LEARNING LATENT PLANS FROM PLAY DATA

Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, & Pierre Sermanet
Google Brain
{coreylynch,khansari,tedxiao,vikashplus,tompson,slevine}@google.com

ABSTRACT

We propose learning from teleoperated play data as a way to scale up multi-task robotic skill learning. **Learning from play (LfP)** offers three main advantages: 1) It is cheap. Large amounts of play data can be collected quickly as it does not require scene staging, task segmenting, or resetting to an initial state. 2) It is general. It contains both functional and non-functional behavior, relaxing the need for a predefined task distribution. 3) It is rich. Play involves repeated, varied behavior and naturally leads to high coverage of the possible interaction space. These properties distinguish play from expert demonstrations, which are rich, but expensive, and scripted unattended data collection, which is cheap, but insufficiently rich. Variety in play, however, presents a multimodality challenge to methods seeking to learn control on top. To this end, we introduce **Play-LMP** (play-supervised latent motor plans), a method designed to handle variability in the LfP setting by organizing it in an embedding space. Play-LMP jointly learns 1) reusable latent plan representations unsupervised from play data and 2) a single goal-conditioned policy capable of decoding inferred plans to achieve user-specified tasks. We show empirically that Play-LMP, despite not being trained on task-specific data, is capable of generalizing to 18 complex user-specified manipulation tasks with average success of 85.5%, outperforming individual models trained on expert demonstrations (success of 70.3%). Furthermore, we find that play-supervised models, unlike their expert-trained counterparts, 1) are more robust to perturbations and 2) exhibit retrying-till-success. Finally, despite never being trained with task labels, we find that our agent learns to organize its latent plan space around functional skills. Videos of our experiments are available at https://sites.google.com/corp/view/sslmp

1 INTRODUCTION

Learning from play is a fundamental and general method humans use to acquire a repertoire of complex skills and behaviors (Wood & Attfield (2005)).

In this work, we propose *learning from play data* (LfP), or *"play-supervision"*, as a way to scale up multi-task robotic skill learning. A human operator teleoperates the robot in a playground environment, interacting with all the objects available in as many ways that they can think of. Humans provide the necessary properties of curiosity, boredom, and affordance priors to guide rich object play. After this is collected, we learn goal-conditioned control self-supervised on top. Examples of the play data fed into our system are shown in Fig. 3. We underline that this data is not task specific, but rather intends to cover as much as possible of the full object interaction space allowed by the environment.

Teleoperation play data is 1) *Cheap*: it involves no scene staging, task segmenting, or resetting. 2) *General*: It relaxes the need for a discrete, predefined task distribution. 3) *Rich*. Play stresses "means over ends" and naturally lead to high coverage of the possible interaction space. In this way, LfP compares favorably to both expert demonstrations–which are rich, but not scalable, and scripted collection–which is highly scalable, but insufficiently rich to learn complex manipulation.

However, the "repeat, but vary" property of play interactions presents a multimodal representation learning challenge to methods seeking to learn control on top. Policies must be expressive enough

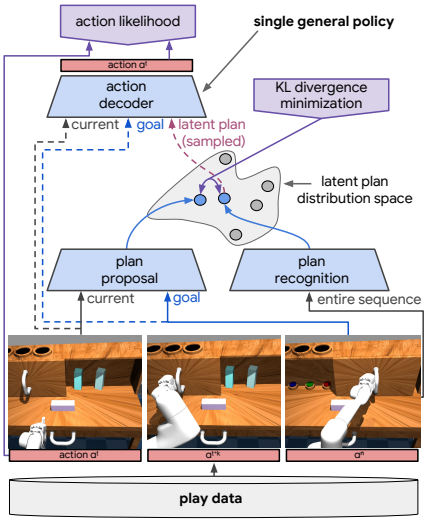

Figure 1: **Play-LMP:** A single model that self-supervises control from play data, then generalizes to a wide variety of manipulation tasks. Play-LMP learns to recognize and organize a repertoire of behaviors executed during play in a **latent plan space**, then reuse them to achieve user-specified goals.

to account for all the possible ways to reach a given goal. Our approach, described in 2.2, models this variation explicitly, by learning to recognize a repertoire of reusable behaviors from play unsupervised and organize them in an embedding space. In this paper, we introduce the following contributions: 1) Learning from play (LfP), or "play-supervision", a paradigm for scaling up multi-task robotic skill learning by self-supervising on cheap and rich user teleoperated play data. We show empirically its benefits over learning from segmented demonstrations (LfD), especially in regards to scalability, robustness to perturbations, and failure recovery. 2) Play-LMP, a method that jointly learns reusable latent plan representations from play data and goal-conditioned control, capable of generalizing to a wide variety of complex user-specified manipulation tasks.

