The MAGICAL Benchmark for Robust Imitation

Sam Toyer   **Rohin Shah**   **Andrew Critch**   **Stuart Russell**
Department of Electrical Engineering and Computer Sciences
University of California, Berkeley
{sdt,rohinmshah,critch,russell}@berkeley.edu

Abstract

Imitation Learning (IL) algorithms are typically evaluated in the same environment that was used to create demonstrations. This rewards precise reproduction of demonstrations in one particular environment, but provides little information about how robustly an algorithm can generalise the demonstrator's intent to substantially different deployment settings. This paper presents the MAGICAL benchmark suite, which permits *systematic* evaluation of generalisation by quantifying robustness to different kinds of distribution shift that an IL algorithm is likely to encounter in practice. Using the MAGICAL suite, we confirm that existing IL algorithms overfit significantly to the context in which demonstrations are provided. We also show that standard methods for reducing overfitting are effective at creating narrow perceptual invariances, but are not sufficient to enable transfer to contexts that require substantially different behaviour, which suggests that new approaches will be needed in order to robustly generalise demonstrator intent. Code and data for the MAGICAL suite is available at `https://github.com/qxcv/magical/`.

1 Introduction

Imitation Learning (IL) is a practical and accessible way of programming robots to perform useful tasks [6]. For instance, the owner of a new domestic robot might spend a few hours using tele-operation to complete various tasks around the home: doing laundry, watering the garden, feeding their pet salamander, and so on. The robot could learn from these demonstrations to complete the tasks autonomously. For IL algorithms to be useful, however, they must be able to learn how to perform tasks from few demonstrations. A domestic robot wouldn't be very helpful if it required thirty demonstrations before it figured out that you are deliberately washing your purple cravat separately from your white breeches, or that it's important to drop bloodworms *inside* the salamander tank rather than next to it. Existing IL algorithms assume that the environment observed at test time will be identical to the environment observed at training time, and so they cannot generalise to this degree. Instead, we would like algorithms that solve the task of *robust IL*: given a small number of demonstrations in one training environment, the algorithm should be able to generalise the intent behind those demonstrations to (potentially very different) deployment environments.

One barrier to improved algorithms for robust IL is a lack of appropriate benchmarks. IL algorithms are commonly tested on Reinforcement Learning (RL) benchmark tasks, such as those from OpenAI Gym [37, 23, 27, 8]. However, the demonstrator intent in these benchmarks is often trivial (e.g. the goal for most of Gym's MuJoCo tasks is simply to run forward), and limited variation in the initial state distribution means that algorithms are effectively being evaluated in the same setting that was used to provide demonstrations. Recent papers on Inverse Reinforcement Learning (IRL)—which is a form of IL that infers a reward under which the given demonstrations are near-optimal—have instead used "testing" variants of standard Gym tasks which differ from the original demonstration environment [17, 39, 32, 33]. For instance, Fu et al. [17] trained an algorithm on demonstrations from the standard "Ant" task from Gym, then tested on a variant of the task where two of the creature's

Demonstrations   **Colour test variant**   **Shape test variant**   **Layout test variant**
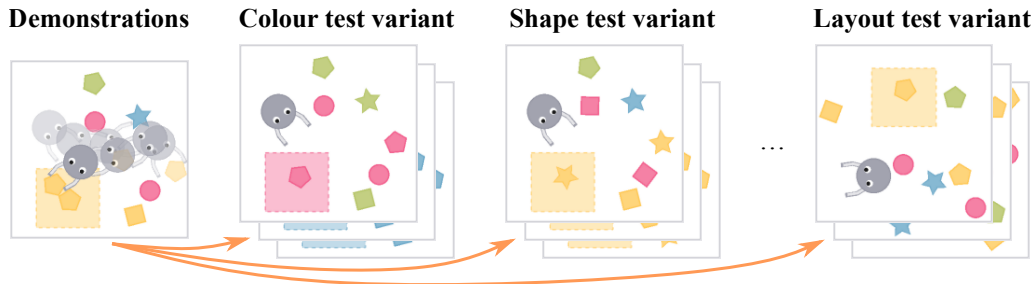
Figure 1: Unlike existing IL benchmarks, MAGICAL makes a distinction between *demonstration* and *test* variants of a task. Demonstrations are all provided in one particular configuration of the world (the "demonstration variant"). The learnt policy (or reward function) is then evaluated across a set of *test variants*, each of which randomise one aspect of the environment, such as block colour or shape, environment layout, dynamics, etc. This makes it possible to understand precisely which aspects of the underlying task the algorithm has been able to infer from demonstrations.

four legs were disabled. Splitting the environment into such "training" and "test" variants makes it possible to measure the degree to which an algorithm overfits to task-irrelevant features of the supplied demonstrations. However, there is so far no standard benchmark for robust IL, and researchers must instead use ad-hoc adaptations of RL benchmarks—such as the modified Ant benchmark and similar alternatives discussed in Section 5—to evaluate intent generalisation.

To address the above issues, we introduce the Multitask Assessment of Generalisation in Imitative Control ALgorithms (MAGICAL). Each MAGICAL task occurs in the same 2D "MAGICAL universe", where environments consist of a robot with a gripper surrounded by a variable number of objects in a fixed-size workspace. Each task is associated with a *demonstration variant*, which is a fixed initial state from which all human demonstrations are provided. A task is also associated with a set of *test variants* for which no demonstrations are provided. As illustrated in Fig. 1, the test variants each randomise a different aspect of the world, such as object colour, transition dynamics, or object count. Randomising attributes of objects and the physics of the world lets us evaluate the ability of a robust IL algorithm to perform *combinatorial generalisation* [5]. For instance, given a demonstration of the robot pushing a red square across the workspace, an algorithm should be able to push a yellow circle across the workspace; given a demonstration of three green and yellow blocks being placed in a line, an algorithm should also be able to place four red and blue blocks in a line; and so on.

MAGICAL has several advantages over evaluation methods for standard (non-robust) IL:

- **No "training on the test set".** Evaluating in the same setting that was used to give demonstrations allows algorithms to exploit features that might not be present during deployment. Having separate test variants for a task allows us to identify this kind of overfitting.
- **Distinguishes between different types of transfer.** Each test variant evaluates robustness to a distinct, semantically meaningful axis of variation. This makes it possible to characterise precisely which aspects of the provided demonstrations a given algorithm is relying on, and to diagnose the causes of over- or under-fitting.
- **Enables knowledge reuse between tasks.** Each MAGICAL task requires similar concepts and low-level skills to solve. Different tasks can therefore provide "background knowledge" for multi-task and meta-IL algorithms, such as knowledge that objects can have different colours, or that objects with different shapes respond in a particular way when grasped.

Our experiments in Section 4 demonstrate the brittleness of standard IL algorithms, particularly under large shifts in object position or colour. We also show that common methods for improving generalisation—such as multitask training, data augmentation, and alternative camera views— sometimes improve robustness to small changes, but still fail to generalise to more extreme ones.

2 MAGICAL: Systematically evaluating robust IL

We will now introduce the main elements of the MAGICAL benchmark. We first describe the abstract setup of our benchmark, then detail the specific tasks and variants available in the benchmark.

2.1 Benchmark setup

The MAGICAL benchmark consists of a set of *tasks* $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_m$. Each task can in turn be broken down into *variants* of a single base Markov Decision Process (MDP) that provide different state distributions and "physics" for an agent. Formally, each task $\mathcal{T} = (S, v^D, \mathcal{V})$ consists of a *scoring function* $S(\tau)$, a *demonstration variant* v^D, and a set of additional *test variants* $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}$. The scoring function $S(\tau)$ takes a trajectory $\tau = (s_0, a_0, s_1, a_1, \ldots, s_T, a_T)$ and assigns it a score $S(\tau) \in [0, 1]$, where 0 is the score of a no-op policy, and 1 is the score of a perfect demonstrator. Unlike a reward function, $S(\tau)$ need not be Markovian. In order to evaluate generalisation, the variants are split into a single demonstration variant v^D and a set of test variants $\mathcal{V}$.

In our domestic robotics analogy, v^D might represent a single room and time-of-day in which demonstrations for some domestic task collected, while each test variant $v \in \mathcal{V}$ could represent a different room, different time-of-day, and so on. Algorithms are expected to be able to take demonstrations given only in demonstration variant v^D, then generalise the intent behind those demonstrations in order to achieve the same goal in each test variant $v \in \mathcal{V}$. This can be viewed either as a form of domain transfer, or as ordinary generalisation using only a single sample from a hypothetical distribution over all possible variants of each task.

Formally, each variant associated with a task $\mathcal{T}$ defines a distribution over reward-free MDPs. Specifically, a variant $v = (p_0, p_\rho, H)$ consists of an *initial state distribution* $p_0(s_0)$, a *dynamics distribution* $p_\rho(\rho)$, and a *horizon* H. States are fully observable via an image-based observation space. Further, all variants have the same state space, the same observation space, and the same action space, which we discuss below. In addition to sampling an initial state $s_0 \sim p_0(s_0)$, at the start of each trajectory, a *dynamics vector* $\rho \in \mathbb{R}^d$ is also sampled from the dynamics distribution $p_\rho(\rho)$. Unlike the state, ρ is not observable to the agent; this vector controls aspects of the dynamics such as friction and motor strength. Finally, the horizon H defines a fixed length for all trajectories sampled from the MDP associated with the variant v. Each variant associated with a given task has the same fixed horizon H to avoid "leaking" information about the goal through early termination [27].

All tasks and variants in the MAGICAL benchmark share a common continuous state space $\mathcal{S}$. A state $s \in \mathcal{S}$ consists of a configuration (pose, velocity, and gripper state) q_R for the robot, along with object configurations $\mathcal{O} = \{o_1, \ldots, o_E\}$ (where the number of objects in s_0 may be random). In addition to pose, each object configuration o_i includes an object *type* and a number of fixed attributes. Objects can be of two types: *blocks* (small shapes that can be pushed around by the agent) and *goal regions* (coloured rectangles that the agent can drive over, but not push around). Each block has a fixed shape (square, pentagon, star, or circle) and colour (red, green, blue, or yellow). Each goal region has a fixed colour, width, and height. In order to facilitate generalisation across tasks with a different number of objects, we use a common image-based observation space and discrete, low-level action space for all tasks, which we describe in detail in Appendix A.1. At an implementation level, we expose each variant of each task as a distinct Gym environment [8], which makes it straightforward to incorporate MAGICAL into existing IL and RL codebases.

2.2 Tasks and variants

With the handful of building blocks listed in the previous section, we can create a wide variety of tasks, which we describe in Section 2.2.1. The object-based structure of the environment also makes it easy to evaluate combinatorial generalisation by randomising one or more attributes of each object while keeping the others fixed, as described in Section 2.2.2.

