Learning Latent Plans from Play

Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar,
Jonathan Tompson, Sergey Levine, Pierre Sermanet
Google Brain

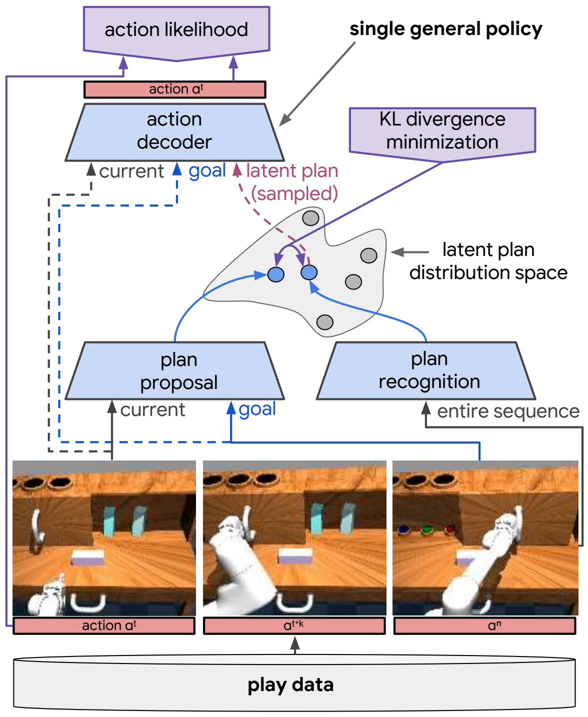
Abstract: Acquiring a diverse repertoire of general-purpose skills remains an open challenge for robotics. In this work, we propose self-supervising control on top of human teleoperated play data as a way to scale up skill learning. Play has two properties that make it attractive compared to conventional task demonstrations. Play is cheap, as it can be collected in large quantities quickly without task segmenting, labeling, or resetting to an initial state. Play is naturally rich, covering $\sim$4x more interaction space than task demonstrations for the same amount of collection time. To learn control from play, we introduce Play-LMP, a self-supervised method that learns to organize play behaviors in a latent space, then reuse them at test time to achieve specific goals. Combining self-supervised control with a diverse play dataset shifts the focus of skill learning from a narrow and discrete set of tasks to the full continuum of behaviors available in an environment. We find that this combination generalizes well empirically—after self-supervising on unlabeled play, our method substantially outperforms individual expert-trained policies on 18 difficult user-specified visual manipulation tasks in a simulated robotic tabletop environment. We additionally find that play-supervised models, unlike their expert-trained counterparts, are more robust to perturbations and exhibit retrying-till-success behaviors. Finally, we find that our agent organizes its latent plan space around functional tasks, despite never being trained with task labels. Videos, code and data are available at learning-from-play.github.io

1 Introduction

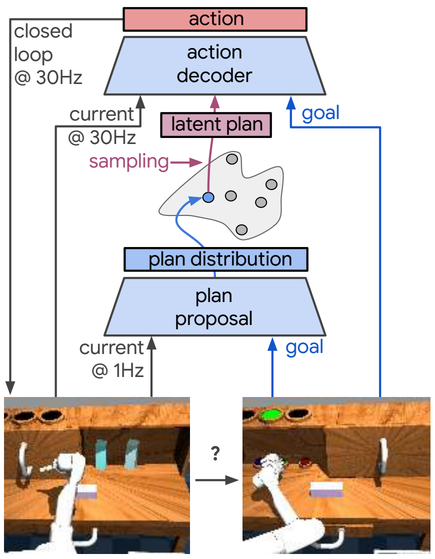
There has been significant recent progress showing that robots can be trained to be competent specialists, learning complex individual skills like grasping ([1]), locomotion, and dexterous manipulation ([2]). In this work, we are motivated by the idea of a robot generalist: A single agent capable of learning a wide variety of skills. This remains a challenging open problem in robotics.

Conventionally, obtaining multiple skills involves defining a discrete set of tasks we care about, collecting a large number of labeled and segmented expert demonstrations per task, then training one specialist policy per task in a learning from demonstration (LfD) [3] scenario. Alternatively, we might turn to reinforcement learning as a means of obtaining a set of skills, which would entail manually designing one reward per task to drive policy learning. Unfortunately, designing reward functions for robotic skills is very challenging, especially when learning from raw observations, typically requiring manually-designed perception systems. Additionally, using reinforcement learning in complex settings such as robotics requires overcoming significant exploration challenges, typically addressed by introducing manual scripting primitives to an unsupervised collection ([4]) that increase the likelihood of behavior with non-zero reward. In general for both paradigms, for each new skill a robot is required to perform, a corresponding, sizeable, and non-transferable human effort must be expended.

Furthermore, in real world settings, agents will be expected to perform not just a small discrete set of tasks, but rather a wide continuum of behaviors. This presents a challenge for conventional methods—if a slight variation of a skill is needed, e.g. opening a drawer by grasping the handle from the top down rather than bottom up, an entirely new set of demonstrations or reward functions might be required to learn the behavior. To address this, we are motivated by the idea of an agent capable of *task-agnostic control*: the ability to reach *any* reachable goal state from any current state [5]. In this setting, the notion of "task" is no longer discrete, but continuous—indexed by the pair (current state



(a) **Play-LMP** training.



