Final Project: Extraction and Utilization State Representation for Visual Imitation

Kai Yan
University of Illinois Urbana-Champaign
kaiyan3@illinois.edu

Abstract

In this report, we study the extraction and utilization of state representation in Visual Imitation (VI). Based on two relevant papers, we reimplement a codebase that tests the works on dm-control, an established visual RL testbed new to the original codebase, from scratch and studied the performance of the state representation learned by different methods, including end-to-end behavior cloning with data augmentation, BYOL, SIMCLR and VICREG. See code in `https://github.com/Kai289371298/CS498final`.

1. Introduction

Visual Imitation (VI) [4, 22, 27, 29] is an increasingly popular topic in the robotics community where a robot learns from expert video demonstration [17], keyframes, or even pictures of goals [10]. Such setting is very useful under a variety of applications: For example, when the kinethestic data of expert is unavailable, when the robot need to learn from video in-the-wild, such as Youtube [17], or when learning from an expert with different embodiments [39] and actions are no longer applicable.

When solving the VI (and visual RL) problem, a good state representation is very important and extensively studied by the researchers [18, 22, 30, 38]. The importance of state representation comes from two reasons: 1) *summarizing large space*. When we try to move a pointmass around a 2-dimensional space, with proprioceptive state, only the coordinate and velocity of the pointmass is relevant, which gives us a small 4-dimensional state space; however, with a visual input of RGB channel and 84×84 of the space, the dimension of states becomes a staggering amount of $3 \times 84 \times 84$, which is hard to directly dealt with; 2) *Better generalization*. For example, when the instruction is irrelevant of color, a red robotic arm should be viewed as the same item as a green robotic arm, though the two might be wildly different pixelwise. This is also the point where visual input might work even better than proprioceptive states - it is almost impossible to generalize to an unseen location of the target item in a proprioceptive state, but very likely with a proper semantic abstraction in a good state representation [39].

Thus, In this report we will focus on exploring the extraction and utilization of state representations for visual imitation tasks. To be more specifically, we will be focusing on the reimplementation of two works, VINN [22] and RAD [18], which are two important work in recent years. The former is a visual imitation work that uses contrastive learning methods for state representation, and does nearest neighbour matching over the demonstration data, which gives a simple but effective solution for the visual imitation task; the latter is a reinforcement learning (RL) work that explores the effect of data augmentation to reinforcement learning. RAD is a representative of a line of work [18, 30, 37, 38] that utilizes data augmentation for better state representation in RL, from which we borrow the augmentation method but utilize on imitation learning. For both works, we will test their performance, and analyze their state representations extracted from the environment.

Our work can be summarized as follows: 1) reimplement VINN and RAD's data augmentation from scratch, integrate them in the same codebase, and test their performance on the OpenAI dm-control [1] testbed which is novel to VINN, and 2) give an analysis for the state representation produced by both works. We believe this will be helpful to people who are interested in visual imitation as their research focus.

2. Related Work

Our work is closely related to the following three areas of research: visual imitation, contrastive learning and data augmentation.

Visual Imitation (VI). VI aims to make robot learn from expert video demonstrations, which is increasingly popular as video data is much more abundant than other perception data [5, 17], such as Lidar or sensors; it also has much better generalizability due to the rich semantic of visual input [39]. The two great challenges in VI is learning across different embodiment and viewpoints, and learning with expert action missing. For the former, a set of embedding learning method is developed [29, 31, 39]; for the latter, the most popular solution in robotics community is RL/MPC with similarity-based reward design [19, 26, 36] and pseudolabeling of action using inverse dynamic model [6, 17, 32], and there is also more mathematical solution from the RL community. VI is the major study field of this work.

Contrastive Learning. Contrastive learning [14] is a subfield of self-supervised learning that aims to learn an embedding space where similar data points have small distance, while dissimilar ones are widely separated. Such methods often uses pairs of data with the same or different kind to construct loss function [9, 20, 25], or uses scoring function to tell a sample apart from noises such as InfoNCE [21]. Most recent works of contrastive learning [3, 8, 11] in vision adopts the idea of *parallel augmentation* [35]: that is, the semantic meaning of a picture should remain consistent under different data augmentation methods. Our work extensively uses contrastive learning for the training of the encoder, and all three contrastive learning methods tested in our work utilizes the idea of parallel augmentation.

Data Augmentation. Data augmentation [28, 34] is a technique in vision-related deep learning to avoid overfitting due to the large number of parameters in the model and relative few pictures as the training set, such as flipping the picture, adding noises such as jittering, rotation of pictures, or cropping the picture. Many works have been done on finding a better data augmentation method with reinforcement learning or imitation learning [18, 30, 37, 38]; in RAD [18], the author suggests that the most helpful data augmentation for RL environment is translation and cropping, which forces the model to pay attention to the edge of the agent.

