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 goals. 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.

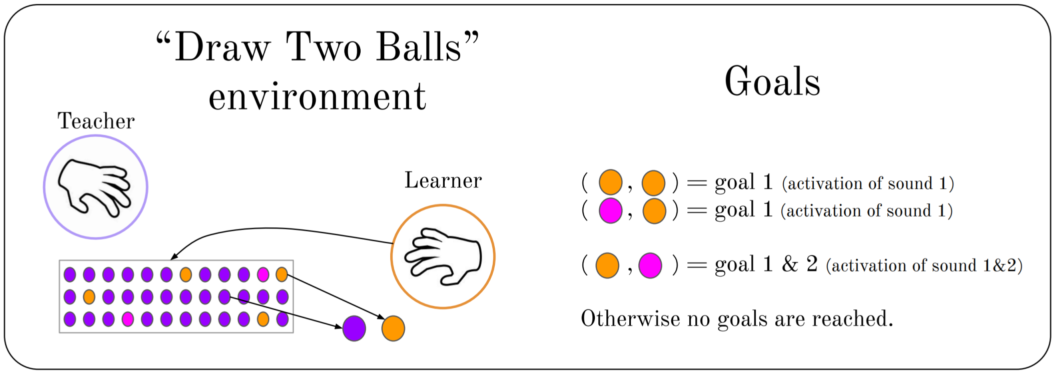

"Draw Two Balls" environment
Teacher
Learner
Goals
(● , ●) = goal 1 (activation of sound 1)
(● , ●) = goal 1 (activation of sound 1)
(● , ●) = goal 1 & 2 (activation of sound 1&2)
Otherwise no goals are reached.

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

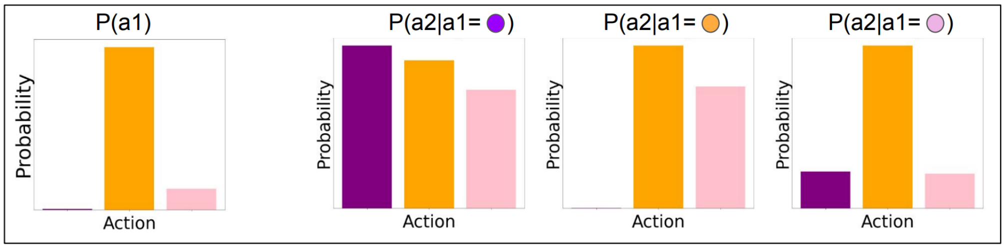
Naive/pedagogical teacher implementation. The naive teacher's policy π_T is conditioned on the goal and parametrized by two probability distributions; the probability of selecting the first ball given the goal g: $\mathbb{P}(\text{first ball} = x|g), x \in \{orange, pink, purple\}$, and then the probability 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

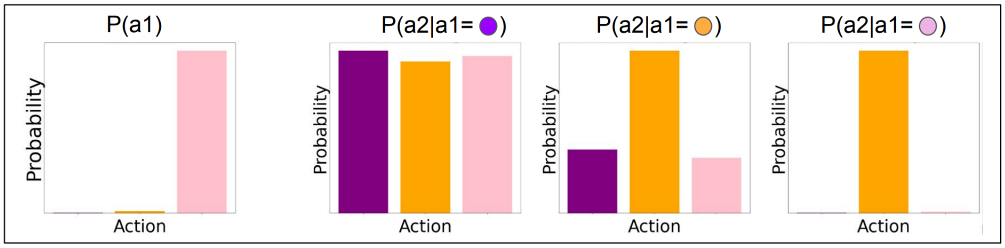
Literal/pragmatic learner implementation. The probability of selecting an action sequence is increased if the goal is correctly predicted and reached. As in FBS, pragmatism is implemented in DTB identically to the pedagogy mechanism of the teacher described above.

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 the demonstrations (orange,



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).

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.

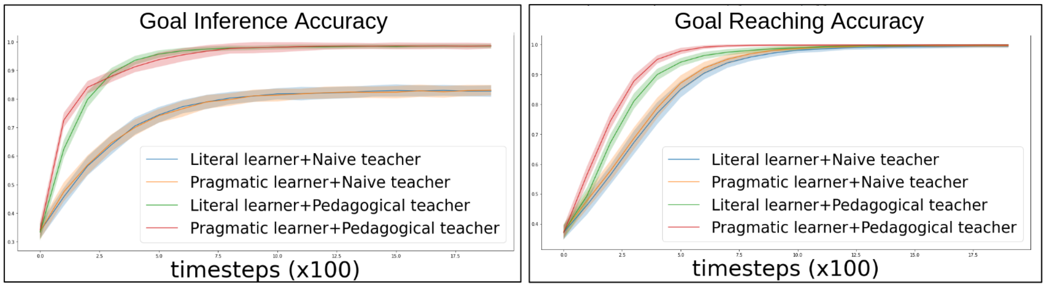


Fig. 7: Left: Goal Inference Accuracy evaluated during training. Right: Goal Reaching accuracy evaluated during training.

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. 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 Hyperparameters

B.2.3 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 10 times with 10 random seeds and results report error bars (standard deviation).

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.

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

- 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.2.4 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.2.5 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.2.6 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.

B.2.7 Additional results with less than 100 demonstrations per goal

The results with 10 demonstrations per goal show that none of the approaches are able to master all goals. However, the best combination is still a pedagogical teacher with a pragmatic learner, as Fig. 8 shows.

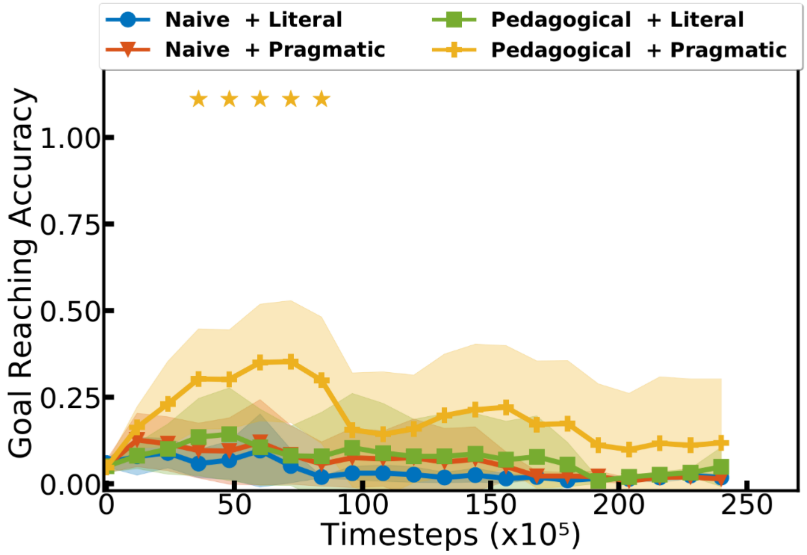


Fig. 8: Results for FBS environment (Goal Reaching Accuracy (GRA) with 10 demonstrations per goal). Stars indicate significance (tested against naive+literal).