Imitation from Observation With Bootstrapped Contrastive Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

Imitation from observation is a paradigm that consists of training agents using observations of expert demonstrations without direct access to the actions. Depending on the problem configuration, these demonstrations can be sequences of states or raw visual observations. One of the most common procedures adopted to solve this problem is to train a reward function from the demonstrations, but this task still remains a significant challenge. We approach this problem with a method of agent behavior representation in a latent space using demonstration videos. Our approach exploits recent algorithms of contrastive learning of image and video and uses a bootstrapping method to progressively train a trajectory encoding function with respect to the variation of the agent policy. This function is then used to compute the rewards provided to a standard Reinforcement Learning (RL) algorithm. Our method uses only a limited number of videos produced by an expert and we do not have access to the expert policy function. Our experiments show promising results on a set of continuous control tasks and demonstrate that learning a behavior encoder from videos allows for building an efficient reward function for the agent.

1 Introduction

Imitation from Observation (IfO) [16; 11] involves learning a policy for an agent to solve a task using a set of demonstrations of an expert performing the same task. Distinct from Imitation Learning (IL) [9; 13; 12; 2], in IfO, the demonstrations are only sequences of observations of the environment by the expert. The advantage of the IfO formalization is that it approximates as closely as possible the conditions under which humans use their ability to imitate experts during learning. The two main methods in the literature that have been used to train IfO problem agents are adversarial methods and reward learning methods [19]. The adversarial methods [17; 18] adapt ideas from inverse reinforcement learning [8], proposing a GAN-like architecture [5] where the agent is a trajectory generator associated with a discriminator function that evaluates how similar the step transitions produced are to that of an expert. The reward learning methods [10; 11; 3] are based on the training of a reward function that will be used to reward the agent during a classic RL algorithm.

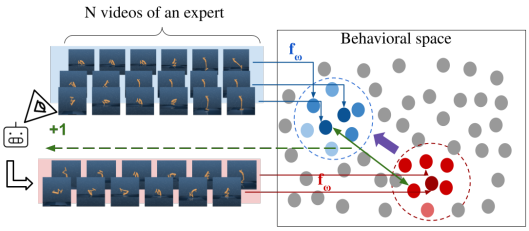


Figure 1: Illustration of our method. An encoder that takes videos of agent trajectories and embeds them in a "behavioral space" is trained using contrastive learning that encourages successful trajectories to be near each other. We use this to encode N expert videos in a region of the behavioral space depicted in blue. The reward function corresponds to the distance of the agent's trajectory to the set of expert trajectories. As the agent progresses, its current trajectories are incorporated as "negative" examples into the contrastive learning in red.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

In this paper, we propose a reward learning IfO algorithm that relies on the representation of agent behaviors in a latent space using raw pixel-based demonstrations, illustrated in Figure 1. Unlike adversarial methods, we use the entire observation sequences with an LSTM to keep track of the agent's effects and to represent the induced behavior of the agent at each instant of the sequence. This method directly exploits a limited set of raw expert videos to train an imitation agent, without having access to the expert policy or actions. Inspired by Berseth *et al.* [1], we also train the reward function progressively with new trajectories generated by the agent. In contrast to [1], our method applies a pre-training of the trajectory encoding function on a set of failure trajectories before agent training. This helps to ensure the consistency of the reward function from the beginning of the agent training. We also use Contrastive Multiview Coding [15] and Dense Predictive Coding [7] methods for self-supervised learning of trajectories. Like [11; 1], the reward value at each step is correlated to how close the agent trajectory is to the one the expert would have had. The reward function is progressively improved during the training of the agent to recognize and distinguish the agent's behavior in order to push it closer to the expert's. Our method successfully solves easy and hard tasks of the Deepmind Control Suite [14] such as Reacher and Hopper and is able to achieve returns close to DDPG experts on these tasks.

