

# Online Adaptation for Enhancing Imitation Learning Policies

1<sup>st</sup> Federico Malato  
School of Computing  
University of Eastern Finland  
Joensuu, Finland  
federico.malato@uef.fi

2<sup>nd</sup> Ville Hautamäki  
School of Computing  
University of Eastern Finland  
Joensuu, Finland  
ville.hautamaki@uef.fi

**Abstract**—Imitation learning enables autonomous agents to learn from human examples, without the need for a reward signal. Still, if the provided dataset does not encapsulate the task correctly, or when the task is too complex to be modeled, such agents fail to reproduce the expert policy. We propose to recover from these failures through online adaptation. Our approach combines the action proposal coming from a pre-trained policy with relevant experience recorded by an expert. The combination results in an adapted action that closely follows the expert. Our experiments show that an adapted agent performs better than its pure imitation learning counterpart. Notably, adapted agents can achieve reasonable performance even when the base, non-adapted policy catastrophically fails.

**Index Terms**—imitation learning, behavioral cloning, inverse reinforcement learning, online adaptation

## I. INTRODUCTION

Reinforcement learning (RL) [1] and Deep reinforcement learning (DRL) [2] have recently gained momentum as a consequence of notable breakthroughs in policy learning [3]–[5], and following the introduction of reinforcement learning from human feedback (RLHF) [6] for fine-tuning large language models (LLMs). In previous research, RL and DRL have been successfully applied in several domains, ranging from (and not limited to) playing video games [7]–[9], autonomous driving [10], to physics [11].

Despite these incredible results, several challenges remain open [12]. Among those, specifying a reward signal for complex, structured tasks is one of the most prominent. A common approach to address this problem is *imitation learning* (IL) [13], that is, agents learn to act from an expert demonstrating the task, without a reward signal. Perhaps, the most known example of IL algorithm is *behavioral cloning* (BC) [14]. In BC, a policy is learned via supervised learning on the observation-action pairs of the expert dataset. Another notable IL-derived framework is *inverse reinforcement learning* (IRL) [15], [16], where a policy is trained using standard RL on a reward model inferred from the expert trajectories.

While IL policies extend RL agents to scenarios where no reward function can be designed, they also suffer from a number of drawbacks. For instance, BC suffers from distributional shift and causal confusion [14], while IRL is never guaranteed to learn an optimal reward model [15]. A solution to these problems is *adversarial imitation learning*, where a policy

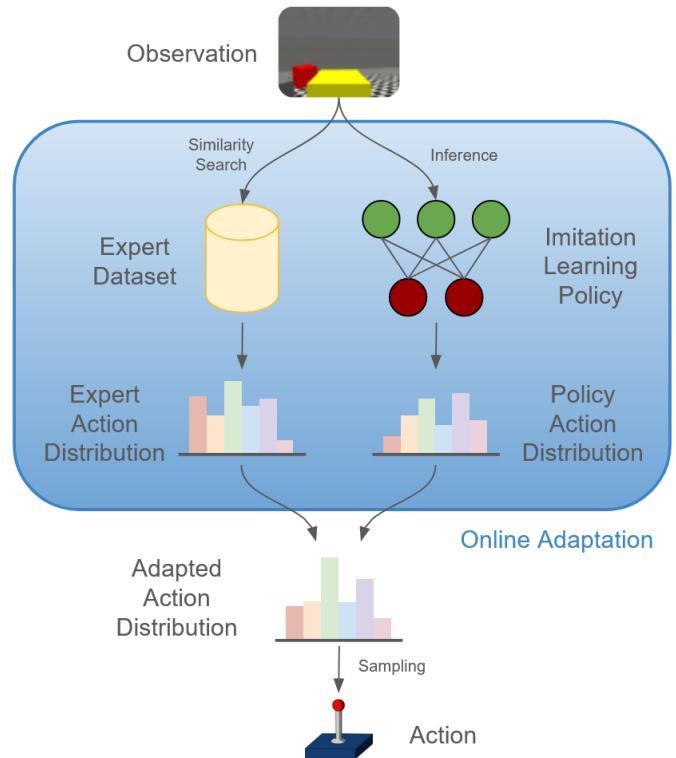


Fig. 1. A visual explanation of our proposed method. At timestep  $t$ , the current observation is fed to the imitation learning policy network to obtain a policy action distribution. Concurrently, we retrieve a number of resembling frames from the expert data and compute an expert action distribution. The two distributions are then combined to obtain a joint, adapted action distribution. Finally, the action is selected by sampling from the joint distribution.

is trained by learning two networks simultaneously in an adversarial fashion. The most notable examples of adversarial algorithms are *generative adversarial imitation learning* (GAIL) [17] and *adversarial inverse reinforcement learning* (AIRL) [18]. GAIL and AIRL take inspiration from generative adversarial networks (GANs) [19]. Specifically, both algorithms train a policy along with a discriminator that tries to distinguish between policy-generated and expert-generated trajectories [17], [18]. AIRL differs from GAIL as it also derives a reward function. Both algorithms have significantly

improved the performance of autonomous agents in a wide range of tasks. Still, adversarial training is unstable and sample inefficient [17]–[19], that is, an adversarial agent might never learn a useful policy, and will typically require very long training procedures.

Additionally, a range of challenges still require a general solution. For instance, autonomous agents should be able to model long-term, causal relationships and plan their actions accordingly [1], [20]. Model-based RL attempts to solve these problems by learning an implicit representation of the environment [20]. More recently, OpenAI proposed Video PreTraining (VPT) [3], a transformer-based [21], causal model to play Minecraft from human demonstrations. VPT was capable of solving long-standing challenges such as the MineRL Diamond challenge [22]. Similarly, recently proposed Dreamer architectures [4], [5] merge several network architectures and combine IL and DRL methods to learn a *world model* to operate informed decisions.

While agents such as VPT and Dreamer indeed achieve outstanding results, they are bounded to use complex architectures [3]–[5] or need massive datasets to be trained successfully [3]. Therefore, their usability in real world scenarios remain uncertain.