3. Preliminaries

Markov Decision Process. Markov Decision Process (MDP) is the framework and foundation for reinforcement learning (RL) and imitation learning (IL), where the decision maker is called an *agent*. A MDP can be described by a quintuple (S, A, T, r, γ), where S stands for state space and A stands for action space. MDP is rolled out in *timesteps*: for timestep $t \in \{1, 2, \dots\}$, a state $s_t \in S$ is given to the agent, and the agent will choose an action $a_t \in A$ using its policy $\pi(a|s) \in \Delta(A)$, where Δ is the probability simplex. After a_t is executed, a reward $r(s_t, a_t)$ will be given, and the MDP transits to a new state s_{t+1} according to the transition function $T(s_{t+1}|s_t, a_t)$. A Complete run of the environment is called an episode. The goal of the agent is to maximize discounted total reward over the episode, which is $\sum_t \gamma^t r(s_t, a_t)$; here, $\gamma \in [0, 1]$ is called discount factor which controls the myopicness of the agent.

Imitation Learning. Imitation learning (IL) aims to learn a good policy $\pi(a|s)$ without access to reward function, which is hard to design in real-life applications. In this work, we will consider the simplest setting: consider we have a set of state-action pairs $D = \{(s_1, a_1), (s_2, a_2), \dots, (s_n, a_n)\}$ collected from expert, which is called the expert dataset; here, s is an image in visual imitation. We will assume that our algorithm only has access to such dataset without reward labels.

4. Methodology

Fig. 1 shows the pipeline of our work, where state s of visual input will first be fed into the encoder and generate a representation $z \in \mathbb{R}^n$, which is then fed into the actor to generate final action. Our work mainly tests three paradigms over the pipeline: 1) train the encoder and actor jointly in an end-to-end manner, but with data augmentation explored in the RAD paper. 2) use different contrastive learning tested in VINN [22] to fine-tune encoder, which are BYOL [11], SIMCLR [8] and VICREG [3], then freeze the encoder and train the actor with behavior cloning, and 3) use contrastive learnign to train VINN, then use nearest neighbour, a non-parametric approach, as the actor.

The rest of the section will introduce the three paradigms one by one, and is organized as follows: In Sec. 4.1, we will introduce Augmented end-to-end Behavior Cloning (ABC); in Sec. 4.2, we will introduce the three self-supervised methods tested in our work, which are BYOL [11], SIMCLR [8] and VICREG [3]; in Sec. 4.2.2, we will introduce how we retreive final policy with paradigm 2) and 3), namely, behavior cloning and nearest neighbours.

4.1. Augmented End-to-End Behavior Cloning

Similar to normal behavior cloning, Augmented end-to-end Behavior Cloning (ABC) maximizes the log likelihood of action a given state s, which is $E_{(s,a)\sim D}\pi(a|s)$. However, different from normal behavior cloning, ABC conducts data augmentation $f(\cdot)$ on state for every sampled minibatch in the training progress. Thus, the optimization objective can be written as

$$\max_\pi E_{(s,a)\sim D} \log \pi(a|f(s)). \quad (1)$$

In this work, we tested two variants of f, which are *translation* and *crop+resize* - the two methods that have been proved empicially by RAD to be the most helpful data augmentation. The former moves the picture to a random direction for a few pixels, leaving black areas in the position before; the latter crops a certain patch the picture, and then resize to the original size. Fig. 2 shows an illustration of the two data augmentation methods.

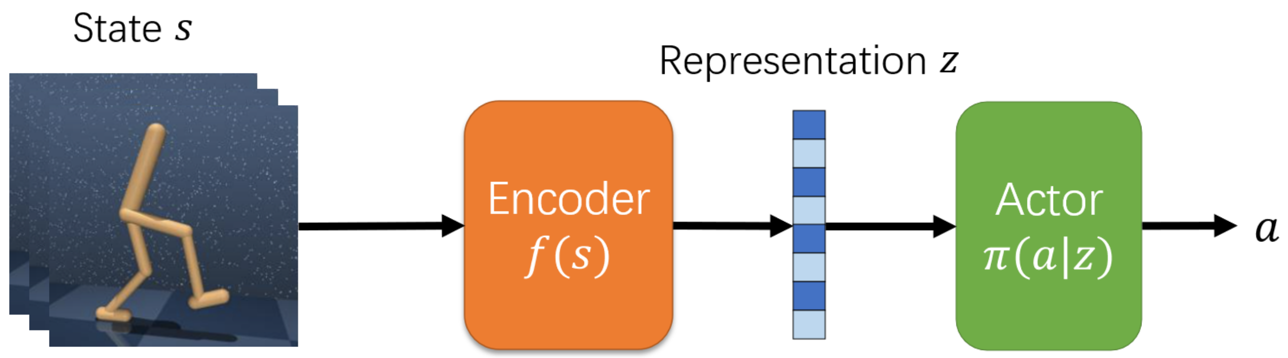


Figure 1. The pipeline of this work. In this work, we explore different ways of training f and π (including end-to-end and two-stage), and conduct analysis on the property of z.

