Extended Abstract

Motivation Multi-task RL and Behavioral Cloning are shown to enable cross-task knowledge transfer via shared parameters. From co-training experiments, we know that knowledge transfer across domains is guided by Structured Representation Alignment, and Importance Reweighting Lei et al. (2026). In this project, we investigate these factors in a multitask agent environment. By observing representations over BC and RL policies, we see how importance reweighting affects task alignment in the model space. We hypothesize that highly aligned tasks require less task-specific data, as a multi-task policy can leverage shared representations to offset smaller mixing ratios without sacrificing performance.

Method This project investigates a multitask agent in a small (32 x 32) gridworld. Every state is encoded by 7 numbers: the agent grid position, the goal position, the position of an object, and the task id. At every state, the agent has 5 choices: move north, east, south, west, or pick up an object. We constructed 4 tasks:

Nav: Move towards the goal and stop on reach.
Follow: Move towards an object that relocates every 3 steps.
Pick: Traverse the grid in a snake pattern, picking up objects encountered.
Collect: Pick up all objects, then move to the goal.

By training this agent using different task-specific data mixing ratios, we could control and identify how importance reweighting affected task alignment.

Implementation To train this agent, we use Goal Conditioned Imitation Learning Ding et al. (2020). The model generates a task embedding, then takes as input the (x, y) position concatenated with the embedding. This vector passes through two ReLU linear layers, with a final linear layer outputting logits over 5 actions, trained with cross-entropy loss against expert demonstrations.
We built 4 deterministic experts per task to avoid multimodality issues, collecting 100 rollouts each. For a mixture with weights $w_1, \ldots, w_4$ (where $\sum w_i = 1$), we sample w_i of task i's data during training. Mixing ratios were swept over all 286 combinations at intervals of 0.1, ordered as navigation-follow-pick-collect.
To replicate this experiment using Goal Conditioned Reinforcement Learning, we trained the same model using the REINFORCE algorithm and controlled the data mixing ratios by changing the number of rollouts the model could collect for each task.

Results We analyzed Pareto optimality over the 286 sampled weight mixes, finding only 26 Pareto-optimal configurations. Pareto-optimal mixes exist across all collect and nearly all pick weights, but adding navigation and follow data yields diminishing returns, suggesting collect and pick contribute more task-specific transfer.
From task embeddings, we observe that the collect task is highly correlated with that of the follow task and partly the navigation task, which affirms the collect task leads to learning transfer in the pick and navigation tasks. We also noticed that increasing pick data significantly improved navigation performance despite pick's low overall success, suggesting better task discernibility reduced negative transfer.
When comparing algorithms, the mechanistic structure of the RL experiment showed that REINFORCE struggled more than BC to identify shared representations, instead learning all tasks differently.

Discussion From the results, we see a pattern of positive transfer between related tasks, and the necessity of more data to improve discernibility between unrelated tasks. When larger embedding distance between tasks result in more successful models, this indicates possible negative transfer between the representations. Similarly, when representations are shared, a task is likely to achieve non-zero success with no training data just through extrapolation.

Conclusion Observing representations across BC and RL policies reveals how importance reweighting affects task alignment and how shared structure drives optimal weight-mixing. Structurally similar tasks transfer learning and require fewer expert samples, while dissimilar tasks need more to avoid negative transfer. Future work will improve reward shaping to elicit richer RL representations, and extend findings to more complex, realistic environments.

Learning Transfer in Multitask Agents

Malti John
Department of Computer Science
Stanford University
malti@stanford.edu

Abstract

Multi-task Reinforcement Learning (RL) and Behavioral Cloning (BC) enable cross-task knowledge transfer, a process guided by structured representation alignment and importance reweighting. This project investigates how expert data mixing ratios and learned representations affect task alignment in a multi-task gridworld environment. We evaluate agents trained via Goal-Conditioned Imitation Learning across four distinct tasks (Navigation, Follow, Pick, and Collect) by sweeping 286 combinations of expert data mixing ratios. Our analysis identifies 26 Pareto-optimal configurations, revealing that tasks with shared structural elements exhibit positive transfer and require fewer task-specific demonstrations. Conversely, tasks lacking shared structure require higher data ratios to avoid negative transfer. Observing representations across BC and RL policies reveals how importance reweighting affects task discernibility and how shared structure drives optimal weight-mixing.

1 Introduction

As decision-making agents move toward real-world deployment, the demand for generalists grows. When building generalist agents, policies need to learn how to do many different tasks in an effective way. Given limited data and model complexity, it is not obvious how training data from different tasks can affect each other and the performance of the model. Simply aggregating task data does not guarantee success due to task interactions within the model. Additionally, agents may encounter out of distribution tasks, requiring the ability to transfer representations from previously learned skills.

Multitask reinforcement learning and imitation learning has been shown to enable transfer of knowledge across different but related tasks due to the shared parameters in the model. This idea has also been explored in co-training tasks with data from other tasks or data domains. Co-training can lead to knowledge transfer between the distributions. However, transfer is not always positive.