In real world use cases, usually gathering data is expensive and demands massive resources [25]–[27]. Hence, autonomous agents should be able to learn from small dataset, while being robust to unpredictable conditions and aligned to human needs. Previous research in this direction have leveraged search as a way to reliably and efficiently select actions [23], [24]. For example, a robotic arm in a low-dimensional domain with continuous action space can average over a set of retrieved relevant actions [23]. In the case of an open-world, high-dimensional visual domain with discrete actions, copying a sequence of actions from relevant past experience of an expert has been proven successful [24]. Despite showing robust performance, such methods lack real time adaptability to unpredictable conditions.

Inspired by this previous research, we propose *Bayesian online adaptation* (BOA), an efficient technique to improve an IL agent action selection process using search, requiring little to none scaling in network complexity. Our approach is explained visually in Fig. 1. BOA leverages Bayesian statistics and search to improve the performance of pure IL agents. Additionally, our method allows a partial explanation of the action selection process, hence improving interpretability of the model.

## II. PRELIMINARIES

*a) Reinforcement Learning:* We model our control problem as a *partially observable Markov decision problem* (POMDP) as a 7-tuple  $(S, A, T, R, \Omega, O, \gamma)$  where  $S \subset \mathbb{R}^d$  is the state space,  $A$  is the action space,  $T : S \times A \rightarrow S$  is the transition dynamics,  $R : S \times A \rightarrow \mathbb{R}$  is the reward function,  $\Omega$  is a set of observations,  $O$  is a set of conditional observation probabilities and  $\gamma \in [0, 1)$  is the discount factor.

*b) Imitation Learning:* In the IL scenario, the reward function  $R$  and the transition dynamics  $T$  are unknown. As such, a policy can not acquire relevant experience by interacting with the environment. Instead, a dataset of observation-action pairs  $\mathcal{D} = \{(o_t, a_t)\}$ ,  $\mathcal{D} \subset S \times A$  with  $t \in [0, \tau]$ ,  $\tau \in \mathbb{N}$  is provided by an expert demonstrating the task. The general aim of an IL policy is to minimize a loss  $\mathcal{L} : A \times A \rightarrow \mathbb{R}$  that describes the difference between the policy predicted actions and the expert actions.

*c) Multinomial Distribution:* The Multinomial distribution is a parametric, discrete distribution characterized by two parameters  $K$  and  $N$ .  $K$  is often referred to as *classes* or *categories*, while  $N$  indicates the number of trials. Given a random variable  $X \sim (N; p_1, \dots, p_K)$ , the distribution has a discrete probability density function of the form

$$\mathbb{P}(X = x) = \frac{N!}{\prod_{i=1}^K x_i!} \prod_{i=1}^K p_i^{x_i} \quad (1)$$

Whenever  $N = 1$ , the Multinomial distribution becomes a *Categorical* distribution.

*d) Dirichlet Distribution:* The Dirichlet distribution is a parametric distribution defined by a scalar parameter  $K$  and a vector parameter  $\alpha$ , called *categories* and *concentration* respectively. The probability density function is

$$f(x; \alpha) = \frac{1}{B(\alpha)} \prod_{i=1}^K x_i^{\alpha_i - 1} \quad (2)$$

where  $B(\alpha)$  is the multivariate beta function.

Among its other properties, the Dirichlet distribution is the *conjugate prior* of the Multinomial distribution. That is, if in Bayesian inference the prior follows a Dirichlet distribution and the likelihood follows Multinomial distribution, then the posterior is known to also follow a Dirichlet distribution.

## III. PROPOSED METHOD

Our method leverages Bayesian inference to update the beliefs of an autonomous agent in real-time. The intuition is as follows: in general, an IL agent tries to imitate the expert’s action distribution, given an observation. It follows that IL works as long as the dataset fully encapsulates the dynamics of a task. Whenever this condition is not satisfied, an IL agent is bound to either fail or show sub-optimal behavior [13], [14]. To mitigate this problem, we provide a learning-based agent with a minibatch of expert solutions for a particular state. Then, we infer the probable action of the expert and update the agent action distribution accordingly.

### A. Search

Similar to *Zero-shot Imitation Policy* (ZIP) [24], as a preliminary step we encode the expert demonstrations dataset using a pre-trained encoder  $h(\cdot)$  inspired from VPT. Details of the encoder we have used are provided in Section VII. For each trajectory  $i$  and for each timestep  $t$ , we pass the expert state  $s_t^{(i)}$  to obtain a latent  $z_t^{(i)} = h(s_t^{(i)})$ . Each latent is then paired with its corresponding action  $a_t^{(i)}$ . Thus, we define the