2 Method

Agent training We aim to train an agent to perform a task only using a set of videos $\mathcal{D}_e$ of an expert performing the task. Our solution is based on the trajectory matching objective, which consists in training the agent to have a trajectory similar to that of the expert. In other words, the only information we seek to extract from the expert's videos is his behavior. Our algorithm is divided into two phases, in the *first phase* we train an image encoding function g_θ and a sequence encoding function f_ω. During this phase, we seek to learn the functions f_ω and g_θ, to encode trajectories coming from two distributions: the distribution of expert trajectories $O \sim p(\mathcal{D}_e)$, and the distribution of trajectories from a random policy. In the *second phase*, we train the agent policy π while updating the image and sequence encoder with additional data observations from our online interactions with the environment. During this second phase, we apply an RL algorithm using the distance between sequence encodings of the expert and current agent to obtain a reward. In this work, we use DrQv2 [20] to train the agent policy, π, and its associated q-value function, Q, though our method is agnostic to the choice of RL algorithm. Specifically, in the second phase at each new episode of the agent, we sample an expert episode $O_e \sim p(\mathcal{D}_e)$ that will be used as a support for the agent. At each agent step, t, we evaluate the distance between $o_{0:t}$, the agent's trajectory until the step t, and $o_{e,0:t}$ the expert's trajectory until t by $d(o_{0:t}, o_{e,0:t}) = ||f_\omega(g_\theta(o_{0:t})) - f_\omega(g_\theta(o_{e,0:t}))||$. The reward assigned to the agent at this step is finally $r_t = -d(o_{0:t}, o_{e,0:t})$. The function f_ω being pre-trained on the distributions $O \sim \mathcal{D}_e$ and $O \sim P(o_0, ..., o_T|\pi_{\text{random}})$, it is not consistent for the other trajectories generated by the policy π during its the improvement. In order to fix this in the second phase, we continue to update g_θ and f_ω on the new trajectories generated by the agent during the training π. In Appendix C, we discuss the effects of pre-training the encoding functions before training the agent. The full algorithm is presented in Appendix B.

Image encoding In the first step, we learn an image-level representation. The image level objective contains three terms: (1) $\mathcal{L}_{triplet}$ which enforces image encodings similarity of adjacent frames, (2) $\mathcal{L}_{ae}$ which permits the encoding to be decoded back to an image, and finally (3) we learn a contrastive multiview encoding following[15] based on a LAB space [4] image representation. The first term $\mathcal{L}_{triplet}$ compares the distance between an anchor image o, a positive image o_p temporally close to the anchor, and a negative image o_n distant from the anchor image in the video:

$$\mathcal{L}_{triplet}(\theta) = ||s - s_p||^2 + \max(\rho - ||s - s_n||^2, 0) \tag{1}$$

where $s = g_\theta(o)$, $s_p = g_\theta(o_p)$. In order to ensure consistency and non-degeneracy of the vectors returned by g_θ, we need to make sure that each state vector s can reconstruct the initial image o. The training objective of the autoencoder is then to minimize the loss function $\mathcal{L}_{ae}(\theta, \gamma) = ||o - q_\gamma(g_\theta(o))||^2$. Finally, we incorporate multiview contrastive coding to enrich the visual representations following [15]. Specifically we consider the color space transformation $\{R, G, B\} \rightarrow \{L, ab\}$ where the component L is the view 1,v_1, processed by CNN, g_{θ_1}, and the component ab is the view 2,v_2, processed by g_{θ_2}. The final encoding s of an image o is given by $s = g_\theta(o) = [g_{\theta_1}(v_1), g_{\theta_2}(v_2)]$. Following the formulation, [15] we align the different views using a contrastive learning loss denoted, $\mathcal{L}_S$. This process is illustrated in the Figure 3 in the Appendix A. Overall the objective function to

Table 1: Evaluation of the average return over 1000-frame episodes of agents trained with the Context translation (CT) [11] and ViRL [1] algorithms. We evaluate the agents on the Reacher Hard, Finger Turn Easy, Hopper Stand and Walker Run tasks.

Agent	Avg. return			
	Reacher Hard	Finger turn easy	Hopper Stand	Walker Run
expert (DrQv2)	850.52 ± 315.70	893.44 ± 219.75	880.93 ± 76.89	789.77 ± 17.25
Ours	$\mathbf{843.16 \pm 274.84}$	199.76 ± 399.02	$\mathbf{651.33 \pm 362.95}$	$\mathbf{79.498 \pm 1.25}$
ViRL	0.44 ± 1.27	160.2 ± 366.52	516.27 ± 417.21	29.31 ± 24.72
CT (n=10)	97.84 ± 254.52	$\mathbf{238.92 \pm 422.07}$	1.42 ± 3.22	25.15 ± 18.71
CT (n=1)	38.04 ± 127.53	199.48 ± 396.99	1.01 ± 1.71	52.42 ± 34.28
Random	6.64 ± 16.94	92.6 ± 202.68	2.44 ± 5.19	29.93 ± 4.52

minimize the training of the image encoder is:

$$\mathcal{L}_{frame}(\theta, \gamma) = \mathcal{L}_S(\theta) + \mathcal{L}_{triplet}(\theta) + \mathcal{L}_{ae}(\theta, \gamma) \tag{2}$$