2 METHOD

2.1 PLAY DATA

Consider play data, an unbounded sequence of states and actions corresponding to self-guided, repeated, non-stereotyped object interaction between an agent and it's environment.

$$\mathcal{D} = \{(s_1, a_1), (s_2, a_2), \cdots, (s_T, a_T)\} \tag{1}$$

In our experiments, we define play data as the states and actions logged during human play teleoperation of a robot in a playground environment. Find an example of such data in Fig. 3.

2.2 PLAY-LMP

The presence of multiple action trajectories for the same (current state, goal state) pair presents a challenge to models seeking to learn goal-conditioned control in the form of counteracting action labels. This can be considered a multimodal representation learning problem: policies must be powerful enough to model all possible high-level behaviors that lead to the same goal outcome.

With this motivation in mind, we introduce Play-LMP (play-supervised latent motor plans), a hierarchical latent variable model for learning goal-conditioned control. Play-LMP simultaneously learns 1) *reusable latent plan representations* from play data and 2) *plan and goal-conditioned policies*, capable of decoding learned latent plans into actions to reach user-specified goal states. We call the representation space learned by Play-LMP "latent plan space". The intent is that individual points in

the space correspond to behaviors recognized during play that got the agent from some initial state to some final state. Local regions of plan space should correspond to distinct solutions to the same task. In this way, we aim for Play-LMP to explicitly model the "multiple solutions" problem in play data, relieving the policy of that effort. At a high level, the architecture consists of three modules: a plan recognizer, a plan proposer, and a goal and plan conditioned policy. We now describe each of the modules in detail and the losses used to train them.

2.2.1 PLAN RECOGNIZER

Consider a sequence of state action pairs τ of window length κ sampled at random from the play dataset $\mathcal{D}$:

$$\tau = \{(s_{k:k+\kappa}, a_{k:k+\kappa})\} \sim \mathcal{D} \tag{2}$$

We define a stochastic sequence encoder, Φ, which takes as input τ and outputs a distribution over plans in latent space. Concretely, Φ is a bidirectional RNN with parameters θ_Φ, mapping from τ to means and variances:

$$\mu_\Phi, \sigma_\Phi = \Phi(\tau; \theta_\Phi) \tag{3}$$

The purpose of this network is to act as "plan recognition", identifying which region of latent space the behavior executed during play corresponds to. At training time, an individual latent plan z is sampled from this distribution via the "reparameterization trick" (Kingma & Welling (2013)) and handed to a plan and goal conditioned policy (described in 2.4) to be decoded into actions.

2.3 PLAN PROPOSER

We also define a stochastic plan proposal network, Ψ, which maps initial state s_i and goal state s_g to a distribution over latent plans. Concretely, Ψ is an MLP with parameters θ_Ψ, mapping from concatenated s_i and s_g to means and variances in the same latent plan space as Φ[1]:

$$\mu_\Psi, \sigma_\Psi = \Psi(s_i, s_g; \theta_\Psi) \tag{4}$$

The goal of this network is to learn to represent the full distribution of possible behaviors that an agent could execute to get from a particular initial state to a particular goal state.

Φ and Ψ are co-trained by minimizing the KL divergence between the two distributions:

$$\mathcal{L}_{\text{KL}} = \text{KL}\Big(\mathcal{N}(z|\mu_\Phi, \text{diag}(\sigma_\Phi^2)) \;||\; \mathcal{N}(z|\mu_\Psi, \text{diag}(\sigma_\Psi^2))\Big) \tag{5}$$

Intuitively, $\mathcal{L}_{\text{KL}}$ forces the plan distribution output by the planner Ψ to place high probability on actual latent plans recognized during play. Simultaneously it enforces a regular geometry over codes output by the plan recognizer Φ, allowing plausible plans to be sampled at test time from regions of latent space that have high probability under the conditional prior Ψ.

2.4 TASK AGNOSTIC, GOAL AND LATENT PLAN CONDITIONED POLICY

Finally, we define a policy π, a stochastic RNN (parameterized by θ_π) that takes as input current state s_t, goal state s_g, and a sampled latent plan z, and outputs action a_t.

