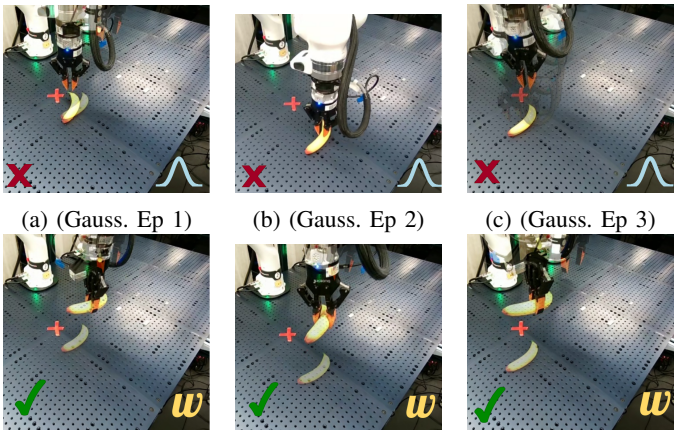
You've Got a Golden Ticket: Improving Generative Robot Policies With A Single Noise Vector

Omkar Patil[1,2], Ondrej Biza[1], Thomas Weng[1], Karl Schmeckpeper[1], Wil Thomason[1]
Xiaohan Zhang[1], Robin Walters[1,3], Nakul Gopalan[2], Sebastian Castro[1], Eric Rosen[1]
[1]Robotics and AI Institute (RAI) [2]Arizona State University (ASU) [3]Northeastern University

***Abstract*—What happens when a pretrained generative robot policy is provided a constant initial noise as input, rather than repeatedly sampling it from a Gaussian? We demonstrate that the performance of a pretrained, frozen diffusion or flow matching policy can be improved with respect to a downstream reward by swapping the sampling of initial noise from the prior distribution (typically isotropic Gaussian) with a well-chosen, constant initial noise input—a *golden ticket*. We propose a search method to find golden tickets using Monte-Carlo policy evaluation that keeps the pretrained policy frozen, does not train any new networks, and is applicable to all diffusion/flow matching policies (and therefore many VLAs). Our approach to policy improvement makes no assumptions beyond being able to inject initial noise into the policy and calculate (sparse) task rewards of episode rollouts, making it deployable with no additional infrastructure or models. Our method improves the performance of policies in 38 out of 43 tasks across simulated and real-world robot manipulation benchmarks, with relative improvements in success rate by up to 58% for some simulated tasks, and 60% within 50 search episodes for real-world tasks. We also show unique benefits of golden tickets for multi-task settings: the diversity of behaviors from different tickets naturally defines a Pareto frontier for balancing different objectives (e.g., speed, success rates); in VLAs, we find that a golden ticket optimized for one task can also boost performance in other related tasks. We release a codebase with pretrained policies and golden tickets for simulation benchmarks using VLAs, diffusion policies, and flow matching policies: https://bdaiinstitute.github.io/lottery_tickets/**



(a) (Gauss. Ep 1) (b) (Gauss. Ep 2) (c) (Gauss. Ep 3)

(d) (G.T. Ep 1) (e) (G.T. Ep 2) (f) (G.T. Ep 3)

Fig. 1: (a-c) A diffusion policy trained to pick a banana across the table, with three different failure spots shown. Every time it produces an action chunk, random initial noise is sampled from an isotropic Gaussian. (d-f) **With the same network and weights**, we can adapt this policy to successfully pick the banana from all locations, by instead using a constant initial noise vector called a *golden ticket (G.T.)*. We optimize initial noise vectors to steer pretrained policies to maximize downstream rewards.

I. Introduction

Conditional diffusion [8] and flow matching models [14] are popular approaches for learning robot control policies. Their ability to represent high-dimensional, multimodal action distributions have made them popular in both single-task policies and large-scale multi-task Vision-Language-Action (VLA) models. However, improving their out-of-the-box performance on downstream tasks currently introduces many computational and model design challenges. Existing policy improvement methods involve either 1) updating the pretrained model weights [26] (which is computationally hard for large models like VLAs), 2) training an additional network [32, 36] (requiring complex model design choices), or 3) assuming access to external critic networks [37] or specific training regime [10] (limiting the scenarios they can be applied in).

Instead, we propose a new method that 1) keeps the pretrained policy weights frozen, 2) does not train an additional network, and 3) can be applied to any diffusion or flow matching framework without additional training or test-time assumptions. Our approach is motivated by a simple question: What happens when a pretrained generative robot policy is provided a constant initial noise as input, rather than repeatedly sampling from a Gaussian?

We demonstrate that the performance of a pretrained, frozen diffusion or flow matching policy can be improved by simply replacing the sampling of initial noise from the prior distribution with a well-chosen, constant initial noise input, which we call a *golden ticket* (Figure 1). This approach is related to the emerging class of latent steering approaches [36], but benefits from only requiring a reward function for a downstream task, without any additional model training. Given a pretrained policy and a reward function and an environment for a new downstream task, we pose finding golden tickets as a search problem, where candidate initial noise vectors, which we call lottery tickets, are optimized to find the ticket that maximizes cumulative discounted expected rewards on the downstream task. We show that golden tickets can be competitive with state-of-the-art Reinforcement Learning (RL)-based latent steering methods [36], in some cases outperforming

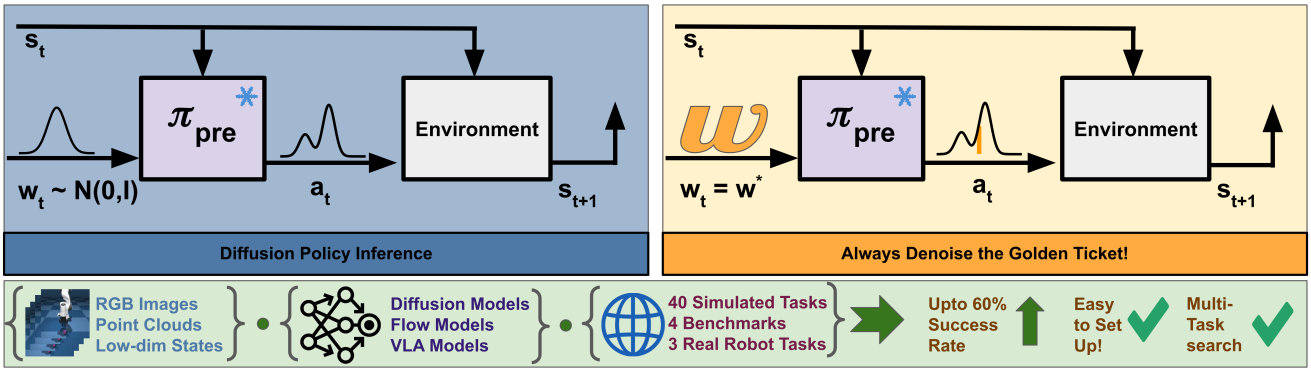


Fig. 2: Overview of standard diffusion policy inference (left) versus our proposed approach of using golden tickets (right). Given a frozen, pretrained diffusion or flow matching policy π_{pre}, rather than sampling from a Gaussian every time an action needs to be computed, we use a constant, well-chosen initial noise vector w, called a golden ticket. We find golden tickets improve policy performance across a range of observation inputs, model architectures, and embodiments.

them in terms of improved success, despite requiring less computational effort and fewer model design choices. Further, we show that golden tickets can exhibit properties favorable to skill learning, such cross-task generalization in goal-conditioned policies like VLAs, and that single-task policies can also be steered to balance competing objectives like speed and success rate.

Our experiments show an improvement in policy performance in 38 out of 43 tasks across four simulated manipulation benchmarks and 3 real-world tasks with a variety of pretrained policy classes. When we apply our approach on hardware, we increase the relative success rate of an object picking task by 18%, and an object pushing policy by 60%, with the ticket search taking less than 150 episodes per task.

