29]; the specific problem of cotraining from simulated domains can be seen as a special case of cross-embodiment training worthy of focused study. We believe simulation will play a key role in robot imitation learning given the progress in synthetic data generation and the ability to perform large-scale reproducible testing. This work aims to provide initial insights towards this future.

Figure 2: An example of planar pushing [32]. The circle is the pusher, the black T is the slider, and the yellow T is the goal.

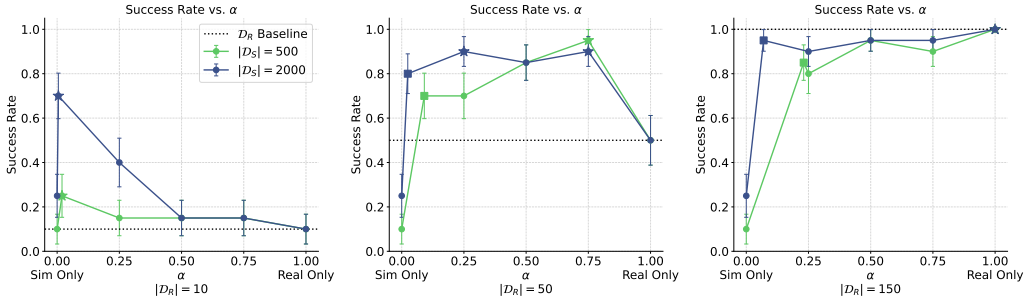


Figure 3: Real-world performance of cotrained policies at different data scales and mixing ratios. ★ depicts the optimal α and ■ depicts the natural mixing ratio. When real data is limited, cotraining with sim data can improve performance by 2-7x.

3 Preliminaries and Problem Formulation

We study a specific instantiation of the cotraining problem, where a behavior cloning policy is jointly trained on simulated and real-world robot data. Let $\mathcal{D}_R = \{\tau_i\}_{i=1}^{N_R}$ and $\mathcal{D}_S = \{\tau_i\}_{i=1}^{N_S}$ be datasets of real and sim trajectories respectively. Let $|\mathcal{D}|$ be the number of trajectories in $\mathcal{D}$. Given $\mathcal{D}_R$ and $\mathcal{D}_S$, cotraining aims to learn a policy that maximizes performance on a real-world performance objective. Here, we cotrain Diffusion Policies [13] (see Appendix C for training details) and measure performance as the success rate on planar pushing from pixels.

Planar Pushing From Pixels: Planar pushing is a manipulation problem that requires a robot to push an object (slider) on a flat surface with a cylindrical end effector (pusher) to a target pose (ex. Figure 2). *"From pixels"* indicates that the observation space contains images as opposed to the full system state. We emphasize that the goal of this work is not to "solve" planar pushing. Instead, planar pushing serves as a test-bed for our investigations into cotraining. We chose this task since it is a canonical task that captures core challenges in robotics [13][30][31], such as high-level reasoning, visuomotor control, and contact, while still admitting methods for data generation [32][33][34].

Data Collection: A human teleoperator collects real data and an optimization-based planner [32] generates simulated trajectories. Each sim trajectory is replayed in `Drake` [35] to render the observations. The simulated planner is "near-optimal" [32], whereas the teleoperator is not. This introduces an action gap between the two datasets that will become relevant for our analysis in Section 6. The simulation mimics the real-world setup. Both datasets are collected on a `KUKA LBR iiwa 7`. The action space is the target x-y position of the pusher (the robot's end effector); the pusher's z-value and orientation are fixed. The observation space includes the pusher's pose and two RGB images from an overhead camera and a wrist camera. Figure 1 visualizes the overhead camera view.

4 Real World Experiments

One way to cotrain from multiple domains is to reweight the datasets [26][28]. We investigate the following questions about this simple, but common, algorithm: 1) Does cotraining with sim data improve real-world performance (Section 4.2)? How do the size and mixing ratio of both datasets affect performance (Section 4.3)? How does cotraining compare to finetuning (Section 4.4)?

4.1 Experimental Setup

We refer to the method above as *vanilla cotraining*. In vanilla cotraining, policies are trained on $\mathcal{D}^\alpha$, a mixture of the real data, $\mathcal{D}_R$, and sim data, $\mathcal{D}_S$. We sample from $\mathcal{D}^\alpha$ by sampling from $\mathcal{D}_R$ with probability α and $\mathcal{D}_S$ otherwise. Training on $\mathcal{D}^\alpha$ is equivalent to minimizing $\mathcal{L}_{\mathcal{D}^\alpha} =$

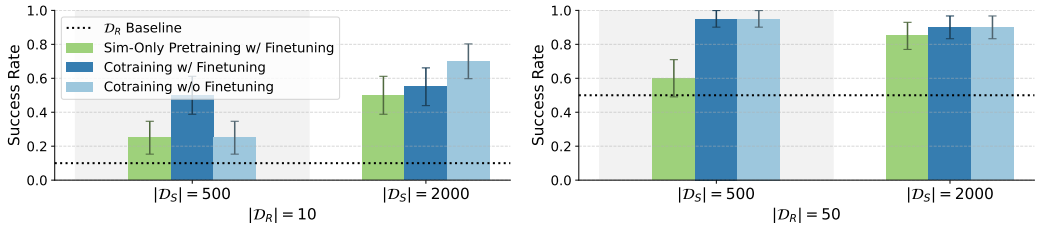


Figure 4: Pretraining with a cotraining mixture significantly outperforms pretraining with sim-only.

$\alpha\mathcal{L}_{\mathcal{D}_R} + (1-\alpha)\mathcal{L}_{\mathcal{D}_S}$. Thus, α acts as a reweighting parameter. Ideally, $\mathcal{L}_{\mathcal{D}_R}$ ($\alpha = 1$) is our loss; however, when $|\mathcal{D}_R|$ is small, this leads to poor performance. We remedy this by adding sim data and using α to prevent $\mathcal{D}_S$ from dominating the loss. Historically, many resources are used to sweep dataset reweightings (i.e. α) since they have an outstanding effect on performance [1][28].

We cotrain policies for all combinations of $|\mathcal{D}_R| \in \{10, 50, 150\}$, $|\mathcal{D}_S| \in \{500, 2000\}$, and $\alpha \in \{0, \frac{|D_R|}{|D_R|+|D_S|}, 0.25, 0.5, 0.75, 1\}$. $\alpha = 0$ and $\alpha = 1$ are equivalent to training solely on $\mathcal{D}_S$ or $\mathcal{D}_R$ respectively (i.e no cotraining); $\alpha = \frac{|\mathcal{D}_R|}{|\mathcal{D}_R|+|\mathcal{D}_S|}$ is the natural mixing ratio since it is equivalent to concatenating $\mathcal{D}_R$ and $\mathcal{D}_S$ without reweighting. We evaluate each policy's real-world success rate and error bars according to Appendix A.

4.2 Does Cotraining Improve Performance?

Figure 3 presents the results. We refer to the three increasing values of $|\mathcal{D}_R| \in \{10, 50, 150\}$ as the low, medium, and high data regimes. In the low and medium data regimes, cotraining *improves performance over the real-only baselines for all α's*. The best cotrained policy for $|\mathcal{D}_R| = 10$ improved performance from 2/20 to 14/20, and the best cotrained policy for $|\mathcal{D}_R| = 50$ improved performance from 10/20 to 19/20. To achieve similar improvements with only real data, we would have needed to increase $|\mathcal{D}_R|$ by several times. In the high data regime, the real-only baseline achieved a success rate of 20/20; however, the policy is not perfect since our experiments have variance. Thus, the performance drop from cotraining in the high-data regime is minor and within the error margins. These findings show that simulated data generation and cotraining are promising ways to scale up imitation learning. We verify these results at a larger-scale in Section 5.2.