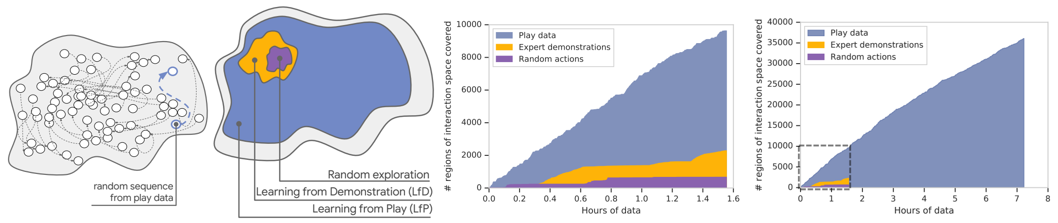
(b) **Task-agnostic policy inference**

Figure 1: **Play-LMP:** A single model that self-supervises control from play data, then generalizes to a wide variety of manipulation tasks. (a) **Training**: 1) sample a random window of experience from a memory of play data; 2) train to recognize and organize a repertoire of behaviors executed during play in a **latent plan space**, 3) train a policy, conditioned on current state, goal state, and a sampled latent plan to reconstruct the actions in the selected window. The latent plan space is shaped by two stochastic encoders: plan recognition and plan proposal. Plan recognition takes the entire sequence, recognizing the exact behavior executed. Plan proposal takes the initial and final state, outputting a distribution over all possible behaviors that connect initial state to final. We minimize the KL divergence between the two encoders, making the plan proposal assign high likelihood to behaviors that were actually executed during play. (b) **Inference**: the policy is conditioned on the current state, the goal state (specified by the user) and a latent plan which is sampled once from a plan distribution (inferred from the current and goal states).

s_c, goal state s_g). Learning in this setting can be formalized as the search for a goal-conditioned policy $\pi_\theta(a|s_c, s_g)$ (Kaelbling [6]).

To generalize to the widest variety of tasks at test time (indexed by the pair (s_c, s_g)), it stands that the agent should see the widest variety of (s_c, s_g) pairs during training, along with actions that connect current and goal states. The ideal dataset to learn task-agnostic control then is both broad and dense in its coverage of the environment's interaction space: Fig. 2a. Unfortunately, it is difficult to obtain datasets with this sort of coverage (Fig. 2b) in practice. Random exploration, while cheap to collect, is typically insufficiently rich to power the learning of complex manipulation. Expert demonstrations, on the other hand, can be arbitrarily complex but are expensive to collect, and still typically form narrow training distributions over visited states, leading to an empirical "distribution shift" problem (Ross et al. [7]) at test time.

In this work, we propose an alternative means of obtaining task-agnostic control—self-supervising on top of unlabeled teleoperated *play data*: continuous logs of low-level observations and actions collected while a human teleoperates the robot and engages in behavior that satisfies their own curiosity. We emphasize two properties that make human play data a compelling choice for the basis of learning goal-conditioned control. Play data is *cheap*: Unlike expert demonstrations (Fig. 5), play requires no task segmenting, labeling, or resetting to an initial state, meaning it can be collected quickly in large quantities. Play data is *rich*. Play is not random but rather structured by human knowledge of object affordances (e.g. if people see a button in a scene, they tend to press it). This makes play much more discriminate than what can be achieved by random scripting. Unlike task demonstrations, operators are driven by their own curiosity during play, trying multiple ways



(a) **Ideal coverage** (b) **Methods coverage** (c) **Coverage** (1.5 hours) (d) **Coverage** (7 hours)

Figure 2: **The continuum of skills and its coverage.** We advocate for learning the full continuum of skills at once rather than discrete ones. (a) The ideal coverage is dense and broad over all regions of the space, providing statistical support for all pairs of (current state, goal state). (b) We hypothesize different approaches yield different coverages. (c) We observe in real datasets that for the same amount of collection time, play data's coverage largely surpasses that of 18 tasks worth of expert demonstrations and random exploration. Unlike the other methods, play data coverage appears to grow linearly with collection time. This prompted us to explore its coverage at larger scales, where we continued to observe the phenomenon. (d). See details in A.4.3 in [8].

of achieving the same outcome or exploring new behaviors. In this way, we can expect play to naturally cover an environment's interaction space. In our datasets Fig. 2c, we find empirically that for the same amount of collection time, play indeed covers 4.2 times more regions of the available interaction space than 18 tasks worth of expert demonstration data, and 14.4 times more regions than random exploration.

In Sec. 3, we propose two self-supervised methods for learning task-agnostic control from play: Play-GCBC and Play-LMP.

2 Related Work

Robotic learning methods generally require some form of supervision to acquire behavioral skills. Conventionally, this supervision either consists of a cost or reward signal, as in reinforcement learning [9, 10, 11], or demonstrations, as in imitation learning Pastor et al. [3]. However, both of these sources of supervision require considerable human effort to obtain: reward functions must be engineered by hand, which can be highly non-trivial in environments with natural observations, and demonstrations must be provided manually for each task. When using high-capacity models, hundreds or even thousands of demonstrations may be required for each task (Zhang et al. [12], Rahmatizadeh et al. [13], Rajeswaran et al. [14], Duan et al. [15]). In this paper, we instead aim to learn general-purpose policies that can flexibly accomplish a wide range of user-specified tasks, using data that is not task-specific and is easy to collect. Our model can in principle use *any* past experience for training, but the particular data collection approach we used is based on human-provided play data.

