Pragmatically Learning from Pedagogical Demonstrations in Multi-Goal Environments

Hugo Caselles-Dupré, Olivier Sigaud, Mohamed Chetouani
Sorbonne Université, CNRS, Institut des Systèmes Intelligents et de Robotique (ISIR)
Paris, France
caselles,olivier.sigaud,mohamed.chetouani@isir.upmc.fr

Abstract

Learning from demonstration methods usually leverage close to optimal demonstrations to accelerate training. By contrast, when demonstrating a task, human teachers deviate from optimal demonstrations and pedagogically modify their behavior by giving demonstrations that best disambiguate the goal they want to demonstrate. Analogously, human learners excel at pragmatically inferring the intent of the teacher, facilitating communication between the two agents. These mechanisms are critical in the few demonstrations regime, where inferring the goal is more difficult. In this paper, we implement pedagogy and pragmatism mechanisms by leveraging a Bayesian model of goal inference from demonstrations. We highlight the benefits of this model in multi-goal teacher-learner setups with two artificial agents that learn with goal-conditioned Reinforcement Learning. We show that combining a pedagogical teacher and a pragmatic learner results in faster learning and reduced goal ambiguity over standard learning from demonstrations, especially in the few demonstrations regime.

1 Introduction

Imagine a teacher-learner setup where 3 colored blocks are lying on a table, the green and red blocks are close to each other and the blue block is further away. Assume the teacher wants to demonstrate a goal state where the blue block is close to the green one (see Fig. 1). A naive teacher may move the blue block to the green one, but this would be ambiguous: is the goal to put the blue block close to the green one or to the red one? By contrast, a pedagogical teacher would move the green block close to the blue one, hence away from the red one, resolving the ambiguity as illustrated in Fig. 1. To infer that this demonstration facilitates learning, the pedagogical teacher may ask herself what goal she would infer if she was the learner. Using her policy and Bayesian inference, she can determine the probability that she would infer the right goal given this demonstration, and assume the learner would do the same.

Now, the learner has to interpret the demonstration to infer the goal. Given the same inference mechanism as the teacher, a literal learner would perform this inference using her own policy. On top of that, a pragmatic learner could first train her policy to predict her own goals from her own trajectories, increasing her likelihood to correctly infer the teacher's goal.

Those mechanisms, pedagogy from the teacher and pragmatism from the learner, facilitate communication between the two agents and thus improve learning by reducing ambiguity about the inferred goal. Pedagogy and pragmatism are concepts borrowed from cognitive science research. On the one hand, pedagogy is defined as the optimization of teaching concepts from examples [8, 9]. On the other hand, pragmatism is a property used to resolve ambiguities of intention interpretation from teaching signals, which can be language with for instance the rational speech act (RSA) [34, 20], or actions with pedagogical demonstrations [38].

Preprint. Under review.

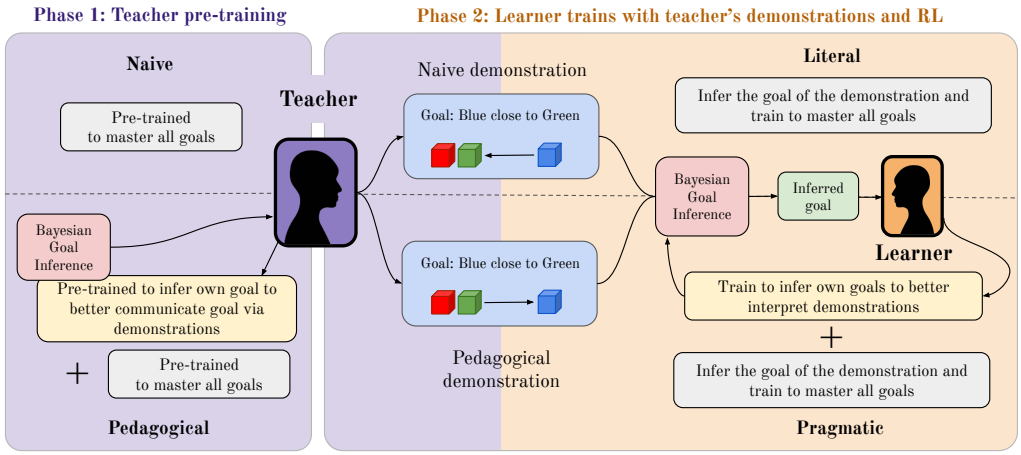


Fig. 1: Overview of our teacher-learner setup. All goal inferences are performed using Bayesian Goal Inference, an inference method that computes the goal probability from a demonstration using the agent's policy. In the first phase, the naive teacher is pre-trained to master all goals while the pedagogical teacher is additionally pre-trained to better infer goals from its own trajectories and effectively modifies its policy to produce demonstrations with less ambiguous goals. In the second phase, the learner is trained using the teacher's demonstrations and goal-conditioned Reinforcement Learning. The literal learner infers goals from demonstrations and is rewarded for correct inferences while the pragmatic learner additionally trains itself to infer its own goals from its own trajectories and effectively modifying its policy to better infer goals from the teacher's demonstrations.

Such ideas have been leveraged into the general theory of "Inferential Social Learning" (ISL) [22] in order to explain our abilities as learners and teachers to "interpret and generate evidence in social contexts". ISL is characterized as an inference mechanism guided by an intuitive understanding of how people think, plan, and act. It leverages Bayesian inference to explain several key aspects involved in pedagogy and pragmatism: pedagogy can help avoid goal ambiguity [26, 27] or accelerate goal inference from a learner (i.e. legibility) [29], and pragmatism can help better infer goals [14], or detect pedagogy [22], or generalize from induction [23] (what can I deduce from a teaching signal?). In this paper, we show that using Bayesian inference for disambiguating goals (pedagogy) and better infer goals (pragmatism) can improve the learning capabilities of artificial agents. In particular, we show that this mechanism is critical to reduce ambiguity when the teacher performs few demonstrations, which may contribute to more feasible robot teaching in the future.

We present an approach for implementing inferential mechanisms to improve learning from demonstrations in an artificial teacher-learner setup, as illustrated in Fig. 1. We use two multi-goal environments: a simple one for illustrating our point, and a more complex simulated robotics block manipulation environment to show the robustness of our approach. In both environments, we create naive teachers trained to master all goals, and pedagogical teachers trained to provide demonstrations that best disambiguate between all goals and thus facilitate goal inference for learners. We then create literal learners, which learn from teacher demonstrations, and pragmatic learners, which adapt their policy to better infer goals from the teacher's demonstrations. The mechanism for implementing pedagogy and pragmatism is an additional reward for correct goal inference of the agent's own trajectories. Goal inference is implemented using a Bayesian model and the agent's policy.

In the supplementary material, we provide a link to our codebase and to an illustrative video.

We make the following contributions:

1. Inspired by the ISL framework, we introduce pedagogical teaching and pragmatic learning mechanisms based on Bayesian Goal Inference and show that these mechanisms can be leveraged in a cooperative artificial teacher-learner training setup where both agents use goal-conditioned policies.
2. We show that pedagogy and pragmatism help reduce goal ambiguity and result in faster learning, which is especially helpful when the teacher performs fewer demonstrations.

2 Related work

Our work is related to several strongly connected areas.

