Adaptive Coordination in Social Embodied Rearrangement

Andrew Szot [1 2]   Unnat Jain [1]   Dhruv Batra [1 2]   Zsolt Kira [2]   Ruta Desai [1]   Akshara Rai [1]

Abstract

We present the task of "Social Rearrangement", consisting of cooperative everyday tasks like setting up the dinner table, tidying a house or unpacking groceries in a simulated multi-agent environment. In Social Rearrangement, two robots coordinate to complete a long-horizon task, using onboard sensing and egocentric observations, and no privileged information about the environment. We study zero-shot coordination (ZSC) in this task, where an agent collaborates with a new partner, emulating a scenario where a robot collaborates with a new human partner. Prior ZSC approaches struggle to generalize in our complex and visually rich setting, and on further analysis, we find that they fail to generate diverse coordination behaviors at training time. To counter this, we propose Behavior Diversity Play (BDP), a novel ZSC approach that encourages diversity through a discriminability objective. Our results demonstrate that BDP learns adaptive agents that can tackle visual coordination, and zero-shot generalize to new partners in unseen environments, achieving 35% higher success and 32% higher efficiency compared to baselines.

1. Introduction

Consider a human-robot or robot-robot team, collaborating at everyday tasks like unloading groceries, preparing dinner or cleaning the house. Such an assistive robot should coordinate with its partner to efficiently complete the task, without getting in their way. For example, while tidying the house, if its partner starts cleaning the kitchen, the robot could start cleaning the living room to maximize efficiency. If the robot notices its partner loading the dishwasher, it should prioritize bringing dirty dishes from the living room to the kitchen, instead of rearranging cushions. The robot should

[1]Meta AI [2]Georgia Institute of Technology. Correspondence to: Andrew Szot <aszot3@gatech.edu>.

be able to reason about its embodiment to avoid getting in the way of its partner while acting to effectively assist them. There are several challenges in building such a collaborative system. (1) The robot needs to adapt to preferences of its partner, which might be unobserved, and change over time. For example, its partner might do a different part of the task in different situations, and the robot must adapt to such changes. (2) The environment and the partner are partially observed through the robot's egocentric cameras, making both inferring the state of the partner and the environment challenging. (3) The tasks are complex and long-horizon, with feasibility constraints that affect both the robot's and its partner's actions. For example, once the robot infers that its partner is loading dishes, it must bring dishes to the kitchen to enable its partner to succeed. All of these challenges make multi-agent collaboration in visually-realistic, long-horizon tasks challenging.

Zero-shot coordination (ZSC) (Lanctot et al., 2017; Strouse et al., 2021) – a two-stage learning framework that first trains a diverse population of agents (typically enforced through random policy initializations), and then trains a coordination agent to collaborate with this population – has been used to study such problems. However, so far ZSC approaches have only been applied to simplistic environments and tasks, with complete (privileged) information, like Overcooked (Carroll et al., 2019). Instead, real-world coordination requires dealing with partial information, and high-dimensional, continuous observations like images. Such a visually-rich setting requires bulky policy architectures, and the naive strategy of random policy initializations for generating different behaviors (Strouse et al., 2021) is not enough. As a result, we observe that most agents in the population exhibit similar behaviors, like always reaching for the bowl when setting the dinner table. A coordination agent trained with such a population is not adaptive to other partner preferences, like reaching for the fruit. To solve this problem, we propose a novel approach for ZSC – Behavior Diversity Play (BDP) – which uses a shared policy architecture and a discriminability objective to encourage behavioral diversity. Specifically, we train a discriminator network to distinguish the population behaviors given a history of states, encouraging the population behaviors to be distinct. For example, when setting a dinner table, different agents in the population attempt to do different parts of the task, like reach for the
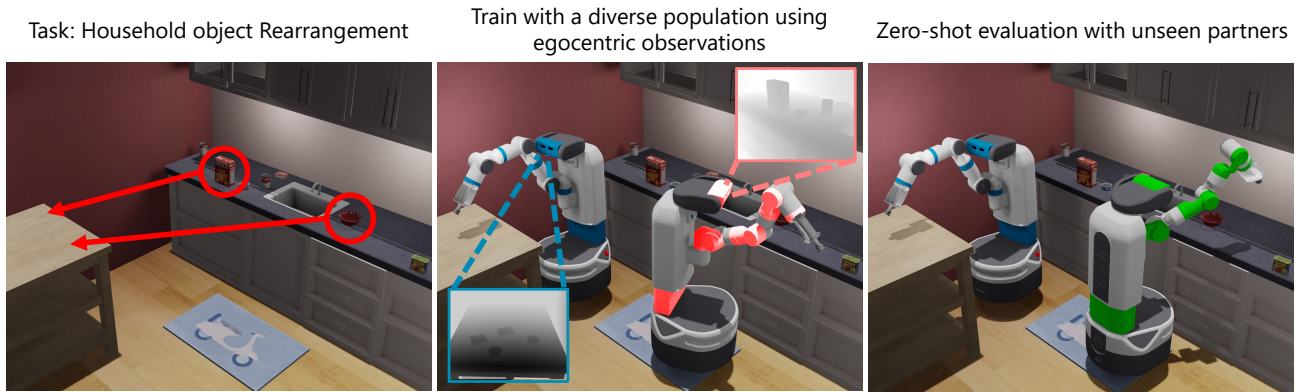
Figure 1: **Overview**. (Left) Task: Rearranging two objects from a start to a goal location. (Middle) The blue robot learns to coordinate to rearrange the objects as efficiently as possible, with diverse red partners. The robots operate from egocentric visual and proprioceptive observations. (Right) The blue robot now coordinates with an unseen green robot *zero-shot*.

fruit or the bowl. A coordination agent trained with this population is robust to either behavior at test-time, making it adaptive to unseen partners. Moreover, our shared policy architecture allows sample-efficient learning of bulky visual encoders, and parameterizes populations using a behavior latent space instead of random policy initialization. This architecture makes ZSC scalable for multi-agent collaboration in realistic, high-dimensional environments.

Our second contribution is a realistic multi-agent collaboration environment "Social Rearrangement", consisting of everyday tasks like setting up the dinner table, tidying a house or unpacking groceries. Social Rearrangement is simulated in AI Habitat (Savva et al., 2019; Szot et al., 2021) – a high-throughput physics-enabled photo-realistic 3D simulator. Two Fetch robots (Fetch, 2020) are instantiated in a fully-furnished apartment from the ReplicaCAD dataset (Szot et al., 2021), and tasked with solving everyday, long-horizon tasks (see Figure 1). The robots do not have access to any privileged information, like a bird's eye view of the house, or actions of their partner, and must operate entirely from onboard camera and proprioceptive sensing. We treat robot-robot cooperation as a proxy for human-robot cooperation and thus, don't assume access to the unobserved preferences or inner workings (policies) of the partner. The agent must coordinate with new partners, as observed through its egocentric cameras. Unseen partners have (scripted or learned) task-specific preferences, some might do only one part of the task, and others might do nothing; the coordination agent must adapt to this range of behavior.

We evaluate ZSC at Social Rearrangement, and observe that state-of-the-art ZSC approaches have poor generalization performance in this environment, due to a lack of behavior diversity in their learned population. Instead, BDP learns diverse coordination behaviors in its population, with the help of the discriminability objective which encourages agents

in the population to exhibit distinct behaviors, and in turn, can be used to train an adaptive coordination agent. Our experiments show BDP achieves 35% higher success and is 32% more efficient when coordinating with unseen agents compared to the approach from (Jaderberg et al., 2019), averaged over 3 tasks. Finally, we present approaches for analysing population behavior and diversity, and show that higher diversity results in stronger zero-shot coordination.

Overall, the key contributions of our work are: (1) We propose a novel ZSC approach – Behavior Diversity Play (BDP) which outperforms prior ZSC approaches at visually-rich tasks. (2) We present the Social Rearrangement task for collaborative embodied AI research, featuring realistic everyday home tasks like tidying up a house. (3) We benchmark ZSC approaches at complex, long-horizon tasks, against unseen learned and scripted agents over 10,000 rearrangement problems in 60 environments. All code is available at https://bit.ly/43vNgFk.

2. Related Work

Visual Embodied Agents. Embodied AI has seen great advancements in simulation platforms (Kolve et al., 2019; Chang et al., 2017; Xia et al., 2018; Savva et al., 2019; Xia et al., 2019; Weihs et al., 2020b; Xiang et al., 2020; Puig et al., 2018; Szot et al., 2021) and new task specifications (Savva et al., 2019; Anderson et al., 2018; Batra et al., 2020b; Chattopadhyay et al., 2021; Chaplot et al., 2020; Wani et al., 2020; Chen et al., 2020; Gan et al., 2021). Object rearrangement, where a robot must interact with the environment to achieve a desired environment configuration is an important task for home robotics (Batra et al., 2020a), and a variety of simulators support it (Weihs et al., 2021a; Shridhar et al., 2020; Padmakumar et al., 2021; Ehsani et al., 2021; Szot et al., 2021). We utilize the Home Assistant Benchmark (HAB) in AI Habitat proposed by (Szot et al., 2021), consisting of home tasks like "tidy the house", "set the table",

and "prepare the groceries". Our Social Rearrangement task extends the HAB to multi-agent setting.