4.3 The Effect of α, $|\mathcal{D}_R|$, and $|\mathcal{D}_S|$ on Cotraining

Effect of α: Performance is sensitive to α, especially for $|\mathcal{D}_R| = 10$. The optimal α's appear to increase with $|\mathcal{D}_R|$. Intuitively, overfitting to $\mathcal{L}_{\mathcal{D}_R}$ is less detrimental when $|\mathcal{D}_R|$ is large, so it is desirable to bias the mixing ratio towards real. Performance decreases nearly discontinuously as $\alpha \searrow 0$. This suggests that even a small mixture of real data can drastically improve performance.

Effect of $|\mathcal{D}_R|$: Figure 3 suggests that cotraining is most effective in the low to medium data regime. We reached the high-data regime for planar-pushing from pixels with just 150 real demos; however, other tasks often remain in the medium data regime even with thousands of demos [36]. Thus, in most settings, we expect the value of cotraining to be high.

Effect of $|\mathcal{D}_S|$: Scaling $|\mathcal{D}_S|$ improved or maintained performance across the board. Scaling $|\mathcal{D}_S|$ also reduced the policy's sensitivity to α, which is highly desirable.

4.4 Finetuning Comparison

We compare cotraining with two methods for finetuning. *1) Sim-only pretraining:* pretrain with $\mathcal{D}_S$, then finetune with $\mathcal{D}_R$. *2) Cotraining with finetuning:* cotrain with $\mathcal{D}_S$, $\mathcal{D}_R$, and optimal α, then finetune with $\mathcal{D}_R$. Figure 4 shows that methods that employ data mixing outperform methods that train on sim-and-real separately. In the single-task setting, finetuning the cotrained models does not consistently improve performance. We hypothesize that the real-world is already in distribution for the cotrained policies; thus, finetuning could cause overfitting on a case-by-case basis.

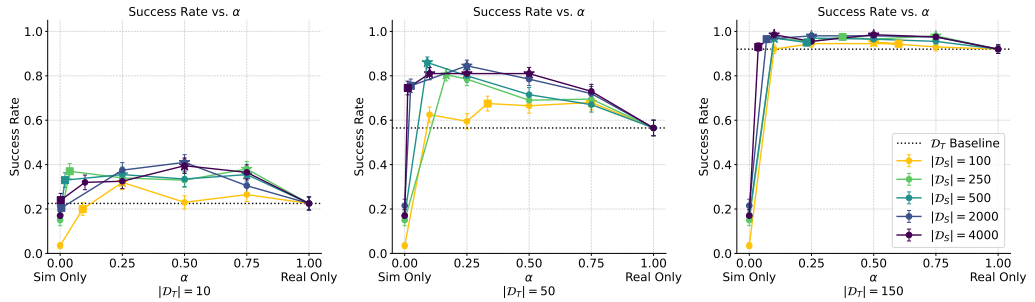


Figure 5: Performance of cotrained policies in simulation at different data scales and mixing ratios. The results of the simulated experiments agree with the real-world results. Scaling up $|\mathcal{D}_S|$ improves performance until a performance plateau. This plateau can be improved by adding more *target* data.

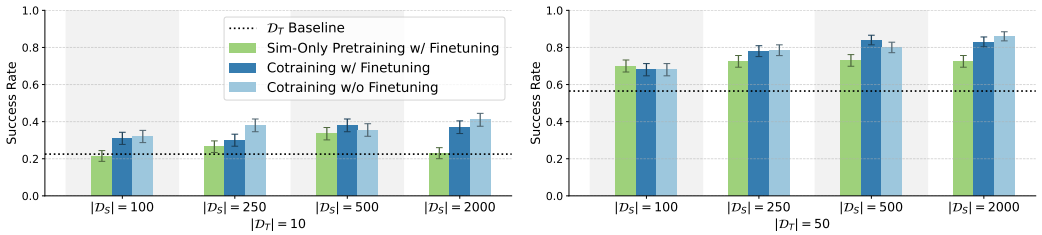


Figure 6: Comparing cotraining and finetuning in simulation. The results are consistent with the real-world experiments (Figure 4).

5 Simulation Experiments

In this section, we conduct experiments in simulation on a *sim-and-sim* setup that mimics the *sim-and-real* setup. Simulation enables experiments that would be challenging to conduct in the real-world by providing two main advantages: 1) automated, high-confidence evaluations, 2) explicit control over the *sim2real* gap between the two cotraining environments. Section 5.2 uses *Advantage 1* to scale up the real-world experiments. Section 5.3 uses *Advantage 2* to study the effect of distribution shifts on cotraining.

5.1 Sim-and-Sim Setup

Cotraining from sim-and-sim requires 2 distinct simulation environments. We reuse the same sim environment from our real-world experiments and continue referring to it is as *sim*. We introduce a second *target* sim environment as a surrogate for the *real-world* environment (visualized in Appendix B.1). Given data from both the *sim* environment and the *target sim* environment, we aim to learn policies that maximize performance in the *target* simulation.

To ensure our simulation experiments are informative for the real world, we designed the *target* sim environment to mimic the sim2real gap along 3 different axis: visual gap, physical gap, and action gap (see Table 2 and Figure 12). We collect data in both environments according to Section 3. Although the *target* sim dataset serves the same purpose as $\mathcal{D}_R$ in our real-world experiments, we denote it by $\mathcal{D}_T$ to make the distinction clear. We evaluate policies in simulation as per Appendix B.5. Notably, we test each policy on 200 random trials instead of the 20 used in Section 4.

5.2 Single-Task Cotraining Asymptotes

We scale up the experiments from Section 4 in our sim-and-sim setup to answer the following research questions: 1) Can we verify our real-world results with higher-confidence evaluations? 2) How does performance scale with $|\mathcal{D}_S|$?

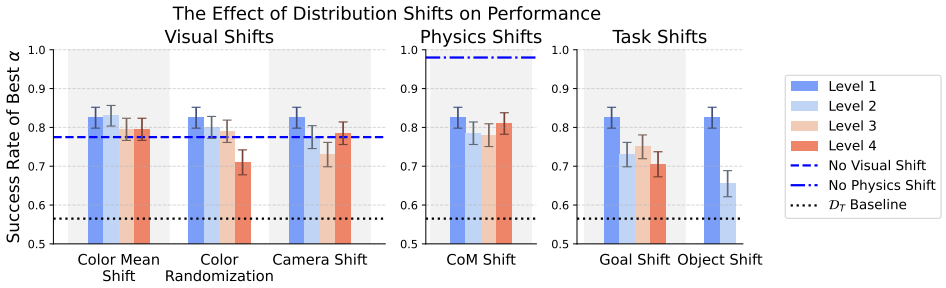


Figure 7: Performance of the best data mixture for each shift and intensity. $|\mathcal{D}_T| = 50$, $|\mathcal{D}_S| = 2000$.

The results in Figure 5 mostly agree with Section 4.2. We highlight key differences. First, cotraining increased performance even in the high-data regime; but the overall improvements were lower than Section 4. Second, the optimal α's were lower. We hypothesize that this is because the *sim2target* gap was less than the *sim2real* gap. The finetuning results (Figure 6) are consistent with Section 4.4.

Scaling $\mathcal{D}_S$ improves performance, but this trend eventually plateaus. This suggests that sim data cannot replace real data; real data is still needed to increase the cotraining ceiling. Figure 13 compares the performance plateaus at different data scales to the performance of policies trained on only *target sim* data. In our specific setup, the benefits from cotraining with sim data were equivalent to increasing our *target sim* dataset by roughly 2-3x.

5.3 Distribution Shift Experiments

We use our sim-and-sim setup to study how various sim2real gaps affect cotraining and their implications for simulator design and data generation. We find that larger sim2real gaps reduce performance, but some distribution shifts are more impactful than others. This agrees with intuition and theoretical bounds [26]. We present our methodology and highlight a few noteworthy trends.