In order to distill non task-specific experience into a general-purpose policy, we set up our model to be conditioned on the user-specified goal. Goal conditioned policies have been explored extensively in the literature for reinforcement learning [6, 16, 17, 18, 19, 20], as well as for control via inverse models [21, 22, 23, 24]. Learning powerful goal-conditioned policies with reinforcement learning can produce policies with good long-horizon performance, but is difficult in terms of both the number of samples required and the need for extensive on-policy exploration [25, 26]. We instead opt to train our model with supervised learning. This introduces a major challenge, since the distribution over actions that can reach a temporally distant goal from the current state based on the data can be highly multimodal. Even single-task imitation models of this sort must contend with multimodality [27], and goal-conditioned models are typically restricted to short and relatively simple tasks, such as pushing [21], re-positioning rope [22], or short-distance navigation [28]. We tackle substantially more temporally extended tasks, using our proposed latent plan model, which models the multimodality explicitly using a hierarchical latent variable model. Hausman et al. [29] similarly learn a continuous latent space of closely related manipulation skills, instead learning the space with a discrete set of reinforcement learned tasks, defined by per-task rewards. In contrast to prior work on few-shot learning from demonstration ([30, 31]), our method does not require a meta-training phase, any expensive task-specific demonstrations, or a predefined task distribution. In contrast to prior work that uses reinforcement learning (Paine et al. [32]), it does not require any reward function or costly RL phase. Finally, Nachum et al. [33] derive a similar architecture to Play-LMP, justifying it in a hierarchical reinforcement learning setting.

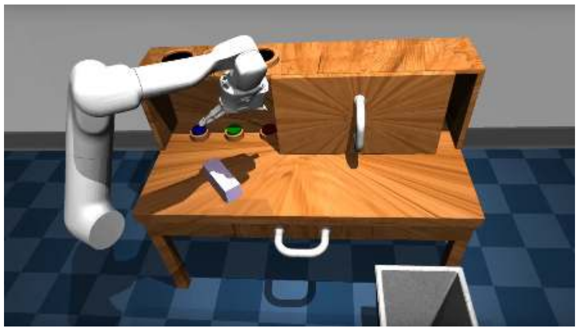


Figure 3: **The Playground environment.** Details in A.3.1 in [8]

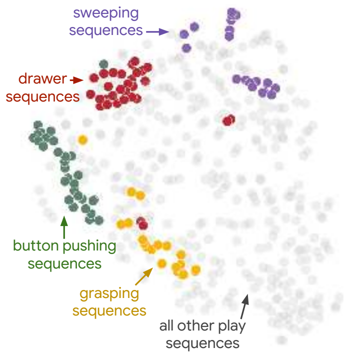


Figure 4: **Latent plan space t-SNE**

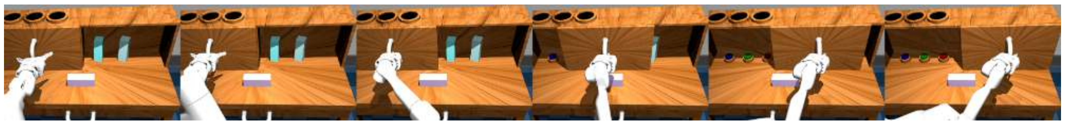


Figure 5: **Example of a supervised demonstration** sequence labeled and segmented for the ”sliding” task.

3 Learning Task-Agnostic Control from Play Data

First we give a definition of the type of data we collect for our experiments and our assumptions about its collection. We create a simulated “playground environment” (Fig. 3 and A.3.1 in [8]) for play collection and task evaluation. In this environment an 8-DOF simulated robot (arm and gripper) is situated in front of a desk with a sliding door and a drawer. On the desk is a rectangular block and 3 buttons that control lights. See an example of a play sequence in that environment in A.3.2 in [8].

What is Play? We propose play data is generated as follows: A human operator, given the current state of the world s_c, formulates a mental image of a goal state they would like to reach next s_g, driven by their curiosity or some other intrinsic motivation. For example, in an environment with a ball and cup sitting next to one other, the operator might choose s_g representing “ball in cup”. Given the current state s_c (“ball next to cup”), and goal state s_g (“ball in cup”), the operator considers all the different high-level behaviors b that would achieve the goal. E.g., “place ball in cup”, “toss ball in cup”, “bounce ball into cup” would all lead to s_g. We can consider a prior distribution over all the valid ways of reaching s_g from s_c, $p(b|s_c, s_g)$, a behavioral repertoire encoding knowledge of object affordances and environment dynamics. To actually achieve the desired outcome, they sample a single high-level behavior plan from the distribution $b \sim p(b|s_c, s_g)$ and execute it, producing the observed stream of low level state and action logs. We emphasize that play is not arbitrary behavior, nor “random” actions, but rather the very deliberate goal-conditioned behavior that a person engages in under their own direction.

3.1 Play-Supervised Goal-Conditioned Behavioral Cloning

We now describe “play-supervised goal-conditioned behavioral cloning”, or Play-GCBC, a method that extracts goal-conditioned policies using self-supervision on top of raw unlabeled play data. Let D be a play dataset, the unsegmented stream of high-dimensional sensory observations and actions logged during teleoperation play. D consists of paired $(O_t,\ a_t)$ tuples $D = \{(O_1, a_1), \cdots, (O_T, a_T)\}$. O_t is the set of observations from each of the robot’s N sensory channels $\{o^1, ..., o^N\}_t$ at time t and a_t is the logged teleoperation action. In our experiments, $O = \{I, p\}$ consists of I, an RGB image observation from a fixed first-person viewpoint, and p, the internal 8-DOF proprioceptive state of the agent. See A.2 in [8] for details.

