International Conference on Information, Communication and Automation Technologies (ICAT 2025)

PARTNR: Pick and place Ambiguity Resolving by Trustworthy iNteractive leaRning

Jelle Luijkx[1], Zlatan Ajanović[1*], Laura Ferranti[1], Jens Kober[1]
[1]Department of Cognitive Robotics, Delft University of Technology, The Netherlands
{J.D.Luijkx, Z.Ajanovic, L.Ferranti, J.Kober}@tudelft.nl
partnr-learn.github.io
[*]Affiliated with TU Delft at the time this work was conducted.

***Abstract*—Several recent works show impressive results in mapping language-based human commands and image scene observations to direct robot executable policies (e.g., pick and place poses). However, these approaches do not consider the uncertainty of the trained policy and simply always execute actions that are suggested by the current policy as the most probable ones. This makes them vulnerable to domain shift and inefficient in the number of required demonstrations. We extend previous works and present the PARTNR algorithm that can detect ambiguities in the trained policy by analyzing multiple modes in the probability distributio of pick and place poses using topological analysis. In this way uncertainty in action can be estimated with single inference (and training single model) instead of using ensemble of models. Additionally, PARTNR employs an adaptive, sensitivity-based, gating function that decides if additional user demonstrations are required. User demonstrations are aggregated to the dataset and used for subsequent training. In this way, the policy can adapt promptly to domain shift and it can minimize the number of required demonstrations for a well-trained policy. The adaptive threshold enables to achieve the user-acceptable level of ambiguity to execute the policy autonomously and in turn, increase the trustworthiness of our system. We demonstrate the performance of PARTNR in a table-top pick and place task.**

I. Introduction

Despite the numerous exciting results in robot learning, only a few methods are actually robust enough to be employed in everyday life. Many manipulation tasks, such as pick-and-place in household scenarios, are challenging for robots, while they are actually easy for humans. To overcome this performance mismatch, we can exploit the human domain knowledge through (interactive) imitation learning [1]. This requires novel methods with an intuitive interface to transfer non-expert user knowledge to robotic systems. The impressive capabilities of recently introduced foundation models can possibly ease this transfer of knowledge. Foundation models can be trained on language data only (e.g., Transformers [2], BERT [3], or GPT-3 [4]) or can be trained on multi-modal data, such as images and their captions (e.g., CLIP [5]). In the field of robotics, language foundation models can be used for task planning [6] and interpreting human commands [7], [8] as well as corrections [9]. In particular, in the setting of (interactive) imitation learning, it is a natural choice to exploit language foundation models, since it allows the user to give instructions or corrections in an intuitive manner. Interactive imitation learning is a subclass of imitation learning in which the human is influencing the learning loop while executing the task [1]. To be practical and trustworthy, the robot should ask for help when it is uncertain about the outcome of its actions. At the same time, humans should not be bothered too much. In this work, we address this problem by introducing Pick and place Ambiguity Resolving by Trustworthy iNteractive leaRning (PARTNR). Our work is related to the seminal work that introduced dataset aggregation (DAgger) for imitation learning [10]. DAgger addresses the compounding errors in imitation learning caused by covariate shift through the collection of on-policy data. Many variants were introduced afterward, such as Human-Gated DAgger (HG-DAgger) [11], where the expert can take over control if deemed necessary, and Ensemble-DAgger[12], where the robot queries expert input when a novel or risky situation is faced. Regarding the ambiguity resolution, our work is related to LIRA [13] which treats ambiguities in discrete reference frames, while here we focus on actions. PARTNR can be used to interactively train vision-based pick and place models, such as Transporter networks [14] and its extension for language commands CLIPort [7]. PARTNR asks for a human demonstration in case the model predictions are ambiguous. We consider a prediction to be ambiguous if it results in multiple options with similar value estimates. To be trustworthy, the threshold of the gating function is adaptive and allows it to satisfy a user-defined sensitivity, balancing between potential failing and asking the user unnecessarily.