Completely aligned task representations can result in loss of discernibility between tasks, leading to negative transfer, while misaligned representations can result in no transfer. Shared alignment with discernibility is essential for policy learning transfer, but policy execution is ultimately governed by an Importance Reweighting Effect, where the agent's action distribution is dynamically modulated by data mixing ratios, dataset sizes, and domain gaps.

Using these philosophies from co-training, we can investigate task knowledge transfer on a multi-task agent by training policies on different mixing ratios of example task rollouts. We hypothesize that if a task has high alignment with another, training this multi-task policy requires less task-specific data as it learns representations from the other task's data, and will have internally aligned, yet distinguishable task representations.

Stanford CS224R 2025 Final Report

2 Related Work

Multitask reinforcement learning and imitation learning: These have been shown to enable transfer of knowledge across different but related tasks due to the shared parameters in the model, as well as generalize well to unseen, similar tasks. Zhang et al. (2023)

Representation Alignment and Importance Reweighting: Co-training enables knowledge transfer across tasks, but depends critically on learned representations. Fully aligned representations cause negative transfer by eliminating task discernibility, while misaligned representations prevent transfer entirely. Policy execution is ultimately governed by an Importance Reweighting Effect, where action distributions are modulated by data mixing ratios, dataset sizes, and domain gaps. Lei et al. (2026)

Cotraining: Strategies to harness more value from data is a question explored in cotraining Lin[1,2] et al. (2026). Other work has explored what level of similarity between the "digital cousin" in real and sim produce the best performance in cotrained policies, with task-aware and task-agnostic data Maddukuri et al. (2025). Additionally, it was found that co-training with diverse simulation data results in better generalization to novel objects and scenes.

In this project, we differentiate from the prior work by applying a mechanistic framework to a multitask agent environment. By observing representations over BC and RL policies, we see how importance reweighting affects task alignment in the model space. Furthermore, we investigate generalizability from tasks with shared representations to novel scenarios.

3 Method

Our project relied on 2 major algorithms: Goal-conditioned BC and Goal-conditioned REINFORCE. We formulate the algorithms as follows:

$$\mathcal{L}_{\text{BC}}(\theta) = -\sum_{(s,a,g)\sim\mathcal{D}} \sum_{c=1}^{\text{actions}} \frac{e^{\theta(s,g)_c}}{\sum_{i=1}^{\text{actions}} e^{\theta(s,g)_i}} \mathbb{1}\{c = a\} \tag{1}$$

$$\mathcal{L}_{\text{REINFORCE}}(\theta) = \sum_{\tau\sim\mathcal{D}} \sum_{i=1}^{\tau} \left(\log p_\theta(a_i^{(\tau)} | s_i^{(\tau)}, g^{(\tau)})\right) \left(\sum_{t=i}^{\tau} r(s_t^{(\tau)}, a_t^{(\tau)}, g^{(\tau)}) - b(s_t, g)\right) \tag{2}$$

For our main experiments, we use the BC algorithm as it has faster convergence and the data mixing was more controlled. For our BC algorithm, we used a training set of 100 expert rollouts across different tasks, and ran the model for 200 epochs. Our BC model generates a task embedding, then takes as input the observation concatenated with the embedding. This vector passes through two ReLU linear layers with a hidden dimension of 64, and with a final linear layer outputting logits over 5 actions, trained with cross-entropy loss against expert demonstrations.

To analyze our results, we used the concept of Pareto dominance, where we say that x Pareto dominates y (denoted as $x \succ y$) if and only if:

$$\forall i \in \{1,2,3,4\}, \quad \text{TaskSuccess}_i(x) \geq \text{TaskSuccess}_i(y)$$
$$\exists i \in \{1,2,3,4\}, \quad \text{TaskSuccess}_i(x) > \text{TaskSuccess}_i(y)$$

We also conducted a follow up experiment with the REINFORCE algorithm. The RL policy reused the architecture we created in the BC experiment in order to control our variables. Since vanilla policy gradient is highly unstable, we built shaped rewards into the environment (see experimental setup) and ran the experiments for 2000 epochs with 8192 trajectories sampled in every step. The REINFORCE algorithm was trained over 4 different seeds and 4 different mixing ratios , two of which Pareto-dominated the other. These ratios were chosen because they resulted in the highest level of domination across all ratios and all tasks had nonzero success.

Our baseline $b(s_t)$ for the REINFORCE algorithm was calculated as follows:

$$b(s_t, g_i) = \frac{\sum_{\tau\sim\mathcal{D}} \sum_{(s,a,g)\sim\tau} r(s,a,g) \mathbb{1}\{s[0] = s_t[0], s[1] = s_t[1], g = g_i\}}{\sum_{\tau\sim\mathcal{D}} \sum_{(s,a,g)\sim\tau} \mathbb{1}\{s[0] = s_t[0], s[1] = s_t[1], g = g_i\}}$$

To analyze representations across tasks, we also downproject the internal embeddings of our model using a UMAP projection, with hyperparameters n_neighbors = 15 and min_dist=0.1.