2.2.1 Tasks

Tasks in the MAGICAL suite were chosen to balance three desiderata. First, given a handful of trajectories from the demonstration variant of a task, it should be possible for a human observer to infer the goal with sufficient accuracy to solve the test variants. We have chosen demonstration variants (illustrated in Fig. 2) that rule out obvious misinterpretations, like mistakenly identifying colour as being task-relevant when it is not. Second, the tasks should be constructed so that they involve complementary skills that meta- and multi-task learning algorithms can take advantage of. In our tasks, these "shared skills" include block manipulation; identification of colour or shape; and relational reasoning. Third, the demonstration variant of each task must be solvable by existing (non-robust) IL algorithms. This ensures that the main challenge of the MAGICAL suite lies in
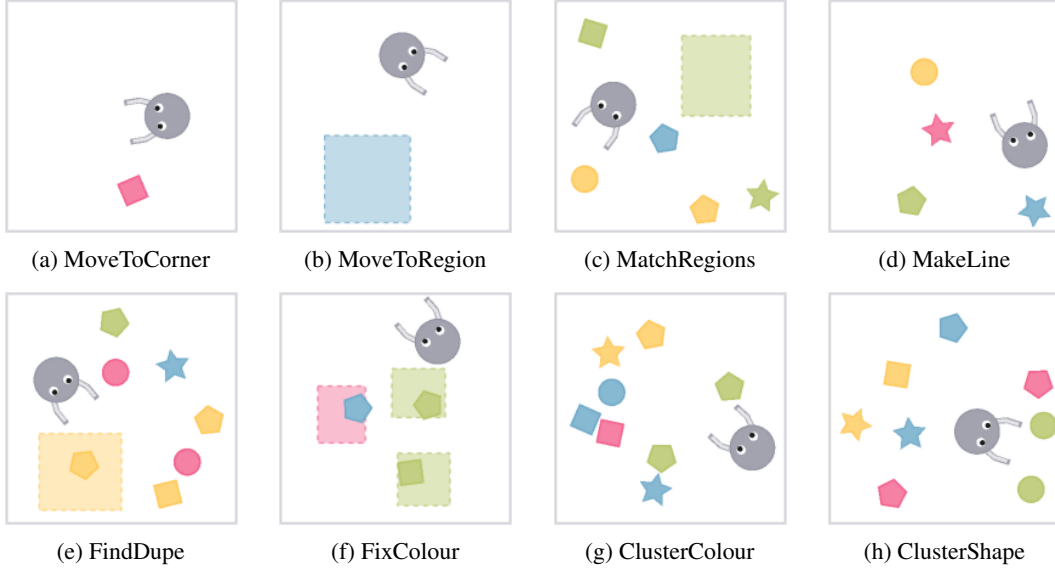
Figure 2: Demonstration variants for MAGICAL tasks. Appendix A shows an example demonstration for each task.

generalising to the test variants (robust IL), as opposed to reproducing the demonstrator's behaviour in the demonstration variant (standard IL). This section briefly describes the resulting tasks; detailed discussion of horizons, score functions, etc. is deferred to Appendix A.

Move to Corner (MTC) The robot must push a single block from one corner of the workspace to the diagonally opposite corner. Test variants are constrained so that the robot and block start near the lower right corner. The score is $S(\tau) = 1$ if the block finishes the trajectory in the top left eighth of the workspace, and decreases to zero as the block gets further from the top left corner.

MoveToRegion (MTR) The robot must drive inside a goal region and stay there. There are no blocks in the demonstration or test variants. Further, variants only have one goal region to ensure that the objective is unambiguous. The agent's score is $S(\tau) = 1$ if the robot's body is inside the goal region at the end of the trajectory, and $S(\tau) = 0$ otherwise.

MatchRegions (MR) There is a set of coloured blocks and a goal region visible to the robot, and the robot must push all blocks of the same colour as the goal region into the goal region. Test variants are constrained to have one goal region and at least one block of the same colour as that goal region. A perfect score is given upon termination if the goal regions contains all and only blocks of the goal region's colour, with penalties for excluding any blocks of the goal colour, or including other blocks.

MakeLine (ML) Here the objective is for the robot to arrange all the blocks in the workspace into a single line. A perfect score is given if all blocks are approximately colinear and close together; a penalty is given for each block that does not form part of the longest identifiable line. Refer to Appendix A for details on how a "line" is defined.

FindDupe (FD) Similar to MatchRegions, except the goal region initially contains a "query" block which has the same shape and colour as at least one other block outside the goal region. The objective is to push at least one of those duplicate blocks into the goal region, which yields a perfect score. Penalties are given for knocking the query block out of the goal region, failing to find a duplicate, or pushing non-duplicate blocks into the goal region.

FixColour (FC) In each variant of this task, the workspace contains a set of non-overlapping goal regions. Each goal region contains a single block, and exactly one block in the workspace will have a different colour to its enclosing goal region. A perfect score is given for pushing that block out of its enclosing goal region and into an unoccupied part of the workspace, without disturbing other blocks.

ClusterColour (CC) and ClusterShape (CS) The robot is confronted with a jumble of blocks of different colours and shapes. It must push the blocks into clusters of either uniform colour (in the CC

task), or uniform shape (in the CS task). Test variants are constrained to include at least one block of each colour and each shape. A perfect score is given for creating four spatially distinct clusters corresponding to each of the four colours (CC) or shapes (CS), with a penalty proportional to the number of blocks that do not belong to an identifiable cluster.

2.2.2 Test variants

In addition to its demonstration variant, each of the tasks above has a set of associated test variants. Some variants are not supported for tasks that do not have any blocks, or where the initial state is otherwise restricted, as documented in Table 2 of Appendix A.

Jitter Takes demo variant and randomly perturbs the poses of the robot and all objects by up to 5% of the maximum possible range. Failure on this variant indicates severe overfitting to the demonstration variant (e.g. by memorising action sequences).

Layout Completely randomises the position and orientation of the robot and all blocks, plus position and dimensions of goal regions; a more challenging version of Jitter.

Colour Block colours are randomly reassigned as appropriate for the task. This tests whether the agent is responsive to block colour (when it is task-relevant, like in CC and MR), or is correctly ignorant of colour (when it is irrelevant, like in MTC and CS).

Shape Similar to Colour, except the shapes of blocks are randomised rather than the colours. This variant either tests for appropriate responsiveness or invariance to shape, depending on whether shape is task-relevant.

CountPlus The number of blocks is randomised (along with shape, colour, and position) to test whether the agent can handle "larger" or "smaller" problems (i.e. "generalisation to n" [35]).

Dynamics Subtly randomises friction of objects and the robot against the workspace, as well as force of robot motors (for rotation, forward/backward motion, and the gripper).

All Combines all applicable variants for a task (e.g. Layout, Colour, Shape, CountPlus, Dynamics).

3   Data-efficient intent disambiguation

Succeeding at the MAGICAL benchmark requires agents to generalise the intent behind a set of demonstrations to substantially different test variants. We anticipate that resolving the ambiguity inherent in this task will require additional sources of information about the demonstrator's goal beyond just single-task demonstrations. In this section, we review two popular non-robust IL algorithms, as well as some common ways in which alternative sources of goal information are incorporated into these algorithms to improve generalisation.

3.1   Baseline methods

Our first baseline method is Behavioural Cloning (BC). BC treats a demonstration dataset $\mathcal{D}$ as an undistinguished collection of state-action pairs $\{(s_1, a_1), \ldots, (s_M, a_M)\}$. It then optimises the parameters θ of the policy $\pi_\theta(a \mid s)$ via gradient descent on the log loss

$$\mathcal{L}_{\mathrm{bc}}(\theta; \mathcal{D}) = - \mathop{\mathbb{E}}_{\mathcal{D}} \log \pi_\theta(a \mid s) \,.$$

Our second baseline method is Generative Adversarial IL (GAIL) [23]. GAIL casts IL as a GAN problem [19], where the generator $\pi_\theta(a \mid s)$ is an imitation policy, and the discriminator $D_\psi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ is tasked with distinguishing imitation behaviour from expert behaviour. Specifically, GAIL uses alternating gradient descent to approximate a saddle point of

$$\max_\theta \min_\psi \left\{ \mathcal{L}_{\mathrm{adv}}(\theta, \psi; \mathcal{D}) = - \mathop{\mathbb{E}}_{\pi_\theta} \log D_\psi(s, a) - \mathop{\mathbb{E}}_{\mathcal{D}} \log(1 - D_\psi(s, a)) + \lambda H(\pi_\theta) \right\} \,,$$

where H denotes entropy and $\lambda \geq 0$ is a policy regularisation parameter.

We also included a slight variation on GAIL which (approximately) minimises Wasserstein divergence between occupancy measures, rather than Jensen-Shannon divergence. We refer to this baseline as

WGAIL-GP. In analogy with WGAN-GP [20], WGAIL-GP optimises the cost

$$\max_\theta \min_\psi \left\{ \mathcal{L}_{\text{w-gp}}(\theta, \psi; \mathcal{D}) = \mathop{\mathbb{E}}_{\mathcal{D}} D_\psi(s, a) - \mathop{\mathbb{E}}_{\pi_\theta} D_\psi(s, a) + \lambda_{\text{w-gp}} \mathop{\mathbb{E}}_{\frac{1}{2}\pi_\theta + \frac{1}{2}\mathcal{D}} (\|\nabla_s D(s, a)\|_2 - 1)^2 \right\},$$

The gradient penalty approximately enforces 1-Lipschitzness of the discriminator by encouraging the norm of the gradient to be 1 at points between the support of π_θ and $\mathcal{D}$. Since actions were discrete, we did not enforce 1-Lipschitzness with respect to the action input. We also did not backpropagate gradients with respect to the gradient penalty back into the policy parameters θ, since the gradient penalty is only intended as a soft constraint on D.

In addition to these baselines, we also experimented with Apprenticeship Learning (AL). Unfortunately we could not get AL to perform well on most of our tasks, so we defer further discussion of AL to Appendix B.