Visual Deep Multi-Agent RL. Multi-agent RL (MARL) deals with learning policies for multiple embodied agents act to complete a task e.g., synchronized moving of furniture, that necessarily requires two agents (Jain et al., 2019; 2020). Beyond collaborative tasks, competitive tasks like hide-and-seek and soccer have been investigated (Chen et al., 2019; Juliani et al., 2018; Weihs et al., 2021b; Kurach et al., 2020). Visual MARL has been studied in the heterogeneous setting – where embodied agents have different capabilities (Thomason et al., 2020; Roman et al., 2020; Patel et al., 2021) and teacher-student framework (Weihs et al., 2020a; Jain et al., 2021). Visual MARL has also been deployed to realistic, and procedurally-generated abstract environments (Jaderberg et al., 2019; Team et al., 2021). Prior work on committed exploration for MARL (Mahajan et al., 2019) shows basic adaptation to structural changes in environment and task setup. Building on, but the above works, we learn agents that can adapt to *novel partners* at evaluation time. We make the choice to not model communication between the agents and study coordination purely based on observing the partner.

Adaptability in Multi-Agent RL. Ad-hoc teamwork (Stone et al., 2010; Barrett et al., 2011) studies how agents can adapt their behavior to join teams. Similarly, theory of mind (Premack & Woodruff, 1978) studies how modeling the behavior of partner agents improves coordination robustness (Choudhury et al., 2019; Sclar et al., 2022). Previous works (Puig et al., 2020; Carroll et al., 2019) study how to coordinate with humans, but assume privileged information about the partner in the form of a learned or planner-based explicit model of the partner. The related problem of *zero-shot coordination* (ZSC) (Hu et al., 2020) studies how agents generalize to new partners, without any fine-tuning. Overcooked (Carroll et al., 2019) is a simulated, simplified kitchen benchmark for studying ZSC with a discrete state and action space. Hanabi (Bard et al., 2020) is another common benchmark for ZSC (Hu et al., 2020; 2021; Lupu et al., 2021). In contrast to these low-dimensional environments, we study ZSC in Social Rearrangement, a complex, visually-realistic 3D environment. (Charakorn et al., 2020; McKee et al., 2022) show that multi-agent RL benefits from diversity over partners and environments. We address diversity through a novel ZSC approach and benchmark it over 10,000 different rearrangement problems in 60 environments.

Zero-shot coordination (ZSC). Some ZSC methods rely on known symmetries in the environment (Hu et al., 2020), known environment models (Hu et al., 2021), or simplified state spaces for manually defining coordination events (Wu et al., 2021). Alternatively, population-play (Jaderberg et al.,

2017) is a two-stage ZSC framework which first trains a population of agents through random pairing, and next trains an agent to coordinate with all agents in the population. Such approaches strive for diverse policy distributions at train time (Heinrich & Silver, 2016; Heinrich et al., 2015), which results in an adaptable coordination agent (sometimes called the "best response policy" in this literature). Fictitious co-play (Strouse et al., 2021) extends this by incorporating previous checkpoints in the population to represent varied skill levels. These approaches rely on random network initializations and stochastic optimization to achieve behavior diversity, which is not sufficient for diversity in our tasks. Other works introduce auxiliary diversity objectives based on action distributions (Lupu et al., 2021; Zhao et al., 2021; Rahman et al., 2022), which are also not well-suited to embodied tasks where different actions can lead to the same states. Instead, we use a discriminability-based diversity objective, conditioned on a history of states, with a new policy architecture to aid learning in visual environments. Specifically, we share policy parameters between agents of a population, to enable scalable, sample-efficient population training, and parametrize the population using a behavior latent space.

Modeling Diverse Behaviors. Akin to *quality diversity* (Pugh et al., 2016; Cully et al., 2015; Krause & Golovin, 2014), our method learns policies that are diverse and proficient at rearrangement. Prior work has explored low-dimensional latent spaces for behaviors (Derek & Isola, 2021), though not in the context of ZSC. DIAYN (Eysenbach et al., 2018) learns diverse skills using an unsupervised objective in single-agent settings. MAVEN (Mahajan et al., 2019) uses latent spaces to learn diverse exploration strategies in a multi-agent setting. (Wang et al., 2022) use a latent space to learn diverse behaviors from a multi-agent dataset. We also use a behavior latent space, and policies conditioned on this space, but using RL, and in the context of ZSC.

3. Social Rearrangement

We introduce the task of *Social Rearrangement* where two Fetch robots (Fetch, 2020) solve a long-horizon everyday task (like tidying a house) in a realistic, visual 3D environment. While both agents work together, they do not know each other's policy, similar to how assistive robots must adapt to their partner's behavior. The robots also do not have access to any privileged information (like a bird's eye view of the house or actions of the partner) and must operate entirely from an onboard egocentric camera and proprioceptive sensing. At evaluation time, learned agents coordinate with new partners in new environments with new object placements and furniture layout. This emulates realistic collaboration, where two agents complete a rearrangement task together, while implicitly inferring their

partner's state to aid or avoid getting in each other's way. This is a significantly more complex setting than previous environments used to test multi-agent collaboration, like Overcooked (Carroll et al., 2019), which assumes privileged information about the environment (top-down map) and operates in a low-dimensional, discrete state and action space.

In Social Rearrangement, the agents must move N objects from known start to end positions, both specified by 3D coordinates in each robot's start coordinate frame. We build on the Home Assistant Benchmark (Szot et al., 2021) in the AI Habitat simulator (Savva et al., 2019) that studies Rearrangement (Batra et al., 2020a). Social Rearrangement extends Rearrangement to a collaborative setting, where two agents coordinate to rearrange objects as efficiently as possible. Both Fetch robots are equipped with the same observation space: (1) a head mounted depth camera with $90°$ FoV and 256×256 pixel resolution, (2) proprioceptive state measured with arm joint angles and (3) base egomotion (providing relative distance and heading since the start of the episode, sometimes called GPS+Compass). Agents can also sense the relative distance and heading to their partner. Note that this does not reveal the partner's actions and intents, or even the partner's full state, like their arm joint angles or visual observations, but enables the agents to learn to avoid collisions, etc. Additionally, each agent receives the distance and heading to the target objects' start and desired end positions, as a way of specifying the task. If objects are in a closed receptacle, like a drawer or fridge, the robot needs to reason that it must first open the receptacle, before picking the object. Both agents move their base through linear and angular base velocity (2D, continuous) and move their arm by setting desired delta joint angle (7D, continuous). Grasping is controlled by engaging a suction gripper when in contact with an object (1D, binary).

The agents are rewarded for completing the task in as few simulation steps as possible, and if they collide, the episode ends with failure. Social Rearrangement consists of the following three tasks adapted from (Szot et al., 2021):

- **Set Table:** Move a bowl from the fridge to the table, and place a fruit from the fridge in the bowl. Both the fridge and drawer are initially closed, and they must be opened before removing the objects inside.
- **Tidy House:** Move two objects from initial locations to target locations. The objects are spawned across six open receptacles, and assigned a goal on one of the 6 receptacles, different from the starting receptacle.
- **Prepare Groceries:** Move one object from an open fridge to the counter and another from the kitchen to the fridge.

The different tasks elicit different coordination strategies. For example, in Set Table an agent might prefer to always pick the bowl or always pick the fruit. On the other hand, when tidying the house an agent might be indifferent to the object type and always tidy the closest object first. These everyday tasks study the ability of coordination agents to perform complex, long-horizon tasks with unseen partners and realistic sensing. While all tasks can be completed by a single agent, coordinating with a team would result in improved efficiency by dividing up the task. We follow the standard dataset split in the ReplicaCAD (Szot et al., 2021) scene dataset with YCB objects (Calli et al., 2015); agents are evaluated in new layouts of the house with new object placements. We train and evaluate policies for each task independently. Details about the tasks are in Appendix A.

4. Behavior Diversity Play

Collaboration in everyday tasks requires adapting to unseen partner behaviors. For example, a partner may choose to do different parts of a task in different episodes and the learned 'coordination agent' should adapt to such variations. We propose a new method called Behavior Diversity Play (BDP) which enables zero-shot coordination (ZSC) with unseen partners. Like prior ZSC methods, BDP consists of a two-stage training framework illustrated in Figure 2. It first learns to generate task-relevant diverse behaviors and then trains the coordination agent to coordinate with these diverse partners. By coordinating with diverse partners at training time, the coordination agent can generalize to unseen partners at test time. Specifically, in the first stage, we train a behavior policy generator π^b capable of generating diverse behaviors (Fig. 2, left). Next, the coordination agent π^c is trained to coordinate with the diverse behaviors generated by π^b (Fig. 2, middle), and finally evaluated against unseen holdout policies Π^h (Fig. 2, right). We first introduce our notation and a formal description of ZSC. Next, we detail the two stage training process and finally provide practical details on implementing BDP.

BDP learns to generate diverse behaviors through a single behavior generator π^b by conditioning on different behavior latents, as opposed to prior works that learn independent policies. This choice is conducive to our visually-rich setting where policy architectures can be large, slow to train, and difficult to fit multiple on GPUs. π^b shares weights across different population agents, increasing training efficiency. Moreover, unlike prior work (Strouse et al., 2021), BDP does not rely on random initialization of policies to ensure diverse behaviors, and instead incorporates an explicit diversity objective through a discriminator. This allows BDP to train adaptive coordination agents across multiple tasks and experimental scenarios.