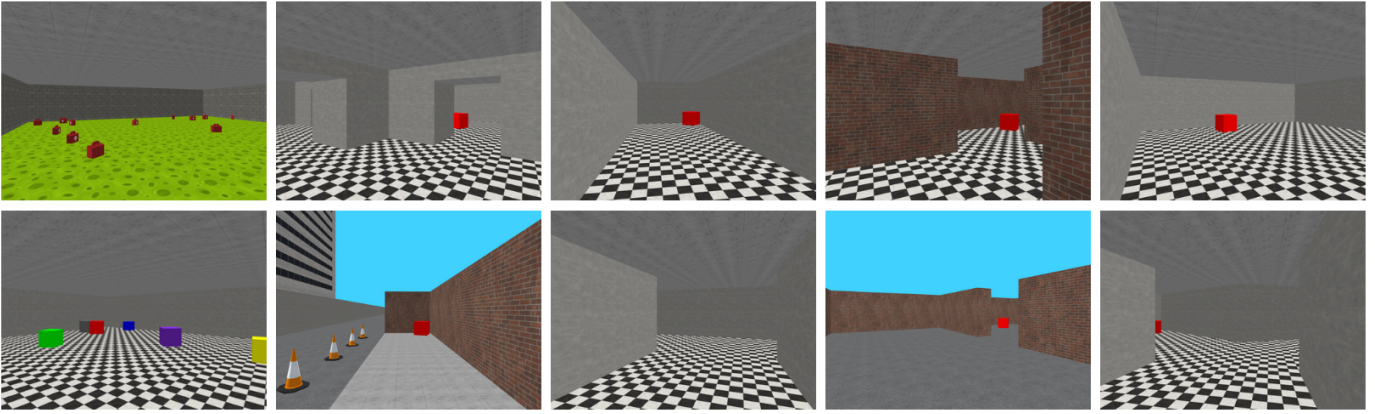


Fig. 2. Screenshots from the 10 MiniWorld environments used in our experiments. MiniWorld uses minimal graphics, while still providing variance in the visual domain. From left to right, top to bottom: CollectHealth, FourRooms, Hallway, MazeS3, OneRoom, PutNext, Sidewalk, TMaze, WallGap, YMaze. Images are upscaled to  $800 \times 600$  for visual clarity.

expert latent space  $\mathcal{D}_E = \{(z_t^{(i)}, a_t^{(i)})\}$  as the set of encoded observation-action pairs of expert trajectories.

During inference, at each timestep  $t$  we encode the current observation  $o_t^*$ , obtaining  $z_t^* = h(o_t^*)$ , and retrieve the  $k$ -most similar latents from  $\mathcal{D}_E$ . Then, we count the number of occurrences of each action and store them in a vector  $\mathbf{c}_t$ . Finally, we model the expert action distribution as  $\pi_E(a_t^{(E)}|s_t)$ , where  $\pi_E(a_t^{(E)} = i|s_t) = \frac{c_t^{(i)}}{k}$ .

### B. Bayesian Online Adaptation

At timestep  $t$ , an agent (that is, a policy  $\theta$ ) observes a state  $s_t$  and selects a discrete action  $a_t^{(\theta)}$ . We can model the prior distribution  $\pi_\theta(a_t^{(\theta)}|s_t)$  as a Dirichlet distribution with  $K = |A|$  components and concentration vector  $\alpha_{\text{prior}}$  with  $\alpha_{\text{prior},i} = \pi_\theta(a_t^{(\theta)} = i|s_t)$ . That is, each pseudo-count is the probability of the corresponding action as inferred by the IL agent.

We formulate our inference problem as a Bayesian adaptation problem. Our aim is to update the beliefs of an IL agent, given a set of actions retrieved from the expert. Therefore, given the prior  $\pi_\theta(a_t^{(\theta)}|s_t)$ , we would need to find the likelihood  $\pi_E(a_t^{(E)}|a_t^{(\theta)}, s_t)$  to estimate the posterior.

In our adaptation setting,  $a_t^{(E)}$  is selected by feature similarity search between the current encoded observation  $z_t^*$  and the previously encoded expert latents. Since  $z_t^{(i)} = h(s_t^{(i)})$ , clearly  $a_t^{(E)}$  is conditionally dependent on  $s_t^{(i)}$ , i.e.  $\pi_E(a_t^{(E)}|s_t)$ . Conversely, in our pipeline the action selected by the policy  $a_t^{(\theta)}$  does not affect the choice of  $a_t^{(E)}$ . Therefore, we can safely state that, in our setting,  $a_t^{(E)}$  is conditionally independent from  $a_t^{(\theta)}$ . Therefore

$$\pi_E(a_t^{(E)}|a_t^{(\theta)}, s_t) = \pi_E(a_t^{(E)}|s_t). \quad (3)$$

Thus, we can use the result of our search as likelihood. Leveraging the fact that the Dirichlet distribution is the conjugate prior of the Multinomial distribution, we can state that the posterior  $\pi_E(a_t^{(\theta)}|a_t^{(E)}, s_t)$  also follows the Dirichlet distribution with  $K$  components and an updated  $\alpha_{\text{posterior}} = \alpha_{\text{prior}} + \mathbf{c}_t$ .

Given that  $\forall i, \alpha_{\text{prior},i} \in [0, 1]$  while  $\forall i, c_t^{(i)} \in \mathbb{N}$ , we argue that  $\alpha_{\text{posterior}}$  might be unbalanced towards  $\mathbf{c}_t$ , hence giving more importance to  $\pi_E(a_t^{(E)}|s_t)$ . To balance the terms, we multiply each  $\alpha_{\text{prior},i}$  by the number of searched vectors  $k$ , so that  $\alpha_{\text{posterior}} = k \cdot \alpha_{\text{prior}} + \mathbf{c}_t$ .

Finally, we can sample the Dirichlet posterior  $\text{Dir}(K, \alpha_{\text{posterior}})$  and obtain a Categorical distribution  $\text{Cat}(\alpha_{\text{posterior}})$ . From this, the new action  $\tilde{a}_t$  can be obtained by sampling

$$\tilde{a}_t \sim \text{Cat}(\alpha_{\text{posterior}}). \quad (4)$$

### C. Search complexity

Autonomous agents should act within a very short, constant amount of time. When introducing search for inference, a dependency on the complexity of the search space follows from it. In general, the more complex and dense the search space, the longer the query time.

In our work, we mitigate this dependency using *faiss* [28], a library for efficient search leveraging GPUs. *faiss* provides methods for both exact and approximate search. In our study, we encode the expert trajectories into  $d$ -dimensional vectors and project them in a search space with the same dimension. Then, we use (exact) L2 search to retrieve the  $k$  closest samples to the current observation. In our experiments, *faiss* was able to handle as many as 150 trajectories without significant delays in inference time (average search time  $6.37 \pm 0.933$ ms).