Sequence encoding In order to allow us to construct a reward function, we learn to represent a sequence of frames [6; 7] with a semantically meaningful distance between them. We utilized two losses for this (1) $\mathcal{L}_Z$ a standard video representation learning approach Dense Predictive Coding (DPC) [7] (2) and $\mathcal{L}_O$ bootstrapped expert behavior loss, which encourages separation between expert and non-expert trajectories. We note that $\mathcal{L}_O$ lies at the heart of our method and allows the sequence representation to gradually improve as non-expert trajectories improve. DPC is based on the principle of predicting the next frame using the sequence of previous frames as input. Let us consider a dataset of videos $\mathcal{D} = \{O^i\}_{1 \leq i \leq N} = \{o^i_{0:T}\}_{1 \leq i \leq N}$. Denote, d_ϕ, which from an encoding z_t of a sequence of states $s_{0:t}$, predicts the next state $\hat{s}_{t+1}$ of the sequence. Following [7] we can use this for spatio-temporal representation learning. We first predict the next state $\hat{s}_{t+1} = d_\phi(f_\omega(s_1, \ldots, s_t))$ and subsequently use this to predict K-1 additional states $\hat{s}_{t+k} = d_\phi(f_\omega(s_1, \ldots, s_t, \hat{s}_{t+1}, \cdots, \hat{s}_{t+k-1}))$. We then apply the learning objective to minimize the loss function $\mathcal{L}_Z$ given by:

$$\mathcal{L}_Z(\theta, \omega, \phi) = -\mathop{\mathbb{E}}_{o_{0:T}} \left[\frac{1}{K} \sum_{1 \leq k \leq K} L_{t+k} \right] ; \; L_t = \log \frac{\mathtt{h}(\hat{s}_t, s_t)}{\mathtt{h}(\hat{s}_t, s_t) + \sum\limits_{t' \neq t} \mathtt{h}(\hat{s}_t, s_{t'})} \tag{3}$$

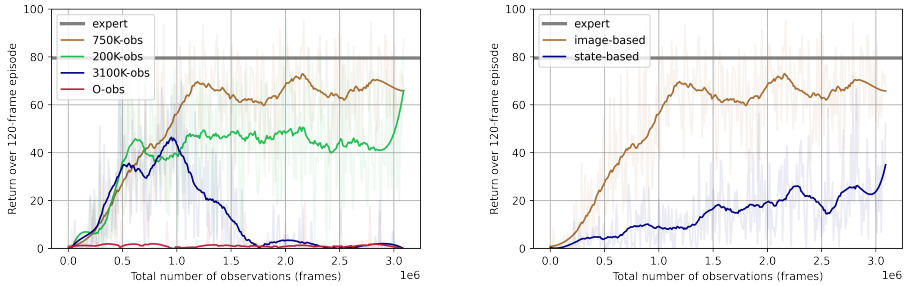
where $\mathtt{h}(a, b) = \exp(\frac{a^T b}{\tau \|a\| \|b\|})$ and τ is the temperature parameter. Indeed, the loss function is also a contrastive loss function where the positive pairs are the pairs $(\hat{s}_t, s_t)$ and the negative pairs are the pairs $(\hat{s}_t, s_{t'})_{t \neq t'}$ in a same video. In order to bring sequences coming from the same distribution closer together, we use another contrastive loss that applies to the sequence encoding vectors z returned by f_ω. Thus for a sequence O belonging to one of the two trajectory distributions, we consider a positive sequence O_p of the same distribution and k negative $\{O_{n,i}\}_{1 \leq i \leq k}$ of different distributions. The objective function to be minimized is $\mathcal{L}_O$:

$$\mathcal{L}_O(\theta, \omega) = -\mathop{\mathbb{E}}_{(O, O_p, \{O_{n,i}\})} \left[\log \frac{\mathtt{h}(z, z_p)}{\mathtt{h}(z, z_p) + \sum\limits_i \mathtt{h}(z, z_{n,i})} \right] ; \;\; z = f_\omega(g_\theta(o_0), \ldots, g_\theta(o_T)) \tag{4}$$