We explored the effect of 6 different types of distribution shifts at increasing intensity levels. These 6 shifts fall under 3 categories of sim2real gap. **Color mean shift**, **color randomization**, and **camera shift** are examples of *visual* gap; slider **center-of-mass (CoM) shift** is an example of a *physics* gap; **goal** and **object shift** are examples of *task* gap. We quantify and visualize the 4 intensity levels for each shift in Appendix B.3. Concretely, we collect a single *target* sim dataset, $\mathcal{D}_T$, with 50 demos and reuse it for all policies in this experiment. Distribution shifts are introduced individually by modifying $\mathcal{D}_S$. Figure 7 reports the performance of the best data mixture for each shift.

For Level 1 CoM shift, a physics gap remains since the sim and *target* sim environment use different physics models (see Table 2). Similarly, for the Level 1 visual shifts, a visual gap remains since the *target* environment contains shadows while the sim environment does not (see Figure 14). To study the effect of entirely eliminating visual or physics gaps, we collect 2 additional simulated datasets: one with *no visual gap*, and one with *no physics gap*. We cotrain policies on these datasets and present them in Figure 7 as well.

Performance is sensitive to task and physics shifts, and least sensitive to color mean shift. The policy trained with no physics shift achieves 15.5% higher success rate than the policy trained with Level 1 shift. This is a significant difference in success rate. Performance is less sensitive to subsequent increases in the physics gap. Nonetheless, the magnitude of the initial drop suggests that **accurate simulation physics are important for contact-rich tasks**. Tasks that emphasize semantic or visual reasoning and prehensile tasks might exhibit less performance degradation.

Roughly speaking, performance improves with better rendering. This is consistent with previous works [37][38]; but paradoxically, the policy cotrained with no visual gap is slightly weaker. **This suggests that a small amount of visual gap is desirable.** Without it, the policy cannot distinguish between the two environments. We hypothesize that this harms action prediction. This is a subtle, but important point that deserves more discussion. We examine it more closely in Section 6.

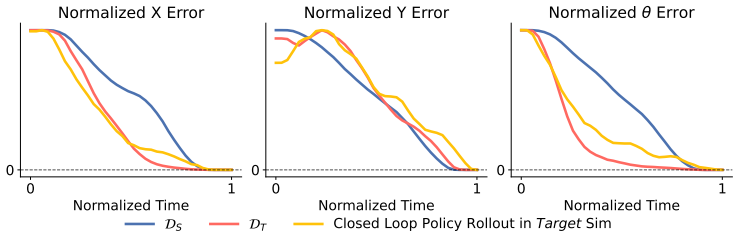


Figure 8: The red and blue lines show the average normalized slider error over time for demos in $\mathcal{D}_T$ and $\mathcal{D}_S$ respectively. The yellow line plots the same statistic for 200 rollouts in the *target* sim environment for a cotrained policy with $|\mathcal{D}_T| = 50$, $|\mathcal{D}_S| = 2000$, $\alpha = 0.024$. Although $|\mathcal{D}_S| \gg |\mathcal{D}_T|$ and $\alpha \approx 0$, the yellow line is closer to the error for $\mathcal{D}_T$ (red) than $\mathcal{D}_S$ (blue).

6 How does cotraining succeed?

We analyze factors that contributed to the success of cotraining. Section 6.1 shows that a policy's ability to identify its domain is crucial. High-performing policies behave distinctly more like $\mathcal{D}_R$ when deployed in real, and more like $\mathcal{D}_S$ when deployed in sim. These findings were also true in our simulated experiments. We remind the reader that $\mathcal{D}_S$ and $\mathcal{D}_R$ contain noticeable action gap.

If cotrained policies learn distinct behaviors for each domain, then how does training from one improve performance in the other? Section 6.2 discusses two mechanisms that facilitate this positive transfer: 1) *dataset coverage*, 2) *power laws* on real-world performance metrics.

6.1 Sim-and-Real Discernability

Figure 8 demonstrates that the behaviors of cotrained policies in the *target* sim environment are distinctly closer to $\mathcal{D}_T$ than $\mathcal{D}_S$. Table 1 shows this is not a coincidence: binary probes on the learned embeddings of cotrained policies can accurately classify sim and real. Unsurprisingly, the observation embedding carries information about the environment label; however, policies choose to preserve this information (which would otherwise be destroyed by the data processing inequality) until the final activation. In other words, cotrained policies learn that the environment label is important for action prediction.

High-performing policies must differentiate sim from real since each environment's physics require different actions. Figure 9 supports this intuition. When $\mathcal{D}_S$ and $\mathcal{D}_T$ contain physics gap (solid lines in Figure 9), removing visual differences between sim and *target* decreases success rate. Adding a one-hot encoding for the environment recovers this performance. On the other hand, when $\mathcal{D}_S$ and $\mathcal{D}_T$ do not contain physics gap (dotted lines), the opposite trend emerges: removing the visual gap improves performance. This supports our claim: sim-and-real discernibility is important because the physics between the two environments are different.

Given this analysis, we study two alternative cotraining formulations in Appendix D: one that uses classifier-free guidance [14], and another that modifies the loss for better representation learning.

6.2 Mechanisms For Positive Transfer in Cotraining

Although cotrained policies learn different actions for each domain, learning from sim still improves performance. We discuss mechanisms and evidence for this positive transfer.

Coverage: We hypothesize that cotraining with $\mathcal{D}_S$ improves performance by filling gaps in $\mathcal{D}_R$. In other words, a policy deployed in real learns most of its behavior from $\mathcal{D}_R$ (see Figure 8) and relies on $\mathcal{D}_S$ in states that were not covered by $\mathcal{D}_R$. We illustrate this by analyzing when a policy's output was learned from real vs sim. We assume that predicting action $\mathbf{A}$ given observation $\mathbf{O}$ is learned from $\mathcal{D}_R$ if more of its nearest observation-action neighbors are in $\mathcal{D}_R$, and vice-versa for $\mathcal{D}_S$[2]. We

[2]Given pairs of observation and action horizons $(\mathbf{O}_1, \mathbf{A}_1)$ and $(\mathbf{O}_2, \mathbf{A}_2)$, we compute their distance as $\|\mathbf{A}_1 - \mathbf{A}_2\|_2 + \tau\|g(\mathbf{O}_1) - g(\mathbf{O}_2)\|_2$, where g is the observation embedding.

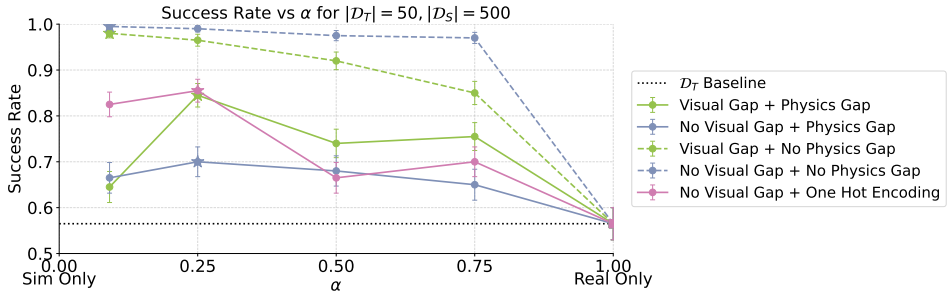


Figure 9: When a physics gap exists, removing visual gap harms performance. The opposite is true when a physics gap does not exist. This shows that visual gaps (or one-hot encodings) are important for enabling cotrained policies to identify the underlying physics of their deployment domain.