## IV. EXPERIMENTS

We compare five agents, namely PPO [29], BC [14], GAIL [17], ZIP [24], and BOA on a range of tasks from *MiniWorld* [30], a modular and customizable library with 3D minimalistic graphics for fast rendering. *MiniWorld* features tasks of varying complexity to test diverse skills, such as navigation, memory and planning. Our experiments are run on 10 tasks that provide an explicit terminal condition and access to a scalar reward. A more detailed description of the goals for each environment is provided in Section VIII. All environments provide  $60 \times 80$  RGB observations with integer

pixel values in the range  $[0, 255]$ . In our study, we apply minimal pre-processing to the images by simply scaling the pixel down to lie within  $[0, 1]$  range. Fig. 2 shows an example of observation from each environment.

We manually collected 20 trajectories for each task and trained all agents on the same dataset within a task. All agents used the same encoder within a task, empirically chosen to be a VPT-inspired [3] encoder architecture with a residual [32] backbone and no attention obtained by training GAIL.

As a preliminary step, we study the hyperparameters of our approach, namely, the number  $k$  of retrieved samples during search and the number of trajectories encoded in latent space. We test  $k = \{1, 5, 10, 15, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$ , and estimate the best value for each task by letting a BOA agent play 3 runs of 30 episodes per value, using all available data. We repeat this test for both GAIL-based and BC-based BOA, as the two IL algorithms treat the action distribution differently. In particular, BC tends to be overconfident in its predictions [14], while GAIL predictions are usually much smoother [17].

To evaluate the effect of the number of encoded trajectories on performance, we manually gather an additional 130 trajectories for the *MazeS3* task, up to a total of 150. Then, we assess the performance of BOA using 1-150 trajectories, incrementing the number of encoded trajectories by 5 each time. For this test, we use the environment-wise optimal value of  $k$  found in the previous step and test each configuration over 5 runs of 30 episodes each.

Additionally, we compare our agents by observing their mean episodic return on each task. We enforce here that *in our setting IL and adapted agents never observe the reward*, and that we collect it for the sole purpose of comparing performance. The only agent using the reward signal during training is PPO. We extensively test the agents on 6 runs of 100 episodes for each task.

The source code to reproduce the experiments can be found at [https://github.com/fmalato/online\\_adaptation](https://github.com/fmalato/online_adaptation).

## V. RESULTS

In the first part of this section we study how varying the hyperparameters of our approach affect performance. In the second subsection, we establish a numerical comparison between the tested agents.

### A. Hyperparameters ablation

Fig. 3 and 4 show the ablation study over the number of retrieved samples  $k$  and the number of encoded trajectories  $n$ , respectively.

*a) Number of retrieved samples:* Fig. 3 compares BOA+BC and BOA+GAIL on each environment as the number of retrieved examples  $k$  increases. Overall, no significant trend is visible, and no agent consistently dominates the other. If we consider only GAIL, though, we can notice that in most environments performance is stronger when  $k$  is small.

We explain this descending pattern as a joint effect of the learned latent representation and the expert trajectories. In

particular, whenever the counts vector  $\mathbf{c}_t$  has  $c_t^{(i)} = k$  for some value of  $k$ , the retrieved sample is heavily polarized towards action  $i$ . In this case, the expert will influence  $\alpha_{\text{posterior}}$  the most. On the other hand, if the expert-provided dataset includes similar observations with different actions, the effect of the expert on the posterior will, in general, be mitigated. Still in this case, if  $k$  is small enough, the action distribution has a significant chance of being sharp, hence having a sensible effect over  $\alpha_{\text{posterior}}$ . Considering that such a descending effect is seen for the most part in maze exploration tasks, where similar observations *might* carry different actions (e.g. when facing a wall, an agent can either turn left or right), the dominance for small values of  $k$  is clear. On the contrary, when the environment is simple enough (e.g. Hallway) or is less likely to encounter the above-described situation, the agent show more robust performance when varying  $k$ .

*b) Number of encoded trajectories:* In Fig. 4 we observe how the number of encoded trajectories affects the success rate of both adapted agents. Following from the findings of the previous paragraph, we adopt  $k = 1$  for BOA+GAIL and  $k = 5$  for BOA+BC. The dominance of BOA+GAIL over BOA+BC visible in Fig. 4 confirms the result of the previous experiment, where BOA+GAIL succeeded  $44.00\% \pm 6.48\%$  of the times, while BOA+BC yielded a mean success rate of  $30.00\% \pm 2.72\%$ . Overall, changing  $n$  does not seem to affect any of the adapted agents, even though a faint trend seems to suggest that BOA+BC would benefit from additional data.

The best performance of BOA+GAIL is  $56.00\% \pm 5.96\%$  ( $n = 40$ ), while the worst mean success rate is  $39.34\% \pm 9.25\%$  ( $n = 130$ ). As for BOA+BC, the best and worst results are  $38.67\% \pm 6.06\%$  ( $n = 90$ ) and  $24.66\% \pm 3.80\%$  ( $n = 50$ ) respectively.

### B. Numerical performance

Fig. 5 shows the average return for each tested agent. At first glance, BOA agents either outperform or match the performance of their corresponding IL agent in all environments. Notably, the adaptation generally improves the reward even when the underlying IL policy catastrophically fails.

From Fig. 5 we see how PPO is able to complete a good portion of the exploration tasks, except for TMaze and WallGap. Still, PPO does not achieve reliable performance in any of them. We hypothesize this is due to the sparse nature of the reward. Our other baseline, ZIP, achieves strong performance on every task except PutNext. We believe this follows from the inherent difficulty of the environment that, unlike all the others, requires two sequential steps to achieve the goal.

GAIL and BC solve some of the tasks with acceptable results and fail in others. We suppose that in those cases, BC fails in environments where data does not extensively encapsulate the complexity of the environment, while GAIL failures are due to its inherent instability.