3.2   Using multi-task data

As noted earlier, the MAGICAL benchmark tasks have similar structure, and should in principle benefit from multi-task learning. Specifically, say we are given a multi-task dataset $\mathcal{D}_{\text{mt}} = \{\mathcal{D}(\mathcal{T}_i, v_i^D, n_i)\}_{i=1}^M$, where $\mathcal{D}(\mathcal{T}_i, v, n)$ denotes a dataset of n trajectories for variant v of task $\mathcal{T}_i$. For BC and GAIL, we can decompose the policy for task $\mathcal{T}_i$ as $\pi_\theta^i = g_\theta^i \circ f_\theta$, where $f_\theta : \mathcal{S} \to \mathbb{R}^d$ is a multi-task state encoder, while $g_\theta^i : \mathbb{R}^d \to \Delta(\mathcal{A})$ is a task-specific policy decoder. We can also decompose the GAIL discriminator as $D_\psi^i = s_\psi^i \circ r_\psi$, where $r_\psi : \mathbb{S} \times \mathbb{A} \to \mathbb{R}^d$ is shared and $s_\psi^i : \mathbb{R}^d \to [0, 1]$ is task-specific. We then modify the BC and GAIL objectives to

$$\mathcal{L}_{\text{bc}}(\theta; \mathcal{D}_{\text{mt}}) = \sum_{i=1}^M \mathcal{L}_{\text{bc}}(\theta; \mathcal{D}(\mathcal{T}_i, v_i^D, n_i)) \text{ and } \mathcal{L}_{\text{adv}}(\theta, \psi; \mathcal{D}_{\text{mt}}) = \sum_{i=1}^M \mathcal{L}_{\text{adv}}(\theta, \psi; \mathcal{D}(\mathcal{T}_i, v_i^D, n_i)).$$

3.3   Domain-specific priors and biases

Often the most straightforward way to improve the robustness of an IL algorithm is to constrain the solution space to exclude common failure modes. For instance, one could use a featurisation that only captures task-relevant aspects of the state. Such priors and biases are generally domain-specific; for the image-based MAGICAL suite, we investigated two such biases:

- **Data augmentation:** In MAGICAL, our score functions are invariant to whether objects are repositioned or rotated slightly; further, human observers are typically invariant to small changes in colour or local image detail. As such, we used random rotation and translation, Gaussian noise, and colour jitter to augment training data for the BC policy and GAIL discriminator. This can be viewed as a post-hoc form of domain randomisation, which has previously yielded impressive results in robotics and RL [2]. We found that GAIL discriminator augmentations were necessary for the algorithm to solve more-challenging tasks, as previously observed by Zolna et al. [41]. In BC, we found that policy augmentations improved performance on both demonstration and test variants.

- **Ego- and allocentric views:** Except where indicated otherwise, all of the experiments in Section 4 use an *egocentric* perspective, which always places the agent at the same position (and in the same orientation) within the agent's field of view. This contrasts with an *allocentric* perspective, where observations are focused on a fixed region of the environment (in our case, the extent of the workspace), rather than following the agent's position. In the context of language-guided visual navigation, Hill et al. [22] previously found that an egocentric view improved generalisation to unseen instructions or unseen visual objects, despite the fact that it introduces a degree of partial observability to the environment.

4   Experiments

Our empirical evaluation has two aims. First, to confirm that single-task IL methods fail to generalise beyond the demonstration variant in the MAGICAL suite. Second, to analyse the ways in which the common modifications discussed in Section 3 affect generalisation.

Method	Demo	Jitter	Layout	Colour	Shape
BC (single-task)	0.64±0.29	0.56±0.27	0.14±0.16	0.39±0.30	0.52±0.33
Allocentric	0.58±0.33	0.48±0.29	0.04±0.04	0.42±0.32	0.50±0.37
No augmentations	0.55±0.37	0.37±0.30	0.12±0.15	0.33±0.30	0.41±0.33
No trans./rot. aug.	0.55±0.37	0.41±0.31	0.13±0.15	0.33±0.30	0.43±0.35
Multi-task	0.59±0.33	0.53±0.31	0.14±0.18	0.30±0.25	0.51±0.36
GAIL (single-task)	0.72±0.35	0.69±0.33	0.22±0.23	0.27±0.24	0.60±0.42
Allocentric	0.57±0.46	0.49±0.40	0.03±0.03	0.39±0.36	0.50±0.45
No augmentations	0.44±0.42	0.32±0.31	0.09±0.12	0.19±0.23	0.28±0.33
WGAIL-GP	0.42±0.38	0.33±0.32	0.14±0.20	0.10±0.11	0.33±0.33
Multi-task	0.37±0.41	0.33±0.36	0.16±0.25	0.11±0.12	0.28±0.36

Table 1: Score statistics for a subset of variants and compared algorithms. We report the mean and standard deviation of test scores aggregated across *all* tasks, with five seeds per algorithm and task. Darker colours indicate higher scores.

4.1 Experiment details

We evaluated all the single- and multi-task algorithms in Section 3, plus augmentation and perspective ablations, on all tasks and variants. Each algorithm was trained five times on each task with different random seeds. In each run, the training dataset for each task consisted of 10 trajectories from the demo variant. All policies, value functions, and discriminators were represented by Convolutional Neural Networks (CNNs). Observations were preprocessed by stacking four temporally adjacent RGB frames and resizing them to 96×96 pixels. For multi-task experiments, task-specific weights were used for the final fully-connected layer of each policy/value/discriminator network, but weights of all preceding layers were shared. The BC policy and GAIL discriminator both used translation, rotation, colour jitter, and Gaussian noise augmentations by default. The GAIL policy and value function did not use augmented data, which we found made training unstable. Complete hyperparameters and data collection details are listed in Appendix B. The IL algorithm implementations that we used to generate these results are available on GitHub,[1] as is the MAGICAL benchmark suite and all demonstration data.[2]

4.2 Discussion

Due to space limitations, this section addresses only a selection of salient patterns in the results. Table 1 provides score statistics for a subset of algorithms and variants, averaged across *all* tasks. See Section 2.2.1 for task name abbreviations (MTR, FC, etc.). Because the tasks vary in difficulty, pooling across all tasks yields high score variance in Table 1. Actual score variance for each method is much lower when results are constrained to just one task; refer to Appendix C for complete results.

Overfitting to position All algorithms exhibited severe overfitting to the *position* of objects. The Layout, CountPlus, and All variants yielded near-zero scores in all tasks except MTC and MTR, and on many tasks there was also poor transfer to the Jitter variant. For some tasks, we found that the agent would simply execute the same motion regardless of its initial location or the positions of task-relevant objects. This was true on the FC task, where the agent would always execute a similar forward arc regardless of its initial position, and also noticeable on MTC and FD, where the agent would sometimes move to the side of a desired block when it was shifted slightly. For BC, this issue was ameliorated by the use of translation and rotation augmentations, presumably because the policy could better handle small deviations from the motions seen at training time.

Colour and shape transfer Surprisingly, BC and GAIL both struggled with colour transfer to a greater degree than shape transfer on several tasks, as evidenced by the aggregated statistics for Colour and Shape variants in Table 1. Common failure modes included freezing in place or moving in the wrong direction when confronted with an object of a different colour to that seen at training time. In contrast, in most tasks where shape invariance was desirable (including MTC, MR, ML, and FC), the agent had no trouble reaching and manipulating blocks of different shapes. Although colour

---

[1] Multi-task imitation learning algorithms: `https://github.com/qxcv/mtil/`
[2] Benchmark suite and links to data: `https://github.com/qxcv/magical/`

jitter was one of the default augmentations, the BC ablations in Table 1 suggest that almost all of the advantage of augmentations comes from the use of translation/rotation augmentations. In particular, we did not find that colour jitter greatly improved performance on tasks where the optimal policy was colour-invariant. In spite of exposing the networks to a greater range of colours at train time, multitask training also failed to improve colour transfer, as we discuss below. Although translation and rotation sometimes improved colour transfer (e.g. for BC on FindDupe in Table 7), it is not clear why this was the case. We speculate that these augmentations could have encouraged the policy to acquire more robust early-layer features for edge and corner detection that did not rely on just one colour channel.

Multi-task transfer   Plain multi-task learning had mixed effects on generalisation. In some cases it improved generalisation (e.g. for BC on FC), but in most cases it led to unchanged or *negative* transfer, as in the Colour test variants for MTC, MR, and FD. This could have been because the policy was using colour to distinguish between tasks. More speculatively, it may be that a multi-task BC or GAIL loss is not the best way to incorporate off-task data, and that different kinds of multi-task pretraining are necessary (e.g. learning forward or inverse dynamics [9]).

Egocentric view and generalisation   The use of an allocentric (rather than egocentric) view did not improve generalisation or demo variant performance for most tasks, and sometimes decreased it. Table 1 shows the greatest performance drop on variants that change object position, such as Layout and Jitter. For example, in MTR we found that egocentric policies tended to rotate in one direction until the goal region was in the centre of the agent's field of view, then moved forward to reach the region, which generalises well to different goal region positions. In contrast, the allocentric policy would often spin in place or get stuck in a corner when confronted with a goal region in a different position. This supports the hypothesis of Hill et al. [22] that the egocentric view improves generalisation by creating positional invariances, and reinforces the value of being able to independently measure generalisation across distinct axes of variation (position, shape, colour, etc.).

5   Related work

There are few existing benchmarks that specifically examine robust IL. The most similar benchmarks to MAGICAL have appeared alongside evaluations of IRL and meta-IL algorithms. As noted in Section 1, several past papers employ "test" variants of standard Gym MuJoCo environments to evaluate IRL generalisation [17, 39, 32, 33], but these modified environments tend to have trivial reward functions (e.g. "run forward") and do not easily permit cross-environments transfer. Xu et al. [38] and Gleave and Habryka [18] use gridworld benchmarks to evaluate meta- and multi-task IRL, and both benchmarks draw a distinction between demonstration and execution environments within a meta-testing task. This distinction is similar in spirit to the demonstration/test variant split in MAGICAL, although MAGICAL differs in that it has more complex tasks and the ability to evaluate generalisation across different axes. We note that there also exist dedicated IL benchmarks [30, 26], but they are aimed at solving challenging robotics tasks rather than evaluating generalisation directly.

There are many machine learning benchmarks that evaluate generalisation outside of IL. For instance, there are several challenging benchmarks for generalisation [31, 12, 13] and meta- or multi-task learning [40] in RL. Unlike MAGICAL, these RL benchmarks have no ambiguity about what the goal is in the training environment, since it is directly specified via a reward function. Rather, the challenge is to ensure that the learnt policy (for model-free methods) can achieve that clearly-specified goal in different contexts (RL generalisation), or solve multiple tasks simultaneously (multi-task RL), or be adapted to new tasks with few rollouts (meta-RL). There are also several instruction-following benchmarks for evaluating generalisation in natural language understanding [29, 34]. Although these are not IL benchmarks, they are similar to MAGICAL in that they include train/test splits that systematically evaluate different aspects of generalisation. Finally, the Abstract Reasoning Corpus (ARC) is a benchmark that evaluates the ability of supervised learning algorithms to extrapolate geometric patterns in a human-like way [11]. Although there is no sequential decision-making aspect to ARC, Chollet [11] claims that solving the corpus may still require priors for "objectness", goal-directedness, and various geometric concepts, which means that methods suitable for solving MAGICAL may also be useful on ARC, and vice versa.

Although we covered some simple methods of improving IL robustness in Section 3, there also exist more sophisticated methods tailored to different IL settings. Meta-IL [15, 25] and meta-IRL [38, 39]

algorithms assume that a large body of demonstrations is available for some set of "train tasks", but only a few demonstrations are available for "test tasks" that might be encountered in the future. Each test task is assumed to have a distinct objective, but one that shares similarities with the train tasks, making it possible to transfer knowledge between the two. These methods are likely useful for multi-task learning in the context of MAGICAL, too. However, it's worth noting that past meta-IL work generally assumes that meta-train and meta-test settings are similar, whereas this work is concerned with how to generalise the intent behind a few demonstrations given in one setting (the demo variant) to other, potentially very different settings (the test variants). Similar comments apply to existing work on multi-task IL and IRL [18, 10, 14, 3].