The policy is trained via a maximum likelihood loss $\mathcal{L}_\pi$ to reconstruct the actions in the sampled play sequence τ[2]:

[1]For simplicity, we choose a unimodal multivariate Gaussian to represent distributions in latent plan space; nothing in principle stops us from using more complicated distributions.

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

$$\mathcal{L}_\pi = -\frac{1}{\kappa}\sum_{t=k}^{k+\kappa} \ln\big(\pi(a_t|s_t, s_g, z)\big) \tag{6}$$

2.5 FULL OBJECTIVE

Following Higgins et al. (2017), we introduce a weight β, controlling $\mathcal{L}_{\text{KL}}$'s contribution to the total loss. Setting $\beta < 1$ was sufficient to avoid "posterior collapse" (Bowman et al. (2016)), a commonly identified problem in VAE training in which an over-regularized model combined with a powerful decoder tends to ignores the latent variable z. The full Play-LMP training objective is:

$$\mathcal{L}_{LMP} = \frac{1}{\kappa}\mathcal{L}_\pi + \beta\mathcal{L}_{\text{KL}} \tag{7}$$

We describe the full Play-LMP minibatch training pseudocode in Algorithm 1.

2.6 ZERO-SHOT CONTROL AT TEST TIME

At test time, we use Play-LMP to achieve user-specified manipulation goals in zero shot. Given current state s_c and user specified goal s_g, the agent infers a distribution over plans, samples one, then decodes it into actions to achieve the goal. Note that we allow the agent to resample new latent plans every κ steps (matching the planning horizon it was trained with). See Fig. 2 for details.

2.7 PLAY-GCBC

We also define Play-GCBC (play-supervised goal conditioned behavioral cloning). The training procedure is similar to Play-LMP, but has no explicit latent plan inference. We train an RNN, conditioned on current state s_t and goal state s_g to reconstruct actions in a sampled play sequence τ. See Algorithm 2 for full minibatch training pseudo-code.

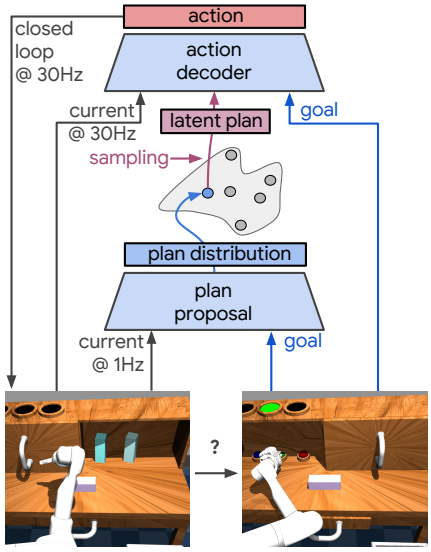

Figure 2: **Task-agnostic policy inference**. The policy is conditioned on a latent plan which is sampled once from a plan distribution (inferred from the current and goal states). The policy is also conditioned on the current state as well as the goal state desired by the user.

3 EXPERIMENTS

We explore whether Play-LMP , a single general purpose policy trained on non-task specific data, can generalize to 18 user-specified manipulation tasks. A complete description of each task is available in Appendix 6.1. We compare Play-LMP to 18 individual behavioral cloning policies, trained on expert demonstrations for each task. Additionally we investigate whether decoupling latent plan inference and plan decoding (Play-LMP) generalizes better than coupled plan and action inference (Play-GCBC).

In Fig. 5, we find that Play-LMP generalizes to 18 user-specified manipulation tasks with an average success of 85.5%, outperforming expert-trained demonstrations, who reach an average 70.3%. Additionally, we find that endowing play-supervised models with latent plan inference helps generalization to downstream tasks, with Play-LMP significantly outperforming Play-GCBC (average success of 85.5% vs. 78.4% respectively). Results are summarized in table 1.

Furthermore, we find that play-supervised models, unlike their expert-trained counterparts, show stronger robustness to perturbations (Appendix 6.2) and emergent "retry until success" behavior (Appendix 6.3).

Finally, we investigate the latent plan space learned by Play-LMP . Surprisingly, we find that despite never being trained explicitly with task labels, Play-LMP appears to organize its latent plan space functionally. See 7 for details.