Bayesian inference. Bayesian Inference was already used as a key mechanism for goal inference in the context of inverse planning [7, 5, 43]. It uses the Bayes formula to compute the probabilities of goals given the actions and the policy. In our work we use it as a tool to produce non-ambiguous demonstrations on the teacher side and pragmatic inference on the learner side.

Pedagogical demonstrations. In an attempt to explain how humans demonstrate tasks to each other, Ho et al. [29, 28, 26] introduce the Observer Belief MDP model and show that producing more pedagogical demonstrations results in better performance. However, their work does not investigate how this affects an artificial learner, nor the effect of pragmatism in the learner.

Legibility in Human-Robot Interaction. Similarly to us, [13, 32] modify robot action sequences to allow an observer (a human in their case) to more quickly and successfully understand a robot's goal from a more legible trajectory. In our work, Bayesian goal inference is employed in pedagogical teachers to allow them to generate less ambiguous goal demonstration using additional reward for correct goal inference on their own motion.

Pragmatic inference. A long line of work in linguistics, natural language processing, and cognitive science has studied pragmatics: how linguistic meaning is affected by context and communicative goals [18, 17, 21]. Another line of work on pragmatics, more related to our work, focuses on inferring meaning from action in a context. It has been applied to Robotics [33, 15], where a pragmatic robot better infers the objective of a teacher by considering its pedagogical intentions, in a multi-agent game theory context. In our case, we are interested in how a pragmatic learner can learn by itself to better infer the goals of a demonstrator, resulting in faster task learning, which is different from detecting if the teacher is pedagogical or not.

Demonstrations in goal-conditioned tasks. Our learning from demonstration mechanism is related to work on learning from demonstrations in multi-goal environments, by contrast with the more common setup where the agent only learns one task. Most approaches targeting fewer demonstrations address the one task setup [1, 44, 42, 40]. Approaches using demonstrations for goal-conditioned tasks such as GAIL [25], DCRL [10] and CLIC [16] combine goal-conditioned RL and imitation learning with additional rewards. In our work, we also combine a goal-conditioned RL algorithm (GANGSTR [3]) and a slightly modified version of an algorithm learning both from demonstration and from the agent's own experience (SQIL [24]). The specificity of our pipeline is that we can easily improve it with pedagogical demonstrations and pragmatic inference to accelerate training, but in principle all goal-conditioned RL algorithms using demonstrations could be improved in such a way.

3 Methods

We consider multi-goal environments with teacher-learner scenarios where learners are both trained with teachers' demonstrations and using their own exploration. The teacher T and the learner L are both represented by their respective goal-conditioned policies $\pi_T(.|g)$ and $\pi_L(.|g)$. These policies might be learned using any algorithm (multi-armed bandits, RL, evolution strategies, etc.), here we use a goal-conditioned RL (GCRL) algorithm [11]. Our multi-goal environments present goal ambiguity, as the same trajectory can simultaneously reach at least two goals (g_1, g_2). Considering this hypothesis, it is generally not possible to reliably predict the pursued goal from a demonstration reaching both g_1 and g_2.

Both agents share common goal, state and action spaces. They communicate through teacher's demonstrations, learner's inferred goal and teacher feedback on this inference. To efficiently exploit these signals, we introduce a Bayesian Goal Inference (BGI) mechanism helping agents infer goals from demonstrations. The teacher uses it to generate less ambiguous demonstrations (pedagogy), while the learner uses it to infer the teacher's goal from the demonstration (pragmatism).

The training process is spread across two phases: 1) the teacher is pre-trained to master all goals in the environment using GCRL, and 2) the learner infers the goal from a teacher's demonstration using Bayesian Goal Inference, the teacher provides feedback on the inference, and the learner is rewarded for correct predictions. The learner then attempts at reaching the inferred goal and iteratively improves its policy using GCRL.

Below we formally define BGI, then we define naive teachers and literal learners, which do not leverage the full benefits of BGI. Finally we introduce pedagogical teachers and pragmatic learners, which use BGI to facilitate communication with each other.

3.1 Bayesian Goal Inference in Teacher-Learner Interactions

We formally introduce the BGI mechanism used to infer goals from a goal-conditioned policy. In our work, 1) the teacher uses it to implement pedagogy, 2) the learner uses it to infer goals from the demonstrations of the teacher, and 3) the learner uses it to implement pragmatism.

Inferring the Goal from a Demonstration In a goal-conditioned Markov environment, the probability of observing a demonstration $d = ((s_1, a_1), .., (s_n, a_n))$ given a goal g and the goal-conditioned policy $\pi(.|g)$ that generated it can be written [26, 6] as:

$$\begin{aligned}\mathbb{P}(d|g) &= \mathbb{P}((s_1, a_1), .., (s_n, a_n)|g) \\ &= \prod_{i=1}^{n} \pi(a_i|s_i, g) \cdot \mathbb{P}(s_{i+1}|s_i, a_i) = \prod_{i=1}^{n} \pi(a_i|s_i, g),\end{aligned} \tag{1}$$

where $\mathbb{P}(s_{i+1}|s_i, a_i) = 1$ as we consider a deterministic environment.

Now, to infer a goal given a demonstration, by using Bayes' rule we can derive $\mathbb{P}(G|d)$, the probability distribution over the goal space G given the demonstration:

$$\mathbb{P}(G|d) \propto \mathbb{P}(d|G) \cdot \mathbb{P}(G) = \prod_{i=1}^{n} \pi(a_i|s_i, G) \cdot \mathbb{P}(G). \tag{2}$$

For each goal g, the prior $\mathbb{P}(G)$ is uniform if not specified otherwise. Given $\mathbb{P}(G|d)$, an agent can infer the goal of a demonstration by either taking the most probable goal, or sampling from the distribution. To perform this inference, the agent uses its own policy.

Learning Goal Inference from Own Trajectories When playing its policy, an agent produces trajectories which we call demonstrations when they are produced for other agents. A GCRL agent can leverage the BGI mechanism on its own trajectories to improve its ability to infer goals. To do so, during training, the agent selects a goal, performs a trajectory, infers the goal from the trajectory, and rewards itself if the inference is correct. This reinforces the policy towards actions that lead to better goal inference. We use this to implement both pedagogy in the teacher and pragmatism in the learner.

3.2 Naive/Pedagogical Teacher and Literal/Pragmatic Learner training

We now present the two training phases in our teacher-learner setup. First, the teacher is pre-trained, and then it provides demonstrations for the learner to train with. The two-phases training process is presented in Algorithm 1.

Phase 1: Teacher pre-training. The teacher is pre-trained before providing demonstrations to the learner. The **naive teacher**'s policy is trained to master all goals: it maintains a goal set G_T to which it adds newly encountered goals, it samples goals from G_T and pursues them to collect trajectories added to a replay buffer. Finally, it applies GCRL for policy updates.

The **pedagogical teacher** is also trained to master all goals, but additionally trains itself to predict its own goals using BGI. When training, each time it successfully reaches a goal, it takes its own trajectory and infers the pursued goal using BGI, effectively asking itself: "would a learner be able to predict the goal from this demonstration?". If it correctly infers the goal from its own trajectory, it rewards itself to reinforce its probability to select this trajectory. This is done with GCRL by adding a "pedagogical reward" to the trajectory for which the teacher correctly infers its own goal. This biases policy learning towards finding demonstrations which avoid ambiguity in goal inference, while guaranteeing high performance on the actual task.