PARTNR consists of two main steps: 1) Detecting ambiguity in pick and place heatmaps by finding multiple local maxima using topological persistence and query user demonstrations if needed. 2) Aggregating data from new human demonstrations in DAgger style to learn from feedback and resolve the ambiguity. PARTNR has several advantageous features compared to the other state-of-the-art approaches: 1) By querying a new demonstration based on the level of ambiguity, it avoids gathering demonstrations for situations that are already learned by the agent, therefore reducing the number of required demonstrations. 2) By specifying the desired sensitivity level, the user can set its preferred balance between the frequency of queries by the robot and the failure rate, therefore increasing the system's trustworthiness. 3) By gathering data during execution, PARTNR can adapt to changing environments as well as include *failure states*, i.e., new states visited by making mistakes, in the

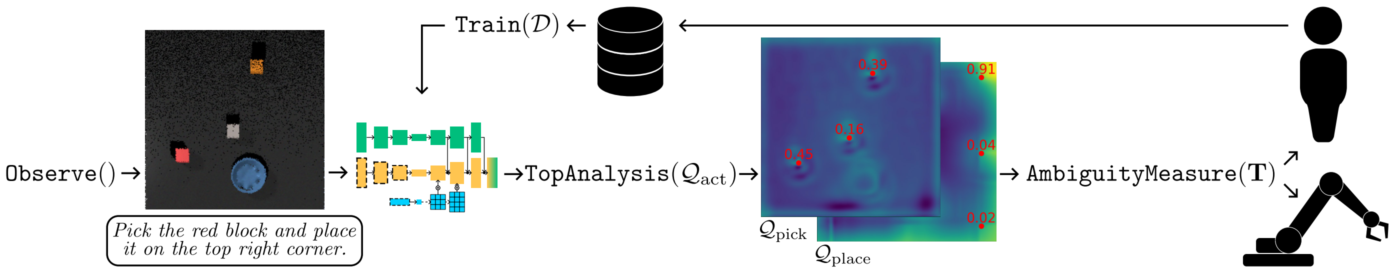
Train($\mathcal{D}$)
Observe()
Pick the red block and place it on the top right corner.
TopAnalysis($\mathcal{Q}_{\text{act}}$)
0.39
0.16
0.45
0.91
0.04
0.02
$\mathcal{Q}_{\text{pick}}$
$\mathcal{Q}_{\text{place}}$
AmbiguityMeasure($\mathbf{T}$)

Fig. 1: PARTNR framework on an example task.

dataset so it can learn to recover from them. We demonstrate these on a simulated table-top robot pick and place task, where we show an improvement of the performance with respect to the baseline (CLIPort variant).

The rest of the paper is organized as follows. Section II presents preliminaries as seen in the works [14], [7] and lays the formal problem formulation for our work. Section III presents our method. This is followed by experiments and conclusion sections.

Additional material is available at: partnr-learn.github.io.

II. Preliminaries and Problem Definition

We follow [14], [7] and describe the pick and place problem as finding a mapping from an observation $\mathbf{o}_t$ to a pick and place action $\mathbf{a}_t$ at time step t, that is, $f(\mathbf{o}_t) \rightarrow \mathbf{a}_t = (\mathcal{T}_{\text{pick}}, \mathcal{T}_{\text{place}}) \in \mathcal{A}$, where $\mathcal{A}$ is the set of possible actions and $\mathcal{T}_{\text{pick}}$ and $\mathcal{T}_{\text{place}}$ are the end-effector pick and place poses, respectively. We consider a table-top pick and place problem with $\mathcal{T}_{\text{pick}}, \mathcal{T}_{\text{place}} \in \mathbb{R}^2$. Motion primitives can be used to obtain a sequence of lower-level actions from $\mathcal{T}_{\text{pick}}$ and $\mathcal{T}_{\text{place}}$. Furthermore, we consider vision-based manipulation with $\mathcal{T}_i \sim (u,v), i \in \{\text{pick}, \text{place}\}$, where (u,v) is a pixel location of a (projected) top view image. If we model the action-value functions $\mathcal{Q}_{\text{pick}}$ and $\mathcal{Q}_{\text{place}}$, the optimal pick and place locations according to the model are $\mathcal{T}_{\text{pick}} = \arg\max_{(u,v)} \mathcal{Q}_{\text{pick}}((u,v)|\mathbf{o}_t)$ and $\mathcal{T}_{\text{place}} = \arg\max_{(u,v)} \mathcal{Q}_{\text{place}}((u,v)|\mathbf{o}_t, \mathcal{T}_{\text{pick}})$. Note that $\mathcal{T}_{\text{place}}$ is conditioned on $\mathcal{T}_{\text{pick}}$. Normalized heatmaps correlated with pick and place success can be obtained using the $\mathrm{softmax}$ function, i.e., $\mathcal{V}_{\text{pick}} \in \mathbb{R}^{H \times W} = \mathrm{softmax}(\mathcal{Q}_{\text{pick}}((u,v)|\mathbf{o}_t))$, where H and W are the height and width of the top view image, respectively. The action-value functions can be estimated through imitation learning. We build on the standard imitation learning setting where we have a dataset $\mathcal{D} = \{\zeta_1, \zeta_2, ..., \zeta_n\}$, where n is the number of expert demonstration trajectories consisting of one or more tuples of observations and actions, i.e., $\zeta_i = \{(\mathbf{o}_0, \mathbf{a}_0), (\mathbf{o}_1, \mathbf{a}_1), ...\}$.