4.1. Background and Notation

The goal of ZSC is to produce a *coordination agent* π^c that can coordinate with unseen partners. For a given task, the coordination agent is evaluated based on zero-shot perfor-
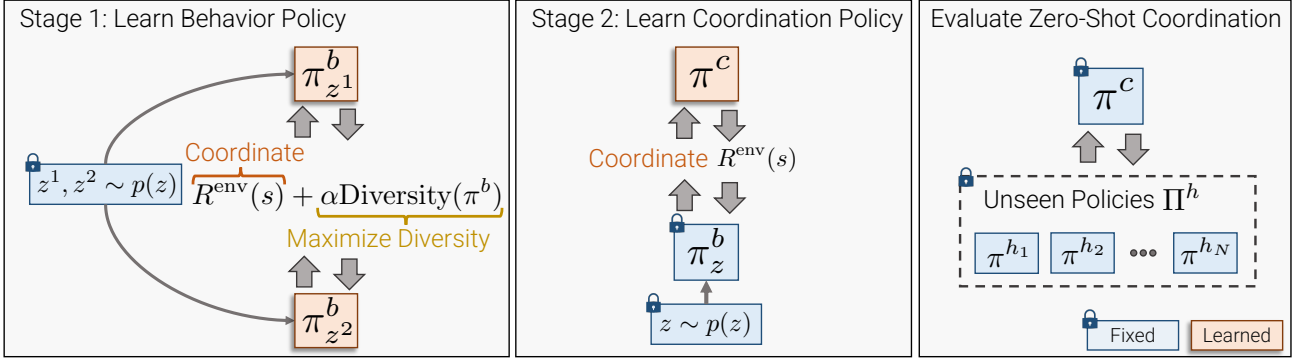
Figure 2: **Behavior Diversity Play**. (Left) Stage 1: We train the behavior policy π^b, which models a diverse set of agent behaviors, conditioned on the behavior latent z. A discriminator q_ϕ then encourages distinguishability between different z. (Middle) Stage 2: A coordination agent π^c learns to coordinate with different behaviors generated by the behavior policy. (Right) Stage 3: We evaluate the coordination agent at ZSC with unseen holdout agents π^h.

mance when paired with agents from an unseen, *holdout policy set*, Π^h. Crucially, the policies in Π^h are never seen during training. n agents solve the Social Rearrangement task, which we formulate using a decentralized partially-observed markov decision process (Dec-POMDP) consisting of the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{R}, \mathcal{P}, \gamma, n, T \rangle$. In this work we use $n = 2$ agents, but our approach remains unchanged for $n > 2$. At each time step t, the global environment state is denoted as $s_t \in \mathcal{S}$. Each agent i, receives an observation $o_t^i \in \mathcal{O}$, takes an action $a_t^i \in \mathcal{A}$, forming a joint action $\boldsymbol{a_t} \in \mathcal{A}^n$, resulting in the next environment state s_{t+1}, following the transition function $\mathcal{P}$. This gets joint reward $r_t \in \mathbb{R}$ according to the deterministic reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A}^n \to \mathbb{R}$ for an episode of length T. Agent i's policy π^i maps its observations to a distribution over actions. To handle partial observability and history, policies are modeled with recurrent neural networks. The combined expected return of policies π^1, π^2, is $J(\pi^1, \pi^2) = \sum_{t=0}^{T} \mathbb{E}_{\boldsymbol{a_t} \sim (\pi^1, \pi^2)} [\gamma^t \mathcal{R}(s_t, \boldsymbol{a_t})]$. The goal of the coordination agent π^c is to maximize the average performance over the holdout population Π^h, $\mathbb{E}_{\pi^h \sim \Pi^h} [J(\pi^c, \pi^h)]$.

4.2. Stage 1: Behavior Policy Generator

We model diverse behaviors through a *Behavior Policy Generator* policy $\pi^b(a_t | o_t, z)$, which conditioned on a *behavior latent* $z \sim p(z)$ generates distinct behaviors per z. The behavior prior $p(z)$ is modelled as a uniform categorical distribution with K categories, where K is a hyperparameter, analogous to a population size. We train π^b to maximize a joint task and diversity objective that encourages agent trajectories to be distinct, while completing the task at hand. Specifically, our training objective consists of two components: (1) task performance, aimed to learn agents that can solve the rearrangement task, and (2) diversity – a discriminator-based reward to encourage distinct behaviors per behavior latent. At the start of each episode, we sample two latents $z^1, z^2 \sim p(z)$, one for each partner. For brevity,

we denote $\pi^b(\cdot | \cdot, z)$ as π_z^b. Next, we optimize:

$$\max_{\pi^b} J(\pi_{z^1}^b, \pi_{z^2}^b) + \alpha \, \text{Diversity} \left(\pi^b \right) \qquad (1)$$

where J is the return described in Sec. 4.1 and Appendix A. Diversity is a measure of how diverse of behaviors π^b can produce, while α is weights the diversity objective.

By increasing diversity, we reduce the conditional entropy of the latent z given the state history, while also increasing the entropy of the policy. Minimizing the entropy of z given the state history encourages z to be predictable from the agent's behaviors, making the different behaviors generated by π_z^b distinct. Maximizing the entropy of the policy ensures that different z are diverse enough to cover the space of possible behaviors. To optimize this objective, BDP learns a *trajectory discriminator* q_ϕ that predicts which behavior latent corresponds to an agent's trajectory. Since the discriminator is only used during training, it enjoys access to each agent's privileged state trajectories $\tau^i = s_0^i, \ldots s_T^i$, which are not available during evaluation. Ideally, $q_\phi(z | \tau^i)$, should allocate high probability to behavior latent for agent i, *i.e.*, $z^i \in \{1 \ldots K\}$. We adapt the skill diversity formulation from (Eysenbach et al., 2018) for coordination by conditioning on state trajectories, instead of states:

$$\begin{aligned} \text{Diversity}(\pi^b) &= -\mathcal{H}\left(z | \tau\right) + \mathcal{H}(a | o, z) \\ &\geq \mathbb{E}_{z \sim p(z), \tau \sim \pi_z^b} \left[\log q_\phi(z | \tau) \right] + \mathcal{H}(a | o, z) \qquad (2) \end{aligned}$$

where $\mathcal{H}(\cdot)$ is the entropy function and the second line gives the variational lower bound on the diversity objective. $\log q_\phi(z | \tau)$ is a discriminator loss that enforces distinguishability, and $\mathcal{H}(a | o, z)$ is an entropy objective that encourages the policies to cover a large space of behaviors.

Trajedi (Lupu et al., 2021) proposes an approach that encourages distinct action distributions induced by different policies to achieve population diversity. Instead, we propose to measure diversity in terms of induced state trajectories,

which are more indicative of behaviors than actions. For example, different actions can lead to the same state changes, and hence result in the same high-level behaviors. See Appendix B for a detailed description of the difference in the diversity objectives of BDP and Trajedi.

4.3. Stage 2: Learning the Coordination Policy

After stage 1, the behavior policy generator π^b can generate diverse behaviors when conditioned on different latents z. Next, we train a new policy coordination agent $\pi^c(a|o)$ to coordinate with *all* behaviors produced by π^b from the first stage (middle panel of Fig. 2), while keeping π^b fixed. π^c is initialized randomly, and then trained to maximize task performance $J(\pi^c, \pi_z^b)$ when paired with π_z^b for a randomly-sampled z. It is hard for π^c to learn against a rapidly changing z as the behaviors it is paired with change constantly. For this reason, we sample a new z only once every several updates of π^c. Exact training details for both stages are in Appendix C.

Once trained, π^c is able to adapt to new partners zero-shot since it was trained to coordinate with the diverse behaviors from π^b (right panel of Fig. 2). Next, it is evaluated against unseen partners from the holdout set Π^h to measure its generalization performance.

While we describe the above sections assuming $n = 2$ agents collaborating, the approach remains unchanged for $n > 2$ agents. For $n > 2$, during Stage 1 training, BDP would sample n agents instead of 2 and use the same diversity objective (Eq 2). In Stage 2 training, the coordination agent would learn to collaborate with $n - 1$ partners generated by the behavior policy.

4.4. Implementation Details

We use a two-layer hierarchical policy architecture for all baselines, where a high-level policy selects a low-level skill to execute based on observations. This has shown to be effective in rearrangement tasks (Gu et al., 2022). We consider a known, fixed library of low-level skills, which can achieve instructions like 'navigate to the fridge', or 'pick an apple'. These low-level skills directly interact with the environment via low-level base and arm actions. The action space of the learned high-level policy is a discrete selection over all possible combinations of skill and object/receptacle parameterizations allowed at all steps. For navigation, we allow all possible furniture pieces to be navigable to, and manipulation skills can interact with any articulated receptacles, target objects, and goals. We also include additional high-level navigation actions, like move-forward, turn-left, turn-right, and no-op in the action space that facilitate coordination between agents, like move out of the way, if it sees its partner coming towards itself to avoid collision.

If the policy chooses to execute an infeasible action, like

pick an object when its not within reach (based on hand-defined pre-conditions), the action results in a no-op and no change to the environment. Since the focus of our work is on high-level coordination, we assume access to perfect low-level skills for all approaches. For manipulation skills, this means kinematically applying the hand-defined post-conditions of the skill, like attaching target objects in the scene to the gripper after executing the pick skill. For the navigation skill, we use a shortest path navigation module which moves the robot from its current position to the desired position (such as a receptacle or object), but does not take the partner agent into account. Avoiding collisions and coordination with the partner are dealt with by the high-level policy. More details on the hierarchical policy, policy architecture, discriminator, and pseudocode are in Appendix C. While we use a hierarchical policy architecture, BDP itself is agnostic to the policy type. Prior works like (Szot et al., 2021) have shown that such a policy architecture is well-suited to learning long-horizon rearrangement tasks, and hence the architecture used in our work. Additionally, we make some simplifications to our simulation environment by using a partially simulated physics engine that considers collisions with objects like tables and the partner agent, but ignores collisions with the fridge door. On the other hand, aspects that are important for coordination, like colliding with the partner, are fully simulated. More details on the simulation can be found in Supp. A.

5. Experiments

In this section, we compare our approach (BDP) to state-of-the-art (ZSC) methods. First, we introduce baselines, metrics, and quantitative results, then analyze the learned populations from the different approaches with a focus on measuring diversity. Lastly, we run ablations on BDP to quantify the contribution of the policy architecture and the discriminator loss in the overall performance of BDP.