As expected, both BOA agents improved the capabilities of their corresponding IL policies. BOA+BC achieves almost-zero performance only in TMaze and PutNext. We suspect

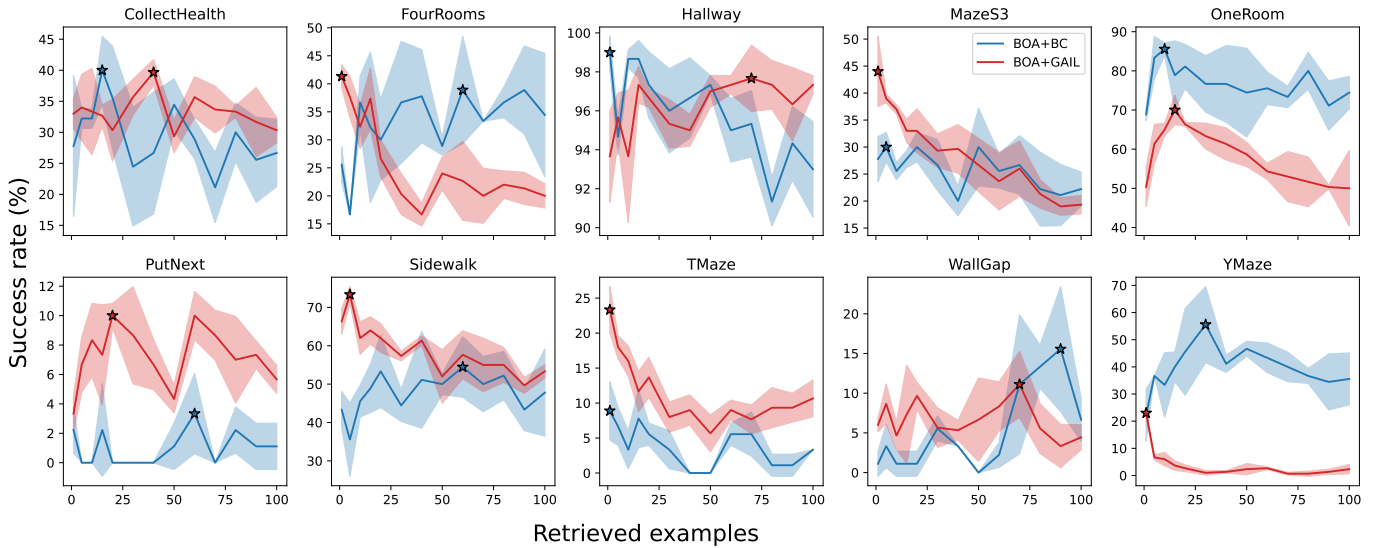


Fig. 3. Mean success rate for different numbers of retrieved samples. Each graph corresponds to one environment. Measurements retrieved on 3 runs of 30 episodes each. The best value is marked with a star. Red lines represent a BOA agent adapting GAIL agent, while a blue line denotes a BC agent adapted with BOA.

TABLE I  
BEST CHOICE OF  $k$  FOR ADAPTED AGENTS BOA+GAIL AND BOA+BC FOR EACH ENVIRONMENT, WITH ASSOCIATED MEAN SUCCESS RATE.

		CollectHealth	FourRooms	Hallway	MazeS3	OneRoom	PutNext	Sidewalk	TMaze	WallGap	YMaze
<b>BOA+GAIL</b>	$k$	40	1	70	1	15	20	5	1	70	1
	Success Rate (%)	39.67 ±2.05	41.33 ±2.05	97.67 ±1.70	44.00 ±6.48	70.00 ±3.74	10.00 ±0.82	73.33 ±2.05	23.33 ±3.30	11.11 ±4.16	23.00 ±1.41
<b>BOA+BC</b>	$k$	15	60	1	5	10	60	60	1	90	30
	Success Rate (%)	40.00 ±5.44	38.89 ±9.56	99.00 ±0.82	30.00 ±2.72	85.56 ±1.57	3.33 ±2.72	54.44 ±7.86	8.89 ±4.16	15.56 ±7.86	55.56 ±13.97

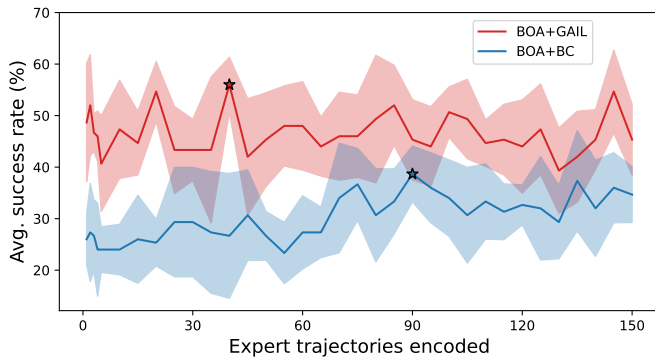


Fig. 4. Mean success rate for different numbers of encoded trajectories. The test is conducted on both BOA+BC (in blue) and BOA+GAIL (red line) on 5 runs of 30 episodes each.  $n$  varies between 1 and 150. We highlight the best value for each agent with a star-shaped mark.

this happens for two reasons: first, BC completely fails on the task; second, the IL policy assigns a very high probability to the selected action, leading the adaptation to fail as well. When

this is not true, such as for GAIL, we see how the adaptation still improves the results.

These findings confirm our hypothesis: while selecting actions directly from the expert distribution is likely to yield better results, adapting IL agents with search on the expert data improves performance. Nonetheless, while ZIP blindly copies actions based on past situations, BOA always takes into account the possibility of different situations. In a more unpredictable and complex scenario we expect BOA to perform better than ZIP, as suggested by the *PutNext* task. We motivate this claim with the following reasoning.