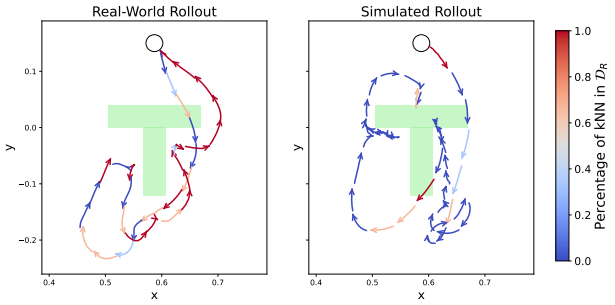


Figure 10: This plot visualizes when actions were learned from $\mathcal{D}_R$ (red) or $\mathcal{D}_S$ (blue). Each arrow is a chunk of actions and its color shows the percentage of its nearest neighbors in $\mathcal{D}_R$. The green T is the goal pose. The policy was trained with $|\mathcal{D}_R| = 50$, $|\mathcal{D}_S| = 500$, $\alpha = 0.75$.

roll out a cotrained policy twice: once in the real-world and once in sim. Figure 10 visualizes the percentage of the top-k nearest-neighbors from $\mathcal{D}_R$ for every chunk of predicted actions.

The real-world rollout is mostly red (learned from $\mathcal{D}_R$). The opposite is true for the simulation rollout. This result is notable since the policy was cotrained with 10x more sim data than real data; yet the majority of the nearest neighbors from the real-world rollout were still from $\mathcal{D}_R$. Figure 10 suggests that policies primarily rely on data from their deployment domain. Cotraining improves performance by providing actions when the policy encounters missing states from the deployment domain's dataset. This helps explain why performance plateaus even as $|\mathcal{D}_S|$ grows: once missing states in $\mathcal{D}_R$ are filled, additional simulated data provides diminishing returns. It also explains why policies cotrained with $|\mathcal{D}_R| = 10$ perform poorly: simulation can fill gaps, but we need real-world data to learn compatible high-level strategies and behaviors for the real-world physics.

Power Laws: Despite the *sim2target* gap, scaling $\mathcal{D}_S$ predictably decreases the *test loss* and *action mean squared error (MSE)* in the *target* domain. This is evidence for positive transfer in cotraining.

$$\mathcal{L}_{\mathcal{D}_T^{\text{test}}} \propto |\mathcal{D}_S|^{-0.33} \cdot |\mathcal{D}_T|^{-0.40},\ R^2 = 0.95; \quad \text{MSE}_{\mathcal{D}_T^{\text{test}}} \propto |\mathcal{D}_S|^{-0.29} \cdot |\mathcal{D}_T|^{-0.59},\ R^2 = 0.98 \quad (1)$$

Figure 15 visualizes the power laws in equation (1). These equations provide intuition for the relative impact of both datasets. For example, the exponents on $|\mathcal{D}_T|$ are larger in magnitude than $|\mathcal{D}_S|$. This aligns with our intuition that real data is more valuable than sim data. Although (1) is specific to our setup, the existence of power laws that accurately fit the performance is remarkable. Larger experiments are needed to determine if (1) are also *scaling laws* [39]. For instance, $\text{MSE}_{\mathcal{D}_T^{\text{test}}}$ appears to plateau for large $|\mathcal{D}_S|$. This could explain the performance plateaus in Section 5.2.

7 Conclusion

We present a thorough empirical study of sim-and-real cotraining for robot imitation learning. Our results show that scaling up simulated data generation and cotraining are effective strategies for improving policy performance. We investigate the effects of mixing ratios, data scales, and distribution shifts. Lastly, we analyze the mechanisms underlying cotraining and present results for 2 alternative formulations for cotraining in Appendix D. We hope this work offers valuable insights for sim-and-real cotraining in robot learning.

8 Limitations

All evaluations were conducted on planar-pushing from pixels. This allows us to be thorough, but limits the scope of our results. Nonetheless, we hope our findings are informative for cotraining. A larger-scale evaluation would require a massively multi-task data generation pipeline. This is an exciting direction for future work. Repeating the experiments with other imitation learning algorithms [30][40] and data generation pipelines [5][6] would also be valuable.

Acknowledgments

This work was supported by Amazon.com PO# 2D-06310236, PO# 2D-15431240, CC OTH 00596792 2021 TR/PO 2D-15694048, The AI Institute, the Toyota Research Institute PO-003816, ONR N00014-22-1-2121, the National Science Foundation Graduate Research Fellowship Program under Grant No. 2141064, and the Natural Sciences and Engineering Research Council of Canada CGS D-587703. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. We thank MIT Supercloud [41] for providing computational resources.

References

[1] G. Team. Gemini: A family of highly capable multimodal models, 2024. URL https://arxiv.org/abs/2312.11805.

[2] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners, 2020. URL https://arxiv.org/abs/2005.14165.

[3] C. Sun, A. Shrivastava, S. Singh, and A. Gupta. Revisiting unreasonable effectiveness of data in deep learning era, 2017. URL https://arxiv.org/abs/1707.02968.

[4] R. Firoozi, J. Tucker, S. Tian, A. Majumdar, J. Sun, W. Liu, Y. Zhu, S. Song, A. Kapoor, K. Hausman, B. Ichter, D. Driess, J. Wu, C. Lu, and M. Schwager. Foundation models in robotics: Applications, challenges, and the future, 2023. URL https://arxiv.org/abs/2312.07843.

[5] M. Dalal, A. Mandlekar, C. Garrett, A. Handa, R. Salakhutdinov, and D. Fox. Imitating task and motion planning with visuomotor transformers, 2023. URL https://arxiv.org/abs/2305.16309.

[6] H. Ha, P. Florence, and S. Song. Scaling up and distilling down: Language-guided robot skill acquisition, 2023. URL https://arxiv.org/abs/2307.14535.

[7] A. Mandlekar, S. Nasiriany, B. Wen, I. Akinola, Y. Narang, L. Fan, Y. Zhu, and D. Fox. Mimicgen: A data generation system for scalable robot learning using human demonstrations, 2023. URL https://arxiv.org/abs/2310.17596.

[8] E. Collaboration, A. O'Neill, A. Rehman, A. Gupta, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar, A. Jain, A. Tung, A. Bewley, A. Herzog, A. Irpan, A. Khazatsky, A. Rai, A. Gupta, A. Wang, A. Kolobov, A. Singh, A. Garg, A. Kembhavi, A. Xie, A. Brohan, A. Raffin, A. Sharma, A. Yavary, A. Jain, A. Balakrishna, A. Wahid, B. Burgess-Limerick, B. Kim, B. Schölkopf, B. Wulfe, B. Ichter, C. Lu, C. Xu, C. Le, C. Finn, C. Wang, C. Xu, C. Chi, C. Huang, C. Chan, C. Agia, C. Pan, C. Fu, C. Devin, D. Xu, D. Morton, D. Driess, D. Chen, D. Pathak, D. Shah, D. Büchler, D. Jayaraman, D. Kalashnikov, D. Sadigh, E. Johns, E. Foster, F. Liu, F. Ceola, F. Xia, F. Zhao, F. V. Frujeri, F. Stulp,