In this work, we extend previous problem formulation and consider situations when taking the most probable action by $\arg\max$ is not sufficient, e.g. when there is no single distinctive maximum. We tackle the interactive learning problem where the robot needs to hand over the control back to the human and learn from new human demonstrations.

III. The Method

PARTNR is an interactive imitation learning algorithm that asks the human to take over control in case it considers the situation to be ambiguous. We consider the situation to be ambiguous when the learned policy does not provide a single dominant solution, i.e., there are multiple local maxima with close values in the action space. User demonstrations are aggregated to the dataset $\mathcal{D}$ and used for subsequent training, as shown in Algorithm 1. Figure 1 shows the PARTNR framework in an example where a human asks the robot to pick the red block and place it in the top right corner. As we can see on the heatmaps for pick and place, there are multiple local maxima for this command. In the pick heatmap $\mathcal{Q}_{\text{pick}}$ there are at least three local maxima, each related to one of the blocks. The maximum related to the red block is the highest (0.45). However, the orange block is also relatively close (0.39). In this case, the situation might be ambiguous (depending on the sensitivity level) and the robot might query the teacher for a demonstration.

A. Topological Analysis Ambiguity measure

The robot observes, at each execution step, a human-provided natural language command and the state of the environment (e.g., a top-view image of the table). Based on the observation, the policy provides the heatmap, representing the value of the action (e.g., $\mathcal{Q}_{\text{pick}}$ representing pick location). The heatmap ($\mathcal{Q}_{\text{pick}}$ and subsequently $\mathcal{Q}_{\text{place}}$) is then analyzed to detect multiple local maxima (in `TopAnalysis`). In this work, we rely on computational topology methods for finding local maxima [15]. Specifically we use a persistent homology method [16]. Then, in `AmbiguityMeasure`, the obtained corresponding values of the local maxima $\mathbf{T}$, are normalized using the $\mathrm{softmax}$ function and the maximum value $\hat{p}_{\text{act}}$ is then used to decide if the situation is ambiguous. If $\hat{p}_{\text{act}}$ is smaller than the threshold $p_{\text{act}}^{\text{thr}}$, the situation is ambiguous. In case the situation is ambiguous, the robot is not executing the policy but queries the human teacher.