The key idea behind Play-GCBC is that a random window of (observation, action) pairs extracted from play describes exactly how the robot got from a particular initial state to a particular final state. Furthermore, it is guaranteed that the final state is reachable from the initial state under the intervening actions. We can exploit this simple structure to create a self-supervised labeling scheme for a goal-conditioned policy, treating the initial state of a random sequence as “current state”, the final state as “reachable goal state”, and the actions taken as the labels to reproduce.

Encoding perceptual inputs: Since our logs are raw observations, we define one encoder per sensory channel $\Phi = \{E_1, ..., E_N\}$ with parameters θ_Φ, mapping N high-dimensional observations in each O per timestep to one low-dimensional fused state $s_t = concat([E_1(o_1), ..., E_N(o_N)])$. For simplicity we refer to this operation as $s_t \leftarrow \Phi(O_t)$.

Goal-conditioned policy: Let $\pi_{GCBC}(a_t|s_t, s_g)$ be a stochastic RNN goal-conditioned policy with parameters θ_{GCBC}, mapping from current state s_t and goal state s_g to the parameters of a distribution over next action a_t. We train Play-GCBC on batches of random play sequences as follows: For each training batch, and each batch sequence element: we sample a κ-length sequence of observations and actions τ. We extract the final observation in τ as the synthetic goal state and encode it. At each timestep t in τ, π_{GCBC} takes as input the current state $s_t \leftarrow \Phi(O_t)$ and goal state s_g, and maps to the parameters of a distribution over next action a_t. Both the encoders and the policy are trained end-to-end to maximize the log likelihood of each action taken during the sampled play sequence.

$$\mathcal{L}_{GCBC} = -\frac{1}{\kappa} \sum_{t=k}^{k+\kappa} log\big(\pi_{GCBC}(a_t|s_t, s_g)\big) \tag{1}$$

We describe the minibatch training pseudo-code in Algorithm 1.

Multimodality problem: A challenge in self-supervising control on top of play is that in general, there are many valid high-level behaviors that might connect the same (s_c, s_g) pair. This presents multiple counteracting action label trajectories, which can impede learning. Therefore, policies must be expressive enough to model all possible high-level behaviors that lead to the same goal outcome.

3.2 Play-supervised Latent Motor Plans

Motivation. We propose that unsupervised representation learning is well poised to address the multimodality problem. Consider the following as motivation: if we could learn compact representations of all the different high-level plans that take an agent from a current state to goal state (essentially learning $p(b|s_c, s_g)$) and condition a policy on a single sampled plan, we could convert a multimodal policy learning problem into a unimodal one. That is, a policy previously tasked with a difficult multimodal *plan inference* problem would now be relieved of that problem, and free to use the entirety of its capacity for unimodal *plan execution*. Ideally, individual points in the space correspond to reusable common behaviors executed during teleoperation play. But how do we learn good representations of high-level behavior unsupervised? Furthermore, how would we connect plan representation learning to our main goal of extracting goal-conditioned policies?

Plan Representation Learning Leads to Goal-Conditioned Control. We turn to the widely influential variational autoencoder (VAE) ([34] framework to learn plans from play. VAEs combine latent representation learning with deep generative models of observed data. Interestingly, we find that by starting with a pure plan representation learning problem and respecting the fact that plans depend on observed current and goal state, the generative decoder part of the model becomes equivalent to a goal and plan-conditioned policy. See A.1.1 in [8] for discussion.

We call this method "Play-supervised Latent Motor Plans", or Play-LMP, a unified objective for learning reusable plan representations and task-agnostic control from unlabeled play data. Formally, Play-LMP is a conditional sequence-to-sequence VAE (seq2seq CVAE) (Sohn et al. [35], Bowman et al. [36]), autoencoding random experiences extracted from play through a latent plan space.

As a CVAE, Play-LMP consists of three components trained end-to-end: 1) Plan recognition: a stochastic sequence encoder, taking a randomly sampled play sequence τ as input and mapping it to a distribution $q_\phi(z|\tau)$ in latent plan space, the learned variational posterior. 2) Plan proposal: a stochastic encoder taking the initial state s_c and final state s_g from the same sequence τ, outputting $p_\theta(z|s_c, s_g)$, the learned conditional prior. The goal of this encoder is to represent the full distribution over *all* high-level behaviors that might connect current and goal state, potentially capturing multiple solutions. 3) Plan and goal-conditioned policy: A policy conditioned on the current state s_c, goal state s_g and latent plan z sampled from the posterior $q_\phi(z|\tau)$, trained to reconstruct the goal-directed actions taken during play, following inferred plan z.

Like Play-GCBC, Play-LMP takes as input batches of randomly sampled play sequences τ and is trained as follows:

Plan encoding. For each training batch, and each batch sequence element τ: We first map the sequence of raw observations in τ to a sequence of encoded states, using perceptual encoders Φ: $\tau* = \Phi(\tau)$. V_{enc} ("video encoder"), a bidirectional sequence encoder with parameters θ_V, implements the posterior, taking preprocessed $\tau*$ as input and mapping it to the parameters of a distribution in latent plan space: $\mu_\phi, \sigma_\phi = V_{enc}(\tau*)$. As is typical with training VAEs, we assume the encoder has a diagonal covariance matrix, i.e. $z \sim \mathcal{N}(\mu_\phi, \text{diag}(\sigma_\phi^2))$. Individual latent plans z are sampled from this distribution at training time via the "reparameterization trick" (Kingma and Welling [34]) and handed to a latent plan and goal conditioned action decoder (described next) to be decoded into reconstructed actions[1].