Phase 2: Training the Literal/Pragmatic Learner with Teacher's demonstrations. The **literal learner**'s goal-conditioned policy $\pi_L(.|g)$ is trained to master all goals in the environment using teacher's demonstrations and GCRL.

At each iteration, the teacher samples a desired goal g_d it wants the learner to achieve. It then presents a demonstration d for the chosen goal to the learner. Using BGI with its own policy, the learner infers the goal $\widehat{g_d}$ from the teacher's demonstration. The teacher provides feedback, telling the learner if $g_d = \widehat{g}$. If the learner correctly inferred the goal of the teacher's demonstration ($\widehat{g_d} = g_d$), the demonstration is added to the learner's replay buffer and used for further training. Then the learner plays its policy conditioned on the predicted goal $\pi_L(.|\widehat{g_d})$ and obtains a trajectory t resulting in an achieved goal g_a, which is added to the replay buffer. If g_a is achieved for the first time, then the teacher updates the set of goals from which it can sample. Finally, GCRL improves the policy from its own trajectories and demonstrations drawn from the replay buffer.

A **pragmatic learner** improves over a literal learner by training its own policy to better infer the right goals from the teacher's demonstrations. If the learner is able to infer its own goals using BGI, then it will better infer the teacher's goals. The pragmatic learner thus adds a "pragmatic reward" to its trajectories for which it can correctly infer its own goals.

Algorithm 1 Two-phases training of the teacher and the learner

PHASE 1: Teacher pre-training ($\pi_T(.|g)$)
Initialize $\pi_T(.|g)$ and goal set G_T to {first goal}
for $epoch = 1, 2, \ldots$ **do**
    Sample goal g from goal set G_T
    Run policy $\pi_T(.|g)$, obtain trajectory $t = ((s_1, a_1, r_1), .., (s_n, a_n, r_n))$ and achieved goal g_a
    if Teacher is pedagogic and $g = g_a$ **then**
        Infer own goal $\widehat{g}$ with BGI on t: if $\widehat{g} = g$, add pedagogical reward to t
    end if
    Add t to replay buffer and if g_a is new, add g_a to goal set G_T
    Update policy π_T with GCRL
end for

PHASE 2: Learner ($\pi_L(.|g)$) training with Teacher's demonstrations
Initialize $\pi_L(.|g)$ and goal set G_L to {first goal}
for $epoch = 1, 2, \ldots$ **do**
    Teacher samples goal g_d from goal set G_L and provides demonstration d by running $\pi_T(.|g_d)$
    Learner infers goal $\widehat{g_d}$ of d with BGI
    if $\widehat{g_d} = g_d$ (feedback provided by the teacher) **then**
        Add d to learner's replay buffer
    end if
    Run policy $\pi_L(.|\widehat{g_d})$, obtain trajectory $t = ((s_1, a_1, r_1), .., (s_n, a_n, r_n))$ and achieved goal g_a
    if Learner is pragmatic and $\widehat{g_d} = g_a$ **then**
        Infer own goal $\widehat{g_o}$ with BGI on t: if $\widehat{g_o} = g_a$, add pragmatic reward to t
    end if
    Add t to learner's replay buffer and if g_a is new, add g_a to goal set G_L
    Update policy π_L with GCRL
end for

4 Experimental Setup

We use two environments as test-beds for our experiments: a simple one inspired from research in Developmental Psychology [22] called "Draw two balls" (DTB) that we initially used for illustrating our approach, and a more complex one called "Fetch block-stacking" (FBS) to test our approach on a more challenging domain [36]. Results on DTB are presented in Appendix A, and the main paper focuses on FBS. Both DTB and FBS are multi-goal environments and present goal ambiguity.

4.1 Environment: Fetch Block Stacking

FBS is a block-stacking environment with two Fetch robots (teacher and learner) equipped with robotic arms, see Fig. 2. It is based on MuJoCo [41] and derived from the Fetch tasks [36].

The teacher and learner share goal/state/action spaces. Actions are 4-dimensional: 3D gripper velocities and grasping velocity. Observations are the Cartesian and angular positions and velocities of the gripper and the three blocks. Following [2], we adopt a semantic predicates representation where we add to the observation a binary vector telling whether two blocks are close, and whether any block is on top of any other. The agent uses these binary vectors as goals (with 35 possible goals including stacks and pyramids). Given a particular goal vector configuration, the agent gets a $+1$ reward for each pair of blocks (3 in total) if true predicates about the pair of blocks are the same in the current vector and the goal vector. There are several possibilities to match the true predicates in the goal vector, which induces goal ambiguity. Details about the environment implementation are provided in Appendix B.1.

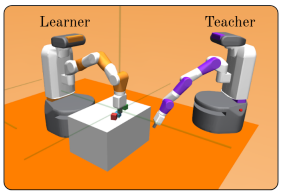


Fig. 2: Fetch Block Stacking.

Naive/Pedagogical teachers implementation. The training procedure and policy architecture of the naive teacher are taken from GANGSTR [3] which already implements Fetch Block Stacking. All architecture, training and hyperparameters details are provided in Appendix B.2. The policy is a message passing graph neural network [19] parametrized by θ: $\pi_{\theta,T}(a|s_t, g) = \mathbb{P}(a|s_t, g)$. This goal-conditioned policy is trained using Soft Actor Critic (SAC) [24], a state-of-the-art RL algorithm, combined with Hindsight Experience Replay (HER) [4]. Additionally, the pedagogical teacher rewards itself with the "pedagogical reward" (1 here) when it correctly infers the goal of its successful trajectories.

Literal/Pragmatic learners implementation. To implement literal and pragmatic learner training, we use a combination of GANGSTR combined with a slightly modified version of the Soft Q-Imitation Learning (SQIL) algorithm [37]. Originally, SQIL rewards demonstrations to 1 and experience to 0. By contrast, we set the demonstration reward to 1 + the maximum reward obtainable in the environment (3 here), while the reward of trajectories performed in the environment is unchanged. This allows the agent to learn from demonstrations and collected trajectories at the same time using SAC as the GCRL algorithm. As in the original paper, the percentage of demonstrations versus collected trajectories is set to 0.5. On top of that, the pragmatic learner adds the "pragmatic reward" (1 here) to its trajectories from which it can infer the goal.

4.2 Metrics

We use four metrics for all of our experiments: 1) Goal Inference Accuracy (GIA): "is the learner able to correctly infer goals given demonstrations from a teacher?", 2) Own Goal Inference Accuracy (OGIA): "is the agent (teacher or learner) able to correctly infer goals from its own trajectories?", 3) Goal Reaching Accuracy (GRA): "is the learner able to reach all goals in the environment?", and 4) the product of GIA and GRA (GIAxGRA), interpreted as the ability of the learner to predict its goal given a demonstration from the teacher and then reach it. GIA and OGIA help us understand if the pedagogy and pragmatism mechanism actually work, and are computed using a test set of 50 demonstrations per goal generated by the teacher for GIA and by the same agent being tested for OGIA ($50 * 35 = 1750$ demonstrations in total). GRA allows us to evaluate the performance of agents and is computed by testing the agent on each goal 50 times. All demonstrations are long enough to reach the goal (100 timesteps in our case). We provide means μ and standard deviations over 5 seeds and report statistical significance using a two-tail Welch's t-test with null hypothesis $\mu_1 = \mu_2$, at level $\alpha = 0.05$ (noted by star markers in figures).