Our contributions are:

1) A formulated lottery ticket hypothesis for improving pretrained robot policies by using a single initial noise vector—a *golden ticket*—to adapt robot behavior for downstream reward functions (Figure 2).
2) A search method to finding golden tickets that maximize rewards in a given robot environment.
3) Demonstrating that golden tickets improve policy performance across several model architectures, observation input modalities, and embodiments, in four simulated benchmarks and three real-world picking and pushing tasks.
4) Experiments demonstrating that golden tickets can match, and even outperform, existing state-of-the-art methods for latent steering in five simulated tasks, and evidence of cross-task generalization for goal-conditioned policies in three task suites comprising 30 related tasks.
5) Open-source code with golden tickets (and ticket search implementations) for pretrained VLA, diffusion, and flow matching policies in three simulated manipulation benchmarks.

II. Background

A. Markov Decision Process

We model robot manipulation as a Markov Decision Process (MDP). An MDP comprises a tuple $\{S, A, T, R, p_0, \gamma\}$, where S and A are sets of states and actions (respectively), $T(s' \mid s, a) = \Pr(s' \mid s, a)$ is the transition dynamics, $R(s, a)$ is the reward function, $p_0 \in \triangle_S$ is the initial state distribution over S, and γ is the discount factor. A policy π rolls out episodes from an initial state $s_0 \sim p_0$ by iteratively sampling an action $a_t = \pi(*|s_t)$ to execute at each time step t to get the next state $s_{t+1} \sim T(*|s_t, a_t)$ and reward $r_t = R(s_t, a_t)$. The Q-function $Q^\pi(s, a)$ for π gives the cumulative discounted expected rewards of π rolled out from a state s with initial action a, following π for all subsequent actions. That is, $Q^\pi(s, a) = \mathbb{E}_{P^{\pi,T}(*)}[\sum_{t>0} \gamma^t R(s_t, a_t) \mid s_0 = s, a_0 = a]$.

B. Diffusion and flow matching

Diffusion and flow matching models are generative methods that iteratively refine a sample from a source distribution into a desired target data distribution. We focus on the flow matching framework with a Gaussian source distribution for simplicity, but our work is also applicable to diffusion models and more broadly latent variable generative models (e.g., conditional variational auto-encoders (CVAEs) [41]).

In flow matching, given a dataset $D = \{x_i \sim p_{\text{data}}\}_{i=1}^n$, the forward process gradually corrupts a datapoint $x \in D$ over time τ[1] by linearly interpolating it with a sample from an isotropic Gaussian:

$$z_{\tau+1} = (1 - \tau)x + \tau\epsilon \quad \text{where } \epsilon \sim \mathcal{N}(0, I) \tag{1}$$

where z_τ represents the partially noised datapoint x at time τ. At time $\tau = 0$, $z_\tau = z_0 = x$ is uncorrupted, while at time $\tau = 1$, $z_\tau = z_1 = \epsilon$ is fully corrupted noise.

Generating new samples from the data distribution p_{data} requires reversing the forward process. Specifically, a

[1]Note that we use τ for the forward/reverse process time index to distinguish it from the MDP time index t, introduced in Section II-A.

denoising model $\hat{u} = \hat{u}(z_\tau, \tau)$ is trained to undo the forward process, and new samples from p_{data} can be generated by iteratively denoising a sample z_τ with $z_1 \sim \mathcal{N}(0, I)$, treating the denoising model $\hat{u}$ as a velocity field and solving the ODE:

$$z_j = z_k + \hat{u} \cdot (j - k) \tag{2}$$

While diffusion and other flow variants differ in how they define the forward and reverse processes, our method is agnostic to these variations; it focuses only on making z_1, the initial noise vector used for inference, a constant vector rather than repeatedly sampling it from a Gaussian distribution.

A *conditional* denoising model $\hat{u} = \hat{u}(z_k, s_t, \tau)$ takes in auxiliary state information s_t to adjust the denoising process (the process above can be trivially extended to this setting). When $\hat{u}$ is trained to denoise actions (or action chunks [41]) conditioned on a state s_t, the samples generated from the reverse process are interpreted as actions sampled from a policy π parameterized by the conditional denoising model $\hat{u}$ and the source noise distribution.

III. Related Work

We review existing approaches to policy improvement via latent steering, as well as related lottery ticket hypotheses from the computer vision community on optimizing initial noise vectors to improve performance of diffusion models.

A. Improving Robot Policies via Latent Steering

While there is a diverse range of policy improvement methods (see Appendix A for more details), we focus on those that use latent steering, since they most closely relate to our approach. Latent steering was pioneered by diffusion steering via reinforcement learning (DSRL) [36], which swaps the source noise distribution with an observation-conditioned noise policy trained via reinforcement learning (RL). DSRL freezes the weights of the pretrained diffusion policy, providing a strong regularization towards the behaviors already encoded in the pretrained model, and can be applied to any diffusion or flow matching policy. However, DSRL presents two major challenges in practice: 1) it trains an additional neural network, which involves modeling decisions to ensure sufficient model capacity, and 2) it integrates with an RL framework to train the noise policy, which requires hyperparameter tuning [36].

DSRL has inspired a wave of new approaches that investigate latent steering for policy improvement [13], but these all use observation-conditioned noise policies. We introduce *observation-independent* noise policies (i.e., using a single initial noise vector to perform latent steering), which: 1) do not require designing and training a separate network from the original policy, and 2) are not impacted by shifting observation distributions since they are by design invariant to observation.

B. Lottery Ticket Hypothesis for Diffusion/Flow Models

Our work is motivated by a "lottery ticket hypothesis" proposed by Mao et al. [20] in the context of image generation[2]: "randomly initialized Gaussian noise images contain special pixel blocks (winning tickets) that naturally tend to be denoised into specific content independently." Followup works have investigated how these "winning tickets" (also referred to as "golden noises" [42, 1, 21], or "golden tickets" in our work) can be optimized to achieve higher text-image alignment and higher human preference when they are used instead of sampling from isotropic Gaussian noise. Methods to find golden tickets for image generation have used a variety of metrics, such as noise inversion [29] and stability [25], semantic and natural appearance consistency [30], 3D geometry [27], and regularization to the original Gaussian distribution [35].

[2]The term "lottery ticket hypothesis" was first introduced by Frankle and Carbin [4] in the context of finding sparse subnetworks within denser networks, though this does not relate to our work.

IV. Lottery Ticket Hypothesis for Robot Control

To our knowledge, **we are the first to make a connection between lottery ticket hypotheses for improving text-to-image diffusion models and latent steering for improving robot diffusion/flow matching policies**. There are important differences between the robot control and image generation settings that require the development of new methods to optimize initial noise vectors (see Appendix B for details). These include 1) a robot policy interacts with an environment where previous decisions impact future states, 2) collecting samples from the environment can be costly, and 3) the reward function may not be differentiable. To overcome these challenges, we propose a unique lottery ticket hypothesis for improving robot policies that is based in reinforcement learning, along with a search method that addresses the three aforementioned challenges to find golden tickets that optimize rewards in a downstream environment.

A. Problem setting

We assume we are given a pretrained, frozen diffusion or flow matching robot control policy π. We treat this policy as a black box and do not assume we can access any intermediate steps of the denoising process. We also assume we are given an MDP representing a downstream task from which we can sample transitions (i.e., we can simulate the MDP but do not have access to the underlying model). Our goal is to adapt the policy π behavior to maximize rewards while keeping the parameters frozen. While we do not make any assumptions about the pretrained policy, our method only works if the policy is steerable [36], which is not guaranteed.

B. Lottery Ticket Hypothesis

> *The lottery ticket hypothesis for robot control* proposes that the performance of a pretrained, frozen diffusion or flow matching policy can be improved by replacing the sampling of initial noise from a prior distribution with a well-chosen, constant initial noise input, called a *golden ticket*.

Well-chosen means the ticket is optimized to steer the pretrained policy to maximize rewards for a downstream task.