Plan prior matching. We simultaneously extract synthetic "current" and "goal" states from the same sequence τ that V_{enc} just encoded: $s_c \leftarrow \Phi(O_t)$ and $s_g \leftarrow \Phi(O_{t+\kappa})$. We define CG_{enc} ("current, goal encoder"), to be a feedforward network with parameters θ_{CG} implementing the learned conditional prior. CG_{enc} takes concatenated s_c and s_g, and outputs the parameters of a distribution in latent plan space: $\mu_\psi, \sigma_\psi = CG_{enc}(s_c, s_g)$. V_{enc} and CG_{enc} are trained jointly by minimizing the KL divergence between their predicted distributions:

$$\mathcal{L}_{\text{KL}} = \text{KL}\Big(\mathcal{N}(z|\mu_\phi, \text{diag}(\sigma_\phi^2)) \;||\; \mathcal{N}(z|\mu_\psi, \text{diag}(\sigma_\psi^2))\Big) \tag{2}$$

Intuitively, $\mathcal{L}_{\text{KL}}$ forces the distribution over plans output by CG_{enc} to place high probability on actual latent plans recognized during play by V_{enc} .

Plan decoding. Finally we define π_{LMP} , a stochastic RNN with parameters θ_{LMP}. π_{LMP} takes as input current state s_t, goal state s_g, and a sampled latent plan z, and outputs the parameters of a distribution in the agent's action space A. The purpose of π_{LMP} is both to act as a decoder in a representation learning context and a goal and plan-conditioned policy in a task-agnostic control context. We note that by taking plan z as input, the policy is relieved from having to represent multiple high-level plans implicitly, aligning well with the original motivation. We compute the action reconstruction cost as follows: For each timestep t in the input sequence τ, we feed in $s_t \leftarrow \Phi(O_t)$, s_g, and z to π_{LMP} , which outputs the parameters for a probability distribution over observed action a_t. We compute the maximum likelihood action reconstruction loss[2] for each timestep:

$$\mathcal{L}_\pi = -\frac{1}{\kappa}\sum_{t=k}^{k+\kappa} log\big(\pi_{LMP}(a_t|s_t, s_g, z)\big) \tag{3}$$

Gradients from this loss are backpropagated through π_{LMP} , the reparameterized sampling operation, V_{enc} , and encoders Φ, optimizing the entire architecture end-to-end. The full Play-LMP training objective is:

$$\mathcal{L}_{LMP} = \mathcal{L}_\pi + \beta\mathcal{L}_{\text{KL}} \tag{4}$$

Note that following Higgins et al. [37], we introduce a weight β, controlling $\mathcal{L}_{\text{KL}}$'s contribution to the total loss. Setting β <1 was sufficient to avoid "posterior collapse" (Bowman et al. [36]), a commonly identified problem in VAE training in which an over-regularized model combined with a powerful decoder tends to ignore the latent variable z. We describe the full Play-LMP minibatch training pseudocode in Algorithm 2.

Task-agnostic control at test time. Here we describe how Play-LMP solves user-provided manipulation tasks at test time. At the beginning of each test episode, the agent starts in some current state O_c and receives a perceptual human-provided goal O_g. Both are encoded in state space ($s_c \leftarrow \Phi(O_c)$, $s_g \leftarrow \Phi(O_g)$), concatenated, and fed to the learned conditional prior, CG_{enc} , which outputs a distribution over high-level latent behavior plans z that should take the agent from s_c to s_g. The agent samples a single plan z from the distribution, then decodes it in closed loop in the environment. At each timestep of the decoding, the agent feeds (s_t, s_g, z) to π_{LMP} , a low level action is sampled $a_t \sim \pi_{LMP}(a_t|s_t, s_g, z)$. We allow the agent to "replan" by inferring and sampling new latent plans every κ timesteps (matching the average planning horizon it was trained with). In our experiments, our agent gets observations and takes low-level actions at 30hz. We set κ to 32, meaning that the agent replans at roughly 1hz. See Fig. 1b for details.

[1]We note that V_{enc} is only used at training time to help learn a latent plan space, and is discarded at test time. While we could in principle use V_{enc} at test time to perform full sequence imitation, in this work we restrict our attention to tasks specified by individual user-provided goal states.

[2]We can optionally also have the decoder output state predictions, and add another loss term penalizing a state reconstruction loss.

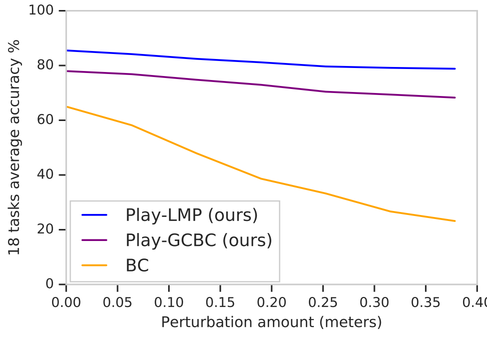
4 Experiments

In our experiments, we aim to answer the following questions: 1) Can a single play-supervised policy generalize to a wide variety of user-specified visual manipulation tasks, despite not being trained on task-specific data? 2) Are play-supervised models trained on cheap to collect play data (LfP) competitive with specialist models trained on expensive expert demonstrations for each task (LfD)? 3) Does decoupling latent plan inference and plan decoding into independent problems, as is done in Play-LMP, improve performance over goal-conditioned Behavioral Cloning (Play-GCBC), (which does no explicit latent plan inference)?