Figure Figure 2 shows a visual example of how the ambiguity measure is obtained. The input image together with the correct action (green arrow) is shown in Figure Figure 2 **(a)**. Here, the language command is: '*Pick the red block and place it on the top right corner*". With `TopAnalysis`, we obtain local

```
Algorithm 1: PARTNR - detailed algorithm
input  : $\mathcal{D}^{\text{init}}$                      // initial demonstrations
output : $\mathcal{Q}_{\text{pick}}, \mathcal{Q}_{\text{place}}$          // pick and place value
         functions
$\mathcal{D} \leftarrow \mathcal{D}^{\text{init}}$                          // initial Dataset
$\mathcal{Q}_{\text{pick}}, \mathcal{Q}_{\text{place}} \leftarrow$ Train($\mathcal{D}^{\text{init}}$)
TP, TN, FP, FN $\leftarrow \varnothing$
$p^{\text{thr}}_{\text{pick}}, p^{\text{thr}}_{\text{place}} \leftarrow$ init()
for $t \leftarrow 0$ to $t_{\max}$ do // while experiment runs
    $\mathbf{o}_t \leftarrow$ Observe()
    foreach act $\in$ {pick, place} do
        isUpdated $\leftarrow$ false
        $\mathbf{T} = \{(u_1, v_1), \ldots, (u_k, v_k)\} \leftarrow$
         TopAnalysis($\mathcal{Q}_{\text{act}}((u, v)|\mathbf{o}_t)$)
        $\mathbf{a}_{\max} \leftarrow \arg\max_{(u,v)\in\mathbf{T}} \mathcal{Q}_{\text{act}}(u, v)$
        $\hat{p}_{\text{act}} \leftarrow$ AmbiguityMeasure($\mathbf{T}$)
        if $\hat{p}_{\text{act}} \leq p^{\text{thr}}_{\text{act}}$ then // if ambiguous
            $\mathbf{a}_t \leftarrow$ QueryTeacher($\mathbf{T}$)
            $\mathcal{D} \leftarrow \mathcal{D} \cup (\mathbf{o}_t, \mathbf{a}_t)$  // adding user input to
             the Dataset
            isUpdated $\leftarrow$ true
            if $\mathbf{a}_t == \mathbf{a}_{\max}$ then
                FP $\leftarrow$ FP $\cup\ t$           // adding False
                 Positive flag
            else
                TP $\leftarrow$ TP $\cup\ t$         // adding True
                 Positive flag
            Act($\mathbf{a}_t$)
        else // if not ambiguous
            Act($\mathbf{a}_{\max}$)
            if $\mathbf{a}_{\text{corr}} \leftarrow$ ObserveCorrection() $\neq \varnothing$ then
             // if teacher corrects
                $\mathcal{D} \leftarrow \mathcal{D} \cup (\mathbf{o}_t, \mathbf{a}_{\text{corr}})$
                isUpdated $\leftarrow$ true
                FN $\leftarrow$ FN $\cup\ t$           // adding False
                 Negative flag
            else
                TN $\leftarrow$ TN $\cup\ t$  // adding True Negative
                 flag
        $p^{\text{thr}}_{\text{act}} \leftarrow$ UpdateThreshold($p^{\text{thr}}_{\text{act}}$, TP, TN, FP, FN)
        if isUpdated then
            $\mathcal{Q}_{\text{act}} \leftarrow$ Train($\mathcal{D}$)      // update the model
             with new data
```

maxima $\mathbf{T}$, which are shown in Figure Figure 2 **(b)** and Figure Figure 2 **(c)** for the pick and place poses, respectively. The corresponding values are shown in Figure Figure 2 **(d)**. After normalization using the softmax function, we obtain Figure Figure 2 **(e)**. The local maxima with a normalized value greater than 0.01 are shown in Figure Figure 2 **(f)** and Figure Figure 2 **(g)** for the pick and place poses, respectively. The maximum of the normalized values is used as ambiguity measure.

B. Adaptive Threshold

The threshold $p^{\text{thr}}_{\text{act}}$ is updated continuously, by the function UpdateThreshold, to satisfy a user-defined sensitivity value (more details in Section III-B). Whenever there is a teacher input, the data is aggregated and the policy is updated using the function Train (like in [10]). A detailed version of the