6 Conclusion

In this paper, we introduced the MAGICAL benchmark suite, which is the first imitation learning benchmark capable of evaluating generalisation across distinct, semantically-meaningful axes of variation in the environment. Unsurprisingly, results for the MAGICAL suite confirm that single-task methods fail to transfer to changes in the colour, shape, position and number of objects. However, we also showed that image augmentations and perspective shifts only slightly ameliorate this problem, and multi-task training can sometimes make it *worse*. This lack of generalisation stands in marked contrast to human imitation: even 14-month-old infants have been observed to generalise demonstrations of object manipulation tasks across changes in object colour and shape, or in the appearance of the surrounding room [4]. Closing the gap between current IL capabilities and human-like few-shot imitation could require significant innovations in multi-task learning, action and state representations, or models of human cognition. The MAGICAL suite provides a way of evaluating such algorithms which not only tests whether they generalise well "on average", but also shines a light on the specific kinds of generalisation which they enable.

7 Broader impact

This paper presents a new benchmark for robust IL and argues for an increased focus on algorithms that can generalise demonstrator intent across different settings. We foresee several possible follow-on effects from improved IL robustness:

Economic effects of automation Better IL generalisation could allow for increased automation in some sectors of the economy. This has the positive flow-on effect of increased economic productivity, but could lead to socially disruptive job loss. Because our benchmark focuses on robust IL in robotics-like environments, it's likely that any effect on employment would be concentrated in sectors involving activities that are expensive to record. This could include tasks like surgery (where few demonstrators are qualified to perform the task, and privacy considerations make it difficult to collect data) or packaging retail goods for postage (where few-shot learning might be important when there are many different types of goods to handle).

Identity theft and model extraction More robust IL could enable better imitation of *specific* people, and not just imitation of people in general. This could lead to identify theft, for instance by mimicking somebody's speech or writing, or by fooling biometric systems. Because this benchmark focuses on control and manipulation rather than media synthesis, it's unlikely that algorithms designed to solve our benchmark will be immediately useful for this purpose. On the other hand, this concern is still relevant when applied to machine behaviour, rather than human behaviour. In NLP, it's known that weights for ML models can be "stolen" by observing the model's outputs for certain carefully chosen inputs [28]. Similarly, more robust IL could make it possible to clone a robot's policy by observing its behaviour, which could make it harder to sell robot control algorithms as standalone products.

Learnt objectives Hadfield-Menell et al. [21] argues that it is desirable for AI systems to infer their objectives from human behaviour, rather than taking them as fixed. This can avoid problems that arise when an agent (human, robot, or organisation) doggedly pursues an easy-to-measure but incorrect objective, such as a corporate executive optimising for quarterly profit (which is easy to measure) over long-term profitability (which is actually desired by shareholders). IL makes it possible to learn objectives from observed human behaviour, and

more robust IL may therefore lead to AI systems that better serve their designers' goals. However, it's worth noting that unlike, say, HAMDPs [16] or CIRL games [21], IL cannot request clarification from a demonstrator if the supplied demonstrations are ambiguous, which limits its ability to learn the right objective in general. Nevertheless, we hope that insights from improved IL algorithms will still be applicable to such interactive systems.

References

[1] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *ICML*, 2004.

[2] Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving Rubik's Cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

[3] Monica Babes, Vukosi Marivate, Kaushik Subramanian, and Michael L Littman. Apprenticeship learning about multiple intentions. In *ICML*, 2011.

[4] Sandra B Barnat, Pamela J Klein, and Andrew N Meltzoff. Deferred imitation across changes in context and object: Memory and generalization in 14-month-old infants. *Infant Behavior & Development*, 19(2):241, 1996.

[5] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[6] Aude Billard, Sylvain Calinon, Ruediger Dillmann, and Stefan Schaal. Survey: Robot programming by demonstration. In Bruno Siciliano and Oussama Khatib, editors, *Handbook of robotics*, chapter 59. Springer, 2008.

[7] Robert C Bolles and Martin A Fischler. A RANSAC-based approach to model fitting and its application to finding cylinders in range data. In *IJCAI*, volume 1981, pages 637–643, 1981.

[8] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. *arXiv:1606.01540*, 2016.

[9] Daniel S Brown, Russell Coleman, Ravi Srinivasan, and Scott Niekum. Safe imitation learning via fast bayesian reward inference from preferences. *arXiv:2002.09089*, 2020.

[10] Jaedeug Choi and Kee-Eung Kim. Nonparametric bayesian inverse reinforcement learning for multiple reward functions. In *NIPS*, pages 305–313, 2012.

[11] François Chollet. The measure of intelligence. *arXiv preprint arXiv:1911.01547*, 2019.

[12] Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. *arXiv:1812.02341*, 2018.

[13] Karl Cobbe, Christopher Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. *arXiv:1912.01588*, 2019.

[14] Christos Dimitrakakis and Constantin A Rothkopf. Bayesian multitask inverse reinforcement learning. In *European Workshop on Reinforcement Learning*. Springer, 2011.

[15] Yan Duan, Marcin Andrychowicz, Bradly Stadie, OpenAI Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. One-shot imitation learning. In *NeurIPS*, pages 1087–1098, 2017.

[16] Alan Fern, Sriraam Natarajan, Kshitij Judah, and Prasad Tadepalli. A decision-theoretic model of assistance. In *IJCAI*, 2007.

[17] Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arXiv:1710.11248*, 2017.

[18] Adam Gleave and Oliver Habryka. Multi-task maximum entropy inverse reinforcement learning. *arXiv preprint arXiv:1805.08882*, 2018.

[19] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NIPS*, 2014.

[20] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of Wasserstein GANs. In *NIPS*, 2017.

[21] Dylan Hadfield-Menell, Stuart J Russell, Pieter Abbeel, and Anca Dragan. Cooperative inverse reinforcement learning. In *NIPS*, pages 3909–3917, 2016.

[22] Felix Hill, Andrew K Lampinen, Rosalia Schneider, Stephen Clark, Matthew Botvinick, James L McClelland, and Adam Santoro. Environmental drivers of systematicity and generalization in a situated agent. In *ICLR*, 2020.

[23] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *NIPS*, 2016.

[24] Jonathan Ho, Jayesh Gupta, and Stefano Ermon. Model-free imitation learning with policy optimization. In *ICML*, 2016.

[25] Stephen James, Michael Bloesch, and Andrew J Davison. Task-embedded control networks for few-shot imitation learning. *CORL*, 2018.

[26] Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J Davison. Rlbench: The robot learning benchmark & learning environment. *arXiv preprint arXiv:1909.12271*, 2019.

[27] Ilya Kostrikov, Kumar Krishna Agrawal, Debidatta Dwibedi, Sergey Levine, and Jonathan Tompson. Discriminator-actor-critic: Addressing sample inefficiency and reward bias in adversarial imitation learning. In *ICLR*, 2019.

[28] Kalpesh Krishna, Gaurav Singh Tomar, Ankur P Parikh, Nicolas Papernot, and Mohit Iyyer. Thieves on Sesame Street! Model extraction of BERT-based APIs. *ICLR*, 2020.

[29] Brenden M Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. *ICML*, 2018.

[30] Raphael Memmesheimer, Ivanna Mykhalchyshyna, Viktor Seib, and Dietrich Paulus. Simitate: A hybrid imitation learning benchmark. *arXiv preprint arXiv:1905.06002*, 2019.

[31] Alex Nichol, Vicki Pfau, Christopher Hesse, Oleg Klimov, and John Schulman. Gotta learn fast: A new benchmark for generalization in RL. *arXiv:1804.03720*, 2018.

[32] Xue Bin Peng, Angjoo Kanazawa, Sam Toyer, Pieter Abbeel, and Sergey Levine. Variational discriminator bottleneck: Improving imitation learning, inverse RL, and GANs by constraining information flow. *arXiv preprint arXiv:1810.00821*, 2018.

[33] Ahmed H Qureshi, Byron Boots, and Michael C Yip. Adversarial imitation via variational inverse reinforcement learning. *ICLR*, 2019.

[34] Laura Ruis, Jacob Andreas, Marco Baroni, Diane Bouchacourt, and Brenden M Lake. A benchmark for systematic generalization in grounded language understanding. *arXiv:2003.05161*, 2020.

[35] Iude W Shavlik. Acquiring recursive and iterative concepts with explanation-based learning. *Machine Learning*, 1990.

[36] Adam Stooke and Pieter Abbeel. rlpyt: A research code base for deep reinforcement learning in pytorch. *arXiv preprint arXiv:1909.01500*, 2019.

[37] Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. In *IJCAI*, 2018.

[38] Kelvin Xu, Ellis Ratner, Anca Dragan, Sergey Levine, and Chelsea Finn. Learning a prior over intent via meta-inverse reinforcement learning. *ICML*, 2019.

[39] Lantao Yu, Tianhe Yu, Chelsea Finn, and Stefano Ermon. Meta-inverse reinforcement learning with probabilistic context variables. In *NeurIPS*, pages 11749–11760, 2019.

[40] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-World: A benchmark and evaluation for multi-task and meta reinforcement learning. In *CoRL*, 2020.

[41] Konrad Zolna, Scott Reed, Alexander Novikov, Sergio Gomez Colmenarej, David Budden, Serkan Cabi, Misha Denil, Nando de Freitas, and Ziyu Wang. Task-relevant adversarial imitation learning. *arXiv preprint arXiv:1910.01077*, 2019.

A Additional benchmark details

In this section we provide more details about our benchmark tasks, including horizons, scoring functions, and so on. We also list the test variants available for each task in Table 2.

Task	Test variant						
	Jitter	Layout	Colour	Shape	CountPlus	Dynamics	All
MoveToCorner	✓	✗	✓	✓	✗	✓	✓
MoveToRegion	✓	✓	✓	✗	✗	✓	✓
MatchRegions	✓	✓	✓	✓	✓	✓	✓
MakeLine	✓	✓	✓	✓	✓	✓	✓
FindDupe	✓	✓	✓	✓	✓	✓	✓
FixColour	✓	✓	✓	✓	✓	✓	✓
ClusterColour	✓	✓	✓	✓	✓	✓	✓
ClusterType	✓	✓	✓	✓	✓	✓	✓

Table 2: Available variants for each task. Some variants are not defined for certain tasks because they may make task completion impossible, make task completion trivial (i.e. the null policy often completes the task), or do not provide a meaningful axis of variation (e.g. MoveToRegion does not feature any blocks, and so there are no shapes to randomise).
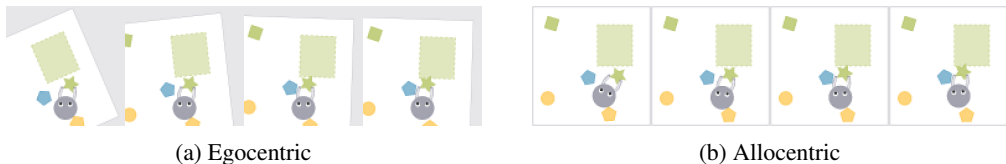
A.1 Action and observation space

(a) Egocentric   (b) Allocentric

Figure 3: Egocentric and allocentric views of a demonstration on MoveToRegion. The four 96×96 RGB frames shown in each subfigure would normally be stacked together along the channels axis before being passed to an agent policy or discriminator.