Tasks and Dataset: We define 18 visual manipulation tasks (see A.3.3 in [8]) in the same environment that play was collected in (Fig. 3 and A.3.1 in [8]). To compare our play-supervised models to a conventional scenario, we collect a training set of 100 expert demonstrations per task in the environment, and train one behavioral cloning policy (**BC**, details in A.1.2 in [8]) on the corresponding expert dataset. This results in 1800 demonstrations total or ∼1.5 hours of expert data. We additionally train a single multi-task behavioral cloning baseline conditioned on state and task id, **Multitask BC** (Rahmatizadeh et al. [27]), trained on all 18 BC expert demonstration datasets. We collect play datasets (example in A.3.2 in [8]) of various sizes as training data for **Play-LMP** and **Play-GCBC**, up to ∼7 hours total. We define two sets of experiments over these datasets: **pixel experiments**, where we study the multi-task visual manipulation problem, and **state experiments**, where we ignore the visual representation learning problem and provide all models with ground truth states (positions and orientations of all objects in the scene) as observations. The motivation of the state experiments is to understand the how all methods compare on the control problem independent of visual representation learning, which could potentially be improved independently via other self-supervised methods e.g. Sermanet et al. [38].

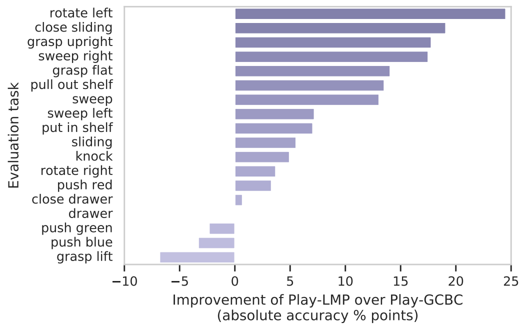
	training data		
Method	**labels**	**input**	**success %**
BC	labeled	pixels	66.5% ± 12.1
Play-GCBC (ours)	unlabeled	pixels	58.7% ± 11.6
Play-LMP (ours)	unlabeled	pixels	**69.4%** ±10.8
BC	labeled	states	70.3%
Multitask BC	labeled	states	66.2%
Play-GCBC (ours)	unlabeled	states	77.9%
Play-LMP (ours)	unlabeled	states	**85.5%**

(a) **18-task success.**



(b) **Robustness to variations.**

Figure 6: **Quantitative task success and robustness**. (a) Play-LMP consistently outperforms the baselines, whether trained on groundtruth states or directly on pixels. Success is reported with confidence intervals over 3 seeded training runs for pixel experiments. (b) models trained on play data are more robust to perturbations to the initial position. See Sec. 4 for details.



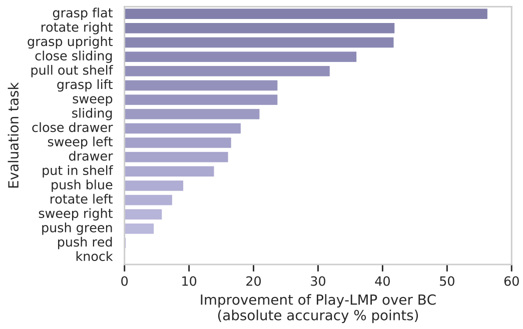


Figure 7: **Improvement per task of Play-LMP** over Play-GCBC (left) and BC baselines (right), in absolute percentage points of accuracy (model trained on states).

Task success with play-supervision: We present our main findings for both experiments in Table 6a. First, we find that despite not being trained on task labels, a single Play-LMP policy outperforms the 18 specialized and supervised BC models (answering experimental questions 1 and

2). Additionally, we find that the decoupling happening in Play-LMP compared to Play-GCBC is beneficial and yields systematic improvements in performance. We report in Fig. 7 the absolute improvement per task in percentage points of Play-LMP over the baselines, with up to 50 points of improvement.

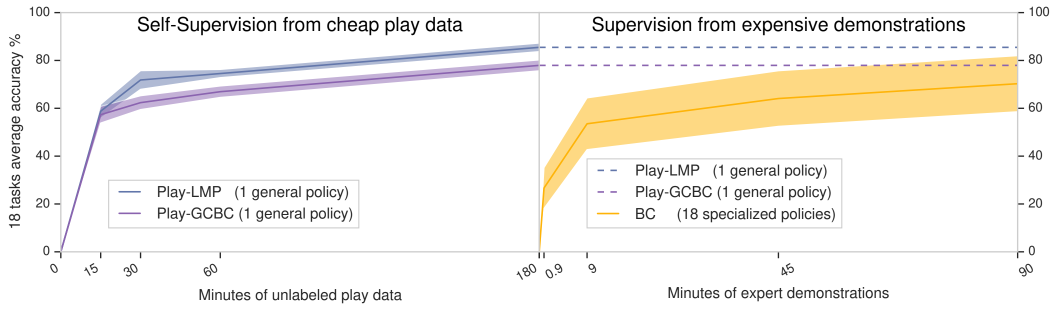


Figure 8: **18-tasks average success** for self-supervised models trained on various amounts of cheap play data (left) vs. expert-trained models trained on expensive task demonstrations (right). A single task-agnostic Play-LMP policy, trained on unlabeled play data generalizes with 85.5% success to the 18 test-time tasks with no finetuning, outperforming a collection of 18 expert-trained BC policies who reach 70.3% average success. This holds true even when Play-LMP is artificially restricted to only 30 minutes of play data (71.8%), despite play being easier and cheaper to collect than expert demonstrations. These data ablation numbers were obtained from models trained on ground truth state observations. Shaded regions indicate 95% confidence intervals over 20 rollouts. See Sec. 4 for details.