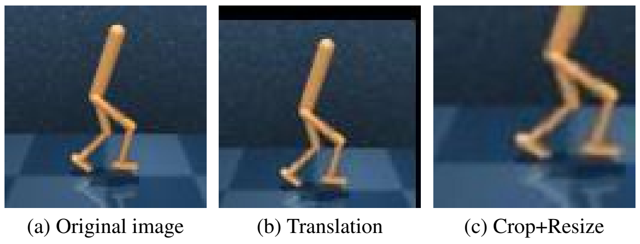


Figure 2. An illustration of the two data augmentation methods in ABC, which are translation and crop+resize.

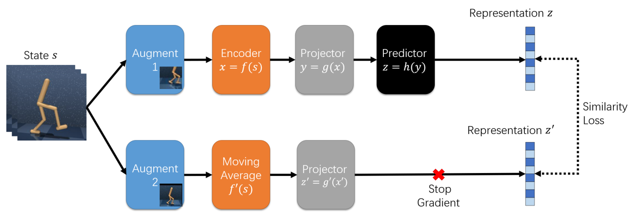


Figure 3. The architecture of BYOL. Note the output from current network goes through an extra module, which is the black predictor $z = h(y)$.

4.2. Contrastive-Learning-Based Methods

4.2.1 Contrastive Learning

In this work, we mainly consider three different contrastive learning methods for representation learning: BYOL, SIMCLR and VICREG.

BYOL. As introduced in Sec. 2, BYOL is a parallel augmentation method, which means it tries to minimize the discrepancy between embedding over the same image with different augmentations. Specifically, BYOL aims to minimize the discrepancy between the encoder and its Exponential Moving Average (EMA) [16]. Fig. 3 shows an illustration of BYOL, where encoder is a convolutional network that will be eventually used, and projectors and encoders are MLPs that will be discarded after the encoder is trained. As illustrated in the figure, BYOL takes the same state (image) s, go through different data augmentations, and feed them respectively through the current and exponential moving average network, to get representation $z = h(g(f(x)))$ and $z' = g'(f'(x))$. Then, after normalizing to unit vector, BYOL minimizes the similarity loss $L_0 + L_1$, where $L_0 = \|z - z'\|_2^2$, and L_1 is L_0 with augments exchanged between that fed into the current network and the average network to ensure symmetry.

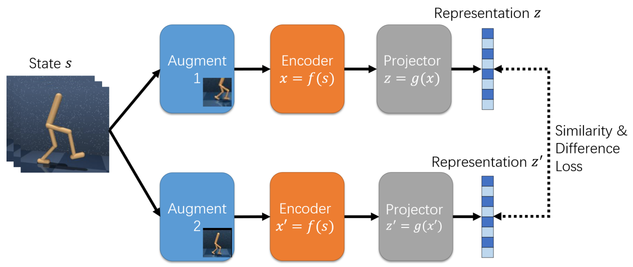


Figure 4. The architecture of SIMCLR. There are three key differences between SIMCLR and BYOL: removal of predictor, no stop-gradient operator, and different loss function.

SIMCLR. Fig. 4 shows the architecture of SIMCLR. There are three key differences between SIMCLR and BYOL: 1) SIMCLR has no predictor, and the two branches of parallel augmentation is identical, 2) SIMCLR has no stop gradient operator, and 3) the loss function is different. When training, a batch of N examples $s_1, s_2, \ldots, s_N$ are sampled from the dataset; the two branches will form two sets of representations $Z = (z_1, z_2, \ldots, z_N)$ and $Z' = (z'_1, z'_2, \ldots, z'_N)$, where for any $i \in \{1, 2, \ldots, N\}$, z_i and z'_i are representation vectors. Then, for s_i, we only consider *normalized* z_i and z'_i as positive examples, thus all the other $2(N-1)$ representations are negative samples. For state s_i with positive sample (z_i, z'_i) and negative sample (z_j, z'_j)

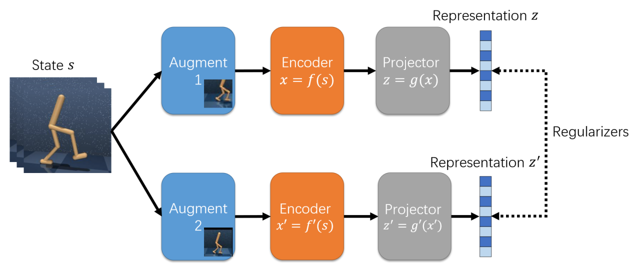


Figure 5. The architecture of VICREG. Note the two branches are with different parameters, as opposed to SIMCLR.

from $j \neq i$, the loss l_i can be written as

$$l_i = -(\log \frac{\exp(\frac{z_i^T z_j'}{\tau})}{\sum_{k=1,k\neq i}^N \exp(\frac{z_i^T z_k'}{\tau})} + \log \frac{\exp(\frac{z_i'^T z_j}{\tau})}{\sum_{k=1,k\neq i}^N \exp(\frac{z_i'^T z_k}{\tau})}), \tag{2}$$

where τ is a temperature hyperparameter. The final loss L for the batch can then be written as

$$L = \frac{1}{2N}\sum_{i=1}^N l_i. \tag{3}$$

Note the loss function considers both similarity and difference between the samples, as opposed to BYOL.