Finally, to compare pedagogical and naive demonstrations, we derive an Ambiguity Score: given two ambiguous goals that can be achieved simultaneously and a starting state, the ambiguous score is 1 if their associated demonstrations achieve the same goals (the demonstrations are thus ambiguous), and 0 otherwise. This score assumes that the demonstrator already knows how to achieve the goals. When it is computed from several situations, it evaluates the degree of ambiguity in the demonstrations. For more details on this score please refer to Appendix B.2.5.

5 Experiments and Results

Here we first study learning results from Phase 1 (naive/pedagogical teachers training) and Phase 2 (literal/pragmatic learners trained with teacher's demonstrations). We analyze the impact of demonstrations to accelerate learning, and then scrutinize two algorithmic choices in our approach: goal inference with BGI and learning from demonstrations with SQIL. We take the GANGSTR policy architecture and algorithm as they are and refer to the original paper [3] for their justification.

5.1 Experiments on Phase 1: Naive and Pedagogical Teachers

We verify that the pedagogical teacher can indeed better predict goals from its demonstrations compared to a naive teacher. Quantitatively, the pedagogical teacher achieves an OGIA of $95.6\% \pm 0.1$ while the naive teacher achieves $83.4\% \pm 0.2$. By training to infer its own goals from its demonstrations, it produces less ambiguous demonstrations. For a qualitative evaluation of this phenomenon, we implement the example in the introduction. We compare demonstrations from a common starting situation (red block close to green block, blue block away) and for two different goals: "Put blue block close to red" and "Put all three blocks close together". As a result, the naive teacher provides the same demonstrations (putting the blue block close to red), while the pedagogical teacher provides two different demonstrations to better disambiguate between the two goals (putting the red block close to blue and putting the blue block close to red and green).

We compute the Ambiguity Score from 500 ambiguous situations sampled from a list of ambiguous situations detailed in Appendix B.2.5 (two ambiguous goals and a starting state) and obtain an unequivocal result: $9\% \pm 1$ for the pedagogical teacher vs $64\% \pm 2$ for the naive teacher. Pedagogical demonstrations are thus 7 times less ambiguous than naive ones.

5.2 Experiments on Phase 2: Literal and Pragmatic Learners

Table 1: Goal Reaching Accuracy (GRA) and Goal Inference Accuracy (GIA) for the Fetch Block Stacking environment.

Teacher + Learner	GIA	GRA	GIAxGRA
With 100 demonstrations per goal			
Naive + Literal	$30.9 \pm 8.3\%$	$50.2 \pm 2.7\%$	0.15
Naive + Pragmatic	$61.3 \pm 2.9\%$	$84.8 \pm 4.1\%$	0.52
Pedagogical + Literal	$49.6 \pm 6.6\%$	$62.7 \pm 5.4\%$	0.31
Pedagogical + Pragmatic	$\mathbf{69.2 \pm 4.2\%}$	$\mathbf{89.4 \pm 2.0\%}$	**0.62**
With 500 demonstrations per goal			
Naive + Literal	$68.9 \pm 1.1\%$	$90.6 \pm 1.4\%$	0.62
Naive + Pragmatic	$76.3 \pm 0.4\%$	$95.5 \pm 0.6\%$	0.73
Pedagogical + Literal	$78.8 \pm 0.7\%$	$91.5 \pm 1.1\%$	0.72
Pedagogical + Pragmatic	$\mathbf{82.2 \pm 1.2\%}$	$\mathbf{96.8 \pm 0.6\%}$	**0.80**
With 1000 demonstrations per goal			
Naive + Literal	$75.3 \pm 0.9\%$	$93.3 \pm 1.2\%$	0.70
Naive + Pragmatic	$76.0 \pm 0.9\%$	$96.6 \pm 0.4\%$	0.73
Pedagogical + Literal	$80.0 \pm 1.6\%$	$94.5 \pm 1.2\%$	0.76
Pedagogical + Pragmatic	$\mathbf{84.7 \pm 1.8\%}$	$\mathbf{97.1 \pm 1.2\%}$	**0.82**

We train pragmatic/literal learners with naive/pedagogical teachers and present our results in Fig. 3 and Table 1. In order to evaluate how the mechanisms would respond to training regimes with few demonstrations, we performed the experiments in FBS with a different number of available demonstrations per goal (1000, 500 and 100) which are performed for random starting states. As presented in Fig. 3 and Table 1, it is highly preferable to opt for a pedagogical teacher along with a pragmatic learner. The benefits of adopting pedagogy and pragmatism are particularly prominent in the few demonstrations regime (100 per goal) where pedagogical+pragmatic combination performs 4 times better (going from 0.15 to 0.62) than the naive+literal one in terms of GIAxGRA.

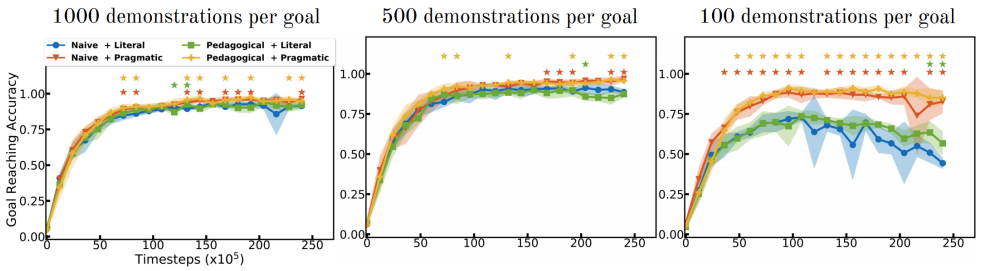


Fig. 3: Results for FBS environment (Goal Reaching Accuracy (GRA) with different numbers of demonstrations per goal). Stars indicate significance (tested against naive+literal).

An analysis of the pragmatic learner policies reveals that they are better at predicting their own goals, as intended, with an average of 12.4% relative difference in OGIA between a literal and a pragmatic learner. This, coupled with the effectiveness of pedagogical demonstrations shown in the previous section, helps the teacher and learner efficiently communicate and improves learning speed. The benefit of using pragmatism is prominent in the few demonstrations regime: the addition of pragmatism alone (red curve in Fig. 3) helps mastering all goals while pedagogy on its own (green curve in Fig. 3) is not enough to do so.

5.3 Does using the teacher accelerate training for the learner?

In Phase 2 of our setup, the learner is trained using teacher's demonstrations and RL on its own experience. But is the teacher useful? The answer lies in the training speed. We thus compare the training speed of pragmatic learners trained with 100, 500 and 1000 pedagogical demonstrations and their own collected experience (using the same training procedure as before, with SQIL and GANGSTR), and a learner trained with the same architecture and training process, but no demonstration. We end up with a common result in the Learning from Demonstrations literature: the learner trains much faster (here roughly twice faster to obtain a GRA of 95%) than the teacher, as illustrated in Fig. 4. This results justifies training learners from demonstrations.