Scalability: We see in Fig. 8 that even when trained on only 30 minutes of play data, individual Play-LMP policies outperform 18 BC policies trained on 90 minutes of expert task-specific demonstrations. We feel this highlights the scalability and generality of the approach—that models trained only on random windows extracted from play are prepared for specific tasks presented to them at test time. We believe this comparison is fair for two reasons: 1) the baseline gets 3x more training data, 2) the baseline training data consists of curated task-specific demonstrations of optimal behavior, whereas there is no guarantee that 30 minutes of play data contains optimal task demonstrations.

Robustness: In Fig. 6b, we find that models trained on play data (Play-LMP and Play-GCBC) are significantly more robust to perturbations than the model trained on expert demonstrations only (BC), a phenomenon we attribute to the inherent coverage properties of play data over demonstration data. More details in A.4.1 in [8].

Unsupervised task discovery: We investigate the latent plan spaced learned by Play-LMP, seeing whether or not it is capable of encoding task information despite not being trained with task labels. In Fig. 4 we embed 512 randomly selected windows from the play dataset as well as all validation task demonstrations, using the Φ plan recognition model. Surprisingly, we find that despite not being trained explicitly with task labels, Play-LMP appears to organize its latent plan space functionally. E.g. we find certain regions of space all correspond to drawer manipulation, while other regions correspond to button manipulation.

Emergent Retrying: We find qualitative evidence that play-supervised models, unlike models trained solely on expert demonstrations, make multiple attempts to retry the task after initial failure. See A.4.2 in [8].

5 Conclusion

In this work, we advocate for learning the full continuum of tasks using unlabeled play data, rather than discrete tasks using expert demonstrations. We introduce a self-supervised plan representation learning algorithm able to discover task semantics despite never seeing any task labels. By learning to generate actions for its task-agnostic policy, the model is able to train an entire deep sensory stack from scratch. We showed that play brings scalability to data collection, as well as robustness to the models trained with it. We explore the setting where play data and test-time tasks are defined over the same playroom environment. Future work includes exploring whether generalization is possible to novel objects or novel environments, as well as exploring the effects of imbalance in play data distributions as discussed in A.5 in [8].

Acknowledgements
We thank Michael Wu for designing and iterating on our playground environment. We would also like to thank Karol Hausmann, Ben Poole, Eric Jang for many fruitful discussions on the design and training of our model.

References

[1] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, et al. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *arXiv preprint arXiv:1806.10293*, 2018.

[2] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

[3] P. Pastor, H. Hoffmann, T. Asfour, and S. Schaal. Learning and generalization of motor skills by learning from demonstration. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 763–768. IEEE, 2009.

[4] F. Ebert, C. Finn, S. Dasari, A. Xie, A. Lee, and S. Levine. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control. *arXiv preprint arXiv:1812.00568*, 2018.

[5] D. Warde-Farley, T. V. de Wiele, T. Kulkarni, C. Ionescu, S. Hansen, and V. Mnih. Unsupervised control through non-parametric discriminative rewards. *CoRR*, abs/1811.11359, 2018. URL `http://arxiv.org/abs/1811.11359`.

[6] L. P. Kaelbling. Learning to achieve goals. In *IJCAI*, pages 1094–1099. Citeseer, 1993.

[7] S. Ross, G. J. Gordon, and J. A. Bagnell. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011. URL `https://arxiv.org/pdf/1011.0686.pdf`.

[8] C. Lynch, M. Khansari, T. Xiao, V. Kumar, J. Tompson, S. Levine, and P. Sermanet. Learning latent plans from play. *CoRR*, abs/1903.01973, 2019. URL `http://arxiv.org/abs/1903.01973`.

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

[10] J. Kober, J. A. Bagnell, and J. Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

[11] M. P. Deisenroth, G. Neumann, J. Peters, et al. A survey on policy search for robotics. *Foundations and Trends® in Robotics*, 2(1–2):1–142, 2013.

[12] T. Zhang, Z. McCarthy, O. Jow, D. Lee, K. Goldberg, and P. Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. *CoRR*, abs/1710.04615, 2017. URL `http://arxiv.org/abs/1710.04615`.

[13] R. Rahmatizadeh, P. Abolghasemi, L. Bölöni, and S. Levine. Vision-based multi-task manipulation for inexpensive robots using end-to-end learning from demonstration. *CoRR*, abs/1707.02920, 2017. URL `http://arxiv.org/abs/1707.02920`.

[14] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

[15] Y. Duan, M. Andrychowicz, B. C. Stadie, J. Ho, J. Schneider, I. Sutskever, P. Abbeel, and W. Zaremba. One-shot imitation learning. *CoRR*, abs/1703.07326, 2017. URL `http://arxiv.org/abs/1703.07326`.