We use the same discrete action space for all tasks. Although this benchmark was inspired by robotic IL, where the underlying action space is generally continuous, we opted to use discrete actions so that we could elicit human demonstrations using only a standard keyboard. The underlying state space is still continuous, so each discrete action applies a preset combination of forces to the robot, such as a force that pushes the gripper arms together, or a force that moves the robot forward or backward. In total, the agent has 18 distinct actions. These are formed from the Cartesian product of two gripper actions (push closed/allow to open), three longitudinal motion actions (forward/back/stop), and three angular motions (left/straight/right).

We use the same image-based observation space for each task. In all of our experiments, we provide the agent with stacked 96×96 pixel RGB frames depicting the workspace at the current time step and three preceding time steps. At our 8Hz control rate, this corresponds to around 0.5s of interaction context. Using an image-based observation space makes it easy to generalise policies and discriminators across different numbers and types of objects, without having to resort to, e.g., graph networks or structured learning. An image-based observation space also means that the agent gets access to a similar representation as the human demonstrator. This makes it possible to resolve ambiguities and improve generalisation by exploiting features of the human visual system, as we do when we apply the small image augmentations described in Appendix B.

By default, observations employ an egocentric (robot-centred) perspective on the workspace, as illustrated in Fig. 3a. Unlike the allocentric perspective, depicted in Fig. 3b, the egocentric often does not allow the agent to observe the full workspace. However, we found that an egocentric perspective resulted in faster training and better generalisation, as we note in the ablations of Section 4. Similar benefits to generalisation were previously observed by Hill et al. [22].

A.2 Detailed task descriptions
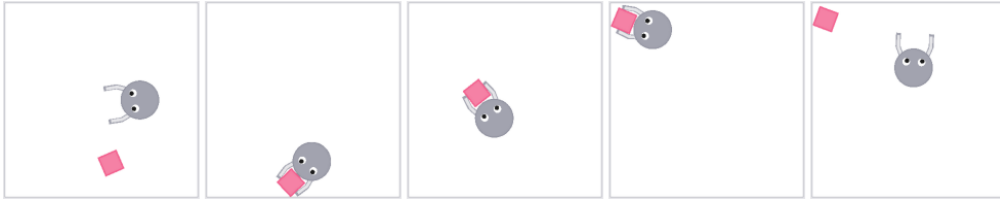
A.2.1 MoveToCorner (MTC)

Figure 4: A demonstration on MoveToCorner.

In MoveToCorner, the robot must push a single block from the bottom right corner of the workspace to the top left corner of the workspace. Test variants are also constrained so that there is only ever one block, and it always starts close to the bottom right corner of the workspace. These constraints preclude use of the CountPlus test variant, since block count cannot be changed without making the task ambiguous. It also precludes use of the Layout variant, since fully randomising block position might make the desired block location ambiguous (e.g. pushing the block into top left corner versus pushing it to the opposite side of the workspace). The horizon for all variants is $H = 80$ time steps.

Trajectories receive a score of $S(\tau) = 1$ if the block spends the last frame of the rollout within $\sqrt{2}/2$ units of the top left corner of the workspace (the whole workspace is 2×2 units). $S(\tau)$ decays linearly from 1 to 0 as the block moves from inside that region to more than $\sqrt{2}$ units away from the corner.

A.2.2 MoveToRegion (MTR)
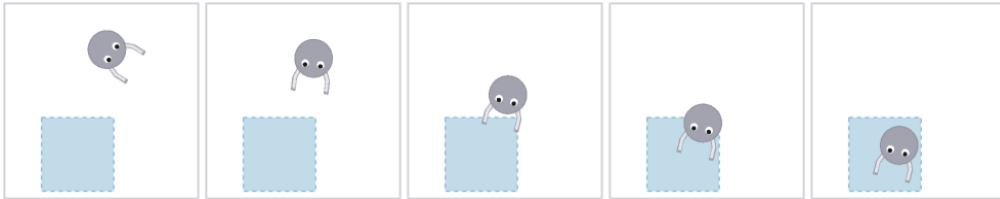
Figure 5: A demonstration on MoveToRegion.

The objective of the MoveToRegion task is for the robot to drive inside a goal region placed in the workspace. There is only ever one goal region, and no blocks are present in the train or test variants. Hence the CountPlus and Shape variants are not applicable. However, the Colour variant is still applicable. as it randomises the colour of the goal region. The horizon is set to $H = 40$.

Scoring for MoveToRegion is binary. If at the end of the episode, the centre of the robot's body is inside the goal region, then it receives a score of 1. Otherwise it receives a score of 0.

A.2.3 MatchRegions (MR)
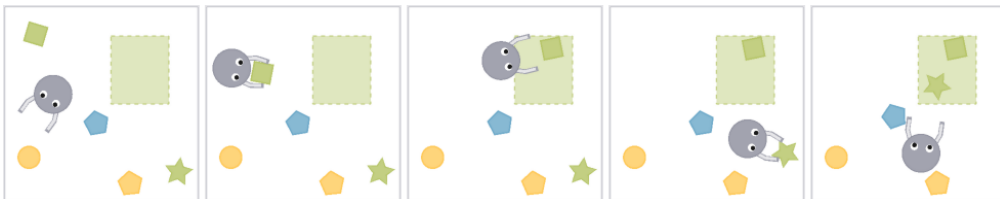
Figure 6: A demonstration on MatchRegions.

In MatchRegions, the agent is confronted with a single goal region and several blocks of different colours. The objective is to move all (and only) blocks of the same colour as the goal region into the goal region. All test variants are applicable to this version, although CountPlus only randomises the number of blocks (and not the number of goal regions) in order to avoid ambiguity about which goal region(s) the robot should fill with blocks. The horizon is fixed to $H = 120$.

At the end of a trajectory τ, the robot receives a score of

$$S(\tau) = \underbrace{\frac{|\mathcal{T} \cap \mathcal{R}|}{|\mathcal{T}|}}_{\text{Target bonus}} \times \underbrace{\left(1 - \frac{|\mathcal{D} \cap \mathcal{R}|}{|\mathcal{R}|}\right)}_{\text{Distractor penalty}} .$$

Here $\mathcal{T}$ is the set of *target blocks* of the same colour as the goal region, $\mathcal{D}$ is the set of *distractor blocks* of a different colour, and $\mathcal{R}$ is the set of blocks inside the goal region in the last state s_T of the rollout τ. The agent gets a perfect score of 1 for placing all the target blocks and none of the distractors in the goal region. Its score decreases for each target block it fails to move to the goal region (target bonus) and each distractor block it improperly places in the goal region (distractor penalty).

A.2.4 MakeLine (ML)

Figure 7: A demonstration on MakeLine.

The objective of the MakeLine task is to arrange all of the blocks in the workspace into a line. The orientation and location of the line are ignored, as are the shapes and colours of the blocks involved. The horizon for this task is $H = 180$.

Scoring for MakeLine is a function of the relative positions of blocks in the final state of a trajectory, and in particular the number of blocks that form the largest identifiable "line". To identify lines of blocks, we use a line-fitting methods that is similar in spirit to RANSAC [7], but with constraints to ensure that blocks are spread out along the length of the line rather than "bunching up". Our definition of what constitutes a line is based on a relation between triples of blocks: we say that a block b_k is considered to be part of a line between blocks b_i and b_j if:

1. b_k **is an inlier:** it must lie a distance of at most $d_i = 0.18$ units from the (geometric) line that links b_i and b_j (recall that the workspace is 2×2 units).

2. b_k **is close to other blocks in the line:** if b_k is not the first or last block in the line of blocks, then it must be a distance of at most $d_c = 0.42$ units from the previous and next blocks. Here the distance is measured along the direction of the geometric line between b_i and b_j. That is, by projecting the previous and next inliers onto the geometric line between b_i and b_j, then taking the distance between those projections and the projection for b_k.

Note that if b_i and b_j are a long way apart, then there may be several subsets of inliers for the line between b_i and b_j, each of which is separated from the other subsets than d_c units. For any given pair of blocks (b_i, b_j), let $\#(b_i, b_j)$ be the number of blocks that form the *largest* such subset for the line between b_i and b_j (potentially including b_i and/or b_j, if they are close enough to the other inliers). Further, let n be the number of blocks in the workspace, and $m = \max_{i,j} \#(b_i, b_j)$ be the largest number of blocks on a line between any two blocks in the final state. If $m = n$, then all blocks belong to the same line, and so $S(\tau) = 1$. If $m = n - 1$, then exactly one block is not a part of the largest identifiable line, and $S(\tau) = 0.5$. Otherwise, if $m < n - 1$, the agent receives a score of $S(\tau) = 0$.
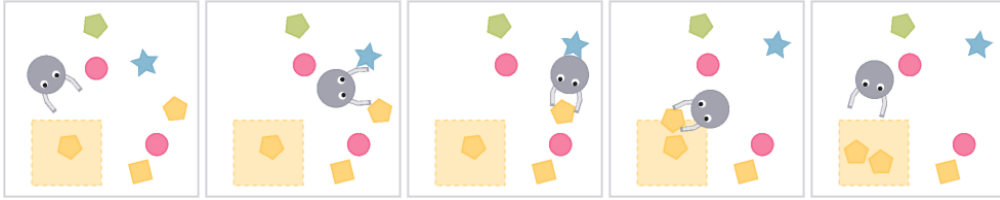
15

A.2.5 FindDupe (FD)

Figure 8: A demonstration on FindDupe.

FindDupe presents the agent with a goal region that has a single "query" block inside it, along with a mixture of blocks outside the goal region. The agent's objective is to locate at least one block outside the goal region with the same shape and colour as the query block, and push it inside the goal region. Variants are constrained so that there is only ever one goal region and query block, and so that there is at least one duplicate of the query block outside the goal region. The horizon for this task is $H = 100$.

The score for this task is a function of the set of blocks present in the goal area at the end of the trajectory. Let $\mathcal{R}$ denote the set of blocks inside the region at the end of the episode, let $\mathcal{T}$ denote the set of all *target* blocks with the same shape and colour as the query block, and let $\mathcal{D}$ denote the set of all *distractor* blocks with a different shape or colour. Further, let q refer to the original query block. The score $S(\tau)$ for a trajectory is

$$S(\tau) = \underbrace{\mathbb{I}[q \in \mathcal{R}] \times \mathbb{I}[\mathcal{T} \cap \mathcal{R} \neq \varnothing]}_{\text{Query satisfied?}} \times \underbrace{\left(1 - \frac{|\mathcal{D} \cap \mathcal{R}|}{|\mathcal{R}|}\right)}_{\text{Distractor penalty}} .$$

The first factor ensures that the query block remains inside the goal region. The second factor ensures that at least one other block with the same attributes as the query block is in the goal region. Finally, the last factor creates a penalty for pushing distractor blocks into the goal region.