VICREG. VICREG adopts the same framework as SIMCLR, as illustrated in Fig. 5. VICREG is proposed to address the problem of representation collapse in self-supervised learning, i.e., the two branches output an identical constant vector, and aims to achieve good result with more flexible architecture design that is useful for multi-modal tasks. To address these two issues, VICREG uses three tecniques:

1. Two encoders with different parameters in the two branches;

2. Loss function composes of three regularizers. For a batch of representation $Z = (z_1, z_2, \ldots, z_n)$ and $Z' = (z_1', z_2', \ldots, z_n')$ by another branch, The first regularizer is

$$v(Z) = \frac{1}{d}\sum_{j=1}^d \max(0, 1 - \sqrt{\text{Var}(Z^j) + \epsilon}), \tag{4}$$

where representation $z \in \mathbb{R}^d$ (i.e. d is the number of dimensions), and $\text{Var}(Z^j)$ means the variance of the dimension j across the batch, and the loss encourages the variance across the batch to be larger than 1 in each dimension. The second regularizer is

$$C(Z) = \frac{1}{d}\sum_{i\neq j}[C(Z)^2]_{i,j} \tag{5}$$

,

which encourages the covariance matrix to be diagonal. The third regularizer is the mean squared error $S(Z, Z') = \frac{1}{n}\sum_{i=1}^n \|z_i - z_i'\|_2^2$. Thus, the total loss for the batch (Z, Z') is

$$c_0 S(Z, Z') + c_1(V(Z) + V(z')) + c_2(C(Z) + C(Z')), \tag{6}$$

where c_0, c_1 and c_2 are hyperparameters. In our experiment, following VICREG, we set $c_0 = c_1 = 25$, $c_2 = 1$.

4.2.2 Policy Retrieval

After the embedding module is trained, we tested two ways for the retrieval of policy: behavior cloning and nearest neighbour, where the latter is proposed by the VINN paper.

Behavior Cloning. Given a representation z and corresponding action a, we aim to optimize

$$E_{(s,a)\sim D} \log \pi(a|z), \tag{7}$$

where z is the representation of s, and the objective can be optimized by standard behavior cloning (i.e. supervised learning with z as feature and a as label). In this work, we choose to simply optimize mean squared error between the predicted action and ground-truth expert action.

Nearest Neighbour. Ideally, if the representation is good enough, the action for a particular state can be inferred from expert data with embedding in its adjacency. In this work, we test two types of distance metric: Manhattan distance $d(z, z') = \|z - z'\|_1$ and Euclidean distance $d(z, z') = \|z - z'\|_2$. For each distance, following the VINN [22] work, we test 1-nearest neighbour and soft-min local regression, which are using the action from the nearest expert state alone and weighted sum with weight proportional to $\exp(-d(z, z'))$.

5. Experiments

In this section, we evaluate our framework on a new environment different from that tested in VINN. We plan to study the following two questions: 1) Can the encoder and actor and end-to-end ABC work well on a new environment? 2) What property does the visual representation have on this new environment? We will introduce the detailed settings in Sec. 5.1 and the results in Sec. 5.2. Code can be seen in `https://github.com/Kai289371298/CS498final`.

5.1. Experiment Settings

Experimental Details. We test the performance of the framework elaborated in Sec. 4 on the dm-control

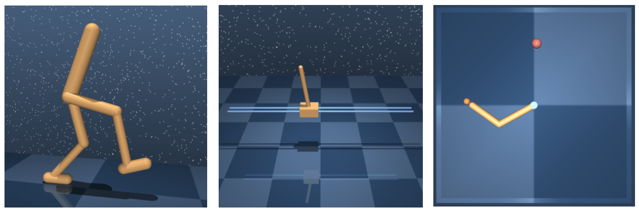
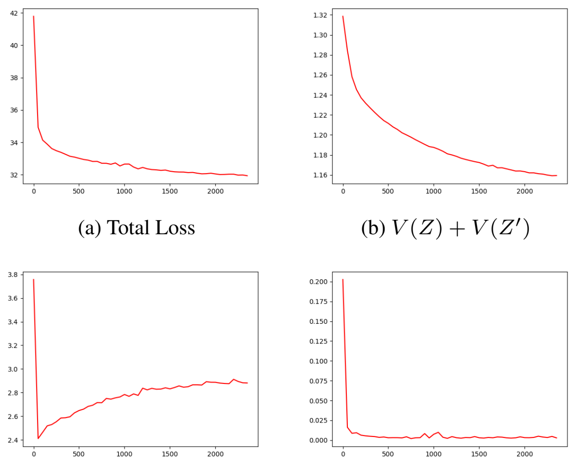

(a) Walker-Walk (b) Cartpole-Swingup (c) Reacher-Easy

Figure 6. Illustrations of the environments.

testbed [1] by OpenAI, mainly on three visual RL environments: walker-walk, cartpole-swingup and reacher-easy. The environment is illustrated in Fig. 6. In walker-walk, the agent needs to control a bipedal robot to walk forward; in cartpole-swingup, the agent controls a cart with a pole attached to it, trying to swing the pole upwards and maintain in that position; and in reacher-easy, the agent needs to control a robotic arm to touch a particular point in a 2D plane.