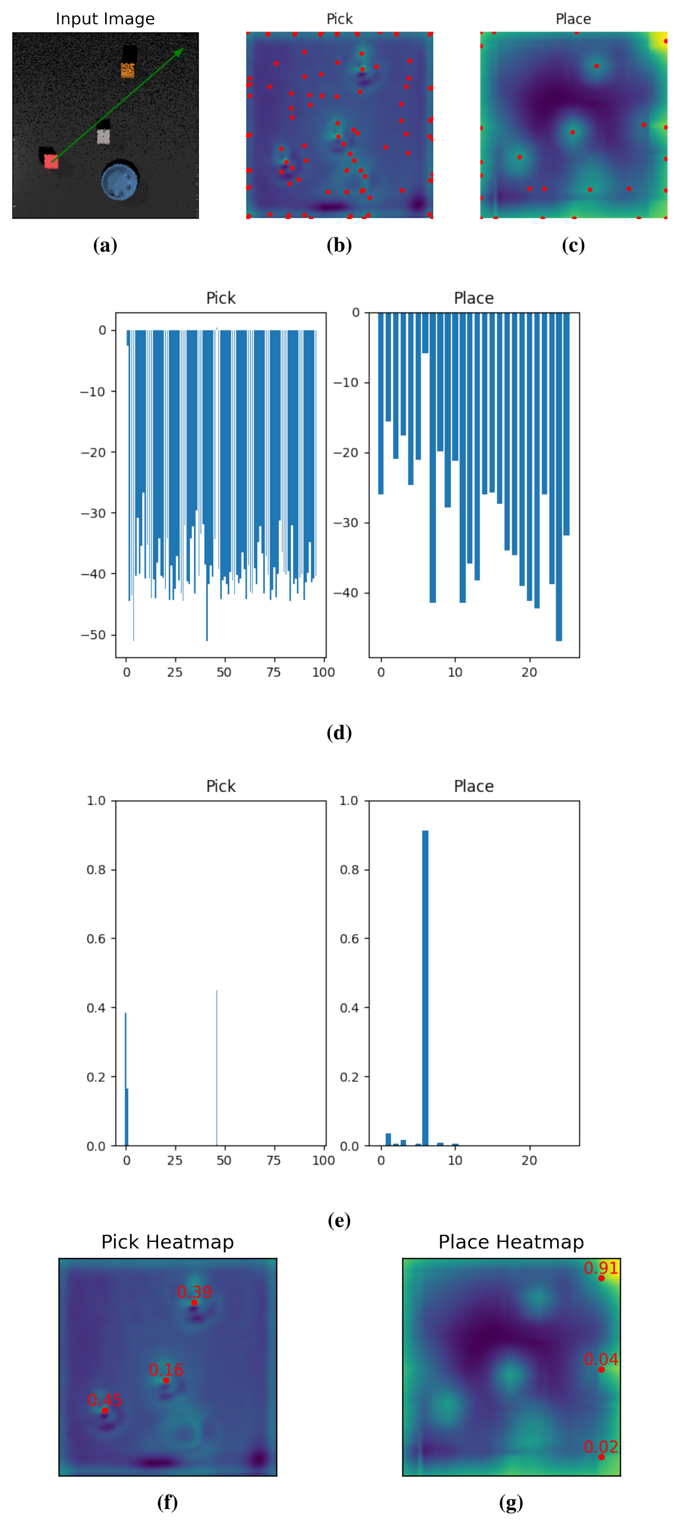

Fig. 2: Visual example of obtaining the ambiguity measure. The input image together with the correct action (green arrow) is shown in **(a)**. Here, the language command is: '*Pick the red block and place it on the top right corner*". With TopAnalysis, we obtain local maxima $\mathbf{T}$, which are shown in **(b)** and Figure Figure 2 **(c)** for the pick and place poses, respectively. The corresponding values are shown in **(d)**. After normalization using the softmax function, we obtain **(e)**. The local maxima with a normalized value greater than 0.01 are shown in **(f)** and **(g)** for the pick and place poses, respectively. The maximum of the normalized values is used as ambiguity measure.

Algorithm 2: `UpdateThreshold`

```
input : Initial threshold $p_0^{\text{thr}}$, Desired sensitivity $s_{\text{des}}$,
        Window length $w_n$, Adaptation rate $a$.
output: threshold $p^{\text{thr}}$
1 begin
2     $k_{\text{TP}}, k_{\text{TN}}, k_{\text{FP}}, k_{\text{FN}} \leftarrow$
        MovHorCnt($w_n$, TP, TN, FP, FN)
        // Counting occurrence in the window $w_n$
3     $\hat{s} \leftarrow \frac{k_{\text{TP}}}{k_{\text{TP}}+k_{\text{FN}}}$
4     $p^{\text{thr}} \leftarrow p_0^{\text{thr}} - a \cdot (s_{\text{des}} - \hat{s})$
```

adaptive threshold algorithm is shown in Algorithm 2. Here, the number of true positives, true negatives, false positives, and false negatives are counted over a window ($k_{\text{TP}}, k_{\text{TN}}, k_{\text{FP}}, k_{\text{FN}}$, respectively). Subsequently, the sensitivity can be estimated ($\hat{s}$) [17]. The definition of positives and negatives is shown in Table I.