4 RELATED WORK

Robotic learning methods generally require some form of supervision to acquire behavioral skills – conventionally, this supervision either consists of a reward signal, as in reinforcement learning Sutton & Barto (2018); Kober et al. (2013); Deisenroth et al. (2013), or demonstrations, as in imitation learning Pastor et al. (2009); Argall et al. (2009). Both of these sources of supervision require considerable human effort to obtain: reward functions must be engineered by hand, and demonstrations must be provided manually for each task (Zhang et al. (2017); Rahmatizadeh et al. (2017); Rajeswaran et al. (2017); Duan et al. (2017)). 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 method learns goal-conditioned control, which has been explored extensively in the literature for reinforcement learning Kaelbling (1993); Pong et al. (2018); Nair et al. (2018); Schaul et al. (2015); Andrychowicz et al. (2017); Levy et al. (2017); Rauber et al. (2017); Cabi et al. (2017); Sukhbaatar et al. (2017), as well as for control via inverse models Agrawal et al. (2016); Nair et al. (2017); Christiano et al. (2016); Torabi et al. (2018).

Our work on learning latent plans is most related to Hausman et al. (2018), who present a method for reinforcement learning of closely related manipulation skills, parameterized via an explicit skill embedding space. They assume a fixed set of initial tasks at training time, with access to accompanying per task reward functions to drive policy and embedding learning. Our method, in contrast, relies on unsegmented play data with no predefined task distribution.

Our self-supervised learning method for learning latent plans relates to other works in self-supervised representation learning from sequences Wang & Gupta (2015); Misra et al. (2016); Sermanet et al. (2018). It decouples high and low level planning to achieve better task generalization, a strategy well studied in the literature (Sermanet et al. (2009)).

Lastly, our work is related to prior research on few-shot learning of skills from demonstrations (Finn et al. (2017); Wang et al. (2017); James et al. (2017); Alet et al. (2018); Duan et al. (2017)). While our method does not require demonstrations to perform new tasks – only the goal state – it can readily incorporate demonstrations simply by treating each subsequent frame as a goal.

5 CONCLUSION

In this work, we emphasize the benefits of training a single, task-agnostic, goal-conditioned policy on unstructured, unsegmented play data, as opposed to training individual models from scratch for

each task. We stress that play data strikes a good balance on the cost-richness tradeoff, compared to expensive expert demonstrations and insufficiently rich scripted collection. We introduce a self-supervised plan representation learning and goal-conditioned policy learning algorithm, Play-LMP, designed to scale to a difficult behavioral cloning regime with large amount of natural variability in the data. Surprisingly we find that its latent plan space learns to embed task semantics despite never being trained with task labels. Finally we find that models trained on play data are far more robust to perturbation than models trained solely on positive demonstrations, and exhibit natural failure recovery despite not being trained explicitly to do so.

REFERENCES

Pulkit Agrawal, Ashvin Nair, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Learning to poke by poking: Experiential learning of intuitive physics. *CoRR*, abs/1606.07419, 2016. URL http://arxiv.org/abs/1606.07419.

Ferran Alet, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Modular meta-learning. *CoRR*, abs/1806.10166, 2018. URL http://arxiv.org/abs/1806.10166.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, pp. 5048–5058, 2017.

Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. 2016.

Serkan Cabi, Sergio Gomez Colmenarejo, Matthew W. Hoffman, Misha Denil, Ziyu Wang, and Nando 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.

Paul Christiano, Zain Shah, Igor Mordatch, Jonas Schneider, Trevor Blackwell, Joshua Tobin, Pieter Abbeel, and Wojciech Zaremba. Transfer from simulation to real world through learning deep inverse dynamics model. *arXiv preprint arXiv:1610.03518*, 2016.

Marc Peter Deisenroth, Gerhard Neumann, Jan Peters, et al. A survey on policy search for robotics. *Foundations and Trends® in Robotics*, 2(1–2):1–142, 2013.

Yan Duan, Marcin Andrychowicz, Bradly C. Stadie, Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. One-shot imitation learning. *CoRR*, abs/1703.07326, 2017. URL http://arxiv.org/abs/1703.07326.

Chelsea Finn, Tianhe Yu, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot visual imitation learning via meta-learning. *arXiv preprint arXiv:1709.04905*, 2017.

Karol Hausman, Jost Tobias Springenberg, Ziyu Wang, Nicolas Heess, and Martin Riedmiller. Learning an embedding space for transferable robot skills. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rk07ZXZRb.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. β-VAE: Learning basic visual concepts with a constrained variational framework. *International Conference on Learning Representations (ICLR)*, 2017.

Stephen James, Andrew J. Davison, and Edward Johns. Transferring end-to-end visuomotor control from simulation to real world for a multi-stage task. *CoRR*, abs/1707.02267, 2017. URL http://arxiv.org/abs/1707.02267.