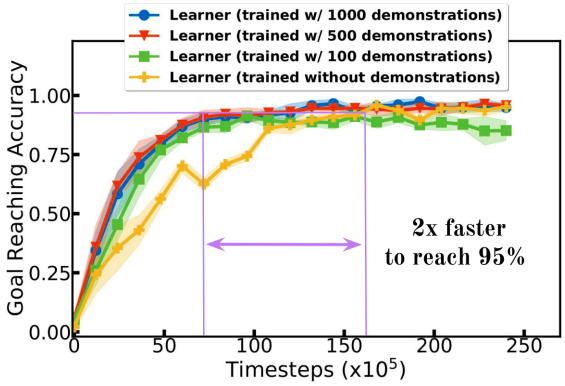


Fig. 4: Demonstrations double the learner's training speed.

5.4 Could we use another goal inference method instead of Bayesian Goal Inference?

Table 2: Goal Inference Accuracy (GIA) comparison between using BGI and a Goal Prediction Neural Network (GPNN) on FBS.

GIA for Pedagogical demonstrations			
Nb of demos per goal	100 demos	500 demos	1000 demos
GPNN	46.2%	70.0%	82.0%
BGI	**69.2%**	**82.2%**	**84.7%**
GIA for Naive demonstrations			
Nb of demos per goal	100 demos	500 demos	1000 demos
GPNN	38.4%	67.1%	68.2%
BGI	**61.3%**	**76.3%**	**76.0%**

Our learner relies on BGI using its own policy to infer the goals associated with the demonstrations of the teacher. Could we instead train a separate goal prediction module to learn how to achieve goal inference? We tested this by creating a goal prediction neural network (GPNN) based on a LSTM architecture [30] that inputs a demonstration and outputs a goal, and we trained it using the same demonstrations and goals as for the BGI case. In order to compare our BGI approach to this alternative, we trained GPNN on a different number of demonstrations per goal and compared the results on a separate test set. We performed this experiment on FBS and provide details in Appendix B.2.3. Results from Table 2 show that BGI achieves higher GIA in all cases (naive or pedagogical demonstrations), and again performs especially well in the few demonstrations regime where the GPNN approach struggles because of too few training samples.

5.5 What if we used another method to learn from demonstrations?

Our modified version of SQIL helps the learner leverage both demonstrations and the experience it collects. We justify the use of this method by comparing it to baseline approaches: learning only from demonstrations (B1), adding behavioural cloning on the demonstrations (B2) and the original version of SQIL (B3). All implementation details are provided in Appendix B.2.4. We perform the comparison using teacher's pedagogical demonstrations and pragmatism in the learner for all methods. B1 is not able to learn, B2 is able to achieve a reasonable GRA score but struggles to predict goals from demonstrations because the behavioural cloning updates interfere with the BGI mechanism by changing the action probabilities. B3 and our modified SQIL approach perform well, with a significant advantage for our approach. The original SQIL was designed for single goal environments and does not provide enough goal-reward associations, while our modification extends the approach to multi-goal environments by providing a reward signal for the collected experience.

Table 3: Learning from demonstrations baselines.

Method	GIA (%)	GRA (%)	GIAxGRA
B1	5.3 ± 1.5	6.2 ± 0.7	0.00
B2	2.3 ± 1.8	52.5 ± 2.1	0.00
B3	59.1 ± 2.4	73.8 ± 1.7	0.43
Ours	$\mathbf{69.2 \pm 4.2}$	$\mathbf{89.4 \pm 2.0}$	**0.62**

6 Conclusion

In this paper, building on the pedagogy and pragmatism concepts from Developmental Psychology, we have shown how learning from demonstration can benefit from a Bayesian goal inference mechanism. It improves learning speed and resolves ambiguity in communication, especially in the few demonstrations regime. In our work, the teacher's demonstration was insensitive to the learner's interpretation, as the teacher did not use a model of the learner's policy to provide a demonstration tailored to that specific learner. Conversely, the learner did not have a model of the teacher's policy which may help it interpret the goal the teacher wanted to convey. In the future, adding in each agent a model of the other agent's policy would result in closing the interaction loop and favoring the emergence of richer interaction mechanisms, such as both partners using signals to help the other update their model of each other during training. Besides, our approach was focused on goal disambiguation for pedagogy and improving goal inference for pragmatism. Pedagogy could be further improved with a curriculum of goals that would improve exploration [3]. Pragmatism could be improved with pedagogy detection [22] and inductive generalization [23]. In the case of multiple teachers, the learner could also develop a mechanism to evaluate teachers and select the most suited one [35]. We leave all this for future work.

Finally, a special trait of our learner is that it learns both from its own signals and from interaction with its teacher, in accordance with the guidelines of [39]. While pedagogy and pragmatism are mechanisms that improve how agents can be taught, they suggest a number of other mechanisms to do so, which in combination to our approach could be beneficial such as language guided learning, internalization, motivation regulation and observational learning.

References

[1] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1, 2004.

[2] Ahmed Akakzia, Cédric Colas, Pierre-Yves Oudeyer, Mohamed Chetouani, and Olivier Sigaud. Grounding language to autonomously-acquired skills via goal generation. In *ICLR 2021-Ninth International Conference on Learning Representation*, pages 1–21, 2021.

[3] Ahmed Akakzia, Olivier Serris, Cédric Colas, and Olivier Sigaud. Help me explore: Minimal social interventions for graph-based autotelic agents. *arXiv preprint arXiv:XXX*, 2022.

[4] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

[5] Chris L. Baker, Noah D. Goodman, and Joshua B. Tenenbaum. Theory-based social goal inference. In *Proceedings of the thirtieth annual conference of the cognitive science society*, pages 1447–1452. Citeseer, 2008.

[6] Chris L. Baker, Rebecca Saxe, and Joshua B. Tenenbaum. Action understanding as inverse planning. *Cognition*, 113(3):329–349, 2009.

[7] Chris L. Baker, Joshua B. Tenenbaum, and Rebecca R. Saxe. Goal inference as inverse planning. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 29, 2007.

[8] Daphna Buchsbaum, Alison Gopnik, Thomas L. Griffiths, and Patrick Shafto. Children's imitation of causal action sequences is influenced by statistical and pedagogical evidence. *Cognition*, 120(3):331–340, 2011.

[9] Lucas P. Butler and Ellen M. Markman. Preschoolers use pedagogical cues to guide radical reorganization of category knowledge. *Cognition*, 130(1):116–127, 2014.

[10] Théo Cachet, Christopher R. Dance, and Julien Perez. Conditioned reinforcement learning for few-shot imitation. In *International Conference on Machine Learning*, pages 2376–2387. PMLR, 2021.

[11] Cédric Colas, Tristan Karch, Olivier Sigaud, and Pierre-Yves Oudeyer. Intrinsically motivated goal-conditioned reinforcement learning: a short survey. *arXiv preprint arXiv:2012.09830*, 2020.

[12] Lisandro D Dalcin, Rodrigo R Paz, Pablo A Kler, and Alejandro Cosimo. Parallel distributed computing using python. *Advances in Water Resources*, 34(9):1124–1139, 2011.

[13] Anca D. Dragan, Kenton C.T. Lee, and Siddhartha S. Srinivasa. Legibility and predictability of robot motion. In *2013 8th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 301–308. IEEE, 2013.

[14] Terje Falck-Ytter, Gustaf Gredebäck, and Claes Von Hofsten. Infants predict other people's action goals. *Nature neuroscience*, 9(7):878–879, 2006.