Algorithm 1 Random Search for Optimizing Initial Noise

Input: Pretrained policy π, set of environments E, reward function R, number of tickets n
Output: Optimized initial noise vector $w^\star$

Sample n initial noise vectors $\{w_i\}_{i=1}^n$, $w_i \sim \mathcal{N}(0, I)$
Initialize best reward $\bar{R}_{\text{best}} \leftarrow -\infty$
for $i = 1$ to n **do**
    Initialize cumulative reward $S_i \leftarrow 0$
    for all $e \in E$ **do**
        Roll out π in environment e with fixed noise w_i
        Obtain episode return R_e
        $S_i \leftarrow S_i + R_e$
    Compute average episodic returns $\bar{R}_i \leftarrow \frac{1}{|E|} S_i$
    if $\bar{R}_i > \bar{R}_{\text{best}}$ **then**
        $\bar{R}_{\text{best}} \leftarrow \bar{R}_i$
        $w^\star \leftarrow w_i$
return $w^\star$

C. Golden ticket search

To improve the policy, we must find a ticket that has better performance than repeatedly sampling from the Gaussian prior. We propose a variant of *random search* (Algorithm 1) that estimates Monte-Carlo returns on policy rollouts to find golden tickets. Our approach only assumes that initial noise can be injected into the pretrained policy, and that (sparse) task rewards can be calculated for rollouts. This approach has many appreciable benefits: 1) it does not require setting up any additional infrastructure or running/training models, and 2) it is applicable to all diffusion and flow matching policies that can be ran as black-box systems, which includes many VLAs. In this paper, we therefore focus on (and now describe) our approach to demonstrate one way to search for golden tickets, and discuss future work improvements in Section VII.

Algorithm 1 shows our random search method for finding golden tickets. The algorithm first samples n initial noise vectors (this is the main hyperparameter of our method), which we refer to as lottery tickets, and then evaluates each lottery ticket in a set of search environments E by performing policy rollouts. Each search environment is formulated as a single MDP; for single-task policies, only the starting state distribution may differ between search environments, whereas for multi-task policies, the reward function may also differ between environments.

For each ticket, we run the policy in each environment, fixing the initial noise vector fed into the policy. We then calculate the average cumulative discounted rewards, providing an empirical estimate (i.e., Monte-Carlo estimate) of the value function for the ticket. After all tickets have been evaluated, we select and return the best performing ticket.

Note that this algorithm does not guarantee that the ticket returned is a golden ticket (i.e., that it has higher average performance than sampling from the Gaussian prior). To determine if the returned ticket is "golden", we assume there is a held-out set of initial states or environments which we use to evaluate the performance of both the best ticket and the base policy. We do this to avoid test/train contamination, because it is possible that the tickets we find during search are overfit to the particular environments they were searched in. If a ticket has good performance on the search environments, but worse performance on the held-out environments, we are overfitting and not meaningfully able to fix the initial noise to control the performance of the policy.

For a fixed compute budget, there exists a trade-off between number of tickets n and number of search environments E on which to evaluate those tickets. More tickets result in better coverage and potentially better search performance, but risk not generalizing to held-out environments. Conversely, more search environments increase the likelihood a ticket keeps its search performance on a held-out environment, but overall performance may degrade since fewer tickets are assessed.

V. Experiments

Our experimental evaluation assesses the hypothesis that fixing the initial noise vector can improve the performance of latent variable generative models. We first present the 4 simulated manipulation benchmarks (constituting 40 different tasks) and 3 real-world tasks that we use for our hardware experiments, and then detail our experimental questions.

A. Experimental benchmarks

We investigate a wide range of model architectures, observation inputs, and environments. We consider 4 simulation benchmarks (Figure 3) and three real-world hardware benchmarks (Figure 4). More details are Appendix C and D.

1) Flow matching policy in franka_sim: We use the franka_sim MuJoCo environment that was originally released in Luo et al. [17]. The task involves a single-arm Franka robot picking a cube, which randomly spawns in a $\frac{1}{2}$ square meter region in front of the robot. We train a flow matching policy that takes as input the low-dimensional state of the cube and robot. We use a task-and-motion planning heuristic to generate expert demonstrations to train the policy via behavior cloning. **This benchmark lets us examine golden tickets in a regime where we have the ability to easily generate varied datasets.**

2) SmolVLA in LIBERO*:* We use the LIBERO benchmark [15], which contains multiple manipulation task suites related to varying object layouts, object types, and goals. We use SmolVLA [31], an open-source VLA from HuggingFace, which operates on images, robot state, and language. (Specifically, the publicly released LIBERO-finetuned checkpoint). We evaluate on 3 task suites, each of which consists of 10 tasks: LIBERO-OBJECT, LIBERO-GOAL, and LIBERO-SPATIAL. **This benchmark lets us evaluate whether golden tickets exist in VLAs and multi-task policies.**

3) DPPO in robomimic: We use the same robomimic benchmark [19] and set of pretrained policies that were used in the original DSRL work [36]. Specifically, we evaluate diffusion policies that were trained using DPPO [26] on 4 separate tasks: `lift`, `can`, `square`, and `transport`. These policies take as input low-dimensional state information about

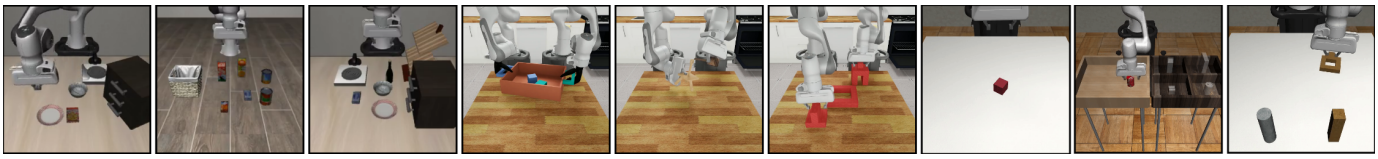
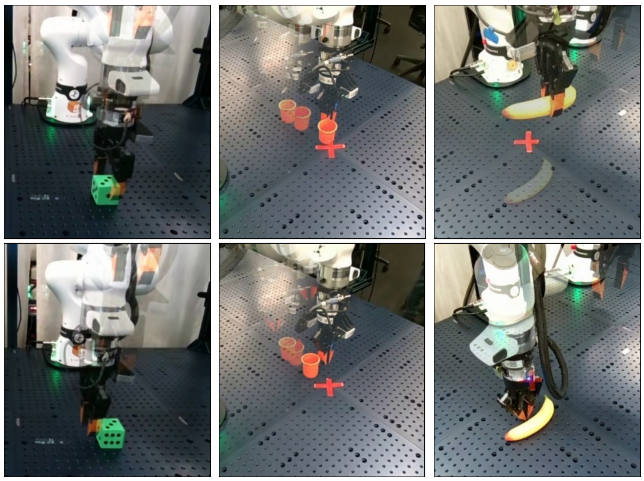


Fig. 3: Sample images from some of our simulated benchmarks: (1-3) LIBERO-OBJECT, LIBERO-SPATIAL, and LIBERO-GOAL task suites, (4-6) DexMimicGen `Tray Lift`, `Threading`, and `Piece Assembly` tasks, and (7-9) robomimic `lift`, `can` and `square` tasks.



(a) Pick cube (b) Push cup (c) Pick banana

Fig. 4: Rollouts from diffusion policies sampling with Gaussian noise that we use in our hardware experiments. (top) An example successful rollout (bottom) An example failed rollout.

the state of the objects and the robot. **This benchmark lets us evaluate whether diffusion policies trained via RL (not just behavior cloning) have golden tickets.**

4) RGB diffusion policy in DexMimicGen: We use the DexMimicGen benchmark [11] which involves five bimanual tasks with Franka arms: Drawer Cleanup, Lift Tray, Threading, Box Cleanup, and Piece Assembly. We train diffusion policies that take as input RGB images and robot state. **This benchmark lets us systematically test whether golden tickets exist for bimanual visuomotor diffusion policies.**

5) Franka hardware - RGB diffusion policy: We use Franka hardware to conduct real-world experiments. We train RGB diffusion policies (one wrist camera, two static external cameras, and proprioception data) via behavior cloning on a block picking task (Figure 4a). We use a binary success signal, given at the end of the episode if the robot picks the cube, as the reward function to optimize. **This benchmark lets us determine if golden tickets exist for RGB policies on real hardware.**

6) Franka hardware - Pointcloud diffusion policy: We use the same Franka hardware described above, but use two pointcloud policies (using two static external cameras) for two separate tasks: 1) picking a banana (Figure 4c), and 2) pushing a cup to the center of table (Figure 4b). We again use a binary success signal as the reward function to optimize. **This benchmark lets us determine if golden tickets exist for pointcloud policies on real hardware.**

B. Experimental questions

We conduct experiments to answer four questions:

*1) **How do golden tickets compare against using standard Gaussian noise?:*** This lets us understand whether golden tickets exist at all, and what their best possible performance is regardless of computational or sampling cost. If the best lottery tickets are often worse than sampling from a Gaussian on held-out test environments, then the lottery ticket hypothesis cannot reliably be used to improve robot control policies. Due to the continuous and high-dimensional nature of the noise vectors, it is infeasible to test all possible tickets, but we attempt to search for as many tickets as possible to get an approximate upper bound. More experimental details are included in the supplementary material, but to give a sense of the search budgets: our smallest runs are in franka_sim which involve only ≈ 100 tickets with 50 search environments each, and our largest runs are in robomimic, where we search for 5000 tickets per task with 100 search environments each.