A.2.6 FixColour (FC)
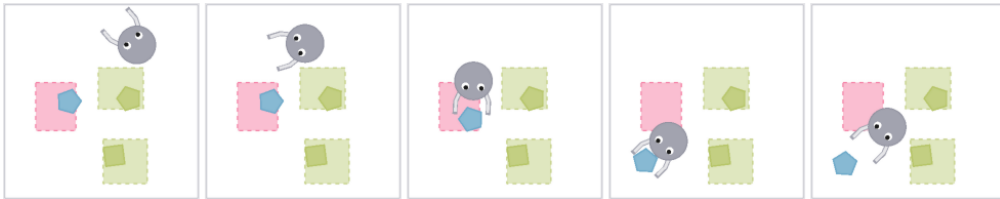
Figure 9: A demonstration on FixColour.

FixColour variants always include several non-overlapping goal regions, each containing a single block. Exactly one of those blocks will be of a different colour to its enclosing goal region; we'll call this the "mismatched block". The agent's objective is to identify the mismatched block and push it out of its goal region, into an unoccupied part of the workspace, thereby "fixing" the mismatch. The horizon for this task is $H = 60$.

Scoring for FixColour is binary. A score of $S(\tau) = 1$ is given if, in the final state, the mismatched block is not in its original goal region. All other goal regions must contain exactly the same block that they started with (and in particular cannot contain the mismatched block). If any of these conditions is not satisfied, then the score is zero.

A.2.7 ClusterColour (CC) and ClusterShape (CS)
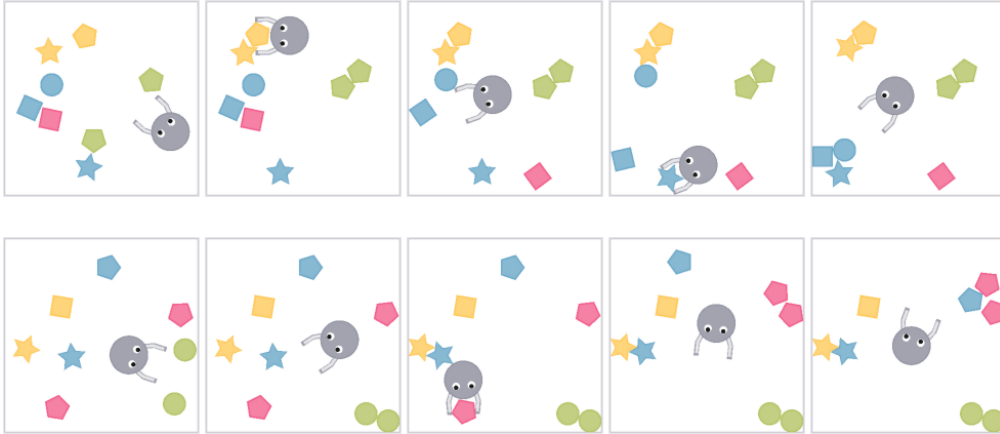
Figure 10: Demonstrations on ClusterColour (top) and ClusterShape (bottom).

In both ClusterColour and ClusterShape, the workspace is initially filled with a jumble of blocks of different colours and types, and the agent must push the blocks into clusters according to some attribute. For ClusterColour, blocks should belong to the same cluster iff they have the same colour, while ClusterShape applies the analogous criterion to block shape. All variants are applicable to these tasks. Because these tasks require interaction with most or all blocks in the workspace, the horizon is set to $H = 320$ (40s at 8Hz).

The score $S(\tau)$ takes the same form for both ClusterColour and ClusterShape, but with a different attribute-of-interest (either colour or shape). Specifically, $S(\tau)$ is computed by applying a K-means-like objective to the final state s_T of the rollout τ. For each value a of the attribute-of-interest (either red/green/blue/yellow for ClusterColour or square/circle/pentagon/star for ClusterShape), a *centroid* x_a is computed from the mean positions of blocks with the corresponding attribute value. Formally, this is

$$x_a = \frac{1}{|\mathcal{B}_a|} \sum_{b \in \mathcal{B}_a} b.\text{pos} \ ,$$

where $\mathcal{B}_a$ is the set of blocks with the relevant attribute set to value a, and b.pos is the position of block b in state s_T. In order for an individual block b with relevant attribute value a to be considered correctly clustered, the squared distance

$$d(b, a) = \|b.\text{pos} - x_a\|_2^2$$

between it and its associated centroid must be at most a third the squared distance $d(b, a')$ between it and the nearest centroid for any other attribute value a'. Specifically, we must have

$$d(b, a) < \frac{1}{3} \min_{a' \neq a} d(b, a') \ .$$

When 50% or fewer of blocks are correctly clustered in the final state of a trajectory, the score $S(\tau) = 0$. As the fraction of correctly clustered blocks increases from 50% up to 100%, the score $S(\tau)$ increases linearly from 0 to 1.

B Addition experiment details

This section documents the full set of hyperparameters we used for BC and GAIL, along with additional details on how we collected and preprocessed our demonstrations.

Dataset and data preprocessing details   We collected training datasets of 25 demonstration trajectories for the demonstration variant of each task. These trajectories were recorded by the authors to show several distinct strategies for solving the task within the demonstration variant. For instance,

in ClusterColour, there are demonstrations that place clusters in different locations or construct them in a different order. Appendix A shows a single demonstration for each task in the dataset.

Each algorithm run used only 10 of the 25 total trajectories for each task (or 10 trajectories for each task, in the multi-task case). The subset of 10 trajectories was sampled at random based on the seed for that run. We did not hold out any trajectories for testing or validation; rather, our evaluation is based on the test variant scores assigned to the trained policy produced by each algorithm. For all policies, value functions, and discriminators, we constructed an observation by concatenating four temporally adjacent RGB frames along the channels axis, scaling the pixel values into the $[0, 1]$ range, and resizing the stacked frames to 96×96 pixels. For BC, we performed the additional preprocessing step of removing samples with noop actions from the demonstration dataset, as described below.

Evaluation details   For single-task BC and GAIL, we do five training runs on each task with different random seeds. After each run, we take the trained policy, use it to perform 100 rollouts on each test variant of the original task, and retain the mean scores from those 100 trajectories. In tables, we report "mean score $\pm$ standard deviation of score", where the mean and standard deviation are taken over the mean evaluation scores for each of the five runs on each algorithm and task. Multitask evaluations are similar, except we pool data from all tasks together, and consequently only perform five runs in total rather than five runs per task. To reduce variance, we used the same five random seeds (and consequently the same five subsets of 10 training trajectories each) for all algorithms and tasks.

Default augmentation set   Throughout the text, we refer to noise, translation, rotation, and colour jitter augmentations. Concretely, these augmentations involved the following operations:

- **Noise:** Each (RGB) channel of each pixel is independently perturbed by additive noise sampled from $\mathcal{N}(0, 0.01)$.

- **Translation:** The image is mirror-padded and randomly translated along the x and y axes by up to 5% of their respective range (so ± 4.8px, for 96×96 pixels).

- **Rotation:** Image is mirror padded and then rotated around its centre by up to ± 5 degrees.

- **Colour jitter:** For this augmentation, images are translated to the CIELab colour space. The luminance channel is rescaled by a randomly sampled factor between 0.99 and 1.01, while the a and b channels are treated as a 2D vectors and randomly rotated by up to ± 0.15 radians. We use the same luminance scaling factor and colour rotation for each pixel an a given image. After these operations, images are converted back to RGB.

For the translation, rotation, and colour jitter augmentations, we apply the same randomly sampled transformation to each image in a four-image "stack" of frames, but different, independently sampled transformations to each stack in a training batch.

Single- and multi-task BC hyperparameters   The hyperparameters for BC are given in Table 3. BC hyperparameters were manually tuned to ensure that losses plateaued on most single-task problems. Note that hyperparameters for single- and multi-task learning were identical. In particular, we retained the same batch size for multi-task experiments, and randomly sampled demonstration states from each task with a weighting that ensured equal representation from all tasks. Initially, we found that training BC to convergence would cause the policy to get "stuck" in states where the most probable demonstrator action was a noop action. We avoided this problem by removing all state/action pairs with noop actions from the dataset in our BC experiments; we did not do this in our GAIL experiments.

Single- and multi-task GAIL hyperparameters   Hyperparameters for GAIL are listed in Table 4. For policy optimisation, we used the PPO implementation from rlpyt [36]; PPO hyperparameters that are not listed in Table 4 took their default values in rlpyt. To prevent value and advantage magnitudes from exploding in PPO, we normalised rewards produced by the discriminator to have zero mean and a standard deviation of 0.1, both enforced using a running average and variance updated over the course of training. Again, multi-task hyperparameters were the same as single-task hyperparameters, and we split each policy and discriminator training batch evenly between the tasks.

Hyperparameter	Value	Range Considered
Total opt. batches	20,000	5,000–20,000
Batch size	32	-
SGD learning rate	10^{-3}	-
SGD momentum	0.1	-
Policy augmentations	Noise, trans., rot., colour jit.	-

Table 3: Hyperarameters for BC experiments.

Hyperparameter	Value	Range Considered
Policy (PPO)		
Sampler batch size	32	16 to 64
Sampler time steps	8	8 to 20
Opt. epochs per update	12	2 to 10
Opt. minibatch size	64	42 to 64
Initial Adam step size	6×10^{-5}	10^{-6} to 10^{-3}
Final Adam step size	0 (lin. anneal)	-
Discount γ	0.8	0.8 to 1.0
GAE λ	0.8	0.8 to 1.0
Entropy bonus	10^{-5}	10^{-6} to 10^{-4}
Advantage clip ϵ	0.01	0.01 to 0.2
Grad. clip ℓ_2 norm	1.0	-
Augmentations	N/A	-
Discriminator		
Batch size	24	-
Adam step size	2.5×10^{-5}	10^{-5} to 5×10^{-4}
Augmentations	Noise, trans., rot., colour jit.	-
$\lambda_{\text{w-gp}}$ (WGAIL-GP)	100	-
Misc.		
Disc. steps per PPO update	12	8 to 32
Total env. steps of training	10^6	5×10^5 to 5×10^6
Reward norm. std. dev.	0.1	-

Table 4: Hyperarameters for GAIL experiments.