The objective function that we seek to minimize to train the sequence encoding function is $\mathcal{L}_{seq}(\theta, \omega, \phi) = \mathcal{L}_Z(\theta, \omega, \phi) + \mathcal{L}_O(\theta, \omega)$. Finally, for a triplet $(O, O_p, \{O_{n,i}\}_{1 \leq i \leq k})$, the frames used to evaluate $\mathcal{L}_{frame}$ come from the sequences of the sample, which allows us to compute the objective loss function $\mathcal{L}(\theta, \gamma, \omega, \phi) = \mathcal{L}_{frame}(\theta, \gamma) + \mathcal{L}_{seq}(\theta, \omega, \phi)$.

3 Experiments

Training setup The image encoding function g_θ is a pair of CNN g_{θ_1} and g_{θ_2} of similar architectures operating on the L and ab channels respectively. The sequence encoding function f_ω is a 2-layer long Short-term memory neural network with input and output size 128. The output sequence is followed by a fully-connected layer with input and output sizes equal to 128. The next-state predictor d_ϕ is a sequence of two 128×128 fully-connected layers. Unless otherwise specified, all images used in the experiments are 64×64. For each task, the $\mathcal{D}_e$ dataset of expert episodes contains 5000 videos of 61



(a) Impact of the encoder training duration

(b) Evaluation of the state-based agent

Figure 2: Evolution of the average return of the agent on the Reacher Hard task over 5 episodes of 120 frames. In (a) we evaluate the agent w.r.t the training duration of the encoding functions estimated in total number of training observations. In (b) we evaluate the agent w.r.t. to the input information processed by the agent: raw pixels or image encodings.

frames. The $\mathcal{D}_a$ dataset also contains a maximum of 5000 videos during encoder pre-training and policy training. The training of the encoders is done during $N_\omega = 750K$ agent training observations. The training of the agent is done during $N_\pi = 3100K$ training observations.

Experiments and results We perform the training on the Reacher Hard, Finger Turn Easy, Hopper Stand, and Walker Run tasks from the Deepmind Control Suite [14]. The demonstrations of our agents acting in these tasks are presented at the figure 6 in Appendix E. We compare our algorithm to the context translation method (CT) [11] (by varying the number n of expert videos used by the agent in each episode) and to ViRL [1]. The results, presented in table 1, show that our method is successful in understanding the expert's behavior and in performing the tasks. Details on the evolution of the average episodic returns are presented in the figure 5 in Appendix D. We notice that when the coordination of actions is necessary to perform the task, the agent has more difficulty earning rewards. In particular, we find that when the environment is subject to external forces such as gravity over which the agent has no visual information, imitation becomes more difficult. In our method, we consider the training duration of the encoding functions in the second part of the method during the training of the agent. This duration is evaluated using the total number of frames observed by the agent in all training episodes. The interest of this period of training is to avoid the reward function to be too severe in rewarding the agent, knowing that if there is no variation in reward, the agent receives no signal to differentiate optimal from bad actions. Thus, we evaluate the agent with encoders trained during a period of 0, 200K, 750K and 3100K agent total observations. Encoder parameter update happens every 100 observations. The results of these experiments on the Reacher Hard task are presented in figure 2a. When the encoding functions do not train on enough agent videos, the reward function is not precise and strict enough. When the encoding functions train a lot on too many videos, the reward function remains severe due to overtraining on expert videos. An important point in this method is to understand how informative the states returned for the image encoding function g_θ are for the agent. For this, we use the image encodings s_t returned by g_θ as input data of the agent and freeze the parameters of g_θ during the training of the agent. We present in figure 2b the evolution of the average return of both agents on the Reacher Hard task. These results show that the agent processing the image encodings has more difficulties to succeed in the task and shows slower progress. This suggests that the amount of information encoded during the training of the g_θ function remains quite limited. In future work, we will consider using pre-trained ConvNet models on ImageNet that may provide more consistent encodings.