*2) **How competitive is finding golden tickets via random search to state-of-the-art latent steering methods?:*** This asks how competitive our approach is compared to other current state-of-the-art latent steering methods that use RL. For these experiments, we compare our random search method proposed in Section IV-C against DSRL's open-source implementation in 5 tasks in DexMimicGen (`Drawer Cleanup`, `Tray Lift`, `Threading`, `Box Cleanup`, and `Piece Assembly`). We report the average success rate performance (and standard deviation) of golden tickets and the trained DSRL policy at various levels of episode budgets ($5,000$ and $10,000$) and different number of DDIM steps (2 and 8).

*3) **Do golden tickets optimized for one task act as golden tickets for other tasks?:*** This is specifically targeted at multi-task policies like VLAs, and lets us understand if golden tickets optimized for one task can be repurposed for others. We investigate this question in LIBERO with SmolVLA, using three task suites: LIBERO-OBJECT, LIBERO-GOAL, and LIBERO-SPATIAL, each containing 10 related tasks. We search for golden tickets in all 3 task families (totaling 30 tasks) by calculating success rates in each task, and then test their performance in other tasks within the task suite. We use the same number of search tickets and environments for golden ticket search as the experiments in our first question. To answer our question, we test whether golden tickets on one task also exhibit higher performance than the base policy on other tasks.

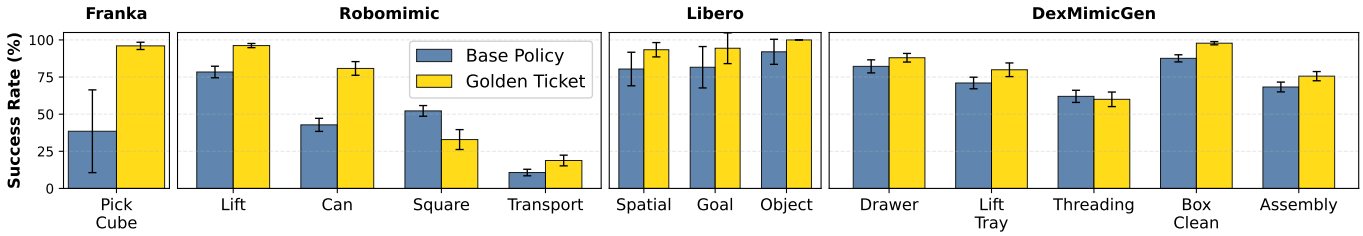


Fig. 5: Comparison of task performance of the base policy (blue, left) and our approach using golden tickets (gold, right) on simulated benchmarks. We report mean and standard deviation of success rates (details in Section V-B). Our open-source repository contains code and golden tickets for the first three benchmarks: franka_sim, robomimic, and LIBERO.

TABLE I: Success rates for base policy and lottery tickets for hardware cube picking task. The base policy was evaluated over 50 episodes. We searched 6 tickets, each for 25 episodes, and then two took the two most extreme results (Tickets 5 and 6) and additionally evaluated over 50 episodes.

Policy	Search Success Rate	Eval Success Rate
Base Policy	–	40/50 (80%)
Ticket 1	11/25 (44%)	–
Ticket 2	24/25 (96%)	–
Ticket 3	10/25 (40%)	–
Ticket 4	10/25 (40%)	–
Ticket 5	25/25 (100%)	49/50 (98%)
Ticket 6	1/25 (4%)	2/48 (4%)

*4) **Do lottery tickets define a Pareto frontier when given multiple objectives to optimize?**:* This seeks to understand how many different ways a single-task pretrained model can be steered via lottery tickets. We investigate this question in franka_sim with a block picking task, where we present two objectives: maximizing success rate, and maximizing speed of successful episodes. Balancing these objectives requires a tradeoff: the faster the robot moves, the harder it becomes to successfully pick up the block. While RL methods would require designing various reward functions that weight these objectives differently, our approach eschews that entirely: We run 400 tickets in 50 different start states, and then evaluate each ticket's performance on the two separate objectives afterwards. We hypothesize that if there is sufficient variation between the behaviors induced by different lottery tickets, then when we plot lottery ticket's performance on these two objectives, a Pareto frontier should become apparent where tickets satisfy these objectives to different extents.

VI. Results

We now discuss our results and their implications on the experimental questions described in Section V-B.

A. How do golden tickets compare against Gaussian noise?

Golden tickets outperform Gaussian noise in 38 out of 43 tasks, and at least match it in 41: The results comparing the best tickets we found versus using standard Gaussian noise on held-out environments can be found in Figure 5. Overall, our results demonstrate that the lottery ticket hypothesis is often true, in the sense that golden tickets can likely be found that improve over the base policy performance.

1) franka_sim: franka_sim is the most extreme example, where the base policy performance for cube picking have an average success rate of 38.5%, whereas the best golden tickets have an average success rate of 96%.

2) robomimic: Golden tickets were found for 3 out of 4 of the tasks (`lift`, `can`, and `transport`), whereas there is only one task (`square`) for which we did not find a ticket that outperformed the base policy. The most extreme performance improvements are in `can`, where average base policy success rate is 42.8%, and is improved to 80.8% with golden tickets, resulting in a 38% improvement.

3) LIBERO*:* For each task suite, we report the average performance of the best performing tickets for each of the tasks in the task suite, emulating a setting where we optimize tickets in a per-task setting (See Table II and III for details). We evaluate golden tickets and the base policy on 50 episodes. [3] In this setting, we are able to find golden tickets that boost the performance of the base policy in all three task suites: 13% increase for LIBERO-Spatial, 12.8% for LIBERO-Goal, and 8% for LIBERO-Object.

4) DexMimicGen: Golden tickets were found in 4 out of 5 tasks (`Drawer Cleanup`, `Tray Lift`, `Box Cleanup`, and `Piece Assembly`). Overall, the performance gains of the best golden ticket are milder, with the biggest gap coming from `Box Cleanup` where the average success rate of the base policy and best golden tickets perform at 87.6% and 97.8% respectively. The one task for which we did not find a golden ticket, `Threading`, has relatively similar average success rate between the base policy (62%) and the best lottery ticket (60%).

5) Hardware experiments: Complete results for our block picking RGB policy can be found in Table I. When using standard Gaussian noise, our block-picking policy successfully picks up the cube 80% of the time (40 out of 50). After searching for 6 tickets, each with 25 episode rollouts, 150 episodes in total, we find a golden ticket (Ticket 5) that succeeded 25 out of 25 times, and also a ticket (Ticket 6) that

[3] Our reported numbers on SmolVLA's success rate for the publicly released checkpoint differs by $\approx 5\%$ from the original paper's [31] reported results. This is because Shukor et al. [31] evaluates on 10 episodes for each LIBERO task suite, whereas we do 50 since 10 was not sufficiently reliable.

TABLE II: Success rates (%) for base policy and lottery tickets on LIBERO-GOAL and LIBERO-OBJECT with SmolVLA. T0 - T9 represent different tasks. Top-1 ticket per task shown (unique set) plus best average ticket (⋆).