[15] Jaime F Fisac, Monica A Gates, Jessica B Hamrick, Chang Liu, Dylan Hadfield-Menell, Malayandi Palaniappan, Dhruv Malik, S. Shankar Sastry, Thomas L. Griffiths, and Anca D. Dragan. Pragmatic-pedagogic value alignment. In *Robotics Research*, pages 49–57. Springer, 2020.

[16] Pierre Fournier, Cédric Colas, Mohamed Chetouani, and Olivier Sigaud. Clic: Curriculum learning and imitation for object control in nonrewarding environments. *IEEE Transactions on Cognitive and Developmental Systems*, 13(2):239–248, 2019.

[17] Michael C Frank and Noah D Goodman. Predicting pragmatic reasoning in language games. *Science*, 336(6084):998–998, 2012.

[18] Daniel Fried, Ronghang Hu, Volkan Cirik, Anna Rohrbach, Jacob Andreas, Louis-Philippe Morency, Taylor Berg-Kirkpatrick, Kate Saenko, Dan Klein, and Trevor Darrell. Speaker-follower models for vision-and-language navigation. *Advances in Neural Information Processing Systems*, 31, 2018.

[19] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

[20] Noah D. Goodman and Michael C. Frank. Pragmatic language interpretation as probabilistic inference. *Trends in cognitive sciences*, 20(11):818–829, 2016.

[21] Noah D Goodman and Andreas Stuhlmüller. Knowledge and implicature: Modeling language understanding as social cognition. *Topics in cognitive science*, 5(1):173–184, 2013.

[22] Hyowon Gweon. Inferential social learning: cognitive foundations of human social learning and teaching. *Trends in Cognitive Sciences*, 2021.

[23] Hyowon Gweon, Joshua B. Tenenbaum, and Laura E. Schulz. Infants consider both the sample and the sampling process in inductive generalization. *Proceedings of the National Academy of Sciences*, 107(20):9066–9071, 2010.

[24] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[25] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016.

[26] Mark K. Ho, Fiery Cushman, Michael L. Littman, and Joseph L. Austerweil. Communication in action: Planning and interpreting communicative demonstrations, 2019.

[27] Mark K Ho and Thomas L Griffiths. Cognitive science as a source of forward and inverse models of human decisions for robotics and control. *Annual Review of Control, Robotics, and Autonomous Systems*, 5, 2021.

[28] Mark K Ho, Michael L Littman, Fiery Cushman, and Joseph L Austerweil. Effectively learning from pedagogical demonstrations. In *Proceedings of the annual conference of the cognitive science society*, 2018.

[29] Mark K. Ho, Michael L. Littman, James MacGlashan, Fiery Cushman, and Joseph L. Austerweil. Showing versus doing: Teaching by demonstration. In *NeurIPS*, 2016.

[30] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[31] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[32] Christina Lichtenthäler, Tamara Lorenz, and Alexandra Kirsch. Towards a legibility metric: How to measure the perceived value of a robot. In *International Conference on Social Robotics, ICSR 2011*, 2011.

[33] Smitha Milli and Anca D Dragan. Literal or pedagogic human? analyzing human model misspecification in objective learning. In *Uncertainty in artificial intelligence*, pages 925–934. PMLR, 2020.

[34] Stephen Neale. Paul Grice and the philosophy of language. *Linguistics and philosophy*, pages 509–559, 1992.

[35] Sao Mai Nguyen and Pierre-Yves Oudeyer. Active choice of teachers, learning strategies and goals for a socially guided intrinsic motivation learner. *Paladyn*, 3(3):136–146, 2012.

[36] Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, et al. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*, 2018.

[37] Siddharth Reddy, Anca D Dragan, and Sergey Levine. Sqil: Imitation learning via reinforcement learning with sparse rewards. *arXiv preprint arXiv:1905.11108*, 2019.

[38] Patrick Shafto, Noah D. Goodman, and Thomas L. Griffiths. A rational account of pedagogical reasoning: Teaching by, and learning from, examples. *Cognitive psychology*, 71:55–89, 2014.

[39] Olivier Sigaud, Hugo Caselles-Dupré, Cédric Colas, Ahmed Akakzia, Pierre-Yves Oudeyer, and Mohamed Chetouani. Towards teachable autonomous agents. *arXiv preprint arXiv:2105.11977*, 2021.

[40] Bradly C. Stadie, Siyan Zhao, Qiqi Xu, Bonnie Li, and Lunjun Zhang. One demonstration imitation learning. *Advances in neural information processing systems*, 30, 2019.

[41] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

[42] Tianhao Zhang, Zoe McCarthy, Owen Jow, Dennis Lee, Xi Chen, Ken Goldberg, and Pieter Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5628–5635. IEEE, 2018.

[43] Tan Zhi-Xuan, Jordyn L. Mann, Tom Silver, Joshua B. Tenenbaum, and Vikash K. Mansinghka. Online bayesian goal inference for boundedly-rational planning agents. *arXiv preprint arXiv:2006.07532*, 2020.

[44] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, Anind K Dey, et al. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] This work proposes a general framework for implementing pedagogy and pragmatism in learning from demonstrations, which does not have extra negative societal impacts beyond learning from demonstrations.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix B.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See all figures and tables.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix B.2.2.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Experiments on toy environment

A.1 Draw Two Balls environment

The DTB environment is inspired from the squeaking balls environment presented in [22], see Fig. 5. In our variant, there is a bucket of purple, orange and pink balls to choose from. Purple balls are more numerous than any other balls. The agent must pick two balls consecutively, and upon its choice, it can obtain three possible outcomes. If the picked balls are (orange, orange) or (pink, orange), goal 1 is reached. If the picked balls are (orange, pink), goal 1 and goal 2 are achieved. Otherwise, no goal is reached and nothing happens, which we will call reaching goal 0. The achievement of the goals are communicated to the agent via a characteristic sound is played upon the reaching of a goal as in [22]. Note that we purposely introduced goal ambiguity in the environment: if the agent selects the actions (orange, pink), one does not know which goal it was aiming for without an hypothesis on the agent. The DTB environment can then be adapted as a teacher-learner environment where the teacher can demonstrate the goals and the learner can infer the goals from the demonstrations.

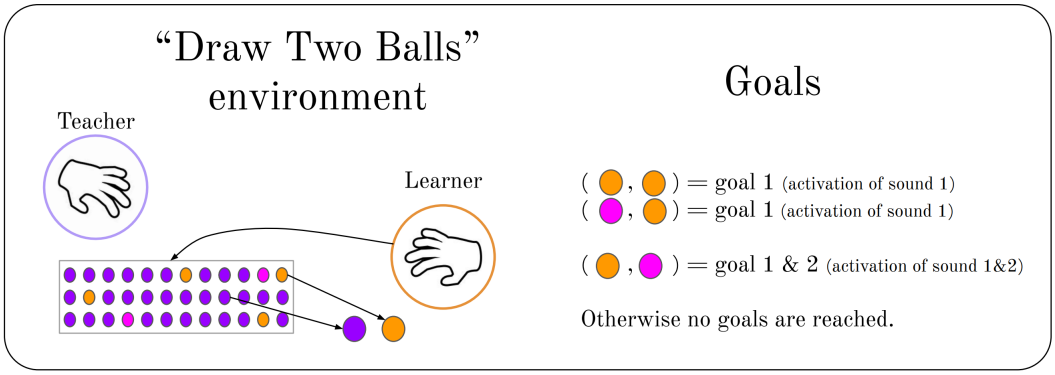


Fig. 5: The "Draw Two balls" (DTB) environment.