G. Zhou, G. S. Sukhatme, G. Salhotra, G. Yan, G. Feng, G. Schiavi, G. Berseth, G. Kahn, G. Yang, G. Wang, H. Su, H.-S. Fang, H. Shi, H. Bao, H. B. Amor, H. I. Christensen, H. Furuta, H. Bharadhwaj, H. Walke, H. Fang, H. Ha, I. Mordatch, I. Radosavovic, I. Leal, J. Liang, J. Abou-Chakra, J. Kim, J. Drake, J. Peters, J. Schneider, J. Hsu, J. Vakil, J. Bohg, J. Bingham, J. Wu, J. Gao, J. Hu, J. Wu, J. Wu, J. Sun, J. Luo, J. Gu, J. Tan, J. Oh, J. Wu, J. Lu, J. Yang, J. Malik, J. Silvério, J. Hejna, J. Booher, J. Tompson, J. Yang, J. Salvador, J. J. Lim, J. Han, K. Wang, K. Rao, K. Pertsch, K. Hausman, K. Go, K. Gopalakrishnan, K. Goldberg, K. Byrne, K. Oslund, K. Kawaharazuka, K. Black, K. Lin, K. Zhang, K. Ehsani, K. Lekkala, K. Ellis, K. Rana, K. Srinivasan, K. Fang, K. P. Singh, K.-H. Zeng, K. Hatch, K. Hsu, L. Itti, L. Y. Chen, L. Pinto, L. Fei-Fei, L. Tan, L. J. Fan, L. Ott, L. Lee, L. Weihs, M. Chen, M. Lepert, M. Memmel, M. Tomizuka, M. Itkina, M. G. Castro, M. Spero, M. Du, M. Ahn, M. C. Yip, M. Zhang, M. Ding, M. Heo, M. K. Srirama, M. Sharma, M. J. Kim, N. Kanazawa, N. Hansen, N. Heess, N. J. Joshi, N. Suenderhauf, N. Liu, N. D. Palo, N. M. M. Shafiullah, O. Mees, O. Kroemer, O. Bastani, P. R. Sanketi, P. T. Miller, P. Yin, P. Wohlhart, P. Xu, P. D. Fagan, P. Mitrano, P. Sermanet, P. Abbeel, P. Sundaresan, Q. Chen, Q. Vuong, R. Rafailov, R. Tian, R. Doshi, R. Mart'in-Mart'in, R. Baijal, R. Scalise, R. Hendrix, R. Lin, R. Qian, R. Zhang, R. Mendonca, R. Shah, R. Hoque, R. Julian, S. Bustamante, S. Kirmani, S. Levine, S. Lin, S. Moore, S. Bahl, S. Dass, S. Sonawani, S. Tulsiani, S. Song, S. Xu, S. Haldar, S. Karamcheti, S. Adebola, S. Guist, S. Nasiriany, S. Schaal, S. Welker, S. Tian, S. Ramamoorthy, S. Dasari, S. Belkhale, S. Park, S. Nair, S. Mirchandani, T. Osa, T. Gupta, T. Harada, T. Matsushima, T. Xiao, T. Kollar, T. Yu, T. Ding, T. Davchev, T. Z. Zhao, T. Armstrong, T. Darrell, T. Chung, V. Jain, V. Kumar, V. Vanhoucke, W. Zhan, W. Zhou, W. Burgard, X. Chen, X. Chen, X. Wang, X. Zhu, X. Geng, X. Liu, X. Liangwei, X. Li, Y. Pang, Y. Lu, Y. J. Ma, Y. Kim, Y. Chebotar, Y. Zhou, Y. Zhu, Y. Wu, Y. Xu, Y. Wang, Y. Bisk, Y. Dou, Y. Cho, Y. Lee, Y. Cui, Y. Cao, Y.-H. Wu, Y. Tang, Y. Zhu, Y. Zhang, Y. Jiang, Y. Li, Y. Li, Y. Iwasawa, Y. Matsuo, Z. Ma, Z. Xu, Z. J. Cui, Z. Zhang, Z. Fu, and Z. Lin. Open x-embodiment: Robotic learning datasets and rt-x models, 2024. URL https://arxiv.org/abs/2310.08864.

[9] D. Collaboration. Droid: A large-scale in-the-wild robot manipulation dataset, 2024. URL https://arxiv.org/abs/2403.12945.

[10] OpenAI and et. al. Solving rubik's cube with a robot hand, 2019. URL https://arxiv.org/abs/1910.07113.

[11] N. Rudin, D. Hoeller, P. Reist, and M. Hutter. Learning to walk in minutes using massively parallel deep reinforcement learning, 2022. URL https://arxiv.org/abs/2109.11978.

[12] L. Ankile, A. Simeonov, I. Shenfeld, M. Torne, and P. Agrawal. From imitation to refinement – residual rl for precise assembly, 2024. URL https://arxiv.org/abs/2407.16677.

[13] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 2024.

[14] J. Ho and T. Salimans. Classifier-free diffusion guidance, 2022. URL https://arxiv.org/abs/2207.12598.

[15] L. Yang, H. T. Suh, T. Zhao, B. P. Græsdal, T. Kelestemur, J. Wang, T. Pang, and R. Tedrake. Physics-driven data generation for contact-rich manipulation via trajectory optimization. *arXiv preprint arXiv:2206.10787*, 2025.

[16] M. Torne, A. Simeonov, Z. Li, A. Chan, T. Chen, A. Gupta, and P. Agrawal. Reconciling reality through simulation: A real-to-sim-to-real approach for robust manipulation, 2024. URL https://arxiv.org/abs/2403.03949.

[17] N. Pfaff, H. Dai, S. Zakharov, S. Iwase, and R. Tedrake. Steerable scene generation with post training and inference-time search, 2025. URL https://arxiv.org/abs/2505.04831.

[18] N. Pfaff, E. Fu, J. Binagia, P. Isola, and R. Tedrake. Scalable real2sim: Physics-aware asset generation via robotic pick-and-place setups, 2025. URL https://arxiv.org/abs/2503.00370.

[19] T. Marcucci, M. Petersen, D. von Wrangel, and R. Tedrake. Motion planning around obstacles with convex optimization, 2022. URL https://arxiv.org/abs/2205.04422.

[20] T. Chen, J. Xu, and P. Agrawal. A system for general in-hand object re-orientation, 2021. URL https://arxiv.org/abs/2111.03043.

[21] 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, 2017. URL https://arxiv.org/abs/1703.06907.

[22] N. Fey, G. B. Margolis, M. Peticco, and P. Agrawal. Bridging the sim-to-real gap for athletic loco-manipulation, 2025. URL https://arxiv.org/abs/2502.10894.

[23] A. Maddukuri, Z. Jiang, L. Y. Chen, S. Nasiriany, Y. Xie, Y. Fang, W. Huang, Z. Wang, Z. Xu, N. Chernyadev, S. Reed, K. Goldberg, A. Mandlekar, L. Fan, and Y. Zhu. Sim-and-real co-training: A simple recipe for vision-based robotic manipulation, 2025. URL https://arxiv.org/abs/2503.24361.

[24] S. Nasiriany, A. Maddukuri, L. Zhang, A. Parikh, A. Lo, A. Joshi, A. Mandlekar, and Y. Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots, 2024. URL https://arxiv.org/abs/2406.02523.

[25] W.-H. Li, X. Liu, and H. Bilen. Universal representation learning from multiple domains for few-shot classification, 2021. URL https://arxiv.org/abs/2103.13841.

[26] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. Vaughan. A theory of learning from different domains. *Machine Learning*, 79:151–175, 2010. URL http://www.springerlink.com/content/q6qk230685577n52/.

[27] S. M. Xie, H. Pham, X. Dong, N. Du, H. Liu, Y. Lu, P. Liang, Q. V. Le, T. Ma, and A. W. Yu. Doremi: Optimizing data mixtures speeds up language model pretraining, 2023. URL https://arxiv.org/abs/2305.10429.

[28] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, Q. Vuong, T. Kollar, B. Burchfiel, R. Tedrake, D. Sadigh, S. Levine, P. Liang, and C. Finn. OpenVLA: An open-source vision-language-action model, 2024. URL https://arxiv.org/abs/2406.09246.

[29] J. Hejna, C. Bhateja, Y. Jian, K. Pertsch, and D. Sadigh. Re-mix: Optimizing data mixtures for large scale imitation learning, 2024. URL https://arxiv.org/abs/2408.14037.

[30] S. Lee, Y. Wang, H. Etukuru, H. J. Kim, N. M. M. Shafiullah, and L. Pinto. Behavior generation with latent actions, 2024. URL https://arxiv.org/abs/2403.03181.

[31] C. Lynch, A. Wahid, J. Tompson, T. Ding, J. Betker, R. Baruch, T. Armstrong, and P. Florence. Interactive language: Talking to robots in real time, 2022. URL https://arxiv.org/abs/2210.06407.