	LIBERO-Goal												LIBERO-Object										
#Ticket	**T0**	**T1**	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**	**T8**	**T9**	**Avg**	**#Ticket**	**T0**	**T1**	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**	**T8**	**T9**	**Avg**
Base Policy	72	94	86	52	92	78	80	**100**	94	68	**81.6**	Base Policy	82	98	98	98	76	82	**100**	90	96	**100**	**92.0**
#03c2 ⋆	84	**100**	92	40	**100**	64	**94**	**100**	98	18	79.0	#015a ⋆	36	62	**100**	**100**	16	92	**100**	94	**100**	**100**	80.0
#2cee	78	84	**100**	2	98	76	22	**100**	86	0	64.6	#5159	20	86	**100**	**100**	60	**100**	94	36	92	**100**	78.8
#1ff8	98	64	86	14	**100**	0	38	96	**100**	2	59.8	#184f	98	0	94	86	**100**	0	78	**100**	86	44	68.6
#4d97	2	96	16	**68**	98	92	30	54	**100**	4	56.0	#1944	**100**	38	72	94	94	4	30	90	72	0	59.4
#672f	0	12	88	0	72	4	38	**100**	2	**90**	40.6	#14e7	14	**100**	0	0	0	32	56	64	56	0	32.2
#0f3f	**100**	2	0	0	68	56	0	76	38	0	34.0	#096d	50	0	0	4	0	0	0	**100**	0	0	15.4
#0310	0	12	0	0	**100**	**100**	10	4	42	0	26.8												

TABLE III: Success rates (%) for base policy and lottery tickets on LIBERO-SPATIAL with SmolVLA. T0 - T9 represent different tasks. Top-1 ticket per task shown (unique set) plus best average ticket (⋆).

	LIBERO-Spatial										
#Ticket	**T0**	**T1**	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**	**T8**	**T9**	**Avg**
Base Policy	78	94	84	62	84	58	90	90	78	86	**80.4**
#60c0 ⋆	82	82	88	**100**	74	24	98	80	82	68	77.8
#a68f	76	72	92	50	**92**	**90**	56	70	**90**	82	77.0
#ecf0	68	96	80	76	12	30	98	**96**	82	68	70.6
#0e0c	80	60	86	**100**	10	24	90	92	84	74	70.0
#05c7	70	92	88	72	52	0	**100**	64	44	78	66.0
#517b	72	84	**92**	78	2	20	66	92	62	52	62.0
#6a21	64	**98**	90	30	**92**	30	30	74	58	44	61.0
#7c0e	64	68	64	4	22	60	6	88	44	**92**	51.2
#3b9d	**84**	36	46	94	14	6	88	22	16	0	40.6

succeeded only 1 out of 25 times). When we then evaluate those two tickets an additional 50 times, Ticket 5 successfully picked the cube 98% of the time (49 out of 50), where Ticket 6 only succeeded 2 out of 50 times. We therefore show we can drive success rate from 80% to 98%, an 18% increase, using just 150 search episodes, or conversely steer the policy to miss with extreme reliability.

For the banana picking pointcloud policy, we initially evaluate the base policy at a single location 10 times, where it successfully picks the banana 30% of the time. After searching for 10 tickets in the same location for 5 episodes each, we found a ticket that performed 100%. We then evaluated that golden ticket and the base policy at 4 other locations on the table, 10 episodes each. The base policy has an average success rate of 50% whereas the golden ticket performs at 68%. This results in a 18% improvement across the table after only 50 episodes.

For the cup pushing pointcloud policy, we evaluated the base policy 10 times and observed an average success rate of 40%. We searched for 10 tickets, with 5 rollouts, and found a ticket that achieved 100% success rate. When evaluated an additional 10 times, we found that it continued to achieve a 100% success rate. This constitutes our largest improvements on hardware, where the average success rate was increased by 60%.

B. How competitive is finding golden tickets via random search to state-of-the-art latent steering methods?

Golden tickets found with random search can be competitive with DSRL across 5 tasks. Our results comparing our search method to DSRL for DexMimicGen can be found in Figure 6. Our approach outperforms DSRL when using 8 DDIM steps, which does not even match the base policy performance. However, we note that when using a lower number of DDIM steps (2), DSRL slightly outperforms our approach, although we do still find golden tickets that outperform the base policy. Despite our method not requiring new neural networks to be trained (thereby having lower computation cost and complexity), it is competitive with state-of-the-art RL methods for latent steering.

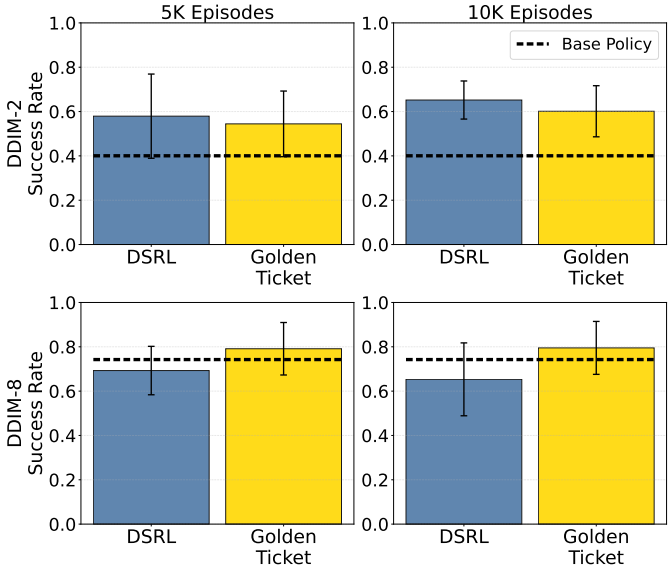


Fig. 6: Comparison of DSRL vs. our proposed method in DexMimicGen (we report average success rate and standard deviation across all 5 tasks). Top/bottom row show varying numbers of DDIM steps (2 vs. 8), and left/right column show results at 5k and 10k episodes. Golden tickets are on-par with DSRL, and require less computation and model design.

C. Do golden tickets optimized for one task act as golden tickets for other tasks?

Golden tickets for one task can be golden tickets for other tasks: We show the results of various golden tickets we found for three task suites in LIBERO in Figure Table II and III. Across all 30 tasks, golden tickets at least match the base policy's performance, and the 3 tasks where performance is matched (LIBERO-GOAL T7, and LIBERO-OBJECT T6, T9) is when both achieve 100% success rate. More inter-

estingly, we see that certain golden tickets do not perform better than the base policy in only one task, but in multiple tasks. For example, in LIBERO-GOAL, we find a ticket (ID: `#03c2`) that achieves 100% success rate in 3 separate tasks (`T1`, `T4`, `T7`), whereas the base policy only achieves 100% success in `T7`. For LIBERO-OBJECT, we find a golden ticket (ID: `#015a`) that achieves 100% in 5 tasks (`T2`, `T3`, `T6`, `T8`, `T9`), where the base policy only achieves 100% success rate in 2 tasks (`T6`, `T9`). For LIBERO-SPATIAL, we find a golden ticket (ID: `#a68f`) that outperforms the base policy in three tasks (`T4`, `T5`, `T8`), where it achieves at least 90% success rate.

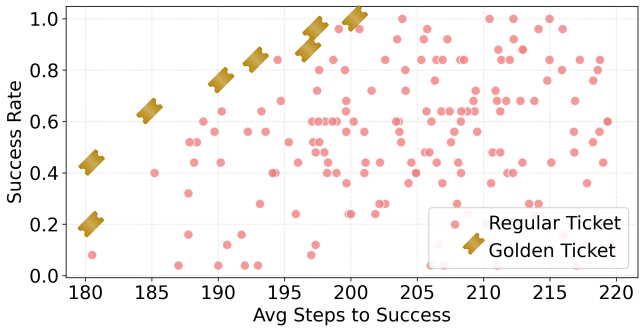


Fig. 7: Various tickets (pink) for the franka_sim `pick` policy, evaluated according to success rate and speed (determined by length of successful episodes). Higher is better success rate, left is faster time to success. Because lottery tickets exhibit extreme differences in policy performance, a Pareto frontier is defined by tickets that are further left/up than others (represented with golden ticket icons).

We note that we did not find any golden tickets with better performance than using Gaussian noise when averaged across the entire task suite. The smallest gap was in LIBERO-SPATIAL (80.4% and 77.8% for the base policy and ticket `#60c0` respectively), and the most dramatic gap was in LIBERO-OBJECT (92% and 80% for the base policy and ticket `#015a`). We hypothesize that this is because some tasks benefit from distinct behaviors (e.g., grasping from the top vs. the side), and so the useful motions that are emphasized by a golden ticket for one task may be counterproductive for others.