ZIP assumes that an expert can provide optimal solutions to any situation within a specified task. Therefore, ZIP will reliably solve tasks that are represented at least once within the expert dataset. As the complexity of the task increases, ZIP will either require a much larger expert dataset or encounter limitations. On the contrary, an IL policy should be able to generalize its knowledge, thus being capable of addressing some shortcomings of the expert dataset. In such setting, problems are likely to arise whenever a certain situation is

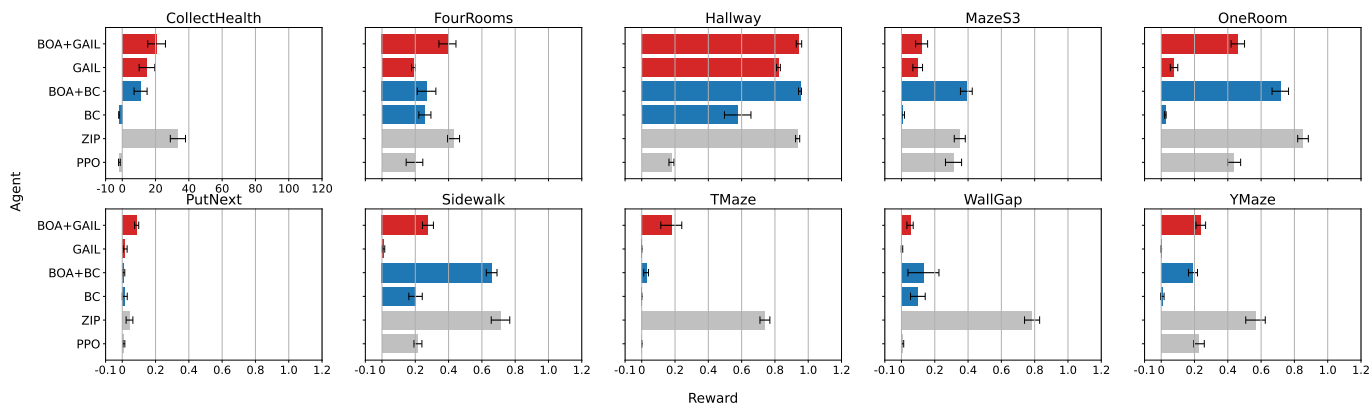


Fig. 5. Average reward comparison over the 10 selected tasks. Higher is better. In all environments, the reward is in the range  $[0, 1]$  except for *CollectHealth*, where the reward is in  $[-2, +\infty)$ . We highlight BOA agents with striped bars and link them to the corresponding IL agent by matching the color of the bar. Grey bars represent baseline methods.

under-represented in the dataset.

BOA complements the two approaches: on one hand, it is capable of addressing the under-representation problem of IL methods thanks to search; on the other hand, it allows generalization in situations where the encoded experience is contradictory. Still, BOA needs a softer search mechanism than ZIP to allow generalization. As such, BOA might fall short of ZIP in simple tasks, but is bound to improve on complex ones such as *PutNext*. Most importantly, BOA *always* improve its underlying policy thanks to search, while keeping generalization capabilities intact.

### C. Perceptual evaluation

As an additional set of experiments, we evaluate the adapted agents perceptually. To support the statements of this subsection, we have released a video on our YouTube channel (visible at <https://youtu.be/WoWalj4CVmM>). Moreover, to keep the amount of visual material limited and the evaluation concise, we evaluate three environments with diverse purposes, namely *FourRooms* for navigation, *PutNext* for the sequential tasks, and *CollectHealth* to evaluate the survival skills. For each environment, we watch 10 episodes played by ZIP, followed by BC and GAIL, and finally by BOA+BC and BOA+GAIL.

*a) FourRooms:* According to our evaluation, ZIP possesses the best “human-like” skills: if it gets stuck in front of a wall, it will most likely recover and can confidently pass through doors. Still, when seeing the red box, in a significant number of instances, ZIP has ignored it. This is a consequence of action copy: if a new search is triggered while the red box is in sight, ZIP will reach the goal. Otherwise, the new search will likely lead to more exploration.

Despite being tested as stochastic, BC will likely get stuck. The agent possesses decent navigation skills, but the overconfidence in its prediction will make it often fixate on a single action. As expected, GAIL possesses better navigation skills and can confidently explore. While it also gets stuck at times, it is much more likely to recover than BC.

The first clear difference between both BOA agents and the other agents is that whenever the red box is spotted, they actively aim for it. Judging from the behavior of the plain IL policies, this is likely an effect of the adaptation. Between BOA+BC and BOA+GAIL, the latter possesses the best navigation skills. While BOA+BC can confidently navigate (resembling ZIP sometimes), it can not overcome door obstacles.

*b) PutNext:* ZIP can confidently explore the room. Additionally, it usually picks up some block and drops it next to another. Still, the case where those two coincide with a yellow and red block is low, hence probability of success is even lower.

BC aims for a block most of the time but then fails to pick it up. We suspect this is a consequence of the unbalance in data, as pick up and drop actions are well under-represented with respect to movement actions. GAIL also seems to be aiming for blocks and can sometimes pick them up, but its actions resemble more a slightly conditioned random policy than a confident one. We suspect that in this case, the training collapsed.

BOA agents show interesting and complementary behavior. Specifically, while this would need to be confirmed further with additional experiments, BOA+BC seems to actively aim for the yellow box and, sometimes, picks it up. Then, it is unable unlikely to drop it. On the contrary, BOA+GAIL can pick up any block and, sometimes, drop it near another. Still, it does not seem to actively look for the yellow one.

*c) CollectHealth:* ZIP shows confident exploratory behavior and is capable of picking up a couple of health kits before dying.

BC mostly turns right. At times, it attempts to aim for a health kit, but the rapidity of the environment does not allow it to pursue the goal. GAIL is more confident in walking towards health kits, but only sometimes manages to pick one up.

BOA+BC corrects the turning behavior of plain BC and allows the agent to reach, sometimes, for a health kit. Similarly, BOA+GAIL can aim for health kits, but only sometimes it can

reach one.

## VI. CONCLUSIONS

We have presented Bayesian online adaptation (BOA), a hybrid search-learning approach for adapting an imitation learning agent in real-time via search on expert-provided data. We have shown how our approach performs better than pure IL agents while falling short of action copy. Regardless, BOA agents carry the advantage of potentially being able to adapt to unpredictable situations, leveraging their underlying IL policy. Future research could investigate the effect of adaptation on RL agents such as PPO, or how to improve training time of RL agents by using adapted actions during training. Another direction could study balancing the effect of the adaptation based on the relevance of each retrieved sample. Finally, combining BOA with classical RL algorithms could prove adaptation as a versatile strategy to boost learning and improve sample efficiency.