5.1. Baselines

We compare Behavior Diversity Play (BDP) to a range of zero-shot coordination (ZSC) methods.

- **Self-Play (SP)** (Heinrich & Silver, 2016) learns a population of size N by randomly initializing N policies and training them via self-play.
- **Population-Based Training (PBT)** (Jaderberg et al., 2019) initializes N random policies, and pairs them randomly during training. Both PBT and SP generate diversity through random policy initializations.
- **Fictitious Co-Play (FCP)** (Strouse et al., 2021) uses SP but adds earlier checkpoints from each policy to the population when training the coordination agent.
- **Trajectory Diversity (TrajeDi)** (Lupu et al., 2021) adds a diversity objective to population training that encourages

	PBT-State *(Oracle, No Vision)*	GT Coord *(Oracle, No ZSC)*	SP (Heinrich & Silver, 2016)	PBT (Jaderberg et al., 2019)	FCP (Strouse et al., 2021)	TrajeDi (Lupu et al., 2021)	BDP *(Ours)*
Set Table							
Train-Pop Eval	70.74 ± 0.05	90.52 ± 0.05	57.74 ± 0.01	46.67 ± 0.02	29.90 ± 0.07	43.24 ± 0.09	74.81 ± 0.01
ZSC Eval	50.39 ± 0.09	-	17.97 ± 0.04	30.34 ± 0.04	37.50 ± 0.04	32.52 ± 0.04	**46.43 ± 0.08**
Tidy House							
Train-Pop Eval	74.90 ± 21.59	92.28 ± 1.66	34.18 ± 6.05	36.13 ± 0.98	12.04 ± 2.28	39.65 ± 0.59	73.83 ± 7.03
ZSC Eval	68.08 ± 0.09	-	52.67 ± 0.06	56.88 ± 0.07	34.07 ± 0.09	63.58 ± 0.05	**66.71 ± 0.05**
Prepare Groceries							
Train-Pop Eval	85.74 ± 2.82	93.63 ± 0.28	47.07 ± 29.88	69.34 ± 1.76	44.40 ± 6.38	34.56 ± 27.94	89.67 ± 2.51
ZSC Eval	77.01 ± 0.05	-	56.04 ± 0.07	56.08 ± 0.09	30.00 ± 0.07	53.84 ± 0.08	**75.85 ± 0.05**

Table 1: Evaluation of Social Rearrangement with training population and ZSC with unseen agents. BDP outperforms prior ZSC works and closes the gap to oracle methods (columns in gray). Average and standard error across 3 seeds.

diverse action distributions as opposed to the diverse state distributions encouraged by BDP.

Implementation details are in Appendix D. All of the above and BDP follow two-stage training and only differ in how they obtain a population (the second stage of training the coordination agent is identical between them). The policy and environment setup for baselines is identical to the setup for BDP described in Section 4.4. For BDP, we model the behavior latent prior $p(z)$ as a fixed 8-dimension uniform categorical distribution. Likewise, all baselines train a population of 8 agents . Additionally, we implement two 'oracle' baselines that have privileged information to disentangle the two challenges of Social Rearrangement– high-dimensional visual observations and zero-shot coordination.

- **PBT-State** (Oracle, No-Vision) To highlight the challenges of operating from visual input, we implement PBT with ground-truth environment state (PBT-State), that captures the complete environment state (details of ground-truth state in Appendix D.3). PBT-State operates in a fully observable and low-dimensional environment, similar to prior work like Overcooked (Carroll et al., 2019).
- **GT Coord** (Oracle, No-ZSC) To highlight the challenges of zero-shot coordination, we train two visual policies together (GT Coord). GT Coord operates on high-dimensional visual observations, but is trained together with its partner agent, and hence has no ZSC challenges.

5.2. ZSC Evaluation

As introduced in Sec. 4.1, to evaluate the coordination agent π^c trained using the different approaches, we task them to coordinate with a set of holdout agents Π^h unseen during training. For each task, the holdout set consists of three scripted and eight learned holdout agents, described below. Further details of the holdout agents in Appendix D.1.

Scripted holdout agents execute a fixed sequence of hard-coded task plans, for *e.g.*, "navigate to the table, pick up the object, navigate to the counter and drop the object", exhibiting different behavioral preferences. Note that the scripted holdout agents are not reactive, *i.e.*, they will not update their plan based on the coordination agent's actions, or even

try to avoid bumping into the coordination agent. This is out-of-distribution for the coordination agent, since it is trained with partners that react to its actions, making coordinating with scripted holdout agents especially challenging.

Learned holdout agents are separately trained using GT Coord and unseen during training. The coordination agent needs to adapt to these unseen policies, which were trained to expect a particular behavior from their partner.

Metrics. For the three tasks – Set Table, Tidy House, Prepare Groceries – we compare the methods using the portion of successful task completions (1) when paired with agents from the training population (*train population eval*, or *train-pop eval* for short), (2) and ZSC with unseen agents from the holdout set (*ZSC eval*). We report mean and standard deviation across 3 randomly seeded runs calculated on 100 episodes in unseen scene configurations.

5.3. ZSC Quantitative Analysis

Evaluation metrics. BDP outperforms prior ZSC baselines (SP, PBT, FCP, TrajeDi) across all Social Rearrangement tasks when comparing both train-pop eval and ZSC eval success rate (Tab. 1). For Set Table, BDP improves train-pop eval success rate by 17% (from SP's 57.74% → BDP's 74.81%). Looking closely, we find that SP is unable to coordinate with holdout policies,achieving a low ZSC eval success rate of 17.97% (while BDP can reach 46.63%). Similar boosts in ZSC eval success rate of 3.1% (63.58% → 66.71%) and 19.8% (56.08% → 75.85%) are observed in Tidy House and Prepare Groceries tasks, respectively

BDP bridges gap to oracle methods. Despite the use of privileged information and assumptions by oracle methods, BDP achieves comparable ZSC performance as them. In Table 1, we see that BDP can reason about partner and environment state from it's visual observations, and performs close to PBT-State, BDP's 46.43% vs. PBT-State's 50.39% for ZSC eval success rate in Set Table task. Unsurprisingly, there is indeed a scope for improvement in the overall coordination, since BDP's performance is still lower than oracle GT Coord (BDP's 74.81% vs. GT Coord's 90.52%). Appendix E.1 contains further analysis of ZSC performance by

breaking down ZSC by holdout agent type.

BDP coordinates efficiently. An interesting observation for some baselines, like FCP at Set Table, is that their ZSC eval success is higher than train-pop eval (37.50% vs. 29.90%). This is because some of the ZSC eval agents are more adept at solving the tasks than the ones learned in the training population. As a result, it is important to look not just at the task success, but also the *cooperation efficiency gain* of solving the task as a pair (Tab. 2).

	PBT-State	SP	PBT	FCP	TrajeDi	BDP (*Ours*)
Set Table						
Train-Pop Eval	+84%	+49%	+34%	−28%	+36%	+51%
ZSC Eval	+58%	−30%	−13%	−7%	−1%	+19%
Tidy House						
Train-Pop Eval	+85%	+6%	+10%	−28%	+21%	+57%
ZSC Eval	+57%	+16%	+24%	−11%	+30%	+32%
Prepare Groceries						
Train-Pop Eval	+77%	+26%	+45%	+18%	+12%	+71%
ZSC Eval	+52%	+22%	+21%	−6%	+17%	+44%

Table 2: **Cooperation Efficiency Gain**: BDP improves efficiency for both the training population and in ZSC compared to all baselines in all tasks. The oracle PBT-State method takes privileged state information as input.

To calculate the cooperation efficiency gain of ZSC methods, we compute the average number of steps it takes a single agent to solve a task and compare it to the average number of steps taken by 2 agent teams, where both agents can act every single step. In Table 2, we observe that ZSC baselines like PBT can actually *make the partner slower*, even if the task completion rate is high. On the other hand, BDP improves the efficiency of unseen partners (-13% using PBT versus +19% using BDP on the Set Table task). Again, as compared to the oracle PBT-State, we observe that BDP has lower ZSC and train-pop efficiency, owing to not having complete state information, and pointing towards scope for improvement. This points towards scope for improvement, but also an interesting future direction where cooperation efficiency might be studied (and improved) over repeated interactions with the same partner.

5.4. Qualitative Diversity Analysis

Characterizing the behaviors of a population is challenging, and agent behavior itself consists of long trajectories and diverse interactions, making existing diversity metrics not as meaningful (McKee et al., 2022). We present a qualitative diversity analysis approach, by pairing the *same* agent with a population of agents. Specifically, we study the behavior of the learned coordination agent when paired with agents from the training population, and holdout set. Ideally, the coordination agent exhibits diverse behaviors during training,

Figure 3: Behavior of the coordination agent in the Tidy House task with *unseen partners* during ZSC (top) and training population partners (bottom). Columns correspond to different partners for the coordination agent, rows are different sub-goals, and the cells display the probability of the sub-goal being completed by the coordination agent.

and adapts its behavior to unseen test partners.

To do this, we define sub-goals that occur in the successful completion of a task, for example, to successfully complete Tidy House, agents must rearrange 2 objects. We record the portion of these sub-goals completed by the coordination agent. If the coordination agent is biased towards only doing some parts of the task, it will fail to coordinate with partners who prefer the same portions of the task. Note that this hand-designed task decomposition is only used during evaluation, and not imposed during training, and the observed behaviors emerge solely through our diversity objective. See Appendix D.4 for more details.

BDP adapts to unseen partners. Figure 3 (top) shows the probability of coordination agent completing different sub-goals (rows), when paired with unseen holdout partners (columns). The coordination agent trained with BDP (Fig. 3, top left) performs different portions of the task, depending on its partner. In contrast, coordination agent trained with PBT (Fig. 3, top right) is biased towards a set of sub-goals, and hence can't coordinate with partners with the same bias.