Apprenticeship learning baseline In addition to our BC and (W)GAIL baselines, we also attempted to train a feature expectation matching Apprenticeship Learning (AL) baseline [1, 24]. Given a feature function $\Phi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^n$, the goal of AL is to find a policy π_θ that matches the expected value of the feature function Φ under the demonstration distribution with its expected value under the novice distribution. That is, we seek a π_θ such that $\mathbb{E}_{\pi_\theta} \Phi(s, a) = \mathbb{E}_{\mathcal{D}} \Phi(s, a)$. Matching feature expectations is equivalent to finding a policy π_θ that drives the cost

$$\sup_{\|w\| \leq 2} \left[\mathbb{E}_{\mathcal{D}} w^T \Phi(s, a) - \mathbb{E}_{\pi_\theta} w^T \Phi(s, a) \right] \tag{1}$$

to zero. Observe that if w^* is a weight vector that attains the supremum in Eq. (1), then

$$-\nabla_\theta \mathbb{E}_{\pi_\theta} w^{*T} \Phi(s, a)$$

is a subgradient of Eq. (1) with respect to the policy parameters θ. Thus, our training procedure consisted of alternating between optimising Eq. (1) to convergence with respect to w, and taking a PPO step on the policy parameters using the reward function $r(s, a) = w^{*T} \Phi(s, a)$ (recall that RL maximises return, but we want to minimise Eq. (1)). To optimise w, we used 512 samples from the expert and the novice, and to optimise π_θ, we used the same generator hyperparameters as our GAIL runs. This single-task AL baseline is denoted "AL (ST)" in results tables.
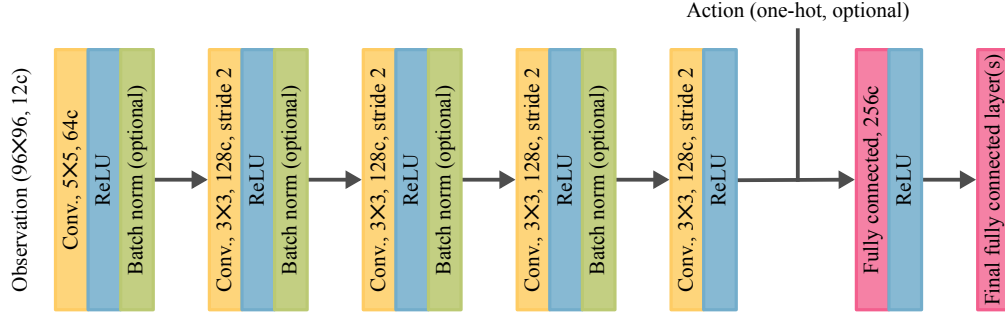
Figure 11: Base architecture for policies, value functions and discriminators. "nc" is used as an abbreviation for "n channels". Refer to main text for a discussion of which networks use the optional features (batch norm, action input), and for a description of the final layer for each network type.

The feature function Φ used for AL was acquired by removing the final (logit) layer of our GAIL discriminator network architecture and optimsing the remaining layers to minimise an autoencoder loss. In creating the encoder, our only modification to the GAIL discriminator network architecture was to replace the 256-dimensional penultimate layer with a 32-dimensional one, to produce a 32-dimensional feature function Φ. This optimisation was performed for 8,192 size-24 batches of expert data, which we empirically found was enough to get clear reproduction of most input images. After autoencoder pretraining, the encoder weights were kept frozen for the remainder of each training run.

Unfortunately, we could not get AL to produce adequate policies for any task except MoveToCorner. We suspect that the poor performance of AL was due to inadequate autoencoder features. The autoencoder was only trained on expert samples, and we found that for some problems it would not correctly reproduce images of states that were far from the support of the demonstrations. It may be possible to improve results by training the autoencoder on both random rollouts and expert samples, or by training it on more diverse multi-task data.

Network architecture   Fig. 11 shows the base architecture for all neural networks used in the experiments (including discriminators, policies, and value functions). Some experiments use slight variations on this basic policy architecture for some of the networks:

- The one-hot action input is only used for discriminators, which concatenate the one-hot action representation to the activations of the final convolution layer before performing a forward pass through the linear layers.

- Batch norm is only used for the BC policy and GAIL discriminator, not for the GAIL policy and value function.

- In GAIL experiments, which train a policy via RL, the policy and value function share all layers *except* the final fully-connected layer.

- In multitask experiments, the policy, value function, and discriminator share weights between tasks for all layers except the last. The final layer uses a single, separate set of weights corresponding to each task.

Computing infrastructure and experiment running time   Experiments were performed on machines with 2$\times$ Xeon Gold 6130 CPUs (16 cores each, 2.1GHz base clock), 128–256GB RAM, and 4$\times$ GTX 1080-Ti GPUs. Each "run"—that is, the training and evaluation of a specific algorithm on a specific task with one seed—took an average of 10h03m (GAIL) and 32m (BC). It should be noted that these wall time figures were recorded while performing up to 16 runs in parallel on each machine. Because we did not use task-specific training durations, there was little variance in execution time between the different configurations (multi-task, egocentric, allocentric, etc.) of each of the two main base algorithms (BC and GAIL).

C Full experiment results

Full results for all methods, along with corresponding ablations, are shown in Table 5, Table 6, Table 7 and Table 8. We abbreviate behavioural cloning as "BC" and generative adversarial IL as "GAIL", while apprenticeship learning is "AL". Single-task methods are denoted with "(ST)" and multi-task methods with "(MT)". "Allo." is for experiments using an allocentric view; all other expeirments use an egocentric view. For GAIL, "WGAIL-GP" denotes a version of GAIL that approximately minimises Wasserstein divergence while using a gradient penalty to encourage 1-Lipschitzness of the discriminator. For augmentation ablations, we use "no trans./rot. aug." to denote removal of translation/rotation; and "no aug." to denote removal of all three default augmentations (colour, translation/rotation, Gaussian noise).

	MoveToCorner							
	Demo	Jitter	Layout	Colour	Shape	CountPlus	Dynamics	All
AL (ST)	0.00±0.00	0.00±0.00	-	0.00±0.00	0.00±0.00	-	0.00±0.00	0.00±0.00
BC (MT)	0.97±0.04	0.91±0.02	-	0.73±0.16	0.98±0.01	-	0.92±0.04	0.62±0.11
BC (ST)	0.98±0.04	0.86±0.07	-	0.96±0.05	0.97±0.03	-	0.91±0.05	0.84±0.06
BC (ST, allo.)	0.94±0.05	0.89±0.04	-	0.93±0.04	0.97±0.02	-	0.90±0.04	0.91±0.02
BC (ST, no aug.)	0.96±0.04	0.77±0.09	-	0.80±0.06	0.81±0.12	-	0.86±0.05	0.60±0.05
BC (ST, no trans./rot. aug.)	0.96±0.04	0.85±0.04	-	0.83±0.14	0.88±0.06	-	0.94±0.05	0.67±0.11
GAIL (MT)	0.31±0.31	0.33±0.31	-	0.16±0.09	0.34±0.30	-	0.30±0.27	0.16±0.10
GAIL (ST)	0.99±0.01	0.91±0.06	-	0.78±0.10	0.95±0.03	-	0.95±0.05	0.65±0.16
GAIL (ST, allo.)	1.00±0.00	0.82±0.05	-	0.90±0.08	0.99±0.01	-	0.99±0.01	0.59±0.12
GAIL (ST, no aug.)	0.56±0.36	0.36±0.23	-	0.39±0.30	0.34±0.24	-	0.46±0.27	0.11±0.10
WGAIL-GP (ST)	0.35±0.24	0.22±0.12	-	0.17±0.20	0.32±0.21	-	0.30±0.21	0.04±0.05

	MoveToRegion							
	Demo	Jitter	Layout	Colour	Shape	CountPlus	Dynamics	All
AL (ST)	0.51±0.42	0.47±0.39	0.22±0.17	0.21±0.22	-	-	0.51±0.41	0.09±0.05
BC (MT)	0.79±0.22	0.77±0.26	0.52±0.13	0.60±0.16	-	-	0.81±0.24	0.26±0.04
BC (ST)	0.89±0.11	0.88±0.11	0.48±0.12	0.60±0.13	-	-	0.88±0.10	0.28±0.06
BC (ST, allo.)	0.63±0.08	0.57±0.14	0.09±0.02	0.56±0.19	-	-	0.61±0.12	0.10±0.02
BC (ST, no aug.)	0.88±0.12	0.83±0.11	0.44±0.08	0.75±0.17	-	-	0.87±0.12	0.36±0.07
BC (ST, no trans./rot. aug.)	0.91±0.05	0.85±0.10	0.44±0.10	0.73±0.13	-	-	0.89±0.06	0.33±0.05
GAIL (MT)	1.00±0.00	0.99±0.02	0.69±0.20	0.34±0.09	-	-	1.00±0.00	0.31±0.07
GAIL (ST)	1.00±0.00	1.00±0.00	0.71±0.09	0.40±0.07	-	-	1.00±0.00	0.29±0.07
GAIL (ST, allo.)	1.00±0.00	0.98±0.02	0.08±0.02	0.95±0.02	-	-	1.00±0.00	0.10±0.03
GAIL (ST, no aug.)	0.99±0.03	0.79±0.08	0.34±0.10	0.56±0.12	-	-	0.93±0.06	0.20±0.06
WGAIL-GP (ST)	0.94±0.03	0.87±0.04	0.60±0.06	0.23±0.01	-	-	0.94±0.03	0.18±0.04

Table 5: Scores for all compared methods on two tasks, reported as "mean (std.)" over five training runs (individual run means were computed with 100 rollouts each). A colour scale ( ) grades mean scores from poor (lightest) to perfect (darkest). See main text in Appendix C for abbreviations.