Other than improving performance, BOA implicitly addresses the problem of explainability in machine learning. While being far from solving it, such a hybrid approach can always track the effect of the expert-retrieved samples, hence providing additional insight into the action selection process. For instance, during our perceptual evaluation, we noticed that in some instances the effect of the adaptation was immediately clear. Thus, an interesting future research direction could leverage this intuition to enhance explainable agents.

## REFERENCES

- [1] R. S. Sutton, A. G. Barto, "Reinforcement learning: an introduction," The MIT Press, 2nd Edition, 2018.
- [2] K. Arulkumaran, M. P. Deisenroth, M. Brundage, A. A. Bharath, "A Brief Survey of Deep Reinforcement Learning," in IEEE Signal Processing Magazine, Special Issue on Deep Learning for Image Understanding, 2017.
- [3] B. Baker, I. Akkaya, P. Zhokhov, J. Huizinga, J. Tang, A. Ecoffet, B. Houghton, R. Sampedro, J. Clune, "Video PreTraining (VPT): Learning to Act by Watching Unlabeled Online Videos," in arXiv, 2022.
- [4] D. Hafner, T. Lillicrap, M. Norouzi, J. Ba, "Mastering Atari with Discrete World Models," at International Conference on Learning Representations, 2021.
- [5] D. Hafner, J. Pasukonis, J. Ba, T. Lillicrap, "Mastering Diverse Domains through World Models," in arXiv, 2023.
- [6] P. Christiano, J. Leike, T. B. Brown, M. Martic, S. Legg, D. Amodei, "Deep reinforcement learning from human preferences," in arXiv, 2017.
- [7] A. Kanervisto, J. Karttunen, V. Hautamäki, "Playing Minecraft with Behavioral Cloning," at NeurIPS Competitions & Demonstrations Track, 2019.
- [8] A. Kanervisto, J. Pussinen, V. Hautamäki, "Benchmarking End-to-End Behavioural Cloning on Video Games," in IEEE CIG, vol. 2020-August, 2020.
- [9] O. Vinyals, I. Babuschkin, W. M. Czarnecki, et al., "Grandmaster level in StarCraft II using multi-agent reinforcement learning," in Nature 575, 350–354, 2019.
- [10] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Al Sallab, S. Yogamani, P. Pérez, "Deep Reinforcement Learning for Autonomous Driving: A Survey," in IEEE Transactions on Intelligent Transportation Systems, 2020.
- [11] J. Degraeve, F. Felici, J. Buchli, et al., "Magnetic control of tokamak plasmas through deep reinforcement learning," Nature 602, 414–419, 2022.
- [12] K. Arulkumaran, M. P. Deisenroth, M. Brundage and A. A. Bharath, "Deep Reinforcement Learning: A Brief Survey," in IEEE Signal Processing Magazine, vol. 34, no. 6, pp. 26-38, Nov. 2017.
- [13] B. Zheng, S. Verma, J. Zhou, I. Tsang, F. Chen, "Imitation Learning: Progress, Taxonomies and Challenges," in arXiv, 2021.
- [14] F. Torabi, G. Warnell, P. Stone, "Behavioral cloning from observations," at IJCAI, 2018.
- [15] S. Arora, P. Doshi, "A Survey of Inverse Reinforcement Learning: Challenges, Methods and Progress," in arXiv, 2018.
- [16] A. Y. Ng, S. J. Russell, "Algorithms for Inverse Reinforcement Learning," in ICML '00, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 663–670, 2000.
- [17] J. Ho, S. Ermon, "Generative Adversarial Imitation Learning," at NeurIPS, 2017.
- [18] J. Fu, K. Luo, S. Levine "Learning Robust Rewards with Adversarial Inverse Reinforcement Learning," at ICLR, 2018.
- [19] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, "Generative Adversarial Networks," in Proceedings of NeurIPS. pp. 2672–2680, 2014.
- [20] T. M. Moerland, J. Broekens, A. Plaat and C. M. Jonker, "Model-based Reinforcement Learning: A Survey," Foundations and Trends® in Machine Learning: Vol. 16: No. 1, pp 1-118, 2023.
- [21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, "Attention is All You Need," at NeurIPS, 2017.
- [22] A. Kanervisto, S. Milani, K. Ramanauskas, N. Topin, Z. Lin, J. Li, et al., "MineRL Diamond 2021 Competition: Overview, Results, and Lessons Learned," in Proceedings of Machine Learning Research 176:13–28, 2022.
- [23] J. Pari, N. M. Shafiqullah, S. P. Arunachalam, L. Pinto, "The Surprising Effectiveness of Representation Learning for Visual Imitation," in Proceedings of Robotics: Science and Systems, 2021.
- [24] F. Malato, F. Leopold, A. Melnik, V. Hautamäki, "Zero-shot Imitation Policy via Search in Demonstration Dataset," at ICASSP 2024, Seoul, Korea, Republic of, pp. 7590-7594, 2024.
- [25] L. Fan, G. Wang, Y. Jiang, A. Mandlekar, Y. Yang, H. Zhu, A. Tang, D. Huang, Y. Zhu, A. Anandkumar, "MineDojo: Building Open-Ended Embodied Agents with Internet-Scale Knowledge," at NeurIPS, 2022.
- [26] S. Milani, A. Kanervisto, K. Ramanauskas, S. Schulhoff, B. Houghton, R. Shah, "BEDD: The MineRL BASALT Evaluation and Demonstrations Dataset for Training and Benchmarking Agents that Solve Fuzzy Tasks," at NeurIPS 2023 Datasets and Benchmarks Oral, 2023.
- [27] R. Gozalo-Brizuela, E. C. Garrido-Merchan, "ChatGPT is not all you need. A State of the Art Review of large Generative AI models," in arXiv, 2023.
- [28] M. Douze, J. Johnson, H. Jégou, "Billion-scale similarity search with GPUs," in IEEE Transactions on Big Data, vol.7, No. 3, pp 535-547, 2019.
- [29] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, "Proximal policy Optimization Algorithms," in arXiv, 2017.
- [30] M. Chevalier-Boisvert, B. Dai, M. Towers, R. de Lazcano, L. Willems, S. Lahlou, S. Pal, P. Samuel Castro, J. Terry, "Minigrid & Miniworld: Modular & Customizable Reinforcement Learning Environments for Goal-Oriented Tasks," in CoRR, abs/2306.13831, 2023.
- [31] C. Burns, P. Izmilov, J. H. Kirchner, B. Baker, L. Gao, L. Aschenbrenner, Y. Chen, A. Ecoffet, M. Joglekar, J. Leike, I. Sutskever, J. Wu, "Weak-to-Strong Generalization: Eliciting Strong Capabilities With Weak Supervision," in arXiv, 2023.
- [32] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," in IEEE CVPR, Las Vegas, NV, USA, pp. 770-778, doi: 10.1109/CVPR.2016.90, 2016.
- [33] V. Mnih, K. Kavukcuoglu, D. Silver, et al. "Human-level control through deep reinforcement learning," in Nature 518, 529–533, 2015.