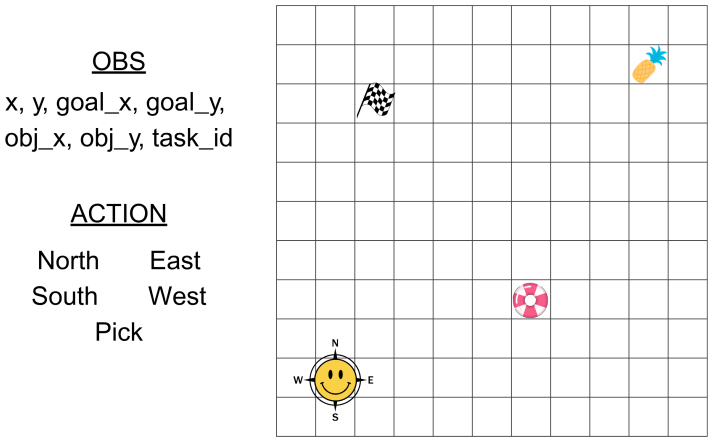

Figure 1: Gridworld

4 Experimental Setup

This project investigates a multitask agent in a small (32 x 32) gridworld. Every state is encoded by 7 numbers: the agent grid position, the goal position, the position of an object, and the task id. At every state, the agent has 5 choices: move north, east, south, west, or pick up an object.

We constructed 4 tasks:

- **Nav**: Move towards the goal and stop on reach.
- **Follow**: Move towards an object that relocates every 3 steps.
- **Pick**: Traverse the grid, picking up objects encountered, exit when reach the end.
- **Collect**: Pick up all objects, then move to the goal.

To collect our expert rollouts, we built a deterministic policy for each task in order to avoid errors from multimodality. Here are the deterministic expert policies:

- **Nav**: Move vertically, then horizontally to reach the goal.
- **Follow**: Move vertically, then horizontally to the object.
- **Pick**: Move through the grid in a snakelike pattern. If a cell has an object, pick.
- **Collect**: Make the same choices as the follow policy until an object is reached. Then pick. Once the object reads (-1, -1), mimic the Nav policy.

As the collect task is the combination of the previous three tasks, we expected that our embeddings would show a shared representation between collect and all three of them. Additionally, since follow and nav are so similar, we expected that they would have a high degree of learned similarity.

When running the BC experiment, we wanted to control the data mixing in order to understand the effects of importance reweighting on our shared representations. To do this, we sampled 100 rollouts from our collected expert demonstrations. If we have the weight ratio w_1, w_2, w_3, w_4, where $\sum_{i=1}^{4} w_i = 1$, then we will select $100 \cdot w_i$ of our rollouts from the expert dataset for task i. We generated models for each combination of data ratios, swept from $0, 0, 0, 1.0$ to $1.0, 0, 0, 0$ with intervals of 0.1 for 286 total weights. In this convention, the first task weight correlates with the navigation task, then follow, pick, and finally collect.

When running the REINFORCE algorithm, we built shaped rewards in our environment. The following rewards are listed for each task:

- **Nav**: The reward is the negative distance between the current position and the goal position, with a reward of 10 for reaching the goal.

- **Follow**: The reward is the negative distance between the current position and the object position, with a reward of 10 for reaching the object.
- **Pick**: The reward is -1 for not matching the deterministic Pick expert's move at that state. Otherwise, it is 10 if they pick up an object or if they reach the end of the grid.
- **Collect**: The reward is the negative distance between the current position and the object position, with a reward of 10 for reaching the object and then an additional reward of 10 for picking it up, and a -1 if not. Once all objects have been picked up, the reward is the negative distance between the current position and the goal position, with a reward of 10 for reaching the goal.

Since our state space is large, when conducting REINFORCE, many of our states collected during rollouts were unique, so we use a baseline that was computed as the marginalized average reward for the agent x, y position and the task id.

For all models trained and evaluated, we trained across 4 different seeds and averaged the results to reduce statistical significance of a "lucky seed."