Implementation Details. We implement the codebase from scratch using Pytorch [23], with reference to the following codebases: https://github.com/jyopari/VINN/tree/main and https://github.com/MishaLaskin/rad. VINN uses a pretrained ResNet50 [13] for the initialization of the encoder; however, due to computational resource limit, we use ResNet18 instead. For actor, following VINN, we use a Multi-Layer Perceptron (MLP) with two hidden layers with width 1000 and 2048 respectively, and ReLU as activation function. For encoder, we finetune the encoder for 100 epoches with batch size 512; for behavior cloning (including end-to-end BC with RAD augmentation and BC with fine-tuned encoder), we train the agent for 120K timesteps. For both the encoder and actor, we use Adam [15] optimizer and learning rate of 10^{-3}. For end-to-end Augmented Behavior Cloning (ABC), the translation amount is limited to 20% of the image width/height, and for cropping, at most 8 pixels will be cropped for each dimension (i.e., for 84×84 image, the patch will be no smaller than 76×76). All hyperparameters and data augmentation for BYOL, SIMCLR and VICREG a follows the VINN [22] work.

Dataset Details. We generate the dataset by first training a Soft-Actor-Critic (SAC) [12] agent with stable-baselines3 [24], on proprioceptive state with 1M timesteps and default hyperparameter settings in stable-baselines3. SAC is the state-of-the-art online RL method, and we ensure they reach an expert-level average reward by comparing to the results reported in RAD [18]; then, we generate the 84×84 image and corresponding action of 50 expert trajectories using the trained SAC agent.


(a) Total Loss (b) $V(Z) + V(Z')$

(c) $C(Z) + C(Z')$ (d) $S(Z, Z')$

Figure 7. Loss curves of VICREG training on walker-walk environment. The x-axis is the gradient step, and the y-axis is the loss. It is worth noting that the MSE loss $S(Z, Z')$ is very close to 0, but covariance loss $C(Z) + C(Z')$ and variance loss $V(Z) + V(Z')$ are far from 0, regardless of the coefficient being small or large. More importantly, $C(Z) + C(Z')$ increases during training; such phenomenon indicates that the visual representation space is too complex to be described by by a basis independent across dimensions.

5.2. Results

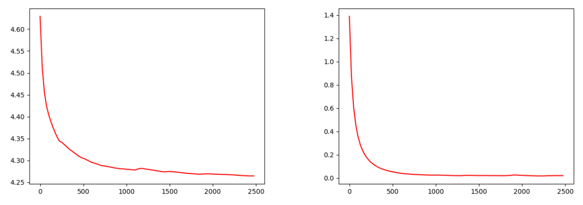
Performance Summary. Tab. 1 summarizes the final performance of each tested method; the result is an average of 3 different seeds. The result shows that most variants fail to complete the target task. Behavior cloning performs marginally better over nearest neighbour (which fails regardless of the distance metric and soft/hard max) on walker-walk and reacher-easy environment, and similar on cartpole-swingup. Such results indicates that the state representation is not good enough so as to cluster states with similar expert policy together. end-to-end ABC with augmentation seems to be better, but still with large variance.

The Loss Curves. To better analyze the property of state representation, we plot the training curves of BYOL, SIMCLR and VICREG; Fig. 7 and Fig. 8 shows the loss curve of the three embedding methods respectively. The result shows that, for VICREG and BYOL, the model extracts semantic meaning between, as the similarity loss approaches zero; however, the representation space is too complicated for VICREG to learn an ideal representation with each dimension being independent. Also, SIMCLR does not converge to low loss on the expert dataset, which indicates its failure to find a good state representation.

Fig. 9 shows the loss curve of behavior cloning over representations and end-to-end ABC with translation and