Leslie Pack Kaelbling. Learning to achieve goals. In *IJCAI*, pp. 1094–1099. Citeseer, 1993.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

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

Andrew Levy, Robert Platt Jr., and Kate Saenko. Hierarchical actor-critic. *CoRR*, abs/1712.00948, 2017. URL http://arxiv.org/abs/1712.00948.

Ishan Misra, C Lawrence Zitnick, and Martial Hebert. Shuffle and learn: unsupervised learning using temporal order verification. In *European Conference on Computer Vision*, pp. 527–544. Springer, 2016.

Ashvin Nair, Dian Chen, Pulkit Agrawal, Phillip Isola, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Combining self-supervised learning and imitation for vision-based rope manipulation. *CoRR*, abs/1703.02018, 2017. URL http://arxiv.org/abs/1703.02018.

Ashvin V Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. In *Advances in Neural Information Processing Systems*, pp. 9209–9220, 2018.

Peter Pastor, Heiko Hoffmann, Tamim Asfour, and Stefan Schaal. Learning and generalization of motor skills by learning from demonstration. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pp. 763–768. IEEE, 2009.

Vitchyr Pong, Shixiang Gu, Murtaza Dalal, and Sergey Levine. Temporal difference models: Model-free deep rl for model-based control. *arXiv preprint arXiv:1802.09081*, 2018.

Rouhollah Rahmatizadeh, Pooya Abolghasemi, Ladislau Bölöni, and Sergey 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.

Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

Paulo Rauber, Filipe Mutz, and Jürgen Schmidhuber. Hindsight policy gradients. *CoRR*, abs/1711.06006, 2017. URL http://arxiv.org/abs/1711.06006.

Stéphane Ross, Geoffrey J Gordon, and J Andrew 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.

Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International Conference on Machine Learning*, pp. 1312–1320, 2015.

Pierre Sermanet, Raia Hadsell, Marco Scoffier, Matt Grimes, Jan Ben, Ayse Erkan, Chris Crudele, Urs Miller, and Yann LeCun. A multirange architecture for collision-free off-road robot navigation. *Journal of Field Robotics (JFR)*, 26(1):52–87, 2009. URL http://yann.lecun.com/exdb/publis/pdf/sermanet-jfr-09.pdf.

Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, and Sergey Levine. Time-contrastive networks: Self-supervised learning from video. *International Conference in Robotics and Automation (ICRA)*, 2018. URL http://arxiv.org/abs/1704.06888.

Sainbayar Sukhbaatar, Ilya Kostrikov, Arthur Szlam, and Rob Fergus. Intrinsic motivation and automatic curricula via asymmetric self-play. *CoRR*, abs/1703.05407, 2017. URL http://arxiv.org/abs/1703.05407.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. *CoRR*, abs/1805.01954, 2018. URL http://arxiv.org/abs/1805.01954.

Xiaolong Wang and Abhinav Gupta. Unsupervised learning of visual representations using videos. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2794–2802, 2015.

Ziyu Wang, Josh S Merel, Scott E Reed, Nando de Freitas, Gregory Wayne, and Nicolas Heess. Robust imitation of diverse behaviors. In *Advances in Neural Information Processing Systems*, pp. 5320–5329, 2017.

Elizabeth Wood and Jane Attfield. *Play, learning and the early childhood curriculum*. Sage, 2005.

Tianhao Zhang, Zoe McCarthy, Owen Jow, Dennis Lee, Ken Goldberg, and Pieter Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. *CoRR*, abs/1710.04615, 2017. URL http://arxiv.org/abs/1710.04615.

6 APPENDIX

Algorithm 1 Training Play-LMP

Input: Play data $\mathcal{D} : \{(s_1, a_1), \cdots, (s_T, a_T)\}$
Randomly initialize model parameters $\theta = \{\theta_\Phi, \theta_\Psi, \theta_\pi\}$
while not done **do**:
    Sample a sequence $\tau = \{(s_{k:k+\kappa}, a_{k:k+\kappa})\} \sim \mathcal{D}$
    Set current and goal state: $s_i \leftarrow s_k,\ s_g \leftarrow s_{k+\kappa}$
    Encode the sequence τ: $\mu_\Phi, \sigma_\Phi = \Phi(\tau; \theta_\Phi)$
    Encode s_i and s_g: $\mu_\Psi, \sigma_\Psi = \Psi(s_i, s_g; \theta_\Psi)$
    Compute KL loss using Eq. 5.
    Compute action loss using Eq. 6.
    Update θ by taking a gradient step to minimize Eq. 7.