Finally, the threshold p^{thr} is updated proportionally to the error between the desired and estimated sensitivity. In our experiments, we used the following values: $p_0^{\text{thr}} = 0.5$, $s_{\text{des}} = 0.9$, $w_n = 50$ and $a = 0.005$.

IV. Experiments and Results

We evaluated the performance of the proposed method in a simulated table-top pick and place task, which is shown in Figure 3. This task is very similar to tasks from [7], [14], [18]. The goal of this task is to execute language commands in the form "*Pick the [pick color] box and place it in the [place color] bowl.*", where the pick and place colors are sampled at the beginning of an episode, based on the colors of the objects present in the scene. The task is simulated using the PyBullet simulator [19] and the implementation is adapted from [18]. At the beginning of each episode, three boxes and three bowls are placed at random locations on the table. Similar to [7], the colors of the objects are either sampled from the color set of $\mathcal{C}_{\text{all}} \cup \mathcal{C}_{\text{seen}}$ or the color set $\mathcal{C}_{\text{all}} \cup \mathcal{C}_{\text{unseen}}$, where $\mathcal{C}_{\text{all}} = \{$red, blue, green$\}$, $\mathcal{C}_{\text{seen}} = \{$yellow, brown, gray, cyan$\}$, and $\mathcal{C}_{\text{unseen}} = \{$orange, purple, pink, white$\}$. The set of seen colors is used for offline training, while both sets are used for evaluation and interactive learning, to simulate a domain shift.

As a baseline, the CLIPort variant used in [18], [6] is employed and trained on a dataset consisting of demonstrations from a scripted expert. We also trained CLIPort models using the PARTNR algorithm with the same architecture as the baseline interactively, as described in Algorithm 1. The interactive models are initially trained offline and updated interactively while executing the task. To have a fair comparison between the baseline and the interactive models, both have the same number of total demonstrations and a total number of model updates. Since real-life demonstrations are never perfect, we also evaluated the method with noisy demonstrations.

We follow [7] and evaluate each model in 100 episodes consisting of three pick-and-place commands. The percentage of successfully performed pick and place commands is used as evaluation metric. The results in Table II show that the PARTNR algorithm improves the baseline performance, both in the in-distribution and out-of-distribution scenarios (seen and unseen case). The improvement in the unseen case indicates that by collecting on-policy data, the methods improves robustness against domain shifts. Because the PARTNR algorithm collects data from the state distribution induced by the novice policy, it can learn to recover after mistakes, while this is not the case when learning offline from expert demonstrations. That is to say, the expert does not make any mistakes and therefore failure states are not visited by the expert policy. An example of such a recovery learned interactively is shown in Figure 4.

Interestingly, both the baseline and PARTNR performance improved substantially when adding noise to the pick and place demonstrations from the scripted expert. To be noted, the final performance is lower than obtained with the original CLIPort model in [7]. Most likely, this is due to the usage of a simplified variant, a lower number of data augmentations and a lower number of camera perspectives. However, this is not relevant as the main focus here is to make a comparison against a non-interactive baseline, and not to obtain optimal performance.

V. Conclusions and Outlook

This work introduced PARTNR, an interactive learning algorithm for resolving ambiguities in pick-and-place tasks. The PARTNR algorithm improves the baseline performance, both in the in-distribution and out-of-distribution scenarios. Furthermore, sampling efficiency is improved (even up to 20% more data-efficient), since demonstrations are only collected

TABLE I: Definition of positives and negatives. In the ideal case, the teacher is only queried in the case that the robot's action would result in a failure (true positive).