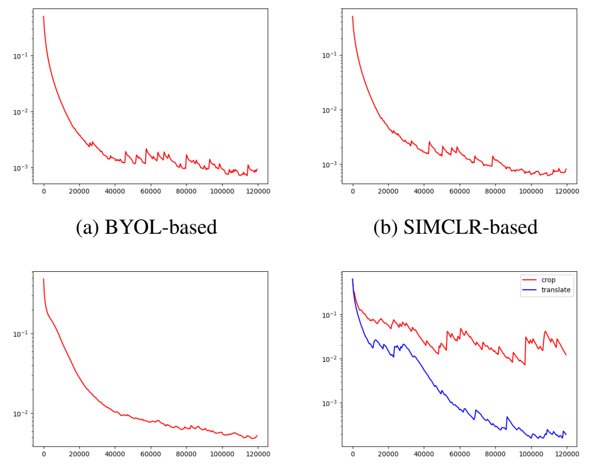
Encoder	Actor	Walker-Walk	Cartpole-Swingup	Reacher-Easy
BYOL	BC	75($\pm$61)	124($\pm$47)	67($\pm$44)
BYOL	NN-Manhattan-Soft	24($\pm$3)	100($\pm$71)	70($\pm$62)
BYOL	NN-Euclidean-Soft	29($\pm$8)	99($\pm$68)	107($\pm$57)
BYOL	NN-Manhattan-Hard	32($\pm$11)	98($\pm$70)	75($\pm$71)
BYOL	NN-Euclidean-Hard	29($\pm$9)	100($\pm$71)	120($\pm$51)
SIMCLR	BC	28($\pm$5)	102($\pm$31)	100($\pm$74)
SIMCLR	NN-Manhattan-Soft	33($\pm$11)	126($\pm$69)	57($\pm$49)
SIMCLR	NN-Euclidean-Soft	29($\pm$4)	170($\pm$11)	41($\pm$21)
SIMCLR	NN-Manhattan-Hard	38($\pm$11)	126($\pm$68)	90($\pm$29)
SIMCLR	NN-Euclidean-Hard	29($\pm$9)	100($\pm$71)	120($\pm$51)
VICREG	BC	28($\pm$2)	164($\pm$30)	127($\pm$78)
VICREG	NN-Manhattan-Soft	29($\pm$2)	166($\pm$8)	123($\pm$68)
VICREG	NN-Euclidean-Soft	28($\pm$8)	21($\pm$20)	108($\pm$89)
VICREG	NN-Manhattan-Hard	28($\pm$4)	164($\pm$7)	120($\pm$83)
VICREG	NN-Euclidean-Hard	29($\pm$6)	100($\pm$62)	124($\pm$109)
End-to-End	Augmented-BC-Crop	115($\pm$31)	75($\pm$1)	407($\pm$317)
End-to-End	Augmented-BC-Translate	169($\pm$55)	119($\pm$79)	180($\pm$235)
Proprioceptive	SAC	934($\pm$47)	808($\pm$15)	937($\pm$18)

Table 1. The mean reward and the corresponding standard deviation for each methods. Actor has two types: behavior cloning (BC), or nearest neighbour (NN) with two different distance metrics (Manhattan, Euclidean) and two paragidms (softmax, hardmax). Proprioceptive SAC is the expert agent that generates the dataset.



(a) SIMCLR (b) BYOL

Figure 8. Loss curves for SIMCLR and BYOL; The x-axis is the gradient step, and the y-axis is the loss. BYOL reaches a loss close to 0, which indicates the algorithm fits well on the dataset, while SIMCLR with a high final loss does not.



(a) BYOL-based (b) SIMCLR-based

(c) VICREG-based (d) End-to-End Augmentation

Figure 9. Loss curves of behavior cloning training on walker-walk environment. The x-axis is the gradient step, and the y-axis is the loss.

cropping as augmentation for the walker-walk environment. While for all methods but ABC-crop tested in this work, BC loss is low (< 0.005), lower behavior cloning loss does not necessarily mean better performance. This indicates that there are other factors, such as generalizability, that is important to the visual imitation performance.

Is Adjacent Frame Close in the Representation Space, and How Does It change Along the Trajectory? Ideally, a good state representation should reveal the inner pattern of expert policy with adjacent steps being close to each other, and a good policy should roughly follow the trajectory in the state representation space. Fig. 10 shows the visualization result of representation and pixel euclidean distance between the adjacent states (on the left for each subfigure) and t-SNE [33] visualization of actor trajectory and expert trajectories in the representation space for each method tested (on the right for each subfigure) in the reacher-easy environment. The result shows that: 1) representation distance is almost proportional to the pixel distance for SIMCLR, BYOL and VICREG, and is high at the beginning of the episode. This indicates that the representations do not extract semantics that connects the first few steps together at the beginning of the episode as expected. ABC-Crop and ABC-Translate has lower embedding distances, which corresponds to better performance on the reacher-easy environment; 2) The representation of expert trajectory is mostly clustered, but far from the actor representation. This explains the reason why the behavior cloning fails: most representations seen in evaluation are out-of-distribution, which indicates that the generalizability of the representation is limited. However, ABC-Crop and ABC-Translate exhibits some extent of low dimensionality in the representation space, which indicates better grasping of semantics.

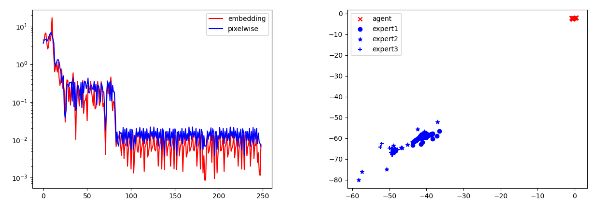
How Close is the Nearest Neighbour in the Representation Space? In VINN [22], the researchers find that nearest neighbour in the representation space is suffice for a good policy. Ideally, such neighbour should be close to the state witnessed in the expert dataset in the representation space. Fig. 11 shows the distance of the nearest neighbour in the expert dataset in an evaluation trajectory in the cartpole-swingup environment (on the left), and also illustrates the coefficients for the top $1, 5$ and 10 neighbour(s) when using exponential of Euclidean distance (on the right). All six figures shown in Fig. 11 exhibit a periodical pattern, which corresponds to the essence of the cartpole environment where the pole on the car is rotating back and forth. The results indicates that although the nearest neighbour is close to the agent, the expert state representation is uniformly distributed in the representation space such that there do not exist a small set of action that dominates the generation of agent action, which is a reason for failure.