BDP results in diverse population. Next, we study the training population trained in stage 1 by both BDP and PBT. Ideally, the coordination agent should exhibit diverse behaviors when paired with the training population, indicating that the population agents themselves have diverse behaviors and the coordination agent learns to adapt to them. In the bottom left of Figure 3, we see that when trained with BDP the coordination agent is unbiased, and almost equally likely to perform any sub-goal, making it highly adaptable. In contrast, using PBT, the coordination agent only typically picks the first object (Fig. 3, bottom right), implying that the training population from PBT is biased towards picking the second object, and hence the coordination agent does not see diverse behavior during training.

5.5. Ablation Experiments

We ablate the new diversity objective in BDP and show that removing the discriminator objective adversely impacts performance. Specifically, we implement the following variants of BDP: **BDP - [Discrim]** keeps the shared network parameters and latent space, but we remove the discriminator objective. Diversity only comes from different input z. **BDP - [Discrim, Latent]** uses a shared visual encoder, but removes the latent space and diversity objective, instead using random initializations for diversity. Finally, **PBT** has neither a shared latent, discriminator objective, or shared encoder. The results in Table 3 show BDP - [Discrim] suffers in ZSC eval (even though train-pop eval performance is high) since without the discriminator, there is nothing to enforce different latents to have different behaviors, making ZSC eval poor. Both BDP - [Discrim, Latent] and PBT rely on different network parameters to achieve behavior diversity, insufficient for our task, reinforcing the importance of a shared latent and discriminator objective. Appendix E.2 contains policy architecture ablations.

	BDP - [Discrim]	BDP - [Discrim, Latent]	PBT	BDP
Train-Pop Eval	77.18 ± 0.00	42.16 ± 0.01	46.67 ± 0.02	74.81 ± 0.01
ZSC Eval	22.92 ± 0.05	33.20 ± 0.04	30.34 ± 0.04	46.43 ± 0.08

Table 3: Ablations on the new diversity objective in BDP.

6. Conclusion

We present the Social Rearrangement task, consisting of collaborative, everyday tasks like tidying a house, setting a dinner table and preparing groceries. Social Rearrangement is simulated using realistic, high-dimensional observations, with no privileged information like top-down maps, or partner actions. We present a novel approach Behavior Diversity Play (BDP) for zero-shot coordination (ZSC) and evaluate it on Social Rearrangement. BDP trains an adaptable coordination agent that can collaborate with a set of unseen holdout policies, and improves the efficiency of its partner over solving the task alone. Through analysis and ablations, we show that this improvement comes from a diverse training population obtained via BDP's discrimibility objective.

While BDP is able to learn adaptive agents that can use partial information about the environment and their partners to coordinate, its performance is worse than oracles with privileged state and partner information. This implies that there is still scope for improvement for BDP at ZSC tasks. Furthermore, Social Rearrangement deals with a limited set of rearrangement tasks, with some simplifying assumptions like clean visual inputs and simplified physics.

In the future, we hope to include more complex coordination tasks like furniture assembly or cooking a meal, which might even require additional inputs like language. Future work can also improve BDP by exploring how theory of mind (ToM) can improve coordination by having the coordination policy predict the behavior policy's internal state or future actions. Such a ToM objective can help BDP learn representations that predict the other agent's intentions, improving generalization. We hope that Social Rearrangement can serve as a realistic test-bed for multi-agent collaboration research, and the approaches and analysis presented in our work can enable future research on ZSC.

7. Acknowledgments

References

Anderson, P., Wu, Q., Teney, D., Bruce, J., Johnson, M., Sünderhauf, N., Reid, I., Gould, S., and van den Hengel, A. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *CVPR*, 2018. 2

Bard, N., Foerster, J. N., Chandar, S., Burch, N., Lanctot, M., Song, H. F., Parisotto, E., Dumoulin, V., Moitra, S., Hughes, E., et al. The hanabi challenge: A new frontier for ai research. *Artificial Intelligence*, 280:103216, 2020. 3

Barrett, S., Stone, P., and Kraus, S. Empirical evaluation of ad hoc teamwork in the pursuit domain. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pp. 567–574, 2011. 3

Batra, D., Chang, A. X., Chernova, S., Davison, A. J., Deng, J., Koltun, V., Levine, S., Malik, J., Mordatch, I., Mottaghi, R., et al. Rearrangement: A challenge for embodied ai. *arXiv preprint arXiv:2011.01975*, 2020a. 2, 4

Batra, D., Gokaslan, A., Kembhavi, A., Maksymets, O., Mottaghi, R., Savva, M., Toshev, A., and Wijmans, E. Objectnav revisited: On evaluation of embodied agents navigating to objects. *arXiv preprint arXiv:2006.13171*, 2020b. 2

Calli, B., Singh, A., Walsman, A., Srinivasa, S., Abbeel, P., and Dollar, A. M. The ycb object and model set: Towards common benchmarks for manipulation research. In *2015 international conference on advanced robotics (ICAR)*, pp. 510–517. IEEE, 2015. 4

Carroll, M., Shah, R., Ho, M. K., Griffiths, T., Seshia, S., Abbeel, P., and Dragan, A. On the utility of learning about humans for human-ai coordination. *Advances in neural information processing systems*, 32, 2019. 1, 3, 4, 7

Chang, A., Dai, A., Funkhouser, T., Halber, M., Niessner, M., Savva, M., Song, S., Zeng, A., and Zhang, Y. Matterport3d: Learning from rgb-d data in indoor environments. *3DV*, 2017. 2

Chaplot, D. S., Salakhutdinov, R., Gupta, A., and Gupta, S. Neural topological slam for visual navigation. *CVPR*, 2020. 2

Charakorn, R., Manoonpong, P., and Dilokthanakul, N. Investigating partner diversification methods in cooperative multi-agent deep reinforcement learning. In *International Conference on Neural Information Processing*, pp. 395–402. Springer, 2020. 3

Chattopadhyay, P., Hoffman, J., Mottaghi, R., and Kembhavi, A. Robustnav: Towards benchmarking robustness in embodied navigation. *ICCV*, 2021. 2

Chen, B., Song, S., Lipson, H., and Vondrick, C. Visual hide and seek. *arXiv preprint arXiv:1910.07882*, 2019. 3

Chen, C., Jain, U., Schissler, C., Gari, S. V. A., Al-Halah, Z., Ithapu, V. K., Robinson, P., and Grauman, K. Soundspaces: Audio-visual navigation in 3d environments. *ECCV*, 2020. 2

Choudhury, R., Swamy, G., Hadfield-Menell, D., and Dragan, A. D. On the utility of model learning in hri. In *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pp. 317–325. IEEE, 2019. 3

Cully, A., Clune, J., Tarapore, D., and Mouret, J.-B. Robots that can adapt like animals. *Nature*, 521(7553):503–507, 2015. 3

Derek, K. and Isola, P. Adaptable agent populations via a generative model of policies. *Advances in Neural Information Processing Systems*, 34:3902–3913, 2021. 3

Ehsani, K., Han, W., Herrasti, A., VanderBilt, E., Weihs, L., Kolve, E., Kembhavi, A., and Mottaghi, R. Manipulathor: A framework for visual object manipulation. In *CVPR*, 2021. 2

Eysenbach, B., Gupta, A., Ibarz, J., and Levine, S. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018. 3, 5

Fetch. Fetch. http://fetchrobotics.com/, 2020. 2, 3

Gan, C., Schwartz, J., Alter, S., Mrowca, D., Schrimpf, M., Traer, J., De Freitas, J., Kubilius, J., Bhandwaldar, A., Haber, N., et al. Threedworld: A platform for interactive multi-modal physical simulation. *NeurIPS Datasets and Benchmarks Track*, 2021. 2

Gu, J., Chaplot, D. S., Su, H., and Malik, J. Multi-skill mobile manipulation for object rearrangement. *arXiv preprint arXiv:2209.02778*, 2022. 6

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *CVPR*, 2016. 14

Heinrich, J. and Silver, D. Deep reinforcement learning from self-play in imperfect-information games. *arXiv preprint arXiv:1603.01121*, 2016. 3, 6, 7, 14

Heinrich, J., Lanctot, M., and Silver, D. Fictitious self-play in extensive-form games. In *International conference on machine learning*, pp. 805–813. PMLR, 2015. 3

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 14

Hu, H., Lerer, A., Peysakhovich, A., and Foerster, J. "other-play" for zero-shot coordination. In *International Conference on Machine Learning*, pp. 4399–4410. PMLR, 2020. 3

Hu, H., Lerer, A., Cui, B., Pineda, L., Brown, N., and Foerster, J. Off-belief learning. In *International Conference on Machine Learning*, pp. 4369–4379. PMLR, 2021. 3

Jaderberg, M., Dalibard, V., Osindero, S., Czarnecki, W. M., Donahue, J., Razavi, A., Vinyals, O., Green, T., Dunning, I., Simonyan, K., et al. Population based training of neural networks. *arXiv preprint arXiv:1711.09846*, 2017. 3, 14

Jaderberg, M., Czarnecki, W. M., Dunning, I., Marris, L., Lever, G., Castaneda, A. G., Beattie, C., Rabinowitz, N. C., Morcos, A. S., Ruderman, A., et al. Human-level performance in 3d multiplayer games with population-based reinforcement learning. *Science*, 2019. 2, 3, 6, 7

Jain, U., Weihs, L., Kolve, E., Rastegari, M., Lazebnik, S., Farhadi, A., Schwing, A. G., and Kembhavi, A. Two body problem: Collaborative visual task completion. In *CVPR*, 2019. 3