4 Conclusion

We have presented a new method of Imitation from Observation that relies on the encoding of agent behavior using demonstrative videos. This method uses self-supervised image and video learning algorithms to train progressively a reward function for the agent to be more precise and consistent. Our results demonstrate the success of this method on a set of tasks only using a limited set of expert videos, but note the challenge of using image encodings as input data, which still remains an option to reduce the computation cost of the training.

References

[1] Glen Berseth, Florian Golemo, and Christopher Pal. Towards learning to imitate from a single video demonstration. *arXiv preprint arXiv:1901.07186*, 2019.

[2] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[3] Wei-Di Chang, Juan Camilo Gamboa Higuera, Scott Fujimoto, David Meger, and Gregory Dudek. Il-flow: Imitation learning from observation using normalizing flows. *arXiv preprint arXiv:2205.09251*, 2022.

[4] Christine Connolly and Thomas Fleiss. A study of efficiency and accuracy in the transformation from rgb to cielab color space. *IEEE transactions on image processing*, 6(7):1046–1048, 1997.

[5] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[6] Daniel Gordon, Kiana Ehsani, Dieter Fox, and Ali Farhadi. Watching the world go by: Representation learning from unlabeled videos. *arXiv preprint arXiv:2003.07990*, 2020.

[7] Tengda Han, Weidi Xie, and Andrew Zisserman. Video representation learning by dense predictive coding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019.

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

[9] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2):1–35, 2017.

[10] Daiki Kimura, Subhajit Chaudhury, Ryuki Tachibana, and Sakyasingha Dasgupta. Internal model from observations for reward shaping. *arXiv preprint arXiv:1806.01267*, 2018.

[11] YuXuan Liu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Imitation from observation: Learning to imitate behaviors from raw video via context translation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1118–1125. IEEE, 2018.

[12] Stéphane Ross, Geoffrey Gordon, and Drew 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*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

[13] Stefan Schaal. Learning from demonstration. *Advances in neural information processing systems*, 9, 1996.

[14] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

[15] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. In *European conference on computer vision*, pages 776–794. Springer, 2020.

[16] Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. *arXiv preprint arXiv:1805.01954*, 2018.

[17] Faraz Torabi, Garrett Warnell, and Peter Stone. Generative adversarial imitation from observation. *arXiv preprint arXiv:1807.06158*, 2018.

[18] Faraz Torabi, Garrett Warnell, and Peter Stone. Imitation learning from video by leveraging proprioception. *arXiv preprint arXiv:1905.09335*, 2019.

[19] Faraz Torabi, Garrett Warnell, and Peter Stone. Recent advances in imitation learning from observation. *arXiv preprint arXiv:1905.13566*, 2019.

[20] Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning. *arXiv preprint arXiv:2107.09645*, 2021.

A Feed-forwarding process of observation sequences into trajectory encoder

In Figure 3, we present how observation sequences are fed through the functions f, g, q and d.

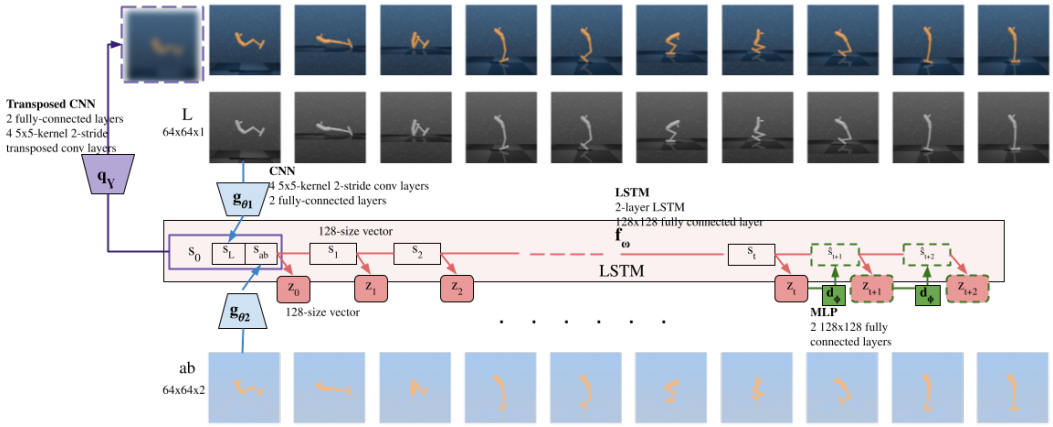


Figure 3: Training principle of the encoding functions. For each episode, the video is decomposed in the Lab color space, constituting the L and ab views. Each frame is encoded by g_θ and decoded by q_γ. The resulting state sequence $s_0, ..., s_t$ is encoded using the LSTM f_ω to provide the sequence encoding z_t. z_t is then processed by d_ϕ and f_ω to predict future image encodings.