6. Conclusions and Discussion

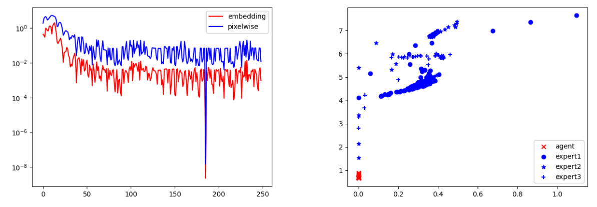
In this report, we reproduced the VINN [22] and augmentation in RAD [18] from scratch, and test them on the dm-control testbed [1]. With experiments, we found the following facts:

1. The framework proven to be success in other task does not seem to work well on the dm-control testbed. Specifically, nearest neighbour does not seem to work as well as behavior cloning, and state representation with contrastive learning methods such as BYOL, SIMCLR and VICREG does not seem to work well as end-to-end behavior cloning with image cropping and translation.

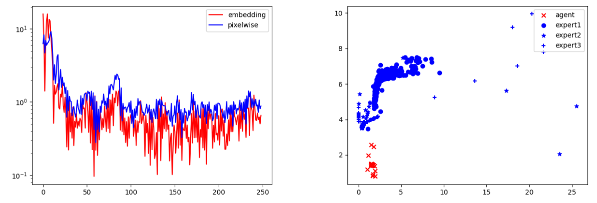
2. Contrastive learning methods do not seem to learn a good embedding that clusters states with similar expert policy together as expected. While BYOL has low similarity loss, SIMCLR and VICREG both fails to optimize to a good local optimal. The generalizability is limited, which causes out-of-distribution state representations in test time, and thus the reward performance is limited.



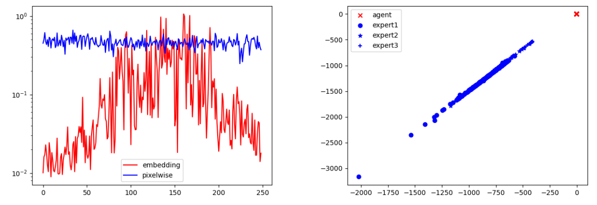
(a) BYOL



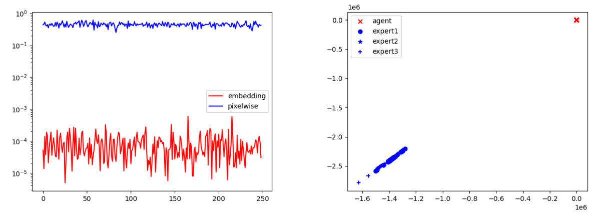
(b) SIMCLR



(c) VICREG

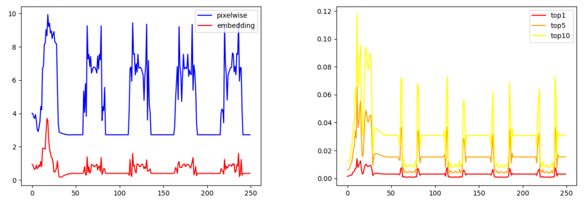


(d) ABC-Crop

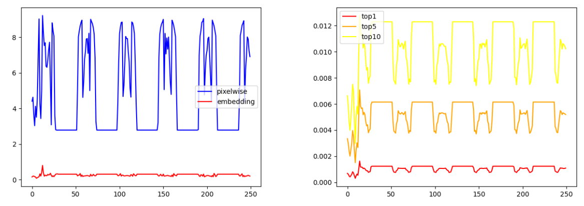


(e) ABC-Translate

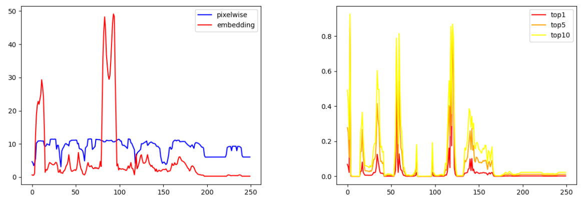
Figure 10. On the left is the representation (red) and pixel (blue) distance between the adjacent states on the left, where the x-axis is the number of steps and the y-axis is the distance. On the right is the t-SNE visualization of actor trajectory (red) and expert trajectories (blue, distinguished with different markers) in the representation space. Both figures are tested in the reacher-easy environment.



(a) BYOL



(b) SIMCLR



(c) VICREG

Figure 11. On the left is the distance in the representation space (red) and pixel space (blue) to the nearest neighbour along an evaluation trajectory, where the x-axis is the number of steps and the y-axis is the distance; on the right is the coefficient with Euclidean distance softmax for top-1 (red), -5 (orange) and -10 (yellow) components (the component for the whole dataset sums up to 1), where the x-axis is the number of steps and the y-axis is the coefficient.

3. Nearest neighbour in VINN [22] fails on the dm-control [1] testbed probably because the distribution of expert state in the representation space is too uniform; there is no recognized expert action that dominates the generation of agent action.