[16] A. V. Nair, V. Pong, M. Dalal, S. Bahl, S. Lin, and S. Levine. Visual reinforcement learning with imagined goals. In *Advances in Neural Information Processing Systems*, pages 9209–9220, 2018.

[17] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. P. Abbeel, and W. Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, pages 5048–5058, 2017.

[18] A. Levy, R. P. Jr., and K. Saenko. Hierarchical actor-critic. *CoRR*, abs/1712.00948, 2017. URL `http://arxiv.org/abs/1712.00948`.

[19] P. Rauber, F. Mutz, and J. Schmidhuber. Hindsight policy gradients. *CoRR*, abs/1711.06006, 2017. URL `http://arxiv.org/abs/1711.06006`.

[20] S. Cabi, S. G. Colmenarejo, M. W. Hoffman, M. Denil, Z. Wang, and N. de Freitas. The intentional unintentional agent: Learning to solve many continuous control tasks simultaneously. *CoRR*, abs/1707.03300, 2017. URL `http://arxiv.org/abs/1707.03300`.

[21] P. Agrawal, A. Nair, P. Abbeel, J. Malik, and S. Levine. Learning to poke by poking: Experiential learning of intuitive physics. *CoRR*, abs/1606.07419, 2016. URL http://arxiv.org/abs/1606.07419.

[22] A. Nair, D. Chen, P. Agrawal, P. Isola, P. Abbeel, J. Malik, and S. Levine. Combining self-supervised learning and imitation for vision-based rope manipulation. *CoRR*, abs/1703.02018, 2017. URL http://arxiv.org/abs/1703.02018.

[23] P. Christiano, Z. Shah, I. Mordatch, J. Schneider, T. Blackwell, J. Tobin, P. Abbeel, and W. Zaremba. Transfer from simulation to real world through learning deep inverse dynamics model. *arXiv preprint arXiv:1610.03518*, 2016.

[24] F. Torabi, G. Warnell, and P. Stone. Behavioral cloning from observation. *CoRR*, abs/1805.01954, 2018. URL http://arxiv.org/abs/1805.01954.

[25] L. Pinto and A. Gupta. Supersizing Self-supervision: Learning to Grasp from 50K Tries and 700 Robot Hours. *IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[26] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen. Learning Hand-Eye Coordination for Robotic Grasping with Deep Learning and Large-Scale Data Collection. *International Journal of Robotics Research*, 2017.

[27] R. Rahmatizadeh, P. Abolghasemi, L. Boloni, and S. Levine. Vision-based multi-task manipulation for inexpensive robots using end-to-end learning from demonstration. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3758–3765. IEEE, 2018.

[28] D. Pathak, P. Mahmoudieh, G. Luo, P. Agrawal, D. Chen, Y. Shentu, E. Shelhamer, J. Malik, A. A. Efros, and T. Darrell. Zero-shot visual imitation. In *ICLR*, 2018.

[29] K. Hausman, J. T. Springenberg, Z. Wang, N. Heess, and M. Riedmiller. Learning an embedding space for transferable robot skills. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rk07ZXZRb.

[30] C. Finn, T. Yu, T. Zhang, P. Abbeel, and S. Levine. One-shot visual imitation learning via meta-learning. *arXiv preprint arXiv:1709.04905*, 2017.

[31] Z. Wang, J. S. Merel, S. E. Reed, N. de Freitas, G. Wayne, and N. Heess. Robust imitation of diverse behaviors. In *Advances in Neural Information Processing Systems*, pages 5320–5329, 2017.

[32] T. L. Paine, S. G. Colmenarejo, Z. Wang, S. E. Reed, Y. Aytar, T. Pfaff, M. W. Hoffman, G. Barth-Maron, S. Cabi, D. Budden, and N. de Freitas. One-shot high-fidelity imitation: Training large-scale deep nets with RL. *CoRR*, abs/1810.05017, 2018. URL http://arxiv.org/abs/1810.05017.

[33] O. Nachum, S. Gu, H. Lee, and S. Levine. Near-optimal representation learning for hierarchical reinforcement learning. *CoRR*, abs/1810.01257, 2018. URL http://arxiv.org/abs/1810.01257.

[34] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[35] K. Sohn, H. Lee, and X. Yan. Learning structured output representation using deep conditional generative models. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 3483–3491. Curran Associates, Inc., 2015.

[36] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio. Generating sentences from a continuous space. 2016.

[37] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. β-VAE: Learning basic visual concepts with a constrained variational framework. *International Conference on Learning Representations (ICLR)*, 2017.

[38] P. Sermanet, C. Lynch, Y. Chebotar, J. Hsu, E. Jang, S. Schaal, and S. Levine. Time-contrastive networks: Self-supervised learning from video. *International Conference in Robotics and Automation (ICRA)*, 2018. URL http://arxiv.org/abs/1704.06888.

[39] T. Salimans, A. Karpathy, X. Chen, and D. P. Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *CoRR*, abs/1701.05517, 2017. URL http://arxiv.org/abs/1701.05517.

[40] V. Kumar and E. Todorov. Mujoco haptix: A virtual reality system for hand manipulation. In *Humanoid Robots (Humanoids), 2015 IEEE-RAS 15th International Conference on*, pages 657–663. IEEE, 2015.

[41] A. van den Oord, O. Vinyals, et al. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, pages 6306–6315, 2017.