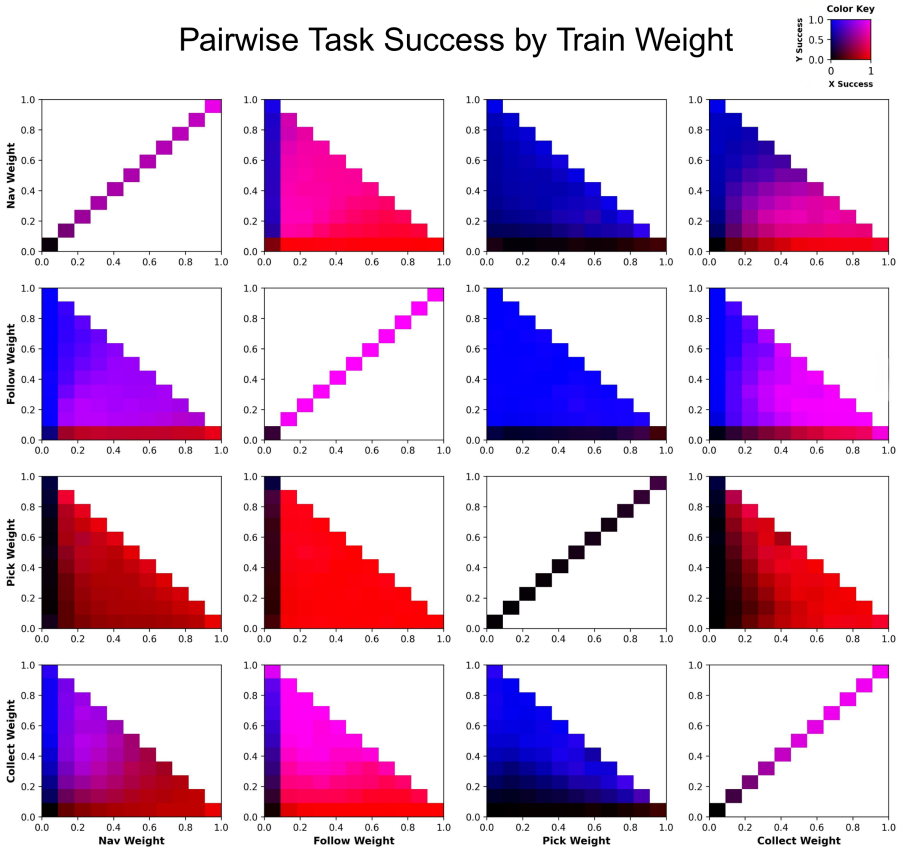

Figure 2: Pairwise task performance

5 Results

After running the outlined experiments, we analyzed three sets of information.

- Pareto-optimal weight mixture ratios

- Internal embeddings and latent structure of the tasks in the BC policies
- Task embeddings and latent structure of the RL policies

5.1 Quantitative Evaluation

From Figure 2, we can see that follow and collect tasks showed mutual improvement, with follow learning readily from collect demonstrations. Navigation and follow success also appeared correlated. Notably, increasing pick data significantly improved navigation performance despite pick's low overall success, suggesting better task discernibility reduced negative transfer.

Table 1: Pareto-optimal co-training weight mixtures (seed-averaged success rates).

ID	w_{nav}	w_{follow}	w_{pick}	w_{collect}	Nav	Follow	Pick	Collect
A	0.0	0.0	0.0	1.0	0.176	0.874	0.168	0.958
B	0.0	0.0	0.9	0.1	0.020	0.470	0.288	0.712
C	0.0	0.1	0.0	0.9	0.094	1.000	0.090	0.968
D	0.0	0.2	0.0	0.8	0.138	1.000	0.212	0.962
E	0.0	0.2	0.1	0.7	0.046	1.000	0.008	0.974
F	0.0	0.2	0.2	0.6	0.032	1.000	0.010	0.970
G	0.0	0.3	0.0	0.7	0.116	1.000	0.256	0.960
H	0.0	0.4	0.0	0.6	0.164	1.000	0.160	0.950
I	0.1	0.0	0.8	0.1	0.718	0.120	0.132	0.516
J	0.1	0.0	0.9	0.0	0.938	0.100	0.186	0.000
K	0.1	0.1	0.4	0.4	0.752	1.000	0.120	0.946
L	0.1	0.1	0.6	0.2	0.804	1.000	0.068	0.942
M	0.1	0.2	0.0	0.7	0.526	1.000	0.000	0.966
N	0.1	0.2	0.2	0.5	0.656	1.000	0.058	0.948
O	0.1	0.2	0.5	0.2	0.694	1.000	0.158	0.814
P	0.2	0.0	0.5	0.3	0.772	0.124	0.122	0.802
Q	0.2	0.0	0.6	0.2	0.690	0.146	0.180	0.492
R	0.2	0.0	0.7	0.1	0.836	0.112	0.040	0.228
S	0.2	0.1	0.2	0.5	0.798	1.000	0.012	0.950
T	0.2	0.1	0.4	0.3	0.774	1.000	0.116	0.924
U	0.2	0.1	0.5	0.2	0.820	1.000	0.072	0.794
V	0.2	0.1	0.6	0.1	0.708	0.996	0.192	0.476
W	0.2	0.2	0.2	0.4	0.776	1.000	0.040	0.956
X	0.3	0.0	0.4	0.3	0.826	0.160	0.058	0.520
Y	0.3	0.0	0.5	0.2	0.790	0.158	0.100	0.354
Z	0.3	0.0	0.6	0.1	0.822	0.152	0.078	0.170

We analyzed Pareto optimality over the 286 sampled weight mixes, averaged over 4 seeds, finding only 26 Pareto-optimal configurations. Pareto-optimal mixes exist across all collect and nearly all pick weights, but adding navigation and follow data yields diminishing returns, suggesting collect and pick contribute more task-specific transfer. A surprising result is mixture A, where pick had a higher success rate with zero training demonstrations than when it had more training samples, indicating strong extrapolation from follow and collect without obfuscation from its own data.