Algorithm 2 Training Play-GCBC

Input: Play data $\mathcal{D} : \{(s_1, a_1), \cdots, (s_T, a_T)\}$
Randomly initialize model parameters θ_{GCBC}.
while not done **do**:
    Sample a sequence $\tau = \{(s_{k:k+\kappa}, a_{k:k+\kappa})\} \sim \mathcal{D}$
    Set current and goal state: $s_i \leftarrow s_k,\ s_g \leftarrow s_{k+\kappa}$
    Compute action loss
    $\mathcal{L}_{GCBC} = -\frac{1}{\kappa} \sum_{t=k}^{k+\kappa} \ln\big(\pi_{GCBC}(a_t|s_t, s_g)\big)$
    Update θ_{GCBC} by taking the gradient step to minimize $\mathcal{L}_{GCBC}$.

6.1 TASKS DESCRIPTIONS

Here we list the 18 tasks we use to evaluate Play-LMP, Play-GCBC, and BC at test time.

- Grasp lift: Grasp a block out of an open drawer and place it on the desk surface.
- Grasp upright: Grasp an upright block off of the surface of the desk and lift it to a desired position.
- Grasp flat: Grasp a block lying flat on the surface of the desk and lift it to a desired position.
- Open sliding: Open a sliding door from left to right.
- Close sliding: Close a sliding door from right to left.
- Drawer: Open a closed desk drawer.
- Close Drawer: Close an open desk drawer.

- Sweep object: Sweep a block from the desk into an open drawer.
- Knock object: Knock an upright object over.
- Push red button: Push a red button inside a desk shelf.
- Push green button: Push a green button inside a desk shelf.
- Push blue button: Push a blue button inside a desk shelf.
- Rotate left: Rotate a block lying flat on the table 90 degrees counter clockwise.
- Rotate right: Rotate a block lying flat on the table 90 degrees clockwise.
- Sweep left: Sweep a block lying flat on a table a specified distance to the left.
- Sweep right: Sweep a block lying flat on a table a specified distance to the right.
- Put into shelf: Place a block lying flat on a table into a shelf.
- Pull out of shelf: Retrieve a block from a shelf and put on the table.

Method	**success %**	**success with ∼0.4m perturbations**	**training** data	**collection cost**	**training shots per task**
BC	$70.3\% \pm 11.7$	23.2%	labeled	expensive	100
Play-GCBC	$77.9\% \pm 2.2$	68.3%	unlabeled	cheap	0
Play-LMP	**85.5%** ± 1.7	**78.8%**	unlabeled	cheap	0

Table 1: 18-task success.

6.2 Robustness

In Fig. 6, we see how robust each model is to variations in the environment at test time. To do so, prior to executing trained policies, we perturb the initial position of the robot end effector. We find that the performance of policies trained solely from positive demonstration degrades quickly as the norm of the perturbation increases, and in contrast, models trained on play data are more robust to the perturbation. We attribute this behavior to the well-studied "distribution drift" problem in imitation learning (Ross et al. (2011)). Intuitively, models trained on expert demonstrations are susceptible to compounding errors when the agent encounters observations outside the expert training distribution. In interpreting these results we posit 1) the lack of diversity in the expert demonstrations allowed policies to overfit to a narrow initial starting distribution and 2) a diverse play dataset, with repeated, non-stereotyped object interaction and continuous collection, has greater coverage of the space of possible state transitions. This would make it more difficult for an initial error (or perturbation) to put the agent in an observation state outside its training distribution, ameliorating the compounding drift problem.

6.3 Emergent Retrying

We find qualitative evidence that play-supervised models make multiple attempts to retry the task after initial failure. In Fig. 8 we see an example where our Play-LMP model makes 3 attempts to close a sliding door before finally achieving it. Similarly in 9, we see that the Play-LMP model, tasked with picking up an upright object, moves to successfully pick up the object it initially had knocked over. We find that this behavior does not emerge in models trained solely on expert demonstrations. We posit that the unique "coverage" and "incompletely functional" properties of play lend support to this behavior. A long, diverse play dataset covers many transitions between arbitrary points in state space. We hypothesize despite initial errors at test time lead the agent off track, it might still have (current state, goal state) support in a play dataset to allowing a replanning mechanism to succeed. Furthermore, the behavior is "incompletely functional"–an operator might be picking a block up out of a drawer, accidentally drop it, then pick it right back up. This behavior naturally contains information on how to recover from, say, a "pick and place" task. Furthermore, it would discarded from an expert demonstration dataset, but not a play dataset.