Jain, U., Weihs, L., Kolve, E., Farhadi, A., Lazebnik, S., Kembhavi, A., and Schwing, A. G. A cordial sync: Going beyond marginal policies for multi-agent embodied tasks. In *ECCV*, 2020. 3

Jain, U., Liu, I.-J., Lazebnik, S., Kembhavi, A., Weihs, L., and Schwing, A. G. Gridtopix: Training embodied agents with minimal supervision. In *ICCV*, 2021. 3

Juliani, A., Berges, V.-P., Teng, E., Cohen, A., Harper, J., Elion, C., Goy, C., Gao, Y., Henry, H., Mattar, M., et al. Unity: A general platform for intelligent agents. *arXiv preprint arXiv:1809.02627*, 2018. 3

Kadian, A., Truong, J., Gokaslan, A., Clegg, A., Wijmans, E., Lee, S., Savva, M., Chernova, S., and Batra, D. Sim2real predictivity: Does evaluation in simulation predict real-world performance? *RA-L*, 2020. 12

Kolve, E., Mottaghi, R., Han, W., VanderBilt, E., Weihs, L., Herrasti, A., Gordon, D., Zhu, Y., Gupta, A., and Farhadi, A. AI2-THOR: an interactive 3d environment for visual AI. *arXiv preprint arXiv:1712.05474*, 2019. 2

Krause, A. and Golovin, D. Submodular function maximization. *Tractability*, 3:71–104, 2014. 3

Kurach, K., Raichuk, A., Stańczyk, P., Zajkac, M., Bachem, O., Espeholt, L., Riquelme, C., Vincent, D., Michalski, M., Bousquet, O., et al. Google research football: A novel reinforcement learning environment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 4501–4510, 2020. 3

Lanctot, M., Zambaldi, V., Gruslys, A., Lazaridou, A., Tuyls, K., Pérolat, J., Silver, D., and Graepel, T. A unified game-theoretic approach to multiagent reinforcement learning. *Advances in neural information processing systems*, 30, 2017. 1

Lupu, A., Cui, B., Hu, H., and Foerster, J. Trajectory diversity for zero-shot coordination. In *International Conference on Machine Learning*, pp. 7204–7213. PMLR, 2021. 3, 5, 6, 7, 12, 15

Mahajan, A., Rashid, T., Samvelyan, M., and Whiteson, S. Maven: Multi-agent variational exploration. *Advances in Neural Information Processing Systems*, 32, 2019. 3

McKee, K. R., Leibo, J. Z., Beattie, C., and Everett, R. Quantifying the effects of environment and population diversity in multi-agent reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 36(1):1–16, 2022. 3, 8

Padmakumar, A., Thomason, J., Shrivastava, A., Lange, P., Narayan-Chen, A., Gella, S., Piramithu, R., Tur, G., and Hakkani-Tur, D. TEACh: Task-driven embodied agents that chat. *arXiv*, 2021. URL https://arxiv.org/abs/2110.00534. 2

Patel, S., Wani, S., Jain, U., Schwing, A., Lazebnik, S., Savva, M., and Chang, A. Interpretation of emergent communication in heterogeneous collaborative embodied agents. *ICCV*, 2021. 3

Premack, D. and Woodruff, G. Does the chimpanzee have a theory of mind? *Behavioral and brain sciences*, 1(4):515–526, 1978. 3

Pugh, J. K., Soros, L. B., and Stanley, K. O. Quality diversity: A new frontier for evolutionary computation. *Frontiers in Robotics and AI*, pp. 40, 2016. 3

Puig, X., Ra, K., Boben, M., Li, J., Wang, T., Fidler, S., and Torralba, A. Virtualhome: Simulating household activities via programs. In *CVPR*, 2018. 2

Puig, X., Shu, T., Li, S., Wang, Z., Liao, Y.-H., Tenenbaum, J. B., Fidler, S., and Torralba, A. Watch-and-help: A challenge for social perception and human-ai collaboration. *arXiv preprint arXiv:2010.09890*, 2020. 3

Rahman, A., Fosong, E., Carlucho, I., and Albrecht, S. V. Towards robust ad hoc teamwork agents by creating diverse training teammates. *arXiv preprint arXiv:2207.14138*, 2022. 3

Roman, H. R., Bisk, Y., Thomason, J., Celikyilmaz, A., and Gao, J. Rmm: A recursive mental model for dialog navigation. In *EMNLP Findings*, 2020. 3

Savva, M., Kadian, A., Maksymets, O., Zhao, Y., Wijmans, E., Jain, B., Straub, J., Liu, J., Koltun, V., Malik, J., Parikh, D., and Batra, D. Habitat: A Platform for Embodied AI Research. *ICCV*, 2019. 2, 4

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 14

Sclar, M., Neubig, G., and Bisk, Y. Symmetric machine theory of mind. In *International Conference on Machine Learning*, pp. 19450–19466. PMLR, 2022. 3

Shridhar, M., Thomason, J., Gordon, D., Bisk, Y., Han, W., Mottaghi, R., Zettlemoyer, L., and Fox, D. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *CVPR*, 2020. 2

Stone, P., Kaminka, G. A., Kraus, S., and Rosenschein, J. S. Ad hoc autonomous agent teams: Collaboration without pre-coordination. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010. 3

Strouse, D., McKee, K., Botvinick, M., Hughes, E., and Everett, R. Collaborating with humans without human data. *Advances in Neural Information Processing Systems*, 34:14502–14515, 2021. 1, 3, 4, 6, 7, 15

Szot, A., Clegg, A., Undersander, E., Wijmans, E., Zhao, Y., Turner, J., Maestre, N., Mukadam, M., Chaplot, D., Maksymets, O., et al. Habitat 2.0: Training home assistants to rearrange their habitat. *arXiv preprint arXiv:2106.14405*, 2021. 2, 4, 6, 12, 13

Szot, A., Yadav, K., Clegg, A., Berges, V.-P., Gokaslan, A., Chang, A., Savva, M., Kira, Z., and Batra, D. Habitat rearrangement challenge 2022. https://aihabitat.org/challenge/2022_rearrange, 2022. 12

Team, O. E. L., Stooke, A., Mahajan, A., Barros, C., Deck, C., Bauer, J., Sygnowski, J., Trebacz, M., Jaderberg, M., Mathieu, M., et al. Open-ended learning leads to generally capable agents. *arXiv preprint arXiv:2107.12808*, 2021. 3

Thomason, J., Murray, M., Cakmak, M., and Zettlemoyer, L. Vision-and-dialog navigation. In *CoRL*, 2020. 3

Wang, C., Pérez-D'Arpino, C., Xu, D., Fei-Fei, L., Liu, K., and Savarese, S. Co-gail: Learning diverse strategies for human-robot collaboration. In *Conference on Robot Learning*, pp. 1279–1290. PMLR, 2022. 3

Wani, S., Patel, S., Jain, U., Chang, A., and Savva, M. Multion: Benchmarking semantic map memory using multi-object navigation. *NeurIPS*, 2020. 2

Weihs, L., Jain, U., Salvador, J., Lazebnik, S., Kembhavi, A., and Schwing, A. Bridging the imitation gap by adaptive insubordination. *arXiv preprint arXiv:2007.12173*, 2020a. 3

Weihs, L., Salvador, J., Kotar, K., Jain, U., Zeng, K.-H., Mottaghi, R., and Kembhavi, A. Allenact: A framework for embodied ai research. *arXiv preprint arXiv:2008.12760*, 2020b. 2

Weihs, L., Deitke, M., Kembhavi, A., and Mottaghi, R. Visual room rearrangement. In *CVPR*, 2021a. 2

Weihs, L., Kembhavi, A., Ehsani, K., Pratt, S. M., Han, W., Herrasti, A., Kolve, E., Schwenk, D., Mottaghi, R., and Farhadi, A. Learning generalizable visual representations via interactive gameplay. In *ICLR*, 2021b. 3

Wijmans, E., Kadian, A., Morcos, A., Lee, S., Essa, I., Parikh, D., Savva, M., and Batra, D. Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames. *ICLR*, 2019. 14

Wu, S. A., Wang, R. E., Evans, J. A., Tenenbaum, J. B., Parkes, D. C., and Kleiman-Weiner, M. Too many cooks: Bayesian inference for coordinating multi-agent collaboration. *Topics in Cognitive Science*, 13(2):414–432, 2021. 3

Xia, F., Zamir, A. R., He, Z., Sax, A., Malik, J., and Savarese, S. Gibson env: Real-world perception for embodied agents. *CVPR*, 2018. 2

Xia, F., Shen, W. B., Li, C., Kasimbeg, P., Tchapmi, M., Toshev, A., Martín-Martín, R., and Savarese, S. Interactive gibson: A benchmark for interactive navigation in cluttered environments. *arXiv preprint arXiv:1910.14442*, 2019. 2

Xiang, F., Qin, Y., Mo, K., Xia, Y., Zhu, H., Liu, F., Liu, M., Jiang, H., Yuan, Y., Wang, H., Yi, L., Chang, A. X., Guibas, L. J., and Su, H. Sapien: A simulated part-based interactive environment. In *CVPR*, 2020. 2

Zhao, R., Song, J., Haifeng, H., Gao, Y., Wu, Y., Sun, Z., and Wei, Y. Maximum entropy population based training for zero-shot human-ai coordination. *arXiv preprint arXiv:2112.11701*, 2021. 3

Appendix

To view qualitative behavior, please view our supplemental video. We structure the Appendix as follows:

A Additional details about the Social Rearrangement setup including reward structure.

B Comparison of the diversity objective of the proposed Behavior Diversity Play and prior work of Trajedi (Lupu et al., 2021).