For the REINFORCE experiments, we ran 4 different weight ratios, two of which (0.1, 0.2, 0.5, 0.2) and (0.2, 0.1, 0.6, 0.1) respectively Pareto-dominate the others (0.1, 0.7, 0.1, 0.1) and (0.1, 0.6, 0.1, 0.2) in our BC runs. These were also the two weight mixtures that resulted in the largest pair dominations. We can see that for all weights, follow reached perfect success, collect had 0 success, and then depending on the seed, either nav was successful or pick was successful.

The results in Table 2 are very different from the BC results as before, follow was strongly correlated with collect success. Here, it seems like the weight mixture does not have a great effect on the model overall, and most of the variance is due to stochasticity. Similar to the BC learning, the follow task was easiest for the model, but the success rate of pick was very surprising. This suggests that the RL

Table 2: RL evaluation success (%) at epoch 1950, averaged over training seeds (mean ± std). Eval uses 5 rollout seeds per task.

$(w_{\text{nav}}, w_{\text{fol}}, w_{\text{pick}}, w_{\text{col}})$	Train seeds	Nav	Follow	Pick	Collect
(0.1, 0.2, 0.5, 0.2)	0–3	69.8 ± 39.1	100.0 ± 0.0	24.5 ± 42.4	0.0 ± 0.0
(0.1, 0.7, 0.1, 0.1)	0–3	76.2 ± 39.4	100.0 ± 0.0	24.5 ± 42.4	0.0 ± 0.0
(0.2, 0.1, 0.6, 0.1)	0–3	66.5 ± 33.3	100.0 ± 0.0	49.0 ± 49.0	0.0 ± 0.0
(0.1, 0.6, 0.1, 0.2)	0–2	56.0 ± 37.4	100.0 ± 0.0	33.0 ± 46.0	0.0 ± 0.0

algorithm did not construct shared representations between any of the tasks, and learned them all independently, thus only learning the tasks with the least complexity.

5.2 Qualitative Analysis

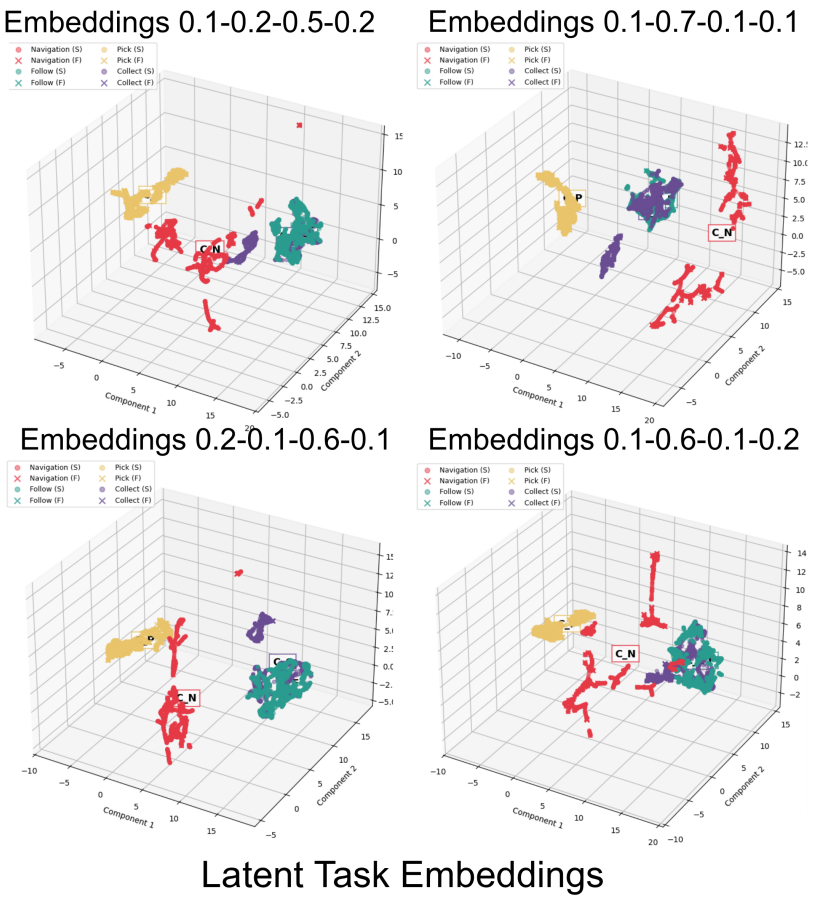

Figure 3: Task embeddings BC

The performance of the models with weight ratios on the left of Figure 3 respectively Pareto-dominate those on the right. We observe the dominating models primarily have larger embedding distance between the navigation and pick tasks than the dominated, indicating possible negative transfer between the representations. The dominating models also have tighter standard deviations for the pick task, while other task stddevs seem similar, suggesting diminishing returns in learning the other three tasks from additional data in the set.