[32] B. P. Graesdal, S. Y. C. Chia, T. Marcucci, S. Morozov, A. Amice, P. A. Parrilo, and R. Tedrake. Towards tight convex relaxations for contact-rich manipulation, 2024. URL https://arxiv.org/abs/2402.10312.

[33] A. Aydinoglu, A. Wei, W.-C. Huang, and M. Posa. Consensus complementarity control for multi-contact mpc, 2024. URL https://arxiv.org/abs/2304.11259.

[34] S. Kang, G. Liu, and H. Yang. Global contact-rich planning with sparsity-rich semidefinite relaxations, 2025. URL https://arxiv.org/abs/2502.02829.

[35] R. Tedrake and the Drake Development Team. Drake: Model-based design and verification for robotics, 2019. URL https://drake.mit.edu.

[36] T. Z. Zhao, J. Tompson, D. Driess, P. Florence, K. Ghasemipour, C. Finn, and A. Wahid. Aloha unleashed: A simple recipe for robot dexterity, 2024. URL https://arxiv.org/abs/2410.13126.

[37] M. N. Qureshi, S. Garg, F. Yandun, D. Held, G. Kantor, and A. Silwal. Splatsim: Zero-shot sim2real transfer of rgb manipulation policies using gaussian splatting, 2024. URL https://arxiv.org/abs/2409.10161.

[38] K. Rao, C. Harris, A. Irpan, S. Levine, J. Ibarz, and M. Khansari. Rl-cyclegan: Reinforcement learning aware simulation-to-real, 2020. URL https://arxiv.org/abs/2006.09001.

[39] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. Scaling laws for neural language models, 2020. URL https://arxiv.org/abs/2001.08361.

[40] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn. Learning fine-grained bimanual manipulation with low-cost hardware, 2023. URL https://arxiv.org/abs/2304.13705.

[41] A. Reuther, J. Kepner, C. Byun, S. Samsi, W. Arcand, D. Bestor, B. Bergeron, V. Gadepally, M. Houle, M. Hubbell, M. Jones, A. Klein, L. Milechin, J. Mullen, A. Prout, A. Rosa, C. Yee, and P. Michaleas. Interactive supercomputing on 40,000 cores for machine learning and data analysis. In *2018 IEEE High Performance extreme Computing Conference (HPEC)*, page 1–6. IEEE, 2018.

[42] T. Z. Zhao, S. Karamcheti, T. Kollar, C. Finn, and P. Liang. What makes representation learning from videos hard for control? RSS Workshop on Scaling Robot Learning, 2022.

[43] J. Song, C. Meng, and S. Ermon. Denoising diffusion implicit models, 2022. URL https://arxiv.org/abs/2010.02502.

[44] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models, 2020. URL https://arxiv.org/abs/2006.11239.

[45] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2015. URL https://arxiv.org/abs/1512.03385.

[46] H. Edwards and A. Storkey. Censoring representations with an adversary, 2016. URL https://arxiv.org/abs/1511.05897.

[47] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks, 2014. URL https://arxiv.org/abs/1406.2661.

[48] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13(25):723–773, 2012. URL http://jmlr.org/papers/v13/gretton12a.html.

A Real-World Experiments

A.1 Binary Probing Experiments

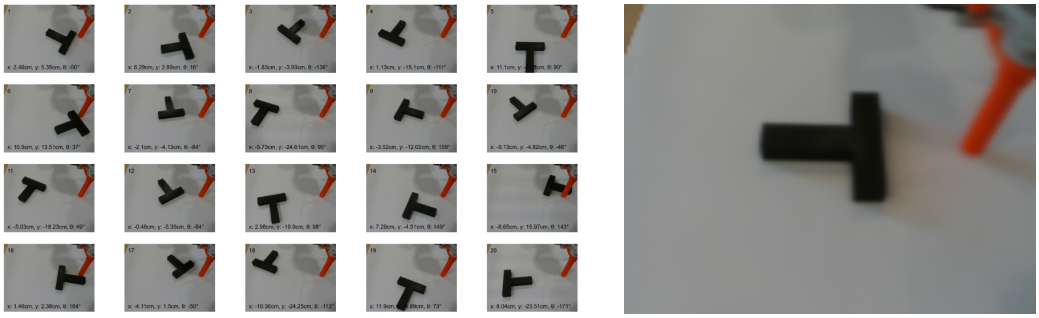
We add small binary probes to different layers of the cotrained policies to determine whether the learn representations contain information about the environment label. Concretely, each probe is a small 3 layer MLP with ReLU activations that is trained to classify an embeddings collected from both $\mathcal{D}_R$ and $\mathcal{D}_S$. Given a new embedding (from real or from sim), the classifier is asked to predict the probability that the embedding came from a "simulated" dataset. Table 1 shows that binary probes can reliably discern the robot's deployment environment from the embeddings alone, indicating that the cotrained policies have learned to distinguish sim-and-real.

Policy			**Binary Probe Location**	
$\lvert\mathcal{D}_R\rvert$	$\lvert\mathcal{D}_S\rvert$	α	Observation Embedding	Final Activation
50	500	0.75	100%	74.2%
50	2000	0.75	100%	89%
10	500	0.75	100%	84%
10	2000	5e-3	100%	93%

Table 1: The environment classification accuracy for binary probes at different layers of several cotrained policies.

A.2 Evaluation

For each policy, we report the performance of the best checkpoint over the 20 initial conditions shown in Figure 11. A trial is successful if a human teleoperator would not readjust the final slider pose. The time limit is 90s. We plot standard error (SE) bars, calculated as $\sqrt{p(1-p)/n}$, where p is the empirical success rate and n is the number of trials. Note that these error bars capture the performance distribution of the best checkpoint, but *do not* capture variability due to stochasticity in training.



(a) (b)

Figure 11: a) Initial conditions for the real-world trials. b) An overlay of the final state for the 160 successful demos. We use the same success criteria to evaluate the policy rollouts. The overlay is fairly noiseless, illustrating the strictness of our evaluations.

B Simulation Experiments

B.1 Sim2target vs sim2real gap

Figure 12 visualizes the sim2real vs sim2*target* gap. Table 2 explicitly outlines the differences between the simulated and *target* environments (which were designed to mimic the sim2real gap in the real-world experiments).

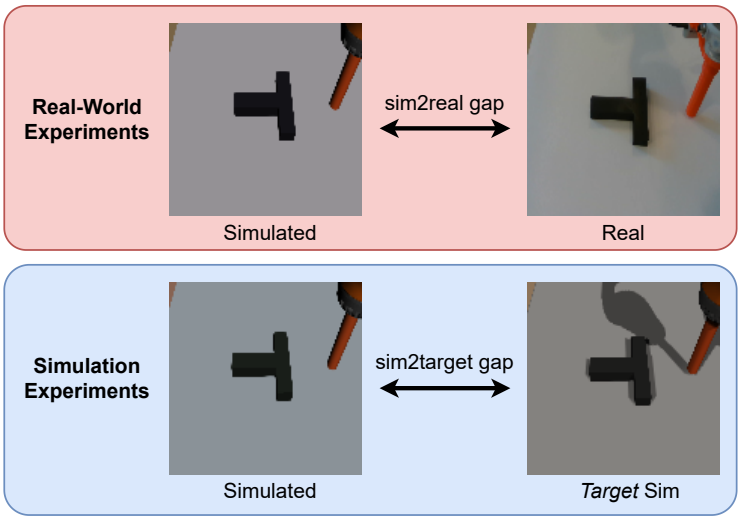


Figure 12: A visualization and comparison of the *sim2real* gap in Section 4 and the *sim2target* gap in Section 5

Gap	Simulation	Real World	*Target* Sim
Visual	white light, camera & color offset	natural light, shadows	warm light, shadows
Physics	Quasistatic	Real physics	`Drake` physics
Action	Motion planning	Human teleop	Human teleop

Table 2: The *sim2target* gap was designed to mimic the *sim2real* gap along 3 different axes.