		Human input was necessary	
		True	False
Ambiguous	True	True Positive (TP)	False Positive (FP)
	False	False Negative (FN)	True Negative (TN)

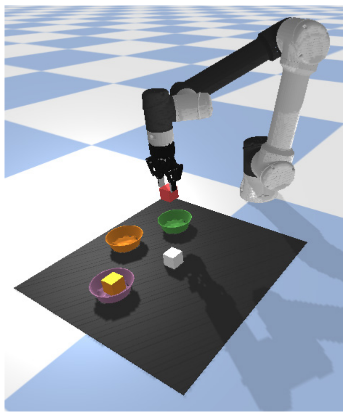

Fig. 3: The *put-blocks-in-bowls* task. Given a language command and observation of the scene, the robot needs to pick the block of a given color and place it in the bowl of a given color.

		put-blocks-in-bowls seen-colors				put-blocks-in-bowls unseen-colors		
algorithm	data split	500	1000	1000 noisy	1500	500	1000	1500
Baseline	100% off	28.3	51.7	82.7	62.7	19.0	22.0	16.7
PARTNR	50% off + 50% int	**30.3**	**57.3**	**91.0**	**80.3**	**30.7**	**53.0**	**78.3**
PARTNR (80% data)	50% off + 30% int	28.0	39.3	77.7	68.0	20.3	28.3	57.3

TABLE II: The performance of the PARTNR algorithm is evaluated against the performance of the non-interactive baseline (CLIPort variant). Here the success rate (%) is shown for a number of demonstrations, i.e., 500, 1000 and 1500. The data split indicates the percentage of demonstrations that were obtained offline (off) and interactively (int), so in the last row, 20% fewer demonstrations were collected. Since real demonstrations are often noisy, we also evaluated both methods with noise ($\sim \mathcal{N}(0, 3^2)$ pixels) added to the pick and place locations (1000 noisy).

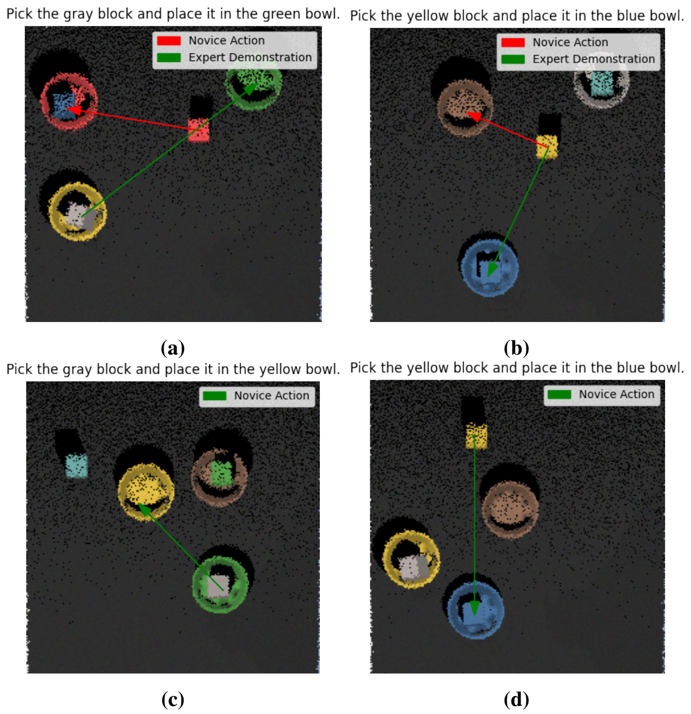
Pick the gray block and place it in the green bowl.
Novice Action
Expert Demonstration
(a)
Pick the yellow block and place it in the blue bowl.
Novice Action
Expert Demonstration
(b)
Pick the gray block and place it in the yellow bowl.
Novice Action
(c)
Pick the yellow block and place it in the blue bowl.
Novice Action
(d)

Fig. 4: Example of learning how to recover thanks to on-policy data collection. The figures **(a)** and **(b)** show demonstrations for failure states, i.e., in **(a)** the block to be picked is already in another bowl, and in **(b)** there is already a block in the blue bowl. Such demonstrations of failure states are collected only in the interactive case. Figures **(c)** and **(d)** show that the novice can learn to recover from such failure states.

when needed, based on the user-specified sensitivity. In the future, we plan to evaluate PARTNR with the original CLIPort baseline as well and to further address the epistemic uncertainty of the model, e.g., through an ensemble approach. Also, we wish to extend the method with sequence prediction and feedback control. Finally, we plan to monitor the human cognitive load in a real-world participant study.