| | **MatchRegions** | | | | | | | |
	Demo	Jitter	Layout	Colour	Shape	CountPlus	Dynamics	All
AL (ST)	0.00±0.00	0.00±0.00	0.01±0.01	0.00±0.00	0.00±0.00	0.01±0.01	0.00±0.00	0.01±0.01
BC (MT)	0.69±0.10	0.60±0.09	0.05±0.01	0.32±0.06	0.65±0.07	0.04±0.02	0.64±0.09	0.04±0.01
BC (ST)	0.77±0.09	0.69±0.11	0.07±0.02	0.42±0.04	0.69±0.10	0.07±0.03	0.68±0.08	0.05±0.02
BC (ST, allo.)	0.72±0.06	0.58±0.14	0.01±0.01	0.58±0.11	0.60±0.09	0.02±0.01	0.56±0.10	0.03±0.02
BC (ST, no aug.)	0.71±0.05	0.49±0.07	0.04±0.01	0.35±0.05	0.59±0.05	0.05±0.02	0.54±0.05	0.04±0.02
BC (ST, no trans./rot. aug.)	0.75±0.05	0.54±0.05	0.06±0.01	0.34±0.03	0.62±0.07	0.06±0.01	0.63±0.02	0.05±0.02
GAIL (MT)	0.19±0.12	0.20±0.10	0.05±0.02	0.07±0.03	0.19±0.12	0.02±0.01	0.20±0.11	0.04±0.01
GAIL (ST)	0.94±0.03	0.92±0.03	0.21±0.02	0.31±0.10	0.93±0.05	0.14±0.04	0.92±0.04	0.14±0.04
GAIL (ST, allo.)	0.64±0.13	0.58±0.11	0.01±0.01	0.36±0.08	0.62±0.11	0.01±0.01	0.57±0.11	0.02±0.02
GAIL (ST, no aug.)	0.44±0.24	0.35±0.18	0.04±0.03	0.18±0.13	0.35±0.20	0.03±0.02	0.36±0.19	0.02±0.01
WGAIL-GP (ST)	0.32±0.05	0.30±0.04	0.15±0.04	0.07±0.00	0.32±0.05	0.11±0.03	0.28±0.03	0.08±0.02

| | **MakeLine** | | | | | | | |
	Demo	Jitter	Layout	Colour	Shape	CountPlus	Dynamics	All
AL (ST)	0.00±0.00	0.00±0.00	0.04±0.01	0.00±0.00	0.00±0.00	0.03±0.01	0.00±0.00	0.03±0.01
BC (MT)	0.31±0.07	0.29±0.02	0.18±0.03	0.18±0.03	0.30±0.05	0.16±0.02	0.28±0.07	0.14±0.02
BC (ST)	0.48±0.08	0.43±0.07	0.20±0.04	0.32±0.04	0.41±0.05	0.18±0.05	0.42±0.07	0.18±0.04
BC (ST, allo.)	0.25±0.07	0.24±0.04	0.04±0.01	0.11±0.02	0.21±0.07	0.03±0.01	0.19±0.03	0.02±0.01
BC (ST, no aug.)	0.19±0.03	0.12±0.06	0.11±0.03	0.11±0.02	0.14±0.03	0.09±0.02	0.11±0.05	0.09±0.02
BC (ST, no trans./rot. aug.)	0.23±0.05	0.17±0.06	0.12±0.04	0.13±0.02	0.15±0.02	0.12±0.06	0.14±0.04	0.11±0.03
GAIL (MT)	0.02±0.01	0.01±0.01	0.06±0.02	0.02±0.00	0.02±0.01	0.05±0.02	0.02±0.01	0.06±0.03
GAIL (ST)	0.33±0.19	0.33±0.21	0.20±0.02	0.19±0.07	0.27±0.16	0.17±0.05	0.28±0.15	0.17±0.04
GAIL (ST, allo.)	0.01±0.01	0.01±0.01	0.05±0.02	0.01±0.01	0.01±0.01	0.04±0.01	0.01±0.01	0.03±0.01
GAIL (ST, no aug.)	0.05±0.03	0.03±0.02	0.05±0.02	0.02±0.01	0.03±0.02	0.05±0.02	0.04±0.03	0.04±0.02
WGAIL-GP (ST)	0.10±0.04	0.08±0.03	0.11±0.01	0.06±0.02	0.07±0.03	0.11±0.04	0.08±0.02	0.08±0.01

Table 6: Additional results; refer to Table 5 for details.

	FindDupe							
	Demo	Jitter	Layout	Colour	Shape	CountPlus	Dynamics	All
AL (ST)	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.01	0.00±0.00	0.00±0.00
BC (MT)	0.89±0.04	0.78±0.12	0.05±0.03	0.31±0.07	0.81±0.07	0.03±0.01	0.81±0.08	0.05±0.03
BC (ST)	0.89±0.03	0.76±0.02	0.04±0.01	0.60±0.04	0.80±0.02	0.06±0.02	0.77±0.04	0.05±0.01
BC (ST, allo.)	0.93±0.02	0.79±0.07	0.01±0.01	0.72±0.11	0.75±0.09	0.02±0.01	0.74±0.08	0.02±0.01
BC (ST, no aug.)	0.94±0.06	0.38±0.06	0.06±0.03	0.36±0.04	0.75±0.08	0.04±0.02	0.78±0.06	0.06±0.02
BC (ST, no trans./rot. aug.)	0.93±0.03	0.45±0.10	0.09±0.01	0.43±0.05	0.81±0.07	0.05±0.03	0.75±0.09	0.06±0.03
GAIL (MT)	0.43±0.24	0.41±0.21	0.02±0.02	0.06±0.04	0.38±0.24	0.03±0.02	0.39±0.24	0.02±0.02
GAIL (ST)	0.98±0.02	0.97±0.01	0.10±0.02	0.23±0.06	0.95±0.04	0.05±0.02	0.96±0.02	0.05±0.01
GAIL (ST, allo.)	0.95±0.03	0.84±0.03	0.00±0.00	0.50±0.09	0.87±0.05	0.00±0.01	0.90±0.04	0.01±0.01
GAIL (ST, no aug.)	0.46±0.28	0.36±0.27	0.01±0.01	0.10±0.07	0.34±0.25	0.02±0.01	0.37±0.27	0.02±0.01
WGAIL-GP (ST)	0.70±0.09	0.46±0.11	0.02±0.00	0.09±0.04	0.64±0.04	0.02±0.01	0.62±0.03	0.03±0.02

	FixColour							
	Demo	Jitter	Layout	Colour	Shape	CountPlus	Dynamics	All
AL (ST)	0.00±0.01	0.03±0.04	0.02±0.01	0.03±0.03	0.00±0.01	0.01±0.01	0.01±0.01	0.01±0.02
BC (MT)	0.76±0.18	0.61±0.14	0.15±0.03	0.23±0.05	0.72±0.18	0.13±0.02	0.70±0.21	0.14±0.07
BC (ST)	0.62±0.14	0.45±0.18	0.18±0.04	0.19±0.03	0.63±0.12	0.18±0.04	0.71±0.13	0.13±0.03
BC (ST, allo.)	0.88±0.07	0.47±0.24	0.11±0.03	0.32±0.03	0.86±0.06	0.13±0.04	0.84±0.04	0.14±0.05
BC (ST, no aug.)	0.55±0.18	0.29±0.12	0.19±0.03	0.21±0.09	0.57±0.18	0.22±0.03	0.52±0.23	0.22±0.02
BC (ST, no trans./rot. aug.)	0.46±0.14	0.28±0.07	0.17±0.04	0.17±0.07	0.48±0.16	0.23±0.03	0.39±0.13	0.20±0.04
GAIL (MT)	0.99±0.02	0.65±0.17	0.30±0.08	0.21±0.06	0.99±0.01	0.18±0.10	0.95±0.07	0.16±0.06
GAIL (ST)	0.99±0.01	0.84±0.07	0.32±0.07	0.25±0.04	0.97±0.03	0.19±0.02	0.96±0.02	0.20±0.01
GAIL (ST, allo.)	0.98±0.00	0.66±0.21	0.06±0.02	0.36±0.01	0.99±0.01	0.09±0.04	0.98±0.01	0.09±0.03
GAIL (ST, no aug.)	0.98±0.02	0.64±0.12	0.16±0.05	0.27±0.03	0.87±0.07	0.13±0.04	0.84±0.07	0.14±0.03
WGAIL-GP (ST)	0.93±0.04	0.72±0.12	0.08±0.03	0.18±0.02	0.93±0.04	0.02±0.02	0.91±0.05	0.05±0.02

Table 7: Additional results; refer to Table 5 for details.

	ClusterColour							
	Demo	Jitter	Layout	Colour	Shape	CountPlus	Dynamics	All
AL (ST)	0.00±0.00	0.00±0.00	0.00±0.00	0.01±0.01	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00
BC (MT)	0.11±0.05	0.10±0.03	0.00±0.01	0.01±0.01	0.10±0.05	0.00±0.00	0.09±0.06	0.01±0.00
BC (ST)	0.16±0.05	0.17±0.04	0.01±0.01	0.01±0.00	0.17±0.04	0.01±0.01	0.16±0.05	0.01±0.00
BC (ST, allo.)	0.10±0.05	0.11±0.02	0.00±0.00	0.00±0.00	0.10±0.04	0.00±0.00	0.07±0.03	0.00±0.00
BC (ST, no aug.)	0.04±0.02	0.02±0.01	0.01±0.01	0.01±0.01	0.02±0.01	0.00±0.00	0.04±0.01	0.00±0.00
BC (ST, no trans./rot. aug.)	0.05±0.01	0.02±0.00	0.00±0.00	0.01±0.01	0.04±0.02	0.00±0.00	0.04±0.02	0.00±0.00
GAIL (MT)	0.01±0.01	0.01±0.01	0.01±0.00	0.01±0.00	0.01±0.01	0.00±0.00	0.01±0.01	0.00±0.00
GAIL (ST)	0.12±0.03	0.11±0.04	0.01±0.00	0.01±0.00	0.08±0.04	0.01±0.00	0.11±0.03	0.01±0.01
GAIL (ST, allo.)	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.01±0.01	0.00±0.00	0.01±0.01	0.00±0.00
GAIL (ST, no aug.)	0.02±0.01	0.01±0.01	0.00±0.00	0.01±0.01	0.02±0.01	0.01±0.01	0.02±0.01	0.00±0.00
WGAIL-GP (ST)	0.00±0.00	0.00±0.00	0.00±0.00	0.01±0.01	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00

	ClusterShape							
	Demo	Jitter	Layout	Colour	Shape	CountPlus	Dynamics	All
AL (ST)	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.01±0.00
BC (MT)	0.23±0.06	0.19±0.05	0.00±0.00	0.02±0.01	0.00±0.00	0.00±0.00	0.19±0.04	0.01±0.00
BC (ST)	0.35±0.11	0.26±0.06	0.01±0.01	0.06±0.02	0.01±0.00	0.00±0.00	0.31±0.04	0.01±0.01
BC (ST, allo.)	0.21±0.02	0.23±0.10	0.00±0.00	0.13±0.04	0.00±0.00	0.00±0.00	0.16±0.05	0.01±0.00
BC (ST, no aug.)	0.12±0.03	0.10±0.02	0.00±0.01	0.02±0.01	0.01±0.01	0.00±0.01	0.09±0.04	0.01±0.01
BC (ST, no trans./rot. aug.)	0.11±0.05	0.11±0.02	0.00±0.00	0.01±0.01	0.00±0.00	0.00±0.01	0.11±0.04	0.01±0.00
GAIL (MT)	0.01±0.01	0.01±0.01	0.01±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.01±0.02	0.01±0.01
GAIL (ST)	0.44±0.05	0.44±0.08	0.01±0.02	0.02±0.01	0.01±0.00	0.00±0.00	0.39±0.01	0.01±0.00
GAIL (ST, allo.)	0.00±0.00	0.00±0.00	0.01±0.01	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.01±0.00
GAIL (ST, no aug.)	0.01±0.02	0.02±0.01	0.00±0.00	0.00±0.00	0.00±0.01	0.01±0.00	0.01±0.01	0.01±0.00
WGAIL-GP (ST)	0.01±0.00	0.01±0.00	0.01±0.01	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00

Table 8: Additional results; refer to Table 5 for details.