Embeddings of the collect task are nested within that of the follow task and partly the navigation task, which affirms the collect task leads to learning transfer in the pick and navigation tasks. There is a large distance between pick embeddings and follow and collect embeddings, suggesting the model has learned very different representations between the two tasks.

Observing the embeddings of the RL algorithm, we see significantly different relationships between the structures of the data. Below, we visualized the cosine similarity of the task id embeddings in the policy. Red arrows indicate negative similarity, and blue arrows indicate positive similarity.

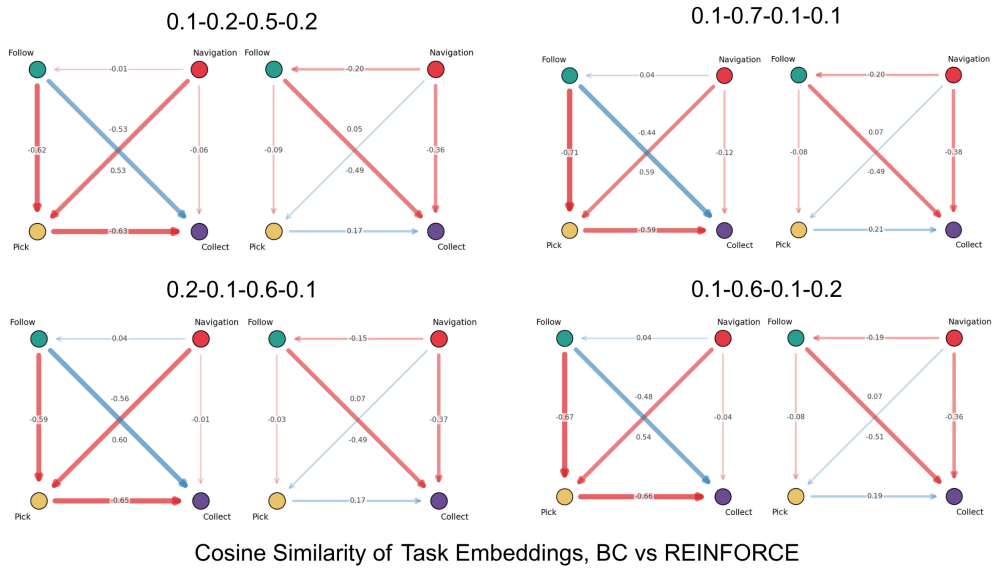

Figure 4: Cosine similarity of Task Embeddings

As we can see, while our BC model learned a strong positive correlation between the follow and collect tasks, the RL model learned a strong negative relationship. This holds across all the training weight mixtures tested as well. The only positive correlations are between navigation and pick and navigation and collect, which is surprising because collect had 0 success the entire time, and the model either chose to succeed at nav or succeed at pick - it never had nonzero success for both tasks simultaneously. This tells us that the REINFORCE model is very sensitive to its seed and rewards and isn't able to pick out shared representations in the data as strongly as a standard BC algorithm.

6 Discussion

Our results reveal a consistent pattern: tasks with shared structural elements exhibit positive transfer, while tasks with divergent structure require higher data ratios to avoid negative transfer. We now interpret these findings through the lens of representation alignment and importance reweighting, and discuss the striking divergence between BC and RL behavior.

Positive Transfer and Data Ratios

The Pareto-optimal configurations confirm our central hypothesis: tasks with high structural alignment require less task-specific data to achieve strong performance. The clearest example is the follow and collect tasks, which share navigational sub-structure and achieve mutual high success across a wide range of weight mixtures. Collect demonstrations appear to teach follow behavior, but the reverse does not seem to be necessarily true. It appears that the more rich a task is, the better it is at imparting positive learning transfer to smaller tasks.

Negative Transfer and Extrapolation

The most striking performance of learning transfer was in our Pareto-optimal mixture A, which only contained collect training data but had non-zero success on all 4 tasks. Mixture A having higher performance on pick with no training data than mixture I, which was 80% pick training data tells us that pick shared structure with collect, but had negative transfer with the navigation task. Our embeddings support the finding that pick had negative transfer with navigation, as our Pareto-dominant mixtures all had the hallmark of greater distance between the pick and navigation than the Pareto-dominated tasks. While our BC data did not support a positive correlation between pick and any of the tasks, the non-zero success of pick in mixture A can possibly be explained through successful extrapolation of the behavior of collect always picking when it reached an object. Thus, tasks that have some theoretical shared structure may not display any learned structure in the model, regardless of the data mixture, but they still might have successful extrapolation.

Our findings from the Positive and Negative transfer confirm insights from the paper inspiration for this project Lei et al. (2026). The tradeoff highlighted in the paper between collapsed representations (negative transfer from difficulty distinguishing the task) as well as positive transfer from shared representations (follow, collect, and nav sharing common threads) is confirmed through our mixing ratios.