We use the same two phases training process in DTB as in FBS, and describe thereafter the details of training on DTB.

A.2 Phase 1: Teacher pre-training

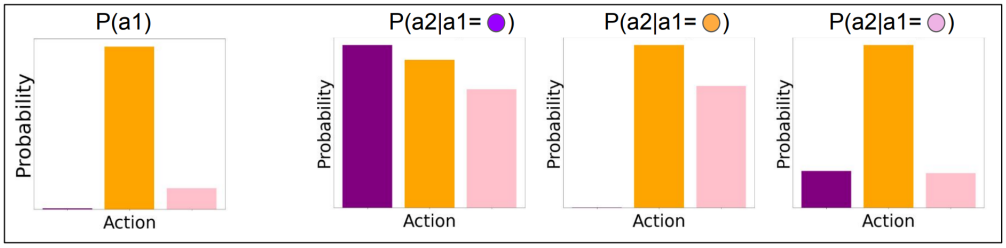
Naive/pedagogical teacher implementation. The naive teacher's policy π_T is conditioned on the goal and parametrized by two probability distributions; the probability of selecting the first ball given the goal g: $\mathbb{P}(\text{first ball} = x|g), x \in \{orange, pink, purple\}$, and then the probability of selecting the second ball given the first ball and the goal g: $\mathbb{P}(\text{second ball} = y|\text{first ball} = x, g)$. To get trained, the teacher randomly samples a goal and plays the corresponding policy using the conditional probabilities. The policy is then trained using a simple update rule: if the action sequence (pick the first and second ball) leads to the achievement of the pursued goal, we increase the probability of selecting this action sequence. Otherwise, we decrease this probability. For the pedagogical teacher, we increase even more the probability of an action sequence that reaches the goal and for which the teacher is able to infer the goal.

A.3 Phase 2: Training the Literal/Pragmatic Learner with Teacher's demonstrations

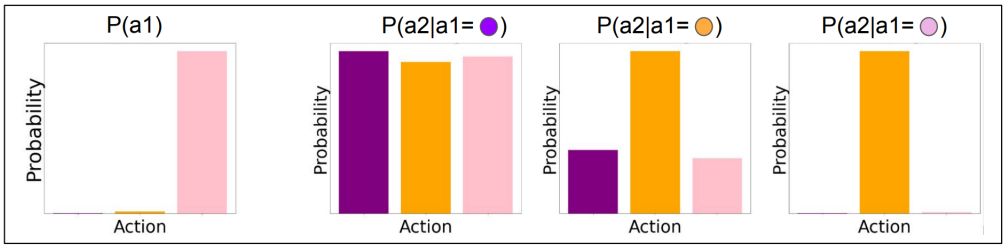
Literal/pragmatic learner implementation. The probability of selecting an action sequence is increased if the goal is correctly predicted and reached. As in FBS, pragmatism is implemented in DTB identically to the pedagogy mechanism of the teacher described above.

A.4 Experiments and results

Phase 1: Qualitatively, what is the difference between a pedagogical and a naive demonstration? In order to answer this question, we analyze the teachers' policies. Fig. 6, presents the teacher policies for goal 1, illustrating the difference between a naive and a pedagogical teacher on DTB environment. Contrary to the naive teacher, the pedagogical teacher specifically avoids



Naive teacher policy



Pedagogical teacher policy

Fig. 6: Analysis of naive vs pedagogical teachers on DTB for goal 1, which can be achieved by selecting either (pink, orange) or (orange, pink). Colored bars represent the probability of selecting the ball: on the left for the first action and on the right for the second action, given the first action. With a pedagogical demonstration, there is no ambiguity regarding the pursued goal: the pedagogical teacher always selects the unambiguous demonstration (pink, orange), while the naive teacher alternatively selects (pink, orange) or (orange, pink).

the demonstrations (orange, orange) for goal 1 because it does not allow to directly detect goal 1 after the first ball is picked. Moreover, it avoids the ambiguous demonstration (orange, pink), which reaches both goal 1 and goal 2. By carefully selecting among all possible demonstrations for a goal, this pedagogical teacher policy maximally avoids ambiguity in demonstrations. Quantitatively, this results in a Own Goal Inference Accuracy (OGIA) of 82% for the naive teacher, and 100% for the pedagogical teacher.

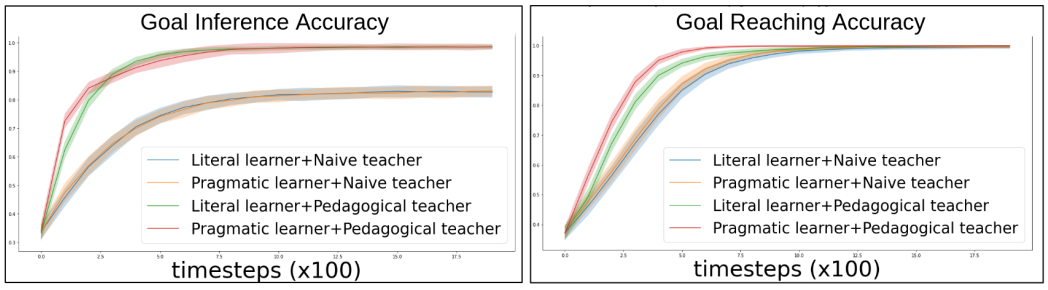


Fig. 7: Left: Goal Inference Accuracy evaluated during training. Right: Goal Reaching accuracy evaluated during training.

Phase 2: What are the benefits and drawbacks of using a pedagogical teacher over a naive teacher? A pragmatic learner over a literal learner? We experiment with pedagogical/naive tutors and pragmatic/literal learners just as in FBS, and present our results on Fig. 7. When a learner is trained with a pedagogical tutor, it consistently learns faster, and can predict the goals from all demonstrations, whereas it is not capable of disambiguating goal 1 from goal 2 with the (orange, pink) demonstration from a naive tutor. Moreover, a learner benefits from pragmatism if the tutor is pedagogical, resulting in the best tutor-learner combination. The pedagogical teacher + pragmatic learner combination reaches a GRA of 100% twice faster than the naive teacher + literal learner combination. Furthermore, the pedagogical+pragmatic combination reaches a GRA X GIA of 1 while

the naive+literal combination only reaches 0.82. The conclusions from these results are thus the same as with FBS. Considering the difference of environments, policy architectures and training processes between the experiments on DTB and FBS, it confirms that pedagogy and pragmatism do generally improve learning from demonstrations.

B Implementation details

B.1 Fetch Block Stacking environment

As in [2], the environment goals are based on two predicates: the close and the above binary predicates. For the 3 objects we consider, these predicates are applied to all permutations of object pairs: 6 permutations for the above predicate and 3 combinations for the close predicate due to its order-invariance. A semantic configuration is the concatenation of these 9 predicates and represents spatial relations between objects in the scene. In the resulting semantic configuration space $\{0,1\}^9$, the agent can reach 35 physically valid configurations, including stacks of 2 or 3 blocks and pyramids.