## VII. APPENDIX A - ENCODER DETAILS

As stated in Section III-A, in our study we use a VPT-inspired encoder  $h(\cdot)$  to build the latent search space. We have designed our encoder with two purposes in mind: on one hand, we want a temporally extended representation to better characterize each state; on the other hand, we want to keep the architecture as shallow as possible, to keep our architecture easily trainable on consumer hardware with small datasets.

The original VPT architecture is built on top of the Inverse Dynamics Model (IDM) [3] for predicting actions from obser-

vations. The input to both models is a stack of 128,  $128 \times 128$  RGB frames. IDM features a 3D convolution followed by three residual stacks and two fully connected layers with 256 and 4096 activations, respectively. Each residual stack is composed of a 2D convolution with max pooling and two residual blocks as described in [32]. The output is then forwarded to four transformer heads which build long-term relationships between frames. With respect to IDM, VPT removes the 3D convolution layer.

In our work, we use the same architecture as IDM, up to the two fully connected layers. Differently from IDM, our FC layers feature 2048 and 1024 activations. Additionally, we retain the 3D convolution layer but use a stack of 4,  $60 \times 80$  RGB images as input to encapsulate temporally close dependencies, similarly to [33]. As a result, our encoder is composed of  $50M$  parameters, a 90% reduction with respect to IDM/VPT.

### VIII. APPENDIX B - ENVIRONMENTS DESCRIPTION

We provide a short description for each of the 10 tasks used in the study. An example of each environment is provided in Fig. 2.

We use 7 unique navigation tasks in our experiments. They all share the same goal, that is, reaching a randomly-spawned red box. The reward is  $+(1 - 0.2 * \frac{t}{T})$  upon reaching the red box, where  $t$  is the current number of timesteps and  $T$  is the maximum duration of an episode; 0 otherwise. The action space is discrete, 3-dimensional featuring actions  $\{left, right, forward\}$ . We provide a brief description of each.

- **FourRooms:** Four rooms with chess floor tiles, grey walls and ceiling, connected in a quadrant pattern.
- **Hallway:** One rectangular room with chess floor tiles, grey walls and ceiling.
- **MazeS3:** Three randomly-generated rooms connected with corridors. MazeS3 features chess tiles, red brick walls and grey ceiling.
- **OneRoom:** One big squared room with chess floor tiles, grey walls and ceiling.
- **TMaze:** Two corridors connected in a T pattern. Goal position is randomly spawned on either branch of the T.
- **WallGap:** Two rooms with no ceiling connected with a gap in a wall. Agent spawns in one room, red box in the other.
- **YMaze:** Similar to TMaze, but arranged in the shape of a "Y".

Additionally, we use three tasks with different rules. Their descriptions are as follows.

**Name:** *CollectHealth*

- **Description:** The environment is composed of a single room with green floor and grey walls and ceiling. Scattered all over the floor are health kits.
- **Goal:** Survive as long as possible.
- **Action space:** Discrete, 8-dimensional. Actions:  $\{left, right, forward, backwards, pick up, drop, toggle, complete\}$ .

- **Rules:** Agent has an initial health of 100. After each timestep, health decreases by 2. If health reaches 0, the agent dies. Whenever an agent collects a health kit, its health is restored back to 100. Collecting a health kit requires actively performing a *pick up* action.
- **Reward:** +2 for each timestep; -100 for dying.

**Name:** *PutNext*

- **Description:** One big, squared room with chess tiles, grey wall and ceiling, filled with colored boxes.
- **Goal:** Place the yellow box next to the red box.
- **Action space:** Discrete, 8-dimensional. Actions:  $\{left, right, forward, backwards, pick up, drop, toggle, complete\}$ .
- **Rules:** An agent reaches the yellow box and performs a *pick up* action. Then, the agent must navigate to the red box and perform a *drop* action.
- **Reward:**  $+(1 - 0.2 * \frac{t}{T})$  when red and yellow boxes are close; 0 otherwise.

**Name:** *Sidewalk*

- **Description:** One rectangular room opened on one side. The opened side is delimited with traffic cones. Red-brick walls, grey floor and no ceiling.
- **Goal:** Reach the red cube within the time limit.
- **Action space:** Discrete, 3-dimensional. Actions:  $\{left, right, forward\}$ .
- **Rules:** Navigate the maze to reach the goal location. If the agent steps out of the sidewalk, it dies.
- **Reward:**  $+(1 - 0.2 * \frac{t}{T})$  when the goal is reached; 0 otherwise.