B Pseudo-code

In Algorithm 1 we present the complete process in pseudo-code of our method.

C Without pre-training of the image and sequence encoders

We also study in our work the importance of pre-training the encoders in phase 1. These functions are trained on two trajectory distributions, the expert's and a random policy's. The primary interest of this pre-training is to provide a reward function initially adapted to adequately reward the random behavior of the agent. However, to some extent, this pre-training may not be necessary in the long term, as the agent will adapt to new and increasingly significant rewards. We present the results of this experiment in figure4.

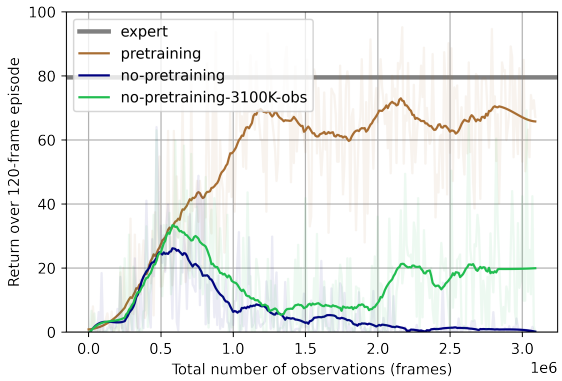
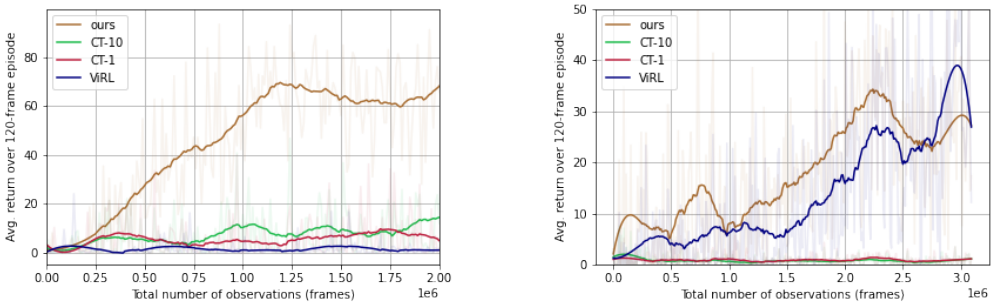


Figure 4: Evaluation of the effect of the pre-training of the encoding functions in phase 1 on the final performance of the agent.

Algorithm 1: Imitation from observation with bootstrapped reinforcement learning

$D_e = \{O_e^i\}_{1 \leq i \leq N} = \{(o_{e,0}^i, o_{e,1}^i, ..., o_{e,T}^i)\}_{1 \leq i \leq N}$
Initialize $f_\omega,\ g_\theta,\ q_\gamma,\ d_\phi$
`// PHASE 1: Pretraining of` f, g
while $k \leq N_{pretrain}$ **do**
  $\{O_e^i\}_{1 \leq i \leq n} \leftarrow sample(D_e)$
  $\{O^i\}_{1 \leq i \leq n} \leftarrow \pi_{random}(env)$
  Eval. $\mathcal{L}(\theta, \gamma, \omega, \phi)$ with $(\{O_e^i\}, \{O^i\})$
  Optimize θ, γ, ω, ϕ
end
`// PHASE 2: Training of` π, Q
Initialize $D_a = \{\}$
Initialize $\mathcal{D} = \{\}$, Q, π `// Replay buffer, Q-value function, policy`
while $step \leq N_\pi$ **do**
  $O_e \leftarrow sample(D_e)$
  $o_0 \leftarrow env.reset()$
  for $t \in 0, .., T-1$ **do**
    $a_t \leftarrow \pi(o_t)$
    $o_{t+1} \leftarrow env.step(a_t)$
    $r_{t+1} \leftarrow -||f(o_{0:t+1}) - f(o_{e,0:t+1})||$
    Save $(o_t, a_t, o_{t+1}, r_{t+1})$ into $\mathcal{D}$
    $(o, a, o', r) \leftarrow sample(\mathcal{D})$
    Update Q and π with (o, a, o', r) using DrQv2
  end
  if $step \leq N_{train}$ *and* $step \mod N_{update} = 0$ **then**
    Save $O = \{o_{0:t+1}\}$ into D_a
    $\{O_e^i\}_{1 \leq i \leq n} \leftarrow sample(D_e)$
    $\{O^i\}_{1 \leq i \leq n} \leftarrow sample(D_a)$
    Eval. $\mathcal{L}(\theta, \gamma, \omega, \phi)$ with $(\{O_e^i\}, \{O^i\})$
    Optimize θ, γ, ω, ϕ
  end
end