C Necessary implementation details of Behavior Diversity Play for reproducibility – training pipeline, policy architecture, and hyperparams.

D Further description of evaluations we conducted for comparing Behavior Diversity Play, four previous works, and privileged baselines.

E New experimental results including a breakdown of Table 1, more ZSC results, full results for Table 2, and new ablations supplementing Table 3.

The source code can be found at: https://bit.ly/43vNgFk

A. Additional Social Rearrangement Details

We follow the same setup as the original Home Assistant Benchmark (Szot et al., 2021) but with modifications for multi-agent collaboration. We provide more details on Social Rearrangement, particularly the episode setup, more details of the three tasks, and reward structure. See Figure 4 for a visual overview of the tasks in Social Rearrangement.

Episode Setup: At the start of each episode, both agents are randomly placed at a collision free location in the scene such that both agents start at least 2 meters apart. The episode is successful if all target objects are placed within 15cm of their locations. The episode fails if the agents collide or if the maximum episode horizon of 750 time steps is reached. We use the same ReplicaCAD scene split as prescribed by the Home Assistant Benchmark (Szot et al., 2021) and the Habitat Rearrangement Challenge (Szot et al., 2022). For training in each task, we use 10,000 object configurations across 60 layouts of furniture in the scene. While there are only 10,000 rearrangement problems, the agent is randomly spawned each episode, providing infinite variety. At evaluation time, we use 100 object configurations across 20 layouts of furniture in the scene, distinct from the furniture layouts in training.

Set Table: The goal of this task is to remove a bowl from a drawer, a fruit from the fridge, and place the fruit in the bowl on the dinner table. Both the fridge and drawer are initially closed, and the robot must open them before removing objects. The fridge and the drawer are next to each other in the kitchen area of the scene. The position of the dinner table relative to the kitchen changes depending on the scene.

Tidy House: The goal of this task is to move 2 objects from accessible initial locations to their target locations. The objects are spawned across 6 open receptacles, and assigned a goal from one of the 6 receptacles which is different from the starting receptacle. The receptacles start in random locations throughout the house and are always unobstructed to the agent accessing them.

Prepare Groceries: The goal of this task is to move 1 object from an open fridge to the counter and another object on the kitchen table to the fridge. The counter and the fridge are always close to each other. The vicinity of the table to the counter and fridge varies depending on the episode.

Simulation: We partially simulate physics to check for collisions between the robot and objects along with other robots. We kinematically move the robot base with a navigation mesh that defines the navigable regions of the scene according to static obstacles like furniture. We do not allow the robot to slide along obstacles. This setup has been shown to transfer well to the real world (Kadian et al., 2020). The robots are able to collide with each other on the navigation mesh, which terminates the episode with failure.

Reward Structure: The reward function for each task provides a sparse reward for task success, intermediate sparse rewards for completing subgoals, and a per time-step penalty. This reward, described in Equation (3), is the same between all three Social Rearrangement tasks.

$$R(s_t) = 10 \cdot \mathbb{1}_{\text{success}} + 0.5 \cdot \mathbb{1}_{\text{subgoal}} - 0.01 \quad (3)$$

The first term, provides a $+10$ reward for overall task success. We also provide a $+0.5$ reward for either of the agents completing any sub-goal necessary for overall task success. These subgoals include picking up the target object, placing the target object at its goal, and for opening receptacles to access the object if it is necessary for the task as in Set Table. We also add a per time step negative penalty of -0.01 to encourage more efficient solutions. The reward is shared between both agents for a cooperative task.

B. Comparing to Trajedi's Diversity Objective

Here we highlight the differences between the diversity objective from the prior work of Trajedi (Lupu et al., 2021) and the proposed BDP. We assume both methods are latent variable conditioned policies with a discrete N dimension latent space with uniform prior $p(z)$. Let $\pi(\tau|z) = \prod_{t=0}^{T} \pi(a_t|\tau_t, z)$ be the joint action probabilities for policy π. The undiscounted Trajedi objective can be expressed as:

$$\text{Diversity}^{\text{Trajedi}}(\pi) = \text{JSD}(\pi(\tau|z_1), \ldots, \pi(\tau|z_N))$$

$$\propto \mathcal{H}\left(\sum_z \pi(\tau|z)\right) - \sum_z \mathcal{H}\left(\pi(\tau|z)\right)$$

Prepare Groceries | Tidy House | Set Table

Figure 4: Overview of the tasks from Social Rearrangement. Circles indicate a possible starting location for each object and the arrows indicate the goal positions of where the objects should be moved to. Each task requires rearranging two objects. In Prepare Groceries, an object must be moved from the fridge to the counter and an object from a receptacle into the counter. In Tidy House, 2 objects starting on random receptacles throughout the house must be moved to a random goal receptacle. In Set Table, a bowl from the drawer and an apple from the fridge must be moved to the table. Both the drawer and fridge start closed in Set Table. We refer to (Szot et al., 2021) for detailed description and visualizations of the tasks.

Where the first term encourages the collective population to cover a diverse joint action distribution and the second term drives the individual policy joint action distribution to be as compact as possible. Meanwhile the BDP objective is:

$$\text{Diversity}^{\text{BDP}}(\pi) = -\mathcal{H}(p(z|s)) - \frac{1}{N}\sum_z \mathcal{H}(\pi(a|o,z))$$

The first term encourages the policy ID to be predictable from the state distributions generated by all policies. In the second term, BDP encourages the opposite of Trajedi for each policy to be diverse. This is because BDP does not need to balance a diversity of the overall population, which one policy could dominate.

In summary, Trajedi encourages diversity from the tension between the 2 objective terms, the first encouraging coverage of the trajectory space, while the second minimizes overlap between the policies. BDP encourages diversity over observed behaviors while Trajedi encourages diversity over joint action distributions.

C. BDP Implementation Details

In this section we provide further details about hierarchical policy training, policy architecture, and the BDP discriminator architecture.

C.1. BDP Pseudocode

Algorithm 1 presents the pseudocode for stage 1 and stage 2 training of BDP. Lines 1-5 initialize the behavior policy, discriminator, discriminator data buffer, and hyperparamters for training. Lines 7-11 train the behavior policy π^b by randomly sampling latents and then pairing the behavior policy with itself conditioned on these latents. Updating π^b requires computing both the task reward and the diver-

sity Next, lines 12-14 random sample trajectories from the discriminator data buffer and use them to update the discriminator to better predict the latent. Then lines 17-19 update the coordination agent against the fixed π^b. Finally, the coordination agent is evaluated in ZSC.

Algorithm 1 Behavior Diversity Play pseudocode

1: Initial behavior policy π^b
2: Initial discriminator network q_ϕ
3: Diversity objective weight α
4: Discriminator data buffer $\mathcal{B}$ with 100k max size
5: Behavior latent prior $p(z)$
6: **for** each epoch in π^b training **do**
7:    $z^1, z^2 \sim p(z)$
8:    Rollout out agent pair: $(\pi^b_{z^1}, \pi^b_{z^2})$
9:    Compute $\mathcal{L}_J = J(\pi^b_{z^1}, \pi^b_{z^2})$
10:   Compute $\mathcal{L}_D = \text{Diversity}(\pi^b) = \mathbb{E}\log q_\phi(z|\tau) + \mathcal{H}(a|o,z)$
11:   Update π^b with PPO on objective $\mathcal{L}_D + \alpha\mathcal{L}_D$
12:   Add $(\tau^1, z^1), (\tau^2, z^2)$ to $\mathcal{B}$
13:   Update ϕ with random batches from $\mathcal{B}$
14: **end for**
15: Initial coordination policy π^c
16: **for** each epoch in π^c training **do**
17:   $z \sim p(z)$
18:   Rollout out agent pair (π^c, π^b_z)
19:   Update π^c with PPO on objective on $J(\pi^c, \pi^b_z)$
20: **end for**
21: Evaluate π^c in zero-shot coordination.

C.2. Hierarchical Training

In this work, all methods learn a high-level policy that control low-level skills. The high-level policy selects a discrete skill and a discrete parameterization for that skill. All the

possible skills are: open, pick, place, and navigate. Each skill is parameterized by which entity to apply the action to. In Social Rearrangement, the possible entities are the target objects (2 for all tasks), the goal positions (2 for all tasks), and all possible receptacles in the scene (10 in total). All tasks have the same action space. We compute all possible actions given the compatibility of the action with the entity. For example, the agent can never pick up the fridge so that is not a possible action. This gives a 21 element discrete high-level actions for each task and an additional 4 primitive actions for no-op, move-forward, turn-left, and turn-right. At each step the agent selects from each of the 25 possible actions.

Training this high-level policy with on-policy RL requires changes to the RL training pipeline due to the separation between high-level actions and low-level robot actions. The robot is executing low-level actions at every time step by controlling the base, arm, and gripper, but the high-level policy is not making decisions at every time step. We only want to learn from the transitions where the high-level policy is acting in the environment. Therefore, we change the rollout collection in PPO to conditionally write the transition to the rollout data storage if the high-level policy acted in that time step. For example, when the robot is navigating, it is only executing base actions and the high-level policy is not acting These variable rollout sizes allow leveraging an efficient vectorized environment rollout implementation.

C.3. Policy and Discriminator Network Architectures

We first describe the policy neural network architecture. The ResNet18 (He et al., 2016) first encodes the 256×256 depth visual observation to a 512-dimension vector. Next, these visual inputs are concatenated with an 18-dimension state information vector which includes the joint angles (8D) along with heading and distance to the object starts (4D), goals (4D), and other agent (2D). These are then fed into a 2-layer LSTM (Hochreiter & Schmidhuber, 1997) with 512 hidden units. This then produces a 512-dimensional vector which is separately fed into separate policy and value function networks. Each of these networks are a single layer linear layer.