B.2 Performance vs $|\mathcal{D}_T|$

Performance of the best cotrained policies and *target*-only policies improve with additional *target* data. Figure 13 compares the performance scaling curves as more *target* data is added. The results show that cotraining with simulate data is equivalent to scaling the amount of *target* sim data by 2-3x.

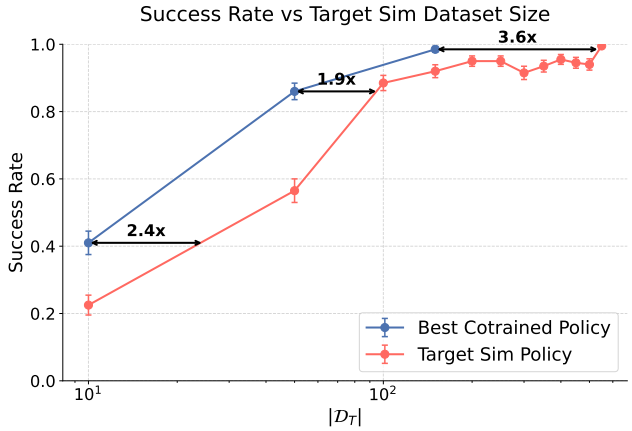


Figure 13: The performance of the best cotrained policies are equivalent to increasing the amount of data from the target domain by 2–3×.

B.3 Distribution Shift Experiments

The 6 distribution shifts explored in Section 5.3 were partially inspired by [42]. Calibrating the *magnitude* of the visual shifts with the physical shifts is difficult since their units differ, but we have attempted to make both distributions reasonably representative. Let $\mathcal{C} = [0, 1]^3$ be the space of RGB colors, $B_r(\boldsymbol{c})$ be a ball of radius r centered at $\boldsymbol{c}$, and $\boldsymbol{c}_i \in \mathcal{C}$ be the true color of object i.

Color Mean Shift: Object colors are shifted as $\boldsymbol{c}_i \leftarrow \boldsymbol{c}_i + \gamma \boldsymbol{u}_i + \boldsymbol{o}_i$, where $\boldsymbol{u}_i$'s are unit vectors. $\boldsymbol{o}_i = -0.05 \cdot \mathbf{1}$ for the slider and $\mathbf{0}$ otherwise. We sweep $\gamma \in \{0, 0.05, 0.2, 0.5\}$.

Color Randomization: Object colors are sampled uniformly from $B_r(\boldsymbol{c_i}) \cap \mathcal{C}$. We sweep $r \in \{0.025, 0.1, 0.5, \sqrt{3}\}$. Note that $B_{\sqrt{3}}(\boldsymbol{c_i}) \cap \mathcal{C} = \mathcal{C}$.

Camera Shift: We fix a frame, F, near the slider's target pose and rotate the camera about the z-axis of F by $\beta \in \{0, -\frac{\pi}{16}, -\frac{\pi}{8}, -\frac{\pi}{4}\}$.

Center of Mass (CoM) Shift: We generate data with incorrect CoM: $y_{\text{CoM}} \leftarrow y_{\text{CoM}} + y_{\text{offset}}$. We sweep $y_{\text{offset}} \in \{0, 3, -3, -6\}$cm. The slider is 16.5cm tall.

Goal Shift: We generate demos that push the slider to the incorrect goal pose: $y_{\text{goal}} \leftarrow y_{\text{goal}} + g_{\text{offset}}$. We sweep $g_{\text{offset}} \in \{0, -2.5, -5, -10\}$cm.

Object Shift: We generate a planar pushing dataset containing 6 objects, none of which are the T.

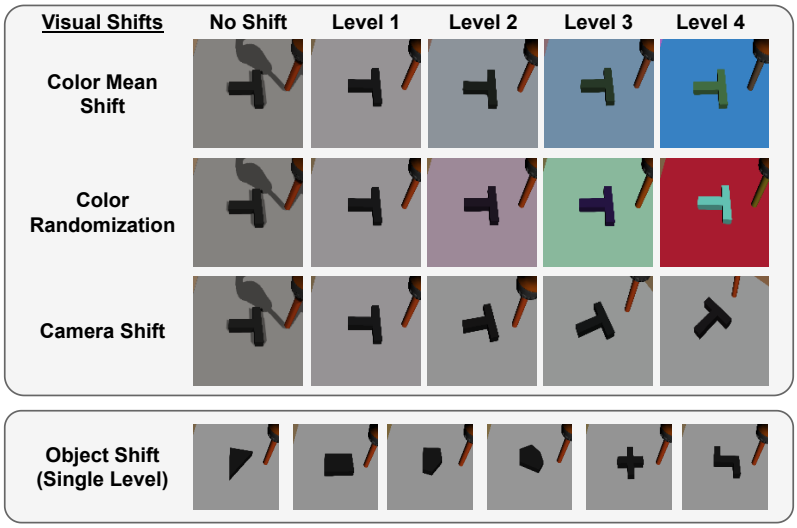


Figure 14: The upper box visualizes the 3 types of visual shifts over 4 intensity levels. The bottom box visualizes the sliders used for object shift.

B.4 Power Laws

Our results show that cotraining with data from the simulated environment reduces performance metrics, such as test loss and test action MSE, on the *target* sim environment according to a power law. These power laws are visualized in Figure 15.

B.5 Evaluation

A trial is successful if the robot returns to its default position and the slider's pose is within 1.5cm and 3.5° of the goal. The time limit is 75s. For each policy, we evaluate all checkpoints for up to 200 trials. During evaluation, we periodically discard low performing checkpoints according to a Bayesian framework to save compute. We report the performance of the best checkpoints and compute SE bars. Both our *target sim* evaluation set up and simulated data generation scripts are available here: **link redacted for anonymous review**.

C Training Details

In this work, we focus our study on the Diffusion Policy [13] paradigm for behavior cloning. Diffusion Policies sample future robot actions given a history of past observations to accomplish a desired task [13]. More concretely, they learn a denoiser for the conditional action distribution $p(\mathbf{A}|\mathbf{O})$ by

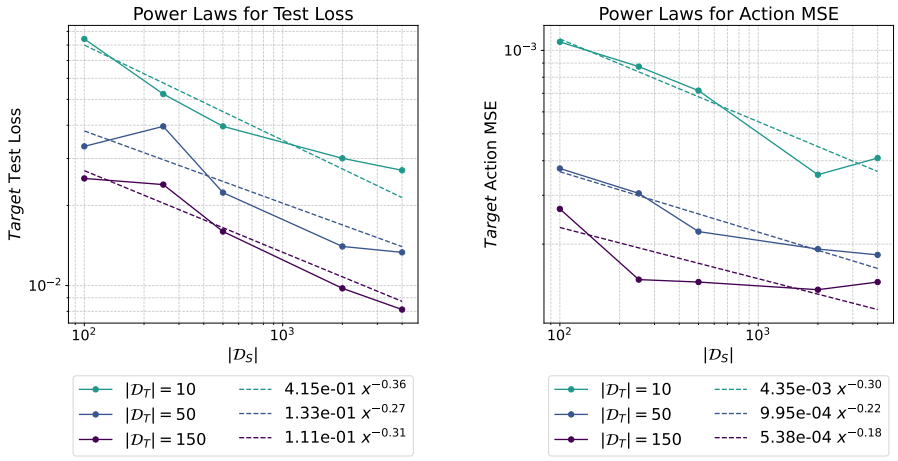


Figure 15: Log-log plots of the *target* test loss and action MSE as a function of $|\mathcal{D}_S|$. We report values for all the checkpoints in Figure 5 that were trained with the natural mixing ratio. Note that the plots exhibit an approximate power law.

minimizing the loss in (2). $\mathbf{A}$ and $\mathbf{O}$ are *horizons* of actions and observations, ρ_t and σ_t are parameters of the noise schedule, and θ are the parameters of the denoiser. We sample from the learned policy, $\pi_\theta(\mathbf{A}|\mathbf{O})$, by iteratively applying ϵ_θ in a denoising process [43][44].

$$\mathcal{L}_\mathcal{D}(\theta) = \mathbb{E}_{t,(\mathbf{O},\mathbf{A})\sim\mathcal{D},\epsilon\sim\mathcal{N}(0,I)}[\|\epsilon - \epsilon_\theta(\rho_t\mathbf{A} + \sigma_t\epsilon,\ \mathbf{O},\ t)\|_2^2] \tag{2}$$

Our experiments train Diffusion Policies since they are a state-of-the-art algorithm for imitation learning [13]. Policies were trained using a fork of the code provided by [13]. We re-used most hyperparameters but performed a small sweep to select the batch size, learning rate, and number of optimizer steps. We used their U-Net architecture with a ResNet18 backbone [45] and trained end-to-end.