Figure 3: **Example of play data:** here we display frames sampled every second from a same sequence and ordered from left to right and top to bottom. We see the human operator engaging in self-guided interaction with a rectangular object through VR teleoperation. In this case, the operator chooses to pick up the object, push it around, uses it to push the door to the left, drops the object inside the cabinet, then finally drops the object off the table. Our play dataset consists of 3 hours of unscripted continuous play similar to this sequence. Note that subsequences could be considered task demonstrations, e.g. when the agent places the block inside the shelf. Although, they might not necessarily be expert demonstrations, but rather incompletely functional, containing misses, inefficient behavior, etc. Also note that not all the behaviors observed during play are evaluated, e.g. when the agent drops the object off the table or opens the door with the block.

Figure 4: **Example of a supervised demonstration** sequence labeled and segmented for the "sliding" task.

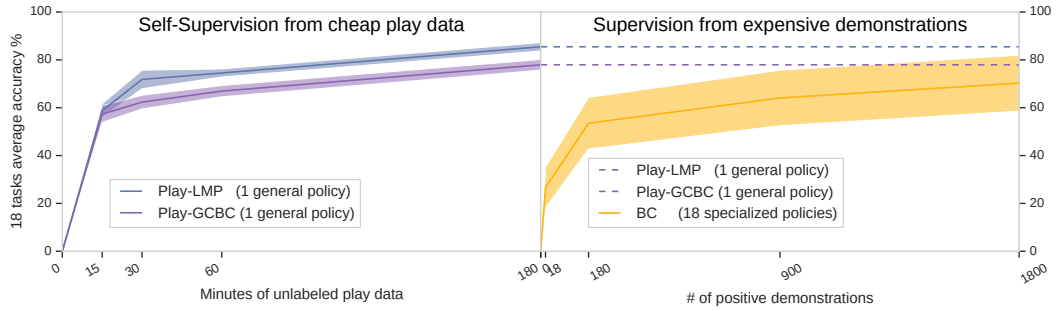

Figure 5: **18-tasks average success** when self-supervising on cheap play data (left), versus training with direct supervision from expensive positive demonstrations (right). A single task-agnostic Play-LMP policy not trained on any task-specific data outperforms 18 specialized policies, each trained on individual expert task demonstrations. The best model trained with play-supervision (LMP) reaches an average of 85.5% success in 0-shot training, while the best expert-supervised model (BC) reaches 70.3% success in 100-shot (per-task) training.

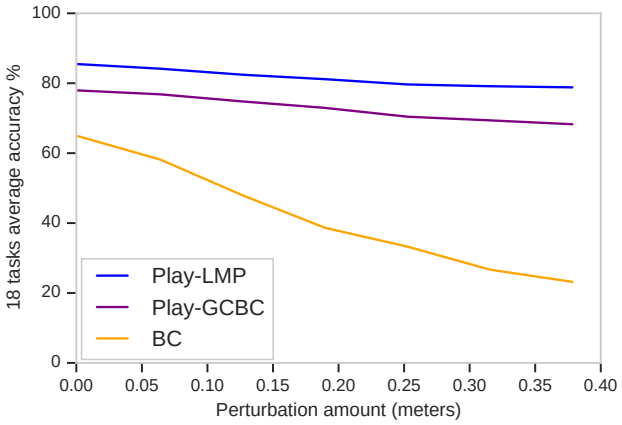

Figure 6: **Robustness to variations** in starting positions compared to the observed sequence from which the goal is extracted. With no perturbations, the successes of Play-LMP, Play-GCBC and BC are 85%, 78% and 70% respectively. However with a perturbation of $\sim$0.4 meters, successes drop to 79%, 68% and 23% respectively. The Play-LMP model is the most robust to changes to initial agent position.