However, due to time and resource limit, such work is limited to a few method and testbeds; thus, the most straightforward future direction would be expanding the analysis onto more methods given more time. A more promising future direction, however, comes from outside the visual imitation area; recently, there is a popular trend in using generative model and transformers for offline imitation or reinforcement learning, such as decision transformer [7] and decision diffuser [2]; such model exhibits better robustness and generalizability on multiple testbeds than traditional RL/IL methods. Thus, future direction should be focused on analyzing latent space of such models, which could help us to figure out the "semantics" and "attention" pattern of a trajectory in MDP similar to that of natural language.

References

[1] dm_control: Software and tasks for continuous control. *Software Impacts*, 2020. 1, 5, 7, 8

[2] Anurag Ajay, Yilun Du, Abhi Gupta, Joshua B. Tenenbaum, Tommi S. Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision making? In *The Eleventh International Conference on Learning Representations*, 2023. 8

[3] Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. *arXiv preprint arXiv:2105.04906*, 2021. 2

[4] Matthew Chang, Arjun Gupta, and Saurabh Gupta. Semantic visual navigation by watching youtube videos. In *NeurIPS*, 2020. 1

[5] Matthew Chang, Arjun Gupta, and Saurabh Gupta. Semantic visual navigation by watching youtube videos. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc. 1

[6] Annie S. Chen, Suraj Nair, and Chelsea Finn. Learning generalizable robotic reward functions from "in-the-wild" human videos. *ArXiv:2103.16817*, 2021. 2

[7] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021. 8

[8] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020. 2

[9] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 539–546 vol. 1, 2005. 2

[10] Benjamin Eysenbach, Sergey Levine, and Ruslan Salakhutdinov. Replacing rewards with examples: Example-based policy search via recursive classification. In *NeurIPS*, 2021. 1

[11] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent a new approach to self-supervised learning. In *NeurIPS*, 2020. 2

[12] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018. 5

[13] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2015. 5

[14] Phúc H. Lê Khac, Graham Healy, and Alan F. Smeaton. Contrastive representation learning: A framework and review. *IEEE Access*, 8:193907–193934, 2020. 2

[15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 5

[16] Frank Klinker. Exponential moving average versus moving exponential average. *Mathematische Semesterberichte*, 58:97–107, 2011. 3

[17] Ashish Kumar, Saurabh Gupta, and Jitendra Malik. Learning navigation subroutines from egocentric videos. In *CoRL*, 2019. 1, 2

[18] Michael Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. In *NeurIPS*, 2020. 1, 2, 5, 7

[19] Youngwoon Lee, Andrew Szot, Shao-Hua Sun, and Joseph J. Lim. Generalizable imitation learning from observation via inferring goal proximity. In *NeurIPS*, 2021. 2

[20] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4004–4012, 2016. 2

[21] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. 2

[22] Jyothish Pari, Nur Muhammad Shafiullah, Sridhar Pandian Arunachalam, and Lerrel Pinto. The surprising effectiveness of representation learning for visual imitation. In *RSS*, 2022. 1, 2, 4, 5, 7, 8

[23] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*. 2019. 5

[24] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. 5

[25] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 815–823, 2015. 2

[26] Pierre Sermanet, Corey Lynch, Jasmine Hsu, and Sergey Levine. Time-contrastive networks: Self-supervised learning from multi-view observation. *ArXiv:1704.06888*, 2017. 2

[27] Pratyusha Sharma, Deepak Pathak, and Abhinav Gupta. Third-person visual imitation learning via decoupled hierarchical controller. In *NeurIPS*, 2019. 1

[28] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019. 2

[29] Laura Smith, Nikita Dhawan, Marvin Zhang, P. Abbeel, and Sergey Levine. Avid: Learning multi-stage tasks via pixel-level translation of human videos. 2020. 1, 2

[30] A. Srinivas, Michael Laskin, and P. Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *ICML*, 2020. 1, 2

[31] Bradly C Stadie, Pieter Abbeel, and Ilya Sutskever. Third-person imitation learning. *ICLR*, 2017. 2

[32] Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. In *IJCAI*, 2018. 2

[33] Laurens van der Maaten and Geoffrey E. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008. 6

[34] David A Van Dyk and Xiao-Li Meng. The art of data augmentation. *Journal of Computational and Graphical Statistics*, 10(1):1–50, 2001. 2

[35] Lilian Weng. Contrastive representation learning. *lilianweng.github.io*, May 2021. 2

[36] Alan Wu, AJ Piergiovanni, and Michael S. Ryoo. Model-based behavioral cloning with future image similarity learning. In *CoRL*, 2019. 2

[37] Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning. *arXiv preprint arXiv:2107.09645*, 2021. 1, 2

[38] Denis Yarats, Ilya Kostrikov, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *International Conference on Learning Representations*, 2021. 1, 2

[39] Kevin Zakka, Andy Zeng, Pete Florence, Jonathan Tompson, Jeannette Bohg, and Debidatta Dwibedi. Xirl: Cross-embodiment inverse reinforcement learning. In *CoRL*, 2021. 1, 2