Acknowledgments

This work was supported by the European Union's H2020 project Open Deep Learning Toolkit for Robotics (OpenDR) under grant agreement #871449 and by the ERC Stg TERI, project reference #804907.

References

[1] C. Celemin, R. Pérez-Dattari, E. Chisari, G. Franzese, L. d. S. Rosa, R. Prakash, Z. Ajanović, M. Ferraz, A. Valada, and J. Kober, "Interactive imitation learning in robotics: A survey," *arXiv preprint arXiv:2211.00600*, 2022.

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[4] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901.

[5] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *International Conference on Machine Learning*. PMLR, 2021, pp. 8748–8763.

[6] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, D. Ho, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, E. Jang, R. J. Ruano, K. Jeffrey, S. Jesmonth, N. J. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, K.-H. Lee, S. Levine, Y. Lu, L. Luu, C. Parada, P. Pastor, J. Quiambao, K. Rao, J. Rettinghouse, D. Reyes, P. Sermanet, N. Sievers, C. Tan, A. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, S. Xu, and M. Yan, "Do As I Can, Not As I Say: Grounding Language in Robotic Affordances," 2022.

[7] M. Shridhar, L. Manuelli, and D. Fox, "Cliport: What and where pathways for robotic manipulation," in *Proceedings of the 5th Conference on Robot Learning (CoRL)*, 2021.

[8] H. Ahn, O. Kwon, K. Kim, J. Jeong, H. Jun, H. Lee, D. Lee, and S. Oh, "Visually Grounding Language Instruction for History-Dependent Manipulation," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 675–682.

[9] P. Sharma, B. Sundaralingam, V. Blukis, C. Paxton, T. Hermans, A. Torralba, J. Andreas, and D. Fox, "Correcting robot plans with natural language feedback," *arXiv preprint arXiv:2204.05186*, 2022.

[10] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, G. Gordon, D. Dunson, and M. Dudík, Eds., vol. 15. Fort Lauderdale, FL, USA: PMLR, 11–13 Apr 2011, pp. 627–635. [Online]. Available: https://proceedings.mlr.press/v15/ross11a.html

[11] M. Kelly, C. Sidrane, K. Driggs-Campbell, and M. J. Kochenderfer, "HG-DAgger: Interactive Imitation Learning with Human Experts," in *2019*

International Conference on Robotics and Automation (ICRA), 2019, pp. 8077–8083.

[12] K. Menda, K. Driggs-Campbell, and M. J. Kochenderfer, "EnsembleDAgger: A Bayesian Approach to Safe Imitation Learning," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 5041–5048.

[13] G. Franzese, C. Celemin, and J. Kober, "Learning interactively to resolve ambiguity in reference frame selection," in *Conference on Robot Learning (CoRL)*, 2020.

[14] A. Zeng, P. Florence, J. Tompson, S. Welker, J. Chien, M. Attarian, T. Armstrong, I. Krasin, D. Duong, V. Sindhwani *et al.*, "Transporter networks: Rearranging the visual world for robotic manipulation," in *Conference on Robot Learning*. PMLR, 2021, pp. 726–747.

[15] H. Edelsbrunner and J. Harer, *Computational Topology: An Introduction*. American Mathematical Soc., 2010.

[16] S. Huber, "Persistent homology in data science," in *Data Science–Analytics and Applications*. Springer, 2021, pp. 81–88.

[17] K. Chu, "An introduction to sensitivity, specificity, predictive values and likelihood ratios," *Emergency Medicine*, vol. 11, no. 3, pp. 175–181, 1999.

[18] A. Zeng, M. Attarian, B. Ichter, K. Choromanski, A. Wong, S. Welker, F. Tombari, A. Purohit, M. Ryoo, V. Sindhwani, J. Lee, V. Vanhoucke, and P. Florence, "Socratic models: Composing zero-shot multimodal reasoning with language," *arXiv*, 2022.

[19] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2021.