All policies are trained with PPO (Schulman et al., 2017). DD-PPO (Wijmans et al., 2019) is used to distribute training to 4 GPUs. Each GPU runs 32 parallel environments and collects 128 simulation steps. We run on NVIDIA V100 GPUs. We train the behavior generator policy in for 100 million low-level steps and the coordination agent for another 100 million steps. In the second stage, we initialize the coordination agent, including the visual encoder, from scratch.

The discriminator is modeled as an MLP with 2 hidden layers with 512 hidden units. For the privileged state trajecto-

ries, the discriminator takes as input the robot x, y positions and a list of actions executed in a window spanning the last 40 steps. In practice, it is difficult to tell the difference between different behavior latents from the first few time steps. Therefore, we do not provide the discriminator reward for the first 10% of maximum time steps in the trajectory. We sample a new z^1, z^2 pair once every 10 policy updates for better training stability in both stages. The discriminator is updated every policy update step with a buffer of at most the last 100,000 agent samples.

C.4. Hyperparameter Selections

For all methods we use the same hyperparameters unless stated otherwise. For PPO policy optimization parameters, we use a learning rate of 0.0003, 2 epochs per-update, 2 mini-batches per-epoch, clip parameter of 0.2, an entropy coefficient of 0.001, and clip the gradient norm to a max value of 0.2. For return estimation, we use a discount factor of $\gamma = 0.99$, GAE with $\lambda = 0.95$. For the discriminator in BDP we also use a learning rate of 0.0003. BDP weighs the diversity reward by 0.01 before adding it to the task reward.

D. Additional Evaluation Details

We include supplementary information about how evaluation is conducted for Social Rearrangement. Particularly, task plans for scripted holdout agents, how we implemented baselines, what we gave as input to PBT-State, and how we obtained qualitative results.

D.1. Scripted Holdout Agents

Here we detail the task plans that the scripted agents execute for each task. For every task, we include a scripted agent that only executes no-ops and two agents that execute a fixed portion of the task. These two scripted agents will do half of the task involving interactions with only 1 object. For example, in the Set Table task, we have 1 scripted agent that will only rearrange the fruit, and another scripted agent that will only rearrange the bowl. For training the learned coordination agents, we train them in two agent populations in the same manner as GT Coord.

D.2. Baselines

We include the necessary details for implementing baselines.

Self-Play (SP) (Heinrich & Silver, 2016) and **Population-Based Training (PBT)** (Jaderberg et al., 2017) only vary in how they pair agents while training the population in stage 1. SP only pairs the policies with themselves while PBT pairs policies with other policies in the population. For **Fictitious Co-Play (FCP)** we use 3 checkpoints from each agent population from the start, middle and end of training

to the final learned population from stage 1, as in (Strouse et al., 2021). We then pair the coordination agent against these older agents in stage 2 training. With **Trajectory Diversity (TrajeDi)** (Lupu et al., 2021) we do not use any discounting factor in the JSD objective.

We fix the policy size to be the same between the behavior latent conditioned policy in BDP and *each* policy in the population of the baselines which maintain a set of N distinct policies. While this increases the parameter count, we sufficiently train all policies to convergence in both stages with 100M steps of training experience per stage.

The hyperparameters are described in Appendix C.4.

D.3. Ground Truth State for PBT-State Baseline

Here we describe the priviliedged ground truth state that the baseline *PBT-State (Oracle, No-Vision)* takes as input. This ground truth state input consists of a set of binary predicates including:

- `robot_at`(R, X) if robot R is at receptacle, object, or goal X.
- `is_holding`(R) if robot R is holding an object.
- `object_at`(X, Y) if the object X is at goal location or receptacle Y.

The truth values are enumerated for all possible inputs. This forms a 1D vector which is passed to the policy. Both agents share the same fully observable state as input.

D.4. Qualitative Result Details (Supplements Figure 3)

Here we include details of how we created Figure 3, particularly, how we measured the portion of subgoals completed by the coordination agent in Figure 3. Let E_i denote the Bernoulli random variable that represents if the coordination agent executed event i when paired with a partner π. π can belong to both the training population or the holdout population set. We then record $p(E_i = 1|\pi)$ to analyze how likely the coordination agent is to perform certain interaction types when paired with π, averaged over a 100 episodes. An agent biased towards an event E_j would have a high $p(E_j = 1|\pi)$ for all partner agents. An adaptive agent on the other hand, will have different $p(E_j = 1|\pi)$, depending on its partner. Darker cells in Figure 3 indicate a higher value for $p(E_1 = 1|\pi)$, meaning the coordination agent is more likely to achieve this subgoal.

E. Additional Experiments & Ablations

E.1. Extended ZSC Results (Supplements Table 1)

In this section we present a more detailed breakdown of the ZSC results from Table 1. Specifically, we separately show the ZSC performance between the scripted and learned

unseen agents in the ZSC evaluation in Table 5. These results indicate that in general ZSC is harder with scripted vs. learned unseen agents. For example, in the Tidy House task, BDP achieves 74.14% success rate when paired with learned agents, but only 46.90% success rate when paired with scripted unseen agents. This same also trend holds for all other methods and tasks.

We experimented with the impact of different state inputs on BDP's performance. We compared a version of BDP, which takes RGB instead of depth images as input. We found that BDP's performance remains mostly unchanged on the Prepare Groceries task when using RGB instead of depth images. The ZSC success rate of BDP is 70% with RGB (averaged across 2 seeds) versus 76% with depth. We also evaluated an oracle version of BDP, like PBT-State, which takes the same privileged state information as input. On Prepare Groceries, this method achieves an average ZSC success rate of 80% (averaged across 2 seeds), compared to PBT-State with 77% and non-privileged BDP with 75% success rate.

E.2. Policy Architecture Ablation (Supplements Table 3)

In Table 3, we ablate the shared policy architecture in BDP. The shared policy architecture lets BDP be more sample-efficient by sharing weights, while generating behaviors through a behavior latent space z. We create two versions of BDP:

BDP - [Latent (Shared Enc)] replaces behavior latent space z with separate policies per agent, initialized randomly, but shares the visual encoder weights between all agents, to enable sample-efficiency. Essentially, the policies share the ResNets, but learn separate LSTM and MLP weights.

BDP - [Latent (Sep Enc)] replaces the shared policy architecture with entirely different networks per agent initialized randomly, with no latent space, and no shared visual encoder. BDP - [Latent (Sep Enc)] is the same as PBT, but trained with a discriminator diversity reward from BDP.

	BDP - [Latent (Shared Enc)]	BDP - [Latent (Sep Enc)]	BDP
Train-Pop Eval	50.09 ± 0.01	39.19 ± 0.00	74.81 ± 0.01
ZSC Eval	33.40 ± 0.06	26.76 ± 0.06	46.43 ± 0.08

Table 4: Ablations on the shared policy architecture in BDP.

Table 4 shows that both the training and ZSC evaluation performance decrease as we decrease weight sharing through the behavior latent space, implying that both are essential for learning an adaptive coordination agent.

	PBT-State	GT Coord	SP	PBT	FCP	TrajeDi	BDP
Set Table							
Train-Pop Eval	70.74 ± 0.00	90.52 ± 0.05	57.74 ± 0.01	46.67 ± 0.02	29.90 ± 0.07	43.24 ± 0.09	74.81 ± 0.01
ZSC Eval	50.39 ± 0.09	-	17.97 ± 0.04	30.34 ± 0.04	37.50 ± 0.04	32.52 ± 0.04	**46.43 ± 0.08**
Scripted Unseen	35.94 ± 26.56	-	7.29 ± 5.02	27.08 ± 7.47	32.81 ± 1.56	26.17 ± 0.05	37.50 ±2.31
Learned Unseen	55.21 ± 8.49	-	21.53 ± 5.24	31.42 ± 4.95	38.28 ± 5.09	34.64 ± 5.17	47.92 ± 9.64
Tidy House							
Train-Pop Eval	74.90 ± 21.59	92.28 ± 1.66	34.18 ± 6.05	36.13 ± 0.98	12.04 ± 2.28	39.65 ± 0.59	73.83 ± 7.03
ZSC Eval	68.08 ± 0.09	-	52.67 ± 0.06	56.88 ± 0.07	34.07 ± 0.09	63.58 ± 0.05	**66.71 ± 0.05**
Scripted Unseen	43.26 ± 15.71	-	35.72 ± 9.14	38.54 ± 11.52	17.35 ± 11.29	39.95 ± 8.87	46.90 ± 9.11
Learned Unseen	77.39 ± 10.11	-	59.02 ± 7.69	63.76 ± 8.75	41.04 ± 10.80	72.44 ± 5.08	74.14 ± 3.61
Prepare Groceries							
Train-Pop Eval	85.74 ± 2.82	93.63 ± 0.28	47.07 ± 29.88	69.34 ± 1.76	44.40 ± 6.38	34.56 ± 27.94	89.67 ± 2.51
ZSC Eval	77.01 ± 0.05	-	56.04 ± 0.07	56.08 ± 0.09	30.00 ± 0.07	53.84 ± 0.08	**75.85 ± 0.05**
Scripted Unseen	60.00 ± 0.00	-	54.86 ± 17.89	50.70 ± 15.56	29.06 ± 12.63	42.09 ± 17.53	60.42 ± 15.76
Learned Unseen	83.38 ± 6.45	-	56.49 ± 7.18	58.10 ± 11.37	30.71 ± 7.31	58.25 ± 9.74	81.64 ± 3.52

Table 5: Detailed breakdown of the ZSC success rates from Table 1 by unseen agent type: scripted or learned. Across most methods and tasks, methods achieve lower success rates when paired with unseen scripted agents.