D. Do lottery tickets define a Pareto frontier when there are multiple objectives to optimize?

Golden tickets are sufficiently varied to define a Pareto frontier that balances success rate and speed for an object picking task. Our results showing the performance of various tickets on success rate for picking vs. speed of successful picks can be seen in Figure 7. We see a large variation in ticket performance on both metrics, with ticket success rates varying from $\approx 5\%$ to nearly 100%, and speeds varying from taking ≈ 180 steps to 220 steps. This diversity results in a small subset of golden tickets that define a Pareto frontier, meaning that they outperform all other regular tickets to the right and below them, and represent a balance between the two objectives. This is a unique property of golden tickets compared to other RL methods: rather than designing multiple reward functions that balance these objectives differently, we naturally find tickets that balance them without any reward tuning. This suggests a simple way to alter a robot's behavior to adapt between objectives online: collect the golden tickets on the Pareto frontier, and switch between them as desired.

VII. LIMITATIONS AND FUTURE WORK

There are important limitations to golden tickets that are addressable in future work. First, by replacing the initial noise distribution with a single input, the resulting policy is deterministic (unless a stochastic reverse sampling process is used). Adding stochasticity back into the policy (either by sometimes injecting Gaussian noise along with the golden ticket, or randomly sampling n golden tickets instead of just using one) may help mitigate this. Secondly, investigating how to identify golden tickets using offline metrics may significantly improve the sample efficiency and search performance. Lastly, using other RL techniques to optimize a golden ticket in a manner similar to DSRL is also promising, potentially leveraging approaches like vector quantization [5].

VIII. CONCLUSION

In this work, we propose an approach to improving pretrained diffusion or flow matching policies that avoids 1) adjusting the original policy weights, 2) training additional neural networks, and 3) making training or test time assumptions on the base model. Our method is based on a proposed lottery ticket hypothesis for robot control: that the performance of a pretrained, frozen diffusion or flow matching policy can be improved by swapping the sampling of initial noise from the prior distribution (typically isotropic Gaussian) with a well-chosen, constant initial noise input, called a golden ticket. Through our experiments, we demonstrate that 1) golden tickets often exist and outperform using Gaussian noise, 2) golden tickets can be competitive with state-of-the-art latent steering methods trained with RL, and 3) golden tickets optimized on one task can also act as golden tickets for different tasks. We also release an open-source codebase with models, environments and golden tickets for VLAs, diffusion policies, and flow matching policies.

ACKNOWLEDGMENTS

We thank Stefanie Tellex for providing valuable feedback on the manuscript.

REFERENCES

[1] Changgu Chen, Libing Yang, Xiaoyan Yang, Lianggangxu Chen, Gaoqi He, Changbo Wang, and Yang Li. Find: Fine-tuning initial noise distribution with policy optimization for diffusion models. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 6735–6744, 2024.

[2] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran

Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, page 02783649241273668, 2023.

[3] Maximilian Du and Shuran Song. Dynaguide: Steering diffusion polices with active dynamic guidance. *arXiv preprint arXiv:2506.13922*, 2025.

[4] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks, 2019. URL https://arxiv.org/abs/1803.03635.

[5] Robert Gray. Vector quantization. *IEEE Assp Magazine*, 1(2):4–29, 1984.

[6] D Hendrycks. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[7] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[8] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020. URL https://arxiv.org/abs/2006.11239.

[9] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *CoRR*, abs/2106.09685, 2021. URL https://arxiv.org/abs/2106.09685.

[10] Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, Danny Driess, Michael Equi, Adnan Esmail, Yunhao Fang, Chelsea Finn, Catherine Glossop, Thomas Godden, Ivan Goryachev, Lachy Groom, Hunter Hancock, Karol Hausman, Gashon Hussein, Brian Ichter, Szymon Jakubczak, Rowan Jen, Tim Jones, Ben Katz, Liyiming Ke, Chandra Kuchi, Marinda Lamb, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Yao Lu, Vishnu Mano, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Charvi Sharma, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, Will Stoeckle, Alex Swerdlow, James Tanner, Marcel Torne, Quan Vuong, Anna Walling, Haohuan Wang, Blake Williams, Sukwon Yoo, Lili Yu, Ury Zhilinsky, and Zhiyuan Zhou. $\pi^*_{0.6}$: a vla that learns from experience, 2025. URL https://arxiv.org/abs/2511.14759.

[11] Zhenyu Jiang, Yuqi Xie, Kevin Lin, Zhenjia Xu, Weikang Wan, Ajay Mandlekar, Linxi Jim Fan, and Yuke Zhu. Dexmimicgen: Automated data generation for bimanual dexterous manipulation via imitation learning. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 16923–16930. IEEE, 2025.

[12] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[13] Yunfei Li, Xiao Ma, Jiafeng Xu, Yu Cui, Zhongren Cui, Zhigang Han, Liqun Huang, Tao Kong, Yuxiao Liu, Hao Niu, et al. Gr-rl: Going dexterous and precise for long-horizon robotic manipulation. *arXiv preprint arXiv:2512.01801*, 2025.

[14] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling, 2023. URL https://arxiv.org/abs/2210.02747.

[15] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.

[16] Guanxing Lu, Wenkai Guo, Chubin Zhang, Yuheng Zhou, Haonan Jiang, Zifeng Gao, Yansong Tang, and Ziwei Wang. Vla-rl: Towards masterful and general robotic manipulation with scalable reinforcement learning. *arXiv preprint arXiv:2505.18719*, 2025.

[17] Jianlan Luo, Zheyuan Hu, Charles Xu, You Liang Tan, Jacob Berg, Archit Sharma, Stefan Schaal, Chelsea Finn, Abhishek Gupta, and Sergey Levine. Serl: A software suite for sample-efficient robotic reinforcement learning, 2024.

[18] Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. An empirical study of catastrophic forgetting in large language models during continual fine-tuning, 2025. URL https://arxiv.org/abs/2308.08747.

[19] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. *arXiv preprint arXiv:2108.03298*, 2021.

[20] Jiafeng Mao, Xueting Wang, and Kiyoharu Aizawa. The lottery ticket hypothesis in denoising: Towards semantic-driven initialization. In *European Conference on Computer Vision*, pages 93–109. Springer, 2024.

[21] Yanting Miao, William Loh, Pacal Poupart, and Suraj Kothawade. A minimalist method for fine-tuning text-to-image diffusion models. *arXiv preprint arXiv:2506.12036*, 2025.

[22] Mitsuhiko Nakamoto, Oier Mees, Aviral Kumar, and Sergey Levine. Steering your generalists: Improving robotic foundation models via value guidance. *arXiv preprint arXiv:2410.13816*, 2024.

[23] Aaditya Prasad, Kevin Lin, Jimmy Wu, Linqi Zhou, and Jeannette Bohg. Consistency policy: Accelerated visuomotor policies via consistency distillation. *arXiv preprint arXiv:2405.07503*, 2024.

[24] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[25] Zipeng Qi, Lichen Bai, Haoyi Xiong, and Zeke Xie. Not all noises are created equally: Diffusion noise selection and optimization. *arXiv preprint arXiv:2407.14041*, 2024.

[26] Allen Z Ren, Justin Lidard, Lars L Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion policy policy optimization. *arXiv preprint*

arXiv:2409.00588, 2024.

[27] Roey Ron, Guy Tevet, Haim Sawdayee, and Amit H Bermano. Hoidini: Human-object interaction through diffusion noise optimization. *arXiv preprint arXiv:2506.15625*, 2025.

[28] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015.

[29] Dvir Samuel, Barak Meiri, Haggai Maron, Yoad Tewel, Nir Darshan, Shai Avidan, Gal Chechik, and Rami Ben-Ari. Lightning-fast image inversion and editing for text-to-image diffusion models. *arXiv preprint arXiv:2312.12540*, 2023.

[30] Dvir Samuel, Rami Ben-Ari, Simon Raviv, Nir Darshan, and Gal Chechik. Generating images of rare concepts using pre-trained diffusion models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 4695–4703, 2024.

[31] Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, et al. Smolvla: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025.

[32] Tom Silver, Kelsey Allen, Josh Tenenbaum, and Leslie Kaelbling. Residual policy learning. *arXiv preprint arXiv:1812.06298*, 2018.