To ensure a fair comparison, we trained each model for approximately the same number of optimizer steps and the same hyperparameters. We saved 4-5 checkpoints along the way. For our finetuning experiments, we reduced the learning rate by 10x and reported performance for the best checkpoint. The configuration files for all training runs are available here: **link redacted for anonymous review**.

D Alternative Cotraining Formulations

We explore 2 ablations inspired by our analysis: one that artifically *amplifies* the sim2real gap (Section D.1), and another that *reduces* the sim2real gap at the representation level (Section D.2).

D.1 Classifier-Free Guidance Ablation

If high-performing policies predict different actions in real vs sim (see Section 6.1), can we improve performance by amplifying this action gap with *classifier-free guidance* (CFG) [14]? To study this question, we add a one-hot encoding of the environment to the policies and cotrain with CFG. More concretely, we sample with the denoiser in (3) which is *guided* by the environment encoding, c.

$$\tilde{\epsilon}(\mathbf{A}_t, \mathbf{O}, c, t) = (1+w)\epsilon_\theta(\mathbf{A}_t, \mathbf{O}, c, t) - w\epsilon_\theta(\mathbf{A}_t, \mathbf{O}, \varnothing, t). \tag{3}$$

CFG artificially amplifies the action gap at sampling-time; w controls the magnitude of this effect. Note that $w = 0$ is equivalent to regular sampling with a one-hot encoding. Figure 16 shows that amplifying the action gap with CFG and $w > 0$ does not improve performance over vanilla cotraining. On the other hand, preformance increases for $w = 0$. This suggests that practitioners should cotrain policies with a *one-hot encoding of the environment* and sample regularly [13].

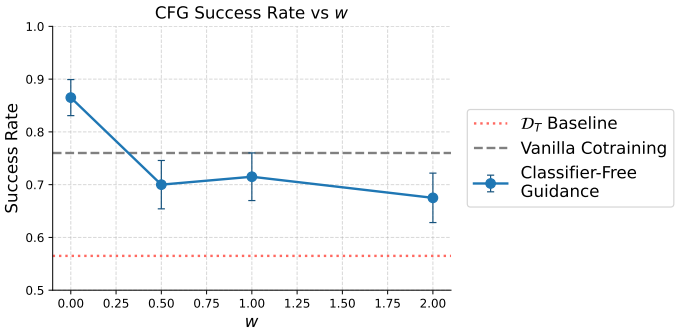


Figure 16: Performance of cotraining with CFG for $|\mathcal{D}_T| = 50$, $|\mathcal{D}_S| = 2000$, $\alpha = 0.25$. The plots are similar for all data scales and mixtures. We present only one for clarity.

D.2 Alternative Cotraining Loss

We showed that smaller sim2real gaps improve performance, but high-performing policies must distinguish sim from real. Given these observations, we explored an alternative loss for cotraining that *reduces* the sim2real gap at the policy's "representation level" [46], while maintaining its ability to identify relevant domain-specific information.

More concretely, given a cotrained policy π_θ, let p_θ^S and p_θ^R be distributions over the learned embeddings for sim and real respectively. Our alternative loss adds a second term to the training objective, $\text{dist}(p_\theta^S, p_\theta^R)$, that penalizes the differences in the learned representations for sim and real.

$$\min_\theta \mathcal{L}_{\mathcal{D}^\alpha}(\theta) + \lambda \cdot \text{dist}(p_\theta^S, p_\theta^R). \tag{4}$$

The second term in (4) encourages the policy to ignore sim2real gaps that are irrelevant for control (ex. differences in lighting and texture). The first term encourages the policy to retain any information that is important for minimizing $\mathcal{L}_{\mathcal{D}^\alpha}$ (ex. the physics of the domain). The two competing objectives are weighted by λ.

We explore two choices for the dist function. Here, $\mathcal{L}_{\text{BCE}}$ is the binary cross entropy loss:

$$\begin{aligned} \text{Adversarial} : \text{dist}(p_\theta^S, p_\theta^R) &= \max_\phi -\mathcal{L}_{\text{BCE}}(\phi, \theta) \\ \text{MMD} : \text{dist}(p_\theta^S, p_\theta^R) &= \text{MMD}(p_\theta^S, p_\theta^R) \end{aligned} \tag{5}$$

The adversarial formulation aims to learn a representation for both sim and real that cannot be reliably distinguished by a classifier, d_ϕ. This objective is minimized with a GAN-like training algorithm [46][47]. The MMD formulation aims to minimize the *maximum mean discrepancy* (MMD) between the learned representations [48]. We use a mixutre of Gaussian kernels for the MMD kernel and minimize (4) via differentiation. We swept the λ hyperparameter for the MMD loss, but not the adversarial loss since the latter encountered the typical GAN-like training instabilities.

The performance of policies trained with (4) on both our real-world and *target sim* environments are presented in Figure 17. The alternative loss formulations do not reliably outperform vanilla cotraining. This is consistent with our observation that sim-and-real discernability is important for performance.

Nonetheless, we chose to present these negative results for two reasons. First, the difference in performance between the adversarial and MMD formulations show that the choice of the dist function can dramatically alter the results. Thus, there could be unexplored choices for dist that outperform the vanilla cotraining baseline. Second, we believe the more general idea of reducing the sim2real gap at the representation level (as opposed to the simulation level) is an interesting direction for future work.

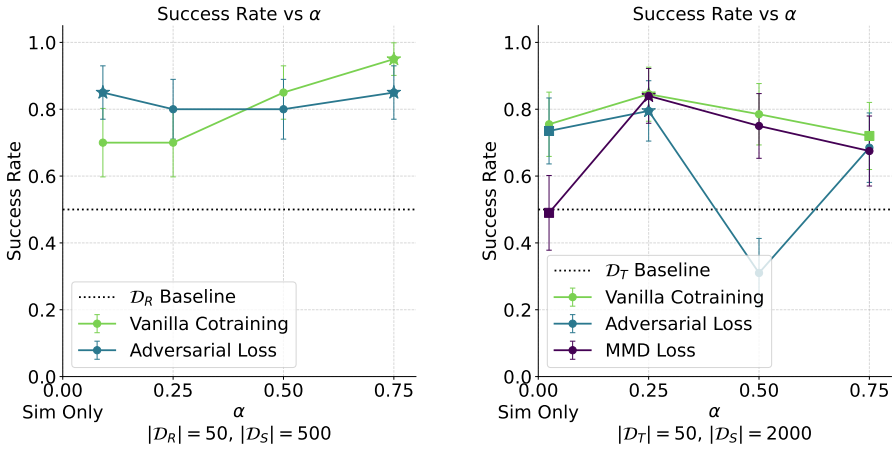


Figure 17: Performance of cotrained policies trained using the vanilla cotraining loss, adversarial loss, and MMD loss. Adding an MMD or GAN-like adversarial objective did not reliably improve performance in our experiments.