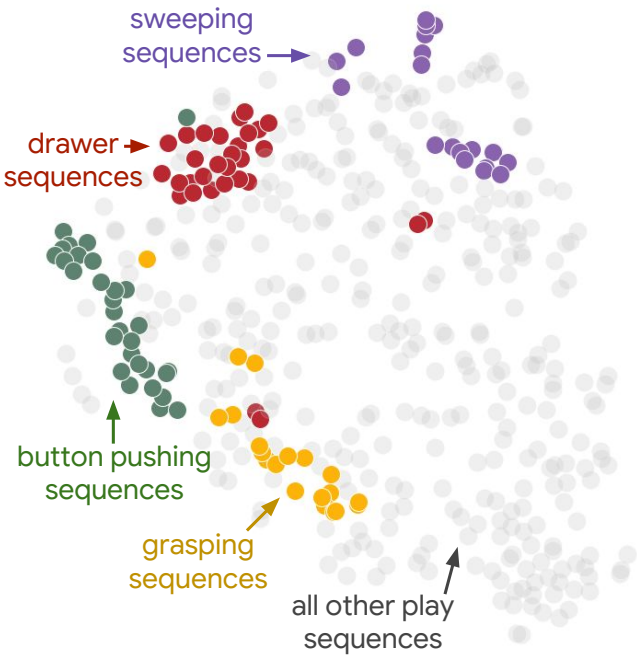

Figure 7: **Latent plan space t-SNE.** Despite never being trained with task labels, Play-LMP learns to organize a learned latent plan space with respect to tasks. We embed 512 randomly selected windows from the play dataset as well as all validation task demonstrations, using the Φ plan recognition model. Embedded positive task demonstrations are colored by task type, random embedded play sequences are colored grey.

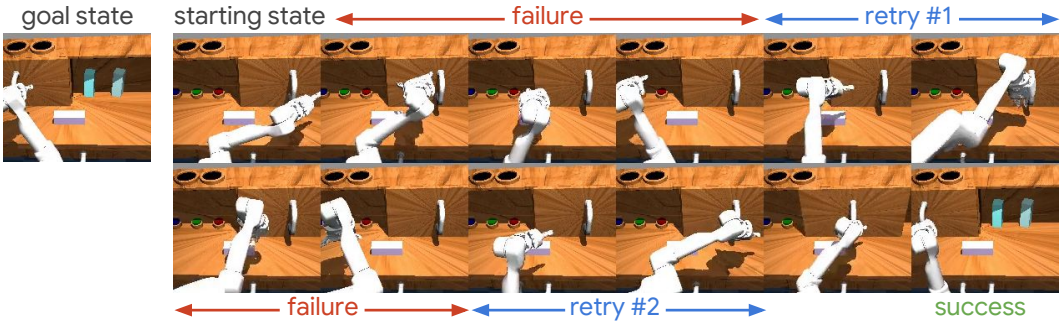

Figure 8: **Naturally emerging retrying behavior:** example run of Play-LMP policy on "close sliding" task (sliding door left to right). The policy is aiming the reach the goal state (left), fails multiple times but retries without being explicitly asked to and is successful at the 3rd attempt.

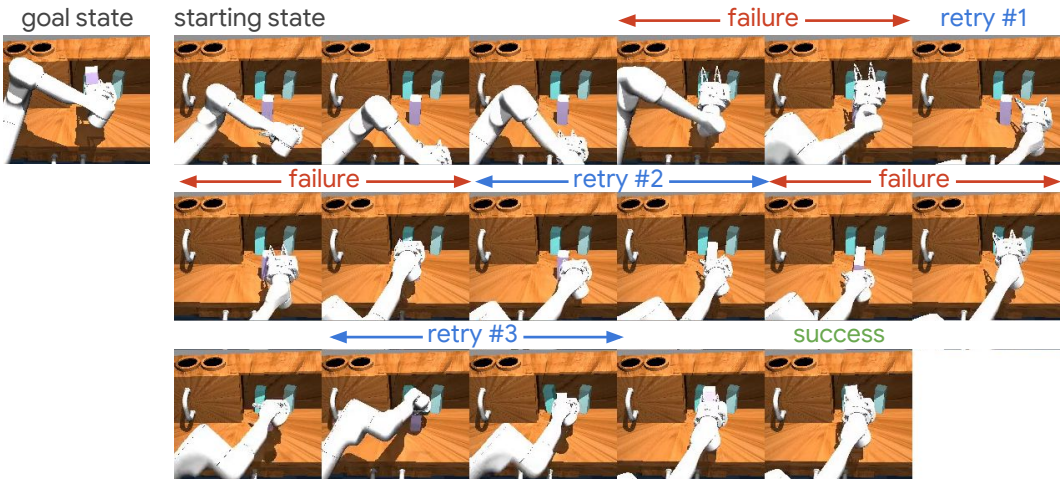

Figure 9: **Naturally emerging retrying behavior:** example run of Play-LMP policy on "grasp upright" task (grasping an object in upright position). The agent fails initially, missing the block at first then knocking it over, then recovers successfully–picking up the knocked over block.