[33] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models, 2022. URL https://arxiv.org/abs/2010.02502.

[34] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[35] Zhiwei Tang, Jiangweizhi Peng, Jiasheng Tang, Mingyi Hong, Fan Wang, and Tsung-Hui Chang. Inference-time alignment of diffusion models with direct noise optimization. *arXiv preprint arXiv:2405.18881*, 2024.

[36] Andrew Wagenmaker, Mitsuhiko Nakamoto, Yunchu Zhang, Seohong Park, Waleed Yagoub, Anusha Nagabandi, Abhishek Gupta, and Sergey Levine. Steering your diffusion policy with latent space reinforcement learning. *arXiv preprint arXiv:2506.15799*, 2025.

[37] Yanwei Wang, Lirui Wang, Yilun Du, Balakumar Sundaralingam, Xuning Yang, Yu-Wei Chao, Claudia Perez-D'Arpino, Dieter Fox, and Julie Shah. Inference-time policy steering through human interactions, 2025. URL https://arxiv.org/abs/2411.16627.

[38] Yanwei Wang, Lirui Wang, Yilun Du, Balakumar Sundaralingam, Xuning Yang, Yu-Wei Chao, Claudia Pérez-D'Arpino, Dieter Fox, and Julie Shah. Inference-time policy steering through human interactions. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pages 15626–15633. IEEE, 2025.

[39] Zhendong Wang, Zhaoshuo Li, Ajay Mandlekar, Zhenjia Xu, Jiaojiao Fan, Yashraj Narang, Linxi Fan, Yuke Zhu, Yogesh Balaji, Mingyuan Zhou, et al. One-step diffusion policy: Fast visuomotor policies via diffusion distillation. *arXiv preprint arXiv:2410.21257*, 2024.

[40] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

[41] Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

[42] Zikai Zhou, Shitong Shao, Lichen Bai, Shufei Zhang, Zhiqiang Xu, Bo Han, and Zeke Xie. Golden noise for diffusion models: A learning framework. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 17688–17697, 2025.

APPENDIX A
RELATED WORK

A. Robot Policy Improvement Methods

One class of approaches to improving pretrained policies involves directly finetuning the model parameters (or applying model adaption methods like Low-Rank Adaptation (LoRA) [9]), using either supervised [2] or reinforcement learning (RL) losses [16]. These approaches apply the same pretraining techniques to improve policies, but can be challenging to use, since adjusting policy weights can lead to catastrophic forgetting [18] and is computationally hard for large models like VLAs. Another class of approaches learns a new model that adjusts policy output, such as residual policy learning [32] and latent steering [36]. These methods avoid adjusting pretrained policy weights, which helps preserve the original behavior, but require carefully designing/training new neural modules with sufficient model capacity to avoid acting as bottlenecks [34]. A last class of approaches involves making additional training/inference-time assumptions (e.g., inference-time steering [38], classifier-free guidance [10], world models [3], value-guidance [22]). These methods avoid adjusting the pretrained policy weights or learning additional neural modules for policy improvement, but they do not apply to all pretrained models (e.g., classifier-free guidance assumes a particular training regime) or assume extra models are available (e.g., access to a world model to use dynamics for guidance, or a user to provide corrective behaviors).

APPENDIX B
LOTTERY TICKET HYPOTHESIS FOR ROBOT CONTROL

We provide a detailed list of additional differences between the image generation setting, and the robot control setting.

1) **Dimensionality of denoising object**: In image generation, the object being denoised is an image, which typically contains 2 spatial dimensions and 3 channel dimensions for RGB color. For denoising an image, the dimensionality of the noise vector could be $128 \times 128 \times 3 = 49152$. In robotics, the object being denoised is an action chunk, whose dimensionality is equal to the product of the size of the one-step action and the chunk horizon length. An example of a particularly large action chunk is SmolVLA [31], which has a one-step action size of 32 and an action chunk horizon of 50, making the total dimensionality of the denoised object $50 \times 32 = 1600$. Therefore, the dimensionality of the denoising object in image generation can be orders of magnitude larger than the denoising object in robotics. For images, not all pixels are equally important, in the sense that some pixels will contain the foreground, target object, whereas the background pixels may be less important to capture for the target query. However, in robotics, there is no equivalent notion of "background" and "foreground" actions; instead, any action produced in the action chunk and executed by the robot may impact performance.
2) **Dimensionality of conditioning object**: In image generation, the object used for conditioning is typically a language description, which is either projected to a single vector or processed by a transformer into a sequence of discrete tokens. In robotics, the objects used for conditioning are typically visual data (e.g: images), proprioception data (e.g., joint positions, velocities, forces), and language (for task-conditioned policies like VLAs). Therefore, the dimensionality of the conditioning object can be orders of magnitude larger than the denoising object in image generation, and typically requires specialized encoders for the various conditioning objects. This makes designing observation-conditioned noise policies particularly challenging since they may need to handle a variety of robot sensor data.
3) **Temporal Decision Making**: Image generation tasks do not involve temporal decision making: once an image is generated, the next image to be generated is completely independent of the previous one. Additionally, for image generation, we may only need to optimize a noise for a single conditioning vector (i.e., just make good images of dogs). In robotics, we are addressing tasks that involve temporal decision making due to interacting in an environment, which means that the action chunk produced by the previous noise vector has large impact on distribution of next conditioning vectors we encounter. Unlike image generation where we may get similar prompts over time, in robotics, we may never get the same exact robot state to condition on again. Since the performance of the policy is not solely determined by any single action chunk, but instead depends on the performance throughout the entire task, evaluating tickets in robotics involves having a testing environment. Evaluating with an environment introduces unique complexities since much of the computational cost can come from the environment instead of running the policy, especially when the policy executes a large fraction of the action chunk before recomputing a new one.
4) **Cost of evaluating tickets**: In image generation, the metric to evaluate tickets is typically Fréchet Inception Distance (FID), which requires doing one full pass through the reverse process with the noise, making the model inference the most computationally expensive component. In robotics, we want to optimize the cumulative discounted expected rewards of the policy, which requires running the policy in an environment. For simulation, this causes evaluation to take more time and compute, but for hardware experiments, this is especially challenging and costly, potentially requiring human effort to deal with resets.
5) **Data manifold complexity**: For image generation tasks, the implicit data manifold that represents the space of desired images to be generated is high-dimensional, even when the text prompt is fixed. This is because there is typically a target concept to be generated in

the foreground, which can be spatially placed in many locations, and the rest of the image is filled in with a background. This results in a combinatorially large number of ways to compose concepts in images. In robotics, the action chunk manifold is typically relatively low-dimensional (sometimes collapsing to just single actions depends on the conditioning vector and data). Therefore, many noises may map to the same action chunk for conditioning vectors, which makes it harder to get a differentiating signal on tickets. Some algorithms (like DSRL-NA [36]) exploit this aliasing property directly.

6) **Evaluation Metric Differentiability**: In image generation, differentiable metrics exists, such as FID score, which makes it possible to optimize the initial noise by calculating gradients through the model. In robotics, the rewards functions are typically not differentiable, therefore making it more challenging to implement gradient-based optimization techniques. However, methods like policy gradients [40], which avoid calculating gradients through the reward function, and transition dynamics can be leveraged to optimize cumulative discounted expected rewards.

APPENDIX C
EXPERIMENTAL BENCHMARKS

We provide more details on our experimental benchmarks.

1) Flow matching policy in franka_sim: We use a 4 layer MLP with GELU activations [6] as the non-linearities, and each layer has 256 hidden dimension. We collect 1000 demonstrations from our task-and-motion planning heuristic, and train 4 model checkpoints for 100 epochs, using a batch size of 20. We use the Adam [12] optimizer with a learning rate of 0.001. The policy outputs an 8 step action chunk, with each step containing the 3D position of the end-effector, and a 1D gripper state, culminating in a total action chunk size of $8 * 4 = 32$. The cube spawns randomly inside bounds of $[0.25 - 0.25]$ (x-bounds, meters) and $[0.55, 0.25]$ (y-bounds, meters) on the ground. The goal is to lift the cube off the ground.

For each of the 4 model checkpoints, we search for ≈ 250 lottery tickets in 25 starting block poses, and evaluate in 25 held-out evaluation block poses.