D Comparison of the average returns during the agent training with other methods



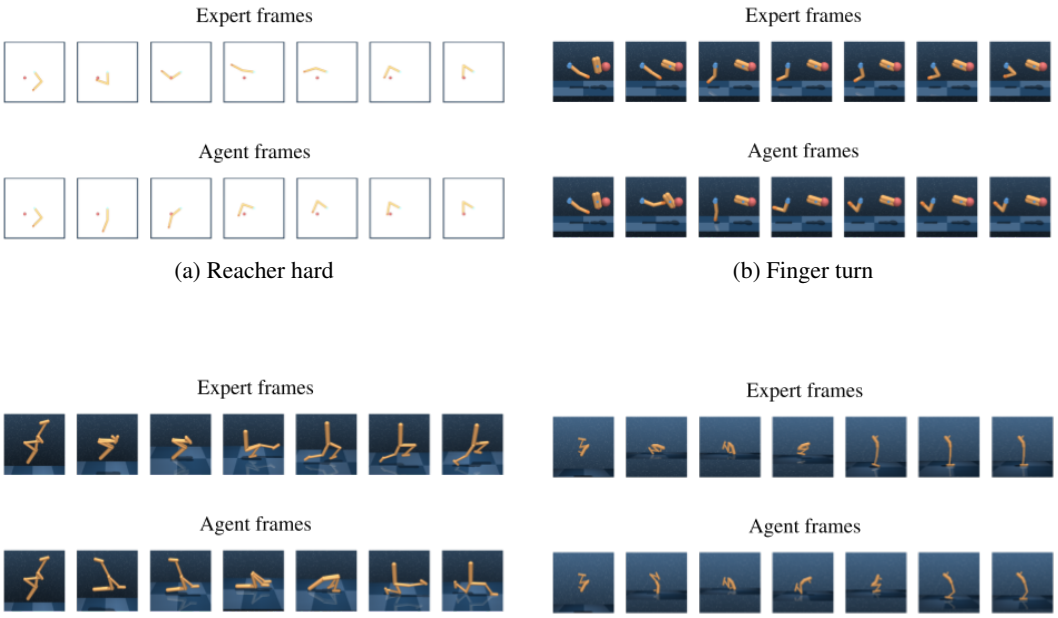
(a) Reacher hard

(b) Hopper stand

Figure 5: Evolution of the average returns during the training of our agent compared to context translation and ViRL agents.

E Demonstrations

We present in figure 6, examples of execution of each of the tasks by the expert and by the agent, under the same initial conditions.



(a) Reacher hard (b) Finger turn

(c) Walker run (d) Hopper stand

Figure 6: Comparison of the actions between expert and agent in each task and in the same initial conditions.

F Hyper-parameters

All the hyper-parameters of our method and the default values are presented in the Table 2.

Table 2: Hyper-parameters and default values of the algorithms

Parameter	Default value
Number of expert videos (N)	5000
Number of frames in each video (T)	61
Size of image frames in each video	64×64
Number encoders pre-training epochs ($N_{pretrain}$)	8000
Number of videos per batch (n)	16×2 (pairs)
Total number of observations during which encoders are trained (N_{train})	750K
Periodicity of encoder parameter update in training observation (N_{update})	100
Total number of training observations (N_{π})	3.1 M
Learning rate	1×10^{-4}
Optimizer	ADAM