BC vs RL Representation Learning

The BC model learned a strong similarity between the follow and collect task, and then appeared to differentiate the other tasks. This was a non-obvious correlation, as the collect task had a lot of semantic similarity with both pick and navigate. We also expected there to be more learned similarity between navigate and follow than there was. This is maybe because by switching the object every three steps, the policy had more signal about the agent's behavior in the expert grid. By changing the rate of object movement, we may be able to see a shift in the representation similarity,

The effect of Importance reweighting was clear in the BC model as we experienced very different behavior across different mixing ratios. This was significantly less pronounced in the RL experiments. Our success across weights was mostly the same, and was high variance. Additionally, while BC learned correlations and shared structure, the RL algorithm built negative correlations between most tasks. This can be explained by amount of noisy, Monte-carlo data the model collects every step. Therefore, with all the noise, the data mixing ratios aren't as controlled because much of the data collected in a rollout could provide poor signal, even if a lot of data is collected. The high-noise, high variance nature of the REINFORCE algorithm makes these balancing experiments especially noisy. Trying this experiment again with a more stable algorithm like PPO of IQL may lead to better results and uncovered structure.

Limitations and Future Work

This experiment was conducted in a relatively simple gridworld with very deterministic tasks and a fully observable state. Expanding this project to real-world multi-task agents requires handling significantly more noise and data. Based on the results from the RL experiment, it seems that structure gets lost in higher variance environments, so in future work, it would be an important problem to see how well this generalizes to noisier and larger datasets. Also, trying different RL algorithms to reduce noise would provide more insight into the learned RL structures.

7 Conclusion

Observing representations across BC and RL policies reveals how importance reweighting affects task alignment and how shared structure drives optimal weight-mixing. Structurally similar tasks transfer learning and require fewer expert samples, while dissimilar tasks need more to avoid negative transfer. Across the 286 sampled mixing ratios, it is significant that only 26 Pareto-optimal ratios emerged, suggesting the interaction between the different task datasets changed the model in a more complex way than just adding more task-relevant data.

The divergence in performance across the BC and RL algorithms also shows that the interplay between the data is strongly dependent on the algorithm used and variance experienced. Rich representations in the data can get lost when mixed with noisy feedback.

Overall, this work shows the tradeoff between leveraging shared representations and imposing task distinguishability in data, and how a balance is essential in learning robust policies. Future work

will explore different algorithms and how the ratios interact there, will look at a full sweep across the REINFORCE algorithm setup, and extend this work to a more complex, high-noise environment, such as robot training data.

Changes from Proposal This project is different from the proposal in that it uses a custom dataset and environment instead of RoboCasa. However, the proposed project and this final report share a common thread of analyzing internal alignment by controlling mixing ratios. This final report also runs an RL algorithm to add complexity to the analysis.

References

Yiming Ding, Carlos Florensa, Mariano Phielipp, and Pieter Abbeel. 2020. Goal-conditioned Imitation Learning. arXiv:1906.05838 [cs.LG] `https://arxiv.org/abs/1906.05838`

Yu Lei, Minghuan Liu, Abhiram Maddukuri, Zhenyu Jiang, and Yuke Zhu. 2026. A Mechanistic Analysis of Sim-and-Real Co-Training in Generative Robot Policies. *arXiv preprint arXiv:2604.13645* (2026).

Fanqi Lin[1,2], Kushal Arora[1], Jean Mercat[1], Haruki Nishimura[1], Paarth Shah[1], Chen Xu[1], Mengchao Zhang[1], Mark Zolotas[1], Owen Pfannenstiehl[1], Maya Angeles[1], Andrew Beaulieu[1], and Jose Barreiros[1]. 2026. A Systematic Study of Data Modalities and Strategies for Co-training Large Behavior Models for Robot Manipulation. (2026). arXiv:2602.01067 [cs.RO] `https://arxiv.org/abs/2602.01067`

Abhiram Maddukuri, Zhenyu Jiang, Lawrence Yunliang Chen, Soroush Nasiriany, Yuqi Xie, Yu Fang, Wenqi Huang, Zu Wang, Zhenjia Xu, Nikita Chernyadev, Scott Reed, Ken Goldberg, Ajay Mandlekar, Linxi Fan, and Yuke Zhu. 2025. Sim-and-Real Co-Training: A Simple Recipe for Vision-Based Robotic Manipulation. In *Proceedings of Robotics: Science and Systems (RSS)*. Los Angeles, CA, USA.

Thomas T. Zhang, Katie Kang, Bruce D. Lee, Claire Tomlin, Sergey Levine, Stephen Tu, and Nikolai Matni. 2023. Multi-Task Imitation Learning for Linear Dynamical Systems. arXiv:2212.00186 [cs.LG] `https://arxiv.org/abs/2212.00186`

A AI Tools Disclosure

AI tools were leveraged to help with coding. I wrote the environments, deterministic expert, rollouts, BC update, RL algorithm, and MLP without coding agent assistance. Then, after smoke testing the code, I used AI agents to optimize and batch the code for GPU rollouts, without changing the underlying functionality. I also used AI tools to write pyplot code for the graphs and embedding visualizations. AI agents were also used to help debug the project and suggest hyperparameters based on the algorithms performance.