2) SmolVLA in LIBERO*:* We use the publicly released SmolVLA model checkpoint that was finetuned for LIBERO https://huggingface.co/HuggingFaceVLA/smolvla_libero, where all model and training details can be found. We use all default configurations for inference included with the model card. The policy takes in 2 RGB images, the low-dimensional state of the robot, and a language description of the task. SmolVLA outputs a one-step action of dimension 32, with a 50 chunk horizon, resulting in a total action chunk size of dimension 1600.

For our experiments, we searched for 1416 tickets in LIBERO-OBJECT, 1338 tickets in LIBERO-GOAL, and 1081 tickets ib LIBERO-SPATIAL. For LIBERO-OBJECT, we evaluated 5 search environments for each ticket, whereas for the other two task suites we varied between 3 and 5 search episodes for different tickets. We test tickets on 50 evaluation initial states for each task.

TABLE IV: Success rates for base policy and lottery tickets for hardware cup pushing task.

Policy	Search Success Rate	Eval Success Rate
Base Policy	–	4/10 (40%)
Ticket 1	1/5 (20%)	–
Ticket 2	5/5 (100%)	10/10 (100%)
Ticket 3	0/5 (0%)	–
Ticket 4	5/5 (100%)	–
Ticket 5	0/5 (0%)	–
Ticket 6	0/5 (0%)	–
Ticket 7	1/5 (25%)	–
Ticket 8	5/5 (100%)	–
Ticket 9	5/5 (100%)	–
Ticket 10	5/5 (100%)	–

TABLE V: Success rates for base policy and lottery tickets for banana picking task.

Policy	Location	Search Success Rate	Eval Success Rate
Base Policy	#1	–	3/10 (30%)
Base Policy	#2	–	8/10 (80%)
Base Policy	#3	–	5/10 (50%)
Base Policy	#4	–	1/10 (10%)
Base Policy	#5	–	8/10 (80%)
Ticket 1	#1	5/5 (100%)	5/5 (100%)
Ticket 1	#2	–	4/10 (40%)
Ticket 1	#3	–	0/10 (0%)
Ticket 1	#4	–	10/10 (100%)
Ticket 1	#5		8/10 (80%)
Ticket 2	#1	1/5 (20%)	–
Ticket 3	#1	1/5 (20%)	–
Ticket 4	#1	0/5 (0%)	–
Ticket 5	#1	0/5 (0%)	–
Ticket 6	#1	0/5 (0%)	–
Ticket 7	#1	0/5 (0%)	–
Ticket 8	#1	0/5 (0%)	–
Ticket 9	#1	0/5 (0%)	–
Ticket 10	#1	0/5 (0%)	–

3) DPPO in robomimic: We use the publicly released checkpoints released from the original DPPO codebase (which were also used in the original DSRL experiments): https://github.com/irom-princeton/dppo. We search for 5000 tickets, for 100 environments each. We evaluate on 100 episodes across 5 random seeds.

4) RGB diffusion policy in DexMimicGen: We use a diffusion policy with a U-Net backbone [28], which has a ResNet-18 encoder for the RGB images, and an MLP for the robot proprioception data. The policy takes in 3 RGB images and the end effector position, quaternion and gripper state for both arms. We train a separate policy for each of the 5 tasks.

We search for 500 tickets in 50 environments each. We then evaluate on 100 environments across 5 random seeds.

5) Franka hardware - RGB diffusion policy: We use RealSense D435 cameras for our static, external cameras, and D405 for the wrist camera. The Franka Research 3 arm is equipped with a Robotiq 2F-85 gripper. For the RGB policies, we use a standard diffusion policy architecture with a U-Net

backbone and ResNet-18 architecture for the image encoders.

6) Franka hardware - Pointcloud diffusion policy: For the pointcloud policies, we use the same U-Net backbone but instead use a PointNet encoder [24] for the pointcloud. We use 2 calibrated extrinsic cameras to generate a single fused pointcloud, and remove all points that are at or below the surface of the table.

APPENDIX D
RESULTS

A. Real World Results

We present complete experimental results (the number of tickets searched and evaluated) for the banana picking task and cup pushing task in Tables V and IV respectively.

For cup pushing, the base policy was evaluated over 10 episodes. We searched 10 tickets, each for 5 episodes, and then the best performing ticket (Ticket 2) and evaluated an additional 10 episodes.

For banana picking, the base policy was evaluated in 5 locations, 10 episodes each. We searched 10 tickets, each for 5 episodes, in one location, then took the best performing ticket and evaluated it in 4 other locations for 10 episodes each. We note that Ticket #1 performs worse than the base policy in two locations (#2 and #3), these two locations are significantly further to the right side of the table than locations #1, #4, and #5. While this suggests that golden tickets can generalize within some convex hull of spatial locations, complete coverage of the table may require the use of multiple tickets across the space.

B. Effect of DDIM Steps on Lottery Ticket Hypothesis

When using diffusion models, our proposed lottery ticket search, along with latent steering approaches like DSRL [36] assume DDIM sampling [33] since it is deterministic and therefore takes less samples to estimate the cumulative discounted expected rewards induced by an initial noise vector. DDIM sampling has been widely adopted in robotics as an alternative to DDPM [7] as it requires fewer sampling steps, although other techniques such as distillation [39, 23] have additionally been proposed. Since our work only uses DDIM, we specifically investigate how changing the number of denoising steps in DDIM impacts performance of the base policy and golden tickets. However, we suspect these results have implications on the other aforementioned sampling techniques.

Figure 8 and 9 for robomimic and DexMimicGen respectively compare the average performance of top-3 golden tickets with the base policy for 2 and 8 DDIM sampling steps. We see that golden tickets found for DDIM-2 match or outperform the base policy (using DDIM-2) for all tasks across both benchmarks. Note that the golden tickets for 2 and 8 DDIM steps are searched separately and with a budget of 5000 and 500 tickets for robomimic and DexMimicGen, respectively. We find that DDIM-2 golden tickets close the performance gap with the base policy using DDIM-8 in several tasks such as `Lift`, `Can` and `Box Cleanup`.

Interestingly, across Figures 8 and 9, we see that golden tickets yield bigger performance improvements as compared to the base policy at 2 DDIM steps than at 8 DDIM steps. Mechanistic explanations for golden tickets are complicated by the temporal nature of policy rollouts, which we leave for future work to investigate further.

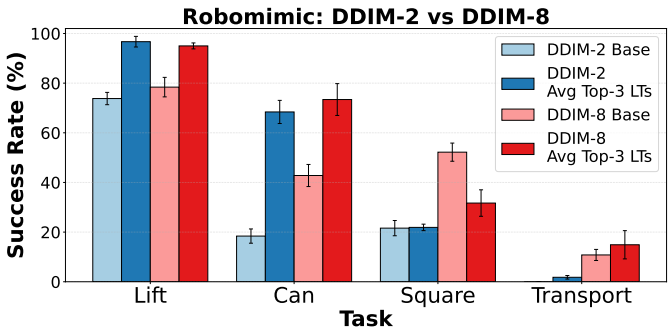


Fig. 8: Base policy and golden ticket performance with 2 and 8 DDIM steps for 4 tasks in robomimic. Golden tickets found for `lift` and `can` at DDIM-2 even outperform the base policy performance with 8 DDIM steps.

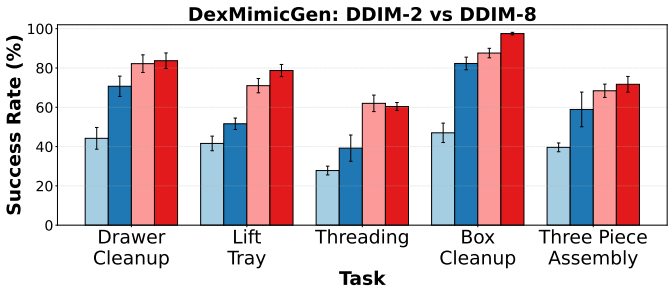


Fig. 9: Base policy and golden ticket performance with 2 and 8 DDIM steps for 5 tasks in DexMimicGen. We see a greater performance increase from golden tickets at DDIM-2 as compared to DDIM-8. The golden tickets for both DDIM-steps were found with the equal search budgets.