To compute the reward, the agent in FBS compares its goal configuration to the current configuration and derives a reward with the following procedure: it adds 1 to the reward for each pair of blocks if the true predicates in the goal configuration match the current configuration. This means that the reward can either be 0, 1 or 3 (2 is not achievable by associativity).

FBS is based on Mujoco which is licensed under the Apache License 2.0.

B.2 Architecture, training, and hyperparameters for the teacher and learner

B.2.1 Object-centered architecture

Both the teacher and learner share the same architectures, goal-conditioned RL training and hyperparameters. They are all based on the GANGSTR agent [3] which is a graph-based goal-conditioned RL agent capable of learning to master all goals on the Fetch Block Stacking environment.

The GANGSTR agent perceives the low-level geometric states and the high-level semantic configurations. Following [3], we encode both object-centered and relational inductive biases in our architecture. We model both the agents policies and critics as Message Passing Graph Neural Networks [19]. We consider a graph of 3 nodes, each representing a single object. All the nodes are interconnected. We consider the agent's body attributes as global features of the policy networks and both the agent's body attributes and the actions as global features of the critic networks. A single forward pass through this graph consists in three steps:

- Message computation is performed for each edge.
- Node-wise aggregation is performed for each node.
- Graph-wise aggregation is performed once for all the graph.

We use max pooling for the node-wise aggregation and summation for the graph-wise aggregation.

The object-centered architecture uses two shared networks, NN_{edge} and NN_{node}, respectively for the message computation and node-wise aggregation. Both are 1-hidden-layer networks of hidden size 256. Taking the output dimension to be equal to 3x the input dimension for the shared networks showed the best results. All networks use ReLU activations and the Xavier initialization. We use the Adam optimizer [31], with a learning rate 10^{-3}. The list of hyperparameters is provided in Table 4.

B.2.2 Goal-conditioned RL training

The RL training procedure relies on SAC [24] for the RL and HER [4] for goal relabelling. It uses the Message Passing Interface [12] to exploit multiple processors. Each of the 24 parallel workers maintains its own replay buffer of size 10^6 and performs its own updates. Updates are summed over the 24 actors and the updated actor and critic networks are broadcast to all workers. Each worker alternates between 10 episodes of data collection and 30 updates with batch size 256. To form an epoch, this cycle is repeated 50 times and followed by the offline evaluation of the agent. Each agent is trained for 100 epochs, totalling $24 * 10^6$ timesteps. Each training is performed 5 times with 5

Table 4: Hyperparameters.

Hyperparam.	Description	Values.
nb_mpis	Number of workers	24
nb_cycles	Number of repeated cycles per epoch	50
$nb\_rollouts\_per\_mpi$	Number of rollouts per worker	10
$rollouts\_length$	Number of episode steps per rollout	40
$nb\_updates$	Number of updates per cycle	30
$replay\_strategy$	HER replay strategy	$final$
k_replay	Ratio of HER data to data from normal experience	4
$batch\_size$	Size of the batch during updates	256
γ	Discount factor to model uncertainty about future decisions	0.99
τ	Polyak coefficient for target critics smoothing	0.95
lr_actor	Actor learning rate	10^{-3}
lr_critic	Critic learning rate	10^{-3}
α	Entropy coefficient used in SAC	0.2

random seeds and results report error bars (standard deviation). We used a large GPU cluster and, in total, the experiments required roughly 6000 GPU hours.

As for the particular case of the learner, we provide additional details:

- Our implementation of SQIL [37] uses 50% experience and 50% demonstrations in the replay buffer.
- To generate demonstrations, the teacher randomly selects a starting state and makes sure that the goal is achieved.
- When training the learner, the goal demonstrated by the teacher is selected randomly among the goals discovered by the learner.

Regarding pedagogy and pragmatism, the extra reward ("pedagogical reward" and "pragmatic reward") is set to 1.

B.2.3 BGI implementation details

Computation with continuous action policy. The BGI computation is performed using the policy of the agent. In the case of Fetch Block Stacking and SAC, the actions are continuous. The policy outputs a goal-conditioned normal distribution with as many dimensions as the dimension of the action space. This distribution is sampled for goal-conditioned action selection. Given a vector of actions and the goal-conditioned normal distribution of actions of a policy, one can compute the probability of observing such action given each goal, and construct a probability distribution over the goal space of observing such action. This is done using the cumulative distribution function of the Normal distribution (probability of observing a value given the parameters of the distribution).

We generalize this computation to a trajectory rather than a single action by taking the product of probabilities over the goal space and normalizing to a probability distribution.

Goal Prediction Neural Network details. In Sec.5.4, we experiment with a Goal Prediction Neural Network that is trained offline to predict goals from demonstrations using a training dataset of demonstration provided by the teacher, in order to provide a comparison to BGI inference. This neural network takes a demonstration as input to a LSTM [30] layer with 512 hidden units, from which the output is fed to a fully connected layer. The final outputs are goal probabilities. The predicted goal of the demonstration is the one with the higher probability. The activation functions are ReLU. The GPNN is trained until convergence (300 epochs) and then tested, just like BGI, on a separate test set of 500 demonstrations, from which we report the results in Tab.2. We use a batch size of 256 and Adam [31] as the optimizer, with a learning rate 10^{-3}.

B.2.4 Baselines of learning from demonstrations

Baseline 1 (B1). For this baseline, we discard the experience collected by the learner in the replay buffer and only use the demonstrations provided by the teacher to perform the same learning procedure as the main experiment in Sec.5.2.

Baseline 2 (B2). In this experiment, we use the same learning procedure as in the main experiment in Sec.5.2, but additionally perform a L2 regularization on the output action probabilities by using the demonstrations as a target. This is done using a Mean Squared Error loss and Adam optimizer with a learning rate of 10^{-3} and a batch size of 256.

Baseline 3 (B3). This baseline is the original version of SQIL [37], please refer to their paper for implementation details.

B.2.5 Ambiguity Score details

We provide additional details about the Ambiguity Score and its computation. We manually created a list of situations from which ambiguity in demonstrations exists. These situations are composed of a starting state (the initial state of relations between blocks before the demonstration begins), and two potentially ambiguous goals. The teacher then performs a demonstration for each of the two goals, with the same initial state. If both demonstrations achieve the same goal (even though they were aiming for different goals), then they are considered ambiguous.

The list of ambiguous situations we use to compute the results in the paper is the following:

- Initial state: green is close to red, blue further apart. Ambiguous goals: green is close to red + blue is close to green and green is close to blue + red further apart.
- Initial state: green is close to red, blue further apart. Ambiguous goals: green is close to red + blue is above green and blue is above green + red further apart.
- Initial state: green is above red, blue further apart. Ambiguous goals: green is above red + blue is close to red and blue is close to green + green further apart.
- Initial state: green is above red, blue further apart. Ambiguous goals: green is above red + blue is close to red and blue is close to red + green further apart.
- Initial state: green is above red, blue further apart. Ambiguous goals: green is above red + blue is close to green and blue is close to green + red further apart.
- Initial state: green is close to red, blue further apart. Ambiguous goals: green is close to red + blue is above green and red in a pyramid and blue is close to red and green + green is close to red.

Note that these ambiguous situations are augmented with all possible permutations of block colors.