B Supplementary Figures

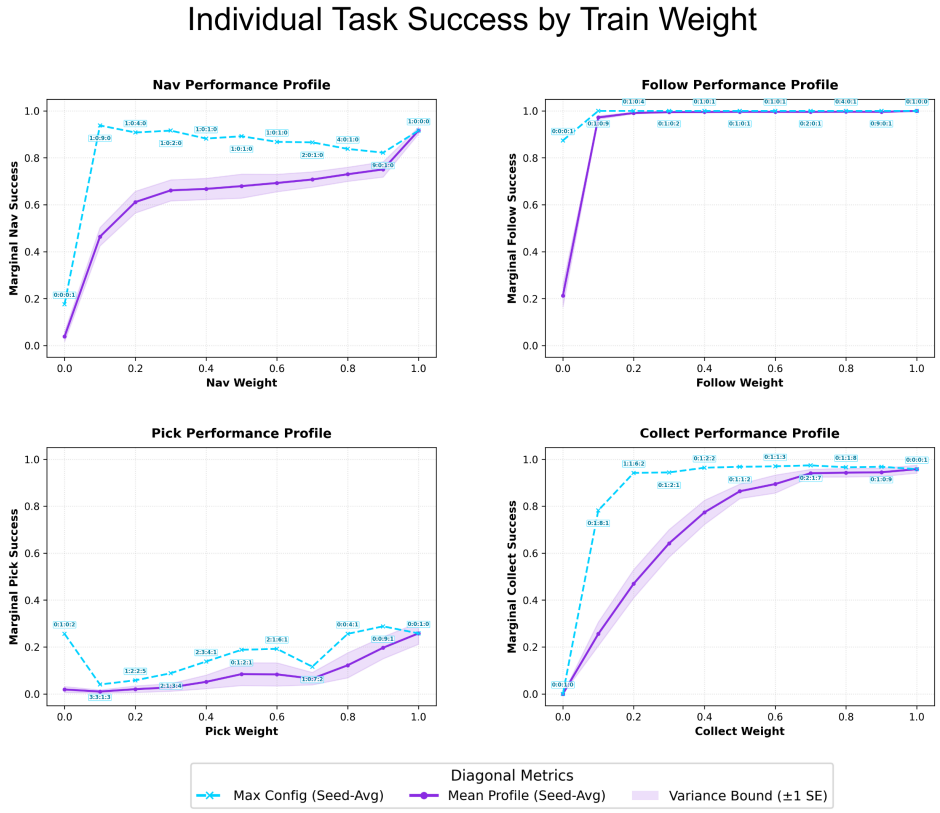

Figure 5: Marginal task performance by data ratio

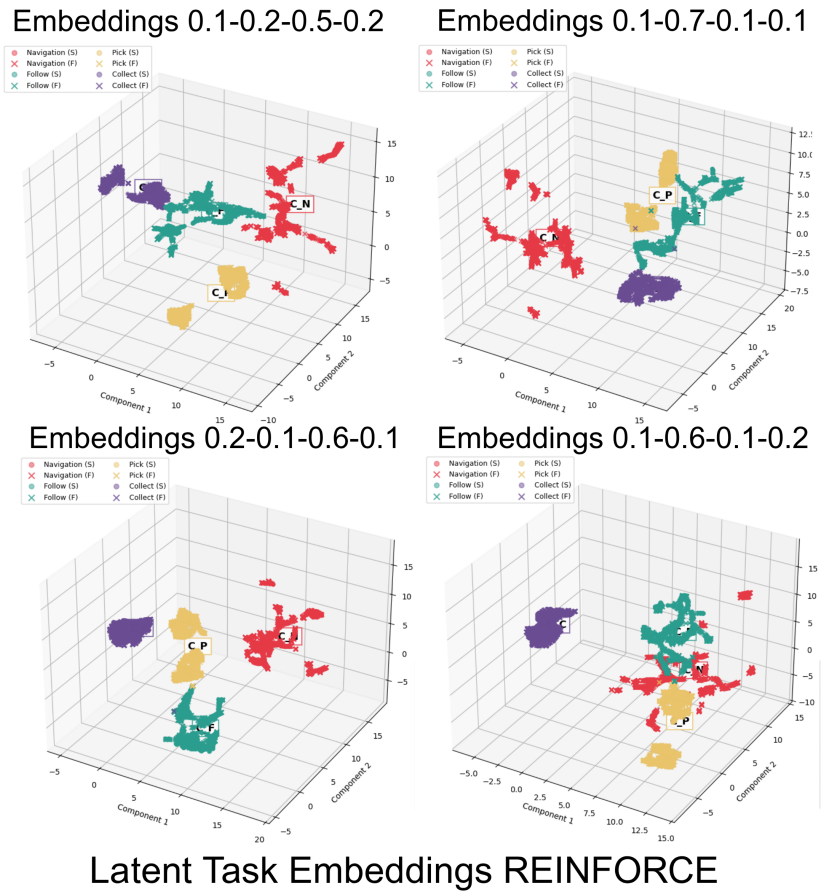

Figure 6: Task Embeddings REINFORCE